High-Performance dsPIC33A Core with Floating-Point Unit, High Resolution PWM, High-Speed ADCs, CAN FD, I3C, Resolver Interface, Security Features
dsPIC33AK256MPS306 Family Data Sheet

Operating Conditions

- 3.0V to 3.6V: -40°C to +85°C, DC to 200MHz
- 3.0V to 3.6V: -40°C to +125°C, DC to 200 MHz Planned
- 3.0V to 3.6V: -40°C to +150°C, DC to 200 MHz Planned

High-Performance DSP CPU

- 32-bit Rich Instruction Set for Optimized Speed and Program Code Size:
  - 16-bit dsPIC33 core compatible
  - Non-paged linear data/Flash 24-bit addressing space
  - 16-bit/32-bit instructions for optimized code size and performance
- 32-Bit Wide Data Paths
- Single and Double Precision Floating-Point Unit (FPU) Coprocessor
- 2-Kbyte Instruction Cache
- Sixteen 32-Bit Working Registers
- Dual 72-Bit Accumulators Supporting 32-Bit and 16-Bit Fixed-Point DSP Operations
- Eight Level Deep Working Register Sets
- Eight Level Deep Accumulator Register Sets
- Eight Level Deep Floating-Point Register Sets

Memory Features

- Up to 256 Kbytes of Program Flash Memory:
  - 10,000 erase/write cycle endurance
  - 20 years minimum data retention
  - Self-programmable under software control
  - Programmable code protection
  - Flash Error Correcting Code (ECC)
  - Dual Flash panel
  - Live update support
  - Programmable OTP regions
  - Entire Flash OTP by ICSP™ write inhibit
  - Separate 64x128-bit OTP
- Up to 64 Kbytes of Static RAM Memory:
  - RAM Error Correcting Code (ECC)
  - RAM Memory Built-In Self-Test (MBIST)

Security Features

- Flash OTP by ICSP Write Inhibit
  - On-Chip Secure Boot Flash Configurable as an Immutable Root of Trust
    - Parts of the Flash memory can be configured as OTP
  - Capabilities include:
    - Secure boot support: Validation of host code image and host code signature
    - Secure update support for host code: Secure encryption key storage and image decryption
    - X.509 certification storage, parsing, validation and revocation, supporting both ECC and RSA
  - 128-bit Unique Device Serial Number for Identification (UUID)
  - Support for Secure Use Cases:
    - Secure boot
    - Key Storage in IRT/Immutable secure boot region for realizing:
      - secure boot
      - secure firmware update
      - secure debug
  - Flash Protection
    - Configuration of up to eight Flash protection regions across ranges of Flash addresses
    - Regions can be configured as:
      - Immutable Root of Trust (IRT)
      - OTP region
      - A combination of R/W/X protections.
    - Regions can be:
      - Made permanent
      - Locked until device reset
      - Enabled/disabled during code execution
    - Flash protection regions can apply to the active partition, the inactive partition, or both
- Crypto Accelerator Module (CAM)
  - AES-128, AES-192, and AES-256: Fully Compliant with NIST FIPS 197
    - ECB, CBC, CFB, OFB, CTR, GCM, CCM, XTS, CMAC modes
  - HASH/MAC
    - SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 and SHAKE256 capability
    - SHA-1, SHA-256, SHA-224, SHA-384 and SHA-512 capability
  - Public Key Cryptography: RSA, DSA, and ECC
    - RSA with/without Chinese Remainder Theorem (CRT). Up to 4096-bit key length:
      - Prime Field P-192, P-224, P-256, P-384, P-521
      - Binary Field K-163, K-233, K-283, K-409, K-571
      - Binary Field B-163, B-233, B-283, B-409, B-571
      - P-224, P-256, P-384, and P-521 Elliptic Curve – ECDSA Sign/Verify
    - DSA support for up to 2048-bit key length
    - ECDH support for P256 and P224 Curves
    - SECP256K1 (Bitcoin/Blockchain curve) ECDSA support

    - 256-bit Brainpool Elliptic Curve support – ECDSA, ECDH
    - Elliptic Curve Diffie Hellman (ECDH/ECDHE) Key Agreement
  - NIST-800-22 and NIST-800-90B Compliant True Random Number Generator (TRNG)

High-Resolution PWM

- Multiple PWM Generators:
  - Four pairs (eight output) generators, with Fine Edge Placement (FEP) resolution down to 78 ps
- Dead Time for Rising and Falling Edges
- Dead-Time Compensation Supports Lower Speed Operation
- Clock Chopping for High-Frequency Operation
- PWM Support for:
  - BLDC, PMSM, ACIM, SRM and stepper motors
  - Constant on-time, hysteretic, burst mode power applications
- Fault and Current Limit Inputs
- Flexible Trigger Configuration for ADC Triggering

High-Speed Analog-to-Digital Converters

- Three 12-bit Resolution SAR ADCs
- Up to 40 Msps Conversion Rate per ADC
- Up to 14 Analog Input Pins
- 16-bit Sampling Capability
- Sixteen Settings Channels. Each Channel:
  - Can be assigned to any analog input (I/O pin or internal signal)
  - Can be set to a different sampling time
  - Can be configured as single-ended or differential
  - Conversion result can be formatted as unsigned or signed
  - Conversion result can be left-aligned (fraction format)
  - Has a separate 32-bit conversion result register
- Supports Four Sampling Modes:
  - Oversampling of multiple samples
  - Integration of multiple samples
  - Window (multiple samples accumulated when the gate signal is active)
  - Single conversion
  - All channels have a digital comparator to detect configurable thresholds
  - The last three setting channels have the second result accumulator to implement second-order filters

Peripheral Features

- Three 4-Wire SPI Modules (up to 50 Mbps):
  - 32-byte FIFO
  - Variable data width
  - I^2S mode

  - Multi-channel digital audio interfaces (I^2S)
  - Three 16-bit resolution capable audio-rate ADCs at 156 kSPS
  - DMA-enabled real-time audio streaming (8 DMA channels available)
  - TDM4/8/16/32 capable using Framed SPI mode with DMA peripheral
  - Audio DSP for voice and acoustic processing
- Two I^2C Modules w/ Address Masking and IPMI Support
- One I^3C Module w/ Primary and Secondary Controller modes, and I^3C/I^2C Target Capability
- Four Protocol UARTs with 8-Character RX/TX FIFOs and Automated Handling Support for:
  - LIN 2.2
  - DMX
  - Smart card (ISO 7816)
  - IrDA®
- Two SENT Modules
- Three Dedicated 32-Bit Timer/Counter Modules
- Output Capture/Compare/PWM/Timer Modules:
  - One MCCP
  - Four SCCPs
  - Flexible configuration as PWM, input capture, output compare or timers
  - Two 16-bit timers or one 32-bit timer in each module
- 8-Channel Hardware Direct Memory Access (DMA) module
- One Quadrature Encoder Interface (QEI) Module:
  - Four inputs: Phase A, Phase B, Home, Index
- Serial Encoder Interface BiSS with up to Four Client Encoders Support
- Four Configurable Logic Cells (CLC) with Internal Connections to Select Peripherals and PPS
- Peripheral Trigger Generator (PTG):
  - 16 possible trigger sources to other peripheral modules
  - CPU independent state machine-based instruction sequencer
- One CAN FD Module
- Integrated Touch Controller (ITC) Module
  - Advanced Capacitive Sensing, Touch Buttons, Sliders and Wheels
  - Up to 24 Self-Capacitance and up to 72 Mutual-Capacitance Channels
- One Resolver-to-Digital Converter (RDC) Module

Controller Features

- High-Current I/O Sink/Source
- Programmable Weak Pull-Up and Pull-Down Resistors
- Programmable Open-Drain Outputs
- Edge or Level Change Notification Interrupt on I/O pins
- Peripheral Pin Select (PPS) Remappable Pins to Reduce Board Layout Complexity
- Multiple Interrupt Vectors with Individual Programmable Priority
- Five External Interrupt Pins

- Selectable Oscillator Options, Including:
  - 8 MHz, 1% at 0ºC-85ºC Internal Fast RC (FRC) oscillator
  - 8 MHz, 2% Internal Backup Fast RC (BFRC) oscillator with 32 kHz divided output
  - High-speed crystal resonator oscillator or external clock
  - Two 1.6 GHz PLLs which can be clocked from the FRC or a crystal oscillator
  - Reference Clock Output (REFO)
- Low-Power Management Modes (Sleep and Idle)
- Power-on Reset and Brown-out Reset
- On-Board 1.1V Buck Voltage Regulator for Core Voltage Supply

Analog Features

- Up to Five 5 nS Analog Comparators with 12-Bit PDM DACs:
  - Input multiplexing
  - Slope compensation
  - Up to two DAC output buffers
- Three Rail-to-Rail 100 MHz Operational Amplifiers with:
  - 40 V/µS slew rate
  - 1 mV offset (typical) with calibration feature
- $V_{REF/2}$ Output Available for Op Amp Input Biasing
- Four 10 µA Constant-Current Sources, Four Programmable Current Sources
- Integrated Touch Controller:
  - Independent from ADC processing capability
  - Self and Mutual CVD Touch Modes support
  - 11 to 24 Touch ADC RX inputs
  - 9 to 15 Touch TX outputs

Safety Features

- Windowed Watchdog Timer (WDT)
- Deadman Timer (DMT)
- Eight I/O Integrity Monitors (IOIM)
- Fail-Safe Clock Monitor (FSCM) with Automatic Switchover to Backup Clock Source, featuring:
  - • Programmable over-frequency/under-frequency thresholds
- Flash Error Correcting Code (ECC)
- RAM Error Correcting Code (ECC)
- RAM Memory Built-In Self-Test (MBIST)
- 32-Bit Cyclic Redundancy Check (CRC) Module
- Entire Flash OTP by ICSP Write Inhibit
- Internal Voltage Regulators
- Virtual PPS Pins for Redundancy and Monitoring
- Temperature Sensor Diode
- Power Monitor for Core Voltage with Configurable Fault Injection

Functional Safety

- Targets:
  - ISO 26262 ASIL B
  - IEC 61508 SIL 2
  - IEC 60730 Class B
- ISO 26262 and IEC 61508 Compliant Device Development

To learn more about various functional safety standards and target safety levels supported by this device family supports, visit www.microchip.com/en-us/products/microcontrollers-and-microprocessors/dspic-dscs/functional-safety.

Qualification

- AEC-Q100 REV G (Grade 3: -40°C to +85°C) Compliant
- AEC-Q100 REV G (Grade 1: -40°C to +125°C) Planned
- AEC-Q100 REV G (Grade 0: -40°C to +150°C) Planned
- ISO 26262 ASIL B, IEC 61508 SIL 2, IEC 60730 Class B
- ISO 26262 and IEC-61508 Compliant Development following the Certified Functional Safety Management System

Programming and Debug Interfaces

- Three Programming and Debugging Interfaces:
  - Two-wire ICSP™ interface with non-intrusive access and real-time data exchange with application
- Five Complex and Five Simple Breakpoints
- IEEE Standard 1149.2 Compatible (JTAG) Boundary Scan
- Hardware Instruction Trace

dsPIC33AK256MPS306 Family Features

The device names, pin counts, memory sizes and peripheral availability of each device are listed in Table 1. The following pages show their pinout diagrams.

Table 1. dsPIC33AK256MPS306 Family Device Features

Product	Pins	Flash (Kbytes)	SRAM (Kbytes)	GPIO/PPS	Crypto Accelerator Module (CAM)	Remappable Peripherals											ADC Modules (12-bit)	ADC (External Inputs)	Touch ADC Inputs	12-bit DACs	DAC Outputs	Analog Comparators	Resolver-to-Digital Converter	Op Amp	DMA (Channels)	PWM Pairs	I^2C/I^3C	UREF	PTG	CRC	DMT	Packages
						BISS	CLC	Dedicated 32-Bit Timers	SCCP/MCCP	SPI	UART	QEI	CAN FD	REFO	SENT	I/O Integrity Monitors																
128K/256K Devices with CAN FD																																
dsPIC33AK128MPS303	36	128K	64K	21/21	Y	1	4	3	4/1	3	4	1	1	2	2	8	3	11	11	5	1	5	1	2	8	4	2/1	1	1	1	1	VQFN
dsPIC33AK128MPS305	48	128K	64K	33/33	Y	1	4	3	4/1	3	4	1	1	2	2	8	3	14	20	5	1	5	1	3	8	4	2/1	1	1	1	1	TQFP/ VQFN
dsPIC33AK128MPS306	64	128K	64K	47/47	Y	1	4	3	4/1	3	4	1	1	2	2	8	3	14	24	5	1	5	1	3	8	4	2/1	1	1	1	1	TQFP/ VQFN
dsPIC33AK256MPS303	36	256K	64K	21/21	Y	1	4	3	4/1	3	4	1	1	2	2	8	3	11	11	5	1	5	1	2	8	4	2/1	1	1	1	1	VQFN
dsPIC33AK256MPS305	48	256K	64K	33/33	Y	1	4	3	4/1	3	4	1	1	2	2	8	3	14	20	5	1	5	1	3	8	4	2/1	1	1	1	1	TQFP/ VQFN
dsPIC33AK256MPS306	64	256K	64K	47/47	Y	1	4	3	4/1	3	4	1	1	2	2	8	3	14	24	5	1	5	1	3	8	4	2/1	1	1	1	1	TQFP/ VQFN
128K/256K Devices without CAN FD																																
dsPIC33AK128MPS103	36	128K	64K	21/21	Y	1	4	3	4/1	3	4	1	0	2	2	8	3	11	11	5	1	5	1	2	8	4	2/1	1	1	1	1	VQFN
dsPIC33AK128MPS105	48	128K	64K	33/33	Y	1	4	3	4/1	3	4	1	0	2	2	8	3	14	20	5	1	5	1	3	8	4	2/1	1	1	1	1	TQFP/ VQFN
dsPIC33AK128MPS106	64	128K	64K	47/47	Y	1	4	3	4/1	3	4	1	0	2	2	8	3	14	24	5	1	5	1	3	8	4	2/1	1	1	1	1	TQFP/ VQFN
dsPIC33AK256MPS103	36	256K	64K	21/21	Y	1	4	3	4/1	3	4	1	0	2	2	8	3	11	11	5	1	5	1	2	8	4	2/1	1	1	1	1	VQFN
dsPIC33AK256MPS105	48	256K	64K	33/33	Y	1	4	3	4/1	3	4	1	0	2	2	8	3	14	20	5	1	5	1	3	8	4	2/1	1	1	1	1	TQFP/ VQFN
dsPIC33AK256MPS106	64	256K	64K	47/47	Y	1	4	3	4/1	3	4	1	0	2	2	8	3	14	24	5	1	5	1	3	8	4	2/1	1	1	1	1	TQFP/ VQFN

Pin Diagrams

36-Pin VQFN

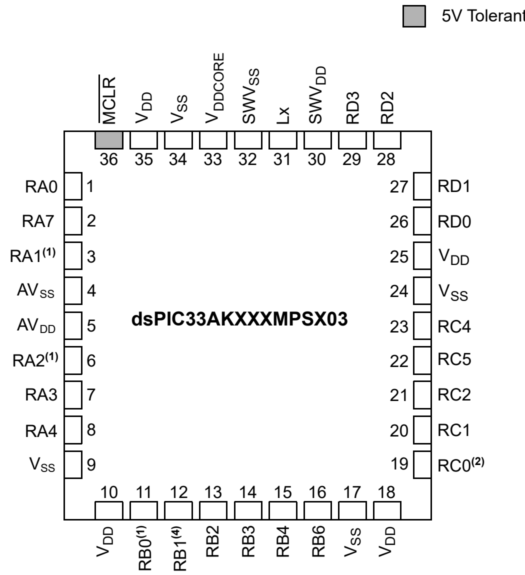


Table 2. 36-Pin VQFN Complete Pin Function Descriptions(1)

Pin	Function	Pin	Function
1	PGD2/AD1ANN1/AD1AN4/CVDAN0/**RP1**/SCL2/IOMAF2/RA0	19	OSCO/CLKO/**RP33**/IOMAF5/RC0
2	AD3AN1/CVDAN7/**RP8**/SDA2/IOMAF1/RA7	20	OSCI/**RP34**/IOMAF6/RC1
3	PGC2/DACOUT1/AD3ANN1/AD3AN0/CVDAN1/CMP4D/**RP2**/RA1	21	PGC3/**RP35**/PWM4H/IOMAD7/RC2
4	AV_{SS}	22	PGD3/**RP38**/I3CSCL1/PWM4L/IOMAD6/RC5
5	AV_{DD}	23	**RP37**/I3CSDA1/IOMAD4/RC4
6	OA1OUT/AD1AN0/CVDAN2/CMP1A/**RP3**/RA2	24	V_{SS}
7	OA1IN-/AD1ANN2/AD1AN2/CVDAN3/CMPCN/CMP1C/**RP4**/RA3	25	V_{DD}
8	UREF/OA1IN+/AD1AN1/AD2AN5/AD3AN4/CVDAN4/CMP1B/**RP5**/INT0/RA4	26	**RP49**/PWM2H/IOMAD3/RD0
9	V_{SS}	27	TCK/**RP50**/PWM2L/IOMAD2/RD1
10	V_{DD}	28	TDO/**RP51**/PWM1H/IOMAD1/RD2
11	OA2OUT/AD2AN0/CVDAN16/CVDTX0/CMP2A/**RP17**/RB0	29	TDI/**RP52**/PWM1L/IOMAD0/RD3
12	TMS/OA2IN-/AD2ANN2/AD2AN2/CVDAN17/CVDTX1/CMP2C/**RP18**/RB1	30	SWV_{DD}
13	OA2IN+/AD2AN1/CVDAN18/CVDTX2/CMP2B/**RP19**/RB2	31	Lx
14	PGD1/AD2ANN1/AD2AN3/CVDAN19/CVDTX3/CMP4A/**RP20**/SDA1/RB3	32	SWV_{SS}
15	PGC1/AD2AN4/CVDAN20/CVDTX4/CMP4B/**RP21**/SCL1/RB4	33	V_{DDCORE}
16	AD3ANN2/AD3AN2/CVDAN22/CVDTX6/ALLCMP/IBIAS0/ISRC0/**RP23**/RB6	34	V_{SS}
17	V_{SS}	35	V_{DD}
18	V_{DD}	36	$\overline{MCLR}$

Notes:

1. **RPn** represents remappable peripheral functions.
2. This pin has 8x drive strength.
3. Unless otherwise stated, pins are 4x drive strength.
4. A pull-up resistor is connected to this pin when the device is erased (JTAG enabled) and during programming.

48-Pin VQFN, TQFP

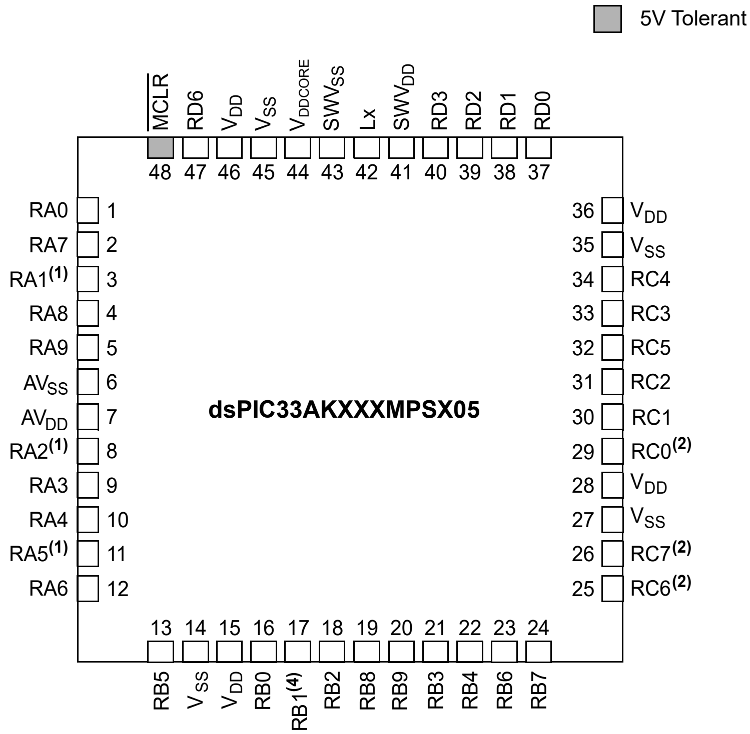


Table 3. 48-Pin VQFN, TQFP Complete Pin Function Descriptions(1)

Pin	Function	Pin	Function
1	PGD2/AD1ANN1/AD1AN4/CVDAN0/**RP1**/SCL2/IOMAF2/RA0	25	**RP39**/ASCL1/RC6
2	AD3AN1/CVDAN7/**RP8**/SDA2/IOMAF1/RA7	26	**RP40**/ASDA1/IOMAF7/RC7
3	PGC2/DACOUT1/AD3ANN1/AD3AN0/CVDAN1/ CMP4D/**RP2**/RA1	27	V_{SS}
4	AD3AN3/CVDAN8/CMP5A/IBIAS3/ISRC3/**RP9**/RA8	28	V_{DD}
5	AD1AN3/CVDAN9/CMP5B/IBIAS2/ISRC2/**RP10**/RA9	29	OSCO/CLKO/**RP33**/IOMAF5/RC0
6	AV_{SS}	30	OSCI/**RP34**/IOMAF6/RC1
7	AV_{DD}	31	PGC3/**RP35**/PWM4H/IOMAD7/RC2
8	OA1OUT/AD1AN0/CVDAN2/CMP1A/**RP3**/RA2	32	PGD3/**RP38**/I3CSCL1/PWM4L/IOMAD6/RC5
9	OA1IN-/AD1ANN2/AD1AN2/CVDAN3/CMPCN/ CMP1C/**RP4**/RA3	33	**RP36**/PWM3H/IOMAD5/RC3
10	UREF/OA1IN+/AD1AN1/AD2AN5/AD3AN4/CVDAN4/ CMP1B/**RP5**/RA4	34	**RP37**/I3CSDA1/PWM3L/IOMAD4/RC4
11	OA3OUT/CVDAN5/CMP3A/**RP6**/INT0/RA5	35	V_{SS}
12	OA3IN-/CVDAN6/CMPDN/CMP3C/**RP7**/RA6	36	V_{DD}
13	OA3IN+/CVDAN21/CVDTX5/CMP3B/**RP22**/RB5	37	**RP49**/PWM2H/IOMAD3/RD0
14	V_{SS}	38	TCK/**RP50**/PWM2L/IOMAD2/RD1
15	V_{DD}	39	TDO/**RP51**/PWM1H/IOMAD1/RD2
16	OA2OUT/AD2AN0/CVDAN16/CVDTX0/CMP2A/**RP17**/RB0	40	TDI/**RP52**/PWM1L/IOMAD0/RD3
17	TMS/OA2IN-/AD2ANN2/AD2AN2/CVDAN17/CVDTX1/ CMP2C/**RP18**/RB1	41	SWV_{DD}
18	OA2IN+/AD2AN1/CVDAN18/CVDTX2/CMP2B/**RP19**/RB2	42	Lx
19	CVDAN24/CVDTX8/CMP2D/RP25/RB8	43	SWV_{SS}
20	CVDAN25/CVDTX9/**RP26**/RB9	44	V_{DDCORE}
21	PGD1/AD2ANN1/AD2AN3/CVDAN19/CVDTX3/CMP4A/ **RP20**/SDA1/RB3	45	V_{SS}
22	PGC1/AD2AN4/CVDAN20/CVDTX4/CMP4B/**RP21**/SCL1/RB4	46	V_{DD}

Table 3. 48-Pin VQFN, TQFP Complete Pin Function Descriptions[1] (continued)

Pin	Function	Pin	Function
23	AD3ANN2/AD3AN2/CVDAN22/CVDTX6/ALLCMP/IBIAS0/ ISRC0/**RP23**/I3CASCL1/RB6	47	**RP55**/RD6
24	CVDAN23/CVDTX7/CMP4C/IBIAS1/ISRC1/**RP24**/I3CASDA1/ IOMAF0/RB7	48	$\overline{\text{MCLR}}$

Note:

1. **RPn** represents remappable peripheral functions.
2. This pin has a 8x drive strength.
3. Unless otherwise stated, pins are 4x drive strength.
4. A pull-up resistor is connected to this pin when the device is erased (JTAG enabled) and during programming.

64-Pin VQFN, TQFP

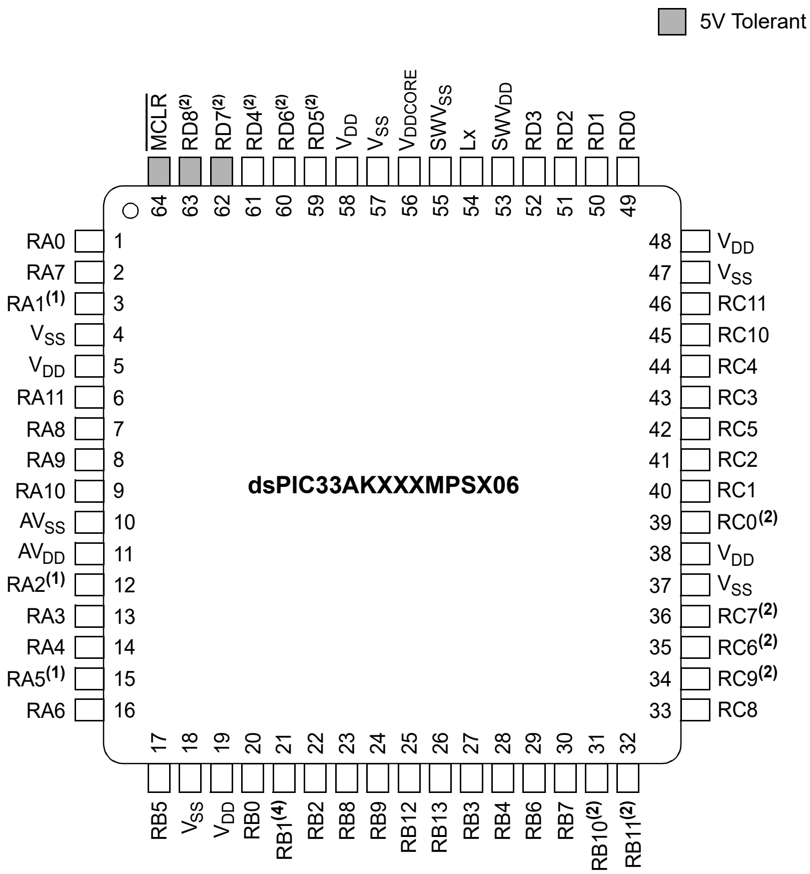


Table 4. 64-Pin VQFN, TQFP Complete Pin Function Descriptions(1)

Pin	Function	Pin	Function
1	PGD2/AD1ANN1/AD1AN4/CVDAN0/**RP1**/SCL2/IOMAF2/RA0	33	**RP41**/RC8
2	AD3AN1/CVDAN7/**RP8**/SDA2/IOMAF1/RA7	34	**RP42**/SDO2/RC9
3	PGC2/DACOUT1/AD3ANN1/AD3AN0/CVDAN1/ CMP4D/**RP2**/RA1	35	**RP39**/ASCL1/RC6
4	V_{SS}	36	**RP40**/ASDA1/IOMAF7/RC7
5	V_{DD}	37	V_{SS}
6	CVDAN11/CMPFN/**RP12**/RA11	38	V_{DD}
7	AD3AN3/CVDAN8/CMP5A/IBIAS3/ISRC3/**RP9**/RA8	39	OSCO/CLKO/**RP33**/IOMAF5/RC0
8	AD1AN3/CVDAN9/CMP5B/IBIAS2/ISRC2/**RP10**/RA9	40	OSCI/**RP34**/IOMAF6/RC1
9	CVDAN10/CMPEN/CMP5C/**RP11**/RA10	41	PGC3/**RP35**/PWM4H/IOMAD7/SS2/FSYNC2/RC2
10	AV_{SS}	42	PGD3/**RP38**/I3CSCL1/PWM4L/IOMAD6/RC5
11	AV_{DD}	43	**RP36**/PWM3H/IOMAD5/RC3
12	OA1OUT/AD1AN0/CVDAN2/CMP1A/**RP3**/RA2	44	**RP37**/I3CSDA1/PWM3L/IOMAD4/RC4
13	OA1IN-/AD1ANN2/AD1AN2/CVDAN3/CMPCN/ CMP1C/**RP4**/RA3	45	**RP43**/RC10
14	UREF/OA1IN+/AD1AN1/AD2AN5/AD3AN4/CVDAN4/ CMP1B/**RP5**/RA4	46	**RP44**/RC11
15	OA3OUT/CVDAN5/CMP3A/**RP6**/INT0/RA5	47	V_{SS}
16	OA3IN-/CVDAN6/CMPDN/CMP3C/**RP7**/RA6	48	V_{DD}
17	OA3IN+/CVDAN21/CVDTX5/CMP3B/**RP22**/RB5	49	**RP49**/PWM2H/IOMAD3/RD0
18	V_{SS}	50	TCK/**RP50**/PWM2L/IOMAD2/RD1
19	V_{DD}	51	TDO/**RP51**/PWM1H/IOMAD1/RD2

Table 4. 64-Pin VQFN, TQFP Complete Pin Function Descriptions(1) (continued)

Pin	Function	Pin	Function
20	OA2OUT/AD2AN0/CVDAN16/CVDTX0/CMP2A/**RP17**/RB0	52	TDI/**RP52**/PWM1L/IOMAD0/RD3
21	TMS/OA2IN-/AD2ANN2/AD2AN2/CVDAN17/CVDTX1/ CMP2C/**RP18**/RB1	53	SWV_{DD}
22	OA2IN+/AD2AN1/CVDAN18/CVDTX2/CMP2B/**RP19**/RB2	54	Lx
23	CVDAN24/CVDTX8/CMP2D/**RP25**/RB8	55	SWV_{SS}
24	CVDAN25/CVDTX9/**RP26**/RB9	56	V_{DDCORE}
25	**RP29**/RB12	57	V_{SS}
26	**RP30**/RB13	58	V_{DD}
27	PGD1/AD2ANN1/AD2AN3/CVDAN19/CVDTX3/CMP4A/ **RP20**/SDA1/RB3	59	TRDATA3/**RP54**/RD5
28	PGC1/AD2AN4/CVDAN20/CVDTX4/CMP4B/**RP21**/SCL1/RB4	60	TRDATA2/**RP55**/RD6
29	AD3ANN2/AD3AN2/CVDAN22/CVDTX6/ALLCMP/IBIAS0/ ISRC0/**RP23**/I3CASCL1/RB6	61	TRDATA1/CVDTX16/**RP53**/RD4
30	CVDAN23/CVDTX7/CMP4C/IBIAS1/ISRC1/**RP24**/I3CASDA1/ IOMAF0/RB7	62	TRDATA0/CVDTX17/**RP56**/ASCL2/IOMAF4/RD7
31	CVDAN26/CVDTX10/**RP27**/SCK2/RB10	63	TRCLKO/CVDTX18/**RP57**/ASDA2/IOMAF3/RD8
32	CVDAN27/CVDTX11/**RP28**/SDI2/RB11	64	$\overline{MCLR}$

Note:

1. **RPn** represents remappable peripheral functions.
2. This pin has 8x drive strength.
3. Unless otherwise stated, pins are 4x drive strength.
4. A pull-up resistor is connected to this pin when the device is erased (JTAG enabled) and during programming.

Pinout I/O Descriptions

Table 5. Pinout I/O Descriptions

Pin Name[1]	Pin Type	Buffer Type	PPS	Description
AD1AN0 - AD1AN4	I	Analog	No	ADC1 positive input channels
AD1ANN1 - AD1ANN2	I	Analog	No	ADC1 negative input channels
AD2AN0 - AD2AN5	I	Analog	No	ADC2 positive input channels
AD2ANN1 - AD2ANN2	I	Analog	No	ADC2 negative input channels
AD3AN0 - AD3AN5	I	Analog	No	ADC3 positive input channels
AD3ANN1 - AD3ANN2	I	Analog	No	ADC3 negative input channels
ADTRG31	I	ST	Yes	ADC Trigger Input 31
CLKI	I	ST/CMOS	No	External Clock (EC) source input. Always associated with OSCI pin function.
CLKO	O	—	No	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSCO pin function.
OSCI	I	ST/CMOS	No	Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise.
OSCO	I/O	—	No	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes.
REFCLKI	I	ST	Yes	Reference clock input
REFCLKO	O	—	Yes	Reference clock output
CVDTX0-CVDTX31	O	—	No	CVD = Capacitive Voltage Divider for the integrated touch controller. CVDTX is the driven capacitive pin.
CVDAN0-CVDAN31	I	Analog	No	Analog input for the integrated touch controller.
INT0	I	ST	No	External Interrupt 0
INT1	I	ST	Yes	External Interrupt 1
INT2	I	ST	Yes	External Interrupt 2
INT3	I	ST	Yes	External Interrupt 3
INT4	I	ST	Yes	External Interrupt 4
IOCA[15:0]	I	ST	No	Interrupt-on-Change input for PORTA
IOCB[15:0]	I	ST	No	Interrupt-on-Change input for PORTB
IOCC[15:0]	I	ST	No	Interrupt-on-Change input for PORTC
IOCD[15:0]	I	ST	No	Interrupt-on-Change input for PORTD
IOMAD[11:0]	O	ST	Yes	I/O Monitor Bank A Reference
IOMBD[11:0]	I	ST	Yes	I/O Monitor Bank B Reference
IOMAF[11:0]	I	ST	Yes	I/O Monitor Bank A Feedback
IOMBF[11:0]	I	ST	Yes	I/O Monitor Bank B Feedback

Legend: CMOS = CMOS compatible input or output; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input; PPS = Peripheral Pin Select; TTL = TTL input buffer

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

Table 5. Pinout I/O Descriptions (continued)

Pin Name[1]	Pin Type	Buffer Type	PPS	Description
QEIA1	I	ST	Yes	QEI Input A1
QEIB1	I	ST	Yes	QEI Input B1
QEINDX1	I	ST	Yes	QEI Index 1 input
QEIHOM1	I	ST	Yes	QEI Home 1 input
QEICMP	O	—	Yes	QEI comparator output
RA0-RA15	I/O	ST	No	PORTA is a bidirectional I/O port
RB0-RB15	I/O	ST	No	PORTB is a bidirectional I/O port
RC0-RC15	I/O	ST	No	PORTC is a bidirectional I/O port
RD0-RD15	I/O	ST	No	PORTD is a bidirectional I/O port
T1CK	I	ST	Yes	Timer1 external clock input
$\overline{\text{U1CTS}}$	I	ST	Yes	UART1 Clear-to-Send
$\overline{\text{U1RTS}}$	O	—	Yes	UART1 Request-to-Send
U1RX	I	ST	Yes	UART1 receive
U1TX	O	—	Yes	UART1 transmit
$\overline{\text{U1DSR}}$	I	ST	Yes	UART1 Data-Set-Ready
U1DTR	O	—	Yes	UART1 Data-Terminal-Ready
$\overline{\text{U2CTS}}$	I	ST	Yes	UART2 Clear-to-Send
$\overline{\text{U2RTS}}$	O	—	Yes	UART2 Request-to-Send
U2RX	I	ST	Yes	UART2 receive
U2TX	O	—	Yes	UART2 transmit
$\overline{\text{U2DSR}}$	I	ST	Yes	UART2 Data-Set-Ready
U2DTR	O	—	Yes	UART2 Data-Terminal-Ready
$\overline{\text{U3CTS}}$	I	ST	Yes	UART3 Clear-to-Send
$\overline{\text{U3RTS}}$	O	—	Yes	UART3 Request-to-Send
U3RX	I	ST	Yes	UART3 receive
U3TX	O	—	Yes	UART3 transmit
$\overline{\text{U3DSR}}$	I	ST	Yes	UART3 Data-Set-Ready
U3DTR	O	—	Yes	UART3 Data-Terminal-Ready
$\overline{\text{U4CTS}}$	I	ST	Yes	UART4 Clear-to-Send
$\overline{\text{U4RTS}}$	O	—	Yes	UART4 Request-to-Send
U4RX	I	ST	Yes	UART4 receive
U4TX	O	—	Yes	UART4 transmit
$\overline{\text{U4DSR}}$	I	ST	Yes	UART4 Data-Set-Ready
U4DTR	O	—	Yes	UART4 Data-Terminal-Ready
SENT1	I	ST	Yes	SENT1 input
SENT2	I	ST	Yes	SENT2 input
SENT1OUT	O	—	Yes	SENT1 output
SENT2OUT	O	—	Yes	SENT2 output
PTGTRG24	O	—	Yes	PTG Trigger Output 24
PTGTRG25	O	—	Yes	PTG Trigger Output 25

Legend: CMOS = CMOS compatible input or output; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input; PPS = Peripheral Pin Select; TTL = TTL input buffer

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

Table 5. Pinout I/O Descriptions (continued)

Pin Name[1]	Pin Type	Buffer Type	PPS	Description
TCKI1-TCKI5	I	ST	Yes	SCCP/MCCP Timer Inputs 1 through 5
ICM1-ICM5	I	ST	Yes	SCCP/MCCP Capture Inputs 1 through 5
OCFA-OCFD	I	—	Yes	SCCP/MCCP Fault Inputs A through D
OCM1-OCM5	O	—	Yes	SCCP/MCCP Compare Outputs 1 through 5
SCK1	I/O	ST	Yes	Synchronous serial clock I/O for SPI1
SDI1	I	ST	Yes	SPI1 data in
SDO1	O	—	Yes	SPI1 data out
$\overline{SS1}$	I/O	ST	Yes	SPI1 Client synchronization or frame pulse I/O
SCK2	I/O	ST	Yes	Synchronous serial clock I/O for SPI2
SDI2	I	ST	Yes	SPI2 data in
SDO2	O	—	Yes	SPI2 data out
$\overline{SS2}$	I/O	ST	Yes	SPI2 Client synchronization or frame pulse I/O
SCK3	I/O	ST	Yes	Synchronous serial clock I/O for SPI3
SDI3	I	ST	Yes	SPI3 data in
SDO3	O	—	Yes	SPI3 data out
$\overline{SS3}$	I/O	ST	Yes	SPI3 Client synchronization or frame pulse I/O
SCL1	I/O	ST	No	Synchronous serial clock I/O for I2C1
SDA1	I/O	ST	No	Synchronous serial data I/O for I2C1
ASCL1	I/O	ST	No	Alternate synchronous serial clock I/O for I2C1
ASDA1	I/O	ST	No	Alternate synchronous serial data I/O for I2C1
SCL2	I/O	ST	No	Synchronous serial clock I/O for I2C2
SDA2	I/O	ST	No	Synchronous serial data I/O for I2C2
ASCL2	I/O	ST	No	Alternate synchronous serial clock I/O for I2C2
ASDA2	I/O	ST	No	Alternate synchronous serial data I/O for I2C2
BISS1SL	I	ST	Yes	BiSS1 Return Input
BISS1GS	I	ST	Yes	BiSS1 Get Sense
BISS1MO	O	ST	Yes	BiSS1 Output
BISS1MA	O	ST	Yes	BiSS1 Clock
I3CSCL1	I/O	ST	No	Synchronous serial clock I/O for I3C1
I3CSDA1	I/O	ST	No	Synchronous serial data I/O for I3C1
AI3CSCL1	I/O	ST	No	Alternate synchronous serial clock I/O for I3C1
AI3CSDA1	I/O	ST	No	Alternate synchronous serial data I/O for I3C1
TMS	I	ST	No	JTAG Test mode select pin
TCK	I	ST	No	JTAG test clock input pin
TDI	I	ST	No	JTAG test data input pin
TDO	O	—	No	JTAG test data output pin
TRCLKO	O	DIG	No	Trace Clock
TRDATA0-3	O	DIG	No	Trace Data

Legend: CMOS = CMOS compatible input or output; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input; PPS = Peripheral Pin Select; TTL = TTL input buffer

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

Table 5. Pinout I/O Descriptions (continued)

Pin Name[1]	Pin Type	Buffer Type	PPS	Description
PCI8-PCI18	I	ST	Yes	PWM PCI Inputs 8 through 18
PCI19-PCI22	I	ST	Yes	PWM PCI Inputs 19 through 22
PWMEA-PWMEF	O	—	Yes	PWM Event Outputs A through F
PWM1L-PWM4L[2]	O	—	Yes	PWM Low Outputs 1 through 4
PWM1H-PWM4H[2]	O	—	Yes	PWM High Outputs 1 through 4
CLCINA-CLCIND	I	ST	Yes	CLC Inputs A through D
CLC1OUT-CLC4OUT	O	—	Yes	CLC Outputs 1 through 4
CMP1A-CMP5A	I	Analog	No	Comparator Channels 1A through 5A inputs
CMP1B-CMP5B	I	Analog	No	Comparator Channels 1B through 5B inputs
CMP1C-CMP5C	I	Analog	No	Comparator Channels 1C through 5C inputs
CMP2D, CMP4D	I	Analog	No	Comparator Channels 2D through 4D inputs
CMPNC-CMPNF	I	Analog	No	All Comparators Negative inputs
ALLCMPP	I	Analog	No	All Comparators Positive input
DACOUT1	O	—	No	DAC1 output voltage
IBIAS0-3, ISRC0-3	O	Analog	No	Constant-Current Outputs 0 through 3
OA1IN+	I	—	No	Op Amp 1+ input
OA1IN-	I	—	No	Op Amp 1- input
OA1OUT	O	—	No	Op Amp 1 output
OA2IN+	I	—	No	Op Amp 2+ input
OA2IN-	I	—	No	Op Amp 2- input
OA2OUT	O	—	No	Op Amp 2 output
OA3IN+	I	—	No	Op Amp 3+ input
OA3IN-	I	—	No	Op Amp 3- input
OA3OUT	O	—	No	Op Amp 3 output
RDCEXC	O	—	Yes	RDC Excitation Output
RDCEXCI	O	—	Yes	RDC Excitation Output Inverted
UREF	O	—	No	UREF Output
PGD1	I/O	ST	No	Data I/O pin for Programming/ Debugging Communication Channel 1
PGC1	I	ST	No	Clock input pin for Programming/ Debugging Communication Channel 1
PGD2	I/O	ST	No	Data I/O pin for Programming/ Debugging Communication Channel 2
PGC2	I	ST	No	Clock input pin for Programming/ Debugging Communication Channel 2
PGD3	I/O	ST	No	Data I/O pin for Programming/ Debugging Communication Channel 3
PGC3	I	ST	No	Clock input pin for Programming/ Debugging Communication Channel 3

Legend: CMOS = CMOS compatible input or output; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input; PPS = Peripheral Pin Select; TTL = TTL input buffer

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

Table 5. Pinout I/O Descriptions (continued)

Pin Name[1]	Pin Type	Buffer Type	PPS	Description
$\overline{MCLR}$	I/P	ST	No	Master Clear (Reset) input. This pin is an active-low Reset to the device.
AV_{DD}	P	P	No	Positive supply for analog modules. This pin must be connected at all times.
AV_{SS}	P	P	No	Ground reference for analog modules. This pin must be connected at all times.
V_{DD}	P	—	No	Positive supply for peripheral logic and I/O pins
V_{SS}	P	—	No	Ground reference for logic and I/O pins
V_{DD} Core	P	—	No	1.1V output from internal buck regulator; positive supply for core logic.
SWV_{DD}	P	—	No	Positive supply for the buck regulator circuit.
SWV_{SS}	P	—	No	No ground reference for the buck regulator circuit.
LX	P	—	No	No switching pin to inductor for the buck regulator circuit.

Legend: CMOS = CMOS compatible input or output; Analog = Analog input; P = Power; ST = Schmitt Trigger input with CMOS levels; O = Output; I = Input; PPS = Peripheral Pin Select; TTL = TTL input buffer

Notes:

1. Not all pins are available in all package variants. See the Pin Diagrams section for pin availability.
2. These pins are remappable as well as dedicated.

To Our Valued Customers

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at doc.errors@microchip.com. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Website at:

www.microchip.com/

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Website: www.microchip.com/
- Your local Microchip sales office

When contacting a sales office, please specify which device, revision of silicon and data sheet (include the literature number) you are using.

Customer Notification System

Register on our website at www.microchip.com/ to receive the most current information on all of our products.

Terminology Cross Reference

Table 6 provides updated terminology for deprecated naming conventions. Register and bit names remain unchanged, however, descriptions and usage guidance may have been updated.

Table 6. Terminology Cross References

Use Case	Deprecated Term	New Term
CPU	Master	Initiator
DMA	Master	Initiator
I^2C	Master	Host
	Slave	Client
SPI	Master	Host
	Slave	Client
UART, LIN Mode	Master	Commander
	Slave	Responder
PWM	Master	Host
	Slave	Client

Table of Contents

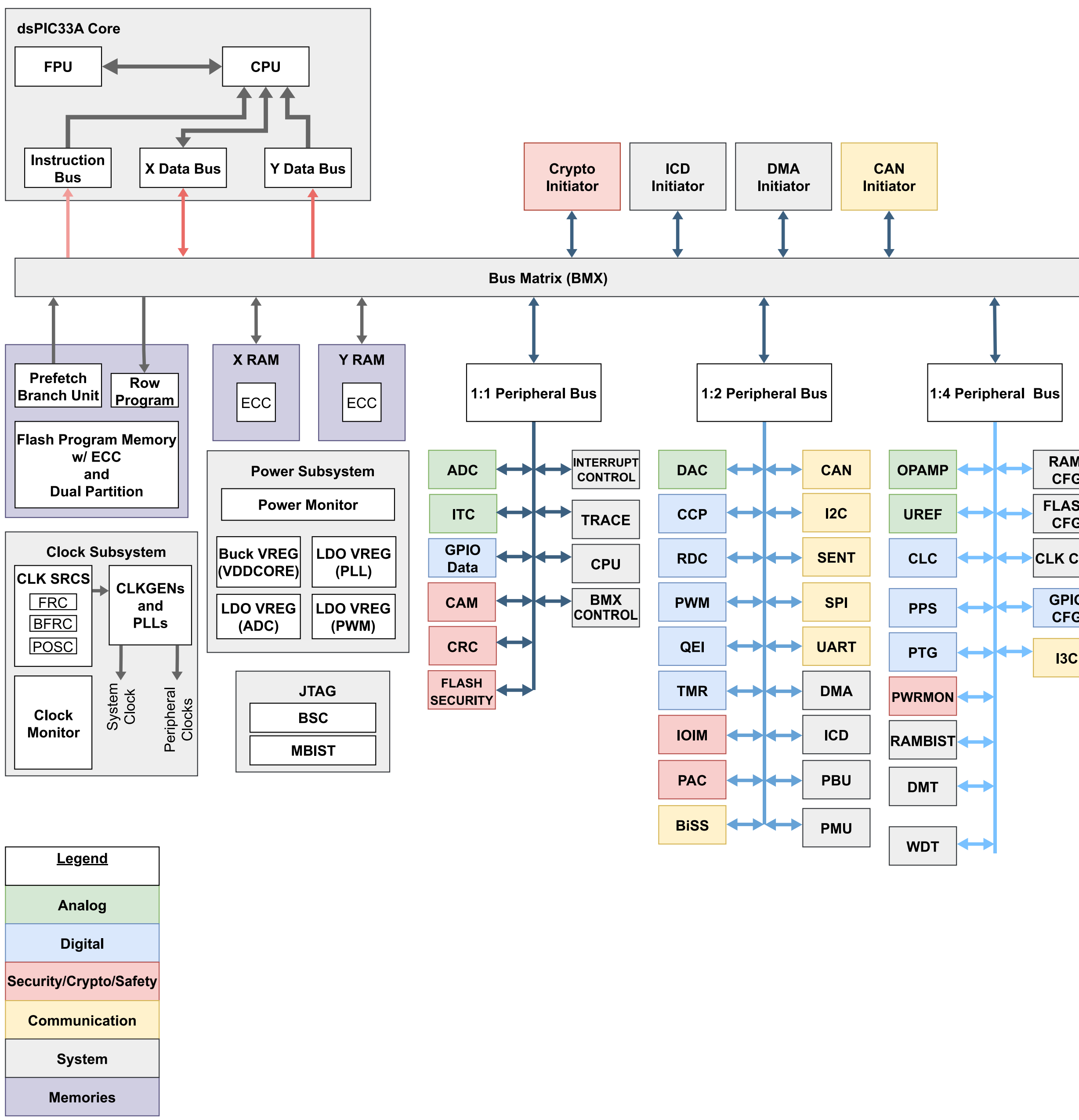
1. Device Overview

This document contains device-specific information for the dsPIC33AK256MPS306 Digital Signal Controller (DSC) family of devices.

dsPIC33AK256MPS306 devices feature Digital Signal Processor (DSP) functionality with a high-performance architecture and a single and double precision Floating Point Unit (FPU). dsPIC33AK256MPS306 family devices have an internal low-voltage buck converter to supply power to the core.

Figure 1-1 shows a general block diagram of the core and peripheral modules of the dsPIC33AK256MPS306 family.

Figure 1-1. dsPIC33AK256MPS306 Family Block Diagram



Notes:

1. Not all I/O pins or features are implemented on all device pinout configurations. See Pinout I/O Descriptions for specific implementations by pin count.
2. Some peripheral I/Os are accessible only through Peripheral Pin Select (PPS).

2. Guidelines for Getting Started with Digital Signal Controllers

2.1. Basic Connection Requirements

Getting started with the dsPIC33AK256MPS306 family devices requires attention to a minimal set of device pin connections before proceeding with development. The following pins must always be connected:

- All V_{DD} and V_{SS} power supply pins must be properly biased with required voltages (see Electrical Characteristics).
- All AV_{DD} and AV_{SS} analog supply pins must be properly biased regardless of which analog modules or components of the dsPIC33A device are used (see Electrical Characteristics).
- The $\overline{MCLR}$ pin is connected with V_{DD} and V_{SS} based on circuit or application needs.
- PGCx/PGDx for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see ICSP Pins)
- OSCI and OSCO pins when an external oscillator source is used (see External Oscillator Pins)

2.2. Decoupling Capacitors

The use of decoupling capacitors on every pair of power supply pins, such as V_{DD}, V_{SS}, AV_{DD} and AV_{SS}, is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** Recommendation of two 0.1 µF (100 nF) in parallel rated at 10-20V. These capacitors should be low-ESR and have a resonance frequency in the range of 20 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on the opposite side of the PCB connected through a via; however, ensure that the trace length from the pin to the capacitor is within one-quarter inch (6 mm) in length.
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise above tens of MHz, add an additional ceramic-type capacitor in parallel to the decoupling capacitors. The value can be in the range of 0.01 µF to 0.001 µF. Place this capacitor next to the primary decoupling capacitors. In high-speed circuit designs, consider implementing a set of capacitances as close to the power and ground pins as possible. For example, 0.1 µF in parallel with 0.01 µF and 0.001 µF.
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. It is equally important to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB track inductance.

2.2.1. Bulk Capacitors

For on boards with power traces running longer than six inches in length, it is suggested to use a bulk capacitor for integrated circuits, including DSCs, to supply a local power source. The value of the bulk capacitor should be determined based on the trace resistance that connects the power supply source to the device and the maximum current drawn by the device in the application. In other words, select the bulk capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7 µF to 47 µF.

2.3. Power Sequencing

The dsPIC33AK256MPS306 family requires power sequencing when running high speeds and heavy loads. Starting the device at full load or waking from sleep can result in a device Reset from a core voltage supervisor trip. The recommended method is to ensure that high-power consumers (PLLs, PLL start-up, etc) are sequenced with delays between one another.

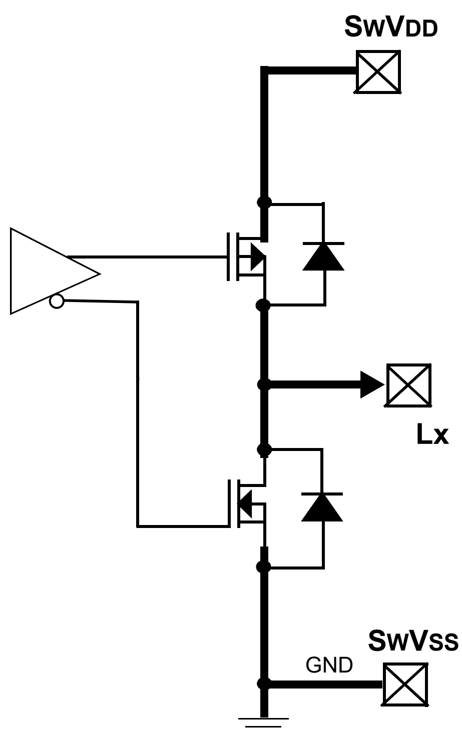
2.4. Buck Converter Guidelines and Considerations

A buck converter is implemented on the dsPIC33AK256MPS306 device family. The buck allows for reduced power loss and heat generated within the device. The connections are:

- SWV_{DD} – This power pin is connected to the internal power switches of the buck converter. This pin has high transient currents and should have a low impedance path to V_{DD}. A 10 µF (4.7 µF minimum) capacitor is recommended as close as possible to the pin.
- SWV_{SS} – This power pin is connected to the internal power switches of the buck converter. This pin has high transient currents and should have a low impedance path to the PCB ground.
- L_X - This pin is the output of the H bridge power switches and provides current to the external inductor. This pin has high transient currents and should have a low impedance trace to the inductor.
- V_{DDCORE} – This pin supplies power from the external inductor to the core. A 10 µF capacitor is required and should be placed as close to the V_{DDCORE} pin and inductor as possible.

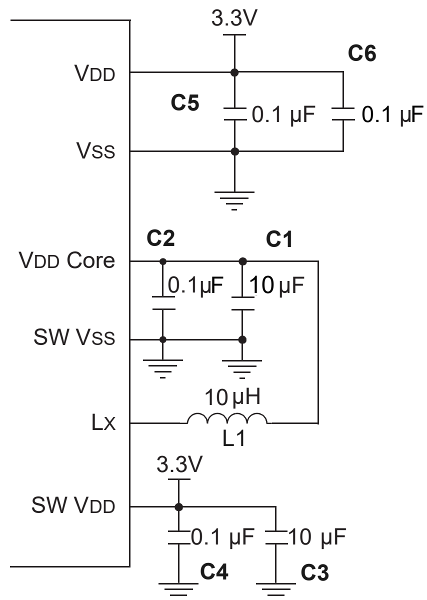
The buck converter adjusts the duty cycle to maintain a constant voltage output as the current demands change. Current is delivered to the core through the external inductor and is filtered with the capacitor in the V_{DDCORE} pin. Figure 2-1 illustrates a simplified topology of the buck components inside the dsPIC®.

Figure 2-1. Buck Regulator Topology



No other connections should be made to the V_{DDCORE} or L_X pins. The output voltage of the buck converter is listed in Electrical Characteristics. An example application circuit is shown in Figure 2-2.

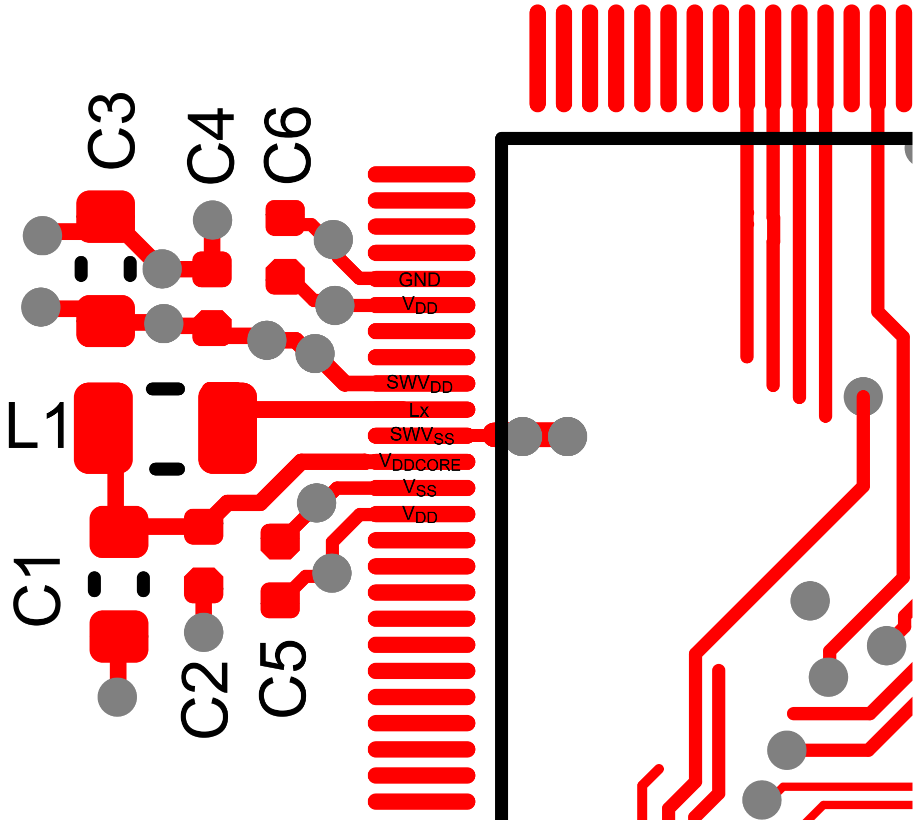
Figure 2-2. Example Application Circuit



2.4.1. Buck Design Considerations

The buck regulator runs at a fixed frequency of 3 MHz nominal, and the inductor and V_{CORE} capacitor are sized accordingly for the buck's control loop performance. Additional parasitic resistance and inductance from the PCB pads and traces can shift the operating parameters and cause instability under certain conditions. Therefore, it is important to minimize all related trace lengths and compact the buck layout as much as possible. It is recommended to place all components on the same side of the PCB as the dsPIC device. If the same side is not physically possible, buck components can be placed on the opposite side with the same goal of minimal trace length and parasitics. An example layout is shown in Figure 2-3.

Figure 2-3. Example Buck Component Layout



2.4.2. Component Selection

The selection of the inductor and filter capacitor is important for robust operation. The temperature ratings should be sufficient for the intended environmental conditions. The saturation current rating and DC resistance requirements are listed in Table 2-1.

Table 2-1. Inductor Selection Guidance

Inductance (µH)	DCR (Ohm)	Isat (mA)
10	<1	>300

2.5. Master Clear ($\overline{MCLR}$) Pin

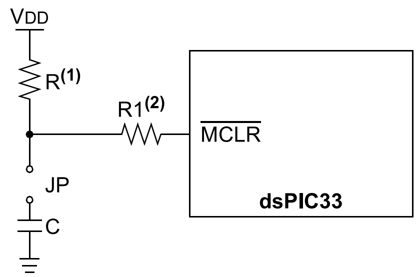
The $\overline{MCLR}$ pin provides two specific device functions:

- Device Reset
- Device Programming and Debugging

During device programming and debugging, the resistance and capacitance that can be added to the pin must be considered. Device programmers and debuggers drive the $\overline{MCLR}$ pin. Consequently, specific voltage levels (V_{IH} and V_{IL}) and fast signal transitions must not be adversely affected. Ensure that the $\overline{MCLR}$ pin (V_{IH} and V_{IL}) voltage specifications are met.

For example, Figure 2-4 shows the $\overline{MCLR}$ pin connections with general circuit components used, such as resistor R, series resistor R1 and capacitor C, and their placements. It is recommended to place these passive components with one-quarter inch (6mm) from the $\overline{MCLR}$ pin.

Figure 2-4. Example of $\overline{\text{MCLR}}$ Pin Connections



Notes:
1. R ≤ 10 kΩ is recommended. A suggested starting value is 10 kΩ. Ensure that the $\overline{\text{MCLR}}$ pin V_{IH} and V_{IL} specifications are met.
2. R1 ≤ 470Ω will limit any current flowing into $\overline{\text{MCLR}}$ from the external capacitor, C, in the event of $\overline{\text{MCLR}}$ pin breakdown due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS). Ensure that the $\overline{\text{MCLR}}$ pin V_{IH} and V_{IL} specifications are met.
3. C ≤ 1 µF may be recommended. However, values of C should be based on Reset timings required for any application. Make sure to isolate C from the $\overline{\text{MCLR}}$ pin during programming and debugging operations.

2.6. ICSP Pins

The PGCx and PGDx pins are used for programming and debugging purposes. There are three pairs of PGCx and PGDx pins to select from based on the application's need of shared pin functions. There are no configuration bits to explicitly choose which pair is used. Instead, the device automatically detects which pair is being used to enter ICSP mode and configures itself,accordingly. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of Ohms, not to exceed 100Ω.

Pull-up resistors, series diodes and capacitors on the PGCx and PGDx pins are not recommended as they will interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they should be removed from the circuit during programming and debugging. Alternatively, refer to the AC/DC characteristics and timing requirements information in the respective device Flash programming specification for information on capacitive loading limits and pin Voltage Input High (V_{IH}) and Voltage Input Low (V_{IL}) requirements.

2.7. External Oscillator Pins

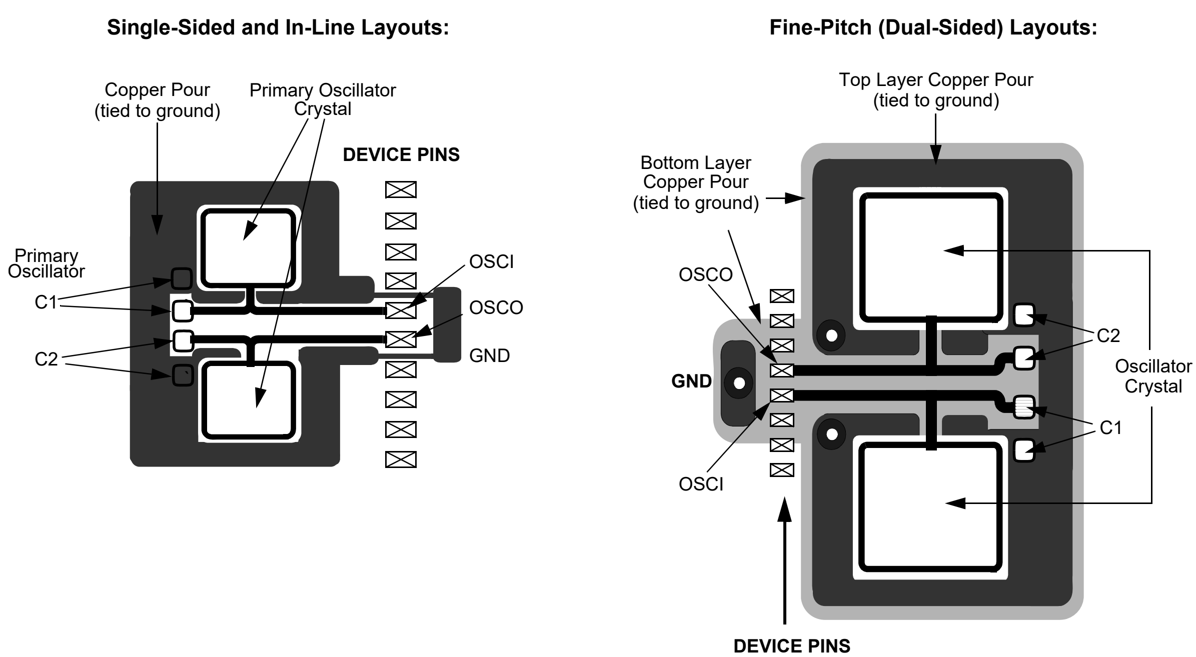
When the Primary Oscillator (POSC) circuit is used to connect a crystal oscillator, special care and consideration are needed to ensure proper operation. The POSC circuit should be tested across the environmental conditions in which the end product is intended to be used. The load capacitors specified in the crystal oscillator data sheet can be used as a starting point; however, the parasitic capacitance from the PCB traces can affect the circuit, and the values may need to be altered to ensure proper start-up and operation. Excessive trace length and other physical interactions can lead to poor signal quality. Poorly tuned oscillator circuits can have reduced amplitude, incorrect frequency (runt pulses), distorted waveforms and long start-up times that may result in unpredictable application behavior, such as instruction misexecution, illegal opcode fetch, etc. Ensure that the crystal oscillator circuit is at full amplitude and the correct frequency before the system begins to execute code. In planning the application’s routing and I/O assignments, ensure that adjacent port pins and other signals in close proximity to the oscillator do not have high frequencies, short rise and fall times, and other similar noise. For further information on the POSC, see Primary Oscillator (POSC).

2.8. External Oscillator Layout Guidance

Use best practices during PCB layout to ensure robust start-up and operation. The oscillator circuit should be placed on the same side of the board as the device. Also, place the oscillator circuit close to the respective oscillator pins, not exceeding one-half inch (12 mm) distance between them. The load capacitors should be placed next to the oscillator itself, on the same side of the board. Use a grounded copper pour around the oscillator circuit to isolate it from surrounding circuits. The grounded copper pour should be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. If using a two-sided board, avoid any traces on the other side of the board where the crystal is placed. Suggested layouts are shown in Figure 2-5. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground. For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the Microchip website (www.microchip.com):

- AN943, *"Practical PICmicro® Oscillator Analysis and Design"*
- AN949, *"Making Your Oscillator Work"*

Figure 2-5. Suggested Placement of the Oscillator Circuit



2.9. Oscillator Value Conditions on Device Start-up

If the PLL of the target device is enabled and configured for the device start-up oscillator, the maximum oscillator source frequency must be limited to a certain frequency (see Phase-Locked Loop (PLL)) to comply with device PLL start-up conditions. This means that if the external oscillator frequency is outside this range, the application must start up in the FRC mode first. The default PLL settings after a POR with an oscillator frequency outside this range will violate the device operating speed.

Once the device powers up, the application firmware can initialize the PLL SFRs, CLKxDIV and PLLxDIV to a suitable value and then perform a clock switch to the Oscillator + PLL clock source.

2.10. Unused I/Os

Unused I/O pins should be configured as outputs and driven to a Logic Low state. Alternatively, connect a resistor (1k-10k ohm) between V_{SS} and unused pins, and drive the output to a logic low.

2.11. Targeted Applications

- Power Factor Correction (PFC)
  - Interleaved PFC, Totem-pole PFC
  - Critical Conduction PFC
  - Multi-Level Flying Capacitor PFC
- DC/DC Converters
  - Buck, Boost, Forward, Flyback, Push-Pull
  - Half/Full-Bridge
  - Phase-Shift Full-Bridge
  - Resonant Converters
- DC/AC
  - Half/Full-Bridge Inverter
  - Resonant Inverter
- Motor Control
  - BLDC
  - PMSM
  - SR
  - ACIM
- Advanced Sensor Interfacing
- High-Performance Embedded Control
- High-Speed Data Acquisition and Processing
- Safety-Critical Designs
- Digital Lighting

3. CPU

The dsPIC33AK256MPS306 family has a fixed-point fractional DSP engine supporting the Central Processing Unit (CPU). The CPU processes instructions out of program memory and utilizes system RAM to perform tasks and calculations. The CPU is interfaced to memory and peripherals through the bus matrix. The CPU supports coprocessors, including the Floating-Point Unit (FPU) for mathematical computation.

CPU key features:

- 32-bit Working Registers
- Unified Memory Map
- 5-Stage Instruction Pipeline
- Conditional Branching with Speculative Execution
- Instruction Prefetch Cache
- Mathematical Support
- Low Overhead Loop Support

3.1. Architectural Overview

The CPU has 32-bit (data) modified Harvard architecture with 32-bit instructions, a 5-stage instruction pipeline and a single-phase clock design.

The CPU has a 32-bit instruction word with a variable length opcode field. The CPU also supports some instructions that are only available in 16-bit format. The Program Counter (PC) is 24 bits wide to access a 16 MB (24-bit address) unified linear address map.

The CPU supports up to eight addressing modes. A 5-stage fully interlocked instruction pipeline, with reduced branch latency and hardware mitigated pipeline hazard stalls, helps maintain throughput and provides predictable execution. Most instructions execute in a single-cycle effective execution rate, except for instructions that change the program flow. A hardware program loop construct is supported by the overhead-free `REPEAT` instruction, which is interruptible at any point. For loops greater than one instruction, the `DBT` (Decrement Test and Branch) instruction may be used to reduce loop overhead.

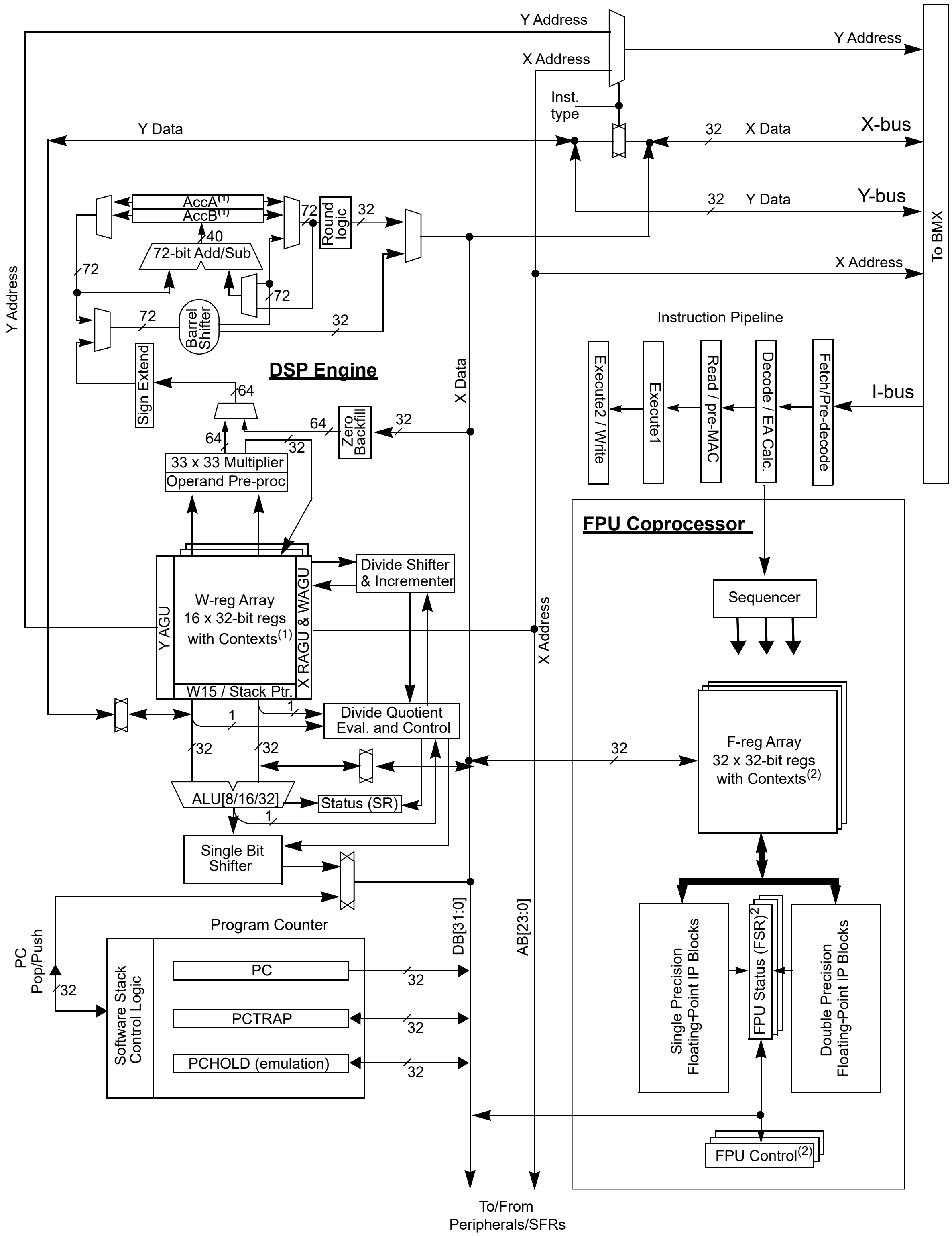
The CPU supports high-performance arithmetic calculation with a tightly coupled 16/32-bit Integer and a Fixed-Point fractional DSP engine with a 72-bit shifter, saturation and rounding support. There is a common issue Single and Double Precision Floating-Point Unit (FPU) coprocessor with an independent load-store execution pipeline.

The CPU and coprocessor support the following features:

- Decode and issue from the CPU pipeline into independent coprocessor pipeline(s)
- Pipeline hazards are detected and mitigated in both the CPU and coprocessor(s)
- Instructions include a dedicated data move and a conditional coprocessor status branch
- Coprocessor interrupt support

Figure 3-1 illustrates the CPU block diagram.

Figure 3-1. Core Conceptual Block Diagram with a FPU Coprocessor



Notes:

1. The CPU includes base plus seven register contexts (one per IPL) for W0 through W7, AccA, AccB, RCOUNT and CORCON[15:0].
2. The FPU includes eight register contexts (one per IPL) for F0 through F7, FSR and FCR.

3.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x00	PC	31:24								
		23:16	PC[23:16]							
		15:8	PC[15:8]							
		7:0	PC[7:0]							
0x04	SPLIM	31:24								
		23:16	SPLIM[23:16]							
		15:8	SPLIM[15:8]							
		7:0	SPLIM[7:0]							
0x08	RCOUNT	31:24	RCOUNT[31:24]							
		23:16	RCOUNT[23:16]							
		15:8	RCOUNT[15:8]							
		7:0	RCOUNT[7:0]							
0x0C	DISIIPL	31:24								
		23:16								
		15:8								
		7:0						DISIIPL[2:0]		
0x10	CORCON	31:24								
		23:16								
		15:8				US		IPLST[2:0]		
		7:0	SATA	SATB	SATDW	ACCSAT			RND	IF
0x14	MODCON	31:24								
		23:16								
		15:8	XMODEN	YMODEN						
		7:0	YWM[3:0]				XWM[3:0]			
0x18	XMODSRT	31:24								
		23:16	XMODSRT[23:16]							
		15:8	XMODSRT[15:8]							
		7:0	XMODSRT[7:0]							
0x1C	XMODEND	31:24								
		23:16	XMODEND[23:16]							
		15:8	XMODEND[15:8]							
		7:0	XMODEND[7:0]							
0x20	YMODSRT	31:24								
		23:16	YMODSRT[23:16]							
		15:8	YMODSRT[15:8]							
		7:0	YMODSRT[7:0]							
0x24	YMODEND	31:24								
		23:16	YMODEND[23:16]							
		15:8	YMODEND[15:8]							
		7:0	YMODEND[7:0]							
0x28	XBREV	31:24								
		23:16								
		15:8		XBREV[14:8]						
		7:0	XBREV[7:0]							
0x2C	PCTRAP	31:24								
		23:16	PCTRAP[23:16]							
		15:8	PCTRAP[15:8]							
		7:0	PCTRAP[7:0]							
0x30	FEX	31:24	FEX[31:24]							
		23:16	FEX[23:16]							
		15:8	FEX[15:8]							
		7:0	FEX[7:0]							
0x34	FEX2	31:24	FEX2[31:24]							
		23:16	FEX2[23:16]							
		15:8	FEX2[15:8]							
		7:0	FEX2[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x38	PCHOLD	31:24								
		23:16	PCHOLD[23:16]							
		15:8	PCHOLD[15:8]							
		7:0	PCHOLD[7:0]							
0x3C	VFA	31:24								
		23:16	VFA[23:16]							
		15:8	VFA[15:8]							
		7:0	VFA[7:0]							
0x40 ... 0x1E0F	Reserved									
0x1E10	HPCCON	31:24								
		23:16								
		15:8	ON		CLR					
		7:0								
0x1E10	HPCSEL0	31:24				SELECT[3][4:0]				
		23:16				SELECT[2][4:0]				
		15:8				SELECT[1][4:0]				
		7:0				SELECT[0][4:0]				
0x1E14	HPCSEL1	31:24				SELECT[7][4:0]				
		23:16				SELECT[6][4:0]				
		15:8				SELECT[5][4:0]				
		7:0				SELECT[4][4:0]				
0x1E18 ... 0x1E1F	Reserved									
0x1E20	HPCCNTL0	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E24	HPCCNTH0	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E28	HPCCNTL1	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E2C	HPCCNTH1	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E30	HPCCNTL2	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E34	HPCCNTH2	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E38	HPCCNTL3	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E3C	HPCCNTH3	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E40	HPCCNTL4	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E44	HPCCNTH4	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E48	HPCCNTL5	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E4C	HPCCNTH5	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E50	HPCCNTL6	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E54	HPCCNTH6	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E58	HPCCNTL7	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E5C	HPCCNTH7	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E60	CHECON	31:24								
		23:16								ISBBUF
		15:8	ON				CHEINV	CHECOH		
		7:0								FLTINJ
0x1E64	CHESTAT	31:24								
		23:16								
		15:8								
		7:0					ISBPE	TPE	ISBE	PARE
0x1E68	CHEFLTINJ	31:24								
		23:16								
		15:8								
		7:0	FLTPTR[7:0]							
0x1E6C	CHEDAT0	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E70	CHEDAT1	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E74	CHEDAT2	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E78	CHEDAT3	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E7C	CHECMD	31:24								
		23:16	RIP3	RIP2	RIP1	RIP0	WIP3	WIP2	WIP1	WIP0
		15:8			RVAL	WVAL	CMPV	WV		
		7:0		ADDR[6:0]						
0x1E80	TAGDAT	31:24								
		23:16	ADDR[23:16]							
		15:8								
		7:0	ADDR[7:0]							
0x1E84	TAGCMD	31:24						RTP0	WTP0	
		23:16		CMPP	EWP	EWD	CMPA	RD		
		15:8								
		7:0		ADDR[6:0]						
0x1E88	ISBDAT0	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E8C	ISBDAT1	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E90	ISBDAT2	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E94	ISBDAT3	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E98	ISBCMD0	31:24	RIP[3:0]				WIP[3:0]			
		23:16		CMPP	EWD	EWP	CMPD	RD		
		15:8						LINE[2:0]		
		7:0	ISBREG[2:0]							
0x1E9C	ISBCMD1_7	31:24	RIP[3:0]				WIP[3:0]			
		23:16		CMPP	EWD	EWP	CMPD	RD		
		15:8						LINE[2:0]		
		7:0	ISBREG[2:0]					SLICE[2:0]		

3.2.1. CPU Program Counter Register

Name: PC
Offset: 0x000

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	PC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – PC[23:0] Program Counter bits

3.2.2. Stack Pointer Limit Value Register

Name: SPLIM
Offset: 0x004

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SPLIM[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SPLIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SPLIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – SPLIM[23:0] Stack Limit Address bits

3.2.3. REPEAT Loop Counter Register

Name: RCOUNT
Offset: 0x008

Bit	31	30	29	28	27	26	25	24
	RCOUNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RCOUNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RCOUNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RCOUNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – RCOUNT[31:0] Loop Counter Value for `REPEAT` Instruction bits

3.2.4. DISIIPL(W) Instruction Current IPL Threshold

Name: DISIIPL
Offset: 0x00C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						DISIIPL[2:0]		
Access						R	R	R
Reset						0	0	0

Bits 2:0 – DISIIPL[2:0] Current IPL Threshold Value bits

3.2.5. Core Mode Control Register[1]

Name: CORCON
Offset: 0x010

Note:
1. The Core Mode Control register (CORCON) has bits that control the operation of the DSP multiplier hardware.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				US			IPLST[2:0]	
Access				R/W		R/W	R/W	R/W
Reset				0		0	0	0

Bit	7	6	5	4	3	2	1	0
	SATA	SATB	SATDW	ACCSAT			RND	IF
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	1	0			0	0

Bit 12 – US Unsigned or Signed Multiplier Mode Select bit

Value	Description
1	Unsigned mode enabled for DSP ops
0	Signed mode enabled for DSP ops

Bits 10:8 – IPLST[2:0] Interrupt Priority Level Supervisor Mode Threshold bits
User mode: These bits are read-only.
Supervisor mode: The bits are R/W-0 (CPU will reset into Supervisor mode)

Value	Description
111	No interrupts will execute in Supervisor mode.
110	Level 7 interrupts will execute in Supervisor mode.
101	Level 6 and 7 interrupts will execute in Supervisor mode.
100	Level 5 through 7 interrupts will execute in Supervisor mode.
011	Level 4 through 7 interrupts will execute in Supervisor mode.
010	Level 3 through 7 interrupts will execute in Supervisor mode.
001	Level 2 through 7 interrupts will execute in Supervisor mode.
000	Level 1 through 7 interrupts will execute in Supervisor mode.

Bit 7 – SATA AccA Saturation Enable bit

Value	Description
1	Accumulator A saturation enabled
0	Accumulator A saturation disabled

Bit 6 – SATB AccB Saturation Enable bit

Value	Description
1	Accumulator B saturation enabled
0	Accumulator B saturation disabled

Bit 5 – SATDW Data Space Write from DSP Engine Saturation Enable bit

Value	Description
1	Data Space write saturation enabled
0	Data Space write saturation disabled

Bit 4 – ACCSAT Accumulator Saturation Mode Select bit

Value	Description
1	9.63 saturation (super saturation)
0	1.63 saturation (normal saturation)

Bit 1 – RND Rounding Mode Select bit

Value	Description
1	Biased (conventional) rounding enabled
0	Unbiased (convergent) rounding enabled

Bit 0 – IF Integer or Fractional Multiplier Mode Select bit

Value	Description
1	Integer mode is enabled for DSP multiply.
0	Fractional mode is enabled for DSP multiply.

3.2.6. Modulo Addressing Control Register[1]

Name: MODCON
Offset: 0x0014

Notes:
1. The MODCON register enables and configures Modulo Addressing (circular buffers).
2. This bit is maintained as '1' always and cannot be set by the user.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	XMODEN	YMODEN						
Access	R	R/W						
Reset	1	0						

Bit	7	6	5	4	3	2	1	0
	YWM[3:0]				XWM[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit 15 – XMODEN X RAGU & X WAGU Modulo Addressing Enable bit[2]

Value	Description
1	X AGU Modulo Addressing enabled
0	X AGU Modulo Addressing disabled

Bit 14 – YMODEN Y AGU Modulo Addressing Enable bit

Value	Description
1	Y AGU Modulo Addressing enabled
0	Y AGU Modulo Addressing disabled

Bits 7:4 – YWM[3:0] Y AGU W Register Select for Modulo Addressing bits

Value	Description
1111	Modulo Addressing disabled (W15 does not support Modulo Addressing)
1110	W14 selected for Modulo Addressing
...	
0000	W0 selected for Modulo Addressing

Bits 3:0 – XWM[3:0] X RAGU & X WAGU W Register Select for Modulo Addressing bits

Value	Description
1111	Modulo Addressing disabled (W15 does not support Modulo Addressing)
1110	W14 selected for Modulo Addressing
...	

Value	Description
0000	W0 selected for Modulo Addressing

3.2.7. X AGU Modulo Addressing Start Register[(1)]

Name: XMODSRT
Offset: 0x0018

Note:
1. The XMODSRT and XMODEND registers hold the start and end addresses for modulo (circular) buffers implemented in the X data memory address space.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	XMODSRT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	XMODSRT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	XMODSRT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – XMODSRT[23:0] X RAGU and X WAGU Modulo Addressing Start Address bits

3.2.8. X AGU Modulo Addressing End Register[1]

Name: XMODEND
Offset: 0x001C

Note:
1. The XMODSRT and XMODEND registers hold the start and end addresses for modulo (circular) buffers implemented in the X data memory address space.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	XMODEND[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	XMODEND[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	XMODEND[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 23:0 – XMODEND[23:0] X RAGU & X WAGU Modulo Addressing End Address bits

3.2.9. Y AGU Modulo Addressing Start Address Register[1]

Name: YMODSRT
Offset: 0x0020

Note:
1. The YMODSRT and YMODEND registers hold the start and end addresses for modulo (circular) buffers implemented in the Y data memory address space.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	YMODSRT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	YMODSRT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	YMODSRT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – YMODSRT[23:0] Y RAGU Modulo Addressing Start Address bits

3.2.10. Y AGU Modulo Addressing End Register[1]

Name: YMODEND
Offset: 0x0024

Note:
1. The YMODSRT and YMODEND registers hold the start and end addresses for modulo (circular) buffers implemented in the Y data memory address space.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	YMODEND[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	YMODEND[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	YMODEND[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 23:0 – YMODEND[23:0] Y RAGU Modulo Addressing End Address bits

3.2.11. X AGU Bit Reversal Addressing Control Register[1]

Name: XBREV
Offset: 0x0028

Note:
1. The XBREV register sets the buffer size used for Bit-Reversed Addressing.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		XBREV[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	XBREV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 14:0 – XBREV[14:0] X AGU Bit Reversed Modifier bits

3.2.12. Captured PC Address at Time of Trap Register

Name: PCTRAP
Offset: 0x002C

Notes:
1. PCTRAP[0] always reads as `0`.
2. If the current IPL is greater or equal to 8, the PC address will not be captured.
3. Hardware update is blocked after the first PCTRAP update occurs, preventing newer traps from overwriting the source address of older ones. An update can be re-enabled by the user attempting to write 24'h000000 to PCTRAP (the write will not occur, preserving PCTRAP contents).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	PCTRAP[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCTRAP[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCTRAP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – PCTRAP[23:0] Captured PC Address at Time of Trap Exception bits(1,2,3)

3.2.13. Force Execution Instruction Register 1

Name: FEX
Offset: 0x0030

Bit	31	30	29	28	27	26	25	24
	FEX[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FEX[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FEX[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FEX[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:0 – FEX[31:0] Force Execution Instruction bits
For two-word operations, FEX contains the first instruction to be executed using the UFEX instruction. FEX is only visible as a R/W register in Debug mode. In all other operating modes, it is read-only for all 0's.

3.2.14. Force Execution Instruction Register 2

Name: FEX2
Offset: 0x0034

Bit	31	30	29	28	27	26	25	24
	FEX2[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FEX2[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FEX2[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FEX2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:0 – FEX2[31:0] Force Execution Instruction 2 bits
For two-word operations, FEX contains the second instruction to be executed using the UFEX instruction. FEX is only visible as a R/W register in Debug mode. In all other operating modes, it is read-only for all 0's.

3.2.15. Debug Hold PC Register

Name: PCHOLD
Offset: 0x0038

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	PCHOLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCHOLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCHOLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – PCHOLD[23:0] Debug Hold PC register bits
PCHOLD is only visible as a R/W register in Debug mode. In all other operating modes, it is read-only for all 0's.

3.2.16. Vector Fail Address Register[(1)]

Name: VFA
Offset: 0x003C

Note:
1. The Reset value of VFA[23:0] must be set to the contents of the Reset vector located at address 0x800000.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VFA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VFA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VFA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – VFA[23:0] Vector Fail Address Register bits

3.2.17. CPU STATUS Register[1]

Name: SR

Note:

1. The CPU STATUS register is not memory mapped. The IPL3 bit is concatenated with the IPL[2:0] bits (SR[7:5]) to form the CPU Interrupt Priority Level.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VF					CTX[2:0]		
Access	R					R	R	R
Reset	0					0	0	0

Bit	15	14	13	12	11	10	9	8
	OA	OB	SA	SB	OAB	SAB		IPL3
Access	R/W	R/W	R/W	R/W	R	R/C		R/C
Reset	0	0	0	0	0	0		0

Bit	7	6	5	4	3	2	1	0
	IPL[2:0]			RA	N	OV	Z	C
Access	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 23 – VF Vector (Fetch) Fail Status bit

Value	Description
1	Indicates to the Bus Error handler that the source of the bus error is a vector fetch. The vector data read will be substituted with the contents of the Vector Fail Address (VFA) SFR.
0	Indicates to the Bus Error handler that the source of the bus error is not a vector fetch.

Bits 18:16 – CTX[2:0] Current (W register) Context Identifier bits

Value	Description
111	Context 7 is currently in use.
110	Context 6 is currently in use.
101	Context 5 is currently in use.
100	Context 4 is currently in use.
011	Context 3 is currently in use.
010	Context 2 is currently in use.
001	Context 1 is currently in use.
000	Context 0 is currently in use.

Bit 15 – OA Accumulator A Fractional Overflow Status bit

Value	Description
1	Accumulator A fractional overflow has occurred (its contents can no longer be represented as a 1.63 fractional value).
0	Accumulator A has not overflowed.

Bit 14 – OB Accumulator B Fractional Overflow Status bit

Value	Description
1	Accumulator B fractional overflow has occurred (its contents can no longer be represented as a 1.63 fractional value).
0	Accumulator B has not overflowed.

Bit 13 – SA Accumulator A Saturation/Sign Overflow ‘Sticky’ Status bit

Value	Description
1	Accumulator A is saturated, or has been saturated at some time or has overflowed into bit 71 (if saturation is disabled).
0	Accumulator A is not saturated or has not overflowed into bit 71 (if saturation is disabled).

Bit 12 – SB Accumulator B Saturation/Sign Overflow ‘Sticky’ Status bit

Value	Description
1	Accumulator B is saturated or has been saturated at some time or has overflowed into bit 71 (if saturation is disabled).
0	Accumulator B is not saturated or has not overflowed into bit 71 (if saturation is disabled).

Bit 11 – OAB Combined Accumulator A or Accumulator B Fractional Overflow Status bit

Value	Description
1	Accumulators A or B fractional overflow has occurred (one or both of their contents can no longer be represented as a 1.63 fractional value).
0	Neither Accumulators A nor B have overflowed.

Bit 10 – SAB Combined Accumulator A or Accumulator B "Sticky" Status bit

Value	Description
1	Accumulators A or B are saturated, or have been saturated at some time or have overflowed into bit 71 (if saturation is disabled).
0	Neither Accumulator A nor B are saturated or have overflowed into bit 71 (if saturation is disabled).

Bit 8 – IPL3 MS-bit of CPU Priority Level Nibble bit
User mode: This bit is R/C-0 (read-only if Supervisor mode supported) and will reset to 0b0.
Supervisor mode: This bit is R/C-0 (CPU will reset into Supervisor mode).

Value	Description
1	CPU Priority ≥ 8 (trap exception underway).
0	CPU Priority < 8 (no trap exception underway).

Bits 7:5 – IPL[2:0] CPU Interrupt Priority Level status bits
User mode: This bit is R/C-0 (read-only if Supervisor mode supported) and will reset to 0b0.
Supervisor mode: This bit is R/C-0 (CPU will reset into Supervisor mode).

Value	Description
111	All interrupts disabled.
110	Level 7 interrupts enabled.
101	Level 6 and 7 interrupts enabled.
100	Level 5 through 7 interrupts enabled.
011	Level 4 through 7 interrupts enabled.
010	Level 3 through 7 interrupts enabled.
001	Level 2 through 7 interrupts enabled.
000	Level 1 through 7 interrupts enabled.

Bit 4 – RA REPEAT Loop Active bit

Value	Description
1	REPEAT loop is in progress.

Value	Description
0	REPEAT loop is not in progress.

Bit 3 – N MCU ALU Negative bit

Value	Description
1	Result was negative.
0	Result was not negative (zero or positive).

Bit 2 – OV MCU ALU Overflow bit

This bit is used for signed arithmetic (two's complement). It indicates an overflow of the magnitude that causes the sign bit to change state.

Value	Description
1	Overflow occurred for signed arithmetic (in this arithmetic operation).
0	No overflow occurred.

Bit 1 – Z ALU 'Sticky' Zero bit

Value	Description
1	An operation which affects the Z bit has set it at some time in the past.
0	The most recent operation which affects the Z bit has cleared it (i.e., a non-zero result).

Bit 0 – C ALU Carry/$\overline{\text{Borrow}}$ bit

SR[31:0] is stacked during exception processing, preserving context.

3.3. Operation

3.3.1. Instruction Set

The dsPIC33A instruction set has two classes of instructions: MCU instructions and DSP instructions. These two classes are seamlessly integrated into the architecture and execute from a single execution unit. The instruction supports integer, fixed point and floating-point math operations.

3.3.2. Data Space Addressing

The Data Space is split into two blocks as X and Y data memory. Each memory block has its own independent Address Generation Unit (AGU). The MCU class of instructions operates solely through the X memory AGU, which accesses the entire memory map as one linear data space. Certain DSP instructions operate through the X and Y AGUs to support dual operand reads, which split the data address space into two parts.

In dsPIC33A devices, overhead-free circular buffers (Modulo Addressing mode) are supported in both X and Y address spaces. The Modulo Addressing removes the software boundary checking overhead for DSP algorithms. The X AGU Circular Addressing can be used with any of the MCU class of instructions. The X AGU also supports the Bit-Reversed Addressing mode to greatly simplify input or output data reordering for radix-2 FFT algorithms.

3.3.3. Addressing Modes

The CPU supports up to eight addressing modes as shown in Table 3-1.

Table 3-1. MCU Instruction Addressing Mode Definitions

Function (Source, ppp)	Function (Destination, qqq)	Description
EA = [Ws + Wb]	EA = [Wd + Wb]	Indirect with (signed) register offset
EA = SR	EA = SR	Status register direct
EA = ++Ws	EA = ++Wd	Register indirect pre-incremented
EA = --Ws	EA = --Wd	Register indirect pre-decremented

Table 3-1. MCU Instruction Addressing Mode Definitions (continued)

Function (Source, ppp)	Function (Destination, qqq)	Description
EA = [Ws++]	EA = [Wd++]	Register indirect post-incremented
EA = [Ws--]	EA = [Wd--]	Register indirect post-decremented
EA = [Ws]	EA = [Wd]	Register indirect
EA = Ws	EA = Wd	Register direct

Each instruction is associated with a predefined addressing mode group, depending upon its functional requirements. For most instructions, the dsPIC33A CPU can execute all of the following functions in a single instruction cycle:

- Data memory read
- Working register (data) read
- Data memory write
- Program (instruction) memory read

As a result, three-operand instructions can be supported, allowing A + B = C operations to be executed in a single cycle.

3.3.4. Programmer's Model

The programmer's model for the dsPIC33A CPU is shown in Figure 3-2. All registers in the programmer's model are memory-mapped and can be manipulated directly by instructions. Table 3-2 provides a description of each register in the programmer's model.

In addition to the registers contained in the programmer's model, the dsPIC33A devices contain control registers for Modulo Addressing, Bit-Reversed Addressing and Interrupts. These registers are described in subsequent sections of this document.

All registers associated with the programmer's model are shown in Figure 3-2.

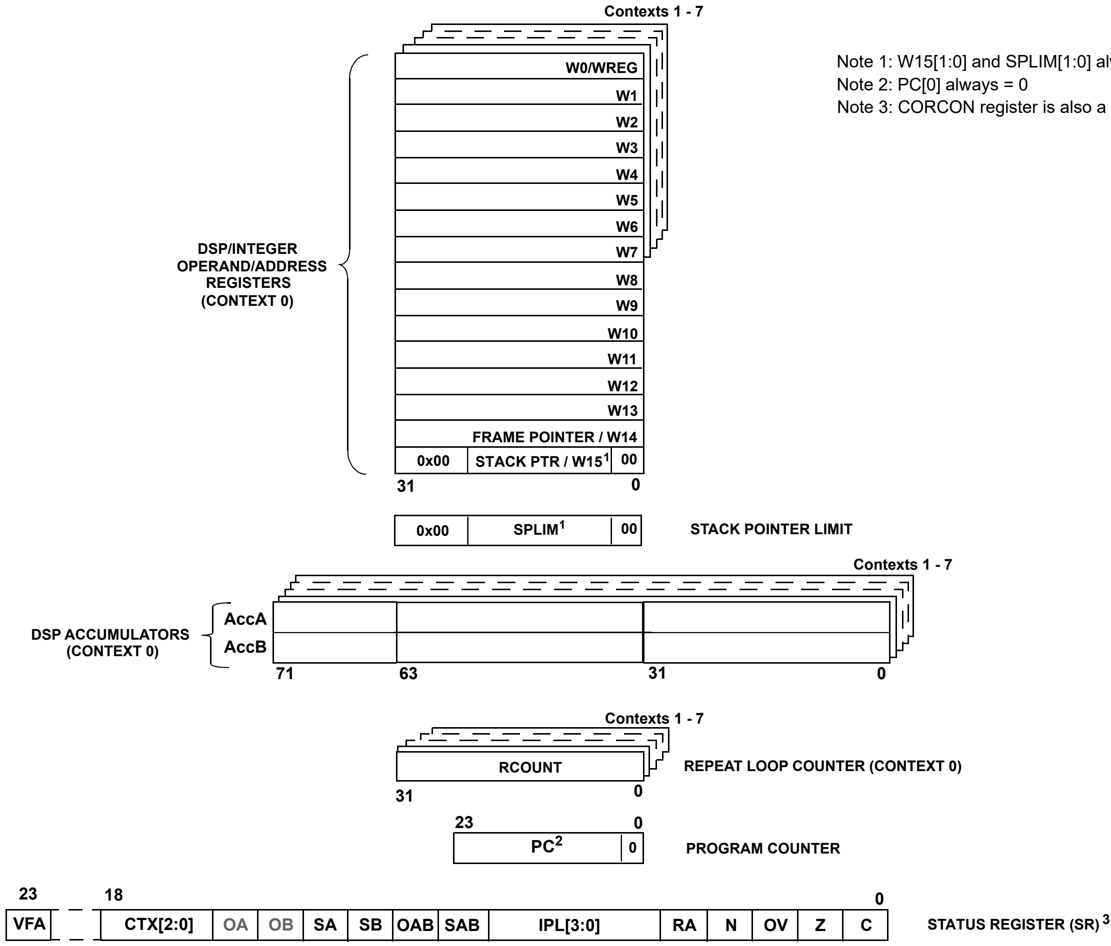
Table 3-2. Programmer's Model Register Descriptions

Register(s) Name	Description
W0 through W15[(1)]	Working Register Array (Default Context)
W0 through W7[(1,2)]	Working Register Array (Alternate Context 1-7)
ACCA,ACCB[(1)]	72-bit DSP Accumulators (Context 0-7)
PC	24-bit Program Counter
SR[(1)]	ALU and DSP Engine Status Register
SPLIM	Stack Pointer Limit Value Register
RCOUNT	`32-bit REPEAT` Loop Count Register (Context 0-7)
CORCON	DSP Engine Configuration

Notes:

1. W0 through W15, ACCx and SR are not mapped to memory.
2. W0 through W7 are part of Alternate W register sets.

Figure 3-2. dsPIC33A CPU Programmer's Model



3.3.5. DSP Engine and Instructions

The DSP engine features:

- A high-speed, 33-bit by 33-bit multiplier
- A 72-bit ALU
- Two 72-bit saturating accumulators
- A 72-bit bidirectional barrel shifter, capable of shifting a 40-bit value up to 32 bits right or up to 32 bits left, in a single cycle

The DSP instructions operate seamlessly with all other instructions and are designed for optimal real-time performance. The `MAC` instruction and other associated instructions can concurrently fetch two data operands from memory while multiplying two W registers. This requires that the data space be split for these instructions and linear for all others.

3.3.6. Exception Processing

The dsPIC33A devices have a vectored exception scheme. Each interrupt source can be assigned to one of seven priority levels.

In addition, each of the Alternate W register contexts can be associated with its own Interrupt Priority Level (IPL) for exception handling. See Alternate Working Register Arrays for more information.

3.3.7. CPU Register Descriptions

3.3.7.1. SR: CPU STATUS Register

The dsPIC33A CPU has a 32-bit STATUS Register (SR). A detailed description of the CPU SR is shown in SR.

SR contains:

- All ALU Operation Status flags
- The CPU Interrupt Priority Level Status bits, IPL[3:0]
- The `REPEAT` Loop Active Status bit, RA (SR[4])
- The DSP Adder/Subtracter Status bits

The SR bits are readable/writable with the following exceptions:

- The RA bit (SR[4]) is read-only.
- The OA, OB (SR[15:14]), OAB (SR[11]), SA, SB (SR[13:12]) and SAB (SR[10]) bits are readable and writable; however, once set, they remain set until cleared by the user application, regardless of the results from any subsequent DSP operations.
  Note: Clearing the SAB bit also clears both the SA and SB bits. Similarly, clearing the OAB bit also clears both the OA and OB bits. A description of the STATUS Register bits affected by each instruction is provided in the *"dsPIC33A Programmer's Reference Manual" (*DS70005540*)*.
- The CTX bit (SR[18:16]) is read-only; it reflects which W register context is currently in use by the CPU.
- The VF bit (SR[23]) is read-only.

3.3.7.2. CORCON: Core Control Register

A detailed description of the CPU CORCON is shown in CORCON.

CORCON contains

- Unsigned or Signed Multiplier Mode Select bit
- Accumulator A and B Saturation Enable bits
- Data Space Write from DSP Engine Saturation Enable bit
- Accumulator Saturation Mode Select bit
- Rounding Mode Select bit
- Integer or Fractional Multiplier Mode Select bit

The CORCON bits are all readable/writable.

3.3.8. Working Register Array

The Working (W) registers can function as data, address or address offset registers. The function of a W register is determined by the addressing mode of the instruction that accesses it.

The dsPIC33A instruction set can be divided into two instruction types: register instructions and file register instructions.

3.3.8.1. Register Instructions

Register instructions can use each W register as a data value or an address offset value. Example 3-1 shows register instructions.

Example 3-1. Register Instructions

```
MOV.l    W0, W1              ; move contents of W0 to W1
MOV.l    W0, [W1]            ; move W0 to address contained in W1
```

```
ADD.l    W0, [W4], W5          ; add contents of W0 to contents pointed
                               ; to by W4. Place result in W5.
```

3.3.8.2. File Register Instructions

File register instructions operate on a specific memory address contained in the instruction opcode and register, W0. W0 is a special Working register used in File register instructions.

The File register address space is determined by the maximum address range of the file instructions, which is either 64 KB (if a W-reg operand is required) or 1 MB (if no W-reg operand is required), and encompasses the user RAM area and Special Function Registers (SFRs) within DS.

Example 3-2 shows File register instructions.

Example 3-2. File Register Instructions

```
ADD.w 0x4500, Wn                 ; (0x4500)+w0 -> 0x4500
ADD.w 0x4500, w0, Wn                       ; (0x4500)+w0 -> 0x4500
ADD.w 0x4500, w4, Wn                       ; (0x4500)+w4 -> 0x4500
```

3.3.8.3. W Register Memory Mapping

The W registers are not memory-mapped, and thus, it is not possible to access a W register in a File register instruction. This helps in eliminating data hazards.

3.3.8.4. W Registers and Byte Mode Instructions

Byte instructions that target the W register array affect only the Least Significant Byte (LSB) of the target register, while word instructions that target the W register array affects only the Least Significant Word (LSW) or the bottom 16 bits of the target register. Since the working registers are not memory-mapped, only the LSB and LSW of these registers are accessible through Byte mode and Word mode instructions, respectively, using Register Direct Addressing only.

3.3.9. Alternate Working Register Arrays

Alternate Working register arrays are a subset of the Working registers (W0 through W7). Depending on the specific device, up to seven Alternate Working register arrays may be implemented. Each set implements registers W0 through W7, AccA, AccB, RCOUNT and DSP related CORCON control bits (US, SATA, SATB, SATDW, ACCSAT, RND, IF). The Alternate W registers are not memory-mapped to data memory space just like the default W array.

All W register arrays are persistent; that is, the contents of the default and Alternate W registers do not change whenever the CPU switches to another set. This saves time by reducing the amount of saving and restoring of register contents, making this very useful for time-critical applications.

Each Alternate W array is inherently assigned to a respective IPL (e.g., IPL4 is assigned to Context 4) and Interrupt Service Routine (ISR) in the application code. The Current Context Identifier (CTX[2:0]) status field is located within the Status Register (SR). Each context is associated with a specific Interrupt Priority Level (IPLV). The context is exited during execution of RETFIE instruction of the interrupt ISR.

During exception processing, the (CTX[2:0]) status field located within the Status Register (SR) is stacked. The stacked SR.CTX[2:0] represents the CPU register context in use at the time of the exception. The value is updated whenever the register context is changed, either through automatic interrupt-based hardware switching or as the result of a context change brought about by the execution of a CTXTSWP{W} instruction.

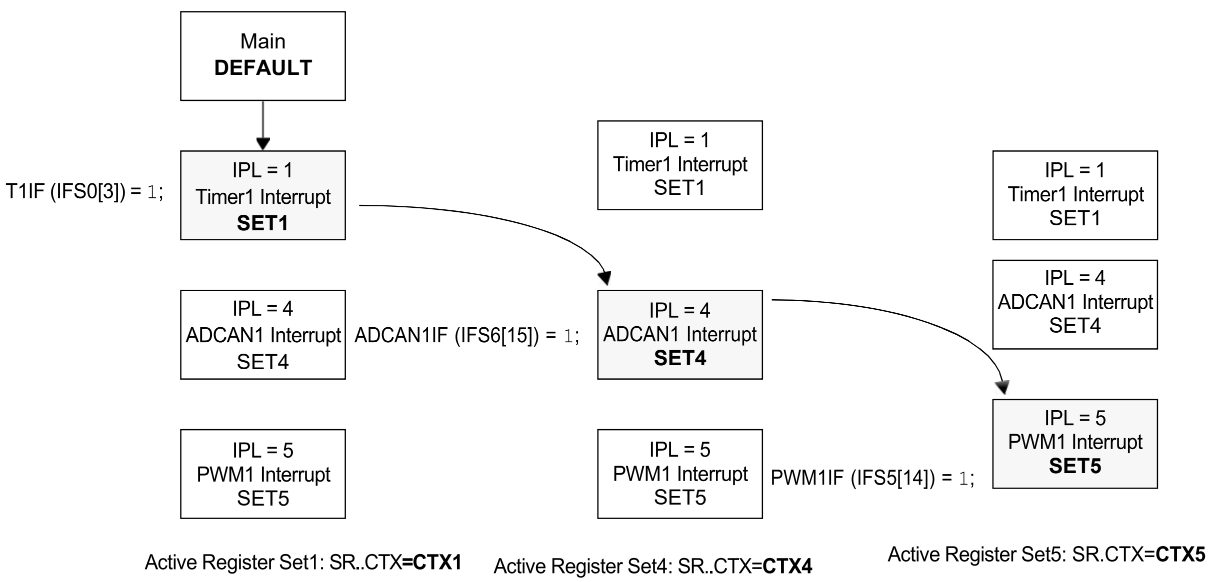
Depending on the device, different context Working register behavior can be observed with nested interrupts. Consider the example, as shown in Figure 3-3, where there are nested interrupts. In this case, the system is configured as follows.

- Timer1 interrupt with an Interrupt Priority Level (IPL) of 1. The Alternate Working Register Set 1 (CTX1) has an IPL of 1.
- ADCAN1 interrupt with an IPL of 4. The Alternate Working Register Set 4 (CTX4) has an IPL of 4.
- PWM1 interrupt with an IPL of 5. The Alternate Working Register Set 5 (CTX5) has an IPL of 5.

The application begins in the main function. At some point in time, the Timer1 interrupt flag is set and the program jumps to the Timer1 ISR. The register set switches from the default Working register set 0 to the Alternate Working register set 1, CTX1. At some point during the Timer1 ISR, the ADCAN1 conversion completes, and its interrupt flag is set. Because it has a higher IPL, the program jumps to the ADCAN1 ISR. The register set switches from the set 1, CTX1 Alternate Working register set to the Alternate Working register set 4, CTX4. At some point during the ADCAN1 ISR, the PWM1 interrupt flag is set. Because the PWM1 IPL is higher than the ADCAN1 IPL, the program jumps to the PWM1 ISR and remains in the Alternate Working register set 5 CTX5.

Once the PWM ISR execution is completed, the program jumps back to the ADCAN1 ISR using CTX4. Similarly, after the execution of the ADCAN1 ISR, the program jumps back to the Timer1 ISR using CTX1. Exceptions above IPL7 (i.e., traps) will execute in whatever register context the CPU was in prior to the trap event.

Figure 3-3. Nested Interrupt Context Flow



3.3.9.1. Alternate Working Register Set

Alternatively, before enabling interrupts associated with a particular context, the application may manually switch to it by executing the `CTXTSWP` instruction. `CTXTSWP` does not affect the CPU IPL; it is used to support software context switching for either context initialization or run-time usage of contexts within procedure calls, thus operating independently from the interrupt system.

3.3.10. Software Stack Pointer

The W15 register serves as a dedicated Software Stack Pointer (SSP) and is automatically modified by exception processing, subroutine calls and returns; however, W15 can be referenced by any instruction in the same manner as all other W registers. This simplifies reading, writing and manipulating the Stack Pointer (for example, creating stack frames).

Note: To protect against misaligned stack accesses, W15[1:0] is fixed to '`00`' by the hardware.

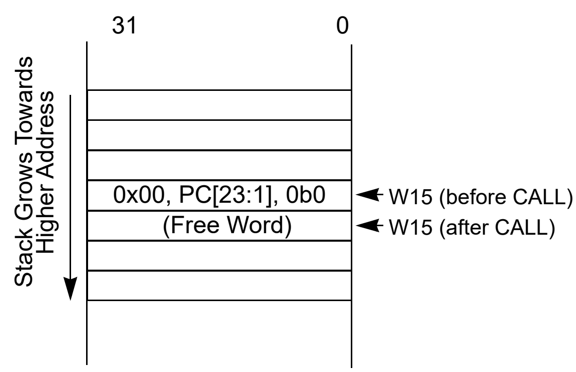
W15 is initialized to 0x4000 during all Resets. This address ensures that the Software Stack Pointer points to valid RAM in all devices and permits stack availability for non-maskable trap exceptions.

These can occur before the SSP is initialized by the user software. Reprogramming the SSP to any location within the data space is possible during initialization.

The Software Stack Pointer always points to the first available free word in the data space (RAM) and fills the software stack, working from lower addresses toward higher ones. Figure 3-4 illustrates how it pre-decrements for a stack pop (read) and post-increments for a stack push (write).

When the PC is pushed onto the stack, PC[23:0] are pushed onto the first available stack word, as shown in Figure 3-4.

Figure 3-4. Stack Operation for a `CALL` Instruction



3.3.10.1. Software Stack Examples

The software stack is manipulated using the `PUSH` and `POP` instructions. The `PUSH` and `POP` instructions are the equivalent of a `MOV` instruction with W15 as the Destination Pointer. For example, the contents of W0 can be pushed onto the stack by:

```
PUSH W0
```

This syntax is equivalent to:

```
MOV.L W0,[W15++]
```

The contents of the Top-of-Stack (TOS) can be returned to W0 by:

```
POP W0
```

This syntax is equivalent to:

```
MOV.L [--W15],W0
```

Figure 3-5 through Figure 3-8 illustrate examples of how the software stack is used. Figure 3-5 illustrates the software stack at device initialization. W15 has been initialized to 0x00004000. This example assumes the values, 0xAAAAAAAA and 0xBBBBBBBB, have been written to W0 and W1, respectively. In Figure 3-6, the stack is pushed for the first time, and the value contained in W0 is copied to the stack. W15 is automatically updated to point to the next available stack location (0x00004004). In Figure 3-7, the contents of W1 are pushed onto the stack. Figure 3-8 illustrates how the stack is popped and the Top-of-Stack value (previously pushed from W1) is written to W3.

Figure 3-5. Stack Pointer at Device Reset

Figure 3-6. Stack Pointer After the First `PUSH` Instruction

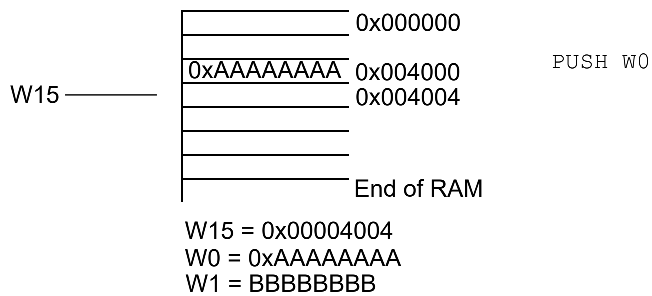


Figure 3-7. Stack Pointer After the Second `PUSH` Instruction

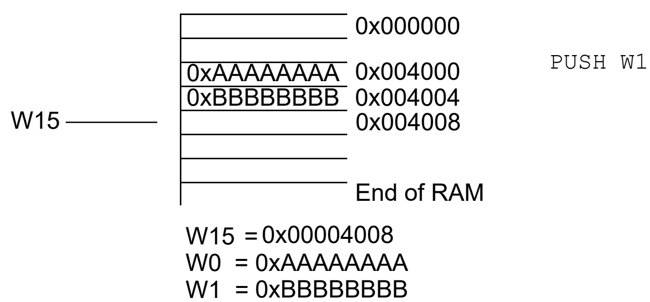
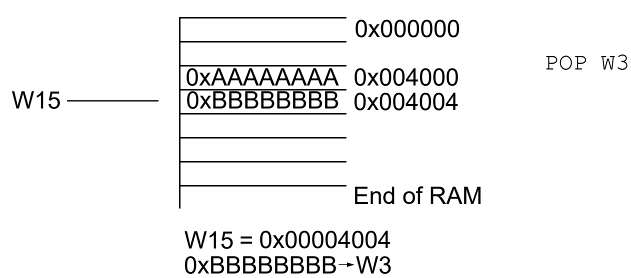


Figure 3-8. Stack Pointer After a `POP` Instruction



3.3.10.2. W14 Software Stack Frame Pointer

A frame is a user-defined section of memory in the stack that is used by a single function. The Working register, W14, can be used as a Stack Frame Pointer with the `LNK` (link) and `ULNK` (unlink) instructions. W14 can be used in a normal Working register by instructions when it is not used as a Frame Pointer.

3.3.10.3. Stack Pointer Overflow

The Stack Pointer Limit (SPLIM) register specifies the size of the stack buffer. SPLIM is a 32-bit register, but SPLIM[1:0] is fixed to '`00`' because all stack operations must be long word-aligned.

The stack overflow check is not enabled until a long word write to SPLIM occurs. After this, it can only be disabled by a device Reset. All Effective Addresses (EAs), generated using W15 as a source or destination, are compared against the value in SPLIM. If Effective Addresses (EAs) exceed the contents of the SPLIM register, and a `PUSH` operation is performed, a stack error trap occurs on a subsequent `PUSH` operation. For example, if it is desirable to cause a stack error trap when the stack grows beyond address 0x5000 in RAM, initialize the SPLIM with the value 0x4FFC.
Note: A stack error trap can be caused by any instruction that uses the contents of the W15 register to generate an Effective Address (EA). Therefore, if the contents of W15 are greater than the contents of the SPLIM register by a value of four, and a `CALL` instruction is executed, or if an interrupt occurs, a stack error trap is generated.

If stack overflow checking is enabled, a stack error trap also occurs if the W15 Effective Address calculation wraps over the end of data space.

A pre/post inc/dec operation is performed on W15 that results in EA[1:0] != 2'b00 (i.e., not long word aligned). This will detect byte and word pre/post inc/dec operations that are otherwise considered aligned but would result in a misaligned Stack Pointer.

Note: A write to the SPLIM should not be followed by an indirect read operation using W15.

3.3.10.4. Stack Pointer Underflow

The stack is initialized to 0x4000 during a Reset. A stack error trap is initiated if the Stack Pointer address is less than 0x4000.

Note: Locations in data space between 0x0000 and 0x3FFF are, in general, reserved for core and peripheral Special Function Registers (SFRs).

3.3.11. Arithmetic Logic Unit (ALU)

The dsPIC33A ALU is 32 bits wide and is capable of addition, subtraction, single-bit shifts and logic operations. Unless otherwise mentioned, arithmetic operations are a two's complement in nature. Depending on the operation, the ALU can affect the values of the following bits in the STATUS Register:

- Carry (C)
- Zero (Z)
- Negative (N)
- Overflow (OV)

The ALU can perform 8/16-bit or 32-bit operations, depending on the mode of the instruction that is used. Data for the ALU operation can come from the W register array or data memory depending on the addressing mode of the instruction. Likewise, output data from the ALU can be written to the W register array or a data memory location.

Note: Byte operations use the 16-bit ALU and can produce results in excess of eight bits. However, to maintain backward compatibility with PIC® MCU devices, the ALU result from all byte operations is written back as a byte (i.e., the MSB is not modified) and the STATUS Register is updated based only upon the state of the LSB of the result.

3.3.11.1. Byte to Word Conversion

The dsPIC33AK256MPS306 CPU has two instructions that are helpful when mixing 8-bit, 16-bit and 32-bit ALU operations.

The Sign-Extend (`SE`) instruction takes a byte or word value in a W register or data memory and creates a sign-extended long word value that is stored in a W register.

The Zero-Extend (ZE) instruction takes a byte or word value in a W register or data memory and clears the 24 or 16 MSbs of a long word in a destination W register for an extension from 8 bits or 16 bits, respectively.

3.3.12. DSP Engine

The DSP engine is a block of hardware that is fed data from the W register array, but contains its own specialized result registers. The DSP engine is driven from the same instruction decoder that directs the MCU ALU. In addition, all operand Extended Addresses (EAs) are generated in the W register array. Concurrent operation with MCU instruction flow is not possible, though both the MCU ALU and DSP engine resources can be shared by all instructions in the instruction set.

The DSP engine consists of the following components:

- High-speed, 33-bit by 33-bit multiplier
- Barrel shifter
- 72-bit adder/subtractor
- Two target Accumulator registers
- Rounding logic with selectable modes
- Saturation logic with selectable modes

Data input to the DSP engine is derived from one of the following sources:

- Directly from the W array for dual source operand DSP instructions; Data values fetched via the X and Y memory data buses.
- From the X memory data bus for all other DSP instructions

Data output from the DSP engine is written to one of the following destinations:

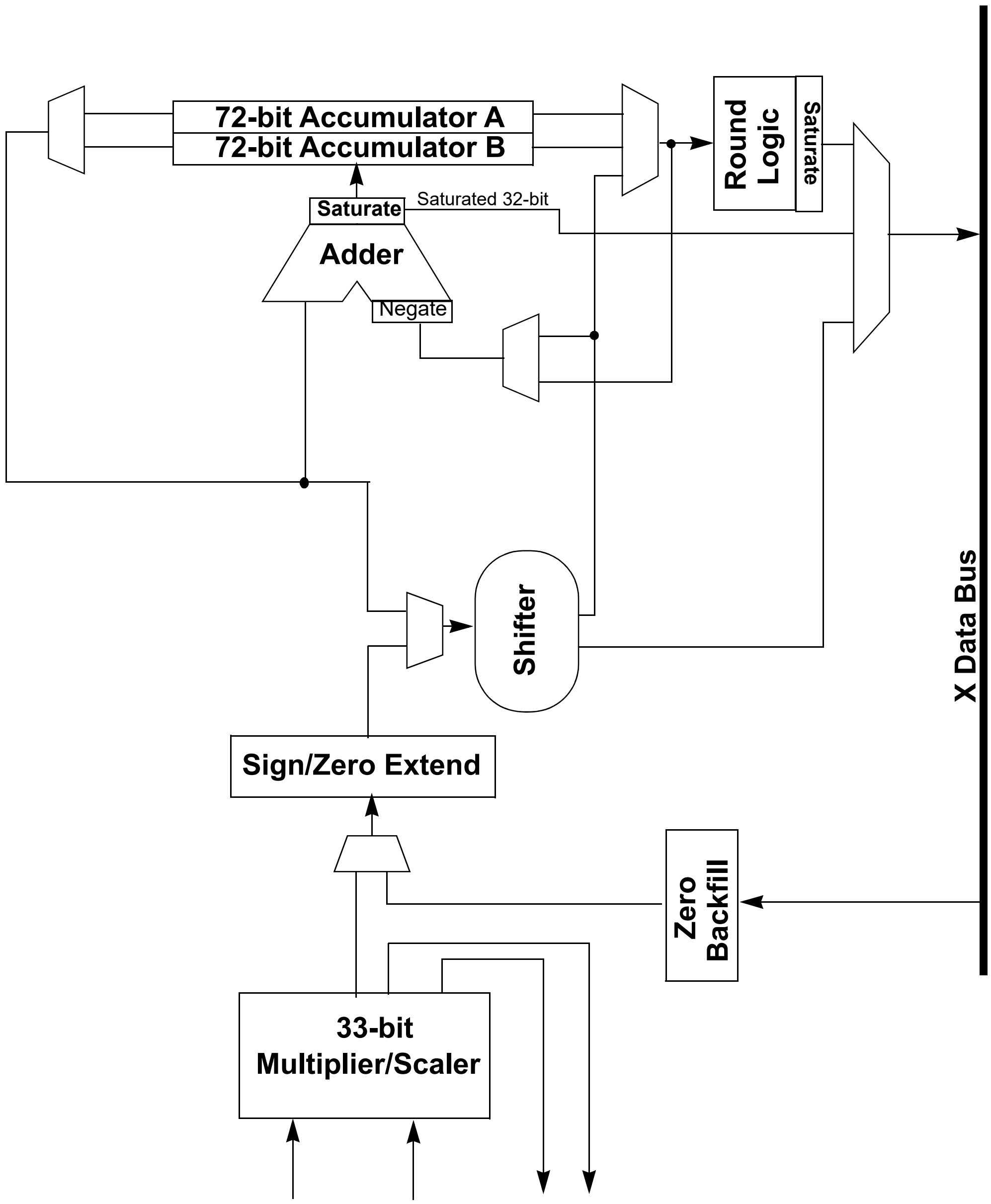
- The target accumulator, as defined by the DSP instruction being executed
- The X memory data bus to any location in the data memory address space

The DSP engine can perform inherent accumulator-to-accumulator operations that require no additional data.

The MCU shift and multiply instructions use the DSP engine hardware to obtain their results. The X memory data bus is used for data reads and writes in these operations.

Figure 3-9 illustrates a block diagram of the DSP engine.

Figure 3-9. DSP Engine Block Diagram



3.3.12.1. Data Accumulators

Two 72-bit data accumulators, ACCA and ACCB, are the Result registers for the DSP instructions listed in DSP Multiply Instructions. Each accumulator is not memory-mapped and is referred to as these three registers, where 'x' denotes the particular accumulator:

- ACCxL: ACCx[31:0]
- ACCxH: ACCx[63:32]
- ACCxU: ACCx[71:64]

For fractional operations that use the accumulators, the radix point is located to the right of bit 31. The range of fractional values that can be stored in each accumulator is -256 to +(256 - 2^{-63}).

For integer operations that use the accumulators, the radix point is located to the right of bit 0. The range of integer values that can be stored in each accumulator is -0x80_0000_0000_0000_0000 to 0x7F_FFFF_FFFF_FFFF_FFFF.

3.3.12.2. Multiplier

The dsPIC33A devices feature a 33-bit-by-33-bit multiplier shared by both the MCU ALU and the DSP engine. The multiplier is capable of a signed, unsigned or mixed-sign operation and supports either 9.31 fractional (Q.31) or 64-bit integer results.

The multiplier takes in 32-bit input data and converts the data to 33 bits. Signed operands to the multiplier are sign-extended. Unsigned input operands are zero-extended. The internal 33-bit representation of data in the multiplier allows the correct execution of mixed-sign and unsigned 32-bit by 32-bit multiplication operations.

The representation of data in hardware for Integer and Fractional Multiplier modes is as follows:

- Integer data is inherently represented as a signed two's complement value, where the Most Significant bit (MSb) is defined as a sign bit. Generally speaking, the range of an N-bit two's complement integer is $-2^{(N-1)}$ to $2^{(N-1)}-1$.
- Fractional data is represented as a two's complement fraction, where the MSb is defined as a Sign bit, and the radix point is implied to lie just after the Sign bit (Q.X format). The range of an N-bit two's complement fraction with this implied radix point is -1.0 to $(1 - 2^{(1-N)})$.

The range of data in both Integer and Fractional modes is listed in Table 3-3. Figure 3-10 and Figure 3-11 illustrate how the multiplier hardware interprets data in Integer and Fractional modes.

The Integer or Fractional Multiplier Mode Select (IF) bit (CORCON[0]) determines the integer/fractional operation for the instructions listed in Table 3-4. The IF bit does not affect MCU multiply instructions listed in Table 3-5, which are always integer operations. The multiplier scales the result one bit to the left for fractional operation. The LSb of the result is always cleared. The multiplier defaults to Fractional mode for DSP operations at a device Reset.

Table 3-3. dsPIC33A Data Ranges

Register Size	Integer Range	Fraction Range	Fraction Resolution
16-Bit	-32768 to 32767	-1.0 to $(1.0 - 2^{-15})$ (Q1.15 Format)	3.052×10^{-5}
32-Bit	-2,147,483,648 to 2,147,483,647	-1.0 to $(1.0 - 2^{-31})$ (Q1.31 Format)	4.657×10^{-10}
64-Bit	-9.223372037e18 to 9.223372037e18	-1.0 to $(1.0 - 2^{-63})$ (Q.1.63 Format)	1.08420×10^{-19}
72-Bit	-2.361183241e21 to 2.361183241e21	-256.0 to $(256.0 - 2^{-63})$ (Q.9.63 Format with 8 Guard bits)	1.08420×10^{-19}

Figure 3-10. Integer and Fractional Representation of 0x40000001

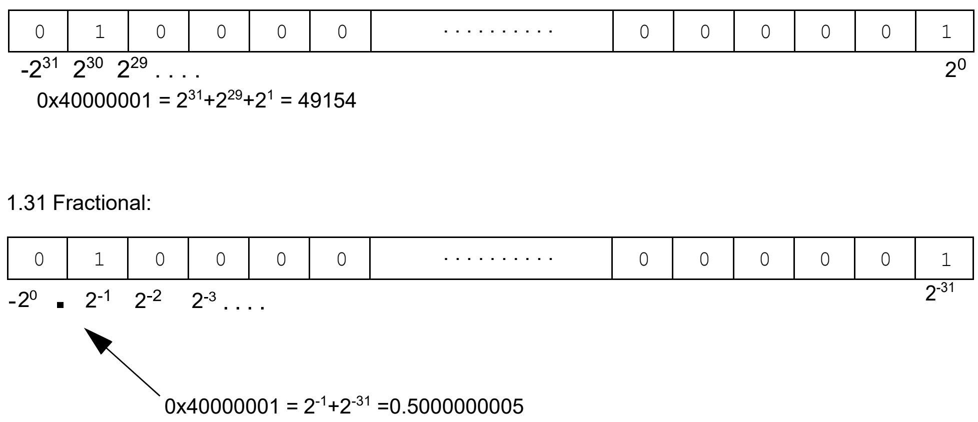
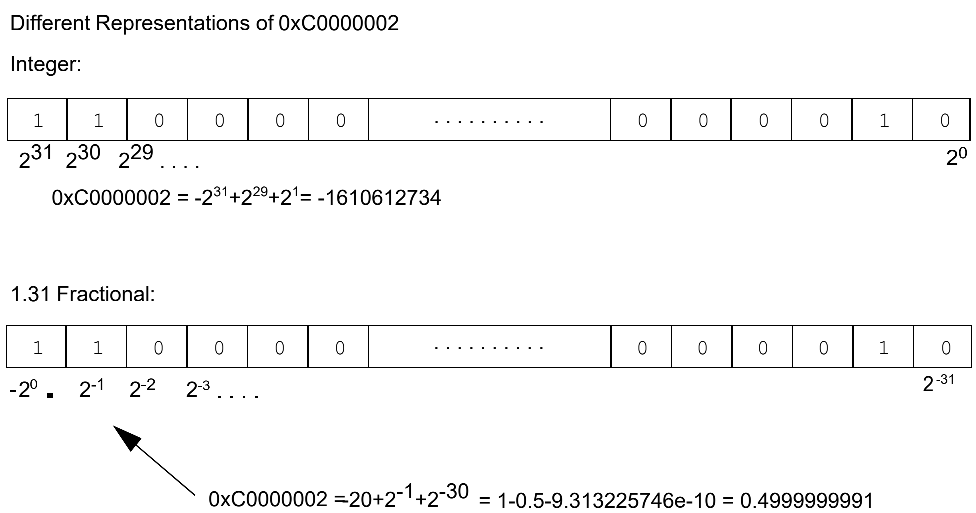


Figure 3-11. Integer and Fractional Representation of 0xC0000002



3.3.12.2.1. DSP Multiply Instructions

The DSP instructions that use the multiplier are summarized in Table 3-4.

Table 3-4. DSP Instructions that Use the Multiplier

DSP Instruction[(1)]	Description	Algebraic Equivalent
MAC	Multiply and Add to Accumulator or Square and Add to Accumulator	$a = a + b * c$ or $a = a + b^2$
MSC	Multiply and Subtract from Accumulator	$a = a - b * c$
MPY	Multiply	$a = b * c$
MPY.N	Multiply and Negate Result	$a = -b * c$

Note:

1. DSP instructions using the multiplier can operate in Fractional (1.15/1.31) or Integer modes.

Table 3-4. DSP Instructions that Use the Multiplier (continued)

DSP Instruction[1]	Description	Algebraic Equivalent
`SQR`	Square to Accumulator	$a = b^2$
`SQRAC`	Square and Accumulate	$a = a + (b^2)$
`ED`	Partial Euclidean Distance	$a = (b – c)^2$
`EDAC`	Add Partial Euclidean Distance to the Accumulator	$a = a + (b – c)^2$

Note:

1. DSP instructions using the multiplier can operate in Fractional (1.15/1.31) or Integer modes.

The DSP Multiplier Unsigned/Signed Control (US bit (CORCON[12])) determines whether the DSP multiply instructions are signed (default), unsigned or mixed-sign. The US[1:0] bit does not influence the MCU multiply instructions, which have specific instructions for a signed or unsigned operation. If US = `1`, the input operands for instructions shown in Table 3-4 are considered as unsigned values, which are always zero-extended into the 33rd bit of the multiplier value. If the US = `0`, the operands are sign-extended.

3.3.12.2.2. MCU Multiply Instructions

The DSP multiplier is also used to support all MCU multiply instructions, which include signed, unsigned or mixed signed/unsigned operations. All instructions support both Word and Long Word operands. All literals are either sign or zero-extended to the operand size prior to use as appropriate for the particular instruction. Some instructions use W-regs for the result destinations, while others can target ACCA or ACCB, using the same result data paths as the DSP multiply instructions. The instructions targeting the accumulator can be either integer or fractional operations, irrespective of the state of the DSP engine fractional/integer control bit at CORCON.IF. This removes the need for software to test/control the CORCON.IF bit. All MCU multiply instructions (other than MUL.x, which are by definition unsigned integer operations) explicitly identify the signed or unsigned characteristic of each operand. The result for all signed and mixed signed/unsigned multiply operations is always sign-extended to the MSb of the target register. Unsigned multiply operations are zero-extended to the MSb of the target accumulator.

Table 3-5. MCU Instructions that Utilize the Multiplier

MCU Instruction[1]	Description
`MUL.{b/w/l}`	Unsigned integer multiplication of Wn and the file register; results written back to default destination register W2.
`MULSS.{w/l/d}`	Word or Long Word integer multiplication of two signed integers; stores the results in W-registers.
`MULSU.{w/l/d}`	Word or Long Word integer of signed and unsigned integers; stores the results in W-registers.
`MULUS.{w/l/d}`	Word or Long Word integer of unsigned and signed integers; stores the results in W-registers.
`MULUU.{w/l/d}`	Word or Long Word integer of two unsigned integers; stores the results in W-registers.
`MULFSS.{w/l}`	Word or Long Word fractional multiplication of two signed fractional values; stores the results in the accumulator.
`MULFSU.{w/l}`	Word or Long Word fractional multiplication of a signed and an unsigned fractional value; stores the results in the accumulator.
`MULFUS.{w/l}`	Word or Long Word fractional multiplication of a signed and an unsigned fractional value; stores the results in the accumulator.

Note:

1. MCU instructions using the multiplier operate only in Integer mode.

Table 3-5. MCU Instructions that Utilize the Multiplier (continued)

MCU Instruction(1)	Description
`MULFUU.{w/l}`	Word or Long Word fractional multiplication of two unsigned fractional values; stores the results in the accumulator.
`MULISS.{w/l}`	Word or Long Word integer multiplication of two signed integers; stores the results in the accumulator.
`MULISU.{w/l}`	Word or Long Word integer multiplication of signed and unsigned integers; stores the results in the accumulator.
`MULIUS.{w/l}`	Word or Long Word integer multiplication of signed and unsigned integers; stores the results in the accumulator.
`MULIUU.{w/l}`	Word or Long Word integer multiplication of two unsigned integers and stores the results in the accumulator.

Note:
1. MCU instructions using the multiplier operate only in Integer mode.

3.3.12.3. Data Accumulator Adder/Subtractor

The data accumulators have a 72-bit adder/subtractor with automatic sign extension logic for the multiplier result (if signed). It can select one of two accumulators (A or B) as its pre-accumulation source and post-accumulation destination. For the `ADD` (accumulator) and `LAC` instructions, the data to be accumulated or loaded can optionally be scaled via the barrel shifter prior to accumulation.

The 72-bit adder/subtractor can optionally negate one of its operand inputs to change the sign of the result (without changing the operands). The negation is used during multiply and subtract (`MSC`) or multiply and negate (`MPY.N`) operations.

The 72-bit adder/subtractor has an additional saturation block that controls accumulator data saturation, if enabled.

3.3.12.3.1. Accumulator Status Bits

Six STATUS Register bits that support saturation and overflow are located in the CPU STATUS Register (SR) and are listed in Table 3-6.

Table 3-6. Accumulator Overflow and Saturation Status Bits

Status Bit (SR Location)	Description
OA ([15])	Accumulator A overflowed into guard bits (ACCA[71:64]).
OB ([14])	Accumulator B overflowed into guard bits (ACCB[71:63]).
SA ([13])	ACCA saturated (bit 63 overflow and saturation) or ACCA overflowed into guard bits and saturated (bit 71 overflow and saturation).
SB ([12])	ACCB saturated (bit 63 overflow and saturation) or ACCB overflowed into guard bits and saturated (bit 71 overflow and saturation).
OAB ([11])	OA logically ORed with OB, clearing OAB clears both OA and OB.
SAB ([10])	SA logically ORed with SB, clearing SAB clears both SA and SB.

The OA and OB bits are modified each time data passes through the accumulator add/subtract logic. When set, they indicate that the most recent operation has overflowed into the accumulator guard bits (ACCx[71:64]). This type of overflow is not catastrophic; the guard bits preserve the accumulator data. The OAB Status bit is the logically OR value of OA and OB.

The OA and OB bits, when set, can optionally generate an arithmetic error trap. The trap is enabled by setting the corresponding Overflow Trap Flag Enable bit (OVATE or OVBTE) in Interrupt Control Register 4 (INTCON4[10:9]) in the interrupt controller. The trap event allows the user to take immediate corrective action, if desired.

The SA and SB bits can be set each time data passes through the accumulator saturation logic. Once set, these bits remain set until cleared by the user application. The SAB Status bit indicates the logical OR value of SA and SB. When set, these bits indicate that the accumulator has overflowed its maximum range (bit 63 for 64-bit saturation or bit 71 for 72-bit saturation) and are saturated (if saturation is enabled).

When saturation is not enabled, the SA and SB bits indicate that a catastrophic overflow has occurred (the sign of the accumulator has been destroyed). If the Catastrophic Overflow Trap Enable (COVTE) bit (INTCON4[8]) is set, SA and SB bits will generate an arithmetic error trap when saturation is disabled. The SA and SB bits can be set in software, enabling efficient context state switching.

3.3.12.3.2. Saturation And Overflow Modes

The dsPIC33A CPU supports three Saturation and Overflow modes.

- **Accumulator 71-Bit Saturation**
  In this mode, the saturation logic loads the maximally positive 9.63 value (0x7F_FFFF_FFFF_FFFF_FFFF) or maximally negative 9.63 value (0x80_0000_0000_0000_0000) into the target accumulator. The SA or SB bit is set and remains set until cleared by the user application. This Saturation mode is useful for extending the dynamic range of the accumulator.

  To configure for this mode of saturation, set the Accumulator Saturation Mode Select (ACCSAT) bit (CORCON[4]). Additionally, set the ACCA Saturation Enable (SATA) bit (CORCON[7] and/or the ACCB Saturation Enable (SATB) bit (CORCON[6]) to enable accumulator saturation.
- **Accumulator 63-Bit Saturation**
  In this mode, the saturation logic loads the maximally positive 1.63 value (0x00_7FFF_FFFF_FFFF_FFFF) or maximally negative 1.63 value (0xFF_8000_0000_0000_0000) into the target accumulator. The SA or SB bit is set and remains set until cleared by the user. When this Saturation mode is in effect, the guard bits, 64 through 71, are not used except for sign extension of the accumulator value. Consequently, the OA, OB or OAB bits in SR are never set.

  To configure for this mode of overflow and saturation, the ACCSAT (CORCON[4]) bit must be cleared. Additionally, the SATA (CORCON[7]) and/or SATB (CORCON[6]) bits must be set to enable accumulator saturation.
- **Accumulator Catastrophic Overflow**
  If the SATA (CORCON[7]) and/or SATB (CORCON[6]) bits are not set, then no saturation operation is performed on the accumulator, and the accumulator is allowed to overflow all the way up to bit 71 (destroying its sign). If the Catastrophic Overflow Trap Enable (COVTE) bit (INTCON4[8] in the interrupt controller) is set, a catastrophic overflow initiates an arithmetic error trap.

Accumulator saturation and overflow detection can only result from the execution of a DSP instruction that modifies one of the two accumulators via the 72-bit DSP ALU. Saturation and overflow detection do not take place when the accumulators are accessed via the MCU class of instructions. Furthermore, the Accumulator Status bits shown in Table 3-6 are not modified. However, the MCU Status bits (Z, N, C, OV, DC) will be modified, depending on the MCU instruction that accesses the accumulator.

Accumulator Catastrophic Overflow

If the SATA (CORCON[7]) and/or SATB (CORCON[6]) bits are not set, then no saturation operation is performed on the accumulator, and the accumulator is allowed to overflow all the way up to bit 71 (destroying its sign). If the Catastrophic Overflow Trap Enable (COVTE) bit (INTCON4[8] in the interrupt controller) is set, a catastrophic overflow initiates an arithmetic error trap.

Accumulator saturation and overflow detection can only result from the execution of a DSP instruction that modifies one of the two accumulators via the 72-bit DSP ALU. Saturation and overflow detection do not take place when the accumulators are accessed via the MCU class of instructions. Furthermore, the Accumulator Status bits shown in Table 3-6 are not modified. However, the MCU Status bits (Z, N, C, OV, DC) will be modified, depending on the MCU instruction that accesses the accumulator.

3.3.12.3.3. Data Space Write Saturation

In addition to adder/subtractor saturation, writes to data space can be saturated without affecting the contents of the source accumulator. This feature allows data to be limited, while not sacrificing the dynamic range of the accumulator during intermediate calculation stages. Data space write saturation is enabled by setting the data space write from the DSP Engine Saturation Enable (SATDW) control bit (CORCON[5]). Data space write saturation is enabled by default at a device Reset.

The data space write saturation feature works with the `SAC` and `SACR` instructions. The value held in the accumulator is never modified when these instructions are executed. The hardware takes the following steps to obtain the saturated write result:

1. The read data are scaled based upon the arithmetic shift value specified in the instruction.
2. The scaled data are rounded (`SACR` only).
3. The scaled/rounded value is saturated to a 16-bit result based on the value of the guard bits. For data values greater than 0x7FFF, the data written to memory are saturated to the maximum positive 1.15 value, 0x7FFF. For input data less than 0xFF8000, data written to memory are saturated to the maximum negative 1.15 value, 0x8000.

3.3.12.3.4. Accumulator Write Back

The `MAC` and `MSC` instructions can optionally write a rounded version of the accumulator that is not the target of the current operation into data space memory. The write is performed across the X-bus into the combined X and Y address space. This accumulator write-back feature is beneficial in certain algorithms, such as FFT and LMS filters.

Two addressing modes are supported by the accumulator write-back hardware:

- W0, W1, W2, W3, W13 or W14, Register Direct: The rounded contents of the non-target accumulator are written into the destination register as a 1.15 (Word mode) or 1.31 (Long Word mode) fractional result.
- [W13]+ = 2 or [W15]+ = 2, Register Indirect with Post-Increment: The rounded contents of the non-target accumulator are written into the address pointed to by W13 or W15 as a 1.15 (Word mode) or 1.31 (Long Word mode) fraction. W13 or W15 is then incremented by 2.

3.3.12.4. Round Logic

The round logic can perform a conventional (biased) or convergent (unbiased) round function during an accumulator write (store). The Round mode is determined by the state of the Rounding Mode Select (RND) bit (CORCON[1]). It generates a 16-bit 1.15 or 32-bit 1.31 data value, which is passed to the data space write saturation logic. If rounding is not indicated by the instruction, a truncated 1.15 or 1.31 data value is stored.

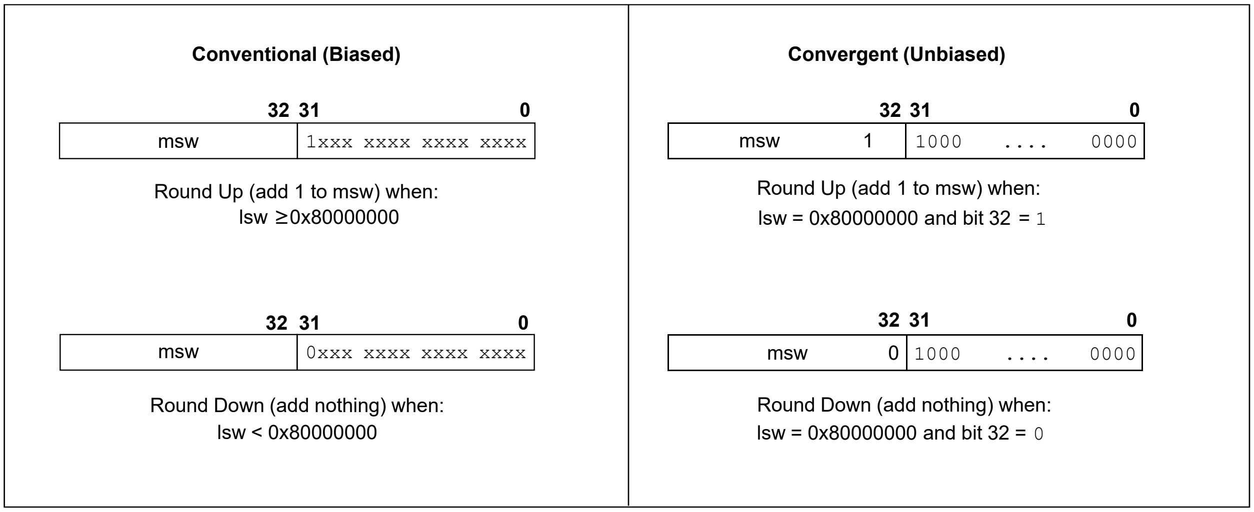
The two Rounding modes are shown in Figure 3-12. Conventional rounding takes bit 31 of the accumulator, zero-extends it and adds it to the most significant word (msw), excluding the guard or overflow bits (bits 32 through 63). If the least significant word (lsw) of the accumulator is between 0x80000000 and 0xFFFFFFFF (0x80000000 included), the msw is incremented. If the lsw of the accumulator is between 0x0000 and 0x7FFFFFFF, the msw remains unchanged. A consequence of this algorithm is that over a succession of random rounding operations, the value tends to be biased slightly positive.

Convergent (or unbiased) rounding operates in the same manner as conventional rounding except when the lsw equals 0x80000000. If this is the case, the LSb of the msw (bit 32 of the accumulator) is examined. If it is '`1`', the msw is incremented. If it is '`0`', the msw is not modified. Assuming that bit 16 is effectively random in nature, this scheme removes any rounding bias that may accumulate.

The `SAC` and `SACR` instructions store either a truncated (`SAC`) or rounded (`SACR`) version of the contents of the target accumulator to data memory via the X-bus (subject to data saturation).

For the MAC class of instructions, the accumulator write-back data path is always subject to rounding. An overflow that occurs as a consequence of a rounding operation will also be subject to saturation.

Figure 3-12. Conventional and Convergent Rounding Modes



3.3.12.5. Barrel Shifter

The barrel shifter can perform up to a 32-bit arithmetic right shift, or up to a 32-bit left shift, in a single cycle. DSP or MCU instructions can use the barrel shifter for multibit shifts.

The shifter requires a signed binary value to determine both the magnitude (number of bits) and direction of the shift operation.

- A positive value shifts the operand right.
- A negative value shifts the operand left.
- A value of '0' does not modify the operand.

The barrel shifter is 72 bits wide to accommodate the width of the accumulators. A 72-bit output result is provided for DSP shift operations, and a 32-bit result is provided for MCU shift operations.

Table 3-7 provides a summary of instructions that use the barrel shifter.

Table 3-7. Instructions that Use the DSP Engine Barrel Shifter

Instruction	Description
ASR	Arithmetic multibit right shift of data memory location
LSR	Logical multibit right shift of data memory location
SL	Multibit shift left of data memory location
SAC	Store DSP accumulator with optional shift
SFTAC	Shift DSP accumulator

3.3.12.6. DSP Engine Mode Selection

These operational characteristics of the DSP engine, discussed in previous sections, can be selected through the CPU Core Configuration register (CORCON).

- Fractional or integer multiply operation
- Conventional or convergent rounding
- Automatic saturation on/off for ACCA

- Automatic saturation on/off for ACCB
- Automatic saturation on/off for writes to data memory
- Accumulator Saturation mode selection

3.3.12.7. DSP Engine Trap Events

Arithmetic error traps that can be generated for handling exceptions in the DSP engine are selected through the Interrupt Control Register 4 (INTCON4). These are:

- Trap on ACCA overflow enable using OVATE (INTCON4[21])
- Trap on ACCB overflow enable using OVBTE (INTCON4[20])
- Trap on catastrophic ACCA and/or ACCB overflow enable using COVTE (INTCON4[19]). Occurrence of the traps is indicated by these error status bits.
  - OVAERR (INTCON4[5])
  - OVBERR (INTCON4[4])
  - COVAERR (INTCON4[3])
  - COVBERR (INTCON4[2])

  An arithmetic error trap is also generated when the user application attempts to shift a value beyond the maximum allowable range (±32 bits) using the `SFTAC` instruction. This trap source cannot be disabled and is indicated by the Shift Accumulator Error Status (SFTACERR) bit (INTCON4[1] in the interrupt controller). The instruction will execute, but the results of the shift are not written to the target accumulator.

3.3.13. Divide Support

The dsPIC33A CPU supports the following types of division operations.

- `DIVF.SD`: 16/16 signed fractional divide
- `DIVF.SD`: 32/16 signed fractional divide
- `DIVF.SD`: 32/32 signed fractional divide
- `DIV.SD`: 32/32 signed divide
- `DIV.UD`: 32/32 unsigned divide
- `DIV.SD`: 32/16 signed divide
- `DIV.UD`: 32/16 unsigned divide
- `DIV.SW`: 16/16 signed divide
- `DIV.UW`: 16/16 unsigned divide

The quotient for all divide instructions can placed in any Working register, Wm. The remainder is placed in W(m+1). The 32/16-bit divisor can be located in any W register. A 32/16-bit dividend can be located in any W register. The integer 16/16 divide instructions will either zero or sign extend the least significant dividend word into the most significant dividend word during the first iteration to create a 32-bit dividend.

All 16-bit/16-bit and 32-bit/16-bit divide instructions are iterative operations and must be executed six times within a `REPEAT` loop. All 32-bit/32-bit divide instructions are iterative operations and must be executed 10 times within a `REPEAT` loop.

The developer is responsible for programming the `REPEAT` instruction. A complete divide operation takes seven or 11 instruction cycles to execute.

The divide flow is interruptible, just like any other `REPEAT` loop. All data are restored into the respective data registers after each iteration of the loop, so the user application is responsible for saving the appropriate W registers in the ISR. Although they are important to the divide hardware, the intermediate values in the W registers have no meaning to the user application. The divide instructions must be executed seven or 11 times in a `REPEAT` loop to produce a meaningful result.

A divide-by-zero error generates a math error trap. This condition is indicated by the Arithmetic Error Status (DIV0ERR) bit (INTCON4[6] in the interrupt controller).

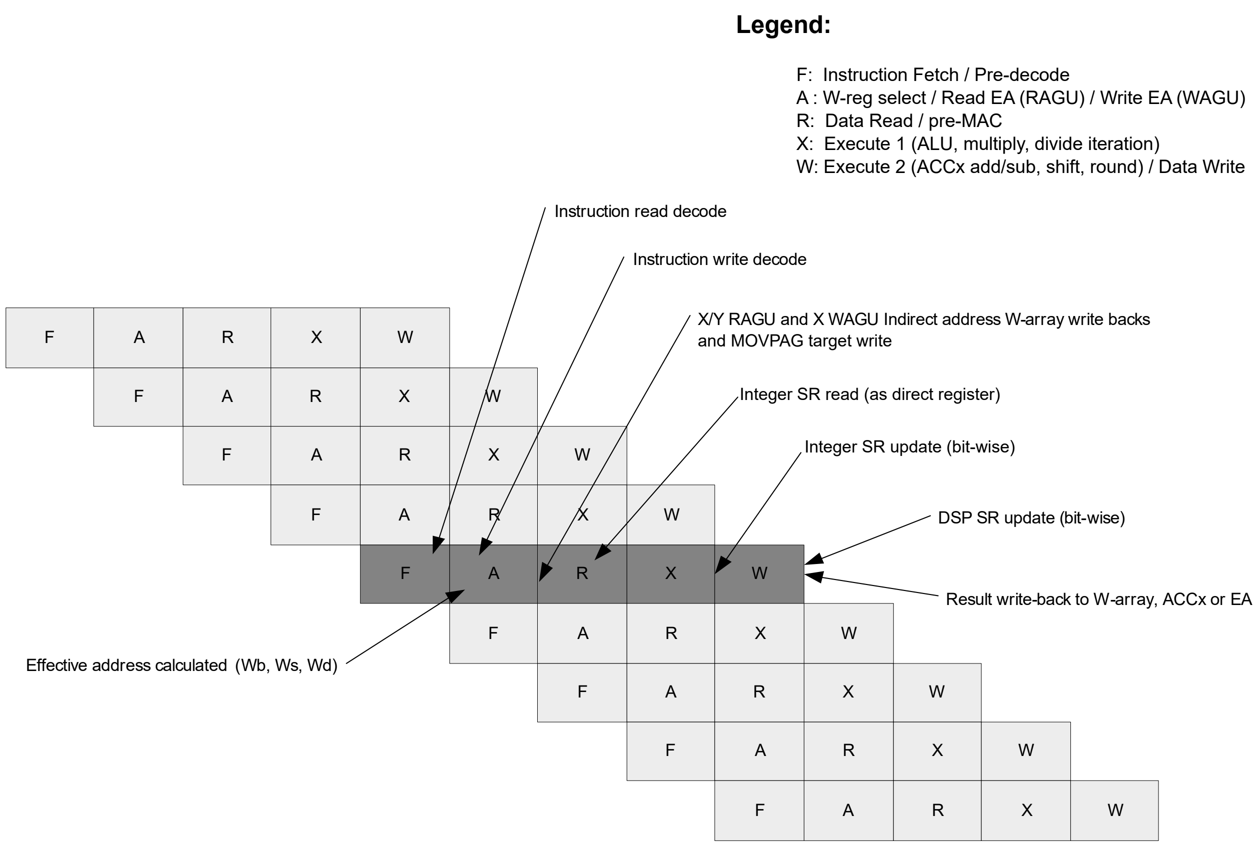
3.3.14. Instruction Flow Types

Most instructions in the device architecture occupy a single word of program memory and execute in a single cycle. However, some instructions take two or more instruction cycles to execute. Consequently, there are seven different types of instruction flow.

3.3.14.1. One Instruction Word, One Instruction Cycle

These instructions take one instruction cycle to execute, as shown in Figure 3-13. Most instructions are one-word, one-cycle instructions.

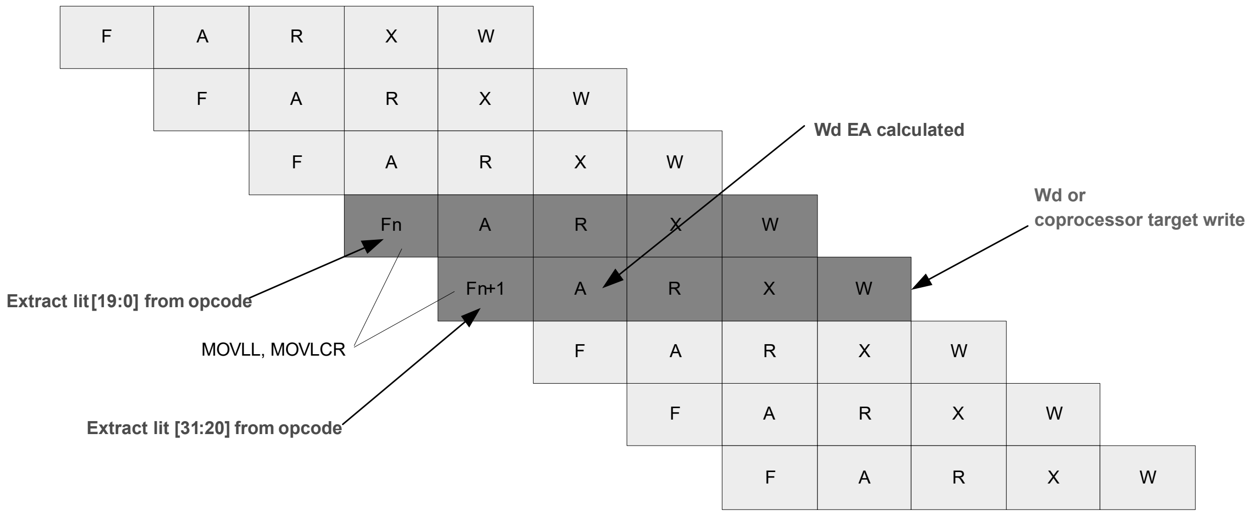
Figure 3-13. One-Word, One-Cycle (Generic) Instruction Flow



3.3.14.2. One Instruction Word, Two Instruction Cycles

In these instructions, there is no prefetch flush. The only instructions of this type are the `MOV.D` instructions (load and store double word), SFR reads and SFR bit operations. Two cycles are required to complete these instructions, as shown in Figure 3-14.

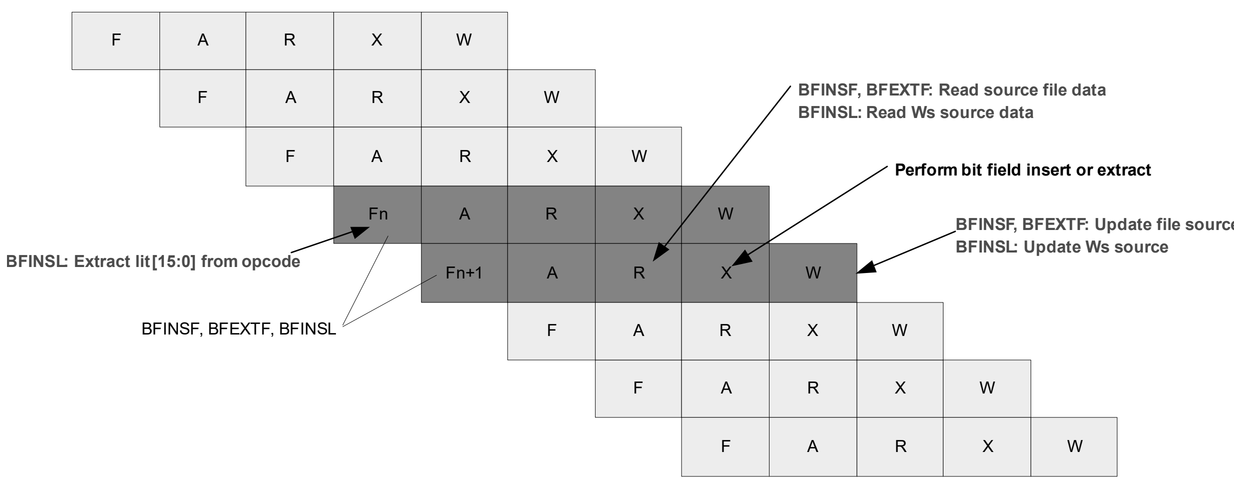
Figure 3-14. `MOVLL` & `MOVLCR` Two-Word, Two-Cycle Instruction Flow



3.3.14.3. One Instruction Word, Two or Four Instruction Cycles (Program Flow Changes)

These instructions include relative call and branch instructions. When an instruction changes the PC (other than to increment it), the program memory prefetch data must be discarded. This makes the instruction take four effective cycles to execute, as shown in Figure 3-15.

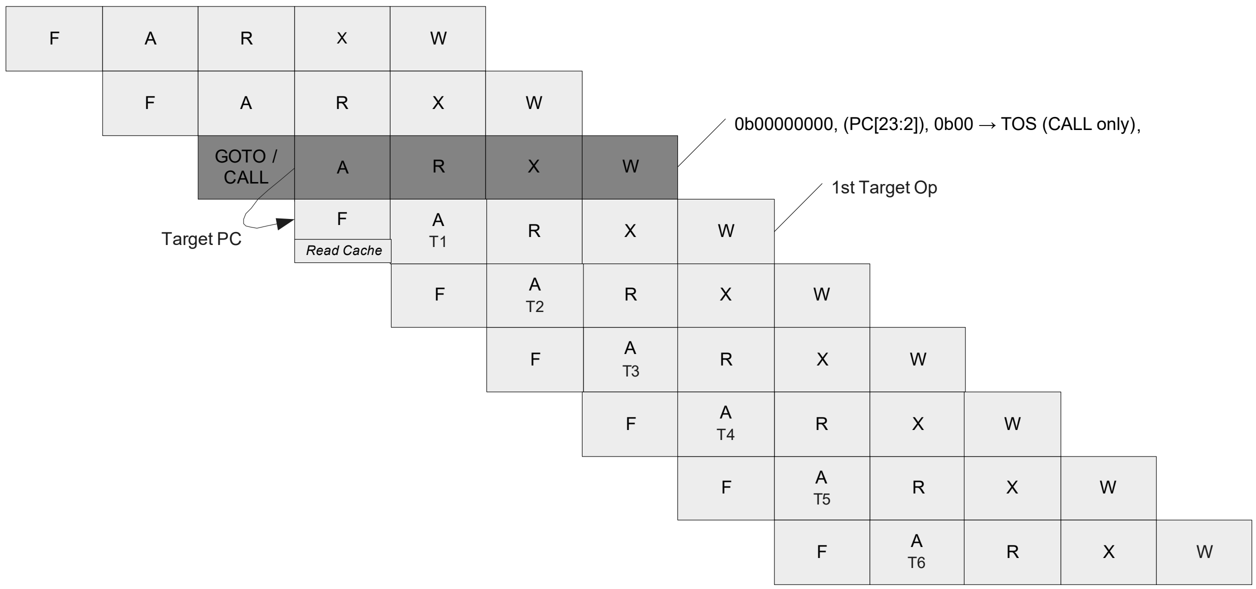
Figure 3-15. `BFEXTF`, `BTINSF` & `BFINSL` Two-Word, Two-Cycle Instruction Flow



3.3.14.4. Two Instruction Words, Four Instruction Cycles – `GOTO` or `CALL`

In these instructions, the fetch after the instruction contains data. This results in a four-cycle instruction, as shown in Figure 3-16. The second word of a two-word instruction is encoded so that it executes as a `NOP` if it is fetched by the CPU when the CPU did not first fetch the first word of the instruction. This is important when a two-word instruction is skipped by a skip instruction (see Figure 3-16).

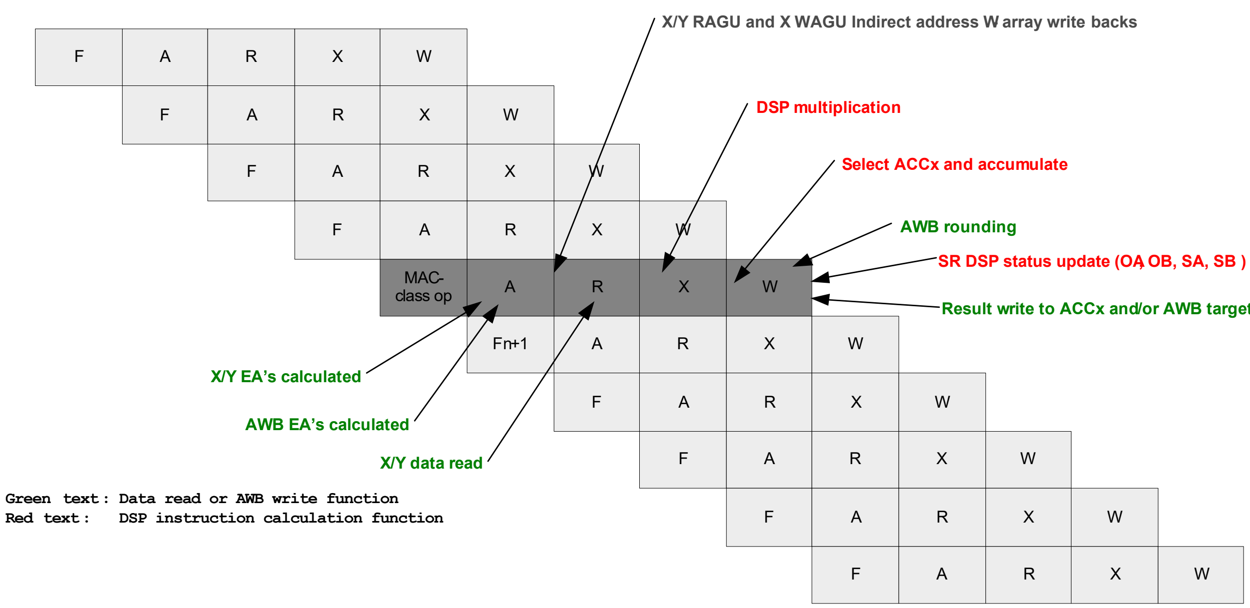
Figure 3-16. `GOTO` & `CALL` Unconditional PFC Instruction Flow



3.3.14.5. Address Register Dependencies

These are instructions subjected to a stall due to a data address dependency between the X data space read and write operations. An additional cycle is inserted to resolve the resource conflict, as discussed in Figure 3-17.

Figure 3-17. MAC-Class One-Cycle Instruction Flow



Note: DSP status cannot be updated prior to the end of the W-stage (i.e., one cycle later than the ALU status).

3.3.15. Loop Constructs

The dsPIC33A CPU supports two `REPEAT` constructs to provide unconditional automatic program loop control. The `REPEAT` instruction implements a single instruction program loop. REPEAT instructions use control bits within the CPU STATUS Register (SR) to temporarily modify the CPU operation.

3.3.15.1. REPEAT Loop Construct

The `REPEAT` instruction causes the instruction that follows it to be repeated a specified number of times. A literal value contained in the instruction, or a value in one of the W registers, can be used to specify the `REPEAT` count value. The W register option enables the loop count to be a software variable.

An instruction in a `REPEAT` loop is executed at least once. The number of iterations for a `REPEAT` loop is the 20-bit literal value + 1 or Wn + 1. The syntax for the two forms is shown in REPEAT Loop Construct.

Example 3-3. `REPEAT` Loop Construct

```
;    Using a literal value as a counter
REPEAT #lit20    ; RCOUNT <-- lit20
(Valid target Instruction)
;
;    Using a W register as a counter
REPEAT Wn    ; RCOUNT <-- Wn
(Valid target Instruction)
```

3.3.15.1.1. REPEAT Operation

The loop count for `REPEAT` operations is held in the 32-bit Repeat Loop Counter register (RCOUNT), which is memory-mapped. RCOUNT is initialized by the `REPEAT` instruction. The `REPEAT` instruction sets the `REPEAT` Loop Active (RA) Status bit (SR[4]) to '`1`' if the RCOUNT is a non-zero value.

RA is a read-only bit and cannot be modified through software. For `REPEAT` loop count values greater than '`0`', the Program Counter is not incremented. Furthermore, Program Counter increments are inhibited until RCOUNT = `0`.

For a loop count value equal to '`0`', `REPEAT` has the effect of a `NOP` and the RA (SR[4]) bit is not set. The `REPEAT` loop is essentially disabled before it begins, allowing the target instruction to execute only once while prefetching the subsequent instruction (i.e., normal execution flow).

Note: The instruction immediately following the `REPEAT` instruction (i.e., the target instruction) is always executed at least one time, and it is always executed one time more than the value specified in the 20-bit literal or the W register operand.

3.3.15.1.2. Interrupting a REPEAT Loop

A `REPEAT` instruction loop can be interrupted at any time.

The state of the RA bit is preserved on the stack during exception processing to enable the user application to execute further `REPEAT` loops from within any number of nested interrupts. After the SR is stacked, the RA Status bit is cleared to restore normal execution flow within the ISR.

Note: If a `REPEAT` loop has been interrupted, and an ISR is being processed, the user application must stack the Repeat Count register (RCOUNT) before it executes another `REPEAT` instruction within an ISR. If a `REPEAT` instruction is used within an ISR, the user application must unstack the RCOUNT register before it executes the `RETFIE` instruction.

Returning into a `REPEAT` loop from an ISR using the `RETFIE` instruction requires no special handling. RETFIE pops the PC, and that becomes the address of the next instruction to be fetched in its F-stage. The RETFIE instruction is "padded" with FNOPs (2), so the target instruction of the RETFIE PFC can execute as normal.

Early Termination of a REPEAT Loop

An interrupted `REPEAT` loop can be terminated earlier than normal in the ISR by clearing the RCOUNT register in the software.

3.3.15.1.3. Restrictions on the REPEAT Instruction

Any instruction can immediately follow a `REPEAT` except for the following:

- Program Flow Control instructions (any branch, compare and skip, subroutine calls, returns, etc.)
- Another `REPEAT` or `DTB` instruction
- `DISICTL`, `ULNK`, `LNK`, `PWRSAV` or `RESET` instruction
  Note: Some instructions and/or Instruction Addressing modes can be executed within a `REPEAT` loop, but it might not make sense to repeat all instructions.

3.3.16. Data Space Address Generation Units (AGUs)

dsPIC33AK256MPS306 family devices contain three independent Address Generator Units (AGUs). The X RAGU and X WAGU support byte (.b), word (.w) and long word (.l)-sized data space reads and writes for MCU instructions and word or long word reads and writes for DSP instructions. The Y AGU supports word and long word-sized data reads for the DSP MAC-class of instructions only. The AGUs are each capable of supporting two types of data addressing.

- Linear Addressing
- Modulo (circular) Addressing

In addition, the X WAGU can support Bit-Reversed Addressing.

Linear and Modulo Data Addressing modes can be applied to any address within the unified address space. Although Bit-Reversed Addressing will work with any EA calculation, and by definition, it is only applicable to data space.

Data space memory is organized as 32-bit words; all Effective Addresses (EAs) point to bytes. Instructions can thus access any byte, aligned word (data words at an even byte address) or aligned long word (data words at an even 32-bit word address).

Misaligned accesses are not supported, and if attempted, they will initiate an address error trap. The least significant two bits of the EA are used to determine the byte or upper/lower 16-bit word access. EA[0] will always be 0b0 for word accesses, and EA[1:0] will always be 0b00 for long word accesses.

SFRs and RAM support byte, word and double-word read or write operations.

When executing instructions that require just one source operand to be fetched from (and one result to be written back to) data space, the X RAGU and X WAGU are used to calculate the EAs of the source and destination, respectively. The AGUs can generate an address to point to anywhere in the 16 Mbyte address space. They support all MCU addressing modes and Modulo Addressing for low overhead circular buffers. The X WAGU also supports Bit-Reversed Addressing to facilitate FFT data reorganization.

When executing instructions which require two source operands to be concurrently fetched (i.e., the MAC class of DSP instructions), both the X RAGU and Y AGU are used simultaneously.

The dsPIC33AK256MPS306 device family contains an X AGU and a Y AGU for generating data memory addresses. Both X and Y AGUs can generate any EA within the available data memory range. However, EAs that are outside of the physical memory provided return all zeros for data reads and writes to those locations and, therefore, have no effect. Furthermore, an address error trap will be generated. For more information on address error traps, refer to Interrupt Controller.

3.3.16.1. X Address Generation Unit

The X AGU is used by all instructions and supports all addressing modes. The X AGU consists of a read AGU (X RAGU) and a write AGU (X WAGU), which operate independently on separate read and write buses during different phases of the instruction cycle. The X read data bus is the return data path for all instructions that view data space as a combined X and Y address space. It is also the X address space data path for the dual operand read instructions (DSP instruction class). The X write data bus is the only write path to the combined X and Y data space for all instructions.

The X AGU supports linear addressing through all of the address space. It can, therefore, generate EAs within the range 0x000000 to 0xFFFFFF.

The X RAGU starts its EA calculation during the prior instruction cycle, using information derived from the just prefetched instruction. The X RAGU EA is presented to the address bus at the beginning of the instruction cycle.

The X WAGU starts its EA calculation at the beginning of the instruction cycle. The EA is presented to the address bus during the write phase of the instruction.

Both the X RAGU and the X WAGU support Modulo Addressing.

Bit-Reversed Addressing is supported by the X WAGU only.

3.3.16.2. Y Address Generation Unit

The Y data memory space has one AGU that supports data reads from the Y data memory space. The Y memory bus is never used for data writes. The function of the Y AGU and Y memory bus is to support concurrent data reads for DSP class instructions.

The Y AGU can generate an EA within the data space address range 0x0000 to 0xFFFFFF.

The Y AGU timing is identical to that of the X RAGU, in that its EA calculation starts prior to the instruction cycle, using information derived from the prefetched instruction. The EA is presented to the address bus at the beginning of the instruction cycle.

The Y AGU supports Modulo Addressing and Post-Modification Addressing modes for the DSP class of instructions that use it.

Note: The Y AGU does not support data writes. All data writes occur via the X WAGU to the combined X and Y data spaces. The Y AGU is only used during data reads for dual source operand DSP instructions.

3.3.16.3. Address Generation Units and DSP Class Instructions

The Y AGU and Y memory data paths are used in concert with the X RAGU by the DSP class of instructions to provide two concurrent data read paths. For example, the MAC instruction can simultaneously fetch two operands to be used in the next multiplication.

DSP class of instructions may use any W-reg (except W15) for either X or Y address space accesses, unlike previous devices. Any data write performed by a DSP class instruction takes place in the combined X and Y data space and the write occurs across the X-bus. Consequently, the data can be written to any address regardless of where the EA is directed.

The Y AGU only supports Post-Modification Addressing modes associated with the DSP class of instructions. The Y AGU also supports Modulo Addressing for automated circular buffers. All other (MCU) class instructions can access the Y data address space through the X AGU when it is regarded as part of the composite linear space.

3.3.16.4. Data Alignment

The ISA supports long word (32-bit), word (16-bit) and byte (8-bit)-sized operations. Data are aligned in data memory and registers as long words, but all data space EAs resolve to bytes. Data word and byte reads will read the complete 32-bit word that contains the word or byte, using the LSbs of any EA to determine which word or byte to select within the CPU. The selected word or byte is placed onto the lsw or byte of the X data path (no byte accesses are possible from the Y data path as the MAC-class of instruction can only fetch words or long words). That is, data memory and registers are organized as four parallel byte-wide entities with a shared (long word) address decode but separate write lines. Data byte writes will only write to the corresponding side of the array or register which matches the byte address.

Note: Byte reads will always read the entire word, so mechanisms to clear or set peripheral status bits when read (e.g. quick flag clearing mechanisms) are not allowed.

As a consequence of this byte addressability, all EA calculations must be scaled to step through long word aligned memory. For example, the core must recognize that Post-Modified Register Indirect Addressing mode, [Ws++], will result in a value of Ws+1 for byte operations, Ws+2 for word operations and Ws+4 for long word operations.

Misaligned word or long word accesses are not supported. For word accesses, the LSb of the EA must be 0. For long word accesses, the least significant two bits of the EA must be 0b00. Therefore, care must be taken when mixing operations of different data widths or translating from 16-bit code. Should a misaligned read or write be attempted, an address error trap will be forced. If the Fault occurs during a read access, the read will be allowed to complete. If the Fault occurs during a write access, the write will also be allowed to complete (inhibiting the write would have been possible but inconsistent with other situations where an errant write could not be inhibited). In both cases, the address error trap will be asserted. The next instruction (already prefetched and underway) will be executed while the exception is arbitrated and acknowledged. When this instruction completes, the trap will then be taken, allowing the system and/or user to examine the machine state subsequent to execution of the address Fault.

Note: Byte and word ALU operations can produce results in excess of a byte or a word. However, to maintain 16-bit backwards code compatibility, the ALU result destination write from all operations maintains the same width as that of the source operands (i.e., MSbs of the destination are not modified), and the SR is updated based only upon the state of the result data.

A sign extend (SE) instruction is provided to allow users to translate 8-bit to 16-bit and 16-bit to 32-bit signed values. Alternatively, for unsigned data, users can clear the MS portion of any W register through executing a byte or word zero extend (ZE).

Note: Care must be taken when mixing byte and word-sized instructions/operands.

Although most instructions are capable of operating on long word, word or byte data sizes, it should be noted that the DSP and some other instructions operate on long word or word-sized data only.

Figure 3-18. Data Alignment

31	23	15	7 0	Address
Byte 3	Byte2	Byte1	Byte 0	0x000000
Byte 7	Byte6	Byte5	Byte 4	0x000004
Byte 11	Byte10	Byte9	Byte 8	0x000008

3.3.17. MAC Instructions

The dual source operand DSP instructions (`ED`,`EDAC`, `MAC`, `MPY`, `MPYN`, `SQR`, `SQRAC`, `MSC`, `SQRSC` and `SQRN`), also referred to as `MAC` instructions, use a simplified set of addressing modes to allow the user application to effectively manipulate the data pointers through register indirect tables.

These instructions support various addressing modes for X and Y data bus, where W-registers accessing these data buses may be any W-reg (except W15) for either X or Y address space accesses. Pre or post-modification values are scaled based upon instruction operand width. The MAC-class instruction also supports the ability to write the contents of the accumulator that is not being used as the instruction result destination to a memory or W-register as defined by the instruction with a restricted set of addressing modes. This is referred to as the Accumulator Write Back (AWB).

Note:
AWB is only intended for use when the DSP engine is operating in Fractional Data mode. It can only write the MS portion of the target accumulator fractional value.

MAC-class instructions are no longer tied to operand reads of X and Y address space. Operands may both be sourced from X-space, resulting in reading the operand data sequentially rather than concurrently. This will add an additional RAM data fetch delay (typically one cycle) to all such instructions.

3.3.18. Modulo Addressing

Modulo Addressing mode is a method of providing an automated means to support circular data buffers using hardware. The objective is to remove the need for software to perform data address boundary checks when executing tightly looped code, as is typical in many DSP algorithms.

Modulo Addressing can operate in either data or program space (since the data pointer mechanism is essentially the same for both). One circular buffer can be supported in each of the X (which also provides the pointers into program space) and Y data spaces. Modulo Addressing can operate on any W register pointer. However, it is not advisable to use W14 or W15 for Modulo Addressing since these two registers are used as the stack frame pointer and stack pointer, respectively.

In general, any particular circular buffer can be configured to operate in only one direction, as there are certain restrictions on the buffer start address (for incrementing buffers) or end address (for decrementing buffers) based upon the direction of the buffer.

The only exception to the usage restrictions is for buffers that have a power-of-two length. As these buffers satisfy the start and end address criteria, they can operate in a Bidirectional mode (that is, address boundary checks are performed on both the lower and upper address boundaries).

3.3.18.1. Start and End Addresses

The Modulo Addressing scheme requires that a starting and ending address be specified and loaded into the 24-bit Modulo Buffer Address registers: XMODSRT, XMODEND, YMODSRT and YMODEND.

Note: Y space Modulo Addressing EA calculations assume word-sized data (LSb of every EA is always clear).

The length of a circular buffer is not directly specified. It is determined by the difference between the corresponding start and end addresses. The maximum possible length of the circular buffer is 32K words (64 Kbytes).

3.3.18.2. W Address Register Selection

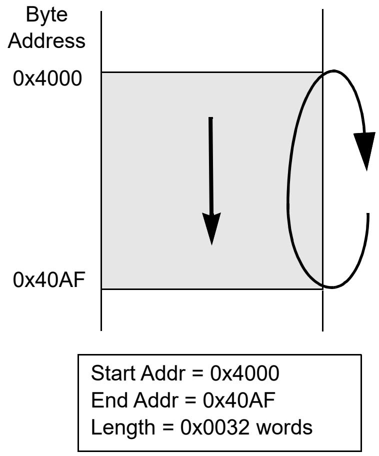
The Modulo and Bit-Reversed Addressing Control register, MODCON[15:0], contains enable flags as well as a W register field to specify the W Address registers. The XWM and YWM fields select the registers that operate with Modulo Addressing.

- If XWM = `1111`, X RAGU and X WAGU Modulo Addressing are disabled.
- If YWM = `1111`, Y AGU Modulo Addressing is disabled.

The X Address Space Pointer W (XWM) register, to which Modulo Addressing is to be applied, is stored in MODCON[3:0]. Modulo Addressing is enabled for the X data space when XWM is set to any value other than '`1111`' and the XMODEN bit is set (MODCON[15]).

The Y Address Space Pointer W (YWM) register, to which Modulo Addressing is to be applied, is stored in MODCON[7:4]. Modulo Addressing is enabled for Y data space when YWM is set to any value other than '`1111`' and the YMODEN bit is set (MODCON[14]).

Figure 3-19. Modulo Addressing Operation Example




```
; Set XMODEN bit and XWM = w7
    ; w7 selected for X-AGU modulo addressing
    MOV.l #0x8007, w6
    MOV.l w6, MODCON

; set modulo end address
    MOV.l #0x40AF, w6
    MOV.l w6, XMODEND

; set modulo start address
    MOV.l #0x4000, w7
    MOV.l w7, XMODSRT
    ; Modulo Addressing buffer length set to 0xAF.
. . .
. . .

    MPY.l [w7]+=4, w5, A  ; w7 = 0x4004
    REPEAT #0x2A
    MAC.l [w7]+=4, w5, A
;Content of W7 rolled back to 0x4000
...
...
```

3.3.18.3. Bit-Reversed Addressing

Bit-Reversed Addressing mode is intended to simplify data reordering for radix-2 FFT algorithms. It is supported by the X AGU for data writes only.

The modifier, which can be a constant value or register contents, is regarded as having its bit order reversed. The address source and destination are kept in normal order. Thus, the only operand requiring reversal is the modifier.

3.3.18.3.1. Bit-Reversed Addressing Implementation

Bit-Reversed Addressing can only be enabled through the use of the movr.(w/l) instruction. This type of addressing is effective when used with pre-modified or post-modified destination addressing. The destination Bit-Reversed Addressing modifier is sourced from XBREV.XB[14:0].

If the length of a bit-reversed buffer is $M = 2^N$ bytes, the last 'N' bits of the data buffer start address must be zeros.

The XB[14:0] bits are the Bit-Reversed Addressing modifier, or 'pivot point', which is typically a constant. In the case of an FFT computation, its value is equal to half of the FFT data buffer size.

Note: All bit-reversed EA calculations assume either word size (where the least significant bit of every effective address is always clear) or long word size (where the two least significant bits of the effective address are always clear), based on the operation data width selected. The XB value is scaled accordingly to generate compatible (byte) addresses.

Bit-Reversed Addressing is only possible when using the MOVR instruction, and it can target a 16-bit or 32-bit-sized data. The MOVR instruction supports register indirect with Pre-Increment or Post-Increment Addressing and 16/32 bit-sized data writes. When Bit-Reversed Addressing is active, the W address pointer is always added to the address modifier (XB) and the offset associated with the Register Indirect Addressing mode is ignored. In addition, the LSb of each 16-bit address and the LS 2-bits of each 32-bit address will always be zero for both source and destination EAs. The MOVR instruction also supports "in-place" data reordering (where only one data buffer is used for both the source and destination). Source and destination indirect addressing may use the same register

Note: Modulo Addressing and Bit-Reversed Addressing can be enabled simultaneously using the same W register, but the Bit-Reversed Addressing operation will always take precedence for data writes when enabled.

If Bit-Reversed Addressing has already been enabled by setting the BREN (XBREV[15]) bit, a write to the XBREV register should not be immediately followed by an indirect read operation using the W register that has been designated as the Bit-Reversed Pointer.

Figure 3-20. Bit-Reversed Addressing Example

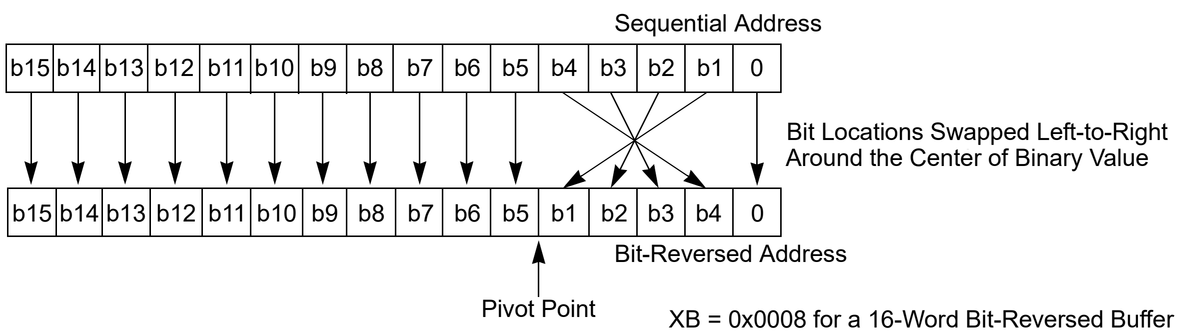


Table 3-8. Bit-Reversed Addressing Sequence (16-Entry)

Normal Address					Bit-Reversed Address				
A3	A2	A1	A0	Decimal	A3	A2	A1	A0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	8
0	0	1	0	2	0	1	0	0	4
0	0	1	1	3	1	1	0	0	12
0	1	0	0	4	0	0	1	0	2
0	1	0	1	5	1	0	1	0	10
0	1	1	0	6	0	1	1	0	6
0	1	1	1	7	1	1	1	0	14
1	0	0	0	8	0	0	0	1	1
1	0	0	1	9	1	0	0	1	9
1	0	1	0	10	0	1	0	1	5
1	0	1	1	11	1	1	0	1	13
1	1	0	0	12	0	0	1	1	3
1	1	0	1	13	1	0	1	1	11
1	1	1	0	14	0	1	1	1	7
1	1	1	1	15	1	1	1	1	15

Example 3-4. 32-Bit Data, Two Buffer Bit-Reversed Data Reordering Example

```
; Two buffer (input and output) bit reversed data re-order subroutine for 32-bit (real)
; data values
;
; W0: Temp
; W1: Data table size N (long words)
; W8: Input data table pointer (natural order) initialized to start of table
; W9: Output data table pointer (bit reversed) initialized to start of table

       push.l w0
       mov.sl #_XBREV, w0
       lsr.l w1, #1, [w0] ; XBREV = N/2

       sub.l #1, w1
       repeat w1
       movr.l [w8++], [w9++] ; Move data from input to output buffer, then
                             ; bump natural order and bit reversed pointers
       pop.l w0
       return
```

3.3.19. Address Register Dependencies

The dsPIC33A architecture supports a data space read (source) and a data space write (destination) for most MCU class instructions. The EA calculation by the AGU and a subsequent data space read or write each take one instruction cycle to complete. This timing causes the data space read and write operations for each instruction to overlap.

3.3.19.1. Read-After-Write Dependency Rules

If the W register is used as a write operation destination in the current instruction, and the W register being read in the prefetched instruction is the same, the following rules apply.

- If the destination write (current instruction) does not modify the contents of Wn, no stalls will occur.
- If the source read (prefetched instruction) does not calculate an EA using Wn, no stalls will occur.

During each instruction cycle, the dsPIC33A hardware automatically checks to see if a RAW data dependency is about to occur. If the conditions specified above are not satisfied, the CPU automatically adds a one-instruction cycle delay before executing the prefetched instruction. The instruction stall provides enough time for the destination W register write to occur before the next (prefetched) instruction that uses the written data. Table 3-9 provides a summary of read-after-write dependency.

Table 3-9. Read-After-Write Dependency Summary

Destination Addressing Mode Using Wn	Source Addressing Mode Using Wn	Status	Examples (Wn = W2)
Direct	Direct	Allowed	`ADD.w W0, W1, W2 MOV.w W2, W3`
Direct	Indirect	Stall	`ADD.w W0, W1, W2 MOV.w [W2], W3`
Direct	Indirect with Modification	Stall	`ADD.w W0, W1, W2 MOV.w [W2++], W3`
Indirect	Direct	Allowed	`ADD.w W0, W1, [W2] MOV.w W2, W3`
Indirect	Indirect	Allowed	`ADD.w W0, W1, [W2]` `MOV.w [W2], W3`
Indirect	Indirect with Modification	Allowed	`ADD.w W0, W1, [W2] MOV.w [W2++], W3`
Indirect with Modification	Direct	Allowed	`ADD.w W0, W1, [W2++] MOV.w W2, W3`

Indirect	Indirect	Stall	`ADD.w W0, W1, [W2]`
			`MOV.w [W2], W3`
			`; W2=0x0004 (mapped W2)`

Indirect	Indirect with Modification	Stall	`ADD.w W0, W1, [W2] MOV.w [W2++], W3` `; W2=0x0004 (mapped W2)`
Indirect with Modification	Indirect	Stall	`ADD.w W0, W1, [W2++] MOV.w [W2], W3`
Indirect with Modification	Indirect with Modification	Stall	`ADD.w W0, W1, [W2++] MOV.w [W2++], W3`

3.3.19.2. Instruction Stall Cycles

An instruction stall is essentially a wait period instruction cycle added in front of the read phase of an instruction to allow the prior write to complete before the next read operation. For interrupt latency, the stall cycle is associated with the instruction following the instruction where it was detected (i.e., stall cycles always precede instruction execution cycles).

If a RAW data dependency is detected, the dsPIC33A CPU begins an instruction stall. During an instruction stall, the following events occur.

- The write operation in progress (for the previous instruction) is allowed to complete as normal.
- Data space is not addressed until after the instruction stall.
- PC increment is inhibited until after the instruction stall.
- Further instruction fetches are inhibited until after the instruction stall.

3.3.19.2.1. Instruction Stall Cycles And Interrupts

When an interrupt event coincides with two adjacent instructions that causes an instruction stall, one of two possible outcomes can occur.

If the interrupt coincides with the first instruction, the first instruction is allowed to complete while the second instruction is executed after the ISR completes. In this case, the stall cycle is eliminated from the second instruction because the exception process provides time for the first instruction to complete the write phase.

If the interrupt coincides with the second instruction, the second instruction and the appended stall cycle are allowed to execute before the ISR. In this case, the stall cycle associated with the second instruction executes normally. However, the stall cycle is effectively absorbed into the exception process timing. The exception process proceeds as if an ordinary two-cycle instruction was interrupted.

3.3.19.2.2. Instruction Stall Cycles and Flow Change Instructions

The `CALL` and `RCALL` instructions write to the stack using Working register, W15, and can, therefore, force an instruction stall prior to the next instruction if the source read of the next instruction uses W15.

The `RETFIE` and `RETURN` instructions can never force an instruction stall prior to the next instruction because they only perform read operations. However, the `RETLW` instruction can force a stall because it writes to a W register during the last cycle.

The `GOTO` and branch instructions can never force an instruction stall because they do not perform write operations.

3.3.19.2.3. Instruction Stalls and `REPEAT` Loops

Other than the addition of instruction stall cycles, RAW data dependencies do not affect the operation of either `DO` or `REPEAT` loops.

The prefetched instruction within a `REPEAT` loop does not change until the loop is complete or an exception occurs. Although register dependency checks occur across instruction boundaries, the dsPIC33A devices effectively compare the source and destination of the same instruction during a `REPEAT` loop.

3.3.19.3. Data Space Arbiter Stalls

A CPU stall can also be a result of competition for extended data space resources. When the data space arbiter logic determines that the CPU cycle must be stalled to allow another bus master (e.g., DMA controller or USB module) access to data memory, instruction execution is suspended until the higher priority bus master completes the data access.

3.4. Prefetch Branch Unit (PBU)

The Prefetch Branch Unit (PBU) in the dsPIC33A core devices accelerates the interface between the Program Flash Memory (PFM) and the CPU instruction bus. The PBU can predictively prefetch the next sequential address and cache fetched program data that are the target of a CPU instruction fetch with improved performance.

The PBU in dsPIC33A core devices supports the following functions:

- PBU accelerates the execution of linear program code flow.
- As cache accelerates, the execution of non-linear program flow changes (branches).

The PBU in the dsPIC33A core devices has the following features:

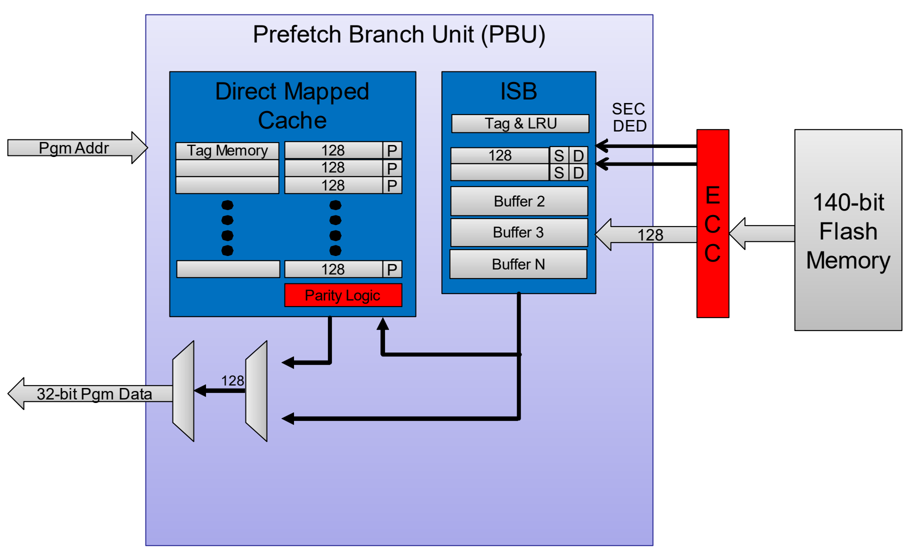
- Provides an interface between the PFM and the CPU instruction bus.
- Instruction Stream Buffers for prefetching and caching of linear PFM instruction flows
- Instruction cache for caching the most frequently hit target instructions
- Provides parity checks on program data stored in the instruction cache to ensure data integrity.

The enhanced version of the PBU with this device includes the following features:

- Expanded parity coverage for cache line to include tag address and valid bit
- Parity bit added for each 32-bit program word in cache line
- Supports run-time (user-accessible) cache memory and ISB (Instruction Stream Buffer) diagnostics for read/write/compare contents

The PBU block diagram in Figure 3-21 shows data paths to and from the PBU in the dsPIC33A environment. The PBU provides data when the CPU fetches program data from Flash memory. It may provide program data from an internal buffer or fetch program data from Flash if the requested program data is not available. Flash fetch operations are, therefore, accelerated when data are sourced from internal PBU buffers.

Figure 3-21. PBU Block Diagram



3.4.1. Architectural Overview

The PBU is a direct-mapped 128-line cache that helps in providing faster program data fetches to the CPU from Flash memory. The PBU provides program data from an internal instruction buffer, but if it is not available in the internal buffer, the PBU may fetch program data from Flash. Flash fetch operations are, therefore, accelerated when data are sourced from internal PBU buffers.

The PBU provides an interface between the Program Flash Memory (PFM) and the CPU instruction bus and has the following components associated for operation.

- Instruction Stream Buffer (ISB) - Also termed as the Prefetch Unit (PFU), it is available for prefetching and caching linear PFM instruction flows. ISB is the component that buffers program data words from the program memory. The ISB consists of one or more buffers of a fixed depth. Each buffer holds one or more lines of data fetches from Flash memory. The data held in each buffer represents a linear code flow. These are defined as internal PBU buffers.
- Instruction Cache (IC) - Used for caching the target instructions that are most frequently hit. The IC refers to both the cache memory and the associated control logic that form the cache. The PBU contains a direct-mapped 128-line cache. The required width for the cache is 129-bits, with the extra bit being required for parity.
- Integrity Checking Logic - Provides parity checks on program data stored in the IC to ensure data integrity. This logic provides parity checking and fault injection on the contents of RAM associated with the IC.

The PBU assumes Flash data width and Flash access speed are sufficient to allow linear program execution at the required speed using only the ISB. The ISB serves as the prefetch buffer and allows the next line of Flash to be fetched as instructions from the current line are executed.

The Instruction Cache (IC) becomes useful when there are frequent program flow changes in the source code. A program flow change will result in extra clock cycles because the current Flash fetch must be allowed to complete and then a new fetch must be initiated at the new location. If the desired program data is available in the IC, the data may be sourced immediately without waiting for the ISB to complete a new fetch from Flash. However, PBU uses a larger, direct-mapped instruction cache and has little control and status interface available to the user as its operations are transparent.

Note: PBU does not provide data or caching for initiators other than the CPU instruction bus. Data access by the CPU data bus and other bus initiators is accomplished via a dedicated read buffer in the NVM controller.

3.4.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E60	CHECON	31:24								
		23:16								ISBBUF
		15:8	ON				CHEINV	CHECOH		
		7:0								FLTINJ
0x1E64	CHESTAT	31:24								
		23:16								
		15:8								
		7:0					ISBPE	TPE	ISBE	PARE
0x1E68	CHEFLTINJ	31:24								
		23:16								
		15:8								
		7:0	FLTPTR[7:0]							
0x1E6C	CHEDAT0	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E70	CHEDAT1	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E74	CHEDAT2	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E78	CHEDAT3	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E7C	CHECMD	31:24								
		23:16	RIP3	RIP2	RIP1	RIP0	WIP3	WIP2	WIP1	WIP0
		15:8			RVAL	WVAL	CMPV	WV		
		7:0		ADDR[6:0]						
0x1E80	TAGDAT	31:24								
		23:16	ADDR[23:16]							
		15:8								
		7:0	ADDR[7:0]							
0x1E84	TAGCMD	31:24						RTP0	WTP0	
		23:16		CMPP	EWP	EWD	CMPA	RD		
		15:8								
		7:0		ADDR[6:0]						
0x1E88	ISBDAT0	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E8C	ISBDAT1	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E90	ISBDAT2	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							
0x1E94	ISBDAT3	31:24	WORD[31:24]							
		23:16	WORD[23:16]							
		15:8	WORD[15:8]							
		7:0	WORD[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E98	ISBCMD0	31:24	RIP[3:0]				WIP[3:0]			
		23:16		CMPP	EWD	EWP	CMPD	RD		
		15:8						LINE[2:0]		
		7:0	ISBREG[2:0]							
0x1E9C	ISBCMD1_7	31:24	RIP[3:0]				WIP[3:0]			
		23:16		CMPP	EWD	EWP	CMPD	RD		
		15:8						LINE[2:0]		
		7:0	ISBREG[2:0]					SLICE[2:0]		

3.4.2.1. Cache Control Register

Name: CHECON
Offset: 0x1E60

Notes:
1. After being set, this bit will be cleared by hardware after the cache and ISB invalidations are completed. Any automatic invalidation will also result in this bit being cleared.
2. This setting is useful when programming non-program data into Flash (emulated EEPROM).
3. After the event of software set or software clear of the ON bit, all cache lines and all ISBs are invalidated. This is true if the CP bit is `0` or `1`.
4. Do not perform diagnostics on the ISB that is currently in use by the PBU module. If ISBBUF is `0`, do not execute diagnostics on ISBBUF_`0`.
5. ON `1≥0` and CP `1≥0` at the same time, clearing the ON has priority.
6. The CP and ON bits are intended to be used independently. If the ON bit is toggled while paused, a cache invalidation occurs.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
								ISBBUF
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
	ON				CHEINV	CHECOH		
Access	R/W				R/S/HC	R/W		
Reset	1				1	1		

Bit	7	6	5	4	3	2	1	0
								FLTINJ
Access								R/S/HC
Reset								1

Bit 16 – ISBBUF ISB Buffer Selection bit

Value	Description
`1`	When ON = `0` or (ON = `1` and CP = `1`), ISB buffer 0 will be used for prefetch.
`0`	When ON = `0` or (ON = `1` and CP = `1`), ISB buffer 1 will be used for prefetch.

Bit 15 – ON Cache ON bit

Value	Description
`1`	Cache and all ISB slices are enabled.
`0`	All cache lines and ISB buffers, except for the first buffer slice, are invalidated. ISB operates with one buffer slice, creating a two-deep buffer (basic Prefetch mode).

Bit 11 – CHEINV Manual Invalidate Control bit[(1)]

Value	Description
`1`	Force invalidation of all cache and ISB lines.

Value	Description
0	Invalidation of the Instruction Cache and ISBs occurs according to the CHECOH bit.

Bit 10 – CHECOH Cache Coherency Control bit[(2)]

Value	Description
1	Invalidate the cache upon a Flash programming event.
0	Do not invalidate the cache on a Flash programming event.

Bit 0 – FLTINJ Fault Inject Control bit

Value	Description
1	Parity Fault injection enabled for one-time event; cache line will be invalidated and flushed when access occurs, and upbs_event[1] will be asserted to indicate an integrity error to the system.
0	Parity Fault injection disabled.

3.4.2.2. Cache Status Register

Name: CHESTAT
Offset: 0x1E64

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					ISBPE	TPE	ISBE	PARE
Access					R/C/HS	R/C/HS	R/W	R/S/HC
Reset					0	0	1	1

Bit 3 – ISBPE ISB Parity Error Status bit

Value	Description
1	An ISB FIFO read parity error event has occurred. The CPU has fetched a word from the ISB with a parity error, resulting in a PBU miss event, or a diagnostic read initiated by the ISBCMD register has resulted in a parity error.
0	No ISB FIFO read error event has occurred.

Bit 2 – TPE TAG Parity Error Status bit

Value	Description
1	A TAG memory read parity error event has occurred. The CPU has fetched a word from the cache with a TAG parity error, resulting in a PBU miss event, or a diagnostic read initiated by the TAGCMD register has resulted in a parity error.
0	No TAG memory read error event has occurred.

Bit 1 – ISBE Read Error Status bit

Value	Description
1	A read error event has occurred; the CPU has fetched a word from the ISB with a security error.
0	No read error event has occurred.

Bit 0 – PARE Cache Parity Error Status bit

Value	Description
1	A parity error event has occurred; the CPU has fetched a word from the cache with a parity error.
0	No parity error event has occurred.

3.4.2.3. Cache Fault Injection Register

Name: CHEFLTINJ
Offset: 0x1E68

Note: There are 132 bit lines which cover the 128 bit data, in addition to 1 parity bit per 32 bit data entry in the cache line.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	FLTPTR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 7:0 – FLTPTR[7:0] Fault Injection Pointer bits

Value	Description
255-129	No effect
131	Bit 131 of cache data line
130	Bit 130 of cache data line
...	
1	Bit 1 of cache data line
0	Bit 0 of cache data line

3.4.2.4. Cache RAM Data Register (Comparator)

Name: CHEDAT
Offset: 0x1E6C,0x1E70,0x1E74,0x1E78

Notes:

1. This register is used for both the value to be written in the cache RAM location and the compare value for a location.
2. There are a total of four CHEDAT registers for inspection of the entire cache data word N. See Register Summary for the full SFR map.
3. User software should not attempt to write to this register on the instruction cycle immediately following a read via the CHECMD SFR. If this is done, the write contents will be lost due to the hardware read of this register.

Bit	31	30	29	28	27	26	25	24
	WORD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WORD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	WORD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WORD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – WORD[31:24] MSB of 32-bit Instruction Word N in Cache Line bits

Bits 23:16 – WORD[23:16] Third byte of 32-bit Instruction Word N in Cache Line bits

Bits 15:8 – WORD[15:8] Second Byte of 32 bit Instruction Word N in Cache Line bits

Bits 7:0 – WORD[7:0] LSB of 32-bit Instruction Word N in Cache Line bits

3.4.2.5. Cache RAM Command Register (Address/Control)

Name: CHECMD
Offset: 0x1E7C

Notes:
1. When CACHE is enabled (CHECON.ON = `1` and CHECON.CP = `0`), writes to the following register bits are ignored: RD, WP, WD, CMPP and CMPD.
2. If WD = `1` OR WP = `1`, a write to the RD bit is ignored.
3. The CMPP and CMPD control bits require a minimum of three CPU clocks to complete the comparison and get set before reading the result.
4. The CMPD bit has priority over the RD bit. If the RD bit is set simultaneously with the CMPD bit, the read operation will be ignored because the data comparison operation has priority.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	RIP3	RIP2	RIP1	RIP0	WIP3	WIP2	WIP1	WIP0
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			RVAL	WVAL	CMPV	WV		
Access			R	W	R/W/HS/HC	W		
Reset			0	0	0	0		

Bit	7	6	5	4	3	2	1	0
		ADDR[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 23 – RIP3 Read Value of Parity bit 3

Bit 22 – RIP2 Read Value of Parity bit 2

Bit 22 – CMPP

Compare the contents of the addressed cache RAM line PARITY to WIP bits in this register. In READ mode, the register will return the result of the comparison.

Value	Description
1	Compare result is good.
0	Compare result has an error.

Bit 21 – RIP1 Read Value of Parity bit 1

Bit 21 – EWP

Enable the four bits of parity to be directly written to the parity bits in the cache RAM.

Value	Description
1	Trigger a write of parity values from the WIP register to the cache RAM parity bits.
0	No operation.

Bit 20 – RIP0 Read Value of Parity bit 0

Bit 20 – EWD
Enable four 32-bit data registers to be written to the cache.

Value	Description
1	Trigger a write of data values from the CHEDAT register to the cache RAM.
0	No operation.

Bit 19 – WIP3
Direct write of the parity bit 3 (when enabled by bit 13), readback of written data.

Bit 19 – CMPD
Compare the contents of the addressed cache RAM line to the contents of data registers. Read and compare with the values supplied.
Note: Read operation allows cache parity hardware to be used to check for parity evaluation. In READ mode, the register will return the result of the comparison. This function will also compare the contents of the cache parity bits against the parity bits in the control registers bits 16-19 (WIPn) if the CMPP bit is enabled.

Value	Description
1	Compare result is good.
0	Compare result has an error.

Bit 18 – WIP2
Direct write of the parity bit 2 (when enabled by bit 13), readback of written data.

Bit 18 – RD
Read/Write four 32-bit register values to address cache RAM into CHEDAT, and the four cache parity bits will be read into the RIP bits of this SFR.

Value	Description
1	Trigger a single READ event.
0	No operation.

Bit 17 – WIP1
Direct write of the parity bit 1 (when enabled by bit 13), read back of written data.

Bit 16 – WIP0
Direct write of the parity bit 0 (when enabled by bit 13), read back of written data.

Bit 13 – RVAL
Value read from the TAG in the addressed CACHE line.

Bit 12 – WVAL
Value written from the TAG in the addressed CACHE line.

Bit 11 – CMPV
Compare the bit in WVAL (bit 12) with the value in the current CACHE line and post the results to this bit.

Value	Description
1	Values are compared.
0	Values are not compared.

Bit 10 – WV
Write the value in the WVAL (bit 12) to the tag valid bit in the CACHE line.

Bits 6:0 – ADDR[6:0]

Address of a 128-bit, four-word cache line.

3.4.2.6. Tag Data Register

Name: TAGDAT
Offset: 0x1E80

Notes:
1. The user software should not attempt to write to this register on the instruction cycle immediately following a read from the TAGCMD register; the write contents will be lost due to the hardware read of this register.
2. The tag address is comprised of bits 23-11 of the program memory address for a 128-line direct-mapped cache.
3. The tag data/address is 23 bits aligned, with unused bits (0-10 being ignored by hardware, and only bits 11-23 will be active).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	ADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Access								
Reset								
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	ADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 22 – ADDR[15:8] Middle Byte of Tag Data bit

Bits 23:16 – ADDR[23:16] MS Byte of Tag Data bits

Bits 7:0 – ADDR[7:0] LS Byte of Tag Data bits

3.4.2.7. Tag Command Register (Address/Control)

Name: TAGCMD
Offset: 0x1E84

Notes:
1. The CMPP and CMPA control bits require a minimum of three CPU clocks to complete the comparison and get set before reading the result.
2. The CMPA bit has priority over the RD bit. If the RD bit is set simultaneously with the CMPD bit, then the read operation will be ignored because the data comparison operation has priority.

Bit	31	30	29	28	27	26	25	24
						RTP0	WTP0	
Access						R/HS/HC	R/W	
Reset						0	0	

Bit	23	22	21	20	19	18	17	16
		CMPP	EWP	EWD	CMPA	RD		
Access		R/W/HS/HC	R/W	R/W	R/W/HS/HC	W		
Reset		0	0	0	0	0		

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		ADDR[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 26 – RTP0 Read Value of TAG Parity bit 0

Bit 25 – WTP0
Direct write of the TAG parity bit 0 (when enabled by bit 20), readback of written data.

Bit 22 – CMPP
Compare the contents of the addressed cache TAG parity to the WIP0 bit in this register. In READ mode, the register will return the result of the comparison.

Value	Description
1	Compare result is good.
0	Compare result has an error.

Bit 21 – EWP
Enable the single bit of parity to be directly written to the parity bits in the TAG RAM.

Bit 20 – EWD
Enable 24-bit address in the data register to be written to TAG RAM.

Bit 19 – CMPA
Compare the contents of the addressed cache TAG entry to the contents of the address register. In READ mode, the register will return the result of the comparison.

Value	Description
1	Compare result is good.
0	Compare result has an error.

Bit 18 – RD

Read/write four 32-bit register values to address cache RAM.

Value	Description
1	Trigger a single READ event.
0	No event.

Bits 6:0 – ADDR[6:0] Address of 128-bit Four-Word Cache Line bits

3.4.2.8. ISB Data Register

Name: ISBDAT
Offset: 0x1E88,0x1E8C,0x1E90,0x1E94

Note: User software should not attempt to write to this register on the instruction cycle immediately following a read via the ISBCMD0/ISBCMD1_7 SFRs. If this is done, the write contents will be lost due to the hardware read of this register.

Bit	31	30	29	28	27	26	25	24
	WORD[31:24]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WORD[23:16]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	WORD[15:8]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WORD[7:0]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – WORD[31:24] MSB of 32-bit Instruction Word N in ISB Line bits

Bits 23:16 – WORD[23:16] Third Byte of 32-bit Instruction Word N in ISB Line bits

Bits 15:8 – WORD[15:8] Second Byte of 32-bit Instruction Word N in ISB Line bits

Bits 7:0 – WORD[7:0] LSB of 32-bit Instruction Word N in ISB Line bits

3.4.2.9. ISB Command 0 Control Register

Name: ISBCMD0
Offset: 0x1E98

Notes:

1. The CMPP and CMPD control bits require a minimum of three CPU clocks to complete once set. There must be at least three CPU clock cycles before reading the result.
2. CHECON.ISBBUF should be set to `0` when performing diagnostics via the ISBCMD1_7 register. Otherwise, unexpected results may occur.
3. CMPD is checked when comparing the contents of the ISB Buffer to the ISBDAT data register. If CMPD and RD are both set, RD will be ignored.
4. WD and WP are active only for valid slices, in this case, Slice 1 to 3, and for Line 0 & 1.
5. For ISB DAT REG = `0`, `1`, `2`, or `3`, the write and compare operations will ignore the 4 least significant bits of ISBDAT[0].
6. With all the ISB DAT REG, users should never attempt to write a value that exceeds the maximum number of lines in an ISB.

Bit	31	30	29	28	27	26	25	24
	RIP[3:0]				WIP[3:0]			
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		CMPP	EWD	EWP	CMPD	RD		
Access		R/W/HS/HC	W	W	R/W/HS/HC	W		
Reset		0	0	0	0	0		

Bit	15	14	13	12	11	10	9	8
						LINE[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
	ISBREG[2:0]							
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bits 31:28 – RIP[3:0] Valid When Reading ISB Data Register bits

Bits 27:24 – WIP[3:0] Write Parity Data, Requires EIP = 1 bits

Bit 22 – CMPP
Compare the contents of parity bits only if ISBREG is set to 101, which equals isb_slice_data.

Bit 21 – EWD
Set to 1 when the control register is written; causes DATA bits to be written manually.

Bit 20 – EWP
Set to 1 when the control register is written; causes PARITY bits to be written manually.

Bit 19 – CMPD
Compare the contents of the addressed ISB register to the contents of data registers.

Bit 18 – RD
Read/write four 32-bit register values to address cache RAM, parity bits into ISBCMD.RIP.

Value	Description
1	READ
0	No action taken.

Bits 10:8 – LINE[2:0] ISB Buffer Line Address bits, 0-7

Value	Description
111	line #7
...	...
010	line #2
001	line #1
000	line #0

Bits 7:5 – ISBREG[2:0]
Specifies the ISB register within a slice to be accessed.

Value	Description
111	NOP
110	NOP
101	isb_slice_data
100	fifo_entry_count
011	stream_start_address
010	fifo_head_addr
001	NOP
000	nvm_prefetch_addr

3.4.2.10. ISB Command 1-7 Control Register

Name: ISBCMD1_7
Offset: 0x1E9C

Notes:
1. The ISBCMD0 register allows for diagnostics of ISB0, while ISB1 is used for buffering. ISB1 is assigned for buffering using the CHECON.ISBBUF control bit.
2. Any diagnostic accesses via the ISBCMD1_7 register when ISB1 has been selected for buffering using CHECON.ISBBUF is a user error and may lead to unexpected results.
3. The CMPP and CMPD control bits require a minimum of three CPU clocks to complete once set. There must be at least three CPU clock cycles before reading the result.
4. WD & WP are only active for valid slices, in this case, Slice 0, and for Line 0 & 1 within the slice.
5. For ISB DAT REG = `0`, `1`, `2`, or `3`, the write and compare operations will ignore the 4 least significant bits of ISBDAT[0].
6. With all the ISB DAT REG, users should never attempt to write a value that exceeds the maximum number of lines in an ISB.

Bit	31	30	29	28	27	26	25	24
	RIP[3:0]				WIP[3:0]			
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		CMPP	EWD	EWP	CMPD	RD		
Access		R/W/HS/HC	W	W	R/W/HS/HC	W		
Reset		0	0	0	0	0		

Bit	15	14	13	12	11	10	9	8
						LINE[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
	ISBREG[2:0]					SLICE[2:0]		
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bits 31:28 – RIP[3:0] Valid When Reading ISB Data Register bits

Bits 27:24 – WIP[3:0] Write Parity Data, Requires EWP = 1 bits

Bit 22 – CMPP
Compare the contents of parity bits only if ISBREG is set to 101 = isb_slice_data.

Bit 21 – EWD
Set to 1 when the control register is written; causes DATA bits to be written manually.

Bit 20 – EWP Set to 1 when control register is written; causes PARITY bits to be written manually

Bit 19 – CMPD
Compare the contents of the addressed ISB register to the contents of the data registers.

Bit 18 – RD

Read/write four 32-bit register values to address cache RAM, parity bits into ISBCMD.RIP.

Value	Description
1	READ
0	No action taken.

Bits 10:8 – LINE[2:0] ISB Buffer Line Address bits, 0-7

Value	Description
111	line #7
...	...
010	line #2
001	line #1
000	line #0

Bits 7:5 – ISBREG[2:0]

Specifies the ISB register within a slice to be accessed.

Value	Description
111	NOP
110	NOP
101	isb_slice_data
100	fifo_entry_count
011	stream_start_address
010	fifo_head_addr
001	NOP
000	nvm_prefetch_addr

Bits 2:0 – SLICE[2:0] ISB buffer slice address, 0-7

Value	Description
111	slice #7
...	...
010	slice #2
001	slice #1
000	slice #0 (NOP) NOT VALID, there is a separate control register for ISB slice 0

3.4.3. Operation

The PBU preliminarily operates on 1:1 PB, or the Fast Peripheral Bus. The PBU registers only have control to enable or disable certain PBU functions. Parameters such as ISB depth, ISB number of buffers, cache associativity, etc., are all fixed.

The CHECON.ON bit is reset to '1' by default. This provides the best CPU performance for both linear code and program flow changes. The ON bit can be cleared in software to disable most caching functions and make the PBU behave as a basic 2-deep prefetch buffer. This results in lower code performance due to longer program flow changes but still gives deterministic execution behavior, and thus, the program flow changes to longer execution time but takes a constant number of cycles.

3.4.3.1. Cache/ISB Manual Invalidation

Manual invalidation of the instruction cache and ISBs is used to force cache coherency when the user knows that the cache and Flash contents may not match. It occurs under the following conditions.

- CHECON.CHEINV control bit: When set by software, this bit will invalidate both the instruction cache and all ISBs. The hardware will clear this bit automatically after the cache and ISB memory have been invalidated.

Note: CHECON.CHEINV is also cleared should an automatic invalidation occur after the bit has been set.

- CHECON.ON control bit: The instruction cache memory and ISB buffers are invalidated when the CHECON.ON bit is cleared. This will be the case out of Reset. Execution continues using only a single default ISB slice. Setting CHECON.ON has no effect with respect to cache/ISB invalidation as it is already invalidated and the active ISB will contain valid data from the current instruction flow.

3.4.3.2. Cache/ISB Automatic Invalidation

Automatic invalidation of the instruction cache and ISBs is used to ensure cache coherency when the device knows that the cache and Flash contents may not match. It occurs under the following conditions:

- Flash write operation: Automatic invalidation only occurs if the Cache Coherency Control bit (CHECON.CHECOH) is set (default) and Flash is programmed/erased. This is only applicable for writing to the active panel in dual-panel devices.
- Parity error: Refer to PBU Data Error Handling for further details. Only the accessed cache line of the ISB buffer is affected, and the remainder of the cache memory does not need to be invalidated.

3.4.3.2.1. Cache Invalidation when Writing to Flash

Whenever the Flash is written, the user has the option to automatically invalidate the instruction cache and ISBs using the CHECON.CHEOH control bit. If instruction data are being written to Flash, invalidation ensures Flash memory contents remain synchronized with the cache and ISB contents.

Note: To fully ensure a correct operation, it is recommended that the final instruction that initiates Flash programming be followed by four NOP instructions to flush the instruction pipeline. This ensures coherency since the remaining instructions in the CPU pipeline will take no action.

3.4.3.3. PBU Data Error Handling

The PBU handles error correction in two ways. First, any data errors that originate from the program memory are tracked. Secondly, internal PBU data errors that may occur while program data are stored in the PBU Cache RAM are monitored.

3.4.3.3.1. Program Memory Data Errors

Error status is captured from the program memory and buffered along with the fetched program data word in the ISB. Consequently, each line in the ISB has 129 bits: 128 bits of data and one bit for error status. The error status bit indicates the data read from the program memory are unusable and incorrect. The program memory data can be invalid for multiple reasons, including an uncorrectable ECC error and a security violation that would suppress the data. In any case, the data are not a valid CPU instruction and should not be executed by the CPU.

The PBU does not generate any kind of event, trap or interrupt when bad data are fetched from the program memory. This is because the ISB may speculatively fetch data that would never be executed by the CPU. Secondly, the CPU may speculatively fetch instructions from the PBU during conditional branches that may never get executed.

A bus error signal is passed with the program data to the CPU for instructions fetched from the PBU. If the program data are invalid with the bus error signal asserted, then the CPU can suspend execution in the pipeline and cause a trap event to occur.

3.4.3.3.2. Cached Data Errors

The second method of PBU error handling occurs when the cache has detected a parity error on a cached line of program word data. When valid program data are cached for later consumption, then the error status bit is stripped, and the program data word is stored in the cache memory. A single even parity bit is calculated and stored along with the data. This parity bit is used to protect the system from data corruption that could occur in the cache RAM.

A maskable interrupt event is generated by the PBU when a parity error is detected on a cache line. In this case, the cache will invalidate the line with the parity error, and the program data must be refetched from the program memory. Other than the interrupt event, the only other effect that can be observed during a cache parity error is additional execution latency caused by Flash program fetch. No address associated with the parity error is captured.

3.4.3.3.3. Corrected Program Memory Data Errors

The program memory error correction logic may correct a data error in the program data supplied to the PBU. These corrections are not reported to the PBU which is usually done for the uncorrectable errors. Specifically, a single-bit corrected error (SEC event) is not reported to the PBU.

The program memory is responsible for tracking and reporting the corrected event. These actions serve as a warning to the system software that integrity of the NVM data may be failing.

3.4.3.4. Cache Fault Injection

A single-bit error can be injected on any of the data bits of the cache line or the associated parity bit. The error injection is performed by XORing the data read from the cache line with a '`1`'. Since the PBU can cache program data from a variety of address locations depending on the program flow, it is impractical to perform error injection for a particular program memory fetch address.

The PBU error injection, when enabled with the FLTINJ bit, will cause a one-time error injection the next time the cache memory is accessed by the CPU. The CHESTAT.PAR bit will indicate when the error injection has been performed. At this time, the PBU will also signal that an integrity error has occurred by creating an interrupt event. The user will not be able to determine which line of the cache buffer caused the event and fetch address.

A write to the FLTPTR register while FLTINJ = `1` will have the effect of resetting the Fault injection. This will help facilitate a software test routine that cycles through an error injection on each bit.

3.4.3.5. Non-Cached Events

Certain fetches from the NVM are not cached. These include:

- Interrupt Vector fetches
- Fetches of debug executive code
- Fetches of invalid program memory data

All these types of fetches are not cached to avoid cache thrashing. Thrashing occurs when other useful data are evicted from the cache and replaced with less useful or invalid data. Inhibiting caching during the above fetches is expected to improve the overall efficacy of the cache, resulting in more cache hits at run time.

Interrupt Vector fetches are a special type of non-cached event. Specifically, only one program word is fetched from the NVM when the CPU indicates a vector fetch and the ISB is bypassed. When an interrupt occurs, the interrupt vector address is fetched from the vector table, then the instruction at the interrupt vector address is fetched. There is no need for an ISB to perform a prefetch and fetch the program word after the one that contains the interrupt vector address. This would be wasteful and produce extra latency in the servicing of the interrupt event.

The PBU monitors whether the CPU is executing user code or debug executive code. Instructions fetched from the debug executive code are not cached. This avoids additional indeterministic behavior when code execution transitions from the debug executive code back to user mission-mode code.

In addition, the CPU supports execution from RAM and a RAM-based Interrupt Vector Table (IVT). Program or vector fetches from RAM are also non-cached events. However, this capability introduces the possibility of both a vector and its associated handler routine being in either NVM or RAM. Whenever the IVT and/or an exception handler (interrupt or trap) are located within RAM, this is treated as a special case by the PBU to maintain efficient operation.

3.4.3.6. PBU Performance Monitoring

Each word of data requested on the CPU instruction bus will be sourced either from the ISB or the instruction cache and not the external NVM. This ensures that each fetch of program data can be completed in minimal time, which maximizes application performance.

Program data that are not already present in the ISB or cache must be fetched from the NVM, which takes additional cycles and decreases overall application performance. Once program data in a particular NVM program word has been consumed by the CPU, it is stored in the instruction cache for later use. The exceptions to this are program data with uncorrectable errors, security violations or debugger executive program data. Once stored, the program data will be available for later reuse in the instruction cache until those contents are erased and replaced with another program data word.

3.4.3.6.1. Cache Busy Cycles

The program word is cached on the cycle following the fetch from NVM. During this time, the IC will be busy because of the write-to-cache memory. If the CPU requests program data during this cycle, the PBU will check the contents of the ISB for an address tag match. In most cases, the data may be sourced from the ISB while the IC is busy. This results in an ISB hit and no extra cycle penalty is incurred. When the IC is busy and no ISB hit occurs, then an extra cycle of latency will be inserted while the PBU waits for the IC write cycle to complete. Then, the IC address tags are checked for an IC hit.

3.4.3.6.2. PBU Performance Event Outputs

The PBU has event outputs that can be connected to external performance counters at the device level for characterization of the PBU performance. These events can be counted over a period of application execution and compared with the total number of executed instructions and/or the total number of elapsed clock cycles to get a measurement of the PBU efficacy. The performance event signals available from the PBU include:

- Instruction cache “hit” event
- Instruction stream buffer “hit” event
- PBU “hit” event
- Instruction cache “busy” event

The IC hit event indicates when a particular instruction was fetched from the cache memory. The ISB hit event indicates when a particular instruction was fetched from the ISB. This generally happens on the second fetch from a program word while that word is written to the cache memory.

The PBU hit event is of most interest for PBU performance analysis. This event signal is the logical OR of the IC hit and the ISB hit events, and it indicates that the PBU was able to source the requested data without initiating a new NVM fetch.

The IC busy event is used to count the number of extra cycles that were inserted when the ISB could not source the requested data and a Wait state was necessary to determine if the data were available in the IC. An IC busy event is expected to be infrequent and would occur during program flow changes.

3.4.3.6.3. Factors Affecting PBU Efficacy

PBU efficacy is not a constant value. For a given code segment, such as function call, the efficacy of the PBU will be very much dependent on these factors:

- The code that was executed prior to a given code segment
- The size of the code
- The specific location of this code in memory
- Flow changes that occur during the execution of a specific code segment

Different performance results are possible when a specific segment of code is executed in one context vs. another context. The prior code executed will determine what code data are present in the cache memory. The prior code may have evicted all program data associated with the segment of interest. However, if the segment of interest is repetitively executed, then there is a strong possibility that program data associated with this segment will remain in the cache memory without eviction.

In general, a small segment of code, which is repetitively executed, will produce the best PBU performance results. This is because the code size is small enough to fit within the cache memory, and the repetitive nature of the code will maximize the reuse of the cache contents with a minimum of evictions. The absolute location of a code segment within memory will impact the PBU performance. This is closely related to how the code is compiled, optimized and linked during the software development process.

Two different program data words in a segment of code could have the same address tags. If these program data words are executed often, then numerous cache evictions and NVM fetches will result during code execution. A larger cache memory and/or increased cache associativity can both help this issue. A larger cache memory increases the number of available address tags, while increased associativity increases the number of location options where a specific program data word could be stored. The more flow changes that occur in each segment of code, the higher the possibility that PBU performance will be reduced.

3.4.3.7. TAG Memory Parity

The TAG (address table) is currently constructed with register memory. Changes to this memory will be enhanced with an odd parity bit for each stored address. Odd parity will have a parity bit value of 0 if there are an odd number of 1 and 0 values. This means a default data value of 0x0000 will have the parity bit set to 1.

3.4.3.7.1. TAG Operation

A cache miss will cause an NVM fetch of a 128-bit cache line into cache RAM. Along with the 4 instructions fetched, a TAG address will be stored. At the time of TAG memory storage, an odd parity bit will be added to the tag memory. This parity will be used when scanning a TAG memory looking for a CPU match address.

When a cache MISS is detected, new data will be fetched from the NVM (flash memory) and the address TAG is written into TAG memory. This write operation will have an odd parity bit generated, concatenated to the address, and stored with each saved address in TAG memory. An odd parity bit has been selected so that a Reset-initialized register value of zero will yield a parity Fault if addressed.

The TPE (TAG Parity Error) bit will be set in the status register. The TAG Parity Error bit will remain set until it is cleared in the status register.

3.4.3.7.2. Clearing TAG Parity Errors

BIT Clear or BCLR, or a write with the correct bit set. The error bit is cleared at hardware Reset. There is no auto-clear function.

3.4.3.8. Cache and ISB Instruction Parity

3.4.3.8.1. Cache Mode

The instruction memory in the NVM Flash will insert odd parity on each of the 4 - 32-bit instructions held in a cache line (128 bit plus ECC parity).

Per instruction word parity bits will be encoded and inserted into the data going to the cache and will not be covered by the ECC parity. By placing the insertion of the per instruction parity in the NVM, the Flash panel will not be changed.

Once the per-word (instruction) parity is inserted into the 32-bit data, it will be stored in the Cache RAM (and ISB). The 33-bit instruction will be passed through the instruction fetch unit in the dsPIC33A. If a parity error is detected in the dsPIC® instruction fetch unit, it will kill the instruction.

3.4.3.8.2. ISB Mode

The cache can be operated in a streaming for FIFO mode. This is a prefetch that reads a 128-cache line from the NVM Flash and feeds the instruction fetch unit of the dsPIC33A. This mode must also use the parity bit on the 32-bit instruction data to be consistent. The parity bit will be inserted, and the stream buffer is written. The inserted parity bit will be passed to the dsPIC33A instruction fetch unit. The parity bit can be checked in streaming mode as it is passed to the dsPIC33A; however, the only action to take on a parity error is to set the error flag in the status and raise an error event.

There is also SFR-based access to the stream buffers in each of the 8 slices. There is odd parity on each of the 32-bit instructions and on the address registers.

3.4.3.8.3. Instruction ECC and Parity Scheme

Figure x-x graphically depicts the error correction scheme used in the Flash, PBU and CPU instruction buses. The diagram assumes a 128-bit Flash instruction word size, but note that this could differ depending on the specific device.

3.4.3.9. SFR-Based Memory Access

The following interface provides a method to understand both cache values (four 32-bit words), the associated TAG addresses (one 32-bit word) for the cache, and ISB stream buffer registers. This interface will also provide a means to compare a given value in each of the memory areas against a value held in a register for testing and hardware validation.

The data register is used to write the instruction data to the cache RAM, and it will also serve as the compare value register. The hardware will generate the EVEN parity bit and use it as part of the comparison operation. If a comparison fails, the current values in the data/comparison registers can then be written back to the cache for continued testing.

Parity Fault assertions can be generated by the use of a control bit that forces the parity bit to an odd parity.

3.4.3.9.1. SFR Operation

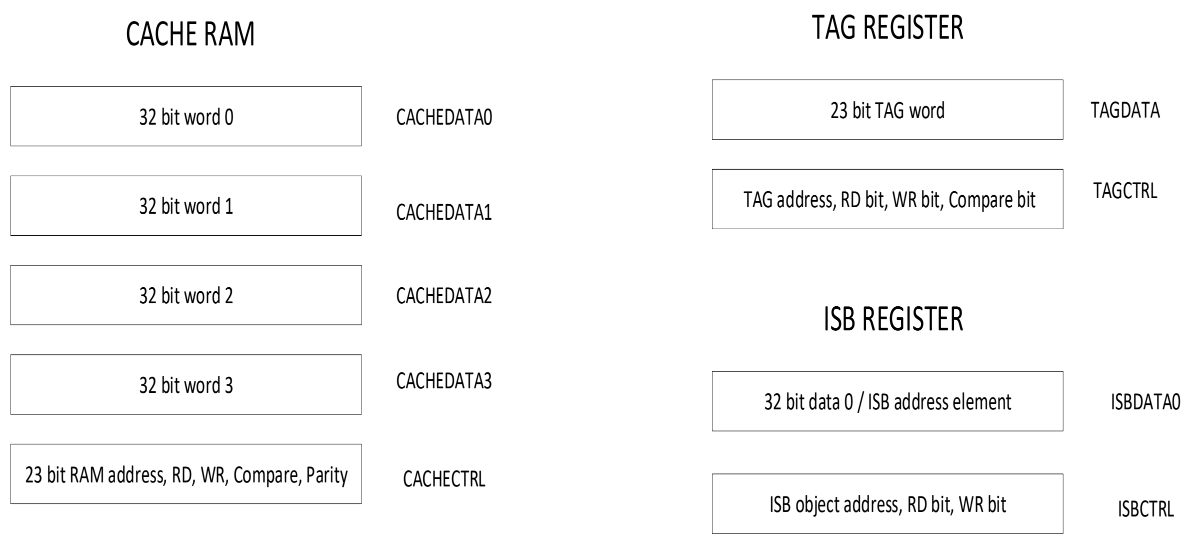
In this SFR interface, the values for the cache and tag memory are written first and are then transferred to the associated memory when the cache or tag control register is written. The control register contains the READ, WRITE and COMPARE directives along with the destination address. This allows all 128 bits of cache data to remain coherent. The stream buffer registers will be written in a similar fashion, where the register element data will be written first and set into hardware when the ISB address/control register is written.

When reading from the cache or tag memory, the address/control register is written to with the READ bit set. This action will cause the contents of the address memory to be placed into the associated data registers for that memory unit.

Additionally, a data comparison value via the DATA registers set is provided that allows the contents of a cache or tag memory to be compared with a stored value. The result of the comparison will set or clear the Fault flag for the operation in the cache status register. The comparison bits will be separated from a normal Fault, so the result of the comparison operation is not affected by the next normal access to the cache memory.

Writing to the Address/Control register (CACHECTRL) will clear previous Faults in the status register for this action. The Fault bits will not be set by the action of writing unless the address of the cache is out of range. For the data being written or read to be correlated to the correct address, it is up to the user to determine that the TAG address/location is correct for the accessed cache RAM location.

Figure 3-22. CACHE RAM Operations



3.4.4. PBU Module Description

3.4.4.1. Overview

The cache provides an improved instruction rate to the CPU from Flash program memory. The cache increases the instruction rate in two ways. First, program data words are stored in a prefetch buffer. This allows the CPU to use program data from one buffer location while a fetch from Flash memory fills other buffer locations. Secondly, it increases the instruction rate by storing (caching) program data words previously fetched from the program memory for later use. Specifically, the instruction rate is improved because the fetch from the local cache memory takes less time than a fetch from the program memory.

The PBU responds to program data fetches from the CPU instruction bus and fetches the necessary program data from Flash memory. Fetches from the CPU data bus and data fetches from other bus masters are sourced through the data path and buffer in the Flash memory.

The cache requested 32-bit program data can come from multiple sources:

- Instruction Stream Buffers (ISB) that serve as the prefetch buffers
- Instruction Cache
- Flash memory if the requested program data is not available from the above sources

When Flash access time is fast and the Flash data word is large enough to contain multiple instructions, a prefetch buffer alone can suffice to support the CPU instruction rate without stall cycles. However, the Instruction Cache is particularly helpful during program flow changes. At these times, the prefetch buffer will not have the required program data, and a new fetch must be started at a new location. The Instruction Cache helps eliminate extra latency associated with Flash fetch at a new program flow location.

3.4.4.2. SFR Interface for Module Operation

The PBU SFR interface provides control and status around the Cache and ISB; it also provides event signals to the CPU, such as memory integrity errors. The operation of the PBU is divided into three operating scenarios:

- Operation when the PBU is disabled (CHECON.ON = 0)
- Normal cache and ISB operation while the PBU is enabled (CHECON.ON = 1). This is the default mode of operation.
- Non-cacheable events that occur while the PBU is enabled

These operating scenarios are described in Module Operation When Cache Disabled, Module Operation When Cache Enabled and ISB Operations.

3.4.4.3. Module Operation When Cache Disabled

When the CHECON.ON bit is 0, all cache operations and advanced ISB operations are disabled. The PBU behaves as a single 2-level prefetch buffer. The single prefetch buffer provides one location for the CPU to consume program data. The appropriate 32-bit portion of the 128-bit program word is supplied to the CPU based on address bits 3:2 of the PC address. The other location in the prefetch buffer predictively fetches the next sequential program memory location (PC + 16).

If no program flow changes occur, the 2-level prefetch buffer is able to keep pace with linear program flow without wait states. This assumes that each fetch from program memory takes 4 clock cycles, and each program word contains at least 4 instructions.

Each line of the prefetch buffer is invalidated when the CPU has consumed the last instruction word (highest address) in the line. If a program flow change occurs outside of the current program data word, the prefetch buffer is invalidated, and a new fetch begins at the requested address.

The number of clock cycles required to fetch the new target address will vary depending on which instruction word in the prefetch line caused the flow change. This is because a prefetch that is already in progress from Flash memory will not be aborted. Therefore, the prefetch buffer must wait for the fetch in progress to be completed before the fetch at the new target location can begin.

3.4.4.4. Module Operation When Cache Enabled

The PBU will be preconfigured so that the Cache functionality is fixed to Instruction Cache mode, which is the only type of operation described in this section.

3.4.4.4.1. ISB Buffers

The ISB provides one or more prefetch buffers (Slices) for Flash program memory. Each ISB requires two storage locations - one for the CPU/Cache to use, and another to allow a prefetch of the next Flash word. Each ISB buffer has address registers, status tracking registers and a FIFO buffer. The FIFO buffer is 129 bits wide. Of the 129 bits, 128 are used to hold the program data word read from memory. The remaining bits are used to indicate the data error status that occurred when reading from the external program memory.

The PBU is configured to have an ISB buffer depth of two. The A two-word depth allows the ISB to prefetch a second word while the first word is used by the CPU.

Each entry of an ISB buffer also includes an internal valid status (data valid) bit to determine whether the FIFO has usable data. This is necessary to ensure a buffer value cannot be used unless previously loaded. Conversely, if a data value is prefetched from Flash but subsequently not required by the CPU, it will not be stored in the target ISB buffer, which will remain unchanged.

The PBU ISB is configured for four slices.

Each slice includes an address valid bit. This bit is cleared upon Reset entry or whenever the slice is initialized and set when the ISB slice addresses are loaded. This functionality ensures that only valid addresses are ever present in the address registers (to prevent a false hit out of Reset).

Each line of the ISB includes an internal Data Valid bit to ensure a line cannot be used unless previously loaded. The state of this bit ultimately drives the data ready signal to the CPU to indicate when valid data is being presented on the instruction data bus. The Data Valid bit is also used to invalidate an ISB line in the event of a context change.

In some scenarios, like the execution of speculative instructions, a fetch may be started by the CPU but will later be abandoned. In such cases, the ISB asserts an internal Discard Prefetch flag to prevent the prefetch from being stored when the data is not needed by the CPU, thereby freeing up that ISB slice for subsequent use.

3.4.4.4.2. ISB HIT and LRU Logic

Each ISB buffer slice can have multiple lines of program memory, but only the top word of the FIFO is visible at any given time. An internal address tag register associated with each buffer slice indicates the address of the top word. Only this address is used to determine an ISB "hit" or "miss." If the desired program word address is available in a deeper location of the FIFO, it will not be detected by the hit logic.

As each program word is consumed from an ISB buffer, the address tag register is incremented to point to the next 128-bit program word. This new address will then be used for subsequent address hit comparisons. Therefore, linear program flow will naturally result in a series of ISB hits as a complete program word is fully consumed, and then the FIFO address is incremented to point to the next program word that was prefetched.

If a program flow change occurs and a cache miss results, the address tag associated with each ISB slice must be considered to determine an ISB hit. If one of the ISB address tags matches, the requested program data will be sourced from the ISB slice that had the address tag match (hit).

The new ISB slice will continue to be used until the program flow changes again. If a program flow change occurs along with a cache miss and an ISB miss, the ISB employs a least recently used algorithm utilizing the LRU status bits associated with each buffer slice to determine which ISB buffer slice will be used for the new flow. A new program memory fetch will be started by the newly selected ISB slice to get the requested data.

Active Buffer Selection Policy

In order to maximize the efficiency of the ISB slices and to avoid discarding useful (or storing useless) data, several criteria are applied when selecting a new ISB slice for program flow.

To assist with this process, an internal prefetch valid flag is used to determine whether a prefetch is underway. It is asserted when the prefetch request from the PBU is granted by the NVM. It is negated when the prefetch completes, at which point the data will be stored in the ISB.

1. When there is **no** address match on any of the buffer slices and **no** SSA match on any of the buffer slices, pick the LRU buffer as the active buffer.
2. When there is an address match on **more than one** buffer slice and data is either valid on **all** or **more than one** of those buffer slices, pick the active buffer as the one with the **lowest** LRU value that has its data valid.
3. When there is an address match on **more than one** buffer slice and data is valid on **only one** of those buffer slices, pick the active buffer as the one which has its data valid.
4. When there is an address match on **more than one** buffer slice and data is **not** valid on **any** of those buffer slices but prefetch_valid **is set** on any one of them, pick the active buffer as the one that has its prefetch valid set (because it will be available for use faster).
5. When there is an address match on **more than one** buffer slice and data is **not** valid on **any** of those buffer slices and prefetch_valid is **not set** on any of them, pick the active buffer as the one with the lowest LRU value that has an address match.
6. When there is an address match on **only one** buffer slice, but data is **not** valid on that buffer slice and prefetch_valid is **not set** on that buffer slice, pick the one that had an address match as the active buffer.
7. When there is **no** address match on any of the buffer slices but an SSA match on **any one** buffer slice, pick the buffer slice that had an SSA match as the active buffer. This will help reduce thrashing of the ISB buffer slices. This behavior will improve ISB performance when a certain section of code is looped repeatedly. The same ISB will get reused when the code flow branches to the beginning of the loop instead of overwriting the contents of another ISB buffer slice.
8. When there is **no** address match on any of the buffer slices but an ISB slice error on **any one** or multiple buffer slices, pick a buffer slice that had an error as the active buffer. This buffer

slice was preloaded with cache data that failed parity check and hence these data are unusable. Therefore, the slice may as well be reused for another instruction flow.

3.4.4.5. ISB Operations

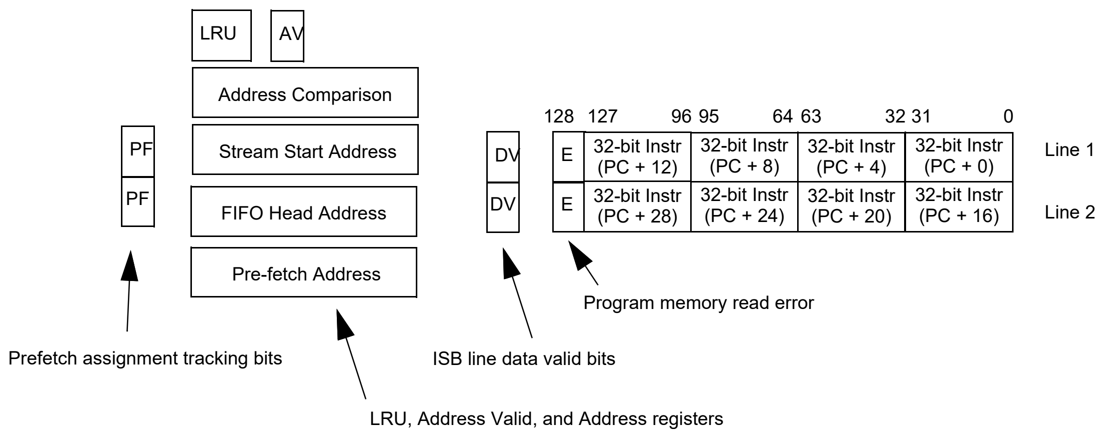
The ISB has these components:

- N buffer slices
- Active buffer selection logic
- ISB hit/miss detection logic
- Cache write logic
- NVM interface logic
- LRU logic

3.4.4.5.1. Buffer Slice Architecture

- Address valid register
  – Indicates that a valid address has been stored in all ISB slice address registers - FIFO head address, SSA address, etc., preventing a false hit coming out of Reset. Keeps ISB from thrashing in multiple loop scenarios. The same buffer slices are used in the repetitive loop.
- Stream start address register (SSA)
  – Stores the start address of an instruction stream. The SSA address stays the same as the FIFO head address changes [(ADDR_WIDTH-1):4].
- FIFO head address register. This is compared to the CPU address to generate a hit signal. The FIFO head address is incremented under two scenarios.
- NVM prefetch address register
- Address comparators
- ISB data FIFO
- ISB prefetcher - tracks where the data needs to go - status bit appended to each FIFO - each FIFO must set a status bit - there should be only one assignment bit set indicating where NVM read data should go. NVM can latch 2 addresses at a time. Time is needed for ECC calculation, so it can take a new address in the third cycle and start the next fetch. There are only two PF bits per slice because two fetches can be pending from Flash at any given time. These bits indicate where the first fetch and second fetch go.
- LRU register
- BTIC second word flag
- ISB slice error register - BTIC mode - BTIC loads from cache to ISB with parity error
- Prefetch-N counter

Figure 3-23. Instruction Stream Buffer - One Slice



Note: Any of the 32-bit instructions shown could also be a pair of 16-bit instructions, allowing up to 8 instructions per 128-bit program word.

3.4.4.6. Cache Buffer

The Cache Buffer is internally configured to provide a variable size (at compile time) and 4-way set associative operation.

The Instruction Cache has memory elements to store these data:

- One 128-bit program memory data word per line, plus a parity bit for each word
- Address tag for each line of cache
- Valid bit for data word (128-bit) in each line

Parity calculation and error detection on a chosen cache line are shown in Figure 3-26.

For an Instruction Cache line to be valid, the valid bit must be set, and the even parity check must pass. If a bit error occurs in the cache memory causing a parity error, recovery is made by invalidating the cache line associated with the error. The cache line can simply be re-fetched from program memory, resulting in additional latency caused by the new fetch.

Figure 3-24 shows how program data are organized within the Instruction Cache line. The Tag address is based on the upper address of the program word, as depicted in Figure 3-25.

Each program word in the Instruction Cache line stores at least four instructions, as shown in Figure 3-24. The program word can store more than four instructions if some of the instructions are 16-bit instructions. If all instructions are 16-bit instructions, then the program word will contain eight instructions. In normal program instruction flow, the typical program word will contain a mix of 16 and 32-bit instructions.

Figure 3-24. Instruction Cache Data Organization

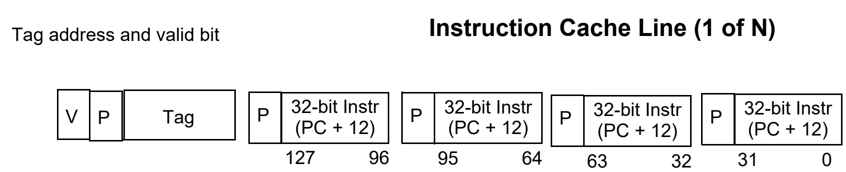
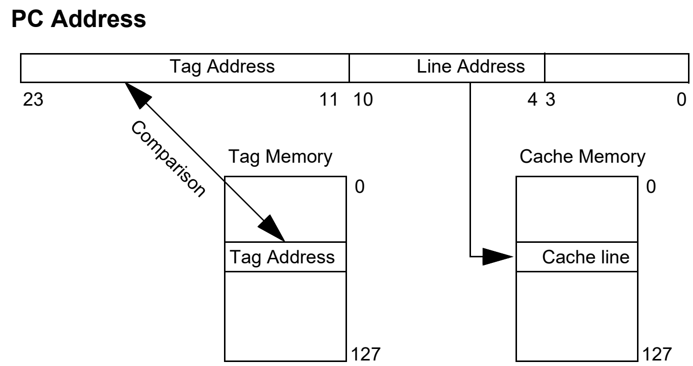


Figure 3-25. Cache Line Addressing (128-Line, Direct Map)



3.4.4.7. Cache HIT Logic

Each line of the Instruction Cache memory has a Tag address register based on the number of cache lines. The number of bits in the Tag address depends on the number of implemented cache memory lines. The Tag address is derived from the most significant bits of the CPU program fetch address.

Figure 3-25 shows how the Tag and cache line addresses are derived from the PC address for a direct-mapped and set-associative cache, respectively.

Bits 3:0 of the program fetch address are not used to form the Tag or cache line address since the cached data size is 16 bytes (four 32-bit words).

3.4.4.8. Program Data Flow in Cache Mode

The CPU program bus has access to both the ISB and the Instruction Cache during a program data fetch. Note that the program data will always be sourced in the order listed. There are these general cases that can occur during a fetch of a program data word:

- The required program data is available in the cache memory (cache hit)
- The required program data is available in one of the ISB buffers (cache miss, ISB hit)
- The required program data is not available in either the cache memory or the ISB buffers (cache miss, ISB miss)

3.4.4.8.1. Cache Hit

If the program data required by the CPU is available in the cache, the cache will source the data. The ISB takes no action in this case, except possibly to complete an instruction fetch related to earlier fetch events.

3.4.4.8.2. Cache Miss, ISB Hit

In the event of a cache miss, the data will be sourced from the ISB buffer with a matching address tag.

Furthermore, the Cache will become busy for one cycle after the ISB hits while the data is transferred from the ISB buffer to the cache memory. The cache memory cannot be accessed by the CPU while data are written from the ISB to the cache. This could result in an unintentional cache miss on the following fetch cycle. However, the necessary program data will be available in the ISB so no stall cycles will occur due to this behavior. After the program data word is transferred to the cache memory, the ISB line will then be invalidated so that future fetches from that program word will occur from the cache memory and not the ISB.

3.4.4.8.3. Cache Miss and ISB Miss

If the program data is not available in the Instruction Cache or an ISB buffer, the PBU must then begin a new fetch from Flash program memory. The PBU waits for any existing fetches in progress to complete before starting the new fetch.

An LRU algorithm is used to determine which ISB slice is used for the fetch if more than one slice is implemented.

When the requested program word is available in the ISB, a cache write cycle is initiated to store the data in the appropriate Instruction Cache line as described in the prior Section and in Section 3.6.8.

3.4.4.9. Interrupt Vector Fetch

When an interrupt occurs, the interrupt vector address needs to be fetched from the interrupt vector table. A signal is used to notify the PBU that a vector fetch is taking place. Only a single fetch is performed; no prefetch occurs.

3.4.4.10. Cache Busy Cycles

Each time program data is consumed by the CPU from the ISB that is not in the Instruction Cache, the program word will be written to the Instruction Cache on the following clock cycle. If the CPU also needs instruction data on the next clock cycle, the instruction data will be sourced from the ISB because the write to the Instruction Cache is in progress. This will result in a cache busy cycle due to the cache write.

On the clock cycle following a cache write, the ISB line will be invalidated because the program data are now stored in the Instruction Cache. This frees the ISB line for future prefetch activity.

3.4.4.11. Program Memory Fetch Errors

When the PBU fetches a program data word from the program memory, a data error may be signaled from the program memory as part of the data word to the PBU for one of the following reasons:

- A double-bit, uncorrected ECC error (DED)
- A security fetch error

These errors may not have significance to the CPU execution, so it is important to delay any kind of error response until it is known if the CPU will use the data. This cannot be determined until after the program data has been fetched by the CPU in some cases. Therefore, it is the responsibility of the CPU to detect and respond appropriately to a data error. Unless the CPU fetches a bad instruction or data and decides to commit the instruction for execution, data errors from the program memory will remain quiet.

In some scenarios, the ISB may predictively prefetch data from invalid areas of program memory that the CPU would never request. This may happen, for example, when the ISB predictively prefetches beyond the boundary of a programmed region of memory. In this case, the data would never be cached because the CPU never attempted to consume it, and all error event signaling would remain quiet.

In other cases, the CPU may fetch invalid data. When invalid instruction data are fetched by the CPU, a bus error signal is asserted so that the CPU can detect that the data are not valid. This scenario typically occurs when the CPU predictively fetches an instruction while making a conditional branch decision. The predicted branch path may be invalid for the current state of the application. For example, the current program memory security settings may not allow a fetch from the predicted path at the present time.

A decision to take the predicted execution path is made later in the instruction pipeline, but before the instruction is committed. If the predicted execution path is not taken, then the CPU pipeline is flushed, and the error should remain quiet. If the predicted execution path is taken, then the CPU must suspend the instruction pipeline and generate a trap event because the current instruction data is not valid.

3.4.4.12. Module Operation During Non-Cached Events

When the PBU is enabled (CHECON.ON = `1`), certain fetches from program memory will not be cached and may result in the generation of an error event depending on the cause. The following fetches from program memory are not cached, even if the data is consumed by the CPU:

- A program memory data error
- A fetch of an interrupt vector address
- A fetch when the CPU is in debug mode
- An interrupt vector fetch

3.4.4.12.1. Program Memory Data Error

The program memory may signal a data error for multiple reasons. See Program Memory Data Errors for more information. In any case, the data word fetched from program memory is not valid. Therefore, it makes no sense to cache the invalid data.

3.4.4.12.2. Interrupt Vector Fetch

Interrupt vector fetches are not cached. The vector fetch will occur as soon as the current program memory fetch in progress (if any) has been completed.

The ISB is effectively bypassed for a vector fetch, and the fetch address is forwarded to the program memory directly. The contents of all ISB slices are preserved during the fetch, and no re-assignment to a new ISB slice occurs. A single program memory word is fetched during the vector fetch, and no predictive prefetching is performed as part of the vector fetch.

After the vector fetch has been completed, program execution will continue and appear as a program flow change as the CPU presents the previously fetched vector address. It is also possible that an interrupt occurred which is of higher priority. In this case, a new vector fetch address will be presented to the PBU.

3.4.4.12.3. Debug Mode

The user's code could be evicted from the cache memory after a breakpoint is taken and the debug executive code runs. However, the ISB continues to operate normally while the PBU is enabled, and the CPU is in debug mode.

3.4.4.13. Cache Coherency

Coherency between cached values in the PBU and Flash must be maintained after Flash write events, a panel swap BOOTSWP operation, and in the event of a cache/ISB parity error. This is achieved through automatic instruction cache and ISB invalidation, though it is optional during Flash programming, and only affects the faulted cache line or ISB in the event of a parity error.

The contents of the cache memory and ISBs can be invalidated manually by setting the CHEINV bit. If the application is operating from Flash memory, then a stall will occur until all contents of the PBU are invalidated and the next program data word can be fetched from Flash memory.

3.4.4.13.1. Cache Coherency After Flash Write Events

The PBU has information on what Flash data were programmed or erased. Therefore, the PBU will monitor the appropriate signals from the NVM Controller and invalidate all cache and ISB buffers. Specifically, the NVM Controller provides a signal that is pulsed whenever the NVMCON.WR bit has been set and the corresponding Flash write addresses a panel that is currently being read to supply instructions to the CPU (Active panel in dual panel devices). This indicates that a Flash write sequence was initiated, after which the PBU should no longer expect Flash contents to match cached values in the PBU.

Automatic coherency control is optional for the PBU. If CHECON.CHECOH = `1` (default) and the NVM Controller pulses, all Instruction Cache lines and all ISB lines that are currently marked as valid will be invalidated. Any subsequent fetches from the CPU instruction bus should be fetched directly from Flash memory. If CHECON.CHECOH = `0`, automatic coherency control will not occur.

3.4.4.13.2. Cache Coherency in the Event Of a Cache Parity Error

The cache will automatically invalidate the affected cache line. Refer to Cached Data Errors. Cache line automatic invalidation in the event of a parity error cannot be disabled. Consequently, the state of CHECON.CHECOH will have no effect in this situation.

3.4.5. Integrity Checking

The device RAMs may be susceptible to faulty events. Therefore, only data that is stored in RAM or register file arrays will have integrity checking.

3.4.5.1. ISB Integrity Checking

The ISB address tag registers, LRU bits, valid bits, FIFOs and storage of program data in these buffers are usually short-term. Therefore, no error correction logic is present on the ISBs.

3.4.5.2. Instruction Cache Parity

Parity calculation and error detection on a chosen Instruction Cache line are shown in Figure 3-26. This error detection scheme assumes that address tag data, valid bits, and LRU status are implemented in registers. Therefore, error detection is provided on the RAM elements of the Instruction Cache only.

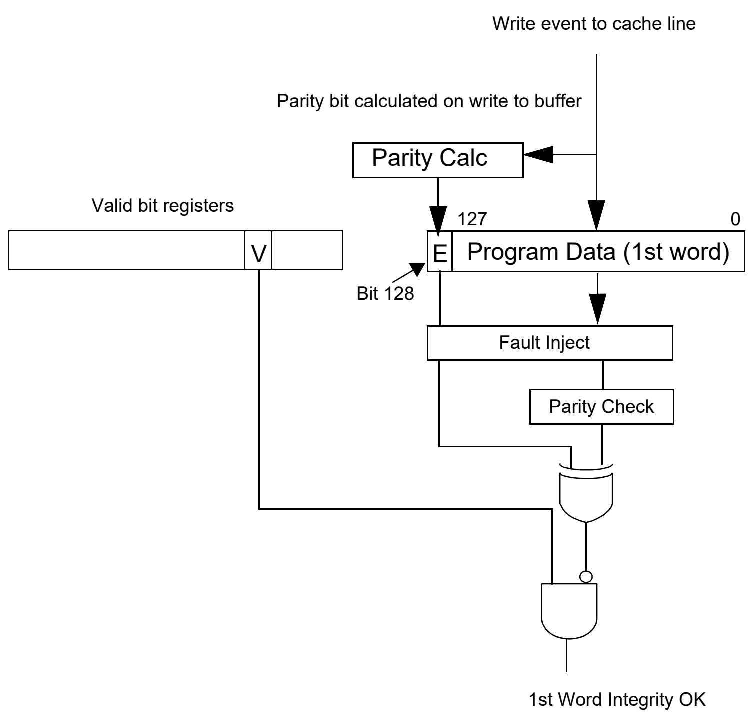
For a cache line to have integrity, the valid bit must be set, and the parity check must pass. If a bit error occurs in the cache memory causing a parity error, recovery is made by invalidating the cache line associated with the error. The cache line can simply be re-fetched from program memory, resulting in additional latency caused by the new fetch.

In the event of a cache hit and an integrity error, an interrupt event is signaled to the CPU, the cache line is invalidated, and a re-fetch is initiated from Flash memory using the appropriate ISB buffer slice according to the LRU algorithm.

3.4.5.3. CPU Program Fetch Aborts

If the PBU has read an uncorrectable value from the PBU or SRAM memory (DED status is 1), then the data should not be consumed by the CPU. In this case, the PBU or SRAM will indicate that the data is not ready, which will stall the CPU. At that time, a trap event for an uncorrectable DED error should occur. The trap event will abort the bad data fetch and cause the CPU to vector to the trap handler.

Figure 3-26. Instruction Cache Line Parity



3.4.6. Operation During Exception Processing

In most application codes, the Interrupt Vector Table (IVT) and all exception handlers will be in the NVM. However, the dsPIC33A system architecture supports RAM-based IVT and/or exception handlers to reduce exception processing latency and improve handler execution efficiency. Consequently, the PBU must be able to accommodate all such permutations, as discussed in the following sections.

3.4.6.1. NVM or RAM Based IVT and/or Exception Handler

The PBU uses signals to indicate whether the vector fetched from the Interrupt Vector Table (IVT) during CPU exception processing is from NVM or RAM.

If the IVT is determined to be RAM-based, the PBU must then determine if the exception handler is RAM or NVM-based and act accordingly. If the IVT is determined to be NVM-based, the PBU will either start prefetching the exception handler instruction stream (NVM-based exception handler) or stall (RAM-based exception handler) and prefetch from the most recently active NVM instruction stream.

In all exception return scenarios, usually the first NVM instruction after the return will have already been prefetched and be available within an ISB. If not, the PBU will stall the CPU while it fetches the return target instruction, after which the CPU will continue execution as normal.

3.4.6.2. IVT in NVM

During exception processing, the BMX asserts CPU memory fetch and CPU memory clear. The PBU will recognize this combination as indicative of an IVT vector fetch from NVM and will fetch the vector data in 4 to 7 cycles.

- Exception Handler in NVM:
  If the corresponding exception handler is in NVM, the CPU will issue an NVM PFC during exception processing to commence execution from the exception handler. The PBU will see this PFC event and begin prefetching the exception handler instruction stream.
- Exception Handler in RAM:
  If the corresponding exception handler is in RAM, the CPU will issue a RAM PFC during exception processing to commence execution from the exception handler. Because the PFC is to RAM, the BMX will stall the PBU after the IVT fetch completes. While stalled, the PBU will recommence prefetching from the most recently active NVM instruction stream.

3.4.6.3. IVT in RAM

During exception processing, the BMX asserts CPU memory fetch and CPU memory clear. The PBU will recognize this combination as indicative of an IVT vector fetch from RAM. The PBU will then enter pause mode, disabling all NVM prefetching for 4 clock cycles or until a PFC occurs.

- Exception Handler in NVM:
  If the corresponding exception handler is in NVM, the CPU will issue an NVM PFC during exception processing and commence execution from the exception handler. The PBU will see this PFC event, exit pause mode, and then commence prefetching the exception handler instruction stream.
- Exception Handler in RAM:
  If the corresponding exception handler is in RAM, the CPU will issue a RAM PFC during exception processing to commence execution from the exception handler. The PBU will not see this PFC event, so it will exit pause mode only after the 4-cycle delay. It will then recommence prefetching from the most recent active NVM instruction stream.

3.5. Performance Monitor Unit (PMU)

The performance monitor provides a method to analyze code efficiency and allows software routines that incur processor stalls to be identified and optimized. In the dsPIC33AK256MPS306 family of devices, the architecture does not have a fixed relationship between the CPU clock speed in MHz and the throughput of the CPU in MIPS (Million Instructions per Second). The throughput of the CPU is dependent on extra cycles incurred from the following:

- CPU pipeline data dependency
- Branches or program flow changes
- Cache misses
- Slow memory or SFR accesses
- Arbitration between bus masters
- A bus that is slower than the CPU

The performance monitor counts the events that cause extra cycles to be inserted into the program flow and the number of elapsed clock cycles. Using this information, the cycles-per-instruction (CPI) can be calculated and the reasons for poor code efficiency can be determined. The CPI value is the number of elapsed clock cycles divided by the number of opcodes that were executed. The stall cycle types listed above will increase the CPI.

The performance monitor uses a set of event signals from the CPU to determine stalls. The module features eight 64-bit counters that can be independently configured to count the occurrence of events from Table 3-11.

3.5.1. Device-Specific Information

Table 3-10. Performance Monitor Summary

Number of counters	Peripheral Bus Speed	Clock Source
8	Standard	Standard Speed Peripheral Clock

Table 3-11. Counter Event Source Selection

SELECT n [4:0]	Event source	Note
18	Fetch stage PBU miss	This event indicates that the requested program data could not be sourced from either the cache memory or the ISB. Therefore, a new fetch from program memory with additional execution cycles was required to obtain the data.
17	Fetch stage PBU hit	This event indicates that the requested program data was sourced from either the cache memory or the ISB. Therefore, no additional execution cycles were required to fetch the instruction.
16	Fetch stage cache busy	Indicates a cycle when the cache was busy transferring data from the instruction stream buffer (ISB) to the cache memory.
15	Fetch stage program memory vector fetch	Indicates that the CPU fetches an interrupt vector and is aligned with a Program Flow Change event. This event can be used to count interrupt events.
14	Fetch stage program memory program flow change	Indicates that a change in program flow has occurred. This could be due to a CALL, RETURN, RETFIE, conditional or unconditional branch, or interrupt event.
13	Fetch stage read stall	Indicates an extra cycle is needed to fetch a program word from memory. This could be caused by a cache miss or an arbitration conflict when fetching program words and data from the same memory.
12	Fetch stage interrupt latency count enable	Indicates the number of cycles due to interrupt latency.
11	Address stage stall	Indicates that CPU pipeline was stalled in the Address stage for any reason, possibly because the instruction is being discarded.
10	Address stage read stall	Indicates that an instruction could not continue because of an extra latency reading a RAM or SFR location.
9	Address stage FPU read stall	Indicates that CPU execution is presently stalled because the CPU cannot read from a FPU register. This occurs because the FPU is currently busy updating the register data.

Table 3-11. Counter Event Source Selection (continued)

SELECT n [4:0]	Event source	Note
8	Address stage FPU instruction stall	Indicates that execution in the FPU coprocessor is currently stalled due to a register data dependency.
7	Address stage hazard	Indicates an extra execution cycle caused by a data dependency upon an earlier instruction in the CPU pipeline, which could not be forwarded.
6	Read stage branch mispredict	Indicates an extra execution cycle caused by mispredicted program flow changes.
5	Read stage conditional branch	Indicates the occurrence of a conditional branch instruction. The count of conditional branch instructions can be compared to the number of branch mispredictions in order to determine the effectiveness of the CPU branch prediction logic.
4	Write stage stall	Indicates that an instruction could not continue because of an extra latency writing to RAM or SFRs.
3	Write stage FPU stall	Indicates that CPU execution is presently stalled because the CPU cannot write to the FPU registers. This occurs because the FPU is currently busy working on the existing register data.
2	CPU instruction completed	Indicates that an instruction in the CPU pipeline has completed.
1	CPU cycle elapsed (reference)	This event count provides the total number of CPU clock cycles elapsed.
0	None	

3.5.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E10	HPCCON	31:24								
		23:16								
		15:8	ON		CLR					
		7:0								
0x1E10	HPCSEL0	31:24				SELECT[3][4:0]				
		23:16				SELECT[2][4:0]				
		15:8				SELECT[1][4:0]				
		7:0				SELECT[0][4:0]				
0x1E14	HPCSEL1	31:24				SELECT[7][4:0]				
		23:16				SELECT[6][4:0]				
		15:8				SELECT[5][4:0]				
		7:0				SELECT[4][4:0]				
0x1E18 ... 0x1E1F	Reserved									
0x1E20	HPCCNTL0	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E24	HPCCNTH0	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E28	HPCCNTL1	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E2C	HPCCNTH1	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E30	HPCCNTL2	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E34	HPCCNTH2	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E38	HPCCNTL3	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E3C	HPCCNTH3	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E40	HPCCNTL4	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E44	HPCCNTH4	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E48	HPCCNTL5	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1E4C	HPCCNTH5	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E50	HPCCNTL6	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E54	HPCCNTH6	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							
0x1E58	HPCCNTL7	31:24	HPCCNT[31:24]							
		23:16	HPCCNT[23:16]							
		15:8	HPCCNT[15:8]							
		7:0	HPCCNT[7:0]							
0x1E5C	HPCCNTH7	31:24	HPCCNT[63:56]							
		23:16	HPCCNT[55:48]							
		15:8	HPCCNT[47:40]							
		7:0	HPCCNT[39:32]							

3.5.2.1. HPCCON Register

Name: HPCCON
Offset: 0x1E10

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		CLR					
Access	R/W		R					
Reset	0		0					

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON On Control bit

Value	Description
1	Module is enabled and counters increment on event signals.
0	Module is disabled and counters do not increment on event signals. Counter values may be read.

Bit 13 – CLR Clear Control bit
A write of a '1' to this location will cause the event counters to clear. This bit may be set at any time whether the PMU is in the Enabled state or the Disabled state. This bit location always reads as '0'.

3.5.2.2. HPCSEL0 Register

Name: HPCSEL0
Offset: 0x1E10

Bit	31	30	29	28	27	26	25	24
				SELECT[3][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				SELECT[2][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				SELECT[1][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				SELECT[0][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 28:24 – SELECT[3][4:0] Counter #3 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-11 for assignments.

Value	Description
11111-00001	Selects the event to be monitored.
00000	No event selected (1’b0); counter is disabled.

Bits 20:16 – SELECT[2][4:0] Counter #2 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-11 for assignments.

Value	Description
11111-00001	Selects the event to be monitored.
00000	No event selected (0b0); counter is disabled.

Bits 12:8 – SELECT[1][4:0] Counter #1 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-11 for assignments.

Value	Description
11111-00001	Selects the event to be monitored.
00000	No event selected (0b0); counter is disabled.

Bits 4:0 – SELECT[0][4:0] Counter #0 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-11 for assignments.

Value	Description
11111-00001	Selects the event to be monitored.
00000	No event selected (0b0); counter is disabled.

3.5.2.3. HPCSEL1 Register

Name: HPCSEL1
Offset: 0x1E14

Bit	31	30	29	28	27	26	25	24
				SELECT[7][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				SELECT[6][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				SELECT[5][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				SELECT[4][4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 28:24 – SELECT[7][4:0] Counter #7 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-11 for assignments.

Value	Description
11111-00001	Selects the event to be monitored.
00000	No event selected (0b0); counter is disabled.

Bits 20:16 – SELECT[6][4:0] Counter #6 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-11 for assignments.

Value	Description
11111-00001	Selects the event to be monitored.
00000	No event selected (0b0); counter is disabled.

Bits 12:8 – SELECT[5][4:0] Counter #5 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-11 for assignments.

Value	Description
11111-00001	Selects the event to be monitored.
00000	No event selected (0b0); counter is disabled.

Bits 4:0 – SELECT[4][4:0] Counter #4 Event Source Selection bits
These control bits determine which event is counted by the associated counter.
See Table 3-11 for assignments.

Value	Description
11111–00001	Selects the event to be monitored.
00000	No event selected (0b0); counter is disabled.

3.5.2.4. HPCCNTLx Register

Name: HPCCNTLx
Offset: 0x1E20, 0x1E28, 0x1E30, 0x1E38, 0x1E40, 0x1E48, 0x1E50, 0x1E58

Bit	31	30	29	28	27	26	25	24
	HPCCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HPCCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HPCCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HPCCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HPCCNT[31:0] Event Counter bits

3.5.2.5. HPCCNTHx Register

Name: HPCCNTHx
Offset: 0x1E24, 0x1E2C, 0x1E34, 0x1E3C, 0x1E44, 0x1E4C, 0x1E54, 0x1E5C

Bit	31	30	29	28	27	26	25	24
	HPCCNT[63:56]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HPCCNT[55:48]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HPCCNT[47:40]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HPCCNT[39:32]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HPCCNT[63:32] Event Counter bits

3.5.3. Operation

The performance monitor operates on the basis of comparing the counter values to the number of CPU cycles. To capture the number of CPU cycles as a reference, one of the available counters is used.

For example, counter 0 can be used for the reference count, and the remaining counters can be used to monitor the available events.

3.5.3.1. Event Selection

Each counter has an associated control to select one of the event sources. The SELECTn[4:0] bits in HPSEL0 and HPSEL1 select one of the signals that are listed in Table 3-11. The CPU cycle elapsed event is the reference and is incremented on each CPU cycle. The CPU instruction completed event indicates that the CPU pipeline has completed. Comparing instructions completed to cycles elapsed yields the CIP value. The ideal value is one instruction completed per one cycle elapsed. The remaining stall, branch or hazard events can be used to determine where stalls occur and what part of the code to optimize.

3.5.3.2. Counters

Each 64-bit counter is split across a pair of 32-bit registers, HPCCNTLx and HPCCNTHx. The registers are read-only and do not have provisions for saturation or rollover events. It is up to user software to halt the module before saturation occurs. The counters can be reset with the CLR bit (HPCCON[13]). The counters are started and stopped using the ON bit (HPCCON[15]). The count values should only be read when ON = '0'.

3.5.3.3. Debugging

Provisions have been made to support the performance monitor in Debug mode. By default, the module is halted in Debug mode to avoid counting cycles associated with the debug executive.

3.5.3.4. Operation in Power-Saving modes

The Performance Monitor module does not operate in Sleep or Idle mode.

3.6. Floating-Point Unit (FPU) Coprocessor

The dsPIC33A FPU Coprocessor includes hardware for the most common floating-point operations for both Single Precision (32-bit) and Double Precision (64-bit) data formats. It is intended to significantly accelerate C compiler floating-point operations when compared to executing software library equivalents and is designed to be compliant with the IEEE 754-2008/2019 floating-point standards. It also includes additional non-IEEE compliant features which may be enabled to handle subnormal values and improve performance.

3.6.1. Features

- Comprehensive IEEE 754-2008/2019 Compliant Instruction Set
  - Supports both Single and Double Precision operations for most instructions
  - Supports all required rounding modes
- Closely Coupled to dsPIC33A CPU Core
  - Instructions issued from CPU core as part of an application instruction stream
  - Independent instruction pipeline and hazard management
- 32 x 32-Bit Data Registers (F-Regs)
  - May be used to hold 32-bit Single Precision or 64-bit Double Precision values
  - Base plus seven partial FPU register contexts
- Optional Subnormal Handling for Improved Performance
  - Subnormal result “Flush-To-Zero” (FTZ) mode
  - Subnormal operand “Subnormals-Are-Zero” (SAZ) mode
- Comprehensive Exception Implementation and Reporting Structure
  - IEEE 754-2019 compliant exception implementation
  - Additional exceptions supported for huge integer results and subnormal operands
- Debug Features Supported:
  - Exception address capture register (FEAR)
  - Exception break signaling
  - NaN propagation

3.6.2. Architectural Overview

The FPU relies on the dsPIC33A CPU for all instruction fetches, most decoding and for all operand movement to and from the system memory. The FPU contains no local memory other than its own register set. Being coupled to the CPU, data size nomenclature is common to both CPU and FPU wherein a word is 16 bits wide, a long word is 32 bits wide and a double word is 64 bits wide.

FPU instructions are part of the CPU instruction set architecture and are executed as part of the CPU code image. FPU instructions are, therefore, executed as a part of the normal execution flow. There are no restrictions with regards to when FPU instructions may appear within the instruction flow.

The CPU can issue, and the FPU can accept, no more than one instruction per clock cycle. However, once issued, the CPU and FPU use independent pipelines to execute the instruction. Consequently, there can be multiple instructions in the process of being executed in both pipelines at any one time. The FPU pipeline will stall the CPU when it is unable to accept any more instructions. The FPU pipeline is also sensitive to speculative instruction control from the CPU (i.e., such that not all issued FPU instructions will be committed). This allows FPU instructions to be located within speculative execution slots that follow conditional branches.

After a successful issue of an FPU instruction, the CPU continues as if executing a single cycle FNOP instruction, and the FPU instruction execution continues within the FPU. Therefore, as some FPU instructions require several cycles to complete, subsequent CPU (and/or FPU) instructions can be fetched, issued and executed (dependencies aside) while the FPU operation progresses. Only when the CPU encounters a hazard with the FPU will it be stalled until the hazard is resolved.

Data and structural hazards are detected and mitigated in both the CPU and FPU and can result in operational stalls which will extend the execution time and increase the effective cycles per instruction of both CPU and FPU instructions.

Note: Refer to the *"dsPIC33A Programmer's Reference Manual"* (DS70005540) for the syntax of all FAND, FIOR, FMUL, etc. instructions.

3.6.2.1. Instruction Pipeline Overview

The pipeline stages consist of Read (RD), Execute (X[n]) and Write-Back (WB), differentiated from the equivalent CPU pipeline stages through the use of different nomenclature. The RD stage is a single-cycle operation (unless stalled). The WB stage is always a single-cycle operation. However, the Execute stage will consist of as many cycles as deemed necessary for the selected instruction functional block. Most basic functions are single-cycle execute operations, though more complex functions (e.g., divide) can be many cycles.

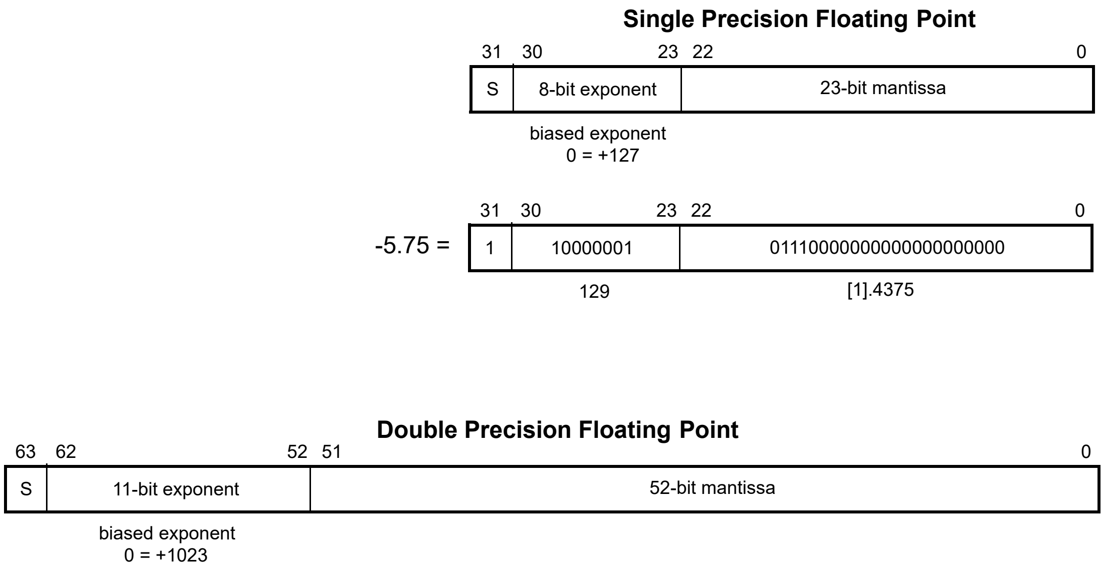
Each instruction that is issued to the FPU must be completed (or killed if speculative) in the order issued. That is, Out of Order (OoO) execution is not supported. However, as the execution time of the FPU instructions can vary considerably, in-order execution requires logic to tag each instruction as it is committed for execution, then track its progress as it flows through the instruction pipeline. Subsequent instructions will therefore be stalled until such time that earlier ones have progressed to allow for sequential, in-order execution.

3.6.2.2. Introduction to Floating Point

The IEEE standard for Floating-Point Arithmetic (IEEE 754-2008) specifies the floating-point data formats which are comprised of a Sign bit, an exponent value and a (fractional) mantissa value. The dsPIC33A Floating-Point Unit (FPU) supports both Single Precision (32-bit, SP) and Double Precision (64-bit, DP) operations for most (though not all) instructions. To avoid the need for another Sign bit in the exponent, the IEEE floating-point format exponent is biased by 127 (SP) or 1023 (DP). Consequently, for any datum, the required IEEE exponent value = datum exponent + bias.

In addition, the '1' to the left of the most significant bit of the mantissa is implied for all numbers except subnormal numbers and is consequently referred to as the leading bit convention's "hidden bit." The mantissa is, therefore, a fractional value with an implied integer value of [1].

Figure 3-27. IEEE Floating-Point Data Formats and Single Precision Example



An IEEE floating-point number can therefore be represented as:

$(-1)^{S} \times [1].(m_{(base2)}) \times 2^{(e-bias)}$

where:

- '*S*' indicates the sign of the number (same values as a signed integer value).
- 'e' represents the exponent value.
- 'm' represents the fractional mantissa value.
- 'bias' is 127 (SP) or 1023 (DP).

For example, -5.75 = $-(1.4375 \times 2^{2})$. In IEEE SP format this would be represented as:

- (-1)1 x [1].4375 x 2(129-127)

or (as shown in Figure 3-27):

- S = 1, exponent = 129_{10}, mantissa = [1].4375

or:

- 0xC0B8 0000

3.6.2.2.1. IEEE 754-2008 Compliance

This module is compliant with the IEEE 754-2008 Standard for Floating-Point Arithmetic for data formats, supported signaling and quiet branch predicates, exception status flags and exception status behavior.

Note that the IEEE 754-2008 minNum(x,y) and maxNum(x,y) definitions are supported only through the largely compatible IEEE 754-2019 minimumNum(x,y) and maximumNum(x,y) operations via the FMINNUM and FMAXNUM instructions. The functional differences related to how +0 and -0 are considered.

- IEEE 754-2008 minNum(x,y) / maxNum(x,y): Operand values +0 and -0 are regarded as equivalent. The result could therefore be either +0 or -0.
- IEEE 754-2019 minimumNum(x,y) / maximumNum(x,y): Operand values +0 and -0 are not regarded as equivalent such that -0 compares to less than +0. The result will therefore be the correct sign of 0 based on the selected operation.

Features Beyond IEEE 754 Requirements

Exception Address Capture Register

The Floating-Point Exception Origination Address Capture register (FEAR) captures the address of the instruction that generates a floating-point exception, provided the associated exception mask bit is clear. If the exception is masked, nothing is captured.

This register is intended for use during system debug, though the FEAR register is read/write in both Mission and Debug modes.

Huge Integer Exception

This exception is signaled whenever a Float-to-Integer Conversion operation (FF2DI or FF2LI) results in an integer value that is larger than the destination register can represent. It is not defined within any IEEE 754 specification apart from a reference to setting the Invalid exception should an integer value exceed the destination size unless this "cannot otherwise be indicated."

3.6.2.2.2. IEEE 754-2019 Compliance

Minimum and Maximum Functions

The FPU module supports all minimum and maximum operations defined in the IEEE 754-2019 standard. The IEEE 754-2008 minNum(x,y) and maxNum(x,y) operations are not directly supported.

- FMINNUM, FMAXNUM: IEEE 754-2019 minimumNumber(x,y)/maximumNumber(x,y) functions. When one of the input operands is a NaN and the other input is a floating-point number (that is not a NaN), the instructions will return the floating-point number. If both input operands are a NaN, the instructions will return a qNaN.
- FMIN, FMAX: IEEE 754-2019 minimum(x,y)/maximum(x,y) functions. When one (or both) of the input operands is a NaN, the instructions will return a qNaN.

Refer to the truth table shown in Table 3-12 for a definition of how NaN operands are handled.

For all minimum and maximum operations, any finite operand value will compare as less than +infinity or greater than -infinity. An operand value of -0 compares to less than +0.

Table 3-12. FMINNUM/FMAXNUM/FMIN/FMAX Operation

Op	Source Operands		Invalid Exception Mask	Result Fd	FSR.INVAL	Invalid Exception Taken?
	Fb	Fs				
FMINNUM FMAXNUM FMIN FMAX	FPN1	FPN2	Don't care	FPN1 or FPN2[1,2,3,4]	0	No
	qNaN1	qNaN2	Don't care	qNaN1 or qNaN2 [5]	0	No
	sNaN	qNaN	1	qNaN (Fs) [6]	1	No
			0			Yes
	qNaN	sNaN	1	qNaN (Fb) [6]	1	No
			0			Yes
	sNaN1	sNaN2	1	Quieted sNaN1 or sNaN2 [5]	1	No
			0			Yes
FMINNUM FMAXNUM	FPN1	qNaN	Don't care	FPN1	0	No
	qNaN	FPN2	Don't care	FPN2	0	No
	FPN1	sNaN	1	FPN1	1	No
			0			Yes
	sNaN	FPN2	1	FPN2	1	No
			0			Yes

Table 3-12. FMINNUM/FMAXNUM/FMIN/FMAX Operation (continued)

Op	Source Operands		Invalid Exception Mask	Result Fd	FSR.INVAL	Invalid Exception Taken?
	Fb	Fs				
FMIN FMAX	FPN1	qNaN	Don't care	qNaN (Fs)	0	No
	qNaN	FPN2	Don't care	qNaN (Fb)	0	No
	FPN1	sNaN	1	Quieted sNaN (Fs)	1	No
			0			Yes
	sNaN	FPN2	1	Quieted sNaN (Fb)	1	No
			0			Yes

Notes:
1. FPN1 and FPN2 are floating-point numbers that are not a NaN (i.e., normal, zero, infinity or sub-normal).
2. Result determined by a FMINNUM/FMIN or FMAXNUM/FMAX operation.
3. Operand value of -0 compares to less than +0.
4. If Fb = Fs (and of the same sign, including infinities), result (Fd) will be loaded with Fb.
5. NaN with largest significand will be passed to result (Fd), quieted if an sNaN.
6. qNaN values have priority over sNaN values (see Table 3-14).

Clamping (Limit) Functions

Although not specified in any IEEE 754 standard, the ISA supports a clamping (or limit) instruction (FFLIM) intended for use where an input operand needs to be constrained between an upper and lower limit. It serves a similar purpose to the integer equivalent FLIM instruction and is essentially a concurrent execution of FMIN and FMAX operations with a common operand. Refer to the truth table shown in Table 3-13 for a definition of how NaN operands are handled.

Any finite operand value will compare as less than +infinity or greater than -infinity. An operand value of -0 compares to less than +0.

For FFLIM operations, when both upper and lower limits are either both qNaN or both sNaN values, a NaN significant comparison is not required, and the Fb NaN will be the default source for the result. This differs from how coincident NaN values are treated in general.

Furthermore, a NaN input value (Fd) will cause the limit values to be ignored and will become the source for the result. That is, checks between input NaNs and limit values (NaNs or otherwise) are also not required.

Table 3-13. FFLIM Operation[4]

Fb[3] (Lower Limit)	Fs[3] (Upper Limit)	Fd (Input Value)	Invalid Exception Mask	Fd (Result)	FSR.INVAL	Exception Taken?
FPNL[1]	FPNU	FPN	Don't care	FPNL or FPNU or FPN[2] or Distinguished qNaN[3]	0	No
FPNL	qNaN_U	FPN	Don't care	qNaN_U	0	No
FPNL	sNaN_U	FPN	1	Quieted sNaN_U	1	No
			0			Yes
qNaN_L	FPNU	FPN	Don't care	qNaN_L	0	No
sNaN_L	FPNU	FPN	1	Quieted sNaN_L	1	No
			0			Yes
sNaN_L	sNaN_U	FPN	1	Quieted sNaN_L[5]	1	No
			0			Yes
sNaN_L	qNaN_U	FPN	1	qNaN_U[6]	1	No
			0			Yes
qNaN_L	sNaN_U	FPN	1	qNaN_L[6]	1	No
			0			Yes
qNaN_L	qNaN_U	FPN	Don't care	qNaN_L[5]	0	No
Don't care	Don't care	sNaN	1	Quieted sNaN[7]	1	No
			0			Yes
Don't care	Don't care	qNaN	Don't care	qNaN	0	No

Notes:
1. FPNL and FPNU are floating-point numbers that are not a NaN.
2. Result determined by FFLIM operation.
3. If Fs is less than Fb (and neither Fs nor Fb are NaN values), the result will be the distinguished qNaN, and the invalid exception will be signaled.
4. FFLIM operation based on IEEE 754-2019 minimum(x,y) and maximum(x,y) operation definitions.
5. Unlike FMIN/FMAX operations, no magnitude comparison of limit NaN values is required. The default result will always be sourced from Fb.
6. qNaN values have priority over sNaN values (see Table 3-14).
7. Unlike FMIN/FMAX operations, no comparison of limit and input (Fd) values are required. The default result will always be sourced from Fd.

NaN Propagation

The FPU supports NaN (payload) propagation to facilitate code debugging. After the CPU issues an instruction to the FPU, the source operands are examined and a NaN value detected, compared and then propagated. Two operand instructions propagate NaN values as shown in Table 3-14.

The FMAC instruction is a special case with respect to NaN propagation as it is essentially three operands consisting of the two source operands (for the multiply) and a prior FMAC result value (i.e., the intermediate used for the accumulate function). The source operands are examined as usual but in conjunction with the selected intermediate result, and any NaN values detected are propagated as defined by Table 3-14.

FFLIM is also a three-operand instruction, though it is ultimately either a two-operand maximum or minimum operation based on the value of the source operand. NaN values detected are propagated as defined by Table 3-13.

Table 3-14. NaN Propagation Priority

Source Operands		Result	Condition	Notes
Fb	Fs			
FPN	sNaN	Quieted sNaN	-	INVAL signaled
FPN	qNaN	qNaN	-	-
sNaN	FPN	Quieted sNaN	-	INVAL signaled
qNaN	FPN	qNaN	-	-
qNaN1	qNaN2	qNaN1	qNaN1 ≥qNaN2	-
		qNaN2	qNaN2 > qNaN1	-
sNaN	qNaN	qNaN	-	INVAL signaled
qNaN	sNaN	qNaN	-	INVAL signaled
sNaN1	sNaN2	Quieted sNaN1	sNaN1 ≥ sNaN2	INVAL signaled
		Quieted sNaN2	sNaN2 > sNaN1	

NaN Propagation Rules

For instructions that generate a result, special propagation rules apply when one or both source operands are NaN values, such that sNaNs can be successfully used as “tracer” values.

When both source operands are NaNs, qNaNs take priority over sNaNs. The appropriate NaN values will be selected as the operation default result as shown in Table 3-14. In the absence of any NaN source operands, any other floating-point numbers will be processed by the FPU module to generate the result.

Note: Source sNaN values will always generate an Invalid exception, but the corresponding quieted sNaN may not always be the operation result.

This magnitude comparison is based on the magnitude of the significand associated with each of these values (the sign is ignored). It is straightforward to implement because:

- The MSb of a sNaN significant is 0 (with any non-zero value in the remaining bits).
- The MSb of a qNaN significant is 1 (with any value in the remaining bits).

An example tracer sNaN propagation is shown in Figure 3-28. When an FPU operation (Op1) executes with a sNaN and a normal floating-point number, the sNaN will be quieted and propagate as the result. In Figure 3-28, this is sNaN1 (the initial tracer) being propagated as qNaN1. Should a subsequent operation (Op2) execute with qNaN1 and, for example, a later sNaN tracer (sNaN2), operand qNaN1 will have priority, thereby maintaining propagation of the original tracer payload. However, should that qNaN1 value then be presented to another FPU operation (Op3) together with another qNaN, the qNaN result could be either of the source qNaNs, depending upon the magnitude of their respective significands.

However, if the significand of the initial sNaN1 tracer is large enough, it will ultimately be able to continue to propagate past all subsequent NaNs and be available to view at the end of the code block, thereby allowing it to be traced back to its source.

Figure 3-28. Tracker sNaN Operand Propagation Example

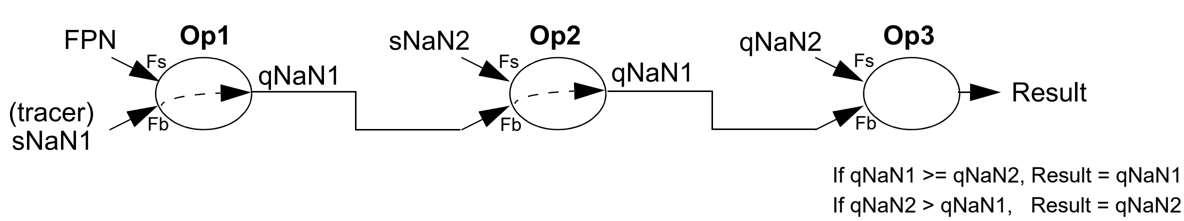


Table 3-15. FMAC NaN Propagation Priority

Multiply Source Operands		Add Source Operands		FMAC Result (Fd)	Notes
Fb or Fs	Fs or Fb	Intermediate Result	Accumulator Source (Fd)		
FP Multiply					
		FP Add			
FPN(1)	FPN	FPN	FPN	FPN	
			qNaN	qNaN	
			sNaN	Quieted sNaN	INVAL signaled
		Distinguished qNaN(3)	FPN	Distinguished qNaN	INVAL signaled
			qNaN1	Distinguished qNaN or qNaN1(2)	INVAL signaled
			sNaN	Distinguished qNaN or Quieted sNaN(2)	INVAL signaled
FPN(1)	sNaN1	Quieted sNaN1	FPN	Quieted sNaN1	INVAL signaled
			qNaN	Quieted sNaN1 or qNaN(2)	INVAL signaled
			sNaN2	Quieted sNaN1(2)	INVAL signaled
FPN(3)	qNaN1	qNaN1	FPN	qNaN1	
			qNaN2	qNaN1 or qNaN2(2)	
			sNaN	qNaN1	INVAL signaled
qNaN1	qNaN2	qNaN1 or qNaN2(2)	FPN	qNaN1 or qNaN2(2)	
			qNaN3	qNaN1 or qNaN2 or qNaN3(2)	
			sNaN	qNaN1 or qNaN2(2)	INVAL signaled
sNaN1	sNaN2	Quieted (sNaN1 or sNaN2)(2)	FPN	Quieted (sNaN1 or sNaN2)(2)	INVAL signaled
			qNaN	Quieted (sNaN1 or sNaN2)(2) or qNaN	INVAL signaled
			sNaN3	Quieted (sNaN1 or sNaN2)(2)	INVAL signaled

Notes:
1. FPN is a floating-point number that is not a NaN.
2. Using significand magnitude comparisons as defined in Table 3-14.
3. A distinguished qNaN intermediate result will arise when operands are 0 and Inf (any sign).

3.6.3. Zero, Infinity, Not a Number (NaN) and Subnormal Values

The IEEE 754-2008/2019 standards reserve data encoding to represent special values, as shown in Figure 3-29.

Zero is conveyed when both the exponent and mantissa are all 0's. Zero is a signed value (for some operations) as determined by the Sign bit. Infinity is conveyed by an exponent value of all 1's with an all 0's mantissa. Infinity is a signed value as determined by the Sign bit.

A Signaling NaN is conveyed by an exponent value of all 1's with the MSb of the mantissa set to 0 (remaining mantissa bits may be set to any value). The Quiet Nan (qNaN, see Not a Number (NaN)) is conveyed by an exponent value of all 1's with the MSb of the mantissa set to 1 (remaining mantissa bits may be set to any value). NaN values are not signed, so the Sign bit may be any state.

A Subnormal value (see Subnormal Number) is conveyed by an exponent of all 0's and any non-zero mantissa value. Subnormals are signed values as determined by the Sign bit.

3.6.3.1. Not a Number (NaN)

The Signaling NaN (sNaN) and Quiet NaN (qNaN) are specific data codes that indicate certain situations. In all cases, an exponent value of all 1's with a non-zero mantissa signifies a NaN (an exponent value of all 1's with an all 0's mantissa is used to convey Infinity).

qNaNs may be generated as the result of an invalid operation, such as taking the square root of a negative floating-point number. A qNaN will propagate through subsequent floating-point operations. Operations that will generate an Invalid exception for each instruction are documented in Table 3-19.

sNaNs are reserved input operands which, under default exception handling, will signal an Invalid exception when encountered. This may be used to indicate uninitialized variables or as debug aids, but they are never generated by arithmetic computations or comparisons. Whenever the source operand of operation is an sNaN, the result will be a qNaN.

Both sNaNs and qNaNs can store "payloads" in the mantissa bit field. The payload must not affect the MSB of the mantissa. The payload can be used as a debugging aid in tracing through complex arithmetic calculations.

3.6.3.1.1. qNaN and sNaN Propagation

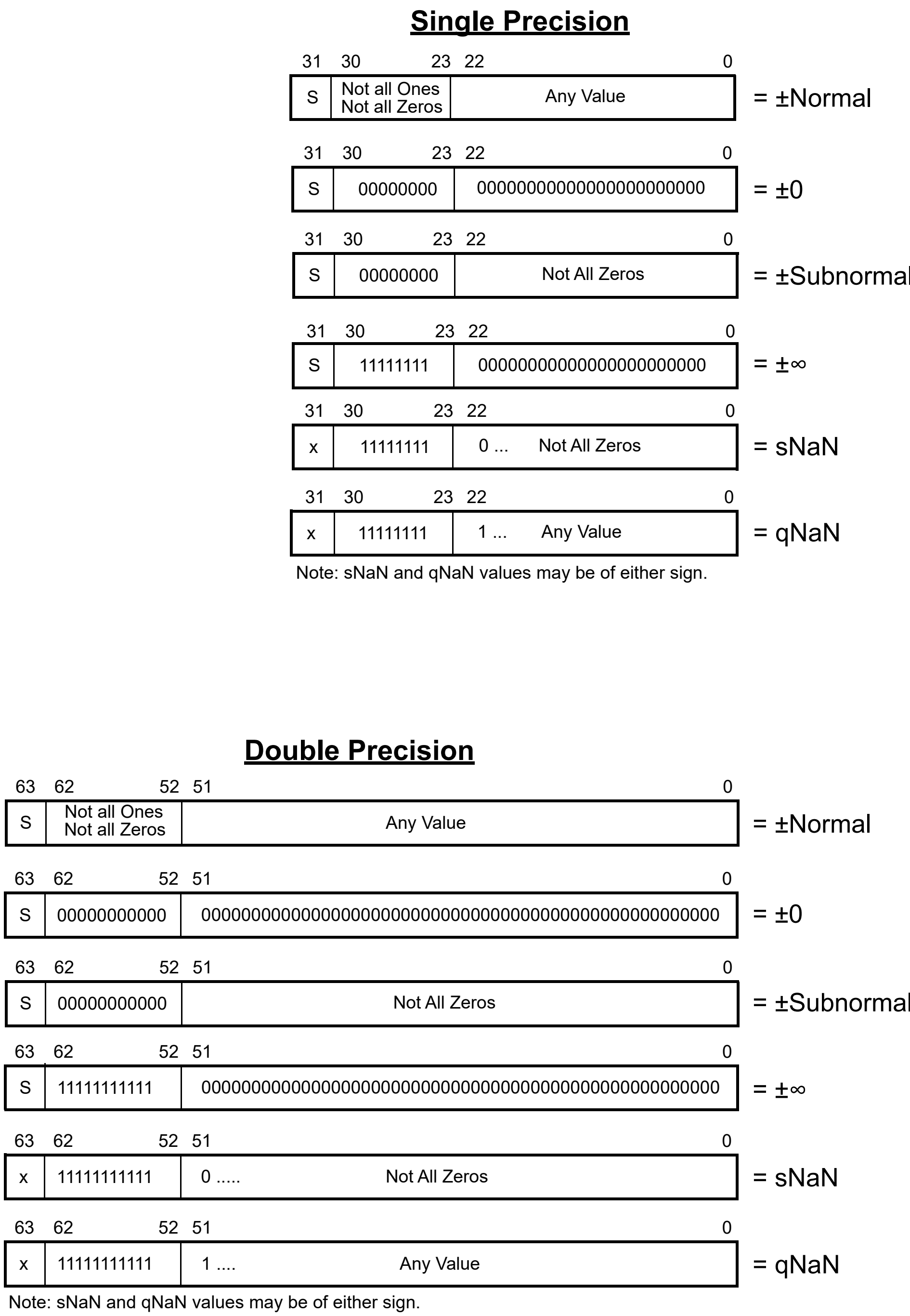
The IEEE 754-2008/2019 standards indicate that source qNaNs should be propagated, including any associated payload. The FPU module does not propagate any source qNaNs, but instead generates fixed distinguished qNaN results.

In keeping with other device floating-point implementations, this module will propagate qNaN and sNaN values where possible. Refer to NaN Propagation for further detail.

For instructions where a source operand qNaN is not available, a distinguished qNaN value will be provided as the result whenever those instructions suffer a computational error:

- Single Precision: Distinguished qNaN = 0x7FC0_0001
- Double Precision: Distinguished qNaN = 0x7FF8_0000_0000_0001

Figure 3-29. Floating-Point Encodings



3.6.3.1.2. NaN Operands with Float-to-Integer Conversion

The FF2DI and FF2LI are the float-to-integer instructions. These instructions can output a huge integer in lieu of invalid integers when the source is a value that would convert to an integer outside the range of the result format under the applicable rounding attribute. This output is implemented as a new exception, Huge Integer (FSR.HUGI).

The IEEE 754-2008/2019 standard calls for Invalid to be signaled if this situation cannot otherwise be indicated. The FSR.HUGI exception is considered an implementation of "otherwise indicated," making the FF2DI and FF2LI instructions compliant.

The module drives status output Invalid (and does not drive Huge Integer) when the source is ±NaN or ±∞ per the IEEE 754-2008/2019 standards. Note that Invalid is also driven for a qNaN input.

3.6.3.2. Subnormal Number

A subnormal number (denormal number) is a *non-zero* floating-point number with a magnitude of less than that of the smallest normal number representable in the given format. The benefit of subnormal numbers is that they allow for gradual underflow when a result is very small (when compared to that without subnormal numbers). The IEEE 754 standard represents subnormal numbers as a special case.

Using Single Precision data format as an example, the smallest normal numbers around 0 are greater than $+2^{-126}$ or less than -2^{-126}, which occur when the floating-point number exponent is 1 (bearing in mind that the 8-bit exponent is defined with a bias of +127) and the mantissa is all 0's.

The exponent value of 0 is reserved for subnormal numbers. However, the IEEE 754 standard treats subnormal numbers as a special case where the hidden mantissa bit becomes 0 and the exponent bias is changed (by +1) to compensate, such that the datum exponent becomes -126. This allows the subnormal range to surround 0 and be between a little greater than -2^{-126} to a little less than -2^{-126}. That is:

$-2^{-126} < subnormal < +2^{-126}$

The minimum exponent value is referred to as Emin, and is -126 for Single Precision and -1023 for Double Precision formats. A subnormal number would therefore be represented as:

$(-1)^S \times [0].m_{base2} \times 2^{Emin}$

where:

- '*S*' indicates the sign of the number (same values as a signed integer value).
- '*m*' represents the fractional mantissa value.

For example, the largest SP positive subnormal number will be when all mantissa bits are all set (0x007F_FFFF), and the smallest number will be when all mantissa bits are all clear (0x0000_0000), which is 0.0.

3.6.3.2.1. Subnormal Number Handling

Should any floating-point calculation generate a subnormal result, the FSR.UDF will be set; if it is not already set, the sticky status FSR.UDFS will also be set. In addition, if any instruction is presented with a subnormal operand value, FSR.SUBO will be set. If it is not already set, the sticky status FSR.SUBOS will also be set.

Subnormal Override Functions

Although not IEEE 754 compliant, subnormal operands and/or results may be overridden to improve the performance of some applications that do not require subnormal number precision. Use of the subnormal override function:

- Avoids the consequences of processing or having to deal with subnormal datum.
- Handles result underflows when a result is subnormal, negating the need to handle an underflow exception.

The subnormal override functions consist of two parts, one to flush subnormal input operands to zero (referred to as Subnormals-Are-Zeros or SAZ mode), and the other to remove subnormal results (referred to as Flush-To-Zero or FTZ mode).

Note: Subnormal override modes are not applicable to FCPS/FCPQ (no result to override), `FAND`, `FIOR`, `FTST`, `FMOV`, `FMOVC` or any CPU to/from a FPU data move instruction.

Subnormals-Are-Zero (SAZ)

Subnormals-Are-Zero (SAZ) mode is enabled when FCR.SAZ is set and will ensure that any subnormal operand input to a functional block is replaced with a 0 value of the same sign as the subnormal value it is replacing. This avoids the consequences of processing or having to deal with subnormal datum. This operation applies to all floating-point instructions except: `FMOV`, `FMOVC`, `FAND`, `FIOR`, `FTST`, `FLI2F` and `FDI2F`.

Note: SAZ mode is applied to `FABS` and `FNEG` instructions to ensure result consistency with that of an equivalent sequence of FPU arithmetic instructions.

Note: Does not apply to FPU to CPU or CPU to FPU move instructions.

3.6.3.2.2. Flush-To-Zero (FTZ)

Flush-To-Zero (FTZ) mode is enabled when both FCR.FTZ and FCR.UDFM are set. If the underflow exception is unmasked (FCR.UDFM = `0`), then the FCR.FTZ bit will have no effect. Should a floating-point operation generate an infinitely precise result that is less than the smallest possible subnormal number, then the functional block will round this to a result of 0 with the same sign as the subnormal value. This will occur irrespective of whether FTZ mode is enabled or not. Both Underflow (FSR.UDF) and Inexact (FSR.INX) will be signaled (if not already set, sticky status FSR.UDFS and FSR.INXS will also be set). Should a floating-point result be a subnormal number (that the functional block has not rounded up to the smallest magnitude normal number), and FTZ mode is enabled, the result will be replaced with 0 of the same sign as the subnormal value it is replacing. Again, both Underflow (FSR.UDF) and Inexact (FSR.INX) will be signaled (if not already set, sticky status FSR.UDFS and FSR.INXS will also be set), though the Underflow exception has to be masked (in order to enable FTZ mode), so no interrupt will be issued. Forcing the result to 0 allows the user to ignore underflows (though at the expense of some accuracy).

The FCR.FTZ bit is only examined during the WB stage of an instruction such that it may be modified as late as the cycle before the instruction enters the WB stage. For example, the following code sequence will only apply the FTZ function to the `FSUB` instruction:

```
 * Assume FCR.FTZ=0 && FCR.UDFM=1 at entry
ADD.s F0, F1,F2                  ; add without FTZ
IOR.s #0x0400, FCR               ; set FCR.FTZ
SUB.s F3, F4,F5                  ; sub with FTZ
AND.s #0xFBFF, FCR               ; clear FCR.FTZADD.s
F2,F6,F7                   ;add without FTZ
```

3.6.3.2.3. Subnormal Operand Exception

Should any affected instruction execute using a subnormal operand, and SAZ mode is disabled, the Subnormal Operand (FSR.SUBO) exception will be signaled. This provides a mechanism to indicate the use of a subnormal value without requiring the operand to be tested (FTST).

Should SAZ mode be enabled and a subnormal operand is encountered (and changed to a 0 value), SUBO will not be signaled.

SAZ mode may be enabled irrespective of whether the SUBO exception is masked or not (though when enabled, it will never signal SUBO).

3.6.4. Floating-Point Data Register (F0-F31)

Name: Fn

Bit	31	30	29	28	27	26	25	24
	Fn[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	Fn[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	Fn[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	Fn[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – Fn[31:0] Floating-Point Data Register bits

3.6.5. Floating-Point Control Register

Name: FCR

Note:

1. Floating-Point Exception Mask bits, FCR [6:0]: Each Exception Mask bit corresponds to an Exception Status flag in the FSR. The Mask bit must be clear to allow the exception event to generate an interrupt to the CPU. The Underflow Mask bit (FCR.UDFM) is also used as part of the Flush-to-Zero (FTZ) mode enable as discussed in Flush-To-Zero (FTZ).

   Floating-Point Rounding Mode Control bits, FCR [9:8]: These bits define the global IEEE 754 Compatible Rounding mode used by the FPU instruction. See Rounding Modes.

   Floating-Point Subnormal Override Mode Control bits, FCR [11:10]: These bits enable the Subnormals-Are-Zero (SAZ) and Flush-To-Zero (FTZ) Subnormal Override modes supported by the FPU.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					SAZ	FTZ	RND [1:0]	
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
		SUBOM	HUGIM	INXM	UDFM	OVFM	DIV0M	INVALM
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 11 – SAZ Subnormals-Are-Zero Operand Mode bit

Value	Description
1	Subnormals-Are-Zero mode is enabled.
0	Subnormals-Are-Zero mode is disabled.

Bit 10 – FTZ Flush-To-Zero Result Mode bit

Value	Description
1	Flush-To-Zero mode is enabled.
0	Flush-To-Zero mode is disabled.

Bits 9:8 – RND [1:0] FPU Rounding Mode bits

Value	Description
11	IEEE Round to Negative Infinity (floor)
10	IEEE Round to Positive Infinity (ceiling)
01	IEEE Round to Zero (truncate)
00	IEEE Round to Nearest (even)

Bit 6 – SUBOM Subnormal Operand Exception Mask bit

Value	Description
1	Subnormal exception is masked.
0	Subnormal exception is not masked.

Bit 5 – HUGIM Huge Integer Exception Mask bit

Value	Description
1	Huge Integer exception is masked.
0	Huge Integer exception is not masked.

Bit 4 – INXM Inexact Exception Mask bit

Value	Description
1	Inexact exception is masked.
0	Inexact exception is not masked.

Bit 3 – UDFM Underflow Exception Mask bit

Value	Description
1	Underflow exception is masked.
0	Underflow exception is not masked.

Bit 2 – OVFM Overflow Exception Mask bit

Value	Description
1	Overflow exception is masked.
0	Overflow exception is not masked.

Bit 1 – DIV0M Divide-By-Zero Exception Mask bit

Value	Description
1	Divide-By-Zero exception is masked.
0	Divide-By-Zero exception is not masked.

Bit 0 – INVALM Invalid Exception Mask bit

Value	Description
1	Invalid exception is masked.
0	Invalid exception is not masked.

3.6.6. Floating-Point Status Register

Name: FSR

Note: Dynamic Floating-Point Exception Status bits, FSR [6:0]: Dynamic Status bits are updated based on the results from each instruction functional block and will be updated after execution of each instruction.

Sticky Floating-Point Exception Status bits, FSR [14:8]: Sticky Status bits can be set based on the results from each instruction functional block but cannot be cleared by hardware (other than at device Reset) and, therefore, represent a history of status since the last time the sticky bits were cleared. The FSR bits can be cleared through software.

Floating-Point Compare Status bits, FSR [19:16]: The status generated by executing a floating-point compare (FCPS/FCPQ) instruction. Used individually or combined to generate the floating-point branch conditions used by the CPU CBRAn instructions.

Floating-point Test Status bits, FSR [27:24]: Floating-Point Datum Characteristic status generated by executing the Floating-Point Test (FTST) instruction.

Bit	31	30	29	28	27	26	25	24
				SUB	INF	FN	FZ	FNAN
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
					GT	LT	EQ	UN
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
		SUBOS	HUGIS	INXS	UDFS	OVFS	DIV0S	INVALS
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 28 – SUB (FTST) Subnormal Status bit

Value	Description
1	Operand is subnormal.
0	Operand result is not subnormal.

Bit 27 – INF (FTST) Infinite Status bit

Value	Description
1	Operand is infinite.
0	Operand is not infinite.

Bit 26 – FN (FTST) Negative Status bit

Value	Description
1	Operand is negative.
0	Operand is not negative.

Bit 25 – FZ (FTST) Zero Status bit

Value	Description
1	Operand is zero.
0	Operand is not zero.

Bit 24 – FNAN (FTST) Not a Number Status bit

Value	Description
1	Operand is a NaN (qNaN or sNaN) value.
0	Operand is not a NaN value.

Bit 19 – GT (FCPS/FCPQ) Greater Than Status bit

Value	Description
1	Minuend is greater than the subtrahend (Fb > Fs).
0	Minuend is not greater than the subtrahend (Fb ≤ Fs).

Bit 18 – LT (FCPS/FCPQ) Less Than Status bit

Value	Description
1	Minuend is less than the subtrahend (Fb < Fs).
0	Minuend is not less than the subtrahend (Fb ≥ Fs).

Bit 17 – EQ (FCPS/FCPQ) Equal Status bit

Value	Description
1	Minuend is equal to the subtrahend (Fb = Fs).
0	Minuend is not equal to the subtrahend (Fb != Fs).

Bit 16 – UN (FCPS/FCPQ) Unordered Status bit

Value	Description
1	Either or both operands are NaN values.
0	Neither operands are NaN values.

Bit 14 – SUBOS Sticky Subnormal Operand Exception Flag bit

Value	Description
1	Subnormal Operand exception has just occurred or at some time in the past.
0	Subnormal Operand exception has not occurred.

Bit 13 – HUGIS Sticky Huge Integer Exception Flag bit

Value	Description
1	Huge Integer exception has just occurred or at some time in the past.
0	Huge Integer exception has not occurred.

Bit 12 – INXS Sticky Inexact Exception Flag bit

Value	Description
1	Inexact exception has just occurred or at some time in the past.
0	Inexact exception has not occurred.

Bit 11 – UDFS Sticky Underflow Exception Flag bit

Value	Description
1	Underflow exception has just occurred or at some time in the past.
0	Underflow exception has not occurred.

Bit 10 – OVFS Sticky Overflow Exception Flag bit

Value	Description
1	Overflow exception has just occurred or at some time in the past.
0	Overflow exception has not occurred.

Bit 9 – DIV0S Sticky Divide by Zero Exception Flag bit

Value	Description
1	Divide by Zero exception has just occurred or at some time in the past.
0	Divide by Zero exception has not occurred.

Bit 8 – INVALS Sticky Invalid Exception Flag bit

Value	Description
1	Invalid exception has just occurred or at some time in the past.
0	Invalid exception has not occurred.

Bit 6 – SUBO Subnormal Operand Exception Flag bit

Value	Description
1	Subnormal Operand exception has occurred.
0	Subnormal Operand exception has not occurred.

Bit 5 – HUGI Huge Integer Exception Flag bit

Value	Description
1	Huge Integer exception has occurred.
0	Huge Integer exception has not occurred.

Bit 4 – INX Inexact Exception Flag bit

Value	Description
1	Inexact exception has occurred.
0	Inexact exception has not occurred.

Bit 3 – UDF Underflow Exception Flag bit

Value	Description
1	Underflow exception has occurred.
0	Underflow exception has not occurred.

Bit 2 – OVF Overflow Exception Flag bit

Value	Description
1	Overflow exception has occurred.
0	Overflow exception has not occurred.

Bit 1 – DIV0 Divide by Zero Exception Flag bit

Value	Description
1	Divide by Zero exception has occurred.
0	Divide by Zero exception has not occurred.

Bit 0 – INVAL Invalid Exception Flag bit

Value	Description
1	Invalid exception has occurred.
0	Invalid exception has not occurred.

3.6.7. Floating-Point Exception Address Capture Register

Name: FEAR

Note:
1. FEAR [1] is always set to 0b0.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	FEAR[22:15]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FEAR[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FEAR[6:0]							EACE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

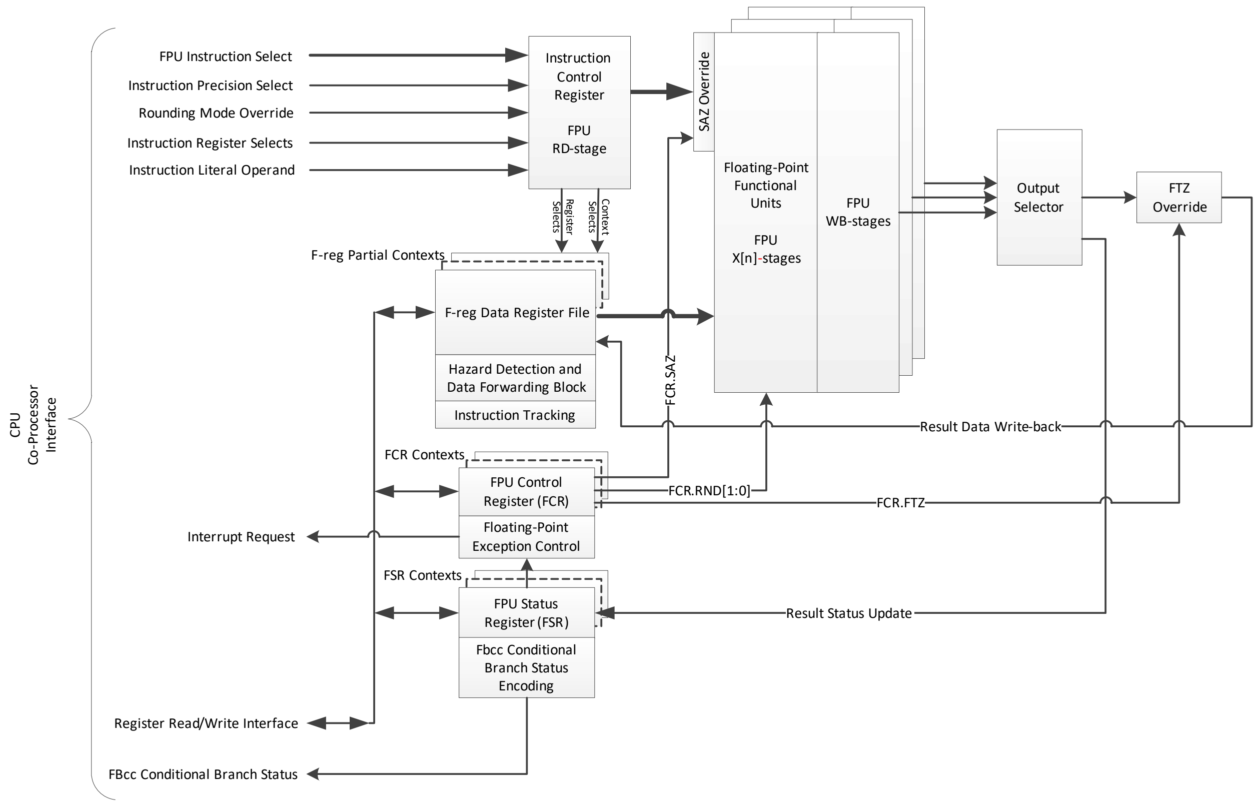
Bits 23:1 – FEAR[22:0] Floating-Point Instruction Exception Address Capture Register bits[1]

Bit 0 – EACE Exception Address Capture Enable bit

Value	Description
1	FEAR register address capture enabled
0	FEAR register address capture disabled (and FEAR [23:0] may contain a captured address)

3.6.8. FPU Module Operation

Figure 3-30. FPU Module Block Diagram



3.6.8.1. Floating-Point Unit Registers

The Floating-Point Unit (FPU) provides a large set of working registers (F-regs).

- 32 x 32-bit (Single Precision, F0 ... F31) or
- 16 x 64-bit (Double Precision, F0, F2 ... F28, F30) or
- A mix of the two sizes aligned as shown in Figure 3-31

In addition to the F-regs, status (FSR) and control (FCR) registers are also supported as shown in Figure 3-31:

- FCR (FPU Control Register, 16-bit)
  - FCR [6:0]: Exception mask control
  - FCR [9:8]: Rounding mode control
  - FCR [10]: Subnormal result “Flush-to-Zero” (FTZ) control
  - FCR [11]: Subnormal operand “Subnormals-are-Zero” (SAZ) control
- FSR (FPU Status Register, 32-bit): Holds the status of retired floating-point instructions
  - FSR [6:0]: Instruction “most-recent” exception status
  - FSR [14:8]: Instruction "sticky" exception status
  - FSR [19:16]: FCPS/FCPQ instruction status
  - FSR [28:24]: FTST instruction status
- FEAR: (FPU Exception Address Capture Register, 24-bit): Holds the address of the first instruction encountered that causes an exception. All subsequent instructions in the FPU pipeline that

subsequently retire will not affect the FEAR, even if they also generate exceptions. The FEAR is intended for use during debugging of the floating-point software.

Note: The FSR upper and lower 16 bits (represented as FSRH and FSRL, respectively) may be read/ written independently of each other by some instructions.

Note: Although inconsistent with device interrupts, where interrupt controls are referred to as enables (where logic 1 represents enabled), it is more conventional (and in keeping with the IEEE-754 specification) that the FPU exception controls be referred to as masks (where logic 1 represents masked). These bits are all set at Reset and masking exceptions by default.

3.6.8.1.1. FPU Register Access

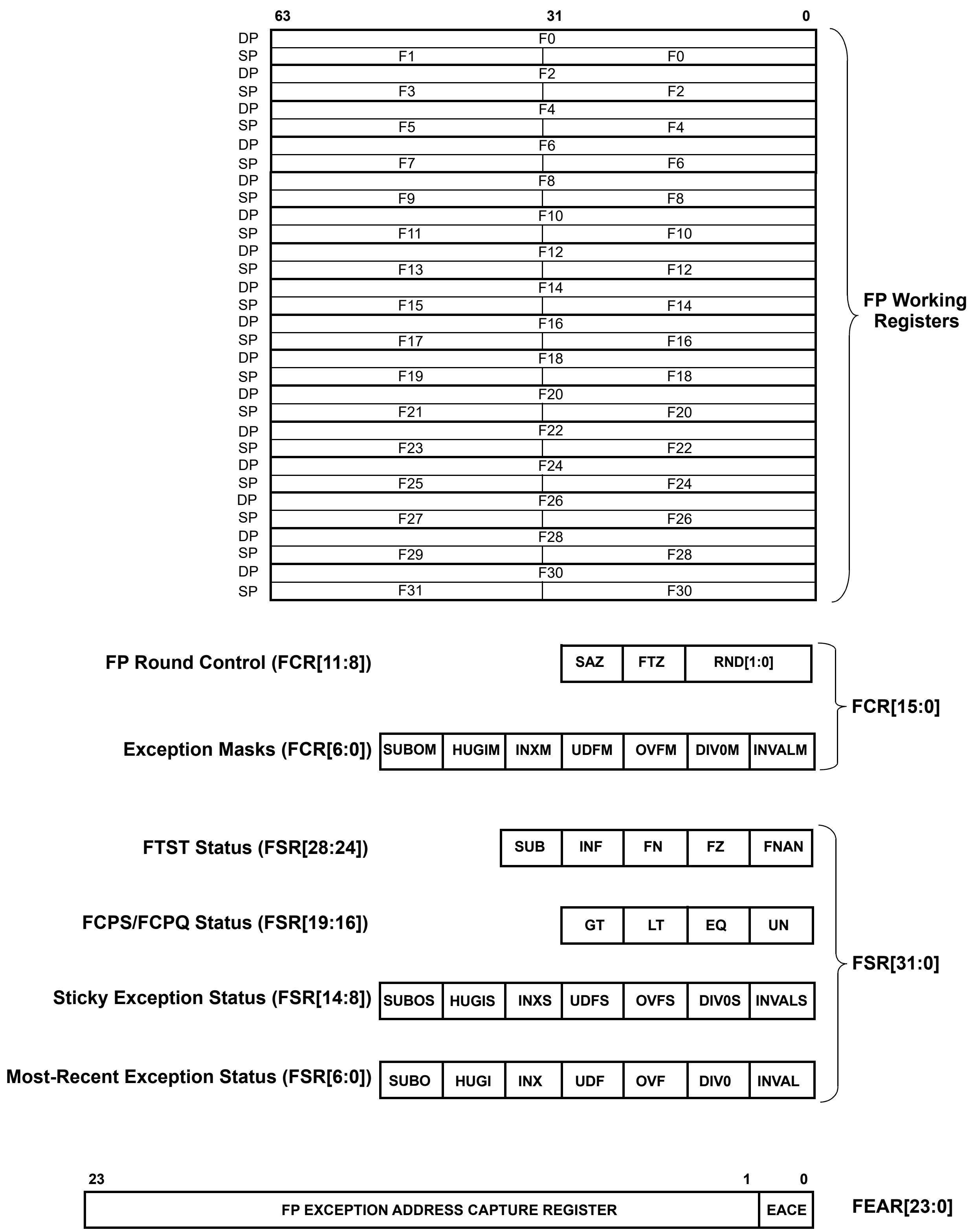
Data may be moved in and out of any FPU register, from OR to W-regs or DS memory, by using dedicated coprocessor register move instructions that execute from within the integer pipeline (refer to `MOVCRW`, `MOVWCR`, `MOVLCR`, `LDWLOCR`, `STWLOCR`, `PUSHCR` and `POPCR` CPU instructions as described in FPU Module Operation).

All data are moved as 32-bit entities, so double precision data moves will require the execution of two instructions (64-bit data moves are not supported in this device).

In addition, the FPU supports `FAND` and `FIOR` instructions that can logically AND or OR a literal value with the FSR (lsw only, exception status), FCR or FEAR (lsw only).

3.6.8.2. FPU Programmer's Model

Figure 3-31. FPU Programmer's Model



3.6.8.3. FPU Register Set

The FPU Programmer's Model of registers is shown in Figure 3-31 and is comprised of floating-point operand registers (F-regs), a floating-point control register (FCR), a floating-point status register (FSR) and a floating-point exception address capture register (FEAR). None of the registers are memory-mapped, and they must be read or written by the CPU using the coprocessor move instructions (`MOVCRW`, `MOVWCR`, `PUSHCR`, `POPCR`, `LDWLOCR`, `STWLOCR` and `MOVLCR`). The FCR, FSR and FEAR registers may also be subjected to a literal AND or OR operation by the `FAND` and `FIOR` instructions, respectively.

3.6.8.3.1. Floating-Point Operand Registers (F-Regs)

To differentiate from the CPU working W-regs, the FPU operand/result data working registers are referred to as F-regs. The FPU supports up to 32 Single Precision values, or up to 16 Double Precision values. Aligned pairs of the F-regs registers (e.g., F1:F0 values may be used to provide data storage for Double Precision values.) Single and Double Precision values may be mixed within the register file. Other than data movement in and out of the FPU, all instructions are register-to-register operations within the FPU register set.

The F-regs are not memory-mapped and can only be accessed by the CPU using specific instructions as discussed in CPU Access of FPU Registers.

The 32 x 32-bit F-reg array, together with additional register contexts, is implemented as a register file. FPU instructions can have 1, 2 or 3 operands (read sources) and 0 or 1 result destinations, and most update the status in the FSR. Registers may be used individually for Single Precision data values or coupled as odd; even pairs (only) should be used to support Double Precision data values (e.g., F1:F0).

Source registers are bound to an instruction when the instruction is issued and are not writable by the CPU until the instruction is committed. At this point, they are clocked into operand registers that drive the target functional block and can, therefore, be subsequently written. Note that a bound source register may be read at any time.

Destination registers (F-regs and FSR) are bound to an instruction when the instruction is committed and are not accessible by the CPU until the instruction has retired.

Floating-Point Control Register (FCR)

The FCR is comprised of the following bit fields as defined in FCR.

Floating-Point Exception Mask bits, FCR [6:0]: Each Exception Mask bit corresponds to an Exception Status flag in the FSR. The Mask bit must be clear to allow the exception event to generate an interrupt to the CPU. The Underflow Mask bit (FCR.UDFM) is also used as part of the Flush-to-Zero (FTZ) mode enable as discussed in Flush-To-Zero (FTZ).

Floating-Point Rounding Mode Control bits, FCR [9:8]: These bits define the Global IEEE 754 Compatible Rounding mode used by the FPU instruction. See Rounding Modes.

Floating-Point Subnormal Override Mode Control bits, FCR [11:10]: These bits enable the Subnormals-Are-Zero (SAZ) and Flush-to-Zero (FTZ) Subnormal Override modes supported by the FPU.

3.6.8.3.2. Floating-Point Unit Register Contexts

To speed up real-time control systems and other time critical applications, the dsPIC33A FPU supports multiple register contexts that are tied to Interrupt Priority Levels.

The FPU includes a set of hardware register contexts. Each context includes the FSR, FCR and four register pairs (i.e., F0 through F7). All other F-regs and FEAR are not included and must be saved and restored through software.

The number of supported register contexts matches that of the CPU and is fixed at seven, which represents one context per CPU Interrupt Priority Level (IPL). Should the CPU change context, then the FPU will follow suit, and all subsequent instructions issued to the FPU will execute within that (new) context. However, all FPU instructions issued in a prior context will be allowed to continue to execute and retire within that context, irrespective of the context change within the CPU. Similarly, any data dependencies that occur within the context of the instruction underway will remain within that context.

As the FSR is part of the register context, exceptions are context specific. Should the FPU change register context, any FPU exceptions generated as a result of the execution of FPU instructions already issued from the prior context will remain pending until the FPU returns to that original context.

Hazard detection is also context-based such that each instruction operand and result register are tagged with their own context. Therefore, hazards can only exist within the same register context.

This concept extends to the FSR and FCR, which have independent representations within each register context. Consequently, the CPU will not stall (assuming no FSR and/or FCR hazard exist within the current context) if it accesses the FSR or FCR while the FPU continues to execute instructions issued from within a different context. These instructions will have access to their own version of the FSR and FCR.

3.6.8.3.3. CPU Access of FPU Registers

The following CPU instructions are provided specifically to support data movement into and out of the coprocessors. The assembler uses the register declarations to direct encoding of the FPU as the target coprocessor within each instruction op code:

- `MOVCRW`: Move any FPU register to a W-reg or DS memory (using indirect addressing).
- `LDWLOCR`: Move the contents of DS memory (read using register+literal offset addressing (`[Ws + Slit14]`)) to any FPU F-reg register.
- `STWLOCR`: Move any FPU F-reg to DS memory (read using register+literal offset addressing (`[Wd+Slit14]`)).
- `PUSHCR`: 16-bit short instruction dedicated to moving any FPU register onto the system stack.
- `MOVWCR`: Move a W-reg or DS memory value (using indirect addressing) to any FPU register.
- `POPCR`: 16-bit short instruction dedicated to moving a value from the system stack to any FPU register.
- `MOVLCR`: Move a 32-bit literal value to any FPU register.

Note: These instructions are referred as `mov.l`, `push.l` or `pop.l`. Please refer to the *“dsPIC33A Programmer’s Reference Manual”* (DS70005540) for the correct syntax of these instructions.

3.6.8.3.4. Intra-FPU Register Moves and Logical Operations

In addition to CPU to/from FPU data movement, the FPU supports instructions that execute within its own pipeline that perform register to register moves or logical operations.

- `FMOV`: Copy any F-reg or F-reg pair into another F-reg or F-reg pair.
- `FMOVC`: Move one of 32 Single or Double Precision constant values into an F-reg or F-reg pair.
- `FAND`: Logically AND a 16-bit literal value (lit16) with the lsw of the FPU FSR, FCR or FEAR.
- `FIOR`: Logically OR a 16-bit literal value (lit16) with the lsw of the FPU FSR, FCR or FEAR.

Note: To allow a subsequent instruction to immediately utilize `FAND` and `FIOR` changes to FCR.RND[1:0] and FCR.SAZ control bits without stalls, these bits are manipulated and updated in the first pipeline stage (RD-stage). However, the remaining FCR bits are not written back until the end of the instruction as usual. Consequently, should the CPU need to read the FCR immediately after modification, it will be stalled by the FPU until the `FAND` or `FIOR` instruction has retired.

3.6.8.4. Data Hazard Management

Read-After-Write (RAW) data hazards can arise due to:

- Data dependencies between FPU instructions

- The result of a register move from an FPU register to the CPU when an FPU instruction underway has not yet completed its result write (to the same register)

Write-After-Read (WAR) data hazards within the FPU pipeline alone are not possible because the pipeline ensures that instruction reads always precede subsequent instruction writes. However, a WAR hazard can arise when the CPU pipeline writes to an FPU register that has yet to be read by a previously issued but stalled FPU instruction.

Write-After-Write (WAW) data hazards are possible should the CPU attempt to write to an FPU register that is also the target of a prior FPU instruction which has not yet completed its result write.

All hazards are detected within the FPU or CPU (or both) and will be mitigated either through data forwarding or pipeline stalls.

3.6.8.5. FPU and CPU Exceptions

Issued FPU instructions that become committed (accepted by the Execute stage) are always atomic with respect to CPU exceptions. No CPU exception (other than a Reset event) can force the FPU to abandon an instruction that is already underway.

CPU exceptions will result in a register context switch in both the CPU and FPU. Furthermore, FPU exceptions are always context specific. That is, any FPU exception occurring after a context switch will remain pending until the FPU returns to the prior context.

FPU exceptions can only be taken and handled when unmasked (referred to as alternate exception handling). The FPU will return the calculated result of each operation and signal any exception via an interrupt to the CPU.

If FPU exceptions are masked, the FPU will return a default result for each operation that generates an exception as defined in Table 3-16. The exception will be signaled by setting the corresponding bit(s) in the FSR, but no interrupt will be issued to the CPU. This is intended to allow code to execute unhindered by exception handling at the time of execution. If required, exception status may be examined at a later time and appropriate action will be taken.

3.6.8.5.1. Huge Integer and Subnormal Exceptions

In addition to the IEEE 754-2008/2019 compliant exception support, the FPU also offers two exceptions and associated masks that some users may find useful.

- Huge Integer: FSR.HUGI
  An exception is signaled whenever a Float-to-Integer conversion operation (FF2DI and FF2LI) results in an integer value that is larger than the destination register can represent.
- Subnormal Operand: FSR.SUBO
  An exception is signaled whenever an operand of an affected instruction is a subnormal value and Subnormals-Are-Zeros (SAZ) mode is disabled (FCR.SAZ = 0). This is the only exception that can be triggered by an operand source condition (all others are related to result conditions).

Table 3-16. Default Exception Results

Exception	FSR Bit Name	Default Result
Invalid	INVAL(2)	Distinguished qNaN or quieted sNaN or Largest integer result (for FF2DI/FF2LI only)
Divide By Zero	DIV0	Correctly signed Infinity(3)

Notes:
1. Under default exception handling, UDF is only set (along with INX) if the result is an inexact underflow. Applies irrespective of whether FTZ mode is enabled or not.
2. FCPS and FCPQ do not generate a result other than an FSR update. However, INVAL will be set by FCPS if either or both operands are a qNaN or sNaN, or by FCPQ if either or both operands are an sNaN.
3. 0/0 is a special case (where both the dividend and divisor are not finite) which will return the distinguished qNaN as the result. INVAL will be set, but DIV0 will not.

Table 3-16. Default Exception Results (continued)

Exception	FSR Bit Name	Default Result		
Overflow	OVF	Rounding Mode	Nearest (Even)	Infinity with a sign of exact result
			Zero	Most positive finite number with sign of exact result
			+Infinity	Positive overflow: +Infinity Negative overflow: Most negative finite number
			-Infinity	Positive overflow: Most positive finite number Negative overflow: -Infinity
Underflow	UDF(1)	FCR.FTZ = 0: Rounded subnormal result		
		FCR.FTZ = 1; Zero with a sign of exact result		
Inexact	INX	Rounded (inexact) result		
Huge Integer	HUGI	Largest integer value with a sign of input operand		
Subnormal Operator	SUBO	N/A (input operand exception)		

Notes:

1. Under default exception handling, UDF is only set (along with INX) if the result is an inexact underflow. Applies irrespective of whether FTZ mode is enabled or not.
2. FCPS and FCPQ do not generate a result other than an FSR update. However, INVAL will be set by FCPS if either or both operands are a qNaN or sNaN, or by FCPQ if either or both operands are an sNaN.
3. 0/0 is a special case (where both the dividend and divisor are not finite) which will return the distinguished qNaN as the result. INVAL will be set, but DIV0 will not.

3.6.8.6. CPU to FPU Interface

The CPU can issue instructions to a coprocessor (FPU), and directly read and write to FPU registers. However, coprocessors otherwise operate independently of the CPU instruction pipeline, executing their instructions within their own pipeline hardware.

An FPU can only receive, send and process data that are funneled through (and under the direction of) the CPU. No CPU addressing capability is shared with an FPU. Consequently, an FPU can only support register direct addressing for all instruction source or destination addressing modes that target a FPU register. Data flow to and from each FPU is controlled using dedicated move instructions that execute within the CPU. Because the CPU and FPU pipelines execute independently, data related hazards that may arise when moving data between the CPU and an FPU are mitigated using a simple request/grant bus which will stall the CPU as needed.

The CPU supports speculative execution of instructions that immediately follow a conditional branch. These could be FPU instructions, so a mechanism exists to allow the CPU to cleanly kill these instructions should the branch prediction prove incorrect.

In case an FPU SFR read is killed, all FPU SFRs (e.g., status and control registers) are defined such that a read of any SFR is not destructive within itself. This will avoid the possibility of a killed SFR read affecting the state of the FPU.

3.6.8.6.1. FPU Pipeline Operation

The CPU decodes all coprocessor instructions during the F-stage. The source and destination coprocessor registers are extracted from the opcode and supplied to the coprocessor, along with a corresponding instruction select and control signals such that no instruction decode is necessary within the coprocessor.

The FPU pipeline stages consist of Read (RD), Execute (X[n]) and Write-Back (WB) stages. The Read and Write-Back stages consist of a single register and are common to all instructions. The Execute stage consists of as many stages as required to execute the specific instruction (i.e., X [0], X[1]..... X[n]) but at least X [0].

One instruction may be issued into the RD-stage, where it will remain for one cycle (hazards aside) until dispatched into the X [0] stage. The number of cycles each instruction remains within the execute phase varies depending upon the operation. In order to avoid stalling the pipeline for the duration of any long instruction, up to four instructions may be dispatched into X[0] and executed concurrently (structural hazards aside).

Instructions retire in the same order in which they are issued. As a consequence of being able to execute multiple instructions with varying execution times, the pipeline Instruction/Hazard Tracker logic ensures that in-order retirement is maintained.

All instructions with an execution latency of four cycles or less are implemented such that the execution stages are fully pipelined. Consequently, assuming no data dependencies (hazards) arise, these instructions can be repeatedly issued at a rate of one per cycle (and receive their results at a rate of one per cycle after an initial execution latency), without incurring a structural hazard stall.

For instructions where the execution latency exceeds four cycles (FDIV and FSQRT), the FPU pipeline will fill the instruction and then stall subsequent instructions (due to a structural hazard) until the required execution resource becomes available.

- FDIV: Floating-point divide is implemented as an iterative operation such that the input data cannot be pipelined until all iterations have completed and the result is passed onto the adjustment stage within the Functional Block. For example, should the CPU issue two sequential FDIV instructions, the second FDIV instruction will stall in the RD-stage until the first FDIV enters the final execution cycle, at which point the second FDIV may be dispatched into the execute stage to commence execution.
- FSQRT: Floating-point square root requires 10 (Single Precision) or 13 (Double Precision) cycles to execute. The hazard tracker can handle up to four issued instructions, so an FSQRT followed by up to three sequential FPU instructions (including FSQRT) may be executing at any one time. The CPU may issue one more instruction, but it will remain in the RD-stage until the oldest FSQRT instruction underway enters the WB-stage, six cycles later, and subsequently retires. At this point, one slot within the hazard tracker is now available for use, and the pending FPU instruction will be committed for execution. Another FSQRT instruction will retire in the next cycle, opening another hazard tracker slot for another issued FSQRT instruction, and so forth, until the hazard tracker is full again and the pipeline must wait a further six cycles for the initial FSQRT to retire. For FSQRT alone, the best case block repeat rate is therefore one per cycle for the initial 4 FSQRT instructions issued, with a subsequent four FSQRT instructions to be issued after six (Single Precision) or nine (Double Precision) cycles have passed. This supports an average execution time of (4+6)/4 or 2.5 cycles/instruction (Single Precision) or (4+9)/4 or 3.25 cycles/instruction (Double Precision).

FPU Read Stage

The FPU pipeline RD-stage receives instructions issued by the CPU. The CPU issues FPU instructions from the A-stage into the FPU RD stage which consists of a single register, such that only one FPU instruction can be held at any one time. The instruction is committed when it is dispatched to X[0], where it will start execution. X[0] holds the instruction such that the CPU is free to issue another instruction into the RD-stage.

The RD-stage is also subject to hazard checks and can therefore be stalled. Should a RAW hazard be detected with a prior instruction that is already executing within the FPU pipeline, the hazard will be detected in the RD-stage which will then be stalled until such time that the hazard is resolved.

Should the CPU subsequently attempt to issue additional FPU instructions, the RD-stage will not be able to accept them which will also stall the CPU until such time that the RAW hazard has been resolved. From the CPU perspective, this scenario is viewed as a structural hazard.

The RD-stage will also stall the CPU under the following conditions:

1. Whenever the number of instructions (default value is four) are in their execute X[n] stages, an instruction is pending in the RD-stage, and the CPU is attempting to issue a further instruction. In this situation, the Instruction/Hazard Tracker is full so the FPU cannot dispatch another instruction from the RD-stage into X [0] until one of the instructions currently executing passes into the WB-stage. Assuming the default value is four, this can occur when the pipeline is executing instructions that take longer than four cycles to execute, and additional FPU instructions are issued while the long instruction is still executing (i.e., not yet in the WB-stage). The longer instruction(s) execute and retire at a rate which is slower than the rate at which the Instruction/Hazard Tracker can be filled, resulting in the CPU being stalled.
2. Whenever the CPU attempts to issue more than two FDIV instructions while a previously issued and a dispatched FDIV instruction is still executing (i.e., not yet in the WB-stage). FDIV is a special case where no more than one instance can be executed within the pipeline at any one time. Consequently, executing another FDIV while a prior instance is still executing will cause this second FDIV to be issued but held pending in the RD-stage (i.e., CPU will not stall). But attempting to issue a third FDIV instruction, while the pending (second) instance has not yet been dispatched to X[0], will result in a CPU (issue) stall. The RD-stage also includes special logic to support the FAND and FIOR operations (refer to FAND and FIOR Instructions).

FPU Execute Stage

Each instruction may consist of one or more execute stages depending upon the functional block targeted by the operation. When the instruction enters the X [0]-stage, it is registered such that the RD-stage is free to receive another instruction issued by the CPU.

All instructions (other than FDIV) are pipelined through as many X[n] stages as deemed necessary to meet the timing requirements.

The pipeline stages will be added such that the propagation delay of each is as balanced as possible, and the sequential issue of the same instruction may be fully pipelined (i.e., instructions using the same functional block may be sequentially issued without incurring a structural hazard in the Execute stage).

FPU Write-Back Stage

The WB-stage captures each Single Precision or Double Precision result as they exit the Execute stage in dedicated registers. FPU instruction execution time is variable, but only one instruction is permitted to be in the WB-stage at any one time. If more than one instruction has completed execution and is in a position to retire, the pipeline will determine which instruction to retire to maintain the instruction execution order and eliminate any WAW hazards. The instruction will then complete the write back in one cycle during the WB-stage before being retired. The Instruction/Hazard Tracker logic will ensure instructions enter the WB-stage in the same order as they were issued.

Prior to writing the result, if FTZ mode is enabled (see Flush-To-Zero (FTZ)), the result is modified accordingly if subnormal. This final value is also passed onto the RAW hazard mitigation forwarding logic.

`FAND` and `FIOR` Instructions

The `FAND` and `FIOR` instructions operate with a 16-bit literal and can only target the FCR, FSR and FEAR. They are considered a special case as they are executed using custom blocks that are implemented within the RD-stage for some FCR bits and the WB-stage for everything else.

To allow subsequent instruction to immediately use `FAND` and `FIOR` changes to FCR.RND [1:0] and FCR.SAZ control bits without (RAW hazard) stalls, these bits are modified during the RD-stage then updated at the end of the RD-stage, so they are available for immediate use by any subsequent instruction.

The remaining FCR bits and all FSR and FEAR bits are read, modified and written back during the WB-stage. Reading the FSR late (i.e., in the WB-stage rather than the RD-stage) avoids a potential

RAW hazard arising between a prior instruction FSR update and a subsequent FAND or FIOR FSR operation.

3.6.8.7. FPU Hazards

The coprocessor interface can suffer from structural and data dependencies as described in the following sections. RAW, WAR and WAW data hazards are possible; RAR hazards are not.

3.6.8.7.1. FPU Structural Hazards

When a requested FPU resource is unavailable, a structural hazard will be detected. This may result in the coprocessor stalling the CPU until the hazard is resolved.

Hazards that arise from actions within the FPU are referred to as “internal” hazards. Those that arise due to actions between the CPU and FPU are referred to as “external” hazards. Depending upon how the CPU/FPU pipeline is viewed (separate or conjoined), some of these hazards may be viewed as either structural (i.e., a resource is unavailable) or data related.

FPU Pipeline Full or Busy

When the CPU attempts to issue an instruction to the coprocessor, and it is unable to accept it because the pipeline is full or busy, an external structural hazard will result, and the coprocessor will stall the CPU until such time that the instruction can be accepted.

When an issued instruction is stalled in the FPU RD-stage due to a RAW hazard with a prior currently executing instruction, the FPU pipeline is considered busy such that further FPU instructions cannot be accepted. Consequently, should the CPU attempt to issue any additional FPU instructions while the RD-stage is stalled, the FPU will stall the CPU until such time that the hazard resolves, resulting in an external structural hazard as shown in Figure 3-34.

The pipeline is considered full when the Instruction/Hazard Tracker FIFO is full, which occurs when the number of instructions (default value is four) are active within it, including the one waiting in the RD-stage for dispatch into X[0]. The pipeline will remain full until the oldest instruction enters the WB-stage. Should the CPU attempt to issue another FPU instruction, the FPU will stall the CPU until such time that the Instruction/Hazard Tracker FIFO is no longer full.

FPU Functional Block Unavailable

If the FPU pipeline is not full, and the FPU attempts to dispatch an instruction from the RD-stage that uses a functional block that is already in use by a prior instruction, an internal structural hazard will result, and the RD-stage will be stalled until such time that the functional block is no longer in use. If the CPU attempts to issue another FPU instruction before this occurs, the FPU will then stall the CPU until the hazard resolves.

This scenario can arise as a result of in-order retirement where instructions that target the same functional block will be stalled in the pipeline waiting for slower, older instructions to complete execution. An example is shown in Figure 3-3 where a slow instruction (`FSIN`) is followed by multiple instructions that target the same MISC_SP functional block. The first `FMOV` will stall in X [0] waiting for the FSIN to retire, resulting in an internal structural hazard. The subsequently issued `FMOV` will issue but will be stalled in the RD-stage because it cannot progress into X [0] until the first `FMOV` is able to move into the WB-stage, another internal structural hazard. As the RD-stage is now stalled, should the CPU attempt to issue any additional FPU `FMOV` or `FMOVC` instructions (which share the same functional block), the FPU will stall the CPU until such time that the pipeline can advance again, causing an external structural hazard.

This scenario will always arise for a sequential issue of the multi-cycle iterative FDIV instruction (all other instructions can be pipelined) as shown in Figure 3-4.

FPU Register Unavailable to Read

When the CPU attempts to read a register that is bound to an issued FPU instruction, an external structural hazard will result and the coprocessor will not be able to read until the register becomes available, creating a read stall for the CPU.

FPU Register Unavailable to Write

When the CPU attempts to write to a register that is bound to an issued FPU instruction, the coprocessor will not be able to write until the register becomes available, creating a write stall for the CPU (see FPU WAW Hazards).

3.6.8.7.2. FPU Data Hazards

The coprocessor interface can suffer from data dependencies leading to RAW, WAR and WAW data hazards (RAR hazards are not possible). Unlike the CPU integer pipeline, a coprocessor hazard does not necessarily prevent the pipeline from progressing for other coprocessor instructions, unless subject to other hazards.

Internal RAW Hazards

An internal RAW (Read-After-Write) data dependency hazard will occur when the result of an FPU instruction is not available at the time it is selected as the source operand (F-reg) of a subsequent FPU instruction. The affected instruction will be stalled in the RD-stage until such time that the hazard is resolved.

In order to mitigate the hazards, the coprocessor includes data forwarding paths between the FPU execution result data output and the coprocessor RD-stage (as shown in Figure 3-8). This path will forward the write data value should the write and read instructions target a common register. Forwarding as soon as the result data is available (i.e., prior to the FPU register write) will help mitigate the impact of the hazard.

External RAW Hazards

An external RAW (Read-After-Write) data dependency hazard will occur should the contents of an FPU register be unavailable at the time it is read by the CPU because the register is bound to a previously issued FPU instruction. The coprocessor will detect the hazard and read will be stalled until such time that the register becomes available (i.e., after the result has be written), creating a read stall for the CPU.

In addition, an external RAW hazard will occur if:

- A CPU write to a coprocessor register is immediately followed by the CPU issuing a coprocessor instruction that uses the same register as an operand source.

  or

- A CPU write to a coprocessor register is immediately followed by a CPU read of the same register.

In both of these CPU RAW hazard scenarios, the CPU is responsible for detecting the hazard and inserting the necessary stall cycle for the coprocessor to resolve the hazard. Hazard detection is the same for both scenarios.

In order to resolve these hazards, the coprocessor includes data forwarding paths between the CPU W-stage and both the coprocessor RD-stage (as shown in Figure 3-5) and the CPU read data output (as shown in Figure 3-6). These paths will forward the write data value should the write and read instructions target a common register.

Note: When the CPU attempts to write to an F-reg, it is possible that an instruction in the RD-stage is using the same register as an operand source, and it is stalled as the result of an internal RAW hazard. Forwarding the new CPU write data into this register would then be incorrect because the RD-stage instruction was issued prior to the CPU write instruction. Consequently, should the instruction in the RD-stage have been there for more than one cycle (i.e., be stalled), the FPU will disable the forward path and allow the stall mechanism to recognize the hazard as usual. This will prevent the CPU write from completing until such time that the instruction in the RD-stage has been dispatched to start execution.

CPU write data forwarding to the coprocessor RD-stage allows the CPU to issue a coprocessor instruction earlier than would be possible if the CPU coprocessor write had to complete. CPU write data forwarding to the CPU read data path together with a CPU stall cycle (detected and inserted by the CPU) resolves the (unlikely) hazard that arises when a CPU write is followed immediately by a CPU read of the same coprocessor register. The converse scenario where a CPU read of an FPU register into a W-reg is immediately followed by a CPU write of the same W-reg to another F-reg is shown in Figure 3-7.

FPU WAR Hazards

The WAR (Write-After-Read) anti-dependency hazard can occur should the pipeline allow read and write execution to be out of (instruction sequence) order. That is, a WAR hazard will arise whenever an instruction writes to a register before the same register is read by a *prior* instruction. That is, the read and write occur out of execution order resulting in the (older) read instruction ultimately using the (later) write data which would be incorrect.

Under normal sequential execution conditions, a WAR hazard should never arise because the read of all older instructions always precedes the writes of later ones. However, a WAR hazard can arise within the coprocessor pipeline when a slow instruction (e.g., `FPU FSIN`) has a result data dependency (RAW hazard) with a later instruction, and that later instruction is followed by a `MOVWCR` or `POPCR` instruction that targets the same register as the dependency. This is because the dependency will force an FPU pipeline stall until the result data are available and the RAW hazard is resolved, but the `MOVWCR` or `POPCR` move instructions (which do not execute using the FPU pipeline) will not be stalled. Consequently, it is possible that the write from the `MOVWCR` or `POPCR` instruction would occur prior to the stalled instruction continuing execution (after the RAW hazard). The write would then be overwritten by the FPU pipeline and, therefore, lost. This scenario is detected as a WAR hazard and prevented from happening by stalling the most recent write, such that the write order remains correct. CPU to FPU move instructions that do not target the register involved in the stall will still execute as normal (i.e., without stalling).

The coprocessor must, therefore, detect the possibility of such a hazard and force in-order execution of all dependent instructions by stalling the most recent CPU write instruction in the W-stage until after the prior read is completed.

An example WAR hazard and its resolution are as follows: A RAW hazard between the `FSIN.s` and `FMOV.s` instructions will stall `FMOV.s` to resolve the hazard (stall cycles shown in green), but this will also set the pipeline up for a possible WAR hazard because the subsequent `MOVWCR` instruction is not prevented from continuing execution.

As is the case in this example, should the `MOVWCR` instruction destination be the same F-reg as that used by the `FMOV.s` as a source, the `MOVWCR` must be prevented from writing until the prior (`FSIN.s`) has been able to forward the write data to the `MOV.s` RD-stage. The `FSIN.s` and `MOVWCR` enter their respective write stages together, and the FPU prioritizes the CPU write, maintaining correct write ordering. This results in a one-cycle stall of the `MOVWCR` instruction to resolve the hazard.

FPU WAW Hazards

The WAW (Write-After-Write) hazard is a further consequence of allowing instructions to continue execution while others are stalled or taking longer to execute. As is the case for WAR hazards, instruction writes can end up out of order, leaving an incorrect (stale) value in a destination register.

WAW output dependency is possible because once the coprocessor instruction is issued, the CPU and coprocessor pipelines operate independently. A multi-cycle coprocessor instruction may therefore complete after one or more CPU instructions that were subsequently issued (i.e., out of order). A WAW hazard will exist when the CPU instruction is ready to retire before the coprocessor instruction retires and either:

- The same register is a destination for both the coprocessor instruction and the CPU instruction that follows it.

  or

- The CPU instruction write targets a coprocessor register that is being used by the prior coprocessor instruction.

In both cases, the resource cannot be shared.

An internal WAW hazard can arise between successive FPU instructions that have differing execution times. However, each issued instruction is tracked by pushing its associated functional unit ID into a FIFO, which is emptied in the same order as it is filled when instructions move results from their functional units into the WB-stage. Should an expected (from the FIFO) functional unit result not be ready, this knowledge is used in the Write stage to complete the destination write in the correct sequence, stalling those instructions that arrive out of order, thereby eliminating the WAW hazard.

If the CPU and FPU pipelines are viewed as conjoined, a WAW hazard is also possible should the CPU attempt to write a value to the same register as also targeted by a previously issued FPU instruction whose write has not yet completed. However, access to the write target(s) of an instruction is inhibited as soon as the instruction is committed (see FPU and CPU Exceptions). Consequently, any attempt by the CPU to write to an FPU register that is already bound to a prior FPU instruction being executed will result in the write grant failure (and the CPU write stalling).

Note: For the purpose of WAW hazard detection, the FSR is considered as a single entity.

Note: The FSR is bound to all FPU instructions except for `FMOVC` and `FMOV` (these ops do not update the FSR), and FAND and FIOR, unless they will modify the FSR. The FEAR is bound to all FPU instructions except for `FMOVC`, `FMOV` and `FTST`. It is also not bound to FAND or FIOR unless it will be modified by them. Note that this applies irrespective of whether FEAR is enabled or not (i.e., FEAR.EACE is a “don’t care” with respect to FEAR hazard detection).

3.6.8.7.3. Instruction/Hazard Tracker

The Instruction/Hazard Tracker is a mechanism whereby hazard-related information is required while an instruction is progressing through the execute stages and is captured in a FIFO for each issued instruction when that instruction is committed and enters the FPU pipeline X [0]-stage. The FIFO depth (default is four) defines how many instructions may be sequentially dispatched into the Execution stage before it is regarded as full.

Each FIFO entry includes the following information which is used during the X-stages.

1. Entry valid flag
2. Flags to indicate which functional block (function and operation precision) is targeted.
3. Operand source register identification and valid flag such that RAW hazards may be identified as the instruction progresses.
4. Flags to support Single Precision and Double Precision NaN propagation logic.
5. Flag to indicate if an instruction is FDIV or FSUB (where the operand order is reversed).
6. Flag to indicate if an instruction is FMAC (special case for NaN propagation).

Each FIFO entry requires a ‘valid’ bit which is clear whenever the entry is empty or after it has been used in the WB-stage. This bit will inhibit any associated hazard detection after an instruction has retired.

Operation precision partially identifies the selected functional block but also directs the hazard logic. Single Precision operations need to only check for hazards involving single F-regs whereas Double Precision must check F-reg pairs for hazards.

Operand register identification and valid flags log which F-regs are used for operands (not all instructions require all three source operands) for hazard tracking. In addition, each FIFO entry includes the following information (also detected in the RD-stage) which is used during the instruction WB-stage.

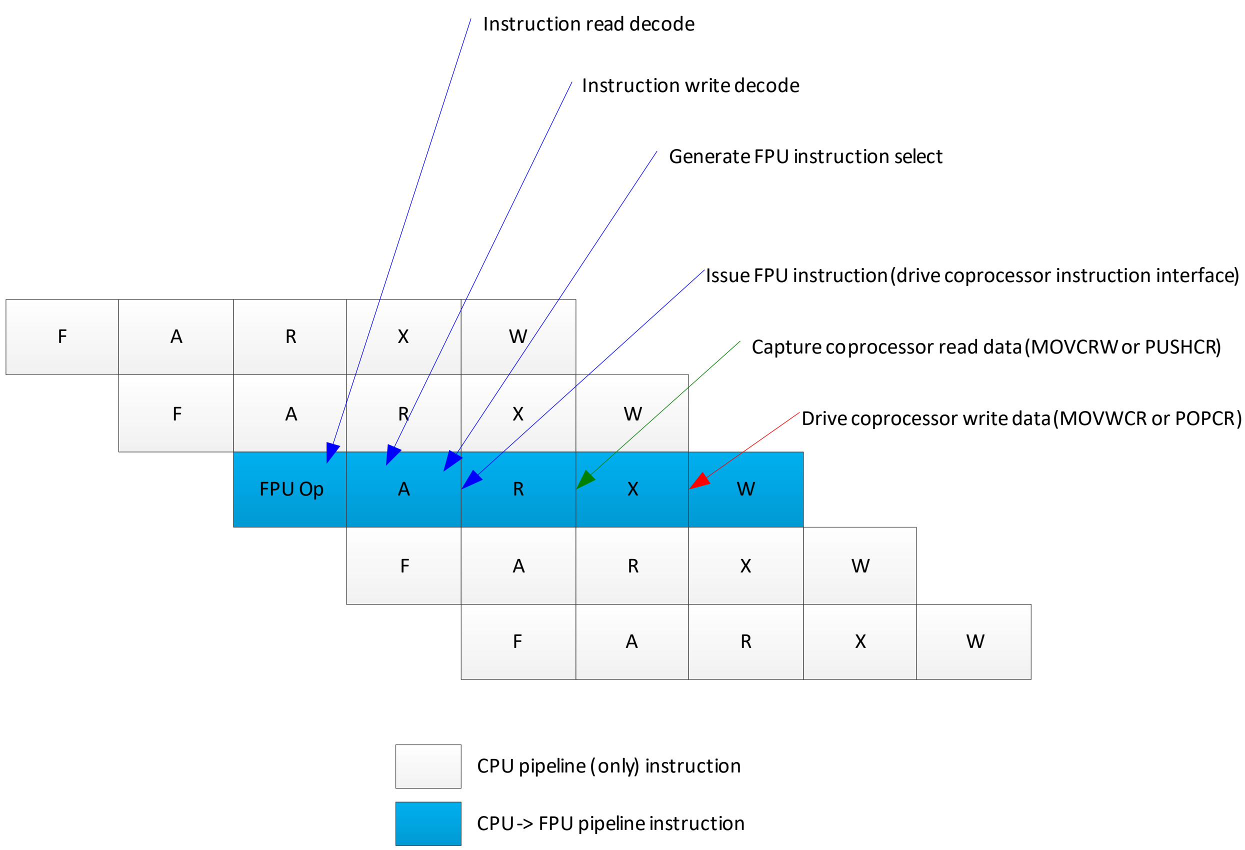
1. Flags to indicate if any result is to be written to an F-reg and whether the FSR is to be updated.
2. Result destination register (and context) identification (defined as DP targets). Additional flags select the active registers (i.e., DP F-reg pair or one of two SP F-reg destinations).
3. Flag to indicate if the instruction permits a FTZ override of the result.

4. CPU A-stage instruction address to capture in the FEAR (if enabled) should the instruction generate an exception.
5. Flags to indicate if the instruction is a FAND or IOR, and the associated FSR/FCR/FEAR target register select bits.
6. The presence of a subnormal operand (when SAZ mode is disabled) is captured and used to signal the subnormal exception (i.e., at the same time as any other exceptions the instruction may generate).

3.6.8.7.4. CPU Write Stalls

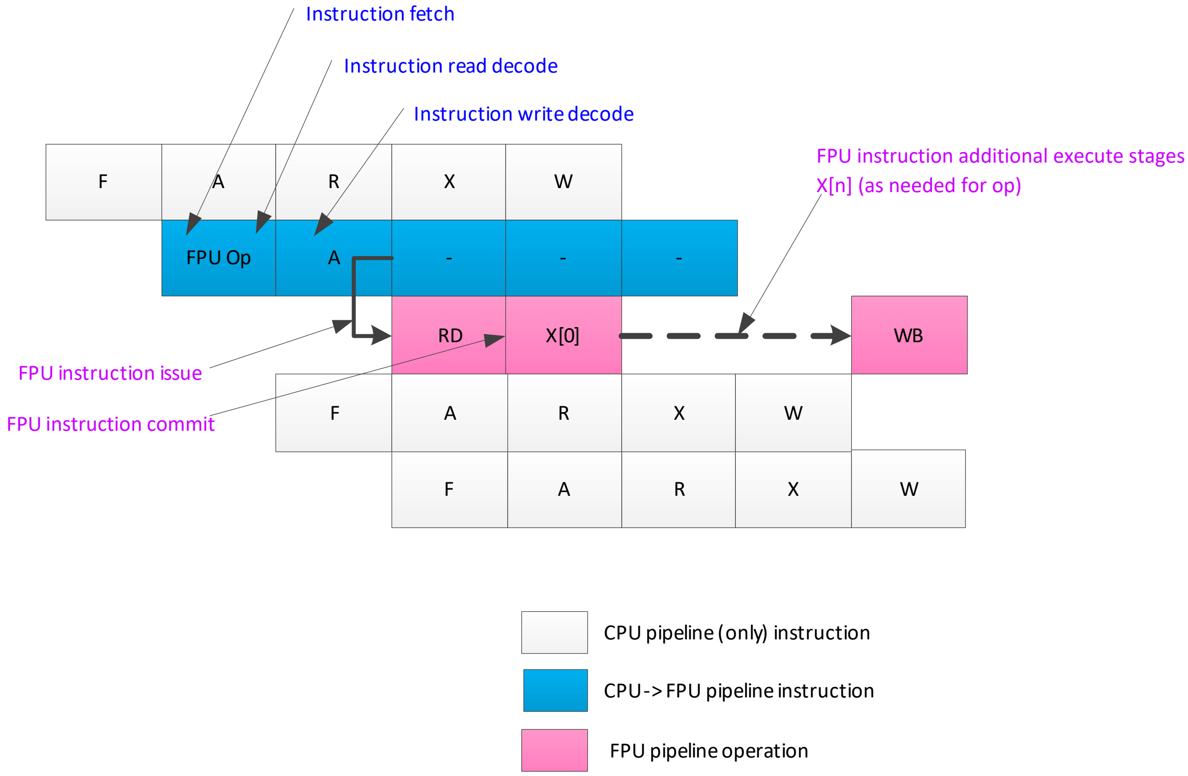
Whenever the CPU encounters a write stall, the entire integer pipeline is stalled (because the CPU only supports in-order execution). No subsequent instruction is permitted to move into the W-stage to retire until the write stall is resolved. Different Pipeline stages are explained in FPU Pipeline Operation.

Figure 3-32. CPU Pipeline Coprocessor Interface Flow



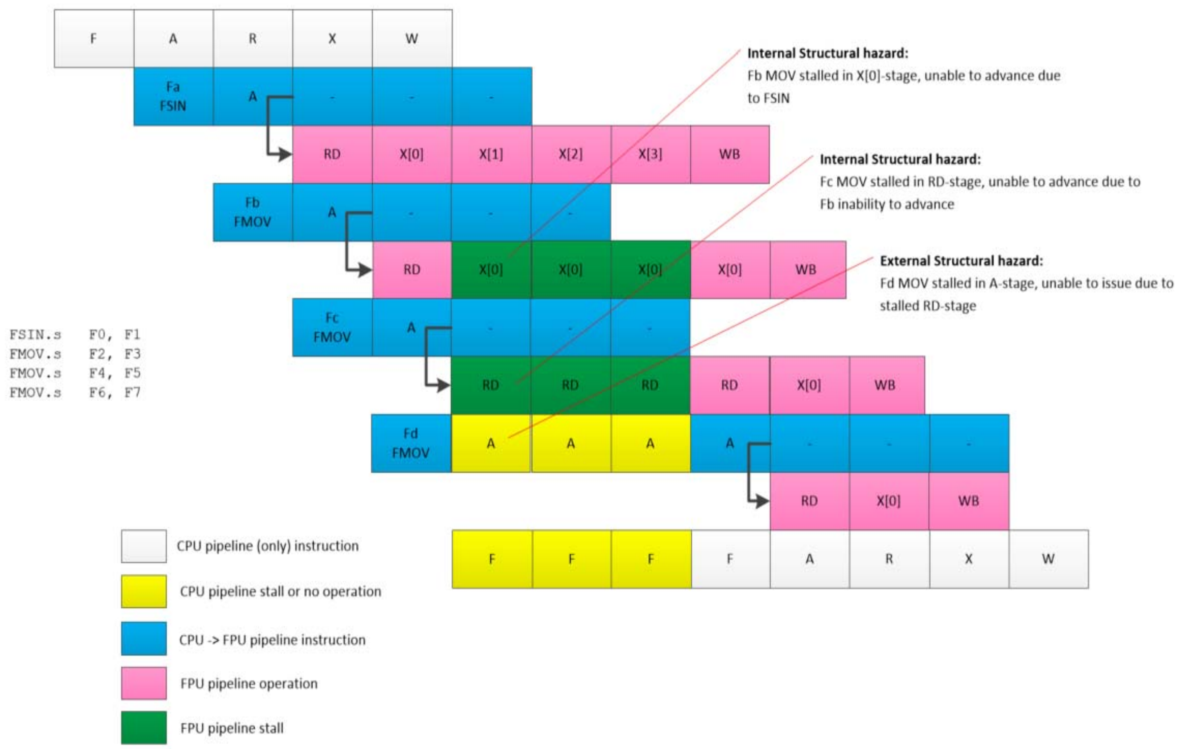
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-33. CPU Pipeline Coprocessor Issue Flow



Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-34. Pipeline and Functional Block Busy Internal/External Structural Hazards

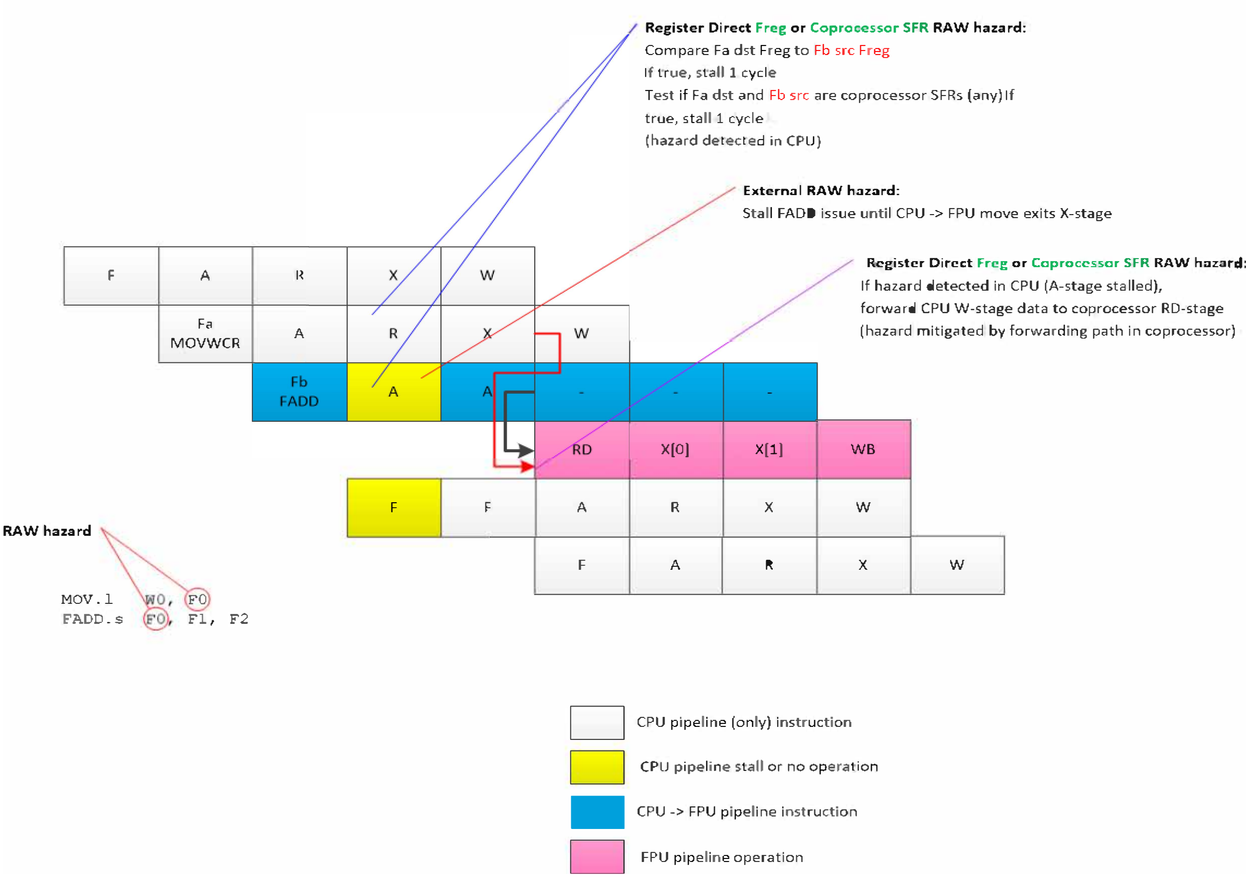


Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-35. FDIV Pipeline and Functional Block Busy Internal/External Structural Hazards

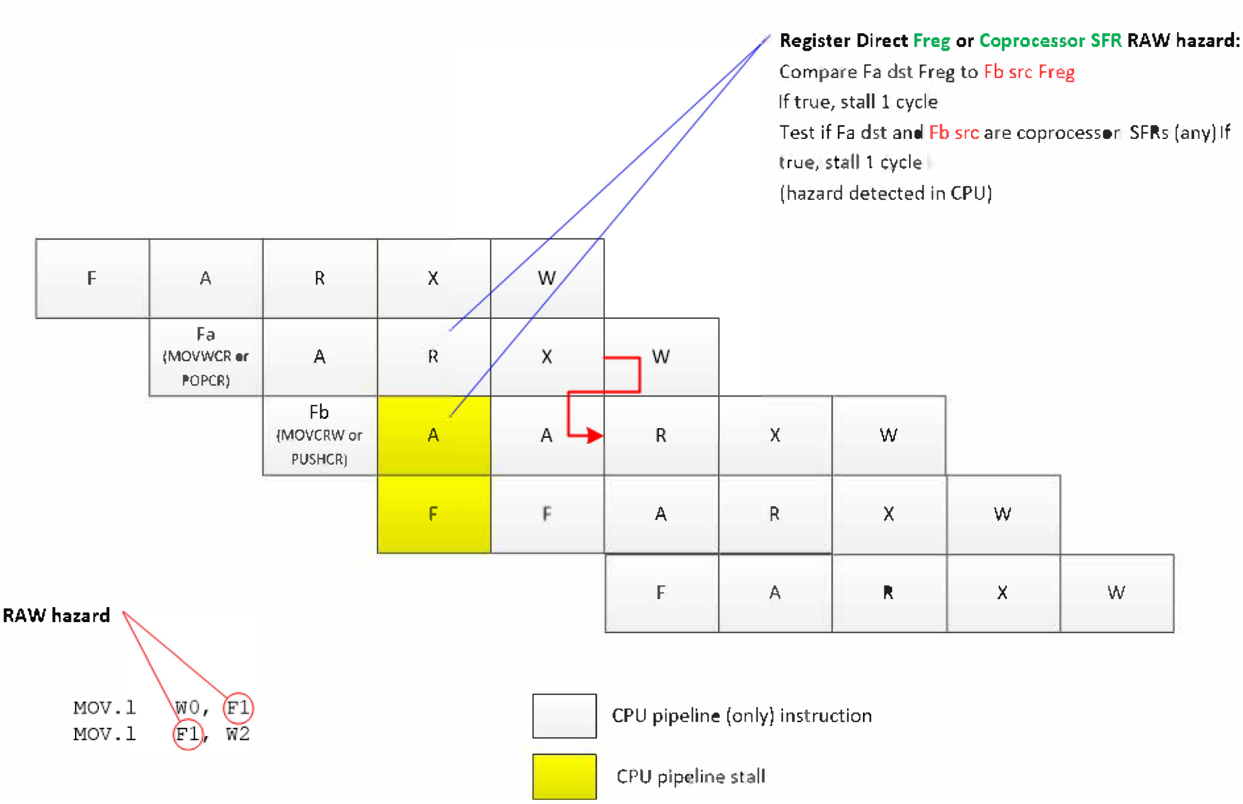
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-36. External RAW Hazard (CPU Write Data to FPU Read Forwarding)



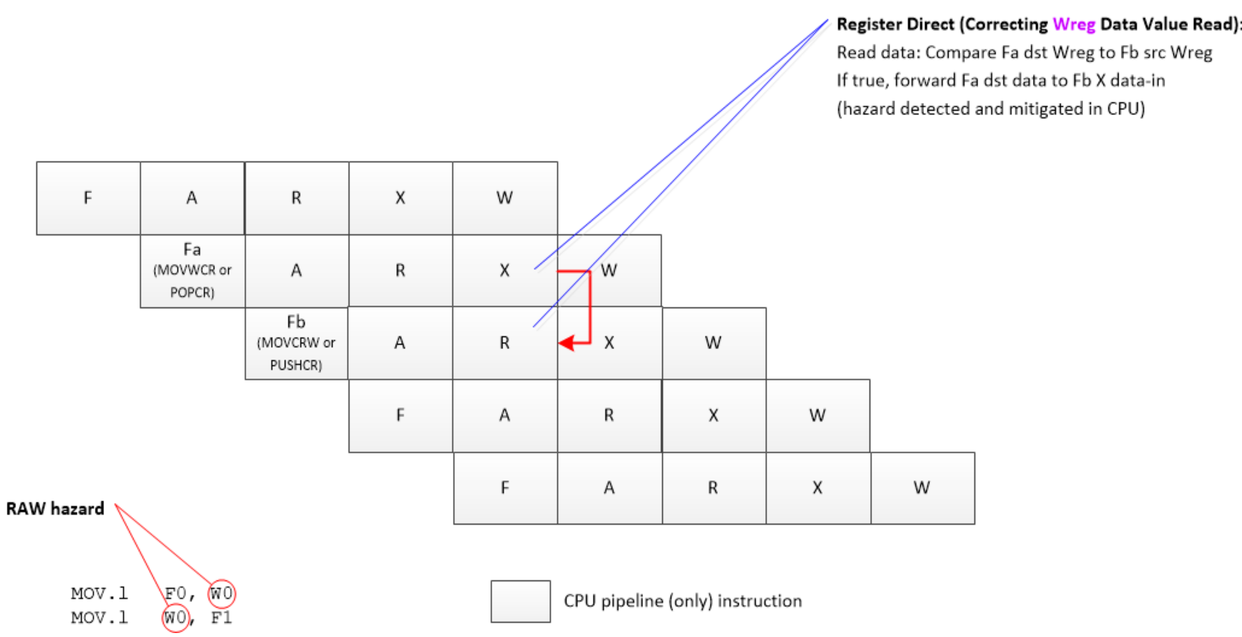
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-37. External Raw Hazard (CPU F-REG Write Data to CPU F-REG Read Forwarding)



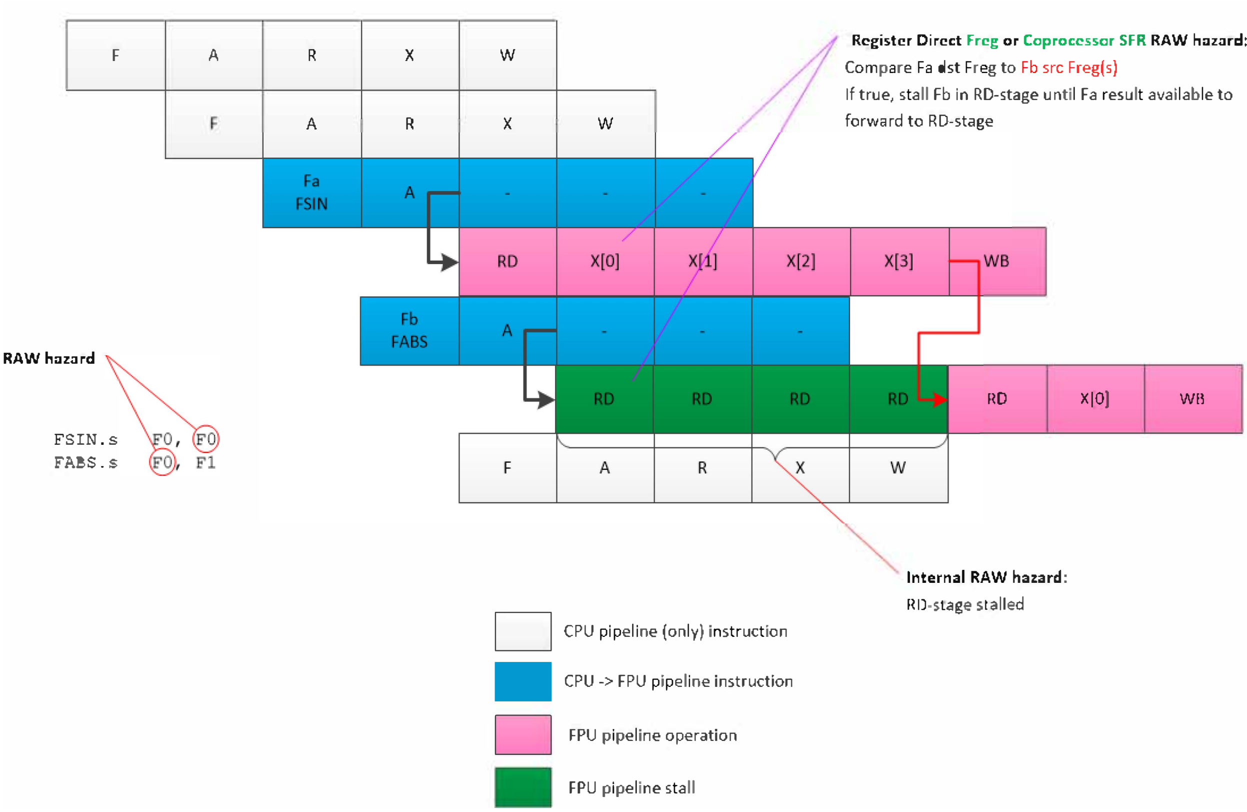
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-38. External Raw Hazard (CPU W-REG Write Data to CPU W-REG Read Forwarding)



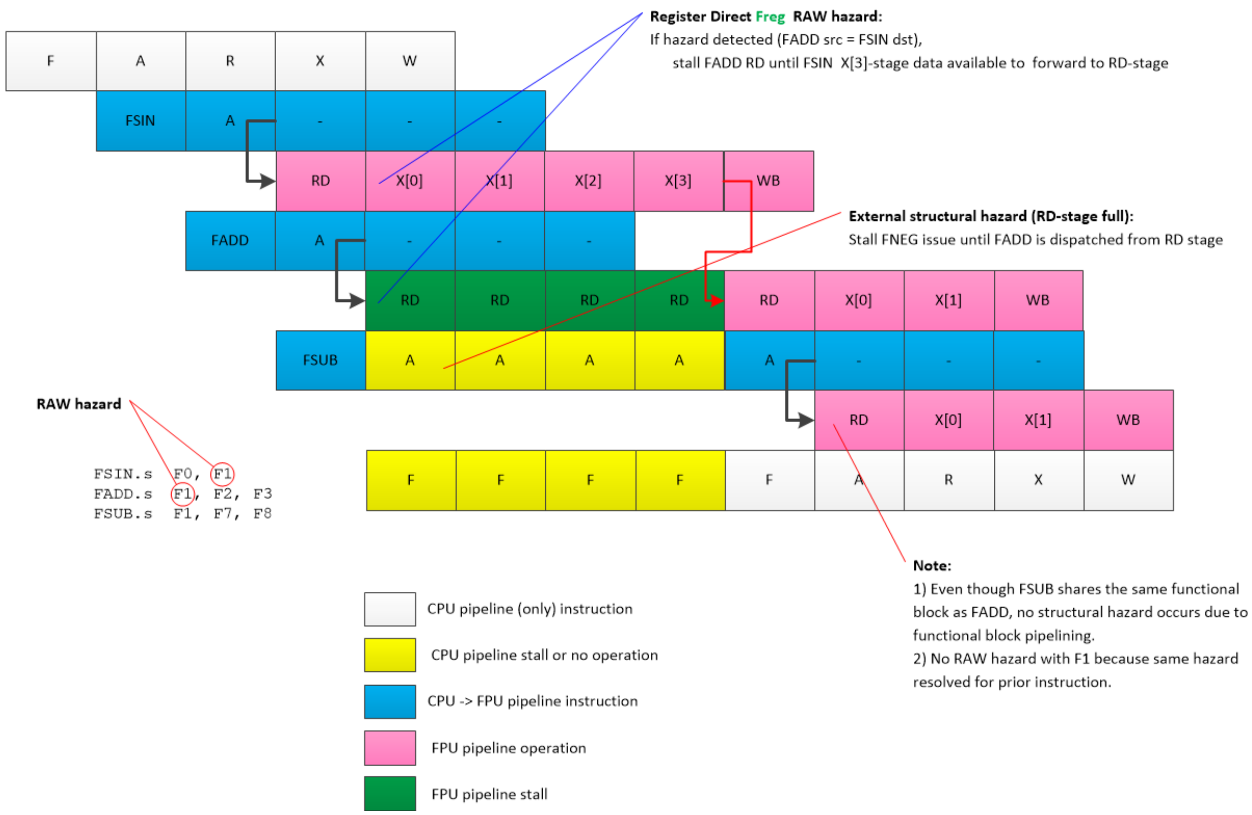
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-39. Internal Raw Hazard (FPU Write Data to FPU Read Forwarding)



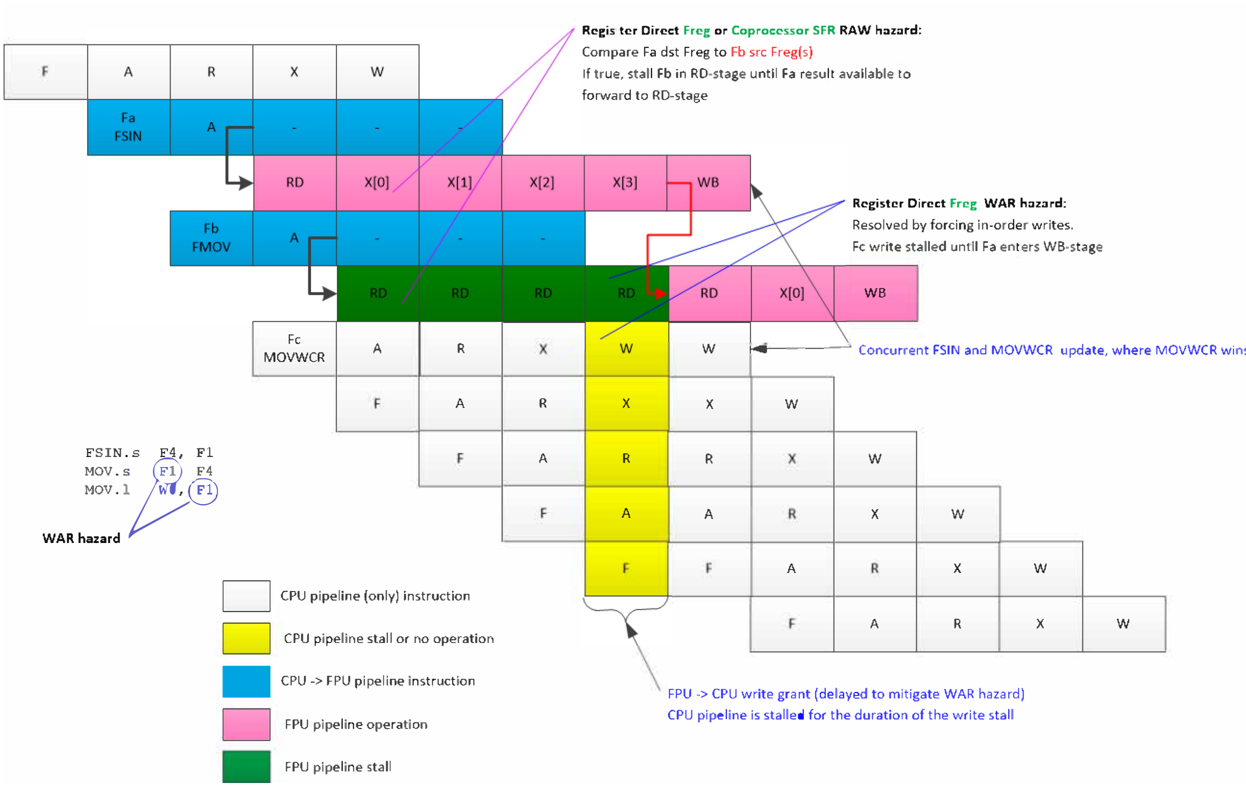
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-40. Internal Raw Hazard, External and Internal Structural Hazards



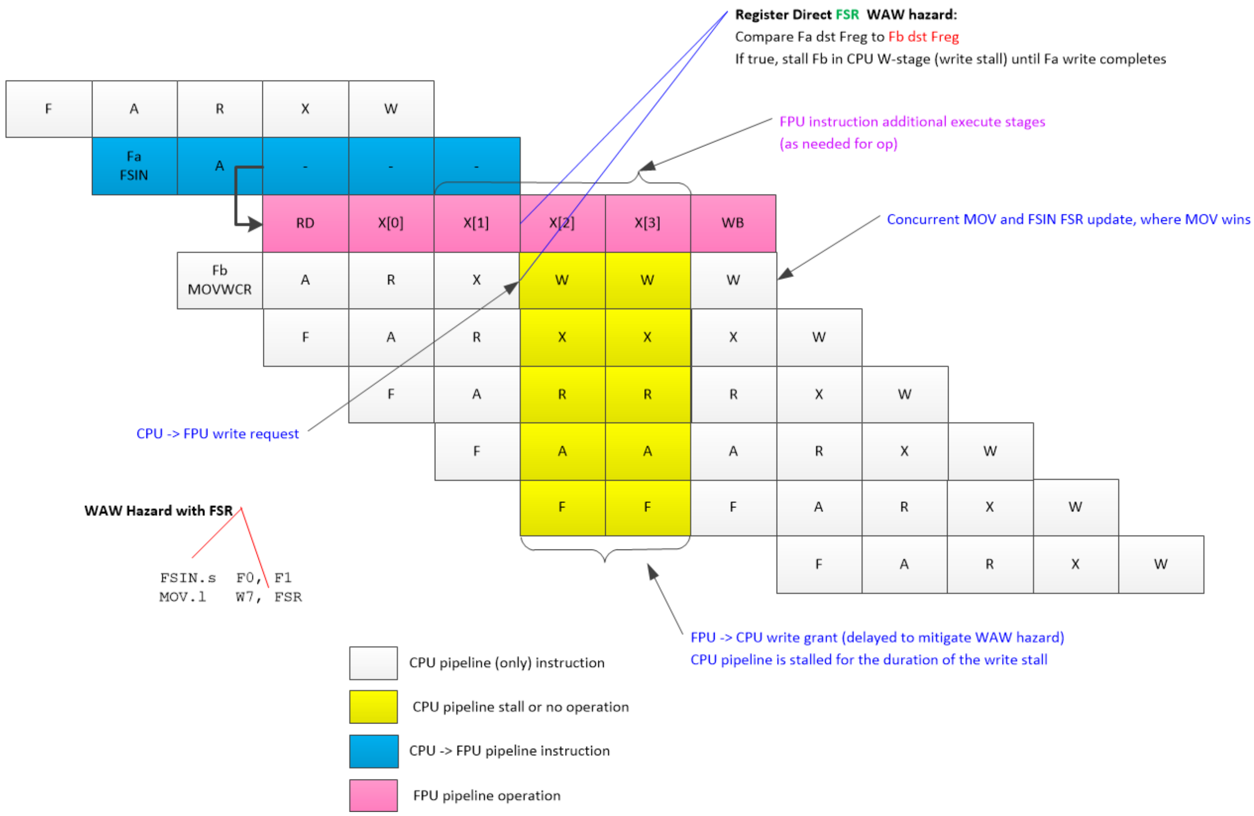
Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-41. FPU WAR Hazard



Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

Figure 3-42. CPU/FPU WAW Hazard



Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

3.6.8.8. Operand Pre-Processing

Floating-point operands are subject to examination during the RD-stage in order to implement NaN propagation and the subnormal value override function. This is necessary to apply rules that determine the outcome in the presence of one or more NaN input values and evaluate operands for special conditions.

3.6.8.8.1. NaN Propagation Operand Detection

For instructions that generate a result, special propagation rules apply when one or both source operands are NaN values, such that sNaNs can be successfully used as "tracer" values. Should a NaN be deemed as propagated, then it will replace the operation result.

With reference to NaN Propagation, all instructions will examine the operands for NaN values during the RD-stage.

- Two operand instructions:
  If one or both operands are NaN values, the RD-stage will apply a propagation priority as shown in Table 3-14.
- Three operand FMAC instructions:
  The source operands are examined in the RD-stage as usual but in conjunction with the selected intermediate result, and any NaN values detected are propagated as shown in Table 3-15.
- Three operand FFLIM instructions:
  If one or both limit operands are NaN values, the RD-stage will apply a propagation priority as shown in Table 3-13. If the FFLIM input value is a NaN, the limit values are ignored and the input NaN value is propagated (quieted if an sNaN).

In all cases, if a NaN is to be propagated, the corresponding NaN value is entered as the operand value in the Instruction/Hazard Tracker FIFO entry for that instruction. The instruction FIFO entry also sets a flag to indicate that NaN propagation is enabled.

3.6.8.8.2. Subnormals Operands

The FPU supports a subnormal operand override mode, Subnormals-Are-Zero (SAZ), the functionality of which is defined in Subnormal Operand Exception. Subnormals-Are-Zero (SAZ) mode is enabled when FCR.SAZ is set.

Should a subnormal operand be detected when SAZ mode is disabled, the subnormal exception will be signaled by setting FSR.SUBO (and FSR.SUBOS if not already set) during the WB-stage (i.e., at the same time as when all other exceptions are signaled). If SAZ mode is enabled, the subnormal exception will not be signaled.

Note: SAZ mode is not applicable to `FAND`, `FIOR`, `FMOV`, `FMOVC` or any CPU to/from FPU data move instruction, none of which can modify any FPU status. In addition, SAZ mode is ignored by `FTST` such that a subnormal operand will always be recognized as such by the instruction, irrespective of the state of FCR.SAZ. However, SAZ mode can influence FF2LI/FF2DI operands. In these cases, subnormal or zero operands will write the same result (integer value of 0). But if the operand is subnormal and SAZ mode is disabled, a subnormal exception will also be signaled. Conversely, if the operand is subnormal and SAZ mode enabled, a subnormal exception will not be signaled.

3.6.8.9. Result Post-Processing

Floating-point results are subject to examination during the WB-stage to implement the subnormal result override Flush-To-Zero (FTZ) mode and NaN propagation results.

3.6.8.9.1. Subnormal Results

The FPU supports a subnormal result override mode, Flush-To-Zero (FTZ), the functionality of which is defined in Flush-To-Zero (FTZ). Flush-To-Zero (FTZ) is enabled when both FCR.FTZ and FCR.UDFM are set. Should the underflow exception be unmasked (FCR.UDFM = 0), then the FCR.FTZ bit will have no effect.

This mode is implemented within the WB-stage such that results written to the destination register (and those forwarded) will be adjusted accordingly if FTZ mode is enabled. The FCR.FTZ bit is only examined during the WB-stage of an instruction such that it may be modified as late as the cycle before the instruction enters the WB-stage.

Note: FTZ mode is not applicable to `FAND`, `FIOR`, `FMOV`, `FMOVC` or any CPU to/from FPU data move instruction, none of which can modify any FPU status. It is also not applicable to FTST because the FSR is the only possible destination for this operation. In addition, FTZ mode will have no effect on FF2LI/FF2DI and FLI2F/FDI2F instruction results because FF2LI/FF2DI results are integers and FLI2F/FDI2F destination data may be zero but never subnormal.

3.6.8.9.2. NaN Propagation Result Write

NaN operand values are detected in the RD-stage, prioritized and then passed (via an Instruction/Hazard Tracker FIFO entry) to the instruction WB-stage. A valid NaN propagation will cause the operation result from the Execute stage to be ignored, and the propagated NaN value to be written into the result destination instead, as discussed in NaN Propagation.

3.6.8.9.3. Rounding Modes

The Rounding mode for each instruction functional block is defined by the value written into FCR.RND [1:0] as defined in FCR. The FPU treats the Rounding mode input as an operand supplied from the RD-stage when the instruction is dispatched into the Execute stage.

Note: Rounding modes are not applicable to `FAND`, `FIOR`, `FCPQ`, `FCPS`, `FTST`, `FABS`, `FNEG`, `FFLIM`, `FMAX`, `FMIN`, `FMAXNUM`, `FMINNUM`, `FMOV`, `FMOVC` or any CPU to/from FPU data move instruction.

There is a 3-bit Rounding mode input (rnd [2:0]) to support up to eight different rounding modes for all FPU conversion operations. Setting rnd [2] = 1 and mapping rnd [1:0] to FCR [9:8] will allow a user selection of the IEEE 754 compliant modes as defined in FCR.

The integer/floating-point conversion instructions (`FDI2F`, `FLI2F`, `FF2DI`, `FF2LI`) may either specify the Rounding mode within the instruction syntax or default to that defined in FCR.RND [1:0]. CPU will issue one of these instructions, and the FPU will use it to determine the Functional Block Rounding mode as shown in Table 3-17.

Table 3-17. FPU Conversion OP Rounding Modes Control

Rounding Mode Bits in Opcode[2:0]	Functional Block Rounding Mode
`111`	IEEE Round to Negative Infinity (floor)
`110`	IEEE Round to Positive Infinity (ceiling)
`101`	IEEE Round to Zero (truncate)
`100`	IEEE Round to Nearest (even)
`0xx`	Global mode (defined by FCR.RND[1:0])

3.6.8.10. Floating-Point Status

The FPU generates four types of statuses.

- Exception condition “most-recent” status from most instructions (see Table 3-20). These bits are located within FSR [6:0]: INX, HUGI, OVF, UDF, DIV0, INVAL, SUBO.
- Exception condition "sticky" status from most instructions (see Table 3-20). These bits are located within FSR [14:8]: INXS, HUGIS, OVFS, UDFS, DIV0S, INVALS, SUBOS.
- Value ordering relations status indicates the result of the FCPS/FCPQ compare instructions. These bits are located within FSR [19:16]: GT, LT, EQ, UN.
- Operand characteristic status from the FTST datum inspection/classify instruction. These bits are located within FSR [28:24]: SUB, INF, FZ, FN, FNAN.

Operand comparisons are likely to be used frequently, so the compare status bits generated by the FCPS/FCPQ instructions are supported with CPU conditional branch instructions. All other statuses must be read into the CPU (using the MOVCRW instruction) or pushed onto the stack (using PUSHCR) and then acted upon as necessary.

Note: Irrespective of whether an exception is masked or not, writing a logic 1 to an exception status flag using any instruction that can write 1’s to the FSR will not result in any associated exception being taken.

3.6.8.10.1. Compare Status and Predicates

IEEE 754-2008/2019 standards specify quiet and signaling compare predicates (equations) as shown in Table 3-18. A “signaling” predicate signals (i.e., attempts to generate an exception) when a Quiet NaN or Signaling NaN (qNaN or sNaN) operand is detected.

A “quiet” predicate will not signal when a qNaN operand is detected.

An sNaN will always signal an exception when detected as an operand for all instructions except for those that do not generate any exceptions (`FMOV`, `FMOVC`, `FABS`, `FNEG` and `FTST`).

The FPU coprocessor supports signaling and quiet predicates by supporting two floating-point compare options, one signaling (FCPS), one quiet FCPQ) and a set of floating-point branch operations that test for the required predicates. Each compare instruction will set one of the four mutually exclusive ordering relations (GT, LT, EQ, UN status bits) located in the FSR to indicate the result of the comparison.

- FCPS (signaling compare)
  - qNaN or sNaN: If either or both operands are a qNaN or sNaN value, the compare is considered unordered which will cause the FSR.UN bit to be set. In addition, the FSR.INVAL bit will be set, causing the CPU to be signaled via the invalid exception (assuming that the exception is not masked).
- FCPQ (quiet compare)

  - qNaN: If one or more operands contain a qNaN value, the compare is considered unordered which will cause the FSR.UN bit to be set. A qNaN will not set the FSR.INVAL bit, so no signaling will occur.
  - sNaN: If either or both operands are a sNaN value, the compare is considered unordered which will cause the FSR.UN bit to be set. In addition, the FSR.INVAL bit will be set, causing the CPU to be signaled via the invalid exception (assuming that the exception is not masked).

The compare operation subtracts Fs (subtrahend) from Fb (minuend). The EQ, GT and LT status bits are set as follows:

- If the minuend is equal to the subtrahend (Fb = Fs), the EQ status bit is set.
- If the minuend is greater than the subtrahend (Fb > Fs), the GT status bit is set.
- If the minuend is less than the subtrahend (Fb < Fs), the LT status bit is set.

In addition, the UN status bit is set if one or both operands is a NaN. If this is the case, no other compare status is set (i.e., UN, EQ, GT and LT are mutually exclusive).

Note: The FCPS/FCPQ instructions consider -0 and +0 as equivalent.

Note: Comparing a value to itself should produce an equivalent result. However, UN has precedence over EQ such that should two values be identical but both are NaN, the UN bit will be set, but the EQ bit will be cleared.

FPU Status Conditional Branches

The CPU has the ability to conditionally branch off various status bits generated within the coprocessor. In the case of the FPU, an internal status register (FSR) is supported which is updated at the end of each floating-point operation.

The FPU FSR is comprised of an instruction exception status and a FCPS/FCPQ/FTST instruction status. Conditional branching is supported within the CPU for the FCPS/FCPQ compare instructions only.

The CPU ISA includes a set of generic coprocessor conditional branch instructions, CBRA0 through CBRA15, and each can operate with any instantiated coprocessor and branch based upon the state of a corresponding bit within a vector supplied by each coprocessor. In the case of the FPU, CBRA0 through CBRA13 are used, each represented as an FBRA instruction with its corresponding assembler attribute for the FCPS/FCPQ instruction status branch conditions. The FCPS/FCPQ status is held in FSR [19:16] and indicates the comparison result. CBRA[n] timing is the same as any other CPU conditional branch, such that the condition is examined at the end of the CBRA[n] R-stage. If the condition is true, the branch is taken. If the condition is not true, the branch is not taken and sequential execution continues.

As is the case for all conditional branches, the instruction(s) immediately following the branch are speculatively executed, and they will either be part of the taken or the not taken path, based on the direction of the branch. These instructions are permitted to be floating-point operations. This requires that the FPU accommodate the possibility that these instructions could ultimately be killed due to a branch mispredict.

Note that the FPU will not return the result of FBRA instruction until any FCPS/FCPQ instruction already underway in the coprocessor pipeline has retired. The CPU will consequently stall until such time that the msw of the FSR is available to be read (though these are fast operations, so stalls should be minimal). In effect, a CPU conditional branch instruction operation will synchronize the integer and floating-point pipelines with respect to FPU FCPS/FCPQ status.

The LS 3-bits of the branch opcode concatenated with the Sub-Opcode bit (such that the Sub-Opcode bit becomes the LSb of this value) may be used by the CPU decoder as a bit pointer into the 16-bit branch status test value to select the corresponding branch predicate result. The branch then decides if the outcome is true (taken) or false (not taken) based on the state of the selected bit (where true is when the bit is set, false when clear).

Note: FCPS/FCPQ and FTST instructions update two different portions of the FSR. Consequently, execution of an FTST instruction (which also updates the FSR) will not inhibit the CPU CBRAn instructions from using the branch status generated from the FSR Ordering Relation bits.

3.6.8.10.2. Operand Characteristic (Test) Status

The FTST instruction will test the operand and update the SUB, INF, FN, FZ, FNAN status bits. No exceptions will be generated by this instruction. Due to the relative infrequent use of this instruction, dedicated conditional branches are not supported by the CPU to test these status bits. The user must read the FSR and then act upon the bits of instruction using existing CPU instructions.

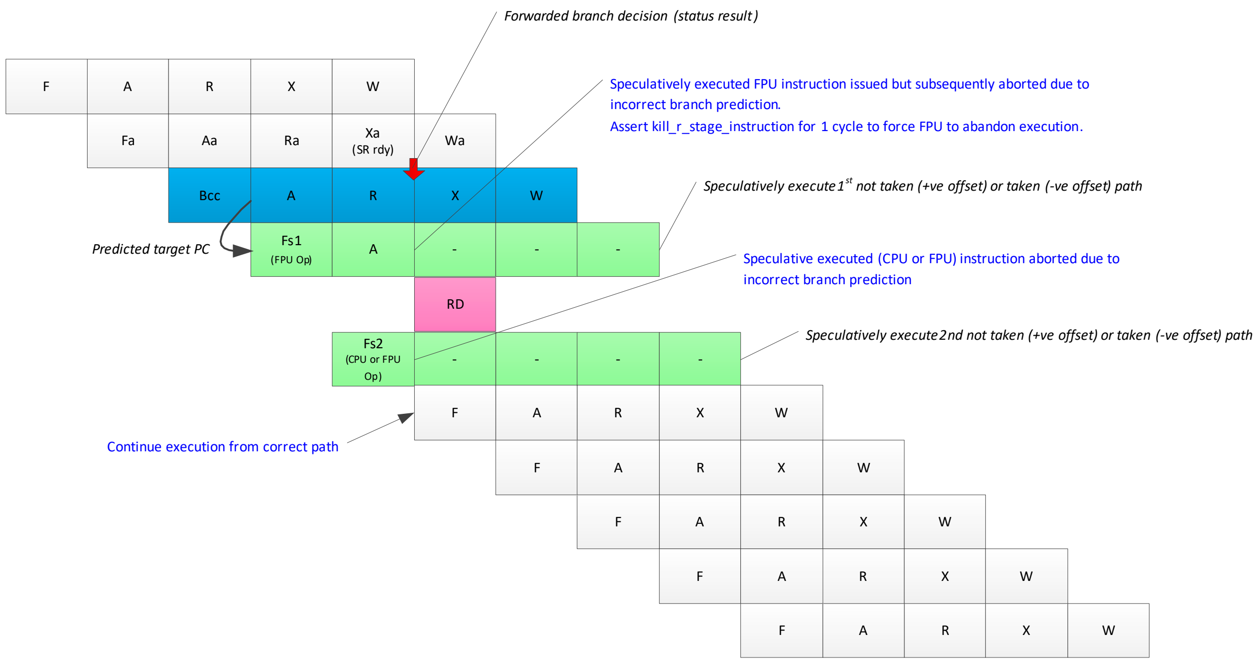
Table 3-18. Floating-Point Conditional Branches and Associated Predicates

Assembler FBRA Attribute	Design Mnemonic CBRA Mapping	Predicates					Assembler FBRA Attribute	Design Mnemonic CBRA Mapping	Negated Predicates				
		Ordering Relation				Definition (Alternative Definition)			Ordering Relation				Definition (Alternative Definition)
		>	<	=	?				>	<	=	?	
EQ	CBRA0	F	F	T	F	Equal	UNE	CBRA1	T	T	F	T	Unordered or Greater Than or Less Than (Unordered or Not Equal)
NE	CBRA2	T	T	F	F	Greater Than or Less Than (Not Equal)	UEQ	CBRA3	F	F	T	T	Unordered or Equal
GT	CBRA4	T	F	F	F	Greater Than	ULE	CBRA5	F	T	T	T	Unordered or Less Than or Equal
GE	CBRA6	T	F	T	F	Greater Than or Equal	ULT	CBRA7	F	T	F	T	Unordered or Less Than
LT	CBRA8	F	T	F	F	Less Than	UGE	CBRA9	T	F	T	T	Unordered or Greater Than or Equal
LE	CBRA10	F	T	T	F	Less Than or Equal	UGT	CBRA11	T	F	F	T	Unordered or Greater Than
OR	CBRA12	T	T	T	F	Ordered	UN	CBRA13	F	F	F	T	Unordered

3.6.8.10.3. FPU Instruction Kill

As is the case for all instructions executed within conditional branch speculative slots, floating-point instructions will be killed if a branch mispredict occurs. The CPU will recognize a mispredict prior to the end of the conditional branch R-stage (i.e., when the prior instruction status is available to forward). If the instruction in the first speculative slot is an FPU instruction, it will be issued to the FPU, but the CPU will assert a signal to kill the instruction for one cycle, forcing the FPU to subsequently abandon execution prior to it being committed. If the instruction in the second speculative slot is an FPU instruction, it will be abandoned prior to being issued to the FPU.

Figure 3-43. FPU Instruction Speculative Execution



Legend: F = Fetch, A = Address decode, R = Read, X = Execute, W = Write back

3.6.8.11. Generating FP Exceptions

The FPU can generate the five IEEE-754 2008 compliant exceptions. Each exception has a flag associated with it in the FSR register.

All instructions, except `FTST`, `FMOV` and `FMOVC`, can affect FSR exception status as a consequence of the operation (any exception status flag bits not set will be cleared). The `FTST`, `FMOV` and `FMOVC` instructions will neither set nor clear any exception flags. Refer to the IEEE-754 2008 standard for a more detailed definition.

"Most-recent" exception status only shows the exception status of the most recent instruction executed. It contains no accrued status from prior operations such that if a status bit is not affected or not signaled by the instruction, it will be cleared.

"Sticky" exception status contains accrued status from all instructions executed.

The following summarizes when exceptions are signaled, and the default results for each exception are summarized in Table 3-16. Refer to Table 3-19 for exception conditions and default results for each instruction.

- Invalid: FSR.INVAL Exception is signaled whenever an operation generates no usefully definable result. INVAL is set under the following conditions:
  - Any operation on a sNaN input
  - Addition of infinities with opposite signs or subtraction of infinities with the same sign
  - Multiplication 0*infinity
  - Division 0/0 and infinity/infinity
  - Square root of a negative floating-point value

Note: INVAL is also set by FF2DI and FF2LI Float-to-Integer conversion instructions in the event of a qNaN or Infinity input value.

- Divide by Zero: FSR.DIV0 Exception is signaled whenever the FDIV instruction dividend is finite and the divisor is 0.

- Overflow: FSR.OVF Exception is signaled whenever an operation results in an overflow, defined as a post-rounded result that exceeds the largest finite number that the destination can represent.
- Underflow: FSR.UDF Exception is signaled whenever an operation results in a tiny but non-zero (subnormal) result (see Underflow with an Exact Rounded Result for a special case).
- Inexact: FSR.INX Exception is signaled whenever the rounded result of an operation is:
  – Not equal to the same result represented using infinite precision (i.e., has suffered a loss of accuracy).
  – Is subnormal and not an exact zero when FTZ mode is enabled.

In addition, the CPU also generates two additional exceptions:

- Huge Integer: FSR.HUGI Exception is signaled whenever a Float-to-Integer conversion operation (FF2DI and FF2LI) results in an integer value that is larger than the destination register can represent.
- Subnormal Operand: FSR.SUBO Exception is signaled whenever an operand of an affected instruction (see Subnormal Override Functions and Table 3-19) is a subnormal value, and Subnormals-Are-Zeros (SAZ) mode is disabled.

3.6.8.11.1. Exception Generation Special Cases

Concurrent Exceptions

The following combinations of exceptions can occur together:

- Huge Integer will always also signal Invalid and Inexact.
- Overflow will always also signal Inexact.
- Underflow may also signal Inexact.

No other exceptions can occur concurrently.

Notes:

1. Inexact can be asserted independently of any other exception.
2. If Overflow, Underflow and Inexact exceptions are all enabled, the exception handler should prioritize Overflow and Underflow above Inexact.

During FF2DI and FF2LI Float-to-Integer conversion instructions, should the HUGI exception be signaled (due to a finite floating-point value conversion that results in a value that exceeds the size of the destination register), the INVAL exception will also be signaled. Users may, therefore, choose to ignore the (not IEEE-754 compliant) HUGI exception.

Underflow with an Exact Rounded Result

An exact subnormal result is not viewed as an Underflow condition by the IEEE 754-2008/2019 standard when operating with default exception handling. Consequently, FPU exceptions operate differently when the underflow exception is disabled (enabling default exception handling for the underflow).

As summarized below, when operating with default exception handling (the underflow exception masked, FCR.UDFM = 1), the FPU will only signal underflow (i.e., set FSR.UDF = 1) if the rounded subnormal result suffers from a loss of accuracy (such that the Inexact status will also be set). Otherwise, no status is affected. In both cases, the rounded (default) result will be delivered.

Note: If FTZ mode is active (FCR.FTZ && FCR.UDFM = 1), Inexact is signaled whenever a result is subnormal and not an exact zero.

When operating with alternate exception handling (the underflow exception is unmasked), underflow will be signaled whenever a subnormal result is detected irrespective of whether it is exact or not. If the result also suffers from any loss of accuracy, Inexact will also be set.

Note: A zero result is not considered an Underflow condition, so will not signal underflow irrespective of the Exception Handling mode.

- Default exceptions: UDF interrupt is masked (UDFM = 1).
  - If an inexact underflow occurs, both UDF and INX (and sticky equivalents) are set.
  - If an exact underflow occurs, no status is set.
  - In both cases, default (rounded) result is delivered, and no interrupts are generated.
- Alternate exceptions: UDF interrupt is unmasked (UDFM = 0).
  - If any underflow occurs, UDF (and sticky equivalent) is always set and an interrupt occurs. INX (and sticky equivalent) will also be set if it is inexact; it is cleared otherwise.
  - Default (rounded) result is delivered and interrupt is generated.

Invalid for a qNaN Input Operand

A qNaN input operand does not typically signal the invalid exception other than when encountered by the following instructions:

- FF2DI and FF2LI Float-to-Integer conversion instructions (because a qNaN cannot, of course, be represented as an integer value).
- FCPS instruction to signal any NaN input.

For both sNaN and qNaN operands, the FF2DI and FF2LI instructions will deliver the integer indefinite value (the most negative number) as the result, and INVAL (but not HUGI) will be signaled.

No result is delivered for the FCPS instruction (other than FCPS/FCPQ status).

3.6.8.11.2. FP Exception Sticky Flags

The INVAL, DIV0, OVF, UDF, INX, HUGI and SUBO exception flags have corresponding “sticky” flags INVALS, DIV0S, OVFS, UDFS, INXS, HUGIS and SUBOS also located within the FSR. These bits are set whenever the corresponding exception flag is set. They will remain set if the exception flag is cleared by a subsequent instruction; therefore, they are considered “sticky.” These bits support the delayed exception handling model required by the IEEE 754-2008 standard and are analogous to conventional interrupt flags.

3.6.8.11.3. Modifying Exception Status Through Software

Software may simultaneously clear both “most-recent” and “sticky” exception flags using the `FAND` instruction. Software may also set both “most-recent” and “sticky” exception flags simultaneously using the `FIOR` instruction, though this operation is of little utility.

Note: Clearing (or setting) a most-recent exception status bit by using the `FAND` (or `FIOR`) instruction will not affect any corresponding sticky exception bits.

3.6.8.11.4. FP Exception Masks

FSR bits INVAL, DIV0, OVF, UDF, INX, HUGI and SUBO have corresponding INVALM, DIV0M, OVFM, UDFM, INXM, HUGIM and SUBOM Exception Mask bits in the FCR. Should a “most-recent” exception flag be set by an operation, the corresponding Exception Mask bit in the FCR must already be clear in order to generate an interrupt.

Should an FCR Exception Mask bit be set when the corresponding FSR exception flag bit is set, no interrupt will be generated. In all cases (except SUBO which is an input operand exception), a default result (see Table 3-16) will be written. However, the corresponding “sticky” exception flag will be set and remain set until manually cleared.

Note: If not masked, interrupts are generated at the time the corresponding flag is set (i.e., they are generated by the leading edge of the flag set operation). However, setting a flag (in FSR [6:0]) when the corresponding exception is enabled will not generate an interrupt. Should forcing an interrupt be required (e.g., during debug), it may be achieved by setting the FPU interrupt flag (in the interrupt controller). Conversely, unmasking a previously masked exception when its flag is already set will not generate an interrupt.

Table 3-19. FP Instruction Exception Conditions

Instruction	Exceptions (FSR[6:0])							Conditions	Default Results and Notes
	SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL		
FMOV	—	—	—	—	—	—	—	—	—
FMOVC	—	—	—	—	—	—	—	—	—
Status Bit Set/Clear/Update Instructions									
FAND	⇓	⇓	⇓	⇓	⇓	⇓	⇓	Logical AND with FSR[15:0]	—
FIOR	⇑	⇑	⇑	⇑	⇑	⇑	⇑	Logical OR with FSR[15:0]	No exceptions are generated as a consequence of setting an exception flag.
FTST	—	—	—	—	—	—	—	—	No result is delivered other than FTST status (FSR[28:24]).
Conversion Instructions									
FLI2F	0	0	⇕	0	0	0	0	INX: See Generating FP Exceptions	INX: Destination will be written with inexact floating-point result.
FDI2F	0	0	⇕	0	0	0	0	INX: See Generating FP Exceptions	INX: Destination will be written with inexact floating-point result.
FF2LI	⇕	⇕	⇕	0	0	0	⇕	HUGI: Result exceeds target register size INX: See Generating FP Exceptions INVAL: ∞, sNaN or qNaN SUBO: Subnormal operand(3)	HUGI: Destination will be written with either most positive or most negative integer, matching sign of input operand. HUGI will also cause INVAL to be set. INX: Destination will be written with inexact integer result. INVAL: Destination will be written with integer indefinite value if HUGI is not set, or value defined above if HUGI is set. Assertion of SUBO will not affect integer result.
FF2DI	⇕	⇕	⇕	0	0	0	⇕	HUGI: Result exceeds target register size INX: See Generating FP Exceptions INVAL: ∞, sNaN or qNaN SUBO: Subnormal operand(3)	HUGI: Destination will be written with either most positive or most negative integer, matching sign of input operand. HUGI will also cause INVAL to be set. **Note:** INX: Destination will be written with inexact integer result. INVAL: Destination will be written with integer indefinite value if HUGI is not set, or value defined above if HUGI is set. Assertion of SUBO will not affect integer result.
Comparison Instructions									
FCPS	⇕	0	0	0	0	0	⇕	INVAL: sNaN or qNaN SUBO: Subnormal operand(3)	No result is delivered other than FCPS/FCPQ status (FSR[19:16]).
FCPQ	⇕	0	0	0	0	0	⇕	INVAL: sNaN SUBO: Subnormal operand(3)	No result is delivered other than FCPS/FCPQ status (GT, LT, EQ or UN).
FFLIM	⇕	0	0	0	0	0	⇕	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-13 for the result delivered. Operand value of -0 compares to less than +0.

Key: ⇕ = set or cleared, ‘0’ = always cleared, — = unchanged, ⇑ = may be cleared but never set, ⇓ = may be set but never cleared

Notes:

1. In all cases where INVAL is signaled as the result of an sNaN operand, the destination will be written with the quieted sNaN.
2. Underflow result is defined in Table 3-16 when SOV mode is enabled and the underflow exception is not enabled.
3. The Subnormal Operand exception (SUBO) is an input operand exception (all other exceptions are related to operation results). SUBO is never signaled if SAZ mode is enabled.

Table 3-19. FP Instruction Exception Conditions (continued)

Instruction	Exceptions (FSR[6:0])							Conditions	Default Results and Notes
	SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL		
FMAX	⇕	0	0	0	0	0	⇕	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-12 for the result delivered. Operand value of -0 compares to less than +0.
FMAXNUM	⇕	0	0	0	0	0	⇕	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-12 for the result delivered. Operand value of -0 compares to less than +0.
FMIN	⇕	0	0	0	0	0	⇕	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-12 for the result delivered. Operand value of -0 compares to less than +0.
FMINNUM	⇕	0	0	0	0	0	⇕	INVAL: sNaN SUBO: Subnormal operand(3)	Refer to Table 3-12 for the result delivered. Operand value of -0 compares to less than +0.
Math Instructions									
FADD	⇕	0	⇕	⇕	⇕	0	⇕	INX: See Generating FP Exceptions UDF: See Generating FP Exceptions OVF: See Generating FP Exceptions INVAL: ((-∞) + ∞) or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with a rounded result. (2) **Note:** INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1)
FSUB	⇕	0	⇕	⇕	⇕	0	⇕	INX: See Generating FP Exceptions UDF: See Generating FP Exceptions OVF: See Generating FP Exceptions INVAL: (∞ - ∞) or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with a rounded result. (2) INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.
FNEG	⇕	—	—	—	—	—	—	SUBO: Subnormal operand(3)	IEEE-754 requires no result exceptions be signaled. Subnormal operand exception is signaled or SAZ mode applied to ensure result consistency with that of an equivalent sequence of FPU arithmetic instructions.
FABS	⇕	—	—	—	—	—	—	SUBO: Subnormal operand(3)	IEEE-754 requires no result exceptions be signaled. Subnormal operand exception is signaled (or SAZ mode applied) to ensure result consistency with that of an equivalent sequence of FPU arithmetic instructions.
FMUL	⇕	0	⇕	⇕	⇕	0	⇕	INX: See Generating FP Exceptions UDF: See Generating FP Exceptions OVF: See Generating FP Exceptions INVAL: (0 * ∞) or (∞ * 0) or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with a rounded result. (2) INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.

Key: ⇕ = set or cleared, '0' = always cleared, — = unchanged, ⇑ = may be cleared but never set, ⇓ = may be set but never cleared
Notes:
1. In all cases where INVAL is signaled as the result of an sNaN operand, the destination will be written with the quieted sNaN.
2. Underflow result is defined in Table 3-16 when SOV mode is enabled and the underflow exception is not enabled.
3. The Subnormal Operand exception (SUBO) is an input operand exception (all other exceptions are related to operation results). SUBO is never signaled if SAZ mode is enabled.

Table 3-19. FP Instruction Exception Conditions (continued)

Instruction	Exceptions (FSR[6:0])							Conditions	Default Results and Notes
	SUBO	HUGI	INX	UDF	OVF	DIV0	INVAL		
FMAC	⇕	0	⇕	⇕	⇕	0	⇕	INX: See Generating FP Exceptions UDF: See Generating FP Exceptions OVF: See Generating FP Exceptions INVAL: (0 * ∞)+c or (∞ * 0)+c or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and result is not an exact subnormal. INX/UDF/OVF: Destination will be written with a rounded result. (2) INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1) INVAL will also be signaled if accumulation is a subtraction of infinities.
FDIV	⇕	0	⇕	⇕	⇕	⇕	⇕	INX: See Generating FP Exceptions UDF: See Generating FP Exceptions OVF: See Generating FP Exceptions DIV0: Finite Dividend with Divisor = 0 INVAL: (0/0) or (∞/∞) or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if OVF is set. Will also be set if UDF is set and the result is not an exact subnormal. INX/UDF/OVF: Destination will be written with a rounded result. (2) DIV0: Result of (±x/±0) will be ±∞ by default, where sign is XOR of operand signs. INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1) (0/0) or (∞/0) are special cases and will not set DIV0. Result of (∞/0) is correctly signed infinity. Result of (0/0) is the distinguished qNaN with INVAL signaled.
FSQRT	⇕	0	⇕	0	0	0	⇕	INVAL: x < 0 or sNaN SUBO: Subnormal operand(3)	INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1) FSQRT(±0) = ±0 (no exception signaled)
FSIN	⇕	0	⇕	⇕	0	0	⇕	INX:See Generating FP Exceptions UDF:See Generating FP Exceptions INVAL: | x | = ∞ or sNaN SUBO: Subnormal operand(3)	INX: Will also be set if UDF is set and the result is not an exact subnormal. INX/UDF: Destination will be written with a rounded result. (2) INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1)
FCOS	⇕	0	⇕	⇕	0	0	⇕	INVAL: | x | = ∞ or sNaN SUBO: Subnormal operand(3)	INVAL: Destination will be written with the distinguished qNaN or quieted sNaN.(1)

Key: ⇕ = set or cleared, ‘0’ = always cleared, — = unchanged, ⇑ = may be cleared but never set, ⇓ = may be set but never cleared
Notes:
1. In all cases where INVAL is signaled as the result of an sNaN operand, the destination will be written with the quieted sNaN.
2. Underflow result is defined in Table 3-16 when SOV mode is enabled and the underflow exception is not enabled.
3. The Subnormal Operand exception (SUBO) is an input operand exception (all other exceptions are related to operation results). SUBO is never signaled if SAZ mode is enabled.

3.6.8.12. FP Instruction Status Effects

Table 3-20 lists the floating-point instructions and their effect on the FPU Status bits.

- FSR [19:16] can only be updated by the `FCPS`/`FCPQ` instructions.
- FSR [28:24] can only be updated by the `FTST` instruction.
- `FMOV`, `FMOVC`, `FABS` and `FNEG` instructions do not generate any status.

3.6.8.12.1. FP Status Access

The msws and lsws of the FSR are implemented as 16-bit registers which can be independently read and written by the CPU and FPU. This is intended to prevent structural hazards arising between

FTST/FCPS/FCPQ/FBcc instructions (that access the FSR msw) and all other instructions that could affect the exception status (located in the FSR lsw). Hazards are not expected to exist between FTST and FCPS/FCPQ instructions.

The FCPS/FCPQ/FTST status located in msw of the FSR (FSRH) can be read and stacked by the PUSHCR instruction while the lsw of the FSR is written (or blocked to be written). The msws and lsws of the FSR are otherwise concatenated into a single 32-bit value (FSR) for read/write by the MOVCRW/MOVWCR and PUSHCR/POPCR instructions.

In addition, the FAND and FIOR instructions operate with a 16-bit literal and can be used to set or clear the bit in the FCR or lsw of the FSR or FEAR. When the FSR is referenced in these ops, users can manipulate the FPU exception status, but access to the FCPS/FCPQ/FTST status in FSRH is not possible. This choice assumes that the need to modify FTST/FCPS/FCPQ status is rare. Consequently, doing so using MOVCRW/MOVWCR or PUSHCR/POPCR is expected to be adequate.

Table 3-20. FP Instruction Status Operations

Instruction	FPU Status Register (FSR)																						
	FSR[28:24]					FSR[19:16]				FSR[14:8]							FSR[6:0]						
	SUB	INF	FN	FZ	FNAN	GT	LT	EQ	UN	SUBOS	HUGIS	INXS	UDFS	OVFS	DIVOS	INVALS							
Move Instructions																							
FMOV	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
FMOVC	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Status Bit Set/Clear/Update Instructions																							
FAND (1)	—	—	—	—	—	—	—	—	—	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓
FIOR (1)	—	—	—	—	—	—	—	—	—	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑
FTST	⇕	⇕	⇕	⇕	⇕	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Conversion Instructions																							
FLI2F	—	—	—	—	—	—	—	—	—	—	—	⇑(2)	—	—	—	—	0	0	⇕(2)	0	0	0	0
FDI2F	—	—	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	0	0	⇕	0	0	0	0
FF2LI	—	—	—	—	—	—	—	—	—	⇑	⇑	⇑	—	—	—	⇑	⇕	⇕	⇕	0	0	0	⇕
FF2DI	—	—	—	—	—	—	—	—	—	⇑	⇑	⇑	—	—	—	⇑	⇕	⇕	⇕	0	0	0	⇕
Comparison Instructions																							
FCPS	—	—	—	—	—	⇕	⇕	⇕	⇕	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FCPQ	—	—	—	—	—	⇕	⇕	⇕	⇕	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FFLIM	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FMAX	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FMAXNUM	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FMIN	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
FMINNUM	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	⇑	⇕	0	0	0	0	0	⇕
Math Instructions																							
FADD	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	—	⇑	⇕	0	⇕	⇕	⇕	0	⇕
FSUB	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	—	⇑	⇕	0	⇕	⇕	⇕	0	⇕
FNEG	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	—	⇕	—	—	—	—	—	—
FABS	—	—	—	—	—	—	—	—	—	⇑	—	—	—	—	—	—	⇕	—	—	—	—	—	—
FMUL	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	—	⇑	⇕	0	⇕	⇕	⇕	0	⇕
FMAC	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	—	⇑	⇕	0	⇕	⇕	⇕	0	⇕
FDIV	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	⇑	⇑	⇑	⇕	0	⇕	⇕	⇕	⇕	⇕
FSQRT	—	—	—	—	—	—	—	—	—	⇑	—	⇑	—	—	—	⇑	⇕	0	⇕	0	0	0	⇕
FSIN	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	—	—	⇑	⇕	0	⇕	⇕	0	0	⇕
FCOS	—	—	—	—	—	—	—	—	—	⇑	—	⇑	⇑	—	—	⇑	⇕	0	⇕	⇕	0	0	⇕

Key: ⇕ set or cleared, ‘0’ always cleared, — unchanged, ⇓ may be cleared but never set, ⇑ may be set but never cleared

Notes:

1. With respect to the FSR, FAND and FIOR can only affect FSR[15:0] (exception status). When set, no exception is signaled.
2. FLI2F.s and FLI2F.d only. Inexact not possible for Long integer to Double Precision float (LI2F.d) conversion.

4. Memory Organization

This section describes the Memory Organization and Bus Matrix (BMX) in dsPIC33A devices. The following features are covered.

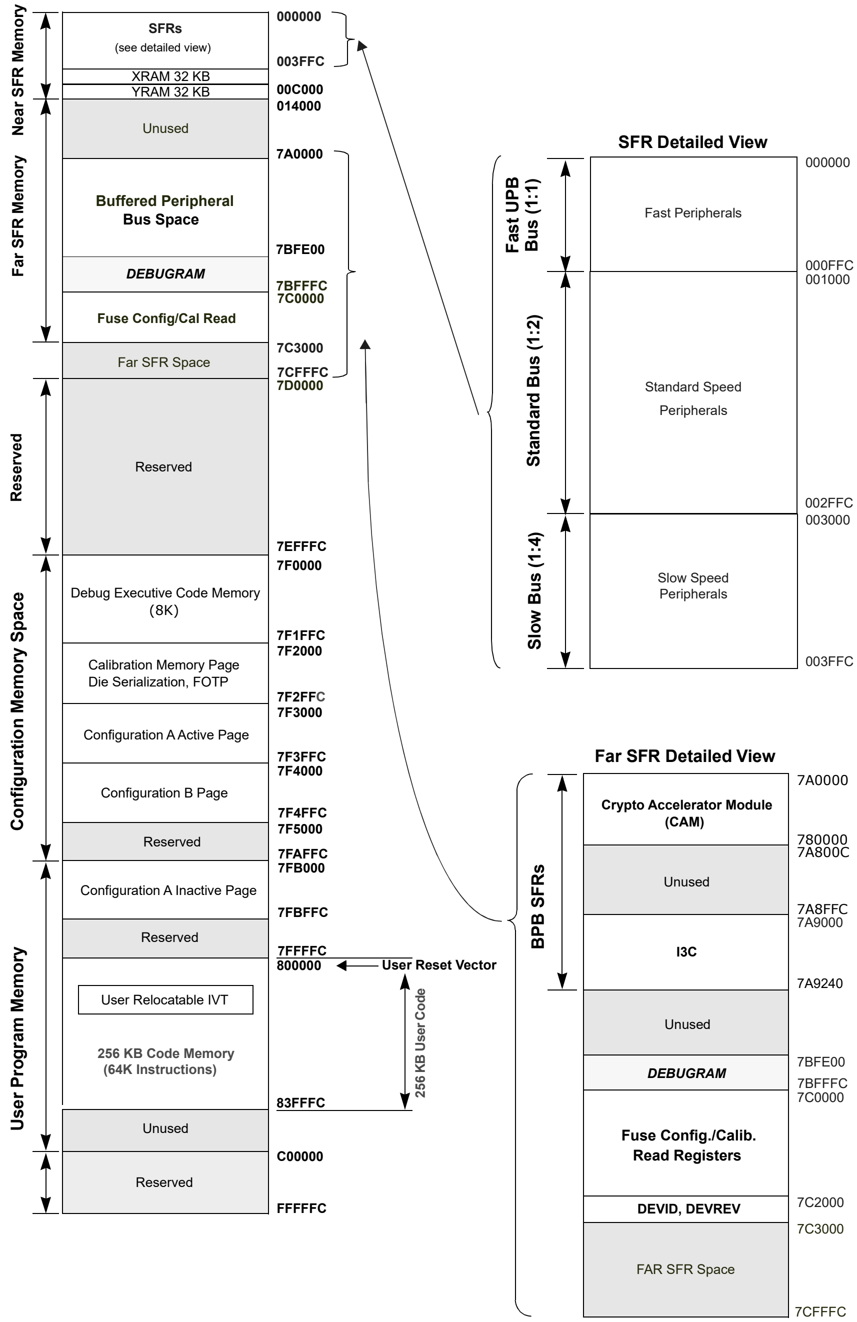
- Memory and SFR Maps
- Modified Harvard Architecture
- Unified Memory Map
- Split Data Bus Speeds
- Bus Matrix:
  - Establishes communication between initiator and targets
  - Decodes addresses and provides arbitration between multiple initiators requesting access to the same target
  - Provides concurrent accesses to multiple targets from different initiators
  - Generates a bus error exception back to an initiator on any failed access
  - Provides support for the CPU to execute from RAM

4.1. Device-Specific Information

4.1.1. Address Space

The program memory maps for dsPIC33AK256MPS306 devices are shown in Figure 4-1.

Figure 4-1. Memory Map for



dsPIC33AK256MPS306

Notes:

1. Memory areas are not shown to scale.
2. Calibration data area includes UDID and ICSP™ Write Inhibit locations.

4.1.1.1. Unique Device Identifier (UDID)

All dsPIC33AK256MPS306 family devices are individually encoded during final manufacturing with a Unique Device Identifier (UDID). The UDID cannot be erased by a bulk erase command or any other user-accessible means. This feature allows for manufacturing traceability of Microchip devices in applications where this is a requirement. It may also be used by the application manufacturer for:

- Tracking the device
- Unique identifying number
- Unique security key
  The UDID is comprised of four full 32-bit words. These 4x32 bit word fields form a unique 128-bit identifier. The UDID is stored in read-only memory locations between 0x7F2120-0x7F212C in the device configuration space. Table 4-1 lists the addresses of the identifier words and shows their contents.

Table 4-1. UDID Address

Address	Description
0x7F2120	UDID Word 1
0x7F2124	UDID Word 2
0x7F2128	UDID Word 3
0x7F212C	UDID Word 4

4.2. Architectural Overview

4.2.1. Program Memory Organization

Program memory addresses are always word-aligned on the lower word, and addresses are incremented or decremented by four during code execution. This arrangement provides compatibility with data memory space addressing and makes data in the program memory space accessible.

4.2.2. Interrupt and Trap Vectors

All dsPIC33AK256MPS306 family devices have the user program space starting at 0x800000. The default interrupt vector table is placed at the start of user program Flash memory (0x800004). The user code can then relocate the IVT to any valid address in Flash/RAM by modifying the IVTBASE register. The Reset vector into user program space is at 0x800000. When a Reset is asserted, the Reset vector read commences and the vector contents are presented to the CPU. The data are then immediately transferred to the program space address bus to create the address of the first instruction to be executed, redirecting the program execution to the appropriate start-up routine.

4.2.3. Bus Matrix

The Bus Matrix (BMX) arbitrates memory accesses in the event that two independent initiators are trying to access the same targets. Initiators are a set of modules that can initiate a read or write transaction to other modules called “Targets.” The BMX connects the initiator to targets and determines which initiator gets priority (based on a fixed priority scheme and the SFR priority). The type of access, program or data, is determined by the initiator bus. It supports concurrent accesses by different initiators as long as they have independent targets. For example, CPU to SFR could be concurrent with DMA to RAM.

Bus Matrix Initiators include:

- Initiator 0 – CPU X Data Bus (CPU XDS)
- Initiator 1 – CPU Y Data Bus (CPU YDS)
- Initiator 2 – DMA
- Initiator 3 – CPU Instruction Bus (CPU IS)

- Initiator 4 – Crypto
- Initiator 5 – CAN 1
- Initiator 6 – Nonvolatile Memory Controller
- Initiator 7 – In-Circuit Debugger

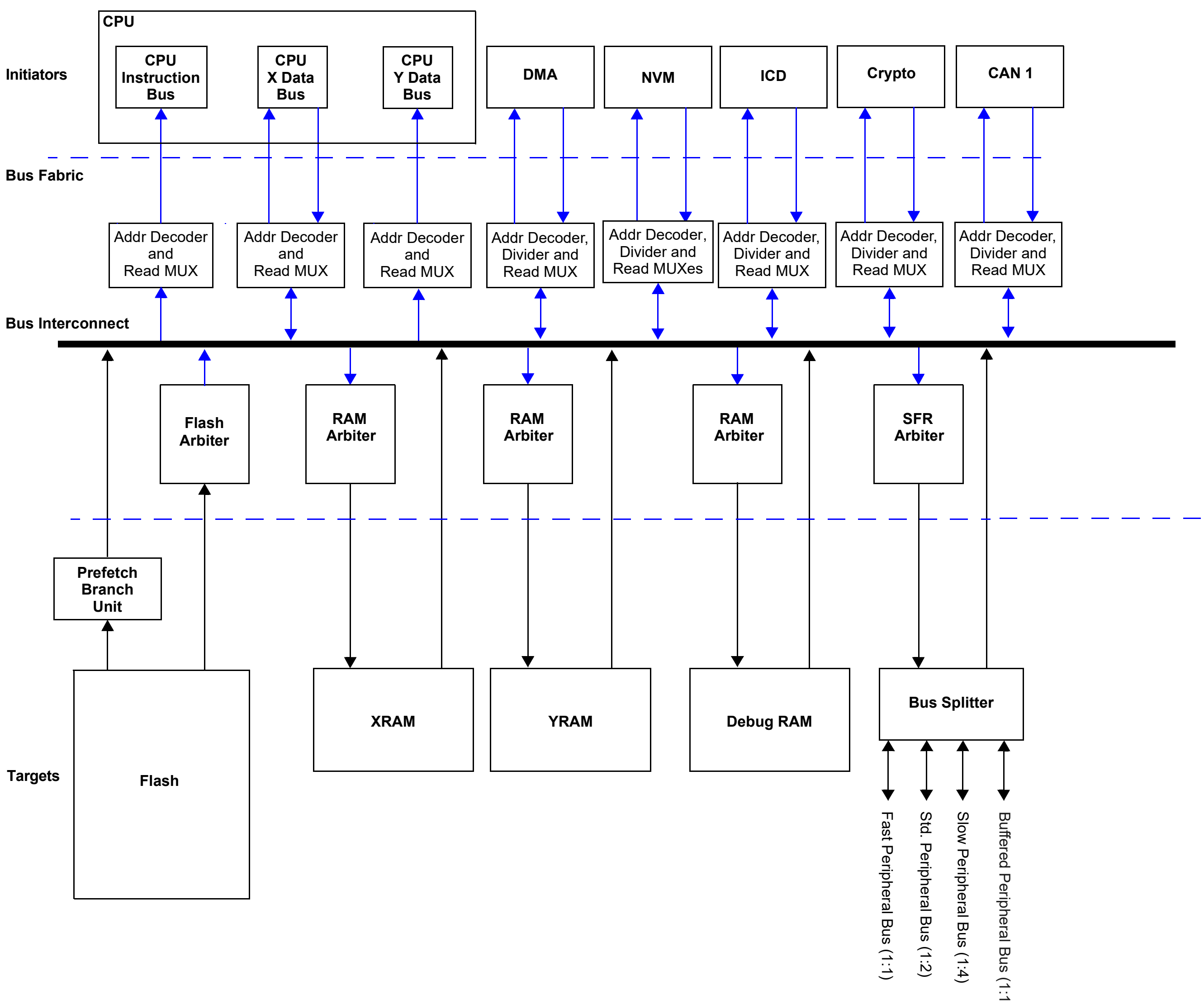
Bus Matrix Targets include:

- Program Flash (data reads)
- Peripheral Buses (through Bus Splitter)
- XRAM interface
- YRAM interface
- Debug RAM
- Crypto Space RAM

Note: XRAM and YRAM simultaneous access is supported.

Figure 4-2 shows a typical block diagram of the Bus Matrix.

Figure 4-2. Bus Matrix Initiators and Targets



Note: See Table 4-4 for initiator/target options.

4.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0770	BMXINITPR	31:24								
		23:16								
		15:8								
		7:0					NVMPR	CAN1PR	CRYPTPR	DMAPR
0x0774	BMXIRAML	31:24	BMXIRAML[31:24]							
		23:16	BMXIRAML[23:16]							
		15:8	BMXIRAML[15:8]							
		7:0	BMXIRAML[7:2]							
0x0778	BMXIRAMH	31:24	BMXIRAMH[31:24]							
		23:16	BMXIRAMH[23:16]							
		15:8	BMXIRAMH[15:8]							
		7:0	BMXIRAMH[7:2]							
0x077C	BMXXDATERR	31:24				CRYPTWERR				
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8				CRYPTRERR	YRAMRERR	XRAMRERR		PGSPCRERR
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0780	BMXYDATERR	31:24								
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8					YRAMRERR	XRAMRERR		
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0784	BMXDMAERR	31:24				CRYPTWERR				
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8				CRYPTRERR	YRAMRERR	XRAMRERR		PGSPCRERR
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0788	BMXCPUERR	31:24								
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8					YRAMRERR	XRAMRERR		
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x078C	BMXCRYPTERR	31:24				CRYPTWERR				
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8				CRYPTRERR	YRAMRERR	XRAMRERR		PGSPCRERR
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0790	BMXCAN1ERR	31:24				CRYPTWERR				
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8				CRYPTRERR	YRAMRERR	XRAMRERR		PGSPCRERR
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0794	BMXICDERR	31:24			DBGWERR	CRYPTWERR				
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8			DBGRERR	CRYPTRERR	YRAMRERR	XRAMRERR		
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR
0x0798	BMXNVMERR	31:24				CRYPTWERR				
		23:16						IRAMWERR	ADDWERR	BADTGTWERR
		15:8				CRYPTRERR	YRAMRERR	XRAMRERR	SFRRERR	PGSPCRERR
		7:0						IRAMRDERR	ADDRERR	BADTGTRERR

4.3.1. Bus Initiator Priority Control Register

Name: BMXINITPR
Offset: 0x770

Note:
1. CPU has the highest priority.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					NVMPR	CAN1PR	CRYPTPR	DMAPR
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – NVMPR NVM Priority Override bit

Value	Description
1	Raise NVM initiator RAM access priority above CPU.
0	No change to NVM initiator RAM access priority.

Bit 2 – CAN1PR CAN1 Priority Override bit

Value	Description
1	Raise CAN 1 initiator priority above CPU.
0	No change to CAN 1 initiator priority.

Bit 1 – CRYPTPR Crypto Priority Override bit

Value	Description
1	Raise Crypto Accelerator priority above CPU.
0	No change to Crypto Accelerator priority.

Bit 0 – DMAPR DMA Priority Override bit

Value	Description
1	Raise DMA initiator RAM access above CPU.
0	No change to DMA initiator RAM access priority.

4.3.2. BMX Instruction RAM Low Address Register

Name: BMXIRAML
Offset: 0x774

Bit	31	30	29	28	27	26	25	24
	BMXIRAML[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BMXIRAML[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BMXIRAML[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BMXIRAML[7:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bits 31:2 – BMXIRAML[31:2] Lower Boundary Address for Instruction RAM bits
Defines the lower boundary address (inclusive) for instruction RAM.

4.3.3. BMX Instruction RAM High Address Register

Name: BMXIRAMH
Offset: 0x778

Bit	31	30	29	28	27	26	25	24
	BMXIRAMH[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BMXIRAMH[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BMXIRAMH[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BMXIRAMH[7:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bits 31:2 – BMXIRAMH[31:2] Upper Boundary Address for Instruction RAM bits
Defines the upper boundary address (non-inclusive) for instruction RAM.

4.3.4. BMX Error Status Register for X Data Bus Initiator

Name: BMXXDATERR
Offset: 0x77C

Bit	31	30	29	28	27	26	25	24
				CRYPTWERR				
Access				R/HS/C				
Reset				0				

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
				CRYPTRERR	YRAMRERR	XRAMRERR		PGSPCRERR
Access				R/HS/C	R/HS/C	R/HS/C		R/HS/C
Reset				0	0	0		0

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 28 – CRYPTWERR Crypto Write Error bit

Value	Description
1	Bus error generated by Crypto space write operation.
0	No error on Crypto space write operation.

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space.
0	No IRAM write address errors.

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space.
0	No invalid target write error.

Bit 12 – CRYPTRERR Crypto Read Error bit

Value	Description
1	Bus error generated by Crypto space read operation.
0	No error on Crypto space read operation.

Bit 11 – YRAMRERR Target y Bus Read Error Flag bit

Value	Description
1	Bus error generated by YRAM read operation .
0	No error on YRAM read operation.

Bit 10 – XRAMRERR Target x Bus Read Error Flag bit

Value	Description
1	Bus error generated by XRAM read operation .
0	No error on XRAM read operation.

Bit 8 – PGSPCRERR Program Space Read Error Flag bit

Value	Description
1	Bus error generated by program space read operation.
0	No error on program space read operation.

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of. IRAM space
0	No IRAM read address errors.

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space.
0	No invalid target error.

4.3.5. BMX Error Status Register for Y Data Bus Initiator

Name: BMXYDATERR
Offset: 0x780

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
					YRAMRERR	XRAMRERR		
Access					R/HS/C	R/HS/C		
Reset					0	0		

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by an invalid instruction write outside of IRAM space.
0	No IRAM write address errors.

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to an invalid address space.
0	No unimplemented address write error.

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space.
0	No invalid target write error.

Bit 11 – YRAMRERR YRAM Read Error Flag bit

Value	Description
1	Bus error generated by YRAM read operation.
0	No error on YRAM read operation.

Bit 10 – XRAMRERR XRAM Read Error Flag bit

Value	Description
1	Bus error generated by XRAM read operation.
0	No error on XRAM read operation.

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space.
0	No IRAM read address errors.

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space.
0	No invalid target error.

4.3.6. BMX Error Status Register for DMA Initiator

Name: BMXDMAERR
Offset: 0x784

Bit	31	30	29	28	27	26	25	24
				CRYPTWERR				
Access				R/HS/C				
Reset				0				

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
				CRYPTRERR	YRAMRERR	XRAMRERR		PGSPCRERR
Access				R/HS/C	R/HS/C	R/HS/C		R/HS/C
Reset				0	0	0		0

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 28 – CRYPTWERR Crypto Write Error bit

Value	Description
1	Bus error generated by Crypto space write operation..
0	No error on Crypto space write operation.

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space.
0	No IRAM write address errors.

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space.
0	No invalid target write error.

Bit 12 – CRYPTRERR Crypto Read Error bit

Value	Description
1	Bus error generated by Crypto space read operation.
0	No error on Crypto space read operation.

Bit 11 – YRAMRERR YRAM Read Error Flag bit

Value	Description
1	Bus error generated by YRAM read operation.
0	No error on YRAM read operation.

Bit 10 – XRAMRERR XRAM Read Error Flag bit

Value	Description
1	Bus error generated by XRAM read operation.
0	No error on XRAM read operation.

Bit 8 – PGSPCRERR Program Space Read Error Flag bit

Value	Description
1	Bus error generated by program space read operation.
0	No error on program space read operation.

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space.
0	No IRAM read address errors.

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space.
0	No invalid target error.

4.3.7. BMX Error Status Register for CPU Initiator

Name: BMXCPUERR
Offset: 0x788

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
					YRAMRERR	XRAMRERR		
Access					R/HS/C	R/HS/C		
Reset					0	0		

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space.
0	No IRAM write address errors.

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space.
0	No invalid target write error.

Bit 11 – YRAMRERR YRAM Read Error Flag bit

Value	Description
1	Bus error generated by YRAM read operation.
0	No error on YRAM read operation.

Bit 10 – XRAMRERR XRAM Read Error Flag bit

Value	Description
1	Bus error generated by XRAM read operation.
0	No error on XRAM read operation.

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space.
0	No IRAM read address errors.

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space.
0	No invalid target error.

4.3.8. BMX Error Status Register for Crypto Module Initiator

Name: BMXCRYPTERR
Offset: 0x78C

Note:
1. See Table 4-3 for target bus error indices.

Bit	31	30	29	28	27	26	25	24
				CRYPTWERR				
Access				R/HS/C				
Reset				0				

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
				CRYPTRERR	YRAMRERR	XRAMRERR		PGSPCRERR
Access				R/HS/C	R/HS/C	R/HS/C		R/HS/C
Reset				0	0	0		0

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 28 – CRYPTWERR Crypto Write Error bit

Value	Description
1	Bus error generated by Crypto space write operation.
0	No error on Crypto space write operation.

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space.
0	No IRAM write address errors.

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space.
0	No invalid target write error.

Bit 12 – CRYPTRERR Crypto Read Error bit

Value	Description
1	Bus error generated by Crypto space read operation.
0	No error on Crypto space read operation.

Bit 11 – YRAMRERR YRAM Read Error Flag bit(1)

Value	Description
1	Bus error generated by YRAM read operation.
0	No error on YRAM read operation.

Bit 10 – XRAMRERR XRAM Read Error Flag bit(1)

Value	Description
1	Bus error generated by XRAM read operation.
0	No error on XRAM read operation.

Bit 8 – PGSPCRERR Program Space Read Error Flag bit

Value	Description
1	Bus error generated by program space read operation.
0	No error on program space read operation.

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space.
0	No IRAM read address errors.

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space.
0	No invalid target error.

4.3.9. BMX Error Status Register for CAN 1 Initiator

Name: BMXCAN1ERR
Offset: 0x790

Note:
1. See Table 4-3 for target bus error indices.

Legend: HS - Hardware Set, C - Clear Only

Bit	31	30	29	28	27	26	25	24
				CRYPTWERR				
Access				R/HS/C				
Reset				0				

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
				CRYPTRERR	YRAMRERR	XRAMRERR		PGSPCRERR
Access				R/HS/C	R/HS/C	R/HS/C		R/HS/C
Reset				0	0	0		0

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 28 – CRYPTWERR Crypto Write Error bit

Value	Description
1	Bus error generated by Crypto space write operation.
0	No error on Crypto space write operation.

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space.
0	No IRAM write address errors.

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space.
0	No invalid target write error.

Bit 12 – CRYPTRERR Crypto Read Error bit

Value	Description
1	Bus error generated by Crypto Accelerator read operation.
0	No error on Crypto Accelerator read operation.

Bit 11 – YRAMRERR YRAM Read Error Flag bit(1)

Value	Description
1	Bus error generated by YRAM read operation.
0	No error on YRAM read operation.

Bit 10 – XRAMRERR XRAM Read Error Flag bit(1)

Value	Description
1	Bus error generated by XRAM read operation.
0	No error on XRAM read operation.

Bit 8 – PGSPCRERR Program Space Read Error Flag bit

Value	Description
1	Bus error generated by program space read operation.
0	No error on program space read operation.

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space.
0	No IRAM read address errors.

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space.
0	No invalid target error.

4.3.10. BMX Error Status Register for Program Space Initiator

Name: BMXNVMERR
Offset: 0x798

Bit	31	30	29	28	27	26	25	24
				CRYPTWERR				
Access				R/HS/C				
Reset				0				

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
				CRYPTRERR	YRAMRERR	XRAMRERR	SFRRERR	PGSPCRERR
Access				R/HS/C	R/HS/C	R/HS/C	R/HS/C	R/HS/C
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 28 – CRYPTWERR Crypto Write Error bit

Value	Description
1	Bus error generated by a crypto write operation.
0	No error generated by a crypto operation.

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space.
0	No IRAM write address errors.

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space.
0	No invalid target write error.

Bit 12 – CRYPTRERR Crypto Read Error bit

Value	Description
1	Bus error generated by Crypto space read operation.
0	No error on Crypto space read operation.

Bit 11 – YRAMRERR YRAM Read Error Flag bit

Value	Description
1	Bus error generated by YRAM read operation.
0	No error on YRAM read operation.

Bit 10 – XRAMRERR XRAM Read Error Flag bit

Value	Description
1	Bus error generated by XRAM read operation.
0	No error on XRAM read operation.

Bit 9 – SFRRERR SFR Read Error bit

Value	Description
1	Bus error generated by SFR read operation.
0	No error on SFR read operation.

Bit 8 – PGSPCRERR Program Space Read Error Flag bit

Value	Description
1	Bus error generated by program space read operation.
0	No error on program space read operation.

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space.
0	No IRAM read address errors.

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space.
0	No invalid target error.

4.3.11. BMX Error Status Register for Debug Initiator

Name: BMXICDERR
Offset: 0x794

Bit	31	30	29	28	27	26	25	24
			DBGWERR	CRYPTWERR				
Access			R/HS/C	R/HS/C				
Reset			0	0				

Bit	23	22	21	20	19	18	17	16
						IRAMWERR	ADDWERR	BADTGTWERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
			DBGRERR	CRYPTRERR	YRAMRERR	XRAMRERR		
Access			R/HS/C	R/HS/C	R/HS/C	R/HS/C		
Reset			0	0	0	0		

Bit	7	6	5	4	3	2	1	0
						IRAMRDERR	ADDRERR	BADTGTRERR
Access						R/HS/C	R/HS/C	R/HS/C
Reset						0	0	0

Bit 29 – DBGWERR Debug RAM Write Error bit

Value	Description
1	Bus error generated by debug RAM write operation.
0	No error on debug RAM write operation.

Bit 28 – CRYPTWERR Crypto Write Error bit

Value	Description
1	Bus error generated by Crypto space write operation.
0	No error on Crypto space write operation.

Bit 18 – IRAMWERR IRAM Write Error Flag bit

Value	Description
1	Error generated by invalid instruction write outside of IRAM space.
0	No IRAM write address errors.

Bit 17 – ADDWERR Invalid Address Write Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 16 – BADTGTWERR Invalid Target Write Error Flag bit

Value	Description
1	Error generated by write to disallowed target space.
0	No invalid target write error.

Bit 13 – DBGRERR Debug RAM Read Error bit

Value	Description
1	Bus error generated by debug RAM read operation.
0	No error on debug RAM read operation.

Bit 12 – CRYPTRERR Crypto Read Error bit

Value	Description
1	Bus error generated by Crypto space read operation.
0	No error on Crypto space read operation.

Bit 11 – YRAMRERR YRAM Read Error Flag bit

Value	Description
1	Bus error generated by YRAM read operation.
0	No error on YRAM read operation.

Bit 10 – XRAMRERR XRAM Read Error Flag bit

Value	Description
1	Bus error generated by XRAM read operation.
0	No error on XRAM read operation.

Bit 2 – IRAMRDERR IRAM Read Error Flag bit

Value	Description
1	Error generated by invalid instruction read outside of IRAM space.
0	No IRAM read address errors.

Bit 1 – ADDRERR Invalid Address Error Flag bit

Value	Description
1	Error generated by read or write to invalid address space.
0	No unimplemented address write error.

Bit 0 – BADTGTRERR Invalid Target Read Error Flag bit

Value	Description
1	Error generated by read to disallowed target space.
0	No invalid target error.

4.4. BMX Operation

The purpose of the BMX is to decode addresses, provide arbitration between multiple initiators requesting access to the same target and to provide concurrent access to multiple targets from different initiators.

4.4.1. Arbitration

The BMX supports a decentralized fixed priority arbitration scheme. Each target has an independent arbitrator that will grant read and write requests to an initiator when that target is available. The default priority of an initiator is fixed and determined by the initiator's index on the bus. The priorities are shown in Table 4-2.

Table 4-2. Initiator Priority

Priority	Initiator Index	Type
Highest	0	CPU X Data Bus (CPU XDS)
	1	CPU Y Data Bus (CPU YDS)

Table 4-2. Initiator Priority (continued)

Priority	Initiator Index	Type
	2	DMA
	3	CPU Instruction Bus (CPU IS)
	4	Crypto
	5	CAN1
	6	NVM
Lowest	7	In-circuit Debugger

4.4.1.1. Priority Overrides

The BMXINITPR register can be used to temporarily override the default priority scheme when accessing RAM targets. When the corresponding bit is set, an initiator will have its priority raised above the native priorities of the CPU and other initiators. The ICD bus master does not support priority overrides; it will always be the lowest priority and does not have an associated BMXINITPR bit.

If multiple override bits are set, priority between the overridden initiators is determined by their natural priority order. For initiators that do not win arbitration, the BMX will stall the initiator of the lower priority transaction until a subsequent cycle, after the target access completes, when the initiator does win arbitration. Losing initiators will be forced to stall until no higher priority initiators are requesting the target.

4.4.2. Concurrency

As the BMX has an independent address decode for each initiator and independent arbiters for each target, it supports concurrent accesses. As long as each initiator is accessing a unique target, all reads and writes can proceed simultaneously.

4.4.3. Write Buffers

The targets can operate on slower clock sources than the initiators, and the BMX requires that targets have the ability to buffer writes. The target is then responsible for completing the write. Posting the write improves performance since the write completes sooner from the perspective of the CPU.

Posting writes improves single write performance only. Back-to-back writes can cause stalls to initiators if the clocking for the target is not set to a 1:1 ratio. Write buffers are only one deep, so repeated writes will be held by the BMX until the buffer is empty and the previous write completes. All transactions to a target complete in order, and, therefore, reads are never allowed to pass writes.

4.4.4. Debug RAM

The BMX supports a RAM target for use in Debug mode only. The debug RAM space is at a fixed location and is from 0x7BFE00 to 0x7BFFFF.

The ICD initiator can always access debug RAM. For other initiators, both read and write access to debug scratch pad RAM are disallowed unless the device is in Debug mode. Access to the debug RAM when outside of Debug mode will generate a bus error and set the ADDRERR (BMXxERR[1]) bit. Additionally, debug RAM is not a valid target for the CPU instruction bus regardless of the Debug state and will cause a bus error.

Note that because the CPU does not handle traps while in Debug mode, in some circumstances, bus errors caused in Debug mode may result in a ‘soft lock’ situation that will require a device Reset to resolve.

4.4.5. Bus Error

The Bus Matrix generates a bus error exception back to an initiator on any failed access. Failed accesses can occur for a number of reasons.

- Unimplemented memory in a valid target space. For example, the DMA accessing XRAM higher than allowed by RAM array size.
- Any location in an invalid target space. For example, ICD addressing program space.
- Instruction bus reads outside of the memory range defined by either of the BMXIRAML/H registers. Note, the BMX will not generate bus errors for instruction bus accesses targeting Flash. The Flash controller will generate the bus error and indicate status in NVMCON register.
- Instruction bus reads of the debug RAM target, regardless of the BMXIRAML/H setting.
- A RAM write request is generated to the instruction RAM space defined by the BMXIRAML/H registers.
- Flash or RAM read results in an ECC Double-Bit Error (ECC DED). This bus error is generated by the target and passed up to the initiator, which will cause the initiator's Bus Error trap.
- The target indicates a bus error (root causes are target-specific).

When a bus error is generated within the BMX, it will set the relevant bit within the BMXxERR register for the initiator which generated the transaction. The user can diagnose the error by examining the error register of the initiator responsible for the trap, using BMXxERR in conjunction with the target error registers. See Table 4-3 for initiator indexes.

As the BMX supports simultaneous read and write operations to some targets, bus errors are split into separate read or write operation errors. This ensures that even if two simultaneous errors occur, they can both be captured.

Table 4-3. Target to Bus Error Index Mapping

TGTRERRy (BMXERRx[13:8])	Target
0	Program Space Read
1	Bus Splitter/SFRs
2	XRAM
3	YRAM
4	Debug RAM Target
5	Crypto

4.4.5.1. Valid Targets

Not all targets are valid destinations for each initiator. Refer to Table 4-4 for details on which targets are valid for each initiator.

Accessing an invalid target will generate a bus error and set the BADTGTWERR (BMXxERR[16]) or BADTGTRERR (BMXxERR[0]) bit.

Table 4-4. Valid Targets by Initiator

Initiators	Targets					
	PS Read	XRAM	YRAM	Debug RAM	SFRs	Crypto
CPU X Data	✓	✓	✓	If Debug mode is enabled	✓	—
CPU Y Data	—	✓	✓	If Debug mode is enabled	—	—
DMA	✓	✓	✓	—	✓	✓
CPU Instruction	—	✓	✓	—	—	—
Crypto	✓	✓	✓	✓	—	✓
CAN1	✓	✓	✓	—	—	—
NVM	✓	—	—	—	—	—
ICD	—	✓	✓	✓	✓	—

4.4.5.2. Initiator Bus Error Handling

The BMX does not directly generate any trap or interrupt signals on error events. Instead, the corresponding initiator will generate the event. Similarly, the BMX will not alter any data.

While initiators will handle bus errors differently, depending on individual module requirements, the most typical handling of a bus error is as follows:

- Ignore or discard returned read data and retain failed write data.
- Abort or halt an operation in progress.
- Generate an interrupt or trap signal to the interrupt controller indicating a Bus Error state.

4.4.5.3. Target Bus Error Handling

A target will generate a bus error in any situation where it is unable to deliver data back to the BMX on a read event or unable to latch data on a write event. The target will retain relevant error status information which can be used to determine the cause of the bus error.

The BMX will set the TGTWERRy (BMXERRx[28:24]) or TGTRERRy (BMXERRx[12:8]) bits for the specific target that caused the error within the Error register for the initiator that generated the bus transaction.

For example, an ECC DED error on an XRAM read by the CPU would set the TGTRERR2 (BMXERR0[10]) bit. Refer to Table 4-3 for target indices.

4.4.5.4. BMXxERR Registers

To properly deal with speculative data fetches, the BMXxERR registers have somewhat different behavior compared to standard error registers in other modules. When a bus error occurs, the BMX will always set the corresponding bit in the initiator’s BMXxERR register and set the bus error signal going to that initiator. However, the initiator may choose to do nothing about the bus error. This is most common in the case of the CPU when it is performing speculative fetches for data that never end up getting used. If the erroneous data fetch is unused, there will be no trap or interrupt error event.

As a result, the BMX does not have clear information on when the BMXxERR error flags can safely be cleared, nor can it ensure a software routine will run that can read and clear set flags. Therefore, to prevent past unused BMX error flags from staying set, the BMXxERR register will clear any previous flags whenever a new error event happens.

This effectively turns the BMXxERR register into a one-hot register, where there will generally only be a single bit set at a time, reflecting the most recent error event. It is only possible for two bits to be set in the event that read and write errors occur simultaneously.

4.4.5.5. Instruction RAM Window Registers

The BMXIRAML and BMXIRAMH registers define the valid address range for instruction RAM. Instruction bus accesses to RAM outside of this window will generate a bus error.

BMXIRAML defines the lower address of the window, including the address defined within BMXIRAML. The upper valid address in the window is BMXIRAMH – 1. Note that this means the BMXIRAMH register is noninclusive, instead of inclusive, which differs from the BMXIRAML register. This is done to provide an easy disable mechanism. Both registers will default to the same value (e.g., ‘0’s), effectively disabling execution from RAM at start-up.

4.4.5.6. Write and Read Protection of Instruction RAM Space

To ensure that the RAM allocated for execution is not accidentally modified, writes to RAM locations between BMXIRAML and BMXIRAMH are prevented by the BMX. A bus error will be generated and the IRAMWRERR (BMXxERR[18]) bit will be set if an initiator attempts to write to instruction RAM space.

Though IRAM space is executable memory, it will not support any read protection via code protect. Data in the IRAM area are readable as normal RAM space.

Attempts to execute RAM space outside of the IRAM window will generate a bus error and set the IRAMRDERR (BMXxERR[2]) bit.

4.4.5.7. Instruction RAM Configuration Sequence

The proper sequence for configuring and using IRAM is as follows:

1. Write BMXIRAML/BMXIRAMH to '0' to ensure no RAM locations are write-protected. This step is optional if the sequence is done at Reset as both registers reset to a value of '0'.
2. Copy instructions into desired RAM locations from Flash or another storage location (e.g., bootloader).
3. Set the IRAM start and end addresses by writing to the BMXIRAML and BMXIRAMH registers.
4. Vector to any RAM location within the IRAM window to begin execution from RAM.

The BMXIRAML and BMXIRAMH registers are implemented with write protection to ensure that users have the ability to lock the designated IRAM range after configuration. The BMX does not support aborts on RAM accesses. The CPU will be responsible for ignoring unneeded instruction data delivered from the RAM access.

When switching from executing from NVM to executing from RAM, the following will occur:

1. Any active NVM fetches would be aborted by the PBU. The new RAM instruction fetch will be immediately sent to the CPU as soon as data are available.
2. To prevent cache thrashing, PBU is put into a Standby state while running from RAM.

4.5. Application Example

4.5.1. Execute from RAM

BMX provides support for the CPU to execute from RAM. Execute from RAM will use the same priority and arbitration scheme as described in BMX Operation.

Example 4-1 executes user code from RAM instead of Flash. The code snippet copies code from Flash to RAM. BMX registers are configured to set up execution from RAM. Once the configuration is complete, the code branches to the RAM space where the Flash code is copied.

Example 4-1. Program Execution from RAM

```
int EnableRAMExecution(void);

// Reserve RAM space for an IVT in RAM + all functions that need to execute
from RAM
uint32_t __attribute__((noload, aligned(0x40), section(".RAMExecutionSpace")))
RAMExecutionSpace[1024];

int main(void)
{
    int isRamExecEnabled;
    isRamExecEnabled = EnableRAMExecution();
    if(isRamExecEnabled)
    {
        // Call the start address of RAM code execution stored in BMXIRAML here
    }
    while(1);
}

void __attribute__((aligned(0x4), section(".RAMExeFuncs"))) RamFunc(void)
{
      // Code that executes from RAM goes here
}

int EnableRAMExecution(void)
{
         // Code size to be executed from RAM
    const uint32_t codeLen    = __builtin_section_size(".RAMExeFuncs");
        // Base address of functions stored in flash, but intended to be
copied and executed from RAM at runtime
    const uint32_t flashFuncs = __builtin_section_begin(".RAMExeFuncs");
```

```
        // Abort if RAMExecutionSpace[] is declared too small
    if(codeLen > sizeof(RAMExecutionSpace))
        return -1;
       // If BMX already configured for RAM execution and ISR vectoring, do
nothing
    if(BMXIRAML != BMXIRAMH)
        return 0;
    /*Copy flash code to RAM*/
    __builtin_memcpy(&RAMExecutionSpace[0], (const uint32_t*)flashFuncs,
codeLen);
    // Configure BMX registers to execute code from RAM
    BMXIRAML = ((uint32_t)RAMExecutionSpace);
    BMXIRAMH = ((uint32_t)RAMExecutionSpace);
    BMXIRAMH += sizeof(RAMExecutionSpace);
    return 1;
}
```

Note: Alternatively, the compiler attribute '`ramfunc`' may be used to allocation a function into RAM for execution out of RAM. Please refer to the attributes section from the XC-DSC user guide for more details. This attribute is available on compiler versions 3.30 and above.

5. Data Memory

This section explains the data memory in dsPIC33AK256MPS306 devices. The dsPIC33AK256MPS306 features a data width of 32 bits. All internal registers and data space memory are organized to be 32 bits wide. The data spaces can be accessed as a single 64 Kbyte linear address range. Data memory is accessed through Address Generation Units (AGUs) and separate data paths.

The dsPIC33AK256MPS306 device family incorporates ECC support for data read/write and forced fault injection capability for use by functional safety focused customers. The ECC controller provides interrupt output for single and double-bit errors.

The following data memory features are implemented:

- Three Address Generation Units: Two for Read and One for Write
- Two SRAM Data Blocks for Independent Read and Write
- ECC Support for Data Read and Write
- ECC Provides Interrupt Output for Single and Double-Bit Errors
- Forced Fault Injection Capability for Use by Functional Safety Focused Customers

5.1. Device-Specific Information

Table 5-1 summarizes the available data memory size and the Start and End addresses for X and Y RAM spaces.

Table 5-1. Data Memory

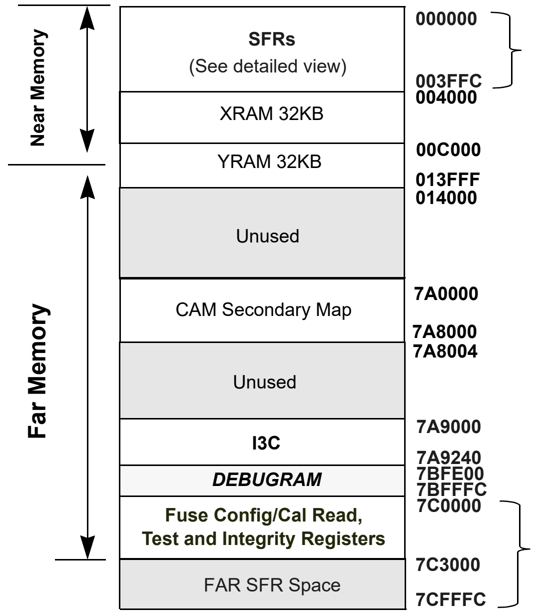
Device Variant	Data Memory	X RAM			Y RAM		
		Memory	Start Address	End Address	Memory	Start Address	End Address
dsPIC33AKXXXMPSXXX	64 Kbytes	32 Kbytes	0x004000	0x00BFFF	32 Kbytes	0x00C000	0x0013FFF

5.2. Architectural Overview

This section describes the features of the data memory. dsPIC33AK256MPS306 devices have a 64 Kbyte data memory space. Figure 5-1 shows a typical memory map for dsPIC33AK256MPS306 family devices.

The dsPIC33AK256MPS306 family of devices implements X and Y RAM in equal sizes. The RAM width is 32-bit in addition to seven parity bits to implement ECC. The DEBUGRAM is 512 bytes.

Figure 5-1. Data Memory Map for dsPIC33AK256MPS306 Family Devices



Note:

1. Memory areas are not shown to scale.
2. The dsPIC33AK256MPS306 devices will have a unified memory map with non-overlapping Program and Data address spaces, as shown in Figure 5-1. The address space is 8 MB total, starting at 000000 and ending at 7CFFFF.

SFR space and data RAM are mapped starting at address 000000 to allow near memory access from instructions that can encode a memory address. The near address range for most file register instructions is 64 KB in total. The SFR space is 16 KB and is separated into different clock speeds via a peripheral bus splitter:

- The address range 000000 - 000FFF is associated with a fast (1:1 System Clock) peripheral bus. A minimum number of SFRs are mapped into this region.
- The address range 001000 - 002FFF is associated with the standard speed (1:2 or slower) peripheral bus. The majority of the peripheral SFRs will be mapped into this region.
- The address range 003000 - 003FFF is associated with the slow speed (1:4 or slower) peripheral bus. Peripheral SFRs that require infrequent access will be mapped into this region.
- The address range 7A0000 - 7A9FFF is associated with the buffered peripheral bus (1:1 system clock), dedicated to modules that require pipelined read signals.
- The address range 7C0000 - 7CFFFF is associated with the slow-speed (1:4 or slower) peripheral bus dedicated to the calibration and configuration registers in the “far” memory space.

5.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3580	RAMXECCCON	31:24								
		23:16								
		15:8	ON							
		7:0								FLTINJ
0x3584	RAMXECCSTAT	31:24								
		23:16								
		15:8				ESEL				
		7:0			SECO	SEC			DEDO	DED
0x3588	RAMXECCFPTR	31:24								
		23:16								
		15:8			FLT2LPTR[5:0]					
		7:0			FLT1LPTR[5:0]					
0x358C	RAMXECCFADDR	31:24	ECCFADDR[31:24]							
		23:16	ECCFADDR[23:16]							
		15:8	ECCFADDR[15:8]							
		7:0	ECCFADDR[7:0]							
0x3590	RAMXECCEADDR	31:24	ECCEADDR[31:24]							
		23:16	ECCEADDR[23:16]							
		15:8	ECCEADDR[15:8]							
		7:0	ECCEADDR[7:0]							
0x3594	RAMXECCEDATA	31:24	ECCEDATA[31:24]							
		23:16	ECCEDATA[23:16]							
		15:8	ECCEDATA[15:8]							
		7:0	ECCEDATA[7:0]							
0x3598	RAMXECCVAL	31:24								
		23:16								
		15:8								
		7:0		ECCVAL[6:0]						
0x359C	RAMXECCSYND	31:24								
		23:16								
		15:8								
		7:0		ECCSYND[6:0]						
0x35A0	PWBXECCCON	31:24								
		23:16								
		15:8	ON							
		7:0								FLTINJ
0x35A4	PWBXECCSTAT	31:24								
		23:16								
		15:8				ESEL				
		7:0			SECO	SEC			DEDO	DED
0x35A8	PWBXECCFPTR	31:24								
		23:16								
		15:8			FLT2LPTR[5:0]					
		7:0			FLT1LPTR[5:0]					
0x35AC	PWBXECCFADDR	31:24	ECCFADDR[31:24]							
		23:16	ECCFADDR[23:16]							
		15:8	ECCFADDR[15:8]							
		7:0	ECCFADDR[7:0]							
0x35B0	PWBXECCEADDR	31:24	ECCEADDR[31:24]							
		23:16	ECCEADDR[23:16]							
		15:8	ECCEADDR[15:8]							
		7:0	ECCEADDR[7:0]							
0x35B4	PWBXECCEDATA	31:24	ECCEDATA[31:24]							
		23:16	ECCEDATA[23:16]							
		15:8	ECCEDATA[15:8]							
		7:0	ECCEDATA[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x35B8	PWBXECCVAL	31:24								
		23:16								
		15:8								
		7:0		ECCVAL[6:0]						
0x35BC	PWBXECCSYND	31:24								
		23:16								
		15:8								
		7:0		ECCSYND[6:0]						
0x35C0	RAMYECCCON	31:24								
		23:16								
		15:8	ON							
		7:0								FLTINJ
0x35C4	RAMYECCSTAT	31:24								
		23:16								
		15:8				ESEL				
		7:0			SECO	SEC			DEDO	DED
0x35C8	RAMYECCFPTR	31:24								
		23:16								
		15:8			FLT2LPTR[5:0]					
		7:0			FLT1LPTR[5:0]					
0x35CC	RAMYECCFADDR	31:24	ECCFADDR[31:24]							
		23:16	ECCFADDR[23:16]							
		15:8	ECCFADDR[15:8]							
		7:0	ECCFADDR[7:0]							
0x35D0	RAMYECCADDR	31:24	ECCEADDR[31:24]							
		23:16	ECCEADDR[23:16]							
		15:8	ECCEADDR[15:8]							
		7:0	ECCEADDR[7:0]							
0x35D4	RAMYECCEDATA	31:24	ECCEDATA[31:24]							
		23:16	ECCEDATA[23:16]							
		15:8	ECCEDATA[15:8]							
		7:0	ECCEDATA[7:0]							
0x35D8	RAMYECCVAL	31:24								
		23:16								
		15:8								
		7:0		ECCVAL[6:0]						
0x35DC	RAMYECCSYND	31:24								
		23:16								
		15:8								
		7:0		ECCSYND[6:0]						
0x35E0	PWBYECCCON	31:24								
		23:16								
		15:8	ON							
		7:0								FLTINJ
0x35E4	PWBYECCSTAT	31:24								
		23:16								
		15:8				ESEL				
		7:0			SECO	SEC			DEDO	DED
0x35E8	PWBYECCFPTR	31:24								
		23:16								
		15:8			FLT2LPTR[5:0]					
		7:0			FLT1LPTR[5:0]					
0x35EC	PWBYECCFADDR	31:24	ECCFADDR[31:24]							
		23:16	ECCFADDR[23:16]							
		15:8	ECCFADDR[15:8]							
		7:0	ECCFADDR[7:0]							
0x35F0	PWBYECCEADDR	31:24	ECCEADDR[31:24]							
		23:16	ECCEADDR[23:16]							
		15:8	ECCEADDR[15:8]							
		7:0	ECCEADDR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x35F4	PWBYECCEDATA	31:24	ECCEDATA[31:24]							
		23:16	ECCEDATA[23:16]							
		15:8	ECCEDATA[15:8]							
		7:0	ECCEDATA[7:0]							
0x35F8	PWBYECCVAL	31:24								
		23:16								
		15:8								
		7:0		ECCVAL[6:0]						
0x35FC	PWBYECCSYND	31:24								
		23:16								
		15:8								
		7:0		ECCSYND[6:0]						
0x3600 ... 0x3AFF	Reserved									
0x3B00	MBISTCON	31:24								
		23:16								
		15:8								FLTINJ
		7:0	MBISTDONE			MBISTSTAT				MBISTEN

5.3.1. RAM ECC Control Register

Name: RAMXECCCON, RAMYECCCON
Offset: 0x3580, 0x35C0

Note:
1. This bit can be set by software but not cleared. It is cleared by any device Reset.

Legend: R = Readable bit; W = Writable bit; S = Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON							
Access	R/S							
Reset	0							

Bit	7	6	5	4	3	2	1	0
								FLTINJ
Access								R/W
Reset								0

Bit 15 – ON Enable ECC functionality bit[(1)]
By default, ECC is disabled at a device Reset. It is the software's responsibility to initialize the RAM locations with valid data, then set this bit to "1".

Value	Description
1	ECC is enabled.
0	ECC is disabled.

Bit 0 – FLTINJ Fault Injection Enable bit

Value	Description
1	Fault injection is enabled when the read address of Data RAM[23:2] matches with RAMxECCFADDR[23:2].
0	Fault injection is disabled.

5.3.2. RAM ECC Status Register

Name: RAMXECCSTAT, RAMYECCSTAT
Offset: 0x3584, 0x35C4

Legend: R = Readable bit; W = Writable bit; C = Clearable bit; HS = Hardware Settable bit
Note:
1. This bit determines whether the RAMxECCEADDR, RAMxECCEDATA, RAMxECCVAL and RAMxECCSYND registers display information related to SEC or DED error events.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				ESEL				
Access				R/W				
Reset				0				

Bit	7	6	5	4	3	2	1	0
			SECO	SEC			DEDO	DED
Access			R/C/HS	R/C/HS			R/C/HS	R/C/HS
Reset			0	0			0	0

Bit 12 – ESEL Error Reporting Select bit[1]

Value	Description
1	Show SEC error event information.
0	Show DED error event information.

Bit 5 – SECO Single Error Correction Event Overflow Status bit

Value	Description
1	SEC event not captured due to overflow.
0	No SEC event overflow detected.

Bit 4 – SEC Single Error Correction Status bit

Value	Description
1	Single-bit error detected.
0	Single-bit error not detected.

Bit 1 – DEDO Double Error Detection Event Overflow Status bit

Value	Description
1	DED event not captured due to overflow.
0	No DED event overflow detected.

Bit 0 – DED Double Error Detection Indicator Status bit

Value	Description
1	Double-bit error detected.
0	Double-bit error not detected.

5.3.3. RAM ECC Fault Pointer Register

Name: RAMXECCFPTR, RAMYECCFPTR
Offset: 0x3588, 0x35C8

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			FLT2LPTR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			FLT1LPTR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 13:8 – FLT2LPTR[5:0] ECC Fault Injection Bit Pointer 2 bits

Value	Description
111111-100111	No Fault injection
100110	Fault injection (bit inversion) occurs on bit 38 of ECC bit order.
...	
000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order.
000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order.

Bits 5:0 – FLT1LPTR[5:0] ECC Fault Injection Bit Pointer 1 bits

Value	Description
111111-100111	No Fault injection
100110	Fault injection (bit inversion) occurs on bit 38 of ECC bit order.
...	
000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order.
000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order.

5.3.4. RAM ECC Fault Injection Address Register

Name: RAMXECCFADDR, RAMYECCFADDR
Offset: 0x358C, 0x35CC

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ECCFADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCFADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCFADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCFADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCFADDR[31:0] ECC Fault Injection Address bits

These register bits are compared against the RAM address read during the fault injection cycle. These register bits' values represent the relative address from the start address of X or Y data spaces, depending on the instance of this register. The start address should be added to this relative address to determine the corresponding data RAM address.

5.3.5. RAM ECC Error Address Register

Name: RAMXECCEADDR, RAMYECCADDR
Offset: 0x3590, 0x35D0

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
	ECCEADDR[31:24]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	ECCEADDR[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	ECCEADDR[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	ECCEADDR[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEADDR[31:0] ECC RAM Read Data Address bits

These bits represent the faulty memory location when Single bit (SEC) or Double bit (DED) ECC errors occur. The values of these register bits represent the relative address from the start address of the X or Y data spaces, depending on the instance of this register. To determine the corresponding system bus address, the start address should be added to this relative address.

5.3.6. RAM ECC Error Data Register

Name: RAMXECCEDATA, RAMYECCEDATA
Offset: 0x3594, 0x35D4

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
	ECCEDATA[31:24]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA[31:0] RAM Read Error Data bits
These bits represent the value stored in the faulty memory location when Single or Double bit ECC errors occur.

5.3.7. RAM ECC Value Register

Name: RAMXECCVAL, RAMYECCVAL
Offset: 0x3598, 0x35D8

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		ECCVAL[6:0]						
Access		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0	0	0	0	0	0	0

Bits 6:0 – ECCVAL[6:0] Error Correcting Code bits
These bits register the Error Correcting Code associated with the RAM read data when Single or Double bit ECC errors occur.

5.3.8. RAM ECC Syndrome Register

Name: RAMXECCSYND, RAMYECCSYND
Offset: 0x359C, 0x35DC

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		ECCSYND[6:0]						
Access		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0	0	0	0	0	0	0

Bits 6:0 – ECCSYND[6:0] Syndrome Value bits
These bits register the syndrome value associated with the RAM read data when Single or Double bit ECC errors occur.

5.3.9. PWB ECC RAM Control Register

Name: PWBXECCCON, PWBYECCCON
Offset: 0x35A0, 0x35E0

Note:
1. This bit can be set by software but not cleared. It is cleared by any device Reset.

Legend: R = Readable bit; W = Writable bit; S = Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON							
Access	R/S							
Reset	0							

Bit	7	6	5	4	3	2	1	0
								FLTINJ
Access								R/W
Reset								0

Bit 15 – ON Enable ECC functionality bit(1)
By default, ECC is disabled at a device Reset. It is the software responsibility to initialize the RAM locations with valid data, then set this bit to “1”.

Value	Description
1	ECC is enabled.
0	ECC is disabled.

Bit 0 – FLTINJ Fault Injection Enable bit

Value	Description
1	Fault injection is enabled when the content of the RAM address matches PWBECCXFADDR[23:2].
0	Fault injection is disabled.

5.3.10. PWB ECC RAM Status Register

Name: PWBXECCSTAT, PWBYECCSTAT
Offset: 0x35A4, 0x35E4

Legend: R = Readable bit; W = Writable bit; C = Clearable bit; HS = Hardware Settable bit
Note:

1. This bit determines whether the PWBxECCEADDR, PWBxECCEDATA, PWBxECCVAL and PWBxECCSYND registers display information related to SEC or DED error events.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				ESEL				
Access				R/W				
Reset				0				

Bit	7	6	5	4	3	2	1	0
			SECO	SEC			DEDO	DED
Access			R/C/HS	R/C/HS			R/C/HS	R/C/HS
Reset			0	0			0	0

Bit 12 – ESEL Error Reporting Select bit[(1)]

Value	Description
1	Show SEC error event information.
0	Show DED error event information.

Bit 5 – SECO Single Error Correction Event Overflow Status bit

Value	Description
1	SEC event not captured due to overflow.
0	No SEC event overflow detected.

Bit 4 – SEC Single Error Correction Status bit

Value	Description
1	Single-bit error detected.
0	Single-bit error not detected.

Bit 1 – DEDO Double Error Detection Event Overflow Status bit

Value	Description
1	DED event not captured due to overflow.
0	No DED event overflow detected.

Bit 0 – DED Double Error Detection Indicator Status bit

Value	Description
1	Double-bit error detected.
0	Double-bit error not detected.

5.3.11. PWB ECC RAM Fault Pointer Register

Name: PWBXECCFPTR, PWBYECCFPTR
Offset: 0x35A8, 0x35E8

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			FLT2LPTR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			FLT1LPTR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 13:8 – FLT2LPTR[5:0] ECC Fault Injection Bit Pointer 2 bits

Value	Description
111111-100111	No Fault injection
100110	Fault injection (bit inversion) occurs on bit 38 of ECC bit order.
...	
000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order.
000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order.

Bits 5:0 – FLT1LPTR[5:0] ECC Fault Injection Bit Pointer 1 bits

Value	Description
111111-100111	No Fault injection
100110	Fault injection (bit inversion) occurs on bit 38 of ECC bit order.
...	
000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order.
000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order.

5.3.12. PWB ECC RAM Fault Injection Address Register

Name: PWBXECCFADDR, PWBYECCFADDR
Offset: 0x35AC, 0x35EC

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ECCFADDR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCFADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCFADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCFADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCFADDR[31:0] ECC Fault Injection Address bits
These register bits are compared against the RAM address read during the fault injection cycle. These register bits' values represent the relative address from the start address of X or Y data spaces depending on the read or write instance of this register. The start address should be added to this relative address to know the corresponding data RAM address.

5.3.13. PWB ECC RAM Error Address Register

Name: PWBXECCEADDR, PWBYECCEADDR
Offset: 0x35B0, 0x35F0

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
	ECCEADDR[31:24]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEADDR[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEADDR[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEADDR[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEADDR[31:0] ECC RAM Read Data Address bits
These bits represent the faulty memory location when Single bit (SEC) or Double bit (DED) ECC errors occur. The values of these register bits represent the relative address from the start address of the X or Y data spaces, depending on the instance of this register. To determine the corresponding system bus address, the start address should be added to this relative address.

5.3.14. PWB ECC RAM Error Data Register

Name: PWBXECCEDATA, PWBYECCEDATA
Offset: 0x35B4, 0x35F4

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
	ECCEDATA[31:24]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA[31:0] RAM Read Error Data bits
These bits represent the value stored in the faulty memory location when Single or Double bit ECC errors occur.

5.3.15. PWB ECC RAM Value Register (a = x or y)

Name: PWBXECCVAL, PWBYECCVAL
Offset: 0x35B8, 0x35F8

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		ECCVAL[6:0]						
Access		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0	0	0	0	0	0	0

Bits 6:0 – ECCVAL[6:0] Error Correcting Code bits
These bits register the Error Correcting Code associated with the RAM read data when Single or Double bit ECC errors occur.

5.3.16. ECC Syndrome Register (a = x or y)

Name: PWBXECCSYND, PWBYECCSYND
Offset: 0x35BC, 0x35FC

Legend: R = Readable bit; HC = Hardware Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		ECCSYND[6:0]						
Access		R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset		0	0	0	0	0	0	0

Bits 6:0 – ECCSYND[6:0] Syndrome Value bits
These bits register the syndrome value associated with the RAM read data when Single or Double bit ECC errors occur.

5.3.17. MBIST Control Register

Name: MBISTCON
Offset: 0x3B00

Notes:
1. Resets only on a true POR Reset.
2. This bit will self-clear when the MBIST test is complete.

Legend: HS = Hardware Settable bit; HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
								FLTINJ
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
	MBISTDONE			MBISTSTAT				MBISTEN
Access	R/W/HS			R				R/W/HC
Reset	0			0				0

Bit 8 – FLTINJ MBIST Fault Inject Control bit(1)

Value	Description
1	The MBIST test will complete and sets MBISTSTAT = 1, simulating an SRAM test failure.
0	The MBIST test will execute normally.

Bit 7 – MBISTDONE MBIST Done Status bit

Value	Description
1	An MBIST operation has been executed.
0	No MBIST operation has occurred on the last Reset sequence.

Bit 4 – MBISTSTAT MBIST Status bit

Value	Description
1	The last MBIST failed.
0	The last MBIST passed.

Bit 0 – MBISTEN MBIST Enable bit(2)

Value	Description
1	MBIST test is armed; an MBIST test will execute at the next device Reset.
0	MBIST test is disarmed.

5.4. Operation

Data memory addresses between 0x0000 and 0x3FFF are reserved for the device's Special Function Registers (SFRs). The SFRs include control and status bits for the CPU and peripherals on the device. The data space memory map is shown in Figure 5-1.

The RAM begins at address 0x4000 and is split into two blocks, X and Y data spaces. For data writes, the X and Y data memory spaces are always accessed as a single, linear data space. For data reads, the X and Y data memory spaces can be accessed independently or as a single, linear space. Data reads for MCU class instructions always access the X and Y data spaces as a single combined data space. Dual source operand DSP instructions, such as the MAC instruction, access the X and Y data spaces separately to support simultaneous reads for the two source operands. MCU instructions can use any W register as an address pointer for a data read or write operation.

5.4.1. Near Data Memory

A 64 Kbyte address space, referred to as near data memory, is reserved in the data memory space between 0x00_0000 and 0x01_0000. Near data memory is directly addressable through a 16-bit absolute address field within all file register instructions.

The memory regions included in the near data region will depend on the amount of data memory implemented for each dsPIC33AK256MPS306 device variant. At a minimum, the near data region will include all of the SFRs and some of the X data memory. For devices that have smaller amounts of data memory, the near data region can include all of the X memory space and possibly some or all of the Y memory space. Refer to Figure 5-1 for more details.

Note: The entire 64 Kbyte near data space can be addressed directly using the `MOV` instruction. Refer to the *"dsPIC33A Programmer's Reference Manual" (*DS70005540*)* for further details.

5.4.2. Error Correcting Code (ECC)

To improve data memory performance and reliability, the dsPIC33AK256MPS306 device family includes Error Correcting Code (ECC) functionality as an integral part of the data memory. ECC can determine the presence of single bit errors in data memory, including which bit is in error, and correct the data without user intervention. Both X and Y memory support ECC and have their own control registers.

Each 32-bit data word in SRAM is completed by seven additional ECC bits, which are not accessible by the user.

Upon any 8/16/32-bit write in the memory, the seven ECC bits are computed and stored along with the data (the 8/16-bits writes are actually composed of an atomic read of 32 bits and modify/write 32 bits).

Upon any 8/16/32-bit read in the memory, if the ECC feature is disabled, then single or double errors are not detected and are not corrected. If the ECC feature is enabled, the ECC syndrome is computed on the related 32 data bits + 7 ECC bits.

Single bit errors are automatically identified and corrected on read back. An interrupt is generated if enabled. Double bit errors will be identified but not corrected. Either bus error or generic error traps are generated based on read or write path double-bit errors.

ECC operations for partial 8/16-bit write/read to data memory utilize PWB registers.

ECC operations for full word 32-bit write/read to data utilize RAM registers.

The user controls the ECC Fault injection through the ECCCON, ECCFPTR, ECCFADDR and ECCSTAT SFRs. Users may either create intentional Faults in data read from the data RAM or in data written into the data RAM. To report ECC Faults, the ECCFADDR must be configured with the target data memory location and subsequently read. Single or double bit Faults may be injected into any location within the data word (i.e., any data bit including the ECC parity bits).

5.4.2.1. Enabling ECC Fault Injection

The ECC Fault injection logic is enabled by setting ECCxCONbits.FLTINJ = 1.

5.4.2.2. Performing Fault Injection

The following sequence should be followed to inject Faults.

1. Load the data RAM target address into ECCFADDR register.
2. Select the first Fault bit determined by ECCFPTRbits.FLT1PTR[5:0]. The data RAM target bit is inverted to create the Fault.
3. If a double Fault is desired, select the second Fault bit determined by ECCFPTRbits.FLT2PTR[5:0], otherwise set ECCFPTR.FLT2PTR[5:0] = 6'h3F.
4. This step applies only to partial read/write operations using PWB registers for a single memory location.
   - Perform read/write operations on more than four memory locations other than the target address before proceeding.
5. Perform a read of the data RAM target address.

Notes:

1. RAM ECC registers operate in a 1:4 clock domain (see Figure 4.1) and are clocked by the Slow Speed Peripheral Clock. Software must allow sufficient delay after writing to these registers for the written values to propagate correctly.
2. For PWB injection to work, its internal FIFO must be flushed. This ensures that data is read directly from memory rather than from the FIFO buffer.

5.4.3. MBIST Overview

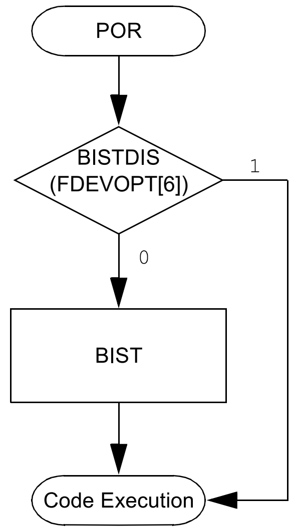
The dsPIC33A device family features a (data) Memory Built-In Self-Test (MBIST) that has the option to be run at start-up or run time. The memory test checks that all memory locations are functional and provides a pass/fail status of the RAM that can be used by software to take action if needed. If a failure is reported, the specific location(s) are not identified.

The MBISTCON register contains control and status bits for BIST operation. The MBISTDONE bit (MBISTCON[7]) indicates if a BIST has been run since the last Reset, and the MBISTSTAT bit (MBISTCON[4]) provides the pass/fail result. BIST is always clocked from an independent, non-configurable PLL, resulting in a 200 MHz clock rate unless CLKGEN4 is modified.

5.4.3.1. BIST at Start-up

The BIST can be configured to automatically run on a POR-type Reset, as shown in Figure 5-2. By default, when BISTDIS (FDEVOPT1[6]) = `1`, the BIST is disabled and will not be part of device start-up. If the BISTDIS bit is cleared during device programming, the BIST will run after all Configuration registers have been loaded and before code execution begins.

Figure 5-2. BIST Flowchart



5.4.3.2. Fault Simulation

A mechanism is available to simulate a BIST failure to allow testing of Fault handling software. When the FLTINJ bit is set during a run-time BIST, the MBISTSTAT bit will be set regardless of the test result. The procedure for a BIST Fault simulation is as follows:

1. Set the MBISTEN bit (MBISTCON[0]).
2. Set the FLTINJ bit (MBISTCON[8]).
3. Execute a Software Reset command.
4. Verify a Software Reset has occurred by reading SWR (RCON[6]) (optional).
5. Verify the MBISTDONE, MBSITSTAT and FLTINJ bits are all set.

5.4.3.3. BIST at Run Time

The BIST can also be run at any time during code execution. Note that a BIST will corrupt all of the RAM contents, including the stack pointer, and requires a subsequent Reset. The system should be prepared for a Reset before a BIST is performed. The BIST is invoked by setting the MBISTEN bit (MBISTCON[0]). The MBISTCON register is protected against accidental writes and requires an unlock sequence prior to writing. Only one bit can be set per unlock sequence. The procedure for a run-time BIST is as follows:

1. Write 0x0001 to the MBISTCON SFR.
2. Execute a Software Reset command.
3. Verify a Software Reset has occurred by reading SWR (RCON[6]) (optional).
4. Verify that the MBISTDONE bit is set.
5. Take action depending on test result indicated by MBISTSTAT.

6. Flash Program Memory

The dsPIC33AK256MPS306 family devices contain an internal programmable Flash for storing and executing application code. The high-endurance Flash provides great flexibility in code development and storage, combining a long retention life with a high number of read/write cycles. The memory is readable, writable and erasable during a normal operation over the entire V_{DD} range.

Flash program memory has the following features.

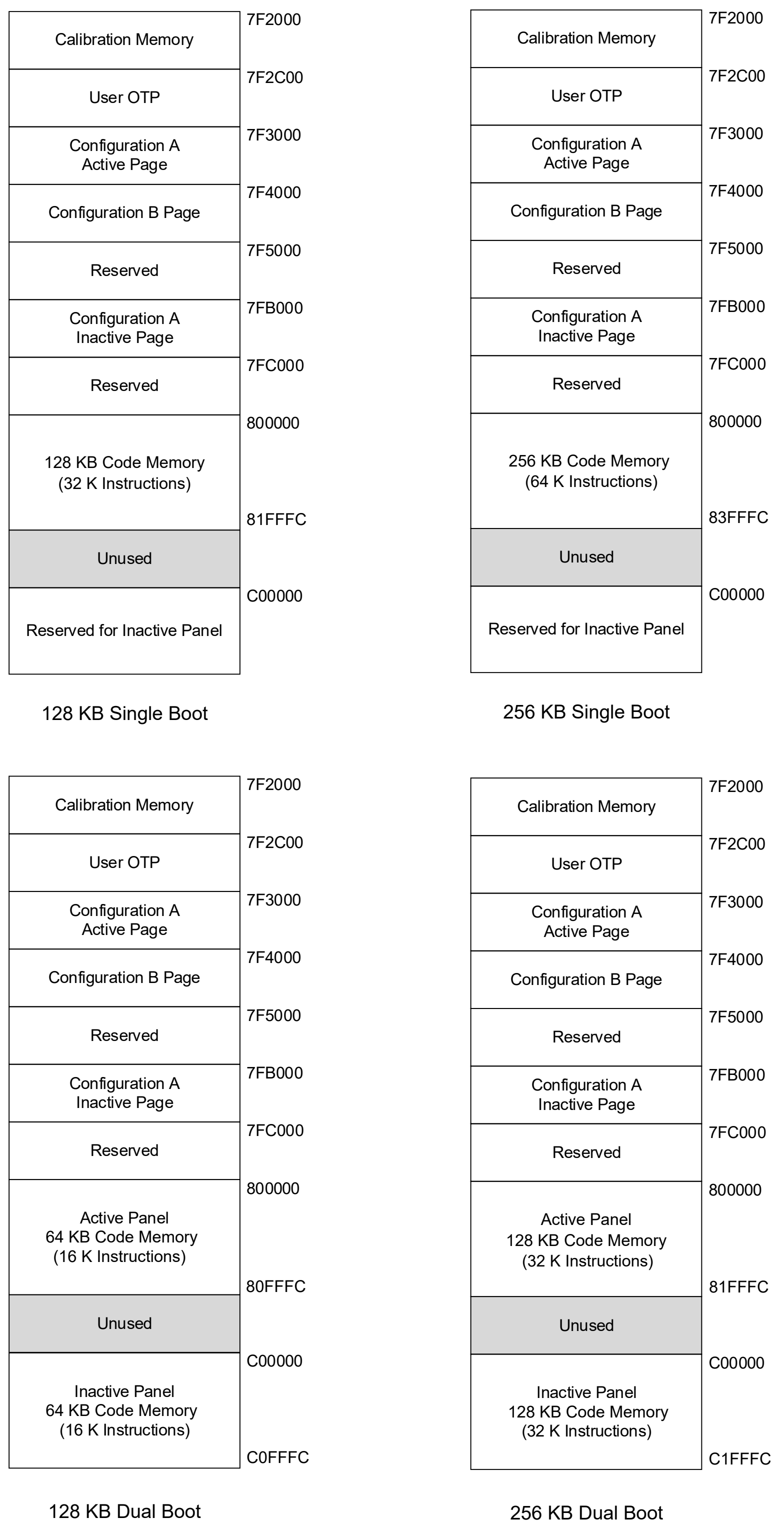
- 128-bit Data Flash Word Programming
- Bus Mastered Row Programming
- Page Erase Operation
- ECC Support for Data Read/Write: Includes Forced Fault Injection Capability For Use by Functional Safety Focused Customers
- Trap Event Output for ECC Double Bit Errors
- Interrupt Output for ECC Single Bit Errors
- New ECC Registers to Improve the Customer's Interface for Safety Relation Application by Improving the Fault Injection Information
- CRC Feature to Verify the Flash Contents

6.1. Device-Specific Information

Table 6-1. Device Memory Attributes

Device	Flash User Code Size (Kbytes)	Address Range	Pages	Page Size		Row Size	Rows Per Page
dsPIC33AK256MPSXXX	256	0x80000-0x83FFC	64	256 x 140 bits	1024 instruction words (32 bits)	32 128-bit words (+ECC+3 spare)	8
dsPIC33AK128MPSXXX	128	0x800000-0x81FFFC	32	256x140 bits	1024 instruction words (32 bits)	32 128-bit words (+ECC+3 spare)	8

Figure 6-1. dsPIC33A Program Memory Map[1,2]



Notes:
1. Memory areas are not shown to scale.
2. Refer to **"Memory Organization"** for the exact memory range of device.

6.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3000	NVMCON	31:24								
		23:16				WRRE		WREC[2:0]		
		15:8	WR	WREN	WRERR	NVMPIDL	SFTSWAP	P2ACTIV		
		7:0	LOCK	DRBV			MVMOP[3:0]			
0x3004	NVMADR	31:24								
		23:16	NVMADR[23:16]							
		15:8	NVMADR[15:8]							
		7:0	NVMADR[7:4]							
0x3008	NVMDATA0	31:24	DATA0[31:24]							
		23:16	DATA0[23:16]							
		15:8	DATA0[15:8]							
		7:0	DATA0[7:0]							
0x300C	NVMDATA1	31:24	DATA1[63:56]							
		23:16	DATA1[55:48]							
		15:8	DATA1[47:40]							
		7:0	DATA1[39:32]							
0x3010	NVMDATA2	31:24	DATA2[95:88]							
		23:16	DATA2[87:80]							
		15:8	DATA2[79:72]							
		7:0	DATA2[71:64]							
0x3014	NVMDATA3	31:24	DATA3[127:120]							
		23:16	DATA3[119:112]							
		15:8	DATA3[111:104]							
		7:0	DATA3[103:96]							
0x3018	NVMSRCADR	31:24								
		23:16	SRCADR[23:16]							
		15:8	SRCADR[15:8]							
		7:0	SRCADR[7:4]							
0x301C	NVMECCCON	31:24								
		23:16								
		15:8					ECCLOCK			
		7:0								FLTINJ
0x3020	NVMECCSTAT	31:24								
		23:16								
		15:8				ESEL			FLEC[1:0]	
		7:0			SECO	SEC			DEDO	DED
0x3024	NVMECCFPTR	31:24								
		23:16								
		15:8	FLT2PTR[7:0]							
		7:0	FLT1PTR[7:0]							
0x3028	NVMECCFADDR	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:4]				Reserved[3:0]			
0x302C	NVMECCEADDR	31:24								
		23:16	ADDR[23:16]							
		15:8	ADDR[15:8]							
		7:0	ADDR[7:4]				Reserved[3:0]			
0x3030	NVMECCEDATA0	31:24	ECCEDATA0[31:24]							
		23:16	ECCEDATA0[23:16]							
		15:8	ECCEDATA0[15:8]							
		7:0	ECCEDATA0[7:0]							
0x3034	NVMECCEDATA1	31:24	ECCEDATA1[63:56]							
		23:16	ECCEDATA1[55:48]							
		15:8	ECCEDATA1[47:40]							
		7:0	ECCEDATA1[39:32]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3038	NVMECCEDATA2	31:24	ECCEDATA2[95:88]							
		23:16	ECCEDATA2[87:80]							
		15:8	ECCEDATA2[79:72]							
		7:0	ECCEDATA2[71:64]							
0x303C	NVMECCEDATA3	31:24	NVMEDATA3[127:120]							
		23:16	NVMEDATA3[119:112]							
		15:8	NVMEDATA3[111:104]							
		7:0	NVMEDATA3[103:96]							
0x3040	NVMECCVAL	31:24								
		23:16								
		15:8								ECCVAL[8]
		7:0	ECCVAL[7:0]							
0x3044	NVMECCSYND	31:24								
		23:16								
		15:8								
		7:0	ECCSYND[7:0]							
0x3048	NVMCRCCON	31:24								
		23:16	DELAY[7:0]							
		15:8	CRCEN	START					CRCIDL[1:0]	
		7:0							CRCEC[1:0]	
0x304C	NVMCRCST	31:24								
		23:16	CRCST[23:16]							
		15:8	CRCST[15:12]				CRCST[11:8]			
		7:0	CRCST[7:0]							
0x3050	NVMCRCEND	31:24								
		23:16	CRCEND[23:16]							
		15:8	CRCEND[15:12]				CRCEND[11:8]			
		7:0	CRCEND[7:0]							
0x3054	NVMCRCSEED	31:24	CRCSEED[31:24]							
		23:16	CRCSEED[23:16]							
		15:8	CRCSEED[15:8]							
		7:0	CRCSEED[7:0]							
0x3058	NVMCRCDATA	31:24	CRCDATA[31:24]							
		23:16	CRCDATA[23:16]							
		15:8	CRCDATA[15:8]							
		7:0	CRCDATA[7:0]							

6.2.1. Nonvolatile Memory (NVM) Control Register

Name: NVMCON
Offset: 0x3000

Notes:
1. A BOR event Reset will indirectly clear WR by forcing the FSM to terminate the operation underway. The WR bit cannot be set if the operation target address held in the NVMADRx register falls within unimplemented address space.
2. Reset only on POR or BOR, but the actual initial state of P2ACTIV visible to the user will depend upon which panel is determined to be active after Reset exit.
3. A BMX address error is likely due to a bad NVMSRCADR value. The same error will generate a bus error TRAP via the interrupt controller. This will aid the software in diagnosing the issue.
4. The WRERR bit will remain set if an attempt is made to execute (WR=`1`) a reserved `PROGOP` command. The WR bit will not remain set.
5. “Word” is defined to be a 128-bit data value plus ECC (140 bits total). However, each word program command may consist of a sequence of fractional word programming operations.
6. Reserved when in Single Boot mode (DUAL_BOOTPRESENT=`0`).

Legend: C = Clearable bit; SO = Settable Only bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				WRRE		WREC[2:0]		
Access				R/W		R/W	R/W	R/W
Reset				0		0	0	0

Bit	15	14	13	12	11	10	9	8
	WR	WREN	WRERR	NVMPIDL	SFTSWAP	P2ACTIV		
Access	R/SO	R/W	R/W	R/W	R/C	R		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
	LOCK	DRBV			MVMOP[3:0]			
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit 20 – WRRE Program/Erase Reset Event bit

Value	Description
`1`	Warm Reset request during program/erase operation.
`0`	No event reported. Write 0 to clear, writing 1 has no effect.

Bits 18:16 – WREC[2:0] Program/Erase Error Code bits

Value	Description
`101`	Row Programming operation is not completed due to warm Reset.
`100`	System bus error during a Row Program operation.
`011`	Error reported by Flash panel control logic.

Value	Description
010	Security access control violation.
001	Invalid Program/Erase operation (PROGOP)
000	No error. Unused codes are reserved. Read as ‘0’ if WRERR=0.

Bit 15 – WR Write Control bit[(1)]

Value	Description
1	Initiates a memory or fuse element Program or Erase operation.
0	Program or Erase operation is complete and inactive.

Bit 14 – WREN Program/Erase Enable bit

Value	Description
1	Allows program/erase cycles.
0	Inhibits programming/erasing of memory or fuse elements. This bit cannot be updated if either the LOCK bit is set or the WR bit is set.

Bit 13 – WRERR Sequence Error Flag bit

Value	Description
1	Indicates an improper program or erase termination due to: • An attempt to execute a reserved `PROGOP` command. • An error detected by the Smart Write module such as Flash failures. • If a BMX address error is detected during a Row Programming operation. This bit cannot be set by software.
0	Either a POR or BOR has occurred or software cleared the WRERR bit.

Bit 12 – NVMPIDL NVM Power Down in Idle Enable bit

Value	Description
1	Flash panels enter a Sleep mode (very low-power mode) when device enters Idle mode.
0	Keep Flash and fuse panels powered in Standby mode when the device enters Idle mode.

Bit 11 – SFTSWAP Soft Swap Status bit

Value	Description
1	When Dual Boot Mode is present: Panels have been successfully swapped using the `BOOTSWP` instruction.
0	Awaiting successful panel swap using the `BOOTSWP` instruction.
	Write 0 to clear; writing 1 has no effect. Read as ‘0’ in single boot mode.

Bit 10 – P2ACTIV Dual Boot Active Region Status bit[(2)]

Value	Description
1	When Dual Boot Mode is present: Panel 2 is mapped into active region.
0	Panel 1 is mapped into active region.
	Read as '0' in single boot mode.

Bit 7 – LOCK Lock bit

Value	Description
1	Program/erase functions are disabled until after the next Reset.

Value	Description
0	Program/erase functions are not disabled. Write 1 to set, writing 0 has no effect.

Bit 6 – DRBV Data Read Buffer Valid bit

Value	Description
1	Data read buffer holds valid data.
0	Data read buffer invalid Write 0 to clear, writing 1 has no effect.

Bits 3:0 – MVMOP[3:0] NVM Operation Select bits(3,4,5,6)

Value	Description
0111-0101	Reserved
0100	Next WR command performs an Inactive Partition Erase operation.
0011	The next WR command will perform a memory Page Erase operation.
0010	The next WR command will perform a Row Program 1 operation.
0001	The next WR command will perform a Word Program 1, 3 operation (data source: NVMDATAx).
0000	Reserved

6.2.2. Nonvolatile Memory Lower Address Register[(1)]

Name: NVMADR
Offset: 0x3004

Note:
1. This register is not writable when WR = 1.

Legend: x = Bit is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	NVMADR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMADR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	NVMADR[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 23:4 – NVMADR[23:4] NVM Address Register bits[(1)]
NVM Address register used to program a Flash word or Flash row or perform a page erase. During row programming, the address may point to any Flash word within the Flash row. The row programming always starts at the beginning of the Flash row. NVMADR [3:0] is hard coded to logic '0000'.

6.2.3. NVM Write Data 0 Register

Name: NVMDATA0
Offset: 0x3008
Reset: 0
Property: R/W

Notes:
1. This register is not writable when WR = 1.
2. This register is readable only when NVMCON.WREN = 0. An attempted read while NVMCON.WREN = 1 will return (by convention) all "1's".
3. This register is cleared after a Flash write operation completes (when NVMCON.WR returns to '0').
4. This register is also mapped into user address space as SLVDATAL.

Bit	31	30	29	28	27	26	25	24
	DATA0[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA0[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA0[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA0[31:0] NVM Write Data Register bits[1,2,3,4]

6.2.4. NVM Write Data 1 Register

Name: NVMDATA1
Offset: 0x300C
Reset: 0
Property: R/W

Notes:

1. This register is not writable when WR = 1.
2. This register is readable only when NVMCON.WREN = 0.
3. This register is not readable when NVMCON.WREN = 1. An attempted read will return (by convention) all '1's.
4. This register is cleared after a Flash write operation completes (when NVMCON.WR returns to '0').
5. This register is also mapped into the user address space as SLVDATAL.

Bit	31	30	29	28	27	26	25	24
	DATA1[63:56]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA1[55:48]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA1[47:40]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA1[39:32]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA1[63:32] NVM Write Data Register bits(1,2,3,4,5)

6.2.5. NVM Write Data 2 Register

Name: NVMDATA2
Offset: 0x3010
Reset: 0
Property: R/W

Notes:

1. This register is not writable when WR = `1`.
2. This register is readable only when NVMCON.WREN = `0`.
3. This register is not readable when NVMCON.WREN = `1`. An attempted read will return (by convention) all '`1`'s.
4. This register is cleared after a Flash write operation completes (when NVMCON.WR returns to '`0`').
5. This register is also mapped into the user address space as SLVDATAL.

Bit	31	30	29	28	27	26	25	24
	DATA2[95:88]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA2[87:80]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA2[79:72]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA2[71:64]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA2[95:64] NVM Write Data Register bits[1,2,3,4,5]

6.2.6. NVM Write Data 3 Register

Name: NVMDATA3
Offset: 0x3014
Reset: 0
Property: R/W

Notes:

1. This register is not writable when WR = 1.
2. This register is readable only when NVMCON.WREN = 0.
3. This register is not readable when NVMCON.WREN = 1. An attempted read will return (by convention) all '1's.
4. This register is cleared after a Flash write operation completes (when NVMCON.WR returns to '0').
5. This register is also mapped into the user address space as SLVDATAL.

Bit	31	30	29	28	27	26	25	24
	DATA3[127:120]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA3[119:112]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA3[111:104]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA3[103:96]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA3[127:96] NVM Write Data Register bits[1,2,3,4,5]

6.2.7. NVM Source Data Address Register[1,2]

Name: NVMSRCADR
Offset: 0x3018

Notes:
1. This register is not writable when WR = 1.
2. This address must be aligned to a RAM word address (word-aligned).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SRCADR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SRCADR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SRCADR[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 23:4 – SRCADR[23:4] RAM Base Address register for Row Programming bits
The address is always on 32-bit word boundaries.

6.2.8. NVM ECC Control Register

Name: NVMECCCON
Offset: 0x301C

Notes:
1. The LOCK bit is settable by software and can only be cleared via a reset.
2. If the ECCLOCK=1, then the FLTINJ bit is a don't care and the fault injection function is disabled.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					ECCLOCK			
Access					R/S			
Reset					0			

Bit	7	6	5	4	3	2	1	0
								FLTINJ
Access								R/W
Reset								0

Bit 11 – ECCLOCK ECC Fault Inject LOCK bit(1)

Value	Description
1	ECC fault injection function is disabled.
0	ECC fault injection functionality is enabled.

Bit 0 – FLTINJ Fault Injection Sequence Enable bit(2)

Value	Description
1	Enable the fault injection to occur on NVM address match with ECCFADR.
0	Fault injection is disabled.

6.2.9. ECC RAM Status Register

Name: NVMECCSTAT
Offset: 0x3020

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				ESEL			FLEC[1:0]	
Access				R/W			R/x	R/x
Reset				0			0	0

Bit	7	6	5	4	3	2	1	0
			SECO	SEC			DEDO	DED
Access			R/C/HS	R/C/HS			R/C/HS	R/C/HS
Reset			0	0			0	0

Bit 12 – ESEL Error Reporting Select bit

Value	Description
1	Select SEC information for ECCEADDR, ECCEDATAx, ECCVAL, ECCSYN registers.
0	Select DED information for ECCEADDR, ECCEDATAx, ECCVAL, ECCSYN registers.

Bits 9:8 – FLEC[1:0] Fuse Load Error Code bit

Value	Description
11	One or more DED errors. One or more default fuse values used.
10	One or more DED errors. No default fuse values used.
01	One or more SEC errors. No DED errors
00	No ECC bit errors

Bit 5 – SECO SEC Event Overflow bit

Value	Description
1	SEC error not captured due to overflow.
0	No SEC error overflow reported.

Bit 4 – SEC SEC Event Reported bit

Value	Description
1	Single bit error reported.
0	Single bit error not reported.

Bit 1 – DEDO DED Event Overflow bit

Value	Description
1	DED error not captured due to overflow.

Value	Description
0	No DED error overflow reported.

Bit 0 – DED DED Event Reported bit

Value	Description
1	Double bit error reported.
0	Double bit error not reported.

6.2.10. NVM ECC Fault Injection Pointer Register

Name: NVMECCFPTR
Offset: 0x3024

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FLT2PTR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLT1PTR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – FLT2PTR[7:0] ECC Fault Injection Bit Pointer 2 bits

Value	Description
11111111-10001001	No fault injection occurs.
10001000	Fault injection (bit inversion) occurs on bit 136 of ECC bit order.
...	
00000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order.
00000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order.

Bits 7:0 – FLT1PTR[7:0] ECC Fault Injection Bit Pointer 1 bits

Value	Description
11111111-10001001	No fault injection occurs.
10001000	Fault injection (bit inversion) occurs on bit 136 of ECC bit order.
...	
00000001	Fault injection (bit inversion) occurs on bit 1 of ECC bit order.
00000000	Fault injection (bit inversion) occurs on bit 0 of ECC bit order.

6.2.11. NVM ECC Fault Injection Address Register

Name: NVMECCFADDR
Offset: 0x3028

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	U	U	U	U
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – ADDR[23:16]

Bits 15:8 – ADDR[15:8]

Bits 7:4 – ADDR[7:4] ECC Fault Injection Address bits
Address of the targeted Flash word for Fault injection.

Bits 3:0 – Reserved[3:0] Maintain as '0'

6.2.12. NVM ECC Error Address Register

Name: NVMECCEADDR
Offset: 0x302C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:4]				Reserved[3:0]			
Access	R/W	R/W	R/W	R/W	U	U	U	U
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – ADDR[23:16]

Bits 15:8 – ADDR[15:8]

Bits 7:4 – ADDR[7:4] ECC Address of Read Data bits
These bits register the location of the NVM read data when the SEC or DED bit is set in the NVMECCSTAT register.

Bits 3:0 – Reserved[3:0] Maintain as '0'

6.2.13. NVM ECC Error Data 0 Register

Name: NVMECCEDATA0
Offset: 0x3030

Legend: HS = Hardware Settable; HC = Hardware Clearable

Bit	31	30	29	28	27	26	25	24
	ECCEDATA0[31:24]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA0[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA0[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA0[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA0[31:0] NVM Read Error Data bits
These bits register the NVM read data (taking Fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.14. NVM ECC Error Data 1 Register

Name: NVMECCEDATA1
Offset: 0x3034

Legend: HS = Hardware Settable

Bit	31	30	29	28	27	26	25	24
	ECCEDATA1[63:56]							
Access	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA1[55:48]							
Access	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA1[47:40]							
Access	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA1[39:32]							
Access	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC	R/S/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA1[63:32] NVM Read Error Data bits
These bits register the NVM read data (taking Fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.15. NVM ECC Error Data 2 Register

Name: NVMECCEDATA2
Offset: 0x3038

Legend: HS = Hardware Settable; HC = Hardware Clearable

Bit	31	30	29	28	27	26	25	24
	ECCEDATA2[95:88]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ECCEDATA2[87:80]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ECCEDATA2[79:72]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ECCEDATA2[71:64]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ECCEDATA2[95:64] NVM Read Error Data bits
These bits register the NVM read data (taking Fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.16. NVM ECC Error Data Register

Name: NVMECCEDATA3
Offset: 0x303C

Legend: HS = Hardware Settable; Hardware Clearable

Bit	31	30	29	28	27	26	25	24
	NVMEDATA3[127:120]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	NVMEDATA3[119:112]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	NVMEDATA3[111:104]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	NVMEDATA3[103:96]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – NVMEDATA3[127:96] NVM Read Error Data bits
These bits register the NVM read data (taking Fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.17. NVM ECC Value Register

Name: NVMECCVAL
Offset: 0x3040

Legend: HS = Hardware Settable; Hardware Clearable

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
								ECCVAL[8]
Access								R/HS/HC
Reset								0

Bit	7	6	5	4	3	2	1	0
	ECCVAL[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 8:0 – ECCVAL[8:0] Error Correcting Code for the Data bits
These bits register the Error Correcting Code associated with the NVM read data (taking Fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.18. NVM ECC Syndrome Register

Name: NVMECCSYND
Offset: 0x3044

Note:

1. The syndrome bit encoding indicates the Hamming code bit locations. This value is meaningful in single bit detection only.

Legend: HS = Hardware Settable; Hardware Clearable

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	ECCSYND[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – ECCSYND[7:0] Syndrome Value bits(1)
These bits register the syndrome value associated with the NVM read data (taking Fault injections into account) when the SEC or DED bit is set in the NVMECCSTAT register.

6.2.19. NVM CRC Control Register

Name: NVMCRCCON
Offset: 0x3048

Legend: HS = Hardware Settable; Hardware Clearable

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DELAY[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCEN	START					CRCIDL[1:0]	
Access	R/W	R/HS/HC					R/W	R/W
Reset	0	0					0	0

Bit	7	6	5	4	3	2	1	0
							CRCEC[1:0]	
Access							R/HS/HC	R/HS/HC
Reset							0	0

Bits 23:16 – DELAY[7:0] Delay Between CRC Accesses bits
Delay count in system clock cycles between CRC accesses.

Bit 15 – CRCEN Enable CRC Function bit

Value	Description
1	CRC function enabled.
0	CRC function disabled.

Bit 14 – START Start CRC calculation bit

Value	Description
1	Start CRC calculation (CRC in progress).
0	CRC calculation complete (CRC function idle).

Bits 9:8 – CRCIDL[1:0] Idle Operation Control bits

Value	Description
3	Reserved
2	CRC operates in Idle mode and is stopped in CPU Run mode.
1	CRC operates in CPU Run mode and is stopped in Idle mode.
0	CRC operates in CPU Run mode and Idle.

Bits 1:0 – CRCEC[1:0] CRC Error Code bits

Value	Description
3	Invalid address (start address greater than end address)
2	Flash ECC DED error
1	Security access control violation

Value	Description
0	No error

6.2.20. NVM CRC Start Address Register

Name: NVMCRCST
Offset: 0x304C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CRCST[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCST[15:12]				CRCST[11:8]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCST[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – CRCST[23:16] Start Address Offset bits

Bits 15:12 – CRCST[15:12] Start Address Offset bits

Bits 11:8 – CRCST[11:8] Start Address (least significant bits, fixed value 0x000) bits

Bits 7:0 – CRCST[7:0] Start Address (least significant bits, fixed value 0x000) bits

6.2.21. NVM CRC End Address Register

Name: NVMCRCEND
Offset: 0x3050

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CRCEND[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCEND[15:12]				CRCEND[11:8]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	CRCEND[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bits 23:16 – CRCEND[23:16] End Address Offset bits

Bits 15:12 – CRCEND[15:12] End Address Offset bits

Bits 11:8 – CRCEND[11:8] End Address (least significant bits fixed value 0xFFF) bits

Bits 7:0 – CRCEND[7:0] End Address (least significant bits fixed value 0xFFF) bits

6.2.22. NVM CRC Seed Register

Name: NVMCRCSEED
Offset: 0x3054

Bit	31	30	29	28	27	26	25	24
	CRCSEED[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	CRCSEED[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	CRCSEED[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	CRCSEED[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CRCSEED[31:0] 32-Bit Seed Value for CRC Calculation bits (initial value with each bit inverted)

6.2.23. NVM CRC Output Data Register

Name: NVMCRCDATA
Offset: 0x3058

Bit	31	30	29	28	27	26	25	24
	CRCDATA[31:24]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCDATA[23:16]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCDATA[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCDATA[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CRCDATA[31:0] NVM CRC Data Output bits

6.3. Operation

6.3.1. Reading Program Memory

Program memory is read linearly, similar to how data memory is read. The `MOV` instruction is used to read the program memory.

Previously, devices used the table read feature to read the Flash program memory. However, in dsPIC33A devices, TBLRDL/TBLRDH features are unimplemented.
Note: The DRBV bit is set if the data read buffer holds valid data. It is recommended to invalidate the buffer by writing 0 to the DRBV bit after a program/erase operation or modification to security access controls to clear stale data from the buffer.

6.3.2. Programming Memory Writes

There are two methods that the user application can use to program Flash memory.

- Run-Time Self-Programming (RTSP)
- In-Circuit Serial Programming™ (ICSP™)

6.3.2.1. Run-Time Self-Programming (RTSP)

RTSP allows the user application to modify Flash program memory contents. The program/erase operations use NVMCON, NVMADR, NVMSRCADR and NVMDATA (0-3) registers to modify the Flash. With RTSP, the user application can erase a single page of Flash memory and program either a word or row of the memory.

Flash is programmed using either a Word Program (128-bits) or Row Program (512 bytes) operation. Flash must be erased before programming. While programming operations are very reliable, the best practice is to verify Flash contents after programming. This can be done using either the integrated CRC function or with a direct comparison of programmed Flash with the source data. A 128-bit Flash word is the smallest unit of Flash memory that can be programmed. The ECC parity bits associated with each Flash word are programmed automatically for both word and row programming. Programing a Flash word more than once will generally result in the word having incorrect ECC parity bit values.

All erase and program operations using the PROGOP interface must check the status of the WR bit and the WRERR bit to determine the completion of the requested erase or programming operation. Alternatively, software can use an interrupt to monitor completion of the programming operation. A completion interrupt is generated if the operation completed successfully or if the operation was terminated by an error.

6.3.2.1.1. Status Bits

Program Flash memory has two status bits in the NVMCON register to check the status of the NVM operation initiated by the user: the WRERR bit and the WREC bit. These bits ensure that the programming sequence has completed successfully.

WRERR Error Flag Bit

When a program or erase operation is initiated using the NVMCON.WR bit, the NVMCON.WRERR bit will indicate if the operation was properly completed. The user may check the state of the WRERR bit at any time after WR is set. Typically, the application will wait for WR to be cleared (indicating the program or erase sequence has completed), then check WRERR to confirm if the sequence completed successfully or not.

If WRERR = `0` at that time, the operation was completed successfully. If WRERR = `1` at that time, the operation did not complete successfully. The WRERR bit is cleared on a cold Reset and should be cleared by firmware before initiating a program/erase operation.

WREC Bit

The WREC field indicates the error type when the WRERR bit is set. Program/erase operation errors include:

- Row programming not completed due to a warm Reset
- System bus error during a row program operation
- Error reported by Flash panel control logic
- Security access control violation
- Invalid program/erase operation (PROGOP)

Note: WREC is cleared on a cold Reset and when WRERR = 0 .

WRRE Bit

The WRRE bit indicates if there was a warm Reset (Implications of Device Reset) request during a program/erase operation.

The PROGOP, WRERR, WREC and WRRE fields are not cleared on a warm Reset. This provides the error status for a program/erase operation initiated before a warm Reset.

6.3.2.1.2. LOCK Bit

The dsPIC33A devices have a one-way LOCK feature implemented through its NVMCON register. Once software sets the LOCK bit, only a Reset will clear the LOCK bit. The LOCK bit, if set, prevents all erase and programming operations. If the LOCK bit is set while an erase or programming operation is in progress, the operation can continue to completion.

A one-way ECCLOCK has been implemented in the ECCCON register. Once the software sets the ECCLOCK bit, only a Reset will clear the ECCLOCK bit. If the ECCLOCK bit is set, the Fault injection feature is disabled.

6.3.2.2. Flash Programming Operations

A program or erase operation is necessary for programming or erasing the internal Flash program memory. The following operations can be used to modify the Flash contents.

1. Word program
2. Row program
3. Page erase

Setting the WR bit (NVMCON[15]) starts the operation. The WR bit is automatically cleared when the operation is finished.

The CPU stalls until the programming operation is finished. The CPU will not execute any instructions or respond to interrupts during this time. If any interrupts occur during the programming cycle, they will remain pending until the cycle completes.

Note: In the event of any Reset other than a POR or BOR, the erase or program operation underway can complete while the device resets.

6.3.2.2.1. Flash Page Erase

A page erase is necessary before programming the Flash memory. This will ensure that the old contents of the page are erased and programmed with all `1`'s before programming with new content. Prior to a page erase operation, the target page is selected by writing an address within the page into the Write Address register, NVMADR. A page erase will erase eight rows of data concurrently.

User Page Erase Sequence

Executing a page erase sequence typically includes the following user code steps, subject to the constraints externally imposed by the Security module:

1. Load NVMADR with an address within the page to be erased.

2. Configure NVMCON to page erase:
   - WREN = 1
   - PROGOP = page erase (see NVMCON)
3. Set the WR bit.
4. The program sequence completes and the WR bit is cleared by hardware.
5. Clear WREN bit.
6. Test the WRERR bit to ensure the erase sequence completed successfully.

6.3.2.2.2. Word Programming

The smallest block of data that can be programmed by the user is one Flash word (128 bits+ECC). The data to be word programmed must be loaded into NVMDATAx before the word programming sequence is initiated. For the word programming PROGOP, the Flash word at the location pointed to by NVMADR is programmed. Note that NVMADR is a word address, so NVMADR [3:0] is defined as 0b0000 to force Flash word alignment of the operation. To keep the contents secure, the NVMDATAx register is cleared upon completion of any Flash write operation.

User Word Programming Sequence

A word programming sequence typically includes the following steps, subject to the constraints externally imposed by the security module:

1. Write data to be programmed to NVMDATAx.
2. Load NVMADR with the Flash address to be programmed.
3. Configure NVMCON to word program:
   - WREN = 1
   - PROGOP = Word program (see NVMCON)
4. Set the WR bit.
5. The program sequence completes, and the WR bit is cleared by hardware.
6. Clear the WREN bit.
7. Test the WRERR bit to ensure the program sequence completed successfully.

Note: Writes to NVMDATAx registers are inhibited while WR = 1.

6.3.2.2.3. Bus Mastered Row Programming

In dsPIC33A devices, row programming is performed directly from a buffer space in data RAM. The location of the RAM buffer is determined by the NVMSRCADR register(s), which are loaded with the data RAM address containing the first word of program data to be written. Before performing the programming operation, the buffer space in RAM must be loaded with the row of data to be programmed.

One row consists of 32 Flash words (128 bits + ECC). Row programming starts at the lowest address in the row and proceeds up to the highest address in the row. The module is not specified to operate when the operation is somewhere in the middle of the row and then wraps around.

The row program command automatically sequences through a row's worth of write data stored (by the user) in local memory. This module implements row programming operations as a sequence of 32, 140-bit word programming sequences.

During a row programming operation, the Flash controller becomes the bus initiator, and system RAM is used to store the row programming data. Firmware loads the row data into system RAM and configures the NVMSRCADR register with the starting address of the data in RAM prior to starting the Row Program operation. This address is word-aligned to 32-bit RAM addresses. The NVMADR address must be aligned to a Flash word address boundary where address [3:0] = 0b0000. During row programming, the NVMADR address may point to any Flash word within the Flash row.

BMX Bus Error During Row Programming

If the NVMSRCADR value is invalid and the NVM is performing a row programming operation, the BMX will not be able to connect to the RAM containing the needed data. The BMX will generate an error if it cannot access the address specified by the NVMSRCADR register. This error triggers the NVM to terminate the row programming operation and set the WRERR and WREC status bits. The NVM Module will generate an NVM Trap signal to the interrupt controller to alert the user software that a bus error has occurred.

User Row Programming Sequence

A row programming sequence typically includes the following steps, subject to the constraints externally imposed by the Security module:

1. Load NVMSRCADR with the address of data in device RAM.
2. Load NVMADR with the Flash address to be programmed.
3. Configure NVMCON to the Row program:
   - WREN = `1`
   - PROGOP = 0x2 (see NVMCON)
4. Set the WR bit.
5. The program sequence completes, and the WR bit is cleared by hardware.
6. Clear the WREN bit to avoid accidental writes.
7. Test the WRERR bit to ensure the program sequence completed successfully.

Note: Writes to RAM used during Bus Mastered Row programming cannot be protected. Consequently, the user must be aware that inadvertent corruption of this area of RAM is possible during programming.

6.3.3. Error Correcting Code (ECC)

To improve program memory performance and durability, select dsPIC33A devices that include Error Correcting Code (ECC) functionality as an integral part of the Flash memory controller. ECC can determine the presence of single-bit errors in program data, including which bit is in error, and correct the data without user intervention. When implemented, ECC is automatic and cannot be disabled.

Additionally, users can inject single-bit errors or double-bit errors into the ECC calculation. This allows functional safety focused users to be able to test the ECC calculation and fault generation logic at run time to verify there are no latent faults present in the system.

When data is written to program memory, ECC generates a 9-bit Hamming code parity value, an 8-bit Single Error Correction (SEC) value and one extra bit to support Double Error Correction (DED) for every 128-bit input data value. Parity data is not memory-mapped and is inaccessible. When the data is read back, the ECC calculates parity on it and compares it to the previously stored parity value. If a parity mismatch occurs, there are two possible outcomes:

- Single-bit errors are automatically identified and corrected on readback. An optional device-level interrupt is also generated.
- Double-bit errors will generate a bus error trap. If special exception handling for the trap is not implemented, a device Reset will also occur.

The user controls the ECC Fault injection through the ECCCON (NVMECCCON), ECCFPTR (NVMECCFPTR), ECCFADDR (NVMECCFADDR) and the ECCSTAT (NVMECCSTAT) SFRs. In keeping with all other NVM Controller SFRs, writes to these registers are inhibited during any Flash write sequence. Users may either create intentional faults in data read from the Flash or in data written into the Flash. Single-bit or double-bit faults may be injected into any location within the data word (i.e., any data bit but also including the ECC parity bits).

6.3.3.1. Enabling ECC Fault Injection

The ECC Fault injection logic is enabled by setting EECCON.FLTINJ = `1`. To help avoid unintentional enabling, the ECCCON.ECCLOCK bit can be set to disable all Fault injection operations. The ECCLOCK bit is only cleared via a Reset.

6.3.3.1.1. Performing Fault Injection

The following sequence should be followed to inject Faults:

1. Load Flash target address into ECCFADDR register.
2. Select first Fault bit determined by ECCCON.FLT1PTR [7:0]. The target bit is inverted to create the Fault.
3. If a double Fault is desired, select the second Fault bit determined by ECCCON.FLT2PTR [7:0], otherwise set ECCCON.FLT2PTR [7:0] = 8'hFF.
4. Perform a read of the Flash target address.

The ECCSTAT is updated whenever a read occurs and the read address matches ECCFADDR. The expected usage of a data read Fault injection is to attempt to read a predefined data set with known good ECC and inject an error immediately prior to the ECC evaluation, thereby testing the response of the ECC block under different data stimuli.

Note: It is recommended to invalidate the data read buffer (clear the DRBV bit) before an ECC Fault injection to clear the stale data and to force a new read.

6.3.3.1.2. Implications of Device Reset

Cold Reset

A loss of power (Cold Reset) during a program/erase immediately terminates the operation. In this case, the targeted Flash may be partially programmed/erased and reads of the affected Flash may return invalid data or generate ECC errors. There is no status register indication that a program or erase was terminated by a Cold Reset. This needs to be determined based on the state of Flash. The NVM Controller CRC function can be used to evaluate the integrity of Flash after a Cold Reset.

Warm Reset

If a Warm Reset request occurs during a word program or page erase operation, the operation continues with the rest of the device held in Reset until the operation completes. The row programming operation cannot be completed due to its use of the RAM, which is not accessible in Reset. The WRRE bit is set when a Warm Reset request occurs during a program/erase operation.

The Fault injection capability of the ECC is a feature intended for use only at run time. It is not possible to inject errors when the Flash based fuses are read (i.e., during the Reset sequence). Should a Reset (other than BOR or POR) occur during a Fault injection write, the write will be allowed to complete just like any other Flash write. Should the Reset be due to a BOR or POR, the write will have been aborted because power was removed from the panel. Flash Controller and ECC Fault injection registers related to Flash writes are subject only to POR. Unless the Reset was a POR, the user may check the state of NVMCON.WRERR to determine if a write had failed prior to Reset.

6.3.4. NVM CRC

The NVM Controller features a CRC-32 checksum function that can calculate or verify checksums over multiple 4kB Flash pages, including user programs, user data or information Flash. This function helps ensure the correct programming or erasure of Flash pages by operating at a rate of one 32-bit word per system clock cycle. It allows for integrity checks without exposing the contents of the Flash, as security controls can restrict data reads while permitting CRC checks. The CRC function's address granularity prevents it from revealing protected region contents.

Additionally, the CRC function can detect ECC errors without triggering a CPU trap with ECC DED events during CRC calculations resulting in a manageable CRC error.

The CRC-32 parameters include:

- The initial value set in the CRCSEED register with each bit inverted
- No input or output reflection required
- The final XOR value of 0xFFFF_FFFF

For standard CRC-32 calculations, the NVMCRCSEED register is set to 0xFFFF_FFFF. This register is also used for multi-segment calculations to indicate the current CRC state. Flash data are processed using little-endian byte ordering.

6.3.4.1. Usage

The user programs the start and end addresses of the Flash memory to be CRC'ed into NVMCRCST and NVMCRCEND, respectively. The CRC module operates on Flash page boundaries (4 KB). The user then sets the ON bit to enable the CRC module and the START bit to start the CRC calculation. When the CRC sequence is complete, the calculated CRC data is available in the NVMCRCDATA register, and the NVMCRCIF flag will be set. To begin another CRC calculation, the ON bit must be cleared and then asserted after the NVMCRCST and NVMCRCEND registers are loaded with the desired start and end addresses.

6.4. Flash Dual Partition

The dual partition feature is implemented in this version of Flash program memory where the Flash memory can be divided into equal halves and each panel can function as two different address spaces.

6.4.1. Architectural Overview

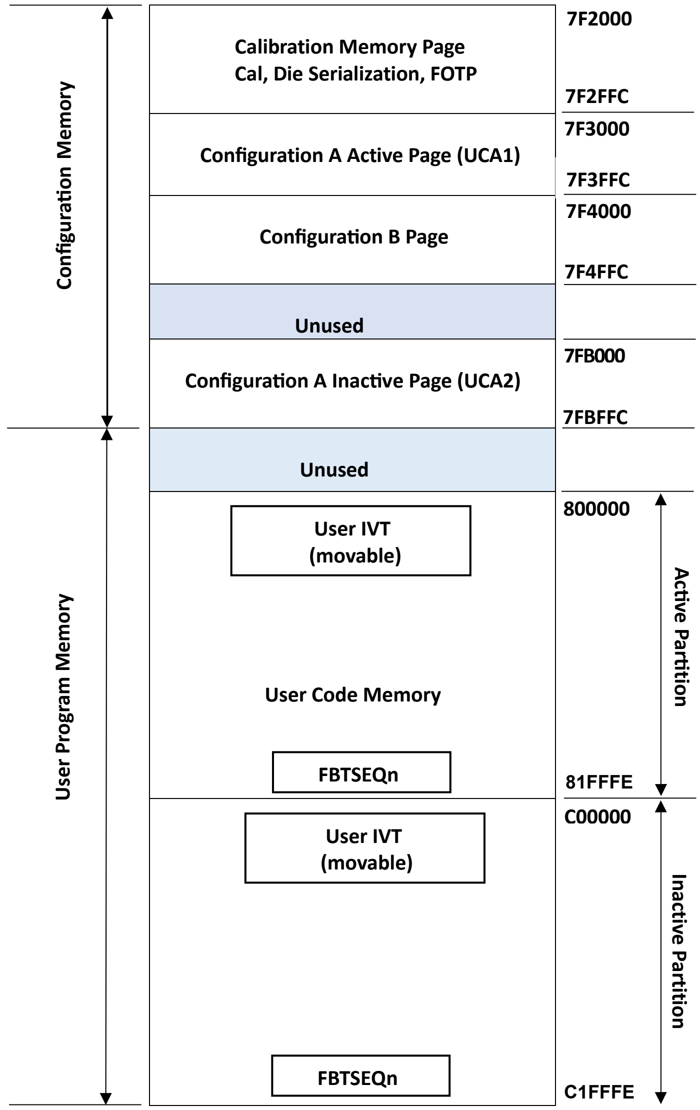
dsPIC33A devices have an 8K x 140-bit program memory address space. The program memory address space in dsPIC devices is divided into two halves, referred to as the user memory space and the configuration memory space. The user memory space can be further divided into two partitions, referred to as the active partition and inactive partition.

The active partition base address is 0x80_0000, and the inactive partition base address is 0xC0_0000. Figure 6-2 shows a memory map when the device is in Dual Partition mode.

For devices with a dual partition feature, when operating in Dual Boot mode, there are two UCA pages: UCA1 for partition 1 and UCA2 for partition 2. The UCA1 base address is 0x7F3000, and the UCA2 base address is 0x7FB000.

The calibration and UCB configuration are shared by the two Flash partitions in Dual Boot mode.

Figure 6-2. dsPIC33A Dual Partition Memory Map



Notes:

1. Memory areas are not shown to scale.
2. Refer to the "Memory Organization" chapter of the specific device data sheet for the exact memory range of the device.

6.4.2. Configuring the Dual Partition

In the Dual Partition modes, two independent applications may be programmed into the device, one to each of two Flash memory partitions, known as Partition 1 and Partition 2. When the device is initialized, one of these is dynamically mapped to the active partition and executed. The other is mapped to the inactive partition, where it remains available to program memory operations. The assignment of a partition to the active or inactive partition is determined automatically by a code signature, known as the Boot Sequence Number (BTSEQn). The code partitions may also be swapped between active and inactive partitions during run time under software control.

6.4.2.1. Selecting the Dual Boot Mode

The Dual Partition Program Memory mode is selected by programming the BTMODE [1:0] bits in the FBOOT Configuration Word, which is a part of the UCB configuration area. When a device is first programmed via In-Circuit Serial Programming™ (ICSP™), the programmer should program FBOOT to correctly set the device Flash Partition mode. Note that it is not possible to reprogram FBOOT at run time using Run-Time Self-Programming (RTSP). The FBOOT bits must be configured in ICSP mode by a programmer.

The Operating mode is always established after (any) Reset and is immediately followed by the Flash Fuse read/load sequence. Four modes of operations are possible depending on the FBOOT value.

Table 6-2. Flash Boot Mode Select

Operating Mode	Boot Mode Value
Reserved	0b00
Protected Dual Boot	0b01
Dual Boot	0b10
Single Boot (default value)	0b11

6.4.2.1.1. Protected Dual Boot Mode

- Two separate bootable address spaces
- Partition 1 is never allowed to be page erased or programmed.
- Partition 1 is write-protected when located in an inactive address space (other than for Chip erase).
- The user may erase/program partition 2 but not partition 1.

6.4.2.1.2. Dual Boot Mode

- Two separate bootable address spaces
- Places no access restrictions on active partition.
- Single software provider or multiple trusted software providers; it is assumed no attempted code theft.

6.4.2.1.3. Single Boot Mode

- One bootable address space
- The entire user program memory is mapped as a continuous memory space, ranging from 800000h to the upper boundary of implemented Flash memory.

6.4.2.2. Selecting the Dual Boot Mode

In Dual Partition modes, there are two methods of determining which partition will be mapped to the active partition and executed: the Boot Sequence Number and the BOOTSWP instruction. The P2ACTIV bit (NVMCON [10]) can be used to determine which physical partition is the active partition. If P2ACTIV = 1, Partition 2 is active; if P2ACTIV = 0, Partition 1 is active.

The BOOTSWP instruction is used to swap active and inactive partitions without a device Reset.

6.4.2.2.1. Configuring Active/Inactive Partitions

When configured for Dual Boot mode after a Reset, the NVM Controller reads the boot sequence numbers contained in the BTSEQn words of each user program partition. It compares the sequence numbers and maps the partition with the lowest sequence number into the active space (0x80_0000) and maps the other partition into the inactive space (0xC0_0000). The current mapping is indicated by the NVMCON.P2ACTIV bit.

The Boot Sequence Number (BTSEQn) is a 12-bit value that is used for automatically determining the active partition upon device Reset. For dsPIC33A devices, BTSEQn is located at the last 128-bit Flash word of each user program partition.

The Boot Sequence Number is stored in two parts: the actual value in the bit field BTSEQn[11:0], and the one’s complement of the value in the IBTSEQn bit field [23:12]. When the Boot Sequence Number is read upon a device Reset, the values of BTSEQn and IBTSEQn are automatically compared. A Boot Sequence Number word is considered to be invalid if either:

- BTSEQn [11:0] ≠ ~BTSEQn [23:12]

OR

- The Boot Sequence Number word read results in an ECC DED error.

BTSEQ [127:24] is reserved and must read as all 0's.

The complement value is not automatically created by hardware, nor is it verified by the hardware upon programming. The application must calculate and program the appropriate value.

Figure 6-3. Boot Sequence Number

BTSEQ active partition address = `(_PROGRAM BASE_ + _PROGRAM_LENGTH - 0x10UL)`

Variant	(Calculation)	Address
256K	0x801000 + 0x3F000 - 0x10	0x83FFF0
128K	0x801000 + 0xF000 - 0x10	0x80FFF0

Notes:

- An unprogrammed word will result in an invalid Boot Sequence Number.
- If a BTSEQn word of a partition is not valid, the Boot Sequence Number for that partition is assigned the value 0xFFF for evaluation. This is the highest possible number.
- Each partition should have a unique Boot Sequence Number. If the Boot Sequence Numbers are equal, Partition 1 is mapped into the active space.

The partitions can be prepared to be swapped during run time by reprogramming the Boot Sequence Number of the inactive partition to have a lower value than the Boot Sequence Number of the active partition. When a Reset is executed, the partition that has the lower value will become active. This method can be used when the inactive partition has been updated and needs to be mapped to the active partition after a Reset.

6.4.2.2.2. BOOTSWP Instruction

The BOOTSWP instruction supports the LiveUpdate feature, by allowing code segments to be swapped between the active and inactive partitions without the need for a device Reset. A partition swap using the BOOTSWP instruction is referred to as a soft swap. To execute a BOOTSWP instruction, the Configuration bit FICD.NOBTSWP must be enabled. If a BOOTSWP instruction is attempted with NOBTSWP set, it will result in a two-cycle NOP instruction.

The BOOTSWP instruction must always be followed by a single-word instruction that writes the PC (e.g., GOTO W, CALL W or BRA W); the target of the instruction must be at an address within 32 Kbytes of the current address. Upon execution, the active and inactive partitions trade places, and the PC vectors to the location specified by the GOTO instruction in the newly active partition.

Note: If the BOOTSWP instruction is executed from within a function that has created a new stack frame using the LNK instruction, a CALL must be used following the BOOTSWP rather than a GOTO; otherwise, the device will generate a stack error trap.

After the execution of the BOOTSWP instruction, the SFTSWP bit (NVMCON [11]) is set. This bit indicates to the firmware that the BOOTSWP instruction has been executed correctly and that the current active partition was entered via BOOTSWP rather than via a device Reset. Status bit P2ACTIV (NVMCON [10]) can also be read to verify which partition is active. It is important to note that after the partition swap, all peripherals and interrupts which were previously enabled remain enabled. Additionally, the RAM and stack maintain their states after the swap. It is highly recommended that applications using soft swaps jump to a routine that re-initializes the device to ensure the application continues to run as expected.

Different from the previous dsPIC33A core devices, the BOOTSWP operation no longer needs a software lock for it to work. An enabled BOOTSWP instruction must perform a validation operation on the inactive panel. This is achieved using the inactive panel BTSEQ value and is intended to prevent an inadvertent swap into an unprogrammed or corrupted panel because of erroneous code execution.

Note: It is important to note that the BTSEQ value should be a valid sequence for the boot swap to happen. It is recommended to use the stored sequence number from the Flash location itself.

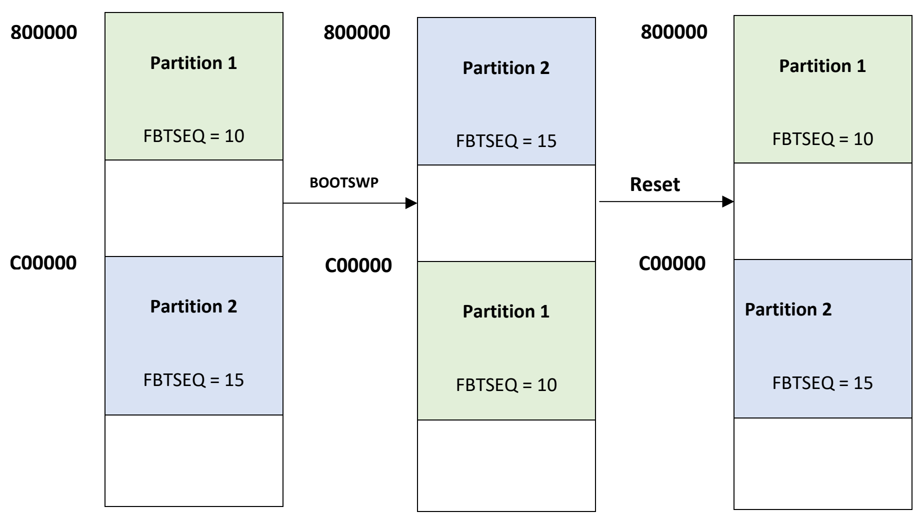
Panel Swap Code Sequence

For a successful BTSEQ validation, a prior instruction (to BOOTSWP) is responsible for reading and loading the inactive panel BTSEQ number into a W-reg. The BOOTSWP instruction must operate on the same register to validate the value read.

The boot sequence number is a 12-bit unsigned value. The BTSEQ word format consists of two 12-bit values with BTSEQ [11:0] containing the true sequence number, and BTSEQ [23:12] containing its complement as shown in Figure 6-4. BTSEQ [127:24] is reserved and must read as all 0’s.

Example 6-1 shows a panel swap example code.

Figure 6-4. BOOTSWP Instruction



Example 6-1. Panel Swap Code Example

```
// Boot sequence number addresses
// BTSEQ is the first word of the last row
#define BTSEQ_ACTIVE_ADDR           (__PROGRAM_BASE + __PROGRAM_LENGTH - 0x10UL)
#define BTSEQ_INACTIVE_ADDR         (0x400000UL | BTSEQ_ACTIVE_ADDR)

// Inactive partition reset vector address
#define INACTIVE_RESET_VECTOR_ADDR      (0x400000UL | __RESET_BASE)

// BTSEQ - Partition 1 boot sequence number
#define BTSEQ_1       0x10
uint32_t __attribute__((space(prog), address(BTSEQ_ACTIVE_ADDR))) btseq_1 =
(~(uint16_t)BTSEQ_1 << 12) | (BTSEQ_1);

// BTSEQ - Partition 2 boot sequence number
#define BTSEQ_2       0x20
uint32_t __attribute__((space(prog), address(BTSEQ_INACTIVE_ADDR))) btseq_2 =
(~(uint16_t)BTSEQ_2 << 12) | (BTSEQ_2);

void __attribute__((address(__PROGRAM_BASE), keep, noinline)) partition_swap(void)
{
  asm volatile("\n    mov.sl      #%0, w1"
               "\n    mov.l       [w1], w1"    // w1 = Inactive Partition FBTSEQ value
(first word of last row)
               "\n    mov.sl      #%1, w0"
               "\n    mov.l       [w0], w0"    // w0 = Inactive Partition reset vector
destination (__reset label on Inactive Partition)
               "\n    bootswp     w1"

              // must use CALL instruction since partition_swap() creates a new stack frame
              // if the function does not create a new stack frame using the LNK
instruction,
```

```
                // a GOTO instruction should be used instead of the CALL instruction
                "\n    call        w0"
                : /* no outputs */ : "i"(BTSEQ_INACTIVE_ADDR),
"i"(INACTIVE_RESET_VECTOR_ADDR) : "w0", "w1");
}
```

If the BOOTSWP is enabled and the BTSEQ validation succeeds, BOOTSWP will be successful, and it will result in a panel swap and SR.Z = 1 and NVMCON.SOFTSWAP = 1.

If BOOTSWP is enabled but the BTSEQ validation fails, BOOTSWP will be unsuccessful. To indicate this result to subsequent code, BOOTSWP will drive SR.Z = 0. NVMCON.SOFTWAP will also not be set.

If BOOTSWP is not enabled but execution is attempted, no error will be flagged, and the instruction will execute as if a two-cycle NOP. Execution will then continue as normal (from the same panel).

The BOOTSWP instruction will always be fetched from the original active panel mapping. The instruction immediately following BOOTSWP may be any instruction other than another BOOTSWP and will be fetched from either the new or current active panel, depending upon the success or failure of the BOOTSWP operation.

Successful BOOTSWP: The CPU will attempt to fetch the instruction following BOOTSWP but will be stalled by the PBU. The PBU will then invalidate the cache and ISB. When complete, the original instruction fetch will be allowed to proceed but from the new active panel.

Unsuccessful BOOTSWP: The instruction following BOOTSWP will be fetched and executed as normal from the current active panel.

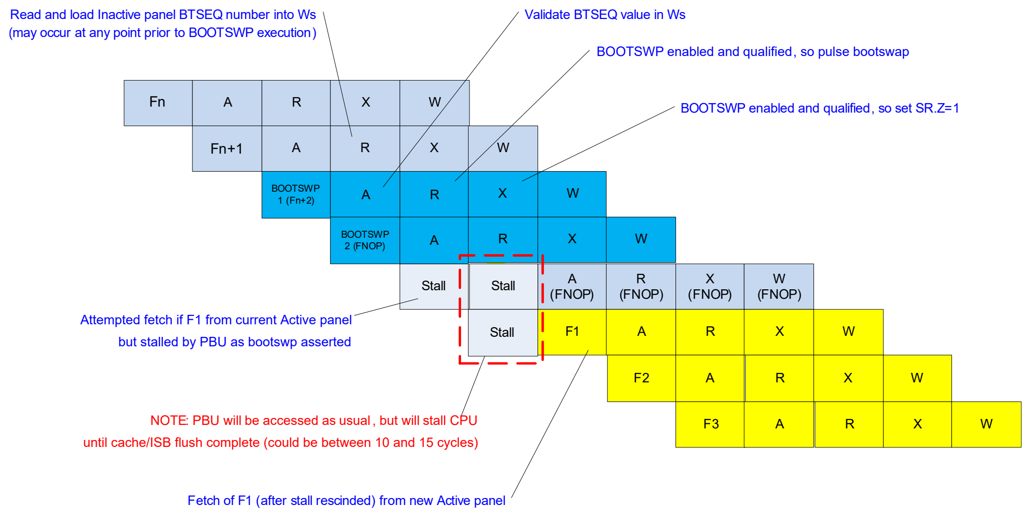
Should the user need to switch back and forth between panels at any point, code located at label (see Example 6-1) should test SR.Z or the NVMCON.SOFTSWAP status bit to determine whether or not the BOOTSWP was successful, then take the appropriate action. For applications where a one way swap is required, code at label”= in the current active panel can assume an unsuccessful panel swap and take appropriate action. Code at the same address within the new active panel can jump directly to the application code, knowing that the panel swap must have succeeded.

Notes:

1. Successful execution of the BOOTSWP instruction will not reload the configuration registers, so those of the original active panel will be persistent.
2. It is the responsibility of the user software to accommodate an unsuccessful operation.
3. Other than providing a mechanism to swap the panel order within address space, we do not facilitate the panel swap in any other way within the device. The user is wholly responsible to guarantee application data coherency, timing, recovery, etc.

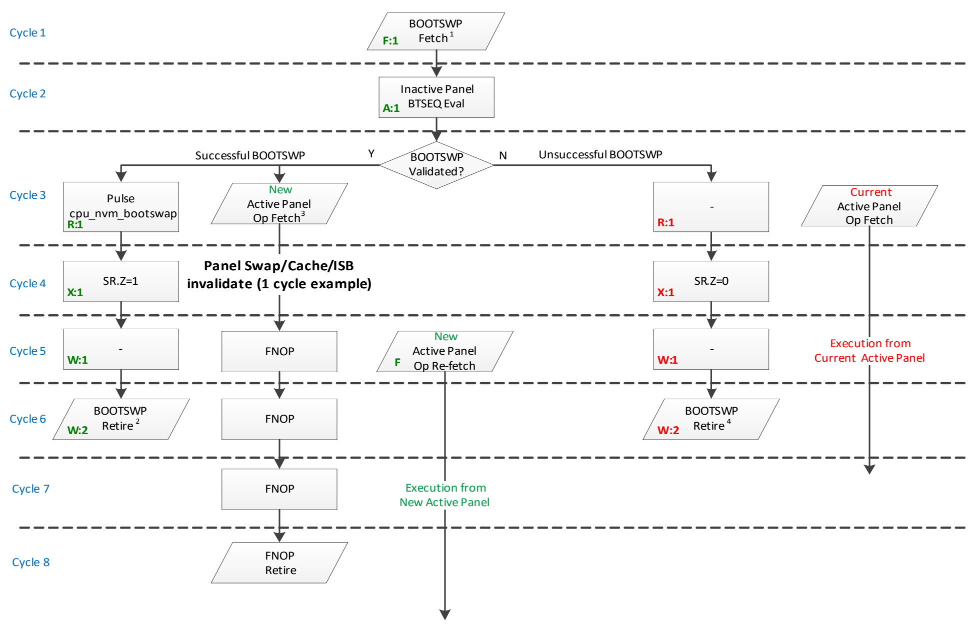
Figure 6-5 shows a BOOTSWP instruction flow diagram.

Figure 6-5. BOOTSWP (Successful) Instruction Flow



[Figure 6-6](#) shows a panel swap sequence flow diagram

Figure 6-6. Panel Swap Sequence Flow Diagram



BOOTSWP and Exceptions

Exceptions are not inhibited by the BOOTSWP instruction and will be handled in the same manner as they are for any two-cycle instruction. If an exception request be asserted at the CPU during the BOOTSWP (A stage), the instruction will be killed during exception processing. If the exception occurs after the BOOTSWP (A stage), the instruction will be allowed to complete during the first cycles of exception processing.

Exception During Successful BOOTSWP Operation

If timing of an exception during BOOTSWP execution meets that outlined in BOOTSWP Instruction, and the BOOTSWP criteria are met, the panel swap will be completed by the time the interrupt or trap handler vector fetch commences (during BOOTSWP W stage). Exception processing will, therefore, vector into the newly active panel, and the handler will return to the address following the BOOTSWP in that panel.

Exception During Unsuccessful BOOTSWP Operation

If timing of an exception during BOOTSWP execution meets that outlined in BOOTSWP Instruction, and the BOOTSWP criteria are not met, the panel swap will not occur. Exception processing will, therefore, vector into the current active panel, and the handler will return to the address following the BOOTSWP in that panel as normal.

BOOTSWP Trace

The BOOTSWP instruction trace is supported by treating the instruction as if it were a conditional branch with respect to the trace interface. A successful BOOTSWP will be communicated as a taken branch to trace. The branch target address will always be the instruction immediately following BOOTSWP (i.e., PC+4 because BOOTSWP is a 32-bit opcode). An unsuccessful BOOTSWP will be communicated as a not-taken branch to trace.

6.4.3. Programming Algorithms

6.4.3.1. Flash Page Erase (Active Partition)

Page erase is necessary before programming the Flash memory. A page erase will ensure that the old contents of the page are erased, and it is programmed with all 1's before programming with new content. Prior to a page erase operation, the target page is selected by writing an address within the page into the Write Address register, NVMADR. A page erase will erase eight rows of data concurrently.

6.4.3.1.1. User Page Erase Sequence

Executing a page erase sequence typically includes the following user code steps, subject to the constraints externally imposed by the Security module:

1. Load NVMADR with an address within the page to be erased.
2. Configure NVMCON to page erase:
   a. WREN = 1
   b. PROGOP = Page erase (0x3)
3. Set the WR bit.
4. The program sequence is completed and the WR bit is cleared by hardware.
5. Clear WREN bit.
6. Test the WRERR bit to ensure the erase sequence is completed successfully.

6.4.3.2. Word Programming (Active Partition)

The smallest block of data that can be programmed by the user is one Flash word (128-bits+ECC). The data to be word programmed must be loaded into NVMDATAx before the word programming sequence is initiated. For the word programming PROGOP, the Flash word at the location pointed to by NVMADR is programmed. Note that NVMADR is a word address, so NVMADR [3:0] is defined as 0b0000 to force Flash word alignment of the operation.

To keep the contents secure, the NVMDATAx register is cleared upon completion of any Flash write operation.

6.4.3.2.1. User Word Programming Sequence

A word programming sequence typically includes the following steps, subject to the constraints externally imposed by the Security module:

1. Write data to be programmed to NVMDATAx.
2. Load NVMADR with the Flash address to be programmed.
3. Configure NVMCON to word program:
   a. WREN = 1
   b. PROGOP = Word program (0x1)

4. Set the WR bit.
5. The program sequence is completed and the WR bit is cleared by hardware.
6. Clear WREN bit.
7. Test the WRERR bit to ensure the program sequence is completed successfully.

Note: Writes to NVMDATAx registers are inhibited while WR = 1.

6.4.3.3. Row Programming (Active Partition)

In dsPIC33A devices, row programming is performed directly from a buffer space in data RAM. The location of the RAM buffer is determined by the NVMSRCADR register(s), which are loaded with the data RAM address containing the first word of program data to be written. Prior to performing the program operation, the buffer space in RAM must be loaded with the row of data to be programmed.

One row consists of 32 Flash words (128-bit+ECC). Row programming starts at the lowest address in the row and proceeds up to the highest address in the row. The module is not specified to operate when the operation is somewhere in the middle of the row and then wraps around.

The row program command automatically sequences through a row worth of write data stored (by the user) in local memory. This module implements row programming operations as a sequence of 32, 140-bit word programming sequences.

During a Row Programming operation, the Flash controller becomes the bus initiator and system RAM is used to store the row programming data. The user loads the system RAM prior to the program operation. The user loads the NVMSRCADR address register with the location of the data in RAM. This address is word aligned to 32-bit RAM addresses. The NVMADR address must be aligned to a Flash word address boundary where address [3:0] = 0b0000. During row programming, the NVMADR address may point to any Flash word within the Flash row.

6.4.3.3.1. BMX Bus Error During Row Programming

If the NVMSRCADR value is invalid and the NVM is performing a row programming operation, the BMX (Bus Matrix) will not be able to connect to the RAM containing the needed data. The bus matrix will generate an error if it cannot access the address specified by the NVMSRCADR register. This error triggers the NVM to terminate the row programming operation.

User Row Programming Sequence

A row programming sequence typically includes the following steps, subject to the constraints externally imposed by the Security module:

1. Load NVMSRCADR with address of data in device RAM.
2. Load NVMADR with the Flash address to be programmed.
3. Configure NVMCON to Row program:
   a. WREN = 1
   b. PROGOP = Row Program (0x2)
4. Set the WR bit.
5. The program sequence is completed and the WR bit is cleared by hardware.
6. Clear WREN bit to avoid accidental writes.
7. Test the WRERR bit to ensure the program sequence is completed successfully.

Note: Writes to RAM used during bus mastered row programming cannot be protected. Consequently, the user must be aware that inadvertent corruption of this area of RAM is possible during programming.

6.4.3.4. Inactive Panel Erase

An inactive panel erase erases the entire inactive partition including the UCA1 and UCA2 pages, depending on which partition is mapped on to the inactive space.

6.4.3.4.1. Inactive Panel Erase Sequence

An inactive panel erase sequence typically includes the following steps, subject to the constraints externally imposed by the Security module:

1. Configure NVMCON to Word program:
   a. WREN = 1
   b. PROGOP = Inactive panel erase (0x4)
2. Set the WR bit.
3. The program sequence is completed and the WR bit is cleared by hardware.
4. Clear WREN bit.
5. Test the WRERR bit to ensure the program sequence is completed successfully.

6.4.3.5. Programming the Entire Inactive Partition (Dual Partition Modes)

To entirely update the code in the inactive partition:

1. Erase the inactive partition. Do an inactive panel erase using the steps described in Inactive Panel Erase Sequence.
2. Write each page of the Inactive Partition using page writes, as described in User Word Programming Sequence.
3. Verify the written data. One suggested method is to perform a CRC on the data to be written and verify the CRC value on the full partition to ensure the data were written correctly.

6.4.3.6. Updating the Active Partition Using a Bootloader

1. Erase and program the entire inactive partition as described in Programming the Entire Inactive Partition (Dual Partition Modes).
2. Read the FBTSEQ Configuration register of the active partition.
3. Decrement the value by one and write to FBTSEQ of the inactive partition.
4. Force a partition swap:
   a. If CPU stalls are not a concern, perform a device Reset. Since the inactive partition has a lower Boot Sequence Number, it will become the active partition after the Reset.
   b. If a CPU stall is not acceptable, execute the BOOTSWP instruction.

6.5. Application Example

Example 6-2. Word-Write

```
long DataWord1;
long DataWord2;
long DataWord3;
long DataWord4;

long TargetWriteAddress;

 NVMCONbits.NVMOP = 1;
 NVMCONbits.WREN = 1;
 NVMADR =  TargetWriteAddress ;
 NVMDATA0 =  DataWord1;
 NVMDATA1 =  DataWord2;
 NVMDATA2 =  DataWord3;
 NVMDATA3 =  DataWord4;

 NVMCONbits.WR = 1;
 while (NVMCONbits.WR == 1);
```

Example 6-3. CRC Checksum Calculation

```
    long StartAddress;
    long StartAddress;
    long EndAddress;
    int InitialValue;
    int DelayValue;
    int IdleValue;

    NVMCRCCONbits.CRCIDL = IdleValue;
    NVMCRCCONbits.DELAY = DelayValue;
    NVMCRCCONbits.ON = 1;

     NVMCRCST = StartAddress;
     NVMCRCEND = EndAddress;

     NVMCRCSEED = InitialValue;

     NVMCRCCONbits.START = 1;

    while(IFS0bits.NVMCRCIF == 0);
```

7. Configuration Bits

In dsPIC33AK256MPS306 family devices, the Configuration Words are implemented as volatile memory. This means that configuration data will be loaded into volatile memory (from the Flash Configuration Words) each time the device is powered up. Their specific locations are shown in Table 7-2. The configuration data are automatically loaded from the Flash Configuration Words to the proper Configuration Shadow registers during device Resets.

In the dsPIC33AK256MPS306 family, the configuration and calibration data is duplicated in Flash for improved robustness. If a double ECC error is detected during the start-up sequence, the device will restart the loading of the calibration and configuration from the alternate set in Flash.

To maintain the integrity of the stored configuration values, logic performs ongoing bit value checks. All device configuration bits are implemented as a complementary set of register bits.

Table 7-1. Configuration Regions

Region	Address Range
CFGA1	0x7F3000 - 0x7F3800
CFGB	0x7F4000 - 0x7F4800
CFGA2	0x7FB000 - 0x7FB800

Table 7-2. dsPIC33AK256MPS306 Configuration Addresses

Register Name	Address	Configuration Region
FCP	0x7F3000	CFGA1
FICD	0x7F3010	CFGA1
FDEVOPT	0x7F3020	CFGA1
FWDT	0x7F3030	CFGA1
FPWRM	0x7F3040	CFGA1
FPR0CTRL	0x7F4000	CFGB
FPR0ST	0x7F4004	CFGB
FPR0END	0x7F4008	CFGB
FPR1CTRL	0x7F4010	CFGB
FPR1ST	0x7F4014	CFGB
FPR1END	0x7F4018	CFGB
FPR2CTRL	0x7F4020	CFGB
FPR2ST	0x7F4024	CFGB
FPR2END	0x7F4028	CFGB
FPR3CTRL	0x7F4030	CFGB
FPR3ST	0x7F4034	CFGB
FPR3END	0x7F4038	CFGB
FPR4CTRL	0x7F4040	CFGB
FPR4ST	0x7F4044	CFGB
FPR4END	0x7F4048	CFGB
FPR5CTRL	0x7F4050	CFGB
FPR5ST	0x7F4054	CFGB
FPR5END	0x7F4058	CFGB
FPR6CTRL	0x7F4060	CFGB
FPR6ST	0x7F4064	CFGB
FPR6END	0x7F4068	CFGB
FPR7CTRL	0x7F4070	CFGB

Table 7-2. dsPIC33AK256MPS306 Configuration Addresses (continued)

Register Name	Address	Configuration Region
FPR7ST	0x7F4074	CFGB
FPR7END	0x7F4078	CFGB
FIRT	0x7F4080	CFGB
FSECDBG	0x7F4090	CFGB
FPED	0x7F40A0	CFGB
FEPUCB	0x7F40B0	CFGB
FWPUCB	0x7F40C0	CFGB
FBOOT	0x7F40D0	CFGB
FCP	0x7FB000	CFGA2
FICD	0x7FB010	CFGA2
FDEVOPT	0x7FB020	CFGA2
FWDT	0x7FB030	CFGA2
FPWRM	0x7FB040	CFGA1

Table 7-3. Device ID and Revision Addresses

Register	Address
DEVID	0x7C2000
DEVREV	0x7C2004

Table 7-4. Family Device Identifier

Device	Device ID Value
dsPIC33AK128MPS303	0xB514
dsPIC33AK128MPS305	0xB515
dsPIC33AK128MPS306	0xB516
dsPIC33AK256MPS303	0xB51C
dsPIC33AK256MPS305	0xB51D
dsPIC33AK256MPS306	0xB51E
dsPIC33AK128MPS103	0xB504
dsPIC33AK128MPS105	0xB505
dsPIC33AK128MPS106	0xB506
dsPIC33AK256MPS103	0xB50C
dsPIC33AK256MPS105	0xB50D
dsPIC33AK256MPS106	0xB50E

7.1. Configuration Register Summary

Note: This summary covers CGA and CFGB areas. The backup copies in CFGA Backup and CFGB Backup are identical and not listed here.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7F3000	FCP	31:24								
		23:16								
		15:8								
		7:0					WPUCA[1:0]		$\overline{CRC}$	CP
0x7F3004 ... 0x7F300F	Reserved									
0x7F3010	FICD	31:24								
		23:16								
		15:8	NOBTSWP							
		7:0	BKBUG		JTAGEN					
0x7F3014 ... 0x7F301F	Reserved									
0x7F3020	FDEVOPT	31:24								
		23:16								
		15:8			SPI2PIN					
		7:0		BISTDIS		ALTI2C2	ALTI2C1			
0x7F3024 ... 0x7F302F	Reserved									
0x7F3030	FWDT	31:24								
		23:16							WDTNVMSL	WDTRSTEN
		15:8	WDTEN	WDTWIN[1:0]		RWDTPS[4:0]				
		7:0	RCLKSEL[1:0]		SWDTPS[4:0]					WINDIS
0x7F3034 ... 0x7F303F	Reserved									
0x7F3040	FPWRM	31:24								
		23:16								
		15:8								
		7:0					PWRM[3:0]			
0x7F3044 ... 0x7F3FFF	Reserved									
0x7F4000	FPR0CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4004	FPR0ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]							
		7:0								
0x7F4008	FPR0END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:0]			
		7:0	END[11:0]							
0x7F400C ... 0x7F400F	Reserved									
0x7F4010	FPR1CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS

Configuration Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7F4014	FPR1ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]							
		7:0								
0x7F4018	FPR1END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:0]			
		7:0	END[11:0]							
0x7F401C ... 0x7F401F	Reserved									
0x7F4020	FPR2CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4024	FPR2ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]							
		7:0								
0x7F4028	FPR2END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:0]			
		7:0	END[11:0]							
0x7F402C ... 0x7F402F	Reserved									
0x7F4030	FPR3CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4034	FPR3ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]							
		7:0								
0x7F4038	FPR3END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:0]			
		7:0	END[11:0]							
0x7F403C ... 0x7F403F	Reserved									
0x7F4040	FPR4CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4044	FPR4ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]							
		7:0								
0x7F4048	FPR4END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:0]			
		7:0	END[11:0]							
0x7F404C ... 0x7F404F	Reserved									
0x7F4050	FPR5CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS

Configuration Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7F4054	FPR5ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]							
		7:0								
0x7F4058	FPR5END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:0]			
		7:0	END[11:0]							
0x7F405C ... 0x7F405F	Reserved									
0x7F4060	FPR6CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x7F4064	FPR6ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]							
		7:0								
0x7F4068	FPR6END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:0]			
		7:0	END[11:0]							
0x7F406C ... 0x7F4073	Reserved									
0x7F4074	FPR7ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]							
		7:0								
0x7F4078	FPR7END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:0]			
		7:0	END[11:0]							
0x7F407C ... 0x7F407F	Reserved									
0x7F4080	FIRT	31:24								
		23:16								
		15:8								
		7:0								IRT
0x7F4084 ... 0x7F408F	Reserved									
0x7F4090	FSECDBG	31:24								
		23:16								
		15:8								
		7:0								SECDBG
0x7F4094 ... 0x7F409F	Reserved									
0x7F40A0	FPED	31:24								
		23:16								
		15:8								
		7:0								ICSPPED
0x7F40A4 ... 0x7F40AF	Reserved									

Configuration Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7F40B0	FEPUCB	31:24	FEPUCB[31:24]							
		23:16	FEPUCB[23:16]							
		15:8	FEPUCB[15:8]							
		7:0	FEPUCB[7:0]							
0x7F40B4 ... 0x7F40BF	Reserved									
0x7F40C0	FWPUCB	31:24	FWPUCB[31:24]							
		23:16	FWPUCB[23:16]							
		15:8	FWPUCB[15:8]							
		7:0	FWPUCB[7:0]							
0x7F40C4 ... 0x7F40CF	Reserved									
0x7F40D0	FBOOT	31:24								
		23:16								
		15:8								
		7:0						PROG	BTMODE[1:0]	

7.1.1. FCP Configuration Register

Name: FCP
Offset: 0x7F3000

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					WPUCA[1:0]		$\overline{CRC}$	CP
Access					R/W	R/W	R/W	R/W
Reset					0	1	1	1

Bits 3:2 – WPUCA[1:0] Write Protect bits

Value	Description
11	UCA1 and UCA2 not write-protected
10	Active partition (at reset) UCA space write-protected
01	UCA1 and UCA2 write-protected
00	Reserved

Bit 1 – $\overline{CRC}$ Code Protect User Program CRC bit

Value	Description
1	NVM controller CRC function is disallowed in ICSP mode when code protection is enabled.
0	NVM controller CRC function is allowed in ICSP mode when code protection is enabled.

Bit 0 – CP Code Protect Enable bit

Value	Description
1	Code protection is disabled.
0	Code protection is enabled.

7.1.2. FICD Configuration Register

Name: FICD
Offset: 0x7F3010

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	NOBTSWP							
Access	R							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	BKBUG		JTAGEN					
Access	R		R/W					
Reset	0		0					

Bit 15 – NOBTSWP BOOTSWP instruction disable bit

Value	Description
1	BOOTSWP instruction is disabled.
0	BOOTSWP instruction is enabled.

Bit 7 – BKBUG Background Debugger Enable bit (active low)

Value	Description
1	Background debugger is disabled.
0	Background debugger functions are enabled.

Bit 5 – JTAGEN JTAG Enable bit

Value	Description
1	JTAG port is enabled.
0	JTAG port is disabled.

7.1.3. FDEVOPT Configuration Register

Name: FDEVOPT
Offset: 0x7F3020

Notes:
1. Fixed pin option is only available for packages larger than 48-pins.
2. Refer to the pin diagrams for alternate pin availability.

Legend: r = Reserved bit, R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			SPI2PIN					
Access			r					
Reset			0					

Bit	7	6	5	4	3	2	1	0
		BISTDIS		ALTI2C2	ALTI2C1			
Access		R/W		R/W	R/W			
Reset		1		1	1			

Bit 13 – SPI2PIN SPI 2 Fast I/O Pad Disable bit(1)

Value	Description
1	SPI2 uses PPS (IO Remap) to make connects with device pins.
0	SPI2 uses direct connections with specified device pins.

Bit 6 – BISTDIS Memory BIST Feature Disable bit

Value	Description
1	mBIST on Reset feature disabled.
0	mBIST on Reset feature enabled.

Bit 4 – ALTI2C2 Alternate I2C2 Pin Mapping bit(2)

Value	Description
1	Default location for SCL2/SDA2 pins.
0	Alternate location for SCL2/SDA2 pins (ASCL2/ASDA2).

Bit 3 – ALTI2C1 Alternate I2C1 Pin Mapping bit(2)

Value	Description
1	Default location for SCL1/SDA1 pins.
0	Alternate location for SCL1/SDA1 pins (ASCL1/ASDA1).

7.1.4. FWDT Configuration Register

Name: FWDT
Offset: 0x0x7F3030

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
							WDTNVMSL	WDTRSTEN
Access							R/W	R/W
Reset							1	1

Bit	15	14	13	12	11	10	9	8
	WDTEN	WDTWIN[1:0]		RWDTPS[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	1	0	0	1	0

Bit	7	6	5	4	3	2	1	0
	RCLKSEL[1:0]		SWDTPS[4:0]					WINDIS
Access	R/PO	R/PO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	1	0	1	0	0

Bit 17 – WDTNVMSL WDT Freeze During Programming Enable bit

Value	Description
1	Freeze WDT during programming.
0	WDT operation unaffected during programming.

Bit 16 – WDTRSTEN WDT Reset bit

Value	Description
1	Enables WDT Reset. Run time WDT event will cause a device Reset.
0	Disables WDT Reset. Run time WDT event generates a trap.

Bit 15 – WDTEN Watchdog Timer Enable bit

Value	Description
1	WDT is enabled in hardware.
0	WDT controlled via the ON bit (WDTCONL[15]).

Bits 14:13 – WDTWIN[1:0] Watchdog Timer Window Select bits

Value	Description
11	WDT window is 25% (Timer Count > 11xxx...xxxxx for the timer to be cleared)
10	WDT window is 37.5% (Timer Count > 101xx...xxxxx for the timer to be cleared).
01	WDT window is 50% (Timer Count > 1xxxx...xxxxx for the timer to be cleared).
00	WDT Window is 75% (Timer Count >01xxx...xxxxx for the timer to be cleared).

Bits 12:8 – RWDTPS[4:0] Run Mode Watchdog Timer Period Select bits
Configures the postscaler value for Run Mode Counter bits. Refer to Table 35-2.

Bits 7:6 – RCLKSEL[1:0] Watchdog Timer Clock Select bits

Value	Description
11	LPRC Oscillator
10	BFRC Oscillator
01	Reserved
00	$F_{OSC}/4$

Bits 5:1 – SWDTPS[4:0] Sleep Mode Watchdog Timer Period Select bits
Configures the postscaler value for Sleep Mode Counter bits. Refer to Table 35-2.

Value	Description
11111	Divide by 2 ^ 31 = 2,147,483,648
11110	Divide by 2 ^ 30 = 1,073,741,824
. . .	
00001	Divide by 2 ^ 1 = 2
00000	Divide by 2 ^ 1 = 2

Bit 0 – WINDIS Watchdog Window Disable bit

Value	Description
1	Disables Window mode.
0	Enables Window mode.

7.1.5. FPWRM Configuration Register

Name: FPWRM
Offset: 0x0x7F3040

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					PWRM[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – PWRM[3:0]

Value	Description
1111-1011	Core voltage monitor is disabled.
1010	Core voltage monitor is enabled.
1001-0000	Core voltage monitor is disabled.

7.1.6. FPRxCTRL Configuration Register

Name: FPRxCTRL
Offset: 0x7F4000, 0x7F4010, 0x7F4020, 0x7F4030, 0x7F4040, 0x7F4050, 0x7F4060

Note:
1. Fixed pin option is only available for higher pin packages (48-pin, 64-pin and 80-pin).

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			PSEL[1:0]				RTYPE[1:0]	
Access			R	R			R/W	R/W
Reset			0	0			1	1

Bit	7	6	5	4	3	2	1	0
	CRC	WR	RD	EX			ERAO	RDIS
Access	R	R/W	R/W	R/W			R/W	R/W
Reset	0	1	1	1			1	1

Bits 13:12 – PSEL[1:0] Partition Select bits

Value	Description
11	Both Panels
10	Panel 2
01	Panel 1
00	Reserved

Bits 9:8 – RTYPE[1:0] Region Type bits

Value	Description
11	Firmware Configurable Region
10	One Time Programmable (OTP) Region
01	Immutable Root of Trust (IRT) Region
00	Reserved

Bit 7 – CRC Flash CRC Permission bit

Value	Description
1	Flash CRC calculation permitted
0	Flash CRC calculation not permitted

Bit 6 – WR Write Permission bit

Value	Description
1	Write (program/erase) permitted

Value	Description
0	Write (program/erase) not permitted

Bit 5 – RD Read Permission bit

Value	Description
1	Read (data read) permitted
0	Read (data read) not permitted

Bit 4 – EX Execute Permission bit

Value	Description
1	Execute (instruction fetch) permitted
0	Execute (instruction fetch) not permitted

Bit 1 – ERAO ECC Report Address Only on ECC Error bit

Value	Description
1	ECC error reporting information restricted to address only
0	No ECC error reporting restrictions

Bit 0 – RDIS Region Disable bit

Value	Description
1	Region disabled
0	Region enabled

7.1.7. FPRxST Configuration Register

Name: FPRxST
Offset: 0x7F4004, 0x7F4014, 0x7F4024, 0x7F4034, 0x7F4044, 0x7F4054, 0x7F4064, 0x7F4074

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		START[22:16]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	START[15:12]							
Access	R/W	R/W	R/W	R/W				
Reset	1	1	1	1				

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 22:16 – START[22:16] Start Address Offset MSb bits
Protection region start address offset

Bits 15:12 – START[15:12] Start Address Offset LSb bits
Protection region start address offset

7.1.8. FPRxEND Configuration Register

Name: FPRxEND
Offset: 0x7F4008, 0x7F4018, 0x7F4028, 0x7F4038, 0x7F4048, 0x7F4058, 0x7F4068, 0x7F4078

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		END[22:16]						
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	1	1	1

Bit	15	14	13	12	11	10	9	8
	END[15:12]				END[11:0]			
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	END[11:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bits 22:12 – END[22:12] End Address Offset LSb bits
Protection region end address offset (most significant bits)

Bits 11:0 – END[11:0] End Address Offset LSb bits
Protection region end address offset (least significant bits) fixed value 0xFFF

7.1.9. FIRT Configuration Register

Name: FIRT
Offset: 0x7F4080

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								IRT
Access								R/W
Reset								1

Bit 0 – IRT Immutable Root of Trust Enable bit

Value	Description
1	IRT disabled
0	IRT enabled

7.1.10. FSECDBG Configuration Register

Name: FSECDBG
Offset: 0x7F4090

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								SECDBG
Access								R/W
Reset								1

Bit 0 – SECDBG Secure Debug Mode Enable bit

Value	Description
1	SECDBG enabled
0	SECDBG disabled

7.1.11. FPED Configuration Register

Name: FPED
Offset: 0x7F40A0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								ICSPPED
Access								R/W
Reset								1

Bit 0 – ICSPPED ICSP Program/Erase Disable bit
Note: UCP write-protect must be set for ICSP inhibit to be enabled.

Value	Description
1	ICSP Write Inhibit disabled (ICSP erase/writes are enabled)
0	ICSP Write Inhibit enabled (ICSP erase/writes are disabled)

7.1.12. FEPUCB Configuration Register

Name: FEPUCB
Offset: 0x7F40B0

Bit	31	30	29	28	27	26	25	24
	FEPUCB[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FEPUCB[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FEPUCB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FEPUCB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – FEPUCB[31:0] UCB Erase Protect Register bits
UCB Erase Protect Code = 32’h84C1_F396

7.1.13. FWPUCB Configuration Register

Name: FWPUCB
Offset: 0x7F40C0

Bit	31	30	29	28	27	26	25	24
	FWPUCB[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	FWPUCB[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	FWPUCB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	FWPUCB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – FWPUCB[31:0] UCB Write Protect Register bits
UCB Write Protect Code = 32’h5B9B_12E4

7.1.14. FBOOT Configuration Register

Name: FBOOT
Offset: 0x7F40D0

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						PROG	BTMODE[1:0]	
Access						R/P	R/W	R/W
Reset						1	0	1

Bit 2 – PROG Boot Mode Configuration Word Programmed bit

Value	Description
1	Boot mode configuration word not programmed
0	Boot mode configuration word programmed

Bits 1:0 – BTMODE[1:0] Device Boot Mode Configuration (status) bits

Value	Description
11	Device is in Single Boot (legacy) mode.
10	Device is in Dual Boot mode.
01	Device is in Dual Boot Protected mode.
00	Reserved

7.2. Device Calibration and Identification

The dsPIC33AK256MPS306 devices have two Identification registers near the end of the configuration memory space that store the Device ID (DEVID) and Device Revision (DEVREV). These registers are used to determine the mask, variant, and manufacturing information about the device. These registers are read-only and are shown in DEVREV and DEVID.

7.2.1. Device Revision Register

Name: DEVREV

Legend: R = Read-only bit; r = Reserved bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	Reserved[1:0]							
Access	r	r						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
					DEVREV[3:0]			
Access					R	R	R	R
Reset								

Bits 15:14 – Reserved[1:0] Maintain as '0'

Bits 3:0 – DEVREV[3:0] Device Revision bits

7.2.2. Device ID Register

Name: DEVID

Note:

1. See Table 7-4 for the list of Device Identifier bits.

Legend: R = Read-only bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FAMID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset								

Bit	7	6	5	4	3	2	1	0
	DEV[7:0]							
Access	R	R	R	R	R	R	R	R
Reset								

Bits 15:8 – FAMID[7:0] Device Family Identifier bits

Value	Description
`1010 0111`	dsPIC33AKXXXMPSXXX families

Bits 7:0 – DEV[7:0] Individual Device Identifier bits(1)
See Table 7-4 for the list of Device Identifier bits.

8. Security Module

The security module protects the operation of the device and intellectual property from unauthorized access, use and modification. The following security features are available on the dsPIC33AK256MPS306 family of devices:

- Secure Boot
- Secure Debug
- Immutable Root of Trust (IRT)
- Code Protect
- ICSP Program/Erase Disable (Entire Flash OTP by ICSP Write Inhibit)
- Firmware IP Protection
- Flash Write Protection
- Cryptographic Accelerator

The security features can be categorized into four categories:

Device Locking prevents unauthorized external access via debugger or programmer ICSP interfaces (local attacks). Device locking features include code protection, entire Flash OTP by ICSP write inhibit and secure debug.

The **Immutable Root of Trust** (IRT) partition protects IRT firmware and data for the implementation of secure boot, secure debug, device attestation, and other security functions.

Eight **Configurable Protection Regions** provide flexible user program Flash access control. The protection regions include: IRT partition, immutable device firmware (OTP), firmware IP protection (execute-only memory), Flash write protection, and code Flash partitioning.

Flash Access Control is provided for the executive, user OTP and user configuration Flash spaces.

The Cryptographic Accelerator features the following crypto functions:

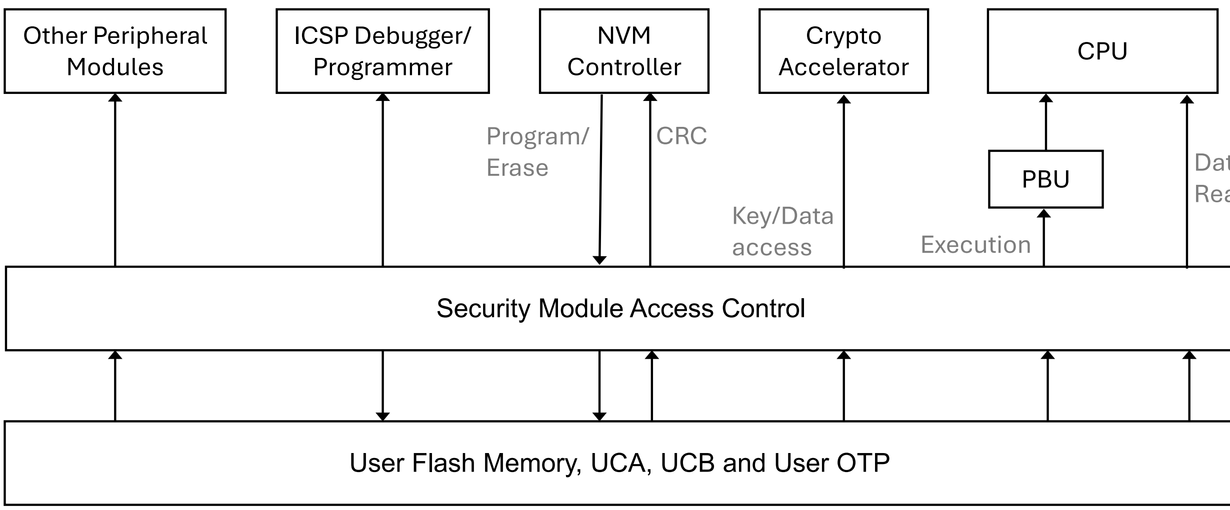
- True Random Number Generator (TRNG) - with NIST 800-90B compliant entropy source and health tests
- AES
  - NIST compliant “Advanced Encryption Standard” (AES), FIPS 197
  - Encryption and Decryption with:
    - Non-chaining: ECB, CTR
    - Chaining: CBC, CFB, OFB
    - Ciphertext stealing: CBC-CS
    - Authentication and confidentiality: CCM, GCM
    - Confidentiality on storage: XTS/XTS-CS
      - Authentication with CMAC (OMAC1) mode
      - Supports 128-, 192- or 256-bits cipher key size
      - Optional: Masking-based countermeasure to improve resistance against SPA and DPA attacks
- Hash
  - FIPS 180-3 compliant HASH functions
  - SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512
  - SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128 and SHAKE-256

  - FIPS-198-1 compliant HMAC for all supported HASH algorithms
- Asymmetric Crypto Accelerator
  - RSA, DSA, DH, ECDH, ECDSA, EdDSA, J-PAKE and SRP
  - RSA, DSA and DH algorithms support up to 4096-bit key sizes
  - Standard ECC curves include: P256, P384, P521, P191, Curve25519 and Ed25519
  - Other ECC curves like Brainpool, Koblitz, Montgomery and Edwards curves are supported via custom parameters
  - Rabin-Miller primality test
  - J-PAKE and SRP-based secure password authentication protocols
  - Optional: Randomization-based countermeasures

8.1. Architectural Overview

A simplified block diagram of the security module is illustrated in Figure 8-1.

Figure 8-1. Security Module Block Diagram



Security module access control is provided for the entire system address space and is based on the address of the access, type of access and the device mode of operation. Access control is enforced in all modes of operation for the CPU, NVM and for all other modules on the peripheral system bus. Accesses can be denied due to a variety of reasons including:

- Code-protect violation
- Flash protection region violation
- Access to reserved space or a Flash space-specific violation (e.g., attempted execution from user configuration space).

An access is allowed only if permitted by all access controls. An access control evaluation is done at the time the Flash memory is accessed. Accesses to the Flash space returned from the instruction cache, prefetch buffer or a NVM read data buffer are not checked.

8.1.1. Access Types

Access control is based on the type of access. There are four main access types:

- Execute (Instruction Fetch)
- Data Read
- Write (Program/Erase)

- CRC

The **Execute Access** type indicates a CPU instruction fetch.

The **Data Read Access** type indicates either a CPU operand or a peripheral system bus data access.

The **Write Access** type indicates either a Flash programming or erase operation. In some instances, access control differentiates between program and erase operations.

The **CRC Access** type indicates an access by CRC hardware integrated into the NVM Controller. The integrated CRC function allows firmware integrity checking without exposing the region contents outside the NVM Controller. This capability can be used for integrity checking of execute-only memory. NVM CRC hardware is limited to calculating a 32-bit CRC over one or more consecutive four Kbyte Flash pages.

8.1.2. Modes of Operation

Access control is based on the device mode of operation. There are three device modes of operation related to security:

- Mission mode
- Debug mode
- ICSP Programming mode

Mission mode is the default device mode of operation. After Reset, code in the user program Flash is executed. After booting, Mission mode also supports code execution from RAM. Execution from RAM is not visible to the security module and does not compromise security module functions.

Debug mode can be entered either at Reset or upon hitting a debug breakpoint. It executes debug executive code stored in a dedicated Flash space separate from user program Flash to support in-circuit debugging.

ICSP Programming mode provides the device access using the external ICSP programming tools.

8.1.3. Configuration

The security module is configured with the Flash Configuration bits and SFRs. The Configuration bits' values are loaded from Flash at Reset and are maintained until the next Reset. Updates to the configuration values in Flash take effect on the next Reset. The Protection Region Descriptor SFRs Reset values are loaded from the Configuration bits as well. These descriptors include a locking capability (modification is disabled). This allows for several configuration options including:

- Dynamic configuration by firmware
- Static configuration by boot firmware
- Permanent configuration using the Flash Configuration bits

Writes to a locked (read-only) Protection Region Descriptor register are ignored.

8.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x02E0	UCPROT	31:24								
		23:16								
		15:8								
		7:0					WPUCAI	WPUCA	WPUCB	EPUCB
0x02E4	IRTCTRL	31:24								
		23:16								
		15:8								
		7:0			IACT	PLCK		DONE	DBG	EAA
0x02E8	IRTSTAT	31:24	STATUS[31:24]							
		23:16	STATUS[23:16]							
		15:8	STATUS[15:8]							
		7:0	STATUS[7:0]							
0x02EC ... 0x02FF	Reserved									
0x0300	PR0CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0304	PR0ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0308	PR0END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x030C	PR0LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0310	PR1CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0314	PR1ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0318	PR1END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x031C	PR1LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0320	PR2CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0324	PR2ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0328	PR2END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x032C	PR2LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0330	PR3CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0334	PR3ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0338	PR3END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x033C	PR3LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0340	PR4CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0344	PR4ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0348	PR4END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x034C	PR4LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0350	PR5CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0354	PR5ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0358	PR5END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x035C	PR5LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0360	PR6CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0364	PR6ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0368	PR6END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x036C	PR6LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0370	PR7CTRL	31:24								
		23:16								
		15:8			PSEL[1:0]				RTYPE[1:0]	
		7:0	CRC	WR	RD	EX			ERAO	RDIS
0x0374	PR7ST	31:24								
		23:16		START[22:16]						
		15:8	START[15:12]				START[11:8]			
		7:0	START[7:0]							
0x0378	PR7END	31:24								
		23:16		END[22:16]						
		15:8	END[15:12]				END[11:8]			
		7:0	END[7:0]							
0x037C	PR7LOCK	31:24	KEY[15:8]							
		23:16	KEY[7:0]							
		15:8								
		7:0							LOCK[1:0]	
0x0380 ... 0x1EBF	Reserved									
0x1EC0	PACCON1	31:24	CM1CONWR	OSCCTRLWR	NVMCONWR	IOIM8CONWR	IOIM7CONWR	IOIM6CONWR	IOIM5CONWR	IOIM4CONWR
		23:16	IOIM3CONWR	IOIM2CONWR	IOIM1CONWR	PCLKCONWR	BMXIRAMHWR	BMXIRAMLWR	IVTCREGWR	IVTBASEWR
		15:8	CM1CONLK	OSCCTRLLK	NVMCONLK	IOIM8CONLK	IOIM7CONLK	IOIM6CONLK	IOIM5CONLK	IOIM4CONLK
		7:0	IOIM3CONLK	IOIM2CONLK	IOIM1CONLK	PCLKCONLK	BMXIRAMHLK	BMXIRAMLLK	IVTCREGLK	IVTBASELK
0x1EC4	PACCON2	31:24	OPAMP3WR	OPAMP2WR	OPAMP1WR	MBISTCONWR			PMDWR	RPCONWR
		23:16	WDTCONWR	CM4RANGEWR	CM4CONWR	CM3RANGEWR	CM3CONWR	CM2RANGEWR	CM2CONWR	CM1RANGEWR
		15:8	OPAMP3LK	OPAMP2LK	OPAMP1LK	MBISTCONLK			PMDLK	RPCONLK
		7:0	WDTCONLK	CM4RANGELK	CM4CONLK	CM3RANGELK	CM3CONLK	CM2RANGELK	CM2CONLK	CM1RANGELK
0x1EC8	PACCON3	31:24								
		23:16							QEI1WR	TRACEWR
		15:8								
		7:0							QEI1LK	TRACELK
0x1ECC ... 0x7A7FFF	Reserved									
0x7A8000	CAMCON	31:24								
		23:16								
		15:8	CAMON		SIDL					
		7:0								

8.2.1. User Configuration Areas Protection Register

Name: UCPROT
Offset: 0x2E0

Legend: x = Loaded from Configuration bits; S = Set Only bit; R = Readable bit

Notes:

1. WPUCA Reset value is based on the WPUCA[1:0] (FCP[3:2]) Configuration bits settings in the UCA area. WPUCA Write Protection bit does not apply in ICSP Programming mode.
2. WPUCB is reset to 1 if the FWPUCB Configuration Word is programmed in the UCB area.
3. EPUCB is reset to 1 if the FEPUCB Configuration Word is programmed in the UCB area.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					WPUCAI	WPUCA	WPUCB	EPUCB
Access					R/S	R/S	R/S	R/S
Reset					x	x	x	x

Bit 3 – WPUCAI UCA Inactive Partition Write-Protect Enable bit
Write 1 to set; Writing 0 has no effect

Value	Description
1	Inactive partition UCA is write-protected (cannot be programmed or erased).
0	Inactive partition UCA is not write-protected.

Bit 2 – WPUCA UCA Write-Protect Enable bit(1)
Write 1 to set; Writing 0 has no effect

Value	Description
1	UCA is write-protected (cannot be programmed or erased).
0	UCA is not write-protected.

Bit 1 – WPUCB UCB Write-Protect Enable bit(2)
Write 1 to set; Writing 0 has no effect

Value	Description
1	UCB is write-protected (cannot be programmed or erased).
0	UCB is not write protected.

Bit 0 – EPUCB UCB Erase-Protect Enable bit[(3)]
Write `1` to set; Writing `0` has no effect

Value	Description
1	UCB is erase-protected (cannot be programmed or erased).
0	UCB is not erase-protected.

8.2.2. IRT Control Register

Name: IRTCTRL
Offset: 0x2E4

Legend: HS = Hardware Settable bit; S = Set Only bit; R = Readable bit

Notes:
1. Register is read-only when PLCK bit is set.
2. If PLCK bit is set, then the access to IRT regions is disabled and the IRTCTRL and IRTSTAT registers are read-only.
3. DBG bit controls debug access to the IRT partition when IRT is enabled.
4. EAA controls the external access via debug and ICSP interfaces when IRT and secure debug are enabled. Setting of EAA bit allows debug and ICSP programmer access; otherwise, these functions are disabled.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			IACT	PLCK		DONE	DBG	EAA
Access			R/HS	R/HS		R/S	R/W	R/W
Reset			0	0		0	0	0

Bit 5 – IACT IRT Interactive Mode bit

Value	Description
1	IRT interactive mode
0	Not IRT interactive mode

Bit 4 – PLCK IRT Partition Lock Status bit [(1,2)]

Value	Description
1	IRT partition is locked.
0	IRT partition is not locked.

Bit 2 – DONE UCA Write-Protect Enable bit
Write `1` to set; Writing `0` has no effect.

Value	Description
1	IRT execution is done.
0	IRT execution is not finished.

Bit 1 – DBG IRT Debug Enable bit [3]
Reset on POR or BOR only.

Value	Description
1	Debug access to IRT partition is allowed.
0	Debug access to IRT partition is disabled.

Bit 0 – EAA External Access (Debugger or Programmer) Enable Bit[4]
Reset on POR or BOR only.

Value	Description
1	External access for debugging, programming and test interfaces is enabled.
0	External access for debugging, programming and test interfaces is disabled.

8.2.3. IRT Status Word Register

Name: IRTSTAT
Offset: 0x2E8

Legend: R = Readable bit; W = Writable bit

Note:

1. Register is read-only when PLCK bit (IRTCTRL[4]) is set.

Bit	31	30	29	28	27	26	25	24
	STATUS[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STATUS[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STATUS[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STATUS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – STATUS[31:0] IRT Status bits[(1)]
Reset only on POR or BOR. IRT is a status word defined in firmware.

8.2.4. Protection Region n Control Register

Name: PRnCTRL
Offset: 0x300, 0x310, 0x320, 0x330, 0x340, 0x350, 0x360, 0x370

Legend: n = Region number; x = Loaded from Configuration bits; W = Writable bit; R = Readable bit

Notes:
1. Register is read-only unless RTYPE[1:0] bits are '11' and LOCK[1:0] bits (PRnLOCK[1:0]) are '11'. Firmware configurable region descriptors are locked (read-only) at Reset and must be unlocked with a write to PRnLOCK before being modified.
2. IRT regions: all access disabled when the PLCK bit (IRTCRTL[4]) is set. For the OTP regions, the region write permissions are disabled, regardless of the WR bit. OTP and IRT are permanent regions.
3. The WR bit is not used for OTP regions; write permissions are always disabled for OTP regions.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			PSEL[1:0]				RTYPE[1:0]	
Access			R	R			R	R
Reset			x	x			x	x

Bit	7	6	5	4	3	2	1	0
	CRC	WR	RD	EX			ERAO	RDIS
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	x	x	x	x			x	x

Bits 13:12 – PSEL[1:0] Partition Select bits

Value	Description
11	Both User Program Partitions
10	User Program Partition 2
01	User Program Partition 1
00	User Data

Bits 9:8 – RTYPE[1:0] Protection Region Type Selection bits (1,2)

Value	Description
11	Firmware Configurable Region
10	One Time Programmable (OTP) Region
01	Immutable Root of Trust (IRT) Region
00	Reserved

Bit 7 – CRC CRC Permission Enable bit

Value	Description
1	NVM Controller CRC is permitted.
0	NVM Controller CRC is disabled.

Bit 6 – WR Write Permission Enable bit[(3)]

Value	Description
1	Write (program/erase) is permitted.
0	Write (program/erase) is disabled.

Bit 5 – RD Read Permission Enable bit

Value	Description
1	Read is permitted.
0	Read is disabled.

Bit 4 – EX Execute Permission Enable bit

Value	Description
1	Execute (instruction fetch) is permitted.
0	Execute (instruction fetch) is disabled.

Bit 1 – ERAO Error Report Address Only bit

Value	Description
1	ECC error reporting information is restricted to address only.
0	No ECC error reporting restrictions

Bit 0 – RDIS Protection Region Disable bit

Value	Description
1	Region is disabled.
0	Region is enabled.

8.2.5. Protection Region n Start Address Offset Register

Name: PRnST
Offset: 0x304, 0x314, 0x324, 0x334, 0x344, 0x354, 0x364, 0x374

Legend: x = Loaded from Configuration bits, R = Readable bit, W= Writable bit

Notes:

1. Register is read-only unless RTYPE[1:0] bits are '11' and LOCK[1:0] bits (PRLOCK[1:0]) are '11'. Firmware configurable region descriptors are locked (read-only) at Reset and must be unlocked with a write to PRLOCK before being modified.
2. Start address offset is a first byte offset from the beginning of the user program Flash (0x800000 address).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		START[22:16]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		x	x	x	x	0	0	0

Bit	15	14	13	12	11	10	9	8
	START[15:12]				START[11:8]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	x	x	x	x	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	START[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 22:16 – START[22:16]

Bits 15:12 – START[15:12] Most Significant bits of the Protection Region First Byte Address Offset[1,2]

Bits 11:8 – START[11:8]

Bits 7:0 – START[7:0] Least Significant bits of the Protection Region First Byte Address Offset. Fixed value 0x000.

8.2.6. Protection Region n End Address Offset Register

Name: PRnEND
Offset: 0x308, 0x318, 0x328, 0x338, 0x348, 0x358, 0x368, 0x378

Legend: x = Loaded from Configuration bits; R = Readable bit; W = Writable bit

Note:

1. Register is read-only unless RTYPE[1:0] bits are '11' and LOCK[1:0] bits (PRLOCK[1:0]) are '11'. Firmware configurable region descriptors are locked (read-only) at Reset and must be unlocked with a write to PRLOCK before being modified.
2. End address offset is a last byte offset from the beginning of the user program Flash (0x800000 address).
3. END value is loaded from Configuration Words.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		END[22:16]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		x	x	x	x	0	0	0

Bit	15	14	13	12	11	10	9	8
	END[15:12]				END[11:8]			
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	x	x	x	x	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	END[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bits 22:16 – END[22:16] Most Significant Bits of the Protection Region Last Byte Address Offset bits[1,2,3]

Bits 15:12 – END[15:12] Most Significant Bits of the Protection Region Last Byte Address Offset bits[1,2]

Bits 11:8 – END[11:8] Least Significant Bits of the Protection Region Last Byte Address Offset bits.
Fixed value 0xFFF.

Bits 7:0 – END[7:0] Least Significant Bits of the Protection Region Last Byte Address Offset bits.
Fixed value 0xFFF.

8.2.7. Protection Region n Lock Control Register

Name: PRnLOCK
Offset: 0x30C, 0x31C, 0x32C, 0x33C, 0x34C, 0x35C, 0x36C, 0x37C

Legend: R = Readable bit; W = Writable bit

Note:

1. The LOCK field does not apply to OTP and IRT region descriptors that cannot be modified by firmware.

Bit	31	30	29	28	27	26	25	24
	KEY[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	KEY[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							LOCK[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 31:16 – KEY[15:0] Write Key bits
Register is read-only if LOCK[1:0] bits are set to '`01`'. Writes to the PRxLOCK registers must be 32 bits with the 0xB737 key field value; otherwise, the write is ignored. The region descriptors are locked (read-only) on Reset to prevent inadvertent writes from corrupting a descriptor. The region descriptor registers are unlocked (writes enabled) by setting LOCK[1:0] bits to '`11`'.

Bits 1:0 – LOCK[1:0] Protection Region Lock Option bits [(1)]
Register is read-only if LOCK[1:0] bits are set to '`01`'. Writes to the PRxLOCK registers must be 32 bits with the 0xB737 key field value; otherwise, the write is ignored. The region descriptors are locked (read-only) on Reset to prevent inadvertent writes from corrupting a descriptor. The region descriptor registers are unlocked (writes enabled) by setting LOCK[1:0] bits to '`11`'.

Value	Description
3	Region descriptor is unlocked (can be modified in SFRs).
2	Reserved
1	Region descriptor is locked and cannot be unlocked until after the next Reset.
0	Region descriptor is locked.

8.2.8. Peripheral Access Control Register 1

Name: PACCON1
Offset: 0x1EC0

Legend: R = Readable bit, W = Writable bit, S = One Way Settable bit

Bit	31	30	29	28	27	26	25	24
	CM1CONWR	OSCCTRLWR	NVMCONWR	IOIM8CONWR	IOIM7CONWR	IOIM6CONWR	IOIM5CONWR	IOIM4CONWR
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	IOIM3CONWR	IOIM2CONWR	IOIM1CONWR	PCLKCONWR	BMXIRAMHWR	BMXIRAMLWR	IVTCREGWR	IVTBASEWR
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	CM1CONLK	OSCCTRLLK	NVMCONLK	IOIM8CONLK	IOIM7CONLK	IOIM6CONLK	IOIM5CONLK	IOIM4CONLK
Access	S/R	S/R	S/R	S/R	S/R	S/R	S/R	S/R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	IOIM3CONLK	IOIM2CONLK	IOIM1CONLK	PCLKCONLK	BMXIRAMHLK	BMXIRAMLLK	IVTCREGLK	IVTBASELK
Access	S/R	S/R	S/R	S/R	S/R	S/R	S/R	S/R
Reset	0	0	0	0	0	0	0	0

Bit 31 – CM1CONWR Clock Monitor 1 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 30 – OSCCTRLWR Oscillator Control Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 29 – NVMCONWR Non-Volatile Memory Write Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 28 – IOIM8CONWR IOIM 8 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 27 – IOIM7CONWR IOIM 7 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 26 – IOIM6CONWR IOIM 6 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 25 – IOIM5CONWR IOIM 5 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 24 – IOIM4CONWR IOIM 4 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 23 – IOIM3CONWR IOIM 3 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 22 – IOIM2CONWR IOIM 2 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 21 – IOIM1CONWR IOIM 1 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 20 – PCLKCONWR PWM Clock Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 19 – BMXIRAMHWR BMX Instruction RAM High Address Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 18 – BMXIRAMLWR BMX Instruction RAM Low Address Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 17 – IVTCREGWR Interrupt Vector Collapse Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 16 – IVTBASEWR Interrupt Vector Base Address Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 15 – CM1CONLK Clock Monitor 4 Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 14 – OSCCTRLLK Oscillator Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 13 – NVMCONLK NVM Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 12 – IOIM8CONLK IO Integrity Monitor Control Register 8 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 11 – IOIM7CONLK IO Integrity Monitor Control Register 7 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 10 – IOIM6CONLK IO Integrity Monitor Control Register 6 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 9 – IOIM5CONLK IO Integrity Monitor Control Register 5 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 8 – IOIM4CONLK IO Integrity Monitor Control Register 4 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 7 – IOIM3CONLK IO Integrity Monitor Control Register 3 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 6 – IOIM2CONLK IO Integrity Monitor Control Register 2 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 5 – IOIM1CONLK IO Integrity Monitor Control Register 1 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 4 – PCLKCONLK PWM Clock Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 3 – BMXIRAMHLK BMX Instruction RAM High Address Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 2 – BMXIRAMLLK BMX Instruction RAM Low Address Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 1 – IVTCREGLK Interrupt Vector Collapse Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 0 – IVTBASELK Interrupt Vector Base Address Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

8.2.9. Peripheral Access Control Register 2

Name: PACCON2
Offset: 0x1EC4

Legend: R = Readable bit, W = Writable bit, S = One Way Settable bit

Bit	31	30	29	28	27	26	25	24
	OPAMP3WR	OPAMP2WR	OPAMP1WR	MBISTCONWR			PMDWR	RPCONWR
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	1	1	1	1			1	1

Bit	23	22	21	20	19	18	17	16
	WDTCONWR	CM4RANGEWR	CM4CONWR	CM3RANGEWR	CM3CONWR	CM2RANGEWR	CM2CONWR	CM1RANGEWR
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	OPAMP3LK	OPAMP2LK	OPAMP1LK	MBISTCONLK			PMDLK	RPCONLK
Access	S/R	S/R	S/R	S/R			S/R	S/R
Reset	0	0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	WDTCONLK	CM4RANGELK	CM4CONLK	CM3RANGELK	CM3CONLK	CM2RANGELK	CM2CONLK	CM1RANGELK
Access	S/R	S/R	S/R	S/R	S/R	S/R	S/R	S/R
Reset	0	0	0	0	0	0	0	0

Bit 31 – OPAMP3WR Op Amp 3 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 30 – OPAMP2WR Op Amp 2 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 29 – OPAMP1WR Op Amp 1 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 28 – MBISTCONWR MBIST Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 25 – PMDWR Peripheral Modules Disable Registers Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 24 – RPCONWR Peripheral Remapping Control Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 23 – WDTCONWR Watchdog Timer Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 22 – CM4RANGEWR Clock Monitor 4 Range (CM4WINPR - CM4LWARN) Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 21 – CM4CONWR Clock Monitor 4 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 20 – CM3RANGEWR Clock Monitor 3 Range (CM3WINPR - CM3LWARN) Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 19 – CM3CONWR Clock Monitor 3 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 18 – CM2RANGEWR Clock Monitor 2 Range (CM2WINPR - CM2LWARN) Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 17 – CM2CONWR Clock Monitor 2 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 16 – CM1RANGEWR Clock Monitor 1 Range (CM1WINPR - CM1LWARN) Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 15 – OPAMP3LK Op Amp 3 Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 14 – OPAMP2LK Op Amp 3 Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 13 – OPAMP1LK Op Amp 1 Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 12 – MBISTCONLK MBIST Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 9 – PMDLK Peripheral Modules Disable Registers Lock bit

Value	Description
1	Registers are write locked.
0	Registers are not write locked.

Bit 8 – RPCONLK Peripheral Remapping Configuration Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 7 – WDTCONLK Watchdog Timer Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 6 – CM4RANGELK Clock Monitor 4 Range Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 5 – CM4CONLK Clock Monitor 4 Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 4 – CM3RANGELK Clock Monitor 3 Range Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 3 – CM3CONLK Clock Monitor 3 Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 2 – CM2RANGELK Clock Monitor 2 Range (CM2WINPR - CM2LWARN) Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 1 – CM2CONLK Clock Monitor 2 Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 0 – CM1RANGELK Clock Monitor 1 Range (CM1WINPR - CM1LWARN) Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

8.2.10. Peripheral Access Control Register 3

Name: PACCON3
Offset: 0x1EC8

Legend: R = Readable bit, W = Writable bit, S = One Way Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
							QEI1WR	TRACEWR
Access							R/W	R/W
Reset							1	1

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							QEI1LK	TRACELK
Access							S/R	S/R
Reset							0	0

Bit 17 – QEI1WR QEI 1 Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 16 – TRACEWR Trace Control Register Write Enable bit

Value	Description
1	Register is writable.
0	Register is not writable.

Bit 1 – QEI1LK QEI 1 Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

Bit 0 – TRACELK Trace Control Register Lock bit

Value	Description
1	Register is write locked.
0	Register is not write locked.

8.3. Flash Memory Map

The Flash memory is divided into several spaces as shown in Table 8-1

Table 8-1. Flash Memory Map

Base Address	Space	Size
0x7F2C00	User OTP (One-Time-Programmed)	1 Kbyte
0x7F3000	UCA1 (User Configuration A Active/ Partition 1)	4 Kbytes
0x7F4000	UCB (User Configuration B)	4 Kbytes
0x7FB000	UCA2 (User Configuration A Inactive/ Partition 2)	4 Kbytes
0x800000	Program (User Code)	Device dependent

8.3.1. User OTP

User OTP space is available for permanent storage of application specific data. User OTP Flash programming, data read and CRC access are permitted in all modes. Execution from User OTP Flash is not permitted. User OTP space is not erased on a chip erase. User OTP Flash words should only be programmed once. This must be enforced by firmware; there is no hardware access control that prohibits programming a Flash word multiple times. The User OTP space is separate from the Flash protection region OTP capability.

8.3.2. User Configuration A (UCA)

The UCA Flash space is used for storage device Configuration bits, which include code-protect and background debugger enable bits. UCA Configuration Words are loaded at Reset. Data read and CRC access to UCA space are permitted in all modes. Execution from UCA space is not permitted. Write permissions are restricted in Mission mode and Debug mode by the UCA Write-Protect WPUCA (UCPROT[2]) bit and/or by the UCA Write-Protect Configuration WPUCA[1:0] (FCP[3:2]) bits.

Subject to other access restrictions, a UCA space may be erased and then programmed in ICSP Programming mode even if the write-protect in Configuration bits WPUCA[1:0] (FCP[3:2]) is enabled. The write access to UCA spaces is not allowed if code-protect and/or Entire Flash OTP by ICSP Write Inhibit is enabled. Row programming is not allowed for UCA space. Programming of the write-protect Configuration bits, WPUCA[1:0] (FCP[3:2]), prevents UCA updates in Mission mode and Debug mode. In this case, UCA can only be updated in an ICSP Programming mode. The WPUCA (UCPROT[2]) bit may also be programmed by application firmware to enable write protection after the next Reset. UCA is always erased on a chip erase, even when write-protected. A chip erase disables UCA write protection and code protection.

8.3.3. User Configuration B (UCB)

The UCB stores security and boot Configuration bits including the Flash protection region descriptor configurations. UCB data read and CRC access is permitted in all modes. Execution from UCB space is not permitted. Write permission is restricted in all modes by the UCB Write Protect WPUCB bit (UCPROT[1]). The UCB write protection is enabled after Reset if the FWPUCB Configuration Word is programmed with a value of 0x5B9B12E4. Also, the UCB area can be erase-protected by the EPUCB bit (UCPROT[0]).

The UCB erase protection is enabled after Reset if the FEPUCB Configuration Word is programmed with a value of 0x84C1F396. Write protection disables both programming and erase. Erase protection only disables erase. The UCB write and erase-protect Configuration Words (FWPUCB and FEPUCB) are 32-bit OTP Flash locations that can only be programmed to their specified values (0x5B9B12E4 and 0x84C1F396). Row programming is not allowed for UCB. UCB is erased on a chip erase unless it is either erase-protected when EPUCB (UCPROT[0]) = ‘1’ or write-protected when WPUCB (UCPROT[1]) = ‘1’. If the UCB erase protection Configuration Word is programmed, all programmed UCB Configuration Words are protected from modification. This allows multiple parties to program firmware and data into user program Flash and permanently protect it from modification using IRT or OTP regions.

An aspect of this capability is UCB overwrite protection. UCB overwrite protection ensures Flash Configuration Words in UCB are only programmed once after each UCB page erase. Once any valid Configuration bit in a UCB 128-bit Flash word is programmed to '0', further programming is not allowed for that Flash word without a page erase. UCB overwrite protection is only provided for a word that has been programmed after the next Reset because the overwrite protection is based on the configuration values loaded at Reset.

Once all protection region descriptors and other UCB Configuration Words are programmed, UCB can be permanently write-protected by programming the UCB write-protect FWPUCB Configuration Word. Typically, the UCB write-protect Configuration bit should be programmed before a device is deployed in a system design. The UCB Write-Protect bit must be programmed to enable the Entire Flash OTP by ICSP Write Inhibit feature and/or the Secure Debug. During development or the system production process, the EPUCB (UCPROT[0]) and WPUCB (UCPROT[1]) bits can be set by firmware to provide UCB protection without programming the UCB write or erase-protect Configuration Words.

8.3.4. Program Memory

User program Flash stores code and data for Mission mode execution. Unless restricted by an access control, there is full access (execute, data read, write, CRC) to user program Flash in all modes. User program Flash access controls include the protection regions and code-protect.

Programming the UCB write-protect word permanently prevents modification of all UCB Configuration bits settings. Protection region and code-protect access controls apply to all of user program space. The chip erase operation erases all of user program Flash except for permanent (OTP and IRT) protection regions. If permanent protection regions are configured and UCB is either erase protected (EPUCB (UCPROT[0]) = '1') or write-protected (WPUCB (UCPROT[1]) = '1'), only the user program memory Flash outside the permanent regions is erased. In this case, UCB, which stores the permanent region descriptors and the permanent region contents, is not affected by the chip erase. If UCB is not erase-protected (EPUCB (UCPROT[0]) = '0') and not write-protected (WPUCB (UCPROT[1]) = '0'), then UCB and the entire user program and user data Flash are erased by a chip erase. UCA is always erased by a chip erase. Chip erase is only allowed in ICSP Programming mode. If Secure Debug is enabled, IRT firmware authorization is required for a chip erase. If Entire Flash OTP by ICSP Write Inhibit is enabled, chip erase is permanently disabled in all modes. Chip erase overrides firmware configurable (non-permanent) protection region write protections. Firmware configurable region contents are erased on a chip erase. Regardless of UCB erase or write protections, if permanent protection regions are configured, only user program Flash outside the permanent regions is erased.

8.3.5. Security Configuration Words

A summary of security related Configuration Words is shown in Table 8-2. Each entry in the table is a 128-bit Flash word that can be independently programmed. Not all 128 bits are used for any Configuration Word; unused bits are reserved. The table shows the primary Configuration Words and their addresses. The Configuration Words have primary and secondary locations, with the secondary location used in case of an uncorrectable bit error when loading the primary location. UCA and UCB secondary word addresses are the primary address + 0x800. FEPUCB and FWPUCB are OTP words that have 32-bit values to prevent inadvertent programming. These words can only be programmed to the specified value (0x84C1F396 for FEPUCB and 0x5B9B12E4 for FWPUCB). If the word has the specified 32-bit value, it is considered programmed and that option is enabled; otherwise, the option is disabled. UCB Configuration Words may not be programmed if any valid bit/field from that Flash word had previously been programmed (not all ones). The UCB overwrite protection is based on the fuse values loaded at Reset, therefore, it only takes effect for a word that has been programmed after the next Reset.

Table 8-2. Security Configuration Words

Primary Configuration Words		Secondary (Backup) Configuration Words	
Address	Name	Address	Name
User Configuration B			
0x7F4000	FPR0CTRL	0x7F4800	FPR0CTRLBKUP
0x7F4004	FPR0ST	0x7F4804	FPR0STBKUP
0x7F4008	FPR0END	0x7F4808	FPR0ENDBKUP
0x7F4010	FPR1CTRL	0x7F4810	FPR1CTRLBKUP
0x7F4014	FPR1ST	0x7F4814	FPR1STBKUP
0x7F4018	FPR1END	0x7F4818	FPR1ENDBKUP
0x7F4020	FPR2CTRL	0x7F4820	FPR2CTRLBKUP
0x7F4024	FPR2ST	0x7F4824	FPR2STBKUP
0x7F4028	FPR2END	0x7F4828	FPR2ENDBKUP
0x7F4030	FPR3CTRL	0x7F4830	FPR3CTRLBKUP
0x7F4034	FPR3ST	0x7F4834	FPR3STBKUP
0x7F4038	FPR3END	0x7F4838	FPR3ENDBKUP
0x7F4040	FPR4CTRL	0x7F4840	FPR4CTRLBKUP
0x7F4044	FPR4ST	0x7F4844	FPR4STBKUP
0x7F4048	FPR4END	0x7F4848	FPR4ENDBKUP
0x7F4050	FPR5CTRL	0x7F4850	FPR5CTRLBKUP
0x7F4054	FPR5ST	0x7F4854	FPR5STBKUP
0x7F4058	FPR5END	0x7F4858	FPR5ENDBKUP
0x7F4060	FPR6CTRL	0x7F4860	FPR6CTRLBKUP
0x7F4064	FPR6ST	0x7F4864	FPR6STBKUP
0x7F4068	FPR6END	0x7F4868	FPR6ENDBKUP
0x7F4070	FPR7CTRL	0x7F4870	FPR7CTRLBKUP
0x7F4074	FPR7ST	0x7F4874	FPR7STBKUP
0x7F4078	FPR7END	0x7F4878	FPR7ENDBKUP
0x7F4080	FIRT	0x7F4880	FIRTBKUP
0x7F4090	FSECDBG	0x7F4890	FSECDBGBKUP
0x7F40A0	FPED	0x7F48A0	FPEDBKUP
0x7F40B0	FEPUCB	0x7F48B0	FEPUCBBKUP
0x7F40C0	FWPUCB	0x7F48C0	FWPUCBBKUP

8.4. Device Locking

8.4.1. Code-Protect

Code-protect is a device locking option that prevents external readout and modification of the user program. Code-protect is enabled with a Configuration bit, CP (FCP[0]), in the UCA configuration area. The effects of enabling Code-protect are described in Table 8-3.

Table 8-3. Effects of Code Protect

Feature	Code-Protect ON	Code-Protect OFF	Notes
Debugging	Disabled	Enabled	
UCA Write Permissions	Disabled	Allowed	UCA can be write-protected via the FCP.WPUCA config bit.
Program memory access in ICSP mode	Disabled	Allowed	Execute, read and write permissions are disabled.

Table 8-3. Effects of Code Protect (continued)

Feature	Code-Protect ON	Code-Protect OFF	Notes
ECC error reporting in ICSP mode	Restricted		To prevent access to protected spaces using ECC error injection and reporting, ECC error reporting data is restricted to address only for user program in ICSP Programming mode when code-protect is enabled.
NVM Controller CRC access	Depends on FCP.CRC and Flash protection regions	Depends on Flash protection regions	If code-protect is enabled with CRC allowed, the ICSP Programming mode may calculate the CRC over user program Flash pages.
Chip Erase	• Program memory is erased. • OTP and IRT regions are not erased	• Program memory is erased. • OTP and IRT regions are not erased.	• Code-protect is disabled after chip erase. • Entire Flash OTP by ICSP Write Inhibit may be enabled to prevent external tools from performing a chip erase.
ICSP Programming Mode	Allowed	Allowed	• External tools may erase and reprogram the part, depending on Entire Flash OTP by ICSP Write Inhibit settings. • This allows for a device to be reprogrammed without revealing the original user program or Flash contents.

8.4.2. ICSP Program/Erase Disable (Entire Flash OTP by ICSP™ Write Inhibit)

ICSP Program/Erase Disable (Entire Flash OTP by ICSP Write Inhibit) permanently disables unsecured external chip erase and Flash programming. Entire Flash OTP by ICSP Write Inhibit is enabled with the TPED (FTPED[0]) Configuration bit and takes effect when the UCB write-protect FWPUCB word is also programmed with 0x5B9B12E4 value. Enabling Entire Flash OTP by ICSP Write Inhibit does the following:

- Disables chip erase in all modes.
- Disables Flash programming and erase in ICSP Programming mode.

Entire Flash OTP by ICSP Write Inhibit is intended to be used with the code-protect to permanently disable external readout or modification of the user program via unsecured programming or debug interfaces. When both code-protect and Entire Flash OTP by ICSP Write Inhibit are enabled, external tools cannot enable debug, access user program, modify Flash or disable code-protect. Programming and erase of the Flash can only be done in Mission mode. Mission mode write permissions are not affected by Entire Flash OTP by ICSP Write Inhibit. IRT firmware may support secure programming when Entire Flash OTP by ICSP Write Inhibit is enabled.

8.4.3. Secure Debug

Secure debug is a device locking option that provides IRT firmware control over external access to the device. Secure debug is enabled with a Configuration bit SECDBG (FSECDBG[0]) in the UCB area. The IRT and UCB write-protect Configuration bits also need to be programmed for secure debug access controls to take effect. When secure debug is enabled, external access is controlled by the EAA (IRTCTRL[0]) bit which is only reset on a cold Reset (POR or BOR). Only IRT firmware can write to IRTCTRL unless the DBG (IRTCTRL[1]) bit is set. When EAA (IRTCTRL[0]) bit = '0', the device is locked. All external access via debug and programming interfaces is disabled.

When the JTAG port is enabled, only the boundary scan function is allowed when the device is locked. When EAA (IRTCTRL[0]) bit = '1', debug and test access is allowed. Unless DBG (IRTCTRL[1]) bit = '1', IRT Flash regions remain protected from the external and debug access. However, a properly timed Reset during IRT firmware execution may leak IRT information in RAM and registers, which retain their state through the Reset. So, when EAA bit = '1', IRT firmware should not access symmetric (secret) or private operational keys if these need to be protected from the external access when the device is unlocked. Public keys may be accessed as disclosure of these keys is not a security issue. IRT firmware may use any means for secure debug external access authorization. The IRT firmware may implement various nonvolatile external access configuration options, including access based on an authenticated unlock token, permanently disabling access and unrestricted access. In any case, when secure debug is enabled, the device must boot after a cold Reset for the IRT firmware to enable access by setting the EAA bit.

Both the Entire Flash OTP by ICSP Write Inhibit and UCB write-protect Configuration bits must be programmed for Entire Flash OTP by ICSP Write Inhibit access controls to be enabled. UCB write-protect is an OTP Configuration Word. Once Entire Flash OTP by ICSP Write Inhibit is enabled, it cannot be disabled.

IRT, secure debug and UCB write-protect Configuration bits must all be programmed for secure debug to be enabled.

UCB write-protect (FWPUCB) is an OTP Configuration Word. Once secure debug is enabled, it cannot be disabled. Secure debug can be used either with or without code protection. When secure debug is enabled without code protection, once authorization is given, debug and test access are allowed with full access to user program and user data Flash (subject to protection region restrictions). When both secure debug and code-protect are enabled, both IRT authorization and a chip erase are required for external access to user program Flash. This allows user program to be changed without revealing its original contents and only with IRT authorization. The permanent regions (IRT and OTP) are not erased on a chip erase. When secure debug is enabled, there is no means to disable or bypass secure debug access controls. For development, secure debug can be emulated by external development tools (debugger). To emulate the secure debug, the secure debug Configuration bit (SECDBG) should not be programmed. The external development tools follow the normal procedure for enabling external access as if secure debug is enabled. The IRT firmware sets the EAA bit state, accordingly. The external development tools check the EAA bit to determine if external access would be permitted if the secure debug was enabled.

8.4.4. Immutable Root of Trust (IRT)

Device security functions including secure boot, secure debug and device attestation require a root of trust. The root of trust executes at boot time and may include multiple stages forming a chain of trust. The chain of trust operates by having each stage authenticate the next stage before transferring control to the next stage. The first stage in the chain of trust is inherently trusted and must have immutable firmware. This first and possibly only root of trust stage is referred to as the Immutable Root of Trust (IRT).

In Mission mode, IRT firmware is the first firmware to execute after a Reset. The root of trust consisting of one or more stages with immutable and optionally updatable firmware components may provide any number of security services including secure boot, secure debug, secure programming, over-the-air (OTA) firmware update and device attestation.

8.4.4.1. IRT Partition

A designated portion of the user program is used for the IRT partition. The IRT Flash space is specified using Flash protection regions. One or more permanent IRT regions may be created for firmware, data and cryptographic keys. Once IRT is enabled, IRT regions are only accessible during IRT execution except for debug access when authorized by the IRT firmware. An IRT region may have write permissions disabled, making it immutable (generally for firmware) or have write permissions enabled (generally for configuration data and keys) and allowing it to be updated.

IRT is enabled by the IRT (FIRT[0]) Configuration bit in the UCB area. By default (erased UCB), IRT is disabled. When IRT is disabled, IRT regions (if any) are not protected and are accessible to firmware outside the IRT partition. IRT regions are only truly permanent once either the UCB erase-protect and/or UCB write-protect words are programmed. For development, the UCB erase and write-protect words may be left unprogrammed. When IRT is enabled, a permanent immutable protection region with IRT firmware must be configured for the beginning of user program Flash. This region is functionally the boot ROM for the device and may be either an OTP region or IRT region with write permission disabled.

OTP region firmware can be shared between the IRT and other firmware components. IRT firmware execution starts after resetting into Mission mode. IRT execution ends when the IRT firmware sets the DONE (IRTCTRL[2]) bit and there is a subsequent instruction fetch from user program Flash not within an IRT region. This includes an attempted instruction fetch that has an error, such as an access privilege violation or an uncorrectable bit error.

IRT firmware may set the DONE bit (IRTCTRL[2]) and transfer control to the application (non-secure) code while executing from an IRT region. IRT firmware must ensure that the application code is not prefetched before the DONE bit is set. This ensures that the start of the application code is fetched after the DONE bit is set, disabling the IRT partition before the application code can execute. This requirement can be met by having at least 32 bytes of separation between the end of IRT firmware and the start of application firmware. IRT firmware must ensure the application start address is in user program Flash and not within an IRT region. For the IRT partition to be secured, either the UCB write-protect word or the UCB erase-protect word must be programmed. For development, the UCB erase and write-protect words may be left unprogrammed. Permission to access the IRT partition is indicated by the partition lock PLCK (IRTCTRL[4]) bit. Partition access is allowed (PLCK bit = ‘0’) when IRT is disabled, during IRT execution and for debug access when the DBG (IRTCTRL<1>) bit is set. Otherwise, partition access is disabled (PLCK bit = ‘1’).

The IRT partition is based on the principle of temporal isolation. Since IRT firmware is the first firmware to execute after Reset, it controls execution of other firmware components. Sensitive IRT data is cleared from registers and RAM before executing any application (non-secure firmware). Likewise, the IRT partition is disabled (PLCK bit = ‘1’) before executing the application (non-secure firmware), preventing it from accessing or corrupting IRT Flash regions. If IRT firmware code needs to be protected from access by non-security firmware, the instruction cache should be invalidated and disabled before transferring control to the application firmware. The IVT base address should be set to application space before attempting to transfer control to the application firmware. This ensures that application firmware exceptions, starting with the first application code instruction, can be handled by application trap handlers.

Secure debug must be enabled and external access must be disabled (EAA (IRTCTRL[0]) bit = ‘0’) for the IRT partition to be fully protected from the external access. When secure debug is not enabled or when it is enabled but external access is allowed (EAA bit = ‘1’), the external access to internal registers and RAM values is possible via ICSP interface. A properly timed Reset during IRT execution may reveal sensitive information in RAM and registers that retain their state through the Reset.

IRT partition access may be extended to updatable root of trust firmware components. One or more updatable components (stages) are authenticated directly by IRT firmware or indirectly with the chain of trust process previously described. The updatable root of trust components executes with the IRT partition enabled. The IRT partition is locked by the IRT or updatable root of trust firmware by setting the DONE bit before starting execution of the application (non-secure code). Alternatively, updatable root of trust components may have an independent security partition. In this case, the IRT disables the IRT partition before transferring control to an updatable root of trust component in a non-IRT region. A separate (non-IRT) protection region can be set up for the updatable root of trust components to store cryptographic keys and configuration data. Access to this region is disabled before transferring control to the application (non-secure firmware). Unlike the IRT partition, this partition is erased on a chip erase and can be accessed by debug without a specific authorization. IRT firmware may implement a security life cycle state using IRT region storage. This is separate from the higher-level device life cycle state controlled by the UCB write-protect word. Additionally, IRT firmware may implement nonvolatile secure debug Configuration options.

IRT partition memory is not erased on a chip erase, and debug access to IRT memory is controlled by IRT firmware.

8.4.4.2. IRT Control Register

The IRT control register, IRTCTRL, includes the EAA, DBG, IACT, DONE and PLCK bits. IRTCTRL is read-only except when PLCK bit = ‘0’. The EAA bit controls debug entry when secure debug is enabled. The DBG bit controls debug access to the IRT partition when IRT and debug are enabled.

The DBG bit is cleared by a cold Reset (POR or BOR) and must be set by IRT firmware to enable the IRT firmware debug.

When the DBG bit = '0', there is no debug access to the IRT partition. CPU breakpoints and debugger memory accesses are disabled during IRT execution. IRT partition access is only enabled (PLCK bit = '0') during IRT execution. When the DBG bit = '1', debug access to the IRT partition is enabled. CPU breakpoints and debugger memory accesses are allowed during IRT execution. IRT partition access is enabled (PLCK bit = '0') when the device resets into Debug mode.

The DONE bit is set by the IRT (or other root of trust) firmware upon completion of root of trust execution. The IRT partition is locked (PLCK bit = '1') when the DONE bit is set, and there is a subsequent instruction fetch from the user program Flash not within an IRT region. This includes an attempted instruction fetch which has an error, such as an access privilege violation or an uncorrectable bit error. The PLCK bit is a read-only bit that indicates if the IRT partition is locked (access disabled). If the PLCK bit = '1', access to IRT regions is disabled and the IRTCTRL and IRTSTAT registers are read-only.

8.4.4.3. IRT Status Register

The IRT status register (IRTSTAT) is a 32-bit read/write register available for IRT firmware status and is only reset on a cold Reset (POR or BOR). The IRTSTAT register usage is defined by the IRT firmware. It can be used to store secure boot and other status information across the device Resets. The IRTSTAT register is read-only except when the PLCK bit = '0'.

8.5. Flash Protection Regions

Eight configurable protection regions are available for user program and user data Flash access control. A protection region can cover one or more consecutive 4 Kbyte pages anywhere in user program Flash. Protection regions provide access control based on the type of access for the designated portion of Flash. Access control is enforced in all modes of operation. Regions may overlap with the most restrictive permissions taking precedence. Subject to other access controls, there is full access (execute, data read, write, CRC) for user program Flash pages not included within any protection region.

Flash protection regions include:

- Flash write protection
- OTP Flash
- IRT partition
- Firmware IP protection (execute only memory)
- Flash code partitioning

8.5.1. Region Descriptor

Each protection region x has a region descriptor comprised of four SFRs: Control (PRxCTRL), Start Address (PRxST), End Address (PRxEND) and Lock (PRxLOCK). Most region descriptor register bit/field Reset values can be configured with Configuration bits (FPRxCTRL, FPRxST and FPRxEND).

Region descriptor configuration options include dynamic configuration by firmware, static configuration by boot firmware and permanent configuration using fuses. This last option enables creating permanent OTP and IRT regions in program Flash that cannot be modified even with a chip erase. Firmware configurable region descriptors also include a locking mechanism. The PRxLOCK register must be configured to allow descriptor writes; otherwise, writes to other descriptor registers are ignored.

8.5.2. Region Control Register

The region descriptor PRxCTRL register includes the region disable bit: RDIS (PRxCTRL[0]), access permission bits: CRC, WR, RD, EX (PRxCTRL[7:4]) and region type field: RTYPE[1:0] (PRxCTRL[9:8]). When RDIS bit = '1', the region is disabled, and it has no effect. When RDIS bit = '0', the region is enabled, and the specified portion of user program space is subject to access control based on the region type and permissions settings.

When read access is disabled, the ECC error reporting is restricted for the region to address information only. The data, parity and syndrome information are not reported. This prevents access to region firmware using ECC error injection and reporting.

There are four permission bits, each corresponds to a Flash access type as follows:

- CRC - NVM Controller CRC
- WR - Write (Program/Erase)
- RD - Read (Data Read)
- EX - Execute (Instruction Fetch)

Access to the region is controlled based on the access type. If the permission bit associated with the access type is set, the access is allowed. Otherwise, the access is not allowed. The OTP regions never allow write operations regardless of the WR bit setting. The CRC access type refers to accesses by CRC hardware integrated into the NVM Controller. The NVM Controller CRC hardware allows firmware integrity verification without exposing the contents of the region.

The RTYPE field which is read-only can be one of three values:

- Firmware Configurable
- One Time Programmable (OTP)
- Immutable Root of Trust (IRT)

OTP and IRT regions are permanent regions configured by Configuration bits and their region descriptors cannot be modified. The permanent (OTP and IRT) region descriptors in UCB and the contents of the regions in user program space are not erased on a chip erase if either EPUCB (UCPROT[2]) bit = ‘1’ or WPUCB (UCPROT[1]) = ‘1’. To make OTP and IRT regions truly permanent, either (or both) the FEPUCB or FWPUCB Configuration Words must be programmed.

Firmware configurable region descriptors can be modified. Firmware configurable is the default type for regions when FPRnCTRL Configuration Words are not programmed.

8.5.3. Region Start and End Address Registers

The region n descriptor registers, PRxST and PRxEND, define the start address offset and end address offset for the region. The address offsets are on four Kbyte Flash page boundaries. The address offsets are an offset from the beginning of either the selected user program partition or user data Flash.

If the start address is greater than the end address, then the region size is zero and the region access controls do not apply to any portion of Flash.

8.5.4. Region Lock Register

The region x descriptor PRxLOCK register has two fields: a 16-bit KEY[15:0] field and 2-bit LOCK[1:0] field.

Writes to the register must be 32 bits (long word) and have the correct KEY value of 0xB737. Otherwise, the write is ignored. The LOCK field has three options:

- Region descriptor unlocked (LOCK[1:0] bits = ‘11’)
- Region descriptor locked, cannot be unlocked until after the next Reset (LOCK[1:0] bits = ‘01’)
- Region descriptor locked (LOCK[1:0] bits = ‘00’)

If the descriptor is locked, the PRxCTRL, PRxST and PRxEND registers are read-only. The Reset value for the LOCK[1:0] field is region descriptor locked, so a PRxLOCK register write is required to unlock the descriptor. The PRxLOCK register is not used for OTP and IRT region descriptors, which cannot be modified.

8.5.5. Permanent Regions

Multiple parties can create permanent OTP and/or IRT Flash regions in a device. If either the UCB erase-protect and/or UCB write-protect Configuration Words are programmed, OTP and IRT region descriptors cannot be modified even with a chip erase.

OTP regions can be used to prevent firmware tampering for devices that do not require firmware updates. This eliminates the need for secure boot in these devices.

Execute-only OTP regions can be used for firmware IP protection. These regions are configured to not allow data reads but allow execution and CRC access for integrity checking without exposing the protected code.

IRT protection regions designate the root of trust partition within the user program space. IRT regions can be used for IRT firmware, data and cryptographic keys. OTP regions can also be used for IRT firmware.

8.5.6. Code Partitioning

Protection regions configured exclusively for code (execute-only) or data (no execute) can be used to partition user program Flash into separate code and data spaces. This partitioning prevents inadvertent execution from data tables or data accesses to code images. Protection regions can be configured to disable all access for unused portions of Flash.

8.6. Cryptographic Accelerator Module (CAM)

The Cryptographic Accelerator Module is designed to reduce CPU workload by accelerating computationally intensive cryptographic operations. The module integrates high-performance symmetric and asymmetric cryptographic accelerators, along with a True Random Number Generator (TRNG) and a dedicated DMA unit into a single module. It can be used for realizing use cases such as secure communication, secure boot, secure firmware upgrade, node authentication or key generation.

The CAM firmware driver and other supporting collateral can be found at www.microchip.com/dspic33a-crypto-accelerator.

Table 8-4. Cryptographic Accelerator Module Summary

Module Instance	Input Clock	Peripheral Bus Speed
1	System Clock	Far Memory - Slow Peripheral Bus speed

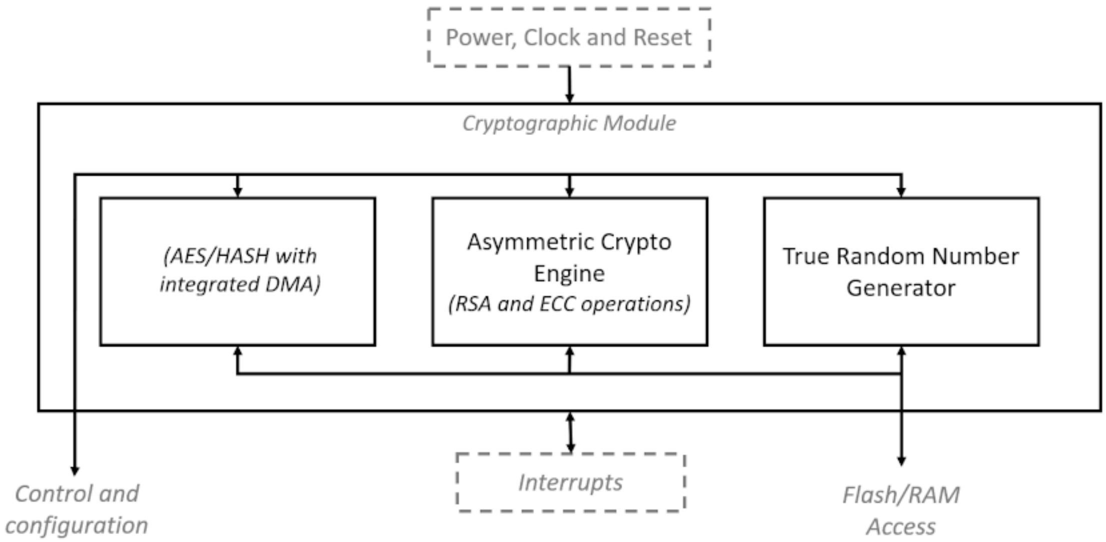
8.6.1. Architectural Overview

The Crypto module is a simple interconnection of following independent blocks through Control/Configuration and memory access ports.

- AES/HASH Engines
- Asymmetric Crypto Engine
- True Random Number Generator (TRNG)

Figure 8-2 shows a simple high-level block diagram of the Crypto Module.

Figure 8-2. High Level Block Diagram



8.6.1.1. True Random Number Generator (TRNG)

Random numbers are used for public/private key pair generation, symmetric keys, nonce generation etc. Typical secure protocols like IPsec, MACsec, TLS/SSL or other wireless authentication protocols are required during authentication/key exchange and data streaming phases.

The TRNG associated with the module is a NIST 800-90B compliant digital true random number generator with conditioning function and health tests as defined by NIST 800-90B.

8.6.1.2. Advanced Encryption Standard (AES)

The AES accelerator supports encryption and decryption operations on 128-, 192- and 256-bit key sizes, with the following modes of operation: ECB, CBC, OFB, CFB, CTR, GCM, CCM, XTS and CMAC.

The AES engine is also equipped with context-switching capabilities, which enables the user to manage larger messages in several data segments. If the message segment being processed is not the initial part, it is imperative to load the context saved from the previous data segment at the beginning of the operation.

8.6.1.3. HASH

The Hash accelerator supports the SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 hashes along with HMAC processing for all hashes. The accelerator also supports storing intermediate state of HASH, allowing hash operation on a message in several parts (several hash update), where core can be initialized with the result (state) of the previous step to perform hash on multiple chunks of data.

Additionally, the Symmetric Crypto Engine includes an integrated DMA with Scatter-Gather/direct modes support, that provides direct access to system memory, thereby effectively freeing up the system's DMA for other tasks.

8.6.1.4. Asymmetric Crypto Engine

The Asymmetric/Public-key crypto Engine implements the following algorithms: RSA, DSA, DH, ECDH, ECDSA, EdDSA, J-PAKE and SRP. RSA, DSA and DH algorithms support up to 4096-bit key sizes.

Standard ECC curves include P-256, P-384, P-521, P-192, Ed-25519 and Curve-25519 with other $GF(p)/GF(2^m)$ ECC curves supported via custom parameters. This includes other NIST, Brainpool, Koblitz, Montgomery and Edwards curves.

Up to 16 kB of system RAM is required as scratch pad RAM used for inputs and outputs. The location and size of the engine's scratch pad RAM are programmable. Similarly, approximately 4 kB of system memory is used for the engine's microcode. The placement of microcode can also be programmed and may be stored in ROM, Flash or RAM. Full public key performance requires zero wait state access to the scratch pad and microcode memory.

The Asymmetric Crypto Engine also supports an additional internal μ-DMA to collect all needed input data from the scratch pad RAM and write back the data upon completion of cryptographic operations.

8.6.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A8000	CAMCON	31:24								
		23:16								
		15:8	CAMON		SIDL					
		7:0								

8.6.2.1. Crypto Accelerator Enable Register

Name: CAMCON
Offset: 0x7A8000

Legend: W = Writable bit; R = Readable bit

Notes:
1. If CAMON = 0, only the CAMCON register can be accessed. An attempt to access other internal module registers generates a bus error trap.
2. If CAMSIDL = 1 and in Idle mode, only the CAMCON register can be accessed.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CAMON		SIDL					
Access	R/W		R/W					
Reset	0		0					

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – CAMON Module Enable bit(1)

Value	Description
1	Module is enabled.
0	Module is disabled.

Bit 13 – SIDL Stop in Idle bit(2)

Value	Description
1	Module stops the operation in Idle mode.
0	Module continues the operation in Idle mode.

8.6.3. Operations

The Reset status of Crypto module is controlled via the CAMCON register. The CAMCON SFR has the following module control bits: CAMON and CAMSIDL.

The CAMON bit enables operation of the module. If CAMON=0, the module is disabled, its internal state is reset and only the CAMCON register can be accessed. Additionally, the module can also be disabled using Peripheral Module Disable SFRs. Refer to Device Power-Saving Modes for more information.

Crypto SFR Access describes the module SFR access rights under each disable scenario.

Table 8-5. Crypto SFR Access

Crypto Module Implemented (Refer to Table 1 for details)	Crypto Module PMD bit PMD4[CRYMD]	CAMCON[CAMON]	ACCESS	ERROR
NO	0	0	NO	BMX Error(1)
YES	-	-	Only the CAMCON register is accessible. Internal crypto registers are not acessible	Bus error when Crypto registers are accessed.
YES	0	1	Both CAMCON and Crypto registers are accessible.	No error
YES	1	1	Both CAMCON and Crypto registers are not accessible.	No error. The CAMCON and Crypto Module is held in Reset and reads 0.

Note:

1. The BMX gives an error as unimplemented (invalid) space. BMXERR.ADDWERR or BMXERR.ADDRERR status bit gets set on trying to access Crypto registers when the module is not implemented in the device. Refer to Section 5.4 BMX Operation for more information.

8.6.3.1. Crypto Key Storage

The Crypto module accepts keys placed in user Flash/RAM. Additionally, keys can also be stored in a non-executable/non-writable IRT/OTP region. Access to keys stored in the IRT region can be revoked upon completion of IRT code execution, thereby securing the keys from being accessed by application firmware code. Any attempt to access keys from restricted space reads zero and logs an error in the BMXCRYPT register.

Refer to Security Module for more information.

8.6.3.2. Cryptographic API

The Crypto v4 library is used to access all cryptographic functions. The user code does not directly access accelerator or TRNG registers.

Note: Additional info on the Cryptographic Accelerator will be available under NDA via Secure Document Extranet. Customers can reach Microchip by following the steps shared on the product page to get additional details.

8.6.3.3. Operations in Sleep/Idle Modes

The clock sources to the module are shut down and the module remains halted when the device enters Sleep mode.

When the device enters Idle mode, the module's operation is governed by the CAMSIDL bit in the CAMCON register. Setting CAMCON[CAMSIDL] bit halts the module in Idle mode. Although the TRNG operation ceases when the module is halted, the TRNG ring oscillators remain active. If CAMSIDL is set to `0`, the module continues to operate in Idle mode, allowing access to all registers.

8.6.3.4. Interrupts

The Crypto module provides three level sensitive interrupt signals to the interrupt controller, one each from symmetric, asymmetric crypto engines and TRNG.

Interrupt processing functions are included in the cryptographic firmware library.

- Integrated DMA of AES/HASH engine – _CRYPTO1Interrupt
  - The CRYPTO1Interrupt request is associated with the module’s integrated DMA transfer status. The engine raises an interrupt request upon completion of data transfer or on occurrence of any error response during the transaction.

- TRNG – _CRYPTO2Interrupt
  - TRNG generates the interrupt request on any health test failure, FIFO read error or FIFO full status.
- Asymmetric Crypto Engine – _CRYPTO3Interrupt
  - Asymmetric crypto engine raises the interrupt upon completion of instructed operation.

Additionally, these interrupts also need to be enabled within internal Crypto registers along with IEC bits for CPU to receive the interrupt request. Refer to Interrupt Controller for more details.

Any error in accessing the user RAM/Flash through Crypto module is logged onto the BMXCRYPT register. Note that such errors shall not raise any interrupts/traps, and the user needs to monitor BMXCRYPT SFR at the end of each transaction.

8.7. Peripheral Access Controller (PAC)

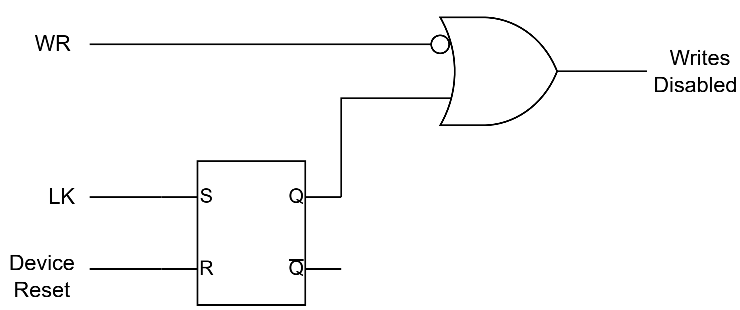
Legacy dsPIC/PIC devices used a locking mechanism where the user wrote 0xAA and 0x55 into NVMKEY in a sequence. This was to prevent accidental enabling or disabling of critical peripherals. This feature has been replaced with a dedicated module called the Peripheral Access Controller (PAC).

Register Locking and Unlocking

The module implements an OR gate between the associated peripheral's PACCONx lock bit and the inverse of the PACCONx write-enable bit. See Figure 8-3 for a diagram of the PAC locking behavior. If the target peripheral LK bit or the inverse of the WR bit is set, the target register or registers cannot be written to, and only read access is allowed. Additionally, whenever the LK or WR bit in PACCON is set or cleared, a minimum of two cycles is required for the lock or unlock to take effect.

At device Reset, all WR bits are set to '1' and all LK bits are set to '0'. This means that writes to all peripheral registers are allowed. The user can then configure access to the peripherals using the WR and LK bits. Please note that LK bits are 'One Way Settable' and will remain set until the next device Reset and cannot be cleared in software.

Figure 8-3. PAC Locking Behavior



Note: If the LK bit is set OR the WR bit is clear, peripheral access is DISABLED. Any updates to the LK or WR bits require a minimum of two cycles to take effect. Therefore, it is recommended to insert two `NOP` instructions between statements that modify these bits and access the respective register.

Individual and Range Mode

The PAC module can lock/unlock individual registers and lock/unlock a range of registers depending on which registers are the target registers. If the PAC module uses Range mode, the entire range of target peripheral registers is covered by that lock and write-enable bit. Table 8-6 is used to determine which registers use Individual or Range mode.

Table 8-6. Individual and Range Mode

Register	Individual or Range Mode
IVTBASE	Individual
IVTCREG	Individual
BMXIRAML	Individual
BMXIRAMH	Individual
PCLKCON	Individual
IOIM1CON	Individual
IOIM2CON	Individual
IOIM3CON	Individual
IOIM4CON	Individual
IOIM5CON	Individual
IOIM6CON	Individual
IOIM7CON	Individual
IOIM8CON	Individual
NVMCON	Individual
OSCCTRL	Individual
CM1CON	Individual
CM1RANGE	Range
CM2CON	Individual
CM2RANGE	Range
CM3CON	Individual
CM3RANGE	Range
CM4CON	Individual
CM4RANGE	Range
WDTCON	Individual
RPCON	Individual
MBISTCON	Individual
APCLKCON	Individual
OPAMP1	Range
OPAMP2	Range
OPAMP3	Range
IBIASCON	Individual

9. Resets

The dsPIC33AK256MPS306 family of devices implements a Reset module to safely control device start-up, faults, and external conditions. There are two types of Resets: cold Reset and warm Reset.

A cold Reset is the result of a Power-On Reset or Brown-out Reset. A warm Reset is the result of all other Reset sources, including the `RESET` instruction.

The device is kept in a Reset state until the system power supplies have stabilized at appropriate levels and the oscillator clock is ready. When the oscillator clock is ready, the processor begins execution from the Reset location. The user application programs a `GOTO` instruction at the Reset address, which redirects program execution to the appropriate start-up routine.

When the device exits the Reset condition (begins normal operation), the device operating parameters (voltage, frequency, temperature, etc.) must be within their operating ranges; otherwise, the device may not function correctly.

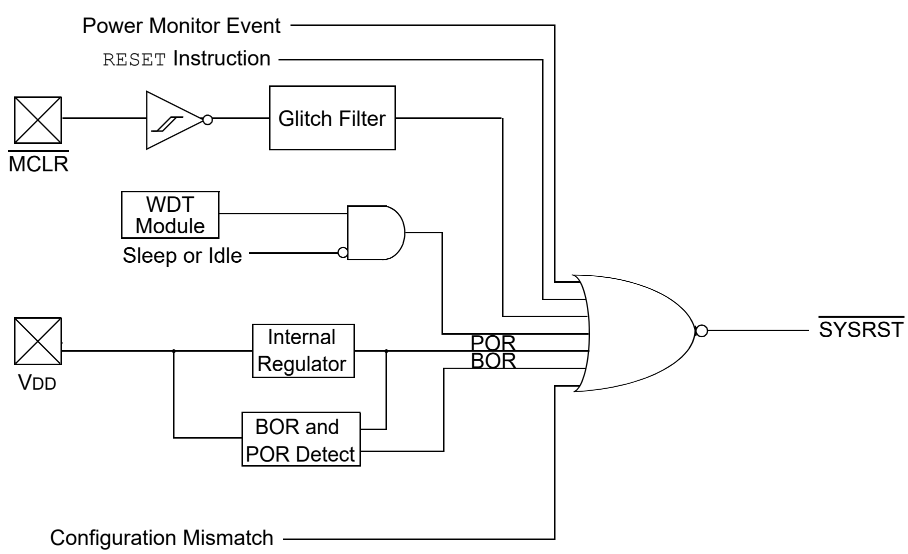
The Reset module combines all Reset sources and controls the system level Reset signal SYSRST. The following is a list of device Reset sources:

- Power-On Reset (POR)
- Power Monitor Reset
- Power Monitor Reset Event
- Brown-out Reset (BOR)
- Master Clear Reset ($\overline{\text{MCLR}}$)
- Watchdog Time-out Reset
- Software Reset (SWR)
- Configuration Mismatch Reset (CM)

9.1. Architectural Overview

A simplified block diagram of the Reset module is shown in Figure 9-1. Any active source of Reset will activate the system Reset signal. Many registers associated with the CPU and peripherals are forced into a known Reset state.

Figure 9-1. System Reset Block Diagram



Note: Multiple regulators may be instantiated with discrete status bits.

9.1.1. Reset Control Register

Name: RCON
Offset: 0x31B0

Legend: R = Readable bit, C = Clearable bit, HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		VREG4R	VREG3R	VREG2R		BUCKR		PWRMR
Access		R/C/HS	R/C/HS	R/C/HS		R/C/HS		R/C/HS
Reset		0	0	0		0		0

Bit	15	14	13	12	11	10	9	8
							CM	
Access							R/C/HS	
Reset							0	

Bit	7	6	5	4	3	2	1	0
	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR
Access	R/C/HS	R/C/HS		R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS
Reset	0	0		0	0	0	1	1

Bit 22 – VREG4R VREG Voltage Regulator 4 Flag bit

Value	Description
1	Voltage Domain 4 lost voltage, and a Reset has occurred.
0	Voltage regulation in Domain 4 remains intact

Bit 21 – VREG3R VREG Voltage Regulator 3 Flag bit

Value	Description
1	Voltage Domain 3 lost voltage, and a Reset has occurred.
0	Voltage regulation in Domain 3 remains intact.

Bit 20 – VREG2R VREG Voltage Regulator 2 Flag bit

Value	Description
1	Voltage Domain 2 lost voltage, and a Reset has occurred.
0	Voltage regulation in Domain 2 remains intact.

Bit 18 – BUCKR Buck Converter Flag bit

Value	Description
1	Buck regulator lost voltage regulation
0	Buck regulation remains intact.

Bit 16 – PWRMR PWRMR Power Monitor Reset bit

Value	Description
1	A power monitor Reset has occurred.
0	A power monitor Reset has not occurred.

Bit 9 – CM Configuration Mismatch Flag bit

Value	Description
1	A Configuration Mismatch Reset has occurred.
0	A Configuration Mismatch Reset has not occurred.

Bit 7 – EXTR External Reset (MCLR) Pin bit

Value	Description
1	A Master Clear (pin) Reset has occurred.
0	A Master Clear (pin) Reset has not occurred.

Bit 6 – SWR Software `RESET` (Instruction) Flag bit

Value	Description
1	A `RESET` instruction has been executed.
0	A `RESET` instruction has not been executed.

Bit 4 – WDTO Watchdog Timer Time-out Flag bit

Value	Description
1	Device Reset has occurred due to WDT time-out.
0	WDT time-out has not occurred.

Bit 3 – SLEEP Wake-up from Sleep Flag bit

Value	Description
1	Device has been in Sleep mode.
0	Device has not been in Sleep mode.

Bit 2 – IDLE Wake-up from Idle Flag bit

Value	Description
1	Device has been in Idle mode.
0	Device has not been in Idle mode.

Bit 1 – BOR Brown-out Reset Flag bit

Value	Description
1	A Brown-out Reset has occurred. Set by hardware upon detection of a BOR event.
0	A Brown-out Reset has not occurred.

Bit 0 – POR Power-on Reset Flag bit

Value	Description
1	A Power-On Reset has occurred. Set by hardware upon detection of a POR event.
0	A Power-On Reset has not occurred.

9.2. Operation

9.2.1. System Reset

The dsPIC33AK256MPS306 family of devices can generate an Internal System Reset (SYSRST) from multiple Reset sources, such as POR, BOR, $\overline{\text{MCLR}}$, Watchdog Time-out Reset, SWR and CM.

A system Reset is active at the first POR and asserted until device configuration settings are loaded and the oscillator clock sources become stable. The system Reset is then deasserted, allowing the CPU to start fetching code after eight system clock cycles (SYSCLK).

BOR, $\overline{MCLR}$ and WDTO Resets are asynchronous events, and to avoid SFR and RAM corruptions, the system Reset is synchronized with the system clock. All other Reset events are synchronous.

9.2.2. Power-On Reset (POR)

A POR circuit ensures the device is reset from power-on. The POR circuit is active until V_{DD} crosses the V_{POR} threshold.

A power-on event generates an internal POR pulse when a V_{DD} rise is detected above V_{POR}. The device supply voltage characteristics must meet the specified starting voltage and rise rate requirements to generate the POR pulse. In particular, V_{DD} must fall below V_{POR} before a new POR is initiated. For more information on the V_{POR} and V_{DD} rise-rate specifications, refer to the Electrical Characteristics section. A POR event can also be generated when the 1.1 V core voltage drops below a safe operating condition.

The POR bit in the Reset Control (RCON[0]) register is set to indicate the POR.

Note: When the device exits the Reset condition (begins normal operation), the device operating parameters (voltage, frequency, temperature, etc.) must be within their operating ranges; otherwise, the device will not function correctly. The user software must ensure that the delay between the time power is first applied and the time the system Reset is released is adequate to get all operating parameters within the specification.

9.2.3. Master Clear Reset ($\overline{MCLR}$)

Whenever the master clear pin ($\overline{MCLR}$) is driven low, the Reset event is synchronized with the System Clock (SYSCLK) before asserting the System Reset (SYSRST), provided the input pulse on $\overline{MCLR}$ is longer than a certain minimum width, as specified in the Electrical Characteristics section.

The $\overline{MCLR}$ pin provides a filter to minimize the effects of noise and to avoid unwanted Reset events. The Status bit, EXTR (RCON[7]), is set to indicate the $\overline{MCLR}$ Reset.

9.2.4. Software Reset (SWR)

Whenever the `RESET` instruction is executed, the device will enter a warm Reset state. The device will be released from a Reset state at the next instruction cycle, and the Reset vector fetch will commence.

The Software Reset (Instruction) Flag bit (SWR) in the Reset Control register (RCON[6]) is set to indicate the software Reset.

9.2.5. Watchdog Time-Out Reset

Whenever a Watchdog Time-Out occurs, the device will asynchronously assert SYSRST. A WDT Time-Out during Sleep or Idle mode will wake up the device, but will not reset the device.

The Watchdog Time-out Flag bit in the Reset Control register (RCON[4]) is set to indicate the Watchdog Reset.

9.2.6. Brown-Out Reset (BOR)

The BOR module generates a device Reset when a brown-out condition occurs to protect against code misexecution. The BOR module is based on an internal voltage reference circuit that monitors V_{DD}. Brown-out conditions are generally caused by glitches on the AC mains (for example, missing portions of the AC cycle waveform due to bad power transmission lines or voltage sags due to excessive current draw when a large inductive load is turned on).

A BOR generates a Reset pulse which resets the device. The BOR status bit (RCON[1]) is set to indicate that a BOR has occurred. The BOR circuit continues to operate while in Sleep or Idle mode and resets the device should VDD fall below the BOR threshold voltage. The BOR threshold voltage is detailed in the Electrical Characteristics section.

9.2.7. On-Chip Voltage Regulators

dsPIC33AK256MPS306 family devices have multiple internal voltage regulators powered via the internal Buck regulator (typical) and to other system resources.

There are three voltage domains to provide isolation of power supplies to minimize noise to peripherals. Each of the following domains has its own linear regulator configured to 1.1V.

- VREG2 – ADC
- VREG3 – PLLs and FRC
- VREG4 – FEP

In the event of a device Reset due to the voltage regulators, the RCON[22:20] register bits will report which domain lost voltage regulation.

9.2.8. Configuration Mismatch Reset (CM)

To maintain the integrity of the stored configuration values, all device Configuration bits are loaded and implemented as a complementary set of bits. As the Configuration Words are being loaded, for each bit loaded as '1', a complementary value of '0' is stored into its corresponding background word location and vice versa. The bit pairs are compared every time the Configuration Words are loaded, including Sleep mode. During this comparison, if the Configuration bit values are not found opposite to each other, a configuration mismatch event is generated, which causes a device Reset.

If a device Reset occurs as a result of a CM Reset, the CM Status bit (RCON[9]) is set.

9.2.9. Using the RCON Status Bits

The user software can read the RCON register after any system Reset to determine the cause of the Reset. Table 9-1 provides a summary of the Reset flag bit operation.

Note: The Status bits in the RCON register should be cleared after they are read so that the next RCON register value after a device Reset will be meaningful.

Table 9-1. Reset Flag Bit Operation

Flag Bit[1]	Set by	Cleared by
POR (RCON[0])	POR	User Software
BOR (RCON[1])	POR, BOR	User Software
IDLE (RCON[2])	`PWRSAV #IDLE` instruction	User Software, POR, BOR
SLEEP (RCON[3])	`PWRSAV #SLEEP` instruction	User Software, POR, BOR
WDTO (RCON[4])	WDT time-out	User Software, POR, BOR
SWR (RCON[6])	`Software Reset` command	User Software, POR, BOR
EXTR (RCON[7])	MCLR Reset	User Software, POR, BOR
CM (RCON[9])	Configuration mismatch	User Software, POR, BOR
BUCKR (RCON[17])	Buck Regulation lost	User Software, POR, BOR
VREGxR (RCON[21:19])	Voltage Domain Regulation lost	User Software
PWRMR (RCON[16])	Power Monitor Event[2]	POR, BOR

Notes:
1. All Reset flag bits may be set or cleared by the user software.
2. See Device Reset Request for specific events that will cause a power monitor event.

9.2.9.1. Device Reset to Code Execution Start Time

The delay between the end of a Reset event and when the device actually begins to execute code is determined by two main factors: the type of Reset and the system clock source coming out of the Reset. The code execution start time for various types of device Resets is summarized in Table 9-2. Individual delays are characterized in the Electrical Characteristics section.

Table 9-2. Code Execution Start Time for Various Device Resets

Reset Type	Clock Source	Power-Up Delay(1,2,3)	System Clock Delay(4,5)	FSCM Delay(6)
POR	EC, FRC, FRCDIV, LPRC	$(T_{pu}$ or $T_{pwrt}) + T_{sysdly}$	—	—
	ECPLL, FRCPLL	$(T_{pu}$ or $T_{pwrt}) + T_{sysdly}$	T_{lock}	T_{fscm}
	XT	$(T_{pu}$ or $T_{pwrt}) + T_{sysdly}$	T_{ost}	T_{fscm}
	XTPLL, HSPLL	$(T_{pu}$ or $T_{pwrt}) + T_{sysdly}$	$T_{ost} + T_{lock}$	T_{fscm}
BOR	EC, FRC, FRCDIV, LPRC	T_{sysdly}	—	—
	ECPLL, FRCPLL	T_{sysdly}	T_{lock}	T_{fscm}
	XT	T_{sysdly}	T_{ost}	T_{fscm}
	XTPLL	T_{sysdly}	$T_{ost} + T_{lock}$	T_{fscm}
MCLR, CMR, SWR, WDTO, DMTO, PWRMR	Any Clock	T_{sysdly}	—	—

For parameter specifications, see the **Electrical Characteristics** chapter.

1. Tpu = Power-up Period with on-chip regulator enabled
2. Tpwrt = Power-up Period (Power-up Timer) with on-chip regulator disabled
3. Tsysdly = Time required to reload Device Configuration Fuses plus eight SYSCLK cycles
4. T_{ost} = Oscillator Start-up Timer
5. T_{lock} = PLL lock time
6. T_{fscm} = Fail-Safe Clock Monitor delay

9.3. Application Example

After a device Reset, the RCON register can be examined by initialization code to confirm the source of the Reset. In certain applications, this information can be used to take appropriate action to correct the problem that caused the Reset to occur.

All Reset status bits in the RCON register should be cleared after reading them to ensure the RCON value will provide meaningful results after the next device Reset.

Example 9-1 illustrates how to determine the source of device Reset using the RCON register.

Example 9-1. Determining the Source of Device Reset

```
int main(void)
{
//... perform application specific startup tasks
// next, check the cause of the Reset
if(RCON & 0x0003)
{
// execute a Power-on Reset handler
// ...
}
else if(RCON & 0x0002)
{
// execute a Brown-out Reset handler
// ...
}
else if(RCON & 0x0080)
{
// execute a Master Clear Reset handler
// ...
}
else if(RCON & 0x0040)
{
// execute a Software Reset handler
// ...
}
else if (RCON & 0x0200)
{
// execute a Configuration Mismatch Reset handler
```

```
// ...
}
else if (RCON & 0x0010)
{
// execute Watchdog Time-out Reset handler
// ...
}
// ... perform other application-specific tasks
while(1);
}
```

9.4. Power Monitor

The dsPIC33AK256MPS306 device family features a power monitoring system that autonomously scans the internal voltage domains and provides real-time status indication. The purpose of the system is to ensure that critical system resources needed for the application's key peripherals are within specification. Each voltage domain and the system bandgap are checked for overvoltage (OV) and undervoltage (UV) conditions. If either condition is detected, a device Reset will occur. The power monitoring system includes the following features:

- Autonomous OV and UV detection
- Real-time status indication
- Fault indication storage
- Dedicated trip points for voltage domains and system bandgap
- Independent clock source
- Independent bandgap reference
- Configurable Fault injection

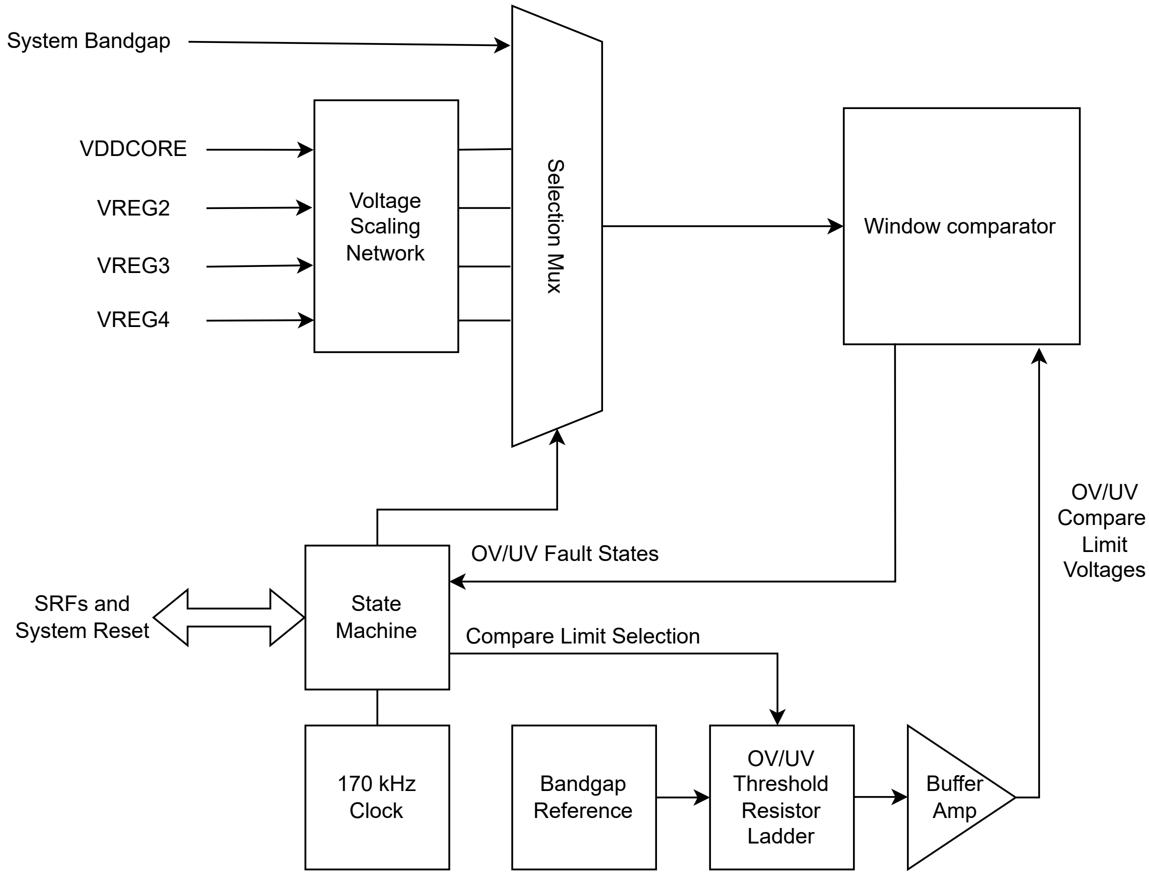
9.4.1. Architectural Overview

The power monitor system is intended to check the integrity of the internal low-voltage power domains and the system bandgap, as critical system resources depend on them operating within specification. Critical system resources include the CPU, oscillator and PLLs, ADC, comparators and PWM. The power monitor is powered by V_{DD} directly and is not dependent on the low-voltage domains that it is checking. The power monitor also has its own independent clock source and can continue to operate in the event of a clock failure or reconfiguration. A dedicated bandgap is provided to check the system's bandgap voltage and provide the OV and UV trip points. A state machine provides autonomous operation independent of the system CPU and controls scanning of voltage domains, trip points, indication and Fault injection support. If an OV/UV event is detected, the state machine issues a system Reset and logs the source of the error to be retrieved after the Reset has occurred.

The power monitor is based around a window comparator that utilizes predefined trip points. The OV/UV trip points are programmed during manufacturing and stored in configuration memory that is not accessible or modifiable by user software. The trip point voltages utilized by the comparators are generated by the power monitor's bandgap and buffered with a reference amplifier. A voltage scaling circuit is provided to scale the internal voltage domains to that of the power monitor's reference and trip point voltages.

Fault injection mechanisms are provided to verify the power monitor's input muxing and trip points. A status register provides an indication of Fault injection status.

Figure 9-2. Power Monitor High Level



9.4.2. State Machine

The state machine processes the independent monitoring of the voltage domains. The state machine updates the source select signal on each rising clock edge using a round-robin scheme. The factory scan sources are listed in Table 9-3. If any of the scan sources trigger an OV or UV event, a device Reset will occur.

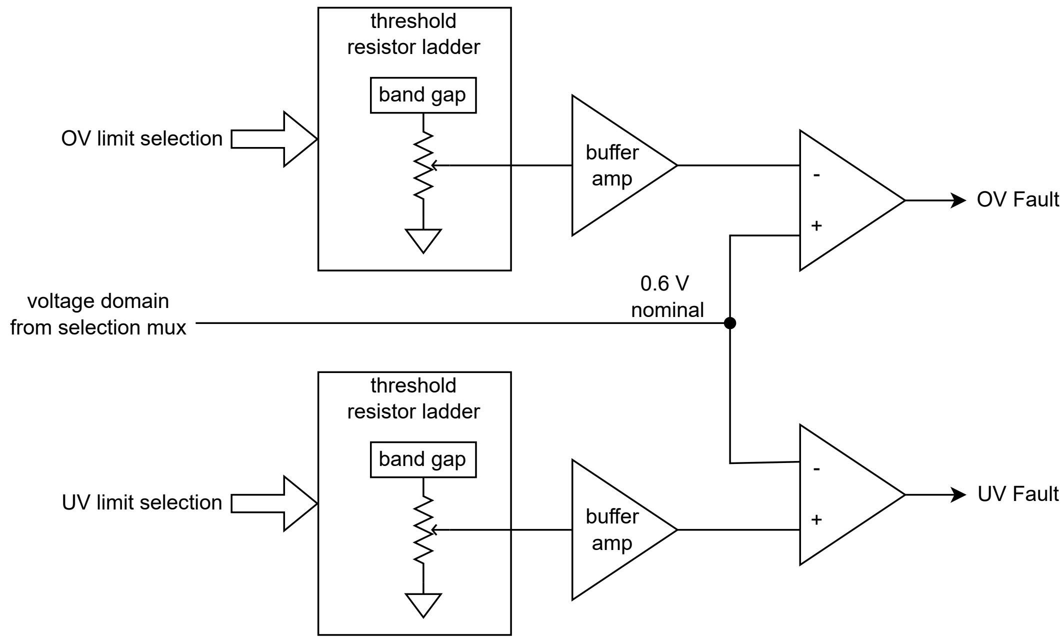
Table 9-3. Power Monitor Scan Sources

Scan Sources
System Bandgap
V_{DDCORE}
VREG2
VREG3
VREG4

9.4.3. Comparator

The trip points are provided to the window comparator, which consists of two separate comparators: one intended for OV and the other for UV. With the input voltage source being monitored and divided down for a 0.6 V comparison, each comparator independently indicates its condition upon detection of such a violation. VMxEVENT will indicate whether an OV or UV event occurred, and the source of the event.

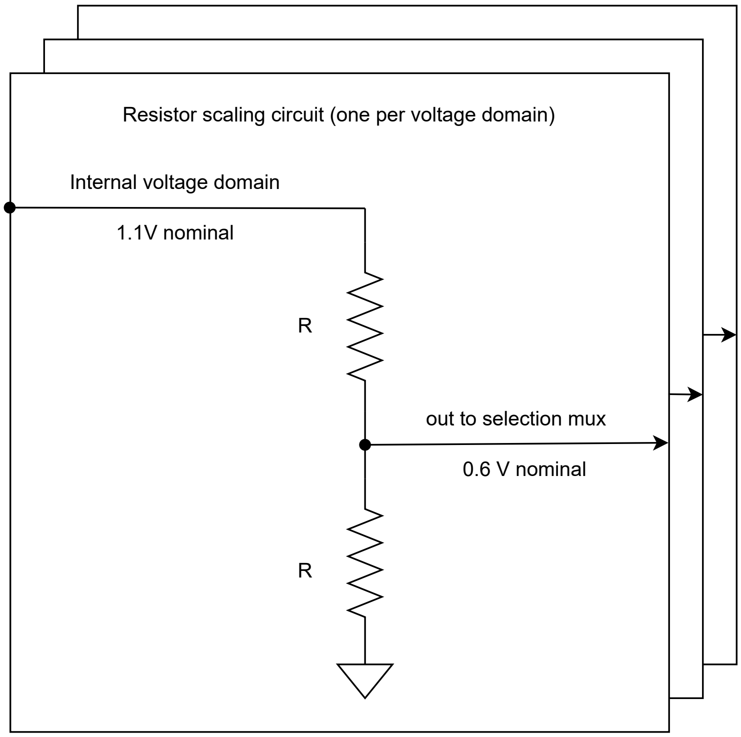
Figure 9-3. Window Comp



9.4.4. Resistor Divider

The resistor divider reduces 1.1V to 0.6V to scale the internal voltage domains so that they are within the range of the voltage monitor trip points. The divide-down function is accomplished by the resistor divider unit to simplify the design, which is responsible for trip point selections while allowing flexibility for the factory and user-configurable fault-injection trip point range.

Figure 9-4. Voltage Scaler



9.4.5. Oscillator

To support independent operation with its controller, the module includes a built-in oscillator. A built-in 70 kHz oscillator maintains input source monitoring if the system oscillator fails due to dependency on monitored input sources. It is designed to operate without having to depend on any V_{DDCORE} level logic and/or the associated voltage source (i.e., LDO), most likely the sources being monitored.

9.4.6. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3B28	VM1CON	31:24	ON							
		23:16								
		15:8						SCANPS[2:0]		
		7:0								CLRSTAT
0x3B2C	VM1STAT	31:24	WRALLOW							FLTRST
		23:16								
		15:8								
		7:0				SCANSRC[3:0]				SCANBG
0x3B30	VM1FLT	31:24	FLTRSTEN		FLTOVUV				FLTOVSEL[4:0]	
		23:16	FLTOVSEL[4:0]			FLTUVSEL[4:0]				
		15:8								
		7:0								
0x3B34	VM1EVENT	31:24								
		23:16				OVSRC[3:0]				OVBG
		15:8								
		7:0				UVSRC[3:0]				UVBG

9.4.6.1. Voltage Monitor Control Register

Name: VM1CON
Offset: 0x3B28

Notes:
1. Bit can only be set by the user; setting the bit initiates a status register clearing sequence between the two clock domains.
2. Bit is cleared by hardware once the status bits are cleared.
3. Bit can only be set by the user.
4. If FLTRSTEN = `1`, the minimum value allowed to be written to this bit field is 0b001.

Bit	31	30	29	28	27	26	25	24
	ON							
Access	R/S/HS							
Reset	x							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						SCANPS[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
								CLRSTAT
Access								R/W/HC
Reset								0

Bit 31 – ON Power Monitor Enable bit[3]

Value	Description
`1`	Power Monitor Module is enabled.
`0`	Power Monitor Module is not enabled.

Bits 10:8 – SCANPS[2:0] Scan Postscaler bits[4]

Value	Description
`111`	16384x duration between scans
`...`	
`010`	16x duration between scans
`001`	4x duration between scans
`000`	Back-to-back, continuous scan at maximum scan frequency (approx. 34 kHz scan rate for 5 inputs plus one extra pulse for WRALLOW using the 170 kHz internal clock)

Bit 0 – CLRSTAT Status Clear Request bit[1,2]

Value	Description
`1`	Request to clear status registers initiated.
`0`	No pending request to clear VMxSTAT or VMxEVENT registers

9.4.6.2. Voltage Monitor Status Register

Name: VM1STAT
Offset: 0x3B2C

Notes:
1. The bit is hardware set at the end of the round-robin sequence and is hardware cleared half a clock cycle later (providing sufficient time to synchronize any updates before the next sequence).
2. Hardware sets the bit when a Reset request is made due to a Fault-injected condition.
3. Hardware sets the bit upon completion of the scan.

Bit	31	30	29	28	27	26	25	24
	WRALLOW							FLTRST
Access	R/HS/HC							R/HS/HC
Reset	0							0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				SCANSRC[3:0]				SCANBG
Access				R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset				0	0	0	0	0

Bit 31 – WRALLOW Configuration Write Allow Status bit(1)

Value	Description
1	VMxCON[SCANPS[2:0]] and VMxFLT can be safely written without creating a false error event.
0	Writing to VMxCON[SCANPS[2:0]] and VMxFLT is not allowed; writing can cause unexpected behavior of the module or a false error event.

Bit 24 – FLTRST Power Monitor Reset Status bit(2)

Value	Description
1	Reset event caused by the module occurred while performing Fault injection.
0	No Reset event caused by Fault injection testing has occurred.

Bits 4:1 – SCANSRC[3:0] Factory Source Scanned Status bit

Value	Description
1000	VREG4 Scan is complete.
0100	VREG3 Scan is complete.
0010	VREG2 Scan is complete.
0001	V_{DDCORE} (Buck) scan is complete.
0000	No factory sources were scanned.

Bit 0 – SCANBG Bandgap Scan Status bit(3)

Value	Description
1	Bandgap scan is complete.
0	Bandgap has not been scanned.

9.4.6.3. Voltage Monitor Fault Injection Configuration Register

Name: VM1FLT
Offset: 0x3B30

Note:
1. Selections 1 through 31 are in steps of 6.2 mV.

Bit	31	30	29	28	27	26	25	24
	FLTRSTEN		FLTOVUV				FLTOVSEL[4:0]	
Access	R/W		R/W/HC				R/W	R/W
Reset	0		0				0	1

Bit	23	22	21	20	19	18	17	16
	FLTOVSEL[4:0]			FLTUVSEL[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 31 – FLTRSTEN Reset Request via Fault-Injection Enable bit

Value	Description
1	Triggers a Reset request when a Fault caused by Fault injection occurs.
0	Prevents a Reset request from being triggered when a Fault is caused by a Fault injection sequence.

Bit 29 – FLTOVUV Reference Voltage Override Enable bit

Value	Description
1	Allows FLTOVSEL[4:0] and FLTUVSEL[4:0] bits to override reference voltages in the next scan.
0	FLTOVSEL[4:0] and FLTUVSEL[4:0] will not override reference voltages in the next scan.

Bits 25:21 – FLTOVSEL[4:0] OV Trip Point Selection for Fault Injection Overrides bits**(1)**
Used only when FLTOVUV = 1

Value	Description
11111	0.743V
11110	0.737V
...	
00010	0.562V
00001	0.556V
00000	0.450V

Bits 20:16 – FLTUVSEL[4:0] UV Trip Point Selection for Fault Injection Overrides bits**(1)**
Used only when FLTOVUV = 1

Value	Description
11111	0.450V

Value	Description
11110	0.456V
...	
00010	0.631V
00001	0.637V
00000	0.743V (skips to highest available setting)

9.4.6.4. Voltage Monitor Event Status Register

Name: VM1EVENT
Offset: 0x3B34

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				OVSRC[3:0]				OVBG
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				UVSRC[3:0]				UVBG
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 20:17 – OVSRC[3:0] Factory Scan Source OV Event Status bit

Value	Description
1000	VREG4 caused an OV event.
0100	VREG3 caused an OV event.
0010	VREG2 caused an OV event.
0001	V_{DDCORE} (Buck regulator) caused an OV event.
0000	Factory Scan Source OV event has not occurred.

Bit 16 – OVBG System Bandgap OV Event Status bit

Value	Description
1	Overvoltage event caused by the main system bandgap has occurred.
0	Overvoltage event caused by the main system bandgap has not occurred.

Bits 4:1 – UVSRC[3:0] Factory Scan Source UV Event Status bit

Value	Description
1000	VREG4 caused a UV event.
0100	VREG3 caused a UV event.
0010	VREG2 caused a UV event.
0001	V_{DDCORE} (Buck regulator) caused a UV event.
0000	Factory Scan Source UV event has not occurred.

Bit 0 – UVBG System Bandgap UV Event Status bit

Value	Description
1	Undervoltage event caused by main system bandgap has occurred.
0	Undervoltage event caused by main system bandgap has not occurred.

9.4.7. Operation

9.4.7.1. Module Enable

The module is enabled by programming the PWRM bits in the FPWRM configuration register with the value 0b1010. Once enabled via the configuration fuse, the module will remain ON always, even after a system Reset, unless a value of 0 is programmed into the FPWRM configuration register and a device POR occurs.

9.4.7.2. Device Reset Request

If the Power Monitor Module detects that the voltage source being monitored is above the specified overvoltage level or below the undervoltage level, it asserts the corresponding signal to inform the system of the violation. With up to six scan sources, the Power Monitor module relies on its built-in oscillator to rotate through all the selected voltage sources to perform its tasks. Following a violation event, a SYSRST can occur. A SYSRST signal is asserted under the following conditions:

- A UV or OV event occurs during normal operation (Fault injection disabled).
- A UV or OV event occurs due to Fault injection with FLTRSTEN=`1`
  The VMxSTAT[FLTRST] indicates the source of the Fault condition after a Reset and should be read after the PWRMR bit, RCON[16], indicates a power monitor Reset as the source to see if the Reset was due to Fault injection or an actual voltage failure.

9.4.7.3. Source Select

The state machine controls the loading of the source select value on each rising edge of the clock. The value is left-shifted starting from 1 to 4 (the number of selectable sources) on every clock edge to create a one-hot selection signal for the analog module. When the source is selected via the state machine round-robin sequencer, the source is scanned and SCANSRC[2:0] is set once the scan is complete. After all sources have been scanned and compared against the UV and OV trip points, SCANSRC[1:0] resets to 0 and then is loaded with 1 on the next clock. The neutral 0 selection is used as the period during which module configuration changes can be made. The VMSTAT.WRALLOW bit is set during this period. After the number of selectable sources is reached, SCANSRC[2:0] is set to 0b4000 on the final clock, before being reset to `0`.

- On each rising edge of the clock, the corresponding VMxSTAT[SCANSRCn] bit is set.
- On each rising edge of the clock, the corresponding VMxEVENT[UVSRCn] bit is set *if* a UV event occurs.
- On each rising edge of the clock, the corresponding VMxEVENT[OVSRCn] bit is set *if* an OV event occurs.

9.4.7.4. Over Voltage and Under Voltage Trip Points

The default trip-point selections are driven by the corresponding calibrated values for the regulator voltages and the bandgap input. All sources will be checked against the OV and UV trip points during the round-robin sequence.

9.4.7.5. Scan Postscaler Configuration

This setting controls the interval between consecutive input scan cycles. VMxCON[SCANPS] sets the frequency of the round-robin tests; the frequency setting determines the inactive time between the round-robin test sequences.

The base duration is approximately 29 µS (34 kHz frequency) for 5 inputs, based on a 170 kHz internal clock. A value of 0b000 enables continuous, back-to-back scanning at the base duration, which is the maximum rate. One additional clock cycle is used for the WRALLOW pulse. Higher SCANPS values apply post-scaling to extend the interval between scans:

- `111`: 16,384x base duration
- ...
- `010`: 16x base duration

- 001: 4x base duration

This allows the scan rate to be adjusted to suit application requirements such as power consumption or processing bandwidth.

9.4.7.6. Fault Injection Testing

If at the beginning of the round-robin sequence the FLTOVUV bit is set, the FLTOVSEL[4:0] and FLTUVSEL[4:0] settings will be used instead. The state machine will also clear the FLTOVUV bit at the end of the scan sequence to ensure the override event is active only for one round-robin sequence.

By default, Fault-injected failures do not cause a Reset. The only effect of the Fault is the setting of the corresponding VMxSTAT and VMxEVENT status bits. By setting the FLTRSTEN bit, any Fault-injected events will also cause a device Reset. However, this must be explicitly enabled before commencing a test. The Fault-injected test is a trip point override test.

The Fault injection test is described in Fault Injection Tests.

9.4.7.7. Fault Injection Tests

Trip Point Override Test

This test shows that monitored sources will trigger OV/UV Faults if threshold limits are tightened.

Procedure:

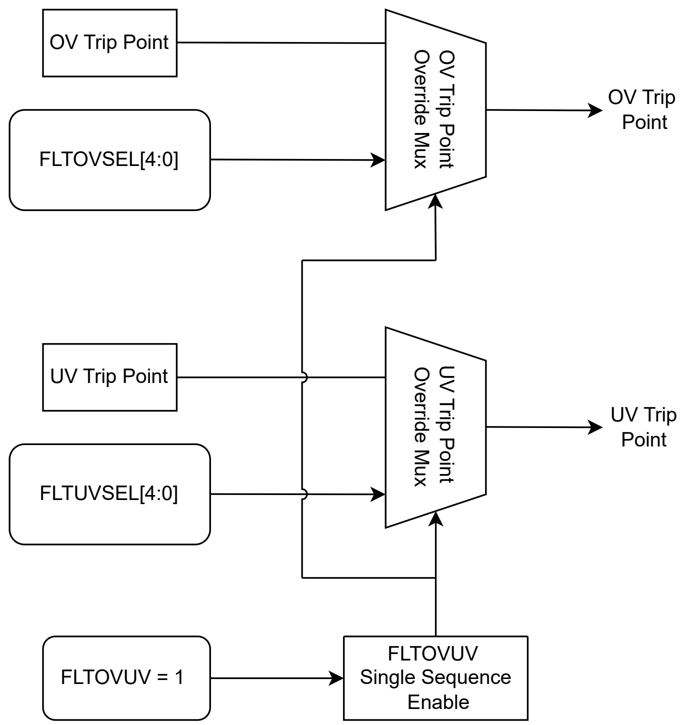
1. Wait for WRALLOW = 1.
2. Adjust trip points via FLTUVSEL[4:0] and FLTOVSEL[4:0].
3. Set the FLTOVUV bit to enable Fault injection.
4. The system continues scanning all regular sources.
5. The Fault status appears in the VMxSTAT register.

If a Fault is detected:

- The corresponding VMxEVENT bit is set.
- If FLTRSTEN = 1: a device Reset will occur on the first detected Fault, and the FLTRST status bit is set.
- If FLTRSTEN = 0: no device Reset occurs.

Note: The test is executed once per activation; the FLTOVUV bit auto-clears after one round-robin cycle. This prevents the system from remaining in a reduced monitoring state after the test.

Figure 9-5. Overriding Trip Point Voltages While Scanning Regular Sources



Low Power Enable

At the completion of the round-robin scan sequence, low power is asserted for the duration configured through SCANPS[2:0] settings. Once the low-power period expires, the mode is deasserted. Note that there is no low-power period when SCANPS[2:0] = `0`. While in Low-Power mode, the bandgap and the oscillator continue to run. It is expected that the mode reduces current consumption from 130 μA to approximately 80 μA.

Interrupts

There is one interrupt associated with the Power Monitor module.

- PWRMIE – Power Monitor Interrupt Enable bit
- PWRMIF – Power Monitor Interrupt Flag - set high when round-robin source scan is complete. It also coincides with the WRALLOW bit being set to indicate Fault settings can safely be adjusted.
- PWRMIP – Power Monitor Interrupt Priority bit

9.5. Effects of Reset

The Reset value for the Reset Control register, RCON, will depend on the type of device Reset, as indicated in Table 9-4.

Table 9-4. Status Bits, Their Significance and the Initialization Condition for RCON Register

Condition	Program Counter	EXTR	SWR	WDTO	SLEEP	IDLE	CM	BOR	POR	PWRMR
Power-on Reset or $\overline{\text{MCLR}}$ set as POR	0x00000000_0000	0	0	0	0	0	0	1	1	1
Brown-out Reset		0	0	0	0	0	0	1	u	1
$\overline{\text{MCLR}}$ Reset during Run Mode		1	u	u	u	u	u	u	u	u
$\overline{\text{MCLR}}$ Reset during Idle Mode		1	u	u	u	1	u	u	u	u
$\overline{\text{MCLR}}$ Reset during Sleep Mode		1	u	u	1	u	u	u	u	u
Software Reset Command		u	1	u	u	u	u	u	u	u
Configuration Word Mismatch Reset		u	u	u	u	u	1	u	u	u
WDT Time-out Reset during Run Mode		u	u	0	u	u	u	u	u	u
WDT Time-out Reset during Idle Mode	PC+2	u	u	0	u	1	u	u	u	u
WDT Time-out Reset during Sleep Mode		u	u	1	1	u	u	u	u	u
Interrupt Exit from Idle Mode	PC+2 or Interrupt Vector	u	u	u	u	1	u	u	u	u
Interrupt Exit from Sleep Mode		u	u	u	1	u	u	u	u	u
Power Monitor Reset Event	0x00000000_0000	u	u	u	u	u	u	1	1	1

Table 9-4. Status Bits, Their Significance and the Initialization Condition for RCON Register (continued)

Condition	Program Counter	EXTR	SWR	WDTO	SLEEP	IDLE	CM	BOR	POR	PWRMR
Legend: u = unchanged										
Note: The Program Counter (PC) is loaded with PC + 2 if the interrupt priority is less than or equal to the CPU interrupt priority level. The PC is loaded with the hardware vector address if the interrupt priority is greater than the CPU interrupt priority level.										

9.5.1. Special Function Register (SFR) Reset States

Most of the SFRs associated with the dsPIC33A CPU and peripherals are reset to a particular value at a device Reset. This also applies to a Reset due to Run mode WDT time-out, which is treated as a full device Reset by the dsPIC33A CPU and peripherals. Refer to register details of the specific peripheral registers for SFR Reset values.

9.5.2. Configuration Word Register Reset States

All Reset conditions force the Configuration settings to be reloaded. The POR sets all the Configuration Word register locations to a '1' before loading the Configuration settings. For all other Reset conditions, the Configuration Word register locations are not reset prior to being reloaded. This difference in behavior accommodates $\overline{\text{MCLR}}$ assertions during Debug mode without affecting the state of the debug operations.

10. Interrupt Controller

The dsPIC33AK256MPS306 family interrupt controller reduces the numerous peripheral interrupt request signals to a single interrupt request signal to its CPU. The core supports a prioritized interrupt and trap exception scheme.

The interrupt controller has the following features:

- Interrupt Vector Table (IVT) for User Memory
- Reset Vector (Not Part of IVT)
- Eight Processor Traps
- Four Generic Traps + One Software Trap
- Seven User Selectable Priority Levels
- A Unique Vector for Each Interrupt or Exception in Full IVT Mode
- A Collapsed Vector for All Peripheral Interrupts
- Fixed Priority Within a Specified User Priority Level
- Software Can Generate Any Peripheral Interrupt
- Relocatable IVT (via IVTBASE Register)

10.1. Device-Specific Information

Table 10-1. Interrupt Vector Details

IRQ #	Interrupt Source	MPLAB® ISR Name
0	_COMMONInterrupt	Common Collapsed Interrupt
1	_CPUFPUInterrupt	FPU Interrupt
2	_XRAMECCInterrupt	X RAM ECC Single Error Interrupt
3	_YRAMECCInterrupt	Y RAM ECC Single Error Interrupt
4	_PBUEInterrupt	PBU Parity Error Interrupt
5	_NVMECCInterrupt	NVM ECC Single Error Interrupt
6	_NVMInterrupt	NVM Program or Erase Operation Completed Interrupt
7	_NVMCRCInterrupt	NVM CRC Operation Completed Interrupt
9	_CLKFInterrupt	Combined Clock Fail Interrupt
10	_CLKEInterrupt	Combined Clock Error Interrupt
11	_CLK1FInterrupt	Clock 1 Failure Interrupt
12	_CLK1WInterrupt	Clock 1 Warning Interrupt
13	_CLK1MInterrupt	Clock 1 Monitor Error Interrupt
14	_CLK1RInterrupt	Clock 1 Ready Interrupt
15	_CLK2FInterrupt	Clock 2 Failure Interrupt
16	_CLK2WInterrupt	Clock 2 Warning Interrupt
17	_CLK2MInterrupt	Clock 2 Monitor Error Interrupt
18	_CLK2RInterrupt	Clock 2 Ready Interrupt
19	_CLK3FInterrupt	Clock 3 Failure Interrupt
20	_CLK3WInterrupt	Clock 3 Warning Interrupt
21	_CLK3MInterrupt	Clock 3 Monitor Error Interrupt
22	_CLK3RInterrupt	Clock 3 Ready Interrupt
23	_CLK4FInterrupt	Clock 4 Failure Interrupt
24	_CLK4WInterrupt	Clock 4 Warning Interrupt
25	_CLK4MInterrupt	Clock 4 Monitor Error Interrupt

Table 10-1. Interrupt Vector Details (continued)

IRQ #	Interrupt Source	MPLAB® ISR Name
26	_CLK4RInterrupt	Clock 4 Ready Interrupt
28	_WDTInterrupt	Wake up from WDT Interrupt
30	_CRYPTO1Interrupt	Crypto 1 Interrupt
31	_CRYPTO2Interrupt	Crypto 2 Interrupt
32	_CRYPTO3Interrupt	Crypto 3 Interrupt
33	_INT0Interrupt	External Interrupt 0
34	_INT1Interrupt	External Interrupt 1
35	_INT2Interrupt	External Interrupt 2
36	_INT3Interrupt	External Interrupt 3
37	_INT4Interrupt	External Interrupt 4
38	_PWMEVTAInterrupt	PWM Event A Interrupt
39	_PWMEVTBInterrupt	PWM Event B Interrupt
40	_PWMEVTCInterrupt	PWM Event C Interrupt
41	_PWMEVTDInterrupt	PWM Event D Interrupt
42	_PWMEVTEInterrupt	PWM Event E Interrupt
43	_PWMEVTFInterrupt	PWM Event F Interrupt
44	_PWM1Interrupt	PWM Generator 1 Interrupt
45	_PWM2Interrupt	PWM Generator 2 Interrupt
46	_PWM3Interrupt	PWM Generator 3 Interrupt
47	_PWM4Interrupt	PWM Generator 4 Interrupt
48	_T1Interrupt	Timer 1 Interrupt
49	_T2Interrupt	Timer 2 Interrupt
50	_T3Interrupt	Timer 3 Interrupt
51	_CCT1Interrupt	CCP 1 Timer Interrupt
52	_CCP1Interrupt	CCP 1 Input Capture or Output Compare Interrupt
53	_CCT2Interrupt	CCP 2 Timer Interrupt
54	_CCP2Interrupt	CCP 2 Input Capture or Output Compare Interrupt
55	_CCT3Interrupt	CCP 3 Timer Interrupt
56	_CCP3Interrupt	CCP 3 Input Capture or Output Compare Interrupt
57	_CCT4Interrupt	CCP 4 Timer Interrupt
58	_CCP4Interrupt	CCP 4 Input Capture or Output Compare Interrupt
59	_C1RXInterrupt	CAN 1 RX Data Ready Interrupt
60	_C1TXInterrupt	CAN 1 TX Data Request Interrupt
61	_C1Interrupt	CAN 1 Combined Error Interrupt
65	_SPI1RXInterrupt	SPI 1 RX Interrupt
66	_SPI1TXInterrupt	SPI 1 TX Interrupt
67	_SPI1EInterrupt	SPI 1 Error Interrupt
68	_SPI2RXInterrupt	SPI 2 RX Interrupt
69	_SPI2TXInterrupt	SPI 2 TX Interrupt
70	_SPI2EInterrupt	SPI 2 Error Interrupt
71	_SPI3RXInterrupt	SPI 3 RX Interrupt
72	_SPI3TXInterrupt	SPI 3 TX Interrupt
73	_SPI3EInterrupt	SPI 3 Error Interrupt
77	_DMA0Interrupt	DMA Channel 0 Interrupt
78	_DMA1Interrupt	DMA Channel 1 Interrupt

Table 10-1. Interrupt Vector Details (continued)

IRQ #	Interrupt Source	MPLAB® ISR Name
79	_DMA2Interrupt	DMA Channel 2 Interrupt
80	_DMA3Interrupt	DMA Channel 3 Interrupt
81	_CMP1Interrupt	Analog Comparator 1 Interrupt
82	_CMP2Interrupt	Analog Comparator 2 Interrupt
83	_CMP3Interrupt	Analog Comparator 3 Interrupt
84	_CMP4Interrupt	Analog Comparator 4 Interrupt
85	_I2C1EInterrupt	I2C 1 Error Interrupt
86	_I2C1Interrupt	I2C 1 General Interrupt
87	_I2C1RXInterrupt	I2C 1 RX Buffer Full Interrupt
88	_I2C1TXInterrupt	I2C 1 TX Buffer Full Interrupt
89	_I2C2EInterrupt	I2C 2 Error Interrupt
90	_I2C2Interrupt	I2C 2 General Interrupt
91	_I2C2RXInterrupt	I2C 2 RX Buffer Full Interrupt
92	_I2C2TXInterrupt	I2C 2 TX Buffer Full Interrupt
98	_U1RXInterrupt	UART 1 RX Interrupt
99	_U1TXInterrupt	UART 1 TX Interrupt
100	_U1EInterrupt	UART 1 Error Interrupt
101	_U1EVTInterrupt	UART 1 Event Interrupt
102	_U2RXInterrupt	UART 2 RX Interrupt
103	_U2TXInterrupt	UART 2 TX Interrupt
104	_U2EInterrupt	UART 2 Error Interrupt
105	_U2EVTInterrupt	UART 2 Event Interrupt
106	_U3RXInterrupt	UART 3 RX Interrupt
107	_U3TXInterrupt	UART 3 TX Interrupt
108	_U3EInterrupt	UART 3 Error Interrupt
109	_U3EVTInterrupt	UART 3 Event Interrupt
110	_U4RXInterrupt	UART 4 RX Interrupt
111	_U4TXInterrupt	UART 4 TX Interrupt
112	_U4EInterrupt	UART 4 Error Interrupt
113	_U4EVTInterrupt	UART 4 Event Interrupt
114	_SENT1Interrupt	SENT1 RX and TX Interrupt
115	_SENT1EInterrupt	SENT 1 Error Interrupt
116	_SENT2Interrupt	SENT2 RX and TX Interrupt
117	_SENT2EInterrupt	SENT 2 Error Interrupt
118	_DMA4Interrupt	DMA 4 Interrupt
119	_DMA5Interrupt	DMA 5 Interrupt
120	_DMA6Interrupt	DMA 6 Interrupt
121	_DMA7Interrupt	DMA 7 Interrupt
122	_CNAInterrupt	Change Notice A Interrupt
123	_CNBInterrupt	Change Notice B Interrupt
124	_CNCInterrupt	Change Notice C Interrupt
125	_CNDInterrupt	Change Notice D Interrupt
126	_CCT5Interrupt	CCP 5 Timer Interrupt
127	_CCP5Interrupt	CCP 5 Input Capture or Output Compare Interrupt
136	_QEI1Interrupt	QEI 1 Position Counter Compare Interrupt

Table 10-1. Interrupt Vector Details (continued)

IRQ #	Interrupt Source	MPLAB® ISR Name
140	_BISS1EInterrupt	BISS 1 Transmission Error Interrupt
141	_BISS1Interrupt	BISS 1 Transmission Finished Interrupt
142	_CRCInterrupt	CRC Interrupt
143	_ICDInterrupt	ICD Interrupt
145	_PTGSTEPInterrupt	PTG Step Interrupt
146	_PTGWDTInterrupt	PTG WDT Interrupt
147	_PTG0Interrupt	PTG Interrupt 0
148	_PTG1Interrupt	PTG Interrupt 1
149	_PTG2Interrupt	PTG Interrupt 2
150	_PTG3Interrupt	PTG Interrupt 3
155-156	Reserved	Reserved
157	_AD1CH0Interrupt	ADC 1 Data Channel 0 Done Interrupt
158	_AD1CMP0Interrupt	ADC 1 Digital Comparator 0 Interrupt
159	_AD1CH1Interrupt	ADC 1 Data Channel 1 Done Interrupt
160	_AD1CMP1Interrupt	ADC 1 Digital Comparator 1 Interrupt
161	_AD1CH2Interrupt	ADC 1 Data Channel 2 Done Interrupt
162	_AD1CMP2Interrupt	ADC 1 Digital Comparator 2 Interrupt
163	_AD1CH3Interrupt	ADC 1 Data Channel 3 Done Interrupt
164	_AD1CMP3Interrupt	ADC 1 Digital Comparator 3 Interrupt
165	_AD1CH4Interrupt	ADC 1 Data Channel 4 Done Interrupt
166	_AD1CMP4Interrupt	ADC 1 Digital Comparator 4 Interrupt
167	_AD1CH5Interrupt	ADC 1 Data Channel 5 Done Interrupt
168	_AD1CMP5Interrupt	ADC 1 Digital Comparator 5 Interrupt
169	_AD1CH6Interrupt	ADC 1 Data Channel 6 Done Interrupt
170	_AD1CMP6Interrupt	ADC 1 Digital Comparator 6 Interrupt
179	_AD2CH0Interrupt	ADC 2 Data Channel 0 Done Interrupt
180	_AD2CMP0Interrupt	ADC 2 Digital Comparator 0 Interrupt
181	_AD2CH1Interrupt	ADC 2 Data Channel 1 Done Interrupt
182	_AD2CMP1Interrupt	ADC 2 Digital Comparator 1 Interrupt
183	_AD2CH2Interrupt	ADC 2 Data Channel 2 Done Interrupt
184	_AD2CMP2Interrupt	ADC 2 Digital Comparator 2 Interrupt
185	_AD2CH3Interrupt	ADC 2 Data Channel 3 Done Interrupt
186	_AD2CMP3Interrupt	ADC 2 Digital Comparator 3 Interrupt
187	_AD2CH4Interrupt	ADC 2 Data Channel 4 Done Interrupt
188	_AD2CMP4Interrupt	ADC 2 Digital Comparator 4 Interrupt
189	_AD2CH5Interrupt	ADC 2 Data Channel 5 Done Interrupt
190	_AD2CMP5Interrupt	ADC 2 Digital Comparator 5 Interrupt
191	_AD2CH6Interrupt	ADC 2 Data Channel 6 Done Interrupt
192	_AD2CMP6Interrupt	ADC 2 Digital Comparator 6 Interrupt
193	_AD2CH7Interrupt	ADC 2 Data Channel 7 Done Interrupt
194	_AD2CMP7Interrupt	ADC 2 Digital Comparator 7 Interrupt
201	_AD3CH0Interrupt	ADC 3 Data Channel 0 Done Interrupt
202	_AD3CMP0Interrupt	ADC 3 Digital Comparator 0 Interrupt
203	_AD3CH1Interrupt	ADC 3 Data Channel 1 Done Interrupt
204	_AD3CMP1Interrupt	ADC 3 Digital Comparator 1 Interrupt

Table 10-1. Interrupt Vector Details (continued)

IRQ #	Interrupt Source	MPLAB® ISR Name
205	_AD3CH2Interrupt	ADC 3 Data Channel 2 Done Interrupt
206	_AD3CMP2Interrupt	ADC 3 Digital Comparator 2 Interrupt
207	_AD3CH3Interrupt	ADC 3 Data Channel 3 Interrupt
208	_AD3CMP3Interrupt	ADC 3 Digital Comparator 3 Interrupt
209	_AD3CH4Interrupt	ADC 3 Data Channel 4 Interrupt
210	_AD3CMP4Interrupt	ADC 3 Digital Comparator 4 Interrupt
211	_AD3CH5Interrupt	ADC 3 Data Channel 5 Interrupt
212	_AD3CMP5Interrupt	ADC 3 Digital Comparator 5 Interrupt
213	_AD3CH6Interrupt	ADC 3 Data Channel 6 Interrupt
214	_AD3CMP6Interrupt	ADC 3 Digital Comparator 6 Interrupt
215	_AD3CH7Interrupt	ADC 3 Data Channel 7 Interrupt
216	_AD3CMP7Interrupt	ADC 3 Digital Comparator 7 Interrupt
217	_AD3CH8Interrupt	ADC 3 Data Channel 8 Interrupt
218	_AD3CMP8Interrupt	ADC 3 Digital Comparator 8 Interrupt
219	_AD3CH9Interrupt	ADC 3 Data Channel 9 Interrupt
220	_AD3CMP9Interrupt	ADC 3 Digital Comparator 9 Interrupt
221	_AD3CH10Interrupt	ADC 3 Data Channel 10 Interrupt
222	_AD3CMP10Interrupt	ADC 3 Digital Comparator 10 Interrupt
223	_AD3CH11Interrupt	ADC 3 Data Channel 11 Interrupt
224	_AD3CMP11Interrupt	ADC 3 Digital Comparator 11 Interrupt
273	_CMP5Interrupt	Analog Comparator 5 Interrupt
277	_CLC1PInterrupt	CLC 1 Positive Edge Interrupt
278	_CLC1NInterrupt	CLC 1 Negative Edge Interrupt
279	_CLC2PInterrupt	CLC 2 Positive Edge Interrupt
280	_CLC2NInterrupt	CLC 2 Negative Edge Interrupt
281	_CLC3PInterrupt	CLC 3 Positive Edge Interrupt
282	_CLC3NInterrupt	CLC 3 Negative Edge Interrupt
283	_CLC4PInterrupt	CLC 4 Positive Edge Interrupt
284	_CLC4NInterrupt	CLC 4 Negative Edge Interrupt
327	_ITCInterrupt	Touch ADC Interrupt
332	_IOIM1Interrupt	GPIO Integrity Monitor 1 Interrupt
333	_IOIM2Interrupt	GPIO Integrity Monitor 2 Interrupt
334	_IOIM3Interrupt	GPIO Integrity Monitor 3 Interrupt
335	_IOIM4Interrupt	GPIO Integrity Monitor 4 Interrupt
336	_IOIM5Interrupt	GPIO Integrity Monitor 5 Interrupt
337	_IOIM6Interrupt	GPIO Integrity Monitor 6 Interrupt
338	_IOIM7Interrupt	GPIO Integrity Monitor 7 Interrupt
339	_IOIM8Interrupt	GPIO Integrity Monitor 8 Interrupt
360	_I3C1GInterrupt	I3C Generic Interrupt
361	_I3C1SERRInterrupt	I3C SERR Interrupt
362	_I3C1DERRInterrupt	I3C1 DERR Interrupt
368	_RDCCORInterrupt	RDC CORDIC Operation Complete Interrupt
369	_RDCEInterrupt	RDC Error Condition Interrupt
370	_CICINInterrupt	CIC Input Sample Processing Done Interrupt
371	_CICOUTInterrupt	CIC Output Sample Ready Interrupt

Table 10-1. Interrupt Vector Details (continued)

IRQ #	Interrupt Source	MPLAB® ISR Name
372	_CICEInterrupt	CIC Error Interrupt
373	_PWRMInterrupt	Power Monitor Interrupt

10.2. Architectural Overview

The interrupt controller module assembles all the interrupt request signals from the peripherals and assigns both a fixed natural order priority and a user assigned priority to each signal. The highest level unmasked interrupt request is then presented to the processor core along with a vector number, which represents an offset into the IVT.

The interrupt controller provides interrupt sources that can be programmed with different priority levels along with six processor traps and other generic traps.

10.2.1. System Traps and Interrupts

- CPU
  - Address error exception
  - Stack error exception
  - Math error trap due to arithmetic divide by zero
  - Math error trap due to arithmetic Accumulator A Overflow
  - Math error trap due to arithmetic Accumulator B Overflow
  - Math error trap due to arithmetic Accumulator A Catastrophic Overflow
  - Math error trap due to arithmetic Accumulator B Catastrophic Overflow
  - Math error trap due to arithmetic attempted out of range SFTAC
  - Program memory bus error
  - X-space read or write bus error
  - Y-space read or write bus error
  - Illegal instruction trap
- NVM Controller
  - NVM ECC single error correction interrupt
  - Erase programming complete/error interrupt CRC done
- DMA
  - DMA bus error trap
  - DMA channel interrupts
- ICD
  - ICD bus error
- DMT
  - DMT event generic trap
- WDT
  - WDT Sleep/Idle interrupt
  - WDT Run event generic trap
- PBU Cache
  - Cache parity error interrupt
- XRAM Controller

- XRAM - ECC single-error correction interrupt
- XRAM - PWB DED generic trap

- YRAM Controller
  - YRAM - ECC single-error correction interrupt
  - YRAM - PWB DED generic trap

The interrupt controller is responsible for pre-processing the peripheral interrupts and processor exceptions prior to them being presented to the processor core. The interrupts and traps are enabled, prioritized and controlled using centralized special function registers.

10.3. Interrupt Vector Table

The Interrupt Vector Table (IVT) resides in the program memory. The IVT contains interrupt vectors plus six processor trap vectors. In general, each interrupt source has its own vector. Each interrupt vector contains a 24-bit wide address. The value programmed into each interrupt vector location is the starting address of the associated Interrupt Service Routine (ISR). See Table 10-1 for interrupt vector details.

The processor core is responsible for performing the interrupt bus cycle: reading the IVT and transferring the address contained in the interrupt vector to the program counter. The interrupt vector is transferred from the program data bus into the program counter via a 24-bit wide multiplexer on the input of the program counter.

The peripheral IVT is relocatable using an SFR (IVTBASE) to set the base address. At a device Reset, the base address value is “0x800000”. The IVT starts at address 0x800000, and code execution begins at the address specified in the Reset vector at 0x800000. The IVT is shown in Figure 10-1.

The user can choose an alternate IVT table based on the IVTBASE value. This allows the erasing and reprogramming of user code without affecting the vector table. This is useful in various software update scenarios.

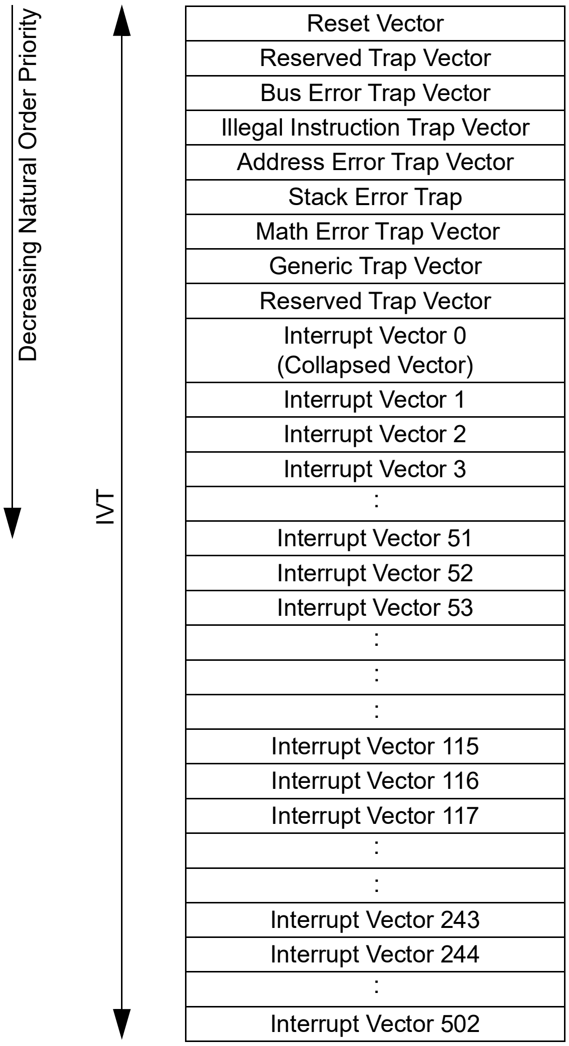
10.3.1. Remappable Interrupt Vector Table

The interrupt controller has the capability to relocate the Interrupt Vector Table location using the IVTBASE register. This feature is referred to as the Remappable Interrupt Vector Table (RIVT). The RIVT proves beneficial in instances where there is a necessity to modify the current vector table, such as updating vector address values, or to redefine the vectors for alternative uses, including debugging processes or different application programs.

Additionally, IVT can be relocated anywhere within Flash/RAM by defining IVT in a new location and modifying the IVTBASE register accordingly.

Note: The value of IVTBASE must be aligned to addresses of 2^6 (i.e., least significant 6 bits of IVTBASE must be set to 0). Refer to the Interrupt Vector Base Address Register (IVTBASE) register summary for more details.

Figure 10-1. Interrupt Vector Table (IVTC = 0, Default)

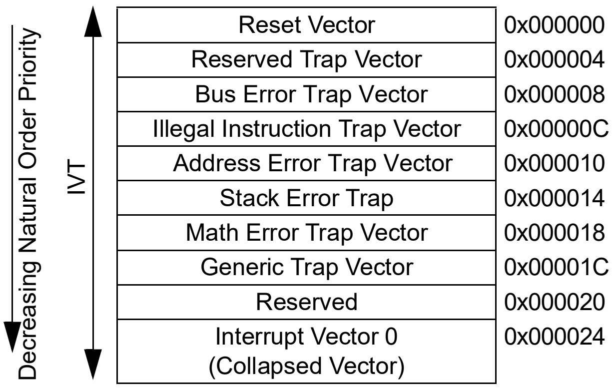


10.3.2. Interrupt Vector Table Collapse

The IVT can be collapsed by configuring the IVTC bit within the IVTCREG register. While trap handlers remain unaffected by this setting, all peripheral interrupts are directed to a single location that follows trap vectors. When the IVTC bit is set to `1`, all peripheral interrupts utilize a common vector, which is located at the offset address 0x24.

The collapsed peripheral interrupt vector is placed in the reserved interrupt location. The trap interrupts are not collapsed, and the peripheral interrupts are pointed to one location, which is placed after the trap's interrupt location, as shown in Figure 10-2.

Figure 10-2. Interrupt Vector Table (IVTC = 1)



The potential application of remappable and/or collapsible Interrupt Vector Tables is in scenarios where the processor is operating within a secure or boot memory segment and encounters a trap or an interrupt. In such cases, the secure boot software is required to assign the interrupt base address to a designated interrupt vector address located within the secure or boot memory segment. Upon completion of its operations, the secure boot software will then assign predetermined values to specific memory segments. Additionally, the secure boot software has the option to set the Interrupt Vector Table Collapse (IVTC) to "`1`", thereby consolidating the interrupt vector for all peripherals into a single entry. Once the secure boot software has finalized its processes, it may reset this bit, effectively deactivating the collapsed IVT feature.

10.4. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x70	INTCON1	31:24	NSTDIS							
		23:16								
		15:8	GIE							
		7:0				STKERR	ADDRERR	BADOPERR		
0x74	INTCON2	31:24								
		23:16								
		15:8								
		7:0				INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
0x78	INTCON3	31:24								
		23:16								
		15:8								
		7:0					CPUBET	DMABET	YRAMBET	XRAMBET
0x7C	INTCON4	31:24								
		23:16			OVATE	OVBTE	COVTE			
		15:8								
		7:0			OVAERR	OVBERR	COVAERR	COVBERR	SFTACERR	DIV0ERR
0x80	INTCON5	31:24	SOFT							
		23:16								
		15:8								
		7:0					YRAM	XRAM	WDTE	DMTE
0x84	INTTREG	31:24								
		23:16			VHOLD					
		15:8			ILR[3:0]					VECNUM[8]
		7:0	VECNUM[7:0]							
0x88	IVTBASE	31:24	IVTBASE[31:24]							
		23:16	IVTBASE[23:16]							
		15:8	IVTBASE[15:8]							
		7:0	IVTBASE[7:6]							
0x8C	IVTCREG	31:24								
		23:16								
		15:8								
		7:0								IVTC
0x90	IFS0	31:24	CRYPT2IF	CRYPT1IF		WDTIF		C4RDYIF	C4MONIF	C4WARNIF
		23:16	C4FAILIF	C3RDYIF	C3MONIF	C3WARNIF	C3FAILIF	C2RDYIF	C2MONIF	C2WARNIF
		15:8	C2FAILIF	C1RDYIF	C1MONIF	C1WARNIF	C1FAILIF	CLKERRIF	CLKFAILIF	
		7:0	NVMCRCIF	NVMIF	NVMECCIF	PBERRIF	YRAMECCIF	XRAMECCIF	CPUFPUIF	IVTCIF
0x94	IFS1	31:24			C1IF	C1TXIF	C1RXIF	CCP4IF	CCT4IF	CCP3IF
		23:16	CCT3IF	CCP2IF	CCT2IF	CCP1IF	CCT1IF	T3IF	T2IF	T1IF
		15:8	PWM4IF	PWM3IF	PWM2IF	PWM1IF	PEVTFIF	PEVTEIF	PEVTDIF	PEVTCIF
		7:0	PEVTBIF	PEVTAIF	INT4IF	INT3IF	INT2IF	INT1IF	INT0IF	CRYPT3IF
0x98	IFS2	31:24				I2C2TXIF	I2C2RXIF	I2C2IF	I2C2EIF	I2C1TXIF
		23:16	I2C1RXIF	I2C1IF	I2C1EIF	CMP4IF	CMP3IF	CMP2IF	CMP1IF	DMA3IF
		15:8	DMA2IF	DMA1IF	DMA0IF				SPI3EIF	SPI3TXIF
		7:0	SPI3RXIF	SPI2EIF	SPI2RXIF	SPI2RXIF	SPI1EIF	SPI1TXIF	SPI1RXIF	
0x9C	IFS3	31:24	CCP5IF	CCT5IF	CNDIF	CNCIF	CNBIF	CNAIF	DMA7IF	DMA6IF
		23:16	DMA5IF	DMA4IF	SENT2EIF	SENT2IF	SENT1EIF	SENT1IF	U4EVTIF	U4EIF
		15:8	U4TXIF	U4RXIF	U3EVTIF	U3EIF	U3TXIF	U3RXIF	U2EVTIF	U2EIF
		7:0	U2TXIF	U2RXIF	U1EVTIF	U1EIF	U1TXIF	U1RXIF		I2C3TXIF
0xA0	IFS4	31:24	AD1CH1IF	AD1CMP0IF	AD1CH0IF					
		23:16		PTG3IF	PTG2IF	PTG1IF	PTG0IF	PTGWDTIF	PTGSTEPIF	
		15:8	ICDIF	CRCIF	BISS1IF	BISS1EIF				QEI1IF
		7:0								
0xA4	IFS5	31:24	AD2CH6IF	AD2CMP5IF	AD2CH5IF	AD2CMP4IF	AD2CH4IF	AD2CMP3IF	AD2CH3IF	AD2CMP2IF
		23:16	AD2CH2IF	AD2CMP1IF	AD2CH1IF	AD2CMP0IF	AD2CH0IF			
		15:8						AD1CMP6IF	AD1CH6IF	AD1CMP5IF
		7:0	AD1CH5IF	AD1CMP4IF	AD1CH4IF	AD1CMP3IF	AD1CH3IF	AD1CMP2IF	AD1CH2IF	AD1CMP1IF

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xA8	IFS6	31:24	AD3CH11IF	AD3CMP10IF	AD3CH10IF	AD3CMP9IF	AD3CH9IF	AD3CMP8IF	AD3CH8IF	AD3CMP7IF
		23:16	AD3CH7IF	AD3CMP6IF	AD3CH6IF	AD3CMP5IF	AD3CH5IF	AD3CMP4IF	AD3CH4IF	AD3CMP3IF
		15:8	AD3CH3IF	AD3CMP2IF	AD3CH2IF	AD3CMP1IF	AD3CH1IF	AD3CMP0IF	AD3CH0IF	
		7:0						AD2CMP7IF	AD2CH7IF	AD2CMP6IF
0xAC	IFS7	31:24								
		23:16								
		15:8								
		7:0								AD3CMP11IF
0xB0	IFS8	31:24				CLC4NIF	CLC4PIF	CLC3NIF	CLC3PIF	CLC2NIF
		23:16	CLC2PIF	CLC1NIF	CLC1PIF				CMP5IF	
		15:8								
		7:0								
0xB4 ... 0xB7	Reserved									
0xB8	IFS10	31:24								
		23:16					IOM8IF	IOM7IF	IOM6IF	IOM5IF
		15:8	IOM4IF	IOM3IF	IOM2IF	IOM1IF				
		7:0	ITCIF							
0xBC	IFS11	31:24								
		23:16			PWRMIF	CICERRIF	CICOSRIF	CICIPDIF	RDCERRIF	RDCCORIF
		15:8						I3C1DEIF	I3C1SEIF	I3C1GIF
		7:0								
0xC0	IEC0	31:24	CRYPT2IE	CRYPT1IE		WDTIE		C4RDYIE	C4MONIE	C4WARNIE
		23:16	C4FAILIE	C3RDYIE	C3MONIE	C3WARNIE	C3FAILIE	C2RDYIE	C2MONIE	C2WARNIE
		15:8	C2FAILIE	C1RDYIE	C1MONIE	C1WARNIE	C1FAILIE	CLKERRIE	CLKFAILIE	
		7:0	NVMCRCIE	NVMIE	NVMECCIE	PBERRIE	YRAMECCIE	XRAMECCIE	CPUFPUIE	
0xC4	IEC1	31:24			C1IE	C1TXIE	C1RXIE	CCP4IE	CCT4IE	CCP3IE
		23:16	CCT3IE	CCP2IE	CCT2IE	CCP1IE	CCT1IE	T3IE	T2IE	T1IE
		15:8	PWM4IE	PWM3IE	PWM2IE	PWM1IE	PEVTFIE	PEVTEIE	PEVTDIE	PEVTCIE
		7:0	PEVTBIE	PEVTAIE	INT4IE	INT3IE	INT2IE	INT1IE	INT0IE	CRYPT3IE
0xC8	IEC2	31:24				I2C2TXIE	I2C2RXIE	I2C2IE	I2C2EIE	I2C1TXIE
		23:16	I2C1RXIE	I2C1IE	I2C1EIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE	DMA3IE
		15:8	DMA2IE	DMA1IE	DMA0IE				SPI3EIE	SPI3TXIE
		7:0	SPI3RXIE	SPI2EIE	SPI2TXIE	SPI2RXIE	SPI1EIE	SPI1TXIE	SPI1RXIE	
0xCC	IEC3	31:24	CCP5IE	CCT5IE	CNDIE	CNCIE	CNBIE	CNAIE	DMA7IE	DMA6IE
		23:16	DMA5IE	DMA4IE	SENT2EIE	SENT2IE	SENT1EIE	SENT1IE	U4EVTIE	U4EIE
		15:8	U4TXIE	U4RXIE	U3EVTIE	U3EIE	U3TXIE	U3RXIE	U2EVTIE	U2EIE
		7:0	U2TXIE	U2RXIE	U1EVTIE	U1EIE	U1TXIE	U1RXIE		
0xD0	IEC4	31:24	AD1CH1IE	AD1CMP0IE	AD1CH0IE					
		23:16		PTG3IE	PTG2IE	PTG1IE	PTG0IE	PTGWDTIE	PTGSTEPIE	
		15:8	ICDIE	CRCIE	BISS1IE	BISS1EIE				QEI1IE
		7:0								
0xD4	IEC5	31:24	AD2CH6IE	AD2CMP5IE	AD2CH5IE	AD2CMP4IE	AD2CH4IE	AD2CMP3IE	AD2CH3IE	AD2CMP2IE
		23:16	AD2CH2IE	AD2CMP1IE	AD2CH1IE	AD2CMP0IE	AD2CH0IE			
		15:8						AD1CMP6IE	AD1CH6IE	AD1CMP5IE
		7:0	AD1CH5IE	AD1CMP4IE	AD1CH4IE	AD1CMP3IE	AD1CH3IE	AD1CMP2IE	AD1CH2IE	AD1CMP1IE
0xD8	IEC6	31:24	AD3CH11IE	AD3CMP10IE	AD3CH10IE	AD3CMP9IE	AD3CH9IE	AD3CMP8IE	AD3CH8IE	AD3CMP7IE
		23:16	AD3CH7IE	AD3CMP6IE	AD3CH6IE	AD3CMP5IE	AD3CH5IE	AD3CMP4IE	AD3CH4IE	AD3CMP3IE
		15:8	AD3CH3IE	AD3CMP2IE	AD3CH2IE	AD3CMP1IE	AD3CH1IE	AD3CMP0IE	AD3CH0IE	
		7:0						AD2CMP7IE	AD2CH7IE	AD2CMP6IE
0xDC	IEC7	31:24								
		23:16								
		15:8								
		7:0								AD3CMP11IE
0xE0	IEC8	31:24				CLC4NIE	CLC4PIE	CLC3NIE	CLC3PIE	CLC2NIE
		23:16	CLC2PIE	CLC1NIE	CLC1PIE				CMP5IE	
		15:8								
		7:0								

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0xE4 ... 0xE7	Reserved									
0xE8	IEC10	31:24								
		23:16					IOM8IE	IOM7IE	IOM6IE	IOM5IE
		15:8	IOM4IE	IOM3IE	IOM2IE	IOM1IE				
		7:0	ITCIE							
0xEC	IEC11	31:24								
		23:16			PWRMIE	CICERRIE	CICOSRIE	CICIPDIE	RDCERRIE	RDCCORIE
		15:8						I3C1DEIE	I3C1GIE	I3C1SEIE
		7:0								
0xF0	IPC0	31:24		NVMCRCIP[2:0]				NVMIP[2:0]		
		23:16		NVMECCIP[2:0]				PBERRIP[2:0]		
		15:8		YRAMECCIP[2:0]				XRAMECCIP[2:0]		
		7:0		CPUFPUIP[2:0]				IVTCIP[2:0]		
0xF4	IPC1	31:24		C2FAILIP[2:0]				C1RDYIP[2:0]		
		23:16		C1MONIP[2:0]				C1WARNIP[2:0]		
		15:8		C1FAILIP[2:0]				CLKERRIP[2:0]		
		7:0		CLKFAILIP[2:0]						
0xF8	IPC2	31:24		C4FAILIP[2:0]				C3RDYIP[2:0]		
		23:16		C3MONIP[2:0]				C3WARNIP[2:0]		
		15:8		C3FAILIP[2:0]				C2RDYIP[2:0]		
		7:0		C2MONIP[2:0]				C2WARNIP[2:0]		
0xFC	IPC3	31:24		CRYPT2IP[2:0]				CRYPT1IP[2:0]		
		23:16						WDTIP[2:0]		
		15:8						C4RDYIP[2:0]		
		7:0		C4MONIP[2:0]				C4WARNIP[2:0]		
0x0100	IPC4	31:24		PEVTBIP[2:0]				PEVTAIP[2:0]		
		23:16		INT4IP[2:0]				INT3IP[2:0]		
		15:8		INT2IP[2:0]				INT1IP[2:0]		
		7:0		INT0IP[2:0]				CRYPT3IP[2:0]		
0x0104	IPC5	31:24		PWM4IP[2:0]				PWM3IP[2:0]		
		23:16		PWM2IP[2:0]				PWM1IP[2:0]		
		15:8		PEVTFIP[2:0]				PEVTEIP[2:0]		
		7:0		PEVTDIP[2:0]				PEVTCIP[2:0]		
0x0108	IPC6	31:24		CCT3IP[2:0]				CCP2IP[2:0]		
		23:16		CCT2IP[2:0]				CCP1IP[2:0]		
		15:8		CCT1IP[2:0]				T3IP[2:0]		
		7:0		T2IP[2:0]				T1IP[2:0]		
0x010C	IPC7	31:24								
		23:16		C1IP[2:0]				C1TXIP[2:0]		
		15:8		C1RXIP[2:0]				CCP4IP[2:0]		
		7:0		CCT4IP[2:0]				CCP3IP[2:0]		
0x0110	IPC8	31:24		SPI3RXIP[2:0]				SPI2EIP[2:0]		
		23:16		SPI2TXIP[2:0]				SPI2RXIP[2:0]		
		15:8		SPI1EIP[2:0]				SPI1TXIP[2:0]		
		7:0		SPI1RXIP[2:0]						
0x0114	IPC9	31:24		DMA2IP[2:0]				DMA1IP[2:0]		
		23:16		DMA0IP[2:0]						
		15:8								
		7:0		SPI3EIP[2:0]				SPI3TXIP[2:0]		
0x0118	IPC10	31:24		I2C1RXIP[2:0]				I2C1IP[2:0]		
		23:16		I2C1EIP[2:0]				CMP4IP[2:0]		
		15:8		CMP3IP[2:0]				CMP2IP[2:0]		
		7:0		CMP1IP[2:0]				DMA3IP[2:0]		
0x011C	IPC11	31:24								
		23:16						I2C2TXIP[2:0]		
		15:8		I2C2RXIP[2:0]				I2C2IP[2:0]		
		7:0		I2C2EIP[2:0]				I2C1TXIP[2:0]		

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0120	IPC12	31:24		U2TXIP[2:0]				U2RXIP[2:0]		
		23:16		U1EVTIP[2:0]				U1EIP[2:0]		
		15:8		U1TXIP[2:0]				U1RXIP[2:0]		
		7:0								
0x0124	IPC13	31:24		U4TXIP[2:0]				U4RXIP[2:0]		
		23:16		U3EVTIP[2:0]				U3EIP[2:0]		
		15:8		U3TXIP[2:0]				U3RXIP[2:0]		
		7:0		U2EVTIP[2:0]				U2EIP[2:0]		
0x0128	IPC14	31:24		DMA5IP[2:0]				DMA4IP[2:0]		
		23:16		SENT2EIP[2:0]				SENT2IP[2:0]		
		15:8		SENT1EIP[2:0]				SENT1IP[2:0]		
		7:0		U4EVTIP[2:0]				U4EIP[2:0]		
0x012C	IPC15	31:24		CCP5IP[2:0]				CCT5IP[2:0]		
		23:16		CNDIP[2:0]				CNCIP[2:0]		
		15:8		CNBIP[2:0]				CNAIP[2:0]		
		7:0		DMA7IP[2:0]				DMA6IP[2:0]		
0x0130 … 0x0133	Reserved									
0x0134	IPC17	31:24		ICDIP[2:0]				CRCIP[2:0]		
		23:16		BISS1IP[2:0]				BISS1EIP[2:0]		
		15:8								
		7:0						QEI1IP[2:0]		
0x0138	IPC18	31:24						PTG3IP[2:0]		
		23:16		PTG2IP[2:0]				PTG1IP[2:0]		
		15:8		PTG0IP[2:0]				PTGWDTIP[2:0]		
		7:0		PTGSTEPIP[2:0]						
0x013C	IPC19	31:24		AD1CH1IP[2:0]				AD1CMP0IP[2:0]		
		23:16		AD1CH0IP[2:0]						
		15:8								
		7:0								
0x0140	IPC20	31:24		AD1CH5IP[2:0]				AD1CMP4IP[2:0]		
		23:16		AD1CH4IP[2:0]				AD1CMP3IP[2:0]		
		15:8		AD1CH3IP[2:0]				AD1CMP2IP[2:0]		
		7:0		AD1CH2IP[2:0]				AD1CMP1IP[2:0]		
0x0144	IPC21	31:24								
		23:16								
		15:8						AD1CMP6IP[2:0]		
		7:0		AD1CH6IP[2:0]				AD1CMP5IP[2:0]		
0x0148	IPC22	31:24		AD2CH2IP[2:0]				AD2CMP1IP[2:0]		
		23:16		AD2CH1IP[2:0]				AD2CMP0IP[2:0]		
		15:8		AD2CH0IP[2:0]						
		7:0								
0x014C	IPC23	31:24		AD2CH6IP[2:0]				AD2CMP5IP[2:0]		
		23:16		AD2CH5IP[2:0]				AD2CMP4IP[2:0]		
		15:8		AD2CH4IP[2:0]				AD2CMP3IP[2:0]		
		7:0		AD2CH3IP[2:0]				AD2CMP2IP[2:0]		
0x0150	IPC24	31:24								
		23:16								
		15:8						AD2CMP7IP[2:0]		
		7:0		AD2CH7IP[2:0]				AD2CMP6IP[2:0]		
0x0154	IPC25	31:24		AD3CH3IP[2:0]				AD3CMP2IP[2:0]		
		23:16		AD3CH2IP[2:0]				AD3CMP1IP[2:0]		
		15:8		AD3CH1IP[2:0]				AD3CMP0IP[2:0]		
		7:0		AD3CH0IP[2:0]						
0x0158	IPC26	31:24		AD3CH7IP[2:0]				AD3CMP6IP[2:0]		
		23:16		AD3CH6IP[2:0]				AD3CMP5IP[2:0]		
		15:8		AD3CH5IP[2:0]				AD3CMP4IP[2:0]		
		7:0		AD3CH4IP[2:0]				AD3CMP3IP[2:0]		

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x015C	IPC27	31:24						AD3CMP10IP[2:0]		
		23:16		AD3CH11IP[2:0]				AD3CMP9IP[2:0]		
		15:8		AD3CH9IP[2:0]						
		7:0		AD3CH8IP[2:0]				AD3CMP8IP[2:0]		
0x0160	IPC28	31:24								
		23:16								
		15:8								
		7:0						AD3CMP11IP[2:0]		
0x0164 ... 0x0177	Reserved									
0x0178	IPC34	31:24		CLC2PIP[2:0]				CLC1NIP[2:0]		
		23:16		CLC1PIP[2:0]						
		15:8								
		7:0		CMP5IP[2:0]						
0x017C	IPC35	31:24								
		23:16						CLC4NIP[2:0]		
		15:8		CLC4PIP[2:0]				CLC3NIP[2:0]		
		7:0		CLC3PIP[2:0]				CLC2NIP[2:0]		
0x0180 ... 0x018F	Reserved									
0x0190	IPC40	31:24		ITCIP[2:0]						
		23:16								
		15:8								
		7:0								
0x0194	IPC41	31:24		IOM4IP[2:0]				IOM3IP[2:0]		
		23:16		IOM2IP[2:0]				IOM1IP[2:0]		
		15:8								
		7:0								
0x0198	IPC42	31:24								
		23:16								
		15:8		IOM8IP[2:0]				IOM7IP[2:0]		
		7:0		IOM6IP[2:0]				IOM5IP[2:0]		
0x019C ... 0x01A3	Reserved									
0x01A4	IPC45	31:24								
		23:16								
		15:8						I3C1DEIP[2:0]		
		7:0		I3CISEIP[2:0]				I3CIGIP[2:0]		
0x01A8	IPC46	31:24								
		23:16		PWRMIP[2:0]				CICERRIP[2:0]		
		15:8		CICOSIP[2:0]				CICIPDIP[2:0]		
		7:0		RDCERIP[2:0]				RDCCORIP[2:0]		

10.4.1. Interrupt Control Register 1

Name: INTCON1
Offset: 0x70

Note:
1. The user is responsible for clearing this bit by writing a zero to it.

Bit	31	30	29	28	27	26	25	24
	NSTDIS							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	GIE							
Access	R/W							
Reset	1							

Bit	7	6	5	4	3	2	1	0
				STKERR	ADDRERR	BADOPERR		
Access				R/W	R/W	R/W		
Reset				0	0	0		

Bit 31 – NSTDIS Interrupt Nesting Disable bit

Value	Description
1	Interrupt nesting is disabled.
0	Interrupt nesting is enabled.

Bit 15 – GIE Global Interrupt Enable bit

Value	Description
1	Interrupts are enabled (assuming associated IE bits are enabled).
0	Interrupts are disabled (traps are still enabled).

Bit 4 – STKERR Stack Error Trap Status bit(1)

Value	Description
1	Stack Error Trap has occurred.
0	Stack Error Trap has not occurred.

Bit 3 – ADDRERR Address Error Trap Status bit(1)

Value	Description
1	Address Error Trap has occurred.
0	Address Error Trap has not occurred.

Bit 2 – BADOPERR Illegal Opcode Error Trap Status bit(1)

Value	Description
1	Illegal Opcode Error Trap has occurred.

Value	Description
0	Illegal Opcode Error Trap has not occurred.

10.4.2. Interrupt Control Register 2

Name: INTCON2
Offset: 0x74

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 4 – INT4EP External Interrupt 4 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

Bit 3 – INT3EP External Interrupt 3 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

Bit 2 – INT2EP External Interrupt 2 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

Bit 1 – INT1EP External Interrupt 1 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

Bit 0 – INT0EP External Interrupt 0 Edge Detect Polarity Select bit

Value	Description
1	Interrupt on negative edge
0	Interrupt on positive edge

10.4.3. Interrupt Control Register 3

Name: INTCON3
Offset: 0x78

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					CPUBET	DMABET	YRAMBET	XRAMBET
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – CPUBET CPU Bus Error Trap Status bit 3 (CPU Inst Data bus error)

Value	Description
1	Bus Error trap 1 has occurred.
0	Bus Error trap 1 has not occurred.

Bit 2 – DMABET DMA Bus Error Trap Status bit 2 (DMA bus error)

Value	Description
1	Bus Error trap 1 has occurred.
0	Bus Error trap 1 has not occurred.

Bit 1 – YRAMBET YRAM Bus Error Trap Status bit 1 (CPU Y Data bus error)

Value	Description
1	Bus Error trap 1 has occurred.
0	Bus Error trap 1 has not occurred.

Bit 0 – XRAMBET XRAM Bus Error Trap Status bit 0 (CPU X Data bus error)

Value	Description
1	Bus Error trap 0 has occurred.
0	Bus Error trap 0 has not occurred.

10.4.4. Interrupt Control Register 4

Name: INTCON4
Offset: 0x7C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			OVATE	OVBTE	COVTE			
Access			R/W	R/W	R/W			
Reset			0	0	0			

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			OVAERR	OVBERR	COVAERR	COVBERR	SFTACERR	DIV0ERR
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 21 – OVATE Accumulator A Overflow Trap Enable bit

Value	Description
1	Enable Accumulator A overflow trap (OVAERR)
0	Trap is disabled.

Bit 20 – OVBTE Accumulator B Overflow Trap Enable bit

Value	Description
1	Enable Accumulator B overflow trap (OVBERR)
0	Trap is disabled.

Bit 19 – COVTE Catastrophic Overflow A/B Trap Enable bit

Value	Description
1	Enable trap on Catastrophic overflow of Accumulator A/B (COVAERR/COVBERR)
0	Trap is disabled.

Bit 5 – OVAERR Accumulator A Overflow Trap Flag bit

Value	Description
1	Trap was caused by overflow of Accumulator A.
0	Trap was not caused by overflow of Accumulator A.

Bit 4 – OVBERR Accumulator B Overflow Trap Flag bit

Value	Description
1	Trap was caused by overflow of Accumulator B.
0	Trap was not caused by overflow of Accumulator B.

Bit 3 – COVAERR Accumulator A Catastrophic Overflow Trap Flag bit

Value	Description
1	Trap was caused by catastrophic overflow of Accumulator A.
0	Trap was not caused by catastrophic overflow of Accumulator A.

Bit 2 – COVBERR Accumulator B Catastrophic Overflow Trap Flag bit

Value	Description
1	Trap was caused by catastrophic overflow of Accumulator B.
0	Trap was not caused by catastrophic overflow of Accumulator B.

Bit 1 – SFTACERR Shift Accumulator Error Status bit

Value	Description
1	Math error trap was caused by an invalid accumulator shift.
0	Math error trap was not caused by an invalid accumulator shift.

Bit 0 – DIV0ERR Arithmetic Divide-By-Zero Error Status bit

Value	Description
1	Math error trap occurred due to arithmetic divide by zero.
0	Math error trap due to divide-by-zero has not occurred.

10.4.5. Interrupt Control Register 5

Name: INTCON5
Offset: 0x80

Bit	31	30	29	28	27	26	25	24
	SOFT							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					YRAM	XRAM	WDTE	DMTE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 31 – SOFT Software Generated Soft Trap Status bit

Value	Description
1	Raise software generated Soft Trap
0	Soft trap has not occurred.

Bit 3 – YRAM Generic Trap Status bit (YRAM PWB DED Error Trap)

Value	Description
1	Generic trap 3 has occurred.
0	Generic trap 3 has not occurred.

Bit 2 – XRAM Generic Trap Status bit (XRAM PWB DED Error Trap)

Value	Description
1	Generic trap 2 has occurred.
0	Generic trap 2 has not occurred.

Bit 1 – WDTE Generic Trap Status bit (WDT Run Event Trap)

Value	Description
1	Generic trap 1 has occurred.
0	Generic trap 1 has not occurred.

Bit 0 – DMTE Generic Trap Status bit (DMT: Deadman Timer)

Value	Description
1	Generic trap 0 has occurred.
0	Generic trap 0 has not occurred.

10.4.6. Interrupt Control and Status Register(1)

Name: INTTREG
Offset: 0x84

Note:
1. The bit fields in the INTTREG register correspond to the value of the vector number, interrupt level and IRQ request flag, when an interrupt is presented to the CPU.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			VHOLD					
Access			R/W					
Reset			0					

Bit	15	14	13	12	11	10	9	8
			ILR[3:0]					VECNUM[8]
Access			R	R	R	R		R
Reset			0	0	0	0		0

Bit	7	6	5	4	3	2	1	0
	VECNUM[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 21 – VHOLD Vector Number Capture Enable bit

Value	Description
1	VECNUM[8:0] bits read the current value of the vector number encoding tree (i.e., highest priority pending interrupt).
0	Vector number latched into VECNUM[8:0] at Interrupt Acknowledge and retained until the next IACK.

Bits 13:10 – ILR[3:0] CPU Interrupt Priority Level bits

Bits 8:0 – VECNUM[8:0] Vector Number of Pending Interrupt bits
Vector number of pending interrupt or last acknowledged interrupt. VECNUM = IRQ + 9.

10.4.7. Interrupt Vector Base Address Register

Name: IVTBASE
Offset: 0x88

Bit	31	30	29	28	27	26	25	24
	IVTBASE[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	IVTBASE[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IVTBASE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	IVTBASE[7:6]							
Access	R/W	R/W						
Reset	0	0						

Bits 31:6 – IVTBASE[31:6] Interrupt Vector Table Base Address bits

10.4.8. Interrupt Vector Collapse Register

Name: IVTCREG
Offset: 0x8C

Note: This register can be write-protected or locked using the corresponding bits in the PACCON register. Refer to the Peripheral Access Controller (PAC) for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								IVTC
Access								R/W
Reset								0

Bit 0 – IVTC Interrupt Vector Table Collapse bit

Value	Description
1	Enables Interrupt Vector Table collapse
0	Interrupt Vector Table collapse disabled

10.4.9. Interrupt Request Flags Register 0

Name: IFS0
Offset: 0x90

Bit	31	30	29	28	27	26	25	24
	CRYPT2IF	CRYPT1IF		WDTIF		C4RDYIF	C4MONIF	C4WARNIF
Access	RW	RW		RW		RW	RW	RW
Reset	0	0		0		0	0	0

Bit	23	22	21	20	19	18	17	16
	C4FAILIF	C3RDYIF	C3MONIF	C3WARNIF	C3FAILIF	C2RDYIF	C2MONIF	C2WARNIF
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	C2FAILIF	C1RDYIF	C1MONIF	C1WARNIF	C1FAILIF	CLKERRIF	CLKFAILIF	
Access	RW	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0
	NVMCRCIF	NVMIF	NVMECCIF	PBERRIF	YRAMECCIF	XRAMECCIF	CPUFPUIF	IVTCIF
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit 31 – CRYPT2IF Crypto Module Interrupt Flag bit 2

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 30 – CRYPT1IF Crypto Module Interrupt Flag bit 1

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 28 – WDTIF Watchdog Timer Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 26 – C4RDYIF Count Ready Interrupt 4 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 25 – C4MONIF Clock Monitor Overflow Interrupt 4 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 24 – C4WARNIF Clock Warning Interrupt 4 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 23 – C4FAILIF Clock Failure Interrupt 4 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 22 – C3RDYIF Count Ready Interrupt 3 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 21 – C3MONIF Clock Monitor Overflow 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 20 – C3WARNIF Clock Warning Interrupt 3 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – C3FAILIF Clock Failure Interrupt 3 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – C2RDYIF Count Ready Interrupt 2 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – C2MONIF Clock Monitor Overflow 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 16 – C2WARNIF Clock Warning Interrupt 2 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 15 – C2FAILIF Clock Failure Interrupt 2 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 14 – C1RDYIF Count Ready Interrupt 1 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 13 – C1MONIF Clock Monitor Overflow 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 12 – C1WARNIF Clock Warning Interrupt 1 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 11 – C1FAILIF Clock Failure Interrupt 1 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 10 – CLKERRIF Clock Error Interrupt (combined) Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 9 – CLKFAILIF Clock Fail Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 7 – NVMCRCIF NVM CRC Operation Completed Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 6 – NVMIF NVM Program/Erase Op Completed or Terminated bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 5 – NVMECCIF NVM Data ECC SEC and/or Instruction SEC Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 4 – PBERRIF PBU Parity Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 3 – YRAMECCIF YRAM Data ECC SEC Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 2 – XRAMECCIF XRAM Data ECC SEC Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 1 – CPUFPUIF CPU/FPU Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 0 – IVTCIF Interrupt Vector Table Collapse Interrupt Flag bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

10.4.10. Interrupt Request Flags Register 1

Name: IFS1
Offset: 0x94

Bit	31	30	29	28	27	26	25	24
			C1IF	C1TXIF	C1RXIF	CCP4IF	CCT4IF	CCP3IF
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CCT3IF	CCP2IF	CCT2IF	CCP1IF	CCT1IF	T3IF	T2IF	T1IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PWM4IF	PWM3IF	PWM2IF	PWM1IF	PEVTFIF	PEVTEIF	PEVTDIF	PEVTCIF
Access	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PEVTBIF	PEVTAIF	INT4IF	INT3IF	INT2IF	INT1IF	INT0IF	CRYPT3IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 29 – C1IF CAN1 Combined Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 28 – C1TXIF CAN1 RX Transmit Interrupt Flag bit

Value	Description
1	Interrupt enabled.
0	Interrupt not enabled.

Bit 27 – C1RXIF CAN1 RX Data Ready Interrupt Flag bit

Value	Description
1	Interrupt enabled.
0	Interrupt not enabled.

Bit 26 – CCP4IF Input Capture/Output Compare 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 25 – CCT4IF Capture/Compare/Timer 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 24 – CCP3IF Input Capture/Output Compare 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 23 – CCT3IF Capture/Compare/Timer 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 22 – CCP2IF Input Capture/Output Compare 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 21 – CCT2IF Capture/Compare/Timer 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 20 – CCP1IF Input Capture/Output Compare 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – CCT1IF Capture/Compare/Timer 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – T3IF Timer3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – T2IF Timer3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 16 – T1IF Timer1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 15 – PWM4IF PWM4 Parameter Interrupt Flag bit

Value	Description
1	PWM4 Parameter interrupt has occurred (must be cleared in software).
0	PWM4 Parameter interrupt event has not occurred.

Bit 14 – PWM3IF PWM3 Parameter Interrupt Flag bit

Value	Description
1	PWM4 Parameter interrupt has occurred (must be cleared in software).
0	PWM4 Parameter interrupt event has not occurred.

Bit 13 – PWM2IF PWM2 Parameter Interrupt Flag bit

Value	Description
1	PWM4 Parameter interrupt has occurred (must be cleared in software).
0	PWM4 Parameter interrupt event has not occurred.

Bit 12 – PWM1IF PWM1 Parameter Interrupt Flag bit

Value	Description
1	PWM4 Parameter interrupt has occurred (must be cleared in software).
0	PWM4 Parameter interrupt event has not occurred.

Bit 11 – PEVTFIF PWM Event F Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 10 – PEVTEIF PWM Event E Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 9 – PEVTDIF PWM Event D Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 8 – PEVTCIF PWM Event C Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 7 – PEVTBIF PWM Event B Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 6 – PEVTAIF PWM Event A Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 5 – INT4IF External Interrupt 4 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 4 – INT3IF External Interrupt 3 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 3 – INT2IF External Interrupt 2 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 2 – INT1IF External Interrupt 1 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 1 – INT0IF External Interrupt 0 Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 0 – CRYPT3IF Crypto Module 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

10.4.11. Interrupt Request Flags Register 2

Name: IFS2
Offset: 0x98

Bit	31	30	29	28	27	26	25	24
				I2C2TXIF	I2C2RXIF	I2C2IF	I2C2EIF	I2C1TXIF
Access				R	R	R/W	R	R
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	I2C1RXIF	I2C1IF	I2C1EIF	CMP4IF	CMP3IF	CMP2IF	CMP1IF	DMA3IF
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DMA2IF	DMA1IF	DMA0IF				SPI3EIF	SPI3TXIF
Access	R/W	R/W	R/W				R/W	R
Reset	0	0	0				0	0

Bit	7	6	5	4	3	2	1	0
	SPI3RXIF	SPI2EIF	SPI2RXIF	SPI2RXIF	SPI1EIF	SPI1TXIF	SPI1RXIF	
Access	R	R/W	R/W	R	R/W	R/W	R	
Reset	0	0	0	0	0	0	0	

Bit 28 – I2C2TXIF I2C2 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 27 – I2C2RXIF I2C2 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 26 – I2C2IF I2C2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 25 – I2C2EIF I2C2 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 24 – I2C1TXIF I2C1 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 23 – I2C1RXIF I2C1 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 22 – I2C1IF I2C1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 21 – I2C1EIF I2C1 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 20 – CMP4IF Comparator 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – CMP3IF Comparator 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – CMP2IF Comparator 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – CMP1IF Comparator 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 16 – DMA3IF Direct Memory Access 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 15 – DMA2IF Direct Memory Access 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 14 – DMA1IF Direct Memory Access 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 13 – DMA0IF Direct Memory Access 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 9 – SPI3EIF SPI3 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 8 – SPI3TXIF SPI3 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 7 – SPI3RXIF SPI3 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 6 – SPI2EIF SPI2 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 5 – SPI2RXIF SPI2 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 4 – SPI2RXIF SPI2 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 3 – SPI1EIF SPI1 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 2 – SPI1TXIF SPI1 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 1 – SPI1RXIF SPI1 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

10.4.12. Interrupt Request Flags Register 3

Name: IFS3
Offset: 0x9C

Bit	31	30	29	28	27	26	25	24
	CCP5IF	CCT5IF	CNDIF	CNCIF	CNBIF	CNAIF	DMA7IF	DMA6IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMA5IF	DMA4IF	SENT2EIF	SENT2IF	SENT1EIF	SENT1IF	U4EVTIF	U4EIF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	U4TXIF	U4RXIF	U3EVTIF	U3EIF	U3TXIF	U3RXIF	U2EVTIF	U2EIF
Access	R	R	R/W	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U2TXIF	U2RXIF	U1EVTIF	U1EIF	U1TXIF	U1RXIF		I2C3TXIF
Access	R	R	R/W	R	R	R		R
Reset	0	0	0	0	0	0		0

Bit 31 – CCP5IF Input Capture/Output Compare 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 30 – CCT5IF Capture/Compare/Timer5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 29 – CNDIF Change Notice D Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 28 – CNCIF Change Notice C Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 27 – CNBIF Change Notice B Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 26 – CNAIF Change Notice A Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 25 – DMA7IF Direct Memory Access 7 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 24 – DMA6IF Direct Memory Access 6 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 23 – DMA5IF Direct Memory Access 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 22 – DMA4IF Direct Memory Access 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 21 – SENT2EIF SENT2 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 20 – SENT2IF SENT2 TX/RX Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – SENT1EIF SENT1 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – SENT1IF SENT1 TX/RX Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – U4EVTIF UART4 Event Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 16 – U4EIF UART4 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 15 – U4TXIF UART4 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 14 – U4RXIF UART4 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 13 – U3EVTIF UART3 Event Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 12 – U3EIF UART3 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 11 – U3TXIF UART3 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 10 – U3RXIF UART3 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 9 – U2EVTIF UART2 Event Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 8 – U2EIF UART2 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 7 – U2TXIF UART2 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 6 – U2RXIF UART2 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 5 – U1EVTIF UART1 Event Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 4 – U1EIF UART1 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 3 – U1TXIF UART1 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 2 – U1RXIF UART1 Receive Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 0 – I2C3TXIF I2C3 Transmit Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

10.4.13. Interrupt Request Flags Register 4

Name: IFS4
Offset: 0xA0

Bit	31	30	29	28	27	26	25	24
	AD1CH1IF	AD1CMP0IF	AD1CH0IF					
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
		PTG3IF	PTG2IF	PTG1IF	PTG0IF	PTGWDTIF	PTGSTEPIF	
Access		R/W	R/W	R/W	R/W	R/W	R/W	
Reset		0	0	0	0	0	0	

Bit	15	14	13	12	11	10	9	8
	ICDIF	CRCIF	BISS1IF	BISS1EIF				QEI1IF
Access	R/W	R/W/HS	R/W/HS	R/W/HS				R/W
Reset	0	0	0	0				0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 31 – AD1CH1IF ADC 1 Channel 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 30 – AD1CMP0IF ADC Digital Comparator 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 29 – AD1CH0IF ADC 1 Channel 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 22 – PTG3IF PTG3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 21 – PTG2IF PTG2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 20 – PTG1IF PTG1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – PTG0IF PTG0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – PTGWDTIF PTG Watchdog Timer Time-out Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – PTGSTEPIF PTG Step Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 15 – ICDIF In-Circuit Debugger Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 14 – CRCIF CRC Interrupt Flag bit

Value	Description
1	Interrupt has occurred (must be cleared by software).
0	Interrupt event has not occurred.

Bit 13 – BISS1IF BiSS 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred (must be cleared by software).
0	Interrupt event has not occurred.

Bit 12 – BISS1EIF BiSS 1 Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred (must be cleared by software).
0	Interrupt event has not occurred.

Bit 8 – QEI1IF QEI1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

10.4.14. Interrupt Request Flags Register 5

Name: IFS5
Offset: 0xA4

Bit	31	30	29	28	27	26	25	24
	AD2CH6IF	AD2CMP5IF	AD2CH5IF	AD2CMP4IF	AD2CH4IF	AD2CMP3IF	AD2CH3IF	AD2CMP2IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	AD2CH2IF	AD2CMP1IF	AD2CH1IF	AD2CMP0IF	AD2CH0IF			
Access	R/W	R/W	R/W	R/W	R/W			
Reset	0	0	0	0	0			

Bit	15	14	13	12	11	10	9	8
						AD1CMP6IF	AD1CH6IF	AD1CMP5IF
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
	AD1CH5IF	AD1CMP4IF	AD1CH4IF	AD1CMP3IF	AD1CH3IF	AD1CMP2IF	AD1CH2IF	AD1CMP1IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – AD2CH6IF ADC 2 Channel 6 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 30 – AD2CMP5IF ADC Digital Comparator 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 29 – AD2CH5IF ADC 2 Channel 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 28 – AD2CMP4IF ADC Digital Comparator 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 27 – AD2CH4IF ADC 2 Channel 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 26 – AD2CMP3IF ADC Digital Comparator 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 25 – AD2CH3IF ADC 2 Channel 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 24 – AD2CMP2IF ADC Digital Comparator 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 23 – AD2CH2IF ADC 2 Channel 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 22 – AD2CMP1IF ADC Digital Comparator 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 21 – AD2CH1IF ADC 2 Channel 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 20 – AD2CMP0IF ADC Digital Comparator 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – AD2CH0IF ADC 2 Channel 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 10 – AD1CMP6IF ADC Digital Comparator 6 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 9 – AD1CH6IF ADC 1 Channel 6 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 8 – AD1CMP5IF ADC Digital Comparator 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 7 – AD1CH5IF ADC 1 Channel 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 6 – AD1CMP4IF ADC Digital Comparator 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 5 – AD1CH4IF ADC 1 Channel 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 4 – AD1CMP3IF ADC Digital Comparator 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 3 – AD1CH3IF ADC 1 Channel 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 2 – AD1CMP2IF ADC Digital Comparator 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 1 – AD1CH2IF ADC 1 Channel 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 0 – AD1CMP1IF ADC Digital Comparator 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

10.4.15. Interrupt Request Flags Register 6

Name: IFS6
Offset: 0xA8

Bit	31	30	29	28	27	26	25	24
	AD3CH11IF	AD3CMP10IF	AD3CH10IF	AD3CMP9IF	AD3CH9IF	AD3CMP8IF	AD3CH8IF	AD3CMP7IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	AD3CH7IF	AD3CMP6IF	AD3CH6IF	AD3CMP5IF	AD3CH5IF	AD3CMP4IF	AD3CH4IF	AD3CMP3IF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AD3CH3IF	AD3CMP2IF	AD3CH2IF	AD3CMP1IF	AD3CH1IF	AD3CMP0IF	AD3CH0IF	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0
						AD2CMP7IF	AD2CH7IF	AD2CMP6IF
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 31 – AD3CH11IF ADC 3 Channel 11 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 30 – AD3CMP10IF ADC Digital Comparator 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 29 – AD3CH10IF ADC 3 Channel 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 28 – AD3CMP9IF ADC Digital Comparator 9 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 27 – AD3CH9IF ADC 3 Channel 9 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 26 – AD3CMP8IF ADC Digital Comparator 8 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 25 – AD3CH8IF ADC 3 Channel 8 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 24 – AD3CMP7IF ADC Digital Comparator 7 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 23 – AD3CH7IF ADC 3 Channel 7 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 22 – AD3CMP6IF ADC Digital Comparator 6 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 21 – AD3CH6IF ADC 3 Channel 6 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 20 – AD3CMP5IF ADC Digital Comparator 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – AD3CH5IF ADC 3 Channel 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – AD3CMP4IF ADC Digital Comparator 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – AD3CH4IF ADC 3 Channel 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 16 – AD3CMP3IF ADC Digital Comparator 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 15 – AD3CH3IF ADC 3 Channel 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 14 – AD3CMP2IF ADC Digital Comparator 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 13 – AD3CH2IF ADC 3 Channel 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 12 – AD3CMP1IF ADC Digital Comparator 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 11 – AD3CH1IF ADC 3 Channel 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 10 – AD3CMP0IF ADC Digital Comparator 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 9 – AD3CH0IF ADC 3 Channel 0 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 2 – AD2CMP7IF ADC Digital Comparator 7 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 1 – AD2CH7IF ADC 2 Channel 7 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 0 – AD2CMP6IF ADC Digital Comparator 6 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

10.4.16. Interrupt Request Flags Register 7

Name: IFS7
Offset: 0xAC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								AD3CMP11IF
Access								R/W
Reset								0

Bit 0 – AD3CMP11IF ADC Digital Comparator 11 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

10.4.17. Interrupt Request Flags Register 8

Name: IFS8
Offset: 0xB0

Bit	31	30	29	28	27	26	25	24
				CLC4NIF	CLC4PIF	CLC3NIF	CLC3PIF	CLC2NIF
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLC2PIF	CLC1NIF	CLC1PIF				CMP5IF	
Access	R/W	R/W	R/W				R/W	
Reset	0	0	0				0	

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 28 – CLC4NIF CLC4 Negative Edge Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 27 – CLC4PIF CLC4 Positive Edge Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 26 – CLC3NIF CLC3 Negative Edge Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 25 – CLC3PIF CLC3 Positive Edge Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 24 – CLC2NIF CLC2 Negative Edge Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 23 – CLC2PIF CLC2 Positive Edge Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 22 – CLC1NIF CLC1 Negative Edge Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 21 – CLC1PIF CLC1 Positive Edge Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – CMP5IF Comparator 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

10.4.18. Interrupt Request Flags Register 10

Name: IFS10
Offset: 0xB8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					IOM8IF	IOM7IF	IOM6IF	IOM5IF
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IOM4IF	IOM3IF	IOM2IF	IOM1IF				
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit	7	6	5	4	3	2	1	0
	ITCIF							
Access	R/W							
Reset	0							

Bit 19 – IOM8IF IO Monitor 8 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – IOM7IF IO Monitor 7 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – IOM6IF IO Monitor 6 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 16 – IOM5IF IO Monitor 5 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 15 – IOM4IF IO Monitor 4 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 14 – IOM3IF IO Monitor 3 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 13 – IOM2IF IO Monitor 2 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 12 – IOM1IF IO Monitor 1 Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 7 – ITCIF Touch Controller Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

10.4.19. Interrupt Request Flags Register 11

Name: IFS11
Offset: 0xBC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			PWRMIF	CICERRIF	CICOSRIF	CICIPDIF	RDCERRIF	RDCCORIF
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						I3C1DEIF	I3C1SEIF	I3C1GIF
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 21 – PWRMIF Power Monitor Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 20 – CICERRIF CIC Error Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – CICOSRIF CIC Over Sample Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – CICIPDIF CIC Processing Done Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – RDCERRIF RDC Error Condition Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 16 – RDCCORIF RDC CORDIC Operation Done Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 10 – I3C1DEIF I3C1 DERR Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 9 – I3C1SEIF I3C1 SERR Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 8 – I3C1GIF I3C Global Interrupt Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

10.4.20. Interrupt Enable Register 0

Name: IEC0
Offset: 0xC0

Bit	31	30	29	28	27	26	25	24
	CRYPT2IE	CRYPT1IE		WDTIE		C4RDYIE	C4MONIE	C4WARNIE
Access	RW	RW		RW		RW	RW	RW
Reset	0	0		0		0	0	0

Bit	23	22	21	20	19	18	17	16
	C4FAILIE	C3RDYIE	C3MONIE	C3WARNIE	C3FAILIE	C2RDYIE	C2MONIE	C2WARNIE
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	C2FAILIE	C1RDYIE	C1MONIE	C1WARNIE	C1FAILIE	CLKERRIE	CLKFAILIE	
Access	RW	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0
	NVMCRCIE	NVMIE	NVMECCIE	PBERRIE	YRAMECCIE	XRAMECCIE	CPUFPUIE	
Access	RW	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	

Bit 31 – CRYPT2IE Crypto Module Interrupt 2 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 30 – CRYPT1IE Crypto Module Interrupt 1 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 28 – WDTIE Watchdog Timer Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 26 – C4RDYIE Count Ready Interrupt 4 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 25 – C4MONIE Clock Monitor Overflow Interrupt 4 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 24 – C4WARNIE Clock Warning Interrupt 4 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 23 – C4FAILIE Clock Failure Interrupt 4 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 22 – C3RDYIE Count Ready Interrupt 3 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 21 – C3MONIE Clock Monitor Overflow Interrupt 3 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 20 – C3WARNIE Clock Warning Interrupt 3 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 19 – C3FAILIE Clock Failure Interrupt 3 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 18 – C2RDYIE Count Ready Interrupt 2 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 17 – C2MONIE Clock Monitor Overflow Interrupt 2 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 16 – C2WARNIE Clock Warning Interrupt 2 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 15 – C2FAILIE Clock Failure Interrupt 2 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 14 – C1RDYIE Count Ready Interrupt 1 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 13 – C1MONIE Clock Monitor Overflow Interrupt 1 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 12 – C1WARNIE Clock Warning Interrupt 1 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 11 – C1FAILIE Clock Failure Interrupt 2 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 10 – CLKERRIE Clock Error Interrupt (combined) Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 9 – CLKFAILIE Clock Fail Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 7 – NVMCRCIE NVM CRC Operation Completed Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 6 – NVMIE NVM Program/Erase Op Completed or Terminated bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 5 – NVMECCIE NVM Data ECC SEC and/or Instruction SEC Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 4 – PBERRIE PBU Parity Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 3 – YRAMECCIE YRAM Data ECC SEC Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 2 – XRAMECCIE XRAM Data ECC SEC Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 1 – CPUFPUIE CPU/FPU Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

10.4.21. Interrupt Enable Register 1

Name: IEC1
Offset: 0xC4

Bit	31	30	29	28	27	26	25	24
			C1IE	C1TXIE	C1RXIE	CCP4IE	CCT4IE	CCP3IE
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CCT3IE	CCP2IE	CCT2IE	CCP1IE	CCT1IE	T3IE	T2IE	T1IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PWM4IE	PWM3IE	PWM2IE	PWM1IE	PEVTFIE	PEVTEIE	PEVTDIE	PEVTCIE
Access	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PEVTBIE	PEVTAIE	INT4IE	INT3IE	INT2IE	INT1IE	INT0IE	CRYPT3IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 29 – C1IE CAN1 Combined Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 28 – C1TXIE CAN1 RX Transmit Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 27 – C1RXIE CAN1 RX Data Ready Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 26 – CCP4IE Input Capture/Output Compare 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 25 – CCT4IE Capture/Compare/Timer 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 24 – CCP3IE Input Capture/Output Compare 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 23 – CCT3IE Capture/Compare/Timer 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 22 – CCP2IE Input Capture/Output Compare 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 21 – CCT2IE Capture/Compare/Timer 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 20 – CCP1IE Input Capture/Output Compare 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 19 – CCT1IE Capture/Compare/Timer 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 18 – T3IE Timer3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 17 – T2IE Timer2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 16 – T1IE Timer1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 15 – PWM4IE PWM4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 14 – PWM3IE PWM3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 13 – PWM2IE PWM2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 12 – PWM1IE PWM1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 11 – PEVTFIE PWM Event F Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 10 – PEVTEIE PWM Event E Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 9 – PEVTDIE PWM Event D Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 8 – PEVTCIE PWM Event C Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 7 – PEVTBIE PWM Event B Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 6 – PEVTAIE PWM Event A Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 5 – INT4IE External Interrupt 4 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 4 – INT3IE External Interrupt 3 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 3 – INT2IE External Interrupt 2 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 2 – INT1IE External Interrupt 1 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 1 – INT0IE External Interrupt 0 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 0 – CRYPT3IE Crypto Module Interrupt 3 Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.22. Interrupt Enable Register 2

Name: IEC2
Offset: 0xC8

Bit	31	30	29	28	27	26	25	24
				I2C2TXIE	I2C2RXIE	I2C2IE	I2C2EIE	I2C1TXIE
Access				R	R	R/W	R	R
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	I2C1RXIE	I2C1IE	I2C1EIE	CMP4IE	CMP3IE	CMP2IE	CMP1IE	DMA3IE
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DMA2IE	DMA1IE	DMA0IE				SPI3EIE	SPI3TXIE
Access	R/W	R/W	R/W				R/W	R
Reset	0	0	0				0	0

Bit	7	6	5	4	3	2	1	0
	SPI3RXIE	SPI2EIE	SPI2TXIE	SPI2RXIE	SPI1EIE	SPI1TXIE	SPI1RXIE	
Access	R	R/W	R/W	R	R/W	R/W	R	
Reset	0	0	0	0	0	0	0	

Bit 28 – I2C2TXIE I2C2 Transmit Interrupt Enable Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 27 – I2C2RXIE I2C2 Receive Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 26 – I2C2IE I2C2 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 25 – I2C2EIE I2C2 Error Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 24 – I2C1TXIE I2C1 Transmit Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 23 – I2C1RXIE I2C1 Receive Interrupt Enable Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 22 – I2C1IE I2C1 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 21 – I2C1EIE I2C1 Error Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 20 – CMP4IE Comparator 4 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 19 – CMP3IE Comparator 3 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 18 – CMP2IE Comparator 2 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 17 – CMP1IE Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 16 – DMA3IE Direct Memory Access 3 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 15 – DMA2IE Direct Memory Access 2 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 14 – DMA1IE Direct Memory Access 1 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 13 – DMA0IE Direct Memory Access 0 Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 9 – SPI3EIE SPI3 Error Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 8 – SPI3TXIE SPI3 Transmit Interrupt Enable Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 7 – SPI3RXIE SPI3 Receive Interrupt Enable Flag bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 6 – SPI2EIE SPI2 Error Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 5 – SPI2TXIE SPI2 Transmit Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 4 – SPI2RXIE SPI2 Receive Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

Bit 3 – SPI1EIE SPI1 General Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 2 – SPI1TXIE SPI1 Transmit Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt has not occurred.

Bit 1 – SPI1RXIE SPI1 Receive Interrupt Enable bit

Value	Description
1	Interrupt has occurred.
0	Interrupt event has not occurred.

10.4.23. Interrupt Enable Register 3

Name: IEC3
Offset: 0xCC

Bit	31	30	29	28	27	26	25	24
	CCP5IE	CCT5IE	CNDIE	CNCIE	CNBIE	CNAIE	DMA7IE	DMA6IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMA5IE	DMA4IE	SENT2EIE	SENT2IE	SENT1EIE	SENT1IE	U4EVTIE	U4EIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	U4TXIE	U4RXIE	U3EVTIE	U3EIE	U3TXIE	U3RXIE	U2EVTIE	U2EIE
Access	R	R	R/W	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U2TXIE	U2RXIE	U1EVTIE	U1EIE	U1TXIE	U1RXIE		
Access	R	R	R/W	R	R	R		
Reset	0	0	0	0	0	0		

Bit 31 – CCP5IE Input Capture/Output Compare 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 30 – CCT5IE Capture/Compare/Timer5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 29 – CNDIE Change Notice D Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 28 – CNCIE Change Notice C Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 27 – CNBIE Change Notice B Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 26 – CNAIE Change Notice A Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 25 – DMA7IE Direct Memory Access 7 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 24 – DMA6IE Direct Memory Access 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 23 – DMA5IE Direct Memory Access 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 22 – DMA4IE Direct Memory Access 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 21 – SENT2EIE SENT2 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 20 – SENT2IE SENT2 TX/RX Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 19 – SENT1EIE SENT1 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 18 – SENT1IE SENT1 TX/RX Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 17 – U4EVTIE UART4 Event Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 16 – U4EIE UART4 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 15 – U4TXIE UART4 Transmit Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 14 – U4RXIE UART4 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 13 – U3EVTIE UART3 Event Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 12 – U3EIE UART3 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 11 – U3TXIE UART3 Transmit Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 10 – U3RXIE UART3 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 9 – U2EVTIE UART2 Event Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 8 – U2EIE UART2 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 7 – U2TXIE UART2 Transmit Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 6 – U2RXIE UART2 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 5 – U1EVTIE UART1 Event Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 4 – U1EIE UART1 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 3 – U1TXIE UART1 Transmit Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 2 – U1RXIE UART1 Receive Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.24. Interrupt Enable Register 4

Name: IEC4
Offset: 0xD0

Bit	31	30	29	28	27	26	25	24
	AD1CH1IE	AD1CMP0IE	AD1CH0IE					
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
		PTG3IE	PTG2IE	PTG1IE	PTG0IE	PTGWDTIE	PTGSTEPIE	
Access		R/W	R/W	R/W	R/W	R/W	R/W	
Reset		0	0	0	0	0	0	

Bit	15	14	13	12	11	10	9	8
	ICDIE	CRCIE	BISS1IE	BISS1EIE				QEI1IE
Access	R/W	R/W/HS	R/W/HS	R/W/HS				R/W
Reset	0	0	0	0				0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 31 – AD1CH1IE ADC 1 Channel 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 30 – AD1CMP0IE ADC Digital Comparator 0 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 29 – AD1CH0IE ADC 1 Channel 0 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 22 – PTG3IE PTG3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 21 – PTG2IE PTG2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 20 – PTG1IE PTG1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 19 – PTG0IE PTG0 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 18 – PTGWDTIE PTG Watchdog Timer Time-out Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 17 – PTGSTEPIE PTG Step Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 15 – ICDIE In-Circuit Debugger Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 14 – CRCIE CRC Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 13 – BISS1IE BiSS 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 12 – BISS1EIE BiSS 1 Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 8 – QEI1IE QEI1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.25. Interrupt Enable Register 5

Name: IEC5
Offset: 0xD4

Bit	31	30	29	28	27	26	25	24
	AD2CH6IE	AD2CMP5IE	AD2CH5IE	AD2CMP4IE	AD2CH4IE	AD2CMP3IE	AD2CH3IE	AD2CMP2IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	AD2CH2IE	AD2CMP1IE	AD2CH1IE	AD2CMP0IE	AD2CH0IE			
Access	R/W	R/W	R/W	R/W	R/W			
Reset	0	0	0	0	0			

Bit	15	14	13	12	11	10	9	8
						AD1CMP6IE	AD1CH6IE	AD1CMP5IE
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
	AD1CH5IE	AD1CMP4IE	AD1CH4IE	AD1CMP3IE	AD1CH3IE	AD1CMP2IE	AD1CH2IE	AD1CMP1IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – AD2CH6IE ADC 2 Channel 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 30 – AD2CMP5IE ADC Digital Comparator 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 29 – AD2CH5IE ADC 2 Channel 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 28 – AD2CMP4IE ADC Digital Comparator 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 27 – AD2CH4IE ADC 2 Channel 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 26 – AD2CMP3IE ADC Digital Comparator 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 25 – AD2CH3IE ADC 2 Channel 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 24 – AD2CMP2IE ADC Digital Comparator 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 23 – AD2CH2IE ADC 2 Channel 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 22 – AD2CMP1IE ADC Digital Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 21 – AD2CH1IE ADC 2 Channel 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 20 – AD2CMP0IE ADC Digital Comparator 0 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 19 – AD2CH0IE ADC 2 Channel 0 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 10 – AD1CMP6IE ADC Digital Comparator 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 9 – AD1CH6IE ADC 1 Channel 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 8 – AD1CMP5IE ADC Digital Comparator 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 7 – AD1CH5IE ADC 1 Channel 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 6 – AD1CMP4IE ADC Digital Comparator 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 5 – AD1CH4IE ADC 1 Channel 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 4 – AD1CMP3IE ADC Digital Comparator 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 3 – AD1CH3IE ADC 1 Channel 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 2 – AD1CMP2IE ADC Digital Comparator 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 1 – AD1CH2IE ADC 1 Channel 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 0 – AD1CMP1IE ADC Digital Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.26. Interrupt Enable Register 6

Name: IEC6
Offset: 0xD8

Bit	31	30	29	28	27	26	25	24
	AD3CH11IE	AD3CMP10IE	AD3CH10IE	AD3CMP9IE	AD3CH9IE	AD3CMP8IE	AD3CH8IE	AD3CMP7IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	AD3CH7IE	AD3CMP6IE	AD3CH6IE	AD3CMP5IE	AD3CH5IE	AD3CMP4IE	AD3CH4IE	AD3CMP3IE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	AD3CH3IE	AD3CMP2IE	AD3CH2IE	AD3CMP1IE	AD3CH1IE	AD3CMP0IE	AD3CH0IE	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0
						AD2CMP7IE	AD2CH7IE	AD2CMP6IE
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 31 – AD3CH11IE ADC 3 Channel 11 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 30 – AD3CMP10IE ADC Digital Comparator 10 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 29 – AD3CH10IE ADC 3 Channel 10 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 28 – AD3CMP9IE ADC Digital Comparator 9 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 27 – AD3CH9IE ADC 3 Channel 9 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 26 – AD3CMP8IE ADC Digital Comparator 8 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 25 – AD3CH8IE ADC 3 Channel 8 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 24 – AD3CMP7IE ADC Digital Comparator 7 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 23 – AD3CH7IE ADC 3 Channel 7 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 22 – AD3CMP6IE ADC Digital Comparator 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 21 – AD3CH6IE ADC 3 Channel 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 20 – AD3CMP5IE ADC Digital Comparator 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 19 – AD3CH5IE ADC 3 Channel 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 18 – AD3CMP4IE ADC Digital Comparator 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 17 – AD3CH4IE ADC 3 Channel 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 16 – AD3CMP3IE ADC Digital Comparator 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 15 – AD3CH3IE ADC 3 Channel 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 14 – AD3CMP2IE ADC Digital Comparator 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 13 – AD3CH2IE ADC 3 Channel 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 12 – AD3CMP1IE ADC Digital Comparator 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 11 – AD3CH1IE ADC 3 Channel 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 10 – AD3CMP0IE ADC Digital Comparator 0 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 9 – AD3CH0IE ADC 3 Channel 0 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 2 – AD2CMP7IE ADC Digital Comparator 7 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 1 – AD2CH7IE ADC 2 Channel 7 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 0 – AD2CMP6IE ADC Digital Comparator 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.27. Interrupt Enable Register 7

Name: IEC7
Offset: 0xDC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								AD3CMP11IE
Access								R/W
Reset								0

Bit 0 – AD3CMP11IE ADC Digital Comparator 11 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.28. Interrupt Enable Register 8

Name: IEC8
Offset: 0xE0

Bit	31	30	29	28	27	26	25	24
				CLC4NIE	CLC4PIE	CLC3NIE	CLC3PIE	CLC2NIE
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLC2PIE	CLC1NIE	CLC1PIE				CMP5IE	
Access	R/W	R/W	R/W				R/W	
Reset	0	0	0				0	

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 28 – CLC4NIE CLC4 Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 27 – CLC4PIE CLC4 Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 26 – CLC3NIE CLC3 Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 25 – CLC3PIE CLC3 Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 24 – CLC2NIE CLC2 Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 23 – CLC2PIE CLC2 Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 22 – CLC1NIE CLC1 Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 21 – CLC1PIE CLC1 Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 17 – CMP5IE Comparator 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.29. Interrupt Enable Register 10

Name: IEC10
Offset: 0xE8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					IOM8IE	IOM7IE	IOM6IE	IOM5IE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IOM4IE	IOM3IE	IOM2IE	IOM1IE				
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit	7	6	5	4	3	2	1	0
	ITCIE							
Access	R/W							
Reset	0							

Bit 19 – IOM8IE IO Monitor 8 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 18 – IOM7IE IO Monitor 7 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 17 – IOM6IE IO Monitor 6 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 16 – IOM5IE IO Monitor 5 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 15 – IOM4IE IO Monitor 4 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 14 – IOM3IE IO Monitor 3 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 13 – IOM2IE IO Monitor 2 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 12 – IOM1IE IO Monitor 1 Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 7 – ITCIE Touch Controller Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.30. Interrupt Enable Register 11

Name: IEC11
Offset: 0xEC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			PWRMIE	CICERRIE	CICOSRIE	CICIPDIE	RDCERRIE	RDCCORIE
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						I3C1DEIE	I3C1GIE	I3C1SEIE
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 21 – PWRMIE Power Monitor Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 20 – CICERRIE CIC Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 19 – CICOSRIE CIC Over Sample Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 18 – CICIPDIE CIC Processing Done Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 17 – RDCERRIE RDC Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 16 – RDCCORIE RDC CORDC Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 10 – I3C1DEIE I3C1 DERR Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 9 – I3C1GIE I3C Global Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

Bit 8 – I3C1SEIE Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is not enabled.

10.4.31. Interrupt Priority Register 0

Name: IPC0
Offset: 0xF0

Bit	31	30	29	28	27	26	25	24
		NVMCRCIP[2:0]				NVMIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		NVMECCIP[2:0]				PBERRIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		YRAMECCIP[2:0]				XRAMECCIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		CPUFPUIP[2:0]				IVTCIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – NVMCRCIP[2:0] NVM CRC Operation Completed Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – NVMIP[2:0] NVM Program/Erase Op Completed or Terminated Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – NVMECCIP[2:0] NVM Data ECC SEC and/or Instruction SEC Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – PBERRIP[2:0] PBU Parity Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – YRAMECCIP[2:0] YRAM Data ECC SEC Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – XRAMECCIP[2:0] XRAM Data ECC SEC Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CPUFPUIP[2:0] CPU/FPU Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – IVTCIP[2:0] Interrupt Vector Table Collapse Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.32. Interrupt Priority Register 1

Name: IPC1
Offset: 0xF4

Bit	31	30	29	28	27	26	25	24
		C2FAILIP[2:0]				C1RDYIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		C1MONIP[2:0]				C1WARNIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		C1FAILIP[2:0]				CLKERRIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		CLKFAILIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 30:28 – C2FAILIP[2:0] Clock Monitor 2 Fall Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – C1RDYIP[2:0] Clock Monitor 1 Ready Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – C1MONIP[2:0] Clock Monitor 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – C1WARNIP[2:0] Clock Monitor 1 Warning Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – C1FAILIP[2:0] Clock Monitor 1 Fail Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CLKERRIP[2:0] Clock Error (Combined) Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CLKFAILIP[2:0] Clock Fail Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.33. Interrupt Priority Register 2

Name: IPC2
Offset: 0xF8

Bit	31	30	29	28	27	26	25	24
		C4FAILIP[2:0]				C3RDYIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		C3MONIP[2:0]				C3WARNIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		C3FAILIP[2:0]				C2RDYIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		C2MONIP[2:0]				C2WARNIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – C4FAILIP[2:0] Count Ready Interrupt 4 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – C3RDYIP[2:0] Count Ready Interrupt 3 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – C3MONIP[2:0] Clock Monitor Overflow Interrupt 3 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – C3WARNIP[2:0] Clock Warning Interrupt 3 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – C3FAILIP[2:0] Count Ready Interrupt 3 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – C2RDYIP[2:0] Count Ready Interrupt 2 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – C2MONIP[2:0] Clock Monitor Overflow Interrupt 2 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – C2WARNIP[2:0] Clock Warning Interrupt 2 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.34. Interrupt Priority Register 3

Name: IPC3
Offset: 0xFC

Bit	31	30	29	28	27	26	25	24
		CRYPT2IP[2:0]				CRYPT1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
						WDTIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
						C4RDYIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	7	6	5	4	3	2	1	0
		C4MONIP[2:0]				C4WARNIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – CRYPT2IP[2:0] Crypto Module Interrupt 2 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – CRYPT1IP[2:0] Crypto Module Interrupt 1 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – WDTIP[2:0] Watchdog Timer Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – C4RDYIP[2:0] Count Ready Interrupt 4 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – C4MONIP[2:0] Clock Monitor Overflow Interrupt 4 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – C4WARNIP[2:0] Clock Warning Interrupt 4 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.35. Interrupt Priority Register 4

Name: IPC4
Offset: 0x100

Bit	31	30	29	28	27	26	25	24
		PEVTBIP[2:0]				PEVTAIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		INT4IP[2:0]				INT3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		INT2IP[2:0]				INT1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		INT0IP[2:0]				CRYPT3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – PEVTBIP[2:0] PWM Event B Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – PEVTAIP[2:0] PWM Event A Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – INT4IP[2:0] External Interrupt 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – INT3IP[2:0] External Interrupt 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – INT2IP[2:0] External Interrupt 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – INT1IP[2:0] External Interrupt 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – INT0IP[2:0] External Interrupt 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – CRYPT3IP[2:0] Crypto Module Interrupt 3 Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.36. Interrupt Priority Register 5

Name: IPC5
Offset: 0x104

Bit	31	30	29	28	27	26	25	24
		PWM4IP[2:0]				PWM3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		PWM2IP[2:0]				PWM1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		PEVTFIP[2:0]				PEVTEIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		PEVTDIP[2:0]				PEVTCIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – PWM4IP[2:0] PWM4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – PWM3IP[2:0] PWM3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – PWM2IP[2:0] PWM2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – PWM1IP[2:0] PWM1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – PEVTFIP[2:0] PWM Event F Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – PEVTEIP[2:0] PWM Event E Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – PEVTDIP[2:0] PWM Event D Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – PEVTCIP[2:0] PWM Event C Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.37. Interrupt Priority Register 6

Name: IPC6
Offset: 0x108

Bit	31	30	29	28	27	26	25	24
		CCT3IP[2:0]				CCP2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		0	0	1

Bit	23	22	21	20	19	18	17	16
		CCT2IP[2:0]				CCP1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		0	0	1

Bit	15	14	13	12	11	10	9	8
		CCT1IP[2:0]				T3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		0	0	1

Bit	7	6	5	4	3	2	1	0
		T2IP[2:0]				T1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – CCT3IP[2:0] Capture/Compare/Timer 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – CCP2IP[2:0] ECCP2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – CCT2IP[2:0] Capture/Compare/Timer 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – CCP1IP[2:0] ECCP1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CCT1IP[2:0] Capture/Compare/Timer 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – T3IP[2:0] Timer 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – T2IP[2:0] Timer 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – T1IP[2:0] Timer 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.38. Interrupt Priority Register 7

Name: IPC7
Offset: 0x10C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		C1IP[2:0]				C1TXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		C1RXIP[2:0]				CCP4IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		CCT4IP[2:0]				CCP3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 22:20 – C1IP[2:0] CAN1 Combined Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – C1TXIP[2:0] CAN1 RX Transmit Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – C1RXIP[2:0] CAN1 RX Data Ready Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CCP4IP[2:0] Input Capture/Output Compare 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CCT4IP[2:0] Capture/Compare/Timer 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – CCP3IP[2:0] Input Capture/Output Compare 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.39. Interrupt Priority Register 8

Name: IPC8
Offset: 0x110

Bit	31	30	29	28	27	26	25	24
		SPI3RXIP[2:0]				SPI2EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
		SPI2TXIP[2:0]				SPI2RXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		SPI1EIP[2:0]				SPI1TXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		SPI1RXIP[2:0]						
Access		RW	RW	RW				
Reset		0	0	0				

Bits 30:28 – SPI3RXIP[2:0] SPI3 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – SPI2EIP[2:0] SPI2 Error Interrupt Priority bit

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – SPI2TXIP[2:0] SPI2 Transfer Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – SPI2RXIP[2:0] SPI2 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – SPI1EIP[2:0] SPI1 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – SPI1TXIP[2:0] SPI1 Transmit Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – SPI1RXIP[2:0] SPI1 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.40. Interrupt Priority Register 9

Name: IPC9
Offset: 0x114

Bit	31	30	29	28	27	26	25	24
		DMA2IP[2:0]				DMA1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		DMA0IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		SPI3EIP[2:0]				SPI3TXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – DMA2IP[2:0] Direct Memory Access 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – DMA1IP[2:0] Direct Memory Access 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – DMA0IP[2:0] Direct Memory Access 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – SPI3EIP[2:0] SPI3 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – SPI3TXIP[2:0] SPI3 Transmit Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.41. Interrupt Priority Register 10

Name: IPC10
Offset: 0x118

Bit	31	30	29	28	27	26	25	24
		I2C1RXIP[2:0]				I2C1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	1		1	0	0

Bit	23	22	21	20	19	18	17	16
		I2C1EIP[2:0]				CMP4IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		0	0	1

Bit	15	14	13	12	11	10	9	8
		CMP3IP[2:0]				CMP2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	1		0	0	1

Bit	7	6	5	4	3	2	1	0
		CMP1IP[2:0]				DMA3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	1		0	0	1

Bits 30:28 – I2C1RXIP[2:0] I2C1 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – I2C1IP[2:0] I2C1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – I2C1EIP[2:0] I2C1 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – CMP4IP[2:0] Comparator 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CMP3IP[2:0] Comparator 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CMP2IP[2:0] Comparator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CMP1IP[2:0] Comparator 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – DMA3IP[2:0] Direct Memory Access 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.42. Interrupt Priority Register 11

Name: IPC11
Offset: 0x11C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						I2C2TXIP[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	1

Bit	15	14	13	12	11	10	9	8
		I2C2RXIP[2:0]				I2C2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	1		1	0	0

Bit	7	6	5	4	3	2	1	0
		I2C2EIP[2:0]				I2C1TXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		0	0	1

Bits 18:16 – I2C2TXIP[2:0] I2C2 Transmit Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – I2C2RXIP[2:0] I2C2 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – I2C2IP[2:0] I2C2 Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – I2C2EIP[2:0] I2C2 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – I2C1TXIP[2:0] I2C1 Transmit Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.43. Interrupt Priority Register 12

Name: IPC12
Offset: 0x120

Bit	31	30	29	28	27	26	25	24
		U2TXIP[2:0]				U2RXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		U1EVTIP[2:0]				U1EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		U1TXIP[2:0]				U1RXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 30:28 – U2TXIP[2:0] UART2 Transmit Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – U2RXIP[2:0] UART 2 Receive Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – U1EVTIP[2:0] UART1 Event Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – U1EIP[2:0] UART1 Framing Error Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – U1TXIP[2:0] UART 1 Transmit Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – U1RXIP[2:0] UART 1 Receive Interrupt Priority

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.44. Interrupt Priority Register 13

Name: IPC13
Offset: 0x124

Bit	31	30	29	28	27	26	25	24
		U4TXIP[2:0]				U4RXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		U3EVTIP[2:0]				U3EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		U3TXIP[2:0]				U3RXIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		U2EVTIP[2:0]				U2EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – U4TXIP[2:0] UART4 Transmit Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – U4RXIP[2:0] UART4 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – U3EVTIP[2:0] UART3 Event Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – U3EIP[2:0] UART3 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – U3TXIP[2:0] UART3 Transmit Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – U3RXIP[2:0] UART3 Receive Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – U2EVTIP[2:0] UART2 Event Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – U2EIP[2:0] UART2 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.45. Interrupt Priority Register 14

Name: IPC14
Offset: 0x128

Bit	31	30	29	28	27	26	25	24
		DMA5IP[2:0]				DMA4IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		SENT2EIP[2:0]				SENT2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		SENT1EIP[2:0]				SENT1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		U4EVTIP[2:0]				U4EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – DMA5IP[2:0] DMA 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – DMA4IP[2:0] DMA 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – SENT2EIP[2:0] SENT2 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – SENT2IP[2:0] SENT2 TX/RX Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – SENT1EIP[2:0] SENT1 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – SENT1IP[2:0] SENT1 TX/RX Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – U4EVTIP[2:0] UART4 Event Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – U4EIP[2:0] UART4 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.46. Interrupt Priority Register 15

Name: IPC15
Offset: 0x12C

Bit	31	30	29	28	27	26	25	24
		CCP5IP[2:0]				CCT5IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		CNDIP[2:0]				CNCIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		CNBIP[2:0]				CNAIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		DMA7IP[2:0]				DMA6IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – CCP5IP[2:0] CCP 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – CCT5IP[2:0] CCT 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – CNDIP[2:0] Change Notice D Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – CNCIP[2:0] Change Notice C Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CNBIP[2:0] Change Notice B Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CNAIP[2:0] Change Notice A Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – DMA7IP[2:0] DMA 7 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – DMA6IP[2:0] DMA 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.47. Interrupt Priority Register 17

Name: IPC17
Offset: 0x134

Bit	31	30	29	28	27	26	25	24
		ICDIP[2:0]				CRCIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		BISS1IP[2:0]				BISS1EIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						QEI1IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bits 30:28 – ICDIP[2:0] In-Circuit Debugger Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – CRCIP[2:0] CRC Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – BISS1IP[2:0] BiSS 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – BISS1EIP[2:0] BiSS 1 Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – QEI1IP[2:0] QEI 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.48. Interrupt Priority Register 18

Name: IPC18
Offset: 0x138

Bit	31	30	29	28	27	26	25	24
						PTG3IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	23	22	21	20	19	18	17	16
		PTG2IP[2:0]				PTG1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		PTG0IP[2:0]				PTGWDTIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		PTGSTEPIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 26:24 – PTG3IP[2:0] PTG 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – PTG2IP[2:0] PTG 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – PTG1IP[2:0] PTG 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – PTG0IP[2:0] PTG 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – PTGWDTIP[2:0] PTG Watchdog Timer Time-out Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – PTGSTEPIP[2:0] PTG Step Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.49. Interrupt Priority Register 19

Name: IPC19
Offset: 0x13C

Bit	31	30	29	28	27	26	25	24
		AD1CH1IP[2:0]				AD1CMP0IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD1CH0IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 30:28 – AD1CH1IP[2:0] ADC 1 Channel 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD1CMP0IP[2:0] ADC 1 Digital Comparator 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD1CH0IP[2:0] ADC 1 Channel 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.50. Interrupt Priority Register 20

Name: IPC20
Offset: 0x140

Bit	31	30	29	28	27	26	25	24
		AD1CH5IP[2:0]				AD1CMP4IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD1CH4IP[2:0]				AD1CMP3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD1CH3IP[2:0]				AD1CMP2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD1CH2IP[2:0]				AD1CMP1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD1CH5IP[2:0] ADC 1 Channel 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD1CMP4IP[2:0] ADC 1 Digital Comparator 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD1CH4IP[2:0] ADC 1 Channel 4 Interrupt Priority bits

Value	Description
1	Interrupt has occurred
0	Interrupt has not occurred

Bits 18:16 – AD1CMP3IP[2:0] ADC 1 Digital Comparator 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD1CH3IP[2:0] ADC 1 Channel 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD1CMP2IP[2:0] ADC 1 Digital Comparator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD1CH2IP[2:0] ADC 1 Channel 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD1CMP1IP[2:0] ADC 1 Digital Comparator 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2

Value	Description
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.51. Interrupt Priority Register 21

Name: IPC21
Offset: 0x144

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						AD1CMP6IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	7	6	5	4	3	2	1	0
		AD1CH6IP[2:0]				AD1CMP5IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 10:8 – AD1CMP6IP[2:0] ADC 1 Digital Comparator 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD1CH6IP[2:0] ADC 1 Channel 6 Interrupt Flag bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD1CMP5IP[2:0] ADC 1 Digital Comparator 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.52. Interrupt Priority Register 22

Name: IPC22
Offset: 0x148

Bit	31	30	29	28	27	26	25	24
		AD2CH2IP[2:0]				AD2CMP1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD2CH1IP[2:0]				AD2CMP0IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD2CH0IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 30:28 – AD2CH2IP[2:0] ADC 2 Digital Comparator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD2CMP1IP[2:0] ADC 2 Digital Comparator 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD2CH1IP[2:0] ADC 2 Channel 1 Interrupt Flag bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD2CMP0IP[2:0] ADC 2 Digital Comparator 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD2CH0IP[2:0] ADC 2 Channel 0 Interrupt Flag bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.53. Interrupt Priority Register 23

Name: IPC23
Offset: 0x14C

Bit	31	30	29	28	27	26	25	24
		AD2CH6IP[2:0]				AD2CMP5IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD2CH5IP[2:0]				AD2CMP4IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD2CH4IP[2:0]				AD2CMP3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD2CH3IP[2:0]				AD2CMP2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD2CH6IP[2:0] ADC 2 Channel 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD2CMP5IP[2:0] ADC 2 Digital Comparator 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD2CH5IP[2:0] ADC 2 Channel 5 Interrupt Flag bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD2CMP4IP[2:0] ADC 2 Digital Comparator 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD2CH4IP[2:0] ADC 2 Channel 4 Interrupt Flag bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD2CMP3IP[2:0] ADC 2 Digital Comparator 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD2CH3IP[2:0] ADC 2 Channel 3 Interrupt Flag bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD2CMP2IP[2:0] ADC 2 Digital Comparator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.54. Interrupt Priority Register 24

Name: IPC24
Offset: 0x150

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						AD2CMP7IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	7	6	5	4	3	2	1	0
		AD2CH7IP[2:0]				AD2CMP6IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 10:8 – AD2CMP7IP[2:0] ADC 2 Digital Comparator 7 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD2CH7IP[2:0] ADC 2 Channel 7 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD2CMP6IP[2:0] ADC 2 Digital Comparator 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.55. Interrupt Priority Register 25

Name: IPC25
Offset: 0x154

Bit	31	30	29	28	27	26	25	24
		AD3CH3IP[2:0]				AD3CMP2IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD3CH2IP[2:0]				AD3CMP1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD3CH1IP[2:0]				AD3CMP0IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD3CH0IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 30:28 – AD3CH3IP[2:0] ADC 3 Channel 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD3CMP2IP[2:0] ADC 3 Digital Comparator 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD3CH2IP[2:0] ADC 3 Channel 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD3CMP1IP[2:0] ADC 3 Digital Comparator 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD3CH1IP[2:0] ADC 3 Channel 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD3CMP0IP[2:0] ADC 3 Digital Comparator 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD3CH0IP[2:0] ADC 3 Channel 0 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.56. Interrupt Priority Register 26

Name: IPC26
Offset: 0x158

Bit	31	30	29	28	27	26	25	24
		AD3CH7IP[2:0]				AD3CMP6IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		AD3CH6IP[2:0]				AD3CMP5IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD3CH5IP[2:0]				AD3CMP4IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		AD3CH4IP[2:0]				AD3CMP3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 30:28 – AD3CH7IP[2:0] ADC 3 Channel 7 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – AD3CMP6IP[2:0] ADC 3 Digital Comparator 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD3CH6IP[2:0] ADC 3 Channel 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD3CMP5IP[2:0] ADC 3 Digital Comparator 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD3CH5IP[2:0] ADC 3 Channel 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – AD3CMP4IP[2:0] ADC 3 Digital Comparator 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD3CH4IP[2:0] ADC 3 Channel 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD3CMP3IP[2:0] ADC 3 Digital Comparator 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.57. Interrupt Priority Register 27

Name: IPC27
Offset: 0x15C

Bit	31	30	29	28	27	26	25	24
						AD3CMP10IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	23	22	21	20	19	18	17	16
		AD3CH11IP[2:0]				AD3CMP9IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
		AD3CH9IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	7	6	5	4	3	2	1	0
		AD3CH8IP[2:0]				AD3CMP8IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 26:24 – AD3CMP10IP[2:0] ADC 3 Digital Comparator 10 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD3CH11IP[2:0] ADC 3 Channel 11 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – AD3CH10IP[2:0] ADC 3 Channel 10 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – AD3CMP9IP[2:0] ADC 3 Digital Comparator 9 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – AD3CH9IP[2:0] ADC 3 Channel 9 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – AD3CH8IP[2:0] ADC 3 Channel 8 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD3CMP8IP[2:0] ADC 3 Digital Comparator 8 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – AD3CMP7IP[2:0] ADC 3 Digital Comparator 7 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.58. Interrupt Priority Register 28

Name: IPC28
Offset: 0x160

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						AD3CMP11IP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bits 2:0 – AD3CMP11IP[2:0] ADC 3 Digital Comparator 11 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.59. Interrupt Priority Register 34

Name: IPC34
Offset: 0x178

Bit	31	30	29	28	27	26	25	24
		CLC2PIP[2:0]				CLC1NIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		CLC1PIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		CMP5IP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bits 30:28 – CLC2PIP[2:0] CLC2 Positive Edge Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – CLC1NIP[2:0] CLC1 Negative Edge Interrupt Priory bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – CLC1PIP[2:0] CLC1 Positive Edge Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CMP5IP[2:0] Comparator 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.60. Interrupt Priority Register 35

Name: IPC35
Offset: 0x17C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						CLC4NIP[2:0]		
Access						R/W	R/W	R/W
Reset						1	0	0

Bit	15	14	13	12	11	10	9	8
		CLC4PIP[2:0]				CLC3NIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		CLC3PIP[2:0]				CLC2NIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 18:16 – CLC4NIP[2:0] CLC1 Negative Edge Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CLC4PIP[2:0] CLC1 Positive Edge Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CLC3NIP[2:0] CLC2 Negative Edge Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – CLC3PIP[2:0] CLC2 Positive Edge Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – CLC2NIP[2:0] CLC1 Negative Edge Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.61. Interrupt Priority Register 40

Name: IPC40
Offset: 0x190

Bit	31	30	29	28	27	26	25	24
		ITCIP[2:0]						
Access		R/W	R/W	R/W				
Reset		1	0	0				

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 30:28 – ITCIP[2:0] Touch Controller Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.62. Interrupt Priority Register 41

Name: IPC41
Offset: 0x194

Bit	31	30	29	28	27	26	25	24
		IOM4IP[2:0]				IOM3IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	23	22	21	20	19	18	17	16
		IOM2IP[2:0]				IOM1IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 30:28 – IOM4IP[2:0] IO Monitor 4 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 26:24 – IOM3IP[2:0] IO Monitor 3 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 22:20 – IOM2IP[2:0] IO Monitor 2 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – IOM1IP[2:0] IO Monitor 1 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.63. Interrupt Priority Register 42

Name: IPC42
Offset: 0x198

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		IOM8IP[2:0]				IOM7IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bit	7	6	5	4	3	2	1	0
		IOM6IP[2:0]				IOM5IP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		1	0	0		1	0	0

Bits 14:12 – IOM8IP[2:0] IO Monitor 8 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – IOM7IP[2:0] IO Monitor 7 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – IOM6IP[2:0] IO Monitor 6 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – IOM5IP[2:0] IO Monitor 5 Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.64. Interrupt Priority Register 45

Name: IPC45
Offset: 0x1A4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						I3C1DEIP[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
		I3CISEIP[2:0]				I3CIGIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bits 10:8 – I3C1DEIP[2:0] Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – I3CISEIP[2:0] Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – I3CIGIP[2:0] Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.4.65. Interrupt Priority Register 46

Name: IPC46
Offset: 0x1A8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		PWRMIP[2:0]				CICERRIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	15	14	13	12	11	10	9	8
		CICOSIP[2:0]				CICIPDIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
		RDCERIP[2:0]				RDCCORIP[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		1	0	0

Bits 22:20 – PWRMIP[2:0] Power Monitor Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 18:16 – CICERRIP[2:0] CIC Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 14:12 – CICOSIP[2:0] CIC Over Sample Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5

Value	Description
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 10:8 – CICIPDIP[2:0] CIC Processing Done Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 6:4 – RDCERIP[2:0] RDC Error Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

Bits 2:0 – RDCCORIP[2:0] RDC CORDIC Interrupt Priority bits

Value	Description
7	Interrupt Priority Level 7 (highest)
6	Interrupt Priority Level 6
5	Interrupt Priority Level 5
4	Interrupt Priority Level 4 (default)
3	Interrupt Priority Level 3
2	Interrupt Priority Level 2
1	Interrupt Priority Level 1
0	Interrupt Priority Level 0 (lowest)

10.5. Operation

10.5.1. Reset Vector

The Reset vector is a true vector without the legacy jump instruction (`GOTO`). The processor clears its registers in response to a Reset, which forces the PC to 0x800000. The processor then begins program execution at the fixed Reset vector at 0x800000. The Reset vector at the Reset address redirects program execution to the appropriate start-up routine.

When a Reset is asserted, the user vector read commences. When available, the Reset vector contents are presented to the CPU. The data are then immediately transferred to the Program Space address bus to create the address of the first instruction to be executed.

10.5.2. Trap Vector Details

Note: Any unimplemented or unused vector locations in the IVT should be programmed with the address of a default interrupt handler routine that contains a `RESET` instruction.

Table 10-2. Trap Vector Details

Vector Source	MPLAB® XC-DSC Trap ISR Name	IRQ #	VECNUM	IVT Offset	Interrupt Bit Location		
					Flag	Enable	Priority
Bus error - CPU X data bus error	_BusErrorTrap	N/A	2	0x8	INTCON3[0]	—	14
Bus error - CPU Y data bus error	_BusErrorTrap	N/A	2	0x8	INTCON3[1]	—	14
Bus error - DMA bus error	_BusErrorTrap	N/A	2	0x8	INTCON3[2]	—	14
Bus error - CPU instruction data bus error	_BusErrorTrap	N/A	2	0x8	INTCON3[3]	—	14
Illegal instruction	_IllegalInstructionTrap	N/A	3	0xC	INTCON1[2]	—	13
Address error	_AddressErrorTrap	N/A	4	0x10	INTCON1[3]	—	12
Stack error	_StackErrorTrap	N/A	5	0x14	INTCON1[4]	—	11
Math error - Divide by Zero	_MathErrorTrap	N/A	6	0x18	INTCON4[0]	—	10
Math error - Accumulator Shift error	_MathErrorTrap	N/A	6	0x18	INTCON4[1]	—	10
Math error - Accumulator B Catastrophic Overflow	_MathErrorTrap	N/A	6	0x18	INTCON4[2]	INTCON4[19]	10
Math error - Accumulator A Catastrophic Overflow	_MathErrorTrap	N/A	6	0x18	INTCON4[3]	INTCON4[19]	10
Math error - Accumulator B Overflow	_MathErrorTrap	N/A	6	0x18	INTCON4[4]	INTCON4[20]	10
Math error - Accumulator A Overflow	_MathErrorTrap	N/A	6	0x18	INTCON4[5]	INTCON4[21]	10
General error - DMT event trap	_GeneralTrap	N/A	7	0x1C	INTCON5[0]	—	9
General error - WDT Run event trap	_GeneralTrap	N/A	7	0x1C	INTCON5[1]	—	9
General error - XRAM PWB DED error	_GeneralTrap	N/A	7	0x1C	INTCON5[2]	—	9
General error - YRAM PWB DED error	_GeneralTrap	N/A	7	0x1C	INTCON5[3]	—	9
General error - soft trap	_GeneralTrap	N/A	7	0x1C	INTCON5[31]	—	9

10.6. Interrupt Control and Status Registers

The dsPIC33AK256MPS306 family devices implement the following registers for the interrupt controller:

- INTCON1
- INTCON2
- INTCON3

- INTCON4
- INTCON5
- INTTREG

10.6.1. INTCON1 through INTCON5

Global interrupt control functions are controlled from INTCON1 through INTCON5. INTCON1 contains the Interrupt Nesting Disable bit (NSTDIS), Global Interrupt Enable bit (GIE) and the status flags for the processor trap sources.

The INTCON2 register controls the external interrupt polarity select bits. INTCON3 contains the status flags for the different bus error traps, and INTCON4 contains control and status flags for various math error traps.

INTCON5 contains the Soft Trap bit and other generic trap bits associated with WDT, DMT events, XRAM and YRAM PWB DED Error status.

10.6.2. IFSx

The IFSx registers maintain all of the interrupt request flags. Each source of interrupt has a status bit, which is set by the respective peripherals or external signal and is cleared via software.

10.6.3. IECx

The IECx registers maintain all of the interrupt enable bits. These control bits are used to individually enable interrupts from the peripherals or external signals.

10.6.4. IPCx

The IPCx registers are used to set the Interrupt Priority Level (IPL) for each source of interrupt. Each user interrupt source can be assigned to one of seven priority levels.

10.6.5. INTTREG

When an interrupt is presented to the CPU, associated details of the interrupt are latched onto the INTTREG register. INTTREG[ILR] holds the priority level of the presented interrupt. If INTTREG[VHOLD] is equal to 1, the value of INTTREG[VECNUM] represents the vector number of the presented interrupt. Otherwise, it represents the vector number of the last acknowledged interrupt.

INTTREG holds the value of ILR and VECNUM until the next interrupt is presented to the CPU.

The interrupt sources are assigned to the IFSx, IECx and IPCx registers in the same sequence as they are listed in Table 10-1. For example, the FPU exception is shown as having a vector number of 10 and an IRQ number of 1. Thus, the FPUIF bit is placed at IFS0[1], the FPUIE bit in IEC0[1] and FPUIP[2:0] bits in IPC0 (IPC0[6:4]).

10.6.6. Vector Fail Address (VFA)

A bus error may occur when reading a vector location, which can result in an incorrect vector value being returned. To prevent the CPU from executing a Program Flow Control (PFC) instruction to an invalid address, the CPU detects this situation, sets the Vector Fail flag (SR.VF) in the Status Register and substitutes a register-based address as the vector address value.

This substitute address is stored in the Vector Fail Address (VFA) Special Function Register (SFR) within the CPU. Users should initialize the VFA to point to the Bus Error handler or a dedicated Bus Error handler. By default, the CPU initializes the VFA to the reset (first instruction) address during the reset sequence, ensuring a valid default value if the user does not set it.

Vector substitution only occurs in the case of a vector fetch bus error, making this a special type of bus error. The CPU does not generate a Bus Error trap request for the Interrupt Controller. However, it does raise its Interrupt Priority Level (IPL) to that of the Bus Error trap (IPL 14), rather than updating the IPL based on the Interrupt Controller’s input. If the VFA contains the Bus Error trap vector address, exception processing continues and enters the Bus Error handler, abandoning the original exception (which remains pending).

The Interrupt Controller records the failed exception's Interrupt Level Register (ILR) and Vector Number Register (VNR) in the INTTREG register as usual, since the CPU does not issue a Bus Error trap request. The Bus Error handler can use this information to determine the appropriate response if the SR.VF bit is set. If the handler determines the vector failure was temporary, it can safely return, allowing the pending exception to be retried. Otherwise, the handler should take further action as defined by the user. The stacked SR.VF bit will be clear, so executing RETFIE from the handler will automatically clear the bit.

If vector fetch bus errors do not need to be handled differently, ignore the SR.VF bit and treat the event as a standard bus error.

Alternatively, if the VFA register is left at its default reset address, a bus error during vector fetch will raise the CPU IPL to IPL 14 and vector to the application reset address. This does not reset the device, but the user can check the SR.VF bit during initialization. If it is set, the user may reset the device by executing a `RESET` instruction.

10.7. Priority

10.7.1. CPU Priority

An interrupt or a trap source must have a priority level greater than the current CPU priority to initiate an exception process. The CPU priority level is defined by a 4-bit value, SR.IPL [3:0]. The IPL [3] bit can be read at any time and may be cleared by software to allow trap handlers to jump to another process without having to execute a RETFIE.

IPL3: MSb of CPU Priority Level Nibble bit

1 = CPU Priority ≥ 8 (trap exception underway)

0 = CPU Priority < 8 (no trap exception underway)

IPL [2:0]: CPU Interrupt Priority Level status bits

111 = All interrupts disabled

110 = Level 7 interrupts enabled

101 = Level 6 and 7 interrupts enabled

100 = Level 5 through 7 interrupts enabled

011 = Level 4 through 7 interrupts enabled

010 = Level 3 through 7 interrupts enabled

001 = Level 2 through 7 interrupts enabled

000 = Level 1 through 7 interrupts enabled

The SR.IPL[2:0] status bits are readable and writable, so the user application can modify these bits to disable all sources of interrupts below a given priority level. For example, if IPL = 011, the CPU would not be interrupted by any source with a programmed priority level of one, two or three. All user interrupt sources can be disabled by setting SR.IPL[2:0] = 111.

Trap events have a higher priority than any user interrupt source. The IPL3 bit will be set by hardware when a trap occurs. This bit can be cleared but not set by the user application. In some applications, the IPL3 bit will need to be cleared when a trap has occurred, and it will need to be branched to an instruction other than the instruction immediately after the one that originally caused the trap to occur.

The CPU interrupt priority is automatically modified during exception processing. However, provided interrupt nesting is enabled (INTCON1.NSTDIS= 0), IPL [2:0] are read/write bits and may also be manipulated by the user to dynamically modify the CPU interrupt priority. If interrupt nesting is disabled (INTCON1.NSTDIS = 1), IPL [2:0] shall by default be set to 0x7 and become read-only bits to prevent the user from inadvertently dropping the CPU interrupt priority (and causing any pending interrupts to nest).

10.7.2. Interrupt Priority

Each peripheral interrupt source can be assigned to one of the seven priority levels. The user assignable interrupt priority control bits for each individual interrupt are located in the Least Significant three bits of each nibble within the IPCx registers. Bit 3 of each nibble is not used and is read as a ‘0’. These bits define the priority level assigned to a particular interrupt. The usable priority levels are one (lowest priority) through seven (highest priority). If all the IPCx bits associated with an interrupt source are cleared, the interrupt source is effectively disabled.

More than one interrupt request source can be assigned to a specific priority level. To resolve priority conflicts within a given user-assigned level, each source of an interrupt has a natural priority order based on its location in the IVT. The lower IRQ-numbered interrupt vectors have higher natural priority, while the higher-numbered vectors have lower natural priority (refer to Table 10-1 for IRQ numbers of interrupt). The overall priority level for any pending source of an interrupt is first determined by the user application-assigned priority of that source in the IPCx register, then by the natural order priority within the IVT/AIVT.

Natural order priority is used only to resolve conflicts between simultaneous pending interrupts with the same user application-assigned priority level. Once the priority conflict is resolved and the exception process begins, the CPU can be interrupted only by a source with a higher user application-assigned priority. Interrupts with the same user application-assigned priority, but a higher natural order priority that become pending during the exception process, remain pending until the current exception process completes.

Each interrupt source can be assigned to one of seven priority levels. This enables the user application to assign a low natural order priority and a very high overall priority level to an interrupt. For example, the UART1 RX Interrupt can be assigned to priority level seven, and the External Interrupt 0 (INT0) can be assigned to priority level one, thereby giving it a very low effective priority.

Note: If an interrupt is programmed with a priority of zero, the interrupt is disabled.

10.8. Interrupt Sequence

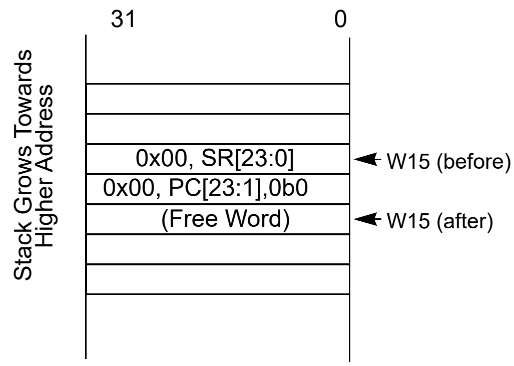
All interrupt event flags are sampled in the system clock by the IFSx registers. A pending Interrupt Request (IRQ) is indicated by the flag bit being equal to a ‘1’ in an IFSx register. The IRQ will cause the interrupt to occur if the corresponding bit in the Interrupt Enable (IECx) register is set. For the next clock cycle, the priorities of all pending interrupt requests are evaluated.

If there is a pending IRQ with a priority level greater than the current processor priority level in the Processor Status register, an interrupt will be presented to the processor. No instruction is aborted when the CPU responds to the IRQ. When the IRQ is sampled, the instruction in progress is completed before the Interrupt Service Routine (ISR) is executed.

The CPU interrupt priority level and its associated vector number are latched into ILR and VECNUM bit fields in the INTTREG register to keep them stable through the interrupt process. The processor reacts to the interrupt request by asserting the IACK (interrupt Acknowledge) signal to prevent the ILR and VECNUM bits from changing during the interrupt process.

The processor then stacks the current Program Counter and the low byte of the processor status register. The low byte of the status register contains the processor priority level at the time prior to the beginning of the interrupt cycle.

Figure 10-3. Exception Stack Frame



The processor then takes the priority level for this interrupt and loads it into the processor status register. This action will disable all lower priority interrupts until the completion of the ISR. Additionally, the CTX bits of the SR register are automatically updated to the value of the IPL, thereby triggering a hardware context switch prior to entering the ISR.

After servicing the interrupt, the RETFIE (Return from Interrupt) instruction will unstack the Program Counter and status registers to return the processor to its state prior to the interrupt sequence.

The processor IPL register is a 4-bit register. The IPL bits are available in the Processor Status register.

The MSB of the SR.IPL register is set if a trap is being processed. There are seven levels of user IPLs and eight priority levels for traps (two levels are reserved plus six traps).

10.8.1. Peripheral Interrupt Vector Address

The vector address is determined by the value of the interrupt request number and the value of the base register bits. The base register - IVTBASE value is used to provide an offset for the base address of the interrupt vector, so that interrupt vector addresses can be mapped anywhere in user Flash or anywhere in the user RAM.

To calculate the address in the vector table where a particular peripheral interrupt vector resides, use the following equation:

address = (Base (Hex) + (IRQ number x4 (decimal)) + 24 (hex)

An example for Interrupt #21 with Base address = 0x80A000:

address = (0x80A000 (hex)+ (21x4 (dec)) +24 (hex) = 0x80A078 (hex)

An example for Interrupt #21 after Reset:

address = (‘800000 (hex)+ (21x4 (dec)) +24 (hex) = 800078 (hex)

10.8.2. Interrupt Nesting

Interrupts are nestable by default. Any ISR in progress can be interrupted by another source of interrupt with a higher user application-assigned priority level. Interrupt nesting can be disabled by setting the Interrupt Nesting Disable bit (NSTDIS) in the INTCON1 register. When the NSTDIS control bit is set, all interrupts in progress force the CPU priority to Level 7 by setting IPL = 0b111. This action effectively masks all other sources of interrupts until a RETFIE instruction is executed. When interrupt nesting is disabled, the user application-assigned IPLs have no effect except to resolve conflicts between simultaneous pending interrupts. The IPL bits (SR) become read-only when interrupt nesting is disabled. This prevents the user application from setting IPL to a lower value, which would effectively re-enable interrupt nesting.

10.9. Non-Maskable Traps

Traps are non-maskable, nestable interrupts that adhere to a fixed priority structure. Traps provide a means to correct an erroneous operation during debugging and the operation of the application. If the user application does not intend to correct a trap error condition, these vectors must be loaded with the address of a software routine to reset the device. Otherwise, the user application must program the trap vector with the address of a service routine that corrects the trap condition.

The following sources of non-maskable traps are implemented in dsPIC33A devices:

- Bus error and ECC DED trap
- Illegal opcode error trap
- CPU address error trap
- CPU stack error trap
- CPU math error trap
- Generic trap

For many of the trap conditions, the instruction that caused the trap is allowed to complete before exception processing begins. Therefore, the user application may have to correct the action of the instruction that caused the trap. Each trap source has a fixed priority as defined by its position in the IVT/IVTC. A bus error trap has the highest priority, while a generic trap has the lowest priority. Refer Table 10-2 for trap vector and priority details.

10.9.1. Soft Traps

Soft traps can be treated like non-maskable sources of an interrupt that adhere to the priority assigned by their position in the IVT. Soft traps are processed like interrupts and require two cycles to be sampled and acknowledged prior to exception processing. Therefore, additional instructions may be executed before a soft trap is acknowledged.

10.9.1.1. Bus Error Traps

This trap is generated if the requested operation cannot be completed due to errors on the bus. Bus matrix initiator modules will generate this error if the BMX or target module reports an error during the transaction (e.g., address errors, ECC DED errors). A bus error occurs when any of the bits within the INTCON3 register are set. Each bit within the INTCON3 register is assigned to a specific bus error condition. Refer to Bus Error for more details on bus error traps.

10.9.1.2. Illegal Opcode Error

An illegal opcode error trap is asserted when an attempt is made to execute an illegal opcode; it is conditional upon the commitment of a speculatively executed instruction (so it cannot be asserted until the R-stage). An illegal opcode trap is asserted for:

- an attempt to execute any opcode slot within the opcode map that is not allocated an instruction. In addition to unused opcode slots, this also includes any unused sub-opcode slots.
- an attempt to PFC to the second word of a two-word instruction, using the instruction word identifier bit.
- an attempt to execute any unimplemented coprocessor instruction, including any opcode that includes coprocessor select opcode bits (‘zz’ bits) for an unimplemented coprocessor.
- an attempt to PFC to an odd 16-bit instruction address within what is defined as a 32-bit instruction word (Most Significant opcode bit set).

10.9.1.3. CPU Address Error Trap

This trap will be taken when any of the following circumstances occur:

1. If a W-reg (W0 through W14) is used by an AGU as a source or destination and a misaligned data word access (long word access at an odd address or word access at an odd byte address) is attempted.

Note: For misaligned writes, the read or write is not inhibited. Address alignment is forced such that read or write is always aligned.

2. If a W-reg (W0 through W15) is used by an AGU, and the address or offset pointed to by the W-reg is beyond 24 bits in size. In other words, an address error trap will be generated if the MSB of the address pointed to by the W-reg and used by the AGU is not set to 8'h00, or if the MSB of the address offset pointed to by the W-reg, which is used for EA calculation, is not 8'h00 or 8'hFF.

   or

   If a W-reg is used by an AGU as an offset where permitted and contains a value where the Most Significant 8-bits are not 8’h00 or 8’hFF (offset can be a signed negative value).

   Note: The CPU will detect X address access to SFRs and treat it as a special case. However, Y space reads of SFR or PS addresses must be detected by the BMX and result in an address error trap.

3. If the MSB of a computed PFC address (`BRAW`, `RCALLW`, `CALLW`, `GOTOW`) is not 8’h00.

10.9.1.4. Stack Error Trap

The stack is initialized to the start of the data RAM address (0x4000) during Reset. A stack error trap is generated if the Stack Pointer effective address (EA) is less than the initial stack value (0x4000). Stack underflow detection is provided to protect the SFR space from being modified by the Stack Pointer. A Stack Limit register (SPLIM) associated with the Stack Pointer is uninitialized at Reset. The stack overflow check is not enabled until a word is written to the SPLIM register.

The Stack Error Status bit (STKERR) in the INTCON1 register is set whenever a stack error occurs. To avoid re-entry into the Trap Service Routine (TSR), the STKERR status flag must be cleared in software.

The stack error trap will be triggered only when the following circumstances occur:

1. A Stack Pointer (W15) based access is attempted with an EA that is less than the start address of data RAM.
2. Stack overflow protection is enabled, and a Stack Pointer-based access is attempted with an EA that is greater than the (user-programmable) limit value written into the SPLIM register.
3. A pre/post increment/decrement operation is performed on W15 (including Stack Pointer modification during exception processing) that results in EA[1:0] != 0b00 (i.e., not long word aligned). This will detect byte and word pre/post increment/decrement operations that are otherwise considered aligned but would result in a misaligned Stack Pointer.

10.9.1.5. Math Error Traps

The math error trap will execute under the circumstances listed below. The associated math error status and enable bits are located in the INTCON4 register.

1. Divide By Zero: Should an attempt be made to divide by zero, the PC stack will point to the iterated instruction being executed at the time the error is detected. The divide iterations, up to the point that exception processing occurs, will execute as usual (though with meaningless results).
2. DSP Overflow: If the following conditions are all true, an Overflow of AccA (OVA) math error trap will be taken.
   a. OVATE bit is set (INTCON4[21]).
   b. Accumulator A is operating with 1.63 saturation disabled or in 9.63 mode.
   c. An arithmetic operation caused an overflow from bit 63 of accumulator A.
3. DSP Overflow: If the following conditions are all true, an Overflow of AccB (OVB) math error trap will be taken.
   a. OVBTE bit is set (INTCON4[20]).

   b. Accumulator B is operating with 1.63 saturation disabled or in 9.63 mode.
   c. An arithmetic operation caused an overflow from bit 63 of accumulator B.
4. DSP Overflow: If the following conditions are all true, an accumulator Catastrophic Overflow (COV) math error trap will be taken.
   a. COVTE bit is set (INTCON4[19]).
   b. Either Accumulator A or B is operating with all saturation disabled.
   c. An arithmetic operation caused an overflow from bit 71 (Catastrophic Overflow) of the accumulator with all saturation disabled.
5. DSP Shift Out of Range: If an attempt is made to execute SFTAC with a shift value of greater than 32 or less than -32, the instruction will complete (without a result write) and a math trap will be generated.

10.9.1.6. Generic Trap

A generic trap occurs when any of the bits within the INTCON5 register are set. Each bit within the INTCON5 register is assigned to a specific trap error condition. Generic traps include WDT (run mode), DMT, DMA address error and software-generated traps.

10.9.1.7. PCTRAP

In any trap event, the Trap Origination Address register (PCTRAP) is loaded with the value of the PC associated with the instruction that caused the trap. The origination address of a trap is captured in the PCTRAP register, given that the current CPU IPL at the time of the trap is less than eight.

Further PCTRAP updates are blocked after the first PCTRAP address capture, preventing newer traps from overwriting the source address of older ones. The register needs to be written with value 24’h000000 for trap address capture to be re-enabled.

This feature is primarily intended to aid system debugging and to locate the cause of system traps.

Note: PCTRAP captures the trap origination address of the system error caused by the CPU only.

10.10. Interrupt Operations

10.10.1. Disabling Interrupts

10.10.1.1. DISI Inhibition Of Interrupts

The DISICTL instruction in dsPIC33A devices provides a means to disable interrupts around blocks of critical code. In addition, it allows the user to select an IPL threshold (IPLT) at which interrupts are disabled. The IPLT may be set to any value between IPL 0 (no interrupts disabled) and IPL 7 (all interrupts disabled). Interrupt requests at an IPL that is at or below the selected IPLT will be inhibited. These requests will remain pending until such time that the IPLT is lowered to a level less than the IPL of the pending interrupts. The IPLT may be defined using a 3-bit literal or register direct Wns source on the DISICTL instruction, and the updated threshold value is reflected in the DISIIPL register.

10.10.1.2. Global Interrupt Disable

A Global Interrupt Enable bit (GIE) is used to enable or disable all interrupts globally. When the GIE bit is cleared, it causes the interrupt controller to behave as if the CPU’s SR.IPL bits are set to seven and disables all interrupts except the traps. When the GIE bit is set again, the interrupt controller acts based on the actual value of SR.IPL; the system will return to the previous operating state, depending on the prior interrupt priority bit settings.

Additionally, individual interrupts can also be disabled by not setting corresponding enable bit in the IEC register or by configuring its priority level to zero.

10.10.2. External Interrupt Requests

The interrupt controller supports up to five external interrupt request signals, INT0 - INT4. These inputs are edge-sensitive, they require a low to high, or a high to low transition to create an interrupt request. The INTCON2 register has five bits, INT0EP - INT4EP, that select the polarity of the edge detection circuitry. Each external interrupt pin can be programmed to interrupt the CPU on a rising edge or falling edge event. INT0-INT2 can also be used as channel trigger sources for DMA or synchronization sources for CCP.

10.10.3. Wake-Up From Sleep, Idle

When an interrupt or trap request is received by the interrupt controller, a wake-up signal will be presented to the processor to wake the processor up. The processor will wake up from Sleep or Idle mode and resume operation.

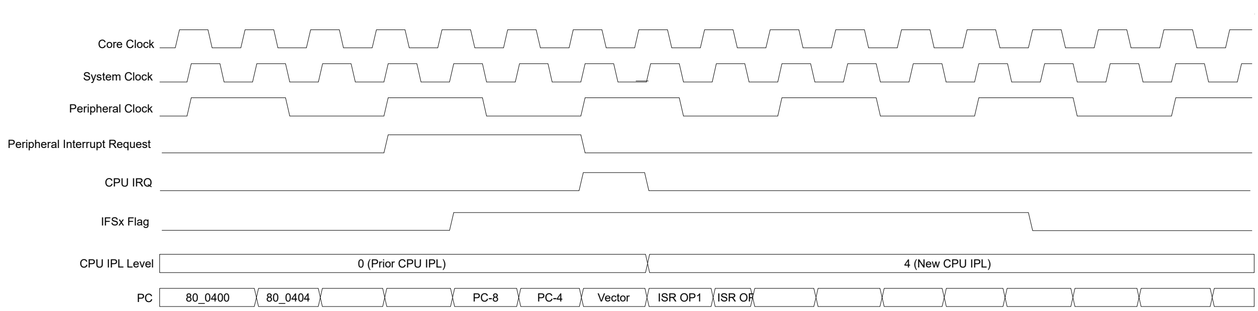
When the device wakes from Sleep or Idle mode, one of two actions occurs:

- If the IPL for that source is greater than the current CPU priority level, the processor will process the interrupt and branch to the ISR for the interrupt source.
- If the user application-assigned IPL for the source is lower than or equal to the current CPU priority level, the processor will continue execution, starting with the instruction immediately following the `PWRSAV` instruction that previously put the CPU in Sleep or Idle mode.
  Note: Interrupts assigned a priority level of zero are disabled for both interrupt handling and wake-up from Power-Saving modes.

10.10.4. Interrupt Processing Timing

The interrupt is sampled in the interrupt controller on the rising edge of the system clock into the IFSx registers. The priority resolution occurs during the next two clock cycles. The resolved interrupt request signal from the interrupt controller to the processor is then presented in the next system clock cycle. The interrupt is then acknowledged during the next system clock cycle.

Figure 10-4. Interrupt Latency



10.10.4.1. Interrupt Latency

Interrupt Latency: CPU Highest Priority Bus Main

If the CPU is set to be the highest priority RAM bus Main (BMX register BMXINITPR[31:0] = 0x00000000), the CPU will offer a variable latency response for all exceptions solely based upon the execution time of the instructions underway (that are completed) at the time of the exception. The interrupt latency from the time when the system clock samples the pending interrupts to the time the first instruction of the ISR has been fetched will be:

max. latency = $t_{arb} + \rho + \eta + \Delta$ cycles

min. latency = $t_{arb} + 1 + \eta + \Delta$ cycles

Where:

t_{arb} Arbitration time (cycles)

ρ = Total instruction execution time during exception processing (cycles)

η = Vector memory access time (cycles)

Δ = Program memory access time (cycles)

The above relationship applies to exceptions occurring during any instruction, including during a PS access. The latency is expressed as a range for any given instruction because the interrupt may arrive at the beginning or end of an instruction.

Note: ρ = Total instruction execution time during exception processing depends on current instruction under execution and may vary anywhere between one to two cycles depending upon the instruction.

η = Vector memory access time may vary anywhere between four to seven cycles and

Δ = Program memory access time may vary between one to seven cycles depending on the placement of the IVT and ISR or the current status of the instruction cache. Refer to the PBU section for more details.

Interrupt Latency:CPU Not Highest Priority Main

If the CPU is not the highest priority RAM bus Main, the CPU will offer a variable latency response for all exceptions that may also include additional delays resulting from higher priority bus master RAM access requests. That is, when the CPU is not operating as the highest priority bus Main, exception processing is no longer an atomic operation and may be stalled as necessary to provide bus access to another Main.

10.10.4.1.1. Interrupt Return Latency

To return from an interrupt, the program must call the RETFIE instruction. The RETFIE instruction in the dsPIC33A family of devices is expected to take around seven to 10 cycles with PBU enabled.

During the first two cycles of a RETFIE instruction, the contents of the PC and the SR register are popped from the stack. After unstacking the SR (in the second cycle), the RETFIE instruction will change the context and CPU.IPL level to that defined by the unstacked SR value. The new context will be immediately available at the start of execution of the next instruction.

The third instruction cycle is used to fetch the instruction addressed by the updated program counter. This cycle is executed as a `NOP` instruction. On the fourth cycle, program execution resumes at the point where the interrupt occurred.

10.10.5. Interrupt Setup Procedures

10.10.5.1. Initialization

To configure an interrupt source, complete the following steps:

1. If nested interrupts are not desired, set the NSTDIS Control bit (INTCON1).
2. Select the user application-assigned priority level for the interrupt source by writing to the control bits in the appropriate IPCx Control register. The priority level depends on the specific application and type of the interrupt source. If multiple priority levels are not desired, the IPCx register control bits for all enabled interrupt sources can be programmed to the same non-zero value.
3. Clear the interrupt flag status bit associated with the peripheral in the associated IFSx Status register.
4. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate IECx Control register.

10.10.5.2. Interrupt Service Routine

The method used to declare an ISR and initialize the IVT/RIVT with the correct vector address depends on the programming language (C or Assembly) and the language development tool suite used to develop the application.

In general, the user application must clear the interrupt flag in the appropriate IFSx register for the source of the interrupt that the ISR handles. Otherwise, the application will re-enter the ISR immediately after it exits the routine. If the ISR is coded in Assembly language, it must be terminated using a RETFIE instruction to unstack the saved PC value, SRL value and old CPU priority level.

10.10.5.3. Trap Service Routine

A Trap Service Routine is coded like an ISR, except that the appropriate trap status flag in the INTCONx register must be cleared to avoid re-entry into the Trap Service Routine.

10.10.5.4. Code Examples

Example 10-1. Enable Global Interrupts

```
void enableInterrupts (void)
{
/* Enable level 1-7 interrupts */
/* No restoring of previous CPU IPL state performed here */
INTCON1bits.GIE = 1;
}
```

Example 10-2. Disable Global Interrupts

```
void disableInterrupts (void)
{
/* Disable level 1-7 interrupts */
/* No restoring of previous CPU IPL state performed here */
INTCON1bits.GIE = 0;
}
```

Example 10-3. ISR for Timer 1 Interrupt

```
void __attribute__((interrupt, context)) _T1Interrupt(void)
{
/* Insert ISR Code Here*/
/* Clear Timer1 interrupt */
IFS1bits.T1IF = 0;
 }
```

11. I/O Ports with Edge Detect

The dsPIC33AK256MPS306 family devices include multiple general-purpose I/O ports to interface with external circuitry. The ports are highly configurable for a wide range of digital and analog applications. The edge detect feature allows sensing pin changes while in Sleep mode, or it can generate an interrupt to eliminate the need for the CPU to poll the pin state. The ports also include an integrity monitor function to verify the pin state in critical applications. The I/O integrity check feature is available on a subset of pins.

The key features of the I/O ports with the edge detect module are:

- Pin Control and Pin Configuration
- Flexible Remapping of Input and Output Signals
- Open Drain Operation
- Configurable Pin Pull-Up or Pull-Down Capability
- Change Notification:
  - Monitors for state change on device pins
  - Detects change with respect to the last PORT value
  - Detects positive and/or negative edge events
  - Individual change status for each pin
  - Generates an interrupt event per port
  - Operates in Sleep mode
- Slew Rate Control

The key features of the I/O integrity module are:

- Fault Detection:
  - Detects short circuits in the signal path
  - Detects open circuits in the signal path
  - Detects faulty signal conditions
  - Detects external tampering events
  - Device interrupt, trap or reset upon Fault detection
- Configurable Blanking Delay
- Fault Injection Capability

11.1. Device-Specific Information

Table 11-1. PORTA Availability

Device	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8					✓	✓	✓	✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
48-Pin	15:8							✓	✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
36-Pin	15:8								
	7:0	✓			✓	✓	✓	✓	✓

Table 11-2. PORTB Availability

Device	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8			✓	✓	✓	✓	✓	✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
48-Pin	15:8								
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
36-Pin	15:8								
	7:0		✓		✓	✓	✓	✓	✓

Table 11-3. PORTC Availability

Device	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8					✓	✓	✓	✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
48-Pin	15:8								
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
36-Pin	15:8								
	7:0			✓	✓		✓	✓	✓

Table 11-4. PORTD Availability

Device	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8								✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
48-Pin	15:8							✓	✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
36-Pin	15:8								
	7:0				✓	✓	✓	✓	✓

Table 11-5. ANSELA Availability

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8					✓	✓	✓	✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
48-Pin	15:8							✓	✓
	7:0	✓			✓	✓	✓	✓	✓
36-Pin	15:8								
	7:0	✓			✓	✓	✓	✓	✓

Table 11-6. ANSELB Availability

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8					✓	✓	✓	✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓
48-Pin	15:8							✓	✓
	7:0	✓	✓	✓	✓	✓	✓	✓	✓

Table 11-6. ANSELB Availability (continued)

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
36-Pin	15:8								
	7:0		✓		✓	✓	✓	✓	✓

Table 11-7. ANSELC Availability

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8								
	7:0	✓	✓						
48-Pin	15:8								
	7:0	✓	✓						
36-Pin	15:8								
	7:0								

Table 11-8. ANSELD Availability

Name	Bit Field	Bit 15/7	Bit 14/6	Bit 13/5	Bit 12/4	Bit 11/3	Bit 10/2	Bit 9/1	Bit 8/0
64-Pin	15:8								
	7:0		✓	✓					
48-Pin	15:8								
	7:0		✓						
36-Pin	15:8								
	7:0								

Table 11-9. PPS Availability by Package

64-Pin	48-Pin	36-Pin
RP1-RP12	RP1-RP10	RP1-RP12
RP17-RP30	RP17-RP26	RP17-RP30
RP33-RP44	RP33-RP40	RP33-RP44
RP49-RP57	RP49-RP52	RP49-RP57
	RP55	

Table 11-10. Selectable Input Sources (Maps Input to Function)

Input Name[1]	Function Name	Register	Register Bitfield
External Interrupt 1	INT1	RPINR0	INT1R[7:0]
External Interrupt 2	INT2	RPINR0	INT2R[7:0]
External Interrupt 3	INT3	RPINR0	INT3R[7:0]
External Interrupt 4	INT4	RPINR1	INT4R[7:0]
Timer1 External Clock	T1CK	RPINR1	T1CKR[7:0]
Timer2 External Clock	T2CK	RPINR1	T2CKR[7:0]
Timer3 External Clock	T3CK	RPINR1	T3CKR[7:0]
SCCP Input Clock 1	TCKI1	RPINR2	TCKI1R[7:0]
SCCP Input Capture 1	ICM1	RPINR2	ICM1R[7:0]

Note:

1. Unless otherwise noted, all inputs use the Schmitt Trigger input buffers.

Table 11-10. Selectable Input Sources (Maps Input to Function) (continued)

Input Name(1)	Function Name	Register	Register Bitfield
SCCP Input Clock 2	TCKI2	RPINR2	TCKI2R[7:0]
SCCP Input Capture 2	ICM2	RPINR2	ICM2R[7:0]
SCCP Input Clock 3	TCKI3	RPINR3	TCKI3R[7:0]
SCCP Input Capture 3	ICM3	RPINR3	ICM3R[7:0]
SCCP Input Clock 4	TCKI4	RPINR3	TCKI4R[7:0]
SCCP Input Capture 4	ICM4	RPINR3	ICM4R[7:0]
SCCP Input Clock 5	TCKI5	RPINR4	TCKI5R[7:0]
SCCP Input Capture 5	ICM5	RPINR4	ICM5R[7:0]
Reserved			
SCCP Fault A	OCFA	RPINR7	OCFAR[7:0]
SCCP Fault B	OCFB	RPINR7	OCFBR[7:0]
SCCP Fault C	OCFC	RPINR7	OCFCR[7:0]
SCCP Fault D	OCFD	RPINR7	OCFDR[7:0]
PWM Input 8	PCI8	RPINR8	PCI8R[7:0]
PWM Input 9	PCI9	RPINR8	PCI9R[7:0]
PWM Input 10	PCI10	RPINR8	PCI10R[7:0]
PWM Input 11	PCI11	RPINR8	PCI11R[7:0]
QEI1 Input A	QEIA1	RPINR9	QEIA1R[7:0]
QEI1 Input B	QEIB1	RPINR9	QEIB1R[7:0]
QEI1 Index Input	QEINDX1	RPINR9	QEINDX1R[7:0]
QEI1 Home Input	QEIHOME1	RPINR9	QEIHOM1R[7:0]
UART1 Receive	U1RX	RPINR13	U1RXR[7:0]
UART1 Data-Set-Ready	U1DSR	RPINR13	U1DSRR[7:0]
UART2 Receive	U2RX	RPINR13	U2RXR[7:0]
UART2 Data-Set-Ready	U2DSR	RPINR13	U2DSRR[7:0]
UART3 Receive	U3RX	RPINR14	U3RXR[7:0]
UART3 Data-Set-Ready	U3DSR	RPINR14	U3DSRR[7:0]
SPI1 Data Input	SDI1	RPINR14	SDI1R[7:0]
SPI1 Clock Input	SCK1IN	RPINR14	SCK1R[7:0]
SPI1 Client Select	SS1IN	RPINR15	SS1R[7:0]
SPI2 Data Input	SDI2	RPINR15	SDI2R[7:0]
SPI2 Clock Input	SCK2IN	RPINR15	SCK2R[7:0]
SPI2 Client Select	SS2IN	RPINR15	SS2R[7:0]
SPI3 Data Input	SDI3	RPINR16	SDI3R[7:0]
SPI3 Clock Input	SCK3IN	RPINR16	SCK3R[7:0]
SPI3 Client Select	SS3IN	RPINR16	SS3R[7:0]
CAN 1 Receive	CAN1RX	RPINR17	CAN1RXR[7:0]
SENT 1 Input	SENT1	RPINR18	SENT1R[7:0]
SENT 2 Input	SENT2	RPINR18	SENT2R[7:0]
Reference Clock 1 Input	REFI1	RPINR18	REFI1R[7:0]
Reference Clock 2 Input	REFI2	RPINR18	REFI2R[7:0]
PWM PCI Input 12	PCI12	RPINR19	PCI12R[7:0]
PWM PCI Input 13	PCI13	RPINR19	PCI13R[7:0]

Note:
1. Unless otherwise noted, all inputs use the Schmitt Trigger input buffers.

Table 11-10. Selectable Input Sources (Maps Input to Function) (continued)

Input Name[1]	Function Name	Register	Register Bitfield
PWM PCI Input 14	PCI14	RPINR19	PCI14R[7:0]
PWM PCI Input 15	PCI15	RPINR19	PCI15R[7:0]
PWM PCI Input 16	PCI16	RPINR20	PCI16R[7:0]
PWM PCI Input 17	PCI17	RPINR20	PCI17R[7:0]
PWM PCI Input 18	PCI18	RPINR20	PCI18R[7:0]
CLC Input A	CLCINA	RPINR20	CLCAR[7:0]
CLC Input B	CLCINB	RPINR21	CLCBR[7:0]
CLC Input C	CLCINC	RPINR21	CLCCR[7:0]
CLC Input D	CLCIND	RPINR21	CLCDR[7:0]
CLC Input E	CLCINE	RPINR21	CLCER[7:0]
CLC Input F	CLCINF	RPINR22	CLCFR[7:0]
CLC Input G	CLCING	RPINR22	CLCGR[7:0]
CLC Input H	CLCINH	RPINR22	CLCHR[7:0]
CLC Input I	CLCINI	RPINR22	CLCIR[7:0]
CLC Input J	CLCINJ	RPINR23	CLCJR[7:0]
ADC Trigger 31 Input	ADTRG31	RPINR23	ADTRG31R[7:0]
UART1 Clear to Send	U1CTS	RPINR23	U1CTSR[7:0]
UART2 Clear to Send	U2CTS	RPINR23	U2CTSR[7:0]
UART3 Clear to Send	U3CTS	RPINR24	U3CTSR[7:0]
BiSS Return Input	BISS1SL	RPINR24	BISS1SLR[7:0]
BiSS Get Sense Input	BISS1GS	RPINR24	BISS1GSR[7:0]
IOMONITOR1 Feedback 12	IOMON1F12	RPINR24	IOIM0R[7:0]
IOMONITOR1 Feedback 13	IOMON1F13	RPINR25	IOIM1R[7:0]
IOMONITOR1 Feedback 14	IOMON1F14	RPINR25	IOIM2R[7:0]
IOMONITOR1 Feedback 15	IOMON1F15	RPINR25	IOIM3R[7:0]
PWM Input 19	PCI19	RPINR26	PCI19R[7:0]
PWM Input 20	PCI20	RPINR27	PCI20R[7:0]
PWM Input 21	PCI21	RPINR27	PCI21R[7:0]
PWM Input 22	PCI22	RPINR27	PCI22R[7:0]
UART4 Receive	U4DS	RPINR28	U4DSR[7:0]
UART4 Data-Set-Ready	U4RX	RPINR28	U4DSR[7:0]
UART4 Clear to Send	U4CTS	RPINR28	U4CTSR[7:0]

Note:

1. Unless otherwise noted, all inputs use the Schmitt Trigger input buffers.

Table 11-11. Pin Correlation to Input Remap #[1]

Index	Source Signal	Remap Input #
181	RPV15	PI181
180	RPV14	PI180
179	RPV13	PI179
178	RPV12	PI178
177	RPV11	PI177
176	RPV10	PI176
175	RPV9	PI175
174	RPV8	PI174

Table 11-11. Pin Correlation to Input Remap #[1] (continued)

Index	Source Signal	Remap Input #
173	RPV7	PI173
172	RPV6	PI172
171	RPV5	PI171
170	RPV4	PI170
169	RPV3	PI169
168	RPV2	PI168
167	RPV1	PI167
166	RPV0	PI166
165	Touch TX8	PI165
164	Touch TX9	PI164
163	Touch TX10	PI163
162	Touch TX11	PI162
161	Touch TX12	PI161
160	Touch TX13	PI160
159	Touch TX14	PI159
158	Touch TX15	PI158
157-152	Reserved	PI157-PI152
151	PWM Off Request - DAC5	PI151
150	PWM On Request - DAC5	PI150
149	PWM Off Request - DAC4	PI149
148	PWM On Request - DAC4	PI148
147	PWM Off Request - DAC3	PI147
146	PWM On Request - DAC3	PI146
145	PWM Off Request - DAC2	PI145
144	PWM On Request - DAC2	PI144
143	PWM Off Request - DAC1	PI143
142	PWM On Request - DAC1	PI142
141	Reserved	PI141
140	PWM Event Out 2	PI140
139	PWM Event Out 1	PI139
138	PTG TRIG[27]	PI138
137	PTG TRIG[26]	PI137
136-134	Reserved	Reserved
133	CMP5	PI133
132	CMP4	PI132
131	CMP3	PI131
130	CMP2	PI130
129	CMP1	PI129
128-58	Reserved	Reserved
57	RD8	RPI57
56	RD7	RPI56
55	RD6	RPI55
54	RD5	RPI54
53	RD4	RPI53
52	RD3	RPI52

Table 11-11. Pin Correlation to Input Remap #[1] (continued)

Index	Source Signal	Remap Input #
51	RD2	RPI51
50	RD1	RPI50
49	RD0	RPI49
48-45	Reserved	RPI48-RPI145
44	RC11	RPI44
43	RC10	RPI43
42	RC9	RPI42
41	RC8	RPI41
40	RC7	RPI40
39	RC6	RPI39
38	RC5	RPI38
37	RC4	RPI37
36	RC3	RPI36
35	RC2	RPI35
34	RC1	RPI34
33	RC0	RPI33
32-31	Reserved	RPI32-RPI31
30	RB13	RPI30
29	RB12	RPI29
28	RB11	RPI28
27	RB10	RPI27
26	RB9	RPI26
25	RB8	RPI25
24	RB7	RPI24
23	RB6	RPI23
22	RB5	RPI22
21	RB4	RPI21
20	RB3	RPI20
19	RB2	RPI19
18	RB1	RPI18
17	RB0	RPI17
16-13	Reserved	RPI16-RPI13
12	RA11	RPI12
11	RA10	RPI11
10	RA9	RPI10
9	RA8	RPI9
8	RA7	RPI8
7	RA6	RPI7
6	RA5	RPI6
5	RA4	RPI5
4	RA3	RPI4
3	RA2	RPI3
2	RA1	RPI2
1	RA0	RPI1

Table 11-11. Pin Correlation to Input Remap #[(1)] (continued)

Index	Source Signal	Remap Input #

Note:
1. This list of output signals can be mapped to any of the peripheral inputs listed in Table 11-10.

Table 11-12. Virtual Outputs to Remappable Ouput Registers[(1)]

Virtual Outputs	Remappable Output Register	Register Bitfield
RPV0	RPOR32	RP128R[6:0]
RPV1	RPOR32	RP129R[6:0]
RPV2	RPOR32	RP130R[6:0]
RPV3	RPOR32	RP131R[6:0]
RPV4	RPOR33	RP132R[6:0]
RPV5	RPOR33	RP133R[6:0]
RPV6	RPOR33	RP134R[6:0]
RPV7	RPOR33	RP135R[6:0]
RPV8	RPOR34	RP136R[6:0]
RPV9	RPOR34	RP137R[6:0]
RPV10	RPOR34	RP138R[6:0]
RPV11	RPOR34	RP139R[6:0]
RPV12	RPOR35	RP140R[6:0]
RPV13	RPOR35	RP141R[6:0]
RPV14	RPOR35	RP142R[6:0]
RPV15	RPOR35	RP143R[6:0]

Note:
1. This list of virtual output signals can be mapped to any of the peripheral inputs listed in Table 11-10.

Table 11-13. Output Selection for Remappable Pins (RPn)

Function	RPnR[5:0]	Output Name
PWM1H	1	RPn tied to PWM1H Output
PWM1L	2	RPn tied to PWM1L Output
PWM2H	3	RPn tied to PWM2H Output
PWM2L	4	RPn tied to PWM2L Output
PWM3H	5	RPn tied to PWM3H Output
PWM3L	6	RPn tied to PWM3L Output
PWM4H	7	RPn tied to PWM4H Output
PWM4L	8	RPn tied to PWM4L Output
CAN1TX	9	RPn tied to CAN1 Output
U1TX	10	RPn tied to UART1 Transmit
U1RTS	11	RPn tied to UART1 Request-to-Send
U2TX	12	RPn tied to UART2 Transmit
U2RTS	13	RPn tied to UART2 Request-to-Send
U3TX	14	RPn tied to UART3 Transmit
U3RTS	15	RPn tied to UART3 Request-to-Send
U4TX	16	RPn tied to UART4 Transmit
U4RTS	17	RPn tied to UART4 Request-to-Send
SDO1	18	RPn tied to SPI1 Data Output
SCK1	19	RPn tied to SPI1 Clock Output
SS1	20	RPn tied to SPI1 Client Select

Table 11-13. Output Selection for Remappable Pins (RPn) (continued)

Function	RPnR[5:0]	Output Name
SDO2	21	RPn tied to SPI2 Data Output
SCK2	22	RPn tied to SPI2 Clock Output
SS2	23	RPn tied to SPI2 Client Select
SDO3	24	RPn tied to SPI3 Data Output
SCK3	25	RPn tied to SPI3 Clock Output
SS3	26	RPn tied to SPI3 Client Select
REFO1	27	RPn tied to Reference Clock 1 Output
REFO2	28	RPn tied to Reference Clock 2 Output
OCM1	29	RPn tied to SCCP1 Output
OCM2	30	RPn tied to SCCP2 Output
OCM3	31	RPn tied to SCCP3 Output
OCM4	32	RPn tied to SCCP4 Output
MCCP5A	33	RPn tied to MCCP5 Output A
MCCP5B	34	RPn tied to MCCP5 Output B
MCCP5C	35	RPn tied to MCCP5 Output C
MCCP5D	36	RPn tied to MCCP5 Output D
MCCP5E	37	RPn tied to MCCP5 Output E
MCCP5F	38	RPn tied to MCCP5 Output F
CMP1	39	RPn tied to Comparator 1 Output
CMP2	40	RPn tied to Comparator 2 Output
CMP3	41	RPn tied to Comparator 3 Output
CMP4	42	RPn tied to Comparator 4 Output
CMP5	43	RPn tied to Comparator 5 Output
PEVTA	44	RPn tied to PWM Event A Output
PEVTB	45	RPn tied to PWM Event B Output
PEVTC	46	RPn tied to PWM Event C Output
PEVTD	47	RPn tied to PWM Event D Output
PWME	48	RPn tied to PWM Event E Output
PWMF	49	RPn tied to PWM Event F Output
QEICMP1	50	RPn tied to QEI1 Comparator Output
CLC1OUT	51	RPn tied to CLC1 Output
CLC2OUT	52	RPn tied to CLC2 Output
CLC3OUT	53	RPn tied to CLC3 Output
CLC4OUT	54	RPn tied to CLC4 Output
PTGTRG24	55	RPn tied to PTG Trigger 24 Output
PTGTRG25	56	RPn tied to PTG Trigger 25 Output
SENT1OUT	57	RPn tied to SENT1 Output
SENT2OUT	58	RPn tied to SENT2 Output
BISSMO1	59	RPn tied to BiSS Line Digital Output 1 Enable
BISSMA1	60	RPn tied to BiSS Line Digital Clock 1
U1DTRn	61	RPn tied to UART1 Data Terminal Ready Output
U2DTRn	62	RPn tied to UART2 Data Terminal Ready Output
U3DTRn	63	RPn tied to UART3 Data Terminal Ready Output
U4DTRn	64	RPn tied to UART4 Data Terminal Ready Output
RDCEXC	65	RPn tied to RDC Excitation Output

Table 11-13. Output Selection for Remappable Pins (RPn) (continued)

Function	RPnR[5:0]	Output Name
RDCEXCI	66	RPn tied to RDC Excitation Inverted Output
PTGTRG18	67	RPn tied to PTG Trigger 18 Output
PTGTRG19	68	RPn tied to PTG Trigger 19 Output
PTGTRG20	69	RPn tied to PTG Trigger 20 Output

Table 11-14. IOIM 1-8 Group A Reference Pins

REFSEL	Pin Function Name	Pin
0111	IOMAD7	RC2
0110	IOMAD6	RC5
0101	IOMAD5	RC3
0100	IOMAD4	RC4
0011	IOMAD3	RD0
0010	IOMAD2	RD1
0001	IOMAD1	RD2
0000	IOMAD0	RD3

Table 11-15. IOIM 1-8 Group A Feedback Pins

FBKSEL	Pin Function Name	Pad
0111	IOMAF7	RC7
0110	IOMAF6	RC1
0101	IOMAF5	RC0
0100	IOMAF4	RD7
0011	IOMAF3	RD8
0010	IOMAF2	RA0
0001	IOMAF1	RA7
0000	IOMAF0	RB7

11.2. Architectural Overview

A general purpose I/O port that shares a pin with a peripheral is generally subservient to the peripheral. Once enabled, the peripheral selects whether the peripheral or the associated port has ownership of the I/O pin.

An MUX and its associated logic controls interaction between peripherals and port logic. When a peripheral is enabled but the peripheral is not actively driving a pin, the port is still allowed to drive the pin. This is useful for “loop through”, in which a port’s digital output drives the input of a peripheral that shares the same pin.

When a peripheral is enabled and actively driving an associated pin, the IO MUX disables the use of the pin as a general purpose output. The I/O pin value may be read by the port, but the LATx[n] output value for the port is ignored.

The output and input circuits of the I/O are independent. The block diagrams of the output and input I/O circuits are shown in Figure 11-1 and Figure 11-2.

Figure 11-1. Output Circuits of the I/O

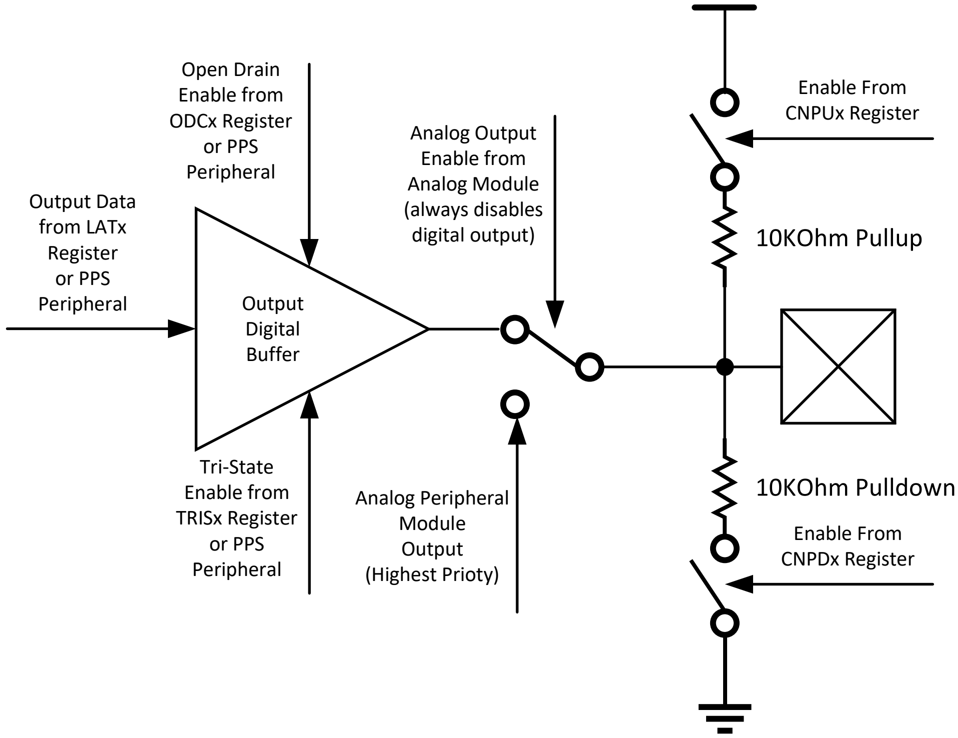
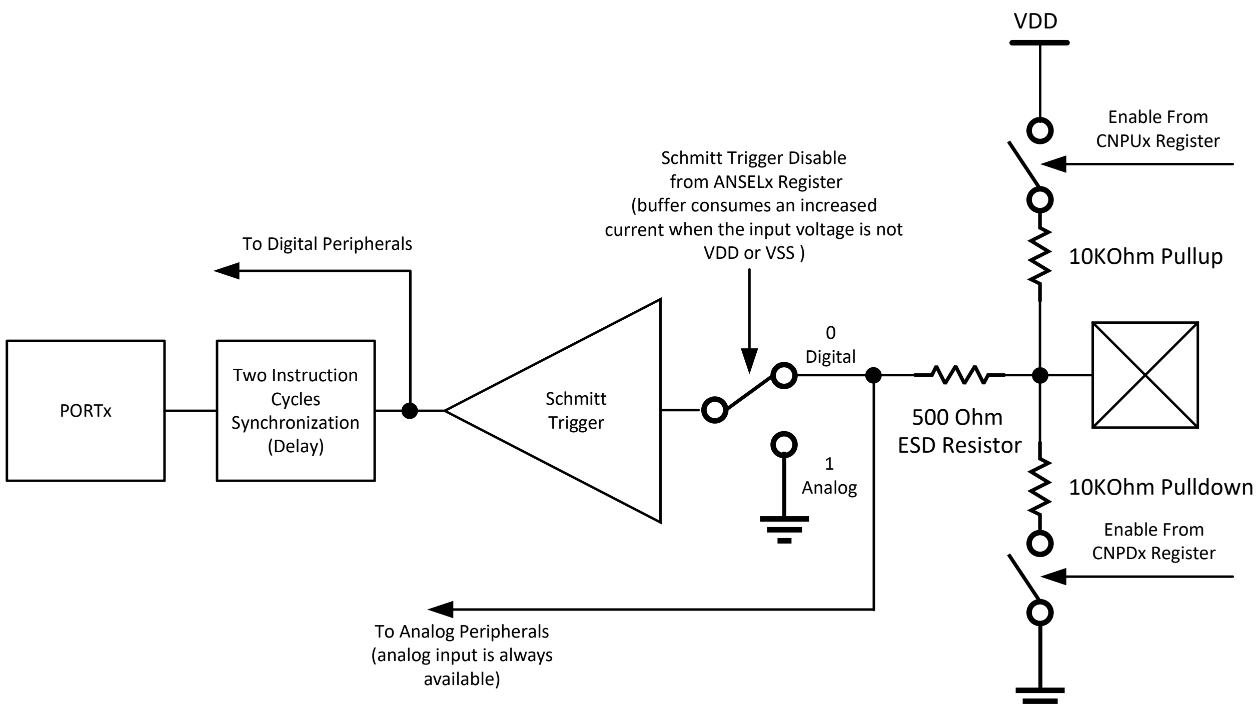


Figure 11-2. Input Circuits of the I/O



11.2.1. Peripheral Pin Select (PPS) Overview

The PPS feature allows remapping of pin and port functions and can be configured to best suit an application. This simplifies board design and allows peripherals to be connected to one another with or without a pin.

Figure 11-3 shows the structure of a remappable input. The control logic is based on the peripheral, through selection of which Remappable Pin (RPn) is used as the input source. Each peripheral is associated with a register bit field to select RPn. More than one peripheral can select the same input pin.

Figure 11-3. Remappable Input for U1RX

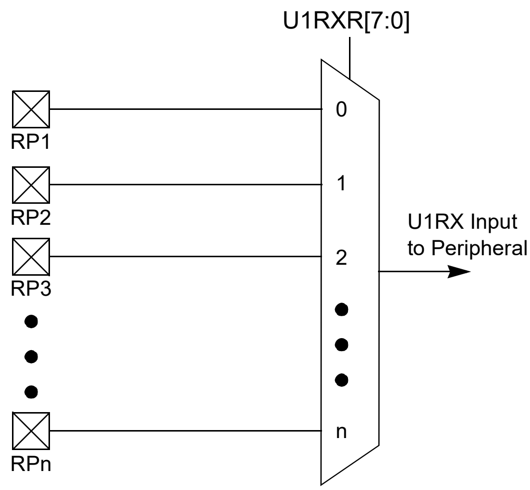
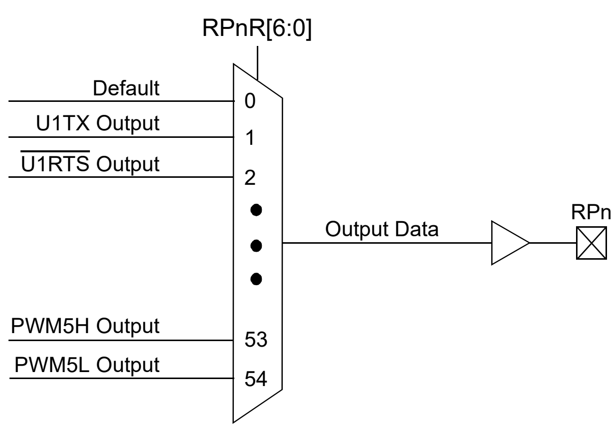


Figure 11-4 shows the structure of a remappable output. Unlike the inputs, the control logic is based on the remappable pin. Each RPn pin has an associated register bit field to select which peripheral output is routed to the pin. The output of one peripheral can be mapped to multiple RPn pins.

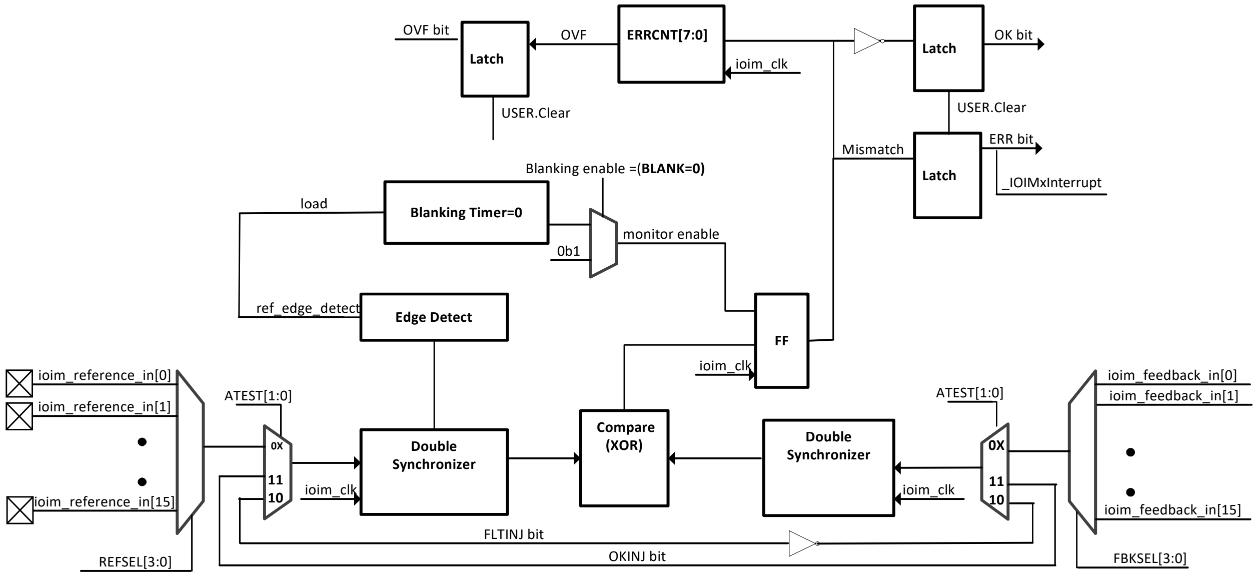
Figure 11-4. Multiplexing of Remappable Outputs for RPn



11.2.2. I/O Integrity Monitor (IOIM) Overview

The port module includes integrity monitoring circuity, as shown in Figure 11-5, to validate I/O functionality in critical applications by comparing a device output signal against a reference signal. If a mismatch is detected, an event is generated to allow software to take action as needed for the application. A programmable blanking timer is included to account for the feedback path delay. The timer is reset on a change of state in the reference signal. A counter is included to keep track of the number of mismatch events.

Figure 11-5. IOIM Block Diagram



11.3. Register Summary

Note: SFR bit availability is defined in Table 11-1 through Table 11-4 for each device variant and port, respectively.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0200	PORTA	31:24								
		23:16								
		15:8	PORTA[15:8]							
		7:0	PORTA[7:0]							
0x0204	LATA	31:24								
		23:16								
		15:8	LATA[15:8]							
		7:0	LATA[7:0]							
0x0208	TRISA	31:24								
		23:16								
		15:8	TRISA[15:8]							
		7:0	TRISA[7:0]							
0x020C	CNSTATA	31:24								
		23:16								
		15:8	CNSTATA[15:8]							
		7:0	CNSTATA[7:0]							
0x0210	CNFA	31:24								
		23:16								
		15:8	CNFA[15:8]							
		7:0	CNFA[7:0]							
0x0214	PORTB	31:24								
		23:16								
		15:8	PORTB[15:8]							
		7:0	PORTB[7:0]							
0x0218	LATB	31:24								
		23:16								
		15:8	LATB[15:8]							
		7:0	LATB[7:0]							
0x021C	TRISB	31:24								
		23:16								
		15:8	TRISB[15:8]							
		7:0	TRISB[7:0]							
0x0220	CNSTATB	31:24								
		23:16								
		15:8	CNSTATB[15:8]							
		7:0	CNSTATB[7:0]							
0x0224	CNFB	31:24								
		23:16								
		15:8	CNFB[15:8]							
		7:0	CNFB[7:0]							
0x0228	PORTC	31:24								
		23:16								
		15:8	PORTC[15:8]							
		7:0	PORTC[7:0]							
0x022C	LATC	31:24								
		23:16								
		15:8	LATC[15:8]							
		7:0	LATC[7:0]							
0x0230	TRISC	31:24								
		23:16								
		15:8	TRISC[15:8]							
		7:0	TRISC[7:0]							
0x0234	CNSTATC	31:24								
		23:16								
		15:8	CNSTATC[15:8]							
		7:0	CNSTATC[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0238	CNFC	31:24								
		23:16								
		15:8	CNFC[15:8]							
		7:0	CNFC[7:0]							
0x023C	PORTD	31:24								
		23:16								
		15:8	PORTD[15:8]							
		7:0	PORTD[7:0]							
0x0240	LATD	31:24								
		23:16								
		15:8	LATD[15:8]							
		7:0	LATD[7:0]							
0x0244	TRISD	31:24								
		23:16								
		15:8	TRISD[15:8]							
		7:0	TRISD[7:0]							
0x0248	CNSTATD	31:24								
		23:16								
		15:8	CNSTATD[15:8]							
		7:0	CNSTATD[7:0]							
0x024C	CNFD	31:24								
		23:16								
		15:8	CNFD[15:8]							
		7:0	CNFD[7:0]							
0x0250 ... 0x1ECF	Reserved									
0x1ED0	IOIM1CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1ED4	IOIM1BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1ED8	IOIM1STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1EDC	IOIM2CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1EE0	IOIM2BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1EE4	IOIM2STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1EE8	IOIM3CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1EEC	IOIM3BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1EF0	IOIM3STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1EF4	IOIM4CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1EF8	IOIM4BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1EFC	IOIM4STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F00	IOIM5CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1F04	IOIM5BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F08	IOIM5STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F0C	IOIM6CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1F10	IOIM6BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F14	IOIM6STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F18	IOIM7CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1F1C	IOIM7BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							
0x1F20	IOIM7STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F24	IOIM8CON	31:24					FLTINJ	OKINJ	ATEST[1:0]	
		23:16	EOVFV[7:0]							
		15:8	ON		SLPEN	SIDL		EXTCLK		
		7:0	FBKSEL[3:0]				REFSEL[3:0]			
0x1F28	IOIM8BCON	31:24								
		23:16								
		15:8	BLANK[15:8]							
		7:0	BLANK[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1F2C	IOIM8STAT	31:24								
		23:16								
		15:8	ERRCNT[7:0]							
		7:0	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
0x1F30 ... 0x32CF	Reserved									
0x32D0	RPCON	31:24								
		23:16								
		15:8					IOLOCK			
		7:0								
0x32D4	RPINR0	31:24	INT3R[7:0]							
		23:16	INT2R[7:0]							
		15:8	INT1R[7:0]							
		7:0								
0x32D8	RPINR1	31:24	T3CKR[7:0]							
		23:16	T2CKR[7:0]							
		15:8	T1CKR[7:0]							
		7:0	INT4R[7:0]							
0x32DC	RPINR2	31:24	ICM2R[7:0]							
		23:16	TCKI2R[7:0]							
		15:8	ICM1R[7:0]							
		7:0	TCKI1R[7:0]							
0x32E0	RPINR3	31:24	ICM4R[7:0]							
		23:16	TCKI4R[7:0]							
		15:8	ICM3R[7:0]							
		7:0	TCKI3R[7:0]							
0x32E4	RPINR4	31:24								
		23:16								
		15:8	ICM5R[7:0]							
		7:0	TCKI5R[7:0]							
0x32E8 ... 0x32EF	Reserved									
0x32F0	RPINR7	31:24	OCFDR[7:0]							
		23:16	OCFCR[7:0]							
		15:8	OCFBR[7:0]							
		7:0	OCFAR[7:0]							
0x32F4	RPINR8	31:24	PCI11R[7:0]							
		23:16	PCI10R[7:0]							
		15:8	PCI9R[7:0]							
		7:0	PCI8R[7:0]							
0x32F8	RPINR9	31:24	HOME1R[7:0]							
		23:16	INDX1R[7:0]							
		15:8	QEB1R[7:0]							
		7:0	QEA1R[7:0]							
0x32FC ... 0x3307	Reserved									
0x3308	RPINR13	31:24	U2DSRR[7:0]							
		23:16	U2RXR[7:0]							
		15:8	U1DSRR[7:0]							
		7:0	U1RXR[7:0]							
0x330C	RPINR14	31:24	SCK1R[7:0]							
		23:16	SDI1R[7:0]							
		15:8	U3DSRR[7:0]							
		7:0	U3RXR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3310	RPINR15	31:24	SS2R[7:0]							
		23:16	SCK2R[7:0]							
		15:8	SDI2R[7:0]							
		7:0	SS1R[7:0]							
0x3314	RPINR16	31:24								
		23:16	SS3R[7:0]							
		15:8	SCK3R[7:0]							
		7:0	SDI3R[7:0]							
0x3318	RPINR17	31:24								
		23:16	CAN1RXR[7:0]							
		15:8								
		7:0								
0x331C	RPINR18	31:24	REFI2R[7:0]							
		23:16	REFI1R[7:0]							
		15:8	SENT2R[7:0]							
		7:0	SENT1R[7:0]							
0x3320	RPINR19	31:24	PCI15R[7:0]							
		23:16	PCI14R[7:0]							
		15:8	PCI13R[7:0]							
		7:0	PCI12R[7:0]							
0x3324	RPINR20	31:24	CLCINAR[7:0]							
		23:16	PCI18R[7:0]							
		15:8	PCI17R[7:0]							
		7:0	PCI16R[7:0]							
0x3328	RPINR21	31:24	CLCINER[7:0]							
		23:16	CLCINDR[7:0]							
		15:8	CLCINCR[7:0]							
		7:0	CLCINBR[7:0]							
0x332C	RPINR22	31:24	CLCINIR[7:0]							
		23:16	CLCINHR[7:0]							
		15:8	CLCINGR[7:0]							
		7:0	CLCINFR[7:0]							
0x3330	RPINR23	31:24	U2CTSR[7:0]							
		23:16	U1CTSR[7:0]							
		15:8	ADTRG31R[7:0]							
		7:0	CLCINJR[7:0]							
0x3334	RPINR24	31:24	IOM0R[7:0]							
		23:16	BISS1GSR[7:0]							
		15:8	BISS1SLR[7:0]							
		7:0	U3CTSR[7:0]							
0x3338	RPINR25	31:24								
		23:16	IOM3R[7:0]							
		15:8	IOM2R[7:0]							
		7:0	IOM1R[7:0]							
0x333C	RPINR26	31:24	PCI19R[7:0]							
		23:16								
		15:8								
		7:0								
0x3340	RPINR27	31:24								
		23:16	PCI22R[7:0]							
		15:8	PCI21R[7:0]							
		7:0	PCI20R[7:0]							
0x3344	RPINR28	31:24								
		23:16	U4DSRR[7:0]							
		15:8	U4CTSR[7:0]							
		7:0	U4RXR[7:0]							
0x3348 ... 0x334F	Reserved									

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3350	RPOR0	31:24		RP4R[6:0]						
		23:16		RP3R[6:0]						
		15:8		RP2R[6:0]						
		7:0		RP1R[6:0]						
0x3354	RPOR1	31:24		RP8R[6:0]						
		23:16		RP7R[6:0]						
		15:8		RP6R[6:0]						
		7:0		RP5R[6:0]						
0x3358	RPOR2	31:24								
		23:16		RP11R[6:0]						
		15:8		RP10R[6:0]						
		7:0		RP9R[6:0]						
0x335C ... 0x335F	Reserved									
0x3360	RPOR4	31:24		RP20R[6:0]						
		23:16		RP19R[6:0]						
		15:8		RP18R[6:0]						
		7:0		RP17R[6:0]						
0x3364	RPOR5	31:24		RP24R[6:0]						
		23:16		RP23R[6:0]						
		15:8		RP22R[6:0]						
		7:0		RP21R[6:0]						
0x3368	RPOR6	31:24		RP28R[6:0]						
		23:16		RP27R[6:0]						
		15:8		RP26R[6:0]						
		7:0		RP25R[6:0]						
0x336C	RPOR7	31:24								
		23:16								
		15:8		RP30R[6:0]						
		7:0		RP29R[6:0]						
0x3370	RPOR8	31:24		RP36R[6:0]						
		23:16		RP35R[6:0]						
		15:8		RP34R[6:0]						
		7:0		RP33R[6:0]						
0x3374	RPOR9	31:24		RP40R[6:0]						
		23:16		RP39R[6:0]						
		15:8		RP38R[6:0]						
		7:0		RP37R[6:0]						
0x3378	RPOR10	31:24		RP44R[6:0]						
		23:16		RP43R[6:0]						
		15:8		RP42R[6:0]						
		7:0		RP41R[6:0]						
0x337C ... 0x337F	Reserved									
0x3380	RPOR12	31:24		RP52R[6:0]						
		23:16		RP51R[6:0]						
		15:8		RP50R[6:0]						
		7:0		RP49R[6:0]						
0x3384	RPOR13	31:24		RP56R[6:0]						
		23:16		RP55R[6:0]						
		15:8		RP54R[6:0]						
		7:0		RP53R[6:0]						
0x3388	RPOR14	31:24								
		23:16								
		15:8								
		7:0		RP57R[6:0]						

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x338C ... 0x33CF	Reserved									
0x33D0	RPOR32	31:24		RP132R[6:0]						
		23:16		RP131R[6:0]						
		15:8		RP130R[6:0]						
		7:0		RP129R[6:0]						
0x33D4	RPOR33	31:24		RP136R[6:0]						
		23:16		RP135R[6:0]						
		15:8		RP134R[6:0]						
		7:0		RP133R[6:0]						
0x33D8	RPOR34	31:24		RP140R[6:0]						
		23:16		RP139R[6:0]						
		15:8		RP138R[6:0]						
		7:0		RP137R[6:0]						
0x33DC	RPOR35	31:24		RP144R[6:0]						
		23:16		RP143R[6:0]						
		15:8		RP142R[6:0]						
		7:0		RP141R[6:0]						
0x33E0 ... 0x363F	Reserved									
0x3640	ANSELA	31:24								
		23:16								
		15:8	ANSELA[15:8]							
		7:0	ANSELA[7:0]							
0x3644	ODCA	31:24								
		23:16								
		15:8	ODCA[15:8]							
		7:0	ODCA[7:0]							
0x3648	CNPUA	31:24								
		23:16								
		15:8	CNPUA[15:8]							
		7:0	CNPUA[7:0]							
0x364C	CNPDA	31:24								
		23:16								
		15:8	CNPDA[15:8]							
		7:0	CNPDA[7:0]							
0x3650	CNCONA	31:24								
		23:16								
		15:8	ON				CNSTYLE	PORT32		
		7:0								
0x3654	CNEN0A	31:24								
		23:16								
		15:8	CNEN0A[15:8]							
		7:0	CNEN0A[7:0]							
0x3658	CNEN1A	31:24								
		23:16								
		15:8	CNEN1A[15:8]							
		7:0	CNEN1A[7:0]							
0x365C ... 0x3663	Reserved									
0x3664	ANSELB	31:24								
		23:16								
		15:8	ANSELB[15:8]							
		7:0	ANSELB[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3668	ODCB	31:24								
		23:16								
		15:8	ODCB[15:8]							
		7:0	ODCB[7:0]							
0x366C	CNPUB	31:24								
		23:16								
		15:8	CNPUB[15:8]							
		7:0	CNPUB[7:0]							
0x3670	CNPDB	31:24								
		23:16								
		15:8	CNPDB[15:8]							
		7:0	CNPDB[7:0]							
0x3674	CNCONB	31:24								
		23:16								
		15:8	ON				CNSTYLE	PORT32		
		7:0								
0x3678	CNEN0B	31:24								
		23:16								
		15:8	CNEN0B[15:8]							
		7:0	CNEN0B[7:0]							
0x367C	CNEN1B	31:24								
		23:16								
		15:8	CNEN1B[15:8]							
		7:0	CNEN1B[7:0]							
0x3680 ... 0x3687	Reserved									
0x3688	ANSELC	31:24								
		23:16								
		15:8	ANSELC[15:8]							
		7:0	ANSELC[7:0]							
0x368C	ODCC	31:24								
		23:16								
		15:8	ODCC[15:8]							
		7:0	ODCC[7:0]							
0x3690	CNPUC	31:24								
		23:16								
		15:8	CNPUC[15:8]							
		7:0	CNPUC[7:0]							
0x3694	CNPDC	31:24								
		23:16								
		15:8	CNPDC[15:8]							
		7:0	CNPDC[7:0]							
0x3698	CNCONC	31:24								
		23:16								
		15:8	ON				CNSTYLE	PORT32		
		7:0								
0x369C	CNEN0C	31:24								
		23:16								
		15:8	CNEN0C[15:8]							
		7:0	CNEN0C[7:0]							
0x36A0	CNEN1C	31:24								
		23:16								
		15:8	CNEN1C[15:8]							
		7:0	CNEN1C[7:0]							
0x36A4 ... 0x36AF	Reserved									

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x36B0	ODCD	31:24								
		23:16								
		15:8	ODCD[15:8]							
		7:0	ODCD[7:0]							
0x36B4	CNPUD	31:24								
		23:16								
		15:8	CNPUD[15:8]							
		7:0	CNPUD[7:0]							
0x36B8	CNPDD	31:24								
		23:16								
		15:8	CNPDD[15:8]							
		7:0	CNPDD[7:0]							
0x36BC	CNCOND	31:24								
		23:16								
		15:8	ON				CNSTYLE	PORT32		
		7:0								
0x36C0	CNEN0D	31:24								
		23:16								
		15:8	CNEN0D[15:8]							
		7:0	CNEN0D[7:0]							
0x36C4	CNEN1D	31:24								
		23:16								
		15:8	CNEN1D[15:8]							
		7:0	CNEN1D[7:0]							

11.3.1. Input Data Register

Name: PORTx
Offset: 0x0200, 0x0214, 0x0228, 0x023C

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PORTx[15:8]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PORTx[7:0]							
Access	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC	R/W/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PORTx[15:0] Input Data bits

11.3.2. Output Data Register

Name: LATx
Offset: 0x0204, 0x0218, 0x022C, 0x0240

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LATx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LATx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LATx[15:0] Output Data bits

11.3.3. Tri-State Enable Register

Name: TRISx
Offset: 0x0208, 0x021C, 0x0230, 0x0244

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	TRISx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	TRISx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:0 – TRISx[15:0] Tri-State Enable bits

Value	Description
1	I/O is tri-stated.
0	I/O is driven with LATx register bit value.

11.3.4. Interrupt Change Notification Status Register

Name: CNSTATx
Offset: 0x020C, 0x0220, 0x0234, 0x0248

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CNSTATx[15:8]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNSTATx[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CNSTATx[15:0] Interrupt Change Notification Status bits
When CNSTYLE (CNCONx[11]) = 0:

Value	Description
1	A change occurred on PORTx bit since last read.
0	A change did not occur on PORTx bit since last read.

11.3.5. Interrupt Change Notification Flag for Register

Name: CNFx
Offset: 0x0210, 0x0224, 0x0238, 0x024C

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CNFx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNFx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CNFx[15:0] Interrupt Change Notification Flag bits
When CNSTYLE (CNCONx[11]) = 1:

Value	Description
1	An enabled edge event occurred on the PORTx[n] bit.
0	An enabled edge event did not occur on the PORTx[n] bit.

11.3.6. Analog Select Register

Name: ANSELx
Offset: 0x3640, 0x3664, 0x3688

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ANSELx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	ANSELx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	0

Bits 15:0 – ANSELx[15:0] Analog Select bits

Value	Description
1	Digital Schmitt Trigger is disabled on the I/O.
0	Digital Schmitt Trigger is enabled on the I/O.

11.3.7. Open-Drain Enable Register

Name: ODCx
Offset: 0x3644, 0x3668, 0x368C, 0x36B0

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ODCx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ODCx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – ODCx[15:0] Open-Drain Enable bits

Value	Description
1	Open-drain is enabled on the I/O.
0	Open-drain is disabled on the I/O.

11.3.8. Change Notification Pull-up Enable Register

Name: CNPUx
Offset: 0x3648, 0x366C, 0x3690, 0x36B4

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CNPUx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNPUx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CNPUx[15:0] Pull-up Enable bits

Value	Description
1	The pull-up for I/O is enabled.
0	The pull-up for I/O is disabled.

11.3.9. Pull-Down Enable Register

Name: CNPDx
Offset: 0x364C, 0x3670, 0x3694, 0x36B8

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CNPDx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNPDx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CNPDx[15:0] Pull-Down Enable bits

Value	Description
1	The pull-down for I/O is enabled.
0	The pull-down for I/O is disabled.

11.3.10. Change Notification Control Register

Name: CNCONx
Offset: 0x3650, 0x3674, 0x3698, 0x36BC

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON				CNSTYLE	PORT32		
Access	R/W				R/W	R/W		
Reset	0				0	0		

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON Change Notification (CN) Control for PORTx On bit

Value	Description
1	CN is enabled.
0	CN is disabled.

Bit 11 – CNSTYLE Change Notification Style Selection bit

Value	Description
1	Edge style (detects edge transitions; CNFx[15:0] bits are used for a Change Notification event)
0	Mismatch style (detects change from last port read; CNSTATx[15:0] bits are used for a Change Notification event)

Bit 10 – PORT32 Selects between 16-bit and 32-bit control of Port SFRs

Value	Description
1	32-bit access selected
0	16-bit access selected

11.3.11. Interrupt Change Notification Enable Register

Name: CNEN0x
Offset: 0x3654, 0x3678, 0x369C, 0x36C0

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CNEN0x[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNEN0x[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CNEN0x[15:0] Interrupt Change Notification Enable bits

Value	Description
1	Interrupt-on-change (from the last read value) is enabled for I/O.
0	Interrupt-on-change is disabled for I/O.

11.3.12. Interrupt Change Notification Edge Select Register

Name: CNEN1x
Offset: 0x3658, 0x367C, 0x36A0, 0x36C4

Note: See pinout diagrams for I/O availability for a given device.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CNEN1x[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNEN1x[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CNEN1x[15:0] Interrupt Change Notification Edge Select bits

11.3.13. Peripheral Remapping Configuration Register

Name: RPCON
Offset: 0x32D0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					IOLOCK			
Access					R/W			
Reset					0			

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 11 – IOLOCK Peripheral Remapping Register Lock bit

Value	Description
1	All Peripheral Remapping registers are locked and cannot be written.
0	All Peripheral Remapping registers are unlocked and can be written.

11.3.14. Peripheral Pin Select Input Register 0

Name: RPINR0
Offset: 0x32D4

Bit	31	30	29	28	27	26	25	24
	INT3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INT2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INT1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – INT3R[7:0] Assign External Interrupt 3 (INT3) to the Corresponding RPn Pin bits

Bits 23:16 – INT2R[7:0] Assign External Interrupt 2 (INT2) to the Corresponding RPn Pin bits

Bits 15:8 – INT1R[7:0] Assign External Interrupt 1 (INT1) to the Corresponding RPn Pin bits

11.3.15. Peripheral Pin Select Input Register 1

Name: RPINR1
Offset: 0x32D8

Bit	31	30	29	28	27	26	25	24
	T3CKR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	T2CKR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	T1CKR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INT4R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – T3CKR[7:0] Assign Timer1 External Clock (T3CK) to the Corresponding RPn Pin bits

Bits 23:16 – T2CKR[7:0] Assign Timer1 External Clock (T2CK) to the Corresponding RPn Pin bits

Bits 15:8 – T1CKR[7:0] Assign Timer1 External Clock (T1CK) to the Corresponding RPn Pin bits

Bits 7:0 – INT4R[7:0] Assign External Interrupt 4 (INT4) to the Corresponding RPn Pin bits

11.3.16. Peripheral Pin Select Input Register 2

Name: RPINR2
Offset: 0x32DC

Bit	31	30	29	28	27	26	25	24
	ICM2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TCKI2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ICM1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TCKI1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – ICM2R[7:0] Assign SCCP Capture 2 (ICM2) Input to the Corresponding RPn Pin bits

Bits 23:16 – TCKI2R[7:0] Assign SCCP Timer2 (TCKI2) Input to the Corresponding RPn Pin bits

Bits 15:8 – ICM1R[7:0] Assign SCCP Capture 1 (ICM1) Input to the Corresponding RPn Pin bits

Bits 7:0 – TCKI1R[7:0] Assign SCCP Timer1 (TCKI1) Input to the Corresponding RPn Pin bits

11.3.17. Peripheral Pin Select Input Register 3

Name: RPINR3
Offset: 0x32E0

Bit	31	30	29	28	27	26	25	24
	ICM4R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TCKI4R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ICM3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TCKI3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – ICM4R[7:0] Assign SCCP Capture 4 (ICM4) Input to the Corresponding RPn Pin bits

Bits 23:16 – TCKI4R[7:0] Assign SCCP Timer4 (TCKI4) Input to the Corresponding RPn Pin bits

Bits 15:8 – ICM3R[7:0] Assign SCCP Capture 3 (ICM3) Input to the Corresponding RPn Pin bits

Bits 7:0 – TCKI3R[7:0] Assign SCCP Timer3 (TCKI3) Input to the Corresponding RPn Pin bits

11.3.18. Peripheral Pin Select Input Register 4

Name: RPINR4
Offset: 0x32E4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ICM5R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TCKI5R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – ICM5R[7:0] Assign SCCP Capture 5 (ICM5) Input to the Corresponding RPn Pin bits

Bits 7:0 – TCKI5R[7:0] Assign SCCP Timer5 (TCKI5) Input to the Corresponding RPn Pin bits

11.3.19. Peripheral Pin Select Input Register 7

Name: RPINR7
Offset: 0x32F0

Bit	31	30	29	28	27	26	25	24
	OCFDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OCFCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	OCFBR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	OCFAR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – OCFDR[7:0] Assign SCCP Fault D (OCFD) Input to the Corresponding RPn Pin bits

Bits 23:16 – OCFCR[7:0] Assign SCCP Fault C (OCFC) Input to the Corresponding RPn Pin bits

Bits 15:8 – OCFBR[7:0] Assign SCCP Fault B (OCFB) Input to the Corresponding RPn Pin bits

Bits 7:0 – OCFAR[7:0] Assign SCCP Fault A (OCFA) Input to the Corresponding RPn Pin bits

11.3.20. Peripheral Pin Select Input Register 8

Name: RPINR8
Offset: 0x32F4

Bit	31	30	29	28	27	26	25	24
	PCI11R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI10R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCI9R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCI8R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – PCI11R[7:0] Assign PWM Input 11 (PCI11) to the Corresponding RPn Pin bits

Bits 23:16 – PCI10R[7:0] Assign PWM Input 10 (PCI10) to the Corresponding RPn Pin bits

Bits 15:8 – PCI9R[7:0] Assign PWM Input 9 (PCI9) to the Corresponding RPn Pin bits

Bits 7:0 – PCI8R[7:0] Assign PWM Input 8 (PCI8) to the Corresponding RPn Pin bits

11.3.21. Peripheral Pin Select Input Register 9

Name: RPINR9
Offset: 0x32F8

Bit	31	30	29	28	27	26	25	24
	HOME1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INDX1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEB1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEA1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – HOME1R[7:0] Assign Home 1 Input (HOM1) to the Corresponding RPn Pin bits

Bits 23:16 – INDX1R[7:0] Assign Index 1 Input to the Corresponding RPn Pin bits

Bits 15:8 – QEB1R[7:0] Assign QEB 1 Input to the Corresponding RPn Pin bits

Bits 7:0 – QEA1R[7:0] Assign QEA 1 Input to the Corresponding RPn Pin bits

11.3.22. Peripheral Pin Select Input Register 13

Name: RPINR13
Offset: 0x3308

Bit	31	30	29	28	27	26	25	24
	U2DSRR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	U2RXR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	U1DSRR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U1RXR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – U2DSRR[7:0] Assign UART2 Data-Set-Ready ($\overline{\text{U2DSR}}$) to the Corresponding RPn Pin bits

Bits 23:16 – U2RXR[7:0] Assign UART2 Receive (U2RX) to the Corresponding RPn Pin bits

Bits 15:8 – U1DSRR[7:0] Assign UART1 Data-Set-Ready ($\overline{\text{U1DSR}}$) to the Corresponding RPn Pin bits

Bits 7:0 – U1RXR[7:0] Assign UART1 Receive (U1RX) to the Corresponding RPn Pin bits

11.3.23. Peripheral Pin Select Input Register 14

Name: RPINR14
Offset: 0x330C

Bit	31	30	29	28	27	26	25	24
	SCK1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SDI1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	U3DSRR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U3RXR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – SCK1R[7:0] Assign SPI1 Clock Input (SCK1IN) to the Corresponding RPn Pin bits

Bits 23:16 – SDI1R[7:0] Assign SPI1 Data Input (SDI1) to the Corresponding RPn Pin bits

Bits 15:8 – U3DSRR[7:0] Assign UART3 Data-Set-Ready ($\overline{\text{U3DSR}}$) to the Corresponding RPn Pin bits

Bits 7:0 – U3RXR[7:0] Assign UART3 Receive (U3RX) to the Corresponding RPn Pin bits

11.3.24. Peripheral Pin Select Input Register 15

Name: RPINR15
Offset: 0x3310

Bit	31	30	29	28	27	26	25	24
	SS2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SCK2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SDI2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SS1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – SS2R[7:0] Assign SPI2 Client Select ($\overline{SS2}$) to the Corresponding RPn Pin bits

Bits 23:16 – SCK2R[7:0] Assign SPI2 Clock Input (SCK2IN) to the Corresponding RPn Pin bits

Bits 15:8 – SDI2R[7:0] Assign SPI2 Data Input (SDI2) to the Corresponding RPn Pin bits

Bits 7:0 – SS1R[7:0] Assign SPI2 Client Select (SS1) to the Corresponding RPn Pin bits

11.3.25. Peripheral Pin Select Input Register 16

Name: RPINR16
Offset: 0x3314

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	SS3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	SCK3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	SDI3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – SS3R[7:0] Assign SPI3 Client Select (SS3) to the Corresponding RPn Pin bits

Bits 15:8 – SCK3R[7:0] Assign SPI3 Clock Input (SCK3IN) to the Corresponding RPn Pin bits

Bits 7:0 – SDI3R[7:0] Assign SPI3 Data Input (SDI3) to the Corresponding RPn Pin bits

11.3.26. Peripheral Pin Select Input Register 17

Name: RPINR17
Offset: 0x3318

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CAN1RXR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 23:16 – CAN1RXR[7:0] Assign CAN1 Input (CAN1RX) to the Corresponding RPn Pin bits

11.3.27. Peripheral Pin Select Input Register 18

Name: RPINR18
Offset: 0x331C

Bit	31	30	29	28	27	26	25	24
	REFI2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	REFI1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SENT2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SENT1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – REFI2R[7:0] Assign Reference clock input (REFI2) to the Corresponding RPn Pin bits

Bits 23:16 – REFI1R[7:0] Assign Reference clock input (REFI1) to the Corresponding RPn Pin bits

Bits 15:8 – SENT2R[7:0] Assign SENT2 Input (SENT2) to the Corresponding RPn Pin bits

Bits 7:0 – SENT1R[7:0] Assign SENT1 Input (SENT1) to the Corresponding RPn Pin bits

11.3.28. Peripheral Pin Select Input Register 19

Name: RPINR19
Offset: 0x3320

Bit	31	30	29	28	27	26	25	24
	PCI15R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI14R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCI13R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCI12R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – PCI15R[7:0] Assign PWM Input 15 (PCI15) to the Corresponding RPn Pin bits

Bits 23:16 – PCI14R[7:0] Assign PWM Input 14 (PCI14) to the Corresponding RPn Pin bits

Bits 15:8 – PCI13R[7:0] Assign PWM Input 13 (PCI13) to the Corresponding RPn Pin bits

Bits 7:0 – PCI12R[7:0] Assign PWM Input 12 (PCI12) to the Corresponding RPn Pin bits

11.3.29. Peripheral Pin Select Input Register 20

Name: RPINR20
Offset: 0x3324

Bit	31	30	29	28	27	26	25	24
	CLCINAR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PCI18R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCI17R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCI16R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – CLCINAR[7:0] Assign CLC Input A (CLCINA) to the Corresponding RPn Pin bits

Bits 23:16 – PCI18R[7:0] Assign PWM Input 18 (PCI18) to the Corresponding RPn Pin bits

Bits 15:8 – PCI17R[7:0] Assign PWM Input 17 (PCI17) to the Corresponding RPn Pin bits

Bits 7:0 – PCI16R[7:0] Assign PWM Input 16 (PCI16) to the Corresponding RPn Pin bits

11.3.30. Peripheral Pin Select Input Register 21

Name: RPINR21
Offset: 0x3328

Bit	31	30	29	28	27	26	25	24
	CLCINER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLCINDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CLCINCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLCINBR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – CLCINER[7:0] Assign CLC Input E (CLCINE) to the Corresponding RPn Pin bits

Bits 23:16 – CLCINDR[7:0] Assign CLC Input D (CLCIND) to the Corresponding RPn Pin bits

Bits 15:8 – CLCINCR[7:0] Assign CLC Input C (CLCINC) to the Corresponding RPn Pin bits

Bits 7:0 – CLCINBR[7:0] Assign CLC Input B (CLCINB) to the Corresponding RPn Pin bits

11.3.31. Peripheral Pin Select Input Register 22

Name: RPINR22
Offset: 0x332C

Bit	31	30	29	28	27	26	25	24
	CLCINIR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLCINHR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CLCINGR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLCINFR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – CLCINIR[7:0] Assign CLC Input I (CLCINI) to the Corresponding RPn Pin bits

Bits 23:16 – CLCINHR[7:0] Assign CLC Input H (CLCINH) to the Corresponding RPn Pin bits

Bits 15:8 – CLCINGR[7:0] Assign CLC Input G (CLCING) to the Corresponding RPn Pin bits

Bits 7:0 – CLCINFR[7:0] Assign CLC Input F (CLCINF) to the Corresponding RPn Pin bits

11.3.32. Peripheral Pin Select Input Register 23

Name: RPINR23
Offset: 0x3330

Bit	31	30	29	28	27	26	25	24
	U2CTSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	U1CTSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ADTRG31R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLCINJR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – U2CTSR[7:0] Assign UART2 Clear-to-Send ($\overline{\text{U2CTS}}$) to the Corresponding RPn Pin bits

Bits 23:16 – U1CTSR[7:0] Assign UART1 Clear-to-Send ($\overline{\text{U1CTS}}$) to the Corresponding RPn Pin bits

Bits 15:8 – ADTRG31R[7:0] Assign ADC Trigger (ADTRG31) to the Corresponding RPn Pin bits

Bits 7:0 – CLCINJR[7:0] Assign CLC Input J (CLCINJ) to the Corresponding RPn Pin bits

11.3.33. Peripheral Pin Select Input Register 24

Name: RPINR24
Offset: 0x3334

Bit	31	30	29	28	27	26	25	24
	IOM0R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BISS1GSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BISS1SLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U3CTSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – IOM0R[7:0] Assign IO Monitor 0 to the Corresponding RPn Pin bits

Bits 23:16 – BISS1GSR[7:0] Assign BiSS1 Get Sense Input to the Corresponding RPn Pin bits

Bits 15:8 – BISS1SLR[7:0] Assign BiSS1 Return Input to the Corresponding RPn Pin bits

Bits 7:0 – U3CTSR[7:0] Assign UART3 Clear-to-Send ($\overline{\text{U3CTS}}$) to the Corresponding RPn Pin bits

11.3.34. Peripheral Pin Select Input Register 25

Name: RPINR25
Offset: 0x3338

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	IOM3R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IOM2R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	IOM1R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – IOM3R[7:0] Assign IO Monitor 3 to the Corresponding RPn Pin bits

Bits 15:8 – IOM2R[7:0] Assign IO Monitor 2 to the Corresponding RPn Pin bits

Bits 7:0 – IOM1R[7:0] Assign IO Monitor 1 to the Corresponding RPn Pin bits

11.3.35. Peripheral Pin Select Input Register 26

Name: RPINR26
Offset: 0x333C

Bit	31	30	29	28	27	26	25	24
	PCI19R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – PCI19R[7:0] Assign PWM Input 19 (PCI19) to the Corresponding RPn Pin bits

11.3.36. Peripheral Pin Select Input Register 27

Name: RPINR27
Offset: 0x3340

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	PCI22R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PCI21R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCI20R[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – PCI22R[7:0] Assign PWM Input 22 (PCI22) to the Corresponding RPn Pin bits

Bits 15:8 – PCI21R[7:0] Assign PWM Input 21 (PCI21) to the Corresponding RPn Pin bits

Bits 7:0 – PCI20R[7:0] Assign PWM Input 20 (PCI20) to the Corresponding RPn Pin bits

11.3.37. Peripheral Pin Select Input Register 28

Name: RPINR28
Offset: 0x3344

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	U4DSRR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	U4CTSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	U4RXR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – U4DSRR[7:0] Assign UART4 Data-Set-Ready (U4DSR) to the Corresponding RPn Pin bits

Bits 15:8 – U4CTSR[7:0] Assign UART4 Clear-to-Send (U4CTS) to the Corresponding RPn Pin bits

Bits 7:0 – U4RXR[7:0] Assign UART4 Receive (U4RX) to the Corresponding RPn Pin bits

11.3.38. Peripheral Pin Select Output Register 0

Name: RPOR0
Offset: 0x3350

Bit	31	30	29	28	27	26	25	24
		RP4R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP3R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP2R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP1R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP4R[6:0] Peripheral Output Function is Assigned to RP4 Output Pin bits

Bits 22:16 – RP3R[6:0] Peripheral Output Function is Assigned to RP3 Output Pin bits

Bits 14:8 – RP2R[6:0] Peripheral Output Function is Assigned to RP2 Output Pin bits

Bits 6:0 – RP1R[6:0] Peripheral Output Function is Assigned to RP1 Output Pin bits

11.3.39. Peripheral Pin Select Output Register 1

Name: RPOR1
Offset: 0x3354

Bit	31	30	29	28	27	26	25	24
		RP8R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP7R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP6R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP5R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP8R[6:0] Peripheral Output Function is Assigned to RP8 Output Pin bits

Bits 22:16 – RP7R[6:0] Peripheral Output Function is Assigned to RP7 Output Pin bits

Bits 14:8 – RP6R[6:0] Peripheral Output Function is Assigned to RP6 Output Pin bits

Bits 6:0 – RP5R[6:0] Peripheral Output Function is Assigned to RP5 Output Pin bits

11.3.40. Peripheral Pin Select Output Register 2

Name: RPOR2
Offset: 0x3358

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		RP11R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP10R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP9R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 22:16 – RP11R[6:0] Peripheral Output Function is Assigned to RP11 Output Pin bits

Bits 14:8 – RP10R[6:0] Peripheral Output Function is Assigned to RP10 Output Pin bits

Bits 6:0 – RP9R[6:0] Peripheral Output Function is Assigned to RP9 Output Pin bits

11.3.41. Peripheral Pin Select Output Register 4

Name: RPOR4
Offset: 0x3360

Bit	31	30	29	28	27	26	25	24
		RP20R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP19R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP18R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP17R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP20R[6:0] Peripheral Output Function is Assigned to RP20 Output Pin bits

Bits 22:16 – RP19R[6:0] Peripheral Output Function is Assigned to RP19 Output Pin bits

Bits 14:8 – RP18R[6:0] Peripheral Output Function is Assigned to RP18 Output Pin bits

Bits 6:0 – RP17R[6:0] Peripheral Output Function is Assigned to RP17 Output Pin bits

11.3.42. Peripheral Pin Select Output Register 5

Name: RPOR5
Offset: 0x3364

Bit	31	30	29	28	27	26	25	24
		RP24R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP23R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP22R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP21R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP24R[6:0] Peripheral Output Function is Assigned to RP24 Output Pin bits

Bits 22:16 – RP23R[6:0] Peripheral Output Function is Assigned to RP23 Output Pin bits

Bits 14:8 – RP22R[6:0] Peripheral Output Function is Assigned to RP22 Output Pin bits

Bits 6:0 – RP21R[6:0] Peripheral Output Function is Assigned to RP21 Output Pin bits

11.3.43. Peripheral Pin Select Output Register 6

Name: RPOR6
Offset: 0x3368

Bit	31	30	29	28	27	26	25	24
		RP28R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP27R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP26R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP25R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP28R[6:0] Peripheral Output Function is Assigned to RP28 Output Pin bits

Bits 22:16 – RP27R[6:0] Peripheral Output Function is Assigned to RP27 Output Pin bits

Bits 14:8 – RP26R[6:0] Peripheral Output Function is Assigned to RP26 Output Pin bits

Bits 6:0 – RP25R[6:0] Peripheral Output Function is Assigned to RP25 Output Pin bits

11.3.44. Peripheral Pin Select Output Register 7

Name: RPOR7
Offset: 0x336C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		RP30R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP29R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 14:8 – RP30R[6:0] Peripheral Output Function is Assigned to RP30 Output Pin bits

Bits 6:0 – RP29R[6:0] Peripheral Output Function is Assigned to RP29 Output Pin bits

11.3.45. Peripheral Pin Select Output Register 8

Name: RPOR8
Offset: 0x3370

Bit	31	30	29	28	27	26	25	24
		RP36R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP35R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP34R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP33R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP36R[6:0] Peripheral Output Function is Assigned to RP36 Output Pin bits

Bits 22:16 – RP35R[6:0] Peripheral Output Function is Assigned to RP35 Output Pin bits

Bits 14:8 – RP34R[6:0] Peripheral Output Function is Assigned to RP34 Output Pin bits

Bits 6:0 – RP33R[6:0] Peripheral Output Function is Assigned to RP33 Output Pin bits

11.3.46. Peripheral Pin Select Output Register 9

Name: RPOR9
Offset: 0x3374

Bit	31	30	29	28	27	26	25	24
		RP40R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP39R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP38R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP37R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP40R[6:0] Peripheral Output Function is Assigned to RP40 Output Pin bits

Bits 22:16 – RP39R[6:0] Peripheral Output Function is Assigned to RP39 Output Pin bits

Bits 14:8 – RP38R[6:0] Peripheral Output Function is Assigned to RP38 Output Pin bits

Bits 6:0 – RP37R[6:0] Peripheral Output Function is Assigned to RP37 Output Pin bits

11.3.47. Peripheral Pin Select Output Register 10

Name: RPOR10
Offset: 0x3378

Bit	31	30	29	28	27	26	25	24
		RP44R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP43R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP42R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP41R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP44R[6:0] Peripheral Output Function is Assigned to RP44 Output Pin bits

Bits 22:16 – RP43R[6:0] Peripheral Output Function is Assigned to RP43 Output Pin bits

Bits 14:8 – RP42R[6:0] Peripheral Output Function is Assigned to RP42 Output Pin bits

Bits 6:0 – RP41R[6:0] Peripheral Output Function is Assigned to RP41 Output Pin bits

11.3.48. Peripheral Pin Select Output Register 12

Name: RPOR12
Offset: 0x3380

Bit	31	30	29	28	27	26	25	24
		RP52R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP51R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP50R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP49R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP52R[6:0] Peripheral Output Function is Assigned to RP52 Output Pin bits

Bits 22:16 – RP51R[6:0] Peripheral Output Function is Assigned to RP51 Output Pin bits

Bits 14:8 – RP50R[6:0] Peripheral Output Function is Assigned to RP50 Output Pin bits

Bits 6:0 – RP49R[6:0] Peripheral Output Function is Assigned to RP49 Output Pin bits

11.3.49. Peripheral Pin Select Output Register 13

Name: RPOR13
Offset: 0x3384

Bit	31	30	29	28	27	26	25	24
		RP56R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP55R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP54R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP53R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP56R[6:0] Peripheral Output Function is Assigned to RP56 Output Pin bits

Bits 22:16 – RP55R[6:0] Peripheral Output Function is Assigned to RP55 Output Pin bits

Bits 14:8 – RP54R[6:0] Peripheral Output Function is Assigned to RP54 Output Pin bits

Bits 6:0 – RP53R[6:0] Peripheral Output Function is Assigned to RP53 Output Pin bits

11.3.50. Peripheral Pin Select Output Register 14

Name: RPOR14
Offset: 0x3388

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
		RP57R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 6:0 – RP57R[6:0] Peripheral Output Function is Assigned to RP57 Output Pin bits

11.3.51. Peripheral Pin Select Output Register 32

Name: RPOR32
Offset: 0x33D0

Note:
1. These are virtual output ports.

Bit	31	30	29	28	27	26	25	24
		RP132R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP131R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP130R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP129R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP132R[6:0] Peripheral Output Function is Assigned to RP132 Output Pin bits

Bits 22:16 – RP131R[6:0] Peripheral Output Function is Assigned to RP131 Output Pin bits

Bits 14:8 – RP130R[6:0] Peripheral Output Function is Assigned to RP130 Output Pin bits

Bits 6:0 – RP129R[6:0] Peripheral Output Function is Assigned to RP129 Output Pin bits

11.3.52. Peripheral Pin Select Output Register 33

Name: RPOR33
Offset: 0x33D4

Note:
1. These are virtual output ports.

Bit	31	30	29	28	27	26	25	24
		RP136R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP135R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP134R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP133R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP136R[6:0] Peripheral Output Function is Assigned to RP136 Output Pin bits

Bits 22:16 – RP135R[6:0] Peripheral Output Function is Assigned to RP135 Output Pin bits

Bits 14:8 – RP134R[6:0] Peripheral Output Function is Assigned to RP134 Output Pin bits

Bits 6:0 – RP133R[6:0] Peripheral Output Function is Assigned to RP129 Output Pin bits

11.3.53. Peripheral Pin Select Output Register 34

Name: RPOR34
Offset: 0x33D8

Note:
1. These are virtual output ports.

Bit	31	30	29	28	27	26	25	24
		RP140R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP139R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP138R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP137R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP140R[6:0] Peripheral Output Function is Assigned to RP140 Output Pin bits

Bits 22:16 – RP139R[6:0] Peripheral Output Function is Assigned to RP139 Output Pin bits

Bits 14:8 – RP138R[6:0] Peripheral Output Function is Assigned to RP138 Output Pin bits

Bits 6:0 – RP137R[6:0] Peripheral Output Function is Assigned to RP137 Output Pin bits

11.3.54. Peripheral Pin Select Output Register 35

Name: RPOR35
Offset: 0x33DC

Note:
1. These are virtual output ports.

Bit	31	30	29	28	27	26	25	24
		RP144R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		RP143R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		RP142R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		RP141R[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:24 – RP144R[6:0] Peripheral Output Function is Assigned to RP144 Output Pin bits

Bits 22:16 – RP143R[6:0] Peripheral Output Function is Assigned to RP143 Output Pin bits

Bits 14:8 – RP142R[6:0] Peripheral Output Function is Assigned to RP142 Output Pin bits

Bits 6:0 – RP141R[6:0] Peripheral Output Function is Assigned to RP141 Output Pin bits

11.3.55. IOIM x Control Register

Name: IOIMxCON
Offset: 0x1ED0, 0x1EDC, 0x1EE8, 0x1EF4, 0x1F00, 0x1F0C, 0x1F18, 0x1F24

Bit	31	30	29	28	27	26	25	24
					FLTINJ	OKINJ	ATEST[1:0]	
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EOVFV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	ON		SLPEN	SIDL		EXTCLK		
Access	R/W		R/W	R/W		R/W		
Reset	0		0	0		0		

Bit	7	6	5	4	3	2	1	0
	FBKSEL[3:0]				REFSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 27 – FLTINJ Fault Injection bit

A write of '1' to this bit will simulate a comparison mismatch event. The ERR bit will become set, the ERRCNT will increment to EOVFV and the OVF will become set if the ERRCNT overflows.

Bit 26 – OKINJ OK Inject bit

A write of '1' to this bit will simulate a transition on the reference signal input, blanking time insertion and a good comparison between reference and feedback signals. The OK bit will become set.

Bits 25:24 – ATEST[1:0] Artificial Test Enable bit

Value	Description
11	Artificial OKINJ Test is enabled.
10	Artificial FLTINJ Test is enabled.
01	Artificial Test is disabled.
00	Artificial Test is disabled.

Bits 23:16 – EOVFV[7:0] Error-Counter Overflow Value bits

Bit 15 – ON Module Enable bit

Value	Description
1	IOIM module is enabled.
0	IOIM module is disabled.

Bit 13 – SLPEN Module Sleep Enable bit

Value	Description
1	Module operates in Sleep mode.
0	Module disabled in Sleep mode.

Bit 12 – SIDL Module Stop in Idle Mode Enable bit

Value	Description
1	Module disabled in Idle mode.
0	Module operates in Idle mode.

Bit 10 – EXTCLK External Clock Source Enable bit

Value	Description
1	CLKGEN13 (200 MHz)
0	T_{CY} (Instruction Cycle) (default)

Bits 7:4 – FBKSEL[3:0] Feedback Input Mux Selection bits
For IOIM Group A, see Table 11-14.

Value	Description
1111	Feedback input [15] is selected.
. . .	
0001	Feedback input [1] is selected.
0000	Feedback input [0] is selected.

Bits 3:0 – REFSEL[3:0] Reference Input Mux Selection bits

Value	Description
1111	Reference input [15] is selected.
. . .	
0001	Reference input [1] is selected.
0000	Reference input [0] is selected.

11.3.56. IOIM x Status Register

Name: IOIMxSTAT
Offset: 0x1ED8, 0x1EE4, 0x1EF0, 0x1EFC, 0x1F08, 0x1F14, 0x1F20, 0x1F2C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ERRCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FFEDGE	FREDGE	RFEDGE	RREDGE		OVF	ERR	OK
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bits 15:8 – ERRCNT[7:0] Error Counter bits
Indicates the number of mismatches between the reference and feedback inputs after the blanking time has expired.

Bit 7 – FFEDGE Feedback Falling Edge Status bit

Value	Description
1	At least one fall transition has been detected on the feedback input.
0	No reference fall edge has occurred.

Bit 6 – FREDGE Feedback Rising Edge Status bit

Value	Description
1	At least one rise transition has been detected.
0	No reference rise edge has occurred on the feedback input.

Bit 5 – RFEDGE Reference Falling Edge Status bit

Value	Description
1	At least one fall transition has been detected on the reference input.
0	No reference fall transition has occurred.

Bit 4 – RREDGE Reference Rise Edge Status bit

Value	Description
1	At least one rise transition has been detected on the reference input.
0	No reference fall transition has occurred.

Bit 2 – OVF Overflow bit

Value	Description
1	Error counter has overflowed since the last time it was cleared.
0	Error counter has not overflowed.

Bit 1 – ERR Error bit

Value	Description
1	At least one mismatch has occurred between the reference and feedback inputs.
0	No mismatches have occurred.

Bit 0 – OK OK bit

Value	Description
1	At least one transition has been detected on the reference input with no mismatch.
0	No transitions detected on the reference input.

11.3.57. IOIM x Blanking Time Register

Name: IOIMxBCON
Offset: 0x1ED4, 0x1EE0, 0x1EEC, 0x1EF8, 0x1F04, 0x1F10, 0x1F1C, 0x1F28

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BLANK[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BLANK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – BLANK[15:0] Blanking Time Register bits
The 16-bit value is to be the banking time value.

11.4. Operation

11.4.1. I/O Port Control

Before reading and writing any I/O port, the desired pin or pins should be properly configured for the application. Each I/O port has nine registers directly associated with the operation of the port and one control register. Each I/O port pin has a corresponding bit in these registers. Throughout this section, the letter 'x' denotes any or all port module instances. For example, TRISx would represent TRISA, TRISB, TRISC and so on. Any bit and its associated data and control registers that are not valid for a particular device will be disabled and will read as zeros.

The TRISx registers configure the data direction flow through port I/O pins. The TRISx register bits determine whether a PORTx I/O pin is an input or an output:

- If a data direction bit is '1', the corresponding I/O port pin is an input.
- If a data direction bit is '0', the corresponding I/O port pin is an output.

A read from a TRISx register reads the last value written to that register. All I/O port pins are defined as inputs after a Power-on Reset (POR).

Note: It is recommended to make the pin an output and drive to zero (TRISx = 0,LATx = 0) prior to making the I/O pin an input (TRISx = 1); this will help in discharging the parasitic capacitance internal to the I/O pin.

The PORTx registers allow I/O pins to be accessed; a write to a PORTx register writes to the corresponding LATx register (PORTx data latch). The I/O port pin(s) configured as outputs are updated. A write to a PORTx register is effectively the same as a write to a LATx register. A read from a PORTx register reads the synchronized signal applied to the port I/O pins.

The LATx registers hold data written to port I/O pins; a write to a LATx register latches data to the corresponding port I/O pins. The I/O port pins configured as outputs are updated. A read from a LATx register reads the data held in the PORTx data latch, not from the port I/O pins.

11.4.2. Open-Drain Configuration (ODCx)

Each I/O pin can be individually configured for either normal digital output or open-drain output. This is controlled by the Open-Drain Control register, ODCx, associated with each I/O pin. If the ODCx bit for an I/O pin is a '1', the pin acts as an open-drain output. If the ODCx bit for an I/O pin is a '0', the pin is configured for a normal digital output (the ODCx bit is valid only for output pins). After a Reset, the status of all the bits of the ODCx register is set to '0'.

The open-drain feature allows the generation of outputs higher than Vdd (e.g., 5V) on any desired 5V tolerant pins by using external pull-up resistors. The maximum open-drain voltage allowed is the same as the maximum Vih specification. The ODCx register setting takes effect in all of the I/O modes, allowing the output to behave as an open-drain, even if a peripheral is controlling the pin. Although the user could achieve the same effect by manipulating the corresponding LATx and TRISx bits, this procedure will not allow the peripheral to operate in Open-Drain mode (except for the default operation of the I^2C pins). Since I^2C pins are already open-drain pins, the ODCx settings do not affect the I^2C pins. Also, the ODCx settings do not affect the JTAG output characteristics as the JTAG scan cells are inserted between the ODCx logic and the I/O.

11.4.3. Configuring Analog and Digital Port Pins (ANSELx)

The ANSELx register controls the operation of the analog port pins. The port pins that are to function as analog inputs must have their corresponding ANSELx and TRISx bits set. To use port pins for I/O functionality with digital modules, such as timers, UARTs, etc., the corresponding ANSELx bit must be cleared.

The ANSELx register has a default value of 0xFFFF; therefore, all pins that share analog functions are, by default, analog and *not* digital.

If the TRISx bit is cleared (output) while the ANSELx bit is set, the digital output level (V_{OH} or V_{OL}) is converted by an analog peripheral, such as the ADC module or the comparator module.

When the PORTx register is read, all pins configured as analog input channels are read as cleared (a low level).

Pins configured as digital inputs do not convert an analog input. Analog levels on any pin defined as a digital input (including the ANx pins) can cause the input buffer to consume current that exceeds the device specifications.

Table 11-16. ANSEL Pin States

ANSELx Bit	TRISx Bit	Pin State
1	0	The output is from the analog module.
1	1	The port always reads as low.
0	0	Digital Output
0	1	Digital Input

11.4.4. Peripheral Pin Select (PPS)

The Peripheral Pin Select Input registers, RPINRx, and the Peripheral Pin Select Output registers, RPORx, provide control for PPS input and output mapping. See Input Mapping and Output Mapping for detailed information on configuring these registers.

11.4.4.1. Available Pins

The number of available pins is dependent on the particular device and its pin count. Pins that support the Peripheral Pin Select feature include the designation, RPn, in their full pin designation, where RP designates a remappable peripheral and n is the remappable pin number. Note that on a device's schematic symbol, remappable input pins are designated as RPIn, and remappable output pins are designated as RPn.

11.4.4.2. Available Peripherals

The peripherals managed by the Peripheral Pin Select are all digital-only peripherals. These include general serial communications (UART and SPI), general purpose timer clock inputs, timer-related peripherals (input capture and output compare) and interrupt-on-change inputs.

In comparison, some digital-only peripheral modules are never included in the Peripheral Pin Select feature. This is because the peripheral's function requires special I/O circuitry on a specific port and cannot be easily connected to multiple pins. These modules include I^{2}C and the motor control PWM. A similar requirement excludes all modules with analog inputs, such as an A/D Converter.

A key difference between remappable and non-remappable peripherals is that remappable peripherals are not associated with a default I/O pin. The peripheral must always be assigned to a specific I/O pin before it can be used. In contrast, non-remappable peripherals are always available on a default pin, assuming that the peripheral is active and not conflicting with another peripheral.

When a remappable peripheral is active on a given I/O pin, it takes priority over all other digital I/Os and digital communication peripherals associated with the pin. Priority is given regardless of the type of peripheral that is mapped. Remappable peripherals never take priority over any analog functions associated with the pin.

11.4.4.3. Input Mapping

The inputs of the Peripheral Pin Select options are mapped based on the peripheral; that is, a control register associated with a peripheral dictates the pin it will be mapped to. The RPINRx registers are used to configure peripheral input mapping. Each register contains sets of 8-bit fields, with each set associated with one of the remappable peripherals. Programming a given peripheral's bit field with an appropriate 8-bit value maps the RPn pin with the corresponding value to that peripheral.

Example: Interrupt 1 input (from Table 11-10) is mapped to PWM Event 1 (from Table 11-11)

_INT1R[7:0] = 139; // 139 = PWM EVENT 1

11.4.4.4. Output Mapping

In contrast to inputs, the outputs of the PPS options are mapped based on the pin. In this case, a control register associated with a particular pin dictates the peripheral output to be mapped. The RPORx registers are used to control output mapping. Each register contains sets of six-bit fields, with each set associated with one RPn pin. The value of the bit field corresponds to one of the peripherals, and that peripheral's output is mapped to the pin (see Table 11-13).

A null output is associated with the Output Register Reset value of '0'. This is done to ensure that, by default, remappable outputs remain disconnected from all output pins.

11.4.4.5. Mapping Limitations

The control scheme of the peripheral select pins is not limited to a small range of fixed peripheral configurations. There are no mutual or hardware enforced lockouts between any of the peripheral mapping SFRs. Any combination of peripheral mappings across any or all of the RPn pins is possible. This includes both many-to-one and one-to-many mappings of peripheral inputs and outputs to pins.

11.4.5. Controlling Configuration Changes

Because peripheral mapping can be changed during run time, some restrictions on peripheral remapping are needed to prevent accidental configuration changes. All devices include a control register lock bit to prevent alterations to the peripheral map.

Under a normal operation, writes to the RPINRx and RPORx registers are not allowed. Attempted writes will appear to execute normally, but the contents of the registers will remain unchanged. To change these registers, they must be unlocked in hardware. The register lock is controlled by the IOLOCK bit (RPCON[11]). Setting IOLOCK prevents writes to the control registers; clearing IOLOCK allows writes.

11.4.6. Considerations for Peripheral Pin Selection

The ability to control Peripheral Pin Selection introduces several considerations into application design. Peripheral Pin Selects are not available on default pins in the device's default (Reset) state. More specifically, because all RPINRx registers reset to '1's and RPORx registers reset to '0's, this means all PPS inputs are tied to Vss, while all PPS outputs are disconnected. This means that before any other application code is executed, the user must initialize the device with the proper peripheral configuration. Because the IOLOCK bit resets in the unlocked state, it is not necessary to execute the unlock sequence after the device has come out of Reset. For application safety, however, it is always better to set IOLOCK and lock the configuration after writing to the control registers.

Choosing the configuration requires a review of all Peripheral Pin Selects and their pin assignments, particularly those that will not be used in the application. In all cases, unused pin-selectable peripherals should be completely disabled. The RPn functions will be cleared by default.

The assignment of a peripheral to a pin does not perform any other configuration of the pin's I/O circuitry. This means that adding a pin-selectable output to a pin may inadvertently drive an existing peripheral input when the output is driven. Users must be familiar with the behavior of other fixed peripherals that share a remappable pin and know when to enable or disable them. To be safe, fixed digital peripherals that share the same pin should be disabled when not in use.

Along these lines, configuring a remappable pin for a specific peripheral does not automatically turn that feature on. The peripheral must be specifically configured for operation and enabled as if it were tied to a fixed pin. Where this happens in the application code (immediately following device Reset and peripheral configuration or inside the main application routine) depends on the peripheral and its use in the application.

A final consideration is that Peripheral Pin Select functions neither override analog inputs nor reconfigure pins with analog functions for digital I/Os. If a pin is configured as an analog input on device Reset, it must be explicitly reconfigured as a digital I/O when used with a Peripheral Pin Select.

Example 11-1 provides a configuration for bidirectional communication with flow control using UART1. The following input and output functions are used:

- Input Functions: U1RX, $\overline{\text{U1CTS}}$
- Output Functions: U1TX, $\overline{\text{U1RTS}}$

Example 11-1. Configuring UART1 Input and Output Functions

```
//Configure Input Functions(See Table 11-10)

// Assign U1Rx To Pin RP35
_U1RXR = 35;

// Assign U1CTS To Pin RP36
_U1CTSR = 36;

// Configure Output Functions
// Assign U1Tx To Pin RP37
_RP37 = 10;                        // 10 = U1TX (see Table 11-13)

// Assign U1RTS To Pin RP38
_RP38 = 11;                      // 11 = U1RTS (see Table 11-13)
```

11.4.7. Virtual Output Pins

The virtual pins (RPVn) enable the user to connect internal peripherals, whose signals may be of significant use to other peripherals, but these outputs may not be presented on a device pin.

The concept of "virtual pins" enables device features to be mapped to other peripherals that do not otherwise have a dedicated connection.

A simple example use is a Fault input to the PWM from the CLC, as shown in Example 11-2. The CLC output is mapped to a virtual output pin (RPV0), and the PWM input is mapped to the virtual pin. Virtual pins are associated with an input index from Table 11-11.

Example 11-2. Virtual PPS connection

```
// Map CLC output to virtual pin RPV0 (RP166)
_RP166R = 52; // 52 = CLC2OUT

// Map Input PWM PCI 8 to virtual output pin RP166
_PCI8R = 166
```

11.4.8. Peripheral Multiplexing

When a peripheral is enabled, the associated pin output drivers are typically module controlled, while a few are user-settable. The I/O pin may be read through the input data path, but the output driver for the I/O port bit is generally disabled.

Note: Some ports are shared with analog module pins. The corresponding bits in the ANSELx registers, if present, must be set to '0' for I/O port functionality.

11.4.9. I/O Multiplexing with Multiple Peripherals

For some devices, particularly those with a small number of I/O pins, multiple peripheral functions may be multiplexed on each I/O pin. The name of the I/O pin defines the priority of each function associated with the pin.

The conceptual I/O pin has two multiplexed peripherals, Peripheral A and Peripheral B, and is named PERA/PERB/PIO.

The I/O pin name is chosen because the user-assigned application can easily determine the priority of the functions assigned to the pin. Peripheral A has the highest priority for control of the pin. If Peripheral A and Peripheral B are enabled at the same time, Peripheral A will take control of the I/O pin.

11.4.9.1. Software Input Pin Control

Some of the functions assigned to an I/O pin may be input functions that do not take control of the pin output driver. An example of one such peripheral is the input capture module. If the I/O pin associated with the input capture is configured as an output, using the appropriate TRISx control bit, the user can manually affect the state of the input capture pin through its corresponding PORTx register. This behavior can be useful in some situations, especially for testing purposes, when no external signal is connected to the input pin.

The organization of the peripheral multiplexers will determine if the peripheral input pin can be manipulated in software using the PORTx register.

In general, the following peripherals allow their input pins to be controlled manually through the PORTx registers:

- External Interrupt Pins
- Timer Clock Input Pins
- Input Capture Pins

- PWM Fault Pins

Most serial communication peripherals, when enabled, take control of the I/O pin so that the input pins associated with the peripheral cannot be affected through the corresponding PORTx registers. Example peripherals include the following:

- SPI
- I^2C
- DCI
- UART
- CAN FD
- QEI

11.4.9.2. Pin Control Summary

When a peripheral is enabled, the associated pin output drivers are typically module controlled, while a few are user-settable. The term, module control, means that the associated port pin output driver is disabled, and the pin can only be controlled and accessed by the peripheral. The term, user settable, means that the associated peripheral port pin output driver is user-configurable in software through the associated TRISx Special Function register (SFR). The TRISx register must be set for the peripheral to function properly. For user-settable peripheral pins, the actual port pin state can always be read through the PORTx SFR.

An input capture peripheral provides an example of a user-settable peripheral. The user application must write the associated TRISx register to configure the input capture pin as an input. Because the I/O pin circuitry is still active when the input capture is enabled, the following method can be used to manually produce capture events using software:

- The input capture pin is configured as an output using the associated TRISx register.
- Then, the software can write values to the corresponding LATx register drive to internally control the input capture pin and force capture events.

As another example, an INTx pin can be configured as an output, and then by writing to the associated LATx bit, an INTx interrupt, if enabled, can be generated.

The UART is an example of a module control peripheral. When the UART is enabled, the PORTx and TRISx registers have no effect and cannot be used to read or write the RX and TX pins. Most communication peripheral functions available on the devices are module control peripherals.

For example, the SPI module can be configured for Host mode, in which only the SDO pin is required. In this scenario, the SDI pin can be configured as a general purpose output pin by clearing (setting to a logic '0') the associated TRISx bit. For more information on how pins can be configured for a module, refer to the specific module section.

11.4.9.3. Multiplexing Digital Input Peripheral

The following conditions are characteristic of a multiplexed digital input peripheral:

- The peripheral does not control the TRISx register. Some peripherals require the pin be configured as an input by setting the corresponding TRISx bit = 1.
- The peripheral input path is independent of the I/O input path and uses an input buffer that is dependent on the peripheral.
- The PORTx register data input path is not affected and is able to read the pin value.

11.4.9.4. Multiplexing Digital Output Peripheral

The following conditions are characteristic of a multiplexed digital output peripheral:

- The peripheral controls the output data. Some peripherals require the pin be configured as an output by setting the corresponding TRISx bit = 0.

- If a peripheral pin has an automatic tri-state feature (e.g., PWM outputs), the peripheral has the ability to tri-state the pin.
- The pin output driver type could be affected by the peripheral (e.g., drive strength, slew rate, etc.).
- PORTx register output data have no effect.

11.4.9.5. Multiplexing Digital Bidirectional Peripheral

The following conditions are characteristic of a multiplexed digital bidirectional peripheral:

- The peripheral automatically configures the pin as an output but not as an input. Some peripherals require the pin be configured as an input by setting the corresponding TRISx bit = `1`.
- Peripherals control the output data.
- The pin output driver type could be affected by the peripheral (e.g., drive strength, slew rate, etc.).
- The PORTx register data input path is not affected and is able to read the pin value.
- PORTx register output data have no effect.

11.4.9.6. Multiplexing Analog Input Peripheral

The following condition is characteristic of a multiplexed analog input peripheral:

- All digital port input buffers are disabled and PORTx registers read '`0`' to prevent a crowbar current.

11.4.9.7. Multiplexing Analog Output Peripheral

The following conditions are characteristic of a multiplexed analog output peripheral:

- All digital port input buffers are disabled and PORTx registers read '`0`' to prevent a crowbar current.
- Analog output is driven onto the pin independent of the associated TRISx setting.

Note: To use pins that are multiplexed with the ADC module for digital I/Os, the corresponding bits in the ANSELx register, if present, must be set to '`0`', even if the ADC module is turned off.

11.4.10. Change Notice (CN)

The Change Notification pins provide devices with the ability to generate interrupt requests to the processor in response to a Change-of-State (COS) on selected input pins (corresponding TRISx bits must be = `1`).

The Change Notice (CN) module supports two detection modes: mismatch-based detection and edge-based detection. In mismatch mode, CN events are generated when the current pin level differs from the last PORTx value read by the software. In edge-detect mode, CN events are generated on configured rising and/or falling edges and are latched independently of PORTx reads.

11.4.10.1. CN Configuration and Operation

The CN pins are configured as follows:

1. Disable CPU interrupts.
2. Set the desired CN I/O pin as an input by setting the corresponding TRISx register bits to `1`.

   Note: If the I/O pin is shared with an analog peripheral, it may be necessary to configure this pin as a digital input.
3. Enable the CN Module by setting the ON bit (CNCONx[15]) = `1`.
4. Enable individual CN input pins; enable optional pull-ups or pull-downs.
5. Clear any pending CN interrupt conditions before enabling interrupts.

6. Configure the CNx Interrupt Priority bits, CNxIP[2:0].
7. Clear the CNx Interrupt Flag bit by setting the CNxIF bit (IFSx register) = 0.
8. Configure the CNx pin interrupt for either Mismatch mode or Edge Detect mode using the CNSTYLE bit (CNCONx[11]):
   - If Mismatch mode (CNSTYLE = 0) is selected, enable the individual CN pins using the CNEN0x bits. The CNEN1x bits are ignored in this mode.
   - If Edge Detect mode (CNSTYLE = 1) is selected, use the CNEN0x bits to enable positive edge detection and the CNEN1x bits to enable negative edge detection.
9. Enable the CNx Interrupt Enable bit by setting the CNxIE bit (IECx register) = 1.
10. Enable CPU interrupts.

    In Mismatch mode (CNSTYLE = 0), the CNSTATx registers reflect a live mismatch condition between the current pin state and the value sampled during the last read of the corresponding PORTx register. A mismatch condition is cleared when the PORTx register is read.

    In Edge Detect mode (CNSTYLE = 1), the CNFx registers latch valid edge-detect events on the corresponding pins. CNFx bits must be cleared by software to enable detection of subsequent edge transitions. In this mode, CNSTATx registers are not used and always read as '0'.

    In Edge Detect mode, CN interrupts can be configured to occur on rising edges, falling edges, or both, depending on the CNENx configuration. Refer to the Electrical Characteristics section to learn more.

    Note: In Edge Detect mode, clearing the CNxIF interrupt flag without clearing all asserted CNFx bits will cause the CN interrupt to reassert immediately. CNFx bits represent the actual interrupt source.

    Use Mismatch mode (CNSTYLE = 0) when only final pin state changes are of interest and reduced interrupt frequency is desired. Use Edge Detect mode (CNSTYLE = 1) when individual signal transitions must not be missed. When using Edge Detect mode, clear CNFx bits in the ISR and consider enabling only a single edge to limit the interrupt rate.

11.4.11. I/O Integrity Module (IOIM)

The I/O Integrity Monitor (IOIM) is typically used to monitor the integrity of a signal generated by any dsPIC/MCU. However, it can also be used to monitor any signal generated outside of the dsPIC/MCU by connecting the reference input and feedback inputs to the appropriate external monitoring points.

The purpose of the IOIM in dsPIC33A is to provide and meet safety component requirements for applications that require monitoring the connectivity of a selected reference signal against a feedback signal, along with the user configurable path delays. A typical application for the I/O Integrity Monitor is to compare the data supplied to an output pin with the data read from the output pin driver. The output pin driver data are supplied to the monitor via an input buffer. If desired, a second pin can be used to monitor the output pin data at a remote location, somewhere on the PCB or far from the chip.

Typical uses include:

- To increase protection for customers who are building safety-sensitive applications.
- To simplify structural tests for safety applications.
- To ensure the pin has been connected correctly in its final environment by using structural tests.

The IOIM detects a change in the reference signal to start the blanking timer, which is used to account for the feedback path delay. After a programmable amount of time, the state of the selected feedback input is compared with the state of the reference input. If a mismatch is sampled at any time or a configured time after the blanking period, an error event is generated through an interrupt. Every error event is used to increment the user-defined error counter value; the IOIM gives an overflow status bit once the error counter reaches the Overflow state.

11.4.11.1. Enabling the Module

The module is enabled when IOIMxCON[ON] = `1`.

When IOIMxCON[ON] = `0`, the module is considered disabled and consumes minimum power.

11.4.11.2. Configuration of IOIMxCON

IOIMxCON selects the I/O monitor option and determines comparison pulse duration, like high pulse, low pulse or both low and high pulse period comparison. ON must be set for the IOIM to operate. All registers can be programmed while ON is clear. The I/O pin must be configured as a digital output for the I/O monitor to be checked, and the presence of the port is subject to package variant.

11.4.11.2.1. Clock Selection

The IOIMxCON.EXTCLK bit determines the clock source that operates the timebase and other module logic.

When EXTCLK = `0`, the module timebase and logic are clocked from the system instruction cycle clock, T_{CY}.

The external clock selection is dependent on the device definition. The external clock source may or may not be synchronous to the Peripheral Clock domain. Synchronization logic is required between the SFR register space (Peripheral Clock domain) and the module logic (I/O Monitor Clock domain) to allow the module to operate from high speed asynchronous clock sources.

11.4.11.2.2. Reference Input Selection

The feedback input is selected from a 16-input multiplexer as shown in Figure 11-5. The value for the reference pin will be taken only from the output of that particular selected pin. For example, if the selected pin is the pin out of PWM, the PWM out is taken as the reference input; this can also be manually updated by setting LAT of the selected pin to 0/1.

11.4.11.2.3. Feedback Input Selection

The feedback input is selected from a 16-input multiplexer as shown in Figure 11-5. The value for the feedback pin can be taken from both output and input of the particular pin. For example, the feedback can be input from an external value or can be manually updated by setting LAT of the selected pin to 0/1.

11.4.11.2.4. IOIM Groups

A grouping scheme is utilized to accommodate more IOIM modules. There are two groups: Group A, which only uses Group A pins, and Group B, which only uses Group B pins. Group A IOIM pins are represented by IOMADx/IOMAFx for reference and feedback, and Group B IOIM pins are represented by IOMBDx/IOMBFx for reference and feedback. Reference and feedback pins must be selected from the same group. The Table 11-14 correlates the two groups' reference and feedback selections with their associated pad connection.

11.4.11.2.5. Write Protection

For write protection, the assigned LOCK and WREN are register bits located in the PACCON1 register, which is responsible for all of the device’s peripheral access control.

11.4.11.2.6. Blanking Time

IOIMBTCON.BLANK sets the blanking time. The mismatch event is valid after the blanking timer has counted down to zero. The blanking time is introduced to account for I/O pad delays or associated routing path delays in the feedback input with respect to reference input. Blanking time is reloaded into the blanking timer (as shown in Example 11-3) every time the reference pin is toggled to high.

Example 11-3. IOIM Code

```
#include "xc.h"
```

```
int main(void)
{
    IOIM1CONbits.ON = 0;           // module is turned off.
    IOIM1STAT = 0;                 // clearing all status flags

    // both of the above steps clears the module status bits.

    IOIM1CONbits.TESTEN = 0x10;  // set 10 for FAULT injection mode and 11 for OK
injection mode.
    IOIM1BCONbits.BLANK = 100;   // set to make sure the required delay is given between
reference and feedback.

    // we dont need to select any any reference and feedback pins as it is internal

    IOIM1CONbits.EOVFVAL = 10;   // set err count overflow value

    _IOM1IF = 0; //clear interrupt flag
    _IOM1IE = 1; //enable interrupt

    IOIM1CONbits.FLTINJ = 0;     // make sure fault inject is clear
    //IOIM1CONbits.OKINJ = 0;    // make sure ok inject is clear

    IOIM1CONbits.ON = 1;         // module is turned on

    IOIM1CONbits.FLTINJ = 1;     // set to inject fault condition
//    IOIM1CONbits.OKINJ = 1;    // set to inject ok condition

    while(1);

    return 0;
}

void __attribute__((interrupt)) _IOIM1Interrupt(void)
{

    _IOM1IF = 0;                // clear interrupt flag
    IOIM1STATbits.ERR = 0;
}
```

11.4.11.2.7. Self-Test

The IOIMxCON SFR contains the FLTINJ bit and OKINJ bit, which allow user software to simulate a IOIM failure/OK. This feature allows the user to fully test Fault-handling software. The mode for artificial test must be set accordingly.

Table 11-17. IOIM Test Conditions

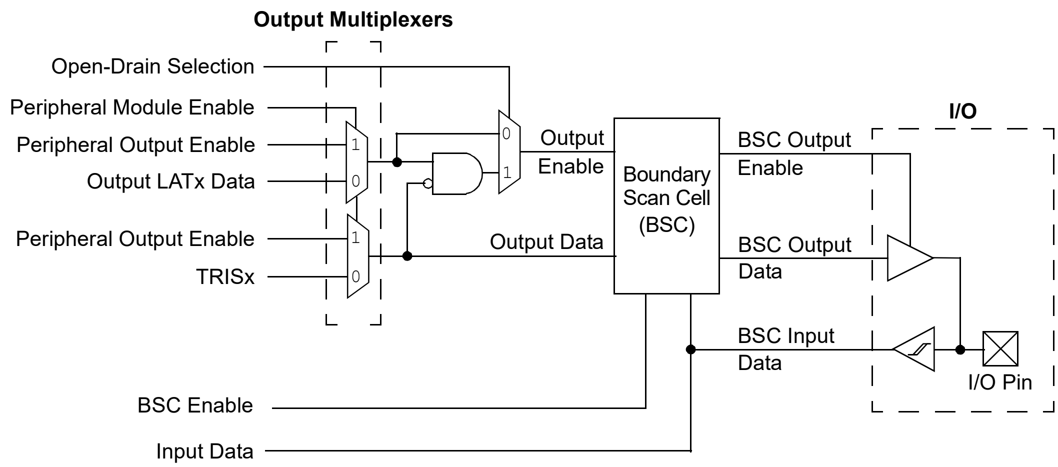
Operating Mode	ATEST Bit	OKINJ Bit	FLTINJ Bit	Register Bits Valid				Comments
				OK	ERR	ERRCNT	OVF	
Functional mode	00/01	x	x	Yes	Yes	Yes	Yes	Reference and feedback signals are taken from external signals.
	11	1	0	1	0	0	0	It is a module health test. Feedback and reference signals are injected from the register bit (OKINJ) by the user.
	10	0	1	0	1	Yes	Yes	It is a module health test. Feedback and reference signals are injected from the register bit (FLTINJ) by the user.

Notes:
1. It is not recommended to set the OKINJ and FLTINJ bits at the same time.
2. The OK bit is not set when FLTINJ = 1.
3. It is recommended to disable and then enable the module using the ON bit after mode changes.

11.4.12. Boundary Scan Cell Connections

The dsPIC33A family devices support a JTAG boundary scan. A Boundary Scan Cell (BSC) is inserted between the internal I/O logic circuit and the I/O pin, as shown in Figure 11-6. Most of the I/O pads have BSCs; however, JTAG pads do not. For a normal I/O operation, the BSC is disabled, and, therefore, is bypassed. The output enable input of the BSC are directly connected to the BSC output enable, and the output data input of the BSC are directly connected to the BSC output data. The pads that do not have BSC are the power supply pads (V_{DD}, V_{SS} and V_{CAP}/V_{CORE}) and the JTAG pads (TCK, TDI, TDO and TMS).

Figure 11-6. Boundary Scan Cell Connections

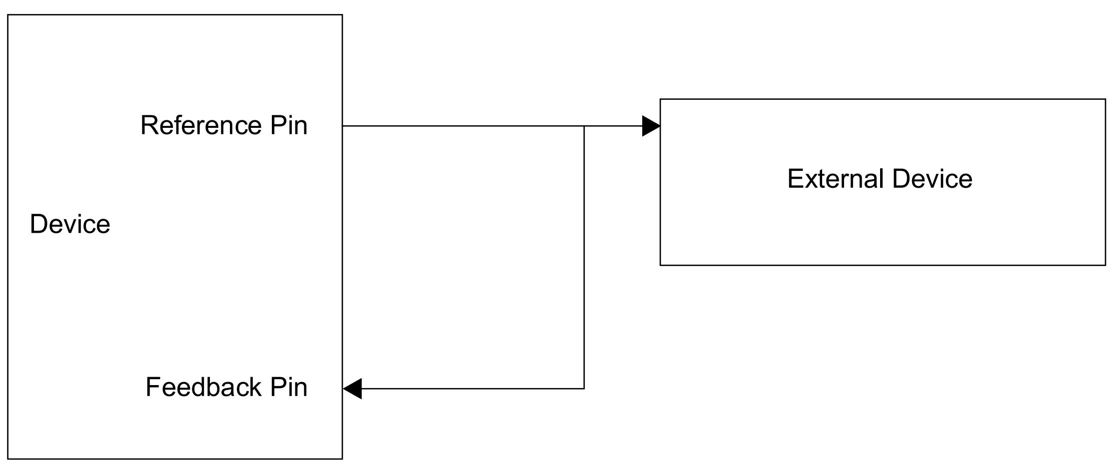


11.5. Application Example

11.5.1. Application Use of IOIM

The setup in the code example is used to showcase the general application of IOIM. The reference pin will be in output configuration, and feedback will be input. The delay between the signal leaving the reference pin and being received in the feedback pin is taken into consideration using the blanking timer.

Figure 11-7. General IOIM Application



Example 11-4.

```
#include "xc.h"

int main(void)
{
    IOIM1CONbits.ON = 0;                  // module is turned off.
    IOIM1STAT = 0;                        // clearing all status flags

    // both of the above steps clears the module status bits.

    IOIM1CONbits.TESTEN = 0x00;          // set 00/01 to disable artificial test.
    IOIM1BCONbits.BLANK = 100; // set to make sure the required delay is given between
reference and feedback.

    IOIM1CONbits.REFSEL = 8;             // this is according to datasheet (pin rc8)
    IOIM1CONbits.FBKSEL = 7;             // this is according to datasheet (pin rc7)
    IOIM1CONbits.EOVFVAL = 10;           // set err count overflow value

    _IOM1IF = 0;                         // clear interrupt flag
    _IOM1IE = 1;                         // enable interrupt

    _TRISC8 = 0;                         // set the reference pin to output
    _LATC8 = 0;                          // set the reference pin value

    IOIM1CONbits.ON = 1;                 // module is turned on

    while(1);

    return 0;
}

void __attribute__((interrupt)) _IOIM1Interrupt(void)
{

    _IOM1IF = 0;                         // clear interrupt flag
    IOIM1STATbits.ERR = 0;
    if(IOIM1STATbits.OVF)
    {
        IOIM1STATbits.OVF = 0;
        // your overflow application code goes here
    }
}
```

11.6. Interrupts

I/O Modules have change notification interrupts.

Interrupt Type	Condition	Flag
Change Notification	Whenever the data to the pin are in accordance with the Change Notification configuration, the interrupt is triggered. There is only one flag for each PORT, and the exact pin should be checked in the specified registers.	CNxIF

IO Integrity Module has error notification interrupt.

Interrupt Type	Condition	Flag
IOIM Interrupt	Whenever the ERR bit is set during the working of IOIM, the interrupt is triggered. There is one interrupt for every instance of the module.	IOIMxIF

11.7. Power-Saving Modes

11.7.1. I/O Port Operation in Sleep Mode

As the device enters Sleep mode, the system clock is disabled; however, the CN module continues to operate asynchronously. If one of the enabled CN pins changes state, the CNxIF bit (IFSx register) will be set. If the CNxIE bit (IECx register) is set, and its priority is greater than the current CPU priority, the device will wake from Sleep mode and execute the CN Interrupt Service Routine (ISR).

If the assigned priority level of the CN interrupt is less than or equal to the current CPU priority level, the CPU will not be awakened and the device will enter Idle mode.

11.7.2. I/O Port Operation in Idle Mode

As the device enters Idle mode, the system clock sources remain functional and the CN module continues to operate synchronously. If one of the enabled CN pins changes state, the CNxIF bit (IFSx register) will be set. If the CNxIE bit (IECx register) is set, and its priority is greater than the current CPU priority, the device will wake from Idle mode and execute the CN Interrupt Service Routine (ISR).

11.7.3. I/O Integrity Module Operations in Sleep Mode

When the device enters Sleep mode, the Peripheral Clock is disabled. The module can be selected to run with an external clock source. The Sleep Enable bit (IOIMxCON.SLPEN) selects whether the monitor function will stop in Sleep mode or continue to operate.

- If SLPEN = `1`, the module is enabled during Sleep and will continue to monitor I/Os.
- If SLPEN = `0`, the module is not enabled during Sleep and will not monitor I/Os.

Notes:

1. If the module is enabled during Sleep and requests an external clock source that can remain active in Sleep mode, the module will continue operation on that clock source in Sleep.
2. The monitor module must be clocked from a source available in Sleep for continued operation in Sleep mode.
3. An Interrupt event will wake the device from Sleep when the module interrupt is enabled.

11.7.4. I/O Integrity Module Operations in Idle Mode

When the device enters Idle mode, the peripheral clock sources remain functional and the CPU stops executing code. The Sleep in Idle bit (IOIMxCON.SIDL) selects whether the monitor function will stop in Idle mode or continue to operate in Idle mode.

- If IOIMxCON.SIDL = 0b0, the module will continue normal operation in Idle mode.
- If IOIMxCON.SIDL = 0b1, the module will stop when the device is in Idle mode. The module will perform the same procedures when stopped in Idle mode as for Sleep mode.

11.7.5. Interrupt Operation

This module generates a single interrupt on a mismatch event. The IOIMSTAT.ERR bits reflect the interrupt.

The mismatch/error is generated when the mismatch is greater than two IOIM clocks. The mismatch is generated on every mismatch event, and the OVF is set when the number of mismatches becomes equal to the defined overflow value; the OVF status bit doesn’t generate an interrupt. The Error Status bits reflection takes seven IOIM clock latencies, and it is a user responsibility to clear Status bits.

11.8. Effects of Various Resets

11.8.1. Device Reset

All I/O registers are forced to their Reset states upon a device Reset.

11.8.2. Power-on Reset

All I/O registers are forced to their Reset states upon a Power-on Reset (POR).

11.8.3. Watchdog Timer Reset

All I/O registers are unchanged upon a Watchdog Timer Reset.

12. Oscillator and Clocking Module

This section describes the Oscillator module in dsPIC33A devices. The Oscillator module supplies clocking to the system, including the CPU and peripherals. The following features are covered in this section:

- CLOCK Generator
- PLL Generator
- Clock Source such as FRC, Primary, Auxiliary or BFRC
- Fail Safe Clock Monitor
- REFO Clock Generator
- Power-Saving Mode

12.1. Device-Specific Information

Table 12-1. Oscillator Summary Table

Clock Generators	PLLs
16	2

Table 12-2. CLKGEN Assignment

Clock Generator	Input Clock Source	Module Clock Destination
Clock Generator 1	All sources	System clock and peripheral bus
Clock Generator 2	FRC	FRC
Clock Generator 3	BFRC	BFRC
Clock Generator 4	All sources	RAM BIST and NVM BIST
Clock Generator 5	All sources	PWM
Clock Generator 6	All sources	ADC
Clock Generator 7	All sources	PDM DAC
Clock Generator 8	All sources	UART
Clock Generator 9	All sources	SPI
Clock Generator 10	All sources	CAN
Clock Generator 11	All sources	PTG
Clock Generator 12	All sources	RDC
Clock Generator 13	All sources	CCP and REFO1
Clock Generator 14	All sources	CLC, IOIM and REFO2
Clock Generator 15	All sources	TRACE
Clock Generator 16	All sources	I3C and BiSS
Note: Clock generator 1/2/3/4 is always ON.		

Table 12-3. Clock Generator Clock Resources

Value	Description
`1111-1011`	Reserved
`1010`	REFI2 – User definable clock source
`1001`	REFI1 – User definable clock source
`1000`	PLL2 VCO output
`0111`	PLL1 VCO output
`0110`	PLL2 F_{OUT} output
`0101`	PLL1 F_{OUT} output

Table 12-3. Clock Generator Clock Resources (continued)

Value	Description
0100	BFRC/244 (32 KHz)
0011	POSC – Primary crystal oscillator (4-32 MHz)
0010	BFRC – Internal backup 8 MHz RC oscillator
0001	FRC – Internal 8 MHz RC oscillator (default)
0000	ICSP clock (PGC)

Table 12-4. PLL Clock Sources

Value	Description
1111-1011	Reserved
1010	REFI1 – User definable clock source
1001	REFI2 – User definable clock source
1000-0100	Reserved
0011	POSC – Primary crystal oscillator (4-32 MHz)
0010	BFRC – Internal backup 8 MHz RC oscillator
0001	FRC – Internal 8 MHz RC oscillator (default)
0000	ICSP clock (PGC)

Table 12-5. Clock Monitor Clock Resources

Value	Description
1111-1011	Reserved
1010	REFI1 – User definable clock source
1001	REFI2 – User definable clock source
1000	PLL2 VCO Div output
0111	PLL1 VCO Div output
0110	PLL2 F_{OUT} output
0101	PLL1 F_{OUT} output
0100	LPRC as BFRC/244
0011	POSC – Primary crystal oscillator (4-32 MHz)
0010	BFRC – Internal backup 8 MHz RC oscillator
0001	FRC – Internal 8 MHz RC oscillator
0000	ICSP clock (PGC)

12.2. Architectural Overview

The oscillator module is based on a number of clock generators (CLKGENs) and PLL generators (PLLGENs). The CLKGENs are preassigned to the CPU and peripherals. Some of the CLKGENs are shared by more than one consumer. Each clock generator selects a clock source from the available oscillators or PLLs and passes it to a selectable divider circuit. The CLKGENs utilize a change request mechanism to prevent unintentional modifications. After the settings are written, a change request bit is set, and when clear, the operation is complete. In the event of a clock failure, the backup clock source is then used by the CLKGEN.

The oscillator has the following main functions/modules:

- Multiple Clock Generators, each with:
  - Selectable clock source
  - A backup clock source
  - Fail-safe clock monitors
- Multiple PLLs, each with:

-
  - Selectable clock source
  - A backup clock source
  - PLL FOUT primary divider output
  - PLL VCO secondary divider output
  - Fail-safe clock monitors
- Clock Monitor Module that Compares a Clock Against a Reference Clock:
  - Clock failure detection
  - Clock frequency drift detection
  - Selectable threshold limits for warning and/or failing
  - Fault injection capability
  - Frequency, pulse width and duty cycle measurements

A high-level block diagram of the dsPIC33A oscillator system is shown in Figure 12-1.

Figure 12-1. Oscillator Module Block Diagram

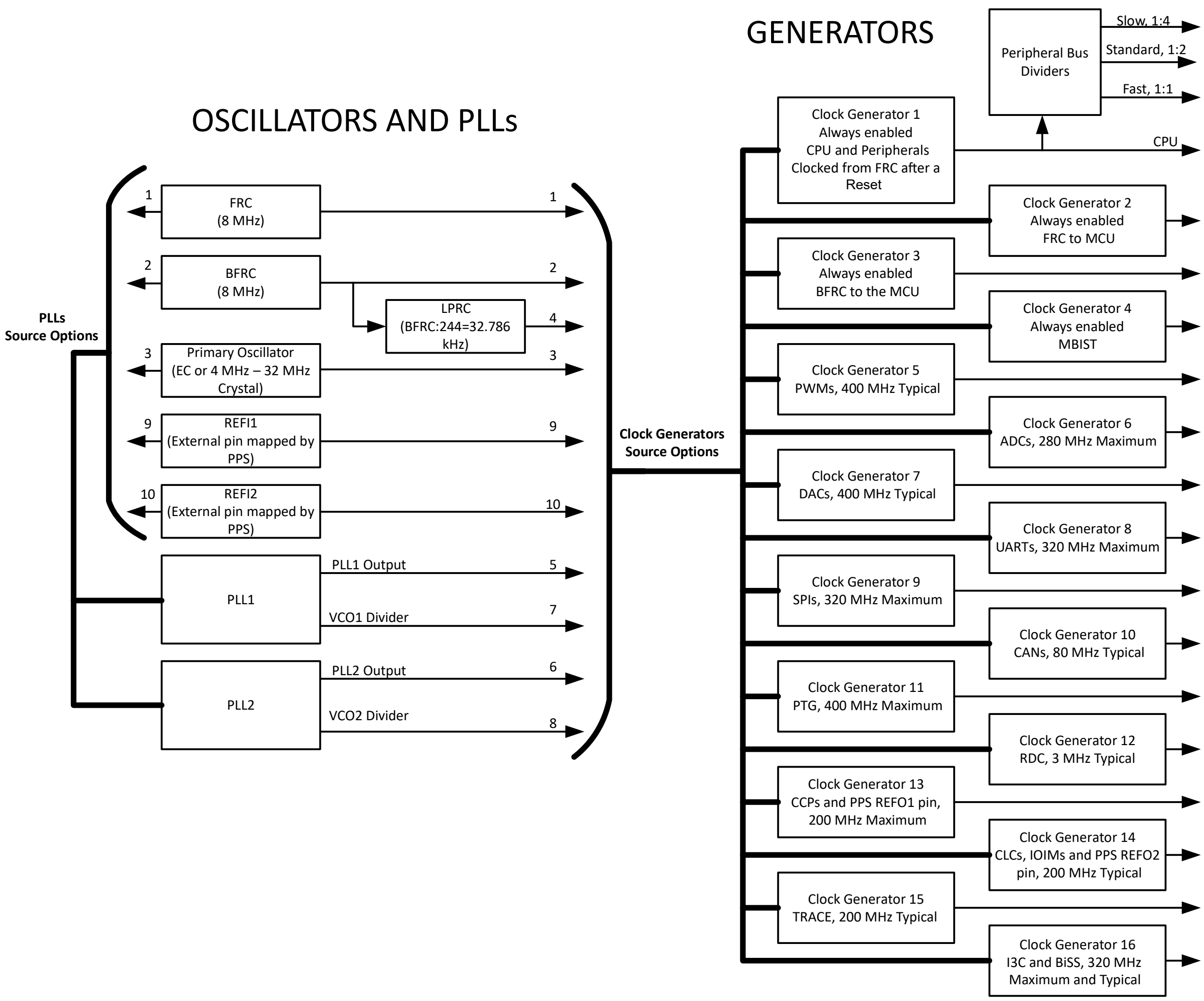
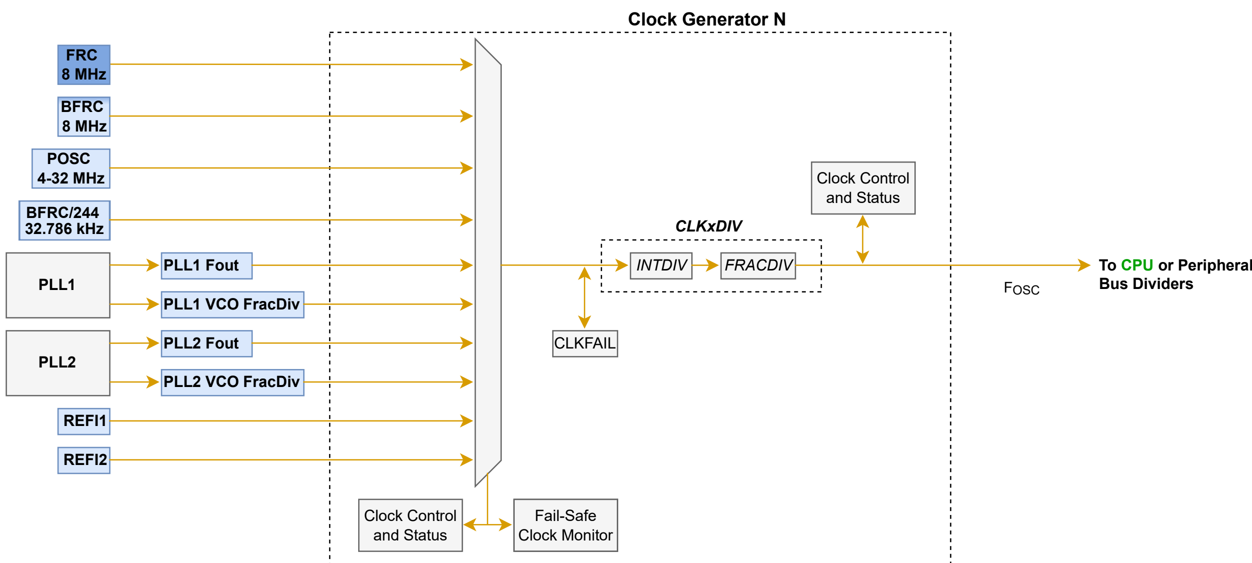


Figure 12-2. Clock Generator

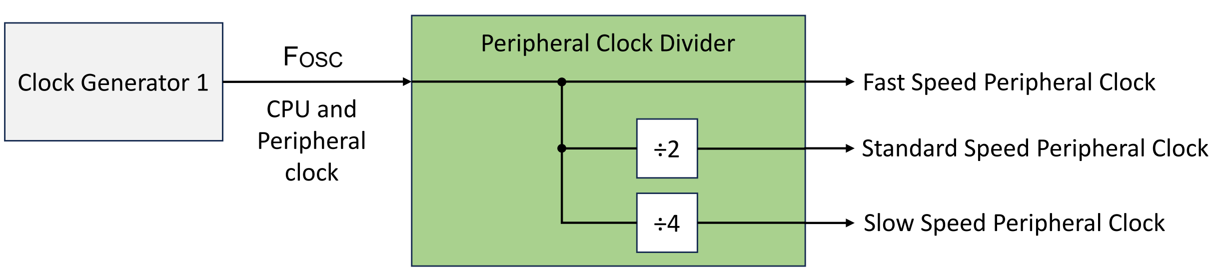


Note: F_{OSC} to CPU is called a CPU instruction clock (F_{CY}). F_{OSC} to Peripheral is called a Fast Peripheral Bus Clock (F_{PB}).

12.2.1. Peripheral Clock Divider

To support multiple peripheral bus speeds, the CPU clock from CLKGEN1 is further divided to provide two additional clock speeds: standard and slow. The fast-speed peripheral clock FPB is the same as the CPU, the standard-speed peripheral clock (FPB/2), and the slow-speed peripheral clock is one-quarter speed of the peripheral clock (FPB/4), as shown in Figure 12-3.

Figure 12-3. Peripheral Clock Divider Block Diagram



12.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3100	OSCCTRL	31:24	DIAGLOCK							
		23:16	CLKLOCK							
		15:8	PLL2RDY	PLL1RDY	LPRCRDY			POSCRDY	BFRCRDY	FRCRDY
		7:0	PLL2EN	PLL1EN	LPRCEN			POSCEN	BFRCEN	FRCEN
0x3104	OSCCFG	31:24								
		23:16							FRCLPWR[1:0]	
		15:8								
		7:0			POSCIOFNC	KICKSTART	GAIN[1:0]		POSCMD[1:0]	
0x3108	CLKFAIL	31:24	SCSMCH						PLLFAIL2	PLLFAIL1
		23:16								
		15:8	CLKFAIL16	CLKFAIL15	CLKFAIL14	CLKFAIL13	CLKFAIL12	CLKFAIL11	CLKFAIL10	CLKFAIL9
		7:0	CLKFAIL8	CLKFAIL7	CLKFAIL6	CLKFAIL5	CLKFAIL4	CLKFAIL3	CLKFAIL2	CLKFAIL1
0x310C	SCSFAIL	31:24							PLL2SCS	PLL1SCS
		23:16								
		15:8	CLKSCS16	CLKSCS15	CLKSCS14	CLKSCS13	CLKSCS12	CLKSCS11	CLKSCS10	CLKSCS9
		7:0	CLKSCS8	CLKSCS7	CLKSCS6	CLKSCS5	CLKSCS4	CLKSCS3	CLKSCS2	CLKSCS1
0x3110 ... 0x3117	Reserved									
0x3118	CLK1CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x311C	CLK1DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3120	CLK2CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3124	CLK2DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3128	CLK3CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x312C	CLK3DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3130	CLK4CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3134	CLK4DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3138	CLK5CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x313C	CLK5DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3140	CLK6CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3144	CLK6DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3148	CLK7CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x314C	CLK7DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3150	CLK8CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3154	CLK8DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3158	CLK9CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x315C	CLK9DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3160	CLK10CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3164	CLK10DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3168	CLK11CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x316C	CLK11DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3170	CLK12CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3174	CLK12DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3178	CLK13CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x317C	CLK13DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3180	CLK14CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3184	CLK14DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3188	CLK15CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x318C	CLK15DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3190	CLK16CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3194	CLK16DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8	FRACDIV[8:1]							
		7:0	FRACDIV[0]							
0x3198	CLK16CON	31:24	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL	OE	NOSC[3:0]			
		7:0	Reserved[3:0]				COSC[3:0]			
0x3198	PLL1CON	31:24	CLKRDY	PLLSWEN	RIS	FOUTSWEN	EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL		NOSC[3:0]			
		7:0					COSC[3:0]			
0x319C	PLL1DIV	31:24					PLLPRE[3:0]			
		23:16								PLLFBDIV[8]
		15:8	PLLFBDIV[7:0]							
		7:0			POSTDIV1[2:0]			POSTDIV2[2:0]		
0x31A0	VCO1DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8								
		7:0								
0x31A4	PLL2CON	31:24	CLKRDY	PLLSWEN	RIS	FOUTSWEN	EXTCFEN	EXTCFSEL[2:0]		
		23:16	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
		15:8	ON		SIDL		NOSC[3:0]			
		7:0					COSC[3:0]			
0x31A8	PLL2DIV	31:24					PLLPRE[3:0]			
		23:16								PLLFBDIV[8]
		15:8	PLLFBDIV[7:0]							
		7:0			POSTDIV1[2:0]			POSTDIV2[2:0]		
0x31AC	VCO2DIV	31:24		INTDIV[14:8]						
		23:16	INTDIV[7:0]							
		15:8								
		7:0								
0x31B0 ... 0x31B3	Reserved									

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x31B4	CLKDIAG	31:24	FLTIJEN	STOPPLL2	STOPPLL1	GENSEL[4:0]				
		23:16					SCSFLTDATA[3:0]			
		15:8	STOPGEN16	STOPGEN15	STOPGEN14	STOPGEN13	STOPGEN12	STOPGEN11	STOPGEN10	STOPGEN9
		7:0	STOPGEN8	STOPGEN7	STOPGEN6	STOPGEN5	STOPGEN4	STOPGEN3	STOPGEN2	STOPGEN1
0x31B8 ... 0x31FF	Reserved									
0x3200	CM1CON	31:24								
		23:16								
		15:8	ON		SIDL	SLPEN				
		7:0			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
0x3204	CM1STAT	31:24								
		23:16								
		15:8					HWT	LWT	HFT	LFT
		7:0						TRIG	SATD	BUFV
0x3208	CM1WINPR	31:24	WINPR[31:24]							
		23:16	WINPR[23:16]							
		15:8	WINPR[15:8]							
		7:0	WINPR[7:0]							
0x320C	CM1SEL	31:24								
		23:16								
		15:8	CNTSEL[7:0]							
		7:0	WINSEL[7:0]							
0x3210	CM1BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x3214	CM1SAT	31:24	SAT[31:24]							
		23:16	SAT[23:16]							
		15:8	SAT[15:8]							
		7:0	SAT[7:0]							
0x3218	CM1HFAIL	31:24	HFAIL[31:24]							
		23:16	HFAIL[23:16]							
		15:8	HFAIL[15:8]							
		7:0	HFAIL[7:0]							
0x321C	CM1LFAIL	31:24	LFAIL[31:24]							
		23:16	LFAIL[23:16]							
		15:8	LFAIL[15:8]							
		7:0	LFAIL[7:0]							
0x3220	CM1HWARN	31:24	HWARN[31:24]							
		23:16	HWARN[23:16]							
		15:8	HWARN[15:8]							
		7:0	HWARN[7:0]							
0x3224	CM1LWARN	31:24	LWARN[31:24]							
		23:16	LWARN[23:16]							
		15:8	LWARN[15:8]							
		7:0	LWARN[7:0]							
0x3228 ... 0x322F	Reserved									
0x3230	CM2CON	31:24								
		23:16								
		15:8	ON		SIDL	SLPEN				
		7:0			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
0x3234	CM2STAT	31:24								
		23:16								
		15:8					HWT	LWT	HFT	LFT
		7:0						TRIG	SATD	BUFV

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3238	CM2WINPR	31:24	WINPR[31:24]							
		23:16	WINPR[23:16]							
		15:8	WINPR[15:8]							
		7:0	WINPR[7:0]							
0x323C	CM2SEL	31:24								
		23:16								
		15:8	CNTSEL[7:0]							
		7:0	WINSEL[7:0]							
0x3240	CM2BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x3244	CM2SAT	31:24	SAT[31:24]							
		23:16	SAT[23:16]							
		15:8	SAT[15:8]							
		7:0	SAT[7:0]							
0x3248	CM2HFAIL	31:24	HFAIL[31:24]							
		23:16	HFAIL[23:16]							
		15:8	HFAIL[15:8]							
		7:0	HFAIL[7:0]							
0x324C	CM2LFAIL	31:24	LFAIL[31:24]							
		23:16	LFAIL[23:16]							
		15:8	LFAIL[15:8]							
		7:0	LFAIL[7:0]							
0x3250	CM2HWARN	31:24	HWARN[31:24]							
		23:16	HWARN[23:16]							
		15:8	HWARN[15:8]							
		7:0	HWARN[7:0]							
0x3254	CM2LWARN	31:24	LWARN[31:24]							
		23:16	LWARN[23:16]							
		15:8	LWARN[15:8]							
		7:0	LWARN[7:0]							
0x3258 ... 0x325F	Reserved									
0x3260	CM3CON	31:24								
		23:16								
		15:8	ON		SIDL	SLPEN				
		7:0			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
0x3264	CM3STAT	31:24								
		23:16								
		15:8					HWT	LWT	HFT	LFT
		7:0						TRIG	SATD	BUFV
0x3268	CM3WINPR	31:24	WINPR[31:24]							
		23:16	WINPR[23:16]							
		15:8	WINPR[15:8]							
		7:0	WINPR[7:0]							
0x326C	CM3SEL	31:24								
		23:16								
		15:8	CNTSEL[7:0]							
		7:0	WINSEL[7:0]							
0x3270	CM3BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x3274	CM3SAT	31:24	SAT[31:24]							
		23:16	SAT[23:16]							
		15:8	SAT[15:8]							
		7:0	SAT[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3278	CM3HFAIL	31:24	HFAIL[31:24]							
		23:16	HFAIL[23:16]							
		15:8	HFAIL[15:8]							
		7:0	HFAIL[7:0]							
0x327C	CM3LFAIL	31:24	LFAIL[31:24]							
		23:16	LFAIL[23:16]							
		15:8	LFAIL[15:8]							
		7:0	LFAIL[7:0]							
0x3280	CM3HWARN	31:24	HWARN[31:24]							
		23:16	HWARN[23:16]							
		15:8	HWARN[15:8]							
		7:0	HWARN[7:0]							
0x3284	CM3LWARN	31:24	LWARN[31:24]							
		23:16	LWARN[23:16]							
		15:8	LWARN[15:8]							
		7:0	LWARN[7:0]							
0x3288 ... 0x328F	Reserved									
0x3290	CM4CON	31:24								
		23:16								
		15:8	ON		SIDL	SLPEN				
		7:0			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
0x3294	CM4STAT	31:24								
		23:16								
		15:8					HWT	LWT	HFT	LFT
		7:0						TRIG	SATD	BUFV
0x3298	CM4WINPR	31:24	WINPR[31:24]							
		23:16	WINPR[23:16]							
		15:8	WINPR[15:8]							
		7:0	WINPR[7:0]							
0x329C	CM4SEL	31:24								
		23:16								
		15:8	CNTSEL[7:0]							
		7:0	WINSEL[7:0]							
0x32A0	CM4BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x32A4	CM4SAT	31:24	SAT[31:24]							
		23:16	SAT[23:16]							
		15:8	SAT[15:8]							
		7:0	SAT[7:0]							
0x32A8	CM4HFAIL	31:24	HFAIL[31:24]							
		23:16	HFAIL[23:16]							
		15:8	HFAIL[15:8]							
		7:0	HFAIL[7:0]							
0x32AC	CM4LFAIL	31:24	LFAIL[31:24]							
		23:16	LFAIL[23:16]							
		15:8	LFAIL[15:8]							
		7:0	LFAIL[7:0]							
0x32B0	CM4HWARN	31:24	HWARN[31:24]							
		23:16	HWARN[23:16]							
		15:8	HWARN[15:8]							
		7:0	HWARN[7:0]							
0x32B4	CM4LWARN	31:24	LWARN[31:24]							
		23:16	LWARN[23:16]							
		15:8	LWARN[15:8]							
		7:0	LWARN[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x32B8 ... 0x32BF	Reserved									
0x32C0	FRCTUN	31:24								
		23:16								
		15:8								
		7:0			TUN[5:0]					
0x32C4	BFRCTUN	31:24								
		23:16								
		15:8								
		7:0			TUN[5:0]					

12.3.1. System Clock Control Register

Name: OSCCTRL
Offset: 0x3100

Legend: HS = Hardware Settable bit, HC = Hardware Clearable bit, R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DIAGLOCK							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
	CLKLOCK							
Access	R/W							
Reset	0							

Bit	15	14	13	12	11	10	9	8
	PLL2RDY	PLL1RDY	LPRCRDY			POSCRDY	BFRCRDY	FRCRDY
Access	HS/HC/R	HS/HC/R	HS/HC/R			HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	x			0	1	1

Bit	7	6	5	4	3	2	1	0
	PLL2EN	PLL1EN	LPRCEN			POSCEN	BFRCEN	FRCEN
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bit 31 – DIAGLOCK Clock Diagnostic Lock Enable bit

Value	Description
1	Clock diagnostic register (CLK_DIAG) is locked.
0	Clock diagnostic register is not locked, configurations may be modified.

Bit 23 – CLKLOCK Clock Lock Enable bit

Value	Description
1	Clock registers are locked.
0	Clock registers are not locked, configurations may be modified.

Bit 15 – PLL2RDY PLL2 Ready Status bit

Value	Description
1	PLL2 Ready
0	PLL2 Not ready

Bit 14 – PLL1RDY PLL1 Ready Status bit

Value	Description
1	PLL1 Ready
0	PLL1 Not Ready

Bit 13 – LPRCRDY LPRC Ready Status bit

Value	Description
1	Low-power 32 KHz RC oscillator is ready.
0	Low-power 32 KHz RC oscillator is not ready.

Bit 10 – POSCRDY Primary Crystal Oscillator Ready Status bit

Value	Description
1	Primary crystal/resonator oscillator is ready.
0	Primary crystal/resonator oscillator is not ready.

Bit 9 – BFRCRDY Backup FRC Ready Status bit

Value	Description
1	Backup FRC oscillator is ready.
0	Backup FRC oscillator is not ready.

Bit 8 – FRCRDY 8 MHz FRC Ready Status bit

Value	Description
1	FRC oscillator is ready.
0	FRC oscillator is not ready.

Bit 7 – PLL2EN PLL2 Enable bit

Value	Description
1	Enables PLL2
0	Disables PLL2

Bit 6 – PLL1EN PLL1 Enable bit

Value	Description
1	Enables PLL1
0	Disables PLL1

Bit 5 – LPRCEN LPRC Clock Enable bit

Value	Description
1	Enables low-power 32 KHz RC oscillator
0	Disables low-power 32 KHz RC oscillator

Bit 2 – POSCEN Primary Crystal Clock Enable bit

Value	Description
1	Enables primary crystal/resonator oscillator
0	Disables primary crystal/resonator oscillator

Bit 1 – BFRCEN Backup FRC Clock Enable bit

Value	Description
1	Enables backup FRC oscillator
0	Disables backup FRC oscillator

Bit 0 – FRCEN 8 MHz FRC Clock Enable bit

Value	Description
1	Enables FRC oscillator
0	Disables FRC oscillator

12.3.2. Oscillator Configuration Register

Name: OSCCFG
Offset: 0x3104

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
							FRCLPWR[1:0]	
Access							R/W	R/W
Reset							0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			POSCIOFNC	KICKSTART	GAIN[1:0]		POSCMD[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	1	1

Bits 17:16 – FRCLPWR[1:0] FRC Low-Power Mode Enable bits

Value	Description
1	Low-Power mode is enabled.
0	Low-Power mode is disabled.

Bit 5 – POSCIOFNC Primary CLKO Enable Configuration bit

Value	Description
1	CLKO output signal active on the OSCO pin; POSC must be disabled or configured for the External Clock mode.
0	CLKO output is disabled.

Bit 4 – KICKSTART Kick-Starter Programmability for Primary Oscillator bit

Value	Description
1	Boosts the kick start
0	Default kick start

Bits 3:2 – GAIN[1:0] Current Gain Programmability for Oscillator (Output Drive) bits
G3>G2>G1>G0

Value	Description
11	Gain is G3 (use for 24-32 MHz crystals)
10	Gain is G2 (use for 16-24 MHz crystals)
01	Gain is G1 (use for 8-16 MHz crystals)
00	Gain is G0 (use for 4-8 MHz crystals)

Bits 1:0 – POSCMD[1:0] Primary Oscillator Selection bits

Value	Description
11	Off (Reset state, enables GPIO)
10-01	XT mode
00	EC

12.3.3. Reference Clock Fail Status Register

Name: CLKFAIL
Offset: 0x3108

Legend: HS = Hardware Settable bit, R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	SCSMCH						PLLFAIL2	PLLFAIL1
Access	R/W						R/W	R/W
Reset	0						0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CLKFAIL16	CLKFAIL15	CLKFAIL14	CLKFAIL13	CLKFAIL12	CLKFAIL11	CLKFAIL10	CLKFAIL9
Access	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLKFAIL8	CLKFAIL7	CLKFAIL6	CLKFAIL5	CLKFAIL4	CLKFAIL3	CLKFAIL2	CLKFAIL1
Access	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – SCSMCH Source Clock Select Mismatch Indicator bit

Value	Description
1	Clock failure occurred due to SCS integrity check mismatch (or FLTINJ). Check the SCSFAIL register for details.
0	Clock failure occurred due to non-integrity check related failure.

Bit 25 – PLLFAIL2 PLL #2 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the PLL generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 24 – PLLFAIL1 PLL #1 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the PLL generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 15 – CLKFAIL16 Clock Generator #16 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 14 – CLKFAIL15 Clock Generator #15 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 13 – CLKFAIL14 Clock Generator #14 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 12 – CLKFAIL13 Clock Generator #13 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 11 – CLKFAIL12 Clock Generator #12 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 10 – CLKFAIL11 Clock Generator #11 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 9 – CLKFAIL10 Clock Generator #10 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 8 – CLKFAIL9 Clock Generator #9 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 7 – CLKFAIL8 Clock Generator #8 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 6 – CLKFAIL7 Clock Generator #7 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 5 – CLKFAIL6 Clock Generator #6 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 4 – CLKFAIL5 Clock Generator #5 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 3 – CLKFAIL4 Clock Generator #4 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 2 – CLKFAIL3 Clock Generator #3 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 1 – CLKFAIL2 Clock Generator #2 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

Bit 0 – CLKFAIL1 Clock Generator #1 Reference Clock Fail Status bit

Value	Description
1	Selected reference clock failed, or an enabled external clock monitor has detected an error in the output clock, and the CLK generator has switched to the selected backup reference clock.
0	Selected reference clock is functioning.

12.3.4. Source Clock Selection Fail Status Register

Name: SCSFAIL
Offset: 0x310C

Legend: HS = Hardware Settable bit, R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
							PLL2SCS	PLL1SCS
Access							HS/R/W	HS/R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CLKSCS16	CLKSCS15	CLKSCS14	CLKSCS13	CLKSCS12	CLKSCS11	CLKSCS10	CLKSCS9
Access	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLKSCS8	CLKSCS7	CLKSCS6	CLKSCS5	CLKSCS4	CLKSCS3	CLKSCS2	CLKSCS1
Access	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W	HS/R/W
Reset	0	0	0	0	0	0	0	0

Bit 25 – PLL2SCS PLL #2 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 24 – PLL1SCS PLL #1 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 15 – CLKSCS16 Clock Generator #16 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 14 – CLKSCS15 Clock Generator #15 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 13 – CLKSCS14 Clock Generator #14 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 12 – CLKSCS13 Clock Generator #13 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 11 – CLKSCS12 Clock Generator #12 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 10 – CLKSCS11 Clock Generator #11 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 9 – CLKSCS10 Clock Generator #10 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 8 – CLKSCS9 Clock Generator #9 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 7 – CLKSCS8 Clock Generator #8 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 6 – CLKSCS7 Clock Generator #7 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 5 – CLKSCS6 Clock Generator #6 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 4 – CLKSCS5 Clock Generator #5 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 3 – CLKSCS4 Clock Generator #4 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 2 – CLKSCS3 Clock Generator #3 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 1 – CLKSCS2 Clock Generator #2 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

Bit 0 – CLKSCS1 Clock Generator #1 SCS (Source Clock Selection) Circuitry Fail Status bit

Value	Description
1	Source clock selection circuitry detected a failure that it cannot correct.
0	Circuit is correctly selecting the chosen source reference clock.

12.3.5. Clock Generator Control Register

Name: CLKxCON
Offset: 0x3118, 0x3120, 0x3128, 0x3130, 0x3138, 0x3140, 0x3148, 0x3150, 0x3158, 0x3160, 0x3168, 0x3170, 0x3178, 0x3180, 0x3188, 0x3190, 0x3198

Legend: HS = Hardware Settable bit, R = Readable bit, W = Writable bit

Note:
1. The number of external clock fail detection modules is device dependent.

Bit	31	30	29	28	27	26	25	24
	CLKRDY		RIS		EXTCFEN	EXTCFSEL[2:0]		
Access	R/HS/HC		R/W		R/W	R/W	R/W	R/W
Reset	1		0		0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	OE	NOSC[3:0]			
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
	Reserved[3:0]				COSC[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	1

Bit 31 – CLKRDY Output Clock is Ready bit
This bit cannot be Reset.

Value	Description
1	Clock output is ready.
0	Clock output is not ready, possibly because the source clock is not ready, a source selection change is in progress, a divider change is in progress, or a clock failure has been detected and the switch to the backup clock has not yet occurred.

Bit 29 – RIS Run In Sleep bit

Value	Description
1	Clock Generator block will continue to operate if SLEEP mode is entered.
0	Clock Generator block will stop when SLEEP mode is entered.

Bit 27 – EXTCFEN External Clock Fail Event Enable bit

Value	Description
1	External clock fail detection is enabled.
0	External clock fail detection is disabled.

Bits 26:24 – EXTCFSEL[2:0] External Clock Fail Event Select bits[(1)]

Value	Description
[0]	External clock fail detection module #1
[1]	External clock fail detection module #2

Value	Description
[2]	External clock fail detection module #3
[3]	External clock fail detection module #4
[4]	External clock fail detection module #5
[5]	External clock fail detection module #6
[6]	External clock fail detection module #7
[7]	External clock fail detection module #8

Bit 23 – OSWEN Oscillator Switch Enable bit

Value	Description
1	Request an oscillator switch to the selection specified by NOSC[3:0] bits.
0	Oscillator switch is complete.

Bit 22 – DIVSWEN Clock RODIV/ROTRIM Switch Enable bit

Value	Description
1	Clock Divider Switching is currently in progress.
0	Clock Divider Switch has completed.

Bit 20 – FSCMEN Fail-Safe Clock Monitor Enable bit

Value	Description
1	Fail-Safe Clock Monitor is enabled.
0	FSCM is disabled.

Bits 19:16 – BOSC[3:0] Backup Reference Clock Select bits
See Device-Specific Information.

Bit 15 – ON Enable Clock Generator bit

Value	Description
1	Clock Generator is enabled.
0	Clock Generator is disabled.

Bit 13 – SIDL Stop in Idle bit

Value	Description
1	Clock Generator block will stop when IDLE mode is entered.
0	Clock Generator block will continue to operate when IDLE mode is entered.

Bit 12 – OE Output Enable bit

Value	Description
1	Clock output is enabled to be output on a device pin.
0	Clock output on a pin is disabled.

Bits 11:8 – NOSC[3:0] New Reference Clock Select bits
See Device-Specific Information.

Bits 7:4 – Reserved[3:0]

Bits 3:0 – COSC[3:0] Current Reference Clock Selection bits
See Device-Specific Information

12.3.6. Clock Generator Divider Register[(1)]

Name: CLKxDIV
Offset: 0x311C, 0x3124, 0x312C, 0x3134, 0x313C, 0x3144, 0x314C, 0x3154, 0x315C, 0x3164, 0x316C, 0x3174, 0x317C, 0x3184, 0x318C, 0x3194

Notes:
1. The FRC/BFRC variants of the CLKGEN2/3 do not implement the clock divider; the divider ratio for the FRC CLKFEN is fixed at 1x. The associated FRACDIV has an active register that contains the current value of INTDIV[14:0] and FRACDIV[8:0]. The CLKxDIV contents are transferred into the FRACDIV active registers after the associated DIVSWEN bit is set. The DIVSWEN bit is cleared when the transfer is completed.
2. When INTDIV[14:0] = 0, the written values of this bit field are not valid.

Bit	31	30	29	28	27	26	25	24
		INTDIV[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTDIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FRACDIV[8:1]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FRACDIV[0]							
Access	R/W							
Reset	0							

Bits 30:16 – INTDIV[14:0] Integer Divider bit
Integer Divider (INTDIV):
Number of Source Clocks in each 1/2 period of the Divided Clock.
Period Ex:
Divided Clock Period = [Source Clock Period] * INT * 2
Frequency Ex:
`111111111111111` = Clock Generator Divided Clock = Source Clock divided by 65,534 (32,767 *2)
`111111111111110` = Clock Generator Divided Clock Clock = Source Clock divided by 65,532 (32,766 *2)

Value	Description
000000000 000011	Clock Generator Divided Clock = Source Clock divided by 6 (3*2)
000000000 000010	Clock Generator Divided Clock = Source Clock divided by 4 (2*2)
000000000 000001	Clock Generator Divided Clock = Source Clock divided by 2 (1*2)
000000000 000000	Clock Generator Divided Clock = Source Clock (no divider)

Bits 15:7 – FRACDIV[8:0] Fractional Divider bit[(2)]
Number of source clock periods added over 512 source clocks, as equally as possible to each half-period of the output clock.
Provides a fractional additive value for 1/2 period of the output clock.

Value	Description
1111_1111_0	510/512 (0.99609375) divisor added to Integer value
1111_1111_1	511/512 (0.998046875) divisor added to Integer value
100000000	256/512 (0.5000) divisor added to Integer value
0000_0001_0	2/512 (0.00390625) divisor added to Integer value
0000_0000_1	1/512 (0.001953125) divisor added to Integer value
0000_0000_0	0/512 (0.0) divisor added to Integer value

12.3.7. PLL Control Register

Name: PLLxCON
Offset: 0x3198, 0x31A4

Legend: HS = Hardware Settable bit; C = Clearable bit

Note:
1. The number of external clock fail detection modules is device dependent.

Bit	31	30	29	28	27	26	25	24
	CLKRDY	PLLSWEN	RIS	FOUTSWEN	EXTCFEN	EXTCFSEL[2:0]		
Access	R/HS/HC	R/S/HC	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	OSWEN	DIVSWEN		FSCMEN	BOSC[3:0]			
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	1	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL		NOSC[3:0]			
Access	R/W		R/W		R/W	R/W	R/W	R/W
Reset	0		0		0	0	0	1

Bit	7	6	5	4	3	2	1	0
					COSC[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	1

Bit 31 – CLKRDY Output Clock is Ready bit
This bit cannot be Reset.

Value	Description
1	Clock output is ready.
0	Clock output is not ready. It may be due to the source clock not ready, a source selection change is in progress, a divider change is in progress or a clock failure has been detected and the backup clock has not yet occurred.

Bit 30 – PLLSWEN PLL Input and Feedback Divider Switch Enabled bit

Value	Description
1	Enable PLL input and feedback divider update.
0	Divider switch has completed.

Bit 29 – RIS Run in Sleep bit

Value	Description
1	PLL block will continue to operate if SLEEP mode is entered.
0	PLL block will stop when SLEEP mode is entered.

Bit 28 – FOUTSWEN Clock Divider Switch Enabled bit

Value	Description
1	Enable PLL output divider update.
0	Divider switch has completed.

Bit 27 – EXTCFEN External Clock Fail Event Enable bit

Value	Description
1	External clock fail detection is enabled.
0	External clock fail detection is disabled.

Bits 26:24 – EXTCFSEL[2:0] External Clock Fail Event Select bits(1)

Value	Description
0011	External clock fail detection module #4
0010	External clock fail detection module #3
0001	
0000	External clock fail detection module #1

Bit 23 – OSWEN Oscillator Switch Enable bit

Value	Description
1	Request oscillator switch to selection specified by NOSC[3:0] bits.
0	Oscillator switch is complete.

Bit 22 – DIVSWEN Clock RODIV/ROTRIM Switch Enable bit

Value	Description
1	Clock divider switching is currently in progress.
0	Clock Divider Switching has completed.

Bit 20 – FSCMEN Fail-Safe Clock Monitor Enable bit

Value	Description
1	Fail-Safe Clock Monitor is enabled.
0	Fail-Safe Clock Monitor is disabled.

Bits 19:16 – BOSC[3:0] Backup Reference Clock Select bits
See Table 12-4.

Bit 15 – ON Enable PLL Generator bit

Value	Description
1	PLL Generator is enabled.
0	PLL Generator is disabled.

Bit 13 – SIDL Stop in Idle bit

Value	Description
1	Clock generator block will stop when IDLE mode is entered.
0	Clock generator block will continue to operate when IDLE mode is entered.

Bits 11:8 – NOSC[3:0] New Reference Clock Select bits
See Table 12-4.

Bits 3:0 – COSC[3:0] New Reference Clock Select bits
See Table 12-4.

12.3.8. PLL Divider Register

Name: PLLxDIV
Offset: 0x319C, 0x31A8

Bit	31	30	29	28	27	26	25	24
					PLLPRE[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	1

Bit	23	22	21	20	19	18	17	16
								PLLFBDIV[8]
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
	PLLFBDIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	0	0	1	0	0	0

Bit	7	6	5	4	3	2	1	0
			POSTDIV1[2:0]			POSTDIV2[2:0]		
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	0	0	0	1	0

Bits 27:24 – PLLPRE[3:0] PLLx Reference Clock Prescale bits
The allowable PLL reference input clock frequency is 5 MHz to 800 MHz.

Value	Description
111111	63x divide
111110	62x divide
...	
000010	2x divide
000001	1x divide
000000	undefined, not allowed

Bits 16:8 – PLLFBDIV[8:0] PLLx Feedback Divider bit
The allowable PLL reference input clock frequency is 5 MHz to 800 MHz.

Value	Description
0010 0000 0000	512x divide
0001 1111 1111	511x divide
...	
0000 0000 0010	2x divide
0000 0000 0001	1x divide
0000 0000 0000	Undefined, not allowed

Bits 5:3 – POSTDIV1[2:0] PLLx Post Divider #1 bit

Value	Description
111	7x divide
110	6x divide
. . .	
010	2x divide
001	1x divide
000	undefined, not allowed

Bits 2:0 – POSTDIV2[2:0] PLLx Post Divider #2 bit

Value	Description
111	7x divide
110	6x divide
. . .	
010	2x divide
001	1x divide
000	undefined, not allowed

12.3.9. PLL VCO Divider Register

Name: VCOxDIV
Offset: 0x31A0, 0x31AC

Bit	31	30	29	28	27	26	25	24
		INTDIV[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTDIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 30:16 – INTDIV[14:0] PLL VCO Integer Divider bits
Integer Divider (INTDIV):
Number of Source Clocks in each 1/2 period of the Divided Clock.
Period Ex:
Divided Clock Period = [Source Clock Period] * INT * 2
Frequency Ex:
`111111111111111` = Clock Generator Divided Clock = Source Clock divided by 65,534 (32,767 *2)

Value	Description
`000000000000011`	Clock Generator Divided Clock = Source Clock divided by 6 (3*2)
`000000000000010`	Clock Generator Divided Clock = Source Clock divided by 4 (2*2)
`000000000000001`	Clock Generator Divided Clock = Source Clock divided by 2 (1*2)
`000000000000000`	Clock Generator Divided Clock = Source Clock (no divider)

12.3.10. User Clock Diagnostics Control Register

Name: CLKDIAG
Offset: 0x31B4

Bit	31	30	29	28	27	26	25	24
	FLTIJEN	STOPPLL2	STOPPLL1	GENSEL[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
					SCSFLTDATA[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STOPGEN16	STOPGEN15	STOPGEN14	STOPGEN13	STOPGEN12	STOPGEN11	STOPGEN10	STOPGEN9
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STOPGEN8	STOPGEN7	STOPGEN6	STOPGEN5	STOPGEN4	STOPGEN3	STOPGEN2	STOPGEN1
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – FLTIJEN SCS[x][3:0]Fault Injection Enable bit

Value	Description
1	Fault is inserted.
0	Fault insertion is disabled.

Bit 30 – STOPPLL2 PLL2 Reference Clock Monitor Disable bit

Value	Description
1	Selected reference clock for PLL2 is disconnected from the associated clock monitor.
0	Selected reference clock for PLL2 is connected to the associated clock monitor.

Bit 29 – STOPPLL1 PLL1 Reference Clock Monitor Disable bit

Value	Description
1	Selected reference clock for PLL1 is disconnected from the associated clock monitor.
0	Selected reference clock for PLL1 is connected to the associated clock monitor.

Bits 28:24 – GENSEL[4:0] Select the Clock Generator or PLL Generator for Fault Injection bits

Value	Description
11111	Reserved for future PLL generators
11110	PLLGEN2
11101	PLLGEN1
10100-01011	Reserved for future clock generators
01010	CLKGEN11
01001	CLKGEN10
01000	CLKGEN9
00111	CLKGEN8

Value	Description
00110	CLKGEN7
00101	CLKGEN6
00100	CLKGEN5
00011	CLKGEN4
00010	CLKGEN3
00001	CLKGEN2
00000	CLKGEN1

Bits 19:16 – SCSFLTDATA[3:0] Fault Data to be Injected bits
Ones invert the SCS[x] data bit.
Zeros do not effect the SCS[x] data.

Bit 15 – STOPGEN16 Stops the selected reference clock to the clock monitor for CLKGEN16 bit

Value	Description
1	Selected reference clock for CLKGEN16 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN16 is connected to the associated clock monitor.

Bit 14 – STOPGEN15 Stops the selected reference clock to the clock monitor for CLKGEN15 bit

Value	Description
1	Selected reference clock for CLKGEN15 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN15 is connected to the associated clock monitor.

Bit 13 – STOPGEN14 Stops the selected reference clock to the clock monitor for CLKGEN14 bit

Value	Description
1	Selected reference clock for CLKGEN14 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN14 is connected to the associated clock monitor.

Bit 12 – STOPGEN13 Stops the selected reference clock to the clock monitor for CLKGEN13 bit

Value	Description
1	Selected reference clock for CLKGEN13 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN13 is connected to the associated clock monitor.

Bit 11 – STOPGEN12 Stops the selected reference clock to the clock monitor for CLKGEN12 bit

Value	Description
1	Selected reference clock for CLKGEN12 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN12 is connected to the associated clock monitor.

Bit 10 – STOPGEN11 Stops the selected reference clock to the clock monitor for CLKGEN11 bit

Value	Description
1	Selected reference clock for CLKGEN11 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN11 is connected to the associated clock monitor.

Bit 9 – STOPGEN10 Stops the selected reference clock to the clock monitor for CLKGEN10 bit

Value	Description
1	Selected reference clock for CLKGEN10 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN10 is connected to the associated clock monitor.

Bit 8 – STOPGEN9 Stops the selected reference clock to the clock monitor for CLKGEN9 bit

Value	Description
1	Selected reference clock for CLKGEN9 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN9 is connected to the associated clock monitor.

Bit 7 – STOPGEN8 Stops the selected reference clock to the clock monitor for CLKGEN8 bit

Value	Description
1	Selected reference clock for CLKGEN8 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN8 is connected to the associated clock monitor.

Bit 6 – STOPGEN7 Stops the selected reference clock to the clock monitor for CLKGEN7 bit

Value	Description
1	Selected reference clock for CLKGEN7 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN7 is connected to the associated clock monitor.

Bit 5 – STOPGEN6 Stops the selected reference clock to the clock monitor for CLKGEN6 bit

Value	Description
1	Selected reference clock for CLKGEN6 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN6 is connected to the associated clock monitor.

Bit 4 – STOPGEN5 Stops the selected reference clock to the clock monitor for CLKGEN5 bit

Value	Description
1	Selected reference clock for CLKGEN5 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN5 is connected to the associated clock monitor.

Bit 3 – STOPGEN4 Stops the selected reference clock to the clock monitor for CLKGEN4 bit

Value	Description
1	Selected reference clock for CLKGEN4 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN4 is connected to the associated clock monitor.

Bit 2 – STOPGEN3 Stops the selected reference clock to the clock monitor for CLKGEN3 bit

Value	Description
1	Selected reference clock for CLKGEN3 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN3 is connected to the associated clock monitor.

Bit 1 – STOPGEN2 Stops the selected reference clock to the clock monitor for CLKGEN2 bit

Value	Description
1	Selected reference clock for CLKGEN2 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN2 is connected to the associated clock monitor.

Bit 0 – STOPGEN1 Stops the selected reference clock to the clock monitor for CLKGEN1 bit

Value	Description
1	Selected reference clock for CLKGEN1 is disconnected from the associated clock monitor.
0	Selected reference clock for CLKGEN1 is connected to the associated clock monitor.

12.3.11. Clock Monitor Control Register

Name: CMxCON
Offset: 0x3200, 0x3230, 0x3260, 0x3290

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	SLPEN				
Access	R/W		R/W	R/W				
Reset	0		0	0				

Bit	7	6	5	4	3	2	1	0
			CNTDIV[1:0]		FLTINJ[1:0]			WIDTH
Access			R/W	R/W	R/W	R/W		R/W
Reset			0	0	0	0		0

Bit 15 – ON Clock Monitor Enable bit

Value	Description
1	Clock Monitor is enabled.
0	Clock Monitor is disabled.

Bit 13 – SIDL Stop in Idle bit

Value	Description
1	Clock Monitor block will stop when IDLE mode is entered.
0	Clock Monitor block will continue to operate when IDLE mode is entered.

Bit 12 – SLPEN Sleep Mode Enable bit

Value	Description
1	Module continues to operate in Sleep modes.
0	Module does not operate in Sleep modes.

Bits 5:4 – CNTDIV[1:0] Counter Divider bits

Value	Description
11	Reserved
10	Divide-by 4
01	Divide-by 2
00	Divide-by 1

Bits 3:2 – FLTINJ[1:0] Fault Injection Sequence Enable bits

Value	Description
1	Fault injection sequence enabled.

Value	Description
0	Fault injection sequence disabled.

Bit 0 – WIDTH Time Window Selection Control bit

This control bit selects whether the Time Window Generator's clock high pulse defines the accumulation time window period.

Value	Description
1	Rising Edge to Next Falling Edge
0	Rising Edge to Rising Edge, see WINPR[31:0]

12.3.12. Clock Monitor Control Register

Name: CMxSTAT
Offset: 0x3204, 0x3234, 0x3264, 0x3294

Note:
1. Do not change the value while in operation.
2. CMxHWT/CMxLWT and CMxHFT/CMxLFT must all be cleared before the Fault can be injected.
3. When set high, the function is intended for the pulse width measurement feature where the monitored clock (as opposed to the reference clock) is expected to clock the time window generator responsible for defining the measurement time period. Conversely, the reference clock is used to clock the counter at a higher rate of switching frequency.
4. When set high, the content of WINPR[31:0] prescaler is ignored while both the time window generator and the counter are reset.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					HWT	LWT	HFT	LFT
Access					R/W	R/W	R/W	R/W
Reset					1	1	1	1

Bit	7	6	5	4	3	2	1	0
						TRIG	SATD	BUFV
Access						R/W	R/W	R/W
Reset						1	1	1

Bit 11 – HWT High Warning Threshold Status bit
This status bit is set by hardware when the captured count value exceeds that of the high warning threshold register content. In other words, the monitored clock is at a higher frequency than desired (or the reference clock is running slower than expected).

Value	Description
1	High threshold warning
0	No warning

Bit 10 – LWT Low Warning Threshold Status bit
This status bit is set by hardware when the captured count value is less than that of the low warning threshold register content. In other words, the monitored clock is at a lower frequency than desired (or the reference clock is running faster than expected).

Value	Description
1	Low threshold warning
0	No warning

Bit 9 – HFT High Failing Threshold Status bit

This status bit is set by hardware when the captured count value exceeds that of the high failing threshold register content. In other words, the monitored clock is at a higher frequency than allowed (or the reference clock is running slower than allowed). When instanced as a Fail-Safe Clock Monitor, a hardware switchover to the backup clock source will occur.

Value	Description
1	High threshold failing
0	No failure

Bit 8 – LFT Low Failing Threshold Status bit

This status bit is set by hardware when the captured count value is less than that of the low failing threshold register content. In other words, the monitored clock is at a lower frequency than allowed (or the reference clock is running faster than allowed). When instanced as a Fail Safe Clock Monitor, a hardware switchover to the backup clock source will occur.

Value	Description
1	Low threshold failing
0	No failure

Bit 2 – TRIG Time Window Generator Trigger Status bit

Value	Description
1	Accumulation time window has (re)started/stopped.
0	Accumulation time window has not started.

Bit 1 – SATD Counter Saturated Status bit

Counter has saturated at SAT[31:0].

Bit 0 – BUFV Buffer Valid Status bit

Value	Description
1	An accumulated count has been captured into BUF[31:0] and is ready for software use.
0	An accumulated count has not been captured into BUF[31:0] yet or has been blocked by catastrophic failure of the monitored clock for more than one accumulation window.

12.3.13. Clock Monitor Prescaler Register

Name: CMxWINPR
Offset: 0x3208, 0x3238, 0x3268, 0x3298

Bit	31	30	29	28	27	26	25	24
	WINPR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WINPR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	WINPR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WINPR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – WINPR[31:0] Clock Monitor Prescaler bits
Value to determine the accumulation window for the reference clock.

12.3.14. Clock Monitor Input Selection Register

Name: CMxSEL
Offset: 0x320C, 0x323C, 0x326C, 0x329C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CNTSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WINSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CNTSEL[7:0] Counter Clock Source bits
Selects the monitored clock source. See Table 12-5.

Bits 7:0 – WINSEL[7:0] Window Clock Source bits
Selects the reference clock source. See Table 12-5.

12.3.15. Clock Monitor Buffer Register

Name: CMxBUF
Offset: 0x3210, 0x3240, 0x3270, 0x32A0

Bit	31	30	29	28	27	26	25	24
	BUF[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BUF[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – BUF[31:0] Monitored Clock Count Value bits
The Clock Monitor Data Buffer register contains the final accumulated count recorded by the counter during the previous accumulation window. Its content feeds the threshold limit comparators.

12.3.16. Clock Monitor Saturation Register

Name: CMxSAT
Offset: 0x3214, 0x3244, 0x3274, 0x32A4

Bit	31	30	29	28	27	26	25	24
	SAT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	SAT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	SAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	SAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 31:0 – SAT[31:0] Clock Monitor Counter Saturation bits
The Clock Monitor Counter Saturation register contains the accumulated count value which causes the counter to saturate. If the counter has reached the count value programmed into this register before being captured, the CMxSATD bit is set with an interrupt invoked.

12.3.17. Clock Monitor High Threshold Failing Register

Name: CMxHFAIL
Offset: 0x3218, 0x3248, 0x3278, 0x32A8

Bit	31	30	29	28	27	26	25	24
	HFAIL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	HFAIL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	HFAIL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	HFAIL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 31:0 – HFAIL[31:0] Clock Monitor High Threshold Failing bits

The Clock Monitor High Threshold Failing register contains the upper failing threshold limit against which the captured count is compared (see HWT). Failure is signaled when CMxBUF[31:0] > CMxHFAIL, so this register defines the fastest monitored frequency, slowest scaled reference frequency or longest reference pulse width that can be measured before triggering a failure. Setting CMxHFAIL = 0xFFFFFFFF will have the effect of disabling this threshold.

12.3.18. Clock Monitor Low Threshold Failing Register

Name: CMxLFAIL
Offset: 0x321C, 0x324C, 0x327C, 0x32AC

Bit	31	30	29	28	27	26	25	24
	LFAIL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LFAIL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LFAIL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LFAIL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LFAIL[31:0] Clock Monitor Low Threshold Failing bits

The Clock Monitor Low Threshold Failing register contains the lower failing threshold limit against which the captured count is compared (see LFT). Failure is signaled when CMxBUF[31:0] < CMxLFAIL, so this register defines the slowest monitored frequency, fastest scaled reference frequency or shortest reference pulse width that can be measured before triggering a failure. Setting CMxLFAIL = 0x00000000 will have the effect of disabling this threshold.

12.3.19. Clock Monitor High Threshold Warning Register

Name: CMxHWARN
Offset: 0x3220, 0x3250, 0x3280, 0x32B0

Bit	31	30	29	28	27	26	25	24
	HWARN[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	HWARN[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	HWARN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	HWARN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 31:0 – HWARN[31:0] Clock Monitor High Threshold Warning bits

The Clock Monitor High Threshold Warning register contains the upper warning threshold limit against which the captured count is compared (see HFT). Warning is signaled when CMxBUF[31:0] > CMxHWARN, so this register defines the fastest monitored frequency, slowest scaled reference frequency or longest reference pulse width that can be measured before triggering a warning. Setting CMxHWARN = 0xFFFFFFFF will have the effect of disabling this threshold.

12.3.20. Clock Monitor Low Threshold Warning Register

Name: CMxLWARN
Offset: 0x3224, 0x3254, 0x3284, 0x32B4

Bit	31	30	29	28	27	26	25	24
	LWARN[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LWARN[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LWARN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LWARN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LWARN[31:0] Clock Monitor Low Threshold Warning bits
The Clock Monitor Low Threshold Warning register contains the lower warning threshold limit against which the captured count is compared (see LWT). Warning is signaled when CMxBUF[31:0] < CMxLWARN, so this register defines the slowest monitored frequency, fastest scaled reference frequency or shortest reference pulse width that can be measured before triggering a warning. Setting CMxLWARN = 0x00000000 will have the effect of disabling this threshold.

12.3.21. 8 MHz FRC Controller Register

Name: FRCTUN
Offset: 0x32C0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			TUN[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 5:0 – TUN[5:0] Internal Fast RC Oscillator Tuning bits
This bit field specifies the user tuning capability for the internal fast RC oscillator.

Value	Description
011111	Maximum Frequency
011110	
...	
000001	
000000	Center Frequency, oscillator is running at a calibrated frequency.
111111	
111110	
...	
100001	
100000	Minimum Frequency

12.3.22. BFRC Controller Register

Name: BFRCTUN
Offset: 0x32C4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			TUN[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 5:0 – TUN[5:0] Internal Fast RC Oscillator Tuning bits
This bit field specifies the user tuning capability for the internal fast RC oscillator.

Value	Description
011111	Maximum Frequency
011110	
...	
000001	
000000	Center Frequency, oscillator is running at a calibrated frequency.
111111	
111110	
...	
100001	
100000	Minimum Frequency

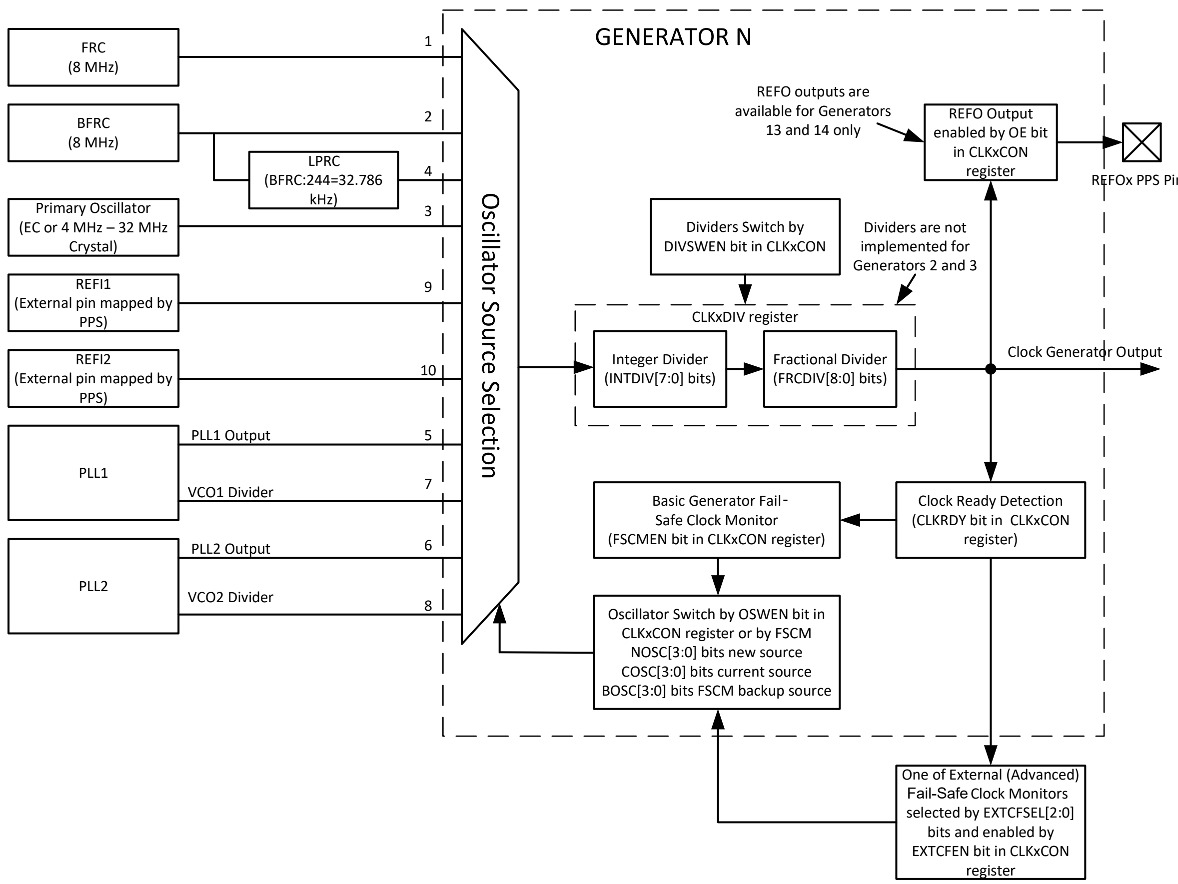
12.4. Operation

12.4.1. Clock Generators

The dsPIC33AK256MPS306 family of devices contains multiple clock generators (CLKGEN). The Clock/PLL generators can be configured for any clock source available. Clock generator 1 will act as the system clock, and it will always be enabled.

An expanded view of the CLKGEN is detailed in Figure 12-4.

Figure 12-4. Clock Generator



The assignment of CLKGEN to peripherals is listed in Table 12-2.

The other clock generators can be configured for any clock source. For example, to reference FRC, set the COSC[3:0] bits to the FRC source and enable clock switching.

Each generator can be enabled either by setting the ON bit in the CLKxCON register or when requested by a consumer (e.g., a peripheral).

The ON bit is logically ORed with clock request signals from other peripherals to enable the CLKGEN or PLL, and it can only be used to disable the CLKGEN or PLL when there are no active clock requests. In addition, clearing the OSCCTRL.PLLxEN bit disables PLLx.

Each clock generator has dedicated dividers (CLKxDIV) and control bits.

The INTDIV bitfield within CLKxDIV is an integer divider. It divides the clock frequency by two multiplied by the value of INTDIV. The FRACDIV is a fractional divider that allows for intermediate division that is not possible by using the integer divider alone. The value of FRACDIV is divided by 512 to create a fraction that is added to the INTDIV value.

Equation 12-1 provides the relationship between the Clock Generator input frequency (F_{IN}) and the Clock Generator output frequency (F_{OUT}).

Equation 12-1. F_{OUT} Calculation

$$F_{OUT} = \frac{F_{IN}}{2\left(INTDIV + \frac{FRACDIV}{512}\right)}$$

Equation 12-2 provides the relationship between the Clock Generator input frequency (F_{IN}) and the Clock Generator FRACDIV value.

Equation 12-2. FRACDIV Calculation

$$FRACDIV = 256\frac{F_{IN}}{F_{OUT}} - 512(INTDIV)$$

12.4.1.1. Fail-Safe Clock Monitor (FSCM)

Each CLKGEN has an FSCM built inside to provide clock safety. Each FSCM is enabled via CLKxCON.FSCMEN. To provide a faster response to a clock failure, the FSCM uses the 8 MHz BFRC as the reference clock.

Note: FSCM supports automatic switchover to a backup clock source with options for programmable over-frequency or under-frequency thresholds.

In the event of an oscillator failure, the FSCM will generate an Oscillator Fail Interrupt, set the associated CLKFAIL/PLLFAIL flag, and switch the clock over to the specified backup Fail-Safe clock source, which is selected via the CLKxCON.BOSC bits. The Fail-Safe condition is exited with either a Reset or by completing a clock switch to a new stable clock input for the CLKGEN.

Note: For CLKGEN1, select ONLY an active and running clock source as the BOSC source (such as FRC, BFRC, etc.). At the time of BOSC initialization, the clock source should be up and running.

A clock monitor external to the CLKGENs is also included and is discussed in the Clock Monitor section.

12.4.2. Setup for Using Clock Generator with 8 MHz Internal FRC

The following process is used to set up the CLKGEN1 to operate the device with an 8 MHz Internal FRC:

1. Enable the clock generator (if not enabled by default) by setting the ON bit in the CLKxCON register.
2. To set up the fail-safe for the clock generator, follow these steps:
   a. Select the backup clock source by selecting the BOSC bits in the CLKxCON register.
   b. Enable a fail-safe clock failure by setting the FSCMEN bit in the CLKxCON register.
   c. Enable ClkFailInterrupt to generate an interrupt during clock failure.
3. To switch to a new oscillator for the clock generator, follow these steps:
   a. Select a clock source to switch by writing to NOSC bits in the CLKxCON register.
   b. Enable switching by writing to the OSWEN bit in the CLKxCON register.
4. To further divide the clock out of the clock generator using CLKxDIV, follow these steps:
   a. Set the integer divide factor bit setting INTDIV bits in the CLKxDIV register.
   b. Set the fractional divide factor bit setting FRACDIV bits in the CLKxDIV register. FRACDIV will not work if INTDIV is configured to 0.
   c. Set the DIVSWEN bit in the CLKxCON register to enable divide factors to get updated.

Example 12-1. Code Example for Clock Generator 6 Switching to FRC

```
// Enable clock generator
CLK6CONbits.ON = 1;

// Configure backup oscillator in case of failure
CLK6CONbits.BOSC = 2; // BFRC
CLK6CONbits.FSCMEN = 1; // Enable fail safe monitor
// Configure clock divide Fdiv = Fin / 2*(INTDIV+(FRACDIV/512))
CLK6DIVbits.INTDIV = 1; // Integer divide factor
```

```
CLK6DIVbits.FRACDIV = 128; // Fractional divide factor
CLK6CONbits.DIVSWEN = 1; // Enable divide factors to get updated
while (CLK6CONbits.DIVSWEN != 0); // Wait for switching (hardware cleared)

// Enable clock switching
CLK6CONbits.NOSC = 1 ; // Select FRC clock source
CLK6CONbits.OSWEN = 1; // Enable clock switching
while (CLK6CONbits.OSWEN != 0); // Wait for switching (hardware cleared)

// Enable clock failure interrupt
IFS0bits.CLKFAILIF = 0;
IEC0bits.CLKFAILIE = 1;
```

12.4.3. Primary Oscillator (POSC)

The dsPIC33AK256MPS306 devices contain one instance of the Primary Oscillator (POSC). The Primary Oscillator is available on the OSCO and OSCI pins of the dsPIC33A devices. This connection enables an external crystal (or ceramic resonator) to provide the clock to the device. The POSC oscillator is enabled by the POSCEN bit in the OSCCTRL register and is ready for operation when the POSCRDY bit is set in the OSCCTRL register. The Primary Oscillator has three modes of operation listed in Table 12-6.

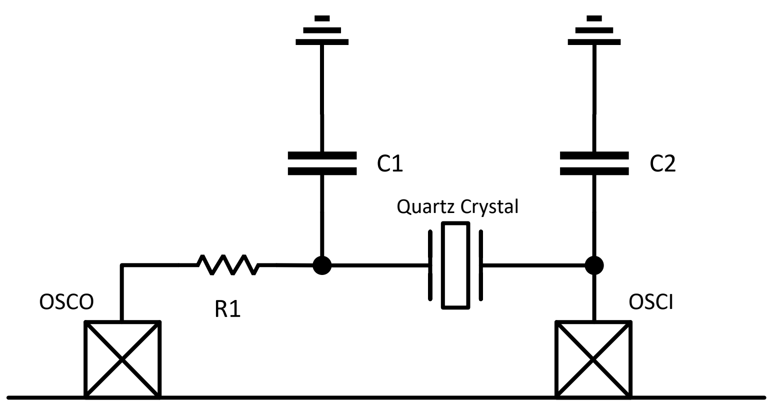
Table 12-6. Primary Oscillator Modes

POSCMD[1:0] bits in OSCCFG register	Mode Description
3	POSC is disabled. OCSI can be used as regular I/O pins. The OSCO pin is controlled by the POSCIOFNC bit in the OSCCFG register.
2-1	Quartz Crystal Mode. OSCO and OSCI pins are connected to a crystal driver.
0	External Clock Mode (EC). The clock source must be connected to the OSCI pin. The OSCO pin is controlled by the POSCIOFNC bit in the OSCCFG register.

If the Primary Oscillator is disabled or configured in EC mode, the OSCO pin can output the device system (CPU) clock when the POSCIOFNC bit in the OSCCFG register is set.

Figure 12-5 shows a recommended Pierce oscillator circuit diagram for the dsPIC33A devices.

Figure 12-5. Recommended Crystal Connection Diagram



Capacitors C1 and C2 form a load capacitance (C_{LOAD}) for the crystal, and the R1 resistor limits the power dissipated in the quartz. The optimum load capacitance for a given crystal is specified by the crystal manufacturer. Load capacitance can be calculated as shown in Equation 12-3.

Equation 12-3. Crystal Load Capacitance

$$C_{LOAD} = C_{STRAY} + \frac{C_1 \times C_2}{C_1 + C_2}$$

Where C_{STRAY} is the stray capacitance between the quartz crystal pads on the board (in many cases, it can be assumed to be 1-3 pF), C_1 and C_2 are capacitors connected to the crystal.

Assuming that $C_1 = C_2$, Equation 12-4 gives the capacitors C_1 and C_2 values for a given crystal load and stray capacitance.

Equation 12-4. External Capacitors Value for the Crystal

$$C_{OSC} = C_1 = C_2 = 2 \times (C_{LOAD} - C_{STRAY})$$

Where C_{LOAD} is a nominal load capacitance from the crystal spec, and C_{STRAY} is the stray capacitance between the quartz crystal pads on the board.

If the power dissipated in the crystal is higher than the value specified by the crystal manufacturer, the current must be limited by resistor R1 to avoid overdriving the crystal. An initial estimation of R1 can be obtained by considering the voltage divider formed by R1 and C1. Thus, the initial value of R1 is equal to the impedance of C1, as shown in Equation 12-5.

Equation 12-5. Drive Level Limiting Resistor Initial Calculation

$$R1 = \frac{1}{2 \times \pi \times F \times C1}$$

Where F is the crystal's nominal oscillation frequency.

The power dissipated in quartz can be verified with an oscilloscope using Equation 12-6.

Equation 12-6. Power Dissipated in Quartz Crystal

$$P = \frac{R_{ESR} \times (\pi \times F \times \left(C2 + \frac{C_{STRAY}}{2} + C_{PROBE}\right))^2 \times V_{pp}^2}{2}$$

Where R_{ESR} is the series resistance from the crystal specification, F is the crystal's nominal oscillation frequency, C_{STRAY} is the stray capacitance between the quartz crystal pads on the board, C_{PROBE} is the capacitance of the oscilloscope probe connected to C2 and V_{pp} is the peak-to-peak amplitude of the signal on the C2 capacitor measured with an oscilloscope.

The calculated power must be less than what is specified in the crystal specifications.

The Primary Oscillator gain for the Crystal mode is selected by the GAIN[2:0] bits. For each gain mode, the GAIN[2:0] bits set the oscillator driver transconductance level as specified in the electrical specifications.

The transconductance of the crystal oscillator driver, required to start the oscillation quickly enough and sustain stable oscillation, must be about four times greater than the transconductance of the quartz crystal circuit.

The transconductance of the quartz crystal circuit can be estimated by using Equation 12-7.

Equation 12-7. Minimum Required Transconductance for the Oscillation

$$G_{min} = 158 \times (R_{ESR} + R1) \times F^2 \times \left(C_{STRAY} + \frac{C_{OSC}}{2}\right)^2$$

Where R_{ESR} is the equivalent series resistance from the crystal spec, R1 is a power-limiting resistor, *F* is the crystal's nominal oscillation frequency, C_{STRAY} is the stray capacitance between the quartz crystal pads on the board and $C_{OSC}/2$ is the external load capacitance for the crystal.

The transconductance selected by the GAIN[2:0] bits should be at least four times larger than the calculated G_{min}; however, gain settings that are too high may overload the crystal.

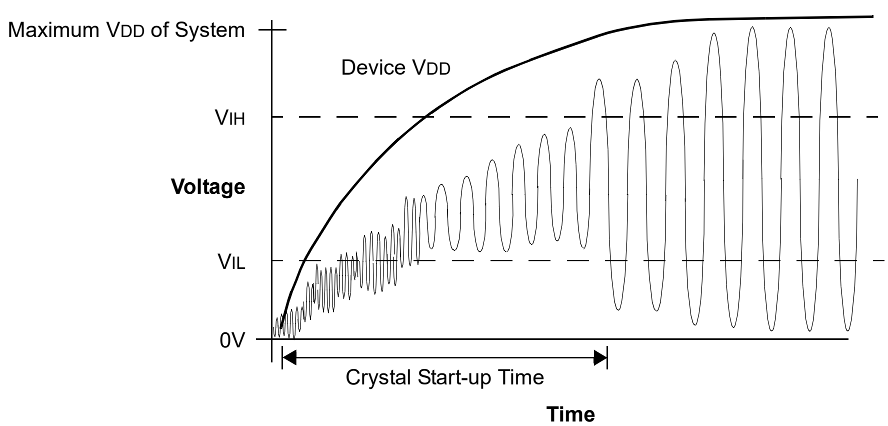
12.4.3.1. Oscillator Start-up Time

As the device voltage increases from V_{SS}, the oscillator will start its oscillations. The time required for the oscillator to start oscillating depends on these factors:

- Crystal and resonator frequency
- Capacitor values used (C1 and C2 in Figure 12-5)
- Device V_{DD} rise time
- System temperature
- Series resistor value and type if used
- Oscillator mode selection of device (selects the gain of the internal oscillator inverter)
- Crystal quality
- Oscillator circuit layout
- System noise

Figure 12-6 illustrates a plot of a typical oscillator and resonator start-up.

Figure 12-6. Example Oscillator and Resonator Start-up Characteristics



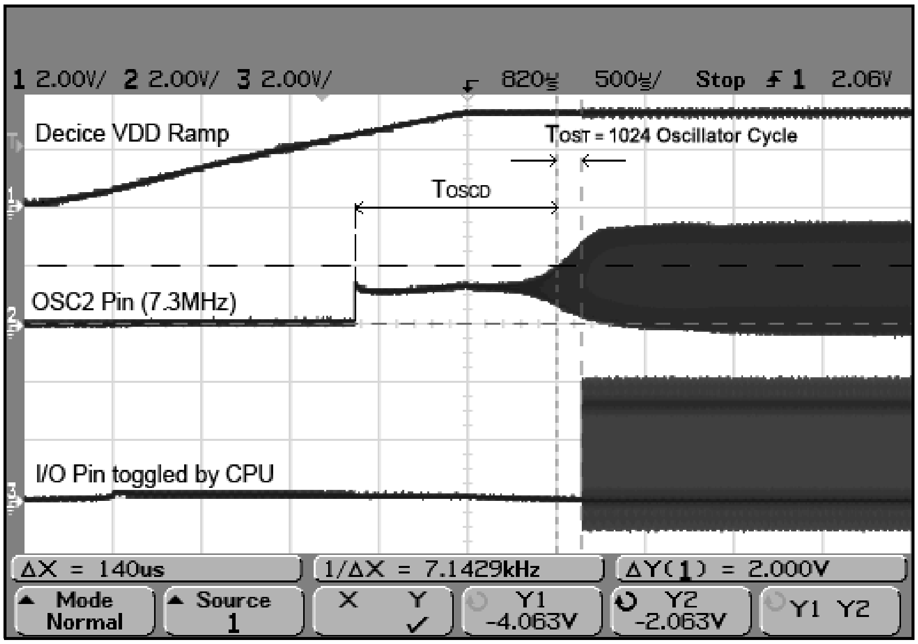
To ensure that a crystal oscillator (or ceramic resonator) has started and stabilized, an Oscillator Start-up Timer (OST) is provided with the POSC. The OST is a simple, 10-bit counter that counts 1024 cycles before releasing the oscillator clock to the rest of the system. This time-out period is denoted as T_{OST}.

The amplitude of the oscillator signal must reach the V_{IL} and V_{IH} thresholds for the oscillator pins before the OST can begin to count cycles. The T_{OST} interval is required every time the oscillator restarts (that is, on POR, BOR and wake-up from Sleep mode) when XT or HS mode is selected in the Configuration Words. The T_{OST} timer does not exist when EC mode is selected.

After the POSC is enabled, it takes a finite amount of time to start oscillating. This delay is denoted as T_{OSCD}. After T_{OSCD}, the OST timer takes 1024 clock cycles (T_{OST}) to release the clock. The total delay for the clock to be ready is: $T_{OSCD} + T_{OST}$. If the PLL is used, an additional delay is required for the PLL to lock. For more information, see Phase-Locked Loop (PLL).

POSC start-up behavior is illustrated in Figure 12-7, where the CPU begins toggling an I/O pin when it starts execution after the $T_{OSCD} + T_{OST}$ interval.

Figure 12-7. Oscillator Start-up Characteristics



12.4.3.2. Primary Oscillator Pin Functionality

The POSC pins (OSCI and OSCO) can be used for other functions when the oscillator is not being used. The POSCMD[1:0] bits in the Oscillator Configuration register (OSCCFG) determine the oscillator pin function. The POSCIOFNC bit (OSCCFG) determines the OSCO pin function.

POSCIOFNC: OSCO Pin Function bit (except in XT mode):

- `1` = OSCO is the clock output and the instruction cycle (F_{CY}) clock is output on the OCSO pin (see Figure 12-8)
- `0` = OSCO is a general-purpose digital I/O pin

The oscillator pin functions are provided in Table 12-7.

Table 12-7. Clock Pin Function Selection

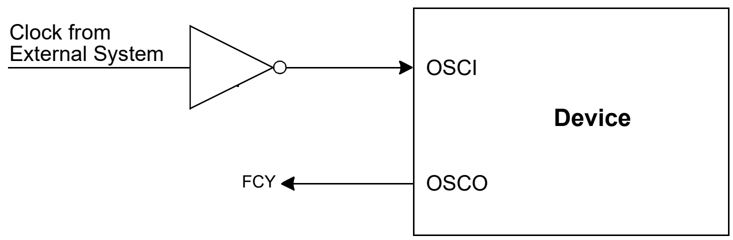
Oscillator Source	POSCIOFNC Value	POSCMD[1:0] Value	OSCI Pin Function(1)	OSCO Pin Function(2)
POSC Disabled	1	11	Digital I/O	Clock Output
POSC Disabled	0	11	Digital I/O	Digital I/O
XT	x	10-01	OSCI	OSCO
EC	1	00	OSCI	Clock Output

Table 12-7. Clock Pin Function Selection (continued)

Oscillator Source	POSCIOFNC Value	POSCMD[1:0] Value	OSCI Pin Function[1]	OSCO Pin Function[2]
EC	0	00	—	Digital I/O

Notes:
1. OSCI pin function is determined by the Primary Oscillator Mode Selection (POSCMOD[1:0]) Configuration bits.
2. OSCO pin function is determined by the Primary Oscillator Mode Selection (POSCMOD[1:0]), OSCIOFNC Configuration bits.

Figure 12-8. OSCO Pin for Clock Output (in EC Mode)



12.4.4. Internal Fast RC (FRC) Oscillator

The dsPIC33A devices contain one instance of the Internal Fast RC (FRC) Oscillator. The FRC provides a nominal 8 MHz clock without requiring an external crystal or ceramic resonator, which results in system cost savings for applications that do not require a precise clock reference. The application software can tune the frequency of the oscillator using the TUN[5:0] bits in the FRC Oscillator Trim register FRCTUN. The FRC oscillator can be enabled or disabled by the FRCEN bit in the OSCCTRL register and is ready when the FRCRDY bit is set in the OSCCTRL register. The FRC is always enabled after any Reset.

The Internal FRC Oscillator starts quickly. Unlike a crystal oscillator, which can take several milliseconds to begin oscillation, the Internal FRC starts oscillating immediately.

12.4.5. BFRC Oscillator

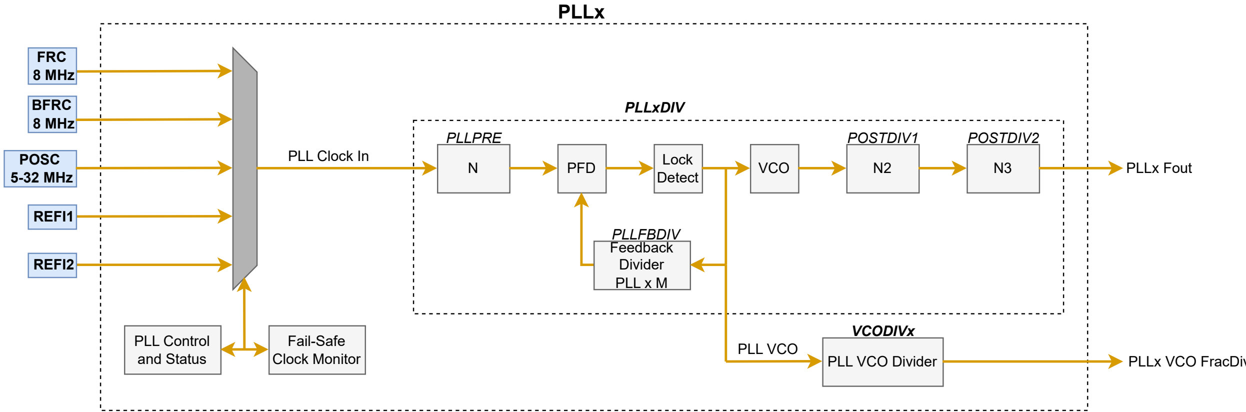
The dsPIC33A devices contain one instance of the Internal Backup Fast RC (BFRC) Oscillator and can function as the system clock in the event of an FRC failure. The BFRC Oscillator provides a nominal 8 MHz clock without requiring an external crystal or ceramic resonator, which results in system cost savings for applications that do not require a precise clock reference. The BFRC oscillator can be enabled or disabled by the BFRCEN bit in the OSCCTRL register and is ready when the BFRCRDY bit is set in the OSCCTRL registers. The BFRC is always enabled after any Reset.

The application software can tune the frequency of the oscillator using the TUN[5:0] bits in the FRC Oscillator Tuning register BFRCTUN.

12.4.6. Phase-Locked Loop (PLL)

The POSC and Internal FRC Oscillator sources can optionally use an on-chip PLL to achieve higher operating speeds. The PLLs can be enabled by the PLLxEN bits in the OSCCTRL register and are ready when the corresponding PLLxRDY bit is set in the OSCCTRL register. Figure 12-9 illustrates a block diagram of the PLL module.

Figure 12-9. PLL Block Diagram



For PLL operation, the frequency range requirements, as specified in Table 42-22, must be met at all times without exception for the following frequencies:

- PLL Input Frequency (FPLLI)
- VCO Frequency
- Feedback Divider
- Output of the PLL module. The first output divider (POSTDIV1) value should be larger than the value for the second output divider (POSTDIV2).

The PLL Phase Detector Input Divider Select bits (PLLPRE[3:0]) in the PLL Divider register (PLLxDIV[29:24]) specify the input divider ratio (N1), which is used to scale down the input clock ($_{FPLLI}$) to meet the PFD input frequency range of 5.0 MHz to 800 MHz.

The PLL Feedback Divider bits (PLLFBDIV[8:0]) in the PLL Divider register (PLLxDIV[16:8]) specify the divider ratio (M), which scales down the VCO Output Frequency (F_{VCO}) for feedback to the PFD input. The VCO Frequency (F_{VCO}) is 'M' times the PFD Input Frequency (F_{PFD}).

There are two PLL VCO output dividers configured through the POSTDIV1[2:0] and POSTDIV2[2:0] select bits. These bits are located in the PLL Divider register (PLLxDIV[6:4] and PLLxDIV[2:0]) and specify the divider ratios (N2 and N3) that limit the PLL Output Frequency (PLL FOUT). Please refer to Table 12-2 for maximum frequencies related to each peripheral.

Equation 12-8 provides the relationship between the PLL Input Frequency ($_{FPLLI}$) and VCO Output Frequency (F_{VCO}).

Equation 12-8. F_{VCO} Calculation

$$FVCO = FPLLI \times \left(\frac{M}{N1}\right) = FPLLI \times \left(\frac{PLLFBDIV[8:0]}{PLLPRE[3:0]}\right)$$

Equation 12-9 provides the relationship between the PLL Input Frequency (F_{PLLI}) and the PLL Output Frequency (F_{PLLO}).

Equation 12-9. F_{PLLO} Calculation

$$FPLLO = FPLLI \times \left(\frac{M}{N1 \times N2 \times N3}\right) = FPLLI \times \left(\frac{PLLFBDIV[8:0]}{PLLPRE[3:0] \times POSTDIV1[2:0] \times POSTDIV2[2:0]}\right)$$

Where:
$M = PLLFBDIV[8:0]$
$N1 = PLLPRE[3:0]$
$N2 = POSTDIV1[2:0]$
$N3 = POSTDIV2[2:0]$

12.4.6.1. Input Clock Limitation at Start-up for PLL Mode

Table 12-8 provides the default values of the PLL prescaler, PLL feedback divider and both PLL postscalers at Power-on Reset (POR).

Table 12-8. PLL Mode Defaults

Register	Bit Field	Value at POR Reset	PLL Divider Ratio
PLLxDIV	PLLPRE[3:0]	0001	N1 = 1
PLLxDIV	POSTDIV1[2:0]	111	N2 = 7
PLLxDIV	POSTDIV2[2:0]	001	N3 = 1
PLLxDIV	PLLFBDIV[8:0]	11001000	M = 200

Given these Reset values, the following equations provide the PLL Input Frequency (F_{PLLI}) and VCO Output Frequency (F_{VCO}) at Power-on Reset.

Equation 12-10. F_{VCO} at Power-on Reset

$$FVCO = FPLLI\left(\frac{M}{N1}\right) = FPLLI\left(\frac{200}{1}\right) = 200\ FPLLI$$

Equation 12-11. FPLLO at Power-on Reset

$$FPLLO = FPLLI\left(\frac{M}{N1 \times N2 \times N3}\right) = FPLLI\left(\frac{200}{1 \times 7 \times 1}\right) = 28.5\ FPLLI$$

To use the PLL with settings other than the default settings and to ensure all PLL requirements are met, follow this process:

1. Power up the device with the Internal FRC.
2. Change the FBDIV, PLLPRE, POSTDIV1 and POSTDIV2 bit values, based on the input frequency, to meet these PLL requirements:
   The PLL Input Frequency (F_{PLLI}) must be in the range specified in Electrical Characteristics.
   The VCO Output Frequency (F_{VCO}) must be in the range specified in Electrical Characteristics.
3. Writing PLLxCON:
   a) Enable PLL Input and Feedback Divider update by setting the PLLSWEN bit in the PLLxCON register.
   b) The first output divider (POSTDIV1) should be larger than the value of the second output divider (POSTDIV2). The output dividers POSTDIV1 and POSTDIV2 should not be changed while the PLL is operating. The input reference clock divider and the feedback divider may be updated during a PLL operation.
   c) Enable the PLL Output Divider update by setting the FOUTSWEN bit in the PLLxCON register.
   d) Select the clock source by setting the NOSC[3:0] bits in the PLLxCON register.

e) Enable clock switching by setting the OSWEN bit in the PLLxCON register.

Notes:

1. It is recommended to change clock divider settings before the initial clock switch occurs. The reference clock (PLLPRE) and feedback dividers (FBDIV) can be changed during a PLL operation, but care should be taken to not generate an invalid clock frequency.
2. When switching from a low-speed clock to a high-speed clock, the DIVSW process should be done before a clock switch to prevent undivided high-speed clocks from passing through. For a high-to-low switch, the clock switch should occur before a DIVSW.

12.4.6.2. PLL Lock Status

Whenever the PLL input frequency, the PLL prescaler or the PLL feedback divider is changed, the PLL requires a finite amount of time (T_{LOCK}) to synchronize to the new settings.

T_{LOCK} is applied when the PLL is selected as the clock source during a clock switching operation. The value of T_{LOCK} is relative to the time at which the clock is available to the PLL input. For example, with the POSC, T_{LOCK} starts after the OST delay. Refer to the Table 42-22 for more information about typical T_{LOCK} values.

The PLLxRDY bit in the Oscillator Control register (OSCCTRL[5]) and the CLKRDY bit in the PLL Control register PLLxCON[31] are read-only status bits that indicate the PLL ready status. The LOCK bit is cleared at a Power-On Reset and during a clock switch operation when the PLL is selected as the destination clock source. It remains clear when any clock source not using the PLL is selected. After a clock switch event in which the PLL is enabled, it is advisable to wait for the LOCK bit to be set before executing other code.

12.4.6.3. PLL Setup

12.4.6.3.1. Setup for Using PLL with the Primary Oscillator (POSC)

The following process is used to set up the PLL to operate the device at 200 MHz with a 10 MHz external crystal:

1. Set the configuration for the POSC to XT mode using POSCMD[1:0] bits in the OSCCFG register.
2. Setting up divider settings: to execute instructions at 200 MHz, a PLL output frequency of 200 MHz will be required. To set up the PLL and meet the requirements of the PLL, follow these steps:
   a. Select the PLL prescaler.
      - Select a PLL prescaler value of N1 = 1.
      - F_{PLLI} = 10 MHz
      - F_{PFD} = 10 MHz(1/N1) = 10 MHz(1) = 10 MHz
   b. Select the feedback divider to meet the VCO output frequency requirement as well as achieve the desired F_{VCO} frequency.
      - Select a feedback divider value of M = 100.
      - $F_{VCO} = F_{PLLI}$ x (M/N1) = 10 MHz x (100/1) = 1 GHz
   c. Select values for the first and second PLL postscalers to achieve the required FPLLO frequency.
      - Select values for the first and second postscalers of N2 = 5 and N3 = 1.
      - FPLL F_{OUT}/FPLLO = F_{VCO}/(N2 x N3) = 1 GHz/5 = 200 MHz
3. Writing PLLxCON (can be done in one PLLxCON write):
   a. Enable the PLL input and Feedback Divider update by setting the PLLSWEN bit in the PLLxCON register. Enable the PLL Output Divider update by setting the FOUTSWEN bit in the PLLxCON register.
   b. Select clock source by setting the NOSC[3:0] bits in the PLLxCON register to Primary (3).

c. Enable clock switching by setting the OSWEN bit in the PLLxCON register.

Example 12-2 illustrates code for using the PLL with the POSC.
Note: PLL1CON writes are provided for clarity. These writes can be achieved in a single PLL1CON write.

Example 12-2. Code Example for Using PLL with the Primary Oscillator (POSC)

```
// code example for 200 MHz PLL clock using 10 MHz crystal primary oscillator
//set crystal configuration using configuration register
OSCCFGbits.POSCMD = 2;              //set desired Primary oscillator mode
// 0b00 : EC mode
// 0b01-0b10 : XT mode
// 0b11 : Disabled
//Enable PLL clock generator
PLL1CONbits.ON = 1;
PLL1CONbits.OE = 1;
//configure backup oscillator in case of failure
PLL1CONbits.BOSC = 2;              //select BFRC backup clock source:
PLL1CONbits.FSCMEN = 1;            //enable fail safe
//configure PLL values
PLL1DIVbits.PLLFBDIV = 100;        //Feedback Divider
PLL1DIVbits.PLLPRE = 1;            //Reference Clock Divider
PLL1DIVbits.POSTDIV1 = 5;          //Post Divider #1
PLL1DIVbits.POSTDIV2 = 1;          //Post Divider #2
// PLL Fout = Fin*FBDIV / (PLLPRE * POSTDIV1 * POSTDIV2)
// PLL Fout = 10M*100/(1*5*1) =200MHz
//Enable PLL Input and Feedback Divider update
PLL1CONbits.PLLSWEN = 1;
while (PLL1CONbits.PLLSWEN == 1);
//Enable PLL Output Divider update
PLL1CONbits.FOUTSWEN = 1;
while (PLL1CONbits.FOUTSWEN);
//select clock switching clock source
PLL1CONbits.NOSC = 3;              //select POSC clock source
//[1] = FRC - Internal 8 MHz RC oscillator
//[2] = BFRC - Internal Backup 8 MHz RC oscillator
//[3] = POSC - Primary crystal oscillator (4-32 MHz)
//[9] = REFI1 - user definable clock source
//[10] = REFI2 - user definable clock source
PLL1CONbits.OSWEN = 1;              //enable clock switching
while (PLL1CONbits.OSWEN);          //wait for switching(hardware clear)
while(!OSCCTRLbits.PLL1RDY);        //wait for clock to be ready
```

12.4.6.3.2. Setup for Using PLL with 8 MHz Internal FRC

The following process is used to set up the PLL to operate the device at 100 MHz with an 8 MHz Internal FRC and configured PLL VCO output to 250 MHz.

1. To execute instructions at 100 MHz, a PLL output frequency of 100 MHz will be required.
2. To set up the PLL and meet the requirements of the PLL, follow these steps:
   a. Select the PLL prescaler to meet the PFD input frequency requirement.
      - Select a PLL prescaler value of N1 = 1.
      - F_{PLLI} = 8 MHz
      - F_{PFD} = 8 MHz(1/N1) = 8 MHz(1) = 8 MHz
   b. Select the feedback divider to meet the VCO output frequency requirement as well as achieve the desired F_{VCO} frequency.
      - Select a feedback divider value of M = 125.
      - F_{VCO} = F_{PLLI} x (M/N1) = 8 MHz x (125/1) = 1 GHz
   c. Select values for the first and second PLL postscalers to achieve the required F_{PLLO} frequency.
      - Select values for the first and second postscalers of N2 = 5 and N3 = 2.
      - F_{PLLO} = F_{VCO}/(N2 x N3) = 1 GHz/10 = 100 MHz

3. Writing PLLxCON (can be done in one PLLxCON write):
   a. Enable the PLL Input and Feedback Divider update by setting PLLSWEN bit in the PLLxCON register. Enable the PLL Output Divider update by setting FOUTSWEN bit in the PLLxCON register.
   b. Select a clock source by setting NOSC[3:0] bits in the PLLxCON register to FRC (1).
   c. Enable clock switching by setting OSWEN bit in the PLLxCON register.
4. PLL VCO output setup is optional for a normal PLL output. PLL VCODIV output is necessary if PLL VCODIV will be chosen as a CLKGEN input. For a PLL VCO output, the PLLxDIV register needs to be configured. INTDIV bits of the PLLxDIV register are configured as two.
   – FVCO / INTDIV*2 = 1 GHz/ 2*2 =250 MHz

Example 12-3 illustrates code for using the PLL with an 8 MHz Internal FRC Oscillator.
Note: PLL1CON writes written out for clarity. These writes can be achieved in a single PLL1CON write.

Example 12-3. Code Example for Using PLL with 8 MHz Internal FRC

```
// code example for 100 MHz PLL clock using 10 MHz crystal primary oscillator
//configure backup oscillator in case of failure
//Enable PLL clock generator
PLL1CONbits.ON = 1;
PLL1CONbits.OE = 1;
PLL1CONbits.BOSC = 2;             //BFRC as backup clock source
PLL1CONbits.FSCMEN = 1;           //enable fail safe
//configure PLL values
PLL1DIVbits.PLLPRE = 1;           //Reference Clock Divider
PLL1DIVbits.POSTDIV1 = 5;         //Post Divider #1
PLL1DIVbits.POSTDIV2 = 2;         //Post Divider #2
// PLL Fout = Fin*FBDIV / (PLLPRE * POSTDIV1 * POSTDIV2)
// PLL Fout = 8M*125/(1*5*2) =100MHz
//Enable PLL Input and Feedback Divider update
PLL1CONbits.PLLSWEN = 1;
while (PLL1CONbits.PLLSWEN == 1);
//Enable PLL Output Divider update
PLL1CONbits.FOUTSWEN = 1;
while (PLL1CONbits.FOUTSWEN);
//select clock switching clock source
PLL1CONbits.NOSC = 1;             //select FRC clock source
//[1] = FRC - Internal 8 MHz RC oscillator
//[2] = BFRC - Internal Backup 8 MHz RC oscillator
//[3] = POSC - Primary crystal oscillator (4-32 MHz)
//[9] = REFI1 - user definable clock source
//[10] = REFI2 - user definable clock source
PLL1CONbits.OSWEN = 1;          //enable clock switching
while (PLL1CONbits.OSWEN);      //wait for switching(hardware clear)
while(!OSCCTRLbits.PLL1RDY)    //wait for clock to be ready
//configure PLL VCO Divider registers for PLL1 VCO Div clock source
VCO1DIVbits.INTDIV = 2;         //integer divide factor
// PLL VCO DIV = PLL VCO clock / 2* INTDIV
//Enable PLL VCO Divider update
PLL1CONbits.DIVSWEN = 1;
while (PLL1CONbits.DIVSWEN);
```

12.4.6.3.3. Internal Low-Power RC (LPRC) Oscillator

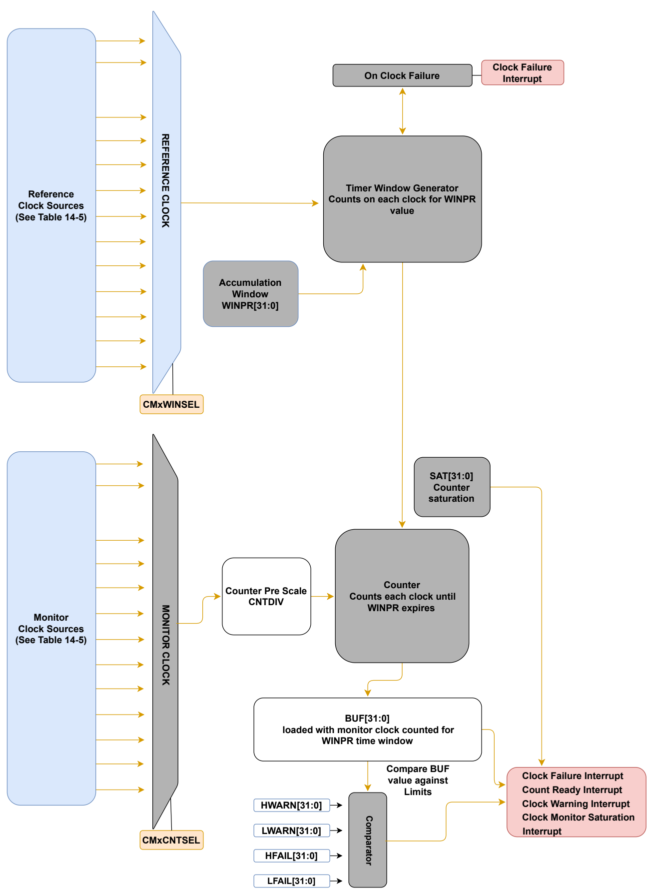
The dsPIC33A devices contain one instance of the Internal LPRC oscillator. The LPRC oscillator is implemented as the BFRC is divided by 244 to yield a clock frequency of 32.768 kHz.

12.4.7. Clock Monitor

The clock monitor's purpose is to detect system clock failures or anomalies and allow the system to take action. The clock monitor uses a second clock (reference) source to compare to the clock being monitored. Clock monitors are assigned to a clock source such as a CLKGEN, PLL output, or other system clocks including FRC. The clock monitor provides the following features:

- Configurable Clock Sources:
  - Monitored clock sources
  - Reference clock sources
- Flexible Detection Capability
- Clock Frequency Drift Detection
- Selectable Threshold Limits for Warning and/or Failing
- Monitored Clock Catastrophic Detection
- Reference Clock Catastrophic Detection
- User Optimizable Accuracy via:
  - Accumulation time adjustment
  - Selectable clock sources
- Register Write Protection
- Fault Injection Capability
- Frequency Measurement
- Pulse Width and Duty Cycle Measurement

Figure 12-10. Clock Monitor Architecture



12.4.7.1. Clock Monitor Overview

The clock monitor is primarily used for constant clock monitoring in real time with the following added features:

- Monitored Clock Frequency Drift Detection
- Under/Over Clocking Warning Detection
- Under/Over Clocking Failing Detection
- Monitored Clock Catastrophic Detection
- Reference Clock Detection
- Unable to Start
- Failure After Normal Start

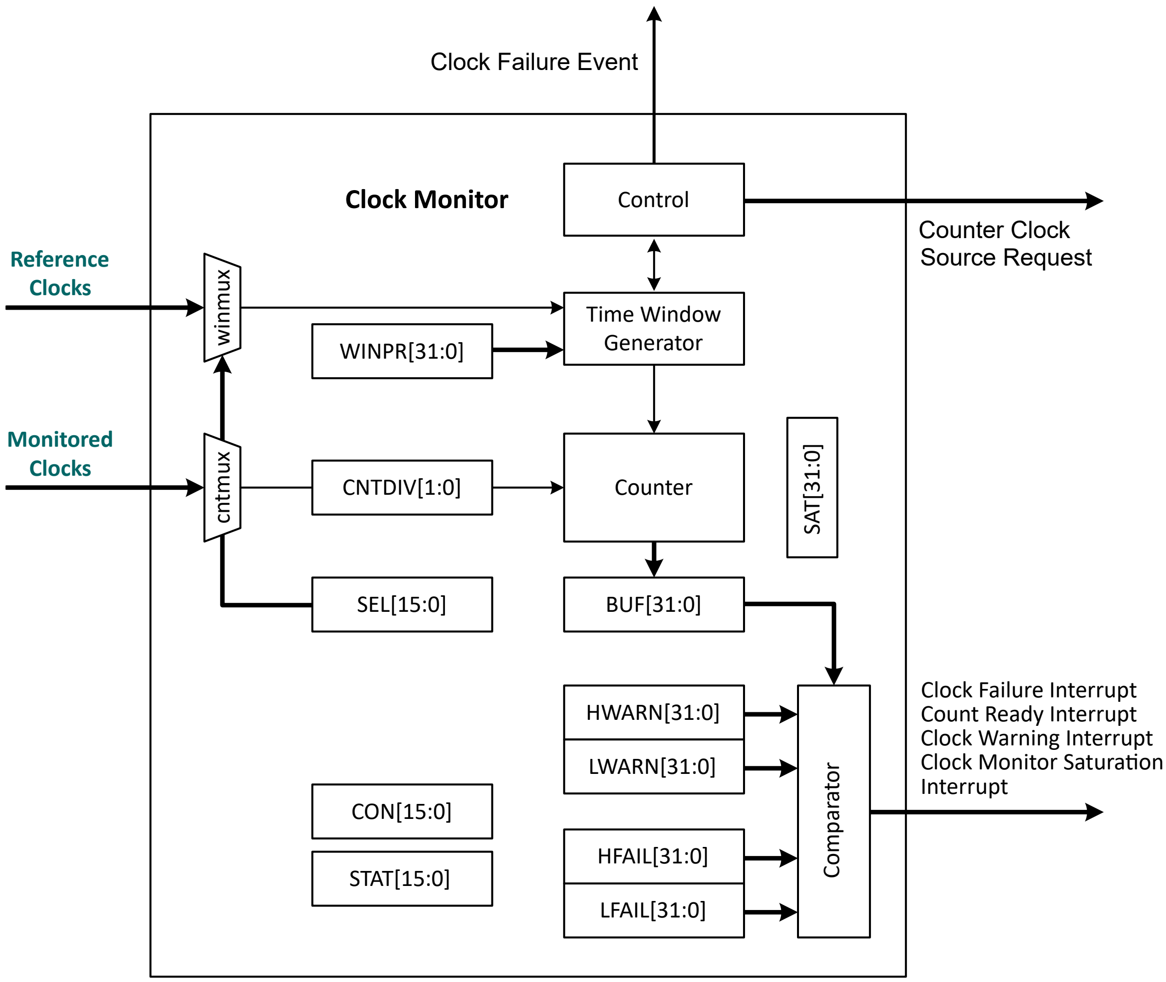
Clock monitoring operates on two different clock sources to accomplish its task, both of which are user selectable from various different clock inputs. Refer to WINSEL and CNTSEL within CMxSEL.

WINSEL selects a reference clock, and CNTSEL selects a monitored clock. The reference clock is used to generate a user-programmable time window defining a known accumulation time period, specified by WINPR[31:0] bits in the CMxWINPR register, over which the monitored clock is allowed to advance the internal counter, thus accumulating its count value.

As each accumulation time period ends, the monitored clock count value is captured into a separate Data Buffer register, see BUF[31:0] in register CMxBUF, while the new accumulation time period begins with a cleared accumulator.

Simultaneously, the ClkxReadyInterrupt is invoked to notify software of the new BUF[31:0] capture contents. The buffer contents are then evaluated by a digital comparator, where various detections are made based on the set and user-defined criteria. Four independent threshold limit registers are employed to allow the user to define the acceptable ranges of the monitored clock frequency. The process is repeated until the module is disabled.

Figure 12-11. Monitor Function

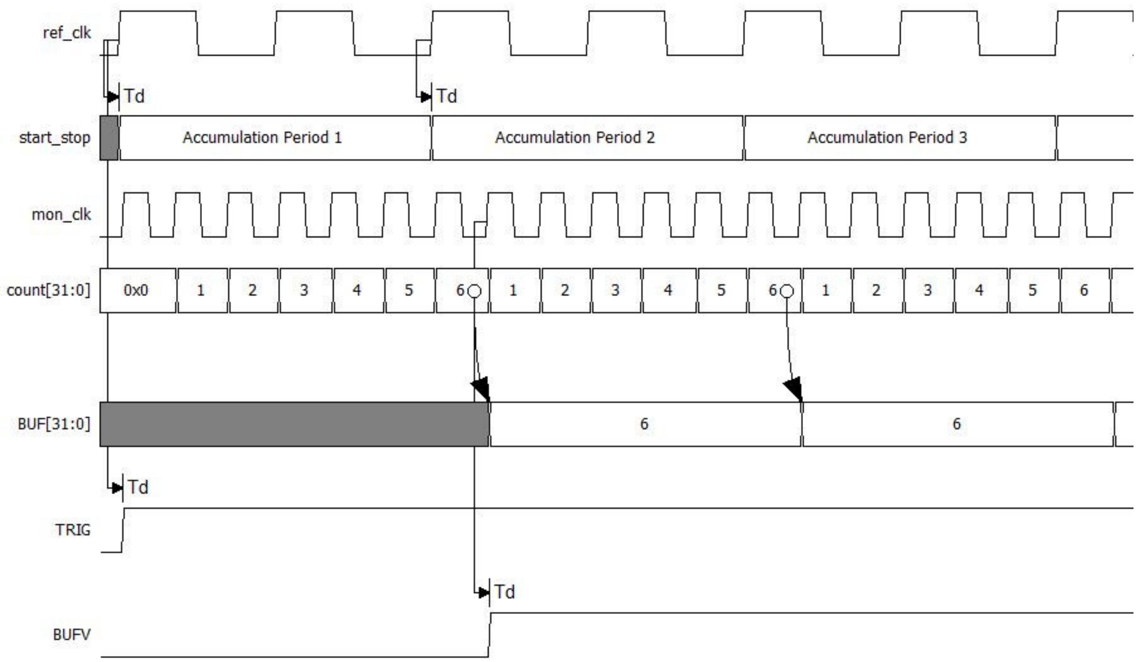


Based on the user selected timings on the reference clock, the start and stop points are chosen to define the desirable time window to accumulate the count value of the counter. The start time initializes the counter to zero and begins adding monitored clock rising edges to itself, whereas the stop time initiates transfer of the accumulated count into the BUF[31:0] register for use by the comparators and software.

Notes:

1. Longer time windows provide more available time for the counter to accumulate monitored clocks, resulting in higher precision.
2. Choosing the source of the reference clock wisely can significantly improve the performance of the module. If monitored clock frequency is significantly higher than that of the reference clock or frequencies are close to each other, then performance may be affected.

Figure 12-12. Internal Timing Diagram – WINPR[31:0] = 1



Note: Synchronization is intended for illustration only and may not be drawn to scale.

Note:
Upon successful count value capturing into the Data Buffer register:

- Count ready interrupt output based on the capturing of the counter invoked
- BUFV set high

12.4.7.2. Detection on Monitored Clock

The clock monitor can detect faulty conditions of both of the clock sources being monitored, and it can also be used as the reference.

12.4.7.2.1. Frequency Drift Detection

The captured count value enters the comparator logic block, and its value is simultaneously compared against the contents of four limit registers. This essentially forms four independent thresholds confining the monitored clock frequency's deviation into two acceptable ranges defined by two pairs of high/low limit registers. Monitored clock frequency drift is, therefore, detectable per user-defined tolerance. Upon detection of a threshold violation, the user is notified via an interrupt event.

- The clock failure interrupt output is based on the fail threshold limit and catastrophic failures invoked.
- The ON bit is cleared.
- The Clock Fail Event signal is provided for the system.

12.4.7.2.2. Catastrophic Failure Detection

Monitored Clock Failure

The monitored clock experiences a catastrophic failure and decreases its toggling rate significantly or ceases to toggle completely.

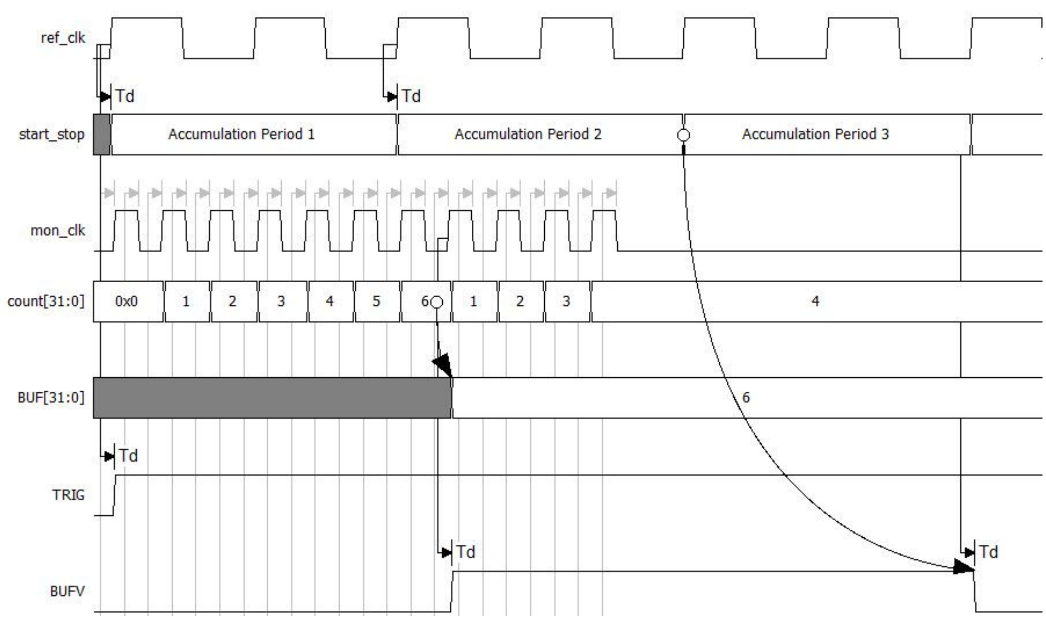
The total latency of the monitored clock frequency drift detection does not only depend on the duration of the accumulation time window on which it operates, but it also depends on the latency of its clock domain crossing logic.

Detection happens on timely updates of the Data Buffer register with the captured count as the periodic accumulation time expires. At the next accumulation time expiration, if BUF[31:0] has not been updated, the monitored clock is considered to have experienced a catastrophic failure event given that the reference clock is still toggling reliably.

The module's response is similar to that of the clock frequency drifting beyond the user-defined catastrophic tolerance limit described above. By asserting a clock fail event, the clock monitor module provides the system with a fast hardware response time often required to deal with a clock failure in control loop applications. The clock fail event triggers the clock fail interrupt with the conditions reflected below:

- Clock failure interrupt output is based on fail threshold limit and catastrophic failures invoked.
- ON bit is cleared.
- Clock Fail Event signal is provided for system.

Figure 12-13. Catastrophic Failure Detection on Monitored Clock — Missing Edges on MON_CLK



Reference Clock Failure

The clock monitor module can detect reference clock failure; the reference clock can fail under the following conditions:

1. If the selected reference clock is completely missing from the start, making the signaling event representing the start of the accumulation time window absent, the TRIG bit remains clear as a result. This condition can be captured by the user software's observing the logic value of this bit once the system has booted up - no interrupt and/or other signaling event generated.
2. However, if the reference clock starts out toggling, setting the TRIG bit high, but subsequently slows down significantly or ceases to toggle completely at some point, the signaling event representing the start/stop of the accumulation time window is also affected by the same proportion. This condition effectively extends the accumulation time window causing the

counter to eventually saturate before capturing in some cases. As a result, the SATD bit is set high - interrupt invoked.

Note: SATD being set high does not exclusively reflect this condition. Also, the reference clock is required to set SATD.

12.4.7.3. Measurement Function

The clock monitor module is also capable of functioning as a frequency measurement unit, usually when employed as a stand-alone peripheral.

12.4.7.3.1. Software Assisted Frequency Measurement

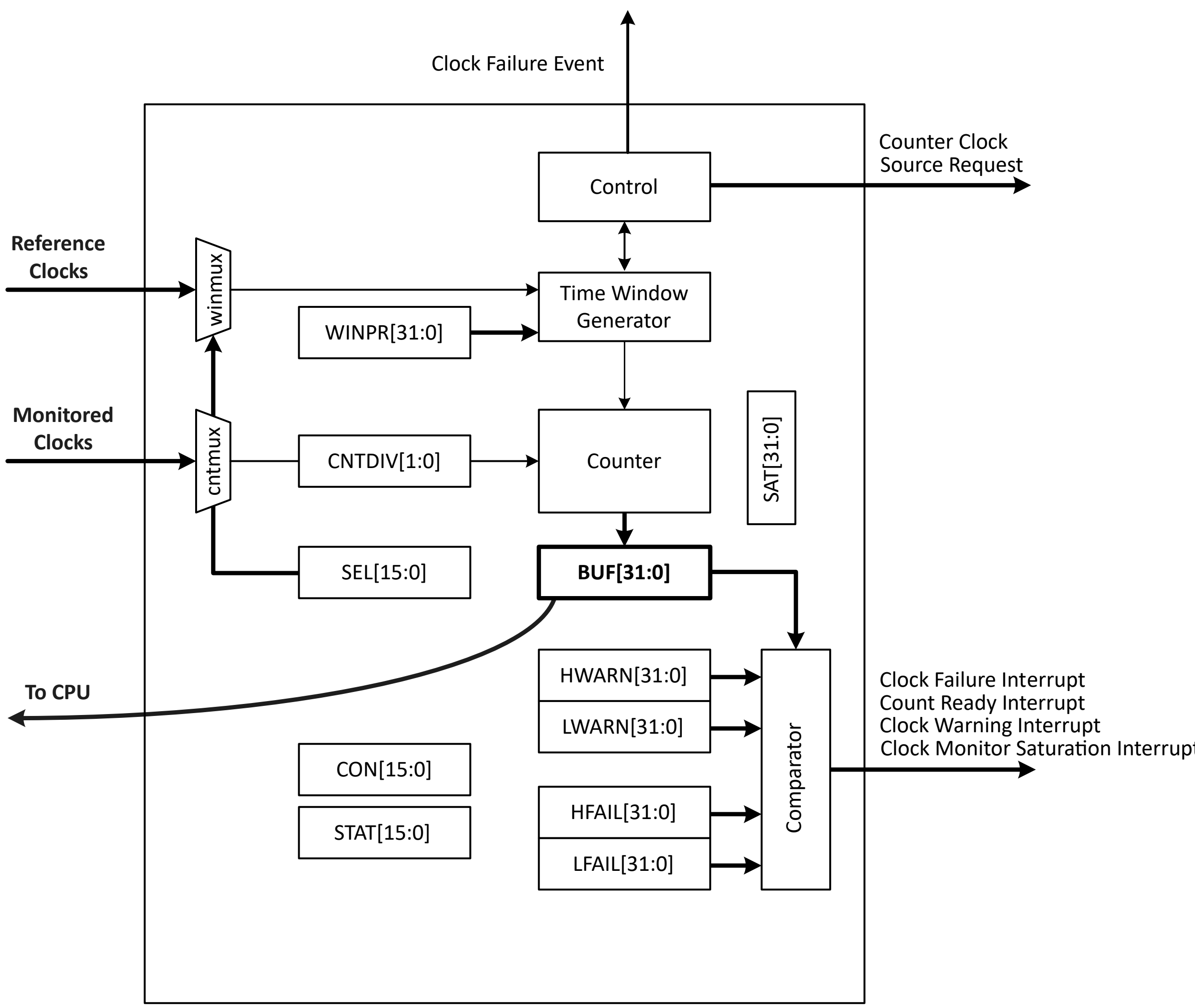
By allowing the CPU access to the accumulated results, firmware can convert the measured counts into frequencies based on existing knowledge of the reference clock frequency or period. The captured count's corresponding register, CMxBUF, is therefore memory-mapped in this peripheral. Figure 12-14 shows the basic block diagram of this mode of operation.

Clocking Configurations

Reference Clock ≥ Time Window Generator

Monitored Clock ≥ Counter

Figure 12-14. Frequency Measurement Function – CPU



If the reference clock is Clock Generator 6 (FRC 8 MHz), the monitor clock is Clock Generator 5 (EC 8 MHz).

Accumulation time = 0x8000 cycles of the reference clock.

The monitor clock is divided by 2, making it 4 MHz.

Time window = 0x8000/8M = 4 ms.

So, for 4ms with 4MHz clock speed, what is the expected accumulation?

0.004096 * 4M = 0x4000

The CMxBUF is expected to hold the value 0x4000 at every time window interval.

If the reference clock is Clock Generator 6 (FRC 8 MHz) and the monitor clock is Clock Generator 5 (EC 8 MHz):

Accumulation Time = 0x8000 cycles of the reference clock.

The monitor clock is divided by 2; making it 4 MHz.

Time window = 0x8000/8M = 4ms

So, for 4ms with a 4MHz clock speed, what is the expected accumulation?

0.004096 * 4M = 0x4000

The CMxBUF is expected to hold the value 0x4000 at every time window interval.

Example 12-4. Frequency Measurement Function

```
//Configure clock generators before enabling monitoring

   IFS0bits.C2FAILIF = 0;     // enable clock Monitor2 failure interrupt
   IEC0bits.C2FAILIE = 1;

   IFS0bits.C2MONIF = 0;     // enable clock saturation failure interrupt
   IEC0bits.C2MONIE = 1;

   IFS0bits.C2WARMIF = 0;     // enable clock Warning failure interrupt
   IEC0bits.C2WARMIE = 1;

   IFS0bits.C2RDYIF = 0;     // enable clock ready  failure interrupt
   IEC0bits.C2RDYIE = 1;

   CM2CONbits.ON = 0;        // disable clock monitor

   //Select monitor clock source
   CM2SELbits.CNTSEL = 4;    //Monitoring clock clock-gen-5
   //[0] = clkgen_clk[1] = System clocks
   //[1] = clkgen_clk[2]
   //...
   //[10] = clkgen_clk[11]
   //[11] = pll_fout_clk[1]
   //[12] = pll_vcodiv_clk[1]
   //[13] = pll_fout_clk[2]
   //[14] = pll_vcodiv_clk[2]
   //[15] = 1?b0 (reserved)
   //[16] = ICSP clock (PGC)
   //[17] = FRC - Internal 8 MHz RC oscillator
   //[18] = BFRC - Internal Backup 8 MHz RC oscillator
   //[19] = POSC - Primary crystal oscillator (4-32 MHz)
   //[25] = REFI1 - user definable clock source
   //[26] = REFI2 - user definable clock source

   //Select reference clock source
   CM2SELbits.WINSEL = 5; //reference clock clock-gen-6
   //[0] = clkgen_clk[1] = System clocks
   //[1] = clkgen_clk[2]
   //...
   //[10] = clkgen_clk[11]
   //[11] = pll_fout_clk[1]
   //[12] = pll_vcodiv_clk[1]
```

```
    //[13] = pll_fout_clk[2]
    //[14] = pll_vcodiv_clk[2]
    //[15] = 1?b0 (reserved)
    //[16] = ICSP clock (PGC)
    //[17] = FRC - Internal 8 MHz RC oscillator
    //[18] = BFRC - Internal Backup 8 MHz RC oscillator
    //[19] = POSC - Primary crystal oscillator (4-32 MHz)
    //[25] = REFI1 - user definable clock source
    //[26] = REFI2 - user definable clock source


    //Counter Divider Selection
    CM2CONbits.CNTDIV = 1;
    //    10 = Divide-by 4
    //    01 = Divide-by 2
    //    00 = Divide-by 1

    CM2WINPR = 0x8000; //monitor clock pre scale
    //Accumulation Time (in cycles) = WINPR[31:0] + 1

    CM2HFAIL = 0x4500;      //CLOCK MONITOR HIGH THRESHOLD FAILING
    CM2LFAIL = 0x3500;      //CLOCK MONITOR LOW THRESHOLD FAILING

    CM2HWARN = 0x4100;      //CLOCK MONITOR HIGH THRESHOLD WARNING
    CM2LWARN = 0x3600;      //CLOCK MONITOR LOW THRESHOLD WARNING

    CM2SAT = 0x5000;       //CLOCK MONITOR COUNTER SATURATION


    CM2CONbits.WIDTH = 0;    // 0 - Frequency Measurement
                             // 1 - Pulse Width and Duty Cycle Measurement

    CM2CONbits.ON = 1;      //enable clock monitor
    while (CM2CONbits.ON);
```

12.4.7.3.2. Pulse Width and Duty Cycle Measurement

The clock monitored module is also capable of measuring the pulse width of the monitored clock source. By allowing only the high pulse time period of the monitored clock source to define the accumulation time window, see WIDTH, the pulse width can be measured by the number of reference clock cycles. It is given that the reference clock frequency is higher than the monitored clock frequency, so the operation is similar to that of the CCP module.

Clocking Configurations:

Monitored Clock ≥ Time Window Generator

Reference Clock ≥ Counter

In pulse measurement, give a slower clock to a time window (reference clock) and a faster clock to a counter (monitor clock).

There is no accumulation window (WINPR= 0). The rising edge to the next falling edge of the reference clock will be the window for CMxBUF to get loaded.

Example 12-5 shows LPRC as the reference clock and FRC as the monitor clock.

Each rising edge to the next falling edge of LPRC (32KHz) will be 15.625 μs.

15.625 μs * 8M = 0x007D is the expected count in the CMxBUF register.

Similarly, clock period measurement can also be accomplished by allowing the time period of the monitored clock source to define the accumulation time window.

With the clock setup given above, the pulse width or clock period represented by the number of reference clock cycles and captured at every accumulation lapsed time can simply be read out of the Data Buffer register, BUF[31:0], for further processing as follows:

- Conversion into unit of time
- Duty cycle calculation (with period measurement)

Table 12-9. Module Control

Function	Clock Source		Control
	win_clk[*:0]	cnt_clk[*:0]	ON/WIDTH
Monitor	reference	monitored	1/0
Measurement - SW	reference	reference	1/0
Measurement - PW	monitored	reference	1/1

Example 12-5. Pulse Width

```
IFS0bits.C2FAILIF = 0;     // enable clock Monitor2 failure interrupt
    IEC0bits.C2FAILIE = 1;

    IFS0bits.C2MONIF = 0;          // enable clock saturation failure interrupt
    IEC0bits.C2MONIE = 1;

    IFS0bits.C2WARMIF = 0;         // enable clock Warning failure interrupt
    IEC0bits.C2WARMIE = 1;

    IFS0bits.C2RDYIF = 0;         // enable clock ready  failure interrupt
    IEC0bits.C2RDYIE = 1;

    CM2CONbits.ON = 0;            // disable clock monitor

    //Select monitor clock source
    CM2SELbits.CNTSEL = 4;        // Monitoring clock clock gen 5

    //Select reference clock source
    CM2SELbits.WINSEL = 20; //reference clock LPRC

    //Counter Divider Selection
    CM2CONbits.CNTDIV = 0;
    //    10 = Divide-by 4
    //    01 = Divide-by 2
    //    00 = Divide-by 1

    CM2WINPR = 0x0000; //monitor clock pre scale
    //Accumulation Time (in cycles) = WINPR[31:0] + 1

    CM2HFAIL = 0x90;         //CLOCK MONITOR HIGH THRESHOLD FAILING
    CM2LFAIL = 0x70;         //CLOCK MONITOR LOW THRESHOLD FAILING

    CM2HWARN = 0x85;         //CLOCK MONITOR HIGH THRESHOLD WARNING
    CM2LWARN = 0x75;         //CLOCK MONITOR LOW THRESHOLD WARNING

    CM2SAT = 0x100;         //CLOCK MONITOR COUNTER SATURATION


    CM2CONbits.WIDTH = 1;   // 0 - Frequency Measurement
    // 1 - Pulse Width and Duty Cycle Measurement

    CM2CONbits.ON = 1;     //enable clock monitor
    while (CM2CONbits.ON);
```

12.4.7.4. Write Protection

To avoid erroneous software, write which impact result of the module has false detection.

The LOCK and WREN are register bits located in the module to control the register write.

Table 12-10. Write Access Control

LOCK	WREN	Access Permission
0	0	Registers write access disabled[1]
0	1	Registers write access enabled[2]

Notes:
1. Access restriction enforced until Reset.
2. Default setting

Table 12-10. Write Access Control (continued)

LOCK	WREN	Access Permission
1	x	Registers write access disabled(1)

Notes:
1. Access restriction enforced until Reset.
2. Default setting

12.4.7.5. Fault Injection

The Fault injection feature allows the user to inject an artificial Fault into the module and observe the expected response to ensure the module's proper operation. To be effective, the Fault injection is required to be performed randomly and periodically with minimal interference to the module's normal operation. The module is not disabled upon detection of any Fault injection.

12.4.7.5.1. Catastrophic Fault Injection

Catastrophic Fault injection is used to confirm that the module is capable of detecting the condition where the monitored clock completely ceases to toggle, also known as catastrophic failure, as discussed earlier.

When the artificial catastrophic Fault is injected into the module, the module's counter no longer gets clocked by the selected monitored clock. It is internally driven to ground via hardware. As a result, the counter is unable to continue accumulating, and the current count value stalls inside the counter when the function is invoked. Without a valid clock edge, the counter is unable to transfer its accumulated count into the Data Buffer register at the end of the accumulation time window.

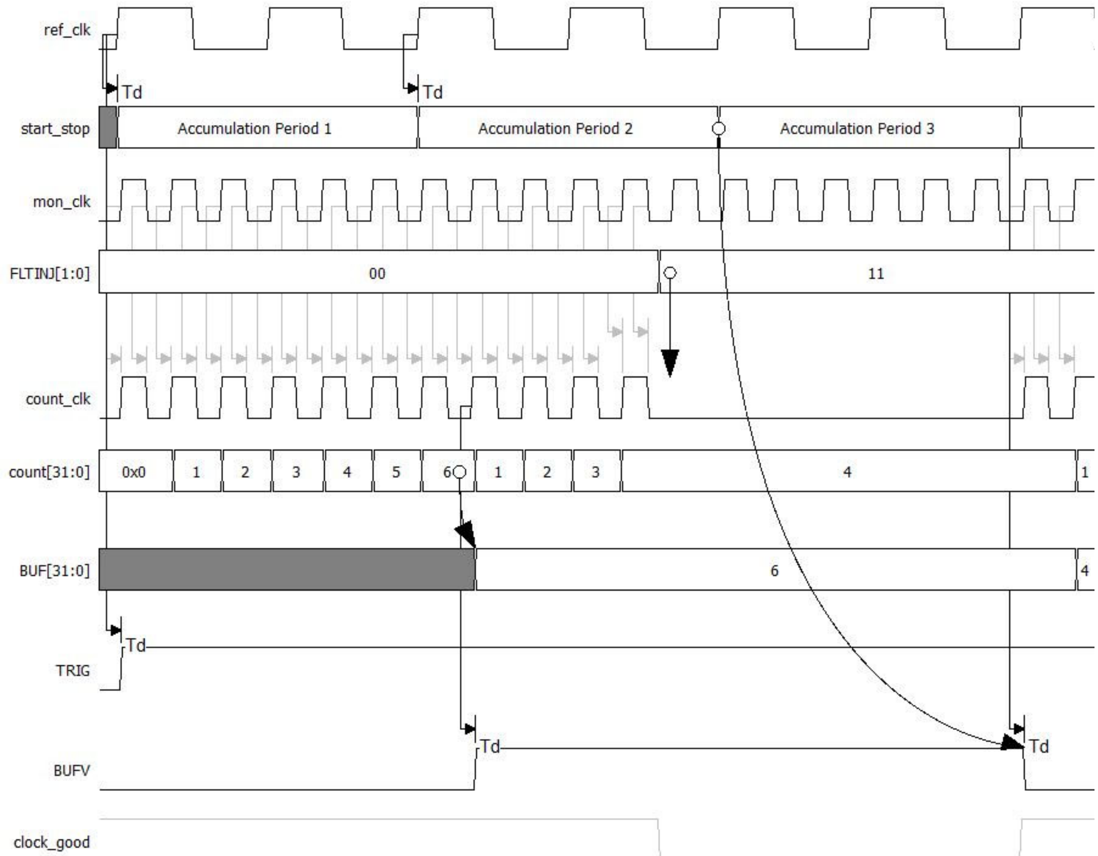
Catastrophic Fault injection can be activated by setting FLTINJ[1:0] to 0b11 at an arbitrary point, subject to LOCK/WREN.

The module's response to catastrophic Fault injection:

- Clock failure interrupt output is based on fail threshold limit and catastrophic failures invoked.
- ON bit is NOT cleared.
- Clock Fail Event signal is NOT provided for the system.

Notes:
1. FLTINJ[1:0] bits are not self-cleared by hardware. They maintain their programmed value until cleared by software to assist the ISR handler with its discovery process.
2. A persisting real Fault is detected in the following accumulation cycle regardless of FLTINJ[1:0] bits being cleared.

Figure 12-15. Catastrophic Fault Injection



12.4.7.5.2. Low-Frequency Drift Fault Injection

To mimic low-frequency drift, the selected monitored clock input is purposely divided down by two before being allowed to clock the counter, resulting in lowering the final accumulation value to half the expected figure.

The count value, once captured, is then compared with the contents of the four threshold limit registers to determine its current deviation against the set criteria.

As expected,

- If BUF[31:0] < LWARN[31:0], the LWT interrupt flag is set high with a clock warning interrupt output based on the warning threshold limit and active high invoked.
- If BUF[31:0] < LFAIL[31:0], the LFT interrupt flag is also set high with a clock failure interrupt output based on the fail threshold limit and catastrophic failures invoked.

A low-frequency drift Fault injection can be activated by setting FLTINJ[1:0] to 1 at random, subject to LOCK/WREN.

Notes:

1. FLTINJ[1:0] bits are not self-cleared by hardware. They maintain their programmed value until cleared by software to assist the ISR handler with its discovery process.
2. Detection of this artificial Fault will not occur if the limit tolerances are set too high. The function is for the detection of 50% margin below nominal or tighter.

12.4.7.5.3. High-Frequency Drift Fault Injection

To mimic high-frequency drift, the selected reference clock input is purposely divided down by two before entering the time window generator, thus allowing twice as much time for the counter to accumulate its count and double the expected figure.

The count value, once captured, is then compared against the contents of the four threshold limit registers to determine its current deviation against the set criteria. As expected:

- If BUF[31:0] > HWARN[31:0], the HWT interrupt flag is set high with a clock warning interrupt output based on the warning threshold limit invoked.
- If BUF[31:0] > HFAIL[31:0], the HFT interrupt flag is also set high with a clock failure interrupt output based on the fail threshold limit and catastrophic failures invoked.
- If the counter saturates before the current accumulation time period ends, SATD status bit is set high with a clock monitor saturation interrupt output based on the counter having saturated.

A high-frequency drift Fault injection can be activated by setting FLTINJ[1:0] to 2 at random, subject to LOCK/WREN.

Notes:

1. FLTINJ[1:0] bits are not self-cleared by hardware. They maintain their programmed value until cleared by software to assist the ISR handler with its discovery process.
2. Detection of this artificial Fault will not occur if the limit tolerances are set too high. The function is for the detection of 100% margin above nominal or tighter.

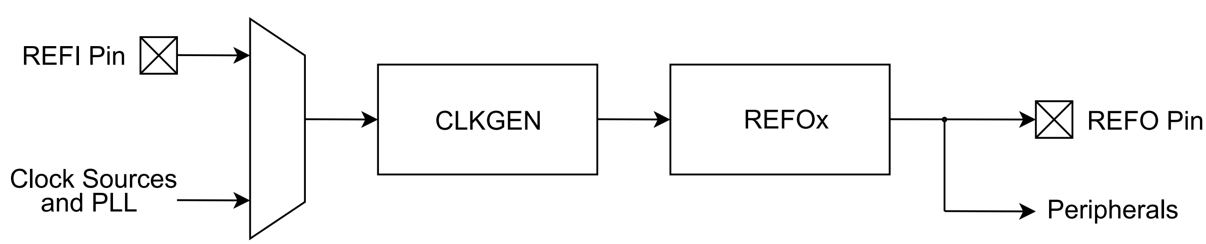
Table 12-11. Module Control

Fault	Clock Connection		Control
	win_clk[*:0]	cnt_clk[*:0]	ON/WIDTH/FLTINJ[1:0]
Catastrophic Fault Injection	reference	monitored	1/0/11
Low Drift Fault Injection	reference	monitored	1/0/01
High Drift Fault Injection	reference	monitored	1/0/10

12.4.8. Reference Clock Output (REFOx)

The dsPIC33AK256MPS306 family features one or more reference clock output (REFOx) modules. The REFOx module clock source is one of the CLKGENs (see Table 12-2 for assignments of CLKGENs to REFOx). The input clock to the REFOx is, therefore, the clock source of the associated CLKGENx, which can be any of the sources defined in Table 12-2. This selection includes reference clock inputs (REFIx) that are mappable to device pins using Peripheral Pin Select (PPS). The REFO output signal is internally routed to select peripherals and can be routed to a device pin using PPS. Figure 12-16 illustrates REFO interconnections.

Figure 12-16. CLKGEN to REFO Assignments



12.4.9. Power-Saving Mode

The Oscillator module supports multiple Sleep and Idle modes for power reduction:

For Clock Generator 1:

- Idle mode - System clock stops, peripheral clock, peripheral clock divided run
- Sleep mode - System clock, peripheral clock, peripheral clock divided stop

Other clock and PLL generators:

- Stop in Idle - CLKGEN/PLL(x) stop in Idle
- Run in Sleep - CLKGEN/PLL(x) run in Sleep

The Stop in Idle (SIDL) and Run In Sleep (RIS) options are user selectable via the SIDLE and RIS bits in the CLKxCON/PLLCONx registers.

If the SIDLE bit is set in a CLKxCON/PLLCONx register, the associated clock will stop if CPU IDLE is asserted. The SIDLE bit has no effect if CPU Sleep is asserted.

If the RIS bit is set in a CLKxCON/PLLCONx register, the associated clock will continue to run even if CPU Sleep is asserted. The RIS bit has no effect if CPU Idle is asserted.

Clock Generator 1 is the clock source for the system clock (sys_clk) and peripheral clock. The PWRSAV instruction that initiates Idle and Sleep modes is primarily responsible for reducing processor and system power consumption.

In Sleep and Idle modes, the processor clock, system and fast peripheral clock, and peripheral clock are all disabled.

13. Direct Memory Access (DMA) Controller

The Direct Memory Access (DMA) controller handles high data throughput peripherals on the SFR bus by enabling direct access to data memory to reduce the need for intensive CPU management. The DMA controller is structured with multiple channels, each of which can be connected to a selectable peripheral module. When a peripheral module triggers its interrupt, the corresponding DMA channel responds by accessing the SRAM without requiring CPU intervention. This direct access not only frees up the CPU to handle other tasks but also optimizes overall system efficiency. Each DMA channel can interrupt the CPU once the DMA session is complete or if other interrupt conditions are met. This mechanism ensures that the CPU is only engaged when necessary, enhancing system performance and making efficient use of CPU resources. By offloading the data transfer tasks from the CPU, the DMA controller significantly contributes to improved system functionality and reduces power consumption in various applications.

The DMA Controller has these features:

- Eight Independent Channels
- Concurrent Operation with the CPU
- DMA Bus Arbitration Using Fixed Priority and Round Robin Scheme
- Four Address Modes
- Four Transfer Modes
- Ping-Pong Mode
- 8-Bit, 16-Bit or 32-Bit Word Support for Data Transfer
- 24-Bit Source and Destination Address Register for Each Channel, Dynamically Updated and Independently Reloadable
- 32-Bit Transaction Count Register, Dynamically Updated and Independently Reloadable
- Upper and Lower Address Limit Registers
- Counter Half/Full Level Interrupt
- Software Triggered Transfer
- Null Write Mode for Symmetric Buffer Operations
- Fixed Priority and Round Robin Channel Arbitration
- DMA Request for Each Channel can be Selected from Any Supported Interrupt Source
- Support for Daisy-Chaining of Channels (One Channel Triggered by Another Channel)
- Set/Clear/Invert Bit Manipulation Capability
- Pattern Match
- Bus Read/Write Error Fault Indication
- Descriptor-Based Operational Capability

13.1. Device-Specific Information

Table 13-1. DMA Summary Table

DMA Module Instances	Channels per Instance	Clock Source	Peripheral Bus Speed
1	8	Standard (1:2 CPU Clock)	Standard

Table 13-2. DMA Channel Trigger Sources

CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)
00h	INT0 - External Interrupt 0	26h	CAM TRNG Interrupt	4Ch	CLC1 Rising Edge

Table 13-2. DMA Channel Trigger Sources (continued)

CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)
01h	INT1 - External Interrupt 1	27h	CAM PubKey Interrupt	4Dh	CLC1 Falling Edge
02h	INT2 - External Interrupt 2	28h	PTG Interrupt 0	4Eh	CLC2 Rising Edge
03h	NVM - NVM Write Complete	29h	PTG Interrupt 1	4Fh	CLC2 Falling Edge
04h	CRC - CRC Generator Interrupt	2Ah	PTG Interrupt 2	50h	CLC3 Rising Edge
05h	TMR1 - Timer 1 Interrupt	2Bh	PTG Interrupt 3	51h	CLC3 Falling Edge
06h	SPI1Rx - SPI 1 Receiver	2Ch	SENT1	52h	CLC4 Rising Edge
07h	SPI1Tx - SPI 1 Transmitter	2Dh	SENT2	53h	CLC4 Falling Edge
08h	SPI2Rx - SPI 2 Receiver	2Eh	BiSS	54h	I2C1 - Generic Interrupt
09h	SPI2Tx - SPI 2 Transmitter	2Fh	ADC1 - AN0	55h	I2C1 - Receive Buffer Interrupt
0Ah	SPI3Rx - SPI 3 Receiver	30h	ADC1 - AN1	56h	I2C1 - Transmit Buffer Interrupt
0Bh	SPI3Tx - SPI 3 Transmitter	31h	ADC1 - AN2	57h	I2C2 - Generic Interrupt
0Ch	TMR2 - Timer 2 Interrupt	32h	ADC1 - AN3	58h	I2C2 - Receive Buffer Interrupt
0Dh	U1RX - UART1 Receiver	33h	ADC1 - AN4	59h	I2C2 - Transmit Buffer Interrupt
0Eh	U1TX - UART1 Transmitter	34h	ADC1 - AN5	5Ah	Touch Controller #1
0Fh	U2RX - UART2 Receiver	35h	ADC1 - AN6	5Bh	DMA0 Interrupt
10h	U2TX - UART2 Transmitter	36h	ADC2 - AN0	5Ch	DMA1 Interrupt
11h	U3RX - UART3 Receiver	37h	ADC2 - AN1	5Dh	DMA2 Interrupt
12h	U3TX - UART3 Transmitter	38h	ADC2 - AN2	5Eh	DMA3 Interrupt
13h	U4RX - UART4 Receiver	39h	ADC2 - AN3	5Fh	DMA4 Interrupt
14h	U4TX - UART4 Transmitter	3Ah	ADC2 - AN4	60h	DMA5 Interrupt
15h	TMR3 - Timer 3 Interrupt	3Bh	ADC2 - AN5	61h	DMA6 Interrupt
16h	CCP1 IC/OC - Input Capture/Output Compare	3Ch	ADC2 - AN6	62h	DMA7 Interrupt
17h	CCP2 IC/OC - Input Capture/Output Compare	3Dh	ADC2 - AN7	63h	ITC TXA
18h	CCP3 IC/OC - Input Capture/Output Compare	3Eh	ADC3 - AN0	64h	ITC TXB
19h	CCP4 IC/OC - Input Capture/Output Compare	3Fh	ADC3 - AN1	65h	ITC TXC
1Ah	CCP5 IC/OC - MCCP Input Capture/Output Compare	40h	ADC3 - AN2	66h	ITC List 0
1Bh	CCP1 Trigger Output	41h	ADC3 - AN3	67h	ITC List 1
1Ch	CCP2 Trigger Output	42h	ADC3 - AN4	68h	ITC List 2
1Dh	CCP3 Trigger Output	43h	ADC3 - AN5	69h	ITC All Lists
1Eh	CCP4 Trigger Output	44h	ADC3 - AN6	6Ah	ITC Load Sequence

Table 13-2. DMA Channel Trigger Sources (continued)

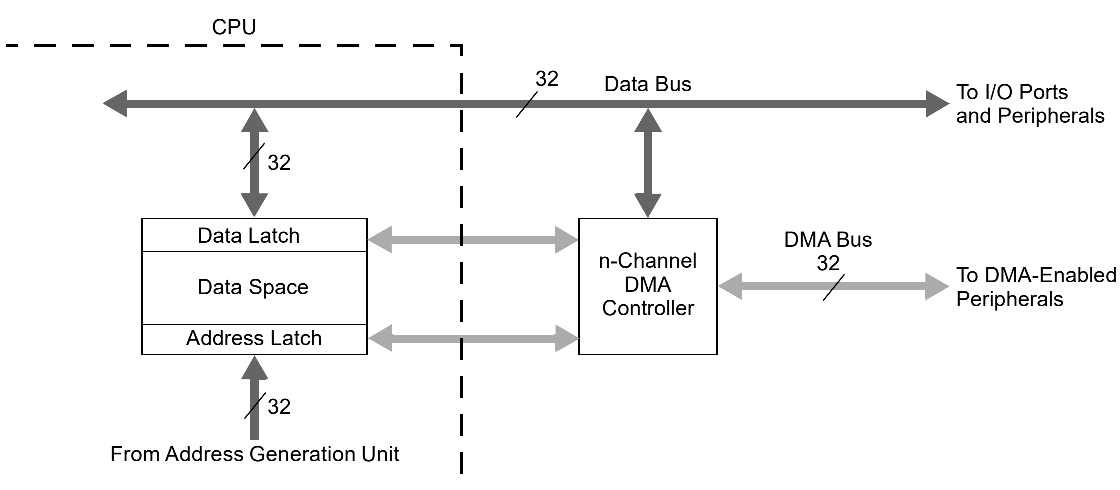
CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)	CHSEL[7:0]	Trigger (Interrupt)
1Fh	CCP5 Trigger Output	45h	ADC3 - AN7	6Bh	I3C Transmit Buffer Interrupt
20h	PWM EVENT A	46h	ADC3 - AN8	6Ch	I3C Receive Buffer Interrupt
21h	PWM EVENT B	47h	ADC3 - AN9	6Dh	I3C Transmit Burst Request CH0
22h	PWM1 Generator 1	48h	ADC3 - AN10	6Eh	I3C Transmit Burst Request CH1
23h	PWM1 Generator 2	49h	ADC3 - AN11	6Fh-7Fh	Reserved; Do Not Use
24h	PWM1 Generator 3	4Ah	CAN-FD Receive Interrupt		
25h	PWM Generator 4	4Bh	CAN-FD Transmit Interrupt		

13.2. Architectural Overview

The DMA Controller functions both as a peripheral and a direct extension of the CPU. It is located on the microcontroller data bus between the CPU and DMA-enabled peripherals, with direct access to data space (Figure 13-1). This partitions the SFR bus into two buses, allowing the DMA Controller access to the DMA-capable peripherals located on the new DMA SFR bus. This also lowers bus loading for less power consumption per access. The controller serves as a host device on the DMA SFR bus, controlling data flow from DMA-capable peripherals.

When the CPU is servicing peripherals that are not on the DMA bus, the DMA Controller is free to service peripherals on the DMA bus while the CPU is performing its operations. In this way, the effective bandwidth for handling data is increased. At the same time, DMA operations can proceed without causing a processor stall. When the CPU and DMA are accessing the SFR simultaneously, the CPU gets priority and the DMA will have to wait until the CPU completes the task.

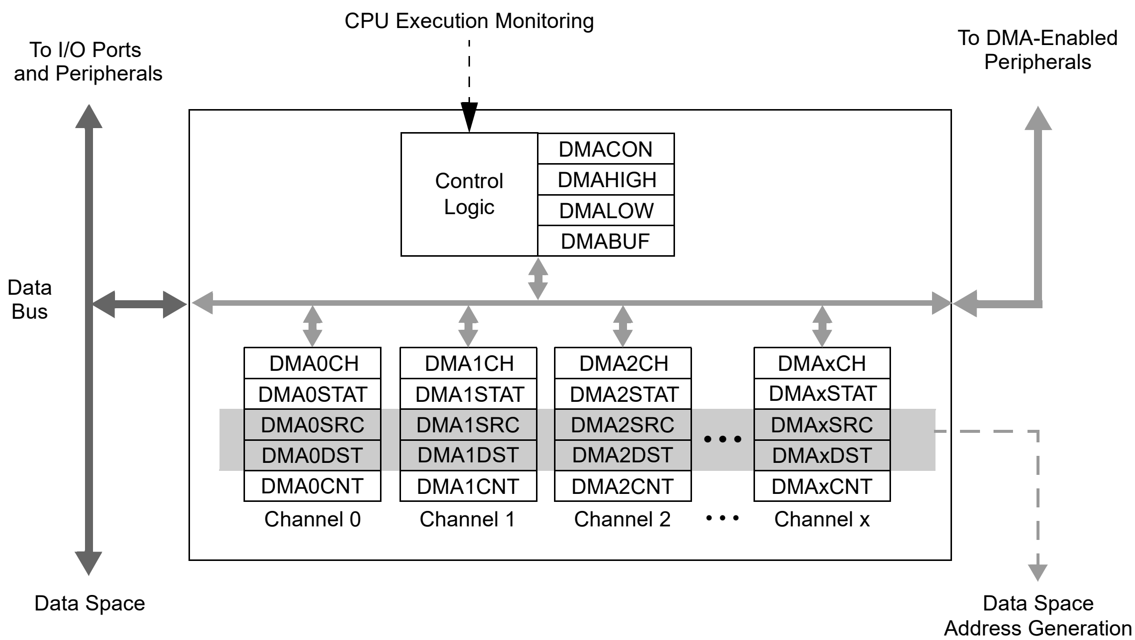
Figure 13-1. DMA Location Block Diagram



The DMA Controller itself is composed of multiple independent DMA channel controllers or simply channels (Figure 13-2). Each channel can be independently programmed to transfer data between different areas of the data space, move data between single or multiple addresses, use a wide range of hardware triggers to initiate transfers and conduct programmed transactions once or many times. Multiple channels may even be programmed to work together to carry out more complex data transfers without CPU intervention. The top-level controller sets the boundary addresses for all

DMA operations, regardless of the channel. It also arbitrates data bus access between the channels based on a user-selectable priority scheme and determines how DMA will operate in power-saving modes.

Figure 13-2. DMA Channel Controllers Block Diagram



13.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2300	DMACON	31:24								
		23:16								
		15:8	ON		SIDL					
		7:0								PRIORITY
0x2304	DMABUF	31:24	DMABUF[31:24]							
		23:16	DMABUF[23:16]							
		15:8	DMABUF[15:8]							
		7:0	DMABUF[7:0]							
0x2308	DMALOW	31:24								
		23:16	LADDR[23:16]							
		15:8	LADDR[15:8]							
		7:0	LADDR[7:0]							
0x230C	DMAHIGH	31:24								
		23:16	HADDR[23:16]							
		15:8	HADDR[15:8]							
		7:0	HADDR[7:0]							
0x2310	DMA0CH	31:24			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
		23:16			SDTEN	DDTEN			DRETEN	RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
0x2314	DMA0SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x2318	DMA0STAT	31:24								
		23:16								
		15:8							BWERR	BRERR
		7:0	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x231C	DMA0SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x2320	DMA0DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x2324	DMA0CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2328	DMA0CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x232C	DMA0SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x2330	DMA0INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x2334	DMA0MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2338	DMA0PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x233C	DMA1CH	31:24			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
		23:16			SDTEN	DDTEN			DRETEN	RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
0x2340	DMA1SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x2344	DMA1STAT	31:24								
		23:16								
		15:8							BWERR	BRERR
		7:0	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x2348	DMA1SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x234C	DMA1DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x2350	DMA1CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2354	DMA1CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x2358	DMA1SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x235C	DMA1INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x2360	DMA1MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x2364	DMA1PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x2368	DMA2CH	31:24			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
		23:16			SDTEN	DDTEN			DRETEN	RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
0x236C	DMA2SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x2370	DMA2STAT	31:24								
		23:16								
		15:8							BWERR	BRERR
		7:0	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2374	DMA2SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x2378	DMA2DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x237C	DMA2CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2380	DMA2CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x2384	DMA2SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x2388	DMA2INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x238C	DMA2MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x2390	DMA2PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x2394	DMA3CH	31:24			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
		23:16			SDTEN	DDTEN			DRETEN	RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
0x2398	DMA3SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x239C	DMA3STAT	31:24								
		23:16								
		15:8							BWERR	BRERR
		7:0	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x23A0	DMA3SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x23A4	DMA3DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x23A8	DMA3CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x23AC	DMA3CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x23B0	DMA3SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x23B4	DMA3INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x23B8	DMA3MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x23BC	DMA3PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x23C0	DMA4CH	31:24			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
		23:16			SDTEN	DDTEN			DRETEN	RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
0x23C4	DMA4SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x23C8	DMA4STAT	31:24								
		23:16								
		15:8							BWERR	BRERR
		7:0	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x23CC	DMA4SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x23D0	DMA4DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x23D4	DMA4CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x23D8	DMA4CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x23DC	DMA4SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x23E0	DMA4INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x23E4	DMA4MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x23E8	DMA4PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x23EC	DMA5CH	31:24			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
		23:16			SDTEN	DDTEN			DRETEN	RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
0x23F0	DMA5SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x23F4	DMA5STAT	31:24								
		23:16								
		15:8							BWERR	BRERR
		7:0	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x23F8	DMA5SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x23FC	DMA5DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x2400	DMA5CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2404	DMA5CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x2408	DMA5SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x240C	DMA5INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x2410	DMA5MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x2414	DMA5PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x2418	DMA6CH	31:24			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
		23:16			SDTEN	DDTEN			DRETEN	RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
0x241C	DMA6SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x2420	DMA6STAT	31:24								
		23:16								
		15:8							BWERR	BRERR
		7:0	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x2424	DMA6SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2428	DMA6DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x242C	DMA6CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x2430	DMA6CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x2434	DMA6SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							
0x2438	DMA6INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x243C	DMA6MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x2440	DMA6PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							
0x2444	DMA7CH	31:24			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
		23:16			SDTEN	DDTEN			DRETEN	RETEN
		15:8	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
		7:0	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
0x2448	DMA7SEL	31:24								
		23:16								
		15:8								
		7:0	CHSEL[7:0]							
0x244C	DMA7STAT	31:24								
		23:16								
		15:8							BWERR	BRERR
		7:0	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
0x2450	DMA7SRC	31:24								
		23:16	SADDR[23:16]							
		15:8	SADDR[15:8]							
		7:0	SADDR[7:0]							
0x2454	DMA7DST	31:24								
		23:16	DADDR[23:16]							
		15:8	DADDR[15:8]							
		7:0	DADDR[7:0]							
0x2458	DMA7CNT	31:24	CNT[31:24]							
		23:16	CNT[23:16]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x245C	DMA7CLR	31:24	CLR[31:24]							
		23:16	CLR[23:16]							
		15:8	CLR[15:8]							
		7:0	CLR[7:0]							
0x2460	DMA7SET	31:24	SET[31:24]							
		23:16	SET[23:16]							
		15:8	SET[15:8]							
		7:0	SET[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2464	DMA7INV	31:24	INV[31:24]							
		23:16	INV[23:16]							
		15:8	INV[15:8]							
		7:0	INV[7:0]							
0x2468	DMA7MSK	31:24	MSK[31:24]							
		23:16	MSK[23:16]							
		15:8	MSK[15:8]							
		7:0	MSK[7:0]							
0x246C	DMA7PAT	31:24	PAT[31:24]							
		23:16	PAT[23:16]							
		15:8	PAT[15:8]							
		7:0	PAT[7:0]							

13.3.1. DMA Module Control Register

Name: DMACON
Offset: 0x2300

Legend: r = Reserved bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		SIDL					
Access	R/W		R/W					
Reset	0		0					

Bit	7	6	5	4	3	2	1	0
								PRIORITY
Access								R/W
Reset								0

Bit 15 – ON DMA Module Enable bit

Value	Description
1	Enables module.
0	Disables module. When set low, all state machines are reset, resulting in immediate termination of all active DMA operation(s); however, the contents of the control registers are NOT reset to their default settings.

Bit 13 – SIDL DMA Stop in Idle bit

Value	Description
1	When system enters Idle mode, the module stops operation.
0	When system enters Idle mode, the module continues operation.

Bit 0 – PRIORITY Channel Priority Scheme Selection bit

Value	Description
1	Round robin scheme
0	Fixed priority scheme

13.3.2. DMA Data Buffer Register

Name: DMABUF
Offset: 0x2304

Bit	31	30	29	28	27	26	25	24
	DMABUF[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMABUF[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DMABUF[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DMABUF[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DMABUF[31:0] DMA Buffer bits
These bits reflect the content of the buffer that the DMA uses to hold the data being moved from the Source Address to the Destination Address.

13.3.3. DMA Low Address Limit Register

Name: DMALOW
Offset: 0x2308

Note:

1. The LADDR[23:0] bits apply to data space only. SFR space outside the range from LADDR[23:0] to HADDR[23:0] is accessible by the DMA and will not result in an interrupt. Setting LADDR[23:0] to an address in SFR will, therefore, be ignored.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	LADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – LADDR[23:0] Low Limit Address bits(1)

These bits indicate the lower address location below which the DMA module initiates a transaction, which results in the bus error trap, DMAFLT[1:0] = `01` and CHEN is cleared.

13.3.4. DMA High Address Limit Register

Name: DMAHIGH
Offset: 0x230C

Note:

1. The HADDR[23:0] bits apply to data space only. SFR space outside the range from LADDR[23:0] to HADDR[23:0] is accessible by the DMA and will not result in an interrupt. Setting HADDR[23:0] to an address in SFR will, therefore, be ignored.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	HADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – HADDR[23:0] High Limit Address bits[1]
These bits indicate the upper address location beyond which the DMA module initiates a transaction, which results in the bus error trap, DMAFLT[1:0] = `10` and CHEN is cleared.

13.3.5. DMA Channel x Control Register

Name: DMAxCH
Offset: 0x2310, 0x233C, 0x2368, 0x2394, 0x23C0, 0x23EC, 0x2418, 0x2444

Legend: HS = Hardware Settable bit, HC = Hardware Clearable bit

Notes:
1. The number of transfers per CHREQ bit setting depends on TRMODE[1:0].
2. The channel enable also depends on PCHEN if PPEN is set high for ping-pong operation support.
3. CNT[31:0] are reloaded in every repeated operation regardless of RELOADC.

Bit	31	30	29	28	27	26	25	24
			PPEN	PCHEN		RELOADC	RELOADD	RELOADS
Access			R/W	R/W/HS/HC		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
			SDTEN	DDTEN			DRETEN	RETEN
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bit	15	14	13	12	11	10	9	8
	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		FLWCON[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SIZE[1:0]			CHREQ	DONEEN	MATCHEN	HALFEN	CHEN
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W/HC
Reset	0	0		0	0	0	0	0

Bit 29 – PPEN Ping-Pong Operation Support Enable bit (refer to Ping-Pong)

Value	Description
1	Ping-pong operation support is enabled.
0	Ping-pong operation support is disabled.

Bit 28 – PCHEN Ping-Pong Channel Enable bit (refer to Ping-Pong)
Intended to support the ping-pong operation; the bit takes effect only when PPEN is set high as follows.

Value	Description
1	The DMA channel is enabled if CHEN is set high; this bit is hardware settable via PCHEN of the other DMA channel pair used in the Ping-Pong mode.
0	The DMA channel is disabled.

Bit 26 – RELOADC CNT[31:0] Reload bit[(3)]

Value	Description
1	CNT[31:0] are reloaded to their previously written value (buffered original content) upon the start of the next operation.
0	CNT[31:0] are not reloaded.

Bit 25 – RELOADD DADDR[23:0] Reload bit

Value	Description
1	DADDR[23:0] are reloaded to their previously written value upon the start of the next operation.
0	DADDR[23:0] are not reloaded.

Bit 24 – RELOADS SADDR[23:0] Reload bit

Value	Description
1	SADDR[23:0] are reloaded to their previously written value upon the start of the next operation.
0	SADDR[23:0] are not reloaded.

Bit 21 – SDTEN Source Descriptor Table Enable bit

Value	Description
1	Source Descriptor Table is enabled.
0	Source Descriptor Table is disabled.

Bit 20 – DDTEN Destination Descriptor Table Enable bit

Value	Description
1	Destination Descriptor Table is enabled.
0	Destination Descriptor Table is disabled.

Bit 17 – DRETEN Descriptor Read Error Trap Enable bit

Value	Description
1	DMA channel suspends the transfer operation on a descriptor read error and generates a DMA trap.
0	DMA channel continues the transfer operation when a descriptor read error is encountered.

Bit 16 – RETEN Read Error Trap Enable bit

Value	Description
1	DMA channel suspends the transfer operation on a bus read error and asserts the dma_trap signal.
0	DMA channel continues the transfer operation when a bus read error is encountered.

Bits 15:14 – SAMODE[1:0] Source Address Mode Selection bits

Value	Description
11	Reserved
10	SADDR[23:0] are decremented based on SIZE[1:0] after a transfer completion.
01	SADDR[23:0] are incremented based on SIZE[1:0] after a transfer completion.
00	SADDR[23:0] remain unchanged after a transfer completion.

Bits 13:12 – DAMODE[1:0] Destination Address Mode Selection bits

Value	Description
11	Reserved
10	DADDR[23:0] are decremented based on SIZE[1:0] after a transfer completion.
01	DADDR[23:0] are incremented based on SIZE[1:0] after a transfer completion.
00	DADDR[23:0] remain unchanged after a transfer completion.

Bits 11:10 – TRMODE[1:0] Transfer Mode Selection bits

Value	Description
11	Repeated Continuous
10	Continuous
01	Repeated One-Shot
00	One-Shot

Bits 9:8 – FLWCON[1:0] Data Flow Control bits

Value	Description
11	Reserved
10	Read from SADDR[23:0] only.
01	Read from SADDR[23:0] followed by a write to DADDR[23:0] and SADDR[23:0] (NULLW).
00	Read from SADDR[23:0] followed by write to DADDR[23:0].

Bits 7:6 – SIZE[1:0] Data Size Selection bits

Value	Description
11	Reserved
10	One 32-bit word is transferred at a time.
01	One 16-bit word is transferred at a time.
00	One byte word is transferred at a time.

Bit 4 – CHREQ DMA Channel Software Request bit[1]
This bit is automatically reset to '0' upon completion of a DMA transfer.

Value	Description
1	One DMA request is initiated by software.
0	No DMA request is initiated by software.

Bit 3 – DONEEN Done Interrupt Enable bit

Value	Description
1	An interrupt is invoked upon DONE flag being set.
0	An interrupt is not invoked by DONE condition.

Bit 2 – MATCHEN Pattern Match Enable bit

Value	Description
1	Pattern match is enabled.
0	Pattern match is disabled.

Bit 1 – HALFEN 50% Completion Watermark bit

Value	Description
1	An interrupt is invoked when the CNT[31:0] counter has reached its halfway point.
0	An interrupt is not invoked by a Half condition.

Bit 0 – CHEN DMA Channel Enable bit [2]

Value	Description
1	The corresponding DMA channel is enabled.
0	The corresponding DMA channel is disabled.

13.3.6. DMA Channel x Selection Register

Name: DMAxSEL
Offset: 0x2314, 0x2340, 0x236C, 0x2398, 0x23C4, 0x23F0, 0x241C, 0x2448

Note:

1. Unused CHSEL bits should be unimplemented (U-0).

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CHSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CHSEL[7:0] DMA Channel Trigger Selection bits[(1)]
These bits select one of the possible DMA triggers connected to the corresponding channel's input. See DMA Trigger Sources for more DMA trigger source information.

13.3.7. DMA Channel x Interrupt Register

Name: DMAxSTAT
Offset: 0x2318, 0x2344, 0x2370, 0x239C, 0x23C8, 0x23F4, 0x2420, 0x244C

Legend: C = Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
							BWERR	BRERR
Access							R/C/HS	R/C/HS
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	ADRERR[1:0]		DONE	HALF	OVERRUN		MATCH	DBUFWF
Access	R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS		R/C/HS	R
Reset	0	0	0	0	0		0	0

Bit 9 – BWERR DMA Fault Status bit 3

Value	Description
1	A bus read error has occurred; invalid data were received by the DMA.
0	No bus read error has occurred.

Bit 8 – BRERR DMA Fault Status bit 2

Value	Description
1	A bus read error has occurred; invalid data were received by the DMA.
0	No bus read error has occurred.

Bits 7:6 – ADRERR[1:0] DMA Fault Status bit 1 and bit 0

Value	Description
11	Address Fault condition(s)
10	Access attempted to address higher than HADDR[23:0], not detected until actual access.
01	Access attempted to address lower than LADDR[23:0], but above the SFR range, not detected.
00	No DMA Fault condition

Bit 5 – DONE DMA Complete Operation Interrupt Flag bit

Value	Description
1	The DMA channel's CNT[31:0] counter has reached 0.
0	The DMA channel's CNT[31:0] counter has not reached 0.

Bit 4 – HALF DMA 50% Watermark Level Interrupt Flag bit

Value	Description
1	The DMA channel's CNT[31:0] counter has reached the halfway point towards 0.
0	The DMA channel's CNT[31:0] counter has not reached the halfway point towards 0.

Bit 3 – OVERRUN DMA Channel Overrun Flag bit

Value	Description
1	The DMA channel is triggered while it is still completing the operation based on the previous trigger.
0	The overrun condition has not occurred.

Bit 1 – MATCH Pattern Match Status bit (see Pattern Match)

Value	Description
1	Pattern match has been detected.
0	Pattern match has not been detected.

Bit 0 – DBUFWF Buffered Data Write Flag bit

Value	Description
1	The content of the DBUF[31:0] (DMABUF[31:0]) bits has not been stored into the location specified in DADDR[23:0] or SADDR[23:0] in Null Write mode.
0	The content of the DBUF[31:0] (DMABUF[31:0]) bits has been stored into the location specified in DADDR[23:0] or SADDR[23:0] in Null Write mode.

13.3.8. DMA Channel x Source Address Register[1,2]

Name: DMAxSRC
Offset: 0x231C, 0x2348, 0x2374, 0x23A0, 0x23CC, 0x23F8, 0x2424, 0x2450

Notes:
1. For SFR, the DMA module operates over the entire address range.
2. For data space, the DMA module operates over the entire range of available memory.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – SADDR[23:0] Source Address bits
These bits indicate the address location from which the DMA module initiates the read operation. DMAxSRC register is dynamically updated based on SAMODE[1:0] and RELOADS bits.

13.3.9. DMA Channel x Destination Address Register[1,2]

Name: DMAxDST
Offset: 0x2320, 0x234C, 0x2378, 0x23A4, 0x23D0, 0x23FC, 0x2428, 0x2454

Notes:
1. For SFR, the DMA module operates over the entire address range.
2. For data space, the DMA module operates over the entire range of available memory.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	DADDR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – DADDR[23:0] Destination Address bits
These bits indicate the address location to which the DMA module initiates the write operation; DADDR[23:0] are dynamically updated based on DAMODE[1:0] and RELOADD.

13.3.10. DMA Channel x Count Register

Name: DMAxCNT
Offset: 0x2324, 0x2350, 0x237C, 0x23A8, 0x23D4, 0x2400, 0x242C, 0x2458

Bit	31	30	29	28	27	26	25	24
	CNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 31:0 – CNT[31:0] Count bits

The CNT[31:0] bits indicate the number of pending transfers. The count bits are decremented for every completed transfer. The bits are automatically reloaded in Repeated Transfer modes.

13.3.11. DMA Channel x Clear Register

Name: DMAxCLR
Offset: 0x2328, 0x2354, 0x2380, 0x23AC, 0x23D8, 0x2404, 0x2430, 0x245C

Bit	31	30	29	28	27	26	25	24
	CLR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CLR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CLR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CLR[31:0] Clear Register bits
Setting these bits high results in the corresponding data bit(s) from the DMABUF register being cleared. This action takes place after inverting and before setting, if applicable, and occurs prior to the data reaching their destination.

13.3.12. DMA Channel x Set Register

Name: DMAxSET
Offset: 0x232C, 0x2358, 0x2384, 0x23B0, 0x23DC, 0x2408, 0x2434, 0x2460

Bit	31	30	29	28	27	26	25	24
	SET[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SET[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SET[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SET[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SET[31:0] Set Register bits
Setting these bits high results in the corresponding data bit(s) from the DMABUF register being set. This action takes place after clear and then inverting, if applicable, and occurs prior to the data reaching their destination.

13.3.13. DMA Channel x Invert Register

Name: DMAxINV
Offset: 0x2330, 0x235C, 0x2388, 0x23B4, 0x23E0, 0x240C, 0x2438, 0x2464

Bit	31	30	29	28	27	26	25	24
	INV[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INV[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INV[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INV[31:0] Inverter Register bits
Setting these bits high results in the corresponding data bit(s) from the DMABUF register being inverted. This action takes place before setting and clearing, if applicable, and occurs prior to the data reaching their destination.

13.3.14. DMA Channel x Mask Register

Name: DMAxMSK
Offset: 0x2334, 0x2360, 0x238C, 0x23B8, 0x23E4, 0x2410, 0x243C, 0x2468

Bit	31	30	29	28	27	26	25	24
	MSK[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSK[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSK[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSK[31:0] Mask Register bits
Setting these bits high results in the corresponding data bit(s) from the DMABUF register and DMAxPAT registers being compared in the pattern match operation.

13.3.15. DMA Channel x Pattern Register

Name: DMAxPAT
Offset: 0x2338, 0x2364, 0x2390, 0x23BC, 0x23E8, 0x2414, 0x2440, 0x246C

Bit	31	30	29	28	27	26	25	24
	PAT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PAT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PAT[31:0] Pattern Register bits
These bits contain a user-provided pattern for the pattern match operation.

13.4. Operation

13.4.1. Data Transfer Options

The DMA Controller transfers data from a source address (DMAxSRC) to a destination address (DMAxDST) upon the receipt of a hardware or software trigger. The transfer occurs as a two-step process: a read from the source address and transfer to the DMA buffer, DMABUF, followed immediately by a write from DMABUF to the destination address. The controller then determines if the operation on this channel has been completed. The active DMA channel may assert an interrupt to indicate the end of a transfer and/or the status of a transfer in progress.

Each channel of the DMA controller can be independently programmed to move data between the data RAM and peripherals (i.e., the SFR area), between peripherals or between areas of data RAM. Transactions can be single occurrences, repeated single occurrences or continuous, based on an event trigger and/or the channel transaction counter. The source and destination address registers can be independently programmed to increment, decrement or remain unchanged during transactions.

13.4.2. DMA Trigger Sources

Each DMA channel can select from the hardware triggers to initiate a DMA transfer. The trigger sources are generally device-level interrupts from peripheral modules, as well as external interrupts and interrupt-on-change sources. The CHSEL[7:0] bits (DMAxSEL[7:0]) select which interrupt (and thus module) is used as a trigger for a particular DMA channel (see Table 13-2). The CHSEL bits may change at any time to select another module to service. However, it is not recommended to make a change while the associated DMA channel is in operation. A DMA channel can be configured to service any memory-mapped peripheral, regardless of the trigger's origin. This is because the trigger (interrupt) source is independent of the DMA source and destination addresses. For example, a

DMA channel configured to respond to the INT0 interrupt could be used to move data into or out of a UART FIFO. In most cases, it makes more sense to use a peripheral's own interrupt for performing a data transfer. However, there are also many cases where it is desirable to use one peripheral interrupt to perform a DMA operation on another peripheral, perhaps even another DMA channel. Examples of such operations are provided in Application Examples. In addition, a DMA channel may be triggered in software by setting the CHREQ bit (DMAxCH[4]). This allows for an application to use the DMA Controller to move data directly, without having to wait for a hardware interrupt. CHREQ is also set when a hardware trigger occurs.

13.4.3. Channel Priority and Priority Schemes

While DMA channels can function independently to service different peripherals at the same time, they are still limited by the presence of a single DMA data bus and a single data channel to data space. When two or more channels request the DMA controller to handle a data transfer at the same time, the controller arbitrates the requests and decides which channel receives priority and bus access.

The controller uses two defined arbitration schemes to assign channel priority: Fixed and Round Robin. The PRIORITY bit (DMACON[0]) determines the scheme to be used. In the Round Robin scheme (DMACON[0] = `1`), the controller assigns priority and bus grant to the lowest numbered channel for the first time when there is channel contention. Grant determination is evaluated after every iteration of data transfers. For each successive transfer conflict, the next higher channel receives preference, continuing as a cycle through all the channels. If the channel that has priority does not make a request at that time, it is skipped for the next channel in the cycle.

As an example, if Channels 0, 1 and 2 all simultaneously request a data transfer, Channel 0 is serviced; Channels 1 and 2 are then serviced in that order. During the next service request, any request from Channel 1 will receive preference; Channel 2 will receive preference in the following round. Any subsequent transfer requests from Channel 0 will be ignored until all the other channels have received priority once. Typical examples of Round Robin arbitration are shown in Table 13-3.

Table 13-3. Examples of Channel Access Using Round Robin Priority Scheme

Requesting DMA Channel(s)				Channel Granted Priority
0	1	2	3	
				None
	X			CH1
X	X	X		CH2
X	X			CH0
X	X			CH1
X	X		X	CH3
X	X			CH0

In contrast, the Fixed scheme (DMACON[0] = `0`) always gives priority to the lowest requesting channel number. Using the previous example, if there are several sequential transfer requests involving Channel 0, Channel 0 will always receive preference over other channels. The Fixed Priority scheme is the default. Typical examples are shown in Table 13-4.

Table 13-4. Examples of Channel Access Using Fixed Priority Scheme

Requesting DMA Channel(s)				Channel Granted Priority
0	1	2	3	
				None
	X			CH1
X	X	X		CH0
X	X			CH0

Table 13-4. Examples of Channel Access Using Fixed Priority Scheme (continued)

Requesting DMA Channel(s)				Channel Granted Priority
0	1	2	3	
	X			CH1
	X		X	CH1
			X	CH3

13.4.4. Memory Boundary

DMA can be used to read data from the following regions: SFR, SRAM, Configuration A Page, Configuration B Page and User Program Memory Space. Additionally, DMA can be used to write data to the SFR and SRAM regions. Refer to the memory map (Figure 4-1) for implemented regions.

While the 24-bit DMAxSRC and DMAxDST registers allow access to the entire data space, there may be circumstances where it is desirable to limit DMA operations to a much narrower range. This may be required for many reasons; for example, to protect program variables or a software stack.

The DMAHIGH and DMALOW registers allow the user to set the upper and lower address limits for DMA operations in the DMA-accessible memory regions. All DMA channels are restricted to the address range set by DMAHIGH and DMALOW. When an active DMA channel initiates a memory transaction outside of the boundary defined by this register pair, an interrupt will be invoked on a channel basis. Fault status bits, ADRERR[1:0] = `10`, indicate DMA operations that attempt to access above DMAHIGH, and ADRERR[1:0] = `01` indicates operations that attempt to access below DMALOW. Boundaries are applicable only to the SRAM region. The memory-mapped SFR range is always accessible by DMA.

13.4.5. Buffer Data Write Bit

The DBUFWF bit (DMAxSTAT[0]) indicates whether buffered data in DMABUF have been stored into the specified destination location. It serves as a protection against data loss due to unexpected termination of the active DMA operation. For example, if the user decides to stop the DMA operation that happens to be between load and store, the buffered data in DMABUF will not reach their destination. The user can examine this bit to see if the buffered data still need to be stored at the specified destination location.

Table 13-5 summarizes the behavior of DBUFWF in various modes of operation.

Table 13-5. Interpretation of the DBUFWF Bit Status

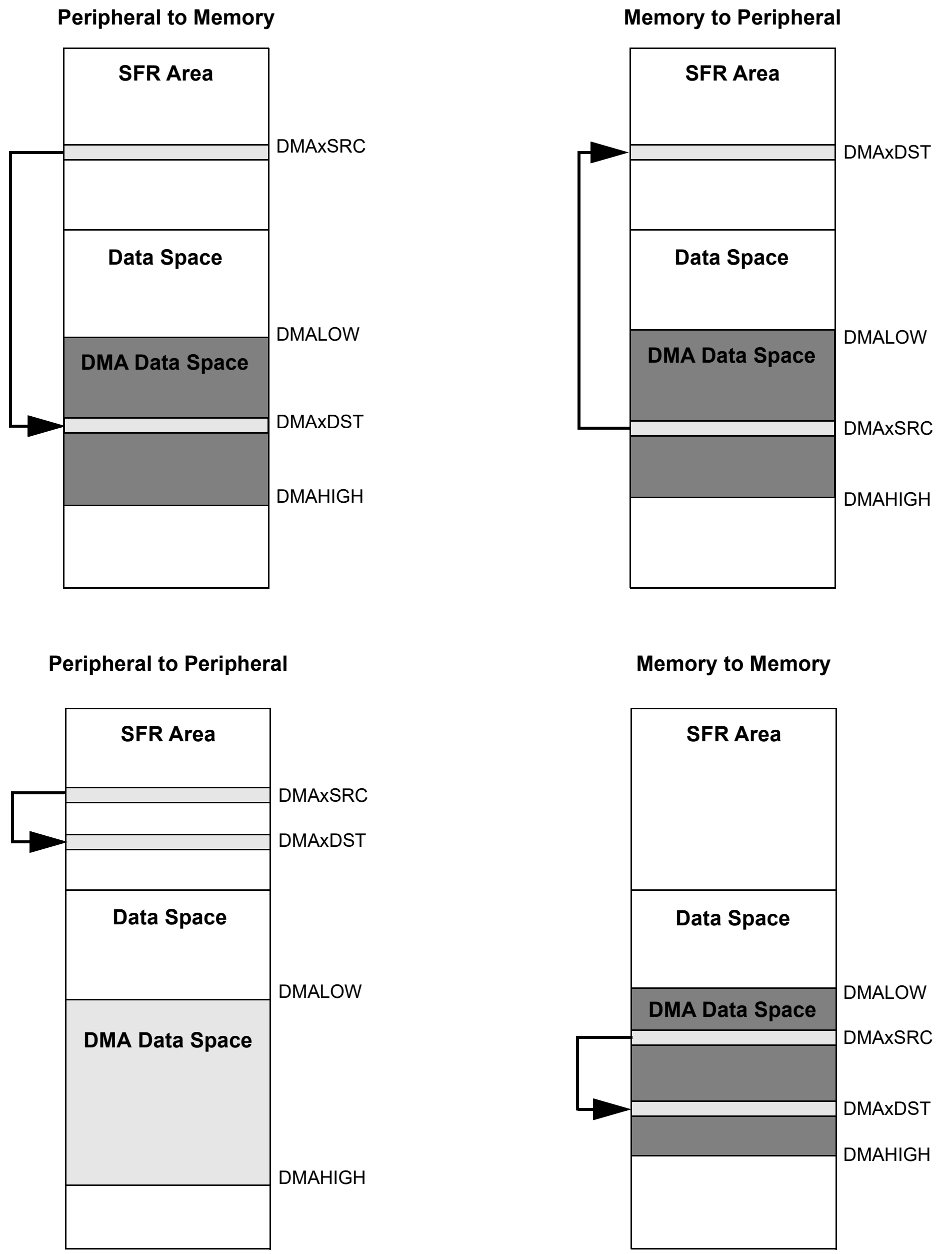
DBUFWF Status	Operation Status		
	Repeated One-Shot	Repeated Continuous	Null Write
`1`	After Loading DMABUF		
`0`	After writing to [DMAxDST]		After Writing to [DMAxSRC]

13.4.6. Types of Data Transfers

All DMA transactions occur solely within the data space address space. In the least restricted case, all Data Space addresses are available to the DMA Controller; this includes the entire SFR space and (by extension) all peripherals. As defined by the source and destination, there are four types of DMA data transfers (Figure 13-3):

- Peripheral to Memory (Receive)
- Memory to Peripheral (Transmit)
- Memory to Memory
- Peripheral to Peripheral

Figure 13-3. Types of DMA Data Transfers



Note: Relative sizes of memory areas are not shown to scale.

13.4.6.1. Peripheral to Memory (Receive)

If a source address register is programmed with an SFR address while the destination register contains a data space address, the controller will read from the peripheral module being serviced and write the retrieved data content to the specified location in data space. This is most suitable for peripherals configured to receive data, such as a UART or SPI module.

13.4.6.2. Memory to Peripheral (Transmit)

If the source address register is programmed with a data space address and the destination address register contains a peripheral (SFR) address, the controller is forced to read from the data space and write to the SFR when triggered. This makes this type of data flow most suitable to support the peripheral modules configured to transmit data, such as a serial communication module.

13.4.6.3. Memory to Memory

If data relocation within data space is required, the source and destination address registers for any channel can simply be programmed with the desired data space memory locations. Obviously, this type of data transfer does not require access to any peripheral; however, the trigger from any of the peripherals can be used to initiate the transfer.

13.4.7. Data Transfers Modes

Data transfers are also defined by how the transaction is structured: the number of data transfers that can occur per trigger event, how events are counted and if the event repeats. The DMA Controller defines four transfer modes, which encompass all these features:

- One-Shot
- Repeated One-Shot
- Continuous
- Repeated Continuous

The Transfer mode is defined by the TRMODE[1:0] bits (DMAxCH[11:10]). In addition, the RELOADS, RELOADD and RELOADC bits (DMAxCH[26:24]) can modify the behavior of some modes.

13.4.7.1. Common Transfer Mode Sequence

Regardless of the transfer mode, all DMA transfers follow the same basic sequence:

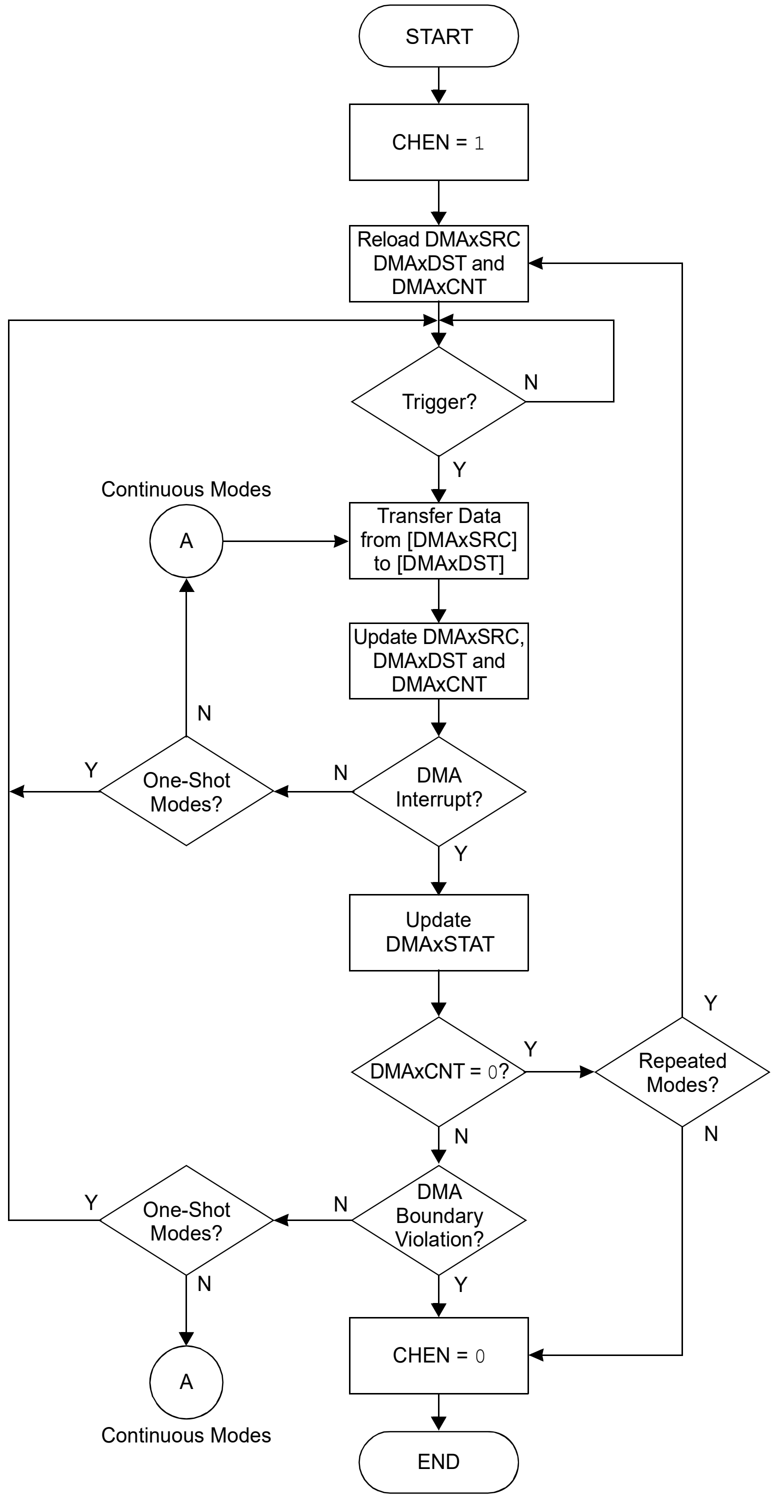
1. Upon the receipt of a DMA trigger or the setting of the CHREQ bit (DMAxCH[4]), data are loaded into DMABUF from the location addressed by DMAxSRC, then stored in the location addressed by DMAxDST.
2. Following the transaction, DMAxSRC and DMAxDST are updated appropriately; one or both may be incremented or decremented, depending on the channel's configuration (see Addressing Modes for additional information). At the same time, DMAxCNT is decremented by one.
3. The module tests for any DMA interrupt conditions. If an interrupt condition has occurred, the DMAxSTAT register flags are updated accordingly:
   - If a DMA interrupt has occurred, all modes continue to step 4.
   - If an interrupt has not occurred, all modes return to step 1. In One-Shot mode, the controller waits for the next trigger. In Continuous mode, the controller repeats the cycle continuously until DMAxCNT value becomes 0 or a DMA interrupt occurs.
4. If DMAxCNT has decremented to zero:
   - The values of DMAxSRC, DMAxDST and DMAxCNT are reloaded and the sequence repeats from step 1 (all Repeated modes).
   - The CHEN bit (DMAxCH[0]) is cleared and the channel is disabled (One-Shot and Continuous modes).
5. If DMAxCNT has not decremented to zero, the controller checks for a memory address boundary violation of DMALOW or DMAHIGH:
   - If one of the boundaries has been crossed, the CHEN bit is cleared, and the channel is disabled.
   - If there is no boundary violation, the controller returns to step 1. For both One-Shot modes, the controller waits for the next trigger. For both Continuous modes, the controller proceeds to performing the next data transfer.

The four data transfer modes differ in the number of data transfers that can take place with a single trigger and how the DMAxCNT register behaves. The common logic flow for all data transfer modes is illustrated in the flowchart in Figure 13-4. The differences between the modes are summarized in Table 13-6.

Table 13-6. Comparison of DMA Data Transfer Modes

Transfer Mode	Transfers per Trigger	DMAxCNT Behavior	
		Decrements on	at 0000h
One-Shot	Single	Trigger	Disable Channel
Repeated One-Shot		Transfer	Reload and Repeat
Continuous	Multiple	Trigger	Disable Channel
Repeated Continuous		Transfer	Reload and Repeat

Figure 13-4. Common Logic Flow for Data Transfer Modes



Legend:
One-Shot Modes: One-Shot and Repeated One-Shot
Continuous Modes: Continuous and Repeated Continuous
Repeated Modes: Repeated One-Shot and Repeated Continuous

Note: When CHEN = 1, in Non-Repeated modes (One-Shot or Continuous), the registers DMAxSRC, DMAxDST and DMATxCNT are reloaded based on their respective reload bits status (RELOADS, RELOADD and RELOADC).

13.4.7.2. One-Shot Mode

In One-Shot mode (TRMODE[1:0] = 00), a single transfer (from DMAxSRC to DMAxDST) is performed for each trigger event. By default, the Reset value of DMAxCNT is 0001h. When a single One-Shot transfer occurs, DMAxCNT is decremented to 0000h; this disables the channel and requires the channel to be re-enabled to perform the next transaction. Of course, it is also possible to store a larger value in DMAxCNT and then conduct a defined number of One-Shot transfers.

Example 13-1 shows a typical code sequence for a One-Shot transfer.

Example 13-1. Typical Code Sequence for a One-Shot Transfer

```
uint8_t SRC[4];
uint8_t DST[4];
int i;

int main()
{
    for (i = 0; i < 4; i++) {
        SRC[i] = i + 1;  //fill with i+1
        DST[i] = 0;      //fill with 0
    }

    DMACONbits.ON = 1;
    DMACONbits.PRIORITY = 1;
    DMALOW = 0x4000;
    DMAHIGH = 0x7FFF;                    //set lower and upper address limit
    DMA0SRC = (unsigned int) SRC;        //load the source address
    DMA0DST = (unsigned int) DST;        //load destination address
    DMA0CNT = 4;                         //Four transfers to be done
    DMA0CH = 0;
    DMA0CHbits.SAMODE = 1;               //Source address increment mode
    DMA0CHbits.DAMODE = 1;               //Destination address increment mode
    DMA0CHbits.TRMODE = 0;               //One-Shot Transfer mode
    DMA0CHbits.CHEN = 1;                 //Enable channel
    IFS2bits.DMA0IF = 0;
    DMA0CHbits.CHREQ = 1;                //First trigger
    while (DMA0CHbits.CHREQ);

    DMA0CHbits.CHREQ = 1;                //Second trigger
    while (DMA0CHbits.CHREQ);

    DMA0CHbits.CHREQ = 1;                //Third trigger
    while (DMA0CHbits.CHREQ);            //HALF=1 since DMACNT is at halfway
point

    DMA0CHbits.CHREQ = 1;                //Fourth trigger DMACNT=0 and transfer
complete
    while (DMA0CHbits.CHREQ);

    while (!DMA0STATbits.DONE);          //Transfer Complete

    //DMA0IF=1 ,DONE=1, CHEN=0;
    IFS2bits.DMA0IF = 0;
    DMA0STATbits.DONE=0;
    DMA0STATbits.HALF=0;

    while (1);
}
```

13.4.7.3. Repeated One-Shot Mode

In Repeated One-Shot mode (TRMODE[1:0] = `01`), single transfers occur repeatedly as long as triggers are being provided or CHREQ is set. Each time a trigger occurs or CHREQ is set, DMAxCNT is decremented. In this case, however, the channel is not disabled when DMAxCNT reaches 0000h. Instead, the original value of DMAxCNT is reloaded. If RELOADS = `1` and RELOADD = `1`, the original values of DMAxSRC and DMAxDST are also reloaded. The entire cycle then starts again on the next trigger. To end the sequence, the channel must be disabled by clearing the CHEN bit in software.

Example 13-2 shows a typical code sequence for a Repeated One-Shot transfer.

Example 13-2. Typical Code Sequence for a Repeated One-Shot Transfer

```
unsigned char Array1[4];
unsigned char Array2[4];

void test(void);

int main()
{
    for (int i=0; i<4; i++)
    {
        Array1[i]=i+1;         //fill with i+1
        Array2[i]=0;           //fill with 0
    }

    DMACONbits.ON=1; //Enable DMA

    //set lower and upper address limit
    DMAHIGH=0x5000;
    DMALOW=0x4000;

    DMA0SRC=(unsigned int)& Array1;     //load the source address
    DMA0DST=(unsigned int)& Array2;     //load destination address
    DMA0CNT=4;

    DMA0CH=0;
    DMA0CHbits.SAMODE=1;               //Source address increment mode
    DMA0CHbits.DAMODE=1;               //Destination address increment mode
    DMA0CHbits.TRMODE=1;               //Transfer mode Repeat One-Shot
    DMA0CHbits.RELOADS=1;              //Reload Source Address
    DMA0CHbits.RELOADD=1;              //Reload Destination Address
    DMA0CHbits.DONEEN=1;               //Enable interrupt on DONE being set
    DMA0CHbits.SIZE=0;                 //One byte transferred at a time
    DMA0CHbits.CHEN=1;                 //Channel enable
    IFS2bits.DMA0IF=0;

    while(1)
    {
        DMA0CHbits.CHREQ=1;            //First trigger
        while(DMA0CHbits.CHREQ);
        DMA0CHbits.CHREQ=1;            //Second trigger
        while(DMA0CHbits.CHREQ);
        DMA0CHbits.CHREQ=1;            //Third trigger
        while(DMA0CHbits.CHREQ);       //HALF=1 since DMACNT is at half way
point
        DMA0CHbits.CHREQ=1;            //Fourth trigger DMACNT=0
        //Wait for transfer complete
        while(DMA0CHbits.CHREQ);       //DMACNT reloaded to 4, DMA0IF=1
        //Since RELOADS=1 and RELOADD=1,DMASRC0/DMADST0 are reloaded.
        while(!IFS2bits.DMA0IF);
        DMA0STATbits.DONE=0;           //Clear DONE and HALIF flag
        DMA0STATbits.HALF=0;
        IFS2bits.DMA0IF=0;
    }
}
```

13.4.7.4. Continuous Mode

In Continuous mode (TRMODE[1:0] = `10`), a single trigger starts a sequence of back-to-back transfers; these continue with each transfer decrementing DMAxCNT until it reaches 0000h. At this point, like One-Shot mode, the channel is disabled.

One-Shot and Continuous modes are similar in that each mode performs a certain number of transfers for one time. The difference is that One-Shot mode requires a trigger for each transfer, while Continuous mode allows many transfers for each trigger. In addition, DMAxCNT is controlled by the number of individual transactions, not the number of triggers.

Example 13-3 shows a typical code sequence for a Continuous transfer.

Example 13-3. Typical Code Sequence for a Continuous Transfer

```
unsigned int Array1[100];
unsigned int Array2[100];

int main()
{
    for (int i=0; i<100; i++)
    {
        Array1[i]=i+1;          //fill with i+1
        Array2[i]=0;            //fill with 0
    }

    DMACONbits.ON=1;            //Enable DMA

    //Set lower and upper address limit
    DMAHIGH=0x5000;
    DMALOW=0x4000;

    DMA0SRC=(unsigned int)& Array1;         // load the source address
    DMA0DST=(unsigned int)& Array2;         // load destination address
    DMA0CNT=100;                            // 100 Transactions per trigger

    DMA0CH=0;
    DMA0CHbits.SAMODE=1;                    //Source address increment mode
    DMA0CHbits.DAMODE=1;                    //Destination address increment mode
    DMA0CHbits.TRMODE=2;                    //Transfer mode Continuous
    DMA0CHbits.DONEEN=1;                    //Enable interrupt on DONE being set
    DMA0CHbits.SIZE=2;                      //One 32-bit word transferred at a
time
    DMA0CHbits.CHEN=1;                      //Channel enable

    IFS2bits.DMA0IF=0;
    DMA0CHbits.CHREQ=1;                     //Enable the transfer by software
trigger
    while(!DMA0STATbits.HALF);              //HALF=1 is set when DMA0CNT
reaches halfway
    while(!IFS2bits.DMA0IF);                //DONE=1;CHAEN=0,DMA0IF=1
    //100 (DMAXCNT=100) transfers complete with one trigger

    IFS2bits.DMA0IF=0;

    while(1);
}
```

13.4.7.5. Repeated Continuous Mode

Repeated Continuous mode (TRMODE[1:0] = 11) can be thought of as a combination of Continuous and Repeated One-Shot modes; data transfers keep occurring as long as triggers are provided, and multiple transfers can occur with each trigger. Like Continuous mode, each transfer decrements DMAxCNT. When it reaches 0000h, the address and count registers are reloaded, and the process is repeated.

Like Repeated One-Shot mode, ending the sequence requires disabling the channel, either by disabling the trigger source or clearing the CHEN bit in software.

Example 13-4 shows a typical code sequence for a Repeated Continuous mode transfer.

Example 13-4. Typical Code Sequence for a Repeated Continuous Transfer (Memory to Memory, RELOADS = 1, RELOADD = 1)

```
unsigned int Array1[100];
unsigned int Array2[100];
int i;

int main()
{

    for (i=0;i<100;i++)
    {
        Array1[i]=i+1;          //fill with i+1
        Array2[i]=0;            //fill with 0
    }

    DMACONbits.ON=1; //Enable DMA

    DMACONbits.PRIORITY = 1;  //Robin is round

    //Set lower and upper address limit
    DMAHIGH=0x5000;
    DMALOW=0x4000;

    DMA0SRC=(unsigned int)& Array1;     // load the source address
    DMA0DST=(unsigned int)& Array2;     // load destination address
    DMA0CNT=100;                        // 100 Transactions per trigger

    DMA0CH=0;
    DMA0CHbits.SAMODE=1;                //Source address increment mode
    DMA0CHbits.DAMODE=1;                //Destination address increment mode
    DMA0CHbits.TRMODE=3;                //Transfer mode Repeat continuous
    DMA0CHbits.RELOADS=1;               //Reload Source Address
    DMA0CHbits.RELOADD=1;               //Reload Destination Address
    DMA0CHbits.DONEEN=1;                //Enable interrupt on DONE being set
    DMA0CHbits.SIZE=2;                  //One 32-bit word transferred at a time
    DMA0CHbits.CHEN=1;                  //Channel enable
    IFS2bits.DMA0IF=0;

    while(1)
    {
        DMA0CHbits.CHREQ=1;             //TIGGER
        while(!IFS2bits.DMA0IF);        //Wait for transaction to complete

        Nop();

        //Clear the Destination Memory
        for (i=0;i<100;i++)
        {
            Array2[i]=0;                //fill with 0
        }

        //Clear done interrupt flag and status bits prior to re-triggering
        IFS2bits.DMA0IF=0;
        DMA0STATbits.DONE=0;
        DMA0STATbits.HALF=0;
    }
}
```

13.4.7.6. Address and Count Reload

Although the Repeated modes explicitly include it, all the Transfer modes allow the automatic reuse of the initial source and destination addresses and transaction counts for multiple operations. Setting the RELOADS (DMAxCH[24]), RELOADD (DMAxCH[25]) and RELOADC (DMAxCH[26]) bits allows the values of DMAxSRC, DMAxDST and DMAxCNT to be restored for the next DMA operation. This causes the registers to be reloaded in One-Shot and Continuous modes after a transfer operation is complete and the channel is re-enabled. Address and transaction count reloading is automatic in Repeated One-Shot and Repeated Continuous modes. DMAxCNT also has its value reloaded after it has been decremented to 0000h, regardless of the setting of the RELOADC or TRMODEx bits. The only exception is if the channel is stopped in mid-operation and restarted later.

Table 13-7 shows the effect of RELOADS/RELOADD/RELOADC on DMAxSRC, DMAxDST and DMAxCNT for the Data Transfer modes.

Table 13-7. RELOADS/RELOADD/RELOADC Bits and Data Transfer Modes

RELOADS/RELOADD/ RELOADC Bits(2)	Transfer Mode	DMAxSRC	DMAxDST	DMAxCNT
1	—	Reloaded	Reloaded	Reloaded
0	Repeated One-Shot/ Continuous	Not Reloaded	Not Reloaded	Reloaded(1)
0	One-Shot/Continuous	Not Reloaded	Not Reloaded	Not Reloaded

Notes:
1. The reload only happens after DMAxCNT has decremented to ‘0’. No reload occurs if the channel is stopped and later resumed.
2. The CHEN bit must be enabled for RELOAD to occur. When CHEN = 1, in Non-Repeated modes (one-shot or continuous), the registers DMAxSRC, DMAxDST and DMATxCNT are reloaded based on their respective reload bits status (RELOADS, RELOADD and RELOADC). Additionally, in ping-pong mode, ensure that the channel is re-enabled whenever the PCHEN signal for that channel is high.

13.4.8. Addressing Modes

Following each transfer, the DMAxSRC and DMAxDST registers may be automatically updated by the channel. This potentially allows the channel to move data between multiple locations without the need for user intervention. Automatic address updating is controlled by the SAMODEx and DAMODEx bits (DMAxCH[15:14] and [13:12]).

The combination of the different address update options (fixed, increment or decrement) provides four supported addressing modes:

- Fixed to Fixed
- Fixed to Block
- Block to Fixed
- Block to Block

Table 13-8 shows the Addressing modes and the various SAMODEx and DAMODEx combinations.

Table 13-8. Configurations for DMA Addressing Modes

Mode(1)	SAMODE[1:0]	DAMODE[1:0]
Fixed to Fixed	00	00
Fixed to Block (Address Increment)	00	01
Fixed to Block (Address Decrement)	00	10
Block to Fixed (Address Increment)	01	00
Block to Fixed (Address Decrement)	10	00
Block to Block (Address Increment)	01	01
Block to Block (Address Decrement)	10	10

Table 13-8. Configurations for DMA Addressing Modes (continued)

Mode(1)	SAMODE[1:0]	DAMODE[1:0]

Note:
1. The increment and decrement of the address will be based on the SIZEx bits value.

13.4.8.1. Fixed to Fixed

Fixed to Fixed mode is set by configuring SAMODE[1:0] and DAMODE[1:0] to '`00`'. In this mode, the source and destination addresses remain the same after each transaction. This mode is suited for One-Shot transfers of a single byte, or word of data, between two fixed addresses.

13.4.8.2. Fixed to Block

In Fixed to Block mode, the source address remains unchanged throughout the transfer, but the destination address is incremented or decremented (depending on the DAMODEx setting). This works well for receiving data from the single-word buffer of a serial communication peripheral and filling a block of addresses designated as a buffer.

Example 13-5 shows sample code for Fixed to Block Addressing. The source address (UART receive) generates an interrupt when the UART buffer has received four bytes; the UART Receive Interrupt Flag (U2RIF) will trigger the transfer.

Example 13-5. Code for Fixed to Block Continuous Transfer (Peripheral to Memory)

```
void UartInit(void);
unsigned char Array2[32];

int main()
{
    UartInit();

    for (int i = 0; i < 32; i++)
    {
        Array2[i] = 0;             //fill with 0
    }

    DMACONbits.ON = 1;             //Enable DMA

    DMAHIGH = 0x5000;              //set lower and upper address limit
    DMALOW = 0x4000;
    DMA0SRC = (uint32_t) &U1RXB;
    DMA0DST = (uint32_t) &Array2[0]; // load destination address
    DMA0CNT = 4; //When the UART buffer has 4 bytes, do an interrupt and
transfer 4 bytes
    DMA0CH = 0;

    DMA0CHbits.SIZE=0;             //One byte transferred at a time
    DMA0CHbits.SAMODE = 0;         //Source address increment mode: do not
increment
    DMA0CHbits.DAMODE = 1;         //Destination address increment mode:
increment 1
    DMA0CHbits.TRMODE = 2;         //Transfer mode: Continuous

    DMA0SELbits.CHSEL = 15;        //Trigger DMA channel 0 on UART1 Receive
    DMA0CHbits.DONEEN=1;           //Enable interrupt on DONE being set
    DMA0CHbits.CHEN = 1;           //Channel 0 enable

    IFS2bits.DMA0IF = 0;
    //Wait for the bytes to be transferred (triggered by UART reception)
    while (!IFS2bits.DMA0IF);     //DONE=1;CHAEN=0,DMA0IF=1 and transfer
complete with one trigger

    IFS2bits.DMA0IF = 0;

    //4 bytes received from UART1 are now be stored in Array2.
    //DMA0DST will point at Array2[4], and further bytes can be transferred by
enabling the channel again.

    while (1);
}
```

```
//Enable UART1 for 115200 baud rate using RB6 as UART1 TX and RB7 as UART1 RX.
void UartInit(void)
{
    //Enable CLKEN8 to provide clock to UART
    CLK8CONbits.ON = 1;

    //Map RB6 (RP23) to UART1 TX
    _RP23R = 10;
    //Map RB7 (RP24) to UART1 RX
    _U1RXR = 24;

    //Set RB6/RB7 as digital I/O
    _ANSELB6 = 0;
    _ANSELB7 = 0;
    //Set RB6 as output
    _TRISB6 = 0;
    //Set RB7 as input
    _TRISB7 = 1;

    U1CONbits.CLKMOD = 1;          //Use fractional baud rate generation
    U1CONbits.CLKSEL = 1;          //Select CLKGEN8 as clock source

    U1CONbits.MODE = 0;            //Asynchronous 8-bit UART

    //Enable RX and TX functionality
    U1CONbits.RXEN = 1;
    U1CONbits.TXEN = 1;

    //In fractional baud rate mode, UxBRG = Input clock / baud rate, for 8MHz
and 115200 baud
    U1BRG = (8000000 / 115200);

    U1STATbits.RXWM = 3;          //Interrupt after 4 transfers

    //Lastly, enable UART1
    U1CONbits.ON = 1;

    _U1RXIF = 0;
}
```

13.4.8.3. Block to Fixed

In Block to Fixed mode, the source address is incremented or decremented throughout the transfer (depending on the SAMODEx setting) while the destination address remains unchanged. This is well-suited for moving a packet of data to be transmitted into the 'single-word' transmit buffer of a serial communication peripheral.

13.4.8.4. Block to Block

In Block to Block mode, both the source and destination addresses increment or decrement (depending on the SAMODEx and DAMODEx bits setting) throughout the transfer. This mode is useful for copying a block of data from one part of the data RAM to another.

13.4.9. Flow Control

The Flow Control bits, FLWCONx (DMAxCH[9:8]), can be used to configure three different types of data transfer:

- Source to Destination mode (default)
- Null Write mode
- Read-Only mode

13.4.9.1. Source to Destination Transfer

By default (FLWCON[1:0] = `00`), a DMA transfer occurs from the source address to the destination address. The behavior of the source and destination addresses will depend on the configuration of SAMODE[1:0] and DAMODE[1:0] (as described in Fixed to Fixed, Fixed to Block, Block to Fixed and Block to Block).

13.4.9.2. Null Write Mode

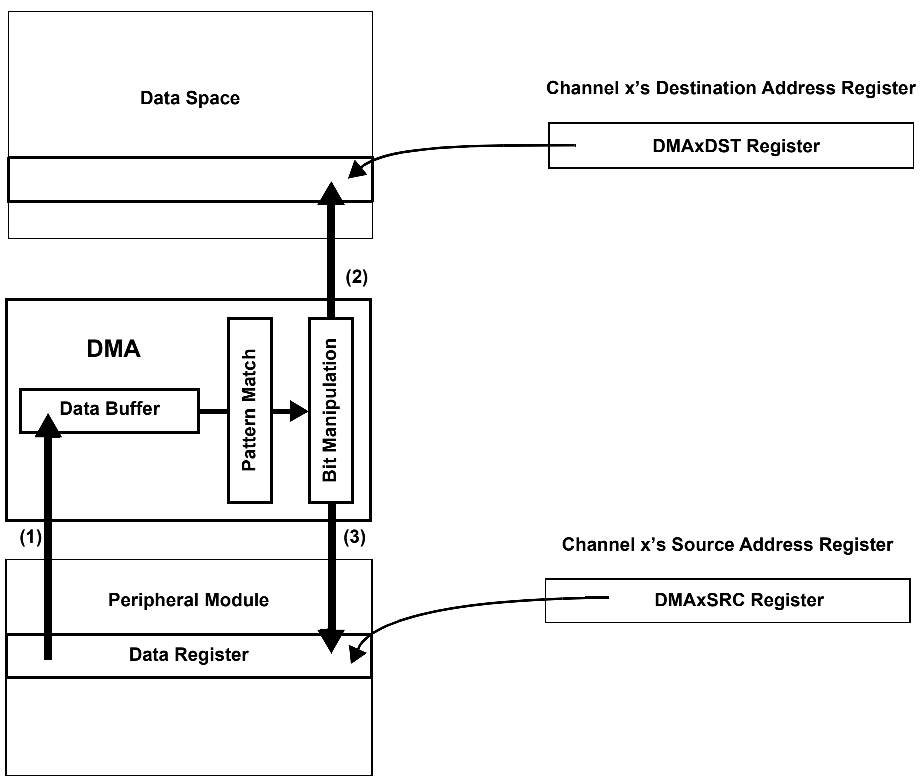
Some communication protocols require symmetrical buffer accesses; that is, for every read operation performed on a buffer, there must be an accompanying write operation. An example of this requirement occurs with the SPIx module operating in Host mode. When the SPIx module is configured for Host mode, and only received data are of interest, some data must be written to the SPIx transmit buffer in order to start the SPIx clock and receive the external data. In this situation, use the Null Write mode of the DMA.

The Null Write mode is used to satisfy this requirement. This mode works by transferring data from the address in DMAxSRC to the address in DMAxDST, like any other DMA operation. Once this is done, however, the transferred data that are still stored in DMABUF are written back to the address specified by DMAxSRC. The write-back occurs before the DMA proceeds to its next transfer. A typical example of this is shown in Figure 13-5. Null Write mode is enabled by setting the flow control bits to '01' (FLWCON[1:0] = 01).

Note: In Null Write mode, the DBUFWF bit is clear only upon the completion of the write to the location specified in DMAxSRC.

Figure 13-5 illustrates the basic DMA data flow in Null Write mode. In Null Write mode, both load and store transactions are initiated at the address location specified in DMAxSRC.

Figure 13-5. Null Write Mode



Notes:
1. Read from the address location specified in the DMAxSRC register and set the DBUFWF bit.
2. Write to the address location specified in the DMAxDST register.
3. Write to the address location specified in the DMAxSRC register and clear the DBUFWF bit.
4. Repeat until completion.

Example 13-6 shows sample code for Null Write Mode working with SPI. In this configuration, no DMA request is issued until the first block of SPIx data is received. However, in Host mode, no data are received until the SPIx transmits first. To initiate the DMA transfers, the user application must use DMA Null Write mode and start DMA Manual Transfer mode.

Example 13-6. Null Write Mode

```
#include <xc.h>
#include<stdint.h>

uint32_t RXREG[4];
uint32_t temp = 0xABCD;

#define BRG      (uint32_t) (1000)

void DMA_Init(void);
void SPI1_Host_Init(void);

void SPI1_Host_Init(void)
{
    //Both RX and TX pins are mapped to same pin for LoopBack
    _RP27R = 22;                    //re-mapping RP27 as SCK2
    _RP42R = 21;                    //re-mapping RP42 as SDO2
    _SDI1R = 42;                    //re-mapping RP42 as SDI2

    SPI1CON1bits.MSTEN    = 1;       //Host mode
    SPI1BRG = BRG; // use FPB/4 clock frequency
    SPI1CON1bits.MODE16   = 0;
    SPI1IMSKbits.SPIRBFEN  = 1;      //Transmit buffer empty generates an
interrupt event
    SPI1CON1bits.SPIEN     = 1;      //Enables module
}

void DMA_Init(void)
{
    DMACONbits.ON=1;                //DMA module is enabled

    DMAHIGH=0x5000;                 //set lower and upper address limit
    DMALOW=0x0000;

    DMA0CNT = 3;                    // Number of bytes received
    DMA0CHbits.SIZE = 0; //8bit

    DMA0SRC=(uint32_t)&SPI1BUF;    // Source address
    DMA0DST=(uint32_t)&RXREG;      // Destination address

    DMA0CHbits.SAMODE = 0; // DMAxSRC remains unchanged after a transfer
completion
    DMA0CHbits.DAMODE = 1; // DMAxDST is incremented based on the SIZE bit
after a transfer completion
    DMA0CHbits.TRMODE = 0; // Transfer mode,One-Shot mode
    DMA0CHbits.FLWCON = 0b01; //Null-Write mode

    DMA0SELbits.CHSEL = 0x6; //Trigger on SPI1 Receive Interrupt
    DMA0CHbits.CHEN   = 1;  //DMA Channel is enabled
}

int main(void)
{
    DMA_Init();
    SPI1_Host_Init();
    SPI1BUF = 0xAB; //The first write needs to be initiated by SPI (Master)
for receive to happen
    while(!DMA0STATbits.DONE); // Indicates DMA Channel 0 has completed all
the transactions
    while(1);
}
```

13.4.9.3. DMA ECC Mode

ECC (Error Correction Code) mode in the DMA controller ensures data integrity by detecting and correcting single-bit errors and reporting double-bit errors during memory transfers. The DMA works alongside the BMX ECC check to enhance this feature. After enabling ECC by setting

BMXECCXCONbits.ON = 1, error detection is activated. The ESEL (Error Reporting Select) bit configures the reporting of errors based on Single Error Correction (SEC) and Double Error Detection (DED). Single-bit errors are automatically corrected and logged in the BMXECCXSTATbits.SEC register, while double-bit errors are reported in the BMXECCXSTATbits.DED.

Refer to Error Correcting Code (ECC) for additional Fault injection information.

13.4.10. Ping-Pong

With its capability to seamlessly transfer data flow between its internal DMA channels, the 32-bit DMA Controller supports ping-pong operation to assist the CPU in effective and uninterrupted data transfer. By alternating the incoming data flow between two active DMA channels, the CPU can process the available data from one DMA channel while the other active DMA channel is making its own data available to be processed when the CPU has completed its current task.

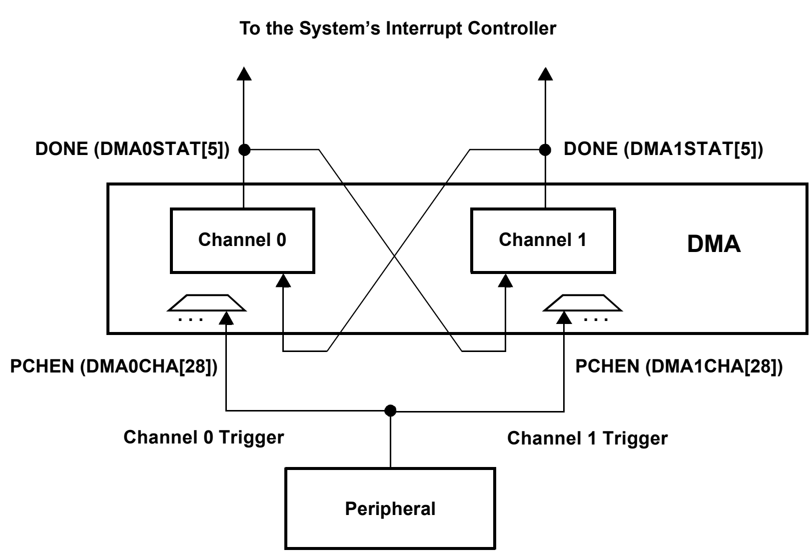
In DMA controllers with Ping-Pong mode, channels are paired in fixed combinations such as (0,1), (2,3) and (4,5). Either channel in each pair serves as the initiator channel enabling continuous data transfer. The initiator channel is determined by setting the PCHEN to 1. Each channel alternates between transferring data and preparing for the next transfer, allowing for an uninterrupted data stream with minimal CPU intervention.

The capability relies on proper hardware/software interactions between the CPU and the two DMA channels, 0 and 1; see Figure 13-6. This is configured by setting CHEN (DMAxCH[0]) = 1 and PPEN (DMAxCH[29]) = 1 for both channels. Initially, before the ping-pong operation begins, PCHEN should be set by user software explicitly for the channel that initiates the ping-pong operation.

Each channel of the pair requires its own individual trigger to facilitate transfer. Each channel must be individually set up with its own control settings, source and destination addresses, transfer count and trigger. The channels operate independently in terms of setup, but they are linked in operation through the ping-pong mechanism.

When one DMA channel completes its operation, it triggers the other active channel's hardware enable input. This action sets the other channel's PCHEN bit (DMAxCH[28]) high. Whenever both PCHEN and CHEN are high (and PPEN = 1), the DMA channel is enabled and ready for data transfer when triggered. This event effectively transfers the data flow from one channel to the other.

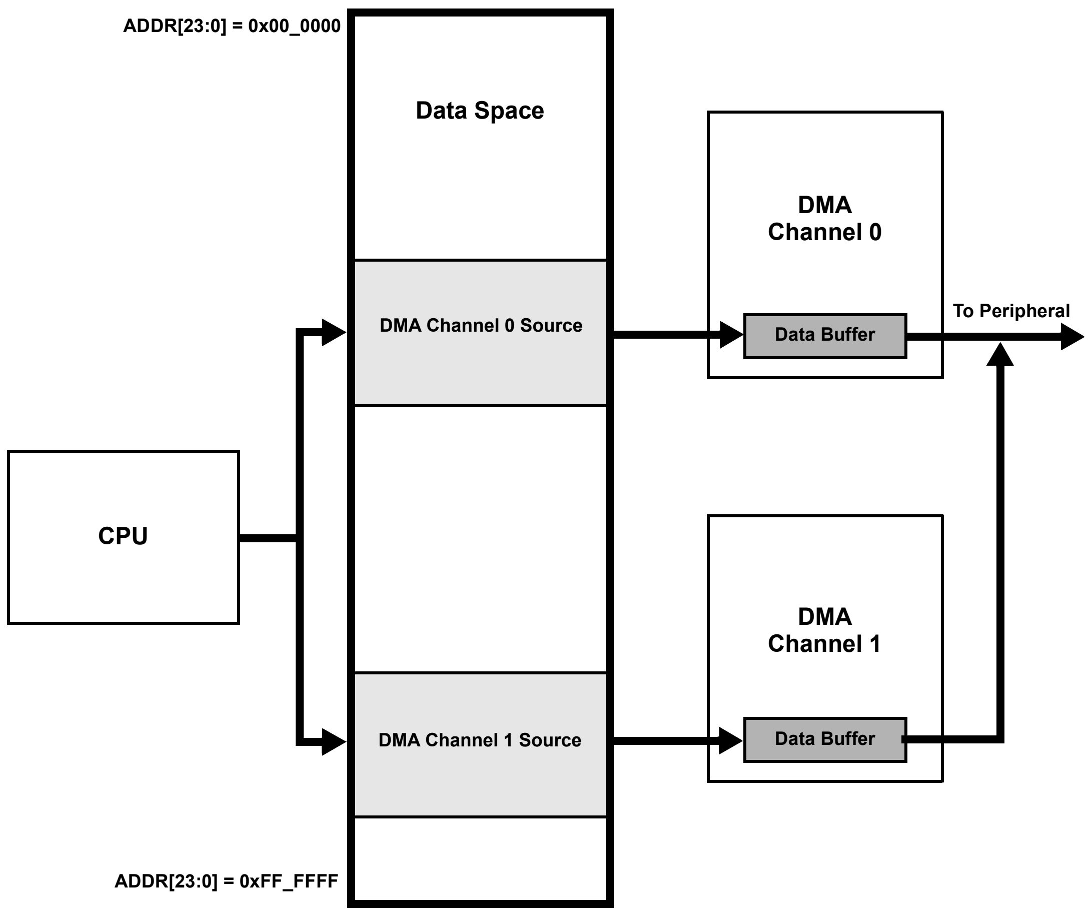
Figure 13-6. Ping-Pong Support Connection



13.4.10.1. Ping-Pong Data Transmit

The ping-pong operation setup, where the DMA controller alternates transmission of outgoing data from two different buffers to a peripheral, is shown in Figure 13-7.

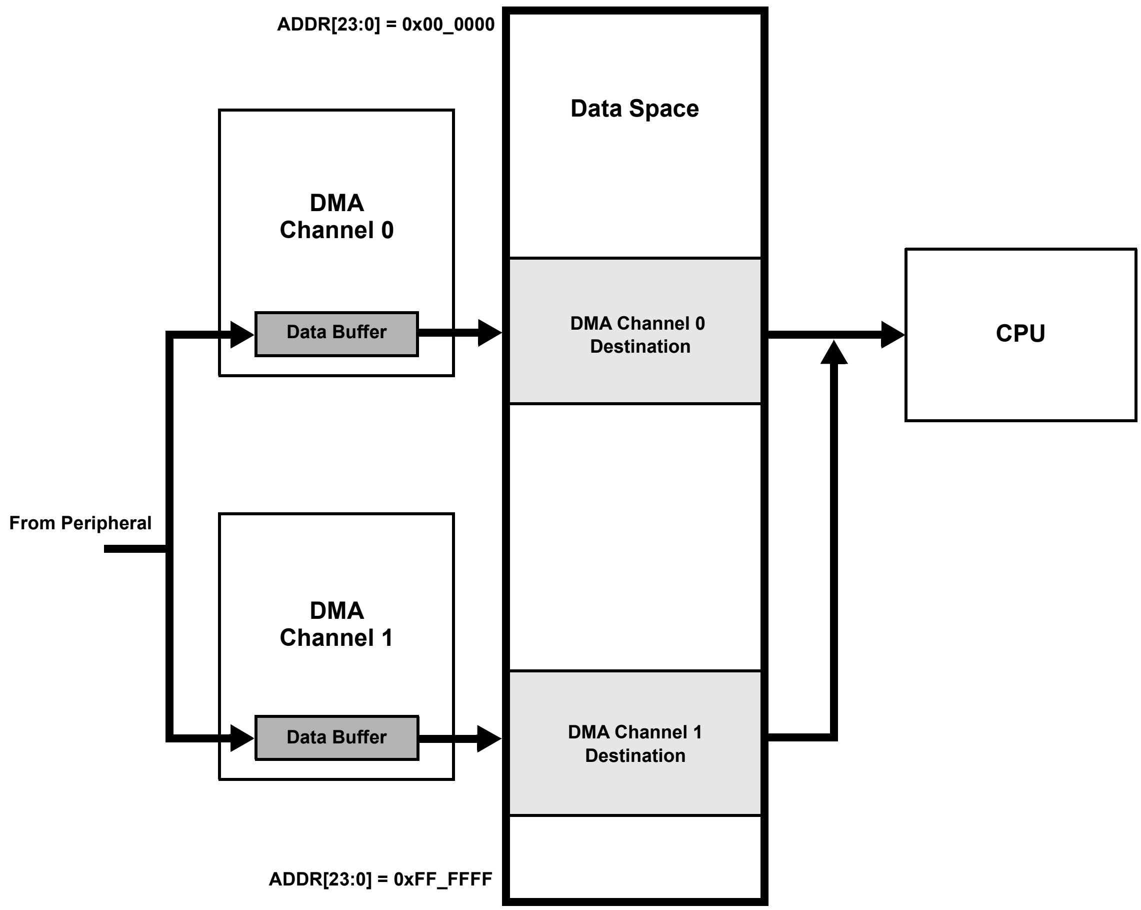
Figure 13-7. Ping-Pong – Transmit



13.4.10.2. Ping-Pong Data Receive

The ping-pong operation setup, where the DMA controller alternates incoming data between two memory buffers, can be seen in Figure 13-8.

Figure 13-8. Ping-Pong – Receive



13.4.11. Bit Manipulation

The 32-bit DMA Controller can perform real-time bit manipulation on the 32-bit data word being moved from the memory-mapped source to the destination location. The bit manipulation is made up of invert, clear and set functions, connected with their associated mask register, and daisy-chained together. Note that each DMA channel is equipped with a one-bit manipulation logic block, and bit manipulation is applied to the DMABUF[31:0] bits during the write cycle, when the data are being stored to the destination location, once the bandwidth has been allocated to the requesting DMA channel.

Each mask register bit (see DMAxINV, DMAxCLR and DMAxSET) is responsible for enabling its bit manipulation function onto the corresponding data bit, where logic '0' maintains the original bit value of the input data. The data coming out of the bit manipulation logic block can be expressed as:

```
{[DMABUF[n] XOR DMAINVx[n]]

AND ~DMACLRx[n]}

OR DMASETx[n]
```

In Byte mode or Word mode, DMA bit manipulation does not automatically account for byte or Word mode settings. For instance, to set bit 0 in Byte mode, bits 0, 8, 16 and 24 may need to be set.

In Byte mode, DMAxSET operates such that [0:7] = [15:8] = [23:16] = [31:24], whereas in Word mode, DMAxSET spans [15:0] = [31:16]. This approach also applies to the DMAxINV and DMAxCLR registers.

13.4.11.1. Priority

The set function is given the highest priority, followed by clear and invert, respectively. This means if the same bit position is enabled for more than one bit manipulation function, the higher priority function becomes dominant. As a result, the output data's logic value only reflects the highest bit manipulation function.

13.4.11.1.1. Channel Chaining

Channel Chaining is used to chain two or more channels together such that once one channel in the sequence—often referred to as the initiating channel—completes its designated task and sets its DONE flag, it sends a signal indicating completion. This signal acts as a trigger for the subsequent channel, known as the chained channel.

Refer to DMA Trigger Sources for details on selecting one channel as the trigger source for another.

13.4.12. Pattern Match

When the content of the incoming data is required in making decisions in real time, the 32-bit DMA Controller can recognize a data pattern in its internal buffer being transferred from the source to the destination locations. The pattern match capability, when enabled, allows a user-programmable data pattern to be compared against a (partial) content of DMABUF[31:0]. Upon match detection, the DMA Controller invokes its interrupt to inform the CPU to take further action. This feature is implemented with control, status and pattern with its corresponding mask registers, along with digital control and comparator units, as shown in Figure 13-9.

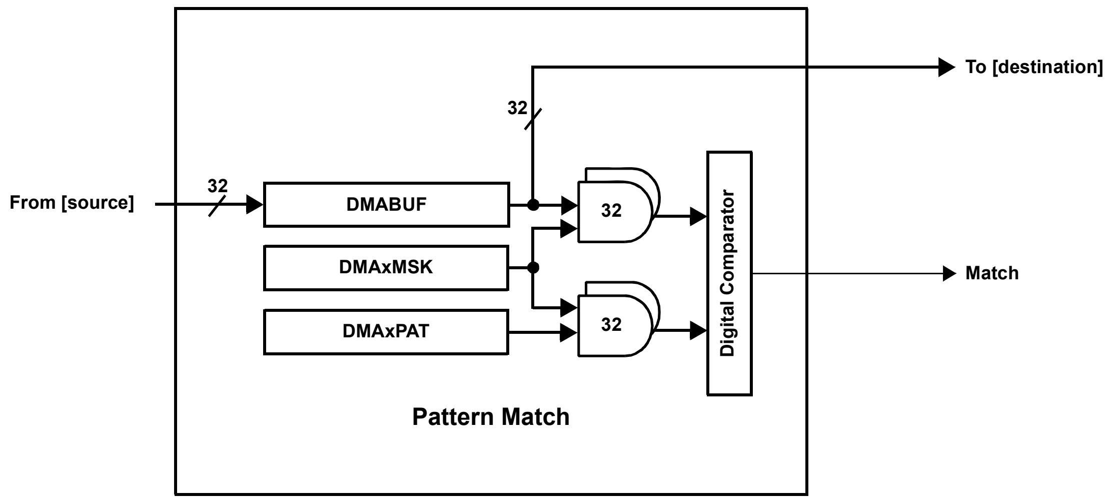
13.4.12.1. Real-Time Pattern Matching

The MATCHEN bit (DMAxCH[2]) is used to enable the feature per selected channel; see DMAxCH. Once enabled, the contents of DMABUF[31:0] and DMAxPAT[31:0] are subjected to the corresponding DMAxMSK[31:0] bits value and compared against each other in an array of two-input digital comparators, as shown below. When a match is detected, the DMA Controller proceeds to invoke its channel interrupt output while setting the MATCH bit (DMAxSTAT[1]), accordingly. Note that the DMA operation continues until completion, unaffected by the real-time matching. Upon interrupt detection by the CPU, the user's software is expected to start processing the incoming data until it comes across the pattern. It may also be necessary to update the DMAxPAT[31:0]/ DMAxMSK[31:0] bits for proper pattern matching associated with the current incoming data stream.

Notes:

1. When a match is first detected, the data still reside in the Data Buffer register. Depending on the bus infrastructure, it takes a varying number of clock cycles for it to reach the destination. It is up to the user's software to manage this latency period to properly recover the correct data.
2. It is up to the user's software to timely clear the MATCH bit (DMAxSTAT[1]) in order to prevent an overrun-like condition, where multiple matches occur with no CPU response. Currently, there is no capability to flag this condition.
3. The pattern match feature does not take SIZE[1:0] into account, so it is left to the user's software to set the mask accordingly.

Figure 13-9. Pattern Match



Note: The diagram is intended for conceptual illustration only.

13.5. Descriptor-Based Operation

In addition to conventional operation, the DMA module is equipped with a descriptor based operational capability that further provides additional flexibility to its feature set. This includes:

- Indirect Addressing mode
- Conditional branch
- Descriptor write back

The descriptor-based operation can be enabled independently per channel by setting DMAxCH.SDTEN or DMAxCH.DDTEN or both. This allows some channels to operate in Conventional mode while others in Descriptor-Based mode depending on the needs of the user application.

When enabled, the DMA module begins each data transfer session by “fetching” a descriptor from a memory mapped location. Then it determines various aspects of its addressing and data flow operations based on the content of the descriptor before it proceeds to complete the operation. Note that with one descriptor corresponding to one data transfer, a descriptor is required for each session.

In this mode, the DMA module relies on pre-allocated memory contents known as descriptors which are typically memory mapped sequentially into a block called Descriptor Table (DT). There are two types of Descriptor Tables: Source Descriptor Table (SDT) and Destination Descriptor Table (DDT). The DMA module also allows the flexibility of utilizing one or the other or both conforming to one consistent format.

To access each descriptor, the module employs a Descriptor Pointer (DP). It can be placed in the DMAxSRC.SADDR[23:0] and DMAxDST.DADDR[23:0]) registers initially to locate the Descriptor Table’s base address (first DP) and during operation automatically to locate the next descriptor (next DP). Note that in the Conventional mode of operation, both DMAxSRC.SADDR[23:0]/ DMAxDST.DADDR[23:0] registers always contain their respective Payload Pointer. In the descriptor-based operation, they can contain either the Payload Pointer or the Descriptor Pointer depending on the phase of the operation.

13.5.1. Descriptor Format

A descriptor is a 32-bit entry in a Descriptor Table. It consists of three sections based on the functionality as follows:

- Payload Pointer: *SADDRn[15:0]/DADDRn[15:0]*
- Transfer Size: *SIZEn[1:0]*
- Descriptor Pointer Offset: DPOnM1[6:0], DPOnM0[6:0]

The structure of a Descriptor Table is as follows:

Table 13-9. Descriptor Table

Memory Position	Descriptor			
	[31:25]	**[24:18]**	**[17:16]**	**[15:0]**
1	DPO1M1[6:0]	DPO1M0[6:0]	SIZE1[1:0]	Payload Pointer
2	DPO2M2[6:0]	DPO2M0[6:0]	SIZE2[1:0]	Payload Pointer
3	DPO3M3[6:0]	DPO3M0[6:0]	SIZE3[1:0]	Payload Pointer
...	...			

13.5.1.1. Payload Pointer

Descriptor[15:0] contains the address pointer used to transfer the payload as follows. See Source Descriptor Table (SDT) and Destination Descriptor Table (DDT).

- *SADDRn[15:0]* is the Source Payload Pointer (SPP) to be placed in DMASRCx.SADDR[23:0]
- *DADDRn[15:0]* is the Destination Payload Pointer (DPP) to be placed in DMASRCx.DADDR[23:0]

13.5.1.2. Transfer Size

Descriptor[17:16] contains the transfer size bits as follows. See section 14.11.2.1 for size conflict handling.

- 11 = Reserved
- 10 = One 32-bit word is transferred at a time
- 01 = One 16-bit word is transferred at a time
- 00 = One byte is transferred at a time

Note that *SIZEn[1:0]* overrules DMAxCH.SIZE[1:0] when its descriptor table is enabled.

13.5.1.3. Descripter Pointer Offset

Descriptor[31:18] contains the Descriptor Pointer Off set (DPO) values used to calculate the address location of the next descriptor by simply adding it to the address location of the current descriptor. See Source Descriptor Table (SDT) and Destination Descriptor Table (DDT).

Next DP = Current DP + DPO

- DPOnM1[6:0] is the Descriptor Table Offset used to calculate the next Descriptor Pointer only if DMAxSTAT.MATCH = `1`, see Table 13-10.
- DPOnM0[6:0] is the Descriptor Table Offset used to calculate the next Descriptor Pointer only if DMAxSTAT.MATCH = `0`, see Table 13-10.

Table 13-10. Descriptor Pointer Offset

DPO[6:0] (Hex/2's Complement)	Offset Value (Decimal)
00/0	0
01/+1	+4
02/+2	+8
03/+3	+12

Table 13-10. Descriptor Pointer Offset (continued)

DPO[6:0] (Hex/2's Complement)	Offset Value (Decimal)
...	...
0F/+15	+60
...	...
3F/+63	+252
40/-64	-256
41/-63	-252
...	...
7F/-1	-4

13.5.2. Source Descriptor Table (SDT)

The Source Descriptor Table can be enabled independently by setting the DMAxCH.SDTEN bit high. With DMAxSRC.SADDR[23:0] preloaded with the Source Descriptor Table's base address (first DP) by the application, the DMA module is ready for its source descriptor-based operation.

The code below is an example of a source descriptor. The descriptor table is created using a structure with padding, as shown. An example is shown in the table below.

Table 13-11. Source Descriptor Example

OffsetIfMatch (7 bits)	OffsetNoMatch (7 bits)	Size (2 bits)	Payload Pointer (16 bits)
3	2	0x2	DMA_Src1
-1	3	0x2	DMA_Src2
-1	1	0x2	DMA_Src3
3	1	0x2	DMA_Src4
-3	1	0x2	DMA_Src4

The DMA begins by reading from the payload pointer (e.g., DMA_Src1) after the source pointer is set to the starting address of the first descriptor (DMA_SrcPtr[0]) and SDTEN is enabled. Following this transaction, the DMA checks if pattern matching is active (MATEN = `1`). If a match is detected, it uses the OffsetIfMatch value to move to the next descriptor. If no match is found, the OffsetNoMatch value is used instead.

```
#include <xc.h>
#include<stdlib.h>
#include<stdint.h>

//Source address for DMA
uint32_t DMA_Src1 __attribute((address(0x5000)))= 0xa1a2a3a4;
uint32_t DMA_Src2 __attribute((address(0x5004)))= 0xa1a2a3a4;
uint32_t DMA_Src3 __attribute((address(0x5008)))= 0xb2b2b2b2;
uint32_t DMA_Src4 __attribute((address(0x500C)))= 0xb2b2b2b2;
uint32_t DMA_Src5 __attribute((address(0x5010)))= 0xa1a2a3a4;

//Dest Address for DMA
uint32_t DMA_Dst[5] __attribute((address(0x4500)))= {0};

typedef struct DescriptorTable
{
    uint32_t PayloadPointer:16;
    uint32_t Size:2;
    int32_t OffsetNoMatch:7;
    int32_t OffsetIfMatch:7;
}Src_DP;


Src_DP DMA_SrcPtr[5] __attribute__((address(0x5500)));

int main(void)
{
    //row 1
```

```
    DMA_SrcPtr[0].PayloadPointer = (uint16_t)&DMA_Src1;
    DMA_SrcPtr[0].Size = 0x2;
    DMA_SrcPtr[0].OffsetNoMatch = 2;
    DMA_SrcPtr[0].OffsetIfMatch =  3;
    //row 2
    DMA_SrcPtr[1].PayloadPointer = (uint16_t)&DMA_Src2;
    DMA_SrcPtr[1].Size = 0x2;
    DMA_SrcPtr[1].OffsetNoMatch = 3;
    DMA_SrcPtr[1].OffsetIfMatch = -1;
    //row 3
    DMA_SrcPtr[2].PayloadPointer = (uint16_t)&DMA_Src3;
    DMA_SrcPtr[2].Size = 0x2;
    DMA_SrcPtr[2].OffsetNoMatch = 1;
    DMA_SrcPtr[2].OffsetIfMatch = -1;
    //row 4
    DMA_SrcPtr[3].PayloadPointer = (uint16_t)&DMA_Src4;
    DMA_SrcPtr[3].Size = 0x2;
    DMA_SrcPtr[3].OffsetNoMatch = 1;
    DMA_SrcPtr[3].OffsetIfMatch = 3;
   //row 5
    DMA_SrcPtr[4].PayloadPointer = (uint16_t)&DMA_Src5;
    DMA_SrcPtr[4].Size = 0x2;
    DMA_SrcPtr[4].OffsetNoMatch = 1;
    DMA_SrcPtr[4].OffsetIfMatch = -3;


    DMACONbits.ON = 0;
    DMA0CH = 0;
    DMAHIGH = 0xFFFFFF;     //Sets the DMA High limit Address
    DMALOW = 0x4000;           //Sets the DMA Low Limit Address
    //Source Descriptor Enable
    DMA0CHbits.SDTEN = 1;

    DMA0PAT = 0xb2b2b2b2;             //Configures the pattern register for pattern match
    DMA0MSK = 0xFFFFFFFF;            //Configures the pattern mask register for pattern match
    DMA0CHbits.MATEN = 1;             //Enables Pattern match
    DMA0CHbits.SAMODE = 0;          //Source address mode unchanged
    DMA0CHbits.DAMODE = 1;        //Destination addr mode increment
    DMA0CHbits.TRMODE = 1;          //Repeated One-Shot Mode
    DMA0CHbits.SIZE = 2;                  //Transfer size of 32-bit

    DMA0SRC = (uint32_t)&DMA_SrcPtr[0];    //Sets source address as first Descriptor
(DMA_SrcPtr[0])
    DMA0DST = (uint32_t)&DMA_Dst[0];          //Sets destination address
    DMA0CNT  = 4;                                                    //Count value of 4 */
    DMA0CHbits.CHEN = 1;      //Enables Channel
    DMACONbits.ON = 1;           //Enables DMA module

    //First Trigger (No match)
    DMA0CHbits.CHREQ = 1;
    while(DMA0CHbits.CHREQ == 1);

    //Second Trigger (Match))
    DMA0CHbits.CHREQ = 1;
    while(DMA0CHbits.CHREQ == 1);

    //Third Trigger (No match))
    DMA0CHbits.CHREQ = 1;
    while(DMA0CHbits.CHREQ == 1);


    while(1);
    return 1;
}
```

13.5.3. Destination Descriptor Table (DDT)

The Destination Descriptor Table can be enabled independently by setting the DMAxCH.DDTEN bit high. With DMAxSRC.DADDR[23:0] preloaded with the Destination Descriptor Table's base address (first DP) by the application, the DMA module is ready for its destination descriptor-based operation.

The following code is an example of a destination descriptor. The descriptor table is created using a structure with padding, as shown. An example is provided below.

Table 13-12. Destination Descriptor Example

OffsetIfMatch (7 bits)	OffsetNoMatch (7 bits)	Size (2 bits)	Payload Pointer (16 bits)
3	2	0x2	DMA_Dst[0]
-1	3	0x2	DMA_Dst[1]
-1	1	0x2	DMA_Dst[2]
3	1	0x2	DMA_Dst[3]
-3	1	0x2	DMA_Dst[4]

The DMA begins by reading from the payload point (e.g., DMA_Dst[0]) after the destination pointer is set to the starting address of the first descriptor (DMA_DestPtr[0]) and DDTEN is enabled. Following this transaction, the DMA checks if pattern matching is active (MATEN = `1`). If a match is detected, it uses the OffsetIfMatch value to move to the next descriptor. If no match is found, the OffsetNoMatch value is used instead.

```
#include <xc.h>
#include<stdlib.h>
#include<stdint.h>
//Source address for DMA
uint32_t DMA_Src1 __attribute((address(0x5000)))= 0xa1a2a3a4;
uint32_t DMA_Src2 __attribute((address(0x5004)))= 0xc1c1c1c1;
uint32_t DMA_Src3 __attribute((address(0x5008)))= 0xa1a2a3a4;
uint32_t DMA_Src4 __attribute((address(0x500C)))= 0xb2b2b2b2;
uint32_t DMA_Src5 __attribute((address(0x5010)))= 0xa1a2a3a4;
//Dest Address for DMA
uint32_t DMA_Dst[5] __attribute((address(0x4500)))= {0};
typedef struct DescriptorTable
{
    uint32_t PayloadPointer:16;
    uint32_t Size:2;
    int32_t OffsetNoMatch:7;
    int32_t OffsetIfMatch:7;
}Dest_DP;
Dest_DP DMA_DestPtr[5] __attribute__((address(0x5500)));

int main(void)
{
    //row 1
    DMA_DestPtr[0].PayloadPointer = (uint16_t)&DMA_Dst[0];
    DMA_DestPtr[0].Size = 0x2;
    DMA_DestPtr[0].OffsetNoMatch = 2;
    DMA_DestPtr[0].OffsetIfMatch =  3;
    //row 2
    DMA_DestPtr[1].PayloadPointer = (uint16_t)&DMA_Dst[1];
    DMA_DestPtr[1].Size = 0x2;
    DMA_DestPtr[1].OffsetNoMatch = 3;
    DMA_DestPtr[1].OffsetIfMatch = -1;
    //row 3
    DMA_DestPtr[2].PayloadPointer = (uint16_t)&DMA_Dst[2];
    DMA_DestPtr[2].Size = 0x2;
    DMA_DestPtr[2].OffsetNoMatch = 1;
    DMA_DestPtr[2].OffsetIfMatch = -1;
    //row 4
    DMA_DestPtr[3].PayloadPointer = (uint16_t)&DMA_Dst[3];
    DMA_DestPtr[3].Size = 0x2;
    DMA_DestPtr[3].OffsetNoMatch = 1;
    DMA_DestPtr[3].OffsetIfMatch = 3;
   //row 5
    DMA_DestPtr[4].PayloadPointer = (uint16_t)&DMA_Dst[4];
    DMA_DestPtr[4].Size = 0x2;
    DMA_DestPtr[4].OffsetNoMatch = 1;
    DMA_DestPtr[4].OffsetIfMatch = -3;

    DMACONbits.ON = 0;
    DMA0CH = 0;
    DMAHIGH = 0xFFFFFF;   //Sets the DMA High limit Address
    DMALOW = 0x4000;      //Sets the DMA Low Limit Address
    //Destination Descriptor Enable
    DMA0CHbits.DDTEN = 1;

    DMA0PAT = 0xc1c1c1c1;   //Configures the pattern register for pattern match
    DMA0MSK = 0xFFFFFFFF;   //Configures the pattern mask register for pattern match
```

```
    DMA0CHbits.MATEN = 1;   //Enables Pattern match
    DMA0CHbits.SAMODE = 1;  //Source address mode increment
    DMA0CHbits.DAMODE = 0;  //Destination address mode unchanged
    DMA0CHbits.TRMODE = 1;  //Repeated One-Shot Mode
    DMA0CHbits.SIZE = 2;    //Transfer size of 32-bit
    DMA0SRC = (uint32_t)&DMA_Src1;       //Sets Source address
    DMA0DST = (uint32_t)&DMA_DestPtr[0]; //Sets destination address as first Descriptor
(DMA_DestPtr[0])
    DMA0CNT  = 4;          //count value of 4
    DMA0CHbits.CHEN = 1;   //Enables channel
    DMACONbits.ON = 1;     //Enables DMA module
    //First Trigger (No match)
    DMA0CHbits.CHREQ = 1;
    while(DMA0CHbits.CHREQ == 1);
    //Second Trigger (Match))
    DMA0CHbits.CHREQ = 1;
    while(DMA0CHbits.CHREQ == 1);
    //Third Trigger (No match))
    DMA0CHbits.CHREQ = 1;
    while(DMA0CHbits.CHREQ == 1);

    while(1);
    return 1;
}
```

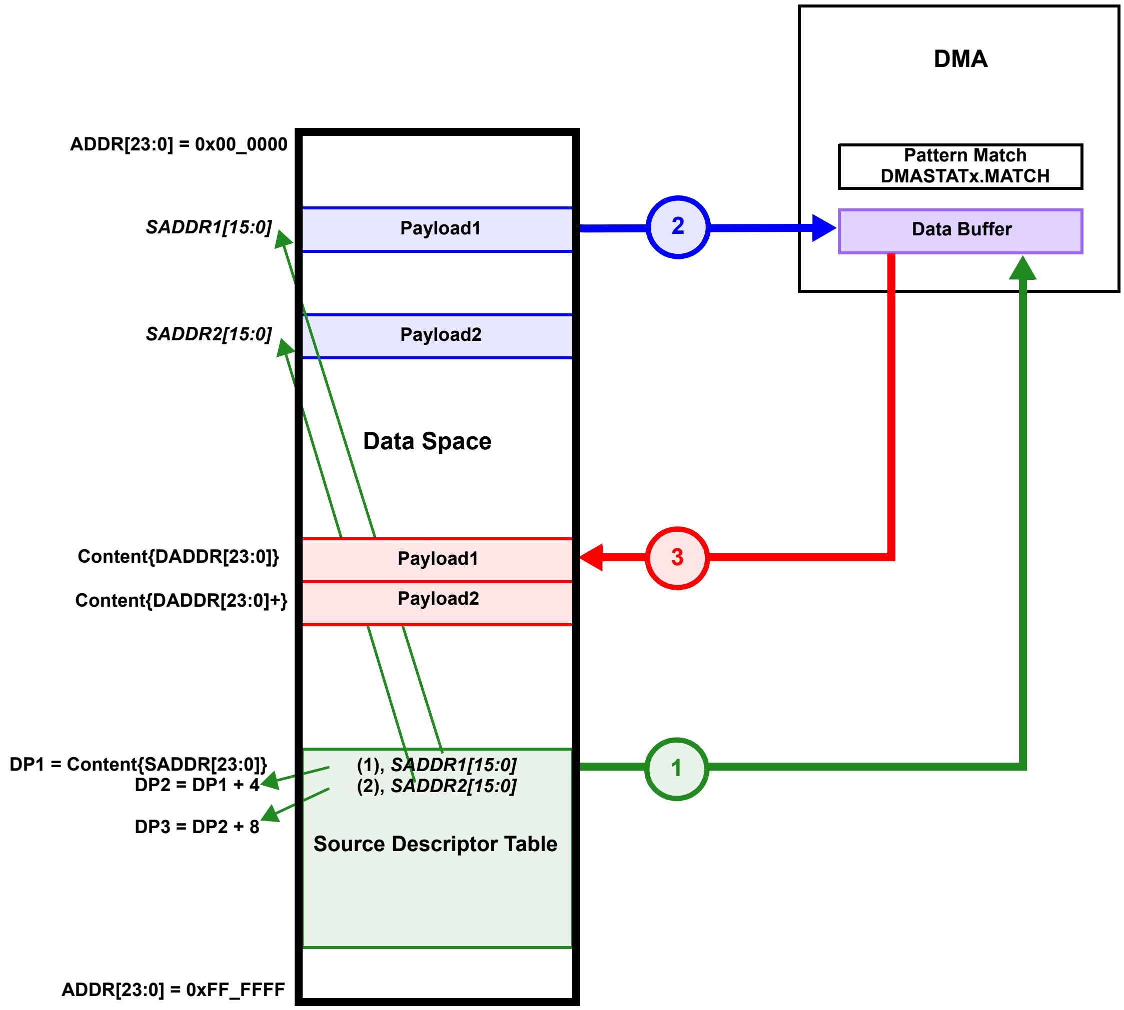
13.5.4. Indirect Addressing Operation

In the descriptor-based operation, DMA begins its operation upon triggered by fetching its descriptor(s) to retrieve the Payload Pointer(s) to be used for the data transfer, and the Descriptor Pointer Offsets to be used to determine the next Descriptor Pointer. It then proceeds to perform its data transfer operation by reading the payload from the location given in the Source Payload Pointer (SPP) or from the descriptor if the Source Descriptor Table (SDT) is enabled, and subsequently writing it to the location given in the Destination Payload Pointer (DPP) or from the descriptor if the Destination Descriptor Table (DDT) is enabled.

To properly support this mode of operation, at least one Descriptor Table must be preloaded into the memory mapped locations with the appropriate Descriptor Pointer(s) also programmed into the SADDR[23:0] or DADDR[23:0] registers or both. All other control and configuration registers must also be programmed accordingly.

Figure 13-10 illustrates the flow of operation with an example when only the Source Descriptor Table (SDT) is enabled.

Figure 13-10. SDT-Enabled Operation - Random to Block Locations

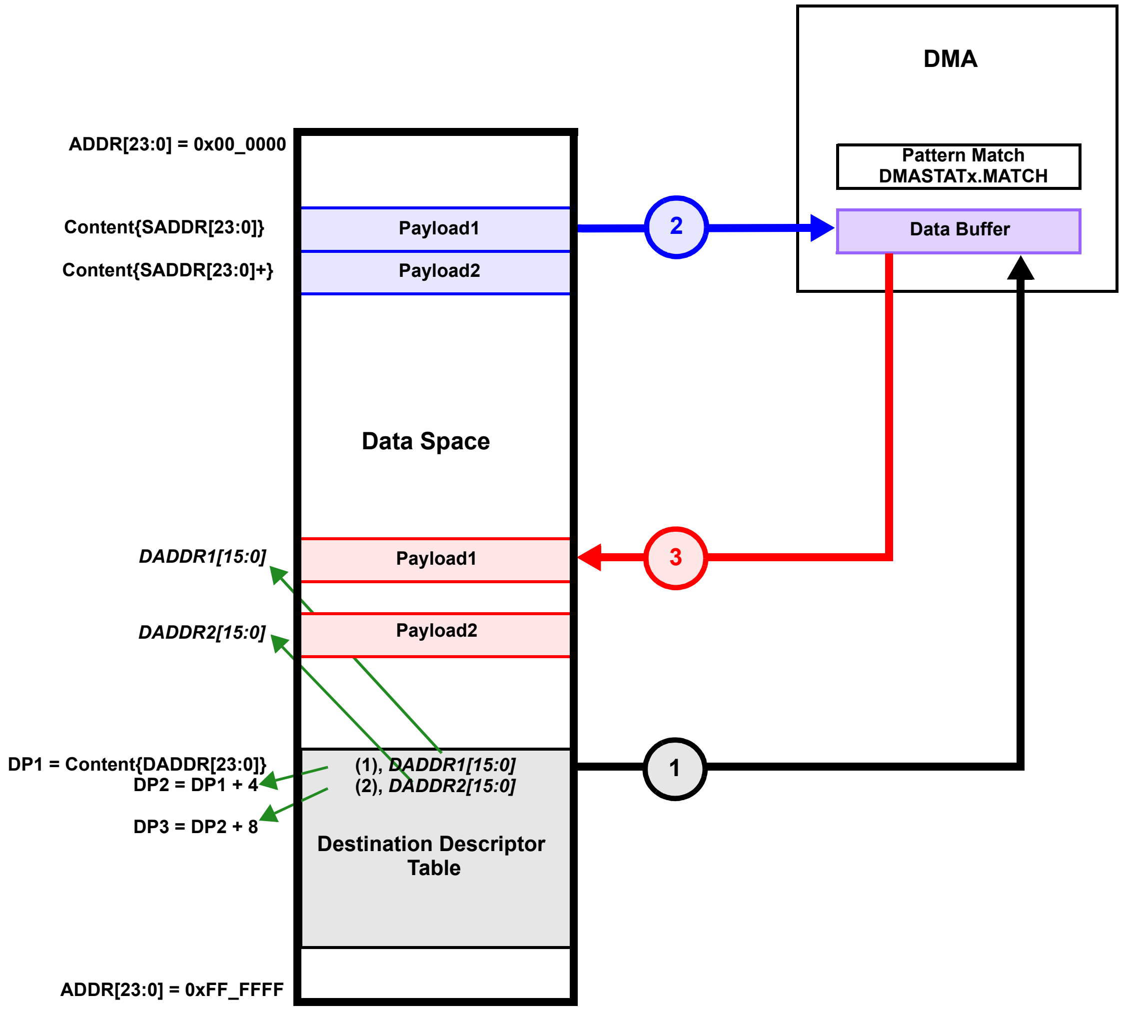


Notes:

1. DPO1M1 = DPO1M0 = 1 for +4, *SIZE1[1:0]* = 10 for 32-bit transfer
2. DPO2M1 = DPO2M0 = 2 for +8, *SIZE2[1:0]* = 10 for 32-bit transfer

Figure 13-11 illustrates the flow of operation with an example when only the Destination Descriptor Table is enabled. This essentially follows the transaction of the indirect addressing operation with the extracted Payload Pointer being completely independent from one another, thus allowing for random access of any memory mapped locations.

Figure 13-11. DDT-Enabled Operation - Block to Random Locations



Notes:

1. DPO1M1 = DPO1M0 = 1 for +4, *SIZE1[1:0]* = 10 for 32-bit transfer
2. DPO2M1 = DPO2M0 = 2 for +8, *SIZE2[1:0]* = 10 for 32-bit transfer

With the descriptor-based operation utilizing indirect addressing, random locations can be accessed consecutively by one single DMA channel without requiring any CPU intervention. This helps to protect the precious system resources.

13.5.5. Conditional Branch

With DMAxSTA.MATCH taken into account in calculating the next Descriptor Pointer (DP), the operational flow can be made dependent on the content of the payload. This conditionality along with the Descriptor Pointer Offset effectively yields the conditional branch operation, see Table 13-13 and Table 13-14 This can be also used to build programming loops, making DMA much more independent of the CPU in processing the peripheral status bits and the corresponding data transfer.

In a typical DMA operation, the module can be configured to test a status bit using its pattern matching capability upon trigger. Then based on the logic state of the selected bit(s) resulting in DMAxSTA.MATCH which affects the next Descriptor Pointer, it would be able to branch to an appropriate data transfer routine with or without address gaps in between transfers, without any CPU intervention.

Table 13-13. Next Source Descriptor Pointer Value Determination(1), (2), (3), (4)

Memory Position	Memory Address Location (DP as a Product of Register Content and Arithmetic Calculation)	Descriptor[31:0]			
		[31:25]	**[24:18]**	**[17:16]**	**[15:0]**
1	SADDR[23:0]	DPOxM1[6:0]	DPOxM0[6:0]	SIZEx[1:0]	SADDRx[15:0]
...	...	...	...	...	...
n	SADDR[23:0] = SADDR[23:0] + DPOxM1 if MATCH = 1 else SADDR[23:0] + DPOxM0 is MATCH = 0	DPOyM1[6:0]	DPOyM0[6:0]	SIZEy[1:0]	SADDRy[15:0]
...	...	...	...	...	...
m	SADDR[23:0] = SADDR[23:0] + DPOyM1 if MATCH = 1 else SADDR[23:0] + DPOyM0 is MATCH = 0	DPOzM1[6:0]	DPOzM0[6:0]	SIZEz[1:0]	SADDRz[15:0]
...	...	...	...	...	...

Notes:
1. Descriptor fetching is subject to memory boundary restriction and address fault detection.
2. Descriptor fetching is a 32-bit word operation.
3. SADDR[23:0] indicates a register content.
4. SADDRn[15:0] indicates a payload pointer value.

Table 13-14. Next Destination Descriptor Pointer Value Determination(1), (2), (3), (4)

Memory Position	Memory Address Location (DP as a Product of Register Content and Arithmetic Calculation)	Descriptor[31:0]			
		[31:25]	**[24:18]**	**[17:16]**	**[15:0]**
1	DADDR[23:0]	DPOxM1[6:0]	DPOxM0[6:0]	SIZEx[1:0]	DADDRx[15:0]
...	...	...	...	...	...
n	DADDR[23:0] = DADDR[23:0] + DPOxM1 if MATCH = 1 else DADDR[23:0] + DPOxM0 is MATCH = 0	DPOyM1[6:0]	DPOyM0[6:0]	SIZEy[1:0]	DADDRy[15:0]
...	...	...	...	...	...

Table 13-14. Next Destination Descriptor Pointer Value Determination(1), (2), (3), (4) (continued)

Memory Position	Memory Address Location (DP as a Product of Register Content and Arithmetic Calculation)	Descriptor[31:0]			
m	DADDR[23:0] = DADDR[23:0] + DPOyM1 if MATCH = 1 else DADDR[23:0] + DPOyM0 is MATCH = 0	DPOzM1[6:0]	DPOzM0[6:0]	SIZEz[1:0]	DADDRz[15:0]
...	...	...	...	...	...

Notes:

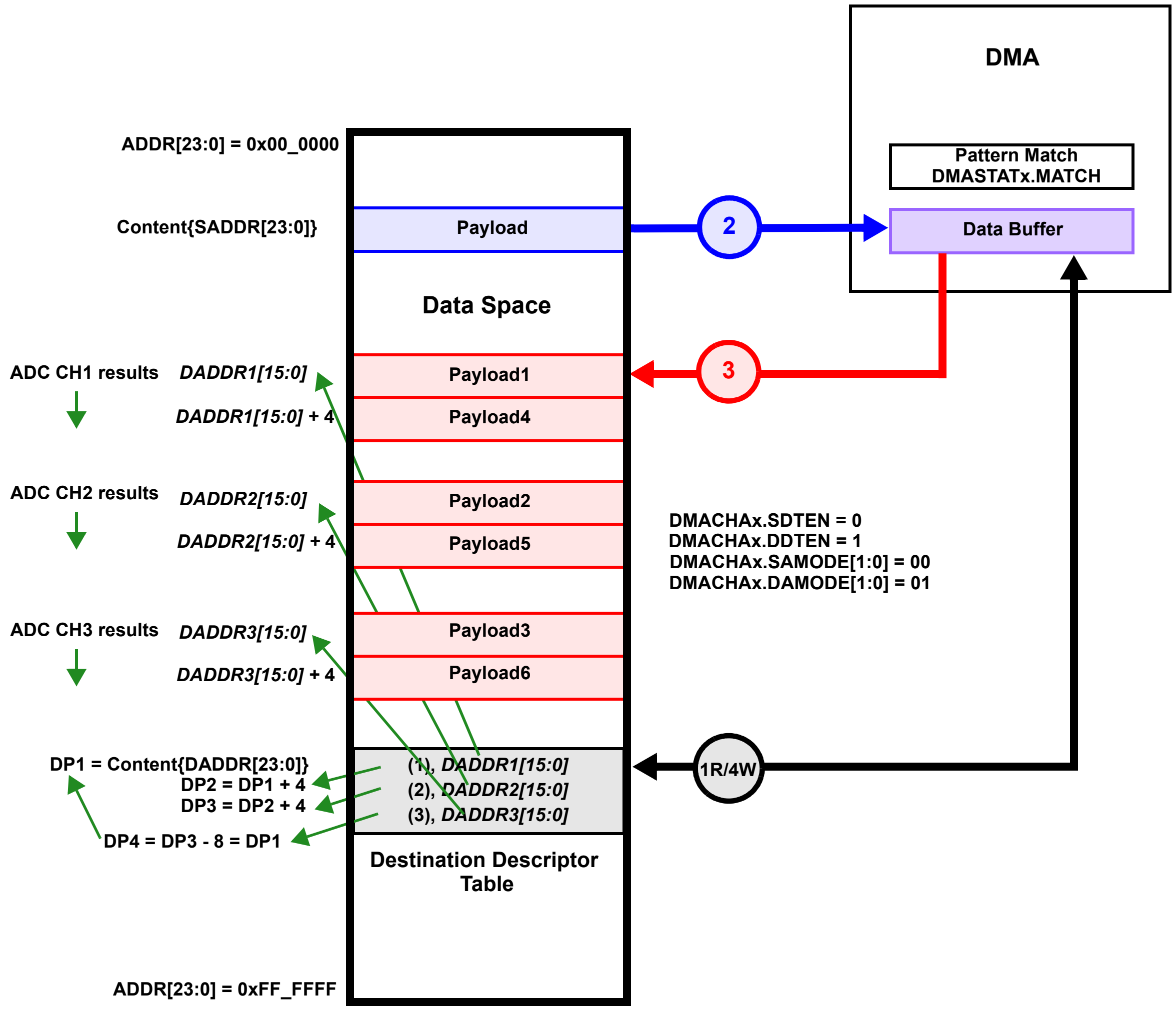
1. Descriptor fetching is subject to memory boundary restriction and address fault detection.
2. Descriptor fetching is a 32-bit word operation.
3. DADDR[23:0] indicates a register content.
4. DADDRn[15:0] indicates a payload pointer value.

13.5.6. Descriptor Write Back

The descriptor-based operation offers a descriptor write-back capability, which allows the Payload Pointer (PP) portion of the descriptor to be dynamically updated (i.e., incremented or decremented) during operation. The write back occurs after the transaction is completed, provided that the Source Address/Destination Address (SA/DA) mode is not zero. After incrementing the Payload Pointer, the DMA controller writes back the updated value to the descriptor, ensuring that the descriptor’s Payload Pointer reflects the current position in memory. With its jump capability using DPO, the descriptors can alternatively be fetched and updated to address a database structure with consecutive locations without having to explicitly pre-load all the descriptors corresponding to each address location. This feature is particularly suitable in any memory-constrained system because it enables the efficient use of system memory.

Figure 13-12 illustrates the use of descriptor write-back capability in the Gather/Scatter operation commonly utilized in multi-sampling ADC applications utilizing three ADC channels which output conversion results onto the same SFR register. By selecting increment or decrement for the address mode(s), SAMODE[1:0]/DAMODE[1:0], an extra memory cycle is generated to update the PP portion of the descriptor after the data transfer phase of the operation is completed. This allows the updated descriptor to be reused again at the next iteration.

Figure 13-12. Gather and Scatter Operation



Notes:

1. DPO1M1 = DPO1M0 = 1 for +4, *SIZE1[1:0]* = 10 for 32-bit transfer
2. DPO2M1 = DPO2M0 = 1 for +4, *SIZE2[1:0]* = 10 for 32-bit transfer
3. DPO3M1 = DPO3M0 = 3E for -8, *SIZE2[1:0]* = 10 for 32-bit transfer

13.5.7. Operational Sequence

A complete descriptor-based operation sequence can be summarized below. This includes both Source Descriptor Table (SDT) and Destination Descriptor Table (DDT) operations with write back enabled.

Table 13-15. Descriptor-Based Operational Sequence

Function	Details[(1)]
Fetch	
Read descriptor from location SADDR[23:0] (SDP) if SDTEN = `1`.	Put *SADDRn[15:0]* (SPP) into SADDR[15:0] register.
Read descriptor from location DADDR[23:0] (DDP) if DDTEN = `1`.	Put *DADDRn[15:0]* (DPP) into DADDR[15:0] register.
Payload Transfer	

Table 13-15. Descriptor-Based Operational Sequence (continued)

Function	Details[1]
Read payload from location SADDR[23:0] (SPP) per *SIZEn[1:0]*.	If SDTEN = 0, apply SAMODE[1:0] else if SDTEN = 1, restore SADDR[23:0] to original SDP.
Write payload from location DADDR[23:0] (DPP) per *SIZEn[1:0]*.	If DDTEN = 0, apply DAMODE[1:0] else if DDTEN = 1, restore DADDR[23:0] to original DDP.
Descriptor Write-Back[2]	
Write back *SADDRm[15:0]* (SPP) subject to SAMODE[1:0] if SDTEN = 1.	• Apply SAMODE[1:0] to *SADDRn[15:0]* (SPP). • Store updated value to location SADDR[23:0].
Write back *DADDRm[15:0]* (DPP) subject to SAMODE[1:0] if DDTEN = 1.	• Apply SAMODE[1:0] to *SADDRn[15:0]* (SPP). • Store updated value to location SADDR[23:0].
Next DP Calculation	
SADDR[23:0] = SADDR[23:0] + DPOxM1[6:0] if MATCH = 1 else SADDR[23:0] + DPOxM0[6:0] is MATCH = 0.	Only if SDTEN = 1.
DADDR[23:0] = DADDR[23:0] + DPOxM1[6:0] if MATCH = 1 else DADDR[23:0] + DPOxM0[6:0] is MATCH = 0.	Only if DDTEN = 1.

Notes:
1. Descriptor fetch is a 32-bit word read operation
2. Write back is a 16-bit word write operation

13.5.8. Application Examples

13.5.8.1. Basic Setup

To set up a DMA channel for any data transfer:

1. Enable the DMA Controller (ON = 1) and select an appropriate channel priority scheme by setting or clearing the PRIORITY bit.
2. Program DMAHIGH and DMALOW with the appropriate upper and lower address boundaries for data RAM operations.
3. Select the DMA channel to be used and disable its operation (CHEN = 0).
4. Program the appropriate source and destination addresses for the transaction into the channel’s DMAxSRC and DMAxDST registers. For PIA Mode Addressing, use the base address value.
5. Program the RELOAD registers for source address, destination address and count depending on the transfer modes used.
6. Program the DMAxCNT register for the number of triggers per transfer (One-Shot or Continuous modes) or the number of words (bytes) to be transferred (Repeated modes).
7. Configure the SIZE[1:0] bits to select the data size.
8. Program the TRMODE[1:0] bits to select the Data Transfer mode.
9. Program the SAMODE[1:0] and DAMODE[1:0] bits to select the Addressing mode.
10. Program the CHSEL[7:0] bits to select the trigger source
11. Enable the DMA channel by setting the CHEN bit.
12. Enable the trigger source interrupt.

13.5.8.2. Standard Operation (Data Transfer)

A basic example of using the DMA Controller is moving a constant stream of data from a serial communication channel, such as a UART, and buffering it in a location in data RAM until the CPU can process it. In this example, DMA channel 0 is used to service the UART for 16 data transfers in one iteration. It is configured as follows:

- DMA channel 0 is configured to use the UART's receive interrupt as a trigger.
- DMA channel 0 is programmed to use a single source address, to auto-increment the destination address with destination address reload enabled and to use Repeated One-Shot Data Transfer mode.
- DMA0SRC is programmed with the address of the UART's receive buffer; DMA0DST is programmed with an address in data RAM.
- To support 16 transfers in one iteration, DMA0CNT is programmed with 0016h.

In this configuration, the sequence of events is as follows:

1. When the UART triggers a receive interrupt, DMA0 transfers the data from the buffer to a data RAM location.
2. After the transfer, the destination address is incremented.
3. After 16 interrupts, DMA0CNT is decremented to 0000h. Because this is a Repeated mode transfer, the original values of DMA0CNT are reloaded and the cycle repeats. The value of DMA0DEST is also reloaded as RELOADD is enabled.

This process allows the CPU to perform different tasks other than buffering incoming serial data and processes the data when it has the time. Because the DMA is overwriting the same 16 locations in memory, it is assumed that the CPU will be able to retrieve the fresh data first.

13.5.8.3. Nested Operation (Wait State Generation)

DMA channels may be nested, using one channel to trigger another in performing a data transfer. When one of the microcontroller's general purpose timers is included, it becomes possible to generate a fixed delay between a service request and the data transfer. In this case, DMA0 and DMA1 are used to service a UART after a forced Wait state. DMA channels 0 and 1 are preconfigured as follows:

- DMA channel 0 is configured to use the UART's receive interrupt as a trigger.
- DMA0SRC is programmed with an address in data RAM; DMA0DST is programmed with the address of the T0CON register.
- DMA channel 1 is configured to use Timer0's interrupt as a trigger.
- DMA1SRC is programmed with the address of the UART's receive buffer; DMA0DST is programmed for the address of a destination in data RAM.

The sequence of events is as follows:

1. When the UART sends an interrupt, DMA0 transfers data into Timer0's Control register.
2. This causes Timer0 to count down once for a fixed interval (the Wait state), then generates an interrupt.
3. When Timer0 sends its interrupt, DMA1 is triggered and transfers data from the UART to data RAM.
4. Note that in this case, neither DMA channel was servicing the module from which it received its trigger.

13.5.8.4. Cascaded Operation (SPI Duplex Servicing)

Another method is to cascade two DMA channels together, allowing one to perform part of a function and then trigger a second channel to perform the other part. A good example is an SPI module operating in Client mode. Using two cascaded DMA channels allows automatic duplex operation, alternately receiving and sending data without the CPU's intervention. DMA0 (the read channel) and DMA1 (the write channel) are configured as follows:

- DMA0 is configured to use the SPI's transfer interrupt as the trigger for Repeated One-Shot transfers, with a fixed source address and a fixed destination address.

- DMA0SRC is programmed with the address of SPIBUF, while DMA0DST is programmed with a destination address in data RAM.
- DMA0CNT is programmed with 0001h (its default).
- DMA1 is configured to use the DMA0 interrupt as the trigger for Repeated One-Shot transfers and for fixed source and destination addresses.
- DMA1SRC is programmed with a data source address in data RAM, while DMA1DST is programmed with the address of SPIBUF.
- DMA1CNT is programmed with 0001h.

The sequence of events is as follows:

1. When the SPI receives data, it causes an SPI transfer interrupt.
2. This triggers DMA0 to transfer the data from the SPI buffer into RAM. At the completion of the transfer, the DMA0 interrupt is triggered.
3. The DMA0 interrupt triggers DMA1 to move data out of the data RAM location into SPIBUF to be transmitted. The process ends at this point.

For simplicity, this example moves one word of data in and out of the SPI. By changing the SAMODEx and DAMODEx bits for DMA0 and DMA1, respectively, and using different values for DMAxCNT, it is also possible to create multiword buffers for larger duplex transactions.

13.6. Interrupts

Each DMA channel has its own set of four interrupt flags, used to indicate a range of conditions during and following data transfers. Setting any of these flags with an interrupt event causes the device-level DMA Channel x Interrupt Flag (DMAxIF) to be set. With one exception, these flags are always enabled and not configurable. The DMAxIE bits, located in the IECx interrupt registers, will determine if a device-level interrupt is actually generated.

Since any of the DMA channel's individual event flags can trigger a device-level interrupt for the channel, the user must include a method within the ISR to determine which flag triggered the interrupt.

The four DMA channel interrupts are:

- DMA completion interrupt
- DMA halfway point interrupt
- Overrun interrupt
- Pattern match interrupt

Apart from these, the DMA trap is generated when there are bus error conditions, such as:

- Address Fault
- Bus write error
- Bus read error

In all of these conditions, the DMA will suspend its operation and the CHEN bit will be cleared.

13.6.1. DMA Channel Interrupts

13.6.1.1. DMA Completion Interrupt

The DONE bit (DMAxSTAT[5]) indicates the completion status of the last DMA operation. It is automatically set when DMAxCNT decrements to 0000h during a One-Shot or Continuous DMA transaction. When the DONE bit gets set, an interrupt is generated, indicating the DMA transfer completion.

By also examining the corresponding CHEN bit (DMAxCH[0]), it is possible to gain additional information on the status of the previous and current transactions. The possible interpretations are shown in Table 13-16.

Note that DONE remains cleared (= 0) when any Repeated Transfer modes are being used. This is because the address registers and transaction counters automatically reload, and the transaction automatically repeats when DMAxCNT decrements to 0000h. Repeated mode transfers must be terminated in software by clearing the CHEN bit.

The DONEEN bit (DMAxCH[3]) enables the DMA completion interrupt. When DONEEN is not set, the interrupt will not be generated.

Table 13-16. DMA Transaction Status

Bit Status		DMA Transaction Status
DONEIF	CHEN	
0	0	Previous transaction ended without completion.
0	1	Current transaction is not yet complete.
1	0	Previous transaction ended with completion.
1	1	Previous transaction ended with completion.

13.6.1.2. DMA Halfway Point Interrupt

The HALF interrupt flag (DMAxSTAT[4]) is an optional interrupt that indicates that the DMAxCNT register is at the halfway point between its original programmed value and 0000h. This can be used with the DONE interrupt to monitor the progress of the DMA transfer.

When enabled, HALF is set only when DMAxCNT reaches the halfway mark, but not thereafter. This results in a non-persistent interrupt. In Repeated modes, the DMA Controller attempts to set HALF every time DMAxCNT reaches the halfway point, whether or not the bit has been cleared. It is the user's responsibility to clear the bit after it has been set.

The HALFEN bit (DMAxCH[1]) enables the halfway point interrupt. If HALFEN is not set, then the interrupt will not be generated.

13.6.1.3. Overrun Interrupt

When a DMA channel receives a trigger while its CHREQ bit is already set (either by software or another hardware trigger), an Overrun condition occurs. This condition indicates that the channel is being requested before its current transaction is finished. This implies that the active channel may not be able to keep up with the demands from the peripheral module being serviced, which may result in data loss. An Overrun condition causes the OVERRUN flag (DMAxSTAT[3]) to be set.

Note that the OVERRUN flag being set does not cause the current DMA operation to terminate. Therefore, the channel for which OVERRUN is set does not need to be the active channel.

Setting the priority scheme correctly also helps to avoid overrun errors. For example, if one of the channels operates more frequently, a fixed priority scheme with that as the channel will help to reduce overrun interrupts.

13.6.1.4. Pattern Match Interrupt

When the MATCHEN bit is enabled, the contents of DMABUF[31:0] and DMAxPAT[31:0] are subjected to the corresponding DMAxMSK[31:0] bits' value and compared against each other . When a match is detected, the DMA Controller proceeds to invoke its channel interrupt output while setting the MATCH bit (DMAxSTAT[1]), accordingly. For more information, refer to Real-Time Pattern Matching.

13.6.2. DMA Bus Error Trap

13.6.2.1. Address Fault

DMA requires that all its Address Pointers be naturally aligned and within their designated ranges. When DMA tries to access an out of range address or a misaligned address, a DMA trap will be generated on a channel basis.

Address Faults are detected as follows:

- A DMA operation has crossed the data RAM address boundaries set by the DMAHIGH and DMALOW registers.
- For a 16-bit word operation (SIZE[1:0] = `01`), when the pointers are not 16-bit word-aligned.
- For a 32-bit word operation (SIZE[1:0] = `10`), when the pointers are not 32-bit word-aligned.

The user can determine the Fault condition by reading DMAxSTAT for the value of ADRERR[1:0]. These flag bits are set on any operation that attempts to read data from, or write data to, an address outside of the DMA boundaries or a misaligned address. An address Fault interrupt immediately terminates any DMA transaction in progress.

13.6.2.2. Bus Read Error

The DMA module responds to read and write errors on the bus. A bus read error Fault occurs when the data read by the DMA are invalid. This might occur, for example, under the following conditions:

- The read was not allowed because of the device security settings.
- The read was attempted at an unimplemented address.

When a bus read error occurs, the BRERR (DMAxSTAT[8]) bit is set so that user software can detect the read error. When a bus read error occurs, the DMA channel can optionally suspend operation based on the setting of the Read Error Trap Enable bit, RETEN (DMAxCH[16]). When the RETEN bit is cleared (default), the DMA channel will continue operation when a bus read error is encountered. The data obtained from the bus read will be transferred to the write destination by the DMA channel. When the RETEN bit is set, the DMA channel will suspend operation and the DMA trap will be asserted.

The RETEN control bit allows the user to make a trade-off between time-sensitive data streaming applications and applications where the data must be accurate. For example, an audio streaming application may be able to tolerate an occasional data error even if it produces audible artifacts. In contrast, a data error in a closed-loop control system could have more severe consequences, and it would be safer to suspend DMA transfers until the cause of the data error has been resolved.

13.6.2.3. Bus Write Error

A bus write error occurs when the data write by the DMA could not be completed. This might occur, for example, under the following conditions:

- The write was not allowed because of the device security settings.
- The write was attempted at an unimplemented address.

When a bus write error occurs, the BWERR (DMAxSTAT[9]) bit is set so that user software can detect the write error. When a bus write error is detected, the DMA halts its operation and a DMA trap is asserted.

13.7. Data Size

The DMA Controller can handle 8-bit, 16-bit and 32-bit transactions. Each DMA channel is individually configurable for the data size to be used with the SIZE[1:0] bits (DMAxCH[7:6]). These bits allow the user to specify whether one byte, one 16-bit word or one 32-bit word is transferred per one load or store transaction.

In Byte mode, where SIZE[1:0] is ‘`00`’, the counter (CNT[31:0]) represents the number of bytes remaining to be transferred. Similarly, the CNT[31:0] bits represent the number of 16-bit words and

32-bit words remaining to be transferred in 16-bit Word (SIZE[1:0] = 01) and 32-bit Word (SIZE[1:0]) modes, respectively.

In Byte mode, byte transfers are accommodated through bit 0 of the address. When bit 0 is '0', the lower byte is addressed, while the upper byte is addressed when bit 0 is '1'. For 16-bit word operation, the address pointers are 16-bit word-aligned. That is, bit 0 is always '0'. For 32-bit word operation, the address pointers are 32-bit word-aligned and bits[1:0] are always '00'. By default (SIZE[1:0] = 00), the channel is configured for byte-size transactions.

13.8. Power-Saving Modes

Though the DMA Controller can be thought of as an extension of the CPU, it is treated as a peripheral when it comes to power-saving operations. Like other peripherals, the DMA Controller also uses Peripheral Module Disable (PMD) bits to further tailor its operation in Low-Power states.

13.8.1. Idle Mode

As the system clock persists through Idle mode, the DMA Controller supports the operation in Idle mode. However, the DMA Controller provides an option to suspend operation upon Idle mode entry using the Stop in Idle Mode bit, DMASIDL (DMACON[13]). In this mode, there will not be any active DMA operations. The controller resumes any partially completed transactions on exiting from Idle mode.

13.8.2. Sleep Mode

When the device enters Sleep mode, all clock sources to the module are shut down and stay at logic '0'. Any transfers in progress are aborted. The controller will not resume any partially completed transactions on exiting from Sleep mode.

Register contents are not affected by the device entering or leaving Sleep mode. It is recommended that DMA transactions be allowed to finish before entering Sleep mode.

13.8.3. Peripheral Module Disable (PMD) Register

The Peripheral Module Disable (PMD) registers provide a method to disable DMA by stopping all clock sources supplied to it.

When DMA is disabled via PMD controlled bits, the DMA Controller is in a Minimum Power Consumption state. The module-level registers (DMACON, DMABUF, DMAHIGH and DMALOW) remain active. However, the control and status registers associated with any disabled channels will be disabled, so writes to those registers will have no effect and read values will be invalid.

13.8.4. Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the DMA Controller and all channels to be turned off, and any transfers in progress to be aborted. All buffer and address registers are initialized to 0 (except the CNT register that holds a value of 1 on Reset).

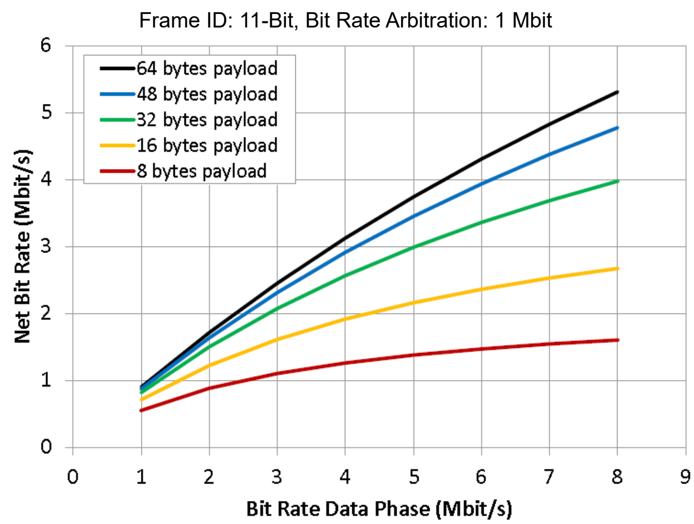
14. CAN Flexible Data-Rate (FD) Protocol Module

The CAN Flexible Data-Rate (FD) addresses the increasing demand for bandwidth on CAN buses. The two major enhancements over CAN 2.0B consist of:

- An increased data field of up to 64 data bytes (from a maximum eight data bytes for CAN 2.0B)
- An option to switch to faster bit rate after the arbitration field

Figure 14-1 shows the possible increase in net bit rate due to the higher Data Bit Rate (DBR) and increased data bytes per frame.

Figure 14-1. Net CAN FD Bit Rate



The CAN FD protocol is defined to allow CAN 2.0 messages and CAN FD messages to co-exist on the same bus. This does not imply that non-CAN FD controllers can be mixed with CAN FD controllers on the same bus. Non-CAN FD controllers will generate error frames while receiving a CAN FD message.

14.1. Device-Specific Information

Table 14-1. CAN Summary Table

CAN Module Instances	PPS Availability	Peripheral Bus Speed	Clock Source
1	All Instances	Standard (1:2 CPU Clock)	CLKGEN10

Table 14-2. CLKSEL Clock Selection bit

Value	Description
1	Standard (1:2 CPU Clock)
0	CLKGEN10

14.2. Features

The CAN FD module has the following features:

General

- Nominal (Arbitration) Bit Rate up to 1 Mbps
- Data Bit Rate up to 8 Mbps

- CAN FD Controller Modes:
  - Mixed CAN 2.0B and CAN FD mode
  - CAN 2.0B mode
- Conforms to ISO 11898-1:2015

Message FIFOs

- 7 FIFOs Configurable as Transmit or Receive FIFOs
- One Transmit Queue (TXQ)
- Transmit Event FIFO (TEF) with a 32-Bit Timestamp

Message Transmission

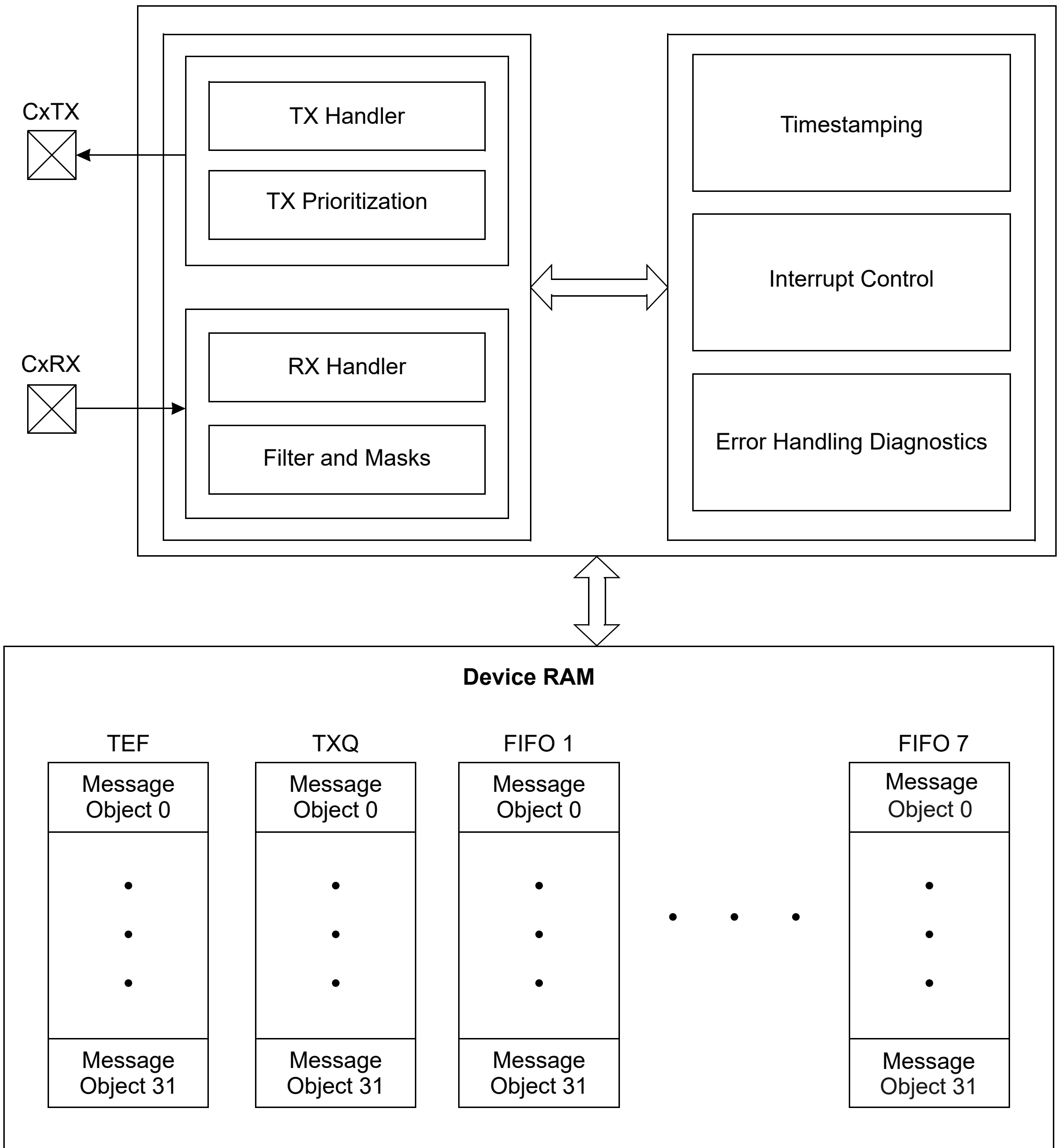
- Message Transmission Prioritization:
  - Based on priority bit field and/or
  - A message with the lowest ID is transmitted first using the TXQ
- Programmable Automatic Retransmission Attempts: Unlimited, Three Attempts or Disabled

Message Reception

- 16 Flexible Filter and Mask Objects
- Each Object can be Configured to Filter either:
  - The standard ID and first 18 data bits or
  - The extended ID
- 32-Bit Timestamp
- The CAN FD Bit Stream Processor (BSP) implements the Medium Access Control (MAC) of the CAN FD protocol described in ISO 11898-1:2015. It serializes and deserializes the bit stream, encodes and decodes CAN FD frames, manages medium access, Acknowledges frames, and detects and signals errors.
- The TX handler prioritizes the messages that are requested for transmission by the transmit FIFOs. It uses the RAM interface to fetch the transmit data from RAM and provides them to the BSP for transmission.
- The BSP provides received messages to the RX handler. The RX handler uses an acceptance filter to filter the messages that shall be stored in the receive FIFOs. It uses the RAM interface to store received data into RAM.
- Each FIFO can be configured either as a transmit or receive FIFO. The FIFO control keeps track of the FIFO head and tail and calculates the user address. In a TX FIFO, the user address points to the address in RAM where the data for the next transmit message are stored. In an RX FIFO, the user address points to the address in RAM where the data of the next receive message will be read. The user notifies the FIFO that a message is written to or read from RAM by incrementing the head/tail of the FIFO.
- The TXQ is a special transmit FIFO that transmits the messages based on the ID of the messages stored in the queue.
- The TEF stores the message IDs of the transmitted messages.
- A free-running Time Base Counter (TBC) is used to time-stamp received messages. Messages in the TEF can also be time-stamped.
- The CAN FD controller module generates interrupts when new messages are received or when messages are transmitted successfully.

Figure 14-2 shows the system block diagram.

Figure 14-2. System Block Diagram



14.3. CAN FD Message Frames

The ISO11898-1:2015 describes the different CAN message frames in detail. Figure 14-3 through Figure 14-8 explain and summarize the construction of the messages and fields.

There are four different CAN data/remote frames (see Figure 14-4):

- CAN Base Frame: Classic CAN 2.0 frame using Standard ID
- CAN FD Base Frame: CAN FD frame using Standard ID
- CAN Extended Frame: Classic CAN 2.0 frame using Extended ID
- CAN FD Extended Frame: CAN FD frame using Extended ID

There are no remote frames in CAN FD frames; therefore, the RTR bit is replaced with the RRS bit (see Figure 14-4). The RRS bit in the CAN FD base frame can be used to extend the SID to 12 bits. When enabled, it is referred to as SID11; it is the Least Significant bit (LSb) of SID[11:0].

Figure 14-5 specifies the control field of the different CAN messages. Before CAN FD was added to the ISO11898-1:2015, the FDF bit was a reserved bit. Now the FDF bit selects between Classic and CAN FD formats.

The BRS bit selects if the bit rate should be switched in the data phase of CAN FD frames. Figure 14-8 illustrates the error and overload frames. These special frames do not change.

Note: If an error is detected during the data phase of a CAN FD frame, the bit rate will be switched back to the Nominal Bit Rate (NBR). Error frames are always transmitted at the arbitration bit rate.

14.3.1. ISO vs. Non-ISO CRC

To support the system validation of non-ISO CRC ECUs, the CAN FD controller module supports both ISO CRC (according to ISO11898-1:2015) and non-ISO CRC (see Figure 14-6 and Figure 14-7). The CRC field is selectable using the ISOCRCEN bit (CxCON[5]). The ISO CRC field contains the stuff count. This count was not included in the original CAN FD specification; it was added to fix a minor issue in the error detection of the original specification.

CAN FD frames use two different lengths of CRC: 17-bit for up to 16 data bytes and 21-bit for 20 or more data bytes. Technically, there are a total of six different CAN data/remote frames in the CAN FD.

Figure 14-3. General Data Frame

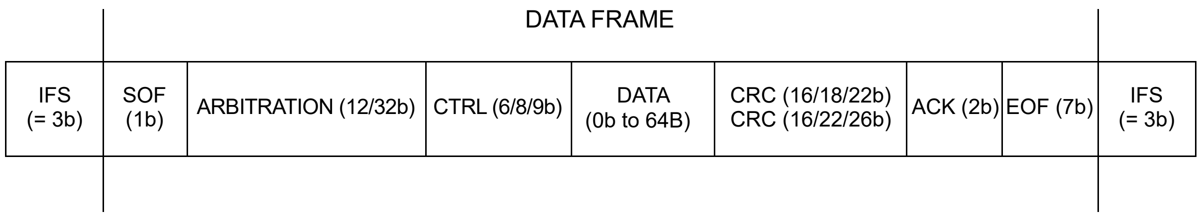


Figure 14-4. Arbitration Field

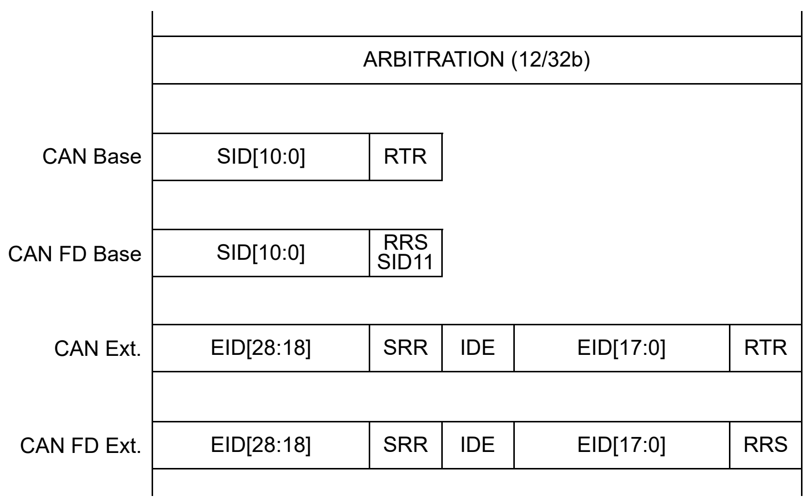


Figure 14-5. Control Field

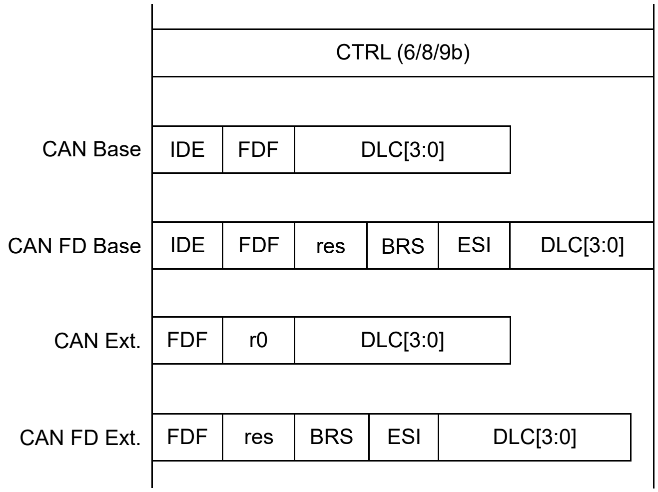


Figure 14-6. ISO CRC Field

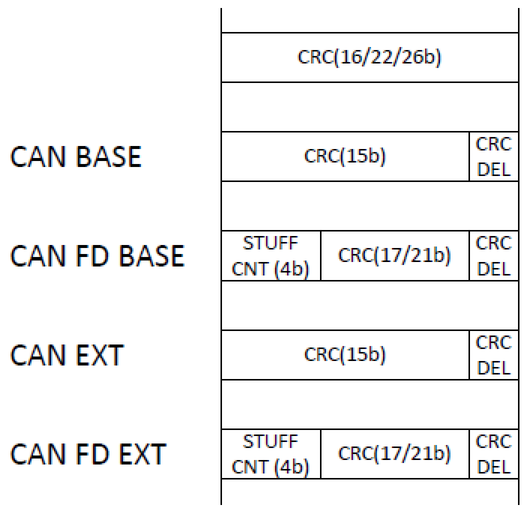


Figure 14-7. Non-ISO CRC Field

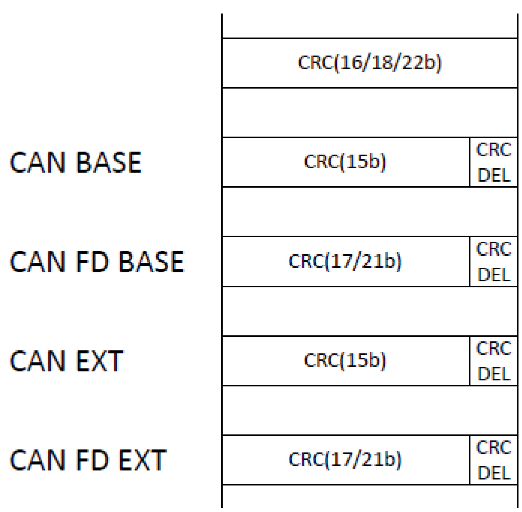
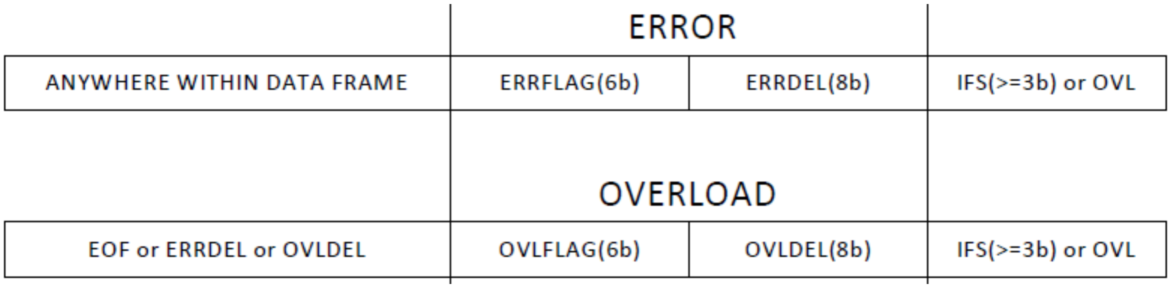


Figure 14-8. Error and Overload Frame



14.3.1.1. DLC Encoding

The Data Length Code (DLC) specifies the number of data bytes a message frame contains. Table 14-3 illustrates the encoding.

Table 14-3. DLC Encoding

Frame	DLC	Number of Data Bytes
CAN 2.0 and CAN FD	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
CAN 2.0	9-15	8
CAN FD	9	12
	10	16
	11	20
	12	24
	13	32
	14	48
	15	64

14.4. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2600	C1CON	31:24	TXBWS[3:0]				ABAT	REQOP[2:0]		
		23:16	OPMOD[2:0]			TXQEN	STEF	SERR2LOM	ESIGM	RTXAT
		15:8	ON		SIDL	BRSDIS	BUSY	WFT[1:0]		WAKFIL
		7:0	CLKSEL	PXEDIS	ISOCRCEN	DNCNT[4:0]				
0x2604	C1NBTCFG	31:24	BRP[7:0]							
		23:16	TSEG1[7:0]							
		15:8		TSEG2[6:0]						
		7:0		SJW[6:0]						
0x2608	C1DBTCFG	31:24	BRP[7:0]							
		23:16				TSEG1[4:0]				
		15:8					TSEG2[3:0]			
		7:0					SJW[3:0]			
0x260C	C1TDC	31:24							EDGFLTEN	SID11EN
		23:16							TDCMOD[1:0]	
		15:8		TDCO[6:0]						
		7:0			TDCV[5:0]					
0x2610	C1TBC	31:24	TBC[31:24]							
		23:16	TBC[23:16]							
		15:8	TBC[15:8]							
		7:0	TBC[7:0]							
0x2614	C1TSCON	31:24								
		23:16						TSRES	TSEOF	TBCEN
		15:8							TBCPRE[9:8]	
		7:0	TBCPRE[7:0]							
0x2618	C1VEC	31:24		RXCODE[6:0]						
		23:16		TXCODE[6:0]						
		15:8				FILHIT[4:0]				
		7:0		ICODE[6:0]						
0x261C	C1INT	31:24	IVMIE	WAKIE	CERRIE	SERRIE	RXOVIE	TXATIE		
		23:16				TEFIE	MODIE	TBCIE	RXIE	TXIE
		15:8	IVMIF	WAKIF	CERRIF	SERRIF	RXOVIF	TXATIF		
		7:0				TEFIF	MODIF	TBCIF	RXIF	TXIF
0x2620	C1RXIF	31:24	RFIF[30:23]							
		23:16	RFIF[22:15]							
		15:8	RFIF[14:7]							
		7:0	RFIF[6:0]							
0x2624	C1TXIF	31:24	TFIF[31:24]							
		23:16	TFIF[23:16]							
		15:8	TFIF[15:8]							
		7:0	TFIF[7:0]							
0x2628	C1RXOVIF	31:24	RFOVIF[30:23]							
		23:16	RFOVIF[22:15]							
		15:8	RFOVIF[14:7]							
		7:0	RFOVIF[6:0]							
0x262C	C1TXATIF	31:24	TFATIF[31:24]							
		23:16	TFATIF[23:16]							
		15:8	TFATIF[15:8]							
		7:0	TFATIF[7:0]							
0x2630	C1TXREQ	31:24	TXREQ[30:23]							
		23:16	TXREQ[22:15]							
		15:8	TXREQ[14:7]							
		7:0	TXREQ[6:0]							TXREQ
0x2634	C1TREC	31:24								
		23:16			TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
		15:8	TERRCNT[7:0]							
		7:0	RERRCNT[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2638	C1BDIAG0	31:24	DTERRCNT[7:0]							
		23:16	DRERRCNT[7:0]							
		15:8	NTERRCNT[7:0]							
		7:0	NRERRCNT[7:0]							
0x263C	C1BDIAG1	31:24	DLCMM	ESI	DCRCERR	DSTUFERR	DFORMERR		DBIT1ERR	DBIT0ERR
		23:16	TXBOERR		NCRCERR	NSTUFERR	NFORMERR	NACKERR	NBIT1ERR	NBIT0ERR
		15:8	EFMSGCNT[15:8]							
		7:0	EFMSGCNT[7:0]							
0x2640	C1TEFCON	31:24				FSIZE[4:0]				
		23:16								
		15:8						FRESET		UINC
		7:0			TEFTSEN		TEFOVIE	TEFFIE	TEFHIE	TEFNEIE
0x2644	C1TEFSTA	31:24								
		23:16								
		15:8								
		7:0					TEFOVIF	TEFFIF	TEFHIF	TEFNEIF
0x2648	C1TEFUA	31:24	TEFUA[31:24]							
		23:16	TEFUA[23:16]							
		15:8	TEFUA[15:8]							
		7:0	TEFUA[7:0]							
0x264C	C1FIFOBA	31:24	FIFOBA[31:24]							
		23:16	FIFOBA[23:16]							
		15:8	FIFOBA[15:8]							
		7:0	FIFOBA[7:0]							
0x2650	C1TXQCON	31:24	PLSIZE[2:0]			FSIZE[4:0]				
		23:16		TXAT[1:0]		TXPRI[4:0]				
		15:8						FRESET	TXREQ	UINC
		7:0	TXEN			TXATIE		TXQEIE		TXQNIE
0x2654	C1TXQSTA	31:24								
		23:16								
		15:8				TXQCI[4:0]				
		7:0	TXABT	TXLARB	TXERR	TXATIF		TXQEIF		TXQNIF
0x2658	C1TXQUA	31:24	TXQUA[31:24]							
		23:16	TXQUA[23:16]							
		15:8	TXQUA[15:8]							
		7:0	TXQUA[7:0]							
0x265C	C1FIFOCON1	31:24	PLSIZE[2:0]			FSIZE[4:0]				
		23:16		TXAT[1:0]		TXPRI[4:0]				
		15:8						FRESET	TXREQ	UINC
		7:0	TXEN	RTREN	RXTSEN	TXATIE	RXOVIE	TFERFFIE	TFHRFHIE	TFNRFNIE
0x2660	C1FIFOSTA1	31:24								
		23:16								
		15:8				FIFOCI[4:0]				
		7:0	TXABT	TXLARB	TXERR	TXATIF	RXOVIF	TFERFFIF	TFHRFHIF	TFNRFNIF
0x2664	C1FIFOUA1	31:24	FIFOUA[31:24]							
		23:16	FIFOUA[23:16]							
		15:8	FIFOUA[15:8]							
		7:0	FIFOUA[7:0]							
0x2668	C1FIFOCON2	31:24	PLSIZE[2:0]			FSIZE[4:0]				
		23:16		TXAT[1:0]		TXPRI[4:0]				
		15:8						FRESET	TXREQ	UINC
		7:0	TXEN	RTREN	RXTSEN	TXATIE	RXOVIE	TFERFFIE	TFHRFHIE	TFNRFNIE
0x266C	C1FIFOSTA2	31:24								
		23:16								
		15:8				FIFOCI[4:0]				
		7:0	TXABT	TXLARB	TXERR	TXATIF	RXOVIF	TFERFFIF	TFHRFHIF	TFNRFNIF
0x2670	C1FIFOUA2	31:24	FIFOUA[31:24]							
		23:16	FIFOUA[23:16]							
		15:8	FIFOUA[15:8]							
		7:0	FIFOUA[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2674	C1FIFOCON3	31:24	PLSIZE[2:0]			FSIZE[4:0]				
		23:16		TXAT[1:0]		TXPRI[4:0]				
		15:8						FRESET	TXREQ	UINC
		7:0	TXEN	RTREN	RXTSEN	TXATIE	RXOVIE	TFERFFIE	TFHRFHIE	TFNRFNIE
0x2678	C1FIFOSTA3	31:24								
		23:16								
		15:8				FIFOCI[4:0]				
		7:0	TXABT	TXLARB	TXERR	TXATIF	RXOVIF	TFERFFIF	TFHRFHIF	TFNRFNIF
0x267C	C1FIFOUA3	31:24	FIFOUA[31:24]							
		23:16	FIFOUA[23:16]							
		15:8	FIFOUA[15:8]							
		7:0	FIFOUA[7:0]							
0x2680	C1FIFOCON4	31:24	PLSIZE[2:0]			FSIZE[4:0]				
		23:16		TXAT[1:0]		TXPRI[4:0]				
		15:8						FRESET	TXREQ	UINC
		7:0	TXEN	RTREN	RXTSEN	TXATIE	RXOVIE	TFERFFIE	TFHRFHIE	TFNRFNIE
0x2684	C1FIFOSTA4	31:24								
		23:16								
		15:8				FIFOCI[4:0]				
		7:0	TXABT	TXLARB	TXERR	TXATIF	RXOVIF	TFERFFIF	TFHRFHIF	TFNRFNIF
0x2688	C1FIFOUA4	31:24	FIFOUA[31:24]							
		23:16	FIFOUA[23:16]							
		15:8	FIFOUA[15:8]							
		7:0	FIFOUA[7:0]							
0x268C	C1FIFOCON5	31:24	PLSIZE[2:0]			FSIZE[4:0]				
		23:16		TXAT[1:0]		TXPRI[4:0]				
		15:8						FRESET	TXREQ	UINC
		7:0	TXEN	RTREN	RXTSEN	TXATIE	RXOVIE	TFERFFIE	TFHRFHIE	TFNRFNIE
0x2690	C1FIFOSTA5	31:24								
		23:16								
		15:8				FIFOCI[4:0]				
		7:0	TXABT	TXLARB	TXERR	TXATIF	RXOVIF	TFERFFIF	TFHRFHIF	TFNRFNIF
0x2694	C1FIFOUA5	31:24	FIFOUA[31:24]							
		23:16	FIFOUA[23:16]							
		15:8	FIFOUA[15:8]							
		7:0	FIFOUA[7:0]							
0x2698	C1FIFOCON6	31:24	PLSIZE[2:0]			FSIZE[4:0]				
		23:16		TXAT[1:0]		TXPRI[4:0]				
		15:8						FRESET	TXREQ	UINC
		7:0	TXEN	RTREN	RXTSEN	TXATIE	RXOVIE	TFERFFIE	TFHRFHIE	TFNRFNIE
0x269C	C1FIFOSTA6	31:24								
		23:16								
		15:8				FIFOCI[4:0]				
		7:0	TXABT	TXLARB	TXERR	TXATIF	RXOVIF	TFERFFIF	TFHRFHIF	TFNRFNIF
0x26A0	C1FIFOUA6	31:24	FIFOUA[31:24]							
		23:16	FIFOUA[23:16]							
		15:8	FIFOUA[15:8]							
		7:0	FIFOUA[7:0]							
0x26A4	C1FIFOCON7	31:24	PLSIZE[2:0]			FSIZE[4:0]				
		23:16		TXAT[1:0]		TXPRI[4:0]				
		15:8						FRESET	TXREQ	UINC
		7:0	TXEN	RTREN	RXTSEN	TXATIE	RXOVIE	TFERFFIE	TFHRFHIE	TFNRFNIE
0x26A8	C1FIFOSTA7	31:24								
		23:16								
		15:8				FIFOCI[4:0]				
		7:0	TXABT	TXLARB	TXERR	TXATIF	RXOVIF	TFERFFIF	TFHRFHIF	TFNRFNIF
0x26AC	C1FIFOUA7	31:24	FIFOUA[31:24]							
		23:16	FIFOUA[23:16]							
		15:8	FIFOUA[15:8]							
		7:0	FIFOUA[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x26B0	C1FLTCON0	31:24	FLTEN3			F3BP[4:0]				
		23:16	FLTEN2			F2BP[4:0]				
		15:8	FLTEN1			F1BP[4:0]				
		7:0	FLTEN0			F0BP[4:0]				
0x26B4	C1FLTCON1	31:24	FLTEN7			F7BP[4:0]				
		23:16	FLTEN6			F6BP[4:0]				
		15:8	FLTEN5			F5BP[4:0]				
		7:0	FLTEN4			F4BP[4:0]				
0x26B8	C1FLTCON2	31:24	FLTEN11			F11BP[4:0]				
		23:16	FLTEN10			F10BP[4:0]				
		15:8	FLTEN9			F9BP[4:0]				
		7:0	FLTEN8			F8BP[4:0]				
0x26BC	C1FLTCON3	31:24	FLTEN15			F15BP[4:0]				
		23:16	FLTEN14			F14BP[4:0]				
		15:8	FLTEN13			F13BP[4:0]				
		7:0	FLTEN12			F12BP[4:0]				
0x26C0	C1FLTOBJ0	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x26C4	C1MASK0	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x26C8	C1FLTOBJ1	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x26CC	C1MASK1	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x26D0	C1FLTOBJ2	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x26D4	C1MASK2	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x26D8	C1FLTOBJ3	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x26DC	C1MASK3	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x26E0	C1FLTOBJ4	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x26E4	C1MASK4	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x26E8	C1FLTOBJ5	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x26EC	C1MASK5	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x26F0	C1FLTOBJ6	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x26F4	C1MASK6	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x26F8	C1FLTOBJ7	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x26FC	C1MASK7	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x2700	C1FLTOBJ8	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x2704	C1MASK8	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x2708	C1FLTOBJ9	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x270C	C1MASK9	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x2710	C1FLTOBJ10	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x2714	C1MASK10	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x2718	C1FLTOBJ11	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x271C	C1MASK11	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x2720	C1FLTOBJ12	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x2724	C1MASK12	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2728	C1FLTOBJ13	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x272C	C1MASK13	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x2730	C1FLTOBJ14	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x2734	C1MASK14	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							
0x2738	C1FLTOBJ15	31:24		EXIDE	SID11	EID[17:13]				
		23:16	EID[12:5]							
		15:8	EID[4:0]					SID[10:8]		
		7:0	SID[7:0]							
0x273C	C1MASK15	31:24		MIDE	MSID11	MEID[17:13]				
		23:16	MEID[12:5]							
		15:8	MEID[4:0]					MSID[10:8]		
		7:0	MSID[7:0]							

14.4.1. CAN FD 1 Control Register

Name: C1CON
Offset: 0x2600

Note:
1. These bits can only be modified in Configuration mode (OPMOD[2:0] = `100`).

Bit	31	30	29	28	27	26	25	24
	TXBWS[3:0]				ABAT	REQOP[2:0]		
Access	R/W	R/W	R/W	R/W	S/HC	R/W	R/W	R/W
Reset	0	0	0	0	0	1	0	0

Bit	23	22	21	20	19	18	17	16
	OPMOD[2:0]			TXQEN	STEF	SERR2LOM	ESIGM	RTXAT
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	1	1	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	BRSDIS	BUSY	WFT[1:0]		WAKFIL
Access	R/W		R/W	R/W	R	R/W	R/W	R/W
Reset	0		0	0	0	1	1	1

Bit	7	6	5	4	3	2	1	0
	CLKSEL	PXEDIS	ISOCRCEN	DNCNT[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	1	0	0	0	0	0

Bits 31:28 – TXBWS[3:0] Transmit Bandwidth Sharing bits
Delay between two consecutive transmissions (in arbitration bit times).

Value	Description
`1111-1100`	4096
`1011`	2048
`1010`	1024
`1001`	512
`1000`	256
`0111`	128
`0110`	64
`0101`	32
`0100`	16
`0011`	8
`0010`	4
`0001`	2
`0000`	No delay

Bit 27 – ABAT Abort All Pending Transmissions bit

Value	Description
`1`	Signals all transmit buffers to abort transmission.
`0`	Module will clear this bit when all transmissions are aborted.

Bits 26:24 – REQOP[2:0] Request Operation Mode bits

Value	Description
111	Sets Restricted Operation mode.
110	Sets Normal CAN 2.0 mode; error frames on CAN FD frames.
101	Sets External Loopback mode.
100	Sets Configuration mode.
011	Sets Listen Only mode.
010	Sets Internal Loopback mode.
001	Sets Disable mode.
000	Sets Normal CAN FD mode; supports mixing of full CAN FD and Classic CAN 2.0 frames.

Bits 23:21 – OPMOD[2:0] Operation Mode Status bits

Value	Description
111	Module is in Restricted Operation mode.
110	Module is in Normal CAN 2.0 mode; error frames on CAN FD frames.
101	Module is in External Loopback mode.
100	Module is in Configuration mode.
011	Module is in Listen Only mode.
010	Module is in Internal Loopback mode.
001	Module is in Disable mode.
000	Module is in Normal CAN FD mode; supports mixing of full CAN FD and Classic CAN 2.0 frames.

Bit 20 – TXQEN Enable Transmit Queue(1) bit

Value	Description
1	Enables TXQ and reserves space in RAM.
0	Does not reserve space in RAM for TXQ.

Bit 19 – STEF Store in Transmit Event FIFO(1) bit

Value	Description
1	Saves transmitted messages in TEF.
0	Does not save transmitted messages in TEF.

Bit 18 – SERR2LOM Transition to Listen Only Mode on System Error(1) bit

Value	Description
1	Transitions to Listen-Only mode on System Error.
0	Transitions to Restricted Operation mode on System Error.

Bit 17 – ESIGM Transmit ESI in Gateway Mode(1) bit

Value	Description
1	ESI is transmitted as recessive when the ESI of the message is high or the CAN controller is error passive.
0	ESI reflects the error status of the CAN controller.

Bit 16 – RTXAT Restrict Retransmission Attempts(1) bit

Value	Description
1	Restricted retransmission attempts, uses TXAT[1:0].
0	Unlimited number of retransmission attempts; TXAT[1:0] bits will be ignored.

Bit 15 – ON CAN Enable bit

Value	Description
1	CAN module is enabled.
0	CAN module is disabled.

Bit 13 – SIDL CAN Stop in Idle Mode bit

Value	Description
1	Stops module operation in Idle mode.
0	Does not stop module operation in Idle mode.

Bit 12 – BRSDIS Bit Rate Switching (BRS) Disable bit

Value	Description
1	Bit rate switching is disabled, regardless of BRS in the transmit message object.
0	Bit rate switching depends on the BRS of the transmit message object.

Bit 11 – BUSY CAN Module is Busy bit

Value	Description
1	The CAN module is active.
0	The CAN module is inactive.

Bits 10:9 – WFT[1:0] Selectable Wake-up Filter Time bits

Value	Description
11	T11 Filter
10	T10 Filter
01	T01 Filter
00	T00 Filter

Bit 8 – WAKFIL Enable CAN Bus Line Wake-up Filter(1) bit

Value	Description
1	Uses CAN bus line filter for wake-up.
0	CAN bus line filter is not used for wake-up.

Bit 7 – CLKSEL CAN Module Clock Source Select(1) bit

Value	Description
1	Reserved
0	CAN module clock is derived from Clock Generator 10.

Bit 6 – PXEDIS Protocol Exception Event Detection Disabled(1) bit

A recessive “reserved bit” following a recessive FDF bit is called a “Protocol Exception.”

Value	Description
1	Protocol exception is treated as a form error.
0	If a protocol exception is detected, CAN will enter the Bus Integrating state.

Bit 5 – ISOCRCEN Enable ISO CRC in CAN FD Frames(1) bit

Value	Description
1	Includes the stuff bit count in the CRC field and uses a nonzero CRC initialization vector.
0	Does not include the stuff bit count in the CRC field and uses CRC initialization vector with all zeroes.

Bits 4:0 – DNCNT[4:0] DeviceNet™ Filter Bit Number bits

Value	Description
11111-10011	Invalid selection (compares up to 18 bits of data with EIDx)
10010	Compares up to Data Byte 2, bit 6 with EID17.
10001	Compares up to Data byte 2, bit 7 with EID16.
...	...

Value	Description
00010	Compares up to Data byte 0 bit 6 with EID1 .
00001	Compares up to Data byte 0 bit 7 with EID0.
00000	Does not compare data bytes.

14.4.2. CAN Nominal Bit Time Configuration Register

Name: C1NBTCFG
Offset: 0x2604

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   – CxNBTCFGT: Accesses the top byte NBTCFG[31:24]
   – CxNBTCFGU: Accesses the upper byte NBTCFG[23:16]
   – CxNBTCFG: Accesses the byte NBTCFG[31:0]
2. This register can only be modified in Configuration mode (OPMOD[2:0] = `100`).

Bit	31	30	29	28	27	26	25	24
	BRP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TSEG1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	1	1	1	1	0

Bit	15	14	13	12	11	10	9	8
		TSEG2[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	1	1	1	1

Bit	7	6	5	4	3	2	1	0
		SJW[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	1	1	1	1

Bits 31:24 – BRP[7:0] Nominal Baud Rate Prescaler bits

Value	Description
`11111111`	$T_Q = 256 \times T_{CY}$
`00000000`	$T_Q = 1 \times T_{CY}$

Bits 23:16 – TSEG1[7:0] Nominal Time Segment 1 (Propagation Segment+Phase Segment 1) bits

Value	Description
`11111111`	Length is 256 x T_Q
. . .	
`00000000`	Length is 1 x T_Q

Bits 14:8 – TSEG2[6:0] Nominal Time Segment 2 (Phase Segment 2) bits

Value	Description
`1111111`	Length is 128 x T_Q
. . .	
`0000000`	Length is 1 x T_Q

Bits 6:0 – SJW[6:0] Nominal Synchronization Jump Width bits

Value	Description
1111111	Length is 128 x T_Q
. . .	
0000000	Length is 1 x T_Q

14.4.3. CAN Data Bit Time Configuration Register

Name: C1DBTCFG
Offset: 0x2608

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxDBTCFGT: Accesses the top byte DBTCFG[31:24]
   - CxDBTCFGU: Accesses the upper byte DBTCFG[23:16]
   - CxDBTCFG: Accesses the byte DBTCFG[31:0]
2. This register can only be modified in Configuration mode (OPMOD[2:0] = `100`).

Bit	31	30	29	28	27	26	25	24
	BRP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				TSEG1[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	1	1	1	0

Bit	15	14	13	12	11	10	9	8
					TSEG2[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	1	1

Bit	7	6	5	4	3	2	1	0
					SJW[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	1	1

Bits 31:24 – BRP[7:0] Data Baud Rate Prescaler bits

Value	Description
`11111111`	$T_Q=T_{CY}/256$
`00000000`	$T_Q=T_{CY}/1$

Bits 20:16 – TSEG1[4:0] Data Time Segment 1 bits (Propagation Segment+Phase Segment 1)

Value	Description
`11111`	Length is 32 x T_Q
`00000`	Length is 1 x T_Q

Bits 11:8 – TSEG2[3:0] Data Time Segment 2 bits (Phase Segment 2)

Value	Description
`1111`	Length is 16 x T_Q
`0000`	Length is 1 x T_Q

Bits 3:0 – SJW[3:0] Data Synchronization Jump Width bits

Value	Description
`1111`	Length is 16 x T_Q

Value	Description
0000	Length is 1 x T_Q

14.4.4. CAN Transmitter Delay Compensation Register

Name: C1TDC
Offset: 0x260C

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTDCT: Accesses the top byte TDC[31:24]
   - CxTDCU: Accesses the upper byte TDC[23:16]
   - CxTDC: Accesses the byte TDC[31:0]
2. This register can only be modified in Configuration mode (OPMOD[2:0] = 100).

Bit	31	30	29	28	27	26	25	24
							EDGFLTEN	SID11EN
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
							TDCMOD[1:0]	
Access							R/W	R/W
Reset							1	0

Bit	15	14	13	12	11	10	9	8
		TDCO[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	1	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			TDCV[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 25 – EDGFLTEN Enable Edge Filtering During Bus Integration State bit

Value	Description
1	Edge filtering is enabled according to ISO 11898-1:2015.
0	Edge filtering is disabled.

Bit 24 – SID11EN Enable 12-Bit SID in CAN FD Base Format Messages bit

Value	Description
1	RRS is used as SID11 in CAN FD base format messages: SID[11:0]={SID[10:0],SID11}.
0	Does not use RRS; SID[10:0].

Bits 17:16 – TDCMOD[1:0] Transmitter Delay Compensation Mode (Secondary Sample Point (SSP)) bits

Value	Description
11-10	Auto: Measures delay and adds TSEG1[4:0] (CxDBTCFG[19:16]; adds TDCO[6:0].
01	Manual: Does not measure, uses TDCV[5:0] +TDCO[6:0] .
00	Disables

Bits 14:8 – TDCO[6:0] Transmitter Delay Compensation Offset (Secondary Sample Point (SSP)) bits
Value is two's complement, offset can be positive, zero or negative.

Value	Description
0111111	63 x T_{CY}
0000000	0 x T_{CY}
11111111	-64 x T_{CY}

Bits 5:0 – TDCV[5:0] Transmitter Delay Compensation Value bits (Secondary Sample Point (SSP)) bits

Value	Description
111111	63 x T_{CY}
000000	0 x T_{CY}

14.4.5. CAN Time Base Counter Register

Name: C1TBC
Offset: 0x2610

Notes:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTBCT: Accesses the top byte TBC[31:24]
   - CxTBCU: Accesses the upper byte TBC[23:16]
   - CxTBC: Accesses the byte TBC[31:0]
2. The Time Base Counter (TBC will be stopped and reset when TBCEN = 0 to save power).
3. The TBC prescaler count will be reset on any write to CxTBC (TBCPREx will be unaffected).

Bit	31	30	29	28	27	26	25	24
	TBC[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TBC[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TBC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TBC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TBC[31:0] CAN Time Base Counter bits
This is a free-running timer that increments every TBCPRE[9:0] clock when TBCEN is set.

14.4.6. CAN Timestamp Control Register

Name: C1TSCON
Offset: 0x2614

Note:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTSCONT: Accesses the top byte TSCON[31:24]
   - CxTSCONU: Accesses the upper byte TSCON[23:16]
   - CxTSCON: Accesses the byte TSCON[31:0]

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						TSRES	TSEOF	TBCEN
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
							TBCPRE[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	TBCPRE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 18 – TSRES Timestamp Reset (CAN FD frames only) bit

Value	Description
1	Timestamp resets at sample point of the bit following the FDF bit.
0	Timestamp resets at sample point of Start-of-Frame (SOF).

Bit 17 – TSEOF Timestamp End-of-Frame (EOF) bit

Value	Description
1	Timestamp when frame is taken valid (11898-1 10.7): • RX no error until last, but one bit of EOF • TX no error until the end of EOF
0	Timestamp at “beginning” of frame: • Classical Frame: At sample point of SOF • FD Frame: see TSRES bit

Bit 16 – TBCEN Time Base Counter (TBC) Enable bit

Value	Description
1	Enables TBC.
0	Stops and resets TBC.

Bits 9:0 – TBCPRE[9:0] CAN Time Base Counter Prescaler bits

Value	Description
1111111111	TBC increments every 1024 clocks.
0000000000	TBC increments every 1 clock.

14.4.7. CAN Interrupt Code Register

Name: C1VEC
Offset: 0x2618

Note:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxVECT: Accesses the top byte VEC[31:24]
   - CxVECU: Accesses the upper byte VEC[23:16]
   - CxVECH: Accesses the byte VEC[31:0]

Bit	31	30	29	28	27	26	25	24
		RXCODE[6:0]						
Access		R	R	R	R	R	R	R
Reset		1	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		TXCODE[6:0]						
Access		R	R	R	R	R	R	R
Reset		1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				FILHIT[4:0]				
Access				R	R	R	R	R
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		ICODE[6:0]						
Access		R	R	R	R	R	R	R
Reset		1	0	0	0	0	0	0

Bits 30:24 – RXCODE[6:0] Receive Interrupt Flag Code bits

Value	Description
1111111-1000001	Reserved
1000000	No interrupt
0111111-0000100	Reserved
0000011	FIFO 3 interrupt (RFIF[3] is set)
0000010	FIFO 2 interrupt (RFIF[2] is set)
0000001	FIFO 1 interrupt (RFIF[1] is set)
0000000	Reserved; FIFO 0 cannot receive

Bits 22:16 – TXCODE[6:0] Transmit Interrupt Flag Code bits

Value	Description
1111111-1000001	Reserved
1000000	No interrupt
0111111-0000100	Reserved
0000011	FIFO 3 interrupt (TFIF[3] is set)
0000010	FIFO 2 interrupt (TFIF[2] is set)
0000001	FIFO 1 interrupt (TFIF[1] is set)

Value	Description
0000000	FIFO 0 interrupt (TFIF[0] is set)

Bits 12:8 – FILHIT[4:0] Filter Hit Number bits

Value	Description
11111	Filter 31
11110	Filter 30
...	
00001	Filter 1
00000	Filter 0

Bits 6:0 – ICODE[6:0] Interrupt Flag Code bits

Value	Description
1111111-1001011	Reserved
1001010	Transmit attempt interrupt (any bit in CxTXATIF is set)
1001001	Transmit event FIFO interrupt (any bit in CxTEFSTA is set)
1001000	Invalid message occurred (IVMIF/IE)
1000111	CAN module mode change occurred (MODIF/IE)
1000110	CAN timer overflow (TBCIF/IE)
1000101	RX/TX MAB overflow/underflow (RX: Message received before previous message was saved to memory; TX: Cannot feed TX MAB fast enough to transmit consistent data.) (SERRIF/IE)
1000100	Address error interrupt (illegal FIFO address presented to system) (SERRIF/IE)
1000011	Receive FIFO overflow interrupt (any bit in CxRXOVIF is set)
1000010	Wake-up interrupt (WAKIF/WAKIE)
1000001	Error interrupt (CERRIF/IE)
1000000	No interrupt
0111111-0100000	Reserved
0011111	FIFO 31 Interrupt (TFIF[31] or RFIF[31] set)
...	
0000001	FIFO 1 Interrupt (TFIF1 or RFIF1 is set)
0000000	FIFO 0 Interrupt (TFIF0 is set)

14.4.8. CAN Interrupt Register

Name: C1INT
Offset: 0x261C

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxINTT: Accesses the top byte INT[31:24]
   - CxINTU: Accesses the upper byte INT[23:16]
   - CxINT: Accesses the byte INT[31:0]
2. Flag is set by hardware and cleared by application.

Bit	31	30	29	28	27	26	25	24
	IVMIE	WAKIE	CERRIE	SERRIE	RXOVIE	TXATIE		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	23	22	21	20	19	18	17	16
				TEFIE	MODIE	TBCIE	RXIE	TXIE
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IVMIF	WAKIF	CERRIF	SERRIF	RXOVIF	TXATIF		
Access	HS/C	HS/C	HS/C	HS/C	R	R		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
				TEFIF	MODIF	TBCIF	RXIF	TXIF
Access				R	HS/C	HS/C	R	R
Reset				0	0	0	0	0

Bit 31 – IVMIE Invalid Message Interrupt Enable bit

Value	Description
1	Invalid message interrupt is enabled.
0	Invalid message interrupt is disabled.

Bit 30 – WAKIE Bus Wake-up Activity Interrupt Enable bit

Value	Description
1	Wake-up activity interrupt is enabled.
0	Wake-up activity interrupt is disabled.

Bit 29 – CERRIE CAN Bus Error Interrupt Enable bit

Value	Description
1	CAN bus error interrupt is enabled.
0	CAN bus error interrupt is disabled.

Bit 28 – SERRIE System Error Interrupt Enable bit

Value	Description
1	System error interrupt is enabled.
0	System error interrupt is disabled.

Bit 27 – RXOVIE Receive Buffer Overflow Interrupt Enable bit

Value	Description
1	Receive buffer overflow interrupt is enabled.
0	Receive buffer overflow interrupt is disabled.

Bit 26 – TXATIE Transmit Attempt Interrupt Enable bit

Value	Description
1	Transmit attempt interrupt is enabled.
0	Transmit attempt interrupt is disabled.

Bit 20 – TEFIE Transmit Event FIFO Interrupt Enable bit

Value	Description
1	Transmit event FIFO interrupt is enabled.
0	Transmit event FIFO interrupt is disabled.

Bit 19 – MODIE Mode Change Interrupt Enable bit

Value	Description
1	Mode change interrupt is enabled.
0	Mode change interrupt is disabled.

Bit 18 – TBCIE CAN Timer Interrupt Enable bit

Value	Description
1	CAN timer interrupt is enabled.
0	CAN timer interrupt is disabled.

Bit 17 – RXIE Receive Object Interrupt Enable bit

Value	Description
1	Receive object interrupt is enabled.
0	Receive object interrupt is disabled.

Bit 16 – TXIE Transmit Object Interrupt Enable bit

Value	Description
1	Transmit object interrupt is enabled.
0	Transmit object interrupt is disabled.

Bit 15 – IVMIF Invalid Message Interrupt Flag bit[(2)]

Value	Description
1	Invalid message interrupt occurred.
0	No invalid message interrupt

Bit 14 – WAKIF Bus Wake-up Activity Interrupt Flag bit[(2)]

Value	Description
1	Wake-up activity interrupt occurred.
0	No wake-up activity interrupt

Bit 13 – CERRIF CAN Bus Error Interrupt Flag bit[(2)]

Value	Description
1	CAN bus error interrupt occurred.
0	No CAN bus error interrupt

Bit 12 – SERRIF System Error Interrupt Flag bit(2)

Value	Description
1	System error interrupt occurred.
0	No system error interrupt

Bit 11 – RXOVIF Receive Buffer Overflow Interrupt Flag bit

Value	Description
1	Receive buffer overflow interrupt occurred.
0	No receive buffer overflow interrupt

Bit 10 – TXATIF Transmit Attempt Interrupt Flag bit

Value	Description
1	Transmit attempt interrupt occurred.
0	No transmit attempt interrupt

Bit 4 – TEFIF Transmit Event FIFO Interrupt Flag bit

Value	Description
1	Transmit event FIFO interrupt occurred.
0	No transmit event FIFO interrupt

Bit 3 – MODIF CAN Mode Change Interrupt Flag bit(2)

Value	Description
1	A CAN module mode change occurred (OPMOD[2:0] has changed to reflect REQOP[2:0]).
0	No mode change occurred.

Bit 2 – TBCIF CAN Timer Overflow Interrupt Flag bit(2)

Value	Description
1	TBC has overflowed.
0	TBC has not overflowed.

Bit 1 – RXIF Receive Object Interrupt Flag bit

Value	Description
1	Receive object interrupt is pending.
0	No Receive object interrupts are pending.

Bit 0 – TXIF Transmit Object Interrupt Flag bit

Value	Description
1	Transmit object interrupt is pending.
0	No transmit object interrupts are pending.

14.4.9. CAN Receive Interrupt Status Register

Name: C1RXIF
Offset: 0x2620

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxRXIFT: Accesses the top byte RXIF[31:24]
   - CxRXIFU: Accesses the upper byte RXIF[23:16]
   - CxRXIF: Accesses the byte RXIF[15:8]
2. RFIFx is the 'or' of all enabled RX FIFO flags (individual flags need to be cleared in the FIFO register).

Bit	31	30	29	28	27	26	25	24
	RFIF[30:23]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RFIF[22:15]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RFIF[14:7]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RFIF[6:0]							
Access	R	R	R	R	R	R	R	
Reset	0	0	0	0	0	0	0	

Bits 31:1 – RFIF[30:0] Receive FIFO Interrupt Pending bits

Value	Description
1	One or more enabled receive FIFO interrupts are pending for the respective FIFO.
0	No enabled receive FIFO interrupts for the respective FIFO are pending.

14.4.10. CAN Transmit Interrupt Status Register

Name: C1TXIF
Offset: 0x2624

Notes:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTXIFT: Accesses the top byte TXIF[31:24]
   - CxTXIFU: Accesses the upper byte TXIF[23:16]
   - CxTXIF: Accesses the byte TXIF[31:0]
2. TFIFx is the ‘or’ of all enabled TX FIFO flags (individual flags need to be cleared in the FIFO register).

Bit	31	30	29	28	27	26	25	24
	TFIF[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TFIF[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TFIF[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TFIF[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TFIF[31:0] Transmit FIFO/TXQ Interrupt Pending bits

Value	Description
1	One or more enabled transmit FIFO/TXQ interrupts are pending for the respective FIFO/TXQ.
0	No enabled transmit FIFO/TXQ interrupts for the respective FIFO/TXQ are pending.

14.4.11. CAN Receive Overflow Interrupt Status Register

Name: C1RXOVIF
Offset: 0x2628

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   – CxRXOVIFT: Accesses the top byte RXOVIF[31:24]
   – CxRXOVIFU: Accesses the upper byte RXOVIF[23:16]
   – CxRXOVIF: Accesses the byte RXOVIF[15:8]
2. RFOVIFx mirrors the overflow bit of its respective FIFO register, individual flags need to be cleared in said FIFO register.

Bit	31	30	29	28	27	26	25	24
	RFOVIF[30:23]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RFOVIF[22:15]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RFOVIF[14:7]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RFOVIF[6:0]							
Access	R	R	R	R	R	R	R	
Reset	0	0	0	0	0	0	0	

Bits 31:1 – RFOVIF[30:0] Receive FIFO Overflow Interrupt Pending bits

Value	Description
1	Interrupt is pending.
0	Interrupt is not pending.

14.4.12. CAN Transmit Attempt Interrupt Status Register

Name: C1TXATIF
Offset: 0x262C

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTXATIFT: Accesses the top byte TXATIF[31:24]
   - CxTXATIFU: Accesses the upper byte TXATIF[23:16]
   - CxTXATIF: Accesses the byte TXATIF[31:0]
2. TFATIFx mirrors the transmit attempt interrupt bit of its respective FIFO register, individual flags need to be cleared in said FIFO register.

Bit	31	30	29	28	27	26	25	24
	TFATIF[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TFATIF[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TFATIF[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TFATIF[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TFATIF[31:0] Transmit FIFO/TXQ Attempt Interrup Pending bits

Value	Description
1	Interrupt is pending.
0	Interrupt is not pending.

14.4.13. CAN Transmit Request Register

Name: C1TXREQ
Offset: 0x2630

Notes:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTXREQT: Accesses the top byte TXREQ[31:24]
   - CxTXREQU: Accesses the upper byte TXREQ[23:16]
   - CxTXREQ: Accesses the byte TXREQ[31:0]
2. These bits are only valid if the associated objects are configured as transmit objects (TXEN = 1). Otherwise, setting them has no effect.

Legend: S = Settable bit; HC = Hardware Clearable bit; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	TXREQ[30:23]							
Access	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TXREQ[22:15]							
Access	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TXREQ[14:7]							
Access	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXREQ[6:0]							TXREQ
Access	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC	S/HC
Reset	0	0	0	0	0	0	0	0

Bits 31:1 – TXREQ[30:0] Message Send Request bits

TXEN = 1 (Object configured as a Transmit Object)
Setting this bit to '1' requests sending a message.
The bit will automatically clear when the message(s) queued in the object is (are) successfully sent.
This bit CANNOT be used to abort a transmission.
TXEN = 0 (Object configured as a Receive Object)
This bit has no effect.

Bit 0 – TXREQ Transmit Queue Message Send Request bit

Setting this bit to '1' requests sending a message.
The bit will automatically clear when the message(s) queued in the object is (are) successfully sent.
This bit CANNOT be used to abort a transmission.

14.4.14. CAN Transmit/Receive Error Count Register

Name: C1TREC
Offset: 0x2634

Note:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTRECT: Accesses the top byte TREC[31:24]
   - CxTRECU: Accesses the upper byte TREC[23:16]
   - CxTREC: Accesses the byte TREC[31:0]

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
Access			R	R	R	R	R	R
Reset			1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TERRCNT[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RERRCNT[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 21 – TXBO Transmitter in Error Bus Off State bit (TERRCNT[7:0] > 255)
In Configuration mode, TXBO is set since the module is not on the bus.

Bit 20 – TXBP Transmitter in Error Bus Passive State bit (TERRCNT[7:0] > 127)

Bit 19 – RXBP Receiver in Error Bus Passive State bit (RERRCNT[7:0] > 127)

Bit 18 – TXWARN Transmitter in Error Warning State bit (128 > TERRCNT[7:0] > 127)

Bit 17 – RXWARN Receiver in Error Warning State bit (128 > RERRCNT[7:0] > 127)

Bit 16 – EWARN Transmitter or Receiver is in Error Warning State bit

Bits 15:8 – TERRCNT[7:0] Transmit Error Counter bits

Bits 7:0 – RERRCNT[7:0] Receive Error Counter bits

14.4.15. CAN Bus Diagnostics Register 0

Name: C1BDIAG0
Offset: 0x2638

Note:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxBDIAG0T: Accesses the top byte BDIAG0[31:24]
   - CxBDIAG0U: Accesses the upper byte BDIAG0[23:16]
   - CxBDIAG0: Accesses the byte BDIAG0[31:0]

Bit	31	30	29	28	27	26	25	24
	DTERRCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DRERRCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NTERRCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	NRERRCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DTERRCNT[7:0] Data Bit Rate Transmit Error Counter bits

Bits 23:16 – DRERRCNT[7:0] Data Bit Rate Receive Error Counter bits

Bits 15:8 – NTERRCNT[7:0] Nominal Bit Rate Transmit Error Counter bits

Bits 7:0 – NRERRCNT[7:0] Nominal Bit Rate Receive Error Counter bits

14.4.16. CAN Bus Diagnostics Register 1

Name: C1BDIAG1
Offset: 0x263C

Note:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxBDIAG1T: Accesses the top byte BDIAG1[31:24]
   - CxBDIAG1U: Accesses the upper byte BDIAG1[23:16]
   - CxBDIAG1: Accesses the byte BDIAG1[31:0]

Bit	31	30	29	28	27	26	25	24
	DLCMM	ESI	DCRCERR	DSTUFERR	DFORMERR		DBIT1ERR	DBIT0ERR
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	23	22	21	20	19	18	17	16
	TXBOERR		NCRCERR	NSTUFERR	NFORMERR	NACKERR	NBIT1ERR	NBIT0ERR
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EFMSGCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EFMSGCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DLCMM DLC Mismatch bit
During a transmission or reception, the specified DLC is larger than the PLSIZEx of the FIFO element.

Bit 30 – ESI ESI Flag of Received CAN FD Message Set bit

Bit 29 – DCRCERR Received Message with CRC Incorrect Checksum in the Data Segment bit
The CRC Checksum of a received message is considered incorrect if the CRC of the incoming message does not match with the CRC calculated from the received data.

Bit 28 – DSTUFERR Received Message with Illegal Sequence in the Data Segment bit
An Illegal Sequence occurs when more than five equal bits appear in sequence in a part of the received message where this is not allowed.

Bit 27 – DFORMERR Received Frame with a Fixed Format Error in Data Segment bit
A fixed format error occurs when a part of the incoming frame with a fixed format has the wrong format.

Bit 25 – DBIT1ERR Transmitted Message Recessive Level in Data Segment bit
During the data segment of a message transmission, the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.

Bit 24 – DBIT0ERR Transmitted Message Dominant Level in Data Segment bit
During the transmission of a message, the device wanted to send a dominant level (logical value '0'), but the monitored bus value was recessive.

Bit 23 – TXBOERR Device Went to Bus Off bit

Bit 21 – NCRCERR Received Message with CRC Incorrect Checksum in Non-Data Segment bit
The CRC checksum of a received message is considered incorrect if the CRC of the incoming message does not match the CRC calculated from the received data.

Bit 20 – NSTUFERR Received Message with Illegal Sequence in Non-Data Segment bit
An Illegal Sequence occurs when more than five equal bits appear in sequence in a part of the received message where this is not allowed.

Bit 19 – NFORMERR Received Frame with a Fixed Format Error in Non-Data Segment bit
A fixed format error occurs when a part of the incoming frame with a fixed format has the wrong format.

Bit 18 – NACKERR Transmitted Message Not Acknowledged bit
Transmitted message was not Acknowledged.

Bit 17 – NBIT1ERR Transmitted Message Dominant Level in Non-Data Segment bit
During the non-data segment of a message transmission, the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.

Bit 16 – NBIT0ERR Transmitted Message Dominant Level in Non-Data Segment bit
During the transmission of a message (or Acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value '0'), but the monitored bus value was recessive. During bus-off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the progress of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

Bits 15:0 – EFMSGCNT[15:0] Error-Free Message Counter bits

14.4.17. CAN Transmit Event FIFO Control Register

Name: C1TEFCON
Offset: 0x2640

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   – CxTEFCONT: Accesses the top byte TEFCON[31:24]
   – CxTEFCONU: Accesses the upper byte TEFCON[23:16]
   – CxTEFCON: Accesses the byte TEFCON[31:0]
2. These bits can only be modified in Configuration mode (OPMOD[2:0] = `100`.

Bit	31	30	29	28	27	26	25	24
				FSIZE[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
						FRESET		UINC
Access						S/HC		S/HC
Reset						1		0

Bit	7	6	5	4	3	2	1	0
			TEFTSEN		TEFOVIE	TEFFIE	TEFHIE	TEFNEIE
Access			R/W		R/W	R/W	R/W	R/W
Reset			0		0	0	0	0

Bits 28:24 – FSIZE[4:0] FIFO Size[(2)] bits

Value	Description
`11111`	FIFO is 32 messages deep.
`00010`	FIFO is 3 messages deep.
`00001`	FIFO is 2 messages deep.
`00000`	FIFO is 1 message deep.

Bit 10 – FRESET FIFO Reset bit

Value	Description
`1`	FIFO will be reset when bit is set, cleared by hardware when FIFO is reset; the user needs to poll whether this bit is clear before taking any action.
`0`	No effect

Bit 8 – UINC Increment Tail bit

Value	Description
`1`	When this bit is set, the FIFO tail will increment by a single message.
`0`	FIFO tail will not increment.

Bit 5 – TEFTSEN Transmit Event FIFO Timestamp Enable[(2)] bit

Value	Description
1	Time-stamps elements in TEF.
0	Does not timestamp elements in TEF.

Bit 3 – TEFOVIE Transmit Event FIFO Overflow Interrupt Enable bit

Value	Description
1	Interrupt is enabled for overflow event.
0	Interrupt is disabled for overflow event.

Bit 2 – TEFFIE Transmit Event FIFO Full Interrupt Enable bit

Value	Description
1	Interrupt is enabled for FIFO full.
0	Interrupt is disabled for FIFO full.

Bit 1 – TEFHIE Transmit Event FIFO Half Full Interrupt Enable bit

Value	Description
1	Interrupt is enabled for FIFO half full.
0	Interrupt is disabled for FIFO half full.

Bit 0 – TEFNEIE Transmit Event FIFO Not Empty Interrupt Enable bit

Value	Description
1	Interrupt is enabled for FIFO not empty.
0	Interrupt is disabled for FIFO not empty.

14.4.18. CAN Transmit Event FIFO Status Register

Name: C1TEFSTA
Offset: 0x2644

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTEFSTAT: Accesses the top byte TEFSTA[31:24]
   - CxTEFSTAU: Accesses the upper byte TEFSTA[23:16]
   - CxTEFSTA: Accesses the byte TEFSTA[31:0]
2. These bits are read-only and reflect the status of the FIFO.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					TEFOVIF	TEFFIF	TEFHIF	TEFNEIF
Access					HS/C	R	R	R
Reset					0	0	0	0

Bit 3 – TEFOVIF Transmit Event FIFO Overflow Interrupt Flag bit

Value	Description
1	Overflow event has occurred.
0	No overflow event has occurred.

Bit 2 – TEFFIF Transmit Event FIFO Full Interrupt Flag(2) bit

Value	Description
1	FIFO is full.
0	FIFO is not full.

Bit 1 – TEFHIF Transmit Event FIFO Half Full Interrupt Flag(2) bit

Value	Description
1	FIFO is greater than or equal to half full.
0	FIFO is less than half full.

Bit 0 – TEFNEIF Transmit Event FIFO Not Empty Interrupt Flag(2) bit

Value	Description
1	FIFO is not empty.
0	FIFO is empty.

14.4.19. CAN Transmit Event FIFO User Address Register

Name: C1TEFUA
Offset: 0x2648

Notes:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTEFUAT: Accesses the top byte TEFUA[31:24]
   - CxTEFUAU: Accesses the upper byte TEFUA[23:16]
   - CxTEFUA: Accesses the byte TEFUA[31:0]
2. This register is not ensured to read correctly in Configuration mode and may only be accessed when the module is not in Configuration mode.

Bit	31	30	29	28	27	26	25	24
	TEFUA[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	23	22	21	20	19	18	17	16
	TEFUA[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	15	14	13	12	11	10	9	8
	TEFUA[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	TEFUA[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 31:0 – TEFUA[31:0] Transmit Event FIFO User Address bits
A read of this register will return the address where the next event is to be read (FIFO tail).

14.4.20. CAN Message Memory Base Address Register

Name: C1FIFOBA
Offset: 0x264C

Legend: R = Readable; W = Writable bit; U = Unimplemented bit

Bit	31	30	29	28	27	26	25	24
	FIFOBA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FIFOBA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FIFOBA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FIFOBA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – FIFOBA[31:0] Message Memory Base Address bits
Defines the base address for the transmit event FIFO, followed by the message objects.

14.4.21. CAN Transmit Queue Control Register

Name: C1TXQCON
Offset: 0x2650

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTXQCONT: Accesses the top byte TXQCON[31:24]
   - CxTXQCONU: Accesses the upper byte TXQCON[23:16]
   - CxTXQCON: Accesses the byte TXQCON[31:0]
2. These bits can only be modified in Configuration mode (OPMOD[2:0] = `100`.

Bit	31	30	29	28	27	26	25	24
	PLSIZE[2:0]			FSIZE[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		TXAT[1:0]		TXPRI[4:0]				
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		1	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						FRESET	TXREQ	UINC
Access						S/HC	R/W/HC	S/HC
Reset						1	0	0

Bit	7	6	5	4	3	2	1	0
	TXEN			TXATIE		TXQEIE		TXQNIE
Access	R			R/W		R/W		R/W
Reset	1			0		0		0

Bits 31:29 – PLSIZE[2:0] Payload Size[(2)] bits

Value	Description
`111`	64 data bytes
`110`	48 data bytes
`101`	32 data bytes
`100`	24 data bytes
`011`	20 data bytes
`010`	16 data bytes
`001`	12 data bytes
`000`	8 data bytes

Bits 28:24 – FSIZE[4:0] FIFO Size[(2)] bits

Value	Description
`11111`	FIFO is 32 messages deep.
`00010`	FIFO is 3 messages deep.
`00001`	FIFO is 2 messages deep.
`00000`	FIFO is 1 messages deep.

Bits 22:21 – TXAT[1:0] Retransmission Attempts bits
This feature is enabled when RTXAT (CxCON[16]) is set.

Value	Description
11	Unlimited number of retransmission attempts
10	Unlimited number of retransmission attempts
01	Three retransmission attempts
00	Disable retransmission attempts

Bits 20:16 – TXPRI[4:0] Message Transmit Priority bits

Value	Description
11111	Highest message priority
00000	Lowest message priority

Bit 10 – FRESET FIFO Reset bit

Value	Description
1	FIFO will be reset when this bit is set. It is cleared by hardware when the FIFO is reset; the user needs to poll whether this bit is clear before taking any action.
0	No effect

Bit 9 – TXREQ Message Send Request bit

Value	Description
1	Requests sending a message; the bit will automatically clear when all the messages queued in the TXQ are successfully sent.
0	Clearing the bit to '0' while set ('1') will request a message abort.

Bit 8 – UINC Increment Head/Tail bit

When this bit is set, the FIFO head will increment by a single message.

Bit 7 – TXEN TX Enable bit

Value	Description
1	The transmit message queue is always configured as a transmitter; this bit will always read as '1'.

Bit 4 – TXATIE Transmit Attempts Exhausted Interrupt Enable bit

Value	Description
1	Enables interrupt
0	Disables interrupt

Bit 2 – TXQEIE Transmit Queue Empty Interrupt Enable bit

Value	Description
1	Interrupt is enabled for TXQ empty.
0	Interrupt is disabled for TXQ empty.

Bit 0 – TXQNIE Transmit Queue Not Full Interrupt Enable bit

Value	Description
1	Interrupt is enabled for TXQ not full.
0	Interrupt is disabled for TXQ not full.

14.4.22. CAN Transmit Queue Status Register

Name: C1TXQSTA
Offset: 0x2654

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTXQSTAT: Accesses the top byte TXQSTA[31:24]
   - CxTXQSTAU: Accesses the upper byte TXQSTA[23:16]
   - CxTXQSTA: Accesses the byte TXQSTA[31:0]
2. The TXQCI[4:0] bits give a zero-indexed value to the message in the TXQ. If the TXQ is four messages deep (FSIZE = 3), TXQCIx will take on a value of 0 to 3, depending on the state of the TXQ.
3. These bits are updated when a message completes (or aborts) or when the TXQ is reset.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				TXQCI[4:0]				
Access				R	R	R	R	R
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXABT	TXLARB	TXERR	TXATIF		TXQEIF		TXQNIF
Access	R	R	R	HS/C		R		R
Reset	0	0	0	0		1		1

Bits 12:8 – TXQCI[4:0] Transmit Queue Message Index(2)
A read of this register will return an index to the message that the FIFO will next attempt to transmit.

Bit 7 – TXABT Message Aborted Status(3) bit

Value	Description
1	Message was aborted.
0	Message completed successfully.

Bit 6 – TXLARB Message Lost Arbitration Status(3) bit

Value	Description
1	Message lost arbitration while being sent.
0	Message did not lose arbitration while being sent.

Bit 5 – TXERR Error Detected During Transmission(3) bit

Value	Description
1	A bus error occurred while the message was being sent.

Value	Description
0	A bus error did not occur while the message was being sent.

Bit 4 – TXATIF Transmit Attempts Exhausted Interrupt Pending bit

Value	Description
1	Interrupt is pending.
0	Interrupt is not pending.

Bit 2 – TXQEIF Transmit Queue Empty Interrupt Flag bit

Value	Description
1	TXQ is empty.
0	TXQ is not empty; at least one message is queued to be transmitted.

Bit 0 – TXQNIF Transmit Queue Not Full Interrupt Flag bit

Value	Description
1	TXQ is not full.
0	TXQ is full.

14.4.23. CAN Transmit Queue User Address Register

Name: C1TXQUA
Offset: 0x2658

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxTXQUAT: Accesses the top byte TXQUA[31:24]
   - CxTXQUAU: Accesses the upper byte TXQUA[23:16]
   - CxTXQUA: Accesses the high byte TXQUA[31:0]
2. This register is not ensured to read correctly in Configuration mode and may only be accessed when the module is not in Configuration mode.

Bit	31	30	29	28	27	26	25	24
	TXQUA[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	23	22	21	20	19	18	17	16
	TXQUA[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	15	14	13	12	11	10	9	8
	TXQUA[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	TXQUA[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 31:0 – TXQUA[31:0] TXQ User Address bits
A read of this register will return the address where the next message is to be written (TXQ head).

14.4.24. CAN 1 FIFO x Control Register

Name: C1FIFOCONx
Offset: 0x265C, 0x2668, 0x2674, 0x2680, 0x268C, 0x2698, 0x26A4

Notes:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - C1FIFOCONyT: Accesses the top byte FIFOCONy[31:24].
   - C1FIFOCONyU: Accesses the upper byte FIFOCONy[23:16].
   - C1FIFOCONy: Accesses the byte FIFOCONy[31:0].
2. [x] denotes FIFO number, from 1 to 7.
3. These bits can only be modified in Configuration mode (OPMOD[2:0] = 100).

Bit	31	30	29	28	27	26	25	24
	PLSIZE[2:0]			FSIZE[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
		TXAT[1:0]		TXPRI[4:0]				
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		1	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						FRESET	TXREQ	UINC
Access						S/HC	R/W/HC	S/HC
Reset						1	0	0

Bit	7	6	5	4	3	2	1	0
	TXEN	RTREN	RXTSEN	TXATIE	RXOVIE	TFERFFIE	TFHRFHIE	TFNRFNIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:29 – PLSIZE[2:0] Payload Size bits(3)

Value	Description
111	64 data bytes
110	48 data bytes
101	32 data bytes
100	24 data bytes
011	20 data bytes
010	16 data bytes
001	12 data bytes
000	8 data bytes

Bits 28:24 – FSIZE[4:0] FIFO Size bits(3)

Value	Description
11111	FIFO is 32 messages deep.
...	
00001	FIFO is 2 messages deep.
00000	FIFO is 1 message deep.

Bits 22:21 – TXAT[1:0] Retransmission Attempts bits
This feature is enabled when RTXAT (CxCON[16]) is set.

Value	Description
11	Unlimited number of retransmission attempts
10	Unlimited number of retransmission attempts
01	Three retransmission attempts
00	Disables retransmission attempts

Bits 20:16 – TXPRI[4:0] Message Transmit Priority bits

Value	Description
11111	Highest message priority
00000	Lowest message priority

Bit 10 – FRESET FIFO Reset bit

Value	Description
1	FIFO will be reset when bit is set, cleared by hardware whenever FIFO is reset; the user needs to poll whether this bit is clear before taking any action.
0	No effect

Bit 9 – TXREQ Message Send Request bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit FIFO)	Requests sending a message; the bit will automatically clear when all the messages queued in the FIFO are successfully sent.
0	TXEN = 1 (FIFO configured as a transmit FIFO)	Clearing the bit to '0' while set ('1') will request a message abort.
x	TXEN = 0 (FIFO configured as a receive FIFO)	This bit has no effect.

Bit 8 – UINC Increment Head/Tail bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit FIFO)	When this bit is set, the FIFO head will increment by a single message.
1	TXEN = 0 (FIFO configured as a receive FIFO)	When this bit is set, the FIFO tail will increment by a single message.

Bit 7 – TXEN TX/RX Buffer Selection bit

Value	Description
1	Transmits message object.
0	Receives message object.

Bit 6 – RTREN Auto-Remove Transmit (RTR) Enable bit

Value	Description
1	When a Remote Transmit is received, TXREQ will be set.
0	When a Remote Transmit is received, TXREQ will be unaffected.

Bit 5 – RXTSEN Received Message Timestamp Enable bit[3]

Value	Description
1	Captures timestamp in a received message object in RAM.
0	Does not capture timestamp

Bit 4 – TXATIE Transmit Attempts Exhausted Interrupt Enable bit

Value	Description
1	Enables interrupt.
0	Disables interrupt.

Bit 3 – RXOVIE Overflow Interrupt Enable bit

Value	Description
1	Interrupt is enabled for overflow event.
0	Interrupt is disabled for overflow event.

Bit 2 – TFERFFIE Transmit/Receive FIFO Empty/Full Interrupt Enable bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit FIFO)	Interrupt is enabled for FIFO empty.
0	TXEN = 1 (FIFO configured as a transmit FIFO)	Interrupt is disabled for FIFO empty.
1	TXEN = 0 (FIFO configured as a receive FIFO)	Interrupt is enabled for FIFO full.
0	TXEN = 0 (FIFO configured as a receive FIFO)	Interrupt is disabled for FIFO full.

Bit 1 – TFHRFHIE Transmit/Receive FIFO Half Empty/Half Full Interrupt Enable bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit FIFO)	Interrupt is enabled for FIFO half empty.
0	TXEN = 1 (FIFO configured as a transmit FIFO)	Interrupt is disabled for FIFO half empty.
1	TXEN = 0 (FIFO configured as a receive FIFO)	Interrupt is enabled for FIFO half-full.
0	TXEN = 0 (FIFO configured as a receive FIFO)	Interrupt is disabled for FIFO half-full.

Bit 0 – TFNRFNIE Transmit/Receive FIFO Not Full/Not Empty Interrupt Enable bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit FIFO)	Interrupt is enabled for FIFO not full.
0	TXEN = 1 (FIFO configured as a transmit FIFO)	Interrupt is disabled for FIFO not full.
1	TXEN = 0 (FIFO configured as a receive FIFO)	Interrupt is enabled for FIFO not empty.
0	TXEN = 0 (FIFO configured as a receive FIFO)	Interrupt is disabled for FIFO not empty.

14.4.25. CAN 1 FIFO x Status Register[(1), (2)]

Name: C1FIFOSTAx
Offset: 0x2660, 0x266C, 0x2678, 0x2684, 0x2690, 0x269C, 0x26A8

Notes:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxFIFOSTAyT: Accesses the top byte FIFOSTAy[31:24].
   - CxFIFOSTAyU: Accesses the upper byte FIFOSTAy[23:16].
   - CxFIFOSTAy: Accesses the byte FIFOSTAy[31:0].
2. [x] denotes FIFO number, from 1 to 7.
3. FIFOCI[4:0] gives a zero-indexed value to the message in the FIFO. If the FIFO is four messages deep (FSIZE = 3), FIFOCIx will take on a value of 0 to 3, depending on the state of the FIFO.
4. This bit is updated when a message completes (or aborts) or when the FIFO is reset.
5. This bit is reset on any read of this register or when the TXQ is reset.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				FIFOCI[4:0]				
Access				[R	[R	[R	[R	[R
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXABT	TXLARB	TXERR	TXATIF	RXOVIF	TFERFFIF	TFHRFHIF	TFNRFNIF
Access	R	R	R	HS/C	HS/C	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 12:8 – FIFOCI[4:0] FIFO Message Index bits[(3)]

Condition	Description
TXEN = 1 (FIFO configured as a transmit buffer)	A read of this register will return an index to the message that the FIFO will next attempt to transmit.
TXEN = 0 (FIFO configured as a receive buffer)	A read of this register will return an index to the message that the FIFO will use to save the next message.

Bit 7 – TXABT Message Aborted Status bit[(5)]

Value	Description
1	Message was aborted.
0	Message completed successfully.

Bit 6 – TXLARB Message Lost Arbitration Status bit[(4)]

Value	Description
1	Message lost arbitration while being sent.
0	Message did not lose arbitration while being sent.

Bit 5 – TXERR Error Detected During Transmission bit(4)

Value	Description
1	A bus error occurred while the message was being sent.
0	A bus error did not occur while the message was being sent.

Bit 4 – TXATIF Transmit Attempts Exhausted Interrupt Pending bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit buffer)	Interrupt is pending.
0	TXEN = 1 (FIFO configured as a transmit buffer)	Interrupt is not pending.
x	TXEN = 0 (FIFO configured as a receive buffer)	Unused, reads as '0'.

Bit 3 – RXOVIF Receive FIFO Overflow Interrupt Flag bit

Value	Condition	Description
x	TXEN = 1 (FIFO configured as a transmit buffer)	Unused, reads as '0'.
1	TXEN = 0 (FIFO configured as a receive buffer)	Overflow event has occurred.
0	TXEN = 0 (FIFO configured as a receive buffer)	No overflow event occurred.

Bit 2 – TFERFFIF Transmit/Receive FIFO Empty/Full Interrupt Flag bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit buffer)	FIFO is empty.
0	TXEN = 1 (FIFO configured as a transmit buffer)	FIFO is not empty, at least one message is queued to be transmitted.
1	TXEN = 0 (FIFO configured as a receive buffer)	FIFO is full.
0	TXEN = 0 (FIFO configured as a receive buffer)	FIFO is not full.

Bit 1 – TFHRFHIF Transmit/Receive FIFO Half Empty/Half Full Interrupt Flag bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit buffer)	FIFO is less than or equal to half-full.
0	TXEN = 1 (FIFO configured as a transmit buffer)	FIFO is greater than half-full
1	TXEN = 0 (FIFO configured as a receive buffer)	FIFO is greater than or equal to half-full
0	TXEN = 0 (FIFO configured as a receive buffer)	FIFO is less than half-full

Bit 0 – TFNRFNIF Transmit/Receive FIFO Not Full/Not Empty Interrupt Flag bit

Value	Condition	Description
1	TXEN = 1 (FIFO configured as a transmit buffer)	FIFO is not full.
0	TXEN = 1 (FIFO configured as a transmit buffer)	FIFO is full.
1	TXEN = 0 (FIFO configured as a receive buffer)	FIFO is not empty, has at least one message.
0	TXEN = 0 (FIFO configured as a receive buffer)	FIFO is empty.

14.4.26. CAN FIFO x User Address Register[(1), (2), (3)]

Name: C1FIFOUAx
Offset: 0x2664, 0x2670, 0x267C, 0x2688, 0x2694, 0x26A0, 0x26AC

Notes:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxFIFOCONyT: Accesses the top byte FIFOCONy[31:24].
   - CxFIFOCONyU: Accesses the upper byte FIFOCONy[23:16].
   - CxFIFOCONy: Accesses the byte FIFOCONy[31:0].
2. [x] denotes FIFO number, from 1 to 7.
3. This register is not ensured to read correctly in Configuration mode and may only be accessed when the module is not in Configuration mode.

Bit	31	30	29	28	27	26	25	24
	FIFOUA[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	23	22	21	20	19	18	17	16
	FIFOUA[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	15	14	13	12	11	10	9	8
	FIFOUA[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	FIFOUA[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 31:0 – FIFOUA[31:0] FIFO User Address bits

Condition	Description
TXEN = 1 (FIFO configured as transmit buffer	A read of this register will return the address where the next message is to be written (FIFO head).
TXEN = 0 (FIFO configured as receive buffer	A read of the register will return the address where the next message is to be read (FIFO tail).

14.4.27. CAN Filter Control Register 0[1]

Name: C1FLTCON0
Offset: 0x26B0

Note:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxFLTCON0T: Accesses the top byte FLTCON0[31:24].
   - CxFLTCON0U: Accesses the upper byte FLTCON0[23:16].
   - CxFLTCON0: Accesses the byte FLTCON0[31:0].

Bit	31	30	29	28	27	26	25	24
	FLTEN3			F3BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FLTEN2			F2BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FLTEN1			F1BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLTEN0			F0BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 31 – FLTEN3 Enable Filter 3 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 28:24 – F3BP[4:0] Pointer to FIFO when Filter 3 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 23 – FLTEN2 Enable Filter 2 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 20:16 – F2BP[4:0] Pointer to FIFO when Filter 2 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 15 – FLTEN1 Enable Filter 1 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 12:8 – F1BP[4:0] Pointer to FIFO when Filter 1 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 7 – FLTEN0 Enable Filter 0 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 4:0 – F0BP[4:0] Pointer to FIFO when Filter 0 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

14.4.28. CAN Filter Control Register 1(1)

Name: C1FLTCON1
Offset: 0x26B4

Note:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxFLTCON1T: Accesses the top byte FLTCON1[31:24].
   - CxFLTCON1U: Accesses the upper byte FLTCON1[23:16].
   - CxFLTCON1: Accesses the byte FLTCON1[31:0].

Bit	31	30	29	28	27	26	25	24
	FLTEN7			F7BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FLTEN6			F6BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FLTEN5			F5BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLTEN4			F4BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 31 – FLTEN7 Enable Filter 7 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 28:24 – F7BP[4:0] Pointer to FIFO when Filter 7 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 23 – FLTEN6 Enable Filter 6 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 20:16 – F6BP[4:0] Pointer to FIFO when Filter 6 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 15 – FLTEN5 Enable Filter 5 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 12:8 – F5BP[4:0] Pointer to FIFO when Filter 5 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 7 – FLTEN4 Enable Filter 4 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 4:0 – F4BP[4:0] Pointer to FIFO when Filter 4 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

14.4.29. CAN Filter Control Register 2(1)

Name: C1FLTCON2
Offset: 0x26B8

Note:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxFLTCON2T: Accesses the top byte FLTCON2[31:24].
   - CxFLTCON2U: Accesses the upper byte FLTCON2[23:16].
   - CxFLTCON2: Accesses the byte FLTCON2[31:0].

Bit	31	30	29	28	27	26	25	24
	FLTEN11			F11BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FLTEN10			F10BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FLTEN9			F9BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLTEN8			F8BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 31 – FLTEN11 Enable Filter 11 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 28:24 – F11BP[4:0] Pointer to FIFO when Filter 11 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 23 – FLTEN10 Enable Filter 10 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 20:16 – F10BP[4:0] Pointer to FIFO when Filter 10 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 15 – FLTEN9 Enable Filter 9 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 12:8 – F9BP[4:0] Pointer to FIFO when Filter 9 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 7 – FLTEN8 Enable Filter 8 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 4:0 – F8BP[4:0] Pointer to FIFO when Filter 8 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

14.4.30. CAN Filter Control Register 3(1)

Name: C1FLTCON3
Offset: 0x26BC

Note:
1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxFLTCON1T: Accesses the top byte FLTCON1[31:24].
   - CxFLTCON1U: Accesses the upper byte FLTCON1[23:16].
   - CxFLTCON1: Accesses the byte FLTCON1[31:0].

Bit	31	30	29	28	27	26	25	24
	FLTEN15			F15BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FLTEN14			F14BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FLTEN13			F13BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLTEN12			F12BP[4:0]				
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 31 – FLTEN15 Enable Filter 15 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 28:24 – F15BP[4:0] Pointer to FIFO when Filter 15 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 23 – FLTEN14 Enable Filter 14 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 20:16 – F14BP[4:0] Pointer to FIFO when Filter 14 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 15 – FLTEN13 Enable Filter 13 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 12:8 – F13BP[4:0] Pointer to FIFO when Filter 13 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

Bit 7 – FLTEN12 Enable Filter 12 to Accept Messages bit

Value	Description
1	Filter is enabled.
0	Filter is disabled.

Bits 4:0 – F12BP[4:0] Pointer to FIFO when Filter 12 Hits bits

Value	Description
11111-00100	Reserved
00011	Message matching filter is stored in FIFO 3.
00010	Message matching filter is stored in FIFO 2.
00001	Message matching filter is stored in FIFO 1.
00000	Reserved, FIFO 0 is the TX Queue and cannot receive messages.

14.4.31. CAN Filter x Object Register(1), (2)

Name: C1FLTOBJx
Offset: 0x26C0, 0x26C8, 0x26D0, 0x26D8, 0x26E0, 0x26E8, 0x26F0, 0x26F8, 0x2700, 0x2708, 0x2710, 0x2718, 0x2720, 0x2728, 0x2730, 0x2738

Notes:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxFLTOBJyT: Accesses the top byte FLTOBJy[31:24].
   - CxFLTOBJyU: Accesses the upper byte FLTOBJy[23:16].
   - CxFLTOBJy: Accesses the byte FLTOBJy[31:0].
2. [x] denotes Filter number, from 0 to 15.

Bit	31	30	29	28	27	26	25	24
		EXIDE	SID11	EID[17:13]				
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EID[12:5]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EID[4:0]					SID[10:8]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 30 – EXIDE Extended Identifier Enable bit

Value	Condition	Description
1	MIDE = 1	Matches only messages with Extended Identifier addresses.
0	MIDE = 1	Matches only messages with Standard Identifier addresses.
x	MIDE = 0	Reserved

Bit 29 – SID11 12th bit of Standard Identifier Filter bit

Bits 28:11 – EID[17:0] Extended Identifier Filter bits
In DeviceNet™ mode, these are the filter bits for the first two data bytes.

Bits 10:0 – SID[10:0] Standard Identifier Filter bits

14.4.32. CAN Mask x Register(1), (2), (3)

Name: C1MASKx
Offset: 0x26C4, 0x26CC, 0x26D4, 0x26DC, 0x26E4, 0x26EC, 0x26F4, 0x26FC, 0x2704, 0x270C, 0x2714, 0x271C, 0x2724, 0x272C, 0x2734, 0x273C

Notes:

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - CxMASKyT: Accesses the top byte MASKy[31:24].
   - CxMASKyU: Accesses the upper byte MASKy[23:16].
   - CxMASKy: Accesses the byte MASKy[31:0].
2. Each Mask is associated with a filter.
3. [x] denotes Filter number, from 0 to 15.

Bit	31	30	29	28	27	26	25	24
		MIDE	MSID11	MEID[17:13]				
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MEID[12:5]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MEID[4:0]					MSID[10:8]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 30 – MIDE Identifier Receive Mode bit

Value	Description
1	Matches only message types (standard or extended address) that correspond to the EXIDE bit of in the filter.
0	Matches either standard or extended address message if filters match (i.e., if (Filter SID) = (Message SID) or if (Filter SID/EID) = (Message SID/EID)).

Bit 29 – MSID11 12th bit of Standard Identifier Mask bit

Bits 28:11 – MEID[17:0] Extended Identifier Mask bits
In DeviceNet™ mode, these are the mask bits for the first two data bytes.

Bits 10:0 – MSID[10:0] Standard Identifier Mask bits

14.5. Modes of Operation

The CAN FD Protocol Module has eight modes of operations:

- Configuration mode
- Normal CAN FD mode: Supports mixing of CAN FD and CAN 2.0 messages.

- Normal CAN 2.0 mode: Will generate error frames while receiving CAN FD messages. The FDF bit is forced to zero and only CAN 2.0 frames are sent, even if the FDF bit is set in the transmit message object.
- Disable mode
- Listen Only mode
- Restricted Operation mode
- Internal Loopback mode
- External Loopback mode

The modes of operations can be grouped into four main groups: Configuration, Normal, Disable/ Sleep and Debug (see Figure 14-9).

14.5.1. Mode Change

Figure 14-9 illustrates the possible mode transitions. New modes of operation are requested by writing to the REQOP[2:0] (CxCON[26:24]) bits. The modes of operations do not change immediately. The modes will only change when the bus is Idle.

The current operating mode is indicated in the OPMOD[2:0] (CxCON[23:21]) bits. The application can enable an interrupt on an OPMODx change or poll the OPMODx bits.

14.5.1.1. Changing Between Normal Modes

Directly changing between Normal modes is not allowed. The Configuration mode must be selected before a new Normal mode can be selected.

14.5.1.2. Changing Between Debug Modes

Directly changing between Debug modes is not allowed. The Configuration mode must be selected before a new Debug mode can be selected.

14.5.1.3. Exiting Normal Mode

The device will transition to Configuration or Sleep mode only after the current message is transmitted.

14.5.1.4. Entering and Exiting Disable Mode

The CAN FD Protocol enters Disable mode after a Disable mode request. The device exits Disable mode after a mode request.

If WAKIE is set, a dominant edge on CxRX will generate an interrupt. The CPU has to enable the CAN module by requesting a Normal mode.

14.5.1.5. Entering and Exiting Sleep Mode

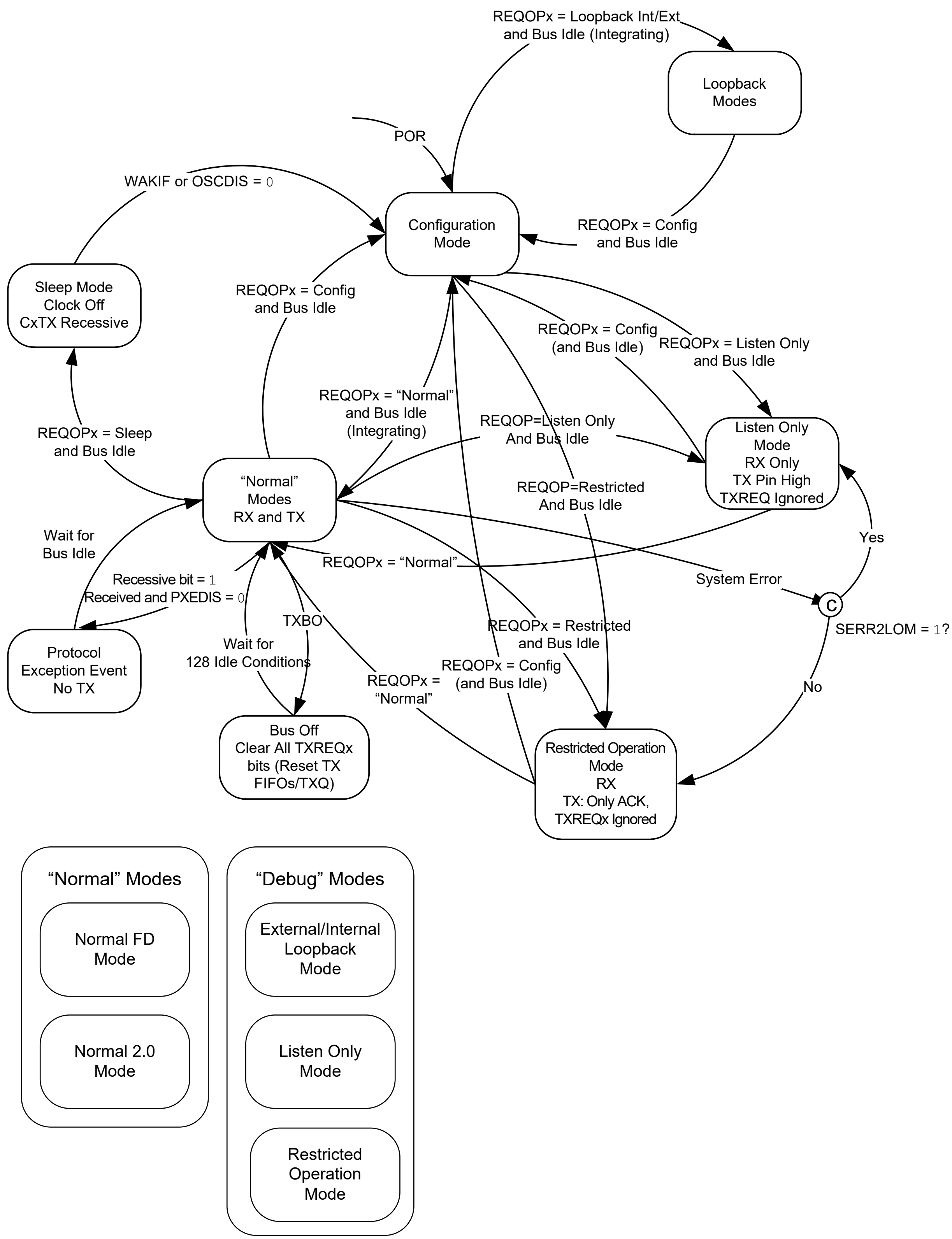
The module enters Sleep mode after a Sleep Mode request. The device exits Sleep mode due to a dominant edge is on RXCAN or by enabling the oscillator. The module will transition automatically to Configuration mode.

14.5.1.6. Bus Integrating Mode

The CAN FD Protocol integrates to the bus, according to the ISO11898-1:2015 specifications (11 consecutive recessive bits), under the following conditions:

- Change from Configuration mode to one of the Normal modes or Debug modes.
- Change from Disable mode to one of the Normal modes.

Figure 14-9. CAN FD Modes of Operation



14.5.2. Configuration Mode

After Reset, the CAN FD Protocol is in Configuration mode. The error counters are cleared and all registers contain the Reset values.

The CAN FD Protocol has to be initialized before activation. This is only possible when the module is in Configuration mode, OPMOD[2:0] = `100`. The Configuration mode is requested by setting REQOP[2:0] = `100`.

The CAN FD Protocol will protect the user from accidentally violating the CAN protocol through programming errors. The following registers and bit fields can only be programmed during Configuration mode:

- CxCON: WAKFIL, CLKSEL, PXEDIS, ISOCRCEN, TXQEN, STEF, SERR2LOM, ESIGM, RTXAT
- CxNBTCFG, CxDBTCFG, CxTDC
- CxTXQCON: PLSIZE[2:0], FSIZE[4:0], TXEN, RXTSEN
- CxFIFOCONx: PLSIZE[2:0], FSIZE[4:0], TEFTSEN
- CxTEFCON: FSIZE[4:0]
- CxFIFOBA

The CAN FD Protocol is not allowed to enter Configuration mode during transmission or reception to prevent the module from causing errors on the CAN bus. The following registers are reset when exiting Configuration mode:

- CxTREC
- CxBDIAG0
- CxBDIAG1

In Configuration mode, FRESET is set in the CxFIFOCONx, CxTXQCON and CxTEFCON registers, and all FIFOs and the TXQ are reset.

14.5.3. Normal Modes

14.5.3.1. Normal CAN FD Mode

Once the device is configured, Normal Operation mode can be requested by setting REQOP[2:0] = `000`.

In this mode, the device will be on the CAN bus. It can transmit and receive messages in CAN FD mode, bit rate switching can be enabled and up to 64 data bytes can be transmitted and received.

14.5.3.2. Normal CAN 2.0 Mode

The Normal CAN 2.0 Operation mode can be requested by setting REQOP[2:0] = `110`.

In this mode, the device will be on the CAN bus. This is a the Classic CAN 2.0 mode. The module will not receive CAN FD frames. It might send error frames if CAN FD frames are detected on the bus. The FDF, BRS and ESI bits in the TX objects will be ignored and transmitted as ‘`0`’.

14.5.4. Disable Mode

Disable mode is similar to Configuration mode, except the error counters are not reset. Disable mode is requested by setting REQOP[2:0] = `001`.

The CAN module will not be allowed to enter Disable mode while a transmission or reception is taking place to prevent causing errors on the CAN bus. The module will enter Disable mode when the current message completes.

The OPMODx bits indicate whether the module successfully entered Disable mode. The application software should use this bit field as a handshake indication for the Disable mode request.

The CxTX pin will stay in the recessive state while the module is in Disable mode to prevent inadvertent CAN bus errors.

14.5.5. Debug Modes

14.5.5.1. Listen Only Mode

Listen Only mode is a variant of normal CAN FD Operation mode or CAN 2.0 mode. If the Listen Only mode is activated, the module on the CAN bus is passive. It will receive messages, but it will not transmit any bits. TXREQx bits will be ignored. No error flags or acknowledge signals are sent. The error counters are deactivated in this state. The Listen Only mode can be used for detecting the baud rate on the CAN bus. It is necessary that there are at least two further nodes that communicate with each other. The baud rate can be detected empirically by testing different values until a message is received successfully. This mode is also useful for monitoring the CAN bus without influencing it.

14.5.5.2. Restricted Operation Mode

In Restricted Operation mode, the node is able to receive data and remote frames and acknowledge valid frames, but it does not send data frames, remote frames, error frames or overload frames. In case of an error or overload condition, it does not send dominant bits; instead, it waits for the bus to enter the Idle condition to resynchronize itself to the CAN communication. The error counters are not incremented.

14.5.5.3. Loopback Mode

Loopback mode is a variant of Normal CAN FD Operation mode or CAN 2.0 mode. This mode will allow internal transmission of messages from the transmit FIFOs to the receive FIFOs. The module does not require an external acknowledge from the bus. No messages can be received from the bus, because the CxRX pin is disconnected.

14.5.5.3.1. Internal Loopback Mode

The transmit signal is internally connected to receive, and the CxTX pin is driven high.

14.5.5.3.2. External Loopback Mode

The transmit signal is internally connected to receive, and transmitted messages can be monitored on the CxTX pin.

14.5.6. Low-Power Modes

14.5.6.1. Sleep Mode

In the CAN module, special conditions need to be met for Sleep mode. The module must first be switched to Disable mode by setting REQOPx = `001`. When OPMODx = `001`, indicating Disable mode has been achieved, the CAN FD Protocol Module enters Sleep mode after a Sleep mode request. In Sleep mode, the register contents do not change, so the OPMODx bits do not change. At the end of Sleep, the module will continue in the mode specified by the OPMODx bits previous to Sleep mode (which should be Disable mode, OPMODx = `001`). If the user executes a `SLEEP` instruction without switching to Disable mode, the module assumes a clock is available to read/write from RAM. Since the system clock input is not available in Sleep mode, the CAN module cannot run as it requires a system clock to transmit or receive. Also, the FIFO is in system RAM, which has no clock in Sleep mode.

Recommended steps:

1. Write the REQOP[2:0] bits to '`001`'; the module will enter Disable mode.
2. Poll the OPMOD[2:0] bits to verify whether they are '`001`', which indicates that the module has successfully entered Disable mode.
3. Disable the clock source of the CAN module.
4. Execute the `SLEEP` instruction.

14.5.6.2. Idle Mode

The system can be set to run in a Low-Power mode, called Idle mode. When the device is in Idle mode, the CPU is disabled and only select peripherals are active.

Based on the configuration of the CAN SIDL bit, the module can stop or continue in Idle mode:

- If SIDL = `0`, the module continues operation in Idle mode. If the module generates an interrupt while in Idle mode, the interrupt may generate a wake-up event.
- If SIDL = `1`, the module stops when the device is in Idle mode. The module performs the same procedures when stopped in Idle mode as it does in Disable mode and the same requirements apply.

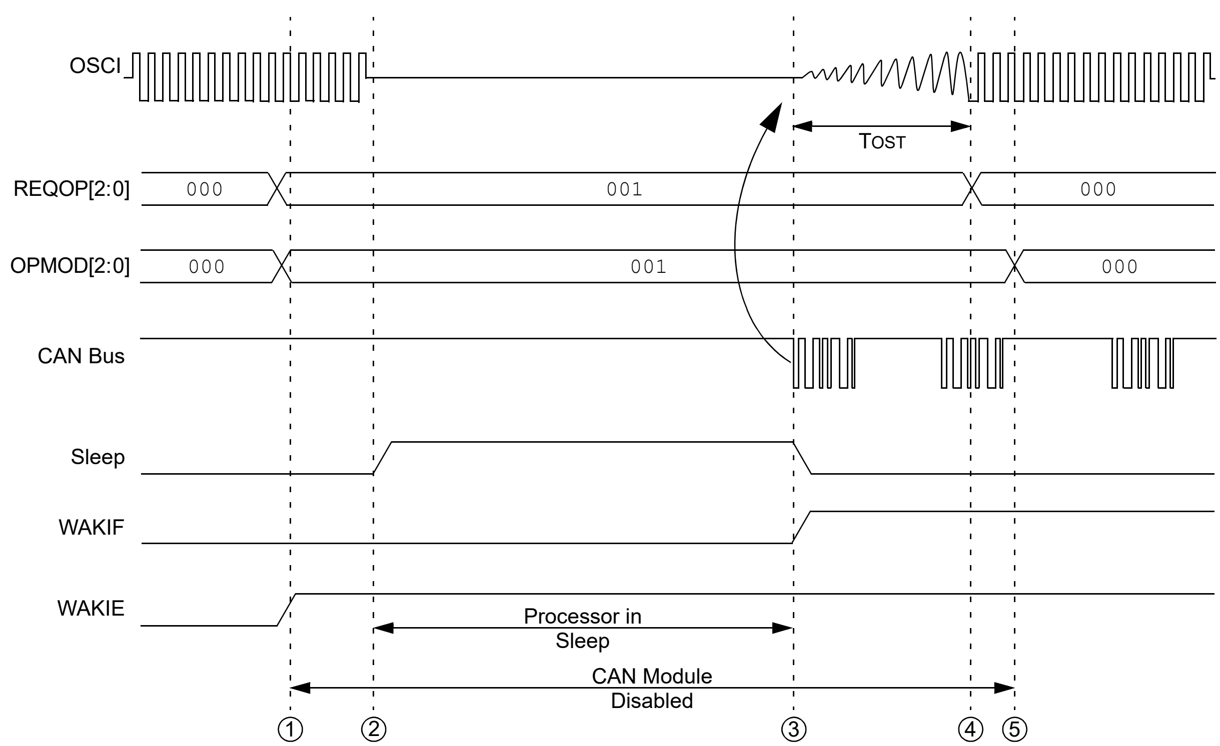
The user should ensure that the module is not active when the CPU transitions to Idle mode with SIDL = `1`. To protect the CAN bus system from fatal consequences due to violation of this rule, the module will drive the TX pin into the recessive state while stopped in Idle mode.

If the CAN SIDL bit is set, the recommended procedure is to bring the module into Disable mode before the device is placed in Idle mode.

14.5.6.3. Wake Up from Sleep

Figure 14-10 depicts how the CAN module will execute the `SLEEP` instruction and how the module wakes up on bus activity. Upon a wake-up from Sleep mode, the WAKIF flag is set.

Figure 14-10. Processor Sleep and CAN Bus Wake-up Interrupt



① – Processor requests and receives Module Disable mode. Wake-up interrupt is enabled.

② – Processor executes `SLEEP` (`PWRSAV #0`) instruction.

③ – SOF of message wakes up processor. Oscillator start time begins. CAN message is lost. WAKIF bit is set.

④ – Processor completes oscillator start time. Processor resumes program or interrupt, based on GIE bits. Processor requests Normal Operating mode. Module waits for 11 recessive bits before accepting CAN bus activity. CAN message is lost.

⑤ – Module detects 11 recessive bits. Module will begin to receive messages and transmits any pending messages.

The module will monitor the CAN receive line for activity while the module is sleeping. The device will generate a wake-up interrupt on the falling edges of CxRX if WAKIE is enabled.

The device will exit Sleep mode after a new mode request or a negative edge on CxRX.

The module will be in Sleep mode if either of the following is true:

- The system is in Sleep mode following Disable mode.
- The system is in Idle mode with SIDL = `1`.

Note: If the module is in Sleep mode, the module generates an interrupt if the WAKIE bit (CxINT[30]) is set and bus activity is detected. Due to delays in starting up the oscillator and CPU, the message activity that caused the wake-up will be lost.
The module can be programmed to apply a low-pass filter function to the CAN receive input line while in Disable, Sleep or Idle mode. This feature can be used to protect the module from wake-up due to short glitches on the CAN bus lines. The WAKFIL bit (CxCON[8]) enables or disables the filter while the module is in Sleep.

14.6. Configuration

14.6.1. Clock Configuration

The sample point of all nodes in a CAN FD network should be at the same position. Hence, it is recommended to use the same clock frequency and bit time settings for all nodes. Therefore, a CAN clock (F_{CAN}) of 80 MHz, 40 MHz or 20 MHz is recommended.

CLKSEL = `0` provides the clock from the CAN clock generator.

14.6.2. CAN Configuration

The CxCON registers contain several bits that can only be configured in Configuration mode.

14.6.2.1. ISO CRC Enable

The module supports ISO CRC (according to ISO11898-1:2015) and non-ISO CRC (see ISO vs. Non-ISO CRC). ISO CRC is enabled by setting the ISOCRCEN bit.

14.6.2.2. Protocol Exception Disable

The negative edge between the FDF bit and the reserved bit in CAN FD frames is important for the calculation of the transceiver delay and for hard synchronization. Therefore, if the reserved bit following the FDF bit is detected recessive, the CAN FD Protocol module will treat this as a form error. This is called Protocol Exception Event Detection Disabled, and it is configured by setting the PXEDIS bit.

The Protocol Exception Event Detection Disabled can be enabled by clearing the PXEDIS bit. As a reaction to the protocol exception event, the error counters are not changed, hard synchronization is enabled, the module sends recessive bits and enters the bus integration state.

14.6.2.3. Wake-Up Filter – WFT[1:0]

The WAKFIL bit is used to enable/disable the low-pass filter on the CxRX pin. The filter is only active during Sleep mode. The WFTx bits allow the configuration of different filter times.

14.6.2.4. Restriction of Transmission Attempts

ISO11898-1:2015 requires that frames that lost arbitration, and are not Acknowledged or are destroyed by errors, are automatically retransmitted. Optionally, the number of retransmission attempts can be limited.

When the RTXAT bit is set, retransmission attempts can be limited using the TXAT[1:0] bits in the FIFO Control registers. If the RTXAT bit is clear, then the TXATx bits in the FIFO Control register are ignored and the retransmission attempts are unlimited.

14.6.2.5. Error State Indicator (ESI) in Gateway Mode

Normally, the ESI bit in a transmitted message reflects the error status of the CAN FD Protocol Module. ESI is transmitted recessive when the module is error passive. In case the module is used in a gateway application, there will be situations where the ESI bit in the message should be transmitted recessive, even though the gateway module is error active. This can be configured by setting the ESIGM bit.

14.6.2.6. Mode Selection in Case of System Error

The SERR2LOM bit selects which mode the module will transition to in case of a system error. The module can either transition to Restricted Operation mode or Listen Only mode.

14.6.2.7. Reserving Message Memory for TXQ and TEF

Setting the TXQEN bit will reserve RAM for the TXQ. If the TXQEN bit is cleared, then the TXQ cannot be used.

Setting the STEF bit will reserve RAM for the TEF, and all transmitted messages will be stored in the TEF.

14.6.3. CAN FD Bit Time Configuration

In order to achieve higher bandwidth, bits in a CAN FD frame are transmitted with two different bit rates:

- Nominal Bit Rate (NBR): Used during arbitration until the sample point of the BRS bit and the sample point of the CRC delimiter reach the End of Frame (EOF).
- Data Bit Rate (DBR): Used during the data and CRC field.

NBR is limited by the propagation delay of the CAN network (see Propagation Delay). In the data phase, only one transmitter remains; therefore, the bit rate can be increased. The transmitting node always compares the intended transmitted bits with the actual bits on the CAN bus. The propagation delay in the data phase can be longer than the bit time. In this case, the data bits are sampled at a Secondary Sample Point (SSP) (see Transmitter Delay Compensation (TDC)).

NBR is the number of bits per second during the arbitration phase. It is the inverse of the Nominal Bit Time (NBT) (see Equation 14-1).

Equation 14-1. Nominal Bit Rate/Time

$$NBR = \frac{1}{NBT}$$

DBR is the number of bits per second during the data phase. It is the inverse of the Data Bit Time (DBT) (see Equation 14-2).

Equation 14-2. Data Bit Rate/Time

$$DBR = \frac{1}{DBT}$$

The Baud Rate Prescaler (BRP) is used to divide the Fcan. The divided Fcan is used to generate the bit times.

There are two prescalers: NBRP for the Nominal Bit Rate Prescaler and DBRP for the Data Bit Rate Prescaler. The Time Quanta (NTQ and DTQ) are selected as shown in Equation 14-3 and Equation 14-4.

Equation 14-3. Nominal Time Quanta

$$NTQ = NBRP \times T_{\mathrm{CAN}} = \frac{NBRP}{F_{\mathrm{CAN}}}$$

Equation 14-4. Data Time Quanta

$$DTQ = DBRP \times T_{\mathrm{CAN}} = \frac{DBRP}{F_{\mathrm{CAN}}}$$

CAN bit times have four segments (see Figure 14-11).

Synchronization Segment (SYNC) – Synchronizes the different nodes connected on the CAN bus. A bit edge is expected to be within this segment. The Synchronization Segment is always 1 Tq.

Propagation Segment (PRSEG) – Compensates for the propagation delay on the bus. PRSEG has to be longer than the maximum propagation delay.

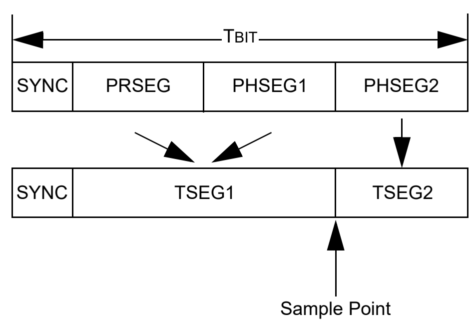
Phase Segment 1 (PHSEG1) – Compensates for errors that may occur due to phase shifts in the edges. The time segment may be automatically lengthened during resynchronization to compensate for the phase shift.

Phase Segment 2 (PHSEG2) – Compensates for errors that may occur due to phase shifts in the edges. The time segment may be automatically shortened during resynchronization to compensate for the phase shift.

In the Bit Time registers, PRSEG and PHSEG1 are combined to create TSEG1. PHSEG2 is called TSEG2. Each segment has multiple Time Quanta (Tq). The sample point lies between TSEG1 and TSEG2.

Table 14-4 and Table 14-5 show the ranges for the bit time configuration parameters.

Figure 14-11. Partition of Bit Time



The total number of Tq in a bit time is programmable and can be calculated using Equation 14-5 and Equation 14-6.

Equation 14-5. Number of NTQ in a NBT

$$\frac{NBT}{NTQ} = NSYNC + NTSEG1 + NTSEG2$$

Equation 14-6. Number of DTQ in a DBT

$$\frac{DBT}{DTQ} = DSYNC + DTSEG1 + DTSEG2$$

Table 14-4. Nominal Bit Rate Configuration Ranges

Segment	Minimum	Maximum
NSYNC	1	1
NTSEG1	2	256
NTSEG2	1	128
NSJW	1	128
NTQ per Bit	4	385

Table 14-5. Data Bit Rate Configuration Ranges

Segment	Minimum	Maximum
DSYNC	1	1
DTSEG1	1	32
DTSEG2	1	16
DSJW	1	16
DTQ per Bit	3	49

14.6.3.1. Sample Point

The sample point is the point in the bit time at which the logic level of the bit is read and interpreted. The sample point in percent can be calculated using Equation 14-7 and Equation 14-8.

Equation 14-7. Nominal Sample Point (%)

$$NSP = \frac{1 + NTSEG1}{\frac{NBT}{NTQ}} \times 100$$

Equation 14-8. Data Sample Point (%)

$$DSP = \frac{1 + DTSEG1}{\frac{DBT}{DTQ}} \times 100$$

14.6.3.2. Propagation Delay

Figure 14-12 illustrates the propagation delay between two CAN nodes on the bus, assuming Node A is transmitting a CAN message. The transmitted bit will propagate from the transmitting CAN Node A through the transmitting CAN transceiver, over the CAN bus, through the receiving CAN transceiver, into the receiving CAN Node B.

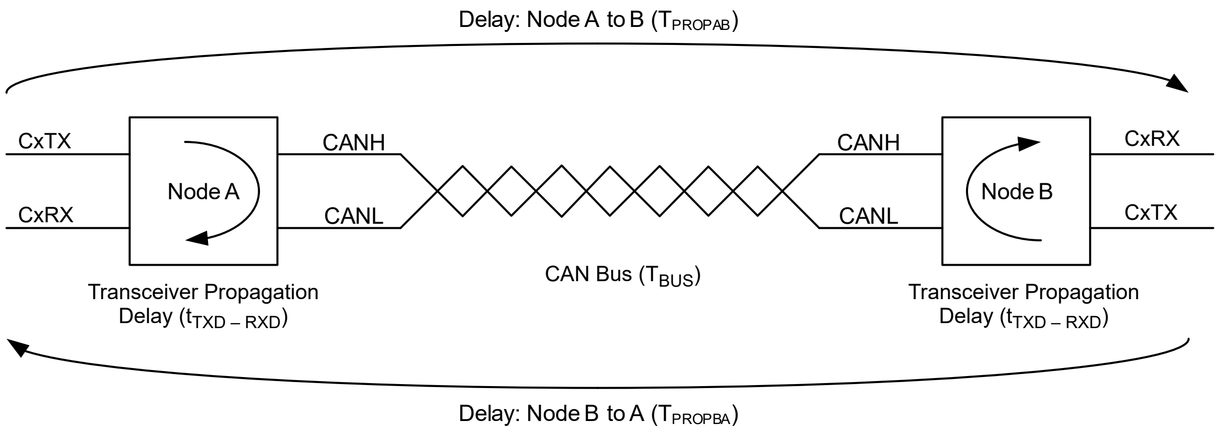
During the arbitration phase of a CAN message, the transmitter samples the CAN bus and checks if the transmitted bit matches the received bit. The transmitting node has to place the sample point after the maximum propagation delay.

Equation 14-9 describes the maximum propagation delay; where $t_{TXD-RXD}$ is the propagation delay of the transceiver, a maximum of 255 ns according to ISO11898-1:2015; T_{BUS} is the delay on the CAN bus, which is approximately 5 ns/m. The factor *2* comes from the worst-case when Node B starts transmitting exactly when the bit from Node A arrives.

Equation 14-9. Maximum Propagation Delay

$$T_{PROP} = 2 \times (t_{TXD-RXD} + T_{BUS})$$

Figure 14-12. Propagation Delay



$$T_{PROP} = T_{PROPAB} + T_{PROPBA} = 2 \times (t_{TXD-RXD} + T_{BUS})$$

14.6.3.3. Transmitter Delay Compensation (TDC)

During the data phase of a CAN FD transmission, only one node is transmitting; the others are receiving. Therefore, the propagation delay does not limit the maximum data rate.

When transmitting via pin CxTX, the CAN FD Protocol module receives the transmitted data from its local CAN transceiver via pin CxRX. The received data are delayed by the CAN transceiver's loop delay. In case this delay is greater than 1 + DTSEG1, a bit error would be detected.

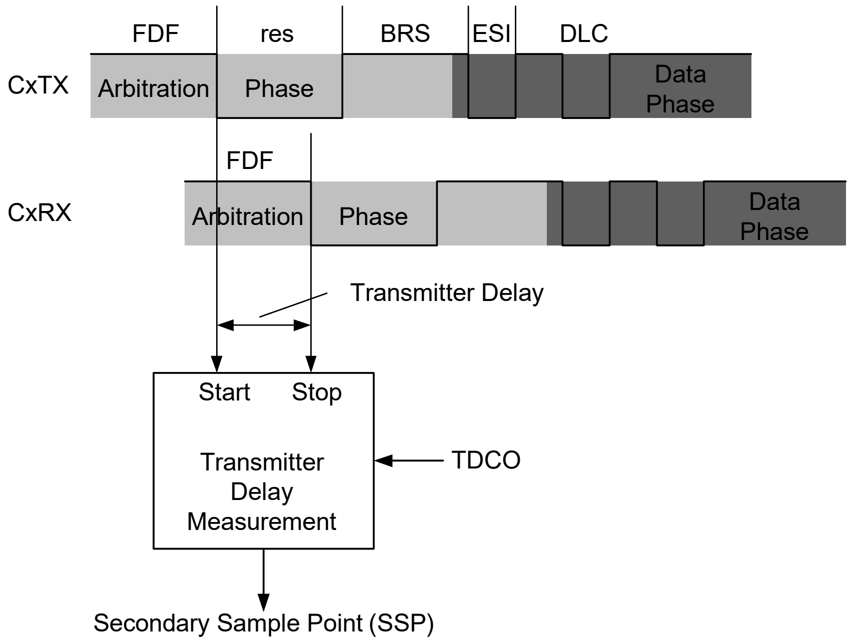
In order to enable a data phase bit time that is shorter than the transceiver loop delay, the Transmitter Delay Compensation (TDC) is implemented. Instead of sampling after DTSEG1, a Secondary Sample Point (SSP) is calculated and used for sampling during the data phase of a CAN FD message.

Figure 14-13 illustrates how the transceiver loop delay is measured, and Equation 14-10 shows how the SSP is calculated.

Equation 14-10. Secondary Sample Point

SSP = TDCV[5:0] + TDCO[6:0]

Figure 14-13. Measurement of Transceiver Delay (TDCV)



14.6.3.4. Synchronization

To compensate for phase shifts between the oscillator frequencies of the nodes on the CAN bus, each CAN controller must be able to synchronize to the relevant edge of the incoming signal.

The CAN controller expects an edge in the received signal to occur within the SYNC segment. Only recessive-to-dominant edges are used for synchronization.

There are two mechanisms used for synchronization:

- **Hard Synchronization** – Forces the edge that has occurred to lie within the SYNC segment. The bit time counter is restarted with SYNC.
- **Resynchronization** – If the edge falls outside the SYNC segment, PHSEG1 or PHSEG2 will be adjusted.

For a more detailed description of the CAN synchronization, please refer to ISO11898-1:2015.

14.6.3.5. Synchronization Jump Width

The Synchronization Jump Width (SJW) is the maximum amount that PHSEG1 and PHSEG2 can be adjusted during resynchronization. SJW is programmable (see Table 14-4 and Table 14-5).

14.6.3.6. Oscillator Tolerance

The oscillator tolerance, *df*, around the nominal frequency of the oscillator, *fnom*, is defined in Equation 14-11.

Equation 14-12 through Equation 14-16 describe the conditions for the maximum tolerance of the oscillator.

Equation 14-11. Oscillator Tolerance

$$(1 - df) \times \text{fnom} \le F_{\text{CAN}} \le (1 + df) \times fnom$$

Equation 14-12. Condition 1

$$df \leq \frac{NSJW}{2 \times 10 \times \frac{NBT}{NTQ}}$$

Equation 14-13. Condition 2

$$df \leq \frac{min(NPHSEG1, NPHSEG2)}{2 \times \left(13 \times \frac{NBT}{NTQ} - NPHSEG2\right)}$$

Equation 14-14. Condition 3

$$df \leq \frac{DSJW}{2 \times 10 \times \frac{DBT}{DTQ}}$$

Equation 14-15. Condition 4

$$df \leq \frac{min(NPHSEG1, NPHSEG2)}{2 \times \left(\left(6 \times \frac{DBT}{DTQ} - DPHSEG2\right) \times \frac{DBRP}{NBRP} + 7 \times \frac{NBT}{NTQ}\right)}$$

Equation 14-16. Condition 5

$$df \leq \frac{DSJW - max\left(0, \left(\frac{NBRP}{DBRP} - 1\right)\right)}{2 \times \left(\left(2 \times \frac{NBT}{NTQ} \times HNSEGP2\right) \times \frac{NBRP}{DBRP} + DPHSEG2 + 4 \times \frac{DBT}{DTQ}\right)}$$

14.6.3.7. Recommendations for Bit Time Configuration

The following recommendations should be considered when configuring the bit time:

- Select the highest available CAN clock frequency:
  - Short T_Q leads to high resolution to select the sample point.
  - Use 20 MHz, 40 MHz or 80 MHz for F_{CAN}.
- Select the lowest NBRP and DBRP:
  - Low BRP leads to short T_Q.
  - NSYNC and DSYNC will be short and reduce the quantization error.
  - The receiving node can synchronize more accurately to the transmitting node.
- Set NBRP equal to DBRP:
  - Identical T_Q in both phases prevent quantization errors during Bit Rate Switching.
- Use the same Nominal Sample Point (NSP) and Data Sample Point (DSP) in all nodes on the CAN FD network:
  - Different sample points in the different nodes lead to different lengths of the BRS and CRC delimiter bits and introduce phase errors when switching the bit rate.
  - NSP need not be equal to the DSP.
  - The SSP can be different in differing CAN FD nodes.
- Select the largest possible NSJW and DSJW:
  - Maximizes the oscillator tolerance.
  - Allows the receiving nodes to quickly resynchronize to the transmitting nodes.
- Enable automatic TDC for DBR of 1 Mbps and higher:
  - Automatic TDC measurement compensates for transmitter delay variations.

14.6.3.8. Bit Time Configuration Example

The following tables illustrate the configuration of the CAN FD Bit Time registers, assuming there is a CAN FD network in an automobile with the following parameters:

- 500 kbps NBR – Sample Point at 80%
- 2 Mbps DBR – Sample Point at 80%
- 40 Meters – Minimum Bus Length

Table 14-6 and Table 14-7 illustrate how the bit time parameters are calculated. Since the parameters depend on multiple constraints and equations, and are calculated using an iterative process, it is recommended to enter the equations in a spreadsheet.

Table 14-8 translates the calculated values into register values. It is recommended to let the CAN FD Protocol Module measure the Transmitter Delay Compensation Value (TDCV). This is accomplished by setting TDCMOD[1:0] (CxTDC[1:0]) = `10` (Automatic mode). In order to set the SSP to 80%, TDCO[6:0] are set to (DBRP * DTSEG1).

Table 14-6. Step-by-Step Nominal Bit Rate Configuration

Parameter	Constraint	Value	Unit	Equations and Comments
NBT	NBT ≥ 1 µs	2	µs	Equation 14-1
F_{CAN}	F_{CAN} ≤ 80 MHz	80	MHz	CAN clock frequency= 80 MHz
NBRP	1 to 256	1	—	Select smallest possible BRP value to maximize resolution.
NTQ	NBT, F_{CAN}	12.5	ns	Equation 14-3
NBT/NTQ	4 to 385	160	—	Equation 14-5
NSYNC	Fixed	1	NTQ	Defined in ISO11898-1:2015.
NPRSEG	NPRSEG > T_{PROP}	95	NTQ	Equation 14-9: T_{PROP} = 910 ns, minimum NPRSEG = T_{PROP}/NTQ = 72.8 NTQ. Selecting 95 will allow up to a 60m bus length.
NTSEG1	2 to 256 NTQ	127	NTQ	Equation 14-7. Select NTSEG1 to achieve 80% NSP.
NTSEG2	1 to 128 NTQ	32	NTQ	There are 32 NTQ left to reach NBT/NTQ = 160.
NSJW	1 to 128 NTQ; SJW ≤ min (NPHSEG1, NPHSEG2)	32	NTQ	Maximizing NSJW lessens the requirement for the oscillator tolerance.

Table 14-7. Step-by-Step Data Bit Rate Configuration

Parameter	Constraint	Value	Unit	Equations and Comments
DBT	DBT ≥ 125 ns	500	ns	Equation 14-2
DBRP	1 to 256	1	—	Selecting the same prescaler as for NBT ensures that the T_Q resolution does not change during the Bit Rate Switching.
DTQ	DBT, F_{CAN}	12.5	ns	Equation 14-4
DBT/DTQ	3 to 49	40	—	Equation 14-6
DSYNC	Fixed	1	DTQ	Defined in ISO11898-1:2015.
DTSEG1	1 to 32 DTQ	31	DTQ	Equation 14-7. Select DTSEG1 to achieve 80% DSP.
DTSEG2	1 to 16 DTQ	8	DTQ	There are 8 DTQ left to reach DBT/DTQ = 40.
DSJW	1 to 16 DTQ; SJW ≤ min (DPHSEG1, DPHSEG2)	8	DTQ	Maximizing DSJW lessens the requirement for the oscillator tolerance.
Oscillator Tolerance Conditions 1-5	Minimum of Conditions 1-5	0.78	%	Equation 14-11 through Equation 14-16

Table 14-8. Bit Time Register Initialization (500k/2M)

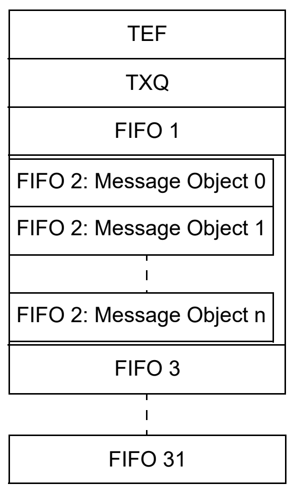
CxNBTCFGL/H	Value	CxDBTCFGL/H	Value	CxTDCL/H	Value
BRP[7:0]	0	BRP[7:0]	0	TDCMOD[1:0]	2
TSEG1[7:0]	126	TSEG1[4:0]	30	TDCO[6:0]	31
TSEG2[6:0]	31	TSEG2[3:0]	7	TDCV[5:0]	0
SJW[6:0]	31	SJW[3:0]	7	—	—

14.6.4. Message Memory Configuration

The message objects of the TEF, TXQ and transmit/receive FIFOs are located in RAM (see Figure 14-14). The application must configure the number of message objects in a FIFO between Message Object 0 and Message Object 31. Additionally, the application must configure the payload size of the message objects in each FIFO. This configuration determines where message objects are located in RAM. The RAM allocation can only be configured in Configuration mode. The start of the Message Memory is defined by CxFIFOBA register, which is word aligned.

In order to optimize RAM usage, the application should start configuring the RAM with the TEF, followed by the TXQ and continue with FIFO 1, FIFO 2, FIFO 3 and so on. In case a user application requires TEF, TXQ and 16 additional FIFOs, it should configure TEF and TXQ, followed by FIFO 1 through FIFO 16. It is not necessary to configure the unused FIFOs 17 through 31.

Figure 14-14. Message Memory Organization



14.6.4.1. Transmit Event FIFO Configuration

In order to reserve space in RAM for the TEF, the STEF bit (CxCON[19]) has to be set. The number of message objects in the TEF is configured using the FSIZE[4:0] bits (CxTEFCON[28:24]). Transmitted messages can be timestamped by setting the TEFTSEN bit (CxTEFCON[5]).

14.6.4.2. Transmit Queue Configuration

In order to reserve space in RAM for the TXQ, the TXQEN bit (CxCON[20]) has to be set. The number of message objects in the TXQ is configured using the FSIZE[4:0] bits (CxTXQCON[28:24]). All objects in the TXQ use the same payload size (number of data bytes), which is configured using the PLSIZE[2:0] bits (CxTXQCON[31:29]).

14.6.4.3. Transmit FIFO Configuration

FIFO 1 through FIFO 31 can be configured as transmit FIFOs by setting TXEN in the CxFIFOCONx register. The number of message objects in each transmit FIFO is configured using the FSIZE[4:0] bits (CxFIFOCONx[28:24]). All objects in one transmit FIFO use the same payload size (number of data bytes), which is determined by the PLSIZE[2:0] bits (CxFIFOCONx[31:29]).

14.6.4.4. Receive FIFO Configuration

FIFO 1 through FIFO 31 can be configured as receive FIFOs by clearing TXEN in the CxFIFOCONx register. The number of message objects in each receive FIFO is configured using the FSIZE[4:0] bits (CxFIFOCONx[28:24]). All objects in one receive FIFO use the same payload size (number of data bytes), which is determined by the PLSIZE[2:0] bits (CxFIFOCONx[31:29]). Received messages can be time-stamped by setting the RXTSEN bit (CxFIFOCONx[5]).

14.6.4.5. Calculation of Required Message Memory

The size of required RAM depends on the configuration of each FIFO. Equation 14-17 through Equation 14-19 specify the sizes of the TEF, TXQ and the FIFOs in bytes. The TEF or TXQ is not used if their size is zero.

Since the size of the integrated RAM is limited, the user must check that the memory configuration fits into RAM. Equation 14-20 can be used to calculate the total RAM usage in bytes.

The size of the TEF objects depends on the enabling of timestamping. If TEFTSEN is set, then tefts = 4, else tefts = 0.

The *PayLoad(i)* is defined in data bytes.

The size of a message object of an RX FIFO varies dependent on the enabling of timestamping. If RXTSEN = `1` and TXEN = `0` for *FIFO(i)*, then *rxts(i)* = `4`, else *rxts(i)* = `0`.

N is defined as the number of FIFOs used in addition to the TEF and the TXQ.

Equation 14-17. Size of TEF

$$S_{TEF} = N_{Elements}(TEF) \times (tefts+8)$$

Equation 14-18. Size of TXQ

$$S_{TXQ} = N_{Elements}(TXQ) \times (8 + PayLoad(TXQ))$$

Equation 14-19. Size of FIFOs

$$S_{FIFO}(i) = N_{Elements}(i) \times (rxts(i) + 8 + PayLoad(i))$$

Equation 14-20. Total RAM Usage

$$S_{RAM} = \left(S_{TEF} + S_{TXQ} + \sum_{i=1}^{N} S_{FIFO}(i)\right)$$

For example:

- If TEF is 4 messages deep ($N_{Elements}$ (TEF) = 4) and TEFTSEN is clear, then the size of TEF = S_{TEF} = 4 x (0 + 8) = 32 bytes.
- If $N_{Elements}$ (TXQ) = 1, PayLoad (TXQ) = 12, then the size of TXQ = S_{TXQ} = 1 x (8 + 12) = 20 bytes.
- If $N_{Elements}$ (FIFO) = 3, PayLoad (FIFO) = 8, then the size of FIFO = S_{FIFO} = 3 x (8 + 8) = 48 bytes.

Therefore, SRAM = S_{TEF} + S_{TXQ} + S_{FIFO} = 32 + 20 + 48 = 100 bytes.

14.6.4.6. Calculation of Start Address

Since the payload size of the FIFOs can be configured individually, the start address of an individual object depends on the configuration of all previous objects. The application can read back the start addresses of each message object to double-check whether they were correctly configured.

The TEF starts at the beginning of the message memory.

Equation 14-21. Start Address of TEF

$$A_{TEF} = Base\ Address = CxFIFOBA$$

The TXQ starts after the Transmit Event FIFO.

Equation 14-22. Start Address of TXQ

$$A_{TXQ} = CxFIFOBA + S_{TEF}$$

The message FIFO object starts after the Transmit Queue.

Equation 14-23. Start Address of Message FIFO

$$A_{FIFO(1)} = CxFIFOBA + S_{TEF} + S_{TXQ}$$

If CxCON.STEF = `0`, then TEF does not exist in RAM and Stef is zero. In this case, if TXQ is available, the TXQ starts at the base address defined by CxFIFOBA.

If CxCON.TXQEN = `0`, then TXQ does not exist in RAM and Stxq is zero.

The start of the nth FIFO can be calculated as follows:

Equation 14-24. Start Address of Nth FIFO

$$A_{\mathrm{FIFO}(n)} = \left(A_{FIFO(1)} + \sum_{i=1}^{n-1} S_{FIFO}(i)\right)$$

14.7. Message Transmission

The application has to configure the FIFO or TXQ before it can be used for transmission (see Transmit FIFO Configuration and Transmit Queue Configuration).

14.7.1. Transmit Message Object

Table 14-9 specifies the transmit message object used by the TXQ and the transmit FIFOs. The transmit objects contain the message ID, control bits and payload.

- **SID:** Standard Identifier or Base Identifier
- **EID:** Extended Identifier
- **DLC:** Data Length Code; specifies the number of data bytes to transmit (see DLC Encoding).
- **IDE:** Identifier Extension; clearing this bit will transmit a base frame, setting this bit will transmit an extended frame.
- **RTR:** Remote Transmit Request; this bit is only specified in CAN 2.0 frames. Setting this bit will request a transmission of a receiving node.
- **FDF:** FD Format; if this bit is set, a CAN FD frame will be transmitted; otherwise, a CAN 2.0 frame will be transmitted. If Normal CAN 2.0 mode is selected, this bit is ignored and only CAN 2.0 frames are transmitted.
- **BRS:** Bit Rate Switch; the data phase of a CAN FD frame will be transmitted using DBR if this bit is set. If the bit is clear, the whole frame will be transmitted using NBR.
- **ESI:** Error State Indicator; normally, the ESI bit reflects the error status of the transmitting node. A recessive ESI bit in a CAN FD frame indicates that the transmitting node is error passive; a dominant bit shows that the transmitting node is error active. If ESIGM (CxCON[1]) = `0`, this bit in the object is ignored. If ESIGM = `1`, the ESI bit in the transmitted message will be transmitted recessive if the CAN FD Protocol Module is error passive, or if the ESI bit in the message object is set. A gateway application would use it to signal that the ESI bit of the transmitting node is set.

- **SEQ:** Sequence Number; SEQ is not transmitted on the CAN bus. It is used to keep track of the transmitted messages. SEQ is stored in the TEF message object.
- **Transmit Buffer Data:** Contains the payload of the message. Only the number of data bytes specified by the DLC are transmitted. Byte 0 is transmitted first, followed by 1, 2 and so on.

14.7.2. Loading Messages into Transmit FIFO

Before loading a message into the FIFO, the application must ensure that the FIFO is not full. There is space in the FIFO if TFNRFNIF (CxFIFOSTAx[0]) is set. Loading a message into a full FIFO can corrupt a message that is being transmitted.

The FIFO user address (CxFIFOUAx) points to the RAM of the next transmit message object where the application should store the message. T0 of the transmit message object is loaded first, followed by T1, T2 and so on. The maximum number of data bytes is limited by the configured payload. Only the number of data bytes specified by the DLC have to be loaded.

After the message object is loaded into RAM, the FIFO needs to be incremented by setting the UINC bit (CxFIFOCONx[8]). Doing so will cause the CAN FD Protocol Module to increment the head of the FIFO and update CxFIFOUAx.

Now the message is ready for transmission, and the next message can be loaded at the new address.

14.7.3. Loading Messages Into Transmit Queue

Loading transmit message objects into the TXQ works similarly to loading message objects into a transmit FIFO. The application must check the CxTXQSTA register to see if there is space in the TXQ. The CxTXQUA registers should be used instead of the CxFIFOUAx registers to calculate the address to load the message and set the UINC bit (CxTXQCON[8]) to increment the head of the TXQ.

Table 14-9. Transmit Message Object (TXQ and TX FIFO)

Words	Bits	Bits 31/23/5/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
T0	31:24	—	—	SID11	EID[17:6]				
	23:16	EID[12:5]							
	15:8	EID[4:0]					SID[10:8]		
	7:0	SID[7:0]							
T1	31:24	SEQ[22:15]							
	23:16	SEQ[14:7]							
	15:8	SEQ[6:0]							ESI
	7:0	FDF	BRS	RTR	IDE	DLC[3:0]			
T2	31:24	Transmit Buffer Data Byte 3							
	23:16	Transmit Buffer Data Byte 2							
	15:8	Transmit Buffer Data Byte 1							
	7:0	Transmit Buffer Data Byte 0							
T3	31:24	Transmit Buffer Data Byte 7							
	23:16	Transmit Buffer Data Byte 6							
	15:8	Transmit Buffer Data Byte 5							
	7:0	Transmit Buffer Data Byte 4							
Ti	31:24	Transmit Buffer Data Byte n							
	23:16	Transmit Buffer Data Byte n - 1							
	15:8	Transmit Buffer Data Byte n - 2							
	7:0	Transmit Buffer Data Byte n - 3							

bit T0.31-30	**Unimplemented:** Read as 'x'
bit T0.29	**SID11:** In FD mode, the standard ID can be extended to 12 bit using r1.
bit T0.28-11	**EID[17:0]:** Extended Identifier
bit T0.10-0	**SID[10:0]:** Standard Identifier
bit T1.31-9	**SEQ[22:0]:** Sequence to keep track of transmitted messages in Transmit Event FIFO
bit T1.8	**ESI:** Error Status Indicator In CAN to CAN Gateway mode (CiCON.EXIGM=1), the transmitted ESI flag is a "logical OR" of T1.ESI and error passive state of the CAN controller; In Normal mode ESI indicates the error status. 1 = Transmitting node is error passive. 0 = Transmitting node is error active.
bit T1.7	**FDF:** FD Frame; distinguishes between CAN and CAN FD formats.
bit T1.6	**BRS:** Bit Rate Switch; selects if data bit rate is switched.
bit T1.5	**RTR:** Remote Transmission Request; not used in CAN FD.
bit T1.4	**IDE:** Identifier Extension Flag; distinguishes between the base and extended format.
bit T1.3-0	**DLC[3:0]:** Data Length Code

14.7.4. Requesting Transmission of Message in Transmit FIFO

After a message is loaded into a transmit FIFO, the message is ready for transmission. The application initiates the transmission of all messages in a FIFO by setting the TXREQ bit (CxFIFOCONx[9]) or by setting the corresponding bit in the CxTXREQ registers. When all messages are transmitted, TXREQ gets cleared. The application can request transmission of multiple FIFOs and the TXQ simultaneously. The FIFO or TXQ with the highest priority will start transmitting first. Messages in a FIFO will be transmitted First-In-First-Out.

Messages can be loaded into a FIFO while the FIFO is transmitting messages. Since TXREQ is cleared by the FIFO automatically after the FIFO empties, UINC and TXREQ of the CxFIFOCONx register must be set at the same time after appending a message. This ensures that all messages in the FIFO are transmitted, including the appended messages.

14.7.5. Requesting Transmission of Message in Transmit Queue

After a message is loaded into the TXQ, the message is ready for transmission. The application initiates the transmission of all messages in the queue by setting TXREQ (CxTXQCON[9]). When all messages have been transmitted, TXREQ will be cleared. The application can request transmission of the TXQ and multiple FIFOs simultaneously. The TXQ or FIFO of the CxTXQCON register with the highest priority will start transmitting first. Messages in the TXQ will be transmitted based on their ID. The message with the highest priority ID and the lowest ID value will be transmitted first.

Messages can be loaded into the TXQ while the TXQ is transmitting messages. Since TXREQ is cleared by the TXQ automatically after the TXQ empties, UINC and TXREQ of the CxTXQCON register must be set at the same time after appending a message. This ensures that all messages in the TXQ are transmitted, including the appended messages.

14.7.6. CxTXREQ Register

The CxTXREQ register contains the TXREQ[31:0] bits of the TXQ and of all the TX FIFOs. They have the following purposes:

- The user application can request transmission of the TXQ and/or one or more TX FIFOs by setting the corresponding bits in the CxTXREQ register. Clearing a bit does NOT abort any transmissions.
- Reading the CxTXREQ register gives information about which transmit FIFOs have transmissions pending.

CxTXREQ[0] is mapped to the TXQ, CxTXREQ[1] is mapped to TX FIFO 1, CxTXREQ[2] is mapped to TX FIFO 2 and so on. CxTXREQ[31] is mapped to TX FIFO 31.

14.7.7. Transmit Priority

The transmit priority of the FIFOs and TXQ needs to be configured using the TXPRIx bits (CxFIFOCONx[20:16] and CxTXQCON[20:16]).

Before transmitting a message, the priorities of the TXQ and the TX FIFOs queued for transmission are compared. The FIFO/TXQ with the highest priority will be transmitted first. For example, if transmit FIFO 1 has a higher priority setting than FIFO 3, all messages in FIFO 1 will be transmitted first. If multiple FIFOs have the same priority, the FIFO with the highest index is transmitted. For example, if FIFO 1 and FIFO 3 have the same priority setting, all messages in FIFO 3 will be transmitted first. If the TXQ and one or more FIFOs have the same priority, all messages in the TXQ will be transmitted first.

The transmit priority will be recalculated after every successful transmission of a single message.

14.7.7.1. Transmit Priority of Messages in FIFO

In this method, the messages in a FIFO are transmitted First-In-First-Out.

14.7.7.2. Transmit Priority of Messages in TXQ

Messages in the TXQ are transmitted based on the message ID. The message with the lowest message ID (highest priority) is transmitted first.

14.7.7.3. Transmit Priority Based on ID

The goal of transmitting CAN messages based on ID is to avoid Inner Priority Inversion. If a low-priority message is waiting to get transmitted due to bus traffic (arbitration), a higher priority message could be prevented from being transmitted. The TXQ solves this issue by reprioritizing the messages in the queue based on priority (ID).

14.7.8. Transmit Bandwidth Sharing

The bandwidth sharing feature works as follows:

- After a successful transmission of a message, the module will remain Idle for *n* arbitration bit times before the module attempts to transmit the next message; it suspends the next transmission.
- After the device has received a message, the module can transmit the next message as soon as the bus is Idle.

This allows other nodes on the bus to transmit their messages, even though they are of lower priority.

The number of arbitration bit times between transmissions can be configured using the TXBWS[3:0] bits (CxCON[31:28]).

14.7.9. Retransmission Attempts

The number of retransmission attempts can be configured as follows:

- Retransmission attempts are disabled.
- Three retransmission attempts
- Unlimited retransmissions

The retransmission attempts can be restricted by setting RTXAT (CxCON[16]). The number of retransmission attempts can be configured individually for each transmit FIFO and the TXQ using TXAT[1:0] (CxFIFOCON[22:21] and CxTXQCON[22:21], respectively).

In case RTXAT = `0`, unlimited retransmission attempts will be used for all transmit FIFOs, and the TXQ and TXATx will be ignored.

14.7.9.1. Retransmission Attempts Disabled

TXREQ will be cleared after the attempt to transmit the message. If the message is not successfully transmitted due to loss of arbitration or due to an error, TXATIF in the CxFIFOSTAx or CxTXQSTA register will be set.

14.7.9.2. Three Retransmission Attempts

In case an error is detected during transmission, the CAN FD Protocol module will decrement the number of remaining attempts and try to retransmit the message the next time the bus is Idle. In case arbitration is lost, the number of remaining attempts will not change. If all retransmission attempts are exhausted, TXREQ will be cleared and TXATIF in CxFIFOSTAx or CxTXQSTA will be set.

Before retransmitting the message, the transmit priority will be recalculated. The retransmission attempts will be reinitialized if a different TX FIFO or TXQ is selected for transmission, or if a message is received after the last transmission attempt.

14.7.9.3. Unlimited Retransmissions

TXREQ will be cleared only after all messages in the TX FIFO or TXQ are successfully transmitted.

14.7.10. Aborting Transmission

A pending transmission can only be aborted before the transmission of the message starts, before the Start-of-Frame (SOF).

The transmission of a specific FIFO can be aborted by clearing TXREQ in the CAN Transmit Queue Control register; it cannot be aborted by clearing the bits in the CxTXREQ registers. Writing a '0' to one of the bits in the CxTXREQ registers will be ignored. The TXABT bit in the CAN FIFO Status x register will be set after a successful abortion. TXREQ will remain set until the message either aborts or is successfully transmitted.

Setting ABAT (CxCON[27]) will abort all pending messages of all FIFOs. After all TXREQx bits are cleared, ABAT has to be cleared in order to be able to transmit new messages.

Clearing TXREQ for a transmit FIFO will attempt to abort all transmissions in the FIFO. If a message is successfully transmitted, the FIFO index will be updated as normal. If the message is successfully aborted, the FIFO index will not change.

The user can then use the FIFO Message Index bits, FIFOCI[4:0] (CxFIFOSTAx[12:8]), to identify messages that are transmitted. To reset the transmit FIFO index and erase all pending messages, the user can set the FRESET bit. The FIFO can then be loaded with new messages to be transmitted.

14.7.11. Remote Transmit Request – RTR

The CAN bus system has a method for allowing a master node to request data from another node. The master sends a message with the RTR bit set. The message contains no data, only an address to trigger a filter match.

Remote frames are only specified for CAN 2.0 frames; they are not supported in CAN FD frames.

The filter that is configured to respond to a Remote Transmit Request will point to a FIFO that is configured for transmission, and RTREN has to be set.

Automatic remote data requests can be handled without MCU intervention. If a FIFO is properly configured, when a filter matches and points to the FIFO, the FIFO will be queued for transmission.

The FIFO must be configured as follows:

- Set TXEN to '1'.
- A filter must be enabled and loaded with a matching message identifier.
- The buffer pointer for that filter must point to the TX FIFO. (Normally, a filter points to an RX FIFO.)

- RTREN bit must be set to '1' to enable RTR.
- The FIFO must be preloaded with at least one message to be sent.

When an RTR message is received, and it matches a filter pointing to a properly configured transmit FIFO, the TXREQ bit is set, queuing the object for transmission according to priorities.

A FIFO will only be transmitted if TXEN and RTREN are set, and if it is NOT empty. When a request for a remote transmission occurs while the FIFO is empty, the event will be treated as an overflow and the RXOVIF bit will be set.

14.7.12. Mismatch of DLC and Payload Size During Transmission

The PLSIZEx bits reserve a certain number of bytes in the transmit FIFO. The CAN FD Protocol module handles mismatches between the DLC and payload size as follows:

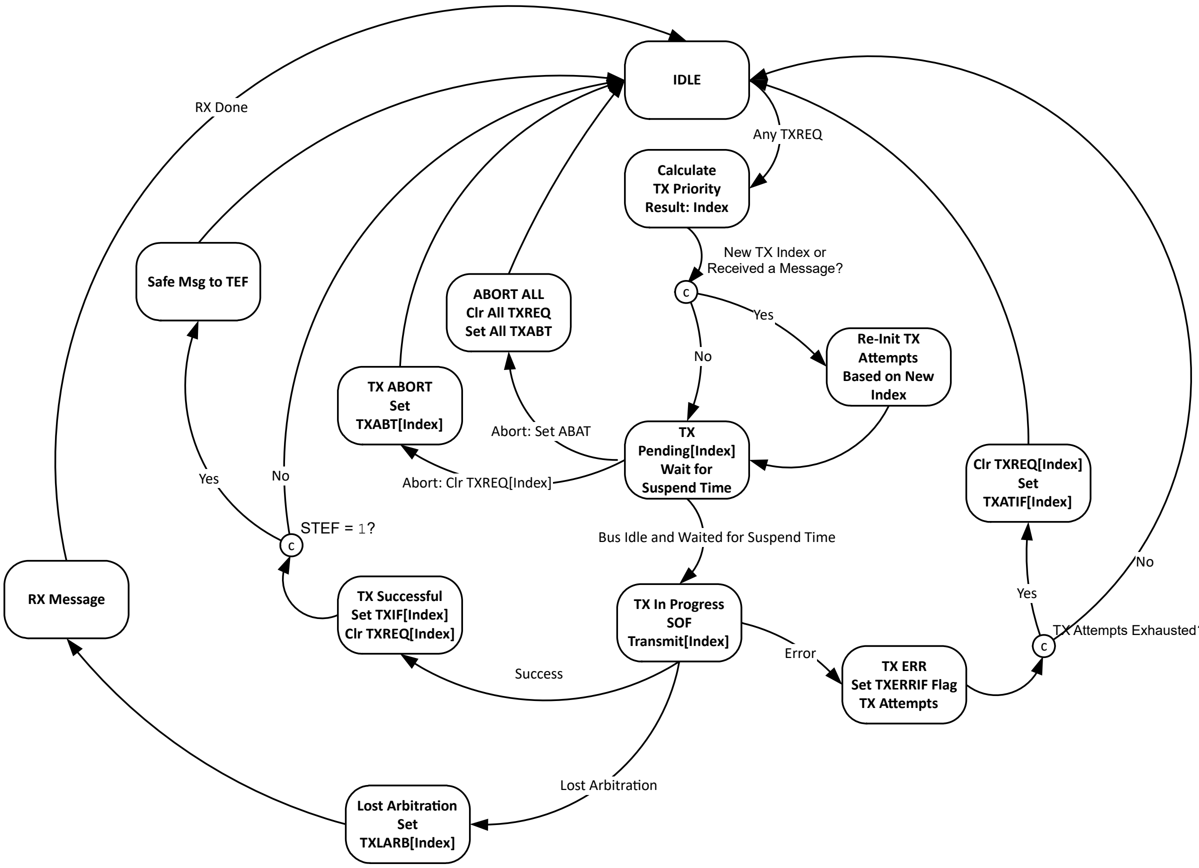
- If the DLC is smaller than the reserved payload, the number of data bytes specified by the DLC will be transmitted.
- If the DLC is bigger than the reserved payload, the module will not transmit the message. Instead, it will set the IVMIF (CxINT[15]) and DLCMM (CxBDIAG1[31]) flags and clear the TXREQ flag. The application can use the TEF to identify the message that is not transmitted.

14.7.13. Transmit State Diagram

Figure 5-1 describes how messages are queued for transmission. It illustrates how the most important transmit flags are set and cleared:

1. Messages are queued for transmission by setting the TXREQ flag.
2. The transmit priority will be determined. The FIFO or TXQ with the highest priority TXPRIx flag will be selected. The index of the TX message in the FIFO or TXQ will be calculated.
3. The TX message is pending for transmission.
4. Transmission can only start when the bus is Idle.
5. A pending transmission can only be aborted before SOF is transmitted.
6. During the transmission of a message, the CAN FD Protocol Module checks for the following:
   - Loss of arbitration during the arbitration field
   - Transmit errors
7. In case a message of a TX FIFO or the TXQ is transmitted successfully, the TXREQ will only be cleared after all messages of the FIFO are transmitted. After the transmission of any message, the status flags of the FIFO or TXQ are updated. In case STEF (CxCON[19]) is set, the message will be stored into the TEF and a timestamp will be attached if enabled.
8. In case arbitration is lost, TXLARB of the TX FIFO or TXQ will be set and the device will switch over to receiving the message (see Message Reception).
9. In case an error is detected during the transmission of a message, an error frame will be transmitted and the appropriate error flags will be set. Messages will be retransmitted according to Retransmission Attempts.

Figure 14-15. Transmit State Diagram



14.7.14. Resetting Transmit FIFO

A Transmit FIFO can be reset by:

Setting FRESET (CxFIFOCONx[10]) or

Placing the module in Configuration mode (OPMOD[2:0] = `100`).

Resetting the FIFO will reset the head and tail pointers and the CxFIFOSTAx register. The settings in the CxFIFOCONx register will not change.

Before resetting a TX FIFO using FRESET, ensure no transmissions are pending.

14.7.15. Resetting Transmit Queue

The Transmit Queue can be reset by:

- Setting FRESET (CxTXQCON[10]) or
- Placing the module in Configuration mode (OPMOD[2:0] = `100`).

Resetting the TXQ will reset the head and tail pointers and the CxTXQSTA register. The settings in the CxTXQCON and CxTXQCON registers will not change.

Before resetting the TXQ using FRESET, ensure no transmissions are pending.

14.7.16. Message Transmission Code Example

Example 14-1. Message Transmission Code

```
#include <xc.h>
/* This code example demonstrates a method to configure the CAN FD module to
transmit Standard and Extended ID CAN FD messages. This uses CAN1, TXQ and
FIFO1. TXQ size is 1 and FIFO1 size is 2. */
/* Include fuse configuration code here. */

#define MAX_WORDS 100
unsigned int __attribute__((aligned(4)))CanTxBuffer[MAX_WORDS];
/*Data structure to implement a CANFD message buffer. */
/* CANFD Message Time Stamp */
typedef unsigned long CANFD_MSG_TIMESTAMP;

/* CAN TX Message Object Control*/
typedef struct _CANFD_TX_MSGOBJ_CTRL {
    unsigned DLC : 4;
    unsigned IDE : 1;
    unsigned RTR : 1;
    unsigned BRS : 1;
    unsigned FDF : 1;
    unsigned ESI : 1;
    unsigned SEQ : 23;
    unsigned unimplemented1 : 16;
} CANFD_TX_MSGOBJ_CTRL;

/* CANFD TX Message ID*/
typedef struct _CANFD_MSGOBJ_ID {
    unsigned SID : 11;
    unsigned long EID : 18;
    unsigned SID11 : 1;
    unsigned unimplemented1 : 2;
} CANFD_MSGOBJ_ID;

/* CAN TX Message Object*/
typedef union _CANFD_TX_MSGOBJ {
    struct {
        CANFD_MSGOBJ_ID id;
        CANFD_TX_MSGOBJ_CTRL ctrl;
    } bF;
    unsigned int word[4];
    unsigned char byte[8];
} CANFD_TX_MSGOBJ;

int main(void) {
    unsigned char index;

    /* The dsPIC33AK256MPS306 device features I/O remap. This I/O remap
configuration for
    the CAN FD module can be performed here. */
    SetIORemapForCANFDModule();

    /* Configure the CRU clock generator unit corresponding to CANFD to yield
     * 40MHz clock. This clock is CANFD module clock */
    ConfigureCANFDClockFor40MHz(); // FCAN = 40 MHz

    /* Enable the CANFD module */
    C1CONbits.ON = 1;

    /* Place CAN module in configuration mode */
    C1CONbits.REQOP = 4;
    while (C1CONbits.OPMOD != 4);

    /* Initialize the C1FIFOBA with the start address of the CAN FIFO message
    buffer area. */
    C1FIFOBA = (unsigned int) &CanTxBuffer;

    /* Set up the CANFD module for 1Mbps Nominal bit rate speed and
     * 2Mbps Data bit rate.
     */
    C1NBTCFG = 0x001E0707;
    C1DBTCFG = 0x000E0303;
    C1TDC = 0x00020F00;             //TDCMOD is Auto
```

```
    /* Configure CANFD module to enable Transmit Queue and BRS*/
    C1CONbits.BRSDIS = 0x0;
    C1CONbits.STEF = 0x0;           //Don't save transmitted messages in TEF
    C1CONbits.TXQEN = 0x1;

    /* Configure TXQ to transmit 1 message*/
    C1TXQCONbits.FSIZE = 0x0;       // single message
    C1TXQCONbits.PLSIZE = 0x7;      // 64 bytes of data

    /* Configure FIFO1 to transmit 2 messages*/
    C1FIFOCON1bits.FSIZE = 0x1;     //2 messages
    C1FIFOCON1bits.PLSIZE = 0x2;    //16 bytes of data
    C1FIFOCON1bits.TXEN = 0x1;      // Set TXEN bit, transmit fifo

    /* Place the CAN module in Normal mode. */
    C1CONbits.REQOP = 0;
    while (C1CONbits.OPMOD != 0);

    /* Get the address of the message buffer to write to. Load the buffer
    and then set the UINC bit.
    Set the TXREQ bit next to send the message. */
    CANFD_TX_MSGOBJ *txObj;

    /* Transmit message from TXQ - CANFD base frame with BRS*/
    /* SID = 0x100, 64 bytes of data */
    txObj = (CANFD_TX_MSGOBJ *) C1TXQUA;
    txObj->bF.id.SID = 0x100;
    txObj->bF.id.EID = 0x0000;
    txObj->bF.ctrl.BRS = 1;         //Switch bit rate
    txObj->bF.ctrl.DLC = 0xF;       //64 bytes
    txObj->bF.ctrl.FDF = 1;         //CANFD frame
    txObj->bF.ctrl.IDE = 0;         //Standard frame
    for (index = 0; index < 0x40; index++) {
        //64 bytes of 0x5A
        txObj->byte[index + 8] = 0x5A;
    }
    C1TXQCONbits.UINC = 1;          // Set UINC bit
    C1TXQCONbits.TXREQ = 1;         // Set TXREQ bit

    /* Transmit message 0 from FIFO 1 - CANFD base frame with BRS*/
    /* SID = 0x300 , 16 bytes of data */
    txObj = (CANFD_TX_MSGOBJ *) C1FIFOUA1;
    txObj->bF.id.SID = 0x300;
    txObj->bF.id.EID = 0x0000;
    txObj->bF.ctrl.BRS = 1;         //Switch bit rate
    txObj->bF.ctrl.DLC = 0xA;       //16 bytes
    txObj->bF.ctrl.FDF = 1;         //CANFD frame
    txObj->bF.ctrl.IDE = 0;         //Standard frame
    for (index = 0; index < 0x10; index++) {
        //16 bytes of 0xA5
        txObj->byte[index + 8] = 0xA5;
    }

    C1FIFOCON1bits.UINC = 1;        //Set UINC bit
    C1FIFOCON1bits.TXREQ = 1;       //Set TXREQ bit
    /* Transmit message 1 from FIFO 1 - CANFD base frame with BRS*/
    /* SID = 0x500, EID = 0xC000, 12 bytes of data */
    txObj = (CANFD_TX_MSGOBJ *) C1FIFOUA1;
    txObj->bF.id.SID = 0x500;
    txObj->bF.id.EID = 0xC000;
    txObj->bF.ctrl.BRS = 1;         //Switch bit rate
    txObj->bF.ctrl.DLC = 0x9;       //12 bytes
    txObj->bF.ctrl.FDF = 1;         //CANFD frame
    txObj->bF.ctrl.IDE = 1;         //Extended frame
    for (index = 0; index < 0xC; index++) {
        //12 bytes of 0x55
        txObj->byte[index + 8] = 0x55;
    }

    C1FIFOCON1bits.UINC = 1;        //Set UINC bit
    C1FIFOCON1bits.TXREQ = 1;       //Set TXREQ bit
    while (1);
}
```

14.8. Transmit Event FIFO - TEF

The TEF allows the application to keep track of the order and time in which the messages are transmitted. The TEF works similarly to a receive FIFO. Instead of storing received messages, it stores transmitted messages. Messages are only saved if STEF (CxCON[19]) is set. The sequence number (SEQ) of the transmitted message is copied into the TEF object. The payload data are not stored. Transmitted messages are time-stamped if TEFTSEN is set.

Table 6-1 specifies the TEF object. The first two words of the TEF object are a copy of the transmit message object. Optionally, the TEF object contains the timestamp when the message is transmitted.

14.8.1. Reading a TEF Object

Before reading a TEF object, the application must check that the TEF is not empty by reading the CxTEFSTA register. The TEF is not empty if TEFNEIF is set.

The TEF user address points to the address in RAM of the next TEF object to read. The actual address in RAM is calculated using Equation 14-25. TE0 of the TEF object is read first, followed by TE1 and TE2.

Equation 14-25. Start Address of TEF Object

$$A = BaseAddress = CxFIFOBA$$

After the TEF object is read from RAM, the TEF needs to be incremented by setting UINC (CxTEFCON[8]). This will cause the CAN FD Protocol module to increment the tail pointer and update CxTEFUA.

Now, the next message can be read from the TEF.

14.8.1.1. Resetting the TEF

TEF can be reset by:

- Setting FRESET (CxTEFCON[10]) or
- Placing the module in Configuration mode (OPMOD[2:0] = `100`).

Resetting the FIFO will reset the head and tail pointers and the CxTEFSTA register. The settings in the CxTEFCON and CxTEFCON registers will not change.

Table 14-10. Transmit Event FIFO Object

Words	Bits	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
TE0	31:24	—	—	SID11	EID[17:6]				
	23:16	EID[12:5]							
	15:8	EID[4:0]					SID[10:8]		
	7:0	SID[7:0]							
TE1	31:24	SEQ[22:15]							
	23:16	SEQ[14:7]							
	15:8	SEQ[6:0]							ESI
	7:0	FDF	BRS	RTR	IDE	DLC[3:0]			
TE2	31:24	TXMSGTS[31:24]							
	23:16	TXMSGTS[23:16]							
	15:8	TXMSGTS[15:8]							
	7:0	TXMSGTS[7:0]							

bit TE0.31-30 **Unimplemented:** Read as ‘x’

bit TE0.29	**SID11:** In FD mode, the standard ID can be extended to 12 bit using r1.
bit TE0.28-11	**EID[17:0]:** Extended Identifier
bit TE0.10-0	**SID[10:0]:** Standard Identifier
bit TE1.31-9	**SEQ[22:0]:** Sequence to keep track of transmitted messages in Transmit Event FIFO.
bit TE1.8	**ESI:** Error Status Indicator 1 = Transmitting node is error passive. 0 = Transmitting node is error active.
bit TE1.7	**FDF:** FD Frame; distinguishes between CAN and CAN FD formats.
bit TE1.6	**BRS:** Bit Rate Switch; selects if data bit rate is switched.
bit TE1.5	**RTR:** Remote Transmission Request; not used in CAN FD.
bit TE1.4	**IDE:** Identifier Extension Flag; distinguishes between base and extended format.
bit TE1.3-0	**DLC[3:0]:** Data Length Code
bit TE2.31-0	**TXMSGTS[31:0]:** Transmit Message Timestamp

14.8.2. Transmit Event FIFO Code Example

A code example to save the transmitted messages using TEF is shown in Example 14-2.

Example 14-2. Using the Transmit Event FIFO Code

```
/* Include fuse configuration code here. */
#define MAX_WORDS 100
unsigned int __attribute__((aligned(4)))CanTxBuffer[MAX_WORDS]; //message
buffer to be written
unsigned int * currentMessageBuffer; //Points to message buffer to be read
/*data structure to implement a CANFD message buffer. */
/* CANFD Message Time Stamp */
typedef unsigned long CANFD_MSG_TIMESTAMP;

/* CAN TX Message Object Control*/
typedef struct _CANFD_TX_MSGOBJ_CTRL {
    unsigned DLC : 4;
    unsigned IDE : 1;
    unsigned RTR : 1;
    unsigned BRS : 1;
    unsigned FDF : 1;
    unsigned ESI : 1;
    unsigned SEQ : 23;
    unsigned unimplemented1 : 16;
} CANFD_TX_MSGOBJ_CTRL;

/* CANFD TX Message ID*/
typedef struct _CANFD_MSGOBJ_ID {
    unsigned SID : 11;
    unsigned long EID : 18;
    unsigned SID11 : 1;
    unsigned unimplemented1 : 2;
} CANFD_MSGOBJ_ID;

/* CAN TX Message Object*/
typedef union _CANFD_TX_MSGOBJ {

    struct {
        CANFD_MSGOBJ_ID id;
        CANFD_TX_MSGOBJ_CTRL ctrl;
    } bF;
    unsigned int word[4];
    unsigned char byte[8];
} CANFD_TX_MSGOBJ;

/* CANFD TEF Message Object */
typedef union _CAN_TEF_MSGOBJ {

    struct {
        CANFD_MSGOBJ_ID id;
        CANFD_TX_MSGOBJ_CTRL ctrl;
        CANFD_MSG_TIMESTAMP timeStamp;
    } bF;
    unsigned int word[6];
```

```
    unsigned char byte[12];
} CANFD_TEF_MSGOBJ;

int main(void) {
    unsigned char index, fifoSize;

    /* The dsPIC33A device features I/O remap. This I/O remap configuration for
    the CAN FD module can be performed here. */
    SetIORemapForCANFDModule();

    /* Configure the CRU clock generator unit corresponding to CANFD to yield
     * 40MHz clock. This clock is CANFD module clock */
    ConfigureCANFDClockFor40MHz(); // FCAN = 40 MHz

    /* Enable the CANFD module */
    C1CONbits.ON = 1;

    /* Place CAN module in configuration mode */
    C1CONbits.REQOP = 4;
    while (C1CONbits.OPMOD != 4);

    /* Initialize the C1FIFOBA with the start address of the CAN FIFO message
    buffer area. */
    C1FIFOBA = (unsigned int) &CanTxBuffer;

    /* Set up the CANFD module for 1Mbps Nominal bit rate speed and
     * 2Mbps Data bit rate.
     */
    C1NBTCFG = 0x001E0707;
    C1DBTCFG = 0x000E0303;
    C1TDC = 0x00020F00;             //TDCMOD is Auto

    /* Configure CANFD module to save transmitted messages in TEF and BRS*/
    C1CONbits.BRSDIS = 0x0;
    C1CONbits.STEF = 0x1;
    C1CONbits.TXQEN = 0x0;          // Disable TXQ

    /* Configure TEF to save 5 messages*/
    C1TEFCONbits.FSIZE = 0x4;       // save 5 messages
    C1TEFCONbits.TEFTSEN = 1;

    /* Configure FIFO1 to transmit 5 messages*/
    C1FIFOCON1bits.FSIZE = 0x4;     // 5 messages
    C1FIFOCON1bits.PLSIZE = 0x7;    // 64 bytes of data
    C1FIFOCON1bits.TXEN = 0x1;      // Set TXEN bit ,transmit fifo

    /* Place the CAN module in Normal mode. */
    C1CONbits.REQOP = 0;
    while (C1CONbits.OPMOD != 0);

    /* Get the address of the message buffer to write to. Load the buffer and
     * then set the UINC bit. Set the TXREQ bit to send the message.
     */
    CANFD_TX_MSGOBJ *txObj;

    /* Transmit 5 messages from FIFO 1 - CANFD base frame with BRS*/
    /* SID = 0x300 , 64 bytes of data */
    for (fifoSize = 0; fifoSize < 5; fifoSize++) {
        txObj = (CANFD_TX_MSGOBJ *) C1FIFOUA1;
        txObj->bF.id.SID = 0x300;
        txObj->bF.id.EID = 0x0000;
        txObj->bF.ctrl.BRS = 1;     //Switch bit rate
        txObj->bF.ctrl.DLC = 0xF;   //64 bytes
        txObj->bF.ctrl.FDF = 1;     //CANFD frame
        txObj->bF.ctrl.IDE = 0;     //Standard frame
        txObj->bF.ctrl.SEQ = fifoSize; //Sequence stored in TEF
        for (index = 0; index < 0x40; index++) {
            //64 bytes of 0xA5
            txObj->byte[index + 8] = 0xA5;
        }
        C1FIFOCON1bits.UINC = 1; // Set UINC bit
    }

    C1FIFOCON1bits.TXREQ = 1; // Set TXREQ bit
    while (C1FIFOCON1bits.TXREQ == 1);

    /* Keep reading the TEF objects until the last transmitted message*/
    for (fifoSize = 0; fifoSize < 5; fifoSize++) {
```

```
        while (C1TEFSTAbits.TEFNEIF == 0);
        CANFD_TEF_MSGOBJ *tefObj;
        tefObj = (CANFD_TEF_MSGOBJ *) C1TEFUA;
        //ProcessTEFMessages (currentMessageBuffer) ;
        C1TEFCONbits.UINC = 1; // Set UINC bit
    }
    while (1);
}
```

14.9. Message Filtering

All messages on a CAN network will be received by all nodes. In order to process only messages of interest, a hardware filtering mechanism is implemented. The CAN FD Protocol module can be configured to receive only messages of interest. The module contains a maximum of 32 acceptance filters. Each acceptance filter contains a filter object and a mask object. The user application configures the specific filter to receive a message with a given identifier by setting the filter object and mask object to match the identifier of the message to be received.

14.9.1. Filter Configuration

The filters are controlled by the CxFLTCON register. The filters must be disabled by clearing the FLTEN bit before changing the filter or mask object; the module need not be in Configuration mode. After the filter object is updated, the buffer pointer, FnBP, has to be initialized and the filter can be enabled by setting the FLTEN bit. The FnBP points to the FIFO where the matching receive message needs to be stored.

14.9.2. Filtering a Received Message

The CAN FD Protocol module starts acceptance filtering after the arbitration field and when the first three data bytes of a message are received. Figure 14-16 describes the flow of message filtering.

The module loops through all the filters, starting with Filter 0, which is the highest priority filter. The message in the Receive Message Assembly Buffer (RXMAB) is compared to the filter and mask. In case the message matches the filter and it is received without any errors, the message will be stored into the RX FIFO pointed to by the FnBP. Acceptance filtering is stopped and the associated RFIF bit is set.

In case an RTR is received, the TXREQ bit of the TX FIFO pointed to by FnBP will be set.

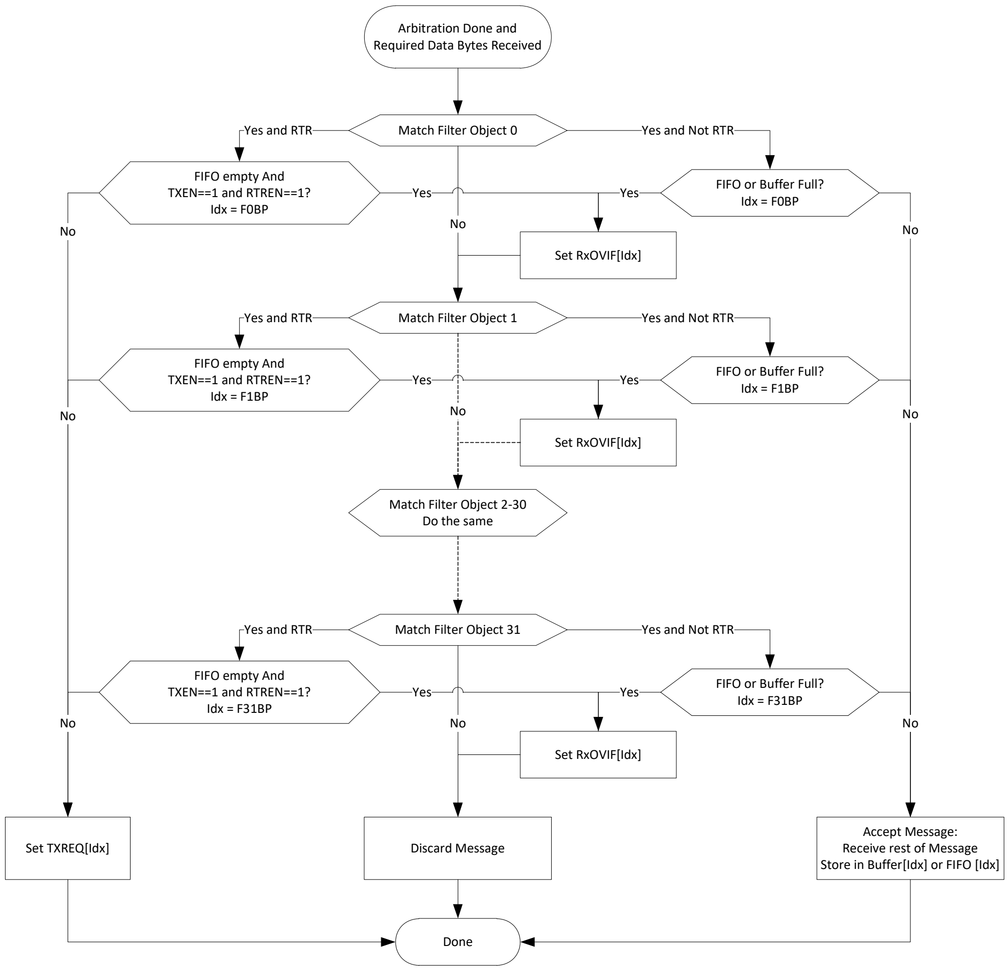
Filtering will continue with the next filter, and RXOVIF will be set only when one of the following happens:

- A filter matches, but the RX FIFO is full.
- When multiple filters match the same message and all matching RX FIFOs are full, only the RXOVIF of the FIFO pointed to by the highest priority filter will be set.
- The RXOVIF bit will be set if the TX FIFO is empty during an RTR (TXEN = 1, RTREN = 1).

If none of the filters match, the received message will be discarded.

Note: If the module receives a message that matches a filter, but the corresponding FIFO is a TX FIFO (TXEN = 1, RTREN = 0), the module will discard the received message.

Figure 14-16. Message Filtering Flow



14.9.2.1. Filtering Standard or Extended Frames

Figure 14-17 illustrates the flow of matching a single filter object to the received message in the RXMAB.

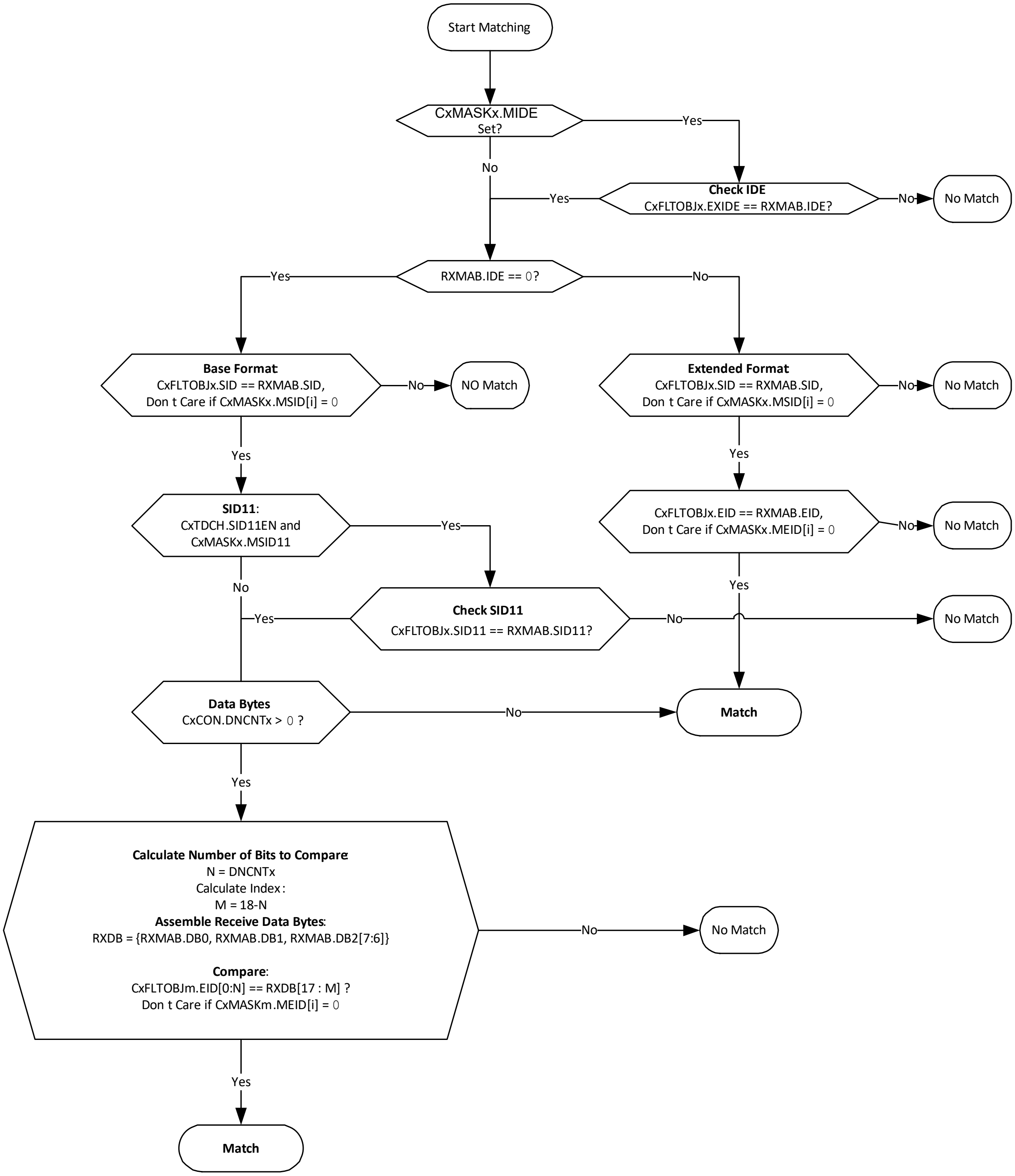
The filter object can be configured to accept either the standard, extended or both frames. If MIDE is clear, both the standard and extended frames will be accepted.

If the filter should only accept standard frames, then MIDE must be set and EXIDE must be cleared. If the filter should only accept extended frames, then both MIDE and EXIDE must be set.

14.9.2.2. Mask Bits

The mask object is used to ignore selected bits of the received identifier. The masked bits (mask bits with a value of '0') of the RXMAB will not be compared with the bits in the filter object. For example, to receive all messages with Identifiers 0, 1, 2 and 3, it is required to mask the lower two bits of the identifier by clearing the corresponding bits of the mask object.

Figure 14-17. Filter Match



14.9.2.3. Filtering on Data Bytes

When the filter is configured to receive standard frames, the EID part of the filter and mask object can be selected to filter the data bytes. The DNCNT[4:0] bits in the CxCON register are used to select how many bits in the data bytes are compared. Table 14-11 explains how many data bits are compared, and which filter bits and data bits are compared.

If DNCNTx is:

- '0', then data byte filtering is disabled.
- Non-zero, the filtering will commence on as many data bits as specified in DNCNTx. A filter hit will require matching of the SIDx bits and a match of *n* data bits with the filter's EID[0:17] bits. Data Byte 0[7] is always compared to EID[0], Data Byte 0[6] to EID[1] and Data Byte 2[6] to EID[17].
- Greater than 18, indicating that the user-selected number of bits is greater than the total number of EIDx bits. The filter comparison will terminate with the 18th bit of the data.
- Greater than 16, and the received message has DLC = 2, indicating a payload of two data bytes. The filter comparison will terminate with the 16th bit of the data.
- Greater than 8, and the received message has DLC = 1, indicating a payload of one data byte. The filter comparison will terminate with the 8th bit of the data.
- Greater than 0, and the received message has DLC = 0, indicating no data payload. The filter comparison will terminate with the identifier.

14.9.2.4. 12-Bit Standard ID

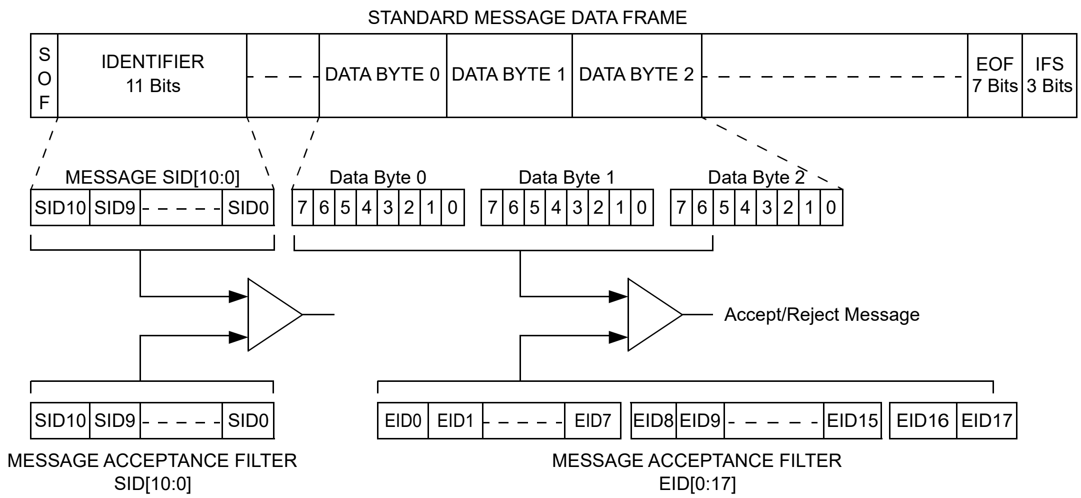
Setting SID11EN (CxTDC[24]) allows the use of RRS as bit 12 of the SIDx (LSB). 12-Bit SID mode is only available for CAN FD base frames. The filter is extended by SID11 and MSID11. Data bytes can also be filtered in this mode.

Table 14-11. Data Byte Filter Configuration

DNCNT[4:0]	Received Message Data Bits to be Compared Byte [bits]	EIDx Bits Used for Acceptance Filter
00000	No Comparison	No Comparison
00001	Data Byte 0[7]	EID[0]
00010	Data Byte 0[7:6]	EID[0:1]
00011	Data Byte 0[7:5]	EID[0:2]
00100	Data Byte 0[7:4]	EID[0:3]
00101	Data Byte 0[7:3]	EID[0:4]
00110	Data Byte 0[7:2]	EID[0:5]
00111	Data Byte 0[7:1]	EID[0:6]
01000	Data Byte 0[7:0]	EID[0:7]
01001	Data Byte 0[7:0] and Data Byte 1[7]	EID[0:8]
01010	Data Byte 0[7:0] and Data Byte 1[7:6]	EID[0:9]
01011	Data Byte 0[7:0] and Data Byte 1[7:5]	EID[0:10]
01100	Data Byte 0[7:0] and Data Byte 1[7:4]	EID[0:11]
01101	Data Byte 0[7:0] and Data Byte 1[7:3]	EID[0:12]
01110	Data Byte 0[7:0] and Data Byte 1[7:2]	EID[0:13]
01111	Data Byte 0[7:0] and Data Byte 1[7:1]	EID[0:14]
10000	Data Byte 0[7:0] and Data Byte 1[7:0]	EID[0:15]
10001	Byte 0[7:0] and Byte 1[7:0] and Byte 2[7]	EID[0:16]
10010 to 11111	Byte 0[7:0] and Byte 1[7:0] and Byte 2[7:6]	EID[0:17]

Figure 14-18 illustrates how the first 18 data bits of the received message data payload are compared with the corresponding EIDx bits of the message acceptance filter (EID[17:0] bits in the CxFLTOBJx registers). The IDE bit of the received message must be '0'.

Figure 14-18. CAN Operation with DeviceNet™ Filtering



14.10. Message Reception

The application has to configure the RX FIFO before it can be used for reception (see Receive FIFO Configuration). In addition, the application has to configure and enable at least one filter (see Filter Configuration).

The CAN FD Protocol module continuously monitors the CAN bus. Messages that match a filter are stored in the RX FIFO pointed to by the filter (see Filtering a Received Message). The message data are stored in the receive message objects.

14.10.1. Receive Message Object

Table 14-12 specifies the receive message object used by the RX FIFOs. The receive objects contain the message ID, control bits, payload and timestamp.

- **SID:** Standard Identifier (ID) or Base ID
- **EID:** Extended Identifier
- **DLC:** Data Length Code; specifies the number of data bytes in the frame (see DLC Encoding).
- **IDE:** Identifier Extension; IDE = 0 means a Base Identifier frame is received. IDE = 1 means an Extended Identifier frame is received.
- **RTR:** Remote Transmit Request; this bit is only specified in CAN 2.0 frames. If this bit is set, the module is requested to respond with a frame transmission.
- **FDF:** FD Frame; if this bit is set, a CAN FD frame is received; otherwise, a CAN 2.0 frame is received.
- **BRS:** Bit Rate Switch; the data phase of a CAN FD frame is received using DBR if this bit is set. If the bit is clear, the whole frame is received using NBR.
- **ESI:** Error Status Indicator; the ESI bit reflects the error status of the transmitting node. A recessive ESI bit in a CAN FD frame indicates that the transmitting node is error passive; a dominant bit shows that the transmitting node is error active.
- **FILHIT:** Indicates the number of the filter that matched the received message.
- **RXMSGTS:** Time-stamp of the Received Message; timestamping can be enabled for each RX FIFO individually using RXTSEN (CxFIFOCONx[5]). The receive message object will not contain RXMSGTS if timestamping is disabled.

- **Receive Buffer Data:** Contains the payload of the message. The maximum payload is configured by the PLSIZEx bits (CxFIFOCONx[31:29]).

14.10.1.1. Reading a Receive Message Object

Before reading a receive message object, the application must ensure that the RX FIFO is not empty by reading the CxFIFOSTAx register. The RX FIFO is not empty if TFNRFNIF is set.

The RX FIFO user address (CxFIFOUAx) points to the RAM of the next receive message object to read. R0 of the receive message object is read first, followed by R1, R2 and so on.

After the receive message object is read from RAM, the RX FIFO needs to be incremented by setting the UINC bit (CxFIFOCONx[8]). This will make the CAN FD Protocol module increment to the tail of the FIFO and update CxFIFOUAx.

Now the application can read the next message from the RX FIFO.

Table 14-12. Receive Message Object

Words	Bits	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
R0	31:24	—	—	SID11	EID[17:6]				
	23:16	EID[12:5]							
	15:8	EID[4:0]					SID[10:8]		
	7:0	SID[7:0]							
R1	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	FILHIT[4:0]					—	—	ESI
	7:0	FDF	BRS	RTR	IDE	DLC[3:0]			
R2	31:24	RXMSGTS[31:24]							
	23:16	RXMSGTS[23:16]							
	15:8	RXMSGTS[15:8]							
	7:0	RXMSGTS[7:0]							
R3	31:24	Receive Buffer Data Byte 3							
	23:16	Receive Buffer Data Byte 2							
	15:8	Receive Buffer Data Byte 1							
	7:0	Receive Buffer Data Byte 0							
R4	31:24	Receive Buffer Data Byte 7							
	23:16	Receive Buffer Data Byte 6							
	15:8	Receive Buffer Data Byte 5							
	7:0	Receive Buffer Data Byte 4							
Ri	31:24	Receive Buffer Data Byte n							
	23:16	Receive Buffer Data Byte n-1							
	15:8	Receive Buffer Data Byte n-2							
	7:0	Receive Buffer Data Byte n-3							

bit R0.31-30 **Unimplemented:** Read as 'x'

bit R0.29 **SID11:** In FD mode, the standard ID can be extended to 12 bit using r1.

bit R0.28-11 **EID[17:0]:** Extended Identifier

bit R0.10-0 **SID[10:0]:** Standard Identifier

bit R1.31-16 **Unimplemented:** Read as 'x'

bit R1.15-11 **FILHIT[4:0]:** Filter Hit, number of filter that matched

bit R1.10-9 **Unimplemented:** Read as 'x'

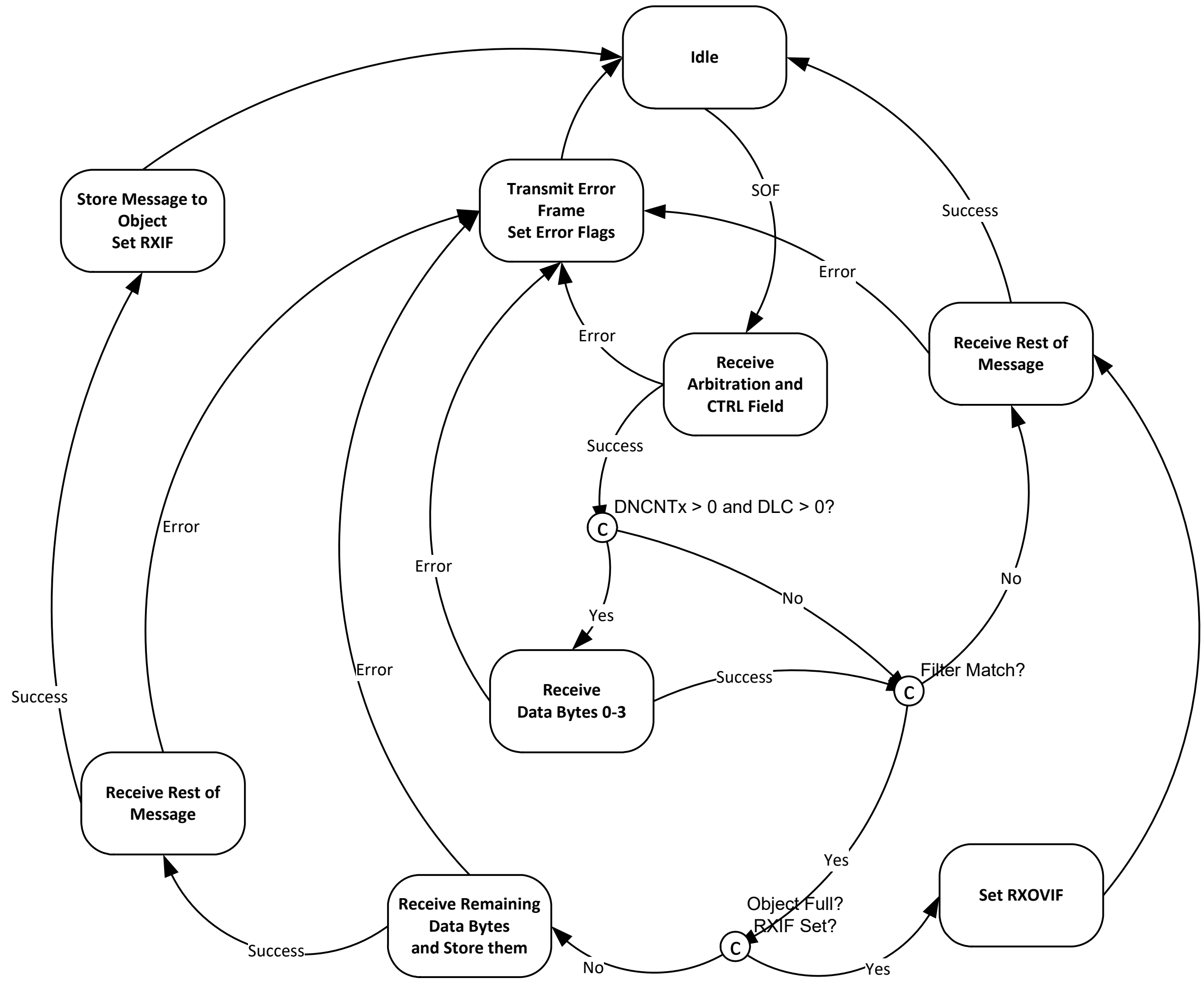
bit R1.8	**ESI:** Error Status Indicator 1 = Transmitting node is error passive. 0 = Transmitting node is error active.
bit R1.7	**FDF:** FD Frame; distinguishes between CAN and CAN FD formats.
bit R1.6	**BRS:** Bit Rate Switch; selects if data bit rate is switched.
bit R1.5	**RTR:** Remote Transmission Request; not used in CAN FD.
bit R1.4	**IDE:** Identifier Extension Flag; distinguishes between base and extended format.
bit R1.3-0	**DLC[3:0]:** Data Length Code
bit R2.31-0	**RXMSGTS[32:0]:** Receive Message Timestamp

14.10.2. Receive State Diagram

Figure 14-19 illustrates how messages are received. It illustrates how the most important receive flags are set and cleared.

- The CAN FD Protocol module remains Idle until a SOF is detected.
- After a SOF is detected, the module will receive the arbitration and control fields.
- Based on the DNCNTx bits and the received DLC, acceptance filtering will start. See Figure 14-16 for more details.
- If none of the filters match, the message will still be received, but it will not be stored.
- If a filter matches, the device checks whether the receive object the filter points to is full.
- If the receive object is full, the RXOVIF bit will be set.
- If the receive object is not full, the rest of the data bytes are received and stored to the receive object.
- If a complete message is received, the message will be stored, a time-stamp will be attached and the receive flags will be set; the FIFO status flags will be updated and the FIFO head will be incremented.
- In case an error is detected during the reception of a message, an error frame will be transmitted and the appropriate error flags will be set.

Figure 14-19. Receive State Diagram



14.10.3. Resetting RX FIFO

A receive FIFO can be reset by:

- Setting FRESET (CxFIFOCONx[10]) or
- Placing the module in Configuration mode (OPMOD[2:0] = `100`).

Resetting the FIFO will reset the head and tail pointers and the CxFIFOSTAx register. The settings in the CxFIFOCONx registers will not change.

Before resetting an RX FIFO using FRESET, ensure that no enabled filter is pointing to the FIFO.

14.10.4. Mismatch of DLC and Payload Size During Reception

The PLSIZEx bits reserve a certain number of bytes in the receive message object. The module handles mismatches between DLC and payload size as follows:

- If the number of bytes specified by the DLC is smaller than the number of bytes specified by the PLSIZEx bits, the received message bytes will be stored in the message object without any padding.
- If the number of bytes specified by the DLC is bigger than the number of bytes specified by the PLSIZEx bits, the data bytes that fit in the receive message object are stored and the other data bytes that do not fit are discarded. The module ensures that the next message object in RAM does not get overwritten. The module will store the message in the receive object, and

the RX FIFO status flags will be updated. In addition, the IVMIF (CxINT[15]) and DLCMM flags (CxBDIAG1[31]) will be set.

14.10.5. Message Reception Code Example

A code example to receive the CAN FD extended frame using Filter 0, and saving the messages in FIFO 1, is shown in Example 14-3.

Example 14-3. Message Reception Code

```
#include <xc.h>
/* This code example demonstrates a method to configure the CAN FD module to
receive the extended ID CAN FD messages. This uses CAN1, FIFO1 and filter 0.
FIFO1 is configured to receive 2 messages. */
/* Include fuse configuration code here. */
#define MAX_WORDS 100
unsigned int __attribute__((aligned(4)))CanRxBuffer[MAX_WORDS];
/*data structure to implement a CANFD message buffer. */
/* CANFD Message Time Stamp */
typedef unsigned long CANFD_MSG_TIMESTAMP;

/* CANFD RX Message Object Control*/
typedef struct _CANFD_RX_MSGOBJ_CTRL {
    unsigned DLC : 4;
    unsigned IDE : 1;
    unsigned RTR : 1;
    unsigned BRS : 1;
    unsigned FDF : 1;
    unsigned ESI : 1;
    unsigned unimplemented1 : 2;
    unsigned FilterHit : 5;
    unsigned unimplemented2 : 16;
} CANFD_RX_MSGOBJ_CTRL; /* CANFD RX Message ID*/

typedef struct _CANFD_MSGOBJ_ID {
    unsigned SID : 11;
    unsigned long EID : 18;
    unsigned SID11 : 1;
    unsigned unimplemented1 : 2;
} CANFD_MSGOBJ_ID;

/* CANFD RX Message Object */
typedef union _CANFD_RX_MSGOBJ {
    struct {
        CANFD_MSGOBJ_ID id;
        CANFD_RX_MSGOBJ_CTRL ctrl;
        CANFD_MSG_TIMESTAMP timeStamp;
    } bF;
    unsigned int word[6];
    unsigned char byte [12]
} CANFD_RX_MSGOBJ;

int main(void) {
    /* The dsPIC33A device features I/O remap. This I/O remap configuration for
    the CAN FD module can be performed here. */
    SetIORemapForCANFDModule();

    /* Configure the CRU clock generator unit corresponding to CANFD to yield
     * 40MHz clock. This clock is CANFD module clock */
    ConfigureCANFDClockFor40MHz(); // FCAN = 40 MHz

    /* Enable the CANFD module */
    C1CONbits.ON = 1;

    /* Place CAN module in configuration mode */
    C1CONbits.REQOP = 4;
    while (C1CONbits.OPMOD != 4);

    /* Initialize the C1FIFOBA with the start address of the CAN FIFO message
    buffer area. */
    C1FIFOBA = (unsigned int) &CanRxBuffer;

    /* Set up the CANFD module for 1Mbps Nominal bit rate speed and
     * 2Mbps Data bit rate.
     */
```

```
    C1NBTCFG = 0x001E0707;
    C1DBTCFG = 0x000E0303;
    C1TDC = 0x00020F00;             //TDCMOD is Auto

    /* Configure CANFD module to enable BRS */
    C1CONbits.BRSDIS = 0x0;
    C1CONbits.STEF = 0x0;           //Don't save transmitted messages in TEF
    C1CONbits.TXQEN = 0x0;          //No TXQ

    /* Configure FIFO1 to Receive 2 messages*/
    C1FIFOCON1bits.FSIZE = 0x1;     //2 messages
    C1FIFOCON1bits.PLSIZE = 0x7;    //64 bytes of data
    C1FIFOCON1bits.TXEN = 0x0;      //Receive fifo
    C1FIFOCON1bits.RXTSEN = 0x1;    //Enable receive fifo timestamp

    /* Configure filter 0 and MASK 0 to accept extended id messages
    with id = 2 and 3 */
    C1FLTCON0bits.F0BP = 1; // message stored in FIFO1
    C1FLTOBJ0 = 0x40001000; // EID = 0x00002 with extended identifier address
    C1MASK0 = 0xFFFFF7FF; // MEID = 0x1FFFE:Last bit is 0. Match message types
    C1FLTCON0bits.FLTEN0 = 1; // Enable the filter 0

    /* Place the CAN module in Normal mode. */
    C1CONbits.REQOP = 0;
    while (C1CONbits.OPMOD != 0);

    /* Get the address of the message buffer to read the received messages.*/
    /* set UINC bit to update the FIFO tail */
    CANFD_RX_MSGOBJ *rxObj;
    rxObj = (CANFD_RX_MSGOBJ *) C1FIFOUA1;
    while (C1FIFOSTA1bits.TFNRFNIF == 0);

    //Process the received messages
    C1FIFOCON1bits.UINC = 1;        // Update the FIFO message pointer.
    while (1);
}
```

14.11. FIFO Behavior

This section explains the FIFO behavior when TEF and TXQ are enabled. FIFO 1 is configured as a TX FIFO, and FIFO 2 as an RX FIFO. The remaining FIFOs are not configured.

Notes:

1. The start addresses are calculated based on the number of objects in the FIFO and the PLSIZEx bits.
2. The start addresses of the FIFOs given in Table 14-13 are calculated when TEF starts at 0x1400.

Table 14-13. Example FIFO Configuration

FIFO	Objects in FIFO	Payload per Object	Time-stamp	Bytes in Object	Bytes in FIFO	Start Address
TEF	12	N/A	Yes	12	144	0x1400
TXQ	8	32	N/A	40	320	0x1490
FIFO 1	5	64	N/A	72	360	0x15D0
FIFO 2	16	64	Yes	76	1216	0x1738
FIFO 3	N/A	—	—	—	—	0x1BF8

14.11.1. FIFO Status Flags

FIFO 1 through FIFO 31 can be configured as transmit or receive FIFOs. The same status flags in CxFIFOSTAx are used for transmit and receive FIFOs. The status flags behave differently based on the selected configuration.

14.11.1.1. TX FIFO Status Flags

There are three transmit status flags:

- TFEIF (**TFE**RFF**IF**): Transmit FIFO Empty Interrupt Flag; set when the FIFO is empty.
- TFHIF (**TFH**RFH**IF**): Transmit FIFO Half Empty Interrupt Flag; set when FIFO is less than half-full.
- TFNIF (**TFN**RFN**IF**): Transmit FIFO Not Full Interrupt Flag; set when FIFO is not full.

The status flags of a transmit FIFO are set when there is space to load a new message object into the FIFO. Before the first message object is loaded (after the FIFO is reset), all status flags are set. When the FIFO is fully loaded, all flags are cleared.

14.11.1.2. RX FIFO Status Flags

There are three receive status flags:

- RFFIF (TFE**RFFIF**): Receive FIFO Full Interrupt Flag; set when the FIFO is full.
- RFHIF (TFH**RFHIF**): Receive FIFO Half Full Interrupt Flag; set when the FIFO is at least half full.
- RFNIF (TFN**RFNIF**): Receive FIFO Not Empty Interrupt Flag; set when there is at least one message in the FIFO.

The status flags of the receive FIFO are set when there are received messages in the FIFO. Before the first message is received (after the FIFO is reset), all status flags are cleared. When the FIFO is full, all flags are set.

14.11.1.3. TXQ Status Flags

There are two TXQ status flags:

- TXQEIF: TXQ Empty Interrupt Flag; set when the TXQ is empty.
- TXQNIF: TXQ Not Full Interrupt Flag; set when TXQ is not full.

The status flags of the TXQ are set when there is space to load a new message object into the TXQ. Before the first message object is loaded (after the TXQ is reset), all status flags are set. When the TXQ is fully loaded, all flags are cleared.

14.11.1.4. TEF Status Flags

There are four TEF status flags:

- TEFFIF: TEF Full Interrupt Flag; set when the TEF is full.
- TEFHIF: TEF Half Full Interrupt Flag; set when the TEF is at least half-full.
- TEFNEIF: TEF Not Empty Interrupt Flag; set when there is at least one message in the TEF.
- TEFOVIF: TEF Overrun Interrupt Flag; set when an overflow has occurred.

The status flags of the TEF are set when there are transmitted messages in the FIFO. Before the first message is stored (after the TEF is reset), all status flags are cleared. When the TEF is full, all flags are set.

14.11.2. Transmit FIFO Behavior

FIFO 1 is configured as a TX FIFO. CxFIFOCON1 is used to control the FIFO. CxFIFOSTA1 contains the status flags and the FIFO Index bits (FIFOCI[4:0]). CxFIFOUA1 contains the user address of the next transmit message object to be loaded.

Figure 14-20 through Figure 14-25 illustrate how the status flags, user address and FIFO index are updated for FIFO 1.

Figure 14-20 shows the status of FIFO 1 after the Reset. Message objects, MO0 to MO4, are empty. All status flags are set. The user address and the FIFO index point to MO0.

Figure 14-20. FIFO 1 – Initial State

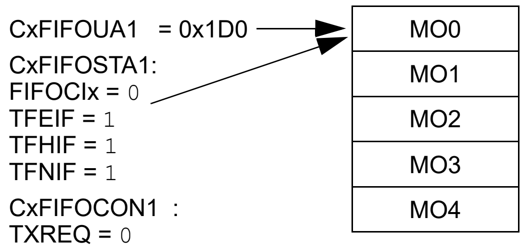


Figure 14-21 illustrates the status of FIFO 1 after the first message (MSG0) is loaded. MO0 now contains MSG0. The user application sets the UINC bit (CxFIFOCON1[8]), which causes the FIFO head to advance. The user address now points to MO1. TFEIF is cleared since the FIFO is no longer empty. The user application now sets TXREQ to request the transmission of MSG0.

Figure 14-21. FIFO 1 – First Message Loaded

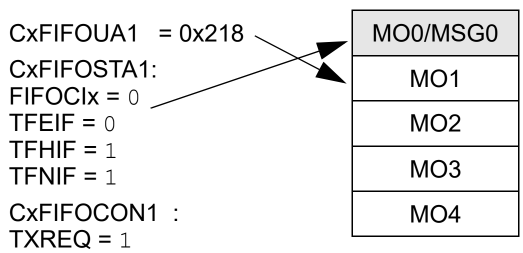


Figure 14-22 illustrates the status of FIFO 1 after MSG0 is transmitted. The FIFO is empty again. TFEIF is set and TXREQ is cleared. FIFOCIx bits now point to MO1 with user address 0x218.

Figure 14-22. FIFO 1 – First Message Transmitted

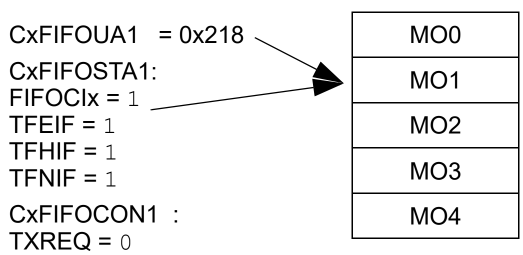


Figure 14-23 illustrates the status of FIFO 1 after three more messages are loaded: MSG1-MSG3. The user address now points to MO4. TFHIF is cleared because the FIFO is now less than half empty.

Figure 14-23. FIFO 1 – Three More Messages Loaded

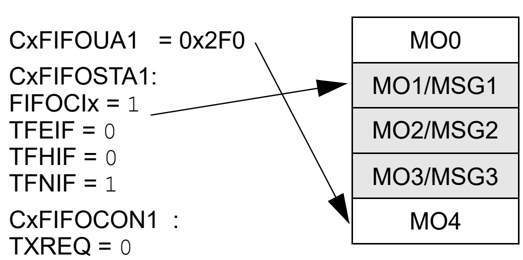


Figure 14-24 illustrates the status of FIFO 1 after two more messages are loaded: MSG4 and MSG5. CxFIFOUA1 now points to MO1. All status flags are now cleared because the FIFO is full. The user address and the FIFO index now point to MO1. The user application now sets TXREQ to request the transmission of MSG1-MSG5.

Figure 14-24. FIFO 1 – FIFO Fully Loaded

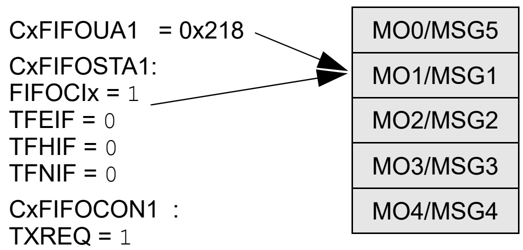
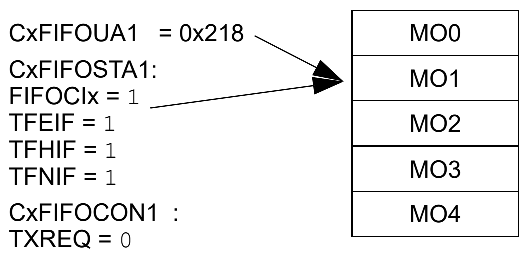


Figure 14-25 illustrates the status of FIFO 1 after MSG1-MSG5 are transmitted. The FIFO is empty again. All status flags are set, and TXREQ is cleared. The user address and the FIFO index point to MO1 again.

Figure 14-25. FIFO 1 – FIFO Fully Transmitted



14.11.3. Receive FIFO Behavior

FIFO 2 is configured as an RX FIFO. CxFIFOCON2 is used to control the FIFO. CxFIFOSTA2 contains the status flags and the FIFO index (FIFOCIx). CxFIFOUA2 contains the user address of the next message object to read.

Figure 14-26 through Figure 14-33 illustrate how the status flags, user address and FIFO index are updated.

Figure 14-26 shows the status of FIFO 2 after the Reset. Message objects, MO0 to MO15, are empty. All status flags are cleared. The user address and the FIFO index point to MO0.

Figure 14-26. FIFO 2 – Initial State

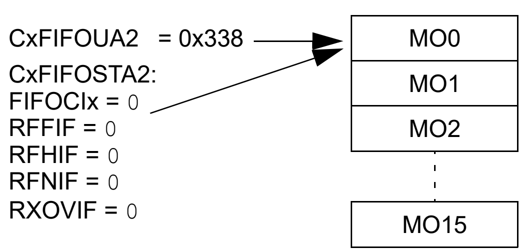


Figure 14-27 illustrates the status of FIFO 2 after the first message (MSG0) is received. MO0 now contains MSG0. The FIFO index now points to MO1. RFNIF is set since the FIFO is not empty anymore.

Figure 14-27. FIFO 2 – First Message Received

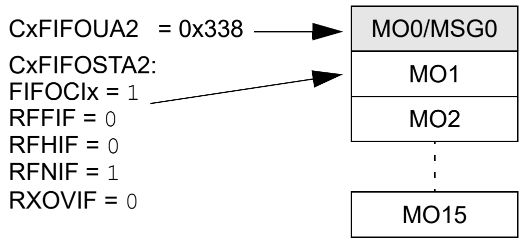


Figure 14-28 illustrates the status of FIFO 2 after MSG0 is read. The user application reads the message from RAM and sets the UINC bit (CxFIFOCON2[8]). The user address increments and points to MO1. The FIFO index is unchanged. The FIFO is empty again. All flags are cleared.

Figure 14-28. FIFO 2 – First Message Read

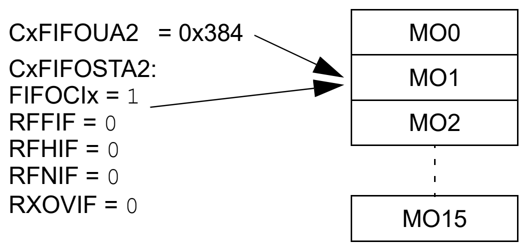


Figure 14-29 illustrates the status of FIFO 2 after eight more messages are received: MSG1-MSG8. The user address still points to MO1. RFNIF and RFHIF are set because the FIFO is now half full. The FIFO index points to MO9.

Figure 14-29. FIFO 2 – Half Full

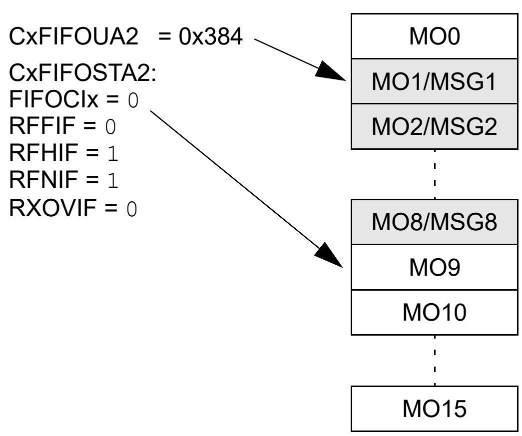


Figure 14-30 illustrates the status of FIFO 2 after 10 more messages are received: MSG5-MSG15. The user address still points to MO1. The FIFO index points to MO0. RFNIF and RFHIF are set.

Figure 14-30. FIFO 2 – FIFO Almost Full

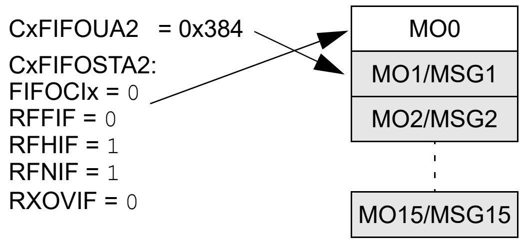


Figure 14-31 illustrates the status of FIFO 2 after one more message is received: MSG16. All status flags are set because the FIFO is full. The user address and the FIFO index point to MO1.

Figure 14-31. FIFO 2 – FIFO Full

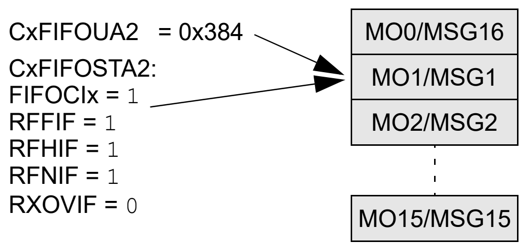


Figure 14-32 illustrates the status of FIFO 2 after one more message is received. Since FIFO 2 is already full, an overflow occurs. The message is discarded, and RXOVIF is set. The user address and FIFO index have not changed.

Figure 14-32. FIFO 2 – FIFO Overflow

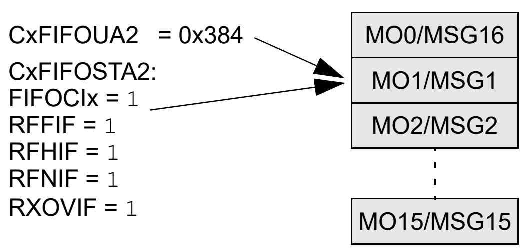
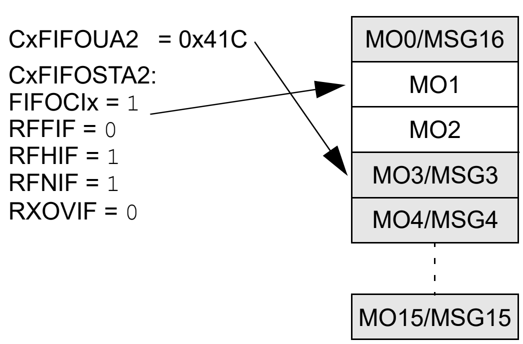


Figure 14-33 illustrates the status of FIFO 2 after the application cleared RXOVIF and read two more messages. RFFIF is clear because the FIFO is not full anymore. The user address points to MO3. The FIFO index has not changed.

Figure 14-33. FIFO 2 – Two More Messages Read



14.11.4. Transmit Queue Behavior

CxTXQCON is used to control the TXQ. CxTXQSTA contains the status flags and the TXQ index (TXQCIx). CxTXQUA contains the user address of the next transmit message object to be loaded.

The TXQCI[4:0] bits are used by the CAN FD Protocol Module to calculate the next message to transmit. TXQCIx bits are not incremented linearly. They are recalculated every time a message gets transmitted or TXREQ gets set.

Figure 14-34 through Figure 14-39 illustrate how the status flags and user address are updated. There is no need for the user application to use TXQCIx; therefore, it is not shown in the figures.

Figure 14-34 shows the status of the TXQ after Reset. Message objects, MO0 to MO7, are empty. All status flags are set. The user address points to MO0.

Figure 14-34. TXQ – Initial State

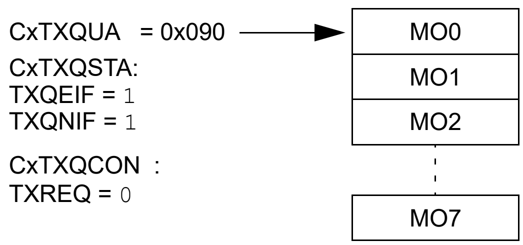


Figure 14-35 illustrates the status of the TXQ after the first message (MSG0) is loaded. MO0 now contains MSG0. The user application sets the UINC bit, which causes the FIFO head to advance. The user address now points to MO1. TXQEIF is cleared, since the queue is not empty anymore. The user application now sets TXREQ to request the transmission of MSG0.

Figure 14-35. TXQ – First Message Loaded

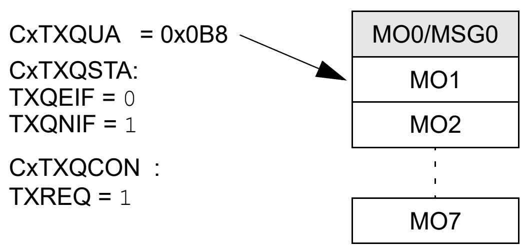


Figure 14-36 illustrates the status of the TXQ after MSG0 is transmitted. The TXQ is empty again. TXQEIF is set, and TXREQ is cleared. The user address still points to MO1 because UINC is not set.

Figure 14-36. TXQ – First Message Transmitted

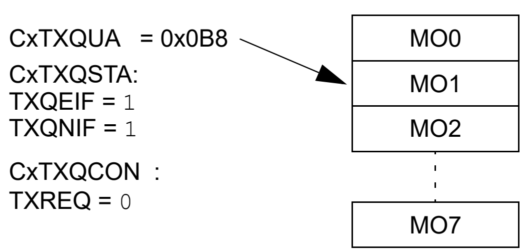


Figure 14-37 illustrates the status of the TXQ after MSG1 is loaded and UINC is set. The user address now points to the next free message object: MO0.

Figure 14-37. TXQ – Next Message Loaded

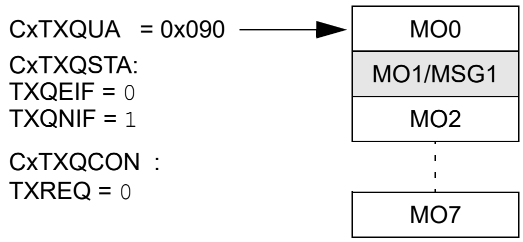


Figure 14-38 illustrates the status of the TXQ after six more messages are loaded: MSG2-MSG7. The user address now points to the last free message object: MO7.

Figure 14-38. TXQ – Next Six Messages Loaded

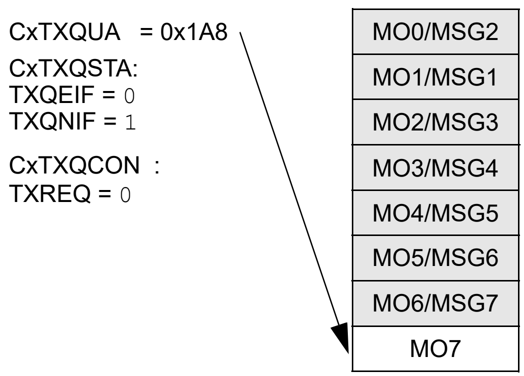
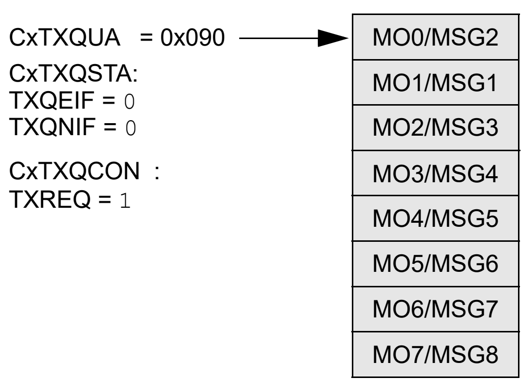


Figure 14-39 illustrates the status of the TXQ after MSG8 is loaded and UINC is set. The TXQ is now full, and all flags are cleared. The user address now points to MO0. The user application now sets TXREQ. The messages will be transmitted based on the priority of their IDs.

Figure 14-39. TXQ – Full



14.11.5. Transmit Event FIFO Behavior

CxTEFCON is used to control the TEF. CxTEFSTA contains the status flags. CxTEFUA contains the user address of the next message object to read.

The actual RAM address is calculated using Equation 14-25.

Figure 14-40 through Figure 14-47 illustrate how the status flags and user address are updated. The TEF stores transmitted messages; therefore, the flags behave similarly to an RX FIFO.

Figure 14-40 shows the status of the TEF after the Reset. Message objects, MO0 to MO11, are empty. All status flags are cleared. The user address points to MO0.

Figure 14-40. TEF – Initial State

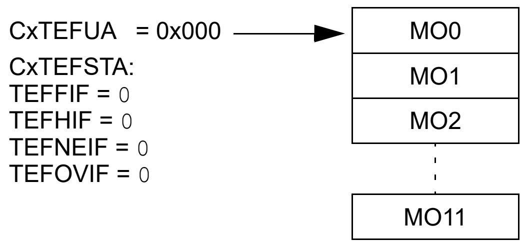


Figure 14-41 shows the status of the TEF after the first transmit message is stored. MO0 contains ID0, the ID of MSG0. TEFNEIF is set since the TEF is not empty. The user address points to MO0.

Figure 14-41. TEF – First Transmit Message is Stored

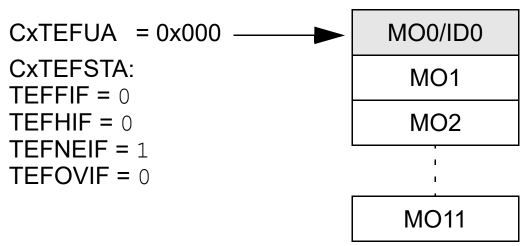


Figure 14-42 illustrates the status of the TEF after ID0 is read. The user application reads the ID from RAM and sets the UINC bit (CxTEFCON[8]). The user address increments and points to MO1. The TEF is empty again. All flags are cleared.

Figure 14-42. TEF – First ID Read

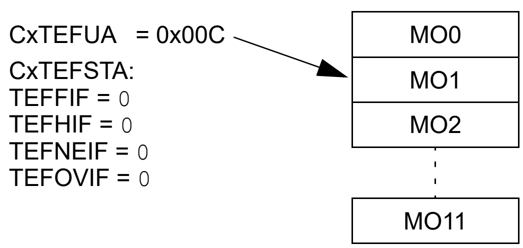


Figure 14-43 illustrates the status of the TEF after six more messages are transmitted: MSG1-MSG6. The user address points to MO1. TEFNEIF and TEFHIF are set because the TEF is now half full.

Figure 14-43. TEF – Half Full

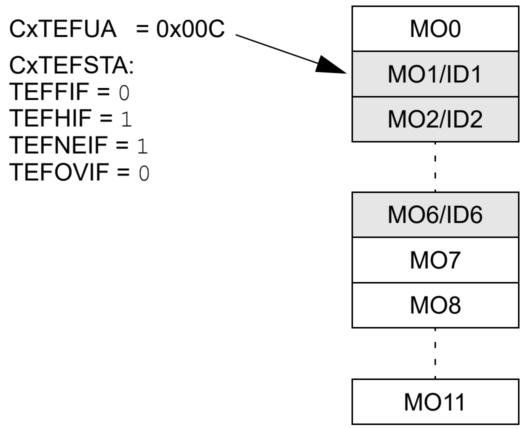


Figure 14-44 illustrates the status of the TEF after five more messages are transmitted: MSG7-MSG11. The user address still points to MO1. TEFNEIF and TEFHIF are set.

Figure 14-44. TEF – Almost Full

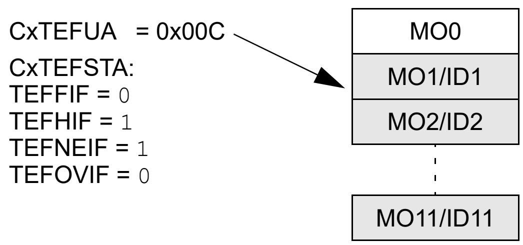


Figure 14-45 illustrates the status of the TEF after one more message is transmitted: MSG12. All status flags are set because the TEF is full. The user address points to MO1.

Figure 14-45. TEF – Full

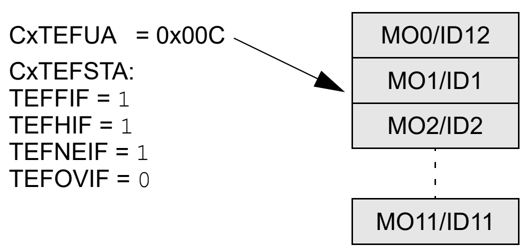


Figure 14-46 illustrates the status of the TEF after one more message is transmitted. Since the TEF is already full, an overflow occurs. The ID is discarded, and TEFOVIF is set. The user address remains unchanged.

Figure 14-46. TEF – Overflow

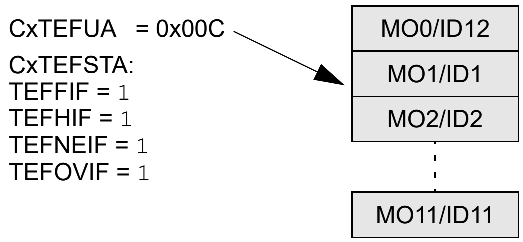
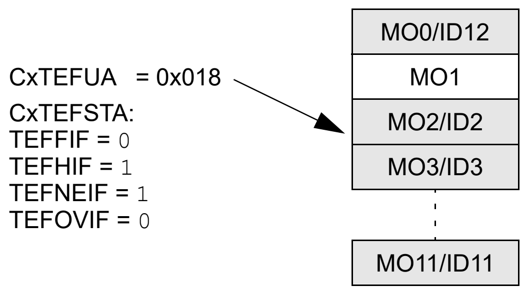


Figure 14-47 illustrates the status of the TEF after the application cleared TEFOVIF and read one more message. TEFFIF is clear because the TEF is not full anymore. The user address points to MO2.

Figure 14-47. TEF – One More ID Read



14.12. Timestamping

The CAN FD Protocol Module contains a Time Base Counter (TBC). The TBC is a 32-bit free-running counter that increments on multiples of Fcy and rolls over to zero when:

- TBCPRE[9:0] bits (CxTSCON[9:0]) are used to configure the prescaler for the TBC.
- Setting TBCEN (CxTSCON[16]) enables the TBC.
- Clearing TBCEN disables, stops and resets the TBC.
- The TBC has to be disabled before writing to CxTBC by clearing TBCEN.
- TEFTSEN (CxTEFCON[5]) has to be set to time-stamp messages in the TEF.
- RXTSEN (CxFIFOCONx[5]) has to be set to time-stamp messages in the individual RX FIFO.
- The application can read CxTBC at any time. Similar to any multibyte counter, the application has to consider that the counter increments and might roll over while reading different bytes of the counter.

All timestamps are 32 bits, allowing timestamps to be used for system time synchronization with high resolution.

A rollover of the TBC will generate an interrupt if TBCIE is set.

Messages can be time-stamped either at the beginning of a frame or at the end, depending on the TSEOF bit (CxTSCON[17]). When TSEOF = 0, TSRES (CxTSCON[18]) specifies if FD frames are time-stamped at SOF or the reserved bit. Table 14-14 specifies the reference points when the timestamping occurs. At the reference point, the value of the TBC (CxTBC) is captured and stored into the message object:

- Receive Message Object: The TBC value is stored in the RXMSGTSx bits (see Receive Message Object).
- TEF Object: The TBC value is stored in the TXMSGTSx bits (see Reading a TEF Object).

Table 14-14. Reference Point

Frame	CAN 2.0	CAN FD
Start of TX	Sample point of SOF	Sample point of SOF or the bit after FDF
Start of RX	Sample point of SOF	Sample point of SOF or the bit after FDF
Valid TX	No error till end of EOF	No error till end of EOF
Valid RX	No error till the last, but one bit of EOF	No error till the last, but one bit of EOF

14.13. Interrupts

Interrupts can be classified into multiple layers. Lower layer interrupts propagate to higher layers by multiplexing them into single interrupts. Figure 14-48 illustrates the layers of interrupts.

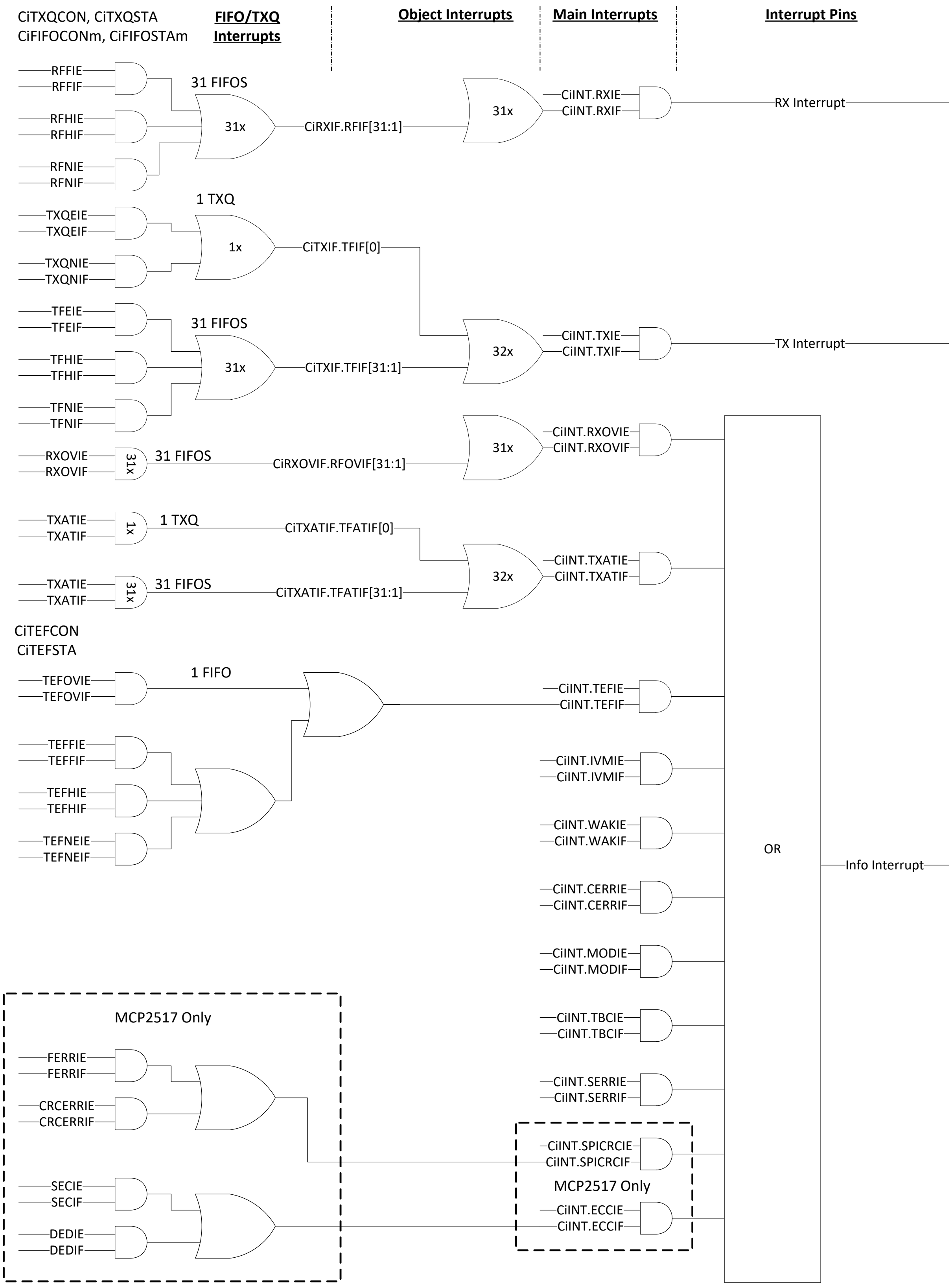
- FIFO Individual Interrupt
- FIFO Combined Interrupt
- Main Interrupt

These interrupts are then funneled into three separate module interrupts:

- Receive Interrupt
- Transmit Interrupt
- Information Interrupt

All module interrupts are persistent, meaning the condition that caused the interrupt must be cleared within the module for the interrupt request to be removed.

Figure 14-48. Interrupt Multiplexing



14.13.1. FIFO Individual Interrupts

CxFIFOCONx contains the interrupt enable flags, and CxFIFOSTAx contains the interrupt flags for the FIFOs. There is a separate register for each FIFO.

14.13.1.1. Transmit Queue Interrupts

CxTXQCON contains the interrupt enable flags, and CxTXQSTA contains the interrupt flags for the TXQ.

The TXQ interrupt occurs when there is a change in the status of the TXQ. There are two interrupt sources:

- TXQ Not Full Interrupt Flag (TXQNIF)
- TXQ Empty Interrupt Flag (TXQEIF)

Both interrupts can be enabled individually. The interrupts cannot be cleared by the application; they will be cleared when the condition of the FIFO terminates.

Both interrupt sources are OR'd together and reflected in the TFIF flag (CxTXIF[31:0]).

14.13.1.2. Receive FIFO Interrupts - RFIF

The receive FIFO interrupts occur when there is a change in the status of the receive FIFO. There are three interrupt sources:

- Receive FIFO Full Interrupt Flag (RFFIF)
- Receive FIFO Half Full Interrupt Flag (RFHIF)
- Receive FIFO Not Empty Interrupt Flag (RFNIF)

All three interrupts can be enabled individually. The interrupts cannot be cleared by the application; they will be cleared when the condition of the FIFO terminates.

The three interrupt sources are OR'd together and reflected in the RFIF[30:0] (CxRXIF[31:1]) flags.

14.13.1.3. Transmit FIFO Interrupts - TFIF

The transmit FIFO interrupts occur when there is a change in the status of the transmit FIFO. There are three interrupt sources:

- Transmit FIFO Not Full Interrupt Flag (TFNIF)
- Transmit FIFO Half Empty Interrupt Flag (TFHIF)
- Transmit FIFO Empty Interrupt Flag (TFEIF)

All three interrupts can be enabled individually. The interrupts cannot be cleared by the application; they will be cleared when the condition of the FIFO terminates.

The three interrupt sources are OR'd together and reflected in the CxTXIF[31:1] flags.

14.13.1.4. Receive FIFO Overrun Interrupt - RXOVIF

When a message is successfully received, but the FIFO is full, the RXOVIF of the individual FIFO is set. The flag must be cleared by the application.

14.13.1.5. Transmit FIFO Attempt Interrupt - TXATIF

When the retransmission of a message fails due to an error, and all retransmission attempts are exhausted, the TXATIF flag is set. The flag must be cleared by the application.

14.13.1.6. Transmit Event FIFO Interrupts - TEFIF

The TEF interrupts occur when there is a change in the status of the TEF. There are four interrupt sources:

- TEF Full Interrupt Flag (TEFFIF)
- TEF Half Full Interrupt Flag (TEFHIF)

- TEF Not Empty Interrupt Flag (TEFNEIF)
- TEF Overrun Interrupt Flag (TEFOVIF)

The TEF interrupts work similarly to the receive FIFO interrupts. All four interrupts can be enabled individually.

TEFFIF, TEFHIF and TEFNEIF cannot be cleared by the application; they will be cleared when the status of the FIFO terminates.

The TEFOVIF must be cleared by the application.

The four interrupt sources are OR'd together and reflected in the TEFIF flag (CxINT[4]).

14.13.2. FIFO Combined Interrupts

The following interrupts are individual FIFO interrupts:

- FIFOs/TXQ: RFIFx, TFIFx, RFOVIFx and TFATIFx

They are combined into single Interrupt Status registers:

- CxRXIF, CxTXIF, CxRXOVIF and CxTXATIF

The bits in the status registers are mapped to the FIFOs as follows: Bit 0 to TXQ, Bit 1 to FIFO 1, Bit 2 to FIFO 2 and up to Bit 31 to FIFO 31. Since Bit 0 corresponds to the TXQ, Bit 0 of CxRXIF and CxRXOVIF is reserved. Hence, by reading one register, the application can check the status of all FIFOs for a particular interrupt (e.g., any RFIFx pending).

The FIFO interrupts are enabled in CxFIFOCONx.

TXQ interrupts are enabled in CxTXQCON.

Clearing of the FIFO interrupts is explained in FIFO Individual Interrupts.

14.13.3. Main Interrupts

The CxINT register contains all the main interrupts. The following interrupts are a logical 'OR' of all combined FIFO interrupts: RXIF, TXIF, RXOVIF and TXATIF. These flags are read-only and must be cleared in preceding hierarchies.

The TEFIF is generated in the TEF. This flag is read-only and must be cleared in preceding hierarchies.

All interrupts in CxINT can be enabled individually.

14.13.3.1. Invalid Message Interrupt - IVMIF

If a CAN bus error or DLC mismatch is detected during the last message transmitted or received, the IVMIF bit will be set. The CxBDIAG1 register sets a flag for each error. The flag must be cleared by the application.

The following CAN bus errors will trigger the interrupt in case an error frame is transmitted: CRC, stuff bit, form, bit or ACK.

The flag will not be set if the ESI of a received message is set.

14.13.3.2. Wake-Up Interrupt - WAKIF

This bit is set if bus activity has been detected while the module is in Sleep mode. The flag must be cleared by the application.

14.13.3.3. CAN Bus Error Interrupt - CERRIF

The CxTREC registers will count the errors during transmit and receive according to the ISO11898-1:2015. The CERRIF flag will be set based on the error counter values. The flag must be cleared by the application.

CERRIF will be set each time a threshold in the TEC/REC counter is crossed by the following conditions:

- TEC or REC exceeds the Error Warning State threshold.
- The transmitter or receiver transitions to the Error Passive State.
- The transmitter transitions to the Bus Off State.
- The transmitter or receiver transitions from the Error Passive to Error Active State.
- The module transitions from the Bus Off to Error Active State after the bus off recovery sequence.

When the user clears CERRIF, it will remain clear until a new counter crossing occurs.

14.13.3.4. CAN Mode Change Interrupt - MODIF

When the OPMOD[2:0] bits change, the MODIF flag will be set. The flag must be cleared by the application.

14.13.3.5. CAN Timer Interrupt - TBCIF

When the time base counter rolls over, TBCIF will be set. The flag must be cleared by the application.

14.13.3.6. System Error Interrupt - SERRIF

- Bus Bandwidth Error:
  Bandwidth errors can happen during receive and transmit.

  Receive Message Assembly Buffer (RX MAB) overflow occurs when the module is unable to write a received CAN message to RAM before the next message arrives.

  Transmit Message Assembly Buffer (TX MAB) underflow occurs when the module cannot feed the TX MAB fast enough to provide consistent data to the bit stream processor.

  The SERRIF flag will be set, and the ICODE[6:0] bits (CxVEC[6:0]) will be set to `100 0101`.
- Handling of RX MAB Overflow Errors:
  RX MAB overflows are not acceptable for some applications. To prevent overflows, frame filtering and data saving start as early as possible; the latest at the beginning of the CRC field of the received message. Updating the FIFO status has to wait until the beginning of the seventh bit of the EOF field, since the received frame is only valid at this point. The complete message has to be saved and the FIFO has to be updated until the end of the arbitration field of the next message.

  In case of an RX MAB overflow, the new message that caused the overflow will be discarded. The module continues to store the message that is completely received and filtered. Afterwards, the module will be able to receive new messages on the bus. The application will be notified using the SERRIF bit.

  The SERRIF bit (CxINT[12]) will be cleared by writing a zero to the bit. This will also clear the SERRIF condition from the ICODEx bits.
- Handling of TX MAB Underflow Errors:
  ISO11898-1:2015 requires MAC data consistency: a transmitted message must contain consistent data. If data errors occur due to ECC errors, or TX MAB underflow, the transmission will not start. If the transmission is in progress, it will stop and the module will transition to either Restricted Operation or Listen Only mode, which is selectable using the SERR2LOM bit (CxCON[18]).

The module handles these errors by stopping the transmission and transitioning to Restricted Operation or Listen Only mode. The CxTX pin will be forced high. Additionally, all TXREQs will be ignored. The application will be notified using SERRIF. The module will continue to receive messages.

Note: There are two types of addressing errors, and both of them will cause a soft trap error on a dsPIC33A device by setting the CAN bit in the INTCON3 register.

The first addressing error occurs when a FIFO is configured with an invalid address. This error most commonly occurs when the FIFO points to an unimplemented address.

The second addressing error commonly occurs when the message destination is illegal; for example, attempting to write a received message to a program Flash, which is not directly writable.

14.13.4. Interrupt Handling

The CAN FD Protocol module allows the application to handle interrupts efficiently by:

- Implementing a lookup table using the CxVEC registers.
- Using the status registers and deciding which interrupt to service first.

The application can also use a combination of these two methods to handle interrupts.

14.13.4.1. Interrupt Lookup Table

The ICODEx and FILHITx bits in the CxVEC register enable the application to use a lookup table to implement the Interrupt Service Routine (ISR).

The following bit fields allow the application to make full use of the three interrupt pins:

- TXCODE[6:0] bits: Reflect which object has a transmit interrupt pending
- RXCODE[6:0] bits: Reflect which object has a receive interrupt pending

A separate lookup table can be implemented for transmit and receive interrupts.

If more than one object has a pending interrupt, the interrupt or FIFO with the highest number will show up in RXCODEx, TXCODEx and ICODEx. Once the interrupt with the highest priority is cleared, the next highest priority interrupt will show up in CxVEC. RXCODEx, TXCODEx and ICODEx are implemented with combinatorial logic using the interrupt flags as inputs.

14.13.4.2. Interrupt Status Registers

The CAN FD Protocol module contains 31 FIFOs and a TXQ. It would be complex to use the ICODEx bits since the interrupt priorities are determined by the module. Therefore, the following measures are taken to ensure efficient servicing of interrupts:

- CxINT and CxINT contain all main interrupt sources. The application can identify the categories of interrupts that are pending and decide the order in which interrupts are to be serviced (e.g., RXIF).
- All categories of interrupts for all FIFOs are combined into individual registers: CxRXIF, CxTXIF, CxRXOVIF and CxTXATIF. The application can identify the RFIFx bits that are pending by reading only one register. The same is true for TFIFx, RXOVIF and TXATIF.
- In the register map, the Interrupt Status registers are arranged in a single block: CxVEC, followed by CxINT, CxRXIF, CxTXIF, CxRXOVIF and CxTXATIF. This arrangement allows all status registers to be read with a single read access.

14.13.5. Interrupt Flags

Table 14-15 summarizes all interrupt flags and lists how interrupts are cleared.

Table 14-15. Interrupt Flags

Flags	Registers	Categories	Cleared by Module(1)	Cleared by Application	Read-Only(2)	Description
RFFIF, RFHIF, RFNIF	CxFIFOSTAx	FIFO	X	—	—	RX FIFO
TFNIF, TFHIF, TFEIF	CxFIFOSTAx	FIFO	X	—	—	TX FIFO
TXQNIF, TXQEIF	CxTXQSTA	TXQ	X	—	—	Transmit Queue
RXOVIF	CxFIFOSTAx	FIFO	—	X	—	RX Overrun
TXATIF	CxFIFOSTAx, CxTXQSTA	FIFO, TXQ	—	X	—	TX Attempt
TEFFIF, TEFHIF, TEFNEIF	CxTEFSTA	FIFO	X	—	—	TEF
TEFOVIF	CxTEFSTA	FIFO	—	X	—	TEF Overrun

Table 14-15. Interrupt Flags (continued)

Flags	Registers	Categories	Cleared by Module(1)	Cleared by Application	Read-Only(2)	Description
RFIF[31:1]	CxRXIF	Combined	—	—	X	All RX FIFOs
TFIF[31:1]	CxTXIF	Combined	—	—	X	All TX FIFOs
RFOVIF[31:1]	CxRXOVIF	Combined	—	—	X	All RX FIFO Overruns
TFATIF[31:0]	CxTXATIF	Combined	—	—	X	All TX FIFO Attempts
RXIF	CxINT	Main	—	—	X	RX
TXIF	CxINT	Main	—	—	X	TX
RXOVIF	CxINT	Main	—	—	X	RX Overrun
TXATIF	CxINT	Main	—	—	X	TX Attempt
TEFIF	CxINT	Main	—	—	X	TEF
IVMIF	CxINT	Main	—	X	—	Invalid Message
WAKIF	CxINT	Main	—	X	—	Wake-up
CERRIF	CxINT	Main	—	X	—	CAN Bus Error
MODIF	CxINT	Main	—	X	—	Mode Change
TBCIF	CxINT	Main	—	X	—	Time Base Counter
SERRIF	CxINT	Main	—	X	—	System Error
SPICRC	CxINT	Main	—	X	—	—
ECC	CxINT	Main	—	X	—	—

Notes:

1. The flags will be cleared when the condition of the FIFO terminates, initiated by the UINC bit (CxFIFOCONx[8]).
2. The flags need to be cleared in the preceding hierarchies.

14.14. Error Handling

Every CAN controller checks the messages on the bus for the following errors: bit, stuff, CRC, form and ACK errors. Whenever the controller detects an error, an error frame is transmitted that deletes the message on the bus. Error frames are always signaled using the nominal bit rate.

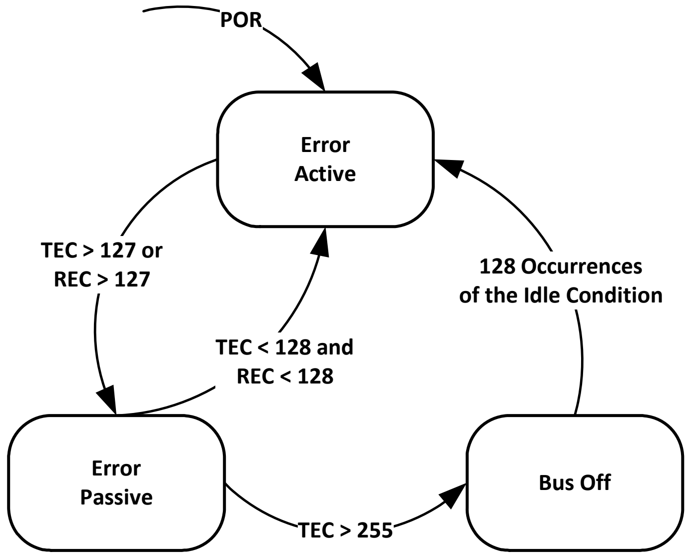
Error detection and Fault confinement are described in the ISO11898-1:2015. CxTREC contains the error counters, TEC and REC (TERRCNTx, RERRCNTx). CxTREC also contains the error warning and error state bits. TEC and REC increment and decrement according to ISO11898-1:2015 specifications.

Figure 14-49 illustrates the different error states of the CAN FD Protocol module. The module starts in the Error Active State. If the TEC or REC exceeds 127, the module transitions to the Error Passive State. If the TEC exceeds 255, the module will transition to the Bus Off State.

The module transmits active error frames when in an Error Active State. It will transmit passive error frames while in an Error Passive State. When the module is in Bus Off, the CxTX pin is always driven high and no dominant bits are transmitted.

To avoid the module from transitioning to the Error Passive State, the module will alert the application when the TEC or REC reaches 96, using the CERRIF interrupt flag (see CAN Bus Error Interrupt - CERRIF). This allows the application to take action before it enters the Error Passive State.

Figure 14-49. Error States



The Bus Diagnostic registers provide additional information about the health of the CAN bus:

- CxBDIAG0 contains separate error counters for receive/transmit and for nominal/data bit rates. The counters work differently than the counters in the CxTREC registers. They are simply incremented by one on every error. They are never decremented but can be cleared by writing '0' to the register.
- CxBDIAG1 keeps track of the kind of error that occurred since the last clearing of the register. The CxBDIAG1 register also contains the error-free message counter. The flags and the counter are cleared by writing '0' to the register.

The error-free message counter, together with the error counters and error flags, can be used to determine the quality of the bus.

14.14.1. Recovery from Bus Off State

If the TEC exceeds 255, the TXBO (CxTREC[21]) and CERRIF (CxINT[13]) bits will be set. The module will go to bus off and start the bus off recovery sequence.

The bus off recovery sequence starts automatically. The module will transition out of the Bus Off State only after the detection of 128 Idle conditions (see *"ISO11898-1:2015: Bus Off Management"*). The module will set FRESET for all transmit FIFOs when entering the Bus Off State to ensure that the module does not try to retransmit indefinitely. The application will be notified by CERRIF and has the option to queue new messages for transmission.

The module signals the exit from the Bus Off State with the CERRIF bit and by setting the TXBOERR bit (CxBDIAG1[23]). Additionally, CxTREC will be reset.

15. High-Resolution PWM with Fine-Edge Placement

This section describes the features and use of the High-Resolution Pulse-Width Modulation (PWM).

This flexible module provides features to support many types of Motor Control (MC) and Power Control (PC) applications, including:

- AC-to-DC Converters
- DC-to-DC Converters
- AC and DC Motor Control: Brushed DC, BLDC, PMSM, ACIM, SRM, Stepper, etc.
- Inverters
- Battery Chargers
- Digital Lighting
- Power Factor Correction (PFC)
- Four Independent PWM Generators, each with Dual Outputs and Fine Edge Placement
  - Resolution down to 78 ps
- Operating Modes:
  - Independent Edge PWM mode
  - Variable Phase PWM mode
  - Independent Edge PWM mode, Dual Output
  - Center-Aligned PWM mode
  - Double Update Center-Aligned PWM mode
  - Dual Edge Center-Aligned PWM mode
  - LLC mode
- Output Modes: Complementary, Independent, Push-Pull
- Dead-Time Generator
- Dead-Time Compensation
- Leading-Edge Blanking (LEB)
- Output Override for Fault Handling
- Flexible Period/Duty Cycle Updating Options
- PWM Control Inputs (PCI) for PWM Pin Overrides and External PWM Synchronization
- Advanced Triggering Options
- Combinatorial Logic Output
- PWM Event Outputs

15.1. Device-Specific Information

Table 15-1. PWM Summary Table

PWM Module Instances	Number of PWM Events	Combinatorial PWM Logic	Outputs per Instance	Peripheral Bus Speed	Clock Generator
4	2	2	2	Standard (1:2 of CPU clock)	CLKGEN5

Table 15-2. MCLKSEL PWM Master Clock Selection

Value	Description
1	CLKGEN5

Table 15-2. MCLKSEL PWM Master Clock Selection (continued)

Value	Description
0	Standard (1:2 CPU Clock)

Table 15-3. PCI Source Selection (PSS)

PSS[n] bit in register PGxyPCI2	Description
bit 31	CLC1 output
bit 30	Comparator 3 output
bit 29	Comparator 2 output
bit 28	Comparator 1 output
bit 27–26	Reserved
bit 25	PWM event C
bit 24	PWM event B
bit 23	PWM event A
bit 22	PCI22R
bit 21	PCI21R
bit 20	PCI20R
bit 19	PCI19R
bit 18	PCI18R
bit 17	PCI17R
bit 16	PCI16R
bit 15	PCI15R
bit 14	PCI14R
bit 13	PCI13R
bit 12	PCI12R
bit 11	PCI11R
bit 10	PCI10R
bit 9	PCI9R
bit 8	PCI8R
bit 7–4	Reserved
bit 3	Internally connected to Combo Trigger B
bit 2	Internally connected to Combo Trigger A
bit 1	Internally connected to the output of PWMPCI[2:0] bits in PGxEVT1 registers
bit 0	Tied to ‘0’

Table 15-4. PWM Peripheral Pin Select Mapping

Signal	Output Option of RPx Pin	Input RPx Pin Selection Bits
PWM1H	1	–
PWM1L	2	–
PWM2H	3	–
PWM2L	4	–
PWM3H	5	–
PWM3L	6	–
PWM4H	7	–
PWM4L	8	–
PCI8	–	PCI8R bits in RPINR8 register
PCI9	–	PCI9R bits in RPINR8 register

Table 15-4. PWM Peripheral Pin Select Mapping (continued)

Signal	Output Option of RPx Pin	Input RPx Pin Selection Bits
PCI10	–	PCI10R bits in RPINR8 register
PCI11	–	PCI11R bits in RPINR8 register
PCI12	–	PCI12R bits in RPINR19 register
PCI13	–	PCI13R bits in RPINR19 register
PCI14	–	PCI14R bits in RPINR19 register
PCI15	–	PCI15R bits in RPINR19 register
PCI16	–	PCI16R bits in RPINR20 register
PCI17	–	PCI17R bits in RPINR20 register
PCI18	–	PCI18R bits in RPINR20 register
PCI19	–	PCI19R bits in RPINR26 register
PCI20	–	PCI20R bits in RPINR27 register
PCI21	–	PCI21R bits in RPINR27 register
PCI22	–	PCI22R bits in RPINR27 register

15.2. High-Resolution Mode (Fine Edge Placement)

Table 15-5. High-Resolution Mode Details

Input Frequency	Resolution
300-800 MHz	up to 78 ps

The PWM Generators may operate in High-Resolution mode to enhance phase, duty cycle and dead-time resolution. A PLL internal to the PWM module locks to the PWM input clock and slices it into 16 pieces to provide high-resolution performance. High-Resolution mode cannot be used with frequency scaling or the clock divider. To enable High-Resolution mode for a given PWM Generator, set the HREN control bit (PGxCON[7]). The HRRDY status bit (PCLKCON[15]) indicates when the high-resolution circuitry is ready, and the HRERR bit (PCLKCON[14]) indicates a clocking error has occurred. When operating in high resolution, Dual PWM mode cannot be used in conjunction with Complementary Output mode.

In the event of a clocking error (HRERR = 1), the HRERR bit must be set to '0' in the software. When '0' is written to HRERR after a clocking error, the High-Resolution mode is disabled and then re-enabled. This is to re-establish the PLL lock for High-Resolution mode. After High-Resolution mode is disabled, the associated outputs will be driven to '0' until the re-enable has been successfully completed.

Note: When using High-Resolution mode, the CLKSEL[1:0] bits (PGxCON[4:3]) must be set to '01' to select pwm_master_clk directly, and the pwm_master_clk must be configured for the correct frequency. Refer to the PWMx Module Timing Requirements within Electrical Characteristics for this value.

15.2.1. High-Resolution Mode Data Registers

When High-Resolution mode is selected, some of the PWM Data registers have limited resolution. For some registers, the Least Significant bits (LSbs) of the data value are forced to '0', regardless of the value written to the registers. When configuring the PWM in High-Resolution mode, first set the HREN bit before writing to data registers whose function is dependent on High-Resolution mode. High-resolution operational differences are summarized in Table 15-6.

Table 15-6. High-Resolution Mode PWM Data Registers

Register	31:20	19:12	11:4	3	2	1	0
PGxLEB		(Note 5)		0	0	0	0
PGxPHASE							
PGxDC							
PGxDCA							
PGxPER							
PGxTRIGA	Note 1			(Note 4)			
PGxTRIGB	Note 1			(Note 4)			
PGxTRIGC	Note 1			0	0	0	0
PGxTRIGD	Note 1			0	0	0	0
PGxTRIGE	Note 1			0	0	0	0
PGxTRIGF	Note 1			(Note 4)			
PGxDT							
PGxCAP				(Note 4)			
FSCL		(Note 3)					
FSMINPER		(Note 3)					
MPHASE							
MDC							

Table 15-6. High-Resolution Mode PWM Data Registers (continued)

Register	31:20	19:12	11:4	3	2	1	0
MPER							

Notes:
1. Bit[31] of the PGxTRIGx registers selects the counter phase that produces the trigger when operating in Center-Aligned modes.
2. Bits[3:0] will read as '0' in High and Standard-Resolution modes. Bit[0] is writable.
3. Not used in High-Resolution mode.
4. The 4 LSBs of the PGxTRIGA and PGxTRIGB will be used by the high-resolution circuitry when Dual PWM mode is selected and High-Resolution mode is enabled. When high resolution is enabled, these bits will be readable and writable. However, they will only be used by the Dual PWM feature.
5. PGxLEB is a 20-bit register. The resolution of the PGxLEB register does not change in High-Resolution mode. PGxLEB[19:4] provides an adjustment of the blanking time in increments of TPWM with TPWM*5 being the minimum achievable blanking time when PGxLEB[19:4] ≤ 4.

15.2.2. High-Resolution Period Synchronization

When operating in High-Resolution mode, it is possible for PWM output edges to not be aligned with PGx_clk in which the rest of the PWM module operates. When PGxPER (or MPER) values are not divisible by eight, the period contains a fractional value of PGx_clk. This fractional clock difference can cause other events, including End-of-Cycle (EOC), triggers, etc. to not align with the output edges. The module contains an accumulator circuit to calculate and minimize the offset over long time periods.

If synchronous behavior is desired, it is recommended to use PGxPER values with bits[2:0] equal to '0'.

The fine edge placement circuit itself adds delay to the PWM outputs when compared to the base PWM signal. Using the base PWM signal for gating and synchronization in High-Resolution mode may cause unexpected results for a few fine edge clock cycles in some cases. For example, using the PCI's auto-terminate feature will remove an override condition at EOC and place the PWM outputs back to their existing state. Override is applied after the fine edge placement circuit. Since the EOC event is based on the base PWM signal, the delay through the fine edge circuit may be observed before the next PWM cycle is started. This behavior can be mitigated by using a PHASE offset equal to PGxPER – 8. In addition to EOC events, using timers operating on the base PWM signal (such as LEB and PGxTRIGy) or other PWM Generators as a source may also be susceptible in some conditions.

15.2.3. Minimum PWM Period and Pulse Width

The PWM module has some restrictions regarding minimum PW periods and pulse widths. Depending on the operating mode, the pulse width can also be dependent on PHASE and TRIGy, in addition to duty cycle. The minimum pulse width applies to both active and inactive states; 0% and 100% duty cycles are supported.

Table 15-7 shows restrictions to period and pulse width.

Table 15-7. Minimum Period and Pulse Width

Mode	Minimum Period (PGxPER or MPER)	Minimum Active Pulse Width in Counts	Minimum Inactive Pulse Width in Counts
Standard Resolution	0x0100	0x0010	Period - 0x0010
High Resolution	0x0400	0x0100	Period - 0x0100

15.2.4. Runt Pulse Indication

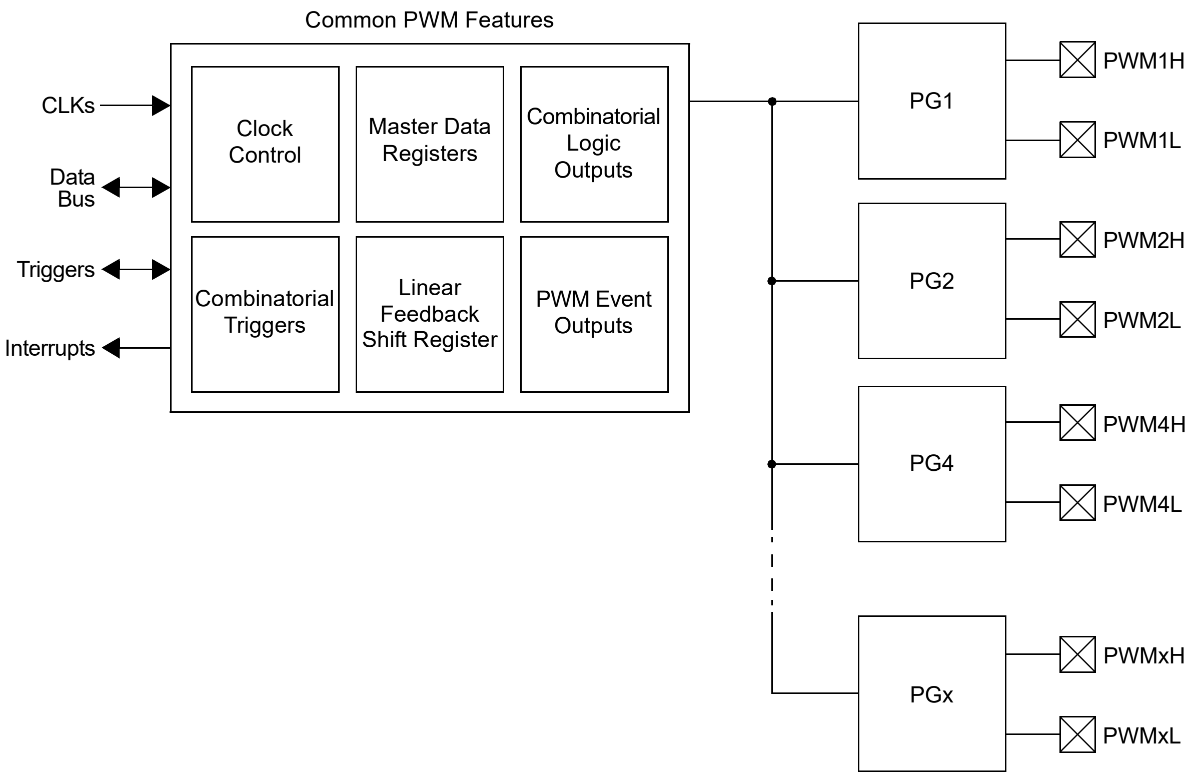
A runt pulse PWM event can be configured similarly to other PWM events discussed in PWM Event Outputs. To configure the runt pulse event, set PWMEVTy.EVTySEL[4:0] = 11010. When a runt pulse is detected on any of the input channels, the runt pulse event signal is asserted high for one cycle to inform the application of the condition if the event is configured.

When the runt pulse indication is asserted, the output also indicates pulse stretching to prevent the runt pulse from reaching the application. If dead time is also configured, it is automatically preserved. The additional dead time is added to the complementary output to compensate for the pulse stretching and to protect the application circuits.

15.3. Architectural Overview

The PWM module consists of a common set of controls and features and multiple instantiations of PWM Generators (PGx). Each PWM Generator can be independently configured or multiple PWM Generators can be used to achieve complex multiphase systems. PWM Generators can also be used to implement sophisticated triggering, protection and logic functions. A high-level block diagram is shown in Figure 15-1.

Figure 15-1. PWM High-Level Block Diagram



Each PWM Generator behaves as a separate peripheral that can be independently enabled from the other PWM Generators. Each PWM Generator consists of a signal generator and an output control block.

The PWM Generators use 'events' to trigger other PWM Generators, ADC conversions and external operations. Each PWM Generator accepts a trigger input and produces a trigger output. The trigger input signals the PWM Generator when to start a new PWM period. The trigger output is generated when the trigger time value is equal to the PWM Generator timer value.

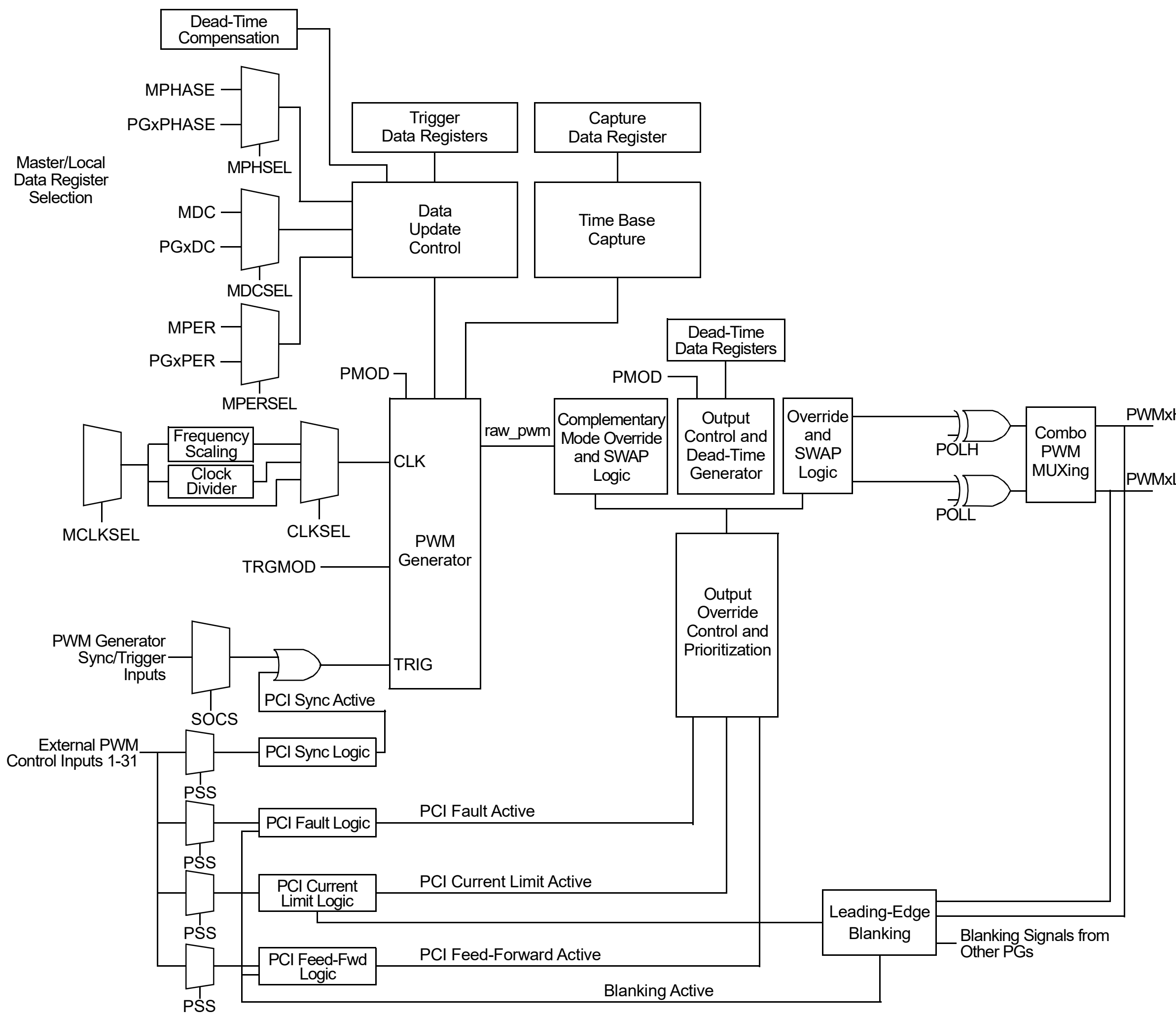
Output control blocks provide the capability to alter the base PWM signal sent to the output pins and incorporate several functions, including:

- Output Mode Selection (Complementary, Push-Pull, Independent)
- Dead-Time Generator

- PWM Control Input (PCI) Block
- Leading-Edge Blanking (LEB)
- Override

Each PWM Generator output block is associated with the control of two PWM output pins. Output blocks contain a PWM Control Input (PCI) that can be used for many purposes, including fault detection, external triggering, and interfacing with other peripherals. The LEB block works in conjunction with the PCI block and allows PCI inputs to be ignored during certain periods of the PWM cycle. The override block determines the PWM output pin states during various types of events, including faults, current limits, and feed-forward control. A block diagram of a single PWM Generator is shown in Figure 15-2.

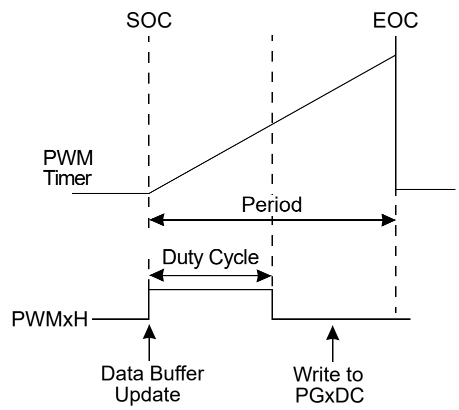
Figure 15-2. Single PWM Generator



PWM Generator operation is based on triggers. To generate a PWM cycle, an SOC (Start-of-Cycle) trigger must be received; the trigger can either be self-triggered or from an external source. Figure 15-3 illustrates a basic PWM waveform, showing SOC and EOC (End-of-Cycle) events. The PWMxH output starts the cycle 'active' (logic high), and when the internal counter reaches the duty cycle value, it transitions to 'inactive' (logic low). EOC is reached when the counter value reaches the period value.

Some operating modes and output modes use multiple counter cycles to produce a single PWM cycle. Refer to PWM Operating Modes and Output Modes for more information.

Figure 15-3. Basic PWM Waveform



15.4. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1000	PCLKCON	31:24								
		23:16								
		15:8	HRRDY	HRERR						LOCK
		7:0			DIVSEL[1:0]			Reserved		MCLKSEL
0x1004	FSCL	31:24								
		23:16					FSCL[15:16]			
		15:8	FSCL[15:8]							
		7:0	FSCL[7:4]							
0x1008	FSMINPER	31:24								
		23:16					FSMINPER[19:16]			
		15:8	FSMINPER[15:8]							
		7:0	FSMINPER[7:4]							
0x100C	MPHASE	31:24								
		23:16					MPHASE[19:16]			
		15:8	MPHASE[15:8]							
		7:0	MPHASE[7:0]							
0x1010	MDC	31:24								
		23:16					MDC[19:0]			
		15:8	MDC[19:0]							
		7:0	MDC[19:0]							
0x1014	MPER	31:24								
		23:16					MPER[19:0]			
		15:8	MPER[19:0]							
		7:0	MPER[19:0]							
0x1018	LFSR	31:24	LFSREN							
		23:16								
		15:8		LFSR[14:8]						
		7:0	LFSR[7:0]							
0x101C	CMBTRIG	31:24								
		23:16					CTB4EN	CTB3EN	CTB2EN	CTB1EN
		15:8								
		7:0					CTA4EN	CTA3EN	CTA2EN	CTA1EN
0x1020	LOGCONA	31:24								
		23:16								
		15:8	PWMS1A[3:0]				PWMS2A[3:0]			
		7:0	S1APOL	S2APOL	PWMLFA[1:0]			PWMLFAD[2:0]		
0x1024	LOGCONB	31:24								
		23:16								
		15:8	PWMS1B[3:0]				PWMS2B[3:0]			
		7:0	S1BPOL	S2BPOL	PWMLFB[1:0]			PWMLFBD[2:0]		
0x1028	LOGCONC	31:24								
		23:16								
		15:8	PWMS1C[3:0]				PWMS2C[3:0]			
		7:0	S1CPOL	S2CPOL	PWMLFC[1:0]			PWMLFCD[2:0]		
0x102C	LOGCOND	31:24								
		23:16								
		15:8	PWMS1D[3:0]				PWMS2D[3:0]			
		7:0	S1DPOL	S2DPOL	PWMLFD[1:0]			PWMLFDD[2:0]		
0x1030	LOGCONE	31:24								
		23:16								
		15:8	PWMS1E[3:0]				PWMS2E[3:0]			
		7:0	S1EPOL	S2EPOL	PWMLFE[1:0]			PWMLFED[2:0]		
0x1034	LOGCONF	31:24								
		23:16								
		15:8	PWMS1F[3:0]				PWMS2F[3:0]			
		7:0	S1FPOL	S2FPOL	PWMLFF[1:0]			PWMLFFD[2:0]		

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1038	PWMEVTA	31:24								
		23:16								
		15:8	EVTAOEN	EVTAPOL	EVTASTRD	EVTASYNC		EVTAPGS[2:0]		
		7:0				EVTASEL[4:0]				
0x103C	PWMEVTB	31:24								
		23:16								
		15:8	EVTBOEN	EVTBPOL	EVTBSTRD	EVTBSYNC		EVTBPGS[2:0]		
		7:0				EVTBSEL[4:0]				
0x1040	PWMEVTC	31:24								
		23:16								
		15:8	EVTCOEN	EVTCPOL	EVTCSTRD	EVTCSYNC		EVTCPGS[2:0]		
		7:0				EVTCSEL[4:0]				
0x1044	PWMEVTD	31:24								
		23:16								
		15:8	EVTDOEN	EVTDPOL	EVTDSTRD	EVTDSYNC		EVTDPGS[2:0]		
		7:0				EVTDSEL[4:0]				
0x1048	PWMEVTE	31:24								
		23:16								
		15:8	EVTEOEN	EVTEPOL	EVTESTRD	EVTESYNC		EVTEPGS[2:0]		
		7:0				EVTESEL[4:0]				
0x104C	PWMEVTF	31:24								
		23:16								
		15:8	EVTFOEN	EVTFPOL	EVTFSTRD	EVTFSYNC		EVTFPGS[2:0]		
		7:0				EVTFSEL[4:0]				
0x1050	PG1CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	HREN			CLKSEL[1:0]		MODSEL[2:0]		
0x1054	PG1STAT	31:24								
		23:16				SEVT	FLT2EVT	FLT1EVT	CLEVT	FFEVT
		15:8			CAPTR	SACT	FLT2ACT	FLT1ACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x1058	PG1IOCON1	31:24								
		23:16	CAPEN	CAPSRC[2:0]			CAPTREN		CAPTRSEL[1:0]	
		15:8					SWAP	FORCEON	PPSEN	DTCMPSEL
		7:0			PMOD[1:0]		PENH	PENL	POLH	POLL
0x105C	PG1IOCON2	31:24								
		23:16	CLMOD		OVRENH	OVRENL				
		15:8			OVRDAT[1:0]		OSYNC[1:0]		FLT2DAT[1:0]	
		7:0	FLT1DAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBGDAT[1:0]	
0x1060	PG1EVT1	31:24	FLT2IEN	FLT1IEN	CLIEN	FFIEN	SIEN		IEVTSEL[1:0]	
		23:16	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
		15:8	DACTREN2	DACTREN1		ADTR1OFS[4:0]				
		7:0	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
0x1064	PG1EVT2	31:24				CAPTROFS[4:0]				
		23:16	CAPTRPS[4:0]							
		15:8				ADTR2OFS[4:0]				
		7:0	ADTR2PS[4:0]					ADTR2EN3	ADTR2EN2	ADTR2EN1
0x1068	PG1F1PCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x106C	PG1F1PCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1070	PG1F2PCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1074	PG1F2PCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1078	PG1CLPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x107C	PG1CLPCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1080	PG1FFPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x1084	PG1FFCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1088	PG1SPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x108C	PG1SPCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1090	PG1LEB	31:24	TRGD	TRGC	TRGB	TRGA	PHR	PHF	PLR	PLF
		23:16					LEB[19:16]			
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x1094	PG1PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:0]							
0x1098	PG1DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:0]							
0x109C	PG1DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]							
0x10A0	PG1PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:0]							
0x10A4	PG1TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:0]							
0x10A8	PG1TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:0]							
0x10AC	PG1TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x10B0	PG1TRIGD	31:24	CAHALF							
		23:16					TRIGD[19:16]			
		15:8	TRIGD[15:8]							
		7:0	TRIGD[7:4]							
0x10B4	PG1TRIGE	31:24	CAHALF							
		23:16					TRIGE[19:16]			
		15:8	TRIGE[15:8]							
		7:0	TRIGE[7:4]							
0x10B8	PG1TRIGF	31:24	CAHALF							
		23:16					TRIGF[19:16]			
		15:8	TRIGF[15:8]							
		7:0	TRIGF[7:0]							
0x10BC	PG1DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x10C0	PG1CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							PG1CAP
0x10C4	PG2CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	HREN			CLKSEL[1:0]		MODSEL[2:0]		
0x10C8	PG2STAT	31:24								
		23:16				SEVT	FLT2EVT	FLT1EVT	CLEVT	FFEVT
		15:8			CAPTR	SACT	FLT2ACT	FLT1ACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x10CC	PG2IOCON1	31:24								
		23:16	CAPEN	CAPSRC[2:0]			CAPTREN		CAPTRSEL[1:0]	
		15:8					SWAP	FORCEON	PPSEN	DTCMPSEL
		7:0			PMOD[1:0]		PENH	PENL	POLH	POLL
0x10D0	PG2IOCON2	31:24								
		23:16	CLMOD		OVRENH	OVRENL				
		15:8			OVRDAT[1:0]		OSYNC[1:0]		FLT2DAT[1:0]	
		7:0	FLT1DAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBGDAT[1:0]	
0x10D4	PG2EVT1	31:24	FLT2IEN	FLT1IEN	CLIEN	FFIEN	SIEN		IEVTSEL[1:0]	
		23:16	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
		15:8	DACTREN2	DACTREN1		ADTR1OFS[4:0]				
		7:0	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
0x10D8	PG2EVT2	31:24				CAPTROFS[4:0]				
		23:16	CAPTRPS[4:0]							
		15:8				ADTR2OFS[4:0]				
		7:0	ADTR2PS[4:0]					ADTR2EN3	ADTR2EN2	ADTR2EN1
0x10DC	PG2F1PCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x10E0	PG2F1PCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x10E4	PG2F2PCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x10E8	PG2F2PCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x10EC	PG2CLPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x10F0	PG2CLPCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x10F4	PG2FFPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x10F8	PG2FFCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x10FC	PG2SPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x1100	PG2SPCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1104	PG2LEB	31:24	TRGD	TRGC	TRGB	TRGA	PHR	PHF	PLR	PLF
		23:16					LEB[19:16]			
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x1108	PG2PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:0]							
0x110C	PG2DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:0]							
0x1110	PG2DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]							
0x1114	PG2PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:0]							
0x1118	PG2TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:0]							
0x111C	PG2TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:0]							
0x1120	PG2TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]							
0x1124	PG2TRIGD	31:24	CAHALF							
		23:16					TRIGD[19:16]			
		15:8	TRIGD[15:8]							
		7:0	TRIGD[7:4]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1128	PG2TRIGE	31:24	CAHALF							
		23:16					TRIGE[19:16]			
		15:8	TRIGE[15:8]							
		7:0	TRIGE[7:4]							
0x112C	PG2TRIGF	31:24	CAHALF							
		23:16					TRIGF[19:16]			
		15:8	TRIGF[15:8]							
		7:0	TRIGF[7:0]							
0x1130	PG2DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x1134	PG2CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							PG2CAP
0x1138	PG3CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	HREN			CLKSEL[1:0]		MODSEL[2:0]		
0x113C	PG3STAT	31:24								
		23:16				SEVT	FLT2EVT	FLT1EVT	CLEVT	FFEVT
		15:8			CAPTR	SACT	FLT2ACT	FLT1ACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x1140	PG3IOCON1	31:24								
		23:16	CAPEN	CAPSRC[2:0]			CAPTREN		CAPTRSEL[1:0]	
		15:8					SWAP	FORCEON	PPSEN	DTCMPSEL
		7:0			PMOD[1:0]		PENH	PENL	POLH	POLL
0x1144	PG3IOCON2	31:24								
		23:16	CLMOD		OVRENH	OVRENL				
		15:8			OVRDAT[1:0]		OSYNC[1:0]		FLT2DAT[1:0]	
		7:0	FLT1DAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBGDAT[1:0]	
0x1148	PG3EVT1	31:24	FLT2IEN	FLT1IEN	CLIEN	FFIEN	SIEN		IEVTSEL[1:0]	
		23:16	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
		15:8	DACTREN2	DACTREN1		ADTR1OFS[4:0]				
		7:0	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
0x114C	PG3EVT2	31:24				CAPTROFS[4:0]				
		23:16	CAPTRPS[4:0]							
		15:8				ADTR2OFS[4:0]				
		7:0	ADTR2PS[4:0]					ADTR2EN3	ADTR2EN2	ADTR2EN1
0x1150	PG3F1PCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x1154	PG3F1PCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1158	PG3F2PCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x115C	PG3F2PCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1160	PG3CLPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1164	PG3CLPCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1168	PG3FFPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x116C	PG3FFCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1170	PG3SPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x1174	PG3SPCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x1178	PG3LEB	31:24	TRGD	TRGC	TRGB	TRGA	PHR	PHF	PLR	PLF
		23:16					LEB[19:16]			
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x117C	PG3PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:0]							
0x1180	PG3DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:0]							
0x1184	PG3DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]							
0x1188	PG3PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:0]							
0x118C	PG3TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:0]							
0x1190	PG3TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:0]							
0x1194	PG3TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]							
0x1198	PG3TRIGD	31:24	CAHALF							
		23:16					TRIGD[19:16]			
		15:8	TRIGD[15:8]							
		7:0	TRIGD[7:4]							
0x119C	PG3TRIGE	31:24	CAHALF							
		23:16					TRIGE[19:16]			
		15:8	TRIGE[15:8]							
		7:0	TRIGE[7:4]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x11A0	PG3TRIGF	31:24	CAHALF							
		23:16					TRIGF[19:16]			
		15:8	TRIGF[15:8]							
		7:0	TRIGF[7:0]							
0x11A4	PG3DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x11A8	PG3CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							PG3CAP
0x11AC	PG4CON	31:24	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
		23:16	TRGMOD[1:0]				SOCS[3:0]			
		15:8	ON					TRGCNT[2:0]		
		7:0	HREN			CLKSEL[1:0]		MODSEL[2:0]		
0x11B0	PG4STAT	31:24								
		23:16				SEVT	FLT2EVT	FLT1EVT	CLEVT	FFEVT
		15:8			CAPTR	SACT	FLT2ACT	FLT1ACT	CLACT	FFACT
		7:0	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
0x11B4	PG4IOCON1	31:24								
		23:16	CAPEN	CAPSRC[2:0]			CAPTREN		CAPTRSEL[1:0]	
		15:8					SWAP	FORCEON	PPSEN	DTCMPSEL
		7:0			PMOD[1:0]		PENH	PENL	POLH	POLL
0x11B8	PG4IOCON2	31:24								
		23:16	CLMOD		OVRENH	OVRENL				
		15:8			OVRDAT[1:0]		OSYNC[1:0]		FLT2DAT[1:0]	
		7:0	FLT1DAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBGDAT[1:0]	
0x11BC	PG4EVT1	31:24	FLT2IEN	FLT1IEN	CLIEN	FFIEN	SIEN		IEVTSEL[1:0]	
		23:16	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
		15:8	DACTREN2	DACTREN1		ADTR1OFS[4:0]				
		7:0	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
0x11C0	PG4EVT2	31:24				CAPTROFS[4:0]				
		23:16	CAPTRPS[4:0]							
		15:8				ADTR2OFS[4:0]				
		7:0	ADTR2PS[4:0]					ADTR2EN3	ADTR2EN2	ADTR2EN1
0x11C4	PG4F1PCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x11C8	PG4F1PCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x11CC	PG4F2PCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x11D0	PG4F2PCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x11D4	PG4CLPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x11D8	PG4CLPCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x11DC	PG4FFPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x11E0	PG4FFCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x11E4	PG4SPCI1	31:24	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
		23:16	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
		15:8	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
		7:0	BPSEL[7:0]							
0x11E8	PG4SPCI2	31:24	PSS[31:24]							
		23:16	PSS[23:16]							
		15:8	PSS[15:8]							
		7:0	PSS[7:0]							
0x11EC	PG4LEB	31:24	TRGD	TRGC	TRGB	TRGA	PHR	PHF	PLR	PLF
		23:16					LEB[19:16]			
		15:8	LEB[15:8]							
		7:0	LEB[7:4]							
0x11F0	PG4PHASE	31:24								
		23:16					PHASE[19:16]			
		15:8	PHASE[15:8]							
		7:0	PHASE[7:0]							
0x11F4	PG4DC	31:24								
		23:16					DC[19:16]			
		15:8	DC[15:8]							
		7:0	DC[7:0]							
0x11F8	PG4DCA	31:24								
		23:16								
		15:8					DCA[11:8]			
		7:0	DCA[7:4]							
0x11FC	PG4PER	31:24								
		23:16					PER[19:16]			
		15:8	PER[15:8]							
		7:0	PER[7:0]							
0x1200	PG4TRIGA	31:24	CAHALF							
		23:16					TRIGA[19:16]			
		15:8	TRIGA[15:8]							
		7:0	TRIGA[7:0]							
0x1204	PG4TRIGB	31:24	CAHALF							
		23:16					TRIGB[19:16]			
		15:8	TRIGB[15:8]							
		7:0	TRIGB[7:0]							
0x1208	PG4TRIGC	31:24	CAHALF							
		23:16					TRIGC[19:16]			
		15:8	TRIGC[15:8]							
		7:0	TRIGC[7:4]							
0x120C	PG4TRIGD	31:24	CAHALF							
		23:16					TRIGD[19:16]			
		15:8	TRIGD[15:8]							
		7:0	TRIGD[7:4]							
0x1210	PG4TRIGE	31:24	CAHALF							
		23:16					TRIGE[19:16]			
		15:8	TRIGE[15:8]							
		7:0	TRIGE[7:4]							
0x1214	PG4TRIGF	31:24	CAHALF							
		23:16					TRIGF[19:16]			
		15:8	TRIGF[15:8]							
		7:0	TRIGF[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1218	PG4DT	31:24		DTH[14:8]						
		23:16	DTH[7:0]							
		15:8		DTL[14:8]						
		7:0	DTL[7:0]							
0x121C	PG4CAP	31:24								
		23:16					CAP[19:16]			
		15:8	CAP[15:8]							
		7:0	CAP[7:4]							PG4CAP

15.4.1. PWM Clock Control Register

Name: PCLKCON
Offset: 0x1000

Notes:
1. These bits cannot be modified while PCLKCON.LOCK = 1.
2. These bits are not present when HREN = 0.
3. User software may write a '0' (after the bit has been set earlier) to this location to turn off and then turn on the PWM high-resolution FEP module. This will reinitialize the PWM high-resolution FEP module.
4. These bits cannot be cleared by clearing the HREN bit. If it has been set prior to clearing the HREN bit, it has to be cleared manually prior to entering high-resolution operation.
5. The PWM clock will also be connected to the clock generator 5 to support the high-resolution FEP. When high-resolution FEP is selected, the MCLKSEL control bits must be set to select the clock generator. Otherwise, unexpected results will occur.

Legend: C = Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	HRRDY	HRERR						LOCK
Access	R	R/C/HS						R/W
Reset	0	0						0

Bit	7	6	5	4	3	2	1	0
			DIVSEL[1:0]			Reserved		MCLKSEL
Access			R/W	R/W		R/W		R/W
Reset			0	0		1		0

Bit 15 – HRRDY High-Resolution Ready bit(2)

Value	Description
1	High-resolution circuitry is ready.
0	High-resolution circuitry is not ready.

Bit 14 – HRERR High-Resolution Error bit(2,3,4)

Value	Description
1	An error has occurred. PWM signals will have limited resolution.
0	No error has occurred. PWM signals will have full resolution when HRRDY = 1.

Bit 8 – LOCK Lock bit

Value	Description
1	Write-protected registers and bits are locked.
0	Write-protected registers and bits are unlocked.

Bits 5:4 – DIVSEL[1:0] PWM Clock Divider Selection bits(1)

Value	Description
11	Divide ratio is 1:16
10	Divide ratio is 1:8
01	Divide ratio is 1:4
00	Divide ratio is 1:2

Bit 2 – Reserved Maintain as '1'

Bit 0 – MCLKSEL PWM Master Clock Selection bit(1,5)

Note: Do not change the MCLKSEL[1:0] bits while ON (PGxCON[15]) = 1.

Value	Description
1	CKLGEN5
0	UPB clock

15.4.2. Frequency Scale Register

Name: FSCL
Offset: 0x1004

Note:
1. These bits can be modified while PGxCON.ON = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					FSCL[15:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FSCL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FSCL[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 19:16 – FSCL[15:16]

Bits 15:8 – FSCL[15:8]

Bits 7:4 – FSCL[7:4] Frequency Scale Register bits(1)
The value in this register is added to the frequency scaling accumulator at each pwm_master_clk. When the accumulated value exceeds the value of FSMINPER, a clock pulse is produced.

15.4.3. Frequency Scaling Minimum Period Register

Name: FSMINPER
Offset: 0x1008

Note:

1. These bits can be modified while PGxCON.ON = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					FSMINPER[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FSMINPER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FSMINPER[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 19:4 – FSMINPER[19:4] Frequency Scaling Minimum Period Register bits[(1)]
This register holds the minimum clock period (maximum clock frequency) that can be produced by the frequency scaling circuit.

15.4.4. Master Phase Register

Name: MPHASE
Offset: 0x100C

Notes:
1. If HREN = 0, the four least significant bits are read as '0000'.
2. These bits cannot be modified while UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					MPHASE[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MPHASE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MPHASE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:0 – MPHASE[19:0] Master Phase Register bits(2)
This register holds the phase offset value that can be shared by multiple PWM Generators.

15.4.5. Master Duty Cycle Register

Name: MDC
Offset: 0x1010

Notes:
1. These bits cannot be modified while UPDATE = 1.
2. If HREN = 0, the four least significant bits are read as '0000'.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					MDC[19:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MDC[19:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MDC[19:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:0 – MDC[19:0] Master Duty Cycle Register bits[1,2]
This register holds the duty cycle value that can be shared by multiple PWM Generators.

15.4.6. Master Period Register

Name: MPER
Offset: 0x1014

Notes:
1. These bits cannot be modified while UPDATE = 1.
2. If HREN = 0, the four least significant bits are read as '0000'.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
					MPER[19:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	MPER[19:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	MPER[19:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:0 – MPER[19:0] Master Period Register bits[1,2]
This register holds the period value that can be shared by multiple PWM Generators.

15.4.7. Linear Feedback Shift Register

Name: LFSR
Offset: 0x1018

Note:
1. Location is read-only.

Bit	31	30	29	28	27	26	25	24
	LFSREN							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
		LFSR[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LFSR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – LFSREN Linear Feedback Shift Register Enable bit

Value	Description
1	Shift register is enabled.
0	Shift register is not enabled.

Bits 14:0 – LFSR[14:0] Linear Feedback Shift Register bits[1]
A read of this register will provide a 15-bit pseudorandom value.

15.4.8. Combinational Trigger Register[(1)]

Name: CMBTRIG
Offset: 0x101C

Note:
1. Caution must be used when modifying register bits while PGxCON.ON = 1; unexpected results may occur.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					CTB4EN	CTB3EN	CTB2EN	CTB1EN
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					CTA4EN	CTA3EN	CTA2EN	CTA1EN
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 19 – CTB4EN Enable Trigger Output from PWM Generator #4 as Source for Combinational Trigger B bit

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal.
0	Disabled

Bit 18 – CTB3EN Enable Trigger Output from PWM Generator #3 as Source for Combinational Trigger B bit

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal.
0	Disabled

Bit 17 – CTB2EN Enable Trigger Output from PWM Generator #2 as Source for Combinational Trigger B bit

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal.
0	Disabled

Bit 16 – CTB1EN Enable Trigger Output from PWM Generator #1 as Source for Combinational Trigger B bit

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal,
0	Disabled

Bit 3 – CTA4EN Enable Trigger Output from PWM Generator #4 as Source for Combinational Trigger A bit

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal.
0	Disabled

Bit 2 – CTA3EN Enable Trigger Output from PWM Generator #3 as Source for Combinational Trigger A bit

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal.
0	Disabled

Bit 1 – CTA2EN Enable Trigger Output from PWM Generator #2 as Source for Combinational Trigger A bit

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal.
0	Disabled

Bit 0 – CTA1EN Enable Trigger Output from PWM Generator #1 as Source for Combinational Trigger A bit

Value	Description
1	Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal.
0	Disabled

15.4.9. Combinatorial PWM Logic Control Register

Name: LOGCONy
Offset: 0x1020, 0x1024, 0x1028, 0x102C, 0x1030, 0x1034

Notes:

1. 'y' denotes a common instance (A-F).
2. Instances of y = A, C and E of LOGCONy assign the logic function output to the PWMxH pin. Instances of y = B, D and F of LOGCONy assign the logic function to the PWMxL pin.
3. Caution should be used when modifying these register bits while PGxCON.ON = 1; unexpected results may occur.
4. The logic function input will be connected to '0' if the PWM channel is not present.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PWMS1y[3:0]				PWMS2y[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	S1yPOL	S2yPOL	PWMLFy[1:0]			PWMLFyD[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bits 15:12 – PWMS1y[3:0] Combinatorial PWM Logic Source #1 Selection bits(4)

Value	Description
1111-1000	Reserved
0111	PWML4
0110	PWMH4
0101	PWML3
0100	PWMH3
0011	PWML2
0010	PWMH2
0001	PWML1
0000	PWMH1

Bits 11:8 – PWMS2y[3:0] Combinatorial PWM Logic Source #2 Selection bits(4)

Value	Description
1111-1000	Reserved
0111	PWML4
0110	PWMH4
0101	PWML3
0100	PWMH3

Value	Description
0011	PWML2
0010	PWMH2
0001	PWML1
0000	PWMH1

Bit 7 – S1yPOL Combinatorial PWM Logic Source #1 Polarity bit

Value	Description
1	Input is inverted.
0	Input is positive logic.

Bit 6 – S2yPOL Combinatorial PWM Logic Source #2 Polarity bit

Value	Description
1	Input is inverted.
0	Input is positive logic.

Bits 5:4 – PWMLFy[1:0] Combinatorial PWM Logic Function Selection bits

Value	Description
11	Reserved
10	PWMS1y ^ PWMS2y (XOR)
01	PWMS1y & PWMS2y (AND)
00	PWMS1y | PWMS2y (OR)

Bits 2:0 – PWMLFyD[2:0] Combinatorial PWM Logic Destination Selection bits(3)

Value	Description
111-100	Reserved
011	Logic Function assigned to PWM4
010	Logic Function assigned to PWM3
001	Logic Function assigned to PWM2
000	No assignment. PWM Logic Function y is disabled.

15.4.10. PWM Event Output Control Register y

Name: PWMEVTy
Offset: 0x1038, 0x103C, 0x1040, 0x1044, 0x1048, 0x104C

Notes:
1. 'y' denotes a common instance (A-F).
2. This is the PWM generator output signal prior to output mode logic and any output override logic.
3. Event output signal pulse will be two system clocks when this bit is set and EVTySTRD = 1.
4. The event signal is stretched using the peripheral bus clock because different PWM Generators may be operating from different clock sources.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	EVTyOEN	EVTyPOL	EVTySTRD	EVTySYNC		EVTyPGS[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
				EVTySEL[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 15 – EVTyOEN PWM Event Output Enable bit

Value	Description
1	Event output signal is output on the PWMEy pin.
0	Event output signal is internal only.

Bit 14 – EVTyPOL PWM Event Output Polarity

Value	Description
1	Event output signal is active-low.
0	Event output signal is active-high.

Bit 13 – EVTySTRD PWM Event Output Stretch Disable bit[(4)]

Value	Description
1	Event output signal pulse width is not stretched.
0	Event output signal is stretched to eight PWM clock cycles minimum.

Bit 12 – EVTySYNC PWM Event Output Sync bit[(3)]
Event output signal pulse will be synchronized to peripheral_clk.

Value	Description
1	Event output signal is synchronized to the system clock.
0	Event output is not synchronized to the system clock.

Bits 10:8 – EVTyPGS[2:0] PWM Event Source Selection bits

Value	Description
111-100	Reserved
011	PG4
010	PG3
001	PG2
0000	PG1

Bits 4:0 – EVTySEL[4:0] PWM Event Selection bits(2)

Value	Description
11111-11110	Reserved when FEP = 0
11101	MPLL Lock event signal ('Reserved' when HREN = 0)
11100	MPLL Lock Lost event signal ('Reserved' HREN = 0)
11011	MPLL Lock Time-Out event signal ('Reserved' when HREN = 0)
11010	Runt Pulse event signal ('Reserved' when HREN = 0)
01100-11001	Reserved
01011	DAC Trigger signal
01010	ADC Trigger 2 signal
01001	ADC Trigger 1 signal
01000	STEER signal (available in push-pull output modes only)
00111	PHASE signal (available in center aligned modes only)
00110	PCI Fault 2 active output signal
00101	PCI Fault 1 active output signal
00100	PCI Current Limit active output signal
00011	PCI Feed-Forward active output signal
00010	PCI Sync active output signal
00001	PWM generator output signal
00000	Source selected by PGTRGSEL [2:0] bits

15.4.11. PWM Generator x Control Register

Name: PGxCON
Offset: 0x1050, 0x10C4, 0x1138, 0x11AC

Notes:
1. These bits cannot be modified while PGxCON.ON = 1.
2. Caution should be used when modifying these bits while PGxCON.ON = 1; unexpected results may occur.
3. Any generator can be used to trigger any other generator.
4. When this bit is cleared, an asynchronous reset is initiated, causing all of the macro outputs to return to their reset/default states, which can appear as glitches. Safe shutdown procedures are highly recommended to ensure all recipients of these glitches enter their benign states prior to this bit being cleared.
5. This bit is not present when FEP = 0.
6. The PCI-selected sync signal is always available to be "OR'd" with the selected SOCx signal per the SOCS[3:0] if the PCI sync function is enabled.
7. The source selected by the SOCS[3:0] bits MUST operate from the same clock source as the PWM generator. If not, the source must be routed through the PCI Sync logic so the trigger signal may be synchronized to the PWM generator clock domain.
8. The PWM generator timebase operates from the Frequency Scaling circuit clock, effectively scaling the duty cycle and period of the PWM generator output. The remainder of the PWM circuit operates from the clock source supplied to the Frequency Scaling circuit.
9. This clock source should not be used when PGxCON.HREN = 1; unexpected results may occur.
10. These bit(s) cannot be modified while PCLKCON.LOCK = 1.

Bit	31	30	29	28	27	26	25	24
	MDCSEL	MPERSEL	MPHSEL		MSTEN	UPDMOD[2:0]		
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRGMOD[1:0]				SOCS[3:0]			
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON					TRGCNT[2:0]		
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit	7	6	5	4	3	2	1	0
	HREN			CLKSEL[1:0]		MODSEL[2:0]		
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 31 – MDCSEL Master Duty Cycle Register Select bit[(2)]

Value	Description
1	PWM Generator uses the MDC register.
0	PWM Generator uses the PGxDC register.

Bit 30 – MPERSEL Master Period Register Select bit(2)

Value	Description
1	PWM Generator uses the MPER register.
0	PWM Generator uses the PGxPER register.

Bit 29 – MPHSEL Master Phase Register Select bit(2)

Value	Description
1	PWM Generator uses the MPHASE register.
0	PWM Generator uses the PGxPHASE register.

Bit 27 – MSTEN Master Update Enable bit(2)

Value	Description
1	PWM Generator broadcasts a software set of the UPDREQ control bit and the EOC signal to other PWM Generators.
0	PWM Generator does not broadcast the UPDREQ status bit state or the EOC signal.

Bits 26:24 – UPDMOD[2:0] PWM Buffer Update Mode Selection bits
See Table 15-10 for details.

Bits 23:22 – TRGMOD[1:0] PWM Generator x Trigger Mode Selection bits(2)

Value	Description
11	Reserved
10	Reserved
01	PWM Generator operates in Retriggerable mode.
00	PWM Generator operates in Single Trigger mode.

Bits 19:16 – SOCS[3:0] Start-of-Cycle Selection bits(2,3,6,7)

Value	Description
1111	TRIG bit or PCI Sync function only (no hardware trigger source selected)
1110-0101	Reserved
0100	PWM4 PG Trigger output is selected by PG4EVT.PGTRGSEL bits.
0011	PWM3 PG Trigger output is selected by PG3EVT.PGTRGSEL bits.
0010	PWM2 PG Trigger output is selected by PG2EVT.PGTRGSEL bits.
0001	PWM1 PG Trigger output is selected by PG1EVT.PGTRGSEL bits.
0000	Local EOC

Bit 15 – ON PWM Generator x Enable bit(4)

Value	Description
1	PWM Generator is enabled.
0	PWM Generator is not enabled.

Bits 10:8 – TRGCNT[2:0] PWM Generator x Trigger Count Select bits(1)

Value	Description
111	PWM Generator produces 8 PWM cycles after being triggered.
110	PWM Generator produces 7 PWM cycles after being triggered.
101	PWM Generator produces 6 PWM cycles after being triggered.
100	PWM Generator produces 5 PWM cycles after being triggered.
011	PWM Generator produces 4 PWM cycles after being triggered.
010	PWM Generator produces 3 PWM cycles after being triggered.
001	PWM Generator produces 2 PWM cycles after being triggered.
000	PWM Generator produces 1 PWM cycle after being triggered.

Bit 7 – HREN High-Resolution Enable bit[5,10]

Value	Description
1	PWM operates in high-resolution mode.
0	PWM operates at standard resolution.

Bits 4:3 – CLKSEL[1:0] Clock Selection bits[2]

Value	Description
11	PWM Generator uses the master clock scaled by the frequency scaling circuit[2].
10	PWM Generator uses the master clock divided by the clock divider circuit[2].
01	PWM Generator uses the master clock selected by the MCLKSEL[1:0] (PCLKCON[1:0]) control bits.
00	No clock selected, PWM Generator is in the lowest power state (default).

Bits 2:0 – MODSEL[2:0] PWM Generator x Mode Selection bits[1]

Value	Description
111	Dual Edge Center-Aligned PWM mode (interrupt/register update twice per cycle)
110	Dual Edge Center-Aligned PWM mode (interrupt/register update once per cycle)
101	Double Update Center-Aligned PWM mode
100	Center-Aligned PWM mode
011	LLC Resonant Converter Support PWM mode
010	Independent Edge PWM mode, dual output
001	Variable Phase PWM mode
000	Independent Edge PWM mode

15.4.12. PWM Generator x Status Register

Name: PGxSTAT
Offset: 0x1054, 0x10C8, 0x113C, 0x11B0

Legend: C = Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				SEVT	FLT2EVT	FLT1EVT	CLEVT	FFEVT
Access				HS/C	HS/C	HS/C	HS/C	HS/C
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			CAPTR	SACT	FLT2ACT	FLT1ACT	CLACT	FFACT
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRSET	TRCLR	CAP	UPDATE	UPDREQ	STEER	CAHALF	TRIG
Access	W	W	R/HS	R	W	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 20 – SEVT PCI Sync Event bit

Value	Description
1	A PCI Sync event has occurred (rising edge on PCI Sync output or PCI Sync output is high when the module is enabled).
0	No PCI Sync event has occurred.

Bit 19 – FLT2EVT PCI Fault 2 Active Status bit

Value	Description
1	A Fault 2 event has occurred (rising edge on PCI Fault output or PCI Fault output is high when module is enabled).
0	No Fault 2 event has occurred.

Bit 18 – FLT1EVT PCI Fault 1 Active Status bit

Value	Description
1	A Fault 1 event has occurred (rising edge on PCI Fault output or PCI Fault output is high when the module is enabled).
0	No Fault 1 event has occurred.

Bit 17 – CLEVT PCI Current Limit Status bit

Value	Description
1	A PCI current limit event has occurred (rising edge on PCI current limit output or PCI current limit output is high when the module is enabled).
0	No PCI current limit event has occurred.

Bit 16 – FFEVT PCI Feed-Forward Active Status bit

Value	Description
1	A PCI feed-forward event has occurred (the rising edge on the PCI feed-forward output or PCI feed-forward output is high when the module is enabled).
0	No PCI feed-forward event has occurred.

Bit 13 – CAPTR Capture Status Trigger bit

Value	Description
1	A newly calculated Trigger value has been directed to a trigger register per CAPTRSEL[1:0].
0	No capture has occurred.

Bit 12 – SACT PCI Sync Status bit

Value	Description
1	PCI Sync output is active.
0	PCI Sync output is inactive.

Bit 11 – FLT2ACT PCI Fault 2 Active Status bit

Value	Description
1	PCI Fault 2 output is active.
0	PCI Fault 2 output is inactive.

Bit 10 – FLT1ACT PCI Fault 1 Active Status bit

Value	Description
1	PCI Fault 1 output is active.
0	PCI Fault 1 output is inactive.

Bit 9 – CLACT PCI Current Limit Status bit

Value	Description
1	PCI current limit output is active.
0	PCI current limit output is inactive.

Bit 8 – FFACT PCI Feed-Forward Active Status bit

Value	Description
1	PCI feed-forward output is active.
0	PCI feed-forward output is inactive.

Bit 7 – TRSET PWM Generator Software Trigger Set bit

User software writes a '1' to this bit location to trigger a PWM Generator cycle. The bit location always reads as '0'. The TRIG bit will indicate '1' when the PWM Generator is triggered.

Bit 6 – TRCLR PWM Generator Software Trigger Clear bit

User software writes a '1' to this bit location to stop a PWM Generator cycle. The bit location always reads as '0'. The TRIG bit will indicate '0' when the PWM Generator is not triggered.

Bit 5 – CAP Capture Status bit

Value	Description
1	PWM Generator time base value has been captured in PGxCAP.
0	No capture has occurred.

Bit 4 – UPDATE PWM Data Register Update Status/Control bit

Value	Description
1	PWM Data register update is pending – user Data registers are not writable.
0	No PWM Data register update is pending.

Bit 3 – UPDREQ PWM Data Register Update Request bit

User software writes a '1' to this bit location to request a PWM Data register update. The bit location always reads as '0'. The UPDATE status bit will indicate a '1' when an update is pending.

Bit 2 – STEER Output Steering Status bit (Push-Pull Output mode only)

Value	Description
1	PWM Generator is in 2nd cycle of Push-Pull mode.
0	PWM Generator is in 1st cycle of Push-Pull mode.

Bit 1 – CAHALF Half Cycle Status bit (Center-Aligned modes only)

Value	Description
1	PWM Generator is in 2nd half of time base cycle.
0	PWM Generator is in 1st half of time base cycle.

Bit 0 – TRIG Trigger Status bit

Value	Description
1	PWM Generator is triggered and PWM cycle is in progress.
0	No PWM cycle is in progress.

15.4.13. PWM Generator x I/O Control 1 Register

Name: PGxIOCON1
Offset: 0x1058, 0x10CC, 0x1140, 0x11B4

Notes:

1. These bits cannot be modified while PGxCON.ON = `1`.
2. These bits cannot be modified while PCLKCON.LOCK = `1`.
3. Caution should be exercised when modifying these bits while PGxCON.ON = `1`; unexpected results may occur.
4. These bits are effective only when CAPTREN is set high (see Capture to Trigger).
5. PGxTRIGF has a dedicated function in Complementary mode (see Output Override Behavior in Complementary Output Mode with PWMxL's Max On-time Adjustment).
6. Care must be taken if the selected trigger is also selected by PGxEVT1.PGTRGSEL[2:0].

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CAPEN	CAPSRC[2:0]			CAPTREN		CAPTRSEL[1:0]	
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	15	14	13	12	11	10	9	8
					SWAP	FORCEON	PPSEN	DTCMPSEL
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
			PMOD[1:0]		PENH	PENL	POLH	POLL
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 23 – CAPEN Timebase Capture Enable bit[(1)]

Value	Description
`1`	Time base value captured constantly based on CAPSRC[2:0].
`0`	Timebase value captured based on CAPSRC[2:0] only after the read of PGxCAP register.

Bits 22:20 – CAPSRC[2:0] Time Base Capture Source Selection bits[(1)]

Note: A capture may be initiated in software at any time by writing a '`1`' to PGxCAP[0].

Value	Description
`111`	Reserved
`110`	Reserved
`101`	Capture time base value at assertion of selected PCI Fault 2 signal.
`100`	Capture time base value at assertion of selected PCI Fault 1 signal.
`011`	Capture time base value at assertion of selected PCI Current Limit signal.
`010`	Capture time base value at assertion of selected PCI Feed-Forward signal.
`001`	Capture time base value at assertion of selected PCI Sync signal.
`000`	No hardware source selected for time base capture – software only.

Bit 19 – CAPTREN Timebase Capture to Trigger Enable bit[1]

Value	Description
1	Time base capture to trigger enabled
0	Time base capture to trigger disabled

Bits 17:16 – CAPTRSEL[1:0] Timebase Capture Trigger Register Selection bits[1,4,5,6]

Value	Description
11	PGxTRIGF selected to store the 50% time base captured value when enabled.
10	PGxTRIGE selected to store the 50% time base captured value when enabled.
01	PGxTRIGD selected to store the 50% time base captured value when enabled.
00	PGxTRIGC selected to store the 50% time base captured value when enabled.

Bit 11 – SWAP Swap PWM Signals to PWMxH and PWMxL Device Pins bit

Value	Description
1	The PWMxH signal is connected to the PWMxL pin, and the PWMxL signal is connected to the PWMxH pin.
0	PWMxH/L signals are mapped to their respective pins.

Bit 10 – FORCEON Force On Select bit[1,2]

Note: This bit applies to the complementary output mode only (PGxIOCON1.OUTMOD[1:0] = 0b00).

Value	Description
1	Active override happens immediately without taking the dead time into account.
0	Active override happens after taking the dead time into account.

Bit 9 – PPSEN Peripheral Pin Select Enable bit[3]

Value	Description
1	Peripheral pin select enabled.
0	Peripheral pin select disabled, as a result, PWM outputs are hard-mapped to pins.

Bit 8 – DTCMPSEL Dead-Time Compensation Select bit[3]

Value	Description
1	Dead-time compensation is controlled by PCI feed-forward limit logic.
0	Dead-time compensation is controlled by PCI Sync logic.

Bits 5:4 – PMOD[1:0] PWM Generator Output Mode Selection bits[2]

Value	Description
11	Reserved
10	PWM Generator outputs operate in Push-Pull mode.
01	PWM Generator outputs operate in Independent mode.
00	PWM Generator outputs operate in Complementary mode.

Bit 3 – PENH PWMxH Output Port Enable bit[2]

Value	Description
1	PWM Generator controls the PWMxH output pin.
0	PWM Generator does not control the PWMxH output pin.

Bit 2 – PENL PWMxL Output Port Enable bit[2]

Value	Description
1	PWM Generator controls the PWMxL output pin.
0	PWM Generator does not control the PWMxL output pin.

Bit 1 – POLH PWMxH Output Polarity bit[2]

Value	Description
1	Output pin is inverted.
0	Output pin is non-inverted.

Bit 0 – POLL PWMxL Output Polarity bit[2]

Value	Description
1	Output pin is inverted.
0	Output pin is non-inverted.

15.4.14. PWM Generator x I/O Control 2 Register

Name: PGxIOCON2
Offset: 0x105C, 0x10D0, 0x1144, 0x11B8

Notes:
1. This bit applies to operation in complementary mode only. When the outputs are swapped, PGxIOCON1.SWAP = `1`, the priority is observed based on Figure 15-5.
2. The PWM PGx FRZ bits are controlled by MPLAB® IDE. To access the peripheral FRZ bit, open **Project Properties** and navigate to the options for the debugger being used (such as ICDx). From the option categories drop-down, select **Freeze Peripherals**, then select the PWM PGx instance. By default, the FRZ bit is set when Debug mode is entered.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	CLMOD		OVRENH	OVRENL				
Access	R/W		R/W	R/W				
Reset	0		0	0				

Bit	15	14	13	12	11	10	9	8
			OVRDAT[1:0]		OSYNC[1:0]		FLT2DAT[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FLT1DAT[1:0]		CLDAT[1:0]		FFDAT[1:0]		DBGDAT[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 23 – CLMOD Current Limit Mode Select bit(1)

Value	Description
1	If PCI current limit is active, then the PWMxH and PWMxL output signals are inverted (bit flipping), and the CLDAT[1:0] bits are not used.
0	If PCI current limit is active, then the CLDAT[1:0] bits define the PWM output levels.

Bit 21 – OVRENH User Override Enable for PWMxH Pin bit

Value	Description
1	OVRDAT[1] provides data for output on PWMHx pin.
0	PWM generator provides data for PWMHx pin.

Bit 20 – OVRENL User Override Enable for PWMxL Pin bit

Value	Description
1	OVRDAT[0] provides data for output on the PWMxL pin.
0	PWM Generator provides data for the PWMxL pin.

Bits 13:12 – OVRDAT[1:0] Data for PWMxH/PWMxL Pins if Override is Enabled bits

Description
If OVRENH = `1`, then OVRDAT[1] provides data for PWMxH.

Description
If OVRENL = 1, then OVRDAT[0] provides data for PWMxL.

Bits 11:10 – OSYNC[1:0] User Output Override Synchronization Control bits

Value	Description
11	User output overrides via the SWAP, OVRENL/H and OVRDAT[1:0] bits are synchronized to the data buffer update of the selected PWM mode. This makes this setting equivalent to setting 10 when UPDMOD[2:0] = 000 with UPDREQ properly set manually.
10	User output overrides via the SWAP, OVRENL/H and OVRDAT[1:0] bits occur when specified by the UPMOD[2:0] bits in the PGxCON register.
01	User output overrides via the SWAP, OVRENL/H and OVRDAT[1:0] bits occur immediately (as soon as possible).
00	User output overrides via the SWAP, OVRENL/H and OVRDAT[1:0] bits are synchronized to the local PWM time base (next start of cycle).

Bits 9:8 – FLT2DAT[1:0] Data for PWMHx/Lx Pins if the FLT Event is Active bits

Description
If Fault active, then FLT2DAT[1] provides data for PWMHx.
If Fault active, then FLT2DAT[0] provides data for PWMLx.

Bits 7:6 – FLT1DAT[1:0] Data for PWMHx,Lx Pins if FLT Event is Active bits

Description
If Fault active, then FLT1DAT[1] provides data for PWMHx.
If Fault active, then FLT1DAT[0] provides data for PWMLx.

Bits 5:4 – CLDAT[1:0] Data for PWMxH/PWMxL Pins if CLMT Event is Active bits

Description
If current limit is active, then CLDAT[1] provides data for PWMxH.
If current limit is active, then CLDAT[0] provides data for PWMxL.

Bits 3:2 – FFDAT[1:0] Data for PWMHx/Lx Pins if the Feed-Forward Event is Active bits

Description
If feed-forward is active, then FFDAT[1] provides data for PWMxH.
If feed-forward is active, then FFDAT[0] provides data for PWMxL.

Bits 1:0 – DBGDAT[1:0] Data for PWMHx/Lx Pins if Debug Mode is Active bits[(2)]

Description
If Debug mode is active and **Freeze peripherals** is selected in **Project Properties** in MPLAB IDE, then DBGDAT[1] provides data for PWMxH.
If Debug mode is active and **Freeze peripherals** is selected in **Project Properties** in MPLAB IDE, then DBGDAT[0] provides data for PWMxL.

15.4.15. PWM Generator x Event 1 Register

Name: PGxEVT1
Offset: 0x1060, 0x10D4, 0x1148, 0x11BC

Notes:
1. Caution should be exercised when modifying these bits while PGxCON.ON =`1`; unexpected results may occur.
2. This source can optionally be used as a PCI input, PCI qualifier, PCI terminator or PCI terminator qualifier.
3. These events are derived from the internal PWM generator time base comparison events.
4. Care must be taken if the selected trigger is also selected by PGxIOCON1.CAPTRSEL[1:0].

Bit	31	30	29	28	27	26	25	24
	FLT2IEN	FLT1IEN	CLIEN	FFIEN	SIEN		IEVTSEL[1:0]	
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	23	22	21	20	19	18	17	16
	PWMPCI[2:0]			UPDTRG[1:0]		PGTRGSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DACTREN2	DACTREN1		ADTR1OFS[4:0]				
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADTR1PS[4:0]					ADTR1EN3	ADTR1EN2	ADTR1EN1
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – FLT2IEN PCI Fault 2 Interrupt Enable bit
Note: An interrupt is only generated on the rising edge of the PCI Fault active signal.

Value	Description
1	Fault interrupt is enabled.
0	Fault interrupt is disabled.

Bit 30 – FLT1IEN PCI Fault 1 Interrupt Enable bit
Note: An interrupt is only generated on the rising edge of the PCI Fault active signal.

Value	Description
1	Fault interrupt is enabled.
0	Fault interrupt is disabled.

Bit 29 – CLIEN PCI Current Limit Interrupt Enable bit
Note: An interrupt is only generated on the rising edge of the PCI current limit active signal.

Value	Description
1	Current limit interrupt is enabled.
0	Current limit interrupt is disabled.

Bit 28 – FFIEN PCI Feed-Forward Interrupt Enable bit

Note: An interrupt is only generated on the rising edge of the PCI feed-forward active signal.

Value	Description
1	Feed-forward interrupt is enabled.
0	Feed-forward interrupt is disabled.

Bit 27 – SIEN PCI Sync Interrupt Enable bit

Note: An interrupt is only generated on the rising edge of the PCI Sync active signal.

Value	Description
1	Sync interrupt is enabled.
0	Sync interrupt is disabled.

Bits 25:24 – IEVTSEL[1:0] Interrupt Event Selection bits(1)

Value	Description
11	Time base interrupts are disabled (sync, Fault, current limit and feed-forward events can be independently enabled).
10	Interrupts CPU at ADC Trigger 1 event
01	Interrupts CPU at TRIGA compare event
00	Interrupts CPU at EOC

Bits 23:21 – PWMPCI[2:0] PWM PCI Source Selection bits(2)

Value	Description
111-100	Reserved
011	PWM Generator #4 output is used as PCI signal.
010	PWM Generator #3 output is used as PCI signal.
001	PWM Generator #2 output is used as PCI signal.
000	PWM Generator #1 output is used as PCI signal.

Bits 20:19 – UPDTRG[1:0] Update Trigger Select bits(1)

Value	Description
11	If PGxCON.MPERSEL = 1, then a write to the MPER register automatically sets the UPDATE bit. If PGxCON.MPERSEL = 0 then a write of PGxTRIGA register automatically sets UPDATE bit
10	If PGxCON.MPHSEL = 1, then a write to the MPHASE register automatically sets the UPDATE bit. If PGxCON.MPHSEL = 0, then a write to the PGxPHASE register automatically sets the UPDATE bit.
01	If PGxCON.MDCSEL = 1, then a write to the MDC register automatically sets the UPDATE bit. If PGxCON.MDCSEL = 0, then a write to the PGxDC register automatically sets the UPDATE bit.
00	User must set the PGxSTAT.UPDREQ bit manually.

Bits 18:16 – PGTRGSEL[2:0] PWM Generator Trigger Output Selection bits(1,3,4)

Value	Description
111	Reserved
110	PGxTRIGF compare event is a PG Trigger
101	PGxTRIGE compare event is a PG Trigger
100	PGxTRIGD compare event is a PG Trigger
011	PGxTRIGC compare event is a PG Trigger
010	PGxTRIGB compare event is a PG Trigger
001	PGxTRIGA compare event is a PG Trigger
000	EOC event is PG Trigger; this selection is required to function if the cycle is terminated by Sync PCI.

Bit 15 – DACTREN2 DAC Trigger Source is PGxTRIGE Compare Event Enable bit

Value	Description
1	PGxTRIGE register compare event enabled as a trigger source for DAC Trigger
0	PGxTRIGE register compare event disabled as a trigger source for DAC Trigger

Bit 14 – DACTREN1 DAC Trigger Source is PGxTRIGD Compare Event Enable bit

Value	Description
1	PGxTRIGD register compare event enabled as a trigger source for DAC Trigger
0	PGxTRIGD register compare event disabled as a trigger source for DAC Trigger

Bits 12:8 – ADTR1OFS[4:0] ADC Trigger 1 Offset Selection bits(1)

Value	Description
11111	Offset by 31 trigger events
. . .	. . .
00010	Offset by 2 trigger events
00001	Offset by 1 trigger event
00000	No offset

Bits 7:3 – ADTR1PS[4:0] ADC Trigger 1 Postscaler Selection bits(1)

Value	Description
11111	1:32
. . .	. . .
00010	1:3
00001	1:2
00000	1:1

Bit 2 – ADTR1EN3 ADC Trigger 1 Source is PGxTRIGC Compare Event Enable bit(1)

Value	Description
1	PGxTRIGC register compare event is enabled as a trigger source for ADC Trigger 1
0	PGxTRIGC register compare event is disabled as a trigger source for ADC Trigger 1

Bit 1 – ADTR1EN2 ADC Trigger 1 Source is PGxTRIGB Compare Event Enable bit(1)

Value	Description
1	PGxTRIGB register compare event is enabled as a trigger source for ADC Trigger 1
0	PGxTRIGB register compare event is disabled as a trigger source for ADC Trigger 1

Bit 0 – ADTR1EN1 ADC Trigger 1 Source is PGxTRIGA Compare Event Enable bit(1)

Value	Description
1	PGxTRIGA register compare event is enabled as a trigger source for ADC Trigger 1
0	PGxTRIGA register compare event is disabled as a trigger source for ADC Trigger 1

15.4.16. PWM Generator x Event 2 Register[1]

Name: PGxEVT2
Offset: 0x1064, 0x10D8, 0x114C, 0x11C0

Notes:
1. Caution should be exercised when modifying this bit(s) while PGxCON.ON = 1; unexpected results may occur.
2. This source can optionally be used as a PCI input, PCI qualifier, PCI terminator or PCI terminator qualifier.

Bit	31	30	29	28	27	26	25	24
				CAPTROFS[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CAPTRPS[4:0]							
Access	R/W	R/W	R/W	R/W	R/W			
Reset	0	0	0	0	0			

Bit	15	14	13	12	11	10	9	8
				ADTR2OFS[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADTR2PS[4:0]					ADTR2EN3	ADTR2EN2	ADTR2EN1
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 28:24 – CAPTROFS[4:0] Timebase capture to Trigger Offset Selection bits
Note: See Capture to Trigger and LLC Resonant Converter Mode

Value	Description
11111	Offset by 31 trigger events
...	
00010	Offset by 2 trigger events
00001	Offset by 1 trigger event
00000	No offset

Bits 23:19 – CAPTRPS[4:0] Timebase capture to Trigger Post Scaler Selection bits
Note: See Capture to Trigger and LLC Resonant Converter Mode

Value	Description
11111	1:32
...	Current limit interrupt is disabled.
00010	1:3
00001	1:2
00000	1:1

Bits 12:8 – ADTR2OFS[4:0] ADC Trigger 2 Offset Selection bits

Value	Description
11111	Offset by 31 trigger events

Value	Description
. . .	. . .
00010	Offset by 2 trigger events
00001	Offset by 1 trigger event
00000	No offset

Bits 7:3 – ADTR2PS[4:0] ADC Trigger 2 Post Scaler Selection bits

Value	Description
11111	1:32
. . .	Current limit interrupt is disabled.
00010	1:3
00001	1:2
00000	1:1

Bit 2 – ADTR2EN3 ADC Trigger 2 Source is PGxTRIGC Compare Event Enable bit

Value	Description
1	PGxTRIGC register compare event is enabled as a trigger source for ADC Trigger 2.
0	PGxTRIGC register compare event is disabled as a trigger source for ADC Trigger 2.

Bit 1 – ADTR2EN2 ADC Trigger 2 Source is PGxTRIGB Compare Event Enable bit

Value	Description
1	PGxTRIGB register compare event is enabled as a trigger source for ADC Trigger 2.
0	PGxTRIGB register compare event is disabled as a trigger source for ADC Trigger 2.

Bit 0 – ADTR2EN1 ADC Trigger 2 Source is PGxTRIGA Compare Event Enable bit

Value	Description
1	PGxTRIGA register compare event is enabled as trigger source for ADC Trigger 2.
0	PGxTRIGA register compare event is disabled as trigger source for ADC Trigger 2.

15.4.17. PWM Generator x F1 PCI 1 Register(1)

Name: PGxF1PCI1
Offset: 0x1068, 0x10DC, 0x1150, 0x11C4

Notes:
1. Caution should be used when modifying this register bit(s) while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWPCI control bit is used as the PCI Source.
3. This bit has no effect when the SWPCI control bit is used as the Termination Qualifier Source.

Bit	31	30	29	28	27	26	25	24
	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BPSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable bit

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits.
0	PCI function is not bypassed.

Bit 30 – SWTERM PCI Software Termination bit
A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 29 – PSYNC PCI Synchronization Control bit

Value	Description
1	PCI source is synchronized to PWM EOC.
0	PCI source is not synchronized to PWM EOC.

Bit 28 – PPS PCI Polarity Select bit(2)

Value	Description
1	Inverted
0	Not inverted

Bit 27 – TERMPS PCI Termination Polarity Select bit

Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < ‘101’.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection bits

Value	Description
111	Qualifier Latched Any Edge (setting 101, below, with PCI/Qualifier inputs swapped)
110	Qualifier Latched Rising Edge (setting 100, below, with the PCI/Qualifier inputs swapped)
101	Latched any edge
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control bit

Value	Description
1	Drives a ‘1’ to PCI logic assigned to by the SWPCIM[1:0] control bits.
0	Drives a ‘0’ to PCI logic assigned to by the SWPCIM[1:0] control bits.

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode bits

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic.
01	SWPCI bit is assigned to acceptance qualifier logic.
00	SWPCI bit is assigned to PCI acceptance logic.

Bit 20 – LATMOD PCI SR Latch Mode bit

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes.
0	SR latch is set-dominant in Latched Acceptance modes.

Bit 19 – TQPS Termination Qualifier Polarity Select bit**(3)**

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection bits

Value	Description
111	SWPCI control bit only (qualifier forced to ‘1’)
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No termination qualifier used (qualifier forced to ‘1’)

Bit 15 – TSYNCDIS Termination Synchronization Disable bit

Value	Description
1	Termination of latched PCI occurs immediately.
0	Termination of latched PCI occurs at PWM EOC.

Bits 14:12 – TERM[2:0] Termination Event Selection bits

Value	Description
111	Selects PCI Source #9
110	Selects PCI Source #8
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive
000	Manual Terminate: Terminate on a write of '1' to the SWTERM bit location

Bit 11 – AQPS Acceptance Qualifier Polarity Select bit(3)

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection bits

Value	Description
111	SWPCI control bit only (qualifier forced to '0')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No acceptance qualifier is used (qualifier forced to '1').

Bits 7:0 – BPSEL[7:0] PCI Bypass Source Selection bits

Notes:

1. If more than one bit is set, the selected bypass sources are OR'ed together.
2. Setting PCI control sourced from own generation is not allowed.

Value	Description
011	PCI control sourced from PWM Generator 4 PCI logic when BPEN = 1
010	PCI control sourced from PWM Generator 3 PCI logic when BPEN = 1
001	PCI control sourced from PWM Generator 2 PCI logic when BPEN = 1
000	PCI control sourced from PWM Generator 1 PCI logic when BPEN = 1

15.4.18. PWM Generator x F1 PCI 2 Register

Name: PGxF1PCI2
Offset: 0x106C, 0x10E0, 0x1154, 0x11C8

Notes:

1. Caution should be used when modifying register bits while PGxCON.ON = 1; unexpected results may occur.
2. If more than one bit is set, the selected PCI sources are OR'ed together.
3. If the PCI software control has a higher priority than PSS, if SWPCIM[1:0] = 2'b00 and SWPCI = 1'b1, a PCI event is generated regardless of PSS[31:0] content.

Bit	31	30	29	28	27	26	25	24
	PSS[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSS[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSS[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSS[31:0] PCI Source Selection bits[1,2,3]
Refer to Table 15-3 for device-specific PSS bit information.

15.4.19. PWM Generator x F2 PCI 1 Register

Name: PGxF2PCI1
Offset: 0x1070, 0x10E4, 0x1158, 0x11CC

Notes:
1. Caution should be used when modifying register bits while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWPCI control bit is used as the PCI Source.
3. This bit has no effect when the SWPCI control bit is used as the Termination Qualifier Source.

Bit	31	30	29	28	27	26	25	24
	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BPSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable bit

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits.
0	PCI function is not bypassed.

Bit 30 – SWTERM PCI Software Termination bit
A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 29 – PSYNC PCI Synchronization Control bit

Value	Description
1	PCI source is synchronized to PWM EOC.
0	PCI source is not synchronized to PWM EOC.

Bit 28 – PPS PCI Polarity Select bit[(2)]

Value	Description
1	Inverted
0	Not inverted

Bit 27 – TERMPS PCI Termination Polarity Select bit

Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection bits

Value	Description
111	Qualifier Latched Any Edge (setting 101, below, with PCI/Qualifier inputs swapped)
110	Qualifier Latched Rising Edge (setting 100, below, with the PCI/Qualifier inputs swapped)
101	Latched any edge(2)
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control bit

Value	Description
1	Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits.
0	Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits.

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode bits

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic.
01	SWPCI bit is assigned to acceptance qualifier logic.
00	SWPCI bit is assigned to PCI acceptance logic.

Bit 20 – LATMOD PCI SR Latch Mode bit

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes.
0	SR latch is set-dominant in Latched Acceptance modes.

Bit 19 – TQPS Termination Qualifier Polarity Select bit(3)

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection bits

Value	Description
111	SWPCI control bit only (qualifier forced to '1')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No termination qualifier used (qualifier forced to '1')

Bit 15 – TSYNCDIS Termination Synchronization Disable bit

Value	Description
1	Termination of latched PCI occurs immediately.
0	Termination of latched PCI occurs at PWM EOC.

Bits 14:12 – TERM[2:0] Termination Event Selection bits

Value	Description
111	Selects PCI Source #9
110	Selects PCI Source #8
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive(3).
000	Manual Terminate: Terminate on a write of '1' to the SWTERM bit location.

Bit 11 – AQPS Acceptance Qualifier Polarity Select bit(3)

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection bits

Value	Description
111	SWPCI control bit only (qualifier forced to '0')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No acceptance qualifier is used (qualifier forced to '1').

Bits 7:0 – BPSEL[7:0] PCI Bypass Source Selection bits

Notes:

1. If more than one bit is set, the selected bypass sources are OR'ed together.
2. Setting PCI control sourced from own generation is not allowed.

Value	Description
BPSEL[7]	PCI control sourced from PWM Generator 8 PCI logic when BPEN = 1
BPSEL[6]	PCI control sourced from PWM Generator 7 PCI logic when BPEN = 1
BPSEL[5]	PCI control sourced from PWM Generator 6 PCI logic when BPEN = 1
BPSEL[4]	PCI control sourced from PWM Generator 5 PCI logic when BPEN = 1
BPSEL[3]	PCI control sourced from PWM Generator 4 PCI logic when BPEN = 1
BPSEL[2]	PCI control sourced from PWM Generator 3 PCI logic when BPEN = 1
BPSEL[1]	PCI control sourced from PWM Generator 2 PCI logic when BPEN = 1
BPSEL[0]	PCI control sourced from PWM Generator 1 PCI logic when BPEN = 1

15.4.20. PWM Generator x F2 PCI 2 Register

Name: PGxF2PCI2
Offset: 0x1074, 0x10E8, 0x115C, 0x11D0

Notes:
1. Caution should be used when modifying register bits while PGxCON.ON = 1; unexpected results may occur.
2. If more than one bit is set, the selected PCI sources are OR'ed together.
3. If the PCI software control has a higher priority than PSS, if SWPCIM[1:0] = 2'b00 and SWPCI = 1'b1, a PCI event is generated regardless of PSS[31:0] content.

Bit	31	30	29	28	27	26	25	24
	PSS[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSS[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSS[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSS[31:0] PCI Source Selection bits[1,2,3]
Refer to Table 15-3 for device-specific PSS bit information.

15.4.21. PWM Generator x CL PCI 1 Register

Name: PGxCLPCI1
Offset: 0x1078, 0x10EC, 0x1160, 0x11D4

Notes:
1. Caution should be used when modifying these register bits while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWPCI control bit is used as the PCI Source.
3. This bit has no effect when the SWPCI control bit is used as the Termination Qualifier Source.

Bit	31	30	29	28	27	26	25	24
	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BPSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable bit

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; the PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits.
0	PCI function is not bypassed.

Bit 30 – SWTERM PCI Software Termination bit

A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 29 – PSYNC PCI Synchronization Control bit

Value	Description
1	PCI source is synchronized to PWM EOC.
0	PCI source is not synchronized to PWM EOC.

Bit 28 – PPS PCI Polarity Select bit(2)

Value	Description
1	Inverted
0	Not inverted

Bit 27 – TERMPS PCI Termination Polarity Select bit

Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection bits

Value	Description
111	Qualifier Latched Any Edge (setting 101, below, with PCI/Qualifier inputs swapped)
110	Qualifier Latched Rising Edge (setting 100, below, with the PCI/Qualifier inputs swapped)
101	Latched any edge(2)
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control bit

Value	Description
1	Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits.
0	Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits.

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode bits

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic.
01	SWPCI bit is assigned to acceptance qualifier logic.
00	SWPCI bit is assigned to PCI acceptance logic.

Bit 20 – LATMOD PCI SR Latch Mode bit

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes.
0	SR latch is set-dominant in Latched Acceptance modes.

Bit 19 – TQPS Termination Qualifier Polarity Select bit(3)

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection bits

Note: Polarity control bit, TQPS, has no effect on these selections.

Value	Description
111	SWPCI control bit only (qualifier forced to '1')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No termination qualifier used (qualifier forced to '1')

Bit 15 – TSYNCDIS Termination Synchronization Disable bit

Value	Description
1	Termination of latched PCI occurs immediately.
0	Termination of latched PCI occurs at PWM EOC.

Bits 14:12 – TERM[2:0] Termination Event Selection bits

Note: Do not use this selection when the ACP[2:0] bits (PGxyPCI1[26:24]) are set for latched on any edge.

Value	Description
111	Selects PCI Source #9
110	Selects PCI Source #8
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive(3).
000	Manual Terminate: Terminate on a write of ‘1’ to the SWTERM bit location.

Bit 11 – AQPS Acceptance Qualifier Polarity Select bit(3)

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection bits

Value	Description
111	SWPCI control bit only (qualifier forced to ‘0’)
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No acceptance qualifier is used (qualifier forced to ‘1’).

Bits 7:0 – BPSEL[7:0] PCI Bypass Source Selection bits

Notes:

1. If more than one bit is set, the selected bypass sources are ORed together.
2. Setting PCI control sourced from its own generation is not allowed.

Value	Description
111	PCI control sourced from PWM Generator 8 PCI logic when BPEN = 1
110	PCI control sourced from PWM Generator 7 PCI logic when BPEN = 1
101	PCI control sourced from PWM Generator 6 PCI logic when BPEN = 1
100	PCI control sourced from PWM Generator 5 PCI logic when BPEN = 1
011	PCI control sourced from PWM Generator 4 PCI logic when BPEN = 1
010	PCI control sourced from PWM Generator 3 PCI logic when BPEN = 1
001	PCI control sourced from PWM Generator 2 PCI logic when BPEN = 1
000	PCI control sourced from PWM Generator 1 PCI logic when BPEN = 1

15.4.22. PWM Generator x CL PCI 2 Register

Name: PGxCLPCI2
Offset: 0x107C, 0x10F0, 0x1164, 0x11D8

Notes:
1. Caution should be used when modifying this register bit(s) while PGxCON.ON = 1; unexpected results may occur.
2. If more than one bit is set, the selected PCI sources are OR'ed together.
3. If the PCI software control has a higher priority than PSS, and if SWPCIM[1:0] = 2'b00 and SWPCI = 1'b1, a PCI event is generated regardless of PSS[31:0] content.

Bit	31	30	29	28	27	26	25	24
	PSS[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSS[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSS[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSS[31:0] PCI Source Selection bits[1,2,3]
Refer to Table 15-3 for device-specific PSS bit information.

15.4.23. PWM Generator x FF PCI 1 Register

Name: PGxFFPCI1
Offset: 0x1080, 0x10F4, 0x1168, 0x11DC

Notes:
1. Caution should be used when modifying register bits while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWPCI control bit is used as the PCI Source.
3. This bit has no effect when the SWPCI control bit is used as the Termination Qualifier Source.

Bit	31	30	29	28	27	26	25	24
	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BPSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable bit

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits.
0	PCI function is not bypassed.

Bit 30 – SWTERM PCI Software Termination bit

A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 29 – PSYNC PCI Synchronization Control bit

Value	Description
1	PCI source is synchronized to PWM EOC.
0	PCI source is not synchronized to PWM EOC.

Bit 28 – PPS PCI Polarity Select bit(2)

Value	Description
1	Inverted
0	Not inverted

Bit 27 – TERMPS PCI Termination Polarity Select bit

Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection bits

Value	Description
111	Qualifier Latched Any Edge (setting 101, below, with PCI/Qualifier inputs swapped)
110	Qualifier Latched Rising Edge (setting 100, below, with the PCI/Qualifier inputs swapped)
101	Latched any edge(2)
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control bit

Value	Description
1	Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits.
0	Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits.

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode bits

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic.
01	SWPCI bit is assigned to acceptance qualifier logic.
00	SWPCI bit is assigned to PCI acceptance logic.

Bit 20 – LATMOD PCI SR Latch Mode bit

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes.
0	SR latch is set-dominant in Latched Acceptance modes.

Bit 19 – TQPS Termination Qualifier Polarity Select bit(3)

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection bits

Note: Polarity control bit, TQPS, has no effect on these selections.

Value	Description
111	SWPCI control bit only (qualifier forced to '1')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No termination qualifier used (qualifier forced to '1')

Bit 15 – TSYNCDIS Termination Synchronization Disable bit

Value	Description
1	Termination of latched PCI occurs immediately.
0	Termination of latched PCI occurs at PWM EOC.

Bits 14:12 – TERM[2:0] Termination Event Selection bits

Note: Do not use this selection when the ACP[2:0] bits (PGxyPCI1[26:24]) are set for latched on any edge.

Value	Description
111	Selects PCI Source #9
110	Selects PCI Source #8
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive(3).
000	Manual Terminate: Terminate on a write of ‘1’ to the SWTERM bit location.

Bit 11 – AQPS Acceptance Qualifier Polarity Select bit(3)

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection bits

Value	Description
111	SWPCI control bit only (qualifier forced to ‘0’)
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No acceptance qualifier is used (qualifier forced to ‘1’).

Bits 7:0 – BPSEL[7:0] PCI Bypass Source Selection bits

Notes:

1. If more than one bit is set, the selected bypass sources are OR’ed together.
2. Setting PCI control sourced from own generation is not allowed.

Value	Description
011	PCI control sourced from PWM Generator 4 PCI logic when BPEN = 1
010	PCI control sourced from PWM Generator 3 PCI logic when BPEN = 1
001	PCI control sourced from PWM Generator 2 PCI logic when BPEN = 1
000	PCI control sourced from PWM Generator 1 PCI logic when BPEN = 1

15.4.24. PWM Generator x FF PCI 2 Register

Name: PGxFFCI2
Offset: 0x1084, 0x10F8, 0x116C, 0x11E0

Notes:
1. Caution should be used when modifying register bits while PGxCON.ON = 1; unexpected results may occur.
2. If more than one bit is set, the selected PCI sources are OR'ed together.
3. If the PCI software control has a higher priority than PSS, if SWPCIM[1:0] = 2'b00 and SWPCI = 1'b1, a PCI event is generated regardless of PSS[31:0] content.

Bit	31	30	29	28	27	26	25	24
	PSS[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSS[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSS[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSS[31:0] PCI Source Selection bits[1,2,3]
Refer to Table 15-3 for device-specific PSS bit information.

15.4.25. PWM Generator x SP PCI 1 Register

Name: PGxSPCI1
Offset: 0x1088, 0x10FC, 0x1170, 0x11E4

Notes:

1. Caution should be used when modifying register bits while PGxCON.ON = 1; unexpected results may occur.
2. This bit has no effect when the SWPCI control bit is used as the PCI Source.
3. This bit has no effect when the SWPCI control bit is used as the Termination Qualifier Source.

Bit	31	30	29	28	27	26	25	24
	BPEN	SWTERM	PSYNC	PPS	TERMPS	ACP[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SWPCI	SWPCIM[1:0]		LATMOD	TQPS	TQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSYNCDIS	TERM[2:0]			AQPS	AQSS[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BPSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – BPEN PCI Bypass Enable bit

Value	Description
1	PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits.
0	PCI function is not bypassed.

Bit 30 – SWTERM PCI Software Termination bit

A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 29 – PSYNC PCI Synchronization Control bit

Value	Description
1	PCI source is synchronized to PWM EOC.
0	PCI source is not synchronized to PWM EOC.

Bit 28 – PPS PCI Polarity Select bit[(2)]

Value	Description
1	Inverted
0	Not inverted

Bit 27 – TERMPS PCI Termination Polarity Select bit

Note: This bit has no effect when the SWTERM control bit is used as the PCI Termination Event or if TERM[2:0] < '101'.

Value	Description
1	Inverted
0	Not inverted

Bits 26:24 – ACP[2:0] PCI Acceptance Criteria Selection bits

Value	Description
111	Qualifier Latched Any Edge (setting 101, below, with PCI/Qualifier inputs swapped)
110	Qualifier Latched Rising Edge (setting 100, below, with the PCI/Qualifier inputs swapped)
101	Latched any edge(2)
100	Latched rising edge
011	Latched
010	Any edge
001	Rising edge
000	Level-sensitive

Bit 23 – SWPCI Software PCI Control bit

Value	Description
1	Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits.
0	Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits.

Bits 22:21 – SWPCIM[1:0] Software PCI Control Mode bits

Value	Description
11	Reserved
10	SWPCI bit is assigned to termination qualifier logic.
01	SWPCI bit is assigned to acceptance qualifier logic.
00	SWPCI bit is assigned to PCI acceptance logic.

Bit 20 – LATMOD PCI SR Latch Mode bit

Value	Description
1	SR latch is Reset-dominant in Latched Acceptance modes.
0	SR latch is set-dominant in Latched Acceptance modes.

Bit 19 – TQPS Termination Qualifier Polarity Select bit(3)

Value	Description
1	Inverted
0	Not inverted

Bits 18:16 – TQSS[2:0] Termination Qualifier Source Selection bits

Note: Polarity control bit, TQPS, has no effect on these selections.

Value	Description
111	SWPCI control bit only (qualifier forced to '1')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No termination qualifier used (qualifier forced to '1')

Bit 15 – TSYNCDIS Termination Synchronization Disable bit

Value	Description
1	Termination of latched PCI occurs immediately.
0	Termination of latched PCI occurs at PWM EOC.

Bits 14:12 – TERM[2:0] Termination Event Selection bits

Note: Do not use this selection when the ACP[2:0] bits (PGxyPCI1[26:24]) are set to latch on any edge.

Value	Description
111	Selects PCI Source #9
110	Selects PCI Source #8
101	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
100	PGxTRIGC trigger event
011	PGxTRIGB trigger event
010	PGxTRIGA trigger event
001	Auto-Terminate: Terminate when PCI source transitions from active to inactive[3].
000	Manual Terminate: Terminate on a write of '1' to the SWTERM bit location.

Bit 11 – AQPS Acceptance Qualifier Polarity Select bit[3]

Value	Description
1	Inverted
0	Not inverted

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection bits

Value	Description
111	SWPCI control bit only (qualifier forced to '0')
110	Selects PCI Source #9
101	Selects PCI Source #8
100	Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits)
011	PWM Generator is triggered.
010	LEB is active.
001	Duty cycle is active (base PWM Generator signal).
000	No acceptance qualifier is used (qualifier forced to '1').

Bits 7:0 – BPSEL[7:0] PCI Bypass Source Selection bits

Notes:

1. If more than one bit is set, the selected bypass sources are ORed together.
2. Setting PCI control sourced from its own generation is not allowed.

Value	Description
BPSEL[7]	PCI control sourced from PWM Generator 8 PCI logic when BPEN = 1
BPSEL[6]	PCI control sourced from PWM Generator 7 PCI logic when BPEN = 1
BPSEL[5]	PCI control sourced from PWM Generator 6 PCI logic when BPEN = 1
BPSEL[4]	PCI control sourced from PWM Generator 5 PCI logic when BPEN = 1
BPSEL[3]	PCI control sourced from PWM Generator 4 PCI logic when BPEN = 1
BPSEL[2]	PCI control sourced from PWM Generator 3 PCI logic when BPEN = 1
BPSEL[1]	PCI control sourced from PWM Generator 2 PCI logic when BPEN = 1
BPSEL[0]	PCI control sourced from PWM Generator 1 PCI logic when BPEN = 1

15.4.26. PWM Generator x S PCI 2 Register

Name: PGxSPCI2
Offset: 0x108C, 0x1100, 0x1174, 0x11E8

Notes:
1. Caution should be used when modifying register bits while PGxCON.ON = 1; unexpected results may occur.
2. If more than one bit is set, the selected PCI sources are ORed together.
3. If the PCI software control has a higher priority than PSS, and if SWPCIM[1:0] = 2'b00 and SWPCI = 1'b1, a PCI event is generated regardless of PSS[31:0] content.

Bit	31	30	29	28	27	26	25	24
	PSS[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSS[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSS[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSS[31:0] PCI Source Selection bits[1,2,3]
Refer to Table 15-3 for device-specific PSS bit information.

15.4.27. PWM Generator x Leading-Edge Blanking Register[1]

Name: PGxLEB
Offset: 0x1090, 0x1104, 0x1178, 0x11EC

Notes:

1. Caution should be exercised when modifying this register while PGxCON.ON = 1; unexpected results may occur.
2. TPWMCLK = 1/FPGx_clk

Bit	31	30	29	28	27	26	25	24
	TRGD	TRGC	TRGB	TRGA	PHR	PHF	PLR	PLF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
					LEB[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LEB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LEB[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit 31 – TRGD PGxTRIGD Kick-Off bit

Value	Description
1	PGxTRIGD compare event will trigger LEB duration counter.
0	LEB ignores PGxTRIGD compare event.

Bit 30 – TRGC PGxTRIGC Kick-Off bit

Value	Description
1	PGxTRIGC compare event will trigger LEB duration counter.
0	LEB ignores PGxTRIGC compare event.

Bit 29 – TRGB PGxTRIGB Kick-Off bit

Value	Description
1	PGxTRIGB compare event will trigger a LEB duration counter.
0	LEB ignores the PGxTRIGB compare event.

Bit 28 – TRGA PGxTRIGA Kick-Off bit

Value	Description
1	PGxTRIGA compare event will trigger the LEB duration counter.
0	LEB ignores PGxTRIGA compare event.

Bit 27 – PHR PWMx Rising Edge Trigger Enable bit

Value	Description
1	Rising edge of PWMx will trigger the LEB duration counter.
0	LEB ignores the rising edge of PWMx.

Bit 26 – PHF PWMxH Falling Edge Trigger Enable bit

Value	Description
1	Falling edge of PWMx will trigger the LEB duration counter.
0	LEB ignores the falling edge of PWMx.

Bit 25 – PLR PWMx Rising Edge Trigger Enable bit

Value	Description
1	Rising edge of PWMx will trigger the LEB duration counter.
0	LEB ignores the rising edge of PWMx.

Bit 24 – PLF PWMx Falling Edge Trigger Enable bit

Value	Description
1	Falling edge of PWMx will trigger the LEB duration counter.
0	LEB ignores the falling edge of PWMx.

Bits 19:4 – LEB[19:4] Leading-Edge Blanking Period bits

These bits select the leading edge blanking period. The 4 LSbs of the blanking time are not implemented, providing a blanking resolution of TPWMCLK. The LEB period is (LEB[15:4]+1)*TPWMCLK. The minimum blanking period is 5, values 0,1,2,3,4 all get LEB period of 5*TPWMCLK.

Note: TPWMCLK = $1/F_{PGx\_clk}$

15.4.28. PWM Generator x Phase Register

Name: PGxPHASE
Offset: 0x1094, 0x1108, 0x117C, 0x11F0

Notes:

1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. If HREN = 0, the four least significant bits are read as '0000'.
3. In variable phase PWM, if PGxPHASE + PGxDC ≥ PGxPER, the falling edge of the PWM signal will be terminated at the period boundary as intended.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					PHASE[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PHASE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PHASE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:0 – PHASE[19:0] PWM Generator x Phase Register bits(1,2,3)

15.4.29. PWM Generator x Duty Cycle Register

Name: PGxDC
Offset: 0x1098, 0x110C, 0x1180, 0x11F4

Notes:

1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. If HREN = 0, the four least significant bits are read as ‘0000’.
3. In variable phase PWM, if PGxPHASE + PGxDC≥ PGxPER, the falling edge of the PWM signal will be terminated at the period boundary as intended.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					DC[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:0 – DC[19:0] PWM Generator x Duty Cycle Register bits(1,2,3,4)

15.4.30. PWM Generator x Duty Cycle Adjustment Register

Name: PGxDCA
Offset: 0x109C, 0x1110, 0x1184, 0x11F8

Note: This register cannot be modified while PGxSTAT.UPDATE = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					DCA[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCA[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bits 11:8 – DCA[11:8] PWM Generator x Duty Cycle Adjustment Value bits
Depending on the state of the selected PCI source, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle. When the PCI source is active, PGxDCA is added.

Bits 11:8 – DCA[11:8] PWM Generator x Duty Cycle Adjustment Value bits
Depending on the state of the selected PCI source, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle. When the PCI source is active, PGxDCA is added.

Bits 7:4 – DCA[7:4] PWM Generator x Duty Cycle Adjustment Value bits
Depending on the state of the selected PCI source, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle. When the PCI source is active, PGxDCA is added.

15.4.31. PWM Generator x Period Register

Name: PGxPER
Offset: 0x10A0, 0x1114, 0x1188, 0x11FC

Notes:
1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. If HREN = 0, the four least significant bits are read as '0000'.

Note:

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					PER[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:0 – PER[19:0] PWM Generator x Period Register bits(1,2)

15.4.32. PWM Generator x Trigger A Register

Name: PGxTRIGA
Offset: 0x10A4, 0x1118, 0x118C, 0x1200

Notes:

1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. If HREN = 0, the four least significant bits are read as ‘0000’.
3. The four least significant bits are R/W accessible when HREN = 1. This enables fine-edge placement in Independent Edge mode.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGA[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRIGA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRIGA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – CAHALF Specifies Where the Trigger Compare Time Occurs bit

Value	Description
1	The second phase of the center-aligned period.
0	The first phase of the center-aligned period.

Bits 19:0 – TRIGA[19:0] PWM Generator x Trigger A bits

15.4.33. PWM Generator x Trigger B Register

Name: PGxTRIGB
Offset: 0x10A8, 0x111C, 0x1190, 0x1204

Notes:
1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. If HREN = 0, the four least significant bits are read as '0000'.
3. The four least significant bits are R/W accessible when HREN = 1. This enables fine-edge placement in Independent Edge mode.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGB[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRIGB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRIGB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – CAHALF Specifies Where the Trigger Compare Time Occurs bit

Value	Description
1	The second phase of the center-aligned period.
0	The first phase of the center-aligned period.

Bits 19:0 – TRIGB[19:0] PWM Generator x Trigger B bits

15.4.34. PWM Generator x Trigger C Register[1,2]

Name: PGxTRIGC
Offset: 0x10AC, 0x1120, 0x1194, 0x1208

Notes:
1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. The content of this register can be automatically updated as part of the LLC mode of operation.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGC[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRIGC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRIGC[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit 31 – CAHALF Specifies Where the Trigger Compare Time Occurs bit

Value	Description
1	The second phase of the center-aligned period.
0	The first phase of the center-aligned period.

Bits 19:16 – TRIGC[19:16]

Bits 15:8 – TRIGC[15:8]

Bits 7:4 – TRIGC[7:4] PWM Generator x Trigger C bits

15.4.35. PWM Generator x Trigger D Register[1,2]

Name: PGxTRIGD
Offset: 0x10B0, 0x1124, 0x1198, 0x120C

Notes:
1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. The content of this register can be automatically updated as part of the LLC mode of operation.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGD[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRIGD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRIGD[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit 31 – CAHALF Specifies Where the Trigger Compare Time Occurs bit

Value	Description
1	The second phase of the center-aligned period.
0	The first phase of the center-aligned period.

Bits 19:4 – TRIGD[19:4] PWM Generator x Trigger D bits

15.4.36. PWM Generator x Trigger E Register[1,2]

Name: PGxTRIGE
Offset: 0x10B4, 0x1128, 0x119C, 0x1210

Notes:
1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. The content of this register can be automatically updated as part of the LLC mode of operation.
3. If HREN = 0, the four least significant bits are read as '0000'.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGE[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRIGE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TRIGE[7:4]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit 31 – CAHALF Specifies Where the Trigger Compare Time Occurs bit

Value	Description
1	The second phase of the center-aligned period.
0	The first phase of the center-aligned period.

Bits 19:4 – TRIGE[19:4] PWM Generator x Trigger E bits

15.4.37. PWM Generator x Trigger F Register(1,2,3)

Name: PGxTRIGF
Offset: 0x10B8, 0x112C, 0x11A0, 0x1214

Notes:
1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. The content of this register can be automatically updated as part of the LLC mode of operation. The value of TRIGF determines the maximum on-time of the PWMxL output in Complementary Output mode.
3. The content of this register is reset at the end of every cycle.
4. If HREN = 0, the four least significant bits are read as '0000'.
5. The four least significant bits are R/W when HREN = 1; this is for fine edge placement in independent edge mode.

Bit	31	30	29	28	27	26	25	24
	CAHALF							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
					TRIGF[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					1	1	1	1

Bit	15	14	13	12	11	10	9	8
	TRIGF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	TRIGF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	0	0	0	0

Bit 31 – CAHALF Specifies Where the Trigger Compare Time Occurs bit

Value	Description
1	The second phase of the center-aligned period.
0	The first phase of the center-aligned period.

Bits 19:0 – TRIGF[19:0] PWM Generator x Trigger F bits

15.4.38. PWM Generator x Dead-Time Register

Name: PGxDT
Offset: 0x10BC, 0x1130, 0x11A4, 0x1218

Notes:
1. This register cannot be modified while PGxSTAT.UPDATE = 1.
2. If HREN = 0, the four least significant bits are read as '0000'.

Bit	31	30	29	28	27	26	25	24
		DTH[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DTH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		DTL[14:8]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DTL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 30:16 – DTH[14:0] PWMx Dead-Time Delay bits[1,2]

Bits 14:0 – DTL[14:0] PWMxL Dead-Time Delay bits[1,2]

15.4.39. PWM Generator x Capture Register

Name: PGxCAP
Offset: 0x10C0, 0x1134, 0x11A8, 0x121C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
					CAP[19:16]			
Access					R	R	R	R
Reset					0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	CAP[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	CAP[7:4]							PGxCAP
Access	R	R	R	R				R
Reset	0	0	0	0				0

Bits 19:4 – CAP[19:4] PGx Time Base Capture bits
Note: A capture event can be manually initiated in software by writing a '1' to PGxCAP[0].
The CAP bit (PGxSTAT[5]) will indicate when a new capture value is available. A read of PGxCAP will automatically clear the CAP bit and allow a new capture event to occur. PGxCAP[3:0] will always read as '0'.

Bit 0 – PGxCAP PWM Timebase Manual Capture Enable bit
This bit can be set high to manually invoke a time base capture event. However, it is always read as 0.

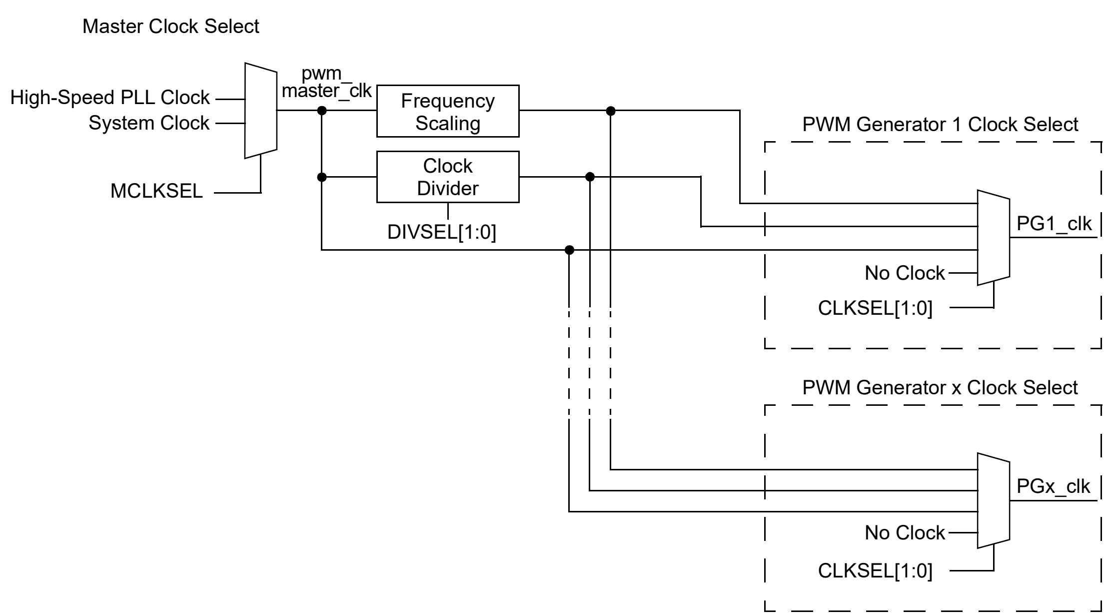
15.5. Operation

15.5.1. PWM Clocking

15.5.1.1. Master Clocking

The PWM module provides several clocking features at the top level of the module. Each PWM Generator can then independently select one of the clock sources, as shown in Figure 15-4. The clock input into the PWM module is selected with the MCLKSEL control bit (PCLKCON[0]). The CLKSEL[1:0] control bits (PGxCON[4:3]) are used to select the clock for each PWM Generator instance; see PWM Generator Clocking for details. Frequency scaling and the clock divider are discussed in Shared Clocking. The CLKSELx bits need to be changed from the default selection to allow the PWM Generator to function.

Figure 15-4. PWM Generator Clocking



Note: Writing MCLKSEL to a non-zero value will request and enable the chosen clock source, whether any PWM Generators are enabled or not. This allows a PLL, for example, to be requested and warmed up before using it as a PWM clock source. For the lowest device power consumption, the MCLKSEL bits should be set to the value, ‘`00`’, if all PWM Generators have been disabled. Changing the MCLKSEL or CLKSEL[1:0] bits while ON (PGxCON[15]) = `1`) is not recommended.

Note: The CPU and PWM typically run at different clock speeds depending on the application requirements. If the PWM clock speed is equal or slower than the CPU, writes to registers may have delayed behavior. For example, if SWTERM is used to clear a Fault, the instruction may need to be stretched with a `REPEAT` instruction to ensure the PWM can detect the edge within its clock cycle.

15.5.1.2. Clocking Equations

Some modes of operation utilize multiple period matches to complete one PWM cycle. The following equations should be used to determine the values needed to output the PWM as desired. Different operating modes use different equations.

Equation 15-1. PWM Period Calculation

Edge-Aligned and Variable Phase Modes

$$F_{PWM} = \frac{16 \bullet F_{PGx\_clk}}{PGxPER + 16}$$

$$PGxPER = \frac{F_{PGx\_clk}}{F_{PWM}} - 16$$

Where:
F_{PWM} = PWM Output Switching Frequency
PWM Period = $1/F_{PWM}$

**Center-Aligned Modes,
Edge-Aligned and Variable Phase Modes
with Push-Pull Output Mode**

$$F_{PWM} = \frac{8 \bullet F_{PGx\_clk}}{(PGxPER + 16)}$$

$$PGxPER = \frac{F_{PGx\_clk}}{F_{PWM}} - 16$$

**Center-Aligned Modes
with Push-Pull Output Mode**

$$F_{PWM} = \frac{4 \bullet F_{PGx\_clk}}{(PGxPER + 16)}$$

$$PGxPER = \frac{F_{PGx\_clk}}{F_{PWM}} - 16$$

Equation 15-2. PWM Duty Cycle, Phase, Trigger and Dead-Time Calculations, Standard

$$MDC \text{ or } PGxDC(A) = (PGxPER + 16) \cdot Duty\ Cycle$$

Where:
Duty Cycle is % between 0 and 100

$$MPHASE \text{ or } PGxPHASE = 16 \cdot F_{PGx\_clk} \cdot Phase$$

$$PGxTRIGy = 16 \cdot F_{PGx\_clk} \cdot Trigger\ Offset$$
$(y = A, B \text{ or } C)$

$$PGxDTy = 16 \cdot F_{PGx\_clk} \cdot Dead\ Time$$
$(y = H \text{ or } L)$

Where:
Phase, Trigger Offset and *Dead Time* are specified in time units (ms, µs or ns)

Resolution

15.5.1.3. Clocking Synchronization

Due to the separate clocking domains of the PWM module and the CPU's system clock, there are inherent synchronization delays associated with SFR reads. This delay is dependent on the relative speeds of the CPU (sys_clk) and the PWM Generator clock (PGx_clk). Typically, the CPU clock will be slower, and SFR data can be delayed up to one sys_clk cycle. It is also important to note that each PWM Generator can run at a different speed, and this can have an effect on interactions between PWM Generators.

15.5.2. PWM Generator (PG) Features

Most of the features and controls of the PWM module are at the PWM Generator level and are controlled using each PWM Generator's SFRs. PWM Generator operation is based on triggers. The PWM Generator must receive a Start-of-Cycle (SOC) Trigger signal to generate each PWM cycle. The trigger signal may be generated outside of the PWM Generator, or the PWM Generator may be self-triggered. When a PWM Generator reaches the end of a PWM cycle, it produces an End-of-Cycle (EOC) trigger that can be used by other PWM Generators.

If multiple PWM Generators run at different frequencies, the triggers can be synchronized using the PCI Sync block.

15.5.2.1. PWM Generator Clocking

Each PWM Generator can be clocked independently of one another for maximum flexibility. There are four clock options available selected by the CLKSEL[1:0] bits (PGxCON[4:3]):

1. No clock (lowest power state)
2. Output of MCLKSEL
3. Output of clock divider
4. Output frequency scaler

This configuration flexibility allows, for example, one group of PWM Generators to operate at a higher frequency, while another group of PWM Generators operates at a lower frequency.

Note: Do not change the MCLKSEL or CLKSEL[1:0] bits while the PWM Generator is in operation (ON (PGxCON[15]) = 1).

15.5.2.2. PWM Operating Modes

The PWM module supports a wide range of PWM modes for both motor control and power supply designs. The following PWM modes are supported:

- Independent Edge PWM mode (default)
- Variable Phase PWM mode
- Independent Edge PWM mode, Dual Output
- Center-Aligned PWM mode
- Double Update Center-Aligned PWM mode
- Dual Edge Center-Aligned PWM mode
- LLC Resonant Converter mode

The PWM modes are selected by setting the MODSEL[2:0] bits (PGxCON[2:0]). Some modes utilize multiple time base cycles to complete a single PWM cycle. Refer to Equation 15-2 for specifics on timing.

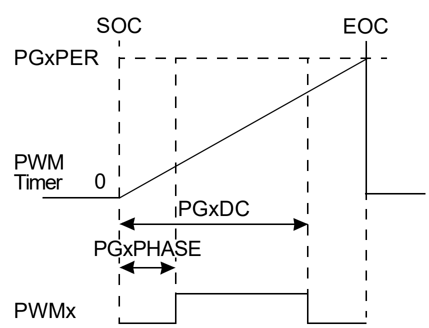
15.5.2.2.1. Independent Edge PWM Mode

Independent Edge PWM mode is used for many applications and can be used to create edge-aligned PWM signals as well as PWM signals with arbitrary phase offsets. This mode is the default operating mode of the PWM generator and is selected when MODSEL[2:0] = `000` (PGxCON[2:0]). Two data registers must be written to define the rising and falling edges. The characteristics of the PWM signal are determined by these three SFRs:

- PGxPHASE: Determines the position of the PWM signal rising edge from the start of the timer count cycle.
- PGxDC: Determines the position of the PWM signal falling edge from the start of the timer count cycle.
- PGxPER: Determines the end of the PWM timer count cycle.

A basic Edge-Aligned PWM mode signal is created by setting PGxPHASE = `0`. Alternatively, multiple PWM generators can be synchronized with one another by using the same PGxPHASE value. A constant value equivalent to the PGxPHASE value of other PWM generators can be used to synchronize multiple PGs. The duty cycle is varied by writing to the PGxDC register. Arbitrary phase PWM signals may be generated by writing to PGxPHASE and PGxDC with the appropriate values. If PGxPHASE = PGxDC, no PWM pulse will be produced. If PGxDC ≥ PGxPER, a 100% duty cycle pulse is produced. If PGxPHASE + PGxDC > PGxPER, the falling edge is terminated at the EOC. Figure 15-5 shows the relationship between the control SFRs and the output waveform.

Figure 15-5. Independent Edge PWM Mode



Multiple PWM generators can be synchronized to make a multiphase system. There are multiple methods that can be used in different applications and PWM modes. SOC triggers can be shared across any PWM generator.

In this example, three PWM generators are set to start and run synchronized:

- Configure all PWM generators for the application (clk, PER, DC, etc.).
- PG1 is self-triggered SOCS = 0b0000.

- PG2 SOCS trigger is PG1, SOCS = 0b0001.
- PG3 SOCS trigger is PG2, SOCS = 0b0010.
- PG2 and PG3 ON bits are set; cycles will not start until triggered by PWM1.
- PG1 ON bit is set; all PWM generators start simultaneously.

In this example, a daisy-chain triggering scheme is used to synchronize and create an offset between each PWM generator's SOC:

1. Configure all PWM generators for the application (clk, PER, DC, etc.).
2. PG1 is self-triggered, SOCS = 0b0000.
3. PG1TRIGA is set to provide a phase offset for PG1 to PG2.
4. PG1 PTGRGSEL = 0001 to select TRIGA as the trigger.
5. PG2 SOCS trigger is PG1, SOCS = 0b0001.
6. PG2 TRIGA is set to provide a phase offset for PG2 to PG3.
7. PG2 PTGRGSEL = 0001 to select TRIGA as the trigger.
8. PG3 SOCS trigger is PG2, SOCS = 0b0010.
9. PG2 and PG3 ON bits are set; cycles will not start until triggered by PWM1.
10. PG1 ON bit is set.

15.5.2.2.2. Variable Phase PWM Mode

The Variable Phase PWM mode differs from Independent Edge mode in that one register is used to select the phase offset from the Start-of-Cycle, and a second register is used to select the width of the pulse. The Variable Phase PWM mode is useful as the PGxDC register is programmed to a constant value, while the PGxPHASE value is modulated. The PWM logic will automatically calculate rising edge and falling edge times to maintain a constant pulse width. Similarly, the user can leave the PGxPHASE register programmed to a constant value to create signals with a constant phase offset and variable duty cycle. The Variable Phase PWM mode is selected when MODSEL[2:0] (PGxCON[2:0]) = `001`. The characteristics of the PWM signal are determined by these three SFRs:

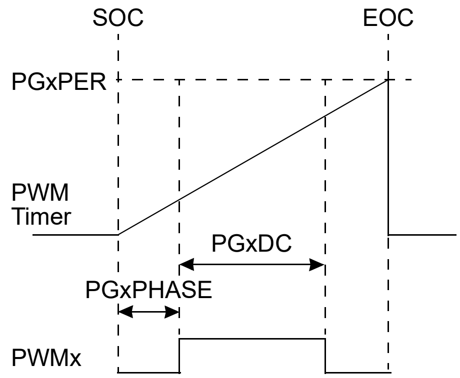
- PGxPHASE: Determines the offset of the PWM signal rising edge from the start of the timer cycle.
- PGxDC: Determines the width of the PWM pulse and location of the PWM signal falling edge.
- PGxPER: Determines the end of the PWM timer count cycle.

When updating PER in Variable Phase mode, ensure that the client PWM Generator PER > host PER. The recommended method is client PER = host PER + max PHASE to ensure that the client PWM Generator does not generate its own SOC.

The host will always provide an SOC to trigger the client PWM Generators. If the host PER is lengthened such that client PER > host PER, a client PWM cycle may be missed as the host SOC trigger arrives when the client cycle is in progress and the trigger is ignored.

Figure 15-6 shows the relationship between the control SFRs and the output waveform.

Figure 15-6. Variable Phase PWM Mode

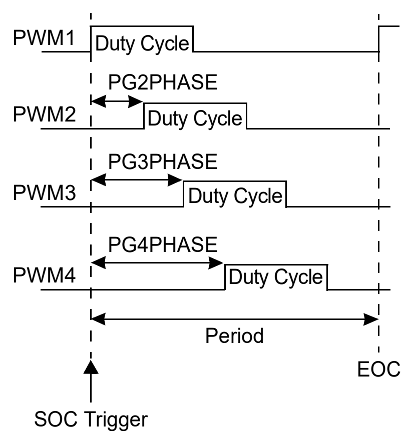


The Master Duty Cycle SFR (MDC) can also be used to change the duty cycle of all phases with a single register write. An example of a multiphase system is shown in Figure 15-7. Variable Phase mode cannot support active duty cycles across EOC boundaries. Ensure Phase + DC ≤ Period to allow for completion of the duty cycle for EOC.

In this scenario, PG1 provides SOC triggers to PG2-PG4. The output wave form is shown in Figure 15-7.

1. Set all PGs to same clock source.
2. Set all PGs in Variable Phase mode, MODSEL = 0b001.
3. Set PG1 for self trigger, SOCS = 0b0000.
4. Set PG2-PG4 to use PG1's SOC trigger, SOCS = 0b0001.
5. Write initial PER, DC and PHASE to all PGs.
6. Enable outputs, PENx = '1'.
7. Set ON = '1' of PG2-PG4. No cycles will start until PG1 sends trigger.
8. Set ON = '1' of PG1. This will start all PG synchronously.

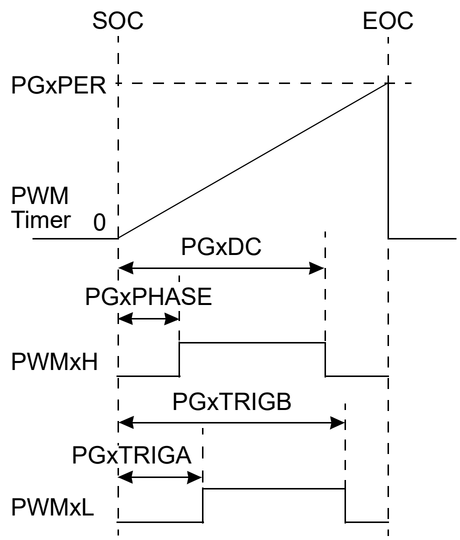
Figure 15-7. Multiphase PWM Example



15.5.2.2.3. Dual PWM Mode (Independent Edge PWM Mode, Dual Output)

The Dual PWM mode allows a single PWM Generator to produce two independent pulse widths on the PWMx output pins. This mode is equivalent to Independent Edge mode, except that it allows a second PWM pulse to be produced if the Independent Output mode is used. The Dual PWM modes are selected when MODSEL[2:0] (PGxCON[2:0]) = `010`. The PGxTRIGA and PGxTRIGB registers function as a second set of PGxPHASE and PGxDC registers to allow control of a second duty cycle generator. Figure 15-8 shows the relationship between the control SFRs and the output waveform.

Figure 15-8. Dual PWM Mode



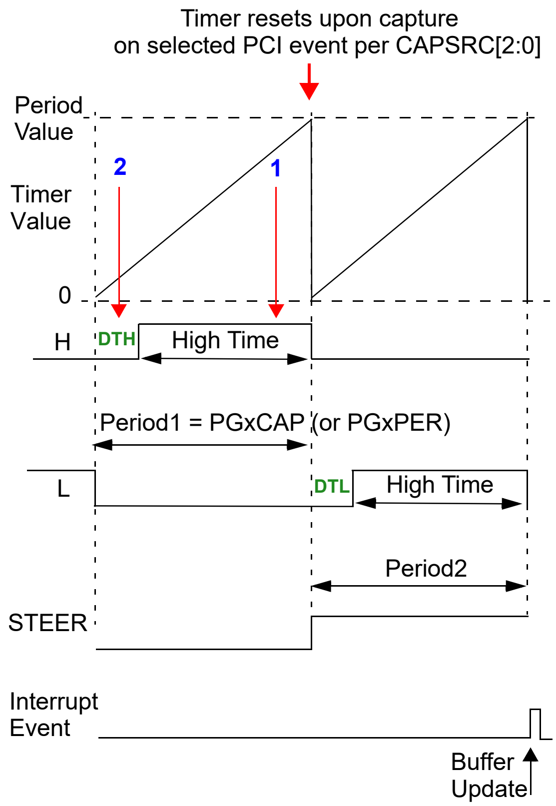
The PGxTRIGA and PGxTRIGB event output signals continue to operate normally in this mode and can still be used as phase offset triggers for other PWM Generators, ADC triggers, etc. If an independent trigger is needed, the PGxTRIGC register can be used. For additional information on ADC triggers, see ADC Triggers.

Since the PWM signals produced on the PWMx pins are generated from the same PWM Generator, they will be equally affected by any PWM Control Input (PCI) signals that are enabled. The PWMx pins will be driven to the states defined in the PGxIOCONx registers. Therefore, it is important that the two PWM outputs are used for related application functions if the PCI signals are to be used.

15.5.2.2.4. LLC Resonant Converter Mode

Support for LLC resonant converter topologies is provided within the PWM hardware. This mode should be operated in push-pull output mode. Therefore, there are two PWM timer cycles, one active PWMxH and the other for PWMxL. To maintain stability, the intent of this mode is to capture the on time of the first half and apply it to the second half while maintaining dead time.

Figure 15-9. LLC PWM Mode



Notes:
1. The capture event can occur before or after the existing period count such that PGxCAP ≤ PGxPER. Captures that occur in this range may cause both outputs to remain low in the cycles following this capture.
2. If no PCI event occurs, then EOC will be considered as a capture event.
3. Software capture is not applicable in LLC mode.

Control is done on a cycle-by-cycle basis using a PCI input to truncate the high time of the PWMxH and the period of the first half of the cycle (when PGxSTAT.STEER = `0`). When the PCI event occurs, the hardware captures the period value (period1) for use on the second half.

Period1 = PGxCAP +1

The contents of PGxPER are unaffected as PGxPER is used as an upper period limit in the case of no PCI truncation event. If no PCI event occurs, Period1 = PGxPER +1.

The hardware also calculates the effective high time of PWMxH. It is defined as:

HighTime = Period1 - DTH

The period and resulting high time of the second half (PGxSTAT.STEER = `1`) is then calculated:

Period2 = HighTime + DTL

The 2nd half of the cycle data then becomes Period2, DTL and HighTime.

Restrictions:

1. This Operating mode is not available in HR because the period boundary must coincide with the PWM clock cycle.
2. A PCI event within the DTH will result is both outputs being low.

3. The PCI override as selected in CAPSRC[2:0] is disabled. The system can rely on the other PCI events for override.
4. PGxDCA is not applicable in LLC mode.
5. LLC mode must be used with the Push-Pull Output mode.

50% High Time Option

An additional mechanism is available to provide additional flexibility in LLC mode, 50% high time. This 50% value is then modified with a scaler and offset to provide a desired result and stored in one of the TRIGy registers for use. See Capture to Trigger for additional details and usage.

15.5.2.2.5. Center-Aligned PWM Mode

Center-Aligned PWM mode signals avoid coincident rising or falling edges between PWM generators when the duty cycles are different, reducing excessive current ripple and filtering requirements in power inverter applications.

The PWM pulse maintains symmetry around the end of the first timer cycle and the beginning of the second cycle. If the duty cycle of the PWM signal is increased, then the position of the rising edge and the falling edge will change to maintain this symmetry. Center-Aligned PWM mode is selected when MODSEL[2:0] (PGxCON[2:0]) = `100`. The Center-Aligned PWM Operating mode uses two timer cycles to produce a single pulse. The characteristics of the PWM signal are defined by two SFRs:

- PGxDC: Determines the width of the PWM pulse from the center of the two timer cycles.
- PGxPER: Determines the end of the PWM timer count cycle.

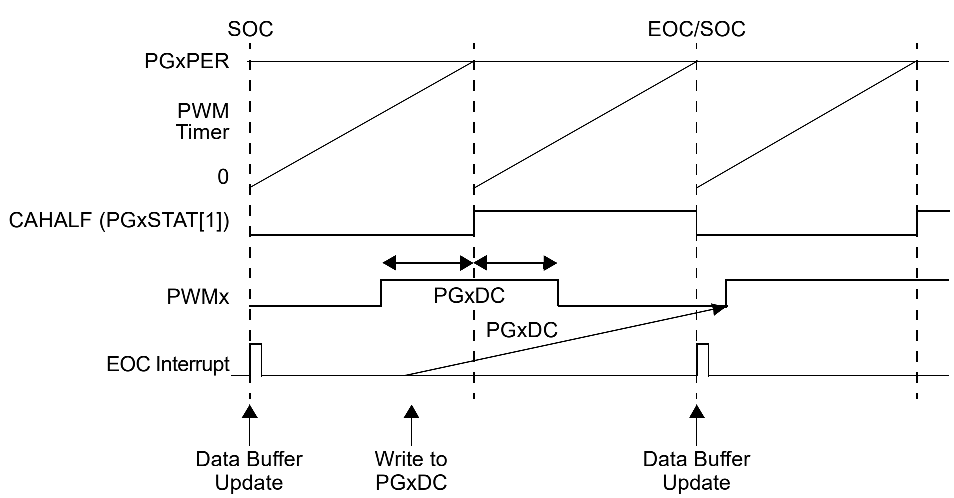
The falling edge occurs when the PWM Generator Timer equals PGxDC, and the rising edge occurs when the PG Timer equals PGxPER - PGxDC + 1. An offset of 1 is added to the rising edge calculation to produce symmetry between the two timer count cycles. A PGxDC value of '`1`', for example, will produce a pulse that is two cycles in duration. When PGxDC = 0, there are restrictions on the maximum PER value. In standard resolution, the maximum value is 0xFFFE. If the PER value is above this value, a short pulse at EOC may occur.

The timer cycle is tracked using the CAHALF status bit (PGxSTAT[1]) and is read as '`0`' in the first half of the cycle and '`1`' in the second half. Buffer updates to the duty cycle or period are allowed only at the beginning of the first timer cycle. The End-of-Cycle (EOC) interrupt is generated only after the completion of both period cycles. Figure 15-10 shows the relationship between the control SFRs and the output waveform. See Data Buffering for additional information on data buffering.

Center-aligned mode data restrictions and behavior:

- PHASE must be > 0; if not rising edge will not occur.
- PHASE must be < PER; if not falling edge will not.
- DC must be > 0; if not falling edge will not occur.
- DC must be < PER; if not falling edge will not occur, 0% DC.
- IF PER=DC, DC is 100%.

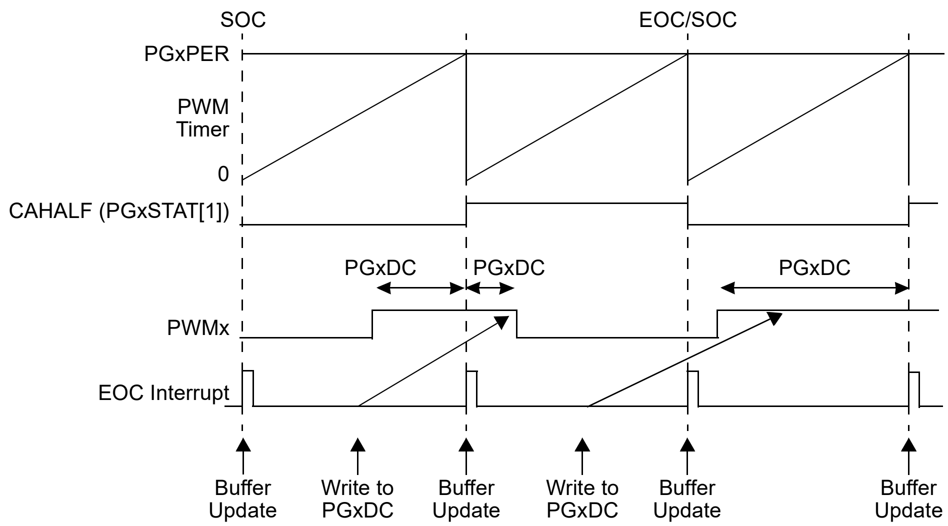
Figure 15-10. Center-Aligned PWM Mode



15.5.2.2.6. Double Update Center-Aligned PWM Mode

Double Update Center-Aligned PWM mode works identically to Center-Aligned PWM mode, except that two interrupts and two data buffer updates occur per PWM cycle. This mode is useful when the user wants to decrease the latency of a control loop response. Note that this will eliminate the symmetrical nature of the Center-Aligned PWM mode pulse, since the rising edge and falling edge of the pulse can be controlled independently. Double Update Center-Aligned PWM mode is selected when MODSEL[2:0] (PGxCON[2:0]) = `0b101`. Figure 15-11 shows the relationship between the control SFRs and the output waveform.

Figure 15-11. Double Update Center-Aligned Mode



15.5.2.2.7. Dual Edge Center-Aligned PWM Mode

Dual Edge Center-Aligned PWM mode works identically to Double Update Center-Aligned PWM mode but allows the rising edge time and the falling edge time to be controlled via separate Data registers. This mode gives the user the most flexibility in the adjustment of the center-aligned pulse, yet offers a lower frequency of interrupt events. Note that this will eliminate the symmetrical nature of the center-aligned PWM pulse unless PGxPHASE = PGxDC.

- PGxPHASE: Determines the rising edge time pulse from the center of the two timer cycles.
- PGxDC: Determines the falling edge time pulse from the center of the two timer cycles.

Note: PGxPHASE must be used for PHASE data; MPHASE is not supported.

Both Single and Double Data Buffer Update modes are available within the Dual Edge Center-Aligned PWM mode. Single Update mode is selected when MODSEL[2:0] = `110` and Double Update mode is selected when MODSEL[2:0] = `111`. In Single Update mode, the user may write a new PGxPHASE and PGxDC value at any time during the cycle to be used on the next center-aligned cycle. In Double Update mode, an interrupt event and a Data register update occur every timer cycle. This provides user software the opportunity to modify the PGxDC value for the falling edge event and PGxPHASE for the rising edge event. User software must check the state of the CAHALF bit (PGxSTAT[1]) to determine the appropriate register to update. If CAHALF = `0` (first half of the center-aligned cycle), the user software should write to the PGxDC register. If CAHALF = `1` (second half of the cycle), the user software should write to the PGxPHASE register. Figure 15-12 and Figure 15-13 show the relationship between the control SFRs and the output waveform.

Figure 15-12. Dual Edge Center-Aligned PWM Mode (MODSEL[2:0] = **110**)

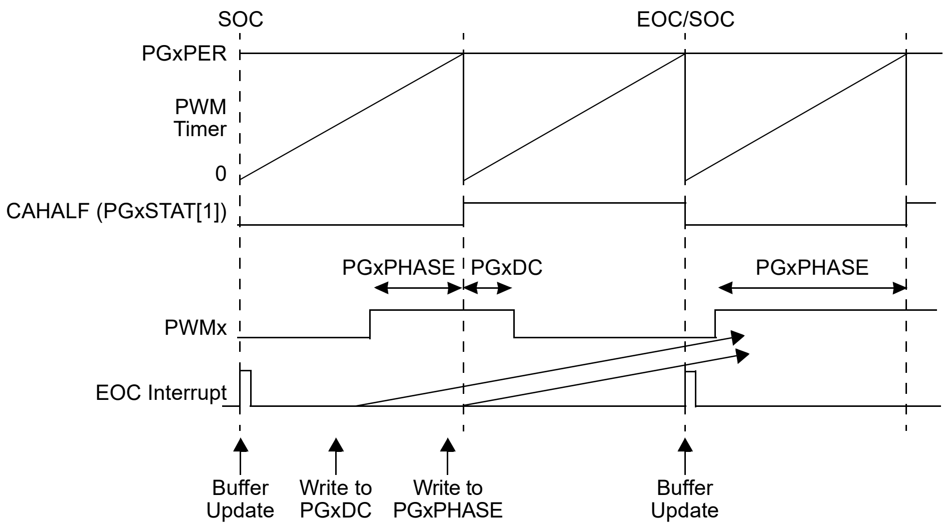
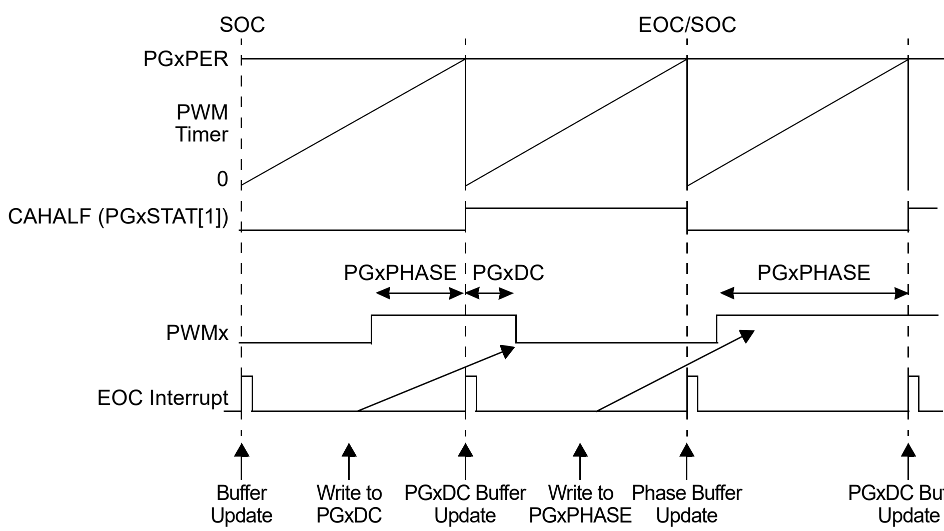


Figure 15-13. Double Update Dual Edge Center-Aligned PWM Mode (MODSEL[2:0] = **111**)



15.5.2.3. Output Modes

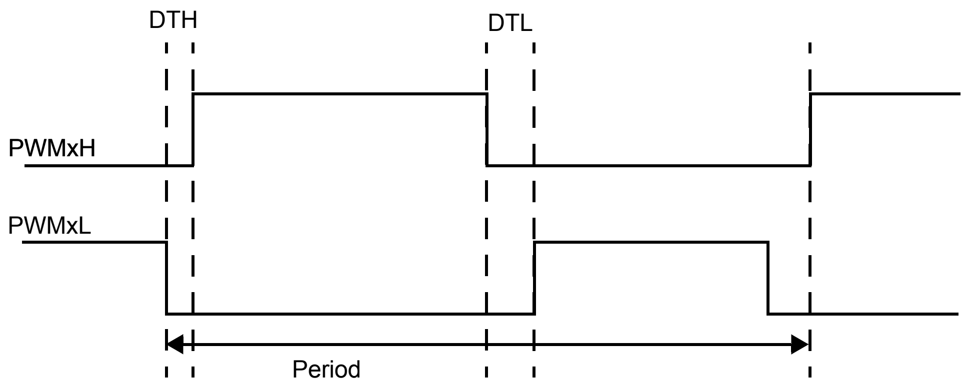
Each PWM Generator can be programmed to one of three output modes to control the behavior of the PWMxH and PWMxL pins. The output mode selection is independent of the PWM mode. The output modes are:

- Complementary Output mode (default)
- Independent Output mode
- Push-Pull Output mode

15.5.2.3.1. Complementary Output Mode

In Complementary Output mode, both the PWMxH and PWMxL signals are never active at the same time. A dead-time switching delay may be inserted between the two signals and is controlled by the PGxDT register. Complementary Output mode is selected when PMOD[1:0] (PGxIOCON1[5:4]) = `00`. In many applications, it may be beneficial to control the maximum on-time of the PWMxL signal (diode emulation mode) rather than remaining active for the duration of the programmed period. PGxTRIGF register provides a way to control the maximum on-time of the low side pulse. To control maximum on-time, PGxTRIGF must be within PGxDC < PGxTRIGF < PGxPER -3. By default, this register is set to 0xFFFFF0 to always drive the complementary leg (no control restrictions). For more information on dead time, see Dead Time.

Figure 15-14. PWMxH/PWMxL Rising and Falling Edges Due to Dead Time



Output Override Behavior in Complementary Output Mode

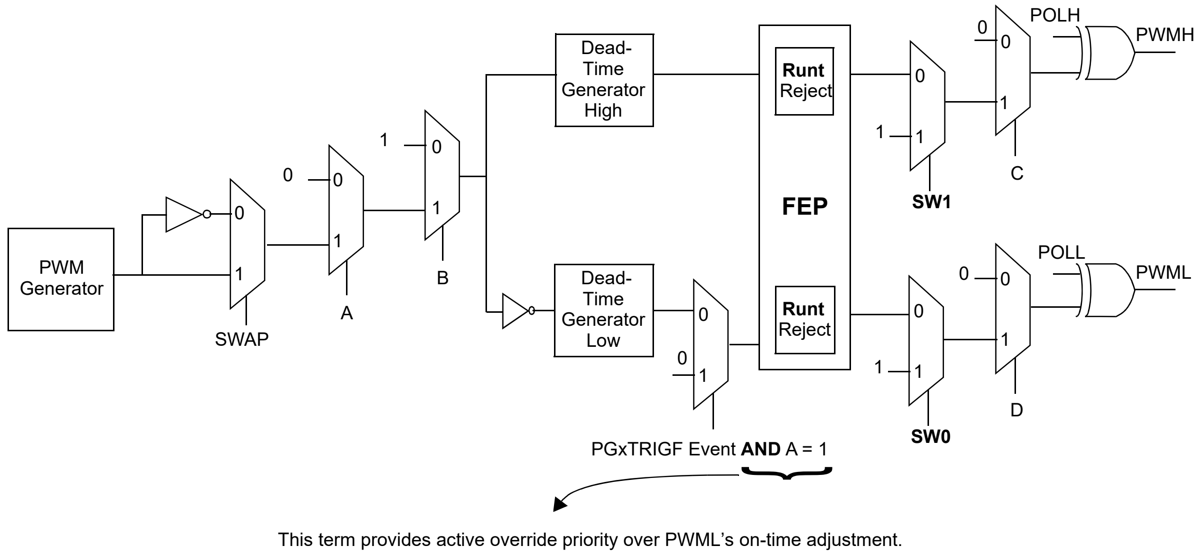
The PWMxH and PWMxL outputs may be controlled by external hardware signals or by software overrides. The output pins are restricted from being placed in a state that violates the complementary output relationship or in a state that violates dead-time insertion delays. An output pin may be driven inactive immediately as a result of a hardware event. However, a pin will not be driven active until the programmed dead-time delay has expired. The different hardware and software states are programmed using the following:

- PCI Fault event, FLTxDAT[1:0] (PGxIOCON2[7:6], PGxIOCON2[9:8])
- PCI current limit event, CLDAT[1:0] (PGxIOCON2[5:4])
- PCI feed-forward event, FFDAT[1:0] (PGxIOCON2[3:2])
- Debugger Halt, DBDAT[1:0] (PGxIOCON2[1:0])
- Software override, OVRENH (PGxIOCON2[21]) and OVRENL (PGxIOCON2[20])
- Swap of PWMxH and PWMxL pins, SWAP (PGxIOCON1[11])

Figure 15-15 shows the signal chain for override behavior in Complementary mode. The SWAP control is applied first and is, therefore, overridden by all other controls. Next, the request to drive a pin active is applied before dead time, so dead time is still applied to the output. Following the dead-time generator, in the PWMxL path only, is logic for an on-time adjustment. This arrangement allows the Inactive state to take precedence over SWAP and an active request. Finally, the polarity control is applied to the pin.

The PCI overrides operate on a priority scheme; see Output Control PCI Blocks for more information.

Figure 15-15. Override and SWAP Signal Flow, Complementary Mode



SW1 switches to tie-high when PGxIOCON1[FORCEON] = 1 **AND** OVRENH = 1 **AND** OVRDAT[1] = 1
SW0 switches to tie-high when PGxIOCON1[FORCEON] = 1 **AND** OVRENL = 1 **AND** OVRDAT[0] = 1

A = 0 => request to drive PWML active (xxxDAT[0] = 1)

B = 0 => request to drive PWMH active (xxxDAT[1] = 1)

C = 0 => request to drive PWMH inactive (xxxDAT[1] = 0)

D = 0 = > request to drive PWML inactive (xxxDAT[0] = 0)

Table 15-8 shows the rules for pin override conditions. The Active state is a '1' on the output pin, and the Inactive state is a '0'. An 'x' denotes a 'don't care' input; ~PWM indicates the complementary output of the PWM generator's output.

Table 15-8. Override Behavior in Complementary Output Mode

Source	FORCEON	SWAP	OVRENH	OVRENL	OVRDAT[1:0]	FFDAT[1:0]	CLDAT[1:0]	FLTxDAT[1:0]	DBGDAT[1:0]	PWMxH Signal	PWMxL Signal
Debug Override											
DEBUG	x	x	x	x	xx	xx	xx	xx	00	Inactive	Inactive
									01	Inactive	Active
									1x	Active	Inactive
Fault Override – Debug Override must be Inactive.											
PCI FLTx	x	x	x	x	xx	xx	xx	00	xx	Inactive	Inactive
								01		Inactive	Active
								1x		Active	Inactive
Software Override – Current Limit, Fault and Debug Overrides must be Inactive.											
Software Override	0	0	0	1	x0	xx	xx	xx	xx	PWM	Inactive
			1	0	0x					Inactive	~PWM AND ~ TRIGF event
		1	0	0	00					~PWM	PWM AND ~ TRIGF event
			0	1	x0					~PWM	Inactive
			1	0	0x					Inactive	PWM AND ~ TRIGF event
		x	0	1	x1					Inactive	Active
			1	0	1x					Active	Inactive
			1	1	00					Inactive	Inactive
			1	1	01					Inactive	Active
			1	1	1x					Active	Inactive
Current Limit Override – Fault and Debug Overrides must be Inactive.											
PCI CL	x	x	x	x	xx	xx	00	xx	xx	Inactive	Inactive
							01			Inactive	Active
							1x			Active	Inactive

Table 15-8. Override Behavior in Complementary Output Mode (continued)

Source	FORCEON	SWAP	OVRENH	OVRENL	OVRDAT[1:0]	FFDAT[1:0]	CLDAT[1:0]	FLTxDAT[1:0]	DBGDAT[1:0]	PWMxH Signal	PWMxL Signal
Software Override	1	0	0	1	x0	xx	xx	xx	xx	PWM	Inactive
			0	1	x1					PWM	Active
			1	0	0x					Inactive	~ PWM AND ~ TRIGF event
			1	0	1x					Active	PWM AND ~ TRIGF event
		1	0	0	00					~ PWM	PWM AND ~ TRIGF event
			0	1	x0					~ PWM	Inactive
			0	1	x1					~ PWM	Active
			1	0	0x					Inactive	PWM AND ~ TRIGF event
			1	0	1x					Active	PWM AND ~ TRIGF event
		x	1	1	00					Inactive	Inactive
			1	1	01					Inactive	Active
			1	1	10					Active	Inactive
			1	1	11					Active	Active
		• No force logic to prevent PWMxH and PWMxL from being 2’b11 (or 2’b00). • Content of OVRDAT[1:0] allowed to drive PWMxH/PWMxL based on OVRENH/OVRENL. • SWAP applies to two additional cases.									
Feed-Forward Override - Software, Current Limit, Fault and Debug Overrides must be Inactive.											
PCI FF	x	x	0	0	xx	00	xx	xx	xx	Inactive	Inactive
						01				Inactive	Active
						1x				Active	Inactive

Output Behavior At Start-Up in Complementary Mode

When the PWM is initialized and the ON bit is set, the outputs immediately go to a Complementary state. There is an output delay as the signals propagate through the PWM logic. This causes the start of the active duty cycle to appear delayed, with the PWMxL output transitioning to an Inactive state (pin high) for four master_pwm_clk cycles. Once the active duty cycle starts, the PWMx pins will behave as stated in Table 15-8.

Output Override Behavior in Complementary Output Mode with Priority Overrides and Force-On

The FORCEON bit is added to allow the PWM outputs to function only based on the register settings for software override. When FORCEON = `1`, active override on either PWMxH or PWMxL happens immediately (without the insertion of dead-time compensation).

Output Override Behavior in Complementary Output Mode with PWMxL's Max On-time Adjustment

In many topologies, PWMxL is used to control a synchronous rectifier switch which allows the power filter to freewheel during the off time. While in Discontinuous Conduction mode, however, it may not be desirable to keep the fully synchronous operation or completely turn off using FET body diodes for rectification. A better way to emulate diode operation is by actively driving the FET during the conducting phase and turning it off when the switch current crosses zero. This technique is common and widely used as an effective efficiency enhancement. With increasing CPU performance, low-side on-time estimation can be implemented in firmware to eliminate the need for external circuitry, thus reducing the additional analog peripherals. The PGxTRIGF register is, therefore, allocated to provide for this functionality. By properly programming this register, the PWML's on-time is turned off upon its trigger event.

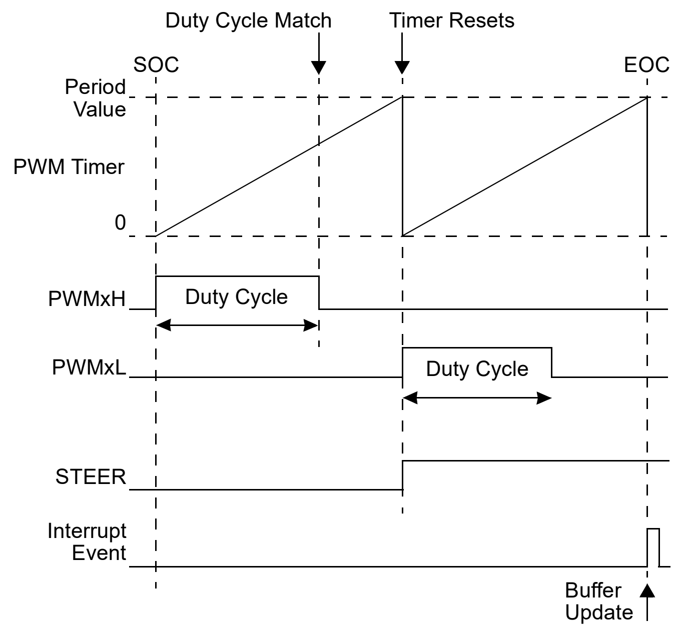
15.5.2.3.2. Independent Output Mode

In Independent Output mode, the output of the PWM Generator is connected to both the PWMxH and PWMxL pins. In most application scenarios, only the PWMxH or PWMxL pin would be enabled. The other pin remains available for GPIO or other peripheral functions. If the Dual PWM mode is selected, the PWM Generator will produce independent pulse widths on PWMxH and PWMxL, as described in Dual PWM Mode (Independent Edge PWM Mode, Dual Output). No dead-time switching delay is used in Independent Output mode. No restrictions exist for the states of the PWMxH and PWMxL pins; they can be controlled by external hardware signals or by software overrides. Independent Output mode is selected when PMOD[1:0] (PGxIOCON1[5:4]) = `01`.

15.5.2.3.3. Push-Pull Output Mode

The Push-Pull Output mode is similar to Independent Edge mode, however, the PWM cycle, as defined by the MODSEL[2:0] bits, is repeated twice each time a SOC trigger is received. The EOC trigger event and updates from Data registers are held off until the end of the second PWM cycle. Figure 15-16 shows the 2nd cycle that is invoked when using Push-Pull Output mode.

Figure 15-16. Push-Pull PWM



Operating the PWM in Push-Pull mode will double the period for a complete cycle, as there are two timer matches per cycle. If PGxTRIGy timers are used for event timing, the STEER signal can be used to gate application timing.

Push-Pull PWM mode is typically used in transformer coupled circuits to ensure that no net DC currents flow through the transformer. Push-Pull mode ensures that the same duty cycle PWM pulse is applied to the transformer windings in alternate directions. The phase of the push-pull count period can be determined by reading the STEER status bit (PGxSTAT[2]). If STEER = `0`, the PWM Generator is generating the first PWM pulse. If STEER = `1`, the PWM Generator is generating the second PWM pulse.

Since dead time is not available in Push-Pull mode, delays can be emulated in the Push-Pull Output mode by introducing a small phase offset with the PGxPHASE register. Similarly, the maximum duty cycle may be limited in software to avoid a pulse that ends too close to the start of the next PWM cycle.

Push-Pull Operation with Center-Aligned Modes

When the PWM Generator is operated in one of the two Center-Aligned modes, and the Push-Pull mode is selected, a complete PWM cycle will comprise four time base cycles.

Figure 15-17 shows the operation of the module with Push-Pull Output mode and Center-Aligned PWM mode. This combination of modes limits PWM buffer updates and interrupt events to every fourth time base cycle. Therefore, the same pulse is produced on the PWMxH and PWMxL pins before any changes to the duty cycle are allowed. Similar interrupt behavior also occurs when Dual Edge Center-Aligned mode (one update per cycle) is selected (MODSEL[2:0] = `110`).

Figure 15-17. Push-Pull PWM: Center-Aligned Mode, Dual Edge Center-Aligned Mode with One Update per Cycle (MODSEL[2:0] = **110**)

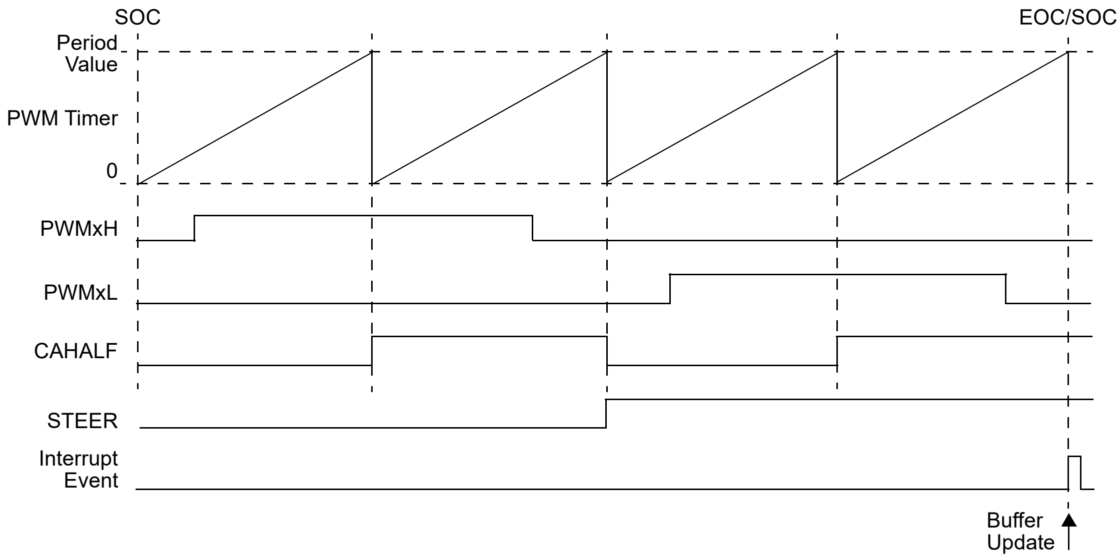
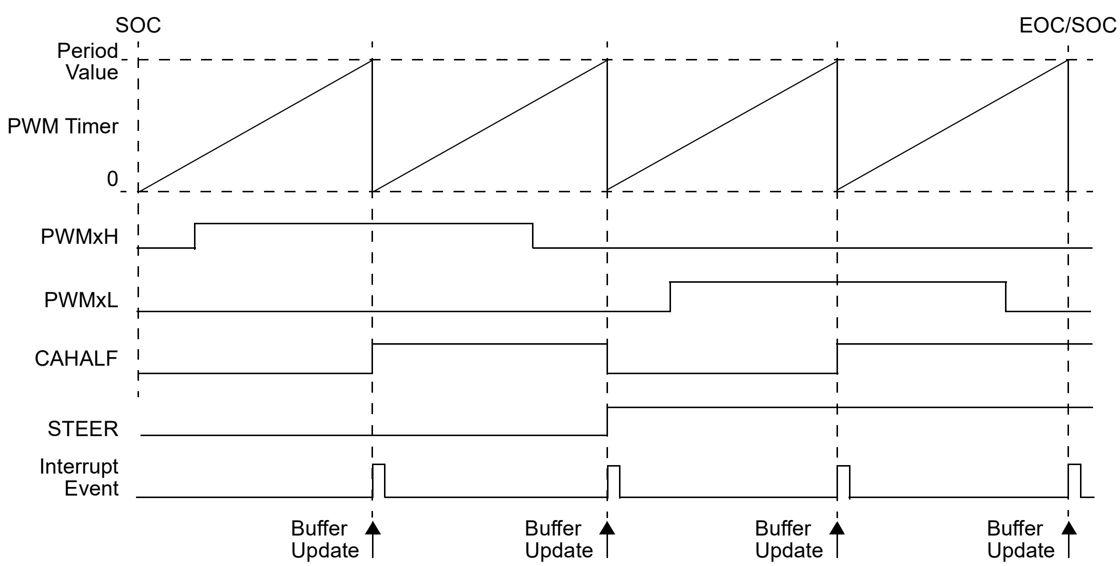


Figure 15-18 shows the operation of the module with Push-Pull Output mode and Dual Edge Center-Aligned PWM mode (two updates per cycle, MODSEL[2:0] = `111`) or Double Update Center-Aligned mode. This combination of modes allows a buffer update and interrupt event on every time base cycle. This operating configuration does not attempt to maintain symmetrical pulses on the PWMxH and PWMxL outputs, which is a requirement for many push-pull applications. User software can change the edge times of the center-aligned pulses after every edge event, which minimizes control loop latency.

Figure 15-18. Push-Pull PWM: Double Update Center-Aligned Mode, Dual Edge Center-Aligned Mode with Four Updates per Cycle (MODSEL[2:0] = **111**)



15.5.2.3.4. Output Override in Push-Pull and Independent Modes

When operating in Push-Pull or Independent Output modes, there is no logic that enforces a complementary relationship between the PWMxH and PWMxL signals. It is possible to drive both pins to an Active state with a software or hardware (PCI) override. This Output state may or may not be desirable, depending on the external circuit that is controlled by the PWM Generator. Therefore, care must be taken when selecting the pin override values. Many push-pull applications require an equal pulse on both the PWMxH and PWMxL outputs to avoid a DC component. If the application is sensitive to this, perform software overrides after two complete timer cycles have taken place. Hardware PCI overrides should be configured to take effect after both timer cycles in the push-pull sequence have occurred. This can be accomplished by using the STEER signal, routed through the event logic to a pin, which can then be selected as an input to the PCI block.

Figure 15-19. Override and SWAP Signal Flow, Push-Pull Output Mode

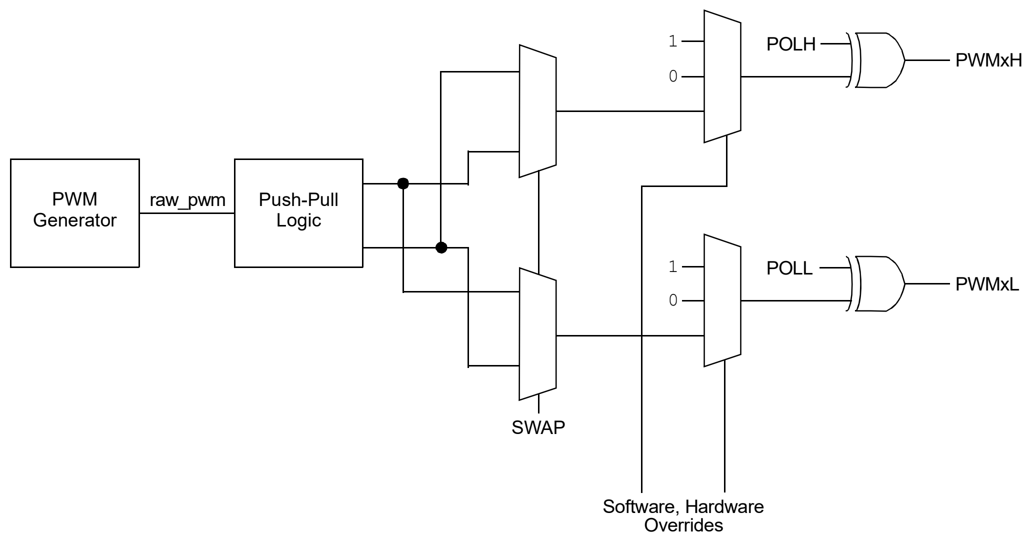


Table 15-9 shows the rules for pin override conditions. The Active state is a '1' on the output pin and the Inactive state is a '0'. An 'x' denotes a 'don't care' input; ~PWM indicates the complementary output of the PWM Generator's output.

Table 15-9. Override and SWAP Behavior in Push-Pull, Independent Output Modes

Source	SWAP	OVRENH	OVRENL	OVRDAT[1:0]	FFDAT[1:0]	CLDAT[1:0]	FLTDAT[1:0]	DBGDAT[1:0]	PWMxH Pin State	PWMxL Pin State
Debug Override										
DEBUG	x	x	x	xx	xx	xx	xx	00	Inactive	Inactive
								01	Inactive	Active
								10	Active	Inactive
								11	Active	Active
Fault Override – Debug Override Must be Inactive										
PCI FLT	x	x	x	xx	xx	xx	00	xx	Inactive	Inactive
							01		Inactive	Active
							10		Active	Inactive
							11		Active	Active
Current Limit Override – Fault and Debug Overrides Must be Inactive										

Table 15-9. Override and SWAP Behavior in Push-Pull, Independent Output Modes (continued)

Source	SWAP	OVRENH	OVRENL	OVRDAT[1:0]	FFDAT[1:0]	CLDAT[1:0]	FLTDAT[1:0]	DBGDAT[1:0]	PWMxH Pin State	PWMxL Pin State
PCI CL	x	x	x	xx	xx	00	xx	xx	Inactive	Inactive
						01			Inactive	Active
						10			Active	Inactive
						11			Active	Active
Feed-Forward Override – Software, Current Limit, Fault and Debug Overrides Must be Inactive										
PCI FF	x	0	0	xx	00	xx	xx	xx	Inactive	Inactive
					01				Inactive	Active
					10				Active	Inactive
					11				Active	Active
Software Override – Current Limit, Fault and Debug Overrides Must be Inactive										
Software Override	0	0	1	x0	xx	xx	xx	xx	PWMH	Inactive
		0	1	x1					PWMH	Active
		1	0	0x					Inactive	PWML
		1	0	1x					Active	PWML
	1	0	1	x0					PWML	Inactive
		0	1	x1					PWML	Active
		1	0	0x					Inactive	PWMH
		1	0	1x					Active	PWMH
	x	1	1	00					Inactive	Inactive
				01					Inactive	Active
				10					Active	Inactive
				11					Active	Active

15.5.2.4. Software Override

The software override feature can be used to take control of the PWM outputs and force certain conditions onto the pins. User software can override the output states of the pins by writing a '1' to the OVRENH and OVRENL control bits located in the PGxIOCON2 register. The state of the pins, when overridden, will be that of the value written to OVRDAT[1:0], unless it conflicts with restrictions imposed by the given Output mode. Most constraints are in Complementary mode and are discussed in Complementary Output Mode.

The OVRDAT[1:0] and OVRENH/L control bits are double-buffered for flexibility. The OSYNC[1:0] control bits in the PGxIOCON2 register specify when the user override values are applied to the PWM outputs. Manual software overrides can be applied at the following times:

- At the start of a new PWM cycle
- Immediately (or as soon as possible)
- As configured by the UPDMOD[2:0] control bits (see Data Buffering).
- As per data buffer update, synchronizing overrides on multiple PGs is done by enabling one of the Host/Client modes as defined by UPDMOD bits.

Like data updates, overrides are applied by a write of the UPDREQ. When UPDATE = 0, the user software may write new values to OVRENx or OVRDAT to set the UPDREQ bit. Setting the UPDREQ bit commits the new values to the PWM Generator. The UPDATE bit will be set, and writes are prohibited. When the process is complete, the UPDATE bit is cleared by hardware.

In general, SOC Update modes are recommended as they allow scheduling of override updates with an interrupt, which provides the maximum time to complete the next write. Otherwise, the UPDATE bit should be polled to ensure writing at a safe time.

Override update process:

1. Configure OSYNC and UPDMOD as needed for the application.
2. Configure PWM interrupt for SOC (default).
3. When an interrupt event occurs:
   a. Read the UPDATE bit and verify it is '0'.
   b. Write new values to OVRENx or OVRDAT.
   c. Set the UPDREQ bit to '1' to commit data to the PWM buffer.

Synchronizing Multiple PWM Generator override updates:

1. Configure host PG for SOC updates by writing OSYNC = '0' and MSTEN = '1'.
2. Configure client(s) PG for 'Client SOC' mode by writing OSYNC = 0b10 and UPDMOD = 0b0101.
3. Configure host PWM interrupt for SOC (default).
4. When an interrupt event occurs:
   a. Read host UPDATE bit and verify it is '0'.
   b. Host writes new data values to OVRENx or OVRDAT for all PWM generators (PGx).
   c. Set host UPDREQ bit to '1' to commit data to the PWM buffer.

On the next SOC, all PWM generators configured as clients will operate with new data values.

15.5.2.5. PWM Generator Triggers

Each PWM generator must receive a Start-of-Cycle (SOC) trigger to begin a PWM cycle. The trigger signal can be supplied by the PWM generator itself (self-triggered) or another external trigger source. The SOC trigger can be generated from three sources:

- An internal source operating from the same clock source selected by the SOCS[3:0] bits (PGxCON[19:16])
- An external source selected by the PWM Control Input (PCI) Sync block
- A software trigger request, write to TRSET (PGxSTAT[7])

Any of the PWM generators may act as a 'host' by providing the trigger for other PWM generators. Many trigger configurations may be achieved, including:

- Multiple PWM outputs with independent periods (no synchronization between PWM generators)
- Multiple PWM outputs with synchronized periods (synchronized operation)
- Multiple PWM outputs with offset phase relationships (triggered operation)

Synchronized operation is achieved by setting a (client) PWM Generator's SOCSx bits to that of another (host) PWM generator, with the host's PGTRGSEL[2:0] bits (PGxEVT1[18:16]) set to '000'. This selects the host's EOC to be used as the client's SOC trigger. When using PGxTRIGy to phase offset PWM generators from one another, a synchronization delay of up to five pwm_master_clk is present. If the TRIG value is '0', there will still be an offset. The TRIGy value may need to be compensated for the application.

Triggered operation is achieved in a similar way, but with the host's PGTRGSEL[2:0] bits (PGxEVT1[18:16]) set to select one of the PGxTRIGy counters (y = A through F). The value specified in the host's TRIGy register defines the client's trigger offset from that of the host's SOC.

The SOCS[3:0] control bits have two special selections. When SOCS[3:0] = 0000, the PWM generator is internally triggered. When SOCS[3:0] = 1111, no trigger source is selected. This selection is useful when the PWM generator will be triggered only by software, using the TRSET bit, or from a source connected to the PCI Sync block. In this mode, the next PWM cycle will not start until another trigger is received. The sources available to the PCI Sync block include external signals, such as comparator events and device I/O pins, etc. One of the important functions of the PCI Sync block is to synchronize external input signals into the clock domain of the PWM generator. See PWM

Control Input (PCI) Logic Blocks for more information on the PCI block. The PCI Sync can be OR'd into any of the other Start-of-Cycle inputs (SOCS[3:0]) as long as the block is enabled. The trigger output of another PWM generator can also be used as a SOC event. See Event Selection Block for configuration options.

15.5.2.5.1. Single Trigger Mode

The Single Trigger mode is the default trigger mode and is selected when TRGMOD[1:0] = 00. The Single Trigger mode is used when a PG timer should be started at the same time as or a time that is offset from another PG timer. This mode is also useful for creating a single PWM pulse, or creating a single delay based on an external event.

When operating in Single Trigger mode, the SOC trigger signal is used to start the PG timer cycle. When the SOC trigger is received, the timer will reset to '0', then start counting. When the timer matches the local PER register and enough timer cycles have occurred to complete the PWM cycle, the timer will stop counting.

The PG timer will not start until another SOC trigger is received. If a timer cycle is currently in progress, any incoming SOC trigger pulses will be ignored. The entire timer cycle must complete before another SOC trigger can restart the timer. The trigger logic looks for a rising edge on the trigger signal.

15.5.2.5.2. Retriggerable Mode

The retriggerable mode is selected when TRGMOD[1:0] = 01. This mode provides the same operation as Single Trigger mode. This mode is different from Single Trigger mode because a PWM cycle may be restarted before the end of a cycle that is already in progress. If the TRGCNT[2:0] bits have been written to produce a multiple cycle PWM event, the count will also be reset whenever a new trigger is received. If the update bit is set, the update happens at the start of the PWM cycle. However, if output mode is set to push-pull, the update only happens when STEER = 1. This mode can be especially useful when a PWM generator is synchronized to an external, off-chip source that operates from a different clock source. Due to the asynchronous nature of the clock source, the SOC trigger may occur slightly before or slightly after the end of a PWM cycle. If the SOC trigger arrives before the end of the PWM cycle, this trigger mode allows a new PWM cycle to be started.

15.5.2.5.3. Software Trigger

A new PWM cycle can be initiated in software by writing a '1' to the TRSET bit in the PGxSTAT register. An existing PWM cycle can be canceled by writing a '1' to the TRCLR bit in PGxSTAT. When the TRCLR bit is written, the PWM time base is reset and the PWM outputs associated with the PWM generator return to their idle states.

For TRGMOD[1:0] = 0b00, TRSET is ignored until TRIG is detected as cleared. For TRGMOD[1:0] = 0b01, TRSET is registered when set, thus causing the next set of operations (burst). Note that the content of TRGCNT[2:0] is unaffected by the state of TRSET when TRIG remains set.

15.5.2.5.4. Trigger Count (Burst Mode)

In some applications, it is desirable to have the PWM cycle repeat a certain number of times after the PWM generator is triggered. The TRGCNT[2:0] control bits in the PGxCON register select the number of times the PWM cycle will be repeated after a trigger event. If the PWM generator operates in the Single Trigger mode, then any incoming triggers will be ignored until all PWM cycles are completed. If the PWM generator operates in the Retriggerable mode, then an incoming trigger will start a new PWM cycle and reset the internal cycle count value.

15.5.2.6. PWM Control Input (PCI) Logic Blocks

The PWM Control Input (PCI) Logic blocks are flexible state machines that can be used for a wide variety of purposes. The PCI blocks condition input signals and provide output signals used to trigger, gate and override the PWM outputs. The PCI also allows PWM generators to interface with one another and external input signals. The PCI blocks can be used to implement output control and trigger algorithms in hardware instead of using software resources. There are four identical PCI blocks available for each PWM generator. The PCI blocks are:

- Faultx (where x = 1,2)
- Current Limit
- Feed-Forward
- Sync

The names of the PCI blocks do not limit their usage; they are given unique names to designate the priority levels. The Sync PCI block is intended for triggering, specifically from external events, including other PWM generators. The Fault, Current Limit and Feed-Forward PCI blocks are used to control the PWM output from external signals and other peripherals. The output state of the PWM pins can be independently configured to a predefined state for each PCI block, and it operates in a priority scheme if more than one PCI block requests control over the PWM outputs. Each PCI block has its own control register, PGxyPCI (with y = F, CL, FF or S), that contains the control bit associated with its operation. The PCI logic has three major components used to create logic functions:

- Inputs:
  - PCI source
  - PCI source qualifier, used to gate the PCI source signal
  - Terminator event, used to stop the ‘PCI_active’ output signal
  - Terminator qualifier event, used to gate the terminator event
- Acceptance logic
- Output and bypass function

Typical PCI source signals may include:

- Outputs to other PWM generators
- Combo triggers (see Combinatorial Triggers)
- Analog-to-Digital Converter (ADC)
- Analog comparator
- Input capture
- Configurable Logic Cell (CLC)
- External input (device pin)

The output of a PCI block (PCI_active signal) is made available to the PWM output logic and other PWM Generators. The status of the output signal of each PCI block is made available in the PGxSTAT register in both current and latched states. The PCI blocks can also generate interrupts (see Event Interrupts for more information). The block diagrams of the PCI function are shown in Figure 15-20 and Figure 15-21.

PCI Sync events will be synchronized into the PWM clock domain prior to use as a PWM generator trigger event. Because of the synchronization, there will be some uncertainty in the timing of the actual trigger event. PCI events that control the state of the output pins should be asynchronous, with some exceptions. This is because PCI events are commonly used to drive PWM pins inactive in the event of a current or voltage Fault. The objective is to place the PWM pins in a known state as quickly as possible.

When the PCI event drives a PWM pin to the Active state and the dead-time generator is in use, a PCI event signal synchronized to the PWM clock domain should be used. This will ensure proper edge detection by the dead-time logic. PWM pins should be driven inactive asynchronously by the PCI event signal when the dead-time generator is enabled for the reasons stated in this section.

Figure 15-20. PCI Function Block Diagram

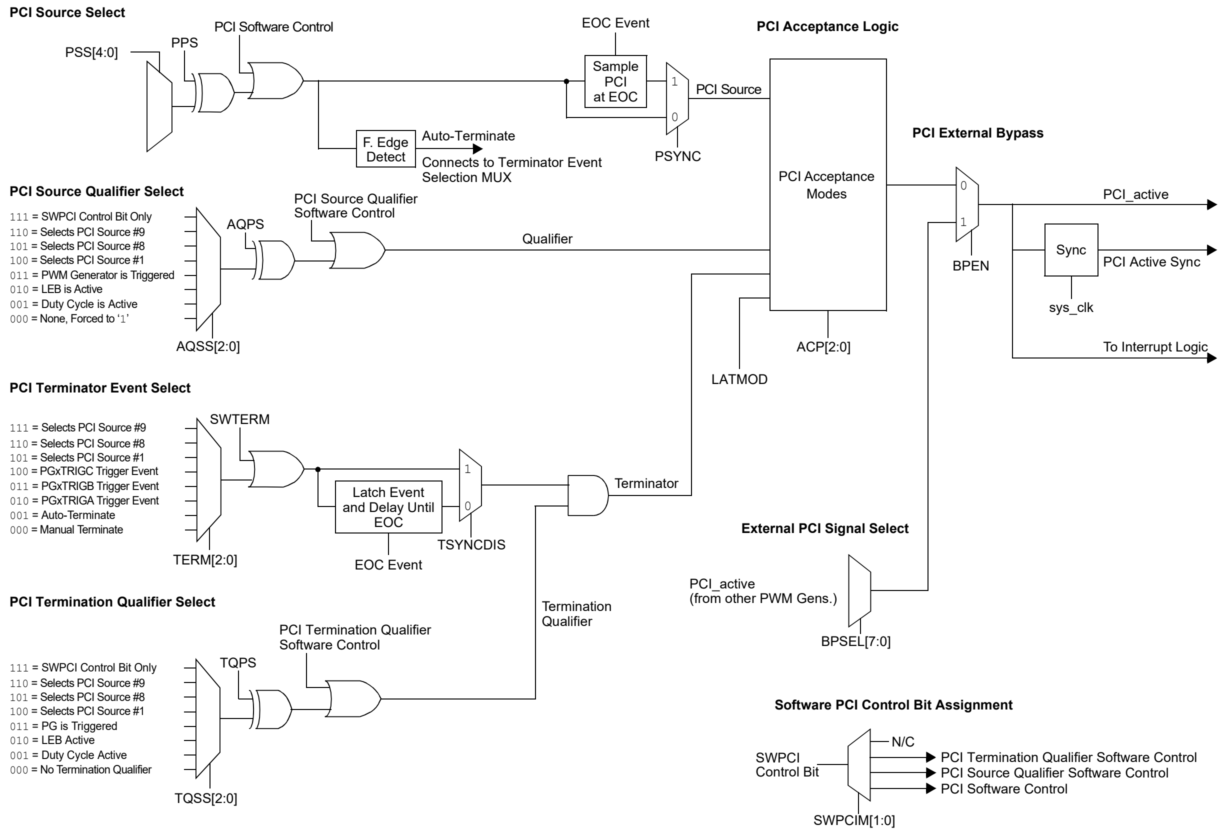
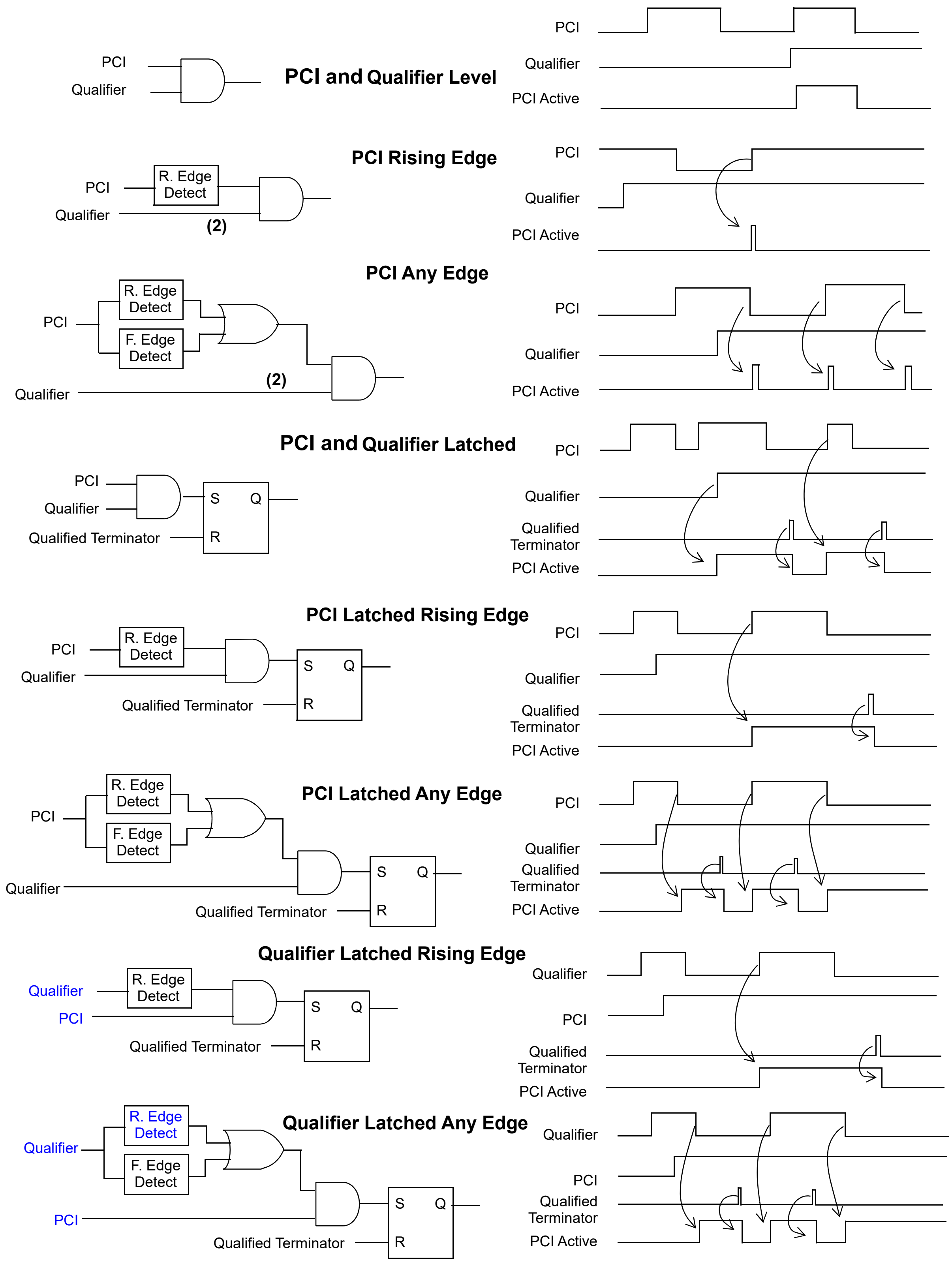


Figure 15-21. PCI Acceptance Modes



Notes:

1. The SR latch is set-dominant when LATMOD = 0 and Reset-dominant when LATMOD = 1.
2. The qualifier signal is synchronized to pwm_clk, and edge detection is done in the pwm_clk domain for this acceptance mode.

15.5.2.6.1. Sync PCI

The main purpose of the Sync PCI block is to trigger and synchronize external events to the PWM clock domain. The synchronization can induce up to a one PWM clock delay. The Sync block is the only PCI block that can initiate a Start-of-Cycle and is available as an input to the SOCS[3:0] (PGxCON[19:16]) MUX. The Sync and Feed-Forward are the only PCI blocks that can trigger alternate dead time (duty cycle adjustment); see Dead-Time Compensation for additional details.

15.5.2.6.2. Output Control PCI Blocks

The three output control type PCI blocks are provided to place the PWM outputs in a predetermined state. The blocks are prioritized in the following order, with further details shown in Complementary Output Mode and Output Override in Push-Pull and Independent Modes:

1. Fault
2. Current Limit
3. Feed-Forward

The Fault PCI block has the highest priority of the PCI blocks and will dictate the state of the outputs when the block is used. A Fault condition is generally considered catastrophic and is typically cleared with software.

The Current Limit PCI block was intended for use with current limit sensing circuitry, either for protection or use as a control loop. Leading-Edge Blanking (LEB) is typically used to ignore switching transients in current sensing applications. See Leading-Edge Blanking for details on LEB.

The Feed-Forward PCI block is intended for use as a control loop for power supply applications. If the sensing circuitry detects a rapid change in load conditions, the system can be configured to take immediate action without having to wait until the next PWM cycle to react.

Once a 'PCI active' signal is asserted, the value stored in the xDAT[1:0] bits (x = FLT, CL or FF) will be applied immediately to the pins. xDAT bits are located in the PGxIOCON2 register.

15.5.2.6.3. PCI Output Control Priority

If multiple PCI output control blocks are enabled to control the same set of PWM output pins, there may be conflicts due to the output state values programmed into the FLT1DAT[1:0], FLT2DAT[1:0], CLDAT[1:0], and FFDAT[1:0] control bits.

The following priority order is used to determine which source has control of the PWM output pins:

1. Debugger Halt Event
2. PCI Fault 1 Event
3. PCI Fault 2 Event
4. Software Override Event
5. PCI Current-Limit Event
6. PCI Feed-Forward Event
7. PWM Generator

The PWM generator has the lowest (default) priority. A halt of the CPU execution due to a debugger operation has the highest priority. Feed-forward events typically (but not always) drive the PWM pins to an active state, so the PCI Feed-Forward function is given a lower priority than a software override of the pins.

15.5.2.6.4. PCI Logic Description

The PCI block contains three major blocks to support a wide range of applications. First are the inputs, with logic for selecting and conditioning the input signals. Second is the PCI acceptance logic, which is the selectable logic functions applied to the inputs, and finally, there is the output logic, including bypass.

PCI Source

The PCI source input is the main input into the PCI block and has the following features:

- Input Selection Multiplexer
- Polarity Control
- Software Control (SW override)
- Edge Detect Circuit for Auto-Terminate
- End-of-Cycle (EOC) Synchronization

The PCI source is selected using the PSS[31:0] control bits (PGxyPCI2[31:0]). The polarity of the PCI input source can be selected using the PPS control bit. The chosen PCI input source may be optionally synchronized to the end of a PWM cycle using the PSYNC control bit. This synchronization is useful when a PCI signal is used to gate PWM pulses, as the PCI signal can be delayed to the next PWM boundary, ensuring that a partial pulse is not produced at the output. A falling edge detect circuit is present and can be used to automatically terminate the PCI active signal when selected by the terminator event selection multiplexer.

PCI Source Qualifier

The PCI source qualifier is a second input signal used to ‘qualify’ the PCI source. The PCI source qualifier is ANDed with the PCI source within the PCI acceptance logic. Inputs into the PCI source qualifier multiplexer include:

- Duty Cycle Active
- LEB Active
- PWM Generator Triggered
- PWMx Output Selected by PWMPCI[2:0] (PGxEVT1[23:21])
- External Input (Another Peripheral or Device Pin)

Like the PCI source input, the PCI source qualifier input has polarity and software control. The PCI source qualifier is used in all PCU acceptance logic types; however, if it is unneeded, AQSS[2:0] (PGxyPCI1[10:8]) can be set to ‘000’ to effectively disable the qualifier.

PCI Terminator Event and Qualifier

The PCI termination event sources are used only in the Latched modes of the PCI acceptance logic functions and are used to reset the latch. Inputs to the terminator event inputs include:

- SWTERM Bit
- Trigger Events (Trigger A, B, C)
- Auto-Termination (Falling Edge Detect on PCI Source)
- PWMx Output Selected by PWMPCI[2:0] (PGxEVT1[23:21])
- External Input (Another Peripheral or Device Pin)

The default option for the PCI terminator is SWTERM. The SWTERM bit must be written at least two PGx_clk cycles prior to the EOC. Otherwise, the PCI termination event will be delayed until the following EOC. The PCI trigger option (Trigger A, B or C) allows the PCI logic to be reset at a particular time in the PWM cycle. User software must select the appropriate PGxTRIG to be used as the PCI trigger source. When using Automatic Termination mode, it is recommended to select none as the

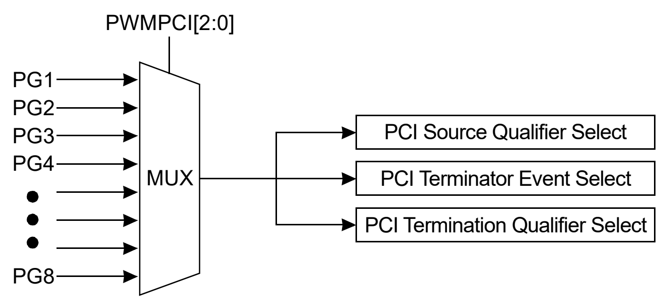
PCI termination qualifier. An EOC synchronization is provided by default and can be disabled by setting the TSYNCDIS bit (PGxyPCI1[15]).

The inputs and features of the termination qualifier are similar to the PCI source qualifier. The termination qualifier is used to create more advanced termination events.

Using a PWMx Output for PCI Function Input

The PWMPCI[2:0] control bits (PGxEVT1[23:21]) are used to select which one of the eight PWM outputs can be used by the PCI block. In some control loops, it is desirable to use the output of one PWM Generator to control another generator. The selected PWMx output is made available as a selection on the PCI source qualifier select, PCI terminator event select and PCI termination qualifier select MUXes, as shown in Figure 15-22.

Figure 15-22. PWM Source Selection for PCI



15.5.2.6.5. PCI Acceptance Logic

PCI acceptance logic is the selectable logic function that is applied to the PCI inputs. The six types of available logic functions shown in Figure 15-21 are:

- PCI Level mode: The PCI signal is passed directly through for use by the PWM Generator. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- PCI Rising Edge mode: The PCI signal is passed through a rising edge detection circuit that generates a pulse event. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- PCI Any Edge mode: The PCI signal is passed through both rising and falling edge detection circuits that generate a pulse event on either edge transition. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- PCI Latched mode: The PCI signal is used to set an SR latch. In this mode, a terminator signal and optional terminator qualifier are used to reset the latch. The entry into the PCI Active state is asynchronously latched and possibly gated by a qualifier signal. The exit from the PCI Active state is determined by a terminator signal and possibly a terminator qualifier signal. The exit from the PCI Active state can also be qualified by the absence of the PCI signal itself. (This is particularly important when the Latched mode is used for fault control applications.)
- PCI Latched Rising Edge mode: The PCI signal is passed through a rising edge detect circuit and optionally qualified to create a pulse event. This pulse event is used to set an SR latch. The SR latch can be reset in a similar fashion to the Latched mode. The Latched Edge Detect mode allows the PCI to become active on a PCI edge event after a qualifier signal is present.
- PCI Latched Any Edge mode: This mode is similar to the Latched Rising Edge mode, except that either a rising or falling edge is used to create the pulse event to set the latch.
- Qualifier Latched Rising Edge mode: Similar to the PCI Latched Rising Edge mode except that the PCI and qualifier signals are interchanged.
- Qualifier Latched Any Edge mode: Similar to the PCI Latched Any Edge mode except that the PCI and qualifier signals are interchanged.

Each mode of the PCI logic is intended to target a particular kind of power control function, although the functions can be applied to a wide variety of applications. The Level mode is useful when the PCI signals are used to affect the state of the PWM outputs asynchronously. For example, the Level mode could be used to allow an external blanking signal to force the PWM output pins to a specific state for a period of time.

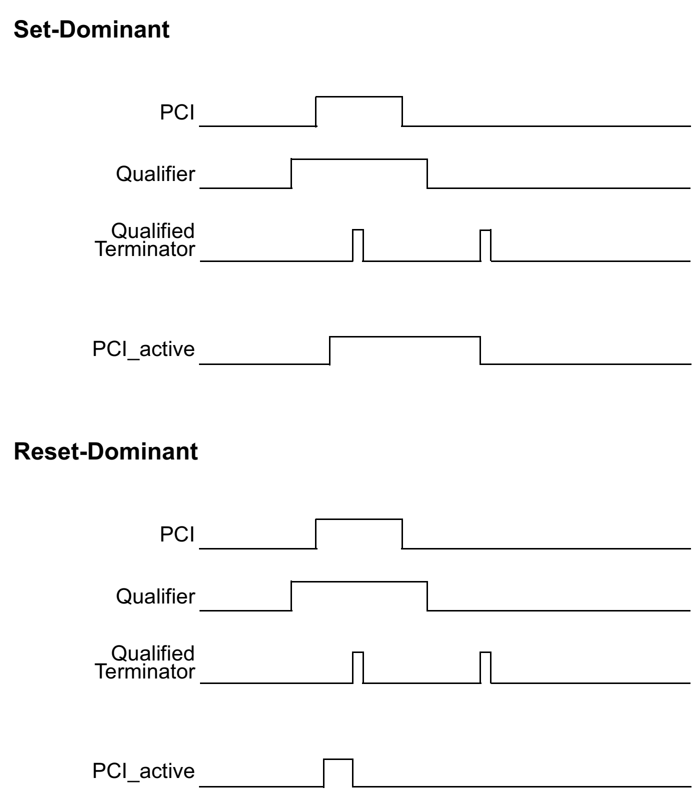
The Edge Event mode is useful when a PCI signal is used to synchronize a PWM Generator time base to an external source. When the PCI logic is used as a synchronization function, the rising edge event of the PCI signal is of primary interest. The edge event causes the PCI logic to generate an internal pulse, which triggers a PWM Generator.

The Latched mode is useful for fault and current-limiting applications. In these applications, it is important for the PCI logic to enter the Active state asynchronously when qualified. The PCI logic will remain active until a selected terminating event occurs. Usually, the terminating event is a software action (manual) or the end of a PWM cycle (automatic). The Latched Edge Detect mode is useful for some types of current control applications. The PCI output cannot become active until a transition of the PCI input occurs after a qualifying condition.

Latching Mode Control

By default, the SR latch used in Latched Acceptance modes is set-dominant. This prevents a Reset of the SR latch if the PCI signal is active when the termination event signal is asserted. The LATMOD control bit (PGxyPCI1[20]) can be used to create a Reset-dominant SR latch for certain PWM control functions. It is not recommended to use a Reset-dominant SR latch when the PCI logic is used to handle fault conditions, as this could allow the Active state of the PCI logic to be reset while the PCI input signal is still active. Examples of Latched modes are shown in Figure 15-23.

Figure 15-23. Latch Mode Control



15.5.2.6.6. PCI External Bypass

An option to use the PCI output of another PWM Generator is possible using the PCI external bypass feature. The PCI bypass function is useful when auxiliary, client or combinatorial PWM Generators require PCI functions based on the host PWM Generator's timing. The local PCI logic can be bypassed, using instead the output of the PCI block from another PWM Generator. Only the same type of PCI (Fault, Overcurrent, Sync or Feed-Forward) block can be utilized from another PWM Generator. When BPEN = 1, the PWM Generator specified by the BPSEL[7:0] bits (PGxyPCI1[7:0]) provides the PCI control. If more than one source is selected, they are OR'd together. The override states of the FLTDAT[1:0], CLDAT[1:0] and FFDAT[1:0] control bits are not affected when BPEN = 1. PWM Pin Override states are always determined by local control bits.

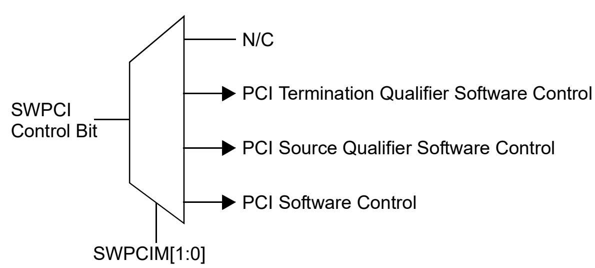
15.5.2.6.7. Software PCI Control

All PCI blocks have provisions for software control to force and clear events or for development debugging. There are three controls that can be used to manually control the PCI inputs:

- SWPCI Control Bit
- SWPCIM Demultiplexer (for SWPCI Control Bit)
- SWTERM to Generate a Termination Event

The SWPCI control bit (PGxyPCI1[23]) can have its programmed state of '0' or '1' routed to one of three destinations, specified by the SWPCIM[1:0] bits (PGxyPCI1[22:21]), as shown in Figure 15-24.

Figure 15-24. Software PCI Control Bit Assignment

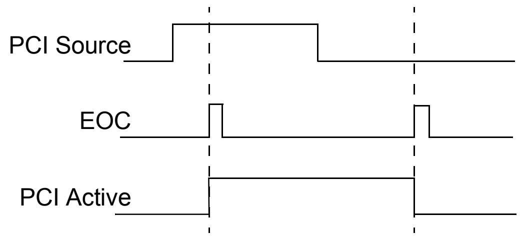


The SWTERM bit is tied directly to the terminator event input logic, and it can be used to manually terminate PCI events by writing a '1' to SWTERM and having the TERM[2:0] bits selection set to '000'. Additionally, the acceptance and terminator qualifier input multiplexers have an option to output a fixed state of '1' or, when used with their respective polarity control, a fixed state of '0'. These fixed states can be used for debugging or when the acceptance function is not needed.

PCI Source Synchronized to EOC, Level Mode

When the PCI acceptance logic is operated in Level mode and the PCI source is synchronized to the EOC event, there is no logic that retains the state of the prior PCI source signal. Therefore, the resultant PCI output is simply the PCI source signal synchronized to the EOC event. This configuration is useful for PWM chopping applications where the PCI source signal is used as a gating signal. The gating signal is automatically aligned to the PWM cycle boundaries, as shown in Figure 15-25.

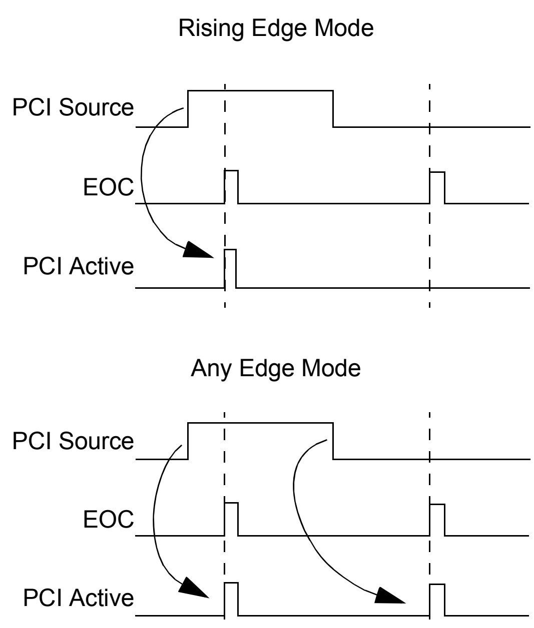
Figure 15-25. PCI Source EOC Sync, Level Acceptance Mode



PCI Source Synchronized to EOC, Edge Modes

When the PCI acceptance logic is operated in the Rising Edge or Any Edge modes and PSYNC = `1`, the PCI source is synchronized to the EOC event, as shown in Figure 15-26. If an edge event is detected, the pulse is delayed until the next EOC event. In the case that a PCI source signal becomes active and then inactive within a single PWM cycle, the PCI active signal will not assert.

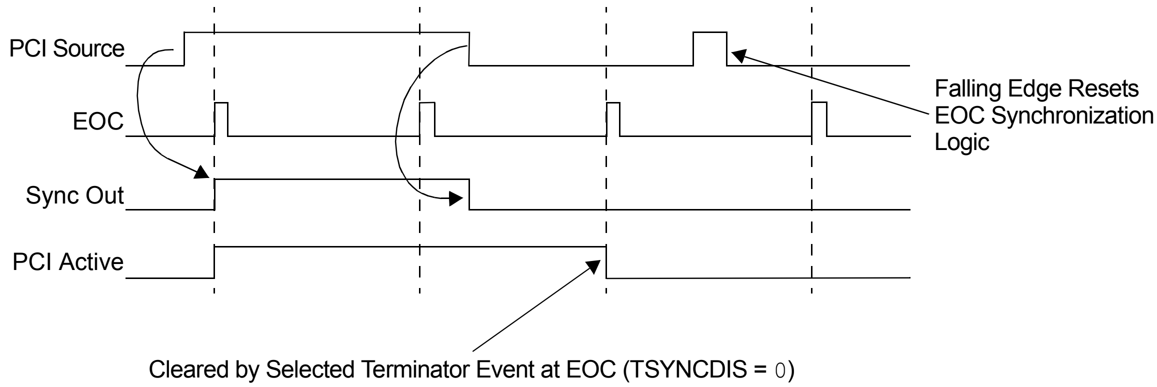
Figure 15-26. PCI Source EOC Sync, Edge Acceptance Modes



PCI Source Synchronized to EOC, Latched Modes

When the PCI acceptance logic is operated in the Latched mode and PSYNC = `1`, the PCI source is synchronized to the EOC event, as shown in Figure 15-27. The synchronization logic delays the rising edge of the PCI source signal until the next occurrence of the EOC signal. The output of the synchronization logic is deasserted on the falling edge of the PCI source signal. The output of the synchronization logic is then used to set the SR latch. A PCI input pulse that operates entirely within one EOC period will not assert the PCI active signal. This is because the falling edge of the PCI input signal resets the EOC synchronization logic before an event can be produced.

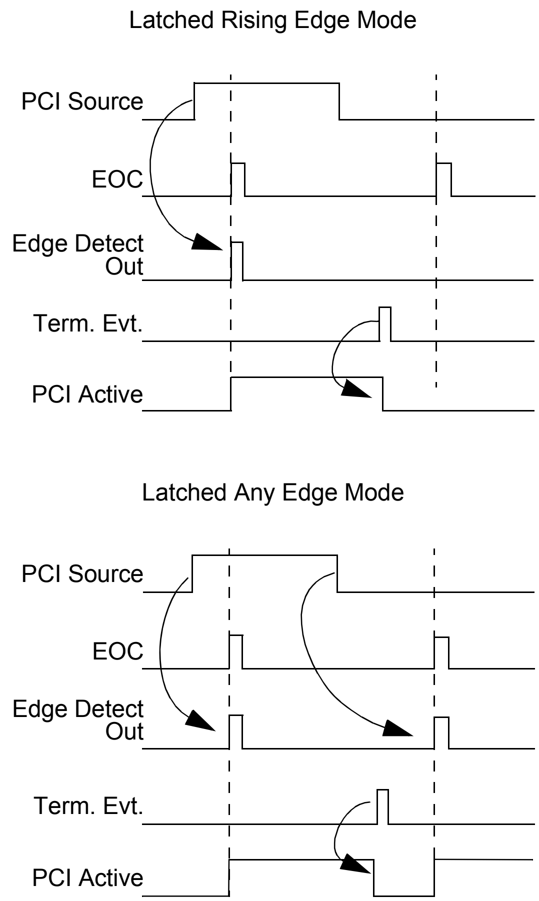
Figure 15-27. PCI Source EOC Sync, Latched Acceptance Mode



PCI Source Synchronized to EOC, Latched Edge Modes

When the PCI acceptance logic is operated in the Latched Edge modes and PSYNC = 1, the PCI source is synchronized to the EOC event, as shown in Figure 15-28. This configuration operates similar to the Rising Edge and Any Edge modes, except that the event output of the synchronization logic is latched. A PCI source input pulse that operates entirely within a PWM cycle will assert the PCI active signal.

Figure 15-28. PCI Source EOC Sync, Latched Edge Acceptance Modes

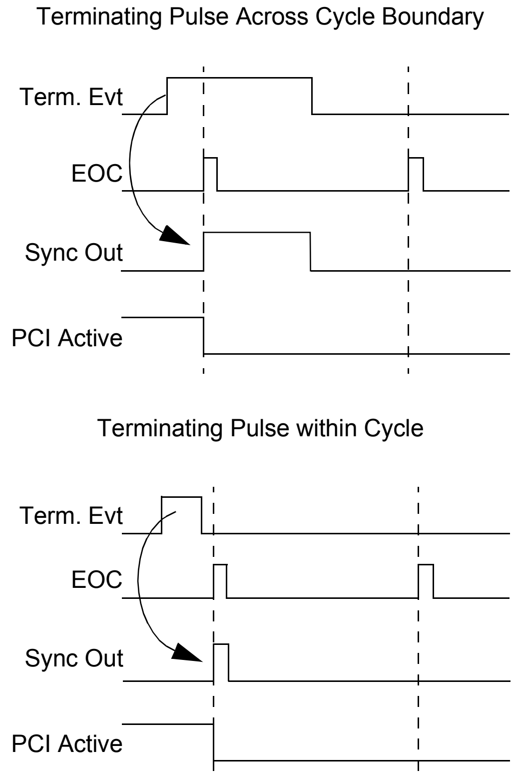


Note: These timing diagrams assume TSYNCDIS = `1`; therefore, the termination event takes effect immediately.

PCI Terminator Synchronized to EOC

By default, the PCI logic synchronizes a terminator event to the PWM EOC. This allows the PWM to resume cleanly at the start of a new cycle. The rising edge of the terminating signal is held off until an occurrence of the EOC event. The terminator signal is usually a pulse event used to reset the latched state of the PCI logic. If a short pulse is received prior to the occurrence of an EOC event, a Reset pulse is produced at the next EOC event. If the terminator signal is a longer pulse, the synchronized output is held active for as long as the terminator signal is present. This behavior can be used to force the PCI logic to a Reset state, if desired. Terminator event synchronization timing is shown in Figure 15-29.

Figure 15-29. PCI Terminator EOC Synchronization

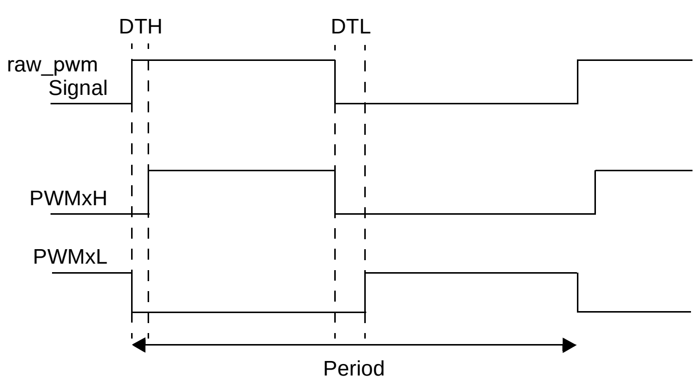


15.5.2.7. Dead Time

The dead-time feature is used to provide a time period where neither complementary outputs are active at the same time. Dead time is used to prevent both output driver devices (switches) in a bridge from conducting at the same time, causing excessive current flow. Since output switch turn-on and turn-off times are non-instantaneous, dead time is set to ensure that only one device is active. Dead time is implemented by holding off the assertion of the Active state. For the PWMxH and PWMxL outputs, this will delay the rising edge, as shown in Figure 15-30.

Dead-time duration is configured using the PGxDT register. The PGxDT register holds a pair of up to 15-bit dead-time values, DTH and DTL, that are applied independently to the PWMxH and PWMxL outputs, respectively. Dead time is typically only used in Complementary Output mode.

Figure 15-30. PWMxH/PWMxL Rising and Falling Edges Due to Dead Time

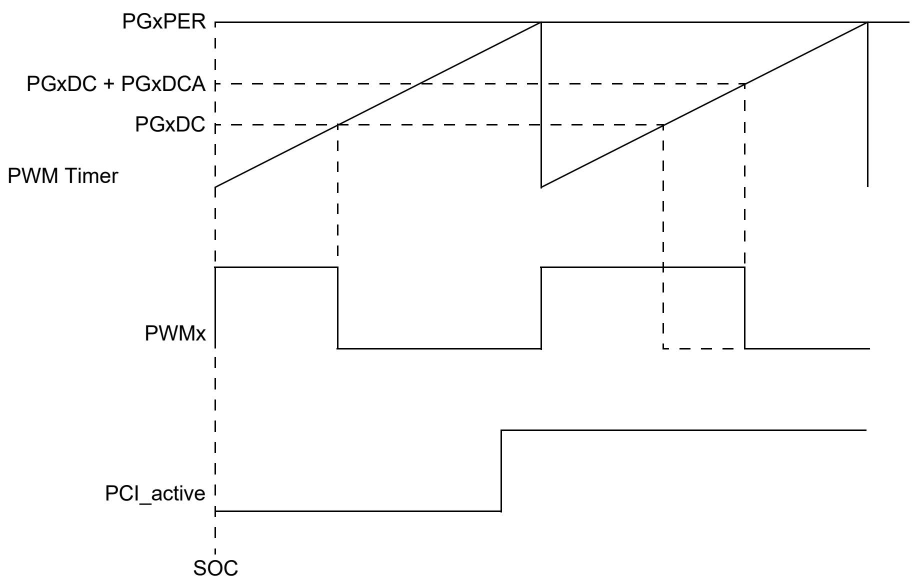


15.5.2.7.1. Dead-Time Compensation

The dead-time compensation feature allows the duty cycle to be selectively controlled by a PCI input. Dead-time compensation is enabled by writing a non-zero value to the PGxDCA (PWM Generator x Duty Cycle Adjustment) register and setting up PCI logic to control the compensation adjustment. When active, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle, as shown in Figure 15-31.

The DTCMPSEL control bit (PGxIOCON1[8]) selects the PCI Logic block to be used for dead-time compensation, which can either be the Feed-Forward or Sync PCI blocks. If the PGxDCA register is '0', the dead-time compensation function is disabled regardless of the DTCMPSEL value. The dead-time compensation input signal from the PCI logic is sampled at the end of a PWM cycle for use in the next PWM cycle. The modification of the duty cycle duration via the PGxDCA registers occurs during the end (trailing edge) of the duty cycle.

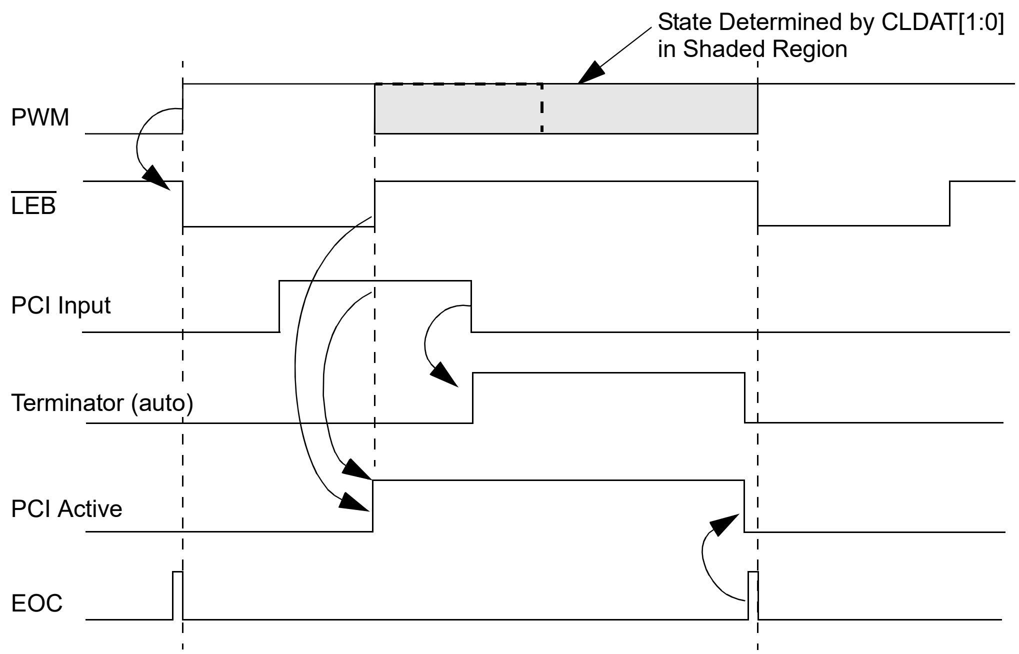
Figure 15-31. Adding PGxDCA Value to the PGxDC Register Value



15.5.2.8. Leading-Edge Blanking

The Leading-Edge Blanking (LEB) feature is used to mask transients that could otherwise cause an erroneous Fault condition. Leading-Edge Blanking can be implemented using any of the PCI blocks and basically ignores an input signal for a specified time following a PWM edge event.

Figure 15-32. Leading-Edge Blanking (LEB)



There are two methods to start the LEB timer: an edge detect or using one of the TRIGy timers. Both the rising and falling edges of both PWMxH and PWMxL signals can be selected to start the LEB timer. The LEB time duration is set by writing a value to the LEB bits (PGxLEB[19:4]). More than one edge (PHR, PHF, PLR, or PLF; refer to the PGxLEB register) may be used; however, if timing overlaps, the counter will be reset on each valid edge. In most applications, only one edge of the PWM signal needs to be selected to trigger the LEB timer. Additionally, the TRIGy timers (where y = A through D) can start the LEB timer to support blanking before an edge.

The LEB counter is commonly used to avoid a false trip when the PCI logic is used for current limiting. In this scenario, the LEB counter can be triggered on both edges of the PWM signal. The PCI logic is operated in Latched Acceptance mode with the LEB active signal used as a disqualifier for the PCI input signal. Figure 15-32 shows a PWM cycle where a PCI input goes active during the LEB timer. There is no PCI active event or output override until the LEB timer has expired.

15.5.2.8.1. Leading-Edge Blanking Counter Period Calculation

The LEB counter value is stored in the PGxLEB[19:4] bits and defines the period of the counter (T_{LEB}). The lower four bits are read-only and always read as '0', yielding a minimum LEB resolution of 16 PGx_clks. The minimum blanking period is five PGx_clks, which occurs when LEB[15:0] ≤ 4.

Equation 15-3. Leading-Edge Blanking Period

$$T_{LEB} = \frac{16 * (LEB[19:4] + 1)}{FPGx\_clk}$$

$$LEB[19:4] = \left(\frac{FPGx\_clk * T_{LEB}}{16}\right) - 1$$

Where:

T_{LEB} is specified in time units (ms, µs or ns)

15.5.2.8.2. Leading-Edge Blanking PCI Configuration

The LEB counter produces an LEB active signal that is supplied to the acceptance qualifier and/or termination qualifier selection multiplexers of the PCI blocks. This allows the LEB timer to be used as a gating signal for the selected PCI or PCI terminator signal. The polarity of the acceptance qualifier and termination qualifier signals can be inverted using the PCI control bits, so that the LEB active signal is changed to LEB inactive. It is recommended that the Latched PCI Acceptance mode be used when using LEB, so that the LEB active signal can only affect entry or exit to/from the PCI active state. Auto-termination can also be used to reset the system after the Fault condition has cleared.

Example LEB initialization sequence:

1. Select the type PCI to use for LEB (y = CL, FF, Fault).
2. Select the PCI input using the PSS[31:0] (PGxyPCI2[31:0]) bits. This is typically connected to a comparator to sense overcurrent.
3. Select LEB as the acceptance qualifier using the AQSS[2:0] (PGxyPCI1[10:8]) bits.
4. Invert the acceptance qualifier using the AQPS (PGxyPCI1[11]) bit.
5. Configure logic for Latched mode using the ACP[2:0] (PGxyPCI1[26:24]) bits.
6. Select auto-terminate using the TREM[2:0] (PGxyPCI1[14:12]) bits.

15.5.2.9. Event Selection Block

Each PWM Generator has a logic block for events, triggers and interrupts. These signals are then used by the Event Output block (Figure 15-33) or as the trigger source to start a PWM cycle (SOCS[3:0]). The Event Selection block has three main functions:

- ADC Trigger Configuration
- PWM Generator Trigger
- Interrupts

15.5.2.9.1. ADC Triggers

Each PWM Generator has the capability to trigger multiple ADCs, either internal or external to the device. ADC triggers are based on the TRIGy (where y = A through F) compare events for timing within the PWM cycle. The ADC triggers are also used as triggers for other peripherals and functions, such as the PTG, DAC and interrupts. The ADC triggers are also made available externally through the Event Output block (Figure 15-33) or internally in conjunction with the CPU interrupt controller.

Multiple TRIGx sources may be enabled to create the ADC trigger output, and when enabled, they are logically OR'd together. If the multiple TRIGx registers are enabled to produce ADC trigger events, they must be configured to allow unique trigger events to the ADC.

Each PWM Generator can generate two ADC triggers: ADC Trigger 1 and ADC Trigger 2. The two trigger outputs are useful for SMPS applications, where it is often desirable to measure two quantities in a single cycle. Each trigger is connected to a separate ADC, or possibly, a separate ADC trigger input. The ADC Triggers output has an additional offset and postscaler function to allow these functions:

- Postscaler, to reduce the frequency of ADC trigger events.
- Offset, a one-time offset may be applied to ADC trigger events. This allows postscaled trigger events to be interleaved with trigger events from other PWM Generators.

ADC Trigger outputs may be postscaled using the ADTRxPS[4:0] control bits to reduce the frequency of ADC conversions. In addition, the ADC Trigger outputs can be offset by a certain number of trigger events using the ADTRxOFS[4:0] control bits. Together, these two sets of control bits allow the user to establish an interleaved set of ADC triggers from multiple PWM Generators. In addition, ADC trigger events may be simply postscaled to reduce the frequency of ADC measurements. If the ADTRxPS[4:0] control bits are set to '00000', an ADC trigger event will be produced on every PWM cycle. When these control bits are set to a non-zero value, an ADC trigger will be produced during the PWM cycle after the ON bit is set and every N cycle thereafter. The ADTRxOFS[4:0] bits value establishes a one-time offset of 0 to 15 trigger events after the ON bit has been set. After this offset has been established, the trigger postscaler will begin to count the number of trigger events determined by the ADTRxPS[4:0] bits value. When interleaving ADC triggers from multiple PWM Generators, all PWM Generators should be programmed to have the same period to ensure consistent spacing between the trigger events.

15.5.2.9.2. PWM Generator Trigger Output

One of the PWM Generator internal events may be selected to drive the PWM Generator trigger output. The PWM Generator trigger output signal is selected using the PGTRGSEL[2:0] bits (PGxEVT1[18:16]) with the selection being either EOC or one of the three TRIGx compare events. Using one of the TRIGx events as an SOC trigger for another PWM Generator is useful for implementing a variable phase PWM. The phase relationship between two different PWM Generators can be controlled by the value written to the TRIGx registers.

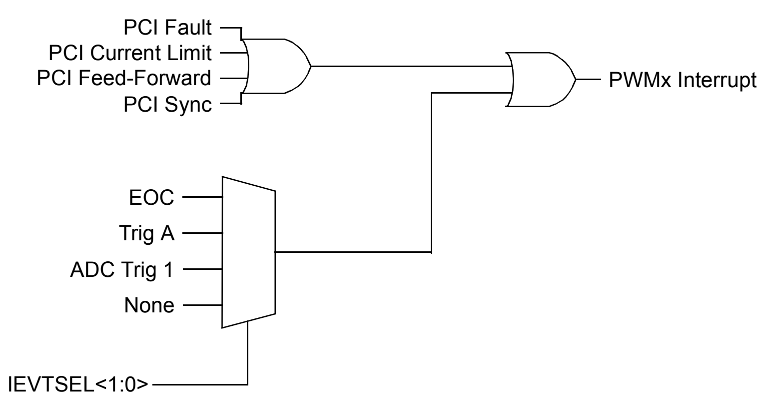
15.5.2.9.3. Event Interrupts

The PWM event that causes a CPU interrupt is programmable for flexibility. The IEVTSEL[1:0] control bits (PGxEVT1[25:24]) allow the user to select one of the following:

- EOC (default)
- TRIGA Compare Event
- ADC Trigger 1 Event
- None (disabled)

The Event Selection block also contains interrupt enables for each of the four PCI blocks. The SIEN, FFIEN, CLIEN and FLTIEN bits in the PGxEVT1 register are used to independently enable interrupts for their respective PCI block. When IEVTSEL[1:0] are set to disabled, the PCI interrupts can still be used independently.

Figure 15-33. Event Selection Block



DAC Triggers

The TRIGD and TRIGE compare events may be selected to drive the DAC Trigger output. Multiple sources may be enabled to create the DAC Trigger output. These sources, when enabled, are logically OR'd together.

If the multiple TRIGy registers are enabled to produce DAC trigger events, they must be programmed to values at least 10 counts apart to provide unique trigger events to the DAC. Trigger count values within 10 counts of each other will not produce unique trigger output pulses.

15.5.2.10. Data Buffering

The PWM module allows for certain SFR data values to be buffered and applied to the PWM output at later events. The following user registers and/or bits are buffered, allowing the user to modify data while the PWM Generator operates on the previous set of data values:

- MPER
- MDC
- MPHASE
- PGxPER
- PGxPHASE
- PGxDC
- PGxTRIGA
- PGxTRIGB
- PGxTRIGC
- PGxDT
- SWAP
- OVRDAT[1:0] (software output override values)
- OVRENL/H (software output override enables)

Data are transferred from the SFR registers to the internal PWM registers at the start of a PWM cycle. This can be every one, two, or four timer cycles, depending on the PWM Generator mode and the Output mode. It may be required that a register be updated immediately to produce an immediate change in a power converter operation. In other cases, it may be desirable to hold off the buffer update until some external event occurs where data coherency between multiple PWM Generators is of concern. The module supports the user in specifying when the contents of the SFRs associated with a PWM Generator are transferred into the “active” internal registers. Available options are:

- Immediately
- At the beginning of the next PWM cycle
- As part of a larger group

The UPMOD[2:0] control bits in the PGxCON register determine the operating mode for register updates. The UPDATE status bit in the PGxSTAT register allows visibility to when register updates are complete, and changes may be applied. When UPDATE = `0`, the user software may write new values to the PWM Data registers and set the UPDREQ bit when done. Setting the UPDREQ bit ‘commits’ the new values to the PWM Generator, and user software cannot modify PWM data values until the bit is cleared by hardware.

In general, SOC Update modes are recommended as they allow scheduling of override updates with an interrupt, which provides the maximum time to complete the next write. Otherwise, the UPDATE bit should be polled to write at a safe time.

Data update process:

1. Configure UPDMOD as needed for the application.
2. Configure PWM interrupt for SOC (default).
3. When an interrupt event occurs:
    a. Read the UPDATE bit and verify it is '0'.
    b. Write new data value(s).
    c. Set the UPDREQ bit to '1' to commit data to the PWM buffer.

On the next SOC, the PWM Generator will operate with new data values.

In order to avoid extra CPU cycles, the data updates can be configured to be automatically performed on a write to one of the PWM Data registers. The register is selected using the UPDTRG[1:0] bits in the PGxEVT1 control register. The default selection is that the UPDREQ bit must be manually set in software. A write to the PGxDC register can trigger an update, since many applications frequently change the duty cycle of the PWM. The PGxPHASE and PGxTRIGA registers may also be chosen as update triggers. These registers may be modified on a frequent basis in variable phase applications. The register that is selected as the update trigger must be the last one to be written if several PWM Data registers are to be updated. The PWM Data registers should not be modified once the UPDATE bit becomes set. User software must wait for the PWM hardware to clear the UPDATE bit before the Data registers can be modified again.

15.5.2.10.1. Synchronizing Multiple PWM Generator Buffer Updates

The MSTEN control bit (PGxCON[27]) allows the PWM Generator to control register updates in other PWM Generators. The UPDREQ control and UPDATE status bits can be effectively broadcast to other PWM Generators to allow coherent register updates among a set of PWM Generators that control a common function. When MSTEN is set and the user software (or the PWM Generator hardware) sets the UPDREQ control bit, this event will be broadcast to all other PWM Generators. If UPDMOD[2:0] = `01x` in a PWM Generator that receives the request, the receiving module will set its local UPDREQ bit. The local UPDATE status bit will then be cleared when the local registers have been updated. The user software may set a local UPDREQ bit manually.

Table 15-10. PWM Data Register Update Modes[(1)]

UPDMOD[2:0]	Mode	Description
`000`	SOC	Update Data registers at start of next PWM cycle if UPDATE = `1`. The UPDATE status bit will be cleared automatically after the update occurs.
`001`	Immediate	Update Data registers immediately, or as soon as possible, if UPDATE = `1`. The UPDATE status bit will be cleared automatically after the update occurs.
`010`	Client SOC	Update Data registers at start of next cycle if a Host update request is received. A Host update request will be transmitted if MSTEN = `1` and UPDATE = `1` for the requesting PWM Generator.
`011`	Client Immediate	Update Data registers immediately, or as soon as possible, when a Host update request is received. A host update request will be transmitted if MSTEN = `1` and UPDATE= `1` for the requesting PWM Generator.
`101-111`	—	Reserved

Note:

1. The UPDREQ bit must be set at least three sys_clk cycles, followed by three PGx_clk cycles, followed by another three sys_clk cycles, before the next PWM cycle boundary in order to take effect. Otherwise, the data update will be delayed until the following PWM cycle.

Synchronizing Multiple PWM Generator data updates:

1. Configure host PG for SOC updates and write MSTEN = '1'.
2. Configure client(s) PG for 'Client SOC' mode by writing UPDMOD = 0b0101.
3. Configure host PWM interrupt for SOC (default).
4. When interrupt event occurs:
   a. Read host UPDATE bit and verify it is '0'.
   b. Write new data values to all PWM generators (PGx).
   c. Set host UPDREQ bit to a '1' to commit data to PWM buffer.

On the next SOC, all PWM generators configured as clients will operate with new data values.

For the purpose of Data register updates, a PWM cycle length is variable. A PWM cycle may comprise one, two or four timer cycles, depending on the PWM Operating mode and the Output mode that is selected. The PWM Data registers may be updated on the next, second or fourth timer cycle when a SOC update has been requested. Table 15-11 summarizes the number of timer cycles between each SOC update vs. the PWM Generator Operating mode and the Output mode.

Table 15-11. Timer Cycles per Data Register Update

PWM Mode	Output Mode	Timer Cycles per PWM Cycle	Timer Cycles per Interrupt and Data Register Update
Independent Edge, Dual PWM or Variable Phase	Independent Output, Complementary	1	1
Independent Edge, Dual PWM or Variable Phase	Push-Pull	2	2
Center-Aligned	Independent Output, Complementary	2	2
Center-Aligned	Push-Pull	4	4
Double Update Center-Aligned or Dual Edge Center-Aligned	Independent Output, Complementary	2	1
Double Update Center-Aligned or Dual Edge Center-Aligned	Push-Pull	4	1

15.5.2.10.2. Immediate Updates

When using Immediate Update mode, there may be latency from the time of commanding a change to it becoming applied. This mode applies changes as soon as possible to prevent unexpected results.

Note: Avoid immediate updates near new or existing edge values to prevent data corruption. When using immediate updates, write timing should be tested in the software to avoid PWM edges.

An immediate update of period value updates to the PGxPER value becomes effective instantaneously. Care should be taken when the PWM period is shortened. If the PWM time base has already counted beyond the new (shorter) PWM period value, a long period will result as the counter must now count to 0xFFFF and then roll over. If immediate updates are required, the best practice is to capture the time base value prior to the period update so a safe minimum period value may be calculated and written.

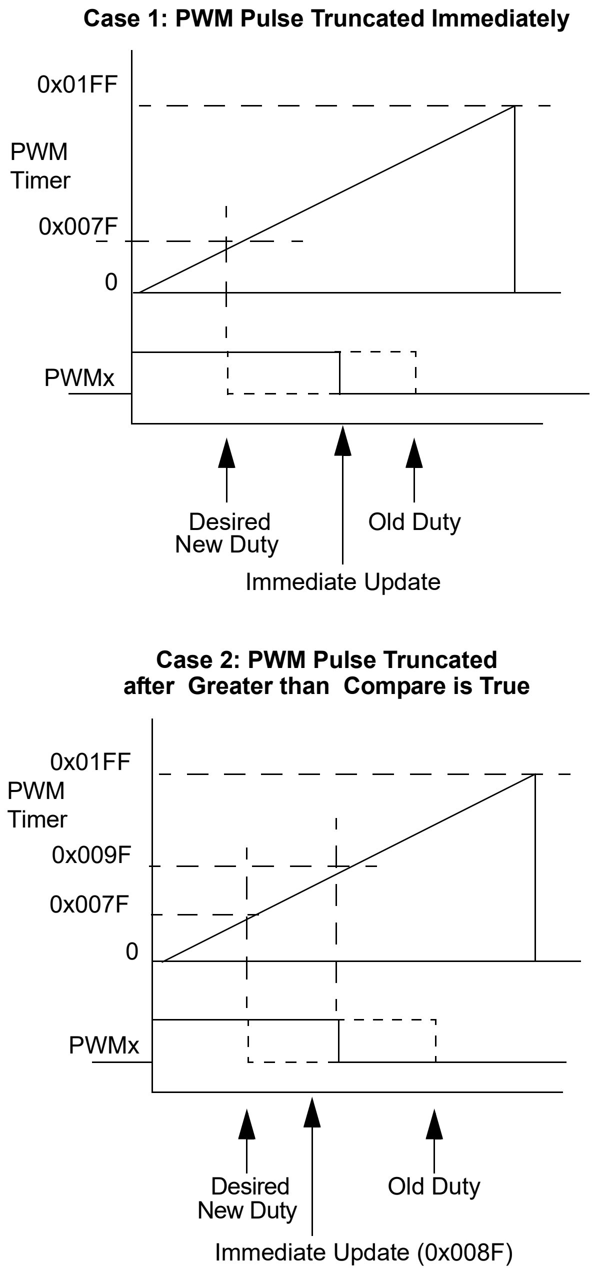
Immediate Updates to Duty Cycle, Phase Offset

The immediate update correction logic keeps track of whether a PWM pulse has already been created in the current PWM cycle, and it will not restart a second pulse within the same cycle. For example, if an immediate update changes the PGxPHASE value to a later time in the cycle, but a rising edge and falling edge have already been generated, then the new value of PGxPHASE will not take effect until the next cycle. An immediate update of PGxPHASE can only be used to delay a PWM pulse which has not occurred yet or cause the pulse to start sooner.

Figure 15-34 shows two examples of a correction during immediate updates. In this Update mode, the PWM module implements a compare function that compares the upper 12-bits of the data to the time base. This logic in hardware takes action as needed for the given conditions. The PWM period is relatively short in these examples, and a large duty cycle adjustment is made to emphasize how the correction works. In both examples, the duty cycle is decreased from 75% to 25% (0x7F) at approximately the mid-point of the PWM cycle. In both cases, the time base has already elapsed beyond the compare time for 25% duty.

In the first case, the immediate update write occurs at approximately 55% duty cycle. The PWM pulse is truncated immediately because the PWM time base is at least 0x8F. The programmed duty cycle is 0x7F, so the value of 0x80 provides a true greater than comparison when compared to 0x7F. In the second case, the immediate update write occurs just beyond the time of the newly programmed duty cycle. The PWM pulse is not truncated until the time base reaches a value of 0x0080 and the greater than comparison becomes true.

Figure 15-34. Immediate Update Correction Examples



Immediate Updates to PER

Using Immediate Update mode with PER data has some inherent hazards and should be avoided unless required by application. If used, user software must manage write time in relation to the PWM counter and known or planned event timing. The time base capture feature can be used for this purpose.

For example, it is possible to write a smaller period after both rising and falling edges have occurred. The PER compare will not occur in the cycle, and PWM counter will roll over at 0xFFFF. Due to no EOC event, the subsequent cycle will have no rising or falling edges. This can be avoided by reading time base capture and gating update if new PGxPER < PGxCAP.

Immediate Update to Dead Time

If a DT blanking is in progress and an immediate update to the DT occurs, the actual dead time after the update will be extended. This extension is due to the DT counter being reloaded before it has expired. Future dead-time delays after the immediate update will be the new time, as expected.

15.5.2.11. Time Base Capture

A time base capture feature is provided as the PWM timer itself is not directly readable. When the timer value is needed by the application, it may be captured and read via the PGxCAP register. The capture feature is gated with the CAPEN (PGxIOCON1[23]) bit. In a Default state of '0', it will only capture one event until the contents of PGxCAP are read, regardless of subsequent capture triggers. With CAPEN is set to '1', subsequent capture triggers will overwrite the PGxCAP register. There are two methods to trigger an event: either manually with software or with hardware on a PCI event. The CAPSRC[2:0] control bits (PGxIOCON1[22:20]) are used to select either a manual capture or one of the four PCI blocks as the trigger for a time base capture.

To manually capture the timer value, write a '1' to PGxCAP[0]. The CAP status bit (PGxSTAT[5]) will set to indicate the capture is complete, and then the user may read the PGxCAP register to determine the time base value at the time of the hardware event. A read operation of PGxCAP will clear the CAP status bit. It is recommended to read the CAP status bit to verify it is set before reading PGxCAP. This is to avoid a read of the PGxCAP register at the same time as the PWM hardware is writing it. An alternative method is to schedule reads with an interrupt to avoid concurrent access.

There will be up to three time base clock cycles of latency between the time of the actual event that caused the capture and the actual time base value that is captured. This delay is due to synchronization and sampling delays.

Time base capture example:

1. Read the CAP status bit and verify CAP is '0' (no pending capture).
2. Initiate capture event (SW or PCI).
3. Poll the CAP status bit and wait for it to set to indicate data are ready.

15.5.2.11.1. Capture to Trigger

To support the LLC Operating mode, a capture to trigger is available. When triggered, the hardware will capture the time base value, perform a computation and store the result in one of the PGxTRIGy registers. The feature is enabled with the CAPTREN (PGxIOCON1[19]) bit and the CAPTRSEL[1:0] (PGxIOCON1[17:16]) bits define the PGxTRIGy (where y = C, D, E or F) register to store the result. The feature works in conjunction with the time base capture and stores the time base in the PGxCAP register as defined by its control bits.

The trigger value to be stored in the Trigger register is 50% of the high time value, defined as:

PGxTRIGy = (PGxCAP - DTH)/2

Offset and postscaler features can be utilized through the CAPTROFS[4:0] and CAPTRPS[4:0] bitfields, found in the PGxEVT2 register. Capture to Trigger outputs may be postscaled using the CAPTRPS[4:0] control bits to reduce the frequency of PWM events by having PWM events trigger every N cycles. Additionally, the Capture to Trigger outputs can have a one-time offset for a certain number of trigger events using the CAPTROFS[4:0] control bits before the postscaler begins to count the number of trigger events determined by CAPTRPS[4:0].

The CAPTR status bit in PGxSTAT[13] indicates that a new trigger value has been successfully stored in the selected Trigger register. It is cleared when PGxCAP is read or CAPTRSEL[1:0] is written.

Table 15-12 summarizes the operation of the capture features.

Table 15-12. Capture Operation Summary

LLC Mode (MOD[2:0] = 011)	CAPEN	CAPTREN	Capture Operation
No	0	0	• Continues per CAPSRC[2:0] after PGxCAP read OR. • Continues immediately after PGxCAP read AND setting PGxCAP[0] high.
	0	1	• Same as immediately above PLUS • Stores trigger value per CAPTRSEL[1:0] with offset based on CAPTROFS[4:0].
	1	0	• Continues per CAPSRC[2:0] (no PGxCAP read needed).
	1	1	• Same as immediately above PLUS • Stores trigger value per CAPTRSEL[1:0] with offset based on CAPTROFS[4:0].
Yes	x	0	• Continues per CAPSRC[2:0] in cycle 1 when STEER = 0.
		1	• Same as immediately above PLUS • Stores trigger value per CAPTRSEL[1:0] with offset based on CAPTROFS[4:0].

Note: The trigger value is equal to exactly one half of the high time count value in LLC mode.

15.5.2.11.2. Operation in Debug Mode

When halting program flow using the debugger, the PWMx output pins can be left in a state that may be harmful to the hardware. To avoid this, logic is included to force the pins to a predetermined state, defined by the DBDAT[1:0] bits (PGxIOCON2[1:0]). The Pin states are still subject to the priority of overrides.

15.5.3. Common Features

15.5.3.1. Master Data Registers

The PWM module has a set of common Data registers that can be optionally assigned to multiple PWM Generators:

- MDC: Master Duty Cycle register
- MPER: Master Period register
- MPHASE: Master Phase register

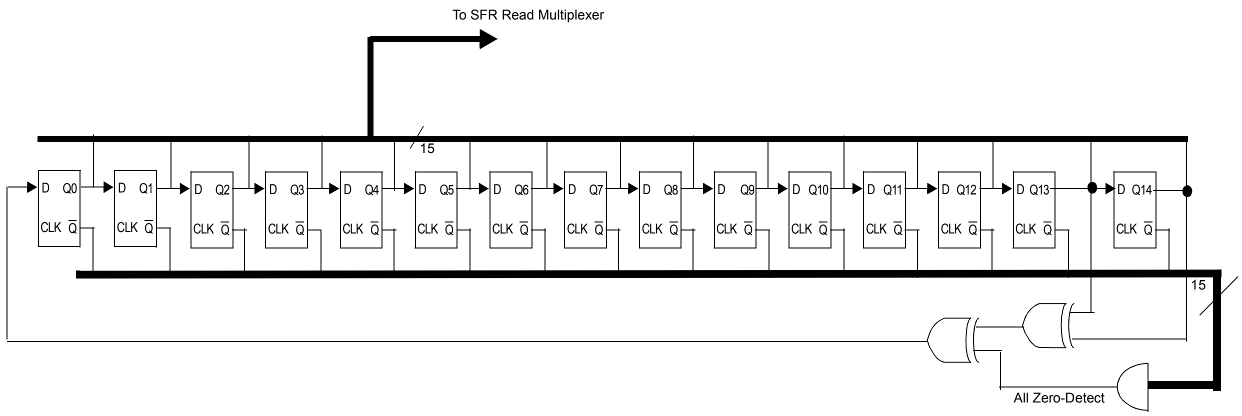
These Master registers allow user software to affect the operation of multiple PWM Generators by writing one Data register. The MDCSEL, MPERSEL and MPHSEL control bits in each PGxCON register determine whether the PWM Generator will use the local Data registers or the Master Data registers.

15.5.3.2. LFSR – Linear Feedback Shift Register

The Linear Feedback Shift Register (LFSR) is a pseudorandom number generator that provides 15-bit values, which can be used in applications to modify either the duty cycle and/or period by a small amount to dither the corresponding switching edges of the application circuit's power transistors. This dithering can be useful in reducing peak EMI (Electromagnetic Interference) emissions.

A read of the LFSR provides the last randomized value generated by the logic and triggers the generation of a new random value. Multiple random values can be obtained by sequentially reading the LFSR register. Additionally, the LFSREN bit can be used to enable the register to continue shifting.

Figure 15-35. LFSR Block Diagram



15.5.3.3. Shared Clocking

The PWM clocking system has two additional clock features to support a wide variety of applications. Each PWM generator's clock selection (CLKSEL[1:0]) can independently select the pwm_master_clk or a divided or scaled version of it.

15.5.3.3.1. Clock Divider

A common clock divider circuit is available for use by all PWM Generators and allows a PWM Generator to be operated at a low frequency. Four different divider ratios may be selected using the DIVSEL[1:0] control bits (PCLKCON[5:4]). The clock divider circuit remains in a Low-Power state if none of the PWM Generators have requested it.

15.5.3.3.2. Frequency Scaling

Frequency scaling provides the ability to drop clocks to effectively stretch the period or duty cycle and is useful for resonant power control applications that require a variable frequency control input. The clock input for the frequency scaling circuit is chosen using the MCLKSEL bits (PCLKCON[1:0]). The frequency scaling clock output is available to each PWM Generator and can be selected using the CLKSEL[1:0] control bits (PGxCON[4:3]).

The FSCL (Frequency Scale) and FSMINPER (Frequency Scaling Minimum Period) registers specify the amount of frequency scaling and are readable/writable at all times. The frequency scaling circuit performs modulo arithmetic where the FSCL value is constantly accumulated until the sum is larger than the FSMINPER register value. When the sum becomes larger than the FSMINPER register value, a clock pulse is produced and the accumulated value is reduced by the value in the FSMINPER register, as shown in Figure 15-36.

Note that the frequency scaling signal is applied only to the PWM time base counter and does not affect the operation of the dead time or the LEB counters. The frequency scaling circuit remains in a Low-Power state if not selected by any of the PWM Generators.

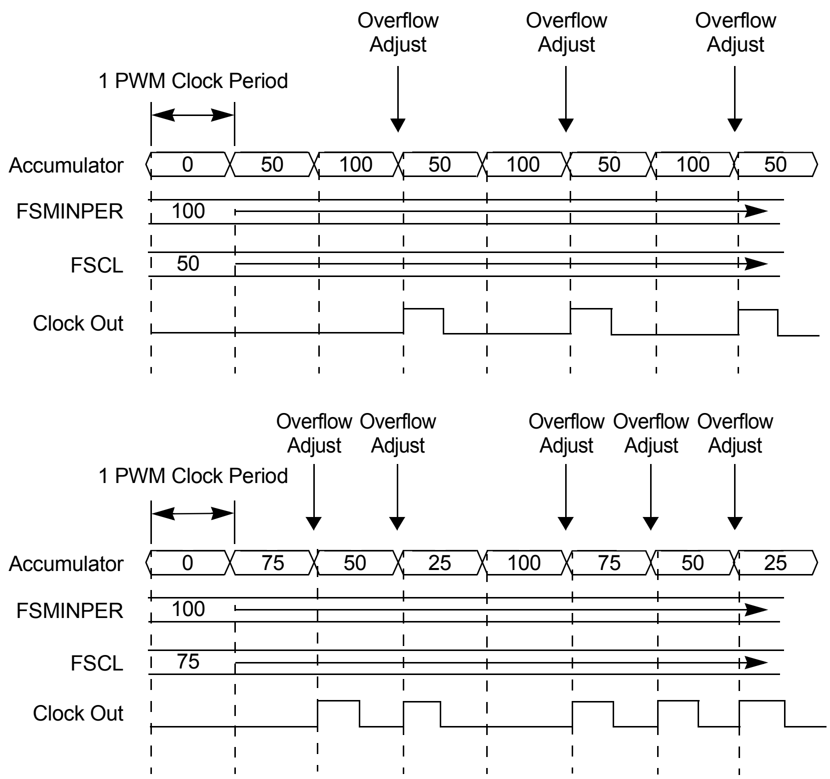
Example: Frequency Scaling Calculation

$F_{FSCL} = (FSCL/FSMINPER) * F_{PWM}$

Where:

FSCL ≤ FSMINPER

Figure 15-36. Frequency Scaling Examples



Note: When the frequency scaling circuit is selected as a PWM Generator clock source, the PWM Generator receives two clocks. One clock is the raw clock used to operate the frequency scaling circuit itself. This clock is also used to operate the dead-time counter and LEB counter within the PWM Generator. The second clock is the output of the frequency scaling circuit. This clock is used to operate the PWM time base counter.

15.5.3.4. Combinatorial Logic Output

The combinatorial logic output feature can be used to generate control signals for synchronous rectification or other applications. One or more PWM Generators can be used to output a logic function with programmable input selections and logic functions. When assigned to a PWM output, the combinatorial logic function replaces the PWM signal that would normally be connected to that pin. The controls include:

- Input sources (PWMSxy)
- Input polarity (SxyPOL)
- Logic AND, OR and XOR functions (PWMLFy)
- Output destination (PWMLFyD)
  Note: An ‘x’ in a bit name denotes Input Source ‘1’ or ‘2’. A ‘y’ denotes a function instance (A-F).

An example of a device with six LOGCONy registers and combinatorial logic output functions, A-F, is shown in Table 15-13.

Table 15-13. Combinatorial Logic Instance Mapping

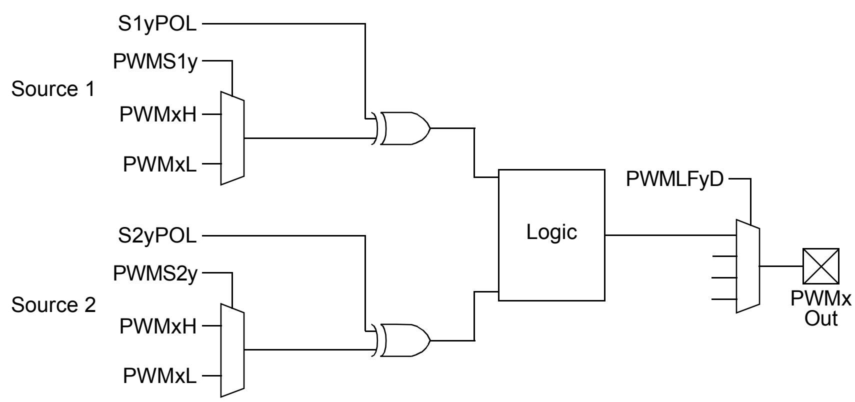
Register	Combinatorial Logic Instance	Available Output Pin Selection
LOGCONA	A	PWM2H-PWM4H
LOGCONB	B	PWM2L-PWM4L
LOGCONC	C	PWM2H-PWM4H

Table 15-13. Combinatorial Logic Instance Mapping (continued)

Register	Combinatorial Logic Instance	Available Output Pin Selection
LOGCOND	D	PWM2L-PWM4L
LOGCONE	E	PWM2H-PWM4H
LOGCONF	F	PWM2L-PWM4L

Figure 15-37 shows the combinatorial logic function block diagram.

Figure 15-37. Combinatorial Logic Function Block Diagram



Notes:

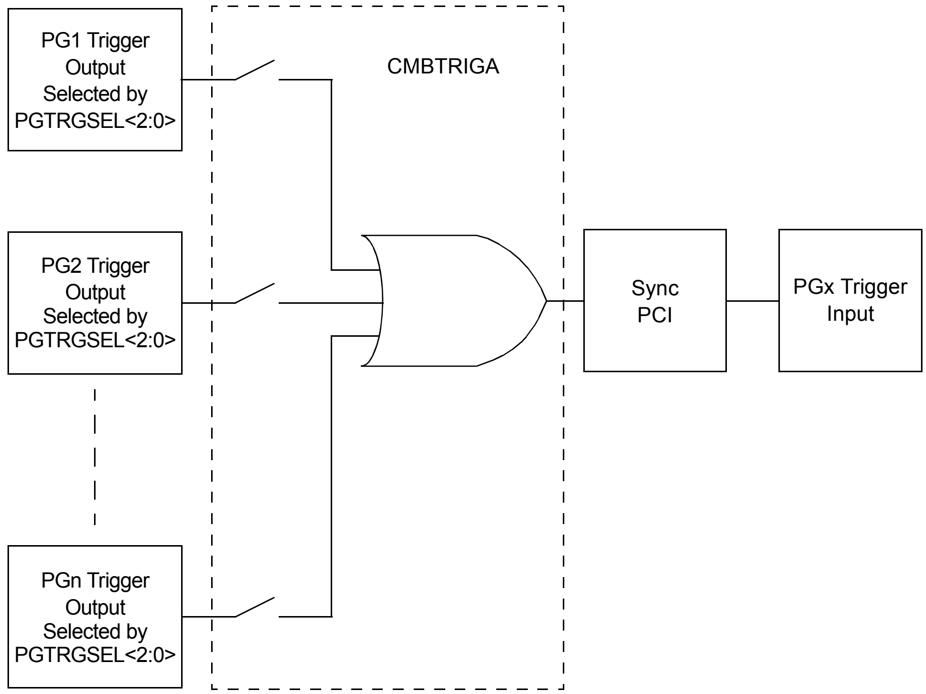
1. When using combinatorial logic, the two inputs from the PWM Generators must operate from the same clock source; otherwise, the outputs may not be valid. The minimum pulse width of the combinatorial output is device-specific and may be limited by the device pins.
2. The PWM Generator outputs selected as the source inputs are taken before the PWMx output polarity control, POLH/POLL (PGxIOCON1[1:0]). If no destination is selected, the combinatorial logic is disabled. The output destination is grouped into pairs where the odd LOGCONy registers (Instances A, C and E) can only be assigned to the PWMxH output pins, and the even LOGCONy registers (Instances B, D and F) can only be assigned to the PWMxL pins. Only PWM2-PWM8 can use the combinatorial logic output; PWM1 is not available. More than one instance (A-F) of a combinatorial logic output can be assigned to a single PWM output, if desired. In the case that multiple combinatorial logic functions have been enabled and assigned to the same PWM output, the function with the lowest letter value will take priority.

15.5.3.5. Combinatorial Triggers

Complex triggering algorithms can be created using the combinatorial trigger feature. There are two independent combinatorial trigger circuits: A and B. This feature allows trigger outputs from multiple PWM Generators to be combined into a single trigger signal, which can be used as the trigger source for another PWM Generator.

The input signals used as sources for the combinatorial trigger logic are the trigger outputs selected by the PGTRGSEL[2:0] control bits in each PGxEVT1 control register. This trigger output can either be End-of-Cycle (EOC) or one of the three PGxTRIGy (y = A, B or C) compare events. These trigger output signals can be enabled and logically OR'd together by setting the appropriate bits in the CMBTRIGH/CMBTRIGL registers. The Combinatorial Trigger A and Combinatorial Trigger B outputs are then made available on the PWM Control Input (PCI) logic input multiplexers and routed through the PCI logic for synchronization. Finally, the signal can then be selected as a PWM Generator's input trigger. A block diagram showing an example is shown in Figure 15-38. See PWM Generator Triggers for details on triggering.

Figure 15-38. Combinatorial Triggers Block Diagram, Example of Instance A



15.5.3.6. PWM Event Outputs

The PWM event output feature provides a mechanism to interface various PWM signals and events to other peripherals and external devices. The PWM event output logic provides a way to select and condition an event from any of the PWM Generators. Each PWM Event Output block has the following configuration options:

- PWM Generator Instance (PG1...PG8)
- Choice of Signal from PWM Generator
- Pulse Stretching
- Output Signal Polarity
- System Clock Synchronization
- Output Enable for the Signal

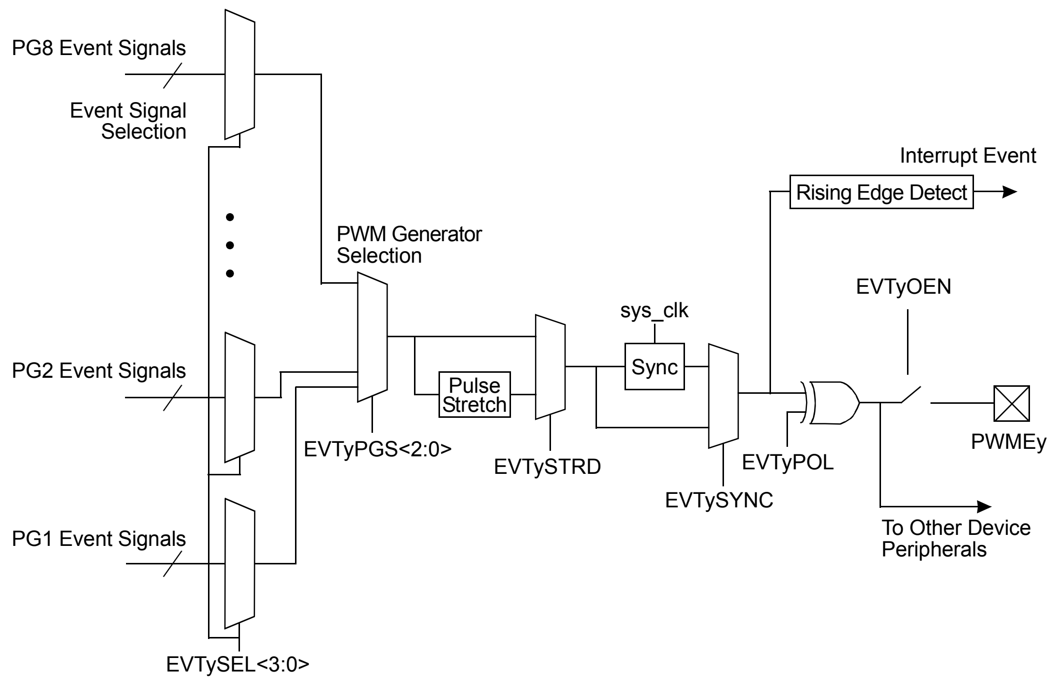
Each PWMEVTy register contains controls for a PWM event output. A device may have multiple instances (A-F) of the PWMEVTy registers, resulting in four or more total PWM event outputs.

The EVTySEL[3:0] (PWMEVTy[7:4]) bits select the signal to be used by the Output block. The default source is the selection determined by PGTRGSEL[2:0] (PGxEVT1[18:16]). For additional information on these signals and configuring the ADC triggers, see Event Selection Block. The EVTyPGS[2:0] bits (PWMEVTy[2:0]) are then used to select which of the eight PWM Generator's event signals is to be used.

Some of the event signals running at high speed have short pulses that may not be detected by other circuits, making it impossible, for example, to connect a PWM event signal to an off-chip destination. A pulse stretching circuit can be used to extend the duration of the pulse by setting the EVTySTRD bit (PWMEVTy[13]). If synchronization to the main PWM clock domain is desired, the EVTySYNC bit (PWMEVTy[12]) can be set. The EVTyPOL (PWMEVTy[14]) control is provided to invert the polarity of the event signal. Finally, an output enable bit, EVTyOEN (PWMEVTy[15]), is provided for control over the output pin PWMEy.

The PWM event output can also generate a system interrupt. An interrupt can be generated from any of the various triggers and events that are input into the Event Output block. A block diagram of the event output function is shown in Figure 15-39.

Figure 15-39. PWM Event Output Function



15.6. Interrupts

The interrupt sources within the PWM system are routed to the CPU in one of two ways:

1. Through a shared signal for each PWM Generator (see Event Interrupts). The signals include:
   a. EOC event
   b. TRIGA compare event
   c. ADC Trigger 1 event
   d. PCI_active event
2. Through the PWM Event Output block, there are various sources of interrupts that can be generated by the PWM module.

For example, a device which has eight PWM Generators and six PWM event outputs would have a total of 14 independent interrupt sources.

Since the PWM module can operate at a much higher frequency than the CPU system clock, care should be taken with the interrupt event configuration. Successive interrupt events on the same interrupt vector, that occur at a rate greater than the CPU can detect and service them, may result in missed processing and unexpected results. This limitation is dependent on the relationship of the system clock frequency, PWM operating frequency and the execution time of the Interrupt Service Routine (The number of instructions is irrelevant. What matters is how long the ISR takes to execute before it yields control back to the interrupted thread).

Interrupts from different sources that occur in close proximity to each other will also not be detected by the CPU as separate interrupt events. Therefore, it is good software practice to check the PWM status flags to differentiate a PCI interrupt event from a PWM time base interrupt. In some cases, it is desirable to separate different types of interrupt events that could be produced by the PWM Generator. When multiple PWM Generators have been synchronized together, it is possible to enable the time base interrupt on a separate PWM Generator that is synchronized to the host PWM Generator. Using this method, the time base interrupt can be serviced by a separate interrupt vector. It is also possible to bring various PWM event signals outside of the PWM module via the PWM Event blocks to an external off-chip destination.

15.7. Power-Saving Modes

This section discusses the operation of the High-Resolution PWM module in Sleep mode and Idle mode.

15.7.1. Sleep Mode

When the device enters Sleep mode, the clocks available to the PWM are disabled. All enabled PWM output pins that were in operation prior to entering Sleep mode will be frozen in their current output states. If the application circuit can be damaged by this condition, the outputs must be placed into a safe state before executing the `PWRSAV` instruction to enter Sleep mode.

15.7.2. Idle Mode

When the device enters Idle mode, the CPU is no longer clocked; however, the PWM remains clocked and operational. If the PWM module is controlling a power conversion/motor control application, the action of putting the device into Idle mode will cause any control loops to be disabled, and most applications will likely experience issues unless they explicitly operate in an Open-Loop mode. It is recommended that the outputs be placed into a safe state before executing the `PWRSAV` instruction to enter Idle mode.

15.8. Application Examples

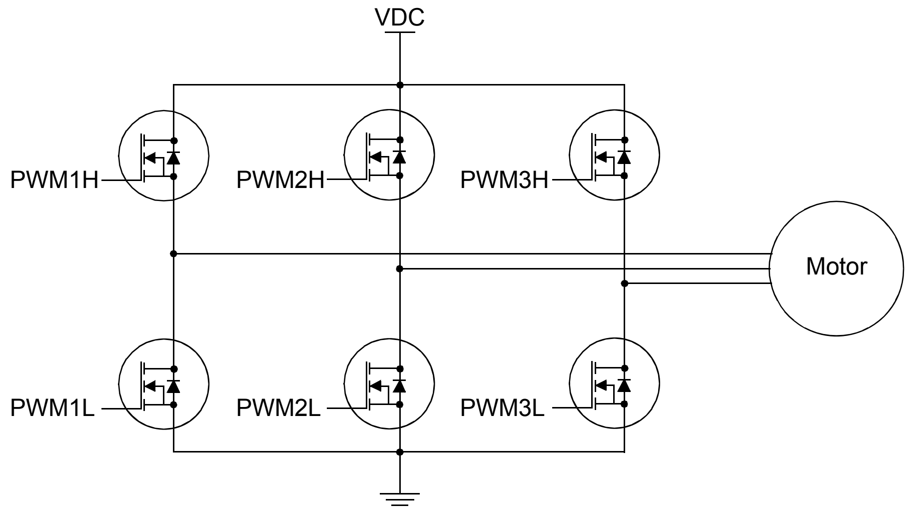
15.8.1. Six-Step Commutation of Three-Phase BLDC Motor

One method for controlling a three-phase Brushless DC (BLDC) is six-step commutation. Six-step commutation is also called 120° commutation, or trapezoidal control, and uses six steps or sectors over one electrical cycle to energize a BLDC motor. Each sector is equivalent to 60 electrical degrees with the six sectors resulting in 360 electrical degrees or one electrical cycle.

Sequencing through these steps moves the motor through one electrical cycle, which in mechanical terms, corresponds to one pair of rotor magnet poles moving past stator windings. A given BLDC motor has a certain number of pole pairs, defined by Np as a positive integer. If the motor is rotated one mechanical revolution, this corresponds to Np electrical cycles. This yields one electrical cycle for a 2-pole motor, two electrical cycles for a 4-pole motor, three electrical cycles for a 6-pole motor and so on.

A six-step commutation drive is typically implemented using a three-phase bridge circuit, as shown in Figure 15-40. Each phase of the motor is connected to a half-bridge driver and controlled with a complementary PWM pair output. At any given time in the six-step commutation scheme, only two of the three motor windings are energized. Current in the motor winding flows from one phase to another in either direction.

Figure 15-40. Three-Phase Bridge and Motor



Various PWM switching techniques can be employed for six-step commutation. Table 15-14 summarizes the three schemes presented in this manual.

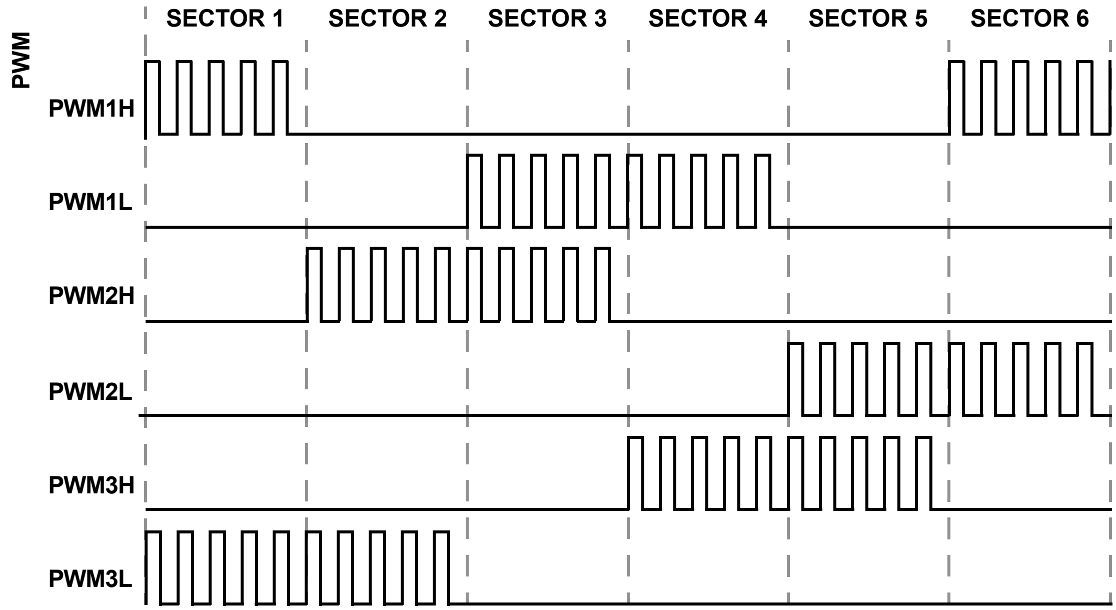
Table 15-14. PWM Switching Schemes for 6-Step Commutation

Scheme	Technique Overview	Advantage	Disadvantage
Scheme1	High side of one active phase and low side of the other active phase driven at any given time.	Simplest scheme; no dead time needed; low switching loss	High-current ripple
Scheme2	One active phase driven complementary and the other active phase's low side is driven to 100% duty cycle.	Low switching loss	Requires dead time
Scheme 3	Both active phases are driven complementary.	Lowest current ripple	Higher switching loss and requires dead time

15.8.1.1. Six-Step Commutation – PWM Scheme 1

In this PWM scheme, only two switches are active at any given time. Of the two active phases, one high-side and one low-side switch are controlled with their phase's corresponding PWM waveform, as shown in Figure 15-41.

Figure 15-41. Six-Step PWM Scheme 1 Waveform



Since only one switch needs to be driven at a time on a given phase, Independent PWM Output mode is used. The output override feature is then used to suppress the unused output. A three-phase scheme is implemented using PWM Generator 1 (PG1) configured as a host and the other two PWM Generators (PG2 and PG3) configured as clients. PG1 is self-triggered, whereas PG2 and PG3 are triggered from PG1's Start-of-Cycle (SOC). Enabling PG1 will start the system in a synchronized fashion.

Configuration Summary:

- Independent Edge PWM mode
- Independent Output mode
- Master period and Duty cycle used
- Override state is drive low.

Example 15-1. Six-Step PWM Scheme 1 Code

```
#include <stdint.h>

//For delay function
#define FCY 8000000           //CPU frequency in Hz
#include "libpic30.h"

uint32_t state = 0;
uint32_t PWMState1[6] =
{0x00100000,0x00100000,0x00300000,0x00200000,0x00200000,0x00300000};
uint32_t PWMState2[6] =
{0x00200000,0x00300000,0x00100000,0x00100000,0x00300000,0x00200000};
uint32_t PWMState3[6] =
{0x00300000,0x00200000,0x00200000,0x00300000,0x00100000,0x00100000};

void PWMInitialization(void);

int main(void)
{

    PWMInitialization();
    while (1)
    {
        for (state = 0; state < 6; state++)
        {
            /* Delay is used to simulate BLDC commutation; In practical
            application, commutation state transition will be based on feedback from Motor
```

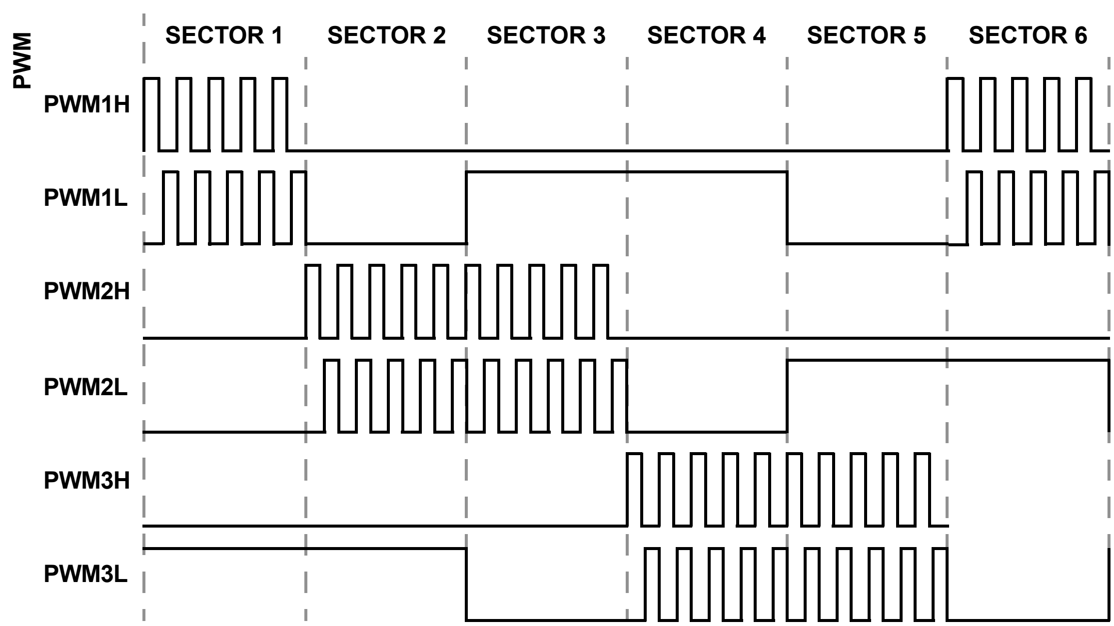
```
*/
            __delay_us(200);
            PG1IOCON2 = PWMState1[state];
            PG2IOCON2 = PWMState2[state];
            PG3IOCON2 = PWMState3[state];
        }
    }
}

void PWMInitialization(void)
{
    PCLKCON      = 0x0000;

    //Set PWM master clock to 400MHz from PLL2 through CLKGEN5
    configure_PLL2_Fout_400MHz();
    clock_PWM_from_PLL2_Fout();

    /* PWM Clock Divider Selection bits
       0b11 = 1:16 ; 0b10 = 1:8 ;0b01 = 1:4 ; 0b00 = 1:2 */
    PCLKCONbits.DIVSEL = 0;
    /* PWM Master Clock Selection bits
       0b01 = CLKGEN5 ; 0b00 = UPB clock */
    PCLKCONbits.MCLKSEL = 1;
    /* Lock bit: 0 = Write-protected registers and bits are unlocked */
    PCLKCONbits.LOCK = 0;

    /* Initialize Master Period Register */
    MPER         = 80000;
    /* Initialize Master Duty Cycle */
    MDC          = 40000;

    /* Initialize PG1CON Registers */
    PG1CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG1CONbits.CLKSEL = 1;
    /* Select PWM Generator duty cycle register as MDC */
    PG1CONbits.MDCSEL  = 1;
    /* Select PWM Generator period register as MPER */
    PG1CONbits.MPERSEL = 1;
    /* PWM Generator broadcasts software set of UPDREQ control bit and EOC signal
     * to other PWM Generators. */
    PG1CONbits.MSTEN = 1;
    /* Start of cycle is local EOC */
    PG1CONbits.SOCS = 0b0000;
    /* PWM Generator operates in Independent Edge PWM mode*/
    PG1CONbits.MODSEL = 0;


   /* Initialize PG1IOCON Registers */
    PG1IOCON1 = 0x00000000;

    /* PWM Generator Output Mode is Independent Mode */
    PG1IOCON1bits.PMOD = 1;
    /* PWM Generator controls the PWMxH output pin */
    PG1IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG1IOCON1bits.PENL = 1;

    PG1EVT1   = 0x00000000;
    /* A write of the PGxDC register automatically sets the UPDREQ bit */
    PG1EVT1bits.UPDTRG = 0;
    /* PWM generator trigger output is EOC*/
    PG1EVT1bits.PGTRGSEL = 0;

    /* Initialize PG2CON Registers */
    PG2CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG2CONbits.CLKSEL = 1;
    /* Select PWM Generator Duty Cycle Register as MDC */
    PG2CONbits.MDCSEL = 1;
    /* Select PWM Generator Period Register as MPER */
    PG2CONbits.MPERSEL = 1;
    /* Start of Cycle is PG1 trigger output selected by
     * PG1EV1Tbits.PGTRGSEL<2:0> bits */
    PG2CONbits.SOCS = 0b0001;
    /* PWM Generator operates in Independent Edge PWM mode*/
    PG2CONbits.MODSEL = 0;

    /* Initialize PG2IOCON Registers */
    PG2IOCON1 = 0x00000000;

    /* PWM Generator Output Mode is Independent Mode */
    PG2IOCON1bits.PMOD = 1;
    /* PWM Generator controls the PWMxH output pin */
    PG2IOCON1bits.PENH = 1;
```

```
    /* PWM Generator controls the PWMxL output pin */
    PG2IOCON1bits.PENL = 1;

    /* Initialize PG3CON Registers */
    PG3CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG3CONbits.CLKSEL = 1;
    /* Select PWM Generator Duty Cycle Register as MDC */
    PG3CONbits.MDCSEL = 1;
    /* Select PWM Generator Period Register as MPER */
    PG3CONbits.MPERSEL = 1;
    /* Start of Cycle is PG1 trigger output selected by
     * PG1EVT1bits.PGTRGSEL<2:0> bits */
    PG3CONbits.SOCS = 0b0001;
    /* PWM Generator operates in Independent Edge PWM mode*/
    PG3CONbits.MODSEL = 0;

    /* Initialize PG3IOCON Registers */
    PG3IOCON1 = 0x00000000;

    /* PWM Generator Output Mode is Independent Mode */
    PG3IOCON1bits.PMOD = 1;
    /* PWM Generator controls the PWMxH output pin */
    PG3IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG3IOCON1bits.PENL = 1;

    /* Enable PWM generator 3 */
    PG3CONbits.ON =1;
    /* Enable PWM generator 2 */
    PG2CONbits.ON =1;
    /* Enable PWM generator 1, starting all PWM generators together */
    PG1CONbits.ON =1;
}
```

15.8.1.2. Six-Step Commutation – PWM Scheme 2

In this PWM scheme, three switches are used to control the two active phases. In a given sector, one active phase is driven with a complementary PWM waveform, and the other active phase has only its low side driven low at 100% duty cycle, as shown in Figure 15-42. Like Scheme 1, overrides are used to control the outputs in each sector.

Figure 15-42. Six-Step PWM Scheme 2 Waveform



In this scheme, Complementary Output mode is used and overridden as needed in each sector. The same three-phase host/client synchronization technique is used as in Scheme 1.

Configuration Summary:

- Independent Edge PWM mode
- Complementary Output mode
- Master period and duty cycle used
- Override state is dependent on Sector state.
- Dead time is applied to the Complementary PWM signal.

Example 15-2. Six-Step PWM Scheme 2 Code

```
#include <stdint.h>
#include <stdlib.h>

//For delay function
#define FCY 8000000     //CPU frequency in Hz
#include "libpic30.h"

uint32_t state = 0;
uint32_t PWMState1[6] =
{0x00000000,0x00300000,0x00301000,0x00301000,0x00300000,0x00000000};
uint32_t PWMState2[6] =
{0x00300000,0x00000000,0x00000000,0x00300000,0x00301000,0x00301000};
uint32_t PWMState3[6] =
{0x00301000,0x00301000,0x00300000,0x00000000,0x00000000,0x00300000};

void PWMInitialization(void);

int main(void)
{
    PWMInitialization();

    while (1)
    {
        for (state = 0; state < 6; state++)
        {

            /* Delay is used to simulate BLDC commutation; In practical
            application, commutation state transition will be based on
feedback from Motor */

            __delay_us(200);
            PG1IOCON2 = PWMState1[state];
            PG2IOCON2 = PWMState2[state];
            PG3IOCON2 = PWMState3[state];
        }
    }
}

void PWMInitialization(void)
{

    PCLKCON      = 0x0000;

    //Set PWM master clock to 400MHz from PLL2 through CLKGEN5
    configure_PLL2_Fout_400MHz();
    clock_PWM_from_PLL2_Fout();

    /* PWM Clock Divider Selection bits
       0b11 = 1:16 ; 0b10 = 1:8 ;0b01 = 1:4 ; 0b00 = 1:2 */
    PCLKCONbits.DIVSEL = 0;
    /* PWM Master Clock Selection bits
       0b01 = CLKGEN5 ; 0b00 = UPB clock */
    PCLKCONbits.MCLKSEL = 1;
    /* Lock bit: 0 = Write-protected registers and bits are unlocked */
    PCLKCONbits.LOCK = 0;

    /* Initialize Master Period Register */
    MPER         = 80000;
    /* Initialize Master Duty Cycle */
    MDC          = 40000;

    /* Initialize PG1CON Registers */
```

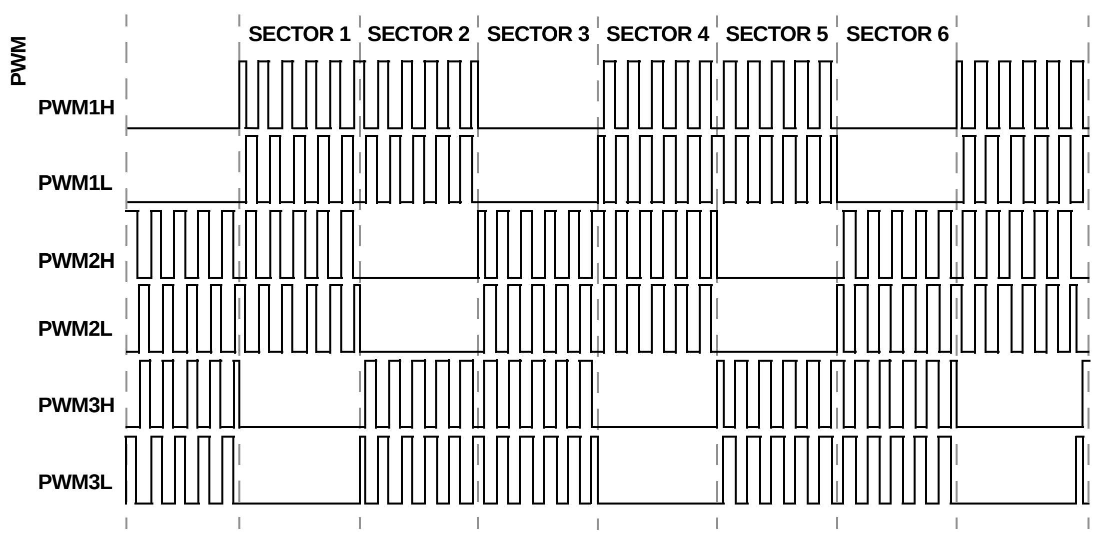
```
    PG1CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG1CONbits.CLKSEL = 1;
    /* Select PWM Generator duty cycle register as MDC */
    PG1CONbits.MDCSEL  = 1;
    /* Select PWM Generator period register as MPER */
    PG1CONbits.MPERSEL = 1;
    /* PWM Generator broadcasts software set of UPDREQ control bit and EOC
signal
     * to other PWM Generators. */
    PG1CONbits.MSTEN = 1;
    /* Start of cycle is local EOC */
    PG1CONbits.SOCS = 0b0000;
    /* PWM Generator operates in Independent Edge PWM mode*/
    PG1CONbits.MODSEL = 0;

    /* Initialize PG1IOCON Registers */
    PG1IOCON1 = 0x00000000;

    /* PWM Generator Output Mode is Complementary Mode */
    PG1IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG1IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG1IOCON1bits.PENL = 1;

    PG1EVT1   = 0x00000000;
    /* A write of the PGxDC register automatically sets the UPDREQ bit */
    PG1EVT1bits.UPDTRG = 0;
    /* PWM generator trigger output is EOC*/
    PG1EVT1bits.PGTRGSEL = 0;

    /* Initialize PG2CON Registers */
    PG2CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG2CONbits.CLKSEL = 1;
    /* Select PWM Generator Duty Cycle Register as MDC */
    PG2CONbits.MDCSEL = 1;
    /* Select PWM Generator Period Register as MPER */
    PG2CONbits.MPERSEL = 1;
    /* Start of Cycle is PG1 trigger output selected by
     * PG1EVTbits.PGTRGSEL<2:0> bits */
    PG2CONbits.SOCS = 0b0001;
    /* PWM Generator operates in Independent Edge PWM mode*/
    PG2CONbits.MODSEL = 0;

 /* Initialize PG2IOCON Registers */
    PG2IOCON1 = 0x00000000;

    /* PWM Generator Output Mode is Complementary Mode */
    PG2IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG2IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG2IOCON1bits.PENL = 1;

    /* Initialize PG3CON Registers */
    PG3CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG3CONbits.CLKSEL = 1;
    /* Select PWM Generator Duty Cycle Register as MDC */
    PG3CONbits.MDCSEL = 1;
    /* Select PWM Generator Period Register as MPER */
    PG3CONbits.MPERSEL = 1;
    /* Start of Cycle is PG1 trigger output selected by
     * PG1EVTbits.PGTRGSEL<2:0> bits */
    PG3CONbits.SOCS = 0b0001;
    /* PWM Generator operates in Independent Edge PWM mode*/
    PG3CONbits.MODSEL = 0;

    /* Initialize PG3IOCON Registers */
    PG3IOCON1 = 0x00000000;
```

```
    /* PWM Generator Output Mode is Complementary Mode */
    PG3IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG3IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG3IOCON1bits.PENL = 1;

    /* Initialize PWM GENERATOR 3 DEAD-TIME REGISTER */
    PG3DT = 0x06400640;
    /* Initialize PWM GENERATOR 2 DEAD-TIME REGISTER */
    PG2DT = 0x06400640;
    /* Initialize PWM GENERATOR 1 DEAD-TIME REGISTER */
    PG1DT = 0x06400640;

    /* Enable PWM generator 3 */
    PG3CONbits.ON =1;
    /* Enable PWM generator 2 */
    PG2CONbits.ON =1;
    /* Enable PWM generator 1, starting all PWM generators together */
    PG1CONbits.ON =1;
}
```

15.8.1.3. Six-Step Commutation – PWM Scheme 3

In this PWM scheme, four switches are driven in a given sector. Two pairs of complementary PWM outputs are applied to the two active phases. The inactive phase is overridden low as needed, as shown in Figure 15-43.

Figure 15-43. Six-Step PWM Scheme 3 Waveform



In this scheme, Center-Aligned PWM mode is used with dead time to prevent high current during switching transitions. The two active phases are driven 180 degrees out of phase to one another using the SWAP feature.

Configuration Summary:

- Center-Aligned PWM mode
- Complementary Output mode
- Master period and duty cycle used
- Override and SWAP state are dependent on Sector state.
- Dead time is applied to the Complementary PWM Signal.

Example 15-3. Six-Step PWM Scheme 3 Code

```
//For delay function
#define FCY 8000000 //CPU frequency in Hz
#include "libpic30.h"

#include <xc.h>
#include <stdint.h>

uint32_t state = 0;
uint32_t PWMState1[6] =
{0x0000000C,0x0000000C,0x0000000C,0x0000080C,0x0000080C,0x0000080C};
uint32_t PWMState2[6] =
{0x00000000,0x00000000,0x00300000,0x00000000,0x00000000,0x00300000};
uint32_t PWMState3[6] =
{0x0000080C,0x0000080C,0x0000000C,0x0000000C,0x0000000C,0x0000080C};
uint32_t PWMState4[6] =
{0x00000000,0x00300000,0x00000000,0x00000000,0x00300000,0x00000000};
uint32_t PWMState5[6] =
{0x0000000C,0x0000080C,0x0000080C,0x0000080C,0x0000000C,0x0000000C};
uint32_t PWMState6[6] =
{0x00300000,0x00000000,0x00000000,0x00300000,0x00000000,0x00000000};

void PWMInitialization(void);

int main(void)
{
    PWMInitialization();

    /* To Update Duty cycle values to PG1-PG3, add two lines of code written
    below (In this example, setting 50% Duty Cycle):
    After writing the MDC register, set the Update request bit
PG1STATbits.UPDREQ.
    This will transfer the MDC value to all the PWM generators PG1-PG3.
    Note that the Update Mode (UPDMOD) of PG2, PG3 is Client EOC and
    PG1 MSTEN bits is '1' */

    PG1STATbits.UPDREQ = 1;

    while (1)
    {
        for (state=0; state<6;state++)
        {
            /* Delay is used to simulate BLDC commutation; In practical
            application, commutation state transition will be based on
feedback from Motor */
            __delay_us(200);

            PG1IOCON1 = PWMState1[state];
            PG1IOCON2 = PWMState2[state];

            PG2IOCON1 = PWMState3[state];
            PG2IOCON2 = PWMState4[state];

            PG3IOCON1 = PWMState5[state];
            PG3IOCON2 = PWMState6[state];
        }
    }
}


void PWMInitialization(void)
{

    PCLKCON      = 0x0000;

    //Set PWM master clock to 400MHz from PLL2 through CLKGEN5
    configure_PLL2_Fout_400MHz();
    clock_PWM_from_PLL2_Fout();

    /* PWM Clock Divider Selection bits
       0b11 = 1:16 ; 0b10 = 1:8 ;0b01 = 1:4 ; 0b00 = 1:2 */
    PCLKCONbits.DIVSEL = 0;
    /* PWM Master Clock Selection bits
       0b01 = CLKGEN5 ; 0b00 = UPB clock */
    PCLKCONbits.MCLKSEL = 1;
```

```
    /* Lock bit: 0 = Write-protected registers and bits are unlocked */
    PCLKCONbits.LOCK = 0;

    /* Initialize Master Period Register */
    MPER          = 40000;
    /* Initialize Master Duty Cycle */
    MDC           = 20000;

    /* Initialize PG1CON Registers */
    PG1CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG1CONbits.CLKSEL = 1;
    /* Select PWM Generator duty cycle register as MDC */
    PG1CONbits.MDCSEL  = 1;
    /* Select PWM Generator period register as MPER */
    PG1CONbits.MPERSEL = 1;
    /* PWM Generator broadcasts software set of UPDREQ control bit and EOC
signal
     * to other PWM Generators. */
    PG1CONbits.MSTEN = 1;
    /* Start of cycle is local EOC */
    PG1CONbits.SOCS = 0b0000;
    /* PWM Generator operates in Center-Aligned PWM mode*/
    PG1CONbits.MODSEL = 4;

    /* Initialize PG1IOCON Registers */
    PG1IOCON1 = 0x00000000;

    /* PWM Generator Output Mode is Complementary Mode */
    PG1IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG1IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG1IOCON1bits.PENL = 1;

    PG1EVT1   = 0x00000000;
    /* A write of the PGxDC register automatically sets the UPDREQ bit */
    PG1EVT1bits.UPDTRG = 0;
    /* PWM generator trigger output is EOC*/
    PG1EVT1bits.PGTRGSEL = 0;

    /* Initialize PG2CON Registers */
    PG2CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG2CONbits.CLKSEL = 1;
    /* Select PWM Generator Duty Cycle Register as MDC */
    PG2CONbits.MDCSEL = 1;
    /* Select PWM Generator Period Register as MPER */
    PG2CONbits.MPERSEL = 1;
    /* Start of Cycle is PG1 trigger output selected by
     * PG1EVTbits.PGTRGSEL<2:0> bits */
    PG2CONbits.SOCS = 0b0001;
    /* PWM Generator operates in Center-Aligned PWM mode*/
    PG2CONbits.MODSEL = 4;

    /* Initialize PG2IOCON Registers */
    PG2IOCON1 = 0x00000000;
    /* PWM Generator Output Mode is Complementary Mode */
    PG2IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG2IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG2IOCON1bits.PENL = 1;

    /* Initialize PG3CON Registers */
    PG3CON = 0x00000000;

    /*PWM Generator uses the master clock selected by the MCLKSEL[1:0]
     * (PCLKCON[1:0] control bits */
    PG3CONbits.CLKSEL = 1;
    /* Select PWM Generator Duty Cycle Register as MDC */
    PG3CONbits.MDCSEL = 1;
    /* Select PWM Generator Period Register as MPER */
    PG3CONbits.MPERSEL = 1;
```

```
    /* Start of Cycle is PG1 trigger output selected by
     * PG1EVTbits.PGTRGSEL<2:0> bits */
    PG3CONbits.SOCS = 0b0001;
    /* PWM Generator operates in Center-Aligned PWM mode*/
    PG3CONbits.MODSEL = 4;

    /* Initialize PG3IOCON Registers */
    PG3IOCON1 = 0x00000000;
    /* PWM Generator Output Mode is Complementary Mode */
    PG3IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG3IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG3IOCON1bits.PENL = 1;

    /* Initialize PWM GENERATOR 3 DEAD-TIME REGISTER */
    PG3DT = 0x06400640;
    /* Initialize PWM GENERATOR 2 DEAD-TIME REGISTER */
    PG2DT = 0x06400640;
    /* Initialize PWM GENERATOR 1 DEAD-TIME REGISTER */
    PG1DT = 0x06400640;

    /* Enable PWM generator 3 */
    PG3CONbits.ON =1;
    /* Enable PWM generator 2 */
    PG2CONbits.ON =1;
    /* Enable PWM generator 1, starting all PWM generators together */
    PG1CONbits.ON =1;
}
```

15.8.2. Three-Phase Sinusoidal Control of PMSM/ACIM Motors

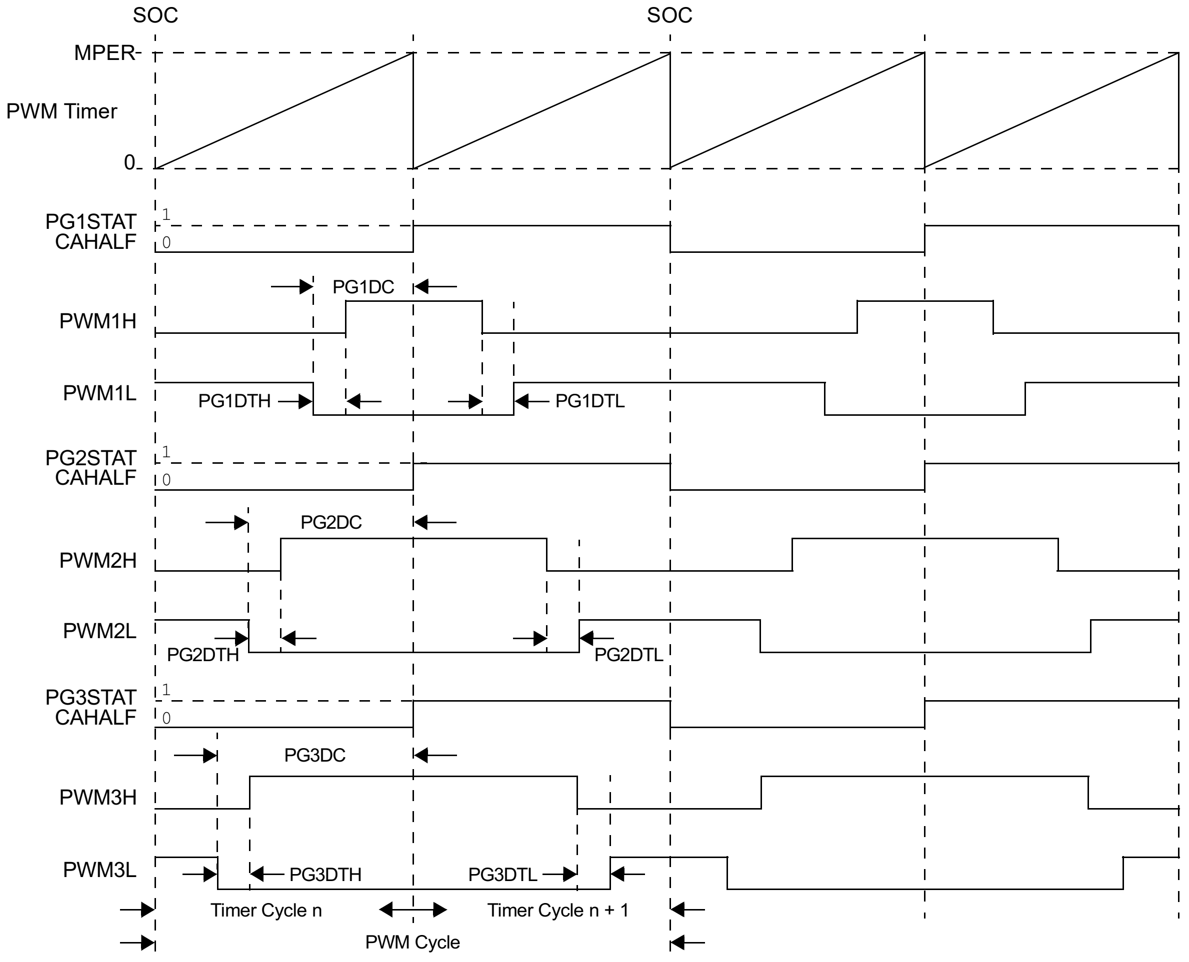
Three-phase sinusoidal control applies voltages to the three-phase motor windings, which are pulse-width modulated to produce sinusoidal currents as the motor spins. This eliminates the torque ripple and commutation spikes associated with trapezoidal commutation. Typically, Center-Aligned Complementary PWMs are used for sinusoidal control of a Permanent Magnet Synchronous Motor (PMSM) or a three-phase AC Induction Motor (ACIM). Center-aligned PWM signals reduce the level of harmonics in output voltages and currents as compared to edge-aligned PWMs. Three PWM Generators are connected to the three-phase power bridge driving the motor, as shown in Figure 15-40.

PWM Generator 1 is configured as a host, and PWM Generator 2 and PWM Generator 3 are configured as client PWMs. PWM configuration used in three-phase sinusoidal control is summarized below:

- PG1-PG3: Uses master period and independent duty cycles.
- PG1-PG3: PWM Operating mode is selected as Center-Aligned mode.
- PG1-PG3: Configured to operate in Single Trigger mode.
- PG1-PG3: PWM Output mode is configured as Complementary Output mode.
- PG2-PG3: Uses PG1 trigger output as Start-of-Cycle, whereas PG1 is self-triggered.
- PG2-PG3: Enabled during initialization.
- PG1 is enabled only after configuring all the control registers; whenever PG1 is enabled, all the generators will start in unison.

Figure 15-44 shows the PWM waveforms for a given point in time. Center-Aligned mode uses two timer cycles to produce a PWM cycle and maintains symmetry at the center of each PWM cycle. Each timer cycle can be tracked using the status bit, CAHALF (PGxSTAT[1]), of the respective PWM Generator. The leading edge is produced when CAHALF = `0`, and the falling edge is produced when CAHALF = `1`. Note that with Center-Aligned mode, as long as the duty cycles are different for each phase, the switching instants occur at different times. (In Edge-Aligned mode, the turn-on times are coincident.) This generally reduces electromagnetic interference.

Figure 15-44. Three-Phase Center-Aligned Waveforms



Example 15-4. Three-Phase Sinusoidal PMSM/ACIM Motor Control Code

```
void PWMInitialization(void);

#include <xc.h>
#include <stdint.h>

int main(void)
{

    PWMInitialization();

    while (1)
    {

    }
}

void PWMInitialization(void)
{
    PCLKCON      = 0x0000;

    //Set PWM master clock to 400MHz from PLL2 through CLKGEN5
    configure_PLL2_Fout_400MHz();
    clock_PWM_from_PLL2_Fout();

    /* PWM Clock Divider Selection bits
```

```
       0b11 = 1:16 ; 0b10 = 1:8 ;0b01 = 1:4 ; 0b00 = 1:2 */
    PCLKCONbits.DIVSEL = 0;
    /* PWM Master Clock Selection bits
       0b01 = CLKGEN5 ; 0b00 = UPB clock */
    PCLKCONbits.MCLKSEL = 1;
    /* Lock bit: 0 = Write-protected registers and bits are unlocked */
    PCLKCONbits.LOCK = 0;

    /* Initialize Master Period Register */
    MPER          = 40000;
    /* Set PWM Phase Register - No phase shift */
    MPHASE = 0;
    /* Set PWM Duty Cycles */
    PG1DC = 10000;
    PG2DC = 20000;
    PG3DC = 30000;
    /* Set Dead Time Registers */
    PG1DT = 0x06400640;
    PG2DT = 0x06400640;
    PG3DT = 0x06400640;

    PG1CON = 0x00000000;
    /* Select PWM Generator Duty Cycle Register as PG1DC */
    PG1CONbits.MDCSEL = 0;
    /* Select PWM Generator Period Register as MPER */
    PG1CONbits.MPERSEL = 1;
    /* Select PWM Generator Phase Register as MPHASE */
    PG1CONbits.MPHSEL = 1;
    /* PWM Generator broadcasts software set of UPDREQ *
    * control bit and EOC signal to other PWM generators. */
    PG1CONbits.MSTEN = 1;
    /* Start of Cycle is Local EOC */
    PG1CONbits.SOCS = 0;
    /* PWM Generator uses Master Clock selected by
    the PCLKCONbits.MCLKSEL bits */
    PG1CONbits.CLKSEL = 1;
    /* PWM Generator operates in Center-Aligned mode*/
    PG1CONbits.MODSEL = 4;

    PG1IOCON1 = 0x00000000;
    /* PWM Generator Output operates in Complementary Mode */
    PG1IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG1IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG1IOCON1bits.PENL = 1;

    PG1EVT1 = 0x00000000;
    /* Time base interrupts are disabled */
    PG1EVT1bits.IEVTSEL = 3;
    /* A write of the PGxDC register automatically sets the UPDREQ bit */
    PG1EVT1bits.UPDTRG = 0;
    /* PWM generator trigger output is EOC*/
    PG1EVT1bits.PGTRGSEL = 0;

    PG2CON = 0x00000000;
    /* Select PWM Generator Duty Cycle Register as PG2DC */
    PG2CONbits.MDCSEL = 0;
    /* Select PWM Generator Period Register as MPER */
    PG2CONbits.MPERSEL = 1;
    /* Select PWM Generator Phase Register as MPHASE */
    PG2CONbits.MPHSEL = 1;
    /* Start of Cycle is PG1 trigger output selected by
    PG1EVTbits.PGTRGSEL<2:0> bits */
    PG2CONbits.SOCS = 1;
    /* PWM Generator uses Master Clock selected by
    the PCLKCONbits.MCLKSEL bits */
    PG2CONbits.CLKSEL = 1;
    /* PWM Generator operates in Center-Aligned mode */
    PG2CONbits.MODSEL = 4;

    PG2IOCON1 = 0x00000000;
    /* PWM Generator output operates in Complementary Mode */
    PG2IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG2IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG2IOCON1bits.PENL = 1;
```

```
    PG3CON = 0x00000000;
    /* Select PWM Generator Duty Cycle Register as PG2DC */
    PG3CONbits.MDCSEL = 0;
    /* Select PWM Generator Period Register as MPER */
    PG3CONbits.MPERSEL = 1;
    /* Select PWM Generator Phase Register as MPHASE */
    PG3CONbits.MPHSEL = 1;
    /* Start of Cycle is PG1 trigger output selected by
    PG1EVTbits.PGTRGSEL<2:0> bits */
    PG3CONbits.SOCS = 1;
    /* PWM Generator uses Master Clock selected by
    the PCLKCONbits.MCLKSEL bits */
    PG3CONbits.CLKSEL = 1;
    /* PWM Generator operates in Center-Aligned mode */
    PG3CONbits.MODSEL = 4;



    PG3IOCON1 = 0x00000000;
    /* PWM Generator output operates in Complementary Mode */
    PG3IOCON1bits.PMOD = 0;
    /* PWM Generator controls the PWMxH output pin */
    PG3IOCON1bits.PENH = 1;
    /* PWM Generator controls the PWMxL output pin */
    PG3IOCON1bits.PENL = 1;

    /* Turning ON the PWM Generator 3 */
    PG3CONbits.ON = 1;
    /* Turning ON the PWM Generator 2 */
    PG2CONbits.ON = 1;
    /* Turning ON the PWM Generator 1;
    Thus starting all the PWM modules in unison */
    PG1CONbits.ON = 1;
}
```

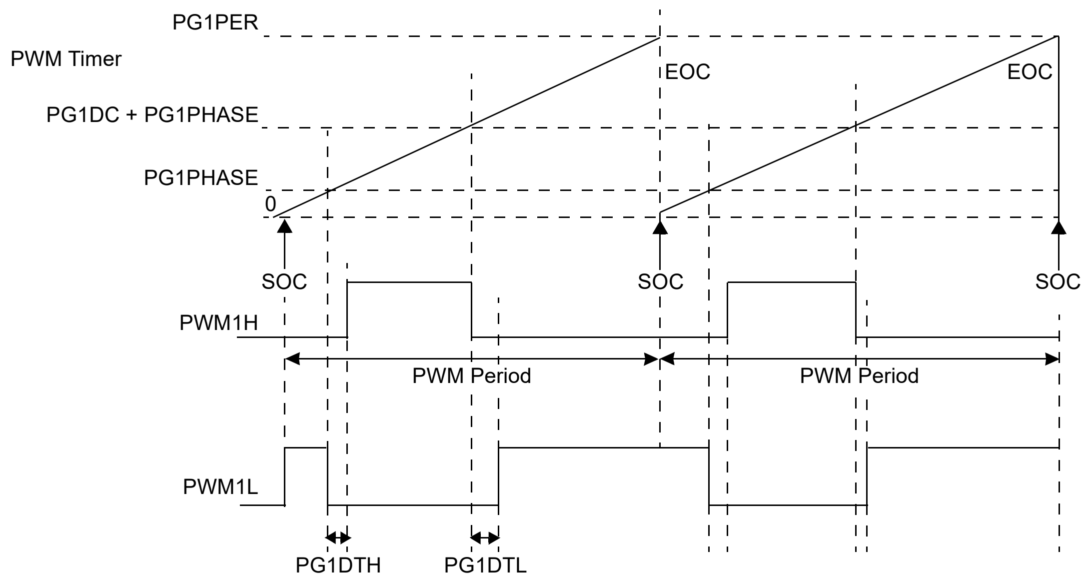
15.8.3. Simple Complementary PWM Output

This complementary PWM example uses a single PWM Generator and can be used for half-bridge applications. The PWM is configured as follows:

- Independent Edge PWM mode
- Complementary Output mode
- Self-Triggered mode

Figure 15-45 shows the timing relations of the PWM signals. In this example, continuous triggering (local EOC) is used in addition to a phase offset (PG1PHASE). Dead time is implemented to prevent simultaneous switch conduction.

Figure 15-45. Timing Diagram for Complementary and Local EOC Triggered PWM Output



Example 15-5. Complementary PWM Output Mode

```
void PWMInitialization(void);

int main() {

    PWMInitialization();

    while(1);

    return 0;
}

void PWMInitialization(void) {
    clock_PWM_from_UPB_clock();

    //PWM Generator 1 uses PG1DC, PG1PER, PG1PHASE registers
    PG1CONbits.MDCSEL = 0;
    PG1CONbits.MPERSEL = 0;
    PG1CONbits.MPHSEL = 0;

    PG1CONbits.CLKSEL = 1;        //PWM Generator 1 uses PWM Master Clock, undivided and
unscaled
    PG1CONbits.MODSEL = 0b000;    //Independent edge triggered mode
    PG1CONbits.TRGMOD = 0b00;     //PWM Generator 1 operates in Single Trigger mode
    PG1CONbits.SOCS = 0b0000;     //Start of cycle (SOC) = local EOC

    PG1IOCON1bits.PMOD = 0b00;   //PWM Generator 1 outputs operate in Complementary mode

    //PWM Generator controls the PWM1H and PWM1L output pins
    PG1IOCON1bits.PENH = 1;
    PG1IOCON1bits.PENL = 1;

    //PWM1H and PWM1L output pins are active high
    PG1IOCON1bits.POLH = 0;
    PG1IOCON1bits.POLL = 0;

    //Given the 400MHz input clock from CLKGEN5, this period will result in 100kHz PWM
frequency
    PG1PER = (4000 << 4);       //Time base units are 1/16 of a PWM clock
    PG1DC = (1000 << 4);        //25% duty cycle
    PG1PHASE = (400 << 4);      //Rising edge has 400 PWM clocks of phase time
    PG1DTbits.DTH = (80 << 4);  //80 PWM clocks of dead time on PWM1H
    PG1DTbits.DTL = (80 << 4);  //80 PWM clocks of dead time on PWM1L

    //Enable PWM Generator 1
    PG1CONbits.ON = 1;

}
```

15.8.4. Cycle-by-Cycle Current Limit Mode

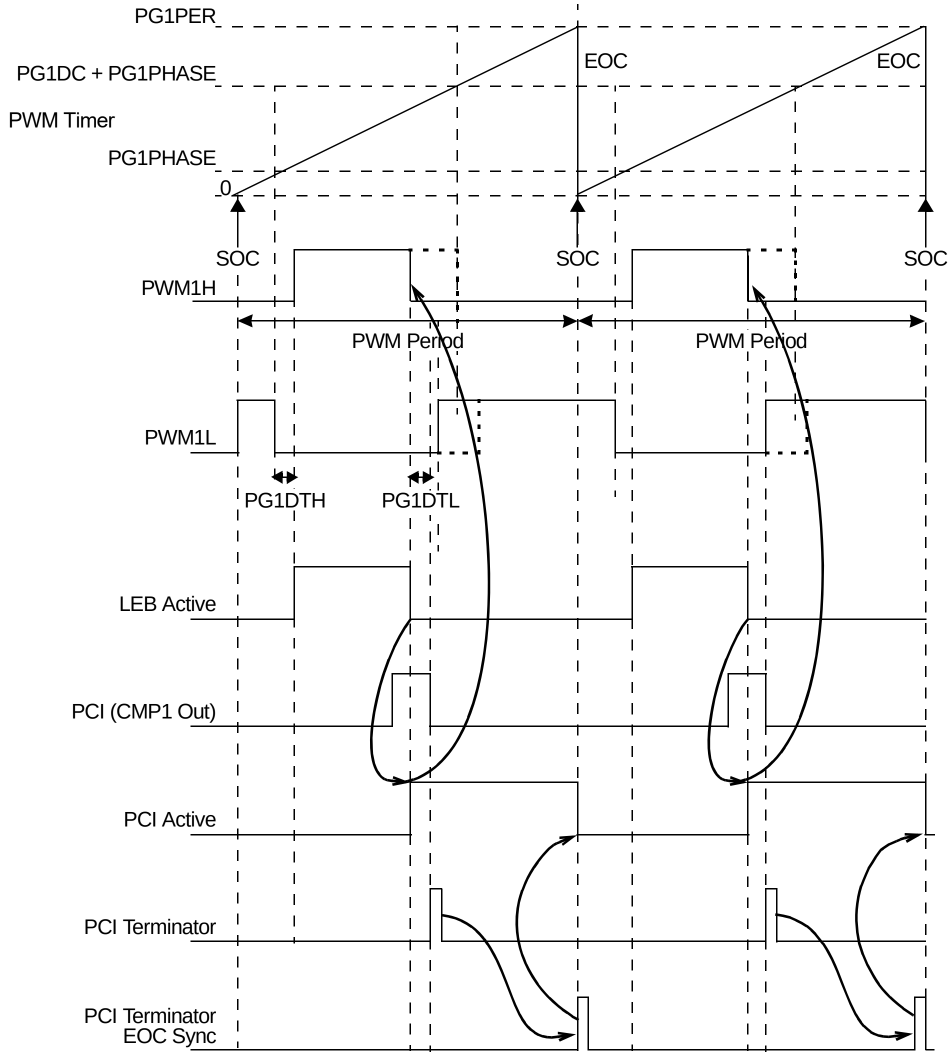
Cycle-by-Cycle Current Limit mode is a widely adopted control strategy for power applications and motor control. Current is measured and limited to a predetermined level using the internal comparator module. Cycle-by-Cycle Current Limit mode control for BLDC motors can automatically limit motor phase currents to a predetermined maximum value, using a comparator to provide an input to the PCI block. This has the advantage of allowing operation to continue when the limit is reached, rather than triggering a Fault. The PWM is configured as follows:

- Independent Edge PWM mode
- Complementary Output mode
- Self-Triggered mode

The current limit PCI block logic is used to control the cycle truncation. The Leading-Edge Blanking feature is used to filter out switching transients and slightly delay cycle truncation, as shown with the upper arrows in Figure 15-46. The duty cycle is truncated when the PCI active signal goes high.

To reset the PCI block for the next cycle, the PCI terminator is configured to detect the falling edge of the comparator (CMP1 Out). The terminator signal is then synchronized to the End-of-Cycle (EOC), and the PCI active signal is reset, as shown with the lower arrows in Figure 15-46.

Figure 15-46. Timing Diagram for Self-Triggered, Complementary Output and Current Limit Cycle-by-Cycle PWM Modes



Example 15-6. Cycle-by-Cycle Current Limit Mode

```
void PWMInitialization(void);
void enable_CMP1();

int main() {

    PWMInitialization();

    //The CMP1A input will be compared against the DAC1 output to create the CMP1 out signal.
    //If CMP1A > DAC output, PWM1 output will be overridden
    //If CMP1A < DAC output, PWM1 output will be active

    while(1);

    return 0;
}

void PWMInitialization(void) {

    configure_PLL2_Fout_200MHz_and_VCODIV_500_MHz();
    clock_PWM_from_PLL2_Fout();
    initialize_CMP1_and_clock_from_PLL2_VCODIV();

    PG1CONbits.CLKSEL = 1;  //PWM generator 1 uses the PWM Host clock, undivided & unscaled
    PG1CONbits.MODSEL = 0b000;  //PWM generator 1 uses independent edge PWM mode
    PG1CONbits.TRGMOD = 0b00;   //PWM generator 1 uses single trigger mode
    PG1CONbits.UPDMOD = 0b000;  //Update data registers at SOC

    //PWM Generator 1 uses PG1DC, PG1PER, PG1PHASE registers
    PG1CONbits.MDCSEL = 0;
```

```
    PG1CONbits.MPERSEL = 0;
    PG1CONbits.MPHSEL = 0;

    PG1CONbits.MSTEN = 0;  //PWM Generator does not broadcast UPDATE status bit state or EOC signal
    PG1CONbits.SOCS = 0b0000;  //Start of cycle (SOC) = local EOC

    PG1IOCON1bits.PMOD = 0b00;   //PWM Generator 1 outputs operate in Complementary mode

    //PWM Generator 1 controls the PWM1H and PWM1L output pins
    PG1IOCON1bits.PENH = 1;
    PG1IOCON1bits.PENL = 1;
    //PWM1H and PWM1L output pins are active high
    PG1IOCON1bits.POLH = 0;
    PG1IOCON1bits.POLL = 0;
    //Current limit data: 1 on PWM1L and 0 on PWM1H
    PG1IOCON2bits.CLDAT = 0b01;


    //Given the 200MHz PWM clock, this period will result in a PWM frequency of 100kHz
    PG1PER = (2000 << 4);        //Time units are 1/16 of a PWM clock
    PG1DC = (1000 << 4);         //50% duty cycle
    PG1PHASE = (200 << 4);       //200 PWM clocks of phase offset in rising edge of PWM
    PG1DTbits.DTH = (40 << 4);   //40 PWM clocks of dead time on PWM1H
    PG1DTbits.DTL = (40 << 4);   //40 PWM clocks of dead time on PWM1L
    PG1LEBbits.PHR = 1;          //Rising edge of PWM1H will trigger the LEB counter
    PG1LEBbits.LEB = (100 << 4); //100 PWM clocks of LEB

    //PCI logic configuration for current limit cycle by cycle mode, comparator 1 output as PCI source
    PG1CLPCI1bits.TERM = 0b001;   //Terminate when PCI source transitions from active to inactive
    PG1CLPCI1bits.TSYNCDIS = 0;   //Termination of latched PCI delays till PWM EOC (for Cycle by cycle
mode)
    PG1CLPCI1bits.AQSS = 0b010;   //LEB active is selected as acceptance qualifier
    PG1CLPCI1bits.AQPS = 1;       //LEB active is inverted to accept PCI signal when LEB duration is over
    PG1CLPCI1bits.PSYNC = 0;      //PCI source is not synchronized to PWM EOC so that current limit
resets PWM immediately
    PG1CLPCI2 = 0x10000000;       //Comparator 1 output is selected as PCI source signal
    PG1CLPCI1bits.PPS = 0;        //PCI source signal is not inverted
    PG1CLPCI1bits.ACP = 0b011;    //latched PCI is selected as acceptance criteria to work when CMP1 out
is active
    PG1CLPCI1bits.TQSS = 0b0000;  //No termination qualifier used so terminator will work straight away
without any qualifier

    //Enable PWM generator 1
    PG1CONbits.ON = 1;
}
```

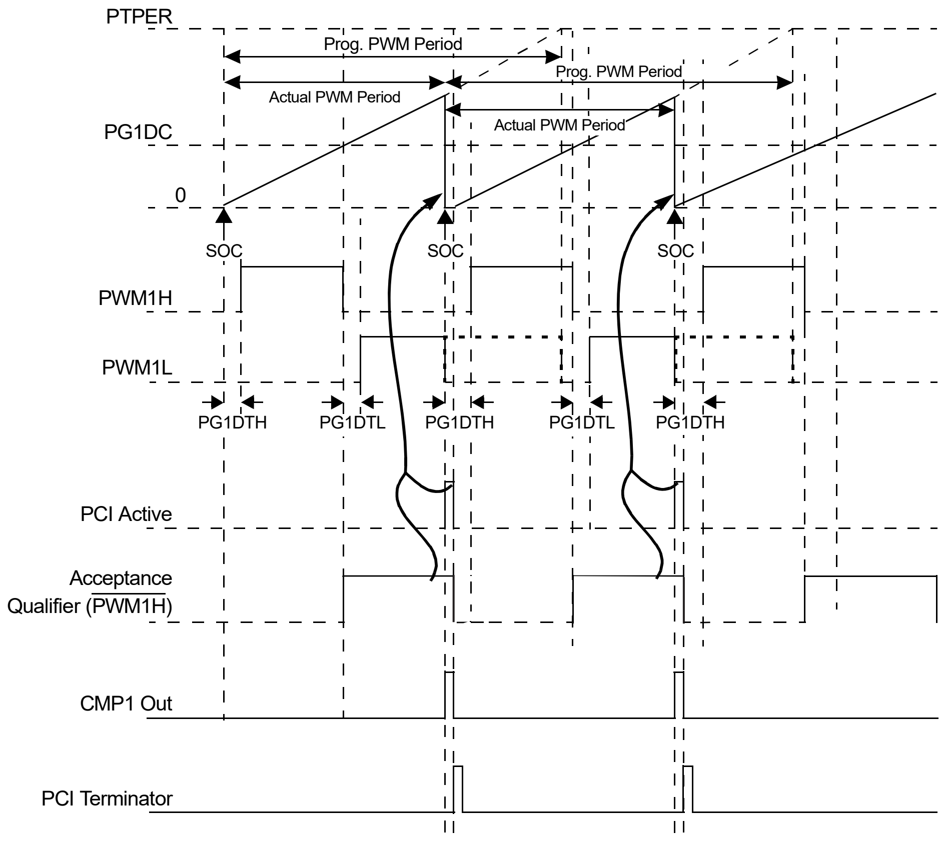
15.8.5. External Period Reset Mode

External Period Reset mode for power control monitors the inductor current and varies the PWM period to control power delivery. When the inductor current returns to '0', the PWM cycle will restart. The PWM is configured as follows:

- Independent Edge PWM mode
- Complementary Output mode
- Self-Triggered mode

The initial programmed PWM period may be shortened depending on the inductor current compared to a predetermined trip point. The Sync PCI block is used to control cycle truncation. The comparator (CMP1 Out) output is used as the input to the Sync PCI block, and an inverted PWM1H signal is used as a qualifier to allow truncation only on the duty cycle inactive time of a cycle, as shown by the arrows in Figure 15-47. The PCI block is reset for the next cycle using the auto-terminate function.

Figure 15-47. Timing Diagram for Self-Triggered, Complementary Output and Current Reset PWM Modes



Example 15-7. External Period Reset Mode

```
void PWMInitialization(void);

int main()
{
    PWMInitialization();
    while (1);
    return 0;
}

void PWMInitialization(void)
{
    configure_PLL2_Fout_200MHz_and_VCODIV_500_MHz();
    clock_PWM_from_PLL2_Fout();
    initialize_CMP1_and_clock_from_PLL2_VCODIV();

    //PWM generator 1 uses PG1DC, PG1PER, PG1PHASE registers
    PG1CONbits.MDCSEL = 0;
    PG1CONbits.MPERSEL = 0;
    PG1CONbits.MPHSEL = 0;
    PG1CONbits.MSTEN = 0;          //PWM Generator does not broadcast UPDATE status bit
state or EOC signal
    PG1CONbits.TRGMOD = 0b01;      //PWM Generator operates in Re-Triggerable mode
    PG1CONbits.SOCS = 0b0000;      //Start of cycle (SOC) = local EOC is OR'd with PCI sync
    PG1CONbits.UPDMOD = 0b000;     //Update the data registers at start of next PWM
cycle(SOC)
    PG1CONbits.MODSEL = 0b000;     //Independent edge triggered mode
    PG1CONbits.CLKSEL = 1;         //PWM Generator 1 uses PWM Master Clock, undivided and
unscaled
    PG1IOCON1bits.PMOD = 0b00;     //PWM Generator outputs operate in Complementary mode

    //PWM Generator controls the PWM1H and PWM1L output pins
    PG1IOCON1bits.PENH = 1;
    PG1IOCON1bits.PENL = 1;

    //PWM1H and PWM1L output pins are active high
    PG1IOCON1bits.POLH = 0;
```

```
    PG1IOCON1bits.POLL = 0;

    //For a 200MHz PWM clock, this will result in 100kHz PWM frequency
    PG1PER = (2000 << 4);        //Time units are 1/16 of a PWM clock
    PG1DC = (500 << 4); //25% duty cycle
    PG1PHASE = 0; //No Phase offset in rising edge of PWM
    PG1DTbits.DTH = (40 << 4);  //40 PWM clocks of dead time on PWM1H
    PG1DTbits.DTL = (40 << 4);  //40 PWM clocks of dead time on PWM1L
    PG1LEBbits.PHR = 1; //Rising edge of PWM1H will trigger the LEB counter
    PG1LEBbits.LEB = (80 << 4); //80 PWM clocks of LEB

    //PCI logic configuration for current reset (PCI sync mode),
    //comparator 1 output (current reset signal) as PCI source,
    //and PWM1H falling edge as acceptance qualifier
    PG1EVT1bits.PWMPCI = 0b000; //PWM Generator #1 output used as PCI signal
    PG1SPCI1bits.TERM = 0b001; //Terminate when PCI source transitions from active to
inactive
    PG1SPCI1bits.TSYNCDIS = 1; //Termination of latched PCI occurs immediately
    PG1SPCI1bits.AQSS = 0b100; //Inverted PWM1H is selected as acceptance qualifier
because PWM should be reset in OFF time
    PG1SPCI1bits.AQPS = 1; //Acceptance qualifier inverted to accept PCI signal when PWM1H
on time is over
    PG1SPCI1bits.PSYNC = 0; //PCI source is not synchronized to PWM EOC so that current
limit resets PWM immediately
    PG1SPCI2 = 0b11011; //CMP1 out is selected as PCI source signal
    PG1SPCI1bits.PPS = 1; //PCI source signal is inverted
    PG1SPCI1bits.ACP = 0b011; //Latched PCI is selected as acceptance criteria to work
when CMP1 out is active
    PG1SPCI1bits.TQSS = 0b000; //No termination qualifier used so terminator will work
straight away without any qualifier

    //Enable PWM generator 1
    PG1CONbits.ON = 1;
}
```

16. 40 MSPS Analog-to-Digital Converter (ADC)

dsPIC33AK256MPS306 devices have three high-speed, 12-bit Analog-to-Digital Converters (ADC) that feature low conversion latency, high resolution and oversampling capabilities to improve performance in AC/DC and DC/DC power converters.

Each 40 MSPS Analog-to-Digital Converter (ADC) includes the following features:

- 12-bit Resolution
- Up to 40 MSPS Conversion Rate per Channel
- Up to 14 Analog Input Pins
- Each Conversion Channel:
  - Can be assigned to any analog input (I/O pin or internal signal)
  - Can be set to a different sampling time
  - Supports discrete configuration
  - Can be configured as single-ended or differential
  - Conversion result can be formatted as unsigned or signed
  - Conversion result can be left-aligned (fraction format)
  - Has a separate 32-bit conversion result register
- All Channels Support Four Sampling Modes:
  - Oversampling of multiple samples
  - Integration of multiple samples
  - Window (multiple samples accumulated when the gate signal is active).
  - Single conversion
  - 16-bit sampling capable
- All channels have a digital comparator to detect when the conversion result is less than, greater than, in bounds or out of bounds for the configurable thresholds.
- Some channels have the second result accumulator, which can be used for a filter implementation.
- Band Gap Reference and Temperature Sensor Diode inputs via UREF outputs
- Operation during CPU Sleep and Idle Modes

16.1. Device-Specific Information

Table 16-1. ADC Summary Table

Number of Cores	Max Number of Channels	Max Input Clock	Clock Source	Peripheral Bus Speed
3	12	320 MHz	CLKGEN6	Fast (1:1 CPU Clock)

The number of available positive and negative analog inputs is dependent on package size, as shown in the table below.

Table 16-2. ADC Input Availability

ADC Input	32-Pin	48-Pin	64-Pin	Comments
AD1ANN0	AV_{SS}			ADC 1 ground negative input 0 supporting differential mode
AD1ANN1		x	x	ADC 1 negative input 1 supporting differential mode

Table 16-2. ADC Input Availability (continued)

ADC Input	32-Pin	48-Pin	64-Pin	Comments
AD1ANN2	x	x	x	ADC 1 negative input 2 supporting differential mode
AD1ANN3	Reserved			Reserved
AD1AN0	x	x	x	ADC 1 positive input 0
AD1AN1	x	x	x	ADC 1 positive input 1
AD1AN2	x	x	x	ADC 1 positive input 2
AD1AN3		x	x	ADC 1 positive input 3
AD1AN4	x	x	x	ADC 1 positive input 4
AD1AN5	Reserved			Reserved
AD1AN6	Internal			ADC 1 0.8V Bandgap
AD1AN7	Internal			ADC 1 UREF input
AD1AN8	Internal			ADC 1 15/16*AV_{DD} or 1/16*AV_{DD} reference input
AD2ANN0	AV_{SS}			ADC 2 ground negative input 0 supporting differential mode
AD2ANN1	x	x	x	ADC 2 negative input 1 supporting differential mode
AD2ANN2	x	x	x	ADC 2 negative input 2 supporting differential mode
AD2ANN3	Reserved			Reserved
AD2AN0	x	x	x	ADC 2 positive input 0
AD2AN1	x	x	x	ADC 2 positive input 1
AD2AN2	x	x	x	ADC 2 positive input 2
AD2AN3	x	x	x	ADC 2 positive input 3
AD2AN4	x	x	x	ADC 2 positive input 4
AD2AN5	x	x	x	ADC 2 positive input 5
AD2AN6	Internal			ADC 2 0.8V Bandgap
AD2AN7	Internal			ADC 2 UREF input
AD2AN8	Internal			ADC 2 15/16*AV_{DD} or 1/16*AV_{DD} reference input
AD3ANN0	AV_{SS}			ADC 3 ground negative input 0 supporting differential mode
AD3ANN1	x	x	x	ADC 3 negative input 1 supporting differential mode
AD3ANN2	x	x	x	ADC 3 negative input 2 supporting differential mode
AD3ANN3	Reserved			Reserved
AD3AN0	x	x	x	ADC 3 positive input 0
AD3AN1	x		x	ADC 3 positive input 1
AD3AN2	x	x	x	ADC 3 positive input 2
AD3AN3		x	x	ADC 3 positive input 3
AD3AN4	x	x	x	ADC 3 positive input 4
AD3AN5	Reserved			Reserved
AD3AN6	Internal			ADC 3 0.8V Bandgap
AD3AN7	Internal			ADC 3 UREF input
AD3AN8	Internal			ADC 2 15/16*AV_{DD} or 1/16*AV_{DD} reference input
AD3AN9	Internal			ADC 3 V_{DDCORE} input

Table 16-3. TRGnSRC Trigger Source Selection Bits

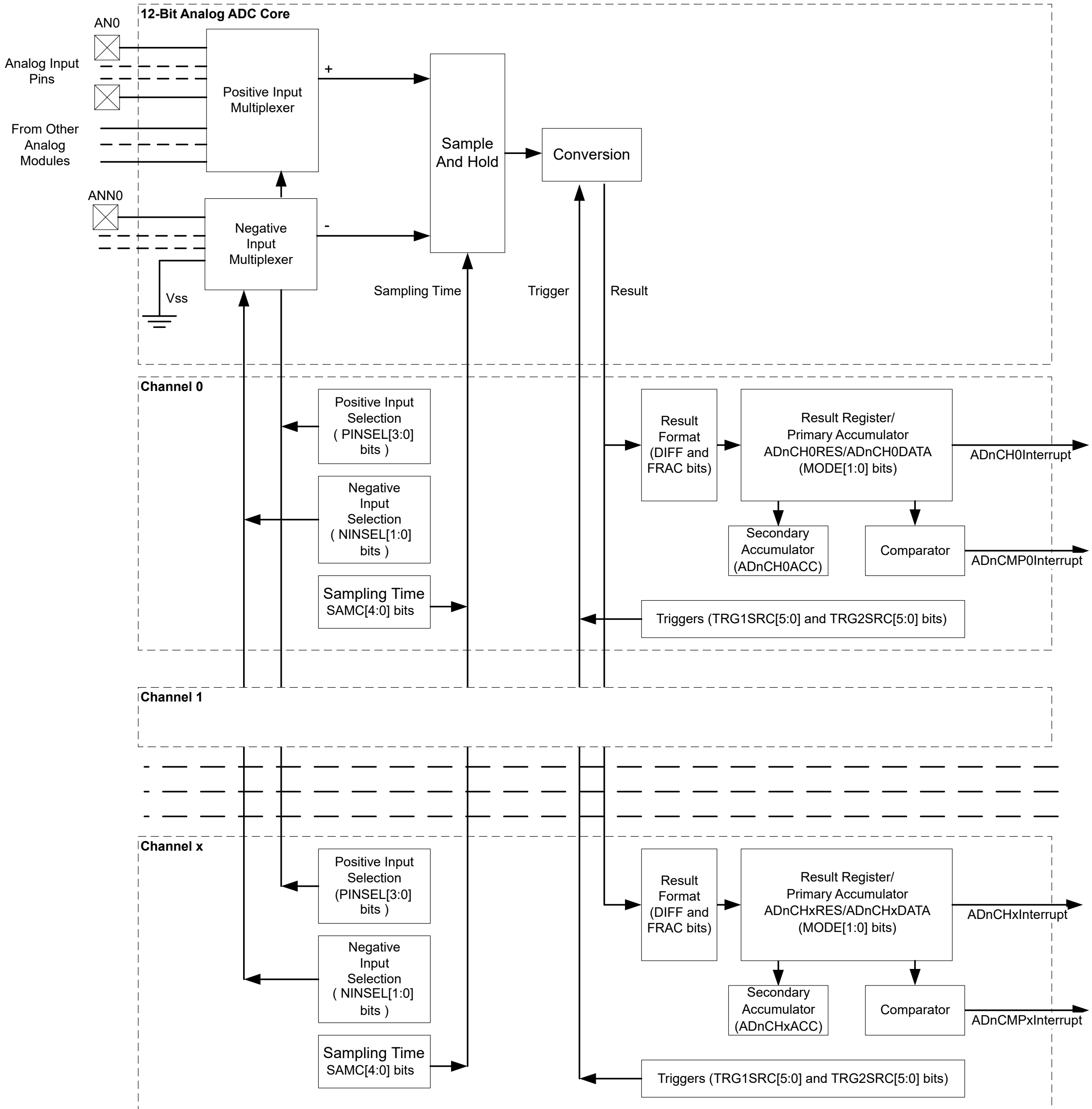
Value	Description
11000	QEI1
10111	RDC
10110	SCCP4
10101	SCCP3
10100	SCCP2
10011	SCCP1
10010	PTG
10001	SCCP5 (MCCP)
10000	CLC4
1111	CLC3
1110	CLC2
1101	CLC1
1100	ITC
1011	PWMGEN 4
1010	PWMGEN 4
1001	PWMGEN 3
1000	PWMGEN 3
111	PWMGEN 2
110	PWMGEN 2
101	PWMGEN 1
100	PWMGEN 1
11	Conversion repeat timer trigger defined by RPTCNT[5:0] (ADnCON[23:18]) bits
10	Immediate re-trigger request
01	Software trigger initiated by using the ADnSWTRG register
0	Triggers are disabled

16.2. ADC Architectural Overview

The module consists of multiple channels using one ADC analog core. The analog inputs (external I/O pins or internal signals) are selected through multiplexers and switches by the channel settings. The ADC core uses the channel information (the analog positive/negative input numbers and sampling time) to process the analog sample. When conversion is complete, the result is added to the result buffer of the channel depending on the channel's Sampling mode (Single Conversion or Accumulation) and then sent to a digital comparator, if it is enabled.

A simplified block diagram of the 40 MSPS 12-Bit ADC is illustrated in Figure 16-1.

Figure 16-1. 40 MSPS 12-Bit ADC Block Diagram



16.3. Register Summary

Legend: n = ADC number; x = Channel number

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0800	AD1CON	31:24	ADRDY	CALRDY	CALREQ	ACALEN	CALRATE[1:0]		MODE[1:0]	
		23:16	RPTCNT[5:0]						Reserved	STNDBY
		15:8	ON					TSTLOCK		TSTEN
		7:0	BUFEN	CALCNT[1:0]						
0x0804	AD1DATAOVR	31:24	DATAOVR[31:24]							
		23:16	DATAOVR[23:16]							
		15:8	DATAOVR[15:8]							
		7:0	DATAOVR[7:0]							
0x0808	AD1STAT	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]RDY							
0x080C	AD1RSTAT	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]RRDY							
0x0810	AD1CMPSTAT	31:24								
		23:16								
		15:8								
		7:0	CMP[7:0]FLG							
0x0814	AD1SWTRG	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]TRG							
0x0818	AD1CH0CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x081C	AD1CH0CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPCNTMOD			ADCMPSTAT[9:8]	
		23:16	ADCMPSTAT[7:0]							
		15:8		CMPMOD[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0820	AD1CH0DATA	31:24	DATA[31:24]							
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x0824	AD1CH0RES	31:24	RESF[11:4]							
		23:16	RESF[3:0]							
		15:8					RES[11:8]			
		7:0	RES[7:0]							
0x0828	AD1CH0CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x082C	AD1CH0CMPLO	31:24	CMPLO[31:24]							
		23:16	CMPLO[23:16]							
		15:8	CMPLO[15:8]							
		7:0	CMPLO[7:0]							
0x0830	AD1CH0CMPHI	31:24	CMPHI[31:24]							
		23:16	CMPHI[23:16]							
		15:8	CMPHI[15:8]							
		7:0	CMPHI[7:0]							
0x0834	AD1CH0ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0838	AD1CH1CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x083C	ADnCH1CON21	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0840	AD1CH1DATA	31:24	DATA1[31:24]							
		23:16	DATA1[23:16]							
		15:8	DATA1[15:8]							
		7:0	DATA1[7:0]							
0x0844	AD1CH1RES	31:24	RESF1[11:4]							
		23:16	RESF1[3:0]							
		15:8					RES1[11:8]			
		7:0	RES1[7:0]							
0x0848	AD1CH1CNT	31:24	CNTSTAT1[15:8]							
		23:16	CNTSTAT1[7:0]							
		15:8	CNT1[15:8]							
		7:0	CNT1[7:0]							
0x084C	AD1CH1CMPLO	31:24	LO1[31:24]							
		23:16	LO1[23:16]							
		15:8	LO1[15:8]							
		7:0	LO1[7:0]							
0x0850	AD1CH1CMPHI	31:24	HI1[31:24]							
		23:16	HI1[23:16]							
		15:8	HI1[15:8]							
		7:0	HI1[7:0]							
0x0854	AD1CH1ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0858	AD1CH2CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x085C	AD1CH2CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0860	AD1CH2DATA	31:24	DATA2[31:24]							
		23:16	DATA2[23:16]							
		15:8	DATA2[15:8]							
		7:0	DATA2[7:0]							
0x0864	AD1CH2RES	31:24	RESF2[11:4]							
		23:16	RESF2[3:0]							
		15:8					RES2[11:8]			
		7:0	RES2[7:0]							
0x0868	AD1CH2CNT	31:24	CNTSTAT2[15:8]							
		23:16	CNTSTAT2[7:0]							
		15:8	CNT2[15:8]							
		7:0	CNT2[7:0]							
0x086C	AD1CH2CMPLO	31:24	LO2[31:24]							
		23:16	LO2[23:16]							
		15:8	LO2[15:8]							
		7:0	LO2[7:0]							
0x0870	AD1CH2CMPHI	31:24	HI2[31:24]							
		23:16	HI2[23:16]							
		15:8	HI2[15:8]							
		7:0	HI2[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0874	AD1CH2ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0878	AD1CH3CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x087C	AD1CH3CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0880	AD1CH3DATA	31:24	DATA3[31:24]							
		23:16	DATA3[23:16]							
		15:8	DATA3[15:8]							
		7:0	DATA3[7:0]							
0x0884	AD1CH3RES	31:24	RESF3[11:4]							
		23:16	RESF3[3:0]				RES3[19:16]			
		15:8	RES3[15:8]							
		7:0	RES3[7:0]							
0x0888	AD1CH3CNT	31:24	CNTSTAT3[15:8]							
		23:16	CNTSTAT3[7:0]							
		15:8	CNT3[15:8]							
		7:0	CNT3[7:0]							
0x088C	AD1CH3CMPLO	31:24	LO3[31:24]							
		23:16	LO3[23:16]							
		15:8	LO3[15:8]							
		7:0	LO3[7:0]							
0x0890	AD1CH3CMPHI	31:24	HI3[31:24]							
		23:16	HI3[23:16]							
		15:8	HI3[15:8]							
		7:0	HI3[7:0]							
0x0894	AD1CH3ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0898	AD1CH4CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x089C	AD1CH4CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x08A0	AD1CH4DATA	31:24	DATA4[31:24]							
		23:16	DATA4[23:16]							
		15:8	DATA4[15:8]							
		7:0	DATA4[7:0]							
0x08A4	AD1CH4RES	31:24	RESF4[11:4]							
		23:16	RESF4[3:0]							
		15:8					RES4[11:8]			
		7:0	RES4[7:0]							
0x08A8	AD1CH4CNT	31:24	CNTSTAT4[15:8]							
		23:16	CNTSTAT4[7:0]							
		15:8	CNT4[15:8]							
		7:0	CNT4[7:0]							
0x08AC	AD1CH4CMPLO	31:24	LO4[31:24]							
		23:16	LO4[23:16]							
		15:8	LO4[15:8]							
		7:0	LO4[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x08B0	AD1CH4CMPHI	31:24	HI4[31:24]							
		23:16	HI4[23:16]							
		15:8	HI4[15:8]							
		7:0	HI4[7:0]							
0x08B4	AD1CH4ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x08B8	AD1CH5CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x08BC	AD1CH5CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x08C0	AD1CH5DATA	31:24	DATA5[31:24]							
		23:16	DATA5[23:16]							
		15:8	DATA5[15:8]							
		7:0	DATA5[7:0]							
0x08C4	AD1CH5RES	31:24	RESF5[19:12]							
		23:16	RESF5[11:4]							
		15:8	RESF5[3:0]				RES5[11:8]			
		7:0	RES5[7:0]							
0x08C8	AD1CH5CNT	31:24	CNTSTAT5[15:8]							
		23:16	CNTSTAT5[7:0]							
		15:8	CNT5[15:8]							
		7:0	CNT5[7:0]							
0x08CC	AD1CH5CMPLO	31:24	LO5[31:24]							
		23:16	LO5[23:16]							
		15:8	LO5[15:8]							
		7:0	LO5[7:0]							
0x08D0	AD1CH5CMPHI	31:24	HI5[31:24]							
		23:16	HI5[23:16]							
		15:8	HI5[15:8]							
		7:0	HI5[7:0]							
0x08D4	AD1CH5ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x08D8	AD1CH6CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x08DC	AD1CH6CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x08E0	AD1CH6DATA	31:24	DATA6[31:24]							
		23:16	DATA6[23:16]							
		15:8	DATA6[15:8]							
		7:0	DATA6[7:0]							
0x08E4	AD1CH6RES	31:24	RESF6[11:4]							
		23:16	RESF6[3:0]							
		15:8					RES6[11:8]			
		7:0	RES6[7:0]							
0x08E8	AD1CH6CNT	31:24	CNTSTAT6[15:8]							
		23:16	CNTSTAT6[7:0]							
		15:8	CNT6[15:8]							
		7:0	CNT6[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x08EC	AD1CH6CMPLO	31:24	LO6[31:24]							
		23:16	LO6[23:16]							
		15:8	LO6[15:8]							
		7:0	LO6[7:0]							
0x08F0	AD1CH6CMPHI	31:24	HI6[31:24]							
		23:16	HI6[23:16]							
		15:8	HI6[15:8]							
		7:0	HI6[7:0]							
0x08F4	AD1CH6ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x08F8 ... 0x08FF	Reserved									
0x0900	AD2CON	31:24	ADRDY	CALRDY	CALREQ	ACALEN	CALRATE[1:0]		MODE[1:0]	
		23:16	RPTCNT[5:0]						Reserved	STNDBY
		15:8	ON					TSTLOCK		TSTEN
		7:0	BUFEN	CALCNT[1:0]						
0x0904	AD2DATAOVR	31:24	DATAOVR[31:24]							
		23:16	DATAOVR[23:16]							
		15:8	DATAOVR[15:8]							
		7:0	DATAOVR[7:0]							
0x0908	AD2STAT	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]RDY							
0x090C	AD2RSTAT	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]RRDY							
0x0910	AD2CMPSTAT	31:24								
		23:16								
		15:8								
		7:0	CMP[7:0]FLG							
0x0914	AD2SWTRG	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]TRG							
0x0918	AD2CH0CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x091C	AD2CH0CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPCNTMOD			ADCMPSTAT[9:8]	
		23:16	ADCMPSTAT[7:0]							
		15:8		CMPMOD[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0920	AD2CH0DATA	31:24	DATA[31:24]							
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x0924	AD2CH0RES	31:24	RESF[11:4]							
		23:16	RESF[3:0]							
		15:8					RES[11:8]			
		7:0	RES[7:0]							
0x0928	AD2CH0CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x092C	AD2CH0CMPLO	31:24	CMPLO[31:24]							
		23:16	CMPLO[23:16]							
		15:8	CMPLO[15:8]							
		7:0	CMPLO[7:0]							
0x0930	AD2CH0CMPHI	31:24	CMPHI[31:24]							
		23:16	CMPHI[23:16]							
		15:8	CMPHI[15:8]							
		7:0	CMPHI[7:0]							
0x0934	AD2CH0ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0938	AD2CH1CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x093C	ADnCH1CON22	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0940	AD2CH1DATA	31:24	DATA1[31:24]							
		23:16	DATA1[23:16]							
		15:8	DATA1[15:8]							
		7:0	DATA1[7:0]							
0x0944	AD2CH1RES	31:24	RESF1[11:4]							
		23:16	RESF1[3:0]							
		15:8					RES1[11:8]			
		7:0	RES1[7:0]							
0x0948	AD2CH1CNT	31:24	CNTSTAT1[15:8]							
		23:16	CNTSTAT1[7:0]							
		15:8	CNT1[15:8]							
		7:0	CNT1[7:0]							
0x094C	AD2CH1CMPLO	31:24	LO1[31:24]							
		23:16	LO1[23:16]							
		15:8	LO1[15:8]							
		7:0	LO1[7:0]							
0x0950	AD2CH1CMPHI	31:24	HI1[31:24]							
		23:16	HI1[23:16]							
		15:8	HI1[15:8]							
		7:0	HI1[7:0]							
0x0954	AD2CH1ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0958	AD2CH2CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x095C	AD2CH2CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0960	AD2CH2DATA	31:24	DATA2[31:24]							
		23:16	DATA2[23:16]							
		15:8	DATA2[15:8]							
		7:0	DATA2[7:0]							
0x0964	AD2CH2RES	31:24	RESF2[11:4]							
		23:16	RESF2[3:0]							
		15:8					RES2[11:8]			
		7:0	RES2[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0968	AD2CH2CNT	31:24	CNTSTAT2[15:8]							
		23:16	CNTSTAT2[7:0]							
		15:8	CNT2[15:8]							
		7:0	CNT2[7:0]							
0x096C	AD2CH2CMPLO	31:24	LO2[31:24]							
		23:16	LO2[23:16]							
		15:8	LO2[15:8]							
		7:0	LO2[7:0]							
0x0970	AD2CH2CMPHI	31:24	HI2[31:24]							
		23:16	HI2[23:16]							
		15:8	HI2[15:8]							
		7:0	HI2[7:0]							
0x0974	AD2CH2ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0978	AD2CH3CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x097C	AD2CH3CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0980	AD2CH3DATA	31:24	DATA3[31:24]							
		23:16	DATA3[23:16]							
		15:8	DATA3[15:8]							
		7:0	DATA3[7:0]							
0x0984	AD2CH3RES	31:24	RESF3[11:4]							
		23:16	RESF3[3:0]				RES3[19:16]			
		15:8	RES3[15:8]							
		7:0	RES3[7:0]							
0x0988	AD2CH3CNT	31:24	CNTSTAT3[15:8]							
		23:16	CNTSTAT3[7:0]							
		15:8	CNT3[15:8]							
		7:0	CNT3[7:0]							
0x098C	AD2CH3CMPLO	31:24	LO3[31:24]							
		23:16	LO3[23:16]							
		15:8	LO3[15:8]							
		7:0	LO3[7:0]							
0x0990	AD2CH3CMPHI	31:24	HI3[31:24]							
		23:16	HI3[23:16]							
		15:8	HI3[15:8]							
		7:0	HI3[7:0]							
0x0994	AD2CH3ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0998	AD2CH4CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x099C	AD2CH4CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x09A0	AD2CH4DATA	31:24	DATA4[31:24]							
		23:16	DATA4[23:16]							
		15:8	DATA4[15:8]							
		7:0	DATA4[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x09A4	AD2CH4RES	31:24	RESF4[11:4]							
		23:16	RESF4[3:0]							
		15:8					RES4[11:8]			
		7:0	RES4[7:0]							
0x09A8	AD2CH4CNT	31:24	CNTSTAT4[15:8]							
		23:16	CNTSTAT4[7:0]							
		15:8	CNT4[15:8]							
		7:0	CNT4[7:0]							
0x09AC	AD2CH4CMPLO	31:24	LO4[31:24]							
		23:16	LO4[23:16]							
		15:8	LO4[15:8]							
		7:0	LO4[7:0]							
0x09B0	AD2CH4CMPHI	31:24	HI4[31:24]							
		23:16	HI4[23:16]							
		15:8	HI4[15:8]							
		7:0	HI4[7:0]							
0x09B4	AD2CH4ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x09B8	AD2CH5CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x09BC	AD2CH5CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x09C0	AD2CH5DATA	31:24	DATA5[31:24]							
		23:16	DATA5[23:16]							
		15:8	DATA5[15:8]							
		7:0	DATA5[7:0]							
0x09C4	AD2CH5RES	31:24	RESF5[19:12]							
		23:16	RESF5[11:4]							
		15:8	RESF5[3:0]				RES5[11:8]			
		7:0	RES5[7:0]							
0x09C8	AD2CH5CNT	31:24	CNTSTAT5[15:8]							
		23:16	CNTSTAT5[7:0]							
		15:8	CNT5[15:8]							
		7:0	CNT5[7:0]							
0x09CC	AD2CH5CMPLO	31:24	LO5[31:24]							
		23:16	LO5[23:16]							
		15:8	LO5[15:8]							
		7:0	LO5[7:0]							
0x09D0	AD2CH5CMPHI	31:24	HI5[31:24]							
		23:16	HI5[23:16]							
		15:8	HI5[15:8]							
		7:0	HI5[7:0]							
0x09D4	AD2CH5ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x09D8	AD2CH6CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x09DC	AD2CH6CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x09E0	AD2CH6DATA	31:24	DATA6[31:24]							
		23:16	DATA6[23:16]							
		15:8	DATA6[15:8]							
		7:0	DATA6[7:0]							
0x09E4	AD2CH6RES	31:24	RESF6[11:4]							
		23:16	RESF6[3:0]							
		15:8					RES6[11:8]			
		7:0	RES6[7:0]							
0x09E8	AD2CH6CNT	31:24	CNTSTAT6[15:8]							
		23:16	CNTSTAT6[7:0]							
		15:8	CNT6[15:8]							
		7:0	CNT6[7:0]							
0x09EC	AD2CH6CMPLO	31:24	LO6[31:24]							
		23:16	LO6[23:16]							
		15:8	LO6[15:8]							
		7:0	LO6[7:0]							
0x09F0	AD2CH6CMPHI	31:24	HI6[31:24]							
		23:16	HI6[23:16]							
		15:8	HI6[15:8]							
		7:0	HI6[7:0]							
0x09F4	AD2CH6ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x09F8	AD2CH7CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x09FC	AD2CH7CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0A00	AD2CH7DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							
0x0A04	AD2CH7RES	31:24	RESF7[11:4]							
		23:16	RESF7[3:0]							
		15:8					RES7[11:8]			
		7:0	RES7[7:0]							
0x0A08	AD2CH7CNT	31:24	CNTSTAT7[15:8]							
		23:16	CNTSTAT7[7:0]							
		15:8	CNT7[15:8]							
		7:0	CNT7[7:0]							
0x0A0C	AD2CH7CMPLO	31:24	LO7[31:24]							
		23:16	LO7[23:16]							
		15:8	LO7[15:8]							
		7:0	LO7[7:0]							
0x0A10	AD2CH7CMPHI	31:24	HI7[31:24]							
		23:16	HI7[23:16]							
		15:8	HI7[15:8]							
		7:0	HI7[7:0]							
0x0A14	AD2CH7ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0A14	AD2CH7ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0A18 ... 0x0A7F	Reserved									
0x0A80	AD3CON	31:24	ADRDY	CALRDY	CALREQ	ACALEN	CALRATE[1:0]		MODE[1:0]	
		23:16	RPTCNT[5:0]						Reserved	STNDBY
		15:8	ON					TSTLOCK		TSTEN
		7:0	BUFEN	CALCNT[1:0]						
0x0A84	AD3DATAOVR	31:24	DATAOVR[31:24]							
		23:16	DATAOVR[23:16]							
		15:8	DATAOVR[15:8]							
		7:0	DATAOVR[7:0]							
0x0A88	AD3STAT	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]RDY							
0x0A8C	AD3RSTAT	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]RRDY							
0x0A90	AD3CMPSTAT	31:24								
		23:16								
		15:8								
		7:0	CMP[7:0]FLG							
0x0A94	AD3SWTRG	31:24								
		23:16								
		15:8								
		7:0	CH[7:0]TRG							
0x0A98	AD3CH0CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0A9C	AD3CH0CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPCNTMOD			ADCMPSTAT[9:8]	
		23:16	ADCMPSTAT[7:0]							
		15:8		CMPMOD[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0AA0	AD3CH0DATA	31:24	DATA[31:24]							
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x0AA4	AD3CH0RES	31:24	RESF[11:4]							
		23:16	RESF[3:0]							
		15:8					RES[11:8]			
		7:0	RES[7:0]							
0x0AA8	AD3CH0CNT	31:24	CNTSTAT[15:8]							
		23:16	CNTSTAT[7:0]							
		15:8	CNT[15:8]							
		7:0	CNT[7:0]							
0x0AAC	AD3CH0CMPLO	31:24	CMPLO[31:24]							
		23:16	CMPLO[23:16]							
		15:8	CMPLO[15:8]							
		7:0	CMPLO[7:0]							
0x0AB0	AD3CH0CMPHI	31:24	CMPHI[31:24]							
		23:16	CMPHI[23:16]							
		15:8	CMPHI[15:8]							
		7:0	CMPHI[7:0]							
0x0AB4	AD3CH0ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0AB8	AD3CH1CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0ABC	ADnCH1CON23	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0AC0	AD3CH1DATA	31:24	DATA1[31:24]							
		23:16	DATA1[23:16]							
		15:8	DATA1[15:8]							
		7:0	DATA1[7:0]							
0x0AC4	AD3CH1RES	31:24	RESF1[11:4]							
		23:16	RESF1[3:0]							
		15:8					RES1[11:8]			
		7:0	RES1[7:0]							
0x0AC8	AD3CH1CNT	31:24	CNTSTAT1[15:8]							
		23:16	CNTSTAT1[7:0]							
		15:8	CNT1[15:8]							
		7:0	CNT1[7:0]							
0x0ACC	AD3CH1CMPLO	31:24	LO1[31:24]							
		23:16	LO1[23:16]							
		15:8	LO1[15:8]							
		7:0	LO1[7:0]							
0x0AD0	AD3CH1CMPHI	31:24	HI1[31:24]							
		23:16	HI1[23:16]							
		15:8	HI1[15:8]							
		7:0	HI1[7:0]							
0x0AD4	AD3CH1ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0AD8	AD3CH2CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0ADC	AD3CH2CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0AE0	AD3CH2DATA	31:24	DATA2[31:24]							
		23:16	DATA2[23:16]							
		15:8	DATA2[15:8]							
		7:0	DATA2[7:0]							
0x0AE4	AD3CH2RES	31:24	RESF2[11:4]							
		23:16	RESF2[3:0]							
		15:8					RES2[11:8]			
		7:0	RES2[7:0]							
0x0AE8	AD3CH2CNT	31:24	CNTSTAT2[15:8]							
		23:16	CNTSTAT2[7:0]							
		15:8	CNT2[15:8]							
		7:0	CNT2[7:0]							
0x0AEC	AD3CH2CMPLO	31:24	LO2[31:24]							
		23:16	LO2[23:16]							
		15:8	LO2[15:8]							
		7:0	LO2[7:0]							
0x0AF0	AD3CH2CMPHI	31:24	HI2[31:24]							
		23:16	HI2[23:16]							
		15:8	HI2[15:8]							
		7:0	HI2[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0AF4	AD3CH2ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0AF8	AD3CH3CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0AFC	AD3CH3CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0B00	AD3CH3DATA	31:24	DATA3[31:24]							
		23:16	DATA3[23:16]							
		15:8	DATA3[15:8]							
		7:0	DATA3[7:0]							
0x0B04	AD3CH3RES	31:24	RESF3[11:4]							
		23:16	RESF3[3:0]				RES3[19:16]			
		15:8	RES3[15:8]							
		7:0	RES3[7:0]							
0x0B08	AD3CH3CNT	31:24	CNTSTAT3[15:8]							
		23:16	CNTSTAT3[7:0]							
		15:8	CNT3[15:8]							
		7:0	CNT3[7:0]							
0x0B0C	AD3CH3CMPLO	31:24	LO3[31:24]							
		23:16	LO3[23:16]							
		15:8	LO3[15:8]							
		7:0	LO3[7:0]							
0x0B10	AD3CH3CMPHI	31:24	HI3[31:24]							
		23:16	HI3[23:16]							
		15:8	HI3[15:8]							
		7:0	HI3[7:0]							
0x0B14	AD3CH3ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0B18	AD3CH4CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0B1C	AD3CH4CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0B20	AD3CH4DATA	31:24	DATA4[31:24]							
		23:16	DATA4[23:16]							
		15:8	DATA4[15:8]							
		7:0	DATA4[7:0]							
0x0B24	AD3CH4RES	31:24	RESF4[11:4]							
		23:16	RESF4[3:0]							
		15:8					RES4[11:8]			
		7:0	RES4[7:0]							
0x0B28	AD3CH4CNT	31:24	CNTSTAT4[15:8]							
		23:16	CNTSTAT4[7:0]							
		15:8	CNT4[15:8]							
		7:0	CNT4[7:0]							
0x0B2C	AD3CH4CMPLO	31:24	LO4[31:24]							
		23:16	LO4[23:16]							
		15:8	LO4[15:8]							
		7:0	LO4[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0B30	AD3CH4CMPHI	31:24	HI4[31:24]							
		23:16	HI4[23:16]							
		15:8	HI4[15:8]							
		7:0	HI4[7:0]							
0x0B34	AD3CH4ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0B38	AD3CH5CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0B3C	AD3CH5CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0B40	AD3CH5DATA	31:24	DATA5[31:24]							
		23:16	DATA5[23:16]							
		15:8	DATA5[15:8]							
		7:0	DATA5[7:0]							
0x0B44	AD3CH5RES	31:24	RESF5[19:12]							
		23:16	RESF5[11:4]							
		15:8	RESF5[3:0]				RES5[11:8]			
		7:0	RES5[7:0]							
0x0B48	AD3CH5CNT	31:24	CNTSTAT5[15:8]							
		23:16	CNTSTAT5[7:0]							
		15:8	CNT5[15:8]							
		7:0	CNT5[7:0]							
0x0B4C	AD3CH5CMPLO	31:24	LO5[31:24]							
		23:16	LO5[23:16]							
		15:8	LO5[15:8]							
		7:0	LO5[7:0]							
0x0B50	AD3CH5CMPHI	31:24	HI5[31:24]							
		23:16	HI5[23:16]							
		15:8	HI5[15:8]							
		7:0	HI5[7:0]							
0x0B54	AD3CH5ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0B58	AD3CH6CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0B5C	AD3CH6CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0B60	AD3CH6DATA	31:24	DATA6[31:24]							
		23:16	DATA6[23:16]							
		15:8	DATA6[15:8]							
		7:0	DATA6[7:0]							
0x0B64	AD3CH6RES	31:24	RESF6[11:4]							
		23:16	RESF6[3:0]							
		15:8					RES6[11:8]			
		7:0	RES6[7:0]							
0x0B68	AD3CH6CNT	31:24	CNTSTAT6[15:8]							
		23:16	CNTSTAT6[7:0]							
		15:8	CNT6[15:8]							
		7:0	CNT6[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0B6C	AD3CH6CMPLO	31:24	LO6[31:24]							
		23:16	LO6[23:16]							
		15:8	LO6[15:8]							
		7:0	LO6[7:0]							
0x0B70	AD3CH6CMPHI	31:24	HI6[31:24]							
		23:16	HI6[23:16]							
		15:8	HI6[15:8]							
		7:0	HI6[7:0]							
0x0B74	AD3CH6ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0B78	AD3CH7CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0B7C	AD3CH7CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0B80	AD3CH7DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							
0x0B84	AD3CH7RES	31:24	RESF7[11:4]							
		23:16	RESF7[3:0]							
		15:8					RES7[11:8]			
		7:0	RES7[7:0]							
0x0B88	AD3CH7CNT	31:24	CNTSTAT7[15:8]							
		23:16	CNTSTAT7[7:0]							
		15:8	CNT7[15:8]							
		7:0	CNT7[7:0]							
0x0B8C	AD3CH7CMPLO	31:24	LO7[31:24]							
		23:16	LO7[23:16]							
		15:8	LO7[15:8]							
		7:0	LO7[7:0]							
0x0B90	AD3CH7CMPHI	31:24	HI7[31:24]							
		23:16	HI7[23:16]							
		15:8	HI7[15:8]							
		7:0	HI7[7:0]							
0x0B94	AD3CH7ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0B98	AD3CH8CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0B9C	AD3CH8CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0BA0	AD3CH8DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							
0x0BA4	AD3CH8RES	31:24	RESF7[11:4]							
		23:16	RESF7[3:0]							
		15:8					RES7[11:8]			
		7:0	RES7[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0BA8	AD3CH8CNT	31:24	CNTSTAT7[15:8]							
		23:16	CNTSTAT7[7:0]							
		15:8	CNT7[15:8]							
		7:0	CNT7[7:0]							
0x0BAC	AD3CH8CMPLO	31:24	LO7[31:24]							
		23:16	LO7[23:16]							
		15:8	LO7[15:8]							
		7:0	LO7[7:0]							
0x0BB0	AD3CH8CMPHI	31:24	HI7[31:24]							
		23:16	HI7[23:16]							
		15:8	HI7[15:8]							
		7:0	HI7[7:0]							
0x0BB4	AD3CH8ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0BB8	AD3CH9CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0BBC	AD3CH9CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0BC0	AD3CH9DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							
0x0BC4	AD3CH9RES	31:24	RESF7[11:4]							
		23:16	RESF7[3:0]							
		15:8					RES7[11:8]			
		7:0	RES7[7:0]							
0x0BC8	AD3CH9CNT	31:24	CNTSTAT7[15:8]							
		23:16	CNTSTAT7[7:0]							
		15:8	CNT7[15:8]							
		7:0	CNT7[7:0]							
0x0BCC	AD3CH9CMPLO	31:24	LO7[31:24]							
		23:16	LO7[23:16]							
		15:8	LO7[15:8]							
		7:0	LO7[7:0]							
0x0BD0	AD3CH9CMPHI	31:24	HI7[31:24]							
		23:16	HI7[23:16]							
		15:8	HI7[15:8]							
		7:0	HI7[7:0]							
0x0BD4	AD3CH9ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0BD8	AD3CH10CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0BDC	AD3CH10CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0BE0	AD3CH10DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0BE4	AD3CH10RES	31:24	RESF7[11:4]							
		23:16	RESF7[3:0]							
		15:8					RES7[11:8]			
		7:0	RES7[7:0]							
0x0BE8	AD3CH10CNT	31:24	CNTSTAT7[15:8]							
		23:16	CNTSTAT7[7:0]							
		15:8	CNT7[15:8]							
		7:0	CNT7[7:0]							
0x0BEC	AD3CH10CMPLO	31:24	LO7[31:24]							
		23:16	LO7[23:16]							
		15:8	LO7[15:8]							
		7:0	LO7[7:0]							
0x0BF0	AD3CH10CMPHI	31:24	HI7[31:24]							
		23:16	HI7[23:16]							
		15:8	HI7[15:8]							
		7:0	HI7[7:0]							
0x0BF4	AD3CH10ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							
0x0BF8	AD3CH11CON1	31:24	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
		23:16	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
		15:8	ACCNUM[1:0]		TRG2SRC[5:0]					
		7:0	MODE[1:0]		TRG1SRC[5:0]					
0x0BFC	AD3CH11CON2	31:24	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
		23:16	ADCMPSTAT[9:0]							
		15:8		CRITERIA[2:0]					ADCMPCNT[9:8]	
		7:0	ADCMPCNT[7:0]							
0x0C00	AD3CH11DATA	31:24	DATA7[31:24]							
		23:16	DATA7[23:16]							
		15:8	DATA7[15:8]							
		7:0	DATA7[7:0]							
0x0C04	AD3CH11RES	31:24	RESF7[11:4]							
		23:16	RESF7[3:0]							
		15:8					RES7[11:8]			
		7:0	RES7[7:0]							
0x0C08	AD3CH11CNT	31:24	CNTSTAT7[15:8]							
		23:16	CNTSTAT7[7:0]							
		15:8	CNT7[15:8]							
		7:0	CNT7[7:0]							
0x0C0C	AD3CH11CMPLO	31:24	LO7[31:24]							
		23:16	LO7[23:16]							
		15:8	LO7[15:8]							
		7:0	LO7[7:0]							
0x0C10	AD3CH11CMPHI	31:24	HI7[31:24]							
		23:16	HI7[23:16]							
		15:8	HI7[15:8]							
		7:0	HI7[7:0]							
0x0C14	AD3CH11ACC	31:24	ACC[31:24]							
		23:16	ACC[23:16]							
		15:8	ACC[15:8]							
		7:0	ACC[7:0]							

16.3.1. ADC n Control Register

Name: ADnCON
Offset: 0x800, 0x900, 0xA80

Notes:
1. Timing is approximate and dependent on the 32K oscillator's accuracy. Changing this value during ADC operation may cause erratic re-calibration timing.
2. Recovery from Standby mode requires 230 ADC clock cycles.
3. Set the ADON bit only after the ADC module has been configured. Changing ADC configuration bits when ADON = 1 will result in unpredictable behavior.

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit; W = Writable bit; S = Set Only bit; C = Clear Only bit

Bit	31	30	29	28	27	26	25	24
	ADRDY	CALRDY	CALREQ	ACALEN	CALRATE[1:0]		MODE[1:0]	
Access	HS/HC/R	HS/HC/R	R/W/HC	R/W	R/W	R/W	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RPTCNT[5:0]						Reserved	STNDBY
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON					TSTLOCK		TSTEN
Access	R/W					R/S		R/W/C
Reset	0					0		0

Bit	7	6	5	4	3	2	1	0
	BUFEN	CALCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit 31 – ADRDY ADC Ready bit

The bit indicates that the ADC has been enabled and has completed its power-up and self-calibration process.

Value	Description
1	ADC is ready.
0	ADC is off.

Bit 30 – CALRDY Calibration Done bit

Value	Description
1	Calibration cycle has finished.
0	Calibration was not started or is in progress.

Bit 29 – CALREQ Software Calibration Cycle Request bit

Value	Description
1	Setting this bit executes the calibration cycle.
0	Calibration cycle is not requested.

Bit 28 – ACALEN Auto Calibration Enable bit

This bit enables periodic ADC recalibration. The calibration cycle's period is defined by the CALRATE (ADnCON[27:26]) bits.

Value	Description
1	Periodic recalibration is enabled.
0	Periodic recalibration is off.

Bits 27:26 – CALRATE[1:0] Auto Re-Calibration Period bits [(1)]

Value	Description
11	Recalibration every 4096 seconds
10	Recalibrate every 1024 seconds
01	Recalibrate every 64 seconds
00	Recalibrate every second

Bits 25:24 – MODE[1:0] ADC Operation Mode Status bits

Value	Description
1x	ADC is on.
01	ADC is in Standby mode.
00	ADC is powered down.

Bits 23:18 – RPTCNT[5:0] Conversion Repeat Timer Period bits

This timer can be used to generate ADC triggers periodically by selecting the RPTCNT timer as a trigger source in the TRG2SRC[5:0] (ADnCHxCON1[13:8]) bits. This timer counts ADC clock cycles.

Value	Description
111111	64 ADC clock cycles between triggers
...	
0000010	3 ADC clock cycles between triggers
000001	2 ADC clock cycles between triggers
000000	1 ADC clock cycle between triggers

Bit 17 – Reserved

Bit 16 – STNDBY ADC Standby Enable Bit[(2)]

Value	Description
1	ADC module is in a power reduced mode.
0	ADC is in normal active mode.

Bit 15 – ON ADC Enable bit[(3)]

Value	Description
1	ADC module is enabled.
0	ADC module is disabled.

Bit 10 – TSTLOCK TSTEN (ADnCON[8]) Lock bit

Value	Description
1	TSTEN bit cannot be set to 1 but can be cleared to 0.
0	TSTEN bit can be set to 1.

Bit 8 – TSTEN Test Mode Enable bit

In the test mode, the result of a conversion for all channels is overwritten with a value from the ADnDATAOVR register.

Value	Description
1	The test mode is enabled.
0	The test mode is disabled.

Bit 7 – BUFEN Buffer Enable bit

Value	Description
1	Input buffer is enabled.
0	Input buffer is disabled and bypassed.

Bits 6:5 – CALCNT[1:0] ADC Idle Cycles Prior to Calibration bits

Value	Description
11	Wait for 16 activity-free ADC clock cycles before initiating the requested calibration
10	Wait for 8 activity-free ADC clock cycles before initiating the requested calibration
01	Wait for 4 activity-free ADC clock cycles before initiating the requested calibration
00	Wait for 2 activity-free ADC clock cycles before initiating the requested calibration

16.3.2. ADC n Test Mode Data Register

Name: ADnDATAOVR
Offset: 0x804, 0x904, 0xA84

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DATAOVR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATAOVR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATAOVR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATAOVR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATAOVR[31:0] Conversion Result Value in Test Mode bits

16.3.3. ADC n Data Ready Flags Register

Name: ADnSTAT
Offset: 0x808, 0x908, 0xA88

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CH[7:0]RDY							
Access	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CH[7:0]RDY Channel x Data Ready bits
Each bit in this register is set by hardware when the corresponding channel x data is written into the ADnDATAx register. The bit is cleared by hardware when the ADnDATAx register is read by software.

16.3.4. ADC n Result Ready Status Register

Name: ADnRSTAT
Offset: 0x80C, 0x90C, 0xA8C

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CH[7:0]RRDY							
Access	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC	HS/HC
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CH[7:0]RRDY Channel x Conversion Result Ready bits
Each bit in this register is set by hardware when the corresponding channel x conversion result is written into the ADnRESx register. The bit is cleared by hardware when the ADnRESx register is read by software.

16.3.5. ADC n Comparators Status Register

Name: ADnCMPSTAT
Offset: 0x810, 0x910, 0xA90

Legend: n = ADC number; HS = Hardware Settable bit; C = Clear Only bit; R = Readable bit

Note: CMP[15:8]FLG bits are available on ADC5 only.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CMP[7:0]FLG							
Access	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R	HS/C/R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CMP[7:0]FLG Channel x Comparator Event Detection bits
These bits are set when ADC channel data meets comparison criteria and cleared when a “0” is written to this bit by software.

16.3.6. ADC n Software Triggers Request Register

Name: ADnSWTRG
Offset: 0x814, 0x914, 0xA94

Legend: n = ADC number; W = Writable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CH[7:0]TRG							
Access	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R	W/HC/R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CH[7:0]TRG Channel x Software Trigger Request bits
A software trigger for channel x is generated when the corresponding bit is set in this register. The software trigger must be selected for the channel in TRG1SRC[5:0] (ADnCHxCON1[5:0]) or TRG2SRC[5:0] (ADnCHxCON1[13:8]) bits.

16.3.7. ADC n Channel 0 Control Register 1

Name: ADnCH0CON1
Offset: 0x818, 0x918, 0xA98

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single-ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit (3)

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
11111	62.5 T_{AD}
11100	60.5 T_{AD}
...	
00010	4.5 T_{AD}
00001	2.5 T_{AD}
00000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits (1)

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits
Refer to Table 16-3 for the available trigger source selection bits.

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[5:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[5:0] trigger.
01	Window gated by TRG1SRC[:0] source. In this mode, the samples are accumulated when a signal selected by TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[5:0] trigger. The number of conversions is limited by CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits
Refer to Table 16-3 for the available trigger source selection bits.

16.3.8. ADC n Channel 0 Control Register 2

Name: ADnCH0CON2
Offset: 0x81C, 0x91C, 0xA9C

Legend: n = ADC number; x = Channel number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPCNTMOD			ADCMPSTAT[9:8]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CMPMOD[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bit 31 – ACCRO Accumulator Roll-Over Enable bit

The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests will be blocked until the process is completed.
0	Oversampling can be interrupted by a high-priority conversion request

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bit 28 – CMPCNTMOD Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits

Bits 25:24 – ADCMPSTAT[9:8] Comparison Violation Count Status bits

These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.
The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 23:16 – ADCMPSTAT[7:0] Comparison Violation Count Status bits

These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.
The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CMPMOD[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) the ADnCMPLOx register.
011	
010	Conversion result is within the bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.9. ADC n Channel 0 Data Register

Name: ADnCH0DATA
Offset: 0x820, 0x920, 0xAA0

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA[31:0] Channel Data/Primary Accumulator bits

16.3.10. ADC n Channel 0 Result Register

Name: ADnCH0RES
Offset: 0x824, 0x924, 0xAA4

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	RESF[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF[3:0]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bit	15	14	13	12	11	10	9	8
					RES[11:8]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – RESF[11:0] Conversion Result bits

Bits 11:0 – RES[11:0] Conversion Result bits

16.3.11. ADC n Channel 0 Counter Register

Name: ADnCH0CNT
Offset: 0x828, 0x928, 0xAA8

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT[15:0] Count Status bits
Number of conversions done in integration (MODE[1:0] bits = '10') and window (MODE[1:0] bits = '01') sampling modes.

Bits 15:0 – CNT[15:0] Count bits
Number of samples for an integration sampling mode (MODE[1:0] bits = '10') and maximum number of samples for a window sampling mode (MODE[1:0] bits = '01').

16.3.12. ADC n Channel 0 Low Compare Register

Name: ADnCH0CMPLO
Offset: 0x82C, 0x92C, 0xAAC

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	CMPLO[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CMPLO[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CMPLO[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPLO[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CMPLO[31:0] Low Threshold Comparator Value bits

16.3.13. ADC n Channel 0 High Compare Register

Name: ADnCH0CMPHI
Offset: 0x830, 0x930, 0xAB0

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	CMPHI[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CMPHI[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CMPHI[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPHI[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CMPHI[31:0] High Threshold Comparator Value bits

16.3.14. ADC 1 Channel x Secondary Accumulator Register

Name: AD1CHxACC
Offset: 0x834, 0x854, 0x874, 0x894, 0x8B4, 0x8D4, 0x8F4

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACC[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ACC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ACC[31:0] Secondary Accumulator bits

16.3.15. ADC n Channel 1 Control Register 1

Name: ADnCH1CON1
Offset: 0x838, 0x938, 0xAB8

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single-ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit (3)

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
1111	62.5 T_{AD}
1110	60.5 T_{AD}
...	
0010	4.5 T_{AD}
0001	2.5 T_{AD}
0000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits (1)

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode, the samples are accumulated when a signal selected by the TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[4:0] trigger. The number of conversions is limited by the CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

16.3.16. ADC n Channel 1 Control Register 2

Name: ADnCH1CON2
Offset: 0x83C, 0x93C, 0xABC

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CRITERIA[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ACCRO Accumulator Roll-Over Enable bit
The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests will be blocked out until the process is completed.
0	Oversampling can be interrupted by a high-priority conversion request

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bits 29:28 – CMPMODE[1:0] Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits.
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits.

Bits 25:16 – ADCMPSTAT[9:0] Comparison Violation Count Status bits
These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.

The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CRITERIA[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) the ADnCMPLOx register.
011	
010	Conversion result is within bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.17. ADC n Channel 1 Data Register

Name: ADnCH1DATA
Offset: 0x840, 0x940, 0xAC0

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATA1[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA1[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA1[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA1[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA1[31:0] Channel Data/Primary Accumulator bits

16.3.18. ADC n Channel 1 Result Register

Name: ADnCH1RES
Offset: 0x844, 0x944, 0xAC4

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	RESF1[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF1[3:0]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bit	15	14	13	12	11	10	9	8
					RES1[11:8]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES1[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – RESF1[11:0] Conversion Result bits

Bits 11:0 – RES1[11:0] Conversion Result bits

16.3.19. ADC n Channel 1 Counter Register

Name: ADnCH1CNT
Offset: 0x848, 0x948, 0xAC8

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT1[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	CNTSTAT1[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	CNT1[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	CNT1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT1[15:0]
Number of conversions done in integration (MODE[1:0] bits = '10') and window (MODE[1:0] bits = '01') sampling modes.

Bits 15:0 – CNT1[15:0]
Number of samples for an integration sampling mode (MODE[1:0] bits = '10') and maximum number of samples for a window sampling mode (MODE[1:0] bits = '01').

16.3.20. ADC n Channel 1 Low Compare Register

Name: ADnCH1CMPLO
Offset: 0x84C, 0x94C, 0xACC

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO1[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO1[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO1[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO1[31:0] Low Threshold Comparator Value bits

16.3.21. ADC n Channel 1 High Compare Register

Name: ADnCH1CMPHI
Offset: 0x850, 0x950, 0xAD0

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI1[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI1[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI1[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI1[31:0] High Threshold Comparator Value bits

16.3.22. ADC n Channel 2 Control Register 1

Name: ADnCH2CON1
Offset: 0x858, 0x958, 0xAD8

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in the ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in the ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit (3)

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
1111	62.5 T_{AD}
1110	60.5 T_{AD}
...	
0010	4.5 T_{AD}
0001	2.5 T_{AD}
0000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits (1)

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode, the samples are accumulated when a signal selected by the TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[4:0] trigger. The number of conversions is limited by the CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

16.3.23. ADC n Channel 2 Control Register 2

Name: ADnCH2CON2
Offset: 0x85C, 0x95C, 0xADC

Legend: n = ADC number; R = Readable bit; W = Writable bit

Note:
1. This bit is available on the last two channels only with an implemented secondary accumulator.

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CRITERIA[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ACCRO Accumulator Roll-Over Enable bit[1]
The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests will be blocked out until the process is completed.
0	Oversampling can be interrupted by a high-priority conversion request.

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bits 29:28 – CMPMODE[1:0] Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits.
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits.

Bits 25:16 – ADCMPSTAT[9:0] Comparison Violation Count Status bits

These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.
The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CRITERIA[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) ADnCMPLOx register.
011	
010	Conversion result is within bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.24. ADC n Channel 2 Data Register

Name: ADnCH2DATA
Offset: 0x860, 0x960, 0xAE0

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATA2[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA2[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA2[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA2[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA2[31:0] Channel Data/Primary Accumulator bits

16.3.25. ADC n Channel 2 Result Register

Name: ADnCH2RES
Offset: 0x864, 0x964, 0xAE4

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	RESF2[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF2[3:0]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bit	15	14	13	12	11	10	9	8
					RES2[11:8]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES2[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – RESF2[11:0] Conversion Result bits

Bits 11:0 – RES2[11:0] Conversion Result bits

16.3.26. ADC n Channel 2 Counter Register

Name: ADnCH2CNT
Offset: 0x868, 0x968, 0xAE8

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT2[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT2[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT2[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT2[15:0]
Number of conversions done in integration (MODE[1:0] bits = '10') and window (MODE[1:0] bits = '01') sampling modes.

Bits 15:0 – CNT2[15:0]
Number of samples for an integration sampling mode (MODE[1:0] bits = '10') and maximum number of samples for a window sampling mode (MODE[1:0] bits = '01').

16.3.27. ADC n Channel 2 Low Compare Register

Name: ADnCH2CMPLO
Offset: 0x86C, 0x96C, 0xAEC

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO2[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO2[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO2[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO2[31:0] Low Threshold Comparator Value bits

16.3.28. ADC n Channel 2 High Compare Register

Name: ADnCH2CMPHI
Offset: 0x870, 0x970, 0xAF0

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI2[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI2[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI2[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI2[31:0] High Threshold Comparator Value bits

16.3.29. ADC n Channel 3 Control Register 1

Name: ADnCH3CON1
Offset: 0x878, 0x978, 0xAF8

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in the ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in the ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit (3)

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
1111	62.5 T_{AD}
1110	60.5 T_{AD}
. . .	
0010	4.5 T_{AD}
0001	2.5 T_{AD}
0000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits (1)

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode, the samples are accumulated when a signal selected by the TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[4:0] trigger. The number of conversions is limited by the CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

16.3.30. ADC n Channel 3 Control Register 2

Name: ADnCH3CON2
Offset: 0x87C, 0x97C, 0xAFC

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CRITERIA[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ACCRO Accumulator Roll-Over Enable bit

The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests will be blocked out until the process is completed.
0	Oversampling can be interrupted by a high-priority conversion request.

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bits 29:28 – CMPMODE[1:0] Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits.
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits.

Bits 25:16 – ADCMPSTAT[9:0] Comparison Violation Count Status bits

These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.

The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CRITERIA[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) the ADnCMPLOx register.
011	
010	Conversion result is within bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.31. ADC n Channel 3 Data Register

Name: ADnCH3DATA
Offset: 0x880, 0x980, 0xB00

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATA3[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA3[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA3[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA3[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA3[31:0] Channel Data/Primary Accumulator bits

16.3.32. ADC n Channel 3 Result Register

Name: ADnCH3RES
Offset: 0x884, 0x984, 0xB04

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	RESF3[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF3[3:0]				RES3[19:16]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RES3[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES3[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – RESF3[11:0] Conversion Result bits

Bits 19:0 – RES3[19:0] Conversion Result bits

16.3.33. ADC n Channel 3 Counter Register

Name: ADnCH3CNT
Offset: 0x888, 0x988, 0xB08

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT3[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT3[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT3[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT3[15:0]
Number of conversions done in integration (MODE[1:0] bits = ‘10’) and window (MODE[1:0] bits = ‘01’) sampling modes.

Bits 15:0 – CNT3[15:0]
Number of samples for an integration sampling mode (MODE[1:0] bits = ‘10’) and maximum number of samples for a window sampling mode (MODE[1:0] bits = ‘01’).

16.3.34. ADC n Channel 3 Low Compare Register

Name: ADnCH3CMPLO
Offset: 0x88C, 0x98C, 0xB0C

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO3[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO3[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO3[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO3[31:0] Low Threshold Comparator Value bits

16.3.35. ADC n Channel 3 High Compare Register

Name: ADnCH3CMPHI
Offset: 0x890, 0x990, 0xB10

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI3[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	HI3[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	HI3[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	HI3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI3[31:0] High Threshold Comparator Value bits

16.3.36. ADC n Channel 4 Control Register 1

Name: ADnCH4CON1
Offset: 0x898, 0x998, 0xB18

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit (3)

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
1111	62.5 T_{AD}
1110	60.5 T_{AD}
...	
0010	4.5 T_{AD}
0001	2.5 T_{AD}
0000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits [(1)]

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode, the samples are accumulated when a signal selected by the TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[4:0] trigger. The number of conversions is limited by the CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

16.3.37. ADC n Channel 4 Control Register 2

Name: ADnCH4CON2
Offset: 0x89C, 0x99C, 0xB1C

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CRITERIA[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ACCRO Accumulator Roll-Over Enable bit

The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests will be blocked until the process is completed.
0	Oversampling can be interrupted by a high-priority conversion request.

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bits 29:28 – CMPMODE[1:0] Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits.
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits.

Bits 25:16 – ADCMPSTAT[9:0] Comparison Violation Count Status bits

These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.

The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CRITERIA[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) the ADnCMPLOx register.
011	
010	Conversion result is within bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.38. ADC n Channel 4 Data Register

Name: ADnCH4DATA
Offset: 0x8A0, 0x9A0, 0xB20

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATA4[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA4[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA4[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA4[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA4[31:0] Channel Data/Primary Accumulator bits

16.3.39. ADC n Channel 4 Result Register

Name: ADnCH4RES
Offset: 0x8A4, 0x9A4, 0xB24

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	RESF4[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF4[3:0]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bit	15	14	13	12	11	10	9	8
					RES4[11:8]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES4[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – RESF4[11:0] Conversion Result bits

Bits 11:0 – RES4[11:0] Conversion Result bits

16.3.40. ADC n Channel 4 Counter Register

Name: ADnCH4CNT
Offset: 0x8A8, 0x9A8, 0xB28

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT4[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT4[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT4[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT4[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT4[15:0]
Number of conversions done in integration (MODE[1:0] bits = ‘10’) and window (MODE[1:0] bits = ‘01’) sampling modes.

Bits 15:0 – CNT4[15:0]
Number of samples for an integration sampling mode (MODE[1:0] bits = ‘10’) and maximum number of samples for a window sampling mode (MODE[1:0] bits = ‘01’).

16.3.41. ADC n Channel 4 Low Compare Register

Name: ADnCH4CMPLO
Offset: 0x8AC, 0x9AC, 0xB2C

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO4[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO4[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO4[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO4[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO4[31:0] Low Threshold Comparator Value bits

16.3.42. ADC n Channel 4 High Compare Register

Name: ADnCH4CMPHI
Offset: 0x8B0, 0x9B0, 0xB30

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI4[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI4[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI4[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI4[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI4[31:0] High Threshold Comparator Value bits

16.3.43. ADC n Channel 5 Control Register 1

Name: ADnCH5CON1
Offset: 0x8B8, 0x9B8, 0xB38

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit [(3)]

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
1111	62.5 T_{AD}
1110	60.5 T_{AD}
...	
0010	4.5 T_{AD}
0001	2.5 T_{AD}
0000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits (1)

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode, the samples are accumulated when a signal selected by TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[4:0] trigger. The number of conversions is limited by CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

16.3.44. ADC n Channel 5 Control Register 2

Name: ADnCH5CON2
Offset: 0x8BC, 0x9BC, 0xB3C

Legend: n = ADC number: R = Readable bit: W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CRITERIA[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ACCRO Accumulator Roll-Over Enable bit
The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests will be blocked out until the process is completed.
0	Oversampling can be interrupted by a high priority conversion request.

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bits 29:28 – CMPMODE[1:0] Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits.
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits.

Bits 25:16 – ADCMPSTAT[9:0] Comparison Violation Count Status bits
These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.

The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CRITERIA[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) the ADnCMPLOx register.
011	
010	Conversion result is within bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.45. ADC n Channel 5 Data Register

Name: ADnCH5DATA
Offset: 0x8C0, 0x9C0, 0xB40

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATA5[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA5[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA5[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA5[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA5[31:0] Channel Data/Primary Accumulator bits

16.3.46. ADC n Channel 5 Result Register

Name: ADnCH5RES
Offset: 0x8C4, 0x9C4, 0xB44

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	RESF5[19:12]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF5[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RESF5[3:0]				RES5[11:8]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES5[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:12 – RESF5[19:0] Conversion Result bits

Bits 11:0 – RES5[11:0] Conversion Result bits

16.3.47. ADC n Channel 5 Counter Register

Name: ADnCH5CNT
Offset: 0x8C8, 0x9C8, 0xB48

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT5[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT5[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT5[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT5[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT5[15:0]
Number of conversions done in integration (MODE[1:0] bits = '10') and window (MODE[1:0] bits = '01') sampling modes.

Bits 15:0 – CNT5[15:0]
Number of samples for an integration sampling mode (MODE[1:0] bits = '10') and maximum number of samples for a window sampling mode (MODE[1:0] bits = '01').

16.3.48. ADC n Channel 5 Low Compare Register

Name: ADnCH5CMPLO
Offset: 0x8CC, 0x9CC, 0xB4C

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO5[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO5[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO5[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO5[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO5[31:0] Low Threshold Comparator Value bits

16.3.49. ADC n Channel 5 High Compare Register

Name: ADnCH5CMPHI
Offset: 0x8D0, 0x9D0, 0xB50

Legend: n = ADC number; x = Channel number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI5[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI5[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI5[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI5[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI5[31:0] High Threshold Comparator Value bits

16.3.50. ADC n Channel 6 Control Register 1

Name: ADnCH6CON1
Offset: 0x8D8, 0x9D8, 0xB58

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit (3)

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
1111	62.5 T_{AD}
1110	60.5 T_{AD}
...	
0010	4.5 T_{AD}
0001	2.5 T_{AD}
0000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits [(1)]

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode, the samples are accumulated when a signal selected by the TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[4:0] trigger. The number of conversions is limited by the CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

16.3.51. ADC n Channel 6 Control Register 2

Name: ADnCH6CON2
Offset: 0x8DC, 0x9DC, 0xB5C

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CRITERIA[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ACCRO Accumulator Roll-Over Enable bit

The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests will be blocked out until the process is completed.
0	Oversampling can be interrupted by a high priority conversion request.

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bits 29:28 – CMPMODE[1:0] Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits.
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits.

Bits 25:16 – ADCMPSTAT[9:0] Comparison Violation Count Status bits

These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.

The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CRITERIA[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) the ADnCMPLOx register.
011	
010	Conversion result is within bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.52. ADC n Channel 6 Data Register

Name: ADnCH6DATA
Offset: 0x8E0, 0x9E0, 0xB60

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATA6[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA6[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA6[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA6[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA6[31:0] Channel Data/Primary Accumulator bits

16.3.53. ADC n Channel 6 Result Register

Name: ADnCH6RES
Offset: 0x8E4, 0x9E4, 0xB64

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	RESF6[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF6[3:0]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bit	15	14	13	12	11	10	9	8
					RES6[11:8]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES6[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – RESF6[11:0] Conversion Result bits

Bits 11:0 – RES6[11:0] Conversion Result bits

16.3.54. ADC n Channel 6 Counter Register

Name: ADnCH6CNT
Offset: 0x8E8, 0x9E8, 0xB68

Legend: n = ADC number; x, HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT6[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT6[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT6[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT6[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT6[15:0]
Number of conversions done in integration (MODE[1:0] bits = '`10`') and window (MODE[1:0] bits = '`01`') sampling modes.

Bits 15:0 – CNT6[15:0]
Number of samples for an integration sampling mode (MODE[1:0] bits = '`10`') and maximum number of samples for a window sampling mode (MODE[1:0] bits = '`01`').

16.3.55. ADC n Channel 6 Low Compare Register

Name: ADnCH6CMPLO
Offset: 0x8EC, 0x9EC, 0xB6C

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO6[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO6[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO6[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO6[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO6[31:0] Low Threshold Comparator Value bits

16.3.56. ADC n Channel 6 High Compare Register

Name: ADnCH6CMPHI
Offset: 0x8F0, 0x9F0, 0xB70

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI6[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI6[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI6[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI6[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI6[31:0] High Threshold Comparator Value bits

16.3.57. ADC 2 Channel x Secondary Accumulator Register

Name: AD2CHxACC
Offset: 0x934, 0x954, 0x974, 0x994, 0x9B4, 0x9D4, 0x9F4, 0x0A14

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACC[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ACC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ACC[31:0] Secondary Accumulator bits

16.3.58. ADC n Channel 7 Control Register 1

Name: ADnCH7CON1
Offset: 0x9F8, 0xB78

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit [(3)]

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
1111	62.5 T_{AD}
1110	60.5 T_{AD}
...	
0010	4.5 T_{AD}
0001	2.5 T_{AD}
0000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits (1)

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode, the samples are accumulated when a signal selected by the TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[4:0] trigger. The number of conversions is limited by the CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

16.3.59. ADC n Channel 7 Control Register 2

Name: ADnCH7CON2
Offset: 0x9FC, 0xB7C

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CRITERIA[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ACCRO Accumulator Roll-Over Enable bit

The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous non-interruptible burst, during which all other conversion requests will be blocked out until the process is completed.
0	Oversampling can be interrupted by a high priority conversion request.

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bits 29:28 – CMPMODE[1:0] Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits.
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits.

Bits 25:16 – ADCMPSTAT[9:0] Comparison Violation Count Status bits

These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.

The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CRITERIA[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) the ADnCMPLOx register.
011	
010	Conversion result is within bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.60. ADC n Channel 7 Data Register

Name: ADnCH7DATA
Offset: 0xA00, 0xB80

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	DATA7[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA7[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA7[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA7[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA7[31:0] Channel Data/Primary Accumulator bits

16.3.61. ADC n Channel 7 Result Register

Name: ADnCH7RES
Offset: 0xA04, 0xB84

Legend: n = ADC number; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	RESF7[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF7[3:0]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bit	15	14	13	12	11	10	9	8
					RES7[11:8]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES7[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – RESF7[11:0] Conversion Result bits

Bits 11:0 – RES7[11:0] Conversion Result bits

16.3.62. ADC n Channel 7 Counter Register

Name: ADnCH7CNT
Offset: 0xA08, 0xB88

Legend: n = ADC number; HS = Hardware Settable bit; HC = Hardware Clearable bit; R = Readable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT7[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT7[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT7[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT7[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT7[15:0]
Number of conversions done in integration (MODE[1:0] bits = '10') and window (MODE[1:0] bits = '01') sampling modes.

Bits 15:0 – CNT7[15:0]
Number of samples for an integration sampling mode (MODE[1:0] bits = '10') and maximum number of samples for a window sampling mode (MODE[1:0] bits = '01').

16.3.63. ADC n Channel 7 Low Compare Register

Name: ADnCH7CMPLO
Offset: 0xA0C, 0xB8C

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO7[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO7[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO7[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO7[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO7[31:0] Low Threshold Comparator Value bits

16.3.64. ADC n Channel 7 High Compare Register

Name: ADnCH7CMPHI
Offset: 0xA10, 0xB90

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI7[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI7[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI7[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI7[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI7[31:0] High Threshold Comparator Value bits

16.3.65. ADC 2 Channel 7 Secondary Accumulator Register

Name: AD2CH7ACC
Offset: 0x0A14

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACC[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ACC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ACC[31:0] Secondary Accumulator bits

16.3.66. ADC 3 Channel x Secondary Accumulator Register

Name: AD3CHxACC
Offset: 0xAB4, 0xAD4, 0xAF4, 0xB14, 0xB34, 0xB54, 0xB74, 0xB94, 0xBB4, 0xBD4, 0xBF4, 0xC14

Legend: n = ADC number; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACC[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACC[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ACC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ACC[31:0] Secondary Accumulator bits

16.3.67. ADC 3 Channel x Control Register 1

Name: AD3CHxCON1
Offset: 0xB98, 0xBB8, 0xBD8, 0xBF8

Legend:x = channel; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DIFF	FRAC	NINSEL[1:0]		PINSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRG1POL	EIEN	IRQSEL	SAMC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCNUM[1:0]		TRG2SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MODE[1:0]		TRG1SRC[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DIFF Differential Input Enable bit

Value	Description
1	Differential input mode; data is output as signed (two's complement).
0	Single ended input mode; data is output as unsigned.

Bit 30 – FRAC Fractional Data Output Format Enable bit

Value	Description
1	Results in ADnDATAx and ADnRESx registers are aligned to the left (in the fractional format).
0	Results in ADnDATAx and ADnRESx registers are aligned to the right.

Bits 29:28 – NINSEL[1:0] Negative Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bits 27:24 – PINSEL[3:0] Positive Analog Input Selection bits
Refer to Table 16-2 for the available positive analog inputs.

Bit 23 – TRG1POL Starting Trigger Polarity Selection bits

Value	Description
1	Active level of the signal selected by TRG1SRC[4:0] bits is low; a falling edge generates a conversion request.
0	Active level of the signal selected by TRG1SRC[4:0] bits is high; a rising edge generates a conversion request.

Bit 22 – EIEN Early Interrupt Enable bit [(3)]

Value	Description
1	Early interrupt is enabled.
0	Normal interrupt timing

Bit 21 – IRQSEL Channel Ready Interrupt Request Select bit

Value	Description
1	The channel interrupt is generated when data is ready in the ADnDATAx register.
0	The channel interrupt is generated after each single conversion when the result is ready in the ADnRESx register.

Bits 20:16 – SAMC[4:0] Sampling Time Selection bits

Value	Description
1111	62.5 T_{AD}
1110	60.5 T_{AD}
...	
0010	4.5 T_{AD}
0001	2.5 T_{AD}
0000	0.5 T_{AD}

Bits 15:14 – ACCNUM[1:0] Oversampling Mode Number of Samples Selection bits [(1)]

Value	Description
11	256 samples, 16 bits result in the ADnDATAx register
10	64 samples, 15 bits result in the ADnDATAx register
01	16 samples, 14 bits result in the ADnDATAx register
00	4 samples, 13 bits result in the ADnDATAx register

Bits 13:8 – TRG2SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

Bits 7:6 – MODE[1:0] Sampling Mode Selection bits

Value	Description
11	Oversampling of multiple samples is defined by the ACCNUM[1:0] bits. The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
10	Integration of multiple samples is defined by: CNTx[15:0] bits (ADnCNTx[15:0]). The first conversion is initiated by the TRG1SRC[4:0] trigger, and all other conversions are executed by the TRG2SRC[4:0] trigger.
01	Window gated by TRG1SRC[4:0] source. In this mode, the samples are accumulated when a signal selected by the TRG1SRC[4:0] bits has an active level. All conversions are initiated by the TRG2SRC[4:0] trigger. The number of conversions is limited by the CNTx[15:0] bits (ADnCNTx[15:0]).
00	Single sample initiated by the TRG1SRC[4:0] trigger.

Bits 5:0 – TRG1SRC[5:0] Multi-Sample Conversions Re-Trigger Source Selection bits

16.3.68. ADC 3 Channel x Control Register 2

Name: AD3CHxCON2
Offset: 0xB9C, 0xBBC, 0xBDC, 0xBFC

Legend: x = channel; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	ACCRO	ACCBRST	CMPVAL	CMPMODE[1:0]			ADCMPSTAT[9:0]	
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	23	22	21	20	19	18	17	16
	ADCMPSTAT[9:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		CRITERIA[2:0]					ADCMPCNT[9:8]	
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	ADCMPCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – ACCRO Accumulator Roll-Over Enable bit
The Roll-Over must be enabled when the ADnACCx register is used.

Value	Description
1	ADnDATAx accumulator is not cleared; it is allowed to roll-over.
0	ADnDATAx accumulator is cleared at the end of a multi-sample sequence.

Bit 30 – ACCBRST Oversampling Burst Mode Enable bit

Value	Description
1	The oversampling is performed as a continuous, non-interruptible burst, during which all other conversion requests will be blocked until the process is completed.
0	Oversampling can be interrupted by a high priority conversion request.

Bit 29 – CMPVAL Comparison Value Selection bit

Value	Description
1	The channel data value in the ADnDATAx register is used for comparison.
0	The immediate conversion value in ADnRESx is used for comparison.

Bits 29:28 – CMPMODE[1:0] Comparison Mode Selection bit

Value	Description
1	Accumulative violation is the basis for the violation count per ADCMPCNT[9:0] bits.
0	Consecutive violation is the basis for the violation count per ADCMPCNT[9:0] bits.

Bits 25:16 – ADCMPSTAT[9:0] Comparison Violation Count Status bits
These read-only register bits display the current violation count value based on CMPVAL and CMPMODE bit settings.

The corresponding CMPxRDY bit is set when the count reaches the value set in ADCMPCNT[9:0].

Bits 14:12 – CRITERIA[2:0] Comparison Criteria Selection bits

Value	Description
111-101	Comparison is disabled.
100	Conversion result is less than or equal to (≤) the ADnCMPLOx register.
011	
010	Conversion result is within bounds of (≥) ADnCMPLOx and (<=) ADnCMPHIx.
001	Conversion result is out of bounds of (<) ADnCMPLOx or (>) ADnCMPHIx.
000	Comparison is disabled.

Bits 9:0 – ADCMPCNT[9:0] Comparison Count bits

Value	Description
1111111111	1023 comparisons matching the criteria are selected.
1111111110	1022 comparisons matching the criteria are selected.
...	
0000000011	3 comparisons matching the criteria are selected.
0000000010	2 comparisons matching the criteria are selected.
0000000001	1 comparison matching the criteria is selected.
0000000000	1 comparison matching the criteria is selected.

16.3.69. ADC 3 Channel x Data Register

Name: AD3CHxDATA
Offset: 0xBA0, 0xBC0, 0xBE0, 0xC00

Legend: x = channel; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DATA7[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA7[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA7[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA7[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DATA7[31:0] Channel Data/Primary Accumulator bits

16.3.70. ADC 3 Channel x Result Register

Name: AD3CHxRES
Offset: 0xBA4, 0xBC4, 0xBE4, 0xC04

Legend: x = channel; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	RESF7[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESF7[3:0]							
Access	R	R	R	R				
Reset	0	0	0	0				

Bit	15	14	13	12	11	10	9	8
					RES7[11:8]			
Access					R	R	R	R
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES7[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – RESF7[11:0] Conversion Result bits

Bits 11:0 – RES7[11:0] Conversion Result bits

16.3.71. ADC 3 Channel x Counter Register

Name: AD3CHxCNT
Offset: 0xBA8, 0xBC8, 0xBE8, 0xC08

Legend: x = channel; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	CNTSTAT7[15:8]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CNTSTAT7[7:0]							
Access	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R	HS/HC/R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CNT7[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT7[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CNTSTAT7[15:0]
Number of conversions done in integration (MODE[1:0] bits = '10') and window (MODE[1:0] bits = '01') sampling modes.

Bits 15:0 – CNT7[15:0]
Number of samples for an integration sampling mode (MODE[1:0] bits = '10') and maximum number of samples for a window sampling mode (MODE[1:0] bits = '01').

16.3.72. ADC 3 Channel x Low Compare Register

Name: AD3CHxCMPLO
Offset: 0xBAC, 0xBCC, 0xBEC, 0xC0C

Legend: x = channel; R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	LO7[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LO7[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LO7[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LO7[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – LO7[31:0] Low Threshold Comparator Value bits

16.3.73. ADC 3 Channel x High Compare Register

Name: AD3CHxCMPHI
Offset: 0xBB0, 0xBD0, 0xBF0, 0xC10

Legend: x = channel, R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	HI7[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HI7[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HI7[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HI7[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HI7[31:0] High Threshold Comparator Value bits

16.4. Operation

The Analog-to-Digital conversion is performed in the following three steps:

1. Sampling of the input signal.
2. Capturing of the input signal (holding) and transferring to the converter.
3. Conversion of the analog signal to its digital representation.

During the sampling of the input signal, the Sample-and-Hold (S/H) circuit capacitor is charged. The sampling time must be adequate so that the capacitor charges to a value equal to the input voltage. At the appropriate time, the input is disconnected from the capacitor, and subsequently, the analog voltage is transferred to the converter. The converter then digitizes the analog signal and provides the result.

16.4.1. Channels

The channel is a group of settings controlling the conversion process. The ADC has up to 48 channels (ADC1 - ADC4 each support 8 channels for a total of 32 channels, and ADC5 supports 16 channels).

All channel settings are placed in the ADnCHxCON1 register (where n is the specific ADC module and x is the respective channel of the ADC).

To resolve simultaneous requests for channel conversions between different channels, the fixed order priority scheme is used. The priority scheme is defined by the channel number, with the channel with the lowest number receiving the highest priority. In other words, channel conversions occur in ascending order of the channel number, with the lowest channel number being converted first.

16.4.2. Analog Inputs

Any analog input (external pin or internal analog signal) can be mapped to any channel by the PINSEL[3:0] bits (ADnCHxCON1[27:24]) for a positive input and by the NINSEL[1:0] bits (ADnCHxCON1[29:28]) for a negative input. The input can be configured as single-ended or differential using the DIFF bit (ADnCHxCON1[31]) (where n is the specific ADC module and x is the respective channel of the ADC).

The ANSELx registers for the I/O ports associated with the analog input are used to configure the corresponding pins as analog pins. A pin is configured as an analog input when the corresponding ANSELx bit = `1`. When the ANSELx bit = `0`, the pin is set to digital control. The ANSELx bits are set when the device comes out of Reset, causing the ADC input pins to be configured as analog inputs by default. The TRISx registers control the digital function of the port pins. The port pin that is required as an analog input must have its corresponding bit set in the specific TRISx register, configuring the pin as an input. If the I/O pin associated with an ADC input is configured as an output by clearing the TRISx bit, the port's digital output level will be converted. After a device Reset, all TRISx bits are set. The PORT register bit reads as '`0`' if its corresponding pin is configured as an analog input. For more information on port pin configuration, refer to "I/O Ports".

16.4.3. Sampling and Conversion Timing

The ADC module is clocked from the Clock Generator 6. This input clock is divided by four to get the analog core clock (T_{AD}). The conversion takes two ADC analog core clock cycles. The input module clock frequency must be in a range between 32 MHz (8 MHz core clock or 4 MSPS conversion rate) and 320 MHz (80 MHz core clock or 40 MSPS conversion rate).

Each channel can be configured for a different sampling time using SAMC[4:0] bits (ADnCHxCON1[20:16]).

16.4.3.1. Sampling Time Requirements

The analog input model of the ADC is illustrated in Figure 16-2.

Time is required to charge the Holding Capacitor (C_{HOLD}) to the input signal level through the source and internal device resistance.

The total acquisition time for the Analog-to-Digital conversion is a function of the Holding Capacitor (C_{HOLD}) charge time. For the ADC module to meet its specified accuracy, the Holding Capacitor (C_{HOLD}) must be allowed to fully charge to the voltage level on the analog input pin. The Signal Source Impedance (R_S) and the Interconnect Impedance (R_{IC}) combine to directly affect the time required to charge the C_{HOLD}. The combined impedance ($R_{TOTAL} = R_S + R_{IC}$) must, therefore, be small enough to fully charge the Holding Capacitor within the selected sample time. To charge the C_{HOLD} with 0.5 LSB error, the sampling time should be more than the time defined by the Minimum Sampling Time equation.

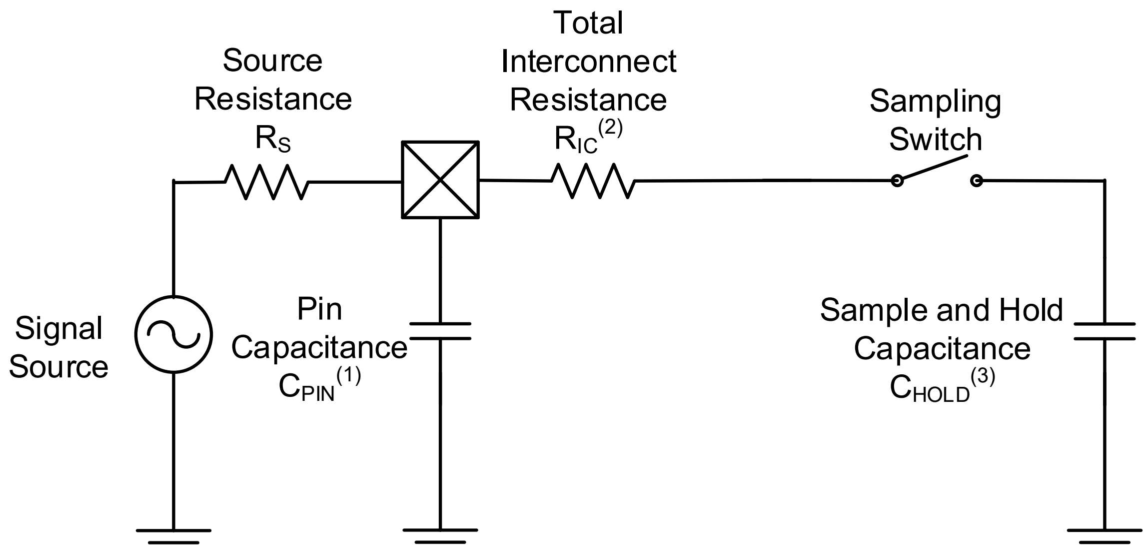
Equation 16-1. Minimum Sampling Time

$T_{SAMPLING} = R_{TOTAL} \times C_{HOLD} \times \ln(2^{(RESOLUTION + 1)})$ or

For 12-Bit Resolution:

$T_{SAMPLING} = 9 \times R_{TOTAL} \times C_{HOLD}$

Figure 16-2. ADC Input Model



Notes:

1. The C_{PIN} value depends on the device package and is not tested. For QFN or TQFP packages, the typical value is about 4 pF.
2. See R_{IC} value in the Electrical Specifications.
3. See C_{HOLD} value in the Electrical Specifications.

16.4.4. Conversions

Each channel can be triggered to do a single conversion or accumulate results of the multiple conversions. The Conversion mode is selected by MODE[1:0] bits (ADnCHxCON1[7:6]). Depending on the Conversion mode, the channel conversions are initiated by triggers defined in TRG1SRC[5:0] and TRG2SRC[5:0] bits (ADnCHxCON1[5:0] and ADnCHxCON1[13:8]).

The following Conversion modes are available:

- **Single conversion (MODE[1:0] bits = '00').** The conversion is initiated by the TRG1SRC[5:0] trigger. When the conversion is finished, the conversion result is stored in the ADnCHxDATA and ADnCHxRES registers, and both CHxRDY and CHxRRDY bits in the ADnSTAT and ADnRSTAT registers are set and the ADnCHxIF interrupt flag in the corresponding IFCx register is set.
- **Window conversions (MODE[1:0] bits = '01').** In this mode, the conversion results are accumulated in the ADnCHxDATA register (primary accumulator) when a signal selected by the TRG1SRC[5:0] bits has an active level. The active level can be changed by the TRG1POL bit (ADnCHxCON1[23]). All conversions are initiated by a trigger selected in the TRG2SRC[5:0] bits (ADnCHxCON1[13:8]). The number of conversions is limited by the CNT[15:0] bits (ADnCHxCNT[15:0]). The number of accumulated conversion results is available in the CNTSTAT[15:0] bits (ADnCHxCNT[31:16]). CHxRDY bit in the ADnSTAT register and the ADnCHxIF interrupt flag in the corresponding IFCx register are set when the gating signal selected by the TRG1SRC[5:0] bits is deactivated, or when the number of accumulated conversion results reaches a limit defined by the CNT[15:0] bits.
- **Integration (MODE[1:0] bits = '10').** In this mode, the number of conversion results defined by the CNT[15:0] bits (ADnCHxCNT[15:0]) is accumulated in the ADnCHxDATA register (primary accumulator). The first conversion is initiated by a trigger selected by the TRG1SRC[5:0] bits, and all subsequent conversions are executed by a trigger selected by TRG2SRC[5:0] bits. When the number of accumulated conversion results reaches a value in the CNT[15:0] bits

(ADnCHxCNT[15:0]), the CHRDY bit in the ADnSTAT register and the ADnCHxIF interrupt flag in the corresponding IFCx register are set.

- **Oversampling (MODE[1:0] bits = ’11’).** In this mode, the number of conversion results defined by the ACCNUM[1:0] bits (ADnCHxCON1[15:14]) are accumulated in the ADnCHxDATA register (primary accumulator). The first conversion is initiated by a trigger selected by the TRG1SRC[5:0] bits, and all subsequent conversions are executed by a trigger selected by the TRG2SRC[5:0] bits. When the number of accumulated conversion results reaches a value specified in the ACCNUM[1:0] bits, the CHxRDY bit in the ADnSTAT register and the ADnCHxIF interrupt flag in the corresponding IFCx register are set. The oversampling process can be delayed by high-priority channel conversions. The ACCBRST bit (ADnCHxCON2[30]) can disable the interruption by other high-priority channels. When the ACCBRST bit is set (=’1’), all other conversions will be suspended until all oversampling conversions for this channel are finished.

16.4.5. Triggers

The channel trigger source is defined in the ADnCHxCON1 register. The trigger source is selected by the TRG1SRC[5:0] and TRG2SRC[5:0] bits.

TRG1SRC[5:0] bits:

- Define a trigger source for a single conversion mode (MODE[1:0] bits = ‘00’).
- Select a signal to enable/gate the TRG2SRC[5:0] triggers in the Window mode (MODE[1:0] bits = ‘01’). The polarity of the TRG1SRC[5:0] signal can be changed by TRG1POL bit.
- Define the start/first trigger source for Integration and Oversampling modes (MODE[1:0] bits = ‘10’ and MODE[1:0] bits = ‘11’).

TRG2SRC[5:0] bits:

- Are not used for a Single Conversion mode (MODE[1:0] bits = ‘00’).
- Select a trigger source for all conversions in the Window mode (MODE[1:0] bits = ‘01’).
- Re-trigger ADC in Integration and Oversampling modes after the first trigger specified by TRG1SRC[5:0] bits (MODE[1:0] bits = ‘10’ and MODE[1:0] bits = ‘11’).

The following types of triggers are available:

- Software
- Back-to-back
- ADC repeat timer
- From other peripheral modules

Software Trigger

The software trigger is used when TRG1SRC[5:0] or TRG2SRC[5:0] bits are set to ‘00001’. Trigger is generated when the corresponding bit in the ADnSWTRG register is set.

Back-to-Back Trigger

The channel is re-triggered immediately after the previous conversion is finished when TRG1SRC[5:0] or TRG2SRC[5:0] bits are set to ‘00010’. The channel conversions are executed back-to-back. The timing is affected (can be delayed) by priorities of other channels.

Repeat Timer Trigger

The channel is triggered from an internal ADC repeat timer when TRG1SRC[5:0] or TRG2SRC[5:0] bits are set to ‘00011’. This timer is clocked from the ADC analog core clock (T_{AD}), and its period is set by RPTCNT[5:0] bits.

Peripheral Modules Triggers

All other TRG1SRC[5:0] and TRG2SRC[5:0] bit settings starting from '00100' select trigger sources from other modules. These trigger options are device specific.

16.4.6. Comparator

Each ADC channel has a dedicated digital comparator that compares each conversion result from ADnCHxRES register or the channel data from ADnCHxDATA register with thresholds stored in the ADnCHxCMPLO and ADnCHxCMPHI registers. The comparator source selection between ADnCHxRES and ADnCHxDATA registers is controlled by CMPVAL bit (ADnCHxCON2[29]). When CMPVAL bit is zero, the ADnCHxRES register is used, and when CMPVAL bit is set, the ADnCHxDATA register is selected for the comparator operation. The following comparison criteria can be set by the CMPMOD[2:0] bits (ADnCHxCON2[14:12]):

- Out of bounds (CMPMOD[2:0] bits = '001') when the comparator data source is less than ADnCHxCMPLO or greater than ADnCHxCMPHI.
- In bounds (CMPMOD[2:0] bits = '010') when the comparator data source is greater or equal to ADnCHxCMPLO and less or equal to ADnCHxCMPHI.
- Greater than (CMPMOD[2:0] bits = '011') when the comparator data source is greater or equal to ADnCHxCMPLO.
- Less or equal (CMPMOD[2:0] bits = '100') when the comparator data source is less or equal to ADnCHxCMPLO.

For all other CMPMOD[2:0] bits options the digital comparator is disabled.

Each channel can be programmed to generate the comparator interrupt upon a set number of the criteria match events. The number of the match events is set by ADCMPCNT[9:0] bits (ADnCHxCON2[9:0]). The ADCMPSTAT[9:0] bits (ADnCHxCON2[25:16]) hold the current number the match events occurred. The comparator can count consecutive or accumulative match events. This is selected by CMPCNTMOD bit (ADnCHxCON2[28]). If the CMPCNTMOD bit is zero, then the comparator will generate an event only when a number of consecutive match events is detected. If the CMPCNTMOD bit is set, then the comparator will accumulate/count the match events and generate the interrupt when the event number will exceed the number programmed in ADCMPCNT[9:0] bits. When the comparison match event is detected, the corresponding channel CMPxFLG bit in ADnCMPSTAT register and the ADnCMPxIF interrupt flag are set.

16.4.7. Interrupts

Each channel has an individual result ready interrupt with a ADnCHxIF flag in the corresponding IFS register. The channel interrupt can be enabled by setting the ADnCHxIE bit in the IEC register. There are two interrupt modes defined by the IRQSEL bit (ADnCHxCON1[21]). If the IRQSEL bit is zero, the channel interrupt is generated after each single conversion when the result is ready in the ADnCHxRES register. If the IRQSEL bit is set, the channel interrupt is generated when data are ready in the ADnCHxDATA register. For the IRQSEL bit = 1 option, the channel interrupt is generated:

- After each conversion for the Single Conversion mode (MODE[1:0] bits = '00').
- For the Window mode (MODE[1:0] bits = `01`), when the number of conversions reaches a value in CNTx[15:0] bits (ADnCHxCNT[15:0]) or when the gate signal defined by TRG1SRC[5:0] bits is deasserted.
- When all conversions are finished for the Oversampling or Integration modes (MODE[1:0] bits = '10' or '11').

The result-ready interrupt can be generated before the result is available in the ADnCHxDATA register. This feature is called "Early Interrupt" and can reduce the ADC channel interrupt latency. This early interrupt for the channel is enabled by the setting of the EIEN bit (ADnCHxCON1[22]). Early interrupts can only be used in Single Conversion mode (MODE[1:0] bits = '00'). When the early interrupt is enabled (EIEN bit = '1'), the channel individual interrupt is generated, and the CHxRDY bit in the ADnSTAT register is set at the start of the sampling time. The software must guarantee that the channel data are ready when the ADnCHxDATA register is read in the interrupt service routine.

Each channel also has a comparator interrupt with an ADnCMPxIF flag in the corresponding IFS register. The channel comparator interrupt is generated upon a set number of criteria match events as defined by the ADCMPCNT[9:0] and CMPMODE bits (see Comparator for details). The comparator interrupt is enabled by setting the ADnCMPxIE bit in the IEC register.

16.4.8. Test Mode

The Test mode allows the ADC controller to be tested. This mode is enabled when the TSTEN bit (ADnCON[8]) is set. When enabled, the result of any conversion is overwritten with a value from the ADnDATAOVR register. The TSTEN bit can be protected from an unintentional write by the setting of the TSTLOCK bit (ADnCON[10]) .

16.4.9. Results Formatting

The result of each single conversion is stored in the ADnCHxRES register. For the Multiple Conversion modes (MODE[1:0] != '00'), the results are added to a primary accumulator (ADnCHxDATA). The result sum in the ADnCHxDATA register is valid only when the corresponding CHxRDY bit is set in the ADnSTAT register.

The result value is formatted using DIFF and FRAC bit settings (ADnCHxCON1[31] and ADnCHxCON1[30]). The format options are explained in Table 16-4.

Table 16-4. Output Format[1]

Differential Format Option DIFF bit	Input Voltage (V_{INP} = Voltage on non-inverting input, V_{INN} = Voltage on inverting input)	Conversion Result			
		FRAC bit = 0		FRAC bit = 1	
		Decimal	Hex	Decimal	Hex
0	V_{INP} = 0	0	0000 0000	0	0000 0000
	$V_{INP} = V_{DD/2}$	+2047	0000 07FF	+2,146,435,072	7FF0 0000
	$V_{INP} \geq V_{DD}$	+4095	0000 0FFF	+4,293,918,720	FFF0 0000
1	$V_{INP} - V_{INN} \leqq -$ $-V_{DD/2}$	-2048	FFFF F800	-2,147,483,648	8000 0000
	$V_{INP} - V_{INN} = -V_{DD/4}$	-1024	FFFF FC00	-1,073,741,824	C000 0000
	$V_{INP} - V_{INN} = 0$	0	0000 0000	0	0000 0000
	$V_{INP} - V_{INN} = V_{DD}/4$	+1023	0000 003FF	+1,072,693,248	3FF0 0000
	$V_{INP} - V_{INN} \geq V_{DD}/4$	+2047	0000 007FF	+2,146,435,072	7FF0 0000

Note:

1. When used in Differential Mode, the absolute maximum voltage difference between positive and negative analog inputs that can be measured by the ADC is equal to $V_{DD}/2$. Voltage differences beyond this range will cause differential ADC conversion results to clamp to the maximum/minimum values of +/- $V_{DD}/2$ V.

16.4.10. Enabling the ADC

The ADC module is enabled when the ON bit (ADnCON[15]) is set. The ON bit should be set only after the module has been configured. When the software sets the ON bit, the hardware requires approximately five ADC clock cycles to begin operation. When the ADC module is enabled, the module will perform an offset calibration cycle (~5,000 ADC clocks). The ADRDY bit (ADnCON[31]) is set by hardware when the ADC is ready for operation.

If the ON bit is cleared (after having been set), all status and ready bits are automatically cleared. The control bits and the result register's contents remain unaffected since it was last programmed/ updated.

The ADC can be in one of three Operation states indicated by the MODE[1:0] bits (ADnCON[25:24]):

- **Off** (MODE[1:0] = '00'): When power of the converter is switched off.

- **Standby** (MODE[1:0] = '01'): When the module is in a Power-Saving mode.
- **Run** (MODE[1:0] = '1X'): When the module is active and ready to convert.

16.4.11. Power-Saving Mode

The power-saving modes, Sleep and Idle, are useful for reducing the conversion noise by minimizing the digital activity of the CPU, buses and other peripherals. To reduce the current consumption when the ADC is idle, the converter can be configured to a Standby mode using STNDBY bit (ADnCON[16]). When the STNDBY bit is set, the ADC enters the Power-Saving mode. The status MODE[1:0] bits (ADnCON[25:24]) are switched to '01' when the ADC is in Standby state.

16.4.12. Filter (Secondary Accumulator)

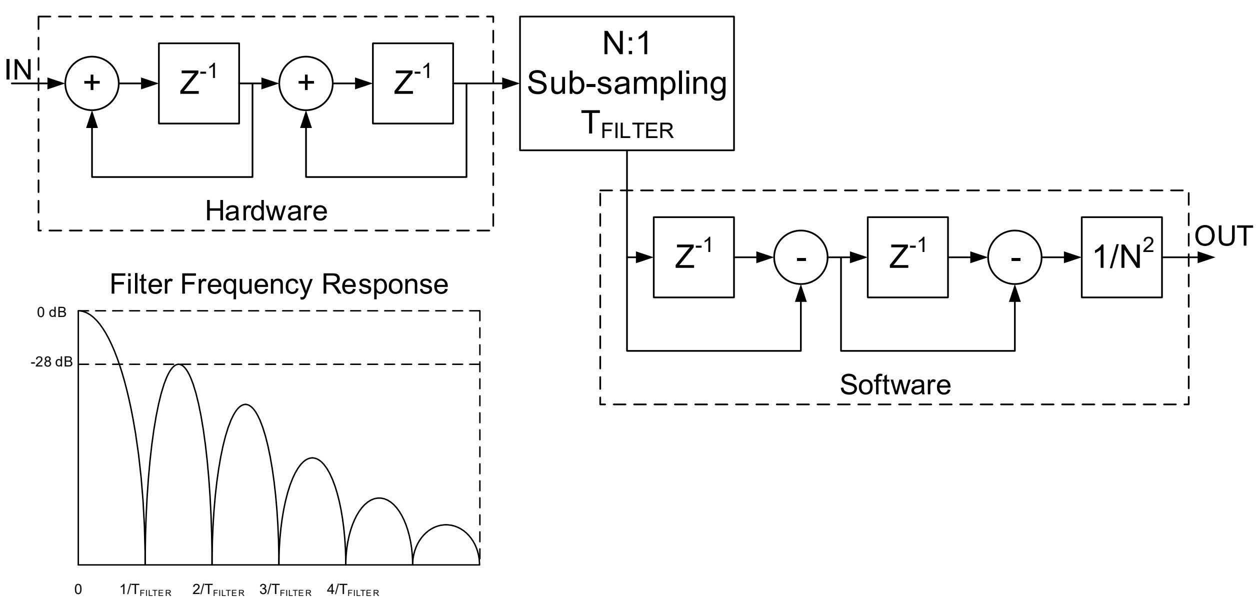
The secondary accumulator is implemented on the last two channels of each ADC, as shown in Table 16-5.

Table 16-5. Channels with Implemented Secondary Accumulators

ADC Number	Channel Number	Secondary Register
1	5	AD1ACC5
	6	AD1ACC6
2	6	AD2ACC6
	7	AD2ACC7
3	10	AD3ACC10
	11	AD3ACC11

The secondary accumulator ADnACCx sums the output of the primary accumulator ADnDATAx. The secondary accumulator is enabled when the ACCRO bit (ADnCHxCON2[31]) is set. If the ACCRO bit = '1', the ADnDATAx and ADnACCx accumulators are not cleared; instead, they will roll over as the data is accumulated over many multi-sample operations. The accumulators function as a Second Order Cascaded-Integrator-Comb filter (CIC). Some of the CIC operations (differentiation functions) need to be performed by the application software as shown in Figure 16-3.

Figure 16-3. Second Order CIC Filter



16.4.13. Calibration

The startup hardware calibration procedure is executed each time the ADC is enabled via setting the ON bit (ADnCON[15]). The startup calibration procedure includes both gain and offset adjustment. The CALRDY bit (ADnCON[30]) status bit provides indication that the process is complete. CALRDY is set by hardware when the calibration cycle is finished, and the hardware clears this bit when the calibration is in progress. The startup calibration takes about 5000 ADC clock cycles.

The ADC offset may drift slightly across temperature. The ADC has hardware to run the offset calibration. The offset calibration procedure takes 14 T_{AD} cycles and can be executed using one of the following methods:

- A software request by setting the CALREQ bit (ADnCON[29]).
- Automatic, periodically. The periodic recalibration is enabled by the ACALEN bit (ADnCON[28]). The time between calibration cycles is set with the CALRATE[1:0] bits from times ranging from 1 second to 1 hour.

The offset calibration has the lowest priority, and it is delayed when a conversion is in progress. The ADC must idle a few ADC clock cycles to start the calibration. This idle time is set by CALCNT[1:0] bits.

16.4.13.1. Software Gain Error Compensation

The start-up hardware calibration precision is limited, and after every ADC enable cycle the gain and offset errors may be different. When more precise and repeatable gain is required, the application can use reference voltages to measure the gain error and apply the corrective coefficient to the results of the conversions. The compensation coefficients can be calculated just once after the start-up calibration. A code example of the procedure is provided in Example 16-4.

16.4.13.1.1. Error Compensation Coefficient Calculation

To calculate the ADC gain error compensation coefficient, two reference voltages are required. The ADC has one selectable input connected to either 1/16 or 15/16 of the AV_{DD} voltage. This selection can be made by setting the VREFSEL bit in the IBIASCON register. When it is assumed that the ADC offset after calibration using the 15/16*AV_{DD} reference selection (IBIASCON.VREFSEL = 1) is zero, Equation 16-2 can be used to calculate the compensation coefficient.

Equation 16-2. Two Reference Voltages Gain Error Compensation Coefficient and Offset Error Calculation

$$Gain\ Error\ Compensation\ Coefficient = \frac{\frac{14}{16} \times 4095}{(\frac{15}{16} \times AVDD\ Result - \frac{1}{16} \times AVDD\ Result)}$$

$$Offset\ Error = \frac{1}{16} \times AVDD\ \ Result - \frac{\frac{1}{16} \times 4095}{Gain\ Error\ Compensation\ Coefficient}$$

16.4.13.1.2. Application of Error Compensation Coefficient

To compensate for the gain error, each ADC result should be multiplied by the calculated coefficient. The compensation can be done as either a fixed-point or floating-point multiplication. The floating-point calculation is shown in Example 16-1.

This operation may take up to 105 nS if the CPU clock is 200 MHz. The multiplication can be accelerated if the fix-point calculation is used as shown in Example 16-2

Example 16-1. ADC Conversion Result Correction Using Floating-Point Calculations

```
float coefficient = 3840.0/adc15div16result; // needed to be done once
……
// takes 21 instruction cycles or 105 nS @ 200 MHz CPU clock
long corrected_result_channel_0 = (long)(coefficient*((float)AD1CH0DATA));
```

Example 16-2. 12-bit Unsigned ADV Conversion Result Correction Using Fixed-Point Calculations

```
unsigned long coefficient = (unsigned long) ((1<<19)*3840.0/
adc15div16result)); // needed to be done once
……
// takes 6 instruction cycles or 30 nS @ 200 MHz CPU clock
unsigned long corrected_result_channel_0 = (coefficient*AD1CH0DATA)>>19;
```

16.5. 16-bit Resolution Mode

Though the ADC SAR core is 12-bit, the resolution of the ADC can be increased to 16-bit using an oversampling technique. The ADC can accumulate 256 measurements, which provides an additional 4 bits to the resolution. The 16-bit mode is enabled when the MODE bits in the ADxCHyCON register are set to 3 (oversampling of 256 samples option). The 16-bit result is available in the ADxCHyDATA register. When all 256 conversions are accumulated and the result is in the ADxCHyDATA register, the CHyRDY bit is set in the ADxSTAT register. This CHyRDY flag is cleared by hardware when the ADxCHyDATA register is read. The interrupt mode selection bit IRQSEL in the ADxCHyCON register must be set (= '1') to detect when the 16-bit data in the ADxCHyDATA register is ready using a channel interrupt (ADxCHyIF bit in the corresponding IFS register). When the ADC operates at the maximum conversion speed (40 MSPS), the 16-bit mode allows converting at 40MSPS/256 = 156 kSPS. The maximum 16-bit result conversion rate of 156 kSPS is achieved when the ADC module is clocked from 320 MHz and the sampling time is set to minimum (0.5TAD, SAMC bits in the ADxCHyCON register are zero).

16.5.1. 16-bit Conversion Example

In Example 16-3, the conversions are started by a software trigger and continued by back-to-back triggers. The number of accumulated conversion results is set to 256 to get the 16-bit result. The oversampling process cannot be interrupted by other channel conversions because the ACCBRST bit is set.

Example 16-3. 16-bit Conversion

```
#include <xc.h>
// The channel output.
long result16bit = 0;
int main(){

clock_ADC_for_40Msps_from_PLL2();

// In this example channel 0 will be used.
// Oversampling conversion mode.
AD3CH0CON1bits.MODE = 3;
// Set number of conversions accumulated to 256/16bit result.
AD3CH0CON1bits.ACCNUM = 3;
// The oversampling if started cannot be interrupted
// by a high priority channels conversion request.
AD3CH0CON2bits.ACCBRST = 1;
// Software trigger will start conversions.
AD3CH0CON1bits.TRG1SRC = 1;
// Re-trigger back to back.
AD3CH0CON1bits.TRG2SRC = 2;
// Select the AN0 analog positive input/pin.
AD3CH0CON1bits.PINSEL = 0;
// Interrupt when oversampling is done.
AD3CH0CON1bits.IRQSEL = 1;
```

```
// Select signal sampling time to 6.5 TADs.
AD3CH0CON1bits.SAMC = 3;
// Enable ADC.
AD3CONbits.ON = 1;
// Wait when ADC will be ready/calibrated.
while(AD3CONbits.ADRDY == 0);
// Trigger channel #0 in software and wait for the 256 samples
// 16 bit oversampling result.
while(1){
// Trigger channel # 0.
AD3SWTRGbits.CH0TRG = 1;
// Wait for a conversion ready flag.
while(AD3STATbits.CH0RDY == 0);
// Read oversampling result. It will clear the conversion ready flag.
result16bit = AD3CH0DATA;
}
return 1;
}
```

16.6. Application Examples

16.6.1. Gain Error Calibration

Example 16-4. Gain Error Calibration Example

```
#include <xc.h>

// The channel output.
long result = 0;
// Gain compensation coefficient.
long coefficient;
long result1div16;
long result15div16;

// Oscillator initialization procedure.
void OscillatorInitialization();
int main(){

    // Initialize the oscillator.
    // Clock generator 6 should provide 320MHZ to the ADCs.
    OscillatorInitialization();

    // Enable ADC.
    AD1CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while(AD1CONbits.ADRDY == 0);
    ////////////////////////////////////////////////////////////////////
    // GET A COEFFICIENT FOR THE GAIN ERROR COMPENSATION
    ////////////////////////////////////////////////////////////////////
    // Select oversampling mode.
    AD1CH1CON1bits.MODE = 3;
    // 256 conversions
    AD1CH1CON1bits.ACCNUM = 3;
    // Software trigger will start a conversion.
    AD1CH1CON1bits.TRG1SRC = 1;
    // Back-to-back conversions
    AD1CH1CON1bits.TRG2SRC = 2;
    // Select the AN8 input which is connected to 1/16 of AVDD
    IBIASCONbits.VREFSEL = 0;
    AD1CH1CON1bits.PINSEL = 8;
    // Select signal sampling time
    AD1CH1CON1bits.SAMC = 7;
    // Flag when all conversions done
    AD1CH1CON1bits.IRQSEL = 1;
    // Average 256 results of the reference voltage
    AD1SWTRGbits.CH1TRG = 1;
    // Wait when the result is ready
    while(AD1STATbits.CH1RDY == 0);
    result1div16 = AD1CH1DATA;
    // Select the AN8 input which is connected to 15/16 of AVDD
    IBIASCONbits.VREFSEL = 1;
    AD1CH1CON1bits.PINSEL = 8;
    // Average 256 results of the reference voltage
```

```
    AD1SWTRGbits.CH1TRG = 1;
    // Wait when the result is ready
    while(AD1STATbits.CH1RDY == 0);
    result15div16 = AD1CH1DATA;

    // Oversampling result is 16 Bit (has additional 4 bits).
    // Calculate the gain compensation coefficient.
    // The coefficient is in fixed-point format (18 bits before point).
    coefficient = (long)(3584.0*16.0*(1<<18)/(result15div16-result1div16));

    ////////////////////////////////////////////////////////////////////
    // CONVERT AND COMPENSATE THE GAIN ERROR
    ////////////////////////////////////////////////////////////////////
    // Clean channel register for new settings.
    AD1CH1CON1 = 0;
    // Select single conversion mode.
    AD1CH1CON1bits.MODE = 0;
    // Software trigger will start a conversion.
    AD1CH1CON1bits.TRG1SRC = 1;
    // Select the AN7 input for conversions
    AD1CH1CON1bits.PINSEL = 7;
    // Select signal sampling time
    AD1CH1CON1bits.SAMC = 3;
    // Trigger channel #1 in software and wait for the result.
    while(1){
        // Trigger channel # 1.
        AD1SWTRGbits.CH1TRG = 1;
        // Wait for a conversion ready flag.
        while(AD1STATbits.CH1RDY == 0);
        // Read result. It will clear the conversion ready flag.
        // The gain error correction coefficient is in fixed-point format (18
bits before point).
        result = (coefficient*AD1CH1DATA)>>18;
        _LATC3 = 0;
//            printf("\nCompensated Gain Error:%08x \r\n\n", result);
    }
    return 1;
}
void OscillatorInitialization(){
    PLL1CONbits.ON = 1;
    OSCCTRLbits.PLL1EN = 1;
    while(OSCCTRLbits.PLL1RDY == 0);
    PLL1CONbits.FSCMEN = 0; // disable clock fail monitor
    VCO1DIVbits.INTDIV = 1; // 1:2 = 320MHz
    PLL1DIVbits.PLLFBDIV = 80; // VCO = 640 MHz
    PLL1DIVbits.PLLPRE = 1;
    PLL1DIVbits.POSTDIV1 = 4;
    PLL1DIVbits.POSTDIV2 = 1;
    PLL1CONbits.DIVSWEN = 1;
    while(PLL1CONbits.DIVSWEN == 1);
    PLL1CONbits.NOSC = 1; // FRC
    PLL1CONbits.OSWEN = 1;
    while(PLL1CONbits.OSWEN == 1);
    PLL1CONbits.FOUTSWEN = 1;
    while(PLL1CONbits.FOUTSWEN == 1);
    PLL1CONbits.PLLSWEN = 1;
    while(PLL1CONbits.PLLSWEN == 1);
    while(PLL1CONbits.CLKRDY == 0);
    CLK1CONbits.NOSC = 5; // PLL1
    CLK1CONbits.OSWEN = 1;
    while(CLK1CONbits.OSWEN == 1);
    while(CLK1CONbits.CLKRDY == 0);
    // ADC high speed clock (Generator 6), should be 320 MHz for 80MHz
operation
    CLK6CONbits.ON = 1;
    CLK6CONbits.NOSC = 7; // PLL1 VCO divider
    CLK6CONbits.OSWEN = 1;
    while(CLK6CONbits.OSWEN == 1);
    while(CLK6CONbits.CLKRDY == 0);
}
```

16.6.2. Single Conversion

In Example 16-5, a software trigger is used to start a single conversion.

Example 16-5. Single Conversion Example

```
#include <xc.h>

// The channel output
long result = 0;

int main()
{
    // Set up clock for 40Msps operation
    clock_ADC_for_40Msps_from_PLL2();

    // In this example channel 1 will be used.
    // Software trigger will start a conversion.
    AD3CH0CON1bits.TRG1SRC = 1;
    // Use a single ended input.
    AD3CH0CON1bits.DIFF = 0;
    // Select the AN0 analog positive input/pin for the signal.
    AD3CH0CON1bits.PINSEL = 0;
    // Select the ANN0 (Vss) analog negative input/pin for the signal.
    AD3CH0CON1bits.NINSEL = 0;
    // Select signal sampling time (6.5 TADs = 81nS).
    AD3CH0CON1bits.SAMC = 3;
    // Enable ADC.
    AD3CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while (AD3CONbits.ADRDY == 0);
    // Trigger channel #1 in software and wait for the result.
    while (1) {
        // Trigger channel # 1.
        AD3SWTRGbits.CH0TRG = 1;
        // Wait for a conversion ready flag.
        while (AD3STATbits.CH0RDY == 0);
        // Read result. It will clear the conversion ready flag.
        result = AD3CH0DATA;
    }
    return 1;
}
```

16.6.3. Windowed Multiple Conversions

In Example 16-6, the conversion results are accumulated until the RD6 pin is at a high level. The conversions are triggered by the internal ADC timer.

Example 16-6. Windowed Conversions Example

```
#include <xc.h>

// The channel output from primary accumulator.
volatile long result = 0;
// The number of accumulated samples.
volatile long number_of_accumulated_samples;


int main() {
    _TRISC8 = 0;

    //Set up clock for 40Msps operation
    clock_ADC_for_40Msps_from_PLL2();
    // RD6/RP55 pin is a trigger input.
    // Make it a digital input
    _ANSELD6 = 0;
    _TRISD6 = 1;
        // Map external pin trigger to RD6/RP55
    _ADTRG31R = 55;
    // In this example channel 0 will be used.
    // Set limit for the accumulated samples number.
    AD3CH0CNTbits.CNT = 0xffff;
    // Window conversion mode.
    AD3CH0CON1bits.MODE = 1;
    // Accumulation will be started/stopped from an external pin.
    AD3CH0CON1bits.TRG1SRC = 31;
    // Logic LOW on RD6/RP55 will trigger conversion
    AD3CH0CON1bits.TRG1POL = 1;
    // Trigger all conversions from the ADC repeat timer.
    AD3CH0CON1bits.TRG2SRC = 3;
    // Select the AN0 analog positive input/pin.
    AD3CH0CON1bits.PINSEL = 0;
    // Select signal sampling time (6.5 TADs = 81nS).
    AD3CH0CON1bits.SAMC = 3;
    // Set period of the triggers timer (63 is maximum).
    AD3CONbits.RPTCNT = 60;
```

```
    // Interrupt when AD3CH0DATA is updated
    AD3CH0CON1bits.IRQSEL = 1;
    // Enable ADC.
    AD3CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while (AD3CONbits.ADRDY == 0);
    // Enable interrupt;
    _AD3CH0IF = 0;
    _AD3CH0IE = 1;
    // Channel 0 is converted and results are accumulated until the RD6 pin is high.
    // On transition from high to low an interrupt is generated.
    while (1);
    return 1;
}

void __attribute__((interrupt)) _AD3CH0Interrupt(){
    _LATC8 ^= 1;

    // Read result in accumulator and clear CH3RDY flag.
    result = AD3CH0DATA;
    number_of_accumulated_samples = AD3CH0CNTbits.CNTSTAT;
    result /= number_of_accumulated_samples;

    // Clear interrupt flag.
    _AD3CH0IF = 0;
}
```

16.6.4. Integration of the Multiple Samples

In Example 16-7, the conversions are started by a software trigger and continued by back-to-back triggers until the number of conversions is less than the value set in the AD3CH0CNT register.

Example 16-7. Integration of the Multiple Samples Example

```
#include <xc.h>

int main(){

    //Set up clock for 40Msps operation
    clock_ADC_for_40Msps_from_PLL2();

    _TRISC8 = 0;

    // In this example channel 15 will be used.
    // Set number of conversions accumulated to 123.
    AD3CH0CNT = 123;
    // Software trigger will start a conversions.
    AD3CH0CON1bits.TRG1SRC = 1;
    // Re-trigger back to back.
    AD3CH0CON1bits.TRG2SRC = 2;
    // Select the AN0 analog positive input/pin.
    AD3CH0CON1bits.PINSEL = 0;
    // Select signal sampling time (6.5 TADs = 81nS).
    AD3CH0CON1bits.SAMC = 3;
    // Enable ADC.
    AD3CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while(AD3CONbits.ADRDY == 0);
    // Trigger channel #15 in software and wait for the 123 samples
    // accumulated result.
    while(1){
       // Trigger channel # 0.
       AD3SWTRGbits.CH0TRG = 1;
       // Wait for a conversion ready flag.
       while(AD3STATbits.CH0RDY == 0);
       // Read oversampling result. It will clear the conversion ready flag.
       result = AD3CH0DATA;

       _LATC8 ^= 1;
    }
    return 1;
}
```

16.6.5. Oversampling

In Example 16-8, the conversions are started by a software trigger and continued by back-to-back triggers. The number of accumulated conversion results is set to 16. The oversampling process cannot be interrupted by other channel conversions because the ACCBRST bit is set.

Example 16-8. Oversampling Example

```
#include <xc.h>

// The channel output.
long result = 0;

int main(){

    //Set up clock for 40Msps operation
    clock_ADC_for_40Msps_from_PLL2();

    _TRISC8 = 0;

    // In this example channel 0 will be used.
    // Oversampling conversion mode.
    AD3CH0CON1bits.MODE = 3;
    // Set number of conversions accumulated to 16.
    AD3CH0CON1bits.ACCNUM = 1;
    // The oversampling if started cannot be interrupted
    // by a high priority channels conversion requests.
    AD3CH0CON2bits.ACCBRST = 1;
    // Software trigger will start a conversions.
    AD3CH0CON1bits.TRG1SRC = 1;
    // Re-trigger back to back.
    AD3CH0CON1bits.TRG2SRC = 2;
    // Select the AN0 analog positive input/pin.
    AD3CH0CON1bits.PINSEL = 0;
    // Select signal sampling time (6.5 TADs = 81nS).
    AD3CH0CON1bits.SAMC = 3;
    // Enable ADC.
    AD3CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while(AD3CONbits.ADRDY == 0);
    // Trigger channel #0 in software and wait for the 16 samples
    // oversampling result.

    while(1){
        // Trigger channel # 0.
        AD3SWTRGbits.CH0TRG = 1;
        // Wait for a conversion ready flag.
        while(AD3STATbits.CH0RDY == 0);
        // Read oversampling result. It will clear the conversion ready flag.
        result = AD3CH0DATA;

        _LATC8 ^= 1;
    }
    return 1;
}
```

16.6.6. Channel Comparator

In Example 16-9, the comparator interrupt is generated each time the conversion result is outside the window.

Example 16-9. Comparator Example

```
#include <xc.h>

// The channel output.
long result = 0;

int main(){

    //Set up clock for 40Msps operation
    clock_ADC_for_40Msps_from_PLL2();

    _TRISC8 = 0;

    // In this example channel 0 will be used.
    // Select single conversion mode.
    AD3CH0CON1bits.MODE = 0;
    // Software trigger will start a conversion.
    AD3CH0CON1bits.TRG1SRC = 1;
    // Select the AN0 analog positive input/pin for the signal.
    AD3CH0CON1bits.PINSEL = 0;
    // Select signal sampling time (6.5 TADs = 81nS).
    AD3CH0CON1bits.SAMC = 3;
    // Enable the comparator for this channel.
```

```
    // Use channel data value in ADnDATAx register for comparison
    AD3CH0CON2bits.CMPVAL = 1;
    // Select out of bounds mode.
    AD3CH0CON2bits.CMPMOD = 1;
    // 1 comparison matching the criteria will trigger comparator event
    AD3CH0CON2bits.ADCMPCNT = 1;
    // Select low limit. To generate comparator event when AD3CH0DATA < 1024.
    AD3CH0CMPLO = 1024;
    // Select high limit. To generate comparator event when AD3CH0DATA > 3072.
    AD3CH0CMPHI = 3072;
    // Enable comparator interrupt.
    _AD3CMP0IE = 1;

    // Set ADC to RUN mode.
    AD3CONbits.MODE = 2;
    // Enable ADC.
    AD3CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while(AD3CONbits.ADRDY == 0);
    // Trigger channel #0 in software and wait for the result.
    while(1) {
        _LATC8 = 0;
        // Trigger channel # 0.
        AD3SWTRGbits.CH0TRG = 1;
        // Wait for a conversion ready flag.
        while(AD3STATbits.CH0RDY == 0);
        // Read result. It will clear the conversion ready flag.
        result = AD3CH0DATA;
    }
    return 1;
}

void __attribute__((interrupt)) _AD3CMP0Interrupt(){
    // Process the comparator event here.
    // Clear the comparator event flag.
    AD3CMPSTATbits.CH0FLG = 0;
    // Clear the comparator flag.
    _AD3CMP0IF = 0;

    _LATC8 ^= 1;
}
```

16.6.7. Multiple Channels Scan

In Example 16-10, three channels are scanned. To scan these channels, they are triggered by the same trigger source.

Example 16-10. Multiple Channels Scan Example

```
#include <xc.h>
volatile long channel_2_an1;
volatile long channel_4_an2;
volatile long channel_6_an3;

// Define peripheral clock frequency.
#define FCY (4000000UL)             // 4MHz
// Define the CCP1 timer frequency.
#define TIMER_FREQUENCY (100UL)       // 1kHz

int main(){

   //Set up clock for 40Msps operation
   clock_ADC_for_40Msps_from_PLL2();
   _TRISC8 = 0;

   // In this example channels ## 2, 4 and 6 will be scanned.
   // To scan channels they must be triggered from one source.
   // The channel with lowest number (#2) will be converted first.
   // The channel with highest number (#6) will be converted last.
   // CHANNEL 2
   // Single conversion mode.
   // CCP1 Timer starts conversion (same for all scanned channels).
   AD1CH2CON1bits.TRG1SRC = 0b010011;
   // Select the AN1 analog input/pin for the channel #2.
```

```
    AD1CH2CON1bits.PINSEL = 1;
    // Select signal sampling time (6.5 TADs = 81nS).
    AD1CH2CON1bits.SAMC = 3;

    // CHANNEL 4
    // Single conversion mode.
    // CCP1 Timer starts conversion (same for all scanned channels).
    AD1CH4CON1bits.TRG1SRC = 0b010011;
    // Select the AN2 analog input/pin for the channel #4.
    AD1CH4CON1bits.PINSEL = 2;
    // Select signal sampling time (8.5 TADs = 106nS).
    AD1CH4CON1bits.SAMC = 4;

    // CHANNEL 6
    // Single conversion mode.
    // CCP1 Timer starts conversion (same for all scanned channels).
    AD1CH6CON1bits.TRG1SRC = 0b010011;
    // Select the AN3 analog input/pin for the channel #6.
    AD1CH6CON1bits.PINSEL = 3;
    // Select signal sampling time (10.5 TADs = 131nS).
    AD1CH6CON1bits.SAMC = 5;

    // Set ADC to RUN mode.
    // Enable ADC.
    AD1CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while(AD1CONbits.ADRDY == 0);

    // Configure CCP1 Timer to trigger all channels (to scan).
    CCP1CON1bits.MOD = 0;
    // Set 32-bit timer.
    CCP1CON1bits.T32 = 1;
    // Set period.
    CCP1PR = FCY/TIMER_FREQUENCY;
    // Run timer.
    CCP1CON1bits.ON = 1;

    // Enable channel # 6 interrupt.
    // This channel is processed last and all other channels results
    // will be ready in the channel #6 ISR.
    _AD1CH6IE = 1;

    while(1);

    return 1;
}
// Channel # 6 interrupt (processed last). All channels
// results in the scan are available here.
void __attribute__((interrupt)) _AD1CH6Interrupt(){
_LATC8 ^= 1;
channel_2_an1 = AD1CH2DATA;
channel_4_an2 = AD1CH4DATA;
channel_6_an3 = AD1CH6DATA;
_AD1CH6IF = 0;
}
```

16.6.8. Channel Filter (Secondary Accumulator)

In Example 16-11, the second order low-pass filter is implemented using the second accumulator.

Example 16-11. Second Order Low Pass Filter Example

```
#include <xc.h>
// VARIABLES OF THE SOFTWARE PART OF THE FILTER.
// These global variables are used in an interrupt.
// That's why they must be declared as "volatile".
// Input for the software part of the filter's first stage.
volatile long ch6_current_1 = 0;
// Input delayed for the first stage.
volatile long ch6_previous_1 = 0;
// Input for the software part of the filter's second stage.
volatile long ch6_current_2 = 0;
// Input delayed for the second stage.
volatile long ch6_previous_2 = 0;
```

```
// The filter output.
volatile long filtered_result = 0;
// Define peripheral clock frequency.
#define FCY (4000000UL) // 4MHz
// Define the CCP1 timer frequency.
#define TIMER_FREQUENCY (1000UL) // 1kHz

int main() {
    //Set up clock for 40Msps operation
    clock_ADC_for_40Msps_from_PLL2();
    _TRISC8 = 0;

    // The device has 2 channels with the secondary
    // accumulator implemented: ## 5 and 6.
    // This example will use channel #6.
    // Enable accumulators roll-over to enable the secondary accumulator.
    AD1CH6CON2bits.ACCRO = 1;
    // Select integration sampling mode.
    AD1CH6CON1bits.MODE = 2;
    // CCP1 Timer starts conversions (1kHz frequency).
    AD1CH6CON1bits.TRG1SRC = 0b010011;
    // CCP1 Timer re-triggers (1kHz frequency).
    AD1CH6CON1bits.TRG2SRC = 0b010011;
    // Select the AN1 analog input/pin for the signal to be filtered.
    AD1CH6CON1bits.PINSEL = 1;
    // Select signal sampling time (6.5 TADs = 81nS).
    AD1CH6CON1bits.SAMC = 3;
    // Set number of conversions = 8 for the filter (sub-sampler).
    // The CH6RDY bit will be set after 8 conversions.
    // The conversions frequency is 1kHz defined by CCP1 Timer period.
    // The signal maximum frequency is in twice less = 500 Hz.
    // The filter cut-off frequency is 500Hz/8 = 62.5 Hz.
    AD1CH6CNT = 8;
    // Interrupt when AD1CH6DATA is updated
    AD1CH6CON1bits.IRQSEL = 1;
    // Enable ADC.
    AD1CONbits.ON = 1;
    // Wait when ADC will be ready/calibrated.
    while (AD1CONbits.ADRDY == 0);
    // Configure CCP1 Timer to trigger the channel # 6.
    CCP1CON1bits.MOD = 0;
    // Set 32-bit timer.
    CCP1CON1bits.T32 = 1;
    // Set period.
    CCP1PR = FCY / TIMER_FREQUENCY;
    // Run timer.
    CCP1CON1bits.ON = 1;
    // Enable channel # 6 interrupt.
    _AD1CH6IE = 1;
    // The AN1 pin filtered result is available in the channel # 6 interrupt.
    while (1);
    return 1;
}

// Channel # 6 interrupt.
// Called when integration is finished (every AD1CH6CNT = 8 conversions).
void __attribute__((interrupt)) _AD1CH6Interrupt() {
    long primary_accumulator;
    // Clear interrupt flag. If the interrupt is persistent then
    // to clear the flag it is required to read the ADC channel
    // result register first.
    primary_accumulator = AD1CH6DATA;
    _AD1CH6IF = 0;
    // Process software part of the filter.
    ch6_current_1 = AD1CH6ACC;
    ch6_current_2 = ch6_previous_1 - ch6_current_1;
    ch6_previous_1 = ch6_current_1;
    filtered_result = ch6_previous_2 - ch6_current_2;
    ch6_previous_2 = ch6_current_2;
    // Divide by 1:(8*8) or 1:64 or shift right by 6
    filtered_result >>= 6;

    _LATC8 ^= 1;
}
```

16.7. Effects of Reset

Following any Reset event, all the ADC control and status registers are reset to their default values with the control bits in a non-active state. This disables the ADC module and sets the analog input pins to Analog Input mode. Any conversion that was in progress will be terminated, and the result will not be written to the result buffer. The values in the ADnCHxDATA and ADnCHxRES registers are initialized to 0000h during a device Reset.

17. Integrated Touch Controller (ITC)

The Integrated Touch Controller (ITC) allows measuring the capacitances of the capacitive sensors connected to CVDANx pins. The sensor's capacitance is processed using a Capacitive Voltage Divider algorithm (CVD). The Integrated Touch Controller CVD procedure is available/predefined in hardware and can be programmed in software-defined Acquisition and Post-Processing/Math sequences.

The Integrated Touch Controller has the following features:

- Up to 32 CVD Records. Each Record can be Assigned to Any Analog Input.
- Up to Three CVD Lists. Each List:
  - Joins a group of the records.
  - Has a balance timer selection the same for all records in the list.
  - Has a processing trigger selection to scan (process) all list records.
  - Can compare the records' results with predefined thresholds (digital comparator).
  - Has a programmable data acquisition sequencer and a post-processing sequencer.
- Data Acquisition Sequencer:
  - Supports up to 16 steps/commands.
  - Controls charge/discharge processes for CVD algorithm implementation.
  - Controls CVD balance time and conversion.
  - Can drive high/low or tri-state the analog inputs and digital TX pins and CVD guards.
- Post-Processing (Math) Sequencer:
  - Supports up to 16 steps/commands.
  - Can add or subtract the conversion results.
  - Has several accumulators to store intermediate data.

17.1. Device-Specific Information

Table 17-1. ITC Summary Table

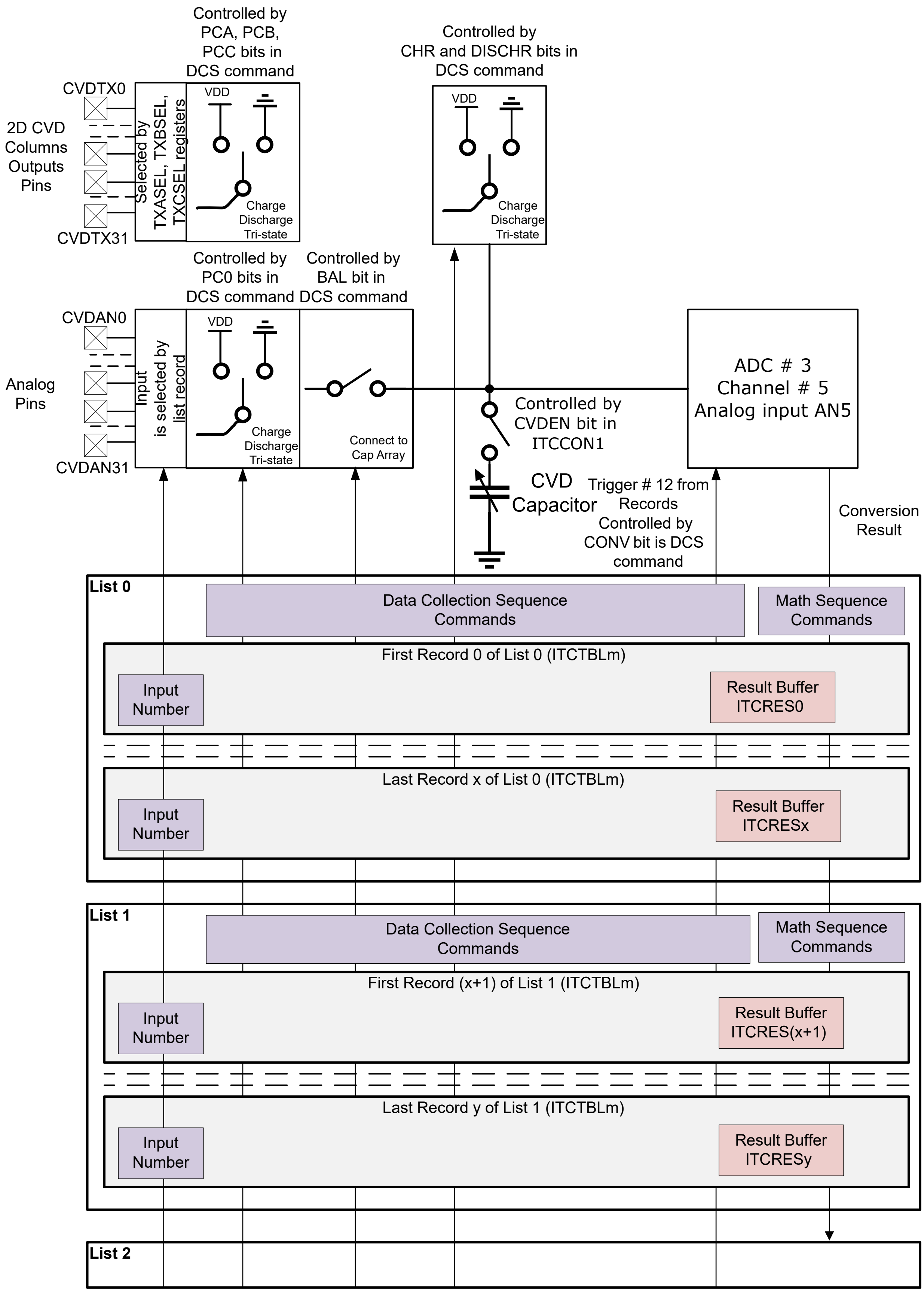
ITC Module Instances	Peripheral Bus Speed
1	Fast (1:1 of CPU clock)

17.2. Architectural Overview

A simplified block diagram of the ITC is illustrated in Figure 17-1.

The ITC has a number of records. These records can be assigned to the analog inputs for processing, and then the records are joined in groups called lists. Each list allows selecting the charge balance time, trigger and some other parameters. The list triggers execute the conversion or CVD procedures on the CVDANx pins defined in the list's records.

Figure 17-1. ITC Block Diagram



17.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0380	ITCCON1	31:24								
		23:16								
		15:8	ON				CVDEN			SIGN
		7:0								
0x0384	ITCCON2	31:24	TMRPR[15:8]							
		23:16	TMRPR[7:0]							
		15:8						TRGEN2	TRGEN1	TRGEN0
		7:0								
0x0388	ITCSTAT	31:24	TSTDATA[15:8]							
		23:16	TSTDATA[7:0]							
		15:8	TSTEN					DRDY		
		7:0						INT2	INT1	INT0
0x038C	ITCTXA	31:24	TXA[31:24]							
		23:16	TXA[23:16]							
		15:8	TXA[15:8]							
		7:0	TXA[7:0]							
0x0390	ITCTXB	31:24	TXB[31:24]							
		23:16	TXB[23:16]							
		15:8	TXB[15:8]							
		7:0	TXB[7:0]							
0x0394	ITCTXC	31:24	TXC[31:24]							
		23:16	TXC[23:16]							
		15:8	TXC[15:8]							
		7:0	TXC[7:0]							
0x0398	ITCHIT	31:24	HIT[31:24]							
		23:16	HIT[23:16]							
		15:8	HIT[15:8]							
		7:0	HIT[7:0]							
0x039C	ITCLS0CON	31:24	MODE[2:0]			WM[1:0]		CM[2:0]		
		23:16	DMAEN		MULEN	SAMC[4:0]				
		15:8	TRGEN	SAMP	TRGCLR	SSRC[4:0]				
		7:0			RECCNT[5:0]					
0x03A0	ITCLS0STAT	31:24	TACT	BUSY						
		23:16			INT					
		15:8								
		7:0			NEXT[5:0]					
0x03A4	ITCLS0CMPHI	31:24	CMPHIGH[31:24]							
		23:16	CMPHIGH[23:16]							
		15:8	CMPHIGH[15:8]							
		7:0	CMPHIGH[7:0]							
0x03A8	ITCLS0CMPLO	31:24	CMPLOW[31:24]							
		23:16	CMPLOW[23:16]							
		15:8	CMPLOW[15:8]							
		7:0	CMPLOW[7:0]							
0x03AC	ITCLS0MUL	31:24	MUL[31:24]							
		23:16	MUL[23:16]							
		15:8	MUL[15:8]							
		7:0	MUL[7:0]							
0x03B0	ITCLS0SEQ	31:24		CVDCAP[2:0]						
		23:16	DATASEQ[2:0]							
		15:8	ACCMODE							
		7:0					ACCCNT[3:0]			
0x03B4	ITCLS0TMR	31:24	TMRD[7:0]							
		23:16	TMRC[7:0]							
		15:8	TMRB[7:0]							
		7:0	TMRA[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x03B8	ITCLS1CON	31:24	MODE[2:0]			WM[1:0]		CM[2:0]		
		23:16	DMAEN		MULEN	SAMC[4:0]				
		15:8	TRGEN	SAMP	TRGCLR	SSRC[4:0]				
		7:0			RECCNT[5:0]					
0x03BC	ITCLS1STAT	31:24	TACT	BUSY						
		23:16			INT					
		15:8								
		7:0			NEXT[5:0]					
0x03C0	ITCLS1CMPHI	31:24	CMPHIGH[31:24]							
		23:16	CMPHIGH[23:16]							
		15:8	CMPHIGH[15:8]							
		7:0	CMPHIGH[7:0]							
0x03C4	ITCLS1CMPLO	31:24	CMPLOW[31:24]							
		23:16	CMPLOW[23:16]							
		15:8	CMPLOW[15:8]							
		7:0	CMPLOW[7:0]							
0x03C8	ITCLS1MUL	31:24	MUL[31:24]							
		23:16	MUL[23:16]							
		15:8	MUL[15:8]							
		7:0	MUL[7:0]							
0x03CC	ITCLS1SEQ	31:24		CVDCAP[2:0]						
		23:16	DATASEQ[2:0]							
		15:8	ACCMODE							
		7:0					ACCCNT[3:0]			
0x03D0	ITCLS1TMR	31:24	TMRD[7:0]							
		23:16	TMRC[7:0]							
		15:8	TMRB[7:0]							
		7:0	TMRA[7:0]							
0x03D4	ITCLS2CON	31:24	MODE[2:0]			WM[1:0]		CM[2:0]		
		23:16	DMAEN		MULEN	SAMC[4:0]				
		15:8	TRGEN	SAMP	TRGCLR	SSRC[4:0]				
		7:0			RECCNT[5:0]					
0x03D8	ITCLS2STAT	31:24	TACT	BUSY						
		23:16			INT					
		15:8								
		7:0			NEXT[5:0]					
0x03DC	ITCLS2CMPHI	31:24	CMPHIGH[31:24]							
		23:16	CMPHIGH[23:16]							
		15:8	CMPHIGH[15:8]							
		7:0	CMPHIGH[7:0]							
0x03E0	ITCLS2CMPLO	31:24	CMPLOW[31:24]							
		23:16	CMPLOW[23:16]							
		15:8	CMPLOW[15:8]							
		7:0	CMPLOW[7:0]							
0x03E4	ITCLS2MUL	31:24	MUL[31:24]							
		23:16	MUL[23:16]							
		15:8	MUL[15:8]							
		7:0	MUL[7:0]							
0x03E8	ITCLS2SEQ	31:24		CVDCAP[2:0]						
		23:16	DATASEQ[2:0]							
		15:8	ACCMODE							
		7:0					ACCCNT[3:0]			
0x03EC	ITCLS2TMR	31:24	TMRD[7:0]							
		23:16	TMRC[7:0]							
		15:8	TMRB[7:0]							
		7:0	TMRA[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x03F0	ITCREC0	31:24		GRDB(0+1)[1:0]		GRDA(0+1)[1:0]				ACCDONE(0+1)
		23:16		PIN(0+1)[6:0]						
		15:8		GRDB0[1:0]		GRDA0[1:0]				ACCDONE0
		7:0		PIN0[6:0]						
0x03F4	ITCREC1	31:24		GRDB(1+1)[1:0]		GRDA(1+1)[1:0]				ACCDONE(1+1)
		23:16		PIN(1+1)[6:0]						
		15:8		GRDB1[1:0]		GRDA1[1:0]				ACCDONE1
		7:0		PIN1[6:0]						
0x03F8	ITCREC2	31:24		GRDB(2+1)[1:0]		GRDA(2+1)[1:0]				ACCDONE(2+1)
		23:16		PIN(2+1)[6:0]						
		15:8		GRDB2[1:0]		GRDA2[1:0]				ACCDONE2
		7:0		PIN2[6:0]						
0x03FC	ITCREC3	31:24		GRDB(3+1)[1:0]		GRDA(3+1)[1:0]				ACCDONE(3+1)
		23:16		PIN(3+1)[6:0]						
		15:8		GRDB3[1:0]		GRDA3[1:0]				ACCDONE3
		7:0		PIN3[6:0]						
0x0400	ITCREC4	31:24		GRDB(4+1)[1:0]		GRDA(4+1)[1:0]				ACCDONE(4+1)
		23:16		PIN(4+1)[6:0]						
		15:8		GRDB4[1:0]		GRDA4[1:0]				ACCDONE4
		7:0		PIN4[6:0]						
0x0404	ITCREC5	31:24		GRDB(5+1)[1:0]		GRDA(5+1)[1:0]				ACCDONE(5+1)
		23:16		PIN(5+1)[6:0]						
		15:8		GRDB5[1:0]		GRDA5[1:0]				ACCDONE5
		7:0		PIN5[6:0]						
0x0408	ITCREC6	31:24		GRDB(6+1)[1:0]		GRDA(6+1)[1:0]				ACCDONE(6+1)
		23:16		PIN(6+1)[6:0]						
		15:8		GRDB6[1:0]		GRDA6[1:0]				ACCDONE6
		7:0		PIN6[6:0]						
0x040C	ITCREC7	31:24		GRDB(7+1)[1:0]		GRDA(7+1)[1:0]				ACCDONE(7+1)
		23:16		PIN(7+1)[6:0]						
		15:8		GRDB7[1:0]		GRDA7[1:0]				ACCDONE7
		7:0		PIN7[6:0]						
0x0410	ITCREC8	31:24		GRDB(8+1)[1:0]		GRDA(8+1)[1:0]				ACCDONE(8+1)
		23:16		PIN(8+1)[6:0]						
		15:8		GRDB8[1:0]		GRDA8[1:0]				ACCDONE8
		7:0		PIN8[6:0]						
0x0414	ITCREC9	31:24		GRDB(9+1)[1:0]		GRDA(9+1)[1:0]				ACCDONE(9+1)
		23:16		PIN(9+1)[6:0]						
		15:8		GRDB9[1:0]		GRDA9[1:0]				ACCDONE9
		7:0		PIN9[6:0]						
0x0418	ITCREC10	31:24		GRDB(10+1)[1:0]		GRDA(10+1)[1:0]				ACCDONE(10+1)
		23:16		PIN(10+1)[6:0]						
		15:8		GRDB10[1:0]		GRDA10[1:0]				ACCDONE10
		7:0		PIN10[6:0]						
0x041C	ITCREC11	31:24		GRDB(11+1)[1:0]		GRDA(11+1)[1:0]				ACCDONE(11+1)
		23:16		PIN(11+1)[6:0]						
		15:8		GRDB11[1:0]		GRDA11[1:0]				ACCDONE11
		7:0		PIN11[6:0]						

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0420	ITCREC12	31:24		GRDB(12+1)[1:0]		GRDA(12+1)[1:0]				ACCDONE(12+1)
		23:16		PIN(12+1)[6:0]						
		15:8		GRDB12[1:0]		GRDA12[1:0]				ACCDONE12
		7:0		PIN12[6:0]						
0x0424	ITCREC13	31:24		GRDB(13+1)[1:0]		GRDA(13+1)[1:0]				ACCDONE(13+1)
		23:16		PIN(13+1)[6:0]						
		15:8		GRDB13[1:0]		GRDA13[1:0]				ACCDONE13
		7:0		PIN13[6:0]						
0x0428	ITCREC14	31:24		GRDB(14+1)[1:0]		GRDA(14+1)[1:0]				ACCDONE(14+1)
		23:16		PIN(14+1)[6:0]						
		15:8		GRDB14[1:0]		GRDA14[1:0]				ACCDONE14
		7:0		PIN14[6:0]						
0x042C	ITCREC15	31:24		GRDB(15+1)[1:0]		GRDA(15+1)[1:0]				ACCDONE(15+1)
		23:16		PIN(15+1)[6:0]						
		15:8		GRDB15[1:0]		GRDA15[1:0]				ACCDONE15
		7:0		PIN15[6:0]						
0x0430	ITCRES0	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0434	ITCRES1	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0438	ITCRES2	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x043C	ITCRES3	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0440	ITCRES4	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0444	ITCRES5	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0448	ITCRES6	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x044C	ITCRES7	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0450	ITCRES8	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0454	ITCRES9	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0458	ITCRES10	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x045C	ITCRES11	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0460	ITCRES12	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0464	ITCRES13	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0468	ITCRES14	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x046C	ITCRES15	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0470	ITCRES16	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0474	ITCRES17	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0478	ITCRES18	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x047C	ITCRES19	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0480	ITCRES20	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0484	ITCRES21	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0488	ITCRES22	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x048C	ITCRES23	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0490	ITCRES24	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0494	ITCRES25	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x0498	ITCRES26	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x049C	ITCRES27	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x04A0	ITCRES28	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x04A4	ITCRES29	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x04A8	ITCRES30	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x04AC	ITCRES31	31:24	RES[31:24]							
		23:16	RES[23:16]							
		15:8	RES[15:8]							
		7:0	RES[7:0]							
0x04B0	ITCCURRES	31:24	CURRES[31:24]							
		23:16	CURRES[23:16]							
		15:8	CURRES[15:8]							
		7:0	CURRES[7:0]							
0x04B4 ... 0x7C2FFF	Reserved									
0x7C3000	SDATACMD0	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3004	SDATACMD1	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3008	SDATACMD2	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C300C	SDATACMD3	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3010	SDATACMD4	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3014	SDATACMD5	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7C3018	SDATACMD6	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C301C	SDATACMD7	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3020	SDATACMD8	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3024	SDATACMD9	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3028	SDATACMD10	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C302C	SDATACMD11	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3030	SDATACMD12	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3034	SDATACMD13	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3038	SDATACMD14	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C303C	SDATACMD15	31:24	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
		23:16	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
		15:8	CHRG	DISCHRG	CONV	BAL				
		7:0	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
0x7C3040	SMATHCMD0	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3044	SMATHCMD1	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3048	SMATHCMD2	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C304C	SMATHCMD3	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3050	SMATHCMD4	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7C3054	SMATHCMD5	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3058	SMATHCMD6	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C305C	SMATHCMD7	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3060	SMATHCMD8	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3064	SMATHCMD9	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3068	SMATHCMD10	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C306C	SMATHCMD11	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3070	SMATHCMD12	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3074	SMATHCMD13	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3078	SMATHCMD14	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C307C	SMATHCMD15	31:24								
		23:16	END	INT	FIRST				CMP	ACCCLR
		15:8	LAST			ACCA	ACCB	WMOV		
		7:0	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
0x7C3080	SDATAMAP	31:24			DATASEQ3[2:0]				SPLIT3[1:0]	
		23:16			DATASEQ2[2:0]				SPLIT2[1:0]	
		15:8			DATASEQ1[2:0]				SPLIT1[1:0]	
		7:0			DATASEQ0[2:0]				SPLIT0[1:0]	
0x7C3084	SMATHMAP	31:24			MATHSEQ3[3:0]				SPLIT3[1:0]	
		23:16			MATHSEQ2[3:0]				SPLIT2[1:0]	
		15:8			MATHSEQ1[3:0]				SPLIT1[1:0]	
		7:0			MATHSEQ0[3:0]				SPLIT0[1:0]	

17.3.1. Control Register 1

Name: ITCCON1
Offset: 0x380

Note:
1. Set the ADON bit only after the ITC module has been configured. Changing ITC configuration bits when ON = 1 will result in unpredictable behavior.

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; S = Bit can be Set only; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON				CVDEN			SIGN
Access	R/W				R/W			R/W
Reset	0				0			0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON ITC Enable bit(1)

Value	Description
1	ITC module is enabled.
0	ITC module is off.

Bit 11 – CVDEN CVD Capacitance Array Enable bit

Value	Description
1	Additional to C_{HOLD} capacitance array is enabled.
0	Additional to C_{HOLD} capacitance array is disabled.

Bit 8 – SIGN Result Sign Format Selection bit

Value	Description
1	The ITCRESx result is in signed format.
0	The ITCRESx result is un-signed.

17.3.2. Control Register 2

Name: ITCCON2
Offset: 0x384

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	TMRPR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TMRPR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						TRGEN2	TRGEN1	TRGEN0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:16 – TMRPR[15:0] Internal Timer Period Selection bits
These bits select the internal timer period in the ITC clock cycles. This timer can be used to trigger the ITC.

Value	Description
65535	PERIOD = 65536 x ADC3 T_{AD}s
...	
1	PERIOD = 2 x ADC3 T_{AD}s
0	PERIOD = ADC3 T_{AD}s

Bit 10 – TRGEN2 List 2 Enable bit

Value	Description
1	Triggers for List 2 are enabled.
0	Triggers for List 2 are disabled.

Bit 9 – TRGEN1 List 1 Enable bit

Value	Description
1	Triggers for List 1 are enabled.
0	Triggers for List 1 are disabled.

Bit 8 – TRGEN0 List 0 Enable bit

Value	Description
1	Triggers for List 0 are enabled.
0	Triggers for List 0 are disabled.

17.3.3. ITC Status Register

Name: ITCSTAT
Offset: 0x388

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	TSTDATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TSTDATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TSTEN					DRDY		
Access	R/W					R		
Reset	0					0		

Bit	7	6	5	4	3	2	1	0
						INT2	INT1	INT0
Access						HS	HS	HS
Reset						0	0	0

Bits 31:16 – TSTDATA[15:0] Test Data Value bits

Bit 15 – TSTEN Test Mode Enable bit

Value	Description
1	The conversion result is replaced with TSTDATA[15:0] value.
0	The Test mode is disabled.

Bit 10 – DRDY ITC Ready bit

Value	Description
1	The ITC can be triggered.
0	The ITC is not ready for operation.

Bit 2 – INT2 List 2 Interrupt Flag bit

Value	Description
1	List 2 generated the ITC interrupt.
0	List 2 did not generate the ITC interrupt.

Bit 1 – INT1 List 1 Interrupt Flag bit

Value	Description
1	List 1 generated the ITC interrupt.
0	List 1 did not generate the ITC interrupt.

Bit 0 – INT0 List 0 Interrupt Flag bit

Value	Description
1	List 0 generated the ITC interrupt.
0	List 0 did not generate the ITC interrupt.

17.3.4. TX Pins A Selection Register

Name: ITCTXA
Offset: 0x38C

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	TXA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TXA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TXA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TXA[31:0] TX Pins A Selection bits

Value	Description
1	CVDTXx pin is selected.
0	CVDTXx pin is disabled.

17.3.5. TX Pins B Selection Register

Name: ITCTXB
Offset: 0x390

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	TXB[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TXB[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TXB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TXB[31:0] TX Pins B Selection bits

Value	Description
1	CVDTXx pin is selected.
0	CVDTXx pin is disabled.

17.3.6. TX Pins C Selection Register

Name: ITCTXC
Offset: 0x394

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	TXC[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TXC[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TXC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TXC[31:0] TX Pins C Selection bits

Value	Description
1	CVDTXx pin is selected.
0	CVDTXx pin is disabled.

17.3.7. Comparator Hit Register

Name: ITCHIT
Offset: 0x398

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	HIT[31:24]							
Access	HS	R/W	X	X	X	X	X	X
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HIT[23:16]							
Access	X	X	X	X	X	X	X	X
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HIT[15:8]							
Access	X	X	X	X	X	X	X	X
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HIT[7:0]							
Access	X	X	X	X	X	X	X	X
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – HIT[31:0] Record x Comparator Event Flag bits
These bits are set when ITC record data meets comparison criteria and cleared when a '0' is written to these bits by software.

17.3.8. List x Control Register

Name: ITCLSxCON
Offset: 0x39C, 0x3B8, 0x3D4

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	MODE[2:0]			WM[1:0]		CM[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMAEN		MULEN	SAMC[4:0]				
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TRGEN	SAMP	TRGCLR	SSRC[4:0]				
Access	HS	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			RECCNT[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 31:29 – MODE[2:0] List x Mode Selection bits

Value	Description
7	One trigger executes all records in back-to-back processing. The list interrupt is generated after the last list record is processed if at least one record's result matches the comparator criteria.
6	One trigger executes all records in back-to-back processing. The list interrupts are generated for records every time the result matches the comparator criteria.
5	One trigger executes all records in back-to-back processing. The list interrupt is generated after the last record is processed.
4	One trigger executes all records in back-to-back processing. The list interrupts are not generated.
3	Reserved
2	One record is processed per trigger. The list interrupt is generated after the last record is processed.
1	One record is processed per trigger. The list interrupt is generated after each record is processed.
0	One record is processed per trigger. The list interrupts are not generated by the list.

Bits 28:27 – WM[1:0] Result Write Mode Selection bits

Value	Description
3	Results are saved when a match does not occur.
2	No results are saved.
1	Results are saved when a match occurs.
0	All result data are always saved.

Bits 26:24 – CM[2:0] Comparison Mode Selection bits

Value	Description
7-5	Reserved

Value	Description
4	Match Outside Window (Accumulator A < ITCLSnCMPLO and Accumulator A > ITCLSnCMPHI).
3	Match Inside Window (ITCLSnCMPLO < Accumulator A < ITCLSnCMPHI).
2	Match Greater Than (Accumulator A > ITCLSnCMPHI).
1	Match Less Than (Accumulator A < ITCLSnCMPLO).
0	Comparison is disabled.

Bit 23 – DMAEN DMA Triggers to Load New Command Enable bit

Value	Description
1	DMA triggers are enabled.
0	ITC does not generate DMA triggers.

Bit 21 – MULEN Multiple Inputs Connection Enable bit

Allow CVDANx pins, as defined by the ITCLSnMUL register list, to be connected together.

Value	Description
1	CVDANx pins defined in the ITCLSnMUL register are connected together.
0	All CVDANx pins are separate.

Bits 20:16 – SAMC[4:0] Balance Counter bits

Value	Description
31	31 T_{AD}s
. . .	
1	1 T_{AD}s
0	0 T_{AD}s

Bit 15 – TRGEN List Trigger Enable bit

Value	Description
1	List trigger is enabled.
0	List trigger is disabled.

Bit 14 – SAMP Balance Switch Control bit

Value	Description
1	Closes the internal switch between the CVDANx pin and a CVD capacitor when the software trigger source is selected (SSRC bits = 0).
0	Opens a CVDANx switch and starts the conversion when the software trigger source is selected (SSRC = 0).

Bit 13 – TRGCLR Trigger Clear bit

Value	Description
1	TRGEN is cleared by hardware after a trigger is received by this list.
0	TRGEN is only cleared by software.

Bits 12:8 – SSRC[4:0] Trigger Source Select bits

Value	Description
21	ADTRG31 (PPS)
20	QEI1
19	CCP4
18	CCP3
17	CCP2
16	PTG
15	CCP1
14	CLC4

Value	Description
13	CLC3
12	CLC2
11	CLC1
10	CCP5
9	PWM4 Trigger 2
8	PWM4 Trigger 1
7	Internal timer periodic trigger as set up by TMRPR bits in the ITCCON2 register.
6	PWM3 Trigger 2
5	PWM3 Trigger 1
4	PWM2 Trigger 2
3	PWM2 Trigger 1
2	PWM1 Trigger 2
1	PWM1 Trigger 1
0	Software trigger is controlled by a SAMP bit. A single trigger is generated when SAMP transitions from 1 to 0.

Bits 5:0 – RECCNT[5:0] Number of Records in List bits

Value	Description
63-33	Reserved
32	32
...	
1	1
0	0

17.3.9. List x Status Register

Name: ITCLSxSTAT
Offset: 0x3A0, 0x3BC, 0x3D8

Notes:
1. Software can clear or set this bit without affecting the conversion; however, setting this bit will generate the list interrupt if enabled.
2. When TRGEN bit = '1' in ITCSLxCON register, the NEXT[5:0] bits are locked from user writes.

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	TACT	BUSY						
Access	HS	HS						
Reset	0	0						

Bit	23	22	21	20	19	18	17	16
			INT					
Access			HS					
Reset			0					

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			NEXT[5:0]					
Access			HS	HC	R/W	X	X	X
Reset			0	0	0	0	0	0

Bit 31 – TACT Trigger Active bit

Value	Description
1	The trigger is asserted.
0	The trigger is not asserted.

Bit 30 – BUSY Busy Flag bit

Value	Description
1	The ITC is busy.
0	The ITC is idle.

Bit 21 – INT List Interrupt Flag bit(1)

Value	Description
1	List interrupt was generated.
0	The interrupt was not generated.

Bits 5:0 – NEXT[5:0] Entry to Convert on Next Trigger bits(2)
Indicates the next entry number on the list that will be converted for the trigger.

Value	Description
63–32	Reserved
31	Record 31 next
...	
1	Record 1 next
0	Record 0 next

17.3.10. List x Comparator High Threshold Value Register

Name: ITCLSxCMPHI
Offset: 0x3A4, 0x3C0, 0x3DC

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	CMPHIGH[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CMPHIGH[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CMPHIGH[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPHIGH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CMPHIGH[31:0] Comparator High Threshold Value bits

17.3.11. List x Comparator Low Threshold Value Register

Name: ITCLSxCMPLO
Offset: 0x3A8, 0x3C4, 0x3E0

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	CMPLOW[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CMPLOW[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CMPLOW[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPLOW[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CMPLOW[31:0] Comparator Low Threshold Value bits

17.3.12. List x Multiple Connections Selection Register

Name: ITCLSxMUL
Offset: 0x3AC, 0x3C8, 0x3E4

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	MUL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MUL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MUL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MUL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MUL[31:0] Pins from CVDAN0 to CVDAN31 Selection to Connected Together bits

17.3.13. ITC List x Acquisition and Post-Processing Control Register

Name: ITCLSxSEQ
Offset: 0x3B0, 0x3CC, 0x3E8

Legend: r = Reserved Bit; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
		CVDCAP[2:0]						
Access		R/W	R/W	R/W				
Reset		0	0	0				

Bit	23	22	21	20	19	18	17	16
	DATASEQ[2:0]							
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	15	14	13	12	11	10	9	8
	ACCMODE							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
					ACCCNT[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 30:28 – CVDCAP[2:0] CVD Internal Capacitance Selection bits

Value	Description
7	17.5 pF
...	
2	5 pF
1	2.5 pF
0	0

Bits 23:21 – DATASEQ[2:0] Acquisition Commands Sequence Select bits

Value	Description
7-4	Software defined acquisition sequences
...	
3	Reserved
2	Hardware CVD acquisition sequence
1	Reserved
0	Default acquisition sequence when the record's analog input is sampled during time defined by SAMC[4:0] bits and then converted.

Bit 15 – ACCMODE Accumulation Mode bit

Value	Description
1	One scan is executed, and each record is accumulated multiple times back-to-back before processing the next record.
0	Multiple scans are executed and the record results are accumulated between scans.

Bits 3:0 – ACCCNT[3:0] Number of Record Accumulations Selection bits

Value	Description
15-1	Each record in the list is executed $2^{ACCCNT[3:0]}$ times.
0	Each record in the list is executed one time.

17.3.14. List x Post-Processing Timers Delay Selection Register

Name: ITCLSxTMR
Offset: 0x3B4, 0x3D0, 0x3EC

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	TMRD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TMRC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TMRB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TMRA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – TMRD[7:0] Post-Processing Command Sequence Timer D Delay Selection Bits in TADs bits

Bits 23:16 – TMRC[7:0] Post-Processing Command Sequence Timer C Delay Selection Bits in TADs bits

Bits 15:8 – TMRB[7:0] Post-Processing Command Sequence Timer B Delay Selection Bits in TADs bits

Bits 7:0 – TMRA[7:0] Post-Processing Command Sequence Timer A Delay Selection Bits in TADs bits

17.3.15. Records Pair Configuration Register

Name: ITCRECx
Offset: 0x3F0, 0x3F4, 0x3F8, 0x3FC, 0x400, 0x404, 0x408, 0x40C, 0x410, 0x414, 0x418, 0x41C, 0x420, 0x424, 0x428, 0x42C

Legend: r = Reserved Bit; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
		GRDB(x+1)[1:0]		GRDA(x+1)[1:0]				ACCDONE(x+1)
Access		R/W	R/W	R/W	R/W			R/W
Reset		0	0	0	0			0

Bit	23	22	21	20	19	18	17	16
		PIN(x+1)[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
		GRDBx[1:0]		GRDAx[1:0]				ACCDONEx
Access		R/W	R/W	R/W	R/W			R/W
Reset		0	0	0	0			0

Bit	7	6	5	4	3	2	1	0
		PINx[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 30:29 – GRDB(x+1)[1:0] CVD Guard B Assignment bits

Value	Description
11	Reserved
10	Guard B is assigned to a pin specified by PIN[6:0] + 1.
01	Guard B is assigned to a pin specified by PIN[6:0] - 1.
00	Guard B is not used.

Bits 28:27 – GRDA(x+1)[1:0] CVD Guard A Assignment bits

Value	Description
11	Reserved
10	Guard A is assigned to a pin specified by PIN[6:0] + 1.
01	Guard A is assigned to a pin specified by PIN[6:0] - 1.
00	Guard A is not used.

Bit 24 – ACCDONE(x+1) Accumulations Done Flag bit

Value	Description
1	The specified number of accumulations has been completed. This bit automatically set by hardware only after the sequence finishes execution and is automatically cleared by hardware when the ITCRESx register associated with the record is read.
0	The specified number of accumulations is not finished.

Bits 22:16 – PIN(x+1)[6:0] CVDANx Analog Input Selection bits

Value	Description
127-32	Reserved
31	CVDAN31
...	
1	CVDAN1
0	CVDAN0

Bits 14:13 – GRDBx[1:0] CVD Guard B Assignment bits

Value	Description
11	Reserved
10	Guard B is assigned to a pin specified by PIN[6:0] + 1.
01	Guard B is assigned to a pin specified by PIN[6:0] - 1.
00	Guard B is not used.

Bits 12:11 – GRDAx[1:0] CVD Guard A Assignment bits

Value	Description
11	Reserved
10	Guard A is assigned to a pin specified by PIN[6:0] + 1.
01	Guard A is assigned to a pin specified by PIN[6:0] - 1.
00	Guard A is not used.

Bit 8 – ACCDONEx Accumulations Done Flag bit

Value	Description
1	The specified number of accumulations has been completed. This bit automatically set by hardware only after the sequence finishes execution and is automatically cleared by hardware when the ITCRESx register associated with the record is read.
0	The specified number of accumulations is not finished.

Bits 6:0 – PINx[6:0] CVDANx Analog Input Selection bits

Value	Description
127-32	Reserved
31	CVDAN31
...	
1	CVDAN1
0	CVDAN0

17.3.16. Record x Result Register

Name: ITCRESx
Offset: 0x430, 0x434, 0x438, 0x43C, 0x440, 0x444, 0x448, 0x44C, 0x450, 0x454, 0x458, 0x45C, 0x460, 0x464, 0x468, 0x46C, 0x470, 0x474, 0x478, 0x47C, 0x480, 0x484, 0x488, 0x48C, 0x490, 0x494, 0x498, 0x49C, 0x4A0, 0x4A4, 0x4A8, 0x4AC

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	RES[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	RES[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	RES[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	RES[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – RES[31:0] The Record x Result bits

17.3.17. Current Result Register

Name: ITCCURRES
Offset: 0x4B0

Legend: HC = Bit is Cleared by Hardware; HS = Bit is Set by Hardware; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	CURRES[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CURRES[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CURRES[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CURRES[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CURRES[31:0] Current Record Result Value bits

17.3.18. ITC Acquisition Sequence Commands Word x Register

Name: SDATACMDx
Offset: 0x7C3000, 0x7C3004, 0x7C3008, 0x7C300C, 0x7C3010, 0x7C3014, 0x7C3018, 0x7C301C, 0x7C3020, 0x7C3024, 0x7C3028, 0x7C302C, 0x7C3030, 0x7C3034, 0x7C3038, 0x7C303C

Legend: r = Reserved Bit; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
	END	LOOP[3:0]				DMASTP	DMATXC	DMATXB
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DMATXA	DMALAST	DMALACC	MSTART	MSEQ[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CHRG	DISCHRG	CONV	BAL				
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit	7	6	5	4	3	2	1	0
	PCC[1:0]		PCB[1:0]		PCA[1:0]		PC0[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – END Last Command in Sequence bit

Value	Description
1	This command is last in the sequence.
0	There is the next command after this command.

Bits 30:27 – LOOP[3:0] Wait bits

Value	Description
12-15	Reserved
11	The command waits for the new command that has been sent by DMA to the SDATACMDx register.
10	The command waits for Timer D with delay defined by the TMRD[7:0] bits in the ITCLSxTMR register.
9	The command waits for Timer C with delay defined by the TMRC[7:0] bits in the ITCLSxTMR register.
5-8	Reserved
4	The command waits for the ADC’s end of conversion.
3	The command waits for delay defined in the SAMC[4:0] bits in the ITCLSxCON register.
2	The command waits for Timer B with delay defined by the TMRB[7:0] bits in the ITCLSxTMR register.
1	The command waits for Timer A with delay defined by the TMRA[7:0] bits in the ITCLSxTMR register.
0	No delay, the next instruction must be executed right away.

Bit 26 – DMASTP DMA Transfer of New Command Trigger bit
The DMA must update/write the ITCSDATACMDx register to clear the DMA trigger.

Value	Description
1	This command requests the DMA to transfer new commands in the ITCSDATACMDx registers.
0	DMA is not requested.

Bit 25 – DMATXC DMA Transfer to Update ITCTXC Register Trigger bit
The DMA must update/write the ITCTXC register to clear the DMA trigger.

Value	Description
1	This command requests the DMA to transfer a new value to the ITCTXC register.
0	DMA is not requested.

Bit 24 – DMATXB DMA Transfer to Update ITCTXB Register Trigger bit
The DMA must update/write the ITCTXB register to clear the DMA trigger.

Value	Description
1	This command requests the DMA to transfer a new value to the ITCTXB register.
0	DMA is not requested.

Bit 23 – DMATXA DMA Transfer to Update ITCTXA Register Trigger bit
The DMA must update/write the ITCTXA register to clear the DMA trigger.

Value	Description
1	This command requests the DMA to transfer a new value to the ITCTXA register.
0	DMA is not requested.

Bit 22 – DMALAST DMA Transfer Only If All Records Are Processed Enable bit

Value	Description
1	DMA triggers specified by DMASTEP, DMATXA, DMATXB and DMATXC bits are sent only when the last record in the last scan is processed.
0	DMA triggers specified by DMASTEP, DMATXA, DMATXB and DMATXC bits are always executed.

Bit 21 – DMALACC DMA Transfer Each Time When a Record is Accumulated Last Time Enable bit

Value	Description
1	DMA triggers specified by DMASTEP, DMATXA, DMATXB and DMATXC bits are sent only when the record is accumulated last time.
0	DMA triggers specified by DMASTEP, DMATXA, DMATXB and DMATXC bits are always executed.

Bit 20 – MSTART Post-Processing Math Sequence Start bit

Value	Description
1	This command executes the math sequence.
0	The math sequence is not executed.

Bits 19:16 – MSEQ[3:0] MSEQ Math Post-Processing Sequence Number bits

Value	Description
15	Math sequence 15
...	
1	Math sequence 1
0	Math sequence 0

Bit 15 – CHRG CVD Capacitors Array Charge bit

Value	Description
1	The command will connect the CVD capacitors array to V_{DD} (power).
0	The charge switch is open.

Bit 14 – DISCHRG CVD Capacitors Array Discharge bit

Value	Description
1	The command will connect the CVD capacitors array to V_{SS} (ground).

Value	Description
0	The discharge switch is open.

Bit 13 – CONV ADC3 Conversion Request bit

Value	Description
1	The command will start an ITC conversion of CVD capacitors array level.
0	The conversion is not requested.

Bit 12 – BAL Balance Charge bit

Value	Description
1	The command will connect the CVD capacitors array to the sensor CVDANx pin.
0	The balance switch is open.

Bits 7:6 – PCC[1:0] ITCTXC Register Level bits

Value	Description
3	CVDTXx pins defined in the ITCTXC register are tri-stated.
2	CVDTXx pins defined in the ITCTXC register are set to a high level.
1	CVDTXx pins defined in the ITCTXC register are set to a low level.
0	ITCTXC register is not used, the pins defined in ITCTXC are controlled by TRIS and LAT registers.

Bits 5:4 – PCB[1:0] ITCTXB Register Level bits

Value	Description
3	CVDTXx pins defined in the ITCTXB register are tri-stated.
2	CVDTXx pins defined in the ITCTXB register are set to a high level.
1	CVDTXx pins defined in the ITCTXB register are set to a low level.
0	ITCTXB register is not used, the pins defined in ITCTXB are controlled by TRIS and LAT registers.

Bits 3:2 – PCA[1:0] ITCTXA Register Level bits

Value	Description
3	CVDTXx pins defined in the ITCTXA register are tri-stated.
2	CVDTXx pins defined in the ITCTXA register are set to a high level.
1	CVDTXx pins defined in the ITCTXA register are set to a low level.
0	ITCTXA register is not used, the pins defined in ITCTXBA are controlled by TRIS and LAT registers.

Bits 1:0 – PC0[1:0] CVDANx Pin Level Bits

Value	Description
3	CVDANx pins are tri-stated.
2	CVDANx pins are set to a high level.
1	CVDANx pins are set to a low level.
0	CVDANx is not used, the pin is controlled by TRIS and LAT registers.

17.3.19. ITC Math Sequence Commands Word x Register

Name: SMATHCMDx
Offset: 0x7C3040, 0x7C3044, 0x7C3048, 0x7C304C, 0x7C3050, 0x7C3054, 0x7C3058, 0x7C305C, 0x7C3060, 0x7C3064, 0x7C3068, 0x7C306C, 0x7C3070, 0x7C3074, 0x7C3078, 0x7C307C

Legend: r = Reserved Bit; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	END	INT	FIRST				CMP	ACCCLR
Access	R/W	R/W	R/W				R/W	R/W
Reset	0	0	0				0	0

Bit	15	14	13	12	11	10	9	8
	LAST			ACCA	ACCB	WMOV		
Access	R/W			R/W	R/W	R/W		
Reset	0			0	0	0		

Bit	7	6	5	4	3	2	1	0
	WM[1:0]		F[1:0]		BIN[1:0]		AIN[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 23 – END Last Command in Sequence bit

Value	Description
1	This command is last in the sequence.
0	There is the next command after this command.

Bit 22 – INT Interrupt Request bit

Value	Description
1	The interrupt is generated by this command.
0	The interrupt is not generated by this command.

Bit 21 – FIRST Zero for First Accumulation Enable bit

Value	Description
1	The input B is forced to zero when the first accumulation is executed.
0	Normal operation when input B is selected by the BIN[1:0] bits.

Bit 17 – CMP Compare Accumulator A bit

Value	Description
1	This command compares Accumulator A with thresholds in the ITCLSxCMPLO and ITCLSxCMPHI registers as defined in the CM[2:0] bits of the ITCLSxCON register.
0	The comparison is disabled.

Bit 16 – ACCCLR Accumulator A and B Clear Request bit

Value	Description
1	The accumulators A and B are cleared by this command.
0	Normal operation for the accumulators A and B.

Bit 15 – LAST Last Accumulation Execution Enable bit

Value	Description
1	The interrupt and comparison specified by the INT and CMP bits in this command are executed for the last accumulation only.
0	During normal operation, the interrupt and comparison are always executed.

Bit 12 – ACCA Latch Result into Accumulator A bit

Value	Description
1	This command latches the math result into Accumulator A.
0	Accumulator A is not updated.

Bit 11 – ACCB Latch Result into Accumulator B bit

Value	Description
1	This command latches the math result into Accumulator B.
0	Accumulator B is not updated.

Bit 10 – WMOV Write Mode Overwrite bit

Value	Description
1	The Result Write mode is replaced with settings in the WM[1:0] bits of this command word.
0	The Result Write mode is defined by the WM[1:0] bits in the ITCLSxCON register.

Bits 7:6 – WM[1:0] Command Write Mode bits
These bits are used instead of list settings when the WMOV bit is set.

Value	Description
3	Results are saved when a match does not occur.
2	No results are saved.
1	Results are saved when a match occurs.
0	All result data are always saved.

Bits 5:4 – F[1:0] Math Operation Select bits

Value	Description
3	BIN - AIN
2	BIN + AIN
1	BIN
0	AIN

Bits 3:2 – BIN[1:0] Input B Select bits

Value	Description
3	ADC3 conversion result
2	Reserved
1	Accumulator B
0	ITCRESx result register

Bits 1:0 – AIN[1:0] Input A Select bits

Value	Description
3	ADC3 conversion result

Value	Description
2	Reserved
1	Accumulator A
0	Zero

17.3.20. ITC Math Sequence Commands Array Map Register

Name: SDATAMAP
Offset: 0x7C3080

Legend: r = Reserved Bit; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
			DATASEQ3[2:0]				SPLIT3[1:0]	
Access			R/W	R/W	R/W		R/W	R/W
Reset			1	1	1		0	0

Bit	23	22	21	20	19	18	17	16
			DATASEQ2[2:0]				SPLIT2[1:0]	
Access			R/W	R/W	R/W		R/W	R/W
Reset			1	1	1		0	0

Bit	15	14	13	12	11	10	9	8
			DATASEQ1[2:0]				SPLIT1[1:0]	
Access			R/W	R/W	R/W		R/W	R/W
Reset			0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
			DATASEQ0[2:0]				SPLIT0[1:0]	
Access			R/W	R/W	R/W		R/W	R/W
Reset			1	1	1		0	0

Bits 29:27 – DATASEQ3[2:0] Acquisition Sequence Number Bits for Commands from ITCSDATACMD12 to ITCSDATACMD15
These bits must match the DATASEQ[2:0] bits settings in the ITCSLxSEQ register.

Bits 25:24 – SPLIT3[1:0] Several Sequences Split Bits for Commands from ITCSDATACMD12 to ITCSDATACMD15

Bits 21:19 – DATASEQ2[2:0] Acquisition Sequence Number Bits for Commands from ITCSDATACMD8 to ITCSDATACMD11
These bits must match the DATASEQ[2:0] bits settings in the ITCSLxSEQ register.

Bits 17:16 – SPLIT2[1:0] Several Sequences Split Bits for Commands from ITCSDATACMD8 to ITCSDATACMD11

Bits 13:11 – DATASEQ1[2:0] Acquisition Sequence Number Bits for Commands from ITCSDATACMD4 to ITCSDATACMD7
These bits must match the DATASEQ[2:0] bits settings in the ITCSLxSEQ register.

Bits 9:8 – SPLIT1[1:0] Several Sequences Split Bits for Commands from ITCSDATACMD4 to ITCSDATACMD7

Bits 5:3 – DATASEQ0[2:0] Acquisition Sequence Number Bits for Commands from ITCSDATACMD0 to ITCSDATACMD3
These bits must match the DATASEQ[2:0] bits settings in the ITCSLxSEQ register.

Bits 1:0 – SPLIT0[1:0] Several Sequences Split Bits for Commands from ITCSDATACMD0 to ITCSDATACMD3

17.3.21. ITC Math Sequence Commands Array Map Register

Name: SMATHMAP
Offset: 0x7C3084

Legend: r = Reserved bit; R = Readable Bit; W = Writable Bit; U = Unimplemented Bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit value is unknown

Bit	31	30	29	28	27	26	25	24
			MATHSEQ3[3:0]				SPLIT3[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	0	0

Bit	23	22	21	20	19	18	17	16
			MATHSEQ2[3:0]				SPLIT2[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	0	0

Bit	15	14	13	12	11	10	9	8
			MATHSEQ1[3:0]				SPLIT1[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	0	0

Bit	7	6	5	4	3	2	1	0
			MATHSEQ0[3:0]				SPLIT0[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	0	0

Bits 29:26 – MATHSEQ3[3:0] Math Sequence Number Bits for Commands from ITCSMATHCMD12 to ITCSMATHCMD15

These bits must match the MSEQ[3:0] bits settings in the acquisition command ITC SDATACMDx register.

Bits 25:24 – SPLIT3[1:0] Several Sequences Split Bits for Commands from ITCSMATHCMD12 to ITCSMATHCMD15

Bits 21:18 – MATHSEQ2[3:0] Math Sequence Number Bits for Commands from ITCSMATHCMD8 to ITCSMATHCMD11

These bits must match the MSEQ[3:0] bits settings in the acquisition command ITCSDATACMDx register.

Bits 17:16 – SPLIT2[1:0] Several Sequences Split bits for Commands from ITCSMATHCMD8 to ITCSMATHCMD11

Bits 13:10 – MATHSEQ1[3:0] Math Sequence Number Bits for Commands from ITCSMATHCMD4 to ITCSMATHCMD7

These bits must match the MSEQ[3:0] bits settings in the acquisition command ITCSDATACMDx register.

Bits 9:8 – SPLIT1[1:0] Several Sequences Split Bits for Commands from ITCSMATHCMD4 to ITCSMATHCMD7

Bits 5:2 – MATHSEQ0[3:0] Math Sequence Number Bits for Commands from ITCSMATHCMD0 to ITCSMATHCMD3

These bits must match the MSEQ[3:0] bits settings in the acquisition command ITCSDATACMDx register.

Bits 1:0 – SPLIT0[1:0] Several Sequences Split Bits for Commands from ITCSMATHCMD0 to ITCSMATHCMD3

17.4. Analog to Digital Converter and Touch Controller Operation

17.4.1. Capacitive Voltage Divider (CVD)

17.4.1.1. Pseudo Differential CVD Algorithm

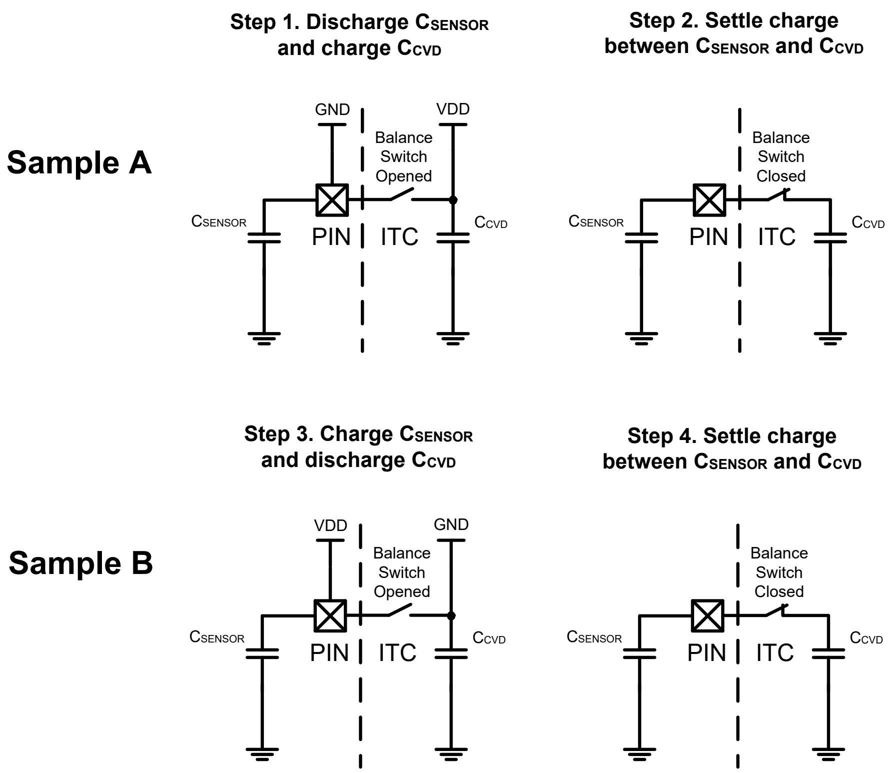
The CVD allows measuring a capacitance connected to the CVDANx input. It can be used to detect a touch event in touch sensor applications.

The CVD measurement algorithm consists of two phases: Sample A and Sample B.

Sample A: The capacitive sensor C_{SENSOR} is connected to I/O power (V_{DD}, charged) and the internal CVD capacitors array C_{CVD} is connected to ground (GND, discharged) as shown in Figure 17-2, step 1. Then, both capacitors are connected to balance a charge between them as shown in Figure 17-2, step 2. After the charge between the capacitors is settled, the resulting voltage is proportional to the ratio of the C_{SENSOR} and C_{SENSOR}+C_{CVD} capacitances. Then this voltage is converted by the ADC3.

Sample B: The capacitive sensor C_{SENSOR} is connected to ground (GND, discharged) and the internal CVD capacitors array C_{CVD} is connected to I/O power (V_{DD}, charged) (as shown in Figure 17-2, step 3). Then, both capacitors are connected to balance a charge between them, as shown in Figure 17-2, step 4. After the charge between the capacitors is settled, the resulting voltage is proportional to the ratio of the C_{CVD} and C_{SENSOR}+C_{CVD} capacitances. Then this voltage is converted by the ADC3.

Figure 17-2. CVD Connection Diagrams for Sample A and Sample B



The ITC module internally calculates the difference between Sample A and Sample B and accumulates the result in the ITCRESx register in signed format. Equation 17-1 shows the relation between the result in the ADnRESx register and the capacitances of C_{SENSOR} and C_{CVD}:

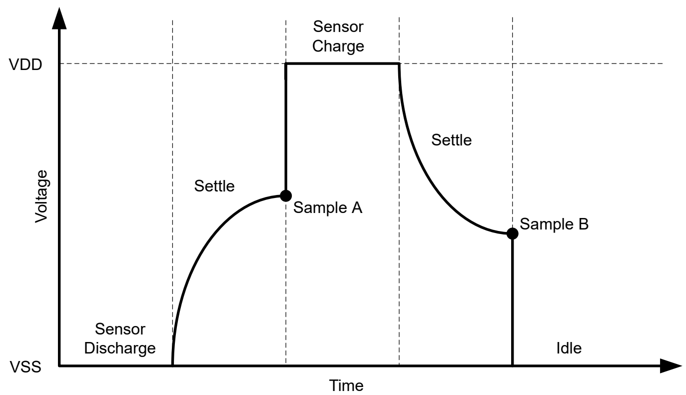
Equation 17-1. CVD Result Calculation

$$ITCRESx = Sample\ A - Sample\ B = 4095\ x\ \frac{C_{SENSOR} - C_{CVD}}{C_{SENSOR} + C_{CVD}}$$

To ensure maximum sensitivity, C_{SENSOR} should be equal to or close to the C_{CVD} value, which can be achieved by adjusting the C_{CVD} capacitance using the CVDCAP[2:0] bits (ITCLSxSEQ[30:28]).

The typical waveform on the CVD Sensor is shown in Figure 17-3 below.

Figure 17-3. CVD Pin Waveform



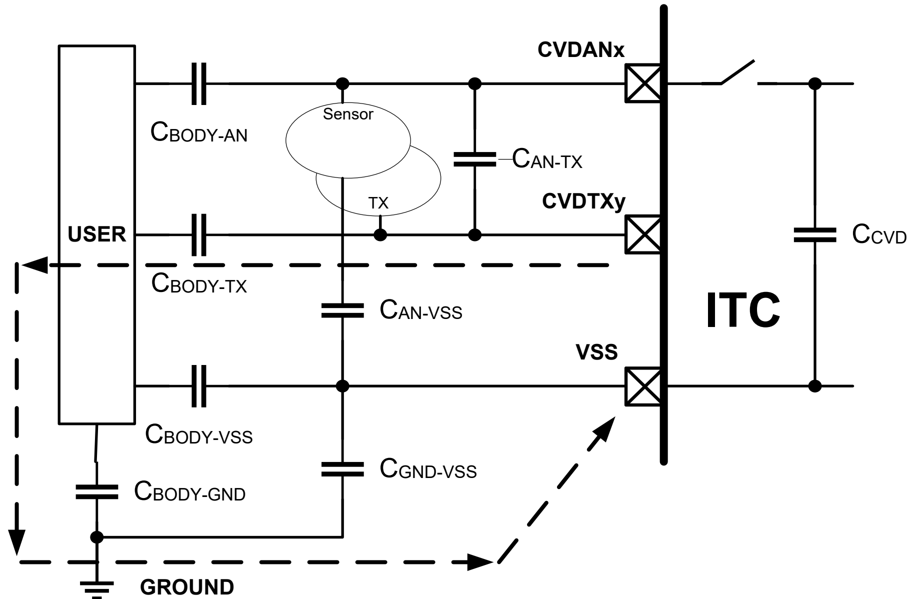
The CVD measures total capacitance, so as the base capacitance of a sensor decreases, the change in signal caused by a measured factor (such as a user's finger in the case of a capacitive touch application) will increase. Active guards are a way of minimizing the base capacitance by reducing the electric potential between the sensor and its surrounding environment. When designing the guard for an application, the best solution is to encircle the sensor and guard trace completely. The guard trace should be placed between the sensor and any low impedance source. The guard signal should be as close as possible to the CVD waveform. To generate the sensor's guard signal, the CVDTXx pins selected in ITCTXA, ITCTXB and ITCTXC registers and controlled in Acquisition Sequencer commands can be used.

17.4.1.2. Mutual Capacitance CVD Sensor

The mutual drive CVDTXx signal is a square wave that is driven low during all of Sample A and high during all of Sample B (so, it is in-phase with the external sensor's starting voltage for each sample). The mutual capacitance is formed by the AC coupling between the sensor CVDANx and a low impedance CVDTXx signal, as shown in Figure 17-4. This TX signal can be common for all sensors because only one sensor at a time is sampled. C_{TX-AN} is the amount of coupling between the TX trace and the sensor without any external influences.

When a user capacitively couples to the system, it pulls charge away from C_{TX-AN}. $C_{BODY-GND}$ tends to be incredibly large with respect to the other magnitudes of capacitance in the circuit model. Thus, if the user and the board share or strongly couple their grounds, the pulled TX charge will dissipate through the human body model coupling path into the board's ground. This path is shown with a dashed line in Figure 17-4. The effect of this dissipation will be to decrease the mutual capacitance between TX and the sensor.

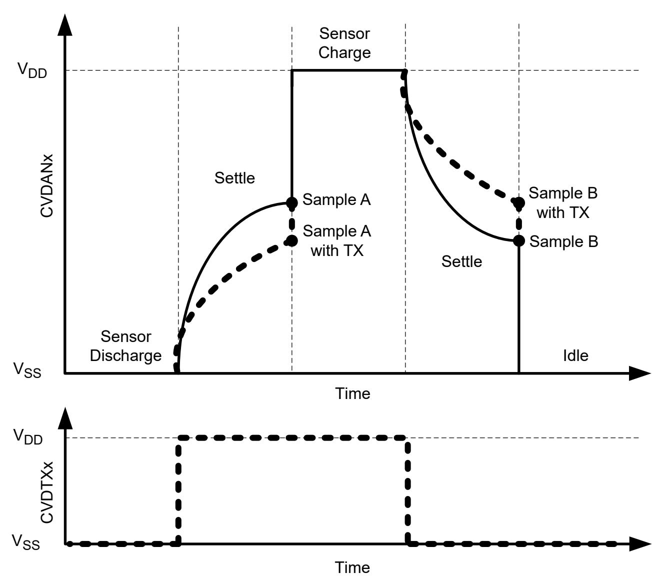
Figure 17-4. Mutual Capacitance CVD Sensor



As shown in Figure 17-5, the Capacitive Voltage Divider (CVD) technique uses two opposite polarity measurements to achieve a differential, self-capacitive reading (Sample A – Sample B). When the self-capacitance of a sensor increases, the first measurement of the CVD (Sample A) will decrease, and the second measurement of the CVD (Sample B) will increase.

If a TX square wave is driven in-phase with the CVD waveform, a mutual capacitance component is introduced to the final signal level. When the TX signal is driven high, the AC coupling between TX and the sensor will cause the final settled voltage to increase; when the TX signal is driven low, the AC coupling will cause the final settled voltage to decrease. It pulls the signal in the direction it's headed. As the mutual capacitance between TX and the sensor becomes stronger or weaker, the strength of the pulling on the signal likewise becomes stronger or weaker.

Figure 17-5. CVD Pin Waveform with Mutual TX Drive

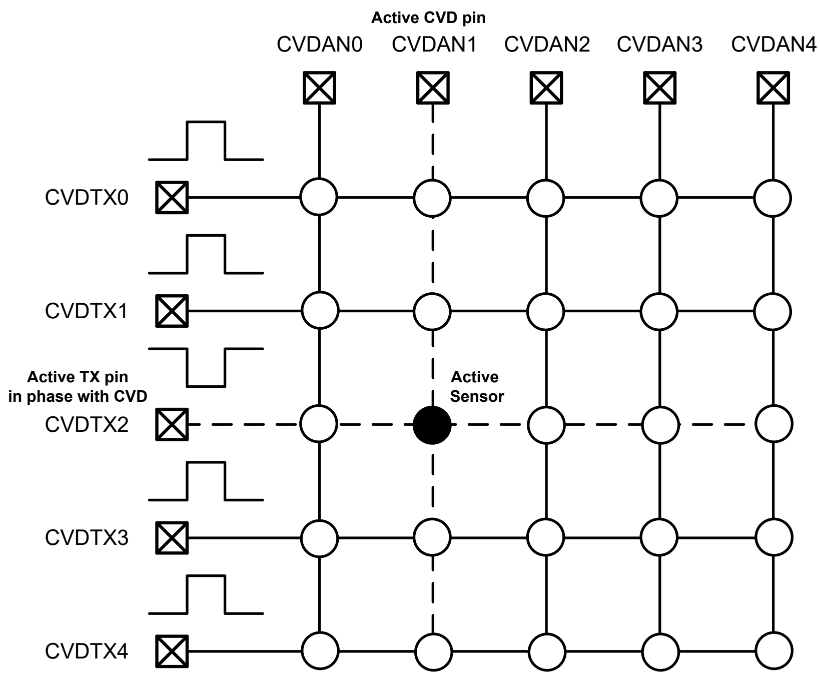


To reduce the base capacitance of a sensor and increase the sensitivity, the neighbor sensors in the 2D matrix can be used as active guards. This feature can be enabled by the GRDBx[1:0] and GRDAx[1:0] bits in the records registers ITCRECx.

17.4.1.3. 2-Dimensional Capacitive Voltage Divider Algorithm (2D CVD)

The 2-dimensional CVD touch sensing algorithm uses CVDTXx output pins to form a matrix of sensors, as shown in Figure 17-6. CVDTXx active outputs are selected in the ITCTXA, ITCTXB and ITCTXC registers. These selected CVDTXx pins are controlled by bits in Acquisition Sequencer commands.

Figure 17-6. 2D CVD 5x5 Sensors Matrix Example



17.4.2. Analog Inputs or CVD Sensors Pins

Any CVDANx pin can be mapped to any record by the PINx[6:0] bits in the record register ITCRECx.

The ANSEL register bits for the I/O ports associated with the analog input must be set to configure the corresponding pins as analog pins. A pin is configured as an analog input when the corresponding ANSELx bit = `1`. When the ANSELx bit = `0`, the pin is set to digital control. The ANSELx bits are set when the device comes out of reset, causing the ADC input pins to be configured as analog inputs by default.

The I/O CVDANx pin is controlled by ITC during ADC conversion or CVD operation. However, when the next CVDANx pin is processed, the control for the idle CVDANx pin is returned to the TRIS and LAT registers of the port.

The TRIS registers switch the pins between the digital inputs and outputs. The port pin that is required for CVD must have its corresponding bit cleared (TRISx = `0`) in the specific TRIS register, configuring the pin as an output. It will keep the idle CVD sensor grounded or connected to V_{DD} depending on LAT register settings.

If the sensor CVDANx pin floats (TRISx = `1`) when it is idle between CVD scans, then the robustness will degrade significantly even with a light noise.

17.4.3. CVD Active Guards Pins

To reduce the base capacitance of the CVDANx sensor and increase sensitivity, the active guards' traces around the sensor following the CVDANx voltage (CVD waveform) can be used. The idle neighbor pins to a CVDANx pin (CVDANx-1 and CVDANx+1) can function as active guards when enabled by the GRDA[1:0] and GRDB[1:0] bits in the record register ITCRECx. For the CVDANx pin assigned by the PIN[6:0] bits, the GRDA[1:0] bits and GRDB[1:0] map the guard signal to the CVDANx-1 and CVDANx+1 pins. The following options are available:

- Active guards are not used/disabled when GRDA[1:0] and GRDB[1:0] = `0` or `3`.
- The PCA[1:0] and/or PCB[1:0] bits in the acquisition commands will control CVDANx-1 when GRDA[1:0] = `1` and/or GRDB[1:0] = `1`.
- The PCA[1:0] and/or PCB[1:0] bits in the acquisition commands will control CVDANx+1 when GRDA[1:0] = `2` and/or GRDB[1:0] = `2`.

17.4.4. CVD Mutual Capacitance Pins

The CVDTXx pins, together with CVDANx pins, can form mutual capacitance sensors. The CVDTXx waveform should follow CVD levels on the CVDANx pin to induce the mutual capacitance. The active CVDTXx are selected in the ITCTXA, ITCTXB and ITCTXC registers.

The pins selected in ITCTXA are controlled by the PCA[1:0] bits in the acquisition commands, the pins selected in ITCTXB are controlled by the PCB[1:0] bits in the acquisition commands, and the pins selected in ITCTXC are controlled by the PCC[1:0] bits in the acquisition sequence commands.

The content (active CVDTXx pins) of ITCTXA, ITCTXB and ITCTXC registers can be updated by DMA triggered from the acquisition sequence.

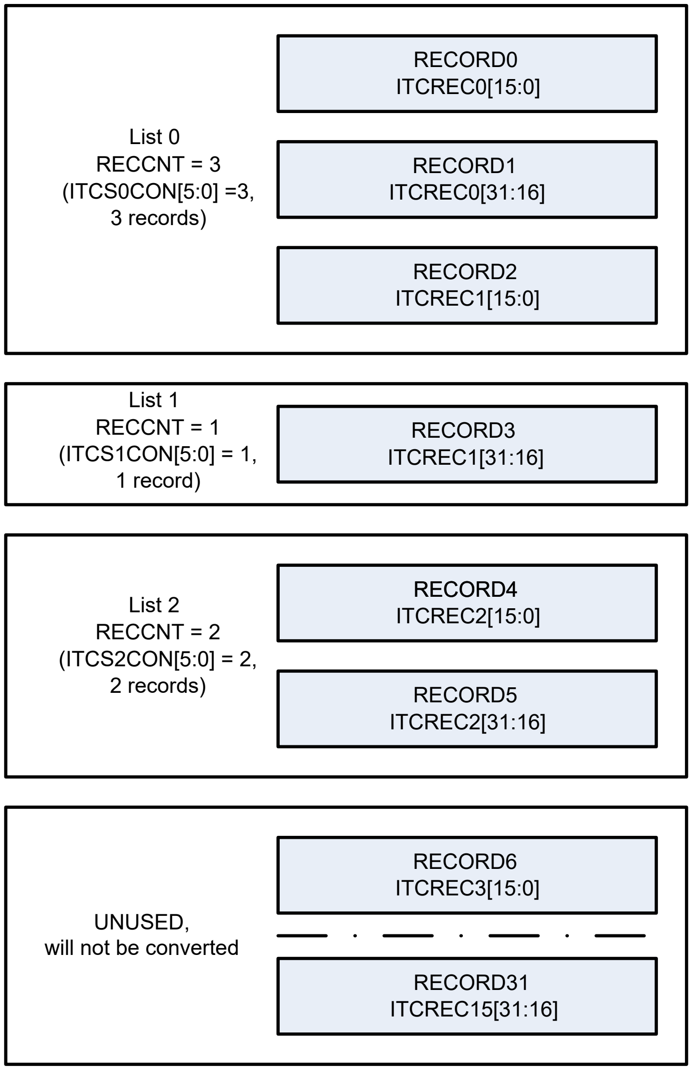
17.4.5. Analog Multiplexor Mode

Several CVDANx pins can be connected together when the MULEN bit is set in the ITCLSxCON register. This allows for the implementation of an analog multiplexer. The connected ANx inputs are selected by bits in the ITCLSxMUL register.

17.4.6. Records and Lists

The record is an element of a processing list. It contains settings to map the CVDANx pin and active CVD guard pins. The records are stored in pairs in ITCRECx registers. All records are distributed among all available lists. The number of records the list converts is defined by the RECCNT bits in the list control register ITCLSxCON. The records assignment starts from List 0 and ends with the last list. For example, three lists are implemented in ITC. List 0 has 3 records; List 1 has one, and List 2 includes 2 records. The records assignment will be as shown in Figure 17-7. List 0 will trigger and process RECORD0, RECORD1, and RECORD2; List 1 will process only RECORD3, and List 2 will process RECORD4 and RECORD5. All other records are unused and will not be converted.

Figure 17-7. Records Assignment Example



The records in the list are processed one by one. The NEXT[5:0] bits in the ITCLSxSTAT register indicate the number of the record to be processed next.

17.4.7. Sampling, Balance, Conversion and Sequencers Timing

The ITC, including the acquisition and math sequencers, runs from the ADC 3 T_{AD} clock. The typical ADC 3 clock frequency is 80 MHz. Therefore, each sequencer command takes 12.5 ns. The CVD charge balance time is controlled in the acquisition commands. Also, the DLYCNT[4:0] bits of the ITCLSxCON register can be used to define the balance delay in the ADC 3 T_{AD} steps (12.5 ns typical). The balance time is started using the BAL bit in the acquisition sequence command. The acquisition sequencer can wait until the time defined by the DLYCNT[4:0] bits has elapsed when LOOP[3:0] bits = 3 in the acquisition sequencer command. The conversion of an analog level on the internal CVD capacitors array is done by ADC 3. The ADC 3 conversion takes two T_{AD} cycles (or 25 ns typical). The conversion is triggered by the CONV bit in the acquisition sequence command. The command with the CONV bit set sends a trigger to ADC 3. The acquisition sequencer can wait until the conversion ends when LOOP[3:0] bits = 4 in the acquisition sequencer command.

17.4.8. Result

When each record is processed by the math sequencer, the math sequencer result can be stored as defined by Write mode bits WM[1:0] in the ITCLSxCON register. Also, ITCLSxCON WM[1:0] bits can be overwritten/replaced with other ITCSMATHCMDx WM[1:0] settings in the math sequencer command. These bits in the math command overwrite the list register settings WM[1:0] only when the WMOV bit = 1 in the math command. The following write result options are available:

- WM[1:0] = 0 means "always write."
- WM[1:0] = 1 means that the math result is stored only when a comparator event is detected.
- WM[1:0] = 2 means "never write."

For each record, each math sequencer command stores the result in the corresponding ITCRESx register as defined by the WM[1:0] bits. The data processed by the math sequencer can be in unsigned or signed formats. The format is selected by the SIGN bit in the ITCCON1 register. SIGN bit = 0 means unsigned, and SIGN bit = 1 means signed format. The CVD algorithm accumulates the difference between Sample A and B. In the case of CVD, the SIGN bit must always be set.

All data in the math sequencer are 24-bit in size. Bit 23 is a sign bit. When the math sequencer's signed result is stored into the 32-bit ITCRESx register, the sign bit is extended automatically.

17.4.9. List Triggers and Interrupts

Each list defines one trigger source to process all records. Depending on the mode, one trigger can start the back-to-back processing of all records in the list (scan), or each record can be triggered separately, one-by-one. For indication, the TACK bit in the ITCLSxSTAT register is set during the trigger processing.

The record is assumed to be processed if all accumulations for the record are done as defined by the ACCNUM[7:0] bits in the ITCLSxSEQ register.

The ITC module has a common interrupt for all lists. The common interrupt is generated when any list interrupt occurs. Depending on the mode, the interrupts can be disabled, generated for each record, or generated when the entire list has been processed.

In addition, the interrupts can be generated by the acquisition sequencer command when the INT bit is set.

The status interrupt flags are available for each list. The INT sticky flags are set in the ITCLSxSTAT registers when list interrupts occur. The flags are set by hardware and should be cleared by software. These INT bits are mirrored in the ITCSTAT register. The list interrupt flag can be cleared by software in two places: ITCLSxSTAT or ITCSTAT.

The records list has several trigger and interrupt modes selected by the MODE[2:0] bits in the ITCLSxCON register. The following options are available:

- MODE[2:0] = 7: One trigger executes back-to-back processing for all records. The list interrupt is generated after the last list record is processed if at least one record result matches the comparator criteria.
- MODE[2:0] = 6: One trigger executes back-to-back processing for all records. The list interrupts are generated for records every time the record result matches the comparator criteria.
- MODE[2:0] = 5: One trigger executes all records in back-to-back processing. The list interrupt is generated after the last record is processed.
- MODE[2:0] = 4: One trigger executes back-to-back processing of all records. The list interrupts are not generated.
- MODE[2:0] = 2: One record is processed per trigger. The list interrupt is generated after the last record is processed.
- MODE[2:0] = 1: One record is processed per trigger. The list interrupt is generated after each record is processed.
- MODE[2:0] = 0: One record is processed per trigger. The interrupts are not generated by the list.

The trigger source for the list is defined in the TRGSRC[4:0] bits of the ITCLSxCON register. The list triggering is enabled by setting the TRGEN bit (ITCLSxCON[15]). The trigger enable bits are mirrored in the ITCCON2 register. The list trigger can be enabled in two places: ITCLSxCON or ITCCON2.

The ITC module has an internal timer to generate periodic triggers. The timer period is set by TMRPR[15:0] bits in the ITCCON2 register, and the timer is clocked from the ADC3/ITC clock. The trigger option for the internal timer is TRGSRC[4:0] bits = 7.

17.4.10. Comparator

Each list has a result comparator. The result of the comparison must be stored in Accumulator A. The comparison criteria are selected by the CM[2:0] bits in the ITCLSxCON register. The following comparison options are available:

- CM[2:0] = 4: Outside Window (Accumulator A < ITCLSxCMPLO and Accumulator A >ITCLSxCMPHI).
- CM[2:0] = 3: Inside Window (Accumulator A >ITCLSxCMPLO and Accumulator A < ITCLSxCMPHI).
- CM[2:0] = 2: Greater Than (Accumulator A >ITCLSxCMPHI).
- CM[2:0] = 1: Match Less Than (Accumulator A < ITCLSxCMPLO).
- CM[2:0] = 0: Comparison is disabled.

The result to be compared should be stored in Accumulator A of the math sequencer. The data in Accumulator A is compared with thresholds defined in the ITCLSxCMPLO and ITCLSxCMPHI registers. The comparison is executed when the CMP bit is set in the math sequencer command.

If the Accumulator A value matches a selected criterion in the CM[2:0] bits, a sticky flag in the ITCHIT register is set. The hardware sets the ITCHIT bits, but they must be cleared by software.

The comparison event may affect storing a result in the ITCRESx register if this is defined by the write mode bits in the ITCLSxCON register or math sequencer command.

The comparison event may control the ITC interrupt behavior if this is defined in the MODE[2:0] bits of the ITCLSxCON register.

17.4.11. Accumulation of Several Measurements.

The ITC module can repeat conversions for each list record (assigned to the CVDANx pin) several times and accumulate the result before storing it in the ITCRESx registers.

The ITC will run a sequence multiple times for a record with the assigned CVDANx pin as it is defined in the ACCNUM[7:0] bits of the ITCLSxSEQ register.

The number of accumulated ADC measurements is ACCNUM[7:0].

When the ACCNUM[7:0] bits are zero, the ITC sequence is done just once (without accumulations).

When all accumulations are done, the ACCDONE bit is set in the ITCRECx register. ACCDONE bits in the records registers are cleared when the corresponding ITCRESx register is read.

Two accumulation modes are available. The accumulation mode is controlled by the ACCMODE bit in the ITCLSxSEQ register. When the ACCMODE bit is set (='1'), each record is accumulated multiple times back-to-back before processing the next record in the one list scan. When the ACCMODE bit is cleared (='0'), multiple scans are executed, and the records' results are accumulated between scans.

17.4.12. CVD Capacitors Array

The ITC has an internal CVD capacitor with a varied capacitance (0 pF-17.5 pF). The CVD capacitor value can be adjusted using the CVDCAP[2:0] bits in the ITCLSxSEQ register with 2.5 pF steps. To achieve the maximum CVD sensitivity, the internal CVD capacitance should be as close as possible to the sensor’s capacitance. The internal CVD capacitor is enabled by the CVDEN bit in the ITCCON1 register. The CVDEN bit must always be set during ITC initialization for the CVD operation. The CVD capacitor can be connected to ground (discharged) when the DISCHRG bit is set in the acquisition sequence command, and the CVD capacitor can be connected to power (charged) when the CHRG bit is set in the acquisition sequence command. The CVD capacitor is connected to the CVDANx pin (sensor) to balance the charge when the BAL bit is set in the acquisition sequence command.

17.4.13. Test Mode

The Test mode allows testing of the ITC controller. This mode is enabled when the TSTEN bit in the ITCSTAT register is set. When enabled, the result of any conversion is overwritten with a value in the TSTDATA[15:0] bits of the ITCSTAT register.

17.4.14. ITC Enable

The ITC is clocked from ADC 3, and the ITC uses Channel 5 of ADC 3 to convert the analog signal. The ADC 3 input 5, connected to the CVD capacitors array, must be selected in the ADC for the ITC operation. ITC triggers ADC 3 when the Channel 5 trigger source is set to option 12 (ITC). Therefore, to enable ITC, ADC 3 must be initialized first.

The ITC is enabled and can execute CVD scans when the following three bits are set:

- ON bit = `1` in ITCCON1 register (ITC enable)
- CVDEN bit = `1` in ITCCON1 register (CVD capacitor array enable)
- DATASEQ[2:0] bits = 1 in ITCLSxSEQ register (CVD sequence)

When the ON bit is set, the ready bit DRDY in the ITCSTAT register must be polled in software. The DRDY bit is set by hardware when the ITC module is ready for operation.

The example of the ADC 3 initialization and ITC enable code is listed in Example 18-1.

Example 17-1. ADC 3 Initialization and ITC Enable Code

```
 // Initialize ADC 3 for operation with ITC
 AD3CH5CON1bits.MODE = 0;             // single conversion
 AD3CH5CON1bits.IRQSEL = 0;           // each conversion interrupt
 AD3CH5CON1bits.PINSEL = 5;           // AD3AN5 is connected to CVD capacitors
array
 AD3CH5CON1bits.NINSEL = 0;           // VSS
 AD3CH5CON1bits.SAMC = 0;             // small balance time
 AD3CH5CON1bits.TRG1SRC = 12          // trigger for ITC
 AD3CONbits.ON = 1;                   // enable ADC 3
 while (AD3ONbits.ADRDY == 0);        // wait for ready
 // Switch on the ITC
 ITCCON1bits.ON = 1; // enable ITC
 while(ITCSTAbits.DRDY == 0);         // wait for ITC ready
 ITCCON1bits.CVDEN = 1;               // enable CVD capacitors array
 ITCLS0SEQ.ACCEN = 1;                 // enable accumulations for List # 0
 ITCLS0SEQ.DATASEQ = 1;               // CVD sequence for List # 0
```

17.4.15. Sequencers

When the list record is triggered, a sequence of microcommands is executed for this record. The microcommands are called sequences and are divided into two types: DCS (data collection or acquisition sequence) and MS (math sequence). The acquisition and math sequences are executed in the order shown in Figure 17-8 and Figure 17-9.

Figure 17-8. Execution of Acquisition and Math Sequences when ACCMOD bit = 0 in ITCLSxCSEQ register

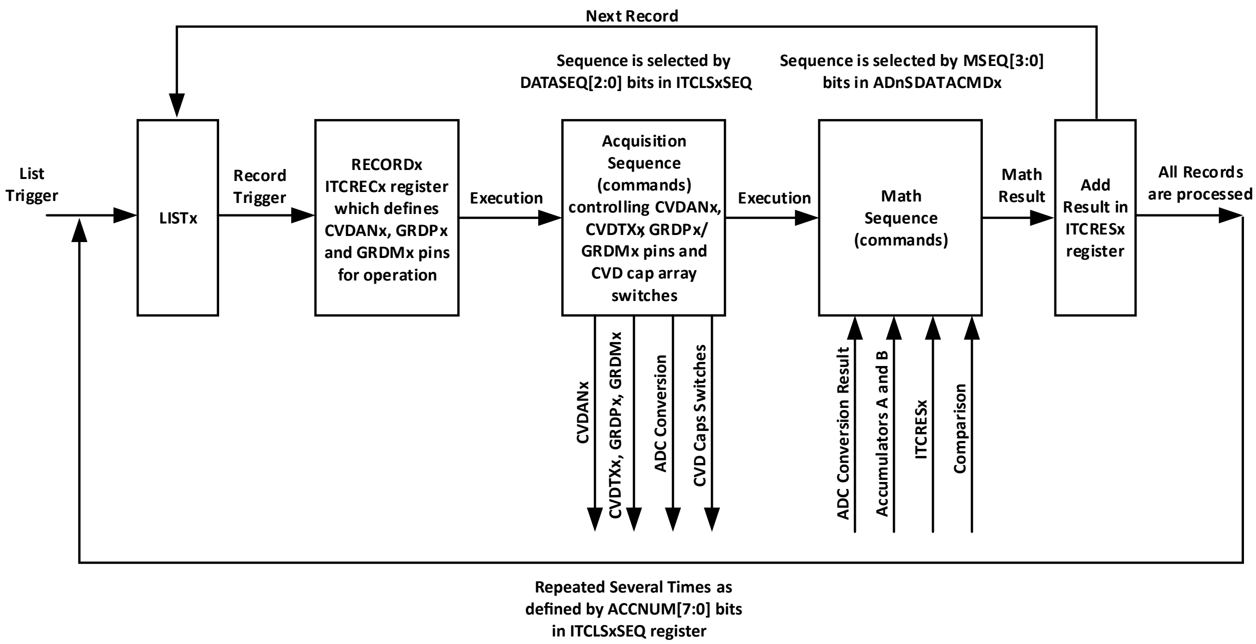
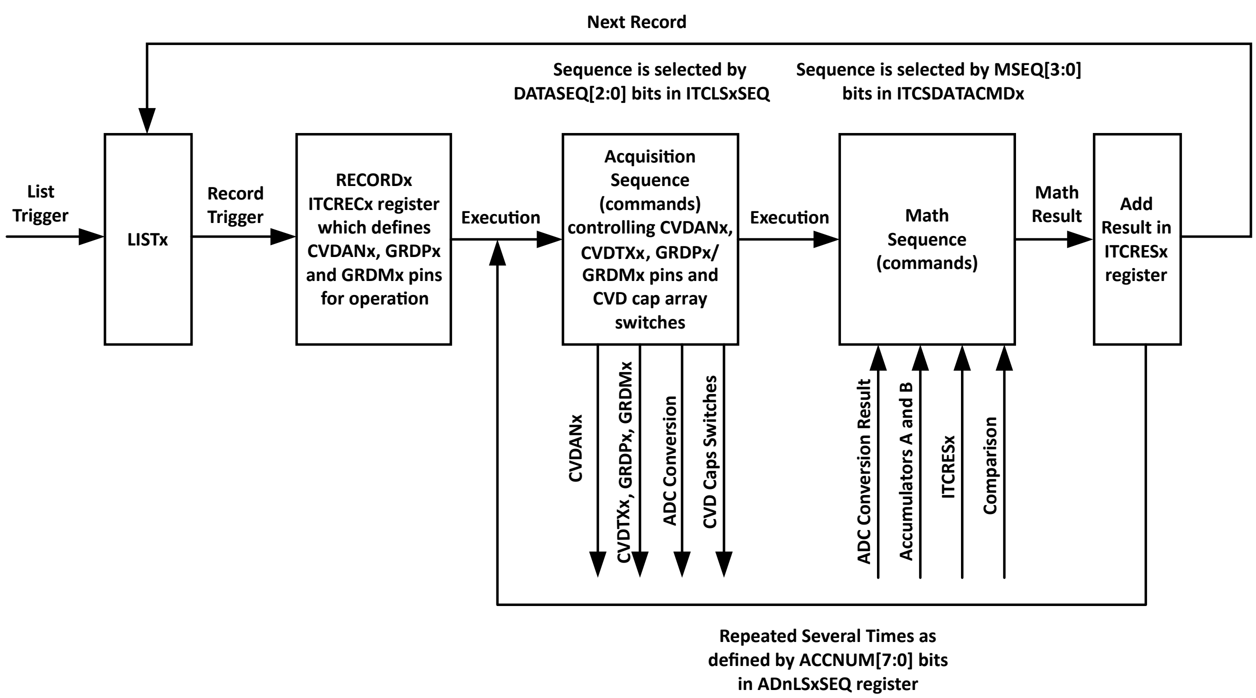


Figure 17-9. Execution of Acquisition and Math Sequences when ACCMOD bit = 1 in ITCLSxSEQ register



There are two hardware predefined sequences: to make ADC conversion (DATASEQ bits = `0` in the ITCLSxSEQ register) and to run CVD (DATASEQ bits = `1` in the ITCLSxSEQ register). Also, the custom sequences commands can be programmed (initialized) by the software in the register's arrays. One array (16 of ITCSDATACMDx command registers) is dedicated for the acquisition sequences, and another array (16 of ITCSMATHCMDx command registers) contains the math sequences. These arrays can be split to store several sequences. For the acquisition commands, they are defined by SPLITx[1:0] bits in the ITCSDATAMAP register, and for the math commands, they are defined by

SPLITx[1:0] bits in the ITCSMATHMAP register. The SPLITx[1:0] bits allow dividing all commands in the arrays into a few sequences, as shown in Table 17-2.

Table 17-2. Dividing the Commands Arrays into a few Sequences using SPLIT[1:0] Bits

Sequencies	Command registers and control SPLIT bits in ITCSDATAMAP and ITCSMATHMAP registers			
	0-3	4-7	8-11	12-15
	SPLIT0[1:0]	SPLIT1[1:0]	SPLIT2[1:0]	SPLIT3[1:0]
One 16 steps sequence (registers: 0-15)	0	0	0	0
Two 8 steps sequences (registers: 0-7 and 8-15)	2	2	2	2
One 8 step and two 4 steps sequences (registers: 0-7, 8-11 and 12-15)	2	2	3	3
Four 4 steps sequences (registers: 0-3, 4-7, 8-11 and 12-15)	3	3	3	3

When the acquisition commands array is split into a few sequencies, the commands in each sequence should be assigned to a sequence number set in the DATASEQ[2:0] bits of the ITCLSxSEQ register. The user can select any unique sequence number from 4 to 7 to identify the sequence. The acquisition sequence number settings example is shown in Example 17-2.

Example 17-2. The Acquisition Sequence Number Assignments

```
// Split all commands in 3 sequences
ITCSDATAMAPbits.SPLIT0 = 2;
ITCSDATAMAPbits.SPLIT1 = 2;
ITCSDATAMAPbits.SPLIT2 = 3;
ITCSDATAMAPbits.SPLIT3 = 3;

// List 0 will be processed by sequence number 5
ITCLS0SEQbits.DATASEQ = 5;
// Assign first 8 steps sequence to number 5 to be processed by list 0
ITCSDATAMAPbits.DATASEQ0 = 5;
ITCSDATAMAPbits.DATASEQ1 = 5;

// List 1 will be processed by sequence number 6
ITCLS1SEQbits.DATASEQ = 6;
// Assign second 4 steps sequence to number 6 to be processed by list 1
ITCSDATAMAPbits.DATASEQ2 = 6;

// List 2 will be processed by sequence number 7
ITCLS2SEQbits.DATASEQ = 7;
// Assign third 4 steps sequence to number 7 to be processed by list 2
ITCSDATAMAPbits.DATASEQ3 = 7;
```

When the math commands array is split into a few sequences, the commands in each sequence should be assigned to a sequence number set in the MSEQ[3:0] bits of the ITCSDATACMDx command register. The user can select any unique sequence number from 4 to 7 to identify the sequence.

The math sequence number settings example is shown in Example 17-3. In this example, the acquisition commands are configured to execute two 8 step math sequences number 5 and 6.

Example 17-3. Math Sequence Number Assignments

```
// Split all commands in 2 sequences
ITCSMATHMAPbits.SPLIT0 = 2;
ITCSMATHMAPbits.SPLIT1 = 2;
ITCSMATHMAPbits.SPLIT2 = 2;
ITCSMATHMAPbits.SPLIT3 = 2;

// First 8 steps sequence is 5
ITCSMATHMAPbits.MATHSEQ0 = 5;
ITCSMATHMAPbits.MATHSEQ1 = 5;
```

```
// Second 8 steps sequence is 6
ITCSMATHMAPbits.MATHSEQ2 = 6;
ITCSMATHMAPbits.MATHSEQ3 = 6;

// This data acquisition command will execute the math sequence 5
ITCSDATACMD0bits.MSEQ = 5;
ITCSDATACMD0bits.MSTART = 1;

// This data acquisition command will execute the math sequence 6
ITCSDATACMD1bits.MSEQ = 6;
ITCSDATACMD1bits.MSTART = 1;
ITCSDATACMD1bits.END = 1;
```

17.4.16. Acquisition Sequencer Commands

The acquisition sequence is executed each time the record is triggered.

The sequence is finished when the END bit is set.

The command can delay the execution as specified in the LOOP[3:0] bits.

Each acquisition sequence must call the math sequence. It is done if the MSTART bit is set in the command. The math sequence to be executed is specified by the MSEQ[3:0] bits of the acquisition command.

The C_{HOLD} and CVD capacitors array, when the array is enabled, are controlled by the DISCHRG (discharge to V_{SS}), CHRG (charge to V_{DD}) and BAL (connect to ANx pin) bits.

When the BAL bit is set to connect the C_{HOLD} and CVD capacitors to the CVDANx pin, the four list timers in the ITCLSxTMR register can be used to provide a sampling/settle delay required. The timer to wait is selected by the LOOP[3:0] bits. In addition, the option LOOP[3:0] = 3 allows using the DLYCNT[4:0] bits in the ITCLSxCON as a timer.

The CONV bit starts a conversion of an analog level on C_{HOLD} (and CVD capacitor, if enabled). The command can wait for the end of the conversion if LOOP[3:0] bits = 4. The BAL bit must always be set when the conversion is started (CONV bit = '1').

The CVDANx pin is controlled by the PC0[1:0] bits. The following options are available:

- 0 – The CVDANx pin is controlled by the corresponding TRIS and LAT bits.
- 1 – The CVDANx pin is driven to a low level.
- 2 – The CVDANx pin is driven to a high level.
- 3 – The CVDANx pin is tri-stated.

In addition, the acquisition command can control up to three CVDTXx pins. These pins can be used to implement the mutual capacitance low impedance signal or active guards.

The PCA[1:0] bits control CVDTXx pins assigned in the ITCTXA register.

The PCB[1:0] bits control CVDTXx pins assigned in the ITCTXB register.

The PCC[1:0] bits control CVDTXx pins assigned in the ITCTXC register.

The available options are the same as ANx pin control:

- 0 – The CVDTXx pin is controlled by the corresponding TRIS and LAT bits.
- 1 – The CVDTXx pin is driven to a low level.
- 2 – The CVDTXx pin is driven to a high level.
- 3 – The CVDTXx pin is tri-stated.

The acquisition command may generate DMA triggers. The DMA triggers must be enabled in the ITCLSxCON register using the DMAEN bit. When the DMATXA bit is set, the command asserts a DMATXA trigger. By this request, the DMA must write into the ITCTXA register to clear the trigger. If this write is not done, the next DMA trigger will not be processed. When the DMATXB bit is set, the command asserts a DMATXB trigger. By this request, the DMA must write into the ITCTXB register to clear the trigger. If this write is not done, the next DMA trigger will not be processed. When the DMATXC bit is set, the command asserts a DMATXC trigger. By this request, the DMA must write into the ITCTXC register to clear the trigger. If this write is not done, the next DMA trigger will not be processed. When the DMASTP bit is set, the command asserts a DMASTP trigger. By this request, the DMA must write into any ITCSDATACMDx register to clear the trigger. If this write is not done, the next DMA trigger will not be processed. The DMASTP trigger allows overwriting the acquisition sequence commands on the fly. It allows using long frequencies which cannot fit in the 16 available command registers.

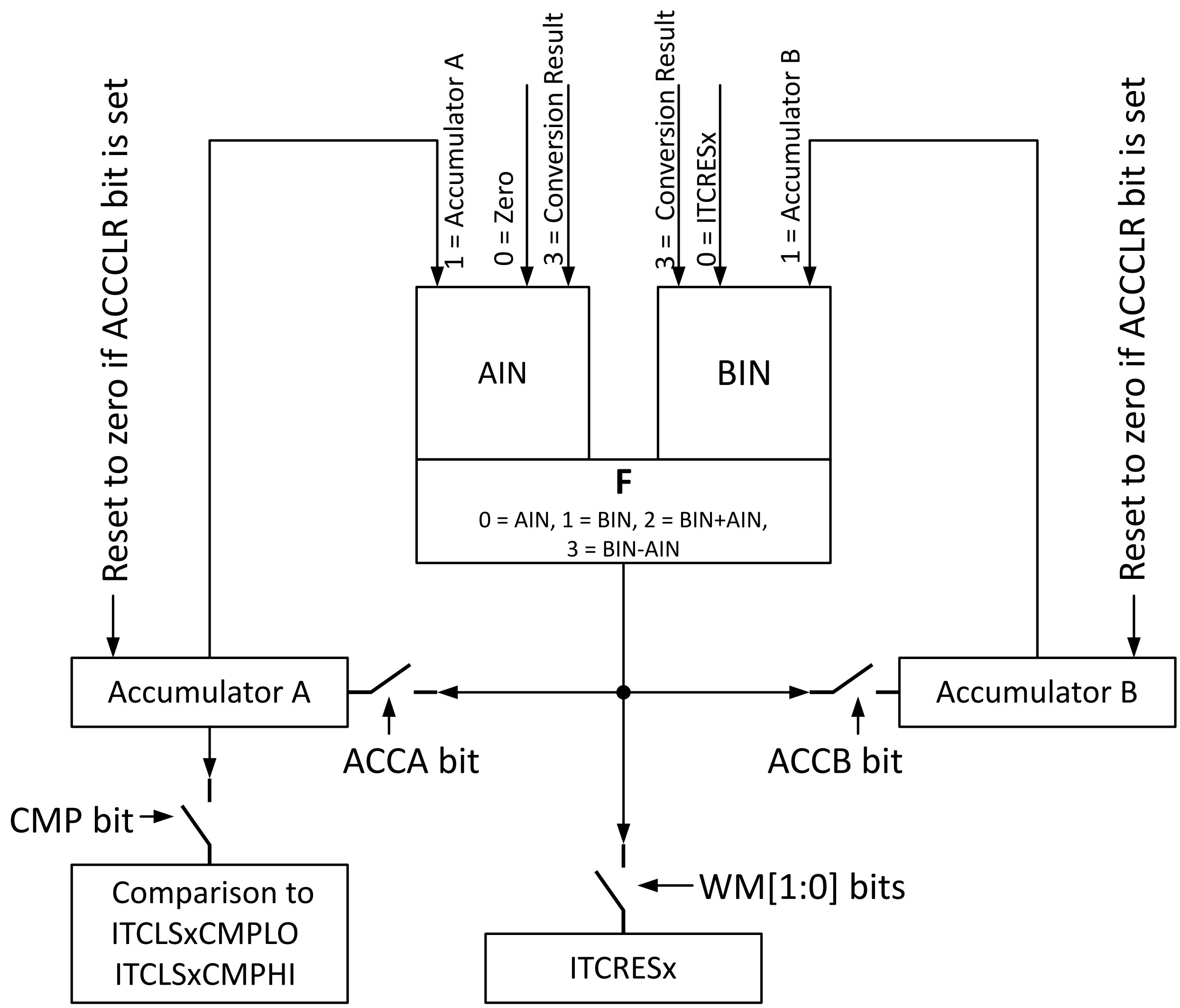
If the record processing should be done a few times (accumulated, as specified by the ACCNUM[7:0] bits in the ITCLSxSEQ register), the DMA triggers (DMATXA, DMATXB, DMATXC, and DMASTP) may be sent only when the last accumulation of the record is finished. When set, the DMALACC bit allows the trigger defined by the DMATXA, DMATXB, DMATXC or DMASTP bits only when the last accumulation is done.

The records in the list are processed/scanned one-by-one. When set, the DMALAST bit allows the generation of DMATXA, DMATXB, DMATXC or DMASTP triggers only when the list scan is complete (the sequence is executed for the last time for the list).

17.4.17. Math Sequencer Commands

The math post processing sequence is called by an acquisition sequence. The internal structure of the math hardware is shown in Figure 17-10.

Figure 17-10. Math Sequence Hardware Block Diagram



The sequence is finished when the END bit is set.

The arithmetic module has two inputs: AIN and BIN. For input A (AIN[1:0] bits), the following sources can be selected:

- `0` = Zero
- `1` = Accumulator A
- `2` = Reserved
- `3` = Conversion Result

For input B (BIN[1:0] bits), the following sources can be selected:

- `0` = ITCRESx register
- `1` = Accumulator B
- `2` = Reserved

- 3 = Conversion Result

The math sequence command executes a math operation as specified by the F[1:0] bits. The following options are available:

- 0 = data from A input is copied to the math result.
- 1 = data from B input is copied to the math result.
- 2 = the math result is a sum of B and A inputs.
- 3 = the math result is a difference between B and A inputs.

The output of the math operation is stored in the ITCRESx register if it is defined by the WM[1:0] bits of the ITCLSxCON register or by the WM[1:0] bits in the ITCSMATHCMDx command when the WMOV bit is set. Also, the math result can be latched into two accumulators when the ACCA and/or ACCB bits are set.

Once the ACCCLR bit is set, Accumulator A and Accumulator B are reset to zero.

The FIRST bit of the math command can be used for the ITCRESx registers reset/initialization. The FIRST bit allows setting the arithmetic module input B to zero when the record is processed for the very first time in the list scan. When the FIRST bit is set ('=1'), the input B is zero for the first record accumulation regardless of the BIN[1:0] bits settings.

When the CMP bit is set, a comparison of the Accumulator A executes with the ITCLSxCMPHI and ITCLSxCMPLO thresholds as defined by the CM[2:0] bits in the ITCLSxCON register.

The math command can generate the interrupt when the INT bit is set (='1').

The ACCCLR, CMP and INT bits can be enabled only when the math sequence is executed for the last record accumulation in the list scan. To do this, the LAST bit must be set (='1').

17.4.18. Hardware Coded Sequences

Two sequences are predefined in the hardware.

When DATASEQ[2:0] bits = 0 in the ITCLSxSEQ register, the ITC module just converts the voltage on the ANx input. The accumulations when ACCNUM[7:0] bits are zero and comparison to the thresholds in the ITCLSxCMPLO and ITCLSxCMOHI registers can be enabled in the CM[2:0] bits of the ITCLSxCON register. The results are stored in the ITCRESx registers.

When DATASEQ[2:0] bits = 1 in the ITCLSxSEQ register, the ITC module executes a basic CVD procedure for ANx input (Sample A – Sample B). The software trigger (ITCLSxCON.TRGSRC[4:0] = 0) cannot be used to start CVD. The SAMP bit in the ITCLSxCON register must be zero when the CVD sequence is selected. The CVDTXx pins selected in the ITCTXA registers are toggled in phase with the charge and discharge levels of the sensor. The CVDTXx pins selected in the ITCTXB register are toggled opposite to the charge and discharge levels of the sensor. The accumulations are controlled by ACCNUM[7:0] bits, and comparison to the thresholds in the ITCLSxCMPLO and ITCLSxCMOHI registers can be enabled in the CM[2:0] bits of the ITCLSxCON register. The results are stored in the ITCRESx registers.

17.5. Application Examples

17.5.1. CVD Scan of Three Analog Inputs Using a Hardcoded Processing Sequence

In the following example, the ITC runs CVD on three analog inputs.

```
#include <xc.h>

// All pins CVD results.
long result[3];

int main(){

    PLL1CONbits.ON = 1;
```

```
    OSCCTRLbits.PLL1EN = 1;
    while(OSCCTRLbits.PLL1RDY == 0);

    PLL1DIVbits.PLLFBDIV = 100; // VCO = 800 MHz
    PLL1DIVbits.PLLPRE = 1;
    PLL1DIVbits.POSTDIV1 = 4;
    PLL1DIVbits.POSTDIV2 = 1;
    PLL1CONbits.DIVSWEN = 1;
    while(PLL1CONbits.DIVSWEN == 1);
    PLL1CONbits.NOSC = 1; // FRC
    PLL1CONbits.OSWEN = 1;
    while(PLL1CONbits.OSWEN == 1);
    PLL1CONbits.FOUTSWEN = 1;
    while(PLL1CONbits.FOUTSWEN == 1);
    PLL1CONbits.PLLSWEN = 1;
    while(PLL1CONbits.PLLSWEN == 1);
    while(PLL1CONbits.CLKRDY == 0);


    // CPU clock 200MHz
    CLK1CONbits.ON = 1;
    CLK1CONbits.NOSC = 5; // PLL1
    CLK1CONbits.OSWEN = 1;
    while(CLK1CONbits.OSWEN == 1);
    while(CLK1CONbits.CLKRDY == 0);

    // ADC (Generator 6) clock 200 MHz
    CLK6CONbits.ON = 1;
    CLK6CONbits.NOSC = 5; // PLL1
    CLK6CONbits.OSWEN = 1;
    while(CLK6CONbits.OSWEN == 1);
    while(CLK6CONbits.CLKRDY == 0);

    // Initialize ADC 3 for operation with ITC
    AD3CH5CON1bits.MODE = 0; // single conversion
    AD3CH5CON1bits.IRQSEL = 0; // each conversion interrupt
    AD3CH5CON1bits.PINSEL = 5; // AD3AN5 is connected to CVD capacitors array
    AD3CH5CON1bits.NINSEL = 0; // VSS
    AD3CH5CON1bits.SAMC = 0; // small balance time
    AD3CH5CON1bits.TRG1SRC = 12; // trigger for ITC
    AD3CONbits.ON = 1; // enable ADC 3
    while (AD3ONbits.ADRDY == 0); // wait for ready

    // List 0 will be processed by hardware sequence number 1 (CVD)
    ITCLS0SEQbits.DATASEQ = 1;
    // Each sequence will be executed 8 times per record
    ITCLS0SEQbits.ACCNUM = 8-1;
    ITCLS0SEQbits.ACCMODE = 1; // back to back

    // Set CVD capacitor to 17.5 pF (maximum)
    ITCCON1bits.CVDEN = 1;
    ITCLS0SEQbits.CVDCAP = 7;

    // AN pins are grounded (LAT = 0) when idle
   TRISAbits.TRISA0 = 0; //AN0
   TRISAbits.TRISA1 = 0; //AN1
   TRISAbits.TRISA2 = 0; //AN2

    // Assign AN pins to the records
    ITCREC0bits.PIN0 = 0; // AN0 pin
    ITCREC0bits.PIN1 = 1; // AN1 pin
    ITCREC1bits.PIN2 = 2; // AN2 pin

    // Set timers for the acquisition sequences (A and B are used only)
    ITCLS0TMRbits.TMRA = 16; // charge/discharge
    ITCLS0TMRbits.TMRB = 32; // balance

    ITCCON2bits.CLKSEL = 1; // Clock from Generator 6 (200MHz)
    ITCCON2bits.CLKDIV = 4; // Divide by 8 to get 25 MHz ADC clock

    // One trigger scans all records.
    // One interrupt is generated after scan
    ITCLS0CONbits.MODE = 5;
    ITCLS0CONbits.WM = 0; // always write to ITCRES registers
    ITCCON2bits.TMRPR = 0xffff; // maximum period of internal timer
    ITCLS0CONbits.TRGSRC = 7; // internal timer trigger
    ITCLS0CONbits.RECCNT = 3; // 3 records are in list 0
    ITCLS0CONbits.TRGEN = 1; // trigger enable
```

```
    ITCCON1bits.SIGN = 1; // signed format is required for CVD
    ITCCON1bits.ON = 1; // enable ADC 2
    while(ITCSTATbits.DRDY == 0);

    while(1){ // scan in infinite loop
        _ITCIF = 0; // clear ITC interrupt flag
        while(_ITCIF == 0); // wait for the scan is done
        // store the results
        result[0] = ITCRES0;
        result[1] = ITCRES1;
        result[2] = ITCRES2;
    }
    return 1;
}
```

17.5.2. CVD Scan of All Analog Inputs Using a Custom Processing Sequence Defined in Software

In the following example, the ITC is initialized to perform CVD scans of all AN pins. In each scan, a new ITCTX pin is selected by DMA.

```
#include <xc.h>

// All pins CVD results.
// 32 ITCAN pins are scanned for 32 different ITCTX pins settings.
long result[32][32];

long* pointer; // Used to store the results.
long cvd_counter; // CVD scans counter.
long tx_counter; // TX settings counter.

// OPTIONAL
// ITCTX pins selections for each CVD scan.
// Moved by DMA after each scan in ITCTXA register.
unsigned long cvd_tx_pins[32] = {
0x2, 0x4, 0x8, 0x10, 0x20, 0x40, 0x80, 0x100,
0x200, 0x400, 0x800, 0x1000, 0x2000, 0x4000, 0x8000, 0x10000,
0x20000, 0x40000, 0x80000, 0x100000, 0x200000, 0x400000, 0x800000, 0x1000000,
0x20000, 0x40000, 0x80000, 0x100000, 0x200000, 0x400000, 0x800000, 0x1
};

int main(){

    PLL1CONbits.ON = 1;
    OSCCTRLbits.PLL1EN = 1;
    while(OSCCTRLbits.PLL1RDY == 0);

    PLL1DIVbits.PLLFBDIV = 100; // VCO = 800 MHz
    PLL1DIVbits.PLLPRE = 1;
    PLL1DIVbits.POSTDIV1 = 4;
    PLL1DIVbits.POSTDIV2 = 1;
    PLL1CONbits.DIVSWEN = 1;
    while(PLL1CONbits.DIVSWEN == 1);
    PLL1CONbits.NOSC = 1; // FRC
    PLL1CONbits.OSWEN = 1;
    while(PLL1CONbits.OSWEN == 1);
    PLL1CONbits.FOUTSWEN = 1;
    while(PLL1CONbits.FOUTSWEN == 1);
    PLL1CONbits.PLLSWEN = 1;
    while(PLL1CONbits.PLLSWEN == 1);
    while(PLL1CONbits.CLKRDY == 0);


    // CPU clock 200MHz
    CLK1CONbits.ON = 1;
    CLK1CONbits.NOSC = 5; // PLL1
    CLK1CONbits.OSWEN = 1;
    while(CLK1CONbits.OSWEN == 1);
    while(CLK1CONbits.CLKRDY == 0);

    // ADC (Generator 6) clock 200 MHz
    CLK6CONbits.ON = 1;
    CLK6CONbits.NOSC = 5; // PLL1
    CLK6CONbits.OSWEN = 1;
    while(CLK6CONbits.OSWEN == 1);
    while(CLK6CONbits.CLKRDY == 0);

    // Initialize ADC 3 for operation with ITC
```

```
    AD3CH5CON1bits.MODE = 0; // single conversion
    AD3CH5CON1bits.IRQSEL = 0; // each conversion interrupt
    AD3CH5CON1bits.PINSEL = 5; // AD3AN5 is connected to CVD capacitors array
    AD3CH5CON1bits.NINSEL = 0; // VSS
    AD3CH5CON1bits.SAMC = 0; // small balance time
    AD3CH5CON1bits.TRG1SRC = 12; // trigger for ITC
    AD3CONbits.ON = 1; // enable ADC 3
    while (AD3ONbits.ADRDY == 0); // wait for ready


    // OPTIONAL
    // Initialize DMA channel to update ITCTX pins by
    // ADC 2 list #0 interrupt after each scan
    // (optional, may be used for 2D CVD).
    DMALOW = 0; // any address
    DMAHIGH = -1UL; // any address
    DMA0CHbits.SIZE = 2; // 32 bits
    DMA0CHbits.TRMODE = 1; // repeated one shot
    DMA0CHbits.DAMODE = 0; // not incremented
    DMA0CHbits.SAMODE = 1; // take next data each transfer
    DMA0CHbits.RELOADC = 1; // reload counter
    DMA0CHbits.RELOADS = 1; // reload source
    DMA0CHbits.CHEN = 1; // enable channel
    DMA0SELbits.CHSEL = 66; // trigger DMA from ADC 2 interrupt
    DMA0DST = (unsigned long)&ITCTXA; // update pins in ITCTXA register
    DMA0SRC = (unsigned long)&cvd_tx_pins[0];
    DMA0CNT = 32; // 32 ITCTX pins updates before rollover
    DMACONbits.ON = 1; // enable DMA

    ITCTXA = 0x1; // start from TX0 pin

    // List 0 will be processed by sequence number 5
    ITCLS0SEQbits.DATASEQ = 5;
    // Each sequence will be executed 4 times per record
    ITCLS0SEQbits.ACCNUM = 4-1;
    ITCLS0SEQbits.ACCMODE = 1; // back to back

    // Use all 16 acquisition commands for one sequence
    ITCSDATAMAPbits.SPLIT0 = 0;
    ITCSDATAMAPbits.SPLIT1 = 0;
    ITCSDATAMAPbits.SPLIT2 = 0;
    ITCSDATAMAPbits.SPLIT3 = 0;

    // Assign sequence to number 5 to be processed by list 0
    ITCSDATAMAPbits.DATASEQ0 = 5;
    ITCSDATAMAPbits.DATASEQ1 = 5;
    ITCSDATAMAPbits.DATASEQ2 = 5;
    ITCSDATAMAPbits.DATASEQ3 = 5;

    // Split 16 math commands in 2 sequences
    ITCSMATHMAPbits.SPLIT0 = 2;
    ITCSMATHMAPbits.SPLIT1 = 2;
    ITCSMATHMAPbits.SPLIT2 = 2;
    ITCSMATHMAPbits.SPLIT3 = 2;

    // First 8 sequence steps are assigned to math sequence # 5
    ITCSMATHMAPbits.MATHSEQ0 = 5;
    ITCSMATHMAPbits.MATHSEQ1 = 5;

    // Second 8 sequence steps are assigned to math sequence # 6
    ITCSMATHMAPbits.MATHSEQ2 = 6;
    ITCSMATHMAPbits.MATHSEQ3 = 6;


    // CVD ACQUISITION SEQUENCE
    // Command 0
    ITCSDATACMD0bits.CHRG = 1; // charge CVD capacitor
    ITCSDATACMD0bits.PC0 = 1; // drive 0 to discharge sensor on AN pin
    ITCSDATACMD0bits.PCA = 2; // drive 1 on pins selected in ITCTXA register
    ITCSDATACMD0bits.LOOP = 1; // wait for timer A in ITCLS0TMR register
    // Command 1
    ITCSDATACMD1bits.BAL = 1; // connect CVD capacitor to AN pin (balance)
    ITCSDATACMD1bits.PC0 = 3; // tri-state AN pin
    ITCSDATACMD1bits.PCA = 2; // drive 1 on pins selected in ITCTXA register
    ITCSDATACMD1bits.LOOP = 2; // wait for timer B in ITCLS0TRM register
    // Command 2
    ITCSDATACMD2bits.PC0 = 3; // tri-state AN pin
    ITCSDATACMD2bits.PCA = 2; // drive 1 on pins selected in ITCTXA register
```

```
     ITCSDATACMD2bits.BAL = 1; // Balance switch must be closed when CONV = 1
     ITCSDATACMD2bits.CONV = 1; // start conversion
     ITCSDATACMD2bits.LOOP = 4; // wait for end of conversion
     ITCSDATACMD2bits.MSEQ = 5; // math sequence # 5
     ITCSDATACMD2bits.MSTART = 1; // start the math sequence

     // Command 3
     ITCSDATACMD3bits.DISCHRG = 1; // discharge CVD capacitor
     ITCSDATACMD3bits.PC0 = 2; // drive 1 to charge sensor on AN pin
     ITCSDATACMD3bits.PCA = 1; // drive 0 on pins selected in ITCTXA register
     ITCSDATACMD3bits.LOOP = 1; // wait for timer A in ITCLS0TMR register
     // Command 4
     ITCSDATACMD4bits.BAL = 1; // connect CVD capacitor to AN pin (balance)
     ITCSDATACMD4bits.PC0 = 3; // tri-state AN pin
     ITCSDATACMD4bits.PCA = 1; // drive 0 on pins selected in ITCTXA register
     ITCSDATACMD4bits.LOOP = 2; // wait for timer B in ITCLS0TRM register
     // Command 5
     ITCSDATACMD5bits.PC0 = 3; // tri-state AN pin
     ITCSDATACMD5bits.PCA = 1; // drive 0 on pins selected in ITCTXA register
     ITCSDATACMD5bits.BAL = 1; // Balance switch must be closed when CONV = 1
     ITCSDATACMD5bits.CONV = 1; // start conversion
     ITCSDATACMD5bits.LOOP = 4; // wait for end of conversion
     ITCSDATACMD5bits.MSEQ = 6; // math sequence # 6
     ITCSDATACMD5bits.MSTART = 1; // start the first math sequence
     ITCSDATACMD5bits.END = 1; // last command, end of the sequence

     // CVD MATH SEQUENCES
     // MATH "-"/Sample B
     ITCSMATHCMD0bits.FIRST = 1; // AIN is zero for the first accumulation
     ITCSMATHCMD0bits.AIN = 3; // ADC result
     ITCSMATHCMD0bits.BIN = 1; // Accumulator B
     ITCSMATHCMD0bits.F = 3; // "-"
     ITCSMATHCMD0bits.ACCB = 1; // latch result in Accumulator B
     ITCSMATHCMD0bits.WM = 2; // never write record
     ITCSMATHCMD0bits.WMOV = 1; // overwrite WM bits in ITCLS0CON register
     ITCSMATHCMD0bits.END = 1; // last command, end of sequence

     // MATH "+"/Sample A
     ITCSMATHCMD8bits.AIN = 3; // ADC result
     ITCSMATHCMD8bits.BIN = 1; // Accumulator B
     ITCSMATHCMD8bits.F = 2; // "+"
     ITCSMATHCMD8bits.ACCB = 1; // latch result in Accumulator B
     ITCSMATHCMD8bits.WM = 0; // always write to ITCRES register
     ITCSMATHCMD8bits.WMOV = 1; // overwrite WM bits in ITCLS0CON register
     ITCSMATHCMD8bits.END = 1; // last command, end of sequence

     // Set CVD capacitor to 10pF
     ITCCON1bits.CVDEN = 1;
     ITCLS0SEQbits.CVDCAP = 4;

     // AN pins are grounded (LAT = 0) when idle
    TRISAbits.TRISA0 = 0; //CVDAN0
    TRISAbits.TRISA1 = 0; //CVDAN1
    TRISAbits.TRISA2 = 0; //CVDAN2
    TRISAbits.TRISA3 = 0; //CVDAN3
    TRISAbits.TRISA4 = 0; //CVDAN4
    TRISAbits.TRISA5 = 0; //CVDAN5
    TRISAbits.TRISA6 = 0; //CVDAN6
    TRISAbits.TRISA7 = 0; //CVDAN7
    TRISAbits.TRISA8 = 0; //CVDAN8
    TRISAbits.TRISA9 = 0; //CVDAN9
    TRISBbits.TRISB0 = 0; //CVDAN10
    TRISBbits.TRISB1 = 0; //CVDAN11
    TRISBbits.TRISB2 = 0; //CVDAN12
    TRISBbits.TRISB3 = 0; //CVDAN13
    TRISBbits.TRISB4 = 0; //CVDAN14
    TRISBbits.TRISB5 = 0; //CVDAN15
    TRISBbits.TRISB6 = 0; //CVDAN16
    TRISBbits.TRISB7 = 0; //CVDAN17
    TRISAbits.TRISA10 = 0; //CVDAN18
    TRISAbits.TRISA11 = 0; //CVDAN19
    TRISBbits.TRISB8 = 0; //CVDAN20
    TRISBbits.TRISB9 = 0; //CVDAN21
    TRISBbits.TRISB10 = 0; //CVDAN22
    TRISBbits.TRISB11 = 0; //CVDAN23
    TRISAbits.TRISA12 = 0; //CVDAN24
    TRISAbits.TRISA13 = 0; //CVDAN25
    TRISAbits.TRISA14 = 0; //CVDAN26
```

```
    TRISAbits.TRISA15 = 0; //CVDAN27
    TRISBbits.TRISB12 = 0; //CVDAN28
    TRISBbits.TRISB13 = 0; //CVDAN29
    TRISBbits.TRISB14 = 0; //CVDAN30
    TRISBbits.TRISB15 = 0; //CVDAN31

     // Assign CVDAN pins to the records
     ITCREC0bits.PIN0 = 0; // CVDAN0 pin
     ITCREC0bits.PIN1 = 1;
     ITCREC1bits.PIN2 = 2;
     ITCREC1bits.PIN3 = 3;
     ITCREC2bits.PIN4 = 4;
     ITCREC2bits.PIN5 = 5;
     ITCREC3bits.PIN6 = 6;
     ITCREC3bits.PIN7 = 7;
     ITCREC4bits.PIN8 = 8;
     ITCREC4bits.PIN9 = 9;
     ITCREC5bits.PIN10 = 10;
     ITCREC5bits.PIN11 = 11;
     ITCREC6bits.PIN12 = 12;
     ITCREC6bits.PIN13 = 13;
     ITCREC7bits.PIN14 = 14;
     ITCREC7bits.PIN15 = 15;
     ITCREC8bits.PIN16 = 16;
     ITCREC8bits.PIN17 = 17;
     ITCREC9bits.PIN18 = 18;
     ITCREC9bits.PIN19 = 19;
     ITCREC10bits.PIN20 = 20;
     ITCREC10bits.PIN21 = 21;
     ITCREC11bits.PIN22 = 22;
     ITCREC11bits.PIN23 = 23;
     ITCREC12bits.PIN24 = 24;
     ITCREC12bits.PIN25 = 25;
     ITCREC13bits.PIN26 = 26;
     ITCREC13bits.PIN27 = 27;
     ITCREC14bits.PIN28 = 28;
     ITCREC14bits.PIN29 = 29;
     ITCREC15bits.PIN30 = 30;
     ITCREC15bits.PIN31 = 31; // CVDAN31 pin

     // Set timers for the acquisition sequences (A and B are used only)
     ITCLS0TMRbits.TMRA = 16; // charge/discharge
     ITCLS0TMRbits.TMRB = 32; // balance


     ITCCON2bits.CLKSEL = 1; // Clock from Generator 6 (200MHz)
     ITCCON2bits.CLKDIV = 4; // Divide by 8 to get 25 MHz ADC clock

     // One trigger scans all records.
     // One interrupt is generated after scan
     ITCLS0CONbits.MODE = 5;
     ITCLS0CONbits.WM = 0; // always write to ITCRES registers
     ITCLS0CONbits.TRGSRC = 0; // software trigger
     ITCLS0CONbits.RECCNT = 32; // 32 records are in list 0
     ITCLS0CONbits.DMAEN = 1; // enable DMA triggers
     ITCLS0CONbits.TRGEN = 1; // trigger enable

     ITCCON1bits.SIGN = 1; // signed format is required for CVD
     ITCCON1bits.ON = 1; // enable ADC 2
     ITCLS0CONbits.SAMP = 1; // arm trigger
     while(ITCSTATbits.DRDY == 0);

     while(1){ // scan in infinite loop

         for(tx_counter=0; tx_counter<32; tx_counter++){
             _ITCIF = 0; // clear ITC interrupt flag
             ITCLS0CONbits.SAMP = 0; // trigger scan
             while(_ITCIF == 0); // wait for the scan is done
             ITCLS0CONbits.SAMP = 1; // arm for the next scan
             pointer = (long*)&ITCRES0;
             // store the results
             for(cvd_counter=0; cvd_counter<32; cvd_counter++){
                 result[tx_counter][cvd_counter] = *pointer++;
             }
         }
     }
```

```
    return 1;
}
```

18. Resolver-to-Digital Converter (RDC)

18.1. Introduction

This section describes the Resolver-to-Digital Converter (RDC) implemented in the dsPIC33AK256MPS306 family of devices. The primary function of the RDC is to produce excitation signals for the resolver and demodulate the resulting output signals. It includes a Cascaded Integrator Comb (CIC) filter for demodulation and filtering, and a Coordinate Rotation Digital Computer (CORDIC) block for computing error signals.

The following high-level features are covered in this section:

- Square wave excitation signal generator
- Resolver output signal demodulation
- Configurable signal processing via integrated second-order CIC filter
  - Digital filter for oversampling
  - Two CIC channels for filtering and demodulation
  - Detection of invalid input samples (out of range)
  - Input from internal register or external signal
- Error signal calculation based on CORDIC algorithm

18.1.1. Device-Specific Information

Table 18-1. Excitation Generator Block Summary

Number of Module Instances	Max Input Clock for Excitation Clock Generation	Clock Source	Peripheral Bus Speed
1	2.56 MHz	CLKGEN12	Standard (1:2 CPU Clock)

Table 18-2. CIC Block Summary

Number of Module Instances	Number of Channels	Peripheral Bus Speed
1	2	Standard (1:2 CPU Clock)

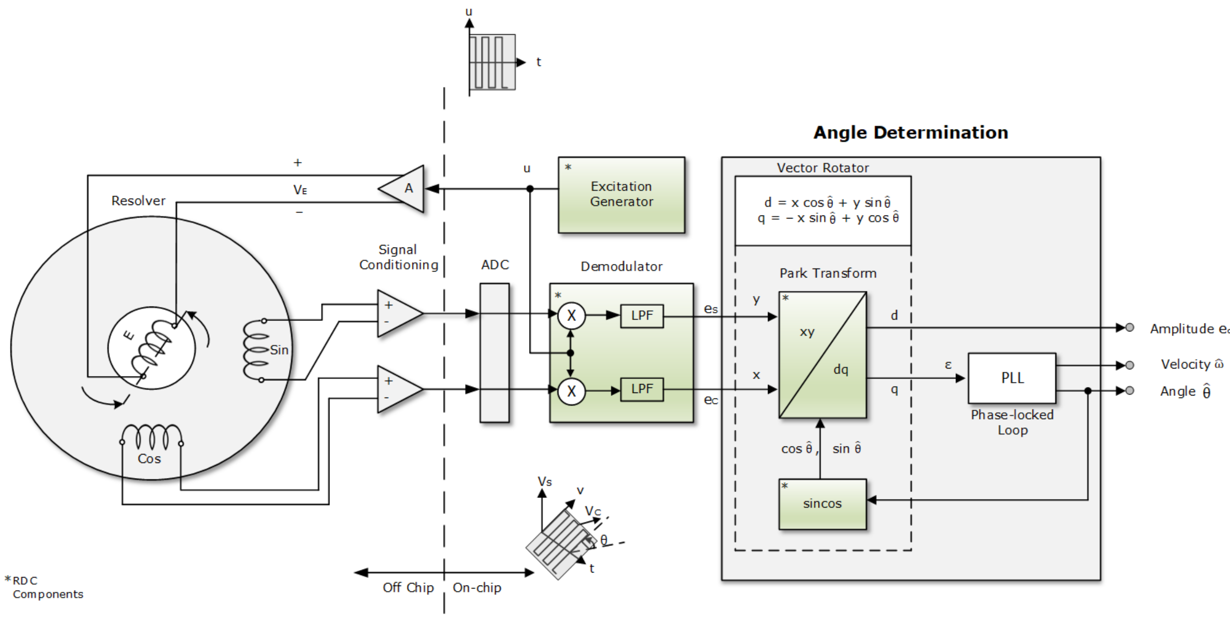
18.2. Architectural Overview

18.2.1. Resolver-to-Digital Converter (RDC)

The primary purpose of the Resolver-to-Digital Converter (RDC) module is to provide excitation input signals and process output signals from an electrical resolver or other position sensors with sinusoidally varying output, such as an LVDT. It can also be used to demodulate other signals with square wave carriers.

Figure 18-1 shows a typical block diagram of the RDC.

Figure 18-1. Typical RDC Block Diagram

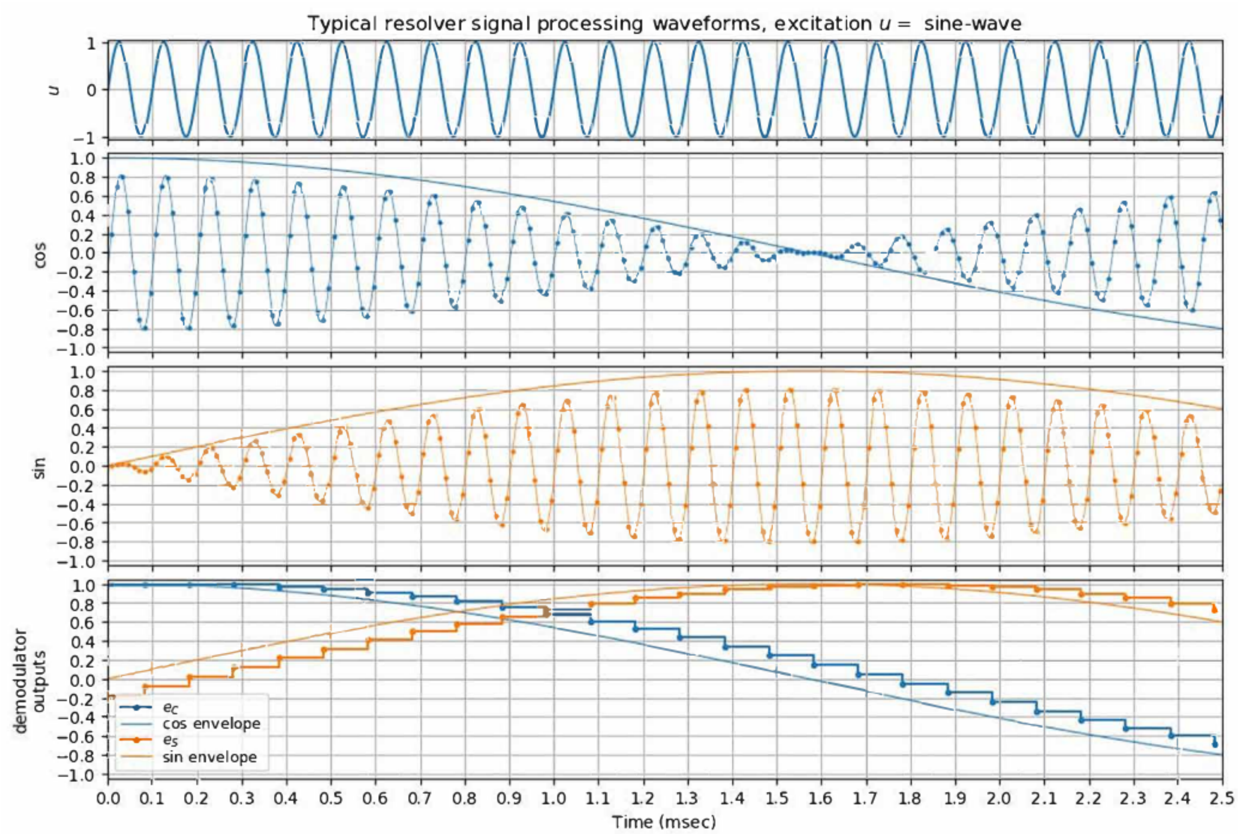


The RDC provides the excitation signal to the resolver and demodulates the sine and cosine outputs to generate a digital representation of angular position and velocity.

A resolver is an electromagnetic transducer that converts mechanical angular position into electrical signals. It consists of a stationary part (stator) and a rotating part (rotor). The stator typically contains one primary winding and two secondary windings. The primary winding is driven by an excitation signal of specified frequency and amplitude that must be resolved. The two secondary windings are positioned 90° out of phase with each other. They resolve the resulting magnetic field into two orthogonal components: “sine” and “cosine.” An example of a resolver as a rotary transformer is shown in Figure 18-4.

Resolvers are typically excited with a sinusoidal signal. The sinusoidal excitation waveform and the demodulated outputs are shown in Figure 18-2.

Figure 18-2. Resolver Signal Processing with Sinusoidal Excitation



Consider the excitation voltage applied to the rotor as follows:

Equation 18-1.

$$V_E = V\sin(\omega\tau)$$

The voltage induced in the sine winding of the stator is given as:

Equation 18-2.

$$V_S = KV_E\sin(\theta) = KV\sin(\omega\tau)\sin(\theta)$$

Where,

K = transformation ratio

θ = rotor's mechanical angle

Similarly, the voltage induced in the cosine winding of the stator is given as:

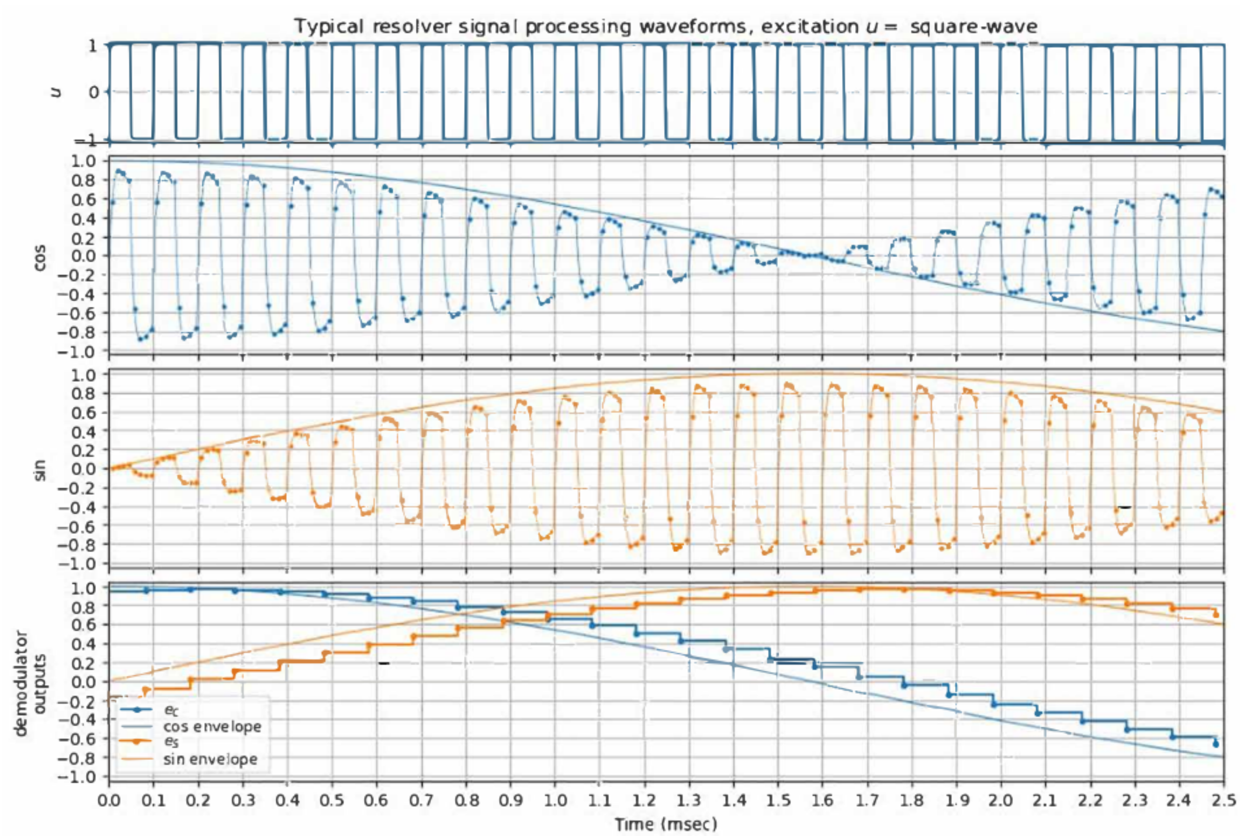
Equation 18-3.

$$V_C = KV_E\cos(\theta) = KV\sin(\omega\tau)\cos(\theta)$$

In the voltage equations of the sine (Equation 18-2) and cosine windings (Equation 18-3), except for the θ, which changes with the shaft, the remaining parameters are fixed.

The dsPIC33AK256MPS306 Resolver-to-Digital converter uses square waves as the excitation signal, in contrast to the conventional approach that employs sinusoidal signals. These square waves drive the stator's primary winding at a fixed frequency and amplitude, producing sine and cosine outputs at the secondary windings. These output signals are then sensed and digitized by the ADC. Demodulation can be performed by multiplying these digitized samples with the excitation signal; for a square wave signal, this can be accomplished by conditionally inverting the resolver outputs when the excitation signal output is at low polarity. This will heterodyne the signals, creating one signal with the excitation frequency components removed, and one signal with components at multiples of the excitation frequency. A low-pass filter can then remove the higher-frequency components, leaving only the sine and cosine signal envelopes. The square wave excitation signal and its typical demodulated output are as shown in Figure 18-3.

Figure 18-3. Resolver Signal Processing with Square Wave Excitation



Once the demodulated sine and cosine values are recovered, a single angle value can be recovered with an arctangent calculation, or a series of values can be recovered with a ratiometric tracking loop.

Using Equation 18-2 and Equation 18-3:

Equation 18-4.

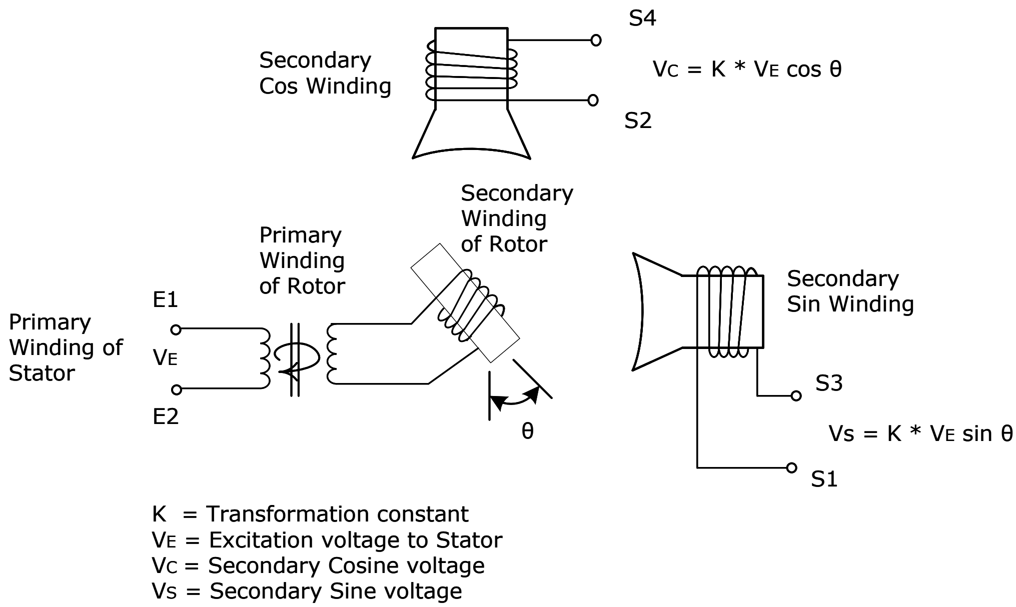
$$e_S / e_C = KV_E \sin(\theta) / KV_E \cos(\theta)$$
$$= \tan(\theta)$$

Equation 18-5.

$$\theta = atan2(e_S, e_S)$$

The RDC provides an optional CORDIC block, which can be used to calculate an error signal for use in the tracking loop.

Figure 18-4. Resolver as Rotary Transformer



18.2.2. Cascaded Integrator Comb (CIC) Filter

A cascaded integrator-comb (CIC) filter is a low-pass filter that can be used for anti-aliasing an analog signal from an ADC by oversampling the input and low-pass filtering to remove high-frequency components. It consists of a series of M accumulators operating at the input frequency, followed by a series of M differencers operating at an output frequency, which is equal to the input frequency divided by a decimation ratio R. The CIC filter of the dsPIC33AK256MPS306 family supports M = 2 and M = 1. Figure 18-5 shows the second-order CIC filter implementation. Figure 18-6 shows the magnitude spectrum of the CIC filter.

Figure 18-5. Typical Second Order Filter Implementation

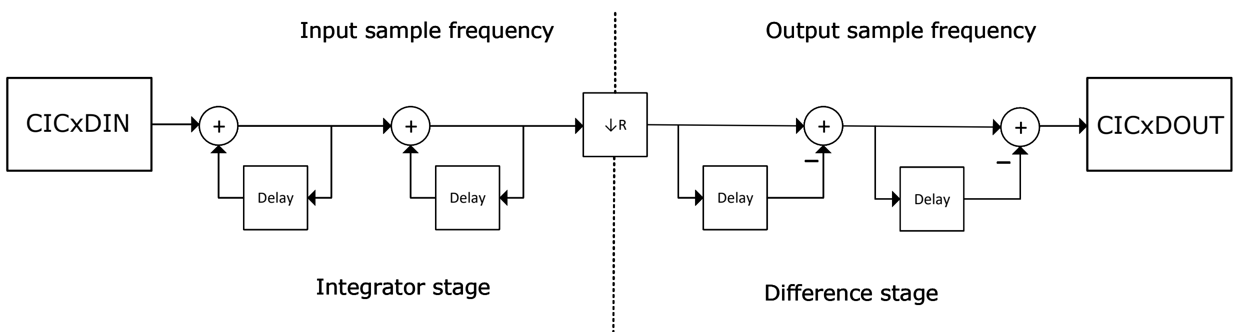
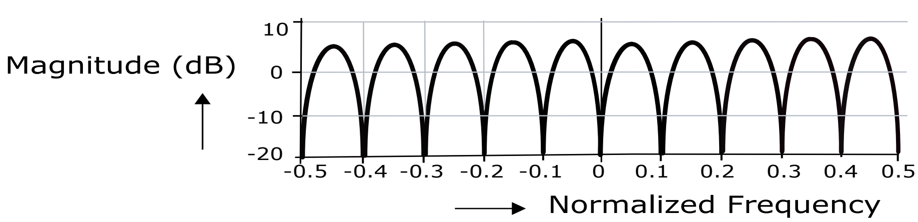


Figure 18-6. Magnitude Spectrum of CIC Filter

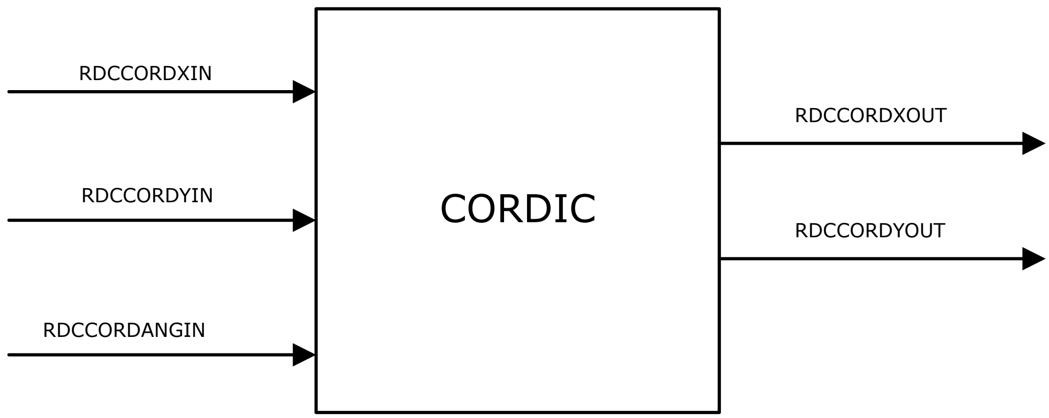


Although the CIC filter in the dsPIC33AK256MPS306 is designed to operate in conjunction with the resolver-to-digital conversion process, it can also be used independently when configured in register operating mode, allowing it to perform integration and decimation of digital samples.

18.2.3. Coordinate Rotation Digital Computer (CORDIC) Block

A CORDIC block employs the CORDIC algorithm to compute error signals. It is an optional block provided by the RDC for use in tracking loops. An overview of the CORDIC block is shown in Figure 18-7.

Figure 18-7. CORDIC Block Overview



18.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1F90	RDCCON	31:24								
		23:16								
		15:8	ON							
		7:0								CORDICSTART
0x1F94	RDCINSEL	31:24								ISIGN1
		23:16	ICHAN1[4:0]					IMUX1[2:0]		
		15:8								ISIGN0
		7:0	ICHAN0[4:0]					IMUX0[2:0]		
0x1F98	RDCSTAT	31:24								
		23:16								
		15:8	SYNCCNTSTAT[7:0]							
		7:0	CORDICDONE	POLFBVAL					IMUXCFGERR	CICINOVF
0x1F9C	RDCEXCCON	31:24								
		23:16	SYNCCNT[7:0]							
		15:8	EXCSYNCEN	EXCOE						
		7:0					EXCFDIV[3:0]			
0x1FA0	RDCEXCDLY	31:24								
		23:16								
		15:8	EXCFBDLY[7:0]							
		7:0							ADCTRGDLY[1:0]	
0x1FA4	RDCCORDXIN	31:24	CXIN[17:10]							
		23:16	CXIN[9:2]							
		15:8	CXIN[1:0]							
		7:0								
0x1FA8	RDCCORDYIN	31:24	CYIN[17:10]							
		23:16	CYIN[9:2]							
		15:8	CYIN[1:0]							
		7:0								
0x1FAC	RDCCORDANGIN	31:24	CANGIN[17:10]							
		23:16	CANGIN[9:2]							
		15:8	CANGIN[1:0]							
		7:0								
0x1FB0	RDCCORDXOUT	31:24	CXOUT[17:10]							
		23:16	CXOUT[9:2]							
		15:8	CXOUT[1:0]							
		7:0								
0x1FB4	RDCCORDYOUT	31:24	CYOUT[17:10]							
		23:16	CYOUT[9:2]							
		15:8	CYOUT[1:0]							
		7:0								
0x1FB8	CICCON1	31:24	INSRC	POLSEL	DEMODEN	CICUPDATE				
		23:16							CH1EN	CH0EN
		15:8	ON		SIDL					ORDER
		7:0			FILTOEN	OUTSHIFT[4:0]				
0x1FBC	CICDECIM	31:24								
		23:16								
		15:8					DECIM[11:8]			
		7:0	DECIM[7:0]							
0x1FC0	CICSTAT	31:24								
		23:16				IISERR	FILTOERR	CICOUTOVF	BUSY	DONE
		15:8					DECIMCNT[11:8]			
		7:0	DECIMCNT[7:0]							
0x1FC4	CICINTHR	31:24	THIGH[15:8]							
		23:16	THIGH[7:0]							
		15:8	TLOW[15:8]							
		7:0	TLOW[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1FC8	CICTIMEOUT	31:24	FILTOCNT[15:0]							
		23:16	FILTOCNT[15:0]							
		15:8	FILTOPR[12:5]							
		7:0	FILTOPR[4:0]							
0x1FCC	CICCON2	31:24								
		23:16								
		15:8								
		7:0								POLOVR
0x1FD0	CICIISxSTAT	31:24					HIISCNT[11:8]			
		23:16	HIISCNT[7:0]							
		15:8					LIISCNT[11:8]			
		7:0	LIISCNT[7:0]							
0x1FD4	CICxDIN	31:24								
		23:16								
		15:8	DIN[15:8]							
		7:0	DIN[7:0]							
0x1FD8	CICxDOUT	31:24	DOUT[31:24]							
		23:16	DOUT[23:16]							
		15:8	DOUT[15:8]							
		7:0	DOUT[7:0]							
0x1FDC	CICxACC	31:24	ACCVAL[31:24]							
		23:16	ACCVAL[23:16]							
		15:8	ACCVAL[15:8]							
		7:0	ACCVAL[7:0]							

18.3.1. RDC Control Register

Name: RDCCON
Offset: 0x001F90

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
								CORDICSTART
Access								R/S/HC
Reset								0

Bit 15 – ON RDC Enable bit

Value	Description
1	Module is enabled.
0	Module is disabled.

Bit 0 – CORDICSTART CORDIC Block Start bit

Value	Description
1	Begins a CORDIC calculation. The CXIN, CYIN and CANGIN bit fields should be loaded with valid inputs before setting this bit. Writes to the CXIN, CYIN or CANGIN bits will be ignored while CORDICSTART is set. This bit is cleared by hardware when the calculation is completed.
0	The CORDIC block calculation is not in progress.

18.3.2. RDC ADC Selection Register

Name: RDCINSEL
Offset: 0x001F94

Notes:
1. Writes to the register when ON (RDCCON[15]) = 1 will be ignored.
2. The table lists example values. Refer to the ADC chapter for details on the number of ADC module instances and the channel count for each instance.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
								ISIGN1
Access								R/W
Reset								0
Bit	23	22	21	20	19	18	17	16
	ICHAN1[4:0]					IMUX1[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
								ISIGN0
Access								R/W
Reset								0
Bit	7	6	5	4	3	2	1	0
	ICHAN0[4:0]					IMUX0[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 24 – ISIGN1 External Input 1 Sign Adjust bit

Value	Description
1	Convert the unsigned representation of the CIC filter input on channel 1 to a signed representation.
0	The selected CIC filter input on channel 1 will not be modified.

Bits 23:19 – ICHAN1[4:0] CIC Filter Input 1 Channel Selector bits(2)
Selects the ADC input signal channel corresponding to the ADC module instance selected by IMUX1. The external input to the CIC filter channel 1 will be sourced through this ADC channel. Set to 0 when IMUX1 selects an input source with no associated channel information.

Value	Description
00001	The source of digital input samples for the CIC filter channel 1 is ADC channel 1 of the ADC module instance selected by IMUX1.
00000	The source of digital input samples for the CIC filter channel 1 is ADC channel 0 of the ADC module instance selected by IMUX1.

Bits 18:16 – IMUX1[2:0] CIC Filter Input Multiplexer 1 bit(2)
Selects the ADC module instance that sources external input signals for the CIC filter channel 1. If the value corresponds to an ADC module instance that is not available with the device, then the IMUXCFGERR(RDCSTAT[1]) bit will be set.

Value	Description
001	ADC module instance 2 sources digital input samples to the CIC filter channel 1.
000	ADC module instance 1 sources digital input samples to CIC filter channel 1.

Bit 8 – ISIGN0 External Input 0 Sign Adjust bit

Value	Description
1	Convert the unsigned representation of the CIC filter input on channel 0 to a signed representation.
0	The selected CIC filter input on channel 0 will not be modified.

Bits 7:3 – ICHAN0[4:0] CIC Filter Input 0 Channel Selector bit[(2)]

Selects the ADC input signal channel corresponding to the ADC module instance selected by IMUX0. The external input to the CIC filter channel 0 will be sourced through this ADC channel. Set to 0 when IMUX0 selects an input source with no associated channel information.

Value	Description
00001	The source of digital input samples for CIC filter channel 0 is ADC channel 1 of the ADC module instance selected by IMUX0.
00000	The source of digital input samples for CIC filter channel 0 is ADC channel 0 of the ADC module instance selected by IMUX0.

Bits 2:0 – IMUX0[2:0] CIC Filter Input Multiplexer 0 bit[(2)]

Selects the ADC module instance that sources external input signals for the CIC filter channel 0. If the value corresponds to an ADC module instance that is not available with the device, then the IMUXCFGERR(RDCSTAT[1]) bit will be set.

Value	Description
001	ADC module instance 2 sources digital input samples to the CIC filter channel 0.
000	ADC module instance 1 sources digital input samples to the CIC filter channel 0.

18.3.3. RDC Status Register

Name: RDCSTAT
Offset: 0x001F98

Notes:
1. The transition of SYNCCNTSTAT may trail the assertion of the RDC sync pulse by 2-3 UPB clock cycles due to synchronization between clock domains.
2. The POLFBVAL bit is valid only if ON(RDCCON[15]) = `1` and EXCSYNCEN(RDCEXCCON[15]) = `1`.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as `'0'`; -n = Value at POR; `'1'` = bit is set; `'0'` = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Access								
Reset								
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	SYNCCNTSTAT[7:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	CORDICDONE	POLFBVAL					IMUXCFGERR	CICINOVF
Access	R/HS/HC	R/HS/HC					R/W/HS	R/W/HS
Reset	1	0					0	0

Bits 15:8 – SYNCCNTSTAT[7:0] Excitation Signal Synchronization Counter Status bits(1)

Value	Description
`xx`	After beginning an excitation signal software synchronization operation by setting the value of SYNCCNT(RDCEXCCON[23:16]) and setting EXCSYNCEN(RDCEXCCON[15]), reads of this bitfield will return the remaining count of cycles until the ADC triggers and excitation signals are generated. If the EXCSYNCEN bit is cleared during the sync period, the value in SYNCCNTSTAT will be kept at the last value that was decremented by the counter.

Bit 7 – CORDICDONE CORDIC Algorithm Done bit

Value	Description
`1`	The CORDIC block input processing is complete, and the RDCCORDXOUT and RDCCORDYOUT registers have been updated with the x and y values.
`0`	The CORDIC algorithm processing has not yet been completed.

Bit 6 – POLFBVAL RDC Feedback Polarity Value bit(2)

Value	Description
`x`	The value of the CIC filter polarity feedback signal for diagnostics. This is equivalent to the polarity of the external signal fed to the CIC filter.

Value	Description
0	Read as 0 if the conditions for operation are not met as per Note 2.

Bit 1 – IMUXCFGERR RDC Configuration Error bit

Value	Description
1	An unimplemented ADC module instance value has been written to IMUX0 (RDCINSEL[2:0]) or IMUX1 (RDCINSEL[18:16]).
0	The values of the IMUX0 (RDCINSEL[2:0]) and IMUX1 (RDCINSEL[18:16]) bits are valid.

Bit 0 – CICINOVF CIC Filter Input Overflow Error bit

Value	Description
1	An input sample valid signal was asserted before a previous CIC filter input was processed.
0	No error condition has occurred.

18.3.4. RDC Excitation Signal Control Register

Name: RDCEXCCON
Offset: 0x001F9C

Notes:
1. Writes to these bits while EXCSYNCEN(RDCEXCCON[15]) = 1 will be ignored.
2. Writes to these bits when ON(RDCCON[15]) = 1 will be ignored.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	SYNCCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXCSYNCEN	EXCOE						
Access	R/W	R/W						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
					EXCFDIV[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 23:16 – SYNCCNT[7:0] Excitation Signal Firmware Synchronization Counter bits(1)
The module will gate off excitation and ADC trigger output signals for the specified number of RDC input clock cycles minus 1 after the EXCSYNCEN bit is set.

Bit 15 – EXCSYNCEN Excitation Signal Firmware Synchronization bit

Value	Description
1	Copy the value of the SYNCCNT bitfield into the SYNCCNTSTAT bitfield and begin decrementing the SYNCCNTSTAT counter. After a number of RDC input clocks specified by SYNCCNT, along with an additional 2-3 clocks for synchronization, the synchronization signal will be asserted for one RDC input clock period. This will be followed by the generation of RDC excitation output and triggering of the ADC. If this bit is set when SYNCCNT equals zero, it will have the same effect as setting it when SYNCCNT = 1.
0	Generation of RDC excitation output and triggering of the ADC will be gated off.

Bit 14 – EXCOE Excitation Signal Output Enable bit
This bit will be used to control the behavior of the RDC excitation output signals only when the excitation signal generation is not active. If set, this bit will enable the outputs via I/O mux output enable and force the outputs of both signals to 0. If cleared, this bit will disable the outputs, resulting in a tri-stated condition. If the user enables the excitation signal output by setting EXCSYNCEN, RDC excitation outputs will take the values of the excitation signals generated by the module once the excitation signal synchronization period elapses, regardless of the EXCOE bit setting.

Value	Description
1	RDC excitation output signals will be driven to 0 and enabled.
0	RDC excitation output signals will not be enabled, and pins are tri-stated.

Bits 3:0 – EXCFDIV[3:0] Excitation Frequency Divider bit(2)

Divides the module input clock signal to generate the excitation clock signal and excitation output signals.

RDC excitation output = (RDC input clock / 8*(EXCFDIV+1))

Value	Description
1111	RDC excitation output = RDC input clock / 128
1110	RDC excitation output = RDC input clock / 120
...	
0001	RDC excitation output = RDC input clock / 16
0000	RDC excitation output = RDC input clock / 8

18.3.5. RDC Excitation Signal Delay Register

Name: RDCEXCDLY
Offset: 0x001FA0

Notes:
1. This register is unused when ADCTRGDLY(RDCEXCCON[15]) = 0.
2. Writes to the register when ON(RDCCON[15]) = 1 will be ignored.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	EXCFBDLY[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
							ADCTRGDLY[1:0]	
Access							R/W	R/W
Reset							0	0

Bits 15:8 – EXCFBDLY[7:0] Excitation Signal Polarity Feedback Delay bits
A delay is inserted between excitation signal generation and the propagation of feedback to the RDC CIC filter. This value must be programmed to compensate for delays introduced by the resolver, external components, and ADC sampling time. The actual delay produced by the EXCFBDLY setting will be between the value of (EXCFBDLY + 1) and (EXCFBDLY + 2), due to the synchronization of the excitation clock with the system clock.

Value	Description
11111111	Delay of 256 system clock cycles
...	
00000010	Delay of 3 system clock cycles
00000001	Delay of 2 system clock cycles
00000000	Delay of 1 system clock cycle

Bits 1:0 – ADCTRGDLY[1:0] ADC Trigger Delay bit
A phase delay is inserted between the RDC excitation clock output signal and the trigger signal to the ADC from the RDC. This value must be selected to ensure that the excitation signal does not transition to a new value during ADC sampling.

Value	Description
11	Delay of 3 RDC input clock cycles
10	Delay of 2 RDC input clock cycles

Value	Description
01	Delay of 1 RDC input clock cycle
00	Delay of 0 RDC input clock cycles

18.3.6. RDC CORDIC Block X Coordinate Input Register

Name: RDCCORDXIN
Offset: 0x001FA4

Note: Writes to the register when CORDICSTART(RDCCON[0]) = 1 will be ignored.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	CXIN[17:10]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CXIN[9:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CXIN[1:0]							
Access	R/W	R/W						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – CXIN[17:10] CORDIC X Coordinate Input Value bits
CORDIC X coordinate input value

Bits 23:16 – CXIN[9:2] CORDIC X Coordinate Input Value bits
CORDIC X coordinate input value

Bits 15:14 – CXIN[1:0] CORDIC X Coordinate Input Value bits
CORDIC X coordinate input value

18.3.7. RDC CORDIC Block Y Coordinate Input Register

Name: RDCCORDYIN
Offset: 0x001FA8

Note: Writes to the register when CORDICSTART(RDCCON[0]) = 1 will be ignored.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	CYIN[17:10]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CYIN[9:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CYIN[1:0]							
Access	R/W	R/W						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – CYIN[17:10] CORDIC Y Coordinate Input Value bits
CORDIC Y coordinate input value

Bits 23:16 – CYIN[9:2] CORDIC Y Coordinate Input Value bits
CORDIC Y coordinate input value

Bits 15:14 – CYIN[1:0] CORDIC Y Coordinate Input Value bits
CORDIC Y coordinate input value

18.3.8. RDC CORDIC Block Angle Input Register

Name: RDCCORDANGIN
Offset: 0x001FAC

Note: Writes to the register when CORDICSTART(RDCCON[0]) = `1` will be ignored.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as `'0'`; -n = Value at POR; `'1'` = bit is set; `'0'` = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	CANGIN[17:10]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CANGIN[9:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CANGIN[1:0]							
Access	R/W	R/W						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – CANGIN[17:10] CORDIC Y Coordinate Input bits
CORDIC angle input value

Bits 23:16 – CANGIN[9:2] CORDIC Y Coordinate Input bits
CORDIC angle input value

Bits 15:14 – CANGIN[1:0] CORDIC Y Coordinate Input bits
CORDIC angle input value

18.3.9. RDC CORDIC Block X Coordinate Output Register

Name: RDCCORDXOUT
Offset: 0x001FB0

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	CXOUT[17:10]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CXOUT[9:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CXOUT[1:0]							
Access	R/W	R/W						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – CXOUT[17:10] CORDIC X Coordinate Output bits
CORDIC X coordinate output value

Bits 23:16 – CXOUT[9:2] CORDIC X Coordinate Output bits
CORDIC X coordinate output value

Bits 15:14 – CXOUT[1:0] CORDIC X Coordinate Output bits
CORDIC X coordinate output value

18.3.10. RDC CORDIC Block Y Coordinate Output Register

Name: RDCCORDYOUT
Offset: 0x001FB4

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	CYOUT[17:10]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CYOUT[9:2]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CYOUT[1:0]							
Access	R/W	R/W						
Reset	0	0						

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 31:24 – CYOUT[17:10] CORDIC Y Coordinate Output bits
CORDIC Y coordinate output value

Bits 23:16 – CYOUT[9:2] CORDIC Y Coordinate Output bits
CORDIC Y coordinate output value

Bits 15:14 – CYOUT[1:0] CORDIC Y Coordinate Output bits
CORDIC Y coordinate output value

18.3.11. CIC Control 1 Register

Name: CICCON1
Offset: 0x001FB8

Notes:
1. Cleared by HW when BUSY is set.
2. Writes to this bit will be ignored when INSRC(CICCON1[31]) = `0` or ON(CICCON1[15]) = `0`.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as `'0'`; -n = Value at POR; `'1'` = bit is set; `'0'` = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	INSRC	POLSEL	DEMODEN	CICUPDATE				
Access	R/W	R/W	R/W	R/S/HC				
Reset	0	0	1	0				

Bit	23	22	21	20	19	18	17	16
							CH1EN	CH0EN
Access							R/W	R/W
Reset							1	1

Bit	15	14	13	12	11	10	9	8
	ON		SIDL					ORDER
Access	R/W		R/W					R/W
Reset	0		0					0

Bit	7	6	5	4	3	2	1	0
			FILTOEN	OUTSHIFT[4:0]				
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 31 – INSRC Filter Input Source Selection bits

Value	Description
1	Filter processes inputs from the module register CICxDIN.
0	Filter processes inputs from an external source.

Bit 30 – POLSEL Filter Polarity Source Selection bit

Value	Description
1	The filter uses the polarity bit (POLOVR(CICCON2[0])).
0	The filter uses the RDC excitation signal for polarity.

Bit 29 – DEMODEN Demodulation Enable bit

Value	Description
1	The filter will demodulate input values based on polarity controlled by POLSEL (CICCON1[30]) and POLOVR (CICCON2[0]).
0	The filter will not demodulate input values.

Bit 28 – CICUPDATE Filter Input Update Command bit(1,2)
Setting this bit triggers the processing of an input sample.

Bit 17 – CH1EN CIC Channel 1 Enable bit

Value	Description
1	Channel 1 is enabled.
0	Channel 1 is disabled. Input signals are ignored.

Bit 16 – CH0EN CIC Channel 0 Enable bit

Value	Description
1	Channel 0 is enabled.
0	Channel 0 is disabled. Input signals are ignored.

Bit 15 – ON CIC Enable bit

Value	Description
1	CIC filter enabled. The filter starts operating with the next input update event or when the CICUPDATE bit is set.
0	CIC filter disabled. Input update events and the CICUPDATE bit toggles are ignored.

Bit 13 – SIDL CIC Stop in Idle bit

Value	Description
1	The operation of the filter stopped when the device was in Idle mode.
0	The operation of the filter continues when the device is in Idle mode.

Bit 8 – ORDER CIC Filter Order bit

Value	Description
1	Filter order is 2 (two integrator stages, two difference stages).
0	Filter order is 1 (one integrator stage, one difference stage).

Bit 5 – FILTOEN CIC Filter Timeout Enable bit

Value	Description
1	The CIC filter input logic will trigger a timeout error if sufficient time elapses between inputs.
0	The CIC filter input logic will not trigger a timeout error.

Bits 4:0 – OUTSHIFT[4:0] Filter Output Shift bit

Filter output will be shifted right by the amount defined by OUTSHIFT when read from the output register. OUTSHIFT = 0 disables shifting. Shifting is done with sign-extension.

18.3.12. CIC Length Register

Name: CICDECIM
Offset: 0x001FBC

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
					DECIM[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DECIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 11:0 – DECIM[11:0] CIC Length bits
DECIM + 1 input samples will be processed before an output sample is generated

18.3.13. CIC Status Register

Name: CICSTAT
Offset: 0x001FC0

Notes:
1. This bit is set by an invalid input event and cleared by the firmware.
2. Writing to this bit has no effect.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				IISERR	FILTOERR	CICOUTOVF	BUSY	DONE
Access				R/W/HS	R/W/HS	R/W/HS	R/HC/HS	R/HC/HS
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
					DECIMCNT[11:8]			
Access					R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DECIMCNT[7:0]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit 20 – IISERR Invalid Input Sample Error bit(1)

Value	Description
1	An invalid input has been detected on an input channel.
0	No invalid input event has occurred since this bit was last cleared.

Bit 19 – FILTOERR CIC Filter Timeout Error bit

Value	Description
1	The value of the filter input counter FILTOCNT(CICTIMEOUT[31:16]) matches or exceeds the period value FILTOPR(CICTIMEOUT[15:3]).
0	The filter input timeout condition has not occurred.

Bit 18 – CICOUTOVF CIC Module Output Overflow bit

Value	Description
1	An output value has been written to the CICxDOUT registers before the previous value was read.
0	No overflow condition has occurred.

Bit 17 – BUSY CIC Module Input Busy bit(2)

Value	Description
1	Module is processing the input sample.

Value	Description
0	Module idle, new input sample can be asserted.

Bit 16 – DONE CIC Output Data Ready bit[(2)]

Value	Description
1	New output data is available.
0	Module is idle or processing data.

Bits 11:0 – DECIMCNT[11:0] DECIM Value from CICDECIM register
The DECIM(CICDECIM[11:0]) value is loaded into these bits when the first sample is processed. It will count down to 0 for each input sample processed.

18.3.14. CIC Invalid Input Sample Detection Control Register

Name: CICINTHR
Offset: 0x001FC4

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	THIGH[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	THIGH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TLOW[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TLOW[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – THIGH[15:0] Invalid Input Sample High Threshold bits
Signed input samples larger than this value are counted.

Bits 15:0 – TLOW[15:0] Invalid Input Sample Low Threshold bits
Signed input samples smaller than this value are counted.

18.3.15. CIC Filter Input Timeout Counter Register

Name: CICTIMEOUT
Offset: 0x001FC8

Note:
1. The FILTOPR bit value may not be modified when the ON (CICCON1[15]) bit is set. This bitfield should not be written when FILTOEN(CICCON1[5]) = `1`.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as `'0'`; -n = Value at POR; `'1'` = bit is set; `'0'` = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	FILTOCNT[15:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FILTOCNT[15:0]							
Access	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC	R/HS/HC
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FILTOPR[12:5]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FILTOPR[4:0]							
Access	R/W	R/W	R/W	R/W	R/W			
Reset	1	0	0	0	0			

Bits 31:16 – FILTOCNT[15:0] CIC Filter Input Timeout Counter bits
This counter is clocked from the system clock. If FILTOEN(CICCON1[5]) = `1` and FILTOPR(CICTIMEOUT[15:3]) is not 0x0000, the filter input timeout counter will be reset and will begin counting when one of the CIC filter input channels receives valid data from the ADC. The counter will stop when valid data input on the other CIC channel is asserted.

Bits 15:8 – FILTOPR[12:5] CIC Filter Input Timeout Period bits(1)
Sets the time period in 8x multiples of the system clock before the filter interrupt timeout is triggered. The counter FILTOCNT will begin counting when the first data is received on any CIC input channel. If the counter value counts FILTOPR cycles before the data input is available on the other enabled channel, the FILTOERR(CICSTAT[19]) bit will be set and an error interrupt will be triggered.

Value	Description
`0x1fff`	An error will be triggered if 65528 (0x1FFF * 8) clock cycles elapse between the valid data input on one of the CIC channels and the valid data input on the other CIC channel.
...	
`0x0002`	An error will be triggered if 16 (0x0002 * 8) clock cycles elapse between the valid data input on one of the CIC channels and the valid data input on the other CIC channel.
`0x0001`	An error will be triggered if 8 (0x0001 * 8) clock cycles elapse between the valid data input on one of the CIC channels and the valid data input on the other CIC channel.
`0x0000`	Timeout is disabled. Setting FILTOPR to zero has the same effect as setting FILTOEN to 0.

Bits 7:3 – FILTOPR[4:0] CIC Filter Input Timeout Period bits[1]

Sets the time period in 8x multiples of the system clock before the filter interrupt timeout is triggered. The counter FILTOCNT will begin counting when the first data is received on any CIC input channel. If the counter value counts FILTOPR cycles before the data input is available on the other enabled channel, the FILTOERR(CICSTAT[19]) bit will be set and an error interrupt will be triggered.

Value	Description
`0x1fff`	An error will be triggered if 65528 (0x1FFF * 8) clock cycles elapse between the valid data input on one of the CIC channels and the valid data input on the other CIC channel.
...	
`0x0002`	An error will be triggered if 16 (0x0002 * 8) clock cycles elapse between the valid data input on one of the CIC channels and the valid data input on the other CIC channel.
`0x0001`	An error will be triggered if 8 (0x0001 * 8) clock cycles elapse between the valid data input on one of the CIC channels and the valid data input on the other CIC channel.
`0x0000`	Timeout is disabled. Setting FILTOPR to zero has the same effect as setting FILTOEN to 0.

18.3.16. CIC Control 2 Register

Name: CICCON2
Offset: 0x001FCC

Notes:

1. The value should only be written if BUSY = 0 (module idle) and will be ignored unless DEMODEN = 1.
2. The POLOVR bit will only be used if POLSEL(CICCON1[30]) = 1.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
								POLOVR
Access								R/W
Reset								1

Bit 0 – POLOVR Input Sample Polarity Manual Configuration bit[1,2]

Value	Description
1	Filter input will be multiplied by 1.
0	Filter input will be multiplied by -1.

18.3.17. CIC Channel x Invalid Input Sample Status Register

Name: CICIISxSTAT
Offset: 0x001FD0

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
					HIISCNT[11:8]			
Access					R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HIISCNT[7:0]							
Access	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
					LIISCNT[11:8]			
Access					R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS
Reset					0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LIISCNT[7:0]							
Access	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS
Reset	0	0	0	0	0	0	0	0

Bits 27:16 – HIISCNT[11:0] Channel x High Invalid Input Sample Count bits
The number of input samples larger than the threshold defined by the THIGH(CICINTHR[31:16]) bits was detected during the calculation of the most recent CIC output value.

Bits 11:0 – LIISCNT[11:0] Channel x Low Invalid Input Sample Count bits
The number of input samples is smaller than the threshold defined by the TLOW(CICINTHR[15:0]) bits detected during the calculation of the most recent CIC output value.

18.3.18. CIC Channel x Input Register

Name: CICxDIN
Offset: 0x001FD4

Note:
1. Writes to this register will be ignored when INSRC(CICCON1[31]) = 0.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	DIN[15:8]							
Access	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DIN[7:0]							
Access	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS	R/W/HC/HS
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – DIN[15:0] CIC Input Register bits
Writing to the DIN register bits sets the input value for channel x.

18.3.19. CIC Channel x Output Register

Name: CICxDOUT
Offset: 0x001FD8

Note:
1. The value of the register will be shifted right by OUTSHIFT(CICCON1[4:0]) when read.

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	DOUT[31:24]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DOUT[23:16]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DOUT[15:8]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DOUT[7:0]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – DOUT[31:0] CIC Output Register bits(1)
CICxDOUT register contains the output values of channel x.

18.3.20. CIC Channel x Decimated Accumulator Value Register

Name: CICxACC
Offset: 0x001FDC

Legend: HC = bit is Cleared by Hardware; HS = bit is Set by Hardware; S = bit can be Set only; R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = bit is set; '0' = bit is cleared; x = bit value is unknown

Bit	31	30	29	28	27	26	25	24
	ACCVAL[31:24]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACCVAL[23:16]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ACCVAL[15:8]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ACCVAL[7:0]							
Access	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS	R/HC/HS
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – ACCVAL[31:0] Channel x Decimated Accumulator Value bits
Accumulator value for filter channel x.

18.4. Operation

The RDC module implements signal generation, demodulation, filtering and tracking mechanisms essential for driving a resolver and processing its output signals. This module can be divided into the following sections:

- Excitation Signal Generation
- Demodulation and CIC Filter
- CORDIC Block

18.4.1. Excitation Signal Generation

Resolvers are typically designed to operate within a specific excitation signal frequency, commonly 5 kHz or 10 kHz, to ensure accurate and reliable position sensing. The RDC module incorporates a dedicated signal generation unit to drive the resolver with a square wave excitation signal and to provide synchronized ADC trigger pulses for capturing the resolver's output. When the ON (RDCCON[15]) bit is set, the RDC module will assert a clock request for the RDC input clock. To ensure proper operation of the resolver, the input clock source should be configured to produce an input clock with a frequency equal to the desired resolver excitation frequency, multiplied by the selected EXCFDIV (RDCEXCCON[3:0]) and the constant scaling factor of four.

The excitation signal generation unit will divide the RDC clock input by a constant scaling factor of four, and further divide it by the oversampling factor defined by the EXCFDIV divider, in order to produce the excitation output signal and its complement, RDCEXC and RDCEXCI. These output signals, along with external amplifiers and buffers, generate the resolver excitation signal. The excitation signal frequency is given by the following equation:

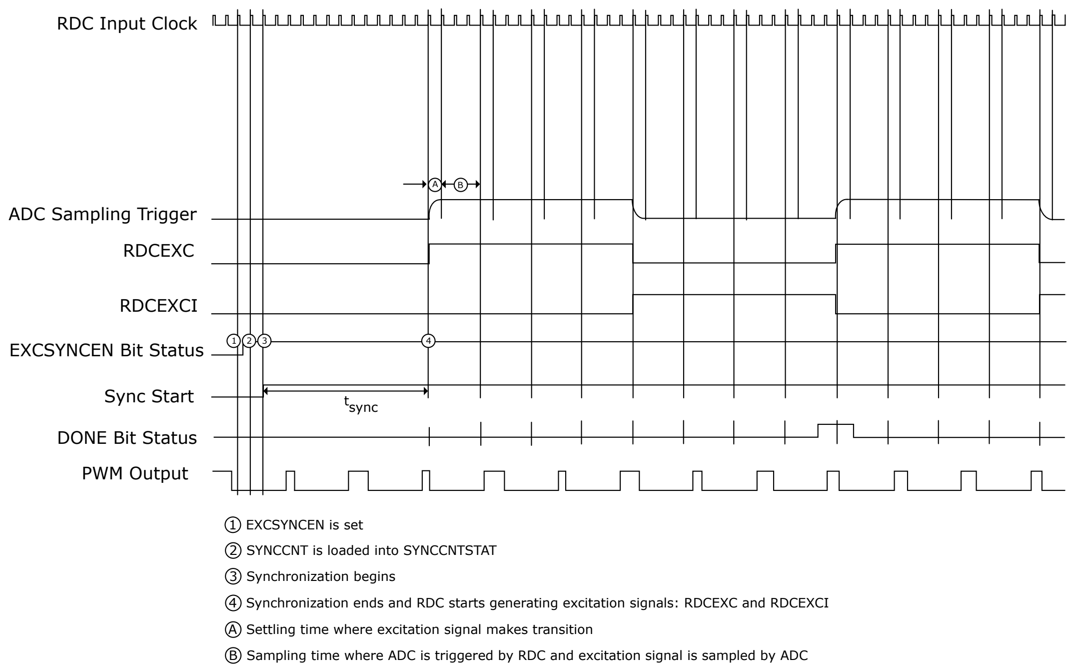
Equation 18-6. Excitation Signal Frequency

$$F_{EXC} = \frac{F_{RDCCLK}}{4 \times (EXCFDIV + 1) \times 2}$$

Where F_{EXC} is the excitation output signal frequency, F_{RDCCLK} is the RDC input signal frequency, and EXCFDIV is the oversampling factor defined in the RDCEXCCON register. This will generate ADC sampling triggers (EXCFDIV+1) times during each half of the excitation square wave.

Figure 18-8 shows the RDC timing diagram.

Figure 18-8. RDC Timing Diagram



When the EXCSYNCEN(RDCEXCCON[15]) bit is set, the RDC will count the number of RDC input clock pulses defined by the SYNCCNT(RDCEXCCON[23:16]) (SYNCCNT - 1 pulses). When the specified counter value elapses, the RDC will start the excitation clock output period and ADC sampling trigger and assert the RDC sync pulse signal. The RDC sync pulse signal will be provided to the CIC filter input and will cause it to reload its decimation count (DECIMCNT(CICSTAT[11:0])) value from the filter length (DECIM(CICDECIM[11:0])). The remaining number of cycles until the counter elapses will be readable from the SYNCCNTSTAT(RDCSTAT[15:8]) bitfield. When configured correctly, this delay should allow the CIC filter to provide its output contemporaneously to a controlling software process.

The excitation signal and inverted excitation signal outputs, RDCEXC and RDCEXCI respectively, will be zero when the excitation signal is gated off. The PPS configured to output this signal will be tri-stated by default. The user may force both signals to zero and enable the output when EXCSYNCEN is clear and before the synchronization delay has elapsed by setting the EXCOE bit.

18.4.2. Demodulation and CIC Filter

The RDCEXC and RDCEXCI outputs from the excitation signal generator are fed to the resolver. The resulting sine and cosine signals from the resolver are then further processed by the CIC filter in coordination with the excitation signal generator and the ADC.

18.4.2.1. ADC Input Selection and Coherent Demodulation

The RDC will automatically trigger the ADC for coherent sampling. The RDCINSEL register allows the selection of the ADC instance and channel used to sample the sine and cosine outputs from the resolver.

The IMUX0 (RDCINSEL[2:0]) and IMUX1 (RDCINSEL[18:16]) bit fields can select the ADC module instances that receive the resolver output to be demodulated. For resolver outputs, the user should configure the ADC to oversample the input signals at a multiple “N” of the excitation frequency; this will be inherently true if using the excitation and trigger frequencies produced by this module due to the availability of the signal dividers using EXCFDIV bits.

If the ADC of the dsPIC33AK256MPS306 family of devices is used, then the resolver output can be connected as input to multiple channels of the ADC. In this case, the ICHAN0 (RDCINSEL[7:3]) and ICHAN1 (RDCINSEL[23:19]) bit fields can select the ADC channels to use for the ADC module instance selected by IMUX0 and IMUX1, respectively. If the IMUX0 and IMUX1 values do not correspond to a valid ADC module instance available in the device, then the IMUXCFGERR (RDCSTAT[1]) bit will be set to `'1'`. CIC filter channel x (x=`0`, `1`) will be fed with digital samples from the corresponding ADC channel selected by ICHANx of the ADC module instance selected by IMUXx. When selecting channels that are sampled sequentially on one input source or which are sampled on different sources with some phase difference, care should be taken to sample the inputs as close to synchronously as possible to minimize output error.

18.4.2.1.1. ADC Trigger Signals

To generate the trigger signal to the ADC, the RDC module will divide the input clock by four and delay it by ADCTRGDLY clocks.

Equation 18-7. ADC Trigger Frequency

$$F_{TRIG} = \frac{F_{RDCCLK}}{4}$$

Where F_{TRIG} is the frequency at which the ADC is triggered by the RDC.

Deriving the excitation signal from the ADC trigger signal creates the possibility that the ADC may sample its inputs during the excitation signal transitions. To prevent this, the ADCTRGDLY (RDCEXCDLY[1:0]) delay bits will add a configurable phase delay between the RDC excitation output and ADC trigger signals. This delay value is selected in clock periods of the RDC input clock and is configurable to allow greater synchronization of outputs with a controlling firmware process.

18.4.2.2. Excitation Signal Feedback Delay

To correctly demodulate the resolver output signals, the excitation signal polarity input should be in phase with the resolver outputs produced by that excitation signal. The EXCFBDLY (RDCEXCDLY[15:8]) bits can introduce a 1-256 UPB cycle clock delay to the excitation signal feedback value to compensate for any delay introduced by the resolver, external circuitry, or ADC sampling time. It may be possible to configure the EXCFBDLY to produce a delay longer than the half-period of the excitation signal, which would prevent correct module operation. It is the user’s responsibility to appropriately configure the delay to avoid this.

18.4.2.3. CIC Filter

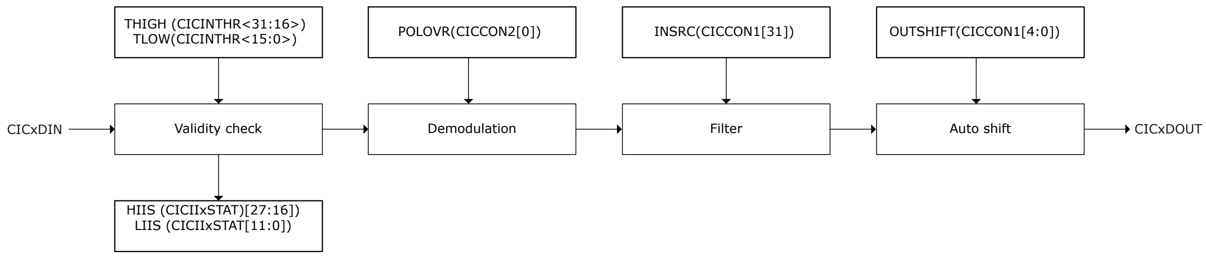
The CIC filter is responsible for demodulation. Its input is provided either through the CICxDIN register or directly from ADC channels, depending on the mode of operation. The filtered output is available in the CICxDOUT registers and can be further processed using a software tracking loop or an arctangent function to determine the resolver angle. For tracking loop implementations, an error signal is required as input; this signal can optionally be computed in hardware using the CORDIC block.

The CIC filter operates in 4 stages as shown in Figure 18-9:

1. Input sample validity check

2. Heterodyne (synchronous demodulation)
3. Filter
4. Auto-shift

Figure 18-9. CIC Channel Block Diagram



18.4.2.3.1. Input Sample Validity Checking

The CIC filter checks whether input samples fall within a valid range, which can be configured by the user using the THIGH (CICINTHR[31:16]) and TLOW (CICINTHR[15:0]) thresholds. If the current input sample is below the threshold defined by TLOW, the module will increase the LIISCNT (CICIISxSTAT[11:0]) count. If the input sample is above the threshold defined by THIGH, the module will increase the HIISCNT (CICIISxSTAT[27:16]) count. Regardless of whether the input sample is valid or invalid, it will still be processed by the subsequent stages of the CIC filter.

TLOW and THIGH are the values stored in the CICINTHR register. The counted value of invalid samples will be updated in the CICIISxSTAT register (HIISCNT and LIISCNT bits) when a CIC output sample is generated. If an invalid input is detected on any channel, the IISERR (CICSTAT[20]) bit will be set, and an interrupt will be asserted.

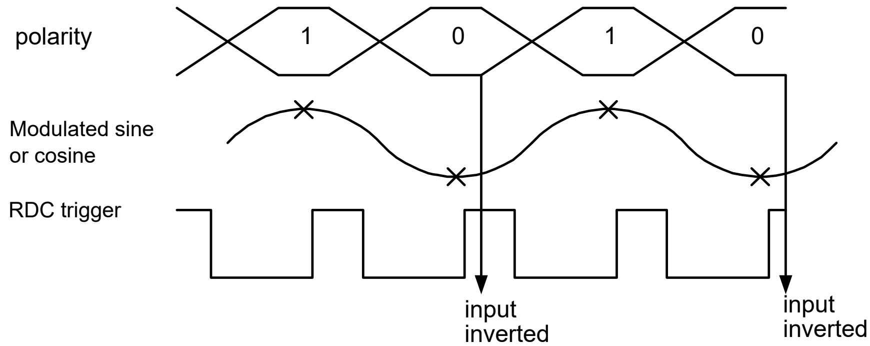
18.4.2.3.2. Heterodyne (Synchronous Demodulation)

The CIC filter can also be used to digitally demodulate the incoming signal. This can be activated or deactivated using the DEMODEN (CICCON1[29]) bit. Digital demodulation is useful if the required signal is modulated onto a carrier signal (amplitude modulation). The demodulation is performed by multiplying the input data with either ‘1’ or ‘-1’, followed by the low-pass filtering provided by the CIC filter to recover the carrier amplitude.

The filter can perform automatic demodulation when it is operated in external signal source mode based on the excitation signal value when POLSEL(CICCON1[30]) = 1. In this mode, the polarity of the external signal is obtained when the input data sample is acquired. When the polarity input is 1, the demodulator passes the input sample without changes (multiplies by ‘1’). When the polarity input is 0, the demodulator multiplies the input sample by -1. Figure 18-10 illustrates the timing diagram of the CIC filter's demodulation in external signal source mode, where the signal is sampled by the ADC at its peak.

When POLSEL = 0, the demodulation will be based on the value of the POLOVR (CICCON2[0]) bit. In this mode, when POLOVR is set to ‘1’, the module multiplies the input sample by ‘1’, and if POLOVR is set to ‘0’, the module multiplies the input sample by ‘-1’.

Figure 18-10. Input Signal Demodulation



18.4.2.3.3. Filtering

The CIC filter supports two operating orders: filter order 1 and filter order 2, which define the number of integrator and differentiator stages in the filter. The desired order is configured using the ORDER bit (CICCON1[8]). When operating in filter order 2, the filter requires a settling time of two output samples. Therefore, the first output sample should be discarded to ensure valid data.

The CIC filter of the dsPIC33AK256MPS306 family of devices has two channels. The filter can process both channels at the same time, only if they are enabled (CH0EN (CICCON1[16]) and CH0EN (CICCON1[17]) bits are set).

Operating Modes

The CIC filter operates in two modes based on the INSRC(CICCON1[31]) bit:

- External signal source mode: The CIC filter takes input values from external digital signals (from ADC).
- Register source mode: The CIC filter takes input from CIC input registers (CICxDIN), written by user software.

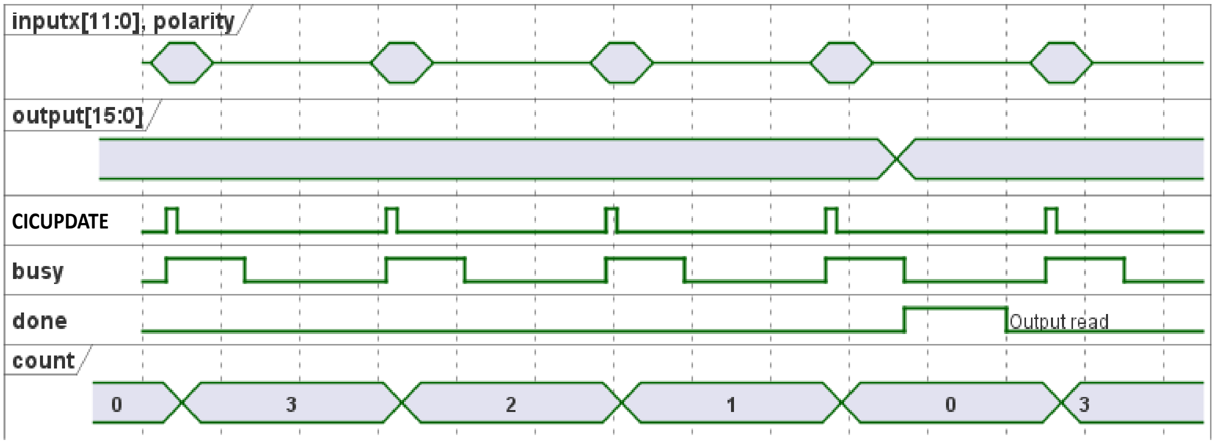
Register Source Mode

When the CIC filter is configured for register source mode (INSRC(CICCON1[31])), the filter's input is provided by the user writing data to the CICxDIN register. The input of all enabled channels will be processed when the CICUPDATE(CICCON1[28]) bit is set, regardless of whether the input registers have been written to or not. Writing to the input registers will not trigger any action. While the input sample is being processed, the BUSY bit is set, and CICUPDATE will be cleared by the filter. When processing is done, the BUSY bit is cleared, and the input process complete interrupt is asserted. Upon completion of each input sample, the DECIMCNT(CICSTAT[11:0]) bit will be decremented. It will count down to zero for each input sample processed.

Once the DECIMCNT becomes zero, the processing by the filter is completed, the DONE bit is set, and the output sample ready interrupt is asserted. The DONE (CICSTAT[16]) bit is cleared when the output of all enabled channels is read. If a new output value becomes available before the previous outputs have been read, the overflow interrupt will be triggered, and the CICOUTOVF (CICSTAT[18]) bit will be set. The CICOUTOVF bit can be cleared by software. The output data of channel x is available at CICxDOUT.

The 32-bit CICxACC register acts as the accumulator for the CIC filter, maintaining a cumulative sum of all previous input samples. It does not reset when the DECIMCNT bit becomes zero following output sample generation. CICxACC will be updated with the cumulative sum only when DECIMCNT becomes zero.

Figure 18-11. Register Source Mode



In summary, here are the steps to execute the CIC filter in register source mode:

1. Write the INSRC(CICCON1[31]) bit to `'1'`.
2. Load DECIM(CICDECIM[11:0]) bits with filter length minus one. For example, if the filter length is four, then load DECIM with three.
3. Set the filter order in ORDER(CICCON1[8]).
4. If demodulation is required, set the DEMODEN(CICON[29]) bit. If POLSEL(CICCON1[30]) = `0`, the demodulation will be based on the value of the POLOVR(CICCON2[0]) bit.
5. Set the channel enable bits, CH0EN and CH1EN. Turn on the CIC filter, ON(CICCON1[15]).
6. Verify that the BUSY bit is cleared before writing values to the input register.
7. Load CICxDIN registers with input samples.
8. Set the CICUPDATE bit to initiate input sample processing. Upon completion, the CICIPDIF interrupt flag will be set, triggering an interrupt event. The DECIMCNT bit will be loaded with DECIM bits on the first input sample and decrements on subsequent samples.
9. Once the DECIMCNT bit becomes zero, the CIC filter updates CICxACC with the total sum of input samples. Following this, the DONE bit is set, and the CICOSRIF interrupt flag triggers an interrupt event.
10. The DONE bit will be cleared once the CICxDOUT register is read. The user can load the input register with new input samples.
11. Thus, out of four input samples, one output sample is obtained. The input samples have been integrated and decimated.

External Signal Source Mode

When the CIC filter is configured for external signal source mode (INSRC(CICCON1[31]) = `0`), the filter's input is provided internally by the ADC. The RDCINSEL register is configured as described in ADC Input Selection and Coherent Demodulation, to select the appropriate ADC instance and its corresponding channel for sampling the resolver signals. Refer to the ADC chapter for its configuration details and for selecting the RDC module as the trigger source for conversions.

In external signal source mode, any writes to the CICxDIN register and the CICUPDATE bit are ignored.

In summary, here are the steps to execute the CIC filter in external signal source mode:

1. Write INSRC(CICCON1[31]) bit to `'0'`.

2. Load DECIM(CICDECIM[11:0]) bits with filter length-1. For example, if the filter length is four, then load DECIM with three.
3. Set the filter order in ORDER(CICCON1[8]).
4. If demodulation is required, set DEMODEN(CICON[29]) bit. If POLSEL(CICCON1[30]) = `1`, the demodulation will be based on the polarity of the excitation signal.
5. Set the channel enable bits, CH0EN and CH1EN. Turn on the CIC filter, ON(CICCON1[15]).
6. Configure the RDCINSEL register by setting the IMUXx and ICHANx bits according to the specific ADC module instance and its channel that provides input samples to the CIC channels.
7. Configure the ADC at the required sampling rate with RDC as its trigger source.
8. Configure RDCEXCCON, RDCEXCDLY, and RDCCON registers to generate the excitation signal.
9. The ADC will feed the digital samples to CIC channels.
10. The CIC filter expects a signed input. When operating in external source mode, if it is connected to an ADC only capable of providing unsigned outputs, the ISIGN0(RDCINSEL[8]) and ISIGN1(RDCINSEL[24]) bits may be used to change the output representation of the inputs. Setting the ISIGNx bit will convert the input samples to signed format by inverting the upper bit of the input sample on channel x. The ADC of the dsPIC33AK256MPS306 family of devices is capable of producing both unsigned and signed samples.
11. Once the DECIMCNT bit becomes zero, the CIC filter updates CICxACC with the total sum of input samples. Following this, the DONE bit is set, and the CICOSRIF interrupt flag triggers an interrupt event.
12. The DONE bit will get cleared once the CICxDOUT register is read.

In external source mode, the CIC filter requires input samples on both channels for processing. Users can configure a timeout between these filter samples using the CIC timer, which is enabled when FILTOEN(CICCON1[5]) is set and FILTOPR(CICTIMEOUT[15:3]) is non-zero. The counter FILTOCNT(CICTIMEOUT[31:16]) will begin counting when the first data is received from the ADC on any CIC input channel. If the counter value counts FILTOPR cycles before the data input is available on the other enabled channel, the FILTOERR(CICSTAT[19]) bit will be set, and an error interrupt will be triggered. The counter will stop when valid data input on the other CIC channel is asserted by the ADC.

If one of the CIC channels receives a new ADC input sample before the other channel receives its corresponding sample, then the CICINOVF(RDCSTAT[0]) error bit will be set to indicate that an input overflow has occurred, which also triggers an RDCERRIF interrupt. This signifies a misconfiguration of the ADC channels, and it's the user's responsibility to ensure proper configuration.

18.4.2.3.4. Auto Shift of Filter Output

The processing capacity of the filter is limited to 32 bits. Depending on the size of the input values, this limit can be exceeded and will cause incorrect filter operation. It is the user’s responsibility to ensure that the size of input values, filter length (decimation ratio) and filter order do not exceed 32 bits, complying with the constraint given by Equation 18-8.

Equation 18-8. Maximum Sizes

$$Input\_bits + length\_bits \times order \leq 32$$

Where *Input_bits* is the number of bits that the input signal actually uses, *length_bits* is the number of bits required by the filter length, and *order* is the actual filter order (1 or 2). For example, an input having a maximum of 12 bits can be filtered with a length of 1024 (10 bits) and order 2. However, using a 16-bit input value and order 2 allows only a maximum length of 256 (8 bits). It is the user’s responsibility to ensure that this is fulfilled.

To ease further processing and transfer of values, the output can be shifted right by using the OUTSHIFT(CICCON1[4:0]) bits. When the CICxDOUT values are read from the register, they will be shifted right by the value set by OUTSHIFT. Shifting can be disabled by setting the OUTSHIFT value to zero.

18.4.3. CORDIC Block

The CORDIC block implements the CORDIC algorithm's rotation mode. It accepts three inputs: x and y coordinates describing the feedback angle, φ, in input CORDIC registers RDCCORDXIN, RDCCORDYIN and RDCCORDANGIN respectively. The CORDIC block performs a vector rotation on the input x and y coordinates. The resulting rotated x and y coordinates are then output to the RDCCORDXOUT and RDCCORDYOUT registers. An example of a software tracking loop is provided in Example 1 of the Application Example.

While the CXIN(RDCCORDXIN[17:0]), CYIN(RDCCORDYIN[17:0]), and CANGIN(RDCCORDANGIN[17:0]) input bits are generally readable and writable, writes are blocked when CORDICSTART is set. When CORDICSTART(RDCCON[0]) = `1`, the module will begin executing the CORDIC algorithm, and the CORDICDONE(RDCSTAT[7]) bit will be cleared. When the execution of the algorithm is completed, the CORDICDONE bit will be reset to `1`, and the RDCCORDXOUT and RDCCORDYOUT registers will be updated with the x and y coordinates. When the CORDICDONE bit is set or the module is disabled by clearing ON(RDCCON[15]), the CORDICSTART bit will be cleared. See Example 2 in the Application Example for a demonstration of x, y, and feedback angle (φ) inputs and their resulting outputs.

18.5. Interrupts

There are five interrupts in the RDC/CIC module:

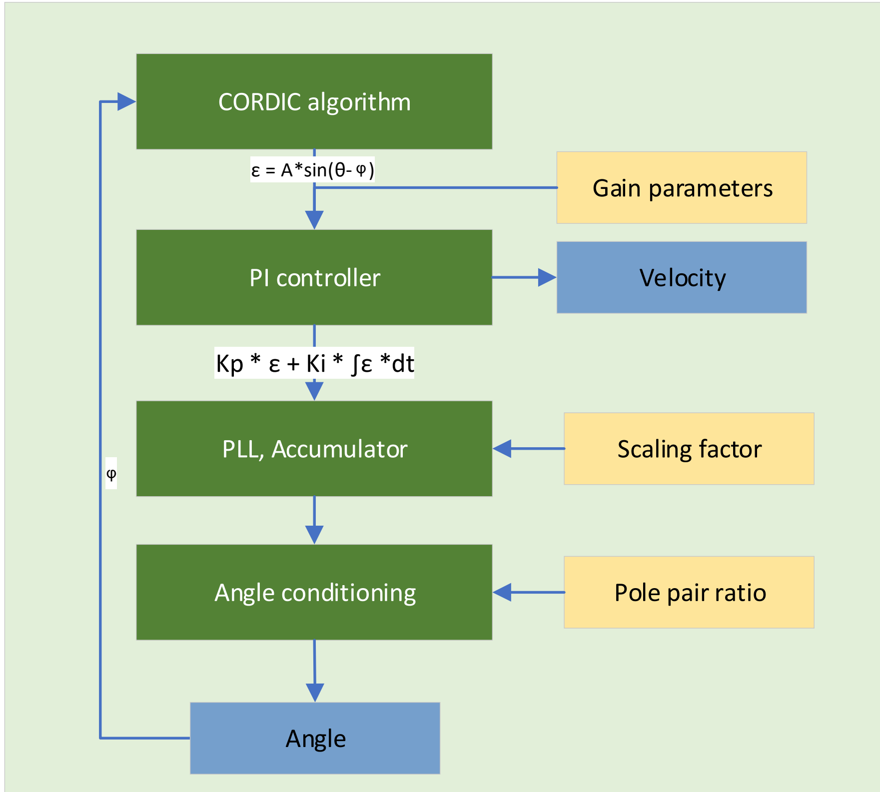
- CICIPDIF interrupt is triggered when input sample processing is done.
- CICOSRIF interrupt is triggered when the output sample is ready.
- CICERRIF interrupt is triggered when one of the following error conditions occurs:
  – A new sample is written into CICxDOUT before the previous sample has been read (CICOUTOVF(CICSTAT[18]) = `1`).
  – A CIC timeout error has occurred (FILTOERR(CICSTAT[19]) = `1`).
  – An invalid input error has occurred (IISERR(CICSTAT[20]) = `1`).
- RDCCORIF interrupt is triggered when CORDIC block operation is completed.
- RDCERRIF interrupt is triggered when a CIC input channel receives a new sample from its ADC channel before the other CIC input channel has received its prior sample.

18.6. Application Example

18.6.1. Example 1

The CORDIC algorithm accepts three inputs: an x and y coordinate describing the resolver angle, θ, and a feedback input angle φ. The x and y inputs may be provided by the demodulated resolver output values at the rotor angle, sin(θ) and cos(θ), and φ may be calculated by the software tracking loop as shown in Figure 18-12.

Figure 18-12. Example Tracking Loop



For the resolver application, the CORDIC algorithm will rotate the vector defined by the CIC filter's outputs, which are proportional to the sine and cosine of the mechanical resolver angle, by the negative value of the previously estimated resolver angle. As the estimated value of the angle approaches the actual measured value of the angle from the resolver, the rotated vector's y-coordinate will approach zero. This y-coordinate output can be used as the error signal input ε, defined in Equation 18-9, to the PI controller to produce the next, more accurate estimate of the angle and angular velocity.

Equation 18-9. CORDIC Error Calculation

$$\epsilon = A\left(\sin\theta\cos\varphi - \cos\theta\sin\varphi\right) = A\sin\left(\theta - \varphi\right) = A\sin\tilde{\theta}$$

The CORDIC algorithm will add a signal gain, Ac, described by Equation 18-10. The combination of this gain with the gain inherent in the selected system components will be referred to as the gain A. Because of this gain, the user must select inputs x and y such that they will not cause an overflow of the CORDIC rotation calculations. Depending on circuit characteristics, this may require a 1-bit right shift of the CIC filter outputs before writing them to the CORDIC x and y input registers.

Equation 18-10. CORDIC Gain

$$A_C = \lim_{n \to \infty} \prod_{i=0}^{n-1} \sqrt{1 + 2^{-2i}} \approx 1.64676$$

18.6.2. Example 2

The angle output of CORDIC is represented with 18-bit resolution, where each bit contributes to the total 360° range. The contribution of each bit at position κ is given by:

Equation 18-11.

$$\frac{180°}{2^{(17-\kappa)}}$$

where κ = 17 to 0.

Table 18-3. Contribution of Each Bit to Angle Representation

Bit Position	Angle
17	180°
16	90°
15	45°
14	22.5°
13	11.25°
12	5.625°
11	2.812°
10	1.406°
9	0.703°
8	0.351°
7	0.175°
6	0.087°
5	0.043°
4	0.021°
3	0.010°
2	0.005°
1	0.002°
0	0.001°

The CORDIC block implements the rotation mode of the CORDIC algorithm. The algorithm begins by rotating the vector (x,y) = (1 , 0), which represents 45°. The inputs are fed to input CORDIC registers after multiplying the coordinates by the CORDIC gain, 0.6072. Hence, the coordinate inputs are RDCCORDXIN = 0x4DB88000 = 0.6072 and RDCCORDYIN = 0. Suppose the resolver angle is 225°, then using Table 18-3, RDCCORDANGIN = 0b101000000000000000 = 225°. Once the algorithm finishes, the resulting output will be available in CORDIC output registers. RDCCORDXOUT = 0xA5818000, RDCCORDYOUT = 0xA57E8000.

18.6.3. Example 3

Below is the code example to configure RDC to generate an excitation signal. The CIC filter is configured for external signal source mode.

Example 18-1. RDC Configuration Code

```
int main()
{
    /*Excitation output remap goes here*/

    /*ADC configuration goes here*/

    /*CIC filter configuration */
    _CICOSRIE = 1;                // Enable output sample read interrupt
    CICCON1bits.CH0EN = 1;        // Enable CIC filter channel 0
    CICCON1bits.CH1EN = 1;        // Enable CIC filter channel 1
    CICCON1bits.INSRC = 0;       /* CIC filter channels are fed from sampled values of
                                    excitation signal*/
    CICCON1bits.ORDER = 0;        // CIC filter in order 1
    CICDECIMbits.DECIM = 3;       // filter length is 4
    CICCON1bits.POLSEL = 0;       // CIC uses RDC excitation signal for polarity
    CICCON1bits.DEMODEN = 1;      // Demodulation is enabled
    CICCON1bits.ON = 1;           // CIC filter is enabled

    /*RDC configuration to generate excitation signal*/
    RDCINSELbits.IMUX0 = 0;      // CIC filter channel 0 is sourced with digital samples by ADC
                                  // module instance 1
    RDCINSELbits.ICHAN0 = 0;      // ADC1 channel 0 sources CIC filter channel 0
    RDCINSEL.IMUX1 = 1;          /* CIC filter channel 1 is sourced with digital samples by
                                    ADC module instance 2*/
    RDCINSELbits.ICHAN1 = 1;      // ADC2 channel 1 sources CIC filter channel 1
    RDCEXCCONbits.EXCFDIV = 1;    // (RDC input clock/16)
    RDCEXCDLYbits.EXCFBDLY = 10;  // Feedback delay of 10 system clock cycles
    RDCEXCDLYbits.ADCTRGDLY = 1;   /* 1 RDC clock phase delay between RDC excitation clock
                                    output signal and the trigger signal to ADC*/
    RDCCONbits.ON = 1;            // Enable RDC block
    RDCEXCCONbits.SYNCCNT = 255;  /* Sync pulse will be generated after counting 255 RDC
                                    clocks*/
    RDCEXCCONbits.EXCSYNCEN  = 1; // Enable synchronization
    while(1);
}
void __attribute__((interrupt, no_auto_psv)) _CICOUTInterrupt(void)
{
    _CICOSRIF = 0;                // Clear output sample read interrupt flag
    /*Read CIC0OUT and CIC1OUT register here*/

}
```

18.7. Power-Saving Modes

This section discusses the operation of the RDC in sleep and idle modes.

18.7.1. Operation in Sleep Mode

When the device enters Sleep mode, the clock available to the RDC is disabled. The RDC measures the position of a physical element in real time, using integrated input samples and feedback from previous estimations. If this process is interrupted by entering a low-power mode, the RDC will not provide valid results. While the module recovers upon exiting sleep mode, it is the user's responsibility to clear any invalid outputs and reinitialize the RDC. Although the excitation signal generator remains active, it will not be able to generate a sync pulse since the sync pulse needs to be synchronized with the RDC clock.

18.7.2. Operation in Idle Mode

When the device enters Idle mode, the CPU is no longer clocked. In Idle mode, the operation of the demodulation and CIC filter can be controlled by the SIDL (CICCON1[13]) bit; however, other module units remain active.

19. High-Speed Analog Comparator with Slope Compensation DAC

The High-Speed Analog Comparator with the Slope Compensation Digital to Analog Converter (DAC) provides a method to monitor voltage, current and other critical signals in a power conversion application that may be too fast for the CPU and ADC to capture. The analog comparator module can be used to implement Peak Current mode control, Critical Conduction mode (variable frequency) and Hysteretic Control mode.

The High-Speed Analog Comparator with the Slope Compensation DAC consists of the following key features:

- Rail-to-Rail Analog Comparator
- Programmable Comparator Hysteresis
- Programmable Output Polarity
- Interrupt Generation Capability
- Dedicated 12-bit Resolution Pulse Density Modulation (PDM) Digital-to-Analog Converter (DAC) for each Analog Comparator
- Multimode Multipole RC Output Filter:
  - Transition mode: provides the fastest response
  - Fast mode: for tracking DAC slopes
  - Steady-State mode: provides 12-bit resolution
- Dedicated Support for the Following Modes:
  - Slope Generation
  - Hysteretic Control
  - Triangular Wave
- Functional Support for the High-Resolution PWM Module, which Includes:
  - PWM duty cycle control
  - PWM period control
  - PWM Fault detect

19.1. Device-Specific Information

Table 19-1. DAC Summary

DAC Module Instances	Inputs per Instance	DAC Outputs	Clock Source	Peripheral Bus Speed
5	5 External, 1 Internal	1	CLKGEN7	Standard (1:2 CPU Clock)

Table 19-2. High-Speed Analog Comparator Module Availability

Comparator Input	36-Pin	48-Pin	64-Pin	PPS
ALLCMPNC		x	x	No
ALLCMPND		x	x	No
ALLCMPNE		—	x	No
ALLCMPNF		—	x	No
ALLCMPP	x	x	x	No
CMPP1A	x	x	x	No
CMPP1B	x	x	x	No
CMPP1C	x	x	x	No
CMPP1D		—	—	No

Table 19-2. High-Speed Analog Comparator Module Availability (continued)

Comparator Input	36-Pin	48-Pin	64-Pin	PPS
CMPP2A	x	x	x	No
CMPP2B	x	x	x	No
CMPP2C	x	x	x	No
CMPP2D		x	x	No
CMPP3A		x	x	No
CMPP3B		x	x	No
CMPP3C		x	x	No
CMPP3D		—	—	No
CMPP4A	x	x	x	No
CMPP4B	x	x	x	No
CMPP4C		x	x	No
CMPP4D	x	x	x	No
CMPP5A		x	x	No
CMPP5B		x	x	No
CMPP5C		—	x	No
CMPP5D		—	—	No

Table 19-3. Slope Start Signal Selection (SLPSTRT)

Slope Start Signal Selection	Source
7	RPV15
6	RPV14
5	RPV13
4	PWM4
3	PWM3
2	PWM2
1	PWM1
0	N/A

Table 19-4. Slope Stop A Signal Select bits (SLPSTOPA)

Slope Stop A Signal Selection	Source
7	RPV15
6	RPV14
5	RPV13
4	PWM4
3	PWM3
2	PWM2
1	PWM1
0	N/A

Table 19-5. Slope Stop B Signal Select bits (SLPSTOPB)

Slope Stop B Signal Selection	Source
7	RPV15
6	RPV14
5	RPV13
4	PWM4
3	PWM3

Table 19-5. Slope Stop B Signal Select bits (SLPSTOPB) (continued)

Slope Stop B Signal Selection	Source
2	PWM2
1	PWM1
0	N/A

Table 19-6. Hysteretic Comparator Function Input Select Bits (HCFSEL)

Hysterectic Comparator Function Input Selection	Description
7	RPV15
6	RPV14
5	RPV13
4	PWM4H
3	PWM3H
2	PWM2H
1	PWM1H
0	N/A

The calibration register FPDMDAC is located in Flash at 0x7F20E0 with the POSINLADJ, NEGINLADJ and DNLADJ bit fields. The location should be copied and written to the corresponding bit fields in the DACCTRL1 SFR at start up.

Table 19-7. FPDMDAC Calibration Register

Name	Address Offset	Bit Field	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
FPDMDAC	0x000	31:24	—	—	—	—	cfg_dac_filter[3:0]			
		23:16	—	—	POSINLADJ[5:0]					
		15:8	—	NEGINLADJ[6:0]						
		7:0	—	—	—	DNLADJ[4:0]				

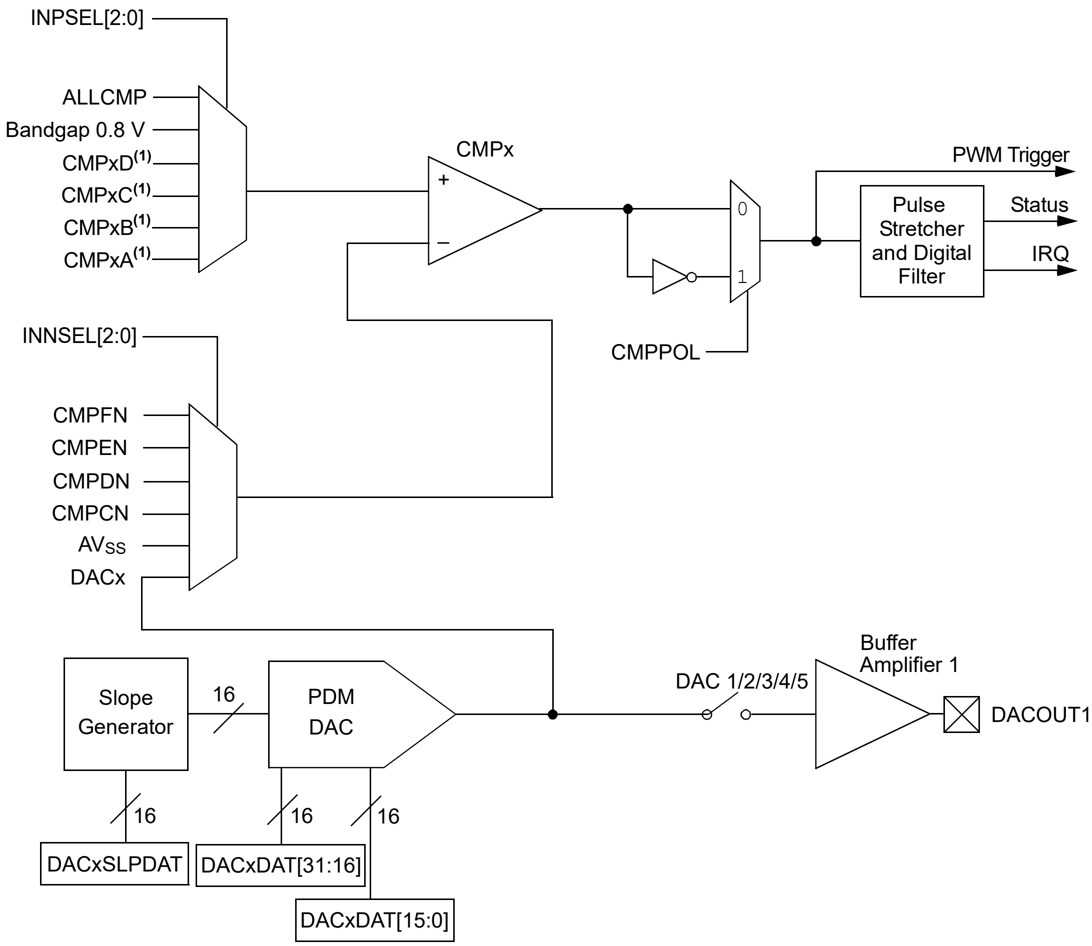
19.2. Architectural Overview

The high-speed analog comparator module is comprised of a high-speed comparator, Pulse Density Modulation (PDM) DAC and a slope compensation unit. The slope compensation unit provides a user-defined slope which can be used to alter the DAC output. This feature is useful in applications, such as Peak Current mode control, where slope compensation is required to maintain the stability of the power supply. The user simply specifies the direction and rate of change for the slope compensation, and the output of the DAC is modified accordingly.

The DAC consists of a PDM unit followed by a digitally-controlled multiphase RC filter. The PDM unit uses a phase accumulator circuit to generate an output stream of pulses. The density of the pulse stream is proportional to the input data value, relative to the maximum value supported by the bit width of the accumulator. The output pulse density is representative of the desired output voltage. The pulse stream is filtered with an RC filter to yield an analog voltage. The output of the DAC is connected to the negative input of the comparator. The positive input of the comparator can be selected using an MUX from the input pins. The comparator provides a high-speed operation with a typical delay of 5 ns.

The output of the comparator can be processed by the pulse stretcher and the digital filter blocks, which prevent a comparator response to unintended fast transients in the inputs. Figure 19-1 shows a block diagram of the high-speed analog comparator module. The DAC module can be operated in one of four modes: Slope Generation, Triangular Wave, Hysteretic or as a normal 12-bit DAC. Each of these modes can be used in a variety of power supply applications.

Figure 19-1. High-Speed Analog Comparator Module Block Diagram



Note:

1. Refer to a specific device pinout for available inputs.

19.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1D40	DACCTRL1	31:24	RREN		POSINLADJ[5:0]					
		23:16		NEGINLADJ[6:0]						
		15:8	ON		SIDL	DNLADJ[4:0]				
		7:0						FCLKDIV[2:0]		
0x1D44	DACCTRL2	31:24							SSTIME[9:8]	
		23:16	SSTIME[7:0]							
		15:8							TMODTIME[9:8]	
		7:0	TMODTIME[7:0]							
0x1D48	DAC1CON	31:24							TMCB[9:8]	
		23:16	TMCB[7:0]							
		15:8	DACEN		IRQM[1:0]			UPDTMDIS		DACOEN
		7:0					UPDTRG[1:0]		UPDATE	UPDREQ
0x1D4C	DAC1CMP	31:24								
		23:16						HYSPOL	HYSSEL[1:0]	
		15:8						CBE		FLTREN
		7:0	CMPSTAT	CMPPOL	INPSEL[2:0]			INNSEL[2:0]		
0x1D50	DAC1DAT	31:24	DACDAT[31:24]							
		23:16	DACDAT[23:16]							
		15:8	DACLOW[15:8]							
		7:0	DACLOW[7:0]							
0x1D54	DAC1SLPCON	31:24	SLOPEN				HME	TWME	PSE	
		23:16								FFSEN
		15:8	HCFSEL[3:0]				SLPSTOPA[3:0]			
		7:0	SLPSTOPB[3:0]				SLPSTRT[3:0]			
0x1D58	DAC1SLPDAT	31:24								
		23:16								
		15:8	SLPDAT[15:8]							
		7:0	SLPDAT[7:0]							
0x1D5C	DAC2CON	31:24							TMCB[9:8]	
		23:16	TMCB[7:0]							
		15:8	DACEN		IRQM[1:0]			UPDTMDIS		DACOEN
		7:0					UPDTRG[1:0]		UPDATE	UPDREQ
0x1D60	DAC2CMP	31:24								
		23:16						HYSPOL	HYSSEL[1:0]	
		15:8						CBE		FLTREN
		7:0	CMPSTAT	CMPPOL	INPSEL[2:0]			INNSEL[2:0]		
0x1D64	DAC2DAT	31:24	DACDAT[31:24]							
		23:16	DACDAT[23:16]							
		15:8	DACLOW[15:8]							
		7:0	DACLOW[7:0]							
0x1D68	DAC2SLPCON	31:24	SLOPEN				HME	TWME	PSE	
		23:16								FFSEN
		15:8	HCFSEL[3:0]				SLPSTOPA[3:0]			
		7:0	SLPSTOPB[3:0]				SLPSTRT[3:0]			
0x1D6C	DAC2SLPDAT	31:24								
		23:16								
		15:8	SLPDAT[15:8]							
		7:0	SLPDAT[7:0]							
0x1D70	DAC3CON	31:24							TMCB[9:8]	
		23:16	TMCB[7:0]							
		15:8	DACEN		IRQM[1:0]			UPDTMDIS		DACOEN
		7:0					UPDTRG[1:0]		UPDATE	UPDREQ
0x1D74	DAC3CMP	31:24								
		23:16						HYSPOL	HYSSEL[1:0]	
		15:8						CBE		FLTREN
		7:0	CMPSTAT	CMPPOL	INPSEL[2:0]			INNSEL[2:0]		

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1D78	DAC3DAT	31:24	DACDAT[31:24]							
		23:16	DACDAT[23:16]							
		15:8	DACLOW[15:8]							
		7:0	DACLOW[7:0]							
0x1D7C	DAC3SLPCON	31:24	SLOPEN				HME	TWME	PSE	
		23:16								FFSEN
		15:8	HCFSEL[3:0]				SLPSTOPA[3:0]			
		7:0	SLPSTOPB[3:0]				SLPSTRT[3:0]			
0x1D80	DAC3SLPDAT	31:24								
		23:16								
		15:8	SLPDAT[15:8]							
		7:0	SLPDAT[7:0]							
0x1D84	DAC4CON	31:24							TMCB[9:8]	
		23:16	TMCB[7:0]							
		15:8	DACEN		IRQM[1:0]			UPDTMDIS		DACOEN
		7:0					UPDTRG[1:0]		UPDATE	UPDREQ
0x1D88	DAC4CMP	31:24								
		23:16						HYSPOL	HYSSEL[1:0]	
		15:8						CBE		FLTREN
		7:0	CMPSTAT	CMPPOL	INPSEL[2:0]			INNSEL[2:0]		
0x1D8C	DAC4DAT	31:24	DACDAT[31:24]							
		23:16	DACDAT[23:16]							
		15:8	DACLOW[15:8]							
		7:0	DACLOW[7:0]							
0x1D90	DAC4SLPCON	31:24	SLOPEN				HME	TWME	PSE	
		23:16								FFSEN
		15:8	HCFSEL[3:0]				SLPSTOPA[3:0]			
		7:0	SLPSTOPB[3:0]				SLPSTRT[3:0]			
0x1D94	DAC4SLPDAT	31:24								
		23:16								
		15:8	SLPDAT[15:8]							
		7:0	SLPDAT[7:0]							
0x1D98	DAC5CON	31:24							TMCB[9:8]	
		23:16	TMCB[7:0]							
		15:8	DACEN		IRQM[1:0]			UPDTMDIS		DACOEN
		7:0					UPDTRG[1:0]		UPDATE	UPDREQ
0x1D9C	DAC5CMP	31:24								
		23:16						HYSPOL	HYSSEL[1:0]	
		15:8						CBE		FLTREN
		7:0	CMPSTAT	CMPPOL	INPSEL[2:0]			INNSEL[2:0]		
0x1DA0	DAC5DAT	31:24	DACDAT[31:24]							
		23:16	DACDAT[23:16]							
		15:8	DACLOW[15:8]							
		7:0	DACLOW[7:0]							
0x1DA4	DAC5SLPCON	31:24	SLOPEN				HME	TWME	PSE	
		23:16								FFSEN
		15:8	HCFSEL[3:0]				SLPSTOPA[3:0]			
		7:0	SLPSTOPB[3:0]				SLPSTRT[3:0]			
0x1DA8	DAC5SLPDAT	31:24								
		23:16								
		15:8	SLPDAT[15:8]							
		7:0	SLPDAT[7:0]							

19.3.1. DAC Control 1 Register

Name: DACCTRL1
Offset: 0x1D40

Note: These bits should only be changed when DACON = 0 to avoid unpredictable behavior.

Bit	31	30	29	28	27	26	25	24
	RREN		POSINLADJ[5:0]					
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
		NEGINLADJ[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	DNLADJ[4:0]				
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						FCLKDIV[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 31 – RREN Ripple Reduction Enable bit

Value	Description
1	Ripple Reduction mode is enabled.
0	Ripple Reduction mode is disabled.

Bits 29:24 – POSINLADJ[5:0] Positive INL Correction Value bits
The number of ones in this register controls the rise time of the PDM drivers. Reducing the number of ones in this register will increase the rise time. This value is shared by all of the PDM DACs.

Bits 22:16 – NEGINLADJ[6:0] Negative INL Correction Value bits
The number of ones in this register controls the fall time of the PDM drivers. Reducing the number of ones in this register will increase driver fall time. This value is shared by all of the PDM DACs.

Bit 15 – ON Common DAC Module Enable bit

Value	Description
1	Enables DAC modules.
0	Disables DAC modules and disables FSCM clocks to reduce power consumption; any pending Slope mode and/or underflow condition is cleared.

Bit 13 – SIDL DAC Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode.
0	Continues module operation in Idle mode.

Bits 12:8 – DNLADJ[4:0] DNL Adjustment Override bits
Each bit assert to a “1”overrides the DNL pulse stretcher finger control. Each “1” reduces the amount of DNL pulse stretching, thus reducing the amount of DNL correction. This value is shared by all of the PDM DACs.

Bits 2:0 – FCLKDIV[2:0] Comparator Filter Clock Divider bits

Value	Description
111	Divide-by-8
110	Divide-by-7
101	Divide-by-6
100	Divide-by-5
011	Divide-by-4
010	Divide-by-3
001	Divide-by-2
000	1x

19.3.2. DAC Control 2 Register

Name: DACCTRL2
Offset: 0x1D44

Bit	31	30	29	28	27	26	25	24
							SSTIME[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	SSTIME[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8
							TMODTIME[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	TMODTIME[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bits 25:16 – SSTIME[9:0] Time from Start of Transition Mode until Steady-State Filter is Enabled bits

Bits 9:8 – TMODTIME[9:8] Transition Mode Duration bits

Bits 7:0 – TMODTIME[7:0] Transition Mode Duration bits

19.3.3. DAC Control Register

Name: DACxCON
Offset: 0x1D48, 0x1D5C, 0x1D70, 0x1D84, 0x1D98

Note:
1. Changing this bit during application run time may cause unpredictable results.

Bit	31	30	29	28	27	26	25	24
							TMCB[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	TMCB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DACEN		IRQM[1:0]			UPDTMDIS		DACOEN
Access	R/W		R/W	R/W		R/W		R/W
Reset	0		0	0		0		

Bit	7	6	5	4	3	2	1	0
					UPDTRG[1:0]		UPDATE	UPDREQ
Access					R/W	R/W	R	W
Reset					0	0	0	0

Bits 25:24 – TMCB[9:8]

Bits 23:16 – TMCB[7:0] DACx Leading-Edge Blanking bits
These register bits specify the blanking period for the comparator following changes to the DAC output during Change-of-State (COS) for the input signal selected by the HCFSEL[3:0] bits in SLPxCONL.

Bit 15 – DACEN Individual DACx Module Enable bit

Value	Description
1	Enables the DACx module.
0	Disables the DACx module to reduce power consumption; any pending Slope mode and/or underflow conditions are cleared.

Bits 13:12 – IRQM[1:0] Interrupt Mode Select bits

Value	Description
11	Generates an interrupt on either a rising or falling edge detect.
10	Generates an interrupt on a falling edge detect.
01	Generates an interrupt on a rising edge detect.
00	Interrupts are disabled.

Bit 10 – UPDTMDIS Update Transition Mode Disable bit

Value	Description
1	The Transition mode is disabled for DACDATA updates; the output voltage will be smoother, but the DAC may be slower in reaching the target voltage.

Value	Description
0	Transition mode is applied following DACDATA updates to reduce the time to reach the new specified DAC output voltage. DAC output voltage transient ripple may be introduced.

Bit 8 – DACOEN DACx Output Buffer Enable bit

Value	Description
1	DACx analog voltage is connected to the DACOUTx pin.
0	DACx analog voltage is not connected to the DACOUTx pin.

Bits 3:2 – UPDTRG[1:0] Update Trigger Select bits(1)

Value	Description
11	Any write to DACDATxDACDAT[15:0]/DACLOW[15:0] or SLPDATxSLPDAT[15:0] sets the UPDATE bit immediately.
10	After any write(s) to DACDATxDACDAT[15:0]/DACLOW[15:0] or SLPDATxSLPDAT[15:0], a stop condition sets the UPDATE bit.
01	After any write(s) to DACDATxDACDAT[15:0]/DACLOW[15:0] or SLPDATxSLPDAT[15:0], a start condition sets the UPDATE bit.
00	After any write(s) to DACDATxDACDAT[15:0]/DACLOW[15:0] or SLPDATxSLPDAT[15:0], the user must manually set the DACCONx.UPDREQ bit.

Bit 1 – UPDATE DAC Register Update Status/Control bit

Value	Description
1	DAC register update is pending – user data registers are not writable.
0	No DAC register update is pending.

Bit 0 – UPDREQ DAC Register Update Request bit

User software writes a ‘1’ to this bit location to request a DAC register update. The bit location always reads as ‘0’. The UPDATE status bit will indicate ‘1’ when an update is pending.

19.3.4. DACx Control Low Register

Name: DACxCMP
Offset: 0x1D4C, 0x1D60, 0x1D74, 0x1D88, 0x1D9C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						HYSPOL	HYSSEL[1:0]	
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
						CBE		FLTREN
Access						R/W		R/W
Reset						0		0

Bit	7	6	5	4	3	2	1	0
	CMPSTAT	CMPPOL	INPSEL[2:0]			INNSEL[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 18 – HYSPOL Comparator Hysteresis Polarity Select bit

Value	Description
1	Hysteresis is applied to the falling edge of the comparator input.
0	Hysteresis is applied to the rising edge of the comparator input.

Bits 17:16 – HYSSEL[1:0] Comparator Hysteresis Select bits

Value	Description
11	45 mv hysteresis
10	30 mv hysteresis
01	15 mv hysteresis
00	No hysteresis is selected.

Bit 10 – CBE Comparator Blank Enable bit

Value	Description
1	Enables the analog comparator output to be blanked (gated off) during the recovery transition following the completion of a slope operation.
0	Disables the blanking signal to the analog comparator; therefore, the analog comparator output is always active.

Bit 8 – FLTREN Comparator Digital Filter Enable bit

Value	Description
1	Digital filter is enabled.
0	Digital filter is disabled.

Bit 7 – CMPSTAT Comparator Status bits

Current state of the comparator output, including CMPPOL selection.

Bit 6 – CMPPOL Comparator Output Polarity Control bit

Value	Description
1	Output is inverted.
0	Output is noninverted.

Bits 5:3 – INPSEL[2:0] Comparator Positive Input Source Select bits

Value	Description
0101	ALLCMP
0100	Bandgap 0.8V
0011	CMPxD(1)
0010	CMPxC
0001	CMPxB
0000	CMPxA

Bits 2:0 – INNSEL[2:0] Comparator Negative Input Source Select bits

Value	Description
0101	CMPFN
0100	CMPEN
0011	CMPDN
0010	CMPCN
0001	AV_{SS}
0000	DACx

19.3.5. DACx Data Register

Name: DACxDAT
Offset: 0x1D50, 0x1D64, 0x1D78, 0x1D8C, 0x1DA0

Bit	31	30	29	28	27	26	25	24
	DACDAT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DACDAT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DACLOW[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DACLOW[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – DACDAT[31:16] DACx High Data bits
In Hysteretic mode, Slope Generator mode and Triangle mode, this register specifies the high data value and/or limit for the DACx module. Valid values are from 205 to 3890.

Bits 15:8 – DACLOW[15:8] DACx Low Data bits
See DAC output level. In Hysteretic mode, Slope Generator mode and Triangle mode, this register specifies the low data value and/or limit for the DACx module. Valid values are from 205 to 3890.

Bits 7:0 – DACLOW[7:0] DACx Low Data bits
See DAC output level. In Hysteretic mode, Slope Generator mode and Triangle mode, this register specifies the low data value and/or limit for the DACx module. Valid values are from 205 to 3890.

19.3.6. DAC Slope x Control Register

Name: DACxSLPCON
Offset: 0x1D54, 0x1D68, 0x1D7C, 0x1D90, 0x1DA4

Bit	31	30	29	28	27	26	25	24
	SLOPEN				HME	TWME	PSE	
Access	R/W				R/W	R/W	R/W	
Reset	0				0	0	0	

Bit	23	22	21	20	19	18	17	16
								FFSEN
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
	HCFSEL[3:0]				SLPSTOPA[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SLPSTOPB[3:0]				SLPSTRT[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – SLOPEN Slope Function Enable/On bit

Value	Description
1	Enables the slope function.
0	Disables the slope function; the slope accumulator is disabled to reduce power consumption.

Bit 27 – HME Hysteretic Mode Enable bit

Value	Description
1	Enables Hysteretic mode for DACx.
0	Disables Hysteretic mode for DACx.

Bit 26 – TWME Triangle Wave Mode Enable bit

Value	Description
1	Enables Triangle Wave mode for DACx.
0	Disables Triangle Wave mode for DACx.

Bit 25 – PSE Positive Slope Mode Enable bit

Value	Description
1	Slope mode is positive (increasing).
0	Slope mode is negative (decreasing).

Bit 16 – FFSEN Fast First Step Mode Enable bit

Value	Description
1	Fast First Step mode is enabled.
0	Fast First Step mode is disabled.

Bits 15:12 – HCFSEL[3:0] Hysteretic Comparator Function Input Select bits
Refer to Table 19-6 for device-specific HCFSEL bit information.

Bits 11:8 – SLPSTOPA[3:0] Slope Stop A Signal Select bits
The selected Slope Stop A signal is logically OR'd with the selected Slope Stop B signal to terminate the slope function. Refer to Table 19-4 for device-specific SLPSTOPA bit information.

Bits 7:4 – SLPSTOPB[3:0] Slope Stop B Signal Select bits
The selected Slope Stop B signal is logically OR'd with the selected Slope Stop A signal to terminate the slope function. Refer to Table 19-5 for device-specific SLPSTOPB bit information.

Bits 3:0 – SLPSTRT[3:0] Slope Start Signal Select bits
Refer to Table 19-3 for device-specific SLPSTRT bit information.

19.3.7. DAC Slope x Data Register

Name: DACxSLPDAT
Offset: 0x1D58, 0x1D6C, 0x1D80, 0x1D94, 0x1DA8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	SLPDAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SLPDAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – SLPDAT[15:0] Slope Ramp Rate Value bits
The SLPDATx value is in 12.4 format.

19.4. Operation

The High-Speed Analog Comparator with the Slope Compensation DAC module is comprised of various blocks, such as a comparator, DAC, etc. The functionality and configuration of different blocks are discussed in this section.

19.4.1. Comparator Stage

19.4.1.1. Comparator Inputs

The inputs to the comparator module are configured using the DACxCON register. The positive input of the comparator is selected by INPSEL[2:0] bits in DACxCMP[5:3]. The negative input of the comparator is selected by INNSEL[2:0] bits in DACxCMP[2:0],

19.4.1.2. Comparator Outputs

The comparator output can be used to trigger PWM modules or interrupts for actions based on the comparator event. When the digital filter is disabled, the comparator signal is made directly available to the PWM module as a current limit and/or Fault signal. This ensures minimal latency for Current-mode applications and for time-critical (safety) applications. The status signal and the interrupt request signal will be processed by the pulse stretcher circuit. When the digital filter is enabled, the PWM trigger signal, status signal and interrupt request signal are all processed by the pulse stretcher and the digital filter logic. This will cause a delay in the current limit/Fault limit event. The polarity of the comparator output is selected by configuring the CMPPOL bit (DACxCMP[6]). The comparator output can be monitored on the I/O pin by configuring the Peripheral Pin Select (PPS) register.

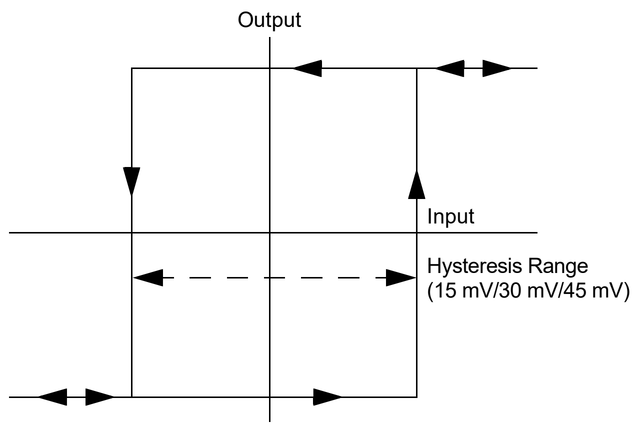
19.4.1.3. Comparator Interrupt

The analog comparator interrupt can be used to service the comparator switching event and can be enabled or disabled from the interrupt controller. The analog comparator interrupt can be configured to interrupt on the rising edge, falling edge or both by setting the IRQM[1:0] bits (DACxCON[14:13]). The comparator interrupt signal is generated on the selected edge of the comparator output, following the polarity processing through the CMPPOL bit (DACxCMP[6]) and the subsequent processing by the pulse stretcher and the digital filter logic. If the CMPPOL bit is changed during operation, the bit change will not cause an interrupt. Only the selected edge of an actual change of the comparator output status will initiate an interrupt.

19.4.1.4. Comparator Hysteresis Control

The HYSSEL[1:0] bits in the DACxCON register specify the amount of hysteresis for the analog comparator. The HYSPOL bit specifies whether hysteresis is applied to the rising edge or falling edge of the signal. Configuration of hysteresis helps the comparator to avoid oscillation (i.e., toggling of the comparator output), which could be caused by noise on the positive input.

Figure 19-2. Hysteresis Control



19.4.1.5. Pulse Stretcher

The High-Speed Analog Comparator can respond to very fast transient signals. To avoid a comparator malfunction, after choosing the comparator output polarity using the CMPPOL bit (DACxCMP[6]), the signal is passed to a pulse stretching circuit. The pulse stretching circuit waits for the comparator output to transition to a high state or a low state and then will stretch the signal for three clock cycles. For example, a comparator output signal of '`01000101000`' will be modified by the pulse stretcher circuit to '`01110111110`'. The pulse stretcher clock operates at a frequency of $F_{DAC}/2$.

19.4.1.6. Digital Filter

In many motor and power control applications, the analog comparator input signals can be corrupted by the large electromagnetic fields generated by the external switching power transistors. Corruption of the analog input signals to the comparator can cause unwanted comparator output transitions. A digital output filter can minimize the effects of the input signal corruption. The digital filter processes the comparator signal from the pulse stretcher circuit. The digital filter is enabled by the FLTREN bit (DACxCON[8]). The digital filter operates with the clock selected by the FCLKDIV[2:0] bits (DACCTRL1[2:0]). The pulse stretcher output signal must be stable, either in a high state or a low state, for at least three times the selected filter clock frequency for it to pass through the digital filter. Assuming the current state is '`0`', a comparator output string of '`0011110000000000`' gets modified by the pulse stretcher to '`0011111100000000`' and to '`0000000001111110`' by the digital filter if the filter clock frequency is divided by two. Because of the requirement of three similar consecutive states for the filter, the selected digital filter clock period must be one third or less than the maximum desired comparator response time. In Sleep mode or Idle mode, the digital filter is bypassed to enable an asynchronous signal from the comparator to the interrupt controller. This asynchronous signal can be used to wake up the processor from Sleep mode or Idle mode. A configuration example to enable the digital filter is provided in Example 19-1.

Example 19-1. Configuration for Digital Filter

```
DACCTRL1bits.FCLKDIV = 1;                     /* Filter Clk Divide by 2 */
DAC1CMPbits.FLTREN = 1;                                     /* Filter enabled*/
```

19.4.2. Pulse Density Modulation (PDM) DAC

Each instance of the High-Speed Analog Comparator with the Slope Compensation DAC has a dedicated DAC that is used to program the comparator threshold voltage via the DACxDAT register. The DAC comprises a digital Pulse Density Modulation (PDM) module, followed by a multistage RC filter. The PDM module generates a high-frequency output signal whose density is proportional to the DACxDAT register value.

The DACDAT[15:0] data limits in the DACxDAT register are bound between 0x0CD and 0xF32 leading to a DAC output of 5% to 95% of V_{DD}. For any intermediate value between 0xCD and 0xF32, the output voltage of the DAC will be proportional. The equation to calculate the DAC output voltage based on V_{DD} voltage source is provided in Equation 19-1. The DAC voltage can be varied in steps of V_{DD}/(2N – 1), where N is the number of DAC bits (N = 12). The DAC modules as a whole are controlled by the ON bit (DACCTRL1[15]). The ON bit enables or disables all comparator/DAC modules instantiated on a given device or device core. The DACEN bit (DACxCON[15]) provides individual control of the DAC module. The individual DAC registers have an output enable bit, DACOEN (DACxCON[8]), which enables the DAC output voltage to be routed to an external output pin, DACOUTx. The DACOUTx pin can only be associated with a single DAC. If more than one DACOEN bit is set, the DACOUTx pin will be a combination of the signals. For devices with more than one DAC output buffer, the odd numbered DAC outputs connect to DACOUT1 with the even channels connecting to DACOUT2. A configuration example to set the DAC output voltage is shown in Example 19-2.

Equation 19-1.

$$V_{DAC} = DACOUT1 \bullet (V_{DD})/4095$$

Where: 0x0CD ≤ DACxDATbits.DACDAT ≤ 0xF32

Example 19-2. Configuration of DAC Register

```
/* DAC Register Settings */
DAC1DATbits.DAC1DATbits.DACDAT = 0x4D9;       /* DAC Output set to 1V (VDD =
3.3V)*/
DAC1CONbits.DACOEN = 1;                /* Enable DAC 1 output on pin DACOUT1 */
DAC1CONbits.DACEN = 1;                 /* Enable DAC 1 */
DACCTRL1bits.ON = 1                    /* Turn ON all DACs */
```

19.4.3. DAC Output Filter Modes

The PDM DAC requires an output filter to convert the digital pulse stream into an analog signal. The output filter is implemented as a configurable multistage RC network. The output filter has three operating modes, as shown in Table 19-8.

Table 19-8. Output Filter Modes

Output Filter Mode	Response Time	Filter Bandwidth
Transition mode	Fastest	Wider
Fast mode	Medium	Medium
Steady-State mode	Slowest	Narrower

Each mode has more filter stages. Transitions are controlled with a set of programmable timers, TMODTIME[9:0] (DACCTRL2L[9:0]) and SSTIME[9:0] (DACCTRL2H[9:0]). TMODTIME defines the duration of Transition mode and SSTIME defines the hold-off time until Steady-State mode. The delta between them is effectively the duration of Fast mode. A write to the data registers DACDAT (and DACLOW, if used) will invoke transition mode and start both TMODTIME and SSTIME timers. Transition and Fast modes can be omitted by setting the timers to zero.

Equation 19-2. Transition Mode Timer Equations

$$TMODTIME[9:0] = T_{TR} * F_{DAC}/2$$

$$T_{TR} = \text{TMODTIME}[9:0] * 2/F_{DAC}$$

Equation 19-3. Steady-State Mode Timer Equations

$$SSTIME[9:0] = T_{SS} * F_{DAC}/2$$

$$T_{SS} = \text{SSTIME}[9:0] * 2/F_{DAC}$$

The control scheme is dependent on the operating mode of the DAC. The operating modes are:

- Static (fixed DC voltage)
- Slope
- Ramp (triangle wave)
- Hysteretic

19.4.3.1. Static Operating Mode

The static operating mode will output a fixed DC voltage as defined by DACDAT. Repeated writes will reset the timers and may prevent the DAC from reaching the other filter modes. It is not recommended to repeatedly write the same DACDAT value, as it incurs transients on the output when filters are switched. In Static mode, the filters operate in the order of Transition, Fast and Steady-State modes, as defined by Table 19-9

Table 19-9. Static Operating Mode Output Filter Timer Conditions

Output Filter Mode	Entry Condition	Exit Condition
Transition mode	Data write	TMODTIME expires
Fast mode	TMODTIME expires	SSTIME expires data write
Steady-State mode	SSTIME expires	Data write

Figure 19-3. Static Mode Output Filter Timing

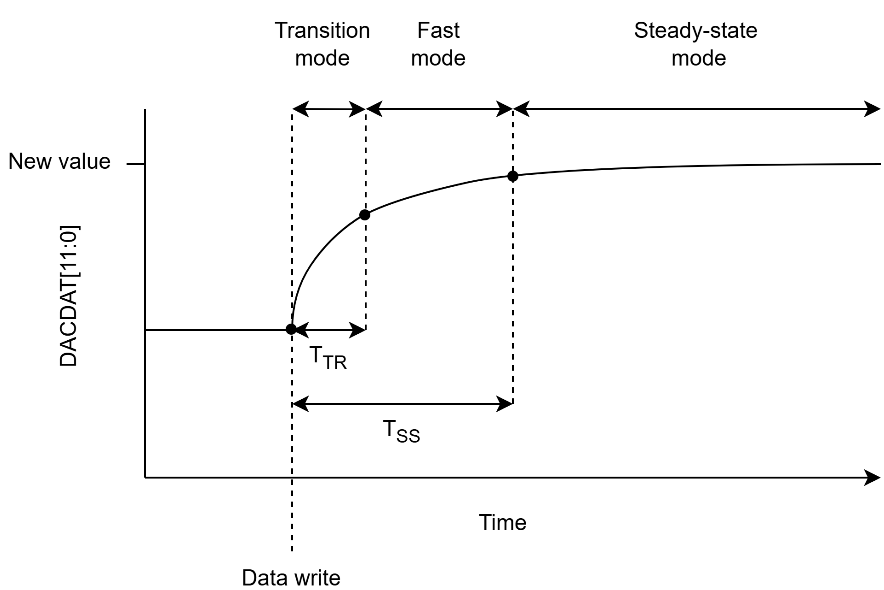


Table 19-10. Static Mode Timer Configurations

TMODTIME	SSTIME	Result
0	0	Immediately to Steady State
0	1 to 0x3FF	Fast mode until SSTIME, Steady State (recommended)
1 to 0x3FF	1 to 0x3FF Where SSTIME > TMODTIME	Transition until TMODTIME, Fast mode until SSTIME, Steady State

19.4.4. Slope Generator

The function of the slope generator is to vary the DAC data value at a user-defined rate to reach a desired endpoint value. The slope generator, along with the DAC, has three modes of operation: Slope Generation mode, Hysteretic mode and Triangle Wave mode.

19.4.4.1. Slope Generation Mode

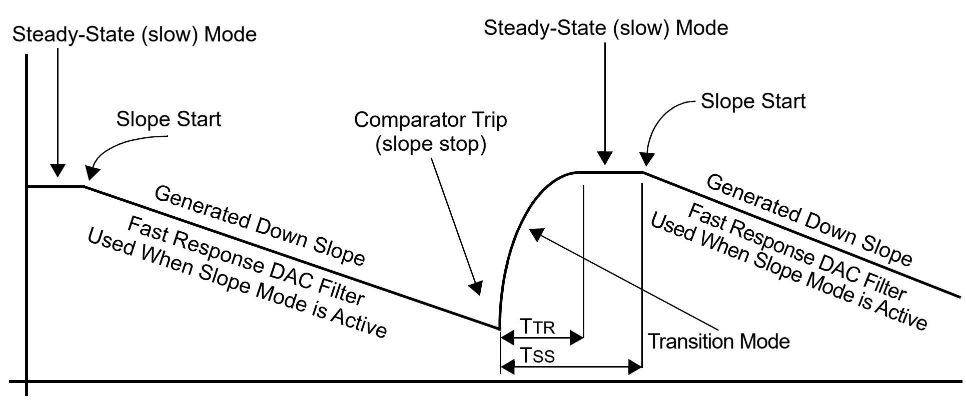
The slope generator function can be utilized in Peak Current-mode control-based power supply applications, where slope compensation is required. The slope function modifies the non-slope PDM DAC value repeatedly, at a user-defined rate, until the DAC data value reaches its endpoint. The slope generation function can be enabled or disabled by the SLOPEN bit (SLPxCON[31]). The slope rate is controlled by the data in the SLPxDAT register. The direction of the slope, being positive or negative, is controlled by the PSE bit (SLPxCON[25]). For negative slopes (default, PSE = `0`), DACDAT holds the nominal non-slope count, while DACLOW holds the count corresponding to the end of the slope. For positive slopes (PSE = `1`), DACLOW holds the nominal non-slope count, while DACDAT holds the count corresponding to the end of the slope. If new DACDAT or DACLOW values are written during Slope Operation mode, a transient may occur. Data writes will put the DAC in Transition mode, which has the fastest response time but reduced filtering.

In Slope mode, the order of output modes is different from Static mode. When the slope cycle begins, the DAC starts in Transition mode, then moves to Steady-State mode, and lastly to Fast mode for the ramp. Both TMODTIME and SSTIME timers are started at the cycle start. TMODTIME defines the duration of Transition mode, and SSTIME defines the end of steady-state time. Therefore, the duration of steady-state is effectively TSS - TTR.

Note that the SSTIME[9:0] count should always be greater than the TMODTIME[9:0] count. At the end of the Steady-State mode, the DAC value settles at the new value and is ready for slope generation. The SLPSTRT[3:0] signal triggers the slope generation process.

Refer to the DAC electrical specifications for additional information on T_{SS} and T_{TR} values. These timing parameters can be further adjusted as needed for the application.

Figure 19-4. Slope Generation Mode DAC Output Waveform



The slope generation is terminated when one of the two stop signals is asserted. The eight control register bits, SLPSTOPA[3:0] (SLPxCON[11:8]) and SLPSTOPB[3:0] (SLPxCON[7:4]), select the control signal to terminate the slope generation. The stop signals are logically ORed so that the slope is terminated when one of the trigger events materializes. In most power supply applications, SLPSTOPA[3:0] can be configured to terminate the slope at the end of the PWM cycle, while SLPSTOPB[3:0] can be configured to trigger when the current reaches a limit under a normal or Fault condition. It should be noted that the stop signal must terminate the slope at least T_{SS} (Steady-State Time) prior to the next PWM cycle start. This is necessary to allow the DAC value to reach and settle at the steady-state value, specified by DACDAT, before the next cycle begins.

The slope rate value to be specified in the SLPxDAT register depends on the start and end values of the slope specified by DACDAT and DACLOW, the PWM time period, the DAC clock frequency and the SSTIME[9:0] bits value. The SLPxDAT value can be determined by using Equation 19-4.

Equation 19-4. Determining the SLPxDAT Value

$$SLPxDAT = \frac{(DACDAT - DACLOW) \bullet 16}{(T_{SLOPE\_DURATION})/T_{DAC}}$$

Where:

DACDAT = DAC value at the start of slope.

DACLOW = DAC value and the end of slope.

$T_{SLOPE\_DURATION}$ = Slope duration time in seconds

$T_{DAC} = 2/F_{DAC}$ in seconds

Note: Multiplication by 16 sets results in the SLPxDAT value in 12.4 format.

19.4.4.2. Hysteretic Mode

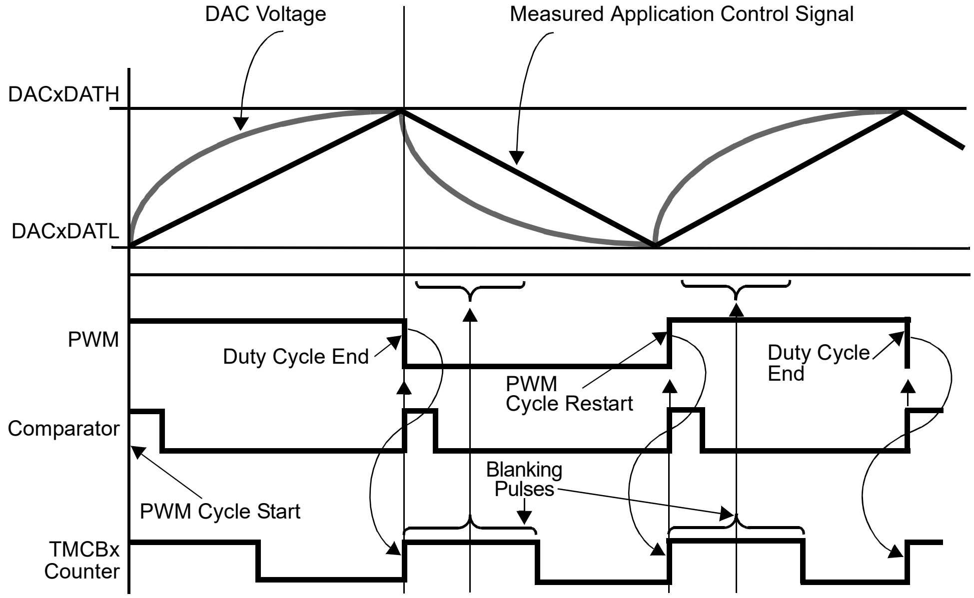
Hysteretic mode control is sometimes called “Bang-Bang” control, where a signal within a power converter is controlled within an upper cutoff and a lower cutoff limit. The Hysteretic mode is used in power supply applications utilizing hysteretic control, such as LED drivers. The Hysteretic mode is enabled by the HME bit (SLPxCON[27]) and requires the SLOPEN bit to be cleared. Hysteretic Control mode enables a single DAC and comparator to monitor both the high and low limits for a signal. DACDAT provides the higher value, while DACLOW provides the lower value. When the DAC changes direction, it uses Transition mode to respond and reach the new value as fast as possible. In Hysteretic mode, the comparator effectively functions as a window comparator. The DAC output races ahead of the monitored voltage in the application circuit. While the DAC is transitioning to the new value, the comparator output is “blanked” via the TMCB[9:0] bits (DACxCON[23:16]) to prevent spurious responses. The state of the PWM output is monitored via the input multiplexer controlled by the HCFSEL[3:0] bits of the SLPxCON register. This module monitors the actual state of the PWM output rather than making assumptions that could damage the application circuit. A configuration example to use Hysteretic mode is shown in Figure 19-5.

Equation 19-5. Transition Mode Comparator Blanking Equations

$TMCB[9:0] = T_{TCB} * F_{DAC}/2$

$T_{MCB} = TMCB[9:0] * 2/F_{DAC}$

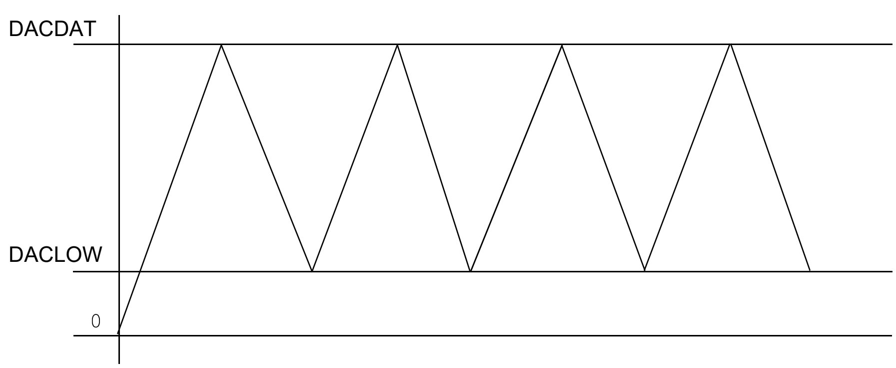
Figure 19-5. Hysteretic Mode DAC Output Waveform



19.4.4.3. Triangle Wave Mode

Triangle Wave mode generates an output voltage that rises and falls with a triangle wave pattern. The Triangle Wave mode is enabled by the bit TWME (SLPxCON[26]) and requires SLOPEN to be set to ‘1’. The high and low points of the waveform are specified via DACDAT and DACLOW. The rise and fall times and the frequency of the triangle wave are controlled via the SLPxDAT register. The very first clock cycle of the slope process selects a scaled SLPxDAT value, instead of the specified value, to provide a prompt DAC response to the DAC trajectory. For all subsequent clock cycles of the slope process, the slope generator uses the specified SLPxDAT data value for incrementing/decrementing the DAC data value. The Fast DAC mode is used in the Triangle Wave mode and is used to provide a fast response. The slope changes direction automatically after reaching either the DACDAT or DACLOW value. The Triangle Wave mode is useful in digital audio applications, where an analog input signal is sampled via an analog comparator using a triangle wave reference signal (Figure 19-6).

Figure 19-6. Triangle Wave Mode



A configuration example to set the Triangle Wave mode is shown in Example 19-3.

Example 19-3. Triangle Wave Mode Configuration[1]

```
/* Triangle Wave Mode Settings */
DAC1DATbits.DACLOW = 0x100;          /* Lower data value */
DAC1DATbits.DACDAT = 0xF00;          /* Upper data value */
DACSLP1DATbits.SLPDAT = 0x1;           /* Slope rate, counts per step */
DACSLP1CONbits.TWME = 1;               /* Enable Triangle Mode */
DACSLP1CONbits.SLOPEN = 1;             /* Enable Slope Mode */
```

Note:

1. The maximum value of DACDAT must be set at 0xF32 – SLPxDAT, and the minimum value of DACLOW must be set at 0xCD + SLPxDAT.

19.4.5. Operation in Sleep and Idle Mode

During Sleep mode, the High-Speed Analog Comparator operates with reduced functionality, allowing the device to wake up when an active signal is applied to the comparator input. To reduce power consumption when the device enters Idle mode, the comparator module can be disabled by setting the DACSIDL bit (DACCTRL1[13]). The DACSIDL bit controls all the comparators on a device or device core.

19.5. Application Examples

The High-Speed Analog Comparator with the Slope Compensation DAC can be used in many power conversion applications. The outputs of the comparator module can be used to perform the following functions:

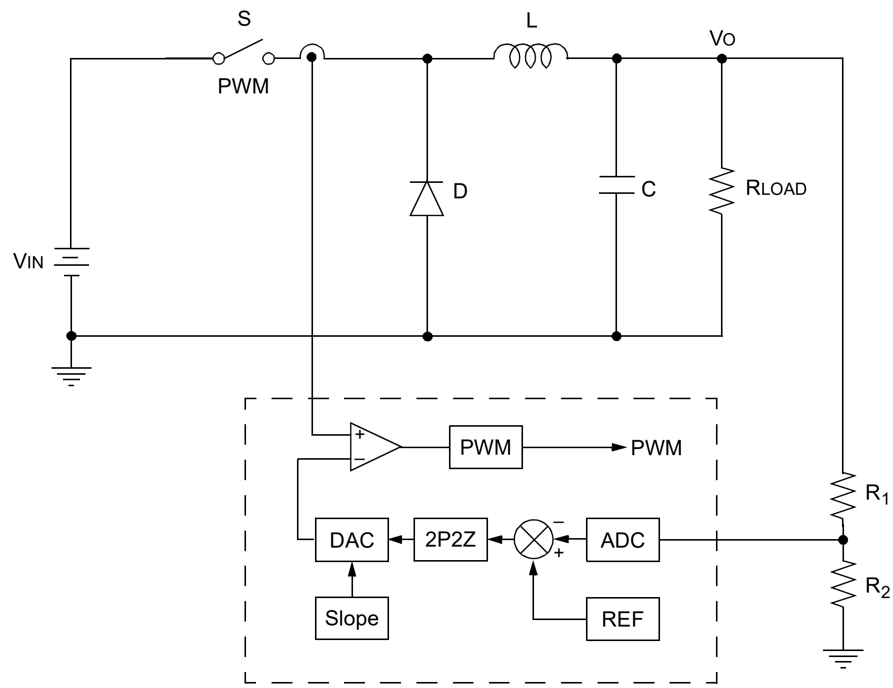
- Generate an Interrupt
- Trigger an ADC Sample and Convert Process
- Truncate the PWM Signal (Current Limit)
- Truncate the PWM Period (Current Reset)
- Extend the PWM Period (Feed Forward)
- Disable the PWM Outputs (Fault Latch)

The output of the comparator module can be used in multiple modes at the same time. For example, the comparator output can be used to generate an interrupt, have the ADC take a sample and convert it and truncate the PWM output, all in response to a voltage being detected beyond its expected value. The SMPS analog comparator module can also be used to wake up the system from Sleep mode or Idle mode when the analog input voltage exceeds the programmed threshold voltage. The slope compensation module allows the user to utilize built-in hardware-based slope compensation in SMPS applications. The potential applications of the comparator module are numerous and varied.

19.5.1. Peak Current-Mode Control

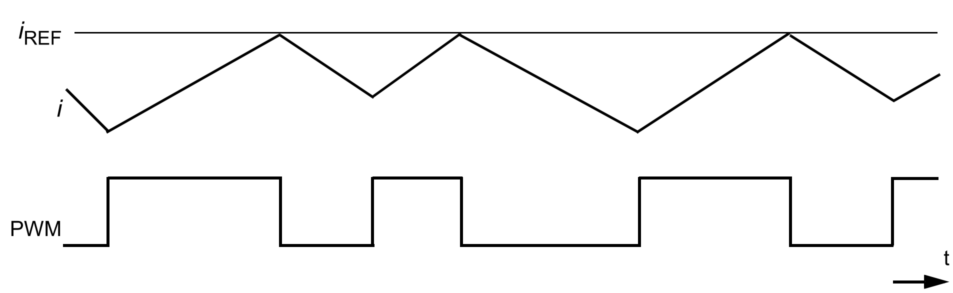
The SMPS topologies, such as Buck, Boost and Buck-Boost, generate subharmonic oscillations when controlled with Peak Current-mode control. These oscillations occur under specific conditions, such as Continuous Current mode and a duty cycle greater than 50%. The subharmonic oscillations can be damped by using slope compensation. The analog comparator module can be utilized for such applications, eliminating the need for additional external analog circuitry to perform slope compensation. The comparator module is used in conjunction with the PWM module to generate the Current-mode PWM signal. A typical Peak Current Buck mode power supply is illustrated in Figure 19-7.

Figure 19-7. Buck Converter with Peak Current-Mode Control



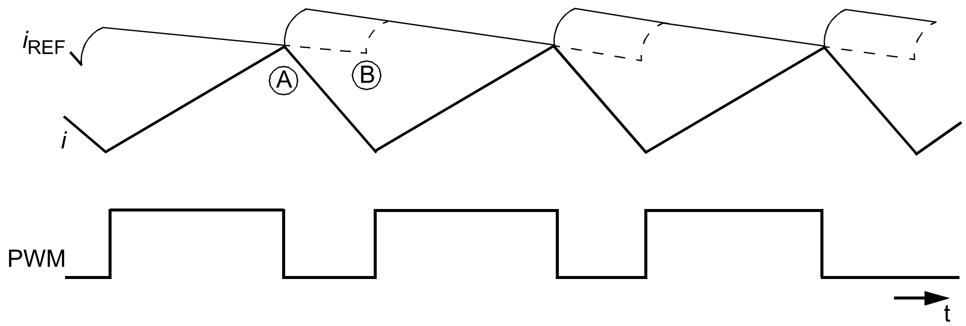
The analog comparator module is configured to reset the PWM module when the measured inductor current peak reaches the current level determined by the outer control loop. The outer control loop consists of the output voltage, measured by the ADC, and compared with a desired voltage reference. The error counts generated are treated with a compensator gain to arrive at the peak current level for the current PWM cycle. The peak current level is applied to the DAC to generate an equivalent analog signal with which the actual inductor current is compared by the comparator. The waveforms of the Peak Current-mode control are as shown in Figure 19-8. Note that the pulse width is different in the consecutive cycles, even though the current reference, i_{REF}, is constant.

Figure 19-8. Peak Current-Mode Waveform without Slope Compensation



The slope compensation module alters the DAC output voltage slope, hence the reference to the inner current loop. In the absence of the slope compensation module, the output of the DAC is held constant for a given PWM cycle (as shown in Figure 19-9). The slope generation module causes the reference current to slope based on the values set in the register. The slope direction can be set to positive or negative depending on the applications, although negative slope is generally used. The rate of the slope is determined by the SLPxDAT register. DACDAT holds the DAC value at the start of the PWM cycle, and the DAC value at the end of the PWM cycle is held by DACLOW. The waveforms of the Peak Current-mode control with slope compensation are as shown in Figure 19-9. Note that the pulse width is the same in all cycles for a constant current reference, i_{REF}.

Figure 19-9. Peak Current-Mode Waveform with Slope Compensation



Example 19-4 shows the settings of the analog comparator module for generating the slope compensated waveforms. In this design, the buck converter operates at an input of 5V and an output of 3.3V, 1A. The operating frequency of the converter is 200 kHz. The period timer for the PWM is set at 200 kHz, and the duty cycle is set at 95%. The clock frequency for the DAC module is set at 500 MHz. The current measurement is connected to the positive input of the comparator. The DAC provides the reference current for the peak current trip and is connected internally to the negative input of the comparator. The reference current is generally the output of the compensator (digital filter), which operates on the outer voltage loop error signal. The PWM cycle is terminated when the measured input current exceeds the DAC reference current, i_{REF}, as shown by Point A in Figure 19-9. This is due to configuration of the SLPSTOPBx trigger in Figure 19-7.The dashed curve shows the reference current, i_{REF}, if SLPSTOPBx is not triggered. In this case, i_{REF} continues until the SLPSTOPAx signal is triggered and is indicated by Point B in Figure 19-9.

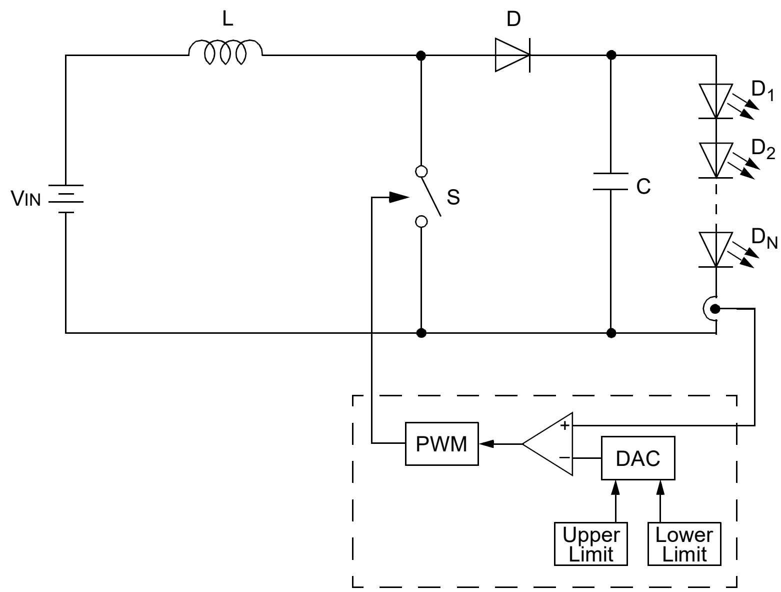
Example 19-4. Initialize DAC with Slope Compensation

```
//Before this example, configure CLKGEN7 @500MHz(the DAC clock)
//PWM clock settings
    PCLKCONbits.MCLKSEL = 1;     //Use CLKGEN5 as PWM MCLK
    PG1CONbits.CLKSEL = 1;       //Use PWM Clock selected by MCLKSEL, undivided and
unscaled

    //PWM clocking based on 200MHz MCLK from CLKGEN5. Each unit is 1/16 of an MCLK period.
    PG1PER = 1000 << 4;         // PWM frequency is 200 kHz, 5 uS period
    PG1DC = 950 << 4;           // 95% duty cycle, 4.75 uS on time

    PG1IOCON1bits.PENH = 1;     // PWM Generator controls the PWMxH output pin
    PG1IOCON1bits.PENL = 1;     // PWM Generator controls the PWMxL output pin

    // PWM PCI setup, use CLDAT when comparator 1 trips
    PG1CLPCI2 = 1 << 28;        // PCI source is Comparator 1
    PG1CLPCI1bits.AQSS = 2;     // LEB active as Acceptance Qualifier
    PG1CLPCI1bits.AQPS = 1;     // Invert Acceptance Qualifier (LEB not active)
    PG1CLPCI1bits.TERM = 1;     // Auto terminate as Termination Event
    PG1CLPCI1bits.ACP = 3;      // Latched PCI Acceptance Criteria
    PG1IOCON2bits.CLDAT = 0b01; // PWM1L = 1 and PWM1H = 0 if CL event is active
    PG1LEBbits.PHR = 1;         // Rising edge of PWMxH triggers the LEB counter

    PG1LEB = 100 << 4; // 500 nS LEB timer

    // PWM to DAC Trigger setup
    PG1TRIGDbits.TRIGD = 300 << 4; // DAC Trigger 1 at 1.5 uS, used as SLPSTRT
    PG1EVT1bits.DACTREN1 = 1;      // PGxTRIGA as trigger source for ADC Trigger 1
    _RP143R = 2;                   // Remap PWM1L to virtual pin RPV14 which is used as
SLPSTOPA
    PG1CONbits.ON = 1;             // Enable PWM

    // DAC Configuration
    DACCTRL2bits.SSTIME = 0x8A;   // Default value 552 nS @ 500MHz
    DACCTRL2bits.TMODTIME = 0x55; // Default value 340 nS @ 500MHz

    DAC1DATbits.DACDAT = 2703;    // 2.17v steady state value
    DAC1DATbits.DACLOW = 1113;    // 0.89v, value at the end slope


    DAC1SLPDATbits.SLPDAT = 41;        // Slope = (2703-1113)*16/((4u-1.5u)/4n)
    DAC1SLPCONbits.SLOPEN = 1;         // Enable Slope compensation
    DAC1SLPCONbits.SLPSTRT = 0b0001;   // PWM1 Trigger
    DAC1SLPCONbits.SLPSTOPA = 0b1101;  // RPV14 (PWM1)Trigger
    DAC1SLPCONbits.SLPSTOPB = 0b1110;  // RPV15 (CMP1)Trigger
    DAC1CMPbits.CBE = 1;               // Enable comparator blanking
    DAC1CONbits.TMCB = 125;            // 125 * 4 nS = 500 nS blanking time
    DAC1CONbits.DACOEN = 1;            // Enable DAC output
    DAC1CONbits.DACEN = 1;             // Enable DAC 1
    DACCTRL1bits.ON = 1;               // Enable DAC system
```

19.5.2. Hysteretic Control for LED Drivers

The hysteretic control offers the fastest response to the changing parameters, such as voltage or current. The hysteretic control finds wide usage in LED applications, where the current is required to be in a limited range around the average value. The disadvantage of the hysteretic topology is the variable frequency of operation. Figure 19-10 shows an example circuit where the LED current is controlled to an average value with a tolerance decided by the register values.

Figure 19-10. Hysteretic Control for LED Drivers



The comparator uses Hysteretic mode for such applications. The Hysteretic mode is controlled by the HME bit in the SLPxCON register. In order to enable the Hysteretic mode, the HME bit must be set to '1' and the SLOPEN bit is cleared. The upper limit of the hysteretic control is defined by DACDAT, while the lower limit is defined by DACLOW. In Hysteretic mode, the comparator module has a pair of output signals that are available as Peripheral Pin Select (PPS) inputs: PWM_Req_on and PWM_Req_off. These signals can then be mapped to the PWM PCI input for controlling the PWM outputs. The DAC settings for Hysteretic mode are shown in Example 19-5.

Example 19-5. DAC Settings for Hysteretic Mode

```
/* PWM Clock Selection */
    PCLKCONbits.MCLKSEL = 1;              // Master Clock Source is CLKGEN5
    PG1CONbits.CLKSEL = 1;                // Clock selected by MCLKSEL

    PG1IOCON1bits.PENH = 1;               // Enable H output
    PG1IOCON1bits.PENL = 1;               // Enable L output
    PG1IOCON2bits.FFDAT = 0b11;           // FF PCI data is 0b11

    // PPS setup
    RPINR8bits.PCI8R = 142;              // 'PWM_Req_On' from DAC to PCI8
    RPINR8bits.PCI9R = 143;              // 'PWM_Req_Off' from DAC to PCI9

    // FF PCI setup
    PG1FFPCI1bits.TSYNCDIS = 1;          // Terminate immediately
    PG1FFPCI1bits.TERM = 0b111;          // PCI 9 (PWM_Req_Off)
    PG1FFPCI2 = 1 << 8;                  // PCI 8 (PWM_Req_On)
    PG1FFPCI1bits.ACP = 0b100;           // Latched rising edge
    PG1CONbits.ON = 1;                   // Enable PG1

    // DAC initialization

    DAC1DATbits.DACLOW = 0x400;         // Lower cmp limit, 0.825 V
    DAC1DATbits.DACDAT = 0xC00;         // Upper cmp limit, 2.475 V

    DAC1CMPbits.CBE = 1;                // Enable comparator blanking
    DAC1CONbits.TMCB = 100;             // 2/500 MHz * 100 = 400 nS
    DAC1SLPCONbits.HCFSEL = 1;          // 1 = PWM1H
    DAC1SLPCONbits.HME = 1;             // Hysteretic Mode
    DAC1CMPbits.INPSEL = 1;             // CMP1B input
    DAC1CONbits.DACOEN = 1;             // Output DAC voltage to DACOUT1 pin
```

```
    DAC1CONbits.DACEN = 1;              // Enable DAC module
    DACCTRL1bits.ON = 1;                // Enable DAC1
```

19.5.2.1. Using DAC as Voltage Reference for External Comparator

If the DAC output is used as a reference voltage for a comparator external to the dsPIC33A device, the Transition mode's transient response can cause unwanted comparator trips when changing DACDAT values. When the DAC is in Transition mode, the blanking feature (CBE = `1`) can be set to mask the internal comparator's output. However, this blanking feature is not available on an external comparator.

A work around for this use case is to set the TMODTIME[9:0] bits in the DACCTRL2 register to zero to disable Transition mode. The DAC will slew slower to the new target voltage.

20. Quadrature Encoder Interface (QEI)

This section describes the Quadrature Encoder Interface (QEI) implemented in the dsPIC33AK256MPS306 family of devices. The QEI is typically used in motor control applications to detect the mechanical position, direction of rotation and speed of rotation of quadrature encoders. The following high-level features are covered in this section:

- Four Input Pins: Two Phase Signals, an Index Pulse and a Home Pulse
- Programmable Digital Noise Filters on Inputs
- Quadrature Decoder Providing Counter Pulses and Count Direction
- x4 Count Resolution
- Index Pulse to Reset the Position Counter
- General Purpose 32-bit Timer/Counter Mode
- Interrupts Generated by QEI or Counter Events
- 32-bit Velocity Counter
- 32-bit Position Counter
- 32-bit Index Pulse Counter
- 32-bit Interval Timer
- 32-bit Position Initialization/Capture/Compare High Word Register
- 32-bit Position Initialization/Capture/Compare Low Word Register
- 4X Quadrature Count Mode
- External Up/Down Count Mode
- External Gated Count Mode
- External Gated Timer Mode
- Interval Timer Mode
- Direction Change Indication
- Input Override Capability
- – Includes forced input override capability for use by Functional Safety-focused customers
- Check Invalid Encoder Transition

The following high-level Hall decoder features are covered in this section:

- Hall Sensor
  - Hall code detection
  - Check valid Hall transitions
  - Programmable event generation delay after a Hall transition
- Window monitoring of Hall transitions

20.1. Device-Specific Information

Table 20-1. QEI Summary

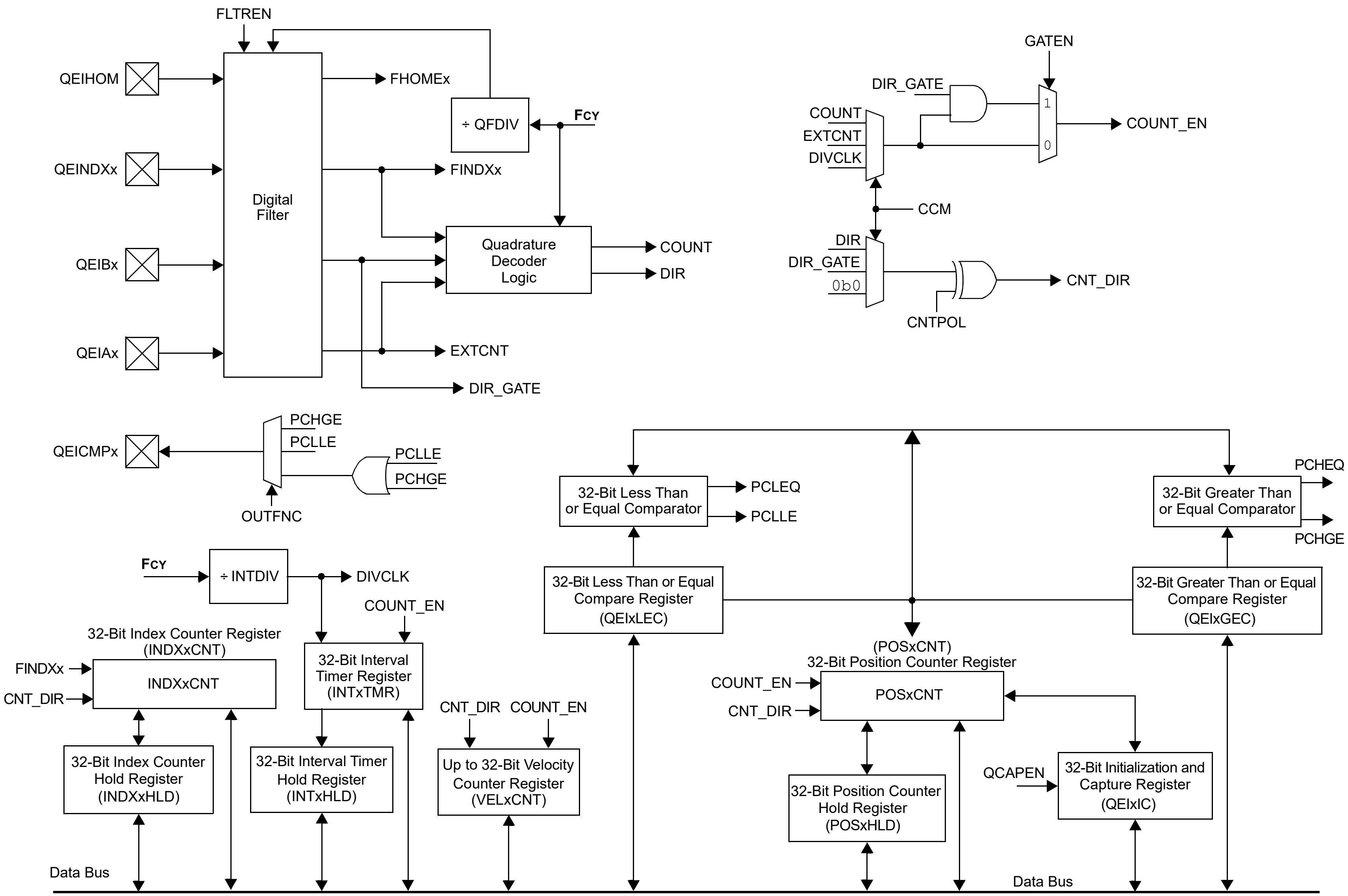
QEI Module Instances	Peripheral Bus Speed	Clock Source
1	Standard	Standard (1:2 CPU Clock)

20.2. Architectural Overview

The Quadrature Encoder Interface (QEI) module provides the interface to incremental encoders for obtaining mechanical position data. Quadrature encoders, also known as incremental encoders or optical encoders, detect the position and speed of rotating motion systems. Quadrature encoders enable closed-loop control of motor control applications, such as Switched Reluctance (SR) and AC Induction Motors (ACIM).

The Hall sensor interface provides the mechanical position data of the rotor. In order to energize the correct stator winding, the rotor position must be known. Typically, the BLDC motors have three Hall effect sensors mounted to the stator and use six-step commutation to drive the motor. When the rotor passes a sensor, it produces either a high or a low signal to indicate which rotor pole (N or S) has passed. This switching of the three Hall effect sensors (from high to low or from low to high) provides the rotor position.

Figure 20-1. Quadrature Encoder Interface (QEI) Module Block Diagram



20.2.1. Quadrature Encoder Interface

A typical quadrature encoder includes a slotted wheel attached to the shaft of the motor and an emitter/detector module that senses the slots in the wheel. Typically, three output channels, Phase A (QEAx), Phase B (QEBx) and Index (INDXx) provide information on the movement of the motor shaft, including distance and direction.

The two channels, Phase A (QEA) and Phase B (QEB), are typically 90° out of phase with respect to each other. The Phase A and Phase B channels have a unique relationship. If Phase A leads Phase B, the direction of the motor is deemed positive or forward. If Phase A lags Phase B, the direction of the motor is deemed negative or reverse. The index pulse occurs once per mechanical revolution and is used as a reference to indicate an absolute position. Figure 20-2 illustrates the Quadrature Encoder Interface signals.

The quadrature signals from the encoder can have four unique states (`'01'`, `'00'`, `'10'` and `'11'`) that reflect the relationship between QEA and QEB. Figure 20-2 illustrates these states for one count cycle. The order of the states get reversed when the direction of travel changes.

The quadrature decoder increments or decrements the 32-bit up/down Position Counter (POSxCNT) for each Change-of-State (COS). The counter increments when QEA leads QEB and decrements when QEB leads QEA.

Figure 20-2. Quadrature Encoder Interface Signals

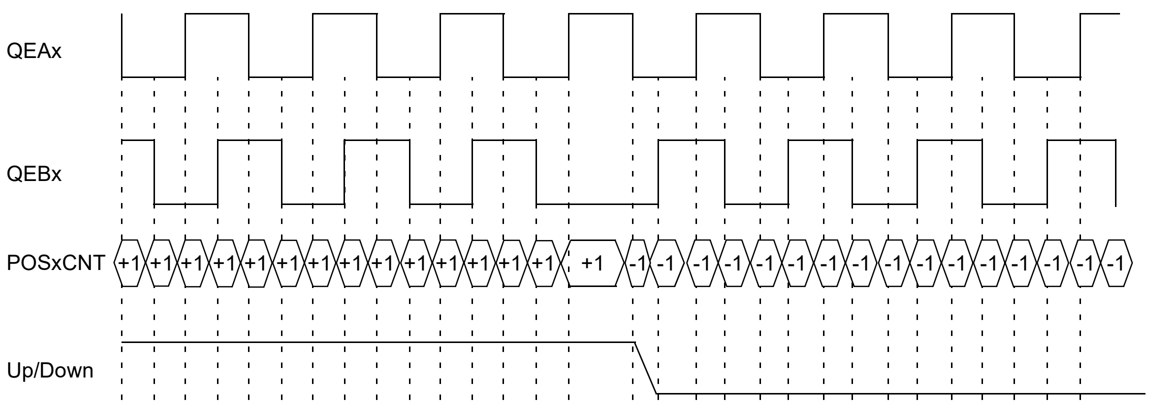


Table 20-2 shows the truth table that describes how the quadrature signals are decoded.

Table 20-2. Truth Table for Quadrature Encoder

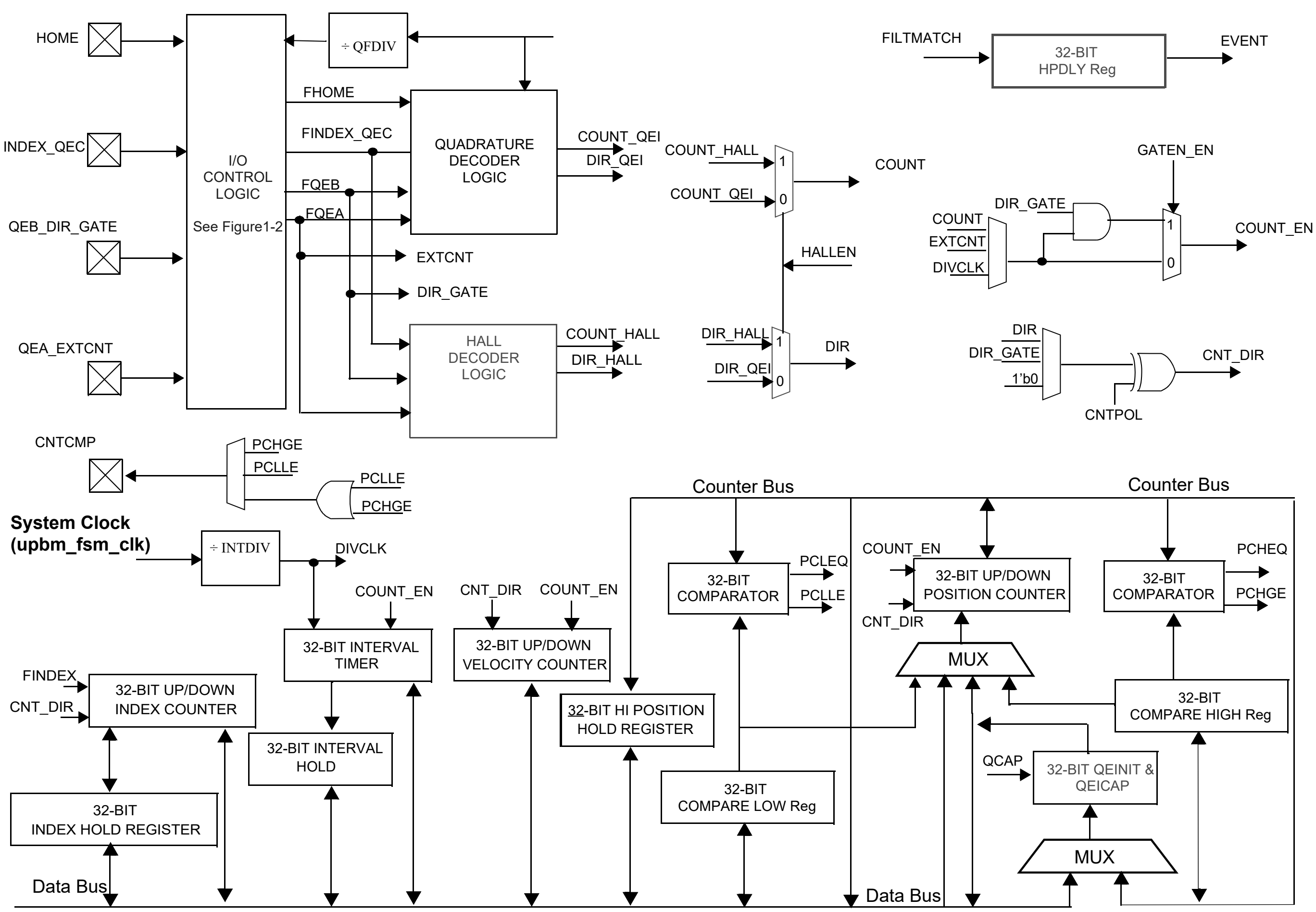
Current Quadrature State		Previous Quadrature State		Action
QEA	QEB	QEA	QEB	
1	1	1	1	No count or direction change
1	1	1	0	Count up
1	1	0	1	Count down
1	1	0	0	Invalid state change, ignore
1	0	1	1	Count down
1	0	1	0	No count or direction change
1	0	0	1	Invalid state change, ignore
1	0	0	0	Count up
0	1	1	1	Count up
0	1	1	0	Invalid state change, ignore
0	1	0	1	No count or direction change
0	1	0	0	Count down

Table 20-2. Truth Table for Quadrature Encoder (continued)

Current Quadrature State		Previous Quadrature State		Action
QEA	QEB	QEA	QEB	
0	0	1	1	Invalid state change, ignore
0	0	1	0	Count down
0	0	0	1	Count up
0	0	0	0	No count or direction change

Figure 20-3 illustrates the simplified block diagram of the QEI module. If the QEI module is used as a quadrature decoder, then the QEI module consists of decoder logic to interpret the Phase A (QEA) and Phase B (QEB) signals, and an up/down counter to accumulate the count. If the QEI module is used as a Hall decoder, then the QEI module consists of decoder logic to interpret the Phase A (QEA), Phase B (QEB) and Phase C (QEC) signals. The counter pulses are generated when the quadrature state/Hall state changes. The count direction information must be maintained in a register until a direction change is detected. The module also includes digital noise filters that condition the input signal.

Figure 20-3. Quadrature Encoder Hall Sensor Interface (QEI) Module Block Diagram



20.2.2. Hall Sensor Interface

A typical Hall effect sensor used in motor control has three digital outputs: Phase A (QEA), Phase B (QEB) and Phase C (QEC). The Hall effect sensors provide rotor position with a resolution of 60 electrical degrees, or which sector of the rotor is located.

Figure 20-4. Hall Sensor Encoder Interface Signals

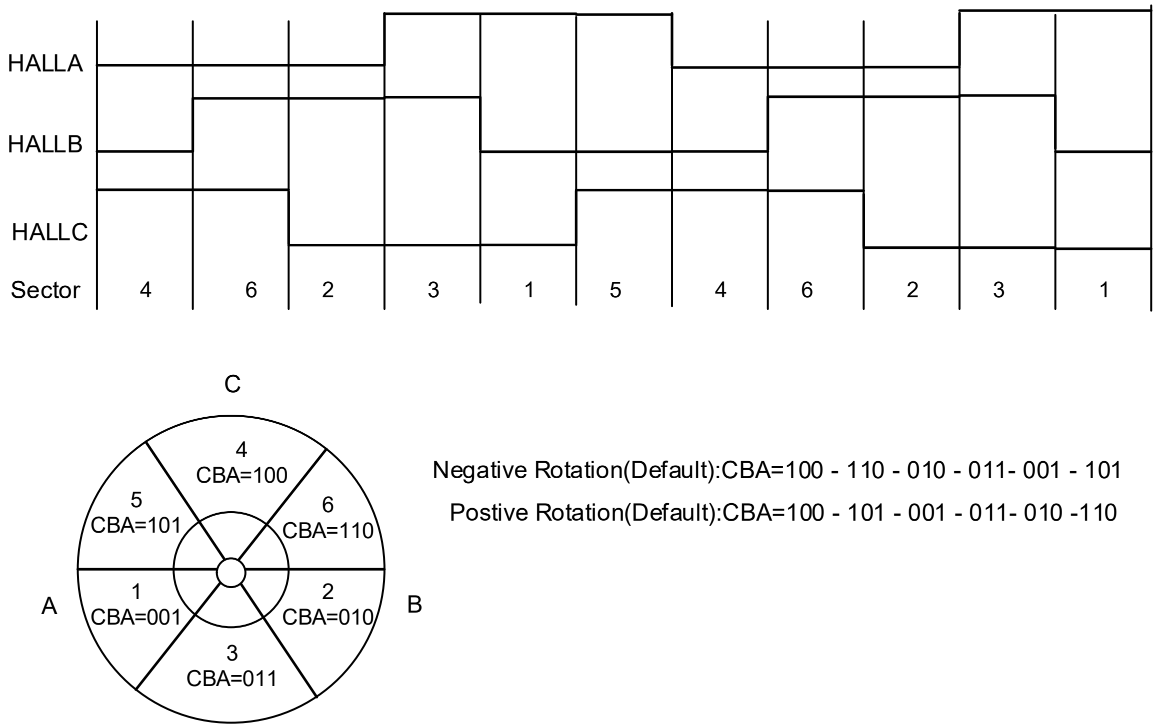


Table 20-3. Truth Table for Hall Decoder(1), (2), (3), (4)

Current Hall State (CBA)	Previous Hall State (CBA)	Current Sector	Previous Sector	Action
100	100	4	4	No count, no direction change
110	100	6	4	Count down
101	100	5	4	Count up
other than (110, 101)	100	other than (6, 5)	4	Invalid state change, Ignore, No count, no direction change
110	110	6	6	No count, no direction change
010	110	2	6	Count down
100	110	4	6	Count up
other than (010, 100)	110	other than (2, 4)	6	Invalid state change, Ignore, No count, no direction change
010	010	2	2	No count, no direction change
011	010	3	2	Count down
110	010	6	2	Count up
other than (011, 110)	010	other than (3, 6)	2	Invalid state change, Ignore, No count, no direction change
011	011	3	3	No count, no direction change
001	011	1	3	Count down
010	011	2	3	Count up

Table 20-3. Truth Table for Hall Decoder[(1), (2), (3), (4)] (continued)

Current Hall State (CBA)	Previous Hall State (CBA)	Current Sector	Previous Sector	Action
other than (001, 010)	011	other than (1, 2)	3	Invalid state change, Ignore, No count, no direction change
001	001	1	1	No count, no direction change
101	001	5	1	Count down
011	001	3	1	Count up
other than (101,011)	001	other than (5, 3)	1	Invalid state change, Ignore, No count, no direction change
101	101	5	5	No count, no direction change
100	101	4	5	Count down
001	101	1	5	Count up
other than (100, 001)	101	other than (4, 1)	5	Invalid state change, Ignore, No count, no direction change

Notes:
1. Position, Velocity and Index Counter counter are not updated for the current Invalid state.
2. Position, Velocity and Index Counter counter are updated for the current valid state when the previous state is valid.
3. Position, Velocity and Index Counter counter are not updated of state error, transition error.
4. Position, Velocity and Index Counter/Timer direction overridden by QEICON.CNTPOL

20.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1A00	QEI1CON	31:24	HALLEN	HPDLYEN	HMATCHEN	HMATCH[28:26]			ECAPPOL	HLDRD
		23:16	HLDWR	HCRE	ECAPEN[21:16]					
		15:8	QEIEN		QEISIDL	PIMOD[2:0]			IMV[1:0]	
		7:0		INTDIV[2:0]			CNTPOL	GATEN	CCM[1:0]	
0x1A04	QEI1IOC	31:24	HOMOVREN	INDXOVREN	QEBOVREN	QEAOVREN	HOMOVR	INDXOVR	QEBOVR	QEAOVR
		23:16								HCAPEN
		15:8	QCAPEN	FLTREN	QFDIV[2:0]			OUTFNC[1:0]		SWPAB
		7:0	HOMPOL	IDXPOL	QEBPOL	QEAPOL	HOME	INDEX	QEB	QEA
0x1A08	QEI1STAT	31:24					ECAPIRQ	ECAPIEN	HPLDYIRQ	HPLDYIEN
		23:16	HSTAMIRQ	HSTAMIEN	HSTACIRQ	HSTACIEN	HSERRIRQ	HSERRIEN	TERRIRQ	TERRIEN
		15:8	DIRIRQ	DIRIEN	PCHEQIRQ	PCHEQIEN	PCLEQIRQ	PCLEQIEN	POSOVIRQ	POSOVIEN
		7:0	PCIIRQ	PCIIEN	VELOVIRQ	VELOVIEN	HOMIRQ	HOMIEN	IDXIRQ	IDXIEN
0x1A0C	POS1CNT	31:24	POSCNT[31:24]							
		23:16	POSCNT[23:16]							
		15:8	POSCNT[15:8]							
		7:0	POSCNT[7:0]							
0x1A10	POS1HLD	31:24	POSHLD[31:24]							
		23:16	POSHLD[23:16]							
		15:8	POSHLD[15:8]							
		7:0	POSHLD[7:0]							
0x1A14	VEL1CNT	31:24	VELCNT[31:24]							
		23:16	VELCNT[23:16]							
		15:8	VELCNT[15:8]							
		7:0	VELCNT[7:0]							
0x1A18	VEL1HLD	31:24	VELHLD[31:24]							
		23:16	VELHLD[23:16]							
		15:8	VELHLD[15:8]							
		7:0	VELHLD[7:0]							
0x1A1C	INT1TMR	31:24	INTTMR[31:24]							
		23:16	INTTMR[23:16]							
		15:8	INTTMR[15:8]							
		7:0	INTTMR[7:0]							
0x1A20	INT1HLD	31:24	INTHLD[31:24]							
		23:16	INTHLD[23:16]							
		15:8	INTHLD[15:8]							
		7:0	INTHLD[7:0]							
0x1A24	INDX1CNT	31:24	INDXCNT[31:24]							
		23:16	INDXCNT[23:16]							
		15:8	INDXCNT[15:8]							
		7:0	INDXCNT[7:0]							
0x1A28	INDX1HLD	31:24	INDXHLD[31:24]							
		23:16	INDXHLD[23:16]							
		15:8	INDXHLD[15:8]							
		7:0	INDXHLD[7:0]							
0x1A2C	QEI1CAP	31:24	QEICAP[31:0]							
		23:16	QEICAP[31:0]							
		15:8	QEICAP[31:0]							
		7:0	QEICAP[31:0]							
0x1A30	QEI1LEC	31:24	QEILEC[31:24]							
		23:16	QEILEC[23:16]							
		15:8	QEILEC[15:8]							
		7:0	QEILEC[7:0]							
0x1A34	QEI1GEC	31:24	QEIGEC[31:0]							
		23:16	QEIGEC[31:0]							
		15:8	QEIGEC[31:0]							
		7:0	QEIGEC[31:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1A38	QEI1INIT	31:24	QEIINIT[31:0]							
		23:16	QEIINIT[31:0]							
		15:8	QEIINIT[31:0]							
		7:0	QEIINIT[31:0]							
0x1A3C	QEI1HPDLY	31:24	QEIDELAY[31:24]							
		23:16	QEIDELAY[23:16]							
		15:8	QEIDELAY[15:8]							
		7:0	QEIDELAY[7:0]							

20.3.1. QEI 1 Control Register

Name: QEI1CON
Offset: 0x1A00

Notes:
1. When CCMx = `10` or CCMx = `11`, all of the QEI counters operate as timers and the PIMOD[2:0] bits are ignored.
2. When CCMx = `00`, and QEAx and QEBx values match the Index Match Value (IMV), the POSxCNT registers are reset.
3. The selected clock rate should be at least twice the expected maximum quadrature count rate.
4. The index match value applies to the A&B inputs after the SWAP, and polarity bits have been applied.
5. The QCAPEN and HCAPEN bits must be cleared during PIMODx modes two through seven to ensure proper functionality.
6. This bit operational only for CCM[1:0] mode “`00`”.
7. A zero value in HMATCH[2:0] will not start the delay counter operation.
8. Polarity bit applied for External Trigger input.
9. QEI coherent capture event cannot trigger itself.
10. Hall sensor redefines the operation of all of the QEI inputs; the home pin functions are not applicable for Hall sensor. The INDEX pin is used as QEC input for Hall state derivation.
11. For Hall Sensor mode, it is not recommended to use any of the position counter modes that utilize index signals. Only PIMOD[2:0] = "`000`" or "`110`" are recommended; mode "`101`" may be used, but in general this mode is not relevant to Hall sensor applications.
12. By default, QEI operates in quadrature count mode when HALLEN = `0`

Bit	31	30	29	28	27	26	25	24
	HALLEN	HPDLYEN	HMATCHEN	HMATCH[28:26]			ECAPPOL	HLDRD
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HLDWR	HCRE	ECAPEN[21:16]					
Access	R/W	R/W	R/W	R/W	R/W			
Reset	0	1	0	0	0			

Bit	15	14	13	12	11	10	9	8
	QEIEN		QEISIDL	PIMOD[2:0]			IMV[1:0]	
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		INTDIV[2:0]			CNTPOL	GATEN	CCM[1:0]	
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 31 – HALLEN Hall Sensor Mode Enable bit(6)

Value	Description
1	Hall mode is enabled.

Value	Description
0	QEI mode is enabled.

Bit 30 – HPDLYEN Hall Programmable Delay Function Enable bit[(7)]

Value	Description
1	Programmable event generation delay after a Hall transition is enabled.
0	Programmable event generation delay after a Hall transition is disabled.

Bit 29 – HMATCHEN Hall Filter Match Function Enable bit

Value	Description
1	Hall Filter match state function is enabled.
0	Hall Filter match state function is disabled.

Bits 28:26 – HMATCH[28:26] Hall Filter Match bits
The HMATCH register bits are used to generate an interrupt if the filter output state matches the HMATCH[2:0] value.

Bit 25 – ECAPPOL Position Counter Capture Polarity Select bit[(8)]

Value	Description
1	Input is inverted.
0	Input is not inverted.

Bit 24 – HLDRD Hold READ Coherent Transfer Select bit

Value	Description
1	Triggers a coherent transfer from POSCNT to POSHLD and IDXCNT to IDXHLD.
0	Does not trigger coherent transfer from POSCNT to POSHLD and IDXCNT to IDXHLD.

Bit 23 – HLDWR Hold Write Coherent Transfer Select bit

Value	Description
1	Triggers a coherent transfer from POSHLD to POSCNT and IDXHLD to IDXCNT.
0	Does not trigger a coherent transfer from POSHLD to POSCNT and IDXHLD to IDXCNT.

Bit 22 – HCRE Hardware Coherent Read Transfer Enable bit

Value	Description
1	Hardware Coherent read transfer is enabled when External Capture Rise Interrupt occurs.
0	Hardware Coherent read transfer is disabled.

Bits 21:19 – ECAPEN[21:16] External Trigger Encoder Count Registers Capture Select bits

Value	Description
11111	ADTRG31(PPS)
11110	ADTRG30(PPS)
11100	ADC2 End of conversion
11011	ADC1 End of conversion
11010	QEI2 coherent read capture event[(9)]
11001	QEI1 coherent read capture event[(9)]
11000	CLC2 Output
10111	CLC1 Output
10110	MCCP9 trigger output
10101	SCCP4 trigger output
10100	SCCP3 trigger output
10011	SCCP2 trigger output

Value	Description
10010	SCCP1 trigger output
10001	PWM8 ADC trigger 1
10000	PWM8 ADC trigger 0
01111	PWM7 ADC trigger 1
01110	PWM7 ADC trigger 0
01101	PWM6 ADC trigger 1
01100	PWM6 ADC trigger 0
01011	PWM5 ADC trigger 1
01010	PWM5 ADC trigger 0
01001	PWM4 ADC trigger 1
01000	PWM4 ADC trigger 0
00111	Reserved
00110	PWM3 ADC trigger 1
00101	PWM3 ADC trigger 0
00100	PWM2 ADC trigger 1
00011	PWM2 ADC trigger 0
00010	PWM1 ADC trigger 1
00001	PWM1 ADC trigger 0
00000	No Counter capture function with external trigger input

Bit 15 – QEIEN Quadrature Encoder Interface Module Counter Enable bit

Value	Description
1	Module counters are enabled.
0	Module counters are disabled, but SFRs can be read or written.

Bit 13 – QEISIDL QEI Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode.
0	Continues module operation in Idle mode.

Bits 12:10 – PIMOD[2:0] Position Counter Initialization Mode Select bits(1,5,10,11)

Value	Description
111	Modulo Count mode for position counter and every index event loads the position counter with QEIxLEC register.
110	Modulo Count mode for position counter.
101	Resets the position counter when the position counter equals the QEIxGEC register.
100	Second index event after home event initializes the position counter with the contents of the QEIxINIT register.
011	First index event after home event initializes the position counter with the contents of the QEIxINIT register.
010	Next index input event initializes the position counter with the contents of the QEIxINIT register.
001	Every index input event resets the position counter.
000	Index input event does not affect the position counter.

Bits 9:8 – IMV[1:0] Index Match Value bits(2,4)

Value	Description
11	Index match occurs when QEBx = 1 and QEAx = 1.
10	Index match occurs when QEBx = 1 and QEAx = 0.
01	Index match occurs when QEBx = 0 and QEAx = 1.
00	Index match occurs when QEBx = 0 and QEAx = 0.

Bits 6:4 – INTDIV[2:0] Timer Input Clock Prescale Select bits(3)

(Interval timer, main timer (position counter), velocity counter and index counter internal clock divider select)

Value	Description
111	1:128 prescale value
110	1:64 prescale value
101	1:32 prescale value
100	1:16 prescale value
011	1:8 prescale value
010	1:4 prescale value
001	1:2 prescale value
000	1:1 prescale value

Bit 3 – CNTPOL Position and Index Counter/Timer Direction Select bit

Value	Description
1	Counter direction is negative unless modified by an external up/down signal.
0	Counter direction is positive unless modified by an external up/down signal.

Bit 2 – GATEN External Count Gate Enable bit

Value	Description
1	External gate signal controls position counter operation.
0	External gate signal does not affect position counter operation.

Bits 1:0 – CCM[1:0] Counter Control Mode Selection bits(12)

Value	Description
11	Internal Timer mode
10	External Clock Count with External Gate mode
01	External Clock Count with External Up/Down mode
00	Quadrature/Hall Sensor Count mode

20.3.2. QEI I/O 1 Control Register

Name: QEI1IOC
Offset: 0x1A04

Note:
1. In Hall mode, the INDEX is also used as QEC.

Legend: x = Bit is unknown

Bit	31	30	29	28	27	26	25	24
	HOMOVREN	INDXOVREN	QEBOVREN	QEAOVREN	HOMOVR	INDXOVR	QEBOVR	QEAOVR
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
								HCAPEN
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
	QCAPEN	FLTREN	QFDIV[2:0]			OUTFNC[1:0]		SWPAB
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HOMPOL	IDXPOL	QEBPOL	QEAPOL	HOME	INDEX	QEB	QEA
Access	R/W	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	x	x	x	x

Bit 31 – HOMOVREN Home Input Override Enable bit

Value	Description
1	Home input is provided by HOMEOVR bit .
0	Home input is provided by input to the module.

Bit 30 – INDXOVREN Index Input Override Enable bit

Value	Description
1	Index input is provided by INDXOVR bit .
0	Index input is provided by input to the module.

Bit 29 – QEBOVREN QEB Input Override Enable bit

Value	Description
1	QEB input is provided by QEBOVR bit.
0	QEB input is provided by input to the module.

Bit 28 – QEAOVREN QEA Input Override Enable bit

Value	Description
1	QEA input is provided by QEAOVR bit.
0	QEA input is provided by input to the module.

Bit 27 – HOMOVR Home Input Override bit

Value	Description
1	Home input override with value "1".
0	Home input override with value "0".

Bit 26 – INDXOVR Index Input Override bit

Value	Description
1	Index input override with value "1".
0	Index input override with value "0".

Bit 25 – QEBOVR QEB Input Override bit

Value	Description
1	QEB input override with value "1".
0	QEB input override with value "0".

Bit 24 – QEAOVR QEB Input Override bit

Value	Description
1	QEA input override with value "1".
0	QEA input override with value "0".

Bit 16 – HCAPEN Position Counter Input Capture by Home Event Enable bit

Value	Description
1	HOMEx input event (positive edge) triggers a position capture event (QCAPEN must be cleared).
0	HOMEx input event (positive edge) does not trigger a position capture event.

Bit 15 – QCAPEN QEI Position Counter Input Capture by Index Match Event Enable bit

Value	Description
1	Index match event (positive edge) triggers a position capture event (HCAPEN must be cleared).
0	Index match event (positive edge) does not trigger a position capture event.

Bit 14 – FLTREN QEAx/QEBx/INDXx/HOMEx Digital Filter Enable bit

Value	Description
1	Input pin digital filter is enabled.
0	Input pin digital filter is disabled (bypassed).

Bits 13:11 – QFDIV[2:0] QEAx/QEBx/INDXx/HOMEx Digital Input Filter Clock Divide Select bits

Value	Description
111	1:128 clock divide
110	1:64 clock divide
101	1:32 clock divide
100	1:16 clock divide
011	1:8 clock divide
010	1:4 clock divide
001	1:2 clock divide
000	1:1 clock divide

Bits 10:9 – OUTFNC[1:0] QEI Module Output Function Mode Select bits

Value	Description
11	The CNTCMPx pin goes high when POSxCNT ≤ QEIxLEC or POSxCNT ≥ QEIxGEC.
10	The CNTCMPx pin goes high when POSxCNT ≤ QEIxLEC.
01	The CNTCMPx pin goes high when POSxCNT ≥ QEIxGEC.

Value	Description
00	Output is disabled.

Bit 8 – SWPAB Swap QEAx and QEBx Inputs bit

Value	Description
1	QEAx and QEBx are swapped prior to Quadrature Decoder logic.
0	QEAx and QEBx are not swapped.

Bit 7 – HOMPOL HOMEx Input Polarity Select bit

Value	Description
1	Input is inverted.
0	Input is not inverted.

Bit 6 – IDXPOL INDXx Input Polarity Select bit

Value	Description
1	Input is inverted.
0	Input is not inverted.

Bit 5 – QEBPOL QEBx Input Polarity Select bit

Value	Description
1	Input is inverted.
0	Input is not inverted.

Bit 4 – QEAPOL QEAx Input Polarity Select bit

Value	Description
1	Input is inverted.
0	Input is not inverted.

Bit 3 – HOME Status of HOMEx Input Pin After Polarity Control bit (read-only)

Value	Description
1	Pin is at logic '1' if HOMPOL bit is set to '0'; Pin is at logic '0' if HOMPOL bit is set to '1'.
0	Pin is at logic '0' if HOMPOL bit is set to '0'; Pin is at logic '1' if HOMPOL bit is set to '1'.

Bit 2 – INDEX Status of INDXx Input Pin After Polarity Control bit (read-only)(1)

Value	Description
1	Pin is at logic '1' if the IDXPOL bit is set to '0'; Pin is at logic '0' if the IDXPOL bit is set to '1'.
0	Pin is at logic '0' if the IDXPOL bit is set to '0'; Pin is at logic '1' if the IDXPOL bit is set to '1'.

Bit 1 – QEB Status of QEBx Input Pin After Polarity Control and SWPAB Pin Swapping bit (read-only)

Value	Description
1	Physical pin, QEBx, is at logic '1' if QEBPOL bit is set to '0' and SWPAB bit is set to '0'; physical pin, QEBx, is at logic '0' if QEBPOL bit is set to '1' and SWPAB bit is set to '0'; physical pin, QEAx, is at logic '1' if QEBPOL bit is set to '0' and SWPAB bit is set to '1'; physical pin, QEAx, is at logic '0' if QEBPOL bit is set to '1' and SWPAB bit is set to '1'.

Value	Description
0	Physical pin, QEBx, is at logic ‘0’ if QEBPOL bit is set to ‘0’ and SWPAB bit is set to ‘0’; physical pin, QEBx, is at logic ‘1’ if QEBPOL bit is set to ‘1’ and SWPAB bit is set to ‘0’; physical pin, QEAx, is at logic ‘0’ if QEBPOL bit is set to ‘0’ and SWPAB bit is set to ‘1’; physical pin, QEAx, is at logic ‘1’ if QEBPOL bit is set to ‘1’ and SWPAB bit is set to ‘1’.

Bit 0 – QEA Status of QEAx Input Pin After Polarity Control and SWPAB Pin Swapping bit (read-only)

Value	Description
1	Physical pin, QEAx, is at logic ‘1’ if QEAPOL bit is set to ‘0’ and SWPAB bit is set to ‘0’; physical pin, QEAx, is at logic ‘0’ if QEAPOL bit is set to ‘1’ and SWPAB bit is set to ‘0’; physical pin, QEBx, is at logic ‘1’ if QEAPOL bit is set to ‘0’ and SWPAB bit is set to ‘1’; physical pin, QEBx, is at logic ‘0’ if QEAPOL bit is set to ‘1’ and SWPAB bit is set to ‘1’.
0	Physical pin, QEAx, is at logic ‘0’ if QEAPOL bit is set to ‘0’ and SWPAB bit is set to ‘0’; physical pin, QEAx, is at logic ‘1’ if QEAPOL bit is set to ‘1’ and SWPAB bit is set to ‘0’; physical pin, QEBx, is at logic ‘0’ if QEAPOL bit is set to ‘0’ and SWPAB bit is set to ‘1’; physical pin, QEBx, is at logic ‘1’ if QEAPOL bit is set to ‘1’ and SWPAB bit is set to ‘1’.

20.3.3. QEI 1 Status Register

Name: QEI1STAT
Offset: 0x1A08

Notes:
1. This status bit is only applicable to PIMOD[2:0] modes, '011' and '100'.
2. Outside of Hall mode, this bit is always read '0'.
3. This bit can be used in both Hall mode and Encoder mode.
4. This bit operational only for Hall mode.

Legend: C = Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
					ECAPIRQ	ECAPIEN	HPLDYIRQ	HPLDYIEN
Access					R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
	HSTAMIRQ	HSTAMIEN	HSTACIRQ	HSTACIEN	HSERRIRQ	HSERRIEN	TERRIRQ	TERRIEN
Access	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DIRIRQ	DIRIEN	PCHEQIRQ	PCHEQIEN	PCLEQIRQ	PCLEQIEN	POSOVIRQ	POSOVIEN
Access	R/W/HS	R/W/HS	R/W/HS	R/W	R/W/HS	R/W	R/W/HS	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PCIIRQ	PCIIEN	VELOVIRQ	VELOVIEN	HOMIRQ	HOMIEN	IDXIRQ	IDXIEN
Access	R/W/HS	R/W	R/W/HS	R/W	R/W/HS	R/W	R/W/HS	R/W
Reset	0	0	0	0	0	0	0	0

Bit 27 – ECAPIRQ External Capture Rise Interrupt Status bit(3)

Value	Description
1	External Capture Rise Interrupt was generated.
0	No External Capture Rise Interrupt was generated.

Bit 26 – ECAPIEN External Capture Rise Interrupt Enable bit(3)

Value	Description
1	External Capture Rise interrupt is enabled.
0	External Capture Rise interrupt is disabled.

Bit 25 – HPLDYIRQ Hall Programmable Delay Interrupt Status bit(2)

Value	Description
1	Hall programmable delay interrupt was generated.
0	Hall programmable delay interrupt was not generated.

Bit 24 – HPLDYIEN Hall Programmable Delay Interrupt Enable bit(2)

Value	Description
1	Hall programmable delay interrupt is enabled.

Value	Description
0	Hall programmable delay interrupt is disabled.

Bit 23 – HSTAMIRQ Hall state match interrupt Status bit(2)

Value	Description
1	Hall state match interrupt was generated.
0	Hall state match interrupt was not generated.

Bit 22 – HSTAMIEN Hall State Match Interrupt Enable bit

Value	Description
1	Hall state match interrupt is enabled.
0	Hall state match interrupt is disabled.

Bit 21 – HSTACIRQ Hall State Change Interrupt Status bit(2)

Value	Description
1	Hall state change interrupt was generated.
0	Hall state change interrupt was not generated.

Bit 20 – HSTACIEN Hall State Change Interrupt Enable bit(4)

Value	Description
1	Hall state change interrupt is enabled.
0	Hall state change interrupt is disabled.

Bit 19 – HSERRIRQ Hall State Error interrupt Status bit(2)

Value	Description
1	Hall State Error interrupt was generated.
0	Hall State Error interrupt was not generated.

Bit 18 – HSERRIEN Hall State Error Interrupt Enable bit(4)

Value	Description
1	Hall State Error Interrupt is enabled.
0	Hall State Error Interrupt is disabled.

Bit 17 – TERRIRQ Quadrature/Hall Transition Error interrupt Status bit(4)

Value	Description
1	Quadrature/Hall Transition Error interrupt was generated.
0	Quadrature/Hall Transition Error interrupt was not generated.

Bit 16 – TERRIEN Quadrature/Hall Transition Error Interrupt Enable bit(4)

Value	Description
1	Quadrature/Hall Transition Error interrupt is enabled.
0	Quadrature/Hall Transition Error interrupt is disabled.

Bit 15 – DIRIRQ Direction Change Interrupt Status bit

Value	Description
1	Direction change interrupt was generated.
0	Direction change interrupt was not generated.

Bit 14 – DIRIEN Direction change Interrupt Enable bit

Value	Description
1	Direction change Interrupt is enabled.
0	Direction change Interrupt is disabled.

Bit 13 – PCHEQIRQ Position Counter Greater Than Compare Status bit

Value	Description
1	POSxCNT > QEIxGEC
0	POSxCNT < QEIxGEC

Bit 12 – PCHEQIEN Position Counter Greater Than Compare Interrupt Enable bit

Value	Description
1	Position Counter Greater Than Compare Interrupt is enabled.
0	Position Counter Greater Than Compare Interrupt is disabled.

Bit 11 – PCLEQIRQ Position Counter Less Than Compare Status bit

Value	Description
1	POSxCNT < QEIxLEC
0	POSxCNT > QEIxLEC

Bit 10 – PCLEQIEN Position Counter Less Than Compare Interrupt Enable bit

Value	Description
1	Position Counter Less Than Compare Interrupt is enabled.
0	Position Counter Less Than Compare Interrupt is disabled.

Bit 9 – POSOVIRQ Position Counter Overflow Status bit

Value	Description
1	Position Counter Overflow has occurred.
0	Position Counter Overflow has not occurred.

Bit 8 – POSOVIEN Position Counter Overflow Interrupt Enable bit

Value	Description
1	Position Counter Overflow Interrupt is enabled.
0	Position Counter Overflow Interrupt is disabled.

Bit 7 – PCIIRQ Position Counter (Homing) Initialization Process Complete Status bit[(1)]

Value	Description
1	POSxCNT was reinitialized.
0	POSxCNT was not reinitialized.

Bit 6 – PCIIEN Position Counter (Homing) Initialization Process Complete Interrupt Enable bit

Value	Description
1	Position Counter (Homing) Initialization Process Complete Interrupt is enabled.
0	Position Counter (Homing) Initialization Process Complete Interrupt is disabled.

Bit 5 – VELOVIRQ Velocity Counter Overflow Status bit

Value	Description
1	Velocity Counter Overflow has occurred.
0	Velocity Counter Overflow has not occurred.

Bit 4 – VELOVIEN Velocity Counter Overflow Interrupt Enable bit

Value	Description
1	Velocity Counter Overflow Interrupt is enabled.
0	Velocity Counter Overflow Interrupt is disabled.

Bit 3 – HOMIRQ Home Event Status bit

Value	Description
1	Home event has occurred.
0	No home event has occurred.

Bit 2 – HOMIEN Home Input Event Interrupt Enable bit

Value	Description
1	Home Input Event Interrupt is enabled.
0	Home Input Event Interrupt is disabled.

Bit 1 – IDXIRQ Index Event Status bit

Value	Description
1	Index event has occurred.
0	No index event has occurred.

Bit 0 – IDXIEN Index Input Event Interrupt Enable bit

Value	Description
1	Index Input Event Interrupt is enabled.
0	Index Input Event Interrupt is disabled.

20.3.4. Position 1 Counter Register

Name: POS1CNT
Offset: 0x1A0C

Bit	31	30	29	28	27	26	25	24
	POSCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	POSCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	POSCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	POSCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – POSCNT[31:0] Position Counter Register bits

20.3.5. Position Counter 1 Hold Register

Name: POS1HLD
Offset: 0x1A10

Bit	31	30	29	28	27	26	25	24
	POSHLD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	POSHLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	POSHLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	POSHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – POSHLD[31:0] Position Counter Hold Register bits

20.3.6. Velocity 1 Counter Register

Name: VEL1CNT
Offset: 0x1A14

Bit	31	30	29	28	27	26	25	24
	VELCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	VELCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VELCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VELCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – VELCNT[31:0] Velocity Counter bits

20.3.7. Position Counter 1 Hold Register

Name: VEL1HLD
Offset: 0x1A18

Bit	31	30	29	28	27	26	25	24
	VELHLD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	VELHLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	VELHLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	VELHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – VELHLD[31:0] Velocity Counter Hold Register bits

20.3.8. Interval 1 Timer Register

Name: INT1TMR
Offset: 0x1A1C

Bit	31	30	29	28	27	26	25	24
	INTTMR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INTTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INTTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INTTMR[31:0] Interval Timer Register bits

20.3.9. Interval Timer Hold Register

Name: INT1HLD
Offset: 0x1A20

Bit	31	30	29	28	27	26	25	24
	INTHLD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INTHLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INTHLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INTHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INTHLD[31:0] Interval Timer Hold Register (INTxHLD) bits

20.3.10. Index 1 Counter Register

Name: INDX1CNT
Offset: 0x1A24

Bit	31	30	29	28	27	26	25	24
	INDXCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INDXCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INDXCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INDXCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INDXCNT[31:0] Index Counter Value bits

20.3.11. Index Counter 1 Hold Register

Name: INDX1HLD
Offset: 0x1A28

Bit	31	30	29	28	27	26	25	24
	INDXHLD[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	INDXHLD[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INDXHLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	INDXHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – INDXHLD[31:0] Index Counter Hold Register bits

20.3.12. QEI 1 Capture Register

Name: QEI1CAP
Offset: 0x1A2C

Bit	31	30	29	28	27	26	25	24
	QEICAP[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	QEICAP[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEICAP[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEICAP[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – QEICAP[31:0] Captured Position Counter Value bits

20.3.13. QEI 1 Less Than or Equal Compare Register

Name: QEI1LEC
Offset: 0x1A30

Bit	31	30	29	28	27	26	25	24
	QEILEC[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	QEILEC[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEILEC[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEILEC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – QEILEC[31:0] Lesser Than or Equal Compare bits

20.3.14. QEI 1 Greater Than or Equal Compare Register

Name: QEI1GEC
Offset: 0x1A34

Bit	31	30	29	28	27	26	25	24
	QEIGEC[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	QEIGEC[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEIGEC[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEIGEC[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – QEIGEC[31:0] Greater Than or Equal Compare bits

20.3.15. QEI 1 Initialization Register

Name: QEI1INIT
Offset: 0x1A38

Bit	31	30	29	28	27	26	25	24
	QEIINIT[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	QEIINIT[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEIINIT[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEIINIT[31:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – QEIINIT[31:0] Initialization bits

20.3.16. QEI 1 Programmable Delay Compare Register

Name: QEI1HPDLY
Offset: 0x1A3C

Bit	31	30	29	28	27	26	25	24
	QEIDELAY[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	QEIDELAY[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	QEIDELAY[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	QEIDELAY[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – QEIDELAY[31:0] Programmable Delay Compare bits

20.4. Operation

20.4.1. Position Counter

The Position Counter is 32 bits wide and counts the number of pulses generated by an encoder.

If the POSOVIEN bit in the QEIx Status register (QEIxSTAT[8]) is set and the Position Counter rolls over from 0x7FFFFFFF to 0x80000000, or from 0x80000000 to 0x7FFFFFFF, an interrupt will be generated.

The operating mode of the Position Counter is controlled by the CCM[1:0] bits in the QEIx Control register (QEIxCON[1:0]). The Position Counter supports the following operating modes:

- Quadrature Count mode
- External Count with External Up/Down mode
- External Count with External Gate mode
- Internal Timer mode

20.4.2. Quadrature Count Mode

Quadrature count mode is selected by the CCM[1:0] = `0` and HALLEN = `0` in the QEIx Control register (QEIxCON[1:0]). In this mode, the QEA/EXTCNT and QEB/DIR/GATE inputs are decoded to generate count pulses and direction information to control the POSxCNT and VELxCNT registers.

The INDXxCNT register counts when a valid edge is detected on the INDXx input. Figure 20-1 illustrates the timing diagram of the Quadrature Count mode operation.

.

20.4.3. Hall Count Mode

Hall count mode is selected by the CCM[1:0] = `0` and HALLEN = `1` in the QEIx Control register (QEIxCON[1:0]). In this mode, the QEA, QEB and INDEX/QEC inputs are decoded to generate count pulses and direction information to control the POSxCNT and VELxCNT registers. Figure 20-4 illustrates the timing diagram of the Hall count mode operation.

20.4.3.1. Hall Mode Operation

The Hall Operating mode is selected by writing HALLEN= `1` and CCM[1:0] = `00` in the QEIxCON register.

In Hall Operation mode, control logic QEA, QEB and QEC inputs represent the phase A, B and C of a Hall sensor, respectively. The three of Hall inputs (QEC, QEB, QEA) define the current Hall state. The six possible states are 3'b001, 3'b011, 3'b010, 3'b110, 3'b100 and 3'b101. By comparing the current state of the inputs with the previous state of the inputs, the direction and count status can be decoded.

Negative Rotation (Default): CBA=100 → 110 - 010 - 011- 001 - 101

Positive Rotation (Default): CBA=100 - 101 - 001 - 011- 010 -110

Table 20-3 shows the truth table that describes how the Hall sensor signals are decoded. Count pulses are generated when the Hall state changes. The Count direction is determined by the direction of the state changes.

The Hall filter checks for the valid Hall value. If an invalid Hall code is detected (`000` or `111`), the Hall state error interrupt flag, HSERRIRQ, will be set in the QEIx Status register (QEIxSTAT[19]). If HSERRIEN bit in the QEIx Status register is set (QEIxSTAT[18]), then an interrupt will be generated.

The Hall filter state match function can be enabled using HMATCHEN bit in the QEIxCON register. The user can configure the required Hall filter state match using the HMATCH[2:0] bits in the QEIxCON register. When the Hall filter state matches the HMATCH bits, the error interrupt flag, HSTAMIRQ, will be set in the QEIx Status register (QEIxSTAT[23]). If HSTAMIEN bit in QEIx status register is set (QEIxSTAT[22]), an interrupt will be generated.

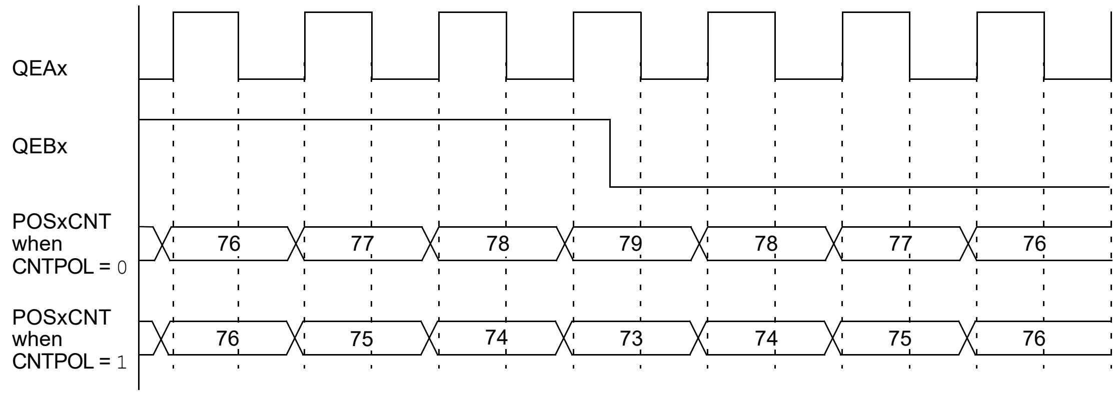
Optionally, the programmable delay counter can be used to generate an interrupt after a programmable delay time. The delay time must be written in the Programmable Delay Compare register (QEIxHPDLY). The delay counter starts counting when the HMATCH[2:0] matches the Hall input code, and only if the respective HMATCH[2:0] value is different from zero. The HPDLYIRQ Interrupt Flag is generated when the delay counter reaches the QEIxHPDLY value. If HPDLYIEN bit in QEIx Status register is set (QEIxSTAT[24]), an interrupt will be generated.

The Position Counter (POSxCNT) starts incrementing/decrementing when a new Hall state transition is detected. The value of the counter is constantly monitored to detect error conditions. The window monitor lower threshold is defined by the Compare Low register (QEIxLEC), and the upper threshold value is defined by the Compare High register (QEIxGEC). If a Hall transition is detected and the counter is outside the window thresholds, error interrupt flags are set. The window upper threshold error status bit, PCHEQIRQ, in the QEIx Status register (QEIxSTAT[13]) is set, if the counter value is higher than QEIxGEC. If the PCHEQIEN bit in QEIx Status register (QEIxSTAT[12]) is set, an interrupt will be generated. The window lower threshold error status bit, PCLEQIRQ, in the QEIx Status register (QEIxSTAT[11]) is set, if the counter value is lower than QEIxLEC. If the PCLEQIEN bit in the QEIx Status register (QEIxSTAT[10]) is set, then an interrupt will be generated.

20.4.4. External Count with External Up/Down Mode

In this mode, the QEAx/EXTCNT input is considered as an external count signal and the QEBx/DIR/GATE input provides the count direction information. The count direction is positive unless overridden by the CNTPOL bit in the QEIx Control register (QEIxCON[3]). Figure 20-5 illustrates the timing diagram of an external count with External Up/Down mode operation.

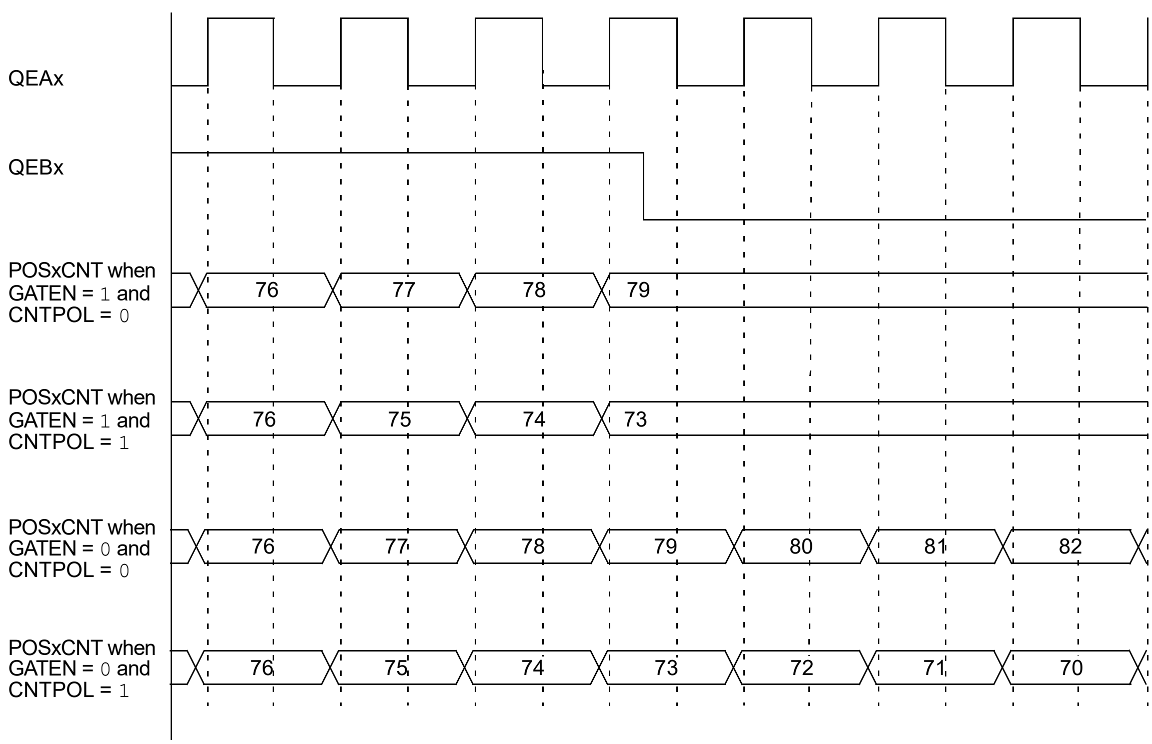
Figure 20-5. External Count with External Up/Down Mode



20.4.5. External Count with External Gate Mode

In this mode, the QEAx/EXTCNT input is considered as an external count signal. If the GATEN bit in the QEIx Control register (QEIxCON[2]) is set, and QEBx/DIR/GATE = 0, the QEBx/DIR/GATE input will inhibit the counter signal. If the GATEN bit is cleared, the gate signal does not affect the counter operation. The default count direction is positive. If the CNTPOL bit in the QEIx Control register (QEIxCON[3]) is set, the count direction is negative. Figure 20-6 illustrates the timing diagram of an external count with External Gate mode operation.

Figure 20-6. External Count with External Gate Mode

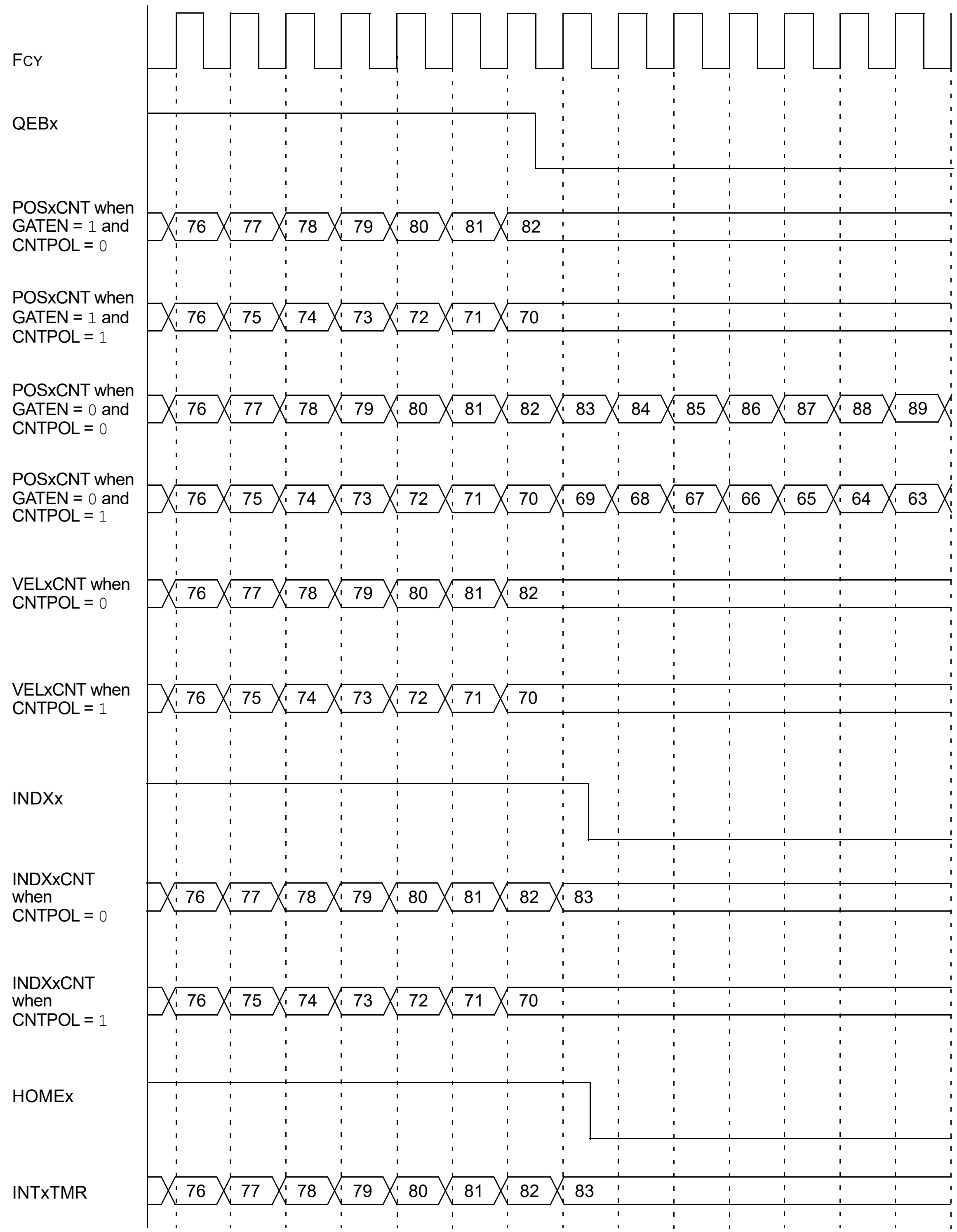


20.4.6. Internal Timer Mode

In this mode, the Position Counter, Velocity, Index and Interval Counters use an internal clock as the count source. The internal clock is divided by the clock divider using the INTDIV[2:0] bits in the QEIx Control register (QEIxCON[6:4]). If the GATEN bit in the QEIx Control register (QEIxCON[2]) is set and QEBx/DIR/GATE = 0, the QEBx/DIR/GATE input will inhibit the counter signal. If the GATEN bit is cleared, the gate signal does not affect the operation of the counter. The default count direction is positive. If the CNTPOL bit in the QEIx Control register (QEIxCON[3]) is set, the count direction is negative. Figure 20-7 illustrates the timing diagram of an Internal Timer mode operation.

The INDEX input enables and disables (gates) the counting of the Index Counter. The HOME input enables and disables (gates) the counting of the Interval Counter.

Figure 20-7. Internal Timer Mode

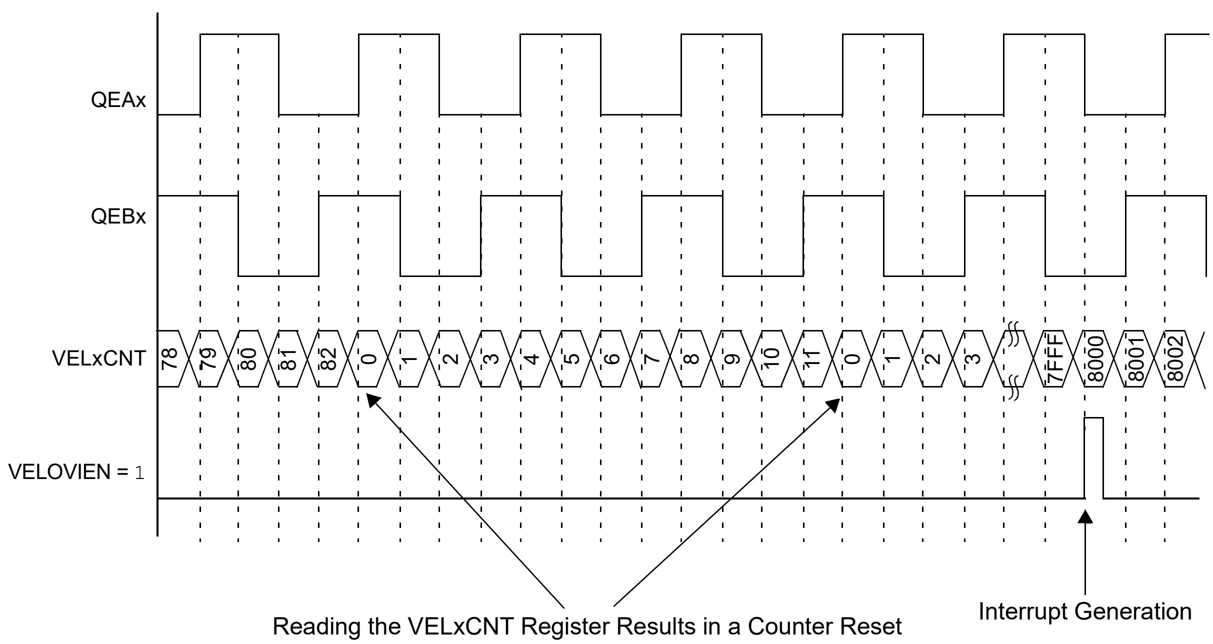


20.4.7. Velocity Counter

The Velocity Counter (VELxCNT) is a register that is up to 32 bits wide and increments or decrements based on the signal from the quadrature decoder logic. Reading this register results in a counter Reset. The Index input or any of the modes specified by the PIMOD[2:0] bits in the QEIx Control register (QEIxCON[12:10]) do not affect the operation of the Velocity Counter. If the Velocity Counter rolls over from 0x7FFF to 0x8000, or from 0x8000 to 0x7FFF, and the VELOVIEN bit in the QEIx Status register (QEIxSTAT[4]) is set, an interrupt will be generated. Figure 20-8 illustrates the timing diagram of the Velocity Counter operation.

Note: The Velocity Counter specifies the distance traveled between the time interval of each sample. Reading the VELxCNT register results in a counter Reset. The user application should read the Velocity Counter at a rate of 1-4 kHz.

Figure 20-8. Velocity Counter



20.4.8. Index Counter

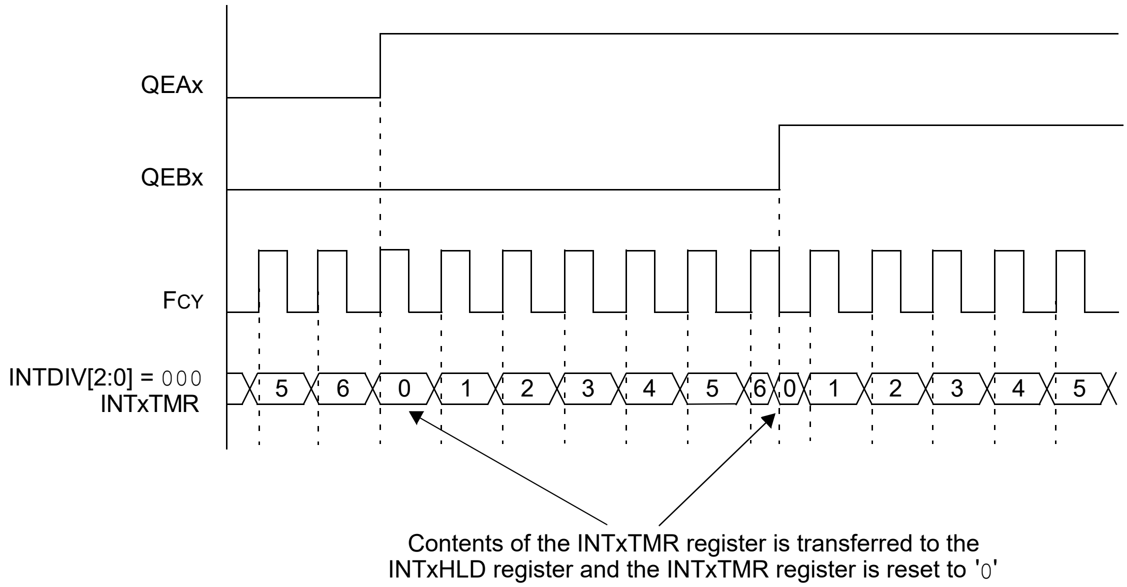
The 32-bit wide Index Counter (INDXxCNT) register counts index events and is incremented or decremented based on the direction output of the quadrature logic decoder (see Figure 20-1). For more information, refer to Index Event.

20.4.9. Interval Timer

When a motor runs at a very low speed, the encoder does not generate enough pulses for accurate speed measurement. Therefore, instead of counting the number of pulses, the pulse duration can be measured. The 32-bit Interval Timer (INTxTMR) is used to measure the time interval between each decoded quadrature count pulse when the motor operates at a very low speed. The timer counts at a rate specified by the INTDIV[2:0] bits in the QEIx Control register (QEIxCON[6:4]). The Interval Timer is cleared when the first count pulse is detected. When the next count pulse is detected, the current contents of the Interval Timer are transferred to the Interval Hold register (INTxHLD), the Interval Timer is cleared and then the process repeats. The Interval Hold registers always contain the most recent completed timing measurements. The Interval Timer is automatically cleared when the module gets disabled. Figure 20-9 illustrates the timing diagram of the Interval Timer operation.

Note: If the INTxHLD register is read when a new position count pulse is detected, the contents of the INTxHLD register are not updated to avoid incoherent data reading.

Figure 20-9. Interval Timer



20.4.10. Initialization Register

The 32-bit QEIx Initialization register (QEIxINIT) is a general purpose register that can be used to initialize the position counter.

The initialization operation is done by the PIMOD[2:0] bits of the QEIx Control register (QEIxCON[12:10]). To initialize the Position Counter mode, the contents of the QEIxINIT register are loaded into the POSxCNT register based on the condition set by the PIMOD[2:0] bits.

20.4.11. Capture Register

The 32-bit QEIx Capture register (QEIxCAP) is a general purpose register that can be used to capture the contents of the Position counter.

In Capture mode, the input signal is used to capture the contents of the Position register into the QEIxCAP register. When used for position capture, an index match event (QCAPEN = 1) or a home event (HCAPEN = 1) can cause the QEIxCAP register to take a copy of the current position counter contents.

20.4.12. Coherent Counter Access

The content of POSxCNT and INDxCNT registers can be coherently read by hardware using the ECAPEN and HCRE bits in the QEIxCON register.

If an external trigger event selected by ECAPEN bit in the QEIxCON register occurs, then the following events occur:

- POSxHLD and INDXxHLD registers will capture the contents of the POSxCNT and INDXxCNT when the external trigger event occurs and HCRE=1 (Equivalent to software writing HLDREAD = 1).
- ECAPIRQ flag in the QEIx status register (QEIxSTAT[27]) is set so that software can determine that external capture event has occurred. If the interrupt enable bit (ECAPIEN) is set, an interrupt is generated. The user can write ECAPIRQ flag = 0 to clear the flag.
- As long as the ECAPIRQ flag is set, further external capture events would be suppressed, so that software can take its time to read POSxCNT and INDXxCNT.

20.4.13. Software Coherent Read and Write

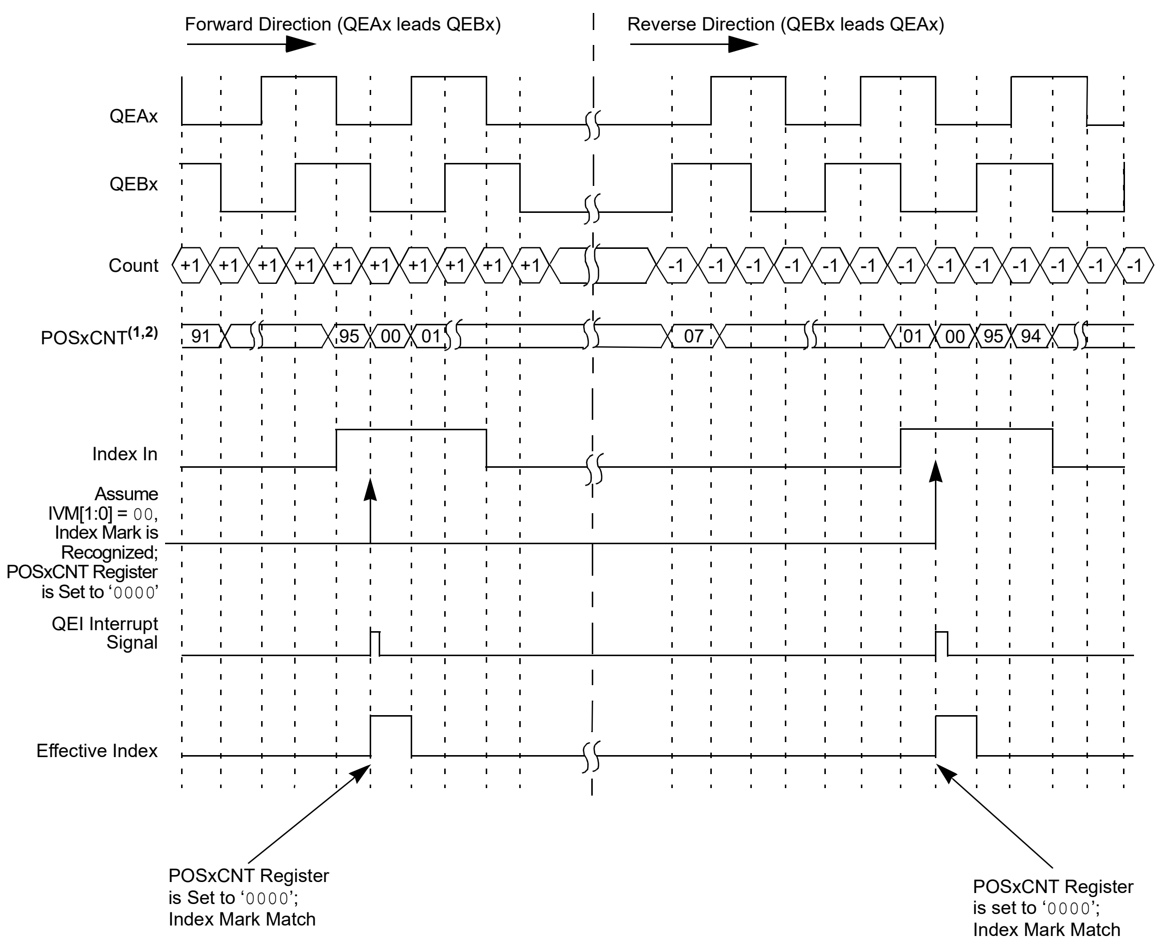
If the HLDRD bit in the QEIxCON register is set, software should be able to trigger a simultaneous transfer from POSxCNT and INDxCNT to POSxHLD and INDXxHLD registers respectively, for coherent reads.

If the HLDWR bit in the QEIxCON register is set, software should be able to trigger a simultaneous transfer from POSxHLD and INDXxHLD to POSxCNT and INDXxCNT registers respectively, for coherent writes.

20.4.14. Position Comparator

The 32-bit Compare registers (QEIxGEC and QEIxLEC) and associated comparator allow the user application to compare the contents of the position counter to a specified value. The comparator provides two outputs: greater than and less than. When a suitable condition is met, the comparator generates and sets the PCHEQIRQ or PCLEQIRQ bit in the QEIx Status register (QEIxSTAT[13] and QEIxSTAT[11], respectively). If the interrupt enable bit, PCHEQIEN or PCLEQIEN, is set, an interrupt is generated. The comparator output is available on the CNTCMPx pin. The selection of a condition is made by the OUTFNC[1:0] bits of the QEIx I/O Control register (QEIxIOC[10:9]). The comparator can also be used to reset the position counter when a match is detected. The selection is made by the PIMOD[2:0] bits of the QEIx Control register (QEIxCON[12:10]). Figure 20-10 illustrates the Index Reset Position Counter operation.

Figure 20-10. Index Reset Position Counter Operation



Notes:

1. Position count update shown is when CCM[1:0] = 00.
2. Position Counter (POSxCNT) contents are incremented and decremented at each new count state although it is not shown in this diagram.

20.4.15. Index Event

When CCM[1:0] = 00 and HALLEN=0 (Quadrature mode), the IMV[1:0] bits in the QEI Control register (QEIxCON[9:8]) specify the state of the QEAx and QEBx input signals required to acknowledge an index event. An index event is accepted when an index pulse occurs while the value of the QEAx and QEBx inputs match the condition set in the IMV[1:0] bits. This prevents further index events from being accepted until the index input signal is deasserted and ensures that only one index event occurs for each index input pulse. Figure 20-10 illustrates the Index Reset Position Counter operation.

When CCM[1:0] = 01, 10 or 11 (count mode), or 00 and HALLEN = 1 (Hall mode), the IMV[1:0] bits are not used for index matching because there is no Quadrature state in this mode.

20.4.16. Position Counter Initialization Modes

By using the PIMOD[2:0] bits in the QEIx Control register (QEIxCON[12:10]), the user application can specify how the Position Counter is initialized during the module operation.

- **Mode 0** – The Position Counter is unaffected by the index input.
- **Mode 1** – The Position Counter is cleared whenever an index input event is detected.
- **Mode 2** – The Position Counter is initialized with the contents of the QEIxINIT register on the next detected index input event. When the index event occurs, the PIMOD[2:0] bits are cleared and then the counter operates in Mode 0.
- **Mode 3** – The Position Counter is initialized with the contents of the QEIxINIT register on the next detected index input event following the assertion of the home input. When an index event occurs following the home event, the PIMOD[2:0] bits are cleared and then the counter operates in Mode 0.
- **Mode 4** – The Position Counter is initialized with the contents of the QEIxINIT register on the second detected index input event following the assertion of the home input. When the second index event occurs following the home event, the PIMOD[2:0] bits are cleared and then the counter operates in Mode 0.
- **Mode 5** – The Position Counter is cleared when the Position Counter value equals the QEIxGEC register value.
- **Mode 6** – The Position Counter is loaded with the contents of the QEIxLEC register when the Position Counter value equals the QEIxGEC register value and a count up pulse is detected. The counter is loaded with the contents of the QEIxGEC register when the Position Counter value equals the QEIxLEC register value and a count down pulse is detected.
- **Mode 7** – The Position Counter is loaded with the contents of the QEIxLEC register when the Position Counter equals the QEIxGEC register's contents and a count up pulse is detected. The Position Counter is loaded with the contents of the QEIxGEC register when the Position Counter equals the QEIxLEC register contents and a count down pulse is detected. If an index pulse is detected, the Position Counter is loaded with the content of the QEIxLEC register.

Table 20-4. PIMOD Supports in Varies Mode(2), (3)

PIMOD[2:0]	QEI Mode	Hall Mode	Comments
Mode 0	Yes	Yes(1)	An index input event does not affect position counter.
Mode 1	Yes	No	Every index input event resets the position counter.

Table 20-4. PIMOD Supports in Varies Mode(2), (3) (continued)

PIMOD[2:0]	QEI Mode	Hall Mode	Comments
Mode 2	Yes	No	Next index input event initializes the position counter with contents of INIT register.
Mode 3	Yes	No	First Index event after home event initializes position counter with contents of INIT register.
Mode 4	Yes	No	Second Index event after home event initializes position counter with contents of INIT register.
Mode 5	Yes	Yes	When the position counter reaches QEICMPH, the position counter becomes zero.
Mode 6	Yes	Yes	Modulo Count mode for position counter
Mode 7	Yes	No	Modulo Count mode for position counter and every index input event resets the position counter.

Note:

1. For CCM = 0, the index event is the Quadrature state match event generated when QEA and QEB matches IMV[1:0] bits.
2. The QEA and QEB inputs are decoded to generate count pulses for the QEI mode position counter.
3. The QEA , QEB and INDEX/QEC inputs are decoded to generate count pulses for the QEI mode position counter.

20.4.17. Digital Input Filter

The QEI module uses digital noise filters to reject noise on the incoming index and quadrature phase signals. These filters reject low-level noise and large, short duration noise spikes that typically occur in motor systems.

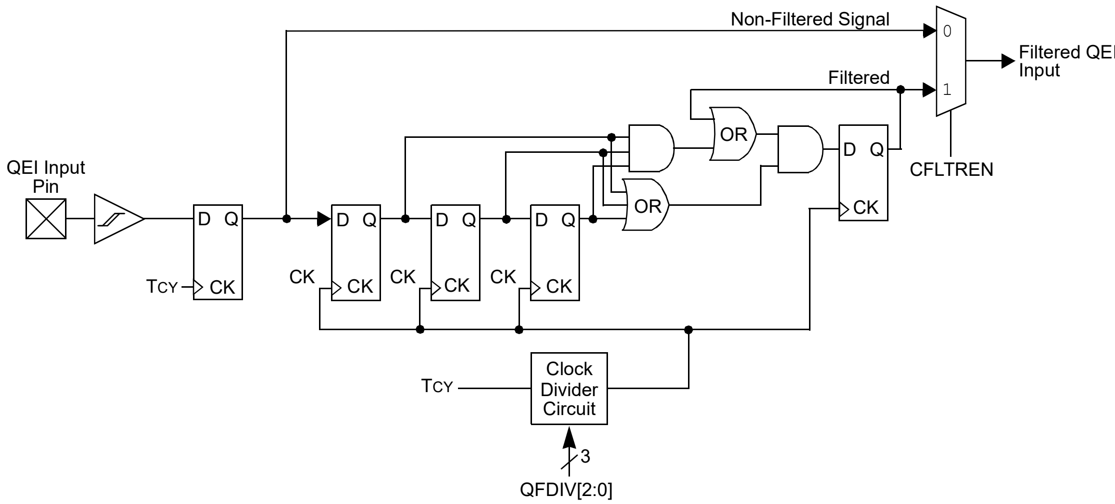
The filtered output signals can change only after an input level has the same value for three consecutive rising clock edges. The result is that short noise spikes between rising clock edges are ignored, and pulses shorter than two clock periods are rejected.

The filter clock's rate determines the low passband of the filter. A slower filter clock results in a passband rejecting lower frequencies.

The digital filter is enabled by setting the FLTREN bit in the QEIx I/O Control register (QEIxIOC[14]). The QFDIV[2:0] bits in the QEIx I/O Control register (QEIxIOC[13:11]) select the filter clock divider ratio for the clock signal.

Figure 20-11 illustrates the simplified block diagram of the digital noise filter.

Figure 20-11. Digital Noise Filter Block Diagram



20.4.18. Self Test Features

The QEIxIOC register contains override bits, which allow the user to inject an artificial input into the QEIx module for a module health test. This function allows the user to inject an artificial input into the module and observe the expected response to ensure the module's proper operation.

This self-test allows functional safety focused customers to exercise the module with the known input and output responses for a module health test. The QEIxIOC[31:24] register contains individual override bits, which allow the user to inject an artificial individual input into the QEIx module for a self-test.

The QEI state and status registers cannot be written directly. Instead, the input override logic is used to cause errors. The QEIx count registers may be incorrect and it may not be possible to regain the correct QEI count when disabling the override. The override function intent is for the functional safety software to check operation and then disable overrides and reset the QEI count to zero.

20.5. Application Example

Often at system power-up, a machine or motor must be “homed” to a known position to establish a reference point. In this case, the motor and/or machine is moved in a specific direction at a slow speed until a home sensor is detected. The home switch position is not precise. Therefore, the system will use an index pulse to determine the precise point of “home”.

Once the motor is homed, the position counter is required to be loaded with a known value.

In this application scenario, the position counter could be initialized to Mode 3 (PIMOD[2:0] = 3), and the QEIxIC register could be loaded with the initialization value. Optionally, an interrupt could be generated at the end of this position counter (homing) initialization process by setting the PCIIEN bit in the QEIxSTAT register.

```
int main()
{
/*Remap of QEA, QEB, Index pins goes here*/

/*Clear position counter*/
POS1CNT = 0;

/*Enable QEI interrupt*/
_QEI1IE = 1;

/*Enable QEI interrupt on initializing the position counter on completion of
homing process*/
QEI1STATbits.PCIIEN = 1;
```

```
/*Initialize QEIxIC register*/
QEI1IC = 0x50;

QEI1CONbits.CCM = 0;       // Position counter in quadrature mode
QEI1CONbits.PIMOD = 3;     // Initialize the position counter on completion of
homing process
QEI1CONbits.ON = 1;    // Enable QEI module
}

void __attribute__((interrupt, no_auto_psv)) _QEI1Interrupt(void)
{
     _QEI1IF = 0;

    // Check if the source of interrupt is homing initialization
    if(QEI1STATbits.PCIIRQ)
    {
         // User application code goes here
    }
}
```

20.6. Interrupts

The following are the sources of QEI interrupts:

- Position Counter Overflow or Underflow Event (POSOVIRQ)
- Velocity Counter Overflow or Underflow Event (VELOVIRQ)
- Position Counter Initialization Process Complete (PCIIRQ)
- Position Counter Greater Than or Equal Compare Interrupt (PCEHQIRQ)
- Position Counter Less Than or Equal Compare Interrupt (PCLEQIRQ)
- Index Event Interrupt (IDXIRQ) (When not in Internal Timer mode CCM = 0b11)
- Home Event Interrupt (HOMIRQ) (When not in Internal Timer mode CCM = 0b11)
- QEI/Hall Transition Error (TERRIRQ)
- QEI/Hall Direction Change (DIRIRQ)
- Hall State Error (HSERRIRQ)
- QEI/Hall State Change (STACIRQ)
- Programmable Event Generation Delay after a Hall Transition (HPDLYIRQ)
- Hall State Match Interrupt(HSTAMIRQ)
- External Capture Interrupt (ECAPIRQ)

The QEIx Status register (QEIxSTAT) contains the individual interrupt enable bits and the corresponding interrupt status bits for each interrupt source. A status bit indicates that an interrupt request has occurred. The module reduces all of the QEI interrupts to a single interrupt signal to the interrupt controller module.

20.7. Error Events

The QEI/Hall decoder can generate error events in case of:

- Detection of invalid states
- Invalid transition from a previously valid state

20.7.1. Hall State Error

The Hall State Error Interrupt Flag, HSERRIRQ, is set in the QEIx Status register (QEIxSTAT[19]) when an invalid Hall code (`000` or `111`) is detected by the Hall decoder. If the HSERRIEN bit in QEI Status register (QEIxSTAT[18]) is set, an interrupt will be generated.

20.7.2. Transition Error

QEI mode: Decoder compares QEA and QEB states (after polarity inversion/ swap/filtering) against previous state. If more than one bit is different, it is an invalid transition that should never occur for properly working hardware, and a transition error flag, TERRIRQ, will be set in the QEI Status register (QEIxSTAT[17]). If TERRIEN bit in QEI Status register (QEIxSTAT[16]) is set, an interrupt will be generated.

Hall mode: Decoder compares QEA, QEB and QEC signals (after polarity/swap/filtering) against previous Hall state. If more than one bit changes, it is an invalid transition that should never occur for properly working hardware, and a Transition Error Flag, TERRIRQ, will be set in QEI Status register (QEIxSTAT[17]). If the TERRIEN bit in the QEI Status register (QEIxSTAT[16]) is set, an interrupt will be generated.

20.8. QEI Operation in Power-Saving Modes

20.8.1. Sleep Mode

When the device enters Sleep mode, QEI operations cease. The POSxCNT register stops at the current value. The QEI does not respond to active signals on the QEAx, QEBx or INDXx pins. The QEIxCON register remains unchanged.

20.8.2. Idle Mode

When the device enters Idle mode, the QEISIDL bit in the QEIx Control register (QEIxCON[13]) determines whether the QEI module stops in Idle mode or continues to operate in Idle mode.

If QEISIDL = 1, the QEI module enters into a Power-Saving mode and performs the same functions as in Sleep mode. If QEISIDL = `0`, the module does not enter into a Power-Saving mode and continues operation in Idle mode.

21. Universal Asynchronous Receiver Transmitter (UART)

The Universal Asynchronous Receiver Transmitter (UART) is a flexible serial communication peripheral used to interface microcontrollers with other equipment, including computers and peripherals. The UART is a full-duplex, asynchronous communication channel that can be used to implement protocols, such as RS-232 and RS-485.

The UART also supports the following hardware extensions:

- LIN 2.2/J2602
- IrDA®
- Digital Multiplex 512 (DMX)
- Smart Card (ISO 7816)

The primary features of the UART are:

- Full or Half-Duplex Operation
- Up to 8-Deep TX and RX First-In First-Out (FIFO) Buffers
- 8-Bit or 9-Bit Data Width
- Configurable Stop Bit Length
- Flow Control
- Auto-Baud Calibration
- Parity, Framing and Buffer Overrun Error Detection
- Address Detect
- Break Transmission
- Transmit and Receive Polarity Control
- Operation in Sleep Mode
- Wake from Sleep on Sync Break Received Interrupt

21.1. Device-Specific Information

Table 21-1. UART Summary Table

UART Module Instances	PPS Availability	Peripheral Bus Speed	Input Clock Speed	Clock Source
4	All Instances	Standard (1:2 CPU Clock)	See Electrical Characteristics	See Table 21-2

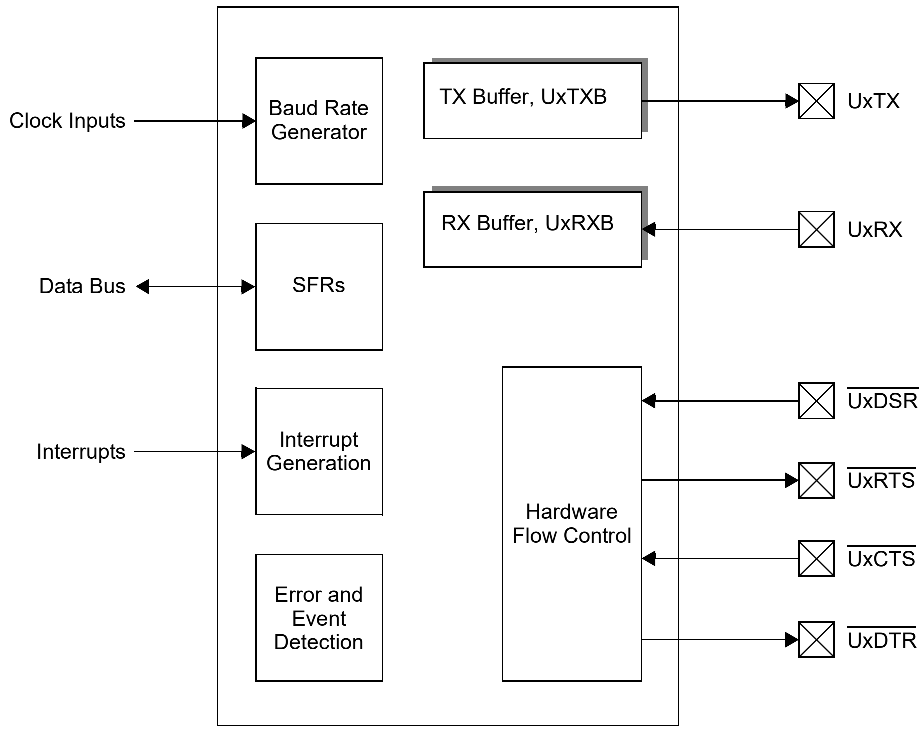
Table 21-2. UART Clock (F_{UART}) Source Selection bits

Value	Description
1	CLKGEN8
0	Standard (1:2 CPU Clock)

21.2. Architectural Overview

The UART transfers bytes of data to and from device pins using First-In First-Out (FIFO) buffers up to eight bytes deep. The status of the buffers and data is made available to user software through Special Function Registers (SFRs). The UART implements multiple interrupt channels for handling transmit, receive and error events. A simplified block diagram of the UART is shown in Figure 21-1.

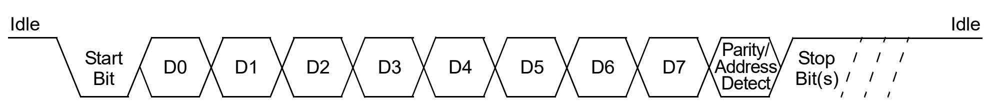
Figure 21-1. Simplified UARTx Block Diagram



21.2.1. Character Frame

A typical UART character frame is shown in Figure 21-2. The Idle state is high with a Start condition indicated by a falling edge. The Start bit is followed by a number of data, Parity/Address Detect and Stop bits defined by the MODE[3:0] (UxCON[3:0]) bits selected.

Figure 21-2. UART Character Frame



21.2.2. Data Buffers

Both transmit and receive functions use buffers to store data shifted to/from the pins. These buffers are FIFOs and are accessed by reading the SFRs, UxTXB and UxRXB, respectively. Each data buffer has multiple flags associated with its operation to allow the software to read the status. Interrupts can also be configured based on the space available in the buffers. The transmit and receive buffers can be cleared, and their pointers reset using the associated TX/RX Buffer Empty Status bits, TXBE and RXBE (U1STAT).

21.2.3. Protocol Extensions

The UART provides hardware support for LIN/J2602, DMX and smart card protocol extensions to reduce software overhead. A protocol extension is enabled by writing a value to the MODE[3:0] (UxCON[3:0]) selection bits and further configured using the UARTx Timing Parameter registers, UxPA and UxPB. Details regarding operation and usage are discussed in their respective chapters. Not all protocols are available on all devices.

21.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1700	U1CON	31:24	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
		23:16	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
		15:8	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
		7:0	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
0x1704	U1STAT	31:24		TXWM[2:0]				RXWM[2:0]		
		23:16	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
		15:8	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
		7:0	TXMTIF	PERIF	ABDOVIF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
0x1708	U1BRG	31:24								
		23:16					BRG[19:16]			
		15:8	BRG[15:8]							
		7:0	BRG[7:0]							
0x170C	U1RXB	31:24								
		23:16								
		15:8								
		7:0	RXB[7:0]							
0x1710	U1TXB	31:24								
		23:16								
		15:8	LAST							
		7:0	TXB[7:0]							
0x1714	U1PA	31:24	WIP							P2[8]
		23:16	P2[7:0]							
		15:8								P1[8]
		7:0	P1[7:0]							
0x1718	U1PB	31:24	WIP							
		23:16	P3[23:16]							
		15:8	P3[15:8]							
		7:0	P3[7:0]							
0x171C	U1CHK	31:24								
		23:16	RXCHK[7:0]							
		15:8								
		7:0	TXCHK[7:0]							
0x1720	U1SCCON	31:24			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
		23:16			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
		15:8								
		7:0			TXRPT[1:0]		CONV	T0PD	PRTCL	
0x1724	U1UIR	31:24								
		23:16								
		15:8								
		7:0	WUIF	ABDIF				ABDIE		
0x1728 ... 0x173F	Reserved									
0x1740	U2CON	31:24	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
		23:16	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
		15:8	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
		7:0	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
0x1744	U2STAT	31:24		TXWM[2:0]				RXWM[2:0]		
		23:16	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
		15:8	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
		7:0	TXMTIF	PERIF	ABDOVIF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
0x1748	U2BRG	31:24								
		23:16					BRG[19:16]			
		15:8	BRG[15:8]							
		7:0	BRG[7:0]							
0x174C	U2RXB	31:24								
		23:16								
		15:8								
		7:0	RXB[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1750	U2TXB	31:24								
		23:16								
		15:8	LAST							
		7:0	TXB[7:0]							
0x1754	U2PA	31:24	WIP							P2[8]
		23:16	P2[7:0]							
		15:8								P1[8]
		7:0	P1[7:0]							
0x1758	U2PB	31:24	WIP							
		23:16	P3[23:16]							
		15:8	P3[15:8]							
		7:0	P3[7:0]							
0x175C	U2CHK	31:24								
		23:16	RXCHK[7:0]							
		15:8								
		7:0	TXCHK[7:0]							
0x1760	U2SCCON	31:24			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
		23:16			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
		15:8								
		7:0			TXRPT[1:0]		CONV	T0PD	PRTCL	
0x1764	U2UIR	31:24								
		23:16								
		15:8								
		7:0	WUIF	ABDIF				ABDIE		
0x1768 ... 0x177F	Reserved									
0x1780	U3CON	31:24	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
		23:16	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
		15:8	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
		7:0	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
0x1784	U3STAT	31:24		TXWM[2:0]				RXWM[2:0]		
		23:16	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
		15:8	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
		7:0	TXMTIF	PERIF	ABDOVIF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
0x1788	U3BRG	31:24								
		23:16					BRG[19:16]			
		15:8	BRG[15:8]							
		7:0	BRG[7:0]							
0x178C	U3RXB	31:24								
		23:16								
		15:8								
		7:0	RXB[7:0]							
0x1790	U3TXB	31:24								
		23:16								
		15:8	LAST							
		7:0	TXB[7:0]							
0x1794	U3PA	31:24	WIP							P2[8]
		23:16	P2[7:0]							
		15:8								P1[8]
		7:0	P1[7:0]							
0x1798	U3PB	31:24	WIP							
		23:16	P3[23:16]							
		15:8	P3[15:8]							
		7:0	P3[7:0]							
0x179C	U3CHK	31:24								
		23:16	RXCHK[7:0]							
		15:8								
		7:0	TXCHK[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x17A0	U3SCCON	31:24			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
		23:16			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
		15:8								
		7:0			TXRPT[1:0]		CONV	T0PD	PRTCL	
0x17A4	U3UIR	31:24								
		23:16								
		15:8								
		7:0	WUIF	ABDIF				ABDIE		
0x17A8 ... 0x17BF	Reserved									
0x17C0	U4CON	31:24	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
		23:16	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
		15:8	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
		7:0	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
0x17C4	U4STAT	31:24		TXWM[2:0]				RXWM[2:0]		
		23:16	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
		15:8	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
		7:0	TXMTIF	PERIF	ABDOVIF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
0x17C8	U4BRG	31:24								
		23:16					BRG[19:16]			
		15:8	BRG[15:8]							
		7:0	BRG[7:0]							
0x17CC	U4RXB	31:24								
		23:16								
		15:8								
		7:0	RXB[7:0]							
0x17D0	U4TXB	31:24								
		23:16								
		15:8	LAST							
		7:0	TXB[7:0]							
0x17D4	U4PA	31:24	WIP							P2[8]
		23:16	P2[7:0]							
		15:8								P1[8]
		7:0	P1[7:0]							
0x17D8	U4PB	31:24	WIP							
		23:16	P3[23:16]							
		15:8	P3[15:8]							
		7:0	P3[7:0]							
0x17DC	U4CHK	31:24								
		23:16	RXCHK[7:0]							
		15:8								
		7:0	TXCHK[7:0]							
0x17E0	U4SCCON	31:24			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
		23:16			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
		15:8								
		7:0			TXRPT[1:0]		CONV	T0PD	PRTCL	
0x17E4	U4UIR	31:24								
		23:16								
		15:8								
		7:0	WUIF	ABDIF				ABDIE		

21.3.1. UARTx Configuration Register

Name: UxCON
Offset: 0x1700, 0x1740, 0x1780, 0x17C0

Note: R/HS/HC in DMX and LIN modes.

Bit	31	30	29	28	27	26	25	24
	SLPEN	ACTIVE			CLKMOD	CLKSEL[1:0]		HALFDPLX
Access	R/W	R			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RUNOVF	RXPOL	STP[1:0]		C0EN	TXPOL	FLO[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	WUE	RXBIMD		BRKOVR	SENDB
Access	R/W		R/W	R/W	R/W		R/W	R/W/HC
Reset	0		0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	BRGS	ABDEN	TXEN	RXEN	MODE[3:0]			
Access	R/W	R/W/HC	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – SLPEN Run During Sleep Enable bit

Value	Description
1	UART BRG clock runs during Sleep.
0	UART BRG clock is turned off during Sleep.

Bit 30 – ACTIVE UART Running Status bit

Value	Description
1	UART clock request is active (the user should not update the UxCON register).
0	UART clock request is not active (the user can update the UxCON register).

Bit 27 – CLKMOD Baud Clock Generation Mode Select bit

Value	Description
1	Uses fractional Baud Rate Generation.
0	Uses a legacy divide-by-x counter for baud clock generation (x = 4 or 16 depending on the BRGS bit).

Bits 26:25 – CLKSEL[1:0] Baud Clock Source Selection bits
See Table 21-2.

Bit 24 – HALFDPLX UART Half-Duplex Selection Mode bit
(HALFDPLX is ignored in LIN and Smart Card modes.)

Value	Description
1	Half-Duplex mode
0	Full-Duplex mode

Bit 23 – RUNOVF Run During Overflow Condition Mode bit

Value	Description
1	When an Overflow Error (RXFOIF) condition is detected, the RX shifter continues to run to remain synchronized with incoming RX data; data are not transferred to UxRXB when it is full (i.e., no UxRXB data are overwritten).
0	When an Overflow Error (RXFOIF) condition is detected, the RX shifter stops accepting new data; data are transferred to UxRXB when one empty slot becomes available in the buffer (Legacy mode).

Bit 22 – RXPOL UART Receive Polarity bit

Value	Description
1	Inverts RX polarity; the Idle state is low.
0	Input is not inverted; the Idle state is high.

Bits 21:20 – STP[1:0] Number of Stop Bits Selection bits

Value	Description
11	2 Stop bits sent, 1 checked at receive.
10	2 Stop bits sent, 2 checked at receive.
01	1.5 Stop bits sent, 1.5 checked at receive.
00	1 Stop bit sent, 1 checked at receive.

Bit 19 – C0EN Enable Legacy Checksum (C0) Transmit and Receive bit

Value	Description
1	Checksum Mode 1 (Enhanced LIN checksum in LIN mode; add all TX/RX words in all other modes.)
0	Checksum Mode 0 (Legacy LIN checksum in LIN mode; not used in all other modes.)

Bit 18 – TXPOL UART Transmit Polarity bit

Value	Description
1	Inverts TX polarity; TX is low in the Idle state.
0	Output data are not inverted; TX output is high in the Idle state.

Bits 17:16 – FLO[1:0] Flow Control Enable bits (only valid when MODE[3:0] = 0xxx)

Value	Description
11	Reserved
10	UxRTS-UxDSR (for TX side)/UxCTS-UxDTR (for RX side) hardware flow control
01	XON/XOFF software flow control
00	Flow control off

Bit 15 – ON UART Enable bit

Value	Description
1	UART is ready to transmit and receive.
0	UART state machine, FIFO buffer pointers, and counters are reset; registers are readable and writable.

Bit 13 – SIDL UART Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when the device enters Idle mode.
0	Continues module operation in Idle mode.

Bit 12 – WUE Wake-up Enable bit

Value	Description
1	Module will continue to sample the UxRX pin – an interrupt is generated on the falling edge, and the bit is cleared in hardware on the following rising edge; if ABDEN is set, Auto-Baud Detection (ABD) will begin immediately.
0	UxRX pin is not monitored, nor is the rising edge detected.

Bit 11 – RXBIMD Receive Break Interrupt Mode bit

Value	Description
1	RXBKIF flag when a minimum of 23 (DMX)/11 (asynchronous or LIN/J2602) low bit periods are detected.
0	RXBKIF flag when the break makes a low-to-high transition after being low for at least 23/11 bit periods.

Bit 9 – BRKOVR Send Break Software Override bit
Overrides the TX Data Line:

Value	Description
1	Makes the TX line active (Output 0 when TXPOL = 0, Output 1 when TXPOL = 1).
0	TX line is driven by the shifter.

Bit 8 – SENDB UART Transmit Break bit[(1)]

Value	Description
1	Sends Sync Break on the next transmission; cleared by hardware upon completion.
0	Sync Break transmission is disabled or has been completed.

Bit 7 – BRGS High Baud Rate Select bit

Value	Description
1	High speed
0	Low speed

Bit 6 – ABDEN Auto-Baud Detect Enable bit (read-only when MODE[3:0] = 1xxx)

Value	Description
1	Enables baud rate measurement on the next character – requires reception of a Sync field (55h); cleared in hardware upon completion.
0	Baud rate measurement is disabled or has been completed.

Bit 5 – TXEN UART Transmit Enable bit

Value	Description
1	Transmit enabled – except during Auto-Baud Detection.
0	Transmit disabled – all transmit counters, pointers, and state machines are reset; the TX buffer is not flushed, and status bits are not reset.

Bit 4 – RXEN UART Receive Enable bit

Value	Description
1	Receive enabled – except during Auto-Baud Detection.
0	Receive disabled – all receive counters, pointers, and state machines are reset; the RX buffer is not flushed, and status bits are not reset.

Bits 3:0 – MODE[3:0] UART Mode bits

Value	Description
Other	Reserved
1111	Smart card
1110	IrDA®
1101	Reserved
1100	LIN Commander/Responder
1011	LIN Responder only
1010	DMX
1001-0101	Reserved
0100	Asynchronous 9-bit UART with address detect, ninth bit = 1 signals address
0011	Asynchronous 8-bit UART without address detect, ninth bit is used as an even parity bit

Value	Description
0010	Asynchronous 8-bit UART without address detect, ninth bit is used as an odd parity bit
0001	Asynchronous 7-bit UART
0000	Asynchronous 8-bit UART

21.3.2. UARTx Status Register

Name: UxSTAT
Offset: 0x1704, 0x1744, 0x1784, 0x17C4

Note:
1. The receive watermark interrupt is not set if PERIF, FERIF or TXCIF is set and the corresponding IE bit is set.

Legend: S = Settable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
		TXWM[2:0]				RXWM[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
	TXWRE	STPMD	TXBE	TXBF	RCIDL	XON	RXBE	RXBF
Access	R/W/HS	R/W	R/S	R	R	R	R/S	R
Reset	0	0	1	0	1	1	1	0

Bit	15	14	13	12	11	10	9	8
	TXMTIE	PERIE	ABDOVIE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXMTIF	PERIF	ABDOVIF	CERIF	FERIF	RXBKIF	RXFOIF	TXCIF
Access	R	R	R/W/HS	R/W/HC	R	R/W/HC	R/W/HC	R/W/HC
Reset	1	0	0	0	0	0	0	0

Bits 30:28 – TXWM[2:0] UART Transmit Interrupt Select bits

Value	Description
111	Sets transmit interrupt when there is one empty slot left in the buffer.
. . .	
010	Sets transmit interrupt when there are six empty slots or more in the buffer.
001	Sets transmit interrupt when there are seven empty slots or more in the buffer.
000	Sets transmit interrupt when there are eight empty slots in the buffer; the TX buffer is empty.

Bits 26:24 – RXWM[2:0] UART Receive Interrupt Select bits(1)

Value	Description
111	Triggers receive interrupt when there are eight words in the buffer; RX buffer is full.
. . .	
001	Triggers receive interrupt when there are two words or more in the buffer.
000	Triggers receive interrupt when there is one word or more in the buffer.

Bit 23 – TXWRE TX Write Transmit Error Status bit

LIN and Parity Modes:

1 = A new byte was written when the buffer was full or when P2[8:0] = 0 (must be cleared by software).

0 = No error

Address Detect Mode:

1 = A new byte was written when the buffer was full or to P1[8:0] when P1x was full (must be cleared by software).
0 = No error.
Other Modes:
1 = A new byte was written when the buffer was full (must be cleared by software).
0 = No error

Bit 22 – STPMD Stop Bit Detection Mode bit

Value	Description
1	Triggers RXIF at the end of the last Stop bit.
0	Triggers RXIF in the middle of the first (or second, depending on the STP[1:0] setting) Stop bit.

Bit 21 – TXBE UART TX Buffer Empty Status bit

Value	Description
1	Transmit buffer is empty; writing '1' when TXEN = 0 will reset the TX FIFO pointers and counters.
0	Transmit buffer is not empty.

Bit 20 – TXBF UART TX Buffer Full Status bit

Value	Description
1	Transmit buffer is full.
0	Transmit buffer is not full.

Bit 19 – RCIDL Receive Idle bit

Value	Description
1	UART RX line is in the Idle state.
0	UART RX line is receiving something.

Bit 18 – XON UART in XON Mode bit
Only valid when FLO[1:0] control bits are set to XON/XOFF mode.

Value	Description
1	UART has received XON.
0	UART has not received XON, or XOFF was received.

Bit 17 – RXBE UART RX Buffer Empty Status bit

Value	Description
1	Receive buffer is empty; writing '1' when RXEN = 0 will reset the RX FIFO pointers and counters.
0	Receive buffer is not empty.

Bit 16 – RXBF UART RX Buffer Full Status bit

Value	Description
1	Receive buffer is full.
0	Receive buffer is not full.

Bit 15 – TXMTIE Transmit Shifter Empty Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 14 – PERIE Parity Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 13 – ABDOVIE Auto-Baud Rate Acquisition Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 12 – CERIE Checksum Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 11 – FERIE Framing Error Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 10 – RXBKIE Receive Break Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 9 – RXFOIE Receive Buffer Overflow Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 8 – TXCIE Transmit Collision Interrupt Enable bit

Value	Description
1	Interrupt is enabled.
0	Interrupt is disabled.

Bit 7 – TXMTIF Transmit Shifter Empty Interrupt Flag bit

Value	Description
1	Transmit Shift Register (TSR) is empty (the TXMTIF bit is set at the end of the last Stop bit; the TXMTIF bit behavior is independent of the STPMD bit).
0	Transmit Shift Register is not empty.

Bit 6 – PERIF Parity Error/Address Received

LIN and Parity Modes:
1 = Parity error detected.
0 = No parity error detected.
Address Mode:
1 = Address received.
0 = No address detected.
All Other Modes:
Not used.

Bit 5 – ABDOVIF Auto-Baud Rate Acquisition Interrupt Flag bit

Value	Description
1	BRG rolled over during the auto-baud rate acquisition sequence.
0	BRG has not rolled over during the auto-baud rate acquisition sequence.

Bit 4 – CERIF Checksum Error Interrupt Flag bit (must be cleared by software)

Value	Description
1	Checksum error
0	No checksum error.

Bit 3 – FERIF Framing Error Interrupt Flag bit

Value	Description
1	Framing Error: Inverted level of the Stop bit corresponding to the topmost character in the buffer; it propagates through the buffer with the received character.
0	No framing error.

Bit 2 – RXBKIF Receive Break Interrupt Flag bit (must be cleared by software)

Value	Description
1	A Break was received.
0	No Break was detected.

Bit 1 – RXFOIF Receive Buffer Overflow Interrupt Flag bit (must be cleared by software)

Value	Description
1	Receive buffer has overflowed.
0	Receive buffer has not overflowed.

Bit 0 – TXCIF Transmit Collision Interrupt Flag bit (must be cleared by software)

Value	Description
1	Transmitted word is not equal to the received word.
0	Transmitted word is equal to the received word.

21.3.3. UARTx Baud Rate Register

Name: UxBRG
Offset: 0x1708, 0x1748, 0x1788, 0x17C8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					BRG[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BRG[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BRG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 19:0 – BRG[19:0] Baud Rate Divisor bits

21.3.4. UARTx Receive Buffer Register

Name: UxRXB
Offset: 0x170C, 0x174C, 0x178C, 0x17CC

Note:

1. The RXB[8] bit is used only in Address Detect mode.

Legend: x = Bit is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	RXB[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – RXB[7:0] Received Character Data bits 8-0[(1)]

21.3.5. UARTx Transmit Buffer Register

Name: UxTXB
Offset: 0x1710, 0x1750, 0x1790, 0x17D0

Legend: x = Bit is unknown

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LAST							
Access	W							
Reset	x							

Bit	7	6	5	4	3	2	1	0
	TXB[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	x	x	x	x	x	x	x	x

Bit 15 – LAST Last Byte Indicator for Smart Card Support bit

Bits 7:0 – TXB[7:0] Transmitted Character Data bits 7-0
If the buffer is full, further writes to the buffer are ignored.

21.3.6. UARTx Timing Parameter A Register

Name: UxPA
Offset: 0x1714, 0x1754, 0x1794, 0x17D4

Notes:
1. The WIP bit is relevant when the UART clock differs from the CPU clock. It indicates whether the UART and CPU clocks are synchronized for writes to the parameter registers.
2. To write to P2 without affecting P1, use UxPAbits.P2 = value. Writing to the entire UxPA register overwrites P1, potentially causing unexpected behavior in LIN and Address Detect modes. When accessing through pointers, ensure a 16-bit pointer targets P2's base address for writing.

Bit	31	30	29	28	27	26	25	24
	WIP							P2[8]
Access	R/W							R/W
Reset	0							

Bit	23	22	21	20	19	18	17	16
	P2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
								P1[8]
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
	P1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – WIP UxPA Write in Progress bit(1)

Value	Description
1	Write is still in progress (the user should not update the UxPA register).
0	No write in progress (user can update the UxPA register).

Bits 24:16 – P2[8:0] Parameter 2 bits(2)

DMX RX:
The First Byte Number to Receive – 1, not including the start code (bits[8:0]).
LIN Responder TX:
Number of bytes to transmit (bits [7:0]).
Asynchronous RX with Address Detect:
ADDR to match (bits[7:0]).
Smart Card Mode:
Block Time Counter (BTC) bits. This counter is operated on the bit clock, whose period is always equal to one ETU (bits[8:0]).
Other Modes:
Not used.

Bits 8:0 – P1[8:0] Parameter 1 bits

DMX TX:
Number of bytes to transmit – 1 (not including the start code).

LIN Commander TX:
PID to transmit (bits[5:0]).
Asynchronous TX with Address Detect:
Address to transmit. A ‘1’ is automatically inserted into bit 9 (bits [7:0]).
Smart Card Mode:
Guard Time Counter bits. This counter operates on the bit clock, whose period is always equal to one ETU (bits[8:0]).
Other Modes:
Not used.

21.3.7. UARTx Timing Parameter B Register

Name: UxPB
Offset: 0x1718, 0x1758, 0x1798, 0x17D8

Note:

1. The WIP bit is relevant when the UART clock differs from the CPU clock. It indicates whether the UART and CPU clocks are synchronized for writes to the parameter registers.

Bit	31	30	29	28	27	26	25	24
	WIP							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
	P3[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	P3[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	P3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – WIP UxPB Write in Progress bit[1]

Value	Description
1	Write still in progress (the user should not update the UxPB registers).
0	No write in progress (user can update the UxPB registers).

Bits 23:0 – P3[23:0] Parameter 3 bits

DMX RX:
The last byte number to receive is 1, not including the start code (bits[8:0]).
LIN Responder RX:
Number of bytes to receive (bits [7:0]).
Asynchronous RX:
Used to mask the P2 address bits; 1 = P2 address bit is used, 0 = P2 address bit is masked off (bits [7:0]).
Smart Card Mode:
Waiting Time Counter bits (bits [23:0]).
Other Modes:
Not used.

21.3.8. UART Checksum Result Register

Name: UxCHK
Offset: 0x171C, 0x175C, 0x179C, 0x17DC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	RXCHK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	TXCHK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – RXCHK[7:0] Receive Checksum bits (calculated from RX words)
LIN Modes:
C0EN = 1: Sum of all received data + additional carries, including PID.
C0EN = 0: Sum of all received data + additional carries, excluding PID.
LIN Responder:
Cleared when a Break is detected.
LIN Commander/Responder:
Cleared when a Break is detected.
Other Modes:
C0EN = 1: Sum of every byte received + additional carries.
C0EN = 0: Value remains unchanged.

Bits 7:0 – TXCHK[7:0] Transmit Checksum bits (calculated from TX words)
LIN Modes:
C0EN = 1: Sum of all transmitted data + addition carries, including PID.
C0EN = 0: Sum of all transmitted data + addition carries, excluding PID.
LIN Responder:
Cleared when Break is detected.
LIN Commander/Responder:
Cleared when Break is detected.
Other Modes:
C0EN = 1: Sum of every byte transmitted + addition carries.
C0EN = 0: Value remains unchanged.

21.3.9. UARTx Smart Card Configuration Register

Name: UxSCCON
Offset: 0x1720, 0x1760, 0x17A0, 0x17E0

Bit	31	30	29	28	27	26	25	24
			RXRPTIF	TXRPTIF		BTCIF	WTCIF	GTCIF
Access			R/W/HS	R/W/HS		R/W/HS	R/W/HS	R/W/HS
Reset			0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
			RXRPTIE	TXRPTIE		BTCIE	WTCIE	GTCIE
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			TXRPT[1:0]		CONV	T0PD	PRTCL	
Access			R/W	R/W	R/W	R/W	R/W	
Reset			0	0	0	0	0	

Bit 29 – RXRPTIF Receive Repeat Interrupt Flag bit

Value	Description
1	Parity error has persisted after the same character has been received five times (four retransmits).
0	Flag is cleared.

Bit 28 – TXRPTIF Transmit Repeat Interrupt Flag bit

Value	Description
1	Line error has been detected after the last retransmit per TXRPT<1:0>.
0	Flag is cleared.

Bit 26 – BTCIF Block Time Counter Interrupt Flag bit

Value	Description
1	Block time counter has reached 0.
0	Block time counter has not reached 0.

Bit 25 – WTCIF Waiting Time Counter Interrupt Flag bit

Value	Description
1	Waiting time counter has reached 0.
0	Waiting time counter has not reached 0.

Bit 24 – GTCIF Guard Time Counter Interrupt Flag bit

Value	Description
1	Guard time counter has reached 0.
0	Guard time counter has not reached 0.

Bit 21 – RXRPTIE Receive Repeat Interrupt Enable bit

Value	Description
1	An interrupt is invoked when a parity error has persisted after the same character has been received five times (four retransmits).
0	Interrupt is disabled.

Bit 20 – TXRPTIE Transmit Repeat Interrupt Enable bit

Value	Description
1	An interrupt is invoked when a line error is detected after the last retransmit per TXRPT<1:0> has been completed.
0	Interrupt is disabled.

Bit 18 – BTCIE Block Time Counter Interrupt Enable bit

Value	Description
1	Block time counter interrupt is enabled.
0	Block time counter interrupt is disabled.

Bit 17 – WTCIE Waiting Time Counter Interrupt Enable bit

Value	Description
1	Waiting time counter interrupt is enabled.
0	Waiting time counter interrupt is disabled.

Bit 16 – GTCIE Guard Time Counter Interrupt Enable bit

Value	Description
1	Guard time counter interrupt is enabled.
0	Guard time counter interrupt is disabled.

Bits 5:4 – TXRPT[1:0] Transmit Repeat Selection bits

Value	Description
11	Retransmit the error byte four times.
10	Retransmit the error byte three times.
01	Retransmit the error byte twice.
00	Retransmit the error byte once.

Bit 3 – CONV Logic Convention Selection bit

Value	Description
1	Inverse logic convention
0	Direct logic convention

Bit 2 – T0PD Pull-Down Duration for T = 0 Error Handling bit

Value	Description
1	2 ETU
0	1 ETU

Bit 1 – PRTCL Smart Card Protocol Selection bit

Value	Description
1	T = 1
0	T = 0

21.3.10. UARTx Interrupt Register

Name: UxUIR
Offset: 0x1724, 0x1764, 0x17A4, 0x17E4

Note:

1. After the occurrence of the WAKE event, the WUIF flag can only be cleared once the wake (WUE) bit is cleared by hardware following the rising edge.

Legend: HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	WUIF	ABDIF				ABDIE		
Access	R/W/HS	R/W/HS				R/W		
Reset	0	0				0		

Bit 7 – WUIF Wake-up Interrupt Flag bit[(1)]

Value	Description
1	Sets when WUE = 1 and RX makes a 1-to-0 transition; triggers an event interrupt (must be cleared by software).
0	WUE is not enabled, or WUE is enabled, but no wake-up event has occurred.

Bit 6 – ABDIF Auto-Baud Completed Interrupt Flag bit

Value	Description
1	Sets when the ABD sequence makes the final 1-to-0 transition; triggers event interrupt (must be cleared by software).
0	ABDEN is not enabled, or ABDEN is enabled but auto-baud has not completed.

Bit 2 – ABDIE Auto-Baud Completed Interrupt Enable Flag bit

Value	Description
1	Allows ABDIF to set an event interrupt.
0	ABDIF does not set an event interrupt.

21.4. Operation

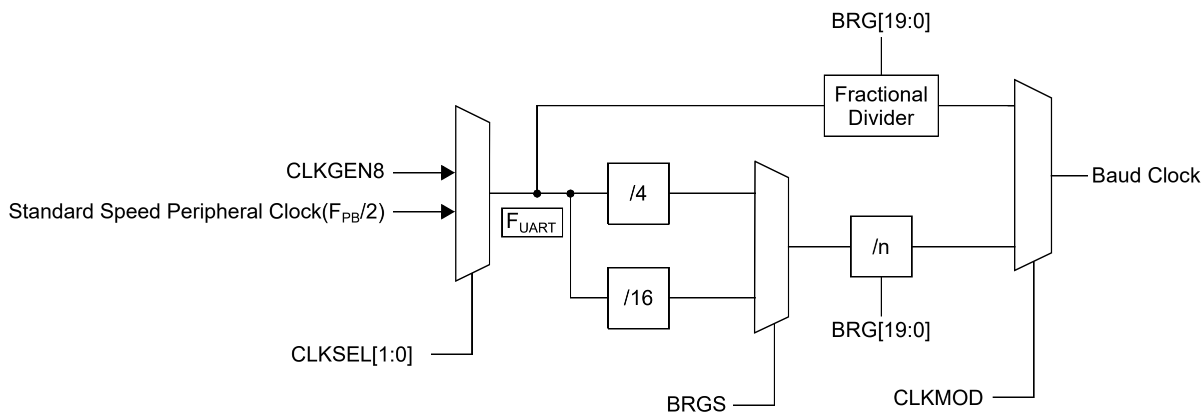
21.4.1. Clocking and Baud Rate Configuration

The UART supports multiple clock sources and two types of Baud Rate Generation (BRG). One of the clock sources provided is selected by the CLKSEL[1:0] bits (UxCON[26:25]). Clock source selection and prescaler can only be changed when the ON bit (UxCON[15]) is cleared. The baud clock can be generated with one of the following methods:

- Legacy mode, fixed division
- Fractional Division mode

To allow synchronized time of clock domains, do not make write-to-back to the UxBRG register. UxBRG should be written only when ON = `0` to avoid corruption of an ongoing transmission or reception. A block diagram of the UART clocking is shown in Figure 21-3.

Figure 21-3. UART Clocking Diagram



21.4.1.1. Legacy Mode

In Legacy mode, the clock source is divided down to the desired baud clock using integer division. Legacy mode is selected when CLKMOD = 0 (UxCON[27]). A selectable prescaler is present to support a wide range of baud rates and is controlled by the BRGS bit (UxCON[7]). Up to a 20-bit value of BRG (UxBRG[19:0]) is used to further divide down the input clock to the final baud rate.

Equation 21-1 and Equation 21-2 show formulas for baud rate and BRG value given by the BRGS for all protocol modes.

Equation 21-1. Baud Rate When BRGS = 0, CLKMOD = 0

$$\text{Baud Rate} = \frac{F_{UART}}{16 \times (BRG + 1)}$$

$$BRG = \frac{F_{UART}}{16 \times \text{Baud Rate}} - 1$$

Note: F_{UART} = UART Clock Frequency.

Equation 21-2. Baud Rate When BRGS = 1, CLKMOD = 0

$$\text{Baud Rate} = \frac{F_{UART}}{4 \times (BRG + 1)}$$

$$BRG = \frac{F_{UART}}{4 \times \text{Baud Rate}} - 1$$

Note: BRG values should be three or more for proper smart card communication.

The UART fixed division baud rate setup procedure:

1. Select the clock input source with the CLKSEL[1:0] bits.
2. Clear the CLKMOD bit.
3. Select the clock prescaler by writing a value to BRGS.
4. Using Equation 21-1 or Equation 21-2, calculate the value for BRG and write to the UxBRG register.
5. Set the ON bit.

21.4.1.2. Fractional Division Mode

To reduce the baud rate error, a fractional division scheme can be used by setting CLKMOD = 1. The fractional baud clock circuit works by occasionally extending clock pulses of the 16x baud clock to achieve a baud clock closer to the ideal baud rate. This mode allows for faster operation of the UART while maintaining the noise rejection benefits of 16x oversampling, where in Legacy mode, a small value of BRG results in an unacceptable error.

The fractional Baud Rate Generation logic performs modulo arithmetic, where the value 16 is constantly accumulated in a counter until the sum is larger than the UxBRG register value. When the sum becomes larger than the UxBRG register value, a clock pulse is produced and the accumulated value is reduced by the value in the UxBRG register.

A timing example for the fractional baud clock circuit is shown in Figure 21-4. In this example, a 50 MHz UART clock and a target baud rate of 921600 baud are used. This results in UxBRG = 54. The upper waveform shows one full bit time, while the lower waveform shows the accumulation process. In this example, the 16x baud clock pulses are generated every three to four UART clock (F_{UART}) cycles. While this results in 16x baud clock sampling pulses with unequal periods, each full bit time (16 pulses of the 16x baud clock) will be equal. The resulting bit period generated will be closer to the ideal bit period than with the legacy divider-based BRG.

Equation 21-3 shows the baud rate formulas.

Equation 21-3. Baud Rate Formulas

$$\text{Baud Rate} = \frac{F_{UART}}{BRG}$$

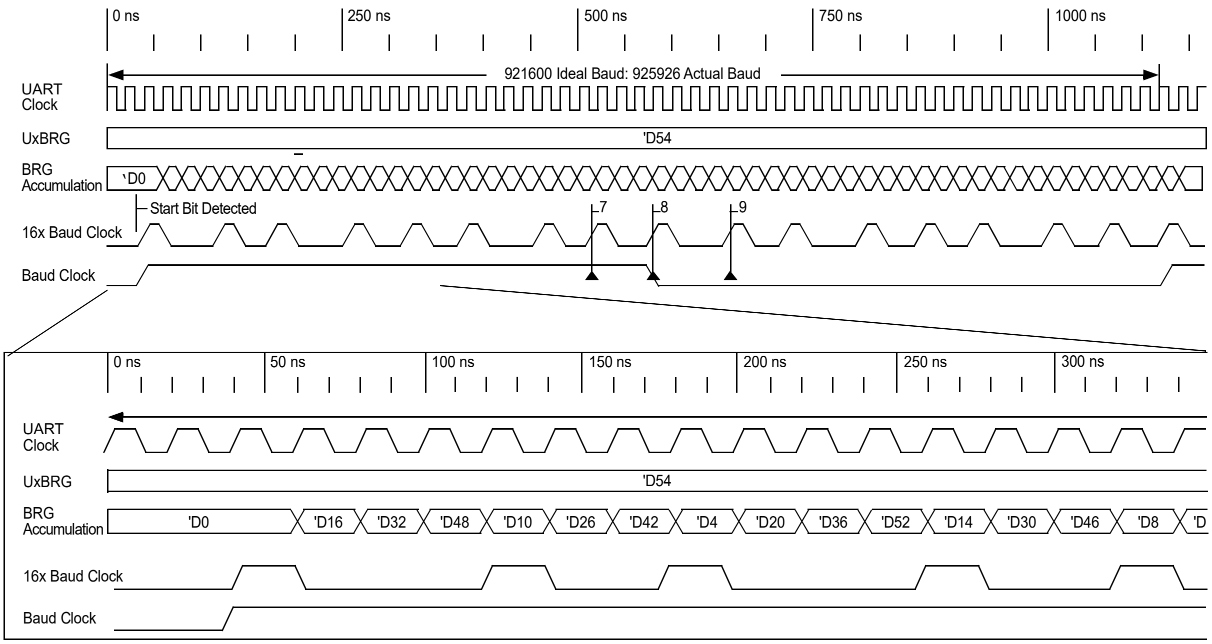
$$BRG = \frac{F_{UART}}{\text{Baud Rate}}$$

Note: When CLKMOD (UxCON[27]) = `1`, the minimum BRG value is 16.

UART fractional baud rate setup procedure:

1. Select the clock input source with the CLKSEL[1:0] bits.
2. Set the CLKMOD bit.
3. Using Equation 21-3, calculate the value for BRG and write to the UxBRG register.
4. Set the ON bit.

Figure 21-4. Fractional Division Mode



21.4.1.3. UART Baud Rate Tables

UART baud rates are provided in Table 21-3, Table 21-4 and Table 21-5 for different UART Clock Frequencies (F_{UART}). The minimum and maximum baud rates for each frequency are also shown.

Table 21-3. UART Baud Rates (CLKMOD = 0 and BRGS = 0)

BAUD RATE	F_{UART} = 100 MHz			F_{UART} = 80 MHz		
	Actual Baud Rate	% Error	BRG Value	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	650	9615.4	0.16	519
19,200	19230.8	0.16	324	19230.8	0.16	259
38,400	38580.2	0.47	161	38461.5	0.16	129
56,000	56306.3	0.55	110	56179.8	0.32	88
115,000	115740.7	0.64	53	116279.1	1.11	42
250,000	250000.0	0.00	24	250000.0	0.00	19
300,000	312500.0	4.17	19	312500.0	4.17	15
500,000	520833.3	4.17	11	500000.0	0.00	9
1,000,000	1041666.7	4.17	5	1000000.0	0.00	4
Min.	5.96	0.00	1048575	4.77	0.00	1048575
Max.	6250000.0	0.00	0	5000000.0	0.00	0

BAUD RATE	F_{UART} = 50 MHz			F_{UART} = 32 MHz			F_{UART} = 4 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9615.4	0.16	324	9615.4	0.16	207	9615.4	0.16	25
19,200	19290.1	0.47	161	19230.8	0.16	103	19230.8	0.16	12
38,400	38580.2	0.47	80	38461.5	0.16	51	41666.7	8.51	5
56,000	56818.2	1.46	54	57142.9	2.04	34	62500.0	11.61	3
115,000	115740.7	0.64	26	117647.1	2.30	16	125000.0	8.70	1
250,000	260416.7	4.17	11	250000.0	0.00	7	250000.0	0.00	0
300,000	312500.0	4.17	9	333333.3	11.11	5			
500,000	520833.3	4.17	5	500000.0	0.00	3			
1,000,000	1041666.7	4.17	2	1000000.0	0.00	1			
Min.	2.98	0.00	1048575	1.91	0.00	1048575	0.24	0.00	1048575
Max.	3125000.0	0.00	0	2000000.0	0.00	0	250000.0	0.00	0

Table 21-4. UART Baud Rates (CLKMOD = 0 and BRGS = 1)

BAUD RATE	F_{UART} = 100 MHz			F_{UART} = 80 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	2603	9601.5	0.02	2082
19,200	19201.2	0.01	1301	19212.3	0.06	1040
38,400	38402.5	0.01	650	38461.5	0.16	519
56,000	56053.8	0.10	445	56022.4	0.04	356
115,000	115207.4	0.18	216	115606.9	0.53	172
250,000	250000.0	0.00	99	250000.0	0.00	79

Table 21-4. UART Baud Rates (CLKMOD = 0 and BRGS = 1) (continued)

BAUD RATE	F_{UART} = 100 MHz			F_{UART} = 80 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
300,000	301204.8	0.40	82	303030.3	1.01	65
500,000	500000.0	0.00	49	500000.0	0.00	39
1,000,000	1000000.0	0.00	24	1000000.0	0.00	19
Min.	23.84	0.00	1048575	19.07	0.00	1048575
Max.	25000000.0	0.00	0	20000000.0	0.00	0

BAUD RATE	F_{UART} = 50 MHz			F_{UART} = 32 MHz			F_{UART} = 4 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	1301	9603.8	0.04	832	9615.4	0.16	103
19,200	19201.2	0.01	650	19230.8	0.16	415	19230.8	0.16	51
38,400	38461.5	0.16	324	38461.5	0.16	207	38461.5	0.16	25
56,000	56053.8	0.10	222	56338.0	0.60	141	58823.5	5.04	16
115,000	115740.7	0.64	107	115942.0	0.82	68	125000.0	8.70	7
250,000	250000.0	0.00	49	250000.0	0.00	31	250000.0	0.00	3
300,000	304878.0	1.63	40	307692.3	2.56	25	333333.3	11.11	2
500,000	500000.0	0.00	24	500000.0	0.00	15	500000.0	0.00	1
1,000,000	1041666.7	4.17	11	1000000.0	0.00	7	1000000.0	0.00	0
Min.	11.92	0.00	1048575	7.63	0.00	1048575	0.95	0.00	1048575
Max.	12500000.0	0.00	0	8000000.0	0.00	0	1000000.0	0.00	0

Table 21-5. UART Baud Rates (CLKMOD = 1)

BAUD RATE	F_{UART} = 100 MHz			F_{UART} = 80 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	10416	9600.4	0.00	8333
19,200	19201.2	0.01	5208	19203.1	0.02	4166
38,400	38402.5	0.01	2604	38406.1	0.02	2083
56,000	56022.4	0.04	1785	56022.4	0.04	1428
115,000	115074.8	0.07	869	115107.9	0.09	695
250,000	250000.0	0.00	400	250000.0	0.00	320
300,000	300300.3	0.10	333	300751.9	0.25	266
500,000	500000.0	0.00	200	500000.0	0.00	160
1,000,000	1000000.0	0.00	100	1000000.0	0.00	80
Min.	95.37	0.00	1048575	76.29	0.00	1048575
Max.	6250000.0	0.00	0	5000000.0	0.00	16

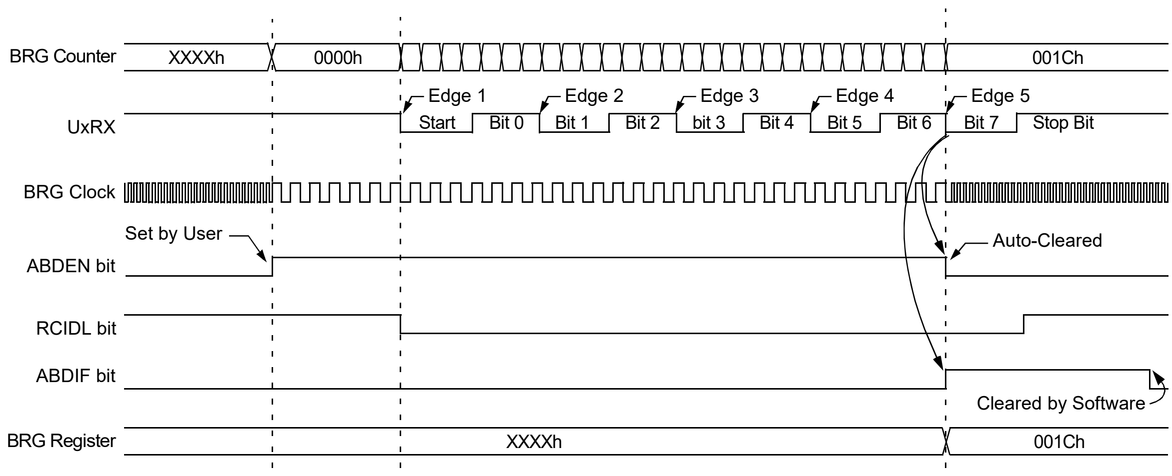
BAUD RATE	F_{UART} = 50 MHz			F_{UART} = 32 MHz			F_{UART} = 4 MHz		
	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)	Actual Baud Rate	% Error	BRG Value (Decimal)
9,600	9600.6	0.01	5208	9601.0	0.01	3333	9615.4	0.16	416
19,200	19201.2	0.01	2604	19207.7	0.04	1666	19230.8	0.16	208
38,400	38402.5	0.01	1302	38415.4	0.04	833	38461.5	0.16	104
56,000	56053.8	0.10	892	56042.0	0.08	571	56338.0	0.60	71
115,000	115207.4	0.18	434	115107.9	0.09	278	117647.1	2.30	34
250,000	1250000.0	0.00	200	250000.0	0.00	128	250000.0	0.00	16
300,000	301204.8	0.40	166	301886.8	0.63	106			
500,000	500000.0	0.00	100	500000.0	0.00	64			
1,000,000	1000000.0	0.00	50	1000000.0	0.00	32			
Min.	47.68	0.00	1048575	30.52	0.00	1048575	3.81	0.00	1048575
Max.	3125000.0	0.00	16	2000000.0	0.00	16	250000.0	0.00	16

21.4.1.4. Auto-Baud Feature

The auto-baud feature allows the receiver to determine the baud rate of the transmitter and synchronize to it. The transmitter sends a byte value of 0x55 (Sync byte) to the receiver, and the receiver calculates the average bit time from the falling edges. The UxBRG register is then written with the corresponding value. Sync byte (0x55) will not be stored in the Rx buffer. Auto-baud is supported in both Legacy and Fractional Baud Rate Generation modes (CLKMOD = `1` or `0`). The Sync byte may be preceded with a break.

To enable auto-baud, the ABDEN bit (UxCON[6]) is set and the UART will begin to look for a falling edge (Start bit of Sync byte). While the auto-baud sequence is in progress, the UART state machine is held in Idle mode. On the fifth RX pin falling edge, an accumulated BRG counter value totaling the proper BRG period is transferred to the UxBRG register. Once the auto-baud process is complete, the ABDEN bit will be cleared by hardware and the ABDIF flag (UxUIR[6]) is set. If the ABDIE (UxUIR[2]) interrupt enable bit is set, an event interrupt will be generated. See Figure 21-5 for the Auto-Baud Detection sequence.

Figure 21-5. Auto-Baud Detection



If the fifth and final falling edge is not detected before the BRG counter rolls over, the ABDOVIF flag (UxSTAT[5]) will set to indicate the condition. The flag cannot be cleared until ABDEN is cleared. If the ABDOVIE bit (UxSTAT[13]) is set, an error interrupt will be generated. For more information on interrupts, see Interrupts.

Auto-baud setup procedure:

1. Configure the UART for receive operation as detailed in Asynchronous Receive.
2. Set the ABDEN bit. If a Break precedes the Sync byte, also set the WUE (UxCON[12]) bit to configure the UART to perform the auto-baud procedure on the Sync and not the Break. The RXBKIF flag (UxSTAT[2]) will not be set.
3. Poll the ABDEN or ABDIF bit to determine when the auto-baud has finished.

Alternatively, if a Break precedes the Sync and it is desired to detect the Break, use the following sequence:

1. Configure the UART for receive operation as detailed in Asynchronous Receive.
2. Wait for the RXBKIF flag to set (see Break Character Reception for details).
3. Immediately set the ABDEN bit.
4. Poll the ABDEN or ABDIF bit to determine when the auto-baud has finished.

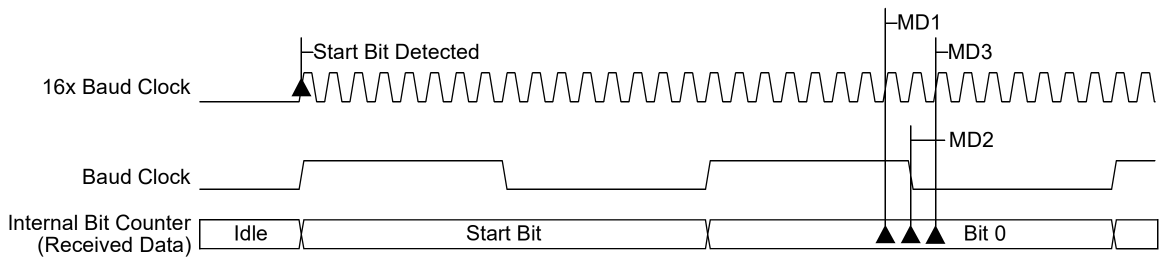
21.4.1.5. Data Bit Detection

21.4.1.5.1. Legacy Mode

Low-Speed Mode (CLKMOD = 0 AND BRGS = 0)

In Low-Speed mode, each bit of the received data is 16 clock pulses wide. To detect the value of an incoming data bit, the bit is sampled at the seventh, eighth and ninth rising edges of the clock. These rising edges are called Majority Detection (MD) edges. This mode is more robust than High-Speed mode.

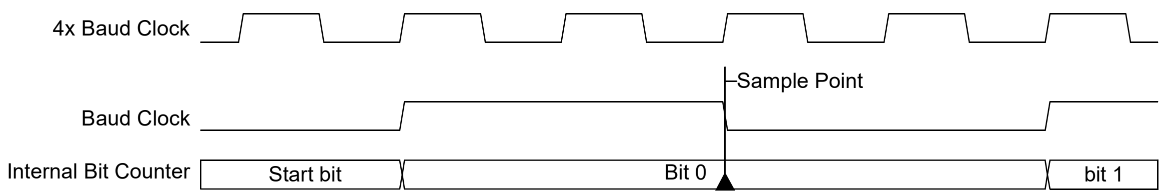
Figure 21-6. Low-Speed Mode with Majority Detection



High-Speed Mode (CLKMOD = 0 and BRGS = 1)

In High-Speed mode, each bit of the received data is four clock pulses wide. This mode does not provide enough edges to support the Majority Detection method. Therefore, the received data are sampled at the one-half bit width.

Figure 21-7. High-Speed Mode without Majority Detection



21.4.1.5.2. Fractional Division Mode (CLKMOD = 1)

In Fractional Division mode, each bit of the received data is 16 clock pulses wide. To detect the value of an incoming data bit, the bit is sampled at the seventh, eighth and ninth rising edges of the clock. Refer to Figure 21-4.

21.4.2. UART Modes

21.4.2.1. Asynchronous Mode

Asynchronous mode supports standard UART communication with the following configurable options:

- 7, 8-Bit Data Width
- 1, 1.5 or 2 Stop Bits
- Even, Odd or No Parity (Ninth data bit)
- Independently Selectable TX and RX Polarity
- Address Detect (Ninth data bit)
- Auto-Baud
- Break Transmission/Detection
- Flow Control (XON/XOFF and HW)
- Half/Full-Duplex TX Pin Control
- TX and RX Interrupt Configuration

The MODE[3:0] bits (UxCON[3:0]) are used to select the High-Level Operational mode of the UART for both transmit and receive. The five Asynchronous mode selections configure data width, parity and address detect, with the rest of the configuration options as spate controls.

21.4.2.1.1. Asynchronous Transmit

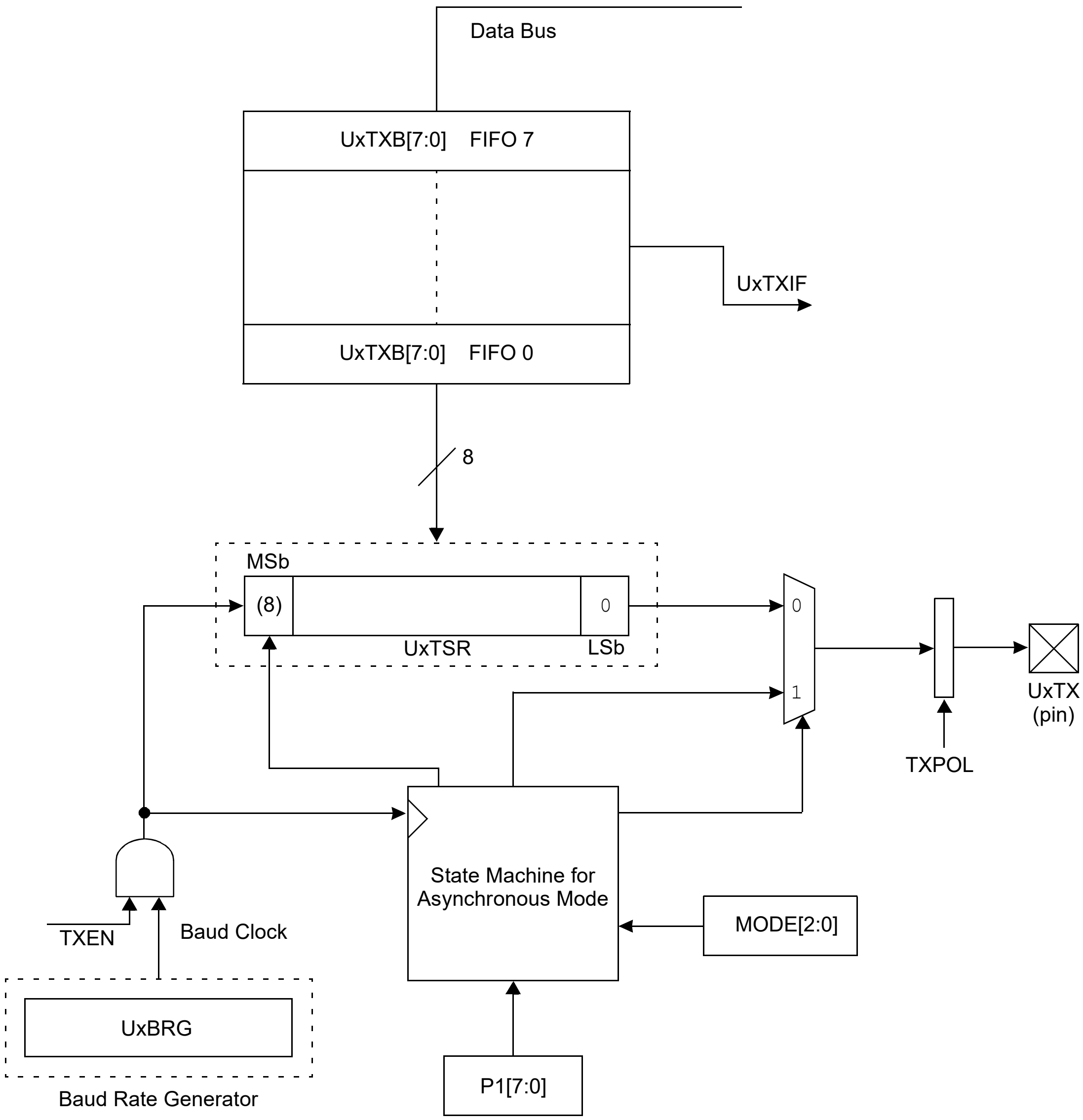
The transmitter block diagram of the UART module is illustrated in Figure 21-8. The important part of the transmitter is the UARTx Transmit Shift Register (UxTSR). The Shift register obtains its data from the transmit FIFO buffer, UxTXB. The UxTXB register is loaded with data in software.

The UxTSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the UxTSR is loaded with new data from the UxTXB register, if available.

Note: The UxTSR register is not mapped in data memory, so it is not available to the user application.

The transmission is enabled by setting the TXEN enable bit (UxCON[5]). The actual transmission will not occur until the UxTXB register has been loaded with data and the Baud Rate Generator (UxBRG) has produced a shift clock (Figure 21-8). Normally, when the first transmission is started, the UxTSR register is empty, so a transfer to the UxTXB register will result in an immediate transfer to UxTSR when the TXEN bit is set. When the TXEN bit is written to '0', the transmit process ends at the end of the current byte. As a result, the UxTX pin will revert to a High-Impedance state.

Figure 21-8. Asynchronous Transmitter Block Diagram



Note: 'x' denotes the UART number.

Setup for UART Transmit

The following procedure is used to transmit a byte of data:

1. Configure the clock input and baud rate as detailed in Clocking and Baud Rate Configuration.
2. Configure the data width and parity by writing a selection to the MODE[3:0] bits.
3. Configure the polarity, Stop bit duration and flow control.
4. Configure the TX interrupt watermark using the TXWM[2:0] bits (UxSTAT[30:28]).
5. Configure the address detect if needed as detailed in Address Detect.
6. Set the ON bit (UxCON[15]).
7. Set the TXEN bit (UxCON[5]).
8. Write the data byte value to the UxTXB register.

A TX interrupt will be generated according to the TXWM[2:0] bits' interrupt watermark setting. The TXWMx bits can be configured to generate a TX interrupt when the buffer has one to eight empty slots.

The UARTx Transmit Buffer (UxTXB) has two associated flags to indicate its contents. The TX Buffer Empty Status bit, TXBE (UxSTAT[21]), indicates that the buffer is empty, and the TX Buffer Full Status bit, TXBF (UxSTAT[20]), indicates that there are no empty slots in the buffer and it should not be written to.

Example 21-1. UART1 Transmission with Interrupts

```
#include <xc.h>

//Clock settings for CPU @8MHz, UART1 clocked from 1/2 speed peripheral bus @4MHz
#define FUART 4000000
#define BAUDRATE 9600
//Baud rate calculation for fractional baud rate divider
#define BRGVAL (FUART/BAUDRATE)

#define TRANSMIT_CHAR_SIZE 16
uint8_t transmitChar[TRANSMIT_CHAR_SIZE] = "ABCDEFGHIJKLMNOP";
uint8_t transmitCount = 0;


int main(void) {

    //Configure I/O
    _RP114R = _RPOUT_U1TX;   //Assign UART1 TX output functionality to RP114 (RH1)
    _TRISH1 = 0;             //Set RH1 as output

    U1CONbits.MODE = 0;      // Asynchronous 8-bit UART
    U1CONbits.CLKSEL = 0;    // FPB/2 as Baud Clock source
    U1CONbits.STP = 0;       // 1 stop bit

    //Use fractional baud rate divider
    U1CONbits.CLKMOD = 1;
    U1BRG = BRGVAL;         // Baud Rate setting for 9600

    U1STATbits.TXWM = 0;   // Interrupt when TX buffer empty (8 empty slots)
    IEC3bits.U1TXIE = 1;   // Enable Transmit interrupt

    U1CONbits.ON = 1;      // Enable UART
    U1CONbits.TXEN = 1;    // Enable UART TX. This generates TX interrupt.

    while(1) {
        //Re-transmit periodically
        if (IEC3bits.U1TXIE == 0) { //Check if TX interrupt was disabled
            //Delay
            for (uint16_t i = 0; i < 0x1234; i++);
            //Re-enable TX interrupt to resume transmission
            IEC3bits.U1TXIE = 1;
        }
    }

    return 0;
}

void __attribute__((interrupt)) _U1TXInterrupt(void)
{
    IFS3bits.U1TXIF = 0;                   // Clear TX interrupt flag
    U1TXB = transmitChar[transmitCount++]; // Transmit one character

    if (transmitCount >= TRANSMIT_CHAR_SIZE) {
        transmitCount = 0;
        //Stop transmitting by disabling the TX interrupt.
        IEC3bits.U1TXIE = 0;
    }
}
```

Transmit Errors and Events

The UART is capable of detecting bus collisions. The received byte is compared against the last byte transmitted to identify differences. The UxTX and UxRX pin functions need to be mapped to separate pins using Peripheral Pin Select (PPS). The UxRX pin has to be able to receive a byte in order for the comparison to happen. If the pin is stuck at Vdd or ground, such that a valid Start and Stop bit are not detected, the comparison cannot take place. If a bus collision is detected, it is flagged by the TXCIF bit (UxSTAT[0]). If the TXCIE bit (UxSTAT[8]) is set, an error interrupt will be generated.

If a write to UxTXB is done when the buffer is already full, a transmit write error is indicated by the TXWRE bit (UxSTAT[23]).

The Transmit Shift Register (TSR) has a status flag, TXMTIF (UxSTAT[7]), associated with it to indicate when a byte transmission is complete. An interrupt can be generated by setting the TXMTIE bit (UxSTAT[15]).

Half-Duplex Transmit

In a half-duplex application, the UxTX and UxRX lines are shorted together; this allows a reduction in wire count. However, control is needed to avoid both devices transmitting at the same time. Setting the HALFDPLX bit (UxCON[24]) configures the UxTX pin to only drive the line during a byte transmission and is tri-stated at all other times. In UART systems, tri-state is not a permissible state, so it is important that the user enable a weak pull-up on the UART pad when such half-duplex systems are used. The RCIDL bit (UxSTAT[19]) can be read to determine if the line is Idle and a byte can be sent. However, a collision can still occur during the transmission. The Transmit Collision Interrupt Flag bit, TXCIF (UxSTAT[0]), can be read to determine if the byte was transmitted successfully. The receiver has to remain enabled for the transmission collision to be detected. If TXCIE (UxSTAT[8]) is set, the receive watermark interrupt will not be generated when TXCIF (UxSTAT[0]) is set.

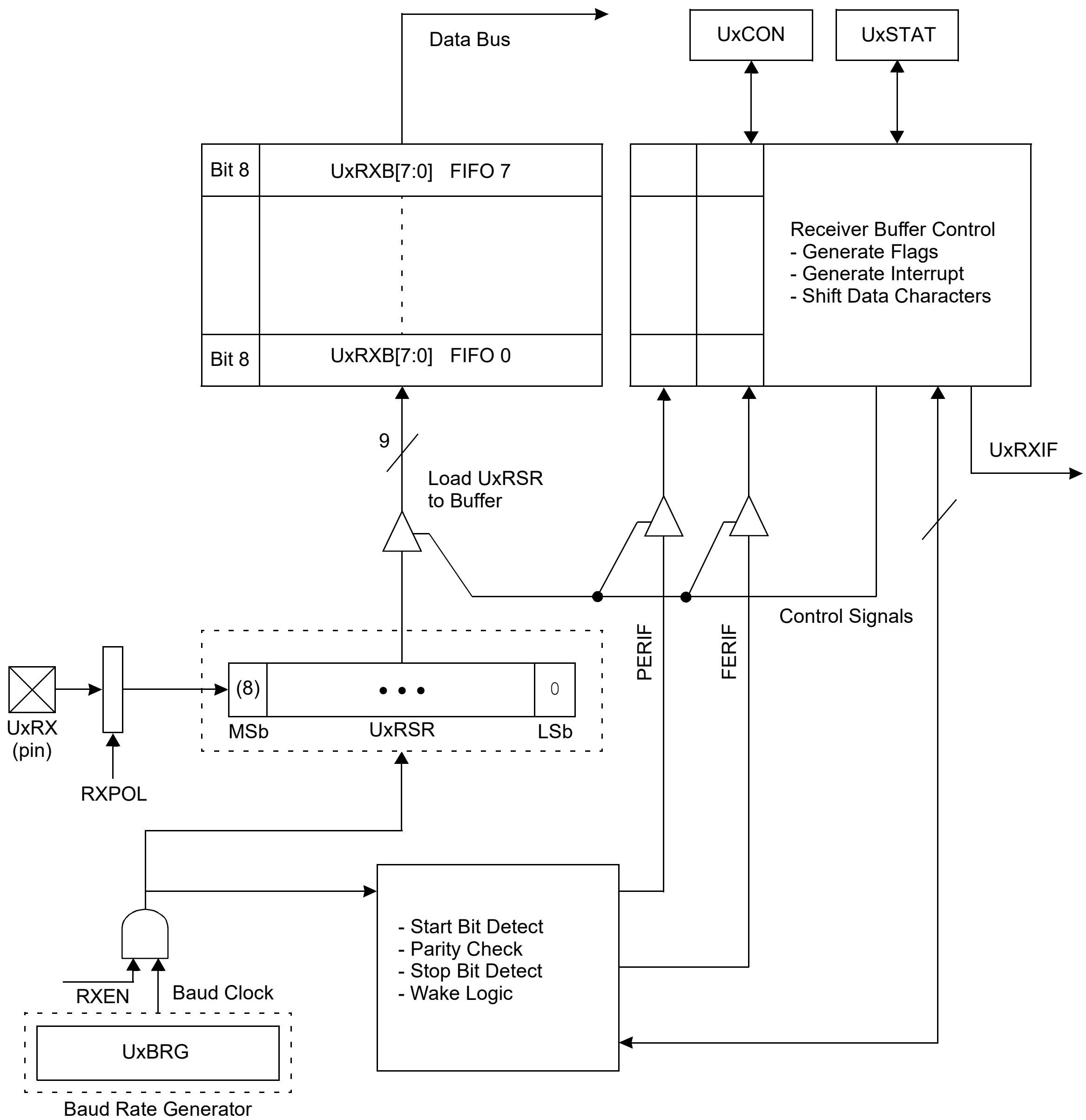
21.4.2.1.2. Asynchronous Receive

The receiver block diagram of the UART module is illustrated in Figure 21-9. The important part of the receiver is the UARTx Receive (Serial) Shift Register (UxRSR). The data are received on the UxRX pin. After sampling the UxRX pin for the Stop bit, the received data in the UxRSR register are transferred to the receive FIFO, if it is empty.

Note: The UxRSR register is not mapped in data memory, so it is not available to the user application.

The reception is enabled by setting the RXEN bit (UxCON[4]). Clearing the RXEN bit causes the receive shifter to stop. As a consequence, RXIDL (UxSTAT[19]) is set to '1'.

Figure 21-9. Asynchronous Receiver Block Diagram



Note: 'x' denotes the UART number.

Setup for UART Receive

The following procedure is used to receive a byte of data:

1. Configure the clock input and baud rate as detailed in Clocking and Baud Rate Configuration.
2. Configure the data width and parity by writing a selection to the MODE[3:0] bits.
3. Configure the polarity, Stop bit duration and flow control.
4. Configure the RX interrupt watermark using the RXWM[2:0] bits (UxSTAT[26:24]).
5. Configure the address detect if needed as detailed in Address Detect.
6. Set the ON bit (UxCON[15]).
7. Set the RXEN bit (UxCON[4]).

An RX interrupt will be generated when a byte is received, according to the UART Receive Interrupt Select bits setting, RXWM[2:0] (UxSTAT[26:24]). The RXWMx bits can be configured to generate an RX interrupt when the RX buffer contains one to eight bytes.

Software can then read the data from the UxRXB register. The time, relative to the Stop bit when the RX interrupt is generated, is configurable using the STPMD bit (UxSTAT[22]). By default, an RX interrupt is generated in the middle of the Stop bit. Writing a '1' will move the RX interrupt to the end of the Stop bit.

The RXBF status bit (UxSTAT[16]) can be read by software to determine if the receive buffer is full, and a read operation of UxRXB is required to allow reception of additional bytes. Similarly, the RXBE status bit (UxSTAT[17]) can be read with software to determine if the receive buffer is empty.

Example 21-2. UART1 Reception with Interrupts

```
#include <xc.h>

//Clocking based on CPU @8MHz, UART1 running on 1/2 speed peripheral bus @4MHz
#define FP 4000000
#define BAUDRATE 9600
//Baud rate calculation for fractional baud rate mode
#define BRGVAL (FP/BAUDRATE)

#define RECEIVED_CHAR_SIZE 16
uint8_t receivedChar[RECEIVED_CHAR_SIZE];
uint8_t receivedCount = 0;

int main(void) {

        //Configure I/O
    _U1RXR = 50; //Assign RP50 (RD1) to UART1 RX input function
    _TRISD1 = 1; //Set RC6 as input

    U1CONbits.MODE = 0;                // Asynchronous 8-bit UART
    U1CONbits.CLKSEL = 0;              // 1/2 speed peripheral clock as Baud Clock source
    U1CONbits.STP = 0;                 // 1 stop bit

    //Use fractional baud rate divider
    U1CONbits.CLKMOD = 1;
    U1BRG = BRGVAL;                    // Baud Rate setting for 9600

    U1STATbits.RXWM = 0;               // Interrupt when there is one word or more in buffer
    IEC3bits.U1RXIE = 1;               // Enable Receive interrupt

    U1CONbits.ON = 1;                  // Enable UART
    U1CONbits.RXEN = 1;                // Enable UART RX.

    while(1) {}

    return 0;
}

void __attribute__((interrupt)) _U1RXInterrupt(void)
{
    IFS3bits.U1RXIF = 0;               // Clear RX interrupt flag
    while(U1STATbits.RXBE == 0)        // Check if RX buffer has data to read
    {
        receivedChar[receivedCount++] = U1RXB; // Read a character from RX buffer

        //Avoid buffer overrun
        if (receivedCount >= RECEIVED_CHAR_SIZE) {
            receivedCount = 0;
        }
```

```
        }
}
```

Receive Errors and Events

The receive framing and parity errors are associated with each byte received. Frame and parity error flags, indicated by the FERIF and PERIF bits, will indicate only the error status of the last data byte received. As UxRXB is read, the flags indicate the status of the current (top) byte in the buffer. This behavior differs from prior UART modules.

If a byte is received when the receive buffer is full, the RXFOIF bit (UxSTAT[1]) will set. Setting the corresponding error interrupt enable will generate an error interrupt. To clear the RXFOIF bit, the receive buffer needs to be read at least once. This behavior differs from prior UART modules. The receiver can handle overflow conditions in one of two options, defined by the RUNOVF (UxCON[23]) bit. By default, when RUNOVF = `0`, the receiver will stop receiving data when the RX buffer is full. Alternatively, when RUNOVF = `1`, the receiver will continue to receive data and overwrite the contents of the RX shifter.

A line Idle condition (line high) is indicated by the RCIDL bit (UxSTAT[19]). The flag will clear when a Start bit is detected and a reception is in progress.

21.4.2.1.3. Parity Support

Parity is a simple method of single-bit error detection. The data bits are summed and compared to the parity value indicating a bit error. Parity selection can either be even or odd and is represented by the ninth data bit. To calculate parity, the number of data bits that are a ‘`1`’ are counted.

- Even parity is defined as an odd number of data bits whose values are ‘`1`’.
- Odd parity is defined as an even number of data bits whose values are ‘`1`’.

The parity bit itself is then added to the count, hence the even or odd designation. Parity calculation and checking are enabled by selecting one of the two 8-Bit Asynchronous Parity modes (MODE[3:0] = `0b001x`).

21.4.2.1.4. Break Character

A Break character is defined as several consecutive low-bit times, usually longer than a whole byte. In Asynchronous mode, the UART will transmit a 13-bit long duration Break, and in receive, will flag 11 low-bit times as a Break sequence.

Transmitting a Break Character

A Break character is transmitted by setting the SENDB bit (UxCON[8]) and then writing any value to UxTXB. The contents of UxTXB will follow the Break character.

Alternatively, the BRKOVR bit (UxCON[9]) can be controlled by software to override and drive the UxTX line for any duration. When TXPOL (UxCON[18]) = `0`, the UxTX line will be driven low, and when TXPOL = `1`, the UxTX line will be driven high.

Break Character Reception

The receiver is always looking for a Break sequence and can detect one, even in the middle of a byte reception. A Break reception is indicated by the RXBKIF flag (UxSTAT[2]). An interrupt can be optionally generated by setting the RXBKIE bit (UxSTAT[10]). The Break detection criteria can be configured using the RXBIMD bit (UxCON[11]). By default, the RXBKIF flag will set when the line makes a low-to-high transition after 11 low-bit times, signaling the end of the Break sequence. Alternatively, when RXBIMD = `1`, the flag will set when the eleventh low-bit time is detected.

21.4.2.1.5. Checksum

For LIN mode, two kinds of checksums are available: legacy and enhanced. In the legacy checksum, only data bytes D0 through D7 are used to calculate the checksum. In the enhanced checksum, data bytes D0 through D7 and PID[5:0], P0 and P1 are included. The type of checksum used in the calculation can be controlled by software using the C0EN bit. Refer to LIN/J2602 for more information on the checksum calculation.

For all other modes, the C0EN bit is ignored, and the UART calculates the checksum for every transmitted or received byte. The checksum register UxCHK is cleared upon receiving a Break sequence in all protocol modes. These registers can also be cleared by the user.

21.4.2.1.6. Address Detect

Address Detect mode is used when multiple receivers are connected to a transmitter. It allows a receiver to determine if the message is intended for it and to ignore those that are not. If the ninth data bit is a '1', the data are recognized as addresses to be processed by the receiver. An address mask is provided to allow multiple receivers to accept the same address.

If an address match is successful, the unmasked address is present in UxRXB and an RX interrupt is generated. If the address match is not successful, all of the following data are ignored until a byte with the ninth bit set is received.

In 8-Bit Address Detect mode, the transmitted address IDs are written to Parameter 1. For receivers, the expected address is written to Parameter 2 (P2[7:0]) and the mask value to Parameter 3 (P3[7:0]). Mask bit values of '1' will include the respective bit position in the compare, whereas a '0' indicates a don't care. A mask value of 0x00 will accept all address values, effectively disabling the address detect feature. A mask value of 0xFF will allow only one matching value.

Address Detect Transmit

The following procedure is used to transmit in Address Detect mode:

1. Configure the UART for asynchronous transmit as detailed in Asynchronous Transmit with the MODE[3:0] bits set to '0b0100'.
2. If a Break is desired, write a '1' to SENDB (UxCON[8]).
3. Write the address value to Parameter 1.
   a. If SENDB = 0, the contents of Parameter 1 will be transmitted with the ninth bit set.
   b. If SENDB = 1, a Break will be transmitted, followed by the contents of Parameter 1 with the ninth bit set.
4. Write data to be transmitted to the UxTXB register.

Address Detect Receive

The following procedure is used for receive in Address Detect mode. A framing error will not prevent an address match.

1. Configure the UART for asynchronous receive as detailed in Asynchronous Receive with the MODE[3:0] bits set to '0b0100'.
2. Write the address match value to Parameter 2.
3. Write the optional address mask value to Parameter 3.
4. Upon the reception of a valid address, the PERIF bit will be set to indicate that the value stored in UxRXB is an address. The subsequent data can be read from UxRXB as they become available.

21.4.2.1.7. Flow Control

Flow control is used to prevent data loss between two devices. One device may be slower or have to process data. Flow control allows a device to tell the other to wait before sending additional bytes that may overrun its buffers. The UART supports two types of flow control:

- XON/OFF Messaging
- Hardware Flow Control ($\overline{\text{UxRTS}}$, $\overline{\text{UxCTS}}$, $\overline{\text{UxDTR}}$, $\overline{\text{UxDSR}}$)

XON/XOFF

XON/XOFF uses messages implemented as special byte values and does not require additional HW lines. An XON command is implemented by sending a byte value of 0x11, and an XOFF is implemented by a value of 0x13. There are two states of the control mechanism as indicated by the XON bit (UxSTAT[18]). By default, the UART is in the XON = 1 state and will transmit data as they become available in the TX buffer. If the device receives an XOFF command, the XON status bit is cleared and transmission stops until another XON command is received. XON/OFF commands are transmitted in the same manner as regular data.

The receiver keeps track of the UART buffer; if the RX buffer is left with two empty slots, an XOFF command is sent by the module automatically. After that, the module sends an XON command if more than two empty slots become available in the RX buffer.

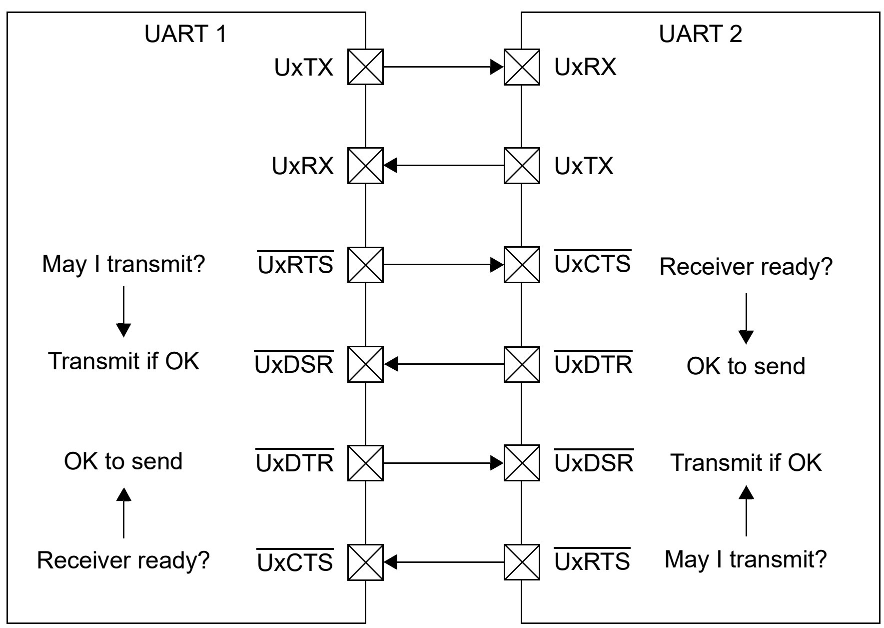
Hardware Flow Control

Hardware flow control uses up to four additional pins to indicate when a device is ready to receive additional data. The four active-low device pins are shown in Table 21-6. Figure 21-10 shows connections between two UARTs.

Table 21-6. Hardware Flow Control Pin Functions

Signal Name	Description	Used By	Direction
$\overline{\text{UxDSR}}$	Data-Set-Ready	Transmitter	Input
$\overline{\text{UxRTS}}$	Request-to-Send	Transmitter	Output
$\overline{\text{UxCTS}}$	Clear-to-Send	Receiver	Input
$\overline{\text{UxDTR}}$	Data-Terminal-Ready	Receiver	Output

Figure 21-10. Hardware Flow Control Pins and Connections



The transmitter asserts (drives low) the $\overline{\text{UxRTS}}$ output when it has one or more bytes in its TX buffer to indicate that it wants to send a byte. Then, the transmitter listens to the $\overline{\text{UxDSR}}$ to see if it is OK to do so. If $\overline{\text{UxDSR}}$ is active (low), the transmitter sends one byte. If $\overline{\text{UxDSR}}$ is inactive (high), it will wait.

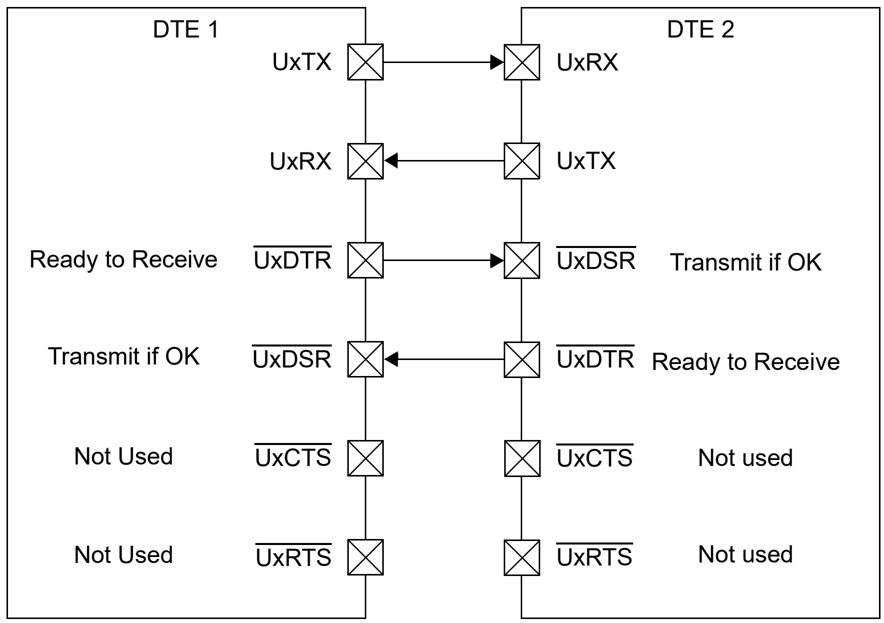
When the receiver detects the $\overline{\text{UxCTS}}$ signal going active (low), it checks to see if there are two empty slots in the receive buffer. If so, the receiver asserts (drives low) the $\overline{\text{UxDTR}}$ pin to indicate it is ready to receive data. No register setup is needed to enable the flow control pins. However, most devices will have the UART and associated flow control pins routed through the Peripheral Pin Select (PPS) feature.

When using flow control, devices can be categorized into two groups: DTE (Data Terminal Equipment) and DCE (Data Carrier Equipment). A typical DTE can be a computer or a microcontroller, and a DCE is typically a modem. Not all hardware flow control pins are needed in all cases. The following sections show which flow control pins are used to interface different devices to one another.

DTE to DTE Configuration

When interfacing two DTE devices together, connect them as shown in Figure 21-11. The $\overline{\text{UxDTR}}$ output is connected to the $\overline{\text{UxDSR}}$ input terminal of the other. This allows the receiver to tell the transmitter that it is OK to transmit.

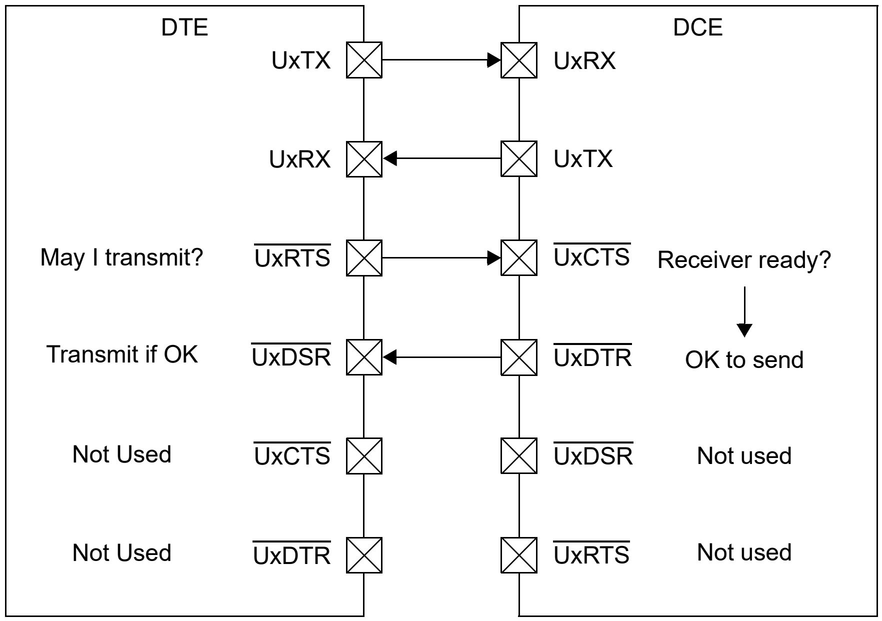
Figure 21-11. DTE to DTE Pin Connections



DTE to DCE Configuration

When interfacing a DTE device to a DCE device, connect them as shown in Figure 21-12. The $\overline{\text{UxRTS}}$ output of the DTE is connected to the $\overline{\text{UxCTS}}$ input terminal of the DCE, and the $\overline{\text{UxDTR}}$ output of the DCE is connected to the $\overline{\text{UxDSR}}$ input of the DTE. This allows the DCE to tell the DTE when it is ready to receive data.

Figure 21-12. DTE to DCE Pin Connections



21.4.2.2. LIN/J2602

The UART provides support for the Local Interconnect Network (LIN) protocol for both Commander and Responder processes to reduce software overhead. The LIN protocol is typically used in automotive applications and packages bytes into message frames. The LIN protocol has two types of processes: Commander and Responder. A network can have only one Commander and multiple Responders. The Commander process transmits a header containing a command that the Responder(s) can respond to. The Commander process, part of a LIN message frame, consists of the following:

1. Break character (11 bits minimum received, 13 transmitted).
2. Delimiter bit.
3. Sync byte (0x55).
4. Protected ID (PID) field.

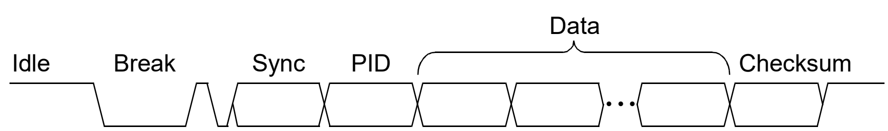
The Responder processes and then completes the message frame by transmitting the requested data and checksum.

1. Data (up to eight bytes).
2. Checksum.

The UART has two LIN modes, Commander/Responder and Responder Only, selected by the MODE[3:0] bits (UxCON[3:0]). The Commander/Responder mode allows a single instance of a UART to handle both Commander and Responder software processes.

A LIN frame starts with the Commander process sending a Break, followed by a Sync to allow the receiver to synchronize the baud rate with the transmitter. The PID byte follows and is used by the Responder to determine if, or how, to respond. A Responder process then responds with up to eight bytes of data and a checksum. A LIN frame is shown in Figure 21-13.

Figure 21-13. LIN Frame



The PID byte consists of six bits of data and two parity bits, P0 followed by P1. The PID value is written to Parameter 1 (P1[5:0]), and the parity bits are automatically calculated. Parameter 1 can only be written when the transmitter is Idle. The parity bits are calculated as follows:

P0 = PID[0] XOR PID[1] XOR PID[2] XOR PID[4]

P1 = NOT (PID[1] XOR PID[3] XOR PID[4] XOR PID[5])

The UART automatically calculates the checksum. Two types of LIN checksums are supported and selected by the C0EN bit (UxCON[19]). When C0EN = `0` (default), the legacy LIN checksum method is used, which uses only data bytes. When C0EN = `1`, the checksum also includes the PID. The checksum is calculated by adding the number of data bytes, defined by Parameter 2, adding the carry result, and finally inverting the sum.

Table 21-7 provides an example checksum calculation for a LIN frame of four data bytes in length, with data values of 0x4A, 0x55, 0x93 and 0xE5.

Table 21-7. LIN Checksum Example (C0EN = 1 or 0)

Action	Hex	Carry	D7	D6	D5	D4	D3	D2	D1	D0
0x4A	0x4A		0	1	0	0	1	0	1	0
+0x55 Add Carry	0x9F 0x9F	0	1 1	0 0	0 0	1 1	1 1	1 1	1 1	1 1

Table 21-7. LIN Checksum Example (C0EN = 1 or 0) (continued)

Action	Hex	Carry	D7	D6	D5	D4	D3	D2	D1	D0
+0x93 Add Carry	0x132 0x33	1	0 0	0 0	1 1	1 1	0 0	0 0	1 1	0 1
+0xE5 Add Carry	0x118 0x19	1	0 0	0 0	0 0	1 1	1 1	0 0	0 0	0 1
Invert	0xE6(1)		1	1	1	0	0	1	1	0
Receiver Verification										
Check Local + Received	0x19(2)+0xE6(1)									

Notes:

1. This is the checksum value transmitted as the last byte.
2. This is the checksum value calculated by the receiver.

For a transmit and receive operation, the calculated checksum is stored in UxCHK.

21.4.2.2.1. LIN Commander/Responder Transmit

The following procedure is used for Commander/Responder transmit:

1. Configure the clock input and baud rate as detailed in Clocking and Baud Rate Configuration.
2. Configure LIN mode by writing '`0b1100`' to the MODE[3:0] bits.
3. Configure the checksum type by writing the C0EN bit.
4. Set the ON, RXEN and TXEN bits.
5. Write the 6-bit PID value to Parameter 1 (P1[5:0]).

Writing to Parameter 1 will cause the Break, Sync and PID to be transmitted. If it is desired to complete the message frame in Commander/Responder mode, the following steps are used:

1. Wait for the RXBKIF bit to be set.
2. Write the number of bytes to transmit to Parameter 2 (P2[7:0]).
3. Write data to transmit to the UxTXB register.

Note: To write to P2 without affecting P1, use UxPAbits.P2 = value. Writing to the entire UxPA register overwrites P1 and potentially causes unexpected behavior in LIN mode. When accessing through pointers, ensure a 16-bit pointer targets P2's base address for writing.

21.4.2.2.2. LIN Responder Only Receive

The Responder is typically listening for a PID that it should respond to. Reception of a Break resets the checksum, parity calculation and contents of Parameter 3. The baud rate is automatically calculated and written to UxBRG. A Break after the auto-baud sequence starts will not be detected. Either not enough edges will be received and the BRG will overflow or the auto-baud sequence will complete on edges following the Break, resulting in a wrong baud rate value. The following procedure is used for Responder Only receive:

1. Configure LIN Responder Only mode by writing '`0b1011`' to the MODE[3:0] bits.
2. Set the ON and RXEN bits.

Upon reception of the Break, the RXBKIF flag is set. Upon reception of the PID, an RX interrupt is generated regardless of the RXWM[2:0] bits setting.

1. The PID can be read from UxRXB. If a parity mismatch (P0 and P1) has occurred, the PERIF flag will be set.
2. Write the number of bytes to receive to Parameter 3 (P3[7:0]).
3. Configure the RX watermark interrupt setting using the RXWM[2:0] bits.

4. Read data from UxRXB upon an interrupt event.

The UART automatically verifies the checksum. The checksum that the receiver calculates is stored in the RXCHK[7:0] bits (UxCHK[23:16]). If this value doesn't match that of the received checksum, the CERIF flag (UxSTAT[4]) will be set, and if the CERIE (UxSTAT[12]) bit was set, an interrupt will be generated.

21.4.2.2.3. LIN Responder Only Transmit

A LIN Responder Only transmit is a response to a header sent by the Commander and is preceded by a receive event. The baud rate is already determined by the reception of the Sync field. The following procedure is used for Responder Only transmit:

1. Configure LIN Responder Only mode by writing '`0b1011`' to the MODE[3:0] bits.
2. Configure the checksum type by writing C0EN.
3. Set the ON, RXEN and TXEN bits.

Upon reception of the Break, the RXBKIF flag is set. Upon reception of the PID, an RX interrupt is generated regardless of the RXWM[2:0] bits setting.

1. The PID can be read from UxRXB. If a parity mismatch (P0 and P1) has occurred, the PERIF flag will be set.
2. Write the number of bytes to transmit to Parameter 2 (P2[2:0]).
3. Load UxTXB with data to transmit.

After a Sync is received, writing to UxTXB will cause the data and checksum to be transmitted. A TX interrupt will indicate transmission according to the TXWM[2:0] bits setting.

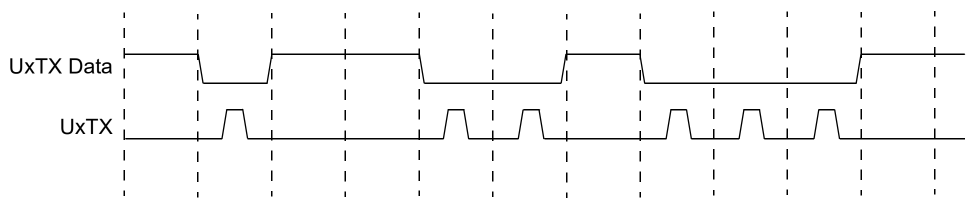
21.4.2.3. IrDA®

The UART supports the infrared IrDA protocol with a built-in encoder/decoder, so an external codec is not required. Each bit time is divided into 16 clock cycles; therefore, the clock prescaler BRGS is forced to '0', and Equation 21-1 is used for the baud rate calculation. The IrDA encoding/decoding consists of the following translations:

- A value of '1' is translated to '0' for all of the 16 clocks.
- A value of '0' is translated to '0' for the first seven clocks, '1' for the next three, and '0' for the remaining six clocks.

To enable IrDA, set MODE[3:0] = `0b1110` (UxCON[3:0]). Typical IrDA requires an active-low Idle; therefore, it is recommended to operate the UART with both TXPOL and RXPOL set to '1'. This allows the UxTX and UxRX pins to connect directly to an infrared transceiver. Figure 21-14 shows an example of an IrDA waveform.

Figure 21-14. IrDA® Encoding/Decoding When TXPOL = 1



To configure the UART for IrDA mode, the following sequence is used:

1. Configure the clock input and baud rate as detailed in Clocking and Baud Rate Configuration.
2. Configure IrDA mode by writing '`0b1110`' to the MODE[3:0] bits.
3. Set the TXPOL and RXPOL bits.
4. Configure the interrupt watermark using the TXWM[2:0] and RXWM[2:0] bits.
5. Set the ON bit.
6. Set the RXEN and TXEN bits.

In IrDA mode, the UART functions similarly to Asynchronous mode regarding errors, events and interrupts.

21.4.2.4. Smart Card

The UART module supports communication with ISO 7816 smart cards. In a typical application, the UART module is intended to act as the host or terminal that always initiates communication transactions. The smart card acts as a client and always responds to commands and other stimuli from the terminal. Figure 21-15 shows a smart card subsystem using a microcontroller with a UART module for smart card data communication.

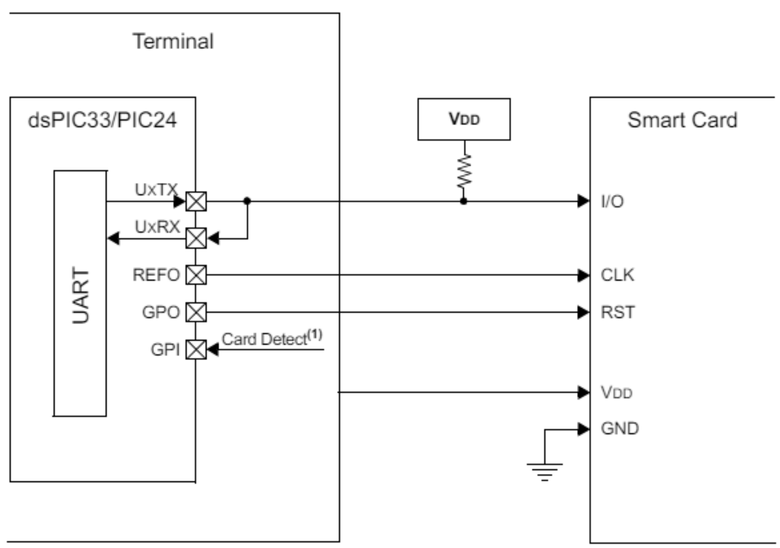
The terminal is also responsible for powering, clocking and resetting the smart card. The clock can be sourced by using the REFO output pin, and the Reset signal can be implemented with a general purpose output. The system is based on a half-duplex single wire, requiring the UART UxTX and UART UxRX pins to be shorted externally and pulled to V_{DD} with a weak pull-up.

The module can be configured to support either block (T = `1`) or byte (T = `0`) protocol. Block mode is set up for a predetermined message block size, whereas Byte mode transmits one byte at a time.

Upon detection of the card insertion, the terminal pulls the Reset line low to initiate a Reset sequence. The smart card responds with an Answer-to-Reset (ATR), which contains parameters used for communication details. The ATR baud rate is predetermined at the REFO clk/372. The terminal will need to be configured for this baud rate at the time the Reset pulse is sent to the smart card. Typical REFO clock rates are 1 MHz to 5 MHz. See ISO 7816 for additional details on ATR.

Note: Protocol characteristics, electrical characteristics of the smart card, Answer-To-Reset (ATR), PPS (Protocol Parameter Selection), calculation of guard time and wait times are out of the scope for this data sheet. Please refer to the licensed version of the ISO 7816-3 document for details about smart card communication.

Figure 21-15. Smart Card Subsystem



Note: Use a general purpose input to detect insertion of a smart card.

21.4.2.4.1. Protocol and Frame Details

The smart card communication scheme is based on an Elementary Time Unit (ETU) that is also the bit clock. The smart card will provide the ETU value in the ATR, and the terminal is configured accordingly. A character frame consists of ten bits, a Start bit, eight data bits and a Parity bit. Depending on the mode, guard and wait times are used to separate bytes and message transitions.

The ISO 7816 specification defines two communication logic conventions: direct and inverse. Direct convention is defined as LSB first and a high state as logic one. Inverse mode is defined as MSB first, and a low on the line is interpreted as a logic high. The logic convention is set using the CONV bit (UxSCCON[3]).

Guard Time

Guard time is defined as the minimum delay between two consecutive character frames. The ISO 7816 specification defines both a Character Guard Time (CGT) and a Block Guard Time (BGT). In both T = `0` and T = `1` modes, CGT is defined as the minimum delay between the leading edges of the two consecutive characters in the same direction of transmission. Block Guard Time (BGT) for T = `1` mode only is defined as the minimum delay between the leading edges of the two consecutive characters in the opposite directions. The BGT has a standard fixed value of 22 ETUs.

Wait Time

Wait time is the maximum delay allowed between two consecutive characters transmitted by the card or an interfacing device. The Character Wait Time (CWT) is the maximum delay between the leading edges of the two consecutive characters in the block, as shown in Figure 21-16. The minimum delay is CGT. The Block Wait Time (BWT) is the maximum delay between the leading edge of the last character of the block received by the card and the leading edge of the first character of the next block transmitted by the card, as shown in Figure 21-17. BWT helps the interfacing device in detecting the unresponsive smart cards. The minimum delay is BGT.

Note: The LAST bit (UxTXB[15]) set by user software is used to automatically start the guard or wait timers, depending on the state of the module.

Figure 21-16. Character Guard and Wait Time

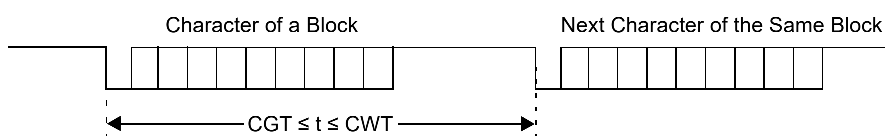
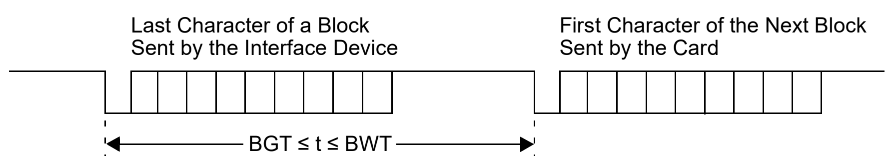


Figure 21-17. Block Guard and Wait Time

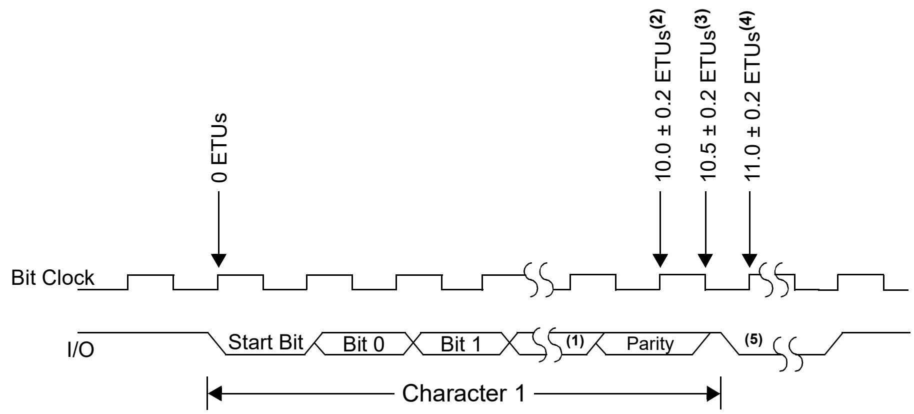


Error Detection

The transmitter is responsible for calculating the parity bit value. Parity is always even, defined as the number of logic ones and the parity bit always being an even count. The receiver also calculates the parity value and compares it to the received parity bit. If a discrepancy is found, the error is flagged by the receiver, pulling the line low for a duration defined by the T0PD bit (UxSCCON[2]).

The transmitter tests the I/O line at time 11 ± 0.2 ETUs after the leading edge of the Start bit of a character is sent. If the transmitter detects an error by detecting a low state on the I/O line, it will repeat the disputed character after a delay of at least two ETUs following the detection of the error. The number of repeats is configured with the TXRPT[1:0] bits (UxSCCON[5:4]). See Figure 21-18 for timing details in T = `0` mode.

Figure 21-18. T = 0 Character Repetition Timing



Notes:

1. 8-bit character.
2. At 10.0 ± 0.2 ETUs, the transmitter disables the driver.
3. At 10.5 ± 0.2 ETUs, the receiver sets the I/O line low if a parity error is detected.
4. At 11.0 ± 0.2 ETUs, the transmitter tests the I/O line.
5. See the T0PD bit (UxSCCON[2]) in UxSCCON.

21.4.2.4.2. Smart Card Operation

Pre-ATR Initialization

The module should be configured in Receive mode prior to the Reset line being pulled low to initiate the smart card's response as follows:

1. Write the BRG register with a value corresponding to REFO/372.
2. Configure Smart Card mode by writing '`0b1111`' to the MODE[3:0] bits.
3. Set the ON, RXEN and TXEN bits.
4. Configure the RX interrupt watermark using the RXWM[2:0] bits (UxSTAT[26:24]).
5. Read data out of UxRXB as they become available and save for ATR processing.

Post-ATR Initialization

After the terminal has done a Reset of the Smart Card and received the setup parameters contained in the ATR, the user software can configure the module for communication as follows:

1. Disable the UART for configuration changes by clearing the ON bit.
2. Set the PRTCL (UxSCCON[1]), T0PD (UxSCCON[2]), CONV (UxSCCON[3]) and TXRPT[1:0] bits (UxSCCON[5:4]) according to ATR parameters.
3. Program the UxBRG register for the ETU defined in ATR.
4. Program guard time using Parameter 1 and set the GTCIE bit (UxSCCON[16]).
5. Program the wait time using Parameter 3 and set the WTCIE bit (UxSCCON[17]).
6. Configure the RX interrupt watermark using the RXWM[2:0] bits (UxSTAT[16:24]).
7. Set the ON, TXEN and RXEN bits.

T = 0 Protocol Communication

Transmission with T = `0`:

1. Write data into UxTXB.

2. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or TXRPTIF bits are set.
3. Set LAST = 1 (UxTXB[15]) for the last byte of the data.
   Note: Due to the UxTX and UxRX pins being shorted, a receive interrupt will be generated on transmission of a character if enabled. It is recommended to disable receive interrupts when transmitting.

Reception with T = `0`:

1. Read the UxRXB as the data become available upon a receive interrupt.
2. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or RXRPTIF bits are set.
   Note: For the last character, the user must ensure that the guard time is satisfied before transmitting the response. The GTC may be used for this purpose, whereas the WTC interrupt may be disabled or ignored.

T = 1 Protocol Communication

Transmission with T = `1`:

1. Write data into UxTXB.
2. Program the value of the BWT to Parameter 2 and set the WTCIE bit (UxSCCON[17]).
3. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or TXRPTIF bits are set.
4. Set LAST = 1 (UxTXB[15]) for the last byte of the data.

Reception with T = `1`:

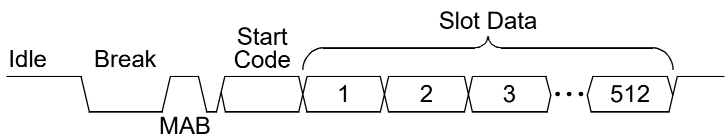
1. Read the UxRXB as the data become available upon a receive interrupt.
2. Program the value of the CWT to Parameter 3 and set the WTCIE bit (UxSCCON[17]) to ‘`1`’.
3. Take appropriate actions according to the ISO 7816 standard if the WTCIF, GTCIF and/or RXRPTIF bits are set.
   Note: For the last character, the user must ensure the guard time is satisfied before transmitting the response. The GTC may be used for this purpose, whereas the WTC interrupt may be disabled or ignored.

21.4.2.5. DMX

Digital Multiplex 512 (DMX) is a protocol used typically for stage lighting and effects. It supports a universe of up to 512 channels and is typically implemented using EIA-485 at the physical layer. DMX communication is one way only, with the controller only sending messages and the client device only receiving. There is no error checking or confirmation that a command has been received. DMX operates at a baud rate of 250k with no parity and two Stop bits. A DMX message frame consists of a header and up to 512 slots (data bytes). A client device can be configured to accept more than one slot, given start and stop assignment values.

A DMX message frame consists of a Break, a Mark After Break (MAB), a start code and finally, the slot data bytes. The MAB is three bit times in length. The start code specifies the type of data, and it is typically at 0x00. Figure 21-19 shows a DMX frame.

Figure 21-19. DMX Frame



21.4.2.5.1. DMX Transmit

The following procedure is used for DMX transmit:

1. Configure the clock input and baud rate as detailed in Clocking and Baud Rate Configuration for 250 kbaud.
2. Configure DMX mode by writing '`0b1010`' to the MODE[3:0] bits.
3. Set STP to '0b10'.
4. Configure the TX interrupt watermark using the TXWM[2:0] bits.
5. Write to Parameter 1 equal to the number of bytes – 1 (not including the start code).
6. Set the ON and TXEN bits.
7. Write the start code value to UxTXB.

Writing the start code to UxTXB will transmit a 25-bit Break, MAB and start code.

1. Write slot data bytes to UxTXB.

If not all bytes defined by Parameter 1 are written, the line will return to the Idle state. Once all bytes are written, the frame is considered complete, and the next write to UxTXB will send a Break for the next frame.

21.4.2.5.2. DMX Receive

The following procedure is used for DMX receive:

1. Configure the clock input and baud rate as detailed in Clocking and Baud Rate Configuration for 250 kbaud.
2. Configure DMX mode by writing '`0b1010`' to the MODE[3:0] bits.
3. Configure the RX interrupt watermark using the RXWM[2:0] bits.
4. Set the ON bit.
5. Set the RXEN bit.
6. Wait for the RXBKIF bit to set.
7. Write the byte start value – 1 to Parameter 2 (not including start code).
8. Write the byte end value – 1 to Parameter 3 (not including start code).

Once a Break is received, the UART will load the start code byte into UxRXB and generate an RX interrupt, regardless of the RX watermark setting (RXWM[2:0]). The range of bytes defined by

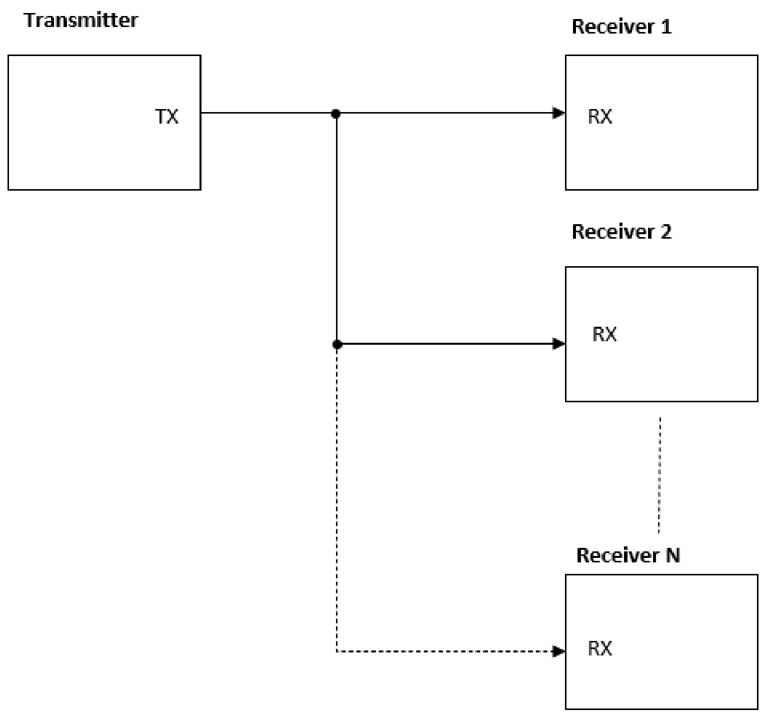
Parameter 2 and Parameter 3 is then loaded into UxRXB, and an RX interrupt is generated according to the RXWM[2:0] bits.

21.5. Application Examples

21.5.1. UART Communication Between a Transmitter and Multiple Receiver using Address Detect Feature

A transmitter is communicating with multiple receivers as shown in Figure 21-20. When a transmitter wants to send data to any one of the receivers, it will first send the receiver's address followed by the data bytes. These data bytes will be ignored by the rest of the receivers. The mask value can also be configured so that more than one receiver can receive the same data bytes.

Figure 21-20. Address Detect Block Diagram



21.5.1.1. Transmission

Example 21-3. Address Detect Transmission

```
#include <xc.h>

//Clocking based on CPU @8MHz, UART1 clocked from 1/2 speed peripheral bus @4MHz
#define FUART 4000000
#define BAUDRATE 9600
//Baud rate calculation for fractional baud rate mode
#define BRGVAL (FUART/BAUDRATE)

#define ADDRESS1 0x45
#define ADDRESS2 0x55

#define TRANSMIT_CHAR_SIZE 4

uint8_t transmitChar1[TRANSMIT_CHAR_SIZE] = "abcd";
uint8_t transmitChar2[TRANSMIT_CHAR_SIZE] = "efgh";


int main(void) {

    //Configure I/O
    _RP29R = _RPOUT_U1TX;      //Assign UART1 TX output functionality to RP29 (RB12)
    _TRISB12 = 0;                //Set RB12 as output

    U1CONbits.MODE = 4;        // Asynchronous 9-bit UART with address detect
    U1CONbits.CLKSEL = 0;      // FPB/2 as Baud Clock source
    U1CONbits.STP = 0;         // 1 stop bit

    //Use fractional baud rate divider
    U1CONbits.CLKMOD = 1;
    U1BRG = BRGVAL;            // Baud Rate setting for 9600

    U1CONbits.ON = 1;          // Enable UART
    U1CONbits.TXEN = 1;        // Enable UART TX.

    //Transmit using Address1
    U1PAbits.P1 = ADDRESS1;   // Write the address1 value to Parameter1

    /* Send data bytes for address1 */
    for (int i = 0; i < TRANSMIT_CHAR_SIZE; i++) {
        U1TXB = transmitChar1[i];
    }
    //Wait for transmission to complete
    while(U1STATbits.TXMTIF == 0);

    //Transmit using Address2
    U1PAbits.P1 = ADDRESS2; // Write the address2 value to Parameter1

    /* Send data bytes for address2 */
    for (int i = 0; i < TRANSMIT_CHAR_SIZE; i++) {
        U1TXB = transmitChar2[i];
    }
    //Wait for transmission to complete
    while(U1STATbits.TXMTIF == 0);

    //Periodically repeat the transmission
    while(1) {

        //Delay before re-transmitting
        for (int i = 0; i < 0x12345678; i++);

        U1PAbits.P1 = ADDRESS1; // Write the address1 value to Parameter1

        /* Send data bytes for address1 */
        for (int i = 0; i < TRANSMIT_CHAR_SIZE; i++) {
            U1TXB = transmitChar1[i];
        }
        //Wait for transmission to complete
        while(U1STATbits.TXMTIF == 0);

        U1PAbits.P1 = ADDRESS2; // Write the address2 value to Parameter1
        /* Send data bytes for address2 */
        for (int i = 0; i < TRANSMIT_CHAR_SIZE; i++) {
            U1TXB = transmitChar2[i];
        }
        //Wait for transmission to complete
        while(U1STATbits.TXMTIF == 0);

    }

    return 0;
}
```

21.5.1.2. Reception

Example 21-4. Address Detect Reception

```
#include <xc.h>

//Clocking based on CPU @8MHz, UART1 clocked from 1/2 speed peripheral bus @4MHz
#define FUART 4000000
#define BAUDRATE 9600
//Baud rate calculation for fractional baud rate mode
#define BRGVAL (FUART/BAUDRATE)

#define RECEIVED_CHAR_SIZE 16
uint8_t receivedChar[RECEIVED_CHAR_SIZE];
uint8_t receivedCharCount=0;

int main(void) {

    //Configure I/O
    _U1RXR = 30;             //Assign RP30 (RB13) to UART1 RX input function
    _TRISB13 = 1;             //Set RB13 as input

    U1CONbits.MODE = 4;     // Asynchronous 9-bit UART with address detect
    U1CONbits.CLKSEL = 0;   // FPB/2 as Baud Clock source
    U1CONbits.STP = 0;      // 1 stop bit

    //Use fractional baud rate divider
    U1CONbits.CLKMOD = 1;
    U1BRG = BRGVAL; // Baud Rate setting for 9600

    U1PAbits.P2 = 0x45;      // Write parameter 2 with Address to match
    U1PBbits.P3 = 0xFF;      // Write parameter 3 register with mask value

    U1STATbits.RXWM = 0;     // Interrupt when there is one word or more in the buffer
    IEC3bits.U1RXIE = 1;     // Enable Receive interrupt

    U1CONbits.ON = 1;        // Enable UART
    U1CONbits.RXEN = 1;      // Enable UART RX.

    while(1) {}

    return 0;
}

void __attribute__((interrupt)) _U1RXInterrupt(void)
    {
    IFS3bits.U1RXIF = 0;     // Clear RX interrupt flag
    while(U1STATbits.RXBE == 0) // Check if RX buffer has data to read
    {
        receivedChar[receivedCharCount++] = U1RXB; // Read a character from RX buffer

        //Wrap around array index when array is full
        if (receivedCharCount >= RECEIVED_CHAR_SIZE) {
            receivedCharCount = 0;
        }
    }
}
```

21.6. Interrupts

The UART has four separate interrupts. To determine which event has caused the interrupt, the associated flag needs to be read and evaluated. All interrupt sources are listed in Table 21-8.

Table 21-8. Interrupts

Interrupt Type	Condition	Flag
TX	Number of Empty Slots in UxTXB Defined by TXWM[2:0] bits	TXIF
RX	Number of Words in UxRXB Defined by RXWM[2:0] bits Address Match	RXIF PERIF
Event	Auto-Baud Complete RX Break Received Wake Event (line is high-to-low) Smart Card Guard Time Counter Match Smart Card Block Time Counter Match	ABDIF RXBKIF WUIF GTCIF BTCIF

Table 21-8. Interrupts (continued)

Interrupt Type	Condition	Flag
Error	Parity Error Framing Error	PERIF FERIF
	Transmit Collision	TXCIF
	Transmit Shift Register Empty	TXMTIF
	RX Buffer Overflow	RXFOIF
	Auto-Baud Rollover	ABDOVIF
	Checksum Error (LIN mode only)	CERIF
	Smart Card Receive Repeat	RXRPTIF
	Smart Card Transmit Repeat	TXRPTIF
	Smart Card Waiting Time Counter Match	WTCIF

21.6.1. Interrupt Watermarks

Both TX and RX interrupt frequencies can be configured using the watermark setting. For transmit, the TXWM[2:0] bits setting allows the TX interrupt frequency to be based on the number of empty slots left in the TX buffer (UxTXB). When the transmitter is initially enabled (TXEN = `1`), the TX interrupt bit will be set on the condition that the module is enabled (ON = `1`) as well. The user should clear the TX interrupt bit in the ISR. For receive, the RXWM[2:0] bits setting allows the RX interrupt frequency to be based on how many bytes are in the RX buffer (UxRXB). By default, an RX interrupt will be generated when the RX buffer has at least one byte in it. The receive watermark interrupt will not be set if PERIF, FERIF or TXCIF are set and the corresponding PERIE, FERIE or TXCIE bits are set.

21.7. Power-Saving Modes

The UART provides support in power-saving modes, including the capability to run in Sleep and Idle modes. If a transmission or reception is in progress, and a power save command is executed, the operation will abort. SFR data, including UxCON, UxSTAT and UxBRG and the RX and TX buffers, will retain their values upon a wake condition and do not need to be reinitialized.

21.7.1. Sleep

When a device enters Sleep mode, the system clock used by the core processor and peripherals is halted. To run in Sleep mode, a clock source other than the system clock must be selected, and the SLPEN bit (UxCON[31]) must be set. Clock sources are device-specific (see Clocking and Baud Rate Configuration for details). This allows the UART to request the selected clock source and keep it active. The UART has the ability to continue transmitting the contents of UxTXB, receiving data and storing them in UxRXB.

The UART can also wake the processor from Sleep mode (when SPLEN = 0) upon the detection of an incoming byte, including Break and Sync characters used for auto-baud. To enable the wake feature, set the WUE bit (UxCON[12]). When a wake from Sleep occurs, the WUIF (UxUIR[7]) bit is set, and an event interrupt is generated, effectively waking the processor. If auto-baud is desired on wake, set the ABDEN bit before executing a `SLEEP` command.

21.7.2. Idle

In Idle mode, the core processor is halted. However, the peripherals, including the UART, continue to run. To halt the UART in Idle, the UART Stop in the Idle bit, SIDL (UxCON[13]), must be set.

22. Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) module is a synchronous serial interface useful for communicating with external peripherals and other microcontroller devices. These peripheral devices may be a serial EEPROM, shift register, display driver, ADC or an audio codec. The dsPIC33A family SPI module is compatible with Motorola® SPI and SIOP interfaces. Some of the key features of this module are:

- Host and Client Modes Support
- Four Different Clock Formats
- Framed SPI Protocol Support
- Standard and Enhanced Buffering Modes
- User-Configurable 8-bit, 16-bit and 32-bit Data Width
- Two Separate Shift Registers for Transmission and Reception
- SPIx Receive and Transmit Buffers are FIFO Buffers in Enhanced Buffering Mode
- User-Configurable Variable Data Width, From 2 to 32-bit
- Programmable Interrupt Event on Every 8-bit, 16-bit and 32-bit Data Transfer
- Audio Protocol Interface Mode
- I^2S Multi-Channel Audio Interface
- Audio Codec Interface Support
- TDM4/8/16/32 Digital Audio Modes Support Using Framed SPI Mode with DMA Channels
- DMA-Supported Audio Data Streaming

dsPIC33A devices support audio codec serial protocols, such as Inter-IC Sound (I^2S), Left Justified, Right Justified and PCM/DSP modes for 16, 24 and 32-bit audio data.

22.1. Device-Specific Information

Table 22-1. SPI Summary Table

SPI Module Instances	Max Clock Frequency	Peripheral Bus Speed	Clock Source
3	See Electrical Characteristics	Standard (1:2 CPU Clock)	See Table 22-2

Table 22-2. SPI Host Clock Source Selection bit

Value	Description
1	CLKGEN9
0	Standard (1:2 CPU Clock)

Table 22-3. SPI FIFO Depth

Setting	Size
Default	4-bytes
Note: 32-bit data equates to a buffer depth of X/4 = 1.	

Note: All devices in the dsPIC33AK256MPS306 family include three SPI modules. On 64-pin devices, the SPI instance SPI2 can operate at maximum speed when dedicated pins are selected. The selection is done using the SPI2PIN bit (FDEVOPT[13]). If the bit for SPI2PIN is ‘1’, the PPS pin will be used. When SPI2PIN is ‘0’, the SPI signals are routed to dedicated pins.

22.2. Architectural Overview

The SPI module is a synchronous serial communication interface used for short distances in embedded systems. SPI devices communicate in Full Duplex mode using a host/client architecture with a single host. The host device originates the frame for reading and writing. Multiple Client devices may be supported through selection with individual chip select (SSx) lines.

The SPI serial interface consists of four pins:

- SDIx: Serial Data Input
- SDOx: Serial Data Output
- SCKx: Shift Clock Input or Output
- SSx: Active-Low Client Select or Frame Synchronization I/O Pulse

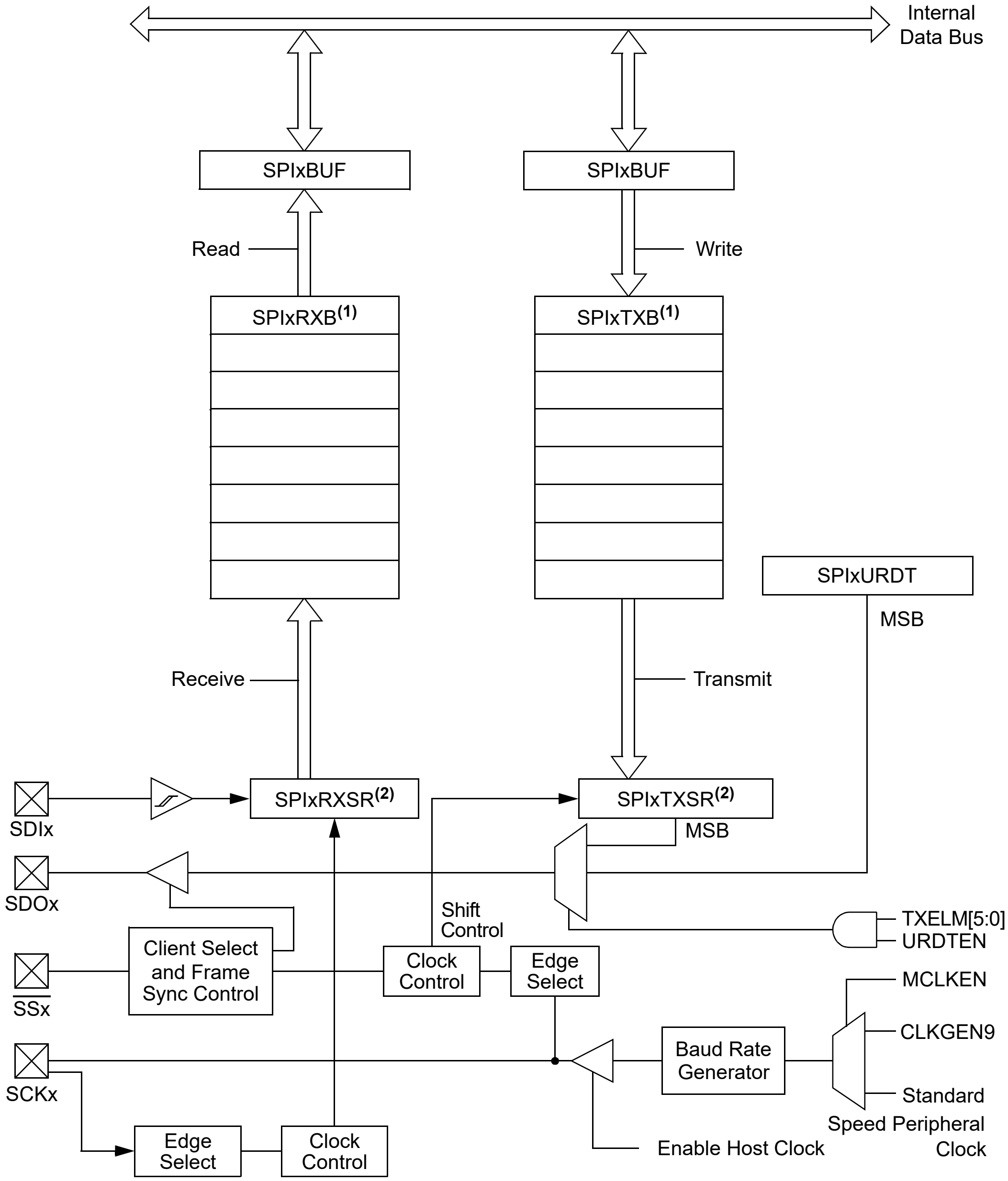
During each SPI clock cycle, a full-duplex data transmission occurs. The host sends a bit on the SDO line and the client reads the bit, while the client sends a bit on the SDI line and the host reads the bit. This sequence is the same for one-directional data transfer.

The SPI module offers the following operating modes:

- 8-bit, 16-bit and 32-bit Data Transmission modes
- 8-bit, 16-bit and 32-bit Data Reception modes
- Host and Client modes
- Framed SPI modes
- Audio Protocol Interface mode

Figure 22-1 shows the block diagram of the SPI module.

Figure 22-1. SPIx Module Block Diagram



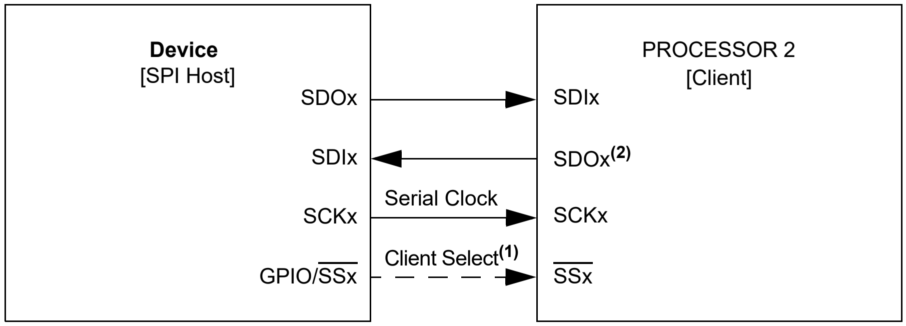
Note:

1. The SPIx Receive Buffer (SPIxRXB) and SPIx Transmit Buffer (SPIxTXB) registers are accessed through the SPIxBUF register and are multi-element FIFO buffers in Enhanced Buffer mode.
2. The SPIx Shift register is not directly accessible by application software.

22.2.1. Normal Mode SPI Operation

In Normal mode operation, the SPI host controls the generation of the serial clock. The number of output clock pulses corresponds to the transfer data width: 8, 16 or 32 bits, or depending upon variable data width configuration, from 2 to 32-bit. Figure 22-2 and Figure 22-3 illustrate SPI host-to-client and client-to-host device connections.

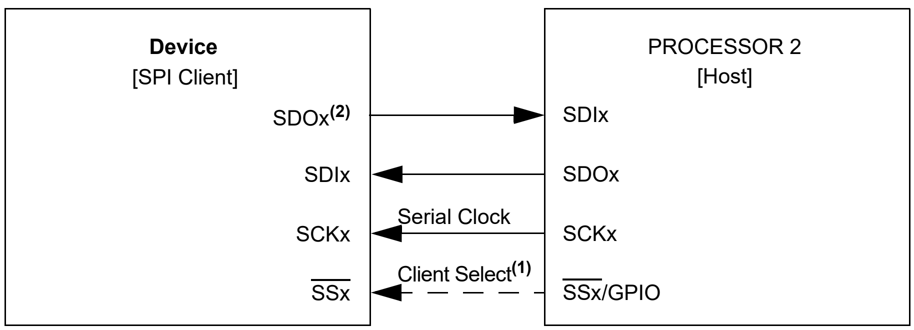
Figure 22-2. Typical SPIx Host-to-Client Device Connection Diagram



Notes:

1. In Normal mode, the usage of the Client Select pin ($\overline{SSx}$) is optional.
2. Control of the SDOx pin can be disabled for Receive Only modes.

Figure 22-3. Typical SPIx Client-to-Host Device Connection Diagram



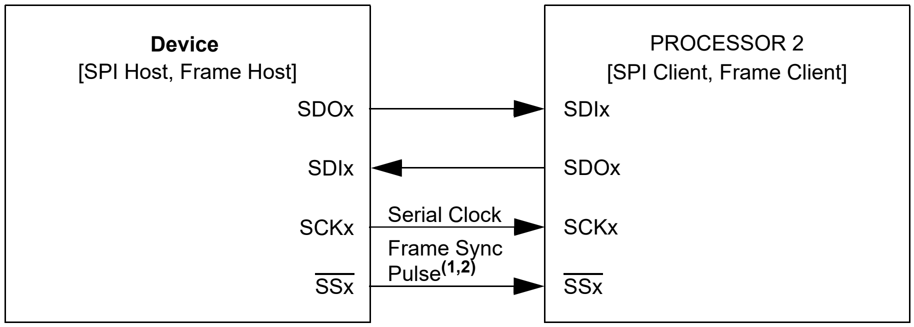
Notes:

1. In Normal mode, the usage of the Client Select pin ($\overline{SSx}$) is optional.
2. The control of the SDOx pin can be disabled for Receive Only modes.

22.2.2. Framed Mode SPI Operation

In Framed mode operation, the frame host controls the generation of the frame synchronization pulse. The SPI clock is still generated by the SPI host and is continuously running. Figure 22-4 and Figure 22-5 illustrate SPI frame host and frame client device connections.

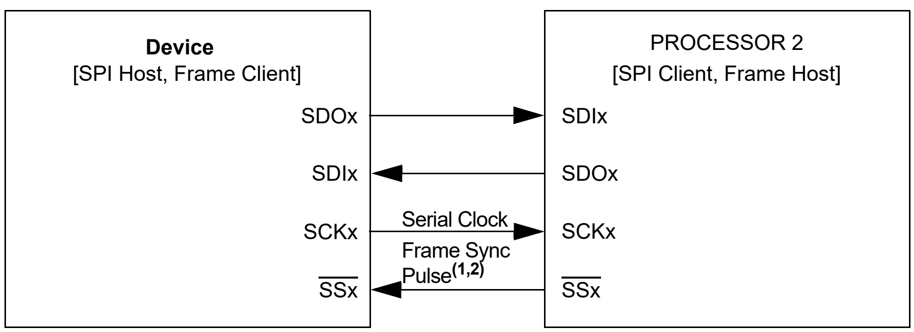
Figure 22-4. Typical SPIx Host, Frame Host Connection Diagram



Notes:

1. In Framed SPI mode, the $\overline{SSx}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI mode requires the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

Figure 22-5. Typical SPIx Host, Frame Client Connection Diagram



Notes:

1. In Framed SPI mode, the $\overline{SSx}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI mode requires the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

22.2.3. Audio Protocol Interface Mode

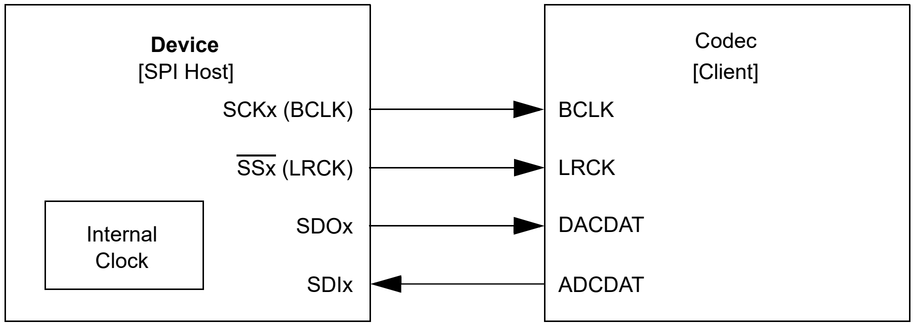
The SPI module supports audio interfaces and is configurable for a variety of protocols including:

- PCM/DSP mode
- Right Justified mode
- Left Justified mode
- I^2S mode

22.2.3.1. SPI in Audio Host Mode Connected to a Codec Client

Figure 22-6 shows the Bit Clock (BCLK) and Left/Right Channel Clock (LRCK) as generated by the dsPIC33A SPI module.

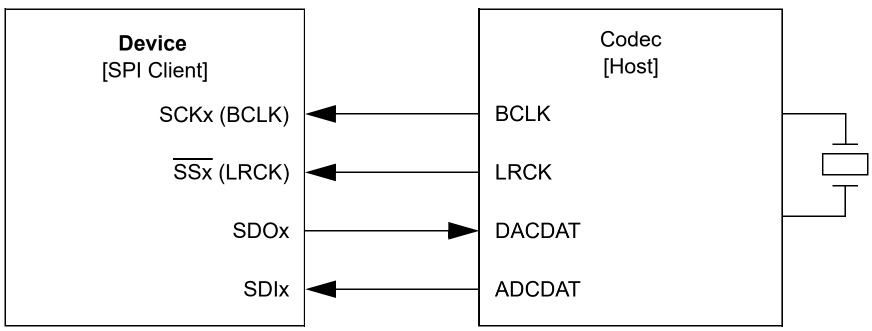
Figure 22-6. Host Generating its Own Clock – Output BCLK and LRCK



22.2.3.2. SPI in Audio Client Mode Connected to a Codec Host

Figure 22-7 shows the BCLK and LRCK as generated by the codec host.

Figure 22-7. Codec Device as Host Generates Required Clock via External Crystal



22.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1800	SPI1CON1	31:24	AUDEN	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD[1:0]	
		23:16	FRMEN	FRMSYNC	FRMPOL	MSSEN	FRMSYPW	FRMCNT[2:0]		
		15:8	ON		SIDL	DISSDO	MODE[32,16]		SMP	CKE
		7:0	SSEN	CKP	MSTEN	DISSDI	DISSCK	MCLKEN	SPIFE	ENHBUF
0x1804	SPI1CON2	31:24								
		23:16								
		15:8								
		7:0				WLENGTH[4:0]				
0x1808	SPI1STAT	31:24						RXELM[2:0]		
		23:16						TXELM[2:0]		
		15:8				FRMERR	BUSY			SPITUR
		7:0	SRMT	SPIROV	SPIRBE		SPITBE		SPITBF	SPIRBF
0x180C	SPI1BUF	31:24	SPI1BUF[31:24]							
		23:16	SPI1BUF[23:16]							
		15:8	SPI1BUF[15:8]							
		7:0	SPI1BUF[7:0]							
0x1810	SPI1BRG	31:24								
		23:16								
		15:8				SPI1BRG[12:8]				
		7:0	SPI1BRG[7:0]							
0x1814	SPI1IMSK	31:24	RXWIEN					RXMSK[2:0]		
		23:16	TXWIEN					TXMSK[2:0]		
		15:8				FRMERREN	BUSYEN			SPITUREN
		7:0	SRMTEN	SPIROVEN	SPIRBEN		SPITBEN		SPITBFEN	SPIRBFEN
0x1818	SPI1URDT	31:24	SPI1URDT[31:24]							
		23:16	SPI1URDT[23:16]							
		15:8	SPI1URDT[15:8]							
		7:0	SPI1URDT[7:0]							
0x181C ... 0x181F	Reserved									
0x1820	SPI2CON1	31:24	AUDEN	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD[1:0]	
		23:16	FRMEN	FRMSYNC	FRMPOL	MSSEN	FRMSYPW	FRMCNT[2:0]		
		15:8	ON		SIDL	DISSDO	MODE[32,16]		SMP	CKE
		7:0	SSEN	CKP	MSTEN	DISSDI	DISSCK	MCLKEN	SPIFE	ENHBUF
0x1824	SPI2CON2	31:24								
		23:16								
		15:8								
		7:0				WLENGTH[4:0]				
0x1828	SPI2STAT	31:24						RXELM[2:0]		
		23:16						TXELM[2:0]		
		15:8				FRMERR	BUSY			SPITUR
		7:0	SRMT	SPIROV	SPIRBE		SPITBE		SPITBF	SPIRBF
0x182C	SPI2BUF	31:24	SPI2BUF[31:24]							
		23:16	SPI2BUF[23:16]							
		15:8	SPI2BUF[15:8]							
		7:0	SPI2BUF[7:0]							
0x1830	SPI2BRG	31:24								
		23:16								
		15:8				SPI2BRG[12:8]				
		7:0	SPI2BRG[7:0]							
0x1834	SPI2IMSK	31:24	RXWIEN					RXMSK[2:0]		
		23:16	TXWIEN					TXMSK[2:0]		
		15:8				FRMERREN	BUSYEN			SPITUREN
		7:0	SRMTEN	SPIROVEN	SPIRBEN		SPITBEN		SPITBFEN	SPIRBFEN
0x1838	SPI2URDT	31:24	SPI2URDT[31:24]							
		23:16	SPI2URDT[23:16]							
		15:8	SPI2URDT[15:8]							
		7:0	SPI2URDT[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x183C ... 0x183F	Reserved									
0x1840	SPI3CON1	31:24	AUDEN	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD[1:0]	
		23:16	FRMEN	FRMSYNC	FRMPOL	MSSEN	FRMSYPW	FRMCNT[2:0]		
		15:8	ON		SIDL	DISSDO	MODE[32,16]		SMP	CKE
		7:0	SSEN	CKP	MSTEN	DISSDI	DISSCK	MCLKEN	SPIFE	ENHBUF
0x1844	SPI3CON2	31:24								
		23:16								
		15:8								
		7:0				WLENGTH[4:0]				
0x1848	SPI3STAT	31:24						RXELM[2:0]		
		23:16						TXELM[2:0]		
		15:8				FRMERR	BUSY			SPITUR
		7:0	SRMT	SPIROV	SPIRBE		SPITBE		SPITBF	SPIRBF
0x184C	SPI3BUF	31:24	SPI3BUF[31:24]							
		23:16	SPI3BUF[23:16]							
		15:8	SPI3BUF[15:8]							
		7:0	SPI3BUF[7:0]							
0x1850	SPI3BRG	31:24								
		23:16								
		15:8				SPI3BRG[12:8]				
		7:0	SPI3BRG[7:0]							
0x1854	SPI3IMSK	31:24	RXWIEN					RXMSK[2:0]		
		23:16	TXWIEN					TXMSK[2:0]		
		15:8				FRMERREN	BUSYEN			SPITUREN
		7:0	SRMTEN	SPIROVEN	SPIRBEN		SPITBEN		SPITBFEN	SPIRBFEN
0x1858	SPI3URDT	31:24	SPI3URDT[31:24]							
		23:16	SPI3URDT[23:16]							
		15:8	SPI3URDT[15:8]							
		7:0	SPI3URDT[7:0]							

22.3.1. SPIx Control Register 1

Name: SPIxCON1
Offset: 0x1800, 0x1820, 0x1840

Notes:

1. When AUDEN = 1, this module functions as if CKE = 0, regardless of its actual value.
2. When FRMEN = 1, SSEN is not used.
3. MCLKEN can only be written when the ON bit = 0.
4. This channel is not meaningful for PCM/DSP mode as LRC follows FRMSYPW.
5. SPI operates with DMA in Standard Buffer mode only, ENHBUF = 0.
6. SPISGNEXT function is available for 8, 16 and 32-bit data length transfers only.

Legend: R = Readable bit, W = Writable bit

Bit	31	30	29	28	27	26	25	24
	AUDEN	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FRMEN	FRMSYNC	FRMPOL	MSSEN	FRMSYPW	FRMCNT[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	DISSDO	MODE[32,16]		SMP	CKE
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SSEN	CKP	MSTEN	DISSDI	DISSCK	MCLKEN	SPIFE	ENHBUF
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – AUDEN Audio Codec Support Enable bit[1]
Table 22-4 shows the AUDEN bit setting requirements for TDM mode support.

Table 22-4. AUDEN TDM Mode Support Requirements

	AUDEN=1 (Audio Protocol Mode)	AUDEN=0 (Framed-SPI + DMA)	Note
TDM4/8/16/32	✗	✓	Multi-slot TDM requires AUDEN=0

Value	Description
1	Audio protocol is enabled; MSTEN controls the direction of both SCKx and frame (a.k.a. LRC), and this module functions as if FRMEN = 1, FRMSYNC = MSTEN, FRMCNT[2:0] = 001 and SMP = 0, regardless of their actual values.
0	Audio protocol is disabled. When AUDEN = 0, TDM mode is supported using SPI Frame Mode and DMA channels.

Bit 30 – SPISGNEXT SPIx Sign-Extend RX FIFO Read Data Enable bit[6]

Value	Description
1	Data from RX FIFO is sign-extended (upper unused bits should replicate the MSb of the received data).
0	Data from RX FIFO is not sign-extended (upper unused bits are always 1'b0).

Bit 29 – IGNROV Ignore Receive Overflow bit

Value	Description
1	A Receive Overflow (ROV) is NOT a critical error; during ROV, data in the FIFO is not overwritten by the received data.
0	An ROV is a critical error that stops SPI operation.

Bit 28 – IGNTUR Ignore Transmit Underrun bit

Value	Description
1	A Transmit Underrun (TUR) is NOT a critical error and data indicated by URDTEN is transmitted until the SPIxTXB is not empty.
0	A TUR is a critical error that stops SPI operation.

Bit 27 – AUDMONO Audio Data Format Transmit bit(2)

Value	Description
1	Audio data is mono (i.e., each data word is transmitted on both the left and right channels).
0	Audio data is stereo.

Bit 26 – URDTEN Transmit Underrun Data Enable bit(3)

Value	Description
1	Transmits data out of the SPIxURDT register during Transmit Underrun conditions.
0	Transmits the last received data during Transmit Underrun conditions.

Bits 25:24 – AUDMOD[1:0] Audio Protocol Mode Selection bits

Value	Description
11	PCM/DSP mode
10	Right Justified mode: This module functions as if SPIFE = 1, regardless of its actual value.
01	Left Justified mode: This module functions as if SPIFE = 1, regardless of its actual value.
00	I^2S mode: This module functions as if SPIFE = 0, regardless of its actual value.

Bit 23 – FRMEN Framed SPIx Support bit

Value	Description
1	Framed SPIx support is enabled ($\overline{SSx}$ pin is used as the FSYNC input/output).
0	Framed SPIx support is disabled.

Bit 22 – FRMSYNC Frame Sync Pulse Direction Control bit

Value	Description
1	Frame Sync pulse input (client)
0	Frame Sync pulse output (host)

Bit 21 – FRMPOL Frame Sync/Client Select Polarity bit

Value	Description
1	Frame Sync pulse/Client Select is active high.
0	Frame Sync pulse/Client Select is active low.

Bit 20 – MSSEN Host Mode Client Select Enable bit

Value	Description
1	SPIx Client Select support is enabled with polarity determined by FRMPOL (SSx pin is automatically driven during transmission in Host mode).
0	Client Select SPIx support is disabled (SSx pin will be controlled by port I/O).

Bit 19 – FRMSYPW Frame Sync Pulse-Width bit

Value	Description
1	Frame Sync pulse is one serial word length wide (as defined by MODE[32,16]/WLENGTH[4:0]).
0	Frame Sync pulse is one clock (SCKx) wide.

Bits 18:16 – FRMCNT[2:0] Frame Sync Pulse Counter bits
Controls the number of serial words per Sync pulse.

Value	Description
111	Reserved
110	Reserved
101	Generate/Receive a Frame Sync pulse every 32 serial words.
100	Generate/Receive a Frame Sync pulse every 16 serial words.
011	Generate/Receive a Frame Sync pulse every 8 serial words.
010	Generate/Receive a Frame Sync pulse every 4 serial words.
001	Generate/Receive a Frame Sync pulse every 2 serial words (value used by audio protocols).
000	Generate/Receive a Frame Sync pulse each serial word.

Bit 15 – ON SPIx On bit

Value	Description
1	Enables module.
0	Turns off and resets module, disables clocks, disables interrupt event generation, allows SFR modifications.

Bit 13 – SIDL SPIx Stop in Idle Mode bit

Value	Description
1	Halts in CPU Idle mode.
0	Continues to operate in CPU Idle mode.

Bit 12 – DISSDO Disable SDOx Output Port bit

Value	Description
1	SDOx pin is not used by the module; pin is controlled by port function.
0	SDOx pin is controlled by the module.

Bits 11:10 – MODE[32,16] Serial Word Length bits(1,4)

MODE32	MODE16	AUDEN	Communication
1	x	0	32-Bit
0	1		16-Bit
0	0		8-Bit
1	1	1	24-Bit Data, 32-Bit FIFO, 32-Bit Channel/64-Bit Frame
1	0		32-Bit Data, 32-Bit FIFO, 32-Bit Channel/64-Bit Frame
0	1		16-Bit Data, 16-Bit FIFO, 32-Bit Channel/64-Bit Frame
0	0		16-Bit FIFO, 16-Bit Channel/32-Bit Frame

Bit 9 – SMP SPIx Data Input Sample Phase bit
Client Mode:
Input data is always sampled in the middle of the data output time, regardless of the SMP setting.
Host Mode:

Value	Description
1	Input data is sampled at the end of data output time.
0	Input data is sampled at the middle of data output. time

Bit 8 – CKE SPIx Clock Edge Select bit(1)

Value	Description
1	Transmit happens on the transition from the active clock state to the Idle clock state.
0	Transmit happens on the transition from the Idle clock state to the active clock state.

Bit 7 – SSEN Client Select Enable bit (Client mode)(2)

Value	Description
1	$\overline{SSx}$ pin is used by the module in Client mode; the $\overline{SSx}$ pin is used as the Client Select input.
0	$\overline{SSx}$ pin is not used by the module (the pin is controlled by the port function).

Bit 6 – CKP Clock Polarity Select bit

Value	Description
1	Idle state for the clock is a high level; the active state is a low level.
0	Idle state for the clock is a low level; the active state is a high level.

Bit 5 – MSTEN Host Mode Enable bit

Value	Description
1	Host mode
0	Client mode

Bit 4 – DISSDI Disable SDIx Input Port bit

Value	Description
1	SDIx pin is not used by the module; the pin is controlled by the port function.
0	SDIx pin is controlled by the module.

Bit 3 – DISSCK Disable SCKx Output Port bit

Value	Description
1	SCKx pin is not used by the module; the pin is controlled by the port function.
0	SCKx pin is controlled by the module.

Bit 2 – MCLKEN SPI Host Clock Source Selection bit(3)

See Table 22-2.

Bit 1 – SPIFE Frame Sync Pulse Edge Select bit

Value	Description
1	Frame Sync pulse (Idle-to-active edge) coincides with the first bit clock.
0	Frame Sync pulse (Idle-to-active edge) precedes the first bit clock.

Bit 0 – ENHBUF Enhanced Buffer Enable bit(5)

Value	Description
1	Enhanced Buffer mode is enabled.
0	Enhanced Buffer mode is disabled.

22.3.2. SPIx Control Register 2

Name: SPIxCON2
Offset: 0x1804,0x1824, 0x1844

Notes:
1. These bits are effective when AUDEN = 0 only.
2. Varying the length by changing these bits does not affect the depth of the TX/RX FIFO.

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				WLENGTH[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 4:0 – WLENGTH[4:0] Variable Word Length bits[1,2]

Value	Description
11111	32-bit data
11110	31-bit data
11101	30-bit data
11100	29-bit data
11011	28-bit data
11010	27-bit data
11001	26-bit data
11000	25-bit data
10111	24-bit data
10110	23-bit data
10101	22-bit data
10100	21-bit data
10011	20-bit data
10010	19-bit data
10001	18-bit data
10000	17-bit data
01111	16-bit data
01110	15-bit data
01101	14-bit data
01100	13-bit data
01011	12-bit data

Value	Description
01010	11-bit data
01001	10-bit data
01000	9-bit data
00111	8-bit data
00110	7-bit data
00101	6-bit data
00100	5-bit data
00011	4-bit data
00010	3-bit data
00001	2-bit data
00000	See MODE[32,16] bits in SPIxCON1[11:10]

22.3.3. SPIx Status Register

Name: SPIxSTAT
Offset: 0x1808 + (x-1)*0x04 [x=1..3]

Note:
1. SPITUR is cleared when SPIEN = 0. When IGNTUR = 1, SPITUR provides the dynamic status of the Transmit Underrun condition, but does not stop RX/TX operation and does not need to be cleared by software.

Legend: C = Clearable bit; HS = Hardware Settable bit; HSC = Hardware Settable/Clearable bit

Bit	31	30	29	28	27	26	25	24
						RXELM[2:0]		
Access						R/HS	R/HS	R/HS
Reset						0	0	0

Bit	23	22	21	20	19	18	17	16
						TXELM[2:0]		
Access						R/HSC	R/HSC	R/HSC
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
				FRMERR	BUSY			SPITUR
Access				R/C/HS	R/HSC			R/HSC
Reset				0	0			0

Bit	7	6	5	4	3	2	1	0
	SRMT	SPIROV	SPIRBE		SPITBE		SPITBF	SPIRBF
Access	R/HSC	R/C/HS	R/HSC		R/HSC		R/HSC	R/HSC
Reset	0	0	1		1		0	0

Bits 26:24 – RXELM[2:0] Receive Buffer Element Count bits (valid in Enhanced Buffer mode)

Bits 18:16 – TXELM[2:0] Transmit Buffer Element Count bits (valid in Enhanced Buffer mode)

Bit 12 – FRMERR SPIx Frame Error Status bit

Value	Description
1	Frame error is detected.
0	No frame error is detected.

Bit 11 – BUSY SPIx Activity Status bit

Value	Description
1	Module is currently busy with some transactions.
0	No ongoing transactions (at time of read).

Bit 8 – SPITUR SPIx Transmit Underrun Status bit(1)

Value	Description
1	Transmit buffer has encountered a Transmit Underrun condition.
0	Transmit buffer does not have a Transmit Underrun condition.

Bit 7 – SRMT Shift Register Empty Status bit

Value	Description
1	No current or pending transactions (i.e., neither SPIxTXB or SPIxTXSR contains data to transmit).
0	Current or pending transactions.

Bit 6 – SPIROV SPIx Receive Overflow Status bit

Value	Description
1	A new byte/half-word/word has been completely received when the SPIxRXB is full.
0	No overflow.

Bit 5 – SPIRBE SPIx RX Buffer Empty Status bit

Standard Buffer Mode:
Automatically set in hardware when SPIxBUF is read from, reading SPIxRXB. Automatically cleared in hardware when SPIx transfers data from SPIxRXSR to SPIxRXB.
Enhanced Buffer Mode:
Indicates RXELM[2:0] = `00`.

Value	Description
1	RX buffer is empty.
0	RX buffer is not empty.

Bit 3 – SPITBE SPIx Transmit Buffer Empty Status bit

Standard Buffer Mode:
Automatically set in hardware when SPIx transfers data from SPIxTXB to SPIxTXSR. Automatically cleared in hardware when SPIxBUF is written, loading SPIxTXB.
Enhanced Buffer Mode:
Indicates TXELM[2:0] = `00`.

Value	Description
1	SPIxTXB is empty.
0	SPIxTXB is not empty.

Bit 1 – SPITBF SPIx Transmit Buffer Full Status bit

Standard Buffer Mode:
Automatically set in hardware when SPIxBUF is written, loading SPIxTXB. Automatically cleared in hardware when SPIx transfers data from SPIxTXB to SPIxTXSR.
Enhanced Buffer Mode:
Indicates TXELM[2:0] = `11`.

Value	Description
1	SPIxTXB is full.
0	SPIxTXB not full.

Bit 0 – SPIRBF SPIx Receive Buffer Full Status bit

Standard Buffer Mode:
Automatically set in hardware when SPIx transfers data from SPIxRXSR to SPIxRXB. Automatically cleared in hardware when SPIxBUF is read from, reading SPIxRXB.
Enhanced Buffer Mode:
Indicates RXELM[2:0] = `11`.

Value	Description
1	SPIxRXB is full.
0	SPIxRXB is not full.

22.3.4. SPI Buffer Register

Name: SPIxBUF
Offset: 0x180C + (x-1)*0x04 [x=1..3]

Note: Changing the BRG value when SPIEN = 1 causes undefined behavior.

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	SPIxBUF[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SPIxBUF[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SPIxBUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SPIxBUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SPIxBUF[31:0] SPIx FIFO Data bits
Table 22-5 summarizes the valid data fields for possible values of the MODE32, MODE16 and WLENGTH[4:0] bits.

Table 22-5. MODE32, MODE16 and WLENGTH[4:0] Data Fields

MODE32	MODE16	WLENGTH[4:0]	COMMUNICATION	Valid Data Field Data
1	x	0	32-bit	DATA[31:0]
0	1	0	16-bit	DATA[15:0]
0	0	0	8-bit	DATA[07:0]
x	x	16 < N < 31	(N+1)-bit	DATA[31:(31-N)]
x	x	8 < N < 15	(N+1)-bit	DATA[15:(15-N)]
x	x	1 < N < 7	(N+1)-bit	DATA[07:(07-N)]

22.3.5. SPIx Baud Rate Generator Register

Name: SPIxBRG
Offset: 0x1810 + (x-1)*0x04 [x=1..3]

Note:
1. Changing the BRG value when SPIEN = 1 causes undefined behavior.

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				SPIxBRG[12:8]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SPIxBRG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 12:0 – SPIxBRG[12:0] SPI Baud Rate Generator Divisor bits[(1)]

22.3.6. SPIx Interrupt Mask Register

Name: SPIxIMSK
Offset: 0x1814, 0x1834, 0x1854

Note:

1. Mask values higher than Value 4 are not valid. The module will not trigger a match for any value in this case.

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	RXWIEN					RXMSK[2:0]		
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit	23	22	21	20	19	18	17	16
	TXWIEN					TXMSK[2:0]		
Access	R/W					R/W	R/W	R/W
Reset	0					0	0	0

Bit	15	14	13	12	11	10	9	8
				FRMERREN	BUSYEN			SPITUREN
Access				R/W	R/W			R/W
Reset				0	0			0

Bit	7	6	5	4	3	2	1	0
	SRMTEN	SPIROVEN	SPIRBEN		SPITBEN		SPITBFEN	SPIRBFEN
Access	R/W	R/W	R/W		R/W		R/W	R/W
Reset	0	0	0		0		0	0

Bit 31 – RXWIEN Receive Watermark Interrupt Enable bit

Value	Description
1	Triggers receive buffer element watermark interrupt when RXMSK[2:0] ≤ RXELM[2:0].
0	Disables the receive buffer element watermark interrupt.

Bits 26:24 – RXMSK[2:0] RX Buffer Mask bits(1)

RX mask bits; used in conjunction with the RXWIEN bit.

Bit 23 – TXWIEN Transmit Watermark Interrupt Enable bit

Value	Description
1	Triggers transmit buffer element watermark interrupt when TXMSK[2:0] ≤ TXELM[2:0].
0	Disables the transmit buffer element watermark interrupt.

Bits 18:16 – TXMSK[2:0] TX Buffer Mask bits(1)

TX mask bits; used in conjunction with the TXWIEN bit.

Bit 12 – FRMERREN Enable Interrupt Events via FRMERR bit

Value	Description
1	Frame error generates an interrupt event.
0	Frame error does not generate an interrupt event.

Bit 11 – BUSYEN Enable Interrupt Events via SPIBUSY bit

Value	Description
1	BUSY generates an interrupt event.
0	BUSY does not generate an interrupt event.

Bit 8 – SPITUREN Enable Interrupt Events via SPITUR bit

Value	Description
1	Transmit Underrun (TUR) generates an interrupt event.
0	Transmit Underrun does not generate an interrupt event.

Bit 7 – SRMTEN Enable Interrupt Events via SRMT bit

Value	Description
1	Shift Register Empty (SRMT) generates interrupt events.
0	Shift Register Empty does not generate interrupt events.

Bit 6 – SPIROVEN Enable Interrupt Events via SPIROV bit

Value	Description
1	SPIx Receive Overflow (ROV) generates an interrupt event.
0	SPIx Receive Overflow does not generate an interrupt event.

Bit 5 – SPIRBEN Enable Interrupt Events via SPIRBE bit

Value	Description
1	SPIx RX buffer empty generates an interrupt event.
0	SPIx RX buffer empty does not generate an interrupt event.

Bit 3 – SPITBEN Enable Interrupt Events via SPITBE bit

Value	Description
1	SPIx transmit buffer empty generates an interrupt event.
0	SPIx transmit buffer empty does not generate an interrupt event.

Bit 1 – SPITBFEN Enable Interrupt Events via SPITBF bit

Value	Description
1	SPIx transmit buffer full generates an interrupt event.
0	SPIx transmit buffer full does not generate an interrupt event.

Bit 0 – SPIRBFEN Enable Interrupt Events via SPIRBF bit

Value	Description
1	SPIx receive buffer full generates an interrupt event.
0	SPIx receive buffer full does not generate an interrupt event.

22.3.7. SPIx Underrun Data Register

Name: SPIxURDT
Offset: 0x1818 + (x-1)*0x04 [x=1..3]

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	SPIxURDT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SPIxURDT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SPIxURDT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SPIxURDT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SPIxURDT[31:0] SPI Underrun Data bits
These bits are only used when URDTEN = 1. This register holds the data to transmit when a Transmit Underrun condition occurs.
Table 22-6 summarizes the valid data fields for possible values of the MODE32, MODE16 and WLENGTH[4:0] bits.

Table 22-6. MODE32, MODE16 and WLENGTH[4:0] Data Fields

MODE32	MODE16	WLENGTH[4:0]	COMMUNICATION	Valid Data Field
1	x	0	32-bit	DATA[31:0]
0	1	0	16-bit	DATA[15:0]
0	0	0	8-bit	DATA[07:0]
x	x	16 < N < 31	(N+1)-bit	DATA[31:(31-N)]
x	x	8 < N < 15	(N+1)-bit	DATA[15:(15-N)]
x	x	1 < N < 7	(N+1)-bit	DATA[07:(07-N)]

22.4. Operation

22.4.1. Modes of Operation

The SPI module offers the following operating modes:

- 8-bit, 16-bit and 32-bit Data Transmission modes
- 8-bit, 16-bit and 32-bit Data Reception modes
- Host and Client modes
- Framed SPI modes
- Audio Protocol Interface mode

22.4.1.1. SPI Host Mode Clock Frequency

The SPI module allows flexibility in baud rate generation through the 13-bit SPIxBRG register. SPIxBRG is readable and writable and determines the baud rate. The clock provided to the SPI module, F_{SPICLK}, can be found in the Device-Specific Information. This clock is divided based on the value loaded into SPIxBRG. The SCKx clock, obtained by dividing F_{SPICLK}, has a 50% duty cycle, and it is provided to the external devices through the SCKx pin.

Note: The SCKx clock is not free-running for Non-Framed SPI modes. It will only run for 8, 16, or 32 pulses when SPIxBUF is loaded with data. It will, however, be continuous for Framed modes.

Equation 22-1 defines the SCKx clock frequency as a function of SPIxBRG settings.

Equation 22-1. SCKx Frequency

$$F_{SCK} = \frac{F_{SPICLK}}{2 \cdot (SPIxBRG + 1)}$$

Therefore, the maximum baud rate possible is $F_{SPICLK}/2$ (SPIxBRG = 0), and the minimum baud rate possible is $F_{SPICLK}/16384$.

Some sample SPI clock frequencies are shown in Table 22-7.

Table 22-7. Sample SCKx Frequencies[1]

SPIxBRG Setting	0	15	31	63	85	127	255	511
F_{SPICLK} = 32 MHz	16.00 MHz	10.0 MHz	500 kHz	257 kHz	190.48 kHz	125 kHz	62.5 kHz	31.25 kHz
F_{SPICLK} = 25 MHz	12.50 MHz	781.25 kHz	390.63 kHz	145.35 kHz	97.66 kHz	281.25 kHz	48.83 kHz	24.41 kHz
F_{SPICLK} = 20 MHz	10.00 MHz	625 kHz	312.50 kHz	156.25 kHz	116.28 kHz	78.13 kHz	39.06 kHz	19.53 kHz
F_{SPICLK} = 12 MHz	6.00 MHz	375 MHz	187.50 kHz	93.75 kHz	69.77 kHz	46.88 kHz	23.44 kHz	11.72 kHz
F_{SPICLK} = 10 MHz	5.00 MHz	312.50 kHz	156.25 kHz	78.13 kHz	58.14 kHz	39.06 kHz	19.53 kHz	9.77kHz
F_{SPICLK} = 8 MHz	4.00 MHz	250 kHz	125 kHz	62.50 kHz	46.51 kHz	31.25 kHz	15.63 kHz	7.81 kHz

Note:

1. Not all clock rates are supported. For further information, refer to the SPI timing specifications in the Electrical Characteristics.

22.4.1.2. 8-Bit, 16-Bit and 32-Bit Operations

The SPI module allows three types of data widths when transmitting and receiving data over an SPI bus. The selection of data width determines the minimum length of SPI data. For example, when the selected data width is 32, all transmission and receptions are performed in 32-bit values. All reads and writes from the CPU are also performed in 32-bit values. Accordingly, the application software should select the appropriate data width to maximize its data throughput.

Two control bits, MODE32 and MODE16 (SPIxCON1[11:10]), which are referred to as MODE[32,16], define the mode of operation. To change the mode of operation on-the-fly, the SPI module must be Idle (i.e., not performing any transactions). If the SPI module is switched off (SPIxCON1[15] = `0`), the new mode will be available when the module is again switched on.

Additionally, the following items should be noted in this context:

- The MODE[32,16] bits should not be changed when a transaction is in progress.
- The first bit to be shifted out from SPIxTXSR varies with the selected mode of operation:
  - 8-bit mode, bit 7
  - 16-bit mode, bit 15
  - 32-bit mode, bit 31
- In each mode, data are shifted into bit 0 of the SPIxRXSR.
- The number of clock pulses at the SCKx pin are also dependent on the selected mode of operation:
  - 8-bit mode, 8 clocks
  - 16-bit mode, 16 clocks
  - 32-bit mode, 32 clocks

22.4.1.3. Buffer Modes

There are two SPI Buffering modes: Standard and Enhanced.

22.4.1.3.1. Standard Buffer Mode

The SPIx Data Receive/Transmit Buffer (SPIxBUF) register is actually two separate internal registers: the Transmit Buffer (SPIxTXB) and the Receive Buffer (SPIxRXB). These two unidirectional registers share the SFR address of SPIxBUF.

When a complete byte/word is received, it is transferred from SPIxRXSR to SPIxRXB and the SPIRBF bit is set. If the software reads the SPIxBUF buffer, the SPIRBF bit is cleared.

As the software writes to SPIxBUF, the data are loaded into the SPIxTXB and the SPITBF bit is set by hardware. As the data are transmitted out of SPIxTXSR, the SPITBF bit is cleared.

The SPI module double-buffers transmit/receive operations and allows continuous data transfers in the background. Transmission and reception occur simultaneously in SPIxTXSR and SPIxRXSR, respectively.

22.4.1.3.2. Enhanced Buffer Mode

The Enhanced Buffer Enable (ENHBUF) bit in the SPIx Control Register 1 (SPIxCON1[0]) can be set to enable the Enhanced Buffer mode.

In Enhanced Buffer mode, two FIFO buffers are used for the SPIx Transmit Buffer (SPIxTXB) and the SPIx Receive Buffer (SPIxRXB). SPIxBUF provides access to both the receive and transmit FIFOs. The data transmission and reception in the SPIxSR buffer are identical to that in the Standard Buffer mode. The FIFO depth depends on the data width chosen by the Word/Half-Word Byte Communication Select (MODE[32,16]) bits in the SPIx Control Register 1 (SPIxCON1[11:10]). The FIFO depth varies between devices. For a device with FIFO depth 'X', the MODE field will modify it as follows:

- MODE = 8-bit, FIFO depth = X
- MODE = 16-bit, FIFO depth = X/2
- MODE = 32-bit, FIFO depth = X/4

Note: The FIFO depth does not change when a variable word length is configured. Refer to Table 22-3 for the value of FIFO depth 'X'.

The SPITBF status bit is set when all of the elements in the transmit FIFO buffer are full, and it is cleared if one or more of those elements are empty. The SPIRBF status bit is set when all of the elements in the receive FIFO buffer are full, and it is cleared if the SPIxBUF buffer is read by the software.

The SPITBE status bit is set if all the elements in the transmit FIFO buffer are empty and is cleared otherwise. The SPIRBE bit is set if all of the elements in the receive FIFO buffer are empty and is cleared otherwise.

There is underrun or overflow protection against reading an empty receive FIFO element or writing to a full transmit FIFO element. The SPIxSTAT register provides the SPIx Transmit Underrun bit (SPITUR) and the Receive Overflow Status bit (SPIROV). Depending on the requirements, IGNTUR and IGNROV can be configured for SPI operation to continue or not at the time of error. When a Transmit Underrun occurs, the last received data or the data in the SPIxURDT register can be transmitted by configuring the URDTEN bit (SPIxCON1[26]).

The Receive Buffer Element Count bits (RXELM[2:0]) in the SPIx Status Register (SPIxSTAT[26:24]) indicate the number of unread elements in the receive FIFO. The Transmit Buffer Element Count bits (TXELM[2:0]) in the SPIx Status Register (SPIxSTAT[18:16]) indicate the number of elements not transmitted in the transmit FIFO.

When configured for non-framed Client mode, it is important to ensure that the software can reload the transmit buffer quickly enough to keep up with the configured transfer rate. If the SPIxTXB is empty at the start of a transaction, then the transmit behavior will be undefined and is likely to cause errors (duplicate transmissions, missed bits, etc.) in the received data.

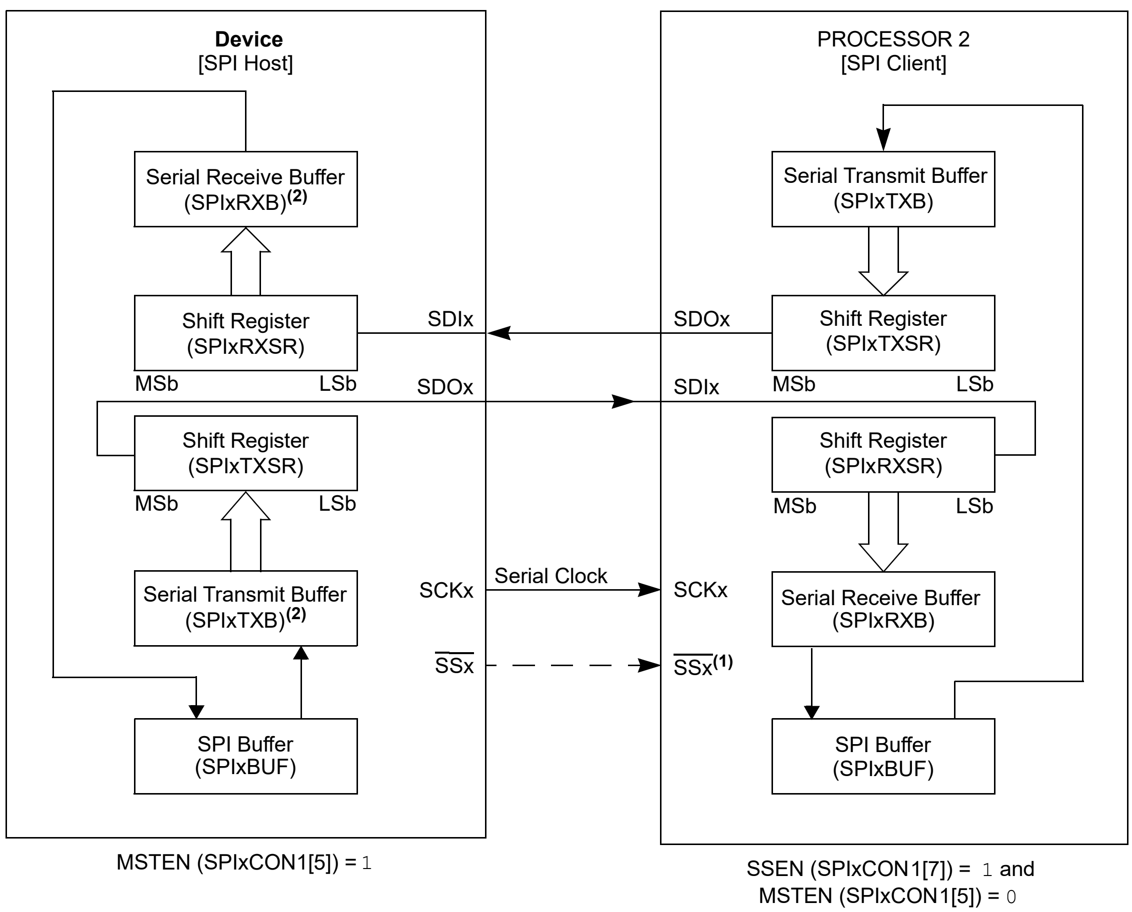
22.4.1.4. Variable Word Length Operation

The SPI module allows variable word length when transmitting and receiving data over an SPI bus. Word length can vary from 2 to 32 bits. Different word lengths can be configured by changing the WLENGTH[4:0] bits (SPIxCON2[4:0]). The number of clock pulses at the SCKx pin will correspond to the word length that is selected.

22.4.1.5. Host and Client Modes

In Host and Client modes, data can be thought of as taking a direct path between the MSb of one module's Shift register and the LSb of the other, and then moving them into the appropriate transmit or receive buffer. The module configured as the host module provides the serial clock and synchronization signals (as required) to the client device. The relationship between the host and client modules is shown in Figure 22-8.

Figure 22-8. SPIx Host/Client Connection Diagram



Notes:

1. Using the SSx pin in Client mode of operation is optional.
2. The user must write transmit data to SPIxBUF and read received data from SPIxBUF. The SPIxTXB and SPIxRXB registers are memory mapped to SPIxBUF.

22.4.1.5.1. Host Mode Operation

Perform the following steps to set up the SPI module for the Host mode operation:

1. Disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit.
3. Clear the receive buffer.
4. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
5. SPIx interrupts are not going to be used, skip this step. Otherwise, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.

   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
6. Write to the Baud Rate register, SPIxBRG.
7. Clear the SPIROV bit (SPIxSTAT[6]).
8. Write the desired settings to the SPIxCON1 register with MSTEN (SPIxCON1[5]) = 1.
9. Enable the SPI operation by setting the SPIEN bit (SPIxCON1[15]).
10. Write the data to be transmitted to the SPIxBUF register. Transmission (and reception) will start as soon as data are written to the SPIxBUF register.

Note: The SPI device must be turned OFF prior to changing the mode from Client to Host. (When using the Client Select mode, the $\overline{\text{SSx}}$ pin or another GPIO pin is used to control the Client's $\overline{\text{SSx}}$ input. The pin must be controlled in software).

In Host mode, the PBCLK is divided and then used as the serial clock. The division is based on the settings in the SPIxBRG register. The serial clock is output through the SCKx pin to the client devices. Clock pulses are only generated when there are data to be transmitted; except when in Framed mode, when the clock is generated continuously.

The Host Mode Client Select Enable (MSSEN) bit in the SPIx Control Register 1 (SPIxCON1[20]) can be set to automatically drive the Client Select signal ($\overline{\text{SSx}}$) in Host mode. Clearing this bit disables the Client Select signal support in Host mode. The FRMPOL bit (SPIxCON1[21]) determines the polarity for the Client Select signal in Host mode.

Note: The MSSEN bit is not available on all devices. This bit should not be set when the SPI Framed mode is enabled (i.e., FRMEN = 1).

In devices that do not feature the MSSEN bit, the Client Select signal (in Non-Framed SPI mode) must be generated by using the $\overline{\text{SSx}}$ pin or another general purpose I/O pin under software control.

The CKP (SPIxCON1[6]) and CKE (SPIxCON1[8]) bits determine on which edge of the clock data transmission occurs.

Note: The user must turn OFF the SPI device prior to changing the CKE or CKP bits. Otherwise, the behavior of the device is not ensured.

Both data to be transmitted and data that are received are written to, or read from, the SPIxBUF register, respectively.

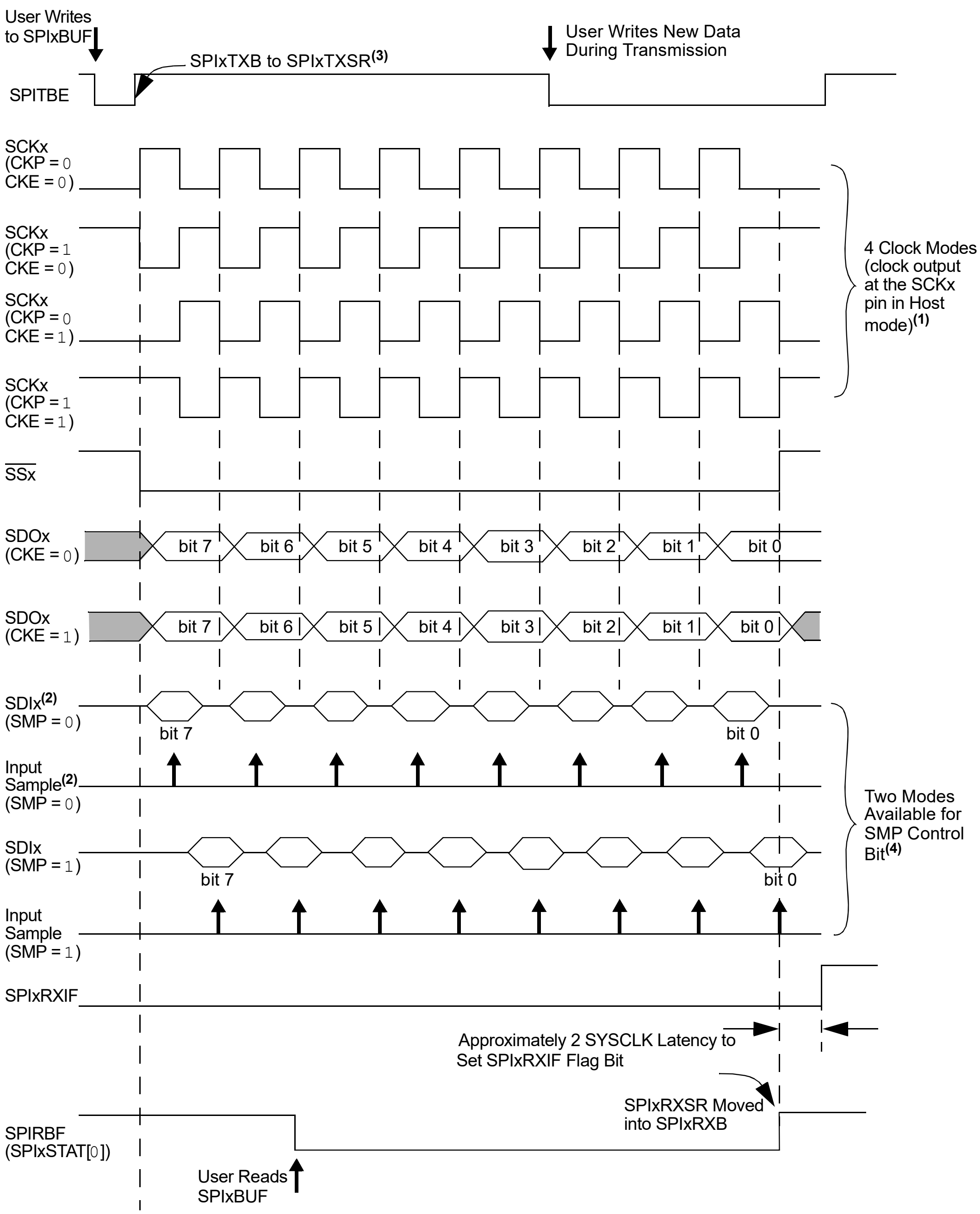
The following progression describes the SPI module operation in Host mode:

1. Once the module is set up for Host mode operation and enabled, data to be transmitted are written to the SPIxBUF register. The SPITBE bit (SPIxSTAT[3]) is cleared.
2. The contents of SPIxTXB are moved to the SPIx Shift register, SPIxTXSR (see Figure 22-9), and the SPITBE bit is set by the module.
3. A series of 8/16/32 clock pulses shifts 8/16/32 bits of transmit data from SPIxTXSR to the SDOx pin and simultaneously shifts the data at the SDIx pin into SPIxRXSR.
4. When the transfer is complete, the following events will occur:
   a. The SPIxRXIF interrupt flag bit is set. SPIx interrupts can be enabled by setting the SPIxRXIE interrupt enable bit. The SPIxRXIF flag is not cleared automatically by the hardware.
   b. Also, when the ongoing transmit and receive operation is completed, the contents of SPIxRXSR are moved to SPIxRXB.
   c. The SPIRBF bit (SPIxSTAT[0]) is set by the module, indicating that the receive buffer is full. Once SPIxBUF is read by the user code, the hardware clears the SPIRBF bit. In Enhanced Buffer mode, the SPIRBE bit (SPIxSTAT[5]) is set when the SPIxRXB FIFO buffer is completely empty and cleared when not empty.

5. If the SPIRBF bit is set (the receive buffer is full) when the SPI module needs to transfer data from SPIxRXSR to SPIxRXB, the module will set the SPIROV bit (SPIxSTAT[6]) indicating an overflow condition.
6. Data to be transmitted can be written to SPIxBUF by the user software at any time, if the SPITBE bit (SPIxSTAT[3]) is set. The write can occur while SPIxTXSR is shifting out the previously written data, allowing continuous transmission. In Enhanced Buffer mode, the SPITBF bit (SPIxSTAT[1]) is set when the SPIxTXB FIFO buffer is completely full and clear when it is not full.

Note: The SPIxTXSR register cannot be written directly by the user. All writes to the SPIxTXSR register are performed through the SPIxBUF register.

Figure 22-9. SPIx Host Mode Operation in 8-Bit Mode (MODE32 = 0, MODE16 = 0)



Notes:

1. Four SPI Clock modes are shown here to demonstrate the functionality of bits, CKP (SPIxCON1[6]) and CKE (SPIxCON1[8]). Only one of the four modes can be chosen for operation.
2. The SDIx and input samples shown here for two different values of the SMP bit (SPIxCON1[9]) are strictly for demonstration purposes. Only one of the two configurations of the SMP bit can be chosen during operation.
3. If there are no pending transmissions, SPIxTXB is transferred to SPIxTXSR as soon as the user writes to SPIxBUF.
4. Operation for the 8-bit mode is shown; 16-bit and 32-bit modes are similar.

Example 22-1. Initialization Code for 16-Bit SPI Host Mode

```
/* The following code example will initialize the SPI1 in Host mode. */
_SPI1TXIP = 4;                // Set SPI Interrupt Priorities
SPI1BRG = 0x1;                // use FSPICLK/4 clock frequency
SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1CON1 = 0x00000420;        // 16 bits transfer, Host mode,ckp=0,cke=0,smp=0
SPI1IMSKbits.SPITBFEN = 1;    // SPI1 transmit buffer full generates interrupt
event
_SPI1TXIE = 1;                // Enable interrupts
SPI1CON1bits.ON = 1;
// from here, the device is ready to transmit and receive data. Buffer can be
loaded to transmit data.
```

22.4.1.5.2. Client Mode Operation

The following steps are used to set up the SPI module for the Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
5. If using interrupts, Steps 6 through 12 are also performed:
6. Clear the SPIx interrupt flags/events in the respective IFSx register.
7. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
8. Set the SPIx interrupt enable bits in the respective IECx register.
9. Clear the SPIROV bit (SPIxSTAT[6]).
10. Write the desired settings to the SPIxCON1 register with MSTEN (SPIxCON1[5]) = `0`.
11. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start as soon as the host provides the serial clock.

Note: The SPI module must be turned OFF prior to changing the mode from Host to Client.

In Client mode, data are transmitted and received as the external clock pulses appear on the SCKx pin. The CKP bit (SPIxCON1[6]) and the CKE bit (SPIxCON1[8]) determine on which edge of the clock data transmission occurs.

Both data to be transmitted and data that are received are respectively written into or read from the SPIxBUF register.

The rest of the operation of the module is identical to that in the Host mode, including Enhanced Buffer mode.

Client Mode Additional Features

The following additional features are provided in the Client mode:

Client Select Synchronization

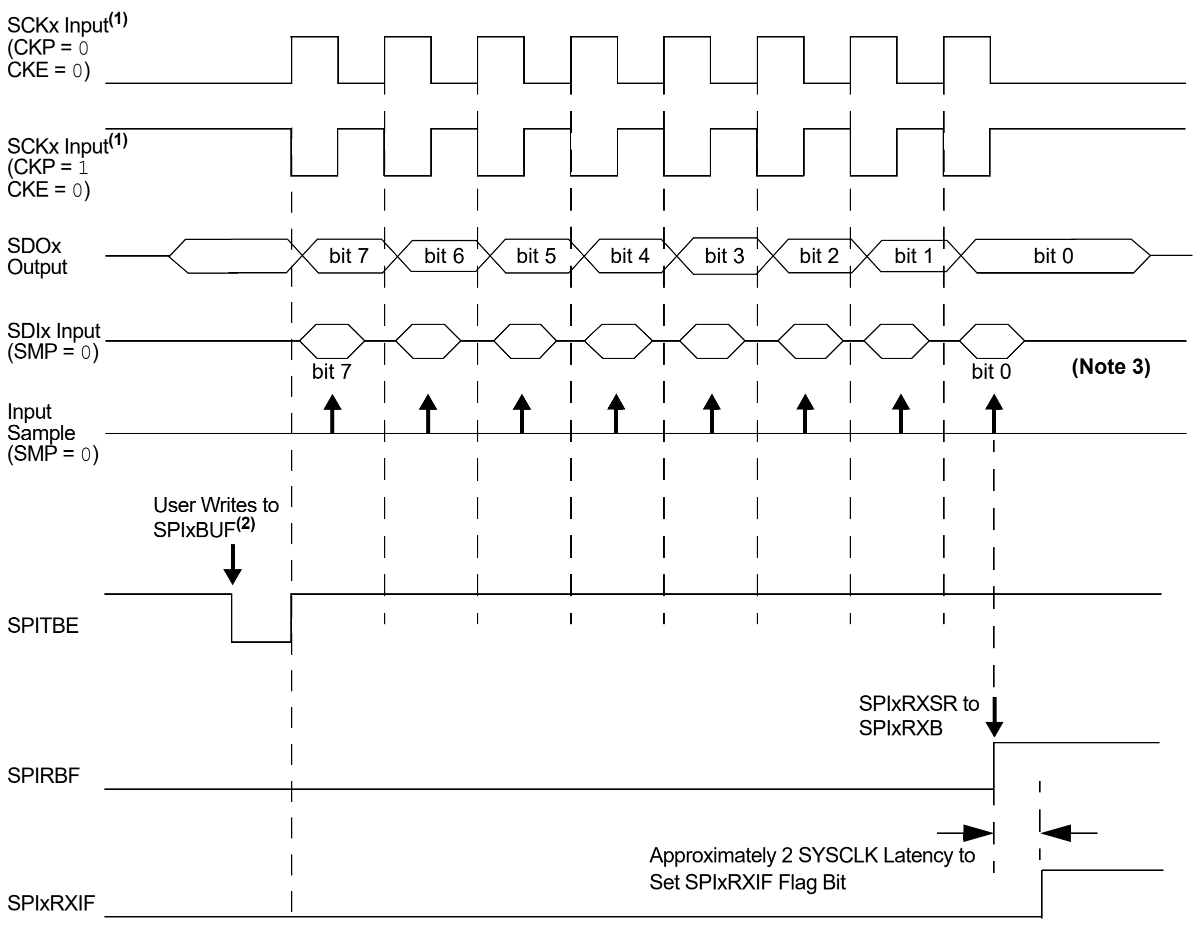
The SSx pin allows a Synchronous Client mode. If the SSEN bit (SPIxCON1[7]) is set, transmission and reception are enabled in Client mode only if the SSx pin is driven to a Low state. The port output or other peripheral outputs must not be driven in order to allow the SSx pin to function as an input. If the SSEN bit is set and the SSx pin is driven high, the SDOx pin is no longer driven and will tri-state, even if the module is in the middle of a transmission. An aborted transmission will be retried the next time the SSx pin is driven low using the data held in the SPIxTXB register. If the SSEN bit is not set, the SSx pin does not affect the module operation in Client mode.

SPITBE Status Flag Operation

The SPITBE bit (SPIxSTAT[3]) has a different function in the Client mode of operation. The following describes the function of SPITBE for various settings of the Client mode of operation:

- If SSEN (SPIxCON1[7]) is cleared, the SPITBE bit is cleared when SPIxBUF is loaded by the user code. It is set when the module transfers SPIxTXB to SPIxTXSR. This is similar to the SPITBE bit function in Host mode.
- If SSEN is set, SPITBE is cleared when SPIxBUF is loaded by the user code. However, it is set only when the SPI module completes data transmission. A transmission will be aborted when the SSx pin goes high and may be retried at a later time. So, each data word is held in SPIxTXB until all bits are transmitted to the receiver.
  Note: Client Select cannot be used when operating in Frame mode.

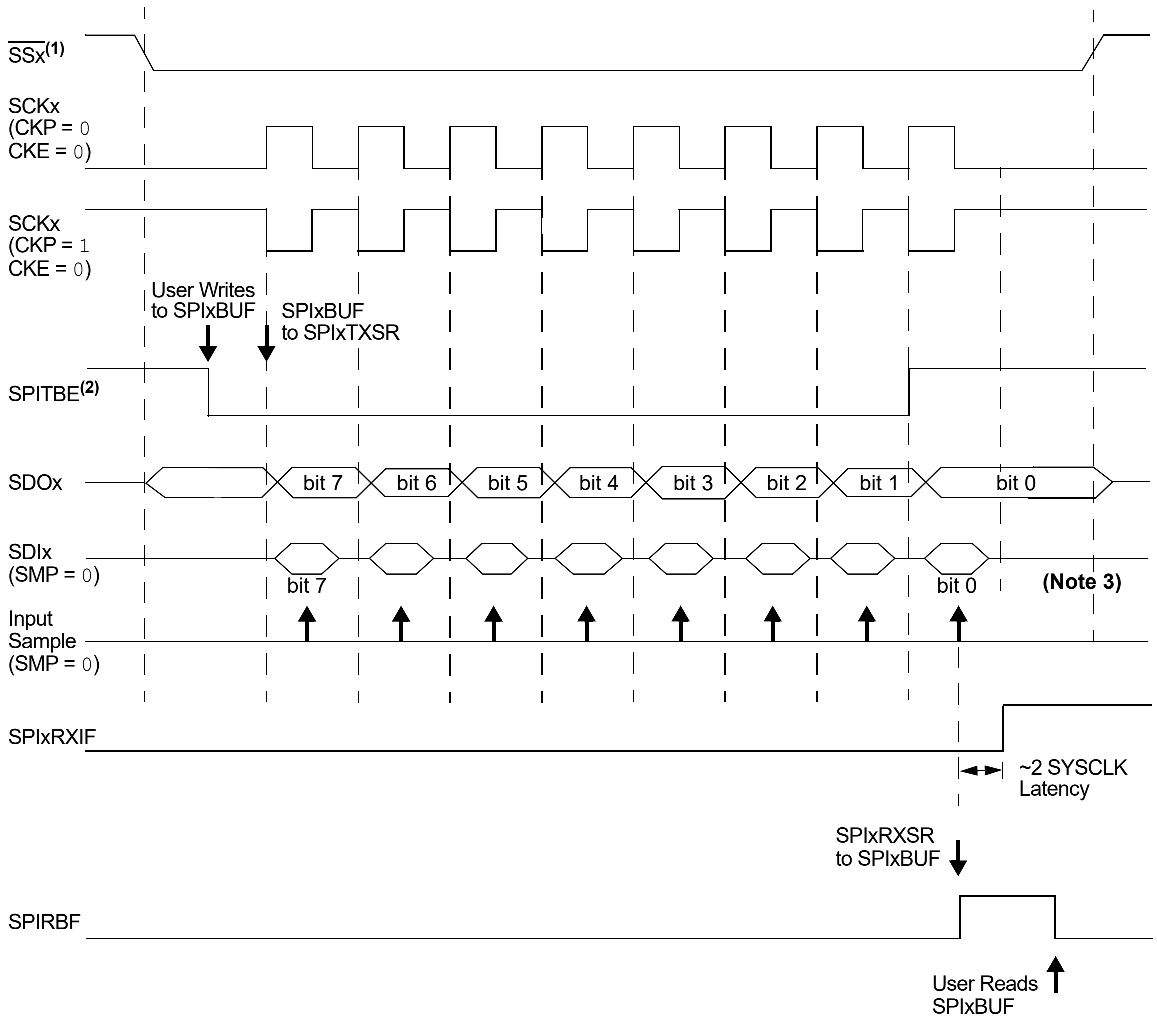
Figure 22-10. SPIx Client Mode Operation in 8-Bit Mode with Client Select Pin Disabled (MODE32 = 0, MODE16 = 0, SSEN = 0)



Notes:

1. Two SPI Clock modes are shown here only to demonstrate the functionality of bits, CKP (SPIxCON1[6]) and CKE (SPIxCON1[8]). Any combination of CKP and CKE bits can be chosen for module operation.
2. If there are no pending transmissions or a transmission is in progress, SPIxBUF is transferred to SPIxTXSR as soon as the user writes to SPIxBUF.
3. Operation for 8-bit mode is shown; 16-bit and 32-bit modes are similar.

Figure 22-11. SPIx Client Mode Operation in 8-Bit Mode with Client Select Pin Disabled (MODE32 = 0, MODE16 = 0, SSEN = 0)



Notes:

1. When the SSEN (SPIxCON1[7]) bit is set to '1', the $\overline{SSx}$ pin must be driven low so as to enable transmission and reception in Client mode.
2. Transmit data are held in SPIxTXB, and SPITBE (SPIxSTAT[3]) remains clear until all bits are transmitted.
3. Operation for 8-bit mode is shown; 16-bit and 32-bit modes are similar.

Example 22-2. Initialization Code for 16-Bit SPI Client Mode

```
/* The following code example will initialize the SPI1 in Client mode. */
_SPI1RXIP = 4;                    //Set SPI Interrupt Priorities
SPI1STATbits.SPIROV = 0;         // clear the Overflow
SPI1CON1 = 0x0400;               // 16 bits transfer, Client
mode,ckp=0,cke=0,smp=0
SPI1IMSKbits.SPIRBFEN = 1;       // SPI1 receive buffer full generates
interrupt event
_SPI1RXIE = 1;                   // Enable interrupts
SPI1CON1bits.ON = 1;
// from here, the device is ready to transmit and receive data. Buffer can be
loaded to
transmit data.
```

22.4.1.6. SPI Error Handling

When a new data word has been shifted into the SPIx Shift register, SPIxRXSR, and the previous contents of the SPIx Receive register, SPIxRXB, have not been read by the user software, the SPIROV bit (SPIxSTAT[6]) will be set. The module will not transfer the received data from SPIxRXSR to the SPIxRXB. Further data reception is disabled until the SPIROV bit is cleared. The SPIROV bit is not cleared automatically by the module and must be cleared by the user software.

22.4.1.7. SPI Receive Only Operation

Setting the DISSDO control bit (SPIxCON1[12]) disables transmission at the SDOx pin. This allows the SPI module to be configured for a Receive Only mode of operation. The SDOx pin will be controlled by the respective port function if the DISSDO bit is set.

The DISSDO function is applicable to all SPI operating modes.

22.4.1.8. Framed SPI Modes

The module supports a very basic framed SPI protocol while operating in either Host or Client mode. The following features are provided in the SPI module to support Framed SPI modes:

- The FRMEN control bit (SPIxCON1[23]) enables Framed SPI mode and causes the SSx pin to be used as a Frame Synchronization pulse input or output pin. The state of SSEN (SPIxCON1[7]) is ignored.
- The FRMSYNC control bit (SPIxCON1[22]) determines whether the $\overline{\text{SSx}}$ pin is an input or an output (i.e., whether the module receives or generates the Frame Synchronization pulse).
- The FRMPOL control bit (SPIxCON1[21]) determines the Frame Synchronization pulse polarity for a single SPI clock cycle.
- The FRMSYPW control bit (SPIxCON1[19]) can be set to configure the width of the Frame Synchronization pulse to one character wide.

The FRMCNT[2:0] control bits (SPIxCON1[18:16]) can be set to configure the number of data characters transmitted per Frame Synchronization pulse.

The following Framed SPI modes are supported by the SPI module:

Frame Host mode

The SPI module generates the Frame Synchronization pulse and provides this pulse to other devices at the $\overline{\text{SSx}}$ pin.

Frame Client mode

The SPI module uses a Frame Synchronization pulse received at the $\overline{\text{SSx}}$ pin.

The Framed SPI modes are supported in conjunction with the Host and Client modes. Therefore, the following Framed SPI mode configurations are available:

- SPI Host mode and Frame Host mode

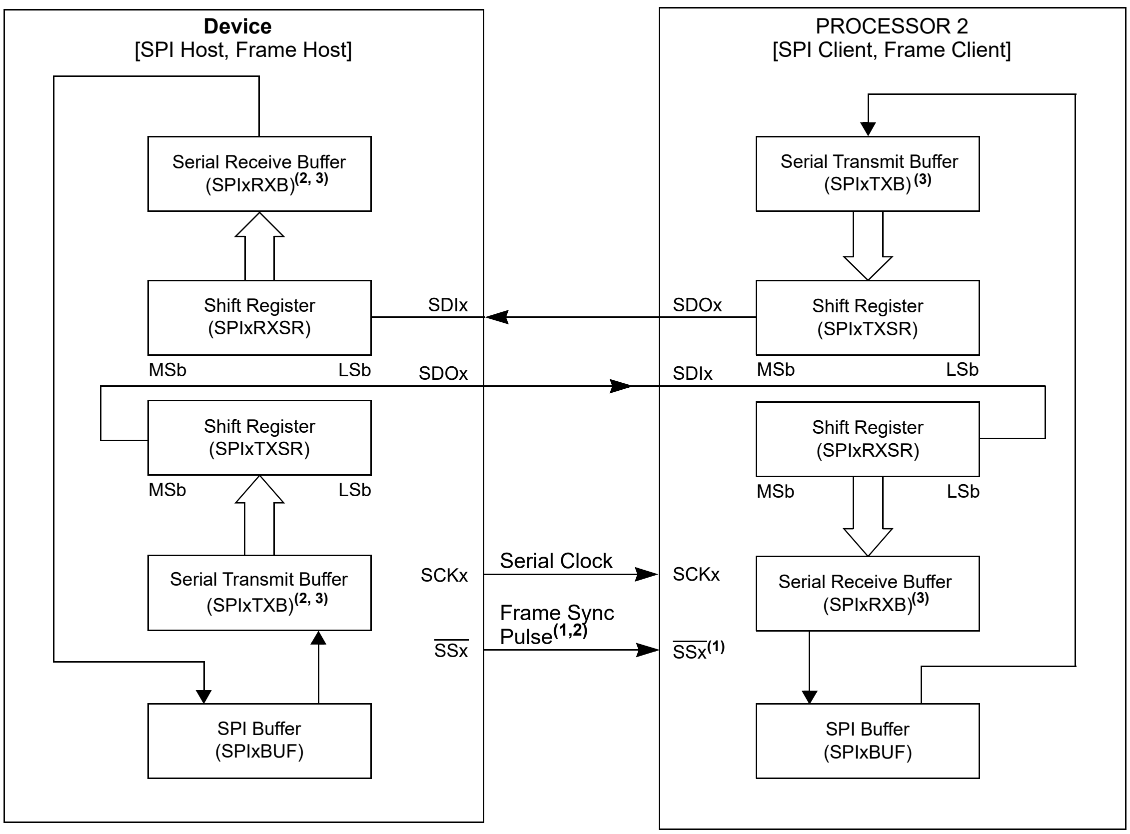
- SPI Host mode and Frame Client mode
- SPI Client mode and Frame Host mode
- SPI Client mode and Frame Client mode

These four modes determine whether or not the SPI module generates the serial clock and the Frame Synchronization pulse.

The ENHBUF bit (SPIxCON1[0]) can be configured to use the Standard Buffering mode or Enhanced Buffering mode in the Framed SPI mode.

In addition, the SPI module can be used to interface to external audio DAC/ADC and codec devices in the Framed SPI mode.

Figure 22-12. SPIx Host, Frame Host Connection Diagram



Notes:
1. In Framed SPI modes, the SSx pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI modes require the use of all four pins (i.e., using the SSx pin is not optional).
3. The SPIxTXB and SPIxRXB registers are memory mapped to the SPIxBUF register.

22.4.1.8.1. SCKx in Framed SPI Modes

When FRMEN (SPIxCON1[23]) = `1` and MSTEN (SPIxCON1[5]) = `1`, the SCKx pin becomes an output and the SPI clock at SCKx becomes a free-running clock.

When FRMEN = `1` and MSTEN = `0`, the SCKx pin becomes an input. The source clock provided to the SCKx pin is assumed to be a free-running clock.

The polarity of the clock is selected by the CKP bit (SPIxCON1[6]). The CKE bit (SPIxCON1[8]) is not used for the Framed SPI modes.

When CKP or CKE = 0, the Frame Sync pulse output and the SDOx data output change on the rising edge of the clock pulses at the SCKx pin. Input data is sampled at the SDIx input pin on the falling edge of the serial clock.

When CKP or CKE = 1, the Frame Sync pulse output and the SDOx data output change on the falling edge of the clock pulses at the SCKx pin. Input data is sampled at the SDIx input pin on the rising edge of the serial clock.

22.4.1.8.2. SPI Buffers in Framed SPI Modes

When FRMSYNC (SPIxCON1[22]) = 0, the SPI module is in the Frame Host mode of operation.

In this mode, the Frame Sync pulse is initiated by the module when the user software writes the transmit data to a SPIxBUF location (thus, writing the SPIxTXB register with transmit data). At the end of the Frame Sync pulse, SPIxTXB is transferred to SPIxTXSR and data transmission/reception begins.

When FRMSYNC = 1, the module is in Frame Client mode. In this mode, the Frame Sync pulse is generated by an external source. When the module samples the Frame Sync pulse, it will transfer the contents of the SPIxTXB register to SPIxTXSR, and data transmission/reception begins. The user must make sure that the correct data are loaded into the SPIxBUF for transmission before the Frame Sync pulse is received.

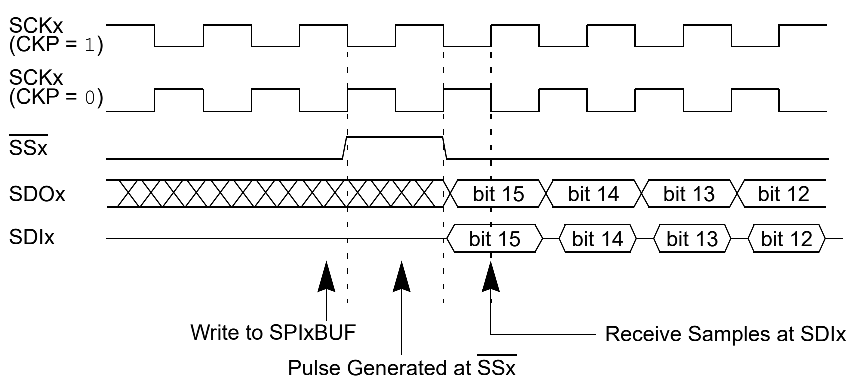
Note: Receiving a Frame Sync pulse will start a transmission, regardless of whether or not data were written to SPIxBUF. If a write was not performed, zeros will be transmitted.

22.4.1.8.3. SPI Host Mode and Frame Host Mode

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1[5]) and the FRMEN bit (SPIxCON1[23]) to '1', and the FRMSYNC bit (SPIxCON1[22]) to '0'. In this mode, the serial clock will be output continuously at the SCKx pin, regardless of whether the module is transmitting. When SPIxBUF is written, the $\overline{SSx}$ pin will be driven active-high or active-low, depending on the FRMPOL bit (SPIxCON1[21]), on the next transmit edge of the SCKx clock.

The $\overline{SSx}$ pin will be high for one SCKx clock cycle. The module will start transmitting data on the next transmit edge of SCKx, as shown in Figure 22-13. A connection diagram indicating signal directions for this operating mode is shown in Figure 22-13.

Figure 22-13. SPIx Host, Frame Host (MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)



22.4.1.8.4. SPI Host Mode and Frame Client Mode

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1[5]), the FRMEN bit (SPIxCON1[23]) and the FRMSYNC bit (SPIxCON1[22]) to '1'. The SSx pin is an input and it is sampled on the sample edge of the SPI clock. When it is sampled active-high or active-low, depending on the FRMPOL bit (SPIxCON1[21]), data will be transmitted on the subsequent transmit edge of the SPI clock, as shown in Figure 22-14. The SPIx Interrupt Flag, SPIxIF, is set when the transmission is complete. The user must make sure that the correct data are loaded into SPIxBUF for transmission before the signal is received at the SSx pin. A connection diagram indicating signal directions for this operating mode is shown in Figure 22-15.

Figure 22-14. SPIx Host, Frame Client (MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)

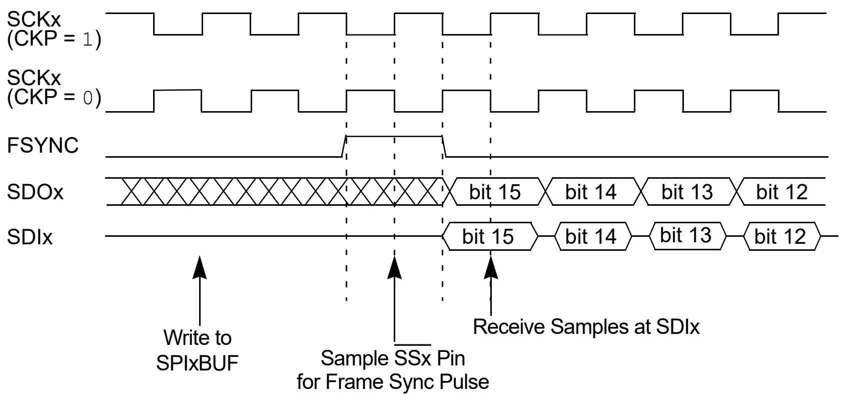
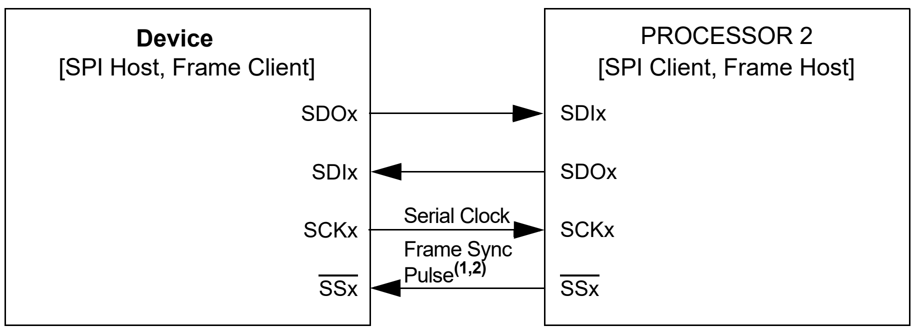


Figure 22-15. SPIx Host, Frame Client Connection Diagram



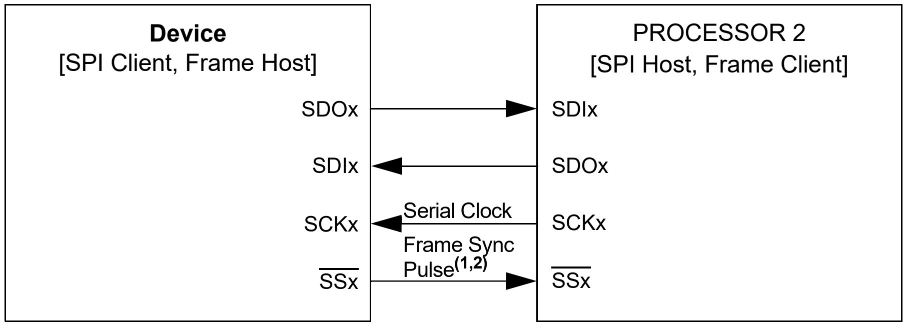
Notes:

1. In Framed SPI modes, the SSx pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI modes require the use of all four pins (i.e., using the SSx pin is not optional).

22.4.1.8.5. SPI Client Mode and Frame Host Mode

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1[5]) to '0', the FRMEN bit (SPIxCON1[23]) to '1' and the FRMSYNC bit (SPIxCON1[22]) to '0'. The input SPI clock will be continuous in Client mode. The SSx pin will be an output when bit, FRMSYNC, is low. Therefore, when SPIxBUF is written, the module will drive the SSx pin active-high or active-low, depending on the FRMPOL bit (SPIxCON1[21]), on the next transmit edge of the SPI clock. The SSx pin will be driven high for one SPI clock cycle. Data transmission will start on the next SPI clock transmit edge. A connection diagram indicating signal directions for this operating mode is shown in Figure 22-16.

Figure 22-16. SPIx Client, Frame Host Connection Diagram



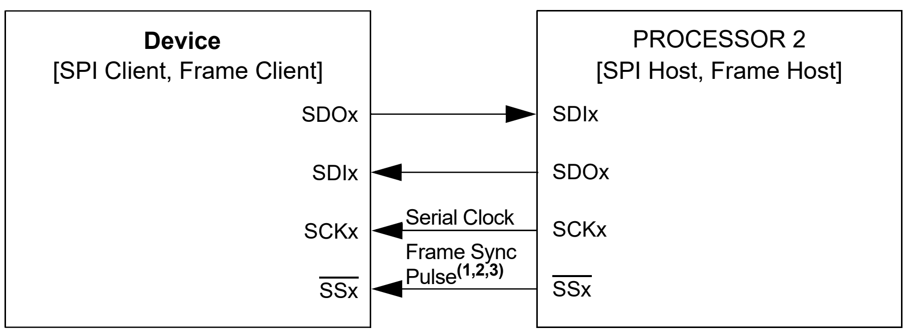
Notes:

1. In Framed SPI modes, the $\overline{\text{SSx}}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI modes require the use of all four pins (i.e., using the $\overline{\text{SSx}}$ pin is not optional).

22.4.1.8.6. SPI Client Mode and Frame Client Mode

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1[5]) to ‘0’, the FRMEN bit (SPIxCON1[23]) to ‘1’ and the FRMSYNC bit (SPIxCON1[22]) to ‘1’. Therefore, both the SCKx and $\overline{\text{SSx}}$ pins will be inputs. The $\overline{\text{SSx}}$ pin will be sampled on the sample edge of the SPI clock. When $\overline{\text{SSx}}$ is sampled active-high or active-low, depending on the FRMPOL bit (SPIxCON1[21]), data will be transmitted on the next transmit edge of SCKx. A connection diagram indicating signal directions for this operating mode is shown in Figure 22-17.

Figure 22-17. SPIx Client, Frame Client Connection Diagram



Notes:

1. In Framed SPI modes, the $\overline{\text{SSx}}$ pin is used to transmit/receive the Frame Synchronization pulse.
2. Framed SPI modes require the use of all four pins (i.e., using the $\overline{\text{SSx}}$ pin is not optional).
3. Client Select is not available when using Frame mode as a client device.

22.4.2. Audio Protocol Interface Mode

The SPI module can be interfaced to most available codec devices to provide dsPIC33A microcontroller-based audio solutions. The SPI module provides support to the audio protocol functionality through four standard I/O pins. The four pins that make up the audio protocol interface modes are:

- SDIx: Serial Data Input for receiving sample Digital Audio Data (ADCDAT)
- SDOx: Serial Data Output for transmitting Digital Audio Data (DACDAT)
- SCKx: Serial Clock, also known as the Bit Clock (BCLK)
- $\overline{\text{SSx}}$: Left/Right Channel Clock (LRCK)

BCLK provides the clock required to drive the data out or into the module, while LRCK provides the synchronization of the frame based on the Protocol mode selected.

In some codecs, Serial Clock (SCK) refers to the Baud/Bit Clock (BCLK). Throughout this section, the signal, $\overline{SSx}$, is to be referred to as LRCK to be consistent with codec naming conventions. The SPI module has the ability to function in Audio Protocol Host and Audio Protocol Client modes. In Host mode, the module generates both the BCLK on the SCKx pin and the LRCK on the $\overline{SSx}$ pin. In certain devices, while in Client mode, the module receives these two clocks from its I^{2}S partner, which is operating in Host mode.

While in Host mode, the SPI module has the ability to generate its own clock internally through the Host Clock (MCLK) from various internal sources, such as the primary clock, PBCLK, USB clock, FRC and other internal sources. In addition, the SPI module has the ability to provide the MCLK to the codec device, which is a common requirement.

To start the Audio Protocol mode, first disable the peripheral by setting the ON bit (SPIxCON1[15]) = `0`. Next, set the AUDEN bit (SPIxCON1[31]) = `1` and then re-enable the peripheral by setting the ON bit = `1`.

When configured in Host mode, the leading edge of SCKx and the LRCK is driven out within one SCKx period of starting the audio protocol. Serial data are shifted in or out with timing determined by the Protocol mode set by the AUDMOD[1:0] bits (SPIxCON1[25:24]). If the transmit FIFO is empty, zeros are transmitted.

In Client mode, the peripheral drives zeros out of SDOx but does not transmit the contents of the transmit FIFO until it sees the leading edge of the LRCK, after which time it starts receiving data (provided SDIx has not been disabled). It will continue to transmit zeros as long as the transmit FIFO is empty.

While in Client or Host mode, the SPI module does not generate an underrun on the TX FIFO after start-up. This allows software to set up the SPI, set up the DMA, turn on the SPI module's audio protocol and then turn on the DMA without getting an error.

After the first write to the TX FIFO (SPIxBUF), the SPI enables underrun detection and generation. To keep the RX FIFO empty until the DMA is enabled, set DISSDI (SPIxCON1[4]) = `1`. After enabling the DMA, set DISSDI = `0` to start receiving.

22.4.2.1. Host Mode

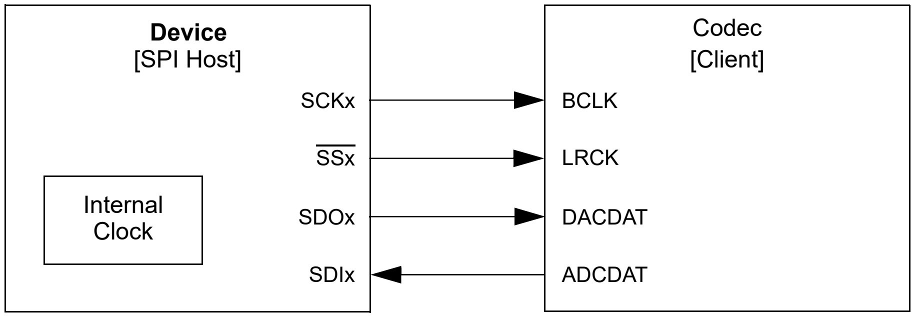
To configure the SPI module device in Audio Protocol Host mode, set both the MSTEN bit (SPIxCON1[5]) and the AUDEN bit (SPIxCON1[31]) to '`1`'.

A few characteristics of Host mode are:

- This mode enables the device to generate SCKx and LRCK pulses as long as the ON bit (SPIxCON1[15]) = `1`.
- The SPI module generates LRCK and SCKx continuously in all cases, regardless of the transmit data while in Host mode.
- The SPI module drives the leading edge of LRCK and SCKx within one SCKx period, and the serial data shift in and out continuously, even when the TX FIFO is empty.

Figure 22-18 shows a typical interface between Host and Client while in Host mode.

Figure 22-18. Host Generating its Own Clock – Output BCLK and LRCK



22.4.2.2. Client Mode

The SPI module can be configured in Audio Protocol Client mode by setting the MSTEN bit = `0` (SPIxCON1[5]) and the AUDEN bit = `1` (SPIxCON1[31]).

A few characteristics of Client mode are:

- This mode enables the device to receive SCKx and LRCK pulses as long as the ON bit (SPIxCON1[15]) = `1`.
- The SPI module drives zeros out of SDOx, but does not shift data out or in (SDIx) until the module receives the LRCK (i.e., the edge that precedes the left channel).
- Once the module receives the leading edge of LRCK, it starts receiving data if DISSDI (SPIxCON1[4]) = `0` and the serial data shift out continuously, even when the TX FIFO is empty.

Figure 22-19 shows the interface between a SPI module in Audio Client Interface mode to a codec host device.

Figure 22-19. Codec Device as Host Generates Required Clock via External Crystal

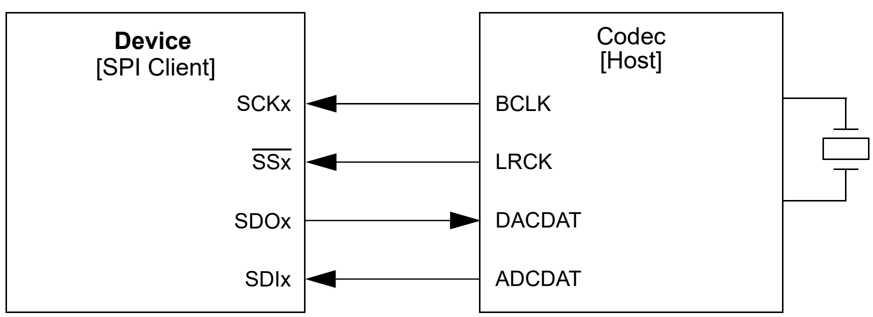
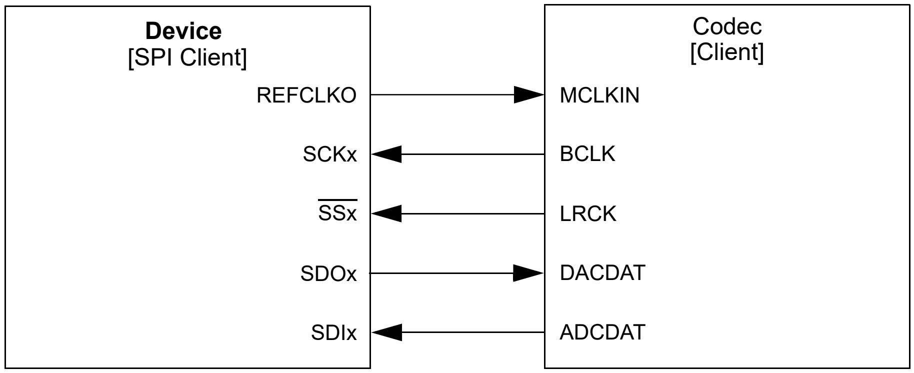


Figure 22-20 shows the interface between an SPI module in Audio Client Interface mode to a codec host device, in which the host clock is being derived from the SPI reference clock out function.

Figure 22-20. Codec Device as Host Derives MCLK from dsPIC33A Reference Clock Out



22.4.2.3. Audio Data Length and Frame Length

While codec devices may generate audio data samples of various word lengths of 8, 16, 20, 24 and 32, the dsPIC33A SPI module supports transmit/receive audio data lengths of 16, 24 and 32.
Note: Actual sample data can be any length, with a maximum of 32 bits, and the data must be packed in one of three (16/24/32) formats.

Table 22-8 illustrates how the MODE[32,16] bits (SPIxCON1[11:10]) control the maximum allowable sample length and frame length (LRCK period on SSx).

Table 22-8. Audio Data Length vs. LRCK Period

SPIxCON1[11:10]		Data Length (bits)	FIFO Width (bits)	Left/Right Channel Sample Length (bits)	Enhanced Buffer FIFO Depth (samples)(1)	LRCK Period Frame Length (bits)
MODE32	MODE16					
0	0	16	16	≤ 16	X/2	32
0	1	16	16	≤ 32	X/2	64
1	1	24	32	≤ 32	X/4	64
1	0	32	32	≤ 32	X/4	64

Note: FIFO depth varies between devices. The data shown are specified considering a device with an available FIFO depth of 'X'.

The parameters of the MODE[32,16] bits (SPIxCON1[11:10]) have the following behavior:

- Controls left/right channel data length, frame length.
- In 16-bit Sample mode, a 32/64-bit frame length is supported.
- In 24/32-bit Sample mode, a 64-bit frame length is supported.
- Defines FIFO width and depth (for example, 24-bit data have a 32-bit wide and X/4-location deep FIFO).
- If the written data are greater than the data selected, the upper bytes are ignored.
- If the written data are less than the data selected, the FIFO Pointers change on the write to the MSB of the selected length.

If this data are written to the transmit FIFO in more than one write, the write order must be from Least Significant to Most Significant bytes.

For example, assume that audio data are 24 bits per sample with 8 bits available at a time. According to Table 22-8, the FIFO width is 32 bits per sample. Therefore, the 8 MSbs, bits[31:24], in each FIFO sample are ignored.

Data written to unused bytes are ignored. Also, transactions that are only to unused bytes are also ignored. Therefore, a byte write to address offset, 0x0023, is completely ignored and does not cause a FIFO push if the data are less than 32 bits wide.

22.4.2.4. Frame Error/LRCK Errors

The SPI module provides detection of frame/LRCK errors for debugging. The frame/LRCK error occurs when the LRCK edge that is defining a channel start happens before the correct number of bits (as defined by MODE[32,16]).

The SPI module immediately sets the FRMERR bit (SPIxSTAT[12]), pushes data in from the SPIxRXSR register into the SPIxRXB register and pops data from the SPIxTXB register into the SPIxTXSR register. The module can be configured to detect frame/LRCK related errors by setting the FRMERREN bit (SPIxIMSK[12]).

Note: In Audio Protocol mode, both the BCLK (on the SCKx pin) and the LRCK (on the $\overline{\text{SSx}}$ pin) are free running, meaning they are continuous. Normally, the LRCK is a fixed number of BCLKs long. In all cases, the SPI module will realign to the new frame edge and will set the FRMERR bit. If operating in a Non-PCM mode, the SPI module will also push the abbreviated data onto the FIFO when the frame is too short.

22.4.2.5. Audio Protocol Modes

The SPI module supports four Audio Protocol modes and can be operated in any one of these modes:

- I^2S mode
- Left Justified mode
- Right Justified mode
- PCM/DSP mode

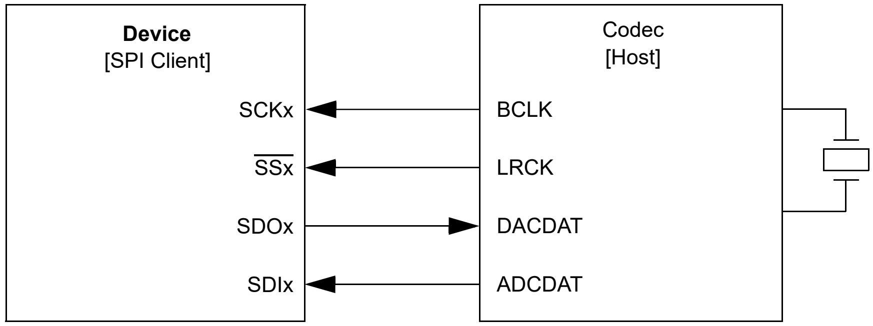
These Audio Protocol modes can be enabled by configuring the AUDMOD[1:0] bits (SPIxCON1[25:24]). These modes enable communication to different types of codecs and control the edge relationships of LRCK and SDIx/SDOx with respect to SCKx.

With respect to data transmit, in all of the Protocol modes, the MSb is first transmitted followed by MSB-1, and so on, until the LSb transmits. The length of the data is discussed in Audio Data Length and Frame Length. If there are SCKx periods left over after the LSb is transmitted, zeros are sent to fill up the frame.

When in Client mode, the relationship between the BCLK (on the SCKx pin) and the period (or frame length) of the LRCK (on the $\overline{\text{SSx}}$ pin) is far less constrained than that of Host mode. In Host mode, the frame length equals 32 or 64 BCLKs, depending on the MODE[32,16] (SPIxCON1[11:10]) bit settings. However, in Client mode, the frame length can be greater than or equal to 32 or 64 BCLKs, but the FRMERR bit (SPIxSTAT[12]) will be set if the frame LRCK edge arrives early.

Figure 22-21 illustrates the general interface between the codec device and the SPI module in Audio mode.

Figure 22-21. SPIx Module in Audio Client Mode – BCLK and WS or LRCK Generated by Host



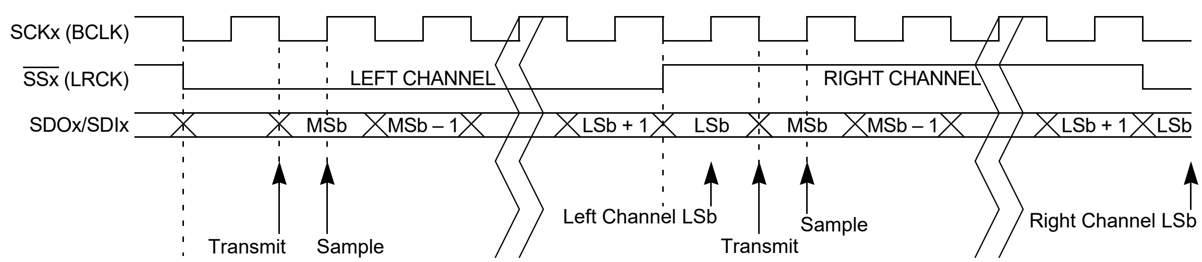
22.4.2.5.1. I²S Mode

The Inter-IC Sound (I²S) protocol enables transmission of two channels of digital audio data over a single serial interface. The I²S protocol defines a 3-wire interface that handles the stereo data using the WS/LRCK line. The I²S specification defines a half-duplex interface that supports transmit or receive but not both at the same time. With both SDOx and SDIx available, full-duplex operation is supported by this peripheral, as shown in Figure 22-22.

- Data Transmit and Clocking:
  - The transmitter shifts the audio sample data's MSb on the first falling edge of SCKx, after an LRCK transition.
  - The receiver samples the MSb on the second rising edge of SCKx.
  - The left channel data shift out while LRCK is low, and the right channel data are shifted out while LRCK is high.
  - The data in the left and right channels consist of a single frame.
- Required Configuration Settings:
  To set the module to I²S mode, the following bits must be set:
  - AUDMOD[1:0] = 00 (SPIxCON1[25:24])
  - FRMPOL = 0 (SPIxCON1[21])
  - CKP = 1 (SPIxCON1[6])

Setting these bits enables the SDOx and LRCK ($\overline{SSx}$) transitions to occur on the falling edge of SCKx (BCLK) and sampling of SDIx to occur on the rising edge of SCKx. Refer to the diagram shown in Figure 22-22.

Figure 22-22. I²S with 16-Bit Data/Channel or 32-Bit Data/Channel



I²S Audio Client Mode of Operation

Use the following steps to set up the SPI module for the I²S Audio Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps need to be performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTAT[6]).

8. Write the desired settings to the SPIxCON1 register.
   a. AUDMOD[1:0] bits (SPIxCON1[25:24]) = `00`
   b. AUDEN bit (SPIxCON1[15]) = `1`
9. Write the desired settings to the SPIxCON1 register:
   a. MSTEN (SPIxCON1[5]) = `0`
   b. CKP (SPIxCON1[6]) = `1`
   c. MODE[32,16] (SPIxCON1[11:10]) = `0` for 16-bit audio channel data
   d. Enable SPI operation by setting the ON bit (SPIxCON1[31]).
10. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Example 22-3. I^{2}S Client Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in I2S Client
mode. */
_SPI1RXIP = 4;
SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1CON1=0x80000440;          // AUDEN=1, I2S mode, stereo mode,
                              // 16 bits/32 channel transfer, Client mode,CKP
=1
SPI1IMSKbits.SPIRBFEN = 1;    // SPI1 receive buffer full generates interrupt
event
_SPI1RXIE = 1;                // Enable interrupts
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

I^{2}S Audio Host Mode of Operation

A typical application could be to play PCM data (8 kHz sample frequency, 16-bit data, 32-bit frame size) when interfaced to a codec client device. In this case, the SPI module is initialized to generate BCLK @ 625 kbps. Assuming a 20 MHz peripheral clock, F_{SPICLK} = 20e6, the baud rate would be determined using Equation 22-2.

Equation 22-2. Baud Rate Calculation

$$Baud\ Rate = \frac{F_{SPICLK}}{2 \cdot (SPIxBRG + 1)}$$

Solving for the value of SPIxBRG is shown in Equation 22-3.

Equation 22-3. Baud Rate Calculation

$$SPIxBRG = \frac{F_{SPICLK}}{2(Baud\ Rate)} - 1$$

The Baud Rate is now equal to 625e3. Equation 22-4 shows the resulting calculation.

Equation 22-4. Baud Rate Calculation

$$SPIxBRG = \frac{20e6}{2(625e3)} - 1 = 15$$

If the result of Equation 22-4 is rounded to the nearest integer, SPIxBRG is now equal to 15; therefore, the effective Baud Rate is that of Equation 22-5.

Equation 22-5. Baud Rate Calculation

$$\frac{20e6}{2 \cdot (15+1)} = \frac{20e6}{32} = 625000\ bits\ per\ second$$

The following steps can be used to set up the SPI module for operation in I²S Audio Host mode:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Reset the SPIx Baud Rate Register, SPIxBRG.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
7. If using interrupts, perform these additional steps:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTAT[6]).
9. Write the desired settings to the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to ‘00’ for I²S mode, and the AUDEN bit (SPIxCON1[31]) must be set to ‘1’ to enable the audio protocol.
10. Set the SPIxBRG register to 0x0F (to generate approximately 625 kbps sample rate with SPICLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1 register:
    a. MSTEN (SPIxCON1[5]) = 1
    b. CKP (SPIxCON1[6]) = 1
    c. MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data
    d. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start immediately after the SPIEN bit is set.

Example 22-4. I²S Host Mode, 625 kbps BCLK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in I2S Host mode. */
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1BRG = 0x000F;             // to generate 625 kbps sample rate, SPICLK @ 20 MHz
SPI1CON1=0x80000460;          // AUDEN=1, I2S mode, stereo mode,
                              // 16 bits/32 channel transfer, Host mode, CKP = 1
SPI1IMSKbits.SPITBFEN = 1;    // SPI1 transmit buffer full generates interrupt event
_SPI1TXIE = 1;                // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

22.4.2.5.2. Left Justified Mode

The Left Justified mode is similar to I2S mode, however, in this mode, the SPI shifts the audio data’s MSb on the first SCKx edge that is coincident with an LRCK transition. On the receiver side, the SPI module samples the MSb on the next SCKx edge.

In general, a codec using justified protocols defaults to transmitting data on the rising edge of SCKx and receiving data on the falling edge of SCKx.

- Required Configuration Settings
  To set the module to Left Justified mode, the following bits must to be set:
  - AUDMOD[1:0] = `01` (SPIxCON1[25:24])
  - FRMPOL = `1` (SPIxCON1[21])
  - CKP = `0` (SPIxCON1[6])

This enables the SDOx and LRCK transitions to occur on the rising edge of SCKx. Refer to the sample waveform diagrams shown in Figure 22-23 and Figure 22-24 for 16, 24 and 32-bit audio data transfers.

Figure 22-23. Left Justified with 16-Bit Data/Channel or 32-Bit Data/Channel

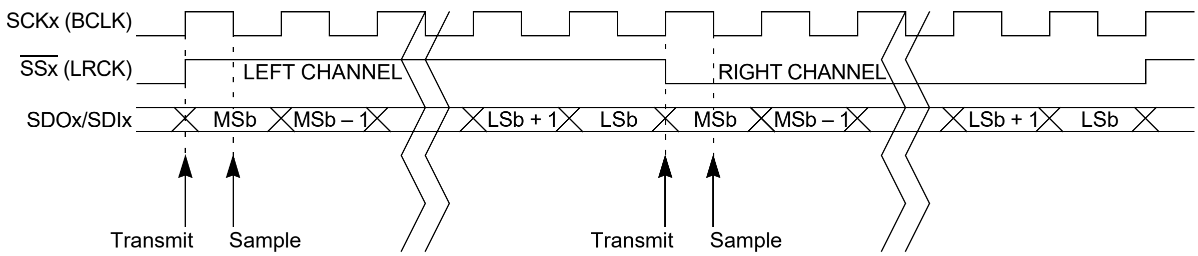
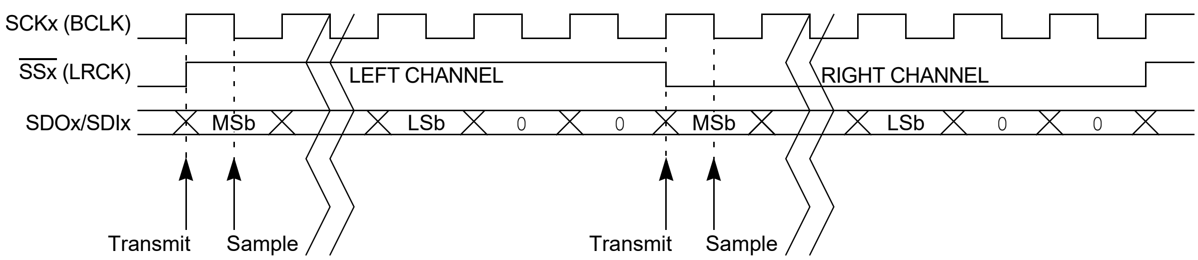


Figure 22-24. Left Justified with 16/24-Bit Data and 32-Bit Channel



Left Justified Audio Client Mode Operation

Use the following steps to set up the SPI module for the Left Justified Audio Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTAT[6]).
8. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '`01`' for Left Justified mode, and the AUDEN bit (SPIxCON1[31]) must be set to '`1`' to enable the audio protocol.
9. Write the desired settings to the SPIxCON1 register:

   a. Set to Client mode, MSTEN (SPIxCON1[5]) = 0.
   b. Set Clock Polarity, CKP (SPIxCON1[6]) = 0.
   c. Set Frame Polarity, FRMPOL (SPIxCON1[5]) = 1.
   d. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
   e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
10. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Example 22-5. Left Justified Client Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Left
Justified Client mode. */
_SPI1RXIP = 4;
SPI1STATbits.SPIROV = 0;          // clear the Overflow
SPI1CON1=0x81200400;              // AUDEN=1, Left justified mode, stereo
mode,
                                  // FRMPOL = 1,16 bits/32 channel transfer,
SPI1IMSKbits.SPIRBFEN = 1;        // SPI1 receive buffer full generates
interrupt event
_SPI1RXIE = 1;                    // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

Left Justified Audio Host Mode Operation

Use the following steps to set up the SPI module for the Left Justified Audio Host mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control register 1, SPIxCON1.
4. Reset the SPIx Baud Rate register, SPIxBRG.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
7. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTAT[6]).
9. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '01' for left justified, and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register, SPIxBRG, to 0x0F (to generate approximately 625 kbps sample rate with SPICLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1 register:
   a. Set to Host mode, MSTEN (SPIxCON1[5]) = 1.
   b. Set Clock Polarity, CKP (SPIxCON1[6]) = 0.
   c. Set Frame Polarity, FRMPOL (SPIxCON1[21]) = 1.
   d. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
   e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start immediately after the ON bit is set.

Example 22-6. Left Justified Host Mode, 625 kbps BLCK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Left
Justified Host mode. */
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;     // clear the Overflow
SPI1BRG = 0x000F;            // to generate 625 kbps sample rate, SPICLK @ 20
MHz
SPI1CON1 = 0x81200420;       // AUDEN =1, Left Justified mode, stereo mode,
                             // FRMPOL = 1, 16 bits/32 channel transfer,Host
mode,CKP = 0
SPI1IMSKbits.SPITBFEN = 1;   // SPI1 transmit buffer full generates interrupt
event
_SPI1TXIE = 1;               // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

22.4.2.5.3. Right Justified Mode

In Right Justified mode, the SPI module shifts the audio sample data's MSb after aligning the data to the last clock cycle. The bits preceding the audio sample data can be driven to logic level '0' by setting the DISSDO bit (SPIxCON1[12]) to '0'. When DISSDO = 0, the module ignores the unused bit slot.

- Required Configuration Settings
  To set the module to Right Justified mode, the following bits must to be set:
  - AUDMOD[1:0] (SPIxCON1[25:24]) = 10
  - FRMPOL (SPIxCON1[21]) = 1
  - CKP (SPIxCON1[6]) = 0

This enables the SDOx and LRCK transitions to occur on the rising edge of SCKx, after the LSb is aligned to the last clock cycle. Refer to the sample waveform diagrams shown in Figure 22-25 and Figure 22-26 for 16, 24 and 32-bit audio data transfers.

Figure 22-25. Right Justified with 16-Bit Data/Channel or 32-Bit Data/Channel

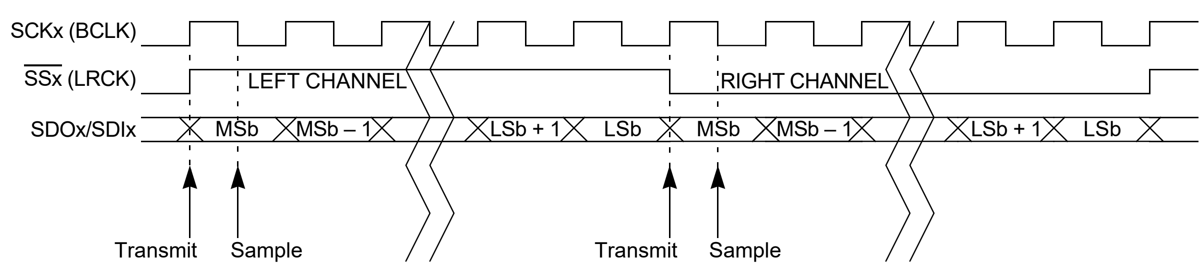
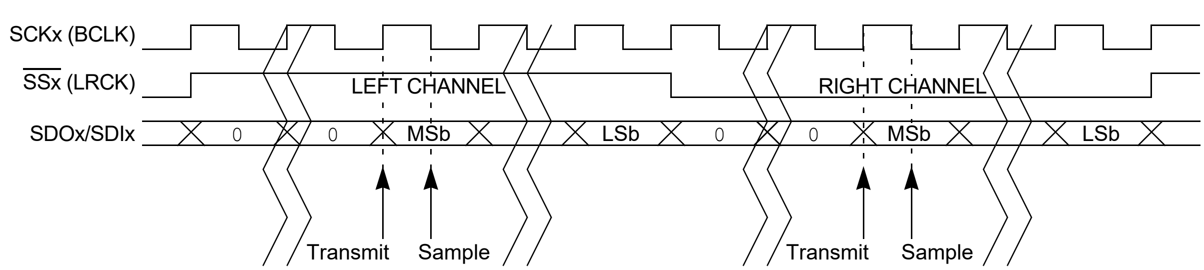


Figure 22-26. Right Justified with 16/24-Bit Data and 32-Bit Channel



Right Justified Audio Client Mode Operation

Use the following steps to set up the SPI module for the Right Justified Audio Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
6. If using interrupts, perform the following steps:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTAT[6]).
8. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '10' for Right Justified mode, and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
9. Write the desired settings to the SPIxCON1 register:
   a. Set to Client mode, MSTEN (SPIxCON1[5]) = 0.
   b. Set Clock Polarity, CKP (SPIxCON1[6]) = 0.
   c. Set Frame Polarity, FRMPOL (SPIxCON1[21]) = 1.
   d. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
   e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
10. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Example 22-7. Right Justified Client Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Right
Justified Client mode. */
_SPI1RXIP = 4;
SPI1STATbits.SPIROV = 0;       // clear the Overflow
SPI1CON1 = 0x82200400;         // AUDEN=1, Right Justified mode, stereo
mode,FRMPOL=1,
                               // 16 bits/32 channel transfer,Client mode,ckp=0
SPI1IMSKbits.SPIRBFEN = 1;     // SPI1 receive buffer full generates interrupt
event
_SPI1RXIE = 1;                 // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

Right Justified Audio Host Mode Operation

Use the following steps to set up the SPI module for the Right Justified Audio Host mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Reset the SPIx Baud Rate Register, SPIxBRG.
5. Clear the receive buffer.

6. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
7. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTAT[6]).
9. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '10' for Right Justified mode, and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register, SPIxBRG, to 0x0F (to generate approximately 625 kbps sample rate with SPICLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1 register:
    a. Set to Host mode, MSTEN (SPIxCON1[5]) = 1.
    b. Set Clock Polarity, CKP (SPIxCON1[6]) = 0.
    c. Set Frame Polarity, FRMPOL (SPIxCON1[21]) = 1.
    d. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
    e. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start immediately after the ON bit is set.

Example 22-8. Right Justified Host Mode, 625 kbps BLCK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Right
Justified Host mode. */
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;        // clear the Overflow
SPI1BRG = 0x000F;               // to generate 625 kbps sample rate, SPICLK @
20 MHz
SPI1CON1 = 0x82200420;          // AUDEN=1, Right Justified mode,stereo mode,
                                // 16 bits/32 channel transfer,Host mode,ckp=0
SPI1IMSKbits.SPITBFEN = 1;      // SPI1 transmit buffer full generates
interrupt event
_SPI1TXIE = 1;                  // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

22.4.2.5.4. PCM/DSP Mode

The PCM/DSP Protocol mode is available for communication with some codecs and certain DSP (Digital Signal Processor) devices. This mode modifies the behavior of LRCK and audio data spacing. In PCM/DSP mode, the LRCK can be a single bit wide (i.e., 1 SCKx) or as wide as the audio data (16, 24, 32 bits). The audio data is packed in the frame with the left channel data immediately followed by the right channel data. The frame length is still either 32 or 64 clocks when this device is the host.

In PCM/DSP mode, the transmitter drives the audio data's (left channel) MSb on the first or second transmit edge (see the SPIFE bit (SPIxCON1[1])) of SCKx (after an LRCK transition). Immediately after the (left channel) LSb, the transmitter drives the (right channel) MSb.

- Required Configuration Settings:
  To set the module to Left Justified mode, the following bits must to be set:
  – AUDMOD[1:0] bits (SPIxCON1[25:24]) = 11

Refer to the sample waveform diagrams shown in Figure 22-27 and Figure 22-28 for 16, 24 and 32-bit audio data transfers.

Figure 22-27. PCM/DSP with 16-Bit Data/Channel or 32-Bit Data/Channel

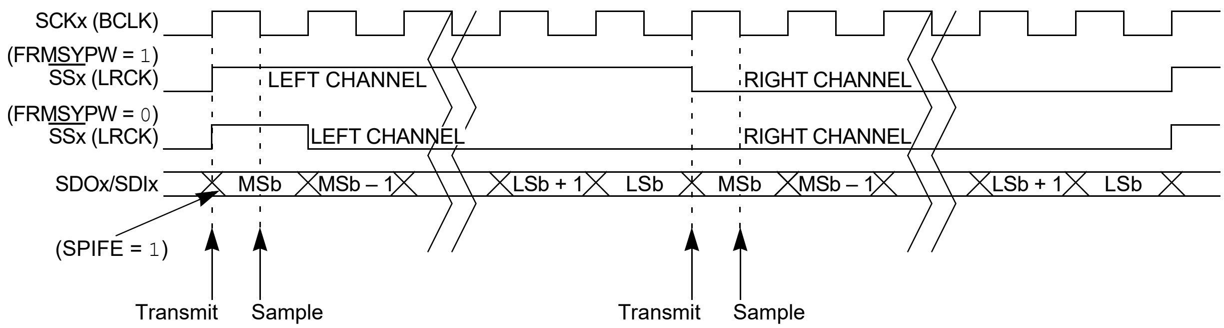
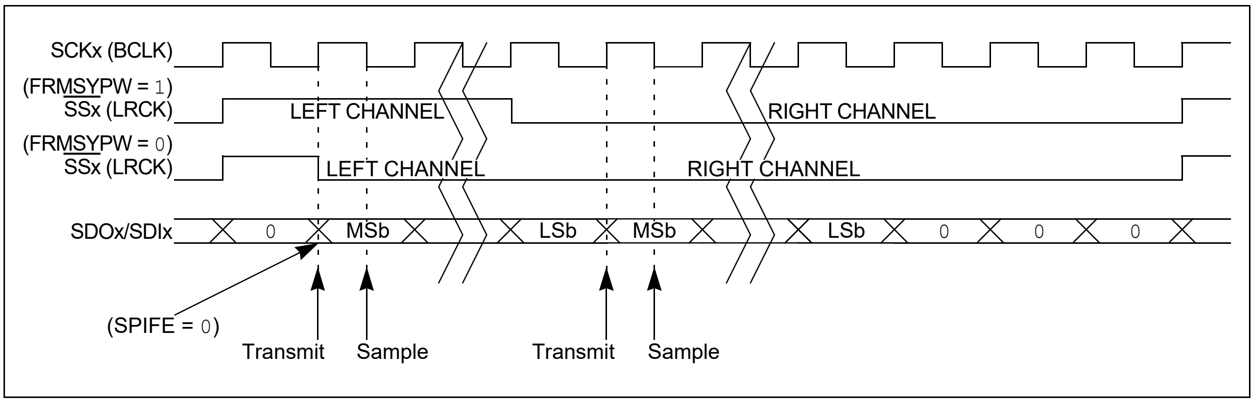


Figure 22-28. PCM/DSP with 16/24-Bit Data and 32-Bit Channel



PCM/DSP Audio Client Mode of Operation

Use the following steps to set up the SPI module for the PCM/DSP Audio Client mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps are performed:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTAT[6]).
8. Write the desired setting in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to ‘11’ for DSP/PCM mode, and the AUDEN bit (SPIxCON1[31]) must be set to ‘1’ to enable audio protocol.
9. Write the desired settings to the SPIxCON1 register:
   a. Set to Client mode, MSTEN (SPIxCON1[5]) = 0.
   b. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
   c. Enable SPI operation by setting the ON bit (SPIxCON1[15]).

10. Transmission (and reception) will start as soon as the host provides the BCLK and LRCK.

Example 22-9. PCM/DSP Client Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in PCM/DSP
Client Mode. */
_SPI1RXIP = 4;
SPI1STATbits.SPIROV = 0;        // clear the Overflow
SPI1CON1 = 0x83200400;          // AUDEN=1,PCM/DSP mode,stereo mode,FRMPOL=1,
                                // 16 bits/32 channel transfer,Client mode,ckp=0
SPI1IMSKbits.SPIRBFEN = 1;      // SPI1 receive buffer full generates interrupt
event
_SPI1RXIE = 1;                  // Enable interrupts
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

PCM/DSP Audio Host Mode of Operation

Use the following steps to set up the SPI module for the PCM/DSP Audio Host mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the ON bit (SPIxCON1[15]).
3. Reset the SPIx Control Register 1, SPIxCON1.
4. Reset the SPIx Baud Rate Register, SPIxBRG.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1[0]) if using Standard Buffer mode, or set the bit if using Enhanced Buffer mode.
7. If using interrupts, perform the following steps:
   a. Clear the SPIx interrupt flags/events in the respective IFSx register.
   b. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
   c. Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTAT[6]).
9. Write the desired settings in the SPIxCON1 register. The AUDMOD[1:0] bits (SPIxCON1[25:24]) must be set to '11' for DSP/PCM mode, and the AUDEN bit (SPIxCON1[31]) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register, SPIxBRG, to 0x0F (to generate approximately 625 kbps sample rate with SPICLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1 register:
    a. Set to Host mode, MSTEN (SPIxCON1[5]) = 1.
    b. Set MODE[32,16] (SPIxCON1[11:10]) = 0 for 16-bit audio channel data.
    c. Enable SPI operation by setting the ON bit (SPIxCON1[15]).
12. Transmission (and reception) will start immediately after the ON bit is set.

Example 22-10. PCM/DSP Host Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in PCM/DSP Host
Mode. */
_SPI1TXIP = 4;
SPI1STATbits.SPIROV = 0;       // clear the Overflow
SPI1BRG = 0x000F;              // to generate 625 kbps sample rate, PBCLK @ 20
MHz
SPI1CON1 = 0x8320;             // AUDEN=1, PCM/DSP mode,stereo mode,FRMPOL=1
SPI1CON1 = 0x0400;             // 16 bits/32 channel transfer,Host mode,ckp=0
SPI1IMSKbits.SPITBFEN = 1;     // SPI1 transmit buffer full generates interrupt
event
_SPI1TXIE=1;                   // Enable interrupts
```

```
SPI1CON1bits.ENHBUF = 1;
SPI1CON1bits.ON = 1;
// from here, the device is ready to receive and transmit data
```

22.4.2.6. Audio Protocol Mode Features

22.4.2.6.1. BCLK/SCKx and LRCK Generation

BCLK and LRCK generation is a key requirement in Host mode. The frame frequency of SCKx and LRCK is defined by the MODE[32,16] bits (SPIxCON1[11:10]). When the frame is 64 bits, SCKx is 64 times the frequency of LRCK. Similarly, when the frame is 32 bits, SCKx is 32 times the frequency of LRCK. The frequency of SCKx must be derived from the toggling rate of LRCK and the frame size.

For example, to sample 16-bit channel data at 8 kHz with SPICLK = 36.864 MHz, set the SPIxBRG register to 0x47 to generate an 8 kHz LRCK.

22.4.2.6.2. Host Mode Clocking and MCLK

The SPI module as a host has the ability to generate BCLK and LRCK by internally generating using SPICLK (MCLKEN = 0). In addition, the 33A device can generate the clock for external codec devices using the reference output, REFCLKO function (see Figure 22-30), although some codecs may have the ability to generate their own MCLK from a crystal to provide accurate audio sample rates.

Figure 22-30 shows that the REFCLKO clock can be used as MCLKIN by the codec.

Figure 22-29. SPIx Host Clock Generation

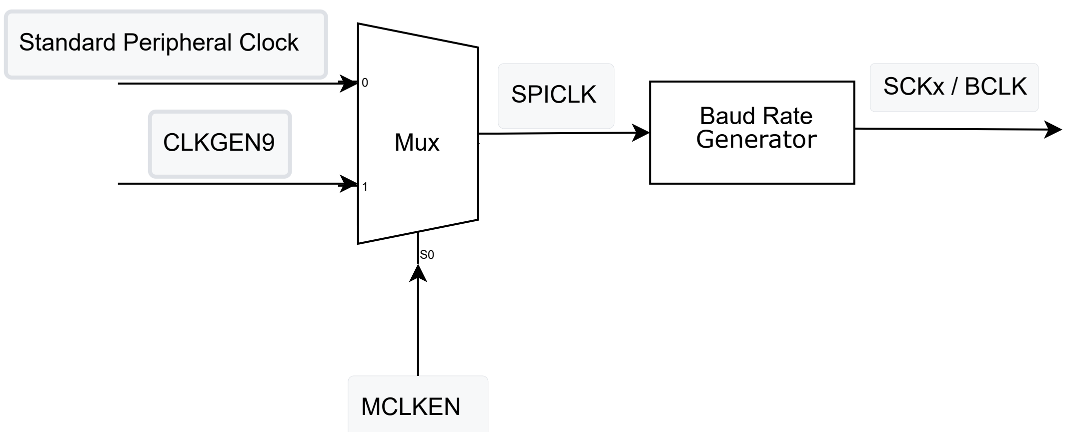
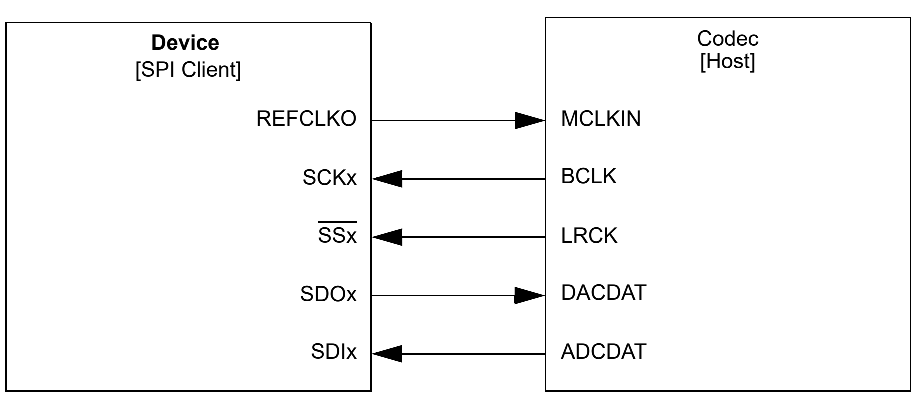


Figure 22-30 shows the interface between an SPI client and a codec host, deriving the clock from the MCLK input interface.

Figure 22-30. SPIx Client and Codec Host – Clock Derived from MCLK



22.4.2.7. Mono Mode vs. Stereo Mode

The SPI module enables the audio data transmission in Mono or Stereo mode by setting the AUDMONO bit (SPIxCON1[27]). When the AUDMONO bit is set to '0' (Stereo mode), the SPIx Shift register uses each FIFO location once, which gives each channel a unique stream of data for stereo data. When the AUDMONO bit is set to '1' (Mono mode), the SPIx Shift register uses each FIFO location twice to give each channel the same mono stream of audio data.

Note: Receive data are not affected by AUDMONO bit settings.

22.4.2.8. Streaming Data Support and Error Handling

Most audio streaming applications transmit or receive data continuously. This is required to keep the channel active during the period of operation and ensures the best possible accuracy. Due to streaming audio, the data feeds could be bursty or packet loss can occur causing the module to encounter underrun. The software needs to be involved to recover from an underrun.

The Ignore Transmit Underrun (IGNTUR) bit (SPIxCON1[28]), when set to a '1', ignores an Underrun condition. This is helpful for cases when software does not care or does not need to know about underrun conditions. When an underrun is encountered, the SPI module sets the SPITUR bit (SPIxSTAT[8]), and when URDTEN = 1 (SPIxCON1[26]), the module remains in an Error state until the software clears the state or the ON bit = 0 (SPIxCON1[15]).

During the Underrun condition, the SPI module loads the SPIxTXSR with the data in the SPIxURDT register when the URDTEN bit is set to '1'. If URDTEN is not set to '1', then the last received data during underrun are loaded to SPIxTXSR. The module samples the Underrun condition on channel boundaries, so transmission of SPIxURDT data can start with either the left or right audio channel.

When the condition clears (i.e., SPIxTXB is not empty), the logic loads audio data from the transmit buffer into the SPIxTXSR on the next LRC frame boundary. Because recovery from the Underrun condition occurs on the LRC frame boundary (i.e., at the end of a full left and right channel pair), software must ensure the left and right audio data are always transferred to the FIFO in pairs.

The Ignore Receive Overflow (IGNROV) bit (SPIxCON1[29]), when set to a '1', ignores a Receive Overflow condition. This is useful when there is a general performance problem in the system that software must handle. An alternate method to handle the Receive Overflow is by setting the DISSDI bit = 1 (SPIxCON1[4]) when the system does not need to receive audio data. After changing the DISSDI bit on-the-fly, the SPIx Receive Shift register starts a receive on the leading LRCK edge.

If an RX overflow occurs when IGNROV = 0, the I^{2}S will behave just like it would in SPI mode, that is, it will stop writing to the RX FIFO. However, recovery from overflow is different from SPI mode. When the CPU gets around to reading the RX FIFO, the I^{2}S will restart receiving into the RX FIFO only when two additional conditions are met:

- The I^{2}S is on an LRC boundary.
- There are a multiple of two locations free in the RX FIFO.

These conditions will ensure the received data will start at the beginning of the LEFT channel and there is room to receive the RIGHT channel information immediately following.

22.5. Interrupts

The SPI module has the ability to generate interrupts reflecting the events that occur during the data communication. The following types of interrupts can be generated:

- Receive data available interrupts are signaled by SPIxRXIF. This event occurs when:
  - RX watermark interrupt
  - SPIROV = 1
  - SPIRBF = 1
  - SPIRBE = 1, provided respective mask bits are enabled in SPIxIMSK

- Transmit buffer empty interrupts are signaled by SPIxTXIF. This event occurs when:
  - TX watermark interrupt
  - SPITUR = 1
  - SPITBF = 1
  - SPITBE = 1, provided respective mask bits are enabled in SPIxIMSK
- General or Error interrupts are signaled by SPIxEIF. This event occurs when:
  - FRMERR = 1
  - BUSY = 1
  - SRMT = 1, provided respective mask bits are enabled in SPIxIMSK

All of these interrupt flags, which must be cleared in software, are located in the IFSx registers. Refer to the **"Interrupt Controller"** chapter for details. To enable the SPIx interrupts, use the respective SPIx Interrupt Enable bits, SPIxRXIE, SPIxTXIE and SPIxEIE, in the corresponding IECx registers. The Interrupt Priority Level (IPL) bits must be also be configured using the SPIxIP bits in the corresponding IPCx registers.

When using Enhanced Buffer mode, the SPIx transmit buffer can be configured to interrupt at different FIFO levels using mask bits, TXMSK[2:0] in SPIxIMSK[18:16]. Also, the Transmit Watermark Interrupt bit, TXWIEN (SPIxIMSK[23]), should be enabled.

Similarly, the SPIx receive buffer can be configured to interrupt at different FIFO levels using mask bits, RXMSK[2:0] (SPIxIMSK[26:24]). Also, the Receive Watermark Interrupt, RXWIEN (SPIxIMSK[31]), should be enabled.

22.5.1. Interrupt Configuration

Each SPI module has three dedicated interrupt flag bits: SPIxIF, SPIxRXIF and SPIxTXIF, and corresponding interrupt enable bits, SPIxIE, SPIxRXIE and SPIxTXIE. These bits are used to determine the source of an interrupt and to enable or disable an individual interrupt source. Note that all the interrupt sources for a specific SPI module share one interrupt vector. Each SPI module can have its own priority level independent of other SPI modules.

Note: SPIxTXIF, SPIxRXIF and SPIxIF bits will be set without regard to the state of the corresponding enable bit. The interrupt flag bits can be polled by software, if desired.

The SPIxIE, SPIxTXIE and SPIxRXIE bits are used to define the behavior of the interrupt controller when a corresponding SPIxIF, SPIxTXIF or SPIxRXIF bit is set. When the corresponding interrupt enable bit is clear, the interrupt controller does not generate a CPU interrupt for the event. If the interrupt enable bit is set, the interrupt controller will generate an interrupt to the CPU when the corresponding interrupt flag bit is set (subject to the priority).

It is the responsibility of the user's software routine that services a particular interrupt to clear the appropriate interrupt flag bit before the service routine is complete.

The priority of each SPI module can be set independently with the SPIxIP[2:0] bits. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of seven (the highest priority), to a value of zero (which does not generate an interrupt). An interrupt being serviced will be preempted by an interrupt in a higher priority group.

The priority group bits allow more than one interrupt source to share the same priority. If simultaneous interrupts occur in this configuration, the natural order of the interrupt sources within a priority group pair determines the interrupt generated. The natural priority is based on the vector numbers of the interrupt sources. The lower the vector number, the higher the natural priority of the interrupt. Any interrupts that were overridden by natural order will then generate their respective interrupts based on priority and natural order, after the interrupt flag for the current interrupt is cleared.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The user's code at this vector address should perform any application-specific operations required and clear interrupt flags, SPIxIF, SPIxTXIF or SPIxRXIF, and then exit.

With Enhanced Buffering mode, the user application should clear the interrupt request flag after servicing the interrupt condition.

If an SPIx interrupt has occurred, the ISR should read the SPIx Data Buffer (SPIxBUF) register and then clear the SPIx interrupt flag.

22.6. Power-Saving and Debug Modes

22.6.1. Sleep Mode

When the device enters Sleep mode, the system clock is disabled. The exact SPI module operation during Sleep mode depends on the current mode of operation. The following sub-sections describe mode-specific behavior.

22.6.1.1. Host Mode in Sleep Mode

The following items should be noted in Sleep mode:

- The Baud Rate Generator (BRG) is stopped and may be reset.
- On-going transmission and reception sequences are aborted. The module may not resume aborted sequences when Sleep mode is exited.
- Once in Sleep mode, the module will not transmit or receive any new data.

Note: To prevent unintentional abort of transmit and receive sequences, wait for the current transmission to be completed before activating Sleep mode.

22.6.1.2. Client Mode in Sleep Mode

In Client mode, the SPI module operates from the SCKx provided by an external SPI host. Since the clock pulses at SCKx are externally provided for Client mode, the module will continue to function in Sleep mode. It will complete any transactions during the transition into Sleep. On completion of a transaction, the SPIRBF flag is set. Consequently, the SPIxRXIF bit will be set.

If SPIx interrupts are enabled (SPIxRXIE = `1`) and the SPI Interrupt Priority Level is greater than the present CPU priority level, the device will wake from Sleep mode and the code execution will resume at the SPIx interrupt vector location. If the SPI Interrupt Priority Level is lower than or equal to the present CPU priority level, the CPU will remain in Idle mode.

The module is not reset on entering Sleep mode if it is operating as a client device. Register contents are not affected when the SPI module is going into or coming out of Sleep mode.

22.6.2. Idle Mode

When the device enters Idle mode, the system clock sources remain functional.

22.6.2.1. Host Mode in Idle Mode

Bit, SPISIDL (SPIxCON1[13]), selects whether the module will stop or continue functioning in Idle mode.

- If SPISIDL = `1`, the module will discontinue operation in Idle mode. The module will perform the same procedures when stopped in Idle mode that it does for Sleep mode.
- If SPISIDL = `0`, the module will continue operation in Idle mode.

22.6.2.2. Client Mode in Idle Mode

The module will continue operation in Idle mode irrespective of the SPISIDL bit setting. The behavior is identical to the one in Sleep mode.

22.6.3. Debug Mode

22.6.3.1. Operation of SPIxBUF

22.6.3.1.1. Reads During Debug Mode

During Debug mode, SPIxBUF can be read, but the read operation does not affect any status bits. For example, if the SPIRBF bit (SPIxSTAT[0]) is set when Debug mode is entered, it will remain set on an exit from Debug mode, even though the SPIxBUF register was read in Debug mode.

23. Inter-Integrated Circuit (I^2C)

The Inter-Integrated Circuit (I^2C) module is a serial interface useful for communicating with other peripheral or microcontroller (MCU) devices. The external peripheral devices may be serial EEPROMs, display drivers, ADC and so on.

The I^2C module can operate as any one of the following in the I^2C system:

- Client Device
- Host Device in a Single Host System (Client May Be Active)
- Host or Client Device in a Multi-Host System (Bus Collision Detection and Arbitration are Available)

Key features of the I^2C module include the following:

- Independent Host and Client Logic
- Supports 100 kHz, 400 kHz and 1MHz Bus Specifications
- 7-bit and 10-bit Device Addresses.
- Client Mode can be Configured for:
  - Two unique addresses
  - Range of addresses
  - General call address
  - Address bit masking
- Automatic Clock Stretching Provides Delays for the Processor to Respond to a Client Data Request
- Multi-Host Support Which Prevents Message Losses in Arbitration
- Smart Mode for Minimal User Interaction and Simpler Application Code
- Supports Data Hold Time for SMBus (300 nS or 150 nS) in Client Mode
- Supports SMBus v2.0 and v3.0 Input Voltage Levels.
- Supports the Intelligent Platform Management Interface (IPMI) Standard
- System Management Bus (SMBus) and Power Management Bus (PMBus) Support:
  - Packet Error Checking (PEC) using CRC-8 calculator
  - Clock low time-out, Bus idle time-out and cumulative time-out
  - Frame error detection

23.1. Device-Specific Information

Table 23-1. I^2C Summary Table

I^2C Module Instances	Clock Source	Peripheral Bus Speed
2	Standard (1:2 CPU Clock)	Standard

23.2. Architectural Overview

The I^2C bus is a two-wire serial interface. Figure 23-1 illustrates the schematic of an I^2C connection between a dsPIC33AK256MPS306 device and a 24LC256 I^2C serial EEPROM, which is a typical example for any I^2C interface.

The I^2C interface uses a comprehensive protocol to ensure reliable transmission and reception of the data. When communicating, one device acts as the host, initiates transfer on the bus and generates the clock signals to permit that transfer, while the other devices act as the client responding to the transfer. The clock line, SCLx, is the output from the host and the input to the client, although occasionally the client drives the SCLx line. The data line, SDAx, may be the output and the input from both the host and client.

Because the SDAx and SCLx lines are bidirectional, the output stages of the devices driving the SDAx and SCLx lines must have an open-drain in order to perform the wired-AND function of the bus. External pull-up resistors are used to ensure a high level when no device is pulling the line down.

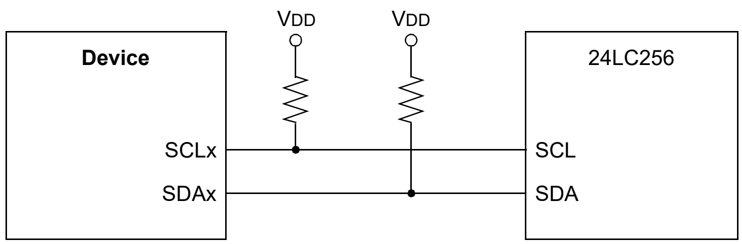
Note: SCLx and SDAx must be configured as digital.

In the I^{2}C interface protocol, each device has an address. When a host needs to initiate a data transfer, it first transmits the address of the device that it wants to communicate. All the devices listen to see if this is their address. Within this address, bit 0 specifies whether the host wants to read from or write to the client device. The host and client are always in opposite modes (Transmitter or Receiver) of operation during a data transfer. That is, they operate in either of the following two relationships:

- Host-transmitter and client-receiver
- Client-transmitter and host-receiver

In both cases, the host originates the SCLx clock signal.

Figure 23-1. Typical I^{2}C Interconnection Block Diagram



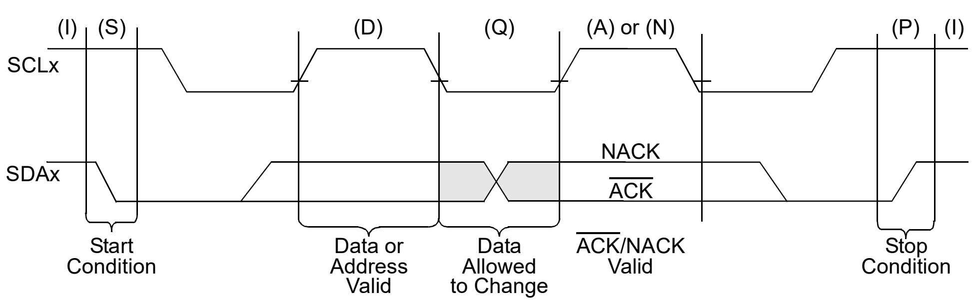
23.2.1. Bus Protocol

The following I^{2}C bus protocol has been defined:

- The data transfer may be initiated only when the bus is not busy.
- During the data transfer, the data line must remain stable whenever the SCLx clock line is high. Any changes in the data line while the SCLx clock line is high will be interpreted as a Start or Stop condition.

Accordingly, the bus conditions are defined as illustrated in Figure 23-2.

Figure 23-2. I^{2}C Bus Protocol States



23.2.1.1. Start Data Transfer (S)

After a bus Idle state, a high-to-low transition of the SDAx line while the clock (SCLx) is high determines a Start condition. All data transfers must be preceded by a Start condition.

23.2.1.2. Stop Data Transfer (P)

A low-to-high transition of the SDAx line while the clock (SCLx) is high determines a Stop condition. All data transfers must end with a Stop condition.

23.2.1.3. Repeated Start (R)

After a Wait state, a high-to-low transition of the SDAx line while the clock (SCLx) is high determines a Repeated Start condition. Repeated Starts allow a host to change bus direction or address a client device without relinquishing control of the bus.

23.2.1.4. Data Valid (D)

After a Start condition, the state of the SDAx line represents valid data when the SDAx line is stable for the duration of the high period of the clock signal. There is one bit of data per SCLx clock.

23.2.1.5. Acknowledge (A) or Not Acknowledge (N)

All data byte transmissions must be Acknowledged ($\overline{ACK}$) or Not Acknowledged (NACK) by the receiver. The receiver will pull the SDAx line low for an $\overline{ACK}$ or release the SDAx line for a NACK. The Acknowledge is a 1-bit period using one SCLx clock.

23.2.1.6. Wait/Data Invalid (Q)

The data on the line must be changed during the low period of the clock signal. The devices may also stretch the clock low time by asserting a low on the SCLx line, causing a Wait on the bus.

23.2.1.7. Bus Idle (I)

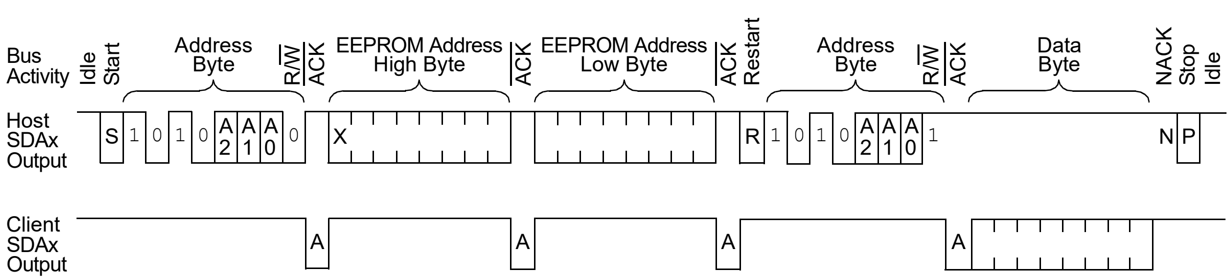
Both data and clock lines remain high after a Stop condition and before a Start condition.

23.2.2. Message Protocol

A typical I^{2}C message is illustrated in Figure 23-3. In this example, the message will read a specified byte from a 24LC256 I^{2}C serial EEPROM. The I^{2}C device will act as the host, and the 24LC256 device will act as the client.

Figure 23-3 illustrates the data as driven by the host device and the client device, taking into account that the combined SDAx line is a wired-AND of the host and client data. The host device controls and sequences the protocol. The client device will only drive the bus at specifically determined times.

Figure 23-3. A Typical I^{2}C Message: Read of Serial EEPROM (Random Address Mode)



23.2.2.1. Start Message

Each message is initiated with a Start condition and terminated with a Stop condition. The number of data bytes transferred between the Start and Stop conditions is determined by the host device. As defined by the system protocol, the bytes of the message may have special meaning, such as the device address byte or the data byte.

23.2.2.2. Address Client

In Figure 23-3, the first byte is the device address byte, which must be the first part of any I^{2}C message. It contains a device address and a R/W status bit. Note that R/W = 0 for this first address byte indicates that the host will be a transmitter and the client will be a receiver.

23.2.2.3. Client Acknowledge

The receiving device is obliged to generate an Acknowledge signal, ACK, after the reception of each byte. The host device must generate an extra SCLx clock, which is associated with this Acknowledge bit.

23.2.2.4. Host Transmit

The next two bytes, sent by the host to the client, are data bytes that contain the location of the requested EEPROM data byte. The client must Acknowledge each of the data bytes.

23.2.2.5. Repeated Start

The client EEPROM has the required address information that is required to return the requested data byte to the host. However, the R/W status bit from the first device address byte specifies the host transmission and the client reception. The direction of the bus must be reversed for the client to send data to the host.

To perform this function without ending the message, the host sends a Repeated Start. The Repeated Start is followed with a device address byte containing the same device address as before, and with R/W = `1`, to indicate the client transmission and the host reception.

23.2.2.6. Client Reply

The client transmits the data byte by driving the SDAx line, while the host continues to originate clocks but releases its SDAx drive.

23.2.2.7. Host Acknowledge

During reads, a host must terminate data requests to the client by generating a NACK on the last byte of the message.

23.2.2.8. Stop Message

The host sends a Stop signal to terminate the message and returns the bus to an Idle state.

23.3. I2C System Overview

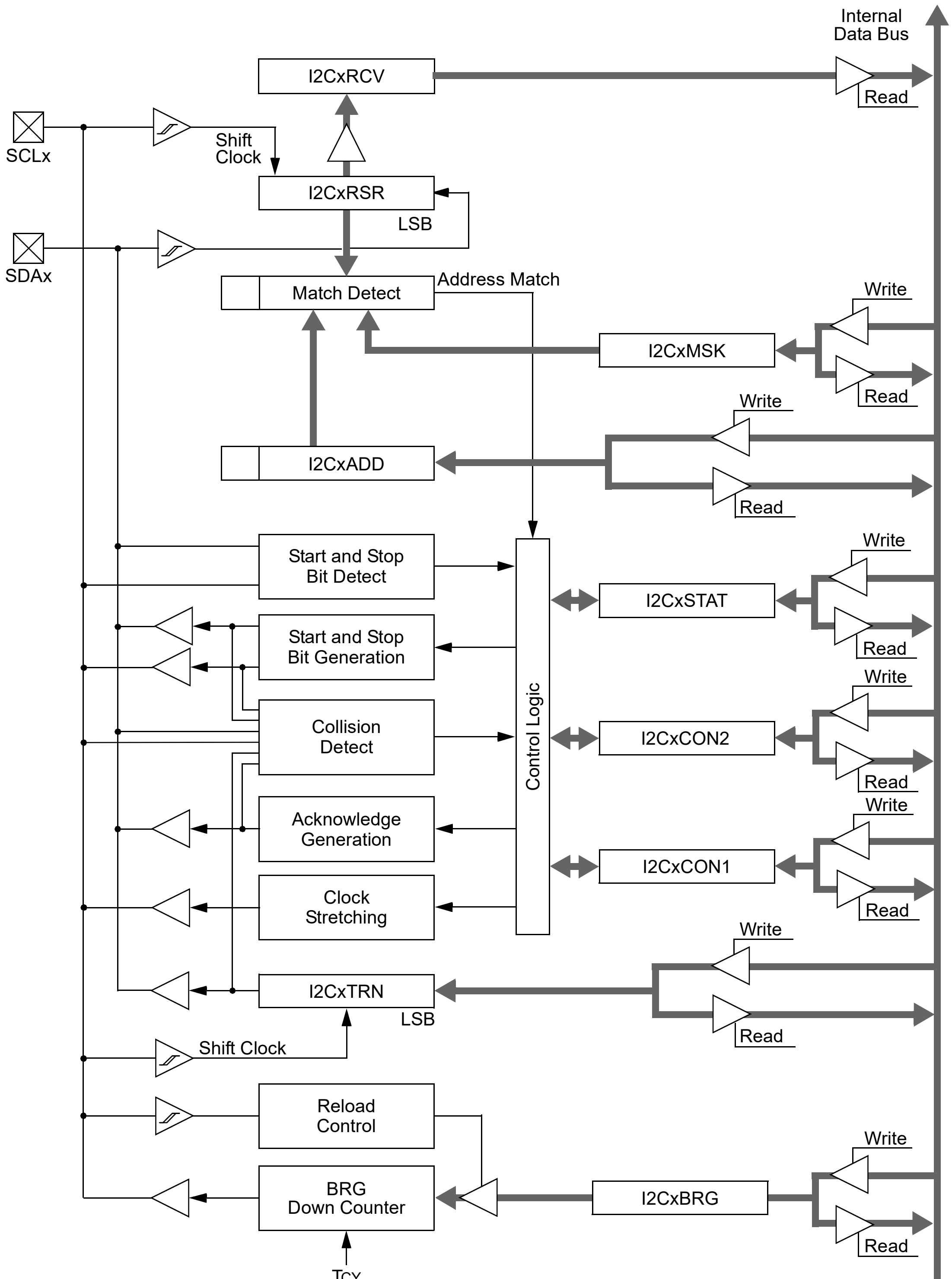
The I^2C module contains an independent I^2C host logic and an I^2C client logic, which generates interrupts based on their events. In the multi-host systems, the user software is simply partitioned into the host controller and the client controller.

When the I^2C host logic is active, the client logic also remains active, detecting the state of the bus and potentially receiving messages from itself in a single-host system or from the other hosts in a multi-host system. No messages are lost during the multi-host bus arbitration.

In a multi-host system, the bus collision conflicts with the other hosts in the system when detected, and the module provides a method to terminate and then restart the message.

The I^2C module contains a Baud Rate Generator (BRG). The I^2C BRG does not consume other timer resources in the device. Figure 23-4 illustrates the I^2C module block diagram.

Figure 23-4. I²C Block Diagram



23.4. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1880	I2C1CON1	31:24							SMBEN[1:0]	
		23:16		PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
		15:8	ON		SIDL	SCLREL	STRICT	A10M	DISSLW	
		7:0	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x1884	I2C1STAT1	31:24								
		23:16								
		15:8	ACKSTAT	TRSTAT	ACKTIM			BCL	GCSTAT	ADD10
		7:0	IWCOL	I2COV	D/A	P	S	R/W	RBF	TBF
0x1888	I2C1ADD	31:24								
		23:16								
		15:8							ADD[9:8]	
		7:0	ADD[7:0]							
0x188C	I2C1MSK	31:24								
		23:16								
		15:8							MSK[9:8]	
		7:0	MSK[7:0]							
0x1890	I2C1HBRG	31:24								
		23:16	I2CHBRG[23:16]							
		15:8	I2CHBRG[15:8]							
		7:0	I2CHBRG[7:0]							
0x1894	I2C1TRN	31:24								
		23:16								
		15:8								
		7:0	I2CTXDATA[7:0]							
0x1898	I2C1RCV	31:24								
		23:16								
		15:8								
		7:0	I2CRXDATA[7:0]							
0x189C	I2C1CON2	31:24	AMODE[1:0]		PECC[1:0]		BSCLTE	HBCTE	CBCTE	EPSZE
		23:16	ACKC[1:0]		HNACKIGN	EOPSC[1:0]		ND/$\overline{A}$	SMEN	BITE
		15:8	PSZ[15:8]							
		7:0	PSZ[7:0]							
0x18A0	I2C1LBRG	31:24								
		23:16	I2C1LBRG[23:16]							
		15:8	I2C1LBRG[15:8]							
		7:0	I2C1LBRG[7:0]							
0x18A4	I2C1INTC	31:24	I2CEIE		CBCLIE	HPCIE	HSCIE	HBCLIE	EOPIE	
		23:16	BSCLTIE	HBCTIE	CBCTIE	BITIE	FRMEIE	NACKIE		CRCIE
		15:8	BCLIE	HSBCLIE	HSTIE	CLTIE	HACKSIE	CADDRIE		
		7:0	RXIE	TXIE		CDRXIE	CDTXIE		HDTXIE	HDRXIE
0x18A8	I2C1STAT2	31:24	SSPND	CLTACT	HSTACT				HSBCL	EOP
		23:16	BSCLTO	HBCLTO	CBCLTO	BITO	FRME	NACKE		CRCE
		15:8	STOPE	STARTE	HSTIF	CLTIF	ERR			
		7:0					SCLCNT[3:0]			
0x18AC	I2C1PEC	31:24								
		23:16								
		15:8	CCRC[7:0]							
		7:0	RCRC[7:0]							
0x18B0	I2C1BTO	31:24	BTOCLK[1:0]							
		23:16	BTOPR[23:16]							
		15:8	BTOPR[15:8]							
		7:0	BTOPR[7:0]							
0x18B4	I2C1HBCTO	31:24								
		23:16	HBCTOTMR[23:16]							
		15:8	HBCTOTMR[15:8]							
		7:0	HBCTOTMR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x18B8	I2C1CBCTO	31:24								
		23:16	CBCTOTMR[23:16]							
		15:8	CBCTOTMR[15:8]							
		7:0	CBCTOTMR[7:0]							
0x18BC	I2C1BITO	31:24								
		23:16	BITOTMR[23:16]							
		15:8	BITOTMR[15:8]							
		7:0	BITOTMR[7:0]							
0x18C0	I2C1SDASUT	31:24	SDASUTEN							
		23:16								
		15:8	SDASUT[15:8]							
		7:0	SDASUT[7:0]							
0x18C4 ... 0x18CB	Reserved									
0x18CC	I2C1HDLYC	31:24	HIDLYCEN							
		23:16								
		15:8	HIDLYC[15:8]							
		7:0	HIDLYC[7:0]							
0x18D0	I2C2CON1	31:24							SMBEN[1:0]	
		23:16		PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
		15:8	ON		SIDL	SCLREL	STRICT	A10M	DISSLW	
		7:0	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x18D4	I2C2STAT1	31:24								
		23:16								
		15:8	ACKSTAT	TRSTAT	ACKTIM			BCL	GCSTAT	ADD10
		7:0	IWCOL	I2COV	D/A	P	S	R/W	RBF	TBF
0x18D8	I2C2ADD	31:24								
		23:16								
		15:8							ADD[9:8]	
		7:0	ADD[7:0]							
0x18DC	I2C2MSK	31:24								
		23:16								
		15:8							MSK[9:8]	
		7:0	MSK[7:0]							
0x18E0	I2C2HBRG	31:24								
		23:16	I2CHBRG[23:16]							
		15:8	I2CHBRG[15:8]							
		7:0	I2CHBRG[7:0]							
0x18E4	I2C2TRN	31:24								
		23:16								
		15:8								
		7:0	I2CTXDATA[7:0]							
0x18E8	I2C2RCV	31:24								
		23:16								
		15:8								
		7:0	I2CRXDATA[7:0]							
0x18EC	I2C2CON2	31:24	AMODE[1:0]		PECC[1:0]		BSCLTE	HBCTE	CBCTE	EPSZE
		23:16	ACKC[1:0]		HNACKIGN	EOPSC[1:0]		ND/$\overline{A}$	SMEN	BITE
		15:8	PSZ[15:8]							
		7:0	PSZ[7:0]							
0x18F0	I2C2LBRG	31:24								
		23:16	I2C2LBRG[23:16]							
		15:8	I2C2LBRG[15:8]							
		7:0	I2C2LBRG[7:0]							
0x18F4	I2C2INTC	31:24	I2CEIE		CBCLIE	HPCIE	HSCIE	HBCLIE	EOPIE	
		23:16	BSCLTIE	HBCTIE	CBCTIE	BITIE	FRMEIE	NACKIE		CRCIE
		15:8	BCLIE	HSBCLIE	HSTIE	CLTIE	HACKSIE	CADDRIE		
		7:0	RXIE	TXIE		CDRXIE	CDTXIE		HDTXIE	HDRXIE

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x18F8	I2C2STAT2	31:24	SSPND	CLTACT	HSTACT				HSBCL	EOP
		23:16	BSCLTO	HBCLTO	CBCLTO	BITO	FRME	NACKE		CRCE
		15:8	STOPE	STARTE	HSTIF	CLTIF	ERR			
		7:0					SCLCNT[3:0]			
0x18FC	I2C2PEC	31:24								
		23:16								
		15:8	CCRC[7:0]							
		7:0	RCRC[7:0]							
0x1900	I2C2BTO	31:24	BTOCLK[1:0]							
		23:16	BTOPR[23:16]							
		15:8	BTOPR[15:8]							
		7:0	BTOPR[7:0]							
0x1904	I2C2HBCTO	31:24								
		23:16	HBCTOTMR[23:16]							
		15:8	HBCTOTMR[15:8]							
		7:0	HBCTOTMR[7:0]							
0x1908	I2C2CBCTO	31:24								
		23:16	CBCTOTMR[23:16]							
		15:8	CBCTOTMR[15:8]							
		7:0	CBCTOTMR[7:0]							
0x190C	I2C2BITO	31:24								
		23:16	BITOTMR[23:16]							
		15:8	BITOTMR[15:8]							
		7:0	BITOTMR[7:0]							
0x1910	I2C2SDASUT	31:24	SDASUTEN							
		23:16								
		15:8	SDASUT[15:8]							
		7:0	SDASUT[7:0]							
0x1914 ... 0x191B	Reserved									
0x191C	I2C2HDLYC	31:24	HIDLYCEN							
		23:16								
		15:8	HIDLYC[15:8]							
		7:0	HIDLYC[7:0]							

23.4.1. I²Cx Control Register

Name: I2CxCON1
Offset: 0x1880, 0x18D0

Notes:
1. Automatically cleared to '0' at the beginning of client transmission; automatically cleared to '0' at the end of client reception.
2. Automatically cleared to '0' at the beginning of client transmission.
3. This bit must be set before any I²C operations can be performed. This bit should be set in a separate instruction from any of the other enable bits.
4. 16 peripheral clock cycles should be waited before performing any operation.

Legend: HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
							SMBEN[1:0]	
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
		PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	SCLREL	STRICT	A10M	DISSLW	
Access	R/W		R/W	R/W/HC	R/W	R/W	R/W	
Reset	0		0	1	0	0	0	

Bit	7	6	5	4	3	2	1	0
	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
Access	R/W	R/W	R/W	R/W/HC/HS	R/W/HC/HS	R/W/HC	R/W/HC	R/W/HC
Reset	0	0	0	0	0	0	0	0

Bits 25:24 – SMBEN[1:0] SMBus Input Levels Enable bit

Value	Description
11	Reserved
10	Enable SMBus 3.0 input threshold voltage specification.
01	Enable SMBus 2.0 input threshold voltage specification.
00	Enable I2C input threshold voltage specification (disable SMBus specific input).

Bit 22 – PCIE Stop Condition Interrupt Enable bit (Client mode only)

Value	Description
1	Enables interrupt on detection of Stop condition.
0	Stop detection interrupts are disabled.

Bit 21 – SCIE Start Condition Interrupt Enable bit (Client mode only)

Value	Description
1	Enables interrupt on detection of Start or Restart conditions.
0	Start detection interrupts are disabled.

Bit 20 – BOEN Buffer Overwrite Enable bit (Client mode only)

Value	Description
1	I2CxRCV is updated, and an ACK is generated for a received address/data byte, ignoring the state of the I2COV bit only if the RBF bit = 0.
0	I2CxRCV is only updated when I2COV is clear.

Bit 19 – SDAHT SDAx Hold Time Selection bit

Value	Description
1	Minimum of 300 ns hold time on SDAx after the falling edge of SCLx.
0	Minimum of 100 ns hold time on SDAx after the falling edge of SCLx.

Bit 18 – SBCDE Mode Bus Collision Detect Enable bit (Client mode only)

If, on the rising edge of SCLx, SDAx is sampled low when the module is outputting a high state, the BCL bit is set and the bus goes Idle. This Detection mode is only valid during data and ACK transmit sequences.

Value	Description
1	Enables client bus collision interrupts.
0	Client bus collision interrupts are disabled.

Bit 17 – AHEN Client Address Hold Enable bit

Value	Description
1	Following the eighth falling edge of SCLx for a matching received address byte, the SCLREL bit (I2CxCON1[12]) will be cleared and the SCLx will be held low.
0	Address holding is disabled.

Bit 16 – DHEN Client Data Hold Enable bit

Value	Description
1	Following the eighth falling edge of SCLx for a received data byte, client hardware clears the SCLREL bit (I2CxCON1[12]) and SCLx is held low.
0	Data holding is disabled.

Bit 15 – ON I^{2}Cx Enable bit (writable from software only)**(3,4)**

Value	Description
1	Enables the I2Cx module and configures the SDAx and SCLx pins as serial port pins.
0	Disables the I2Cx module; all I^{2}C pins are controlled by port functions.

Bit 13 – SIDL I^{2}Cx Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when the device enters Idle mode.
0	Continues module operation in Idle mode.

Bit 12 – SCLREL SCLx Release Control bit (I^{2}C Client mode only)**(1)**

<u>If STREN = 1:</u>**(2)**

User software may write '0' to initiate a clock stretch and write '1' to release the clock. Hardware clears at the beginning of every client data byte transmission. Hardware clears at the end of every client address byte reception. Hardware clears at the end of every client data byte reception.

<u>If STREN = 0:</u>

User software may only write '1' to release the clock. Hardware clears at the beginning of every client data byte transmission. Hardware clears at the end of every client address byte reception.

Value	Description
1	Releases the SCLx clock.

Value	Description
0	Holds the SCLx clock low (clock stretch).

Bit 11 – STRICT I^2Cx Strict Reserved Address Rule Enable bit

Value	Description
1	Strict reserved addressing is enforced for reserved addresses. (In Client mode) – The device doesn't respond to reserved address space and addresses falling in that category are NACKed. (In Host mode) – The device is allowed to generate addresses within the reserved address space.
0	Reserved addressing would be Acknowledged. (In Client mode) – The device will respond to an address falling in the reserved address space. When there is a match with any of the reserved addresses, the device will generate an ACK. (In Host mode) – Reserved.

Bit 10 – A10M 10-Bit Client Address Flag bit

Value	Description
1	I2CxADD is a 10-bit client address.
0	I2CxADD is a 7-bit client address.

Bit 9 – DISSLW Slew Rate Control Disable bit

Value	Description
1	Slew rate control is disabled for Standard Speed mode (100 kHz, also disabled for 1 MHz mode).
0	Slew rate control is enabled for High-Speed mode (400 kHz).

Bit 7 – GCEN General Call Enable bit (in I^2C Client mode only)

Value	Description
1	Enables interrupt when a general call address is received in I2CxRSR; module is enabled for reception.
0	General call address is disabled.

Bit 6 – STREN SCLx Clock Stretch Enable bit

In I^2C Client mode only; used in conjunction with the SCLREL bit.

Value	Description
1	Enables clock stretching.
0	Disables clock stretching.

Bit 5 – ACKDT Acknowledge Data bit

In I^2C Host mode during Host Receive mode, the value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
In I^2C Client mode when AHEN = `1` or DHEN = `1`, the value that the client will transmit when it initiates an Acknowledge sequence at the end of an address or data reception.

Value	Description
1	NACK is sent.
0	ACK is sent.

Bit 4 – ACKEN Acknowledge Sequence Enable bit

In I^2C Host mode only; applicable during Host Receive mode.

Value	Description
1	Initiates the Acknowledge sequence on SDAx and SCLx pins and transmits the ACKDT data bit.
0	Acknowledge sequence is Idle

Bit 3 – RCEN Receive Enable bit (in I^{2}C Host mode only)

Value	Description
1	Enables Receive mode for I^{2}C; automatically cleared by hardware at the end of an 8-bit receive data byte.
0	Receive sequence is not in progress.

Bit 2 – PEN Stop Condition Enable bit (in I^{2}C Host mode only)

Value	Description
1	Initiates Stop condition on SDAx and SCLx pins.
0	Stop condition is Idle.

Bit 1 – RSEN Restart Condition Enable bit (in I^{2}C Host mode only)

Value	Description
1	Initiates Restart condition on SDAx and SCLx pins.
0	Restart condition is Idle.

Bit 0 – SEN Start Condition Enable bit (in I^{2}C Host mode only)

Value	Description
1	Initiates Start condition on SDAx and SCLx pins.
0	Start condition is Idle.

23.4.2. I^{2}Cx Control Register

Name: I2CxCON2
Offset: 0x189C, 0x18EC

Notes:

1. Automatically cleared to '0' at the beginning of client transmission; automatically cleared to '0' at the end of client reception.
2. Automatically cleared to '0' at the beginning of client transmission.
3. When EPSZE is enabled, Smart mode (SMEN=1), clock stretching (STREN=1), and EOP function (EOPSC= "10" or "01") should be enabled.
4. In Host mode, when ACKC != 00, hardware will automatically set the ACKEN (I2CxCON1[4]) bit.
5. In Host mode, this bit will be used by PSZ (I2CxCON2[15:0]) to count the data bytes.
6. In Client mode, this bit is used for TX/RX interrupt generation.
   If '1', an interrupt is generated only for data bytes. If '0', an interrupt is generated for both address and data bytes.
7. In Client mode, it should set the ND/A to '1' before enabling transfers through DMA for data transfer.
8. The packet size should exclude address byte(s). It should not be changed on the fly and should be changed when the bus is in the Idle state.
9. In Host mode, the EOPSC and ND/A bit control the PSZ to decrement.
10. For the host, the software has to clear the CRC calculator by setting I2CxCON2.PECC to "11" for a new data frame transaction.
11. To use the Auto-Append mode, the PECC needs to be set to "01" at least one data byte earlier than the CRC byte of data.

Legend: HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
	AMODE[1:0]		PECC[1:0]		BSCLTE	HBCTE	CBCTE	EPSZE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ACKC[1:0]		HNACKIGN	EOPSC[1:0]		ND/$\overline{A}$	SMEN	BITE
Access	R/W	R/W	R/W	R/W/HC	R/W/HC	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSZ[15:8]							
Access	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSZ[7:0]							
Access	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC
Reset	0	0	0	0	0	0	0	1

Bits 31:30 – AMODE[1:0] Address Mode bit

Value	Description
11	Reserved
10	The client responds to the range of addresses between and including I2CxADD and I2CxMSK. I2CxMSK is the upper limit.
01	The client responds to the two unique addresses in I2CxADD and I2CxMSK.
00	I2CxMSK is used as a mask for the I2CxADD register.

Bits 29:28 – PECC[1:0] PEC Control bit(10,11)

Value	Description
11	PEC Reset
10	PEC append is disabled. The CRC-8 calculator will be active. On a read request (receive), the calculated CRC-8 is copied into CCRC (I2CxPEC[15:8]) at EOP. On a write request (transmit), the calculated CRC-8 is copied into CCRC (I2CxPEC[15:8]) at EOP.
01	The calculated CRC-8 is appended at the end of a packet. On a read request (receive), the calculated CRC-8 is copied into CCRC (I2CxPEC[15:8]) and the received CRC will be copied into RCRC (I2CxPEC[7:0]) at EOP. On a write request (transmit), the calculated CRC-8 will be automatically appended and also copied into CCRC (I2CxPEC[15:8]) at the end of the data transmission. PEC will be reset after appending.
00	PEC disabled

Bit 27 – BSCLTE Bus SCL Time-out Enable bit

Value	Description
1	SCL low time-out enabled.
0	SCL low time-out disabled.

Bit 26 – HBCTE Host Bus SCL Cumulative (Extended Time) Low Time-Out Enable Bit

Value	Description
1	SCL Cumulative low extended time-out enable.
0	SCL Cumulative low extended time-out disable.

Bit 25 – CBCTE Client Bus SCL Cumulative (Extended time) Low Time-Out Enable bit

Value	Description
1	SCL Cumulative low extended time-out enable.
0	SCL Cumulative low extended time-out disable.

Bit 24 – EPSZE Extended Packet Size Enable bit (valid for Client Receive mode only)(3)

Value	Description
1	Extended packet size enabled after EOP = 1.
0	Extended packet size disable.

Bits 23:22 – ACKC[1:0] ACK Control bits(4)

Value	Description
11	Host: ACK all the bytes except the CRC byte; for the CRC byte, a NACK will be sent. Client: ACK all the bytes, and for the last byte, which is the CRC byte, ACK or NACK based on the CRC result. • If the CRC is 0, an ACK is sent. • If the CRC is non-zero, a NACK is sent (to be used only on PECC[1:0] = “01” Auto-Append mode).
10	ACK all the bytes, and the last byte will be NACKed (to be used only for the last packet).
01	ACK all bytes including the end of the packet (to be used for extended packets).
00	ACK/NACK based on ACKDT and BOEN (client). ACK/NACK based on ACKEN and ACKDT (host).

Bit 21 – HNACKIGN Host NACK Response Ignore Control bit

Value	Description
1	Host treats all NACK responses as ACK.
0	Normal operation; host treats NACK responses as NACK only.

Bits 20:19 – EOPSC[1:0] End of Packet Status Control bits

Value	Description
11	Reserved
10	I2CxSTAT2.EOP will be set after data bytes and PEC.
01	I2CxSTAT2.EOP will be set after data bytes.
00	End of packet function is disabled.

Bit 18 – ND/$\overline{\text{A}}$ Next Data/Address Byte Transmission bit(5,6,7)

Value	Description
1	Next transmission is a data byte transmission.
0	Next transmission is an address byte transmission.

Bit 17 – SMEN Smart Mode Enable bit

Value	Description
1	Smart mode is enabled.
0	Smart mode is disabled.

Bit 16 – BITE Bus Idle Time-Out Enable bit

Value	Description
1	Bus idle time-out enabled.
0	Bus idle time-out disabled.

Bits 15:0 – PSZ[15:0] Packet Size bits(8,9)

Sets the size of the packet to transfer/receive; the valid range is from 0 to 65,535 bytes. Use I2CxCONC.PSZ=0 for the SMBus Quick Command protocol.

23.4.3. I^2Cx Status Register

Name: I2CxSTAT1
Offset: 0x1884, 0x18D4

Legend: C = Clearable bit; HS = Hardware Settable bit; HSC = Hardware Settable/Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ACKSTAT	TRSTAT	ACKTIM			BCL	GCSTAT	ADD10
Access	R/HSC	R/HSC	R/HSC			R/C/HSC	R/HSC	R/HSC
Reset	0	0	0			0	0	0

Bit	7	6	5	4	3	2	1	0
	IWCOL	I2COV	D/A	P	S	R/W	RBF	TBF
Access	R/C/HS	R/C/HS	R/HSC	R/HSC	R/HSC	R/HSC	R/HSC	R/HSC
Reset	0	0	0	0	0	0	0	0

Bit 15 – ACKSTAT Acknowledge Status bit (updated in all Host and Client modes)

Value	Description
1	Acknowledge was not received from client.
0	Acknowledge was received from client.

Bit 14 – TRSTAT Transmit Status bit (when operating as I^2C host; applicable to host transmit operation)

Value	Description
1	Host transmit is in progress (eight bits + ACK).
0	Host transmit is not in progress.

Bit 13 – ACKTIM Acknowledge Time Status bit (valid in I^2C Client mode only)

Value	Description
1	Indicates the I^2C bus is in an Acknowledge sequence, set on the eighth falling edge of the SCLx clock.
0	Not an Acknowledge sequence, cleared on the ninth rising edge of the SCLx clock.

Bit 10 – BCL Bus Collision Detect bit
(Host/Client mode; cleared when I^2C module is disabled, I2CEN = 0)

Value	Description
1	A bus collision has been detected during a host or client transmit operation.
0	No bus collision has been detected.

Bit 9 – GCSTAT General Call Status bit (cleared after Stop detection)

Value	Description
1	General call address was received.

Value	Description
0	General call address was not received.

Bit 8 – ADD10 10-Bit Address Status bit (cleared after Stop detection)

Value	Description
1	10-bit address was matched.
0	10-bit address was not matched.

Bit 7 – IWCOL I2Cx Write Collision Detect bit

Value	Description
1	An attempt to write to the I2CxTRN register failed because the I^2C module is busy; it must be cleared in software.
0	No collision.

Bit 6 – I2COV I2Cx Receive Overflow Flag bit

Value	Description
1	A byte was received while the I2CxRCV register was still holding the previous byte.
0	No overflow.

Bit 5 – D/$\overline{\text{A}}$ Data/$\overline{\text{Address}}$ bit (when operating as I^2C Client)

Value	Description
1	Indicates that the last byte received was data.
0	Indicates that the last byte received or transmitted was an address.

Bit 4 – P Stop bit
Updated when Start, Reset or Stop is detected; cleared when the I^2C module is disabled, I2CEN = 0.

Value	Description
1	Indicates that a Stop bit has been detected last.
0	Stop bit was not detected last.

Bit 3 – S I2Cx Start bit
Updated when Start, Reset or Stop is detected; cleared when the I^2C module is disabled, I2CEN = 0.

Value	Description
1	Indicates that a Start (or Repeated Start) bit has been detected last.
0	Start bit was not detected last.

Bit 2 – R/$\overline{\text{W}}$ Read/$\overline{\text{Write}}$ Information bit (when operating as I^2C Client)

Value	Description
1	Read: Indicates that the data transfer is output from the client.
0	Write: Indicates that the data transfer is input to the client.

Bit 1 – RBF Receive Buffer Full Status bit

Value	Description
1	Receive is complete; I2CxRCV is full.
0	Receive is not complete; I2CxRCV is empty.

Bit 0 – TBF Transmit Buffer Full Status bit

Value	Description
1	Transmit is in progress; I2CxTRN is full (eight bits of data).
0	Transmit is complete; I2CxTRN is empty.

23.4.4. I^{2}Cx Client Address Register

Name: I2CxADD
Offset: 0x1888, 0x18D8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
							ADD[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	ADD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 9:0 – ADD[9:0] Client Device Address bits

23.4.5. I²Cx Client Mode Address Mask Register

Name: I2CxMSK
Offset: 0x188C, 0x18DC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
							MSK[9:8]	
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	MSK[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 9:0 – MSK[9:0] I²Cx Mask for Address Bit x Select bits

Value	Description
1	Enables masking for bit x of the incoming message address; a bit match is not required in this position.
0	Disables masking for bit x; a bit match is required in this position.

23.4.6. I²Cx SCL High Baud Rate Generator Register

Name: I2CxHBRG
Offset: 0x1890, 0x18E0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	I2CHBRG[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	I2CHBRG[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	I2CHBRG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – I2CHBRG[23:0] I²Cx SCL High Time Baud Rate Generator Value bits

23.4.7. I²Cx Transmit Data Register

Name: I2CxTRN
Offset: 0x1894, 0x18E4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	I2CTXDATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – I2CTXDATA[7:0] I²C Transmit Data Buffer bits

23.4.8. I^2Cx Receive Data Register

Name: I2CxRCV
Offset: 0x1898, 0x18E8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	I2CRXDATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – I2CRXDATA[7:0] I^2Cx Receive Data bits

23.4.9. I²Cx SCL Low Baud Rate Generator Register

Name: I2CxLBRG
Offset: 0x18A0, 0x18F0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	I2CxLBRG[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	I2CxLBRG[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	I2CxLBRG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – I2CxLBRG[23:0] I²Cx SCL Low time Baud Rate Generator Value bits

23.4.10. I²Cx Interrupt Control Register

Name: I2CxINTC
Offset: 0x18A4, 0x18F4

Bit	31	30	29	28	27	26	25	24
	I2CEIE		CBCLIE	HPCIE	HSCIE	HBCLIE	EOPIE	
Access	R/W		R/W	R/W	R/W	R/W	R/W	
Reset	0		0	0	0	0	0	

Bit	23	22	21	20	19	18	17	16
	BSCLTIE	HBCTIE	CBCTIE	BITIE	FRMEIE	NACKIE		CRCIE
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		0

Bit	15	14	13	12	11	10	9	8
	BCLIE	HSBCLIE	HSTIE	CLTIE	HACKSIE	CADDRIE		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
	RXIE	TXIE		CDRXIE	CDTXIE		HDTXIE	HDRXIE
Access	R/W	R/W		R/W	R/W		R/W	R/W
Reset	0	0		0	0		0	0

Bit 31 – I2CEIE Error Interrupt Enable bit

Value	Description
1	Assert I2CIF if I2CEIF asserts. Merge error interrupts onto I2CIF. The ERR bit is not affected by setting this bit.
0	Disabled

Bit 29 – CBCLIE Client Bus Collision Message Event Interrupt Enable bit (I²C Client mode only)

Value	Description
1	Enable CLTIF interrupt on bus collision detect.
0	Bus collision interrupt is disabled.

Bit 28 – HPCIE Host STOP Condition Interrupt Enable bit

Value	Description
1	Assert HSTIF when I2CxSTAT.P is set.
0	Host STOP interrupt is disabled.

Bit 27 – HSCIE Host START Condition Interrupt Enable bit

Value	Description
1	Assert HSTIF when I2CxSTAT.S is set.
0	Host START interrupt disabled.

Bit 26 – HBCLIE Host Bus Collision Message Event Interrupt Enable bit (I²C Host mode only)

Value	Description
1	Enable HSTIF interrupt on bus collision detect.
0	Bus collision interrupt is disabled.

Bit 25 – EOPIE End of Packet Interrupt Enable bit

Value	Description
1	End-of-packet interrupt is enabled.
0	End-of-packet interrupt is disabled.

Bit 23 – BSCLTIE Bus SCL Low Time-Out Enable bit

Value	Description
1	Assert I2CEIF when I2CxSTAT2.BSCLTO is set.
0	Bus SCL time-out not enabled.

Bit 22 – HBCTIE Host Bus Cumulative Time-Out Interrupt Enable bit

Value	Description
1	Assert I2CEIF when I2CxSTAT2.HBCLTO is set.
0	Host bus cumulative time-out not enabled.

Bit 21 – CBCTIE Client Bus Cumulative Time-Out Interrupt Enable bit

Value	Description
1	Assert I2CEIF when I2CxSTAT2.CBCLTO is set.
0	Client bus cumulative time-out not enabled.

Bit 20 – BITIE Bus Idle Time-Out Enable Interrupt bit

Value	Description
1	Assert I2CIF when I2CxSTAT2.BIT is set.
0	Bus free interrupt is disabled.

Bit 19 – FRMEIE Framing Error Detect Interrupt Enable bit

Value	Description
1	Enable the I2CEIF interrupt if FRME is set due to a framing error. A framing error occurs in Client mode if a STOP or START is received during the byte transfer time or $\overline{\text{ACK}}$ transfer time.
0	Frame error interrupt is disabled.

Bit 18 – NACKIE NACK Detect Interrupt Enable bit

Value	Description
1	Enable I2CEIF interrupt on NACK detect as an error.
0	NACK interrupt is disabled.

Bit 16 – CRCIE CRC Error Interrupt Enable bit

Value	Description
1	Enable I2CEIF interrupt on CRC error.
0	CRC interrupts are disabled.

Bit 15 – BCLIE Bus Collision Detect Interrupt Enable bit

Value	Description
1	Enable I2CEIF interrupt on bus collision.
0	Bus collision interrupts are disabled.

Bit 14 – HSBCLIE Host Bus Collision Detect Interrupt Enable bit

Value	Description
1	Enable interrupt during the STOP sequence after receiving a bus collision.

Value	Description
0	An interrupt during the STOP sequence after receiving a bus collision is disabled.

Bit 13 – HSTIE Host Interrupt Enable bit

Value	Description
1	Assert I2CIF when I2CxSTAT2.HSTIFis set.
0	Host interrupt is disabled.

Bit 12 – CLTIE Client Interrupt Enable bit

Value	Description
1	Assert I2CIF when I2CxSTAT2.CLIIF is set.
0	Client interrupt is disabled.

Bit 11 – HACKSIE Host ACK Sequence Interrupt Enable bit

Value	Description
1	Enable HSTIF interrupt on ACK sequence.
0	ACK sequence interrupt is disabled.

Bit 10 – CADDRIE Client Address Transaction Interrupt Enable bit

Value	Description
1	Enable CLTIF interrupt on address detect.
0	Address detect interrupt is disabled.

Bit 7 – RXIE Receive Interrupt Enable bit

Value	Description
1	Enable the I2CRXIF interrupt when I2CxSTAT.RBF is set.
0	Disable the I2CRXIF interrupt.

Bit 6 – TXIE Transmit Interrupt Enable bit

Value	Description
1	Enable the I2CTXIF interrupt when I2CxSTAT.TBF is cleared.
0	Disable the I2CTXIF interrupt.

Bit 4 – CDRXIE Client Data Receive Buffer Full Interrupt Enable bit

Value	Description
1	Include the I2CxSTAT.RBF=1 for the CLTIF interrupt.
0	Exclude the I2CxSTAT.RBF=1 for the CLTIF interrupt.

Bit 3 – CDTXIE Host Data Transmit Buffer Empty Interrupt Enable bit

Value	Description
1	Include the host I2CxSTAT.TBF=0 for the HSTIF interrupt.
0	Exclude the I2CxSTAT.TBF=0 for the HSTIF interrupt.

Bit 1 – HDTXIE Host Data Transmit Buffer Empty Interrupt Enable bit

Value	Description
1	Include the host I2CxSTAT.TBF=0 for the HSTIF interrupt.
0	Exclude the I2CxSTAT.TBF=0 for the HSTIF interrupt.

Bit 0 – HDRXIE Host Data Receive Buffer Full Interrupt Enable bit

Value	Description
1	Include the I2CxSTAT.RBF=1 for the HSTIF interrupt.
0	Exclude the I2CxSTAT.RBF=1 for the HSTIF interrupt.

23.4.11. I²Cx Status Register

Name: I2CxSTAT2
Offset: 0x18A8, 0x18F8

Note:

1. Cleared by any Stop detected on the bus. Cleared by I2CxSTAT2.BSCLTO or I2CxSTAT.BCL conditions. Cleared by any address match failure in 10-bit Addressing mode. Cleared by any address match failure following a Restart detect. Cleared when NACK is either received or transmitted to the client. Cleared when the client goes to IDLE after ACK/NACK CRC byte.
2. Cleared when BCL is set. Cleared when Stop is sent. Cleared for I2CxSTAT2.BSCLTO condition, after the host successfully sends a STOP condition.
3. It is the user's responsibility to make sure the EOP = "0" to start new packet transmission.
4. ERRF is a combined Error Status bit of I2CxSTAT2.BSCLTO + I2CxSTAT2.CBCTO + I2CxSTAT2.HBCTO + I2CxSTAT2.FRAME + I2CxSTAT2.CRC + I2CxSTAT1.BCL + I2CxSTAT2.NACKE.

Legend: C = Clearable bit; HS = Hardware Settable bit; HSC = Hardware Settable/Clearable bit

Bit	31	30	29	28	27	26	25	24
	SSPND	CLTACT	HSTACT				HSBCL	EOP
Access	R/HS/HC	R/HS/HC	R/HS/HC				R/C/HS	R/C/HS
Reset	0	0	0				0	0

Bit	23	22	21	20	19	18	17	16
	BSCLTO	HBCLTO	CBCLTO	BITO	FRME	NACKE		CRCE
Access	R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS		R/C/HS
Reset	0	0	0	0	0	0		0

Bit	15	14	13	12	11	10	9	8
	STOPE	STARTE	HSTIF	CLTIF	ERR			
Access	R/C/HS	R/HSC	R/HSC	R/C/HSC	R/C/HSC			
Reset	0	0	0	0	0			

Bit	7	6	5	4	3	2	1	0
					SCLCNT[3:0]			
Access					R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset					0	0	0	0

Bit 31 – SSPND I²C Suspended bit

During a host transaction, I²C bus activity is suspended to allow the user to service the I²C interrupt. During a client transaction, the SCL line is pulled low to clock stretch until the user services the I²C interrupt.

Value	Description
1	The I²C is in a suspended state.
0	The I²C is not in a suspended state.

Bit 30 – CLTACT Client State Machine Active Status bit

Value	Description
1	Set after the eighth falling SCL edge of a received matching client address.
0	Client is idle[(1)].

Bit 29 – HSTACT Host State Machine Active Status bit

Value	Description
1	Host mode state machine is active; set when the host state machine asserts a Start on the bus. Under these conditions, the host function is enabled, and a START/RESTART condition is sent on the I^2C bus, followed by data from the transmit buffer.
0	Host is Idle(2).

Bit 25 – HSBCL Host Stop Bus Collision Detect Flag bit

Value	Description
1	Host stop bus collision is detected.
0	Host stop bus collision is not detected.

Bit 24 – EOP End-of-Packet Flag bit(3)

Value	Description
1	End-of-packet condition detected.
0	End-of-packet condition not detected.

Bit 23 – BSCLTO Bus SCL Time-Out Flag bit

Value	Description
1	SCL bus time-out occurred.
0	SCL bus time-out has not occurred.

Bit 22 – HBCLTO Host Bus Cumulative Extended Time-Out Flag bit

Value	Description
1	Host bus cumulative extended time-out occurred.
0	Host bus cumulative time-out has not occurred.

Bit 21 – CBCLTO Client Bus Cumulative Extended Time-out Flag bit

Value	Description
1	Client bus cumulative extended time-out occurred.
0	Client bus cumulative extended time-out has not occurred.

Bit 20 – BITO Bus Idle Time-Out Flag bit

Value	Description
1	Bus idle time-out occurred.
0	Bus idle time-out has not occurred.

Bit 19 – FRME Frame Error Detect Flag bit

Value	Description
1	Frame error detected in Client mode; STOP or START is received during the data byte transfer or data ACK transfer time.
0	Frame error has not been detected in Client mode.

Bit 18 – NACKE NACK Detect Error Flag bit

Value	Description
1	NACK detected as an error.
0	NACK has not been detected as an error.

Bit 16 – CRCE CRC Error Flag bit

Value	Description
1	CRC error occurred.

Value	Description
0	CRC error has not occurred.

Bit 15 – STOPE Stop Condition Detect Event Flag bit

Value	Description
1	Stop condition detected.
0	Stop condition has not been detected.

Bit 14 – STARTE Start Condition Detect Event Flag bit

Value	Description
1	Start condition event detected.
0	Start condition event not detected.

Bit 13 – HSTIF Host Detect Interrupt Flag bit

Value	Description
1	Host interrupt detected.
0	Host interrupt not detected.

Bit 12 – CLTIF Client Detect Interrupt Flag bit

Value	Description
1	Client interrupt detected.
0	Client interrupt not detected.

Bit 11 – ERR Error Detect Flag bit[(4)]

Value	Description
1	Error is detected.
0	Error is not detected.

Bits 3:0 – SCLCNT[3:0] SCL Count bits
Number of SCL clocks to be stored for a Frame Error.

23.4.12. I²Cx PEC Register

Name: I2CxPEC
Offset: 0x18AC, 0x18FC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CCRC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RCRC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CCRC[7:0] Calculated Packet Error Check CRC Value bits
Provides a readback of the CRC-8 calculator output for the current transaction.

Bits 7:0 – RCRC[7:0] Received Packet Error Check CRC Value bits
Provides a readback of the CRC-8 received CRC for the current transaction.

23.4.13. I²Cx Bus Time-out Selection Register

Name: I2CxBTO
Offset: 0x18B0, 0x1900

Bit	31	30	29	28	27	26	25	24
	BTOCLK[1:0]							
Access	R/W	R/W						
Reset	0	0						

Bit	23	22	21	20	19	18	17	16
	BTOPR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BTOPR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BTOPR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:30 – BTOCLK[1:0]

Bits 23:0 – BTOPR[23:0]

23.4.14. I²Cx Host Bus Cumulative Time-Out Register

Name: I2CxHBCTO
Offset: 0x18B4, 0x1904

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	HBCTOTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	HBCTOTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HBCTOTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – HBCTOTMR[23:0] I²C Host Cumulative Bus Time-Out Timer bits
Provides the Reset value for the HCBCTO timer. The timer resets on detection of a START, ACK, or STOP and continues to run when SCL low is extended and HSTACT=1. The timer is disabled when HBCTE=0.

23.4.15. I²Cx Client Bus Cumulative Time-out Register

Name: I2CxCBCTO
Offset: 0x18B8, 0x1908

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
	CBCTOTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	CBCTOTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	CBCTOTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – CBCTOTMR[23:0] I²C Client Cumulative Bus Time-Out Timer bits
Provides the Reset value for the CBCTO timer. The timer resets on detection of a START or STOP and continues to run when SCL is low. The timer is disabled when CBCTE=0.

23.4.16. I²Cx Bus Idle/Free Time-Out Register

Name: I2CxBITO
Offset: 0x18BC, 0x190C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	BITOTMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BITOTMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BITOTMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:0 – BITOTMR[23:0] I²C Bus Idle (inactive/Free) Time-Out Timer bit

23.4.17. I²Cx Bus SDA Set-Up Time Register

Name: I2CxSDASUT
Offset: 0x18C0, 0x1910

Notes:
1. Timer runs with the peripheral clock.
2. This provides a delay between writing to the transmit buffer and releasing the SCL by hardware; it is used for the release of SCL by hardware.
3. In Smart mode, the SDA setup timer is used by hardware to release SCL.

Bit	31	30	29	28	27	26	25	24
	SDASUTEN							
Access	R/W							
Reset	0							
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Access								
Reset								
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	SDASUT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	SDASUT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – SDASUTEN I²C Bus SDA-to-SCL Set-Up Time Enable bit

Bits 15:0 – SDASUT[15:0] I²C Bus SDA-to-SCL Set-Up Time Timer bits

23.4.18. I²C Host Input Delay Compensation Register

Name: I2CxHDLYC
Offset: 0x18CC, 0x191C

Note:
1. Default value: assumed UPB clock is 50 MHz, and path delay is 360 ns with 2.3V and 1.1K pull-up resistance (worst case).

Bit	31	30	29	28	27	26	25	24
	HIDLYCEN							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	HIDLYC[15:8]							
Access								
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	HIDLYC[7:0]							
Access								
Reset	0	0	0	0	0	0	0	0

Bit 31 – HIDLYCEN I2C Host Input Delay Compensation Enable bit

Value	Description
1	Programmable hardware host input delay compensation enabled (default).
0	Fixed hardware host input delay compensation used.

Bits 15:0 – HIDLYC[15:0] Host Input Delay Compensation Value bits
These bits are used to compensate for input path delays for the host mode sampling point.

Value	Description
1111	HIDLYC Value x I2C UPB CLK TIME
0010	18 x I2C UPB CLK TIME (default)(1)
0001	1 x I2C UPB CLK TIME
0000	0 x I2C UPB CLK TIME

23.5. Operation

23.5.1. Enabling I²C Operation

The I²C module is enabled by setting the I2CEN bit (I2CxCON1[15]. The I²C module fully implements all host and client functions. When the module is enabled, the host and client functions are active simultaneously and will respond according to the user software or bus events.

When initially enabled, the module will release the SDAx and SCLx pins, putting the bus into an Idle state. The host functions will remain in an Idle state unless the user software sets the SEN control bit and the data are loaded into the I2CxTRN register. These two actions initiate a host event.

When the host logic is active, the client logic also remains active. Therefore, the client functions will begin to monitor the bus. If the client logic detects a Start event and a valid address on the bus, the client logic will begin a client transaction.

23.5.1.1. I^{2}C I/O Pins

Two pins are used for the bus operation. These are the SCLx pin, which is the clock, and the SDAx pin, which is the data. When the module is enabled, assuming no other module with higher priority has control, the module will assume control of the SDAx and SCLx pins. The user software need not be concerned with the state of the port I/O of the pins as the module overrides the port state and direction. At initialization, the pins are tri-stated (released).

Note: SCLx and SDAx must be configured as digital.

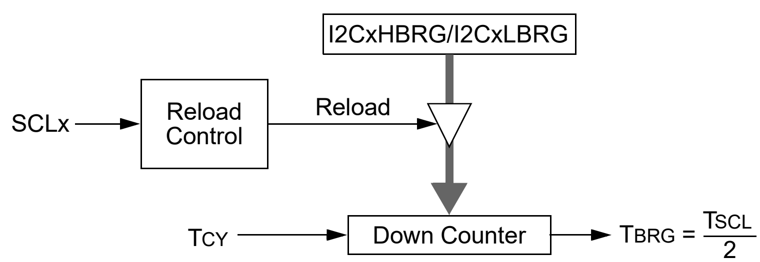
23.5.1.2. Setting Baud Rate When Operating as a Bus Host

When operating as an I^{2}C host, the module must generate the system SCLx clock. Generally, the I^{2}C system clocks are specified to be either 100 kHz, 400 kHz or 1 MHz. The system clock rate is specified as the minimum SCLx low time (I2CxLBRG), plus the minimum SCLx high time (I2CxHBRG). In most cases, that is defined by two BRG periods (T_{BRG}).

The reload value for the BRG is the I2CxHBRG/I2CxLBRG register, as illustrated in Figure 23-5. When the BRG is loaded with this value, the generator counts down to zero and stops until another reload has taken place. The BRG is reloaded automatically on baud rate restart. For example, if clock synchronization is taking place, the BRG will be reloaded when the SCLx pin is sampled high.

Note: The I2CxHBRG/I2CxLBRG register values that are less than four are not supported.

Figure 23-5. Baud Rate Generator Block Diagram



Equation 23-1 shows the formula for computing the BRG reload value.

Equation 23-1. BRG Reload Value Calculation

$$I2CxHBRG/I2CxLBRG[23:0] = \left[\left(\left(\frac{1}{2 \bullet FSCL}\right) - Delay\right) \bullet FPB\right] - 3$$

Where the typical value of delay is 200 ns.

Note: Equation 23-1 is only for a design guideline. Due to system-dependent parameters, the actual baud rate may differ slightly. Testing is required to confirm that the actual baud rate meets the system requirements; otherwise, the value of the I2CxHBRG/I2CxLBRG register has to be adjusted.

Table 23-2. I^{2}C Clock Rates

F_{PB}	FSCL	I2CxLBRG/I2CxHBRG (Decimal)
200 MHz	1 MHz	57
200 MHz	400 kHz	207
200 MHz	100 kHz	957
100 MHz	1 MHz	27

Table 23-2. I²C Clock Rates (continued)

F_{PB}	FSCL	I2CxLBRG/I2CxHBRG (Decimal)
100 MHz	400 kHz	102
100 MHz	100 kHz	477
50 MHz	1 MHz	12
50 MHz	400 kHz	50
50 MHz	100 kHz	237
16 MHz	1 MHz	—
16 MHz	400 kHz	13
16 MHz	100 kHz	73
8 MHz	1 MHz	—
8 MHz	400 kHz	5
8 MHz	100 kHz	35

23.5.1.3. Host Input Compensation

The host input compensation delay (I2CxHIDLYC) is used for feedback input delay compensation. The feedback path delay is expected within the calculated BRG time-out, but the estimated path delay may go slightly higher due to system-dependent parameters (external voltage and pull-up resistance connected) to the SCK and SDA. When the feedback path delay is greater than the calculated BRG time-out, the BRG time-out expiration must be extended using the programmable delay compensation.

Equation 23-2. HDLYC Calculation

$$HIDLYCVALUE[15:0] = [(F_{PB}/(2 * FSCL) – BRG[23:0] * 1/F_{PB}] +2$$

Notes:

1. The I2CxHIDLYC is most useful to delay compensation for 1 MHz I²C clock.
2. Due to system-dependent parameters, the actual delay may differ slightly with respect to BRG. Testing is required to confirm that the actual delay values meets the system requirements. Otherwise, the value of the HIDLYC register may need to be adjusted.

23.5.2. Communicating as a Host in a Single Host Environment

The I²C module's typical operation in a system is using the I²C to communicate with an I²C peripheral, such as an I²C serial memory. In an I²C system, the host controls the sequence of all data communication on the bus. In this example, the dsPIC33AK256MPS306 device and its I²C module have the role of the single host in the system. As the single host, it is responsible for generating the SCLx clock and controlling the message protocol.

The I²C module controls individual portions of the I²C message protocol; however, sequencing the components of the protocol to construct a complete message is performed by the user software.

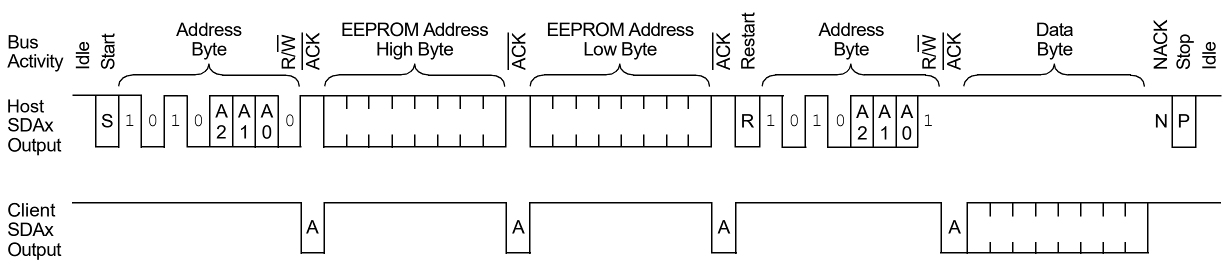
For example, a typical operation in a single host environment is to read a byte from an I²C serial EEPROM. Figure 23-6 illustrates the example message.

To accomplish this message, the user software will sequence through the following steps:

1. Assert a Start condition on SDAx and SCLx.
2. Send the I²C device address byte to the client with a write indication.
3. Wait for and verify an Acknowledge from the client.
4. Send the serial memory address high byte to the client.
5. Wait for and verify an Acknowledge from the client.
6. Send the serial memory address low byte to the client.

7. Wait for and verify an Acknowledge from the client.
8. Assert a Repeated Start condition on SDAx and SCLx.
9. Send the device address byte to the client with a read indication.
10. Wait for and verify an Acknowledge from the client.
11. Enable the host reception to receive serial memory data.
12. Generate an ACK or NACK condition at the end of a received byte of data.
13. Generate a Stop condition on SDAx and SCLx.

Figure 23-6. A Typical I^2C Message: Read of Serial EEPROM (Random Address Mode)



The module supports Host mode communication with the inclusion of the Start and Stop generators, data byte transmission, data byte reception, Acknowledge generator and a BRG. Generally, the user software will write to a control register to start a particular step, then wait for an interrupt or poll status to wait for completion. These operations are discussed in the subsequent sections.

Note: The I^2C module does not allow the queuing of events. For example, the user software is not allowed to initiate a Start condition and immediately write the I2CxTRN register to initiate transmission before the Start condition is complete. In this case, the I2CxTRN register will not be written to and the IWCOL status bit (I2CxSTAT1[7]) will be set, indicating that this write to the I2CxTRN register did not occur.

23.5.2.1. Generating Start Bus Event

To initiate a Start event, the user software sets the SEN bit (I2CxCON1[0]. Prior to setting the Start bit, the user software should check the bus free timer flag BITO (I2CxSTAT2) to ensure that the bus is in an Idle state or wait until the bus is in an Idle state.

Figure 23-7 illustrates the timing of the Start condition.

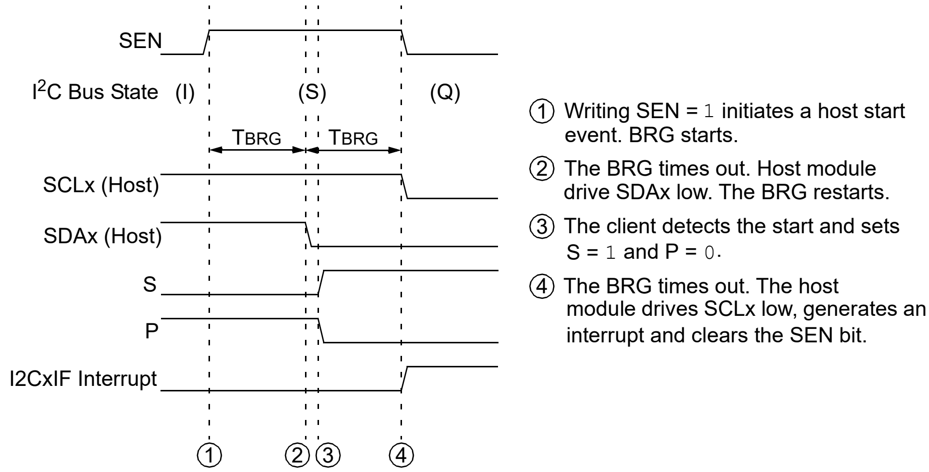
- Client logic detects the Start condition, sets the S status bit (I2CxSTAT1[3]) and clears the P status bit (I2CxSTAT1[4])
- STARTE(I2CxSTA2[14]) bit is set.
- The SEN bit is automatically cleared at completion of the Start condition.
- The I2CxIF interrupt is generated at completion of the Start condition if the HSCIE (I2CxINTC[27]) bit and HSTIE(I2CxINTC[13]) are enabled.
- The HSTACT (I2CxSTAT2[29]) bit will be set on completion of the Start condition.
- After the Start condition, the SDAx line and SCLx lines are left low (Q state).

23.5.2.1.1. IWCOL Status Flag

If the user software writes to the I2CxTRN register when a Start sequence is in progress, the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the transmit buffer are unchanged (the write does not occur).

Note: As the queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the Start condition is complete.

Figure 23-7. Host Start Timing Diagram



23.5.2.2. Host Transmission

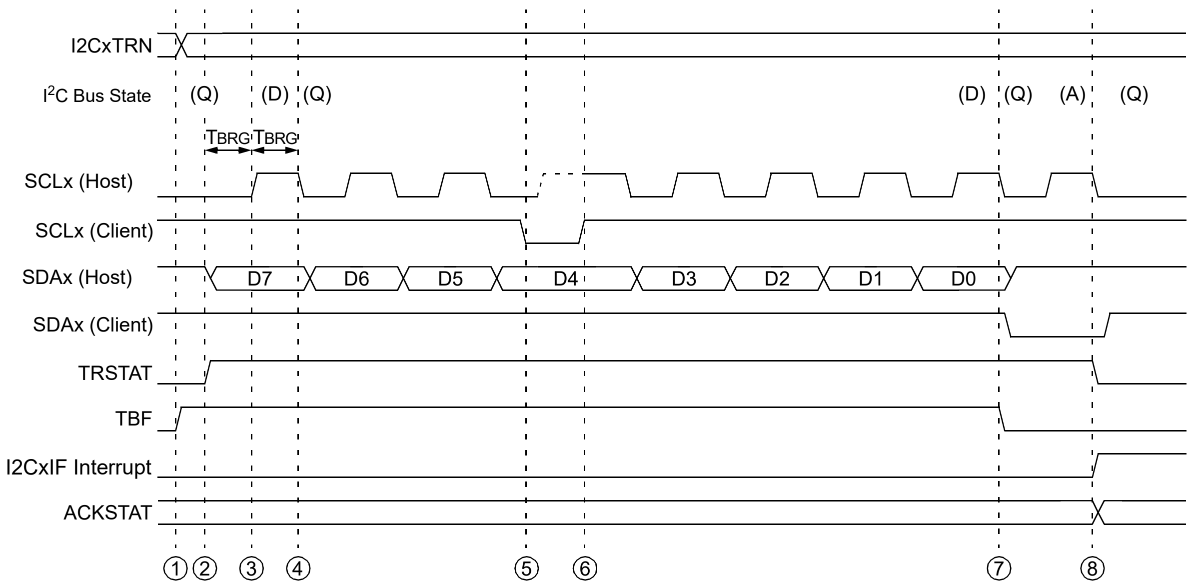
User software should configure data packet size in PSZ (I2CxCON2) excluding the address byte and PEC, if enabled. The transmission of a data byte, a 7-bit device address byte or the second byte of a 10-bit address is accomplished by writing the appropriate value to the I2CxTRN register. Before transmitting, ND/$\overline{\text{A}}$ (I2CxCON2[18]) has to be cleared/set to an indicated address/data transmission. Loading the I2CxTRN register will start the following process:

1. The user software loads the I2CxTRN register with the data/address byte to transmit.
2. Writing to the I2CxTRN register sets the TBF bit (I2CxSTAT1[0]).
3. The data/address byte is shifted out through the SDAx pin until all eight bits are transmitted. Each bit of address or data will be shifted out onto the SDAx pin after the falling edge of SCLx.
4. On the ninth SCLx clock, the module shifts in the ACK bit from the client device and writes its value into the ACKSTAT status bit (I2CxSTAT1[15]).
5. The module generates the I2CxIF interrupt at the end of the ninth SCLx clock cycle if the HDTXIE (I2CxINTC[1]) bit and HSTIE (I2CxINTC[13]) are enabled.

The module does not generate or validate the data bytes. The contents and usage of the bytes are dependent on the state of the message protocol maintained by the user software.

The sequence of events that occur during the host transmission are provided in Figure 23-8.

Figure 23-8. Host Transmission Timing Diagram



① Writing to the I2CxTRN register will start a host transmission event. The TBF status bit is set.

② The BRG starts. The MSB of the I2CxTRN register drives SDAx. SCL remains low. The TRSTAT status bit is set.

③ The BRG times out. SCLx is released and the BRG restarts.

④ The BRG times out. SCLx is driven low. After SCLx is detected low, the next bit of the I2CxTRN register drives SDAx.

⑤ While SCLx is low, the client can also pull SCLx low to initiate a Wait (clock stretch).

⑥ Host has already released SCLx and client can release to end the Wait . The BRG restarts.

⑦ At the falling edge of the eighth SCLx clock, the host releases SDAx. The TBF status bit is cleared.
The client drives an $\overline{\text{ACK}}$/NACK .

⑧ At the falling edge of the ninth SCLx clock, the host generates the interrupt. SCLx remains low until the next event.
The client releases SDAx and the TRSTAT status bit is clear.

23.5.2.2.1. Sending a 7-Bit Address to the Client

Sending a 7-bit device address involves sending one byte to the client. A 7-bit address byte must contain the seven bits of the I²C device address, and a R/W status bit that defines whether the message will be a write to the client (host transmission and client reception) or a read from the client (client transmission and host reception).

Note: In a 7-Bit Addressing mode, each node using the I²C protocol should be configured with a unique address that is stored in the I2CxADD register.
While transmitting the address byte, the host must shift the address bits [7:0] left by one bit and configure bit 0 as the R/$\overline{\text{W}}$ bit.

23.5.2.2.2. Sending a 10-Bit Address to the Client

Sending a 10-bit device address involves sending two bytes to the client. The first byte contains five bits of the I²C device address reserved for 10-Bit Addressing modes and two bits of the 10-bit address. As the next byte, which contains the remaining eight bits of the 10-bit address, must be received by the client, the R/W status bit in the first byte must be ‘0’, indicating host transmission and client reception. If the message data are also directed toward the client, the host can continue sending data. However, if the host expects a reply from the client, a repeated start sequence and upper 10-bit address with the R/W status bit at ‘1’ will change the R/W state of the message to a read of the client.

Note: In a 10-Bit Addressing mode, each node using the I^2C protocol should be configured with a unique address that is stored in the I2CxADD register.
While transmitting the first address byte, the host must shift the address bits[9:8] left by one bit and configure bit 0 as the R/W bit.

23.5.2.2.3. Receiving Acknowledge from the Client

On the falling edge of the eighth SCLx clock, the TBF status bit is cleared and the host will deassert the SDAx pin, allowing the client to respond with an Acknowledge. The host will then generate a ninth SCLx clock.

This allows the client device being addressed to respond with an $\overline{\text{ACK}}$ bit during the ninth bit time if an address match occurs or data were received properly. A client sends an Acknowledge when it has recognized its device address (including a general call) or when the client has properly received its data.

The status of $\overline{\text{ACK}}$ is written into the ACKSTAT bit (I2CxSTAT1[15]) on the falling edge of the ninth SCLx clock. After the ninth SCLx clock, the module generates the I2CxIF interrupt if the HDTXIE (I2CxINTC[1]) bit and HSTIE(I2CxINTC[13]) are enabled, and enters into the Idle state until the next data byte is loaded into the I2CxTRN register.

23.5.2.2.4. ACKSTAT Status Flag

The ACKSTAT bit (I2CxSTAT1[15]) is cleared when the client has sent an Acknowledge ($\overline{\text{ACK}}$ = `0`) and is set when the client does not Acknowledge ($\overline{\text{ACK}}$ = `1`).

23.5.2.2.5. TBF Status Flag

When transmitting, the TBF status bit (I2CxSTAT1[0]) is set when the CPU writes to the I2CxTRN register and is cleared when all eight bits are shifted out.

23.5.2.2.6. IWCOL Status Flag

If the user software attempts to write to the I2CxTRN register when a transmit is already in progress (that is, the module is still shifting a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur). The IWCOL status bit must be cleared in the user software.

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the transmit condition is complete.

23.5.2.2.7. Data Transmission to Client

User software should set ND/$\overline{\text{A}}$ before data are transmitted, and I2CxTRN is then loaded with the data. Packet size PSZ (I2CxSTAT2) decrements on each data transfer.

The host can be configured to ignore the NACK client response to continue transmits to the client. The HNACKIGN(I2CxCON2[21]) has to be set, and the transmit can then continue.

Once the packet size becomes zero, the end of packet (EOP) will be set (I2CxSTAT2[24]), if enabled (EOPSC, I2CxCON2[20:19]).

If the transmit buffer (I2CxTRN) is empty and packet size (I2CxCON2[15:0]) has not reached zero, the I2CxSTAT.SSPND bit is set and the host drives SCL low. Once the transmit buffer (I2CxTRN) is loaded, I2CxSTAT.SSPND will be cleared and the transmission will be resumed (see Host Transmission (7-bit Address) and Host Transmission (10-bit Address)).

23.5.2.3. Receiving Data from a Client Device

The host can receive data from the client device after the host has transmitted the client address with a R/W status bit value of ‘`1`’. After receiving an ACK from the client for the address, configure the packet size (PSZ,(I2CxCON2[15:0]) and then enable the RCEN bit (I2CxCON1[3]) for the first data byte to be received. If SMART mode SMEN(I2CxCON2 [17]) is enabled, then the hardware will automatically set RCEN(I2CxCON1[3]) based on the status of RBF(I2CxSTAT1[1]) and PSZ(I2CxCON2[15:0]) for remaining bytes; otherwise, the user software should set RCEN(I2CxCON1[3]) for every byte to be received. The host logic begins to generate clocks, and before each falling edge of the SCLx, the SDAx line is sampled and data are shifted into the I2CxRSR register.

Note: The lower five bits of the I2CxCON1 register must be '0' before attempting to set the RCEN bit. This ensures that the host logic is inactive.

After the falling edge of the eighth SCLx clock, the following events occur:

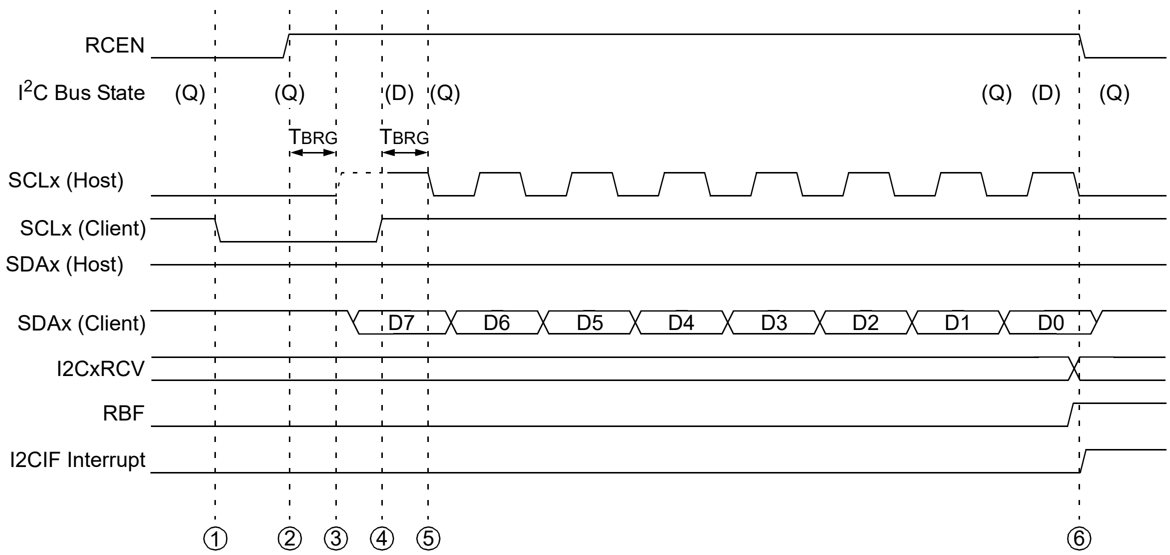
- The RCEN bit is automatically cleared.
- The contents of the I2CxRSR register transfer into the I2CxRCV register.
- The RBF status bit (I2CxSTAT1[1]) is set.
- The I^2C module generates the I2CxIF interrupt if the HDRXIE(I2CxINTC[0]) bit and HSTIE(I2CxINTC[13]) bit are enabled.

When the CPU reads the receive buffer (I2CxRCV), the RBF status bit is automatically cleared. The user software can process the data and then execute an Acknowledge sequence.

If SMART mode is enabled, the hardware will automatically set RCEN(I2CxCON1[3]) based on the status of RBF(I2CxSTAT1[1]) and PSZ(I2CxCON2[15:0]) for the remaining bytes; otherwise the user software should set RCEN(I2CxCON1[3]) for every byte to be received.

If the receive buffer (I2CxRCV) is full and the packet size (I2CxCON2[15:0]) has not reached zero, the I2CxSTAT.SSPND bit is set and the host drives SCL low. The user should remove the Receiver Full condition by reading the receive buffer (I2CxRCV). Once the full condition is removed, I2CxSTAT.SSPND will be cleared by hardware. Once the packet size becomes zero, the end of packet (EOP) will be set (I2CxSTAT2[24]), if enabled (EOPSC, I2CxCON2[20:19]). The sequence of events that occurs during the host reception is illustrated in Figure 23-9.

Figure 23-9. Host Reception Timing Diagram



① Typically, the client can pull SCLx low (clock stretch) to request a Wait to prepare the data response. The client will drive the MSB of the data response on SDAx when ready.

② Writing the RCEN bit will start a host reception event. The BRG starts. SCLx remains low.

③ The BRG times out. The host attempts to release SCLx.

④ When the client releases SCLx, the BRG restarts.

⑤ The BRG times out. The MSB of the response is shifted to the I2CxRSR register. SCLx is driven low for the next baud interval.

⑥ At the falling edge of the eighth SCLx clock, the I2CxRSR register is transferred to the I2CxRCV register. The module clears the RCEN bit. The RBF status bit is set. The host generates the interrupt.

23.5.2.3.1. RBF Status Flag

When receiving data, the RBF status bit (I2CxSTAT1[1]) is set when a device address or data byte is loaded into the I2CxRCV register from the I2CxRSR register. It is cleared when the user software reads the I2CxRCV register.

23.5.2.3.2. I2COV Status Flag

If another byte is received in the I2CxRSR register while the RBF status bit remains set and the previous byte remains in the I2CxRCV register, the I2COV status bit (I2CxSTAT1[6]) is set and the data in the I2CxRSR register are lost.

Leaving the I2COV status bit set does not inhibit further reception. If the RBF status bit is cleared by reading the I2CxRCV register, and the I2CxRSR register receives another byte, that byte will be transferred to the I2CxRCV register.

23.5.2.3.3. IWCOL Status Flag

If the user software writes to the I2CxTRN register when a receive is already in progress (that is, the I2CxRSR register is still shifting in a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur) (see Host Transmission (7-bit Address) and Host Transmission (10-bit Address)).

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the data reception condition is complete.

23.5.2.4. Acknowledge Generation

User software should configure an Acknowledge sequence using ACKC bits (I2CxCON2[23:22]). When ACKC is configured with "00", setting the ACKEN bit (I2CxCON1[4]) enables the generation of a host Acknowledge sequence.

Note: The lower five bits of the I2CxCON or I2CxCONL register must be '0' (host logic inactive) before attempting to set the ACKEN bit.

Figure 23-10 illustrates an $\overline{\text{ACK}}$ sequence, and Figure 23-11 illustrates a NACK sequence. The ACKDT bit (I2CxCON1[5] specifies an $\overline{\text{ACK}}$ or NACK sequence.

After two baud periods, the ACKEN bit is automatically cleared and the module generates the I2CxIF interrupt, if the HACKSIE(I2CxINTC[11]) bit and HSTIE(I2CxINTC[13]) are enabled.

Figure 23-10. Host Acknowledge ($\overline{\text{ACK}}$) Timing Diagram

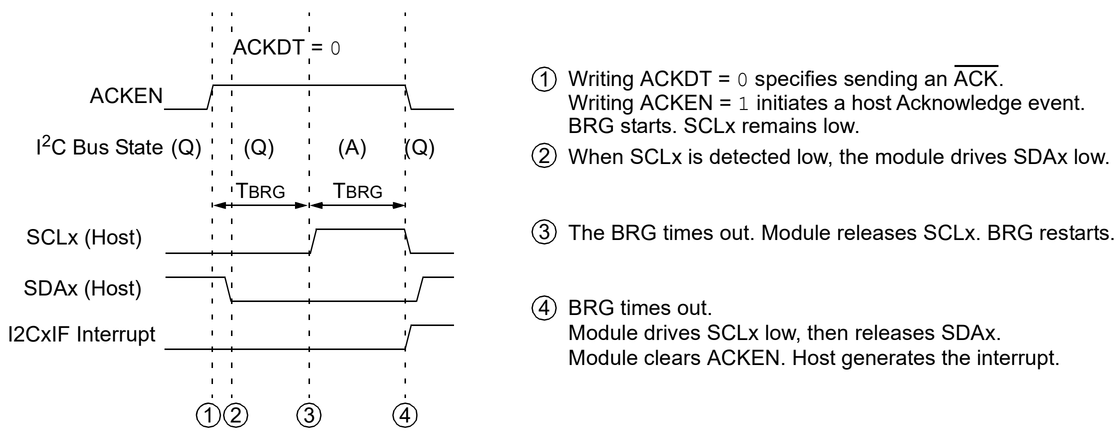
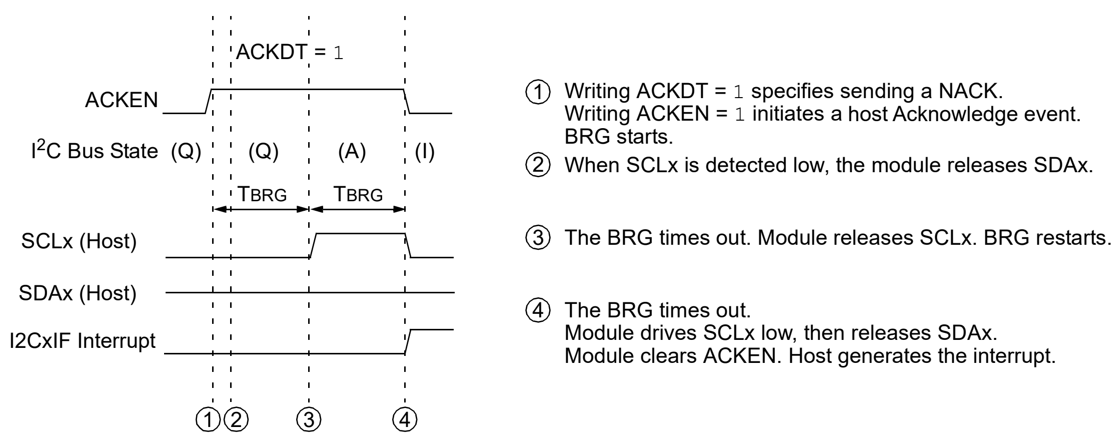


Figure 23-11. Host Not Acknowledge (NACK) Timing Diagram



23.5.2.4.1. IWCOL Status Flag

If the user software writes to the I2CxTRN register when a receive is already in progress (that is, the I2CxRSR register is still shifting in a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur) (see Host Transmission (7-bit Address) and Host Transmission (10-bit Address)).

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the data reception condition is complete.

23.5.2.5. Generating a Stop Bus Event

Setting the PEN bit (I2CxCON1[2]) enables the generation of a host Stop sequence.

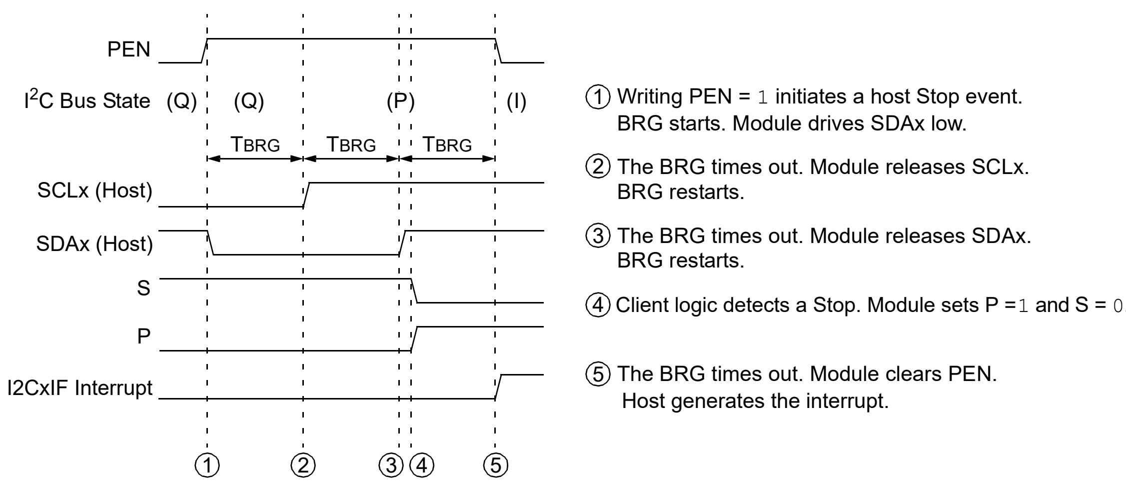
Note: The lower five bits of the I2CxCON1 register must be '0' (host logic inactive) before attempting to set the PEN bit.

When the PEN bit is set, the host generates the Stop sequence, as illustrated in Figure 23-12.

- The client detects the Stop condition, sets the P status bit (I2CxSTAT1[4]) and clears the S status bit (I2CxSTAT1[3]).
- STOPE(I2CxSTAT2[15]) bit is set.
- The PEN bit is automatically cleared.
- The HSTACT(I2CxSTAT2[29]) bit is cleared when the STOP is sent.
- The module generates the I2CxIF interrupt if the HPCIE(I2CxINTC[28]) bit and HSTIE(I2CxINTC[13]) are enabled.

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the Stop condition is complete.

Figure 23-12. Host Stop Timing Diagram



23.5.2.5.1. IWCOL Status Flag

If the user software writes to the I2CxTRN register when a receive is already in progress (that is, the I2CxRSR register is still shifting in a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur) (see Host Transmission (7-bit Address) and Host Transmission (10-bit Address)).

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the data reception condition is complete.

23.5.2.6. Generating a Repeated Start Bus Event

Setting the RSEN bit (I2CxCON1[1]) enables the generation of a host repeated start sequence, as illustrated in Figure 23-13.

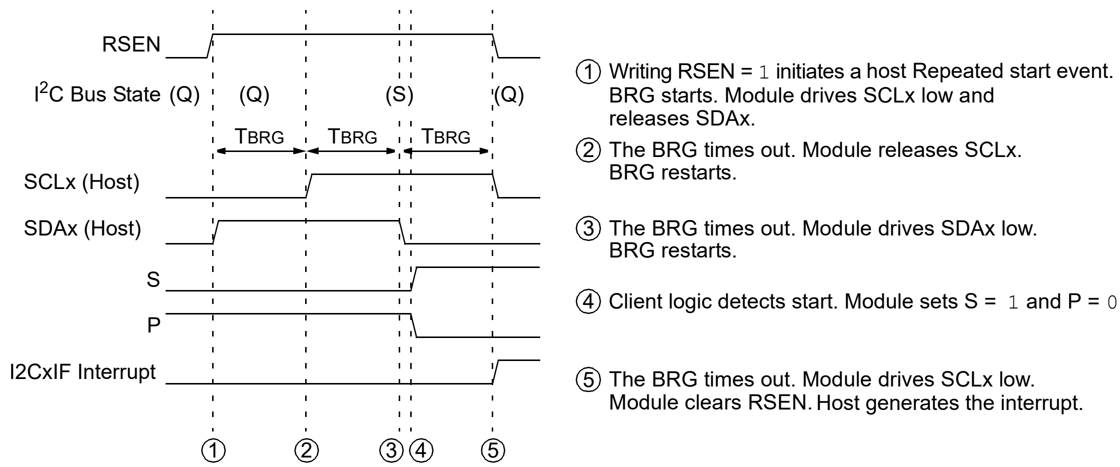
Note: The lower five bits of the I2CxCON1 register must be '0' (host logic inactive) before attempting to set the RSEN bit.

To generate a Repeated Start condition, the user software sets the RSEN bit. The host module asserts the SCLx pin low. When the module samples the SCLx pin low, the module releases the SDAx pin for 1 T_{BRG}. When the BRG times out and the module samples SDAx high, the module deasserts the SCLx pin. When the module samples the SCLx pin high, the BRG reloads and begins counting. SDAx and SCLx must be sampled high for 1 T_{BRG}. This action is then followed by assertion of the SDAx pin low for 1 T_{BRG} while SCLx is high.

The following is the repeated start sequence:

1. The client detects the Start condition, sets the S status bit (I2CxSTAT1[3]) and clears the P status bit (I2CxSTAT1[4]).
2. The RSEN bit is automatically cleared.
3. The I²C module generates the I2CxIF interrupt if the HSCIE(I2CxINTC[27]) bit and HSTIE(I2CxINTC[13]) are enabled.

Figure 23-13. Host Repeated Start Timing Diagram



23.5.2.6.1. IWCOL Status Flag

If the user software writes to the I2CxTRN register when a receive is already in progress (that is, the I2CxRSR register is still shifting in a data byte), the IWCOL status bit (I2CxSTAT1[7]) is set and the contents of the buffer are unchanged (the write does not occur) (see Host Transmission (7-bit Address) and Host Transmission (10-bit Address)).

Note: Because queuing of events is not allowed, writing to the lower five bits of the I2CxCON1 register is disabled until the data reception condition is complete.

23.5.2.7. Building Complete Host Messages

As described in Communicating as a Host in a Single Host Environment, the user software is responsible for constructing messages with the correct message protocol. The module controls individual portions of the I²C message protocol; however, sequencing of the components of the protocol to construct a complete message is performed by the user software.

The user software can use polling or interrupt methods while using the module. The timing diagrams shown in this document use interrupts for detecting various events.

The user software can use the SEN, RSEN, PEN, RCEN and ACKEN bits (Least Significant five bits of the I2CxCON1 register), and the TRSTAT status bit as a state flag when progressing through a message. For example, Table 23-3 shows some example state numbers associated with bus states.

Table 23-3. Host Message Protocol States

Example State Number[(1)]	I2CxCON[4:0] or I2CxCONL[4:0]	TRSTAT (I2CxSTAT[14])	State
0	00000	0	Bus Idle or Wait
1	00001	N/A	Sending Start Event
Note:			
1. The example state numbers are for reference only. The user software can assign the state numbers as desired.			

Table 23-3. Host Message Protocol States (continued)

Example State Number(1)	I2CxCON[4:0] or I2CxCONL[4:0]	TRSTAT (I2CxSTAT[14])	State
2	00000	1	Host Transmitting
3	00010	N/A	Sending Repeated Start Event
4	00100	N/A	Sending Stop Event
5	01000	N/A	Host Reception
6	10000	N/A	Host Acknowledgment

Note:
1. The example state numbers are for reference only. The user software can assign the state numbers as desired.

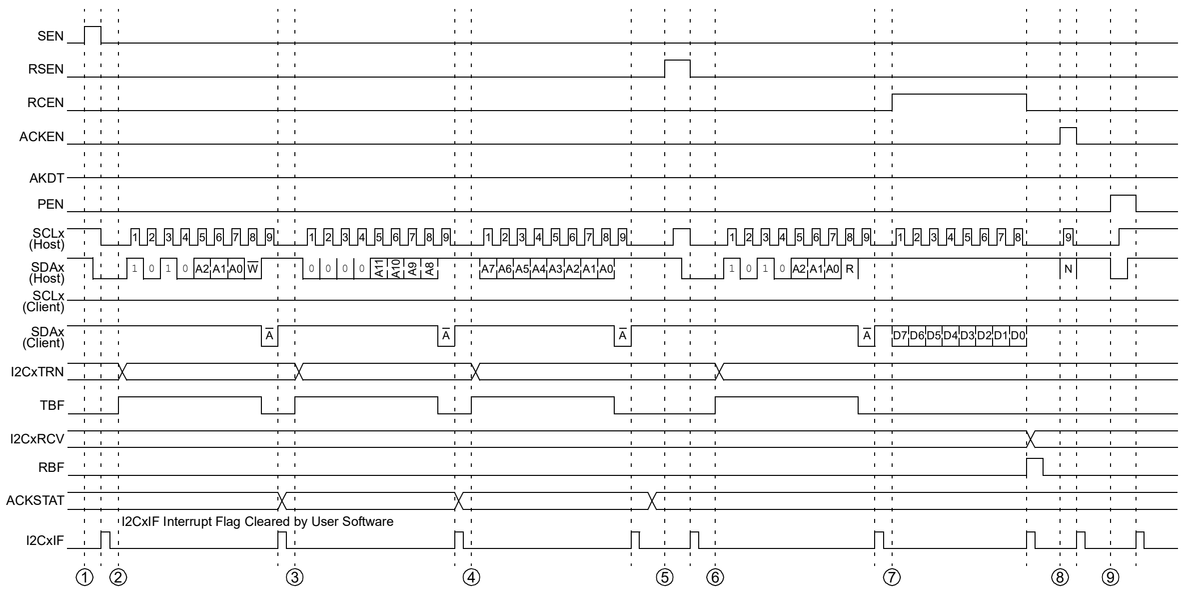
The user software will begin a message by issuing a Start condition. The user software will record the state number corresponding to the Start.

As each event completes and generates an interrupt, the interrupt handler may check the state number. Therefore, for a Start state, the interrupt handler will confirm execution of the start sequence and then start a host transmission event to send the I^{2}C device address, changing the state number to correspond to the host transmission.

On the next interrupt, the interrupt handler will again check the state, determining that a host transmission just completed. The interrupt handler will confirm successful transmission of the data and then move on to the next event, depending on the contents of the message. In this manner, on each interrupt, the interrupt handler will progress through the message protocol until the complete message is sent.

Figure 23-14 provides a detailed examination of the same message sequence as shown in Figure 23-6. Figure 23-15 provides a few simple examples of the messages using a 7-bit addressing format. Figure 23-18 provides an example of a 10-bit addressing format message sending data to a client. Figure 23-19 provides an example of a 10-bit addressing format message receiving data from a client.

Figure 23-14. Host Message (Typical I^{2}C Message: Read of Serial EEPROM)



① Setting the SEN bit begins a Start event.

② Writing the I2CxTRN register starts a host transmission. The data are the serial EEPROM device address byte, with the R/W status bit clear, indicating a write.

③ Writing the I2CxTRN register starts a host transmission. The data are the first byte of the EEPROM data address.

④ Writing the I2CxTRN register starts a host transmission. The data are the second byte of the EEPROM data address.

⑤ Setting the RSEN bit starts a Repeated Start event.

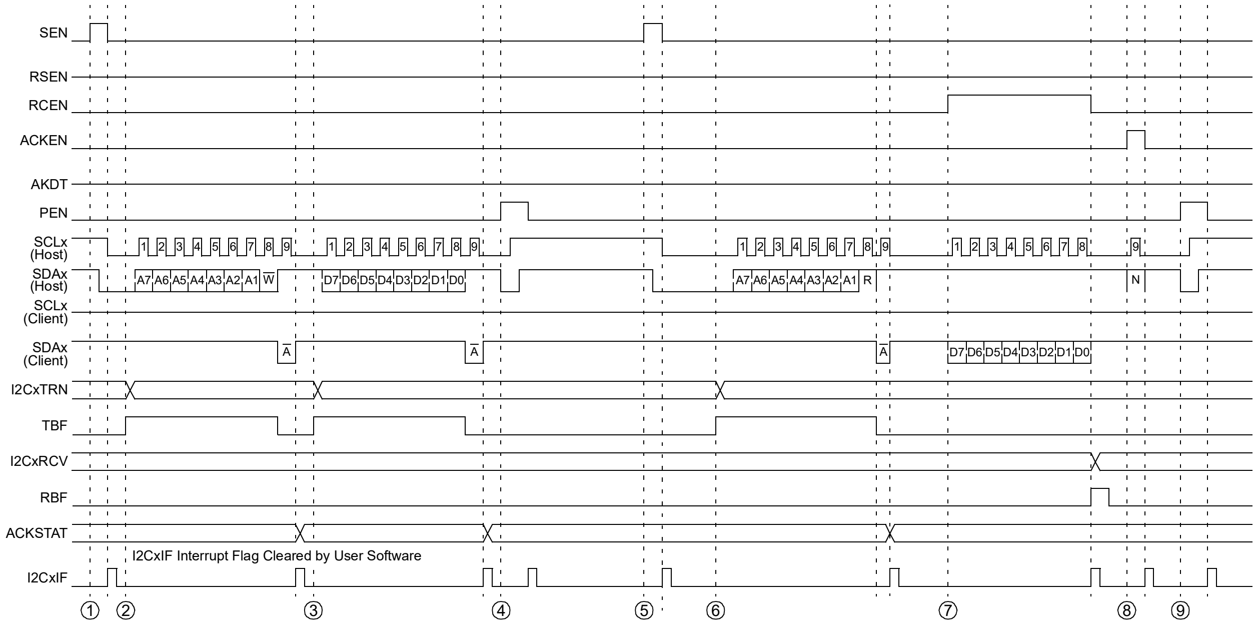
⑥ Writing the I2CxTRN register starts a host transmission. The data are a resend of the serial EEPROM device address byte, but with R/W status bit set, indicating a read.

⑦ Setting the RCEN bit starts a host reception. On interrupt, the user software reads the I2CxRCV register, which clears the RBF status bit.

⑧ Setting the ACKEN bit starts an Acknowledge event. ACKDT = 1 to send a NACK.

⑨ Setting the PEN bit starts a host Stop event.

Figure 23-15. Host Message (7-Bit Address: Transmission and Reception)



① Setting the SEN bit begins a Start event.

② Writing the I2CxTRN register starts a host transmission. The data are the address byte with the R/W̄ status bit clear.

③ Writing the I2CxTRN register starts a host transmission. The data are the message byte.

④ Setting the PEN bit starts a host Stop event.

⑤ Setting the SEN bit begins a Start event. An interrupt is generated on completion of the Start event.

⑥ Writing the I2CxTRN register starts a host transmission. The data are the address byte with the R/W̄ status bit set.

⑦ Setting the RCEN bit starts a host reception.

⑧ Setting the ACKEN bit starts an Acknowledge event. ACKDT = `1` to send a NACK.

⑨ Setting the PEN bit starts a host Stop event.

Figure 23-16. I²C Host, 7-Bit Address, Read with RXIF without CRC

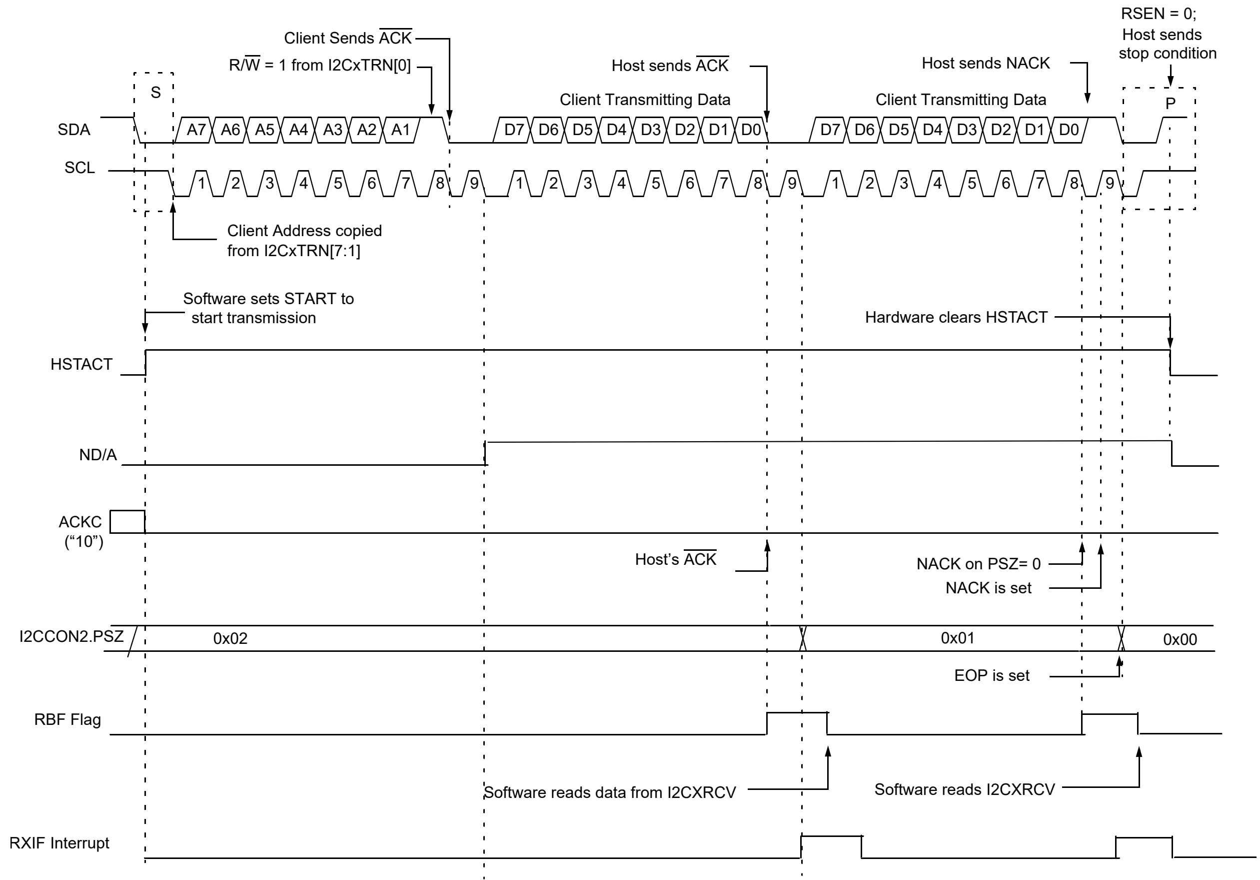


Figure 23-17. I2C Host, 7-Bit Address, Write with TXIF without CRC

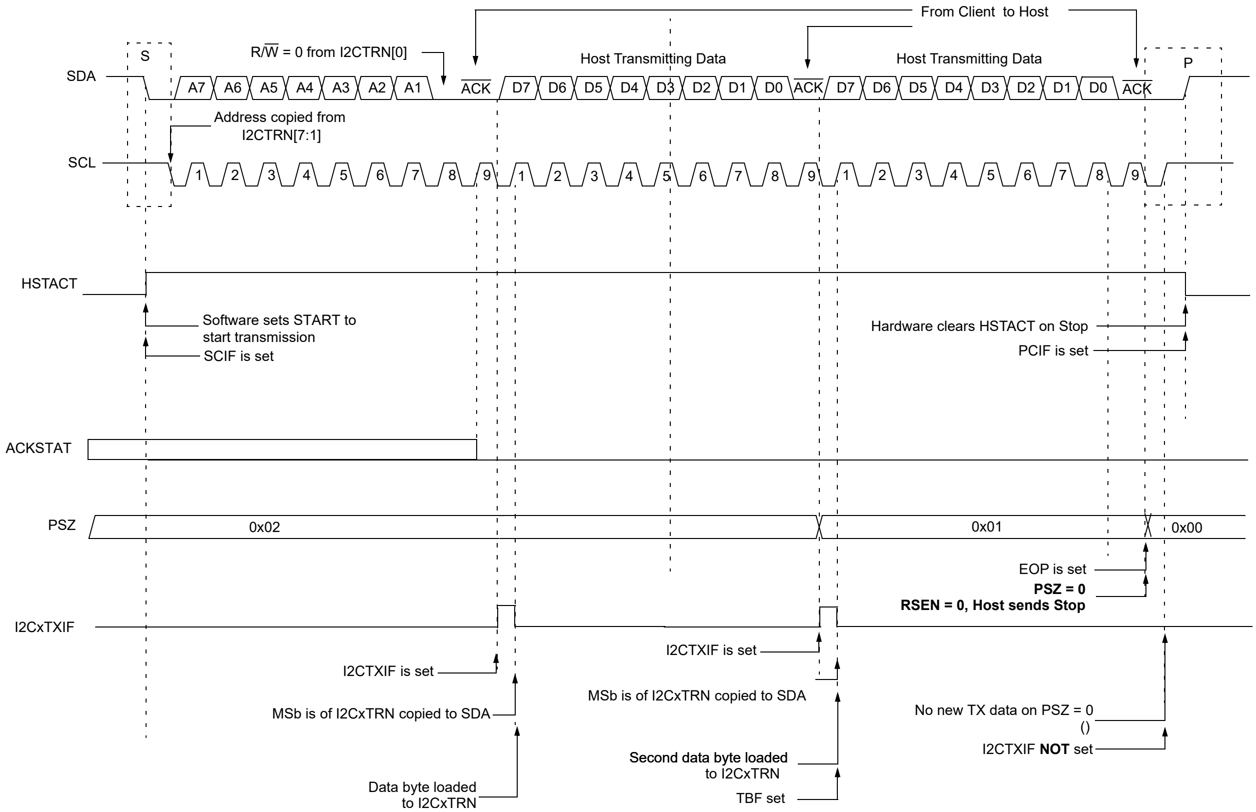
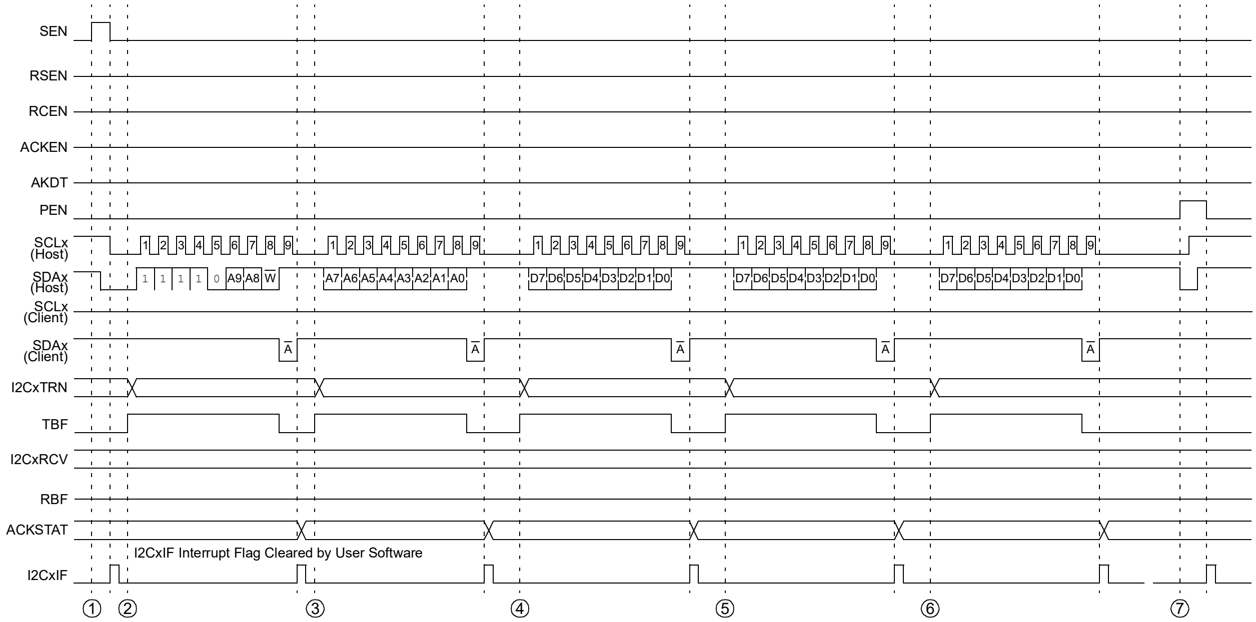


Figure 23-18. Host Message (10-Bit Transmission)



① Setting the SEN bit begins a Start event.

② Writing the I2CxTRN register starts a host transmission. The data are the first byte of the address.

③ Writing the I2CxTRN register starts a host transmission. The data are the second byte of the address.

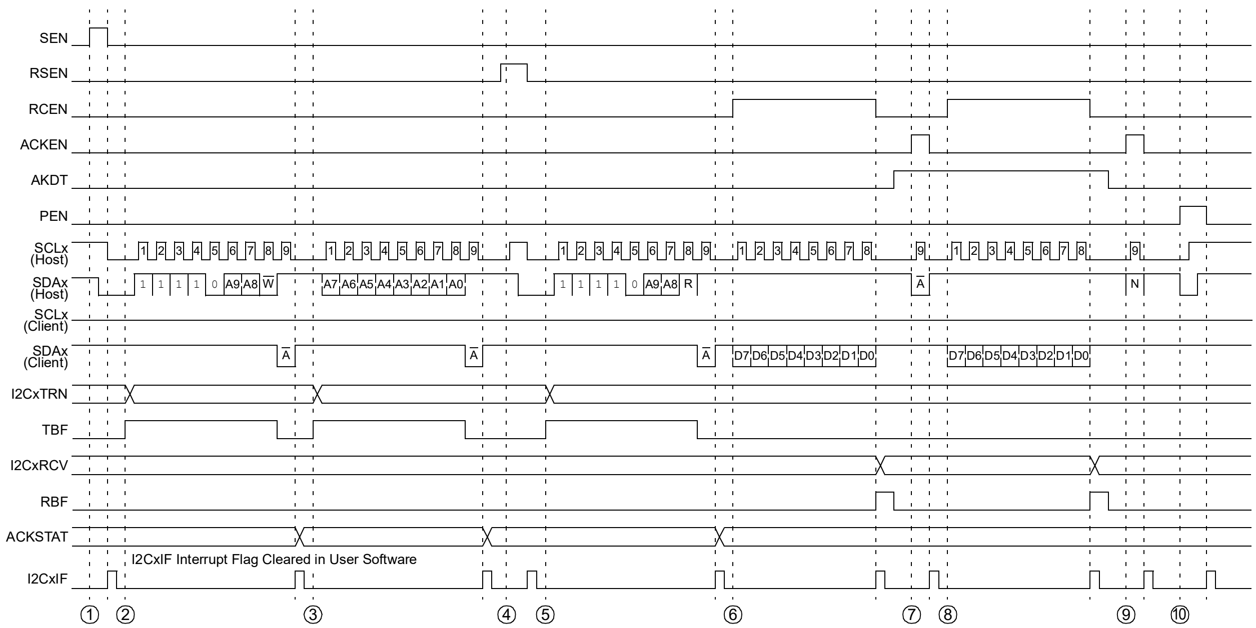
④ Writing the I2CxTRN register starts a host transmission. The data are the first byte of the message data.

⑤ Writing the I2CxTRN register starts a host transmission. The data are the second byte of the message data.

⑥ Writing the I2CxTRN register starts a host transmission. The data are the third byte of the message data.

⑦ Setting the PEN bit starts a host Stop event.

Figure 23-19. Host Message (10-Bit Reception)



① Setting the SEN bit begins a Start event.

② Writing the I2CxTRN register starts a host transmission. The data are the first byte of the address with the R/W status bit cleared.

③ Writing the I2CxTRN register starts a host transmission. The data are the second byte of the address.

④ Setting the RSEN bit starts a host Restart event.

⑤ Writing the I2CxTRN register starts a host transmission. The data are a resend of the first byte with the R/W status bit set.

⑥ Setting the RCEN bit starts a host reception. On interrupt, the user software reads the I2CxRCV register, which clears the RBF status bit.

⑦ Setting the ACKEN bit starts an Acknowledge event. ACKDT = `0` to send $\overline{\text{ACK}}$.

⑧ Setting the RCEN bit starts a host reception.

⑨ Setting the ACKEN bit starts an Acknowledge event. ACKDT = `1` to send NACK.

⑩ Setting the PEN bit starts a host Stop event.

23.5.3. Communicating As a Host In A Multi-Host Environment

The I²C protocol allows more than one host to be attached to a system bus. Taking into account that a host can initiate message transactions and generate clocks for the bus, the protocol has methods to account for situations where more than one host is attempting to control the bus. The clock synchronization ensures that multiple nodes can synchronize their SCLx clocks to result in one common clock on the SCLx line. The bus arbitration ensures that if more than one node attempts a message transaction, only one node will be successful in completing the message. The other nodes lose bus arbitration and are left with a bus collision.

23.5.3.1. Multi-Host Operation

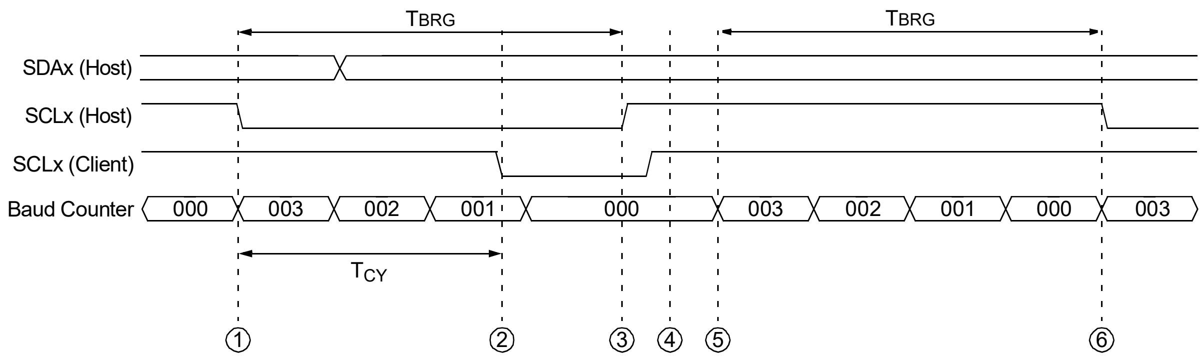
The host module has no special settings to enable the multi-host operation. The module performs the clock synchronization and bus arbitration at all times. If the module is used in a single host environment, clock synchronization only occurs between the host and clients, and bus arbitration does not occur.

23.5.3.2. Host Clock Synchronization

In a multi-host system, different hosts can have different baud rates. The clock synchronization ensures that when these hosts are attempting to arbitrate the bus, their clocks will be coordinated.

The clock synchronization occurs when the host deasserts the SCLx pin (SCLx intendeds to float high). When the SCLx pin is released, the BRG is suspended from counting until the SCLx pin is actually sampled high. When the SCLx pin is sampled high, the BRG is reloaded with the contents of I2CxHBRG[23:0] and I2CxLBRG[23:0] begin counting. This ensures that the SCLx high time will always be at least one BRG rollover count in the event that the clock is held low by an external device, as illustrated in Figure 23-20.

Figure 23-20. Baud Rate Generator Timing with Clock Synchronization



① The baud counter decrements twice per T_{CY}. On rollover, the host SCLx will transition.

② The client has pulled SCLx low to initiate a Wait.

③ At what would be the host baud counter rollover, detecting SCLx low holds the counter.

④ Logic samples SCLx once per T_{CY}. Logic detects SCLx high.

⑤ The baud counter rollover occurs on the next cycle.

⑥ On the next rollover, the host SCLx will transition.

23.5.3.3. Bus Arbitration and Bus Collision

The bus arbitration supports the multi-host system operation. The wired-AND nature of the SDAx line permits arbitration. Arbitration takes place when the first host outputs ‘1’ on the SDAx by letting the SDAx float high, and simultaneously, the second host outputs ‘0’ on the SDAx by pulling SDAx low. The SDAx signal will go low. In this case, the second host has won bus arbitration. The first host has lost bus arbitration, and thus, has a bus collision.

For the first host, the expected data on SDAx are ‘1’, while the data sampled on SDAx are ‘0’. This is the definition of a bus collision.

The first host will set the BCL bit (I2CxSTAT1[10]) and generate an error (I2CxEIF) interrupt if the BCLDIE (I2CxINTC[15]) bit is enabled. The Host module will reset the I^2C port to its Idle state.

In a multi-host operation, the SDAx line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by the host logic, with the result placed in the BCL status bit.

The states where arbitration can be lost are:

- Start condition
- Repeated Start condition
- Address, Data or Acknowledge bit
- Stop condition

23.5.3.4. Detecting Bus Collisions and Resending Messages

When a bus collision occurs, the host module sets the BCL status bit and generates an error (I2CxEIF) interrupt if the BCLDIE (I2CxINTC[15]) bit is enabled. If a bus collision occurs during a byte transmission, the transmission is stopped, the TBF status bit is cleared and the SDAx and SCLx pins are deasserted. If a bus collision occurs during a Start, Repeated Start, Stop or Acknowledge condition, the condition is aborted, the respective control bits in the I2CxCON register are cleared and the SDAx and SCLx lines are deasserted.

If the user software is expecting an interrupt at the completion of the host event, the user software can check the BCL status bit to determine if the host event completed successfully or a bus collision occurred (or it may branch to a bus collision interrupt on a bus collision), or a host interrupt occurred in case of a successful host event. If a bus collision occurs, the user software must abort sending the rest of the pending message and prepare to resend the entire message sequence, beginning with the Start condition, after the bus returns to the Idle state. The user software can monitor the S and P status bits to wait for an Idle bus. When the user software executes the host Interrupt Service Routine (ISR) and the I^{2}C bus is free, the user software can resume communication by asserting a Start condition.

23.5.3.5. Bus Collision During a Start Condition

Before issuing a Start condition, the user software should verify an Idle state of the bus using the S and P status bits. Two hosts may attempt to initiate a message at a similar point in time. Typically, the hosts will synchronize clocks and continue arbitration into the message until one loses arbitration. Any of the following conditions can cause a bus collision to occur during a Start:

- If the SDAx and SCLx pins are at a Low Logic state at the beginning of the Start condition.
- If the SCLx line is at a Low Sogic State before the SDAx line is driven low.

In either case, the host that loses arbitration during the Start condition generates a bus collision interrupt.

23.5.3.6. Bus Collision During a Repeated Start Condition

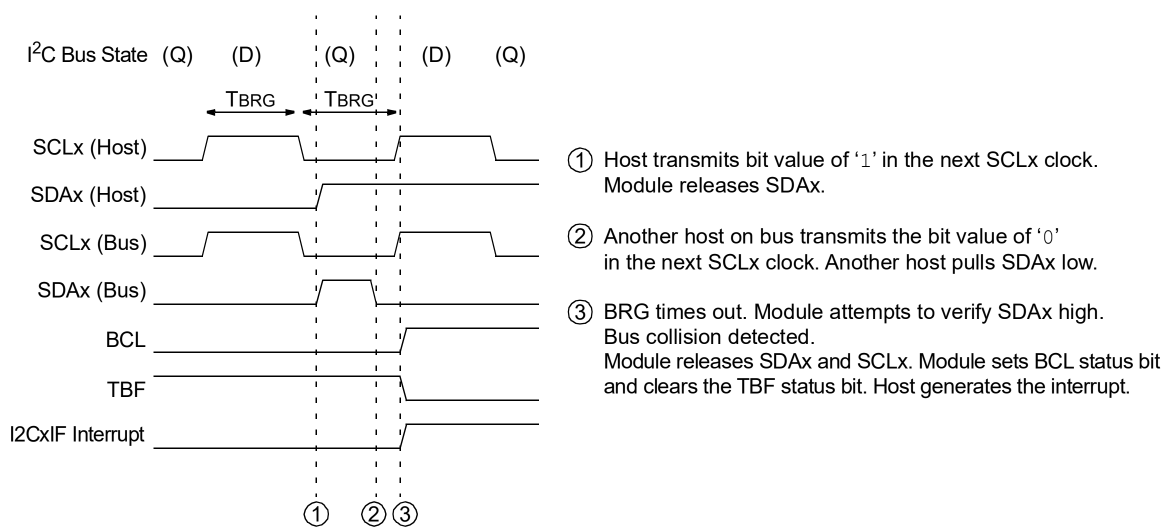
When the two hosts do not collide throughout an address byte, a bus collision can occur when one host attempts to assert a Repeated Start while another transmits data. In this case, the host generating the Repeated Start loses arbitration and generates a bus collision interrupt.

23.5.3.7. Bus Collision During Message Bit Transmission

The most typical case of data collision occurs while the host attempts to transmit the device address byte, a data byte or an Acknowledge bit.

If the user software properly checks the bus state, it is unlikely that a bus collision will occur on a Start condition. However, because another host can, at the same time, check the bus and initiate its own Start condition, it is likely that SDAx arbitration will occur and synchronize the Start of two hosts. In this condition, both hosts begin and continue to transmit their messages until one host loses arbitration on a message bit. The SCLx clock synchronization keeps the two hosts synchronized until one loses arbitration. Figure 23-21 illustrates an example of the message bit arbitration.

Figure 23-21. Bus Collision During Message Bit Transmission



23.5.3.8. Bus Collision During a Stop Condition

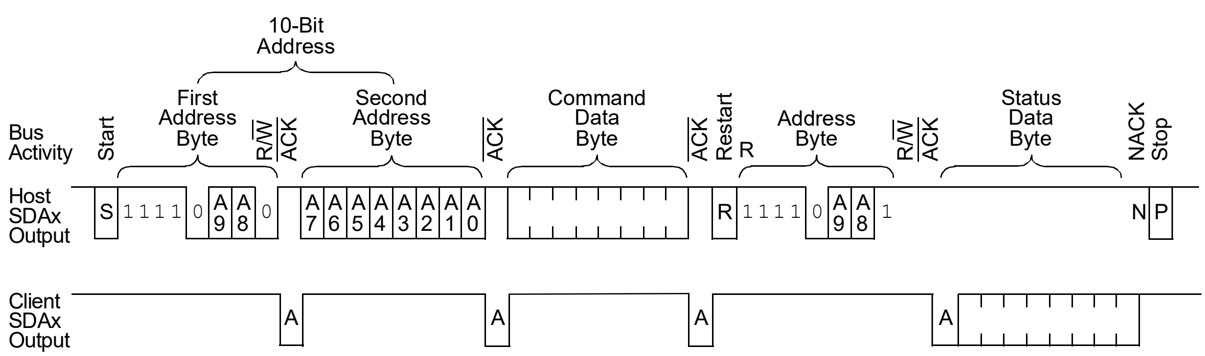
If the host software loses track of the state of the I2C bus, many existing conditions can cause a bus collision during a Stop condition. In this case, the host generating the Stop condition will lose arbitration and generate a bus collision interrupt. The HSBCL (I2CxSTAT2[25]) bit will be set to collision during a Stop condition.

23.5.4. Communicating As a Client

In some systems, particularly where multiple processors communicate with each other, the dsPIC33AK256MPS306 device can communicate as a client, as illustrated in Figure 23-22. When the I^2C module is enabled, the client is active. The client cannot initiate a message; it can only respond to a message sequence initiated by a host. The host requests a response from a particular client as defined by the device address byte in the I^2C protocol. The client replies to the host at the appropriate times as defined by the protocol.

As with the host module, sequencing the components of the protocol for the reply is a user software task. However, the client detects when the device address matches the address specified by the user software for that client.

Figure 23-22. A Typical Client I^2C Message: Multiprocessor Command/Status



After a Start condition, the client receives and checks the device address. The client can specify either a 7-bit address or a 10-bit address. When a device address is matched, the module will generate an interrupt to notify the user software that its device is selected. Based on the R/W status bit sent by the host, the client either receives or transmits data. If the client is to receive data, the client automatically generates the Acknowledge (ACK), loads the I2CxRCV register with the received value currently in the I2CxRSR register and notifies the user software through an interrupt. If the client is to transmit data, the user software must load the I2CxTRN register.

23.5.4.1. Sampling Receive Data

All the incoming bits are sampled with the rising edge of the clock (SCLx) line.

23.5.4.2. Detecting Start and Stop Conditions

The client detects the Start and Stop conditions on the bus and indicates that status on the S status bit (I2CxSTAT1[3]), STARTE (I2CxSTAT2[14]) and P status bit (I2CxSTAT1[4]), STOPE (I2CxSTAT2[15]). The Start (S) and Stop (P) status bits are cleared when a Reset occurs or when the module is disabled. After detection of a start or repeated start event, the S status bit is set and the P status bit is cleared.

After detection of a stop event:

- The P status bit is set and the S status bit is cleared.
- The CLTACT(I2CxSTAT2[30]) bit is cleared.

23.5.4.2.1. Interrupt On Start/Repeated Start and Stop Conditions (Client Mode)

The user software is notified through a client interrupt if the SCIE bit (I2CxCON1[21]) is set for a Start/Repeated Start condition or if the PCIE bit (I2CxCON1[22]) is set for a Stop condition.

23.5.4.3. Detecting the Address

Once the I^{2}C has been enabled, the client waits for a Start condition to occur. After a Start, depending on the A10M bit (I2CxCON1[10]), the client attempts to detect a 7-bit or 10-bit address. The client compares one received byte for a 7-bit address or two received bytes for a 10-bit address. A 7-bit address also contains a R/W status bit that specifies the direction of the data transfer after the address. If R/W = `0`, a write is specified and the client receives data from the host. If R/W = `1`, a read is specified and the client sends data to the host. The 10-bit address contains a R/W status bit; however, by definition, it is always R/W = `0` because the client must receive the second byte of the 10-bit address.

23.5.4.3.1. Addressing Modes

The client can be configured into different addressing modes using AMODE (I2CxCON2[31:30]). The received address will be compared to an address in:

- I2CxADDR when AMODE is `00`.
- I2CxADDR or I2CxAMSK when AMODE is `01`.
- Range of an address specified I2CxADDR and I2CxAMSK when AMODE is `10`.

23.5.4.3.2. Client Address Masking

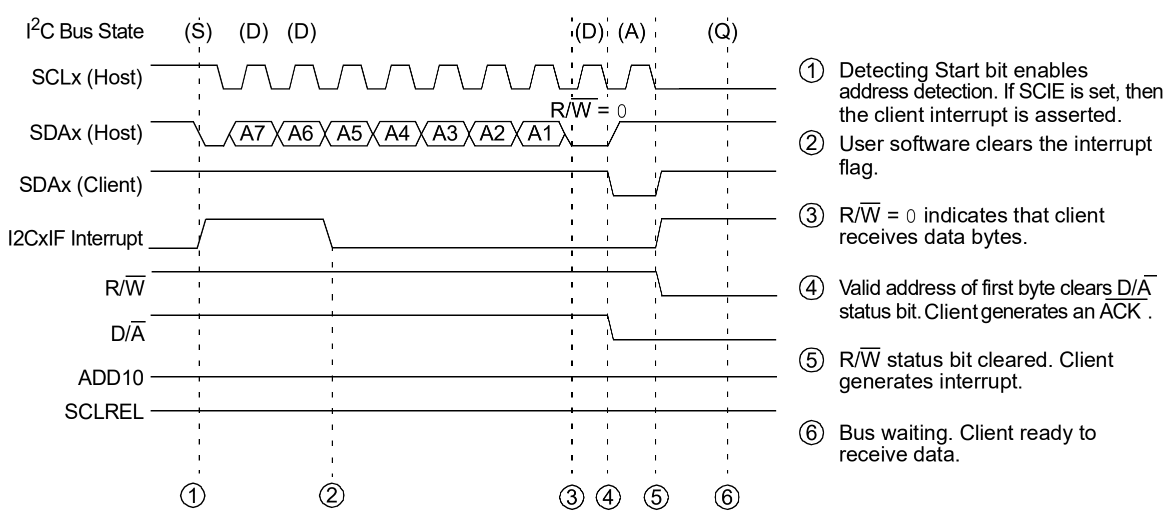
The I2CxMSK register masks the address bit positions, designating them as don't care bits for both 10-bit and 7-bit Addressing modes. When a bit in the I2CxMSK register is set (= `1`), the client responds when the bit in the corresponding location of the address is a '`0`' or '`1`'. For example, in 7-bit Client mode with I2CxMSK = `0100000`, the client module Acknowledges addresses '`0000000`' and '`0100000`' as valid.

23.5.4.3.3. 7-Bit Address and Client Write

After the Start condition, the module shifts eight bits into the I2CxRSR register, as illustrated in Figure 23-23. The value of the I2CxRSR register is evaluated against that of the I2CxADD and I2CxMSK registers on the falling edge of the eighth clock (SCLx). If the address is valid (that is, an exact match between unmasked bit positions), the following events occur:

- An ACK is generated if the AHEN bit is clear.
- The D/A and R/W status bits are cleared.
- The CLTACT (I2CxSTAT2[30]) bit is set.
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE(I2CxINTC[12]) are enabled.
- The module waits for the host to send data.

Figure 23-23. Client Write 7-Bit Address Detection Timing Diagram

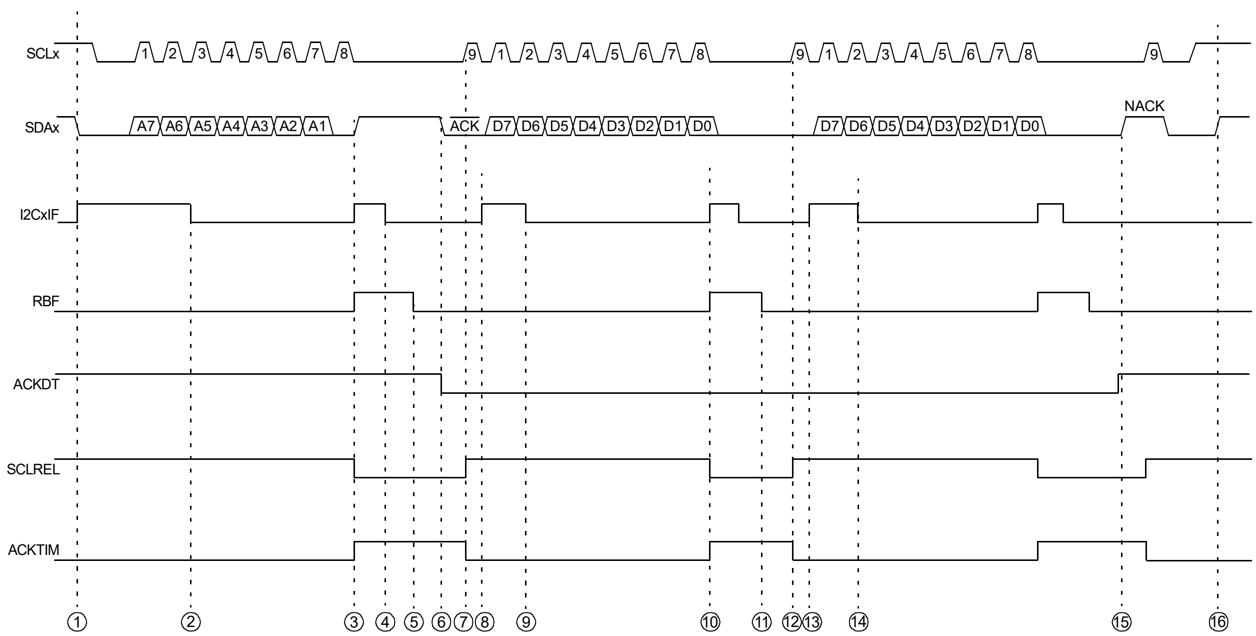


23.5.4.3.4. 7-Bit Address and Client Write with the AHEN and DHEN Bits

The client device reception, with the AHEN and DHEN bits set, operates with extra interrupts and clock stretching added after the eighth falling edge of SCLx. These additional interrupts allow the client software to decide whether it wants to `ACK` the receive address or data byte rather than the hardware.

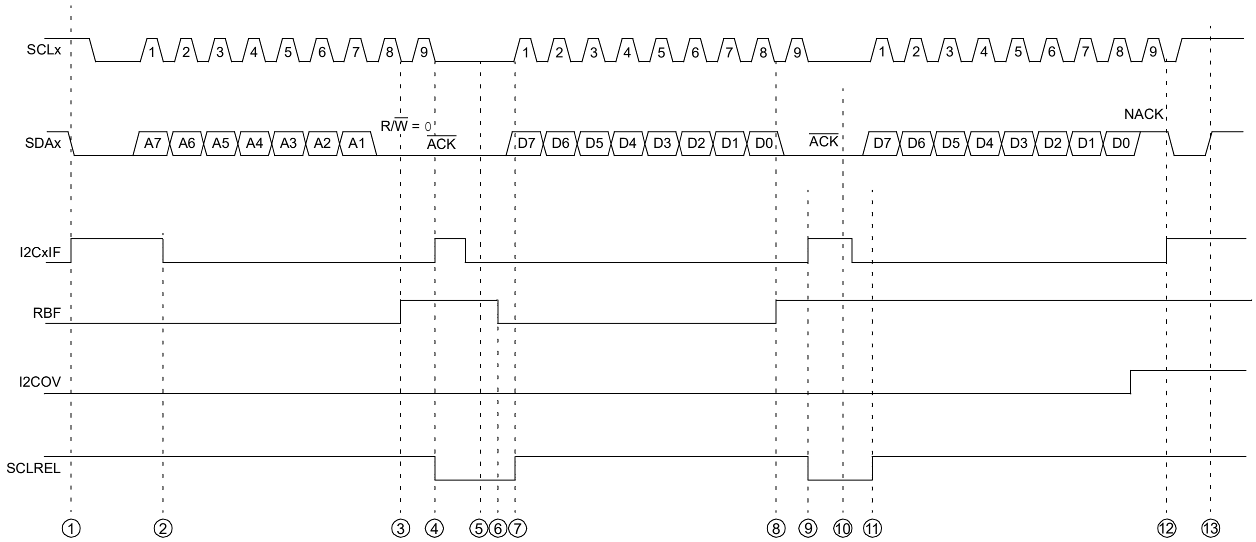
Note: The I2CxIF interrupt is still set after the ninth falling edge of the SCLx clock, even if there is no clock stretching and the RBF bit has been cleared. The I2CxIF interrupt is not asserted if a NACK is sent to the host.

Figure 23-24. I^{2}C Client, 7-Bit Address, Reception (STREN = 0, AHEN = 1, DHEN = 1)



1. Detecting Start bit enable address detection, interrupt flag is set if SCEN is set.
2. User software clears the interrupt flag.
3. Client receives the address byte with R/$\overline{\text{W}}$ = `0`. Hardware clears SCLREL. Interrupt flag is asserted. ACKTIM is asserted. I2CxRCV is loaded with I2CxRSR and RBF is asserted.
4. User software clears the interrupt flag.
5. User software reads I2CxRCV, that clears the RBF flag.
6. ACKDT is written with $\overline{\text{ACK}}$ by user software.
7. User software sets SCLREL bit to release clock, ACKTIM is cleared by hardware.
8. Interrupt flag is set (not set if NACK is received).
9. User software clears the interrupt flag.
10. If DHEN = `1`, hardware clears the SCLREL bit. I2CxRCV is loaded with I2CxRSR; ACKTIM is asserted at the end of eighth falling edge of SCLx by hardware.
11. User software reads I2CxRCV; clears the RBF flag.
12. User software releases the SCLREL bit, ACKTIM is cleared by hardware.
13. Interrupt flag is set.
14. User software clears the interrupt flag.
15. NACK.
16. Client recognizes the Stop event.

Figure 23-25. I²C Client, 7-Bit Address, Reception (STREN = 1, AHEN = 0, DHEN = 0)



① Detecting Start bit, enables address detection, interrupt is set if SCEN is set.
② User software clears the interrupt flag.
③ RBF is set on the 8th falling clock, address is loaded into I2CxRCV. RBF is asserted.
④ Interrupt is asserted.
⑤ SCLx is stretched low until SCLREL is set.
⑥ User software reads the I2CxRCV buffer, that clears the RBF flag.
⑦ User software releases the SCLx line by writing SCLREL to '1'.
⑧ Data is loaded into I2CxRCV. RBR flag is asserted.
⑨ On the 9th falling clock edge, interrupt is asserted.
⑩ SCLx is stretched and held low until SCLREL is set.
⑪ User software releases SCLx line by writing SCLREL to '1'.
⑫ NACK is received (SCLx is not stretched to low).
⑬ Client recognizes the Stop event.

23.5.4.3.5. 7-Bit Address and Client Read

When a client read is specified by having R/W = 1 in a 7-bit address byte, the process of detecting the device address is similar to that of a client write, as illustrated in Figure 23-26. If the addresses match, the following events occur:

- An $\overline{\text{ACK}}$ is generated if the AHEN bit is clear.
- The D/$\overline{\text{A}}$ status bit is cleared, and the R/W status bit is set.
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE(I2CxINTC[12]) are enabled.

Because the client is expected to reply with data at this point, it is necessary to suspend the operation of the I²C bus to allow the user software to prepare a response. This is done automatically when the module clears the SCLREL bit. With SCLREL low, the client will pull down the SCLx clock line, causing a Wait on the I²C bus. The SSPND (I2CxSTAT2[31]) bit will be set by the hardware to indicate clock stretching. Packet size PSZ(I2CxCON2[15:0]) and ND/$\overline{\text{A}}$ needs to be configured and then the user software writes the I2CxTRN register with the response data. SSPND(I2CxSTAT2[31]) bit will be automatically cleared by hardware. If Smart mode SMEN(I2CxCON2[17]) is disabled, the user must set SCLREL to release the clock or the hardware will automatically set SCLREL to release the clock.

Data setup time (TSU:DAT) can be configured by writing the required setup time to SDASUT(I2CXSDASUT[15:0]) and then enabling SDASUTEN(I2CXSDASUT[31]). In SMART mode SMEN(I2CxCON2 [17]), hardware will automatically set SCLREL after data setup time.

Once packet size becomes zero, the end of packet (EOP) will be set (I2CxSTAT2[24]).

Note: For more information on the AHEN and DHEN bits, refer to section 7-Bit Address and Client Write with the AHEN and DHEN Bits.
The SCLREL bit will automatically clear after detecting the client read address, irrespective of the state of the STREN bit.

Figure 23-26. Client Read 7-Bit Address Detection Timing Diagram (AHEN = 0)

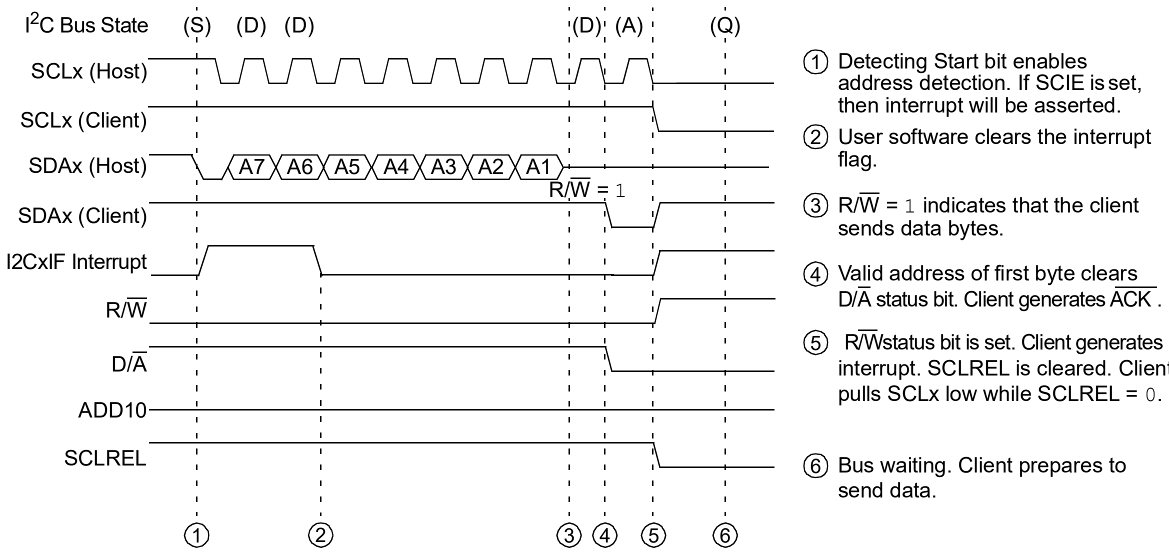
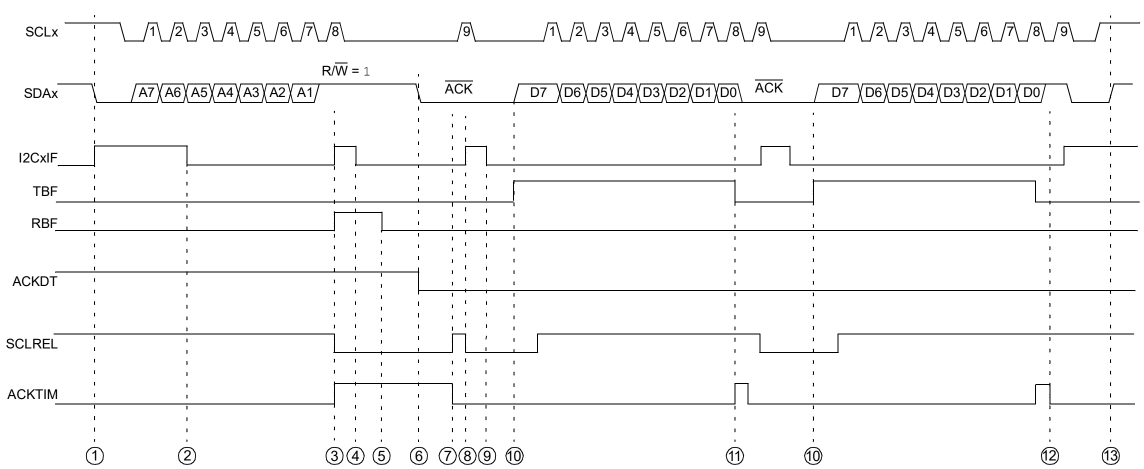


Figure 23-27. I²C Client, 7-Bit Address, Transmission (AHEN = 1)



1. Detecting Start bit enables address detection, interrupt is set if the SCIE bit is set.
2. User software clears the interrupt flag.
3. Client receives the address byte with R/W = 1. Hardware clears SCLREL to suspend host clock. ACKTIM and interrupt flag are asserted.
4. User software clears the interrupt flag.
5. Software reads the I2CxRV register, that clears the RBF flag.
6. ACKDT is written with ACK.
7. User software sets SCLREL to release clock hold. Host clocks in the Acknowledgment sequence. ACKTIM is cleared by hardware.
8. Hardware clears SCLREL to suspend host clock if R/W = 1.
9. User software clears the interrupt flag.
10. User software loads the I2CxTRN register with response data. TBF = 1 indicates that the buffer is full.
11. After last bit, module clears TBF bit, indicating buffer is available for next byte.
12. At the end of ninth clock, if host sent NACK, no more data are expected. Module does not suspend the clock.
13. Module recognizes Stop event.

23.5.4.3.6. 10-Bit Addressing Mode

In 10-Bit Addressing mode, the client must receive two device address bytes, as illustrated in Figure 23-28. The five MSbs of the first address byte specify a 10-bit address. The R/W status bit of the address must specify a write, causing the client device to receive the second address byte. For a 10-bit address, the first byte would equal, '`11110 A9 A8 0`', where `A9` and `A8` are the two MSbs of the address.

The I2CxMSK register can mask any bit position in a 10-bit address. The two MSbs of the I2CxMSK register are used to mask the MSbs of the incoming address received in the first byte. The remaining byte of the register is then used to mask the lower byte of the address received in the second byte.

Following the Start condition, the module shifts eight bits into the I2CxRSR register. The value of the I2CxRSR[2:1] bits is evaluated against the value of the I2CxADD[9:8] and I2CxMSK[9:8] bits, while the value of the I2CxRSR[7:3] bits is compared to '`11110`'. Address evaluation occurs on the falling edge of the eighth SCLx clock. For the address to be valid, the I2CxRSR[7:3] bits must be equal to '`11110`', while the I2CxRSR[2:1] bits must exactly match any unmasked bits in the I2CxADD[9:8] bits (if both bits are masked, a match is not needed). If the address is valid, the following events occur:

- An $\overline{\text{ACK}}$ is generated.
- The D/$\overline{\text{A}}$ and R/$\overline{\text{W}}$ status bits are cleared.
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE(I2CxINTC[12]) bit are enabled.

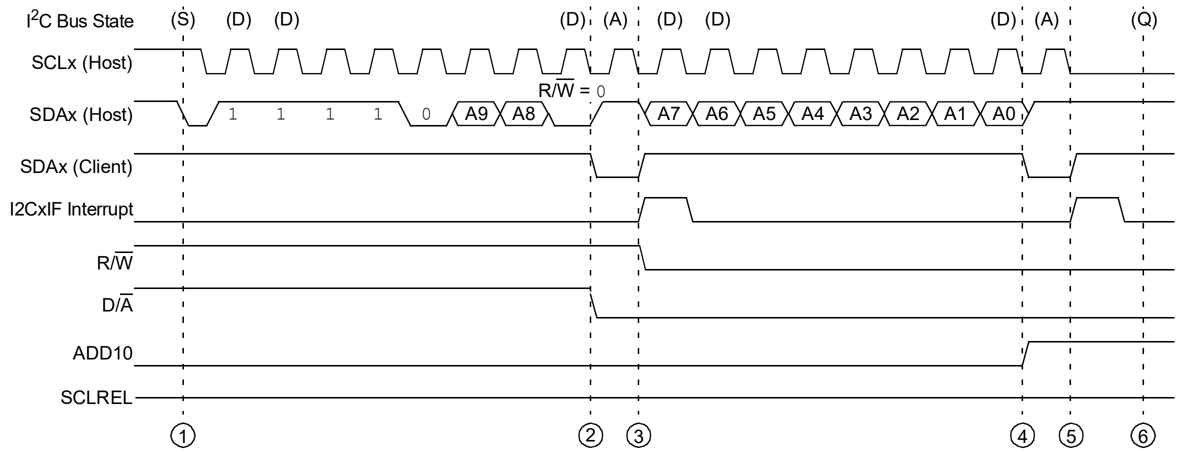
The module does generate an interrupt after the reception of the first byte of a 10-bit address; however, this interrupt is of little use.

The module will continue to receive the second byte into the I2CxRSR register. This time, the I2CxRSR[7:0] bits are evaluated against the I2CxADD[7:0] and I2CxMSK[7:0] bits. If the lower byte of the address is valid, as previously described, the following events occur:

- An $\overline{\text{ACK}}$ is generated.
- The ADD10 status bit is set.
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE (I2CxINTC[12]) are enabled.
- The module will wait for the host to send data or initiate a Repeated Start condition.

  Note: Following a Repeated Start condition in a 10-Bit Addressing mode, the client only matches the first 7-bit address, '`11110 A9 A8 0`'.

Figure 23-28. 10-Bit Address Detection Timing Diagram (AHEN = 0)



① Detecting the Start bit enables address detection.

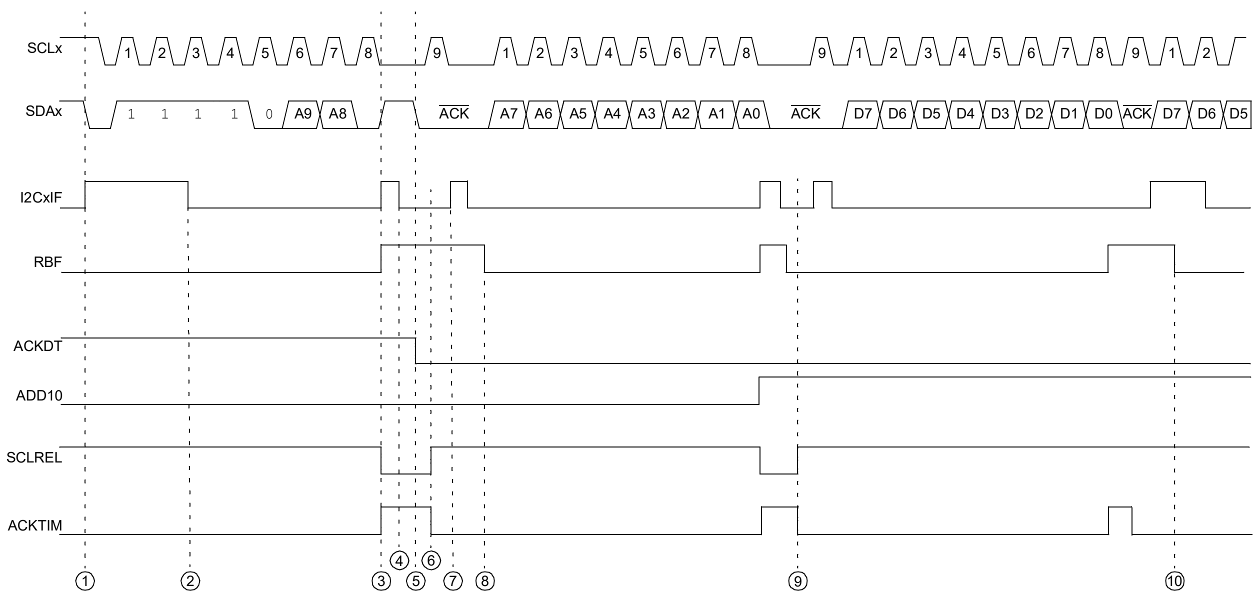
② Address match of first byte clears the D/A status bit and causes client logic to generate an ACK.

③ Reception of first byte clears the R/W status bit. Client logic generates an interrupt.

④ Address match of first and second byte sets the ADD10 status bit and causes client logic to generate an ACK.

⑤ Reception of second byte completes the 10-bit address. Client logic generates an interrupt.

Figure 23-29. I^2C Client, 10-Bit Address, Reception (STREN = 0, AHEN = 1, DHEN = 0)



① Detecting the Start bit enables address detection; interrupt is set if the SCEN bit is set.

② User software clears the interrupt flag.

③ Client receives first address byte. Write indicated. Interrupt flag is asserted. ACKTIM is asserted. If AHEN = 1, client suspends clock. SCLREL is cleared by hardware.

④ User software clears the interrupt flag.

⑤ ACKDT is written with ACK by user software.

⑥ User software sets SCLREL to release clock hold.

⑦ Client interrupt is asserted.

⑧ User software reads I2CxRCV buffer, that clears RBF flag.

⑨ Client Acknowledges the second address byte.

⑩ User software reads data from the I2CxRCV register.

23.5.4.3.7. Client Mode Bus Collision

On a read request from the host, the client begins shifting data out on the SDAx line. If a bus collision is detected and the SBCDE bit (I2CxCON1[18]) is set, then the I2CxBCIF bit will be set. After detecting the bus collision, the client goes into Idle mode and waits to be addressed again. User software can use the I2CxBCIF bit or vectors to the bus collision interrupt to handle a client bus collision.

23.5.4.3.8. General Call Operation

The addressing procedure for the I^{2}C bus is such that the first byte after a Start condition usually determines which client device the host is addressing. The exception is the general call address, which can address all devices. When this address is used, all the enabled devices respond with an Acknowledge. The general call address is one of the eight addresses reserved for specific purposes by the I^{2}C protocol. It consists of all '0's with R/W = 0. The general call is always a client write operation.

The general call address is recognized when the General Call Enable bit, GCEN (I2CxCON1[7], is set, as illustrated in Figure 23-30. Following a Start bit detect, eight bits are shifted into the I2CxRSR register, and the address is compared against the I2CxADD register and the general call address.

If the general call address matches, the following events occur:

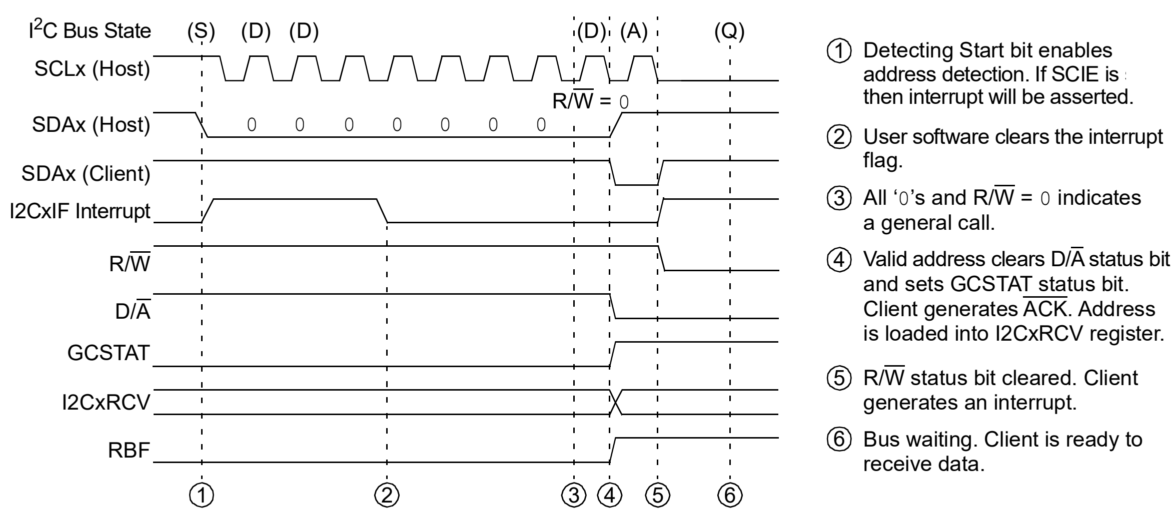
- An $\overline{\text{ACK}}$ is generated.
- The client will set the GCSTAT status bit (I2CxSTAT1[9]).
- The D/$\overline{\text{A}}$ and R/$\overline{\text{W}}$ status bits are cleared.
- The module generates the I2CxIF interrupt on the falling edge of the ninth SCLx clock if the CADDRIE (I2CxINTC[10]) bit and CSTIE(I2CxINTC[12]) bit are enabled.
- The I2CxRSR register is transferred to the I2CxRCV register, and the RBF status bit (I2CxSTAT[1]) is set (during the eighth bit).
- The module waits for the host to send data.

When the interrupt is serviced, the cause for the interrupt can be checked by reading the contents of the GCSTAT status bit to determine if the device address was device-specific or a general call address.

Notes:

1. General call addresses are 7-bit addresses. If configuring the client for 10-bit addresses and the A10M and GCEN bits are set, the client will continue to detect the 7-bit general call address.
2. The client will Acknowledge the general call address (7-bit address, 0x00) only if GCEN is set and independent of the STRICT and A10M bits.

Figure 23-30. General Call Address Detection Timing Diagram (GCEN = 1, AHEN =0)



23.5.4.3.9. Strict Support

The client module Acknowledges all the addresses, including the reserved addresses, when STRICT reserved addressing is not enforced (STRICT = 0). The client device does not Acknowledge the reserved address space if the STRICT bit (I2CxCON1[11]) is set.

Table 23-4. Client Response to Reserved Addresses

STRICT Bit	I2CxADD Client Address	Received Address into I2CxRSR	Client Acknowledge
x	0x1F	0x1F	$\overline{ACK}$
1	0x1F	C-Bus Address	NACK
1	C-Bus Address	C-Bus Address	NACK
0	C-Bus Address	C-Bus Address	$\overline{ACK}$
0	C-Bus Address	0x1F	NACK
0	0x1F	C-Bus Address	NACK

Note: When the STRICT bit is cleared, the $\overline{ACK}$ signal is generated only if the address is matched, even for reserved addresses. The client device does not generate an $\overline{ACK}$ if there is an address mismatch, even if the address is a reserved address. Irrespective of the STRICT bit setting and whether the address is reserved or not, an $\overline{ACK}$ signal is generated for a proper address match.

23.5.4.3.10. When an Address is Invalid

If a 7-bit address does not match the contents of the I2CxADD[6:0] bits, the client will return to an Idle state and ignore any activity on the I^{2}C bus until after the Stop condition.

If the first byte of a 10-bit address does not match the contents of the I2CxADD[9:8] bits, the client will return to an Idle state and ignore all bus activity until after the Stop condition.

If the first byte of a 10-bit address matches the contents of the I2CxADD[9:8] bits, but the second byte of the 10-bit address does not match the I2CxADD[7:0] bits, the client will return to an Idle state and ignore all bus activity until after the Stop condition.

23.5.4.3.11. Addresses Reserved from Masking

Even when enabled, there are several addresses that are ignored by the I^{2}C module. For these addresses, an Acknowledge will not be issued independent of the mask setting. These addresses are listed in Table 23-5.

Table 23-5. Reserved I^{2}C Bus Addresses(3)

7-Bit Address Mode		
Client Address	**R/W Bit**	**Description**
0000 000	0	General Call Address(1)
0000 000	1	Start Byte
0000 001	x	C-Bus Address
0000 010	x	Reserved
0000 011	x	Reserved
0000 1xx	x	HS Mode Master Code
1111 1xx	x	Reserved
1111 0xx	x	10-Bit Client Upper Byte(2)

Notes:

1. Address will be Acknowledged only if GCEN = 1.
2. A match on this address can only occur as the upper byte in 10-Bit Addressing mode.
3. These addresses will not be Acknowledged, independent of mask settings and STRICT = 1.

23.5.4.4. Receiving Data from a Host Device

When the R/$\overline{W}$ status bit of the device address byte is '0' and an address match occurs, the R/$\overline{W}$ status bit (I2CxSTAT1[2]) is cleared. The client enters a state waiting for data to be sent by the host. After the device address byte, the contents of the data byte are defined by the system protocol and are only received by the client. User software should configure data packet size in PSZ (I2CxCON2).

The client shifts eight bits into the I2CxRSR register. On the falling edge of the eighth clock (SCLx), the following events occur:

- The module begins to generate an $\overline{ACK}$ or NACK.
- The RBF status bit (I2CxSTAT1[1]) is set to indicate received data.
- The I2CxRSR register byte is transferred to the I2CxRCV register for access by the user software.
- The D/$\overline{A}$ status bit is set.
- A client interrupt is generated. User software can check the status of the I2CxSTAT register to determine the cause of the event and then clear the I2CxIF interrupt flag if the CDRXIE (I2CxINTC[4]) bit and CSTIE (I2CxINTC[12]) bit are enabled.
- The module waits for the next data byte.
- Once packet size becomes zero, end of packet (EOP) will be set (I2CxSTAT2[24]).

23.5.4.4.1. Acknowledge Generatrion

Normally, the client Acknowledges all the received bytes by sending an $\overline{ACK}$ on the ninth SCLx clock. If the receive buffer is overrun, the client does not generate this $\overline{ACK}$. The overrun is indicated if either (or both) of the following occur:

- The Receive Buffer Full bit, RBF (I2CxSTAT1[1]), was set before the transfer was received.
- The Receive Overflow bit, I2COV (I2CxSTAT1[6]), was set before the transfer was received.

Table 23-6 shows what happens when a data transfer byte is received, given the status of the RBF and I2COV status bits. If the RBF status bit is already set when the client attempts to transfer to the I2CxRCV register, the transfer does not occur, but the interrupt is generated and the I2COV status bit is set. If both the RBF and I2COV status bits are set, the client acts similarly. The shaded cells show the condition where user software did not properly clear the overflow condition.

Reading the I2CxRCV register clears the RBF status bit. The I2COV status bit is cleared by writing to a '0' through user software.

Note: If the BOEN bit (I2CxCON1[20]) is set, then the I2COV bit (I2CxSTAT1[6]) is ignored and only the RBF bit (I2CxSTAT1[1]) determines whether the module will Acknowledge the message or not.

23.5.4.4.2. Receive Buffer Overwrite (I^{2}C Client Mode Only)

If the BOEN bit (I2CxCON1[20]) is set, then the I2COV bit (I2CxSTAT1[6]) is ignored. If the RBF bit (I2CxSTAT1[1]) is not set, then the $\overline{ACK}$ is generated for the receive address or data; the I2CxRCV buffer is updated with I2CxRSR.

Table 23-6. Data Transfer Received Byte Actions

Status Bits as Data Byte Received			Transfer I2CxRSR to I2CxRCV	Generate ACK	Generate I2CxIF Interrupt (interrupt occurs if enabled)	Set RBF	Set I2COV
BOEN(1)	RBF	I2COV					
x	0	0	Yes	Yes	Yes	Yes	No change
x	1	0	No	No	Yes	No change	Yes
x	1	1	No	No	Yes	No change	Yes
0	0	1	Yes	No	Yes	Yes	No change
1	0	1	Yes	Yes	Yes	Yes	No change

23.5.4.4.3. Wait States During Client Receptions

When the client receives a data byte, the host can potentially begin sending the next byte immediately. This allows the user software controlling the nine client SCLx clock periods to process the previously received byte. If this is not enough time, the client software may want to generate a bus Wait period.

The STREN bit (I2CxCON1[6]) enables a bus Wait to occur on client receptions. When STREN = `1` at the falling edge of the ninth SCLx clock of a received byte, the client clears the SCLREL bit. Clearing the SCLREL bit causes the client to pull the SCLx line low, initiating a Wait. The SCLx clock of the host and client will synchronize, as provided in Host Clock Synchronization.

When the user software is ready to resume reception, the user software sets the SCLREL bit. This causes the client to release the SCLx line, and the host resumes clocking. If SMART mode (SMEN, I2CxCON2[17]) is enabled, hardware will automatically release SCL by setting SCLREL when the received byte is read (RBF=`0`) depending on EOP (I2CxSTAT2[24]) and EPSZE (I2CxCON2[24]).

23.5.4.4.4. Examples Messages of Client Reception

Receiving a client message is an automatic process. The user software handling the client protocol uses the client interrupt to synchronize to the events.

When the client detects the valid address, the associated interrupt will notify the user software to expect a message. Upon receiving the data, as each data byte transfers to the I2CxRCV register, an interrupt notifies the user software to unload the buffer.

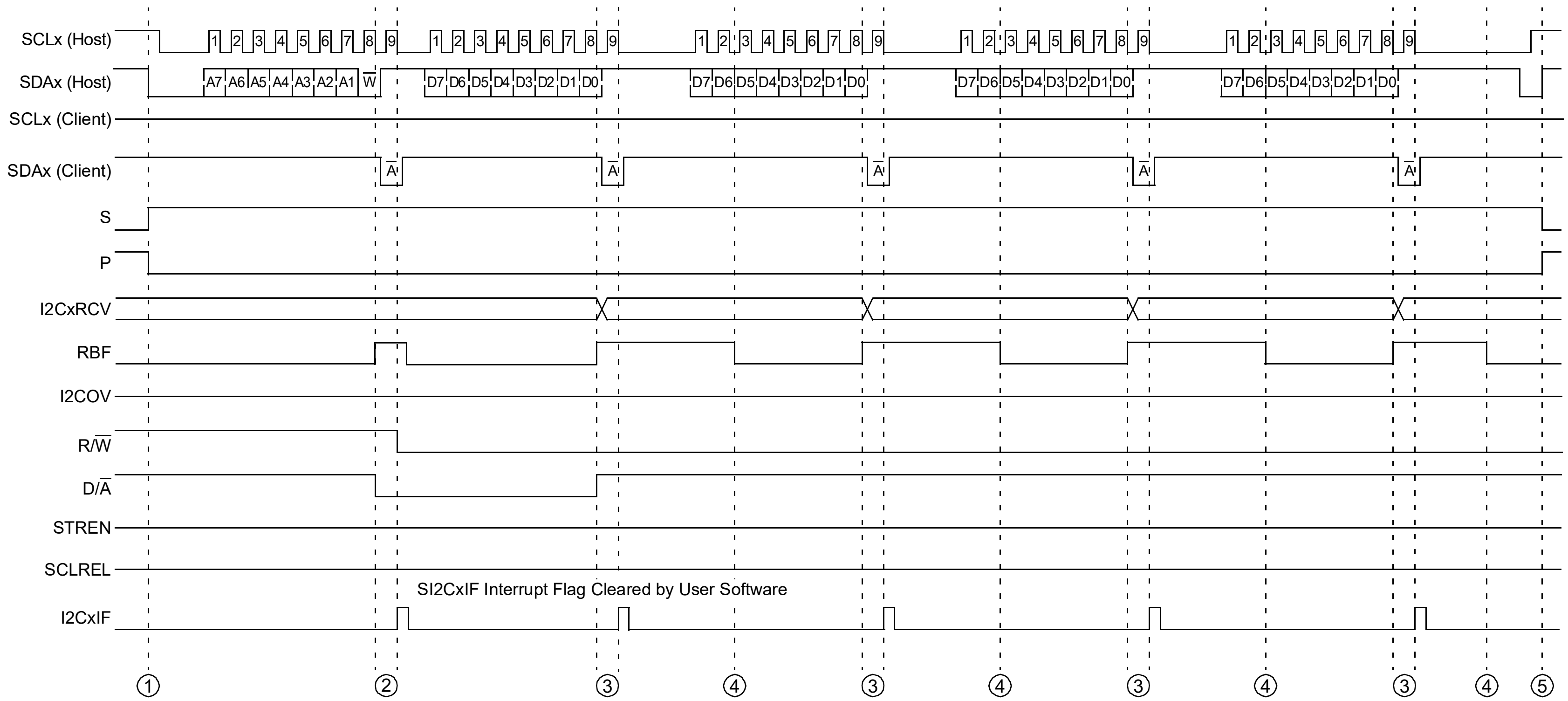
Figure 23-31 illustrates a simple receive message. Because it is a 7-bit address message, only one interrupt occurs for the address bytes. Then, interrupts occur for each of four data bytes. At an interrupt, the user software may monitor the status bits, RBF (I2CxSTAT1[1]), D/$\overline{\text{A}}$ (I2CxSTAT1[5]) and R/$\overline{\text{W}}$ (IC2xSTAT1[2]), to determine the condition of the byte received.

Figure 23-32 illustrates a similar message using a 10-bit address. In this case, two bytes are required for the address.

Figure 23-33 illustrates a case where the user software does not respond to the received byte and the buffer overruns. On reception of the second byte, the module will automatically NACK the host transmission. Generally, this causes the host to resend the previous byte. The I2COV status bit (I2CxSTAT1[6]) indicates that the buffer has overrun. The I2CxRCV register buffer retains the contents of the first byte. On reception of the third byte, the buffer is still full, and again, the module will NACK the host. After this, the user software finally reads the buffer. Reading the buffer will clear the RBF status bit; however, the I2COV status bit remains set. The user software must clear the I2COV status bit (I2CxSTAT1[6]). The next received byte is moved to the I2CxRCV register buffer and the module responds with an $\overline{\text{ACK}}$.

Figure 23-34 highlights clock stretching while receiving data. In the previous examples, the STREN bit (I2CxCON1[6]) is equal to '`0`', which disables clock stretching on receive messages. In this example, the user software sets STREN to enable clock stretching. When STREN = `1`, the module will automatically clock stretch after each received data byte, allowing the user software more time to move the data from the buffer. If RBF = `1` at the falling edge of the ninth clock, the module automatically clears the SCLREL bit (I2CxCON1[12]) and pulls the SCLx bus line low. As shown with the second received data byte, if the user software can read the buffer and clear the RBF status bit before the falling edge of the ninth clock, the clock stretching will not occur. The user software can also suspend the bus at any time. By clearing the SCLREL bit, the module pulls the SCLx line low after it detects the bus SCLx low. The SCLx line remains low, suspending transactions on the bus until the SCLREL bit is set. See Client Reception (7-bit Address) and Client Transmission (10-bit Address) for application examples.

Figure 23-31. Client Message (Write Data to Client: 7-Bit Address; Address Matches; A10M = 0, GCEN = 0, IPMIEN = 0, AHEN = 0, DHEN = 0, STRICT = 0 and BOEN = 0)



① Client recognizes Start event; S and P status bits set/clear accordingly.

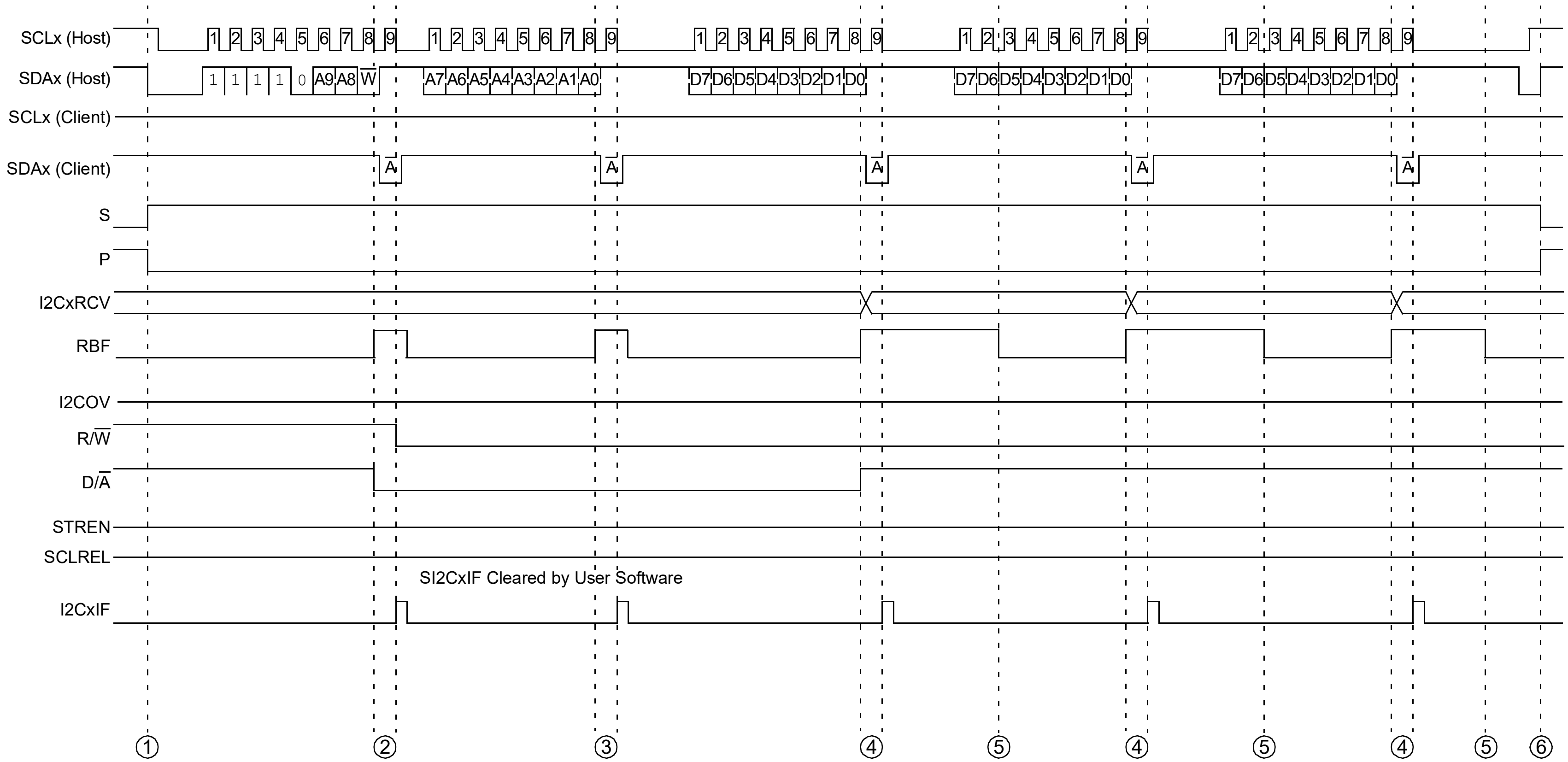
② Client receives address byte. Address matches. Client Acknowledges and generates interrupt. Address byte is moved to the I2CxRCV register and must be read by user software to prevent buffer overflow.

③ Next received byte is message data. The byte moved to the I2CxRCV register sets the RBF status bit. Client generates interrupt. Client Acknowledges reception.

④ User software reads the I2CxRCV register. RBF status bit clears.

⑤ Client recognizes Stop event; S and P status bits set/clear accordingly.

Figure 23-32. Client Message (Write Data to Client: 10-Bit Address; Address Matches; A10M = 1, GCEN = 0, IPMIEN = 0, AHEN = 0, DHEN = 0, STRICT = 0 and BOEN = 0)



① Client recognizes Start event, S and P bits set/clear accordingly.

② Client receives address byte. High-order address matches. Client Acknowledges and generates interrupt. Address byte is moved to the I2CxRCV register and is read by user software to prevent buffer overflow.

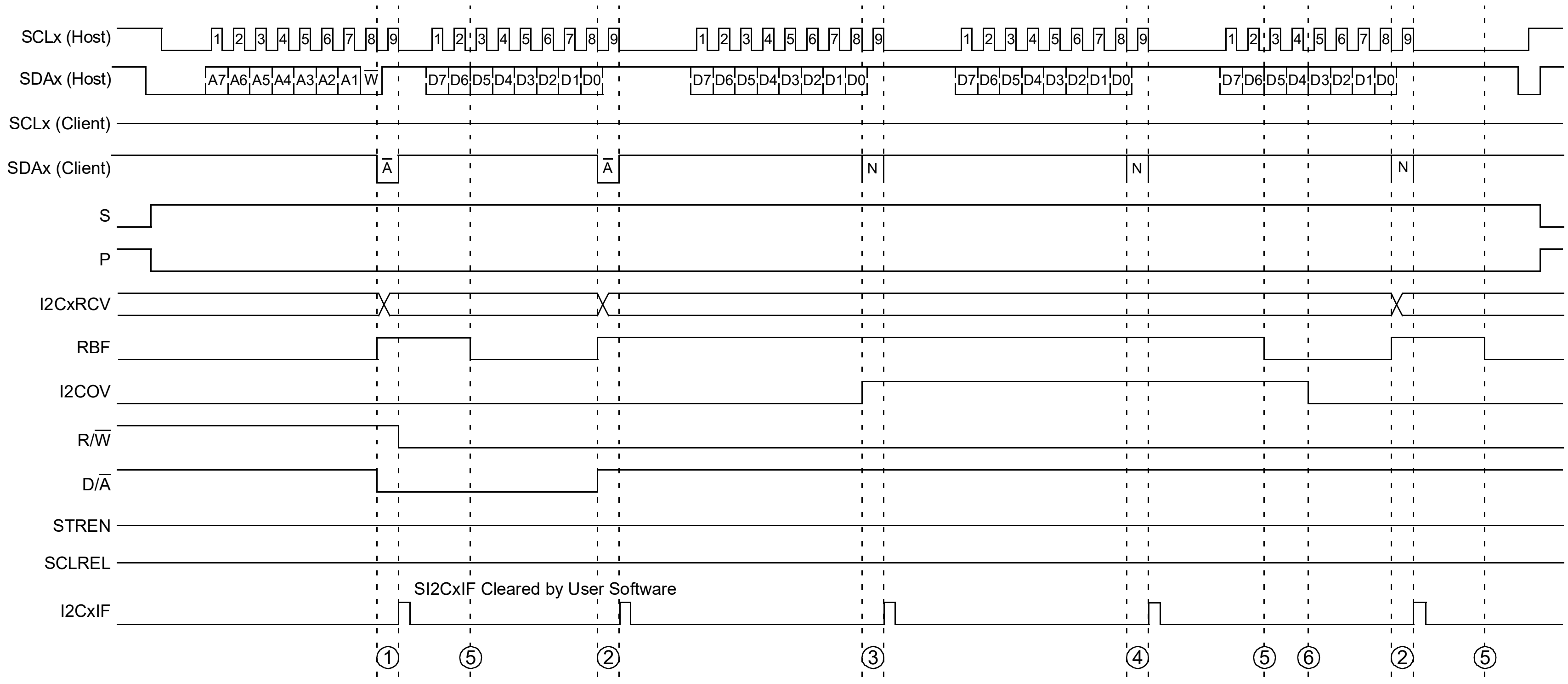
③ Client receives address byte. Low-order address matches. Client Acknowledges and generates interrupt. Address byte is moved to the I2CxRCV register and is read by user software to prevent buffer overflow.

④ Next received byte is message data. Byte moved to the I2CxRCV register, sets RBF. Client Acknowledges and generates interrupt.

⑤ User software reads the I2CxRCV register. RBF bit clears.

⑥ Client recognizes Stop event, S and P bits set/clear accordingly.

Figure 23-33. Client Message (Write Data to Client: 7-Bit Address; Buffer Overrun; A10M = 0, GCEN = 0, IPMIEN = 0, AHEN = 0, DHEN = 0, STRICT = 0 and BOEN = 0)



① Client receives address byte. Address matches. Client generates interrupt. Address byte is moved to the I2CxRCV register and must be read by user software to prevent buffer overflow.

② Next received byte is message data. The byte is moved to the I2CxRCV register, sets RBF. Client generates interrupt. Client Acknowledges reception.

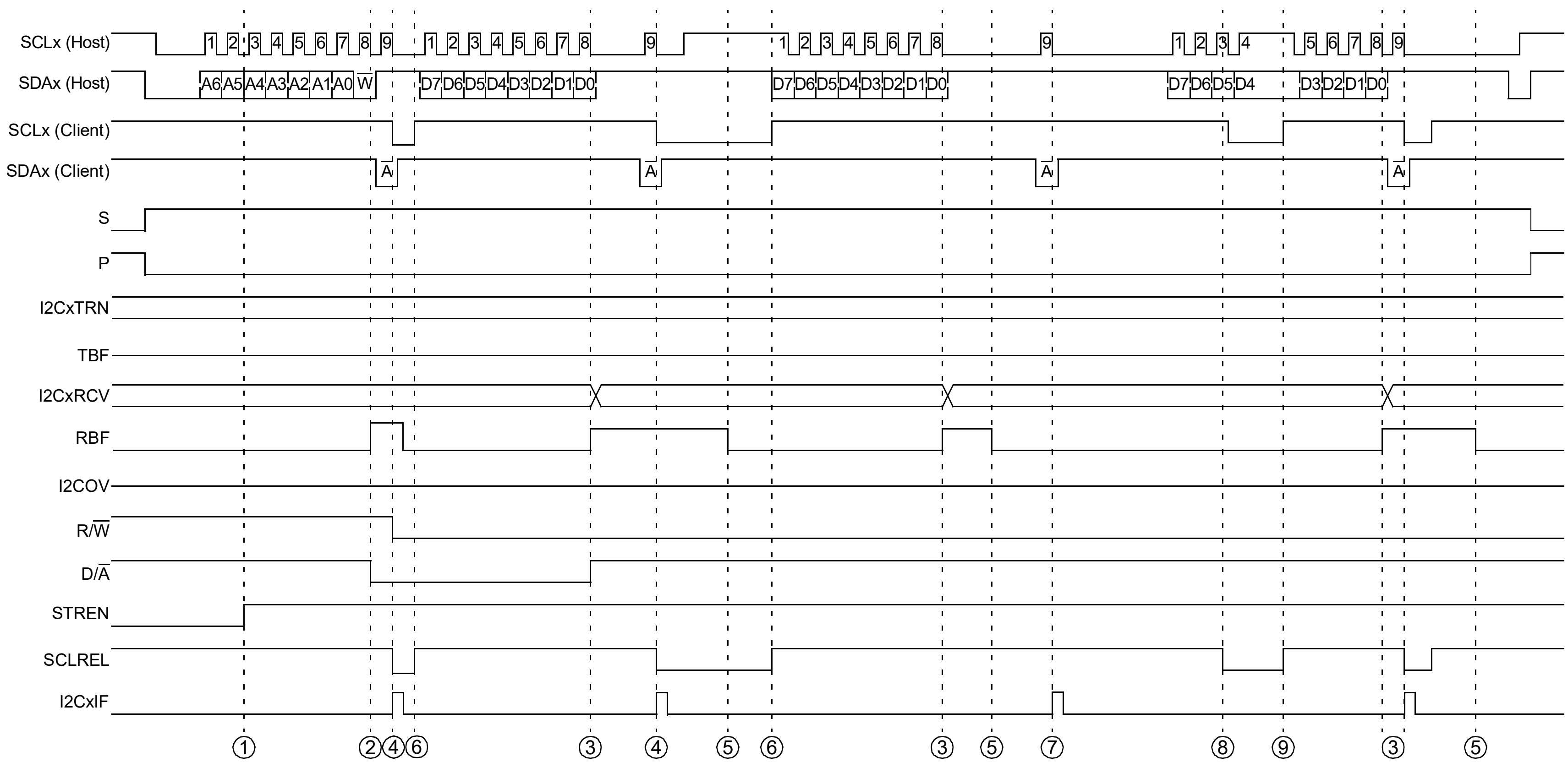
③ Next byte received before I2CxRCV is read by software. I2CxRCV register is unchanged. I2COV overflow bit is set. Client generates interrupt. Client sends NACK for reception.

④ Next byte also received before I2CxRCV is read by software. I2CxRCV register is unchanged. Client generates interrupt. Client sends NACK for reception. The host state machine should not be programmed to send another byte after receiving a NACK in this manner. Instead, it should abort the transmission with a Stop condition or send a Repeated Start condition and attempt to retransmit the data.

⑤ User software reads the I2CxRCV register. RBF bit clears.

⑥ User software clears the I2COV bit. Reception will still not be able to proceed normally until the module sees a Stop/Repeated Start bit. If neither of these conditions is met, an additional transmission will be received correctly, but sends a NACK and sets the I2COV bit again.

Figure 23-34. Client Message (Write Data to Client: 7-Bit Address; Clock Stretching Enabled; A10M = 0, GCEN = 0, IPMIEN = 0, AHEN = 0, DHEN = 0, STRICT = 0 and BOEN = 0)



① User software sets the STREN bit to enable clock stretching.

② Client receives address byte. I2CxRCV register is read by user software to prevent buffer overflow.

③ Next received byte is message data. The byte moves to the I2CxRCV register, sets RBF.

④ Because RBF = 1 at ninth clock, automatic clock stretch begins. Client clears SCLREL bit. Client pulls SCLx line low to stretch clock.

⑤ User software reads I2CxRCV register. RBF bit clears.

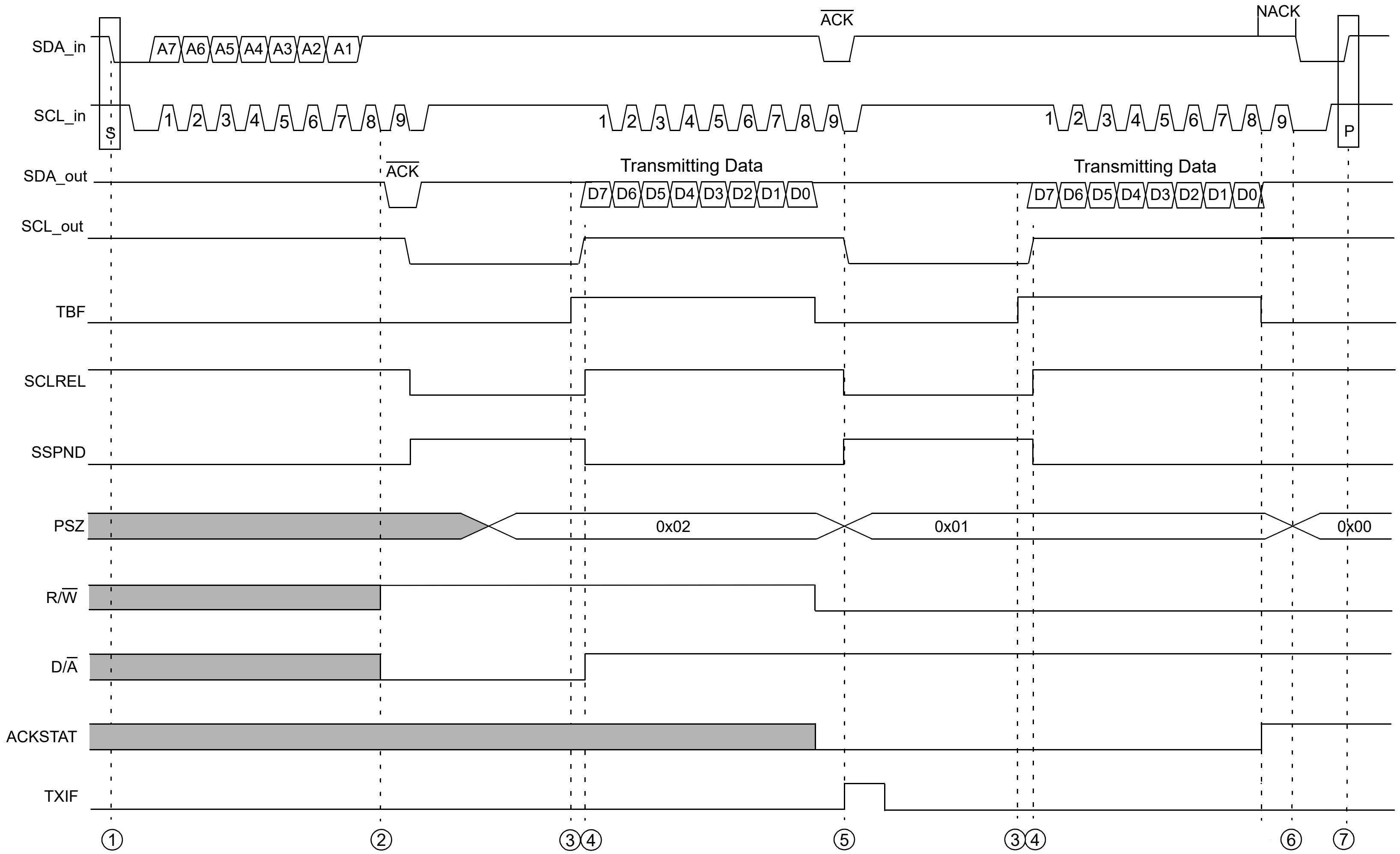
⑥ User software sets SCLREL bit to release clock.

⑦ Client does not clear SCLREL because RBF = 0 at this time.

⑧ User software may clear SCLREL to cause a clock hold. Module must detect SCLx low before asserting SCLx low.

⑨ User software may set SCLREL to release a clock hold.

Figure 23-35. I[2]C Client 7-Bit Address Transmission with Smart Mode Enabled (SMEN = 1)



① Client recognizes Start event, S and P bits set/clear accordingly.

② Client receives address byte. Address matches. Address byte is moved to I2CxRCV register and is read by user software to prevent buffer overflow. R/W = 1 to indicate read from Client.

③ User software writes I2CxTRN with response data. TBF = 1 indicates that buffer is full. Writing I2CxTRN sets D/A, indicating a data byte.

④ SCLREL will be set by hardware automatically to release the clock. SCLREL will be released after data setup if configured. SSPND will be cleared to indicate clock release.

⑤ I2CxTXIF will be set after transmitting data.

⑥ EOP will be set.

⑦ Client recognizes Stop event; S and P bits set/clear accordingly.

23.5.4.5. Sending Data to a Host Device

When the R/W status bit of the incoming device address byte is '1' and an address match occurs, the R/W status bit (I2CxSTAT1[2]) is set. At this point, the host device is expecting the client to respond by sending a byte of data. The contents of the byte are defined by the system protocol and are only transmitted by the client.

When the interrupt from the address detection occurs, the user software can write a byte to the I2CxTRN register to start the data transmission.

The client sets the TBF status bit (I2CxSTAT1[0]). The eight data bits are shifted out on the falling edge of the SCLx input. This ensures that the SDAx signal is valid during the SCLx high time. When all eight bits have been shifted out, the TBF status bit is cleared.

The client detects the Acknowledge from the host receiver on the rising edge of the ninth SCLx clock.

If the SDAx line is low, indicating an $\overline{\text{ACK}}$, the host is expecting more data and the message is not complete. The module generates a client interrupt, and the ACKSTAT status bit (I2CxSTAT1[15]) can be inspected to determine whether more data are being requested.

A client interrupt is generated on the falling edge of the ninth SCLx clock. User software must check the status of the I2CxSTAT register and clear the I2CxIF interrupt flag if the CDTXIE (I2CxINTC[3]) bit and CSTIE(I2CxINTC[12]) are enabled.

If the SDAx line is high, indicating a NACK, the data transfer is complete. User software should not write further data to the I2CxTRN register. The client resets and generates an interrupt, and it waits for detection of the next Start bit.

23.5.4.5.1. Wait States During Client Transmissions

During a client transmission message, the host expects return data immediately after detection of the valid address with R/W = 1. Because of this, the client automatically generates a bus Wait whenever the client returns data.

The automatic Wait occurs at the falling edge of the ninth SCLx clock of a valid device address byte or transmitted byte, Acknowledged by the host, indicating expectation of more transmit data.

The client clears the SCLREL bit (I2CxCON1[12]). Clearing the SCLREL bit causes the client to pull the SCLx line low, initiating a Wait. The SCLx clock of the host and client will synchronize, as shown in Host Clock Synchronization.

When the user software loads the I2CxTRN register and is ready to resume transmission, the user software sets the SCLREL bit. This causes the client to release the SCLx line, and the host resumes clocking.

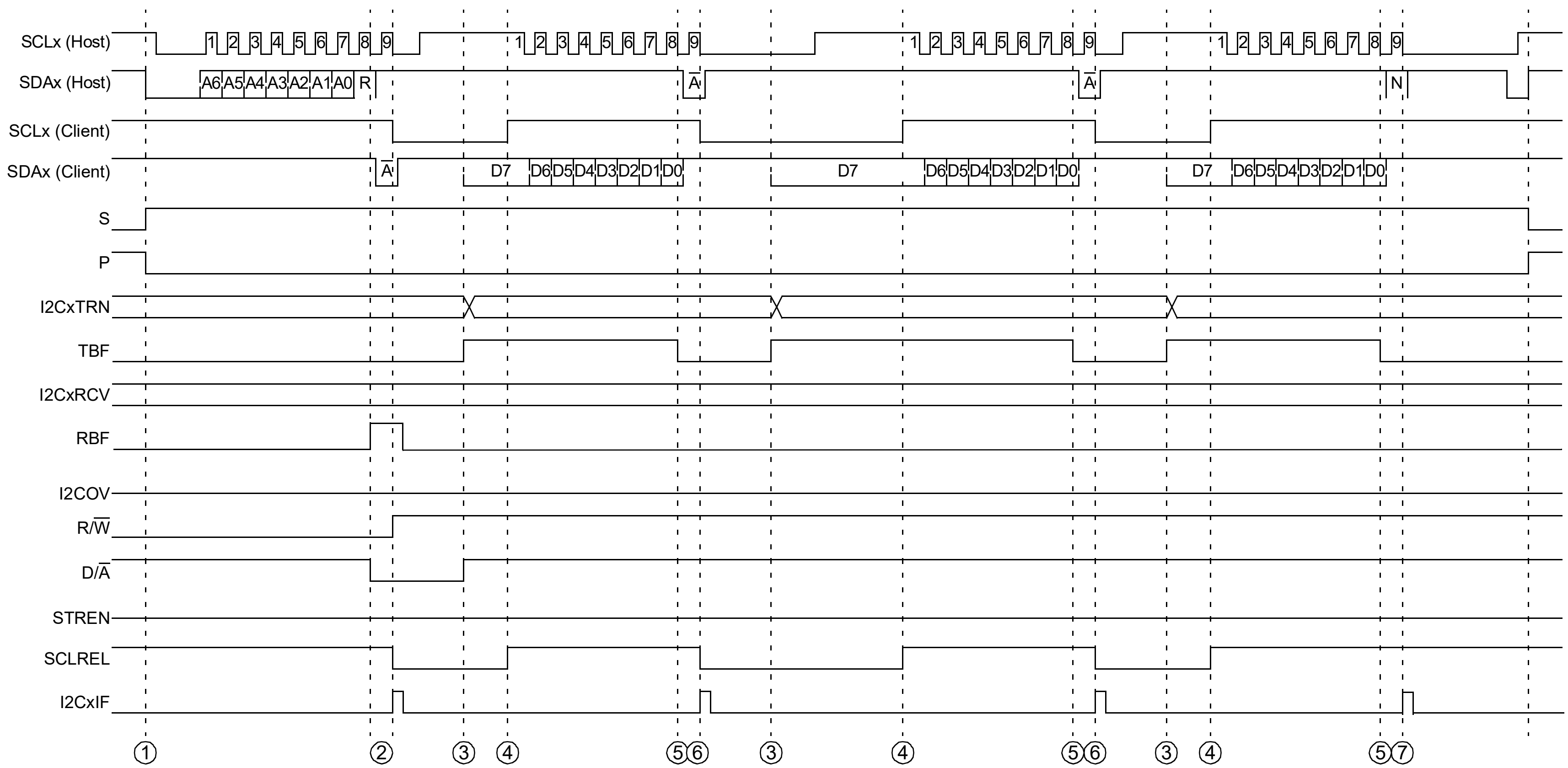
Note: The user software must provide a delay between writing to the transmit buffer and setting the SCLREL bit. This delay must be greater than the minimum setup time for client transmissions, as specified in the Electrical Characteristics section.

23.5.4.5.2. Example Messages of Client Transmission

The client transmissions for 7-bit address messages are illustrated in Figure 23-36. When the address matches and the R/W status bit of the address indicates a client transmission, the module automatically initiates clock stretching by clearing the SCLREL bit and generates an interrupt to indicate that a response byte is required. The user software writes the response byte into the I2CxTRN register. As the transmission completes, the host responds with an $\overline{\text{ACK}}$. If the host replies with an $\overline{\text{ACK}}$, the host expects more data, and the module again clears the SCLREL bit and generates another interrupt. If the host responds with a NACK, no more data are required and the module will not stretch the clock but will generate an interrupt.

The client transmissions for 10-bit address messages require the client to first recognize a 10-bit address. Because the host must send two bytes for the address, the R/$\overline{\text{W}}$ Status bit in the first byte of the address specifies a write. To change the message to a read, the host sends a Repeated Start and repeats the first byte of the address with the R/$\overline{\text{W}}$ status bit, specifying a read. At this point, the client transmission begins as illustrated in Figure 23-37. See Client Transmission (7-bit Address) and Client Transmission (10-bit Address) for application examples.

Figure 23-36. Client Message (Read Data from Client: 7-Bit Address)



① Client recognizes Start event, S and P bits set/clear accordingly.

② Client receives address byte. Address matches. Client generates interrupt. Address byte is moved to I2CxRCV register and is read by user software to prevent buffer overflow. R/W = `1` to indicate read from client. SCLREL = `0` to suspend host clock.

③ User software writes I2CxTRN with response data. TBF = `1` indicates that buffer is full. Writing I2CxTRN sets D/A, indicating a data byte.

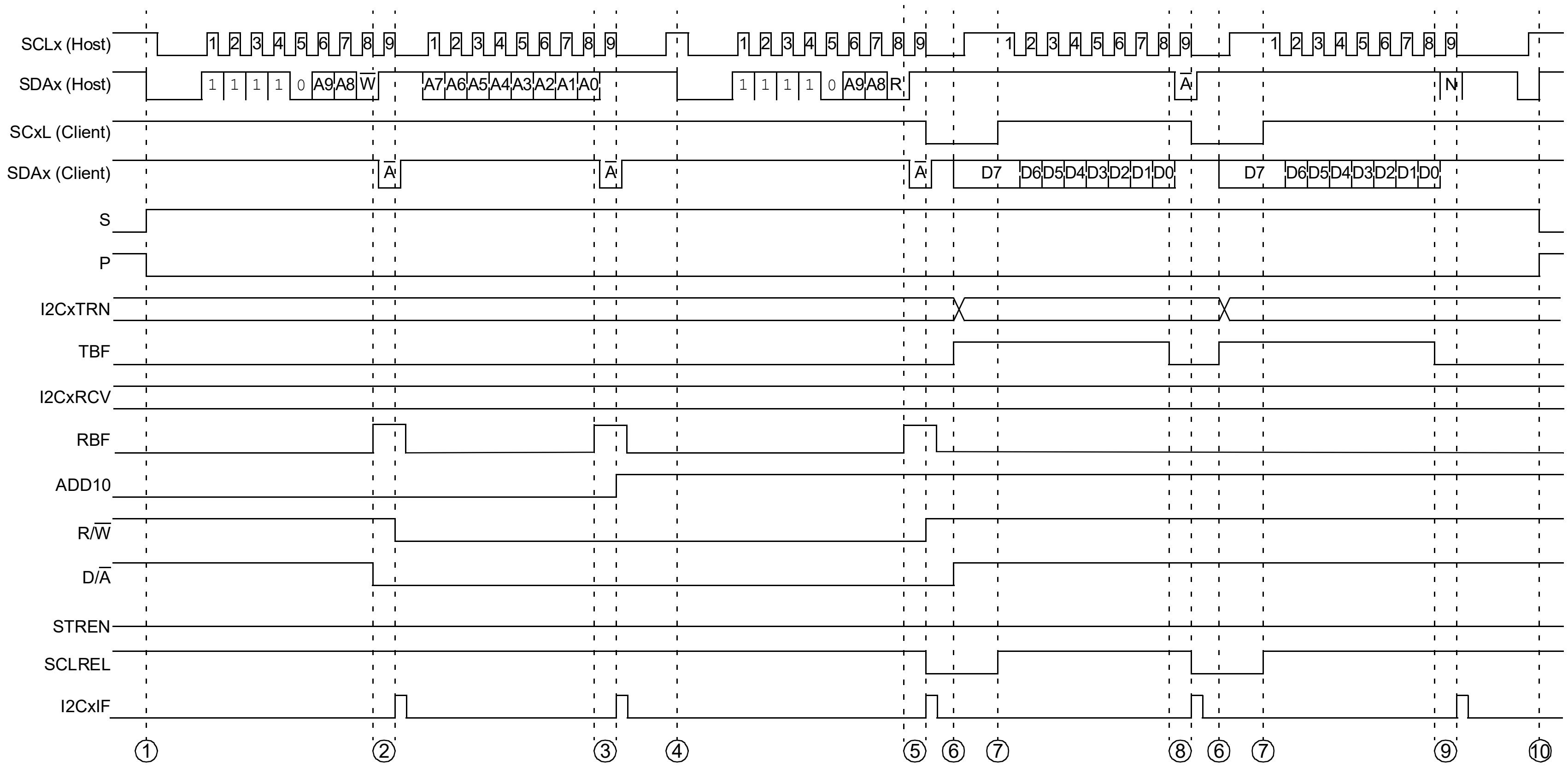
④ User software sets SCLREL to release clock hold. Host resumes clocking and client transmits data byte.

⑤ After last bit, module clears TBF bit, indicating buffer is available for next byte.

⑥ At the end of ninth clock, if the host has sent an ACK, module clears SCLREL to suspend clock. Client generates interrupt.

⑦ At the end of ninth clock, if host sent a NACK, no more data expected. User software should stop writing to I2CxTRN. Module does not suspend clock and will generate an interrupt.

Figure 23-37. Client Message (Read Data from Client: 10-Bit Address)



① Client recognizes Start event; S and P bits set/clear accordingly.

② Client receives first address byte. Write indicated. Client Acknowledges and generates interrupt. User software reads I2CxRCV register.

③ Client receives address byte. Address matches. Client Acknowledges and generates interrupt. User software reads I2CxRCV register.

④ Host sends a Repeated Start to redirect the message.

⑤ Client receives resend of first address byte. User software reads I2CxRCV register. Read indicated. Client suspends clock.

⑥ User software writes I2CxTRN with response data.

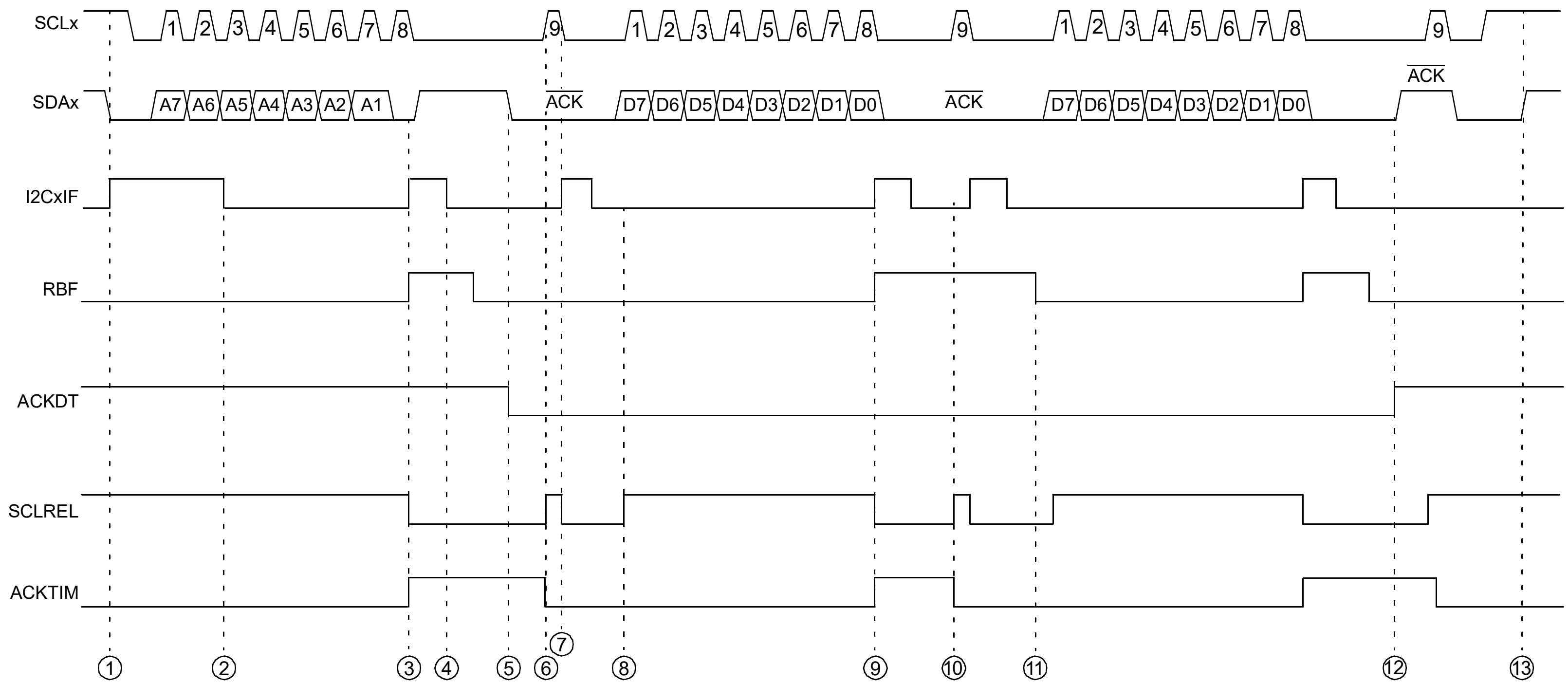
⑦ User software sets SCLREL to release clock hold. Host resumes clocking and client transmits data byte.

⑧ At the end of ninth clock, if host sent an $\overline{\text{ACK}}$, module clears SCLREL to suspend clock. Client generates interrupt.

⑨ At the end of ninth clock, if host sent a NACK, no more data expected. User software should stop writing to I2CxTRN. Module does not suspend clock and will generate an interrupt.

⑩ Client recognizes Stop event; S and P bits set/clear accordingly.

Figure 23-38. I^{2}C Client, 7-Bit Address, Reception (STREN = 1, AHEN = 1, DHEN = 1)



(1) Detecting Start bit, enables address detection, interrupt is set if SCEN is set.

(2) User software clears the interrupt flag.

(3) Client receives first address byte. Write Indicated. If AHEN = `1`, SCLREL is cleared by hardware. ACKTIM and interrupt are asserted.

(4) User software clears the interrupt flag.

(5) ACKDT is written with an $\overline{\text{ACK}}$ by user software.

(6) User software sets SCLREL to release the clock hold; ACKTIM is cleared by hardware.

(7) Hardware stretches the clock after $\overline{\text{ACK}}$ (if STREN = `1`).

(8) User software sets SCLREL to release clock hold.

(9) I2CxRCV is loaded with I2CxRSR. RBF is set. If DHEN = `1`, SCLREL is cleared by hardware and ACKTIM is asserted.

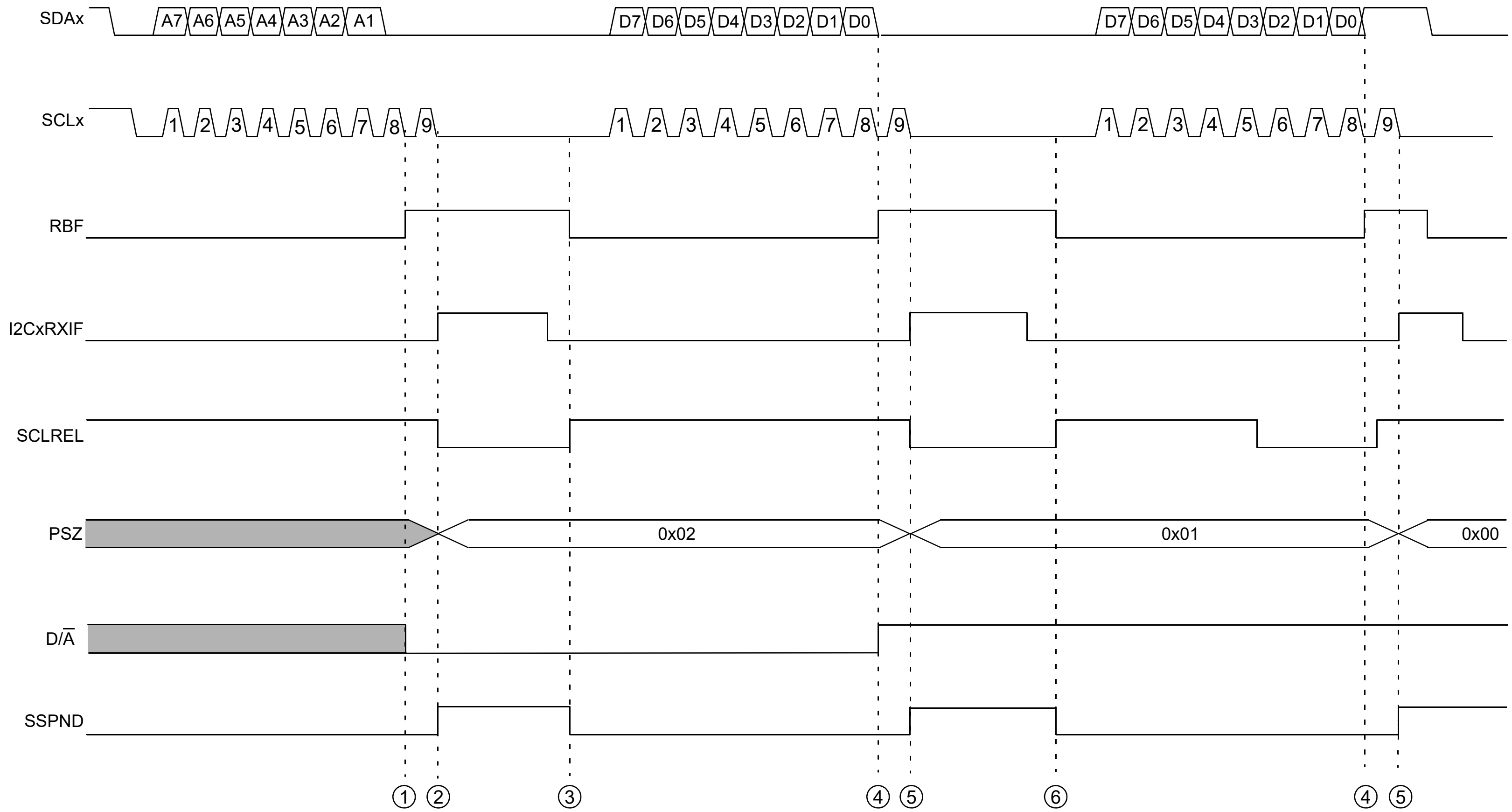
(10) User software sets SCLREL to release the clock hold. ACKTIM is cleared by hardware. After Acknowledgment, hardware stretches the clock.

(11) User software reads I2CxRCV, that clears the RBF flag.

(12) NACK sent by host.

(13) Module recognizes the Stop event.

Figure 23-39. I²C Client 7-Bit Address Reception with Smart Mode Enabled (SMEN = 1)



① Client receives address byte, RBF= 1.

② RBF = 1 at ninth clock, clock will be suspended SSPND = 1. Automatic clock stretch begins. Client clears SCLREL bit. Client pulls SCLx line low to stretch clock.

③ User software reads I2CxRCV register. RBF bit clears. Client will stop suspending clock, SSPND= 0. Client sets SCLREL bit to release the clock.

④ Client receives data byte, RBF= 1.

⑤ RBF = 1 at ninth clock, clock will be suspended SSPND = 1. Automatic clock stretch begins. Client clears SCLREL bit. Client pulls SCLx line low to stretch clock.

⑥ User software reads I2CxRCV register. RBF bit clears. Client will stop suspending clock, SSPND= 0. Client sets SCLREL bit to release the clock.

23.5.4.6. Frame Error

Frame error is used in the PMBus to detect a Fault. A frame error will detect:

- Sending Too Few Bits: If a Start or Stop condition interrupts the transmission while a device is writing to a PMBus device before a complete byte has been sent, this is a data transmission Fault.
- Reading Too Few Bits: If a Start or Stop condition interrupts the transmission while a device is reading from a PMBus device before a complete byte has been read, this is a data reception Fault. In these conditions, the FRME(I2CxSTAT2[19]) bit will be set.
- When the frame error occurs due to a Stop condition, the receiver returns to an Idle state and waits for the Start condition.
- When the frame error occurs due to a Start condition, the client is considered as a new Start condition and moves to the address receive start sequence.
- With a frame error, the number of SCL clocks will be stored in SCLCNT(I2CxSTAT2[3:0]).

23.5.5. Packet Error Checking (PEC)

The Packet Error Checking mechanism improves reliability and communication robustness. A CRC-8 packet error checking based on SMBus v3.0 specification is available. PEC is calculated on all the message bytes (including addresses and read/write bits). The PEC calculation does not include $\overline{\text{ACK}}$/NACK bits, Start/Stop or Repeated Start conditions. PEC is available in both Host and Client mode.

During client/host transmission, the PEC can be enabled or disabled to automatically append at the end of a packet using PECC (I2CxCON2[29:28]) bits.

During reception, when PECC[1:0] is `01`, the calculated PEC value will be written into I2CxPEC.CCRC at the end of the packet. Since the received packet will contain a PEC byte at the end of the packet, the calculated PEC should be `0` if the packet was received correctly. If any mismatch is in calculated PEC and received PEC, CRC(I2CxSTAT[16]) will be set.

23.5.6. Bus Time-out

Different bus time-out options are available, which are required for SMBus and PMBus protocols. There are four time-outs available: SCL low time-out, host extended time-out, client extended time-out and Bus Free/Idle time-out.

Note: The timers' values should be written by software before enabling the timers.

23.5.6.1. SCL Low Time-out ($t_{TIMEOUT}$)

SCL low time-out ($t_{TIMEOUT}$) allows a host or client to conclude that a defective device is holding the clock low indefinitely or that a host is intentionally trying to drive devices off the bus. It is highly recommended that a client device release the bus when it detects any single clock held low longer than $t_{TIMEOUT, MIN}$. Devices that have detected this condition must reset their communication interface and be able to receive a new Start condition in no later than $t_{TIMEOUT, MAX}$.

SCL low time-out can be enabled by writing the time-out value to BSCLTOTMR (I2CxBSCLTO[23:0]) and then setting the BSCLTE (I2CxCON2[27]) bit. The timer will continue to run until SCL stays low or the timer value reloads. If the timer reaches `0` before reloading, then the timeout BSCLTO (I2CxSTAT2[23]) flag and ERR (I2CxSTAT2[11]) are set. The BSCLTIE (I2CxINTC[23]) bit can be enabled to generate an error interrupt (I2CxEIF).

When timeout occurs in Client mode:

- SCL will be released (SCLREL=`1`), irrespective of STREN, by hardware automatically.
- SSPND (I2CxSTAT2[31]) will be cleared.
- SDA will be released after SCL, based on the SDASU timer value specification.
- CLTACT (I2CxSTAT2[30]) will be cleared.

When time-out occurs in Host mode:

- SSPND (I2CxSTAT2[31]) will be cleared.
- Hardware automatically sends the Stop bit to terminate the current transaction.

Note: SCL low time-out can wake up the device from sleep in the client address phase.

23.5.6.2. Client Extended Time-out ($t_{LOW:SEXT:}$)

Client extended time-out allows a host or client to extend its clock cycles measured from the initial Start to the Stop. Client extended time-out can be enabled by the writing time-out value to CBCTOTMR (I2CXCBCTO[23:0]) and then enabling the CBCTE (I2CxCON2[25]) bit. The timer continues to run until it detects a Start or Stop, and then the timer values reload. If the timer reaches `0` before reloading, then the time-out CBCLTO (I2CxSTAT2[21]) flag and ERR (I2CxSTAT2[11]) are set. The CBCTIE (I2CxINTC[21]) bit can be enabled to generate an error interrupt (I2CxEIF).

23.5.6.3. Host Extended Time-out ($t_{LOW:MEXT:}$)

Host extended time-out is used for a clock extension within one byte in a message as measured from:

START to $\overline{\text{ACK}}$

$\overline{\text{ACK}}$ to $\overline{\text{ACK}}$

$\overline{\text{ACK}}$ to STOP

Host extended time-out can be enabled by writing the time-out value to HBCTOTMR (I2CXHBCTO[23:0]) and then enabling the HBCTE (I2CxCON2[26]) bit. The timer continues to run until it detects Start or $\overline{\text{ACK}}$ or Stop, and then the timer values reload. If the timer reaches `0` before reloading, then the timeout HBCLTO (I2CxSTAT2[22]) flag and ERR (I2CxSTAT2[11]) are set. The HBCTIE (I2CxINTC[22]) bit can be enabled to generate an error interrupt (I2CxEIF).

23.5.6.4. Bus IDLE Time-out

Bus idle time-out can be used to define the bus free time (T_{BUF}) between Stop and Start. Bus free time, depending on the baud rate, should be loaded to BITOPR (I2CXBITO[23:0]), and then the bus idle timer should be enabled by setting BITE (I2CxCON2[16]). The bus idle time-out starts when the host software initiates a Start condition. BITO (I2CxSTAT2[20]) will be set after the defined bus idle time. The BITIE (I2CxINTC[20]) bit can be enabled to generate a generic interrupt (I2CxIF).

23.5.7. DMA Operation

The DMA can be used to transfer the data without the participation of the CPU. The I2CxTXIF and I2CxRXIF interrupts can be used to trigger DMA.

Note:
1. Packet size (PSZ) should be configured before data transmit/receive.
2. ND/$\overline{\text{A}}$ should be configured accordingly to indicate address/data transmission.
3. It is recommended to use Smart mode when using DMA to reduce CPU interaction.

23.5.7.1. Client Mode Transmission with DMA

Once the client is addressed by the host, ND/$\overline{\text{A}}$ (I2CxCON2[18]) should be set and then the first data should be loaded by the software. Client hardware automatically clears SCLREL and generates the transmit interrupt (I2CxTXIF) after the $\overline{\text{ACK}}$ if the error transmission I2CEF(I2CxSTAT2[11]) is not set and has not reached the packet size = `0`. The transmit buffer status bit TBF(I2CxSTAT1[0]) is clear when the I2CxTRN does not contain any transmit data. At this point, the transmit interrupt (I2CxTXIF) can trigger the DMA to load another byte into the buffer, and the transmit buffer full (TBF) is set. Client hardware sets SCLREL based on data setup time:

- The SDASUT(I2CxSDASUT[15:0]) value, when the TBF(I2CxSTAT1[0]) is set and Smart mode is enabled SMEN (I2CxCON2[17]).

23.5.7.2. Client Mode Reception with DMA

Once the client is addressed by the host, ND/$\overline{A}$ (I2CxCON2[18]) is set in the software. On receiving data from the host, the RBF (I2CxSTAT1[1]) bit will be set, and if the I2COV (I2CxSTAT1[6]) bit is cleared, the receive interrupt (I2CxRXIF) triggers the DMA to read the I2CRCV register into the data memory.

Note:

1. In Smart mode with DMA, it is recommended to configure DHEN = `0` and STREN = `1` to get a single interrupt.

23.5.7.3. Host Mode Transmission with DMA

Once the host has addressed the client, ND/$\overline{A}$ (I2CxCON2[18]) has to be set to the indicated data transmission and then load the first data to the transmit buffer. The Transmit Buffer Status bit TBF (I2CxSTAT1[0]) is cleared when the I2CxTRN does not contain any transmit data. At this point, the transmit interrupt (I2CxTXIF) can trigger the DMA to load another byte into the buffer. The host hardware sets the TBF bit, and the DMA continues to write the next data until the packet size (PSZ) becomes 0.

23.5.7.4. Host Mode Reception with DMA

Once the host has addressed the client, on receiving data from the client RBF (I2CxSTAT1[1]), the bit will be set and the receive interrupt (I2CxRXIF) triggers the DMA to read the I2CRCV register into the data memory.

Notes:

1. In Smart mode, receive enable (RCEN) will be set by the hardware automatically based on receive buffer status and packet size. For the first data byte, user software should enable receive by setting RCEN (I2CxCON1[3]).
2. To reduce CPU interaction when using DMA, it is recommended to use $\overline{\text{ACK}}$ Control bits (I2CxCON2.ACKC) to automatically send an acknowledgment.

23.5.8. Connection Considerations For I^2C Bus

Because the I^2C bus is a wired-AND bus connection, pull-up resistors (R_P) on the bus are required, as illustrated in Figure 23-40. The series resistors (R_S) are optional and are used to improve the Electrostatic Discharge (ESD) susceptibility.

The values of the resistors, R_P and R_S, depend on the following parameters:

- Supply voltage
- Bus capacitance
- Number of connected devices (input current + leakage current)
- Input level selection (I^2C or System Management Bus (SMBus))

Because the device must be able to pull the bus low against R_P, the current drawn by R_P must be greater than the I/O pin minimum sink current, I_{OL} at V_{OLMAX}, for the device output stage. Equation 23-3 shows the formula for computing the minimum pull-up resistance.

Equation 23-3. Minimum Pull-up Resistance

$$R_{PMIN} = \frac{(V_{DDMAX} - V_{OLMAX})}{I_{OL}}$$

In a 400 kHz system, a minimum rise time specification of 300 ns exists; in a 100 kHz system, the specification is 1000 ns. Because R_P must pull the bus up against the total capacitance, C_B, with a maximum rise time of 300 ns to (V_{DD} – 0.7V), the maximum resistance for the pull-up (R_{PMAX}) is computed using the formula as shown in Equation 23-4.

Equation 23-4. Maximum Pull-up Resistance

$$\frac{-tR}{C_B * [ln(1 - (V_{DDMAX} - V_{ILMAX}))]}$$

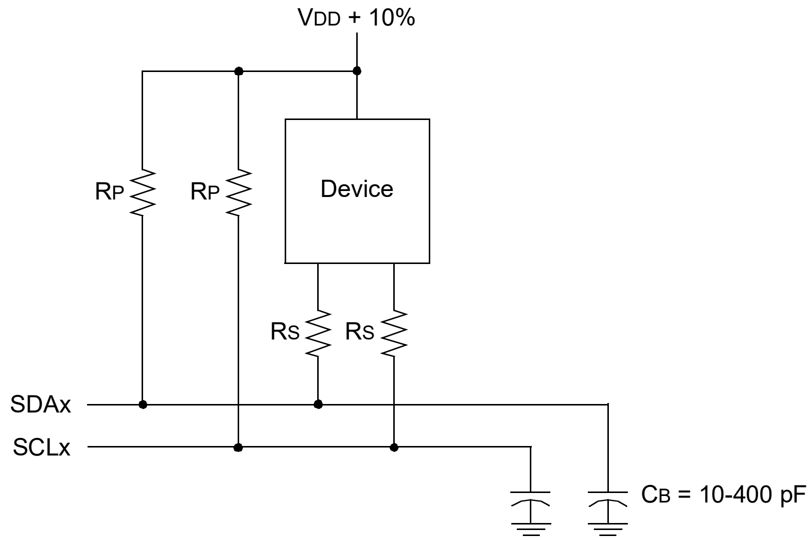
The maximum value for R_S is determined by the desired noise margin for the low level. R_S cannot drop enough voltage to make the device V_{OL} and the voltage across R_S more than the maximum V_{IL}. Equation 23-5 shows the formula for computing the maximum value for R_S.

Equation 23-5. Maximum Series Resistance

$$R_{SMAX} = \frac{(V_{ILMAX} - V_{OLMAX})}{I_{OLMAX}}$$

Note: The SCLx clock input must have a minimum high and low time for proper operation. Refer to the Electrical Characteristics section for more information on the high and low times of the I²C bus specification and requirements of the I²C module and I/O pins.

Figure 23-40. Sample Device Configuration for I²C Bus



23.5.8.1. Integrated Signal Conditioning

The SCLx and SDAx pins have an input glitch filter. The I²C bus requires this filter in both the 100 kHz and 400 kHz systems.

When operating on a 400 kHz bus, the I²C bus specification requires a slew rate control of the device pin output. This slew rate control is integrated into the device. If the DISSLW bit (I2CxCON1[9]) is cleared, the slew rate control is active. For other bus speeds, the I²C bus specification does not require slew rate control and the DISSLW bit should be set.

Some applications/systems with I²C buses require different input levels for V_{ILMAX} and V_{IHMIN}. In a normal I²C system, V_{ILMAX} is 0.3 V_{DD} and V_{IHMIN} is 0.7 V_{DD}. By contrast, in a SMBus system, V_{ILMAX} is set at 0.8V, while V_{IHMIN} is set at 2.1V.

The SMBEN bit (I2CxCON1[25:24]) controls the input levels. Setting SMBEN (= `1`) changes the input levels to SMBus specifications.

23.5.9. SMBus Support

The dsPIC33AK256MPS306 family devices have support for SMBus through options in the input voltage thresholds.

Table 23-7. I^2C Pin Voltage Threshold

Bus Options	SMEN SFR Bit (I2CxCONL[8])
I^2C (default)	0
SMBus 2.0	1
SMBus 3.0	1

23.5.10. Peripheral Module Disable (PMDx) Registers

The Peripheral Module Disable (PMDx) registers provide a method to disable the I^2C modules by stopping all clock sources supplied to that module. When a peripheral is disabled through the appropriate PMDx control bit, the peripheral is in a minimum power consumption state. The control and status registers associated with the peripheral are also disabled, so writes to those registers will have no effect and read values will be invalid. A peripheral module is only enabled if the I2CxMD bit in the PMDx register is cleared.

23.5.11. Effects of a Reset

A Reset disables the I^2C module and terminates any active or pending message activity. Refer to the I2Cx Control (I2CxCON1 or I2CxCON2) and I2Cx Status (I2CxSTAT) registers' definitions for the Reset conditions of those registers.

Note: In this discussion, Idle refers to the CPU Power-Saving state. The lower case idle refers to the time when the I^2C module is not transferring data on the bus.

23.6. Application Examples

23.6.1. Host Transmission (7-bit Address)

Example 23-1. Host Transmission (7-bit Address)

```
#include <xc.h>

#define mCLIENT_ADDRESS 0x4C
#define INIT 0
#define ADDRESS_PHASE 1
#define DATA_WRITE 2

#define PACKET_SIZE 10

unsigned char phase = INIT;

unsigned char hostTransmit[PACKET_SIZE] = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66,
0x77, 0x88, 0x99, 0xAA};
unsigned char transmittedCount = 0;

int main(void) {

    /*Configure I2C pins as digital*/

    //I2C module has Host and Client functionality which may both be active at
the same time.
    //In this example, Host functionality will be configured and used.

    //Configure baud rate for SCL @ 100kHz from 1/2 bus peripheral clock @ 4MHz
    I2C2LBRG = 16; //Set low-time baud rate
    I2C2HBRG = 16; //Set high-time baud rate

    /* Configure Bus IDle timeout*/
    I2C2CON2bits.BITE = 1;
    I2C2BITObits.BITOTMR = 76;

    /* Configured interrupt enable bits*/
    I2C2INTCbits.HACKSIE = 1;       // Assert HSTIF on ACK seq
    I2C2INTCbits.HDTXIE = 1;        // Assert HSTIF on TX
    I2C2INTCbits.HSCIE = 1;         // Assert HSTIF on start
    I2C2INTCbits.HSTIE = 1;         // Assert I2CxIF when HSTIF is set
    I2C2CON2bits.PSZ = PACKET_SIZE; // Packet size
    I2C2CON2bits.EOPSC = 0b01;      // End of packet will be set after data
```

```
bytes
    I2C2CON1bits.ON = 1;            // Enable I2C
    IFS2bits.I2C2IF = 0;            // Clear I2C general interrupt
    IEC2bits.I2C2IE = 1;            // Enable I2C general interrupt

    /*wait for bus idle */
    while (!I2C2STAT2bits.BITO);
    phase = ADDRESS_PHASE;
    I2C2CON1bits.SEN = 1;          // Send Start bit

    while (1) {

        //Once data transmission is complete, repeat the operation.
        if ((transmittedCount >= PACKET_SIZE) && (I2C2STAT2bits.STOPE)) {

            I2C2STAT2bits.STOPE = 0;    //Clear STOPE flag to detect next STOP
            I2C2STAT2bits.STARTE = 0;   //Clear STARTE flag since last START
condition

            transmittedCount = 0;      //Start transmitting from start of data
buffer

            I2C2CON2bits.NDA = 0;      //Will send address first, next byte is
not data
            I2C2STAT2bits.EOP = 0;     //Clear end of packet bit
            I2C2CON2bits.PSZ = 10;     //Re-initialize packet size

            //Wait for bus idle
            while (!I2C2STAT2bits.BITO);
            phase = ADDRESS_PHASE;
            I2C2CON1bits.SEN = 1;     // Send Start bit

        }
    }
}

void __attribute__((interrupt)) _I2C2Interrupt(void) {
    IFS2bits.I2C2IF = 0;

    switch (phase) {
        case ADDRESS_PHASE: {
            /*Verify if start has been sent*/
            if (I2C2STAT2bits.STARTE) {
                /* Transmit client address with RW =0 , writing to client*/
                I2C2TRN = (mCLIENT_ADDRESS << 1) | 0;
                phase = DATA_WRITE;
            }
            break;
        }
        case DATA_WRITE: {
            /* Set NDA to indicate next byte is data */
            if (I2C2CON2bits.NDA != 1) {
                I2C2CON2bits.NDA = 1;
            }
            /* If Packet size is 0, EOP will be asserted,send STOP on EOP*/
            if (I2C2STAT2bits.EOP) {
                I2C2CON1bits.PEN = 1;
            }
            else {
                if ((I2C2STAT1bits.ACKSTAT == 0) && (!I2C2STAT1bits.TBF) && (!
I2C2STAT1bits.TRSTAT)) {
                    I2C2TRN = hostTransmit[transmittedCount++]; // Send data
to client
                }
            }
            break;
        }
    }
}
```

23.6.2. Host Transmission (10-bit Address)

Example 23-2. Host Transmission (10-bit Address)

```
#include <xc.h>

#define INIT 0
#define ADDRESS_PHASE_UPPER_10BIT 1
#define ADDRESS_PHASE_LOWER_10BIT 2
#define DATA_WRITE 3

#define PACKET_SIZE 10

#define mCLIENT_ADDRESS 0x14C

unsigned char hostTransmit[PACKET_SIZE] = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66,
0x77, 0x88, 0x99, 0xAA};
unsigned char transmittedCount = 0;
unsigned char phase = INIT;

int main(void)
{
    /*Configure I2C pins as digital*/

    //I2C module has Host and Client functionality which may both be active at
the same time.
    //In this example, Host functionality will be configured and used.

    //Configure baud rate for SCL @ 100kHz from 1/2 bus peripheral clock @ 4MHz
    I2C2LBRG = 16; //Set low-time baud rate
    I2C2HBRG = 16; //Set high-time baud rate

    /* Configure Bus IDle timeout*/
    I2C2CON2bits.BITE = 1;
    I2C2BITObits.BITOTMR = 76;

    /* Configured interrupt enable bits*/
    I2C2INTCbits.HACKSIE = 1;         // Assert HSTIF on ACK seq
    I2C2INTCbits.HDTXIE = 1;          // Assert HSTIF on TX
    I2C2INTCbits.HSCIE = 1;           // Assert HSTIF on start
    I2C2INTCbits.HSTIE = 1;           // Assert I2CxIF when HSTIF is set
    I2C2CON2bits.PSZ = PACKET_SIZE;   // Packet size
    I2C2CON2bits.EOPSC = 0b01;        // End of packet will be set after data
bytes
    I2C2CON1bits.ON = 1;              // Enable I2C
    IFS2bits.I2C2IF = 0;              // Clear I2C general interrupt
    IEC2bits.I2C2IE = 1;              // Enable I2C general interrupt

    /*wait for bus idle */
    while(!I2C2STAT2bits.BITO);

    phase = ADDRESS_PHASE_UPPER_10BIT;
    I2C2CON1bits.SEN= 1; // Send Start bit


    while(1) {

        //If data transmission is complete, repeat the operation.
        if ((transmittedCount >= PACKET_SIZE) && (I2C2STAT2bits.STOPE)) {

            I2C2STAT2bits.STOPE = 0;    //Clear STOPE flag to detect next STOP
            I2C2STAT2bits.STARTE = 0;   //Clear STARTE flag since last START
condition

            transmittedCount = 0;       //Start transmitting from start of
data buffer

            I2C2CON2bits.NDA = 0;       //Will send address first, next byte
is not data
            I2C2STAT2bits.EOP = 0;      //Clear end of packet bit
            I2C2CON2bits.PSZ = PACKET_SIZE; //Re-initialize packet size

            //Wait for bus idle
            while (!I2C2STAT2bits.BITO);
            phase = ADDRESS_PHASE_UPPER_10BIT;
            I2C2CON1bits.SEN = 1; // Send Start bit
```

```
        }
    }
}

void __attribute__((interrupt)) _I2C2Interrupt(void)
{
    IFS2bits.I2C2IF = 0;
    switch(phase)
    {
        case ADDRESS_PHASE_UPPER_10BIT: {
            /*Verify if start has been sent*/
            if(I2C2STAT2bits.STARTE)
            {
                /* Transmit client address with RW =0 , writing to client*/
                I2C2TRN = ((((mCLIENT_ADDRESS >> 8) & 0x03)<< 1) | 0 ) +
0b11110000;
                phase = ADDRESS_PHASE_LOWER_10BIT;
            }
            break;
        }
        case ADDRESS_PHASE_LOWER_10BIT: {
            /*Verify if the upper 10bit address is ACKed*/
            if (I2C2STAT1bits.ACKSTAT ==0)
            {
            I2C2TRN = mCLIENT_ADDRESS & 0xFF ;
            phase = DATA_WRITE;
            }
            break;
        }
        case DATA_WRITE: {
            /* Set NDA to indicate next byte is data */
            if(I2C2CON2bits.NDA !=1)
            {
                I2C2CON2bits.NDA = 1;
            }
            /* If Packet size is 0, EOP will be asserted, send STOP on EOP*/
            if(I2C2STAT2bits.EOP)
            {
                I2C2CON1bits.PEN =1;
            }
            else
            {
                if ((I2C2STAT1bits.ACKSTAT ==0) && (!I2C2STAT1bits.TBF) && (!
I2C2STAT1bits.TRSTAT))
                {
                    I2C2TRN = hostTransmit[transmittedCount++]; // Send data
to client
                }
            }
            break;
        }
    }
}
```

23.6.3. Host Reception (7-bit Address)

Example 23-3. Host Reception (7-bit Address)

```
#include <xc.h>

#define mCLIENT_ADDRESS 0x4C

#define INIT 1
#define ADDRESS_PHASE 2
#define DATA_READ 3

#define PACKET_SIZE 10

volatile unsigned char phase = INIT;
volatile unsigned char hostReceived[PACKET_SIZE];
volatile unsigned char receivedCount = 0;

int main(void) {
```

```
    /*Configure I2C pins as digital*/

    //I2C module has Host and Client functionality which may both be active at
the same time.
    //In this example, Host functionality will be configured and used.

    //Configure baud rate for SCL @ 100kHz from 1/2 bus peripheral clock @ 4MHz
    I2C2LBRG = 16; //Set low-time baud rate
    I2C2HBRG = 16; //Set high-time baud rate

    /* Configure Bus IDle timeout*/
    I2C2CON2bits.BITE = 1;
    I2C2BITObits.BITOTMR = 76;

    /* Configured interrupt enable bits*/
    I2C2INTCbits.HACKSIE = 1;          // Assert HSTIF on ACK seq
    I2C2INTCbits.HDTXIE = 1;           // Assert HSTIF on TX
    I2C2INTCbits.HDRXIE = 1;           // Assert HSTIF on RX
    I2C2INTCbits.HSCIE = 1;            // Assert HSTIF on start
    I2C2INTCbits.HSTIE = 1;            // Assert I2CxIF when HSTIF is set
    I2C2CON2bits.SMEN = 1;             // Smart mode enabled
    I2C2CON2bits.ACKC = 0b10;          // ACK all the bytes expect last byte
    I2C2CON2bits.PSZ = PACKET_SIZE;    // Packet size
    I2C2CON2bits.EOPSC = 0b01;         // End of packet will be set after data
bytes
    I2C2CON1bits.ON = 1;               // Enable I2C
    IFS2bits.I2C2IF = 0;               // Clear I2C general interrupt
    IEC2bits.I2C2IE = 1;               // Enable I2C general interrupt

    /*wait for bus idle */
    while (!I2C2STAT2bits.BITO);
    phase = ADDRESS_PHASE;
    I2C2CON1bits.SEN = 1;              // Send Start bit

    while (1) {

        //Once reception is complete, repeat the operation.
        if ((receivedCount >= PACKET_SIZE) && (I2C2STAT2bits.STOPE)) {

            I2C2STAT2bits.STOPE = 0;    //Clear STOPE flag to detect next STOP
            I2C2STAT2bits.STARTE = 0;   //Clear STARTE flag since last START
condition

            receivedCount = 0;  //Reset receive buffer index

            I2C2CON1bits.RCEN = 0;          //Not yet receiving data on first
transaction
            I2C2CON2bits.NDA = 0;           //Will send address first, next
byte is not data
            I2C2CON2bits.PSZ = PACKET_SIZE; //Re-initialize packet size
            I2C2STAT2bits.EOP = 0;          //Clear end of packet bit

            /*wait for bus idle */
            while (!I2C2STAT2bits.BITO);
            phase = ADDRESS_PHASE;
            I2C2CON1bits.SEN = 1;           // Send Start bit
        }

    }


}

void __attribute__((interrupt)) _I2C2Interrupt(void) {
    IFS2bits.I2C2IF = 0;
    switch (phase) {
        case ADDRESS_PHASE: {
            /*Verify if start has been sent*/
            if (I2C2STAT2bits.STARTE) {
                phase = DATA_READ;
                /* Transmit client address with RW =1 , read client*/
                I2C2TRN = (mCLIENT_ADDRESS << 1) | 1;
            }
            break;
        }
        case DATA_READ: {
            /* Set NDA to indicate next byte is data */
```

```
            if (I2C2CON2bits.NDA != 1) {
                I2C2CON2bits.NDA = 1;
                /*Enable receive for the 1st time*/
                I2C2CON1bits.RCEN = 1;
            }
            /*Read the received data*/
            if (I2C2STAT1bits.RBF) {
                hostReceived[receivedCount++] = I2C2RCV;
            }
            /* If Packet size is 0, EOP will be asserted,send STOP on EOP*/
            if (I2C2STAT2bits.EOP == 1) {
                I2C2CON1bits.PEN = 1;
            }
            break;
        }
    }
}
```

23.6.4. Host Reception (10-bit Address)

Example 23-4. Host Reception (10-bit Address)

```
#include <xc.h>

#define mCLIENT_ADDRESS 0x14C

#define INIT 0
#define ADDRESS_PHASE_UPPER_10BIT_WRITE 1
#define ADDRESS_PHASE_LOWER_10BIT 2
#define RESTART 3
#define ADDRESS_PHASE_UPPER_10BIT_READ 4
#define DATA_READ 5

#define PACKET_SIZE 10

unsigned char phase = INIT;
unsigned char hostReceived[PACKET_SIZE];
unsigned char receivedCount = 0;



int main(void) {

    /*Configure I2C pins as digital*/

    //I2C module has Host and Client functionality which may both be active at
the same time.
    //In this example, Host functionality will be configured and used.

    //Configure baud rate for SCL @ 100kHz from 1/2 bus peripheral clock @ 4MHz
    I2C2LBRG = 16; //Set low-time baud rate
    I2C2HBRG = 16; //Set high-time baud rate

    /* Configure Bus IDle timeout*/
    I2C2CON2bits.BITE = 1;
    I2C2BITObits.BITOTMR = 76;

    /* Configured interrupt enable bits*/
    I2C2INTCbits.HACKSIE = 1;         // Assert HSTIF on ACK seq
    I2C2INTCbits.HDTXIE = 1;          // Assert HSTIF on TX
    I2C2INTCbits.HDRXIE = 1;          // Assert HSTIF on RX
    I2C2INTCbits.HSCIE = 1;           // Assert HSTIF on start
    I2C2INTCbits.HSTIE = 1;           // Assert I2CxIF when HSTIF is set
    I2C2CON2bits.SMEN = 1;            // Smart mode enabled
    I2C2CON2bits.ACKC = 0b10;         // ACK all the bytes expect last byte
    I2C2CON2bits.PSZ = PACKET_SIZE;   // Packet size

    I2C2CON2bits.EOPSC = 0b01;        // End of packet will be set after data
bytes
    I2C2CON1bits.ON = 1;              // Enable I2C
    IFS2bits.I2C2IF = 0;              // Clear I2C general interrupt
    IEC2bits.I2C2IE = 1;              // Enable I2C general interrupt

    /*wait for bus idle */
```

```
    while (!I2C2STAT2bits.BITO);
    phase = ADDRESS_PHASE_UPPER_10BIT_WRITE;
    I2C2CON1bits.SEN = 1; // Send Start bit

    while (1) {

        //Once reception is complete, repeat the operation.
        if ((receivedCount >= PACKET_SIZE) && (I2C2STAT2bits.STOPE)) {

            I2C2STAT2bits.STOPE = 0;    //Clear STOPE flag to detect next STOP
            I2C2STAT2bits.STARTE = 0;   //Clear STARTE flag since last START
condition

            receivedCount = 0;  //Reset receive buffer index

            I2C2CON1bits.RCEN = 0;          //Not yet receiving data on first
transaction
            I2C2CON2bits.NDA = 0;           //Next byte is address, not data
            I2C2CON2bits.PSZ = PACKET_SIZE; //Re-initialize packet size
            I2C2STAT2bits.EOP = 0;          //Clear end of packet bit

            /*wait for bus idle */
            while (!I2C2STAT2bits.BITO);
            phase = ADDRESS_PHASE_UPPER_10BIT_WRITE;
            I2C2CON1bits.SEN = 1; // Send Start bit
        }

    }
}

void __attribute__((interrupt)) _I2C2Interrupt(void) {
    IFS2bits.I2C2IF = 0;
    switch (phase) {
        case ADDRESS_PHASE_UPPER_10BIT_WRITE: {
            /*Verify if start has been sent*/
            if (I2C2STAT2bits.STARTE) {
                /* Transmit client address with RW =0 , write client*/
                I2C2TRN = ((((mCLIENT_ADDRESS >> 8) & 0x03) << 1) | 0) +
0b11110000;
                phase = ADDRESS_PHASE_LOWER_10BIT;
            }
            break;
        }
        case ADDRESS_PHASE_LOWER_10BIT: {
            /*Verify if the upper 10bit address is ACKed*/
            if (I2C2STAT1bits.ACKSTAT == 0) {
                I2C2TRN = mCLIENT_ADDRESS & 0xFF;
                phase = RESTART;
            }
            break;
        }
        case RESTART: {
            I2C2CON1bits.RSEN = 1; // Send restart
            phase = ADDRESS_PHASE_UPPER_10BIT_READ;
            break;
        }
        case ADDRESS_PHASE_UPPER_10BIT_READ: {
            /* Transmit client address with RW =1 , read client*/
            I2C2TRN = ((((mCLIENT_ADDRESS >> 8) & 0x03) << 1) | 1) +
0b11110000;
            phase = DATA_READ;
            break;
        }
        case DATA_READ: {
            /* Set NDA to indicate next byte is data */
            if (I2C2CON2bits.NDA != 1) {
                I2C2CON2bits.NDA = 1;
                /*Enable receive for the 1st time*/
                I2C2CON1bits.RCEN = 1;
            }
            /*Read the received data*/
            if (I2C2STAT1bits.RBF) {
                hostReceived[receivedCount++] = I2C2RCV;
            }
            /* If Packet size is 0, EOP will be asserted, send STOP on EOP*/
            if (I2C2STAT2bits.EOP) {
                I2C2CON1bits.PEN = 1;
            }
```

```
                break;
            }
        }
}
```

23.6.5. Client Reception (7-bit Address)

Example 23-5. Client Reception (7-bit Address)

```
#include <xc.h>

#define mCLIENT_ADDRESS 0x4C

#define PACKET_SIZE 10

volatile unsigned char clientReceived[PACKET_SIZE + 1];
volatile unsigned char recievedCount = 0;

int main(void) {

    /*Configure I2C pins as digital*/

    //I2C module has Host and Client functionality which may both be active at
the same time.
    //In this example, Client functionality will be configured and used.

    //Set up I2C2 client configuration
    I2C2ADD = mCLIENT_ADDRESS;          // Configure Client address
    I2C2INTCbits.CADDRIE = 1;           // Assert CLTIF on address detect
    I2C2INTCbits.CDRXIE = 1;            // Assert CLTIF on received byte
    I2C2INTCbits.CLTIE = 1;             // Assert I2CxIF when CLTIF is asserted
    I2C2CON2bits.SMEN = 1;              // Enable smart mode
    I2C2CON2bits.PSZ = PACKET_SIZE;     // Packet size
    I2C2CON1bits.ON = 1;                // Enable I2C
    IFS2bits.I2C2IF = 0;                // Clear I2C general interrupt
    IEC2bits.I2C2IE = 1;                // Enable I2C general interrupt

    while (1) {}

}
void __attribute__((__interrupt__)) _I2C2Interrupt(void) {
    IFS2bits.I2C2IF = 0;
    /* Read received address*/
    if ((I2C2STAT1bits.RBF)&& (!I2C2STAT1bits.D_A)) {
        recievedCount = 0;
        clientReceived[recievedCount] = I2C2RCV;
        recievedCount++;
    } /* Read received data byte*/
    else if ((I2C2STAT1bits.RBF)&& (I2C2STAT1bits.D_A)) {
        clientReceived[recievedCount] = I2C2RCV;
        recievedCount++; // Read received data
    }
}
```

23.6.6. Client Reception (10-bit Address)

Example 23-6. Client Reception (10-bit Address)

```
#include <xc.h>

#define mCLIENT_ADDRESS 0x14C

#define PACKET_SIZE 10

volatile unsigned char clientReceived[PACKET_SIZE + 2];
volatile unsigned char recievedCount = 0;

int main(void) {

    /*Configure I2C pins as digital*/
```

```
    //I2C module has Host and Client functionality which may both be active at
the same time.
    //In this example, Client functionality will be configured and used.

    //Set up I2C2 client configuration
    I2C2ADD = mCLIENT_ADDRESS;         // Configure Client address
    I2C2INTCbits.CADDRIE = 1;          // Assert CLTIF on address detect
    I2C2INTCbits.CDRXIE = 1;           // Assert CLTIF on received byte

    I2C2INTCbits.CLTIE = 1;            // Assert I2CxIF when CLTIF is asserted
    I2C2CON2bits.SMEN = 1;             // Enable smart mode
    I2C2CON2bits.PSZ = PACKET_SIZE;    // Packet size
    I2C2CON1bits.A10M = 1;             // 10bit addressing
    I2C2CON1bits.ON = 1;               // Enable I2C
    IFS2bits.I2C2IF = 0;               // Clear I2C general interrupt
    IEC2bits.I2C2IE = 1;               // Enable I2C general interrupt

    while (1) {}
}

void __attribute__((__interrupt__)) _I2C2Interrupt(void) {
    IFS2bits.I2C2IF = 0;

    /* Read received address*/
    if ((I2C2STAT1bits.RBF)&& (!I2C2STAT1bits.D_A)) {
        recievedCount = 0;
        clientReceived[recievedCount] = I2C2RCV;
        recievedCount++;
    }/* Read received data byte*/
    else if ((I2C2STAT1bits.RBF)&& (I2C2STAT1bits.D_A)) {
        clientReceived[recievedCount] = I2C2RCV;
        recievedCount++; // Read received data
    }

}
```

23.6.7. Client Transmission (7-bit Address)

Example 23-7. Client Transmission (7-bit Address)

```
#include <xc.h>

#define mCLIENT_ADDRESS 0x4C

#define PACKET_SIZE 10

volatile unsigned char clientReceived[PACKET_SIZE + 1];
volatile unsigned char recievedCount = 0;
volatile unsigned char transmittedCount = 0;
volatile unsigned char clientTransmit[PACKET_SIZE] = {0x11, 0x22, 0x33, 0x44,
0x55, 0x66, 0x77, 0x88, 0x99, 0xAA};

int main(void) {

    /*Configure I2C pins as digital*/

    //I2C module has Host and Client functionality which may both be active at
the same time.
    //In this example, Client functionality will be configured and used.

    //Set up I2C2 client configuration
    I2C2ADD = mCLIENT_ADDRESS;         // Configure Client address
    I2C2INTCbits.CADDRIE = 1;          // Assert CLTIF on address detect
    I2C2INTCbits.CDRXIE = 1;           // Assert CLTIF on received byte
    I2C2INTCbits.CDTXIE = 1;           // Assert CLTIF on transmit byte
    I2C2INTCbits.CLTIE = 1;            // Assert I2CxIF when CLTIF is asserted
    I2C2CON2bits.SMEN = 1;             // Enable smart mode
    I2C2CON2bits.PSZ = PACKET_SIZE;    // Packet size
    I2C2CON1bits.ON = 1;               // Enable I2C
    IFS2bits.I2C2IF = 0;               // Clear I2C general interrupt
    IEC2bits.I2C2IE = 1;               // Enable I2C general interrupt

    while (1) {}
```

```
}

void __attribute__((__interrupt__)) _I2C2Interrupt(void) {
    IFS2bits.I2C2IF = 0;

    /* Read received address*/
    if ((I2C2STAT1bits.RBF)&& (!I2C2STAT1bits.D_A)) {
        transmittedCount = 0;
        recievedCount = 0;
        clientReceived[recievedCount] = I2C2RCV;
        recievedCount++;
    }/* Read received data byte*/
    else if ((I2C2STAT1bits.RBF)&& (I2C2STAT1bits.D_A)) {
        clientReceived[recievedCount] = I2C2RCV;
        recievedCount++; // Read received data (?)
    }
    /* transmit byte if R_W is set */
    if ((I2C2STAT1bits.R_W) && (!I2C2STAT1bits.ACKSTAT)) {
        I2C2TRN = clientTransmit[transmittedCount++]; // transmit data
    }
}
```

23.6.8. Client Transmission (10-bit Address)

Example 23-8. Client Transmission (10-bit Address)

```
#include <xc.h>

#define mCLIENT_ADDRESS 0x14C

#define PACKET_SIZE 10

volatile unsigned char clientReceived[PACKET_SIZE + 2];
volatile unsigned char recievedCount = 0;
volatile unsigned char clientTransmit[PACKET_SIZE] = {0x11, 0x22, 0x33, 0x44,
0x55, 0x66, 0x77, 0x88, 0x99, 0xAA};
volatile unsigned char transmittedCount = 0;

int main(void) {

    /*Configure I2C pins as digital*/

    //I2C module has Host and Client functionality which may both be active at
the same time.
    //In this example, Client functionality will be configured and used.

    //Set up I2C2 client configuration
    I2C2ADD = mCLIENT_ADDRESS;          // Configure Client address
    I2C2INTCbits.CADDRIE = 1;           // Assert CLTIF on address detect
    I2C2INTCbits.CDRXIE = 1;            // Assert CLTIF on received byte
    I2C2INTCbits.CDTXIE = 1;            // Assert CLTIF on transmit byte
    I2C2INTCbits.CLTIE = 1;             // Assert I2CxIF when CLTIF is asserted
    I2C2CON2bits.SMEN = 1;              // Enable smart mode
    I2C2CON2bits.PSZ = PACKET_SIZE;     // Packet size
    I2C2CON1bits.A10M = 1;              // 10bit addressing
    I2C2CON1bits.ON = 1;                // Enable I2C
    IFS2bits.I2C2IF = 0;                // Clear I2C general interrupt
    IEC2bits.I2C2IE = 1;                // Enable I2C general interrupt

    while (1);
}

void __attribute__((__interrupt__)) _I2C2Interrupt(void) {

    IFS2bits.I2C2IF = 0;
    /* Read received address*/
    if ((I2C2STAT1bits.RBF)&& (!I2C2STAT1bits.D_A)) {
        transmittedCount = 0;
        recievedCount = 0;
        clientReceived[recievedCount] = I2C2RCV;
        recievedCount++;
    }/* Read received data byte*/
    else if ((I2C2STAT1bits.RBF)&& (I2C2STAT1bits.D_A)) {
        clientReceived[recievedCount] = I2C2RCV;
```

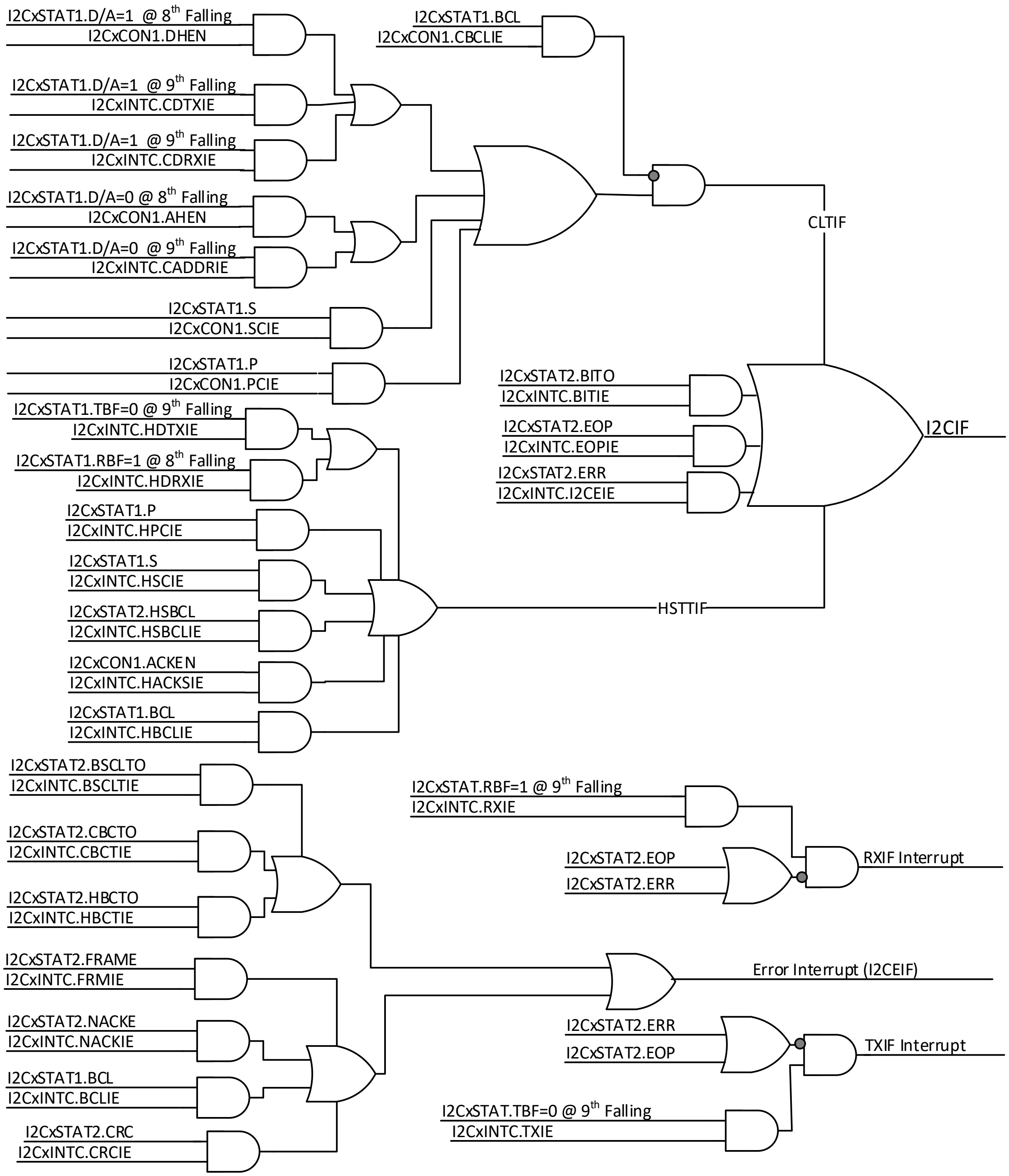
```
            recievedCount++; // Read received data
        }
        /* transmit byte if R_W is set */
        if ((I2C2STAT1bits.R_W) &&(!I2C2STAT1bits.ACKSTAT)) {
            I2C2TRN = clientTransmit[transmittedCount++]; //Transmit data
        }
}
```

23.7. Interrupts

I²C has four separate interrupts. To determine which event has caused the interrupt, the associated flag needs to be read and evaluated. All interrupt sources are shown in Figure 23-41.

- Receive Buffer Full Interrupt (I2CTXIF)
- Transmit Buffer Empty Interrupt (I2CRXIF)
- Generic Module Interrupt (I2CIF)
- Error Interrupt (I2CEIF)

Figure 23-41. Interrupt Logic



- The Receive interrupt will be generated when EOP=1 to read the last byte by DMA.
- The First Transmit Data byte write is expected write by the CPU.
- The Transmit interrupt is not generated after the last byte transmit.
- The HACKSIE controls the ACK Sequence completion interrupt generated at the ninth falling edge after transmitting the ACK|NACK in the Host Receive mode.
- The TXIF,RXIF and I2CEIF are not in the status bits; they are actual interrupt outputs.
- Data transmission interrupt is not generated to get CRC value from CPU or DMA in the CRC Auto Append mode since the transmitted CRC value is already present in the CRC calculator.

23.8. Operation in Power-Saving Modes

23.8.1. Sleep Mode in Client Mode

The I^{2}C module can wake up from Sleep mode on detecting a valid client address match. Because all bit shifting is done with reference to the external SCLx signal, generated by an I^{2}C host, transmissions and receptions can continue even while in Sleep mode. A SCL low time-out interrupt can wake up the client to release the bus to an Idle state. This can be enabled by setting BSCLTE (I2CxCON2[27]).

Note: As per the client I^{2}C behavior, a client interrupt is generated only on an address match. Therefore, when an I^{2}C client is in Sleep mode and it receives a message from the host, the clock required to match the received address is derived from the host. Only on an address match will the interrupt be generated and the device can wake up from Sleep, provided the interrupt has been enabled and an ISR has been defined.

23.8.2. Sleep Mode in Host Mode

If Sleep occurs in the middle of a host transmission and the state machine is partially into a transmission as the clocks stop, the behavior of the module will be undefined. Similarly, if Sleep occurs in the middle of a host reception, the module behavior will also be undefined. The transmitter and receiver will stop at Sleep when in Host mode. Register contents are not affected by going into Sleep mode or coming out of Sleep mode; there is no automatic way to prevent the Sleep entry if a transmission or reception is pending. The user software must synchronize the Sleep entry with an I^{2}C operation to avoid undefined module behavior.

23.8.3. When the Device Enters Idle Mode

When the device executes a PWRSAV 1 instruction, the device enters Idle mode. The module enters a Power-Saving state in Idle mode, depending on the I2CSIDL bit (I2CxCON1[13]). If I2CSIDL = `1`, the module enters a PowerSaving mode, similar to actions while entering Sleep mode. If I2CSIDL = `0`, the module does not enter a Power-Saving mode and continues to operate normally.

24. Improved Inter-Integrated Circuit (I3C)

The Improved Inter-Integrated Circuit (I^3C) is a multi-controller serial data communication interface that builds upon the traditional Inter-Integrated Circuit (I^2C) interface. Devices on the I^3C bus communicate in a Controller/Target environment where either the Controller or the Target device can initiate the communication. The I^3C interface is developed by the MIPI® Alliance and provides a fast, low-cost, low-power managed two-wire digital interface to improve sensor and device integration. The I^3C interface is backward compatible with the I^2C standard and adheres to the MIPI I3C® Basic Specification v1.1.1.

The I^3C module can operate as any one of the following in the I^3C system:

- Primary Controller
- Secondary Controller
- I^3C Target/I^2C Target

Key features of the I^3C module include the following:

- Two-Wire Serial Interface Up to 12.5 MHz.
- Supports Various Data Rates (FM, FM+, SDR, HDR-DDR)
- I^2C Backward Compatible with Some Limitations
- Hardware Assisted Device Role Switching in Secondary Controller Configuration
- Static or Dynamic Target Device Support
- In-Band Interrupt with Payload Support
- Hot-Join Support
- CRC/Parity Generation and Validation
- Supported Bus Monitor
- Bus-Free (in Controller Mode), Bus-Available (in Target Mode) and Bus-Idle (in Target Mode) Conditions
- Target Reset Pattern Feature
- Common Command Codes
- Grouped Addressing
- Vendor Specific CCC (Common Command Code) Transfer Support
- Virtual Targets

24.1. Device-Specific Information

Table 24-1. I^3C Summary Table

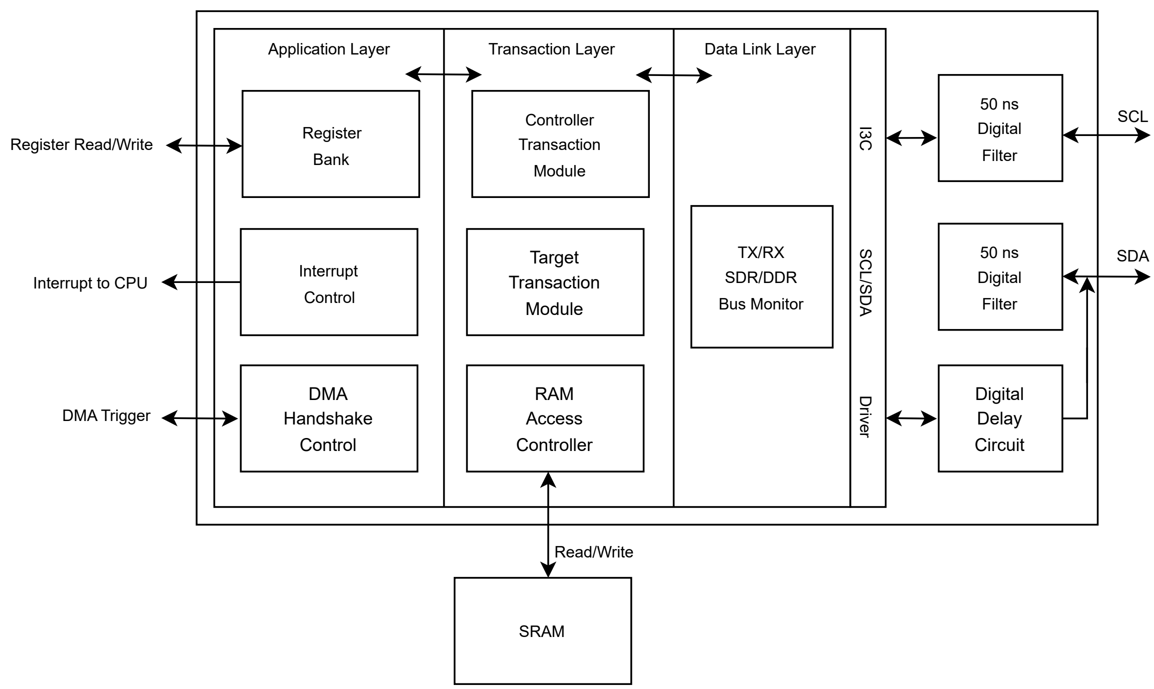
I^3C Module Instances	Clock Source	Peripheral Bus Speed
1	CLKGEN16	Slow (1:4 CPU Clock)

Table 24-2. Buffer Depth

FIFO	Depth
RX	32
TX	32
Extended TX FIFO	8
Command Queue	8
IBI Status Queue	8
IBI Data Queue	32
Device Characteristic Table	72
Device Address Table	18

24.2. Architectural Overview

Figure 24-1. I^3C Hardware Block Diagram



24.2.1. Register Bank

The register bank contains all the registers used to program the Controller for different bus modes, timing parameters and enabling or disabling of interrupts.

The register interface is also used for the transmission of commands and Tx data to the Controller and the reception of Rx data, IBI-data, and responses from the Controller/Target transaction module.

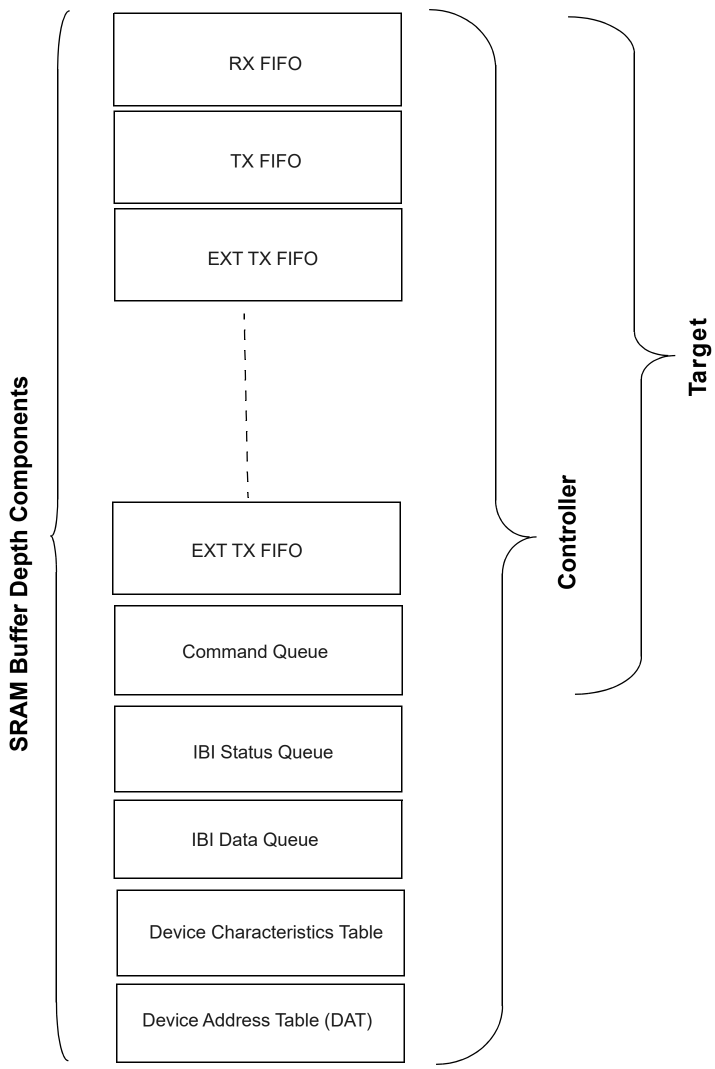
There are registers that are mapped to either FIFOs or Queues present in the SRAM.

24.2.1.1. Single Port RAM

The Single Port RAM (SPRAM) is a shared memory space for storing the following:

- Device address table
- Device characteristics table
- Command FIFO
- Transmit and Receive FIFOs
- IBI Status and Data FIFOs
- Extended Tx FIFOs

Figure 24-2. SRAM Memory Allocation



24.2.2. DMA Handshake Control

The DMA handshake interface module is responsible for the assertion and deassertion of the DMA handshake interface, based on the occupancy levels of Tx FIFO and Rx FIFOs.

This interface is provided only for Tx and Rx FIFOs. DMA acts as the flow controller for both the TX and RX channels, meaning it controls the block size.

24.2.3. Controller Transaction Module

The Controller transaction block is a state machine that controls both transmit and receive operations in the Active Controller mode. This module instructs the transmit and receive control blocks to operate based on commands from the application interface or in-band interrupts from the Targets.

24.2.4. Target Transaction Module

The Target transaction block is a state machine that controls both transmit and receive in the Target and Non-Active Controller mode. This module instructs the transmit and receive control blocks based on incoming transfer from the controller or commands from the application to drive the in-band interrupts.

24.2.5. Transmit Control and Receive Control Blocks

The transmit control and receive control blocks are used to control protocol-related transmit or receive signals (including timing signals) for the Controller and Target, and also for arbitration loss checks.

24.2.6. Bus Monitor

The bus monitor block handles various conditions, including:

- IBI Start and Bus-free in Controller mode
- START, RESTART and STOP conditions in Target mode
- Bus-available and Bus-idle conditions

24.2.7. I^2C Backward Compatibility

I^3C is backward compatible with many legacy I^2C devices, but there are some limitations, such as the lack of support for clock stretching, clock synchronization, high-speed mode and 10-bit addressing.

24.3. Register Summary

Note: Accessing 8 and 16 bits are not allowed. Always access as 32-bit.

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A9000	I3C1CTRL	31:24	ON	RESUME	ABORT	DMAEN	ADPTV	PEC	IDLECNT[1:0]	
		23:16								
		15:8								HOTJOIN
		7:0	I2CTGT							IBA
0x7A9004	I3C1ADD	31:24	DYNADDRVALID							
		23:16		DYNADDR[6:0]						
		15:8	STATADDRVALID							
		7:0		STATADDR[6:0]						
0x7A9008	I3C1HWCAP	31:24								
		23:16	HDRBN		VIRTGT	GRPADDR	IBICAP	HJCAP	DMACAP	HDRCLKPR[5]
		15:8	HDRCLKPR[4:0]					CLOCKPERIOD[5:3]		
		7:0	CLOCKPERIOD[2:0]			HDRTS	HDRDDR	DEVROLE[2:0]		
0x7A900C	I3C1CMDQUE	31:24	COMMAND[31:24]							
		23:16	COMMAND[23:16]							
		15:8	COMMAND[15:8]							
		7:0	COMMAND[7:0]							
0x7A9010	I3C1RESPQUE	31:24	RESPONSE[31:24]							
		23:16	RESPONSE[23:16]							
		15:8	RESPONSE[15:8]							
		7:0	RESPONSE[7:0]							
0x7A9014	I3C1TXRXDATA	31:24	TXRXDATA[31:24]							
		23:16	TXRXDATA[23:16]							
		15:8	TXRXDATA[15:8]							
		7:0	TXRXDATA[7:0]							
0x7A9018	I3C1IBIQUE	31:24	IBISTADAT[31:24]							
		23:16	IBISTADAT[23:16]							
		15:8	IBISTADAT[15:8]							
		7:0	IBISTADAT[7:0]							
0x7A901C	I3C1QUETHLDCON	31:24	IBISTATHLD[7:0]							
		23:16	IBIDATTHLD[7:0]							
		15:8	RESPBUFTHLD[7:0]							
		7:0	CMDEBTHLD[7:0]							
0x7A9020	I3C1BUFTHLD	31:24						RXSTARTTHLD[2:0]		
		23:16						TXSTARTTHLD[2:0]		
		15:8						RXTHLD[2:0]		
		7:0						TXTHLD[2:0]		
0x7A9024	I3C1IBIQNOTIFY	31:24								
		23:16								
		15:8								
		7:0					SIRREJ		MRREJ	HJREJ
0x7A9028 ... 0x7A9033	Reserved									
0x7A9034	I3C1RSTCON	31:24	BUSRST	RSTTYP[2:0]						
		23:16								
		15:8								
		7:0			IBIQRST	RXFIFORST	TXFIFORST	RESPQRST	CMDQRST	SOFTRST
0x7A9038	I3C1TGTESTA	31:24								
		23:16								
		15:8								
		7:0	MWLSTA	MRLSTA	ACTSTA[1:0]		HJINTEN		MRINTEN	

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A903C	I3C1INTSTA	31:24								
		23:16			EXTCMDTXTHLDSTA	EXTCMDSTA	SDARELSTA	GRPADDRSTA	TRSTPATSTA	STARTSTA
		15:8	BUSRSTSTA		BUSOWNSTA	IBIUPDSTA	READREQSTA	DEFTGTSTA	TRANSERRSTA	DYNADDRSTA
		7:0		CCCUPDSTA	TRANSABTSTA	RESPQSTA	CMDQSTA	IBITHLDSTA	RXTHLDSTA	TXTHLDSTA
0x7A9040	I3C1INTSTACON	31:24								
		23:16			EXTCMDTXTHLDSTAEN	EXTCMDSTAEN	SDARELSTAEN	GRPADDRSTAEN	TRSTPATSTAEN	STARTSTAEN
		15:8	BUSRSTSTAEN		BUSOWNSTAEN	IBIUPDSTAEN	READREQSTAEN	DEFTGTSTAEN	TRANSERRSTAEN	DYNADDRSTAEN
		7:0		CCCUPDSTAEN	TRANSABTSTAEN	RESPQSTAEN	CMDQSTAEN	IBITHLDSTAEN	RXTHLDSTAEN	TXTHLDSTAEN
0x7A9044	I3C1INTCON	31:24								
		23:16			EXTCMDTXTHLDINTEN	EXTCMDINTEN	SDARELINTEN	GRPADDRINTEN	TRSTPATINTEN	STARTINTEN
		15:8	BUSRSTINTEN		BUSOWNINTEN	IBIUPDINTEN	READREQINTEN	DEFTGTINTEN	TRANSERRINTEN	DYNADDRINTEN
		7:0		CCCUPDINTEN	TRANSABTINTEN	RESPQINTEN	CMDQINTEN	IBITHLDINTEN	RXTHLDINTEN	TXTHLDINTEN
0x7A9048	I3C1INTFORCE	31:24								
		23:16			EXTCMDTXTHLDFRC	EXTCMDFRC	SDARELFRC	GRPADDRFRC	TRSTPATFRC	STARTFRC
		15:8	BUSRSTFRC		BUSOWNFRC	IBIUPDFRC	READREQFRC	DEFTGTFRC	TRANSERRFRC	DYNADDRFRC
		7:0		CCCUPDFRC	TRANSABTFRC	RESPQFRC	CMDQFRC	IBITHLDFRC	RXTHLDFRC	TXTHLDFRC
0x7A904C	I3C1QLEVEL	31:24				IBISTATCNT[4:0]				
		23:16	IBIBUF[7:0]							
		15:8	RESPBUF[7:0]							
		7:0	CMDQEMTLOC[7:0]							
0x7A9050	I3C1BUFLEVEL	31:24				RXBUFLVL[12:8]				
		23:16	RXBUFLVL[7:0]							
		15:8								
		7:0	TXBUFEMTLVL[7:0]							
0x7A9054	I3C1STATE	31:24				IDLESTATE	CMDTID[3:0]			
		23:16			CMTFRSTAT[5:0]					
		15:8			CMTFRTYP[5:0]					
		7:0						CURRCNTRL	SDALEVEL	SCLLEVEL
0x7A9058	I3C1TGTCCCSTAT	31:24								
		23:16								
		15:8			FRMERR	BUFFNTAVAIL	DATNTRDY	OVFLWERR	TGTBUSY	UDFLWERR
		7:0	ACTIMOD[1:0]		PROTOERR		PNDINGINT[3:0]			
0x7A905C	I3C1ADDRTABPTR	31:24	ADDRTABDPTH[15:8]							
		23:16	ADDRTABDPTH[7:0]							
		15:8	ADDRTABSTRTADDR[15:8]							
		7:0	ADDRTABSTRTADDR[7:0]							
0x7A9060	I3C1CHARTABPTR	31:24								
		23:16	CURRCHTABINDX[4:0]					CHTABDPTH[6:4]		
		15:8	CHTABDPTH[3:0]				CHTABSTRTADDR[11:8]			
		7:0	CHTABSTRTADDR[7:0]							
0x7A9064 ... 0x7A906B	Reserved									
0x7A906C	I3C1VENDSPCPTR	31:24								
		23:16								
		15:8	VENDSTRTADDR[15:8]							
		7:0	VENDSTRTADDR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A9070	I3C1TGTMIPIID	31:24								
		23:16								
		15:8	MIPIMFGID[14:7]							
		7:0	MIPIMFGID[6:0]							PROVID
0x7A9074	I3C1TGTPID	31:24	PARTID[15:8]							
		23:16	PARTID[7:0]							
		15:8	INSTID[3:0]				PIDDCR[11:8]			
		7:0	PIDDCR[7:0]							
0x7A9078	I3C1TGTCON	31:24								
		23:16	HDRCAP[7:0]							
		15:8	DCR[7:0]							
		7:0	DEVICEROLE[1:0]		SDRHDR	BRGID	OFFLINECAP	IBIPYLD	IBIREQCAP	MAXDATSPDLMT
0x7A907C	I3C1TGTMAXLEN	31:24	MRL[15:8]							
		23:16	MRL[7:0]							
		15:8	MWL[15:8]							
		7:0	MWL[7:0]							
0x7A9080	I3C1TGTIBIPYLD	31:24								
		23:16	MAXRDTIME[23:16]							
		15:8	MAXRDTIME[15:8]							
		7:0	MAXRDTIME[7:0]							
0x7A9084	I3C1TGTDATSPD	31:24						SETACTSTATE	ACTSTATE[1:0]	
		23:16	Reserved[3:0]				STPPRMIT	CLKDATTIME[2:0]		
		15:8						MAXRDSPD[2:0]		
		7:0					DEFBYT	MAXWRSPD[2:0]		
0x7A9088 ... 0x7A908B	Reserved									
0x7A908C	I3C1TGTINT	31:24	SIRTGTINDX[3:0]							
		23:16	SIRDATLEN[7:0]							
		15:8	MDB[7:0]							
		7:0	CE3RECOV				MR	SIR[2:0]		
0x7A9090 ... 0x7A9093	Reserved									
0x7A9094	I3C1TGTSIRDAT	31:24	SIRBYT3[7:0]							
		23:16	SIRBYT2[7:0]							
		15:8	SIRBYT1[0]							
		7:0	SIRBYT0[7:0]							
0x7A9098	I3C1TGTIBIRESP	31:24								
		23:16		SIRRESPDATLEN[14:8]						
		15:8	SIRRESPDATLEN[7:0]							
		7:0							IBISTAT[1:0]	
0x7A909C ... 0x7A90AF	Reserved									
0x7A90B0	I3C1CTRLEXT	31:24								
		23:16								
		15:8								
		7:0				DPSLPCAP	REQACKCTRL		DEVOPMOD[1:0]	
0x7A90B4	I3C1SCLODTIM	31:24								
		23:16	ODHCNT[7:0]							
		15:8								
		7:0	ODLCNT[7:0]							
0x7A90B8	I3C1PPTIM	31:24								
		23:16	PPHCNT[7:0]							
		15:8								
		7:0	PPLCNT[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A90BC	I3C1I2CFMTIM	31:24	FMHCNT[15:8]							
		23:16	FMHCNT[7:0]							
		15:8	FMLCNT[15:8]							
		7:0	FMLCNT[7:0]							
0x7A90C0	I3C1I2CFMPTIM	31:24								
		23:16	I2CFMPHCNT[7:0]							
		15:8	I2CFMPLCNT[15:8]							
		7:0	I2CFMPLCNT[7:0]							
0x7A90C4 ... 0x7A90C7	Reserved									
0x7A90C8	I3C1SCLELCNT	31:24	LCNT4[7:0]							
		23:16	LCNT3[7:0]							
		15:8	LCNT2[7:0]							
		7:0	LCNT1[7:0]							
0x7A90CC	I3C1SCLETLCNT	31:24	STOPHLDCNT[3:0]							
		23:16								
		15:8								
		7:0					TERMNLCNT[3:0]			
0x7A90D0	I3C1SDAHLDTIM	31:24								
		23:16						SDATXHLD[2:0]		
		15:8						SDAPPODDLY[2:0]		
		7:0						SDAODPPDLY[2:0]		
0x7A90D4	I3C1BUSTIM	31:24	BUSAVAILTIM[15:8]							
		23:16	BUSAVAILTIM[7:0]							
		15:8	BUSFREETIM[15:8]							
		7:0	BUSFREETIM[7:0]							
0x7A90D8	I3C1BUSIDLETIM	31:24								
		23:16					BUSIDLETIM[19:16]			
		15:8	BUSIDLETIM[15:8]							
		7:0	BUSIDLETIM[7:0]							
0x7A90DC	I3C1CNTRLTIMOUT	31:24							TIMOUTCNT[25:24]	
		23:16	TIMOUTCNT[23:16]							
		15:8	TIMOUTCNT[15:8]							
		7:0	TIMOUTCNT[7:0]							
0x7A90E0 ... 0x7A90E7	Reserved									
0x7A90E8	I3C1QUESIZE	31:24								
		23:16	ETXBUFSZ[3:0]				IBIBUFSZ[3:0]			
		15:8	RESPBUFSZ[3:0]				CMDBUFSZ[3:0]			
		7:0	RXBUFSZ[3:0]				TXBUFSZ[3:0]			
0x7A90EC	I3C1RELSDATIM	31:24								
		23:16					RELSDATIM[19:16]			
		15:8	RELSDATIM[15:8]							
		7:0	RELSDATIM[7:0]							
0x7A90F0	I3C1HDRFLWCON	31:24								
		23:16								
		15:8								
		7:0	DDRCRCWDIND[1:0]		DDRWREARLYTERMN	DDRWRACKNACK				
0x7A90F4 ... 0x7A912F	Reserved									
0x7A9130	I3C1EXTTXDAT0	31:24	EXTTXDAT0[31:24]							
		23:16	EXTTXDAT0[23:16]							
		15:8	EXTTXDAT0[15:8]							
		7:0	EXTTXDAT0[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A9134	I3C1EXTTXDAT1	31:24	EXTTXDAT1[31:24]							
		23:16	EXTTXDAT1[23:16]							
		15:8	EXTTXDAT1[15:8]							
		7:0	EXTTXDAT1[7:0]							
0x7A9138	I3C1EXTTXDAT2	31:24	EXTTXDAT2[31:24]							
		23:16	EXTTXDAT2[23:16]							
		15:8	EXTTXDAT2[15:8]							
		7:0	EXTTXDAT2[7:0]							
0x7A913C	I3C1EXTTXDAT3	31:24	EXTTXDAT3[31:24]							
		23:16	EXTTXDAT3[23:16]							
		15:8	EXTTXDAT3[15:8]							
		7:0	EXTTXDAT3[7:0]							
0x7A9140 ... 0x7A914F	Reserved									
0x7A9150	I3C1EXTCMDTXTHLD	31:24								
		23:16								
		15:8								
		7:0					CMD3TXTHLDSTAT	CMD2TXTHLDSTAT	CMD1TXTHLDSTAT	CMD0TXTHLDSTAT
0x7A9154	I3C1EXT01BSTA	31:24								
		23:16	EXT1TBEL[7:0]							
		15:8								
		7:0	EXT0TBEL[7:0]							
0x7A9158	I3C1EXT23BSTA	31:24								
		23:16	EXT3TBEL[7:0]							
		15:8								
		7:0	EXT2TBEL[7:0]							
0x7A915C ... 0x7A9163	Reserved									
0x7A9164	I3C1EXTCMD0	31:24	EXTCMD0[31:24]							
		23:16	EXTCMD0[23:16]							
		15:8	EXTCMD0[15:8]							
		7:0	EXTCMD0[7:0]							
0x7A9168	I3C1EXTCMD02	31:24	EXTCMD02[31:24]							
		23:16	EXTCMD02[23:16]							
		15:8	EXTCMD02[15:8]							
		7:0	EXTCMD02[7:0]							
0x7A916C ... 0x7A916F	Reserved									
0x7A9170	I3C1EXTCMD1	31:24	EXTCMD1[31:24]							
		23:16	EXTCMD1[23:16]							
		15:8	EXTCMD1[15:8]							
		7:0	EXTCMD1[7:0]							
0x7A9174	I3C1EXTCMD12	31:24	EXTCMD12[31:24]							
		23:16	EXTCMD12[23:16]							
		15:8	EXTCMD12[15:8]							
		7:0	EXTCMD12[7:0]							
0x7A9178 ... 0x7A917B	Reserved									
0x7A917C	I3C1EXTCMD2	31:24	EXTCMD2[31:24]							
		23:16	EXTCMD2[23:16]							
		15:8	EXTCMD2[15:8]							
		7:0	EXTCMD2[7:0]							
0x7A9180	I3C1EXTCMD22	31:24	EXTCMD22[31:24]							
		23:16	EXTCMD22[23:16]							
		15:8	EXTCMD22[15:8]							
		7:0	EXTCMD22[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A9184 ... 0x7A9187	Reserved									
0x7A9188	I3C1EXTCMD3	31:24	EXTCMD3[31:24]							
		23:16	EXTCMD3[23:16]							
		15:8	EXTCMD3[15:8]							
		7:0	EXTCMD3[7:0]							
0x7A918C	I3C1EXTCMD32	31:24	EXTCMD32[31:24]							
		23:16	EXTCMD32[23:16]							
		15:8	EXTCMD32[15:8]							
		7:0	EXTCMD32[7:0]							
0x7A9190 ... 0x7A91C3	Reserved									
0x7A91C4	I3C1ETXQRSTCON	31:24								
		23:16								
		15:8								
		7:0					ETXFIFORST3	ETXFIFORST2	ETXFIFORST1	ETXFIFORST0
0x7A91C8	I3C1ECMDVLDSTA	31:24								
		23:16								
		15:8								
		7:0					CMD3VLD	CMD2VLD	CMD1VLD	CMD0VLD
0x7A91CC	I3C1ECRTCON	31:24								
		23:16						EXTTXBUFTHLD[2:0]		
		15:8								
		7:0						RSPDATTHLD[2:0]		
0x7A91D0	I3C1VIRTGTPTR	31:24	VIRTGTDPTH[15:8]							
		23:16	VIRTGTDPTH[7:0]							
		15:8	VIRTGTSTRTADDR[15:8]							
		7:0	VIRTGTSTRTADDR[7:0]							
0x7A91D4 ... 0x7A91D7	Reserved									
0x7A91D8	I3C1DMACHSEL	31:24								
		23:16								
		15:8								
		7:0	CH1SEL[3:0]				CH0SEL[3:0]			
0x7A91DC ... 0x7A91DF	Reserved									
0x7A91E0	I3C1GRPADDR0STA	31:24								
		23:16								
		15:8				T4GRPADDR0V	T3GRPADDR0V	T2GRPADDR0V	T1GRPADDR0V	T0GRPADDR0V
		7:0		GRPADDR0V[6:0]						
0x7A91E4	I3C1GRPADDR1STA	31:24								
		23:16								
		15:8				T4GRPADDR1V	T3GRPADDR1V	T2GRPADDR1V	T1GRPADDR1V	T0GRPADDR1V
		7:0		GRPADDR1[6:0]						
0x7A91E8	I3C1GRPADDR2STA	31:24								
		23:16								
		15:8				T4GRPADDR2V	T3GRPADDR2V	T2GRPADDR2V	T1GRPADDR2V	T0GRPADDR2V
		7:0		GRPADDR2[6:0]						
0x7A91EC	I3C1GRPADDR3STA	31:24								
		23:16								
		15:8				T4GRPADDR3V	T3GRPADDR3V	T2GRPADDR3V	T1GRPADDR3V	T0GRPADDR3V
		7:0		GRPADDR3[6:0]						

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A91F0 ... 0x7A91FF	Reserved									
0x7A9200	I3C1DEVCHARTAB1LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9200	I3C1SECDEVCHARTAB1	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9204	I3C1DEVCHARTAB1LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9204	I3C1SECDEVCHARTAB2	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9208	I3C1DEVCHARTAB1LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A9208	I3C1SECDEVCHARTAB3	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A920C	I3C1DEVCHARTAB1LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A920C	I3C1SECDEVCHARTAB4	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9210	I3C1DEVCHARTAB2LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9210	I3C1SECDEVCHARTAB5	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9214	I3C1DEVCHARTAB2LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9214	I3C1SECDEVCHARTAB6	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9218	I3C1DEVCHARTAB2LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A9218	I3C1SECDEVCHARTAB7	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A921C	I3C1DEVCHARTAB2LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A921C	I3C1SECDEVCHARTAB8	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9220	I3C1DEVCHARTAB3LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9220	I3C1SECDEVCHARTAB9	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9224	I3C1DEVCHARTAB3LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9224	I3C1SECDEVCHARTAB10	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9228	I3C1SECDEVCHARTAB11	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9228	I3C1DEVCHARTAB3LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A922C	I3C1SECDEVCHARTAB12	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A922C	I3C1DEVCHARTAB3LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9230	I3C1SECDEVCHARTAB13	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9230	I3C1DEVCHARTAB4LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9234	I3C1SECDEVCHARTAB14	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9234	I3C1DEVCHARTAB4LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9238	I3C1SECDEVCHARTAB15	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A9238	I3C1DEVCHARTAB4LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A923C	I3C1SECDEVCHARTAB16	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A923C	I3C1DEVCHARTAB4LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9240	I3C1SECDEVCHARTAB17	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9240	I3C1DEVCHARTAB5LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9244	I3C1SECDEVCHARTAB18	31:24	DYNADDR[7:0]							
		23:16	DCRTYP[7:0]							
		15:8	BCRTYP[7:0]							
		7:0	BCRTYP[7:0]							
0x7A9244	I3C1DEVCHARTAB5LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9248	I3C1DEVCHARTAB5LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A924C	I3C1DEVCHARTAB5LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9250	I3C1DEVCHARTAB6LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9254	I3C1DEVCHARTAB6LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9258	I3C1DEVCHARTAB6LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A925C	I3C1DEVCHARTAB6LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9260	I3C1DEVCHARTAB7LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9264	I3C1DEVCHARTAB7LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A9268	I3C1DEVCHARTAB7LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A926C	I3C1DEVCHARTAB7LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9270	I3C1DEVCHARTAB8LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9274	I3C1DEVCHARTAB8LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9278	I3C1DEVCHARTAB8LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A927C	I3C1DEVCHARTAB8LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9280	I3C1DEVCHARTAB9LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9284	I3C1DEVCHARTAB9LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9288	I3C1DEVCHARTAB9LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A928C	I3C1DEVCHARTAB9LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9290	I3C1DEVCHARTAB10LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9294	I3C1DEVCHARTAB10LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9298	I3C1DEVCHARTAB10LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A929C	I3C1DEVCHARTAB10LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A92A0	I3C1DEVCHARTAB11LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A92A4	I3C1DEVCHARTAB11LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A92A8	I3C1DEVCHARTAB11LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A92AC	I3C1DEVCHARTAB11LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A92B0	I3C1DEVCHARTAB12LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A92B4	I3C1DEVCHARTAB12LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A92B8	I3C1DEVCHARTAB12LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A92BC	I3C1DEVCHARTAB12LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A92C0	I3C1DEVCHARTAB13LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A92C4	I3C1DEVCHARTAB13LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A92C8	I3C1DEVCHARTAB12LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A92CC	I3C1DEVCHARTAB13LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A92D0	I3C1DEVCHARTAB14LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A92D4	I3C1DEVCHARTAB14LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A92D8	I3C1DEVCHARTAB14LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A92DC	I3C1DEVCHARTAB14LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A92E0	I3C1DEVCHARTAB15LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A92E4	I3C1DEVCHARTAB15LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A92E8	I3C1DEVCHARTAB15LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A92EC	I3C1DEVCHARTAB15LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A92F0	I3C1DEVCHARTAB16LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A92F4	I3C1DEVCHARTAB16LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A92F8	I3C1DEVCHARTAB16LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A92FC	I3C1DEVCHARTAB16LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9300	I3C1DEVCHARTAB17LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9304	I3C1DEVCHARTAB17LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9308	I3C1DEVCHARTAB17LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							
0x7A930C	I3C1DEVCHARTAB17LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9310	I3C1DEVCHARTAB18LOC1	31:24	MSBPROVID[31:24]							
		23:16	MSBPROVID[23:16]							
		15:8	MSBPROVID[15:8]							
		7:0	MSBPROVID[7:0]							
0x7A9314	I3C1DEVCHARTAB18LOC2	31:24								
		23:16								
		15:8	LSBPROVID[15:8]							
		7:0	LSBPROVID[7:0]							
0x7A9318	I3C1DEVCHARTAB18LOC3	31:24								
		23:16								
		15:8	BCR[7:0]							
		7:0	DCR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A931C	I3C1DEVCHARTAB18LOC4	31:24								
		23:16								
		15:8								
		7:0	DEVDYNADDR[7:0]							
0x7A9320 ... 0x7A937F	Reserved									
0x7A9380	I3C1DEVADDRTAB1LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A9384	I3C1DEVADDRTAB2LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A9388	I3C1DEVADDRTAB3LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A938C	I3C1DEVADDRTAB4LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A9390	I3C1DEVADDRTAB5LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A9394	I3C1DEVADDRTAB6LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A9398	I3C1DEVADDRTAB7LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A939C	I3C1DEVADDRTAB8LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93A0	I3C1DEVADDRTAB9LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A93A4	I3C1DEVADDRTAB10LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93A8	I3C1DEVADDRTAB11LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93AC	I3C1DEVADDRTAB12LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93B0	I3C1DEVADDRTAB13LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93B4	I3C1DEVADDRTAB14LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93B8	I3C1DEVADDRTAB15LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93BC	I3C1DEVADDRTAB16LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93C0	I3C1DEVADDRTAB17LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93C4	I3C1DEVADDRTAB18LOC1	31:24	DEVICE	DEVNACKRTRYCNT[1:0]						
		23:16	DEVDYNADDR[7:0]							
		15:8	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
		7:0	DDRWRACKNACKEN	STATICADDR[6:0]						
0x7A93C8 ... 0x7A94FF	Reserved									

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A9500	I3C1VIRTGT1CON	31:24	VT1ENABLE							
		23:16								
		15:8	VT1DYNADDRV		VT1DYNADDR[5:0]					
		7:0	VT1STATICADDRV	VT1STATICADDR[6:0]						
0x7A9504	I3C1VIRTGT1MIPIID	31:24								
		23:16								
		15:8	VT1MIPIMFGID[14:7]							
		7:0	VT1MIPIMFGID[6:0]							VT1PROVIDSEL
0x7A9508	I3C1VIRTGT1PID	31:24	VT1PARTID[15:8]							
		23:16	VT1PARTID[7:0]							
		15:8	VT1INSTID[3:0]				VT1PIDDCR[11:8]			
		7:0	VT1PIDDCR[7:0]							
0x7A950C	I3C1VIRTGT1CHARCON	31:24								
		23:16	VT1HDR[7:0]							
		15:8	VT1DCR[7:0]							
		7:0	VT1DEVROLE[1:0]		VT1HDRCAP	VT1BRIDGEID	VT1OFFLINECAP	VT1IBIPAYLD	VT1IBIREQCAP	VT1MAXDATSPDLIM
0x7A9510 ... 0x7A951F	Reserved									
0x7A9520	I3C1VIRTGT2CON	31:24	VT2ENABLE							
		23:16								
		15:8	VT2DYNADDRV		VT2DYNADDR[5:0]					
		7:0	VT2STATICADDRV	VT2STATICADDR[6:0]						
0x7A9524	I3C1VIRTGT2CON	31:24								
		23:16								
		15:8	VT2MIPIMFGID[14:7]							
		7:0	VT2MIPIMFGID[6:0]							VT2PROVIDSEL
0x7A9528	I3C1VIRTGT2PID	31:24								
		23:16	VT2PARTID[7:0]							
		15:8	VT2INSTID[3:0]				VT2PIDDCR[11:8]			
		7:0	VT2PIDDCR[7:0]							
0x7A952C	I3C1VIRTGT2CHARCON	31:24								
		23:16	VT2HDR[7:0]							
		15:8	VT2DCR[7:0]							
		7:0	VT2DEVROLE[1:0]		VT2HDRCAP	VT2BRIDGEID	VT2OFFLINECAP	VT2IBIPAYLD	VT2IBIREQCAP	VT2MAXDATSPDLIM
0x7A9530 ... 0x7A953F	Reserved									
0x7A9540	I3C1VIRTGT3CON	31:24	VT3ENABLE							
		23:16								
		15:8	VT3DYNADDRV		VT3DYNADDR[5:0]					
		7:0	VT3STATICADDRV	VT3STATICADDR[6:0]						
0x7A9544	I3C1VIRTGT3MIPIID	31:24								
		23:16								
		15:8	VT3MIPIMFGID[14:7]							
		7:0	VT3MIPIMFGID[6:0]							VT3PROVIDSEL
0x7A9548	I3C1VIRTGT3PID	31:24								
		23:16	VT3PARTID[7:0]							
		15:8	VT3INSTID[3:0]				VT3PIDDCR[11:8]			
		7:0	VT3PIDDCR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x7A954C	I3C1VIRTGT3CHARCON	31:24								
		23:16	VT3HDR[7:0]							
		15:8	VT3DCR[7:0]							
		7:0	VT3DEVROLE[1:0]		VT3HDRCAP	VT3BRIDGEID	VT3OFFLINECAP	VT3IBIPAYLD	VT3IBIREQCAP	VT3MAXDATSPDLIM
0x7A9550 ... 0x7A955F	Reserved									
0x7A9560	I3C1VIRTGT4CON	31:24	VT4ENABLE							
		23:16								
		15:8	VT4DYNADDRV		VT4DYNADDR[5:0]					
		7:0	VT4STATICADDRV	VT4STATICADDR[6:0]						
0x7A9564	I3C1VIRTGT4MIPIID	31:24								
		23:16								
		15:8	VT4MIPIMFGID[14:7]							
		7:0	VT4MIPIMFGID[6:0]							VT4PROVIDSEL
0x7A9568	I3C1VIRTGT4PID	31:24								
		23:16	VT4PARTID[7:0]							
		15:8	VT4INSTID[3:0]				VT4PIDDCR[11:8]			
		7:0	VT4PIDDCR[7:0]							
0x7A956C	I3C1VIRTGT4CHARCON	31:24								
		23:16	VT4HDR[7:0]							
		15:8	VT4DCR[7:0]							
		7:0	VT4DEVROLE[1:0]		VT4HDRCAP	VT4BRIDGEID	VT4OFFLINECAP	VT4IBIPAYLD	VT4IBIREQCAP	VT4MAXDATSPDLIM
0x7A9570	I3C1CON	31:24			SLPEN			ALTI3C	INTPUR	
		23:16	TTXDMACHSEL1[3:0]				TTXDMACHSEL0[3:0]			
		15:8	ON		SIDL	FLTINJ	SLWEN	SMB3	SMB2	I2C
		7:0	TXDMACHSEL[3:0]				RXDMACHSEL[3:0]			
0x7A9574	I3C1PG	31:24								
		23:16								
		15:8				DDLYEN	DDLYNUMC[3:0]			
		7:0				DFLTEN	PGSMPL[3:0]			
0x7A9578	I3C1FLTINJ	31:24								
		23:16			ECCFADDR[5:0]					
		15:8			FLT2PTR[5:0]					
		7:0			FLT1PTR[5:0]					

24.3.1. Device Control Register

Name: I3CxCTRL
Offset: 0x7A9000

Notes:
1. In Target mode, the I^{3}C operation is enabled only after providing SCL clocks to bring it out of Reset. This can be achieved by scheduling any transfer (placeholder transfer) ending in a STOP before initiating valid transfers. The placeholder transfer is ignored by the Target. Once enabled, the module responds to transfers on the bus only after it observes the Bus Available condition for I3CxBUSTIM[BUSAVAILTIM] * I3CxCTRL[IDLECNT] counts of the peripheral clock period.
2. The Controller gets disabled after the commands in the command queue (if any) are executed, and the Controller is in the IDLE state.

Bit	31	30	29	28	27	26	25	24
	ON	RESUME	ABORT	DMAEN	ADPTV	PEC	IDLECNT[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
								HOTJOIN
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
	I2CTGT							IBA
Access	R/W							R/W
Reset	0							0

Bit 31 – ON I^{3}C Enable bit

Value	Description
1	Enables the I^{3}C operation[(1)].
0	Disables the I^{3}C operation[(2)].

Bit 30 – RESUME I^{3}C Module Resume bit

Value	Description
1	I^{3}C module operation resume enable.
0	I^{3}C module resume operation complete. Auto-cleared once the module resumes.

Bit 29 – ABORT I^{3}C Module Abort bit (Controller mode only)
This bit is used in the Controller mode of operation.

Value	Description
1	Abort ongoing transaction. The Controller issues the STOP condition after the complete data byte is transferred or received.
0	Abort completed, auto-cleared once the transfer is aborted.

Bit 28 – DMAEN DMA Handshake Interface Enable bit

Value	Description
1	Enables the DMA handshake control to interact with the DMA.
0	The DMA handshake control has no significance.

Bit 27 – ADPTV Adaptive mode bit (Target mode only)

Value	Description
1	Hot-Join request is initiated only after it has switched to I^{3}C mode of operation.
0	Initiates a Hot-Join always.

Bit 26 – PEC PEC Enable bit

Value	Description
1	PEC support is enabled.
0	PEC support is not enabled.

Bits 25:24 – IDLECNT[1:0] Idle Count Multiplier bits (Target mode only)

Value	Description
11	I3CxBUSTIM[BUSAVAILTIM] * 8
10	I3CxBUSTIM[BUSAVAILTIM] *4
01	I3CxBUSTIM[BUSAVAILTIM] * 2
00	I3CxBUSTIM[BUSAVAILTIM] * 1

Bit 8 – HOTJOIN Hot-Join ACK/NACK Control bit (Controller mode only)

Value	Description
1	NACK and send broadcast CCC to disable Hot-Join.
0	ACK the Hot-Join request.

Bit 7 – I2CTGT (Controller mode only)

Value	Description
1	I^{2}C Target present.
0	I^{2}C Target not present.

Bit 0 – IBA I^{3}C Broadcast Address Include bit (Controller mode only)

Value	Description
1	I^{3}C Broadcast Address is included for private transfers.
0	I^{3}C Broadcast Address is not included for private transfers.

24.3.2. Device Address Register

Name: I3CxADD
Offset: 0x7A9004

Note: This register is used to program device Dynamic Addresses and their respective valid bits in the Controller mode of operation. The software Reset will not clear this register. In the Target mode of operation, this register reflects the Static and Dynamic Addresses and their respective valid bits.

Bit	31	30	29	28	27	26	25	24
	DYNADDRVALID							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
		DYNADDR[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STATADDRVALID							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
		STATADDR[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 31 – DYNADDRVALID Dynamic Address Valid bit

Value	Description
1	Dynamic Address is valid.
0	Dynamic Address is invalid.

Bits 22:16 – DYNADDR[6:0] Device Dynamic Address bits

Bit 15 – STATADDRVALID Static Address Valid bit

Value	Description
1	Static Address is valid.
0	Static Address is invalid.

Bits 6:0 – STATADDR[6:0] Device Static Address bits

24.3.3. Hardware Capability Register

Name: I3CxHWCAP
Offset: 0x7A9008

This register reflects the configured capabilities of the module.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	HDRBN		VIRTGT	GRPADDR	IBICAP	HJCAP	DMACAP	HDRCLKPR[5]
Access	R/W		R	R	R	R	R	R
Reset	0		1	1	1	1	1	0

Bit	15	14	13	12	11	10	9	8
	HDRCLKPR[4:0]					CLOCKPERIOD[5:3]		
Access	R	R	R	R	R	R	R	R
Reset	0	1	0	0	0	0	1	0

Bit	7	6	5	4	3	2	1	0
	CLOCKPERIOD[2:0]			HDRTS	HDRDDR	DEVROLE[2:0]		
Access	R	R	R	R	R	R	R	R
Reset	1	0	1	0	1	0	1	1

Bit 23 – HDRBN

Value	Description
1	HDR-BT is supported.
0	HDR-BT is not supported.

Bit 21 – VIRTGT

Value	Description
1	Virtual Target is supported.
0	Virtual Target is not supported.

Bit 20 – GRPADDR

Value	Description
1	Group Address is supported.
0	Group Address is not supported.

Bit 19 – IBICAP

Value	Description
1	IBI is capable.
0	IBI is not capable.

Bit 18 – HJCAP

Value	Description
1	Hot-Join is capable.
0	Hot-Join is not capable.

Bit 17 – DMACAP

Value	Description
1	DMA handshake interface is supported.
0	DMA handshake interface is not supported.

Bits 16:11 – HDRCLKPR[5:0]

Reflects the HDR clock period.

Bits 10:5 – CLOCKPERIOD[5:0]

Reflects clock period.

Bit 4 – HDRTS

Value	Description
1	HDR-TS is supported.
0	HDR-TS is not supported.

Bit 3 – HDRDDR

Value	Description
1	HDR-DDR is supported.
0	HDR-DDR is not supported.

Bits 2:0 – DEVROLE[2:0]

Specifies the configured role of module.

Value	Description
100	Target only
011	Secondary Controller
001	Controller only

24.3.4. Command Queue Port Register

Name: I3CxCMDQUE
Offset: 0x7A900C

Bit	31	30	29	28	27	26	25	24
	COMMAND[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COMMAND[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COMMAND[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COMMAND[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – COMMAND[31:0] 32-bit Command bits
Mapped to the Command Queue.

24.3.5. Response Queue Port Register

Name: I3CxRESPQUE
Offset: 0x7A9010

Bit	31	30	29	28	27	26	25	24
	RESPONSE[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RESPONSE[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RESPONSE[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RESPONSE[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – RESPONSE[31:0] 32-bit Response bits
Mapped to the Response Queue.

24.3.6. Transfer Data Register

Name: I3CxTXRXDATA
Offset: 0x7A9014

Bit	31	30	29	28	27	26	25	24
	TXRXDATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TXRXDATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TXRXDATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXRXDATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TXRXDATA[31:0] Transfer Data bits
Mapped to the Transmit and Receive Buffer. Writes data to the TX Buffer and reads data from the RX Buffer.

24.3.7. In-Band Interrupt Queue Register

Name: I3CxIBIQUE
Offset: 0x7A9018

Bit	31	30	29	28	27	26	25	24
	IBISTADAT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	IBISTADAT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	IBISTADAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	IBISTADAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – IBISTADAT[31:0]
Mapped to the IBI Status and Data Queue bits (Controller mode only).

24.3.8. Queue Threshold Control Register

Name: I3CxQUETHLDCON
Offset: 0x7A901C

Bit	31	30	29	28	27	26	25	24
	IBISTATHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bit	23	22	21	20	19	18	17	16
	IBIDATTHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RESPBUFTHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bit	7	6	5	4	3	2	1	0
	CMDEBTHLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – IBISTATHLD[7:0] In-Band Interrupt Status Threshold Value bits
This field controls the number of IBI status entries (or greater) in the IBI queue that trigger the IBITHLDSTA interrupt. The valid range is 0 to 7. A value of 0 sets the threshold for 1 entry, and a value of N sets the threshold for N+1 entries. Each IBI status entry can represent the complete (IBI payload byte size ≤ 4* IBIDATTHLD) IBI payload or a segment (IBI payload byte size > 4* IBIDATTHLD) of the IBI payload.

Bits 23:16 – IBIDATTHLD[7:0] IBI Data Threshold Value bits
This field represents the IBI data segment size in DWORD (4 bytes). The minimum supported segment size is 1 (4 bytes), and the maximum supported size is 31. The IBIDATTHLD field enables the slicing of the incoming IBI data and generates individual status, promoting the cut-through operation in reading out the IBI data.

Bits 15:8 – RESPBUFTHLD[7:0] Response Buffer Threshold Value bits
Controls the number of entries (or greater) in the Response Queue that trigger the RESPQSTA interrupt. The valid range is 0 to 3. The software programs only valid values. A value of 0 sets the threshold for 1 entry, and a value of N sets the threshold for N+1 entries.

Bits 7:0 – CMDEBTHLD[7:0] Command Buffer Empty Threshold Value bits
Controls the number of empty locations (or greater) in the Command Queue that trigger the CMDQSTA interrupt. The valid range is 0 to 7. A value of N ranging from 1 to 7 sets the threshold to N empty locations, and a value of 0 sets the threshold to indicate that the queue is completely empty.

24.3.9. Data Buffer Threshold Control Register

Name: I3CxBUFTHLD
Offset: 0x7A9020

Bit	31	30	29	28	27	26	25	24
						RXSTARTTHLD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	1

Bit	23	22	21	20	19	18	17	16
						TXSTARTTHLD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	1

Bit	15	14	13	12	11	10	9	8
						RXTHLD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	1

Bit	7	6	5	4	3	2	1	0
						TXTHLD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	1

Bits 26:24 – RXSTARTTHLD[2:0] Receive Start Threshold Value bits

Value	Description
100	Controller initiates read transfer/Target ACK's write request whenever RX FIFO has a minimum of 32 empty locations.
011	Controller initiates read transfer/Target ACK's write request whenever RX FIFO has a minimum of 16 empty locations.
010	Controller initiates read transfer/Target ACK's write request whenever RX FIFO has a minimum of 8 empty locations.
001	Controller initiates read transfer/Target ACK's write request whenever RX FIFO has a minimum of 4 empty locations.
000	Controller initiates read transfer/Target ACK's write request whenever RX FIFO has at least one empty location.

Bits 18:16 – TXSTARTTHLD[2:0] Transfer Start Threshold Value bits

Value	Description
100	Controller initiates write transfer/Target ACK's read request whenever TX FIFO has at least 32 entries.
011	Controller initiates write transfer/Target ACK's read request whenever TX FIFO has at least 16 entries.
010	Controller initiates write transfer/Target ACK's read request whenever TX FIFO has at least 8 entries.
001	Controller initiates write transfer/Target ACK's read request whenever TX FIFO has at least 4 entries.
000	Controller initiates write transfer/Target ACK's read request whenever TX FIFO has at least 1 entry.

Bits 10:8 – RXTHLD[2:0] Receive Buffer Threshold Value bits

Value	Description
100	Whenever RX FIFO has a minimum of 32 entries, it triggers the RXTHLDSTA interrupt.
011	Whenever RX FIFO has a minimum of 16 entries, it triggers the RXTHLDSTA interrupt.
010	Whenever RX FIFO has a minimum of 8 entries, it triggers the RXTHLDSTA interrupt.
001	Whenever RX FIFO has a minimum of 4 entries, it triggers the RXTHLDSTA interrupt.
000	Whenever RX FIFO has at least one entry, it triggers the RXTHLDSTA interrupt.

Bits 2:0 – TXTHLD[2:0] Transmit Buffer Threshold Value bits

Value	Description
100	Whenever TX FIFO has a minimum of 32 empty locations, it triggers the TXTHLDSTA interrupt.
011	Whenever TX FIFO has a minimum of 16 empty locations, it triggers the TXTHLDSTA interrupt.
010	Whenever TX FIFO has a minimum of 8 empty locations, it triggers the TXTHLDSTA interrupt.
001	Whenever TX FIFO has a minimum of 4 empty locations, it triggers the TXTHLDSTA interrupt.
000	Whenever TX FIFO has at least one empty location, it triggers the TXTHLDSTA interrupt.

24.3.10. Controller Mode IBI Queue Control Register

Name: I3CxIBIQNOTIFY
Offset: 0x7A9024

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					SIRREJ		MRREJ	HJREJ
Access					R/W		R/W	R/W
Reset					0		0	0

Bit 3 – SIRREJ Notify Rejected Target Interrupt Request Control bit

Value	Description
1	Notify SIR rejected enabled.
0	Notify SIR rejected disabled.

Bit 1 – MRREJ Notify Rejected Controller Request Control bit

Value	Description
100	Notify Controller request rejected enable.
011	Notify Controller request rejected disable.

Bit 0 – HJREJ Notify Rejected Hot-Join Control bit

Value	Description
1	Notify Hot-Join rejected enable.
0	Notify Hot-Join rejected disable.

24.3.11. Reset Control Register

Name: I3CxRSTCON
Offset: 0x7A9034

Bit	31	30	29	28	27	26	25	24
	BUSRST	RSTTYP[2:0]						
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			IBIQRST	RXFIFORST	TXFIFORST	RESPQRST	CMDQRST	SOFTRST
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 31 – BUSRST Bus Reset bit (Controller mode only)

Value	Description
1	Bus Reset pattern generation based on Bus Reset type selection.
0	Bus Reset pattern generation is completed.

Bits 30:28 – RSTTYP[2:0] Bus Reset type bit

Value	Description
110	SCL LOW RESET pattern
001	Target Reset pattern
000	Exit pattern

Bit 5 – IBIQRST IBI Queue Software Reset bit (Controller mode only)

Value	Description
1	Generate IBI Queue Reset.
0	IBI Queue Reset complete.

Bit 4 – RXFIFORST Receive Buffer Software Reset bit

Value	Description
1	Generate Receive Buffer Reset.
0	Receive Buffer Reset complete.

Bit 3 – TXFIFORST Transmit Buffer Software Reset bit

Value	Description
1	Generate Transmit Buffer Reset.
0	Transmit Buffer Reset complete.

Bit 2 – RESPQRST Response Queue Software Reset bit

Value	Description
1	Generate Response Queue Reset.
0	Response Queue Reset complete.

Bit 1 – CMDQRST Command Queue Software Reset bit

Value	Description
1	Generate Command Queue Reset.
0	Command Queue Reset complete.

Bit 0 – SOFTRST Core Software Reset bit

Value	Description
1	Generate software (Queues, buffers, programmable registers, pipelining data, state machine to Idle state) Reset.
0	Command Queue Reset complete.

24.3.12. Target Mode Event Status Register

Name: I3CxTGTESTA
Offset: 0x7A9038

Notes:
1. If this field is not set to 0 by the Target application, it can be set or cleared by the I^3C Controller through ENEC or DISEC CCCs. Once disabled by software, CCCs do not have any effect on this field.
2. Usually, this bit is set or cleared by the I^3C Controller through ENEC or DISEC CCC.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	MWLSTA	MRLSTA	ACTSTA[1:0]		HJINTEN		MRINTEN	
Access	R	R/W1C	R	R	R/W		R/W	
Reset	0	0	0	0	1		1	

Bit 7 – MWLSTA MWL Updated Status bit
This bit is set when SETMWL CCC is received by the Target. This status can be cleared by writing 1'b1 to this field after reading the updated MWL.

Bit 6 – MRLSTA MRL Updated Status bit
This bit is set when the SETMRL CCC is received by the Target. This status can be cleared by writing 1'b1 to this field after reading the updated MRL.

Bits 5:4 – ACTSTA[1:0] Activity State Status bits

Value	Description
11	ENTAS3
10	ENTAS2
01	ENTAS1
00	ENTAS0

Bit 3 – HJINTEN Hot-Join Interrupt Enable bit(1)

Value	Description
1	Enable Hot-Join capability.
0	Disable Hot-Join capability.

Bit 1 – MRINTEN Commander Request Enable bit(2)

Value	Description
1	Enable Controller request capability.
0	Disable Controller request capability.

Bit 1 – SIRINTEN Target Interrupt Request Enable bit(2)

Value	Description
1	Enable Target interrupt request capability.
0	Disable Target interrupt request capability.

24.3.13. Interrupt Status Register

Name: I3CxINTSTA
Offset: 0x7A903C

Note:

1. This interrupt is provided for the user/application (current active Controller) to decide whether the new active Controller has taken over the ownership after the bus handover was completed successfully to move to Target mode. This interrupt is also used to reset the Reset action configured by the RSTACT CCC.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			EXTCMDTXTHLDSTA	EXTCMDSTA	SDARELSTA	GRPADDRSTA	TRSTPATSTA	STARTSTA
Access			R	R	R/W1C	R/W1C	R/W1C	R/W1C
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BUSRSTSTA		BUSOWNSTA	IBIUPDSTA	READREQSTA	DEFTGTSTA	TRANSERRSTA	DYNADDRSTA
Access	R/W1C		R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		CCCUPDSTA	TRANSABTSTA	RESPQSTA	CMDQSTA	IBITHLDSTA	RXTHLDSTA	TXTHLDSTA
Access		R/W1C	R/W1C	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit 21 – EXTCMDTXTHLDSTA I3CxEXTCMDy Transmit Buffer Threshold Status bit (Target mode only)

Value	Description
1	The number of empty locations in any I3CxEXTCMDy Transmit Buffer is greater than or equal to the threshold value specified by I3CxECRTCON [CMDTXBUFTHLD].
0	Cleared automatically when the number of empty locations in all the I3CxEXTCMDy Transmit Buffers is less than the threshold value specified.

Bit 20 – EXTCMDSTA I3CxEXTCMDy Has Finished Status bit (Target mode only)

Value	Description
1	Read transfer has finished in the I3CxEXTCMDy.
0	Cleared automatically when the register I3CxECMDVLDSTA is read.

Bit 19 – SDARELSTA SDA Released from Stuck State Status bit (Target mode only)
This interrupt is generated if the SCL was not changed for the prescribed amount of time defined by the I3CxRELSDATIM register. This bit can be cleared by writing 1'b1.

Bit 18 – GRPADDRSTA Group Address Assigned Status bit (Target mode only)
This interrupt is generated if one of the device's Group Addresses is assigned through SETGRPA CCC or reset through RSTGRPA CCC. This bit can be cleared by writing 1'b1.

Bit 17 – TRSTPATSTA Target Reset Pattern Detection Interrupt bit (Target mode only)
This interrupt is generated when the Target Reset pattern is detected.

Bit 16 – STARTSTA START Detection Interrupt bit (Target mode only)
This field indicates START detected in Target mode of operation. This bit can be cleared by writing 1'b1.**(1)**

Bit 15 – BUSRSTSTA Bus Reset Pattern Generation Done Status bit
This interrupt is generated when the requested Bus Reset pattern generation is completed. This bit can be cleared by writing '1b1'.

Bit 13 – BUSOWNSTA
This interrupt is set when the role of the Controller changes from being a Controller to a Target, or vice versa. This bit can be cleared by writing 1'b1.

Bit 12 – IBIUPDSTA IBI Status is Updated bit (Target mode only)
Indicates that the IBI request initiated through the SIR request register is addressed and the status is updated.

Bit 11 – READREQSTA Read Request Received bit (Target mode only)
Read request received from the current Controller when CMDQ is empty. This bit can be cleared by writing 1'b1.

Bit 10 – DEFTGTSTA Define Target CCC Received Status bit
This interrupt is generated if the DEFSLV CCC is received. This bit can be cleared by writing 1'b1.

Bit 9 – TRANSERRSTA Transfer Error Status bit

Bit 8 – DYNADDRSTA Dynamic Address Assigned Status bit (Target mode only)
This interrupt is generated if the device's Dynamic Address is assigned through SETDASA, SETAASA, SETNEWDA or ENTDAA CCC. This bit can be cleared by writing 1'b1.

Bit 6 – CCCUPDSTA CCC Table Updated Status bit (Target mode only)
This interrupt is generated if any of the CCC registers are updated by the I^3C Controller through CCC commands. This interrupt can be cleared by writing 1'b1.

Bit 5 – TRANSABTSTA Transfer Abort Status bit (Controller mode only)
This interrupt is generated if the transfer is aborted. This interrupt can be cleared by writing 1'b1.

Bit 4 – RESPQSTA Response Queue Ready Status bit
This interrupt is generated when the number of entries in the Response Queue is greater than or equal to the threshold value specified by the I3CxQUETHLDCON[RESPBUFTHLD] register. This interrupt is cleared automatically when the number of entries in the Response Buffer is less than the threshold value specified.

Bit 3 – CMDQSTA Command Queue Ready bit
This interrupt is generated when the number of empty locations in the Command Queue is greater than or equal to the threshold value specified by the I3CxQUETHLDCON[CMDEBTHLD] register. This interrupt is cleared automatically when the number of empty locations in the Command Buffer is less than the specified threshold value.

Bit 2 – IBITHLDSTA IBI Buffer Threshold Status bit
This interrupt is generated when the number of entries in the IBI Buffer is greater than or equal to the threshold value specified by the I3CxQUETHLDCON[IBISTATHLD] register. This interrupt is

cleared automatically when the number of entries in the IBI Buffer is less than the specified threshold value.

Bit 1 – RXTHLDSTA Receive Buffer Threshold Status bit
This interrupt is generated when the number of entries in the Receive Buffer is greater than or equal to the threshold value specified by the I3CxBUFTHLD [RXBUF] register. This interrupt is cleared automatically when the number of entries in the Receive Buffer is less than the threshold value specified.

Bit 0 – TXTHLDSTA Transmit Buffer Threshold Status bit
This interrupt is generated when the number of empty locations in the Transmit Buffer is greater than or equal to the threshold value specified by the I3CxBUFTHLD [TXBUF] register. This interrupt is cleared automatically when the number of empty locations in the Transmit Buffer is less than the specified threshold value.

24.3.14. Interrupt Status Enable Register

Name: I3CxINTSTACON
Offset: 0x7A9040

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			EXTCMDTXTHLDSTAEN	EXTCMDSTAEN	SDARELSTAEN	GRPADDRSTAEN	TRSTPATSTAEN	STARTSTAEN
Access			R	R	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BUSRSTSTAEN		BUSOWNSTAEN	IBIUPDSTAEN	READREQSTAEN	DEFTGTSTAEN	TRANSERRSTAEN	DYNADDRSTAEN
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		CCCUPDSTAEN	TRANSABTSTAEN	RESPQSTAEN	CMDQSTAEN	IBITHLDSTAEN	RXTHLDSTAEN	TXTHLDSTAEN
Access		R/W	R/W	R/W	R/W	R	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 21 – EXTCMDTXTHLDSTAEN I3CxEXTCMDy Transmit Buffer Threshold Status Enable bit
This field is used in Target mode of operation.

Bit 20 – EXTCMDSTAEN I3CxEXTCMDy has Finished Status Enable bit
This field is used in Target mode of operation.

Bit 19 – SDARELSTAEN SDA Released from Stuck State Status Enable bit
This field is used in Target mode of operation.

Bit 18 – GRPADDRSTAEN Group Address Assigned Status Enable bit
This field is used only in Target mode of operation.

Bit 17 – TRSTPATSTAEN Target Reset Pattern Detection Status Enable bit
This field is used only in Target mode of operation.

Bit 16 – STARTSTAEN START Detection Status Enable bit
This field is used only in Target mode of operation.

Bit 15 – BUSRSTSTAEN Bus Reset Pattern Generation Done Status Enable bit
This field is used only in Controller mode of operation.

Bit 13 – BUSOWNSTAEN Bus Owner Updated Status Enable bit

Bit 12 – IBIUPDSTAEN IBI Updated Status Enable bit
This field is used in Target mode of operation.

Bit 11 – READREQSTAEN Read Request Received Status Enable bit
This field is used in Target mode of operation.

Bit 10 – DEFTGTSTAEN Define Target CCC Received Status Enable bit

Bit 9 – TRANSERRSTAEN Transfer Error Status Enable bit

Bit 8 – DYNADDRSTAEN Dynamic Address Assigned Status Enable bit
This field is used in Target mode of operation.

Bit 6 – CCCUPDSTAEN CCC Table Updated Status Enable bit
This field is used in Target mode of operation.

Bit 5 – TRANSABTSTAEN Transfer Abort Status Enable bit
This field is used in Controller mode of operation.

Bit 4 – RESPQSTAEN Response Queue Ready Status Enable bit

Bit 3 – CMDQSTAEN Command Queue Ready Status Enable bit
This field is used in Controller mode of operation.

Bit 2 – IBITHLDSTAEN IBI Buffer Threshold Status Enable bit
This field is used in Controller mode of operation.

Bit 1 – RXTHLDSTAEN Receive Buffer Threshold Status Enable bit

Bit 0 – TXTHLDSTAEN Transmit Buffer Threshold Status Enable bit

24.3.15. Interrupt Signal Enable Register

Name: I3CxINTCON
Offset: 0x7A9044

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			EXTCMDTXTHLDINTEN	EXTCMDINTEN	SDARELINTEN	GRPADDRINTEN	TRSTPATINTEN	STARTINTEN
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BUSRSTINTEN		BUSOWNINTEN	IBIUPDINTEN	READREQINTEN	DEFTGTINTEN	TRANSERRINTEN	DYNADDRINTEN
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		CCCUPDINTEN	TRANSABTINTEN	RESPQINTEN	CMDQINTEN	IBITHLDINTEN	RXTHLDINTEN	TXTHLDINTEN
Access		R/W	R/W	R/W	R/W	R	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 21 – EXTCMDTXTHLDINTEN I3CxEXTCMDy Transmit Buffer Threshold Signal Enable bit
This field is used in Target mode of operation.

Bit 20 – EXTCMDINTEN I3CxEXTCMDy has Finished Signal Enable bit
This field is used in Target mode of operation.

Bit 19 – SDARELINTEN SDA Released from Stuck State Signal Enable bit
This field is used in Target mode of operation.

Bit 18 – GRPADDRINTEN Group Address Assigned Signal Enable bit
This field is used in Target mode of operation.

Bit 17 – TRSTPATINTEN Target Reset Pattern Detection Signal Enable bit
This field is used in Target mode of operation.

Bit 16 – STARTINTEN START Detection Signal Enable bit
This field is used in Target mode of operation.

Bit 15 – BUSRSTINTEN Bus Reset Pattern Generation Done Signal Enable bit
This field is used in Controller mode of operation.

Bit 13 – BUSOWNINTEN Bus Owner Updated Signal Enable bit

Bit 12 – IBIUPDINTEN IBI Updated Signal Enable bit
This field is used in Target mode of operation.

Bit 11 – READREQINTEN Read Request Received Signal Enable bit
This field is used in Target mode of operation.

Bit 10 – DEFTGTINTEN Define Target CCC Received Signal Enable bit

Bit 9 – TRANSERRINTEN Transfer Error Signal Enable bit

Bit 8 – DYNADDRINTEN Dynamic Address Assigned Signal Enable bit
This field is used in Target mode of operation.

Bit 6 – CCCUPDINTEN CCC Table Updated Signal Enable bit
This field is used in Target mode of operation.

Bit 5 – TRANSABTINTEN Transfer Abort Signal Enable bit
This field is used in Controller mode of operation.

Bit 4 – RESPQINTEN Response Queue Ready Signal Enable bit

Bit 3 – CMDQINTEN Command Queue Ready Signal Enable bit

Bit 2 – IBITHLDINTEN IBI Buffer Threshold Signal Enable bit
This field is used in Controller mode of operation.

Bit 1 – RXTHLDINTEN Receive Buffer Threshold Signal Enable bit

Bit 0 – TXTHLDINTEN Transmit Buffer Threshold Signal Enable bit

24.3.16. Interrupt Force Enable Register

Name: I3C1INTFORCE
Offset: 0x7A9048

Individual interrupts can be forcefully triggered if the corresponding Force Enable bit is set, provided the corresponding bit in the I3CxINTSTACON register is set.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
			EXTCMDTXTHLDFRC	EXTCMDFRC	SDARELFRC	GRPADDRFRC	TRSTPATFRC	STARTFRC
Access			W	W	W	W	W	W
Reset			0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	BUSRSTFRC		BUSOWNFRC	IBIUPDFRC	READREQFRC	DEFTGTFRC	TRANSERRFRC	DYNADDRFRC
Access	W		W	W	W	W	W	W
Reset	0		0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
		CCCUPDFRC	TRANSABTFRC	RESPQFRC	CMDQFRC	IBITHLDFRC	RXTHLDFRC	TXTHLDFRC
Access		W	W	W	W	W	W	W
Reset		0	0	0	0	0	0	0

Bit 21 – EXTCMDTXTHLDFRC I3CxEXTCMDy Transmit Buffer Threshold Force Enable bit
This field is used in Target mode of operation.

Bit 20 – EXTCMDFRC I3CxEXTCMDy has Finished Force Enable bit
This field is used in Target mode of operation.

Bit 19 – SDARELFRC SDA Released from Stuck State Force Enable bit
This field is used in Target mode of operation.

Bit 18 – GRPADDRFRC Group Address Assigned Force Enable bit
This field is used in Target mode of operation.

Bit 17 – TRSTPATFRC Target Reset Pattern Detection Interrupt Force Enable bit
This field is used in Target mode of operation.

Bit 16 – STARTFRC START Detection Force Enable bit
This field is used in Target mode of operation.

Bit 15 – BUSRSTFRC Bus Reset Pattern Generation Done Force Enable bit
This field is used in Controller mode of operation.

Bit 13 – BUSOWNFRC Bus Owner Updated Force Enable bit

Bit 12 – IBIUPDFRC IBI Updated Force Enable bit
This field is used in Target mode of operation.

Bit 11 – READREQFRC Read Request Received Force Enable bit
This field is used in Target mode of operation.

Bit 10 – DEFTGTFRC Define Target CCC Received Force Enable bit

Bit 9 – TRANSERRFRC Transfer Error Force Enable bit

Bit 8 – DYNADDRFRC Dynamic Address Assigned Force Enable bit
This field is used in Target mode of operation.

Bit 6 – CCCUPDFRC CCC Table Updated Force Enable bit
This field is used in Target mode of operation.

Bit 5 – TRANSABTFRC Transfer Abort Force Enable bit
This field is used in Controller mode of operation.

Bit 4 – RESPQFRC Response Queue Ready Force Enable bit

Bit 3 – CMDQFRC Command Queue Ready Force Enable bit

Bit 2 – IBITHLDFRC IBI Buffer Threshold Force Enable bit
This field is used in Controller mode of operation.

Bit 1 – RXTHLDFRC Receive Buffer Threshold Force Enable bit

Bit 0 – TXTHLDFRC Transmit Buffer Threshold Force Enable bit

24.3.17. Queue Status Level Register

Name: I3CxQLEVEL
Offset: 0x7A904C

Bit	31	30	29	28	27	26	25	24
				IBISTATCNT[4:0]				
Access				R	R	R	R	R
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	IBIBUF[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RESPBUF[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMDQEMTLOC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	0

Bits 28:24 – IBISTATCNT[4:0] IBI Buffer Status Count bits
This field is used in Controller mode of operation.

Bits 23:16 – IBIBUF[7:0] IBI Buffer Level Value bits
Contains the number of valid IBI data entries in the IBI data buffer. This field is used in Controller mode of operation.

Bits 15:8 – RESPBUF[7:0] Response Buffer Level Value bits
Contains the number of valid data entries in the Response Buffer.

Bits 7:0 – CMDQEMTLOC[7:0] Command Queue Empty Locations bits
Contains the number of empty locations in the Command Buffer.

24.3.18. Data Buffer Status Level Register

Name: I3CxBUFLEVEL
Offset: 0x7A9050

Bit	31	30	29	28	27	26	25	24
				RXBUFLVL[12:8]				
Access				R	R	R	R	R
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RXBUFLVL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	TXBUFEMTLVL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	1	0	0	0	0	0

Bits 28:16 – RXBUFLVL[12:0] Receive Buffer Level Value bits
Contains the number of valid data entries in the Receive Buffer.

Bits 7:0 – TXBUFEMTLVL[7:0] Transmit Buffer Empty Level Value bits
Contains the number of empty locations in the Transmit Buffer.

24.3.19. Present State Register

Name: I3CxSTATE
Offset: 0x7A9054

Notes:
1. This bit is set when all the queues (Command, Response, IBI) and buffers (Transmit and Receive) are empty, along with the Controller State machine being in an Idle state.
2. A software Reset will not have any effect on this field.

Bit	31	30	29	28	27	26	25	24
				IDLESTATE	CMDTID[3:0]			
Access				R	R	R	R	R
Reset				1	0	0	0	0

Bit	23	22	21	20	19	18	17	16
			CMTFRSTAT[5:0]					
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			CMTFRTYP[5:0]					
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
						CURRCNTRL	SDALEVEL	SCLLEVEL
Access						R	R	R
Reset						1	0	0

Bit 28 – IDLESTATE Controller Mode Idle bit

Value	Description
1	Controller is in IDLE State(1) .
0	Controller is not in IDLE State.

Bits 27:24 – CMDTID[3:0]
This field reflects the Transaction ID of the current executing command.

Bits 21:16 – CMTFRSTAT[5:0] Current Controller Transfer State Status (Controller mode only)

Value	Description
010011	Halt state
010010	Clock Extension state
010001	HDR-DDR CRC Data Generation/Receive state
010000	In-Band Interrupt Auto-Disable state
001111	In-Band Interrupt (SIR) Read Data state
001110	Read Data Transfer state
001101	Write Data Transfer state
001100	HDR Command Generation state
001011	CCC Byte Generation state
001000	Target Address Generation State
000111	Dynamic Address Assignment state
000110	Broadcast Read Address Header(7'h7E,R) Generation state
000101	Broadcast Write Address Header(7'h7E,W) Generation state

Value	Description
000100	START Hold Generation for the Target Initiated START state
000011	Stop Generation state
000010	Restart Generation state
000001	Start Generation state
000000	Controller is Idle

Bits 13:8 – CMTFRTYP[5:0] Current Transfer Type Status bits

Value	Description
	In Controller mode of operation:
010001	Reserved
010000	Reserved
001111	Halt state (Controller is in Halt state, waiting for the application to resume through DEVICE_CTRL register.)
001110	Servicing In-Band Interrupt Transfer
001101	Private HDR Double-Data Rate(DDR) read transfer
001100	Private HDR Double-Data Rate(DDR) write transfer
001011-001010	Reserved
001001	Private I²C SDR read transfer
001000	Private I²C SDR write transfer
000111	Private I³C SDR read transfer
000110	Private I³C SDR write transfer
000101	SETDASA Address assignment transfer
000100	ENTDAA Address assignment transfer
000011	Directed CCC read transfer
000010	Directed CCC write transfer
000001	Broadcast CCC write transfer
000000	IDLE (Controller is in Idle state, waiting for commands from application or Target initiated In-Band Interrupt)
	In Target mode of operation:
000110	Target controller in Halt state, waiting for resume from application.
000101	Controller read transfer ongoing.
000100	Read Data Prefetch state
000011	Controller write transfer ongoing.
0000010	IBI Transfer state
000001	Hot-Join Transfer state
000000	IDLE (Controller is in Idle state.)

Bit 2 – CURRCNTRL Current Controller State Status bit(2)

Value	Description
1	Controller is current Controller.
0	Controller is not current Controller.

Bit 1 – SDALEVEL

This bit reflects the status of the SDA pin.

Bit 0 – SCLLEVEL

This bit reflects the status of the SCL pin.

24.3.20. Target Mode Device Operating Status Register (Target mode only)

Name: I3CxTGTCCCSTAT
Offset: 0x7A9058

Notes:
1. Setting this bit requires a few SCL clocks after the Target has NACKed the read transfer.
2. This bit is set if the Target terminates a read transfer due to the unavailability of data in the transmit buffer.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			FRMERR	BUFFNTAVAIL	DATNTRDY	OVFLWERR	TGTBUSY	UDFLWERR
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ACTIMOD[1:0]		PROTOERR		PNDINGINT[3:0]			
Access	R	R	R		R	R	R	R
Reset	0	0	0		0	0	0	0

Bit 13 – FRMERR Frame Error bit

Value	Description
1	Frame error in HDR-DDR (private write)
0	Cleared when Controller reads status through GETSTATUS CCC

Bit 12 – BUFFNTAVAIL Buffer Not Available bit

Value	Description
1	Private write request from Controller is NACKed because the RX Buffer does not have the I3C1BUFTHLD [RXSTART] number of empty locations or the Response Buffer is full.
0	Cleared when the Controller reads the status through the GETSTATUS CCC. In SDR mode, it is also cleared upon space becoming available in the buffer and the successful completion of the next write transfer.

Bit 11 – DATNTRDY Data Not Ready bit

Value	Description
1	Private write request from Controller is NACKed because the Command FIFO is empty, the Transmit FIFO threshold is not met or the Response FIFO is full[(1)] .
0	Cleared when Controller reads status through GETSTATUS CCC

Bit 10 – OVFLWERR Overflow Error bit

Value	Description
1	Overflow error condition detected during Controller write transfer.
0	Cleared when the Controller reads status through GETSTATUS CCC.

Bit 9 – TGTBUSY Target Busy bit

Value	Description
1	Change is made by the current Controller into the MRL register or occurrence of any error.
0	Target application resumes the target operation by setting the I3CxCTRL[RESUME] bit.

Bit 8 – UDFLWERR Underflow Error bit

Value	Description
1	Underflow Error during private Controller read transfer(2).
0	Cleared when the Controller reads the status through GETSTATUS CCC.

Bits 7:6 – ACTIMOD[1:0] Activity Mode bits

This bit reflects the activity mode of the Target.

Bit 5 – PROTOERR Protocol Error bit

Value	Description
1	Parity/CRC error during data transfer write.
0	No error.

Bits 3:0 – PNDINGINT[3:0] Pending Interrupt bit

Value	Description
1	Pending interrupt.
0	No pending interrupt.

24.3.21. Controller Mode Device Address Table Pointer Register

Name: I3CxADDRTABPTR
Offset: 0x7A905C

Bit	31	30	29	28	27	26	25	24
	ADDRTABDPTH[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	ADDRTABDPTH[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	1	0	0	1	0

Bit	15	14	13	12	11	10	9	8
	ADDRTABSTRTADDR[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	1	1

Bit	7	6	5	4	3	2	1	0
	ADDRTABSTRTADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	0	0	0	0	0	0	0

Bits 31:16 – ADDRTABDPTH[15:0] Depth of Device Address Table bits

Bits 15:0 – ADDRTABSTRTADDR[15:0] Start Address of Device Address Table bits

24.3.22. Controller Mode Device Characteristics Table Pointer Register

Name: I3CxCHARTABPTR
Offset: 0x7A9060

Notes:
1. The first winning device information is stored in the device characteristics table index 0, the second winning device information in index 1, and so on.
2. In the non-active Controller, this field is always read-only.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	23	22	21	20	19	18	17	16
	CURRCHTABINDX[4:0]					CHTABDPTH[6:4]		
Access	R/W	R/W	R/W	R/W	R/W	R	R	R
Reset	0	1	0	0	0	1	0	0
Bit	15	14	13	12	11	10	9	8
	CHTABDPTH[3:0]				CHTABSTRTADDR[11:8]			
Access	R	R	R	R	R	R	R	R
Reset	1	0	0	0	0	0	1	0
Bit	7	6	5	4	3	2	1	0
	CHTABSTRTADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 23:19 – CURRCHTABINDX[4:0] Current Index of Device Characteristics Table[1,2] bits

Bits 18:12 – CHTABDPTH[6:0] Depth of Device Characteristics Table bits

Bits 11:0 – CHTABSTRTADDR[11:0] Start Address of Device Characteristics Table bits

24.3.23. Controller Mode Vendor Specific Pointer Register

Name: I3CxVENDSPCPTR
Offset: 0x7A906C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VENDSTRTADDR[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VENDSTRTADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	0	1	1	0	0	0	0

Bits 15:0 – VENDSTRTADDR[15:0] Start Address of Vendor Specific Registers bits (Controller mode only)

24.3.24. Target Mode Manufacturer ID Register

Name: I3CxTGTMIPIID
Offset: 0x7A9070

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	MIPIMFGID[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MIPIMFGID[6:0]							PROVID
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:1 – MIPIMFGID[14:0] Specifies the MIPI Manufacturer ID bits

Bit 0 – PROVID Specifies the Provisional ID Type Selector bits

24.3.25. Target Mode Provisional ID Register

Name: I3CxTGTPID
Offset: 0x7A9074

Bit	31	30	29	28	27	26	25	24
	PARTID[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PARTID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	INSTID[3:0]				PIDDCR[11:8]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PIDDCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – PARTID[15:0] Specifies the Part ID of Module (PID[31:16]) bits

Bits 15:12 – INSTID[3:0] This field is used to program the instance ID of the Target bits

Bits 11:0 – PIDDCR[11:0] Specifies the additional 12-bit ID of module (PID[11:0]) bits

24.3.26. Target Mode Characteristic Register

Name: I3CxTGTCON
Offset: 0x7A9078

Notes:
1. This field is set to 1 by default, Controller capable. If the application chooses to operate as "Target only" through programming, then the Device Role can be overwritten as Target (BCR[7:6] = 2'b00).
2. Programming this field to 0 does not disable the HDR feature itself. This bit can be modified by the application if it does not want to advertise Target HDR capability to the Controller.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	HDRCAP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8
	DCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DEVICEROLE[1:0]		SDRHDR	BRGID	OFFLINECAP	IBIPYLD	IBIREQCAP	MAXDATSPDLMT
Access	R/W	R/W	R/W	R	R	R	R	R/W
Reset	0	1	1	1	0	1	1	1

Bits 23:16 – HDRCAP[7:0] I3C Device HDR Capability Register Value bits

Value	Description
11111111 – 00000010	Reserved
00000001	HDR- DDR capable
00000000	Reserved

Bits 15:8 – DCR[7:0] I3C Device Characteristic Value bits

Bits 7:6 – DEVICEROLE[1:0] Device Role Field in Bus Characteristic Register (BCR[7:6])(1) bits

Bit 5 – SDRHDR SDR Only or SDR and HDR Capable Field in Bus (BCR[5])(2) bit

Value	Description
1	HDR DDR capable.
0	SDR capable.

Bit 4 – BRGID Bridge Identifier Field in Bus Characteristic Register (BCR[4]) bit

Value	Description
1	Virtual Target, or exposes other downstream device(s).
0	Not a Virtual Target and does not expose any other downstream device(s).

Bit 3 – OFFLINECAP Offline Capable Field in Bus Characteristic Register (BCR[3]) bit

Value	Description
1	Target will not always respond to I^3C Bus commands.
0	Target will always respond to I^3C Bus commands.

Bit 2 – IBIPYLD IBI Payload Field in Bus Characteristic Register bit (BCR[2])

Value	Description
1	One data byte (MDB) shall follow the accepted IBI, and additional data bytes may follow.
0	No data bytes follow the accepted IBI.

Bit 1 – IBIREQCAP IBI Request Capable Field in Bus Characteristic Register bit (BCR[1])

Value	Description
1	Capable
0	Not capable.

Bit 0 – MAXDATSPDLMT Max Data Speed Limitation Field in Bus Characteristic Register bit (BCR[0])

Value	Description
1	Target returns the data in the I3CxCLTDATSPD and I3CxCLTIBIPYLD registers in response to the GETMXDS CCC sent by the Controller.
0	Target NACKs the GETMXDS CCC sent by the Controller.

24.3.27. Target Mode Max Write/Read Length Register

Name: I3CxTGTMAXLEN
Offset: 0x7A907C

Bit	31	30	29	28	27	26	25	24
	MRL[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	MRL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	MWL[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	MWL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	1	1	1	1	1	1	1	1

Bits 31:16 – MRL[15:0] I^3C Device Max Read Length bits

Bits 15:0 – MWL[15:0] I^3C Device Max Write Length bits

24.3.28. Maximum supported IBI Payload Size Register

Name: I3CxTGTIBIPYLD
Offset: 0x7A9080

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	MAXRDTIME[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MAXRDTIME[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MAXRDTIME[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bits 23:0 – MAXRDTIME[23:0] Specifies the maximum read turnaround time (in microseconds (μs)) of the Target.

24.3.29. Target Mode Maximum Data Speed Register

Name: I3CxTGTDATSPD
Offset: 0x7A9084

Bit	31	30	29	28	27	26	25	24
						SETACTSTATE	ACTSTATE[1:0]	
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	23	22	21	20	19	18	17	16
	Reserved[3:0]				STPPRMIT	CLKDATTIME[2:0]		
Access	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						MAXRDSPD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
					DEFBYT	MAXWRSPD[2:0]		
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 26 – SETACTSTATE

Value	Description
1	The Active Controller should set the bus to the activity state indicated by bits 25:24 before handing off to this device.
0	The Active Controller should not set the bus to any activity state before handing off to this device.

Bits 25:24 – ACTSTATE[1:0]

Value	Description
11	Acts according to Activity State 3.
10	Acts according to Activity State 2.
01	Acts according to Activity State 1.
00	Acts according to Activity State 0.

Bits 23:20 – Reserved[3:0]

Bit 19 – STPPRMIT

Specifies the stop support between write and read.

Value	Description
1	The Target permits the write and read to be split by a stop.
0	Stop will cancel the read.

Bits 18:16 – CLKDATTIME[2:0]

Specifies the clock-to-data turnaround time of the Target.

Value	Description
0111-0101	Reserved
0100	12 ns

Value	Description
0011	11 ns
0010	10 ns
0001	9 ns
0000	8 ns

Bits 10:8 – MAXRDSPD[2:0]

Specifies the maximum sustained data rate for non-CCC messages sent by the Target to the Controller device.

Value	Description
0111-0101	Reserved
0100	2 MHz
0011	4 MHz
0010	6 MHz
0001	8 MHz
0000	12.5 MHz

Bit 3 – DEFBYT

Specifies the defining byte support of GETMXDS CCC.

Value	Description
1	Supports the defining byte.
0	Does not support the defining byte.

Bits 2:0 – MAXWRSPD[2:0]

Specifies the maximum sustained data rate for non-CCC messages sent by the Controller device to the Target.

Value	Description
0111-0101	Reserved
0100	2 MHz
0011	4 MHz
0010	6 MHz
0001	8 MHz
0000	12.5 MHz

24.3.30. Target Mode Interrupt Request Register

Name: I3CxTGTINT
Offset: 0x7A908C

Notes:
1. This field should be programmed by the Target application to provide the data length for the payload data other than the MDB byte.
2. This field is only applicable when BCR[2] is set, indicating that SIR supports payload data.
3. When the conditions for entering a CE3 error are met (expiry of the application running timer and I3CxCLTINT [SIR] is not set to '0'), program this bit to recover from the CE3 error.
4. Upon receiving this bit, the target will move to Controller mode (update I3CXSTATE.CURRCNTRL bit to 1) and generate a START (also clear the I3CxCLTINT [SIR]) and continue by generating 7E followed by STOP.
5. If a NACK response is received for the MR, the Controller reattempts the MR upon detecting the next START condition from the Controller or after the Bus Available time. Once set, the application cannot clear this bit.
6. The target reattempts the SIR either upon detecting the next START condition from the Controller OR after the Bus Available Time if a NACK response is received for the SIR.
7. This feature is used in the CE3 error recovery flow. Once set, the application/software cannot clear this bit.

Bit	31	30	29	28	27	26	25	24
	SIRTGTINDX[3:0]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	0				

Bit	23	22	21	20	19	18	17	16
	SIRDATLEN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MDB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CE3RECOV				MR	SIR[2:0]		
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

Bits 31:28 – SIRTGTINDX[3:0] SIR Target Index bits

Value	Description
1111-0101	Reserved
0100	Virtual Target 4
0011	Virtual Target 3
0010	Virtual Target 2
0001	Virtual Target 1
0000	Virtual Target 0

Bits 23:16 – SIRDATLEN[7:0] SIR Data Length bits(1,2)

Bits 15:8 – MDB[7:0] Mandatory Data Byte bits(2)
Mandatory data byte that is required to be transmitted after the Controller Acknowledges the Target address during SIR transfer.

Bit 7 – CE3RECOV Recovery From CE3 Error bit(3,4)

Value	Description
1	Recover from CE3 error.
0	Recovery complete, moved to Controller mode.

Bit 3 – MR Controller Request bit

Value	Description
1	Issue the MR on the I^3C bus.
0	Clears when the current Controller accepts (ACK) or if the Controller is unable to issue the MR, then the Controller clears this bit automatically and updates the I3CxCLTIBIRESP [IBISTAT] field(5) .

Bit 1 – SIRCTRL Target Interrupt Request Control bit

Value	Description
1	Pull the SDA line low and release SDA after sampling SCL low (pulled by the Controller). Used for CE3 recovery during the Controller handover procedure.
0	Indicates the source of SIR data to be from the registers.

Bits 2:0 – SIR[2:0] Target Interrupt Request bits

Value	Description
1	Target attempts to issue the SIR on the I^3C bus based on the SIRCTRL field.
0	If SIRCTRL = 00, the Target clears this bit automatically and updates the I3CxCLTIBIRESP [IBISTAT] field when either the current Controller accepts by generating (ACK) or if the Target controller is unable to issue the SIR (6). If SIRCTRL = 01, the Target controller clears this bit automatically after sampling the SCL Low (pulled by Controller) for the generated SDA low. The Target will release the SDA line upon sampling the SCL low(7).

24.3.31. Target Interrupt Request Data Register

Name: I3CxTGTSIRDAT
Offset: 0x7A9094

This register, I3CxCLTSIRDAT, containing four bytes of SIR Data, will be in effect only when I3CxCLTINT [SIRCTRL] is programmed to 0000. SIR DATA will be passed after the MDB.

Bit	31	30	29	28	27	26	25	24
	SIRBYT3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SIRBYT2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SIRBYT1[0]							
Access	R/W							
Reset	0							

Bit	7	6	5	4	3	2	1	0
	SIRBYT0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – SIRBYT3[7:0] Target Interrupt Request Data Byte 3 bits

Bits 23:16 – SIRBYT2[7:0] Target Interrupt Request Data Byte 2 bits

Bits 22:15 – SIRBYT1[7:0] Target Interrupt Request Data Byte 1 bits

Bits 7:0 – SIRBYT0[7:0] Target Interrupt Request Data Byte 1 bits

24.3.32. Target IBI Response Register

Name: I3CxTGTIBIRESP
Offset: 0x7A9098

This register reflects the IBI status and the number of data bytes remaining due to early termination.

Note:

1. The value in this field will indicate the number of bytes remaining that were not transmitted because the Controller terminated the SIR transfer early.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		SIRRESPDATLEN[14:8]						
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SIRRESPDATLEN[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
							IBISTAT[1:0]	
Access							R	R
Reset							0	0

Bits 22:8 – SIRRESPDATLEN[14:0] Data Length for SIR Response bits[(1)]

Bits 1:0 – IBISTAT[1:0] IBI Completion Status bits

Value	Description
11	IBI not attempted.
10	Controller Early Terminate (only for SIR with data).
01	IBI accepted by the Controller (ACK response received) .
00	Reserved

24.3.33. Device Control Extended Register

Name: I3CxCTRLEXT
Offset: 0x7A90B0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				DPSLPCAP	REQACKCTRL		DEVOPMOD[1:0]	
Access				R/W	R/W		R	R
Reset				0	0		0	0

Bit 4 – DPSLPCAP Deep Sleep Capability of the Secondary Controller Device bit

Value	Description
1	Module may enter a deep sleep state during which it may miss some Broadcast DEFTGTS and DEFGRPA CCCs sent by the Active Controller. It will require re-synchronization upon re-entering a normal operating state, before it can accept the Controller role with the GETACCCR CCC.
0	Module shall remain active and continue to monitor the I^3C bus to listen for these Broadcast CCCs, and shall not enter a deep sleep state from which it must be re-synchronized by the Active Controller before it can accept the Controller role.

Bit 3 – REQACKCTRL

Value	Description
1	NACK GETACCMST CCC
0	ACK GETACCMST CCC

Bits 1:0 – DEVOPMOD[1:0]

Value	Description
011-010	Reserved
001	Target
000	Controller

24.3.34. Controller Mode SCL I3C Open Drain Timing Register

Name: I3CxSCLODTIM
Offset: 0x7A90B4

Note:
1. The count value takes the number of peripheral clocks to maintain the I/O SCL High/Low Period timing.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ODHCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	ODLCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – ODHCNT[7:0] I^3C Open-Drain High Count bits[(1)]

Bits 7:0 – ODLCNT[7:0] I^3C Open-Drain Low Count bits[(1)]

24.3.35. Controller Mode SCL I^{3}C Push Pull Timing Register

Name: I3CxPPTIM
Offset: 0x7A90B8

Note:
1. The count value takes the number of peripheral clocks to maintain the I/O SCL High/Low Period timing.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	PPHCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	1	0	1	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	PPLCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	1	0	1	0

Bits 23:16 – PPHCNT[7:0] I^{3}C Push-Pull High Count bits

Bits 7:0 – PPLCNT[7:0] I^{3}C Push-Pull Low Count bits

24.3.36. Controller Mode SCL I²C Fast Mode Timing Register

Name: I3CxI2CFMTIM
Offset: 0x7A90BC

Note:
1. The count value takes the number of peripheral clocks to maintain the I/O SCL High/Low Period timing.

Bit	31	30	29	28	27	26	25	24
	FMHCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	FMHCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	FMLCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FMLCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	0	0	0

Bits 31:16 – FMHCNT[15:0] I²C Fast Mode High Count bits
The SCL open-drain high count timing for I²C fast mode transfers.

Bits 15:0 – FMLCNT[15:0] I²C Fast Mode Low Count bits
The SCL open-drain low count timing for I²C fast mode transfers.

24.3.37. Controller Mode SCL I^2C Fast Mode Plus Timing Register

Name: I3CxI2CFMPTIM
Offset: 0x7A90C0

Note:
1. The count value takes the number of peripheral clocks to maintain the I/O SCL High/Low Period timing.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	I2CFMPHCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	I2CFMPLCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	I2CFMPLCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	0	0	0

Bits 23:16 – I2CFMPHCNT[7:0] I^2C Fast Mode Plus High Count bits
The SCL open-drain high count timing for I^2C fast mode plus transfers.

Bits 15:0 – I2CFMPLCNT[15:0] I^2C Fast Mode Plus Low Count bits
The SCL open-drain low count timing for I^2C fast mode plus transfers.

24.3.38. Controller Mode SCL Extended Low Count Timing Register

Name: I3CxSCLELCNT
Offset: 0x7A90C8

Note: This register sets the extended low periods for the I^{3}C transfers to allow the low data rates of the Target devices as specified in the GETMXDS CCC.

Bit	31	30	29	28	27	26	25	24
	LCNT4[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	LCNT3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	LCNT2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LCNT1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bits 31:24 – LCNT4[7:0] I^{3}C Extended Low Count Register 4 bits
SDR4 uses this register field for data transfer.

Bits 23:16 – LCNT3[7:0] I^{3}C Extended Low Count Register 3 bits
SDR3 uses this register field for data transfer.

Bits 15:8 – LCNT2[7:0] I^{3}C Extended Low Count Register 2 bits
SDR2 uses this register field for data transfer.

Bits 7:0 – LCNT1[7:0] I^{3}C Extended Low Count Register 1 bits
SDR1 uses this register field for data transfer.

24.3.39. Controller Mode SCL Termination Bit Low Count Timing Register

Name: I3CxSCLETLCNT
Offset: 0x7A90CC

Note: This register is used to extend the SCL Low period for the Read Termination bit. This register is applicable only in Controller mode.

Bit	31	30	29	28	27	26	25	24
	STOPHLDCNT[3:0]							
Access	R/W	R/W	R/W	R/W				
Reset	0	0	0	1				

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					TERMNLCNT[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 31:28 – STOPHLDCNT[3:0] STOP HOLD Count bits
Stop Hold Count in terms of the peripheral clock, which is used for the generation of Stop Hold in Controller mode.

Bits 3:0 – TERMNLCNT[3:0] I3C Read Termination Bit Low Count bits
Extended I^3C Read Termination Bit low count for I^3C read transfers. Effective Termination-Bit low period is derived based on the SDR speed as shown below:

- SDR0 speed: PPLCNT + TERMNLCNT
- SDR1 speed: LCNT1 + TERMNLCNT
- SDR2 speed: LCNT2 + TERMNLCNT
- SDR3 speed: LCNT3 + TERMNLCNT
- SDR4 speed: LCNT4 + TERMNLCNT

24.3.40. Controller Mode SDA Hold and Mode Switch Delay Timing Register

Name: I3CxSDAHLDTIM
Offset: 0x7A90D0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						SDATXHLD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	1

Bit	15	14	13	12	11	10	9	8
						SDAPPODDLY[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	7	6	5	4	3	2	1	0
						SDAODPPDLY[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bits 18:16 – SDATXHLD[2:0]
This field controls the hold time (in terms of the peripheral clock period) of the transmit data (SDA) with respect to the SCL edge in FM, FM+, SDR and DDR speed modes of operation. The valid values are 1 to 7. Others are Reserved.

Bits 10:8 – SDAPPODDLY[2:0]
This field is used to delay the SDA out with respect to the peripheral clock while switching the transfer from PP1 (Push-Pull Mode SDA=1) to OD1 (Open-Drain SDA=1). The valid value can range from 0 to 4. Others are Reserved.

Bits 2:0 – SDAODPPDLY[2:0]
This field is used to delay the peripheral clock with respect to SDA out while switching the transfer from OD1 (Open-Drain Mode SDA=1) to PP1 (Push-Pull Mode SDA=1). The valid values can range from 0 to 4. Others are Reserved.

24.3.41. Target Mode Bus Free Avail Timing Register

Name: I3CxBUSTIM
Offset: 0x7A90D4

Bit	31	30	29	28	27	26	25	24
	BUSAVAILTIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BUSAVAILTIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BUSFREETIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BUSFREETIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bits 31:16 – BUSAVAILTIM[15:0] Bus Available Count Value bits
These bits are used by the Target/Non-Current Controller to initiate an IBI after a STOP condition.

Bits 15:0 – BUSFREETIM[15:0] I^3C Bus Free Count Value bits
In a pure bus system, this field represents the tCAS parameter. In a mixed bus system, this field is expected to be programmed to tLOW of I^2C Timing.

24.3.42. Target Mode Bus Idle Timing Register

Name: I3CxBUSIDLETIM
Offset: 0x7A90D8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					BUSIDLETIM[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BUSIDLETIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BUSIDLETIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bits 19:0 – BUSIDLETIM[19:0] Bus Idle Count Value bits
This field is used in Target mode to initiate a Hot-Join request if the Dynamic Address is not valid.

24.3.43. Controller Mode SCL Low Extended Timeout Register[1]

Name: I3CxCNTRLTIMOUT
Offset: 0x7A90DC

Note:

1. This register is applicable only in Controller mode.

Bit	31	30	29	28	27	26	25	24
							TIMOUTCNT[25:24]	
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	TIMOUTCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	1	0	1	0	1

Bit	15	14	13	12	11	10	9	8
	TIMOUTCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	1	0	1

Bit	7	6	5	4	3	2	1	0
	TIMOUTCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	1	0	0	0	0	0

Bits 25:0 – TIMOUTCNT[25:0]
This count defines the number of peripheral clock periods to count for the generation of the SCL Low Bus Reset Pattern.

24.3.44. Queue Size Capability Register

Name: I3CxQUESIZE
Offset: 0x7A90E8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	ETXBUFSZ[3:0]				IBIBUFSZ[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	1	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8
	RESPBUFSZ[3:0]				CMDBUFSZ[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	1	0	0	1	0

Bit	7	6	5	4	3	2	1	0
	RXBUFSZ[3:0]				TXBUFSZ[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	0	1	0	0	0	1	0	0

Bits 23:20 – ETXBUFSZ[3:0] Extended TX Buffer Size bits

Value	Description
011	16 DWORDS
010	8 DWORDS
001	4 DWORDS
000	2 DWORDS

Bits 19:16 – IBIBUFSZ[3:0] IBI Queue Size bits

Value	Description
011	16 DWORDS
010	8 DWORDS
001	4 DWORDS
000	2 DWORDS

Bits 15:12 – RESPBUFSZ[3:0] Response Queue Size bits

Value	Description
011	16 DWORDS
010	8 DWORDS
001	4 DWORDS
000	2 DWORDS

Bits 11:8 – CMDBUFSZ[3:0] Command Queue Size bits

Value	Description
011	16 DWORDS
010	8 DWORDS
001	4 DWORDS

Value	Description
000	2 DWORDS

Bits 7:4 – RXBUFSZ[3:0] Receive Data Buffer Size bits

Value	Description
101	64 DWORDS
100	32 DWORDS
011	16 DWORDS
010	8 DWORDS
001	4 DWORDS
000	2 DWORDS

Bits 3:0 – TXBUFSZ[3:0] Transmit Data Buffer Size bits

Value	Description
101	64 DWORDS
100	32 DWORDS
011	16 DWORDS
010	8 DWORDS
001	4 DWORDS
000	2 DWORDS

24.3.45. Target Mode Release SDA Timing Register

Name: I3C1RELSDATIM
Offset: 0x7A90EC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					RELSDATIM[19:16]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RELSDATIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	0	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	RELSDATIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bits 19:0 – RELSDATIM[19:0] Release SDA Count Value bits
This field is used in the Target mode of operation to release the SDA line if SCL does not change for a prescribed amount of time programmed in this register (RELSDATIM * peripheral clock period).

24.3.46. Target HDR Flow Control Register

Name: I3CxHDRFLWCON
Offset: 0x7A90F0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DDRCRCWDIND[1:0]		DDRWREARLYTERMN	DDRWRACKNACK				
Access	R	R	R	R				
Reset	1	1	1	0				

Bits 7:6 – DDRCRCWDIND[1:0]
Indicates whether the CRC word follows an early termination.

Value	Description
011	No CRC word follows early termination.
010	Reserved
001	CRC word follows early termination.
000	Reserved

Bit 5 – DDRWREARLYTERMN
Indicates the capability of the Target for WRITE early termination requests.

Value	Description
1	The Target does not have early-termination capability
0	The Target has early-termination capability

Bit 4 – DDRWRACKNACK
Indicates the capability of the Target to ACK/NACK DDR write commands.

Value	Description
1	The Target does not have ACK/NACK capability.
0	The Target has ACK/NACK capability.

24.3.47. Target Mode Transfer Data Port Register for I3CxEXTCMD0[(1)]

Name: I3CxEXTTXDAT0
Offset: 0x7A9130

Note:
1. The data stored in this register is in big-endian mode. This register is applicable only in Target mode.

Bit	31	30	29	28	27	26	25	24
	EXTTXDAT0[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTTXDAT0[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTTXDAT0[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTTXDAT0[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTTXDAT0[31:0] Transmit Data Port for I3CxEXTCMD0 bits
The transmitted data should always be packed as 4-byte aligned data words. If the command length is not aligned to 4 bytes, then the additional bytes are ignored.

24.3.48. Target Mode Transfer Data Port Register for I3CxEXTCMD1(1)

Name: I3CxEXTTXDAT1
Offset: 0x7A9134

Note:

1. The data stored in this register is in big-endian mode. This register is applicable only in Target mode.

Bit	31	30	29	28	27	26	25	24
	EXTTXDAT1[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTTXDAT1[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTTXDAT1[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTTXDAT1[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTTXDAT1[31:0] Transmit Data Port for EXTTXDAT1 bits
The transmit data should always be packed as 4-byte aligned data words. If the command length is not aligned to 4 bytes, then the additional bytes are ignored.

24.3.49. Target Mode Transfer Data Port Register for I3CxEXTCMD2(1)

Name: I3CxEXTTXDAT2
Offset: 0x7A9138

Note:
1. The data stored in this register is in big-endian mode. This register is applicable only in Target mode.

Bit	31	30	29	28	27	26	25	24
	EXTTXDAT2[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTTXDAT2[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTTXDAT2[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTTXDAT2[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTTXDAT2[31:0] Transmit Data Port for EXTTXDAT2 bits
The transmitted data should always be packed as 4-byte aligned data words. If the command length is not aligned to 4 bytes, then the additional bytes are ignored.

24.3.50. Target Mode Transfer Data Port Register for I3CxEXTCMD3[(1)]

Name: I3CxEXTTXDAT3
Offset: 0x7A913C

Note:

1. The data stored in this register is in big-endian mode. This register is applicable only in Target mode.

Bit	31	30	29	28	27	26	25	24
	EXTTXDAT3[31:24]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTTXDAT3[23:16]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTTXDAT3[15:8]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTTXDAT3[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTTXDAT3[31:0] Transmit Data Port for EXTTXDAT3 bits
The transmitted data should always be packed as 4-byte aligned data words. If the command length is not aligned to 4 bytes, then the additional bytes are ignored.

24.3.51. Target Mode Extended command Register's Transmit Buffer Threshold Status

Name: I3CxEXTCMDTXTHLD
Offset: 0x7A9150

Note: This status is set when the number of empty locations in the Transmit Buffer is greater than or equal to the threshold value specified by I3CxECRTCON[CMDTXBUFTHLD]. This interrupt is cleared automatically when the number of empty locations in the Transmit Buffer is less than the specified threshold value.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					CMD3TXTHLDSTAT	CMD2TXTHLDSTAT	CMD1TXTHLDSTAT	CMD0TXTHLDSTAT
Access					R	R	R	R
Reset					1	1	1	1

Bit 3 – CMD3TXTHLDSTAT I3CXEXTCMD3 Transmit Buffer Threshold Status bit

Bit 2 – CMD2TXTHLDSTAT I3CXEXTCMD2 Transmit Buffer Threshold Status bit

Bit 1 – CMD1TXTHLDSTAT I3CXEXTCMD1 Transmit Buffer Threshold Status bit

Bit 0 – CMD0TXTHLDSTAT I3CXEXTCMD0 Transmit Buffer Threshold Status bit

24.3.52. Target Mode Transmit Data Buffer Status Level Register for I3CxEXTCMD0 and I3CxEXTCMD1

Name: I3CxEXT01BSTA
Offset: 0x7A9154

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	EXT1TBEL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	EXT0TBEL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	0

Bits 23:16 – EXT1TBEL[7:0] I3CxEXTCMD1 Transmit Buffer Empty Level Value bit
Contains the number of empty locations in the I3CxEXTCMD1 Transmit Buffer.

Bits 7:0 – EXT0TBEL[7:0] I3CXEXTCMD0 Transmit Buffer Empty Level Value bits
Contains the number of empty locations in the I3CxEXTCMD0 Transmit Buffer.

24.3.53. Target Mode Transmit Data Buffer Status level Register for I3CxEXTCMD2 and I3CxEXTCMD3

Name: I3CxEXT23BSTA
Offset: 0x7A9158

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	EXT3TBEL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	EXT2TBEL[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	1	0	0	0

Bits 23:16 – EXT3TBEL[7:0] I3CxEXTCMD3 Transmit Buffer Empty Level Value bit
Contains the number of empty locations in the I3CxEXTCMD3 Transmit Buffer.

Bits 7:0 – EXT2TBEL[7:0] I3CxEXTCMD2 Transmit Buffer Empty Level Value bits
Contains the number of empty locations in the I3CxEXTCMD2 Transmit Buffer.

24.3.54. Extended Command Register 0 Word 1

Name: I3CxEXTCMD0
Offset: 0x7A9164

Bit	31	30	29	28	27	26	25	24
	EXTCMD0[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	EXTCMD0[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	EXTCMD0[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	EXTCMD0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTCMD0[31:0] Transmit Command for Extended Command Register-0 in Target Mode bits

24.3.55. Extended Command Register 0 Word 2

Name: I3CxEXTCMD02
Offset: 0x7A9168

This register is provided to program Word-2 of the Extended Transmit Command register-0 (I3CxEXTCMD0). This register must be written with the appropriate value before I3CxEXTCMD0 is written.

Bit	31	30	29	28	27	26	25	24
	EXTCMD02[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTCMD02[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTCMD02[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTCMD02[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTCMD02[31:0] Word-2 of Extended Transmit Command Register-0 bits

24.3.56. Extended Command Register 1 Word 1

Name: I3CxEXTCMD1
Offset: 0x7A9170

Bit	31	30	29	28	27	26	25	24
	EXTCMD1[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	EXTCMD1[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	EXTCMD1[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	EXTCMD1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTCMD1[31:0] Transmit Command for Extended Command Register-1 in Target Mode bits

24.3.57. Extended Command Register 1 Word 2

Name: I3CxEXTCMD12
Offset: 0x7A9174

This register is provided to program the Word-2 of the Extended Transmit Command register-1 (I3CxEXTCMD1). This register must be written with the appropriate value before I3CxEXTCMD1 is written.

Bit	31	30	29	28	27	26	25	24
	EXTCMD12[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTCMD12[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTCMD12[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTCMD12[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTCMD12[31:0] Word-2 of Extended Transmit Command Register-1 bits

24.3.58. Extended Command Register 2 Word 1

Name: I3CxEXTCMD2
Offset: 0x7A917C

Bit	31	30	29	28	27	26	25	24
	EXTCMD2[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTCMD2[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTCMD2[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTCMD2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTCMD2[31:0] Transmit Command for Extended Command Register-2 in Target Mode bits

24.3.59. Extended Command Register 2 Word 2

Name: I3CxEXTCMD22
Offset: 0x7A9180

This register is provided to program Word-2 of the Extended Transmit Command register-2 (I3CxEXTCMD2). This register must be written with the appropriate value before I3CxEXTCMD2 is written.

Bit	31	30	29	28	27	26	25	24
	EXTCMD22[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTCMD22[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTCMD22[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTCMD22[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTCMD22[31:0] Word-2 of Extended Transmit Command Register-2 bits

24.3.60. Extended Command Register 3 Word 1

Name: I3CxEXTCMD3
Offset: 0x7A9188

Bit	31	30	29	28	27	26	25	24
	EXTCMD3[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTCMD3[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTCMD3[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTCMD3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTCMD3[31:0] Transmit Command for Extended Command Register-3 in Target Mode bits

24.3.61. Extended Command Register 3 Word 2

Name: I3CxEXTCMD32
Offset: 0x7A918C

This register is provided to program the Word-2 of the Extended Transmit Command register-3 (I3CxEXTCMD3). This register must be written with the appropriate value before I3CxEXTCMD3 is written.

Bit	31	30	29	28	27	26	25	24
	EXTCMD32[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	EXTCMD32[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	EXTCMD32[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	EXTCMD32[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – EXTCMD32[31:0] Word-2 of Extended Transmit Command Register-3 bits

24.3.62. Target Mode Extended Command Individual Vendor Buffer Reset Register

Name: I3CxETXQRSTCON
Offset: 0x7A91C4

This register is used for individual I3CxEXTCMDy buffer Reset.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					ETXFIFORST3	ETXFIFORST2	ETXFIFORST1	ETXFIFORST0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – ETXFIFORST3 I3CxEXTCMD3 Transmit Buffer Software Reset bit

Value	Description
1	Reset the I3CxEXTCMD3 Transmit Buffer.
0	Cleared automatically once the I3CxEXTCMD3 Transmit Buffer Reset is completed.

Bit 2 – ETXFIFORST2 I3CxEXTCMD2 Transmit Buffer Software Reset bit

Value	Description
1	Reset the I3CxEXTCMD2 Transmit Buffer.
0	Cleared automatically once the I3CxEXTCMD2 Transmit Buffer Reset is completed.

Bit 1 – ETXFIFORST1 I3CxEXTCMD1 Transmit Buffer Software Reset bit

Value	Description
1	Reset the I3CxEXTCMD1 Transmit Buffer.
0	Cleared automatically once the I3CxEXTCMD1 Transmit Buffer Reset is completed.

Bit 0 – ETXFIFORST0 I3CxEXTCMD0 Transmit Buffer Software Reset bit

Value	Description
1	Reset the I3CxEXTCMD0 Transmit Buffer.
0	Cleared automatically once the I3CxEXTCMD0 Transmit Buffer Reset is completed.

24.3.63. Target Mode Extended Command Valid Status Register

Name: I3CxECMDVLDSTA
Offset: 0x7A91C8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
					CMD3VLD	CMD2VLD	CMD1VLD	CMD0VLD
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – CMD3VLD I3CxEXTCMD3 has Finished Sending the Command bit

Bit 2 – CMD2VLD I3CxEXTCMD2 has Finished Sending the Command bit

Bit 1 – CMD1VLD I3CxEXTCMD1 has Finished Sending the Command bit

Bit 0 – CMD0VLD I3CxEXTCMD0 has Finished Sending the Command bit

24.3.64. Target Mode Extended Transmit Buffer and Receive Data Response Threshold Register

Name: I3CxECRTCON
Offset: 0x7A91CC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
						EXTTXBUFTHLD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
						RSPDATTHLD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	1

Bits 18:16 – EXTTXBUFTHLD[2:0] Extended TX Data Buffer Threshold Value bits
This field controls the number of empty locations (or above) in the Extended Transmit FIFO that trigger the I3CxINTSTA[EXTCMDTXTHLDSTA] interrupt.

Value	Description
010	8 DWORDS
001	4 DWORDS
000	1 DWORDS

Bits 2:0 – RSPDATTHLD[2:0] Target Receive Data Threshold Value for Initiating Response bits
This field controls the number of received data bytes after which a response is generated in the Response FIFO.

Value	Description
100	252 DWORDS
011	128 DWORDS
010	64 DWORDS
001	32 DWORDS
000	16 DWORDS

24.3.65. Target Mode Virtual Target Pointer Register

Name: I3CxVIRTGTPTR
Offset: 0x7A91D0

Bit	31	30	29	28	27	26	25	24
	VIRTGTDPTH[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	VIRTGTDPTH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VIRTGTSTRTADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	0	1

Bit	7	6	5	4	3	2	1	0
	VIRTGTSTRTADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – VIRTGTDPTH[15:0] Depth of Virtual Target Address Table bits

Bits 15:0 – VIRTGTSTRTADDR[15:0] Start Address of Virtual Target Address Table bits

24.3.66. Extended Command TX FIFO DMA Channel Selection Register

Name: I3CxDMACHSEL
Offset: 0x7A91D8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	CH1SEL[3:0]				CH0SEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	0	0	0	0

Bits 7:4 – CH1SEL[3:0]
This field is used to select which I3CxEXTCMDy TX FIFO is connected to DMA Channel 1.

Value	Description
011	I3CxEXTCMD3 TX FIFO
010	I3CxEXTCMD2 TX FIFO
001	I3CxEXTCMD1 TX FIFO
000	I3CxEXTCMD0 TX FIFO

Bits 3:0 – CH0SEL[3:0] Start Address of Virtual Target Address Table bits
This field is used to select which I3CxEXTCMDy TX FIFO is connected to DMA Channel 0.

Value	Description
011	I3CxEXTCMD3 TX FIFO
010	I3CxEXTCMD2 TX FIFO
001	I3CxEXTCMD1 TX FIFO
000	I3CxEXTCMD0 TX FIFO

24.3.67. Group Address 0 Status Register

Name: I3CxGRPADDR0STA
Offset: 0x7A91E0

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				T4GRPADDR0V	T3GRPADDR0V	T2GRPADDR0V	T1GRPADDR0V	T0GRPADDR0V
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		GRPADDR0V[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 12 – T4GRPADDR0V Valid Bit for 'Group Address 0' of 'Virtual Target-4'

Value	Description
1	GRPADDR0 is valid for Target 4.
0	GRPADDR0 is invalid for Target 4.

Bit 11 – T3GRPADDR0V Valid Bit for 'Group Address 0' of 'Virtual Target-3'

Value	Description
1	GRPADDR0 is valid for Target 3.
0	GRPADDR0 is invalid for Target 3.

Bit 10 – T2GRPADDR0V Valid Bit for 'Group Address 0' of 'Virtual Target-2'

Value	Description
1	GRPADDR0 is valid for Target 2.
0	GRPADDR0 is invalid for Target 2.

Bit 9 – T1GRPADDR0V Valid Bit for 'Group Address 0' of 'Virtual Target-1'

Value	Description
1	GRPADDR0 is valid for Target 1.
0	GRPADDR0 is invalid for Target 1.

Bit 8 – T0GRPADDR0V Valid Bit for 'Group Address 0' of 'Virtual Target-0'

Value	Description
1	GRPADDR0 is valid for Target 0.
0	GRPADDR0 is invalid for Target 0.

Bits 6:0 – GRPADDR0V[6:0]

Device Group Address 0 is assigned by the Active Controller and used in Target mode of operation.

24.3.68. Group Address 1 Status Register

Name: I3CxGRPADDR1STA
Offset: 0x7A91E4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				T4GRPADDR1V	T3GRPADDR1V	T2GRPADDR1V	T1GRPADDR1V	T0GRPADDR1V
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		GRPADDR1[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 12 – T4GRPADDR1V Valid Bit for 'Group Address 1' of 'Virtual Target-4'

Value	Description
1	GRPADDR1 is valid for Target 4.
0	GRPADDR1 is invalid for Target 4.

Bit 11 – T3GRPADDR1V Valid Bit for 'Group Address 1' of 'Virtual Target-3'

Value	Description
1	GRPADDR1 is valid for Target 3.
0	GRPADDR1 is invalid for Target 3.

Bit 10 – T2GRPADDR1V Valid Bit for 'Group Address 1' of 'Virtual Target-2'

Value	Description
1	GRPADDR1 is valid for Target 2.
0	GRPADDR1 is invalid for Target 2.

Bit 9 – T1GRPADDR1V Valid Bit for 'Group Address 1' of 'Virtual Target-1'

Value	Description
1	GRPADDR1 is valid for Target 1.
0	GRPADDR1 is invalid for Target 1.

Bit 8 – T0GRPADDR1V Valid Bit for 'Group Address 1' of 'Virtual Target-0'

Value	Description
1	GRPADDR1 is valid for Target 0.
0	GRPADDR1 is invalid for Target 0.

Bits 6:0 – GRPADDR1[6:0]

Device Group Address 1 is assigned by the Active Controller and used in Target mode of operation.

24.3.69. Group Address 2 Status Register

Name: I3CxGRPADDR2STA
Offset: 0x7A91E8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				T4GRPADDR2V	T3GRPADDR2V	T2GRPADDR2V	T1GRPADDR2V	T0GRPADDR2V
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		GRPADDR2[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 12 – T4GRPADDR2V Valid Bit for 'Group Address 2' of 'Virtual Target-4'

Value	Description
1	GRPADDR2 is valid for Target 4.
0	GRPADDR2 is invalid for Target 4.

Bit 11 – T3GRPADDR2V Valid Bit for 'Group Address 2' of 'Virtual Target-3'

Value	Description
1	GRPADDR2 is valid for Target 3.
0	GRPADDR2 is invalid for Target 3.

Bit 10 – T2GRPADDR2V Valid Bit for 'Group Address 2' of 'Virtual Target-2'

Value	Description
1	GRPADDR2 is valid for Target 2.
0	GRPADDR2 is invalid for Target 2.

Bit 9 – T1GRPADDR2V Valid Bit for 'Group Address 2' of 'Virtual Target-1'

Value	Description
1	GRPADDR2 is valid for Target 1.
0	GRPADDR2 is invalid for Target 1.

Bit 8 – T0GRPADDR2V Valid Bit for 'Group Address 2' of 'Virtual Target-0'

Value	Description
1	GRPADDR2 is valid for Target 0.
0	GRPADDR2 is invalid for Target 0.

Bits 6:0 – GRPADDR2[6:0]
Device Group Address 2 is assigned by the Active Controller and used in Target mode of operation.

24.3.70. Group Address 3 Status Register

Name: I3CxGRPADDR3STA
Offset: 0x7A91EC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				T4GRPADDR3V	T3GRPADDR3V	T2GRPADDR3V	T1GRPADDR3V	T0GRPADDR3V
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
		GRPADDR3[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 12 – T4GRPADDR3V Valid Bit for 'Group Address 3' of 'Virtual Target-4'

Value	Description
1	GRPADDR3 is valid for Target 4.
0	GRPADDR3 is invalid for Target 4.

Bit 11 – T3GRPADDR3V Valid Bit for 'Group Address 3' of 'Virtual Target-3'

Value	Description
1	GRPADDR3 is valid for Target 3.
0	GRPADDR3 is invalid for Target 3.

Bit 10 – T2GRPADDR3V Valid Bit for 'Group Address 3' of 'Virtual Target-2'

Value	Description
1	GRPADDR3 is valid for Target 2.
0	GRPADDR3 is invalid for Target 2.

Bit 9 – T1GRPADDR3V Valid Bit for 'Group Address 3' of 'Virtual Target-1'

Value	Description
1	GRPADDR3 is valid for Target 1.
0	GRPADDR3 is invalid for Target 1.

Bit 8 – T0GRPADDR3V Valid Bit for 'Group Address 3' of 'Virtual Target-0'

Value	Description
1	GRPADDR3 is valid for Target 0.
0	GRPADDR3 is invalid for Target 0.

Bits 6:0 – GRPADDR3[6:0]
Device Group Address 3 is assigned by the Active Controller and used in Target mode of operation.

24.3.71. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB1LOC1
Offset: 0x7A9200

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.72. Secondary Controller Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB1
Offset: 0x7A9200

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.73. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB1LOC2
Offset: 0x7A9204

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] Least Significant 16-bit Value of the Provisional-ID bits

24.3.74. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxSECDEVCHARTAB2
Offset: 0x7A9204

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.75. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB1LOC3
Offset: 0x7A9208

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.76. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB3
Offset: 0x7A9208

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.77. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB1LOC4
Offset: 0x7A920C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.78. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB4
Offset: 0x7A920C

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.79. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB2LOC1
Offset: 0x7A9210

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.80. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB5
Offset: 0x7A9210

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.81. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB2LOC2
Offset: 0x7A9214

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.82. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB6
Offset: 0x7A9214

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.83. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB2LOC3
Offset: 0x7A9218

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.84. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB7
Offset: 0x7A9218

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.85. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB2LOC4
Offset: 0x7A921C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.86. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB8
Offset: 0x7A921C

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.87. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB3LOC1
Offset: 0x7A9220

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.88. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB9
Offset: 0x7A9220

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.89. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB3LOC2
Offset: 0x7A9224

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.90. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB10
Offset: 0x7A9224

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.91. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB11
Offset: 0x7A9228

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.92. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB3LOC3
Offset: 0x7A9228

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.93. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB12
Offset: 0x7A922C

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.94. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB3LOC4
Offset: 0x7A922C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.95. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB13
Offset: 0x7A9230

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.96. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB4LOC1
Offset: 0x7A9230

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.97. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB14
Offset: 0x7A9234

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.98. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB4LOC2
Offset: 0x7A9234

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.99. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB15
Offset: 0x7A9238

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.100. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB4LOC3
Offset: 0x7A9238

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.101. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB16
Offset: 0x7A923C

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.102. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB4LOC4
Offset: 0x7A923C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.103. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB17
Offset: 0x7A9240

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.104. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB5LOC1
Offset: 0x7A9240

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.105. Secondary Commander Device Characteristic Table Location of Device(n) Register

Name: I3CxSECDEVCHARTAB18
Offset: 0x7A9244

Bit	31	30	29	28	27	26	25	24
	DYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	23	22	21	20	19	18	17	16
	DCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
	BCRTYP[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – DYNADDR[7:0] The Dynamic Address of Device(n) bits

Bits 23:16 – DCRTYP[7:0] The DCR TYPE of Device(n) bits

Bits 15:8 – BCRTYP[7:0] The BCR TYPE of Device(n) bits

Bits 7:0 – BCRTYP[7:0] The Static Address of Device(n) bits

24.3.106. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB5LOC2
Offset: 0x7A9244

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.107. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB5LOC3
Offset: 0x7A9248

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.108. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB5LOC4
Offset: 0x7A924C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.109. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB6LOC1
Offset: 0x7A9250

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.110. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB6LOC2
Offset: 0x7A9254

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.111. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB6LOC3
Offset: 0x7A9258

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.112. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB6LOC4
Offset: 0x7A925C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.113. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB7LOC2
Offset: 0x7A9264

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.114. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB7LOC3
Offset: 0x7A9268

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.115. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB7LOC4
Offset: 0x7A926C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.116. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB7LOC1
Offset: 0x7A9260

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.117. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB8LOC1
Offset: 0x7A9270

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.118. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB8LOC2
Offset: 0x7A9274

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.119. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB8LOC3
Offset: 0x7A9278

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.120. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB8LOC4
Offset: 0x7A927C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.121. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB9LOC1
Offset: 0x7A9280

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.122. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB9LOC2
Offset: 0x7A9284

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.123. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB9LOC3
Offset: 0x7A9288

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.124. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB9LOC4
Offset: 0x7A928C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.125. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB10LOC1
Offset: 0x7A9290

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.126. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB10LOC2
Offset: 0x7A9294

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.127. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB10LOC3
Offset: 0x7A9298

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.128. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB10LOC4
Offset: 0x7A929C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.129. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB11LOC1
Offset: 0x7A92A0

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.130. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB11LOC2
Offset: 0x7A92A4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.131. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB11LOC3
Offset: 0x7A92A8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.132. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB11LOC4
Offset: 0x7A92AC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.133. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB12LOC1
Offset: 0x7A92B0

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.134. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB12LOC2
Offset: 0x7A92B4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.135. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB12LOC3
Offset: 0x7A92B8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.136. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB12LOC4
Offset: 0x7A92BC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.137. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB13LOC1
Offset: 0x7A92C0

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.138. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB13LOC2
Offset: 0x7A92C4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] Least Significant 16-bit Value of the Provisional-ID bits

24.3.139. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB12LOC3
Offset: 0x7A92C8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.140. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB13LOC4
Offset: 0x7A92CC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.141. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB14LOC1
Offset: 0x7A92D0

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.142. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB14LOC2
Offset: 0x7A92D4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] The Least Significant 16-bit Value of the Provisional-ID bits

24.3.143. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB14LOC3
Offset: 0x7A92D8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.144. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB14LOC4
Offset: 0x7A92DC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.145. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB15LOC1
Offset: 0x7A92E0

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.146. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB15LOC2
Offset: 0x7A92E4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] Least Significant 16-bit Value of the Provisional-ID bits

24.3.147. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB15LOC3
Offset: 0x7A92E8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.148. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB15LOC4
Offset: 0x7A92EC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.149. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB16LOC1
Offset: 0x7A92F0

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.150. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB16LOC2
Offset: 0x7A92F4

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] Least Significant 16-bit Value of the Provisional-ID bits

24.3.151. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB16LOC3
Offset: 0x7A92F8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.152. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB16LOC4
Offset: 0x7A92FC

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.153. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB17LOC1
Offset: 0x7A9300

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.154. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB17LOC2
Offset: 0x7A9304

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] Least Significant 16-bit Value of the Provisional-ID bits

24.3.155. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB17LOC3
Offset: 0x7A9308

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.156. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB17LOC4
Offset: 0x7A930C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.157. Device Characteristic Table Location-1 of Device(n) Register

Name: I3CxDEVCHARTAB18LOC1
Offset: 0x7A9310

Bit	31	30	29	28	27	26	25	24
	MSBPROVID[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	MSBPROVID[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	MSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	MSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – MSBPROVID[31:0] Most Significant 32-bit Value of the Provisional-ID bits

24.3.158. Device Characteristic Table Location-2 of Device(n) Register

Name: I3CxDEVCHARTAB18LOC2
Offset: 0x7A9314

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	LSBPROVID[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	LSBPROVID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – LSBPROVID[15:0] Least Significant 16-bit Value of the Provisional-ID bits

24.3.159. Table Location-3 of Device(n) Register

Name: I3CxDEVCHARTAB18LOC3
Offset: 0x7A9318

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	BCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DCR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BCR[7:0] Bus Characteristic Value bits

Bits 7:0 – DCR[7:0] Device Characteristic Value bits

24.3.160. Device Characteristic Table Location-4 of Device(n) Register

Name: I3CxDEVCHARTAB18LOC4
Offset: 0x7A931C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	DEVDYNADDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DEVDYNADDR[7:0] Device Dynamic Address Assigned bits

24.3.161. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB1LOC1
Offset: 0x7A9380

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for a Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ

In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT

One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT

Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND

Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN

Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN

Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.162. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB2LOC1
Offset: 0x7A9384

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I²C
0	I³C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of a Dynamic Address and a Parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I³C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller request from devices.

Value	Description
1	NACK the Controller Request and send auto-disable CCC.
0	ACK the Controller Request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target Interrupt request from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.163. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB3LOC1
Offset: 0x7A9388

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of the Dynamic Address and Parity Bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller Request and send auto-disable CCC.
0	ACK the Controller Request.

Bit 13 – SIRREJ

In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target Interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT

Mandatory one or more data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT

Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND

Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN

Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN

Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.164. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB4LOC1
Offset: 0x7A938C

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for the particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of Dynamic address and parity bit. The LSB bits [22:16] should consist of Dynamic Address field indicates the address to be assigned for the winning I3C device when using ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16]).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.165. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB5LOC1
Offset: 0x7A9390

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]

This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity bits

This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ

In-Band Controller Request Reject field is used to control, per device, whether to accept Controller request from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ

In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt request from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT

One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT

Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND

Enable DDR transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN

Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN

Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.166. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB6LOC1
Offset: 0x7A9394

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity bits
This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.167. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB7LOC1
Offset: 0x7A9398

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]

This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity bits

This field consists of the Dynamic Address and parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for a Specific Device bit

Bit 14 – MRREJ

In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.168. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB8LOC1
Offset: 0x7A939C

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I²C
0	I³C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of Dynamic address and parity bit. The LSB bits [22:16] should consist of Dynamic Address field indicates the address to be assigned for the winning I³C device when using ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16]).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ

In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT

One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT

Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND

Enable DDR transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN

Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN

Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.169. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB9LOC1
Offset: 0x7A93A0

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I²C
0	I³C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of Dynamic Address and parity bit. The LSB bits [22:16] should consist of Dynamic Address field indicates the address to be assigned for the winning I³C device when using ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16]).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.170. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB10LOC1
Offset: 0x7A93A4

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for the particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.171. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB11LOC1
Offset: 0x7A93A8

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.172. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB12LOC1
Offset: 0x7A93AC

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]

This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity

This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ

In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.173. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB13LOC1
Offset: 0x7A93B0

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity bits
This field consists of the Dynamic Address and parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target Interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.174. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB14LOC1
Offset: 0x7A93B4

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity bits
This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ

In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT

One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT

Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND

Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN

Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN

Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.175. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB15LOC1
Offset: 0x7A93B8

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity bits
This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. The PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.176. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB16LOC1
Offset: 0x7A93BC

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I²C
0	I³C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I³C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for the ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ

In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT

One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT

Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND

Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN

Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN

Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.177. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB17LOC1
Offset: 0x7A93C0

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]

This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity bits

This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ

In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ
In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT
One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by the T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT
Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND
Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN
Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN
Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.178. Device Address Table Location of Device(n) Register

Name: I3CxDEVADDRTAB18LOC1
Offset: 0x7A93C4

Bit	31	30	29	28	27	26	25	24
	DEVICE	DEVNACKRTRYCNT[1:0]						
Access	R/W	R/W	R/W					
Reset	0	0	0					

Bit	23	22	21	20	19	18	17	16
	DEVDYNADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TS	MRREJ	SIRREJ	IBIWITHDAT	IBIWITHDAT		DDRERLYTERMNCRCIND	DDRWRERLYTERMNEN
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	DDRWRACKNACKEN	STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – DEVICE Type of Device bit

Value	Description
1	I^2C
0	I^3C

Bits 30:29 – DEVNACKRTRYCNT[1:0]
This field is used to set the Device NACK Retry count for a particular device.

Bits 23:16 – DEVDYNADDR[7:0] Device Dynamic Address with Parity
This field consists of a Dynamic Address and a parity bit. The LSB bits [22:16] should consist of the Dynamic Address field, which indicates the address to be assigned to the winning I^3C device when using the ENTDAA command. The MSB[23] bit is the odd parity of the 7-bit Dynamic Address used for ENTDAA address assignment (~XOR(DEVDYNADDR [22:16])).

Bit 15 – TS Marker for Timestamping IBI for Specific Device bit

Bit 14 – MRREJ
In-Band Controller Request Reject field is used to control, per device, whether to accept Controller requests from devices.

Value	Description
1	NACK the Controller request and send auto-disable CCC.
0	ACK the Controller request.

Bit 13 – SIRREJ

In-Band Target Interrupt Request Reject field is used to control, per device, whether to accept Target interrupt requests from devices.

Value	Description
1	NACK the SIR and send auto-disable CCC.
0	ACK the SIR.

Bit 12 – IBIWITHDAT

One or more mandatory data bytes follow the accepted IBI from the device. Data byte continuation is indicated by T-bit.

Value	Description
1	IBI with one or more mandatory bytes.
0	IBI without a mandatory byte.

Bit 11 – IBIWITHDAT

Packet Error Check enabled for accepted IBI from the device. PEC byte is appended at the end of IBI data from the device. This bit controls whether a PEC check should be performed for IBI data from the device. This bit also controls whether the PEC byte has to be sent for auto-disable CCC when the Controller NACKs the IBI.

Value	Description
1	Packet Error Check enabled for IBI.
0	Packet Error Check disabled for IBI.

Bit 9 – DDRERLYTERMNCRCIND

Enable DDR Transfer Early Termination CRC Word Indicator bit

Bit 8 – DDRWRERLYTERMNEN

Enable DDR WRITE Early Termination Request bit

Bit 7 – DDRWRACKNACKEN

Enable ACK/NACK Capability for DDR WRITE Command bit

Bits 6:0 – STATICADDR[6:0] Device Static Address bits

24.3.179. Virtual Target 1 Address Register

Name: I3CxVIRTGT1CON
Offset: 0x7A9500

Bit	31	30	29	28	27	26	25	24
	VT1ENABLE							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VT1DYNADDRV		VT1DYNADDR[5:0]					
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT1STATICADDRV	VT1STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – VT1ENABLE Controls Whether or Not Virtual Target 1 is Enabled bit

Value	Description
1	Enable Virtual Target 1.
0	Disable Virtual Target 1.

Bit 15 – VT1DYNADDRV Virtual Target 1 Dynamic Address Valid bit

Value	Description
1	Virtual Target 1 Dynamic Address is valid.
0	Virtual Target 1 Dynamic Address is invalid.

Bits 13:8 – VT1DYNADDR[5:0] Virtual Target 1 Dynamic Address bits

Bit 7 – VT1STATICADDRV Virtual Target 1 Static Address Valid bit

Value	Description
1	Virtual Target 1 Static Address is valid.
0	Virtual Target 1 Static Address is invalid.

Bits 6:0 – VT1STATICADDR[6:0] Virtual Target 1 Static Address bits

24.3.180. Manufacturer ID Register for Virtual Target 1 Register

Name: I3CxVIRTGT1MIPIID
Offset: 0x7A9504

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VT1MIPIMFGID[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT1MIPIMFGID[6:0]							VT1PROVIDSEL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:1 – VT1MIPIMFGID[14:0] Specifies the MIPI Manufacturer ID (PID[47:33]) bits

Bit 0 – VT1PROVIDSEL Specifies the Provisional ID Type Selector (PID[32]) bit

24.3.181. Provisional ID Register for Virtual Target 1 Register

Name: I3CxVIRTGT1PID
Offset: 0x7A9508

Bit	31	30	29	28	27	26	25	24
	VT1PARTID[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	VT1PARTID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VT1INSTID[3:0]				VT1PIDDCR[11:8]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT1PIDDCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – VT1PARTID[15:0] Specifies the Part ID of the Device (PID[31:16]) bits

Bits 15:12 – VT1INSTID[3:0] Program the Instance ID of the Target bits

Bits 11:0 – VT1PIDDCR[11:0] Specifies the Additional 12-bit ID of the Device (PID[11:0]) bits

24.3.182. I^{3}C Target Characteristic Register for Virtual Target 1 Register

Name: I3CxVIRTGT1CHARCON
Offset: 0x7A950C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VT1HDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8
	VT1DCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT1DEVROLE[1:0]		VT1HDRCAP	VT1BRIDGEID	VT1OFFLINECAP	VT1IBIPAYLD	VT1IBIREQCAP	VT1MAXDATSPDLIM
Access	R	R	R	R	R	R	R	R
Reset	0	0	1	1	0	1	1	1

Bits 23:16 – VT1HDR[7:0] I^{3}C Device HDR Capability Register Value bits

Bits 15:8 – VT1DCR[7:0] I^{3}C Device Characteristic Value bits

Bits 7:6 – VT1DEVROLE[1:0] Device Role Field in Bus Characteristic Register (BCR[7:6]) bit

Bit 5 – VT1HDRCAP SDR Only or SDR and HDR Capable field in Bus Characteristic Register (BCR[5]) bit

Value	Description
1	HDR DDR capable.
0	SDR capable.

Bit 4 – VT1BRIDGEID Bridge Identifier Field in Bus Characteristic Register (BCR[4]) bit

Value	Description
1	Virtual Target, or exposes other downstream device(s) .
0	Not a Virtual Target and does not expose other downstream device(s).

Bit 3 – VT1OFFLINECAP Offline Capable Field in Bus Characteristic Register (BCR[3]) bit

Value	Description
1	Target will not always respond to I^{3}C Bus commands .
0	Target will always respond to I^{3}C Bus commands.

Bit 2 – VT1IBIPAYLD IBI Payload Field in Bus Characteristic Register (BCR[2]) bit

Value	Description
1	One data byte (MDB) shall follow the accepted IBI, and additional data bytes may follow.
0	No data bytes follow the accepted IBI.

Bit 1 – VT1IBIREQCAP IBI Request Capable field in Bus Characteristic Register (BCR[1]) bit

Value	Description
1	Capable.
0	Not capable.

Bit 0 – VT1MAXDATSPDLIM Max Data Speed Limitation field in Bus Characteristic Register (BCR[0]) bit

Value	Description
1	Target returns the data in the I3CxCLTDATSPD and I3CxCLTIBIPYLD registers in response to the GETMXDS CCC sent by the Controller.
0	Target NACKs the GETMXDS CCC sent by Controller.

24.3.183. Virtual Target 2 Address Register

Name: I3CxVIRTGT2CON
Offset: 0x7A9520

Bit	31	30	29	28	27	26	25	24
	VT2ENABLE							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VT2DYNADDRV		VT2DYNADDR[5:0]					
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT2STATICADDRV	VT2STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – VT2ENABLE Controls Whether or Not Virtual Target 2 is Enabled bit

Value	Description
1	Enable Virtual Target 2.
0	Disable Virtual Target 2.

Bit 15 – VT2DYNADDRV Virtual Target 2 Dynamic Address Valid bit

Value	Description
1	Virtual Target 2 Dynamic Address is valid.
0	Virtual Target 2 Dynamic Address is invalid.

Bits 13:8 – VT2DYNADDR[5:0] Virtual Target 2 Dynamic Address bits

Bit 7 – VT2STATICADDRV Virtual Target 2 Static Address Valid bit

Value	Description
1	Virtual Target 2 Static Address is valid.
0	Virtual Target 2 Static Address is invalid.

Bits 6:0 – VT2STATICADDR[6:0] Virtual Target 2 Static Address bits

24.3.184. Virtual Target 2 Address Register

Name: I3CxVIRTGT2CON
Offset: 0x7A9524

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VT2MIPIMFGID[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT2MIPIMFGID[6:0]							VT2PROVIDSEL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:1 – VT2MIPIMFGID[14:0] Specifies the MIPI Manufacturer ID (PID[47:33]) bits

Bit 0 – VT2PROVIDSEL Specifies the Provisional ID Type Selector (PID[32]) bit

24.3.185. Provisional ID Register for Virtual Target 2

Name: I3CxVIRTGT2PID
Offset: 0x7A9528

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VT2PARTID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VT2INSTID[3:0]				VT2PIDDCR[11:8]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT2PIDDCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – VT2PARTID[7:0] Specifies the Part ID of the Device (PID[31:16]) bits

Bits 15:12 – VT2INSTID[3:0] Instance ID of the Target Program bits

Bits 11:0 – VT2PIDDCR[11:0] Specifies the Additional 12-bit ID of Device (PID[11:0]) bits

24.3.186. I³C Target Characteristic Register for Virtual Target 2

Name: I3CxVIRTGT2CHARCON
Offset: 0x7A952C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VT2HDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8
	VT2DCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT2DEVROLE[1:0]		VT2HDRCAP	VT2BRIDGEID	VT2OFFLINECAP	VT2IBIPAYLD	VT2IBIREQCAP	VT2MAXDATSPDLIM
Access	R	R	R	R	R	R	R	R
Reset	0	0	1	1	0	1	1	1

Bits 23:16 – VT2HDR[7:0] I³C Device HDR Capability Register Value bits

Bits 15:8 – VT2DCR[7:0] I³C Device Characteristic Value bits

Bits 7:6 – VT2DEVROLE[1:0] Device Role Field in Bus Characteristic Register (BCR[7:6]) bits

Bit 5 – VT2HDRCAP SDR Only or SDR and HDR Capable Field in Bus Characteristic Register (BCR[5]) bit

Value	Description
1	HDR DDR capable.
0	SDR capable.

Bit 4 – VT2BRIDGEID Bridge Identifier Field in Bus Characteristic Register (BCR[4]) bit

Value	Description
1	Virtual Target, or exposes other downstream device(s) .
0	Not a Virtual Target and does not expose other downstream device(s).

Bit 3 – VT2OFFLINECAP Offline Capable Field in Bus Characteristic Register (BCR[3]) bit

Value	Description
1	Target will not always respond to I³C Bus commands.
0	Target will always respond to I³C Bus commands.

Bit 2 – VT2IBIPAYLD IBI Payload field in Bus Characteristic Register (BCR[2]) bit

Value	Description
1	One data byte (MDB) shall follow the accepted IBI, and additional data bytes may follow.
0	No data bytes follow the accepted IBI.

Bit 1 – VT2IBIREQCAP IBI Request Capable Field in Bus Characteristic Register (BCR[1]) bit

Value	Description
1	Capable.
0	Not capable.

Bit 0 – VT2MAXDATSPDLIM Max Data Speed Limitation Field in Bus Characteristic Register (BCR[0]) bit

Value	Description
1	Target returns the data in the I3CxCLTDATSPD and I3CxCLTIBIPYLD registers in response to the GETMXDS CCC sent by the Controller.
0	Target NACKs the GETMXDS CCC sent by Controller.

24.3.187. Virtual Target 3 Address Register

Name: I3CxVIRTGT3CON
Offset: 0x7A9540

Bit	31	30	29	28	27	26	25	24
	VT3ENABLE							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VT3DYNADDRV		VT3DYNADDR[5:0]					
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT3STATICADDRV	VT3STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – VT3ENABLE Controls Whether or Not Virtual Target 3 is Enabled bit

Value	Description
1	Enable Virtual Target 3.
0	Disable Virtual Target 3.

Bit 15 – VT3DYNADDRV Virtual Target 3 Dynamic Address Valid bit

Value	Description
1	Virtual Target 3 Dynamic Address is valid.
0	Virtual Target 3 Dynamic Address is invalid.

Bits 13:8 – VT3DYNADDR[5:0] Virtual Target 3 Dynamic Address bits

Bit 7 – VT3STATICADDRV Virtual Target 3 Static Address Valid bit

Value	Description
1	Virtual Target 3 Static Address is valid.
0	Virtual Target 3 Static Address is invalid.

Bits 6:0 – VT3STATICADDR[6:0] Virtual Target 3 Static Address bits

24.3.188. Manufacturer ID Register for Virtual Target 3 Register

Name: I3CxVIRTGT3MIPIID
Offset: 0x7A9544

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VT3MIPIMFGID[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT3MIPIMFGID[6:0]							VT3PROVIDSEL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:1 – VT3MIPIMFGID[14:0] Specifies the MIPI Manufacturer ID (PID[47:33]) bits

Bit 0 – VT3PROVIDSEL Specifies the Provisional ID Type Selector (PID[32]) bit

24.3.189. Provisional ID Register for Virtual Target 3

Name: I3CxVIRTGT3PID
Offset: 0x7A9548

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VT3PARTID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VT3INSTID[3:0]				VT3PIDDCR[11:8]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT3PIDDCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – VT3PARTID[7:0] Specifies the Part ID of the Device (PID[31:16]) bits

Bits 15:12 – VT3INSTID[3:0] Instance ID of the Target Program bits

Bits 11:0 – VT3PIDDCR[11:0] Specifies the Additional 12-bit ID of Device (PID[11:0]) bits

24.3.190. I3C Target Characteristic Register for Virtual Target 3

Name: I3CxVIRTGT3CHARCON
Offset: 0x7A954C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VT3HDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8
	VT3DCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT3DEVROLE[1:0]		VT3HDRCAP	VT3BRIDGEID	VT3OFFLINECAP	VT3IBIPAYLD	VT3IBIREQCAP	VT3MAXDATSPDLIM
Access	R	R	R	R	R	R	R	R
Reset	0	0	1	1	0	1	1	1

Bits 23:16 – VT3HDR[7:0] I^{3}C Device HDR Capability Register Value bits

Bits 15:8 – VT3DCR[7:0] I^{3}C Device Characteristic Value bits

Bits 7:6 – VT3DEVROLE[1:0] Device Role Field in Bus Characteristic Register (BCR[7:6]) bits

Bit 5 – VT3HDRCAP SDR Only or SDR and HDR Capable Field in Bus Characteristic Register (BCR[5]) bit

Value	Description
1	HDR DDR capable.
0	SDR capable.

Bit 4 – VT3BRIDGEID Bridge Identifier Field in Bus Characteristic Register (BCR[4]) bit

Value	Description
1	Virtual Target, or exposes other downstream device(s).
0	Not a Virtual Target and does not expose other downstream device(s).

Bit 3 – VT3OFFLINECAP Offline Capable Field in Bus Characteristic Register (BCR[3]) bit

Value	Description
1	Target will not always respond to I^{3}C Bus commands.
0	Target will always respond to I^{3}C Bus commands.

Bit 2 – VT3IBIPAYLD IBI Payload Field in Bus Characteristic Register (BCR[2]) bit

Value	Description
1	One data byte (MDB) shall follow the accepted IBI, and additional data bytes may follow.
0	No data bytes follow the accepted IBI.

Bit 1 – VT3IBIREQCAP IBI Request Capable Field in Bus Characteristic Register (BCR[1]) bit

Value	Description
1	Capable.
0	Not capable.

Bit 0 – VT3MAXDATSPDLIM Max Data Speed Limitation Field in Bus Characteristic Register (BCR[0]) bit

Value	Description
1	Target returns the data in the I3CxCLTDATSPD and I3CxCLTIBIPYLD registers in response to the GETMXDS CCC sent by the Controller.
0	Target NACKs the GETMXDS CCC sent by Controller.

24.3.191. Virtual Target 4 Address Register

Name: I3CxVIRTGT4CON
Offset: 0x7A9560

Bit	31	30	29	28	27	26	25	24
	VT4ENABLE							
Access	R/W							
Reset	0							

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VT4DYNADDRV		VT4DYNADDR[5:0]					
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT4STATICADDRV	VT4STATICADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – VT4ENABLE Controls Whether or Not Virtual Target 4 is Enabled bit

Value	Description
1	Enable Virtual Target 4.
0	Disable Virtual Target 4.

Bit 15 – VT4DYNADDRV Virtual Target 4 Dynamic Address Valid bit

Value	Description
1	Virtual Target 4 Dynamic Address is valid.
0	Virtual Target 4 Dynamic Address is invalid.

Bits 13:8 – VT4DYNADDR[5:0] Virtual Target 4 Dynamic Address bits

Bit 7 – VT4STATICADDRV Virtual Target 4 Static Address Valid bit

Value	Description
1	Virtual Target 4 Static Address is valid.
0	Virtual Target 4 Static Address is invalid.

Bits 6:0 – VT4STATICADDR[6:0] Virtual Target 4 Static Address bits

24.3.192. Manufacturer ID Register for Virtual Target 4 Register

Name: I3CxVIRTGT4MIPIID
Offset: 0x7A9564

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	VT4MIPIMFGID[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT4MIPIMFGID[6:0]							VT4PROVIDSEL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:1 – VT4MIPIMFGID[14:0] Specifies the MIPI Manufacturer ID (PID[47:33]) bits

Bit 0 – VT4PROVIDSEL Specifies the Provisional ID Type Selector (PID[32]) bit

24.3.193. Provisional ID Register for Virtual Target 4

Name: I3CxVIRTGT4PID
Offset: 0x7A9568

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VT4PARTID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	VT4INSTID[3:0]				VT4PIDDCR[11:8]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT4PIDDCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 23:16 – VT4PARTID[7:0] Specifies the Part ID of the Device (PID[31:16]) bits

Bits 15:12 – VT4INSTID[3:0] Instance ID of the Target Program bits

Bits 11:0 – VT4PIDDCR[11:0] Specifies the Additional 12-bit ID of the Device (PID[11:0]) bits

24.3.194. I3C Target Characteristic Register for Virtual Target 4

Name: I3CxVIRTGT4CHARCON
Offset: 0x7A956C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	VT4HDR[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8
	VT4DCR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	VT4DEVROLE[1:0]		VT4HDRCAP	VT4BRIDGEID	VT4OFFLINECAP	VT4IBIPAYLD	VT4IBIREQCAP	VT4MAXDATSPDLIM
Access	R	R	R	R	R	R	R	R
Reset	0	0	1	1	0	1	1	1

Bits 23:16 – VT4HDR[7:0] I^{3}C Device HDR Capability Register Value bits

Bits 15:8 – VT4DCR[7:0] I^{3}C Device Characteristic Value bits

Bits 7:6 – VT4DEVROLE[1:0] Device Role Field in Bus Characteristic Register (BCR[7:6]) bits

Bit 5 – VT4HDRCAP SDR Only or SDR and HDR Capable Field in Bus Characteristic Register (BCR[5]) bit

Value	Description
1	HDR DDR capable.
0	SDR capable.

Bit 4 – VT4BRIDGEID Bridge Identifier Field in Bus Characteristic Register (BCR[4]) bit

Value	Description
1	Virtual Target, or exposes other downstream device(s).
0	Not a Virtual Target and does not expose other downstream devices.

Bit 3 – VT4OFFLINECAP Offline Capable Field in Bus Characteristic Register (BCR[3]) bit

Value	Description
1	Target will not always respond to I^{3}C Bus commands.
0	Target will always respond to I^{3}C Bus commands.

Bit 2 – VT4IBIPAYLD IBI Payload field in Bus Characteristic Register (BCR[2]) bit

Value	Description
1	One data byte (MDB) shall follow the accepted IBI, and additional data bytes may follow.
0	No data bytes follow the accepted IBI.

Bit 1 – VT4IBIREQCAP IBI Request Capable Field in Bus Characteristic Register (BCR[1]) bit

Value	Description
1	Capable.
0	Not capable.

Bit 0 – VT4MAXDATSPDLIM Max Data Speed Limitation Field in Bus Characteristic Register (BCR[0]) bit

Value	Description
1	Target returns the data in the I3CxCLTDATSPD and I3CxCLTIBIPYLD registers in response to the GETMXDS CCC sent by the Controller.
0	Target NACKs the GETMXDS CCC sent by Controller.

24.3.195. I³C Control Register

Name: I3CxCON
Offset: 0x7A9570

Notes:
1. The Fault injection (I3CCON.FLTINJ) must be disabled and re-enabled to capture the new value from the I3CxFLTINJ register.
2. Applicable only for Controller mode to enable I²C/SMB2/SMB3 voltage levels.

Bit	31	30	29	28	27	26	25	24
			SLPEN			ALTI3C	INTPUR	
Access			R			R/W	R/W	
Reset			0			0	0	

Bit	23	22	21	20	19	18	17	16
	TTXDMACHSEL1[3:0]				TTXDMACHSEL0[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	FLTINJ	SLWEN	SMB3	SMB2	I2C
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TXDMACHSEL[3:0]				RXDMACHSEL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 29 – SLPEN Sleep Enable bit

Value	Description
1	Module operates in Sleep mode.
0	Module disabled in Sleep mode.

Bit 26 – ALTI3C Alternate I³C Pin Mapping bit

Value	Description
1	Alternate location for I3CSCL/I3CSDA (I3CASCL/I3CASDA).
0	Default location for I3CSCL/I3CSDA pins.

Bit 25 – INTPUR Internal Pull-Up Pin Enable bit

Value	Description
1	Internal pull-up enabled.
0	Internal pull-up disabled.

Bits 23:20 – TTXDMACHSEL1[3:0] Target Transmit DMA Acknowledge Channel “0” Selection bits

Value	Description
1111	Check the Channel “15” done acknowledgment state.
… .	
0011	Check the Channel “3” done acknowledgment state.
0010	Check the Channel “2” done acknowledgment state.

Value	Description
0001	Check the Channel “1” done acknowledgment state.
0000	Check the Channel “0” done acknowledgment state.

Bits 19:16 – TTXDMACHSEL0[3:0] Target Mode Transmit DMA Acknowledge Channel “0” Selection bits

Value	Description
1111	Check the Channel “15” done acknowledgment state.
… .	
0011	Check the Channel “3” done acknowledgment state.
0010	Check the Channel “2” done acknowledgment state.
0001	Check the Channel “1” done acknowledgment state.
0000	Check the Channel “0” done acknowledgment state.

Bit 15 – ON Module Enable bit

Value	Description
1	Module enabled.
0	Module disabled.

Bit 13 – SIDL Stop in Idle bit

Value	Description
1	Module stops operation in Idle mode.
0	Module continues operation in Idle mode.

Bit 12 – FLTINJ Fault Injection Sequence Enable bit

Value	Description
1	Fault injection is disabled.
0	Fault injection is enabled when the content of RAM matches ECCFADDR.

Bit 11 – SLWEN Slew Rate Control Enable bit

Value	Description
1	Slew rate control enabled for high-speed mode (400 kHz).
0	Slew rate control disabled for standard speed mode (disabled for 1 MHz mode).

Bit 10 – SMB3 SMB3 Enable bit**(2)**

Value	Description
1	Enable SMBus 3.0 bus configuration.
0	Disable SMBus 3.0 bus configuration.

Bit 9 – SMB2 SMB2 Enable bit**(2)**

Value	Description
1	Enable SMBus 2.0 bus configuration.
0	Disable SMBus 2.0 bus configuration.

Bit 8 – I2C I^2C Enable bit**(2)**

Value	Description
1	Enable I^2C bus operation.
0	Disable I^2C bus operation.

Bits 7:4 – TXDMACHSEL[3:0] Transmit DMA Acknowledge Channel Selection bits

Value	Description
1111	Check the Channel “15” done acknowledgment state.
… .	
0011	Check the Channel “3” done acknowledgment state.
0010	Check the Channel “2” done acknowledgment state.
0001	Check the Channel “1” done acknowledgment state.
0000	Check the Channel “0” done acknowledgment state.

Bits 3:0 – RXDMACHSEL[3:0] Receive DMA Acknowledge Channel Selection bits

Value	Description
1111	Check the Channel “15” done acknowledgment state.
… .	
0011	Check the Channel “3” done acknowledgment state.
0010	Check the Channel “2” done acknowledgment state.
0001	Check the Channel “1” done acknowledgment state.
0000	Check the Channel “0” done acknowledgment state.

24.3.196. I3C Pulse Gobbler Register

Name: I3CxPG
Offset: 0x7A9574

Notes:
1. This register setting is required for I2C mode.
2. It is the user's responsibility to write to the I3CxPG register before the module is enabled. The I3CxPG register is not allowed to change during operation.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
				DDLYEN	DDLYNUMC[3:0]			
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				DFLTEN	PGSMPL[3:0]			
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	1	1	1	0

Bit 12 – DDLYEN Digital Delay Enable bit

Value	Description
1	SDA output delay enable.
0	SDA output delay disabled.

Bits 11:8 – DDLYNUMC[3:0] Digital Delay Cycle Number bits
The SDA output delay count value.

Bit 4 – DFLTEN Digital Filter Enable bit

Value	Description
1	Enabled 50 ns spike filter.
0	Disabled 50 ns spike filter.

Bits 3:0 – PGSMPL[3:0] Digital Pulse Gobbler Sample bits
Digital Pulse Gobbler sample time value.

24.3.197. Fault Injections Register

Name: I3CxFLTINJ
Offset: 0x7A9578

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
			ECCFADDR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
			FLT2PTR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
			FLT1PTR[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 21:16 – ECCFADDR[5:0] ECC Fault Injection Address bits
Address of the targeted RAM word for Fault injection.

Bits 13:8 – FLT2PTR[5:0] ECC Fault Injection Bit Pointer 2 bits
Points to the bit that will be flipped on an error injection event

Value	Description
111111-100111	No Fault injection.
100110	Disabled 50 ns spike filter.
...	
000001	ECC Fault Injection Bit Pointer 1
000000	ECC Fault Injection Bit Pointer 0

Bits 5:0 – FLT1PTR[5:0] ECC Fault Injection Bit Pointer 1 bits

Value	Description
111111-100111	No Fault injection.
100110	Disabled 50 ns spike filter.
...	
000001	ECC Fault Injection Bit Pointer 1
000000	ECC Fault Injection Bit Pointer 0

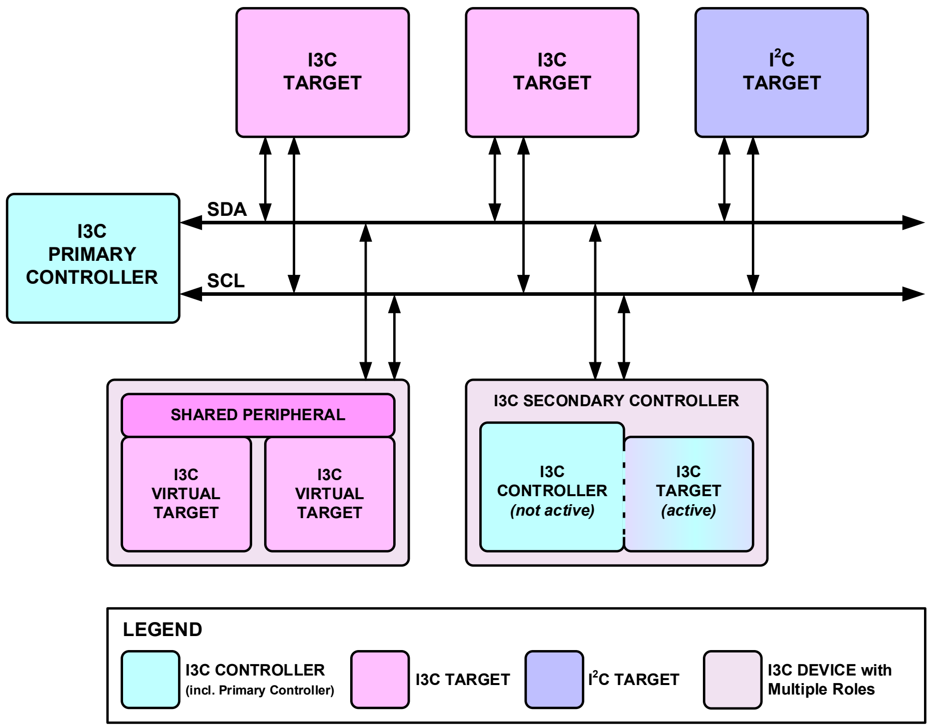
24.4. Operation

24.4.1. Device Characteristics

I^3C Bus can have different roles:

- Primary Controller
- Secondary Controller
- Target

Figure 24-3. Typical I3C Bus



24.4.1.1. Primary Controller

In an I^3C bus, there can be only one Primary Controller device. The device designated as the Primary Controller retains its role at all times, even when it is not the active Controller of the I^3C bus. The Primary Controller has the authority for the initial configuration of the bus and all devices, including any legacy I^2C devices.

24.4.1.2. Secondary Controller

In I^3C, a Target can also hold the role of Secondary Controller by requesting the Primary Controller to become the Active Controller. The Controller role can only be passed after the primary Controller has performed bus initialization. Through the Controller-to-Controller Handoff Procedure, an Active (Primary) Controller hands off the Controller role to the Secondary Controller.

24.4.1.3. Target

The I^3C Target device does not generate the bus clock, and always follows the bus clock generated by the Active Controller. The Target can generate In-Band Interrupts, Hot-Join events and also request to become the Active Controller.

24.4.2. Bus Configuration

The I^3C module supports three types of I^3C bus configurations, as defined by the MIPI I^3C Specification:

- Pure Bus: The configuration with only I^3C devices present on the bus.
- Mixed Fast Bus: The configuration with both I^3C devices and Legacy I^2C Targets present on the bus. In this case, the legacy I^2C Targets are restricted to those that are generally permissible (i.e., Target-only, no clock stretching, and have a true I^2C 50 ns glitch filter on SCL).
- Mixed Slow/Limited Bus: The configuration with both I^3C devices and legacy I^2C Targets present on the bus. In this case, the legacy I^2C Targets are restricted to those that are selectively backward compatible with the I^2C standard (i.e., Target-only and no clock stretching). However, these do not have a true I^2C 50 ns glitch filter on SCL.

In a mixed bus configuration, the maximum possible data rate with I^3C devices depends on the compliance of the I^2C Target as defined by the I^2C specification. The maximum data rate as specified in the I^3C specification is possible only if all I^2C Targets have the 50 ns spike filter.

In the absence of spike filters or if the presence of a filter is unknown, the maximum data rate is limited to only FM or FM+, even for I^3C devices (as per the I^3C Specification). I^2C Targets are not allowed to extend the clock.

24.4.3. I3C Bus Operation

An I^3C bus operates with an Active Controller and one or more Targets, exchanging data using two lines, SDA and SCL, with many similarities to the I^2C bus.

SCL: The **S**erial **CL**ock line is the used by the controller to clock data on the SDA line. SCL is driven by the Active Controller in Push-Pull mode.

SDA: The **S**erial **DA**ta line mostly carries data in/out and is also used for additional signaling purposes. Unlike SCL, SDA can be driven either by the Active Controller or by a Target. SDA can operate in either Open-Drain or Push-Pull configuration, depending on the bus state during bus transactions.

Note: Bus high keeper should be enabled by configuring the respective port pull-up.

24.4.3.1. Format of Data Transfer

I^3C bus transactions are delimited by the same START (S), Repeated START (Sr), and STOP (P) conditions that I^2C uses.

A bus transmission is done by sending 9-bit sequences. A sequence can be either control or data. One or more control sequences are sent first, then data sequences may follow.

Control Sequence format:

- Either a **7-bit Address** or a **Command Code**, issued by the Controller.
- **RnW bit:** 1=Read, 0=Write, issued by the Controller.
- **ACK bit:** 0=ACK, 1=NACK (i.e., not-ACK). Emitted by one or more Targets.

Data Sequence:

- **8-bit Data:** Read or write, depending on the RnW bit of the previous control sequence.
- **Ninth bit:** The meaning of this bit differs between read transfers and write transfers:
  - **During Writes** the ninth bit indicates the parity of the data, as an integrity check (using odd parity)
  - **During Reads** the ninth bit is called **Transition Bit or T-Bit**, and it tells the Controller whether the Target has more data to send: if the T-Bit is 1, then the Target has more data to send; if 0, then there's no more data to send.
    For reads, the Target switches the SDA line to Hi-Z on the rising edge of the SCL signal. This allows the Controller to either continue or stop the transfer by issuing a STOP (i.e., holding and then raising SDA) or a repeated START.

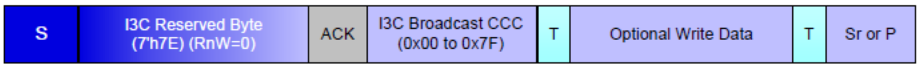
24.4.3.2. Bus Transfers

In I^{3}C, a Controller initiates bus transfers, except for a few special cases such as an in-band interrupt or Hot-Join. Controller-initiated bus transfers always start with the I^{3}C Reserved Address, whose value 7'h7E is ignored by I^{2}C Targets. This makes it possible to detect whether there are any I^{3}C Targets active on the bus, and if so, to then switch to Push-Pull mode for a more efficient transfer.

I^{3}C extends I^{2}C's basic read and write concepts by adding commands, called Common Command Codes (CCC), which may or may not have an associated data sequence. Immediately following the Reserved address ACK, a CCC is sent, possibly followed by a data sequence.

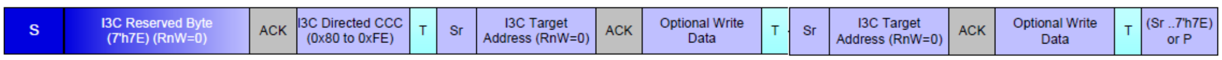
Different Bus Transfers:

- **Broadcast CCC Writes** perform a write to all targets that are active on the bus.



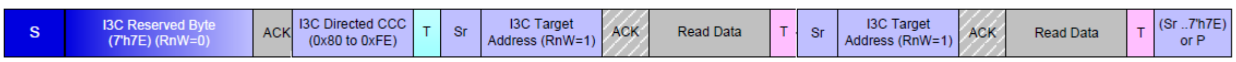
I3C Broadcast CCC Write

- **Directed CCC Writes** perform a write to a single addressed target.



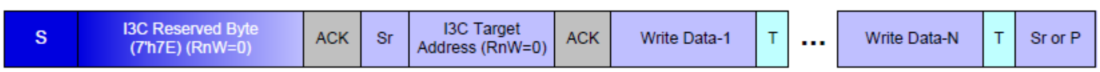
I3C Directed CCC Write

- **Directed CCC Reads** perform a read from a single addressed target.



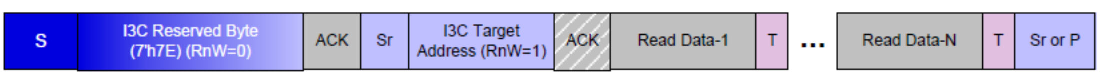
I3C Directed CCC Read

- **Private Write** transfer if RnW=0, followed by a data sequence sent by the controller, with T=odd parity.



I3C Private Write Transfer

- **Private Read** transfer if RnW=1, followed by a data sequence sent by the target, until T=0 (no more data).



I3C Private Read Transfer

24.4.3.3. Data Rate

24.4.3.3.1. Single Data Rate (SDR) Mode

SDR mode is the default mode of the I^{3}C bus. SDR mode is also used to enter other modes, sub-modes, states, and for built-in features such as Common Commands (CCCs), in-band interrupts, and transition from I^{2}C to I^{3}C by assignment of a Dynamic Address.

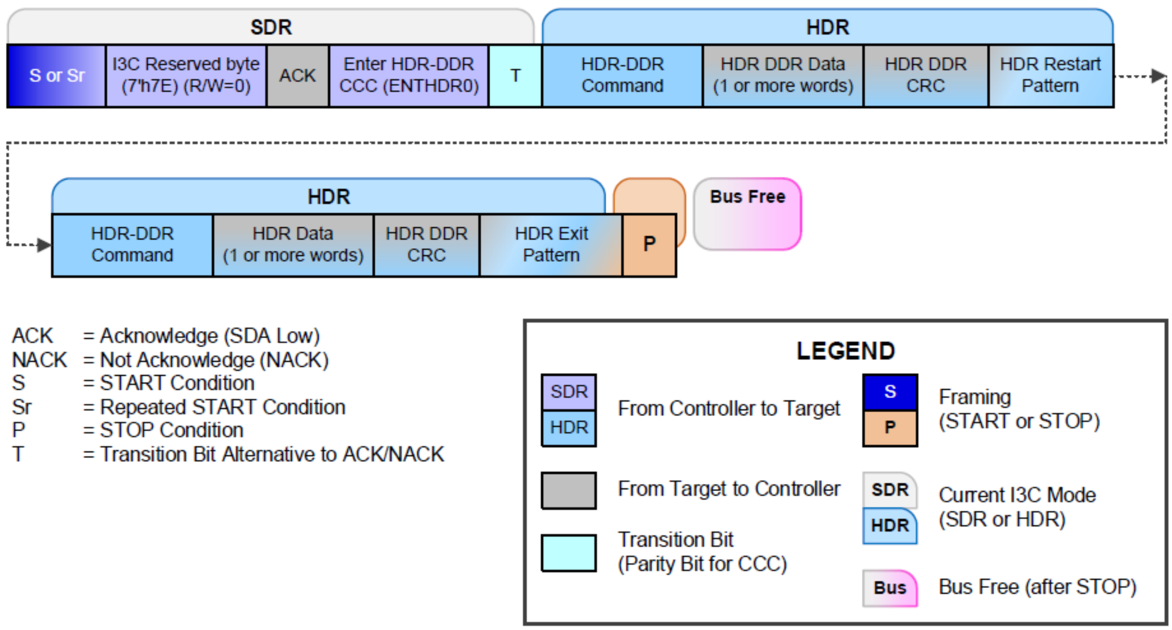
I^{3}C SDR mode is significantly similar to the I^{2}C protocol in terms of procedures and conditions, and as a result, I^{3}C devices and many Legacy I^{2}C Target devices (but not Legacy I^{2}C Controller devices) can coexist on the same I^{3}C bus. However, SDR mode also includes numerous new features that are not present in I^{2}C. For the procedures and conditions that I^{3}C shares with I^{2}C, SDR mode closely follows the definitions in the I^{2}C Specification. I^{2}C traffic from an I^{3}C Controller to an I^{2}C Target will be properly ignored by all I^{3}C Targets because the I^{3}C protocol is designed to allow I^{2}C traffic. I^{3}C traffic from an I^{3}C Controller to an I^{3}C Target will not be seen by most Legacy I^{2}C Target devices because the I^{2}C Spike Filter is opaque to I^{3}C's higher clock speed.

24.4.3.3.2. HDR Double Data Rate Mode (HDR-DDR)

HDR-DDR mode uses SCL as a clock; however, unlike SDR, data and commands change on SDA on both SCL edges (when high and when low), effectively doubling the data rate. By contrast, in SDR mode, SDA is changed only when SCL is low.

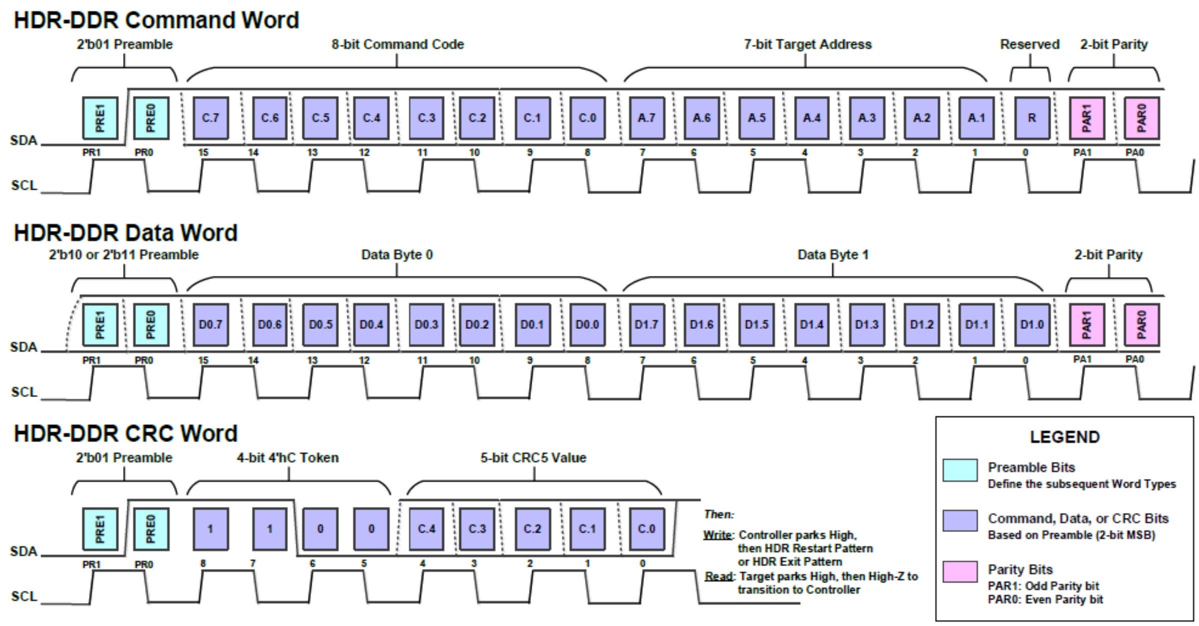
HDR-DDR moves data by words. A word generally contains 16 payload bits, as two bytes in a byte stream and two parity bits. Four HDR-DDR word types are defined: Command Word, Data Word, CRC Word and Reserved Word.

Figure 24-4. Typical HDR-DDR Mode Frame Format



The HDR-DDR Protocol precedes each 18-bit word with a 2-bit preamble, for a total of 20 bits per word.

Figure 24-5. HDR-DDR Word Formats



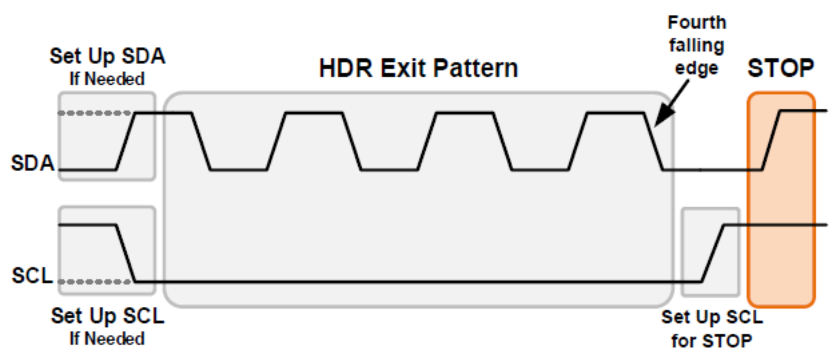
The preamble:

- Indicates the type of data that follows: either Command, Data, or CRC.
- Allows the controller to terminate a read and determine whether the Target is willing to respond to a read.
- Allows the Target to request a write termination.

HDR Exit Pattern and HDR Restart Pattern

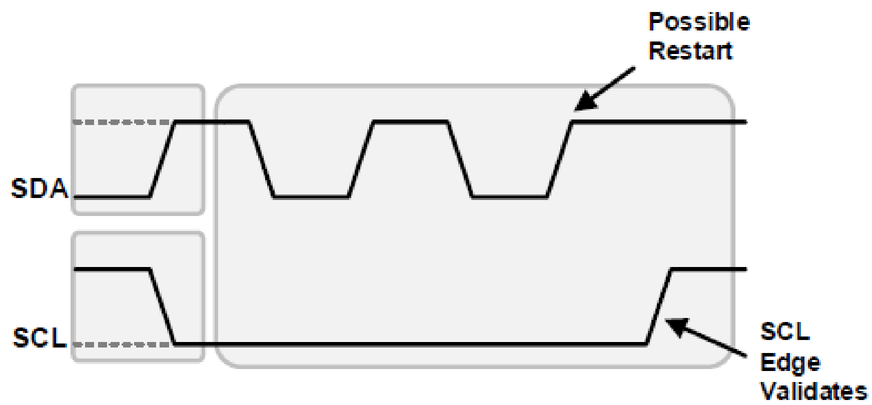
Once an HDR mode is entered, the HDR exit pattern is used to leave it, always exiting back to SDR mode.

Figure 24-6. HDR Exit Pattern Followed by STOP



The HDR restart pattern allows multiple messages to be sent while in HDR mode, without forcing intervening exits to SDR mode.

Figure 24-7. HDR Restart Pattern



24.4.4. Controller Mode

The Controller mode generates the bus clock and sends I^{3}C commands (CCC), addressing either all targets (broadcast CCCs) or specific individual targets (directed CCCs). Controller mode manages Dynamic Address Assignment (DAA) and Address Arbitration.

24.4.4.1. SCL Generation and Timings

The peripheral bus clock is suitably divided to generate the desired value of SCL. Different timing registers are used to divide the peripheral bus clock and generate the SCL. Push-pull and open-drain speeds of operation are automatically achieved by the controller based on the programmed values in the timing registers.

The controller supports the following registers to include the user-provided timing requirements of the I^{3}C protocol:

- SCL I^{3}C Open Drain Timing Register (I3CxSCLODTIM)
- SCL I^{3}C Push Pull Timing Register (I3CxPPTIM)
- SCL I^{2}C Fast Mode Timing Register (I3CxI2CFMTIM)
- SCL I^{2}C Fast Mode Plus Timing Register (I3CxI2CFMPTIM)
- SCL Extended Low Count Timing Register (I3CxSCLELCNT)
- Bus Free and Available Timing Register (I3CxBUSTIM)
- SDA Hold Delay Timing Register (I3CxSDAHLDTIM)
- SCL Extended Termination Low Count Timing Register (I3CxSCLETLCNT)

Count values need to programmed, which are calculated based on the required period and the peripheral clock.

Count = Required period/Peripheral clock period.

The required period can be:

- High time or low time for Open-Drain/Push-Pull mode
- Bus free time

Example of programming the timing register fields for I^{3}C transfers with the given peripheral clock frequency::

The calculation of the I3CxSCLODTIM [ODLCNT], I3CxPPTIM [PPLCNT] and I3CxPPTIM[PPHCNT] fields, as explained in this section, is based on the following assumptions:

- Peripheral clock frequency = 128 MHz (7.8 ns period)
- I^3C speed = Maximum SCL clock frequency (12.8 MHz)
- I^3C Push-Pull SCL minimum low period = 33 ns
- I^3C Push-Pull SCL minimum high period = 41 ns
- I^3C Open-Drain SCL minimum low period = 200 ns
- I^3C Open-Drain SCL maximum high period = 41 ns

PPLCNT = I^3C Push-Pull SCL minimum low period/peripheral clock period = 33 ns/7.8 ns ~=5

PPHCNT = I^3C Push-Pull SCL minimum high period/ peripheral clock period = 41 ns/7.8 ns ~=5

ODLCNT = I^3C Open-Drain SCL minimum low period/ peripheral clock period = 200 ns/7.8 ns ~=25

Note: The minimum required peripheral clock frequency is 128 MHz.

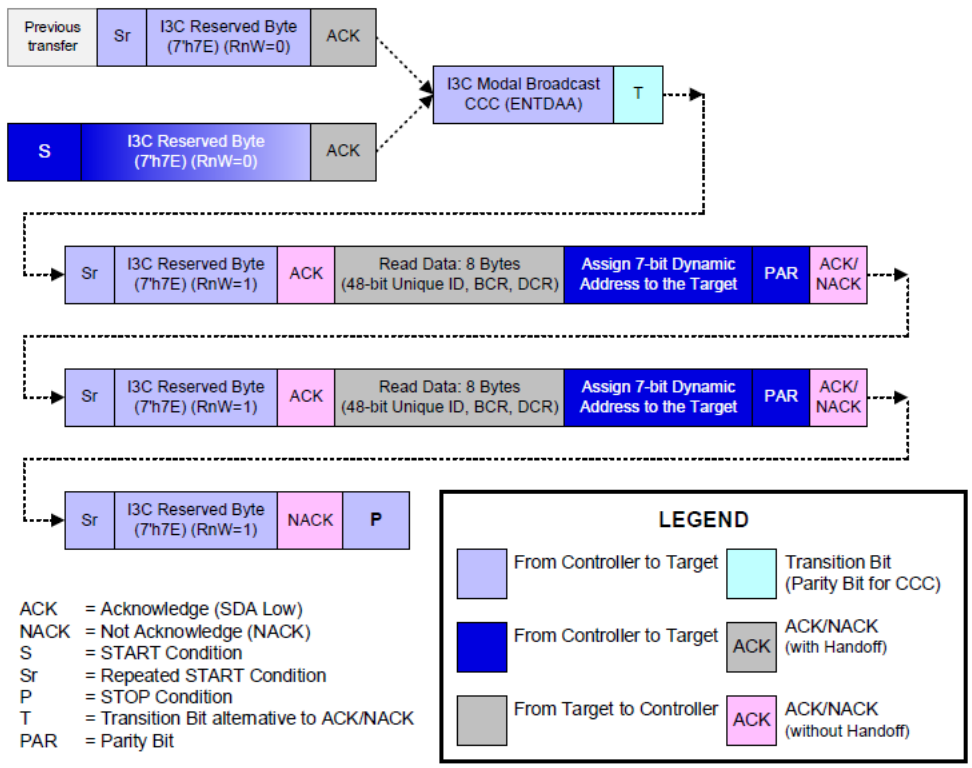
24.4.4.2. Dynamic Address Assignment (DAA)

The Controller must perform a Dynamic Address Assignment procedure in order to provide a unique Dynamic Address to each I^3C device (i.e., with a Target or Secondary Controller role) that is connected to the bus.

Once a Target or Secondary Controller receives a Dynamic Address, that Dynamic Address shall be used in all subsequent transactions on the I^3C bus, until and unless the Controller changes the Dynamic Address.

The Dynamic Address Assignment process includes an Address Arbitration procedure similar to I^2C's. The I3C arbitration procedure differs from I^2C by using the values of the 48-bit Provisioned ID (PID), Bus Characteristic Register (BCR) and Device Characteristic Registers (DCR) concatenated. The device on the I^3C bus with the lowest concatenated value wins each arbitration round in turn, and the Controller assigns a unique Dynamic Address to each winning device.

Figure 24-8. Dynamic Address Assignment Transaction (ENTDAA)

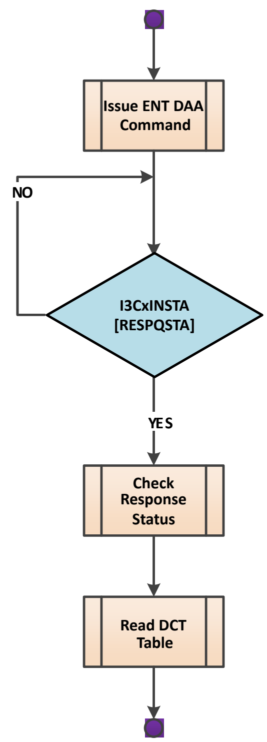


Dynamic Address can be assigned using any of the following commands: ENTDAA, SETDASA, SETAASA or SETNEWDA.

The steps required for Dynamic Address assignment are:

1. Program the device address table (I3CxDEVADDRTAByLOC1) depending on the number of devices connected.
2. Send the command by writing to the Command Queue (I3CxCMDQUE) with the Address Assignment Command (refer to Address Assignment Command Data Structure).
3. The Dynamic Address assignment continues until one of the following conditions occurs:
   a. A NACK response is received for the header 0x7E/W (No I^{3}C devices present).
   b. A NACK response is received for the Static Address (Not a valid Static Address or the device does not exist).
   c. A NACK response is received for the assigned Dynamic Address.
   d. DEVICE_COUNT written to the I3CxCMDQUE reaches zero.
4. The Controller writes the transfer complete status into the Command Response (I3CxRESPQUE). The data length field of the response data structure (Response Data Structure) indicates the remaining device count in case the transfer is terminated abruptly due to a NACK response from the Target.
5. The first winning device gets the first Dynamic Address pointed to by the DEV_INDEX of the command. The second winning device gets the second Dynamic Address pointed to by the DEV_INDEX+1, and so on. The received 48-bit PID, BCR, and DCR along with the assigned Dynamic Address are captured in the device characteristics table (I3CxDEVCHARTAByLOC1, I3CxDEVCHARTAByLOC2, I3CxDEVCHARTAByLOC3, I3CxDEVCHARTAByLOC4).

Figure 24-9. ENTDAA Transfer Flow



Notes:

1. For the SETAASA command, use Transfer Command Data Structure without Data Payload.
2. For the SETNEWDA command, use Transfer Command Data Structure with Target address and new Target address as data payload (refer to Directed Write and Read Transfers).

24.4.4.3. In-Band Interrupt (IBI) Detection and Handling

The I^3C Target devices can initiate communication with the Active Controller through In-Band Interrupt (IBI). The following types of IBIs are possible on an I^3C bus:

- Hot-Join Request (HJ) from a Hot-Join capable Target
- Target Interrupt Request (SIR) from a Target
- Controller Ownership Request (MR) from a Controller capable Target (Secondary Controller)

The Controller detects (SDA low) and receives the In-Band interrupt (IBI ID). This enables the Controller to start providing the SCL clocks (period = I3CxSCLODTIM [ODLCNT] + I3CxSCLODTIM [ODHCNT]) to receive the IBI ID from the requesting Target device. The Controller detects the In-Band interrupt in the following scenarios:

- Upon detecting a low on the SDA input port after a Power-On-Reset (POR).
- Upon detecting an arbitration loss during an address phase of any Controller-initiated transfer following a START condition (not RESTART).
- Upon detecting the SDA input port going low (not initiated by the Controller) following a STOP condition (Target Initiated IBI).

24.4.4.3.1. I^3C Hot-Join (HJ) Request

If the received IBI ID matches the Hot-Join ID (7'b0000010, RnW = 0), then the response to the received HJ request is based on the programmed value of I3CxCNTRL[HOTJOIN].

If the I3CxCNTRL[HOTJOIN] is set to 0, the controller responds to the HJ request with ACK and sets the 'IBI_STS' field in IBI Queue Data Structure to ACK to indicate to the application that the controller has ACK'd the received HJ request.

If the I3CxCNTRL[HOTJOIN] is set to 1, the Controller responds to the HJ request with a NACK followed by issuing a broadcast DISEC CCC command (DISHJ bit set) with the RESTART condition. This disables the HJ request generation from all the unaddressed devices at that instant. The Controller sets the 'IBI_STS' field in IBI Queue Data Structure to NACK to indicate to the application that the Controller has rejected the received HJ request. The application can optionally set the HJ Reject Notify Control (I3CxIBIQNOTIFY) to get an IBI status for a rejected HJ request. Otherwise, the Controller moderates the IBI Status generation for rejected HJ requests.

Based on the IBI status for the HJ request, the application must issue the ENTDAA to assign the dynamic address for the devices that generate the HJ request.

24.4.4.3.2. I^{3}C Controller Request

If the received IBI ID matches the Dynamic Address of any one of the valid entries in the DAT and the received RnW bit is 0, the response for the received MR is based on the programmed value of the MRREJ field of the device address table (DAT).

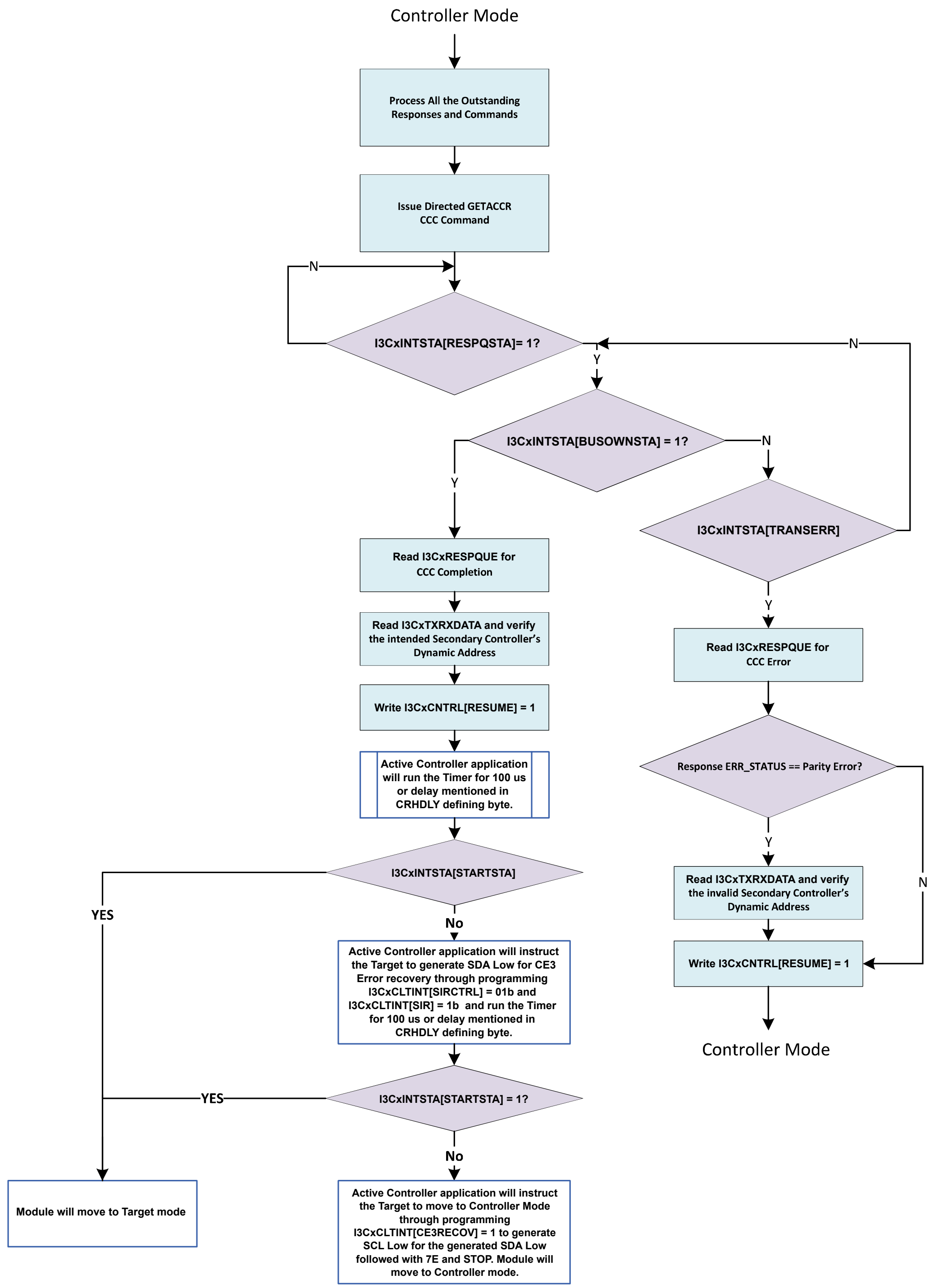
The Controller mode allows the application to either set the I3CxDEVADDRTABnLOC1[MRREJ] to 0 or 1 individually to accept or reject the MR request, respectively. The application is expected to always set I3CxDEVADDRTABnLOC1[MRREJ] to 1. This requirement allows the Controller to reject any MR request if received unexpectedly from a malfunctioning Target device.

If the I3CxDEVADDRTABnLOC1[MRREJ] is set to 1, the Controller responds to the MR request with NACK, followed by issuing a directed DISEC CCC command (DISMR bit set) with the RESTART condition to the MR-initiated Target. The Controller sets the 'IBI_STS' field in IBI Queue Data Structure as NACK to indicate to the application that the Controller has rejected the received MR request. The application can optionally set the MR Reject Notify Control (I3CxIBIQNOTIFY) to get an IBI Status for a rejected MR request. Otherwise, the Controller moderates the IBI Status generation for rejected MR requests.

If the I3CxDEVADDRTABnLOC1[MRREJ] is set to 0, then the Controller responds to the MR request with ACK and sets the 'IBI_STS' field in IBI Queue Data Structure to ACK, which indicates to the application that the Controller has ACK'd the received MR request.

Based on the IBI status for the MR request, the application must follow the Controller ownership handover procedure.

Figure 24-10. Programming Flow to Switch from Controller to Target Mode



Note: If the received IBI ID (MR) does not match any of the dynamic address entries of the DAT table, the Controller sends a NACK response and generates IBI status to the application. The Controller does not send any disable event command and expects the application to take necessary action if the unknown device repeatedly interrupts. In this case, the generation of IBI status is independent of the MR Reject Notify Control settings.

24.4.4.3.3. I3C Target Interrupt Request (SIR)

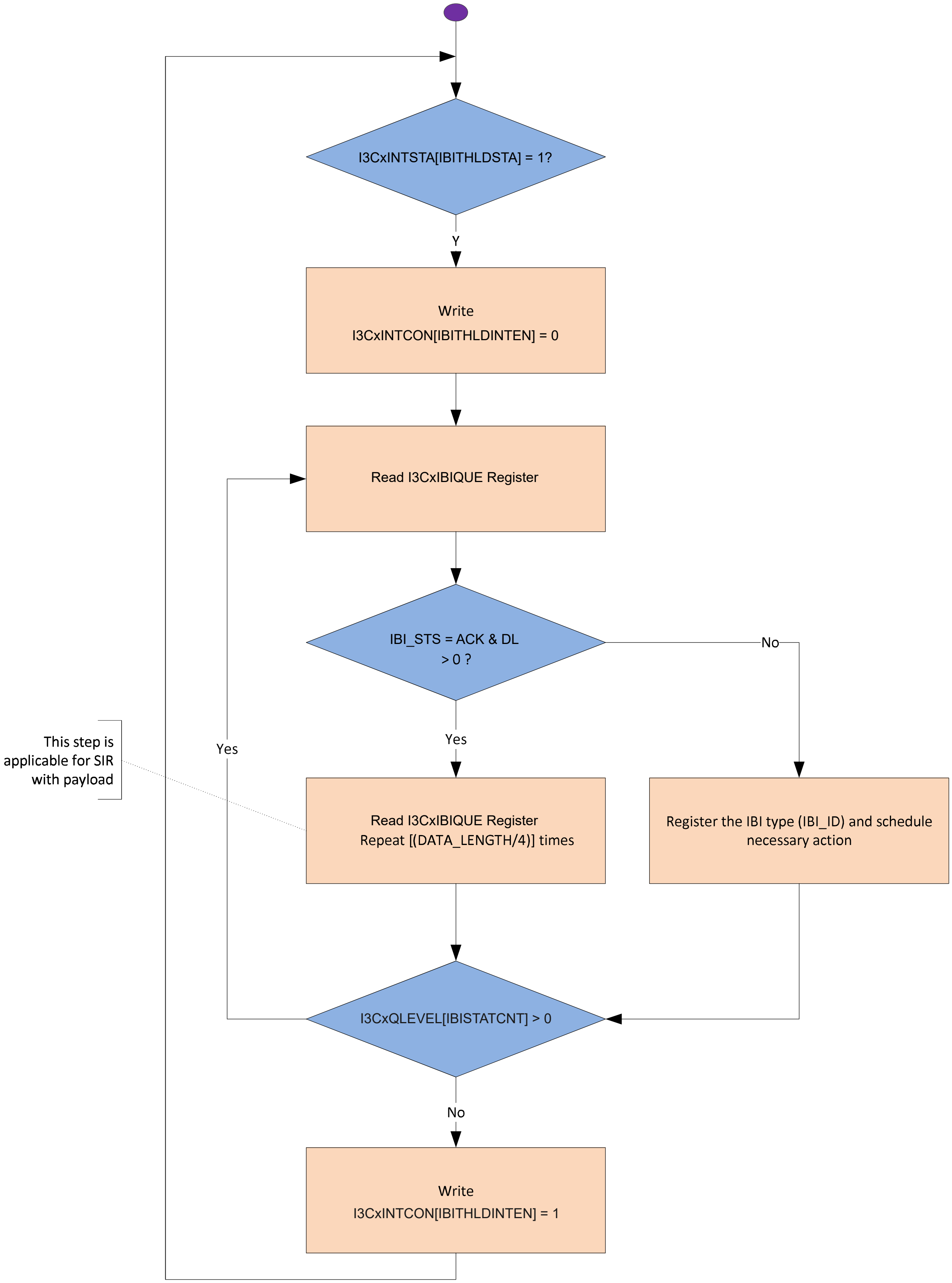
If the received IBI ID matches the Dynamic Address of any one of the valid entries in the DAT and the received RnW bit is 1, the response for the received SIR is based on the programmed SIRREJ field of the device address table (DAT).

If the I3CxDEVADDRTABnLOC1[SIRREJ] is set to 1, the Controller NACKs the SIR and then issues a directed DISEC CCC command (DISINT bit set) with the RESTART condition targeting the matching Dynamic Address. This disables the SIR generation in the requested Target device. The IBI status for the corresponding SIR indicates to the application that the incoming IBI is NACK'd through the 'IBI_STS' field in IBI Queue Data Structure.

If I3CxDEVADDRTABnLOC1[SIRREJ] is set to 0, the Controller ACKs the SIR. If the IBI Payload Control is set to 1, the Controller continues to generate the SCL clocks after the ACK to accept the IBI payload bytes starting from the MDB (Mandatory Data Byte) until the Target device terminates the transfer. The application must set I3CxDEVADDRTABnLOC1[IBIWITHDAT] to 1 only when the Target device supports the mandatory byte (BCR[2]=1). If the I3CxDEVADDRTABnLOC1[IBIWITHDAT] is set to 0, the Controller stops generating the SCL clock after acknowledging the SIR. The Controller then notifies the application to take necessary action for the accepted SIR. The application must not set the I3CxDEVADDRTABnLOC1[IBIWITHDAT] to 1 when the Target device does not support the mandatory data byte (BCR[2]=0).

Note: If the received IBI ID (SIR) does not match any of the dynamic address entries in the DAT table, then the Controller sends a NACK response and generates an IBI status to the application. The Controller does not send any disable event command and expects the application to take necessary action if the unknown device repeatedly interrupts. In this case, the generation of IBI status is independent of the SIR Reject Notify Control settings.

Figure 24-11. IBI Handling Procedure

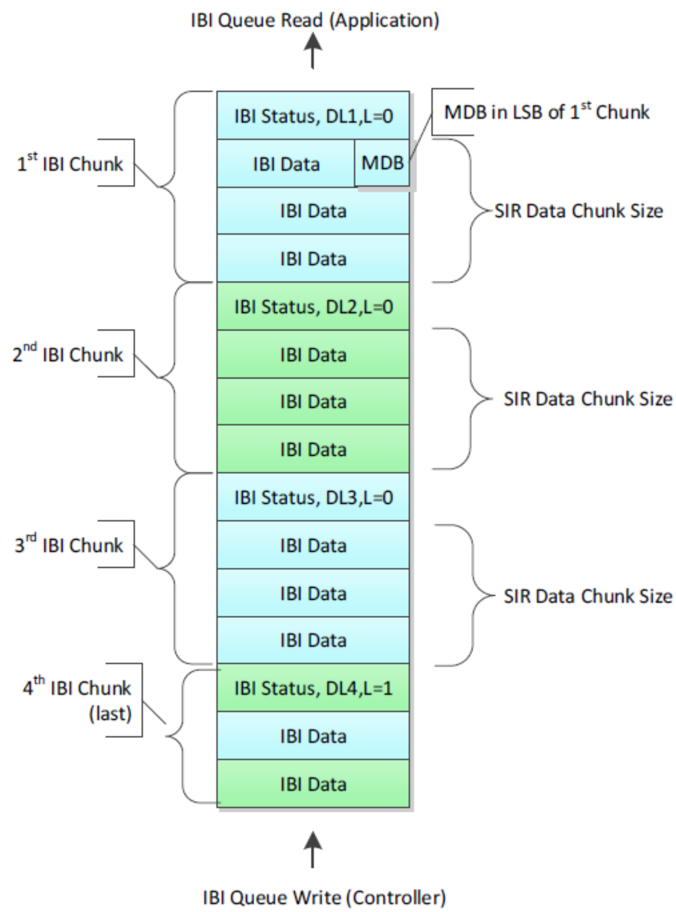


24.4.4.3.4. IBI Queue Data Structure

The IBI queue comprises the interleaved data received from the Target device, and the status generated by the Controller for the respective data slice of SIR or HJ/MR. For configurations that support IBI without payload, only the IBI status entry is written into the IBI queue.

The data portion of the structure comprises either the data received from the Target device along with the IBI (MDB + Other payload bytes, if any) or the data received for Auto-Command (if supported) issued for the requesting device. If the data payload size of the SIR exceeds the programmed SIR Data chunk size, the Controller slices the incoming data bytes into multiple chunks and generates an IBI status for each chunk. The last data chunk is indicated as LAST_STATUS=1 in the corresponding IBI status. Figure 24-12 shows an example of the IBI queue data structure holding the IBI data (SIR) and the status for an IBI with non-multiple chunk size data.

Figure 24-12. IBI with Payload Structure



The application reads the IBI queue upon detecting the interrupt status I3CxINTSTA [IBITHLDSTA]. The interrupt can be moderated by setting the I3CxQUETHLDCON [IBISTATHLD] field. The first location read always provides the IBI status for the first chunk, which includes the data length of that chunk in bytes. The application can read the data portion of the chunk without waiting for any further interrupt. The number of data reads must be limited to the data length field of the corresponding status.

24.4.4.4. Controller Mode Data Structure

24.4.4.4.1. Command Data Structure

There are four types of data structures that are used to write the command queue register (I3CxCMDQUE):

- Transfer Command Data Structure
- Transfer Argument Data Structure
- Short Data Argument Data Structure
- Address Assignment Command Data Structure

CMD_TYPE \ BIT_OFFSET	31	30	29	28	27	26	25	24	23 : 21	20 : 16	15	14 : 8	7	6	5 : 3	2 : 0
Transfer Command	PEC	TOC	RESV	RnW	SDAP	ROC	DBP	RESV	SPEED	DEV_INDX	CP	CMD		TID		CMD_ATTR (0)
Transfer Argument	DATA_LENGTH										DB		RESV			CMD_ATTR (1)
Short Data Argument	DATA_BYTE_3								DATA_BYTE_2		DATA_BYTE_1		RESV		BYTE_STRB	CMD_ATTR (2)
Address Assignment command	RESV	TOC	RESV			ROC	DEV_COUNT			DEV_INDX	RESV	CMD		TID		CMD_ATTR (3)

Transfer Command Data Structure

The Transfer Command is used to initiate CCC and private transfers. For transfers with data payloads, an additional data structure (either Transfer Argument or Short Data Argument) is used to provide payload details.

Note: If a transfer consists of a payload of at least one byte, then either the Transfer Argument or the Short Data Argument must be written into the command port prior to the Transfer Command.

Table 24-3. Transfer Command Data Structure

SL No	Field Name	Bit Field	Width	Description
1	CMD_ATTR	2:0	3	Command Attribute Defines the command type and its bit field format. • 4-7: Reserved • 3: Address Assignment Command • 2: Short Data Argument • 1: Transfer Argument • 0: Transfer Command
2	TID	6:3	4	Transaction ID This field is used as the identification tag for the commands. The I^{3}C Controller returns this ID along with the response upon completion or upon error. • 4'b0000 - 4'b0111 - User-Defined TID • 4'b1000 - 4'b1111 - Reserved for I^{3}C Controller
3	CMD	14:7	8	Transfer Command This field is used to define the transfer command type. The field can be programmed to: 1. 8-bit Common Command Code for CCC transfers. 2. 7-bit Command Code for HDR-DDR transfers (bit[14] is reserved).
4	CP	15	1	Command Present This bit is used to control whether the transfer should be initiated with the transfer command represented in the 'CMD' filed or not. • 1 - CMD field is valid This bit is applicable for CCC and HDR transfers. • 0 - CMD field is not valid

Table 24-3. Transfer Command Data Structure (continued)

SL No	Field Name	Bit Field	Width	Description
5	DEV_INDX	20:16	5	Device Index This field is used to reference the Device Address Table for getting the Target address. DEV_INDX field points to the offset address of Device Address Table.
6	SPEED	23:21	3	Speed This field is used to indicate the speed in which the transfer should be driven. Values (I3C Mode): • 7: I^{2}C FM (JEDEC broadcast CCC only) • 6: HDR-DDR • 5: Reserved • 4: SDR4 • 3: SDR3 • 2: SDR2 • 1: SDR1 • 0: SDR0 Note that HDR-DDR is supported only if the Transfer Command Structure is set-up to use Transfer Argument and NOT Short Data Argument. Values (I^{2}C mode): – 2 -7: Reserved – 1: I2C FM+ – 0: I2C FM
7	RESV	24	1	Reserved
8	DBP	25	1	Defining Byte Present DBP indicates whether the current CCC command is with Defining Byte or not. This bit field is valid only when CP (command Present) bit is enabled, otherwise this bit is ignored. • 1: Defining Byte is present. This bit is applicable only for Broadcast and Directed SDR CCC Transfers. • 0: Defining Byte is not present.
9	ROC	26	1	Response on Completion. This field indicates whether the Response Status is required or not after the execution of this command for the successful transfer. • 1 - Response Status is required. • 0 - Response Status is not required. **Notes:** a. The exception to the control is that the response status is generated when the transfer has encountered an error condition. b. Always set the ROC bit to 1 for the Read commands (RnW=1), so that the number of data received is indicated (through the DATA_LENGTH field in the Response Queue) if the Target terminates earlier than the Controller.

Table 24-3. Transfer Command Data Structure (continued)

SL No	Field Name	Bit Field	Width	Description
0	SDAP	27	1	Short Data Argument Present This field indicates whether the command written prior to the Base command should be treated as a Short Data Argument or a Transfer Argument. • 1 - Prior written command is a Short Data Argument. • 0 - Prior written command is a Transfer Argument.
11	RnW	28	1	Read and Write This bit controls whether a Read or Write transfer is performed. • 1 - Read Transfer Note: In HDR transfers, this bit is used to set the Read/Write flag of the HDR-DDR Command Code. • 0 - Write Transfer
12	TGT_RST	29	1	Target Reset Pattern Generation This bit indicates whether the issued command generates a Target Reset Pattern at the end of the transfer. This bit is enabled only with: • TOC field set to 1. • SPEED field set to 0 to 4. • CP set to 1 • CMD set to RSTACT Note: It is recommended to use this bit only for RSTACT CCC's.
13	TOC	30	1	Termination On Completion This bit controls whether a STOP needs to be issued after the completion of the transfer or not. • 1 - STOP issued after this transfer. • 0 - The next transfer starts with a RESTART condition.
14	PEC	31	1	Parity Error Check Enable This bit enables the generation and validation of the PEC byte for SDR CCC and private transfers. • 1: PEC check is enabled. This bit is valid only for SDR Transfers and not for HDR Transfers. • 0: PEC check is disabled.

Transfer Argument Data Structure

The Transfer Argument Data Structure is used to provide payload-related information. This data structure must be used when the TX FIFO is used to get the payload bytes.

Table 24-4. Transfer Argument Data Structure

SL No	Field Name	Bit Field	Width	Description
1	CMD_ATTR	2:0	3	Command Attribute Defines the command type and its bit-field format. • 4-7: Reserved • 3: Address Assignment Command • 2: Short Data Argument • 1: Transfer Argument • 0: Transfer Command
2	RESV	7:3	5	Reserved
3	DB	15:8	8	Defining Byte Value DB indicates the 8-bit defining byte to be transferred in the CCC transfer. This byte is valid only when both CP and DBP bits are enabled; otherwise, the controller ignores this byte.
4	DL	31:16	16	Data Length: This field is used to indicate the data length of the transfer. For CCC transfers, it is expected that the correct length be programmed. The controller deals with all CCCs transparently.

Short Data Argument Data Structure

The Short Data Argument Data Structure is used to provide payload-related information for the TX transfers when the payload size is less than or equal to three bytes.

Note: Short Data Argument is not supported for HDR transfers because, in HDR modes, a maximum of two words can be transferred in this format.

Table 24-5. Short Data Argument Data Structure

SL.No	Field Name	Bit Field	Width	Description
1	CMD_ATTR	2:0	3	Command Attribute Defines the command type and its bit field format. • 4-7: Reserved • 3: Address Assignment Command • 2: Short Data Argument • 1: Transfer Argument • 0: Transfer Command
2	BYTE_STRB	5:3	3	Byte Strobe This field is used to select the valid data bytes of the Short Data Argument. • BYTE_STRB[0] - Data Byte -0 Valid Qualifier • BYTE_STRB[1] - Data Byte -1 Valid Qualifier • BYTE_STRB[2] - Data Byte -2 Valid Qualifier • Valid combinations = 3'b001, 3'b011 and 3'b111
3	RESV	7:6	2	Reserved
4	DATA_BYTE _0/DB	15:8	8	Data Byte -0/Defining Byte: This field is used for storing the Data Byte -0. DB indicates the 8-bit Defining Byte to be transferred in the CCC transfer. This byte is valid as a Defining Byte (DB) when both CP and DBP bits are enabled; otherwise, it's used as DATA_BYTE_0. **Note:** When the Defining Byte (DBP) is enabled in Short Data Argument: 1. The BYTE_STRB[0] bit is treated as always enabled, whether or not you set it. 2. The maximum possible payload bytes to be sent is '2' due to the first byte being occupied by the Defining Byte.
5	DATA_BYTE_1	23:16	8	Data Byte -1: This field is used for storing Data Byte -1.

Table 24-5. Short Data Argument Data Structure (continued)

SL.No	Field Name	Bit Field	Width	Description
6	DATA_BYTE_2	31:23	8	Data Byte -2: This field is used for storing Data Byte -2.

Address Assignment Command Data Structure

In the Controller mode of operation, the Address Assignment Command is used to initiate ENTDAA and SETDASA transfers.

Table 24-6. Address Assignment Command Data Structure

SL.No	Field Name	Bit Field	Width	Description
1	CMD_ATTR	2:0	3	Command Attribute Defines the command type and its bit field format. • 4-7: Reserved • 3: Address Assignment Command • 2: Short Data Argument • 1: Transfer Argument • 0: Transfer Command
2	TID	6:3	4	Transaction ID This field is used as the identification tag for the commands. The I^3C controller returns this ID along with the response upon completion or upon error. • 4'b0000 - 4'b0111 - User-Defined TID • 4'b1000 - 4'b1111 - Reserved for I^3C controller.
3	CMD	14:7	8	Address Assignment CCC: This field is used to define the Address Assignment Command type used in the transfer. This field is used for representing the ENTDAA or SETDASA Common Command codes.
4	RSVD	15	1	Reserved
5	DEV_INDX	20:16	5	Device Index This field is used to indicate the start pointer of the Device Table from where the Dynamic Address is to be picked and assigned to the I^3C devices.
6	DEV_COUNT	25:21	5	Device Count This field is used to represent the number of devices to be assigned a Dynamic Address.
7	ROC	26	1	Response on Completion This field indicates whether the Response Status is required or not after the execution of this command for a successful transfer. • 1 - Response Status is required. • 0 - Response Status is not required. Note: The exception to this control is that the response status is generated when the transfer has encountered an error condition.
8	RSVD	29:27	3	Reserved
9	TOC	30	1	Termination On Completion This field controls whether a STOP need to be issued after the completion of the transfer. • 1 - STOP issued after this transfer. • 0 - The next transfer starts with RESTART condition.
10	RSVD	31	1	Reserved

24.4.4.4.2. Response Data Structure

Table 24-7. Response Data Structure

SL.No	Field Name	Bit Field	Width	Description
1	DATA_LENGTH	15:00	16	For Write transfers, this field represents the remaining data length of the transfer if the transfer is terminated early (remaining data length = requested data length - transferred data length). For Read transfers, this field represents the actual amount of data received in bytes. For the Address Assignment command, this field represents the remaining device count.
2	CCCT	23:16	8	CCC/HDR Header Type: This field represents the CCC type of the received vendor extension CCC packet or the HDR header of the received HDR transaction. This field indicates the CCC type when the TID is set to 4'b1111 (reserved for all other transactions). During the controller transactions and target non-HDR transactions, this field returns 8'b0000_0000 and can be considered as don't care.
3	TID	28:24	5	This field is used as the identification tag for the commands. The I^{3}C controller returns the ID received through commands. • 4'b0000 - 4'b0111 - User-Defined TID (specified in the Transfer/Address Assignment Command) • 4'b1000 - Controller Write Data Status (as target only) • 4'b1111 - DEFSLVS Status (as target only) • 4'b1001 - 4'b1110 - Reserved for I^{3}C controller
4	ERR_STS			Defines the Error Type of the processed command or received vendor extension CCC packet (Target mode). • 13–15: Reserved • 12: PEC Error. This bit is set if a PEC byte validation error occurs in read transfers when the TRANSFER_COMMAND[PEC] bit is set to 1. • 11: GETACCCR – Dynamic Address Mismatch Error. • 10: Reserved • 9: I^{2}C Target Write Data NACK Error/Write Early Terminate in HDR-DDR. • 8: Transfer Terminated • 7: Reserved • 6: Receive Buffer Overflow/Transmit Buffer Underflow (Only for HDR Transfers) • 5: Address NACKed. This bit is set in case the Target NACKs for Dynamic Address Assignment during the ENTDAA process. • 4: I^{3}C Broadcast Address NACK Error • 3: Frame Error • 2: Parity Error • 1: CRC Error • 0: No Error

24.4.4.4.3. IBI Status and Data Structure

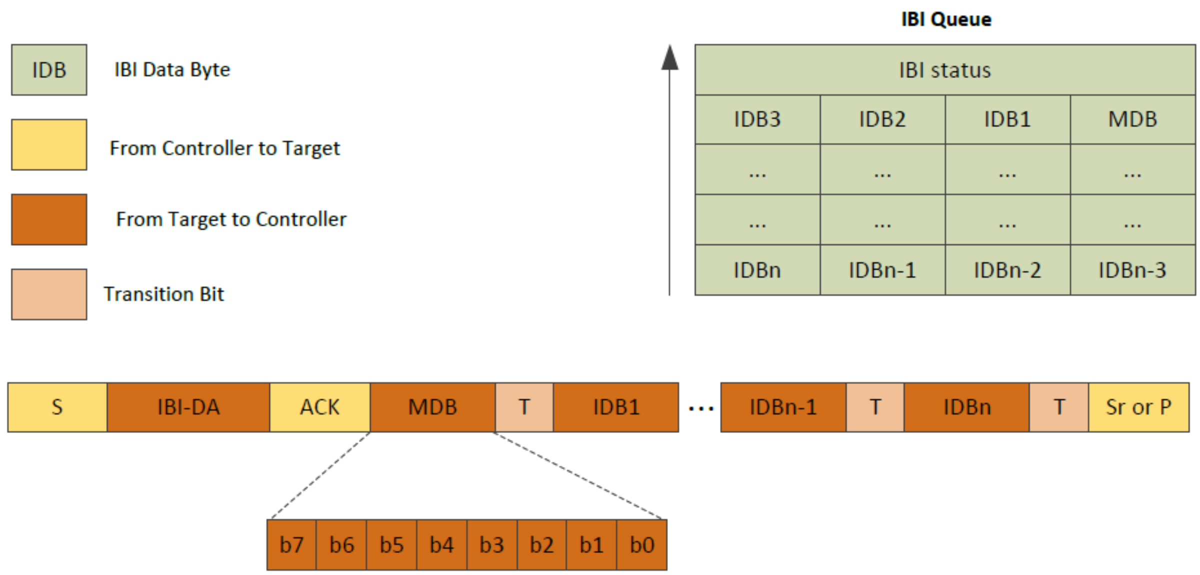
When an In-Band Interrupt (IBI) request is received, the status of the received IBI is placed into the IBI queue (I3CxIBIQUE) with the following data structure:

Table 24-8. IBI Status and Data Structure

Bits	Field Name	Access Type	Reset Value	Description
31	IBI_STS	R	0x0	IBI Status Indicates the status of the response returned for the received IBI. 1'b0: The received IBI is responded to with an ACK. Any non-zero value of the DATA_LEN field indicates the presence of a data payload for the ACK'd IBI. 1'b1: The received IBI is responded to with a NACK. An auto-disable CCC command is issued if the received IBI address is valid and matches the DAT entry. If an IBI is received from an unknown address (not a valid entry in DAT), the IBI_STS is set to 1.
30	ERROR	R	0x0	Error Indicates that during the IBI Auto command, an error is encountered and the data from the target device is partially or fully discarded. The following errors are detected: • CRC/Parity Error (applicable for HDR transaction) • Target Address NACK • 0x7E Address NACK • IBI buffer overflow (applicable for IBI HDR modes) 1'b0: No Error. Transaction complete with no errors. 1'b1: Error. Error encountered during the IBI Auto command phase.
29:26	RESV	R	0x0	—
25	TS	R	0x0	IBI Timestamp Present indicates whether a timestamp is available for the IBI. Values: • 1'b1: ON: IBI is timestamped • 1'b0: OFF: IBI is not timestamped.
24	LAST_STATUS	R	0x0	Last Status When set, this indicates that this status is the last for the received IBI. If the payload of the received SIR exceeds the programmed IBI data threshold, then the Controller splits the IBI payload into multiple chunks of IBI_DATA_THLD size (max), which includes the timestamp bytes if enabled.
23:16	RESV	R	8'h00	Reserved
15:8	IBI_ID	R	8'h00	IBI identifier This field indicates the target address byte along with the RnW bit received from the target device.
7:0	DATA_LENGTH	R	8'h00	IBI Data Length Number of data bytes received from the target device either along with the SIR or with the auto-command (read). The data length is limited by QUEUE_THLD_CTRL[IBI_DATA_THLD].

The Controller receives the bytes in the same order as on the bus and places the first byte received in the Least Significant Byte (LSB) position, and the following bytes in the LSB + 1 location, until all four bytes of a word are received in the IBI Queue. The data is always preceded by IBI status and indicates the number of valid IBI data bytes as shown in Figure 24-13.

Figure 24-13. IBI with Payload Data Structure



24.4.4.4.4. Device Address Table Data Structure

The device address table is used to store the addresses of the devices that are attached to the I^{3}C bus. It is referred to by the Address Assignment Command and the Transfer Command through the DEV_INDEX field for getting the programmed Target addresses.

The device address table is used to refer to the targeted address to initiate the transfers. Hence, the application must fill this table with the target addresses of the devices to address them. The I3CxDEVADDRTAByLOC1 register must be programmed.

24.4.4.4.5. Device Characteristics Table Data Structure

The device characteristics table is used to capture the received device characteristics information (PID, BCR, DCR) and the assigned dynamic address by the controller for the participating devices during ENTDAA CCC command execution (Active Controller mode) and the four bytes (per device) received during DEFSLVS CCC command reception (Non-Active Controller). The information captured during ENTDAA CCC can be used by the application to know what dynamic address is assigned to a particular target with the captured characteristics. The application should update the DAT table based on the characteristics received during the ENTDAA procedure. For example, whether or not a device is capable of sending an IBI payload indicated by (BCR[2]) should be reflected in the 'IBI_PAYLOAD' bit of the DAT table for that particular target.

The following two types of data structures go into the device characteristic table:

- Device characteristics table structure during ENTDAA, captured in I3CxDEVCHARTAB1LOC1, I3CxDEVCHARTAB1LOC3, I3CxDEVCHARTAB1LOC3 and I3CxDEVCHARTAB1LOC4.
- Device characteristics table structure during DEFSLVS, captured in I3CxSECDEVCHARTAB1

Figure 24-14. Device Characteristics Table Structure During ENTDAA

	31–24	23–16	15–8	7–0
I3CxDEVCHARTAB1LOC4	RESERVED			DA
I3CxDEVCHARTAB1LOC3	RESERVED		BCR[7:0]	DCR[7:0]
I3CxDEVCHARTAB1LOC2	RESERVED		PID[15:8]	PID[7:0]
I3CxDEVCHARTAB1LOC1	PID[47:40]	PID[39:32]	PID[31:24]	PID[23:16]

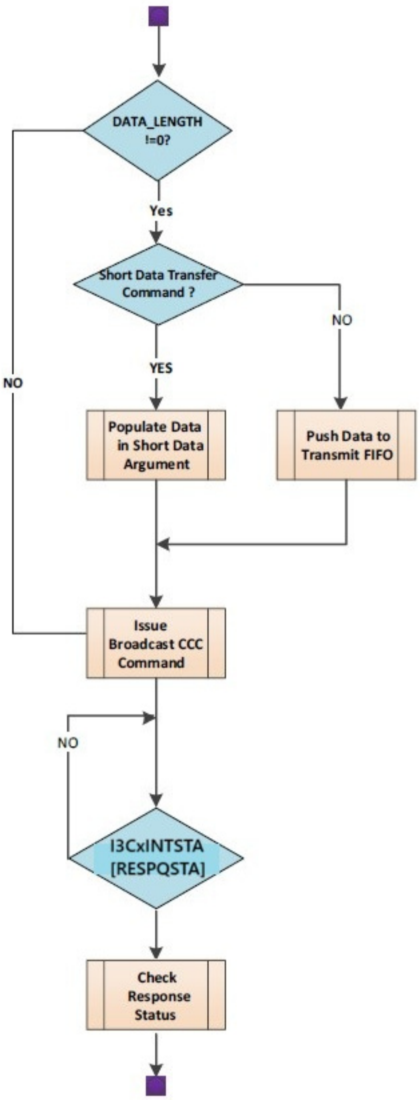
Figure 24-15. Device Characteristics Table Structure During DEFSLVS Command

	31–24	23–16	15–8	7–0
I3CxSECDEVCHARTAB1	DA1	DCR1	BCR1	SA1

24.4.4.5. Bus Transfer Initiation

24.4.4.5.1. Broadcast CCC Write Transfers

Figure 24-16. Broadcast CCC Transfer



The Broadcast CCC Write transfers are initiated on the bus based on the I3CxCMDQUE settings, as shown in Table 24-9.

Table 24-9. Broadcast CCC Write Transfer Required Programming Values

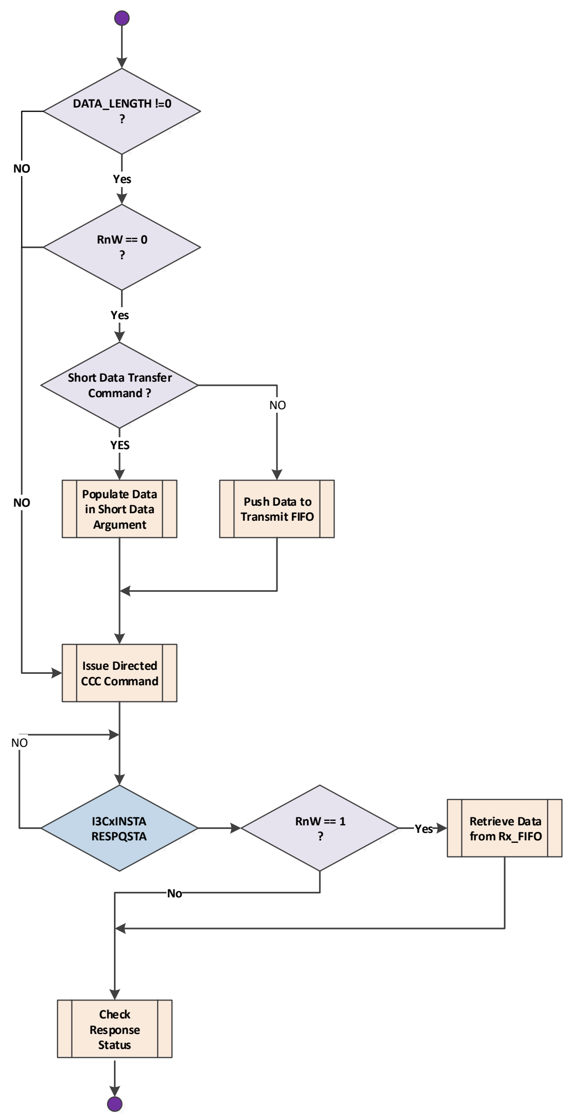
Command Attribute	Bit Field	Programmed Value	Description
Transfer Command	CP	1	Indicates to the controller to consider the CMD field.
	CMD[14]	0	Indicates to the controller that the transfer is a Broadcast CCC transfer.
	CMD[13:7]	0x0 – 0x7F	Indicates the CCC command to be transferred.
	DEV_INDX	NA	This field is not used since the transfer is a Broadcast CCC.
	SPEED	0 (SDR0) or 7 (I2C FM)	Indicates to the Controller that the transfer should go in SDR0 Speed/ Mode or I^2C FM Speed. **Note:** All CCC transfers are initiated with SDR0 Speed in I^3C bus topology. CCC transfers can be either initiated with SDR0 or I^2C FM speed based on the context of the bus in JEDEC bus topology.
	DBP	0 or 1	Indicates whether the defining byte is present for Broadcast CCC transfer or not. 1: Defining byte is present 0: Defining byte is not present
	SDAP	0 or 1	1: Indicates the Controller to consider the transmit data from the command if RnW is set to 0. 0: Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0.
	RnW	0	Indicates whether the transfer is a write or read transfer. 1: Read Transfer 0: Write Transfer
	TGT_RST	0 or 1	Indicates whether the Controller should generate Target Reset Pattern at the end of the Broadcast RSTACT CCC transfer. 1: Target Reset Pattern is generated. 0: Target Reset Pattern is not generated.
	PEC	0 or 1	Indicates whether Packet Error Check is enabled for Broadcast CCC transfer. 1: PEC check is enabled. 0: PEC check is disabled.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Short Data Argument (SDAP=1)	BYTE_STRB	0, 1, 3,7	Indicates that the respective data bytes of the Immediate command are valid.

Notes:

1. The data in Tx-FIFO (Transfer Argument) or Data Bytes (Short Data Argument) are not required for some broadcast CCCs which do not have payload data indicated through Data Length (Transfer Argument) or BYTE_STRB (Short Data Argument). If the Broadcast CCC does not consist of a payload, then you must indicate it with zero in either the Data Length or BYTE_STRB fields based on the command issued.
2. The Controller halts in case of receiving a NACK (no I3C device on the bus) for the address header of the Broadcast CCC transfer. The Controller updates the 'ERR_STS' field with appropriate error information in the Response status (I3CxRESPQUE), halts the Controller, and gives back control to the application to resume the operation of the Controller by writing '1' to the I3CxCNTRL [RESUME] bit after taking suitable action.

24.4.4.5.2. Directed Write and Read Transfers

Figure 24-17. Directed CCC Transfer



The Directed CCC Write/Read transfers are initiated on the bus based on the I3CxCMDQUE settings shown in Table 24-10.

Table 24-10. Directed Write and Read Transfers

Command Attribute	Bit Field	Programmed Value	Description
Transfer Command	CP	1	Indicates to the controller to consider the CMD field.
	CMD[14]	1	Indicates to the controller that the transfer is a Directed CCC transfer.
	CMD[13:7]	0x0 - 0x7F	Indicates the CCC Command to be transferred.
	DEV_INDX	DEV_INDX	Indicates the index of the device table, which consists of the target address to be targeted.
	SPEED	0 (SDR0) or 7 (I2C FM)	Indicates to the controller that the transfer should go in SDR0 Speed/Mode or I2C FM Speed. **Note:** All CCC transfers are initiated with SDR0 Speed in I3C bus topology. CCC transfers can be either initiated with SDR0 or I2C FM Speed based on the context of the bus in JEDEC bus topology.
	DBP	0 or 1	Indicates whether the Defining Byte is present for Directed CCC transfer or not. 1: Defining Byte is present. 0: Defining Byte is not present.
	SDAP	0 or 1	1: Indicates that the controller should consider the transmit data from the command if RnW is set to 0. 0: Indicates that the transmit data from the Transmit FIFO should be considered if RnW is set to 0.
	RnW	0 or 1	1: Indicates the transfer is a read transfer. 0: Indicates the transfer is a write transfer.
	TGT_RST	0 or 1	Indicates whether the controller should generate a Target Reset Pattern at the end of a Directed RSTACT CCC transfer. 1: Target Reset Pattern is generated. 0: Target Reset Pattern is not generated. Indicates whether Packet Error Check is enabled for Directed CCC transfer.
	PEC	0 or 1	1: PEC check is enabled. 0: PEC check is disabled.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Short Data Argument (SDAP=1)	BYTE_STRB	0,1,3,7	Indicates the respective data bytes of the Immediate command are valid.

Note: The data in Tx-FIFO (I3CxTXRXDATA) or data bytes (Immediate Data Transfer) are not required for some directed write CCCs which do not have payload data, as indicated through data length (Transfer Argument) or BYTE_STRB (Short Data Argument). If the directed CCC does not consist of a payload, indicate this with a zero in either the data length or BYTE_STRB fields based on the command issued.

Directed CCC Transfer Targeted to Multiple Targets

Each transfer command initiates a directed CCC transfer to only one Target since it consists of only one DEV_INDX. If the requirement is to transfer the directed CCC command to multiple devices, then you must pipeline the multiple transfer commands in the I3CxCMDQUE with TOC bits set to zero and with the different DEV_INDX fields pointing to multiple targets. The Controller decodes the pipelined 'Transfer command' during the transfer of directed CCC transfer and decides the next transfer based on the following:

- If the current command and the pipelined command have the same Directed CCC, then the Controller targets the next Target without ending the CCC command.
- If the current command and the pipelined command are not the same Directed CCC, then the Controller ends the CCC command and starts issuing the next transfer as indicated by the pipelined transfer command.

The application can set the ROC bit to ‘0’ for the subsequent directed CCC commands and enable the ROC bit in the last CCC command if the directed CCC transfer is targeted to multiple devices to avoid unnecessary responses.

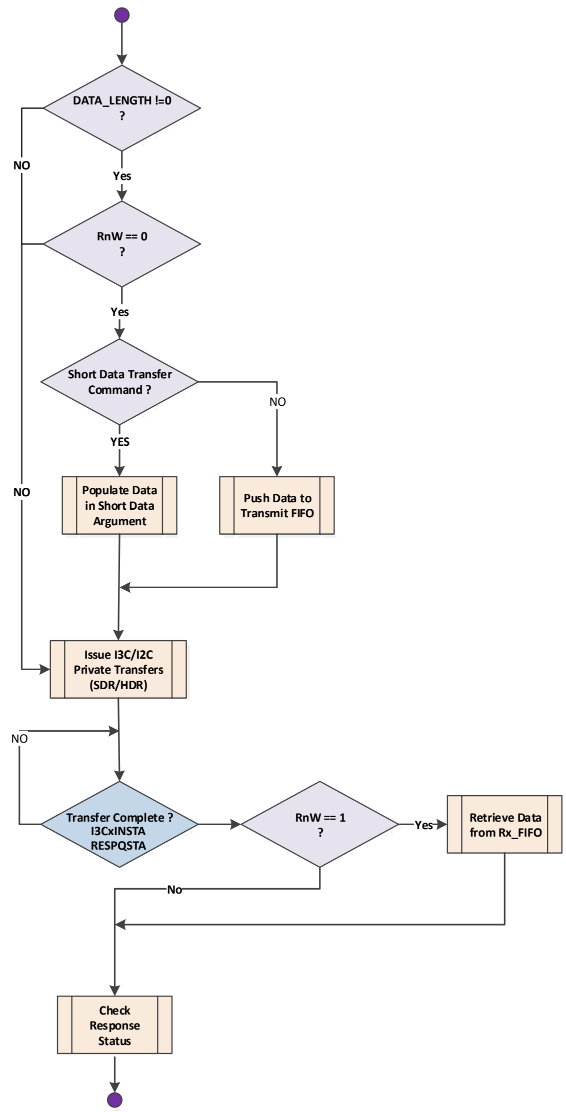
The Controller halts in case of the following conditions:

- Receiving a NACK for the address header of the directed CCC transfer (It means no I^3C device on the bus).
- Receiving a NACK for the Target address of the directed CCC transfer.

The Controller updates the ‘ERR_STS’ field with appropriate error information in the response status (I3CxRESPQUE), halts the Controller, and returns control to the application to resume the operation of the Controller by writing ‘1’ to the I3CxCTRL [RESUME] bit.

24.4.4.5.3. I^{3}C Private Write or Read Transfers

Figure 24-18. Private Transfer



The I^{3}C private write and read transfers are initiated on the bus based on the I3CxCMDQUE and settings shown in Table 24-11.

Table 24-11. I³C Private Write or Read Transfers

Command Attribute	Bit Field	Programmed Value	Description
Transfer Command	CP	0	Indicates to the controller not to consider the CMD field.
	CMD[14]	NA	This field is not applicable since the CP bit is set to 0.
	CMD[13:7]	NA	This field is not applicable since the CP bit is set to 0.
	DEV_INDX	DEV_INDX	Indicates the index of the device table, which consists of the target address to be targeted.
	SPEED	0 to 4	Indicates to the controller that the transfer should proceed in SDR mode.
	SDAP	0 or 1	1: Indicates the controller should consider the transmit data from the command if RnW is set to 0. 0: Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0.
	RnW (Read and Write)	0 or 1	1: Indicates the transfer is a read transfer. 0: Indicates the transfer is a write transfer.
			Indicates whether Packet Error Check is enabled.
			Enabled for private SDR transfers.
	PEC	0 or 1	0: PEC check is disabled.
			1: PEC check is enabled.
Transfer Argument (CMD_ATTR=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Short Data Argument (CMD_ATTR=1)	BYTE_STRB	0,1,3,7	Indicates that the respective data bytes of the Immediate command are valid.

Notes:

1. To avoid the initial latencies of the transfer, the controller uses TXSTART/RXSTART in the I3CxBUFTHLD register before initiating the transfer. TXSTART ensures the threshold level of data is present in the Transmit buffer for write transfer, and RXSTART ensures the level of space is available in the Receive buffer for the Read transfer before initiating the transfer. This threshold is only applicable for the transfers that are initiated with the START condition and not applicable for the transfers that are initiated with the RESTART condition for SDR Transfers.
2. To allow priority for IBI from the devices, the controller can include the Address header for the I³C private transfers by enabling the IBA bit in the 'I3CxCTRL' register, since IBI always wins when arbitrated with the Address header.

The controller halts in the following conditions:

- Receiving a NACK for the address header of the private transfers if the I3CxCTRL[IBA] bit is enabled.
- Receiving a NACK for the target address of the private transfers.

The controller updates the 'ERR_STS' field with appropriate error information in the response status (I3CxRESPQUE), halts the controller, and returns control to the application to resume the operation of the controller by writing '1' to the I3CxCTRL [RESUME] bit.

24.4.4.5.4. I^{2}C Private Write or Read Transfers

The I^{2}C private write and read transfers are initiated on the bus based on the I3CxCMDQUE settings, as shown in Table 24-12.

Table 24-12. I2C Private Write or Read Transfers

Command Attribute	Bit Field	Programmed Value	Description
Transfer Command	CP	0	Indicates to the controller not to consider the CMD field.
	CMD[14]	NA	This field is not applicable since the CP bit is set to 0.
	CMD[13:7]	NA	This field is not applicable since the CP bit is set to 0.
	DEV_INDX	DEV_INDX	Indicates the index of the device table, which consists of the target address to be targeted.
	SPEED	0 or 1	Indicates to the controller that the transfer should go in I^{2}C SDR mode. Values (I^{2}C mode): 0x0: I2C Fm 0x1: I2C Fm+
	SDAP	0 or 1	0: Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0. 1: Indicates the controller to consider the receive data if RnW is set to 1.
	RnW (Read and Write)	0 or 1	Transmit data from the command if RnW is set to 0. 1: Indicates the transfer is a read transfer. 0: Indicates the transfer is a write transfer.
	PEC	0 or 1	Indicates whether Packet Error Check is enabled for Private SDR transfers. 1: PEC check is enabled. 0: PEC check is disabled.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65535	Indicates the transfer length of the transfer.
Short Data Argument (SDAP=1)	BYTE_STRB	0,1,3,7	Indicates that the respective data bytes of the Immediate command are valid.

Note: The I3CxDEVADDRTAB1LOC1[DEVICE] bit in the device address table pointed to by the 'DEV_INDX' field of the Transfer Command must be set to '1' for the I^{2}C private transfers. The controller uses the I^{2}C protocol to initiate the I^{2}C transfers for the Legacy I^{2}C devices. To allow the priority for the IBI from the I3C devices, the controller can include the address header for the I^{2}C private transfers by enabling the IBA bit in the 'I3CxCTRL' register.

24.4.4.5.5. High Data Rate-Double Data Rate (HDR-DDR) Transfers

The HDR-DDR write and read transfers are initiated on the bus based on the I3CxCMDQUE and settings shown in Table 24-13.

Table 24-13. High Data Rate-Double Data Rate (HDR-DDR) Transfers

Command Attribute	Bit Field	Programmed Value	Description
Transfer Command	CP	1	Indicates to the controller to consider the CMD field.

Table 24-13. High Data Rate-Double Data Rate (HDR-DDR) Transfers (continued)

Command Attribute	Bit Field	Programmed Value	Description
	CMD[13:7]	0x00 – 0xFF	This field indicates either a Write or Read command used in the command code of HDR Transfer. • 0x00 – 0x1F: I3C Reserved Write commands • 0x20 – 0x7F: I3C Vendor Write commands • 0x80 – 0x9F: I3C Reserved Read Commands • 0xA0 – 0xFF: I3C Vendor Read Commands
	DEV_INDX	DEV_INDX	Indicates the index of the device table, which consists of the target address to be targeted.
	SPEED	6 (HDR-DDR)	Indicates to the Controller that the transfer should go in HDR-DDR mode.
	SDAP	0	0 - Indicates to consider the transmit data from the Transmit FIFO if RnW is set to 0.
	RnW (Read and Write)	0 or 1	• 1 - Indicates the transfer is a read transfer. Note: This bit is used for the Write or Read command used in the HDR-CMD (CMD[14]) field. • 0 - Indicates the transfer is a write transfer.
Transfer Argument (SDAP=0)	DATA_LENGTH	0 - 65564	Indicates the transfer length of the transfer.

The HDR Transfers are supported only with the 'Transfer Argument command' and not with the 'Short Data Argument command' since, at most, two words can be transferred in the Short Data Argument command format, and it really does not justify the overhead in terms of ENTHDR* CCC, HDR Command, entry and exit patterns.

The controller generates the 'ENTHDR0' CCC to enter into the HDR-DDR mode and forms the Command code through the CMD field of the Transfer command and target address. The preamble and parity bits are generated by the controller as per the protocol and sent along with the write data word. The word from the Tx-FIFO is appended with the parity bits and preamble bits to form the Write Data word and transmits the data. The HDR-RESTART and HDR-EXIT patterns are generated in place of the RESTART and STOP conditions.

To avoid the initial latencies of the transfer, the controller uses TXSTART/RXSTART in the I3CxBUFTHLD register before initiating the transfer. TXSTART ensures the threshold level of data is present in the Transmit buffer for a write transfer, and RXSTART ensures the level of space is available in the Receive buffer for a read transfer before initiating the transfer. This threshold is only applicable for transfers initiated with the START condition and not applicable for transfers initiated with the RESTART condition. If the threshold amount of write data in the Tx-FIFO or empty space in the Rx-FIFO is not available, the controller waits until the threshold amount of data is available to initiate the transfer. If the threshold amount of data is not available for the HDR-DDR transfer initiated, which must be initiated with RESTART, then the controller generates an EXIT pattern followed by STOP for the current transfer and waits for the threshold amount of data to initiate the next transfer with a START condition.

The controller halts in the case of the following conditions:

- Receiving NACK for the address header of the private transfers.
- Receiving NACK for the target address (HDR-CMD) of the HDR-DDR private transfers.
- If the controller experiences Transfer Underflow or Receive Overflow during the HDR-DDR transfers.
- After executing a software-initiated terminate.

- The controller decodes the parity bits, CRC byte, frame mismatch, and validates them for the read transfer and reports in the 'ERR_STS' field of the response status (I3CxRESPQUE) if there is any error.

The controller updates the 'ERR_STS' field with appropriate error information in the response status, halts the controller, and gives back control to the application to resume the operation of the controller by writing '1' to the Resume bit of the I3CxCTRL register.

Notes:

1. In SDR mode, the Controller extends the clock by pulling the SCL low when the TX-FIFO is empty, or the RX-FIFO is full in the middle of the transaction.
2. In HDR-DDR mode, the Controller terminates the transfer when the TX-FIFO is empty, or the RX-FIFO is full in the middle of the transaction.
3. When TOC in Transfer Command *is set to 0: In HDR-DDR mode, if the Start Threshold of the Next Transfer is not met, HDR-EXIT is generated.*

24.4.4.6. Clock Stalling

The controller stalls the SCL clock during specific scenarios of non-HDR transfers. This is to accommodate intermittent system latencies during command pipelining, transmit data pre-fetching, response reading, and so on. The controller also provides additional knobs in the form of Start Thresholds and Buffer Thresholds to avoid clock stalling.

Clock stalling is a feature that helps avoid overrun and underrun errors in the case of SDR, notwithstanding all of the mentioned measures. The controller software receiving these errors is an indication of a larger issue in itself, and neither clock stalling by itself nor termination of transfers during clock stalling is the solution.

Table 24-14. Clock Stalling Conditions

Sl No.	Transfer Command	Previous Command Condition	Condition to Enter Clock Stalling
I^3C/I^2C Transfer, ACK/NACK Phase			
1	Write Transfer Regular command	TOC bit is set to '0'.	TX-FIFO is empty.
2	Read Transfer Regular command	TOC bit is set to '0'.	RX-FIFO is full.
3	Follow-up Directed CCC command without payload (Immediate/Regular) (Not 1st Directed CCC command)	Previous Directed CCC command TOC bit is set to '0'.	CMD-QUEUE is empty.
4	Broadcast CCC Transfer with Regular command	TOC bit is set to '0'.	TX-FIFO is empty.
5	Directed CCC Write Transfer Regular command	TOC bit is set to '0'.	TX-FIFO is empty.
6	Directed CCC Read Transfer Regular command	TOC bit is set to '0'.	RX-FIFO is full.
7	Directed CCC command without Payload and ROC bit is set to '1'	TOC bit is set to '0'.	RESP-Queue is full.
8	Middle of I^2C Write Transfer with Regular Transfer command	NA	TX-FIFO is empty.
9	Middle of I^2C Write Transfer with Regular Transfer command	NA	RX-FIFO is full.
10	End of I^2C Write Transfer Regular command (only controller terminates) and TOC bit set to '0'.	NA	Next command unavailable.
11	End of I^2C Write Transfer Regular command (only controller terminates) and ROC bit is set to '1'.	NA	RESP-Queue is full.
12	End of I^2C Read Transfer Regular command (either controller/target terminates) and TOC bit set to '0'.	NA	Next command unavailable.
13	End of I^2C Read Transfer Regular command (only controller terminates) and ROC bit is set to '1'.	NA	RESP-Queue is full.

Table 24-14. Clock Stalling Conditions (continued)

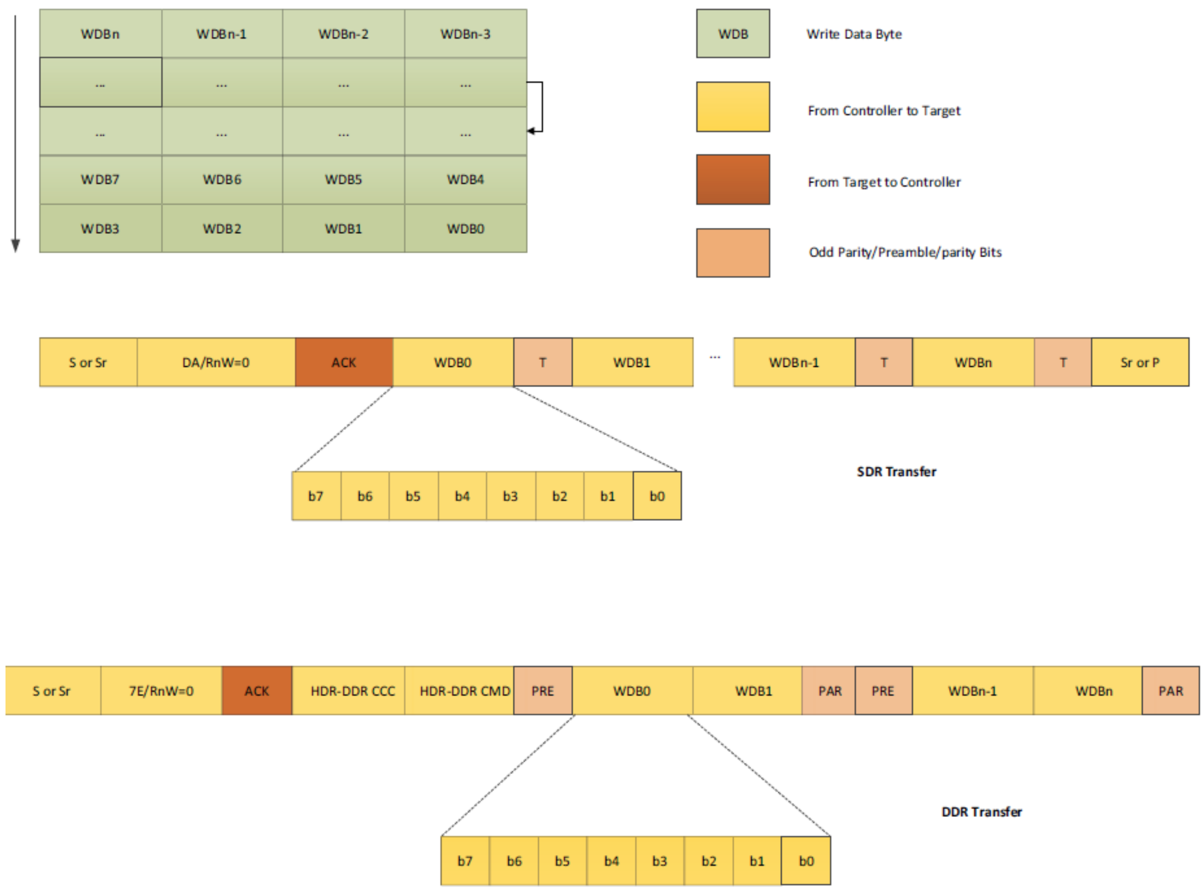
Sl No.	Transfer Command	Previous Command Condition	Condition to Enter Clock Stalling
Write Data Transfer, Parity bit			
14	Middle of Write Transfer Regular command	NA	TX-FIFO is empty.
15	End of Write Transfer Regular command (only controller terminates) and TOC bit set to '0'.	NA	Next command unavailable.
16	End of Write Transfer Regular command (only controller terminates) and ROC bit is set to '1'.	NA	RESP-Queue is full.
I3C Read Transfer, Transition bit			
17	Middle of Read Transfer Regular command	NA	RX-FIFO is full.
18	End of Read Transfer Regular command (either controller/target terminates) and TOC bit set to '0'.	NA	Next command unavailable.
19	End of Read Transfer Regular command (only controller terminates) and ROC bit is set to '1'.	NA	RESP-Queue is full.
I3C IBI Transfer, Transition bit			
20	Middle of IBI Read Data Transfer	NA	IBI-Data FIFO is full.
21	Middle of Auto Command Read Data Transfer	NA	IBI-Data FIFO is full.

24.4.4.7. Data Packing and Unpacking

24.4.4.7.1. Transmission Data

The data to be transmitted is expected to be provided to the controller in the Transmit FIFO, as shown in Figure 24-19. The controller places the Least Significant Byte (LSB) of data as the first byte on the I3C bus, followed by successive bytes.

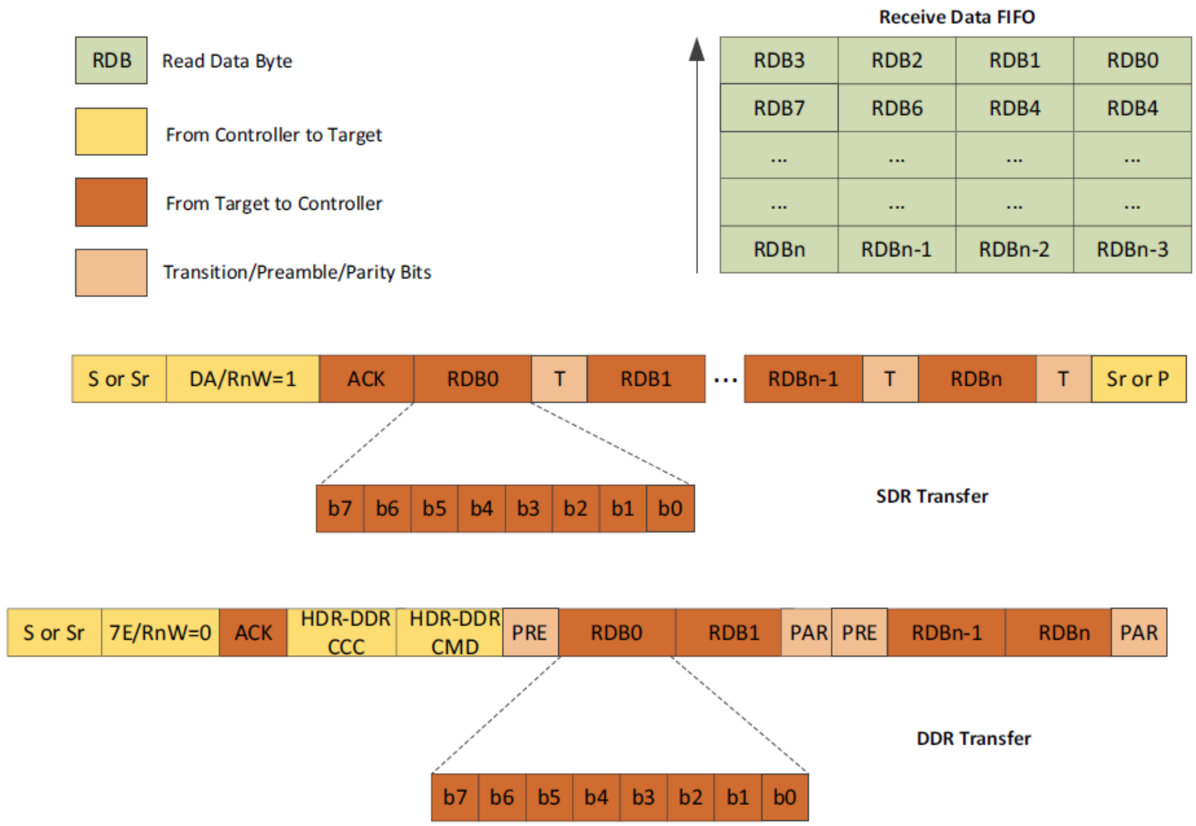
Figure 24-19. Byte Ordering - Transmit Data



24.4.4.7.2. Received Data

The Controller receives the bytes in the same order as on the bus and places the first byte received in the LSB position, and the following bytes in the LSB + 1 location, until all four bytes of a word are received in the Receive FIFO, as shown in Figure 24-20.

Figure 24-20. Byte Ordering - Receive Data



The Controller does not distinguish between read data coming from a private read transfer and read data coming from a CCC. The Controller is also transparent to CCC; it does not decode the CCC as a Controller and does not treat CCCs differently from one another. The data packing logic for CCCs also follows the same rules as mentioned for private transfers. Therefore, for CCCs, it is expected that the software performs a byte-swap while reading the data to get the required byte endianness.

Figure 24-21 and Figure 24-22 show the byte-swapping to be performed by software for GETSTATUS CCC and GETPID CCC, respectively.

Figure 24-21. CCC - GETSTATUS (0x90)

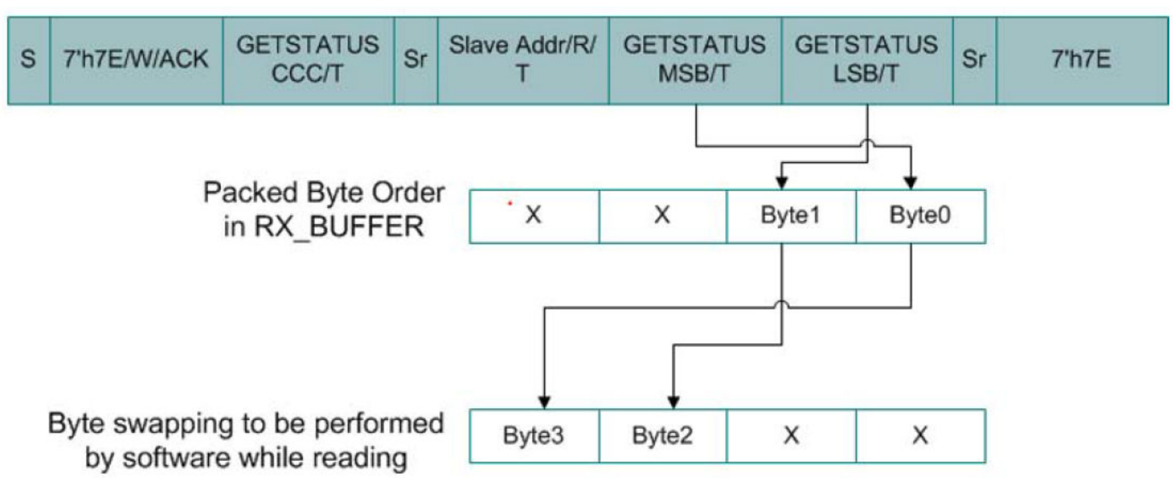
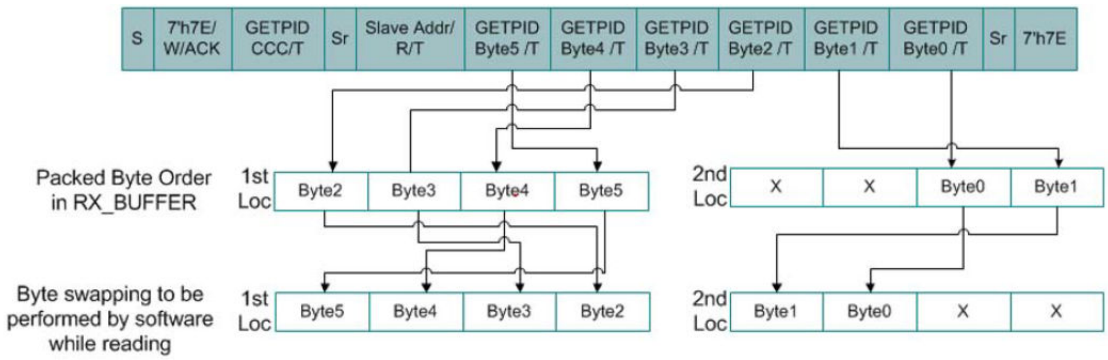


Figure 24-22. CCC - GETPID (0x8D)



24.4.4.8. Error Detection

This section describes the error detection methods that are applicable to Controller mode.

24.4.4.8.1. Detecting Error Type in the Processed Commands

The ERR_STATUS field in the I3CxRESPQUE register is used to detect the error type in the processed commands. This field indicates errors based on the codes shown in Table 24-15.

Table 24-15. Error Type in the Processed Commands

ERROR_CODE	ERROR_TYPE	ERROR_DESCRIPTION
0x0	NO_ERR	The initiated transfer was successful without any errors.
0x1	CRC_ERR	A CRC error occurred in the HDR-DDR read transfer.
0x2	PARITY_ERR	A parity error occurred in HDR-DDR read transfers.
0x3	Reserved	Reserved
0x4	ADDR_HDR_NACK_ERR	Received NACK for the address header. This indicates no I^3C target is present in the system.
0x5	ADDR_NACK_ERR	Received NACK for target address of write/read transfer or Received NACK for assign address of ENTDAA Transfer.
0x6	OVL_URL_ERROR	Experienced Receive FIFO Overflow in HDR Transfers or Experienced Transmit FIFO Underflow in HDR Transfers
0x7	Reserved	Reserved
0x8	ABORT_ERR	The transfer is terminated based on the user-initiated terminate.
0x9	I2C_WR_DATA_NACK_ERR	Received NACK for the I^2C Write Data Transfer.
0xC	PEC_ERROR	A PEC byte validation error occurs in read transfers when the TRANSFER_COMMAND[PEC] bit is set to 1.

24.4.4.8.2. SDR Error Detection and Recovery Methods

There are three error types for SDR transfers and their recovery methods, as listed in Table 24-16.

Table 24-16. SDR Error Detection and Recovery

Error Type	Description	Error Detection Method	Error Recovery Method
CE0	Transaction after sending CCC	Detects illegally formatted CCCs.	Stop the transmission, then send STOP and retry the transmission.
CE2	No response to broadcast address (7'h7E)	Controller detects NACK after broadcast address (7'h7E) transmission.	Upon detection of NACK, the controller transmits the HDR Exit Pattern followed by STOP.

Table 24-16. SDR Error Detection and Recovery (continued)

Error Type	Description	Error Detection Method	Error Recovery Method
CE3	Failed controller handoff	Active Controller detects that the new controller does not drive the bus after handoff.	Active Controller regains the controller role and drives the bus to assert its controller role.

Error Type CE0

If the controller detects an illegally formatted CCC, then the controller stops the transmission, sends STOP, and tries the transmission again.

An example of an illegally formatted CCC is when the controller receives just one byte from the target in a GETMWL CCC code. The controller expects two bytes and does not decode the CCC as it is transparent for the application. The controller always initiates the CCC transfer and expects the number of bytes as mentioned in the DATA_LENGTH field of the Transfer Argument. If the target terminates earlier than the DATA_LENGTH expected by the controller, the controller represents it in the DATA_LENGTH of the RESPONSE.

The application has to decode this information through the RESPONSE of the CCC transfer and can re-issue the same command in the COMMAND_QUEUE for re-transmission.

Error Type CE2

If the controller does not receive an ACK for a transmitted broadcast address (7'h7E), then it transmits the HDR Exit Pattern followed by STOP. The controller always generates STOP upon receiving a NACK for the broadcast address (7'h7E) and enters the HALT state. This can be detected through the I3CxINTSTA[TRANSERR].

Error Type CE3

After the Active Controller prepares for the handoff, the Active Controller issues a GETACCCR CCC followed by a STOP. Following the STOP and Bus Available Condition, the selected secondary controller assumes the role of the Active Controller and takes control of the I^3C Bus. The former Active Controller should monitor the I^3C Bus to ensure that the new Active Controller asserts its Controller Role. After the handoff procedure, the new Active Controller (NCR) issues a START to assert its Controller Role within 100µs or the time interval indicated by its Activity State (as reported by the GETMXDS CCC with optional Defining Byte), whichever is greater. If the new Active Controller does not issue a START, then it responds to SDA being pulled Low by pulling SCL Low; this is a START, and the new Active Controller may follow it with any valid Bus activity that is sufficient to assert the Controller Role. If the new controller does not respond by pulling SCL Low within that period, then the former Active Controller pulls SCL Low to regain the Bus Controller Role.

Defining Byte Support

The Defining Byte Support feature is an extended feature of the existing CCC transfers model. The Defining Byte is included after the CCC Code and before the payload (in broadcast CCCs)/Restart (Directed CCCs) to indicate an extended capability or an action associated with the CCC. The Defining Byte is supported for both Broadcast and Directed SDR CCC transfers. In both cases, the Defining Byte is included after the CCC Code byte as a sub-command before the Payload. Figure 24-23 and Figure 24-24 describe the placement of the Defining Byte in Broadcast CCC and Directed CCC transfers, respectively.

Figure 24-23. Broadcast CCC with Defining Byte

START S (Sr)	Address Header (0x7E,w)	Broadcast CCC Code (CCC,T)	Defining Byte (DF_Byte,T)	Data Payload (DB,T)	STOP (P/Sr)

Figure 24-24. Directed CCC with Defining Byte

START (S/Sr)	Address Header (0x7E,w)	Directed CCC Code (CCC,T)	Defining Byte (DF_Byte,T)	RESTART (Sr)	...

...	Target Address (SA,w/r)	Data Payload (DB,T)	STOP (P/Sr)

The Defining Byte is enabled for Broadcast or Directed SDR CCC transfers by setting the 'Defining Byte Present (DBP)' bit in the Transfer Command, and the Defining Byte value is captured from the 'Defining Byte (DB)' field of the Transfer Argument or Short Data Argument.

24.4.4.9. Packet Error Check

The Packet Error Check feature enables the generation and validation of an 8-bit CRC on the data stream between the Controller and Target for reliable communication. The transmitter always inserts an extra PEC byte at the end of the data stream, and the receiver validates it. The PEC is applicable for SDR CCC, IBI and SDR Private Transfers. The Controller or device implements an 8-bit Packet Error Code (PEC). The PEC is a CRC-8 value calculated on all the message bytes except for START, REPEATED START, STOP conditions or T-bits, ACK and NACK, and IBI header (7'h7E followed by W=0) bits. The polynomial for CRC-8 calculation is 'X8 + X2 + X1 + 1'. The PEC byte is supported only for SDR Transfers and is not applicable for HDR Transfers.

Figure 24-25. Broadcast CCC with PEC Byte

START S (Sr)	Address Header (0x7E,w)	Broadcast CCC Code (CCC,T)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)

Figure 24-26. Directed CCC with PEC Byte

START S(Sr)	Address Header (0x7E,w)	Direct CCC Code (CCC,T)	PEC Byte (PEC,T)	RESTART (Sr)

Target Address (SA,w)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)

Figure 24-27. Private Write Transfer with PEC Byte

START S (Sr)	Target Address (SA,w)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)

Figure 24-28. Private Read Transfer with PEC Byte

START S (Sr)	Target Address (SA,r)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)

The Packet Error Check can be enabled for SDR Transfers by setting the 'PEC' bit of the Transfer Command to '1'. When this bit is set, the Controller generates a PEC byte at the end of a write transfer and validates the PEC byte at the end of a read transfer. If the received PEC byte for the read transfer does not match the internally generated PEC byte based on the data received from the Target, then the Controller generates an I3CxINTSTA[TRANSERR] interrupt and indicates it as a 'PEC Error' in the response status.

24.4.4.10. BUS RESET Generation

To prevent a malfunctioning device from locking up the I3C/Sideband bus, the I3C protocol and JEDEC Sideband Specifications define a bus protocol RESET mechanism. The Controller supports the generation of the following three standalone patterns:

- Timed Reset
- HDR-EXIT Pattern
- Target Reset Pattern

The Reset Pattern is selected in the I3CxRSTCON[RSTTYP] bits, and enable I3CxRSTCON[BUSRST] to generate the Reset. The Controller considers this request only when it is in the IDLE state and gives priority over scheduled regular commands. The Controller generates the selected Reset pattern based on the selection in I3CxRSTCON[RSTTYP].

3'b000: HDR Exit Pattern

3'b110: Timed Reset (Drives SCL Low for the I3CxMSTTIMOUT [TIMOUTCNT] period of time)

3'b001: Target Reset Pattern

Once the Controller completes the Requested Reset Pattern, the Controller auto-clears the I3CxRSTCON[BUSRST] request bit. The software can poll this request bit to know whether the Controller has completed the generation of the Requested Reset pattern. An interrupt is generated, indicating that the Request Reset pattern has been issued.

Figure 24-29, Figure 24-30 and Figure 24-31 show the SCL Timed Reset, HDR-EXIT Pattern and Target Reset Pattern, respectively.

Figure 24-29. SCL Low Timeout

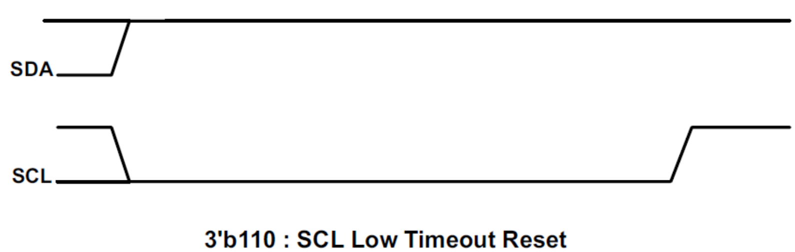


Figure 24-30. HDR Exit Pattern

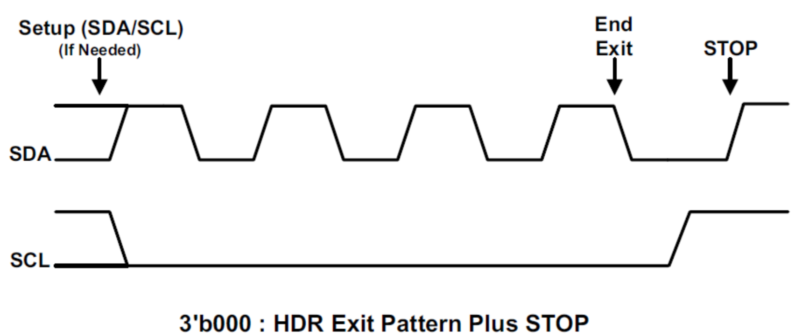
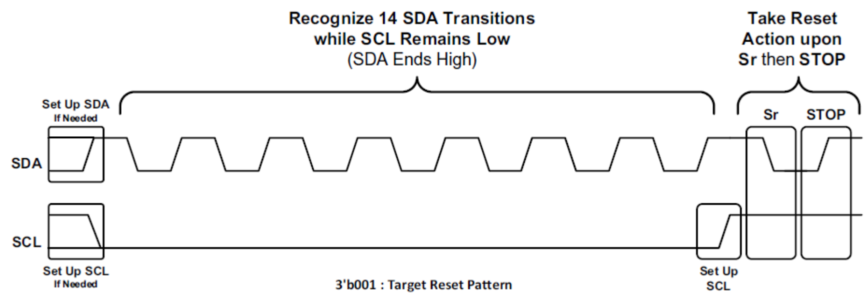


Figure 24-31. Target Reset Pattern



24.4.4.11. Target Reset

The Target Reset mechanism allows the Controller to reset one or more selected targets and avoids resetting all targets. Target Reset supports different levels of Reset, ranging from resetting only the I^{3}C Peripheral within a Target to resetting the whole Target device, communicated through the RSTACT CCC.

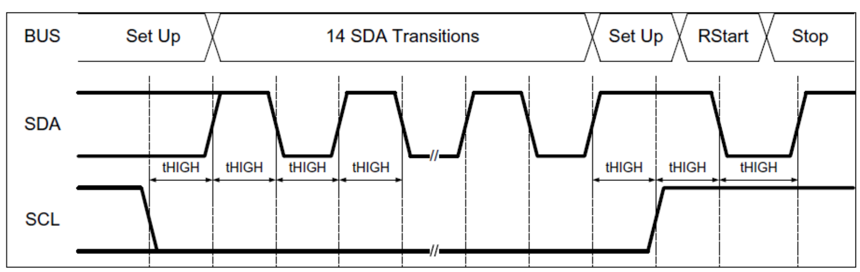
The Target Reset Pattern begins with fourteen SDA transitions while SCL is kept low and ends with a Repeated START followed by a STOP.

The Target Reset Pattern can either be generated as a Single Pattern or generated along with the RSTACT CCC transfer.

- Single Target Reset Pattern Generation: Generated by programming the I3CxRSTCON[RSTTYP] bits = 3’b001 and I3CxRSTCON[BUSRST] =1.
- Target Reset Pattern along with RSTACT CCC: Generated by programming the TGT_RST bit of the Transfer command.

The typical waveform of the Target Reset Pattern and its related timing parameters are described in Figure 24-32

Figure 24-32. Target Reset Pattern and Timing



24.4.4.12. Group Address Support in Controller

The group address feature enables multiple I^{3}C Target devices to share a single group address or multiple group addresses, allowing the Controller device to send a given I^{3}C Message to all Target devices in the group at once rather than one at a time.

Controller supports group address related CCCs (SETGRPA CCC, RSTGRPA CCC, DEFGRPA CCC) as vendor-specific CCCs with the existing CCC transfer model.

Store the group addresses in the device address table as shown in Figure 24-33.

Figure 24-33. Data Bytes Format for GETCAPS CCC without Defining Byte

31	30	29	23	16	0
Device (0)	DEV_NACK_RETRY_COUNT	RSVD	Group Address	RSVD	

The Controller can write to a group address just as it would to any I^{3}C Dynamic Address. However, the Controller does not attempt to read data from I^{3}C Devices using a group address. Both rules apply to private SDR and HDR Messages, and to Broadcast and Direct CCC commands.

24.4.5. Target Mode

The I^{3}C Target mode listens to the I^{3}C Bus for relevant I^{3}C Commands (CCCs) sent by the Active Controller and responds accordingly. This includes all Broadcast Commands (CCC) and any Directed Commands (CCC) addressed specifically to that I^{3}C Target device. An I^{3}C Target device does not generate the Bus clock and always follows the Bus clock generated by the Active Controller.

Target mode is enabled by setting I3CxCNTRLEXT [DEVOPMOD] to 1 before enabling the module (I3CxCNTRL [ENABLE]).

Note: To enter Target mode, it is necessary to provide SCL clocks to make it come out of Reset. This can be achieved by scheduling any transfer (placeholder transfer) ending in a STOP before initiating valid transfers. The placeholder transfer is ignored by the Target.

If the Target is selected as a Static Address device, then the device responds to ENTDAA, SETDASA and SETAASA CCC commands from the Active Controller until the Dynamic Address is assigned successfully.

If the Target is selected as a dynamic address device, then the device responds only to the ENTDAA CCC command from the Active Controller until the Dynamic Address is assigned successfully.

Once the dynamic address is assigned and valid, the Target stops responding to the ENTDAA, SETDASA, and SETAASA CCC commands until the Dynamic Address is reset through the RSTDAA CCC command.

SETNEWDA CCC from the Active Controller allows the Target to replace the current Dynamic Address with a new Dynamic Address.

24.4.5.1. CCC Transfers with Target

The CCC transfers shown in Table 24-17 are supported when operating in Target mode. All unsupported CCC commands are treated as Vendor Specific CCCs by the Target.

Table 24-17. Supported CCC Transfers

CCC	Type
ENTDAA	Broadcast
SETDASA	Directed
GETSTATUS	Directed
GETMXDS	Directed
ENTHDR0	Broadcast
ENTHDR1	Broadcast
ENTHDR2	Broadcast
GETMRL	Directed
SETMRL	Broadcast, Directed

Table 24-17. Supported CCC Transfers (continued)

CCC	Type
GETMWL	Directed
SETMWL	Broadcast, Directed
ENEC	Broadcast, Directed
DISEC	Broadcast, Directed
RSTDAA	Broadcast
SETNEWDA	Directed
GETPID	Directed
GETBCR	Directed
GETCAPS	Directed
ENTAS0	Broadcast, Directed
ENTAS1	Broadcast, Directed
ENTAS2	Broadcast, Directed
ENTAS3	Broadcast, Directed
DEFTGTS	Broadcast
GETACCCR	Directed
GETDCR	Directed
SETAASA	Broadcast
SETGRPA	Directed
RSTGRPA	Broadcast, Directed
RSTACT	Directed

All the CCCs in Table 24-17 are handled within the Target Controller without involving the Target application. The CCC write data from the Active Controller is either captured in a register or consumed within the Target. The CCC read data from the Target is sourced either from the configured parameters or the registers maintained within the Target.

Optionally, the Target (if enabled) can generate a common interrupt (CCCUPDSTA interrupt in the I3CxINTSTA register) when the Active Controller updates any of the following register values through a CCC transfer:

- Dynamic Address Assignment through ENTDAA/SETDASA (including RSTDAA, SETNEWDA, SETAASA: Assign new DA)
- Maximum Read Length
- Maximum Write Length
- Enable/Disable Target events command ENEC, DISEC
- Enter activity state ENTAS0/1/2/3

24.4.5.1.1. Handling GETCAPS CCC

The Target supports GETCAPS CCC with an optional defining byte and returns other capabilities based on the value of the defining byte. The length of data returned depends on the defining byte and its associated capabilities.

GETCAPS Without Defining Byte

The target returns three bytes of data. Table 24-18 details the data format for GETCAPS CCC without the defining byte.

Table 24-18. Data Bytes Format for GETCAPS CCC without Defining Byte

Byte	Bit	Field	Field Value
BYTE 1	0	HDR mode 0	1
	1	HDR mode 1	0
	2	HDR mode 2	0
	3	HDR mode 3	0
	7:4	Reserved	Reserved
BYTE 2	3:0	I^3C 1.x Specification Version	Minor version number of the MIPI I^3C Specification with which this I^3C v1.x device complies. Returns 4'b0001 v1.1.1 compliant device.
	5:4	Group address capabilities	3: Can be assigned three or more group addresses
	6	HDR-DDR Write Abort	Returns 0, as this feature is not supported.
	7	HDR-DDR Abort CRC	Returns 0, as this feature is not supported.
BYTE 3	0	Multi-Lane (ML) Data Transfer Support	Returns 0, as the Multi-Lane feature is not supported.
	1	Device to Device Transfer (D2DXFER) Support	Returns 0, as D2DXFER is not supported in Target mode.
	2	Device to Device Transfer (D2DXFER) IBI Capable	Returns 0, as D2DXFER is not supported in Target mode.
	3	Defining Byte Support in GETCAPS	Returns 1, as defining byte is supported for v1.1.1 compliant devices.
	4	Defining Byte Support in GETSTATUS	Returns 1, as defining byte is supported for v1.1.1 compliant devices.
	5	HDR-BT CRC-32 Support	Returns 0, as this feature is not supported.
	6	IBI MDB Support for Pending Read Notification	Returns 0, as this feature is not supported.
	7	Reserved	

GETCAPS With Defining Byte

The Target returns its particular capabilities based on the value of the defining byte. The length of data returned depends on the defining byte and its associated capabilities. Table 24-19 details the specifics associated with possible defining bytes.

Table 24-19. GETCAPS Defining Byte Values

Value	Encoding	Description	Data Bytes Returned
0x00	TGTCAPS	Describes standard (Target) capabilities and features. Equivalent to GETCAPS Format 1 (same CCC but without a defining byte).	Returns three bytes of data.
0x01 – 0x59	Reserved	Reserved	These defining bytes are unsupported and are NACKed.
0x5A	TESTPAT	Returns a fixed 32-bit test pattern (0xA55AA55A).	Four bytes
0x5B – 0x90	Reserved	Reserved	These defining bytes are unsupported and are NACKed.
0x91	CRCAPS	Describes capabilities and features related to a Controller-capable (that is, a Secondary Controller) device, and its behavior during the Controller handoff.	1-2 bytes. Refer to Table 24-20 below.
0x92	Reserved	Reserved	This defining byte is unsupported and is NACKed.

Table 24-19. GETCAPS Defining Byte Values (continued)

Value	Encoding	Description	Data Bytes Returned
0x93	VTCAPS	Describes the capabilities and features related to a virtual Target-capable device.	1 byte. Refer to Table 24-21 below.
0x94 – 0xD6	Reserved	Reserved	These defining bytes are unsupported and it is NACKed.
0xD7	DBGCAP S	Supported as a Vendor Extension for Debug Capable Device.	1 – N Bytes (DBGCAP1–DBGCAPN)
0xD8 – 0xDF	Reserved	Reserved	These defining bytes are unsupported, and it is NACK'd.
0xE0 – 0xFE	Vendor Extension s	Target treats the GETCAPS CCC as a Directed Vendor CCC.	These defining bytes are supported only if "IC_SLV_VEND_CCC_EN" is set to 1.
FF	Reserved	Reserved	This defining byte is unsupported and is NACK'd.

The byte format for defining byte '91' is documented in Table 24-20.

Table 24-20. CRCAPS Byte Format

Byte	Bit	Field	Field Value
BYTE 1	0	Hot-Join Support	0
	1	Group Management Support	1
	2	Multi-Lane Support	Returns 0, as it is not supported.
	7:3	Reserved	Returns 0, as this field is Reserved.
BYTE 2	0	In-Band Interrupt Support	1
	1	Controller Pass-Back	Returns 1, as this device automatically passes the Controller role back to the former Active Controller from which it received its Controller role.
	2	Deep Sleep Capable	The Reset value of this field is 1'b0, and the value is returned from the DEVICE_CTRL_EXTENDED[DEEP_SLEEP_CAP] register field, which can be programmed by the application.
	3	Delayed Controller Handoff	Returns 1. This feature is supported in Secondary Controller.
	7:4	Reserved	Returns 0, as this field is Reserved.

The byte format for defining byte '93' is documented in Table 24-21.

Table 24-21. VTCAPS Byte Format

Byte	Bit	Field	Field Value
BYTE 1	2:0	Virtual Target Type	Returns 5; supports multiple virtual Target addresses.
	3	Group Management Support	Returns 0 as this field is Reserved.
	4	Side Effects	Returns 1. Configuration CCCs sent to this virtual Target's dynamic address may impact other virtual Targets that are connected to or exposed by this device.
	5	Shared Peripheral Detect	Returns 1. Device uses shared Peripheral logic and supports the Virtual Target Detect operation.
	7:6	Reserved	Returns 0 as this field is Reserved.

24.4.5.1.2. Handling GETMXDS CCC

Whether or not GETMXDS CCC is ACKed by the Target is determined by the value programmed in I3CxTGTCON [MAXDATSPDLMT].

Handling GETMXDS CCC With Defining Byte

Target supports GETMXDS CCC with an optional defining byte and returns other capabilities based on the value of the defining byte. The length of data returned depends on the defining byte and its associated capabilities. Table 24-22 describes the specifics associated with possible defining bytes.

Table 24-22. GETMXDS Defining Byte Values

Value	Encoding	Description
0x00	WRRDTURN	Describes standard (Target) write/read speed parameters and the optional maximum read turnaround time. Equivalent to the GETMXDS CCC without a defining byte.
0x01 – 0x90	Reserved	These defining bytes are unsupported and are NACKed.
0x91	CRHDLY	Describes delay parameters for a Controller-capable (Secondary Controller) device and its expected activity state during the Controller handoff.
0x92 – 0xBF	Reserved	These defining bytes are unsupported and are NACKed.
0xC0 – 0xDF	Reserved	These defining bytes are unsupported and are NACKed.
0xE0 – 0xFE	Vendor extensions	Target treats the GETMXDS CCC as a directed vendor CCC.
0xFF	Reserved	This defining byte is unsupported and is NACK'd.

The byte format for defining byte '91' is described in Table 24-23.

Table 24-23. CRHDLY Byte Format

Bit	Field	Value
7:3	Reserved	Returns 0, as this field is Reserved.
2	Set Bus Activity State	The value is returned from the I3CxCLTDATSPD [SETACTSTATE] register field, which can be programmed by the application.
1:0	Controller Handoff Activity State	The value is returned from the I3CxCLTDATSPD [ACTSTATE] register field, which can be programmed by the application.

24.4.5.1.3. Handling the GETSTATUS CCC

The supported format for the GETSTATUS CCC is shown in Figure 24-34.

Figure 24-34. Handling of the GETSTATUS CCC

The GETSTATUS vendor-reserved fields are used to indicate the error conditions that have occurred in the previous transfer. In case of any error, the next private transfer coming from the Controller is NACKed. The Controller can understand the status of the transfer, which is not successfully accepted or transmitted, by issuing the GETSTATUS CCC.

The I3CxCLTCCCSTAT register reflects all the information passed to the Controller when the Controller issues the GETSTATUS CCC.

A detailed description of the error fields and other status fields present in the GETSTATUS CCC format supported is provided in the description of the I3CxCLTCCCSTAT register.

Handling GETSTATUS CCC With Defining Byte

Target mode supports GETSTATUS CCC with an optional defining byte. The length of data returned depends on the defining byte. Table 24-24 describes the specifics associated with possible defining bytes.

Table 24-24. GETSTATUS Defining Byte Values

Value	Encoding	Description
0x00	TGTSTAT	Returns Target status (standard format). Equivalent to GETSTATUS Format 1 (without a defining byte).
0x01 – 0x90	Reserved	These defining bytes are unsupported and are NACKed.
0x91	PRECR	Returns an alternate status format describing a Controller-capable device (Secondary Controller).
0x92 – 0xBF	Reserved	These defining bytes are unsupported and are NACKed.
0xC0 – 0xDF	Reserved	These defining bytes are unsupported and are NACKed.
0xE0 – 0xFE	Vendor Extensions	Target treats the GETSTATUS CCC as a directed vendor CCC.
0xFF	Reserved	This defining byte is unsupported and is NACKed.

The byte format for defining byte ‘91’ is described in Table 24-25.

Table 24-25. PRECR Byte Format

Byte	Bit	Field	Field Value
BYTE 1	0	Deep Sleep Detected	Returns 0 as Deep Sleep Detection Capability is not supported.
	1	Handoff Delay NACK	The value of the I3CxCNTRLEXT [REQACKCTRL] register field can be programmed by the application.
	7:2	Reserved	0 is returned as this field is Reserved.
BYTE 2	15:8	Vendor Reserved	0 is returned as this field is Reserved.

24.4.5.1.4. Handling ENTDAA, GETPID and GETDCR

The value of the 48-bit Provisional ID can be programmed in the I3CxCLTMIPIID and I3CxCLTPID. The value of DCR can be programmed in the I3CxTGTCON [DCR] field in the SLV_CHAR_CTRL register. These values are automatically fetched when the ENTDAA, GETPID and GETDCR commands are received.

24.4.5.1.5. Vendor Specific CCCs

The Target treats all the CCC transfers as Vendor Specific CCCs. The Target mode is transparent to the application to transfer and receive the Vendor Specific CCCs. The Target mode uses the private receive (Controller write) transfer flow for Vendor Specific Write CCCs and provides a configurable unique command and transfer data ports for Vendor Specific read CCCs. The details of handling Vendor Specific write and read CCCs are described in the subsequent sections.

Vendor Specific Write CCCs

The Vendor Specific Write CCCs use the same Controller write transfer flow. The Target accepts both 'Vendor Specific Broadcast Write CCCs' and 'Address Matched Vendor Specific Directed Write CCCs' and passes them to the application through the Receive Buffer (Rx-FIFO) and indicates the control information of the Vendor Specific write CCC in the response passed through the Response Queue (I3CxRESPQUE).

The Target accepts the write address (ACK response) when all the following conditions are met:

- The Receive Buffer has space (in terms of empty locations) equal to or more than the programmed value of the I3CxBUFTHLD[RXSTART].
- The Response Queue is not full, and it has some space to hold the response for the current vendor-specific write transfer. Otherwise, the Target responds with a NACK for the vendor-specific directed write CCC transfer and ignores the broadcast vendor-specific write transfer.

Once the Vendor Specific directed CCC write address is acknowledged with ACK (accepted), the Target expects the RX FIFO space to be available until the end of the transfer to avoid an overflow condition. In the non-DMA mode of operation, use either the I3CxINTSTA [RXTHLDSTA] interrupt to free up the RX FIFO space while the RX data is being received, or configure the RX FIFO to accommodate the entire write transfer data (when the maximum write transfer size is defined and small). In the DMA mode of operation, the DMA request of the handshake interface is initiated when the RX FIFO level reaches the programmed I3CxBUFTHLD [RXBUF] level.

When an overflow condition is encountered (when the system latency is too high to consume the receiving data), the Target sets the I3CxCLTCCCSTAT [OVFLWERR] bit of the register and drops the further incoming data until the termination (either STOP or RESTART) is detected.

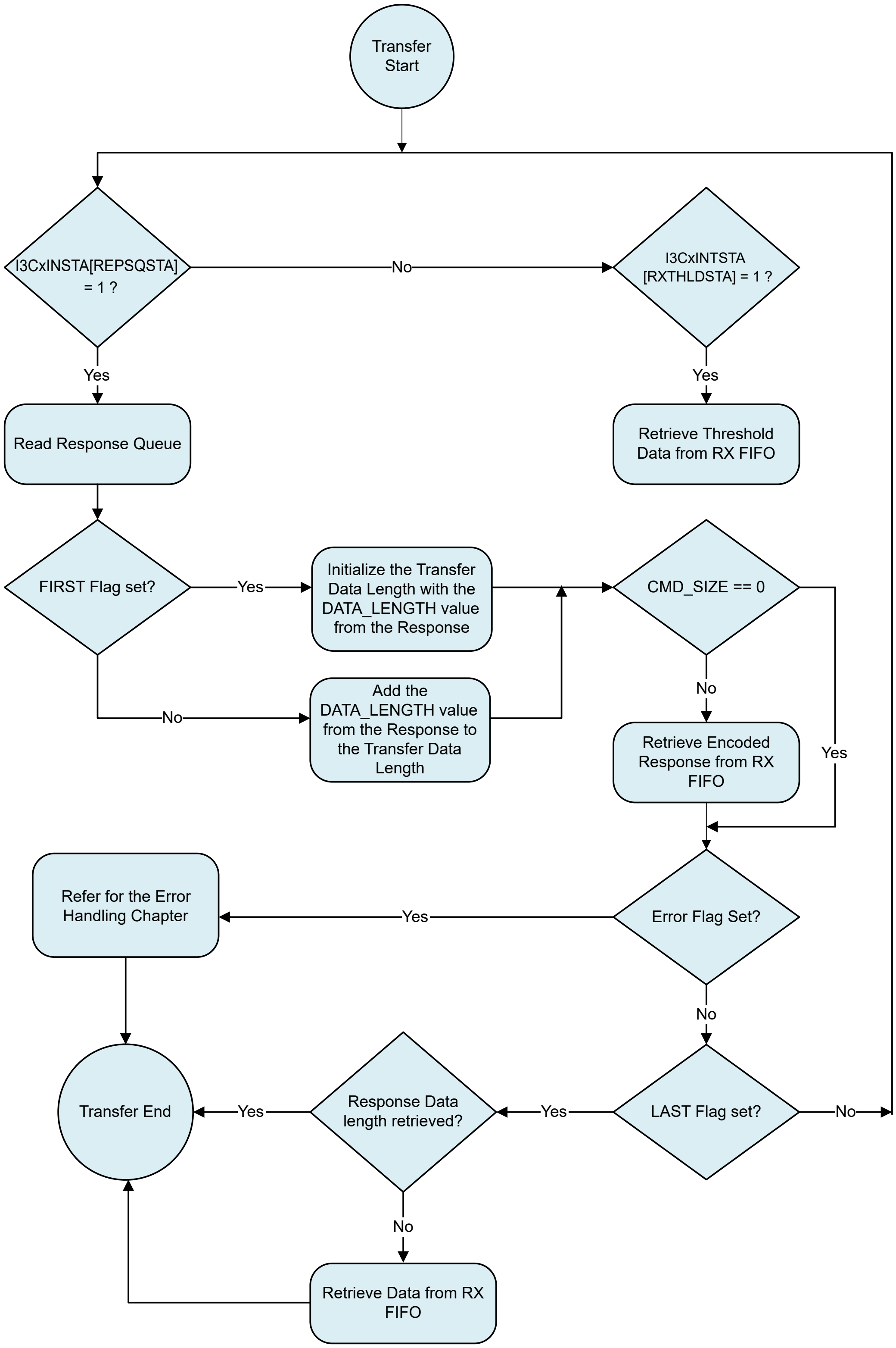
When a parity error is encountered during the vendor-specific CCC write transfer, the Controller sets the I3CxCLTCCCSTAT [PROTOERR] bit of the register and drops the further incoming data until the termination is detected.

Once the Target sets the overflow error or protocol error bit, it rejects (NACK) any further private or Vendor Specific CCC transfer requests from the Controller until the Controller reads the Target status through a GETSTATUS CCC, and the Target application resumes the Target operation by setting the bit I3CxCNTRL [RESUME].

If the receive FIFO does not have a threshold amount of space to accommodate the write transfer, the Controller NACKs the request and the I3CxCLTCCCSTAT [BUFFNTAVAIL] bit is set. Once the buffer not available bit is set, it rejects (NACKs) any further private or Vendor Specific CCC write transfer requests from the Controller until space is available in the receive FIFO. As the broadcast CCCs do not have any mechanism to NACK the incoming data, the incoming broadcast CCC is not sent to the user application.

The Target mentions the Vendor Specific CCC code along with the received data length (in bytes) in the generated response (retrieved from Response Queue I3CxRESPQUE) for the Vendor Specific CCC write transfer. The defining byte (if any) is passed to the application either through the Response Queue (for directed Vendor Specific CCC) or Rx-FIFO (for broadcast Vendor Specific CCC). The defining byte (if any) for broadcast Vendor Specific CCC is passed to Rx-FIFO as the Target cannot differentiate between the defining byte and the data payload.

Figure 24-35. Vendor Specific Receive Transfer



Vendor Specific Directed Read CCC's

The Target mode supports simultaneous Vendor Specific directed read CCC transfers by providing multiple Extended Commands and their related Transmit Buffers. Each extended command is mapped to its own extended Transmit Buffer to hold the CCC data to be returned for the incoming Vendor Specific directed read CCCs.

Program the extended command registers for the Vendor Specific CCC code and its defining byte (if any), which are used by the Target to match the incoming Vendor Specific CCC from the Controller and respond accordingly. The data structure of the Extended Command is in Extended Transmit Command Data Structure.

The data transmission for a vendor-specific directed read CCC is initiated by issuing a TX transfer command. The TX transfer command can be issued by writing to the Extended Command register (I3CxEXTCMDy).

The Target responds to a Vendor Specific directed read address with ACK when all the following conditions are met:

- The vendor-specific CCC and defining byte received from the Controller are matched with the programmed values in one of the I3CxEXTCMDy registers.
- The extended TX FIFO corresponding to the programmed and matched I3CxEXTCMDy is non-empty.
- The response queue is not full, and it has some space to hold the response for the current vendor-specific read transfer.

To determine under which condition the NACK occurred, the Target provides further information as follows:

- An additional interrupt, I3C1INTSTA [READREQSTA], is asserted when there is no valid command in the Command Queue.
- The I3CxCLTCCCSTAT [DATNTRDY] bit is set when the Extended TX FIFO corresponding to the programmed and matched I3CxEXTCMDy is empty.
- The I3CxCLTCCCSTAT [DATNTRDY] bit is also set if the response queue is full.

Once the read address is acknowledged with ACK (accepted), the Target expects the application to provide enough data in the Extended TX FIFO to avoid underrun conditions. It is recommended to program enough data in the corresponding I3CxEXTTXDATy before programming the command in its equivalent I3CxEXTCMDy register. Vendor-specific CCC's command structure does not consist of a data length. Hence, the Extended TX FIFO being empty is considered the transfer end.

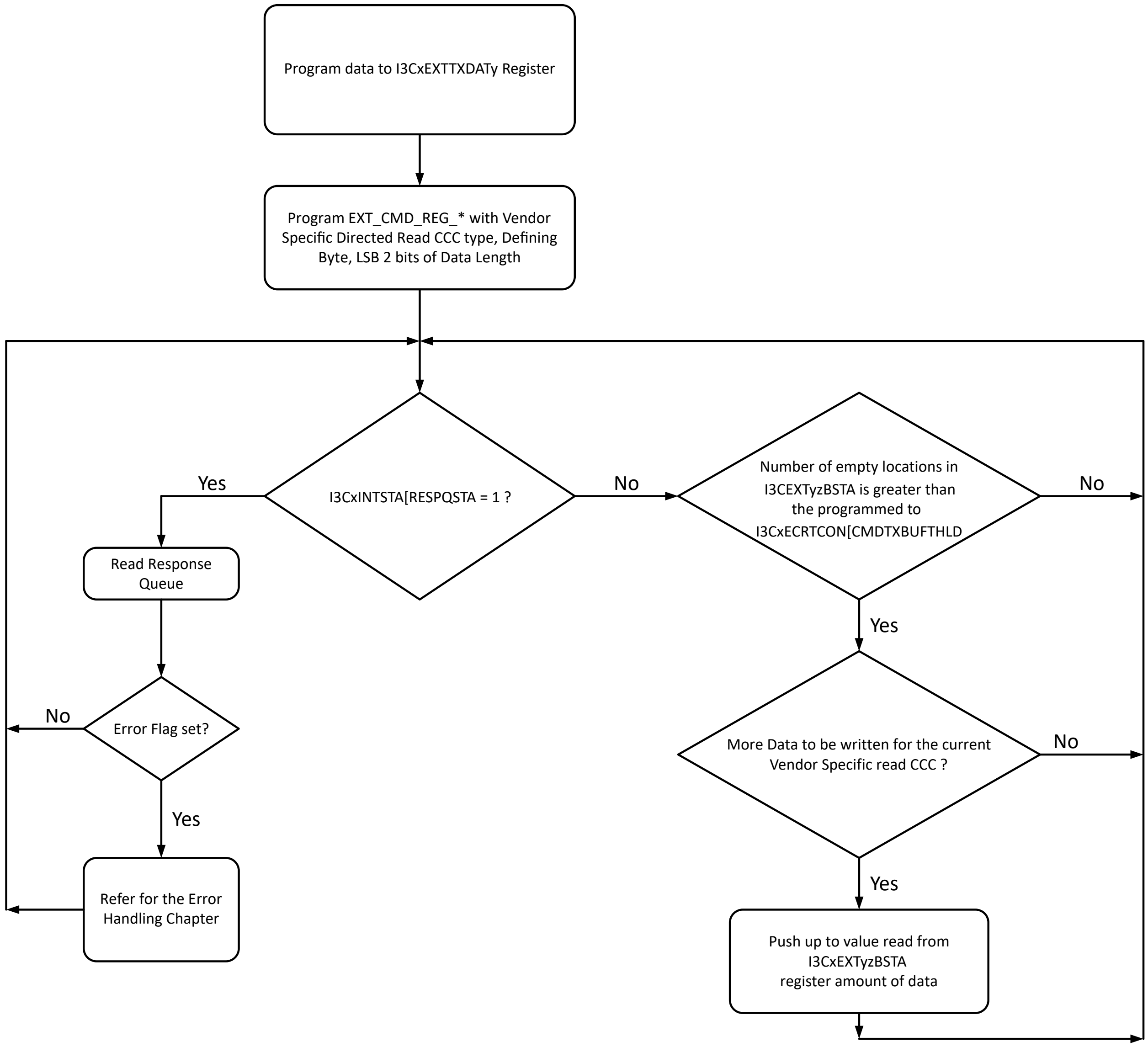
Once the TX Command is programmed in the I3CxEXTCMDy, the command is valid for only one Vendor Specific CCC Read transfer. You are expected to program I3CxEXTCMDy and I3CxEXTTXDATy again for the next Vendor Specific CCC transfer after receiving the response for the previously programmed command.

If the data length of the Vendor Specific CCC read transfer is more than the configured Extended FIFO buffer depth, it is expected that the application will monitor whether half the space is available through I3CxEXTyzBSTA registers and program the subsequent data to be transmitted by the Target.

You can program up to four TX commands at a time to respond to the incoming Vendor Specific read CCC transfers.

As Vendor Specific CCC transfers are pre-agreed transfers between the Controller and Target, the Target application is expected to program the Extended Transmit buffers with the exact amount of data that can be processed by the Controller. If the Controller terminates Vendor Specific read transfers early, the Extended Transmit buffer of the corresponding command should be flushed by the Target application through the I3CxETXQRSTCON register to continue with programming new TX command data.

Figure 24-36. Vendor Specific Transmit Transfer



24.4.5.2. Private Data Transfers

The Target supports private read and write transfers through the I^{3}C interface over the SCL and SDA lines.

For Private Receive and Transmit Transfers in SDR mode of operation, if the Target can accept or transmit the transaction from the Controller, it accepts the address by providing an ACK response; otherwise, the address is NACKed.

In HDR mode, if the Target Controller is not able to accept the Private Write transaction from the Controller, the data is dropped because there is no mechanism to NACK a private write transfer. For a private read transfer, a NACK response is issued.

Receiving a NACK means that an error exists in the current or previous transfer read/write, and the Controller can choose to do one of the following:

- Issue a Start/Restart and re-attempt the transfer. If the previous transfer was NACKed because of the unavailability of internal buffer space, the second transfer might be successful.

- Issue a GETSTATUS CCC to check if the NACK is due to a more permanent condition encountered by the previous transfer because of some protocol error like parity/CRC or due to overflow/underflow errors, and then re-issue the NACK transfer.

24.4.5.2.1. Handling Private Receive (Controller Write) Transfers

In SDR mode of operation, the Target accepts the write address (ACK response) when all of the following conditions are met:

- The Receive Buffer has space (in terms of empty locations) equal to or more than the programmed value of the I3C1BUFTHLD [RXSTART].
- The Response Queue is not full, and it has some space to hold the response for the current transfer.

Otherwise, the Target responds with a NACK for the private write address.
Note: The frequency of the responses for the write transfer is controlled by the threshold I3CxECRTCON [RSPDATTHLD]. It indicates after how many data bytes a new transfer is generated. For the last response of the transfer, the indicated data length is less than or equal to the threshold I3CxECRTCON [RSPDATTHLD]. The data structure of the response is in Response Data Structure.

Once the write address is acknowledged with ACK (accepted), the Target expects the RX FIFO space to be available until the end of the transfer to avoid an overflow condition. In the non-DMA mode of operation, use either the I3C1INTSTA[RXTHLDSTA] interrupt to free up the RX FIFO space while the RX data is being received, or configure the RX FIFO to accommodate the entire write transfer data (when the maximum write transfer size is defined and small). In the DMA mode of operation, the DMA request of the handshake interface is initiated when the RX FIFO level reaches the programmed I3C1BUFTHLD [RXBUF] level.

When an overflow condition is encountered (when the system latency is too high to consume the receiving data), the Target sets the I3CxCLTCCCSTAT [OVFLWERR] bit of the register and drops the further incoming data until the termination (either STOP or RESTART) is detected.

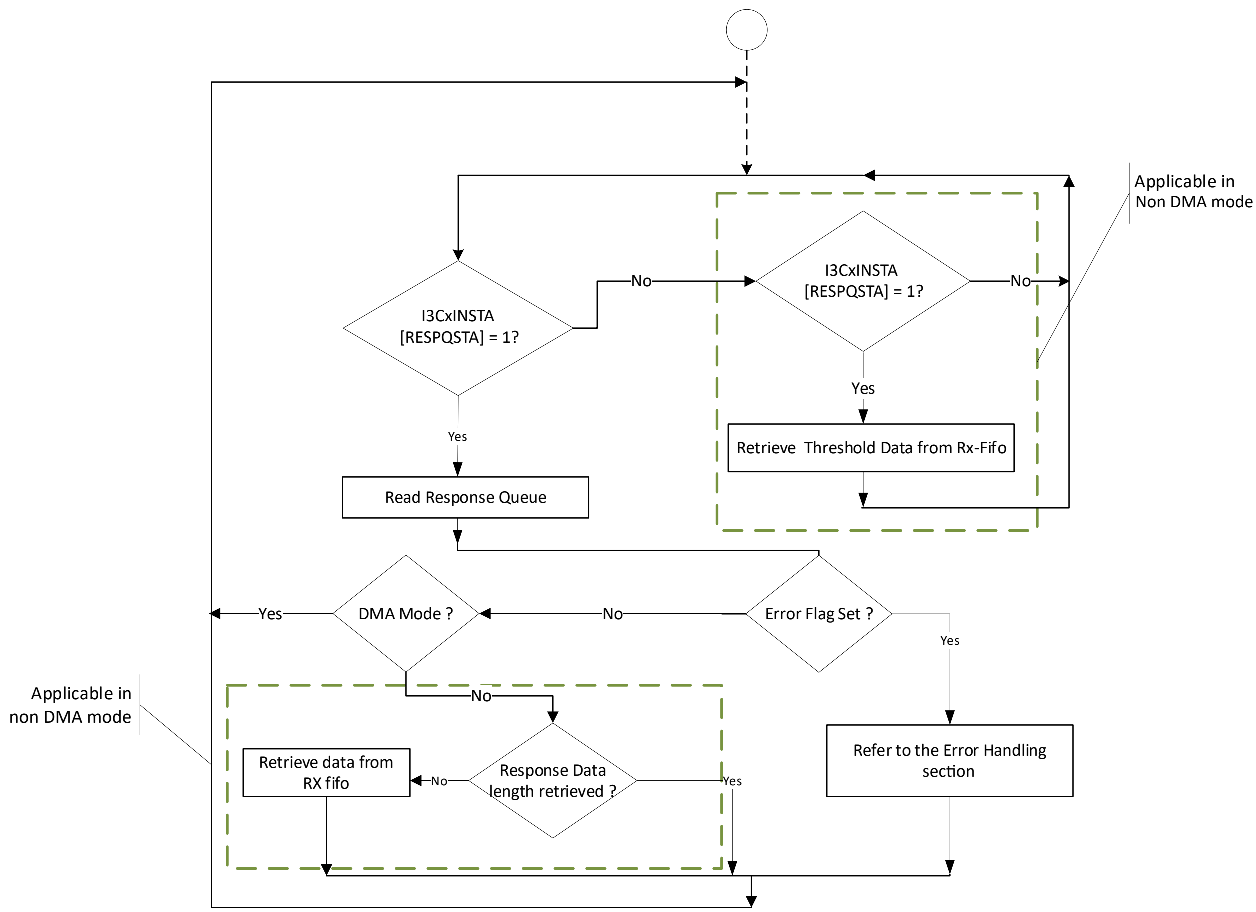
When a parity error is encountered during the Controller write transfer, the Controller sets the I3CxCLTCCCSTAT [PROTOERR] bit of the register and drops the further incoming data until the termination is detected.

Once the Target sets the overflow error or protocol error bit, it rejects (NACK) any further private or Vendor Specific CCC transfer requests from the Controller until the Controller reads the Target status through a GETSTATUS CCC, and the Target application resumes the Target operation by setting the bit I3CxCNTRL [RESUME].

If the receive FIFO does not have a threshold amount of space to accommodate the write transfer, the Controller NACKs the request and the I3CxCLTCCCSTAT [BUFFNTAVAIL] bit is set. Once the buffer not available bit is set, it rejects (NACKs) any further private write transfer requests from the Controller until space is available in the receive FIFO.

Note: In I^2C mode of operation, if an overflow error is encountered in a write transfer, the Target terminates the ongoing transfer by sending a NACK. It rejects (NACKs) any further private transfer requests from the Controller until the Target application resumes the Target operation by setting the bit I3CxCNTRL [RESUME].

Figure 24-37. Private Receive Transfer



24.4.5.2.2. Handling Private Transmit (Controller Read) Transfers

The data transmission for a private Controller read is initiated by issuing a TX transfer command. The TX transfer command can be issued by writing the TX command data structure into the I3CxCMDQUE register (refer to Transmit Command Data Structure). The Target responds to a private read address with ACK when all of the following conditions are met:

- TX Command is valid in the Command Queue.
- TX FIFO has data equal to either the data length size of the command or the I3CxBUFTHLD[TXSTART] size is met.
- Response Queue Non-Full.

Otherwise, the Target responds with a NACK for the private read address.

To determine under which condition the NACK occurred, the Controller provides further information as follows:

- An additional interrupt, I3CxINTSTA [READREQSTA], is asserted when there is no valid command in the Command Queue.
- The I3CxCLTCCCSTAT [DATNTRDY] bit is set when the Extended TX FIFO corresponding to the programmed and matched I3CxEXTCMDy is empty.
- The I3CxCLTCCCSTAT [DATNTRDY] bit is also set if the Response Queue is full.

During HDR transfers (ENTHDR CCC detected), the Target responds with either a "Target NACK" preamble to indicate the Target's inability to transmit the data for the private read command when one of the conditions is not met.

Once the read address is acknowledged with ACK (accepted), the Target expects the application to provide enough data in the TX FIFO to avoid underrun conditions. It is recommended to program enough data in the corresponding I3CxEXTTXDATy before programming the command in its equivalent I3CxEXTCMDy register.

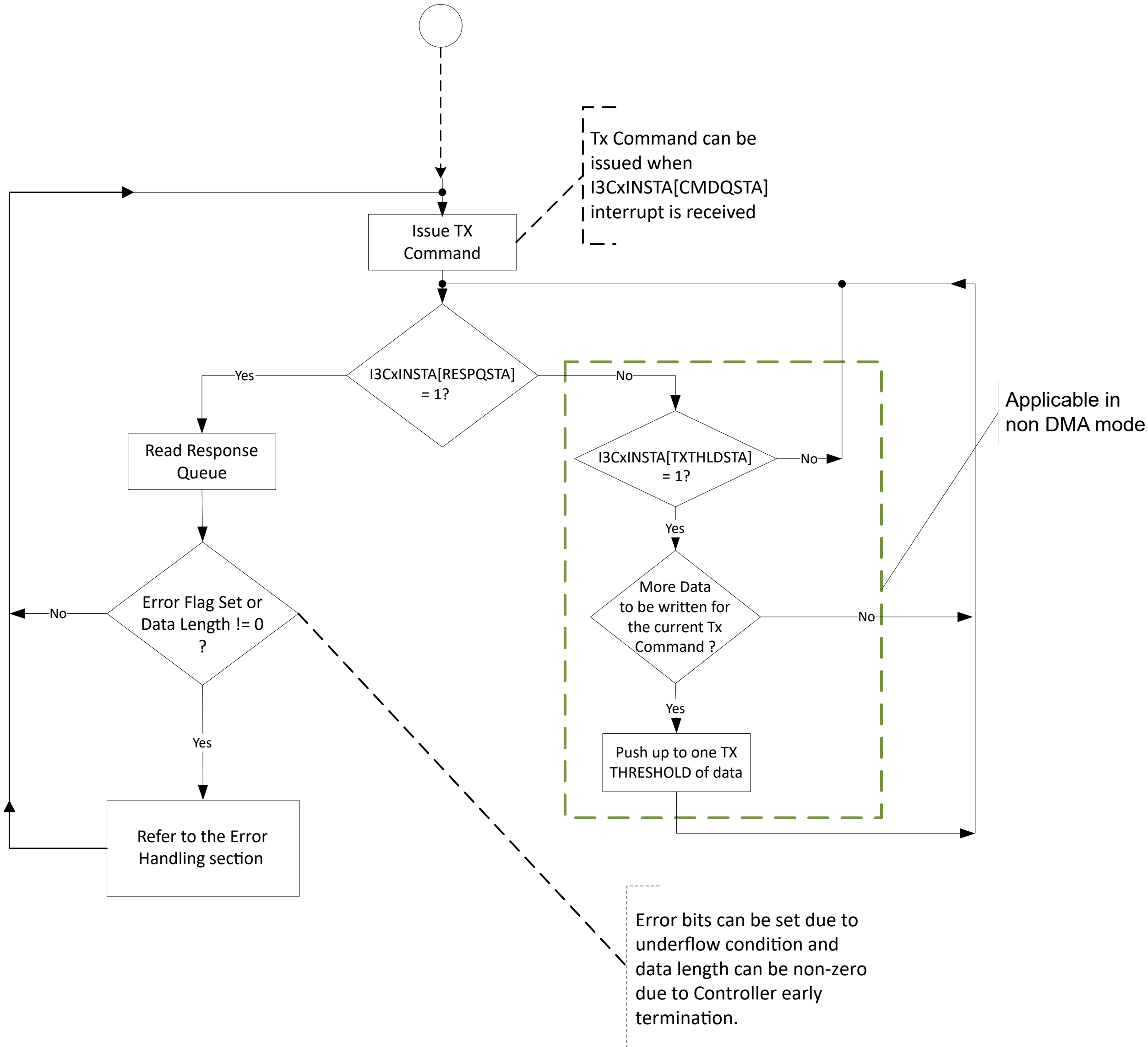
In the non-DMA operation, use the I3CxINTCON[TXTHLDINTEN] interrupt to fill the TX FIFO while the TX data is being transmitted on the I^3C bus. In the DMA mode of operation, the DMA request of the handshake interface is initiated when the TX FIFO empty location level reaches the programmed I3CxBUFTHLD[TXTHLD] level.

When an underrun condition is encountered (when the system latency is too high to provide the transmit data), the Target sets the I3CxCLTCCCSTAT[UDFLWERR] bit and terminates the transfer on the I^3C bus.

Once the Target sets the I3CxCLTCCCSTAT [UDFLWERR] bit, it rejects (NACK) any further private transfer requests from the Controller until the Controller reads the device status through a GETSTATUS CCC, and the Target application resumes the Target operation by setting the I3CxCNTRL [RESUME] bit of the register.

If the transmit FIFO does not have the threshold amount of data to respond to the Controller's read request, the Target NACKs the request, and the I3CxCLTCCCSTAT [DATNTRDY] bit is set. Once the Target sets the I3CxCLTCCCSTAT [DATNTRDY] bit, it rejects (NACK) any further private read transfer requests from the Controller until data is available in the transmit FIFO.

Figure 24-38. Private Transmit Transfer



24.4.5.2.3. Virtual Target Transfers

HDR-DDR Flow Control elements provide a mechanism for the Target to terminate an HDR-DDR Write transfer with or without CRC and also generate and send a CRC word in cases where the Controller has terminated an HDR-DDR read transfer.

A broadcast and directed CCC 'ENDXFER' provides the framework for Controllers and Targets to exchange setup parameters for ending data transfer in HDR-DDR Mode. Flow control parameters supported for HDR-DDR transfer are as follows:

- Enable the ACK/NACK capability for HDR-DDR write command.
- Enable write early termination request.
- Enable CRC word indicator for write/read early termination request.

By default, HDR-DDR Abort CRC and HDR-DDR Write Abort capabilities are enabled. The Controller's application, in turn, is expected to negotiate the flow control capabilities with the Target by issuing the ENDXFER CCC.

The Controller is expected to send a broadcast or directed ENDXFER CCC with the defining byte 0xF7, with the additional data byte programmed as per the flow control requirements. The ENDXFER CCC with the defining byte 0xF7 needs to be followed by another broadcast/directed ENDXFER CCC with the defining byte 0xAA. This is used by the Target to confirm and configure its flow control capabilities as per the previous ENDXFER CCC (defining byte 0xF7), whose data byte structure is shown in Table 24-26.

Table 24-26. ENDXFER CCC Additional Data Byte for Sub-Command 0xF7 (HDR-DDR Protocol)

Bits	Description	Values
[7:6]	CRC word indicator	2'b11: No CRC word follows early termination request
		2'b01: CRC word follows early termination request Other: Reserved
[5]	Enable WRITE early termination request	1'b0: Enable 1'b1: Disable
[4]	Enable ACK/NACK capability for the WRITE command	1'b0: Enable 1'b1: Disable
[3:0]	Reserved for future use	

Following the configuration of flow control capabilities using the ENDXFER CCCs, the Target functions as follows for each of the flow control capabilities.

- Command level ACK/NACK Capability: If this capability is enabled by the Active Controller, then the Target decides to either ACK or NACK the write or read command based on the following
  - Address match (dynamic address, group address, Virtual Target Address)
  - Availability of data and response queues
  - Detection of parity error in the Command Word
- Write Data Early Termination Request: If this capability is enabled by the Active Controller, the Target might terminate a write transfer in the following cases:
  - Rx FIFO / Response FIFO Overflow
  - Parity error in the received data word

CRC word follows Early Termination Request: If this capability is enabled by the Active Controller, then the Target drives or receives the CRC word after HDR-DDR Read or Write Early Termination scenarios.

Virtual Target Transfers

I^3C can be configured to present multiple Virtual Targets (up to four) on the I^3C Bus and behaves as though it has multiple I^3C Targets. The Virtual Target transfers through a single I^3C interface over the SCL and SDA lines. If the Virtual Target is selected as a Static Address Target, the Virtual Target responds to ENTDAA, SETDASA and SETAASA CCC commands from the Active Controller until the Dynamic Address is assigned successfully. If the Virtual Target is selected as a Dynamic Address Target, the Virtual Target responds only to the ENTDAA CCC command from the Active Controller until the Dynamic Address is assigned successfully. Once the Dynamic Address is assigned and valid, the Virtual Target stops responding to the ENTDAA, SETDASA, and SETAASA CCC commands until the Dynamic Address is reset through the RSTDAA CCC command. The SETNEWDA CCC from the Active Controller allows the Virtual Target to replace the current Dynamic Address with a new Dynamic Address.

Virtual Target Write Transfers

The Virtual Target write transfers use the same Controller write transfer flow. The shared peripheral logic accepts the write address (ACK response) when the following conditions are met:

- The Receive Buffer has space (in terms of empty locations) equal to or more than the programmed value of the I3C1BUFTHLD [RXSTART].
- The Response Queue is not full, and it has some space to hold the first response for the current write transfer. Otherwise, the shared peripheral logic responds with a NACK response for the write transfer and ignores the write transfer.

Once the write address is acknowledged with ACK (accepted), the Target expects the RX FIFO space to be available until the end of the transfer to avoid an overflow condition. In the non-DMA mode of operation, use either the I3C1INTSTA[RXTHLDSTA] interrupt to free up the RX FIFO space while the RX data is being received, or configure the RX FIFO to accommodate the entire write transfer data (when the maximum write transfer size is defined and small). In the DMA mode of operation, the DMA request of the handshake interface is initiated when the RX FIFO level reaches the programmed I3C1BUFTHLD[RXBUF] level.

When an overflow condition is encountered (when the system latency is too high to consume the receiving data), the Target sets the I3CxCLTCCCSTAT [OVFLWERR] bit of the register and drops the further incoming data until the termination (either STOP or RESTART) is detected.

When a parity error is encountered during the write transfer, the Shared Peripheral Logic sets the I3CxCLTCCCSTAT [PROTOERR] bit of the register and drops the further incoming data until the termination is detected.

Once the Shared Peripheral Logic sets the overflow error or protocol error bit, it rejects (NACK) any further private or Vendor Specific CCC transfer requests from the Controller until the Controller reads the Target status through a GETSTATUS CCC and the Target application resumes the Target operation by setting the bit I3CxCNTRL [RESUME].

If the receive FIFO does not have a threshold amount of space to accommodate the write transfer, the Controller NACKs the request and the I3CxCLTCCCSTAT [BUFFNTAVAIL] bit is set. Once the buffer not available bit is set, it rejects (NACKs) any further private write transfer requests from the Controller until space is available in the receive FIFO. As the broadcast CCCs do not have any mechanism to NACK the incoming data, the incoming broadcast CCC is not sent to the user application.

The response FIFO full condition is met during a transfer, and if another response has to be generated during the response FIFO full condition, the overflow error is asserted and data is lost after this point. This behavior is similar to the data FIFO overflow during the transfer. Response FIFO overflow recovery is treated similarly to data FIFO overflow recovery.

The frequency of the responses for the write transfer is controlled by the threshold I3CxECRTCON [RSPDATTHLD]. It indicates after how many data bytes a new transfer is generated. For the last response of the transfer, the indicated data length is less than or equal to the threshold I3CxECRTCON [RSPDATTHLD].

The Shared Peripheral Logic includes the following in the generated response (retrieved from the response queue) for the write transfer:

- DATA_LENGTH: Reports the data length of the transfer or part of the transfer that the response represents. If DEFTGTS == 0, or if DEFTGTS == 1, it represents the device count byte for DEFTGTS CCC.
- Target_ID: Reports the Virtual Target/Targets that this transfer is intended for.
- CMD_SIZE: Reports if the first data entry in the RX FIFO for the transfer is an encoded command or not, and how many bytes are valid from that command. This field is valid only if the FIRST field is '1'.
- ERR_STATUS: Reports the error status of the write transfer.
- CCC: Reports if the write transfer is a vendor-specific write CCC or a private write.
- FIRST: Reports if the response is the first response of the write transfer.

- LAST: Reports if the response is the last response of the write transfer.
- DEFTGTS: Reports if the Received CCC is DEFTGTS CCC or not. If DEFTGTS == 1, CMD_SIZE == 0, and no response data is entered in the Data FIFO for this CCC.
- ATTR: Reports the Response Attribute. Only value 0 is supported.

Based on the CMD_SIZE field in the generated response, the first data word (four bytes) corresponding to the transfer can be part of the generated response:

- 0x0: No Command Word. The first word from Rx FIFO is the first data word of the transfer.
- 0x1: One Command Word in the RX FIFO. Only byte 0 is valid. It contains either the HDR CMD code or the CCC Code based on the CCC bit in the response.
- 0x2: One Command Word in the RX FIFO. Only byte 0 and byte 1 are valid. It contains either the HDR CMD code or the CCC Code based on the CCC bit in the response in byte 0. It contains the defining byte for the CCC in byte 1.
- 0x4: One Command Word in the RX FIFO. All 4 bytes are valid. It contains HDR Bulk Transfer Command words 0, 1, 2 and 3.

Figure 24-39 represents how the Extended Response is distributed in the Rx-FIFO.

Figure 24-39. Response in RX FIFO

Response in RX FIFO

	31:24	23:16	15:8	7:0
0	Reserved		Defining Byte/ Reserved	CCC Code/HDR Command Code
1	Data0			
2	Data1			
⋮	⋮			

The Extended Response Structure is defined in Table 24-31.

The defining byte (if any) is passed to the application either through the response part from Data FIFO (for directed vendor-specific CCC) or RX-FIFO Data (for broadcast vendor-specific CCC). The defining byte (if any) for broadcast vendor-specific CCC is passed to RX-FIFO as the Target cannot differentiate between the defining byte and data payload.

Virtual Target Read Transfers

The Target mode supports simultaneous read transfers by providing multiple Extended Commands and their related Transmit Buffers. Each extended command is mapped to its own extended Transmit Buffer to hold the data to be returned for the incoming private read or vendor-specific directed read CCCs.

After programming, the extended command registers are used by the Target to match the incoming read transfer from the Controller and respond accordingly. The data structure of the Extended Command is documented in Table 24-30.

The data transmission for a read transfer is initiated by issuing a TX transfer command. The TX transfer command can be issued by writing to the extended command registers pair (I3CxEXTCMDy_2, I3CxEXTCMDy). The command is considered valid after I3CxEXTCMDy.VALID is written to '1'.

Note: The application should not program the same type of command concurrently for the same Virtual Target in two different command registers.

The Target responds to a vendor-specific directed read address with ACK when the following conditions are met:

- Read transfer type and Vendor Specific received from the Controller is matched with the programmed values in one of the I3CxEXTCMDy2 and I3CxEXTCMDy registers:
  - ADDR_OFFSET: Transfer matches only with the dynamic/static address that is assigned to the respective Virtual Target.
  - CCC: Selected if a private read or vendor-specific directed read is matched.
  - CCC_HDR_HEADER
    - If CCC == 0 and CCC_HDR_HEADER == 0x00, it matches only SDR Private Read.
    - If CCC == 0 and CCC_HDR_HEADER != 0x00, it matches only HDR Private Read that have cmd_code equal to CCC_HDR_HEADER.
    - If CCC == 1, then it matches the CCC code of the incoming Vendor Specific Read CCC.
  - DEFINING_BYTE: If CCC == 1, then it matches the Defining Byte of the incoming Vendor Specific Read CCC. CCCs that have DEFINING_BYTE == 0x00 also match the CCC without a defining byte.
- The extended TX FIFO corresponds to the programmed and matched I3CxEXTCMDy and is non-empty.

To determine under which condition the NACK occurred, the Shared Peripheral Logic provides further information as follows:

- An additional interrupt, I3CxINTSTA [READREQSTA], is asserted when there is no valid command in the Command Queue.
- The I3CxCLTCCCSTAT [DATNTRDY] bit is set when the Extended TX FIFO corresponding to the programmed and matched I3CxEXTCMDy is empty.

Once the read address is acknowledged with ACK (accepted), the Target expects the application to provide enough data in the Extended TX FIFO to avoid underrun conditions. It is recommended to program enough data in the corresponding I3CxEXTTXDATy before programming the command in its equivalent I3CxEXTCMDy register.

The data length of the read transfer can be programmed with infinite data length by setting the I3CxEXTCMDy.FINITE_DL field to a 1'b0 value. Setting this field to 1'b1 represents a fixed data length read transfer with the length defined by the I3CxEXTCMDy.DATA_LENGTH.

When finite data length is configured, if the Controller tries to read less data than configured in the command, the early termination error is asserted. Only the Target ends the transfer.

When infinite data length is used, the Target ends the transfer when the transmit FIFO is empty. If the Controller ends the transfer, then the early termination error is asserted. The data length transmitted by the Target is always a multiple of four bytes.

If infinite length transfer is configured, then the Extended TX FIFO empty is considered the transfer end.

Once the TX Command is programmed in the I3CxEXTCMDy registers, the command is valid for only one read transfer. You are expected to program the I3CxEXTCMDy registers and I3CxEXTTXDATy again for the next read transfer after reading the response for the previously programmed command.

If the data length of the read transfer is more than the configured Extended FIFO buffer depth, the application is expected to monitor whether half the space is available through I3CxEXTyzBSTA registers and program the subsequent data to be transmitted by the Target.

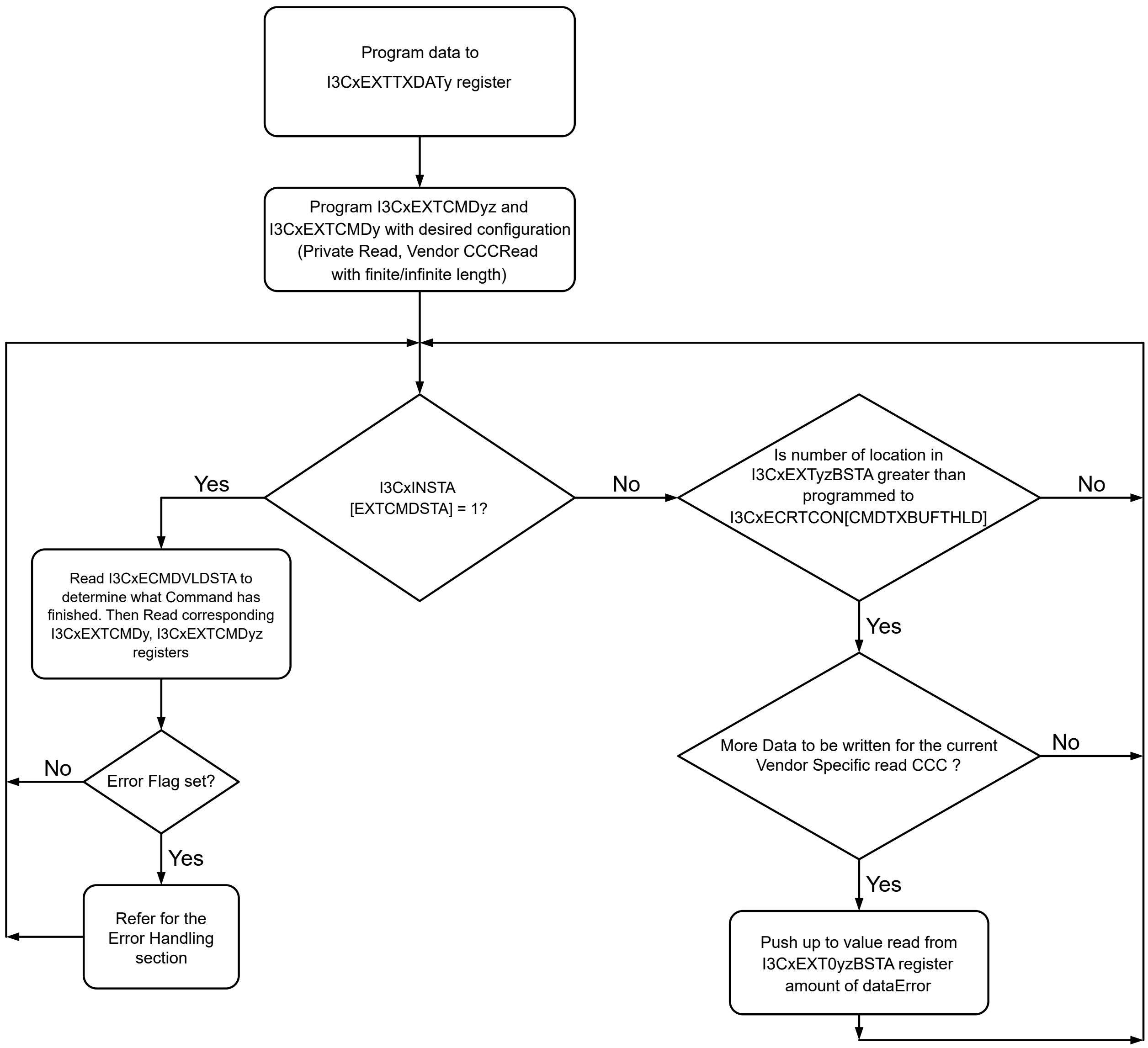
You can program up to four TX commands at a time to respond to the incoming read transfers.

When a read transfer ends, the corresponding I3CxEXTCMDy.[VALID] bit is cleared by the hardware. At the same time, the corresponding I3CxECMDVLDSTA[CMDyVLD] bit is set and the I3C1INTSTA [EXTCMDSTA] is set. I3C1INTSTA [EXTCMDSTA] is cleared after a read from the I3CxECMDVLDSTA register.

The error status for the finished command is read from I3CxEXTCMDy [ERR_STATUS].

As read transfers are pre-agreed transfers between the Controller and Target, the Target application is expected to program the Extended Transmit buffers with the exact amount of data that can be processed by the Controller. If the Controller terminates the read transfer early, the Extended Transmit buffer of the corresponding command should be flushed by the Target application through the I3CxETXQRSTCON register to continue with programming new TX command data.

Figure 24-40. Extended Transmit Flow



Target Interrupt Request Generation with Virtual Target

The application of the Shared Peripheral Logic can decide to send a Target Interrupt Request by asserting the SIR bit in the Target Interrupt Request Register along with the source of payload data and the Virtual Target ID.

When the Virtual Target is enabled, I3CxCLTINT[SIRTGTINDX] is available to select what Dynamic address is used for the generation of the IBI.

24.4.5.3. Hot-Join Request

I^3C's Hot-Join mechanism allows an I^3C device to inform the Active Controller that a newly joined Target is present on the I^3C bus and is ready to receive a dynamic address in order to become fully functional on the bus.

The Hot-Join mechanism requires the joining Target to send a special In-Band-Interrupt (IBI) Request to trigger the Hot-Join procedure. The Target must wait for a Bus Idle condition before sending the Hot-Join Request.

Note: To initiate a Hot-join request after power-on, the I3CxADD[DYNADDRVALID] bit of the I^3C Target should be set to 0.

Once a joining Target has determined that the bus is in the Bus Idle condition, it may initiate a Hot-Join Request by performing the following sequence:

1. The joining Target pulls SDA low, then waits for the Active Controller to act.
2. The Active Controller, detecting the condition, pulls SCL low, thus generating a START.
3. The Active Controller then clocks the SCL line to get the Reserved Target Address (i.e., 7'h02).
4. The joining Target issues the Reserved Target Address 7'b000 0010 with the 8th bit set to 0 (write) to notify the Active Controller that a Target device on the bus needs a dynamic address to become fully operational.
   - Typically, the joining Target is the only device on the bus with no Dynamic Address assigned, so if the Active Controller initiates the Dynamic Address procedure, then the joining Target will be the only device to participate in the Dynamic Address Arbitration (DAA) and receive a Dynamic Address.
   - However, it is also possible for multiple devices to send their Hot-Join Requests simultaneously. In this case, the DAA procedure will iterate and assign Dynamic Addresses to all Targets requesting an address.
5. Once the joining Target has been assigned its new Dynamic Address, it is fully operational on the I^3C bus.

If a NACK response is received or when an arbitration loss is encountered after the first attempt, the Target attempts to resend the Hot-join request at the next START condition seen on the bus, or upon meeting the bus idle condition (whichever happens first).

The Target Controller does not generate the Hot-Join request until a valid I^3C CCC/private transfer with the Broadcast address (7'h7E) is seen on the I^3C bus, if the I3CxCNTRL[ADPTV] is set. It is recommended to set I3CxCNTRL[ADPTV] to 1 if the Target application does not have prior knowledge of the bus (I^2C/I^3C) to which the Target is connected.

The application can program I3CxCLTESTA [HJINTEN] to 0 to disable the HJ capability of the Target. When this bit is set to 0, the Target does not generate an HJ request and behaves as if it's a non-HJ device. Note that programming of I3CxCLTESTA [HJINTEN] is a one-time task, which is done only during the initial configuration and when the Target is disabled.

24.4.5.4. Target Interrupt Request Generation

The Target can generate the Target Interrupt Request (SIR), which is an In-Band Interrupt (IBI), to get the attention of the Active Controller. The application of the Target can decide to send a Target Interrupt Request by asserting the SIR bit in the Target Interrupt Request Register along with the source of payload data.

The Target can send up to four bytes of payload data along with the mandatory byte (MDB).

The source of the four bytes of payload data is from the I3CxCLTSIRDAT register. Set the I3CxCLTINT [SIRCTRL] bits to 2'b00 to indicate the source of the payload data from the I3CxCLTSIRDAT register. The I3CxCLTINT [MDB] is meant for the Mandatory Data Byte that is required to be sent after the Target address is transmitted on the line. The application should program the appropriate MDB values depending on the event for which it wants to trigger the SIR.

The status of the IBI generation is updated in the I3CxCLTIBIRESP[IBISTAT], which is informed to the application by the I3CxINTSTA[IBIUPDSTA] interrupt. Upon successful completion of the I3CxCLTIBIRESP[IBISTAT] update, the I3CxCLTINT[SIR] bit is automatically cleared.

The Target does not attempt to issue the IBI and generates the 'Not Attempted (2'b11)' status under the following conditions:

- Active Controller has not assigned the Dynamic Address.
- Active Controller has cleared the assigned Dynamic Address through RSTDAA.
- Active Controller has disabled the SIRINTEN through DISEC CCC (SIRINTEN in I3CxCLTESTA register).
- The Target has switched the role to Controller (applicable only for Secondary Controller configuration).

The I3CxCLTIBIRESP [SIRRESPDATLEN] indicates the number of bytes of data successfully transmitted on the I^3C line. A non-zero value indicates the residual data in the case of I^3C Controller early termination of payload data. The Target goes to a halt state once the I^3C Controller early terminates the SIR payload data and expects the application to flush the Transmit FIFO, and then resumes the Controller by writing into the RESUME bit of the I3CxCNTRL register.

Notes: The Target does not support the following:

- Retransmission of SIR payload data once the Controller terminates early.
- Provision to enable the Controller to read the residue data after the Controller terminates early.

Figure 24-41. Flow Diagram for Target Interrupt Request (SIR) Generation

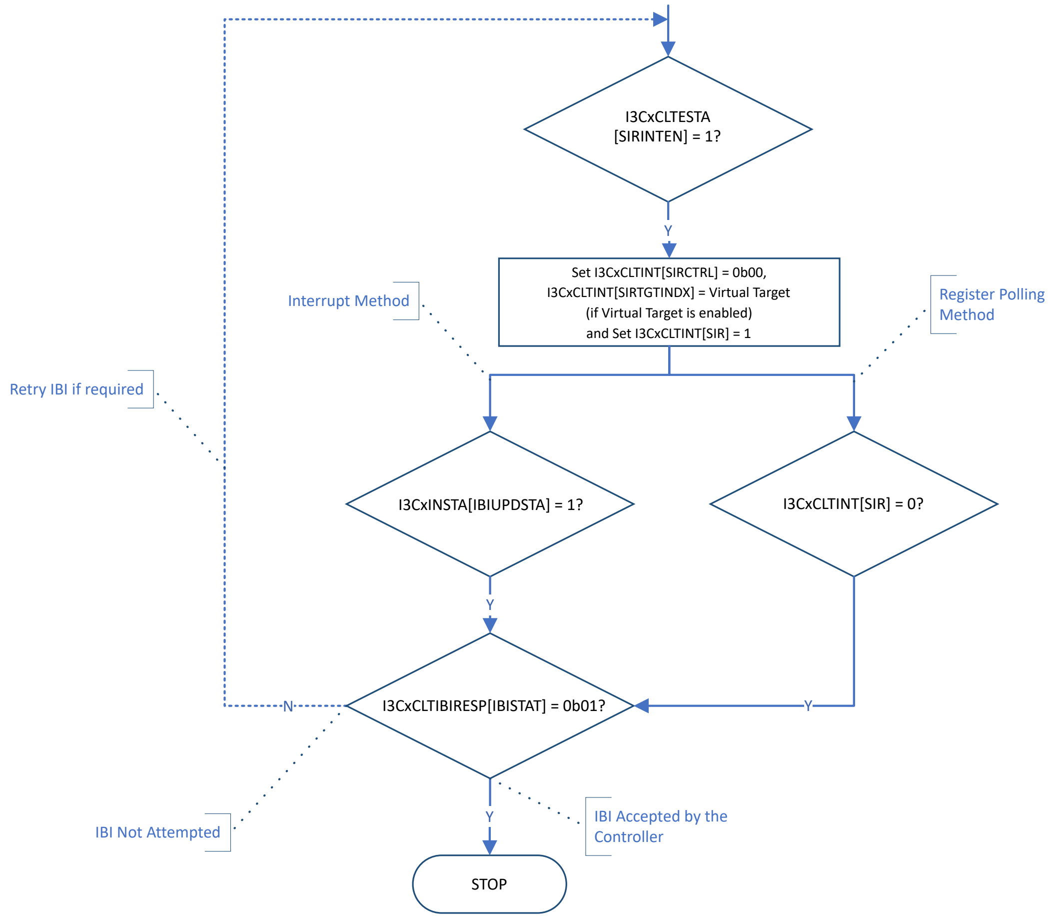
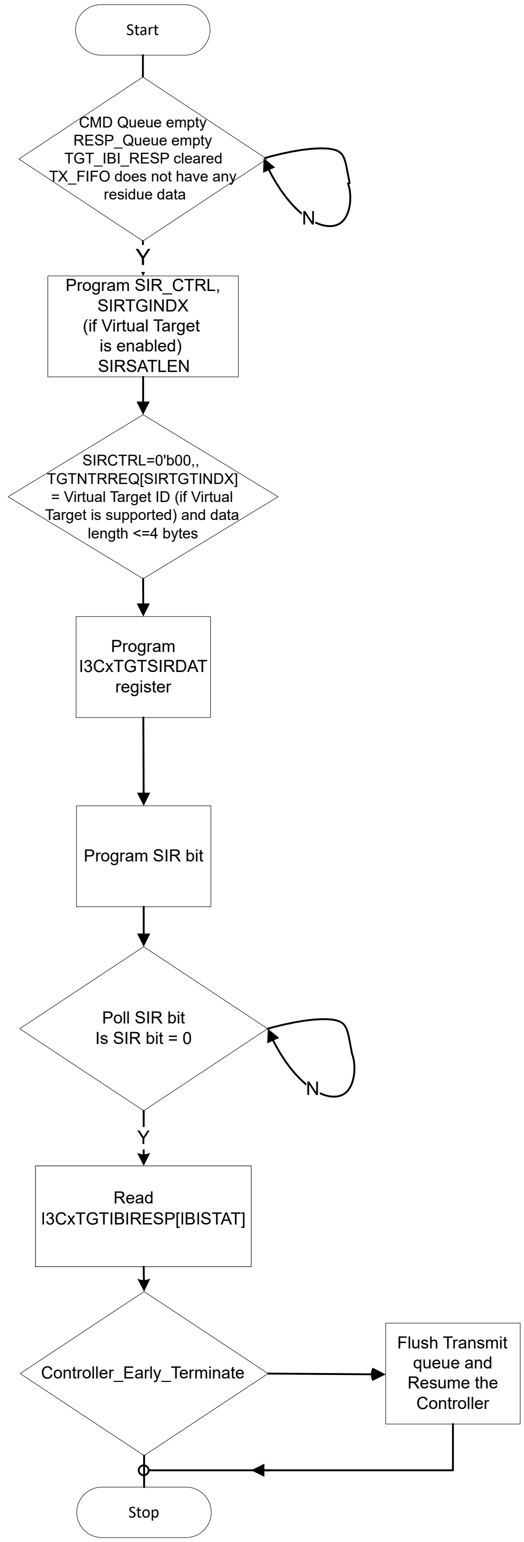


Figure 24-42. Target Interrupt Generation with Data



24.4.5.5. Controller Request Generation

The Target can generate the Controller Request (MR) in non-Active Controller mode to request I^3C bus ownership from the Active Controller.

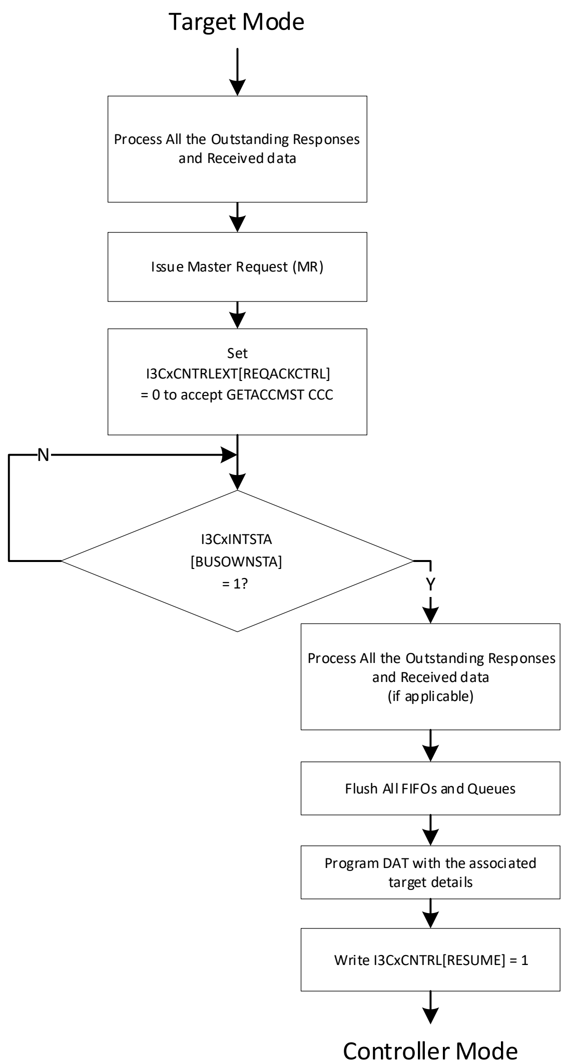
The application can decide to send a Controller Request by asserting the MR bit in the I3CxCLTINT.

The status of the IBI generation is updated in the I3CxCLTIBIRESP[IBISTAT], which is communicated to the application by the I3CxINTSTA[IBIUPDSTA] interrupt. Upon successful completion of the I3CxCLTIBIRESP[IBISTAT] update, the I3CxCLTINT[MR] bit is automatically cleared.

The Target does not attempt to issue the IBI and generates the 'Not Attempted (2'b11)' status under the following conditions.

- The Controller has not assigned the Dynamic Address.
- Controller has cleared the assigned Dynamic Address through RSTDAA.
- Controller has disabled the MRINTEN through DISEC CCC (MRINTEN in I3CxCLTESTA register).
- The Target has already switched the role to Controller.

Figure 24-43. Target to Controller



24.4.5.6. Target Mode Data Structure

This section describes the Data Structures of Command and Response used in Target mode.

24.4.5.6.1. Transmit Command Data Structure

The Transmit Command Structure in Target is used to respond with data for a private read command from the Active Controller.

Table 24-27. Transmit Command Data Structure

Bits	Name	Memory Access	Description
31:16	DATA_LENGTH	W	Data Length This field is used to indicate the data length of the controller read transfer.
15:6	RSVD	W	Reserved
5:3	TID	W	Transmit Transaction ID. This field is used as the identification tag for the command. NOTE: TID value '7' is reserved for Vendor Specific CCCs and recommended to avoid setting TID value to 7 for Private transfers.
2:0	CMD_ATTR	W	Command Attribute defines the Command attribute and its field format. • 1-7 - Reserved • 0 - Transmit Command without IBI

24.4.5.6.2. Response Data Structure

The Response Structure in Target is used to indicate the completion of the transmit command (Private Controller Read operation), the completion of the Controller Write operation (Private Controller Write operation) or the completion of Vendor Specific Write and Read CCC transfers.

Table 24-28. Response Data Structure

Bits	Name	Memory Access	Description
31:28	ERR_STATUS	R	0: No Error 1: CRC error (controller writes in DDR mode) 2: Parity error (controller writes in both DDR and SDR modes) 3: Frame error (HDR mode controller writes) 6: Underflow/Overflow error 8: SDA is released from the stuck state after the prescribed amount of time defined by register I3CxRELSDATIM. 10: Controller early termination
27	RX_RSP	R	Transaction Type
			This field is used to identify the type of transaction: 0: Transmit Response 1: Receive Response
26:24	TID	R	Transmit Transaction ID This field is used as the identification tag for the transmit command. NOTE: TID value '7' indicates the response is meant for vendor-specific CCCs or DEFSLVS CCC transfers.
23:16	CCC_HDR_HEADER	R	Command Code of CCC or HDR Command Code 8'h00 - SDR Private Transfer CCC Code - Vendor Specific CCCs Others - HDR Command Code or, in the case of the Secondary Controller, whenever DEFSLV CCC is received, the DEFSLV CCC command code is reflected in this field. The application comes to know that this field reflects the CCC command code if the TID field is set to 3'b111.
15:8	Defining Byte/ MSB of Data Length	R	Defining byte or MSB of data length If the TID field is set to 3'b111, it indicates the defining byte value received for the vendor specific Directed Write/Read CCC transfers. If the TID field is not set to 3'b111, it indicates the MSB of the data length in bytes of the Controller Write Transfer or the remaining length of the Controller Read Transfer.

Table 24-28. Response Data Structure (continued)

Bits	Name	Memory Access	Description
7:0	LSB of Data Length	R	If the TID field is not set to 3'b111, it indicates the data length in bytes of the Controller Write Transfer, or the remaining length of the Controller Read Transfer (Controller termination or Target underrun). If the TID field is set to 3'b111, it indicates the data length received or transmitted in bytes for vendor-specific CCC transfers. In the case of DEFSLVS, which is indicated by the TID and RX_RESP fields all set to one, this field indicates the device count to the application.

Note: Target Data Length:The Data Length field in Response of a read/write transfer that has experienced any error (has the appropriate Error Status flag set in Response) does not accurately reflect the number of bytes sent or received before the occurrence of the error. Since the Error Handling flow recommends that the appropriate queues be Flushed, the Data length Reported has no particular significance.

24.4.5.6.3. Extended Transmit Command Data Structure

The Extended Transmit Command Structure in Target is used to respond with data for a Vendor-Specific Directed Read command from the Active Controller.

Table 24-29. Extended Transmit Command Data Structure

Bits	Name	Memory Access	Description
31:24	CCC	R/W	Common Command Code Contains the Common Command Code of a specific transfer. This field is used by the Target to match the incoming transfer to return the data from a corresponding transmit buffer I3CxEXTTXDATy.
23:16	DEFINING_BYTE	R/W	Defining byte Contains the defining byte of the respective common command code. This field is used by the Target to match the incoming transfer to return the data from a corresponding transmit buffer I3CxEXTTXDATy. NOTE: If the defining byte is not present for the Vendor Specific Read CCC transfer, then this field is set to 8'h0.
15:8	RSVD		Reserved Field: Yes
7:6	DL_LSB	R/W	LSB 2 bits of data length Contains the last two bits of data length. This field is used by the Target to identify if the complete data length of the transfer programmed in the transmit buffer is not aligned to 4 bytes. These 2 bits act as a byte enable for the last data DWORD programmed in the corresponding transmit buffer.
5:3	TID	R/W	Transaction ID For a Vendor Specific CCC, TID should always be 3'b111.
2:0	RSVD		Reserved Field: Yes

24.4.5.6.4. Extended Transmit Command Data Structure Format 2

The Extended Transmit Command Structure Format 2 in Target is used to respond with data for a Private Read or Vendor Specific Directed Read command from the Active Controller when the Virtual Target is enabled.

Table 24-30. Extended Transmit Command Data Structure Format 2

Word Number	Bits	Name	Memory Access	Description
1	31:28	ADDR_OFFSET	R/W	Address Mask: 0x0: Assign to Virtual Target 0 0x1: Assign to Virtual Target 1 0x2: Assign to Virtual Target 2 0x3: Assign to Virtual Target 3 0x4: Assign to Virtual Target 4
1	27	Reserved		
1	26:25	ADDR_MSK	R/W	Address Mask: 2'b00: No Address Mask
1	24	CCC	R/W	CCC Enable: 1: Command is CCC Direct Read 0: Command is Private Read
1	23:17	Reserved		
1	16	CMD_VLD	R/W	Command Valid bit: 1: Command is Valid 0: Command is not Valid
1	15:8	ERR_STATUS	R	Defines the Error Type of the processed command.
1	7	Reserved		
1	6	FINITE_DL	R/W	Type of Transfer: 1: Finite Length Transfer 0: Infinite Length Transfer
1	5:3	Reserved		
1	2:0	CMD_ATTR	R/W	Defines the type of command used: 001b: Extended Command Others: Reserved
2	31:16	DATA_LENGTH	R/W	Define the data length for the used command from a corresponding transmit buffer (I3CxEXTTXDATy).
2	15:08	DEFINING_BYTE	R/W	Defining byte. Contains the defining byte of the respective common command code. This field is used by the Target to match the incoming transfer to return the data from a corresponding transmit buffer (I3CxEXTTXDATy). **Note:** If the defining byte is not present for the Vendor Specific Read CCC transfer, then this field is set to 8'h0.
2	7:0	CCC_HDR_HEAD ER	R/W	Command Code of CCC or HDR Command Code 8'h00 - SDR Private Transfer CCC Code - Vendor Specific CCC's

24.4.5.6.5. Extended Response Data Structure Format 2

The Response Structure in Target is used to indicate the completion of the Controller Write operation (Private Controller Write operation) or the completion of Vendor Specific Write CCC transfers when the Virtual Target is enabled.

Table 24-31. Extended Response Data Structure Format 2

Bits	Name	Memory Access	Description
31:24	DATA_LENGTH	R	Represents the data length of the transfer
23:21	Reserved		
20:16	TARGET_ID		Represents the Target/Targets that this transfer is intended for: - TARGET_ID[0]: Assigned to Virtual Target 0 - TARGET_ID[1]: Assigned to Virtual Target 1 - TARGET_ID[2]: Assigned to Virtual Target 2 - TARGET_ID[3]: Assigned to Virtual Target 3 - TARGET_ID[4]: Assigned to Virtual Target 4

Table 24-31. Extended Response Data Structure Format 2 (continued)

Bits	Name	Memory Access	Description
15:12	CMD_SIZE	R	Reports if the first data entry in the RX FIFO for the transfer is an encoded command: - 0x0: No Command Word. The first word from the RX FIFO is the first data word of the transfer. - 0x1: 1 Command Word in the RX FIFO. Only byte 0 is valid. It contains either the HDR CMD code or the CCC code based on the CCC bit in the response. - 0x2: 1 Command Word in the RX FIFO. Only byte 0 and byte 1 are valid. It contains either the HDR CMD code or the CCC code based on the CCC bit in the response in byte 0. It contains the defining byte for the CCC in byte 1. - 0x4: One Command Word in the RX FIFO. All 4 bytes are valid. It contains HDR Bulk Transfer Command words 0, 1, 2 and 3.
11:8	ERR_STATUS	R	Error Status: - 0: No Error - 1: CRC error (controller writes in DDR mode) - 2: Parity error (controller writes in both DDR and SDR modes) - 3: Frame error (HDR mode controller writes) - 6: Underflow/Overflow error
7	CCC	R	Response is for Private Write - Response is for CCC Write
6	Reserved		
5	DEFTGTS	R	This field indicates that this response is for DEFTGTS CCC.
4	FIRST	R	This is the first response of the transfer.
3	LAST	R	This is the last response of the transfer.
2:0	ATTR	R	Response Attribute

24.4.5.7. Target Reset Detection

The Target Reset mechanism allows the Controller to reset one or more selected Targets and avoids resetting all Targets. Target Reset supports different levels of Reset, ranging from resetting only the I^3C Peripheral within a Target to resetting the whole Target device, communicated through RSTACT CCC.

The RSTACT CCC which indicates different levels of Reset is expected to be treated as a Vendor Specific CCC. Target mode uses Vendor Specific CCC channel to notify the Target application regarding RSTACT CCC reception and its fields.

Target mode supports I3CxINTSTA[TRSTPATSTA] interrupt to notify the application that Target detected the Target Reset Pattern.

24.4.5.8. Packet Error Check

The Packet Error Check feature enables the generation and validation of an 8-bit CRC on the data stream between the Controller and the Target for reliable communication. The transmitter always inserts an extra PEC byte at the end of the data stream, and the receiver validates it. The PEC is applicable for SDR CCC, IBI and SDR private transfers.

The PEC feature is enabled by programming the I3CxCNTRL[PEC] register bit after receiving a DEVCTRL CCC (treated as a vendor-specific Write transfer).

The PEC is a CRC-8 value calculated on all the message bytes except for START, REPEATED START, STOP conditions or T-bits, ACK and NACK, and Address header (7'h7E followed by W=0) bits. The polynomial for CRC-8 calculation is 'X8 + X2 + X1 + 1'. The PEC byte is supported only for SDR Transfers and is not applicable for HDR Transfers. Figure 24-44 to Figure 24-48 show the inclusion of the PEC byte for different types of transfers.

Figure 24-44. Broadcast CCC with PEC Byte

START S (Sr)	Address Header (0x7E,w)	Broadcast CCC Code (CCC,T)	Data Payload (DP,T)	PEC Byte (PEC,T)	STOP (P/Sr)

Figure 24-45. Directed CCC Write with PEC Byte

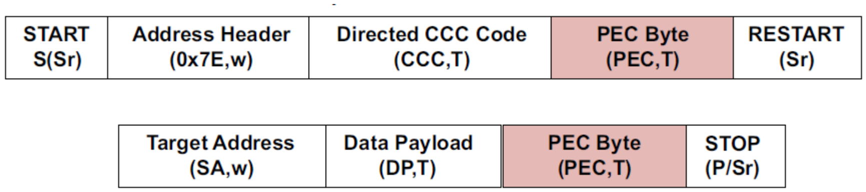


Figure 24-46. Directed CCC Read with PEC Byte

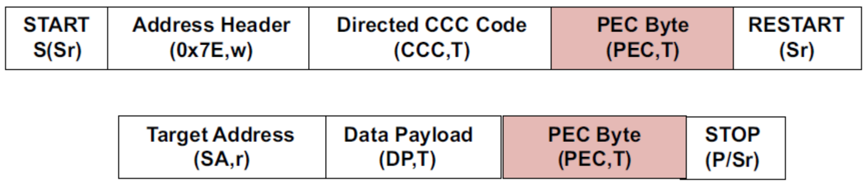


Figure 24-47. Private Write Transfer with PEC Byte

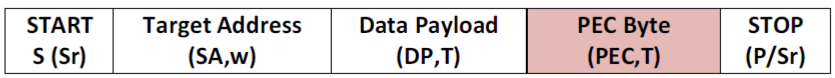
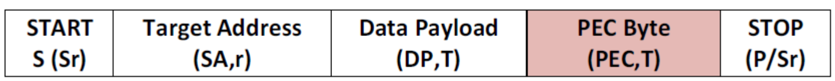


Figure 24-48. Private Read Transfer with PEC Byte



When PEC is enabled:

Controller READ and IBI Transfers: The Target generates a PEC byte and appends it at the end of the transfers.

Controller Write Transfers: The Target validates the PEC byte received at the end of Controller Write Transfers. If the received PEC byte for the write transfer does not match the internally generated PEC byte based on the data received from the Controller, then the Target generates an I3CxINTSTA[TRANSERR] interrupt and indicates it as a 'PEC Error' in Response Status (I3CxRESPQUE) in the case of a vendor-specific CCC transfer. For the internally decoded CCCs and Private Write transfers, the controller is expected to issue a GETSTATUS CCC to recover from the PEC Error.

The PEC feature mandates both the Controller and the Target to pre-agree on the transfer length. During a PEC-enabled transfer, there always exists an extra one byte of data (PEC byte) between START/RESTART and the following RESTART/STOP. The Target enters the PEC state to append the PEC byte after the completion of the prescribed data length mentioned in the DATA_LENGTH field of the Transmit Command.

24.4.5.9. Error Detection and Handling

This section describes the error detection and handling in the Target mode of operation.

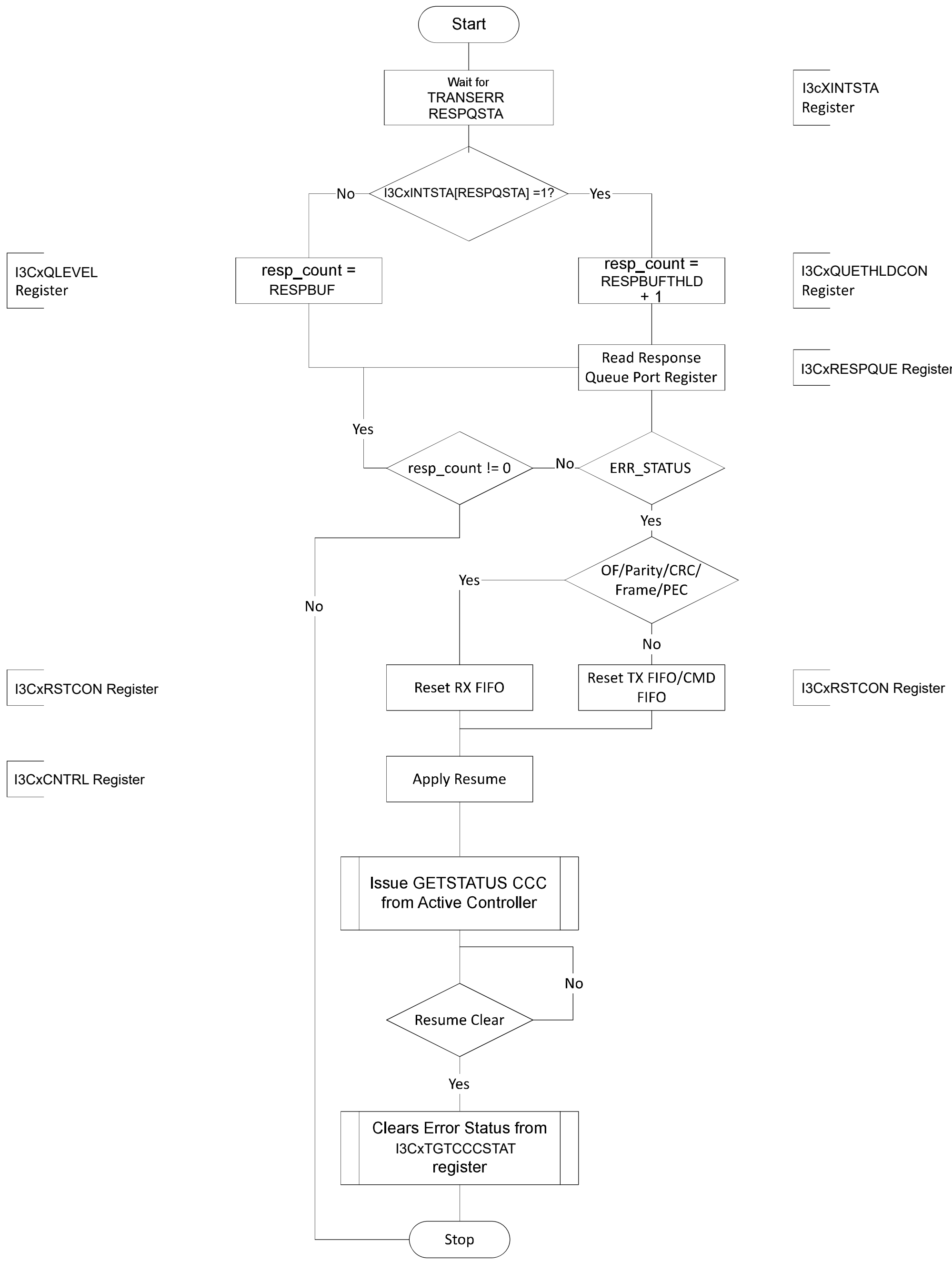
TE0-TE5 Error Handling

The Target supports the detection and recovery from TE0-TE5 errors. Detection and recovery are transparent to the Target application and are internally handled by the Target State Machine. Handling of TE0-TE5 is internal to the Target and transparent to the application; that is, it does not require the intervention of the Target application and is not reported in the Response.

- TE0 and TE1: All transfers are ignored until the HDR exit pattern is detected.

- TE2: For CCC parity error, the CCC value is not updated, and the old values are retained. For private read and write transfers, the appropriate error field in response is updated.
- TE3-TE4: Updates due to CCCs are halted, and the Target retains the current values (without updating) until a STOP/START is observed.
- TE5: Illegally formatted CCCs are handled internally. The CCC value is not updated, and the old values are retained. No information is provided for the application.

Figure 24-49. Error Recovery



24.4.5.9.1. SDA 100 µs Read Abort Mechanism

Target supports an SDA read detector that releases the SDA to High-Z upon detecting that the SCL is unchanged (toggled) during the Read transfer for the user-programmed I3CxRELSDATIM [RELSDATIM] *peripheral clock.

Upon detecting the read abort condition, the Target application is indicated through the interrupt I3CxINTSTA[SDARELSTA] and enters Halt State to flush the FIFOs and expects the application to resume the Target by programming I3CxCNTRL[RESUME] to ‘1’.

24.4.5.9.2. Dead Bus Recovery

For detection and recovery from dead bus mode in Target mode, the application follows these steps to detect and recover from a dead bus error.

Detecting Dead Bus

When the Target application randomly requests an IBI (In-Band Interrupt) or CRR (Controller Role Request) and finds no response within 50 ms (expected to be handled by the application), it is treated as a Dead Bus scenario. The sequence of steps to initiate the IBI/CRR is as follows:

Set I3CxCLTINT[SIRCTRL] = 01/00b (to raise IBI/CRR) and I3CxCLTINT[SIR] = 1b. Program an external timer with 50 ms. The Target application waits for a response based on the polling or interrupt method.

Polling Method: Poll I3CxCLTINT[SIR]. If it is 0, then there is no error scenario, and reset the timer. Otherwise, if the timer expires and I3CxCLTINT[SIR] is still not 0, then the I^3C bus is in Dead Mode, and it is recommended to follow the recovery sequence.

Interrupt Method: The Target application must wait for the IBI completion status interrupt I3CxINTSTA[IBIUPDSTA] and then read the I3CxCLTIBIRESP[IBISTAT] field to check if the Controller accepts IBI. In the meantime, if the timer expires and the application layer does not receive any interrupt, then the I^3C bus is in Dead mode and should follow the recovery sequence.

Recovery Sequence

The Target mode taking control over the I3C Dead Bus is similar to the CE3 recovery process documented in Error Type CE3. The Recovery Sequence includes the following steps:

- Set I3CxCLTINT [CE3RECOV] = 1 to generate the CE3 Error Recovery flow. Wait for SLV_INTR_REQ[SIR] to be cleared.
- If the Secondary Controller successfully takes over control of the bus, then I3CxCLTINT [SIR] is cleared; otherwise, the previous Active Controller on the bus takes over control of the bus.

24.4.5.10. Group Address Support in Target

The Group Address feature enables multiple I^3C Target Devices to share a single Group Address or multiple Group Addresses, allowing the Controller Device to send a given I^3C Message to all Target Devices in the Group at once rather than one at a time.

The Target supports multiple Group Addresses, one address for each Group in which the Target is included, up to a maximum of four. All the Group Addresses are in effect simultaneously.

Target indicates the supported number of Group addresses through GETCAPS

CCC to the Controller. The Target accepts group addresses assigned through SETGRPA CCC until the assigned Group Addresses count reaches four. If the assigned group addresses count exceeds four, then the Target NACKs the SETGRPA CCC.

The assigned Group Addresses can be reset by the Controller using RSTGRPA CCC or RSTDAA CCC.

The Target NACKs the Read transfer issued for a Group Address. If the transfer with the Group address is a Private write transfer or a Broadcast or Directed CCC write (including Vendor-specific write CCC) transfer, then the Target acts upon it as it does with the I^3C Dynamic Address.

The Virtual Target, using the shared peripheral logic, reports its Group Address capabilities using the GETCAPS CCC. Each Virtual Target supports multiple groups, up to a maximum of four.

The Target’s shared Peripheral logic matches incoming Private Write transfers addressed to any Group Address and delivers them internally to the appropriate logic for each Virtual Target assigned to that Group Address.

24.4.5.10.1. Group Address Support for Virtual Target

For better utilization of the resources and register space, the Target mode supports the sharing of Group Addresses between the configured Virtual Targets.

The acceptance of Group Addresses by each Virtual Target 'x' through SETGRPA CCC is determined by checking the following conditions:

- Whether the assigned Group Addresses to Virtual Target 'x' are less than four.
- Whether the total number of unique Group Addresses is less than four.

If either of these conditions is not met, the SETGRPA CCC sent to Virtual Target 'x' is NACK'd Common Command Codes (CCC).

24.4.6. DMA Operation

A DMA operation can be initiated by setting I3CxCTRL[DMAEN] = `1`. Different DMA triggers are available, as shown below:

- TX FIFO
- RX FIFO
- Target Channel 0 TX
- Target Channel 1 TX

TX and RX FIFO triggers will be generated based on the threshold level configured in I3CxBUFTHLD (TXTHLD/RXTHLD). Target channel 0 and 1 TX triggers will be generated based on the channel selected in I3CxDMACHSEL and the threshold level configured in I3CxECRTCON[EXTTXBUFTHLD].

Note: The DMA channel that is used should be configured in I3CxCON for the acknowledgment. The next trigger for DMA will be based on the acknowledgment received.

24.4.7. SRAM Error Correcting Code (ECC)

The module is capable of correcting one-bit errors and detecting two-bit errors in the SRAM read data stream.

Each 32-bit RAM word has corresponding ECC parity bits for single-bit error correction (SEC) and double-bit error detection (DED). Any single-bit error across the 32 bits of data or 7 parity bits in a RAM word is corrected on reads. Any two-bit errors in a RAM word are detected. More than two-bit errors in a single RAM word may not be correctly classified as an uncorrectable DED error.

Once read, the SRAM contents are examined in real-time. If a single-bit error is detected, the module automatically corrects the bit error on the read data bus while capturing the ECC value, and it invokes an interrupt to the CPU (IFSx[I3CxSEIF]). This informs the user to take the appropriate course of action for the given event. If the error is dual-bit, the same (uncorrected) data is presented as output and invokes an interrupt to the CPU (IFSx[I3CxDEIF]). This informs the user to take the appropriate course of action for the given event.

During write operations, ECC parity bits are automatically calculated and written for each 32-bit RAM word.

24.4.7.1. ECC FAULT INJECTION

Fault injection is enabled by setting the I3CCON.FLTINJ bit. When fault injection is enabled, bit errors are injected on RAM reads at the RAM word address specified by the ECCFADDR register. Either a single bit error or a double bit error can be generated. The ECCFPTR register has two pointer fields. Each field can be independently configured to select either no error or the location of the bit error. The bit error target can be any one of the 32 data bits or 7 ECC bits in a RAM word.

24.5. Application Examples

24.5.1. Controller Mode

Example 24-1. Controller Mode: Dynamically Address the Target and Perform Private Write and Read

```
typedef union I3C_CMD_TransferCommandDataStruct_Tag
{
    struct
{
    uint32_t CMD_ATTR:3;
    uint32_t TID:4;
    uint32_t CMD:8;
    uint32_t CP:1;
    uint32_t DEV_INDEX:5;
    uint32_t SPEED:3;
    uint32_t RESERVED:1;
    uint32_t DBP:1;
    uint32_t ROC:1;
    uint32_t SDAP:1;
    uint32_t RnW:1;
    uint32_t TGT_RST:1;
    uint32_t TOC:1;
    uint32_t PEC:1;
}F;
uint32_t TransferCommand;
} I3C_CMD_TransferCommandDataStruct_T;

typedef union I3C_CMD_TransferArgumentDataStruct_Tag
{
struct
{
    uint32_t CMD_ATTR:3;
    uint32_t RESERVED:5;
    uint32_t DB:8;
    uint32_t DL:16;
}F;
uint32_t TransferArgument;
} I3C_CMD_TransferArgumentDataStruct_T;

typedef union I3C_CMD_ShortDataArgumentStruct_Tag
{
    struct
    {
        uint32_t CMD_ATTR:3;
        uint32_t BYTE_STRB0:1;
        uint32_t BYTE_STRB1:1;
        uint32_t BYTE_STRB2:1;
        uint32_t RESERVED:2;
        uint32_t DATA_BYTE_0:8;
        uint32_t DATA_BYTE_1:8;
        uint32_t DATA_BYTE_2:8;
    }F;
    uint32_t ShortDataArgument;
} I3C_CMD_ShortDataArgumentDataStruct_T;

typedef union I3C_CMD_AddrAssignCommandDataStruct_Tag
{
    struct
    {
        uint32_t CMD_ATTR:3;
        uint32_t TID:4;
        uint32_t CMD:8;
        uint32_t RESERVED:1;
        uint32_t DEV_INDEX:5;
        uint32_t DEV_COUNT:5;
        uint32_t ROC:1;
        uint32_t RESERVED1:3;
        uint32_t TOC:1;
        uint32_t RESERVED2:1;
    }F;
    uint32_t AddrAssignCommand;
} I3C_CMD_AddrAssignCommandDataStruct_T;
typedef union I3C_ResponseDataStruct_Tag
```

```
{
    struct
    {
        uint32_t DL:16;
        uint32_t CCCT:8;
        uint32_t TID:4;
        uint32_t ERR_STS:4;
    }F;
    uint32_t RespondData;
} I3C_ResponseDataStruct_T;

typedef struct I3C_DCT_Tag
{
    uint32_t DCT_LOC1;
    uint32_t DCT_LOC2;
    uint32_t DCT_LOC3;
    uint32_t DCT_LOC4;

} I3C_DCT_T;

I3C_CMD_AddrAssignCommandDataStruct_T cmd;
I3C_CMD_TransferCommandDataStruct_T transCmd;
I3C_CMD_TransferArgumentDataStruct_T transArg;
I3C_ResponseDataStruct_T respQ;
I3C_DCT_T target1;
uint32_t receivedData[100];

int main()
{
/* Configure System Clock and I3C Clock Generator*/



   /*Enable Internal pull up which acts as Bus High Keeper*/
    _CNPUC4 = 1;
    _CNPUC5 = 1;
/
********************************************************************************
*/
    I3C1CONbits.ON = 1 ;                              // I3C module enable
    I3C1CTRLEXTbits.DEVOPMOD = 0 ;                    // Controller mode
    I3C1ADDbits.DYNADDR = 0x4C;                       // Controller Self Dynamic
Address Assignment
    I3C1ADDbits.DYNADDRVALID = 1;                     // Dynamic Address
valid

    I3C1SCLODTIM = 0x0007001A;                        // Configure SCL Open Drain
timing
    I3C1PPTIM = 0x00070007;                           // Configure SCL Push pull
timing

    I3C1QUETHLDCON =  (I3C1QUETHLDCON & 0xFFFF00FF) | 0x00000000;   //
Configure response threshold to 1 level
    I3C1BUFTHLDbits.RXTHLD = 1 ;                       // Configure RX threshold
to 4 levels

    I3C1INTSTACONbits.RXTHLDSTAEN = 1;                 //Enable RX status
    I3C1INTCONbits.RXTHLDINTEN = 1;                    // Enable Rx interrupt
    I3C1INTSTACONbits.RESPQSTAEN = 1;                  // Enable Response status
    I3C1INTCONbits.RESPQINTEN = 1;                     // Enable response
interrupt

    I3C1CNTRLbits.IBA  = 1 ;                           // Include I3C broadcast
address
    /*Write address table (DAT)*/
    /* Target device is I3C
     * Dynamic address 0x33, after calculate odd parity 0xB3
     * IBI with data supported
     * Static address 0x45 */
    I3C1DEVADDRTAB1LOC1= 0x00B31045;

    I3C1CNTRLbits.ENABLE  = 1 ;                        // Enable I3C operation

    /* Enable interrupt signal*/
    IFS11bits.I3C1GIF = 0;
    IEC11bits.I3C1GIE = 1;

/*****************Dynamic Addressing - ENTDAA
```

```
Command**********************/
    cmd.F.CMD_ATTR = 0x03;                  // Address Assignment Command
    cmd.F.TID = 0x1;                        // Transaction ID
    cmd.F.CMD = 0x07;                       // ENTDAA command
    cmd.F.DEV_COUNT = 1;                    // One device
    cmd.F.DEV_INDEX = 0;                    // 0th index
    cmd.F.TOC = 1;                          // Termination On Completion
    cmd.F.ROC = 1;                          // Response On Completion
    I3C1CMDQUE = cmd.AddrAssignCommand;     // Write to CMD queue

   /**************Controller Private Write******************************/
    /*Write data to Transmit FIFO*/
    I3C1TXRXDATA = 0x5555AAA1;
    I3C1TXRXDATA = 0x5555AAA2;
    I3C1TXRXDATA = 0x5555AAA3;
    I3C1TXRXDATA = 0x5555AAA4;

    transArg.F.CMD_ATTR = 0x1;              // Transfer Argument
    transArg.F.DL = 0x10;                   // Data length

    transCmd.F.CMD_ATTR = 0;                // Transfer Command
    transCmd.F.TID = 0x2;                   // Transaction ID
    transCmd.F.TOC = 1;                     // Termination On Completion
    transCmd.F.ROC = 1;                     // Response On Completion
    I3C1CMDQUE = transArg.TransferArgument;  // Write to CMD queue
    I3C1CMDQUE = transCmd.TransferCommand;  // Write to CMD queue

/*********************Controller Private Read***************************/


    transArg.F.CMD_ATTR = 0x1;               // Transfer Argument
    transArg.F.DL = 0x10;                   // Data length

    transCmd.F.CMD_ATTR = 0;                 // Transfer Command
    transCmd.F.TID = 0x3;                    // Transaction ID
    transCmd.F.RnW = 1;                      // Read
    transCmd.F.TOC = 1;                     // Termination On Completion
    transCmd.F.ROC = 1;                     // Response On Completion
    I3C1CMDQUE = transArg.TransferArgument; // Write to CMD queue
    I3C1CMDQUE = transCmd.TransferCommand;  // Write to CMD queue

/
*******************************************************************************/

    while(1);
    return 0;
}

void __attribute__((interrupt, no_auto_psv)) _I3C1Interrupt()
{
/*Response interrupt is enabled so check if interrupt is due to response
received*/
    if(I3C1INTSTAbits.RESPQSTA)
    {
        respQ.RespondData = I3C1RESPQUE;
        if(respQ.F.ERR_STS == 0) // No error
        {
            if (respQ.F.TID = 0x1) // Transaction ID for ENTDAA
            {
                /*Read Device characteristic Table(DCT)*/
                target1.DCT_LOC1 = I3C1DEVCHARTAB1LOC1;
                target1.DCT_LOC2 = I3C1DEVCHARTAB1LOC2;
                target1.DCT_LOC3 = I3C1DEVCHARTAB1LOC3;
                target1.DCT_LOC4 = I3C1DEVCHARTAB1LOC4;
            }
            if(respQ.F.TID = 0x2) // Transaction ID for transmit
            {

            }
            if(respQ.F.TID = 0x3) // Transaction ID for Receive
            {
                if(I3C1INTSTAbits.RXTHLDSTA) // Check threshold value, read
RXFIFO
                {
                    receivedData[0]= I3C1TXRXDATA;
                    receivedData[1]= I3C1TXRXDATA;
                    receivedData[2]= I3C1TXRXDATA;
                    receivedData[3]= I3C1TXRXDATA;
```

```
                    }
                }
            }
            else
            {
                /*Error handling*/
            }
        }
        IFS11bits.I3C1GIF = 0;
}
```

24.5.2. Target Mode

Example 24-2. Target Mode: After Dynamic Address is assigned, Perform Private Read and Write

```
typedef union I3C_TARGET_TRANSMIT_COMMAND_Tag
{
    struct
{
    uint32_t CMD_ATTR:3;
    uint32_t TID:3;
    uint32_t RESERVED:10;
    uint32_t DATA_LENGTH:16;
}F;
uint32_t TransmitCommand;
} I3C_TARGET_TRANSMIT_COMMAND_T;

I3C_TARGET_TRANSMIT_COMMAND_T cmd;
uint32_t receivedData[100];
uint16_t dataCount;


int main()
{
/* Configure System Clock and I3C Clock Generator*/


   /*Enable Internal pull up which acts as Bus High Keeper*/
    _CNPUC4 = 1;
    _CNPUC5 = 1;

//
********************************************************************************
*/
    I3C1CONbits.ON = 1 ;                              // I3C module enable
    I3C1CTRLEXTbits.DEVOPMOD = 1 ;                    // Target mode

    I3C1TGTPID = 0x12345678;                           // Write PIDDCR,INSTID
PARTID
    I3C1TGTMIPIID = 0x00000559;                        // write PROVID,MIPIMFGID


    I3C1BUFTHLDbits.RXTHLD = 1;                        // Configure RX threshold
to 4 levels
    I3C1INTSTACONbits.RXTHLDSTAEN = 1;                 // Enable RX status
    I3C1INTCONbits.RXTHLDINTEN = 1;                    // Enable Rx interrupt

    I3C1CNTRLbits.ENABLE = 1;                          // Enable I3C operation

/* Enable interrupt signal*/
    IFS11bits.I3C1GIF = 0;
    IEC11bits.I3C1GIE = 1;
    /* I3C operation in target mode will be enabled only after receiving
     * Placeholder transfer from Controller.
     */
    while(!I3C1CNTRLbits.ENABLE);           // wait for I3C to enter Target
mode

   /*************Target Private Write****************************/
    /*Write data to Transmit FIFO*/
    I3C1TXRXDATA = 0x5555AAA1;
    I3C1TXRXDATA = 0x5555AAA2;
    I3C1TXRXDATA = 0x5555AAA3;
    I3C1TXRXDATA = 0x5555AAA4;
```

```
    cmd.F.CMD_ATTR = 0;                         // Transmit Command
    cmd.F.DATA_LENGTH = 0x10;                   // Data length

    I3C1CMDQUE = cmd.TransmitCommand;           // Write to CMD queue


    while(1);
    return 0;
}

void __attribute__((interrupt, no_auto_psv)) _I3C1Interrupt()
{
    if(I3C1INTSTAbits.RXTHLDSTA)                // Check if interrupt is for
received data
    {
        while((I3C1BUFLEVEL >> 16))             // Read all the data in RXFIFO
        {
            receivedData[dataCount++]= I3C1TXRXDATA;
        }
    }
    IFS11bits.I3C1GIF = 0;
}
```

24.6. Interrupts

There are three interrupts:

1. Generic interrupt that consolidates all the interrupt sources in the I3CxINTCON (I3CxGIF)
2. Single bit ECC error (I3CxSEIF)
3. Double bit ECC error (I3CxDEIF)

24.7. Operation in Power Saving modes

24.7.1. Sleep in Controller Mode Operation

When a device enters Sleep mode, the system clock used by the core processor and peripherals is halted.

If I3CxCON[SLPEN] = 1, the module will continue to request the I^3C peripheral clock when the device enters Sleep mode.

If I3CxCON[SLPEN] = 0, the module will discontinue requesting the I^3C peripheral clock when the device enters Sleep mode.

The SCL clock generator stops when the I3CxCON[SLPEN] = 0 because the I^3C peripheral clock stops. If the I^3C peripheral clock stops in the middle of a transaction, then the transmission/reception stops in place, and stopping the clock in the middle of the data communication will result in undefined behavior on the I^3C bus. It is highly recommended that you enter sleep mode to complete data communication with acknowledgment.

Upon exit from sleep, the clocks start up, the clock generator continues where it left off, and SCL resumes. Any in-progress transaction may continue, as the module still thinks that it is in the middle of a transaction and waits for more data. Hence, the following transaction is usually missed. However, the module snaps back as soon as it sees a stop on the bus. A Stop or a Restart on the bus means the transaction has ended, so the module can recover by itself.

There is no automatic way to prevent sleep entry if a transmission or reception is pending. The user software must synchronize sleep entry with I^3C operation to avoid aborted transmissions. It is highly recommended to restart the I^3C communication after exiting Sleep mode.

If the module clocks are suspended in Sleep mode while having Controller mode active, the module will most likely need to be re-enabled to synchronize it back up with the I^3C bus activity.

24.7.2. Sleep in Target Mode Operation

In Target mode, incoming SCL is used to receive or transmit data. However, it uses the I^3C peripheral clock to run the state machine of the Target to transmit and receive data from the application.

When operating in Target mode and the device is put into Sleep, the I^3C peripheral clock will not run the Target state machine.

The I^3C module does not function in Target mode operation while the device is in Sleep.

There is no automatic way to prevent sleep entry if a transmission or reception is pending. The user software must synchronize sleep entry with I^3C operation to avoid aborted transmissions. It is highly recommended to restart the I^3C communication after exiting Sleep mode.

If the module clocks are suspended in Sleep mode while having Target mode active, the module will most likely need to be re-enabled to synchronize it back up with the I^3C bus activity.

24.7.3. Idle in Controller Mode Operation

When the device enters Idle mode, all clock sources remain functional. If the module intends to power down, it disables its own clocks. If the module clocks are suspended in Idle mode while having controller mode active, the module will most likely need to be re-enabled to synchronize it back up with the I^3C bus activity.

24.7.4. Idle in Target Mode Operation

When the device enters Idle mode, all clock sources remain functional. If the module intends to power down, it disables its own clocks.

In Target mode, incoming SCL is used to receive or transmit data. However, it uses the I^3C peripheral clock to run the state machine of the Target to transmit and receive data from the application.

If the module clocks are suspended in Idle mode while having target mode active, the module will most likely need to be re-enabled to synchronize it back up with the I^3C bus activity.

25. Single-Edge Nibble Transmission (SENT)

SENT is a unidirectional, single-wire communications protocol that is based on SAE J2716, "SENT – Single-Edge Nibble Transmission for Automotive Applications". The protocol supports point-to-point transmission of signal values using a signal system based on successive falling edges. It allows for high-resolution data transmission with a lower system cost than available serial data solutions, and it is intended for use in applications where data need to be communicated from a sensor to a central controller, such as an Engine Control Unit (ECU).

The 16-bit SENT module is a dedicated hardware implementation of SAE J2716. The module can be configured for three main modes of operation:

- Asynchronous Transmitter (default)
- Synchronous Transmitter
- Receiver

The module also includes these features:

- Automatic Data Rate Synchronization
- Optional Automatic Detection of CRC Errors in Receive Mode
- Optional Hardware Calculation of CRC in Transmit Mode
- Support for Optional Pause Pulse Period
- Data Buffering for One Message Frame
- Selectable Data Length for Transmit/Receive from Three to Six Nibbles
- Automatic Detection of Framing Errors
- Separately Mappable Input and Output Functions on Devices with Peripheral Pin Select (PPS)

25.1. Device-Specific Information

Table 25-1. SENT Summary Table

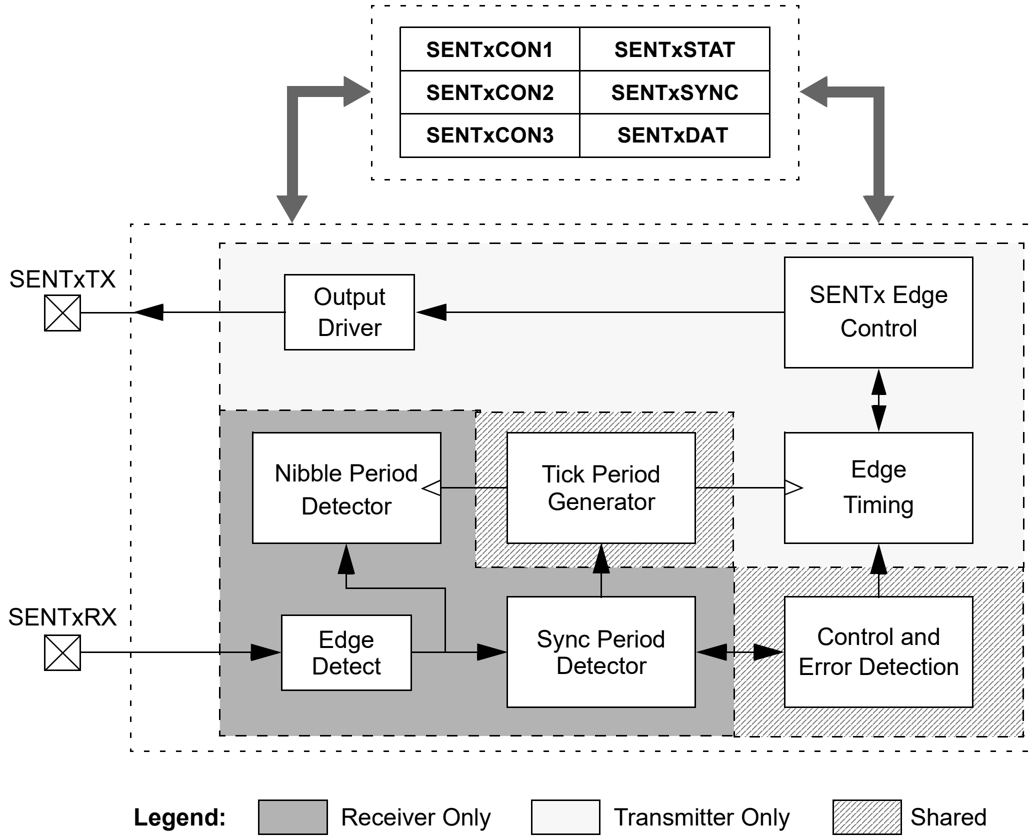
SENT Module Instances	PPS Available	Peripheral Bus Speed	Clock Source
1	All Instances	Standard (1:2 CPU Clock)	Standard Speed Peripheral Clock

25.2. Architectural Overview

SENT messages are encoded and decoded based on the time between falling edges. The protocol's timing is based on a predetermined time unit, T_{TICK}, which can vary from 3 to 90 µs. Both the transmitter and receiver must be preconfigured for the same value of T_{TICK}. The SENT specification allows messages to be validated with up to a 20% variation in T_{TICK}. This allows for the transmitter and receiver to run from different clocks that may be inaccurate and drift with time and temperature.

An overview of the SENT module is shown in Figure 25-1.

Figure 25-1. SENTx Module Block Diagram

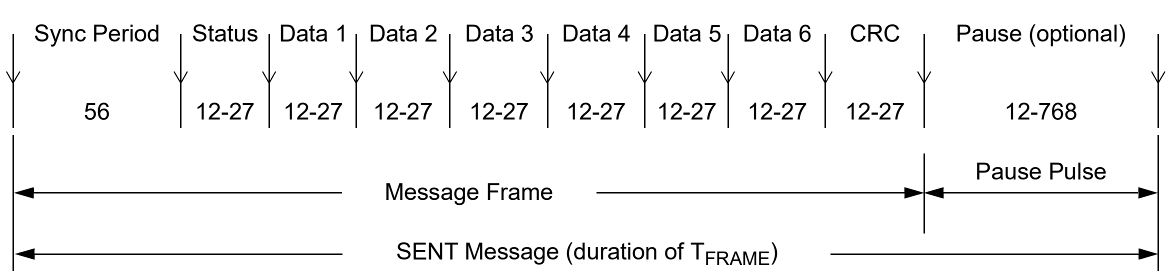


A SENT message consists of the following:

- A Synchronization/Calibration Period (pulse) of 56 Tick Times
- A Status Nibble of 12 to 27 Tick Times
- Up to Six Data Nibbles of 12 to 27 Tick Times
- A CRC Nibble of 12 to 27 Tick Times
- An Optional Pause Pulse Period of 12 to 768 Tick Times

The period from the start of the Sync period to the end of the CRC nibble comprises the message frame. When the optional Pause period is present, this makes one SENT message with a length of T_{FRAME} (µs). Figure 25-2 shows the construction of a typical six-nibble data frame, with the numbers representing the minimum or maximum number of tick times for each section.

Figure 25-2. SENT Message Format



The Sync period starts the message frame and is used for synchronization of T_{TICK} between the transmitter and receiver. When configured for Transmit mode, the module drives the line low for five ticks and to a High-Impedance state for 51 ticks.

A four-bit status nibble follows the Sync pulse and may be used for device status, identification or alternatively, used as additional data. The status nibble is formatted the same as a data nibble.

After the status nibble is one or more (up to six) data nibbles. They are four bits in length and are encoded as the data value plus 12 ticks. This yields a minimum value of 12 ticks for 0h and a maximum value of 27 ticks for Fh. When configured for Transmit mode, the module drives the line low for five ticks and into a High-Impedance state for the remaining 7 to 22 ticks.

The CRC data nibble follows the data payload. This is a 4-bit CRC of the six data nibbles only. The CRC is calculated using a polynomial of, $x^4 + x^3 + x^2 + 1$, with a seed value of '`0101`'. It is then padded with '`0`' to help detect shift errors. The CRC nibble is formatted the same as a data nibble.

Since the data values are encoded in the time between falling edges, the SENT protocol may produce a variable length message. In some applications, the pause pulse period is used to pad the message length so that messages will always be received at a constant time interval. The module provides support to automatically calculate the pause duration needed for periodic transmissions. When configured for Transmit mode, the module drives the line low for five ticks and into a High-Impedance state for the remaining pause time.

Note: A SENT message frame will always have a status and CRC nibble. The shortest message frame with one data nibble (SENTxCON1[2:0] = `001`) will have a length of one Sync and three nibbles.

25.2.1. Short and Enhanced Serial Message Formats

The J2716 specification defines two optional message formats: a Short Serial Message format and an Enhanced Serial Message format. Both message formats encode a longer serial message using two bits of the status nibble. A single message is encoded using multiple SENT data frames.

If the module is configured as a transmitter, the application must encode these serial messages by writing the appropriate value. If the module is configured as a receiver, the application must decode these serial messages by storing and analyzing the contents of the received status nibbles.

25.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x19C0	SENT1CON1	31:24								
		23:16								
		15:8	ON		SIDL		RCVEN	TXM	TXPOL	CRCEN
		7:0	PPP	SPCEN		PS		NIBCNT[2:0]		
0x19C4	SENT1CON2	31:24								
		23:16								
		15:8	TICKTIMESYNCMAX[15:8]							
		7:0	TICKTIMESYNCMAX[7:0]							
0x19C8	SENT1CON3	31:24								
		23:16								
		15:8	FRAMETIMESYNCMIN[15:8]							
		7:0	FRAMETIMESYNCMIN[7:0]							
0x19CC	SENT1STAT	31:24								
		23:16								
		15:8								
		7:0	PAUSE	NIB[2:0]			CRCERR	FRMERR	RXIDLE	SYNCTXEN
0x19D0	SENT1SYNC	31:24								
		23:16								
		15:8	SENTSYNC[15:8]							
		7:0	SENTSYNC[7:0]							
0x19D4	SENT1DAT	31:24	STAT[3:0]				DATA1[3:0]			
		23:16	DATA2[3:0]				DATA3[3:0]			
		15:8	DATA4[3:0]				DATA5[3:0]			
		7:0	DATA6[3:0]				CRC[3:0]			
0x19D8 ... 0x19DF	Reserved									
0x19E0	SENT2CON1	31:24								
		23:16								
		15:8	ON		SIDL		RCVEN	TXM	TXPOL	CRCEN
		7:0	PPP	SPCEN		PS		NIBCNT[2:0]		
0x19E4	SENT2CON2	31:24								
		23:16								
		15:8	TICKTIMESYNCMAX[15:8]							
		7:0	TICKTIMESYNCMAX[7:0]							
0x19E8	SENT2CON3	31:24								
		23:16								
		15:8	FRAMETIMESYNCMIN[15:8]							
		7:0	FRAMETIMESYNCMIN[7:0]							
0x19EC	SENT2STAT	31:24								
		23:16								
		15:8								
		7:0	PAUSE	NIB[2:0]			CRCERR	FRMERR	RXIDLE	SYNCTXEN
0x19F0	SENT2SYNC	31:24								
		23:16								
		15:8	SENTSYNC[15:8]							
		7:0	SENTSYNC[7:0]							
0x19F4	SENT2DAT	31:24	STAT[3:0]				DATA1[3:0]			
		23:16	DATA2[3:0]				DATA3[3:0]			
		15:8	DATA4[3:0]				DATA5[3:0]			
		7:0	DATA6[3:0]				CRC[3:0]			

25.3.1. SENTx Control Register 1

Name: SENTxCON1
Offset: 0x19C0, 0x19E0

Notes:

1. The bit has no function when RCVEN = 1.
2. The bit has no function when RCVEN = 0.
3. CCP3 OC output is internally connected with the SENT1OUT pin; CCP4 OC output is internally connected with the SENT2OUT pin.

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		SIDL		RCVEN	TXM	TXPOL	CRCEN
Access	R/W		R/W		R/W	R/W	R/W	R/W
Reset	0		0		0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PPP	SPCEN		PS		NIBCNT[2:0]		
Access	R/W	R/W		R/W		R/W	R/W	R/W
Reset	0	0		0		1	1	0

Bit 15 – ON SENTx Enable bit

Value	Description
1	Module is enabled.
0	Module is disabled.

Bit 13 – SIDL SENTx Stop in Idle Mode bit

Value	Description
1	Module stops operation in Idle mode.
0	Module continues operation in Idle mode.

Bit 11 – RCVEN SENTx Receive Enable bit

Value	Description
1	Module operates as a receiver.
0	Module operates as a transmitter.

Bit 10 – TXM SENTx Transmit Mode bit**(1)**

Value	Description
1	Module transmits a data frame only when triggered using the SYNCTXEN status bit.
0	Module transmits a data frame continuously while enabled.

Bit 9 – TXPOL SENTx Transmit Polarity bit[1]

Value	Description
1	Idle state of the data output pin is low.
0	Idle state of the data output pin is high.

Bit 8 – CRCEN CRC Enable bit

In Receive Mode (RCVEN = 1):
1 = CRC verification is performed using the J2716 method.
0 = CRC verification is not performed.
In Transmit Mode (RCVEN = 0):
1 = CRC is calculated using the J2716 method.
0 = CRC is not calculated.

Bit 7 – PPP Pause Pulse Present bit

Value	Description
1	SENTx messages transmitted/received with a Pause pulse.
0	SENTx messages transmitted/received without a Pause pulse.

Bit 6 – SPCEN Short PWM Code Enable bit[2, 3]

Value	Description
1	SPC control from an external source is enabled.
0	SPC control from an external source is disabled.

Bit 4 – PS Prescale Select bit

Value	Description
1	1:4 (module clock is F_{SENT}/4)
0	1:1 (module clock is F_{SENT})

Bits 2:0 – NIBCNT[2:0] Nibble Count Control bits

Value	Description
111	Reserved: do not use.
110	Six data nibbles per data packet
101	Five data nibbles per data packet
100	Four data nibbles per data packet
011	Three data nibbles per data packet
010	Two data nibbles per data packet
001	One data nibble per data packet
000	Reserved: do not use.

25.3.2. SENTx Control Register 2

Name: SENTxCON2
Offset: 0x19C4, 0x19E4

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	TICKTIMESYNCMAX[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	TICKTIMESYNCMAX[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:0 – TICKTIMESYNCMAX[15:0] Tick or Maximum Synchronization Period bits

Module in Transmit Mode (RCVEN = 0):

TICKTIMESYNCMAX[15:0]: This register value specifies the period for the tick clock generator.

Module in Receive Mode (RCVEN = 1):

TICKTIMESYNCMAX[15:0]: This register value specifies the maximum time limit for a valid Sync period.

25.3.3. SENTx Control Register 3

Name: SENTxCON3
Offset: 0x19C8, 0x19E8

Note:
1. The module will not produce a Pause period with less than 12 ticks, regardless of the FRAMETIME[15:0] value. FRAMETIME[15:0] values beyond 2047 will have no effect on the length of a data frame.

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	FRAMETIMESYNCMIN[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	FRAMETIMESYNCMIN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:0 – FRAMETIMESYNCMIN[15:0] Frame Ticks or Minimum Synchronization Period bits
Module in Transmit Mode (RCVEN = 0):
FRMTIMESYNCMIN[15:0] This register value specifies the total number of ticks in a data frame if PPP = 1.[(1)]
Module in Receive Mode (RCVEN = 1):
FRMTIMESYNCMIN[15:0] This register value specifies the minimum time limit for a valid Sync period.

25.3.4. SENTx Status Register

Name: SENTxSTAT
Offset: 0x19CC, 0x19EC

Note:
1. This bit is read-only in Receive mode and writable (settable and clearable) in Transmit mode.

Legend: R = Readable bit; W = Writable bit; C = Clearable bit; HC = Hardware Clearable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	PAUSE	NIB[2:0]			CRCERR	FRMERR	RXIDLE	SYNCTXEN
Access	R	R	R	R	R/C	R/C	R	R/W/HC
Reset	0	0	0	0	0	0	0	0

Bit 7 – PAUSE Pause Period Status bit

Value	Description
1	The module is transmitting/receiving a Pause period.
0	The module is not transmitting/receiving a Pause period.

Bits 6:4 – NIB[2:0] Nibble Status bits
In Transmit Mode (RCVEN = 0):
111 = Module is transmitting CRC nibble.
110 = Module is transmitting Data Nibble 6.
101 = Module is transmitting Data Nibble 5.
100 = Module is transmitting Data Nibble 4.
011 = Module is transmitting Data Nibble 3.
010 = Module is transmitting Data Nibble 2.
001 = Module is transmitting Data Nibble 1.
000 = Module is transmitting status nibble or Pause period, or it is not transmitting.
In Receive Mode (RCVEN = 1):
111 = Module is receiving CRC nibble or was receiving this nibble when an error occurred.
110 = Module is receiving Data Nibble 6 or was receiving this nibble when an error occurred.
101 = Module is receiving Data Nibble 5 or was receiving this nibble when an error occurred.
100 = Module is receiving Data Nibble 4 or was receiving this nibble when an error occurred.
011 = Module is receiving Data Nibble 3 or was receiving this nibble when an error occurred.
010 = Module is receiving Data Nibble 2 or was receiving this nibble when an error occurred.
001 = Module is receiving Data Nibble 1 or was receiving this nibble when an error occurred.

000 = Module is receiving Status nibble or waiting for Sync.

Bit 3 – CRCERR CRC Status bit (Receive mode only)

Value	Description
1	A CRC error occurred for the data nibbles in the SENTxDATA register.
0	No CRC error has occurred.

Bit 2 – FRMERR Framing Error Status bit (Receive mode only)

Value	Description
1	A data nibble was received with less than 12 tick periods or greater than 27 tick periods.
0	No framing error has occurred.

Bit 1 – RXIDLE Receiver Idle Status bit (Receive mode only)

Value	Description
1	The SENTx data bus has been Idle (high) for a period of SYNCMAX or greater.
0	The SENTx data bus is not Idle.

Bit 0 – SYNCTXEN Synchronization Period Status/Transmit Enable bit[(1)]

In Receive Mode (RCVEN = 1):
1 = A valid synchronization period was detected; the module is receiving nibble data.
0 = No synchronization period has been detected; the module is not receiving nibble data.
In Synchronous Transmit Mode (RCVEN = 0, TXM = 1):
1 = The module is transmitting a SENT data frame.
0 = The module is not transmitting a data frame; software may set SYNCTXEN to start another data frame transmissio.n
In Asynchronous Transmit Mode (RCVEN = 0, TXM = 0):
SYNCTXEN always reads as ‘1’, indicating the module is transmitting frames continuously.

25.3.5. SENTx Synchronization Time Capture Register

Name: SENTxSYNC
Offset: 0x19D0, 0x19F0

Note:

1. These register bits are not available in Transmit mode (RCVEN = 0).

Legend: R = Readable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	SENTSYNC[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SENTSYNC[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – SENTSYNC[15:0] Captured Sync Period bits[1]

25.3.6. SENTx Data Register

Name: SENTxDAT
Offset: 0x19D4, 0x19F4

Notes:
1. Register bits are read-only in Receive mode (RCVEN = 1).
2. In Transmit mode, the CRC[3:0] bits are read-only when automatic CRC calculation is enabled (RCVEN = 0, CRCEN = 1).

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	STAT[3:0]				DATA1[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	DATA2[3:0]				DATA3[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	DATA4[3:0]				DATA5[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA6[3:0]				CRC[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:28 – STAT[3:0] Status Nibble Data bits[(1)]

Bits 27:24 – DATA1[3:0] Data Nibble #1 Data bits[(1)]

Bits 23:20 – DATA2[3:0] Data Nibble #2 Data bits[(1)]

Bits 19:16 – DATA3[3:0] Data Nibble #3 Data bits[(1)]

Bits 15:12 – DATA4[3:0] Data Nibble #4 Data bits[(1)]

Bits 11:8 – DATA5[3:0] Data Nibble #5 Data bits[(1)]

Bits 7:4 – DATA6[3:0] Data Nibble #6 Data bits[(1)]

Bits 3:0 – CRC[3:0] CRC Nibble Data bits[(1,2)]

25.4. Operation

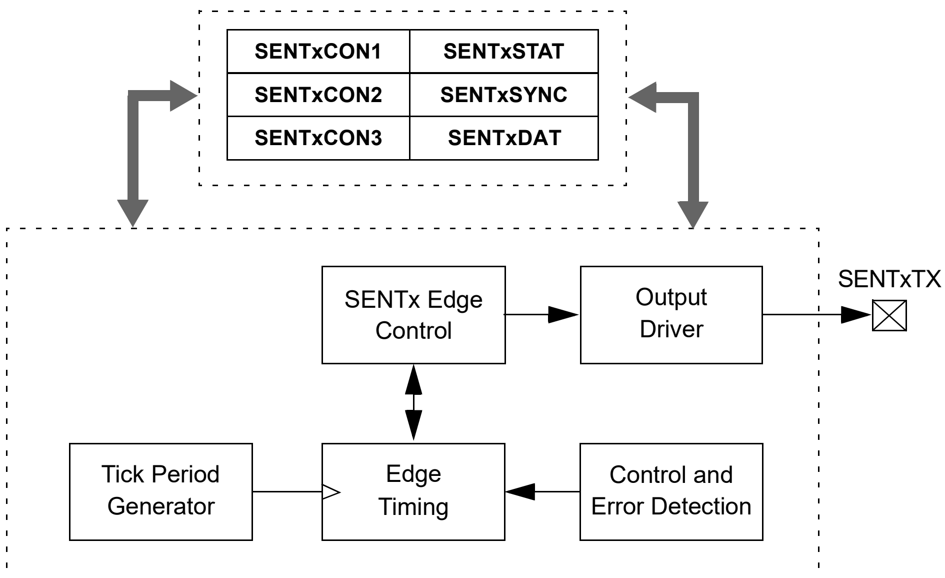
25.4.1. Transmit Mode

When RCVEN (SENTxCON1[11]) = 0, the module operates as a transmitter. Message frames are generated using the Configuration and Data registers.

The module has two Transmit Operating modes selected by the TXM bit (SENTxCON1[10]). Asynchronous mode (TXM = 0) continuously sends data message frames when the ON bit (SENTxCON1[15]) is set. Synchronous mode (TXM = 1) sends messages under software control to support additional capabilities, including Short PWM Code (SPC).

A block diagram of Transmit mode is shown in Figure 25-3.

Figure 25-3. SENTx Transmit Mode Block Diagram



25.4.1.1. Timing Calculations for Transmit Mode

In Transmit mode, SENTxCON2 and SENTxCON3 hold the values for TICKTIME and FRAMETIME. The tick period used by the SENT transmitter (T_{TICK}) is set by writing TICKTIME to the SENTxCON2 register. The tick period calculations are shown in Equation 25-1. The F_{CLK} value is either F_{SENT} or $F_{SENT}/4$, depending on the value of the Prescaler Enable bit, PS (SENTxCON1[4]).

Equation 25-1. Tick Period Calculation

$$TICKTIME = (T_{TICK} \cdot F_{CLK}) - 1$$

Where:

$$F_{CLK} = F_{SENT}/Prescaler$$

Note: F_{SENT} is the SENT Clock Frequency.

If the Pause pulse is to be used, a frame period (T_{FRAME}) must be defined. This is done by writing the value of FRAMETIME to SENTxCON3. The formulas used to calculate the value of FRAMETIME (in the same units as T_{TICK}) are shown in Equation 25-2. The FRAMETIME ranges for all settings are summarized in Table 25-2.

Equation 25-2. Calculating FRAMETIME

$$FRAMETIME = T_{FRAME}\,(\mu s)/T_{TICK}$$

Where:

$848 + 12N \geq FRAMETIME \geq 122 + 27N$, for $1 \leq N \leq 6$

Table 25-2. Range of FRAMETIME Values

Number of Data Nibbles	Min. FRAMETIME Value	Max. FRAMETIME Value
1	149	860
2	176	872
3	203	884
4	230	896
5	257	908
6	284	920

25.4.1.2. CRC Calculation

The module can optionally calculate the CRC using the recommended method shown in Example 25-1. The CRC is calculated when the CRCEN bit (SENTxCON1<8>) is set and the CRC[3:0] (SENTxDAT[3:0]) register bits become read-only. The hardware computed CRC value is indicated in the CRC[3:0] register bits when the calculation is finished.

When CRCEN = `0`, no CRC is computed in hardware and the CRC[3:0] bits become writable by software. The application must compute a CRC value and write it to CRC[3:0].

Example 25-1. Recommended J2716 CRC Implementation

```
#define NUM_NIBBLES 6
// Array holding received nibbles
rec_data[NUM_NIBBLES];
// CRC lookup table
crc_table = {0,13,7,10,14,3,9,4,1,12,6,11,15,2,8,5};
// Initialize checksum to seed value
Checksum = 5;
// For each data nibble, bit-wise XOR with lookup value from table
for(i=0;i<NUM_NIBBLES;i++)
{
    Checksum = rec_data[i] ^ crc_table[Checksum];
}
// Bit-wise XOR with additional 0 value
Checksum = 0 ^ crc_table[Checksum];
```

25.4.1.3. Transmitter Status Bits

The SENTxSTAT register provides status information and control when in Transmit mode. The NIB[2:0] status bits (SENTxSTAT[6:4]) display the current data nibble transmitting during a message frame. If the pause period is enabled (PPP (SENTxCON1[7]) = `1`), the PAUSE bit (SENTxSTAT[7]) indicates when a pause period transmission is in progress.

The SYNCTXEN bit (SENTxSTAT[0]) is used to initiate a synchronous transmission when TXM is set. SYNCTXEN is automatically cleared by hardware when all data nibbles, the CRC nibble and the pause period have completed.

25.4.1.4. Transmit Polarity Option

The polarity of the SENT data pin can be inverted by setting the TXPOL bit (SENTxCON1[9]). This feature can be useful for implementing an external transistor drive circuit. Note that the High-Impedance (Idle) condition on the line remains in the same state (high) due to the external pull-up resistor network.

25.4.1.5. Transmit Output Pin and PPS

The SENTx Transmit (SENTxTX) pin function is a remappable feature. To use the module in Transmit mode, SENTxTX must be mapped to an available I/O pin using the appropriate RPORx register.

25.4.1.6. Asynchronous Transmitter Mode

The module is, by default, configured as an asynchronous transmitter. In this mode, the module continuously transmits message frames as long as the ON bit is set (SENTxCON1[15]). The final falling edge of the CRC nibble also serves as the first falling edge of the Sync pulse. An interrupt is generated at the completion of the CRC nibble.

Figure 25-4 and Figure 25-5 show the relationship between the control, status and interrupt events.

Figure 25-4. SENTx Data Transmission, Asynchronous Mode

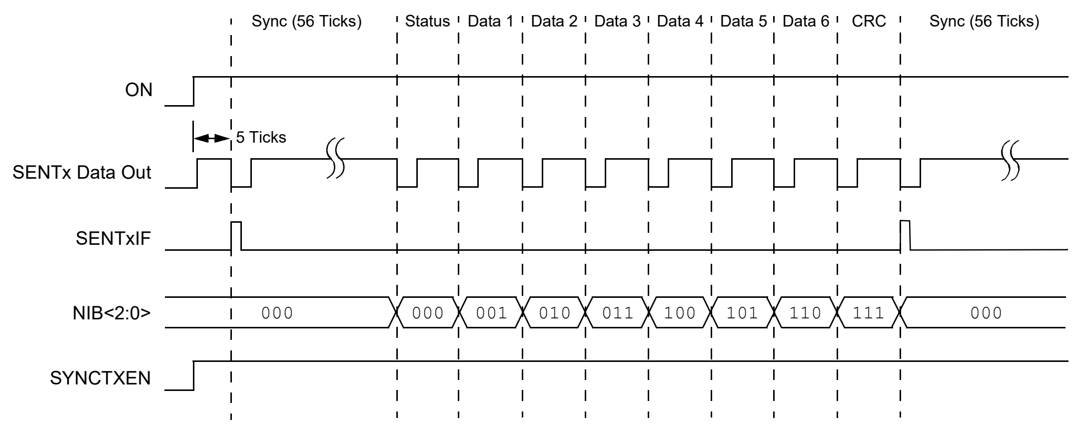
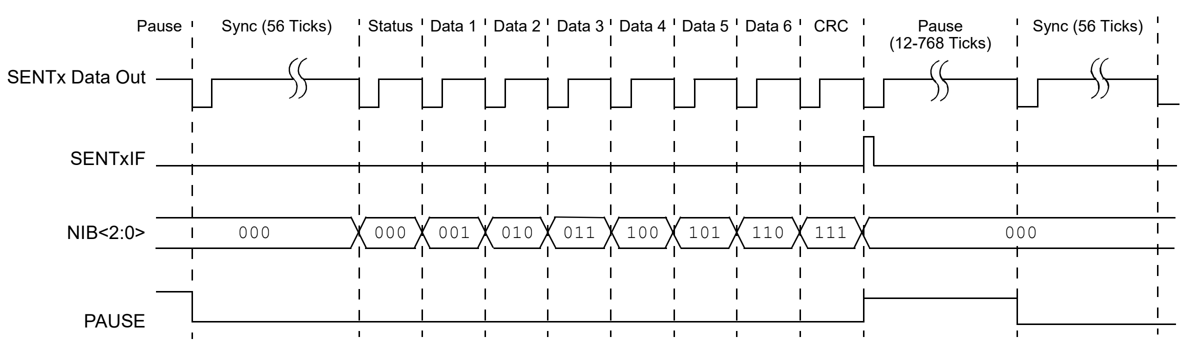


Figure 25-5. SENTx Data Transmission with Pause Period



To fully configure the module, the following must be known:

- Tick Time, T_{TICK} (Equation 25-1)
- Number of data nibbles, *N*
- Hardware or application calculated CRC
- Use of pause period for a fixed message period
- The overall duration of the SENT Message if the pause period is present (Equation 25-2).

To initialize the module:

1. Clear RCVEN (SENTxCON1[11]) for Transmit mode.
2. Clear TXM (SENTxCON1[10]) for asynchronous transmit.
3. Write the value of the desired frame length to NIBCNT[2:0] (SENTxCON1<2:0>).
4. Set or clear CRCEN (SENTxCON1[8]) to configure the module for hardware or software CRC calculation.
5. Write the value for the desired TICKTIME to SENTxCON2.
6. If the optional pause period is required, set PPP (SENTxCON1[7]) to enable the feature, then write the value of FRAMETIME to SENTxCON3.
7. Enable the SENT interrupt(s) and set the interrupt priority.
8. Write the initial status and data values to SENTxDATA. If application-based CRC is being used (CRCEN = 0), also calculate the message CRC and write it to CRC<[:0].
9. Set ON (SENTxCON1[15]) to enable the module.

Updates to SENTxDATA must be performed after the completion of the CRC and before the next message frame's status nibble. The recommended method is to use the message frame completion interrupt to trigger data writes.

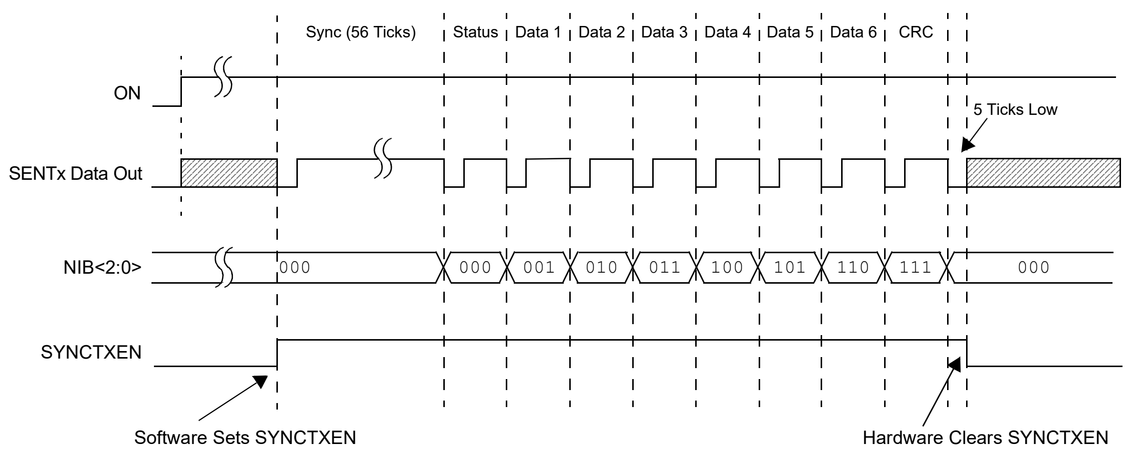
Example 25-2. SENT1 Asynchronous Transmission Code

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#define mFclk (4E+6)
#define mTickTime (70E-6)
#define mFrameTime (25E-3)
int main(void) {
    _RP26R = 81; // Assign SENT1OUT to pin RP26
    SENT1CON1bits.RCVEN = 0;                        // Module operates as a transmitter
    SENT1CON1bits.TXM = 0;                          // Asynchronous Transmit
    SENT1CON1bits.NIBCNT = 6;                       // 6 data nibbles per data packet
    SENT1CON1bits.CRCEN = 1;                        // CRC is calculated using the J2716
method
    SENT1CON1bits.PPP = 1;                          // SENTx messages transmitted with
Pause Pulse
    SENT1CON1bits.PS = 0;                           // Module clock is FSENT
    SENT1CON2 = ( (mTickTime * mFclk) - 1);         // TICKTIME
    SENT1CON3 = ( mFrameTime / mTickTime) ;         // FRAMETIME
    _SENT1IE = 1; // Enable SENT1 interrupt
    _SENT1IP = 4; // SENT interrupt priority
    SENT1DATbits.STAT = 0;                         // Status Nibble
    SENT1DATbits.DATA1 = 1;                        // Data Nibble 1
    SENT1DATbits.DATA2 = 2;                        // Data Nibble 2
    SENT1DATbits.DATA3 = 3;                        // Data Nibble 3
    SENT1DATbits.DATA4 = 4;                        // Data Nibble 4
    SENT1DATbits.DATA5 = 5;                        // Data Nibble 5
    SENT1DATbits.DATA6 = 6;                        // Data Nibble 6
    SENT1CON1bits.ON = 1;                          // Enable SENT module
    while(1);
    return 0;
}
void __attribute__((__interrupt__,__auto_psv__)) _SENT1Interrupt (void)
{
_SENT1IF = 0; // Clear interrupt flag
// Update SENT1DAT here
}
```

25.4.1.7. Synchronous Transmitter Mode

The module can be alternatively configured as a synchronous transmitter. In this mode, the module will transmit only one message frame each time the SYNCTXEN bit (SENTxSTAT[0]) is set. When the data frame is complete, the SYNCTXEN bit will be cleared in hardware. The line will be driven low for five ticks to complete the CRC nibble and then the line will tri-state and remain in the Idle state until SYNCTXEN is set again. An interrupt is generated, five ticks after the completion of the CRC nibble. Figure 25-6 shows the relationship between the control, status and interrupt events.

Figure 25-6. SENTx Data Transmission, Synchronous Mode



To fully configure the module, the following must be known:

- Tick Time, T_{TICK} (Equation 25-1)
- Number of data nibbles
- Hardware or application calculated CRC

To initialize the module for synchronous transmission:

1. Clear the RCVEN bit (SENTxCON1[11]) for Transmit mode.
2. Set the TXM bit (SENTxCON1[10]) for synchronous transmit operation.
3. Write the desired data frame length to NIBCNT[2:0] (SENTxCON1[2:0]).
4. Set or clear CRCEN (SENTxCON1[8]) to configure the module for the hardware or software CRC calculation.
5. Write the desired value for TICKTIME to SENTxCON2.
6. Enable the SENT interrupts and set the interrupt priority.
7. Set the ON bit (SENTxCON1[15]) to enable the module.

When the application is ready to transmit data:

1. Write the data to be transmitted to the SENTxDATA register.
2. Set the SYNCTXEN bit to begin transmission.

Updates to SENTxDATA must be performed after the completion of the CRC and before the next message frame's status nibble. The recommended method is to use the message frame completion interrupt or poll the SYNCTXEN bit.

Note: Software may need to wait additional time before starting a new message frame. The J2716 specification allows up to 18 µs of rise time on the SENT data line. The rise time will be a function of the external pull-up resistor and EMI filtering on the SENT data line. It is recommended that the wait time be longer than one sync time of 56 ticks + 20%.

Example 25-3. SENT1 Synchronous Transmission Code

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#define mFclk (4E+6)
#define mTickTime (70E-6)
#define mFrameTime (25E-3)
int main(void) {
    _RP26R = 81;                                    // Assign SENT1OUT to pin RP26
    SENT1CON1bits.RCVEN = 0;                        // Module operates as a transmitter
    SENT1CON1bits.TXM = 1;                          // Synchronous Transmit
    SENT1CON1bits.NIBCNT = 6;                       // 6 data nibbles per data packet
    SENT1CON1bits.CRCEN = 1;                        // CRC is calculated using the J2716
method
    SENT1CON1bits.PPP = 1;                          // SENTx messages transmitted with
Pause Pulse
    SENT1CON1bits.PS = 0;                           // Module clock is FSENT
    SENT1CON2 = ( (mTickTime * mFclk) - 1);         // TICKTIME
    SENT1CON3 = ( mFrameTime / mTickTime) ;         // FRAMETIME
    _SENT1IE = 1; // Enable SENT1 interrupt
    _SENT1IP = 4; // SENT interrupt priority
    SENT1DATbits.STAT = 0; // Status Nibble
    SENT1DATbits.DATA1 = 1; // Data Nibble 1
    SENT1DATbits.DATA2 = 2; // Data Nibble 2
    SENT1DATbits.DATA3 = 3; // Data Nibble 3
    SENT1DATbits.DATA4 = 4; // Data Nibble 4
    SENT1DATbits.DATA5 = 5; // Data Nibble 5
    SENT1DATbits.DATA6 = 6; // Data Nibble 6
    SENT1CON1bits.ON = 1; // Enable SENT module
    SENT1STATbits.SYNCTXEN = 1;                    // Initiate Synchronous Transmission
    while(1);
    return 0;
}
void __attribute__((__interrupt__,__auto_psv__)) _SENT1Interrupt (void)
{
_SENT1IF = 0;                                      // Clear interrupt flag
                                                   // Update SENT1DAT here
}
```

25.4.1.7.1. Short PWM Code (SPC) Support

Short PWM Code can be implemented with user software using Synchronous mode. SPC allows bidirectional communication, such as allowing the receiver to request a message frame from the transmitter, change modes or to calibrate a sensor.

The SPC pulse is an active-low pulse initiated by the receiver. Since the module does not provide hardware functionality for the detection of SPC pulses, the application will need to detect the SPC pulses on the SENT data pin.

When the transmitter is Idle, user software can use one of these methods to detect an SPC pulse on the SENT data pin:

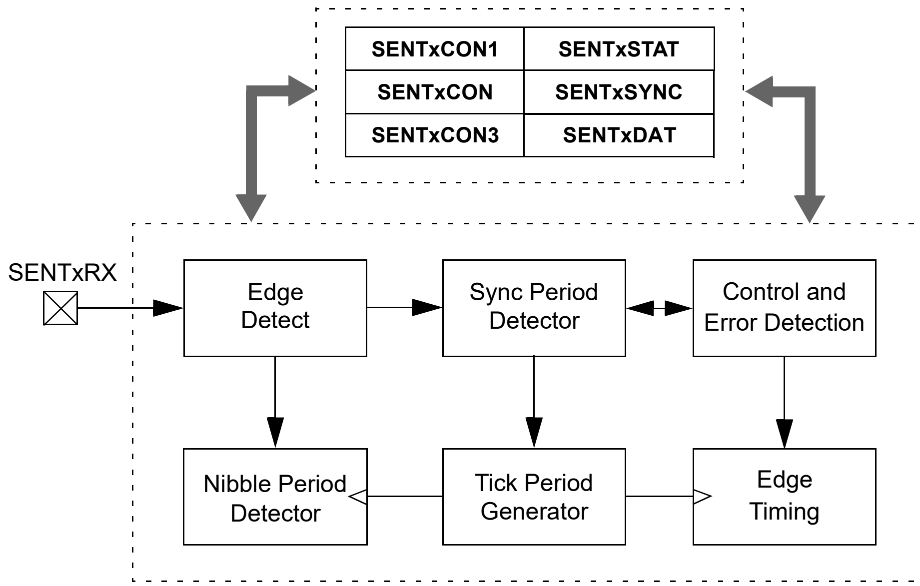
- Poll the data pin using the PORTx register associated with the pin.
- Enable an input capture peripheral that is multiplexed on the SENT data pin.
- Enable a Change Notification (CN) input or a comparator that is multiplexed on the SENT data pin.

After an SPC pulse is found, the application disables any peripherals associated with SPC pulse detection, then initiates a SENT transmission by setting the SYNCTXEN bit.

25.4.2. Receive Mode

The module can be operated as a receiver when RCVEN (SENTxCON1[11]) = `1`. If the serial data are valid, they are decoded by the module and made available in the SENTxDAT register. Error checking and status information are made available in the SENTxSTAT and SENTxSYNC registers. The captured sync period value is readable in the SENTxSYNC register. A block diagram of the module in Receive mode is shown in Figure 25-7.

Figure 25-7. SENTx Receive Mode Block Diagram



25.4.2.1. Receive Mode Timing Calculations

When configured for Receive mode, the SENTxCON2 and SENTxCON3 registers are used to hold the maximum (SYNCMAX) and minimum (SYNCMIN) boundary values of the sync period for validation of the sync pulse. SYNCMIN and SYNCMAX are ±20% of the nominal sync period. Received sync periods outside this window will be rejected by the receiver.

The equation for SYNCMIN and SYNCMAX is shown in Equation 25-3. Like the Transmit modes, the value for F_{RCV} is either F_{SENT} or $F_{SENT}/4$, depending on the setting of the PS bit.

Equation 25-3. Calculating SYNCMIN and SYNCMAX

$$SyncCount = 8 \bullet F_{RCV} \bullet T_{TICK}$$
$$SYNCMIN = 0.8 \bullet SyncCount$$
$$SYNCMAX = 1.2 \bullet SyncCount$$

Where:

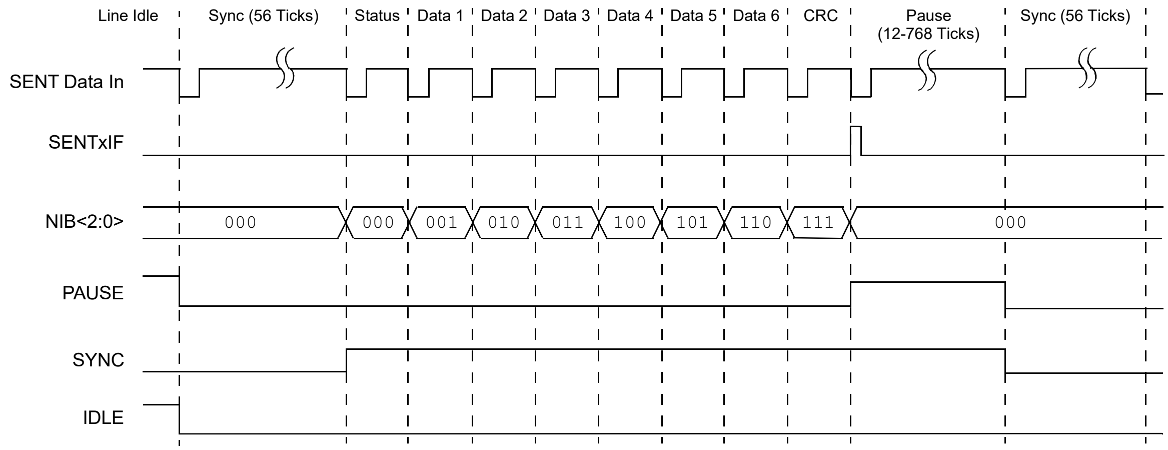
$$F_{RCV} = F_{SENT}/Prescaler$$

Note: The SENT protocol allows no more than 1.5625% (1 part in 64) timing variation between successive messages. To verify this condition is met, save the value captured in SENTxSYNC after each data frame is received, then compare it to the value received during the next data frame.

25.4.2.2. Receiver Status

When the module is configured as a receiver (RCVEN = 0), the status of the received message (Nibble status, Line state, Sync and Pause states) is stored in the SENTxSTAT register. Figure 25-8 shows the relationship between the SENT data in the signal and the SENTxSTAT status bits.

Figure 25-8. SENTx Data Reception with Pause Period



Note: The receiver is in the Idle state when the SENT data line has been high for more than the maximum time allowed for a sync period (SYNCMAX bits register value).

25.4.2.3. Receive Input Pin and PPS

The SENT Receive (SENTxRX) pin function is an independently remappable feature. To use the module in Receive mode, SENTxRX must be mapped to an available I/O pin using the appropriate RPINRx or RPORx register. If Short PWM Code support is required, a bidirectional mappable pin must be used in order to support the pulse output.

25.4.2.4. Receive Setup Procedure

To fully configure the module, the following must be known:

- T_{TICK} (Equation 25-1)
- Number of data nibbles
- Hardware or application calculated CRC validation
- Pause period present

Note: Application software can be used to implement an alternate CRC algorithm. In these instances, disable hardware CRC checking by clearing CRCEN (SENTxCON1[8]). The received CRC value (SENTxDATA[3:0]) can be read and compared against the application calculated CRC value.

To initialize the module for Receive mode:

1. Set RCVEN (SENTxCON1[11]) for Receive mode.
2. Write the desired data frame length to NIBCNT[2:0] (SENTxCON1[2:0]).
3. Set or clear CRCEN (SENTxCON1[8]) to configure the module for a hardware or software CRC calculation.
4. If the pause period is present, set PPP (SENTxCON1[7]).
5. Write the value of SYNCMAX (nominal Sync period + 20%) to SENTxCON2.
6. Write the value of SYNCMIN (nominal Sync period – 20%) to SENTxCON3.
7. Enable the SENT interrupts and set the interrupt priority.
8. Set the ON bit (SENTxCON1[15]) to enable the module.

Incoming data are read from the SENTxDATA register after the completion of the CRC and before the next message frame's status nibble. The recommended method is to use the message frame completion interrupt trigger.

Example 25-4. SENT1 Reception Code

```
#include<xc.h>
#define mFclk (4E+6)
#define mTickTime (70E-6)
#define mFrameTime (25E-3)
#define mSyncCount (8 * mFclk * mTickTime)
#define mSyncMin (0.8 * mSyncCount)
#define mSyncMax (1.2 * mSyncCount)

    uint8_t ReceivedData[6];
    uint8_t i;
    int main(void) {
    _SENT1R = 27; // RP27 as SENT1 RX pin
    SENT1CON1bits.RCVEN = 1;                      // Module operates as a receiver
    SENT1CON1bits.NIBCNT = 6;                     // 6 data nibbles per data
packet
    SENT1CON1bits.CRCEN = 1;                      // CRC is calculated using the
J2716 method
    SENT1CON1bits.PPP = 1;                        // SENTx messages transmitted
with Pause Pulse
    SENT1CON1bits.PS = 0;                         // Module clock is FSENT
    SENT1CON2 = mSyncMax;                         // TICKTIME
    SENT1CON3 = mSyncMin;                         // FRAMETIME
    _SENT1IE = 1;                                 // Enable SENT1 interrupt
    _SENT1IP = 4;                                 // set SENT interrupt priority
    SENT1CON1bits.ON = 1;                         // Enable SENT module
    while(1);
    return 0;
}
void __attribute__((__interrupt__,__auto_psv__)) _SENT1Interrupt (void)
{
    _SENT1IF = 0; // Clear interrupt flag
    i=0;
    ReceivedData[i++] = SENT1DATbits.DATA1;      // Read Data Nibble 1
    ReceivedData[i++] = SENT1DATbits.DATA2;      // Read Data Nibble 2
    ReceivedData[i++] = SENT1DATbits.DATA3;      // Read Data Nibble 3
    ReceivedData[i++] = SENT1DATbits.DATA4;      // Read Data Nibble 4
    ReceivedData[i++] = SENT1DATbits.DATA5;      // Read Data Nibble 5
    ReceivedData[i++] = SENT1DATbits.DATA6;      // Read Data Nibble 6
}
```

25.4.2.5. Error Handling

The module has the capability to automatically detect and flag framing errors and a CRC mismatch. A framing error is the detection of a status or data nibble period, less than 12 ticks or greater than 27 ticks. If a framing error is detected, the FRMERR bit (SENTxSTAT[2]) is set and a receive error interrupt is generated. After a framing error has been detected, the module clears SYNCTXEN (SENTxSTAT[0]) and begins looking for another valid sync period. The FRMERR bit remains set until another valid sync period has been detected and SYNCTXEN = 1. The application may optionally clear the FRMERR bit.

Note: If the PPP bit is set, the module will not generate a framing error on successive valid sync pulses. This is due to the possibility that a pause pulse could be interpreted as a valid sync, and the following actual sync pulse could be interpreted as a frame error. Framing errors for nibble data will still be detected when PPP = 1.

If CRC verification fails, the CRCERR bit (SENTxSTAT[3]) is set and a receive error interrupt is generated. The CRCERR bit remains set until a valid sync period for a new message has been received. Software may optionally clear the CRCERR bit. The CRCEN bit (SENTxCON1[8]) must be set to receive an interrupt on a CRC error.

25.4.2.6. Short PWM Code (SPC) Support

The SENT module provides support for implementing SPC with assistance from other external peripherals. The SPCEN (SENTxCON1[6]) bit enables an external Output Compare (OC) peripheral to control the SENT data input pin. In general, a specific OC module is linked in hardware to a specific SENTx module.

To initialize the SENT module for SPC operation:

1. Set the SPCEN (SENTxCON1[6]) bit to enable control of the SENT data pin by an external source.
2. For devices with Peripheral Pin Select, map the SENTxTX function to the same I/O pin as SENTxRX.
3. Configure the Output Compare module as follows:
   a. Configure the module for Triggered mode.
   b. Configure for a single-shot, active-high pulse.
   c. Set the Period and Duty Cycle registers for the desired pulse duration.

After configuration, to use the SPC pulse trigger:

1. Verify that the line is in a High-Impedance state by polling the RXIDLE bit (SENTxSTAT[1]).
2. Set the Trigger bit of the OC module to trigger the SPC pulse.

During the active period of the SPC pulse, the SENT receiver edge detection is disabled and the SENT data input pin is driven low by the module. At this time, the receiver logic is reset to prepare for a new data frame. When the pulse is completed, the module releases control of the SENT data input pin and input edge detection is re-enabled, so a data frame can be received from the sensor.

Note: To implement the SPC protocol, the SENT transmitting device(s) must leave the data bus in a High-Impedance state after the falling edge that completes the CRC nibble period. At this time, the SENT data line will be pulled high by the external pull-up resistor. The data bus should not be driven by any transmitter devices until the receiver device requests data by placing a low pulse on the SENT data line.

It is also possible to manually control the SENT data pin to implement the SPC protocol. This can be done as follows:

1. Clear the ON bit to disable receiver operation.
2. Manipulate the PORTx and TRISx registers associated with the SENT data pin to drive the data pin low for the desired time.
3. Return the SENT data pin to a High-Impedance condition using the TRISx register.
4. Set the ON bit to resume receiver operation.

25.5. Application Examples

Example 25-5. Short PWM Code (SPC) Support

```
#include <xc.h>
#define FCY (8E+06)
#include <libpic30.h>

#define mFclk (4E+6)
#define mTickTime (50E-6)
#define mFrameTime (25E-3)
#define mSPCPulseWidth (56 * mTickTime)
void InputCapture_configure(void);
void SENT_Tx_configure(void);
uint8_t count = 0;
uint16_t falling_edge_capture = 0, rising_edge_capture = 0,delta,firstRead;
float pulseWidth;

int main(void) {
    SENT_Tx_configure();              // Configure SENT for Transmission
    InputCapture_configure();         // Configure Input Capture
    while (1);
    return 0;
}

void InputCapture_configure(void) {
    _ICM1R = 27;                         // RP27 as Input Capture 1
    CCP1CON1bits.CCSEL = 1;              // Input capture mode
    CCP1CON1bits.CLKSEL = 0;             // Set the clock source (FPB/2)
    CCP1CON1bits.T32 = 0;                // 16-bit Dual Timer mode
    CCP1CON1bits.MOD = 3;                // Capture every edge of the event
    CCP1CON2bits.ICS = 0;                // Capture rising edge on the Pin
```

```
    CCP1CON1bits.OPS = 0;                // Interrupt on every input capture event
    CCP1CON1bits.TMRPS = 0;              // Set the clock pre-scaler (1:1)
    CCP1CON1bits.ON = 1;                 // Enable CCP/input capture
    CCP1TMR = 0;
    __delay_ms(10);
    _CCP1IF = 0;                         // Clear CCP interrupt flag
    _CCP1IE = 1;                         // Enable CCP interrupt

}

void SENT_Tx_configure(void) {
    _RP27R = 57;                 // RP27 as SENT1 output
    SENT1CON1bits.RCVEN = 0;             // Module operates as a transmitter
    SENT1CON1bits.TXM = 1;               // Synchronous Transmit
    SENT1CON1bits.NIBCNT = 6;            // 6 data nibbles per data packet
    SENT1CON1bits.CRCEN = 1;             // CRC is calculated using the J2716 method
    SENT1CON1bits.PPP = 1;               // SENTx messages transmitted with Pause Pulse
    SENT1CON1bits.PS = 0;                // Module clock is FSENT
    SENT1CON2 = ((mTickTime * mFclk) - 1);    // TICKTIME
    SENT1CON3 = (mFrameTime / mTickTime);     // FRAMETIME
    _SENT1IE = 1; // Enable SENT1 interrupt
    SENT1CON1bits.ON = 1;                     // Enable SENT module
    SENT1DATbits.STAT = 0;                    // Status Nibble
    SENT1DATbits.DATA1 = 1;                   // Data Nibble 1
    SENT1DATbits.DATA2 = 2;                   // Data Nibble 2
    SENT1DATbits.DATA3 = 3;                   // Data Nibble 3
    SENT1DATbits.DATA4 = 4;                   // Data Nibble 4
    SENT1DATbits.DATA5 = 5;                   // Data Nibble 5
    SENT1DATbits.DATA6 = 6;                   // Data Nibble 6
}

void __attribute__((__interrupt__, __no_auto_psv__)) _CCP1Interrupt(void) {

    _CCP1IF = 0;                              // Clear interrupt flag
    count++;
    if (count == 1)                           // Falling edge of the SPC pulse
    {
        firstRead = CCP1BUF;
        falling_edge_capture = CCP1BUF;
    } else if (count == 2)                    // Raising edge of the SPC pulse
    {
        count = 0; // Clear count
        rising_edge_capture = CCP1BUF;
        if (rising_edge_capture <= falling_edge_capture) {
            delta = (0xFFFF - falling_edge_capture) + rising_edge_capture;  // Rollover
case
        } else {
            delta = rising_edge_capture - falling_edge_capture;             // Non-
rollover case
        }
        pulseWidth = delta / mFclk;           // Calculate the pulse width

        // Check if pulse width is within the range
        if ((mSPCPulseWidth * 1.2) >= pulseWidth && (mSPCPulseWidth * 0.8) <= pulseWidth) {
            _CCP1IE = 0;                         // Disable CCP interrupt
            CCP1CON1bits.ON = 0;                 // Disable CCP/input capture
            SENT1STATbits.SYNCTXEN = 1;          // Initiate Synchronous Transmission
        }
    }
}

void __attribute__((__interrupt__, __no_auto_psv__)) _SENT1Interrupt(void) {
    _SENT1IF = 0;                                // Clear SENT1 interrupt flag
    while (SENT1STATbits.PAUSE == 1);            // Wait till PAUSE pulse is transmitted
    CCP1CON1bits.ON = 1;                         // Enable CCP/input capture
    CCP1TMR = 0;
    __delay_ms(1);
    _CCP1IF = 0;                                 // Clear CCP interrupt flag
    _CCP1IE = 1;                                 // Enable CCP interrupt
}
```

Example 25-6. SENT Reception (SPC Pulse Transmission)

```
#include <xc.h>
#define FCY (8E+6)
#include<libpic30.h>

#define mFclk (4E+6)
#define mTickTime (50E-6)
#define mFrameTime (25E-3)
#define mSyncCount (8 * mFclk * mTickTime)
#define mSyncMin (0.8 * mSyncCount)
#define mSyncMax (1.2 * mSyncCount)
void OutputCompare_configure();
```

```
void SENT_RX_configure();
void SendSPCPulse();
uint8_t mReceivedData[7];
uint8_t i;

int main(void) {
    ANSELB = 0;
    _TRISC6 = 1;                             // Connect a Pull-Up switch
to RC15
    OutputCompare_configure();               // Configure Output Compare
    SENT_RX_configure();                     // Configure SENT for
Reception
    while(_RC6 == 0);                        // Wait till pull-up switch
is pressed
    SendSPCPulse();                          // Send SPC pulse
    while(1)
    {
        if(SENT1STATbits.RXIDLE == 1)        // Check if the line is Idle
        {
            //Receiver can request for SENT data through SendSPCPulse()
function
            SendSPCPulse();
        }
    }
    return 0;
}
void OutputCompare_configure(){
    _CCP3IF = 0;                             // Clear CCP interrupt flag
    _CCP3IE = 1;                             // Enable CCP interrupt
    // Set MCCP operating mode
    CCP3CON1bits.CCSEL = 0;                  // Set MCCP operating mode
(OC mode)
    CCP3CON1bits.MOD = 0b100;                // Set mode
    //Configure MCCP Timebase
    CCP3CON1bits.T32 = 0;                    // Set timebase width (16-bit)
    CCP3CON1bits.TMRSYNC = 0;                // Set timebase
synchronization
    CCP3CON1bits.CLKSEL = 0b000;             // Set the clock source
(FPB/2)
    CCP3CON1bits.TMRPS = 0b00;               // Set the clock pre-scaler
(1:1)
    CCP3CON1bits.ONESHOT = 1;
    CCP3CON1bits.TRIGEN = 1;                 // Set Sync/Triggered mode
(Synchronous)
    CCP3CON1bits.SYNC = 0b1001;              // Select Sync/Trigger source
    //Configure MCCP output for PWM signal
    CCP3CON2bits.OCAEN = 1;                  // Enable desired output
signals (OC1A)
    CCP3CON3bits.POLACE = 1;                 // Configure output polarity
(Active High)
    CCP3CON3bits.OSCNT = 0;
    CCP3TMR = 0x0000;                        // Initialize timer prior to
enable module.
    CCP3RA = (56 * mTickTime * mFclk);       // Set the rising edge
compare value
    CCP3RB = (56 * mTickTime * mFclk)*2;     // Set the falling edge
compare value
    CCP3PR = CCP3RB;                         // Configure timebase period
    CCP3CON1bits.ON = 1;                     // Enable MCCP module
}
void SENT_RX_configure()
{
    _RP26R = 57;                             // RP26 as SENT1 output
    _SENT1R = 26;                            // RP26 as SENT1 input
    SENT1CON1bits.RCVEN = 1;                 // Module operates as a
receiver
    SENT1CON1bits.NIBCNT = 6;                // 6 data nibbles per data
packet
    SENT1CON1bits.CRCEN = 1;                 // CRC is calculated using
the J2716 method
    SENT1CON1bits.PPP = 1;                   // SENTx messages transmitted
with Pause Pulse
    SENT1CON1bits.PS = 0;                    // Module clock is FSENT
    SENT1CON1bits.SPCEN = 1;                 // Enable SPC
    SENT1CON2 = mSyncMax;                    // TICKTIME
    SENT1CON3 = mSyncMin;                    // FRAMETIME
    _SENT1IE = 1;                            // Enable SENT1 interrupt
    SENT1CON1bits.ON = 1;                    // Enable SENT module
```

```
}
void SendSPCPulse()
{
    CCP3STATbits.TRSET = 1; // Set the Trigger
    while(CCP3STATbits.CCPTRIG);
}

void __attribute__((__interrupt__,__no_auto_psv__)) _CCP3Interrupt (void)
{
    _CCP3IF = 0;                                    // Clear interrupt flag
    CCP3STATbits.TRSET = 0;                         // Set the Trigger
    CCP3CON1bits.ON = 0;                            // Disable MCCP module
}
void __attribute__((__interrupt__,__no_auto_psv__)) _SENT1Interrupt (void)
{
    _SENT1IF = 0; // Clear interrupt flag
    i=0;
    mReceivedData[i++] = SENT1DATbits.STAT;      // Read STAT
    mReceivedData[i++] = SENT1DATbits.DATA1;     // Read DATA1
    mReceivedData[i++] = SENT1DATbits.DATA2;     // Read DATA2
    mReceivedData[i++] = SENT1DATbits.DATA3;     // Read DATA3
    mReceivedData[i++] = SENT1DATbits.DATA4;     // Read DATA4
    mReceivedData[i++] = SENT1DATbits.DATA5;     // Read DATA5
    mReceivedData[i++] = SENT1DATbits.DATA6;     // Read DATA6
    while(SENT1STATbits.PAUSE == 1);             // Wait till PAUSE pulse is
received
    CCP3CON1bits.ON = 1;                         // Enable MCCP module
}
```

25.6. Interrupts

Each SENT module has two interrupts associated with its operation: the SENTx Transmit/Receive Interrupt Flag (SENTxIF) and the SENTx Error Interrupt Flag (SENTxEIF). Setting the corresponding SENTx Interrupt Enable bits (SENTxIE and SENTxEIE) allows the module to generate device-level interrupts.

The transmit/receive interrupt is generated after the transmission of a message frame in Transmit mode or the successful reception of a message frame in Receive mode.

The error interrupt is generated after a frame error or a CRC error in Receive mode. There are no error interrupts generated in Transmit mode.

25.7. Power-Saving Modes

25.7.1. Sleep Mode

The SENT module does not support an operation when the device is in Sleep mode. If the application requires the device to enter Sleep mode, the module should be halted by clearing the ON bit (SENTxCON1[15]). If operated as a transmitter, the application can wait until the module is done transmitting a message frame before entering Sleep.

25.7.2. Idle Mode

The SENT module provides two options of operation when the device enters Idle mode. If SIDL (SENTxCON1[13]) = `1`, module operation stops when the device enters Idle mode. The same system considerations described in Sleep mode apply.

If SIDL = `0`, the module will continue the transmission or reception process after the device enters Idle mode. When operating in Transmitter mode, the module will continue to send messages with the data contained in SENTxDAT unless the device wakes from Idle to write new data. If the module is operating in Receive mode, any old data in the SENTxDAT register will be lost unless the device wakes from Idle to read the data.

25.8. Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the SENT module to be turned off, and any message frames in progress to be aborted. All SENT pins that are multiplexed with analog inputs are configured as analog inputs. The corresponding TRISx bits are also set, effectively making all pins inputs.

26. Bidirectional Serial Synchronous (BiSS) Module

The Bidirectional Serial Synchronous (BiSS) protocol is an open source digital interface for sensors and actuators. This protocol and module are compatible with the industrial standard Serial Synchronous Interface (SSI).

The BiSS interface is used in position control applications. The interface enables a complete closed-loop position control system by providing the real-time position feedback to the host of the control motor. These applications can include:

- Bidirectional Communication in Multi Sensor Systems
- Linear and Rotary Encoders
- Motor Feedback Systems
- PLC Systems
- Drives

The Bidirectional Serial Synchronous interface consists of the following key features:

- Bidirectional Data Communication that is Serial, Synchronous and Continuous
- Four Serial Communication Lines
- Four Data Channel Support with Ability to Add More
- 64-bit Sensor Data Lengths
- Cyclic at High Speeds
- Speeds Up to 10 MHz (BiSS) or 1.5 MHz (SSI)
- Auto Sense Frequency
- Variable Clock Rate
- Line Delay Compensation
- Safety Features Such as CRC, Error Flags and Warning Flags
- Ready to Use with RS485/422 Physical Interfaces
- BiSS/BiSS-C and SSI Compatibility
- Host and Client Configurations (Point-to-Point)
- Host and Multiple Clients (Point-to-Point)
- Host and Multiple Clients on Bus (Bus Configuration or BiSS)

26.1. Device-Specific Information

Table 26-1. BiSS Summary Table

BiSS Module Instances	Peripheral Bus Speed	Clock Source
1	Standard (1:2 CPU Clock)	See Table 26-2

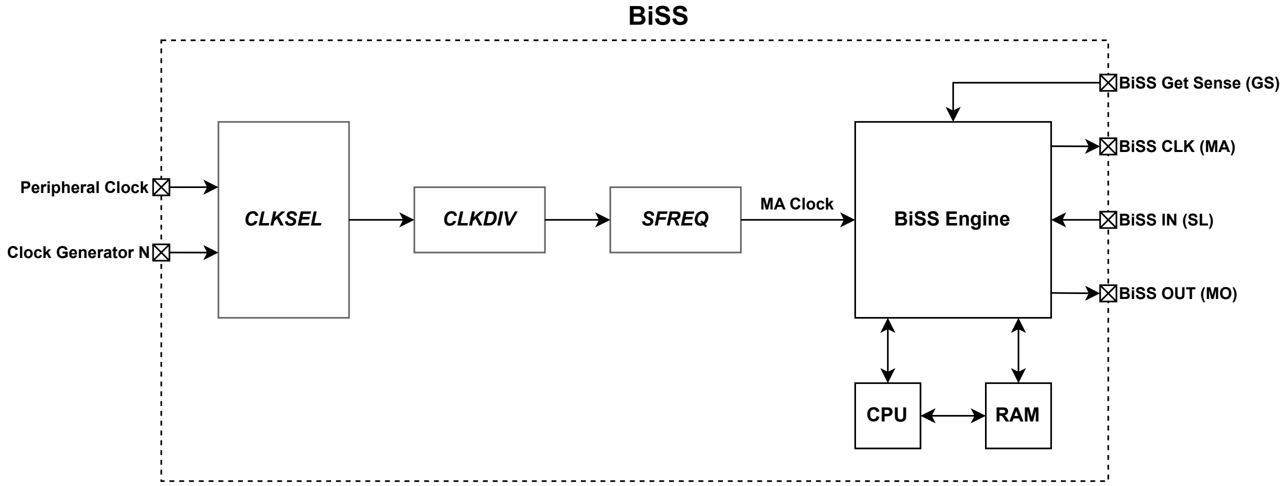
Table 26-2. CLKSEL Selection bit

Value	Description
1	CLKGEN 16
0	Standard (1:2 CPU Clock)

26.2. Architectural Overview

BiSS is a four-channel serial interface. Figure 26-1 provides a schematic of the BiSS connection between a dsPIC33A device and two BiSS sensors.

Figure 26-1. BiSS High-Level Block Diagram



Note: Refer to Terminology for signal abbreviations in BiSS module figures.

The devices in the BiSS system can operate in the following relationships:

- Point-to-Point – Single Client
- Point-to-Point – Multiple Clients
- Bus Configuration – Multiple Clients

Figure 26-2. Point-to-Point – Single Client

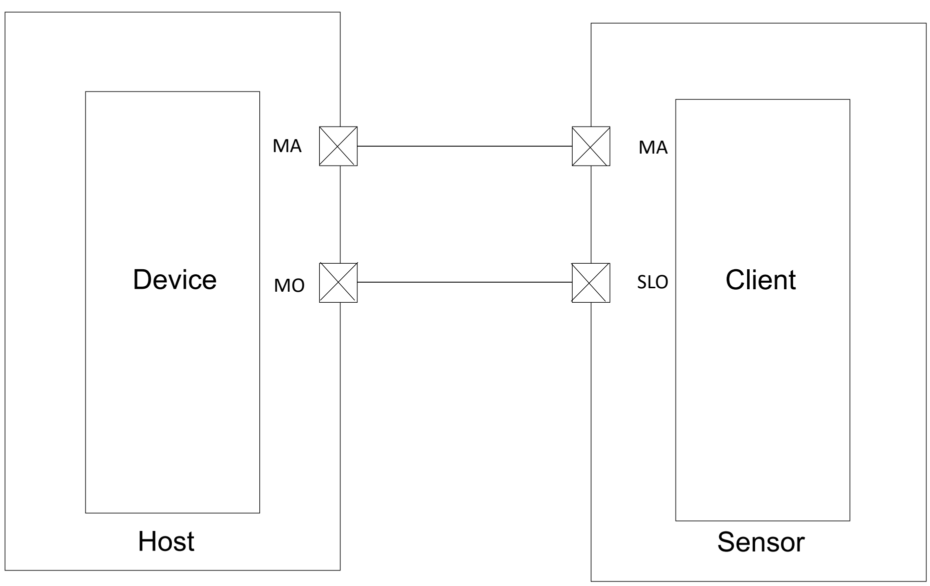


Figure 26-3. Point to Point – Multiple Clients

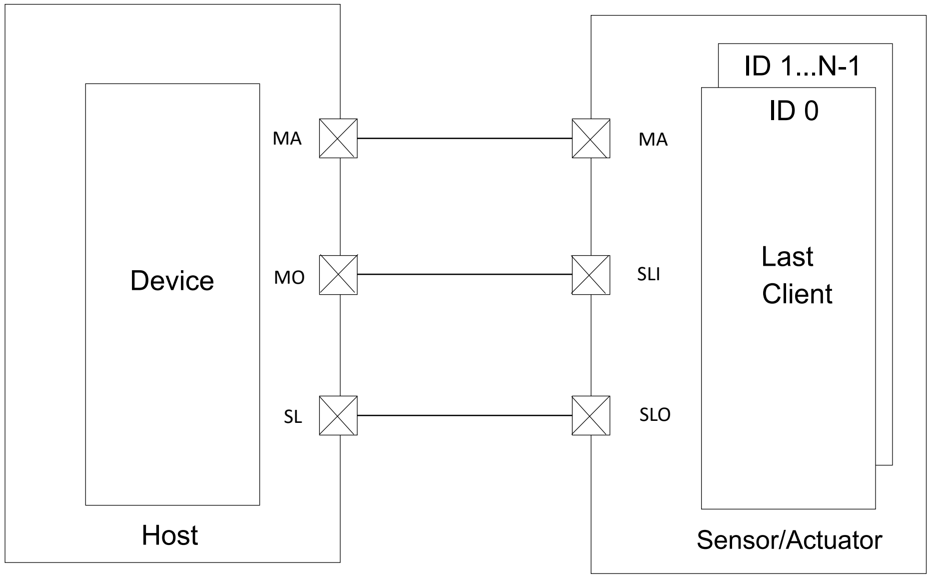
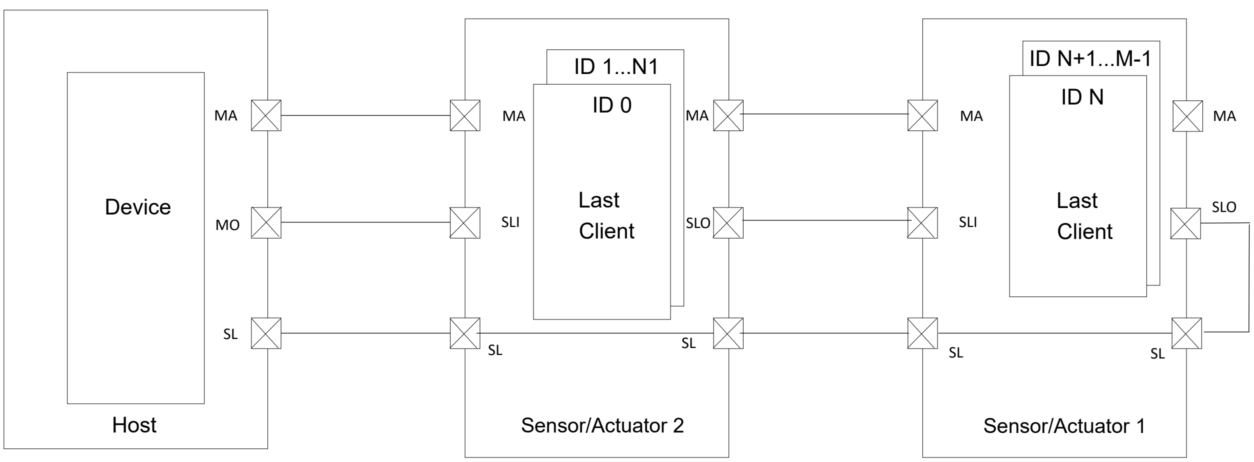


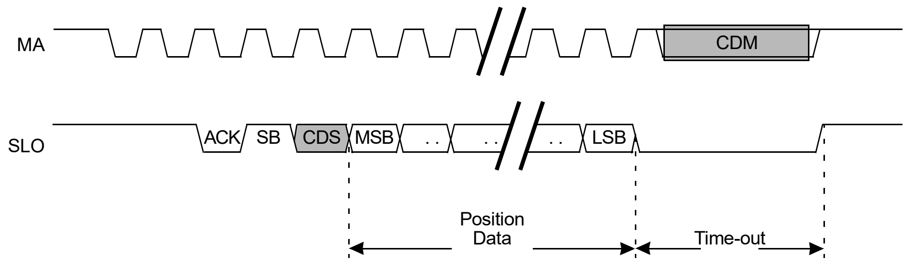
Figure 26-4. Bus Configuration – Multiple Clients



26.2.1. BiSS Frame

In the following point-to-point configuration, a simple example of a BiSS frame is shown. Figure 26-5 shows the clock generated on MA and transmitted to the client. In this case, the client is a sensor, and a packet of data is sent from the sensor or client back to the host.

Figure 26-5. Protocol Format



Note: See Bus Protocol Details for more information regarding the BiSS frame label definitions.

26.2.2. BiSS Supported Modes

The BiSS module supports BiSS/BiSS-C and SSI modes. There are a few differences between BiSS and SSI, but the main considerations are speed and whether control communication is needed. BiSS can operate at a maximum of 10 MHz, while SSI can operate at a maximum of 1.5 MHz. BiSS can communicate data every cycle and can also communicate control data or commands. SSI only communicates data each cycle. Setting the CFGCHx bits in the BiCHCON register for each channel configures the channel in the desired BiSS Operating mode.

26.2.3. BiSS Data Types

There are two major types of transmission types, or packets, that are used in BiSS. These transmission types can be separated into Single Cycle Data and Control Data. The Single Cycle Data and Control Data form the complete BiSS data packet.

26.2.3.1. Single Cycle Data (SCD)

The Single Cycle Data are the primary data and are newly generated and completely transmitted in each cycle. This data channel is primarily used for fast and cyclical sensor or actuator data. Since the data are cyclical, meaning they are based on repeated intervals of time, rather than based on specific state conditions, the data are predictable and able to be operated at high speeds. Sensor data can be considered a data transmission from a client to a host. Actuator data can be considered as a data transmission from a host to a client.

26.2.3.2. Single Cycle Data (SCD) Gray Code

Normally, BiSS uses a binary encoder and sends an updated complete binary value for each SCD. If desired, in SSI mode, a gray code encoder can be enabled. Enabling the gray code encoder allows only a single bit to be changed from the previous SCD, allowing for less errors during data transmission. The gray code encoder is enabled by setting the GRAY bit in the respective BiCLTCONx register.

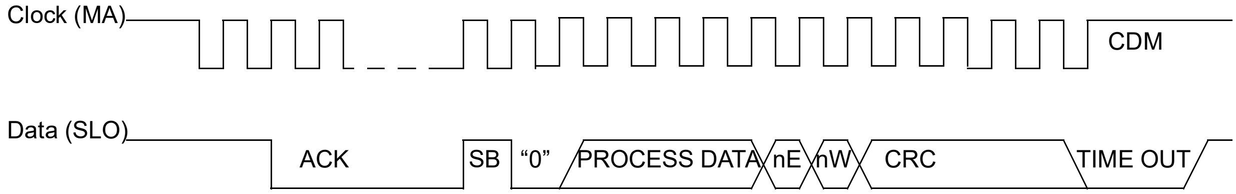
26.2.3.3. Control Data (CD)

The Control Data have two different types: Control Data Host (CDM) and Control Data Client (CDS). The Control Data Host is transmitted one bit per BiSS frame, or cycle, on the MA line during time-out on the SLO line. The Control Data Client is transmitted on the SLO line one bit per BiSS frame, or cycle, after the Start bit but before the data are transmitted. After multiple BiSS frames or cycles, an entire control frame is constructed from the individual CDM and CDS bits that were sent per cycle.

26.2.4. Bus Protocol Details

Figure 26-6 shows a simple example of the BiSS module operating in point-to-point configuration to illustrate the different components that combine to form a complete BiSS frame.

Figure 26-6. Example of BiSS Protocol



Note: Clock and data are both active high.

26.2.4.1. Acknowledge (ACK)

When a host or client device in the serial chain sends a Start bit, the client must provide an Acknowledge of the Start. If multiple clients are in a chain and connected to the host, the host sends a Start, the last client in the chain sends an Acknowledge and a Start and then the next client sends an Acknowledge and Start; this continues until the first client sends an Acknowledge and a Start.

26.2.4.2. Start bit (SB)

The Start bit signifies that a CDS bit, followed by a data transfer, is about to occur.

26.2.4.3. Control Data Client (CDS)

The CDS bit is the first bit after the Start bit, and "0" is always the state transmitted by the host. This bit is used by clients for ID confirmation and replies to host control data. This bit can be "1" or "0" when transmitted by the client. The eventual combination from each cycle's CDS bits creates an entire control communication packet. Please see Register Communication for more information.

26.2.4.4. Process Data

Process data are the data from the client's sensors/actuators that are transmitted to or from the host. Bit length is determined by the number of clients in the system. For a single client, this is equivalent to the Single Cycle Data (SCD); they are the cycle-to-cycle data that are transmitted every cycle. An example of data that are transmitted every cycle could be the encoded position of a sensor.

26.2.4.5. Error bit (nE)

An Error bit is an optional error detection bit that can be enabled. This bit is inverted and transmitted after the main data packet.

The Error bit and Warning bit (nW) allow for precautionary measures during every cycle. This allows the host and clients to respond to an error or warning during every cycle instead of waiting for any control data.

26.2.4.6. Warning bit (nW)

The Warning bit is an optional bit that can be enabled. Similar to the Error bit (nE), this bit is also inverted and transmitted after the Error bit.

26.2.4.7. Cyclic Redundancy Check (CRC)

If enabled, after the nW bit, the CRC for data can be transmitted. The CRC is used to provide additional safety measures. More details can be found in CRC.

26.2.4.8. Time-out

Once all data have been sent in addition to the optional bits, there should be a time-out on the SLO line while the host transmits the CDM bit. In the BiSS time-out, no further clock pulses are sent to the MA by the host. The Inverse state of the MA line during the BiSS time-out is the state of the CDM bit. At the end of the data transmission, the host sets its output MO to the Idle state '1'. The clients then pass on this '1' received at SLI to their output SLO as soon as they have detected the expiration of the time-out. This ensures the BiSS time-out on the line SL is only signaled to the host when all connected clients have detected the time-out. When the BiSS time-out expires, all clients return to the Idle state; all lines are set to the high signal level (‘1’) in the process.

26.2.4.9. Control Data Main (CDM)

The CDM bit for control communication is the inverted state of the MA clock line during the BiSS time-out. Each cycle-to-cycle CDM bit eventually combines together to create an entire CDM control packet; this is how the host communicates control information to clients.

26.2.5. Device Setup

The following conditions are required to start the BiSS module:

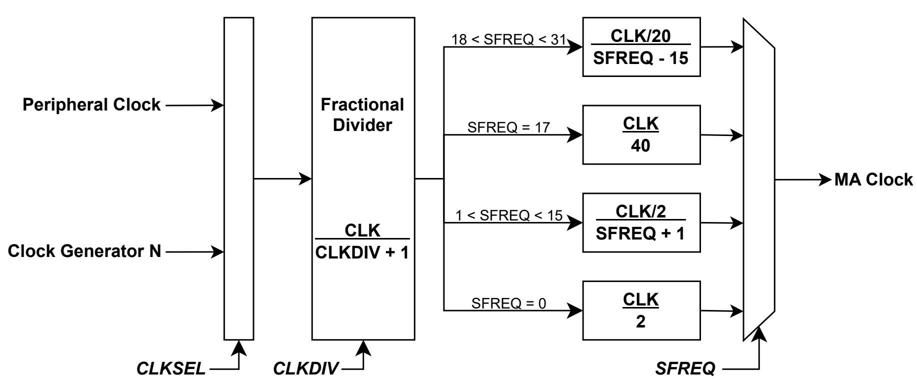
- I/O pins and PPS must be set.
- Interrupts must be taken into account.
- Communication frequencies must be specified.

Once these conditions are met, the data transmission needs to be configured.

26.2.5.1. Clocking and Baud Rate Configuration

The BiSS module supports multiple clock sources to drive the single-cycle data frequency. The clock tree with relevant registers is shown in Figure 26-7. The code is equivalent to the decimal number that the register is set.

Figure 26-7. Clock Tree



26.2.5.1.1. Clock Frequency Generation

The BiSS module supports multiple clock sources that can be selected using CLKSEL register bits. After the clock source is selected, the first clock divider can be set up by setting the CLKDIV register. The BiSS module is expected to receive a maximum of 20 MHz CLK from the CLKDIV block. Select the CLKDIV value so that the CLK value is not more than 20 MHz. Setting the SFREQ register configures the SCD frequency (MA Clock). With a module divided clock frequency of 20 MHz, the clock frequency at MA ranges from 10 MHz down to 62.5 kHz.

26.2.5.1.2. Protocol Interface Pins

MA is an output from the host and represents the host clock that will be transmitted to all clients simultaneously. MA is also the line where the control data from the host are transmitted.

MO is the host data output and is the line where the host will transmit actuator data (host to client operation).

SL or SLO (depending on SSI or BiSS mode) is the client return (SSI) or the client output (BiSS). SL/SLO is the line in which sensor data will be transmitted from the sensors to the host (client to host operation).

26.2.5.1.3. BiSS I/O Pins

Three pins are used for the bus operation: MA (Host Clock), MO (Host Output Data) and SL (Return Data).

26.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2100	B1SCDATA0L	31:24	SCDATA0L[31:24]							
		23:16	SCDATA0L[23:16]							
		15:8	SCDATA0L[15:8]							
		7:0	SCDATA0L[7:0]							
0x2104	B1SCDATA0H	31:24	SCDATA0H[31:24]							
		23:16	SCDATA0H[23:16]							
		15:8	SCDATA0H[15:8]							
		7:0	SCDATA0H[7:0]							
0x2108	B1SCDATA1L	31:24	SCDATA1L[31:24]							
		23:16	SCDATA1L[23:16]							
		15:8	SCDATA1L[15:8]							
		7:0	SCDATA1L[7:0]							
0x210C	B1SCDATA1H	31:24	SCDATA1H[31:24]							
		23:16	SCDATA1H[23:16]							
		15:8	SCDATA1H[15:8]							
		7:0	SCDATA1H[7:0]							
0x2110	B1SCDATA2L	31:24	SCDATA2L[31:24]							
		23:16	SCDATA2L[23:16]							
		15:8	SCDATA2L[15:8]							
		7:0	SCDATA2L[7:0]							
0x2114	B1SCDATA2H	31:24	SCDATA2H[31:24]							
		23:16	SCDATA2H[23:16]							
		15:8	SCDATA2H[15:8]							
		7:0	SCDATA2H[7:0]							
0x2118	B1SCDATA3L	31:24	SCDATA3L[31:24]							
		23:16	SCDATA3L[23:16]							
		15:8	SCDATA3L[15:8]							
		7:0	SCDATA3L[7:0]							
0x211C	B1SCDATA3H	31:24	SCDATA3H[31:24]							
		23:16	SCDATA3H[23:16]							
		15:8	SCDATA3H[15:8]							
		7:0	SCDATA3H[7:0]							
0x2120 ... 0x217F	Reserved									
0x2180	B1RDATA0	31:24	RDATAy [23:0]							
		23:16	RDATAy [23:0]							
		15:8	RDATAy [23:0]							
		7:0	RDATAy/IDS0 [7:0]							
0x2184	B1RDATA1	31:24	RDATAy [23:0]							
		23:16	RDATAy [23:0]							
		15:8	RDATAy [23:0]							
		7:0	RDATAy/IDS1 [7:0]							
0x2188	B1RDATA2	31:24	RDATAy [23:0]							
		23:16	RDATAy [23:0]							
		15:8	RDATAy [23:0]							
		7:0	RDATAy/IDS2 [7:0]							
0x218C	B1RDATA3	31:24	RDATAy [23:0]							
		23:16	RDATAy [23:0]							
		15:8	RDATAy [23:0]							
		7:0	RDATAy/IDS3 [7:0]							
0x2190	B1RDATA4	31:24	RDATAy [23:0]							
		23:16	RDATAy [23:0]							
		15:8	RDATAy [23:0]							
		7:0	RDATAy/IDS4 [7:0]							
0x2194	B1RDATA5	31:24	RDATAy [23:0]							
		23:16	RDATAy [23:0]							
		15:8	RDATAy [23:0]							
		7:0	RDATAy/IDS5 [7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x2198	B1RDATA6	31:24	RDATAy [23:0]							
		23:16	RDATAy [23:0]							
		15:8	RDATAy [23:0]							
		7:0	RDATAy/IDS6 [7:0]							
0x219C	B1RDATA7	31:24	RDATAy [23:0]							
		23:16	RDATAy [23:0]							
		15:8	RDATAy [23:0]							
		7:0	RDATAy/IDS7 [7:0]							
0x21A0 ... 0x21BF	Reserved									
0x21C0	B1CLTCON0	31:24	CRCSEED0[15:8]							
		23:16	CRCSEED0[7:0]							
		15:8	CRCSEL0	CRCLEN0[6:0]CRCPOLY0[6:0]						
		7:0	GRAY0LSTOP 0	SCDEN0	SCDLEN0[5:0]					
0x21C4	B1CLTCON1	31:24	CRCSEED1[15:8]							
		23:16	CRCSEED1[7:0]							
		15:8	CRCSEL1	CRCLEN1[6:0]CRCPOLY1[6:0]						
		7:0	GRAY1LSTOP 1	SCDEN1	SCDLEN1[5:0]					
0x21C8	B1CLTCON2	31:24	CRCSEED2[15:8]							
		23:16	CRCSEED2[7:0]							
		15:8	CRCSEL2	CRCLEN2[6:0]CRCPOLY2[6:0]						
		7:0	GRAY2LSTOP 2	SCDEN2	SCDLEN2[5:0]					
0x21CC	B1CLTCON3	31:24	CRCSEED3[15:8]							
		23:16	CRCSEED3[7:0]							
		15:8	CRCSEL3	CRCLEN3[6:0]CRCPOLY3[6:0]						
		7:0	GRAY3LSTOP 3	SCDEN3	SCDLEN3[5:0]					
0x21D0 ... 0x21DF	Reserved									
0x21E0	B1RCCON	31:24			REGNUM[5:0]					
		23:16	WNR	STRTADDR[6:0]						
		15:8								
		7:0								
0x21E4	B1CTRLCON	31:24							NOCRC	BANKSWEN
		23:16	Reserved[2:0]			SFREQ[4:0]				
		15:8	CTS	PROTOSEL	CMDCLNTID21[1:0]		IDADISCLNTID0		MOEN	HOLDCDM
		7:0	CHSEL[7:0]							
0x21E8	B1CCON	31:24								
		23:16								
		15:8	MODELAY[7:0]							
		7:0	FREQAGS[7:0]							
0x21EC	B1CHCON	31:24	ACTSEN[7:0]							
		23:16								
		15:8							CLCHCFG0[1:0]	
		7:0	CLNTLOC[7:1]							
0x21F0	B1STAT	31:24	CDMTO	CDSSEL	REGBYTESV[5:0]					
		23:16								
		15:8	CLSCDV3		CLSCDV2		CLSCDV1		CLSCDV0	
		7:0	ERR	AGSERR	DLYERR	SCDTXERR	REGERR	REGEND		EOT
0x21F4	B1INSTR	31:24								
		23:16								
		15:8								
		7:0	BREAK	BNKLOCK	SWBANK	INIT	INSTR[2:0]			AGS

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x21F8	B1CHSTAT	31:24								BKSWERR
		23:16								
		15:8								
		7:0							CDS0	SL0
0x21FC	B1CON	31:24	INSTRWA	INSTRWE	REGAE			TXRDEN	SCDRST	REGRST
		23:16	CLKDIV[7:0]							
		15:8	ON		SLPEN	SIDL	CDM	CDS	SENSESEL	GETSENSE
		7:0				REGACC	BNKNUM	ACTIVE		CLKSEL

26.3.1. BiSS Single Cycle Data Register[1]

Name: B1SCDATAxL
Offset: 0x2100, 0x2108, 0x2110, 0x2118

Notes:
1. Number of registers are implemented based on the number of clients supported in a product.
2. x = 1 to BSN (BiSS Client Number), each client has maximum of 64-bits.

Bit	31	30	29	28	27	26	25	24
	SCDATAxL[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SCDATAxL[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SCDATAxL[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SCDATAxL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SCDATAxL[31:0] Single Cycle Data for Sensor/Actuator Data bits[2]

26.3.2. BiSS Single Cycle Data Register

Name: B1SCDATAxH
Offset: 0x2104, 0x210C, 0x2114, 0x211C

Note:
1. x = 1 to BSN (BiSS Client Number), each client has a maximum of 64-bits.

Bit	31	30	29	28	27	26	25	24
	SCDATAxH[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	SCDATAxH[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SCDATAxH[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	SCDATAxH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – SCDATAxH[31:0] Single Cycle Data for Sensor/Actuator Data bits(1)

26.3.3. BiSS 1 Register Data Register

Name: B1RDATAx
Offset: 0x2180, 0x2184, 0x2188, 0x218C, 0x2190, 0x2194, 0x2198, 0x219C

Notes:
1. x = BiSS Register Number or IDS number, 0-7
2. y = BiSS Register Data Number, 0-31
3. Bits 0-7 reside in the same bit space but have different uses based on the operation mode. For example, RDATA is used in control communication, and IDS is used in command communication.

Bit	31	30	29	28	27	26	25	24
	RDATAy [23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	RDATAy [23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	RDATAy [23:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RDATAy/IDSx [7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:8 – RDATAy [23:0] Register Data(1)

Bits 7:0 – RDATAy/IDSx [7:0]
Register Data for control communication/Client Addressing ID for command communication(3) Bits 7-0 are shared by RDATAy and IDSx

26.3.4. BiSS Client Configuration Register

Name: B1CLTCONx
Offset: 0x21C0, 0x21C4, 0x21C8, 0x21CC

Note:
1. x = Client Number, 0 to 3

Bit	31	30	29	28	27	26	25	24
	CRCSEEDx[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCSEEDx[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCSELx	CRCLENx[6:0]CRCPOLYx[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	GRAYxLSTOPx	SCDENx	SCDLENx[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:16 – CRCSEEDx[15:0] Polynomial SCD CRC Calculation Start Value bits

Bit 15 – CRCSELx Select CRC bit

Value	Description
1	CRC polynomial(7:1) in CRCPOLYx
0	CRC bit length in CRCLENx will apply dedicated CRC polynomials.

Bits 14:8 – CRCLENx[6:0]CRCPOLYx[6:0] Polynomial Selection by Length for SCD CRC Check bit/Polynomial for SCD CRC Check bits

CRCLENx and CRCPOLYx share the same register location.

Value	Description
	When CRCLENx is selected:
0010000	CRC polynomial 0b1.1001.0000.1101.1001 = 0x190D9
0001000	CRC polynomial 0b1.0010.1111 = 0x12F
0000111	CRC polynomial 0b1000.1001 = 0x89
0000110	CRC polynomial 0b100.0011 = 0x43
0000101	CRC polynomial 0b10.0101 = 0x25
0000100	CRC polynomial 0b1.0011 = 0x13
0000011	CRC polynomial 0b1011 = 0xB
0000000	CRC for single cycle data not present, CRC verification deactivated, CRCSELx = 0b0.
	When CRCPOLYx is selected:
11111111-0000001	CRC polynomial for single-cycle data
0000000	CRC polynomial 0x00 not applicable with CRCSELx = 0b1

Bit 7 – GRAYxLSTOPx Enable SCD Gray to Binary Conversion bit (SSI only)/Leading STOP bit on Single Cycle bit (actuator data only)

GRAYx and LSTOPx share the same register location.

Value	Description
	When GRAYx is selected
1	SSI single-cycle data gray coded.
0	SSI single-cycle data binary coded.
	When LSTOPx is selected:
1	Leading STOP Bit on Single-Cycle Actuator Data (Do not send STOP bit before Actuator data).
0	No Leading STOP Bit on Single-Cycle Actuator Data (Send STOP bit before Actuator data).

Bit 6 – SCDENx Enable Single Cycle Data bit

Value	Description
1	Single-cycle data available.
0	Single-cycle data not available.

Bits 5:0 – SCDLENx[5:0] Single Cycle Data Length bits

Value	Description
111111	Single cycle data length = 64
...	
000001	Single cycle data length = 2
000000	Single cycle data length = 1

26.3.5. BiSS Register Communication Configuration Register

Name: B1RCCON
Offset: 0x21E0

Bit	31	30	29	28	27	26	25	24
			REGNUM[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	WNR	STRTADDR[6:0]						
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 29:24 – REGNUM[5:0] Register Quantity of Access bits

Bit 23 – WNR Write/Read Register Selector bit

Value	Description
1	Write register data
0	Read register data

Bits 22:16 – STRTADDR[6:0] Register Access Start Address bits

26.3.6. BiSS Control Communication Configuration Register

Name: B1CTRLCON
Offset: 0x21E4

Note:
1. With CLK = 20 MHz, clock frequencies ranging from 62.5 kHz to 10 MHz can be selected for sensor data transmission.

Bit	31	30	29	28	27	26	25	24
							NOCRC	BANKSWEN
Access							R/W	R/W
Reset							0	0

Bit	23	22	21	20	19	18	17	16
	Reserved[2:0]			SFREQ[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CTS	PROTOSEL	CMDCLNTID21[1:0]		IDADISCLNTID0		MOEN	HOLDCDM
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit	7	6	5	4	3	2	1	0
	CHSEL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 25 – NOCRC CRC for SCD Not to be Stored in RAM bit

Value	Description
1	All client CRC of SCD is not stored in RAM.
0	CRC of SCD is stored in RAM (only applicable with active CRC verification and CRC polynomial > 0).

Bit 24 – BANKSWEN Single RAM Bank Enabled bit

Value	Description
1	One RAM bank is used for SCD.
0	Two RAM banks are used for SCD.

Bits 23:21 – Reserved[2:0]

Bits 20:16 – SFREQ[4:0] Sensor Data Clock Frequency (FSCD) bits(1)

Value	Description
11111 – 10010	CLK / 20 / (Value - 15)
10001	CLK / 40
10000	Not permitted
01111 – 00001	CLK / 2 / (Value + 1)
00000	CLK/2

Bit 15 – CTS Register Transmission or Instruction Selector bit

Value	Description
1	Register communication
0	Command/instruction communication

Bit 14 – PROTOSEL Register Access Protocol Selection A/B or C Selector bit

Value	Description
1	Register communication BiSS C
0	Reserved

Bits 13:12 – CMDCLNTID21[1:0] CMD/CLNTID21 bit 1 and bit 2 – Command/Instruction bit in which the command is determined by Client/CLNTID21 Client ID of Accessed Client bit 1 and 2 onlyCMD and CLNTID21 share the same register location.

Bit 11 – IDADISCLNTID0 ID-Acknowledge Disable bit

Value	Description
1	Immediate execution
0	The client's feedback (IDA) is tested before execution (EX bit after IDA).

Bit 9 – MOEN MO Enable - Enable Output at MOx for Actuator Data or Delayed Start bit

Value	Description
1	Parameterized processing time by host when the MO signal is active (length = MODELAY).
0	MO is forced to low.

Bit 8 – HOLDCDM Hold CDM (Control Data Host) - Length of CDM bit

Value	Description
1	Clock line consistent with the CDM bit until start of the next cycle.
0	Clock line high at the end of cycle.

Bits 7:0 – CHSEL[7:0] Channel Selection bits

Value	Description
1	Channel N is used for communication.
0	Channel N is not used for communication.

26.3.7. BiSS Communication Configuration Register

Name: B1CCON
Offset: 0x21E8

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	MODELAY[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	FREQAGS[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – MODELAY[7:0] Delay of Start Bit at Output MOx bits

Value	Description
11111111-00000001	Value * 1 /FSCD
00000000	No start delay

Bits 7:0 – FREQAGS[7:0] AutoGetSense Frequency - Controls Automatic SCD Timing bits

Value	Description
0x80-0xFF	CLK/ (625 * (Value+1)
0x7D	AGSINFINITE - Requires a trigger event to start next SCD cycle, such as GETSENSE or INSTR.
0x7C	AGSMIN - Host automatically restarts the next SCD cycle after the prior finishes.
0x00-0x7B	CLK/ (20 * (FREQAGS(6:0)+1))

26.3.8. BiSS Channel Configuration Register

Name: B1CHCON
Offset: 0x21EC

Bit	31	30	29	28	27	26	25	24
	ACTSEN[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
							CLCHCFG0[1:0]	
Access							R/W	R/W
Reset							0	0

Bit	7	6	5	4	3	2	1	0
	CLNTLOC[7:1]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bits 31:24 – ACTSEN[7:0] Sensor or Actuator Data Selector bits

Value	Description
1	Actuator data client 1...8
0	Sensor data client 1...8

Bits 9:8 – CLCHCFG0[1:0] Channel 0 Configuration bits

Value	Description
11	Channel is not used.
10	SSI
01	BiSS C
00	Reserved

Bits 7:1 – CLNTLOC[7:1] Client Location bits

Value	Description
1	Following clients are assigned to the next channel.
0	Clients are assigned to this channel.

26.3.9. BiSS Communication Status Register

Name: B1STAT
Offset: 0x21F0

Bit	31	30	29	28	27	26	25	24
	CDMTO	CDSSEL	REGBYTESV[5:0]					
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CLSCDV3		CLSCDV2		CLSCDV1		CLSCDV0	
Access	R/W		R/W		R/W		R/W	
Reset	0		0		0		0	

Bit	7	6	5	4	3	2	1	0
	ERR	AGSERR	DLYERR	SCDTXERR	REGERR	REGEND		EOT
Access	R/W	R/W	R/W	R/W	R/W	R/W		R/W
Reset	0	0	0	0	0	0		1

Bit 31 – CDMTO CDM Timeout Reached bit

Value	Description
1	CDMTIMEOUT reached.
0	CDMTIMEOUT not reached.

Bit 30 – CDSSEL Selected Channel CDS bit

Value	Description
1	CDS of the selected channel = 1
0	CDS of the selected channel = 0

Bits 29:24 – REGBYTESV[5:0] Valid Register Data Transmitted Before Error bits

Value	Description
0x01-0x3F	After transfer, the number of successfully transferred registers before a register communication error.
0x00	After transfer, no register communication error.

Bit 15 – CLSCDV3 Client 3 Single-Cycle Data Valid bit

Value	Description
1	Single-cycle data is valid.
0	Single-cycle data is invalid.

Bit 13 – CLSCDV2 Client 2 Single-Cycle Data Valid bit

Value	Description
1	Single-cycle data is valid.
0	Single-cycle data is invalid.

Bit 11 – CLSCDV1 Client 1 Single-Cycle Data Valid bit

Value	Description
1	Single-cycle data is valid.
0	Single-cycle data is invalid.

Bit 9 – CLSCDV0 Client 0 Single Cycle Data Valid bit

Value	Description
1	Single cycle data valid
0	Single cycle data invalid

Bit 7 – ERR Transmission Error bit

Value	Description
1	Error
0	No Error

Bit 6 – AGSERR AGS Error - Unable to Start SCD Frame bit

Value	Description
1	AGS error
0	No AGS error

Bit 5 – DLYERR Delay Error bit - Missing Start bit During Register Communication

Value	Description
1	Delay error
0	No Delay error

Bit 4 – SCDTXERR SCD Transmission Error bit

Value	Description
1	Error in the last single cycle data transmission.
0	No Error in the last single cycle data transmission.

Bit 3 – REGERR Register Communication Error bit

Value	Description
1	Error in the last register data transmission.
0	No Error in the last register data transmission.

Bit 2 – REGEND End Of Register Communication Error bit

Value	Description
1	Register data transmission is completed.
0	No valid register data is available.

Bit 0 – EOT End Of Transmission bit

Value	Description
1	Data transmission is finished.
0	Data transmission is active.

26.3.10. BiSS Instruction Register

Name: B1INSTR
Offset: 0x21F4

Note:
1. The SWBANK bit does not clear automatically and should be checked and cleared in software before issuing the next SWBANK instruction.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	BREAK	BNKLOCK	SWBANK	INIT		INSTR[2:0]		AGS
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – BREAK Data Transmission Interrupt bit

Value	Description
1	Abort data transmission.
0	No change

Bit 6 – BNKLOCK Inhibit RAM Bank Switching bit

Value	Description
1	Bank switching lock
0	No bank switching lock

Bit 5 – SWBANK Switch RAM Bank bit[(1)]

Value	Description
1	RAM banks are switched.
0	RAM banks are not switched.

Bit 4 – INIT Start INIT Sequence bit

Value	Description
1	Initialize data channel
0	No changes on the data channel

Bits 3:1 – INSTR[2:0] SCD Control Instruction bits

Bit 0 – AGS Automatic Get Sensor Data bit

Value	Description
1	Automatic data transmission
0	No automatic data transmission

26.3.11. BiSS Channel Status Register

Name: B1CHSTAT
Offset: 0x21F8

Bit	31	30	29	28	27	26	25	24
								BKSWERR
Access								R
Reset								0

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							CDS0	SL0
Access							R	R
Reset							0	x

Bit 24 – BKSWERR Switch Bank Failed bit

Value	Description
1	Bank switching (SCD) not successful
0	Bank switching (SCD) successful

Bit 1 – CDS0 Channel 0 CDS bit

Value	Description
1	CDS0 = 1
0	CDS0 = 0

Bit 0 – SL0 SL0 Input Line State bit

Value	Description
1	SL0 line level high
0	SL0 line level low

26.3.12. BiSS Configuration Register

Name: B1CON
Offset: 0x21FC

Notes:
1. It is not recommended to set "00" for CLKDIV bits.
2. Input to the CLKDIV depends on the CLKSEL value.
3. The clock frequency (CLK) derived from CLKDIV must be a maximum of 20 MHz.

Bit	31	30	29	28	27	26	25	24
	INSTRWA	INSTRWE	REGAE			TXRDEN	SCDRST	REGRST
Access	R	R/C	R/C			R/W	R/W	R
Reset	0	0	0			0	0	0

Bit	23	22	21	20	19	18	17	16
	CLKDIV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8
	ON		SLPEN	SIDL	CDM	CDS	SENSESEL	GETSENSE
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				REGACC	BNKNUM	ACTIVE		CLKSEL
Access				R/W	R/W	R/W		R/W
Reset				0	0	0		0

Bit 31 – INSTRWA Instruction Write Active bit

Value	Description
1	Previous write is not complete.
0	Previous write is complete.

Bit 30 – INSTRWE Instruction Write Error bit

Value	Description
1	Disallowed attempt to write to the Instruction register (must reset in software).
0	No instruction write error.

Bit 29 – REGAE Register Mode Data Access Error bit

Value	Description
1	Disallowed attempt to the access Register mode data (must reset in software).
0	No register access error.

Bit 26 – TXRDEN Transmit RAM CPU Read Enable bit

Value	Description
1	Transmit RAM is allowed to read by CPU.
0	Receive RAM is allowed to read by CPU.

Bit 25 – SCDRST SCD RAM Reset bit

Write '1' to reset the Register communication RAM data buffers.
Reading this bit returns '1' if the Register RAM is in Reset, and '0' if Reset is completed. Writing '0' to this bit must occur when this bit is '1' to come out of Reset; otherwise, writing '0' to this bit has no effect.

Bit 24 – REGRST Register RAM Reset bit

Write '1' to reset the Register communication RAM data buffers.
Reading this bit returns '1' if the Register RAM is in Reset, and '0' if Reset is completed. Writing '0' to this bit is required to come out of Reset when this bit is '1'; otherwise, writing '0' to this bit has no effect.

Bits 23:16 – CLKDIV[7:0] Clock Divider bits[1,2,3]

Bit 15 – ON Module Enable bit

Value	Description
1	Module is enabled.
0	Module is disabled.

Bit 13 – SLPEN Module Sleep Enable bit

Value	Description
1	Module operates in Sleep mode.
0	Module disabled in Sleep mode.

Bit 12 – SIDL Module Stop in Idle Mode Enable bit

Value	Description
1	Module disabled in Idle mode.
0	Module operates in Idle mode.

Bit 11 – CDM Control Data bit (Host)

Value	Description
1	Host sends the control data bit (CDM).
0	Host has not sent the control data bit (CDM).

Bit 10 – CDS Control Data bit (Client)

Value	Description
1	Client sends the control data bit (CDS).
0	Client has not sent the control data bit (CDS).

Bit 9 – SENSESEL Sense Selection bit

Value	Description
1	Use external sense.
0	Use software sense.

Bit 8 – GETSENSE Software Get-Sense bit

Value	Description
1	Software data transmission is triggered.
0	Software data transmission is not triggered.

Bit 4 – REGACC Register RAM Access Status bit

Value	Description
1	BiSS Protocol Engine is accessing register RAM.
0	BiSS Protocol Engine is not accessing register RAM.

Bit 3 – BNKNUM SCD RAM Access Status bit

Value	Description
1	BiSS Protocol Engine is accessing RAM1.
0	BiSS Protocol Engine is accessing RAM2.

Bit 2 – ACTIVE BiSS Active Status bit

Value	Description
1	The BiSS module is active.
0	The BiSS module is inactive.

Bit 0 – CLKSEL BiSS Baud Clock Selection bit
See Table 26-2.

26.4. Operation

One of the advantages of the host using BiSS is that data can be continuously sent from an actuator or a sensor, while also accessing registers and sending commands. With the combination of the two data types, sensor/actuator data and control/register data, it is important to know each separately and how they interact to form a BiSS frame.

26.4.1. General Setup

The following sections describe the setup of the SCD communication and how it can be used to configure the different communication types. Since each control communication requires an SCD communication, this setup portion is also relevant to the control communication configuration.

26.4.1.1. Peripheral Pin Select (PPS)

This device has PPS to allow virtual mapping pins to any available PPS port. Virtually mapped pins allow a versatile pinout, but they must be configured properly before any BiSS communication can occur.

26.4.1.2. Clock and SCD Frequency

The BiSS clock is generated from the clock source selected using CLKSEL bits. Use the CLKDIV register bits to divide this clock further. It is the user's responsibility to select the appropriate value of CLKDIV to generate a 20 MHz protocol clock based on BiSS clock selection.

After the clock division, the clock is fed through the SFREQ divider and becomes the clock at which single-cycle data will be transmitted.

26.4.1.3. Getsense Frequency

In addition to setting the SCD frequency, it also necessary to set the desired sample rate for getsense. The SCD frequency is the rate of the individual SCD transmission, but the rate that the SCD transmissions are triggered is the getsense frequency; this is also the rate that the B1SCDATAxX registers are updated. There are three selections to choose from: AGSMIN, AGSINFINITE and a user value.

26.4.1.3.1. User Value

The user value works like a clock divider and allows the user to set a value for the divider to determine when getsense will trigger.

26.4.1.3.2. AGSMIN

The hexadecimal value equivalent of AGSMIN is 0x7C. This setting allows the BiSS engine to automatically trigger getsense as soon as possible according to the SCD frequency selected.

26.4.1.3.3. AGSINF

The hexadecimal value equivalent of AGSINF is 0x7D. This setting prevents getsense from automatically triggering and is used when the user wants to manually trigger getsense.

26.4.2. Sensor Mode/Actuator Communication

In Sensor mode, the bus is operated like the SSI protocol. The host sends clock pulses, and eventually the data line drops low to Acknowledge and then data transmission occurs. The MA transmits clock pulses that are transmitted to all clients on the bus. The host triggers a Start condition and the clients respond and transmit their data.

26.4.2.1. Configuring Single-Cycle Data

Client location one is set by default and no further action is needed if a single client is used. Channel one must be configured for the protocol used, which requires the user to use the CLCHCFG bit to select between BiSS C and SSI. At this point, the actuator and sensor selection can be configured using ACTSEN. By default, this is set for sensor data.

Next, the appropriate values must be configured for the SCD and CRC. Depending on what hardware is being used as the client, there are multiple options available for SCD and CRC selections. For this example, consider a common setup where the client expects 28 bits of sensor data to be transmitted, with an error and warning bit, along with the very common 0x43 CRC polynomial which accounts for six bits.

For this BiSS module, the SCD length is considered the 28 bits with two additional bits from the warning and error bits which combined create an SCD of 30. The appropriate bits can now be set up which include, SCDLEN, CRCSEL and CRCLEN; then the module can be enabled by using SCDEN.

26.4.2.2. Reading a Sensor Value

Once all the settings are configured, values from the sensor can be read. The simplest way to read a value is to turn on the BiSS module, set the AGS bit and read from the B1SCDATAxX register. This is the minimum required to read a value, however, the following additional steps are highly recommended:

1. After configuring all settings, turn on the BISS module using the ON bit. This is also a good time to monitor the SLx Status bit. The SLx bit quickly checks the status of the SL line to determine if the client is pulling the line high, which is required for successful BiSS transmission.
   Note: Monitoring the SLx status bit is used to determine if the client is powered on and connected.
2. Initialize the BiSS module using the INIT bit. Initialization has two useful purposes: when the INIT bit is set, the status information is reset to provide assurance that the BiSS module starts off without any erroneous errors. Initialization also sends two pulses that allow the BiSS engine to determine the line delay introduced on MA. This value can immediately be read in the B1SCDATA0L register.
3. After the BiSS module is initialized, poll INSTRWA. It is critical to operations that the instructions are not written to while still in progress. Then, the AGS bit can be set to start triggering the getsense to allow the user to read the B1SCDATAxX registers for the sensor values.

26.4.2.3. Reading SCDATA Registers

There are also two suggested methods for reading an SCD register:

- The register can be read directly within the host code from SCDATA, or
- The BiSS interrupt can be configured to trigger at the end of every BiSS transmission.

To configure the BiSS interrupt, set the interrupt enable, clear the interrupt flag and confirm there is an Interrupt Service Routine (ISR). In both methods, it is recommended to poll for an end of transmission (`_EOT`) to verify a successful transmission. If CRC is being used, it is also recommended to poll after the Client SCD valid bit (`_CLSCDVx`) is set. This bit is set after a successful CRC value is detected. Once they are polled and successfully checked, the B1SCDATA0L and B1SCDATA0H can be read by the user.

26.4.2.4. Transmitting Sensor Data

To summarize the following sections, use the following steps and the Sensor Communication Code Example.

1. Configure BiSS PPS.
2. Configure the SCD frequency and getsense frequency.
3. Select the client location, the protocol used on the channel and determine if the client device is a sensor or an actuator.
4. Configure the SCD length.
5. Configure the CRC used and the CRC length.
6. Turn the module on.
7. Initialize the BiSS module and poll INSTRWA (can read B1SCDATAxX now if line delay calculations are desired later).
8. Turn AGS on.
9. Poll end of transmission, EOT, Status bit and/or CRC Valid bit to confirm a successful transmission.
10. Read the value from B1SCDATAxX.

26.4.2.5. Manual Getsense

AGS is an easy way to quickly get sensor data from a device, but sometimes only single getsenses are desired. In this case, enable BISS ON and set the AGS frequency to 0x7D, which is also AGSINF. This forces AGS to wait for a manually triggered getsense. To enable a manually triggered getsense, set AGS, and once the manually triggered getsense is desired, the _GETSENSE bit must be set. Setting the _GETSENSE bit triggers the clock to generate enough pulses for SCD and to update B1SCDATAxX. Then, getsense must be cleared by the user to be able to set it again and trigger another getsense.

26.4.3. Control Communication

Control communication is built from multiple SCD communications. Each time an SCD is sent, bits are stored. Once enough bits are stored, then a full control communication is formed. The addition of a control frame allows the ability to read and write to client registers and also send commands to addresses or all clients. To create this control frame, each CDM and CDS bit from the BiSS frames are combined to create a control "packet." To initiate a control frame, 14 zeroes must be transmitted by the host as the CDM. Because of this, each control communication has SCD communication, which means to setup control communication, the SCD communication still must be set up. In this case, the setup for sensor communication can be used, and then the control communication for reading and writing to registers can be focused on next.

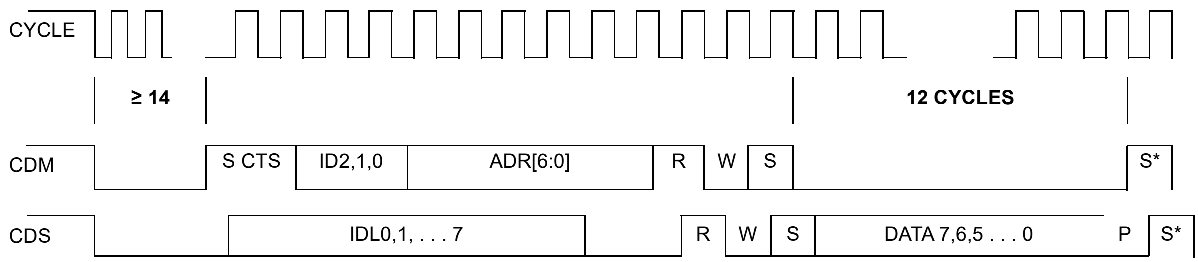
26.4.3.1. Register Communication

After the SCD/sensor communication has been set up, the additional setup for reading registers can be completed. CTS must be set up before any communication occurs. CTS selects between only register communication, which is communication between host and client registers, or command communication, which is a bus style communication. The client ID must also be set to make sure the exact device is communicated with; this is done in the CLNTID register.

26.4.3.1.1. Read Register

Adding Register mode to Sensor mode allows SSI to evolve into BiSS. Register mode allows control of the reading and writing of registers by "saving" multiple CDS and CDM bits from multiple BiSS frames and combining them to form a full control frame. Therefore, Register mode always includes Sensor mode. There are different ways to use Register mode, such as reading and writing to registers and sending commands (see Initiating a Register Read).

Figure 26-8. Register Read



Note:

1. The Control Select bit (CTS) is one (CTS =1) for the register communication.
2. The secondary register access has only three ID select bits and seven bits of register address.
3. * A Stop bit (P = 0) is at the end of the frame. Following the Stop bit, there can be an optional Start bit if the main must perform a sequential access of additional secondary registers.
4. After the Start bit, the main sends 120 bits and one Stop bit. The secondary responds with eight bits of data protected with four bits of CRC.

Initiating a Register Read

Like standard SCD communication, INIT should be set, INSTRWA should be polled and AGS should be set.

Before this instruction is set, CDMTO should be polled. This status bit allows the user to poll CDMTO to confirm that the required 14 CDM low bits have been transmitted. Once this is set up, an instruction value can be written into the INSTR register to instruct the BiSS engine the type of instruction/command that will be executed. INSTR should not be written to while it is in progress of fulfilling an instruction, therefore INSTRWA should be polled again.

Communicating Register Read

To summarize the following sections, use the following steps and the Read Register Code Example.

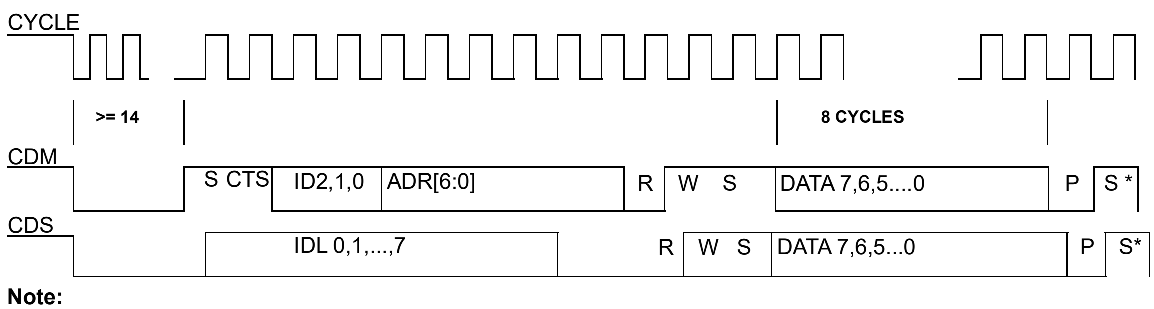
1. Configure BiSS PPS.
2. Configure the SCD frequency and getsense frequency.
3. Select the client location, the protocol used on the channel and determine if the client device is a sensor or an actuator.
4. Configure the SCD length.
5. Configure the CRC used and the CRC length.
6. Select a start address to read registers from and select how many consecutive bits are desired to be read.
7. Set the read or write bit to read.
8. Set the CTS bit to register communication and the Prototype Select bit to BiSS C.
9. Set the client ID.
10. Turn the module on.
11. Initialize the BiSS module and poll INSTRWA (can read B1SCDATAxX now if line delay compensation is desired).
12. Set AGS to start AutoGetsense.
13. Poll CDMTO to confirm 14 CDM low bits have been received.

14. Set the instruction register to the instruction for reading and poll the INSTRWA bit to make sure nothing is written to INSTR while the current instruction is in progress.
15. Poll the End of Transmission Status bit and/or CRC Valid bit and/or Register End bit, REGEND, to confirm that a successful SCD receive and a successful register communication are received.
16. Read values from B1SCDATAxX and B1RDATAx.

26.4.3.1.2. Write Registers

The same setup from the regular single-cycle data example can be applied to writing to a BiSS C client's registers. Once set up, the specific registers related to writing to registers can be considered.

Figure 26-9. Register Write



Note:

1. The Control Select bit (CTS) is one (CTS =1) for the register communication.
2. The client register access has only three ID select bits and seven bits of register address.
3. * A Stop bit (P = 0) is at the end of the frame. Following the Stop bit, there can be an optional Start bit if the host must perform a sequential access of additional secondary registers.

Additional Setup for Register Write Communication

The TX Read Enable bit must be set to allow writing to the TX buffers. Otherwise, this bit is cleared and prevents unwanted writes while reading. The Write/Read Register Select bit also needs to be set to determine the direction of communication.

In both read and write, BiSS can consecutively increment and read/write register addresses. To do this, a starting address should be designated using the Start Address bit. In the Read Register Code Example, the register reading started at 0x00. In this example, the register reading starts at 0x50. Since reading/writing has been determined to occur consecutively in the BiSS module, the user can select how many registers to read/write consecutively by setting the register number register. The register number register is not equivalent to bits. A single register communication for REGNUM purposes is eight bits (see REGNUM). For the same reasons listed in the read register example, the CTS, PROTOSEL and CLNTID registers can all be set now.

REGNUM

For this BiSS engine and communication between registers, the REGNUM register determines that a single register is considered eight bits and cannot be `0`. If REGNUM is set to `0`, a single REGNUM register will be transmitted, which is equivalent to eight bits.

Initializing a Register Write

To initiate a write, follow the same steps Initiating a Register Read. When initializing a register write, B1RDATAx must be set to the desired value to be written. Also, the INSTR register must be entered with the instruction value for writing instead of reading. The remaining steps are the same as the read example.

Communicating Register Write

To summarize the following sections, use the following steps and the Write Register Code Example.

1. Configure BiSS PPS.
2. Configure the SCD frequency and getsense frequency.
3. Select the client location, the protocol used on the channel and determine if the client device is a sensor or an actuator.
4. Configure the SCD length.
5. Configure the CRC used and the CRC length.
6. Select a start address to write registers from and select how many consecutive bits are desired to be written.
7. Set the read or write bit to write. Set the CTS bit to register communication and the Prototype Select bit to BiSS C.
8. Set the client ID.
9. Set TX Read Enable bit to allow writing to the B1SCDATAxX registers.
10. Turn the BiSS module on.
11. Initialize the BiSS module with INIT and poll INSTRWA (can read B1SCDATAxX right now if line delay compensation is desired).
12. Set the AGS bit to start Autogetsense.
13. Set B1RDATAxX to the desired value to be written to the client register.
14. Poll CDMTO to confirm 14 CDM low bits have been received.
15. Set the Instruction register to the instruction for writing and poll the INSTRWA Status bit to make sure nothing is written to INSTR while the current instruction is in progress.
16. Poll end of transmission, EOT, Status bit and/or CRC Valid bit and/or Register End bit, REGEND, to confirm a successful SCD receive and a successful register communication has been transmitted.
17. Read value from B1SCDATAxX and B1RDATAx.

26.4.4. Command Communication

The BiSS module allows commands to be sent simultaneously to four clients in any desired combination. This can be achieved by addressing specific clients or broadcasting to all clients on the bus.

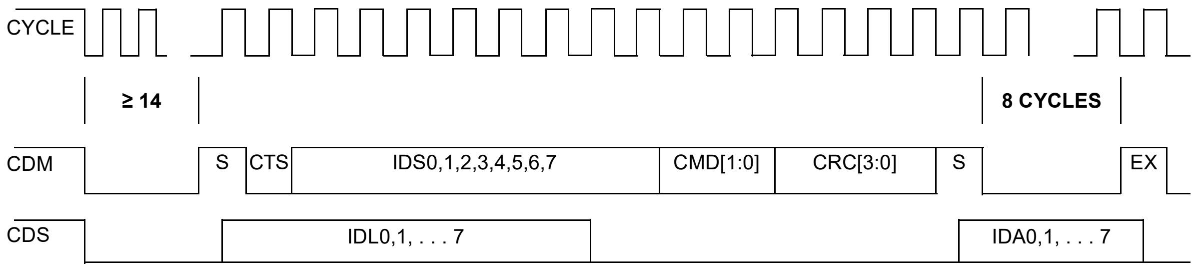
Table 26-3. Broadcast Command Options

Command	Description
11	Reserved
10	All bus couplers switched to bypass.
01	Register and command communication are activated.
00	Data channels are deactivated.

Table 26-4. Addressed Command Options

Command	Description
11	Definable for each secondary
10	The addressed bus couple switches from bypass to line operation.
01	Register and command communication of addressed secondary are deactivated.
00	Addressed data channel is activated.

Figure 26-10. Command Frame



Note:

1. The Control Select bit (CTS) is zero (CTS =0) for the command frame.
2. The secondary ID is assigned according to the sequence in the chain.
3. To send commands to all secondaries (broadcast) IDS bits= 00000000 (none of the IDS bits is set.).

26.4.4.1. Broadcast Communication

Follow the same setup procedures as described in Register Communication with a few modifications. Set the value of the command desired using the Command bit and changing CTS from '1' for register communication to '0' for command communication, then setting the IDSx to '0'. A '0' in the IDSx register signals a broadcast command that will go to all clients. For an example, see Broadcast Command Code Example.

26.4.4.2. Addressed Communication

The Addressed Command mode is the same as the Broadcast mode, except instead of '0' in IDSx, an actual address is used. This will send a command to the specific client addressed. For an example, see Addressed Command Code Example.

26.4.5. Additional BiSS Module Features

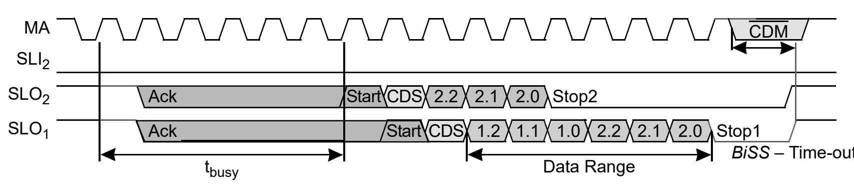
In addition to several communication options, the BiSS module also has options for processing times, line delays and how the memory banks are used.

26.4.5.1. Processing Time Requests

Before sending sensor data, a client can request additional processing time. This request can be used in an instance where the client needs additional time for analog-to-digital conversion or memory access. To request additional processing time, the client can hold the ACK signal low for an additional clock cycle. This, in turn, delays the Start bit, which is the indicator that additional processing time is requested.

If a device in point-to-point configuration consists of several clients, then all but the last client will pass through the data of their predecessor received at SLI and then send this to SLO following its own data. The client with the longest processing time should be used as the last client since it determines the total processing time.

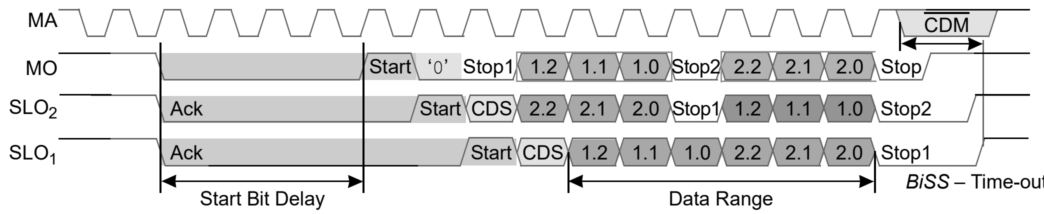
Figure 26-11. Requests for Processing Time (Point-To-Point Configuration)



26.4.5.2. Processing Time Configuration

In the bus configuration, the host delays the output of the Start bit to MO. For this purpose, the host is configured to the maximum delay time of all connected clients during the bus establishment.

Figure 26-12. Configurable Processing Time (Bus Configuration)



26.4.5.3. Line Delay Compensation

Due to various cable lengths expected in BiSS applications, automatic line delay compensation has been included as part of the BiSS engine. Automatic line delay compensation can be calculated as the difference in time from when the client first receives a Clock signal from the host to the time when the host first receives an Acknowledge from the client (see Figure 26-13). The included INIT bit within the BiINSTR register not only resets status registers, but also initiates the line delay compensation by initializing the channel. The INIT bit can be used and then read from the BiSCDATA0L register to read the detected delay value. To see the behavior of the INIT sequence, see Figure 26-14.

Figure 26-13. Line Delay Compensation

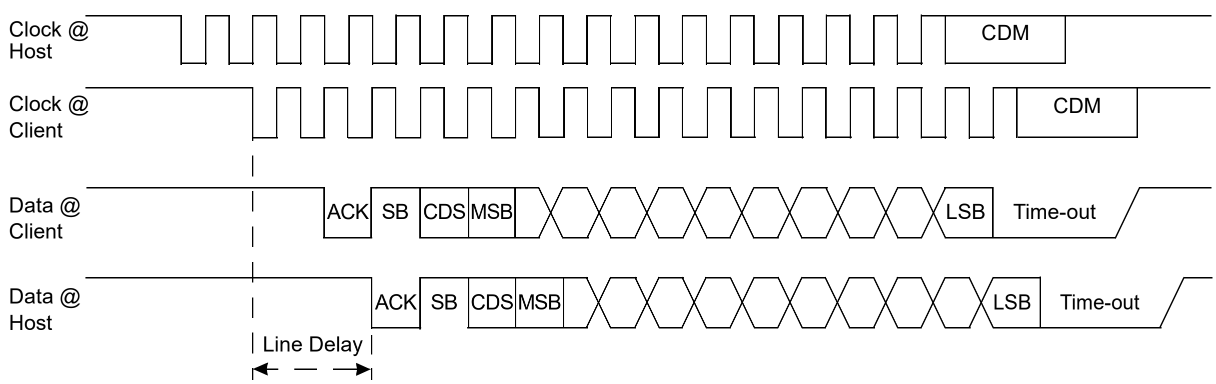
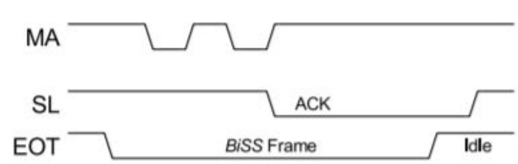


Figure 26-14. BiSS C Initialization Sequence



26.4.5.4. Clock Frequencies for RS422 Interfaces

BiSS is commonly used with RS422 and RS485 interfaces. BiSS is able to match the SSI protocol easily as long as proper lengths of cable and speeds are set. For an approximation of different cable lengths for RS422 interfaces, refer to Table 26-5.

Table 26-5. RS422 Interfaces

Cable Length	Clock Frequency
up to 10m	10 MHz
up to 25m	5 MHz
up to 60m	2 MHz
up to 100m	1 MHz
up to 200m	500 KHz
up to 500m	200 KHz
up to 1000m	100 KHz

Note:
1. Data rate and clock frequency depend on the physical layer used for BiSS communications.

26.4.5.5. Memory Bank Switching

Since BiSS can operate at high speeds of transmission, the BiSS module incorporates memory bank switching. The BiSS module has two memory banks. The status of which memory bank is active can be read by monitoring the BNKNUM bit. It is critical that the registers are not written to during bank switching; therefore, there is also a register access bit to monitor if registers are currently being accessed. By default, bank switching is enabled, but the user can disable it using the SBANK bit. The bank switching feature is also able to hold the bank switch during reads to confirm that data integrity is maintained. There is also the ability to manually switch banks with the BANKSWEN bit. These bits combine to allow for different ways to maintain control of the module's bank switching.

26.4.6. CRC

Optionally, Cyclic Redundancy Checking can be enabled using the CRCLEN bits within BiCLTCONx for data channels and control channels. This is one way to help determine if any issue has occurred during transmission. To implement this function, a checksum is calculated from the desired transmission and then converted into its CRC value. This value can then be checked during transmission to determine if any issues have already occurred. Depending on which CRC (Data Channel or Control) is being transmitted determines where in the protocol these values will be transmitted. The type of CRC also determines how many bits are available: up to 17 bits for Data Channel CRC and four bits for Control CRC. The length of the CRC data channel can be set in the CRCLEN bits within BiCLTCONx. Settings for CRC can be configured in the BiCLTCONx and BiCTRLCONx registers.

If there is sufficient free memory available for CRC data, the read and inverted CRC bits are stored at the beginning at the highest address downwards. The storage of the CRC can be disabled with NOCRC.

26.5. Application Examples

26.5.1. Sensor Communication Code Example

```
// Read BiSS C SCD Frame, Sensor Data
    int startupDelay; // Variable to read startup delay value into
    uint32_t sensorValue; // Variable to store sensor values

    B1CTRLCONbits.PROTOSEL = 1;     // Setting the protocol selection to BiSS C

    // Configuring Host Clock for BiSS Module
    B1CONbits.CLKDIV = 2;         // Setting CLKDIV divider
    B1CTRLCONbits.SFREQ = 5;        // Setting SFREQ divider
    B1CCONbits.FREQAGS = 0x7C;      // Setting FREQAGS to AGSMIN, host will automatically 
trigger another SCD cycle once the previous has finished

    // Configuring BiSS First Channel, Channel 0
    B1CTRLCONbits.CHSEL = 1;       // Setting CTRLCON such that 1st channel, Channel 0, is 
being used for communication
    B1CHCONbits.CLCHCFG0 = 1;      // Setting CLCHCFG0 such that 1st channel, Channel 0, 
is configured for BiSS C
    B1CLTCON0bits.SCDEN0 = 1;      // Enabling SCD data
```

```
    B1CLTCON0bits.SCDLEN0 = 29;      // Configuring SCD data length, 28 data bits, 1 error
bit, 1 warning bit
    B1CLTCON0bits.CRCSEL0 = 0;     // Setting CRCSEL such that the CRC value will be
calculated based on CRCLEN dedicated polynomial
    B1CLTCON0bits.CRCLEN0CRCPOLY0 = 0b0000110; // Setting CRC polynomial to dedicated
value, 0x43
    B1CTRLCONbits.IDADISCLNTID0 = 0x00; // Setting the Client ID for Channel 0, in this
case the Client ID is 0x00
    B1CTRLCONbits.CMDCLNTID21 = 0b00;

    // Asserting BiSS Communication
    B1CONbits.ON = 1;             // Turn on the BiSS Module
    B1INSTRbits.INIT = 1; // Initialize sequence which resets status registers and
initializes a line delay sense
    while(B1CONbits.INSTRWA == 1);     // Waiting for initialization/instruction to
complete before any further write to instruction register
    startupDelay = B1SCDATA0L;         // Optional to read startup delay and save to
variable
    B1INSTRbits.AGS = 1; // Starting automatic data transmission

    int i = 10; // Creating while loop to read sensor ten times

    while(i--) {
        while(B1STATbits.EOT == 0);     // Waiting for the End of Transmission status bit
to confirm receive has occurred
        while(B1STATbits.CLSCDV0 == 0);   // Waiting for the Client SCD Valid bit to
confirm a successful receive has occurred

        sensorValue = B1SCDATA0L; // Reading sensor value into variable
    }
```

26.5.2. Read Register Code Example

```
// Read BiSS C Register Data
    int registerValue; // Variable to store register value
    B1CTRLCONbits.PROTOSEL = 1;                 // Setting the protocol selection to BiSS
C

    // Configuring Host Clock for BiSS Module
    B1CONbits.CLKDIV = 5;                       // Setting CLKDIV divider
    B1CTRLCONbits.SFREQ = 5;                    // Setting SFREQ divider
    B1CCONbits.FREQAGS = 0x7C; // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished

    // Configuring BiSS First Channel, Channel 0
    B1CTRLCONbits.CHSEL = 1;                // Setting CTRLCON such that 1st channel,
Channel 0, is being used for communication
    B1CHCONbits.CLCHCFG0 = 1;               // Setting CLCHCFG0 such that 1st channel,
Channel 0, is configured for BiSS C
    B1CLTCON0bits.SCDEN0 = 1;               // Enabling SCD data
    B1CLTCON0bits.SCDLEN0 = 29;             // Configuring SCD data length, 28 data bits,
1 error bit, 1 warning bit
    B1CLTCON0bits.CRCSEL0 = 0;             // Setting CRCSEL such that the CRC value will
be calculated based on CRCLEN dedicated polynomial
    B1CLTCON0bits.CRCLEN0CRCPOLY0 = 0b0000110; // Setting CRC polynomial to dedicated
value, 0x43
    B1CTRLCONbits.IDADISCLNTID0 = 0x00; // Setting the Client ID for Channel 0, in this
case the Client ID is 0x00
    B1CTRLCONbits.CMDCLNTID21 = 0b00;

    // Setup for Register Read
    B1CTRLCONbits.CTS = 1; // Setting CTS for Register communication
    B1RCCONbits.WNR = 0; // Setting WNR for Read Register
    B1RCCONbits.STRTADDR = 0x00;            // Start reading data at start address of 0x00
in BiSS Client
    B1RCCONbits.REGNUM = 0; // Declare how many registers to read from the BiSS Client, 0
means read 1 register

    // Asserting BiSS Communication
    B1CONbits.ON = 1; // Turn on the BiSS Module
    B1INSTRbits.INIT = 1; // Initialize sequence which resets status registers and
initializes a line delay sense
    while(B1CONbits.INSTRWA == 1);         // Waiting for initialization/instruction to
complete before any further write to instruction register
    B1INSTRbits.AGS = 1; // Starting automatic data transmission

    // Initializing Register Read
    while(B1STATbits.CDMTO == 0);          // Waiting until CDM Time Out to Read Register
    B1INSTRbits.INSTR = 0b100;             // Using INSTR code to assert a register read
    while(B1CONbits.INSTRWA == 1);         // Waiting for instruction to complete before
any further write to instruction register
    // Waiting for Valid Receive
```

```
    while(B1STATbits.EOT == 0);          // Waiting for the End of Transmission status bit
to confirm receive has occurred
    while(B1STATbits.CLSCDV0 == 0);      // Waiting for the Client SCD Valid bit to
confirm a successful receive has occurred
    while(B1STATbits.REGEND == 0);       // Waiting for the register receive to complete
    registerValue = B1RDATA0;            // Saving register value to a variable
```

26.5.3. Write Register Code Example

```
// Write BiSS C Register Data
    B1CTRLCONbits.PROTOSEL = 1;                 // Setting the protocol selection to BiSS
C

    // Configuring Host Clock for BiSS Module
    B1CONbits.CLKDIV = 5;                      // Setting CLKDIV divider
    B1CTRLCONbits.SFREQ = 5;                   // Setting SFREQ divider
    B1CCONbits.FREQAGS = 0x7C; // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished

    // Configuring BiSS First Channel, Channel 0
    B1CTRLCONbits.CHSEL = 1;                  // Setting CTRLCON such that 1st channel,
Channel 0, is being used for communication
    B1CHCONbits.CLCHCFG0 = 1;                 // Setting CLCHCFG0 such that 1st channel,
Channel 0, is configured for BiSS C
    B1CLTCON0bits.SCDEN0 = 1;                 // Enabling SCD data
    B1CLTCON0bits.SCDLEN0 = 29;               // Configuring SCD data length, 28 data bits,
1 error bit, 1 warning bit
    B1CLTCON0bits.CRCSEL0 = 0;               // Setting CRCSEL such that the CRC value will
be calculated based on CRCLEN dedicated polynomial
    B1CLTCON0bits.CRCLEN0CRCPOLY0 = 0b0000110; // Setting CRC polynomial to dedicated
value, 0x43
    B1CTRLCONbits.IDADISCLNTID0 = 0x00;       // Setting the Client ID for Channel 0, in
this case the Client ID is 0x00
    B1CTRLCONbits.CMDCLNTID21 = 0b00;

    // Setup for Register Write
    B1CTRLCONbits.CTS = 1; // Setting CTS for Register communication
    B1RCCONbits.WNR = 1; // Setting WNR for Write Register
    B1RCCONbits.STRTADDR = 0x50;              // Start writing data at start address of
0x50 in BiSS Client
    B1RCCONbits.REGNUM = 0;                   // Declare how many registers to write from
the BiSS Client, 0 means read 1 register
    B1CONbits.TXRDEN = 1;                     // Setting the transmit RAM enable bit to
enable transmission

    // Asserting BiSS Communication
    B1CONbits.ON = 1; // Turn on the BiSS Module
    B1INSTRbits.INIT = 1; // Initialize sequence which resets status registers and
initializes a line delay sense
    while(B1CONbits.INSTRWA == 1); // Waiting for initialization/instruction to complete
before any further write to instruction register
    B1INSTRbits.AGS = 1; // Starting automatic data transmission
    B1RDATA0 = 0x23; // Setting the desired register value to transmit

    // Initializing Register Write
    while(B1STATbits.CDMTO == 0); // Waiting until CDM Time Out to Write Register
    B1INSTRbits.INSTR = 0b100; // Using INSTR code to assert a register write
    while(B1CONbits.INSTRWA == 1); // Waiting for instruction to complete before any
further write to instruction register

    // Waiting for Valid Transmissions
    while(B1STATbits.EOT == 0); // Waiting for the End of Transmission status bit to
confirm transmission has occurred
    while(B1STATbits.CLSCDV0 == 0); // Waiting for the Client SCD Valid bit to confirm a
successful transmission has occurred
    while(B1STATbits.REGEND == 0); // Waiting for the register transmission to complete
```

26.5.4. Broadcast Command Code Example

```
// Send Broadcast Command
B1CTRLCONbits.PROTOSEL = 1;          // Setting the protocol selection to BiSS C

// Configuring Host Clock for BiSS Module
B1CONbits.CLKDIV = 5;             // Setting CLKDIV divider
B1CTRLCONbits.SFREQ = 5;             // Setting SFREQ divider
B1CCONbits.FREQAGS = 0x7C;           // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished

// Configuring BiSS First Channel, Channel 0
B1CTRLCONbits.CHSEL = 1;             // Setting CTRLCON such that 1st channel, Channel 0,
```

```
is being used for communication
B1CHCONbits.CLCHCFG0 = 1;           // Setting CLCHCFG0 such that 1st channel, Channel 0,
is configured for BiSS C
B1CLTCON0bits.SCDEN0 = 1;           // Enabling SCD data
B1CLTCON0bits.SCDLEN0 = 29;         // Configuring SCD data length, 28 data bits, 1 error
bit, 1 warning bit
B1CLTCON0bits.CRCSEL0 = 0;          // Setting CRCSEL such that the CRC value will be
calculated based on CRCLEN dedicated polynomial
B1CLTCON0bits.CRCLEN0 = 0b0000110;     // Setting CRC polynomial to dedicated value, 0x43
B1CTRLCONbits.IDADISCLNTID0 = 0b1;     // Setting the Client ID for Channel 0, in this
case the Client ID is 0x01
B1CTRLCONbits.CMDCLNTID21 = 0b00;

// Setup for Broadcast Command
B1CTRLCONbits.CTS = 0;              // Setting CTS for Command communication
B1RCCONbits.WNR = 0;             // Setting WNR for Read Register (IS THIS NEEDED?)
B1RCCONbits.STRTADDR = 0x00;        // Start reading data at start address of 0x00 in BiSS
Client
B1RCCONbits.REGNUM = 0;             // Declare how many registers to read from the BiSS
Client, 0 means read 1 register
B1CTRLCONbits.CMDCLNTID21 = 2;         // Setting what command value to send, this should
be described in Client documentation

// Turning the module on and starting communication
B1CONbits.ON = 1;

B1INSTRbits.INIT = 1;            // Initialize sequence which resets status registers and
initializes a line delay sense
while(B1CONbits.INSTRWA == 1);         // Waiting for initialization/instruction to
complete before any further write to instruction register
B1INSTRbits.AGS = 1;                // Starting automatic data transmission

// Initializing Register Read
while(B1STATbits.CDMTO == 0);          // Waiting until CDM Time Out to Read Register
B1INSTRbits.INSTR = 0b100;          // Using INSTR code to send command(INSTR value is the
same as read register command but CTS = 1 means send a command)
while(B1CONbits.INSTRWA == 1);         // Waiting for instruction to complete before any
further write to instruction register

while(B1STATbits.REGEND == 0);         // Waiting for the register transmission to complete
```

26.5.5. Addressed Command Code Example

```
// Send Addressed Command
    B1CTRLCONbits.PROTOSEL = 1; // Setting the protocol selection to BiSS C

    // Configuring Host Clock for BiSS Module
    B1CONbits.CLKDIV = 5;        // Setting CLKDIV divider
    B1CTRLCONbits.SFREQ = 5;     // Setting SFREQ divider
    B1CCONbits.FREQAGS = 0x7C;   // Setting FREQAGS to AGSMIN, host will automatically
trigger another SCD cycle once the previous has finished

    // Configuring BiSS First Channel, Channel 0
    B1CTRLCONbits.CHSEL = 1;    // Setting CTRLCON such that 1st channel, Channel 0, is
being used for communication
    B1CHCONbits.CLCHCFG0 = 1;   // Setting CLCHCFG0 such that 1st channel, Channel 0, is
configured for BiSS C
    B1CLTCON0bits.SCDEN0 = 1;   // Enabling SCD data
    B1CLTCON0bits.SCDLEN0 = 29; // Configuring SCD data length, 28 data bits, 1 error bit,
1 warning bit
    B1CLTCON0bits.CRCSEL0 = 0;  // Setting CRCSEL such that the CRC value will be
calculated based on CRCLEN dedicated polynomial
    B1CLTCON0bits.CRCLEN0CRCPOLY0 = 0b0000110; // Setting CRC polynomial to dedicated
value, 0x43
    B1CTRLCONbits.IDADISCLNTID0 = 0b1; // Setting the Client ID for Channel 0, in this
case the Client ID is 0x01
    B1CTRLCONbits.CMDCLNTID21 = 0b00;

    // Setup for Addressed Command
    B1CTRLCONbits.CTS = 0;         // Setting CTS for Command communication
    B1RCCONbits.WNR = 0; // Setting WNR for Read Register
    B1RCCONbits.STRTADDR = 0x00;   // Start reading data at start address of 0x00 in BiSS
Client
    B1RCCONbits.REGNUM = 0;        // Declare how many registers to read from the BiSS
Client, 0 means read 1 register
    B1CTRLCONbits.CMDCLNTID21 = 2; // Setting what command value to send, this should be
described in Client documentation
    B1IDS0bits.IDS0 = 0x80;        // Setting the Client address for sending command

    // Turning the module on and starting communication
    B1CONbits.ON = 1;
    B1INSTRbits.INIT = 1; // Initialize sequence which resets status registers and
initializes a line delay sense
    while(B1CONbits.INSTRWA == 1); // Waiting for initialization/instruction to complete
```

```
before any further write to instruction register
    B1INSTRbits.AGS = 1; // Starting automatic data transmission

    // Initializing Register Read
    while(B1STATbits.CDMTO == 0); // Waiting until CDM Time Out to Read Register
    B1INSTRbits.INSTR = 0b100; // Using INSTR code to send command(INSTR value is the same
as read register command but CTS = 1 means send a command)
    while(B1CONbits.INSTRWA == 1); // Waiting for instruction to complete before any
further write to instruction register
```

26.6. Interrupts

The BiSS module generates two interrupts: end of transmission (EOT) and transmission error (ERR). These interrupts can be used by clearing and enabling their appropriate interrupt flags.

26.6.1. End Of Transmission (EOT) Interrupt

The EOT interrupt is enabled using the relevant IECx register, and the flag is cleared and set in the appropriate IFSx register. This interrupt is asserted each time a successful transmission has completed.

26.6.2. Transmission Error (ERR) Interrupt

The ERR interrupt is enabled using the relevant IECx register, and the flag is cleared and set in the appropriate IFSx register. This interrupt is asserted when any of the BiSS error bits are set. The user can read the status flags to determine the specific error.

26.7. Power Saving Modes

BiSS provides support in power-saving modes, including the capability to run in Sleep and Idle modes.

26.7.1. Sleep

Setting the SLPEN bit within the BiCON register will keep the module active in Sleep mode.

26.7.2. Idle

Setting the SIDL bit within the BiCON register will turn the module off in Idle mode.

26.8. Terminology

Table 26-6. Terminology

Acronym	Description
MA	Clock Output of the Host
MO	Main Data Output of the Host
SL	Data Output
SLI	Data Input of the Client
SLO	Data Output of the Client
BiSS	Bidirectional Serial Synchronous
SSI	Serial Synchronous Interface
CDM	Control Data Host
CDS	Control Data Client
CRC	Cyclic Redundancy Check
nE	Error Bit
nW	Warning Bit
SCD	Single-Cycle Data
CD	Control Data
CTS	Control Select Bit

Table 26-6. Terminology (continued)

Acronym	Description
ID	Identification
IDS	ID Select
IDA	ID Acknowledge
IDL	ID Lock
ADR	Address
S	Start
P	Stop
CMD	Command
EX	Execute
*S	Optional Start
Cyclical	Occurring in cycles, in a predictive and deterministic manner
Isochronous	Occurring at the same time

27. Timers

The Timer module is a 32-bit timer that can be configured for both synchronous and asynchronous operations. The Timer module features include:

- Asynchronous and Synchronous Operation
- External Clock
- Various Timer/Counter Modes

27.1. Device-Specific Information

Table 27-1. Timer Summary Table

Timer Module Instances	Peripheral Bus Speed	Clock Source
3	Standard (1:2 CPU Clock)	Standard Speed Peripheral Clock(System Clock/2)

Table 27-2. TCS Timer Clock Source Select bit

Value	Description
1	External clock source, TxCK
0	Standard (1:2 CPU Clock)

27.2. Architectural Overview

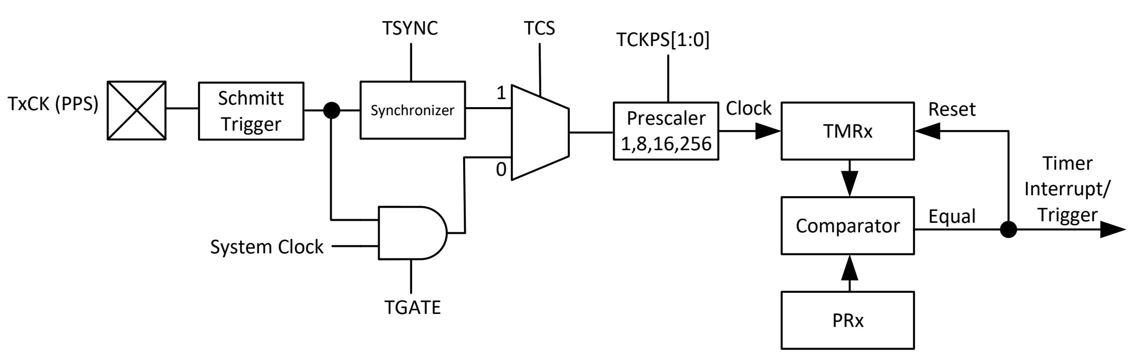
The timer module found in dsPIC33A family devices shares many features with a classic Type A timer. There is typically a single timer module implemented for each CPU core present in a device. Timers are useful for generating accurate time-based periodic interrupt events for software applications or real-time operating systems. Other uses include counting external pulses or accurate timing measurement of external events by using the timer's gate feature.

The timers found in some dsPIC33A family devices include the following features:

- Asynchronous and Synchronous Operation
- Software-Selectable Internal or External Clock Sources
- Programmable Interrupt Generation and Priority
- Gated External Pulse Counter
- Operation During Sleep Mode
- Software Prescalers: 1:1, 1:8, 1:64 and 1:256

Figure 27-1 illustrates the block diagram of a Timer module.

Figure 27-1. Timer Block Diagram



27.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1CE0	T1CON	31:24								
		23:16								
		15:8	ON		SIDL	TMWDIS	TMWIP	PRWIP		
		7:0	TGATE		TCKPS[1:0]			TSYNC	TCS	
0x1CE4	TMR1	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1CE8	PR1	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1CEC ... 0x1CEF	Reserved									
0x1CF0	T2CON	31:24								
		23:16								
		15:8	ON		SIDL	TMWDIS	TMWIP	PRWIP		
		7:0	TGATE		TCKPS[1:0]			TSYNC	TCS	
0x1CF4	TMR2	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1CF8	PR2	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1CFC ... 0x1CFF	Reserved									
0x1D00	T3CON	31:24								
		23:16								
		15:8	ON		SIDL	TMWDIS	TMWIP	PRWIP		
		7:0	TGATE		TCKPS[1:0]			TSYNC	TCS	
0x1D04	TMR3	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1D08	PR3	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							

27.3.1. Timer x Control Register

Name: TxCON
Offset: 0x1CE0, 0x1CF0, 0x1D00

Note:
1. When Timer x is enabled in External Synchronous Counter mode (TCS = 1, TSYNC = 1, TON = 1), any attempts by user software to write to the TMR1 register are ignored.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	TMWDIS	TMWIP	PRWIP		
Access	R/W		R/W	R/W	R	R		
Reset	0		0	0	0	0		

Bit	7	6	5	4	3	2	1	0
	TGATE		TCKPS[1:0]			TSYNC	TCS	
Access	R/W		R/W	R/W		R/W	R/W	
Reset	0		0	0		0	0	

Bit 15 – ON Timer On bit[(1)]

Value	Description
1	Starts 32-bit Timer
0	Stops 32-bit Timer

Bit 13 – SIDL Timer Stop in Idle Mode bit

Value	Description
1	Discontinues the module operation when the device enters Idle mode.
0	Continues the module operation in Idle mode.

Bit 12 – TMWDIS Asynchronous Timer Write Disable bit

Value	Description
1	Timer writes are ignored while a posted write to TMRx or PRx is synchronized to the asynchronous clock domain.
0	Back-to-back writes are enabled in Asynchronous mode.

Bit 11 – TMWIP Asynchronous Timer Write in Progress bit

Value	Description
1	Write to the timer in Asynchronous mode is pending.
0	Write to the timer in Asynchronous mode is complete.

Bit 10 – PRWIP Asynchronous Period Write in Progress bit

Value	Description
1	Write to the Period register in Asynchronous mode is pending.
0	Write to the Period register in Asynchronous mode is complete.

Bit 7 – TGATE Timer Gated Time Accumulation Enable bit

When TCS = 1:

This bit is ignored.

When TCS = 0:

Value	Description
1	Gated time accumulation is enabled.
0	Gated time accumulation is disabled.

Bits 5:4 – TCKPS[1:0] Timer Input Clock Prescale Select bits

Value	Description
11	1:256
10	1:64
01	1:8
00	1:1

Bit 2 – TSYNC Timer External Clock Input Synchronization Select bit[(1)]

When TCS = 0:

This bit is ignored.

When TCS = 1:

Value	Description
1	Synchronizes the external clock input.
0	Does not synchronize the external clock input.

Bit 1 – TCS Timer Clock Source Select bit[(1)]

See Table 27-2.

27.3.2. Timer x Counter Register

Name: TMRx
Offset: 0x1CE4, 0x1CF4, 0x1D04

Bit	31	30	29	28	27	26	25	24
	TMR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TMR[31:0] Timer Value bits

27.3.3. Period Register 1

Name: PRx
Offset: 0x1CE8, 0x1CF8, 0x1D08

Bit	31	30	29	28	27	26	25	24
	PR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1
Bit	23	22	21	20	19	18	17	16
	PR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1
Bit	15	14	13	12	11	10	9	8
	PR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1
Bit	7	6	5	4	3	2	1	0
	PR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 31:0 – PR[31:0] Period Register bits

27.4. Operation

The timer modules found in dsPIC33A family devices support the following modes of operation:

- Synchronous Clock Counter
- Synchronous External Clock Counter
- Gated Timer
- Asynchronous External Counter

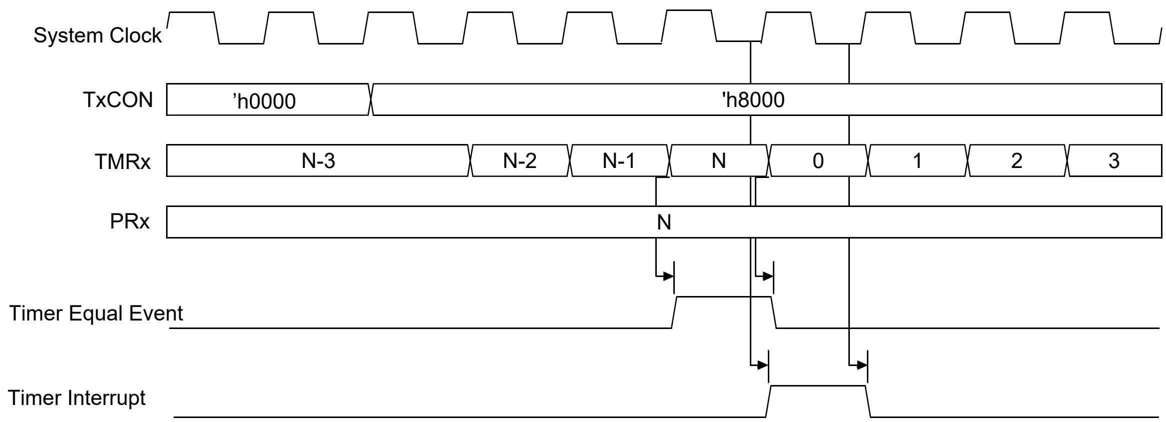
The Timer modes are determined by the following bits:

- TCS (T1CON[1]): Timer Clock Source Select bit
- TGATE (T1CON[7]): Timer Gated Time Accumulation Enable bit
- TSYNC (T1CON[2]): Timer External Clock Input Synchronization Selection bit

27.4.1. Synchronous Clock Counter Mode

When TxCON.TCS bit = `0` and TxCON.TON bit = `1` in the TxCON register, the timer increments on every rising edge of the system clock up to a match value, preloaded into the period register PRx, then rolls over and continues. The use of period registers allows for any timer value to be reached as the maximum while providing a specific period/time interval to be repeated with no firmware intervention (refer to Figure 27-2).

Figure 27-2. Synchronous Clock Counter Mode Timing Diagram (1:1 Prescale)



For the maximum timer period, load the period register with 0xFFFF_FFFF. This incrementing sequence repeats until the timer is disabled by being turned off (TxCON.TON = `0`), stopped in Idle (TxCON.SIDL bit = `1`) or the System Clock is turned off (Sleep mode). The timer count is not reset when the module is turned off.

When the CPU goes into Sleep mode or Idle mode with SIDL (TxCON.SIDL = `1`), the timer will stop incrementing. The timer module logic will resume the incrementing sequence upon termination of the CPU Idle or Sleep mode. If the CPU goes into Idle mode with TxCON.SIDL bit = `0`, the timer will continue to operate normally.

27.4.1.1. Synchronous Clock Counter Considerations

The user may write the timer register to initialize the timer. Since the timer is configured to increment from the internal system clock, the module's interrupt output will be synchronous to the rising edge of internal system clock and no interrupt latency will be added by the module.

27.4.1.2. Synchronous Counter Initialization Steps

The following steps must be performed to configure the timer for Synchronous Counter mode:

1. Clear the TON control bit (TxCON[15] = 0) to disable the timer.
2. Clear the TCS control bit (TxCON[1] = 0) to select the internal system clock source.
3. Select the desired timer input clock prescale.
4. Load/clear the Timer register, TMRx.
5. Load the Timer Period register, PRx, with the desired 32-bit match value.
6. If interrupts are used:
   a. Clear the TxIF Interrupt Flag bit in the IFSx register.
   b. Configure the Interrupt Priority Levels in the IPCx register.
   c. Set the TxIE Interrupt Enable bit in the IECx register.
7. Set the TON control bit (T1CON[15] = 1) to enable the timer.

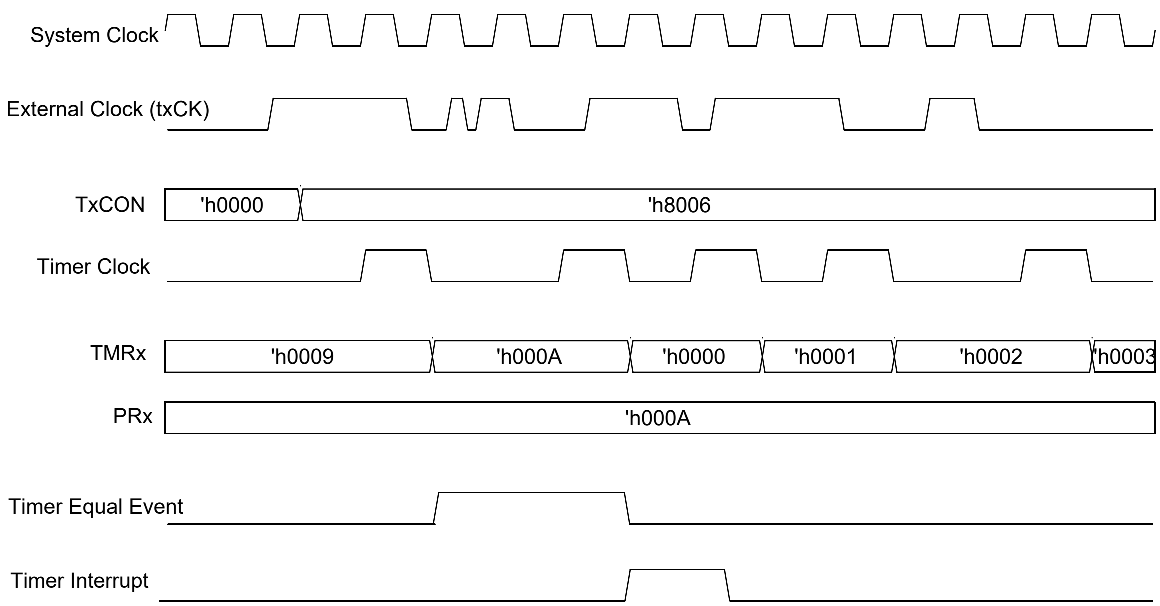
Example 27-1. Synchronous Counter Example Code

```
T1CON = 0x0;             // Stop timer and clear control register,
// set prescaler at 1:1, internal clock source
TMR1 = 0x0;                      // Clear timer register
PR1 = 0xFFFFFFFF;               // Load period register
T1CONbits.ON = 1;              // Start timer
```

27.4.2. Synchronous External Clock Counter Mode

When TxCON.TCS = 1, TxCON.TON = 1 and TxCON.TSYNC = 1, the timer increments on the synchronized rising edge of the applied external clock signal. The synchronization of the input signal occurs with the internal system clock signal. The timer counts up to a match value preloaded in the period register, then resets and continues. This incrementing sequence repeats until the timer is disabled (refer to Figure 27-3).

Figure 27-3. Synchronized External Clock Mode Timing Diagram



When the CPU goes into Sleep mode, or when the timer is configured for the Synchronous mode of operation and the CPU goes into Idle mode with TxCON.SIDL = 1, the timer will stop incrementing.

The timer module logic will resume the incrementing sequence upon termination of the CPU Idle/Sleep mode. If TxCON.SIDL = 0, the timer will continue to operate.

27.4.2.1. Synchronous External Clock Counter Considerations

In Synchronous External Clock Counter mode, the externally supplied clock must be half the internal system clock frequency or slower. Additionally, the external clock high and low times must be at least one full internal system clock period each for reliable operation. If the external clock glitches (positive or negative) or operates faster than half the peripheral clock, the timer may or may not increment. In no case will the timer increment faster than half the internal system clock.

When the timer is configured to increment from the synchronized external clock, the interrupt latency will be '0', relative to the TMRx rollover (transition from TMRx = PRx to '0'). As the timer will be rolling over from a synchronized clock, the interrupt output will be synchronous to the internal system clock rising edge.

When TxCON.TCS = 1, TxCON.ON = 1, TxCON.TCKPS[1:0] ≠ 00 and TxCON.TSYNC = 1, the timer increments on the synchronized rising edge of the prescaler output. The synchronization of the input signal occurs with the internal system clock signal prior to the prescaler. The timer counts up to a match value preloaded in the period register, then resets and continues. This incrementing sequence repeats until the timer is disabled.

27.4.2.2. Synchronous External Counter Initialization Steps

The following steps must be performed to configure the timer for Synchronous Counter mode:

1. Clear the TON Control bit (TxCON[15] = 0) to disable the timer.

2. Set the TCS Control bit (TxCON[1] = 1) to enable the TxCK pin clock selection.
3. Set the TSYNC Control bit (TxCON[2] = 1) to enable TxCK pin clock synchronization.
4. Select the desired timer input clock prescale using TCKPS[1:0] bits in TxCON register.
5. Load/clear the Timer register, TMRx.
6. Load the Timer Period register, PRx, with the desired 32-bit match value.
7. If interrupts are used:
   a. Clear the TxIF Interrupt Flag bit in the IFSx register.
   b. Configure the Interrupt Priority Levels in the IPCx register.
   c. Set the TxIE Interrupt Enable bit in the IECx register.
8. Set the ON Control bit (TxCON[15] = 1) to enable the timer.

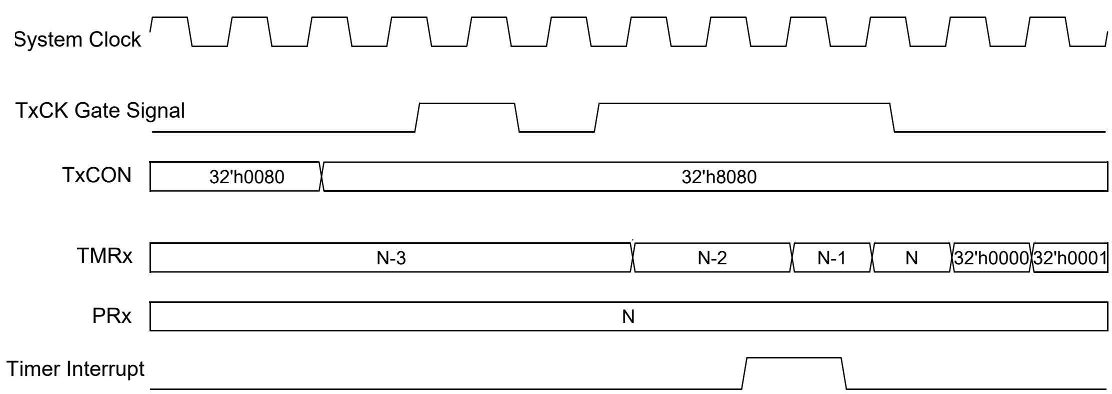
Example 27-2. Synchronous External Counter Example Code

```
T1CON = 0x0;        // Stop timer and clear control register
T1CON = 0x06;              // Set prescaler at 1:1, external clock source
TMR1 = 0x0;                // Clear timer register
PR1 = 0xFFFFFFFF;            // Load period register
T1CONbits.ON = 1;            // Start timer
```

27.4.3. Gated Timer Mode

The timer can be placed in Gated Time Accumulation Operational mode to enable the timer clock source when the gate signal is asserted high. The Timer Control bit TxCON.TGATE must be set to enable this mode. The timer must be enabled, TxCON.TON = 1 (refer to Figure 27-4).

Figure 27-4. Timer Gate Mode Timing Diagram



When configured for this mode, the gate operation starts on a rising edge of the signal applied to the TxCK input and terminates on the falling edge of the signal applied to the TxCk input. The timer will increment while the external gate signal is high. The gate signal must have a minimum high and low time greater than the timer input clock period to ensure that the TxCK input will have sufficient setup time to be sampled by the rising edge of the clock. The edges of the gate signal may occur asynchronously to the timer clock source.

The falling edge of the gate signal generates an interrupt. The latency of the interrupt is one to two time clock cycles after the falling edge. If TGATE = 1, TxCK = 0 and TON is set, an interrupt will be generated despite no falling edge existing on the gate input. The falling edge of the TxCK terminates the count operation but does not reset the timer. The user must reset the timer if desiring to start from zero.

27.4.3.1. Gated Timer Mode Clock Considerations

The timer may be using the prescaler for a slower timer increment rate. Note that the gated timer clock is routed through the prescaler. When using the prescaler, the prescaler will simply retain its current count value as the gate is low. Note that if the user writes the TMRx[15:0] register, the prescaler will be reset. Using the prescaler will not affect the function or timing of the falling edge interrupt.

When the CPU goes into Sleep mode or Idle mode with TxCON.SIDL = `1`, the timer will stop incrementing. The timer module logic will resume the incrementing sequence upon termination of the CPU Idle/Sleep mode. If TxCON.SIDL = `0`, the timer will continue to operate normally.

The period matching function remains operational in Gate mode operation. If the timer matches the period, the timer will reset, but the period match does not generate an interrupt.

Note: Gated Timer mode is overridden if the Timer Clock Source Select bit (TCS) is set to an external clock source, TCS = `1`. For Gated Timer mode operation, the internal clock source must be selected (TCS = 0).

27.4.3.2. Gated Timer Initialization Steps

The following steps must be performed to configure the timer for Gated Timer mode:

1. Clear the TON Control bit (TxCON[15] = 0) to disable the timer.
2. Clear the TCS Control bit (TxCON[1] = 0) to select the internal system clock source.
3. Set the TGATE Control bit (TxCON[7] = 1) to enable Gated Timer mode.
4. Select the desired prescaler.
5. Clear the Timer register, TMRx.
6. Load the Timer Period register, PRx, with the desired 32-bit match value.
7. If interrupts are used:
   a. Clear the TxIF Interrupt Flag bit in the IFSx register.
   b. Configure the Interrupt Priority Levels in the IPCx register.
   c. Set the TxIE Interrupt Enable bit in the IECx register.

8. Set the TON Control bit (TxCON[15] = 1) to enable the timer.

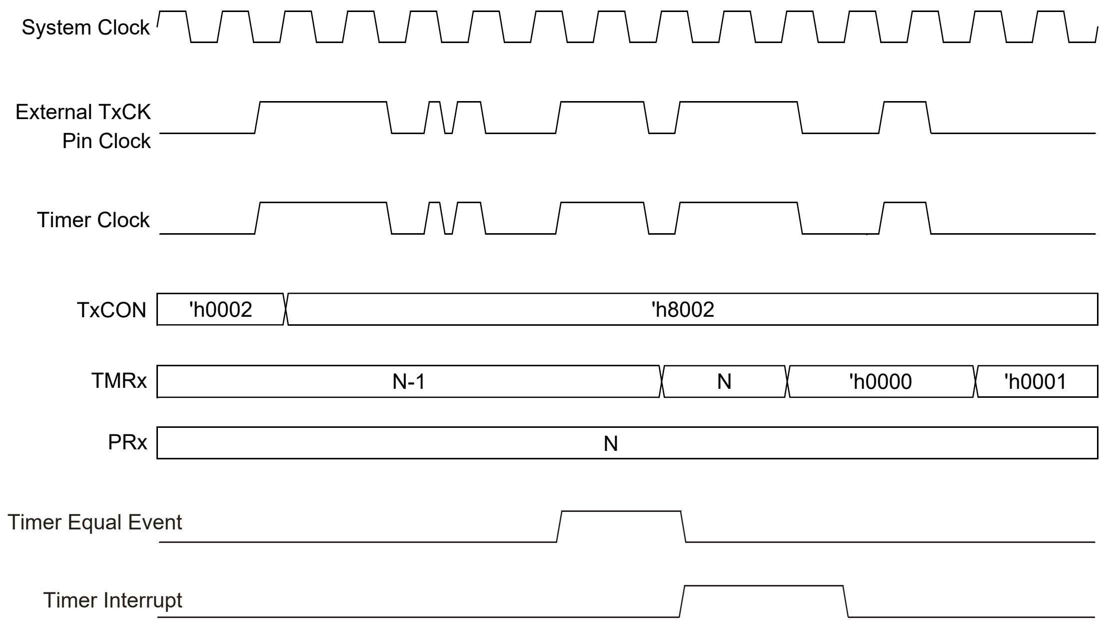
Example 27-3. Gated Timer Example Code

```
T1CON = 0x0;    // Stop timer and clear control register
T1CON = 0x00000080;      // Gated Timer mode, prescaler at 1:1,
internal clock source
TMR1 = 0x0;              // Clear timer register
PR1 = 0xFFFFFFFF;        // Load period register with 32-bit match value
T1CONbits.ON = 1;        // Start timer
```

27.4.4. Asynchronous Clock Counter Mode

When TxCON.TCS = `1`, TxCON.ON = `1` and TxCON.TSYNC = `0`, the timer increments on every rising edge of the applied external clock signal. The timer counts up to a match value preloaded in the respective period register, then rolls over and continues. This incrementing sequence repeats until the timer is disabled (refer to Figure 27-5).

Figure 27-5. Asynchronous External Clock Mode Timing Diagram



When the timer is configured for asynchronous operation and the CPU goes into Idle mode with TxCON.SIDL = `1` or Sleep, the timer continues incrementing.

27.4.5. Asynchronous Mode TMRx Read and Write Operations

Two bits in the TxCON register can be used to ensure that writes during the timer operation will not cause the timer value to be corrupted. The TxCON.TMWDIS bit, when set, will prevent a write to the Timer and Period registers when a previous write to the timer is awaiting synchronization into the asynchronous timer clock domain. The TxCON.TMWIP and TxCON.PRWIP bits indicate when write synchronization is complete, and it is safe to write another value to the Timer register and Period register. These bits have no effect in Synchronous Clock Counter modes.

The Timer Write in Progress bit (TWIP) provides two options for safely writing to the TMR Count register while the Timer is enabled.

Option 1 is the legacy Timer Write mode, TWDIS bit = `0`. To determine when it is safe to write to the TMRx Count register, it is recommended to poll the TWIP bit. When TWIP = `0`, it is safe to perform the next write operation to the TMR1 Count register. When TWIP = `1`, the previous write operation to the TMRx Count register is still being synchronized and any additional write operations should wait until TWIP = `0`.

Option 2 is the new Synchronized Timer Write mode, TWDIS bit = `1`. A write to the TMRx Count register can be performed at any time. However, if the previous write operation to the TMRx Count register is still being synchronized, any additional write operations are ignored.

Note: A write to the TMRx Count register must be performed prior to configuring Timer for Asynchronous mode if the proper procedure to check TWIP and TWDIS is not followed.

27.4.6. Asynchronous Clock Counter Considerations

The asynchronous TMRx value is synchronized before it is read. This allows the user to read a clean timer value; however, this also means the read data is the value of TMRx from two system clock cycles prior to the read. This is also the value stored into the TMRx register when the module is turned off.

When the timer is configured in Asynchronous mode, the act of setting the TxCON.ON bit does not take effect until two rising edges of the external clock input have occurred.

When the timer is configured in Asynchronous mode and the timer is running, a write to TMRx must be synchronized for two asynchronous timer clock pulses before the contents of the Timer register are updated with the timer value. The timer will not increment until two timer clocks have transpired. Two clock cycles are required to synchronize the write data into the clock domain of the asynchronous timer.

When the timer is configured for an asynchronous external clock, the interrupt latency is one asynchronous timer clock period plus two system clock cycles.

27.4.7. Asynchronous External Clock Counter Initialization Steps

The following steps must be performed to configure the timer for 32-bit Asynchronous Counter mode:

1. Clear the TON Control bit (TxCON[15] = 0) to disable the timer.
2. Set the TCS bit (TxCON[1] = 1) to enable external clock selection.
3. Clear the TSYNC Control bit (TxCON[2] = 0) to disable clock synchronization.
4. Select the desired prescaler.
5. Load/clear the Timer register, TMRx.
6. If using period match, load the Timer Period register, PRx, with the desired 32-bit match value.
7. If interrupts are used:
   a. Clear the TxIF Interrupt Flag bit in the IFSx register.
   b. Configure the interrupt priority levels in the IPCx register.
   c. Set the TxIE Interrupt Enable bit in the IECx register.
8. Set the TON Control bit (TxCON[15] = 1) to enable the timer.

Example 27-4. 32-bit Asynchronous Counter Mode Code

```
/*    32-bit Asynchronous Counter Mode Example */
T1CON = 0x0;                         // Stops the Timer and resets the control
register
TMR1 = 0x0;                          // Clear timer register
T1CON = 0x02;                  // Set prescaler 1:1, external clock,asynchronous
mode
PR1 = 0xFFFFFFFF;                    // Load period register
T1CONbits.ON = 1;                    // Start timer
```

27.4.8. Timer Prescalers

Timers provide input clock (peripheral clock or external clock) prescale options of 1:1, 1:8, 1:64 and 1:256, which can be selected by using the TCKPS[1:0] bits (TxCON[5:4]).

The prescaler counter is cleared when any of the following occurs:

- A write to the TMRx register
- Disabling the Timer TON bit (TxCON[15] = 0)
- Any device Reset

27.4.8.1. Period Match and Slow Clock

When using the prescaler, the user software can change the value of the period register faster than one timer period.

Figure 27-8 shows that when the counter is operating with an 8x prescale and PRx is written before the end of the prescaler rollover, a period match is not detected and an interrupt is not generated. In this case, the timer will continue running until a compare match occurs between TMRx and the new value *M*. If *M* is lower than the original period of *N*, the timer will run all the way to 0xFFFF_FFFF and roll over to 0x0000_0000.

Figure 27-6. Synchronous Clock with Prescale with Write to PRx Following Match (TxCON.TCS = 0, TxCON.TCKPS[1:0] =01)

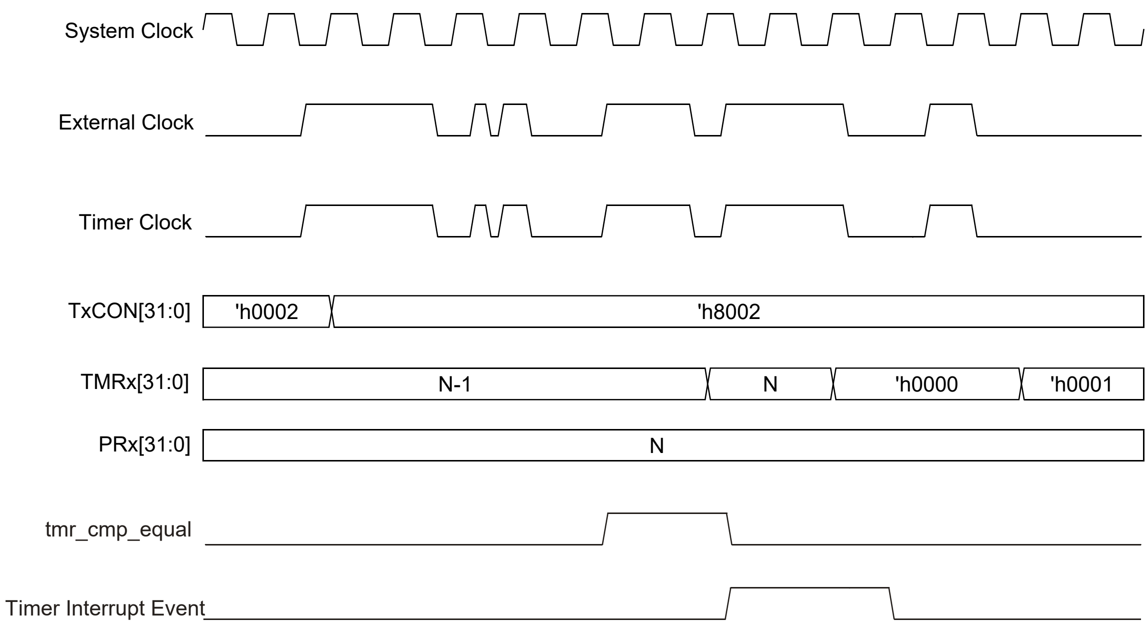


Figure 27-7. Synchronized External Clock with 8:1 Prescale Timing Diagram

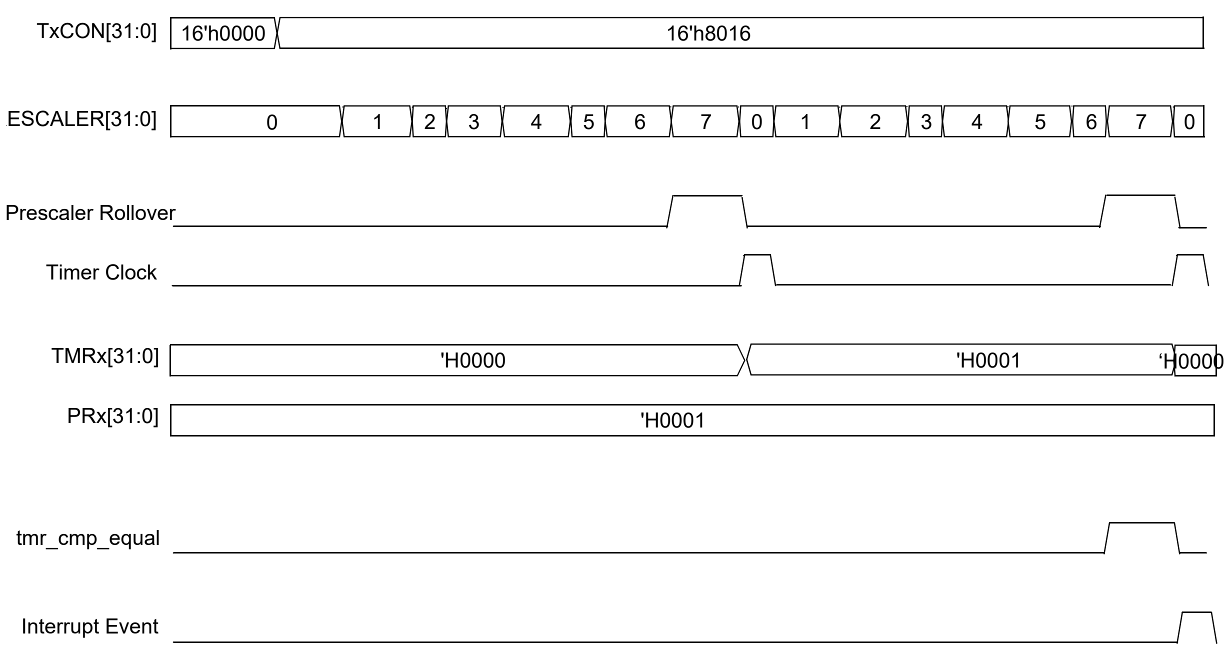
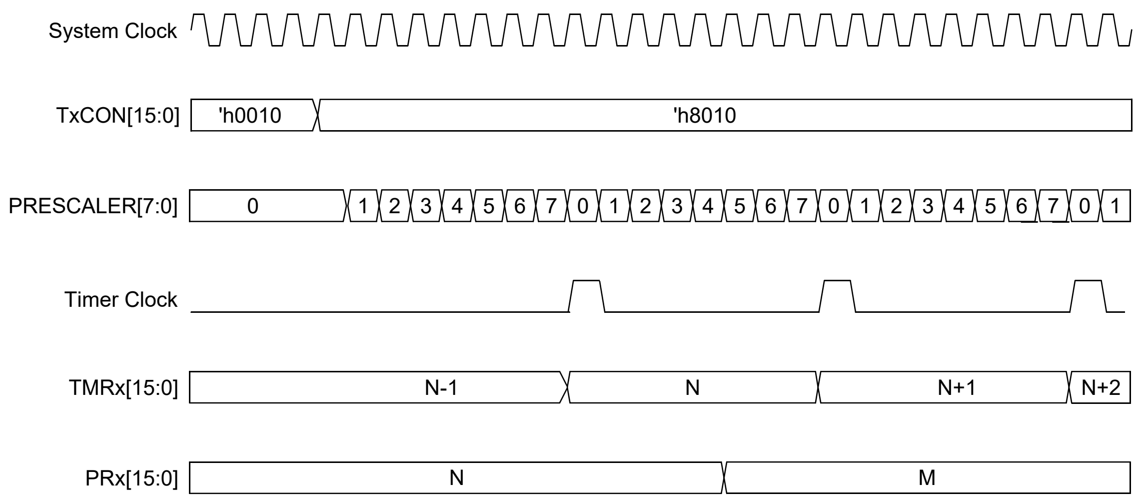


Figure 27-8. Synchronous Clock with Prescale with Write to PRx Following Match (TxCON.TCS = 0, TxCON.TCKPS[1:0] =01)



27.4.9. Writing to TxCON, TMRx and PRx Registers

The Timer module is disabled and powered off when the TON bit (TxCON[15]) = `0`, thus providing maximum power savings.

The timer register can be written while the module is operating. The bus write always has priority over the timer increment.

If 0xFFFFFFFF is written into the timer register, the next timer count clock after this write will cause the timer to roll over to 0x00000000.

To prevent unpredictable timer behavior, it is recommended that the timer be disabled before writing to any of the TxCON register bits or the timer input clock prescaler. Attempting to set the TON bit = 1 and writing to any TxCON register bits in the same instruction may cause an erroneous timer operation.

The PRx Period register can be written to while the module is operating. However, to prevent unintended period matches, writing to the PRx Period register while the timer is enabled (TON bit = `1`) is not recommended.

The user must write the TxCON register to establish the operating mode prior to any updates to the TMRx register. The user is allowed to write the TMRx register while the timer is running. Writes to TMRx while the timer is running require the following synchronization sequence to be completed:

- Three timer domain clocks: two to sync the written data and TMRWIP bit and one to perform the

write into TMRx.

- An additional two system clocks after the write completes are needed to sync TMRx to the system domain

If the user attempts to write the timer again while the current write is awaiting synchronization, the value written to the timer can be corrupted.

The TMRx Count register is not reset to zero when the module is disabled.

For read write operation on timer in external asynchronous mode refer to Asynchronous Clock Counter Mode.

Two bits in the TxCON register can be used to ensure that writes during timer operation will not cause the timer value to be corrupted. The TxCON.TMWDIS bit, when set, will prevent a write to the timer and period registers when a previous write to the timer is awaiting synchronization into the asynchronous timer clock domain. The TxCON.TMWIP bit indicates when write synchronization is complete and it is safe to write another value to the timer.

In Asynchronous mode, writes are synchronized once the high byte is written, at which point the TxCON.PRWIP bit is set.

The TxCON.TMWDIS bit, when set, will prevent a write to the timer or period registers when a previous write to that register is awaiting synchronization into the asynchronous timer clock domain. The TxCON.TMWIP and TxCON.PRWIP bits indicate when write synchronization is complete and it is safe to write another value to the timer.

In synchronous mode, writes have immediate effect upon write to registers PRx and TMRx. TxCON Status bits (TMWIP & PRWIP) will not update in synchronous mode. The TxCON.TMWDIS bit has no effect in synchronous mode.

Note: If a write to TMR is completed (and TMWIP cleared) on the same cycle where TMR = PR, an interrupt will still be generated and the timer will still read as '0' on the subsequent cycle due to the timing of the synchronization logic. Customer code that writes to the asynchronous timer while it is running should be capable of ignoring the unintended interrupt if necessary.

27.5. Interrupts

The Timer module has the ability to generate an interrupt on a period match or falling edge of the external gate signal, depending on the operating mode.

The TxIF bit is set when one of the following conditions is true:

- When the TMRx count matches the respective PRx register and the Timer module is not operating in Gated Time Accumulation mode.
- When the falling edge of the gate signal is detected while the timer is operating in Gated Time Accumulation mode.

The TxIF bit must be cleared in software.

The Timer module is enabled as a source of interrupt via the Timer Interrupt Enable bit, TxIE. The Timer Interrupt Priority Level bits, TxIP[2:0], define the priority group to which the interrupt source will be assigned.

Note: A special case occurs when the PRx register is loaded with '0' and the timer is enabled. An interrupt is not generated for this configuration. A Falling Edge Gate signal will not wake up the device from Sleep or Idle.

27.5.1. Interrupt Configuration

The Timer module has a dedicated Interrupt Flag bit (TxIF) and a corresponding Interrupt Enable Mask bit (TxIE).

The TxIF bit is set when the timer count matches the respective period register and the timer module is not operational in Gated Time Accumulation mode. This bit is also set if the falling edge of the gate signal is detected when the timer is operating in Gated Time Accumulation mode. The TxIF bit is set, regardless of the state of the corresponding TxIE bit. If required, the TxIF bit can be polled by software.

The TxIE bit is used to define the behavior of the interrupt controller when the TxIF bit is set. When the TxIE bit is clear, the interrupt controller does not generate a CPU interrupt for the event. If the TxIE bit is set, the interrupt controller generates an interrupt to the CPU when the T1IF bit is set (subject to the interrupt priority).

It is the responsibility of the user's software routine that services a particular interrupt to clear the appropriate interrupt flag bit before the service routine is complete.

The priority of the Timer module can be set with the TxIP[2:0] bits. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of seven (the highest priority) to a value of zero (which does not generate an interrupt). An interrupt being serviced will be preempted by an interrupt in a higher priority group.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The user's code at this vector address should perform any application-specific operations and clear the TxIF interrupt flag and then exit.

27.6. Power-Saving Modes

27.6.1. Timer Operation in Sleep Mode

As the device enters Sleep mode, the system clock and peripheral clock are disabled.

The Timer can operate asynchronously from an external clock source. Therefore, the Timer module can continue to operate during Sleep mode.

To operate in Sleep mode, the Timer module is configured as follows:

- Timer module is enabled, TON bit (TxCON[15]) = `1`.
- Timer clock source is selected as external, TCS bit (TxCON[1]) = `1`.
- TSYNC bit (TxCON[2]) is set to a logic '`0`' (Asynchronous Counter mode enabled).

When these conditions are met, the timer continues to count and detect period matches when the device is in Sleep mode. When a match between the timer and the period register occurs, the TxIF status bit is set. If the TxIE bit is set, and its priority is greater than the current CPU priority, the device wakes from Sleep or Idle mode and executes the Timer Interrupt Service Routine.

If the assigned priority level of the timer interrupt is less than or equal to the current CPU priority level, the CPU is not awakened and the device enters Idle mode.

27.6.2. Timer Operation in Idle Mode

When the device enters Idle mode, the system clock sources remain functional and the CPU stops executing code. The Timer module can optionally continue to operate in Idle mode.

The setting of the SIDL bit (TxCON[13]) determines whether the Timer module stops in Idle mode or continues to operate normally. If SIDL = `0`, the Timer Module continues operation in Idle mode. If SIDL = `1`, the Timer module stops in Idle mode.

27.7. Effects of Various Resets

27.7.1. Device Reset

All Timer registers are forced to their Reset states on a device Reset.

27.7.2. Power-on Reset (POR)

All Timer registers are forced to their Reset states on a Power-on Reset (POR).

27.7.3. Watchdog Timer Reset

All Timer registers are forced to their Reset states on a Watchdog Timer Reset.

28. Capture/Compare/PWM/Timer Modules (SCCP/MCCP)

This section describes the features and operation of the Single and Multiple Output CCP module. The module has the following features:

- Combines Time Base, Input Capture, Compare and PWM Functions into a Single Peripheral.
- Provides Better Edge Resolution in PWM Mode.
- Provides the Required Functionality in the PWM Mode to Support a Selected Range of Motor Control, Power Supply and Lighting Applications.

The following module features are covered in this section:

- General Purpose Timer
- Input Capture
- Output Compare/PWM

There are two different forms of the module, distinguished by the number of PWM outputs that the module can generate. Single output modules (SCCPs) provide only one PWM output. Multiple output modules (MCCPs) can provide up to six outputs and an extended range of output control features, depending on the pin count of the particular device.

All modules (SCCP and MCCP) include these features:

- User-Selectable Clock Inputs, Including System Clock and External Clock Input Pins
- Input Clock Prescaler for Time Base
- Output Postscaler for Module Interrupt Events or Triggers
- Synchronization Output Signal for Coordinating Other MCCP/SCCP Modules with User-Configurable Alternate and Auxiliary Source Options
- Fully Asynchronous Operation in All Modes and in Low-Power Operation
- Special Output Trigger for A/D Conversions
- 16-Bit and 32-Bit General Purpose Timer Modes with Optional Gated Operation for Simple Time Measurements
- Capture Modes:
  - 16-bit or 32-bit capture of time base on external event
  - Up to four-level deep FIFO capture buffer
  - Capture source input multiplexer
  - Gated capture operation to reduce noise-induced false captures
- Output Compare/PWM Modes:
  - Single Edge and Dual Edge Compare modes
  - Center-Aligned Compare mode
  - Variable Frequency Pulse mode
  - External Input mode
- MCCP Modules also Include these Extended PWM Features:
  - Single Output Steerable Mode
  - Brush DC Motor (Forward and Reverse) Modes
  - Half-Bridge with Dead-Time Delay
  - Push-Pull PWM Mode
  - Output Scan Mode

– Auto-Shutdown with Programmable Source and Shutdown State
– Programmable Output Polarity

28.1. Device-Specific Information

Table 28-1. SCCP Summary Table

SCCP Module Instances	MCCP Module Instances	Peripheral Bus Speed	Clock Source
4	1	Standard (1:2 CPU Clock)	See Table 28-2
Note: CCP1-8 are single output CCP modules and CCP9 is a multiple output CCP module.			

Table 28-2. CLKSEL Time Base Clock Select bits

Value	Description
111	TCKIx (External clock)
110-010	Reserved
001	CLKGEN13
000	Standard (1:2 CPU Clock)

Table 28-3. ICS Input Capture Source Select bits

Value	Description
111	CLC4 Out
110	CLC3 Out
101	CLC2 Out
100	CLC1 Out
011	CMP3 Out
010	CMP2 Out
001	CMP1 Out
000	CCP Input Capture x (ICx) pin (PPS)

Table 28-4. Synchronization Sources

SYNC[4:0]	Synchronization Source
11111	Off (0)
11100-11110	Reserved
11011	CMP5 output
11010	CMP4 output
11001	CMP3 output
11000	CMP2 output
10111	CMP1 output
10110	MCCP1 Sync Out
10101	CLC4 Out
10100	CLC3 Out
10011	CLC2 Out
10010	CLC1 Out
10001	INT2
10000	INT1
1111	INT0
1110	PTGO14
1101	PTGO15
1100	PTGO16
1011	PTGO17

Table 28-4. Synchronization Sources (continued)

SYNC[4:0]	Synchronization Source
00101-01010	Reserved
100	SCCP4 Sync Out
11	SCCP3 Sync Out
10	SCCP2 Sync Out
1	SCCP1 Sync Out
0	MCCPn/SCCPn Timer Sync Out

28.2. Architectural Overview

Select dsPIC33A family devices include one or more Capture/Compare/PWM/Timer (CCP) modules. The multipurpose timer module provides the functionality of the comparable Input Capture, Output Compare and General Purpose Timer peripherals found in all other devices.

CCP modules can operate in one of three major modes:

- **Time Base –** The module generates internal signals, triggers and interrupt events only based on a 16-bit or 32-bit count value. All associated I/O functions are disabled. The time-base clock may be gated using one of the auto-shutdown/gating signal sources.
- **Input Capture –** The value of the 16-bit or 32-bit time base is written to a buffer on certain edge events received from a device input pin. Input Capture signal sources may be gated using one of the auto-shutdown/gating signal sources.
- **Output Compare –** A device output pin is set, reset, toggled or pulsed based on register values compared to the 16-bit or 32-bit time base. Register values may be buffered or non-buffered. The output compare signal can be steered to multiple output pins.

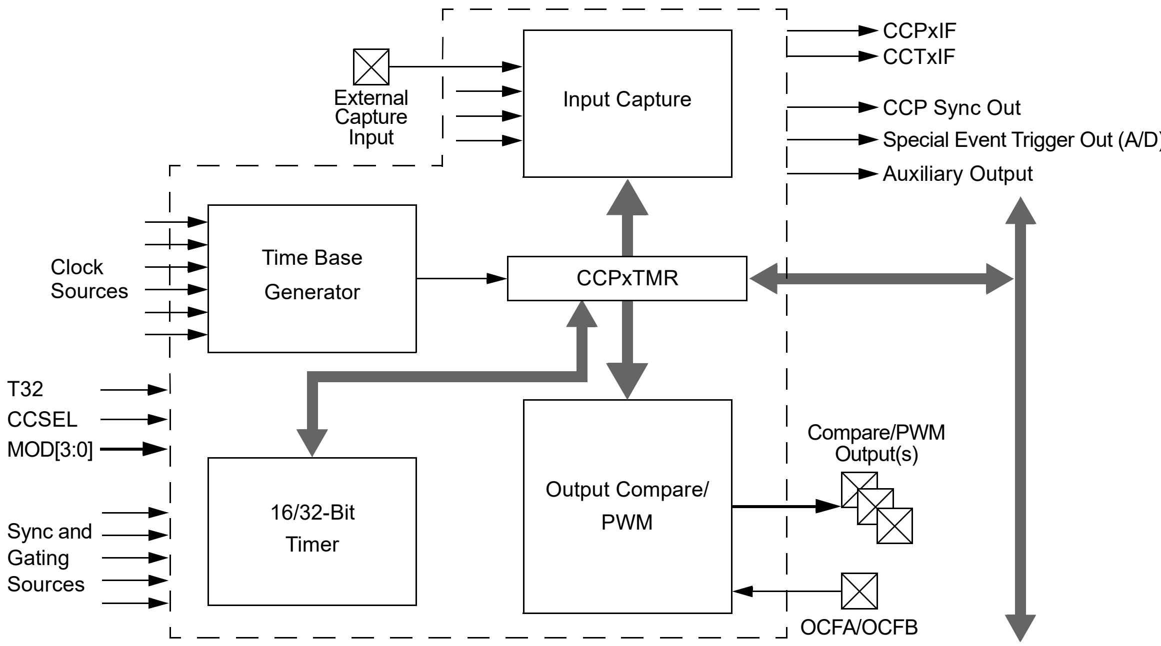
The SCCP module include these features:

- User-Selectable Clock Inputs, Including System Clock and External Clock Input Pins
- Input Clock Prescaler for Time Base
- Output Postscaler for Module Interrupt Events or Triggers
- Synchronization Output Signal for Coordinating Other SCCP Modules with User-Configurable Alternate and Auxiliary Source Options
- Fully Asynchronous Operation in All Modes and in Low-Power Operation
- Special Output Trigger for A/D Conversions
- 16-bit and 32-bit General Purpose Timer Modes with Optional Gated Operation for Simple Time Measurements
- Capture Modes:
  - 16-bit or 32-bit capture of a time base on an external event
  - Up to a four-level deep FIFO capture buffer
  - Capture source input multiplexer
  - Gated Capture operation to reduce noise-induced false captures
- Output Compare/PWM Modes:
  - Single Edge and Dual Edge Compare modes
  - External Input mode

The CCP module can be operated only in one of the three major modes (Capture, Compare or Timer) at any time. The other modes are not available unless the module is reconfigured.

A conceptual block diagram for the module is shown in Figure 28-1. All three modes use the Time-Base Generator and the Common Timer register (CCPxTMR). Other shared hardware components, such as comparators and buffer registers, are activated and used as a particular mode requires.

Figure 28-1. CCP Conceptual Block Diagram



28.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1B00	CCP1CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1B04	CCP1CON2	31:24	OENSYNC		OCFEN	OCEEN	OCDEN	OCCEN	OCBEN	OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1B08	CCP1CON3	31:24	OETRIG	OSCNT[2:0]				OUTM[2:0]		
		23:16			POLACE	POLBDF	PSSACE[1:0]		PSSBDF[1:0]	
		15:8								
		7:0			DT[5:0]					
0x1B0C	CCP1STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1B10	CCP1TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1B14	CCP1PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1B18	CCP1RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1B1C	CCP1RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1B20	CCP1BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x1B24 ... 0x1B2F	Reserved									
0x1B30	CCP2CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1B34	CCP2CON2	31:24	OENSYNC		OCFEN	OCEEN	OCDEN	OCCEN	OCBEN	OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1B38	CCP2CON3	31:24	OETRIG	OSCNT[2:0]				OUTM[2:0]		
		23:16			POLACE	POLBDF	PSSACE[1:0]		PSSBDF[1:0]	
		15:8								
		7:0			DT[5:0]					
0x1B3C	CCP2STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1B40	CCP2TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1B44	CCP2PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1B48	CCP2RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1B4C	CCP2RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1B50	CCP2BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x1B54 ... 0x1B5F	Reserved									
0x1B60	CCP3CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1B64	CCP3CON2	31:24	OENSYNC		OCFEN	OCEEN	OCDEN	OCCEN	OCBEN	OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1B68	CCP3CON3	31:24	OETRIG	OSCNT[2:0]				OUTM[2:0]		
		23:16			POLACE	POLBDF	PSSACE[1:0]		PSSBDF[1:0]	
		15:8								
		7:0			DT[5:0]					
0x1B6C	CCP3STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1B70	CCP3TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1B74	CCP3PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1B78	CCP3RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1B7C	CCP3RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1B80	CCP3BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x1B84 ... 0x1B8F	Reserved									

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1B90	CCP4CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1B94	CCP4CON2	31:24	OENSYNC		OCFEN	OCEEN	OCDEN	OCCEN	OCBEN	OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1B98	CCP4CON3	31:24	OETRIG	OSCNT[2:0]				OUTM[2:0]		
		23:16			POLACE	POLBDF	PSSACE[1:0]		PSSBDF[1:0]	
		15:8								
		7:0			DT[5:0]					
0x1B9C	CCP4STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1BA0	CCP4TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							
0x1BA4	CCP4PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1BA8	CCP4RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1BAC	CCP4RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1BB0	CCP4BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							
0x1BB4 ... 0x1BBF	Reserved									
0x1BC0	CCP5CON1	31:24	OPSSRC	RTRGEN			OPS[3:0]			
		23:16	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
		15:8	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
		7:0	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
0x1BC4	CCP5CON2	31:24	OENSYNC		OCFEN	OCEEN	OCDEN	OCCEN	OCBEN	OCAEN
		23:16	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
		15:8	PWMRSEN	ASDGM		SSDG				
		7:0	ASDG[7:0]							
0x1BC8	CCP5CON3	31:24	OETRIG	OSCNT[2:0]				OUTM[2:0]		
		23:16			POLACE	POLBDF	PSSACE[1:0]		PSSBDF[1:0]	
		15:8								
		7:0			DT[5:0]					
0x1BCC	CCP5STAT	31:24								
		23:16				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
		15:8						ICGARM		
		7:0	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
0x1BD0	CCP5TMR	31:24	TMR[31:24]							
		23:16	TMR[23:16]							
		15:8	TMR[15:8]							
		7:0	TMR[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x1BD4	CCP5PR	31:24	PR[31:24]							
		23:16	PR[23:16]							
		15:8	PR[15:8]							
		7:0	PR[7:0]							
0x1BD8	CCP5RA	31:24								
		23:16								
		15:8	CMPA[15:8]							
		7:0	CMPA[7:0]							
0x1BDC	CCP5RB	31:24								
		23:16								
		15:8	CMPB[15:8]							
		7:0	CMPB[7:0]							
0x1BE0	CCP5BUF	31:24	BUF[31:24]							
		23:16	BUF[23:16]							
		15:8	BUF[15:8]							
		7:0	BUF[7:0]							

28.3.1. CCPx Control Register 1

Name: CCPxCON1
Offset: 0x1B00, 0x1B30, 0x1B60, 0x1B90, 0x1BC0

Notes:
1. The control bit has no function in Input Capture modes.
2. The control bit has no function when TRIGEN = 0.
3. Values greater than '0011' will cause a FIFO buffer overflow in Input Capture mode.
4. See Sync and Triggered Operation for the definition of Sync inputs.
5. 32-bit operation is not available in Dual Edge Output Compare modes.

Bit	31	30	29	28	27	26	25	24
	OPSSRC	RTRGEN			OPS[3:0]			
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TRIGEN	ONESHOT	ALTSYNC	SYNC[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	SLPEN	TMRSYNC	CLKSEL[2:0]		
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TMRPS[1:0]		T32	CCSEL	MOD[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – OPSSRC Output Postscaler Source Select bit[1]

Value	Description
1	Output postscaler scales module trigger output events.
0	Output postscaler scales time base interrupt events.

Bit 30 – RTRGEN Retrigger Enable bit[2]

Value	Description
1	Time base can be retriggered when CCPTRIG (CCPxSTAT[7]) = 1.
0	Time base may not be retriggered when CCPTRIG (CCPxSTAT[7]) = 1.

Bits 27:24 – OPS[3:0] Time Base Interrupt/Input Capture Event Postscale Select bits[3]

Value	Description
1111	Interrupts the CPU every 16th time base period match.
1110	Interrupts the CPU every 15th time base period match.
...	
0100	Interrupts the CPU every 5th time base period match.
0011	Interrupts the CPU every 4th time base period match or after four input capture events.
0010	Interrupts the CPU every 3rd time base period match or after three input capture events.
0001	Interrupts the CPU every 2nd time base period match or after two input capture events.

Value	Description
0000	Interrupts the CPU after each time base period match or each input capture event.

Bit 23 – TRIGEN CCPx Trigger Enable bit

Value	Description
1	Trigger operation of time base is enabled.
0	Trigger operation of time base is disabled.

Bit 22 – ONESHOT One-Shot Mode Enable bit

Value	Description
1	One-Shot Trigger mode is enabled; trigger duration set by the OSCNT[2:0] (CCPxCON3[30:28]) bits.
0	One-Shot Trigger mode is disabled.

Bit 21 – ALTSYNC Synchronization Output Select bit

Value	Description
1	An alternate signal is used as the module synchronization output signal (see Table 28-10).
0	The module synchronization output signal is the time base reset/rollover event.

Bits 20:16 – SYNC[4:0] Capture/Compare/PWM Synchronization Source Select bits(4)

Value	Description
11111	Time base is in the Free-Running mode and rolls over at FFFF.
11110	Time base is synchronized to source #30.
. . .	
00001	Time base is synchronized to source #1.
00000	Time base is self-synchronized and rolls over at FFFF or matches with the Period register.

Bit 15 – ON Module Enable bit

Value	Description
1	Module is enabled with the operating mode specified by the MOD[3:0] control bits.
0	Module is disabled.

Bit 13 – SIDL Stop in Idle Mode bit

Value	Description
1	Discontinues module operation when device enters Idle mode.
0	Continues module operation in Idle mode.

Bit 12 – SLPEN Sleep Mode Enable bit

Value	Description
1	Module continues to operate in Sleep modes.
0	Module does not operate in Sleep modes.

Bit 11 – TMRSYNC Time Base Clock Synchronization bit

Value	Description
1	Module time base clock is synchronized to internal system clocks; timing restrictions apply.
0	Module time base clock is not synchronized to internal system clocks.

Bits 10:8 – CLKSEL[2:0] Time Base Clock Select bits

Bits 7:6 – TMRPS[1:0] Capture/Compare/PWMx Time Base Prescale Select bits

Value	Description
11	1:64 prescaler
10	1:16 prescaler
01	1:4 prescaler
00	1:1 prescaler

Bit 5 – T32 32-Bit Time Base Select bit(1,5)

Value	Description
1	Uses a 32-bit time base for selected Timer, Single Edge Output Compare or Input Capture function.
0	Uses a 16-bit time base for selected Timer, Single Edge Output Compare or Input Capture function.

Bit 4 – CCSEL Capture/Compare Mode Select bit(1)

Value	Description
1	Module operates as an Input Capture peripheral.
0	Module operates as an Output Compare peripheral.

Bits 3:0 – MOD[3:0] CCP Mode Select bits(1)

Value	Description
CCM	1 (Input Capture modes)
1xxx	Reserved
0111	Reserved
0110	Reserved
0101	Capture every 16th rising edge
0100	Capture every 4th rising edge
0011	Capture every rising and falling edge
0010	Capture every falling edge
0001	Capture every rising edge
0000	Capture every rising and falling edge (Edge Detect mode)
CCM	0 (Output Compare modes)
1111	External Input mode, generator disabled; source selected by the ICS[2:0] bits
1110	Reserved
1101	Reserved
1100	Reserved
1011	Reserved
1010	Reserved
1001	Reserved
1000	Reserved
0111	Reserved
0110	Reserved
0101	Dual Edge Compare mode – Buffered
0100	Dual Edge Compare mode
0011	16-Bit/32-Bit Single Edge mode – toggle output on compare match
0010	16-Bit/32-Bit Single Edge mode – drive output low on compare match
0001	16-Bit/32-Bit Single Edge mode – drive output high on compare match
0000	16-Bit/32-Bit Timer mode – output functions disabled

28.3.2. CCPx Control Register 2

Name: CCPxCON2
Offset: 0x1B04, 0x1B34, 0x1B64, 0x1B94, 0x1BC4

Notes:
1. This bit has no effect in Timer modes, Output Compare modes or PWM modes. A write to the ICGARM (CCPxSTAT[10]) bit will re-arm the one-shot gating circuit when ICGSM = `01` or ICGSM = `10`.
2. This bit has no effect on timer gating or Input Capture gating functions.
3. These bits are implemented in MCCP only.

Bit	31	30	29	28	27	26	25	24
	OENSYNC		OCFEN	OCEEN	OCDEN	OCCEN	OCBEN	OCAEN
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	1

Bit	23	22	21	20	19	18	17	16
	ICGSM[1:0]			AUXOUT[1:0]		ICS[2:0]		
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PWMRSEN	ASDGM		SSDG				
Access	R/W	R/W		R/W				
Reset	0	0		0				

Bit	7	6	5	4	3	2	1	0
	ASDG[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – OENSYNC Output Enable Synchronization bit

Value	Description
`1`	Updates by output enable bits occur on the next time base Reset or rollover.
`0`	Updates by output enable bits occur immediately.

Bit 29 – OCFEN Output Enable/Steering Control bit(3)

Value	Description
`1`	OCx pin is controlled by the CCP module and produces an Output Compare or PWM signal.
`0`	OCx pin is not controlled by the CCP module; the pin is available to the port logic or another peripheral multiplexed on the pin.

Bit 28 – OCEEN Output Enable/Steering Control bit(3)

Value	Description
`1`	OCx pin is controlled by the CCP module and produces an Output Compare or PWM signal.
`0`	OCx pin is not controlled by the CCP module; the pin is available to the port logic or another peripheral multiplexed on the pin.

Bit 27 – OCDEN Output Enable/Steering Control bit(3)

Value	Description
1	OCx pin is controlled by the CCP module and produces an Output Compare or PWM signal.
0	OCx pin is not controlled by the CCP module; the pin is available to the port logic or another peripheral multiplexed on the pin.

Bit 26 – OCCEN Output Enable/Steering Control bit(3)

Value	Description
1	OCx pin is controlled by the CCP module and produces an Output Compare or PWM signal.
0	OCx pin is not controlled by the CCP module; the pin is available to the port logic or another peripheral multiplexed on the pin.

Bit 25 – OCBEN Output Enable/Steering Control bit

Value	Description
1	OCx pin is controlled by the CCP module and produces an Output Compare or PWM signal.
0	OCx pin is not controlled by the CCP module; the pin is available to the port logic or another peripheral multiplexed on the pin.

Bit 24 – OCAEN Output Enable/Steering Control bit

Value	Description
1	OCx pin is controlled by the CCP module and produces an Output Compare or PWM signal.
0	OCx pin is not controlled by the CCP module; the pin is available to the port logic or another peripheral multiplexed on the pin.

Bits 23:22 – ICGSM[1:0] Input Capture Gating Source Mode Control bits(1)

Value	Description
11	Reserved
10	One-Shot mode; a falling edge from the gating source will disable future capture events (ICDIS = 1).
01	One-Shot mode; a rising edge from the gating source will enable future capture events (ICDIS = 0).
00	Level Sensitive mode; a high level from the gating source will enable future capture events; a low level will disable future capture events.

Bits 20:19 – AUXOUT[1:0] Auxiliary Output Signal Selection bits

Value	Description
11	Signal output depends on module operating mode (see Table 28-11).
10	Signal output depends on module operating mode (see Table 28-11).
01	Signal output depends on module operating mode (see Table 28-11).
00	No signal output on aux_out.

Bits 18:16 – ICS[2:0] Input Capture Source Select bits

See Table 28-3.

Bit 15 – PWMRSEN CCPx Output Compare Restart Enable bit

Value	Description
1	ASEVT (CCPxSTAT[4]) bit clears automatically at the beginning of the next Output Compare period, after the shutdown input has ended.
0	ASEVT (CCPxSTAT[4]) bit must be cleared in software to resume Output Compare activity on output pins.

Bit 14 – ASDGM CCPx Auto-Shutdown/Gate Control bit(1)

Value	Description
1	Wait until the next time base Reset or rollover for an Output Compare pin shutdown to occur.
0	Output Compare pin shutdown event occurs immediately.

Bit 12 – SSDG CCPx Software Shutdown/Gate Control bit

Value	Description
1	Manually force auto-shutdown, timer clock gate or Input Capture signal gate event (setting of ASDGM bit still applies).
0	Normal module operation.

Bits 7:0 – ASDG[7:0] CCPx Auto-Shutdown/Gating Source Enable bits
Refer to Table 28-9 for Auto-Shutdown and Gating Sources.

Value	Description
1	ASDG source *n* is enabled.
0	ASDG source *n* is disabled.

28.3.3. CCPx Control Register 3

Name: CCPxCON3
Offset: 0x1B08, 0x1B38, 0x1B68, 0x1B98, 0x1BC8

Notes:
1. ONESHOT (CCPxCON1[22]) must be set for the OSCNT[2:0] bits to be effective.
2. These bits are implemented in MCCP mode only.

Bit	31	30	29	28	27	26	25	24
	OETRIG	OSCNT[2:0]				OUTM[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit	23	22	21	20	19	18	17	16
			POLACE	POLBDF	PSSACE[1:0]		PSSBDF[1:0]	
Access			R/W	RW	R/W	R/W	RW	RW
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
			DT[5:0]					
Access			RW	RW	RW	RW	RW	RW
Reset			0	0	0	0	0	0

Bit 31 – OETRIG Output Enable on Trigger Control bit

Value	Description
1	For Triggered mode (TRIGEN = 1), the module does not drive enabled output pins until triggered.
0	Normal output pin operation

Bits 30:28 – OSCNT[2:0] One-Shot Count bits[(1)]

Value	Description
111	Extends one-shot trigger event by 7 time base count cycles (8 time base periods total).
110	Extends one-shot trigger event by 6 time base count cycles (7 time base periods total).
101	Extends one-shot trigger event by 5 time base count cycles (6 time base periods total).
100	Extends one-shot trigger event by 4 time base count cycles (4 time base periods total).
011	Extends one-shot trigger event by 3 time base count cycles (4 time base periods total).
010	Extends one-shot trigger event by 2 time base count cycles (3 time base periods total).
001	Extends one-shot trigger event by 1 time base count cycle (2 time base periods total).
000	Does not extend one-shot trigger event.

Bits 26:24 – OUTM[2:0] Output Mode Control bits[(2)]

Value	Description
111-001	Reserved
000	Steerable Single Output mode

Bit 21 – POLACE CCP Output Pin, OCxA, Polarity Control bit

Value	Description
1	Output pin polarity is active-low.
0	Output pin polarity is active-high.

Bit 20 – POLBDF CCP Output Pins OCxB, OCxD and OCxF Polarity Control bit

Value	Description
1	Output pin polarity is active low.
0	Output pin polarity is active high.

Bits 19:18 – PSSACE[1:0] PWM Output Pin, OCxA, Shutdown State Control bits

Value	Description
11	Pins are driven active when a shutdown event occurs.
10	Pins are driven inactive when a shutdown event occurs.
0x	Pins are tri-stated when a shutdown event occurs.

Bits 17:16 – PSSBDF[1:0] PWM Output Pins OCxB, OCxD, and OCxF Shutdown State Control bits

Value	Description
11	Pins are driven active when a shutdown event occurs.
10	Pins are driven inactive when a shutdown event occurs.
0x	Pins are tri-stated when a shutdown event occurs.

Bits 5:0 – DT[5:0] Capture/Compare/PWM Deadtime Select bits(2)

Value	Description
111111	Insert 63 dead time delay periods between complementary output signals.
000010	Insert 2 dead time delay periods between complementary output signals
000001	Insert 1 dead time delay period between complementary output signals.
000000	Dead time logic disabled.

28.3.4. CCPx Status Register

Name: CCPxSTAT
Offset: 0x1B0C, 0x1B3C, 0x1B6C, 0x1B9C, 0x1BCC

Notes:
1. This is not a physical bit location and will always read as '0'. Writing '1' will initiate the hardware event.
2. This bit has no effect when CLKSEL[2:0] (CCPxCON1[10:8]) = 000 or TMRSYNC = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
				PRLWIP	TMRHWIP	TMRLWIP	RBWIP	RAWIP
Access				R	R	R	R	R
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
						ICGARM		
Access						W		
Reset						0		

Bit	7	6	5	4	3	2	1	0
	CCPTRIG	TRSET	TRCLR	ASEVT	SCEVT	ICDIS	ICOV	ICBNE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 20 – PRLWIP CCPxPRL Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxPRL register with the buffered contents is in progress.
0	An update to the CCPxPRL register is not in progress.

Bit 19 – TMRHWIP CCPxTMRH Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxTMRH register with the buffered contents is in progress.
0	An update to the CCPxTMRH register is not in progress.

Bit 18 – TMRLWIP CCPxTMRL Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxTMRL register with the buffered contents is in progress.
0	An update to the CCPxTMRL register is not in progress.

Bit 17 – RBWIP CCPxRB Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxRB register with the buffered contents is in progress.
0	An update to the CCPxRB register is not in progress.

Bit 16 – RAWIP CCPxRA Write in Progress Status bit(2)

Value	Description
1	An update to the CCPxRA register with the buffered contents is in progress.
0	An update to the CCPxRA register is not in progress.

Bit 10 – ICGARM Input Capture Gate Arm bit(1)
A write of '1' to this location will arm the Input Capture gating logic for a one-shot gate event when ICGSM[1:0] = 01 or 10. The bit location reads as '0' (see Table 28-9).

Bit 7 – CCPTRIG CCPx Trigger Status bit

Value	Description
1	Timer has been triggered and is running.
0	Timer has not been triggered and is held in Reset.

Bit 6 – TRSET CCPx Trigger Set Request bit(1)
A write of '1' to this location will request a trigger of the time base when TRIGEN = 1. The bit will clear automatically after the trigger event has been generated, allowing a new trigger event to be requested.

Bit 5 – TRCLR CCPx Trigger Clear Request bit(1)
A write of '1' to this location will request a trigger cancellation when TRIGEN = 1 and CCPTRIG = 1. The bit clears automatically after the cancellation has been completed, allowing a new cancellation to be requested.

Bit 4 – ASEVT CCPx Auto-Shutdown Event Status/Control bit

Value	Description
1	A shutdown event is in progress; CCP outputs are in the Shutdown state.
0	CCP outputs operate normally.

Bit 3 – SCEVT Single Edge Compare Event Status bit

Value	Description
1	A single edge Compare event has occurred.
0	A single edge Compare event has not occurred.

Bit 2 – ICDIS Input Capture Disable bit

Value	Description
1	Event on the Input Capture pin does not generate a capture event.
0	Event on Input Capture pin will generate a capture event.

Bit 1 – ICOV Input Capture Buffer Overflow Status bit

Value	Description
1	The Input Capture FIFO buffer has overflowed.
0	The Input Capture FIFO buffer has not overflowed.

Bit 0 – ICBNE Input Capture Buffer Status bit

Value	Description
1	Input Capture buffer has data available.
0	Input Capture buffer is empty.

28.3.5. CCPx Time Base Register

Name: CCPxTMR
Offset: 0x1B10, 0x1B40, 0x1B70, 0x1BA0, 0x1BD0

Notes:
1. TMR[31:16] will be available when operating in a valid 32-Bit Operating mode or the dual 16-Bit Time Base mode. TMR[31:16] will read as '0' when operating in all other modes.
2. All writes to CCPxTMR are buffered for atomic update operations. The CCPxTMR value is not updated until the uppermost byte of the timer is written. If the timer clock source is asynchronous, user software must monitor the status bits to ensure the prior write has been completed before performing another write.

Bit	31	30	29	28	27	26	25	24
	TMR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	TMR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TMR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	TMR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – TMR[31:0] 32-Bit Time Base Value bits[1,2]

28.3.6. CCPx Period Register

Name: CCPxPR
Offset: 0x1B14, 0x1B44, 0x1B74, 0x1BA4, 0x1BD4

Note:

1. For Dual 16-Bit Timer mode, the PR[31:16] bits set the count period for the second 16-bit Timer. For 32-bit Timer operation, the PR[31:0] bits set the count period for the single 32-bit Timer. On a device reset, the module will reset to a Dual 16-Bit Timer mode. The CCPxPR reset value of FFFFFFFF provides the maximum count period for both timers. The PR[31:16] bits are not available in 16-Bit Output Compare modes and will read as '0'. The PR[31:0] bits are not available in 32-Bit Output Compare modes and will read as '0'.

Bit	31	30	29	28	27	26	25	24
	PR[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	23	22	21	20	19	18	17	16
	PR[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	PR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	PR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 31:0 – PR[31:0] Period Register bits[(1)]

28.3.7. CCPx Primary Compare Register (Timer/Compare Modes Only)

Name: CCPxRA
Offset: 0x1B18, 0x1B48, 0x1B78, 0x1BA8, 0x1BD8

Note:
1. For operating the module in 32-bit mode, the CCPxRA register contains the lower 16 bits to be compared to the time base.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Access								
Reset								
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
	CMPA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	CMPA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CMPA[15:0] Compare Value bits(1)
The 16-bit value to be compared against the CCP time base.

28.3.8. CCPx Secondary Compare Register (Timer/Compare Modes Only)

Name: CCPxRB
Offset: 0x1B1C, 0x1B4C, 0x1B7C, 0x1BAC, 0x1BDC

Note:
1. For operating the module in 32-bit modes, the CCPxRB register contains the upper 16 bits to be compared to the time base. In certain 16-bit modes, the CCPxRB register sets the module output trigger time.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	CMPB[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – CMPB[15:0] Compare Value bits(1)
The 16-bit value to be compared against the CCP time base.

28.3.9. CCPx Capture Buffer Register (Compare Modes Only)

Name: CCPxBUF
Offset: 0x1B20, 0x1B50, 0x1B80, 0x1BB0, 0x1BE0

Notes:
1. The CCPxBUF[31:16] bits are only available in 32-Bit Input Capture modes. BUF[31:16] will read as '0' when operating in 16-Bit Input Capture mode.
2. CCPxBUF[31:0] will read as '0' for all Timer, Output Compare and PWM operating modes.

Bit	31	30	29	28	27	26	25	24
	BUF[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	BUF[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	BUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – BUF[31:0] Capture Buffer Value bits[1,2]
Indicates the oldest captured time base value in the FIFO.

28.4. Operation

28.4.1. Time-Base Generator

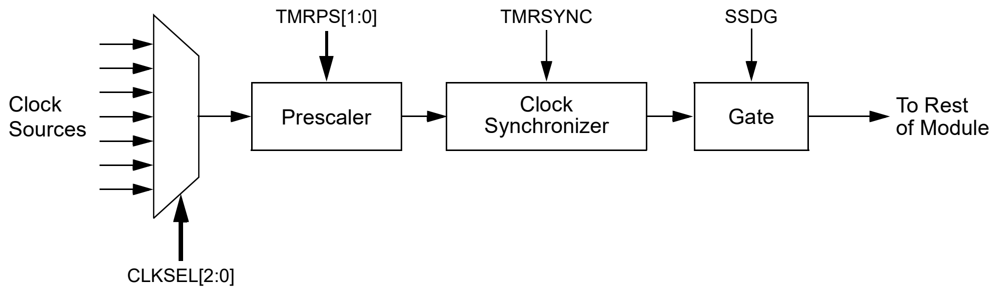
The Time-Base Generator (TBG) provides a time base for the rest of the module using clock signals available on the microcontroller. This serves not only as the time base for the Timer modes, but also allows Input Capture and Output Compare pre-modes to operate without depending on another on-chip timer module.

Up to eight clock inputs are available to the clock generator, including the system clock (T_{CY}) and other on-chip oscillator sources. Depending on the device, external clock inputs may also be available. A prescaler divides the selected clock source to a suitable frequency for use by the module.

The TBG has the ability to synchronize its operation with the selected clock source, subject to input timing restrictions or the module's operating conditions. Setting the TMRSYNC bit (CCPxCON1[11]) enables synchronization of the time base with the clock input.

The TBG is shown in Figure 28-2.

Figure 28-2. Time-Base Clock Generator



28.4.1.1. Gating Logic

The Time-Base Generator incorporates a hardware gate that can disable the timer increment clock to the timer gate, which is available on Timer modes only.

Gating is controlled using the ASDG[7:0] Control bits (CCPxCON2[7:0]) and the SSDG bit (CCPxCON2[12]). All of these bits are logically ORed together to generate a gating enable signal for the TBG.

Setting any one of the ASDGx bits enables its corresponding hardware trigger; any or all of the bits may be set to select multiple sources. The available sources for gating and auto-shutdown are device-dependent and typically include such sources as comparator outputs, I/O pins (including OCFA and OCFB for PWM operation), software control and so on. Any output signal from any of the enabled sources disables the TBG output. Events are generally level-sensitive and not edge-triggered.

The SSDG bit is simply a gating source that can be manipulated in software. Setting SSDG has the same effect as an input from any of the hardware sources.

The gating feature is described in the following sections:

- Timer Gating (see Clock Gating for Timer Modes)
- Auto-Shutdown for Output Compare (see Auto-Shutdown Control)
- Gated Input Capture (see Input Capture Signal Gating)

Regardless of the operating mode, interrupt events are not generated by the CCP module based on the status of the gating inputs. If an interrupt is required for a gating event, the gating source itself must be used to generate the interrupt.

28.4.2. Timer Mode

When CCSEL = `0` and MOD[3:0] = `0000`, the module functions as a timer. There are two basic Timer modes selected by the T32 bit (CCPxCON1[5]); they are shown in Table 28-5. In either mode, the timer can operate as a free-running timer/counter, operate synchronously with other modules or be triggered by other modules or external events.

Table 28-5. Timer Operating Modes

T32	Operating Mode
0	Dual Timer Mode (16-bit)
1	Timer Mode (32-bit)

28.4.2.1. Dual 16-Bit Timer Mode

Dual 16-Bit Timer mode is selected when T32 = `0`. This mode is useful for the following functions:

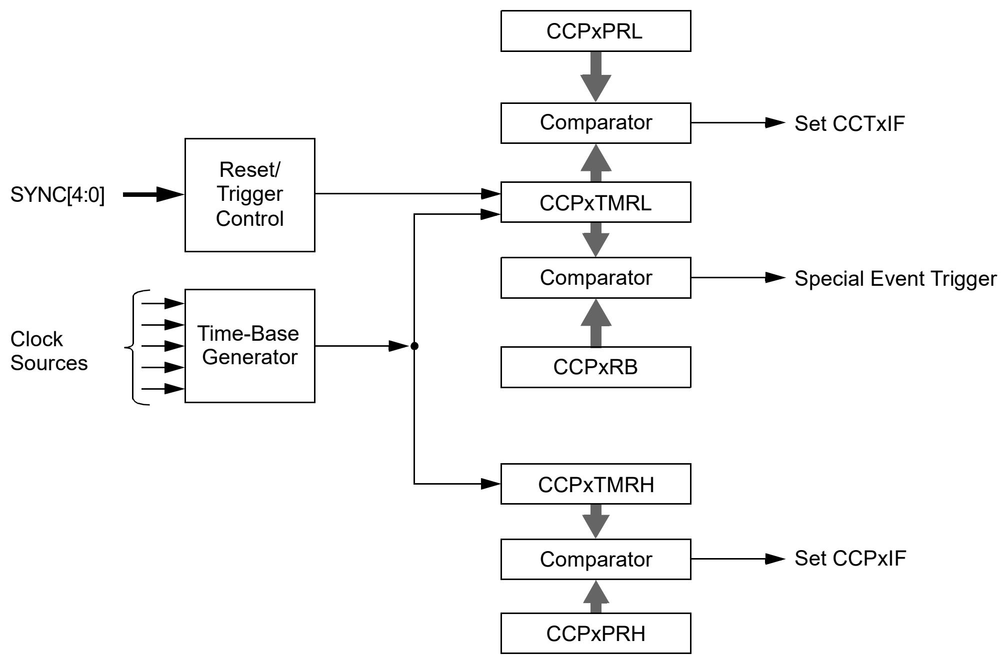
- CCPxTMR Periodic CPU Interrupts

- Master Time Base Function for Synchronizing Other CCP Modules
- Triggering Periodic A/D Conversion
- Periodic Wake from Sleep (if an Appropriate Clock Source is Available)

Note: The CCPxTMRH/L register bit locations may not be readable by the user if a high-speed asynchronous clock source is used to clock the time base. For a low-speed read, a double read can be done and the results compared.

Dual 16-Bit Timer mode provides a simple timer function with two independent 16-bit timer/counters, as shown in Figure 28-3. The primary timer, based on the lower word of the CCPxTMR, is fully functional and can interact with other modules on the device. It can generate the CCP Sync signals for use by other CCP modules. It can also use the SYNC[4:0] signal generated by other modules. The secondary timer, based on the upper word of CCPxTMR, has limited functionality. It is intended to be used only as a periodic interrupt source for scheduling CPU events. It does not generate an output trigger signal like the primary time base.

Figure 28-3. 16-Bit Dual Timer Mode



Both the primary and secondary timers use the same clock source from the TBG, as selected by CLKSEL[2:0]. The CCPxTMRH/L register bit locations provide user access to the two 16-bit time bases. Both timer register bit locations (CCPxTMRL and CCPxTMRH) increment at the same time based on the timer input; however, only the primary timer (CCPxTMRL) can use the timer Sync functionality. The secondary timer (CCPxTMRH) does not have timer Sync functionality.

The CCPxPRL register bit locations control the period for the primary 16-bit time base when SYNC[4:0] = `00000`. When the module is configured to use an external synchronization source, the primary 16-bit time base is reset when the source selected by SYNC[4:0] is asserted. The module's Sync signal is generated whenever the time base rolls over or is reset to '`0`'.

The primary timer can generate the CCP interrupt when the value of CCPxTMRL resets to 0000h. When SYNC[4:0] = `00000`, CCPxTMRL matches CCPxPRL. If SYNC[4:0] is not '`00000`', CCPxTMRL resets and generates a CCT Interrupt Flag (CCTxIF) event whenever the signal selected by SYNC[4:0] is asserted.

The CCPxPRH register bit locations control the count period of the secondary 16-bit timer. The secondary timer does not support external synchronization and is not affected by the selected

SYNC[4:0] input. The secondary time base begins counting when the CCPON bit (CCPxCON1[15]) is set. When a match occurs between the CCPxPRH register bit locations and the CCPxTMRH count value, the secondary 16-bit time base is reset and a timer rollover interrupt event (CCPxIF) is generated.

If either of the 16-bit timers is not used in the application, the timer can be disabled by writing 0000h to the corresponding period register. The timer is held in Reset, and no interrupts are generated as long as the period register's value is '0'. The CCPxPRH and CCPxPRL register bit locations are not buffered in this operating mode.

To use the module in Dual 16-Bit Timer mode:

1. Set CCSEL = 0 to select the Time Base/Output Compare mode of the module.
2. Set T32 = 0 to select the 16-bit time-base operation.
3. Set MOD[3:0] = 0000 to select the Time-Base mode.
4. Set SYNC[4:0] to the desired time base synchronization source:
   – Configure and enable the external source selected by SYNC[4:0] before enabling the timer.
   – If the timer is not using an external Sync source (SYNC[4:0] = 00000), or if the module is synchronizing to itself (the SYNC[4:0] bits select the module's own value as a Sync source), write the desired count period of the primary 16-bit time base to CCPxPRL.
5. If the secondary timer is also being used, write a non-zero value to CCPxPRH to specify the count period.
6. If the special A/D trigger is being used, set CCPxRB for the desired trigger output time.
7. Enable the module by setting the CCPON bit.
8. If an external synchronization source is selected in step four, configure and enable that source to allow the primary 16-bit time base to begin counting.

28.4.2.1.1. Special Event Trigger

In select devices, the Dual 16-Bit Timer mode can be used to generate a Special Event Trigger Output signal. The primary timer can be used to start A/D conversions and trigger other peripheral events. The trigger period is set by the value of the CCPxRB register and must be less than the counter period, as defined by the CCPxPRL register bit locations.

28.4.2.2. 32-Bit Timer Mode

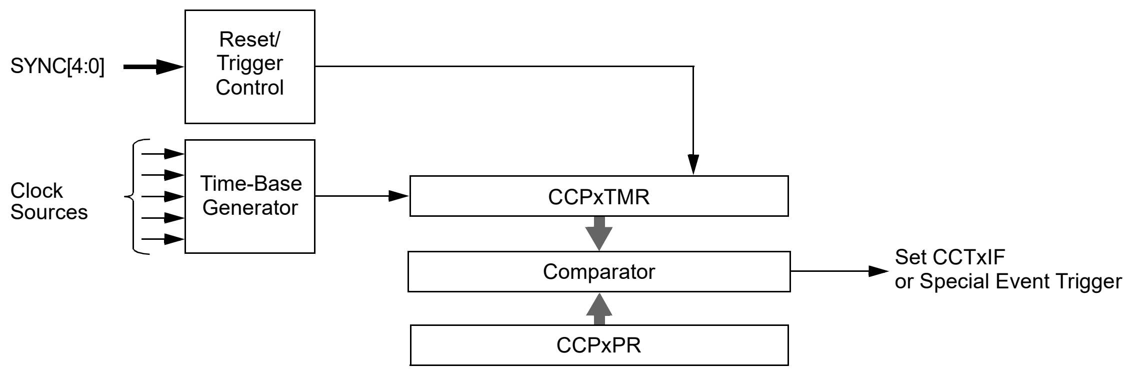
The 32-Bit Timer mode is selected when T32 = 1. In this mode, CCPxTMR will be used as single 32-bit register.

This mode provides a simple timer function when it is important to track long time periods. It is useful for the following functions:

- Periodic CPU Interrupts
- Synchronization and Trigger Generation for Other CCP Modules
- Periodic ADC Conversion Triggering
- Periodic Wake from Sleep (if an Appropriate Clock Source is Available)

No input or output functions are available from the CCP module in this operating mode.

Figure 28-4. 32-Bit Timer Mode



When external synchronization is not selected (SYNC[4:0] = `00000`), the CCPxPR register sets the count period for the timer. A match between the CCPxTMR and the CCPxPR register also automatically generates the Sync output signal whenever the module is enabled (CCPON = `1`).

To use the module in 32-Bit Timer mode:

1. Set CCSEL = `0` to select the Time-Base/Output Compare mode of the module.
2. Set T32 = `1` to select the 32-bit time-base operation.
3. Set MOD[3:0] = `0000` to select the Time-Base mode.
4. Set SYNC[4:0] to the desired timer synchronization source:
   - Configure and enable the external source selected by SYNC[4:0] before enabling the timer.
   - If the timer is not using an external Sync source (SYNC[4:0] = `00000`), or if the module is synchronizing to itself (SYNC[4:0] selects the module's own value as a Sync source), write the desired count period to CCPxPR.
5. Enable the module by setting the CCPON bit.

28.4.2.3. Clock Gating for Timer Modes

When operating in Timer mode, time-base gating can be used to gate the timer's operation (see Gating Logic for more information). This function provides a simple way to measure the time of an external event. Timer clock gating is enabled whenever one or more of the ASDG[7:0] bits (CCPxCON2[7:0]) are set or when the SSDG bit (CCPxCON2[12]) is set.

28.4.3. Input Capture Mode

When CCSEL = `1`, the module is configured for Input Capture mode. This mode is used to capture a timer value from an independent timer base on the occurrence of an event on an input pin. This mode is useful in applications requiring frequency (time period) and pulse measurement.

Input Capture mode uses the CCPxTMR registers as a dedicated 16/32-bit synchronous, up counting timer used for event capture. This value is written to the FIFO buffer when a capture event occurs. The internal value may also be read with a synchronization delay from the CCPxTMR register.

Input Capture mode is the only major mode available when CCSEL is set. The T32 and the MOD[3:0] bits determine the various Capture modes, as shown in Table 28-6.

Figure 28-5 provides a simplified block diagram of the Input Capture mode.

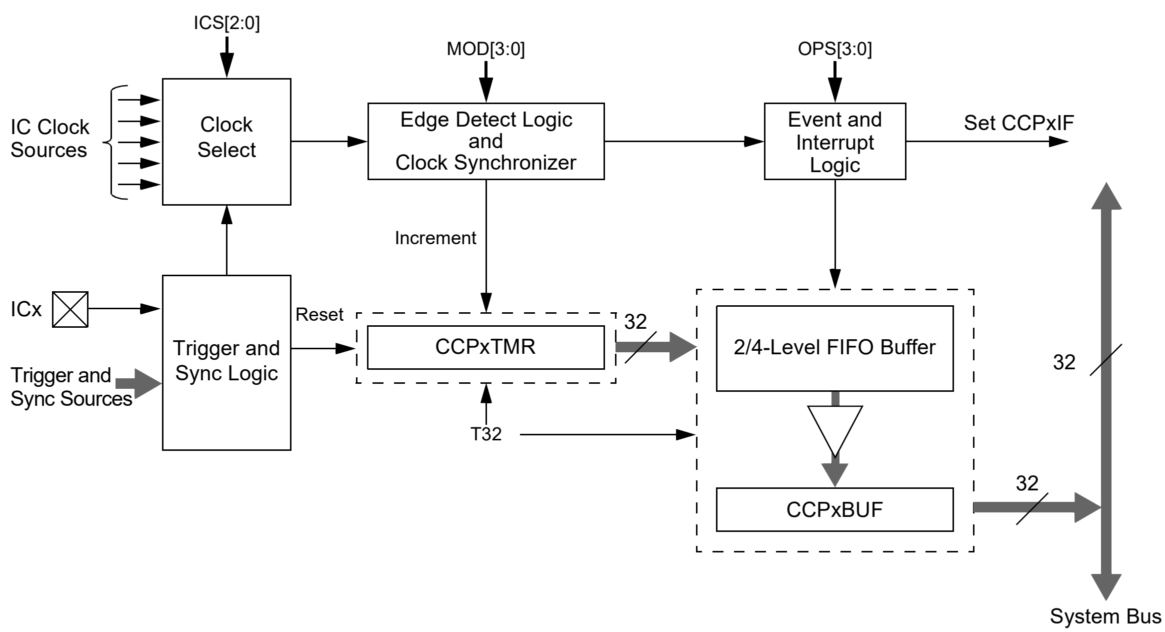
Table 28-6. Capture Modes

T32 (CCPxCON1[15])	MOD[3:0] (CCPxCON1[3:0])	Operating Mode
0	0000	Edge Detect (16-bit capture)
1	0000	Edge Detect (32-bit capture)
0	0001	Every Rising (16-bit capture)

Table 28-6. Capture Modes (continued)

T32 (CCPxCON1[15])	MOD[3:0] (CCPxCON1[3:0])	Operating Mode
1	0001	Every Rising (32-bit capture)
0	0010	Every Falling (16-bit capture)
1	0010	Every Falling (32-bit capture)
0	0011	Every Rise/Fall (16-bit capture)
1	0011	Every Rise/Fall (32-bit capture)
0	0100	Every 4th Rising (16-bit capture)
1	0100	Every 4th Rising (32-bit capture)
0	0101	Every 16th Rising (16-bit capture)
1	0101	Every 16th Rising (32-bit capture)

Figure 28-5. Input Capture Block Diagram



28.4.3.1. Initialization

Since the module can be used for Input Capture/Output Compare/PWM, selecting the correct operation required should be the first task. The best practice is to clear all the associated control registers.

When the CCP module is reset or disabled (CCPON = 0):

- The ICOV and ICBNE status flags are cleared.
- CCPxBUFH/L and their FIFO buffer are cleared.
- CCPxTMRH/L are reset to zero.
- The capture prescaler counter is reset to zero.
- The capture event counter for interrupt generation is reset to zero.

28.4.3.1.1. Mode Selection

As with Timer and Output Capture/PWM modes, the MOD[3:0] bits selects the Capture mode and prescaler options. To avoid inadvertent interrupts, always disable the module by clearing the CCPON bit when changing Capture modes. It is recommended to set the CCSEL bit and configure the MOD[3:0] bits in a single operation before enabling the module.

28.4.3.1.2. Timer Clock Source Selection

dsPIC33A family devices may have one or more Input Capture channels. Each channel can select between one of eight clock sources for its time base by using the CLKSEL[2:0] bits (CCPxCON1[10:8]), as described in Time-Base Generator. The module can be set to use the system clock source or clock from CLKGEN12, with Synchronization mode enabled, in the timer. The Input Capture pin (ICx) should be selected for Input Capture operation. It is recommended that the clock source be selected before enabling the module and not be changed during operation.

32-Bit Input Capture Support

The Input Capture modes have the ability to operate with a 32-bit time base. The 32-bit mode is selected by setting the T32 bit. All input capture functions are the same between 16-bit and 32-bit modes, with these changes in 32-bit operations:

- CCPxTMR is a 32-bit register.
- CCPxBUF is a 32-bit register.
- The FIFO buffer only has two levels available in 32-Bit Operating mode.

Example 28-1 shows a typical procedure for setting up Input Capture mode.

Example 28-1. Setup for Input Capture mode (Every Rising Edge)

```
CCP1CON1bits.CCSEL = 1;      // Input capture mode
CCP1CON1bits.CLKSEL = 0;     // Set the clock source (Tcy)
CCP1CON1bits.T32 = 0;        // 16-bit Dual Timer mode
CCP1CON1bits.MOD = 1;        // Capture ever rising edge of the event
CCP1CON2bits.ICS = 0;        // Capture rising edge on the Pin
CCP1CON1bits.OPS = 0;        // Interrupt on every input capture event
CCP1CON1bits.TMRPS = 0;      // Set the clock pre-scaler (1:1)
CCP1CON1bits.ON = 1;         // Enable CCP/input capture
```

Input Capture Source

The ICS[2:0] (CCPxCON2[18:16]) control bits select the input source that is used for the capture function.

I/O Pin Control

The capture module is input-only and will not prevent other modules from driving its multiplexed pin. It is the user's responsibility to ensure no other modules are driving the capture pin.

28.4.3.2. Capture Event Modes

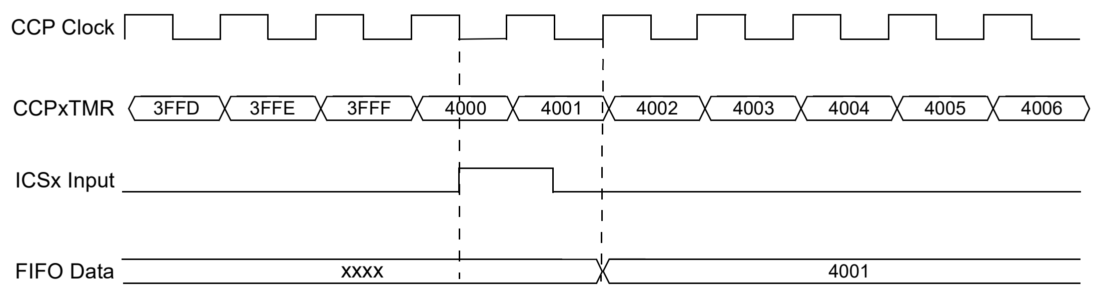
The module can capture a timer value on any of the following ICx pin transitions:

- Every rising edge (MOD[3:0] = `0001`)
- Every falling edge (MOD[3:0] = `0010`)
- Every rising and falling edge (MOD[3:0] = `0000`, `0011`)

Since the Input Capture pin is sampled on the falling edge of the timer clock, the capture pulse width must be greater than the timer clock period, plus some margin.

Because of internal synchronization requirements, the timer value captured will be up to 1.5 CCP clock cycles after the time of the actual capture edge event, as shown in Figure 28-6.

Figure 28-6. Input Capture Timing (Rising Edge)

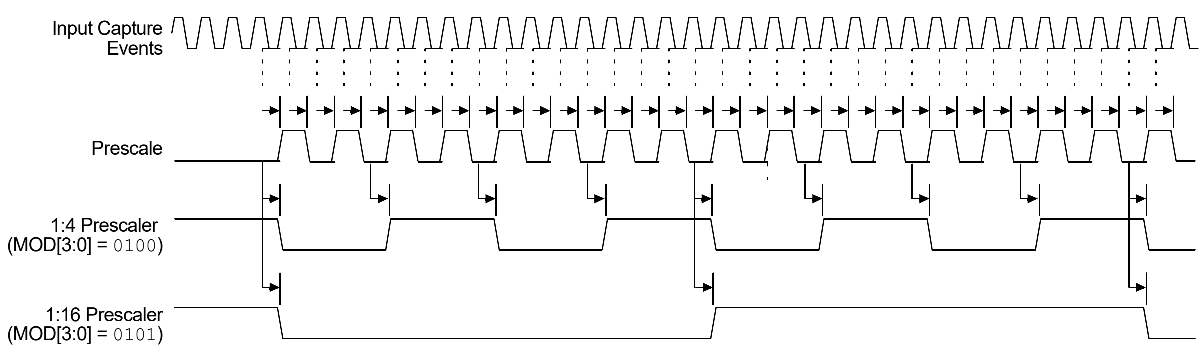


28.4.3.2.1. Input Capture Prescaler

Using the input prescaler, the Input Capture module can capture a timer value on every fourth edge (MOD[3:0] = `0100`) or every sixteenth edge (MOD[3:0] = `0101`) of the ICx input pin.

The capture pulse-width requirements are different than those for Simple Capture mode. Because of synchronization requirements inside, the timer value captured will be one to two timer clock cycles after the time of the edge capture, as shown in Figure 28-7.

Figure 28-7. Input Capture Prescaler



28.4.3.2.2. Edge Detect (Hall Sensor) Mode

Edge Detect mode (MOD[3:0] = `0000`) operates the same as Capture Every Edge mode (MOD[3:0] = `0011`), except that capture interrupt events do not stop when the Input Capture Buffer Overflow Status (ICOV) flag becomes set. This allows a continuous stream of capture events to trigger interrupt events without the need to continuously empty the FIFO.

28.4.3.2.3. Non-Capture Modes

When the module is in any operating mode that is not an Input Capture mode (CCM = `0`), or the module is disabled (CCPON = `0`), input capture functions are disabled and the module generates no input capture events or related interrupts. Reads of the input capture buffer will read as ‘`0`’.

28.4.3.2.4. Input Capture Buffer

The Input Capture FIFO buffer is up to four levels deep, depending on the Capture mode selected. For 16-bit timer captures, there are four levels in the FIFO (16-bit wide); for 32-bit timer captures, there are two levels (32-bit wide). The number of capture events required to generate a CPU interrupt can be selected by the user.

There are two status flags that provide status on the FIFO buffer. The ICBNE status bit (CCPxSTAT[0]) indicates that at least one capture event has occurred. The ICOV status bit (CCPxSTAT[1]) indicates that there have been more events than the buffer’s current depth (four in 16-bit mode, two in

32-bit mode). These status flags operate the same for 16-bit capture operations and 32-bit capture operations.

While the CCP module is in Reset or not in Capture mode:

- The ICOV status flag is cleared.
- The ICBNE status flag is cleared.
- The FIFO is marked as empty.
- A read of the FIFO buffer will return '0'.

The ICBNE status flag is set on the first capture event and remains set until all capture events have been read from the FIFO. For example, if three capture events have occurred, then three reads of the Capture FIFO buffer are required before the ICBNE flag will be cleared. Each read of the FIFO buffer will allow the remaining word(s) to move to the next available top location of the FIFO.

In the event that the FIFO buffer is full with capture events and another capture event occurs prior to a read of the FIFO, an Overflow condition will occur and the ICOV bit becomes set. In addition, the capture event which caused the Overflow is not recorded, and subsequent capture events will not be placed into the FIFO until the Overflow condition is cleared by completely emptying the FIFO.

Overflow conditions cannot occur when the module is not in an Input Capture mode or when Edge Detect mode is enabled (MOD[3:0] = 0000).

Clearing of the Overflow condition can be accomplished in one of the following ways:

1. Disable the module by clearing the CCPON bit.
2. Read the Input Capture buffer until ICBNE = 0 (twice for 32-bit captures, four times for 16-bit captures).
3. Clear the ICOV bit in software. This effectively discards all previously stored data in the FIFO by resetting the data pointers to the beginning of the FIFO buffer. Clearing the ICOV in software also causes the ICBNE bit to be cleared automatically.
4. Perform a device Reset.

Upon clearing the Overflow condition, the ICBNE status flag is cleared, and to resume the Capture mode, the user software must clear the ICOV status flag. If the module is disabled, and then re-enabled in Input Capture mode later, the FIFO buffer contents will be undefined and a read will yield indeterminate results.

In the event that a FIFO read is performed after the last read and no new capture event has been received, the FIFO read and write pointers will be pointing to the first buffer location of the FIFO. A read of the FIFO will return the value held in the first buffer location.

The FIFO pointer is adjusted whenever the most significant word of the buffer result is read by the CPU. This allows the results of a 32-bit input capture to be read by the 16-bit CPU.

28.4.3.3. Input Capture Interrupts

While in Input Capture mode, the module has the ability to generate an interrupt upon a capture event. A capture event is defined by writing a timer value to the FIFO.

The OPS[3:0] control bits (CCPxCON1[27:24]) select the interrupt postscaler, specifying the number of capture events that must occur before an interrupt is generated. Options range from an interrupt on every capture to every fourth capture. The first capture event is defined as the capture event occurring after a mode change from the Disabled state (CCPON = 0) or after ICBNE = 0.

On buffer overflow, the capture events cease and the interrupts stop unless OPS[3:0] = 0000 (interrupt on every capture). Clearing the FIFO by reading it also clears the internal interrupt counter and may affect when an interrupt is generated.

Applications often use the Input Capture pins as auxiliary external interrupt sources. In Edge Detect mode, interrupts occur regardless of FIFO overflow, as specified by OPS[3:0]. There is no need to perform a dummy read on the Input Capture buffer to clear the event because the capture interrupt events will not stop when the FIFO Overflow (ICOV) flag becomes set. This allows a continuous stream of capture events to trigger interrupt events without the need to continuously empty the FIFO.

For example, assume that OPS[3:0] = `0001`, specifying an interrupt on every second capture event. The following sequence of events will produce a single CCPxIF, as shown:

1. Turn on module; event count = 0.
2. Capture first event; FIFO contains one entry, event count = 1.
3. Read FIFO; FIFO is empty, event count = 0.
4. Capture second event; FIFO contains one entry, event count = 1.
5. Capture third event; FIFO contains two entries, event count = 2, set CCPxIF.
6. Clear interrupt count when interrupt is set (event count = 0).
7. Capture fourth event; FIFO contains three entries, event count = 1.
8. Read FIFO three times; FIFO is empty, interrupt count = 0.
9. Capture fifth event; FIFO contains one entry, event count = 1.
10. Read FIFO; FIFO is empty, event count = 0.

28.4.3.3.1. Timer Interrupts in Input Capture Modes

The module produces both timer interrupts (CCTxIF) as well as capture interrupts (CCPxIF) while operating in Input Capture mode. However, the timer interrupts only occur at the timer rollover, from FFFFh to 0000h, since there is no period register available to set the count period. If a shorter timer count period is desired, a second CCP module or external timer may be used to provide a synchronization source for the Input Capture time base.

28.4.3.4. Input Capture Operation with Synchronization and Triggering

By default, the CCP module in Input Capture mode operates with a free-running timer. The CCPxPR register is not available to set a different timer period in Input Capture mode. It is recommended to keep SYNC[4:0] configured as '`11111`' to maintain the free-running timer.

The timer will be held at 0000h under either of these conditions:

- Triggered operation is enabled (TRIGEN = `1`), and a trigger event has not occurred (CCPTRIG = `0`).
- An external Sync source has been selected (SYNC[4:0] has a value other than '`11111`'), which has not been enabled.

In either case, input capture input events will occur; however, a value of 0000h will always be captured in the FIFO. For these reasons, the triggered operation and externally synchronized operation are not recommended.

28.4.3.4.1. Input Capture Signal Gating

The input capture source can optionally be gated by software or hardware to allow windowed capture measurements. This feature provides noise immunity in sensing applications.

The ICDIS bit (CCPxSTAT[2]) provides the status of the input signal gating function. When the ICDIS bit is cleared, capture events generated by the edge detect logic are allowed. When the ICDIS bit is set, events from the edge detect logic are inhibited.

The time-base gating logic is used for Input Capture signal gating (see Gating Logic for more information). The ASDG[7:0] Control bits (CCPxCON2[7:0]) select one or more input sources that are used to clear the ICDIS status/control bit. The SSDG bit (CCPxCON2[12]) may also be used to manually gate Input Capture signals in software.

The behavior of the ASDGx sources and the SSDG bit depends on the Gating Source mode, which is selected using the ICGSM[1:0] Control bits (CCPxCON2[23:22]). Three different options are available:

- When ICGSM[1:0] = 00, gating is level-sensitive. A low input level from the gating source disables subsequent capture events, and the ICDIS bit will be set to reflect this. A high input level enables subsequent capture events, and the ICDIS bit will be cleared to reflect this.
- When ICGSM[1:0] = 01, gating occurs with a rising edge of the gating source; the ICDIS bit is cleared, disabling subsequent capture events. This is a One-Shot mode; subsequent edges from the gating source will have no effect.
- When ICGSM[1:0] = 10, gating occurs on the falling edge of the gating source; the ICDIS bit is set, enabling subsequent capture events. This is a One-Shot mode; subsequent edges from the gating source will have no effect.

When ICGSM[1:0] = 01 or 10, the input capture gating logic operates in a One-Shot mode. The user may arm the gating logic after a gating event by writing a '1' to the ICGARM (CCPxSTAT[10]) bit. This write to ICGARM has the effect of resetting the gate signal edge detection logic and also resets the ICDIS status bit to the appropriate value. User software can determine the state of the one-shot logic by reading the ICDIS status bit:

- When ICGSM[1:0] = 01 and a '1' is written to ICGARM, the gate signal edge detection logic is armed to look for a rising edge, and the ICDIS bit is set to disable input capture events until the rising edge occurs on the gate signal.
- When ICGSM[1:0] = 10 and a '1' is written to ICGARM, the gate signal edge detection logic is armed to look for a falling edge, and the ICDIS bit is cleared to enable input capture events until the next falling edge occurs on the gate signal.

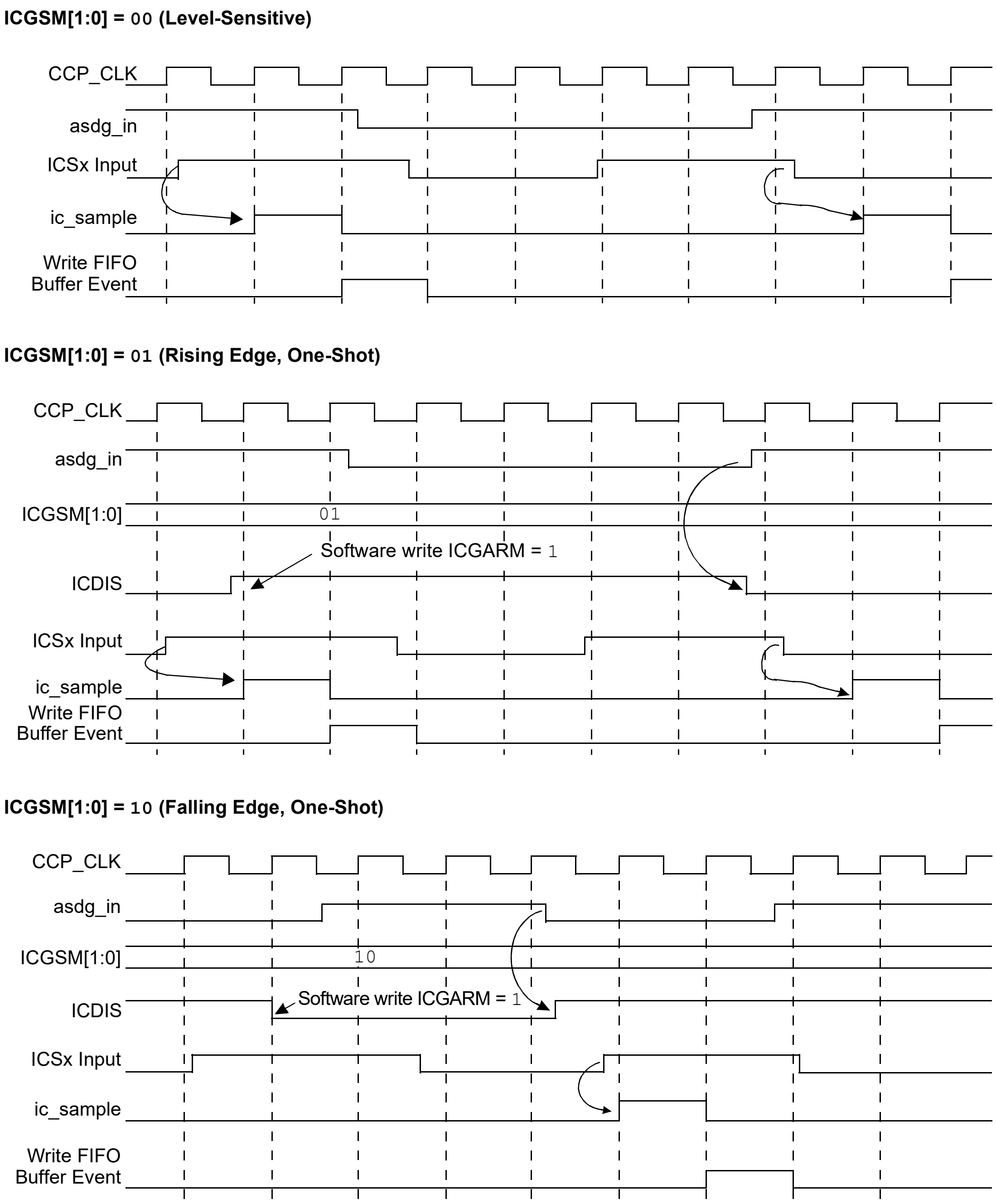
Figure 28-8 shows the timing for gated capture events. Input events are sampled on the falling edge of the clock source. The example assumes that the input capture module is configured to capture every rising and falling edge (MOD[3:0] = 0011).

In the One-Shot modes, the edge detect logic is set to look for the appropriate edge event; the ICDIS bit remains set or clear (depending on the mode) until that type of event occurs. The user may re-arm the gating logic after a gating event by rewriting ICGSM[1:0]. This act of writing to these bits (even if the same value) resets the gate signal edge detection logic and also resets the ICDIS status bit to the appropriate value.

To use input capture gating:

1. Select and configure the gating source.
2. Enable the appropriate gating signal source(s) using the ASDG[7:0] bits; alternatively, set or clear the SSDG bit during the event for software only control.
3. Select the Gating mode using ICGSM[1:0].
4. Configure the module for the desired Input Capture mode and input source using the MOD[3:0] and ICS[2:0] control bits. The module is now armed for a gate event.
5. The next valid rising or falling input signal edge (depending on Capture mode) after ICDIS is cleared will trigger a capture event.

Figure 28-8. Gated Input Capture Examples



28.4.4. Output Compare and PWM Modes

When CCSEL = 0 and the MOD[3:0] bits are any value other than '0000', the module operates in Output Compare mode.

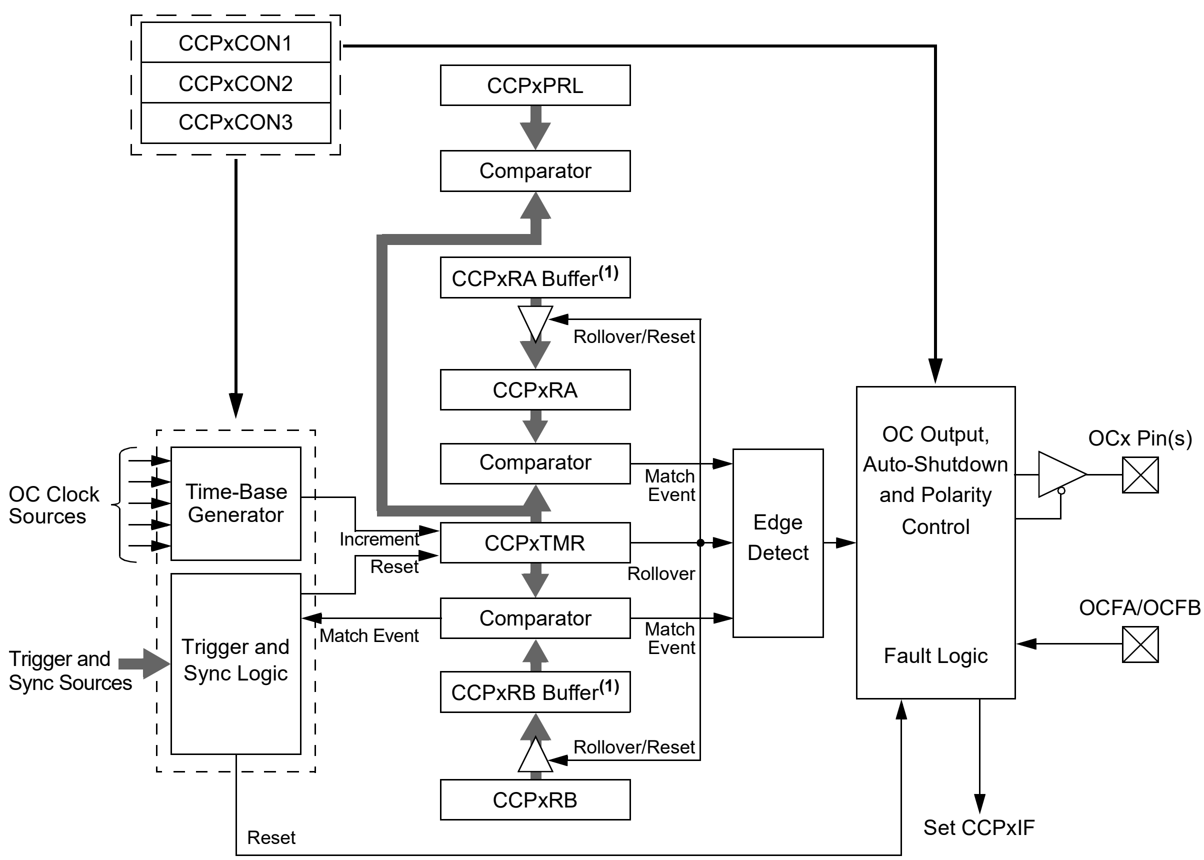
Table 28-7 summarizes the various Output Compare modes.

Table 28-7. Output Compare/PWM Modes

T32	MOD[3:0]	Operating Mode
0	0001	Output High on Compare (16-bit), Single Edge mode
1	0001	Output High on Compare (32-bit), Single Edge mode
0	0010	Output Low on Compare (16-bit), Single Edge mode
1	0010	Output Low on Compare (32-bit), Single Edge mode
0	0011	Output Toggle on Compare (16-bit), Single Edge mode
1	0011	Output Toggle on Compare (32-bit), Single Edge mode
0	0100	Dual Edge Compare (16-bit), Dual Edge mode
0	0101	Dual Edge Compare (16-bit buffered), PWM mode
0	0110	Reserved
0	0111	Reserved

The value of CCPxTMR is compared to one or two Compare registers, depending on its mode of operation. Output Compare mode can generate a single output transition or a train of output pulses and can generate interrupts on match-on-compare events. Figure 28-9 outlines the components used in Output Compare mode. The Output Compare mode can also function as a PWM generator.

Figure 28-9. Output Compare Block Diagram



Note:

1. Buffered Output Compare and PWM modes only.

28.4.4.1. CCP Single Edge Output Compare Mode

When MOD[3:0] = `0001`, `0010` or `0011`, the selected Output Compare channel is configured for these Single Output Compare Match modes:

- Compare forces pin high (MOD[3:0] = `0001`)
- Compare forces pin low (MOD[3:0] = `0010`)
- Compare toggles pin (MOD[3:0] = `0011`)

In Single Compare mode, the CCPxRA register is used. The register is loaded with a value and is compared to the module Timer register. A CPU interrupt is generated on each compare event.

Single Edge Compare mode uses these Timer/Data registers:

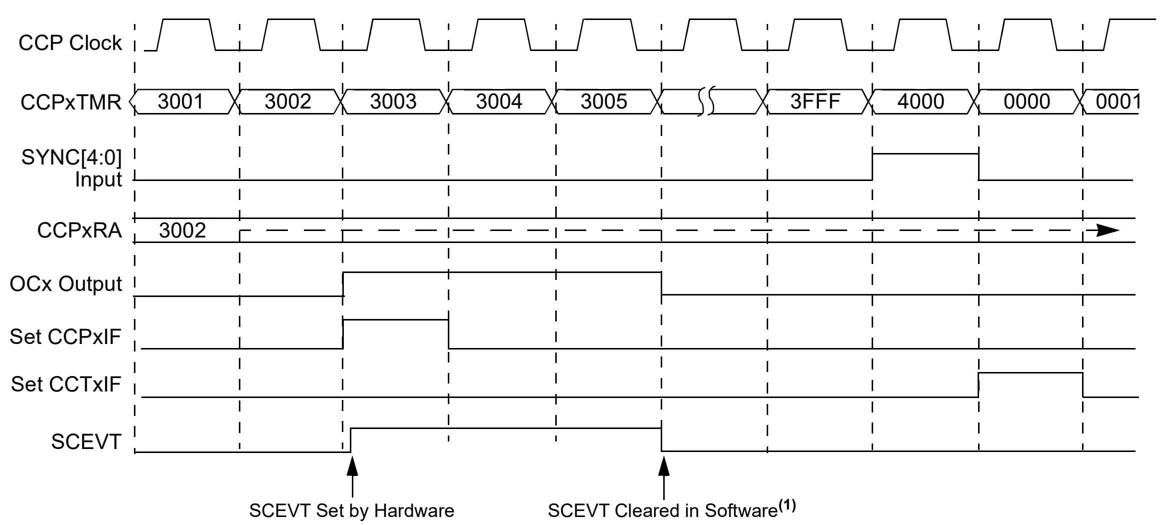
- CCPxTMRL as the Timer register (16-bit mode)
- CCPxTMR as the Timer register (32-bit mode)
- CCPxRA as the Compare Value register (16-bit mode)
- CCPxRB:CCPxRA as the Compare Value register (32-bit mode)
- CCPxPRL as the Timer Period register (16-bit mode only)

28.4.4.1.1. Single Edge Compare Mode (High Output)

In this mode (see Figure 28-10), the output pin is initially driven low and remains low until a match occurs between the timer and CCPxRA register. The key timing events to note are:

- The output pin is driven high, one clock period after a match occurs between the timer and CCPxRA registers. The output pin remains high until a mode change has been made or the module is disabled.
- The timer counts up until it rolls over or until the selected SYNC[4:0] input is asserted (depending on the value of SYNC[4:0]) and then resets to 0000h on the next clock.
- The compare interrupt signal (to set CCPxIF) is asserted, and the output pin is driven high.
- The timer interrupt signal (to set CCTxIF) is asserted for one clock period on a time-base Reset or rollover event.

Figure 28-10. Single Compare Mode (High Output)



Note:

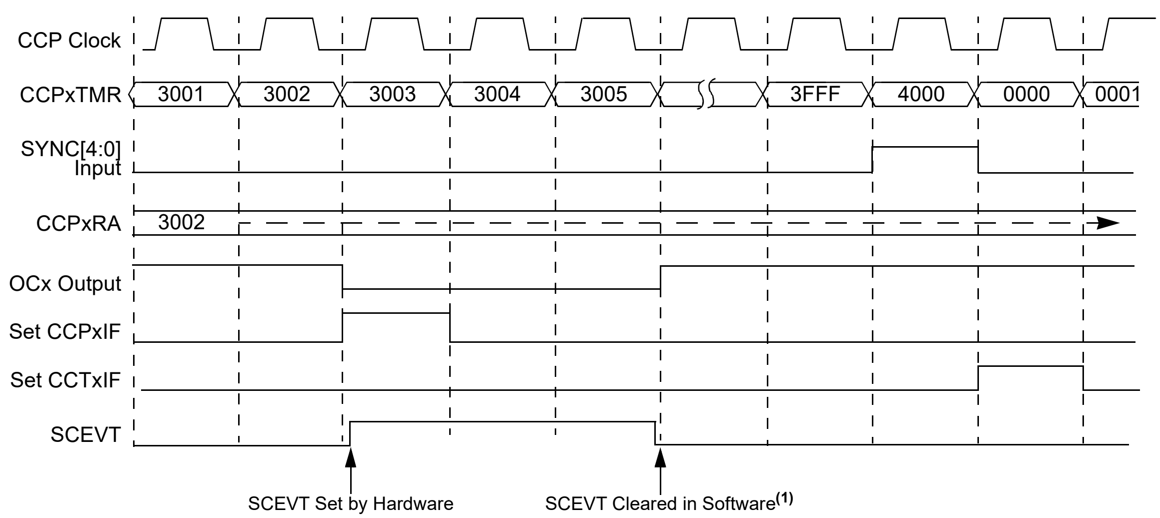
1. SCEVT has to be cleared to enable the next capture.

28.4.4.1.2. Single Compare Mode (Low Output)

Once the Compare mode has been enabled (Figure 28-11), the output pin will initially be driven high, and remain high until a match occurs between the timer and CCPxRA register. The key timing events to note are:

- The output pin is driven low one clock period after a match occurs between the timer and CCPxRA registers. The output pin remains low until a mode change has been made or the module is disabled.
- The timer counts up until it rolls over or until the selected SYNC[4:0] input is asserted, and then it resets to 0000h on the next clock.
- The compare interrupt signal (to set CCPxIF) is asserted, and the output pin is driven low.
- The timer interrupt signal (to set CCTxIF) is asserted for one clock period on a time-base Reset or rollover event.

Figure 28-11. Single Compare Mode (Low Output)



Note:

1. SCEVT has to be cleared to enable the next capture.

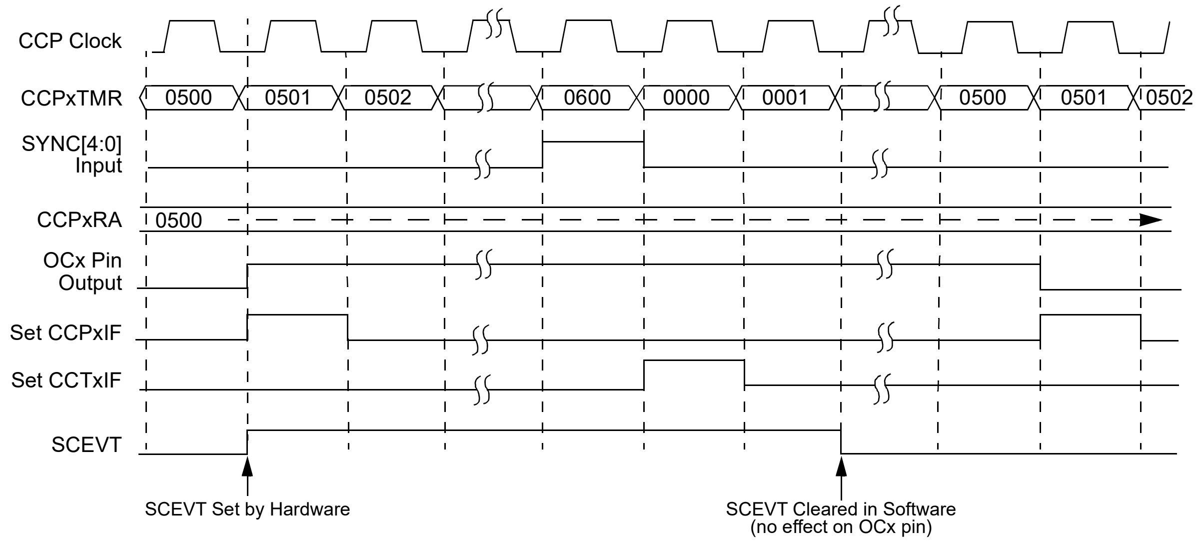
28.4.4.1.3. Single Compare Mode (Toggled Output)

Once this Compare mode has been enabled (Figure 28-12), the output pin is initially driven low and then toggled on each subsequent match event between the timer and CCPxRA register. The key timing events to note are:

- The state of the output pin is toggled one clock period after a match occurs between the timer and CCPxRA registers. The output pin remains at its new state until the next toggle event, until a mode change has been made or the module is disabled.
- The timer counts up until it rolls over, or until the selected SYNC[4:0] input is asserted, and then resets to 0000h on the next clock.
- The respective channel interrupt output (CCPxIF) is asserted when the output pin is toggled.
- The time base interrupt signal (CCTxIF) is generated on a timer Reset or rollover event.

Note: The internal OCx pin output logic is set to a logic '0' on a device Reset; however, the initial output pin state for the Toggle mode can be reversed using the POLACE polarity control bit.

Figure 28-12. Single Compare Mode (Toggle Output)



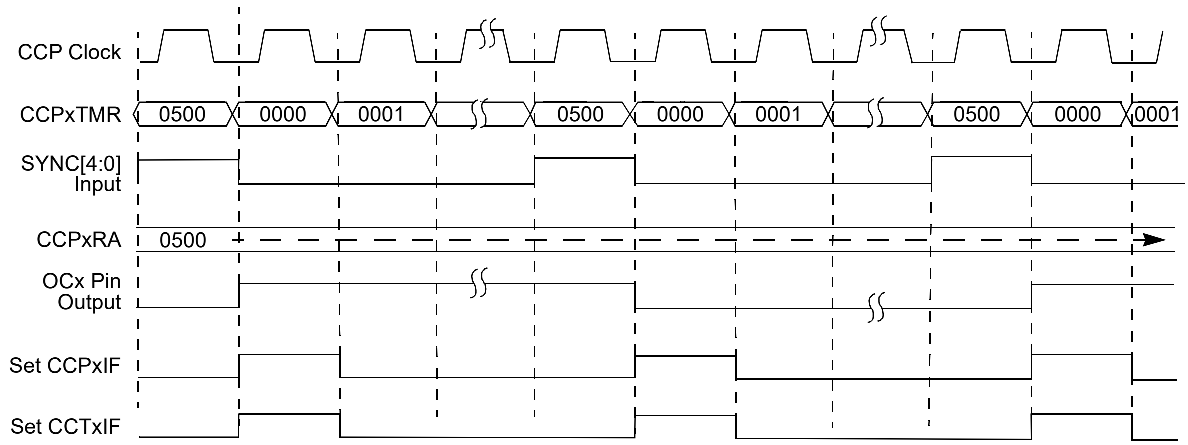
28.4.4.1.4. Special Cases of Single Compare Mode

In Single Edge Compare modes, there are several special cases to consider:

1. When the value of CCPxRA is greater than the timer period, the compare value will always be greater than the timer value. No compare event will ever occur, and the compare output will remain at the initial condition.
2. When the value of CCPxRA equals the timer period, the compare interval is the same as the timer period. Combined with Toggle mode, this can be used to generate a fixed frequency square wave (Figure 28-13).
3. When CCPxRA = 0000h, the timer is held in Reset, either by an asserted trigger source or when CCPTRIG = `0` is the triggered operation. The compare output will remain at the initial condition. The compare output will change once the selected trigger source is deasserted, allowing the timer to operate.
4. If CCPxRA is cleared after a compare event, the SYNC[4:0] signal is asserted and the compare output will remain at its previous state.
5. If, after a compare event, the CCPxRA register is modified to a value greater than the current timer value but less than the timer period, a second compare event will be generated within the count period. The SCEVT bit must also be cleared when MOD[3:0] is '`0001`' or '`0010`' to reset the output pin for the next compare event.

Note: For all special cases, 'timer period' can be defined as either a CCPxPR match or an event by the selected SYNC[4:0] source.

Figure 28-13. Single Compare Mode, Toggle Output, Timer Period = CCPxRA



28.4.4.1.5. Single Edge Output Compare Event Status

The SCEVT bit (CCPxSTAT[3]) indicates the status of a single edge compare event and allows the application to re-arm a single edge compare event without changing the module operating mode or resetting the module. It only functions during single edge compare events; in all other modes, the bit always reads as '0'.

When MOD[3:0] = 0001, the OCx pin is asserted high after a compare event and SCEVT is set to '1' by hardware. The application may clear SCEVT in software. Once the bit is cleared, the OCx pin is reset to a low output and the compare logic is reset to allow the next rising edge compare event.

When MOD[3:0] = 0010, the OCx pin is asserted low after a compare event and SCEVT is set to '1' by hardware. The application may clear SCEVT in software. Once the bit is cleared, the OCx pin is reset to a high output and the compare logic is reset to allow the next falling edge compare event.

When MOD[3:0] = 0011, the OCx pin is toggled after a compare event and SCEVT is set to '1'. The application may clear SCEVT in software, but the state of the OCx pin will not change when the bit is cleared. In this mode, SCEVT only provides event status information and does not affect the OCx pin.

When MOD[3:0] = 0001 or 0010, the application may set SCEVT to '1' to inhibit single edge output compare events. This feature is useful when it is desired to delay an edge event during a particular interval, for example. No changes will occur to the OCx pin during this time. When the SCEVT bit is cleared by software, the OCx output pin will be reset to the initial state, and a rising or falling edge will be generated when the next compare event occurs.

32-Bit Operation with Single Compare Mode

The previous examples all assume 16-bit single compare operations (T32 = 0). Single-Edge Compare modes can also operate with a 32-bit time base, selected by setting T32 = 1. Operation in 32-bit mode is identical, except that the CCPxRB register is paired with CCPxRA to provide a 32-bit compare value for CCPxTMR. CCPxRA is used for the upper 16 bits of the compare value.

No period register is available to set the count period of CCPxTMR. If a count period less than FFFF FFFFh is desired, the module can be synchronized to an external source to set the count period.

28.4.4.2. Dual Edge Compare Mode

When MOD[3:0] = 0100, the Output Compare channel is configured to produce a continuous series of pulses. The parameters for the pulse train are determined by the CCPxRA, CCPxRB and CCPxPRL.

Dual Edge Compare mode is only available in 16-bit mode. The T32 bit has no affect.

Dual Edge Compare mode uses these timer/data registers:

- CCPxTMRL as the Timer register
- CCPxRA for the Rising Edge Value register
- CCPxRB for the Falling Edge Value register
- CCPxPRL for the Timer Period register

Figure 28-14 depicts the signal timing for Dual Edge Compare mode. The typical operation in this mode is as follows:

1. When Dual Edge Compare mode is enabled, the pin state is driven low. At some point, the timer is enabled (triggered) by a hardware or software event to start the count process.
2. Upon the first timer compare match with the Compare register, CCPxRA, the output pin will be driven high.
3. When the incrementing timer count matches the Compare register, CCPxRB, the second and trailing edge (high-to-low) of the pulse is driven onto the output pin. At this second compare, the Output Compare Interrupt flag (CCPxIF) is generated.
4. The Timer Interrupt flag (CCTxIF) is generated, along with the CCP Sync signal, when the timer rolls over (when SYNC[4:0] = `00000`) or when an event is defined by SYNC[4:0].
5. The output pulses continue repeatedly until the mode is terminated by the application or a device Reset occurs.

This is the prototype case, where the timer period and the Match registers are all different values, ordered as Timer Period > CCPxRB > CCPxRA. There are special cases, however, where the conditions differ, resulting in a specific type of output. The cases are listed in Table 28-8.

Figure 28-14. Typical Dual Edge Compare Timing Sequence

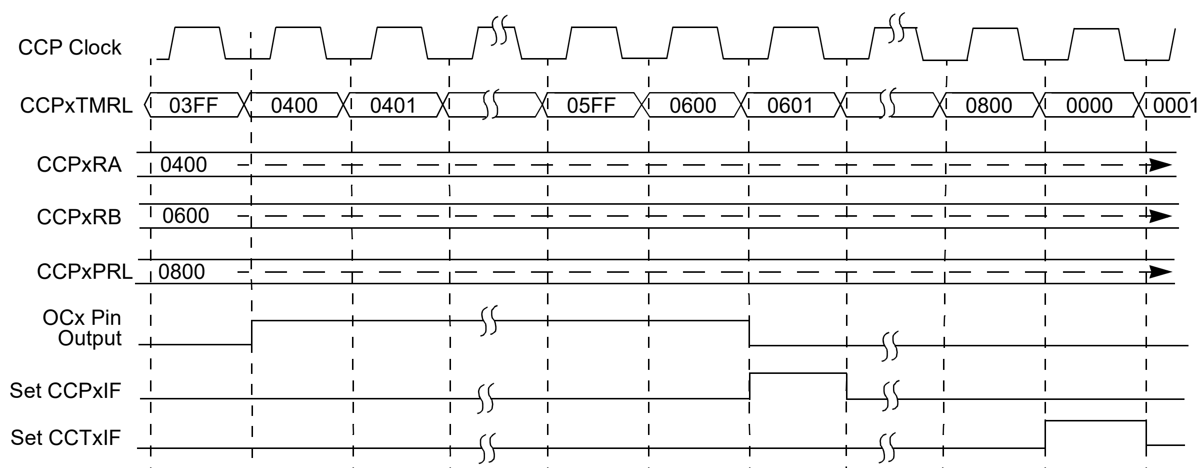


Table 28-8. Special Conditions for Dual Edge Compare Operations

Condition	Output	Output Compare Interrupt on Falling Edge of OCx Pin	Timer Interrupt When Timer Matches Period
CCPxRA = CCPxRB	No output, OCx pin remains low	None	Yes
CCPxRA = CCPxRB + 1	One pulse	Yes	Yes
Timer Period < CCPxRA(1)	No output	N/A(2)	Yes
Timer Period = CCPxRB(1)	OCx goes low at CCPxRB	Yes	Yes
Timer Period < CCPxRB(1)	Continuous high	None	Yes

Table 28-8. Special Conditions for Dual Edge Compare Operations (continued)

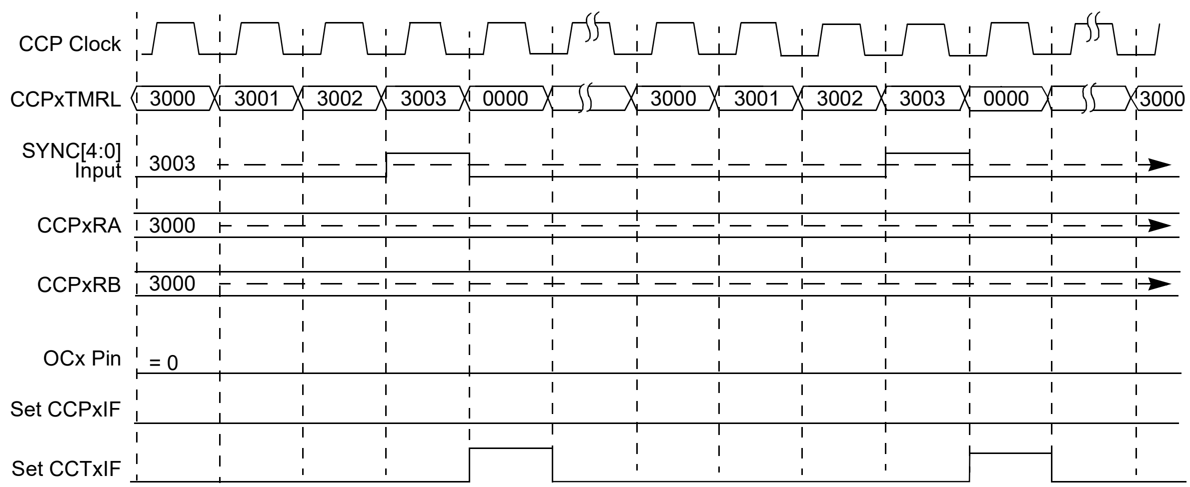
Condition	Output	Output Compare Interrupt on Falling Edge of OCx Pin	Timer Interrupt When Timer Matches Period
Timer Period = CCPxRB, CCPxRA = 0(1)	CCP goes high when TMR = 1 and goes low when CCPxRB matches timer	Yes	Yes
CCPxRA > CCPxRB	Pulse train	Yes	Yes

Notes:
1. The timer period is either the period register value or when the timer is reset by the input selected by SYNC[4:0].
2. If CCPxRB is also less than the timer period, an interrupt will be generated, even though there is no activity on the OCx pin.

28.4.4.2.1. CCPxRA = CCPxRB

If CCPxRA and CCPxRB have the same value, the output is initialized low and stays low; no pulses are generated and no output compare interrupt is generated (Figure 28-15). To put another way, the PWM duty cycle is 0. The Reset/clear-on-CCPxRB match logic overrides the set-on-CCPxRA match logic for a net result of no change in the Output state.

Figure 28-15. Timing for Dual Edge Compare (CCPxRA = CCPxRB)



28.4.4.2.2. CCPxRB = CCPxRA + 1

When the value of CCPxRB is one greater than the value of CCPxRA and both registers are less than the period register, an output pulse that is one CCP clock cycle wide is generated.

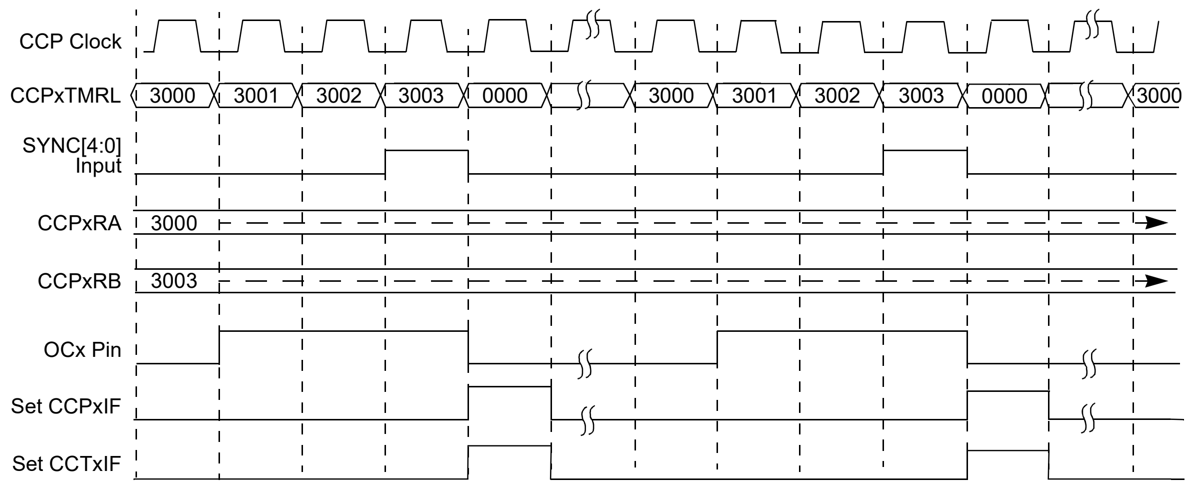
28.4.4.2.3. Timer Period < CCPxRA

When the value of CCPxRA is greater than the timer period, no output pulses are generated.

28.4.4.2.4. Timer Period = CCPxRB

The module will still generate the high-to-low transition when the value of CCPxRB equals the timer period. This is true whether the period is determined in Sync operation from an external source (as shown in Figure 28-16) or on a match with CCPxPRL.

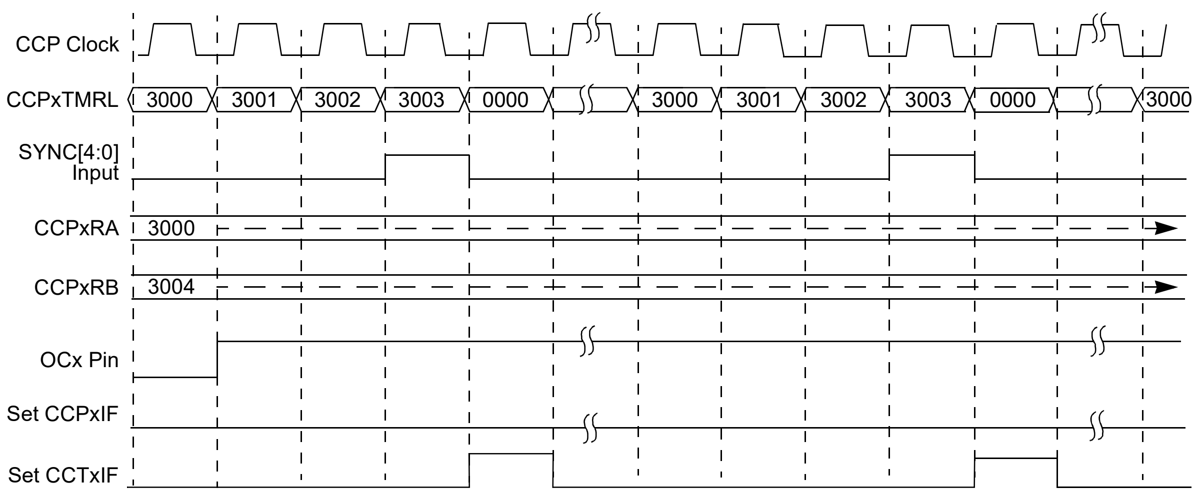
Figure 28-16. Timing for Dual Edge Compare (Timer Period = CCPxRB)



28.4.4.2.5. Timer Period < CCPxRB

If the value of the CCPxPRL is less than that of CCPxRB, but greater than CCPxRA, only one pin transition will be generated until the CCPxRB register contents are changed to a value less than or equal to CCPxPR. No output compare interrupt is generated (Figure 28-17). This condition allows the module to produce a 100% duty cycle output.

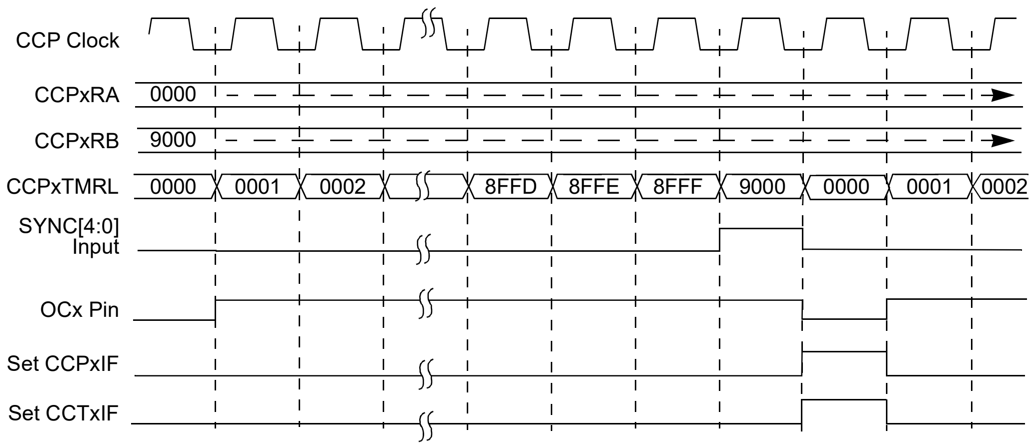
Figure 28-17. Timing for Dual Edge Compare (CCPxPR < CCPxRB)



28.4.4.2.6. CCPxTMRL = CCPxRB and CCPxRA = 0

In Sync operation, if CCPxRA is 0000h, the OCx output is asserted on the first clock after the timer Reset (CCPxTMRL = 0001h). It remains asserted until the value of CCPxRB matches the timer period (when the input selected by SYNC[4:0] is asserted). At this point, the OCx output is deasserted and the CCPxIF is generated on the falling edge (Figure 28-18).

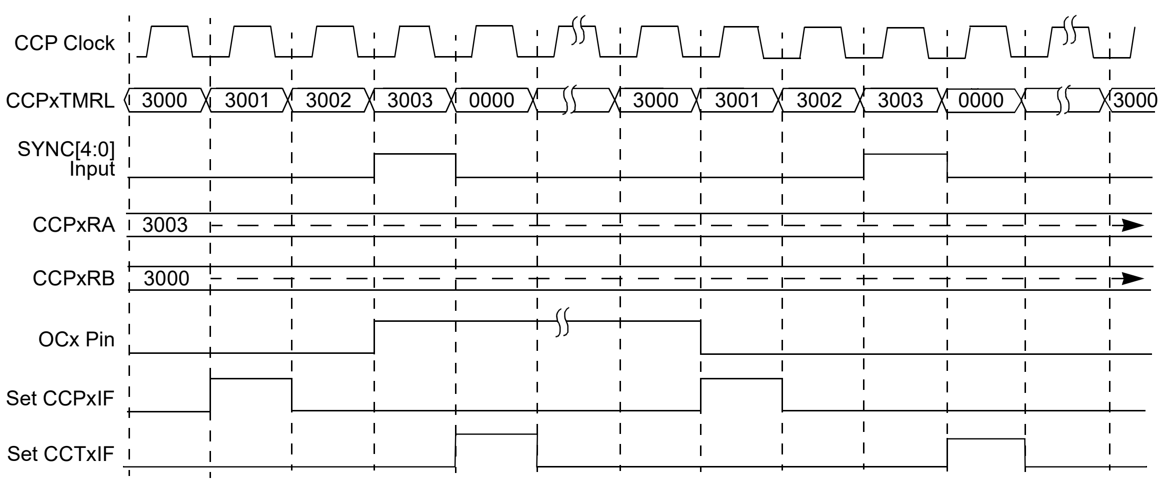
Figure 28-18. Timing for Dual Edge Compare (CCPxRA = 0000h, CCPxRB = Timer Period)



28.4.4.2.7. CCPxRA > CCPxRB

If CCPxRA > CCPxRB, a continuous train of pulses is generated. The timer counts up to the first match (CCPxTMRL = CCPxRA) and the first (rising) edge is generated. CCPxTMRL continues to count up, resetting when the Sync source selected by SYNC[4:0] is asserted. The timer then counts up to the second match (CCPxTMRL = CCPxRB), at which time the second (falling) edge of the signal is generated. The CCPxIF interrupt is generated on the falling edge of the output pulse. The sequence repeats until the module is disabled (Figure 28-19).

Figure 28-19. Timing for Dual Edge Compare (CCPxRA > CCPxRB)



Note: When operating in Dual Compare mode, the CCTxIF signal is asserted on a match between the CCPxRB register value and CCPxTMRL.

28.4.4.3. Dual Edge Buffered Compare (PWM) Mode

When MOD[3:0] = `0101`, the module functions the same as in Dual Edge Compare mode, with the exception that CCPxRA and CCPxRB are double-buffered. In all other respects of output signal generation, operation is the same. Writes to the Data registers (CCPxRA and CCPxRB) are stored in holding buffers. The contents of the buffers are transferred to CCPxRA and CCPxRB on a time-base Reset.

Dual Edge Buffered Compare mode is only available in 16-bit mode. The T32 bit has no effect.

Dual Edge Buffered Compare mode uses these Timer/Data registers:

- CCPxTMRL as the Timer register
- CCPxRA for the Rising Edge Value register of the next period
- CCPxRB for the Falling Edge Value register of the next period
- CCPxPRL for the Timer Period register

The Dual Edge Buffered Compare mode is used to create PWM signals. The buffering of the CCPxRA and CCPxRB registers allows the user to create glitch-free updates to the PWM signal edge times.

If edge-aligned PWM signals are desired, maintain CCPxRA with a value of 0000h. Using a non-zero value for CCPxRA creates PWM signals with arbitrary phase alignments.

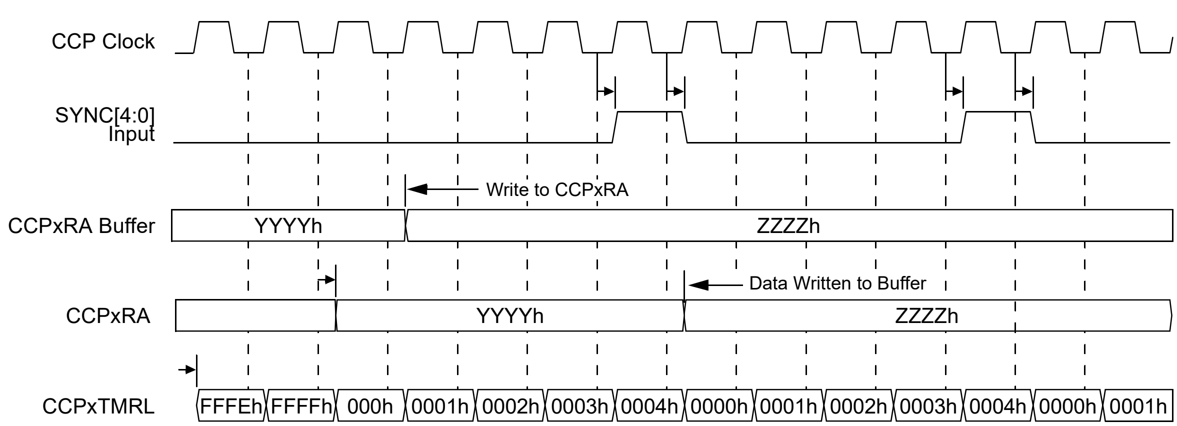
CCPxRA and CCPxRB are double-buffered. Data are written from the buffers to CCPxRA and CCPxRB under these conditions:

- When the timer is reset to 0000h on a sync event (source selected by SYNC[4:0] is asserted).
- When the timer rolls over from FFFFh to 0000h.
- When the module is disabled (CCPON = 0); any writes to CCPxRA and CCPxRB are immediately transferred to their Compare registers.

Figure 28-20 shows the timing for writing to the buffer in Sync operation. CCPxRA and its buffer are shown; CCPxRB and its buffer operate in an identical manner. For output signal generation, refer to Dual Edge Compare Mode.

The procedure for configuring the module for Dual Edge Buffered Compare mode is shown in Example 28-2.

Figure 28-20. Buffer Writes in Dual Edge Buffered Compare Mode



Example 28-2. Setup for Dual Edge Buffered Compare Mode

```
// Set CCP operating mode
CCP1CON1bits.CCSEL = 0;          // Set SCCP operating mode (OC mode)
CCP1CON1bits.MOD = 0b0101;       // Set mode (Buffered Dual-Compare/PWM mode)
//Configure SCCP Timebase
CCP1CON1bits.T32 = 0;            // Set timebase width (16-bit)
CCP1CON1bits.TMRSYNC = 0;        // Set timebase synchronization (Synchronized)
CCP1CON1bits.CLKSEL = 0b000;     // Set the clock source (Tcy)
CCP1CON1bits.TMRPS = 0b00;       // Set the clock pre-scaler (1:1)
CCP1CON1bits.TRIGEN = 0;         // Set Sync/Triggered mode (Synchronous)
CCP1CON1bits.SYNC = 0b00000;     // Select Sync/Trigger source (Self-sync)
//Configure SCCP output for PWM signal
CCP1CON2bits.OCAEN = 1;          // Enable desired output signals (OC1A)
CCP1CON3bits.OUTM = 0b000;       // Set advanced output modes (Standard output)
```

```
CCP1CON3bits.POLACE = 0;          // Configure output polarity (Active High)
CCP1TMR = 0x0000;                 // Initialize timer prior to enable module.
CCP1PR = 0x0000FFFF;              // Configure timebase period
CCP1RA = 0x00001000;              // Set the rising edge compare value
CCP1RB = 0x00008000;              // Set the falling edge compare value
CCP1CON1bits.CCPON = 1;           // Turn on SCCP module
```

28.4.4.4. Output Control for Compare/PWM Modes

When the module operates in an Output Compare mode, the following blocks determine how the Output Compare signal is presented on the output pins:

- Auto-Shutdown Control Block
- Output Polarity Control Block

The auto-shutdown control block responds to asynchronous external inputs or software control, placing all output pins under control of the module into a predetermined state.

The output polarity control block determines the output polarity on each pin under control of the module. This block takes effect after all other control of the output pins.

28.4.4.4.1. MCCP

MCCP has the same features as SCCP, with additional features such as an output mode control block.

The output mode control block is used in the MCCP version of the module to control how the output compare signal is routed to the six available output pins. The output mode control block is not needed on the SCCP, because only one output pin is available to the module.

Output Mode Selection

The CCPxCON3.OUTM[2:0] control bits are used to select the Output mode of the MCCP.

When operating in an Output Compare mode (CCSEL = 0), one of several Output modes may be selected that use the OCxA – OCxF output pins in different ways. In some Output modes, a dead-time delay generator is used to implement switching delays between the output pins.

The output control logic does not determine how the input signal is generated, only how the signal is routed to the output pins. The signal source for the output control logic can be any of the output compare Operating modes that can be selected by the MOD[3:0] control bits.

The output control logic does interact with the input signal generation logic for synchronization purposes. In some Operating modes, the output control logic will wait for an input signal period boundary to switch the signal to a different output pin.

These output modes can be selected using the OUTM[2:0] control bits:

- Steerable Single Output mode (default)
- Brush DC Output mode, forward and reverse
- Half-Bridge Output mode
- Push-Pull Output mode
- Output Scan mode

Output Enable Control

Each of the output pins controlled by the MCCP module may be enabled separately using the CCPxCON2.OCxEN control bits. If one of the OCxEN control bits is set, then that I/O pin receives the Output Compare signal that is generated by the module. The signal generated on the pin is a function of the Output Mode Control bits, OUTM[2:0]. If the OCxEN Control bit is cleared, then the I/O pin is controlled by the port logic or another peripheral of higher priority. The user must use care to ensure that the I/O pin will be in the correct state when a OCxEN Control bit is cleared.

Note: The CCPxCON2.OCAEN bit will reset to '1' by default. This configures the CCP module to use the OCxA output pin for output compare functions by default, simplifying software configuration. The user may enable other pins in software as needed using the OCxEN control bits.

The OCxEN Control bits have no effect on a module operation when the module is operated in an Input Capture mode (CCM = 1) or a Timer mode (CCM = `0` and MOD[3:0] = `0000`).

The OCxEN control bits can be used in different ways depending on the output mode selected by the OUTM[2:0] control bits. The OCxEN bits can provide a steering function to redirect the Output Compare signal to different pins at specific times. This steering functionality is useful in motor and power control applications.

The OCxEN bits can also be used to relocate the module output signals to different sets of output pins. For example the Half-Bridge Output mode replicates the same pair of signals on the OCxA/OCxB, OCxC/OCxD and OCxE/OCxF pins. The user can enable any of these pin pairs using the OCxEN bits to move the signals to a convenient location.

Steerable Single Output Mode

Steerable Single Output mode is the default output mode of the output control logic and is selected when CCPxCON3.OUTM[2:0] = `000`. In this Operating mode, the signal produced by the output compare logic is routed to all available module output pins. User software can enable each output pin separately to produce the output compare signal by setting the appropriate OCxEN Control bit.

Push-Pull Output Mode

The Push-Pull PWM mode is selected when CCPxCON3.OUTM[2:0] = `001`. In Push-Pull PWM mode, the output compare signal is multiplexed between the OCxA and OCxB output pins on alternate timebase cycles. For each time-base cycle, one of the pins is connected to the output compare signal and the other pin is driven to the Inactive state.

The output and port control signals for the OCxA/OCxB pin pair are replicated for the OCxC/OCxD and OCxE/OCxF output pins in this Push-Pull mode. This allows the user to move the push-pull output signals to another pin pair using the OCxEN control bits. The user must set at least one pair of OCxEN control bits to allow the module to control two output pins.

This mode is commonly used to drive transformers in DC/DC and DC/AC power supplies. As shown in Figure 28-22, each output pin drives one side of the transformer winding through an external power transistor. The transformer has a center winding that is connected to a DC bus voltage. The module must produce the same pulse-width for each side of the transformer to prevent DC current flow in the transformer winding. Therefore, the duty cycle must remain the same for two time-base periods. A four-transistor push-pull circuit may also be used as shown in Figure 28-23. Note that a second output pin pair has been used. In addition, the connections to the second pair of transistors have been intentionally swapped so that each pair of diagonal transistors is on at the same time.

Interrupts from the time base Reset events are automatically post-scaled 2:1 in Push-Pull PWM mode so that the user does not have to track the number of time-base periods that have elapsed.

Note: Push-Pull Output mode must be used with a buffered Output Compare mode. The output control logic inhibits buffer updates for the CCPxRA and CCPxRB registers on every other time-base cycle as shown in Figure 28-21. If a buffered Output Compare mode is not selected, the Push-Pull Output mode will be unable to maintain the same duty cycle across two time-base periods.

Note that in Figure 28-21, the module also produces output compare interrupt events in this mode. These events are a result of the time base comparison with the CCPxRA register, which produces the PWM falling edge. If the user requires more frequent interrupts, the output compare interrupt events may be monitored instead of the timer interrupt events.

Dead-Time Delay for Push-Pull Mode

The dead-time delay generator provides optional signal blanking in the Push-Pull Output mode. If a non-zero value is loaded into the DT[5:0] bits, the dead-time delay generator will delay the rising edge of the output compare signal on both output pins until the delay time has expired. This avoids shoot-through in full-bridge applications when the duty cycle is near 100.

Note: The dead-time delay generator provides a blanking time that occurs at the time-base period boundary (sync_trig_in = 1). If the output compare signal rising edge occurs during this blanking time, it will be delayed until the dead-time counter expires. If the rising edge of the output compare signal occurs after the dead-time counter blanking time, then it will not be affected by the dead-time circuit. This scenario would happen, for example, when the MCCP module is operated in Center Aligned mode and the duty cycle is small

Figure 28-21. Push-Pull PWM Timing

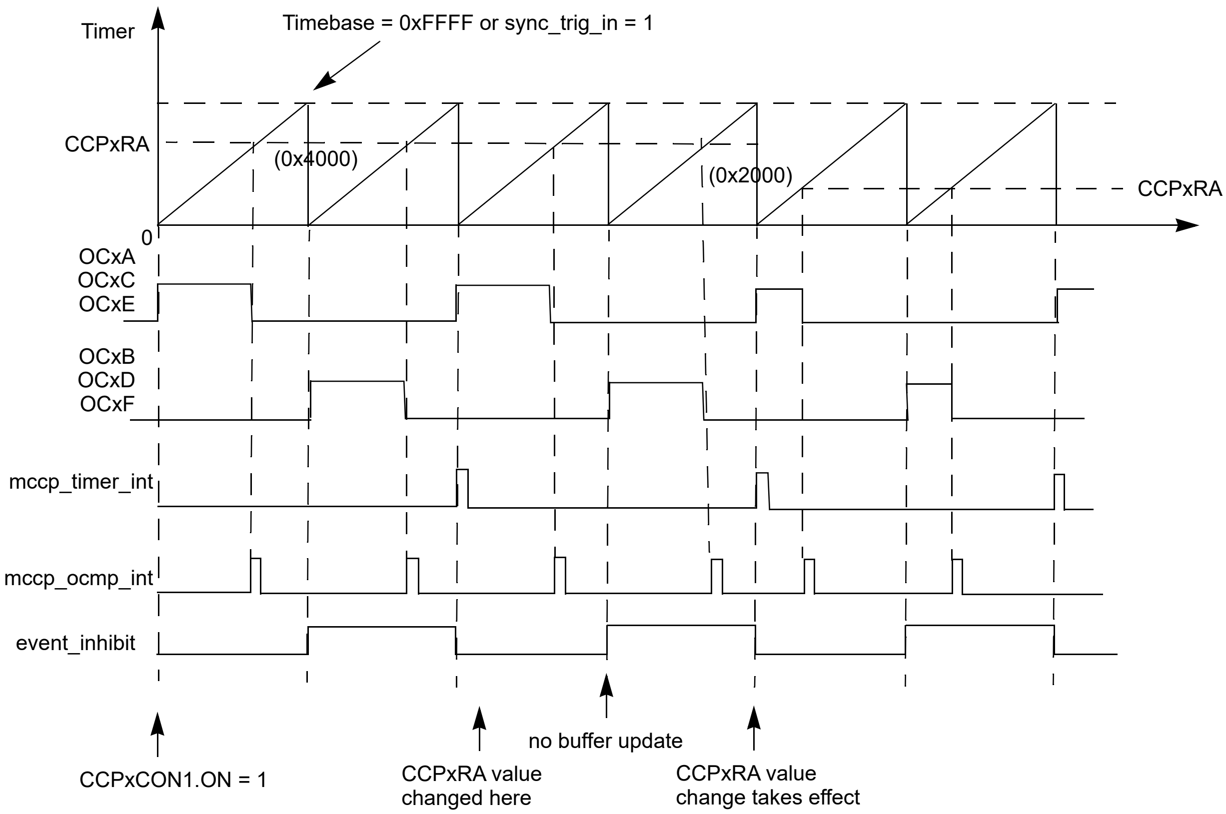


Figure 28-22. Typical Push-Pull Application

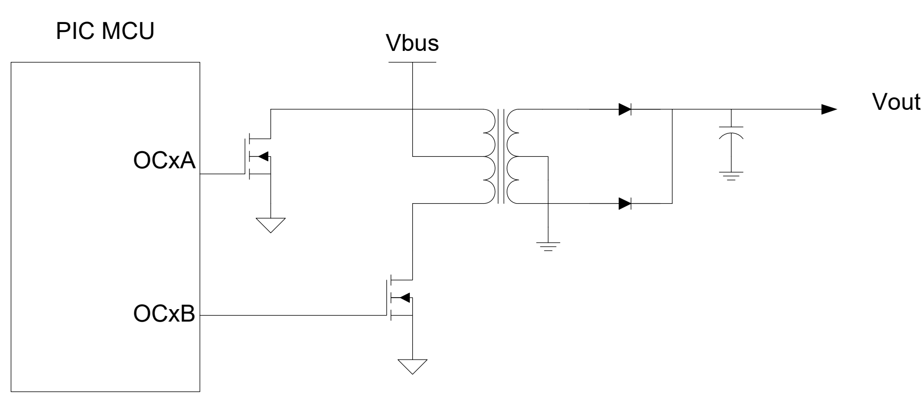
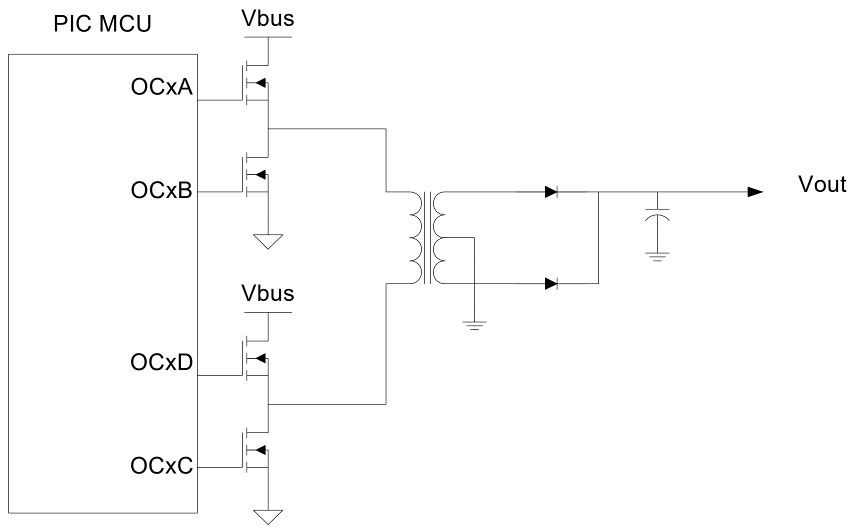


Figure 28-23. Typical Full-Bridge Push-Pull Application



Half-Bridge Output Mode

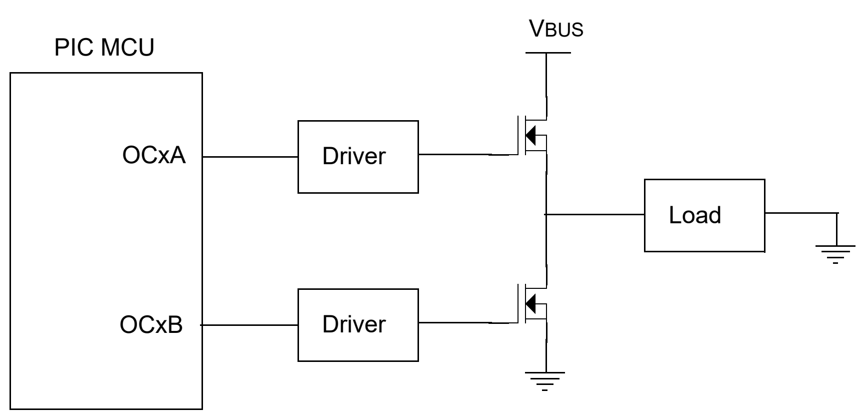
The Half-Bridge Output mode is selected when CCPxCON2.OUTM[2:0] = `010`.

In this mode, two device output pins are controlled. The module produces complementary output signals on OCxA and OCxB. The OCxB signal is the inverse of the OCxA signal, and dead-time delay, if non-zero, is inserted between the switching events of the two pins. The output and port control signals for the OCxA/OCxB pin pair are replicated for the OCxC/OCxD and OCxE/OCxF output pins in this Half-Bridge mode. This allows the user to move the complementary output signals to another pin pair using the OCxEN control bits. The user must set at least one pair of OCxEN control bits to produce half-bridge output signals.

The Half-Bridge PWM mode is typically used to control power circuits like the one shown in Figure 28-24.

If a non-zero dead-time value is programmed into the CCPxCON3.DT[5:0] bits, then a delay is inserted between the switching edges of the OCxA and OCxB signals.

Figure 28-24. Typical Half-Bridge Application



Brush DC Output Modes

The Brush DC Output modes are selected when CCPxCON3.OUTM[2:0] = `101` (Forward mode) or CCPxCON3.OUTM[2:0] = `100` (Reverse mode).

In this mode, the signal produced by the output compare logic is routed to four output pins: OCxA, OCxB, OCxC and OCxD. The user must set the OCAEN, OCBEN, OCCEN and OCDEN control bits in software to allow the module to control these output pins. For each mode, only two of the four output pins are driven to the Active state:

- Forward Mode – CCPxCON3.OUTM[2:0] = `101`
  - OCxA pin receives a PWM generator signal.
  - OCxD pin is driven to the Active state.
  - OCxB and OCxC pins are driven inactive.
- Reverse Mode – CCPxCON3.OUTM[2:0] = `100`
  - OCxC pin receives a PWM generator signal.
  - OCxB pin is driven to the Active state.
  - OCxA and OCxD pins are driven inactive.

The OCxE and OCxF output pins are not controlled by the module in the Brush DC modes. The user may enable these pins using the OCEEN and OCFEN control bits. However, the pins will remain in the Inactive state.

Figure 1-5 shows how the four pins are connected and used to control external circuitry in a typical application. The actual polarity of the four output signals is determined by the output polarity control circuitry.

Dead-Time Delay for Brush DC Modes

Dead time is not required for the Brush DC modes, except during a direction mode change when the duty cycle is at or near 100%. It is expected that the user will switch between Forward mode and Reverse mode during run time. A direction change is accomplished by toggling the OUTM[0] bit in the application software when OUTM[2:0] = `101` or OUTM[2:0] = `100`.

Note: The module logic needs to detect when the OUTM[2:0] Control bits are changed between the values of 100 and 101. A change between these values will trigger the dead-time generator.

The direction change is synchronized to the CCP time-base period and occurs when sync_trig_in = 1.

When a direction change is made with the PWM Generator set for a low duty cycle, dead time is not required because the actively controlled switches will be turned off for a period of time before the direction change occurs. When the duty cycle is near 100%, dead time may be required to ensure that the top and bottom switches controlled by the module will be off for a minimum time.

When the module is in one of the Brush DC modes and the OUTM[0] is toggled in software to change direction, the following chain of events occurs:

1. At the next PWM time-base Reset boundary, the two currently active pins (OCxA and OCxD or OCxB and OCxC) are driven to their Inactive states.
2. If the value of the DT[5:0] bits is zero, then the new pair of output pins is made active immediately.
3. If the value of the DT[5:0] bits is non-zero, then DT[5:0] is loaded into the dead-time delay counter when the MOD[0] bit is toggled. The new pair of output pins is made active after the dead-time counter expires.

Note that dead time is to be inserted only when the following occurs:

- The present value of OUTM[2:0] = `100` or `101`,

AND

- OUTM[0] is toggled by user software.

When the above conditions are true, the dead time blanking will occur when sync_trig_in = 1. Figure 28-26 shows a timing diagram of a direction change when the dead time is programmed to a value of two cycles. Prior to the direction change, the OCxA pin receives the PWM Generator signal. The duty cycle is programmed to a value near 100%, and the output compare pulse terminates one

CCP clock cycle prior to the end of the time-base period. The OCxD pin is driven active. At the PWM period Reset (sync_trig_in = 1), all four output pins are driven inactive for the duration of the dead-time count. After the dead-time delay, the OCxB pin is driven active and the OCxC pin receives the Output Compare signal.

Figure 28-25. Typical Brush DC Application

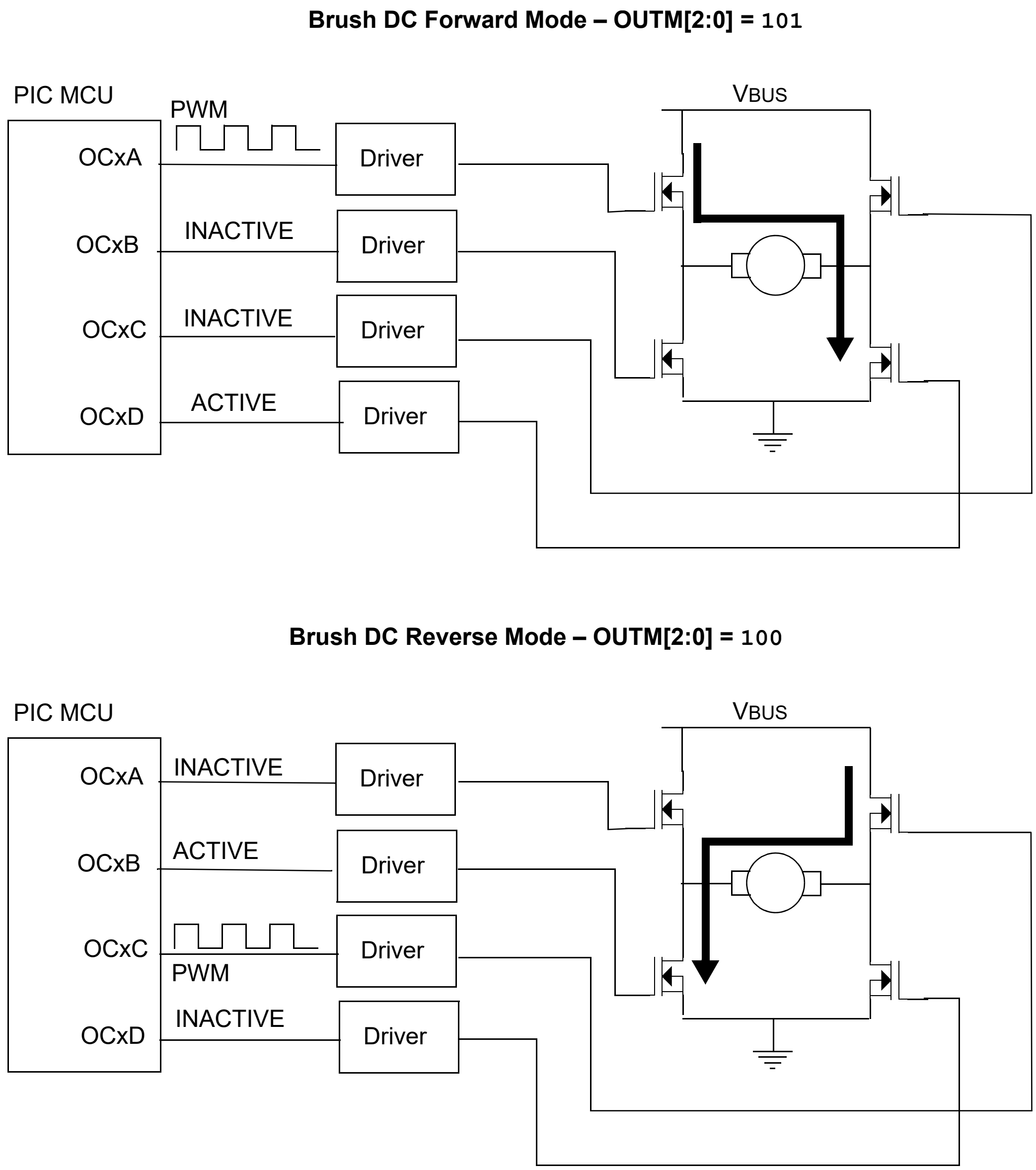
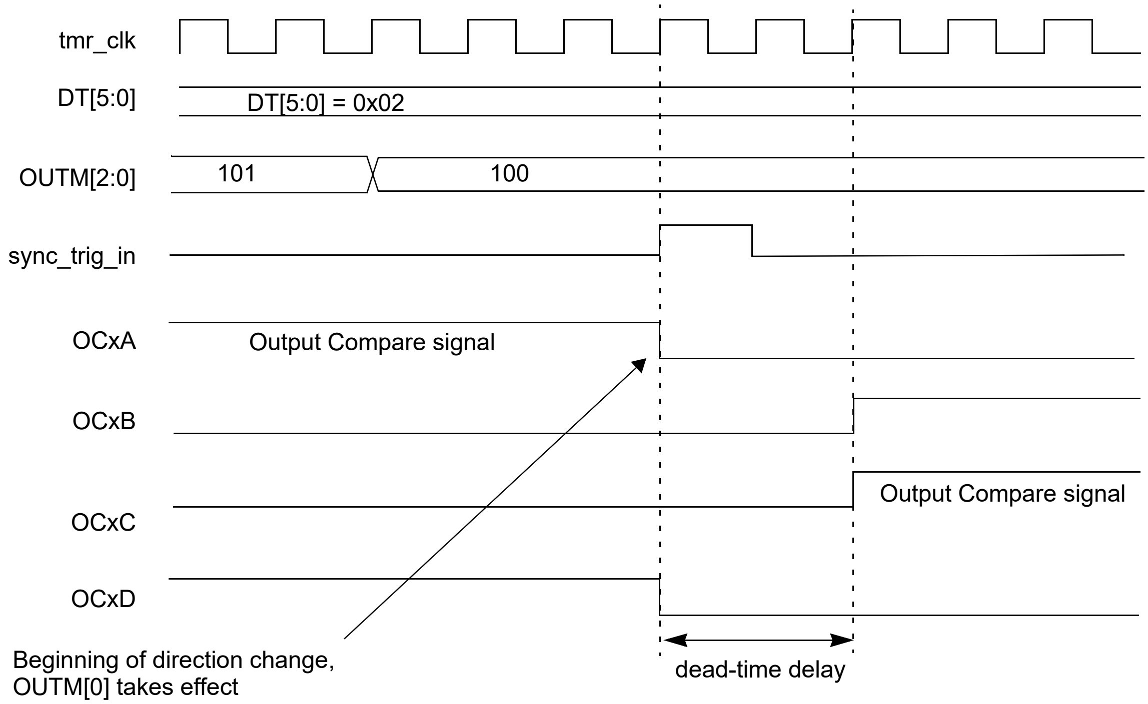


Figure 28-26. Dead-Time Generation For Brush DC Direction Change



Output Scan Mode

The Output Scan mode is selected when CCPxCON3.OUTM[2:0] = `110`.

The Output Scan mode is similar to the Single Output Steerable mode, except that the Output Compare signal can be automatically sequenced among the available output pins. The MCCP output pins to be used during output scan are selected by setting the appropriate OCxEN Control bit in the CCPxCON2 register. If the OCAEN, OCBEN and OCEEN bits are set, for example, the MCCP module will automatically steer the Output Compare signal between the OCxA, OCxB and OCxE output pins.

If CCPxCON1.ON is set to '0', the sequence logic is reset, and the scan sequence will begin on the first enabled output pin after ON = 1. When CCPxCON1.ON = 1 and the time base is triggered or reset (sync_trig_in = 1), the Output Compare signal is moved to the next enabled output in the sequence.

The basic operation of the Output Scan mode is shown in Figure 1-7 and Figure 1-8 for the case of three enabled outputs. The operation is the same in each case, except for the state of each pin during an inactive time-base period. Note that the actual state of the output pin before the module is enabled will depend on the PORT control logic settings or an enabled peripheral of lower priority.

Operation with Synchronized Timer

When the time base operates in a Synchronized mode (TRIGEN = 0), the module switches to the next output pin in the sequence when sync_trig_in = 1. The sequence is reset to the first enabled output pin when CCPxCON1.ON = 0.

Operation with Triggered Timer

When the time base operates in the Triggered mode (TRIGEN = 1), the module switches to the next output pin in the sequence when the time base rolls over from `FFFF` to `0000` or a match with the CCPxPR register occurs.

If the One-Shot Triggered mode is used, one or more output pins may be scanned per the trigger input depending on the settings of the OSCNT[2:0] control bits. If OSCNT = `000`, the time base will count for only one cycle, then stop until another trigger is received. Therefore, the output scan logic will produce Output Compare signals on one output pin and then step to the next output pin in the scan sequence. The next output pin will not be driven until another trigger input has been received.

If OSCNT > 0, the time base will count for multiple cycles when a trigger input is received and produce Output Compare signals on multiple pins in the scan sequence. If the OSCNT value is not large enough to allow all enabled pins to be scanned during one trigger event, then the scan sequence will resume where it left off on the next trigger event.

The CCPxCON3.OETRIG bit selects the Inactive state of each MCCP output pin that participates in a triggered scan sequence. When OETRIG = 0, the output pin is driven to the Inactive state when it does not receive the Output Compare signal. When OETRIG = 1, the output pin is tri-stated when it does not receive the Output Compare signal

Figure 28-27. Output Scan Mode, TRIGEN = 0

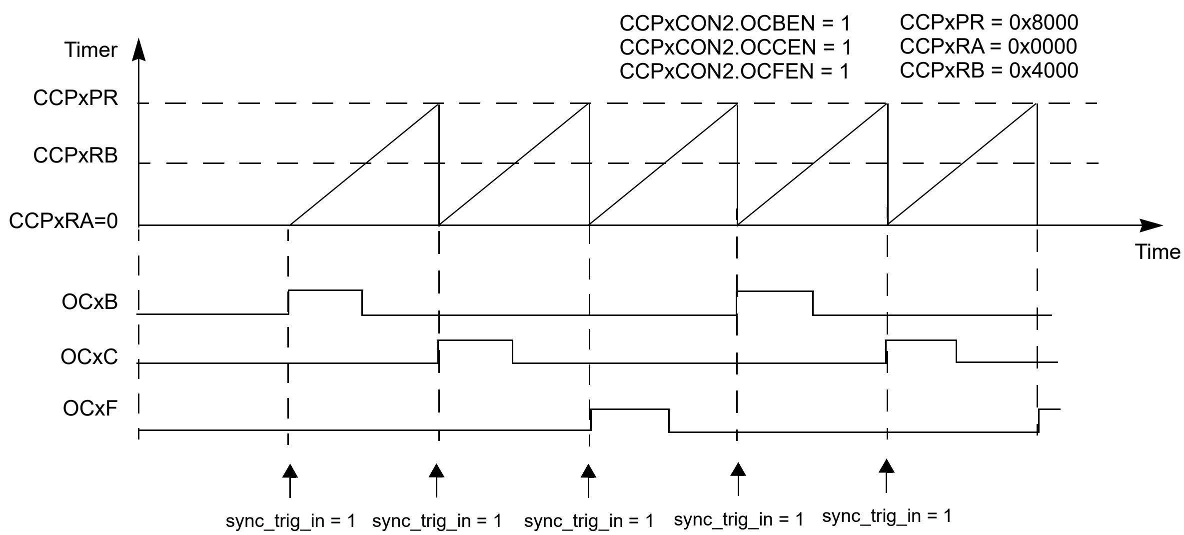


Figure 28-28. Output Scan Mode, OETRIG = 1, TRIGEN =1, ONESHOT = 1, OSCNT = 000

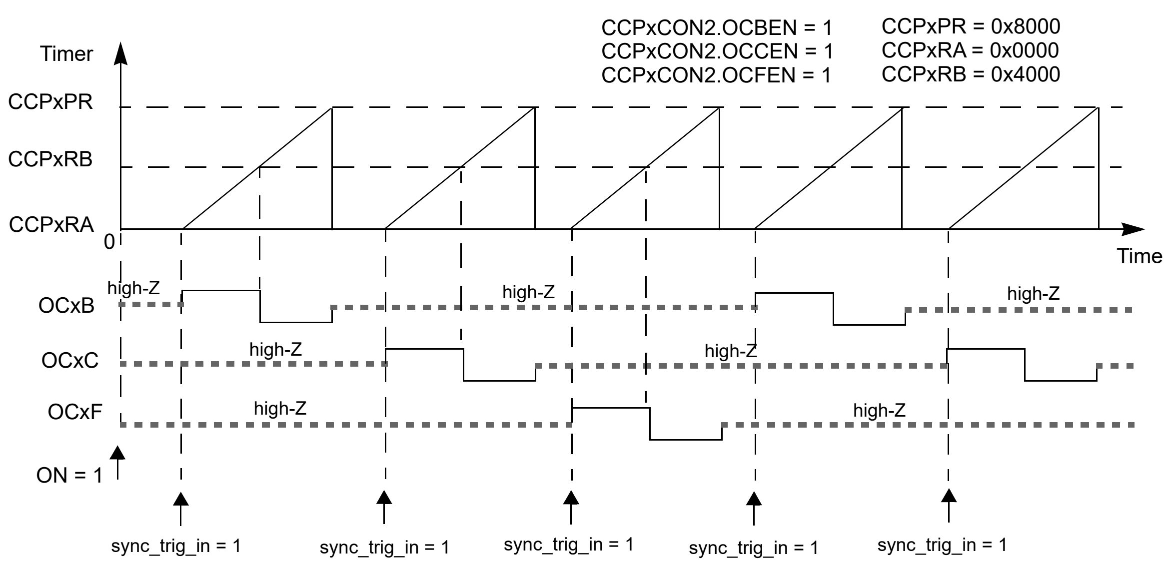


Figure 28-29. Output Scan Mode, TRIGEN = 1, ONESHOT = 1, OSCNT = 010, OETRIG = 0

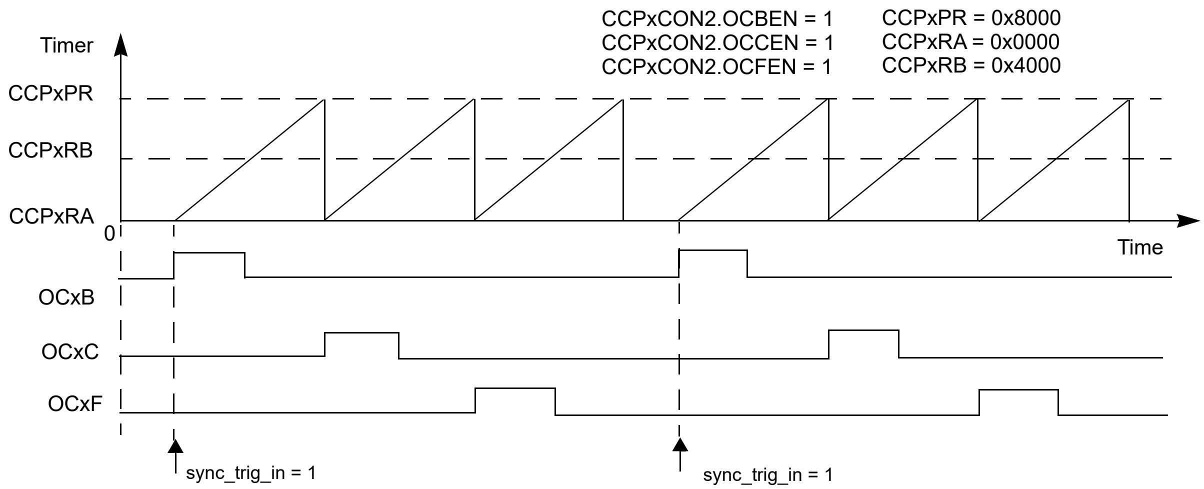
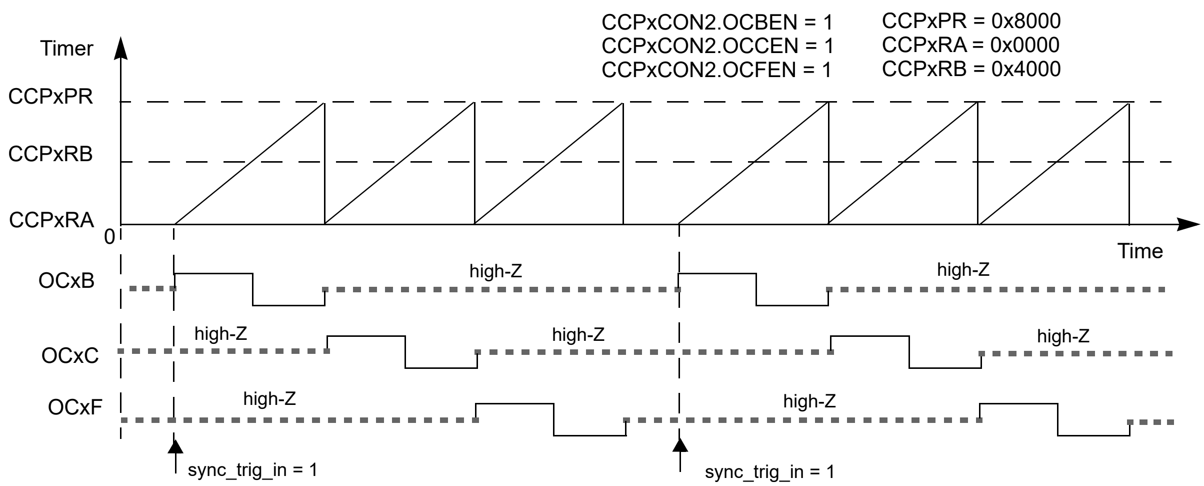


Figure 28-30. Output Scan Mode, TRIGEN = 1, ONESHOT = 1, OSCNT = 010, OETRIG = 1



Dead-Time Delay Generator

This section describes the dead-time delay generator that is available on the MCCP module. It is used in specific output modes of the module. The purpose of the dead-time delay generator is to:

- Create two output signals, true and complement, from the single output signal of a PWM generator.
- Provide a brief delay between the time when one signal is driven inactive and the other signal is driven active.

The dead-time delay generator contains edge detectors to monitor input signal transitions and a digital countdown timer. Each time an input signal edge occurs, the countdown timer is loaded with a programmable countdown value specified by the CCPx-CON3.DT[5:0] control bits. The rising edge of the inactive signal is delayed until the countdown timer counts to zero. If DT[5:0] = 0, the dead-time delay generator is effectively disabled and complementary output signals are produced with zero delay between the transitions on each output. A timing diagram for the dead-time delay generator is provided in Figure 1-11.

Note: When the time base operates from a clock source other than the system clock and TSYNC = 0, user software should not modify the DT[5:0] bits while ON = 1; unexpected results may occur.

There are special cases for dead-time generation:

1. When the input signal pulse width is equal to or less than the programmed dead-time value, the desired output results will not be obtained. Figure 1-12 depicts two scenarios where the duty cycle of the input signal is close to 0% and close to 100%. When the duty cycle is close to 0%, both outputs are driven inactive during the dead-time delay period. The out_high signal does not get driven active for any period of time. When the duty cycle is near 100%, both outputs are driven inactive for the dead-time period, and the out_low signal does not get driven high for any period of time. These are regions of operation that should be avoided by the user.
2. When the input duty cycle is set to 0%, there will be no transitions in the input signal. The out_high signal should remain LOW, and the out_low signal should remain HIGH.
3. When the input duty cycle is set to 100%, there should be no transitions in the input signal. The out_high signal should remain HIGH, and the out_low signal should remain LOW.

As the input duty cycle approaches 0% or 100%, the dead-band delay time becomes a more significant portion of the input signal pulse width. This effect causes a non-linear behavior between the requested duty cycle and the actual system response. Systems that are sensitive to this non-linearity must avoid regions near 0% and 100% duty cycle or use output feedback to compensate for the duty cycle.

Dead-Time Delay Generation for Half-Bridge Mode

When the CCP module is operated in Half-Bridge mode, the OCxA output pin signal is the non-inverted output from the dead-band delay generator. The OCxB output pin signal is the inverted output from the dead-band delay generator. Polarity control is provided after the dead-time delay generator block for these two signal outputs.

The output and port control signals for the OCxA/OCxB pin pair are replicated for the OCxC/OCxD and OCxE/OCxF output pins in this Half-Bridge mode. This allows the user to move the complementary output signals to another pin pair using the OCxEN control bits.

Dead-Time Delay Generation for Brush DC Modes

For the Brush DC operating modes, the dead-time delay generator is used to optionally blank the OCxA, OCxB, OCxC and OCxD output pin signals during a toggle of direction.

The OCxE and OCxF output pins are not controlled by the module in the Brush DC modes. Their respective output enable signals should be driven low.

Dead-Time Delay Generation for Push-Pull Mode

When the CCP module is operated in Push-Pull mode, the dead-time delay generator is used to optionally blank the OCxA – OCxF output pins at timebase period boundaries

Polarity control is provided after the dead-time delay generator block for these six signal outputs

Dead-Time Delay Generation for Output Scan Mode

The dead-time delay generator is not used in Output Scan mode (OUTM[2:0] = `110`).

Figure 28-31. Dead-Time Delay Generator Simplified Block Diagram

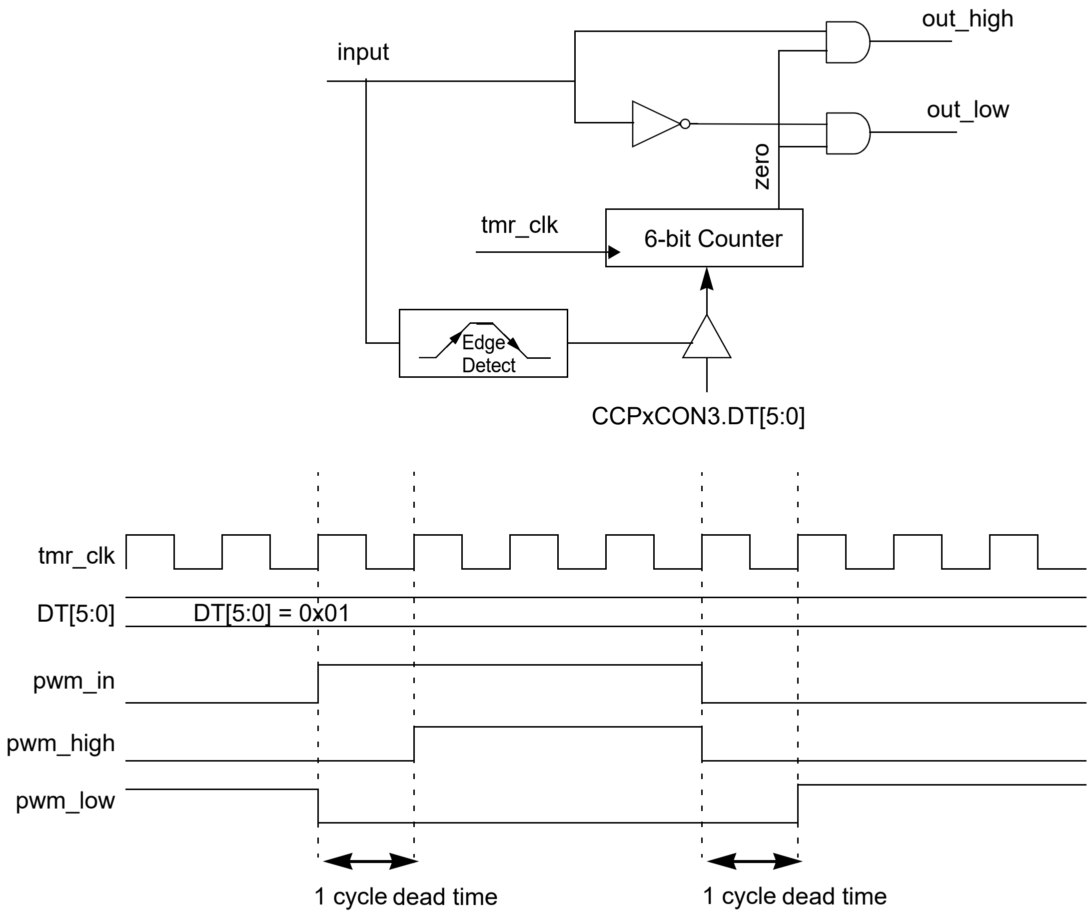
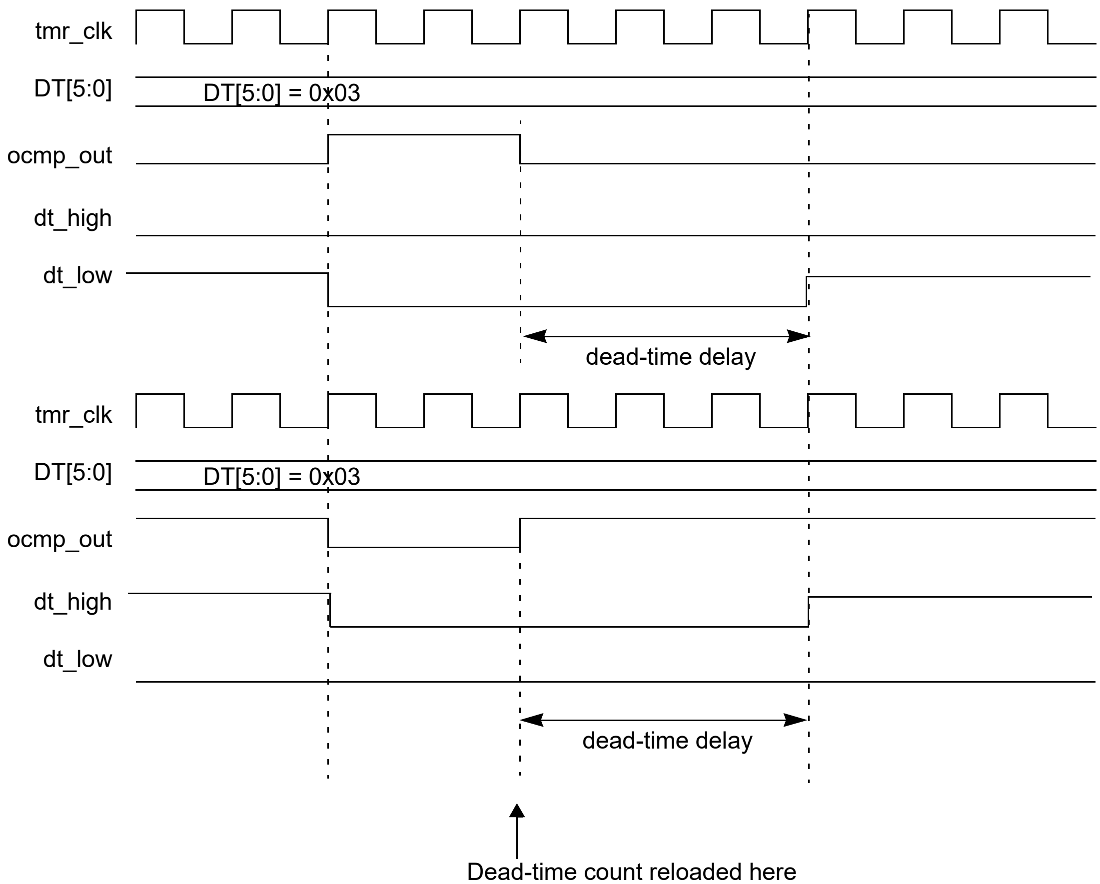


Figure 28-32. Dead-Time Delay Generator Short PWM Pulse-Width



28.4.4.4.2. Output Pin Enable (SCCP)

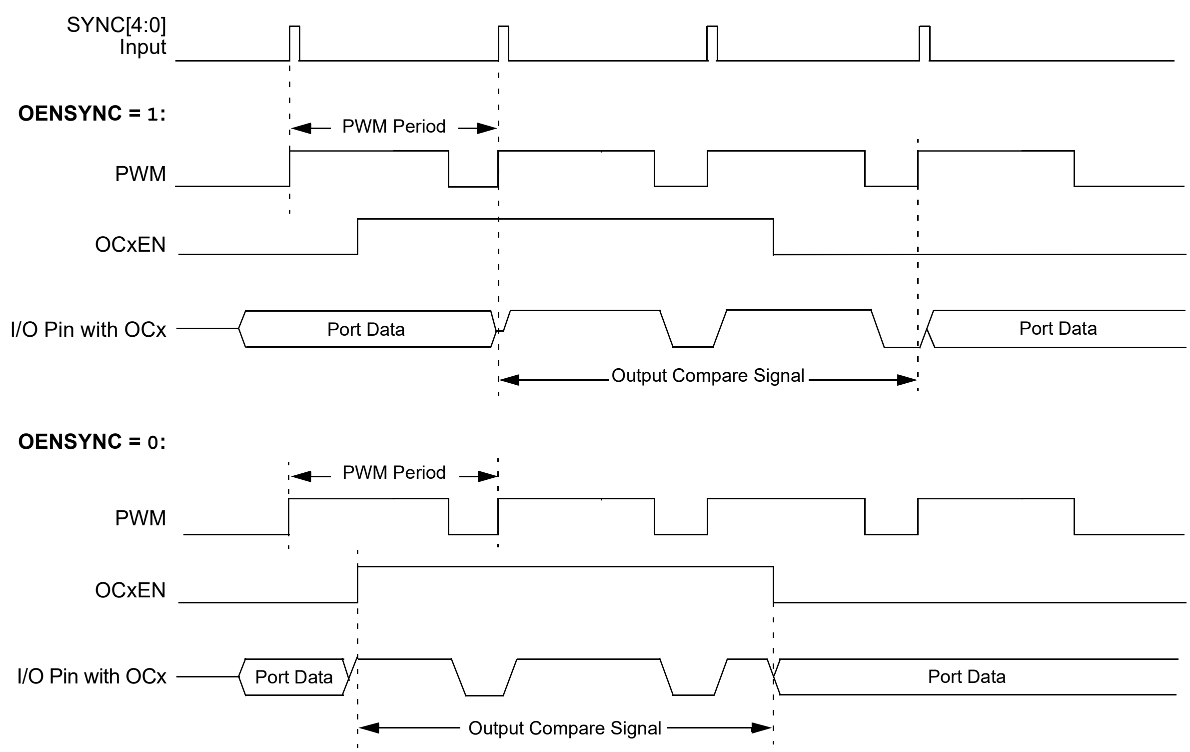
The SCCP module has only one output pin, OCxA, in Output Compare or PWM mode. The OCAEN bit (CCPxCON2[24]) is the only implemented control bit. It determines whether or not the module has control of the output pin. By default, this OCxA output is enabled on device Resets.

28.4.4.4.3. Output Enable Synchronization

Changes to the OCAEN control bit may be optionally synchronized to the timer period, allowing software changes to PWM settings be synchronized to the PWM period's boundaries. This prevents incomplete output pulses as a result of steering changes.

The OENSYNC control bit (CCPxCON2[31]) controls the synchronization of the PWM output to period boundaries. When OENSYNC = `1`, changes to the OCAEN control bit take effect on a timer Reset (i.e., the sync input selected by SYNC[4:0] is asserted). When OENSYNC = `0`, changes to the OCAEN control bit take effect immediately. Figure 28-33 shows the effect of the OENSYNC bit on the I/O pin shared by the OCx output.

Figure 28-33. OENSYNC Bit Operation



28.4.4.4.4. Output Enable on Trigger

The OETRIG control bit (CCPxCON3[31]) allows the user to select whether the CCP output pins are held in a High-Impedance state or driven by the module before the timer is triggered. The OETRIG control bit only affects the module operation in Triggered mode (TRIGEN = `1`).

The operation of the OETRIG bit function varies depending on the Output mode of the module.

The output pin is held in a High-Impedance state when the timer is not triggered (CCPTRIG = `0`).

28.4.4.4.5. Auto-Shutdown Control

The primary function of the auto-shutdown control logic is to place the module output pins in a Safe state when driving external power circuitry. The auto-shutdown function can also be used to place the output pins of the CCP module in a specific state based on an external event.

Auto-shutdown control is implemented as part of the time-base gating (see Gating Logic). The user must select an input source for the auto-shutdown using the ASDG[7:0] Control bits (CCPxCON2[7:0]). The available sources for auto-shutdown are device-dependent and typically include such sources as comparator outputs, I/O pins, software control (i.e., the SSDG bit) and so on. With the exception of the SSDG control bit, which is active-high, a low output from the shutdown source places the module OCA pins in the Shutdown state. The auto-shutdown event is level-sensitive, not edge-triggered. The comparator output and other shutdown sources are not synchronized to the system clocks to provide an immediate response of the CCP module to the shutdown input signal.

When a shutdown occurs, the selected output states are placed onto the module port pins.

Table 28-9. Auto-Shutdown and Gating Sources

ASDG[7:0]	Auto-Shutdown/Gating Source
`1xxx xxxx`	OCFB

Table 28-9. Auto-Shutdown and Gating Sources (continued)

ASDG[7:0]	Auto-Shutdown/Gating Source
x1xx xxxx	OCFA
xx1x xxxx	CLC1
xxx1 xxxx	ICMn
xxxx 1xxx	OCFD
xxxx x1xx	OCFC
xxxx xx1x	Comparator 2 Output
xxxx xxx1	Comparator 1 Output

Auto-Shutdown Pin State

The state of the output pins is controlled by the PSSA[1:0] control bits (CCPxCON3[19:18]). The PSSAx bits affect the states of the OCxA Output pin. These control bits let the user select whether the I/O pin is driven inactive, driven active or put into a High-Impedance state.

Software Shutdown

The user application may invoke a shutdown event at any time by setting the SSDG control bit (CCPxCON2[12]). This bit behaves exactly like an external shutdown source, except that the polarity of the control bit is inverted. A shutdown event will be caused whenever the SSDG bit is set. The module output pins go to their programmed shutdown state and remain in that condition until the SSDG bit is cleared in software. The software shutdown feature may be used by itself or in parallel with an external source.

Notes:

1. The user may also need to clear the ASEVT status bit if automatic restarts are not enabled.
2. Any enabled shutdown source selected by the ASDGx bits and the SSDG software shutdown bit has priority over a software write to the ASEVT bit. The Fault condition cannot be exited unless all shutdown sources are inactive.

Auto-Shutdown Status

The ASEVT status bit (CCPxSTAT[4]) indicates the status of a shutdown event. If the ASEVT bit is cleared, the output pins associated with the CCP module will have normal activity.

If the ASEVT bit is set, then the output pins will be driven to their Shutdown states or held in a High-Impedance state. The ASEVT bit can also be used as a control bit to manually reset the shutdown condition, as described in Automatic Restart Enable.

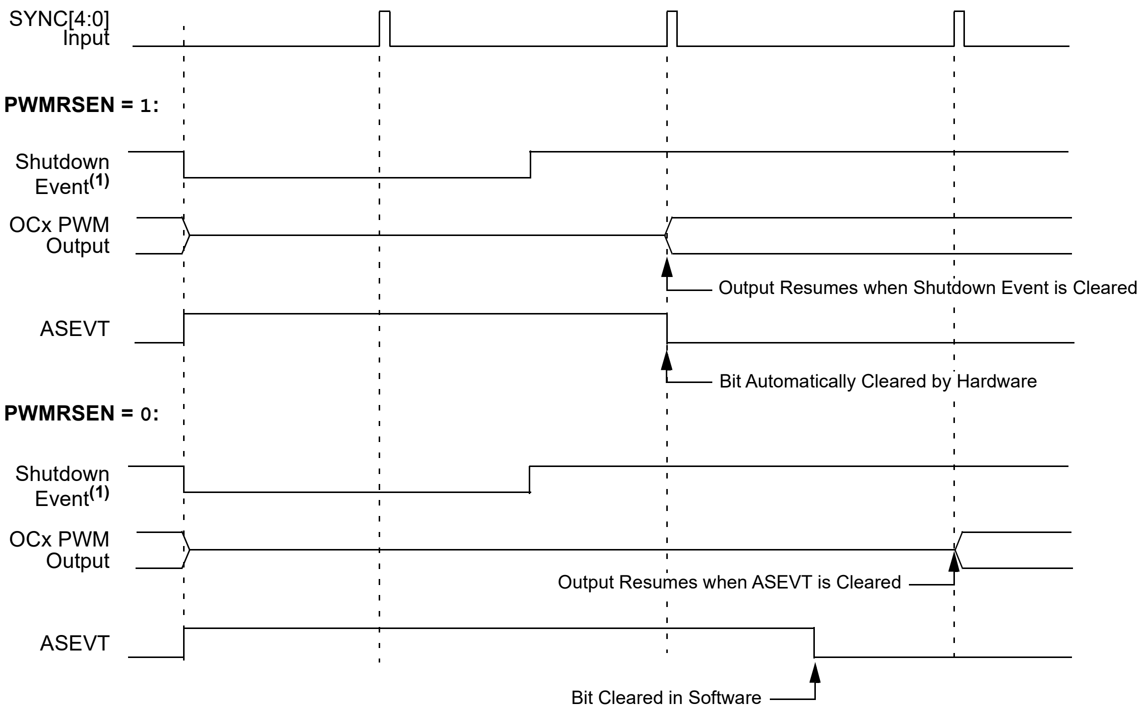
Automatic Restart Enable

The PWMRSEN bit (CCPxCON2[15]) controls how the Shutdown state is ended. If PWMRSEN = 0, the module will wait until the ASEVT status bit is cleared in user software. Normal output pin activity can only be resumed when the ASEVT bit is cleared AND the External Shutdown Source signal is no longer present. If the External Shutdown Source signal is still active, the user cannot clear the ASEVT bit.

If PWMRSEN = 1, normal output pin activity will automatically resume when the External Shutdown Source signal is inactive and the next PWM period begins (i.e., when the sync source selected by SYNC[4:0] is asserted). The ASEVT bit will automatically be cleared in hardware at this time. If the shutdown is still in effect at the time a new cycle begins, that entire cycle is suppressed, thus eliminating narrow, glitch pulses. The PWM outputs are then restarted on the next cycle.

If PWMRSEN = 0, once a Shutdown condition occurs, the PWM remains Idle until manually restarted by the user. The module can be restarted by clearing the ASEVT bit in software.

Figure 28-34. Automatic Restart Enable



Note:

1. Any device-defined hardware shutdown event when the corresponding ASDG bit is set or when the SSDG bit is set.

Gated Auto-Shutdown Mode

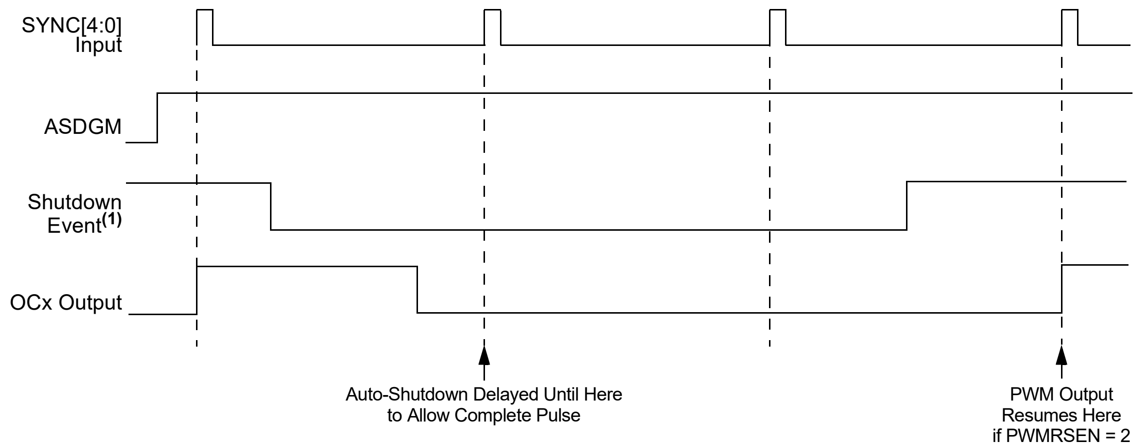
For some types of power control applications, it is useful to delay the effect of the auto-shutdown input. The ASDGM control bit (CCPxCON2[14]) enables gated shutdown operation. When ASDGM is set, the effect of an Auto-Shutdown signal input does not take place until the next PWM period boundary. This allows the PWM Generator to produce the entire pulse that was programmed for the present cycle.

Pulses are terminated by the hardware beginning on the next cycle. Pulses will resume on the next PWM period after the shutdown event has ended.

Note: Pulses only resume automatically if PWMRSEN = `1`. If PWMRSEN = `0`, the ASEVT status bit must be cleared in software for pulses to resume.

The Gated mode allows the Automatic Shutdown mode to be used along with a comparator to implement a ‘Pulse Skipping’ or ‘Gated Oscillator’ Switch mode power supply. The inductor is charged for a fixed period of time with each pulse, putting a specific amount of energy into the supply. If the output voltage (current) is high enough, then the comparator gates the pulses.

Figure 28-35. Gated Auto-Shutdown Mode



Notes:

1. Any device-defined hardware shutdown event when the corresponding ASDGx bit is set or when the SSDG bit is set.
2. (Not related to the Figure) If automatic restarts are not enabled, the application may also need to clear the ASEVT bit.
3. (Not related to the Figure) Any enabled shutdown source selected by the ASDGx bits and the SSDG software shutdown bit takes priority over a software write to the ASEVT bit. The Fault condition cannot be exited unless all shutdown sources are inactive.

28.4.4.4.6. Output Polarity Control

The polarity of OCxA is controlled by the POLA control bit (CCPxCON3[21]).

The output polarity control is applied to the Output signal after auto-shutdown logic. The polarity control bits are effective for all Output Compare and PWM modes of the module.

28.4.4.5. CCP External Input Source Mode

The External Input Source mode is selected when:

- CCM = `0`
- MOD[3:0] = `1111`
- T32 = `x`

The External Input Source mode is provided to bypass all Output Compare signal generation logic in the CCP module. This allows an External signal to be connected to the output control block to create Complementary signals, provide Auto-shutdown control and so on.

The module time base operates in 16-bit mode for the External Input mode and time-base interrupts are generated. If the time base is self-synchronized, the count period is set by the CCPxPR register. The CCPxRA and CCPxRB registers have no effect on the time-base operation for this mode.

The time base remains operational in External Input mode, so certain features of the output control logic can continue to operate. For example, these features must have a time-base period reference:

- The automatic restart feature of the auto-shutdown logic (PWMRSEN = 1) requires a Timer Synchronization signal to operate properly.
- The Auto-Shutdown Gated mode (ASDGM = 1) requires a Time-Base Synchronization signal.

If the time base is not used for synchronization purposes in the External Input mode, then it can be used as a general purpose 16-bit timer to provide periodic CPU interrupts. The External signal input source can be one of the signals connected to the input capture source and is selected using the ICS[2:0] (CCPxCON2[18:16]) control bits.

The external signal will be synchronized to the CCP time-base clock source and, therefore, must meet certain timing requirements. Specifically, the high and low times for the External signal must be no less than one timer clock period.

Note: If the Input Capture pin (ICx) is located on the same pin as one of the Output Compare pins of the same module, and if the ICx pin is enabled as the External Input signal source, to avoid unexpected results, the user should not enable the ICx pin and the OCx pin simultaneously. To avoid this condition, the user should select an alternate input using the ICS[2:0] control bits.

28.4.5. Module Sync Outputs

By default, the CCP module generates a CCP Sync signal from the rollover of the CCPxTMR register. This signal is made available to all of the other CCP modules as a sync source, or it is made available to trigger another peripheral. The CCP Sync signal is separate from the module's device-level interrupts or other outputs.

There may be circumstances where another event signal may serve as a better basis for the CCP Sync signal or where an additional event output, other than the selected signal, is required. The CCP modules include user configuration options to handle these situations.

28.4.5.1. Alternate Sync Out

The ALTSYNC control bit (CCPxCON1[21]) allows the user to substitute a different synchronization/trigger output in place of the timer rollover for the CCP Sync signal. When ALTSYNC = 0, the CCP Sync output is the default timer rollover signal in all operating modes. When ALTSYNC = 1, the Synchronization signal depends on the specific Operating mode. Table 28-10 lists the alternate outputs available.

Table 28-10. Alternate Sync Output Signals

ALTSYNC	CCSEL	MOD[3:0]	Output Signal
0	x	All	Standard (default) CCP Sync Output
1	0	0000	Special Event Trigger Output (Timer)
1	0	All except '0000'	Output Compare Interrupt Event (Compare)
1	1	All	Input Capture Event (Capture)

28.4.5.2. Auxiliary Output Signal

The CCP modules can also generate a secondary output that is different from the CCP Sync signal (or its alternate version if ALTSYNC is set). The auxiliary output is intended to allow other digital peripherals to access internal CCP module signals, such as:

- Time Base Synchronization
- Peripheral Trigger and Clock Inputs
- Signal Gating

The type of output signal is selected using the AUXOUT[1:0] control bits (CCPxCON2[20:19]) and is dependent on the module operating mode. More options are available for each mode than with the alternate Sync output, as shown in Table 28-11.

Table 28-11. Auxiliary Output Signals

AUXOUT[1:0]	CCSEL	MOD[3:0]	Output Signal
00	x	xxxx	Disabled (no output)
01	0	'0000' (Timer modes)	Time Base Period Reset or Rollover
10			Special Event Trigger Output
11			No Output
01	0	'0001' through '1111' (Output Compare modes)	Time Base Period Reset or Rollover
10			Output Compare Event Signal
11			Output Compare Signal
01	1	'xxxx' (Input Capture modes)	Time Base Period Reset or Rollover
10			Reflects the Value of the ICDIS Bit
11			Input Capture Event Signal

28.4.6. Sync and Triggered Operation

Synchronized (Sync) and Triggered mode operations can be thought of as Complementary modes that affect the operation of the CCPxTMR registers in most of the module's major operating modes. Both use the SYNC[4:0] bits (CCPxCON1[20:16]) to determine the input signal source. The difference is how that signal affects the timer.

In Sync mode operation, the timer counts freely when enabled by the CCPON bit and is reset to zero when the input, selected by SYNC[4:0], is asserted. The timer immediately begins to count again from zero unless it is held for some other reason. Sync operation is used whenever the TRIGEN bit (CCPxCON1[23]) is cleared.

In Triggered mode operation, the timer is held in Reset until the input selected by SYNC[4:0] is asserted; when this occurs, the timer starts counting and continues to count until the TRCLR bit (CCPxSTAT[5]) is set. Triggered operation is used whenever the TRIGEN bit is set.

Depending on the specific device, the SYNC[4:0] bits allow for the selection of up to 32 internal or external sources. Some implemented sources may be available for triggered operation but not for sync operation. In addition, '11111' (free-running counter) is not valid for sync operation.

Sync and trigger operations play a major role in the module's operation in Timer and Output Compare modes by allowing the chained and synchronized operation of multiple modules.

28.4.7. Register Access

28.4.7.1. Time-Base Access

The CCPxTMR register provides user access to the time-base value in all operating modes of the module. It is readable and writable. The CCPxTMR register is 16 or 32-bits wide, depending on the operating mode and the value of the T32 (CCPxCON1[5]), CCSEL (CCPxCON1[4]) and MOD[3:0] (CCPxCON1[3:0]) control bits.

28.4.7.2. Input Capture Data Buffer Access

The CCPxBUFL register bits' location provides user access to a four-level deep FIFO used in Input Capture modes. During normal operation, the CCPxBUFL register bits location is read until ICBNE (CCPxSTAT[0]) = 0. The ICOV (CCPxSTAT[1]) status bit tells the user if the FIFO buffer size has been exceeded, therefore losing the most recently captured data.

When operating in a 32-bit Input Capture mode, a two-level FIFO buffer is available. The CCPxBUF register provides access to the FIFO. The ICBNE and ICOV status bits operate similarly to 16-bit capture operations.

28.4.7.3. Compare Modes Data Write

Three data registers are provided for Output Compare modes:

- CCPxPR
- CCPxRA
- CCPxRB

CCPxPR provides a value for comparison to the module timer; the match is used to:

- Set the count period of the time base.
- Generate CCP Sync outputs for other modules.

CCPxRA provides a value for comparison to the module timer; the match is used:

- As a second compare register in Dual Compare mode to create all negative edges.
- As the upper 16 bits of the compare value in 32-bit single edge compare operations.

When the module operates with a 32-bit time base (T32 = `1`), CCPxRB and CCPxRA are combined to form a single 32-bit Compare register. The CCPxRB value is compared to the upper 16 bits of the time base, and the CCPxRA value is compared to the lower 16 bits. The CCPxRB/CCPxRA register pair performs these functions in 32-bit modes:

- Edge generation in Single Edge Output Compare modes
- Generates the ccp_trig_out signal.

A Time-Base Period register is not available in the 32-Bit Single Edge Output Compare mode. The CCP Sync output signal is generated only on a rollover from FFFFFFFF to 00000000.

28.4.7.4. CCPxPRL, CCPxRA and CCPxRB Write Status

In the operating modes, where the CCPxPRL, CCPxRA and CCPxRB are double-buffered, the PRLWIP (CCPxSTAT[20], RAWIP (CCPxSTAT[16]) and RBWIP (CCPxSTAT[17]) status bits indicate the buffer update status for these registers. The appropriate bit will set when one of these registers is written and will remain set until the buffered value takes effect, typically at the time-base period boundary.

Note: It is not necessary to monitor these three bits if software always updates the registers at a specific time in the time-base count period. This would occur, for example, when the time-base period interrupt is used to schedule an update in the software. If, however, software is expected to write these registers at a random time with respect to the time-base count period, then the appropriate status bit should be checked in software prior to performing the write.

Further writes to the CCPxPRL and/or CCPxRA/B registers should not be performed while the associated WIP status bit is set.

28.4.7.5. Data Register Manipulation with Asynchronous Clock Sources

Special consideration must be taken when certain data registers are manipulated using an asynchronous time-base clock source. The time-base clock source will be asynchronous to the system clock when TMRSYNC = `0` and CLKSEL[2:0] != `000`.

28.4.7.5.1. CCPxTMR Reads with Asynchronous Clock Source

When the time-base clock source is asynchronous to the system clock source, a read of CCPxTMR may produce unexpected results. This can be avoided by reading the CCPxTMR twice and comparing the read results, or the time-base value can be captured using the input capture logic.

Note: A second CCP peripheral may be synchronized to the primary CCP and used in Input Capture mode, if needed.

28.4.7.5.2. CCPxTMR Writes with Asynchronous Clock Source

When an asynchronous clock source is used, writes to CCPxTMR are buffered and the write progress is indicated using the TMRHWIP (CCPxSTAT[19]) and TMRLWIP (CCPxSTAT[18]) status bits. The write to the CCPxTMR register is complete when the appropriate WIP status bit is cleared by hardware. Another write can be safely performed after the WIP is cleared.

Note: These status bits are also affected when the clock is synchronous. However, it is not necessary to monitor these bits since the write will occur on the next system clock cycle.

The timer WIP bits are set as follows:

- In Dual 16-Bit Timer mode, TMRHWIP is set upon a write that includes the highest byte of CCPxTMRH. TMRLWIP is set upon a write that includes the highest byte of CCPxTMRL.
- In modes that use a single 16-bit timer, TMRLWIP is set upon a write that includes the highest byte of CCPxTMRL.
- In modes that use a 32-bit timer, TMRLWIP and TMRHWIP are set upon a write that includes the highest byte of CCPxTMRH.

User software should not perform any writes to CCPxTMR while the appropriate WIP bit is set; otherwise, an unexpected timer value may result.

Note: The TMRHWIP and TMRLWIP bits get set automatically when the CCPON bit is set. This ensures any modifications to the CCPxTMR register while the timer module is off will take effect. The user may monitor the WIP bits after the CCPON bit is set to determine when the CCPxTMR contents have been updated and the time base has begun counting.

28.5. Power-Saving Modes

28.5.1. Idle Mode

The behavior of the module in Idle mode is determined by the CCPSIDL bit (CCPxCON1[13]). If CCPSIDL is cleared, the module will continue to operate in Idle mode. If CCPSIDL is set, the module is disabled when the device enters Idle mode. If the module is performing an operation when Idle mode is invoked, in this case, the results will be similar to those with Sleep mode.

28.5.2. Sleep Modes

The behavior of the module in Sleep mode is determined by the CCPSLP bit (CCPxCON1[12]). If CCPSLP is set, the module will continue to operate during Sleep mode, assuming that the selected clock source remains available. The TMRSYNC bit must remain cleared for the module to operate in Sleep mode.

When CCPSLP is cleared and the device enters Sleep mode, the module is disabled. However, if CCPSLP is set and the module is configured for 16-Bit Edge Detect Input Capture mode (MOD[3:0] = `0000`, CCSEL = `1`), the module can generate an interrupt and wake up the device as long as the clock source remains active. In this configuration, the Input Capture pin can function like an external interrupt. The corresponding CCP interrupt must also be enabled (CCPxIE = `1`).

28.5.2.1. Triggered Operation and Sleep Mode

When the module is configured for triggered operation, a Trigger signal received from an external source can also wake the module and its time-base clock source. The module must request the time-base clock source before a triggered operation can begin. When the trigger is received from the external source, the CCP module will enable the selected clock source for the time base. When the clock source becomes available, the module will begin a triggered operation.

If One-Shot Triggered mode is selected, the time-base clock source will be disabled when the CCPTRIG status bit is cleared in hardware. The time base remains disabled until a new trigger signal is received. The trigger signal can also be generated by an internal source that operates from a low-power clock source. When Sleep mode operation is enabled, the module will continue to request the configured clock source when the device enters Sleep mode.

28.5.3. Interrupts

The CCP module has the ability to generate the interrupt on the period match input capture event or output compare event. The CCP module is enabled as a source of interrupt via the CCP Interrupt Enable bits, CCTxIE and CCPxIE. The CCP Interrupt Priority Level is defined by the CCP Interrupt Priority bits CCTxIP[2:0] and CCPxIP[2:0].

CCTxIF is set when the TMRx count matches the respective PRx register. CCPxIF is set whenever there is an input capture event or an output compare equal event. The CCTxIF and CCPxIF bits must be cleared in software.

Note: A special case occurs when the PRx register is loaded with '0' and the timer is enabled. An interrupt is not generated for this configuration.

28.5.3.1. Interrupt Configuration

The CCPx module has a dedicated Interrupt flag bit (CCT/PxIF) and a corresponding Interrupt Enable/Mask bit (CCT/PxIE).

The CCT/PxIE bit is used to define the behavior of the interrupt controller when the CCT/PxIF bit is set. When the CCT/PxIE bit is clear, the interrupt controller does not generate a CPU interrupt for the event. If the CCT/PxIE bit is set, the interrupt controller generates an interrupt to the CPU when the CCT/PxIF bit is set (subject to the interrupt priority).

The software routine that services a particular interrupt must clear the appropriate Interrupt Flag bit before the service routine is complete. The priority of the CCPx module can be set with the CCT/PxIP[2:0] bits. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of seven (the highest priority) to a value of zero (which does not generate an interrupt). An interrupt being serviced will be preempted by an interrupt in a higher priority group.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. Code at this vector address should perform any application-specific operations and clear the CCT/Px1IF interrupt flag and then exit.

28.6. Effects of a Reset

A device Reset forces all registers to their Reset state; this disables the module and returns it to its default configuration (16-bit timer). All buffer and address registers are initialized to 0000h and all status flags are reset.

By default, the pin associated with OCxA resets with the Output Compare function in control of the pin; however, this has no effect when the module is disabled.

29. Configurable Logic Cell (CLC)

The Configurable Logic Cell (CLC) module allows the user to specify combinations of signals as inputs to a logic function and to use the logic output to control other peripherals or I/O pins. This provides greater flexibility and potential in embedded designs since the CLC module can operate outside the limitations of software execution and can support a vast number of output designs. The high-level features include:

- General Purpose Logic Block
- 32 Input Sources, Including External Pins and Other Device Peripheral Outputs
- Combinational Logic Modes
- Sequential Logic Modes
- Edge Detection and Interrupt Generation When Processor is in Sleep
- Delay Generation
- Inversion Logic
- Signal Routing Between Otherwise Unconnected Peripherals

29.1. Device-Specific Information

Table 29-1. CLC Summary Table

CLC Module Instances	Inputs per Instance	CLC Outputs	Peripheral Bus Speed
4	8	4	Slow (1:4 CPU Clock)

Table 29-2. DS1 Data Selection MUX 1 Signal Selection bits

Value (binary)	Description
111	Virtual Pin 5 Output
110	Virtual Pin 4 Output
101	Virtual Pin 3 Output
100	CLCINB
011	CLCINA
010	CLC1 Output
001	CLKGEN14
000	Standard (1:2 CPU Clock)

Table 29-3. DS2 Data Selection MUX 2 Signal Selection bits

Value (binary)	Description
111	Virtual Pin 8 Output
110	Virtual Pin 7 Output
101	Virtual Pin 6 Output
100	CLCINE
011	CLCIND
010	CLCINC
001	CLC2 Output
000	Slow Peripheral Clock (system clock/4)

Table 29-4. DS3 Data Selection MUX 3 Signal Selection bits

Value (binary)	Description
111	Virtual Pin 12 Output
110	Virtual Pin 11 Output

Table 29-4. DS3 Data Selection MUX 3 Signal Selection bits (continued)

Value (binary)	Description
101	Virtual Pin 10 Output
100	Virtual Pin 9 Output
011	CLCING
010	CLCINF
001	CLC3 Output
000	BFRC/244[(1)]

Note:
1. Set the OSCCTRL.LPRCEN bit to '1' to enable the LPRC clock and observe it on the CLCxOUT pin.

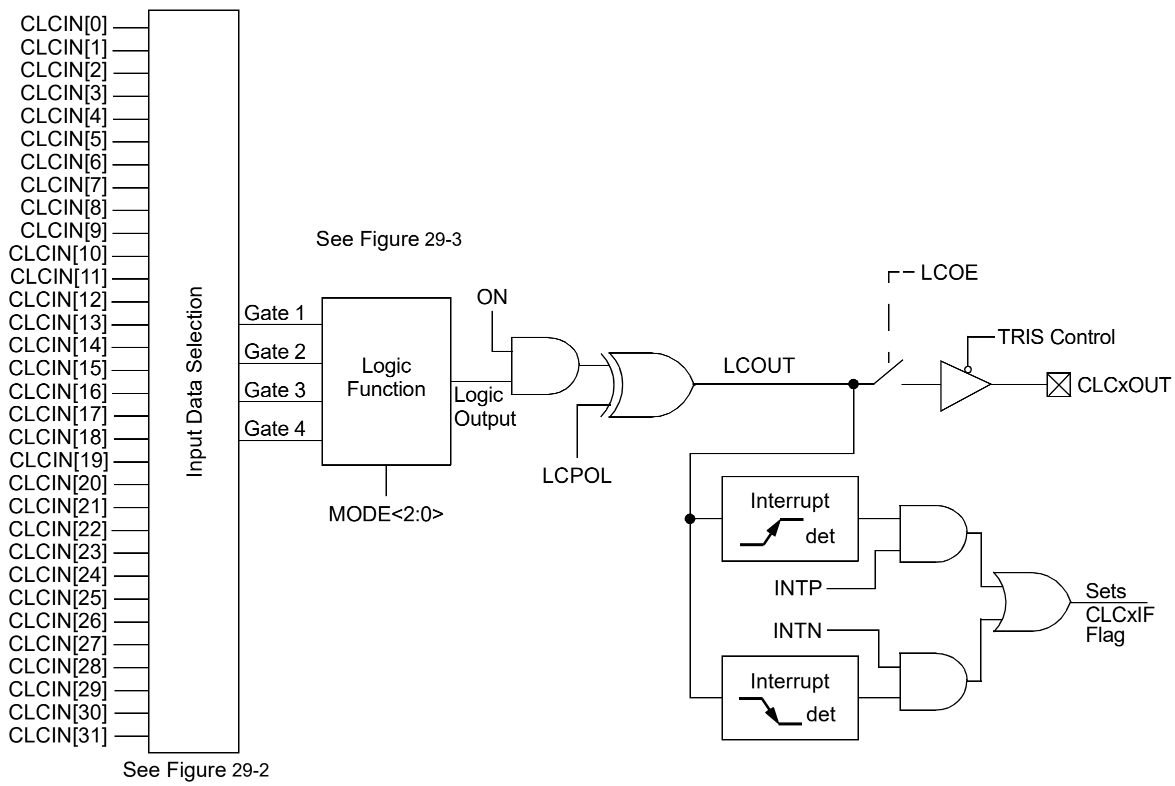
Table 29-5. DS4 Data Selection MUX 4 Signal Selection bits

Value (binary)	Description
111	Virtual Pin 15 Output
110	Virtual Pin 14 Output
101	Virtual Pin 13 Output
100	CLCINJ
011	CLCINI
010	CLCINH
001	CLC4 Output
000	FRC

29.2. Architecture

The CLC consists of four main sections, as shown in Figure 29-1. First, the input data selection MUXes route input signals to the four data gates, as shown in Figure 29-2. Each of the four data gates can then select any of the 32 input signals to pass along to the logic functions shown in Figure 29-3. The output of the logic function is then supplied to the internal logic and external pin and can generate interrupts. The output of a CLC module can be routed to the input of another CLC module to create more complex logic functions.

Figure 29-1. Configurable Logic Cell



Note: All control bits shown in this figure can be found in the CLCxCON register.

Figure 29-2. CLC Input Source Selection Diagram

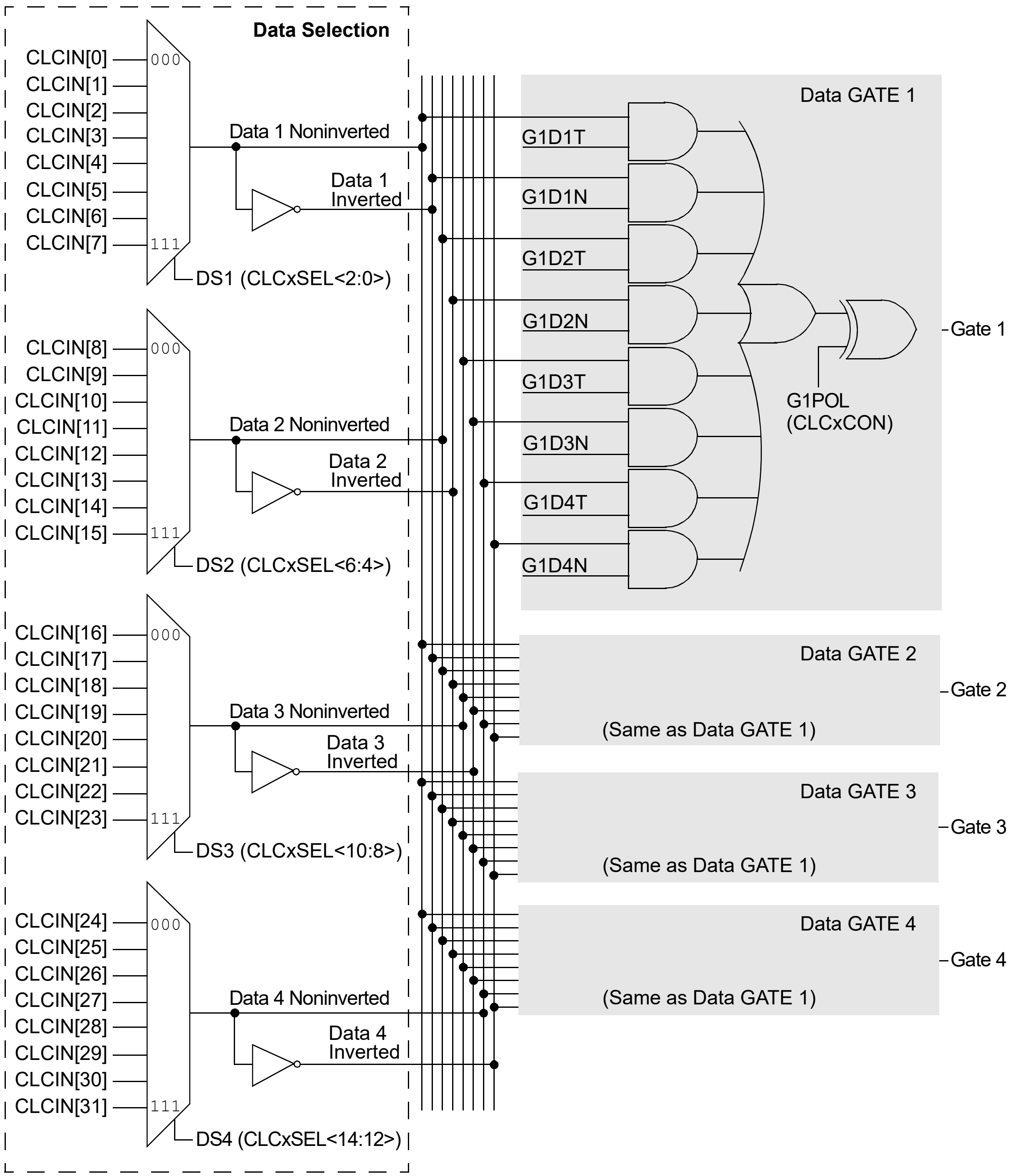
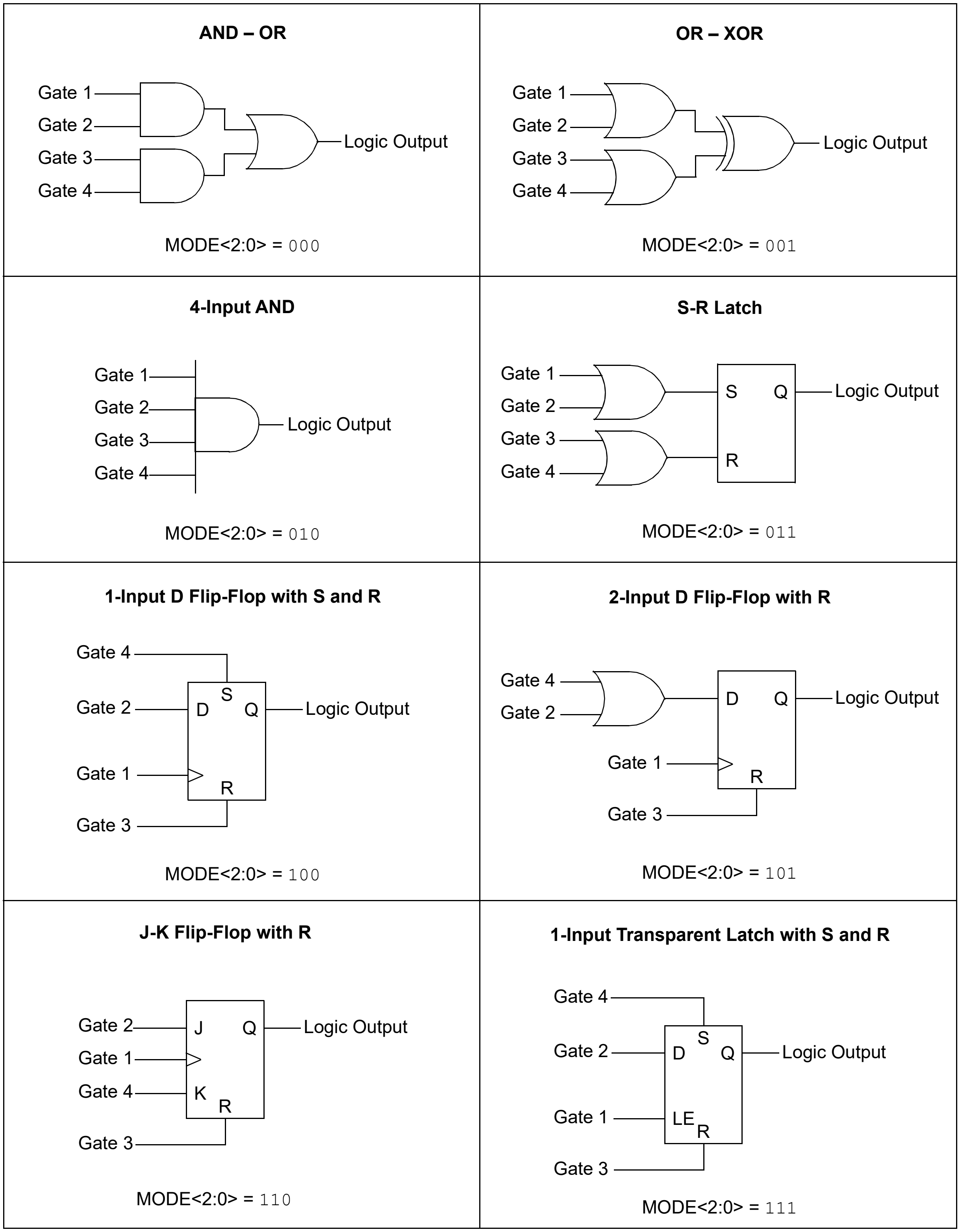


Figure 29-3. Logic Function Combinatorial Options



29.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3A60	CLC1CON	31:24								
		23:16					G4POL	G3POL	G2POL	G1POL
		15:8	ON				INTP	INTN		
		7:0	LCOE	LCOUT	LCPOL			MODE[2:0]		
0x3A64	CLC1SEL	31:24								
		23:16								
		15:8		DS4[2:0]				DS3[2:0]		
		7:0		DS2[2:0]				DS1[2:0]		
0x3A68	CLC1GLS	31:24	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
		23:16	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
		15:8	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
		7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x3A6C ... 0x3A6F	Reserved									
0x3A70	CLC2CON	31:24								
		23:16					G4POL	G3POL	G2POL	G1POL
		15:8	ON				INTP	INTN		
		7:0	LCOE	LCOUT	LCPOL			MODE[2:0]		
0x3A74	CLC2SEL	31:24								
		23:16								
		15:8		DS4[2:0]				DS3[2:0]		
		7:0		DS2[2:0]				DS1[2:0]		
0x3A78	CLC2GLS	31:24	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
		23:16	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
		15:8	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
		7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x3A7C ... 0x3A7F	Reserved									
0x3A80	CLC3CON	31:24								
		23:16					G4POL	G3POL	G2POL	G1POL
		15:8	ON				INTP	INTN		
		7:0	LCOE	LCOUT	LCPOL			MODE[2:0]		
0x3A84	CLC3SEL	31:24								
		23:16								
		15:8		DS4[2:0]				DS3[2:0]		
		7:0		DS2[2:0]				DS1[2:0]		
0x3A88	CLC3GLS	31:24	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
		23:16	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
		15:8	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
		7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x3A8C ... 0x3A8F	Reserved									
0x3A90	CLC4CON	31:24								
		23:16					G4POL	G3POL	G2POL	G1POL
		15:8	ON				INTP	INTN		
		7:0	LCOE	LCOUT	LCPOL			MODE[2:0]		
0x3A94	CLC4SEL	31:24								
		23:16								
		15:8		DS4[2:0]				DS3[2:0]		
		7:0		DS2[2:0]				DS1[2:0]		
0x3A98	CLC4GLS	31:24	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
		23:16	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
		15:8	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
		7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N

29.3.1. Configurable Logic Cell x Control Register

Name: CLCxCON
Offset: 0x3A60, 0x3A70, 0x3A80, 0x3A90

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
					G4POL	G3POL	G2POL	G1POL
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON				INTP	INTN		
Access	R/W				R/W	R/W		
Reset	0				0	0		

Bit	7	6	5	4	3	2	1	0
	LCOE	LCOUT	LCPOL			MODE[2:0]		
Access	R/W	R	R			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bit 19 – G4POL Gate 4 Polarity Control bit

Value	Description
1	The output of Gate 4 logic is inverted when applied to the logic cell.
0	The output of Gate 4 logic is not inverted.

Bit 18 – G3POL Gate 3 Polarity Control bit

Value	Description
1	The output of Gate 3 logic is inverted when applied to the logic cell.
0	The output of Gate 3 logic is not inverted.

Bit 17 – G2POL Gate 2 Polarity Control bit

Value	Description
1	The output of Gate 2 logic is inverted when applied to the logic cell.
0	The output of Gate 2 logic is not inverted.

Bit 16 – G1POL Gate 1 Polarity Control bit

Value	Description
1	The output of Gate 1 logic is inverted when applied to the logic cell.
0	The output of Gate 1 logic is not inverted.

Bit 15 – ON Configurable Logic Cell Enable bit

Value	Description
1	Configurable Logic Cell is enabled and mixes input signals.
0	Configurable Logic Cell is disabled and has logic zero outputs.

Bit 11 – INTP Configurable Logic Cell Positive Edge Interrupt Enable bit

Value	Description
1	Interrupt will be generated when a rising edge occurs on LCOUT.
0	Interrupt will not be generated.

Bit 10 – INTN Configurable Logic Cell Negative Edge Interrupt Enable bit

Value	Description
1	Interrupt will be generated when a falling edge occurs on LCOUT.
0	Interrupt will not be generated.

Bit 7 – LCOE Configurable Logic Cell Port Enable bit

Value	Description
1	Configurable Logic Cell port pin output is enabled.
0	Configurable Logic Cell port pin output is disabled.

Bit 6 – LCOUT Configurable Logic Cell Data Output Status bit

Value	Description
1	Configurable Logic Cell output high
0	Configurable Logic Cell output low

Bit 5 – LCPOL Configurable Logic Cell Output Polarity Control bit

Value	Description
1	The output of the module is inverted.
0	The output of the module is not inverted.

Bits 2:0 – MODE[2:0] Configurable Logic Cell Mode bits

Value	Description
111	Cell is one-input transparent latch with S and R.
110	Cell is J-K flip-flop with Reset.
101	Cell is two-input D flip-flop with R.
100	Cell is one-input D flip-flop with S and R.
011	Cell is SR latch.
010	Cell is four-input AND.
001	Cell is OR-XOR.
000	Cell is AND-OR.

29.3.2. Configurable Logic Cell x Input MUX Select Register

Name: CLCxSEL
Offset: 0x3A64, 0x3A74, 0x3A84, 0x3A94

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
			DS4[2:0]				DS3[2:0]	
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
			DS2[2:0]				DS1[2:0]	
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bits 14:12 – DS4[2:0] Data Selection MUX 4 Signal Selection bits
See Table 29-5.

Bits 10:8 – DS3[2:0] Data Selection MUX 3 Signal Selection bits
See Table 29-4.

Bits 6:4 – DS2[2:0] Data Selection MUX 2 Signal Selection bits
See Table 29-3.

Bits 2:0 – DS1[2:0] Data Selection MUX 1 Signal Selection bits
See Table 29-2.

29.3.3. Configurable Logic Cell x Source Enable Register

Name: CLCxGLS
Offset: 0x3A68, 0x3A78, 0x3A88, 0x3A98

Bit	31	30	29	28	27	26	25	24
	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 31 – G4D4T Gate 4 Data 4 True Enable bit

Value	Description
1	The Data 4 (noninverted) signal is enabled for Gate 4.
0	The Data 4 (noninverted) signal is disabled for Gate 4.

Bit 30 – G4D4N Gate 4 Data 4 Negated Enable bit

Value	Description
1	The Data 4 (inverted) signal is enabled for Gate 4.
0	The Data 4 (inverted) signal is disabled for Gate 4.

Bit 29 – G4D3T Gate 4 Data 3 True Enable bit

Value	Description
1	The Data 3 (noninverted) signal is enabled for Gate 4.
0	The Data 3 (noninverted) signal is disabled for Gate 4.

Bit 28 – G4D3N Gate 4 Data 3 Negated Enable bit

Value	Description
1	The Data 3 (inverted) signal is enabled for Gate 4.
0	The Data 3 (inverted) signal is disabled for Gate 4.

Bit 27 – G4D2T Gate 4 Data 2 True Enable bit

Value	Description
1	The Data 2 (noninverted) signal is enabled for Gate 4.
0	The Data 2 (noninverted) signal is disabled for Gate 4.

Bit 26 – G4D2N Gate 4 Data 2 Negated Enable bit

Value	Description
1	The Data 2 (inverted) signal is enabled for Gate 4.
0	The Data 2 (inverted) signal is disabled for Gate 4.

Bit 25 – G4D1T Gate 4 Data 1 True Enable bit

Value	Description
1	The Data 1 (noninverted) signal is enabled for Gate 4.
0	The Data 1 (noninverted) signal is disabled for Gate 4.

Bit 24 – G4D1N Gate 4 Data 1 Negated Enable bit

Value	Description
1	The Data 1 (inverted) signal is enabled for Gate 4.
0	The Data 1 (inverted) signal is disabled for Gate 4.

Bit 23 – G3D4T Gate 3 Data 4 True Enable bit

Value	Description
1	The Data 4 (noninverted) signal is enabled for Gate 3.
0	The Data 4 (noninverted) signal is disabled for Gate 3.

Bit 22 – G3D4N Gate 3 Data 4 Negated Enable bit

Value	Description
1	The Data 4 (inverted) signal is enabled for Gate 3.
0	The Data 4 (inverted) signal is disabled for Gate 3.

Bit 21 – G3D3T Gate 3 Data 3 True Enable bit

Value	Description
1	The Data 3 (noninverted) signal is enabled for Gate 3.
0	The Data 3 (noninverted) signal is disabled for Gate 3.

Bit 20 – G3D3N Gate 3 Data 3 Negated Enable bit

Value	Description
1	The Data 3 (inverted) signal is enabled for Gate 3.
0	The Data 3 (inverted) signal is disabled for Gate 3.

Bit 19 – G3D2T Gate 3 Data 2 True Enable bit

Value	Description
1	The Data 2 (noninverted) signal is enabled for Gate 3.
0	The Data 2 (noninverted) signal is disabled for Gate 3.

Bit 18 – G3D2N Gate 3 Data 2 Negated Enable bit

Value	Description
1	The Data 2 (inverted) signal is enabled for Gate 3.
0	The Data 2 (inverted) signal is disabled for Gate 3.

Bit 17 – G3D1T Gate 3 Data 1 True Enable bit

Value	Description
1	The Data 1 (noninverted) signal is enabled for Gate 3.
0	The Data 1 (noninverted) signal is disabled for Gate 3.

Bit 16 – G3D1N Gate 3 Data 1 Negated Enable bit

Value	Description
1	The Data 1 (inverted) signal is enabled for Gate 3.
0	The Data 1 (inverted) signal is disabled for Gate 3.

Bit 15 – G2D4T Gate 2 Data 4 True Enable bit

Value	Description
1	The Data 4 (noninverted) signal is enabled for Gate 2.
0	The Data 4 (noninverted) signal is disabled for Gate 2.

Bit 14 – G2D4N Gate 2 Data 4 Negated Enable bit

Value	Description
1	The Data 4 (inverted) signal is enabled for Gate 2.
0	The Data 4 (inverted) signal is disabled for Gate 2.

Bit 13 – G2D3T Gate 2 Data 3 True Enable bit

Value	Description
1	The Data 3 (noninverted) signal is enabled for Gate 2.
0	The Data 3 (noninverted) signal is disabled for Gate 2.

Bit 12 – G2D3N Gate 2 Data 3 Negated Enable bit

Value	Description
1	The Data 3 (inverted) signal is enabled for Gate 2.
0	The Data 3 (inverted) signal is disabled for Gate 2.

Bit 11 – G2D2T Gate 2 Data 2 True Enable bit

Value	Description
1	The Data 2 (noninverted) signal is enabled for Gate 2.
0	The Data 2 (noninverted) signal is disabled for Gate 2.

Bit 10 – G2D2N Gate 2 Data 2 Negated Enable bit

Value	Description
1	The Data 2 (inverted) signal is enabled for Gate 2.
0	The Data 2 (inverted) signal is disabled for Gate 2.

Bit 9 – G2D1T Gate 2 Data 1 True Enable bit

Value	Description
1	The Data 1 (noninverted) signal is enabled for Gate 2.
0	The Data 1 (noninverted) signal is disabled for Gate 2.

Bit 8 – G2D1N Gate 2 Data 1 Negated Enable bit

Value	Description
1	The Data 1 (inverted) signal is enabled for Gate 2.
0	The Data 1 (inverted) signal is disabled for Gate 2.

Bit 7 – G1D4T Gate 1 Data 4 True Enable bit

Value	Description
1	The Data 4 (noninverted) signal is enabled for Gate 1.
0	The Data 4 (noninverted) signal is disabled for Gate 1.

Bit 6 – G1D4N Gate 1 Data 4 Negated Enable bit

Value	Description
1	The Data 4 (inverted) signal is enabled for Gate 1.
0	The Data 4 (inverted) signal is disabled for Gate 1.

Bit 5 – G1D3T Gate 1 Data 3 True Enable bit

Value	Description
1	The Data 3 (noninverted) signal is enabled for Gate 1.
0	The Data 3 (noninverted) signal is disabled for Gate 1.

Bit 4 – G1D3N Gate 1 Data 3 Negated Enable bit

Value	Description
1	The Data 3 (inverted) signal is enabled for Gate 1.
0	The Data 3 (inverted) signal is disabled for Gate 1.

Bit 3 – G1D2T Gate 1 Data 2 True Enable bit

Value	Description
1	The Data 2 (noninverted) signal is enabled for Gate 1.
0	The Data 2 (noninverted) signal is disabled for Gate 1.

Bit 2 – G1D2N Gate 1 Data 2 Negated Enable bit

Value	Description
1	The Data 2 (inverted) signal is enabled for Gate 1.
0	The Data 2 (inverted) signal is disabled for Gate 1.

Bit 1 – G1D1T Gate 1 Data 1 True Enable bit

Value	Description
1	The Data 1 (noninverted) signal is enabled for Gate 1.
0	The Data 1 (noninverted) signal is disabled for Gate 1.

Bit 0 – G1D1N Gate 1 Data 1 Negated Enable bit

Value	Description
1	The Data 1 (inverted) signal is enabled for Gate 1.
0	The Data 1 (inverted) signal is disabled for Gate 1.

29.4. Operation

The CLCx Control register (CLCxCON) is used to enable the module and interrupts, control the output enable bit, select output polarity and select the logic function. The CLCx Control register also allows the user to control the logic polarity of not only the cell output but also some intermediate variables.

The CLCx Input MUX Select register (CLCxSEL) allows the user to select one out of eight input signals for each of the four data selection multiplexers pictured inside the dotted line in Figure 29-2. The output of each of the four data selection multiplexers is connected to the inputs of the logic function selected by the MODE[2:0] bits (CLCxCON[2:0]). See Figure 29-3.

The CLCx Source Enable register (CLCxGLS) allows the user to create any four variable boolean expressions from the four input data sources configured by CLCxSEL. Both the true and complementary values for each of the four signals, chosen by the CLCx Input MUX Select register (CLCxSEL), are available to the sum-of-products circuit pictured in the data gate in Figure 29-2.

29.4.1. Reset

When the ON bit is written to '0', the output of all state logic functions will be reset to '0'. A system Reset returns the CLCxCON, CLCxSEL and CLCxGLS registers to the Default state and disables the module.

Asserting a device Reset returns all bits in the module registers to the Default state. The output of all logic functions is '0' after a Reset; this includes both latch and flip-flop functions. When a device Reset is asserted, ON (CLCxCON[15]) = 0, the state logic is reset and the output of the logic function is forced low.

29.4.2. CLC Setup

CLCxCON selects the logic function and determines and controls the I/O pins. This register also controls output signal polarity. The ON bit (CLCxCON[15]) must be set for the CLC to operate. All registers can be programmed while ON is clear.

The CLCxSEL (CLCxSEL) register controls which input signals are routed to the input bus of Figure 29-2. Both the True (T) and Negated (N) values are made available on the data bus.

The CLCxGLS (CLCxGLS) register selects which signals from the data bus are applied to the input OR gates. True and negated inputs are separately enabled; do not enable both for the same signal.

The final polarity of the CLC module output is controlled by the LCPOL bit (CLCxCON[5]). The output is inverted when LCPOL = 1 and uninverted when LCPOL = 0. The GxPOL bits (CLCxCON[19:16]) control the polarity of the logic function inputs.

The INTP and INTN bits (CLCxCON[11:10]) enable interrupts on the rising and falling edges of the CLC output.

The LCOUT bit is read-only and reflects the status of the logic cell output. To output the CLCxOUT signal to an I/O pin, set the LCOE bit and configure the I/O as a digital output. The CLCxOUT signal is made available through Peripheral Pin Select (PPS) and will need to be configured.

29.4.3. Input Providers

Each logic cell in the CLC takes four inputs, one from each of the four data gates. Each data gate is connected to eight input sources. The data gate allows the selection between the inverted or non-inverted polarity of each input source. Refer to Device-Specific Information for input sources available for use with the CLC.

29.4.3.1. Source Multiplexers

The module has four input source multiplexers. Multiplexer inputs are selected by setting control bits in the CLCxSEL register to define the data source selected through each of the four data selection multiplexers. Each of the four data selection multiplexers feeds one of the four logic function input gates, as shown in Figure 29-3. The module has an internal data bus created from the output of each input source multiplexer (see Figure 29-2). The data bus has both True (T) and Negated (N) versions of each selected input source. Therefore, up to eight signals are available on the internal data bus to connect to the input gates of the logic function.

29.4.3.2. Logic Input Gates

Four logic input gates are used to route input sources from the data selection multiplexers into the four logic function inputs. The true and negated forms of each input source signal are available for use by each logic gate. The input signal sources are enabled for use by each logic function input using the CLCxGLS register. There are up to eight signals that can be enabled for use by each logic function input. Any number of the eight signal sources may be enabled for each of the four logic function inputs. Each logic gate provides a logical OR of the input signals. The selected (True or Negated) signals are OR'd to form the gate output data. The logical NAND is obtained by changing the output polarity with the GxPOL bits. If the logical AND is required instead, select negated inputs and invert the output polarity according to De Morgan's Theorem. If all inputs are negated and applied to a NOR, the result is identical to an AND operation. Written algebraically:

C = A AND B

is the same as:

C = NOT(NOT(A) OR NOT(B)).

Table 29-6 summarizes the basic functions that can be obtained by using the Gate Control bits. The table shows the use of all four input multiplexer sources, but the input gates can be configured to use less. If no inputs are selected (CLCxGLS = 0x00), the output will be zero or one, depending on the GxPOL bits.

Table 29-6. Example Logic Functions

CLCxGLS	GxPOL Bits	Function
0xAAAAAAAA	0	OR (D1, D2, D3, D4)
0xAAAAAAAA	1	NOR (D1, D2, D3, D4)
0x55555555	0	NAND (D1, D2, D3, D4)
0x55555555	1	AND (D1, D2, D3, D4)
0x22222222	1	NOT (D1)
0x00000000	0	Logic '0'
0x00000000	1	Logic '1'

If the output of a gate must be zero or one, the recommended method is to set all of the bits related to that gate in CLCxGLS to zero and use the Gate Polarity bit, GxPOL, to set the desired level.

29.4.3.3. Logic Function

There are eight available logic functions, including:

- AND-OR
- OR-XOR
- AND
- S-R Latch
- D Flip-Flop with Set and Reset
- D Flip-Flop with Reset
- J-K Flip-Flop with Reset
- Transparent Latch with Set and Reset

Logic functions are shown in Figure 29-3. Each logic function has four inputs and one output. The MODE[2:0] bits (CLCxCON[2:0]) set the functional behavior of the logic cell. There are four combinatorial options and four state options. Three of the state options define Input Gate 1 as a rising edge clock with the traditional meanings of D and J-K flip-flops. The fourth state option, MODE[2:0] bits = `111`, is a transparent latch; Q follows D when Latch Enable (LE) is true; Q holds state when LE is false. For options with both S (Set) and R (Reset) inputs, the output changes asynchronously to the clock when S or R is a logic '1'; R is dominant.

29.4.3.4. Software Inputs

The gate data input to the logic function can be directly controlled by software by setting all of the CLCxGLS bits associated with the logic gate to '0' and writing to the appropriate GxPOL bit (see Table 29-6). The gate output will be equal to the value of the GxPOL bit.

29.4.4. Output

LCOUT (CLCxCON[6]) is the logic cell output and is routed to the I/O port pin or to other modules within the device. In all cases, the signal value is taken after the LCPOL inverter. To observe this output on an I/O pin, the user will need to set LCOE (CLCxCON[7]).

29.4.5. Application Logic

The CLC provides both combinatorial and state (see Figure 29-3) logic function options. The outputs of the input gates are applied to the logic function. If CLCxGLS = 0x00, the function receives a logic '0' when the GxPOL bits (CLCxCON[19:16]) are clear or a logic '1' when the GxPOL bits are set.

29.4.5.1. Combinatorial Logic

The combinatorial functions (MODE[2:0] = 010, 001, 000) build on the AND/OR logic of the input gate. The four-input AND can provide an OR function by inverting the inputs and outputs using De Morgan's Theorem. Inverting the output of the XOR is the same as inverting one input (but not both).

The SR function (MODE[2:0] = 011) is not affected when ON (CLCxCON[15]) is cleared, as is the case with the State Logic register. The latch is Reset-dominant, meaning that the Reset signal takes precedent over any Set signal that may be present.

29.4.5.2. State Logic

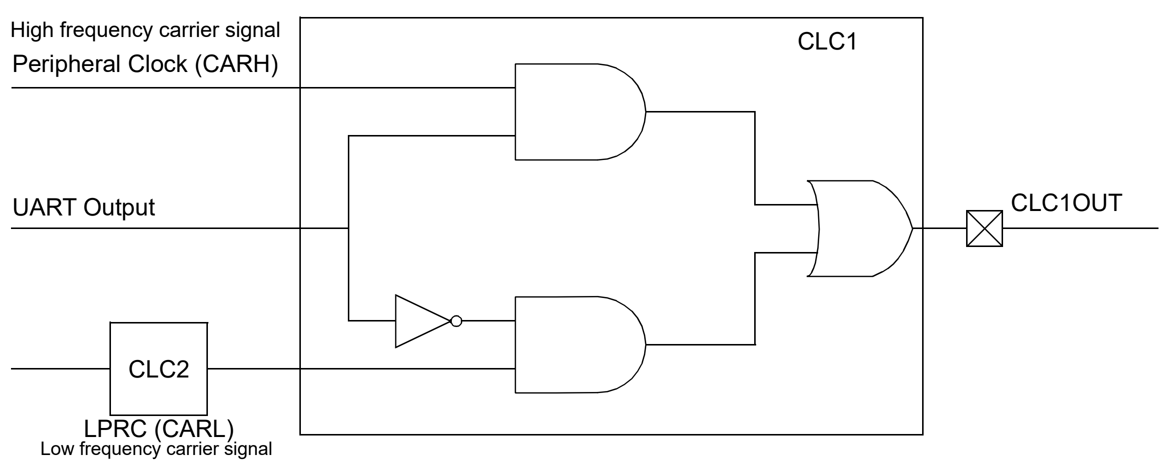
The state functions include both D and J-K flip-flops with asynchronous Set (S) and Reset (R). Input Gate 1 provides a rising edge clock. If a falling edge clock is required, Gate 1 can be inverted in the gate logic (G1POL). Input Gate 2, and sometimes also Gate 4, provide data to the register or latch input(s). When operating in Transparent Latch mode (MODE[2:0] = 111), the output, Q, follows D while LE is high and holds state while LE is low.

The various modes may or may not share state memory and switching modes may or may not change the state of the state variable. For all modes, the register is Reset-dominant.

29.5. CLC Application Example

Figure 29-4 depicts the configuration of CLC to generate Frequency Shift Key (FSK) modulation on a UART signal using CLC.

Figure 29-4. CLC Configuration for FSK Generation



Note: Inverter operation shown in Figure 29-4 is done using the CLC register bits.

The peripherals required for this application are:

- CLC1 and CLC2
- UART data as modulator signal
- Peripheral clock (carrier signal of higher frequency) to modulate logic '1'
- LPRC (carrier signal of lower frequency) to modulate logic '0'

Note: UART, system clock and LPRC are used as examples for the modulator signal and carrier signal, respectively. However, it is possible to choose other sources as modulator and carrier signals.

The following is the application code for FSK modulation of the UART data.

Example 29-1. FSK Modulation of UART Data

```
/*Select input source for CLC1*/
CLC1SELbits.DS1 = 1;   //Peripheral clock as input source to modulate high
signal
CLC1SELbits.DS2 = 3;   //UART TX data as input source
CLC1SELbits.DS4 = 1;   //CLC2 output as input source

 //Configure Gates of CLC1
CLC1GLSbits.G2D2T = 1; // Gate 2 selects Data 2 (here UART TX data)
CLC1GLSbits.G1D1T = 1; // Gate 1 selects Data 1(here peripheral clock)
CLC1GLSbits.G3D2N = 1; // Gate 3 selects inverted Data 2 (here UART TX data)
CLC1GLSbits.G4D4T = 1; // Gate 4 selects Data 4 (here CLC2 output)

/*Select input source for CLC2*/
CLC2SELbits.DS1 = 3;   //LPRC as input source to modulate logic low signal

 //Configure Gates of CLC2
CLC2GLSbits.G1D1T = 1; // All gates select data 1 as input (here LPRC)
CLC2GLSbits.G2D1T = 1;
CLC2GLSbits.G3D1T = 1;
CLC2GLSbits.G4D1T = 1;

 //Configure CLC1 mode and output
CLC1CONbits.LCOE = 1;  // Enable CLC output on IO
CLC1CONbits.MODE = 0;  // Select AND-OR logic function

 //Configure CLC2 mode and output
CLC2CONbits.LCOE = 1;  // Enable CLC output on IO
CLC2CONbits.MODE = 2;  // Select AND logic function


/* UART configuration goes here*/

 //Turn on CLCs
CLC1CONbits.ON = 1;  // Enable CLC1
CLC2CONbits.ON = 1;  // Enable CLC2

/*Turn on UART here*/
```

29.6. CLC Interrupts

The CLC module has two types of interrupts that can be enabled: rising edge interrupt events and falling edge interrupt events. These events are enabled by the INTP (CLCxCON[11]) and INTN (CLCxCON[10]) control bits, respectively.

A valid occurrence of either interrupt will set the CLCx Interrupt Flag, CLCxIF. This will occur when the module is enabled (ON = 1) and either a rising edge output occurs when INTP = 1 or a falling edge event occurs when INTN = 1.

If the Initial Output state of the CLC logic is '1' and INTP = 1, an interrupt will be generated when ON is set to '1'. Likewise, an interrupt will be generated if the Initial Output state of the CLC is '0' and INTN = 1. These conditions must be detected and cleared in software. Similarly, a false interrupt could be generated if INTP or INTN is set while the CLC module is enabled.

The user should be sure to clear any spurious interrupt events that may occur in the initialization process of the CLC module.

If the CLCx Interrupt Enable bit, CLCxIE, is cleared, an interrupt will not be generated. However, the CLCxIF bit will still be set if an Interrupt condition occurs. The user can clear the interrupt in the Interrupt Service Routine (ISR) by clearing CLCxIF. See **Interrupt Controller** for more information.

29.7. Power-Saving Modes

29.7.1. Sleep Mode

The CLC module is not affected by Sleep mode since it does not rely on system clock sources for an operation. However, some input sources might be disabled during Sleep, so the function could be disrupted. If the source continues to operate, so will the module. Refer to the source peripheral's section for more information on its operation in Sleep mode.

29.7.2. Idle Mode

The CLC module is not affected by Idle mode since it does not rely on system clock sources for operation. However, some input sources might be disabled during Idle and the function could be disrupted. If the sources continues to operate, so will the module. Refer to the source peripheral's section for more information on its operation in Idle mode.

30. Peripheral Trigger Generator (PTG)

The dsPIC33AK256MPS306 family Peripheral Trigger Generator (PTG) module is a user-programmable sequencer that is capable of generating complex trigger signal sequences to coordinate the operation of other peripherals. The PTG module interfaces with other modules, such as an Analog-to-Digital Converter (ADC), output compare and PWM modules, timers and interrupt controllers.

The PTG consists of the following key features:

- Behavior is Step Command Driven:
  - Step commands are eight bits wide.
- Commands are Stored in a Step Queue:
  - Queue depth is up to 32 entries.
  - Programmable step execution time (Step delay)
- Supports the Command Sequence Loop:
  - Can be nested one-level deep.
  - Conditional or unconditional loop
  - Two 16-bit loop counters
- Up to 16 Hardware Input Triggers:
  - Sensitive to either positive or negative edges, or a high or low level
- One Software Input Trigger
- Generates up to 32 Unique Output Trigger Signals
- Generates Two Types of Trigger Outputs:
  - Individual
  - Broadcast
- Strobed Output Port for Literal Data Values:
  - 5-bit literal write (literal part of a command)
  - 16-bit literal write (literal held in the PTGL0 register)
- Generates up to 10 Unique Interrupt Signals
- Two 16-Bit General Purpose Timers
- Flexible Self-Contained Watchdog Timer (WDT) to Set an Upper Limit to Trigger Wait Time
- Single-Step Command Capability in Debug Mode
- Configurable Clock from Dedicated Clock Generator Module
- Programmable Clock Divider

30.1. Device-Specific Information

Table 30-1. PTG Summary Table

PTG Module Instances	Input Trigger Sources from other Peripheral Modules	Output Triggers to other Peripheral Modules	Peripheral Bus Speed	Clock Source
1	15	16	Slow (1:4 of CPU Clock)	CLKGEN11

Table 30-2. PTG Input Descriptions

PTG Input Number	PTG Input Description
PTG Trigger Input 0	Trigger Input from PWM1 ADC Trigger 2

Table 30-2. PTG Input Descriptions (continued)

PTG Input Number	PTG Input Description
PTG Trigger Input 1	Trigger Input from PWM2 ADC Trigger 2
PTG Trigger Input 2	Trigger Input from PWM3 ADC Trigger 2
PTG Trigger Input 3	DMA0 Interrupt
PTG Trigger Input 4	DMA1 Interrupt
PTG Trigger Input 5	DMA6 Interrupt
PTG Trigger Input 6	DMA7 Interrupt
PTG Trigger Input 7	Trigger Input from SCCP3
PTG Trigger Input 8	Trigger input from SCCP4
PTG Trigger Input 9	Trigger input from SCCP5
PTG Trigger Input 10	Trigger Input from Comparator 2
PTG Trigger Input 11	Trigger Input from Comparator 3
PTG Trigger Input 12	Trigger Input from CLC1
PTG Trigger Input 13	Trigger Input from ADC1 Ready Signal
PTG Trigger Input 14	Reserved
PTG Trigger Input 15	Trigger Input from INT2 PPS

Table 30-3. PTG Output Descriptions

PTG Output Number	PTG Output Description
PTGO0 to PTGO11	Reserved
PTGO12	ADC TRGSRC[18]
PTGO13	ITC SSRC[16]
PTGO14	CCP Sync Source #14
PTGO15	CCP Sync Source #13
PTGO16	CCP Sync Source #12
PTGO17	CCP Sync Source #1
PTGO18	PPS Output 67
PTGO19	PPS Output 68
PTGO20	PPS Output 69
PTGO21	PPS Input 182
PTGO22	PPS Input 183
PTGO23	PPS Input 184
PTGO24	PPS Output 55
PTGO25	PPS Output 56
PTGO26	PPS Output 137
PTGO27	PPS Output 138
PTGO28-PTGO31	Reserved

Table 30-4. PTG Interrupts

PTG Interrupts
_PTG0Interrupt
_PTG1Interrupt
_PTG2Interrupt
_PTG3Interrupt

Note: PTG strobe is not supported on the dsPIC33AK256MPS306 device family.

30.2. Architectural Overview

The PTG module is a user-programmable sequencer for generating complex peripheral trigger sequences. The PTG module provides the ability to schedule complex peripheral operations, which would be difficult or impossible to achieve via a software solution.

The user writes 8-bit commands, called step commands, to the PTG Queue registers (PTGQUE0-PTGQUE7). Each 8-bit step command is made up of a command code and an option field. Table 30-5 shows the format and encoding of a step command. Based on the commands, the PTG can interact with other peripherals, such as the PWM, ADC, SCCP/MCCP and Peripheral Pin Select. See the device-specific data sheet for the availability of peripherals.

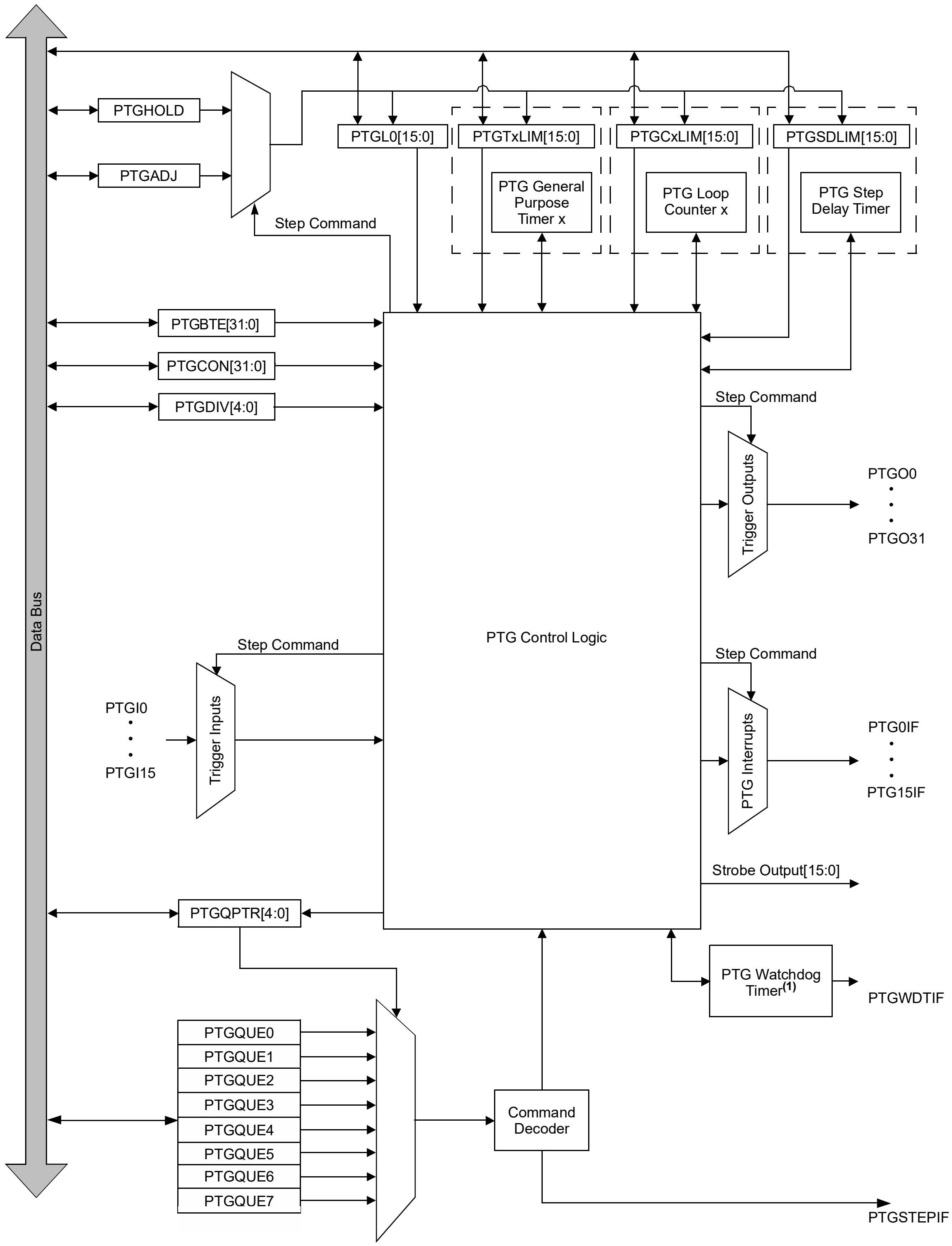
30.2.1. Step Commands and Format

The PTG operates using 11 8-bit step commands to perform higher level tasks. There are four types of step commands:

- Input Event Control
- Control Functions
- Flow Control
- Output Generation

Combinations and sequences of these commands can be used to make decisions and take action without CPU intervention.

Figure 30-1. PTG Block Diagram



Note: This is a dedicated Watchdog Timer for the PTG module and is independent of the device Watchdog Timer.

30.2.1.1. Input Event Control

There are two input event control commands, `PTGWHI` and `PTGWLO`, that wait for a high or low edge on one of the 16 PTGIx inputs. These commands are used in conjunction with a 4-bit OPTION field that specifies which PTGIx is used. Once the specified input transitions in the intended direction, the step queue is incremented and the next step command is evaluated. See Wait for Trigger Input for additional information.

30.2.1.2. Control Functions

There are three commands for control functions, `PTGCTRL`, `PTGADD` and `PTGCOPY`. The `PTGCTRL` command controls the operation of the delay timers, software triggers and the strobe output. The `PTGADD` command is used to add the contents of the PTGHOLD register to other PTG registers, including the counters, timers, step delay and Literal register. The `PTGCOPY` command is used to copy the contents of the PTGHOLD register to other PTG registers, similar to the `PTGADD` command.

30.2.1.3. Flow Control

Flow control is accomplished using the three jump commands, `PTGJMP`, `PTGJMPC0` and `PTGJMPC1`. These jump commands are three bits to allow a larger 5-bit OPTION to match that of the Queue Pointer, PTGQPTR. The `PTGJMP` command simply jumps to the specified queue location, whereas the `PTGJMPCx` commands are conditional jumps based on the comparison of the counters to the PTG Counter Limit registers (PTGCxLIM).

30.2.1.4. Output Generation

Output generation is achieved using the `PTGTRIG`, `PTGIRQ` and `PTGSTRB` commands. `PTGTRIG` is used to select and generate an output trigger (PTGOx). `PTGTRIG` also uses a 5-bit OPTION field to support the 32 PTGOs' selections. The `PTGIRQ` command is used to generate an interrupt request with the OPTION field specifying the interrupt (PTGxIF). See PTG Module Outputs for additional information on triggers and interrupts. The `PTGSTRB` command is used to generate a strobe output, which outputs 16 bits of data to another peripheral, if implemented. See Strobe Output for additional information.

Table 30-5 provides an overview of the PTG commands, and Table 30-6 elaborates on the options available for each command. Example 30-1, Example 30-2 and Example 30-3 provide C code examples for command definitions and their options. Later examples in this FRM refer back to these examples.

Table 30-5. PTG Step Command Format and Description

Step Command Byte			
STEPx[7:0]			
CMD[3:0]		OPTION[3:0]	
bit 7	bit 4	bit 3	bit 0

Bits	Step Command	CMD[3:0]	Command Description
bits 7:4	`PTGCTRL`(1)	`0000`	Execute the control command as described by the OPTION[3:0] bits.
	`PTGADD`(1)	`0001`	Add contents of the PTGADJ register to the target register as described by the OPTION[3:0] bits.
	`PTGCOPY`(1)		Copy contents of the PTGHOLD register to the target register as described by the OPTION[3:0] bits.
	`PTGSTRB`	`001x`	Copy the values contained in the bits, CMD[0]:OPTION[3:0], to the strobe output bits[4:0].
	`PTGWHI`(2)	`0100`	Wait for a low-to-high edge input from a selected PTG trigger input as described by the OPTION[3:0] bits.
	`PTGWLO`(2)	`0101`	Wait for a high-to-low edge input from a selected PTG trigger input as described by the OPTION[3:0] bits.
	—	`0110`	Reserved; do not use.(1)
	`PTGIRQ`(1)	`0111`	Generate an individual interrupt request as described by the OPTION[3:0] bits.
	`PTGTRIG`(1)	`100x`	Generate an individual trigger output as described by the 5-bit field of CMD[0]:OPTION[3:0].
	`PTGJMP`	`101x`	Copy the values contained in the bits, CMD[0]:OPTION[3:0], to the PTGQPTR register and jump to that position in the step queue.
	`PTGJMPC0`	`110x`	PTGC0 = PTGC0LIM: Increment the PTGQPTR register.
			PTGC0 ≠ PTGC0LIM: Increment Counter 0 (PTGC0) and copy the values contained in the bits, CMD[0]:OPTION[3:0], to the PTGQPTR register and jump to that position in the step queue.
	`PTGJMPC1`	`111x`	PTGC1 = PTGC1LIM: Increment the PTGQPTR register.
			PTGC1 ≠ PTGC1LIM: Increment Counter 1 (PTGC1) and copy the values contained in the bits, CMD[0]:OPTION[3:0], to the PTGQPTR register and jump to that position in the step queue.

Notes:

1. Reserved commands or options will execute, but they do not have any effect (i.e., they execute as a `NOP` instruction).
2. Reserved input trigger options must not be used with PTGWHI/PTGWLO.

Example 30-1. PTG Command Definitions

```
/* PTG Commands */
#define PTGCTRL(x)      ((0b00000000)     | ((x) & 0b00001111))
#define PTGADD(x)       ((0b00010000)     | ((x) & 0b00000111))
#define PTGCOPY(x)      ((0b00011000)     | ((x) & 0b00000111))
#define PTGSTRB(x)      ((0b00100000)     | ((x) & 0b00011111))
#define PTGWHI(x)       ((0b01000000)     | ((x) & 0b00001111))
#define PTGWLO(x)       ((0b01010000)     | ((x) & 0b00001111))
#define PTGIRQ(x)       ((0b01110000)     | ((x) & 0b00001111))
#define PTGTRIG(x)      ((0b10000000)     | ((x) & 0b00011111))
#define PTGJMP(x)       ((0b10100000)     | ((x) & 0b00011111))
```

```
#define PTGJMPC0(x)     ((0b11000000)     | ((x) & 0b00011111))
#define PTGJMPC1(x)     ((0b11100000)     | ((x) & 0b00011111))
```

Table 30-6. PTG Command Options

Bits	Step Command	OPTION[3:0]	Command Description
bits 3:0	PTGCTRL[1]	0000	NOP
		0001	Reserved; do not use.
		0010	Disable Step Delay Timer (PTGSD)
		0011	Reserved; do not use.
		0100	Reserved; do not use.
		0101	Reserved; do not use.
		0110	Enable Step Delay Timer (PTGSD)
		0111	Reserved; do not use.
		1000	Start PTGT0 and wait for its value to match the PTGT0LIM register.
		1001	Start PTGT1 and wait for its value to match the PTGT1LIM register.
		1010	Wait for the software trigger (level, PTGSWT = 1).
		1011	Wait for the software trigger (positive edge, PTGSWT = 0 to 1).
		1100	Copy the PTGC0LIM register contents to the strobe output.
		1101	Copy the PTGC1LIM register contents to the strobe output.
		1110	Copy the PTGL0 register contents to the strobe output.
		1111	Generate the triggers indicated in the PTGBTE register.
	PTGADD[1]	0000	Add the PTGADJ register contents to the PTGC0LIM register.
		0001	Add the PTGADJ register contents to the PTGC1LIM register.
		0010	Add the PTGADJ register contents to the PTGT0LIM register.
		0011	Add the PTGADJ register contents to the PTGT1LIM register.
		0100	Add the PTGADJ register contents to the PTGSDLIM register.
		0101	Add the PTGADJ register contents to the PTGL0 register.
		0110	Reserved; do not use.
		0111	Reserved; do not use.

Notes:

1. All reserved options for this command will execute, but they do not have any effect (i.e., they execute as a NOP instruction).
2. Reserved input trigger options must not be used with PTGWHI/PTGWLO.

Bits	Step Command	OPTION[3:0]	Command Description
PTG Command Options (Continued)			

Output Generation (continued)

Bits	Step Command	OPTION[3:0]	Command Description
bits 3:0	PTGCOPY(1)	1000	Copy the PTGHOLD register contents to the PTGC0LIM register.
		1001	Copy the PTGHOLD register contents to the PTGC1LIM register.
		1010	Copy the PTGHOLD register contents to the PTGT0LIM register.
		1011	Copy the PTGHOLD register contents to the PTGT1LIM register.
		1100	Copy the PTGHOLD register contents to the PTGSDLIM register.
		1101	Copy the PTGHOLD register contents to the PTGL0 register.
		1110	Reserved; do not use.
		1111	Reserved; do not use.
	PTGWHI(2) or PTGWLO(2)	0000	PTGI0 (see Table 30-2)
		...	...
		1111	PTGI15 (see Table 30-2 and Table 30-3)
	PTGIRQ(1)	0000	Generate PTG Interrupt 0
		...	...
		0011	Generate PTG Interrupt 4
	PTGTRIG	0000	PTGO0 (see Table 30-3)
		0001	PTGO1
		...	...
		1110	PTGO30
		1111	PTGO31 (see Table 30-3)

Notes:

1. All reserved options for this command will execute, but they do not have any effect (i.e., they execute as a NOP instruction).
2. Reserved input trigger options must not be used with PTGWHI/PTGWLO.

Example 30-2. `PTGCTRL` Options

```
// Used with PTGCTRL command
typedef    enum
{
           stepDelayDisable = 0b0010,
           ptgNop = 0b0000;
           stepDelayEnable = 0b0110,
           t0Wait = 0b1000,
           t1Wait = 0b1001,
           softTriggerLevelWait = 0b1010,
           softTriggerEdgeWait = 0b1011,
           c0Strobe = 0b1100,
           c1Strobe = 0b1101,
           l0Strobe = 0b1110,
           triggerGenerate = 0b1111,
} CTRL_T;
```

Example 30-3. Options for `PTGADD` and `PTGCOPY` Commands

```
// Used with PTGADD and PTGCOPY commands
typedef    enum
{
           c0Limit = 0b0000,
           c1Limit = 0b0001,
           t0Limit = 0b0010,
           t1Limit = 0b0011,
           stepDelay = 0b0100,
           literal0 = 0b0101,
} ADD_COPY_T;
```

30.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3500	PTGCON	31:24	Reserved[2:0]			PTGDIV[4:0]				
		23:16	PTGPWD[3:0]					PTGWDT[2:0]		
		15:8	ON		SIDL	PTGTOGL		PTGSWT	PTGSSEN	PTGIVIS
		7:0	PTGSTRT	PTGWDTO	PTGBUSY				PTGITM[1:0]	
0x3504	PTGBTE	31:24	PTGBTE[31:24]							
		23:16	PTGBTE[23:16]							
		15:8	PTGBTE[15:8]							
		7:0	PTGBTE[7:0]							
0x3508	PTGHOLD	31:24								
		23:16								
		15:8	PTGHOLD[15:8]							
		7:0	PTGHOLD[7:0]							
0x350C	PTGT0LIM	31:24								
		23:16								
		15:8	PTGT0LIM[15:8]							
		7:0	PTGT0LIM[7:0]							
0x3510	PTGT1LIM	31:24								
		23:16								
		15:8	PTGT1LIM[15:8]							
		7:0	PTGT1LIM[7:0]							
0x3514	PTGSDLIM	31:24								
		23:16								
		15:8	PTGSDLIM[15:8]							
		7:0	PTGSDLIM[7:0]							
0x3518	PTGC0LIM	31:24								
		23:16								
		15:8	PTGC0LIM[15:8]							
		7:0	PTGC0LIM[7:0]							
0x351C	PTGC1LIM	31:24								
		23:16								
		15:8	PTGC1LIM[15:8]							
		7:0	PTGC1LIM[7:0]							
0x3520	PTGADJ	31:24								
		23:16								
		15:8	PTGADJ[15:8]							
		7:0	PTGADJ[7:0]							
0x3524	PTGL0	31:24								
		23:16								
		15:8	PTGL0[15:8]							
		7:0	PTGL0[7:0]							
0x3528	PTGQPTR	31:24								
		23:16								
		15:8								
		7:0				PTGQPTR[4:0]				
0x352C ... 0x352F	Reserved									
0x3530	PTGQUE0	31:24	STEP3[7:0]							
		23:16	STEP2[7:0]							
		15:8	STEP1[7:0]							
		7:0	STEP0[7:0]							
0x3534	PTGQUE1	31:24	STEP7[7:0]							
		23:16	STEP6[7:0]							
		15:8	STEP5[7:0]							
		7:0	STEP4[7:0]							
0x3538	PTGQUE2	31:24	STEP11[7:0]							
		23:16	STEP10[7:0]							
		15:8	STEP9[7:0]							
		7:0	STEP8[7:0]							

Register Summary (continued)

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x353C	PTGQUE3	31:24	STEP15[7:0]							
		23:16	STEP14[7:0]							
		15:8	STEP13[7:0]							
		7:0	STEP12[7:0]							
0x3540	PTGQUE4	31:24	STEP19[7:0]							
		23:16	STEP18[7:0]							
		15:8	STEP17[7:0]							
		7:0	STEP16[7:0]							
0x3544	PTGQUE5	31:24	STEP23[7:0]							
		23:16	STEP22[7:0]							
		15:8	STEP21[7:0]							
		7:0	STEP20[7:0]							
0x3548	PTGQUE6	31:24	STEP27[7:0]							
		23:16	STEP26[7:0]							
		15:8	STEP25[7:0]							
		7:0	STEP24[7:0]							
0x354C	PTGQUE7	31:24	STEP31[7:0]							
		23:16	STEP30[7:0]							
		15:8	STEP29[7:0]							
		7:0	STEP28[7:0]							

30.3.1. PTG Control Register

Name: PTGCON
Offset: 0x3500

Legend: R = Readable bit, W = Writable bit, HC = Hardware Clearable bit, HS = Hardware Settable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. This bit is only used with the `PTGCTRL 0b1010/0b1011` step command software trigger options. Refer to Wait for Software Trigger for more information.
3. The PTGSSEN bit may only be written during a debugging session. See Single-Step Mode for more information.
4. These bits apply to the `PTGWHI` and `PTGWLO` commands only.

Bit	31	30	29	28	27	26	25	24
	Reserved[2:0]			PTGDIV[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PTGPWD[3:0]					PTGWDT[2:0]		
Access	R/W	R/W	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	PTGTOGL		PTGSWT	PTGSSEN	PTGIVIS
Access	R/W		R/W	R/W		R/W/HC	R/W	R/W
Reset	0		0	0		0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGSTRT	PTGWDTO	PTGBUSY				PTGITM[1:0]	
Access	R/W/HC	R/W/HS	R/HS/HC				R/W	R/W
Reset	0	0	0				0	0

Bits 31:29 – Reserved[2:0] Maintain as '`0`'

Bits 28:24 – PTGDIV[4:0] PTG Module Clock Prescaler (Divider) bits(1)

Value	Description
`11111`	Divide by 32
`11110`	Divide by 31
...	
`00001`	Divide by 2
`00000`	Divide by 1

Bits 23:20 – PTGPWD[3:0] PTG Trigger Output Pulse-Width (in PTG clock cycles) bits(1)

Value	Description
`1111`	All trigger outputs are 16 PTG clock cycles wide.
`1110`	All trigger outputs are 15 PTG clock cycles wide.
...	
`0001`	All trigger outputs are two PTG clock cycles wide.
`0000`	All trigger outputs are one PTG clock cycle wide.

Bits 18:16 – PTGWDT[2:0] PTG Watchdog Timer Time-out Selection bits(1)

Value	Description
111	Watchdog Timer will time-out after 512 PTG clocks.
110	Watchdog Timer will time-out after 256 PTG clocks.
101	Watchdog Timer will time-out after 128 PTG clocks.
100	Watchdog Timer will time-out after 64 PTG clocks.
011	Watchdog Timer will time-out after 32 PTG clocks.
010	Watchdog Timer will time-out after 16 PTG clocks.
001	Watchdog Timer will time-out after eight PTG clocks.
000	Watchdog Timer is disabled.

Bit 15 – ON PTG Enable bit

Value	Description
1	PTG is enabled.
0	PTG is disabled.

Bit 13 – SIDL PTG Stop in Idle Mode bit

Value	Description
1	Halts PTG operation when device is Idle.
0	PTG operation continues when device is Idle.

Bit 12 – PTGTOGL PTG Toggle Trigger Output bit(1)

Value	Description
1	Toggles state of TRIG output for each execution of `PTGTRIG`.
0	Generates a single TRIG pulse for each execution of `PTGTRIG`.

Bit 10 – PTGSWT PTG Software Trigger bit(2)

Value	Description
1	Asserts the PTG software trigger.
0	Deasserts the PTG software trigger (Level-Sensitive mode)/cleared by hardware (Edge-Sensitive mode).

Bit 9 – PTGSSEN PTG Single-Step Enable bit(3)

<u>If in Debug mode:</u>
1 = Enables Single-Step mode.
0 = Disables Single-Step mode.
<u>If not in Debug mode:</u>
Writes have no effect; read as '0'.

Bit 8 – PTGIVIS PTG Internal Counter/Timer Visibility bit(1,4)

Value	Description
1	Reading the PTGSDLIM, PTGCxLIM or PTGTxLIM registers returns the current values of their corresponding internal Counter/Timer registers (PTGSD, PTGCx and PTGTx).
0	Reading the PTGSDLIM, PTGCxLIM or PTGTxLIM registers returns the value of these Limit registers.

Bit 7 – PTGSTRT PTG Start Sequencer bit(3)

<u>If not in Single-Step mode:</u>
1 = Starts to sequentially execute the commands.
0 = Stops executing the commands.
<u>If in Single-Step mode:</u>
1 = Executes the next step command, then halts the sequencer.

0 = Manually halts the sequencer/execution of signal command has completed (cleared by hardware).

Bit 6 – PTGWDTO PTG Watchdog Timer Time-out Status bit[1]

Value	Description
1	PTG Watchdog Timer has timed out.
0	PTG Watchdog Timer has not timed out.

Bit 5 – PTGBUSY PTG State Machine Busy bit

Value	Description
1	PTG is running on the selected clock source.
0	PTG state machine is not running.

Bits 1:0 – PTGITM[1:0] PTG Input Trigger Operation Selection bits[1,4]

Value	Description
11	Single-level detect with step delay not executed on exit of command, regardless of the PTGCTRL command (Mode 3).
10	Single-level detect with step delay executed on exit of command (Mode 2).
01	Continuous edge detect with step delay not executed on exit of command, regardless of the PTGCTRL command (Mode 1).
00	Continuous edge detect with step delay executed on exit of command (Mode 0).

30.3.2. PTG Broadcast Trigger Enable Register

Name: PTGBTE
Offset: 0x3504

Note:
1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
	PTGBTE[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PTGBTE[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PTGBTE[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGBTE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PTGBTE[31:0] PTG Broadcast Trigger Enable bits[(1)]

Value	Description
1	Generates PTG output trigger corresponding to bit number when the broadcast command is executed.
0	Does not generate corresponding trigger when the broadcast command is executed.

30.3.3. PTG Hold Register

Name: PTGHOLD
Offset: 0x3508

Note:
1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGHOLD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGHOLD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGHOLD[15:0] PTG General Purpose Hold Register bits(1)
This register holds the user-supplied data to be copied to the PTGTxLIM, PTGCxLIM, PTGSDLIM or PTGL0 register using the `PTGCOPY` command.

30.3.4. PTG Timer0 Limit Register

Name: PTGT0LIM
Offset: 0x350C

Legend: R = Readable bit, W = Writable bit

Notes:

1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGT0LIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGT0LIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGT0LIM[15:0] PTG Timer0 Limit Register bits[1,2]
General purpose Timer0 Limit register.

30.3.5. PTG Timer1 Limit Register

Name: PTGT1LIM
Offset: 0x3510

Legend: R = Readable bit, W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGT1LIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGT1LIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGT1LIM[15:0] PTG Timer1 Limit Register bits[1,2]
General purpose Timer1 Limit register.

30.3.6. PTG Step Delay Limit Register

Name: PTGSDLIM
Offset: 0x3514

Legend: R = Readable bit, W = Writable bit

Notes:

1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGSDLIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGSDLIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGSDLIM[15:0] PTG Step Delay Limit Register bits[1,2]
This register holds a PTG step delay value representing the number of additional PTG clocks between the start of a step command and the completion of a step command.

30.3.7. PTG Counter 0 Limit Register

Name: PTGC0LIM
Offset: 0x3518

Legend: R = Readable bit, W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGC0LIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGC0LIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGC0LIM[15:0] PTG Counter 0 Limit Register bits[1,2]
This register is used to specify the loop count for the `PTGJMPC0` step command or as a Limit register for the General Purpose Counter 0.

30.3.8. PTG Counter 1 Limit Register

Name: PTGC1LIM
Offset: 0x351C

Legend: R = Readable bit, W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. The value read from these register bits depends on the PTGIVIS bit (PTGCON[8]). Refer to Control Register Access for more information.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGC1LIM[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGC1LIM[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGC1LIM[15:0] PTG Counter 1 Limit Register bits[1,2]
This register is used to specify the loop count for the `PTGJMPC1` step command or as a Limit register for the General Purpose Counter 1.

30.3.9. PTG Adjust Register

Name: PTGADJ
Offset: 0x3520

Legend: R = Readable bit; W = Writable bit

Note:

1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGADJ[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGADJ[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGADJ[15:0] PTG Adjust Register bits[(1)]
This register holds the user-supplied data to be added to the PTGTxLIM, PTGCxLIM, PTGSDLIM or PTGL0 register using the `PTGADD` command.

30.3.10. PTG Literal 0 Register

Name: PTGL0
Offset: 0x3524

Legend: R = Readable bit; W = Writable bit

Note:

1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	PTGL0[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PTGL0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – PTGL0[15:0] PTG Literal 0 Register bits[(1)]

This register holds a 16-bit value to be written by the strobe output using the `PTGCTRL` command.

30.3.11. PTG Step Queue Pointer Register

Name: PTGQPTR
Offset: 0x3528

Legend: R = Readable bit; W = Writable bit

Note:

1. These bits are read-only when the module is executing step commands.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
				PTGQPTR[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 4:0 – PTGQPTR[4:0] PTG Step Queue Pointer Register bits[(1)]
This register points to the currently active step command in the step queue.

30.3.12. PTG Step Queue 0 Pointer Register

Name: PTGQUE0
Offset: 0x3530

Legend: R = Readable bit; W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 30-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP3[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP3[7:0] PTG Command Step 3 bits[1,2]
A queue location for storage of the STEP3 command byte.

Bits 23:16 – STEP2[7:0] PTG Command Step 2 bits[1,2]
A queue location for storage of the STEP2 command byte.

Bits 15:8 – STEP1[7:0] PTG Command Step 1 bits[1,2]
A queue location for storage of the STEP1 command byte.

Bits 7:0 – STEP0[7:0] PTG Command Step 0 bits[1,2]
A queue location for storage of the STEP0 command byte.

30.3.13. PTG Step Queue 1 Pointer Register

Name: PTGQUE1
Offset: 0x3534

Legend: R = Readable bit; W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 30-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP7[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP6[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP5[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP4[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP7[7:0] PTG Command Step 7 bits[1,2]
A queue location for storage of the STEP7 command byte.

Bits 23:16 – STEP6[7:0] PTG Command Step 6 bits[1,2]
A queue location for storage of the STEP6 command byte.

Bits 15:8 – STEP5[7:0] PTG Command Step 5 bits[1,2]
A queue location for storage of the STEP5 command byte.

Bits 7:0 – STEP4[7:0] PTG Command Step 4 bits[1,2]
A queue location for storage of the STEP4 command byte.

30.3.14. PTG Step Queue 2 Pointer Register

Name: PTGQUE2
Offset: 0x3538

Legend: R = Readable bit; W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 30-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP11[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP10[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP9[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP8[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP11[7:0] PTG Command Step 11 bits[1,2]
A queue location for storage of the STEP11 command byte.

Bits 23:16 – STEP10[7:0] PTG Command Step 10 bits[1,2]
A queue location for storage of the STEP10 command byte.

Bits 15:8 – STEP9[7:0] PTG Command Step 9 bits[1,2]
A queue location for storage of the STEP9 command byte.

Bits 7:0 – STEP8[7:0] PTG Command Step 8 bits[1,2]
A queue location for storage of the STEP8 command byte.

30.3.15. PTG Step Queue 3 Pointer Register

Name: PTGQUE3
Offset: 0x353C

Legend: R = Readable bit; W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 30-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP15[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP14[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP13[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP12[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP15[7:0] PTG Command Step 15 bits(1,2)
A queue location for storage of the STEP15 command byte.

Bits 23:16 – STEP14[7:0] PTG Command Step 14 bits(1,2)
A queue location for storage of the STEP14 command byte.

Bits 15:8 – STEP13[7:0] PTG Command Step 13 bits(1,2)
A queue location for storage of the STEP13 command byte.

Bits 7:0 – STEP12[7:0] PTG Command Step 12 bits(1,2)
A queue location for storage of the STEP12 command byte.

30.3.16. PTG Step Queue 4 Pointer Register

Name: PTGQUE4
Offset: 0x3540

Legend: R = Readable bit; W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 30-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP19[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP18[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP17[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP16[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP19[7:0] PTG Command Step 19 bits[1,2]
A queue location for storage of the STEP19 command byte.

Bits 23:16 – STEP18[7:0] PTG Command Step 18 bits[1,2]
A queue location for storage of the STEP18 command byte.

Bits 15:8 – STEP17[7:0] PTG Command Step 17 bits[1,2]
A queue location for storage of the STEP17 command byte.

Bits 7:0 – STEP16[7:0] PTG Command Step 16 bits[1,2]
A queue location for storage of the STEP16 command byte.

30.3.17. PTG Step Queue 5 Pointer Register

Name: PTGQUE5
Offset: 0x3544

Legend: R = Readable bit; W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 30-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP23[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP22[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP21[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP20[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP23[7:0] PTG Command Step 23 bits[1,2]
A queue location for storage of the STEP3 command byte.

Bits 23:16 – STEP22[7:0] PTG Command Step 22 bits[1,2]
A queue location for storage of the STEP22 command byte.

Bits 15:8 – STEP21[7:0] PTG Command Step 21 bits[1,2]
A queue location for storage of the STEP21 command byte.

Bits 7:0 – STEP20[7:0] PTG Command Step 20 bits[1,2]
A queue location for storage of the STEP20 command byte.

30.3.18. PTG Step Queue 6 Pointer Register

Name: PTGQUE6
Offset: 0x3548

Legend: R = Readable bit; W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 30-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP27[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP26[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP25[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP24[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP27[7:0] PTG Command Step 27 bits[1,2]
A queue location for storage of the STEP27 command byte.

Bits 23:16 – STEP26[7:0] PTG Command Step 26 bits[1,2]
A queue location for storage of the STEP26 command byte.

Bits 15:8 – STEP25[7:0] PTG Command Step 25 bits[1,2]
A queue location for storage of the STEP25 command byte.

Bits 7:0 – STEP24[7:0] PTG Command Step 24 bits[1,2]
A queue location for storage of the STEP24 command byte.

30.3.19. PTG Step Queue 7 Pointer Register

Name: PTGQUE7
Offset: 0x354C

Legend: R = Readable bit; W = Writable bit

Notes:
1. These bits are read-only when the module is executing step commands.
2. Refer to Table 30-5 for the step command encoding.

Bit	31	30	29	28	27	26	25	24
	STEP31[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	STEP30[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	STEP29[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	STEP28[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:24 – STEP31[7:0] PTG Command Step 31 bits[1,2]
A queue location for storage of the STEP31 command byte.

Bits 23:16 – STEP30[7:0] PTG Command Step 30 bits[1,2]
A queue location for storage of the STEP30 command byte.

Bits 15:8 – STEP29[7:0] PTG Command Step 29 bits[1,2]
A queue location for storage of the STEP29 command byte.

Bits 7:0 – STEP28[7:0] PTG Command Step 28 bits[1,2]
A queue location for storage of the STEP28 command byte.

30.4. Operation

30.4.1. Basic Operation

The user loads the step commands (8-bit values) into the PTG Queue registers. The commands define a sequence of events for generating the trigger output signals to the peripherals. The step commands can also be used to generate the interrupt requests to the processor.

The PTG module is enabled and clocked when the ON bit (PTGCON[15]) = `1`. While the PTGSTRT bit (PTGCON[7]) = `0`, the PTG module is in the Halt state.

Notes:
1. The control registers cannot be modified when the PTG module is in the Halt state.
2. The PTG module must be enabled (ON = `1`) prior to attempting to set the PTGSTRT bit.
3. The user should not attempt to set the ON and PTGSTRT bits within the same data write cycle.

Subsequently, setting PTGSTRT = `1` will enable the module for Continuous mode execution of the step command queue. The PTG sequencer will start to read the step queue at the address held in the Queue Pointer (PTGQPTR). Each command byte is read, decoded and executed sequentially. The minimum duration of any step command is one PTG clock as explained in PTG Clock Selection.

Step commands will execute sequentially until any of the following occurs:
- A `PTGJMP`, `PTGJMPC0` or `PTGJMPC1` (flow change) step command is executed.
- The user clears the PTGSTRT bit, stopping the PTG Sequencer. No further step commands are read/decoded and execution halts.
- The internal Watchdog Timer overflows, clearing the PTGSTRT bit and stopping the PTG sequencer. No further step commands are read/decoded and execution halts.
- The PTG module is disabled (ON = `0`).

The step commands can also be made to wait on a condition, such as an input trigger edge, a software trigger or a timer match, before continuing execution. For more information, refer to Stopping the Sequencer.

30.4.2. PTG Clock Selection

The PTG module has multiple clock options and has a selectable prescaler, which divides the PTG clock input from one to 32.

30.4.2.1. Clock Source Selection

The PTG clock source is determined by a dedicated device clock generator, external to the PTG. The assignment of clock generators is device-specific.

30.4.2.2. Clock Prescaler Selection

The PTGDIV[4:0] bits (PTGCON[28:24]) specify the prescaler value for the PTG clock generation logic. These bits can be written only when the PTG module is disabled (ON = `0`).

Note: Any attempt to write to the PTGDIV[4:0] bits while ON = `1` will have no effect.

30.4.2.3. Module Enable Delay

Once the PTG module is enabled (ON = `1`), there is a delay before the PTG starts to execute commands. This delay is expressed in Equation 30-1. The PTG clock period is the effective clock after the prescaler.

Equation 30-1. Enable Delay

$$T_{DLYEN} = 4 \cdot \text{PTG Clock Period (Maximum)}$$

The user must ensure that no control bits are modified during the delay. Also, no external triggers may be asserted prior to the PTG state machine commencing execution; otherwise, the triggers will be missed.

30.4.3. Control Register Access

When the PTG module is enabled (ON = `1`), writes are inhibited to the PTGDIV[4:0] bits in PTGCON. Other bits and control registers may be written to as long as PTGSTRT = `0`. See PTGCON for more information. When the PTG module is actively executing code (ON = `1` and PTGSTART = `1`), only the ON, PTGSWT, PTGSIDL and PTGSTRT bits in PTGCON are writable. All other bits in PTGCON and other control registers are read-only.

When the PTG module is enabled (ON = 1), reads can be performed from any control register at any time; however, the data read from certain control registers depends on the PTGSTRT and PTGIVIS bits.

When the PTG module is disabled (ON = 0), all control registers can be read and written to as normal. The PTGIVIS bit has no effect when ON = 0; all Timer/Counter registers will read as their Limit register values.

30.4.3.1. Internal Control Register Visibility

For debugging purposes, some registers internal to the PTG can be read by the user during PTG operation. When PTGSTRT = 0, a read of any PTG Control register will return the value last written to it. However, when PTGSTRT is set to '1', the values stored in the PTGTxLIM, PTGCxLIM, PTGSDLIM, PTGL0 and PTGQPTR registers are transferred to internal registers, and the user-accessible registers become read-only. Subsequent reads of these registers, as long as PTGSTRT = 1, and for the Timer/Counter registers, PTGIVIS = 0, return the internal register values, which may be modified by step commands.

Note: In order to reliably read the internal registers while the PTG is running, the PTG clock source should be the same as the CPU.

30.4.3.2. Internal Timer/Counter Visibility

When PTGIVIS = 1 during PTG operation, reads of the PTGTxLIM, PTGCxLIM and PTGSDLIM registers return the values of internal counters used to implement the respective timer/counter functionality, instead of the Limit register values. This allows the user to monitor the operation of a timer/counter.

Note: In order to reliably read the internal registers while the PTG is running, the PTG clock source must be the same as the CPU instruction clock (F_{CY}).

30.4.4. Step Queue Pointer

The PTG Step Queue Pointer register (PTGQPTR) addresses the currently active step command in the step queue. While the PTG is not executing step commands (PTGSTRT = 0), any value can be written to PTGQPTR regardless of the state of the ON bit. Once the PTGSTRT bit is set, the PTG begins executing step commands at the queue location indicated by PTGQPTR, and the register becomes read-only. When the PTG is disabled, the PTGQPTR register is cleared once on transition of the ON bit from '1' to '0', after which PTGQPTR becomes writable again.

The user can read the PTGQPTR register at any time. In the Disabled state (ON = 0) and Idle state (ON = 1 and PTGSTRT = 0), a read returns the index of the first step command to execute. In the Active state (ON = 1 and PTGSTRT = 1), a read returns the index of the currently executing step command. The PTGQPTR register is typically incremented during the first cycle of each command. The exceptions to this rule are:

- If the `PTGJMP` command is executed: The Step Queue Pointer is loaded with the target queue address.
- If the `PTGJMPCx` command is executed and PTGCx is less than PTGCxLIM: The Step Queue Pointer is loaded with the target queue address.
- If PTGQPTR points to the last step command in PTGQUEn: The Step Queue Pointer will roll over to '0'.

30.4.5. Command Looping Control

Two 16-bit loop counters are provided (PTGC0 and PTGC1) that can be used by the `PTGJMPCx` command as a block loop counter or delay generator.

Each loop counter consists of an incrementing counter (PTGCx) and a Limit register (PTGCxLIM). The Limit register value can be changed by writing directly to the register (when the module is disabled) or by the PTG sequencer (when the module is enabled). The data read from the Limit register depends upon the state of the PTG Counter/Timer Visibility bit (PTGIVIS). The counters are cleared when the module is in the Reset state or when the PTG module is disabled (ON = 0).

30.4.5.1. Using the Loop Counter as a Block Loop Counter

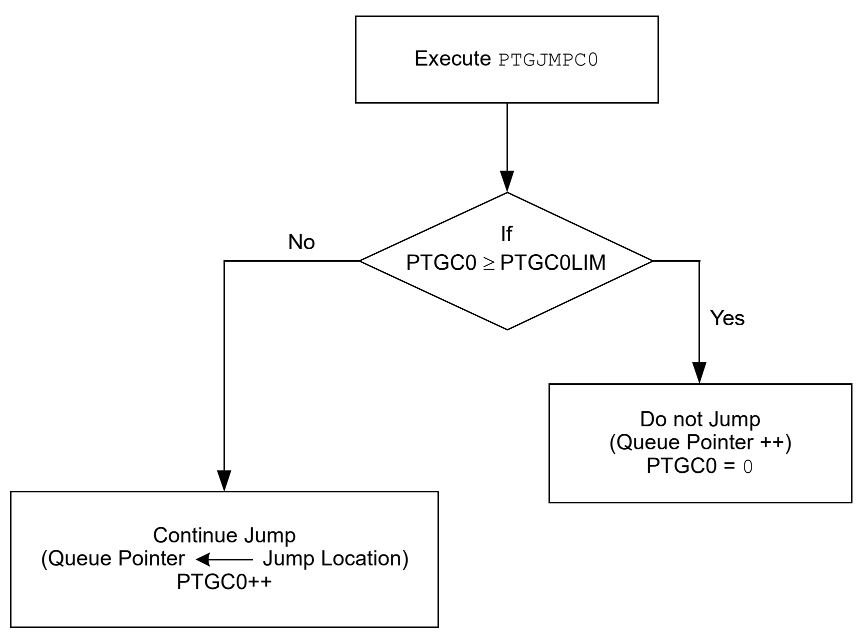
The PTGJMPCx (Jump Conditional) command uses one of the loop counters to keep track of the number of times the PTGJMPCx command is executed, and can, therefore, be used to create code block loops. This is useful in applications where a sequence of peripheral events needs to be repeated several times. The PTGJMPCx command allows the user to create code loops and use fewer step commands.

Each time the PTGJMPCx command is executed, the corresponding internal loop counter is compared to its limit value. If the loop counter has not reached the limit value, the jump location is loaded into the PTGQPTR register and the loop counter is incremented by one. The next command will be fetched from the new queue location. If the counter has reached the limit, the sequencer proceeds to the next command (i.e., increments the Queue Pointer). While preparing for the next PTGJMPCx command loop execution, the corresponding loop counter is cleared (see Figure 30-2).

Note: The loop counter value can be modified (via the PTGADD or PTGCOPY command) prior to execution of the first iteration of the command loop.

The provision for two separate loop counters and associated PTGJMPCx commands allows for the nested loops to be supported (one-level deep). There are no restrictions with regard to which PTGJMPCx command resides in the inner or outer loops.

Figure 30-2. Implementing Block Loop Diagram



30.4.6. Sequencer Operation

All commands are executed in a single cycle, except for the flow change commands and the commands that are waiting for an external input.

30.4.6.1. Step Command Duration

By default, each step command executes in one PTG clock period. There are several methods to slow the execution of the step commands:

- Wait for a trigger input.

- Wait for a GP timer (PTGTxLIM).
- Insert a delay loop using the `PTGJMP` and `PTGJMPCx` commands.
- Enable and insert a step delay (PTGSDLIM) after execution of each command.

30.4.6.2. Wait for Trigger Input

The PTG module can support up to 16 independent trigger inputs. The user can specify a step command that waits for a positive or negative edge, or a high or low level, of the selected input signal to occur. The Operating mode is selected by the PTGITM[1:0] bits in the PTGCON register.

The `PTGWHI` command looks for a positive edge or High state to occur on the selected trigger input. The `PTGWLO` command looks for a negative edge or Low state to occur on the selected trigger input. The PTG repeats the trigger input command (i.e., effectively waits) until the selected signal becomes valid before continuing the step command execution.

The minimum execution time to wait for a trigger is one PTG clock. There is no limit for the PTG wait for a trigger input other than that enforced by the Watchdog Timer. Refer to PTG Watchdog Timer for more information.

The PTG module supports four Input Trigger Command Operating modes (Mode 0-Mode 3), which are selected by the PTGITM[1:0] bits in the PTGCON register.

Note: If the step delay is disabled, Mode 0 and Mode 1 are equivalent in operation and Mode 2 and Mode 3 are equivalent in operation.

30.4.6.2.1. Mode 0: PTGITM[1:0] = `0b00` (Continuous Edge Detect with Step Delay at Exit)

In this mode, the selected trigger input is continuously tested starting immediately after the `PTGWHI` or `PTGWLO` command is executed. When the trigger edge is detected, the command execution completes.

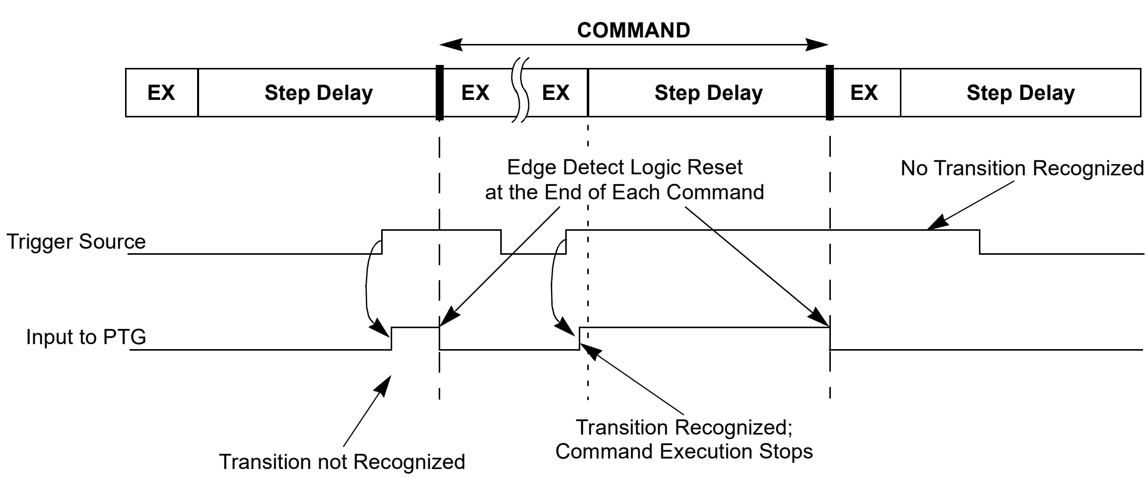
If the step delay counter is enabled, the step delay will be inserted (once) after the valid edge is detected and after the command execution.

If the step delay counter is not enabled, the command will complete after the valid edge is detected and execution of the subsequent command will commence immediately.

Note: The edge detect logic is reset after the command execution is complete (i.e., prior to any step delay associated with the command). For the edge to be detected, the edge should occur during the `PTGWHI` or `PTGWLO` command execution.

Figure 30-3 shows an example timing diagram of Mode 0 operation.

Figure 30-3. Operation of Edge-Sensitive Command with Exit Step Delay



30.4.6.2.2. Mode 1: PTGITM[1:0] = `0b01` (Continuous Edge Detect without Step Delay at Exit)

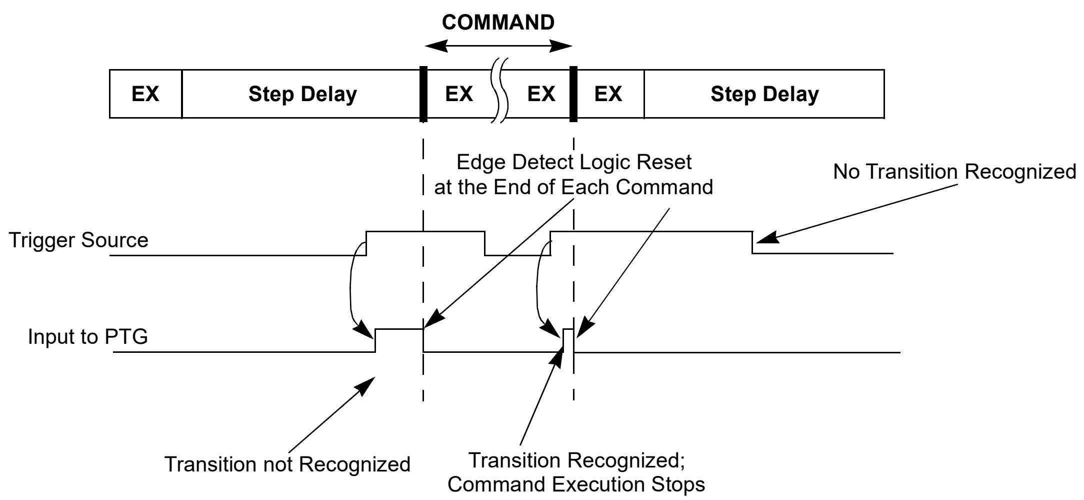
In this mode, the selected trigger input is continuously tested starting immediately after the `PTGWHI` or `PTGWLO` command is executed. When the trigger edge is detected, the command execution completes.

Regardless of whether the step delay counter is enabled or disabled, the step delay will not be inserted after the command execution has completed.

Note: The edge detect logic is reset after the command execution completes. To be detected, the edge may therefore occur during the `PTGWHI` or `PTGWLO` command execution.

Figure 30-4 shows an example timing diagram of Mode 1 operation.

Figure 30-4. Operation of Edge-Sensitive Command without Exit Step Delay



30.4.6.2.3. Mode 2: PTGITM[1:0] = `0b10` (Sampled Level Detect with Step Delay at Exit)

In this mode, the selected trigger input is sample tested for a valid level immediately after the `PTGWHI` or `PTGWLO` command is executed; the trigger input is tested (once per PTG clock).

If the trigger does not occur, and the step delay is enabled, the command waits for the step delay to expire before testing the trigger input again. When the trigger occurs, the command execution completes and the step delay is reinserted.

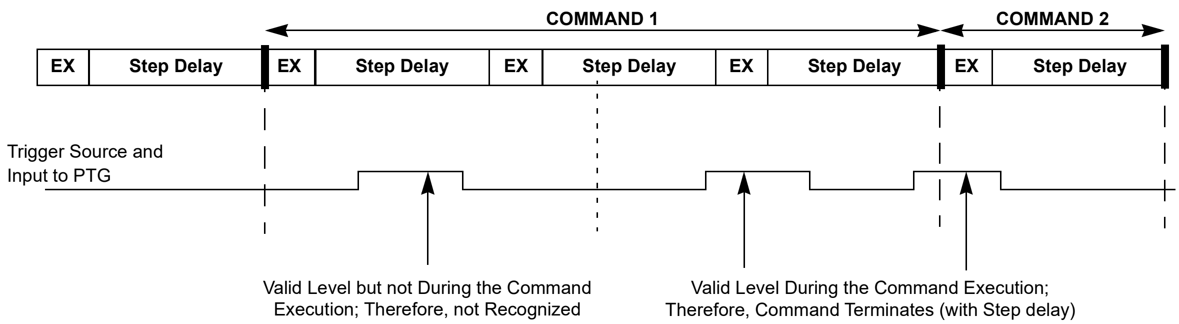
If the trigger does not occur and the step delay is disabled, the command immediately tests the trigger input again during the next PTG clock cycle. When the trigger occurs, the command execution completes and execution of the subsequent command will commence immediately.

Notes:

1. As this Operating mode is level-sensitive, if the input trigger level is true at the start of execution of the `PTGWHI` or `PTGWLO` command, the input test will be instantly satisfied.
2. The input is not latched, therefore, it must be valid when the command executes in order to be recognized.

Figure 30-5 shows an example timing diagram of Mode 2 operation.

Figure 30-5. Operation of Level-Sensitive Command with Exit Step Delay



30.4.6.2.4. Mode 3: PTGITM[1:0] = `0b11` (Sampled Level Detect without Step Delay at Exit)

In this mode, the selected trigger input is sample tested for a valid level immediately after the `PTGWHI` or `PTGWLO` command is executed; the trigger input is tested (once per PTG clock).

If the trigger does not occur and the step delay is enabled, the command waits for the step delay to expire before testing the trigger input again. When the trigger is found to be true, the command execution completes and execution of the subsequent command will commence immediately. The step delay is not inserted.

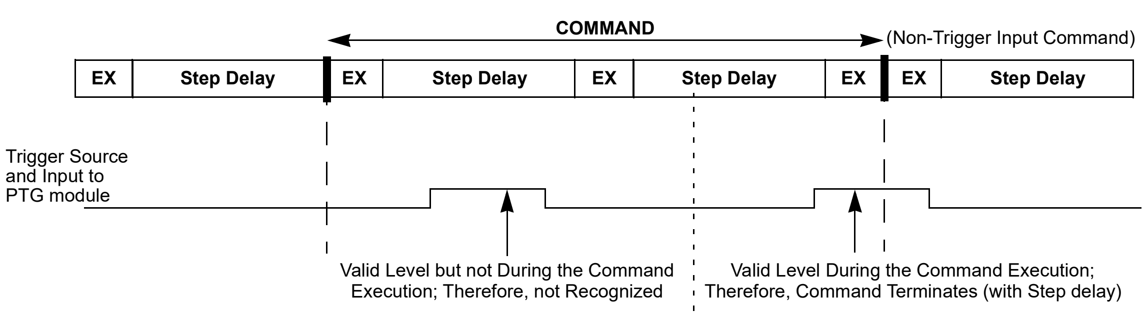
If the trigger does not occur and the step delay is disabled, the command immediately tests the trigger input again during the next PTG clock cycle. When the trigger occurs, the command execution completes and execution of the subsequent command will commence immediately.

Notes:

1. As this Operating mode is level-sensitive, if the input trigger level is true at the start of execution of the `PTGWHI` or `PTGWLO` command, the input test will be instantly satisfied.
2. The input is not latched, therefore, it must be valid when the command executes in order to be recognized.

Figure 30-6 shows an example timing diagram of Mode 3 operation.

Figure 30-6. Operation of Level-Sensitive Command without Exit Step Delay



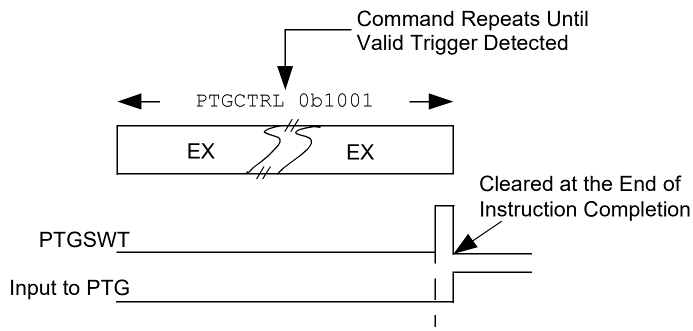
30.4.6.3. Wait for Software Trigger

The user can set either a '`PTGCTRL 0b1011`' (edge-triggered) or '`PTGCTRL 0b1010`' (level-triggered) command to wait for a software generated trigger. This trigger is generated by setting the PTGSWT bit (PTGCON[10]).

The 'PTGCTRL 0b1011' command is sensitive only to the PTGSWT bit transition from '0' to '1'. This transition must occur during the command execution; otherwise, the command will continue to wait. The PTGSWT bit is automatically cleared by hardware on completion of the 'PTGCTRL 0b1011' command execution, initializing the bit for the next software trigger command.

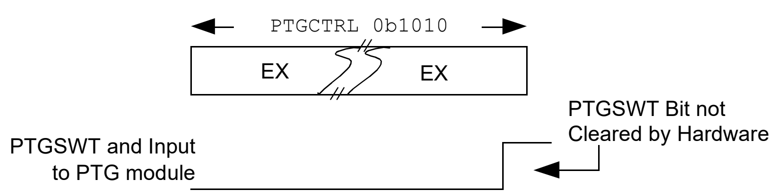
Figure 30-7 explains the operation of the wait for an edge-based software trigger.

Figure 30-7. Operation of Wait for Edge-Based Software Trigger



The 'PTGCTRL 0b1010' command is sensitive to the level of the PTGSWT bit. This command waits until PTGSWT = 1. It will complete immediately if PTGSWT = 1 upon entry to the command. The PTGSWT bit is not automatically cleared by the 'PTGCTRL 0b1010' command. If desired, the PTGSWT bit can be cleared by the user application on completion of the 'PTGCTRL 0b1010' command execution. Figure 30-8 explains the operation of the wait for the level-based software trigger.

Figure 30-8. Operation of Wait for Level-Based Software Trigger



Using the 'PTGCTRL 0b1010' or 'PTGCTRL 0b1011' step commands halts execution of further commands until the PTGSWT bit is set, allowing the user to coordinate activity between the PTG module and the application software.

30.4.6.4. Wait for GP Timer

The PTG has two internal dedicated 16-bit General Purpose (GP) timers (PTGT0 and PTGT1), which can be used by the sequencer to wait for a specified period. The step commands are available for loading, modifying or initializing the GP timers.

Each GP timer consists of an incrementing timer (PTGTx) and a Limit register (PTGTxLIM). The Limit register value can be changed by a CPU write (when the module is disabled) or by the PTG sequencer (when the module is enabled). Data read from the Limit register depend upon the state of the PTG Counter/Timer Visibility bit (PTGIVIS).

When running, the timers increment on the rising edge of the PTG clock, which is defined in the PTGCON register. The user can specify a wait operation using a GP timer by executing the appropriate 'PTGCTRL 0b1000' or 'PTGCTRL 0b1001' command (wait for selected GP timer).

When waiting for the GP timer, the command will wait until the value of the timer (Timer0 or Timer1) reaches its respective limit value (PTGT0LIM or PTGT1LIM). On reaching the limit value, the step command execution completes and the next command will start. The timer is also cleared for its next use. All timers are cleared when the device is in the Reset state or when the PTG module is disabled (ON = `0`).

30.4.6.5. Step Command Delay

The Step Delay Timer (SDLY) is a convenient method to make each step command consume a specific amount of time. Normally, the user specifies a step delay equal to the duration of a peripheral function, such as the ADC conversion time. The step delay enables the user to generate the trigger output signals at a controlled rate, thereby avoiding overload on the target peripheral.

The PTGSDLIM register defines the additional time duration of each step command in terms of PTG clocks.

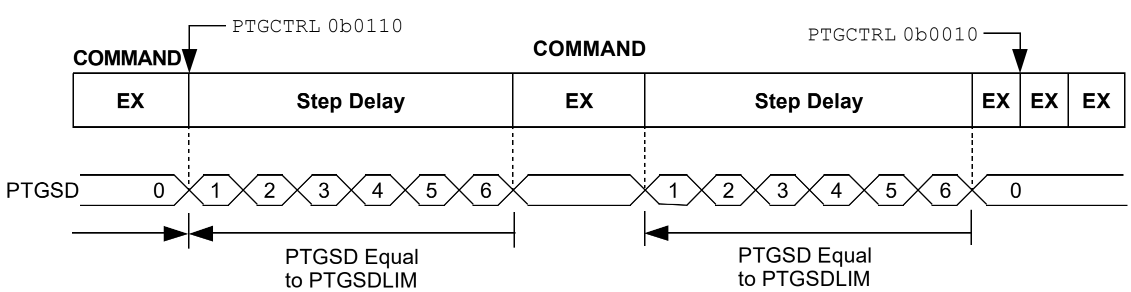
By default, the SDLY is disabled. The user can enable and disable the SDLY via the '`PTGCTRL 0b0110`' or '`PTGCTRL 0b0010`' command, which can be placed in the step queue.

When operating, the SDLY increments at the PTG clock rate. The PTGSDLIM register value is referred to as the step delay timer limit value. The step delay is inserted after each command is executed, so that all step commands are stalled until the PTGSD timer reaches its limit value. On reaching the limit value, the command execution completes and the next command starts execution. The timer is also cleared during execution of each command, so that it is ready for the next command execution.

Note: The PTGSDLIM register value of '`0x0000`' does not insert the additional PTG clocks when the step delay timer is enabled. The PTGSDLIM register value of '`0x0001`' inserts a single PTG step delay (one PTG clock) into every subsequent instruction after the step delay timer is enabled.

The trigger sources for the edge-sensitive commands ('`PTGCTRL 0b1011`' and `PTGWHI`/`PTGWLO` when operated in Edge-Sensitive mode) have additional logic to recognize the appropriate edge transition. The edge detection logic is reset at the end of each command to maintain the edge-sensitive nature of these input triggers. If an additional valid edge occurs during a step delay that has been inserted after the step command has executed, it will not be recognized by any subsequent step command. Figure 30-9 explains the operation of the step command delay. As shown, step delay is inserted immediately following the PTGCTRL 0b0110 command which enables step delay, and is not inserted after the PTGCTRL 0b0010 command which disables step delay; the change is immediate.

Figure 30-9. Operation of Step Command Delay



30.4.7. PTG Watchdog Timer

In some applications, a Watchdog Timer (WDT) may be required as the PTG can wait indefinitely for an external event when executing the following commands:

- Wait for hardware trigger positive edge or High state (`PTGWHI`).
- Wait for hardware trigger negative edge or Low state (`PTGWLO`).

The WDT is enabled, and it starts counting when the command starts to execute. It is disabled when the command completes execution and prior to any step delay insertion. All other commands execute with the predefined cycle counts.

Note: The PTG Watchdog Timer is not enabled during execution of the '`PTGCTRL 0b1011`' or '`PTGCTRL 0b1010`' command. It is assumed that correct operation of the device will be monitored through other means.

30.4.7.1. Operation Overview

If an expected event fails to arrive before the WDT time-out period expires, the PTG module:

1. Aborts the (failing) command underway.
2. Halts the sequencer (PTGSTRT = `0`).
3. Sets PTGWDTO = `1`.
4. Issues a Watchdog Timer error interrupt to the processor.

The user can either use the Watchdog Timer error interrupt or periodically poll the PTGWDTO bit (PTGCON[6]) to determine that a WDT event has occurred.

30.4.7.2. Configuration

The WDT is configured by setting the PTGWDT[2:0] bits (PTGCON[18:16]). The WDT counts the PTG clocks as defined by the PTG input clock selection and the PTGDIV[4:0] bits in the PTGCON register. For more information, refer to PTG Clock Selection. The WDT time-out count value is selected by using the PTGWDT[2:0] bits and is disabled when PTGWDT[2:0] = `0b000`.

Notes:

1. The WDT is disabled prior to insertion of any step delay; therefore, the user does not need to account for the Step delay when calculating a suitable WDT time-out value.
2. Some bits within the PTGCON register are read-only when PTGSTRT = `1` (sequencer executing commands). Refer to PTGCON.

30.4.7.3. Watchdog Timer Event Recovery

If a WDT event occurs, the user can take necessary action to identify and fix the problem and then continue the step command sequence, or the user can simply restart the sequence.

To clear the PTGWDTO bit and to restart the PTG sequencer from the start of the step queue, disable (ON = `0`) and re-enable (ON = `1`) the PTG module and then restart execution (PTGSTRT = `1`).

Alternatively, as the sequencer is only halted (not reset), the user has the option to examine the PTGQPTR register to identify which step command was the source of the problem and then can take a corrective action. The offending command is aborted prior to the PTGQPTR update. Therefore, it will still address the failing command after the WDT event. After the PTGWDTO bit is cleared, the step queue can be restarted at the same command by setting PTGSTRT = `1`.

Note: The user should clear the PTGWDTO bit after a WDT event. Failing to clear the bit will not interfere with the subsequent module operation, but it will not be possible for the bit to poll any future WDT events.

30.4.8. PTG Module Outputs

The PTG module can generate trigger, interrupt and strobed data outputs by execution of specific step commands.

30.4.8.1. Trigger Outputs

The PTG module can generate up to 32 unique trigger output signals. There are two types of trigger output functions:

- Individual
- Broadcast

The PTG module can generate an individual output trigger on any one of the trigger outputs using the `PTGTRIG` command.

The individual trigger outputs may be assigned to various functions, such as ADC conversion triggers or PPS input/output signals. PPS signals can be routed to external pins or used as inputs to certain other peripherals; refer to the **Peripheral Pin Select (PPS) Overview** chapter for more information.

The broadcast trigger output feature is specified by the PTGBTE register. Each bit in the register corresponds to an associated individual trigger output. If a bit is set in the PTGBTE register, and a broadcast trigger step command ('`PTGCTRL 0b1111`') is executed, the corresponding individual trigger output is asserted. The broadcast trigger output enables the user to simultaneously generate a large number of trigger outputs with a single step command.

30.4.8.2. Interrupt Outputs

The PTG module can generate up to 16 unique interrupt request signals. These signals are useful for interacting with an application software to create more complex functions.

The PTG module can generate an individual interrupt pulse by using the `PTGIRQ` command.

30.4.8.3. Strobe Output

The strobe output of the PTG module can be used to output data from the PTG module. Typically, this output is connected to the ADC Channel Selection register, allowing the PTG to loop through ADC channels. The device-specific data sheet will indicate how the PTG strobe output is connected to other peripherals.

The '`PTGCTRL 0b1110`' command writes the PTGL0 register contents to the strobe output. The PTGL0 register can be modified by using the `PTGADD` and `PTGCOPY` commands.

The '`PTGCTRL 0b1100`' command writes the PTGC0LIM register contents to the strobe output. The '`PTGCTRL 0b1101`' command writes the PTGC1LIM register contents to the strobe output.

30.4.8.4. Output Timing

All triggers, interrupts and data strobe outputs are internally asserted by the PTG state machine when the corresponding step command execution starts (i.e., before any additional time specified by the step delay timer) on the rising edge of the PTG execution clock.

Note: The trigger and strobe output pulse generators operate independently of the PTG sequencer and will not stall command execution to allow a pulse to complete. If a trigger output is requested by a step command before a previous trigger pulse has completed, the previous pulse will be terminated and the new trigger output will commence as normal. If a strobed data command is executed before one PTG clock cycle after a previous data strobe has completed, the behavior is undefined.

In Pulse mode (PTGTOGL = `0`), the width of the trigger output signals is determined by the PTGPWD[3:0] bits (PTGCON[23:20]) and can be any value between one and 16 PTG clock cycles. The default value is one PTG clock cycle.

Refer to TRIG Negation When PTGTOGL = 1 when operating in Toggle mode (PTGTOGL = `1`).

When globally controlled by the '`PTGCTRL 0b1111`' broadcast trigger command, the TRIG output pulse width is determined by the PTGPWD[3:0] bits (PTGCON[23:20]) and can be any value between one and 16 PTG clock cycles. The default value is one PTG clock cycle.

Note: The trigger generated by using the '`PTGCTRL 0b1111`' broadcast trigger command can only operate in Pulse mode (i.e., PTGTOGL = don't care).

30.4.8.4.1. TRIG Negation When PTGTOGL = 0

If generating an individual trigger output when the PTGTOGL bit (PTGCON[12]) = `0`, or if generating a broadcast trigger output, the TRIG output(s) pulse width is determined by the PTGPWD[3:0] bits.

30.4.8.4.2. TRIG Negation When PTGTOGL = 1

If generating an individual trigger output when the PTGTOGL bit (PTGCON[12]) = `1`, the TRIG outputs will remain set until the `PTGTRIG` command is executed again. On the start of the `PTGTRIG` command execution, the TRIG outputs are toggled at the beginning of the PTG execution clock.

Note: The PTGTOGL bit has no effect on the operation of the '`PTGCTRL 0b1111`' multiple trigger (broadcast) generation command with the following exception:

- If a target trigger output is already in the Logic '`1`' state, having been triggered individually while PTGTOGL is active, the '`PTGCTRL 0b1111`' command will have no effect and the trigger output will remain at logic '`1`'.

30.4.9. Stopping the Sequencer

When the PTG module is disabled (ON = `0`), the PTG clocks are disabled (except the trigger pulse counter), the sequencer stops execution and the module enters its lowest power state. The PTGSTRT, PTGSWT, PTGWDTO and PTGQPTR[4:0] bits are all reset. All other bits and registers are not modified. All of the control registers can be read or written when ON = `0`.

When the ON bit is cleared, a command that is underway is immediately aborted if the command is waiting for any of the following actions:

- An input from another source
- A timer match
- The step delay to expire (for more information, refer to Step Command Delay)

All other commands are allowed to complete before the PTG module is disabled.

When the PTG module is halted, all of the control registers remain in their present state. The PTG module can be halted by the user by clearing the PTGSTRT bit, or in the event of a Watchdog Timer time-out, which also clears the PTGSTRT bit. Refer to PTG Watchdog Timer for more information.

Note: If a command has triggered the pulse-width delay counter, the counter is synchronously reset with respect to the PTG clock, terminating the pulse (subject to a minimum pulse width of 1 PTG clock cycle).

30.4.10. Single-Step Mode

For debugging purposes, the PTG can be configured to run in Single-Step mode. In this mode, step commands are not executed continuously. Instead, one command is executed at a time, and the PTG halts execution after each command. This allows the user to step through the step queue, similarly to stepping through CPU instructions while debugging.

30.4.10.1. Entering Single-Step Mode

Single-Step mode is entered by setting the PTGSSEN bit (PTGCON[9]) = `1`. This bit may only be accessed from a debugging session and will read as '`0`' at all other times. For example, the following steps can be used to set PTGSSEN:

1. In MPLAB® X IDE, program the device to run in Debug mode.
2. Pause code execution at any point where PTGSTRT has not been set to '`1`'.
3. Add a variable watch for the PTGCON register.
4. Using the watch, set the PTGSSEN bit to '`1`'.

Note: The value of PTGSTRT must be '`0`' when setting the PTGSSEN bit or Single-Step mode will not be entered.

Note: Due to the Debug mode restriction, it is not possible to read PTGSSEN from application code to determine whether Single-Step mode has been entered as it will always read as '`0`'.

30.4.10.2. Executing a Single-Step Command

Once the PTG is in Single-Step mode, setting PTGSTRT = `1` will cause the PTG to execute the command pointed to by PTGQPTR and then halt. Once this step command has been executed, PTGSTRT is cleared by hardware and can be set again to execute another command. PTGQPTR will then point to the next command to execute; however, its value can be changed while PTGSTRT = `0`, allowing single-stepping to take place at any queue location.

Note: PTGSTRT can be cleared manually during single-step execution, halting the sequencer. This is only effective when the currently executing command is waiting for input from an external source or a timer match, as all other commands will be allowed to complete.

30.4.10.3. PTG Step Interrupt

After each step command is executed in Single-Step mode, the PTG step interrupt will occur, if enabled. This notifies the application of the completion of the command.

Example 30-4. Single-Step Demonstration Program

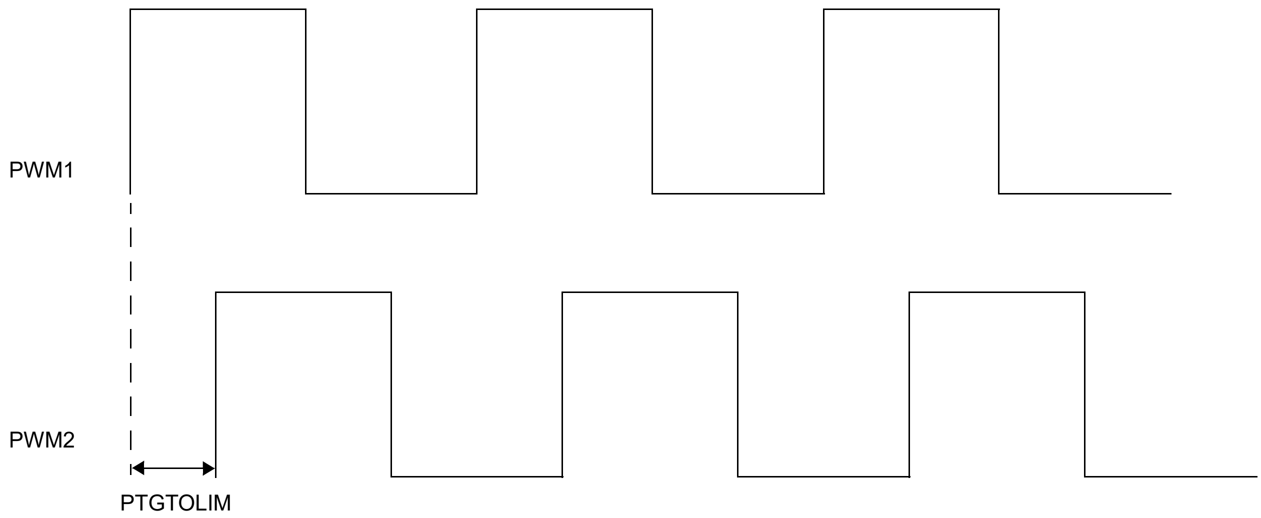
```
//This code shows a basic example of using PTG Single-Step Mode.
//Program must be run in Debug Mode.
//Once the PTGSSEN bit is set, press the push button (RF1) to set PTGSTRT to execute a
single command.
//An LED (RC8) will toggle on the completion of each command.
#include <xc.h>

void PTG_populate_queue() {
    //Set up commands in the PTG Step Queue
    PTGQUE0bits.STEP0 = 0;            //NOP
    PTGQUE0bits.STEP1 = 0b10100000;     //PTGJMP(0)
}

int main (void) {
    _ANSELC1 = 0;
    _TRISC1 = 1;                   //RC1 input connected to push button switch
    _Bool sw_latch = 0;                //Latch button input so one press makes one step

    _TRISC8 = 0;                   //RC8 output connected to external indicator
    _LATC8 = 0;                    //RC8 output low initially

    PTG_populate_queue();              //Set up step commands in PTG queue
    PTGCONbits.ON = 1;               //Place breakpoint on this line. When execution halts,
    //set PTGSSEN = 1 using the debugger, then continue.

    _PTGSTEPIE = 1; //Enable PTG Step Interrupt
    while(1) {
        if (!_RC1) {               //Check if button is pressed (active low)
            if (!sw_latch) {
                _PTGSTRT = 1;         //Execute a single Step command
                sw_latch = 1;
            }
        }
        else {
            sw_latch = 0;           //Reset latch for next button press
        }
    }

    return 0;
}

void __attribute__ ((__interrupt__)) _PTGSTEPInterrupt(void) {
    _LATC8 = !_LATC8;              //Toggle RC8 to show step has taken place
    _PTGSTEPIF = 0;
}
```

30.5. Application Examples

30.5.1. Generating Phase-Shifted Waveforms

Figure 30-10 shows an application example for generating phase-shifted PWM waveforms. In this example, PWM1 generates a waveform and the rising edge of this pulse is the trigger input to the PTG module. When the trigger from PWM1 is received, the PTG module waits on PTG Timer 0 using the PTGCTRL(0b1000) command, inserting a programmable delay and then outputting a PCI signal to PWM2. This signal is the synchronization source for PWM2, which is configured to output a single cycle with the same period and duty cycle as PWM1. As PWM2 is repeatedly triggered by the PTG, it outputs a phase-shifted version of PWM1, with the phase shift determined by the PTG Timer 0 delay.

Figure 30-10. Phase-Shifted Waveform Example Application



Example 30-5 shows code for generating a phase-shifted waveform.

Example 30-5. Generating Phase-Shifted Waveforms

```
#include <xc.h>

void IO_initialize() {
    _PCI12R = 137;                 //Connect PCI 12 to PTG trigger 26
    _TRISD2 = 0;                   //Set RD2 as output (used by PWM1H)
    _TRISD0 = 0;                   //Set RD0 as output (used by PWM2H)
}

void PWM1_initialize() {

    PG1CONbits.CLKSEL = 1;         //Main PWM clock (undivided/unscaled) used
for PWM2
    PG1IOCON1bits.PENH = 1;         //PWM generator 1 controls PWM1H pin
    PG1EVT2bits.ADTR2EN1 = 1;       //PGA1TRIGA match controls PWM1 ADC
Trigger 2

    PG1PER = 8000 << 4;            //Period 1ms for a 8MHz PWM clock
    PG1DC = 4000 << 4;             //50% duty cycle
    PG1PHASE = 0;                  //0 phase offset
    PG1TRIGA = 0;                  //PWM ADC trigger 2 will happen at start
of cycle

    PG1CONbits.ON = 1;             //Enable PWM Generator 1
}

void PWM2_initialize() {

    PG2CONbits.CLKSEL = 1;         //Main PWM clock (undivided/unscaled) used
for PWM2
    PG2CONbits.SOCS = 0b1111;      //PCI sync used for start of cycle
    PG2IOCON1bits.PENH = 1;         //PWM generator 2 controls PWM2H pin

    PG2PER = 8000 << 4;            //Period 1ms for a 8MHz PWM clock
    PG2DC = 4000 << 4;             //50% duty cycle
    PG2PHASE = 0;                  //0 phase offset

    PG2SPCI2 = 0x1100;             //PCI12 as PWM2 sync source
    PG2CONbits.ON = 1;             //Enable PWM Generator 2
}
void PTG_initialize() {

    PTGT0LIM = 1000; //0.125ms T0 delay

    PTGQUE0bits.STEP0 = PTGWHI(0);         //Wait for high-to-low edge on
trigger from PWM1
    PTGQUE0bits.STEP1 = PTGCTRL(t0Wait);   //Wait on T0
    PTGQUE0bits.STEP2 = PTGTRIG(26);       //Trigger PWM2
    PTGQUE0bits.STEP3 = PTGJMP(0);         //Restart sequence

    PTGCONbits.ON = 1;             //Enable PTG
    PTGCONbits.PTGSTRT = 1;        //Start executing commands
}

void clocks_initialize() {

    //Configure both CLKGEN5 (PWM) and CLKGEN11 (PTG) to use 8MHz clock from
FRC.

    //Enable CLKGEN5, if not already enabled
    CLK5CONbits.ON = 1;

    //Reset CLKGEN5 fractional divider for 1:1 ratio
    CLK5DIVbits.INTDIV = 0;
    CLK5DIVbits.FRACDIV = 0;
    //Request CLKGEN5 fractional divider switch
    CLK5CONbits.DIVSWEN = 1;
    //Wait for CLKGEN5 fractional divider switch to complete
    while(CLK5CONbits.DIVSWEN);

    //Select FRC as CLKGEN5's new clock source
    CLK5CONbits.NOSC = 1;
    //Request CLKGEN5 clock source switch
```

```
    CLK5CONbits.OSWEN = 1;
    //Wait for CLKGEN5 clock source switch to complete
    while(CLK5CONbits.OSWEN);

    PCLKCONbits.MCLKSEL = 1; //Use CLKGEN5 to provide PWM MCLK

    //Enable CLKGEN11, if not already enabled
    CLK11CONbits.ON = 1;

    //Reset CLKGEN11 fractional divider for 1:1 ratio
    CLK11DIVbits.INTDIV = 0;
    CLK11DIVbits.FRACDIV = 0;
    //Request CLKGEN11 fractional divider switch
    CLK11CONbits.DIVSWEN = 1;
    //Wait for CLKGEN11 fractional divider switch to complete
    while(CLK11CONbits.DIVSWEN);

    //Select FRC as CLKGEN11's new clock source
    CLK11CONbits.NOSC = 1;
    //Request CLKGEN11 clock source switch
    CLK11CONbits.OSWEN = 1;
    //Wait for CLKGEN11 clock source switch to complete
    while(CLK11CONbits.OSWEN);

}

int main(void) {

    clocks_initialize();
    IO_initialize();
    PWM1_initialize();
    PWM2_initialize();
    PTG_initialize();

    while (1);

    return 0;
}
```

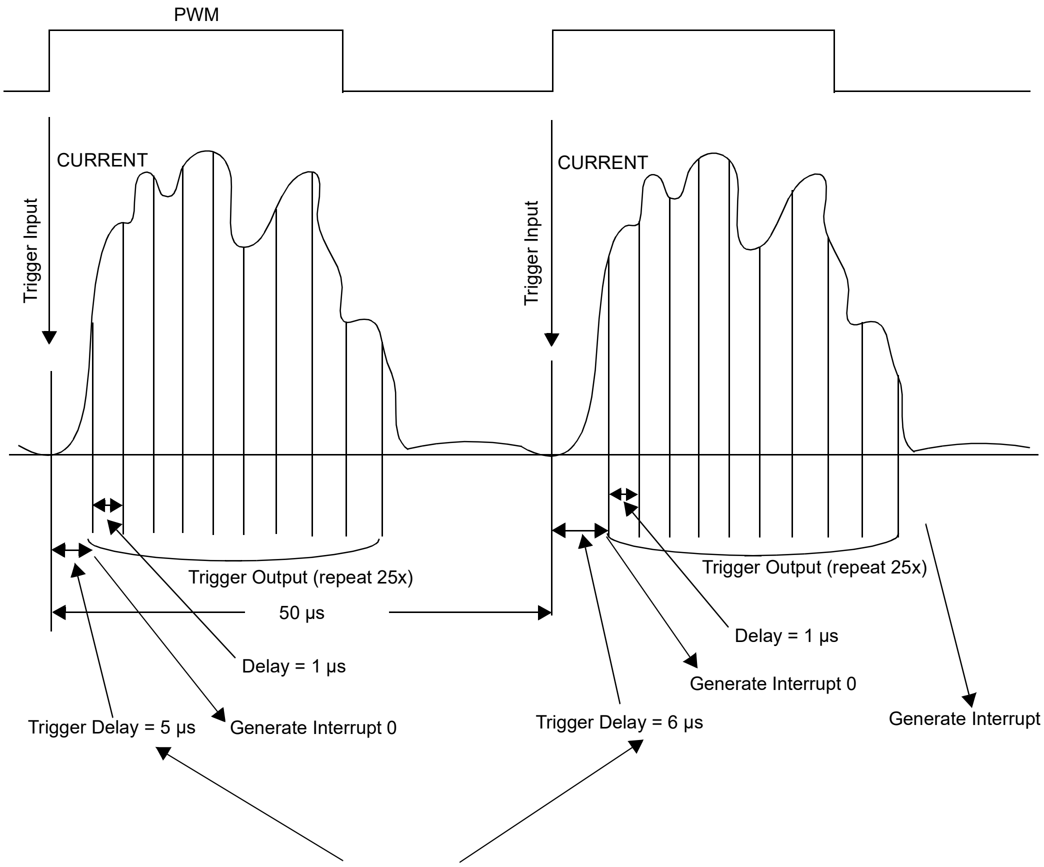
30.5.2. Interleaving Samples Over Multiple Cycles

Figure 30-11 shows the waveforms of an application where the power in a system needs to be accurately measured when the current load is highly dependent on the temperature, voltage and user application. The current waveforms vary widely per usage, but over a few PWM cycles, the waveforms are relatively stable.

The goal of this example is to collect many current and voltage readings over several PWM cycles in an interleaved manner. The data are stored in the memory during acquisition and later processed (integrated) to yield an accurate power value.

This example shows a situation where it would not be practical or possible for software to accurately schedule the ADC samples.

Figure 30-11. Example Application – Average Power Calculation



Note: The trigger delay value is modified to make the subsequent sample triggers shift in time, thereby enabling the interleaving of the samples.

30.5.2.1. Interleaved Sampling Step Command Program

This section describes the step command programming for implementing the timing sequence shown in Figure 30-11.

The following assumptions are made:

1. Trigger Input 0 is connected to the PWM1 signal. The rising edge of this signal starts the sequence. PWM1 has a period of 50 µs.
2. Output Trigger 12 is connected to the ADC module. This signal commands the ADC module to begin a sample and conversion process.
3. Interrupt 0 is used to indicate to the processor that a subsequence has started (provides status).
4. Interrupt 1 is used to indicate to the processor that the entire sequence has been completed.
5. The PTG clock source is 125 MHz.
6. The initial trigger delay is 5 µs.
7. The second trigger delay is 6 µs.
8. In each PWM cycle, the ADC is triggered 25 times.
9. The basic sequence is executed twice.

Example 30-6. Interleaved Sampling Step Command Program

```
#include <xc.h>

//Allocate buffers for ADC results.
//Size of 50 is based on 2 PWM cycles, 25 results per cycle.
#define RESULT_BUFFER_SIZE 50
unsigned int voltage_buffer[RESULT_BUFFER_SIZE];
int voltage_buffer_index = 0;
unsigned int current_buffer[RESULT_BUFFER_SIZE];
int current_buffer_index = 0;
_Bool readings_ready = 0;

void clocks_initialize() {
    //Set up CPU clock (Fcy) to run at 200 MHz.

    //Start by configuring PLL1

    PLL1CONbits.ON = 1; //Enable PLL1, if not already enabled

    //PLL1 has input frequency 8MHz (FRC)
    PLL1DIVbits.PLLPRE = 1;         //Reference input will be 8MHz, no division
    PLL1DIVbits.PLLFBDIV = 125;     //Fvco = 8MHz * 125 = 1000MHz
    PLL1DIVbits.POSTDIV1 = 5;       //Divide Fcvo by 5
    PLL1DIVbits.POSTDIV2 = 1;       //Fpllo = Fvco / 5 / 1 = 200 MHz

    //The PLLSWEN bit controls changes to the PLL feedback divider.
    //Request PLL1 feedback divider switch
    PLL1CONbits.PLLSWEN = 1;
    //Wait for PLL1 feedback divider switch to complete
    while(PLL1CONbits.PLLSWEN);

    //The FOUTSWEN bit controls changes to the PLL output dividers.
    //Request PLL1 output divider switch
    PLL1CONbits.FOUTSWEN = 1;
    //Wait for PLL1 output divider switch to complete
    while(PLL1CONbits.FOUTSWEN);

    VCO1DIVbits.INTDIV = 2; //Divide Fvco by 4
    //The DIVSWEN bit controls changes to the VCO divider.
    //Request PLL1 VCO divider switch
    PLL1CONbits.DIVSWEN = 1;
    //Wait for PLL1 VCO divider switch to complete
    while(PLL1CONbits.DIVSWEN);

    //Reset CLKGEN1 dividers for 1:1 ratio
    CLK1DIVbits.INTDIV = 0;
    CLK1DIVbits.FRACDIV = 0;
    //Request CLKGEN1 divider switch
    CLK1CONbits.DIVSWEN = 1;
    //Wait for divider switch to complete
    while(CLK1CONbits.DIVSWEN);

    CLK1CONbits.NOSC = 5;       //Set PLL1 Fout as CPU clock source
    CLK1CONbits.OSWEN = 1;      //Request clock switch
    while (CLK1CONbits.OSWEN);  //Wait for switch to complete


    //Configure CLKGEN5 to provide a 125MHz PWM MCLK

    CLK5CONbits.ON = 1; //Enable CLKGEN5, if not already enabled

    CLK5DIVbits.INTDIV = 1; //Divide input clock by 2
    CLK5DIVbits.FRACDIV = 0;
    //Request CLKGEN5 divider switch
    CLK5CONbits.DIVSWEN = 1;
    //Wait for CLKGEN5 divider switch to complete
    while(CLK5CONbits.DIVSWEN);

    CLK5CONbits.NOSC = 7;       //Use divided PLL1 VCO (250MHz)
    CLK5CONbits.OSWEN = 1;      //Request clock switch
    while(CLK5CONbits.OSWEN);   //Wait for switch to complete

    PCLKCONbits.MCLKSEL = 1;    //Use CLKGEN5 as PWM MCLK source
```

```
    //Configure CLKGEN6 to provide a 250MHz input clock to the ADC

    CLK6CONbits.ON = 1; //Enable CLKGEN6, if not already enabled

    //Reset CLKGEN6 dividers for 1:1 ratio
    CLK6DIVbits.INTDIV = 0;
    CLK6DIVbits.FRACDIV = 0;
    //Request CLKGEN6 divider switch
    CLK6CONbits.DIVSWEN = 1;
    //Wait for CLKGEN6 divider switch to complete
    while(CLK6CONbits.DIVSWEN);

    CLK6CONbits.NOSC = 7;       //Use divided PLL1 VCO (250MHz)
    CLK6CONbits.OSWEN = 1;      //Request clock switch
    while (CLK6CONbits.OSWEN);  //Wait for switch to complete


    //Configure CLKGEN11 to provide a 125MHz clock for the PTG.

    CLK11CONbits.ON = 1; //Enable CLKGEN10, if not already enabled

    CLK11DIVbits.INTDIV = 1; //Divide input clock by 2
    CLK11DIVbits.FRACDIV = 0;
    //Request CLKGEN11 divider switch
    CLK11CONbits.DIVSWEN = 1;
    //Wait for CLKGEN11 divider switch to complete
    while(CLK11CONbits.DIVSWEN);

    CLK11CONbits.NOSC = 7;      //Use divided PLL1 VCO (125MHz)
    CLK11CONbits.OSWEN = 1;     //Request clock switch
    while (CLK11CONbits.OSWEN); //Wait for switch to complete
}

void PTG_initialize() {

    //Enable PTG interrupts 0 and 1
    _PTG0IE = 1;
    _PTG1IE = 1;

    //Set up control registers
    PTGT0LIM = 625;     //5us T0 delay
    PTGT1LIM = 125;     //1us T1 delay
    PTGC0LIM = 24;      //Repeat C0 loop 25 times
    PTGC1LIM = 1;       //Repeat C1 loop once
    PTGHOLD = 625;      //5us (used to restore T0 delay)
    PTGADJ = 125;       //1us (added to T0 delay)
    PTGQPTR = 0;        //Initialize step queue pointer

    //Outer loop
    PTGQUE0bits.STEP0 = PTGWHI(0);       // Wait for positive edge trigger 0
(PWM1 ADC Trigger 2)
    PTGQUE0bits.STEP1 = PTGCTRL(t0Wait); // Start PTGT0, wait for time out
    PTGQUE0bits.STEP2 = PTGIRQ(0);       // Generate IRQ 0
    // Inner loop
    PTGQUE0bits.STEP3 = PTGTRIG(12);     // Generate output trigger 12 (ADC
conversion)
    PTGQUE1bits.STEP4 = PTGCTRL(t1Wait); // Start PTGT1, wait for time out
    PTGQUE1bits.STEP5 = PTGJMPC0(3); // Go to STEP3 if PTGC0 != PTGC0LIM,
increment PTGC0 (ie. repeat steps 3-5 24 times)
    // End inner loop
    PTGQUE1bits.STEP6 = PTGADD(t0Limit); // Add PTGADJ to PTGT0LIM
    PTGQUE1bits.STEP7 = PTGJMPC1(0);     // Jump to 0 PTGC1LIM times (once,
making 2 iterations)
    // End outer loop

    PTGQUE2bits.STEP8 = PTGIRQ(1);      // Generate IRQ 1
    PTGQUE2bits.STEP9 = PTGCOPY(t0Limit);
```

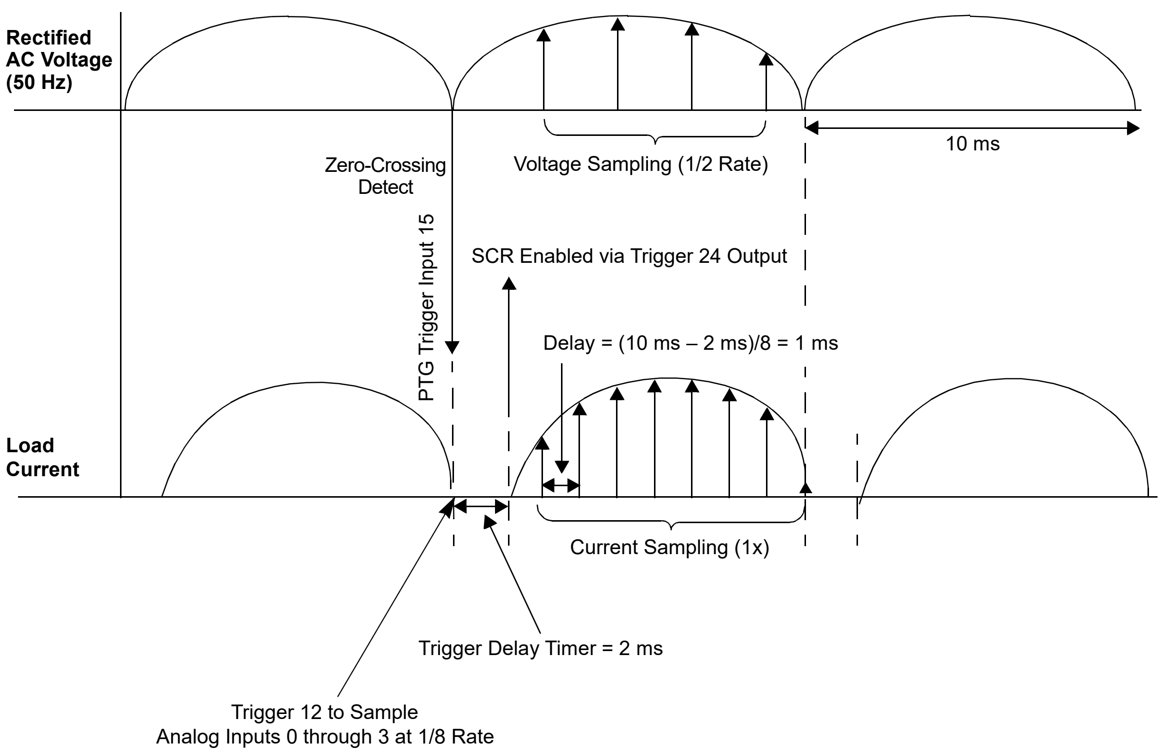
See Example 30-1 for PTG command definitions. See Example 30-2 and Example 30-3 for `PTGCTRL`, `PTGADD` and `PTGCOPY` command options.

30.5.3. Sampling at Multiple Rates

Figure 30-12 shows an application example of sampling an ADC input at a fast rate (1x rate), a second analog input at a slower rate (1/2 rate) and four other inputs at a 1/8 rate. The example is a motor control application using a Silicon Controlled Rectifier (SCR) that triggers at a specified time after the AC line zero-crossing.

While this example uses the simple binary sampling ratios, the PTG module can generate a very wide range of sample ratios to meet the requirements of an application. This example demonstrates coordination between the PTG, ADC, PWM and PPS in order to achieve the required sampling rates.

Figure 30-12. Example Application – Ratioed Sampling



30.5.3.1. Ratioed Sampling Step Command Program

This section describes the step command programming for implementing the timing sequence shown in Figure 30-12.

The following assumptions are made:

- Trigger Input 15 is connected to the zero-crossing detect. The rising edge of the Zero-Crossing Detect signal starts the sequence.
- Trigger Output 24 enables the SCR in the application circuit.
- The trigger delay from Trigger Input 15 to the generation of Trigger Output 24 is 2 ms.
- Trigger Output 26 is connected to PWM1's synchronization signal.
- PWM1 is configured to trigger the ADC to sample at 1/2x rate, and it is configured to trigger PWM2 which will sample the ADC at 1x rate.
- Trigger Output 12 is connected to the ADC to sample other data values (channels 0, 1, 2, an 3) once per cycle.

- The PTG clock is 8 MHz.

Example 30-7. Ratioed Sampling Step Command Program

```
#include <xc.h>

void clocks_initialize() {

    //Configure CLKGEN5 to provide an 8MHz clock for the PWM
    CLK5CONbits.ON = 1;             //Enable CLKGEN5, if not already enabled

    //Reset CLKGEN5 dividers for 1:1 ratio
    CLK5DIVbits.INTDIV = 0;
    CLK5DIVbits.FRACDIV = 0;
    CLK5CONbits.DIVSWEN = 1;
    //Wait for divider switch to complete
    while(CLK5CONbits.DIVSWEN);

    CLK5CONbits.NOSC = 1;           //Select FRC, 8MHz
    CLK5CONbits.OSWEN = 1;          //Request clock switch
    while (CLK5CONbits.OSWEN);      //Wait for switch to complete

    PCLKCONbits.MCLKSEL = 1;        //Use CLKGEN5 for PWM clock


    PLL1CONbits.ON = 1; //Enable PLL generator 1, if not already enabled

    //Set up PLL1
    PLL1DIVbits.PLLPRE = 1;         //Reference input will be 8MHz, no division
    PLL1DIVbits.PLLFBDIV = 125;     //Fvco = 8MHz * 125 = 1000MHz
    PLL1DIVbits.POSTDIV1 = 5;       //Divide Fcvo by 5
    PLL1DIVbits.POSTDIV2 = 1;       //Fpllo = Fvco / 5 / 1 = 200 MHz

    //The PLLSWEN bit controls changes to the PLL feedback divider.
    //Request PLL1 feedback divider switch
    PLL1CONbits.PLLSWEN = 1;
    //Wait for PLL1 feedback divider switch to complete
    while(PLL1CONbits.PLLSWEN);

    //The FOUTSWEN bit controls changes to the PLL output dividers.
    //Request PLL1 output divider switch
    PLL1CONbits.FOUTSWEN = 1;
    //Wait for PLL1 output divider switch to complete
    while(PLL1CONbits.FOUTSWEN);

    VCO1DIVbits.INTDIV = 2;         //Divide Fvco by 4
    //The DIVSWEN bit controls changes to the VCO divider.
    //Request PLL1 VCO divider switch
    PLL1CONbits.DIVSWEN = 1;
    //Wait for PLL1 VCO divider switch to complete
    while(PLL1CONbits.DIVSWEN);

    //Reset CLKGEN1 dividers for 1:1 ratio
    CLK1DIVbits.INTDIV = 0;
    CLK1DIVbits.FRACDIV = 0;
    CLK1CONbits.DIVSWEN = 1;
    //Wait for divider switch to complete
    while(CLK1CONbits.DIVSWEN);

    CLK1CONbits.NOSC = 1;       //Set FRC as CPU clock source
    CLK1CONbits.OSWEN = 1;      //Request clock switch
    while (CLK1CONbits.OSWEN);  //Wait for switch to complete

    //Configure CLKGEN6 to provide a 250MHz input clock to the ADC.

    CLK6CONbits.ON = 1; //Enable CLKGEN6, if not already enabled

    //Reset CLKGEN6 dividers for 1:1 ratio
    CLK6DIVbits.INTDIV = 0;
    CLK6DIVbits.FRACDIV = 0;
    CLK6CONbits.DIVSWEN = 1;
    //Wait for divider switch to complete
    while(CLK6CONbits.DIVSWEN);

    CLK6CONbits.NOSC = 7;       //Set PLL1 VCODIV as ADC clock source
    CLK6CONbits.OSWEN = 1;      //Request clock switch
```

```
    while (CLK6CONbits.OSWEN);  //Wait for switch to complete

    //Configure CLKGEN11 to provide an 8MHz clock for the PTG

    CLK11CONbits.ON = 1; //Enable CLKGEN10, if not already enabled

    //Reset CLKGEN11 dividers for 1:1 ratio
    CLK11DIVbits.INTDIV = 0;
    CLK11DIVbits.FRACDIV = 0;
    //Request CLKGEN11 divider switch
    CLK11CONbits.DIVSWEN = 1;
    //Wait for divider switch to complete
    while(CLK11CONbits.DIVSWEN);

    CLK11CONbits.NOSC = 1;      //Select FRC, 8MHz
    CLK11CONbits.OSWEN = 1;     //Request clock switch
    while (CLK11CONbits.OSWEN); //Wait for switch to complete
}

void PTG_initialize() {

    PTGT0LIM = 16000; // 2 ms T0 delay
    PTGT1LIM = 8000;  // 1 ms T1 delay
    PTGC0LIM = 2;     // 3 iterations of the C0 loop

    PTGQPTR = 0; //Initialize step queue pointer

    //Initialize Step registers
    PTGQUE0bits.STEP0 = PTGWHI(15);        // Wait for trigger from INT2 (zero
crossing detect)
    PTGQUE0bits.STEP1 = PTGTRIG(12);       // Take 1/8 rate samples using ADC
trigger 30
    PTGQUE0bits.STEP2 = PTGCTRL(t0Wait);   // Wait 2ms
    PTGQUE0bits.STEP3 = PTGTRIG(24);       // Trigger PPS output 55 (SCR)
    PTGQUE1bits.STEP4 = PTGCTRL(t1Wait);   // Wait1ms before starting the 1x
and 1/2x triggers
    PTGQUE1bits.STEP5 = PTGTRIG(26);       // Trigger PWM1 to trigger
conversions at 1x and 1/2x rates
    //Start main loop
    PTGQUE1bits.STEP6 = PTGCTRL(t0Wait);   // Wait2ms (pre-trigger delay) for
subsequent triggers
    PTGQUE1bits.STEP7 = PTGTRIG(26);       // Trigger PWM1 to trigger ADC
conversions at 1x and 1/2x rates
    PTGQUE2bits.STEP8 = PTGJMPC0(6);       // Jump to step 6 (twice, for 3
iterations total)
    //End main loop
    PTGQUE2bits.STEP9 = PTGJMP(0);

    //Start the PTG
    PTGCONbits.ON = 1;
    PTGCONbits.PTGSTRT = 1;
}

void ADC_initialize() {

    //Enable analog inputs AD1AN0 - AD1AN5
    _ANSELA2 = 1;
    _ANSELA4 = 1;

    _ANSELA6 = 1;
    _ANSELA5 = 1;
    _ANSELB1 = 1;
    _ANSELB3 = 1;

    //Enable ADC
    AD1CONbits.ON = 1;
    while(!AD1CONbits.ADRDY);

    //Assign ADC inputs to core 1 channels 0 - 5
    AD1CH0CON1bits.MODE = 0;   //Single-sample mode
    AD1CH0CON1bits.PINSEL = 0; //Positive input is AD1AN0/RA2
    AD1CH0CON1bits.NINSEL = 0; //Single-ended mode

    AD1CH1CON1bits.MODE = 0;   //Single-sample mode
    AD1CH1CON1bits.PINSEL = 1; //Positive input is AD1AN1/RA4
    AD1CH1CON1bits.NINSEL = 0; //Single-end
```

30.6. Interrupts

The PTG generates three types of interrupts: the individual interrupt requests, the PTG Step interrupt and the PTG Watchdog Timer time-out interrupt.

30.6.1. PTG Individual Interrupt Requests

The PTG individual interrupt requests are generated using the `PTGIRQ` step command, with the parameter indicating which of several interrupts are supported. See Table 30-6 for specific information. Refer to "Interrupt Controller" for more information.

30.6.2. PTG Step Interrupt

The PTG Step interrupt is generated in Single-Step mode (PTGSSEN = 1) each time a step command has completed execution. This can be used for debugging a PTG step command sequence. See Single-Step Mode for more information about Single-Step mode.

30.6.3. PTG Watchdog Timer (WDT) Interrupt

The PTG WDT interrupt occurs when the WDT times out while waiting for an input trigger using the `PTGWLO`/`PTGWHI` step commands. This indicates that an expected trigger was missed, allowing the application to take corrective action.

Refer to PTG Watchdog Timer for more information about the WDT.

30.7. Power-Saving Modes

The PTG module supports three Power-Saving modes:

- Disabled: the PTG module is not clocked in this mode.
- Idle: the processor core and selected peripherals are shut down.
- Sleep: the entire device is shut down.

30.7.1. Disabled Mode

When ON = 1, the module is considered in an Active mode and is fully powered and functional. When ON = 0, the module is turned off. The PTG clock portions of the circuit that are disabled for maximum current savings. Only the control registers remain functional for reading and writing to allow the software to change the module's Operational mode. The module sequencer is reset.

30.7.2. Idle Mode

To continue full module operation while the device is in Idle mode, the PTGSIDL bit must be cleared prior to entry into Idle mode. If PTGSIDL = 1, the PTG module will behave the same way in Idle mode as it does in Sleep mode.

30.7.3. Sleep Mode

If the device enters Sleep mode while the PTG module is enabled (ON = 1), the module will be suspended in its current state until clock execution resumes. This situation should be avoided as it might result in unexpected operation. It is recommended that all peripherals be shut down in an orderly manner prior to entering Sleep mode.

31. 32-Bit Programmable Cyclic Redundancy Check (CRC) Generator

The Programmable Cyclic Redundancy Check (CRC) module is a software-configurable CRC generator. The module provides a hardware-implemented method of quickly generating checksums for various communication and security applications. The CRC engine calculates the CRC checksum without CPU intervention; moreover, it is much faster than the software implementation.

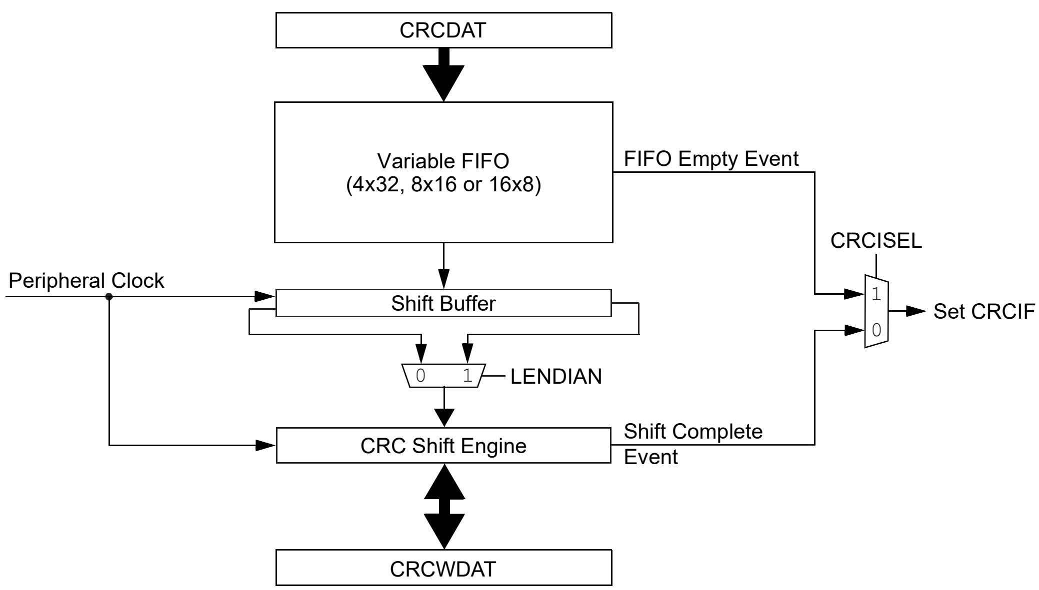
The programmable CRC generator provides the following features:

- User-Programmable CRC Polynomial Equation, up to 32 bits
- Programmable Shift Direction (Little or Big-Endian)
- Independent Data and Polynomial Lengths
- Configurable Interrupt Output
- Data FIFO

31.1. Architectural Overview

The programmable CRC generator module can be divided into two parts: the control logic and the CRC engine. The control logic incorporates a register interface, data FIFO, an interrupt generator and a CRC engine interface. The CRC engine incorporates a CRC calculator, which is implemented using a serial shifter with XOR function. A simplified block diagram is shown in Figure 31-1.

Figure 31-1. Simplified Block Diagram of the Programmable CRC Generator



The checksum is a unique number associated with a message or a particular block of data containing several bytes. Whether it is a data packet for communication or a block of data stored in memory, a checksum helps to validate it before processing. The simplest way to calculate a checksum is to add together all the data bytes present in the message. However, this method of checksum calculation fails badly when the message is modified by inverting or swapping groups of bytes. Also, it fails when null bytes are added anywhere in the message.

The Cyclic Redundancy Checksum (CRC) is a more complicated but robust error-checking algorithm. The main idea behind the CRC algorithm is to treat a message as a binary bit stream and divide it by a fixed binary number. The remainder from this division is considered to be the checksum. Like in division, the CRC calculation is also an iterative process. The only difference is that these operations are done on modulo arithmetic, based on mod 2. For example, division is replaced with the XOR operation (i.e., subtraction without carry). The CRC algorithm uses the term polynomial to perform all of its calculations. The divisor, dividend and remainder that are represented by numbers are termed as polynomials with binary coefficients. For example, the number 25h (`11001`) is represented as:

$(1 * x^4) + (1 * x^3) + (0 * x^2) + (0 * x^1) + (1 * x^0)$ or $x^4 + x^3 + x^0$

In order to perform the CRC calculation, a suitable divisor is first selected. This divisor is called the generator polynomial. Since CRC is used to detect errors, a generator polynomial of a suitable length needs to be chosen for a given application, as each polynomial has different error detection capabilities. Some polynomials are widely used for many applications, but the error detecting capabilities of any particular polynomial are beyond the scope of this reference section.

The CRC algorithm is straightforward to implement in software. However, it requires considerable CPU bandwidth to implement the basic requirements, such as shift, bit test and XOR. Moreover, the CRC calculation is an iterative process, and additional software overhead for data transfer instructions puts an enormous burden on the MIPS requirement of a microcontroller. In contrast, the software-configurable CRC hardware module facilitates a fast CRC checksum calculation with minimal software overhead.

31.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x02C8	CRCCON	31:24				DWIDTH[4:0]				
		23:16				PLEN[4:0]				
		15:8	ON		SIDL	VWORD[4:0]				
		7:0	CRCFULL	CRCEMPTY	CRCISEL	CRCGO	LENDIAN	MOD		
0x02CC	CRCXOR	31:24	X[30:23]							
		23:16	X[22:15]							
		15:8	X[14:7]							
		7:0	X[6:0]							X
0x02D0	CRCDAT	31:24	DATA[31:24]							
		23:16	DATA[23:16]							
		15:8	DATA[15:8]							
		7:0	DATA[7:0]							
0x02D4	CRCWDAT	31:24	CRCWDAT[31:24]							
		23:16	CRCWDAT[23:16]							
		15:8	CRCWDAT[15:8]							
		7:0	CRCWDAT[7:0]							

31.2.1. CRC Control Register

Name: CRCCON
Offset: 0x2C8

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
				DWIDTH[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				PLEN[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	ON		SIDL	VWORD[4:0]				
Access	R/W		R/W	R	R	R	R	R
Reset	0		0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCFULL	CRCEMPTY	CRCISEL	CRCGO	LENDIAN	MOD		
Access	R	R	R/W	R/W	R/W	R/W		
Reset	0	1	0	0	0	0		

Bits 28:24 – DWIDTH[4:0] Data Word Width Configuration bits
Configures the width of the data word (data word width - 1).

Bits 20:16 – PLEN[4:0] Polynomial Length Configuration bits
Configures the length of the polynomial (polynomial length - 1),

Bit 15 – ON CRC Enable bit

Value	Description
1	Enables module.
0	Disables module. All state machines, pointers and CRCSHFT/CRCDAT reset. Other SFRs are not reset.

Bit 13 – SIDL CRC Stop in Idle Mode bit

Value	Description
1	Discontinue the module operation when the device enters Idle mode.
0	Continue the module operation in Idle mode.

Bits 12:8 – VWORD[4:0] Valid Word Pointer Value bits
Indicates the number of valid words in the FIFO.

- Has a maximum value of 16 when DWIDTH[4:0] ≦ 7 (data words 8-bit wide or less).
- Has a maximum value of 8 when DWIDTH[4:0] ≦ 15 (data words from 9-bit to 16-bit wide).
- Has a maximum value of 4 when DWIDTH[4:0] ≦ 31 (data words from 17-bit to 32-bit wide).

Bit 7 – CRCFULL FIFO Full bit

Value	Description
1	FIFO is full.

Value	Description
0	FIFO is not full.

Bit 6 – CRCEMPTY FIFO Empty bit

Value	Description
1	FIFO empty
0	FIFO not empty

Bit 5 – CRCISEL CRC Interrupt Selection bit

Value	Description
1	Interrupt on FIFO empty; final word of data still shifting through CRC.
0	Interrupt on shift complete and results ready.

Bit 4 – CRCGO Start CRC bit

Value	Description
1	Start CRC serial shifter; clearing the bit aborts shifting.
0	CRC serial shifter turned off.

Bit 3 – LENDIAN Little Endian Enable bit

Value	Description
1	Data word is shifted into the CRC starting with the LSb (little endian).
0	Data word is shifted into the CRC starting with the MSb (big endian).

Bit 2 – MOD Accumulator Mode bit

Value	Description
1	Accumulator configured for $x^{16} + x^{12} + x^5 + 1$ (when MOD = 1)
0	Accumulator configured for $x^{16} + x^{12} + x^5 + 1$ (when MOD = 0)

31.2.2. CRC XOR Register

Name: CRCXOR
Offset: 0x2CC

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	X[30:23]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	X[22:15]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	X[14:7]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	X[6:0]							X
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

Bits 31:1 – X[30:0] XOR of Polynomial Term X^N Enable bits

Bit 0 – X XOR of Polynomial Term X^N Enable bit
Note: Bitfield is read-only.

31.2.3. CRC Data Register

Name: CRCDAT
Offset: 0x2D0

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	DATA[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	23	22	21	20	19	18	17	16
	DATA[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset								

Bits 31:0 – DATA[31:0] CRC Input Data bits
Writing to this register fills the FIFO. Reading from this register returns ‘0’.

31.2.4. CRC Result Register

Name: CRCWDAT
Offset: 0x2D4

Legend: R = Readable bit; W = Writable bit

Bit	31	30	29	28	27	26	25	24
	CRCWDAT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	CRCWDAT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CRCWDAT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCWDAT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – CRCWDAT[31:0] CRC Shift Register bits
Writing to this register writes to the CRC shifter register through the CRC write bus. Reading from this register reads the CRC read bus.

31.3. Operation

31.3.1. Polynomial Interface

The CRC module can be programmed for CRC polynomials of up to the thirty-second order using up to 32 bits. Polynomial length, which reflects the highest exponent in the equation, is selected by the PLEN[4:0] bits (CRCCON[20:16]). The CRCXOR registers control which exponent terms are included in the equation. Setting a particular bit includes that exponent term in the equation functionally; this includes an XOR operation on the corresponding bit in the CRC engine. Clearing the bit disables the XOR.

For example, consider two CRC polynomials, one a 16-bit equation and the other a 32-bit equation. To program these polynomials into the CRC generator, set the register bits as shown in Table 31-1.

$x^{16} + x^{12} + x^5 + 1$ and

and

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Table 31-1. CRC Setup Examples for 16 and 32-Bit Polynomials

CRC Control Bits	Bit Values	
	16-Bit Polynomial	32-Bit Polynomial
PLEN[4:0]	01111	11111
X[31:16]	0000 0000 0000 0000	0000 0100 1100 0001

Table 31-1. CRC Setup Examples for 16 and 32-Bit Polynomials (continued)

CRC Control Bits	Bit Values	
	16-Bit Polynomial	32-Bit Polynomial
X[15:1]	`0001 0000 0010 0001`	`0001 1101 1011 0111`

Note that the appropriate positions are set to '`1`' to indicate that they are used in the equation (e.g., X26 and X23). The MSb of the polynomial does not affect the calculation and can be set to any value.

31.3.2. Data Shift Direction

The LENDIAN bit (CRCCON[3]) is used to control the shift direction. By default, the CRC module will shift data through the engine, MSb first (LENDIAN = `0`). Setting LENDIAN to '`1`' causes the CRC module to shift data, LSb first. This setting allows better integration with various communication schemes and removes the overhead of reversing the bit order in software. Note that this only changes the direction the data are shifted into the engine. The result of the CRC calculation will still be a normal CRC result, not a reverse CRC result.

dsPIC33A devices are little-endian. When the CRC module is configured for the big-endian (LENDIAN = `0`), the input data bytes and words must be swapped in the application code before loading them into the data FIFO (CRCDAT registers).

31.3.3. Data FIFO

The module incorporates a FIFO that works with a variable data width. The data width is defined by the DWIDTH[4:0] bits (CRCCON[28:24]). It can be configured to any value between 1 and 32 bits. The logic associated with the FIFO contains a 5-bit counter, VWORD[4:0] bits (CRCCON[12:8]).

The value in the VWORD[4:0] bits indicates the number of unprocessed data elements in the FIFO. The FIFO is:

- 16-word deep when DWIDTH[4:0] ≤ 7 (data words, 8-bit wide or less).
- 8-word deep when DWIDTH[4:0] ≤ 15 (data words from 9 to 16-bit wide).
- 4-word deep when DWIDTH[4:0] ≤ 31 (data words from 17 to 32-bit wide).

The data for the CRC calculation must be written into the FIFO using the CRCDAT registers. Reading the CRCDAT registers always returns zero. To accommodate the MSb first shift method (LENDIAN = `0`), byte and word swapping must be done in software when filling the FIFO.

Note: Ensure that the new data are not written into the CRCDAT registers when the CRCFUL bit is set; if the new data are written, it will be ignored.

When all shifts are done (i.e., the FIFO is empty and the CRC shift engine is Idle), it is possible to change the FIFO width (DWIDTH[4:0] bits) without any information loss or CRC result damage.

With a data width of eight bits or less, the FIFO increments on a write to the lower byte of the CRCDAT register (a byte access to the CRCDAT register must be used). The smallest data element that can be written into the FIFO is one byte.

For example, if DWIDT[20:16] is five, then the size of the data are DWIDT[20:16] + 1 or six. The data are written as a whole byte; the two unused upper bits are ignored. Once the data byte is written into the CRCDAT register, the value of the VWORD[4:0] bits (CRCCON[12:8]) increments by one.

With data widths more than 8 bits and less than or equal to 16 bits, the FIFO increments on a write to the CRCDAT register (16-bit word access to the CRCDAT register must be used). Unused upper data bits are ignored. The value of the VWORD[4:0] bits is incremented for every write to the CRCDAT register.

When the data width is greater than 16 bits, any write to the CRCDAT register increments the VWORD[4:0] bits by one. Writing the lower word into the CRCDAT register must be done before writing the upper word into the CRCDAT register. Unused upper data bits are ignored.

31.3.4. CRC ENGINE

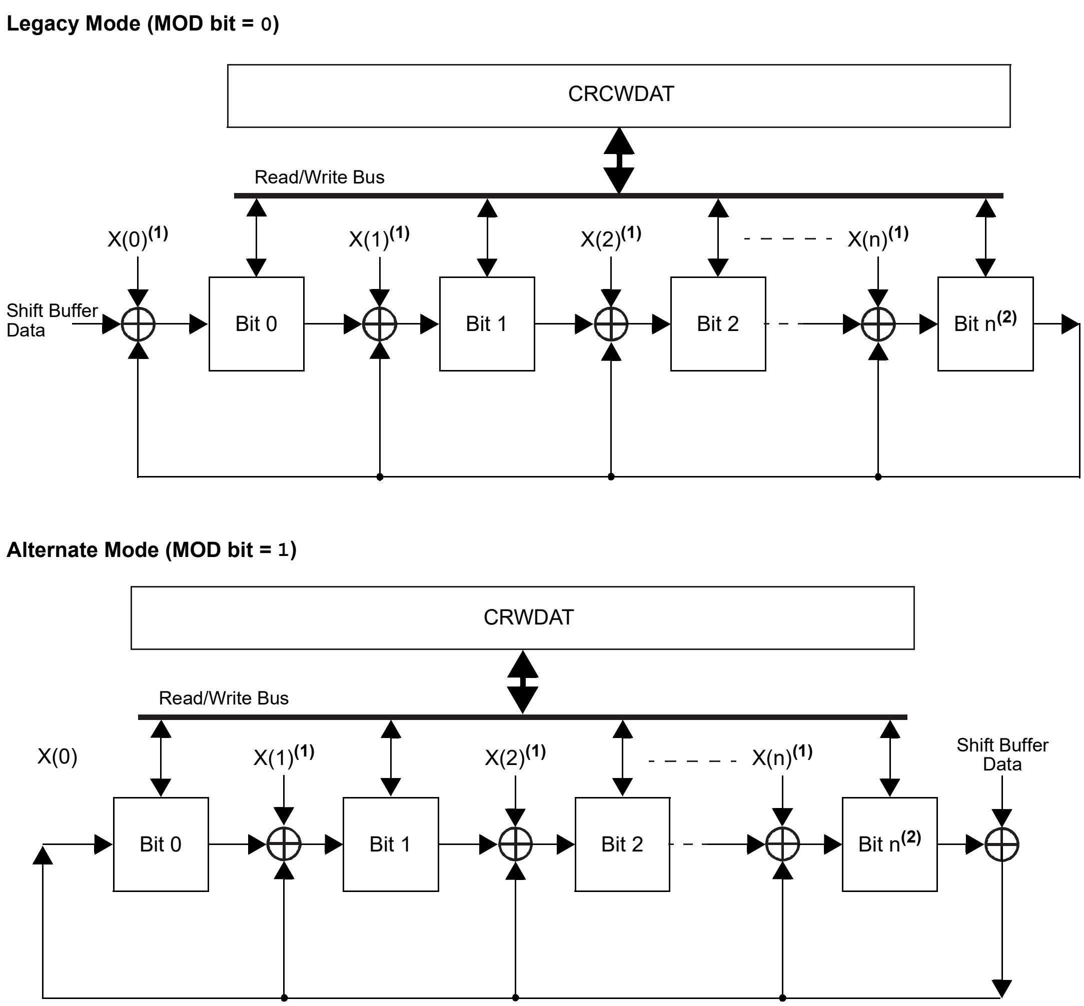
31.3.4.1. Generic CRC Engine

The CRC engine is a serial shifting CRC calculator that is configurable through multiplexer settings. The engine can also be configured to where shift buffer data are introduced using the MOD bit (CRCCON[2]). A simplified diagram of the CRC shift engine is shown in Figure 31-2.

The CRC algorithm uses a simplified form of arithmetic process using the XOR operation instead of binary division. The coefficients of the generator polynomial are programmed with the CRCXOR registers. Writing a '1' into a location enables XORing of that element in the polynomial. The length of the polynomial is programmed using the PLEN[4:0] bits in the CRCCON register (CRCCON[20:16]). The value of PLEN[4:0] signals the length of the polynomial and switches a multiplexer to indicate the tap from which the feedback originated.

The result of the CRC calculation is obtained by reading the CRCWDAT registers.

Figure 31-2. CRC Shift Engine Detail



Notes:

1. Each XOR stage of the shift engine is programmable.
2. Polynomial Length n is determined by (PLEN[4:0] + 1).

31.3.4.2. CRC Engine Interface

31.3.4.3. FIFO to CRC Shift Engine

To start moving the data from the FIFO to the CRC shift buffer, the CRCGO bit (CRCCON[4]) must be set. The serial shifter starts shifting data from the shift buffer to the CRC shift engine, starting from the MSb first for LENDIAN = `0` and LSb first for LENDIAN = `1`, when CRCGO = `1` and the value of VWORD[4:0] is greater than zero. If the CRCFUL bit was set earlier, then it is cleared when the VWORDx bits decrement by one. The VWORD[4:0] bits decrement by one when a FIFO location is moved to the shift buffer. The serial shifter continues shifting until the VWORD[4:0] bits reach zero; at this point, the CRCEMPTY bit becomes set to indicate that the FIFO is empty. If the CRCGO bit is cleared during a CRC calculation, then the CRC shift engine will stop calculating until the CRCGO bit is set.

The application can write into the FIFO while the shift operation is in progress. The CRCFULL bit should be monitored. If the CRCFULL bit is not set, another word can be written into the FIFO. At least one instruction cycle must pass after a write to the CRCDAT registers before a read of the valid value of the VWORD[4:0] bits.

When the VWORD[4:0] bits reach the maximum value for the configured value of the DWIDTH[4:0] bits, the CRCFULL bit becomes set. When the VWORD[4:0] bits reach zero, the CRCEMPTY bit becomes set. The FIFO is emptied and the VWORD[4:0] bits are set to '`00000`' whenever the ON bit is '`0`'.

31.3.4.4. Number of Clock Cycles to Shift Data

The data from FIFO go to the shift buffer. It takes two peripheral clock cycles to start moving the data words from FIFO to the shift buffer. The data from the shift buffer are then shifted to the CRC shift engine. It takes (DWIDTH[4:0] + 1) clock cycles to completely move the data from the shift buffer to the CRC shift engine. For example, if DWIDTH[4:0] = `5`, then the data length is six bits (DWIDTH[4:0] + 1), and six cycles are required to shift the data. In this case, only six bits of a byte are shifted out. The two MSbs of each byte are don't care bits. Similarly, for a 12-bit polynomial selection, the Most Significant four bits of each word are ignored.

31.3.4.5. CRC Initial Value

The access to the CRC shift engine is provided through the CRCWDAT registers. These registers can be loaded with a desired CRC initial value prior to the start of the calculations. The form of this initial value depends on the operating mode selected by the MOD bit (CRCCON[2]).

In Alternate mode (MOD bit = `1`, not available on all devices), the CRC initial value must be in direct form.

In Legacy mode (MOD bit = `0`), the CRC initial value must be in non-direct form. The non-direct initial value is a value for which the CRC calculation gives the desired direct CRC initial value. For example, if the application uses CRC-32 polynomial, 0x04C11DB7, and must start the calculations from the CRC direct initial value, 0xFFFFFFFF, then the non-direct value, 0x46AF6449, must be loaded in the CRCWDAT registers (the CRC of this non-direct value, 0x46AF6449, is 0xFFFFFFFF). When the non-direct initial value is written into the shift engine using the CRCWDAT registers, it will be converted by the CRC module to the direct initial value after (PLEN[4:0] + 1) peripheral clock cycles.

Note: The write to the CRCWDAT registers clears/resets the shift buffer.

Usually, the CRC calculation starts from the same initial value every time. In this case, the non-direct initial value can be found just once and then can be defined as a constant in the application code.

Note: The CRC non-direct initial value of zero is zero.

Example 31-1 shows a possible software routine to get the non-direct initial value from the direct initial value.

The CRC module can be used to get the non-direct initial value. To do this:

1. Enable the CRC module (ON = 1) and shifts (CRCGO = 1).
2. Shift the polynomial value right by one.
3. Reverse the bit order of the shifted polynomial value.
4. Write this result in the CRCXOR registers.
5. Set the data width and polynomial length (DWIDTH[4:0] and PLEN[4:0] bits) to the polynomial order (length).
6. Reverse the bit order of the desired direct initial value.
7. Write the reversed initial value in the CRCWDAT registers.
8. Write a dummy data to the CRCDAT registers and wait two peripheral clock cycles to move the data from the FIFO to the shift buffer and (PLEN[4:0] + 1) peripheral clock cycles to shift out the result. Alternatively, clear the CRC Interrupt Selection bit (CRCISEL = 0) to get the interrupt when shifts from the shift buffer are done, clear the CRC interrupt flag, write dummy data in the CRCDAT registers and wait for the CRC interrupt flag to set.
9. Read the value from the CRCWDAT registers.
10. Reverse the bit order of the read result; it will give the final non-direct initial value.

Example 31-2 shows one way to implement this procedure.

To continue calculations of the full data message in the applications where the intermediate CRC sums must be read in the middle of the calculations, the non-direct value must be calculated and set to the CRCWDAT registers again. In this case, the CRC direct initial value will be an intermediate CRC result read.

Example 31-1. Software Routine to Calculate the Non-Direct Initial Value

```
unsigned int CalculateNonDirectSeed(
unsigned int seed,                      // direct CRC initial value
unsigned int polynomial,                // polynomial
unsigned char polynomialOrder)          // polynomial order
{
unsigned char lsb;
unsigned char i;
unsigned int msbmask;

    msbmask = ((unsigned int)1)<<(polynomialOrder-1);
    for (i=0; i<polynomialOrder; i++) {
        lsb = seed & 1;
        if (lsb) seed ^= polynomial;
        seed >>= 1;
        if (lsb) seed |= msbmask;
}
return seed;                           // return the non-direct CRC initial
value}
```

Example 31-2. Calculating the Non-Direct Initial Value (MOD bit = 0)

```
unsigned int CalculateNonDirectSeed(unsigned int seed,  // direct CRC initial value
          unsigned int polynomial,                         // polynomial
unsigned char polynomialOrder)                          // polynomial order (valid values are
                                                        // 8, 16, 32 bits)
{
    CRCCON1 = 0;
    CRCCON2 = 0;
    CRCCON1bits.CRCEN= 1;                                  // enable CRC
    CRCCON1bits.CRCISEL= 0;                                // interrupt when all shifts
are done
    CRCCON2bits.DWIDTH= polynomialOrder-1;                 // data width
    CRCCON2bits.PLEN= polynomialOrder-1;                   // polynomial length
    CRCCON1bits.CRCGO= 1;                                  // start CRC calculation

    polynomial >>= 1;                                      // shift the polynomial right
    polynomial = ReverseBitOrder(polynomial, polynomialOrder);  // reverse bits order of
the polynomial
```

```
    CRCXOR = polynomial;                                    // set the reversed polynomial
    seed = ReverseBitOrder(seed, polynomialOrder);          // reverse bits order of the
seed value
    CRCWDAT = seed;


    _CRCIF = 0;                                             // clear interrupt flag
    switch(polynomialOrder)                                 // load dummy data to shift
out the
                                                            // seed result
    {
    case 8:
       *((unsigned char*)&CRCDAT) = 0;                      // load byte
        while(!_CRCIF);                                     // wait until shifts are done
        seed = CRCWDAT&0x00ff;                              // read reversed seed
    case 16: CRCDAT = 0;                                    // load short
        while(!_CRCIF);                                     // wait until shifts are done
        seed = CRCWDAT;                                     // read reversed seed
        break;
    case 32:
    // load long
    CRCDAT = 0;
    while(!_CRCIF);                                         // wait for shifts are done
    seed = CRCWDAT;                                         // read reversed seed
    break;
  default:
     ;
    }

    seed = ReverseBitOrder(seed, polynomialOrder);          // reverse the bit order to
get the
                                                            // non-direct seed
    return seed;                                            // return the non-direct CRC
initial value}
// WHERE THE FUNCTION TO REVERSE THE BIT ORDER CAN BE

unsigned int ReverseBitOrder(unsigned int data,             // input data
              unsigned char numberOfBits)                   // width of the input data,
                                                            // valid values are 8,16,32
bits
{
unsignedintmaskin = 0;
unsignedintmaskout = 0;
unsignedintresult = 0;
unsignedchari;

    switch(numberOfBits)
    {
        case 8:
            maskin = 0x80;
            maskout = 0x01;
            break;
        case 16:
            maskin = 0x8000;
            maskout = 0x0001;
            break;
        case 32:
            maskin = 0x80000000;
            maskout = 0x00000001;
            break;
        default:
            ;
    }
    for(i=0; i<numberOfBits; i++)
    {
        if(data&maskin){
            result |= maskout;
        }
            maskin >>= 1;
            maskout <<= 1;
        }
        return result;
}
```

31.3.4.6. CRC Result

Reading the result of a CRC calculation depends on the selected operating mode.

In Alternate mode (MOD bit = `1` not available on all devices), the result is available in the CRCWDAT registers when all the data in the CRC FIFO buffer have been processed. Submitting dummy data to generate extra cycles is not required.

In Legacy mode (MOD bit = 0), the CRC module requires (PLEN[4:0] + 1) extra peripheral clock cycles to finish the calculations. To generate these additional cycles, the dummy data, with the width equal to the polynomial order (length), must be loaded into the CRCDAT registers. After the shifts are finished, the final CRC result can be read from the CRCWDAT registers.

To get the final CRC result after all data are loaded into the CRC module follow the following procedure below.

If the data width (DWIDTH[4:0]) is more than the polynomial length (PLEN[4:0]):

1. Wait for the data FIFO to empty (CRCEMPTY bit is set).
2. Wait (DWIDTH[4:0] + 1) clock cycles to make sure that shifts from the shift buffer are finished.
3. Change the data width to the polynomial length (DWIDTH[4:0] = PLEN[4:0]).
4. Write one dummy data word to the CRCDAT registers.
5. Wait two peripheral clock cycles to move the data from the FIFO to the shift buffer, plus (PLEN[4:0] + 1) clock cycles to shift out the result. Alternatively, clear the CRC Interrupt Selection bit (CRCISEL = 0) to get the interrupt when all shifts are done. Clear the CRC interrupt flag. Write dummy data in the CRCDAT registers and wait until the CRC interrupt flag is set.
6. Read the final CRC result from the CRCWDAT registers.
7. Restore the data width (DWIDTH[4:0] bits) for further calculations (optional).

   If the data width (DWIDTH[4:0]) is equal to, or less than, the polynomial length (PLEN[4:0]), the procedure to get the result can be different.
8. Clear the CRC Interrupt Selection bit (CRCISEL = 0) to get the interrupt when all shifts are done.
9. Suspend the calculation by setting CRCGO = 0.
10. Clear the CRC interrupt flag.
11. Write the dummy data with the total data length equal to the polynomial length in the CRCDAT registers.
12. Resume the calculation by setting CRCGO = 1.
13. Wait until the CRC interrupt flag is set.
14. Read the final CRC result from the CRCWDAT registers.

    When the CRC result is achieved, the CRC non-direct initial value should be written again into the CRCWDAT registers to clear/reset the shift buffer from the previously loaded dummy data to start a new calculation. Example 31-3 shows these steps for the polynomial orders of 8, 16 and 32 bits.

Example 31-3. Routine to Get the Final CRC Result in Legacy Mode (MOD bit = 0)

```
unsigned int GetCRC(unsigned char polynomialOrder,   // valid values are 8,16,32
unsigned char currentDataWidth)                           // valid values are 8,16,32
{
unsignedintcrc = 0;

   while(!CRCCON1bits.CRCMPT);                            // wait until data FIFO is empty

   asmvolatile("repeat %0\n nop" : : "r"(currentDataWidth>>1)); // wait until previous
data
                                                          // shifts are done
CRCCON2bits.DWIDTH = polynomialOrder-1;                   // set data width to polynomial
length
CRCCON1bits.CRCISEL = 0;                                  // interrupt when all shifts are
done

_CRCIF = 0;                                               // clear interrupt flag

switch(polynomialOrder)
   {
      case 8:                                             // polynomial length is 8 bits
         *((unsigned char*)&CRCDAT) = 0;                  // load byte
         while(!_CRCIF);                                  // wait until shifts are done
         crc = CRCWDATL&0x00ff;                           // get crc
```

```
         break;
      case 16:                                          // polynomial length is 16 bits
         CRCDAT = 0;                                    // load short
         while(!_CRCIF);                                // wait until shifts are done
         crc = CRCWDAT;                                 // get crc
         break;
      case 32:                                          // polynomial length is 32 bits
         CRCDAT = 0;
         while(!_CRCIF);                                // wait until shifts are done
         crc = CRCWDAT;                                 // get crc
         break;
      default:
         ;
   }
   CRCCON2bits.DWIDTH = currentDataWidth-1;             // restore data width for
further
                                                        // calculations
   return crc;                                          // return the final CRC value
}
```

31.3.5. Interrupts

The module generates an interrupt that is configurable by the user for either of the two conditions. If CRCISEL is ‘1’, an interrupt is generated when the VWORD[4:0] bits make a transition from a value of ‘1’ to ‘0’. If CRCISEL is ‘0’, an interrupt will be generated when the FIFO is empty and shifts from the shift buffer are finished.

31.4. Application Examples

The CRC is a robust error checking algorithm in digital communication for messages containing several bytes or words. After calculation, the checksum is appended to the message and transmitted to the receiving station. The receiver calculates the checksum with the received message to verify the data integrity.

31.4.1. Variations

The 32-bit programmable CRC module can be programmed to shift out either the MSb or LSb first. MSb first is a popular implementation as employed in XMODEM protocol. In one of the variations (CCITT protocol) for CRC calculation, the LSb is shifted out first. Discussions on all the variations are beyond the scope of this document, but several variations of CRC can be implemented using this module.

The choice of the polynomial length and the polynomial itself are application dependent. Polynomial lengths of 5, 7, 8, 10, 12, 16 and 32 are normally used in various standard implementations. The following sections explain the recommended step-by-step procedure for CRC calculation. Users can decide whether zeros, or any other values, need to be appended to the message stream. Depending on the application, the user may decide whether any value needs to be appended at all.

31.4.2. Typical Operations

To use the module for a typical CRC calculation:

1. Set the CRCEN bit to enable the module.
2. Configure the module for the desired operation:
   a. Program the desired polynomial using the CRCXOR registers and the PLEN[4:0] bits.
   b. Configure the data width and shift direction using the DWIDTH[4:0] and LENDIAN bits.
3. Set the CRCGO bit to start the calculations.
4. Set the desired CRC initial value in the CRCWDAT registers as described in CRC Initial Value.
5. Load all data into the FIFO by writing to the CRCDAT registers as space becomes available (the CRCFUL bit must be zero before the next data loading).
6. Wait until the data FIFO is empty (CRCMPT bit is set).
7. Read the CRC result as described in CRC Result.

Example 31-4 through Example 31-7 show typical code for different combinations of polynomial length, data width, shift direction and CRC Engine modes.

Example 31-4. CRC-SMBus (8-Bit Polynomial with 32-Bit Data, Big-Endian, MOD bit = 1)

```
// This code is used to swap bytes for big endian
#define            Swap(x) __extension__({ \
unsigned            int __x = (x), __v; \
__asm__ ("wsbh %0,%1;\n\t" \
"rotr %0,16" \
: "=d" (__v) \
: "d" (__x)); \
__v; \
})
// ASCII            bytes "12345678"
volatile            unsigned char __attribute__((aligned(4))) message[] =
{'1','2','3','4','5','6','7','8'};
volatile            unsigned char crcResultCRCSMBUS = 0;
int            main    (void)
{
unsigned            int*    pointer;
unsigned            short      length;
unsigned            int     data;
// standard CRC-SMBUS
#define            CRCSMBUS_POLYNOMIAL ((unsigned int)0x00000007)
#define            CRCSMBUS_SEED_VALUE ((unsigned int)0x00000000)  // direct initial value
CRCCON = 0;
CRCCONbits.MOD = 1;                                      // alternate mode
CRCCONbits.ON  = 1;                                      // enable CRC
CRCCONbits.LENDIAN = 0;                                  // big endian
CRCCONbits.CRCISEL = 0;                                  // interrupt when all shifts are
done
CRCCONbits.DWIDTH  = 32-1;                                 // 32-bit data width
CRCCONbits.PLEN    = 8-1;                                  // 8-bit polynomial order
CRCXOR  = CRCSMBUS_POLYNOMIAL;                               // set polynomial
CRCWDAT = CRCSMBUS_SEED_VALUE;                             // set initial value
CRCCONbits.CRCGO   = 1;                                    // start CRC calculation
pointer = (unsigned int*)message;
length  = sizeof(message)/sizeof(unsigned int);
while(1)
{
while(CRCCONbits.CRCFUL);                                  // wait if FIFO is full
data = *pointer++;                                      // load from little endian
data = Swap(data);                                      // swap bytes for big endian
length--;
if(length == 0)
{
break;
}
CRCDAT = data;                                       // 32-bit word access to FIFO
}
CRCCONbits.CRCGO = 0;                                  // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                              // clear the interrupt flag
CRCDAT  = data;                                     // write last data into FIFO
CRCCONbits.CRCGO = 1;                                // resume CRC calculation
while(!IFS0bits.CRCIF);                                  // wait until shifts are done
crcResultCRCSMBUS = (unsigned char)CRCWDAT&0x00ff;               // get CRC result (must
be 0xC7)
while(1);
return 1;
}
```

Example 31-5. CRC-16 (16-Bit Data with 16-Bit Polynomial, Little-Endian, MOD bit = 1)

```
// ASCII bytes "87654321"
volatile            unsigned        short message[] = {0x3738,0x3536,0x3334,0x3132};
volatile            unsigned        short crcResultCRC16 = 0;
int            main        (void)
{
unsigned            short*       pointer;
unsigned            short       length;
unsigned            short       data;
// standard CRC-16
#define            CRC16_POLYNOMIAL ((unsigned int)0x00008005)
#define            CRC16_SEED_VALUE ((unsigned int)0x00000000)               // direct
initial value
CRCCON = 0;
CRCCONbits.MOD = 1;                                             // alternate mode
CRCCONbits.ON  = 1;                                             // enable CRC
CRCCONbits.CRCISEL = 0;                                            // interrupt when all
shifts are done
CRCCONbits.LENDIAN = 1;                                            // little endian
```

```
CRCCONbits.DWIDTH  = 16-1;                                      // 16-bit data width
CRCCONbits.PLEN    = 16-1;                                      // 16-bit polynomial
order
CRCXOR  = CRC16_POLYNOMIAL;                                        // set polynomial
CRCWDAT = CRC16_SEED_VALUE;                                        // set initial value
CRCCONbits.CRCGO = 1;                                           // start CRC calculation
pointer = (unsigned short*)message;
length = sizeof(message)/sizeof(unsigned short);
while(1)
{
while(CRCCONbits.CRCFUL);                                       // wait if FIFO is full
data = *pointer++;                                           // load data
length--;
if(length == 0)
{
break;
}
*((unsigned short*)&CRCDAT) = data;                                  // 16-bit word
access to FIFO
}
CRCCONbits.CRCGO = 0;                                          // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                       // clear the interrupt
flag
*((unsigned short*)&CRCDAT) = data;                                  // write last data
into FIFO
CRCCONbits.CRCGO = 1;                                          // resume CRC calculation
while(!IFS0bits.CRCIF);                                        // wait until shifts are
done
crcResultCRC16 = (unsigned short)CRCWDAT;                               // get CRC result
(must be 0xE716)
while(1);
return 1;
}
```

Example 31-6. CRC-32 (32-Bit Polynomial with 32-Bit Data, Little-Endian, MOD bit = 1)

```
// ASCII bytes "12345678"
volatile            unsigned        char __attribute__((aligned(4))) message[] =
{'1','2','3','4','5','6','7','8'};
// function to reverse the bit order (OPTIONAL)
unsigned            int        ReverseBitOrder(unsigned int data);
volatile            unsigned        int crcResultCRC32 = 0;
int            main(void)
{
unsigned            int*        pointer;
unsigned            short        length;
// standard CRC-32
#define            CRC32_POLYNOMIAL ((unsigned int)0x04C11DB7)
#define            CRC32_SEED_VALUE ((unsigned int)0xFFFFFFFF)            // direct
initial value
CRCCON = 0;
CRCCONbits.MOD = 1;                                            // alternate mode
CRCCONbits.ON  = 1;                                            // enable CRC
CRCCONbits.CRCISEL = 0;                                        // interrupt when all
shifts are done
CRCCONbits.LENDIAN = 1;                                        // little endian
CRCCONbits.DWIDTH  = 32-1;                                        // 32-bit data width
CRCCONbits.PLEN    = 32-1;                                        // 32-bit polynomial
order
CRCXOR  = CRC32_POLYNOMIAL;                                       // set polynomial
CRCWDAT = CRC32_SEED_VALUE;                                       // set initial value
CRCCONbits.CRCGO = 1;                                          // start CRC calculation
pointer = (unsigned int*)message;
length  = sizeof(message)/sizeof(unsigned int);
while(1)
{
while(CRCCONbits.CRCFUL);                                       // wait if FIFO is full
length--;
if(length == 0)
{
break;
}
CRCDAT = *pointer++;                                           // 32-bit word access to
FIFO
}
CRCCONbits.CRCGO = 0;                                          // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                       // clear the interrupt
flag
CRCDAT  = *pointer;                                            // write last data into FIFO
CRCCONbits.CRCGO = 1;                                          // resume CRC calculation
while(!IFS0bits.CRCIF);                                        // wait until shifts are
done
crcResultCRC32 = CRCWDAT;                                         // get the final CRC
result
// OPTIONAL reverse CRC value bit order and invert (must be 0x9AE0DAAF)
```

```
crcResultCRC32 = ~ReverseBitOrder(crcResultCRC32);
while(1);
return 1;
}
unsigned            int ReverseBitOrder(unsigned int data)
{
unsigned            int         maskin;
unsigned            int         maskout;
unsigned            int         result = 0;
unsigned            char i;
maskin  = 0x80000000;
maskout = 0x00000001;
for(i=0; i<32; i++)
{
if(data&maskin){
result |= maskout;
}
maskin >>= 1;
maskout <<= 1;
}
return            result;
}
```

Example 31-7. Data Width Switching (32-Bit Polynomial, Little-Endian, MOD bit = 1)

```
// ASCII bytes "12345678"
volatile            unsigned        int message1[] = {0x34333231,0x38373635};
// ASCII bytes "123"
volatile            unsigned        char message2[] = {'1','2','3'};
volatile            unsigned        int crcResultCRC32 = 0;
int             main(void)
{
unsigned            char*        pointer8;
unsigned            int*         pointer32;
unsigned            short        length;
#define            CRC32_POLYNOMIAL ((unsigned int)0x04C11DB7)
#define            CRC32_SEED_VALUE ((unsigned int)0xFFFFFFFF)    // direct initial value
CRCCON = 0;
CRCCONbits.MOD = 1;                                        // alternate mode
CRCCONbits.ON  = 1;                                        // enable CRC
CRCCONbits.CRCISEL = 0;                                    // interrupt when all shifts are
done
CRCCONbits.LENDIAN = 1;                                    // little endian
CRCCONbits.DWIDTH  = 32-1;                                    // 32-bit data width
CRCCONbits.PLEN    = 32-1;                                    // 32-bit polynomial order
CRCXOR  = CRC32_POLYNOMIAL;                                   // set polynomial
CRCWDAT = CRC32_SEED_VALUE;                                   // set initial value
CRCCONbits.CRCGO = 1;                                      // start CRC calculation
pointer32 = (unsigned int*)message1;
length    = sizeof(message1)/sizeof(unsigned int);
while(1)
{
while(CRCCONbits.CRCFUL);                                     // wait if FIFO is full
length--;
if(length == 0)
{
break;
}
CRCDAT = *pointer32++;                                     // 32-bit word access to FIFO
}
CRCCONbits.CRCGO = 0;                                      // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                   // clear the interrupt flag
CRCDAT  = *pointer32;                                      // write last 32-bit data into FIFO
CRCCONbits.CRCGO = 1;                                      // resume CRC calculation
while(!IFS0bits.CRCIF);                                    // wait until shifts are done
CRCCONbits.DWIDTH = 8-1;                                      // switch the data width to 8-bit
pointer8 = (unsigned char*)message2;                             // calculate CRC
length   = sizeof(message2)/sizeof(unsigned char);
while(length--)
{
while(CRCCONbits.CRCFUL);                                     // wait if FIFO is full
length--;
if(length == 0)
{
break;
}
*((unsigned char*)&CRCDAT) = *pointer8++;                           // byte access to FIFO
}
CRCCONbits.CRCGO = 0;                                      // suspend CRC calculation
IFS0CLR = _IFS0_CRCIF_MASK;                                   // clear the interrupt flag
*((unsigned char*)&CRCDAT) = *pointer8;                             // write last 8-bit data
into FIFO
CRCCONbits.CRCGO = 1;                                      // resume CRC calculation
while(!IFS0bits.CRCIF);                                    // wait until shifts are done
crcResultCRC32 = CRCWDAT;                                     // get the final CRC result
```

```
(must be 0xE092727E)
while(1);
return 1;
}
```

31.5. Power-Saving Modes

31.5.1. Sleep Mode

If Sleep mode is entered while the module is operating, the module is suspended in its current state until clock execution resumes.

31.5.2. Idle Mode

To continue full module operation in Idle mode, the SIDL bit must be cleared prior to entry into the mode.

If SIDL = 1, the module behaves the same way as it does in Sleep mode; pending interrupt events will be passed on, even though the module clocks are not available.

32. Current Bias Generator (CBG)

The Current Bias Generator (CBG) module is a set of constant current sources that can be used for many purposes including pull-up detection and offset correction. The module is made up of four independent 10 μA outputs and four independent selectable sources ranging from 0-200 μA. The two classes of current source share a physical output pin on the device.

32.1. Device-Specific Information

Output availability for each source variant is shown in Table 32-2 and specific output availability for selectable sources is shown in Table 32-4.

Table 32-1. CBG Summary Table

Module Instances	Number of Outputs	Peripheral Bus Speed
1	4	Slow (1:4 CPU Clock)

Table 32-2. Selectable and Fixed Current Source Availability

Source	Quantity	Type	Range
I10ENx (IBIAS)	4	Fixed	10 μA
ISELOUTx (ISRC)	4	Selectable	30-200 μA

Table 32-3. Channel Per Package Availability

Package Type	ISRCx	IBIASx
36-pin	0	0
48-pin	0,1,2,3	0,1,2,3
64-pin	0,1,2,3	0,1,2,3

Table 32-4. Selectable Source Options

Current Value	Units
30	μA
50	μA
100	μA
120	μA
150	μA
200	μA

32.2. Architectural Overview

32.2.1. Module Description

The CBG module consists of two classes of current sources: four 10 μA sources and four selectable sources ranging from 30-200 μA. Both sets of current sources are routed to the same output pin.

Both sets of current sources can be used in general current sourcing applications to generate voltages using an external resistor or to provide biasing to external circuitry or sensors.

One intended use of the selectable current source is to generate an offset voltage to shift a positive external signal to be within the input range of the internal analog peripherals, such as the ADC. Shifting the input voltage maintains the dynamic range of the AC component of the input signal but removes the offset voltage. An external resistor (refer to Figure 32-2) is used in conjunction with the current source to develop the offset voltage.

The external resistors are large due to the small generated currents. This large resistor value protects the device input circuitry by limiting the current injected into the device when the current source is not enabled.

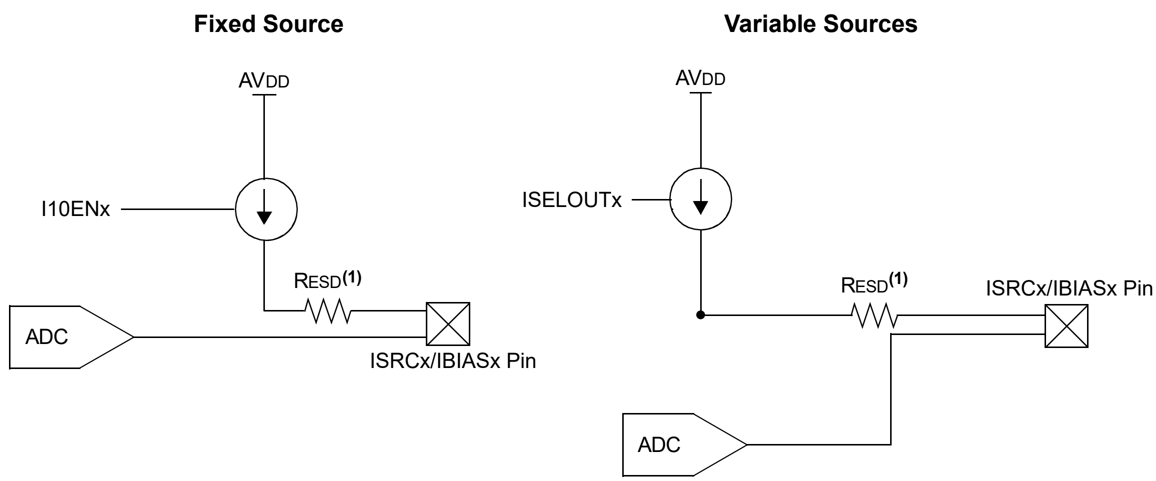
Both classes of current sources can be externally paralleled by connecting the output pins together to increase current.

The large resistors used to create the voltage offset may exceed the ADC input impedance specification. To meet the ADC input requirements, one or more of the following may be required:

- Increase in sampling time
- Use of an internal amplifier, such as an op amp or PGA
- Use of a small capacitor on the input pin if the input signal does not change quickly
- Use of an AC bypass capacitor

A high-level diagram of the CBG circuitry is shown in Figure 32-1.

Figure 32-1. Current Bias Generator Sources



Note:

1. R_{ESD} is typically 300 Ω.

32.3. Current Bias Generator Control Register

Name: IBIASCON
Offset: 0x3B08

Bit	31	30	29	28	27	26	25	24
	Reserved[11:4]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	Reserved[3:0]							VREFSEL
Access	R	R	R	R				R
Reset	0	0	0	0				0

Bit	15	14	13	12	11	10	9	8
	I10EN3	I10EN2	ISELOUT3[2:0]			ISELOUT2[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	I10EN1	I10EN0	ISELOUT1[2:0]			ISELOUT0[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:20 – Reserved[11:0]

Bit 16 – VREFSEL

Value	Description
1	15/16*AV_{DD} Calibration reference selected.
0	1/16*AV_{DD} Calibration reference selected.

Bit 15 – I10EN3 Enable for the ISRC3 Current Source bit

Value	Description
1	Current source is enabled.
0	Current source is disabled.

Bit 14 – I10EN2 Enable for the ISRC2 Current Source bit

Value	Description
1	Current source is enabled
0	Current source is disabled

Bits 13:11 – ISELOUT3[2:0] Current Output Selection for IBIAS3 bits

Value	Description
111	200 µA
110	150 µA
101	120 µA
100	100 µA
011	100 µA
010	50 µA

Value	Description
001	30 µA
000	0 µA

Bits 10:8 – ISELOUT2[2:0] Current Output Selection for IBIAS2 bits

Value	Description
111	200 µA
110	150 µA
101	120 µA
100	100 µA
011	100 µA
010	50 µA
001	30 µA
000	0 µA

Bit 7 – I10EN1 Enable for the ISRC1 Current Source bit

Value	Description
1	Current source is enabled.
0	Current source is disabled.

Bit 6 – I10EN0 Enable for the ISRC0 Current Source bit

Value	Description
1	Current source is enabled.
0	Current source is disabled.

Bits 5:3 – ISELOUT1[2:0] Current Output Selection for IBIAS1 bits

Value	Description
111	200 µA
110	150 µA
101	120 µA
100	100 µA
011	100 µA
010	50 µA
001	30 µA
000	0 µA

Bits 2:0 – ISELOUT0[2:0] Current Output Selection for IBIAS0 bits

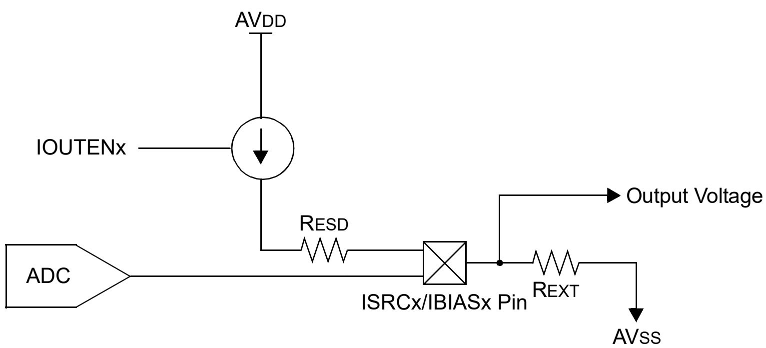
Value	Description
111	200 µA
110	150 µA
101	120 µA
100	100 µA
011	100 µA
010	50 µA
001	30 µA
000	0 µA

32.4. Operation

32.4.1. Basic Operation of the 10 µA Source

The primary application of this source is to generate current to create an external voltage. This voltage can then be measured with the internal ADC or used to bias external circuitry. This class of source can only supply (source) current. To generate an external voltage, an external resistor is connected between the current source pin and AV_{SS} (refer to Figure 32-2). The current flow generates a voltage across the R_EXT resistor (refer to Equation 32-1 and Example 32-1). Multiple sources can be paralleled, as needed, to increase current.

Figure 32-2. 10 µA Current Source



Equation 32-1. Equation for Determining the Value of R_EXT

$V(R_{EXT}) = 10\ \mu A \times R_{EXT}$

$V(_{REXT})$ should not exceed AV_{DD} – 0.5V typical (see Operating Range).

Example 32-1. Enabling a 10 µA Source

R_{EXT} = 10 kOhms, AV_{DD} = 3.3V V_{PIN} = 10k × 10 µA = 100 mv V_{REXT} << 3.3V – .5V, and therefore, meets the $V(R_{EXT})$ requirement

```
// User code to enable a 10 ua source
BIASCONbits.I10EN0 = 1;                // enable 10ua source channel 0
```

32.4.1.1. Device Pin ESD Configuration

Devices have a single ESD resistor on each pin (refer to Figure 32-2 and Figure 32-4).

32.4.2. Basic Operation of the 200 µA Source

The primary application of the 200 µA current source is to remove the DC offset so that the signal to be measured is within the ADC module's input range. Figure 32-3 shows a typical signal to be measured: an AC signal with a DC offset. This class of current source can be used to create a positive shift with an external resistor. The following examples show the basic configurations for shifting the input voltage and the required calculations. The equations in Equation 32-2 are used for positive voltage shift calculations.

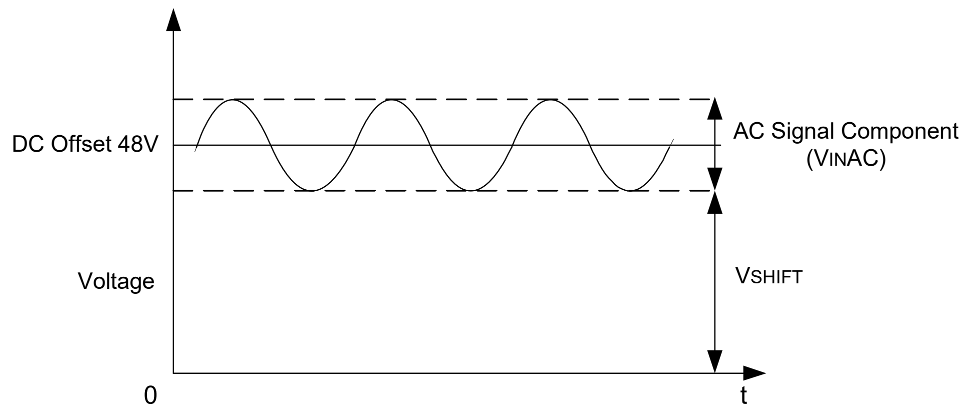
Equation 32-2. Equation for Determining the Value of R_{SHIFT}

$$RSHIFT = \frac{VSHIFT}{I\_SEL\ \mu A}$$

$$VSHIFT = VINDC - \left(\frac{VINAC}{2}\right)$$

Note: $V(R_{SHIFT})$ should not exceed AV_{DD} – 0.7V typical (see Operating Range).

Figure 32-3. AC Signal Component with a DC Offset



32.4.3. Operating Range

The maximum voltage that can be developed across a resistor driven by a current source depends on AV_{DD} and the other voltage sources in the circuit.

When the resistor is connected to AV_{SS}, such as seen in Figure 32-2, the maximum voltage that can be developed is approximately AV_{DD}. However, when the developed voltage is greater than the current source's internal threshold, the output current is reduced. To prevent this, the maximum developed voltage across R_{ESD} + R_{EXT} should be limited to AV_{DD} – 0.5V (typical) with respect to the output current value.

32.4.4. ADC Input Considerations

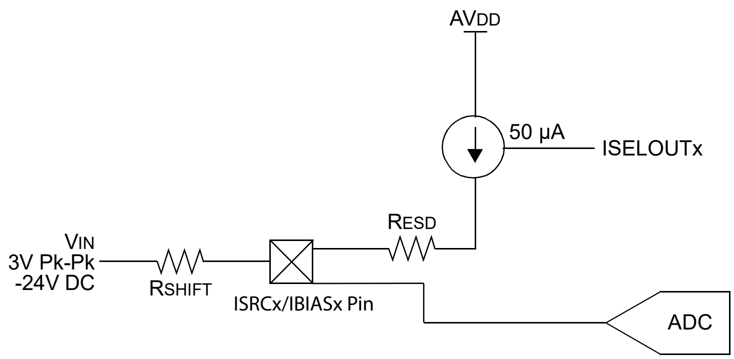
The input impedance for the ADC determines the required change time, specified in ADC clocks or TAD. The impedance consists of the following internal resistances: the ADC channel select switch as well as any external resistance. The large external resistor values required to generate offsets may violate the device's ADC input specifications. This may require the use of an internal amplifier op amp or PGA to isolate the ADC from the large resistance.

32.5. Application Examples

32.5.1. Voltage Shifting for a Positive Input Voltage

To shift a positive input voltage, a single CBG source is used. The source is used to generate a negative voltage to offset the input signal. Figure 32-4, Equation 32-3 and Example 32-2 show the calculations and configuration for this application.

Figure 32-4. Single-Ended Positive Voltage Shift



Equation 32-3. VSHIFT Calculations

VINMIN = – 24V – (3V/2) = – 25.5V

VINMAX = – 24V + (3V/2) = – 22.5V

VSHIFT = |VINMIN|

RSHIFT = 25.5V/50 µA = 510 kOhm

Shift with standard value resistor is *511k × 50 µA = 25.55V*

Input range = (VINMAX – VSHIFT) - (VINMIN – VSHIFT), (49.5V – 46.5V) - (46.5V – 46.5V) = 3V

Example 32-2. Single-Ended Positive Voltage Shift with 50 µA Selection

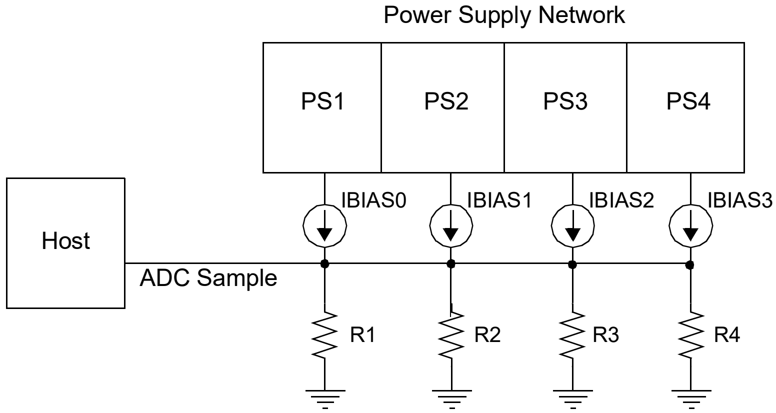
3V p-p Signal with a -24V Offset:

```
// sample code to enable 50 uA current source.
BIASCONbits.ISELOUT1 = 0b010;    // set the 50 uA source on channel 1
```

32.5.2. Instrument Identification Assignment

Another example application of the CBG module is utilizing the 10 µA ISRC output channels to identify and assign an ID for multiple power supplies in a rack-mounted configuration. Typically, rack mounted power supplies use resistors mounted on a backplane to assign IDs to each power supply in the rack. The 10 µA ISRC pins feed these resistors to generate a voltage that is read by the ADC. The ADC value is translated via software into an identifier for each backplane slot.

Figure 32-5. Power Supply Network Identifier



32.6. Interrupts

The current source modules do not generate interrupts.

32.7. Power-Saving Modes

Both classes of current sources continue to operate in Power-Saving modes.

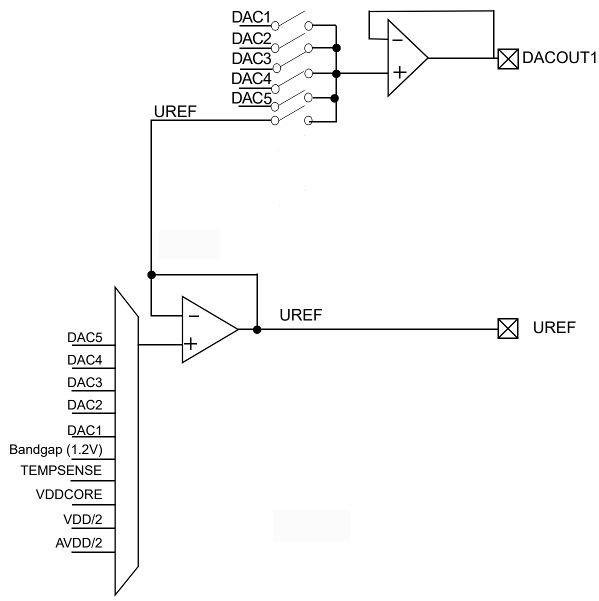
32.8. Effects of a Reset

A Reset forces module registers to their initial Reset values, disabling the current sources.

33. UREF Reference Output

The UREF feature is used to provide a voltage reference for many applications, including biasing op amps and touch guarding. UREF provides a reference voltage to internal nodes and to an external, dedicated pin. A high-level diagram is shown in Figure 33-1.

Figure 33-1. UREF Signal Block Diagram



UREF is implemented as a unity-gain non-inverting buffer. The feedback connection directly connects the amplifier output to the inverting input. The amplifier's non-inverting input can be connected to multiple signals.

33.1. Device-Specific Information

Table 33-1. UREF Summary Table

Module Instances	Clock Input	Peripheral Bus Speed
1	None	Slow (1:4 CPU Clock)

Table 33-2. UREF Input Signals

Value	Selected input
1111	AV_{DD}
1110	AV_{SS}
1101-1011	Reserved
1010	DAC5
1001	DAC4
1000	DAC3
0111	DAC2
0110	DAC1
0101	Temperature sensor
0100	Bandgap (1.2V)
0011	V_{DDCORE}
0010	$V_{DD}/2$
0001	$AV_{DD}/2$

Table 33-2. UREF Input Signals (continued)

Value	Selected input
0000	Reserved

33.2. UREF Control Register 1

Name: UREFCON
Offset: 0x3B24

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON				INSEL[3:0]			
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

Bit	7	6	5	4	3	2	1	0
						OUTSEL[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 15 – ON Module Enable/On bit

Value	Description
1	Enables module.
0	Disables module.

Bits 11:8 – INSEL[3:0] Input Voltage Selection bits
Refer to UREF Reference Output for the list of available UREF signals.

Bits 2:0 – OUTSEL[2:0] Pad Select bits

Value	Description
	Reserved
010	DACBUFFER 1 output
001	UREF output
000	UREF output is disabled.

34. Operational Amplifier (Op Amp)

An Operational Amplifier (Op Amp) is an analog circuit that is used to amplify the difference between the inputs non-inverting (positive) and inverting (negative) inputs. Op Amps can be connected in inverting and non-inverting applications and are commonly used to buffer or amplify signals, such as the signals from sensors, current shunts and resistive dividers. The Op Amps implemented in dsPIC33AK256MPS306 devices provide the following features:

- Up to 100 MHz Gain Bandwidth
- High-Power and Low-Power Operating Modes
- Complementary P and N Channel Differential Pairs on the Input
- User Adjustable Offset

34.1. Device-Specific Information

Table 34-1. Op Amp Summary Table

Op Amp Module Instances	Clock Input	Peripheral Bus Speed
3	None	Slow (1:4 CPU Clock)

Table 34-2. Op Amp Availability by Device Package

Package	Op Amp Availability
64-Pin	OA1, OA2, OA3
48-Pin	OA1, OA2, OA3
36-Pin	OA1, OA2

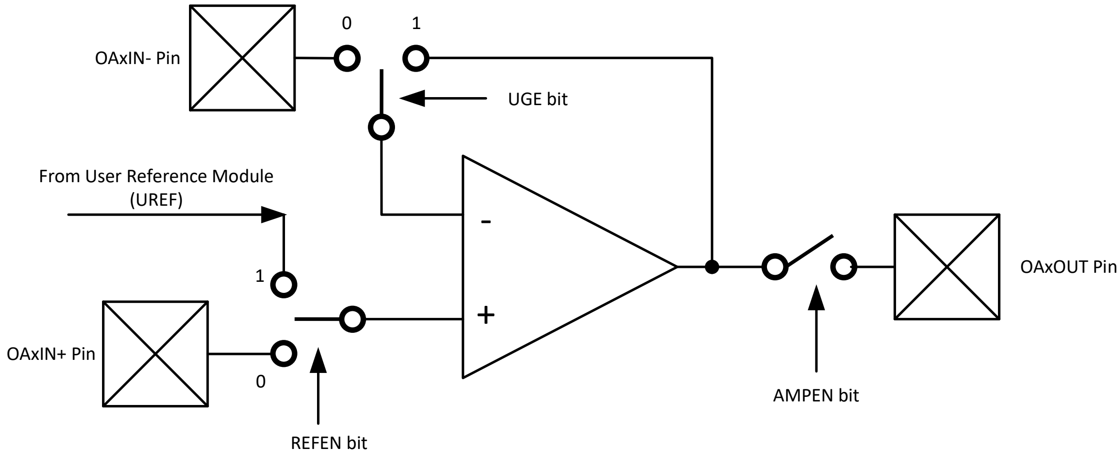
Note: The calibration registers FOPAMPLP and FOPAMPHP are placed at 0x007F20F0 and 0x007F2100, respectively. These registers can be read to determine the factory-calibrated trim settings.

34.2. Architectural Overview

Op amps allow the user to do signal conditioning without dedicated external circuitry. Op Amps in the device support Unity Gain mode without any external connection. The user can adjust the gain of the Op Amp by using external resistors and disabling Unity Gain mode. The Op Amp has an offset trim feature to reduce the effects of input offsets. It is possible to connect the DAC output to an Op Amp internally for test and calibration purposes.

Note: While the Op Amp is not used, the user can disable it by clearing the AMPEN bit to '0' to reduce the power consumption.

Figure 34-1. Basic Op Amp Block Diagram



34.3. Op Amp Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3B0C	AMP1CON1	31:24								
		23:16								
		15:8	AMPEN	HPEN	UGE	DIFFCON[1:0]		REFEN		
		7:0								
0x3B10	AMP1CON2	31:24				POFFSETHP[4:0]				
		23:16				NOFFSETHP[4:0]				
		15:8				POFFSETLP[4:0]				
		7:0				NOFFSETLP[4:0]				
0x3B14	AMP2CON1	31:24								
		23:16								
		15:8	AMPEN	HPEN	UGE	DIFFCON[1:0]		REFEN		
		7:0								
0x3B18	AMP2CON2	31:24				POFFSETHP[4:0]				
		23:16				NOFFSETHP[4:0]				
		15:8				POFFSETLP[4:0]				
		7:0				NOFFSETLP[4:0]				
0x3B1C	AMP3CON1	31:24								
		23:16								
		15:8	AMPEN	HPEN	UGE	DIFFCON[1:0]		REFEN		
		7:0								
0x3B20	AMP3CON2	31:24				POFFSETHP[4:0]				
		23:16				NOFFSETHP[4:0]				
		15:8				POFFSETLP[4:0]				
		7:0				NOFFSETLP[4:0]				

34.3.1. AMPx Control Register 1

Name: AMPxCON1
Offset: 0x3B0C, 0x3B14, 0x3B1C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	AMPEN	HPEN	UGE	DIFFCON[1:0]		REFEN		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – AMPEN Op Amp Enable/On bit

Value	Description
1	Enables the op amp module.
0	Disables the op amp module.

Bit 14 – HPEN High-Power Enable bit

Value	Description
1	Enables Op Amp High-Power (high bandwidth) mode.
0	Disables Op Amp High-Power mode.

Bit 13 – UGE Unity Gain Buffer Enable bit

Value	Description
1	Enables Unity Gain mode.
0	Disables Unity Gain mode.

Bits 12:11 – DIFFCON[1:0] Differential Input Mode Control bits

Value	Description
11	Reserved, do not use.
10	Turn NMOS differential input pair off.
01	Turn PMOS differential input pair off.
00	Use both NMOS and PMOS differential input pair.

Bit 10 – REFEN Input Reference Enable bit

Value	Description
1	Non-inverting input of op-amp connected to internal device voltage source (fixed or adjustable).
0	Non-inverting input of op-amp connected to user pin.

34.3.2. AMPx Control Register 2

Name: AMPxCON2
Offset: 0x3B10, 0x3B18, 0x3B20

Notes:
1. Unit voltage = trim step voltage 3 mV. Therefore, input offset adjustment is between -45 mV and 45 mV.
2. When the Op Amp is configured for gain = `1` and the non-inverting input is set to ~V_{DD}/2, positive values of the offset correction (5'b00001-5'b01111) reduce input offset voltage. Negative values of the offset correction (5'b11110- 5'b10000) increase the input offset voltage.
3. If the Op Amp offset is +6 mV before correction, using an offset correction code of 00010 will change the output voltage by -6 mV, canceling the Op Amp offset.

Bit	31	30	29	28	27	26	25	24
				POFFSETHP[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
				NOFFSETHP[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
				POFFSETLP[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
				NOFFSETLP[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 28:24 – POFFSETHP[4:0] Offset Correction for PMOS Differential Input Pair (High-Power mode) bits
See descriptions for bits[4:0].

Bits 20:16 – NOFFSETHP[4:0] Offset Correction for NMOS Differential Input Pair (High-Power mode) bits
See descriptions for bits[4:0].

Bits 12:8 – POFFSETLP[4:0] Offset Correction for PMOS Differential Input Pair (Low-Power mode) bits
See descriptions for bits[4:0].

Bits 4:0 – NOFFSETLP[4:0] Offset Correction for NMOS Differential Input Pair (Low-Power mode) bits(1)

Value	Description
01111	Reduce the input offset voltage by 15 units of voltage.
. . .	
00111	Reduce the input offset voltage by 7 units of voltage.
00110	Reduce the input offset voltage by 6 units of voltage
00101	Reduce the input offset voltage by 5 units of voltage.
00100	Reduce the input offset voltage by 4 units of voltage.
00011	Reduce the input offset voltage by 3 units of voltage.
00010	Reduce the input offset voltage by 2 units of voltage.

Value	Description
00001	Reduce the input offset voltage by 1 unit of voltage.
00000	No correction from the production calibration value.
11111	No correction from the production calibration value.
11110	Increase the input offset voltage by 1 unit of voltage.
11101	Increase the input offset voltage by 2 units of voltage.
11100	Increase the input offset voltage by 3 units of voltage.
11011	Increase the input offset voltage by 4 units of voltage.
11010	Increase the input offset voltage by 5 units of voltage.
11001	Increase the input offset voltage by 6 units of voltage.
11000	Increase the input offset voltage by 7 units of voltage.
. . .	
10000	Increase the input offset voltage by 15 units of voltage .

34.4. Operations

34.4.1. Enabling the Op Amp

Each Op Amp has its own enable signal. This is controlled by the AMPEN bit in the AMPxCON1 register. Upon reset, AMPEN is set to ‘0’. This keeps all of the Op Amp instances disabled upon reset. To enable instance x of the Op Amp, set the AMPEN bit to ‘1’ in the AMPxCON1 register.

34.4.2. Gain Configurations

34.4.2.1. Unity Gain Setup

The Op Amp supports unity gain operation without requiring any external connection from output to inverted input. To configure the Op Amp in unity gain configuration, set the UGE bit to ‘1’ in the AMPxCON1 register. In this mode, the Op Amp output is internally shorted to the inverting input and the resulting gain will be one.

Example 34-1. Op Amp Unity Gain Buffer

```
AMP1CON1bits.AMPEN = 1;     //Enable Op Amp 1
AMP1CON1bits.UGE = 1;       //Enable unity gain mode
```

34.4.2.2. Uncommitted Op Amp

The Op Amp supports non-unity gain configurations through the use of external resistors. To configure the Op Amp in this gain configuration, set the UGE bit to ‘0’ in the AMPxCON1 register.

Example 34-2. Op Amp Fixed Gain Amplifier with External Resistor

```
AMP1CON1bits.AMPEN = 1;     //Enable Op Amp 1
AMP1CON1bits.UGE = 0;       //Disable unity gain mode
```

34.4.3. Power Modes

The Op Amp Power mode is controlled using the HPEN bit in the AMPxCON1 register. When HPEN bit is set, the Op Amp works in High-Power mode and when cleared, enters the Low-Power mode. In Low-Power mode, the current consumption and bandwidth of the Op Amp are reduced.

34.4.4. Differential Input

The Op Amp supports rail-to-rail operation with both PMOS and NMOS differential pairs. Each pair can be disabled to facilitate independent calibration, by setting the DIFFCON[1:0] bits in the AMPxCON1 register.

The complementary pair inputs allow the Common mode voltage to extend to the supply rails without phase inversion. These are used to design the input stage of single-supply, voltage-feedback amplifiers.

In the region near the V_{SS}, the offset error of the PMOS portion of the Input stage is dominant, and in the region near the V_{DD}, the offset error is dominated by the NMOS transistor pair. That is, as the input Common-mode voltage increases from V_{SS}, the PMOS transistors go into the cutoff region and the input transitions to the NMOS transistors, which enables operation above the positive supply. With a Complementary Pair Input stage, the PMOS transistors enable operation slightly below the negative supply and the NMOS transistors allow for operation slightly above the positive supply.

With the topology used, the amplifier effectively combines the advantages of the PMOS and NMOS transistors for true rail-to-rail input operation. When the input terminals of the amplifier are driven towards the negative rail, the PMOS transistors are turned ON and the NMOS transistors are completely OFF. Conversely, when the input terminals are driven to the positive rail, the NMOS Transistors are ON and PMOS transistors are OFF. This choice of operation allows the users to trade input voltage range for improved INL and offset voltage operation.

34.4.5. Enable Output Monitor

This functionality of Op Amp is controlled by the OMONEN bit in the AMPxCON1 register. The OMONEN bit connects the amplifier output to an input on the ADC to allow the user to read the Op Amp output voltage on the ADC.

34.4.6. Input Offset Trim

It is possible for the user to override factory calibration values and perform user calibration for input-referred voltage offset errors. The AMPxCON2 register is used for input offset, which allows an adjustment to be added or subtracted from the factory offset trim value. The register allows the user to increase or decrease the offset trim up to the maximum or minimum value of trim.

The Op Amp has provisions for the user to measure and adjust the offset. The offset is measured by connecting the DAC input to the Op Amp input. The OMONEN control bit allows the Op Amp output to be connected to the ADC inputs to perform input offset calibration. Note that the user must select the appropriate analog input using the ADC controller and set the OMONEN bit to route these Op Amp signals to the ADC. The OMONEN bit should be set high only for one instance at a time. It is advised not to set the bit high for multiple instances simultaneously.

The Op Amp has high and low bandwidth/power operating modes. The user can trim the input offset error for one or both modes, depending on the application. The complementary P and N channel differential pairs on the input allow full rail-to-rail input voltages to be applied to the amplifier inputs. Each differential pair should be trimmed independently to avoid interactions during the calibration procedure. Therefore, there are four sets of trim adjustment bit fields to account for the two input differential pairs and two power modes described above:

- NOFFSETLP[4:0]
- POFFSETLP[4:0]
- NOFFSETHP[4:0]
- POFFSETHP[4:0]

The procedure for calibrating the input offset error assumes that the user will use minimal external components and will measure voltage offsets using the ADC available on the device:

1. Configure the Op Amp for Unity Gain mode using the internal feedback connection.
2. Connect the non-inverting input of the Op Amp to a midscale voltage reference of $V_{DD}/2$.
3. Select either high power (high bandwidth) or low power (low bandwidth) operational mode.
4. Disable the P-channel differential input pair and enable the N-channel differential pair.

5. Subtract the input voltage measurement from the output voltage measurement value to determine the amount of offset error.
6. Raise or lower the value in the NOFFSETxx register and repeat steps 5 and 6 until the offset error is nulled out.
7. Enable the P channel differential pair of the op amp and disable the N channel differential pair.
8. Repeat steps 5-6 and adjust the POFFSETxx register to null out the offset error.
9. Change the Op Amp power mode (step 3) and repeat steps 4-9 to trim the Op Amp offset voltage in the second power mode, using the appropriate POFFSETxx and NOFFSETxx trim registers.
10. Enable both the P channel and N channel differential pair to return the Op Amp to normal operation with a full input voltage range.
11. Save the values of the POFFSETxx and NOFFSETxx SFRs in non-volatile memory for later retrieval and use by the application.

34.4.7. Interrupts

None.

34.4.8. Power-Saving Modes

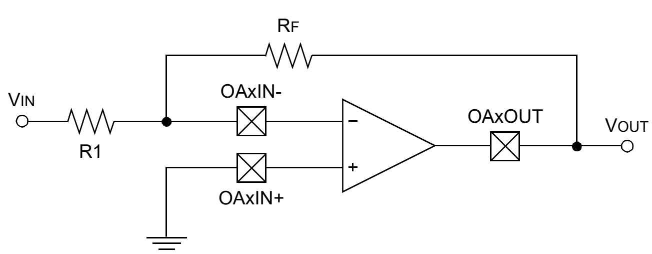
Op amp does not operate in Sleep and Idle modes.

34.5. Op Amp Application Examples

34.5.1. Op Amp Inverting Configuration

In this configuration, an external resistor, R_F, is connected between the OAxOUT pin and OAxIN- pin to provide feedback. An input signal is applied to the OAxIN- pin through resistor R1. The gain in this mode is $-R_F/R1$.

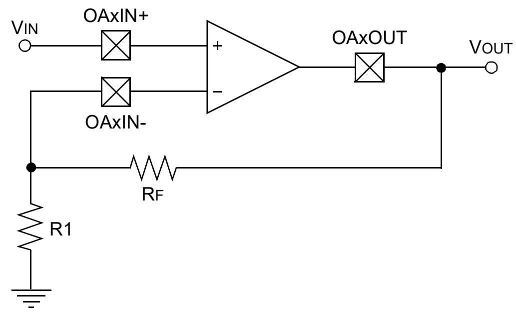
Figure 34-2. Op Amp Inverting Configuration



34.5.2. Op Amp Noninverting Configuration

In this configuration, an external resistor, R_F, is connected between the OAxOUT pin and OAxIN- pin to provide feedback. R1 is connected between OAxIN- and ground. An input signal is applied to the OAxIN+ pin. The gain in this mode is $1 + (R_F/R1)$.

Figure 34-3. Op Amp Noninverting Configuration



35. Watchdog Timer (WDT)

The dsPIC33AK256MPS306 device family WDT can be used to detect system software malfunctions by resetting the device if the WDT is not cleared periodically in software. The WDT can be configured in Window mode or Non-Window mode. Various WDT time-out periods can be selected using the WDT postscaler. The WDT can also be used to wake the device from Sleep or Idle mode (Power-Saving mode).

The Watchdog Timer has the following features:

- Independent Run and Sleep Mode Counters
- May Use Alternate Clock Sources and Postscalers for Run Mode Counter.
- Up to 32 Configurable Time-Out Periods
- Independent 5-bit Postscalers for Run and Sleep Mode Counters
- Hardware and Software Enable
- Windowed WDT Option - Four Window Sizes Controlled by Software Configuration Bits

35.1. Device-Specific Information

Table 35-1. WDT Summary Table

Clock Source	Peripheral Bus Speed
Slow speed peripheral clock	Slow (1:4 CPU Clock)

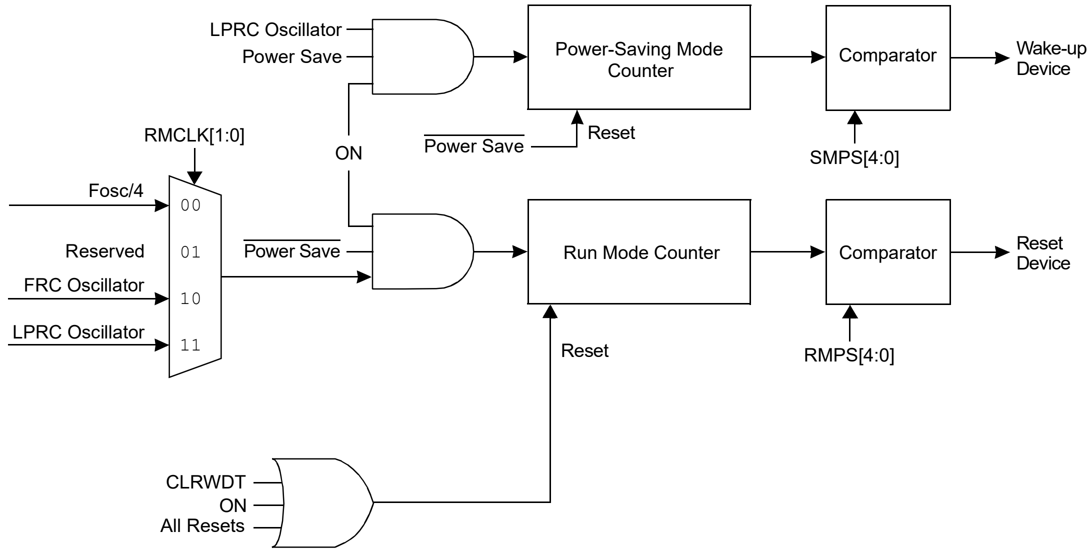
35.2. Architectural Overview

The WDT is a free-running timer with a configurable postscaler. The counter is clocked with an external reference clock until the counter value exceeds the selected WDT period. If enabled, the WDT will continue to operate even if the main processor clock (e.g., the crystal oscillator) fails. The WDT, when enabled, operates from the internal Low-Power RC (LPRC) oscillator clock source or user selectable clock source in Run mode. Refer to Figure 35-1 for a block diagram of the WDT.

The WDT uses separate internal counters for use in Run mode and Sleep/Idle modes. One counter operates only in Run mode; the count value of this counter is frozen when the device is in Sleep or Idle mode. The second counter operates only in Sleep and Idle modes; it is reset when entering Sleep or Idle. In either case, this provides the following benefits:

- A different WDT clock source can be used in Run mode.
- A different postscaler value may be used in Run mode vs. Sleep/Idle modes.
- The Run mode WDT count is preserved while in Sleep or Idle mode, which makes it easier to manage the WDT while in Windowed mode.

Figure 35-1. Watchdog Timer Block Diagram



35.3. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x32C8	WDTCON	31:24								
		23:16								
		15:8	ON	WINSIZE[1:0]		RMPS[4:0]				
		7:0	RMCLK[1:0]		SMPS[4:0]					WINDIS

35.3.1. Watchdog Timer Control Register(1)

Name: WDTCON
Offset: 0x32C8

Legend: cfg = Configurable at Reset

Notes:
1. All these bits reflect the value of FWDT Configuration bits at Reset and can be set or cleared by software.
2. When ON = 1, writes to the WINSIZE[1:0], RMPS[4:0], RMCLK[1:0], SMPS[4:0] and WINDIS are disabled.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON	WINSIZE[1:0]		RMPS[4:0]				
Access	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg
Reset								

Bit	7	6	5	4	3	2	1	0
	RMCLK[1:0]		SMPS[4:0]					WINDIS
Access	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg	R/W-cfg
Reset								

Bit 15 – ON On bit(2)

Value	Description
1	Enables the WDT.
0	Disable and reset WDT.

Bits 14:13 – WINSIZE[1:0] Size of Watchdog Window bits

Value	Description
11	Window size is 25% (Timer Count > 11xxx...xxxxx for the timer to be cleared).
10	Window size is 37.5% (Timer Count > 101xx...xxxxx for the timer to be cleared).
01	Window size is 50% (Timer Count > 1xxxx...xxxxx for the timer to be cleared).
00	Window size is 75% (Timer Count > 01xxx...xxxxx for the timer to be cleared).

Bits 12:8 – RMPS[4:0] Configures the postscaler value for Run Mode Counter bits
Refer to Table 35-2 for the encoding of this bit field.

Bits 7:6 – RMCLK[1:0] Watchdog Timer Run Mode Counter Clock Selection bits

Value	Description
11	LPRC Oscillator
10	BFRC Oscillator
01	Reserved

Value	Description
00	$F_{OSC}/4$

Bits 5:1 – SMPS[4:0] Configures the Postscaler Value for Sleep Mode Counter bits
Refer to Table 35-2 for the encoding of this bit field.

Bit 0 – WINDIS Watchdog Window Mode Disable bit

Value	Description
1	Disables Window mode.
0	Enables Window mode.

35.4. Operation

35.4.1. Watchdog Clock Inputs

The Sleep Mode Counter always uses the 32 kHz clock source, which is typically connected to a BFRC/244 oscillator. The Run Mode Counter will use the clock source selected by the RMCLK[1:0] in (WDTCON[7:6]).

Note: The BFRC/244 oscillator is automatically enabled whenever it is being used as a WDT clock source and the WDT is enabled.

35.4.2. Watchdog Time-out Period Selection

The WDT time-out period is selected by postscaler dividers.

The WDT time-out period can be calculated using Equation 35-1, and WDT time-out period with window option can be calculated using Equation 35-2. The WDT period configurations for WDT time-out period are provided in Table 35-2.

Equation 35-1. Watchdog Timer Time-out Period

$T_{WTO} = (N1) \times (N2) \times (T_{CLK})$

Where:

$N1 = 32$

$N2$ = Postscaler divider ratio (see Table 35-2)

T_{CLK} = WDT clock source

Table 35-2. WDT Period Configurations

RMPS[4:0]or SMPS[4:0]	Post-scale Ratio	Time-out Period		
		32 kHz	8 MHz	50 MHz
00000	1	1 ms	4 µs	640 ns
00001	2	2 ms	8 µs	1280 ns
00010	4	4 ms	16 µs	2.56 µs
00011	8	8 ms	32 µs	5.12 µs
00100	16	16 ms	64 µs	10.24 µs
00101	32	32 ms	128 µs	20.48 µs
00110	64	64 ms	256 µs	41.92 µs
00111	128	128 ms	512 µs	81.92 µs
01000	256	256 ms	1.02 ms	163.8 µs
01001	512	512 ms	2.05 ms	327.6 µs
01010	1024	1.024s	4.1 ms	655.4 µs
01011	2048	2.048s	8.2 ms	1310.8 µs

Table 35-2. WDT Period Configurations (continued)

RMPS[4:0]or SMPS[4:0]	Post-scale Ratio	Time-out Period		
		32 kHz	8 MHz	50 MHz
01100	4096	4.096s	16.4 ms	2.62 ms
01101	8192	8.192s	32.8ms	5.24 ms
01110	16384	16.384s	65.5 ms	10.48 ms
01111	32768	32.768s	131.1 ms	21 ms
10000	65536	00:01:05	262.1 ms	41.94 ms
10001	131072	00:02:11	524.3 ms	83.8 ms
10010	262144	00:04:22	1.05s	167.8 ms
10011	524288	00:08:44	2.1s	335.4 ms
10100	1048576	00:17:28	4.2s	671 ms
10101	2097152	00:34:57	8.4 s	1342.2 ms
10110	4194304	01:09:54	16.8s	2.68s
10111	8388608	02:19:48	33.5s	5.36s
11000	16777216	04:39:37	00:01:07	10.74s
11001	33554432	09:19:14	00:02:14	21.4s
11010	67108864	18:38:28	00:04:28	43s
11011	134217728	1day, 13:16:57	00:08:56	85.8s
11100	268435456	3days, 02:33:55	00:17:53	00:05:40
11101	536870912	6days, 05:07:50	00:35:47	00:11:24
11110	1073741824	12days, 10:15:41	01:11:35	00:22:52
11111	2147483648	24days, 20:31:23	02:23:10	00:45:48

35.4.3. Enabling and Disabling the Watchdog Timer

The Reset value of each bit in the WDTCON SFR can be determined by the FWDT configuration register to allow the WDT to be either disabled or enabled and preconfigured at device start-up.

The Watchdog Timer is enabled or disabled by the WDT ON(WDTCON[15]) bit. When the ON bit = 1, then nothing else can be written. Additionally, writes to the WDTCON register can also be locked using PACCON2[WDWTCONWR] to avoid accidental WDT writes.

If the ON bit is cleared in the WDTCON register, the WDT is disabled and reset, and WDTCON SFR modification is allowed.

The ON bit in WDTCON mirrors the ON bit in the FWDT Configuration register on device Reset. The ON bit allows the user application to enable the WDT for critical code segments and disable the WDT during non-critical segments for maximum power savings.

The WDT flag bit, WDTO (RCON[4]), is not cleared automatically after a WDT time-out. To detect subsequent WDT events, the flag must be cleared in software.

Note: The Run mode WDT is expected to stall when the device is performing any Flash operation while executing the code from the same Flash partition, and it is expected to resume its Run mode counter once the Flash operation is complete.

35.4.4. Watchdog Timer Window

The Watchdog Timer has an optional Windowed mode that enables the WINDIS = 0 bit (WDTCON[0]). In the Windowed mode (WINDIS= `0`), the WDT must be cleared only within the allowed window interval of the Watchdog time-out period, as shown in Figure 35-2. There is also an option to select a WDT window where the WDT should be cleared based on the Watchdog Window Select bits (WINSIZE[1:0]) in the WDTCON register. The bit settings are as follows:

- `11` = WDT Allowed Window is 25% of the WDT period.
- `10` = WDT Allowed Window is 37.5% of the WDT period.
- `01` = WDT Allowed Window is 50% of the WDT period.
- `00` = WDT Allowed Window is 75% of the WDT period.

If the Watchdog Timer is cleared before the allowed window, a system Reset is generated immediately.

The Windowed mode is useful for resetting the device during unexpectedly quick or slow execution of a critical portion of the code.

Equation 35-2. Watchdog Timer Time-out Period with Window Option

$T_{DW} = T_{WTO} - T_{AW}$

$T_{AW} = (N1) \times (N2) \times (T_{CLK}) \times (WINSIZE[1:0])$

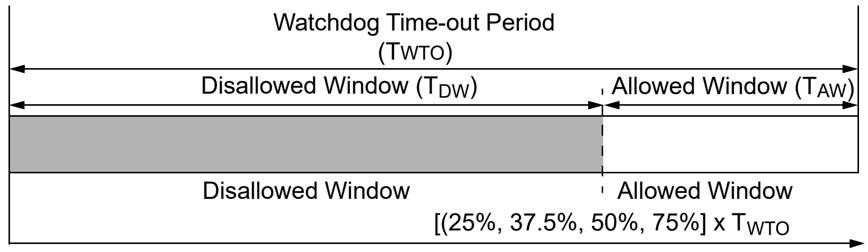
Where:

$N1$ = 32

$N2$ = Postscaler divider ratio (see Table 35-2)

T_{CLK} = WDT clock source

Figure 35-2. Windowed Watchdog Timer



35.4.5. Postscaler

A variable postscaler divides the WDT prescaler output and allows for a wide range of time-out periods. User software may select the postscaler setting for the WDT in Run and Sleep modes. The postscaler is controlled by the RMPS[4:0] bits (WDTCON[12:8]) in Run Mode and SMPS[4:0] bits (WDTCON[5:1]) in Sleep/Idle Mode, which allows the selection of 32 settings, from 1:1 to 1:2147483648. Using the postscaler, time-out periods ranging from 1 ms to 24 days can be achieved.

The RMPS[4:0] selects the postscaler value for the Run mode counter. The RMPS[4:0] bits can only be changed when the ON(WDTCON[15]) bit is cleared.

The SMPS[4:0] selects the postscaler value for the Sleep mode counter. The SMPS bits can only be changed when the ON bit is cleared.

The WDT module time-out period is directly related to the frequency of the WDT clock source. The nominal frequency of the clock source is device dependent. The frequency may vary as a function of the device operating voltage and temperature.

The available clock sources for Run mode are device dependent.

35.5. Watchdog Timer Reset

The Watchdog Timer is reset, and the Run mode WDT counter is cleared by any of the following circumstances:

- Any device Reset
- Execution of the `clrwdt` instruction.
- Disabling WDT by clearing the ON bit of WDTCON register.
- Execution of a `DEBUG` command.

The Sleep mode WDT counter is reset upon entry into Sleep/Idle.

Note: The Run mode WDT is not reset when the device enters a Power-Saving mode.

35.5.1. WDT Time-out in Run Mode

When the WDT times out in Run mode, a device Reset/NMI is generated. Firmware can determine if the cause of the Reset was the WDT time-out in Run mode by testing the WDTO bit in RCON. WDT can be controlled to raise a device Reset or WDT generic trap on WDT time-out by modifying WDTRSTEN (FWDT[16]) bit.

35.5.2. WDT Time-out in Sleep/Idle Mode

When the WDT module times out in Power-Saving mode, it wakes the device and the WDT Run mode resumes counting.

WDT Sleep/Idle mode event can also trigger a WDT interrupt on wake-up from Sleep/Idle mode if IEC0[WDTIE] is enabled.

If the WDT Interrupt Enable bit, WDTIE, is cleared, an interrupt will not be generated on WDT wake-up event. However, the WDTIF bit will still be set if an interrupt condition occurs. The user can clear the interrupt in the Interrupt Service Routine (ISR) by clearing WDTIF. See **Interrupt Controller** for more information.

35.5.3. Wake from Power-Saving Mode by a Non-WDT Event

When the device is awakened from a Power-Saving mode by a non-WDT NMI interrupt, the Power-Saving mode WDT is held in Reset and the WDT Run mode continues counting from the pre-power save count value.

35.6. Power-Saving Modes

35.6.1. WDT Operation in Power-Saving Modes

The WDT, if enabled, will continue operation in Sleep mode or Idle mode and can be used to wake the device. This allows the device to remain in Sleep or Idle mode until the WDT expires or another interrupt wakes the device. If the device does not re-enter Sleep or Idle mode following a wake-up, the WDT must be disabled or periodically serviced to prevent a WDT Run mode NMI.

35.6.2. WDT Operation in Sleep Mode

The WDT module may be used to wake the device from Sleep mode. When entering Sleep mode, the WDT Run mode counter stops counting and the Power-Saving mode WDT begins counting from the Reset state until it times out or the device is woken up by an interrupt. When the WDT times out in Sleep mode, the device wakes up, resumes code execution and resumes the Run mode WDT.

35.6.3. WDT Operation in Idle Mode

The WDT module may be used to wake the device from Idle mode. When entering Idle mode, the WDT Run mode counter stops counting and the Power-Saving mode WDT begins counting from the Reset state until it times out or the device is woken up by an interrupt. Once the device wakes up, either by a sleep WDT time-out or by any interrupt, the device resumes the code execution and the Run mode WDT timer.

35.6.4. Time Delays During Wake-up

There will be a time delay between the WDT event in Sleep and the beginning of code execution. The duration of this delay consists of the start-up time for the oscillator in use. Unlike a wake-up from Sleep mode, there are no time delays associated with wake-up from Idle mode. The system clock is running during Idle mode; therefore, no start-up delays are required at wake-up.

35.7. WDT Generic Trap

WDT Event occurs due to counter overflow or an attempt to clear the WDT in a windowed interval. The FWDT[RSTEN] bit in the configuration fuse can be set to enable a device Reset or be cleared to raise a generic trap in case of a WDT event. When the FWDT[RSTEN] bit is cleared, the WDT event results in a (INTCON5[WDT]) WDT generic trap. In this case, the WDT counter resets and starts counting again soon after the WDT event has occurred.

35.8. WDT Sample Configuration

35.8.1. Run WDT Configuration

Example 35-1 shows a code example to configure Run WDT for 1.024s with LPRC as its clock source.

Example 35-1. Run WDT Configuration

```
//code example to configure Run WDT for 1.024sec
int main()
{

WDTCONbits.RMCLK = 3;          // LPRC as Run WDT Clock
WDTCONbits.RMPS = 10;          // Run Postscaler 1024
WDTCONbits.ON = 1;             // Enable WDT
```

```
}
```

35.8.2. Sleep WDT Configuration

Example 35-2 shows a code example to configure Sleep WDT for 1.024s with LPRC as its default clock source.

Example 35-2. Sleep WDT Configuration Example

```
//code example to configure Sleep WDT for 1.024 sec
int main()
{

WDTCONbits.SMPS = 10;          // Sleep Postscaler 1024
WDTCONbits.ON = 1;             // Enable WDT

}
```

35.8.3. Configuring WDT Via Config Fuse

Example 35-3 shows a code example to configure Run and Sleep WDT for 1.024 sec with LPRC as a Run WDT clock source via config fuse.

Example 35-3. WDT Configuration Example

```
//code example to configure Run and Sleep WDT for 1.024 sec via config
//fuse
// FWDT
#pragma config FWDT_WINDIS = OFF       // Watchdog Timer Window Disable bit (Watchdog Timer operates in
Window mode)
#pragma config FWDT_SWDTMPS = PS1024   // Sleep Mode Watchdog Timer Post Scaler select bits (1:1024)
#pragma config FWDT_RCLKSEL = BFRC256  // Watchdog Timer Clock select bits (WDT Run Mode uses BFRC:244)
#pragma config FWDT_RWDTPS = PS1024    // Run Mode Watchdog Timer Post Scaler select bits (1:1024)
#pragma config FWDT_WDTWIN = WIN25     // Watchdog Timer Window Size Select bits (WDT Window is 25% of
WDT period)
#pragma config FWDT_WDTEN = SW         // Watchdog Timer Enable bit (WDT is controlled by software, use
WDTCON.ON bit)
#pragma config FWDT_WDTRSTEN = ON      // Watchdog Timer Reset Enable bit (WDT event generates a reset)
```

35.8.4. Clearing Watchdog Timer

Example 35-4 shows a code example to clear the Run WDT counter.

Notes:

1. If enabled, Run WDT should be periodically serviced (must be disabled within time-out) to prevent WDT Run mode NMI/Reset.
2. Processor takes a few cycles to execute `ClrWdt` instruction; if this is not taken into account, Run WDT will not be cleared reliably.
3. When Window mode is enabled, use caution when clearing WDT because this can only be done in the allowed Window. Clearing WDT in a disallowed window causes the Run WDT NMI/Reset.

Example 35-4. Clear WDT after 300ms

```
//code example to clear Run Counter after 300ms with WDT timeout as 1.024sec
 int main()
{
WDTCONbits.RMPS = 10;     // Run Postscaler 1024
 WDTCONbits.ON = 1;       // Enable WDT
__delay_ms(300);          // 300ms delay
ClrWdt();                 // Clears Run Counter after 300ms delay

}
```

36. Deadman Timer (DMT)

The Deadman Timer (DMT) module enables users to monitor the health of their application software by requiring periodic timer interrupts within a user-specified timing window. The DMT module is a synchronous counter that, when enabled, counts instruction fetches and is able to cause a soft trap if the DMT counter is not cleared within a set number of instructions. The DMT is typically connected to the system clock that drives the processor (T_{CY}). The user specifies the timer time-out value and a mask value that specifies the range of the window, which is the range of counts that is not considered for the comparison event.

Some of the key features of this module are:

- Software Enabled
- User-Configurable Time-out Period or Instruction Count
- Two Instruction Sequences to Clear Timer
- 32-Bit Configurable Window to Clear Timer

Table 36-1. DMT Summary

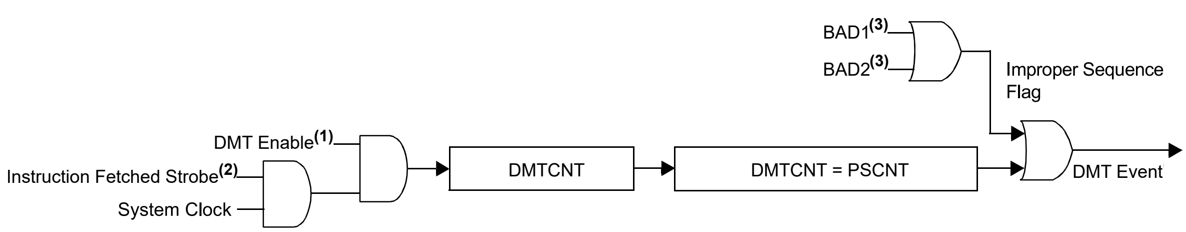
DMT Module Instances	DMT Output	Clock Source	Peripheral Bus Speed
1	Soft Trap	Slow Speed Peripheral Clock	Slow (1:4 CPU Clock)

36.1. Architectural Overview

The primary function of the Deadman Timer (DMT) is to interrupt the processor in the event of a software malfunction. The DMT, which works on the system clock, is a free-running instruction fetch timer, which is clocked whenever an instruction fetch occurs, until a count match occurs. Instructions are not fetched when the processor is in Sleep mode.

DMT can be enabled in the Configuration fuse or by software in the DMTCON register by setting the ON bit. The DMT module consists of a 32-bit counter, the read-only DMTCNT register and a time-out count match value as specified by the PSCNT register. Whenever the count match occurs, a DMT event will occur, which is a soft trap. A DMT is typically used in mission-critical and safety-critical applications where any single failure of the software functionality and sequencing must be detected. Figure 36-1 shows a block diagram of the Deadman Timer module.

Figure 36-1. Deadman Timer Block Diagram



Notes:

1. The DMT is clocked whenever the instructions are fetched by the processor using a system clock. For example, after executing a `GOTO` instruction (which uses four instruction cycles), the DMT counter will be incremented only once.
2. BAD1 and BAD2 are the improper sequence flags.
3. A DMT event is a non-maskable soft trap.

36.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x3A00	DMTCON	31:24								
		23:16								
		15:8	ON							
		7:0								
0x3A04	DMTPRECLR	31:24								
		23:16								
		15:8	STEP1[7:0]							
		7:0								
0x3A08	DMTCLR	31:24								
		23:16								
		15:8								
		7:0	STEP2[7:0]							
0x3A0C	DMTSTAT	31:24								
		23:16								
		15:8								
		7:0	BAD1	BAD2	EVENT					WINOPN
0x3A10	DMTCNT	31:24	COUNTER[31:24]							
		23:16	COUNTER[23:16]							
		15:8	COUNTER[15:8]							
		7:0	COUNTER[7:0]							
0x3A14	PSCNT	31:24	PSCNT[31:24]							
		23:16	PSCNT[23:16]							
		15:8	PSCNT[15:8]							
		7:0	PSCNT[7:0]							
0x3A18	PSINTV	31:24	PSINTV[31:24]							
		23:16	PSINTV[23:16]							
		15:8	PSINTV[15:8]							
		7:0	PSINTV[7:0]							
0x3A1C	PPPC	31:24								
		23:16								
		15:8	NMISTEP1[7:0]							
		7:0								
0x3A20	PPC	31:24								
		23:16								
		15:8								
		7:0	NMISTEP2[7:0]							

36.2.1. Deadman Timer Control Register

Name: DMTCON
Offset: 0x3A00
Property: Read-only

Legend: S = Settable bit

Note:

1. The ON bit is a write once bit, and writes to the PSCNT and PSINTV registers, which are not enabled when ON = 1.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	ON							
Access	R/S							
Reset	0							

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bit 15 – ON On bit(1)

Value	Description
1	Enables the Deadman Timer.
0	The DMT is reset at a RESET instruction.

36.2.2. Deadman Timer Preclear Register

Name: DMTPRECLR
Offset: 0x3A04
Property: R/W

Notes:
1. STEP1[15:8] (DMTPRECLR[15:8]) bits are cleared when a DMT Reset event occurs.
2. STEP1 bits are also cleared if the STEP2 (DMTCLR[7:0]) bits are loaded with the correct value in the correct sequence.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	STEP1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 15:8 – STEP1[7:0] Preclear Enable bits[1,2]

Value	Description
`01000000b`	Enables the Deadman Timer preclear.
`All other write patterns`	Sets the BAD1 flag.

36.2.3. Deadman Timer Clear Register

Name: DMTCLR
Offset: 0x3A08
Property: R/W

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	STEP2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – STEP2[7:0] Clear Enable bits

Value	Description
`00001000b`	Clears STEP1 (DMTPRECLR[15:8]), STEP2 (DMTCLR[7:0]) and the Deadman Timer (STEP2) if, and only if, preceded by correct loading of the Preclear Enable bits (STEP1) in the correct sequence. The write to the STEP2 bits field may be verified by reading DMTCNT and observing the counter being reset.
`All other write patterns`	Sets BAD2 Flag; the value in the STEP1 bits will remain unchanged and the new value being written to the STEP2 bits will be captured.

36.2.4. Deadman Timer Status Register

Name: DMTSTAT
Offset: 0x3A0C
Property: R/W

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	BAD1	BAD2	EVENT					WINOPN
Access	R	R	R					R
Reset	0	0	0					0

Bit 7 – BAD1 BAD STEP1 Value Detect bit
This bit is cleared by a Reset or a successful post NMI clear sequence bit.

Value	Description
1	Incorrect STEP1[7:0] value was detected.
0	Incorrect STEP1[7:0] value was not detected.

Bit 6 – BAD2 BAD STEP2 Value Detect bit
This bit is cleared by a Reset or a successful post NMI clear sequence bit.

Value	Description
1	Incorrect STEP2[7:0] value was detected.
0	Incorrect STEP2[7:0] value was not detected.

Bit 5 – EVENT Deadman Timer Event bit
This bit is cleared by a Reset or a successful post NMI clear sequence bit.

Value	Description
1	DMT counter expired or a BAD1 or BAD2 step occurred.
0	No errors detected.

Bit 0 – WINOPN Deadman Timer Clear Window bit

Value	Description
1	Deadman Timer clear window is open.
0	Deadman Timer clear window is not open.

36.2.5. Deadman Timer Count Register

Name: DMTCNT
Offset: 0x3A10
Property: R/W

Bit	31	30	29	28	27	26	25	24
	COUNTER[31:24]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	COUNTER[23:16]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	COUNTER[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	COUNTER[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – COUNTER[31:0] Read Current Contents of DMT Counter bits

36.2.6. Deadman Timer Counter Limit Register

Name: PSCNT
Offset: 0x3A14
Property: R/W

Bit	31	30	29	28	27	26	25	24
	PSCNT[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSCNT[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSCNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSCNT[31:0] DMT Instruction Count Limit Value bits
This register sets the maximum value of the DMT counter. If the counter counts past this limit value, a DMT soft trap is generated.
This register can be written only when the ON bit is zero.

36.2.7. DMT Interval Status Register

Name: PSINTV
Offset: 0x3A18
Property: R/W

Bit	31	30	29	28	27	26	25	24
	PSINTV[31:24]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	23	22	21	20	19	18	17	16
	PSINTV[23:16]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	PSINTV[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	PSINTV[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 31:0 – PSINTV[31:0] DMT Window Interval bits
The DMT Interval Status register specifies when the DMT counter Clear operation may be performed. The DMTCNT bits must be greater than the PSINTV bits before the Clear operation can be initiated. This register can be written only when the ON bit is zero.

36.2.8. DMT NMI Preclear Register

Name: PPPC
Offset: 0x3A1C
Property: R/W

Note:

1. Bits[15:8] are cleared when a DMT Reset event occurs. NMISTEP1 bits are also cleared if NMISTEP2 (PPC[7:0]) bits are loaded with the correct value in the correct sequence.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
	NMISTEP1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Access								
Reset								

Bits 15:8 – NMISTEP1[7:0] NMI Post-Processing Preclear Enable bits[1]

Value	Description
`01000001b`	Enables the NMI preclear (NMI_STEP1).
`All other write patterns`	This register remains unchanged and the instruction writing to it is considered to be unsuccessful.

36.2.9. DMT NMI Clear Register

Name: PPC
Offset: 0x3A20
Property: R/W

Note:
1. Bits[7:0] are also cleared when a DMT Reset event occurs. NMISTEP1 and NMISTEP2 are also cleared if NMISTEP2 is loaded with the correct value during the Pre-DMT Event Clear sequence.

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
Access								
Reset								

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
	NMISTEP2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – NMISTEP2[7:0] NMI Clear DMT Event Enable bits[(1)]

Value	Description
`10001000b`	If write pattern has been preceded by correct execution of PPPC NMI preclear instruction, the DMT event is cleared.
`All other write patterns`	This register remains unchanged and the instruction writing to it is considered to be unsuccessful.

36.3. Operation

36.3.1. Enabling and Disabling the DMT Module

The DMT module can be enabled by setting the ON (DMTCON[15]) bit through software by writing to the DMTCON register. The ON bit is a Write-Once bit, and disabling the DMT in software is not allowed.

36.3.2. DMT Count Selection

The Deadman Timer count is set by the PSCNT register bits. The current DMT count value can be obtained by reading the DMTCNT register.

The PSCNT[31:0] bits allow the software to read the maximum count selected for the Deadman Timer. Whenever the DMT event occurs, the user can always compare to see whether the current counter value in the DMTCNT register is equal to the value of the PSCNT register, which holds the maximum count value.

The PSINTV[31:0] bits allow the software to read the DMT window interval value. So whenever the DMT current counter value in DMTCNT reaches the value of PSINTV, the window interval opens so that the user can insert the clear sequence to the STEP2 bits, which causes the DMT to reset.

The initial DMT state has the timer cleared; the DMTPRECLR and BAD1/BAD2 flags are cleared. When reset, the DMT clears the DMTSTAT register. When activated, the DMT begins to count whenever an instruction is fetched while clocks are running.

36.3.3. DMT Count Windowed Interval

The DMT module has a Windowed Operation mode. The PSINTV register will set the window interval value. In Windowed mode, software can clear the DMT only when the counter is in its final window before a count match occurs. That is, if the DMT counter value (DMTCNT) is greater than or equal to the value written to the window interval value (PSINTV), then only the clear sequence can be inserted into the DMT module.

The window interval is open when: (COUNTER) ≥ PSINTV. If the DMT is cleared before the allowed window, a Deadman Timer soft trap is immediately generated.

Note: Care should be exercised when selecting the window interval, such that the interval does not begin after the time-out instruction count specified by PSCNT[31:0].

36.3.4. Resetting the DMT

The DMT can be reset in three ways:

1. System Reset.
2. Writing an ordered sequence to the DMTPRECLR and DMTCLR registers. Refer to Clearing Pre-DMT Event.
3. Following an expired DMT count exception, to clear the DMT event, two instructions must be executed in sequence: the PPPC followed by the PPC. Refer to Clearing Post-DMT Event.

36.3.4.1. Clearing Pre-DMT Event

Clearing the DMT counter value requires a special sequence of operations:

1. The STEP1[15:8] bits in the DMTPRECLR register must be written as '01000000' (0x40).
2. The STEP2[7:0] bits in the DMTCLR register must be written as '00001000' (0x08). This can only be done if preceded by Step 1 and the DMT is in the open window interval (WINOPN = 1).

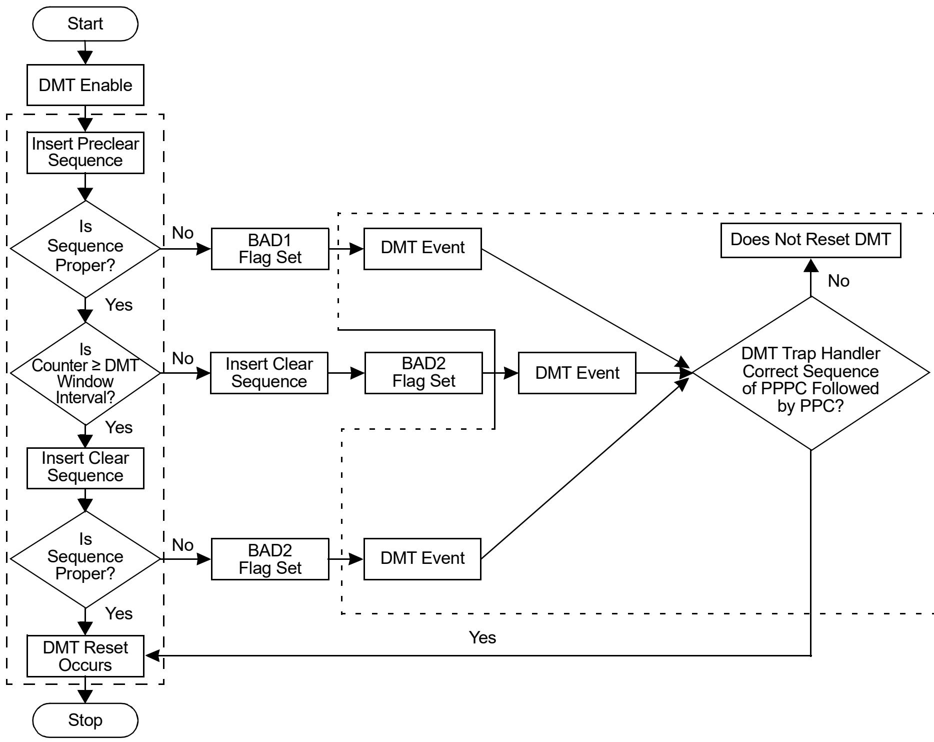
Once these values are written, the DMT counter will be cleared to zero. The DMTPRECLR, DMTCLR and DMTSTAT registers' values will also be cleared to zero. Refer to the pseudocode in Example 36-1 for the correct sequence of operation. This is the normal use of the module while clearing the DMT.

If any value other than 0x40 is written to the STEP1 bits, the BAD1 bit in the DMTSTAT register will be set and it causes a DMT event to occur. Any value other than 0x08 written to the STEP2 bits will cause the BAD2 bit to be set in the DMTSTAT register. Also, if Step 2 is not preceded by Step 1, or Step 2 is not carried out in the open window interval, it causes the BAD2 flag to be set. Immediately, a DMT event will occur.

Example 36-1. Clearing DMT Before DMT Count Match Occurs

```
//Assume PSCNT = 0x0000FFFF and PSINTV = 0x000000FF
//When DMTCNT >= PSINTV WINOPN gets set

DMTPRECLR=0b01000000;             // Valid DMTPRECLR Value
while(DMTSTAT & 0x0001!=0x0001); //Wait till DMT Clear Window is open
                                   i.e WINOPN=1
DMTCLR=0b00001000;                // Valid DMTCLR Value
```

Figure 36-2. Flowchart for Resetting the DMT



36.3.4.2. Clearing Post-DMT Event

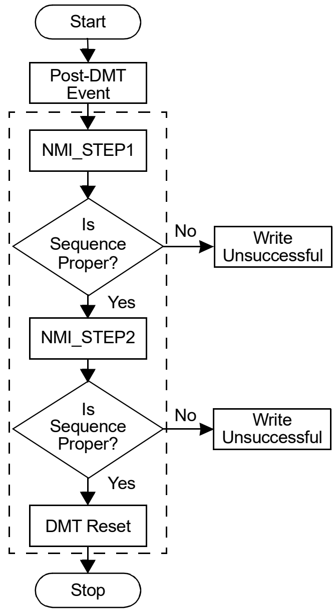
The DMT NMI will cause the processor to execute a trap handler (ISR) that will decide what to do. The user may decide to reset the DMT through the Post-Clearing (PPPC followed by PPC) mechanism, or the user can execute a `RESET` instruction to reset the system in the trap handler.

Clearing the DMT counter post-expiry requires a special sequence of operations:

1. The NMI_STEP1[7:0] bits in the PPPC register must be written as '`01000000b`'.
2. The NMI_STEP2[7:0] bits in the PPC register must be written as '`10001000b`'. This can only be done if preceded by NMI_STEP1.

If any value other than '`01000000b`' is written to the NMI_STEP1 bits, this register remains unchanged and the instruction writing to it is considered to be unsuccessful. Any value other than '`10001000b`' written to the NMI_STEP2 bits renders the operation unsuccessful.

Figure 36-3. Flowchart for Clearing Post-DMT Event



36.3.5. DMT Generic Trap

A DMT event due to counter overflow or a BAD1/BAD2 event results in a DMT generic trap. This being a persistent event, DMT needs to be reset using the PPPC/PPC registers within the ISR before exiting the trap routine.

36.3.6. DMT Operation in Power-Saving Modes

As the DMT module is only incremented by instruction fetches, the count value will not change when the core is inactive. The DMT module remains inactive in Sleep and Idle modes. As soon as the device wakes up from Sleep or Idle, the DMT counter again starts incrementing.

37. Device Power-Saving Modes

The dsPIC33AK256MPS306 family devices provide the ability to manage power consumption by selectively managing clocking to the CPU and the peripherals. In general, a lower clock frequency and a reduction in the number of peripherals being clocked constitutes lower consumed power.

This section describes the power-saving modes implemented in dsPIC33AK256MPS306 devices. The dsPIC33AK256MPS306 device family offers a number of built-in capabilities that allow user applications to select the best balance of performance and low-power consumption.

The power-saving features are:

- Instruction-Based Power-Saving Modes
- Peripheral Module Disable (PMD)

Combinations of these methods can be used to selectively tailor an application's power consumption while still maintaining critical application features, such as timing-sensitive communications.

37.1. Architectural Overview

Reducing the system clock frequency results in power saving that is roughly proportional to the frequency reduction. The dsPIC33A devices provide an on-the-fly clock switching feature that allows the user application to optimize power consumption by dynamically changing the system clock frequency.

There are two ways to reduce power consumption in dsPIC33A devices:

1. Instruction-based power-saving modes which include Sleep and Idle.
   - **Sleep Mode:** In Sleep mode, the CPU, the system clock source and the peripherals that operate on the system clock source are disabled. This is the lowest power mode for the device. Optionally, the peripherals can operate in Sleep mode using specific clock sources. Please refer to the individual peripheral chapter for descriptions on behavior in Sleep mode. The SLEEP status flag bit in the Reset Control register (RCON[3]) is set when the device enters Sleep mode.
   - **Idle Mode:** In Idle mode, the CPU is disabled, but the system clock source continues to operate. The peripherals continue to operate but can optionally be disabled. The IDLE status flag bit in the Reset Control register (RCON[2]) is set when the device enters Idle mode.

   The SLEEP and IDLE status bits are cleared on Power-on Reset (POR) and Brown-out Reset (BOR). These bits can also be cleared in software. For more information on Resets, refer to **Resets**.
2. The peripherals can be selectively disabled using the Peripheral Module Disable (PMD) bit of the corresponding peripheral.

Notes:

1. `SLEEP_MODE` and `IDLE_MODE` are constants defined in the assembler include file for the selected device.
2. Sleep mode does not change the state of the I/O pins.
3. Execution of the `PWRSAV` instruction is ignored while any of the NVM operations are in progress.

37.2. Register Summary

Offset	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x31B0	RCON	31:24								
		23:16		VREG4R	VREG3R	VREG2R		BUCKR		PWRMONR
		15:8							CM	
		7:0	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR
0x31B4 ... 0x3A43	Reserved									
0x3A44	PMD1	31:24					SPI4MD	SPI3MD	SPI2MD	SPI1MD
		23:16					U4MD	U3MD	U2MD	U1MD
		15:8			T3MD	T2MD	T1MD			CCP9MD
		7:0	CCP8MD	CCP7MD	CCP6MD	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD
0x3A48	PMD2	31:24					CLC4MD	CLC3MD	CLC2MD	CLC1MD
		23:16								PDCMD
		15:8								DMAMD
		7:0			SENT2MD	SENT1MD		I2C3MD	I2C2MD	I2C1MD
0x3A4C	PMD3	31:24								
		23:16		C1MD	ITCMD			ADC3MD	ADC2MD	ADC1MD
		15:8								OPAMD
		7:0				BISS1MD	QEI4MD	QEI3MD	QEI2MD	QEI1MD
0x3A50	PMD4	31:24								
		23:16	IOIM8MD	IOIM7MD	IOIM6MD	IOIM5MD	IOIM4MD	IOIM3MD	IOIM2MD	IOIM1MD
		15:8	CM4MD	CM3MD	CM2MD	CM1MD			PWM2MD	PWM1MD
		7:0	UREFMD		CRYMD			DMTMD	CRCMD	PTGMD

37.2.1. Reset Control Register

Name: RCON
Offset: 0x31B0
Property: R/W

Legend: R = Readable bit; C = Clearable bit; HS = Hardware Settable bit

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		VREG4R	VREG3R	VREG2R		BUCKR		PWRMONR
Access		R/C/HS	R/C/HS	R/C/HS		R/C/HS		R/C/HS
Reset		0	0	0		0		0

Bit	15	14	13	12	11	10	9	8
							CM	
Access							R/C/HS	
Reset							0	

Bit	7	6	5	4	3	2	1	0
	EXTR	SWR		WDTO	SLEEP	IDLE	BOR	POR
Access	R/C/HS	R/C/HS		R/C/HS	R/C/HS	R/C/HS	R/C/HS	R/C/HS
Reset	0	0		0	0	0	1	1

Bit 22 – VREG4R VREG Voltage Regulator 4 Flag bit

Value	Description
1	Voltage Domain 4 lost voltage regulation.
0	Voltage Regulation in Domain 4 has not been lost.

Bit 21 – VREG3R VREG Voltage Regulator 3 Flag bit

Value	Description
1	Voltage Domain 3 lost voltage regulation.
0	Voltage Regulation in Domain 3 has not been lost.

Bit 20 – VREG2R VREG Voltage Regulator 2 Flag bit

Value	Description
1	Voltage Domain 2 lost voltage regulation.
0	Voltage Regulation in Domain 2 has not been lost.

Bit 18 – BUCKR Buck Converter Flag bit

Value	Description
1	Buck regulation lost voltage regulation.
0	Buck regulation has not been lost.

Bit 16 – PWRMONR Power Monitor Reset bit

Value	Description
1	A reset event from the Power Monitor has occurred

Value	Description
0	A reset event from the power monitor has not occurred

Bit 9 – CM Configuration Mismatch Flag bit

Value	Description
1	A Configuration mismatch Reset has occurred.
0	A Configuration mismatch Reset has not occurred

Bit 7 – EXTR External Reset ($\overline{\text{MCLR}}$) Pin bit

Value	Description
1	A master clear (pin) Reset has occurred.
0	A master clear (pin) Reset has not occurred.

Bit 6 – SWR Software `RESET` (Instruction) Flag bit

Value	Description
1	A `RESET` instruction has been executed.
0	A `RESET` instruction has not been executed.

Bit 4 – WDTO Watchdog Timer Time-out Flag bit

Value	Description
1	Device Reset has occurred due to a WDT time-out.
0	WDT time-out has not occurred.

Bit 3 – SLEEP Wake-up from Sleep Flag bit

Value	Description
1	Device has been in Sleep mode.
0	Device has not been in Sleep mode.

Bit 2 – IDLE Wake-up from Idle Flag bit

Value	Description
1	Device has been in Idle mode.
0	Device has not been in Idle mode.

Bit 1 – BOR Brown-out Reset Flag bit

Value	Description
1	A Brown-out Reset has occurred. Set by hardware at detection of a BOR event.
0	A Brown-out Reset has not occurred.

Bit 0 – POR Power-on Reset Flag bit

Value	Description
1	A Power-on Reset has occurred. Set by hardware at detection of a POR event.
0	A Power-on Reset has not occurred.

37.2.2. Peripheral Module Disable 1 Register

Name: PMD1
Offset: 0x3A44

Bit	31	30	29	28	27	26	25	24
					SPI4MD	SPI3MD	SPI2MD	SPI1MD
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
					U4MD	U3MD	U2MD	U1MD
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	15	14	13	12	11	10	9	8
			T3MD	T2MD	T1MD			CCP9MD
Access			R/W	R/W	R/W			R/W
Reset			0	0	0			0

Bit	7	6	5	4	3	2	1	0
	CCP8MD	CCP7MD	CCP6MD	CCP5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 27 – SPI4MD SPI4 Module Disable bit

Value	Description
1	SPI4 module is disabled.
0	SPI4 module is enabled.

Bit 26 – SPI3MD SPI3 Module Disable bit

Value	Description
1	SPI3 module is disabled.
0	SPI3 module is enabled.

Bit 25 – SPI2MD SPI2 Module Disable bit

Value	Description
1	SPI2 module is disabled.
0	SPI2 module is enabled.

Bit 24 – SPI1MD SPI1 Module Disable bit

Value	Description
1	SPI1 module is disabled.
0	SPI1 module is enabled.

Bit 19 – U4MD UART4 Module Disable bit

Value	Description
1	UART4 module is disabled.
0	UART4 module is enabled.

Bit 18 – U3MD UART3 Module Disable bit

Value	Description
1	UART3 module is disabled.
0	UART3 module is enabled.

Bit 17 – U2MD UART2 Module Disable bit

Value	Description
1	UART2 module is disabled.
0	UART2 module is enabled.

Bit 16 – U1MD UART1 Module Disable bit

Value	Description
1	UART1 module is disabled.
0	UART1 module is enabled.

Bit 13 – T3MD Timer 3 Module Disable bit

Value	Description
1	Timer 3 module is disabled.
0	Timer 3 module is enabled.

Bit 12 – T2MD Timer 2 Module Disable bit

Value	Description
1	Timer 2 module is disabled.
0	Timer 2 module is enabled.

Bit 11 – T1MD Timer 1 Module Disable bit

Value	Description
1	Timer 1 module is disabled.
0	Timer 1 module is enabled.

Bit 8 – CCP9MD CCP 9 Module Disable bit

Value	Description
1	CCP 9 module is disabled.
0	CCP 9 module is enabled.

Bit 7 – CCP8MD CCP 8 Module Disable bit

Value	Description
1	CCP 8 module is disabled.
0	CCP 8 module is enabled.

Bit 6 – CCP7MD CCP 7 Module Disable bit

Value	Description
1	CCP 7 module is disabled.
0	CCP 7 module is enabled.

Bit 5 – CCP6MD CCP 6 Module Disable bit

Value	Description
1	CCP 6 module is disabled.
0	CCP 6 module is enabled.

Bit 4 – CCP5MD CCP 5 Module Disable bit

Value	Description
1	CCP 5 module is disabled.
0	CCP 5 module is enabled.

Bit 3 – CCP4MD CCP 4 Module Disable bit

Value	Description
1	CCP 4 module is disabled.
0	CCP 4 module is enabled.

Bit 2 – CCP3MD CCP 3 Module Disable bit

Value	Description
1	CCP 3 module is disabled.
0	CCP 3 module is enabled.

Bit 1 – CCP2MD CCP 2 Module Disable bit

Value	Description
1	CCP 2 module is disabled.
0	CCP 2 module is enabled.

Bit 0 – CCP1MD CCP 1 Module Disable bit

Value	Description
1	CCP 1 module is disabled.
0	CCP 1 module is enabled.

37.2.3. Peripheral Module Disable 2 Register

Name: PMD2
Offset: 0x3A48

Bit	31	30	29	28	27	26	25	24
					CLC4MD	CLC3MD	CLC2MD	CLC1MD
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit	23	22	21	20	19	18	17	16
								PDCMD
Access								R/W
Reset								0

Bit	15	14	13	12	11	10	9	8
								DMAMD
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
			SENT2MD	SENT1MD		I2C3MD	I2C2MD	I2C1MD
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit 27 – CLC4MD CLC 4 Module Disable bit

Value	Description
1	CLC 4 module is disabled.
0	CLC 4 module is enabled.

Bit 26 – CLC3MD CLC 3 Module Disable bit

Value	Description
1	CLC 3 module is disabled.
0	CLC 3 module is enabled.

Bit 25 – CLC2MD CLC 2 Module Disable bit

Value	Description
1	CLC 2 module is disabled.
0	CLC 2 module is enabled.

Bit 24 – CLC1MD CLC 1 Module Disable bit

Value	Description
1	CLC 1 module is disabled.
0	CLC 1 module is enabled.

Bit 16 – PDCMD DAC Module Disable bit

Value	Description
1	DAC module is disabled.
0	DAC module is enabled.

Bit 8 – DMAMD DMA Module Disable bit

Value	Description
1	DMA3 module is disabled.
0	DMA3 module is enabled.

Bit 5 – SENT2MD SENT 2 Module Disable bit

Value	Description
1	SENT 2 module is disabled.
0	SENT 2 module is enabled.

Bit 4 – SENT1MD SENT 1 Module Disable bit

Value	Description
1	SENT 1 module is disabled.
0	SENT 1 module is enabled.

Bit 2 – I2C3MD I^2C 3 Module Disable bit

Value	Description
1	I^2C 2 module is disabled.
0	I^2C 2 module is enabled.

Bit 1 – I2C2MD I^2C 2 Module Disable bit

Value	Description
1	I^2C 2 module is disabled.
0	I^2C 2 module is enabled.

Bit 0 – I2C1MD I^2C 1 Module Disable bit

Value	Description
1	I^2C 1 module is disabled.
0	I^2C 1 module is enabled.

37.2.4. Peripheral Module Disable 3 Register

Name: PMD3
Offset: 0x3A4C

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
		C1MD	ITCMD			ADC3MD	ADC2MD	ADC1MD
Access		R/W	R/W			R/W	R/W	R/W
Reset		0	0			0	0	0

Bit	15	14	13	12	11	10	9	8
								OPAMD
Access								R/W
Reset								0

Bit	7	6	5	4	3	2	1	0
				BISS1MD	QEI4MD	QEI3MD	QEI2MD	QEI1MD
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit 22 – C1MD CAN1 Module Disable bit

Value	Description
1	CAN1 module is disabled.
0	CAN1 module is enabled.

Bit 21 – ITCMD ITC Module Disable bit

Value	Description
1	ITC Module is disabled
0	ITC Module is enabled

Bit 18 – ADC3MD ADC 3 Module Disable bit

Value	Description
1	ADC 3 module is disabled.
0	ADC 3 module is enabled.

Bit 17 – ADC2MD ADC 2 Module Disable bit

Value	Description
1	ADC 2 module is disabled.
0	ADC 2 module is enabled.

Bit 16 – ADC1MD ADC 1 Module Disable bit

Value	Description
1	ADC 1 module is disabled.
0	ADC 1 module is enabled.

Bit 8 – OPAMD Op Amp Module Disable bit

Value	Description
1	OP Amp module is disabled.
0	OP Amp module is enabled.

Bit 4 – BISS1MD BiSS 1 Module Disable bit

Value	Description
1	BiSS 1 module is disabled.
0	BiSS 1 module is enabled.

Bit 3 – QEI4MD QEI 4 Module Disable bit

Value	Description
1	QEI 4 module is disabled.
0	QEI 4 module is enabled.

Bit 2 – QEI3MD QEI 3 Module Disable bit

Value	Description
1	QEI 3 module is disabled.
0	QEI 3 module is enabled.

Bit 1 – QEI2MD QEI 2 Module Disable bit

Value	Description
1	QEI 2 module is disabled.
0	QEI 2 module is enabled.

Bit 0 – QEI1MD QEI 1 Module Disable bit

Value	Description
1	QEI 1 module is disabled.
0	QEI 1 module is enabled.

37.2.5. Peripheral Module Disable 4 Register

Name: PMD4
Offset: 0x3A50

Bit	31	30	29	28	27	26	25	24
Access								
Reset								

Bit	23	22	21	20	19	18	17	16
	IOIM8MD	IOIM7MD	IOIM6MD	IOIM5MD	IOIM4MD	IOIM3MD	IOIM2MD	IOIM1MD
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	CM4MD	CM3MD	CM2MD	CM1MD			PWM2MD	PWM1MD
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit	7	6	5	4	3	2	1	0
	UREFMD		CRYMD			DMTMD	CRCMD	PTGMD
Access	R/W		R/W			R/W	R/W	R/W
Reset	0		0			0	0	0

Bit 23 – IOIM8MD IO Integrity Monitor 8 Disable bit

Value	Description
1	IO integrity monitor 8 module is disabled.
0	IO integrity monitor 8 module is enabled.

Bit 22 – IOIM7MD IO Integrity Monitor 7 Disable bit

Value	Description
1	IO integrity monitor 7 module is disabled.
0	IO integrity monitor 7 module is enabled.

Bit 21 – IOIM6MD IO Integrity Monitor 6 Disable bit

Value	Description
1	IO integrity monitor 6 module is disabled.
0	IO integrity monitor 6 module is enabled.

Bit 20 – IOIM5MD IO Integrity Monitor 5 Disable bit

Value	Description
1	IO integrity monitor 5 module is disabled.
0	IO integrity monitor 5 module is enabled.

Bit 19 – IOIM4MD IO Integrity Monitor 4 Disable bit

Value	Description
1	IO integrity monitor 4 module is disabled.
0	IO integrity monitor 4 module is enabled.

Bit 18 – IOIM3MD IO Integrity Monitor 3 Disable bit

Value	Description
1	IO integrity monitor 3 module is disabled.
0	IO integrity monitor 3 module is enabled.

Bit 17 – IOIM2MD IO Integrity Monitor 2 Disable bit

Value	Description
1	IO integrity monitor 2 module is disabled.
0	IO integrity monitor 2 module is enabled.

Bit 16 – IOIM1MD IO Integrity Monitor 1 Disable bit

Value	Description
1	IO integrity monitor 1 module is disabled.
0	IO integrity monitor 1 module is enabled.

Bit 15 – CM4MD Clock Monitor 4 Disable bit

Value	Description
1	Clock monitor 4 module is disabled.
0	Clock monitor 4 module is enabled.

Bit 14 – CM3MD Clock Monitor 3 Disable bit

Value	Description
1	Clock monitor 3 module is disabled.
0	Clock monitor 3 module is enabled.

Bit 13 – CM2MD Clock Monitor 2 Disable bit

Value	Description
1	Clock monitor 2 module is disabled.
0	Clock monitor 2 module is enabled.

Bit 12 – CM1MD Clock Monitor 1 Disable bit

Value	Description
1	Clock monitor 1 module is disabled.
0	Clock monitor 1 module is enabled.

Bit 9 – PWM2MD PWM Module 2 Disable bit

Value	Description
1	PWM module 2 is disabled.
0	PWM module 2 is enabled.

Bit 8 – PWM1MD PWM Module 1 Disable bit

Value	Description
1	PWM module 1 is disabled.
0	PWM module 1 is enabled.

Bit 7 – UREFMD UREF Module Disable bit

Value	Description
1	UREF module is disabled.
0	UREF module is enabled.

Bit 5 – CRYMD Crypto Module Disable bit

Value	Description
1	Crypto accelerator is disabled.
0	Crypto accelerator is enabled.

Bit 2 – DMTMD DMT Module Disable bit

Value	Description
1	DMT module is disabled.
0	DMT module is enabled.

Bit 1 – CRCMD CRC Module Disable bit

Value	Description
1	CRC module is disabled.
0	CRC module is enabled.

Bit 0 – PTGMD PTG Module Disable bit

Value	Description
1	PTG module is disabled.
0	PTG module is enabled.

37.3. Operation

37.3.1. Instruction-Based Power-Saving Modes

The dsPIC33A family of devices can operate in two instruction-based power-saving modes: Sleep and Idle. These modes can be entered by executing a special `PWRSAV` instruction. Sleep mode stops clock operation and halts all code execution. Idle mode halts the CPU and code execution but allows peripheral modules to continue operation. The assembler syntax of the `PWRSAV` instruction is shown in Example 37-1 and Example 37-2.

Note: SLEEP_MODE and IDLE_MODE are constants defined in the assembler include file for the selected device. Sleep and Idle modes can be exited as a result of an enabled interrupt, WDT time-out or a device Reset. When the device exits these modes, it is said to "wake up."

Example 37-1. `PWRSAV` Instruction Syntax in Assembly

```
PWRSAV #SLEEP_MODE;   // Put the device into Sleep mode
PWRSAV #IDLE_MODE;    // Put the device into Idle mode
```

Example 37-2. `PWRSAV` Instruction Syntax in C Language

```
Sleep();     // Put the device into Sleep mode
Idle ();     // Put the device into Idle mode
```

37.3.1.1. Interrupts Coincident with Power-Saving Instructions

Any interrupt that coincides with the execution of a `PWRSAV` instruction is held off until entry into Sleep or Idle mode has completed. If the interrupt is a wake-up event, it will then wake up the device and execute.

37.3.1.2. Sleep Mode

Sleep mode is the lowest Current Consumption state. The CPU and most peripherals are halted. Select peripherals can continue to operate in Sleep mode and can be used to wake the device from Sleep. See the individual peripheral module sections for descriptions of behavior in Sleep mode.

37.3.1.2.1. Sleep Mode Entry

1. The Primary Oscillator (POSC) and Auxiliary Oscillator are disabled.
2. The WDT, clock source and LPRC Oscillator continue to run if the Watchdog Timer is enabled. The WDT, if enabled, is automatically cleared prior to entering Sleep mode. For details, refer to Watchdog Timer.
3. Any attempt to wake up the device while going into Sleep results in erratic behavior.
4. If any peripheral requests the FRC clock in Sleep mode, then FRC will remain active. Also, the FRC Oscillator can be operated in Low-Power mode. This is controlled by the FRCLPWR[1:0] bits in the Oscillator Configuration register (OSCCFG[17:16]). For details, refer to Oscillator Module.
5. The peripherals operating with the system clock are disabled. Optionally, the peripherals can operate in Sleep mode using specific clock sources.
6. The Fail-Safe Clock Monitor (FSCM) does not operate during Sleep mode because the system clock is disabled.

To minimize current consumption in Sleep mode, do the following:

- Ensure that I/O pins do not drive resistive loads.
- Ensure that I/O pins configured as inputs are not floating.
- Disable the WDT.

37.3.1.2.2. Sleep Mode Exit

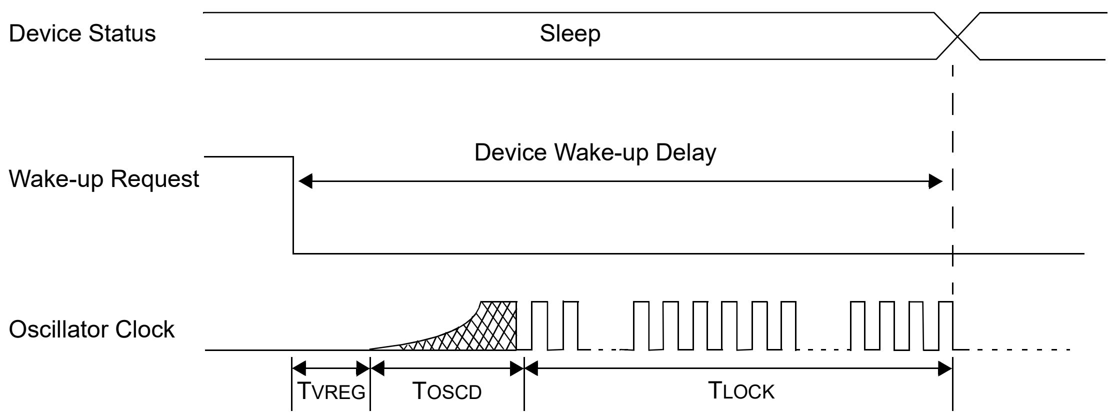
When the device exits Sleep mode, the CPU starts executing instructions within eight system clock cycles. The system clock and peripheral clock are re-enabled. The device restarts with the current clock source as indicated by the Current Clock Source Selection bits (COSC[2:0]) in the Oscillator Control register (OSCCTRL[2:0]). The device is now in Normal Operation mode.

37.3.1.2.3. Delay on Wake-up from Sleep

Figure 37-1 shows the wake-up delay from Sleep mode. This delay consists of the voltage regulator delay and the oscillator delay:

- **Voltage Regulator Delay:** This is the time delay for the voltage regulator to transition from the Standby state to the Active state. This delay is required only if Standby mode is enabled for the voltage regulator.
- **Oscillator Delay:** The time delay for the clock to be ready for various clock sources is shown in Table 37-1. For details, refer to Oscillator and Clocking Module.

Figure 37-1. Wake-up Delay from Sleep Mode



Where: TVREG = Voltage Regulator Standby to Active Mode Transition Time
TOSCD = Oscillator Start-up Delay
TLOCK = PLL Lock Time if PLL is Enabled

Table 37-1. Oscillator Delay(1,2,3)

Oscillator Source	Oscillator Start-up Delay	Oscillator Start-up Timer	PLL Lock Time	Total Delay
FRC, FRCDIV16, FRCDIVN	T_{OSCD}	—	—	T_{OSCD}
FRCPLL	T_{OSCD}	—	T_{LOCK}	T_{OSCD} + T_{LOCK}
XT	T_{OSCD}	T_{OST}	—	T_{OSCD} + T_{OST}
HS	T_{OSCD}	T_{OST}	—	T_{OSCD} + T_{OST}
EC	—	—	—	—
XTPLL	T_{OSCD}	T_{OST}	T_{LOCK}	T_{OSCD} + T_{OST} + T_{LOCK}
HSPLL	T_{OSCD}	T_{OST}	T_{LOCK}	T_{OSCD} + T_{OST} + T_{LOCK}
ECPLL	—	—	T_{LOCK}	T_{LOCK}
LPRC	T_{OSCD}	—	—	T_{OSCD}

Notes:
1. T_{OSCD} = Oscillator start-up delay. Crystal oscillator start-up time varies with crystal characteristics, load capacitance, etc.
2. T_{OST} = Oscillator Start-up Timer delay.
3. T_{LOCK} = PLL lock time if PLL is enabled.

Note: Refer to Electrical Characteristics for T_{VREG}, T_{OST} and T_{LOCK} specifications and also for the T_{OSCD} specifications when using the internal FRC or internal LPRC Oscillator.

37.3.1.3. Idle Mode

Idle mode has the following characteristics:

- The CPU stops executing instructions.
- The system clock source remains active. The clock generators, controlled by the CLKGENx register, can be put into Idle mode by setting the corresponding SIDL bit. Note that the SIDL bit is unimplemented in the CLKGEN1 register. For more details, refer Oscillator and Clocking Module.
- The WDT is automatically cleared.

- If the WDT or FSCM is enabled, the LPRC remains active.
- The peripheral modules, by default, continue to operate normally from the system clock source.
- Peripherals can optionally be shut down using their Stop-in-Idle (SIDL) control bit, which is located in bit position 13 of the control register for most peripheral modules. The generic bit field name format is “xxxSIDL” (where “xxx” is the mnemonic name of the peripheral device). For more details, refer to the respective peripheral sections in the data sheet.
- The FRC Oscillator can be operated in Low-Power mode. This is controlled by the FRCLPWR[1:0] bits in the Oscillator Configuration register (OSCCFG[17:16]). For details, refer to Oscillator and Clocking Module.
- Flash can be put into Idle mode to reduce the power consumption. This is controlled by the NVMSIDL bit in the NVM Control register (NVMCON[12]). For details, refer to Flash Program Memory.

When the device exits Idle mode, the CPU starts executing instructions within eight system clock cycles.

37.3.1.4. Wake-up from Sleep and Idle

Sleep and Idle modes exit on the following events:

- An enabled interrupt event
- A WDT time-out
- Reset from any source (Power-on Reset, Brown-out Reset and $\overline{\text{MCLR}}$)

37.3.1.4.1. Wake-up on Interrupt

An enabled interrupt event wakes up the device from Sleep or Idle mode and then the following occurs:

- If the assigned priority for the interrupt is less than or equal to the current CPU priority, the device wakes up and continues code execution from the instruction following the `PWRSAV` instruction that initiated Sleep/Idle mode.
- If the assigned priority level for the interrupt source is greater than the current CPU priority, the device wakes up and the CPU exception process begins. Code execution continues from the first instruction of the ISR.

37.3.1.4.2. Wake-up on WDT Time-out

If enabled, the Watchdog Timer continues to run during Sleep mode or Idle mode. When the WDT time-out occurs, the device wakes up and code execution continues from where the `PWRSAV` instruction was executed.

The Watchdog Timer Time-out Flag bit (WDTO) in the Reset Control register (RCON[4]) is set to indicate that the wake-up event is due to a WDT time-out.

37.3.1.4.3. Wake-up on Reset

A Reset from any source (Power-on Reset, Brown-out Reset and $\overline{\text{MCLR}}$) causes the device to exit Sleep or Idle mode and begin executing from the Reset vector.

37.3.2. Peripheral Module Disable (PMD)

All peripheral modules (except for I/O ports) in dsPIC33A devices have a control bit that can be selectively disabled to reduce power consumption. These bits, known as the Peripheral Module Disable (PMD) bits, are generically named “xxxMD” (where “xxx” is the mnemonic version of the module’s name). These bits are located in the PMDx Special Function Registers (SFRs). The PMD bit must be set (= `1`) to disable the module. The PMD bit completely shuts down the peripheral, effectively powering down all circuits and removing all clock sources. By default, all peripherals are not disabled by PMD.

37.3.3. Clock Frequency and Clock Switching

The dsPIC33AK256MPS306 family devices allow a wide range of clock frequencies to be selected under application control. If the system clock configuration is not locked, users can choose low-power or high-precision oscillators by setting NOSCx bits (OSCCON[10:8]) and setting the OSCSWEN bit, as well as addressing any CLKxDIV changes required.. The process of changing a system clock during operation, as well as limitations to the process, are discussed in more detail in Oscillator and Clocking Module.

38. JTAG Interface

The dsPIC33AK256MPS306 family of devices support Boundary Scan through a JTAG interface. However, programming is not supported through the JTAG Interface.

39. In-Circuit Debugger

When a supported emulator/debugging tool, such as the MPLAB® ICD5, is selected as a debugger, the in-circuit debugging functionality is enabled. This function allows simple debugging functions when used with MPLAB X IDE. Debugging functionality is controlled through the PGCx (Emulation/ Debug Clock) and PGDx (Emulation/Debug Data) pin functions.

Any of the three pairs of debugging clock/data pins can be used:

- PGC1 and PGD1
- PGC2 and PGD2
- PGC3 and PGD3

To use the in-circuit debugger function of the device, the design must implement ICSP connections to $\overline{MCLR}$, V_{DD}, V_{SS} and the PGCx/PGDx pin pair.

40. Instruction Set Summary

The dsPIC33AK256MPS306 family of devices supports a modified Harvard architecture. For details of the 32-bit Comprehensive Instruction Set for the dsPIC33AK256MPS306 device family, refer to the *"dsPIC33A Programmer's Reference Manual"*, which is available from the Microchip website (www.microchip.com).

The instruction set is highly orthogonal and is grouped into five basic categories:

- Long word, word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- DSP operations
- Control operations

Table 40-1 lists the general symbols used in describing the instructions.

The instruction set summary in Table 40-2 lists all the instructions, along with the status flags affected by each instruction.

Most long word, word or byte-oriented operations with W register instructions (including barrel shift instructions) have three operands:

- The first source operand, which is typically a register 'Wb' without any address modifier.
- The second source operand, which is typically a register 'Ws' with or without an address modifier.
- The destination of the result, which is typically a register 'Wd' with or without an address modifier.

However, long word, word or byte-oriented operations file register instructions have two operands:

- The file register specified by the value 'f'
- The destination, which could be either the file register 'f' or any W register

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or file register (specified by the value of 'Ws' or 'f')
- The bit in the W register or file register (specified by a literal value or indirectly by the contents of register 'Wb')

The literal instructions that involve data movement can use some of the following operands:

- A literal value to be loaded into a W register (specified by 'k')
- The W register or file register where the literal value is to be loaded (specified by 'Wb' or 'f')

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand, which is a register 'Wb' without any address modifier.
- The second source operand, which is a literal value.
- The destination of the result (only if not the same as the first source operand), which is typically a register 'Wd' with or without an address modifier.

The `MAC` class of DSP instructions can use some of the following operands:

- The accumulator (A or B) to be used (required operand)
- The W registers to be used as the two operands

- The X and Y address space prefetch operations
- The X and Y address space prefetch destinations
- The accumulator write-back destination

The other DSP instructions do not involve any multiplication and can include:

- The accumulator to be used (required)
- The source or destination operand (designated as Wso or Wdo, respectively) with or without an address modifier
- The amount of shift specified by a W register 'Wn' or a literal value

The control instructions can use a program memory address.

Most single-word instructions are executed in a single instruction cycle. However, if a conditional test is true and the Program Counter is changed as a result of the instruction, the execution takes multiple instruction cycles, with the additional instruction cycle(s) executed as a `NOP`. Certain instructions that involve skipping over the subsequent instruction require extra cycles if the skip is performed, depending on whether the instruction being skipped is a single-word or two-word instruction.

Note: For more details on the instruction set, refer to the *"dsPIC33A Programmer's Reference Manual (DS70005540)"* (www.microchip.com).

Table 40-1. Symbols Used in Opcode Descriptions

Symbol(1)	Description
`{ }`	Optional field or operation
`[text]`	The location addressed by text
`(text)`	The contents of text
`#text`	The literal defined by text
`{label:}`	Optional label name
`[n:m]`	Register bit field
`.l`	32-bit Long Word mode selection
`.b`	8-bit Byte mode selection
`.sl`	24-bit (literal) Word mode selection
`.V`	Destination data value select (MAXABW, MINABW and FLIMW)
`.w`	16-bit Word mode selection (default)
`AWB`	Accumulator write back destination address register
`bit3`	3-bit bit selection field (used in byte addressed instructions) (0:7)
`bit4`	4-bit bit selection field (used in word addressed instructions) (0:15)
C, N, OV, Z	ALU status bits: Carry, Digit Carry, Negative, Overflow, Zero
d	File register destination (W0, none)
`Expr`	Absolute address, label or expression (resolved by the linker)
`f`	File register address (0x0000:0xFFFF) or (0x00000:0xFFFFF) (addressable space varies depending on instruction class)
`Fd`(2)	One of 32 FPU destination data registers (F0:F31) (Register Direct)
`Fs`(2)	One of 32 FPU source data registers (F0:F31) (Register Direct)
`FSR`, `FSRH`, `FCR`, `FEAR 1`(2)	FPU special (control & status) coprocessor registers (Register Direct)
`label`	Translates to a literal representing the location of the label name
`lit1`	1-bit unsigned literal (0:1)
`lit3`	3-bit unsigned literal (0:7)
`lit5`	5-bit unsigned literal (0:31)
`lit6`	6-bit unsigned literal (0:63)

Table 40-1. Symbols Used in Opcode Descriptions (continued)

Symbol[(1)]	Description
lit8	8-bit unsigned literal (0:255)
lit16	16-bit unsigned literal (0:65535)
lit24	24-bit unsigned literal (0:1677215; LSb must be 0 if an address)
lit32	32-bit unsigned literal (0:4294967295)
none	Field does not require an entry and may be blank
OA, OB, SA, SB	DSC status bits: ACCA Overflow, ACCB Overflow, ACCA Saturate, ACCB Saturate
PC	Program Counter
Rdo	Destination Working register
Rnd	Instruction rounding mode [E, Z, P, N]
Rso	Source Working register
Slit6	Signed 6-bit literal (-32:31)
Slit7	Signed 7-bit literal (-64:63)
Slit8	Signed 8-bit literal (-128:127)
Slit20	Signed 20-bit literal (-524288:524287)
SR	Status Register
text1 ∈ {text2, {text3,…}	text1 must be in the set of text2, text3, …
v	Selects MULxxx operand data types
Wb	Base Working register
Wd	Destination Working register
Wm	One of 16 Working registers (W0:W15)
Wn	Both source and destination Working register (W0:W15)
Wnd	One of 16 destination Working registers
Wns	One of 16 source Working registers
Wm * Wm	Multiplicand and Multiplier Working register for Square instructions
Wm * Wn	Multiplicand and Multiplier W register for DSP instructions
Ws	Source Working register
Wx	X data space fetch address register for DSP instructions
Wy	Y data space fetch address register for DSP instructions

Notes:

1. The range of each symbol is instruction-dependent.
2. Only applicable when the FPU coprocessor is present.
3. Read and Read-Modify-Write (RMW) operations on non-CPU Special Function Registers require additional cycles when updating SFRs belonging to peripherals on different peripheral clock speeds. The bus speed should be taken into consideration for time-sensitive writes.

Table 40-2. Instruction Set Overview

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles[(1)]	Status Flags Affected
1	ABS	ABS Fs, Fd	Absolute value of Fs (FPU Instr)	1	1	SUBO

Table 40-2. Instruction Set Overview (continued)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles[1]	Status Flags Affected
2	ADD	ADD A	Add Accumulators	0.5	1	OA, SA, OB, SB
		ADD Rso,#Slit6, A	16-bit Signed Add to Accumulator	1	1	OA, SA, OB, SB
		ADD f,Wn	f = f + Wn	1	1	C, N, OV, Z
		ADD f,Wn,Wn	Wn = f + Wn	1	1	C, N, OV, Z
		ADD.l #lit5,Wn	Wn = Wn + lit5	0.5	1	C, N, OV, Z
		ADD #lit16,Wn	Wn = Wn + lit16	0.5/1	1	C, N, OV, Z
		ADD Wb,Ws,Wd	Wd = Wb + Ws	1	1	C, N, OV, Z
		ADD Wb,#lit7,Wd	Wd = Wb + lit7 (literal zero-extended)	1	1	C, N, OV, Z
		ADD Fb, Fs, Fd	Fd = Fb + Fs (FPU Instr)			SUBO, INX, UDF, OVF, INVAL
3	ADDC	ADDC f,Wn	f = f + Wn + (C)	1	1	C, N, OV, Z
		ADDC f,Wn,Wn	Wn = f + Wn + (C)	1	1	C, N, OV, Z
		ADDC #lit16,Wn	Wn = Wn + lit16 + (C)	1	1	C, N, OV, Z
		ADDC Wb,Ws,Wd	Wd = Wb + Ws + (C)	0.5/1	1	C, N, OV, Z
		ADDC Wb,#lit7,Wd	Wd = Wb + lit7 + (C)(literal zero-extended)	1	1	C, N, OV, Z
4	AND	AND f,Wn	f = f .AND. Wn	1	1	N, Z
		AND f,Wn,Wn	W0 = f .AND. Wn	1	1	N, Z
		AND #lit16,Wn	Wn = Wn .AND. lit16	1	1	N, Z
		AND Wb,Ws,Wd	Wd = Wb .AND. Ws	0.5/1	1	N, Z
		AND Wb,#lit7,Wd	Wd = Wb .AND. Lit7 (literal zero-extended)	1	1	N, Z
		AND1 Wb,#lit7,Wd	Wd = Wb .AND. Lit7 (literal zero-extended)	1	1	N, Z
		AND #lit16, FSR	FSR = FSR AND lit16 (FPU instr)	1	1	SUBO, HUGI, INX, UDF, OVF, DIV0, INVAL
		AND #lit16, FCR	FCR = FCR AND lit16 (FPU instr)	1	1	None
		AND #lit16, FEAR	FEAR = FEAR AND lit16 (FPU instr)	1	1	None
5	ASR	ASR f	f = Arithmetic Right Shift f by 1	1	1	N, Z
		ASR f,Wn	Wn = Arithmetic Right Shift f by 1	1	1	N, Z
		ASR Ws,Wd	Wd = Arithmetic Right Shift Ws by 1	0.5/1	1	N, Z
		ASR Ws,Wb,Wd	Wnd = Arithmetic Right Shift Ws by Wb	0.5/1	1	N, Z
		ASR Ws,lit5,Wd	Wnd = Arithmetic Right Shift Ws by lit5	0.5/1	1	N, Z
6	ASRM	ASRM Ws, #lit5, Wnd	Wnd = Arithmetic Right Shift Ws by lit5, then logically OR with next lsw	1	2	N, Z
		ASRM Ws, Wb, Wnd	Wnd = Arithmetic Right Shift Ws by Wb, then logically OR with next lsw	1	2	N, Z
7	BCLR	BCLR.b f,bit3	Bit Clear f	1	1	None
		BCLR Ws,bit4	Bit Clear Ws	0.5/1	1	None
8	BFEXT	BFEXT bit4,wid5,Ws,Wb	Bit Field Extract from Ws to Wb	1	1	None
		BFEXT bit4,wid5,f,Wb	Bit Field Extract from f to Wb	2	2	None
9	BFINS	BFINS bit4,wid5,Wb,Ws	Bit Field Insert from Wb into Ws	1	1	None
		BFINS bit4,wid5,Wb,f	Bit Field Insert from Wb into f	2	2	None
		BFINS bit4,wid5,#lit8,Ws	Bit Field Insert lit8 into Ws	2	2	None
10	BOOTSWP	BOOTSWP	Swap the Active and Inactive Program Flash Space	1	2	None

Table 40-2. Instruction Set Overview (continued)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
11	BRA	BRA Label	Branch Unconditionally	1	1	None
		BRA Wn	Computed Branch	1	2	None
		BRA C,Label	Branch if Carry	1	1(2/3)	None
		BRA GE,Label	Branch if greater than or equal	1	1(2/3)	None
		BRA GEU,Label	Branch if unsigned greater than or equal	1	1(2/3)	None
		BRA GT,Label	Branch if greater than	1	1(2/3)	None
		BRA GTU,Label	Branch if unsigned greater than	1	1(2/3)	None
		BRA LE,Label	Branch if less than or equal	1	1(2/3)	None
		BRA LEU,Label	Branch if unsigned less than or equal	1	1(2/3)	None
		BRA LT,Label	Branch if less than	1	1(2/3)	None
		BRA LTU,Label	Branch if unsigned less than	1	1(2/3)	None
		BRA N,Label	Branch if Negative	1	1(2/3)	None
		BRA NC,Label	Branch if Not Carry	1	1(2/3)	None
		BRA NN,Label	Branch if Not Negative	1	1(2/3)	None
		BRA NOV,Label	Branch if Not Overflow	1	1(2/3)	None
		BRA NZ,Label	Branch if Not Zero	1	1(2/3)	None
		BRA Z,Label	Branch if Zero	1	1(2/3)	None
		BRA OA,Label	Branch if accumulator A overflow	1	1(2/3)	None
		BRA OB,Label	Branch if accumulator B overflow	1	1(2/3)	None
		BRA OV,Label	Branch if Overflow	1	1(2/3)	None
		BRA SA,Label	Branch if accumulator A saturated	1	1(2/3)	None
		BRA SB,Label	Branch if accumulator B saturated	1	1(2/3)	None
12	BREAK	BREAK	Stop User Code Execution	1	1	None
13	BSET	BSET.b f,bit3	Bit Set f	1	1	None
		BSET Ws,bit4	Bit Set Ws	0.5/1	1	None
14	BSW	BSW.C Ws,Wb	Write C or Z bit to Ws<Wb>	1	1	None
		BSW.Z Ws,Wb	Write C or Z bit to Ws<Wb>	0.5/1	1	None
15	BTG	BTG.b f,bit3	Bit Toggle f	1	1	None
		BTG Ws,bit4	Bit Toggle Ws	0.5/1	1	None
16	BTSC		Not supported			N/A
17	BTSS		Not supported			N/A
18	BTST	BTST.b f,bit3	Bit Test f	1	1	Z
		BTST.C Ws,bit4	Bit Test Ws to C	0.5/1	1	C, Z
		BTST.Z Ws,bit4	Bit Test Ws to Z	1	1	Z
		BTST.C Ws,Wb	Bit Test Ws<Wb> to C	0.5/1	1	C, Z
		BTST.Z Ws,Wb	Bit Test Ws<Wb> to Z	1	1	Z
19	BTSTS	BTSTS.b f,bit3	Bit Test then Set f	1	1	Z
		BTSTS.C Ws,bit4	Bit Test Ws to C then Set	0.5/1	1	C, Z
		BTSTS.Z Ws,bit4	Bit Test Ws to Z then Set	1	1	Z
20	CALL	CALL Label	Call subroutine (label > ~ 16MB)	1	1	None
			Call subroutine (label < ~ 16MB)	2	2	None
		CALL Wns	Call indirect subroutine at address [W11]	1	2	None
21	CLR	CLR f	f = 0x0000 0000	1	1	None
		CLR Wd	Wd = 0x0000 0000	1	1	None
		CLR A	Clear Accumulator	0.5	1	None
21	CLRWDT	CLRWDT	Clear Watchdog Timer	0.5	1	WDTO, Sleep
22	COM	COM f	f = f	1	1	N, Z
		COM f,Wd	Wd = f	1	1	N, Z
		COM Ws,Wd	Wd = Ws	0.5/1	1	N, Z

Table 40-2. Instruction Set Overview (continued)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
23	CP	CP f,Ws	Compare f with Ws	1	1	C, N, OV, Z
		CP Ws,#lit13	Compare Ws with lit13 (literal zeroextended)	1	1	C, N, OV, Z
		CP Wb,#lit16	Compare Wb with lit16 (literal zeroextended)	1	1	C, N, OV, Z
		CP Wb, Ws	Compare Wb with Ws	0.5/1	1	C, N, OV, Z
24	CP0	CP0 f	Compare f with 0x0000 0000	1	1	C, N, OV, Z
		CP0 Ws	Compare Ws with 0x0000 0000 (substitute CPLS Ws ,#0)	1	1	C, N, OV, Z
25	CPB	CPB f,Ws	Compare f with Ws, with borrow	1	1	C, N, OV, Z
		CP Wb,#lit13	Compare Wb with lit13, with borrow (literal zero-extended)	1	1	C, N, OV, Z
		CP Wb,#lit16	Compare Wb with lit16, with borrow (literal zero-extended)	1	1	C, N, OV, Z
		CPB Wb,Ws	Compare Borrow Wb with Ws	0.5/1	1	C, N, OV, Z
	CPSEQ		Not supported			N/A
	CPBEQ		Not supported			N/A
	CPSGT		Not supported			N/A
	CPBGT		Not supported			N/A
	CPSNE		Not supported			N/A
26	CTXTSWP	CTXTSWP #lit3	Swap to CPU register context defined by lit3	0.5	2	None
		CTXTSWP Wn	Swap to CPU register context defined in Wn[2:0]	1	2	None
27	DAW		Not supported			N/A
28	DEC	DEC f	f = f -1	1	1	C, N, OV, Z
		DEC f,Wd	W5 = f -1	1	1	C, N, OV, Z
		DEC Ws,Wd	Wd = Ws - 1	1	1	C, N, OV, Z
29	DEC2	DEC2 f	f = f -2	1	1	C, N, OV, Z
		DEC2 f,Wd	W5 = f -2	1	1	C, N, OV, Z
		DEC2 Ws,Wd	Wd = Ws - 2	1	1	C, N, OV, Z
30	DISI		Not supported			N/A
31	DISICTL	DISICTL #lit3 {,Wd}	Disable interrupts at IPL <= lit3 Optionally save prior IPL threshold to Wd	1	1	None
		DISICTL Wns {,Wd}	Disable interrupts at IPL <= Wns[2:0] Optionally save prior IPL threshold to Wd	1	1	None
32	DIVF	DIVF Wm/Wn	Interruptible Signed 16/16 or 32/16 Fractional Divide	1	1	C, N, OV, Z
33	DIVFL	DIVFL Wm/Wn	Interruptible Signed 32/32 Fractional Divide	1	1	C, N, OV, Z
34	DIVS	DIVS.w Wm/Wn	Interruptible Signed 16/16-bit Integer Divide	1	1	C, N, OV, Z
		DIVS.l Wm/Wn	Interruptible Signed 32/16-bit Integer Divide	1	1	C, N, OV, Z
35	DIVSL	DIVSL Wm/Wn	Interruptible Signed 32/32 Integer Divide	1	1	C, N, OV, Z
36	DIVU	DIVU.w Wm/Wn	Interruptible Unsigned 16/16-bit Integer Divide	1	1	C, N, OV, Z
		DIVU.l Wm/Wn	Interruptible Unsigned 32/16-bit Integer Divide	1	1	C, N, OV, Z
37	DIVUL	DIVUL Wm/Wn	Interruptible Unsigned 32/32 Integer Divide	1	1	C, N, OV, Z
38	DIV2		Not supported			N/A

Table 40-2. Instruction Set Overview (continued)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
39	DTB	DTB Wn,Label	Decrement Wn, then branch if not zero	1	1(2/3)	None
40	DO		Not supported			N/A
41	ED	ED Wxp * Wyp, A, AWB	Euclidean Distance	1	2	OA, SA, OB, SB
42	EDAC	EDAC Wxp * Wyp, A, AWB	Euclidean Distance Accumulate	1	2	OA, SA, OB, SB
43	EXCH	EXCH Wns,Wnd	Swap Wns with Wnd	1	2	None
44	FBCL	FBCL Ws,Wnd	Find Bit Change from Left (MSb) Side	1	1	C
45	FF1L	FF1L Ws,Wnd	Find First One from Left (MSb) Side	1	1	C
46	FF1R	FF1R Ws,Wnd	Find First One from Right (LSb) Side	1	1	C
47	FLIM	FLIM Wb, Ws	Force Data (Upper and Lower) Range Limit without Limit Excess Result	1	2	N, Z, OV
		FLIM Wb, Ws, Wd	Force Data (Upper and Lower) Range Limit with Limit Excess Flag (Wd=-1)	1	2	N, Z, OV
		FLIM.V Wb, Ws, Wd	Force Data (Upper and Lower) Range Limit with Limit Excess Result	1	2	N, Z, OV
48	GOTO	GOTO Label	Goto address (address < ~ 16MB)	1	2	None
		GOTO Label	(label < ~ 16MB)	2	2	None
		GOTO Wn	Go to indirect address at [W11]	1	2	None
49	INC	INC f	f = f + 1	1	1	C, N, OV, Z
		INC f,Wd	W5 = f + 1	1	1	C, N, OV, Z
		INC Ws,Wd	Wd = Ws + 1	1	1	C, N, OV, Z
50	INC2	INC2 f	f = f + 2	1	1	C, N, OV, Z
		INC2 f,Wd	W5 = f + 2	1	1	C, N, OV, Z
		INC2 Ws,Wd	Wd = Ws +2	1	1	C, N, OV, Z
51	IOR	IOR f,Wn	f = f .IOR. Wn	1	1	N, Z
		IOR f,Wn,Wn	Wn = f .IOR. Wn	1	1	N, Z
		IOR #lit16,Wn	Wn = Wn .IOR. lit16	1	1	N, Z
		IOR Wb,Ws,Wd	Wd = Wb .IOR. Ws	0.5/1	1	N, Z
		IOR Wb,#lit7,Wd	Wd = Wb .IOR. lit7	1	1	N, Z
52	LAC	LAC Rso,#Slit6, A	Load Accumulator (16/32-bit), literal shift	1	1	OA, SA, OB, SB
	LLAC	LLAC.l Rso,#Slit6, A	Load Lower (LSw) of Accumulator (32-bit), literal shift	1	1	OA, SA, OB, SB
	LAUC	LUAC.l Rso,#Slit6, A	Load Upper (LSb) of Accumulator (32-bit), literal shift	1	1	OA, SA, OB, SB
53	LNK	LNK #lit16	Link frame pointer	1	1	None
		LNK #lit7	Link frame pointer (literal < 128)	0.5	1	None
54	LSR	LSR f	f = Logical Right Shift f by 1	1	1	C,N,Z
		LSR f,Wd	Wd = Logical Right Shift f by 1	1	1	C,N,Z
		LSR Ws,Wd	Wd = Logical Right Shift Ws by 1	0.5/1	1	C,N,Z
		LSR Ws,Wb,Wd	Wnd = Logical Right Shift Ws by Wns	0.5/1	1	C,N,Z
		LSR Ws,#lit5,Wd	Wnd = Logical Right Shift Ws by lit5	0.5/1	1	N,Z
		LSRM Ws,#lit5, Wnd	Wnd = Logical Right Shift Ws by lit5, then logically OR with next lsw	1	2	N,Z
		LSRM Ws, Wb, Wnd	Wnd = Logical Right Shift Ws by Wb, then logically OR with next lsw	1	2	N,Z
55	MAC	MAC Wxp * Wyp, A, AWB	Multiply and Accumulate	1	1	OA, SA, OB, SB
56	MAX	MAX Wb, Ws	Force Data Maximum Range Limit	1	1	OA, SA, OB, SB
		MAX A	Force Data Maximum Range Limit	0.5	1	N, Z, OV
		MAX.V A, Rdo	Force Data Maximum Range Limit with Result	1	2	N, Z, OV

Table 40-2. Instruction Set Overview (continued)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
57	MIN	MIN Wb, Ws	Force Data Minimum Range Limit	1	1	N, Z, OV
		MIN A	Force Data Minimum Range Limit	0.5	1	N, Z, OV
		MIN.V A, Rdo	Force Data Minimum Range Limit with Result	1	2	N, Z, OV
58	MOV	MOV Rso,Rdo	Move Ws to Wd	0.5/1	1	None
		MOV.l #lit32,Wnd	Move 32-bit unsigned literal to Wnd	2	2	None
		MOV.sl #lit24,Wnd	Move 24-bit unsigned literal to Wnd; 0 extend to 32-bits	1	1	None
		MOV.w #lit16,Wnd	Move 16-bit unsigned literal to Wnd; 0 extend to 32-bits	1	1	None
		MOV.bz #lit8,Wnd	Move 8-bit unsigned literal to Wnd; 0 extend to 32-bits	1	1	None
		MOV.l [W15-#lit7], Wnd [W14+#slit7], Wnd	Move from system stack with literal offset to Wnd using SP or FP	0.5	1	None
		MOV.l Wns, [W15-#lit7] Wns, [W14+#slit7]	Move from Wns to system stack with literal off-set using SP or FP	0.5	1	None
		MOV.l f,Wnd	Move f to Wnd (Word or Long Word)(f < ~1MB)	1	1	None
		MOV.w f,Wnd	Move f to Wnd (Word or Long Word)(f > ~1MB)	2	2	None
		MOV.b f,Wnd	Move f to Wnd (Byte)	1	1	None
		MOV.l Wns,f	Move Wns to f (Word or Long Word)(f < ~1MB)	1	1	None
		MOV.w Wns,f	Move Wns to f (Word or Long Word)(f > ~1MB)	2	2	None
		MOV.b Wns,f	Move Wns to f (Byte)	1	1	None
		MOV [Wns+#Slit12],Wnd	Move [Wns+Slit12] to Wnd	1	1	None
		MOV Wns, [Wnd+#Slit12]	Move Wns to [Wnd+Slit12]	1	1	None
		MOVIF.l CC, Wb, Wns, Wd	If SR.Z=1 Move W1 to [W15++] Else Move W2 to [W15++]	1	1	None
		MOVIF.w CC, Wb, Wns, Wd	If SR.Z=1 Move W1 to [W15++] Else Move W2 to [W15++]	1	1	None
		MOVIF.bz CC, Wb, Wns, Wd	If SR.Z=1 Move W1 to [W15++] Else Move W2 to [W15++]	1	1	None
		MOVIF.b CC, Wb, Wns, Wd	If SR.Z=1 Move W1 to [W15++] Else Move W2 to [W15++]	1	1	None
		MOVR.l	Move Ws to Wd with destination Bit Reversed	1	1	None
		MOVR.w	Move Ws to Wd with destination Bit Reversed	1	1	None
		MOVS.l #slit16, Wd	Move signed extended 16-bit literal to Wd	1	1	None
		MOVS.w #slit16, Wd	Move 16-bit literal to Wd; sign extend to 32-bits if register direct mode.	1	1	None
		MOVS.b #slit8, Wnd	Move 8-bit literal to Wd; no extension.	1	1	None
59	MOVPAG		Not supported			N/A
60	MOVSAC		Not supported			N/A
61	MPY	MPY Wxp * Wyp, A, AWB	Multiply Wm by Wn to Accumulator	1	1	OA, SA, OB, SB
62	MPYN	MPYN Wxp * Wyp, A, AWB	(negative)(Multiply Wm by Wn) to Accumulator	1	1	OA, SA, OB, SB
63	MSC	MSC Wxp * Wyp, A, AWB	Multiply and Subtract from Accumulator	1	1	OA, SA, OB, SB

Table 40-2. Instruction Set Overview (continued)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles(1)	Status Flags Affected
65	MUL	MUL f, Wn	W2 = f * Wn	1	1	None
		MULISS Wb,Ws,A	Integer: Acc(A or B) = signed(Wb) * signed(Ws)	1	1	None
		MULFSS Wb,Ws,A	Fractional: Acc(A or B) = signed(Wb) * signed(Ws)	1	1	None
		MULISU Wb,Ws,A	Integer: Acc(A or B) = signed(Wb) * unsigned(Ws)	1	1	None
		MULFSU Wb,Ws,A	Fractional: Acc(A or B) = signed(Wb) * unsigned(Ws)	1	1	None
		MULIUS Wb,Ws,A	Integer: Acc(A or B) = unsigned(Wb) * signed(Ws)	1	1	None
		MULFUS Wb,Ws,A	Fractional: Acc(A or B) = unsigned(Wb) * signed(Ws)	1	1	None
		MULIUU Wb,Ws,A	Integer: Acc(A or B) = unsigned(Wb) * unsigned(Ws)	1	1	None
		MULFUU Wb,Ws,A	Fractional: Acc(A or B) = unsigned(Wb) * unsigned(Ws)	1	1	None
		MULISS Wb,#slit8,A	Integer: Acc(A or B) = signed(Wb) * signed(slit8)	1	1	None
		MULFSS Wb,#slit8,A	Integer: Acc(A or B) = signed(Wb) * signed(slit8)	1	1	None
		MULISU Wb,#lit8,A	Integer: Acc(A or B) = signed(Wb) * unsigned(lit8)	1	1	None
		MULFSU Wb,#lit8,A	Integer: Acc(A or B) = signed(Wb) * unsigned(lit8)	1	1	None
		MULIUS Wb,#slit8,A	Integer: Acc(A or B) = signed(Wb) * signed(slit8)	1	1	None
		MULFUS Wb,#slit8,A	Integer: Acc(A or B) = signed(Wb) * signed(slit8)	1	1	None
		MULIUU Wb,#lit8,A	Integer: Acc(A or B) = signed(Wb) * unsigned(lit8)	1	1	None
		MULFUU Wb,#lit8,A	Integer: Acc(A or B) = signed(Wb) * unsigned(lit8)	1	1	None
		MULSS Wb,Ws,Wnd	{Wd}=signed(Wb) * signed(Ws)	0.5/1	1	None
		MULSU Wb,Ws,Wnd	{Wd}=signed(Wb) * unsigned(Ws)	0.5/1	1	None
		MULUS Wb,Ws,Wnd	{Wd}=unsigned(Wb) * signed(Ws)	0.5/1	1	None
		MULUU Wb,Ws,Wnd	{Wd}=unsigned(Wb) * unsigned(Ws)	0.5/1	1	None
		MULSU Wb,#lit8,Wnd	{Wd}=signed(Wb) * unsigned(lit8)	1	1	None
		MULUU Wb,#lit8,Wnd	{Wd}=unsigned(Wb) * unsigned(lit8)	1	1	None
		MULSS Wb,#slit8,Wnd	{Wd}=signed(Wb) * signed(slit8)	1	1	None
		MULUS Wb,#slit8,Wnd	{Wd}=unsigned(Wb) * signed(slit8)	1	1	None
66	NEG	NEG A	Negate Accumulator	1	1	OA, SA, OB, SB
		NEG f	f = f +1	1	1	OA, SA, OB, SB
		NEG f,Wd	Wd = f + 1	1	1	OA, SA, OB, SB
		NEG Ws,Wd	Wd = Ws + 1	1	1	OA, SA, OB, SB
67	NEOP	NEOP	None executable NOP (16-bit instruction pad)	0.5	0	None
68	NOP	NOP	No Operation	1	1	None
		NOPR	No Operation	1	1	None
69	NORM	NORM A, Rdo	Normalize Accumulator	1	1	N,OV,Z
70	POP	POP f	Pop f from top of stack (TOS)	1	1	None
		POP {[--Ws],} Wnd	Pop Wnd Register from system stack.	0.5	1	None
		POP Fd	Pop Fd Register from system stack.	0.5	1	None
71	PUSH	PUSH f	Push f to top of stack (TOS)	1	1	None
		PUSH Wns, {[Wd++]}	Push Wns Register to system stack	0.5	1	None
		PUSH Fs	Push Fs Register to system stack	0.5	1	None

Table 40-2. Instruction Set Overview (continued)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles[1]	Status Flags Affected
72	PWRSAV	PWRSAV mode	Go into sleep mode	0.5	2	WDTO, Sleep
	RCALL	RCALL Label	Relative Call	1	1	None
		RCALL Wns	Computed Call	1	2	None
73	REPEAT	REPEAT #lit15	Repeat Next Instruction lit15+1 times	1	1	RA (if lit15 > 0)
		REPEAT #lit5	Repeat Next Instruction lit5+1 times	0.5	1	RA (if lit5 > 0)
		REPEAT Wn	Repeat Next Instruction (Wn)+1 times	1	1	RA (if Wn > 0)
74	RESET	RESET	Software Device Reset	1	1	None
75	RETFIE	RETFIE	Return from interrupt enable	1	4	None
76	RETLW lit16,Wn	RETLW #lit16,Wn	Return from Subroutine with literal in Wn	1	3	None
77	RETURN	RETURN	Return from Subroutine	0.5	3	None
78	RLC	RLC f	f = Rotate Left through Carry f	1	1	C, N, Z
		RLC f,Wd	Wd = Rotate Left through Carry f	1	1	C, N, Z
		RLC Ws,Wd	Wd = Rotate Left through Carry Ws	0.5/1	1	C, N, Z
79	RLNC	RLNC f	f = Rotate Left (No Carry) f	1	1	N, Z
		RLNC f,Wd	Wd = Rotate Left (No Carry) f	1	1	N, Z
		RLNC Ws,Wd	Wd = Rotate Left (No Carry) Ws	0.5/1	1	N, Z
80	RRC	RRC f	f = Rotate Right through Carry f	1	1	C, N, Z
		RRC f,Wd	Wd = Rotate Right through Carry f	1	1	C, N, Z
		RRC Ws,Wd	Wd = Rotate Right through Carry Ws	0.5/1	1	C, N, Z
81	RRNC	RRNC f	f = Rotate Right (No Carry) f	1	1	N, Z
		RRNC f,Wd	Wd = Rotate Right (No Carry) f	1	1	N, Z
		RRNC Ws,Wd	Wd = Rotate Right (No Carry) Ws	0.5/1	1	N, Z
82	SAC	SAC A,#Slit6,Rdo	Store Accumulator (16/32-bit)	1	1	None
		SACR A,#Slit6,Rdo	Store Rounded Accumulator (16/32-bit), literal shift	1	1	None
		SACRW A,Ws,Rdo	Store Rounded Accumulator (16/32-bit), Wb shift	1	1	None
83	SE	SE Rso,Wnd	Wd = sign-extended Ws	0.5/1	1	C, N, Z
84	SETM	SETM f	f = 0xFFFF FFFF	1	1	None
		SETM Wd	Wd = 0xFFFF FFFF	1	1	None
85	SFTAC	SFTAC A,Wn	Arithmetic Shift by (Wn) Accumulator	1	1	OA, SA, OB, SB
		SFTAC A,#Slit7	Arithmetic Shift by Slit7 Accumulator	1	1	OA, SA, OB, SB
86	SL	SL f	f = Left Shift f by 1	1	1	C, N, Z
		SL f,Wd	Wd = Left Shift f by 1	1	1	C, N, Z
		SL Ws,Wd	Wd = Left Shift Ws by 1	0.5/1	1	C, N, Z
		SL Ws,Wb,Wnd	Wnd = Left Shift Wb by Wns	0.5/1	1	C, N, Z
		SL Ws,#lit5,Wnd	Wnd = Left Shift Ws by lit5	0.5/1	1	C, N, Z
	SLM	SLM Ws, #lit5, Wnd	Wnd = Left Shift Wb by lit5, then logically OR with next msw	1	2	Z
		SLM Ws, Wb, Wnd	Wnd = Left Shift Wb by Wb, then logically OR with next msw	1	2	Z
87	SLAC	SLAC.l A,#Slit6,Rdo	Store Lower (lsw of) Accumulator (32-bit), literal shift	1	1	None
88	SUAC	SUAC.l A,#Slit6,Rdo	Store sign extended Upper (MSB) Accumulator (32- bit), literal shift	1	1	None
89	SUB	SUB A	Subtract Accumulators	0.5	1	OA, SA, OB, SB
		SUB Rso,#Slit6, A	16-bit Signed Subtract from Accumulator	1	1	OA, SA, OB, SB
		SUB f,Wn	f = f - Wn	1	1	C, N, OV, Z
		SUB f,Wn,Wn	Wn = f - Wn	1	1	C, N, OV, Z
		SUB.l #lit5,Wn	Wn = Wn - lit5	0.5	1	C, N, OV, Z
		SUB #lit16,Wn	Wn = Wn - lit16	1	1	C, N, OV, Z
		SUB Wb,Ws,Wd	Wd = Wb - Ws	0.5/1	1	C, N, OV, Z
		SUB Ws,#lit7,Wd	Wd = Ws - lit7 (literal zero-extended)	1	1	C, N, OV, Z

Table 40-2. Instruction Set Overview (continued)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles[1]	Status Flags Affected
90	SUBB	SUBB f,Wn	f = f - Wn - (C)	1	1	C, N, OV, Z
		SUBB f,Wn,Wn	Wn = f - Wn - (C)	1	1	C, N, OV, Z
		SUBB #lit16,Wn	Wn = Wn - lit16 - (C)	1	1	C, N, OV, Z
		SUBB Wb,Ws,Wd	Wd = Wb - Ws - (C)	0.5/1	1	C, N, OV, Z
		SUBB Ws,#lit7,Wd	Wd = Ws - lit7 - (literal zero-extended)	1	1	C, N, OV, Z
91	SUBR	SUBR f,Wn	f = Wn - f	1	1	C, N, OV, Z
		SUBR f,Wn,Wn	Wn = Wn - f	1	1	C, N, OV, Z
		SUBR Wb,Ws,Wd	Wd = Ws - Wb	0.5/1	1	C, N, OV, Z
		SUBR Ws,#lit7,Wd	Wd = lit7 - Ws (literal zero-extended)	0.5/1	1	C, N, OV, Z
92	SUBBR	SUBBR f,Wn	f = Wn - f - (C)	1	1	C, N, OV, Z
		SUBBR f,Wn,Wn	Wn = Wn -f - (C)	1	1	C, N, OV, Z
		SUBBR Wb,Ws,Wd	Wd = Ws - Wb - (C)	0.5/1	1	C, N, OV, Z
		SUBBR Ws,#lit7,Wd	Wd = lit7 - Ws - (C) (literal zero-extended)	1	1	C, N, OV, Z
93	SWAP	SWAP Wn	Wn = Word or byte swap Wn	1	1	None
94	SQR	SQR Wxp, A, AWB	Square to Accumulator	1	1	OA, SA, OB, SB
95	SQRAC	SQRAC Wxp, A, AWB	Square and Accumulate	1	1	OA, SA, OB, SB
96	SQRN	SQRN Wxp, A, AWB	Negated Square to Accumulator	1	1	OA, SA, OB, SB
97	SQRSC	SQRSC Wxp, A, AWB	Square and Subtract from Accumulator	1	1	OA, SA, OB, SB
98	TBLRDH		Not supported			N/A
99	TBLRDL		Not supported			N/A
100	TBLWTH		Not supported			N/A
101	TBLWTL		Not supported			N/A
102	TST	TST f	Test f	1	1	N, Z
		TST f,Wnd	Test f and move f to Wnd	1	1	N, Z
103	ULNK	ULNK	Unlink frame pointer	1	1	None
104	XOR	XOR f,Wn	f = f .XOR. Wn	1	1	N, Z
		XOR f,Wn,Wn	Wn = f .XOR. Wn	1	1	N, Z
		XOR lit16,Wn	Wn = Wn .XOR. lit16	1	1	N, Z
		XOR Wb,Ws,Wd	Wd = Wb .XOR. Ws	0.5/1	1	N, Z
		XOR Wb,lit7,Wd	Wd = Wb .XOR. Lit7 (literal zero-extended)	1	1	N, Z
105	ZE	ZE Rso,Wnd	Wd = Zero-extend Ws	0.5/1	1	C, N, Z

Notes:

1. Read and Read-Modify-Write (e.g., bit operations and logical operations) on non-CPU SFRs incur an additional instruction cycle.
2. For dsPIC33AK256MPS306 devices, the divide instructions must be preceded with a "`REPEAT #5`" instruction, such that they are executed six consecutive times.

41. Development Support

Move a design from concept to production in record time with Microchip's award-winning development tools. Microchip tools work together to provide state of the art debugging for any project with easy-to-use Graphical User Interfaces (GUIs) in our free MPLAB® X and Atmel Studio Integrated Development Environments (IDEs) and our code generation tools. Providing the ultimate ease-of-use experience, Microchip's line of programmers, debuggers and emulators work seamlessly with our software tools. Microchip development boards help evaluate the best silicon device for an application, while our line of third party tools round out our comprehensive development tool solutions.

Microchip's MPLAB X and Atmel Studio ecosystems provide a variety of embedded design tools to consider that support multiple devices, such as PIC® MCUs, AVR® MCUs, SAM MCUs and dsPIC® DSCs. MPLAB X tools are compatible with Windows®, Linux® and Mac® operating systems while Atmel Studio tools are compatible with Windows.

Go to the following website for more information and details:

www.microchip.com/development-tools/

42. Electrical Characteristics

This section provides an overview of the dsPIC33AK256MPS306 family electrical characteristics. Additional information will be provided in future revisions of this document as it becomes available.

Absolute maximum ratings for the dsPIC33AK256MPS306 family are listed below. Exposure to these maximum rating conditions for extended periods may affect device reliability. Functional operation of the device at these or any other conditions above the parameters indicated in the operation listings of this specification is not implied.

Table 42-1. Absolute Maximum Ratings[(1)]

Ambient temperature under bias	-40°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on V_{DD} with respect to V_{SS}	-0.3V to +4.0V
Voltage on AV_{DD} with respect to AV_{SS}	-0.3V to +4.0V
Voltage on any pin that is not 5V tolerant with respect to V_{SS}[(3)]	-0.3V to (V_{DD} + 0.3V)
Voltage on any 5V tolerant pin with respect to V_{SS} when V_{DD} ≥ 3.0V[(3)]	-0.3V to +5.5V
Voltage on any 5V tolerant pin with respect to V_{SS} when V_{DD} < 3.0V[(3)]	-0.3V to +3.6V
Maximum current out of each V_{SS} pin	200 mA
Maximum current into each V_{DD} pin[(2)]	200 mA
Maximum current sunk/sourced by any 4x I/O pin	15 mA
Maximum current sunk/sourced by 8x I/O pin	25 mA
Maximum current sunk by a group of I/Os between two V_{SS} pins[(4)]	80 mA
Maximum current sourced by a group of I/Os between two V_{DD} pins[(4)]	80 mA
Maximum current sunk by all ports[(2)]	200 mA
Maximum current sourced by all ports[(2)]	200 mA

Notes:

1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.
2. Maximum allowable current is a function of device maximum power dissipation (see **DC Characteristics**.
3. See the **Pin Diagrams** section for the 5V tolerant pins.
4. Not applicable to AV_{DD} and AV_{SS} pins.

42.1. DC Characteristics

Table 42-2. Operating MHz vs. Voltage

Characteristic	V_{DD} Range (in Volts)	Temperature Range (in °C)	Maximum MHz dsPIC33AK256MPS306 Family
—	3.0V to 3.6V	-40°C to +85°C	200
	3.0V to 3.6V	-40°C to +125°C	200

Table 42-3. Thermal Operating Conditions

Rating	Symbol	Min.	Typ.	Max.	Unit
Industrial Temperature Devices Operating Junction Temperature Range Operating Ambient Temperature Range	T_J T_A	-40 -40	— —	+125 +85	°C °C
Extended Temperature Devices Operating Junction Temperature Range Operating Ambient Temperature Range	T_J T_A	-40 -40	— —	+140 +125	°C °C
Power Dissipation: Internal Chip Power Dissipation: $P_{INT} = V_{DD} \times (I_{DD} - \Sigma I_{OH})$	P_D	$P_{INT} + P_{I/O}$			W
I/O Pin Power Dissipation: $I/O = \Sigma (\{V_{DD} - V_{OH}\} \times I_{OH}) + \Sigma (V_{OL} \times I_{OL})$					
Maximum Allowed Power Dissipation	P_{DMAX}	$(T_J - T_A)/\theta_{JA}$			W

Table 42-4. Thermal Packaging Characteristics(1)

Characteristic	Symbol	Typ.	Max.	Unit
Package Thermal Resistance, 64-Pin TQFP 10x10x1 mm	θ_{JA}	45.7	—	°C/W
Package Thermal Resistance, 64-Pin VQFN 9x9x0.9 mm	θ_{JA}	23.0	—	°C/W
Package Thermal Resistance, 48-Pin TQFP 7x7 mm	θ_{JA}	62.76	—	°C/W
Package Thermal Resistance, 48-Pin VQFN 6x6 mm	θ_{JA}	33.7	—	°C/W
Package Thermal Resistance, 36-Pin VQFN 5x5 mm	θ_{JA}	40.48	—	°C/W

Note:

1. Junction-to-ambient thermal resistance, Theta-$_{JA}$ (θ_{JA}) numbers are achieved by package simulations.

Table 42-5. Operating Voltage Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated)
Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial
-40°C ≤ T_A ≤ +125°C for Extended

Parameter No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
Operating Voltage							
DC10	V_{DD}	Supply Voltage(1)	3.0	—	3.6	V	—
	SWV $_{DD}$	Switching Regulator Supply Voltage	3.0	—	3.6	V	—

Notes:

1. Device is functional at $V_{BORMIN} < V_{DD} < V_{DDMIN}$. Analog specifications cannot be guaranteed.
2. Parameters are characterized but not tested.

Table 42-5. Operating Voltage Specifications (continued)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
DC11	AV_{DD}	Supply Voltage	Greater of: V_{DD} – 0.3 or 3.0	—	Lesser of: V_{DD} + 0.3 or 3.6	V	The difference between AV_{DD} supply and V_{DD} supply must not exceed ±300 mV at all times, including during device power-up.
DC16	V_{POR}	V_{DD} Start Voltage to Ensure Internal Power-on Reset Signal(2)	—	—	V_{SS}	V	—
DC17	S_{VDD}	V_{DD} Rise Rate to Ensure Internal Power-on Reset Signal(2)	0.03	—	—	V/ms	0V-3V in 100 ms
BO10	V_{BOR}	BOR Event on V_{DD} Transition High-to-Low	2.79	2.9	2.99	V	—

Notes:
1. Device is functional at $V_{BORMIN} < V_{DD} < V_{DDMIN}$. Analog specifications cannot be guaranteed.
2. Parameters are characterized but not tested.

Table 42-6. DC Characteristics: Operating Current (I_{DD})

DC Characteristics	Operating Current		Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended			
Parameter No.	**Typ.(1)**	**Max.**	**Units**	**Conditions**		
Operating Current (I_{DD})(2)						
DC21	5.5	11.5	mA	-40°C	3.3V	20 MHz (N = 1, N2 = 7, N3 =4, M = 70, F_{VCO} = 560 MHz, F_{PLLO} = 20 MHz)
	6	12	mA	+25°C		
	9.5	17.5	mA	+85°C		
	13	22	mA	+125°C		
DC22	7	14	mA	-40°C	3.3V	50 MHz (N = 1, N2 = 4, N3 = 3, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 50 MHz)
	7.5	15	mA	+25°C		
	10.5	20	mA	+85°C		
	14	24.5	mA	+125°C		

Notes:
1. Data in the “Typ.” column are for design guidance only and are not tested in manufacturing.
2. I_{DD} is primarily a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature, also have an impact on the current consumption. Base Run current (I_{DD}) is measured as follows:
   - The oscillator is switched to EC+PLL mode in software.
   - The OSCI pin is driven with an external 8 MHz square wave.
   - OSCO is configured as an I/O.
   - The Watchdog Timer is disabled.
   - All I/O pins (except OSCI) are configured as outputs and driving low.
   - No peripheral modules are operating or being clocked (defined PMDx bits are all ‘`1`’s).
   - JTAG is disabled.
   - `NOP` instructions are executed in a `while(1)` loop.

Table 42-6. DC Characteristics: Operating Current (I_{DD}) (continued)

DC Characteristics	Operating Current		Standard Operating Conditions: 3.0V to 3.6V **(unless otherwise stated)** **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**			
Parameter No.	Typ.[1]	Max.	Units	Conditions		
DC23	9.5	17.5	mA	-40°C	3.3V	100 MHz (N = 1, N2 = 4, N3 = 2, M = 100, F_{VCO} = 800 MHz, F_{PLLO} = 100 MHz)
	10	18	mA	+25°C		
	14	24	mA	+85°C		
	17	29	mA	+125°C		
DC24	11	19	mA	-40°C	3.3V	150 MHz (N = 1, N2 = 2, N3 = 2, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 150 MHz)
	12	21	mA	+25°C		
	15.5	25.5	mA	+85°C		
	19	31.5	mA	+125°C		
DC25	14	24	mA	-40°C	3.3V	200 MHz (N = 1, N2 = 4, N3 = 1, M = 100, F_{VCO} = 800 MHz, F_{PLLO} = 200 MHz)
	14.5	25	mA	+25°C		
	18	30	mA	+85°C		
	21.5	34.5	mA	+125°C		

Notes:

1. Data in the “Typ.” column are for design guidance only and are not tested in manufacturing.
2. I_{DD} is primarily a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature, also have an impact on the current consumption. Base Run current (I_{DD}) is measured as follows:
   - The oscillator is switched to EC+PLL mode in software.
   - The OSCI pin is driven with an external 8 MHz square wave.
   - OSCO is configured as an I/O.
   - The Watchdog Timer is disabled.
   - All I/O pins (except OSCI) are configured as outputs and driving low.
   - No peripheral modules are operating or being clocked (defined PMDx bits are all ‘1’s).
   - JTAG is disabled.
   - `NOP` instructions are executed in a `while(1)` loop.

Table 42-7. Idle Current (I_{IDLE})(2)

DC Characteristics	Idle		Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended			
Parameter No.	Typ.(1)	Max.	Units	Conditions		
DC41	5	11.5	mA	-40°C	3.3V	20 MHz (N = 1, N2 = 7, N3 = 4, M = 70, F_{VCO} = 560 MHz, F_{PLLO} = 20 MHz)
	6	12.5	mA	+25°C		
	10	17	mA	+85°C		
	14.5	21	mA	+125°C		
DC42	5.5	12.5	mA	-40°C	3.3V	50 MHz (N = 1, N2 = 4, N3 = 3, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 50 MHz)
	6.5	13	mA	+25°C		
	11	18	mA	+85°C		
	16.5	24.5	mA	+125°C		
DC43	7	15	mA	-40°C	3.3V	100 MHz (N = 1, N2 = 4, N3 = 2, M = 100, F_{VCO} = 800 MHz, F_{PLLO} = 100 MHz)
	8	15	mA	+25°C		
	12.5	20.5	mA	+85°C		
	18	26.5	mA	+125°C		
DC44	7	14	mA	-40°C	3.3V	150 MHz (N = 1, N2 = 2, N3 = 2, M = 75, F_{VCO} = 600 MHz, F_{PLLO} = 150 MHz)
	8	14.5	mA	+25°C		
	13	21.5	mA	+85°C		
	18.5	27.5	mA	+125°C		
DC45	8.5	16	mA	-40°C	3.3V	200 MHz (N = 1, N2 = 4, N3 = 1, M = 100, F_{VCO} = 800 MHz, F_{PLLO} = 200 MHz)
	9	16	mA	+25°C		
	14	23.5	mA	+85°C		
	20	30	mA	+125°C		

Notes:

1. Data in the “Typ.” column are for design guidance only and are not tested.
2. Base Idle current (I_{IDLE}) is measured as follows:
   - The oscillator is switched to EC+PLL mode in software.
   - The OSCI pin is driven with an external 8 MHz square wave.
   - OSCO is configured as an I/O.
   - FSCM is disabled.
   - The Watchdog Timer is disabled.
   - All I/O pins (except OSCI) are configured as outputs and are driving low.
   - No peripheral modules are operating or being clocked (defined PMDx bits are all ‘1’s).
   - JTAG is disabled.
   - Flash is in standby with NVMPIDL = 1.

Table 42-8. DC Characteristics: Power-Down Current (I_{PD})

DC Characteristics	Sleep		Standard Operating Conditions: 3.0V to 3.6V **(unless otherwise stated)** **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**		
Parameter No.	Typ.(1)	Max.	Units	Conditions	
Power-Down Current (I_{PD})(2)					
DC60	4	8	mA	-40°C	3.3V
	4.5	8.5	mA	+25°C	
	8.5	13.5	mA	+85°C	
	14	20	mA	+125°C	

Notes:

1. Data in the "Typ." column are for design guidance only and are not tested.
2. I_{PD} (Sleep) current is measured as follows:
   - CPU core in Sleep, oscillator is configured in EC mode, and the External Clock is active; OSCI is driven with an external square wave from rail-to-rail (EC clock overshoot/undershoot < 250 mV required).
   - CLKO is configured as an I/O input pin.
   - All I/O pins are configured as output low.
   - $\overline{MCLR}$ = V_{DD}, WDT and FSCM are disabled.
   - All peripheral modules are disabled (PMDx bits are all set).
   - JTAG is disabled.

Table 42-9. DC Characteristics: Watchdog Timer Delta Current (ΔI_{WDT})(1)

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V **(unless otherwise stated)** **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**			
Parameter No.	Typ.	Units	Conditions	
DC61(2)	5	µA	-40°C	3.3V
	5	µA	+25°C	
	7	µA	+85°C	
	10	µA	+125°C	

Notes:

1. Parameter characterized but not tested during manufacturing.
2. ΔI_{WDT} is the additional current consumed when the module is enabled, including the LPRC/BFRC clock source current.

Table 42-10. DC Characteristics: PWM Delta Current[1]

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V **(unless otherwise stated)** **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**				
Parameter No.[2,3]	Typ.	Max.	Units	Conditions	
DC100	2.6	3.6	mA	-40°C, 3.3V	PWM output 500 kHz. PWM input 800 MHz. First instance of PWM generator (PGx).
	2.7	3.7	mA	+25°C, 3.3V	
	2.8	3.9	mA	+85°C, 3.3V	
	3.3	4.5	mA	+125°C, 3.3V	
DC101	1.5	2.1	mA	-40°C, 3.3V	PWM output 500 kHz. PWM input 800 MHz. Additional instance of PWM Generator (PGx).
	1.7	2.3	mA	+25°C, 3.3V	
	1.8	2.5	mA	+85°C, 3.3V	
	2.1	2.9	mA	+125°C, 3.3V	

Notes:
1. Parameters are characterized but not tested in manufacturing.
2. Does not include PLL or CLKGEN currents. PWM in Standard Resolution mode. Both PWMxH and PWMxL pins are driven.
3. Multiple instance PWM current = DC100 + n(DC101) where n is the number of PWM Generators (PGx).

Table 42-11. DC Characteristics: CLKGEN Delta Current[1]

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V **(unless otherwise stated)** **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**				
Parameter No.	Typ	Max	Units	Conditions	
DC150	0.85	1.4	mA	-40°C, 3.3V	Input frequency 800 MHz. Output frequency 400 MHz.
	0.9	1.5	mA	+25°C, 3.3V	
	0.95	1.6	mA	+85°C, 3.3V	
	1.3	2.2	mA	+125°C, 3.3V	
DC151	0.55	1.4	mA	-40°C, 3.3V	Input frequency 400 MHz. Output frequency 400 MHz.
	0.6	1.5	mA	+25°C, 3.3V	
	0.65	1.6	mA	+85°C, 3.3V	
	0.75	1.9	mA	+125°C, 3.3V	
DC152	0.3	1	mA	-40°C, 3.3V	Input frequency 200 MHz. Output frequency 200 MHz.
	0.3	1	mA	+25°C, 3.3V	
	0.6	2	mA	+85°C, 3.3V	
	0.6	2	mA	+125°C, 3.3V	
DC153	0.03	0.8	mA	-40°C, 3.3V	Input frequency 32.768 kHz. Output frequency 32.768 kHz.
	0.03	0.8	mA	+25°C, 3.3V	
	0.03	0.8	mA	+85°C, 3.3V	
	0.03	0.8	mA	+125°C, 3.3V	

Notes:
1. Parameters are characterized but not tested in manufacturing.
2. Multiple instance CLKGEN current = Parameter No. * n, where n is the number of CLKGENs running at the respective frequency.

Table 42-12. DC Characteristics: PLL Delta Current[1]

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V **(unless otherwise stated)** **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**				
Parameter No.[2,3]	Typ.	Max.	Units	Conditions	
DC110	3.5	4.1	mA	-40°C, 3.3V	Input frequency 8 MHz. Output frequency 800 MHz. M = 100, N = 1, N2 = 1, F_{VCO} = 800 MHz.
	3.5	4.1	mA	+25°C, 3.3V	
	3.5	4.1	mA	+85°C, 3.3V	
	4.2	4.9	mA	+125°C, 3.3V	
DC111	2.4	2.6	mA	-40°C, 3.3V	Input frequency 8 MHz. Output frequency 320 MHz. M = 40, N = 1, N2 = 1, F_{VCO} = 640 MHz.
	2.5	2.7	mA	+25°C, 3.3V	
	2.5	2.7	mA	+85°C, 3.3V	
	2.9	3.1	mA	+125°C, 3.3V	
DC112	2.2	2.4	mA	-40°C, 3.3V	Input frequency 8 MHz. Output frequency 200 MHz. M = 100, N = 4, N2 = 1, F_{VCO} = 800 MHz.
	2.2	2.4	mA	+25°C, 3.3V	
	2.2	2.4	mA	+85°C, 3.3V	
	2.6	2.9	mA	+125°C, 3.3V	

Notes:
1. Parameters are characterized but not tested in manufacturing.
2. Values do not include CLKGEN current.
3. If both PLL1 and PLL2 are running, multiply the current for the respective frequency by two to get the total current.

Table 42-13. DC Characteristics: ADC Δ Current[1]

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V **(unless otherwise stated)** **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**				
Parameter No.[2,3]	Typ.	Max.	Units	Conditions	
DC120	2.35	3.3	mA	-40°C, 3.3V	Input frequency 320 MHz. ADC clock 80 MHz. T_{AD} = 12.5 nS. Continuous conversion of single channel of single ADC core. First instance of ADC.
	2.4	3.4	mA	+25°C, 3.3V	
	2.5	3.55	mA	+85°C, 3.3V	
	2.65	3.7	mA	+125°C, 3.3V	
DC121	2.45	3.5	mA	-40°C, 3.3V	Input frequency 320 MHz. ADC clock 80 MHz. T_{AD} = 12.5 nS. Continuous conversion of single channel of single ADC core. Additional instances of ADC.
	2.5	3.65	mA	+25°C, 3.3V	
	2.6	3.8	mA	+85°C, 3.3V	
	2.85	4.1	mA	+125°C, 3.3V	

Notes:
1. Parameters are characterized but not tested in manufacturing.
2. This does not include PLL or CLKGEN currents.
3. Multiple instance ADC current = DC120 + n(DC121) where n is the number of ADC instances.

Table 42-14. DC Characteristics: Comparator + DAC Delta Current(1)

DC Characteristics	Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended				
Parameter No.(2,3,4)	Typ.	Max.	Units	Conditions	
DC130	1.8	2.5	mA	-40°C, 3.3V	Input frequency 500 MHz First instance of DAC + Comparator.
	1.95	2.65	mA	+25°C, 3.3V	
	2.1	3.0	mA	+85°C, 3.3V	
	2.36	3.3	mA	+125°C, 3.3V	
DC131	1.68	2.2	mA	-40°C, 3.3V	Input frequency 500 MHz Additional instance of DAC + Comparator.
	1.83	2.4	mA	+25°C, 3.3V	
	2.0	2.6	mA	+85°C, 3.3V	
	2.25	2.9	mA	+125°C, 3.3V	

Notes:
1. Parameters are characterized but not tested in manufacturing.
2. DAC output pin driven to $V_{DD}/2$. CMP input at V_{SS}.
3. Does not include PLL or CLKGEN currents.
4. Multiple instance DAC+CMP current = DC130 + n(DC131) where n is the number of DAC+CMP instances.

Table 42-15. Op Amp Delta Current(1)

Parameter No.(2,3,4)	Typ.	Max.	Units	Conditions	
DC140	0.5	0.7	mA	-40°C, 3.3V	High-Power mode first instance of Op Amp
	0.55	0.75	mA	+25°C, 3.3V	
	0.6	0.85	mA	+85°C, 3.3V	
	0.9	1.2	mA	+125°C, 3.3V	
DC141	0.5	0.7	mA	-40°C, 3.3V	High-Power mode additional instance of Op Amp
	0.55	0.75	mA	+25°C, 3.3V	
	0.6	0.85	mA	+85°C, 3.3V	
	0.9	1.2	mA	+125°C, 3.3V	
DC142	0.1	0.3	mA	-40°C, 3.3V	Low-Power mode first instance of Op Amp
	0.1	0.3	mA	+25°C, 3.3V	
	0.1	0.3	mA	+85°C, 3.3V	
	0.13	0.33	mA	+125°C, 3.3V	
DC143	0.1	0.3	mA	-40°C, 3.3V	Low-Power mode additional instance of OpAmp
	0.1	0.3	mA	+25°C, 3.3V	
	0.1	0.3	mA	+85°C, 3.3V	
	0.13	0.33	mA	+125°C, 3.3V	

Notes:
1. Parameters are characterized but not tested in manufacturing.
2. Output pin enabled. Inverting input at $V_{DD}/2$.
3. This does not include PLL or CLKGEN currents.
4. Multiple instance Op Amp current = DC140 + n(DC141) where n is the number of Op Amp instances.

Table 42-16. I/O Pin Input Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**							
Parameter No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
DI10	V_{IL}	Input Low Voltage					
		Any I/O Pin and $\overline{MCLR}$	V_{SS}	—	0.2 V_{DD}	V	—
		I/O Pins with SDAx, SCLx	V_{SS}	—	0.3 V_{DD}	V	SMBus disabled.
		I/O Pins with SDAx, SCLx(2)	V_{SS}	—	0.8	V	SMBus enabled.
		I/O Pins with SDAx, SCLx(2)	V_{SS}	—	0.8	V	SMBus 3.0
DI20	V_{IH}	Input High Voltage					
		I/O Pins Not 5V Tolerant(1)	0.8 V_{DD}	—	V_{DD}	V	—
		5V Tolerant I/O Pins and $\overline{MCLR}$(1)	0.8 V_{DD}	—	5.5	V	—
		5V Tolerant I/O Pins with SDAx, SCLx(1)	0.8 V_{DD}	—	5.5	V	SMBus disabled.
		5V Tolerant I/O Pins with SDAx, SCLx(1,2)	2.1	—	5.5	V	SMBus enabled.
		5V Tolerant I/O Pins with SDAx, SCLx(1,2)	1.35	—	5.5	V	SMBus 3.0
		I/O Pins with SDAx, SCLx Not 5V Tolerant(1)	0.8 V_{DD}	—	V_{DD}	V	SMBus disabled.
		I/O Pins with SDAx, SCLx Not 5V Tolerant(1,2)	2.1	—	V_{DD}	V	SMBus enabled.
		I/O Pins with SDAx, SCLx Not 5V Tolerant(1,2)	1.35	—	V_{DD}	V	SMBus 3.0
DI30	I_{CNPU}	Input Change Notification Pull-up Current	175	300	545	µA	V_{DD} = 3.3V, V_{PIN} = V_{SS}
DI31	I_{CNPD}	Input Change Notification Pull-Down Current	200	300	380	µA	V_{DD} = 3.3V, V_{PIN} = V_{DD}

Notes:
1. See Pin Diagrams for the 5V tolerant I/O pins.
2. Parameters are not characterized or tested in manufacturing.

Table 42-17. I/O Pin Input Leakage Specifications(3)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**						
Parameter No.	Symbol	Characteristic	Min.	Max.	Units	Conditions
DI50	I_{IL}	Input Leakage Current(1)				
		I/O Pins 5V Tolerant(2)	-1.5	7.5	µA	
		I/O Pins Not 5V Tolerant(2)	-1	1	µA	—
		$\overline{MCLR}$	-1	7.5	µA	—
		OSCI	-1	1	µA	XT mode

Notes:
1. Negative current is defined as the current sourced by the pin.
2. See Pin Diagrams for the 5V tolerant I/O pins.
3. V_{PIN} = V_{SS} or V_{DD}

Table 42-18. I/O Pin Output Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**							
Parameter No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
DO10	V_{OL}	Output Low-Voltage 4x Sink Driver Pins	—	—	0.4	V	V_{DD} = 3.3V, I_{OL} < 5 mA
		Output Low-Voltage 8x Sink Driver Pins[(1)]	—	—	0.4	V	V_{DD} = 3.3V, I_{OL} < 10 mA
DO20	V_{OH}	Output High-Voltage 4x Source Driver Pins	2.7	—	—	V	V_{DD}= 3.3V, I_{OH} > -5 mA
		Output High-Voltage 8x Source Driver Pins[(1)]	2.7	—	—	V	V_{DD} = 3.3V, I_{OH} > -10 mA

Note:
1. See Pin Diagrams to identify the pin current capability of either 4x or 8x.

Table 42-19. Program Memory Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended**						
Parameter No.	Symbol	Characteristic	Min.	Max.	Units	Conditions
		Program Flash Memory				
D130	E_P	Cell Endurance	10,000	—	E/W	-40°C to +125°C
D131	V_{PR}	V_{DD} for Read	3.0	3.6	V	—
D132b	V_{PEW}	V_{DD} for Self-Timed Write	3.0	3.6	V	—
D134	T_{RETD}	Characteristic Retention	20	—	Year	Provided no other specifications are violated, -40°C to +125°C.
Note: Programming time is non-deterministic and not specified. Refer to Operation for more information.						

42.2. AC Characteristics and Timing Parameters

Figure 42-1. Load Condition for Device Timing Specifications

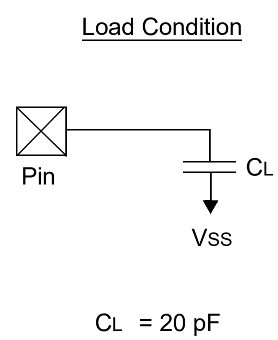


Table 42-20. Capacitive Loading on Output Pins

Parameter No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
DO50	C_{OSCO}	OSCO Pin	—	—	20	pF	In XT mode, when External Clock is used to drive OSCI
DO56	C_{IO}	All I/O Pins and OSCO	—	—	20	pF	EC mode
DO58	C_B	SCLx, SDAx	—	—	400	pF	In I^2C mode

Figure 42-2. External Clock Timing

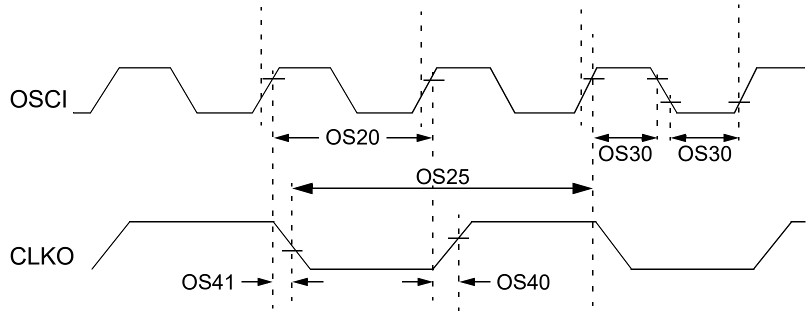


Table 42-21. External Clock Timing Requirements

Parameter No.	Symbol	Characteristic	Min.	Typ.[1]	Max.	Units	Conditions
Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
OS10	F_{IN}	External CLKI Frequency (External Clocks allowed only in EC mode)	DC	—	64	MHz	EC
		Oscillator Crystal Frequency	3.5	—	32	MHz	XT
OS11	F_{REFI}	REFI External Clock Input	—	—	50	MHz	
OS20	T_{OSC}	T_{OSC} = 1/F_{OSC}	15.6	—	—	ns	
OS25	T_{CY}	Instruction Cycle Time	5	—	—	ns	
OS30	$T_{OS}L$, $T_{OS}H$	External Clock in (OSCI) High or Low Time	0.45 x T_{OSC}	—	0.55 x T_{OSC}	ns	EC
OS40	$T_{CK}R$	CLKO Rise Time	—	—	—	ns	Refer to DO31
OS41	$T_{CK}F$	CLKO Fall Time	—	—	—	ns	Refer to DO32
OS42	G_M	External Oscillator Transconductance[3]	2.8	—	3.7	mA/V	PXTALCFG[1:0] = 00, PXTALCFG[2] = 0
			4.5	—	6.2	mA/V	PXTALCFG[1:0] = 00, PXTALCFG[2] = 1
			4.5	—	6.2	mA/V	PXTALCFG[1:0] = 01, PXTALCFG[2] = 0
			7	—	10.5	mA/V	PXTALCFG[1:0] = 01, PXTALCFG[2] = 1
			5.8	—	8.5	mA/V	PXTALCFG[1:0] = 10, PXTALCFG[2] = 0
			8.9	—	14.1	mA/V	PXTALCFG[1:0] = 10, PXTALCFG[2] = 1
			7	—	10.5	mA/V	PXTALCFG[1:0] = 11, PXTALCFG[2] = 0
			10.7	—	17.3	mA/V	PXTALCFG[1:0] = 11, PXTALCFG[2] = 1

Notes:
1. Data in the “Typ.” column are at 3.3V, +25°C unless otherwise stated.
2. Measurements are taken in EC mode. The CLKO signal is measured on the OSCO pin.
3. Parameters are for design guidance only and have not been tested in manufacturing.

Table 42-22. PLLn Timing Specifications[1]

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
OS50	F_{PLLI}	PLL Voltage Controlled Oscillator (VCO) Input Frequency Range	5	—	64	MHz	
OS51	F_{VCO}	On-Chip VCO System Frequency	500	—	1600	MHz	
OS52	M	Feedback Divider Value	16	—	320	N/A	
OS53	T_{LOCK}	PLL Start-up Time (Lock Time)[2]	—	125	188	µs	F_{PLLI} = 8 MHz
OS54	DCLK	CLKO Stability (Jitter)	—	1	—	ps	F_{VCO} = 1600 MHz, F_{OUT} = 400 MHz
OS55	F_{OUT}	PLL PostDiv Output	—	—	800	MHz	
OS56	F_{OUTDIV}	PLL VCO FractDiv Output	—	—	800	MHz	
OS57	F_{INDIV}	PLL FracDiv Input	500	—	800	MHz	

Notes:
1. Parameters are for design guidance only and have not been tested in manufacturing.
2. Parameters are characterized but not tested in manufacturing.

Table 42-23. Peripheral Input Clock Timing Specifications[1]

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended					
Parameter No.	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
F_{INMAX}	CLKGEN	—	800	MHz	CLKGEN input
	PWM	—	800	MHz	
	ADC	32	320	MHz	
	DAC	400	500	MHz	
	DMT	—	200	MHz	
	MCCP	—	200	MHz	System clock
	SPI	—	320	MHz	
	UART	—	320	MHz	
	PTG	—	400	MHz	
	CLC	—	200	MHz	System clock
	I2C	—	100	MHz	
	SENT	—	100	MHz	
	BISS	—	320	MHz	
	CRC	—	200	MHz	
	QEI	—	100	MHz	
	PPS	—	50	MHz	
	GPIO	—	200	MHz	Fast speed clock for PORTx and LATx registers
		—	50	MHz	Slow Speed clock, all other GPIO registers
	JTAG	—	20	MHz	TCK max input clock
	I3C	—	320	MHz	
	RDC	—	100	MHz	
Note: 1. Parameters are for design guidance only and are not tested.					

Table 42-24. Internal FRC Accuracy

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended					
Param No.	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
Internal FRC Accuracy @ FRC Frequency = 8 MHz(1)					
F20	FRC	-1.5	1.5	%	-40°C ≤ T_A ≤ 85°C
		-2	2	%	+85°C ≤ T_A ≤ +125°C
F21	BFRC	-2	2	%	-40°C ≤ T_A ≤ +125°C
F22	BFRC/244 (LPRC)	-3	3	%	-40°C ≤ T_A ≤ +125°C

Note:
1. Frequency is calibrated at +25°C and 3.3V.

Figure 42-3. I/O Timing Characteristics

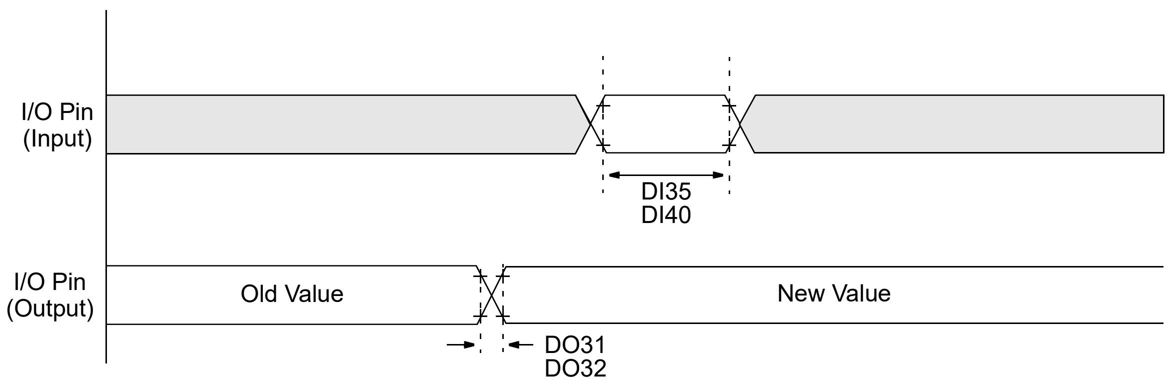


Table 42-25. I/O Timing Requirements

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.(1)**	**Max.**	**Units**	**Conditions**
DO31	$T_{IO}R$	Port Output Rise Time(2)	—	3.2	—	ns	4x output pins, 25 pF load
			—	2.3	—	ns	8x output pins, 25 pF load
DO32	$T_{IO}F$	Port Output Fall Time(2)	—	2.5	—	ns	4x output pins, 25 pF load
			—	1.7	—	ns	8x output pins, 25 pF load
DI35	T_{INP}	INTx Pin High or Low Time (input)(2)	6	—	—	ns	
DI40	T_{RBP}	CNx High or Low Time (input)(2)	2	—	—	T_{CY}	

Notes:
1. Data in the “Typ.” column are at 3.3V, +25°C unless otherwise stated.
2. Parameters are characterized but not tested in manufacturing.

Table 42-25. I/O Timing Requirements (continued)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial** **-40°C ≤ T_A ≤ +125°C for Extended**							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.[1]**	**Max.**	**Units**	**Conditions**
DI45	T_{PORT}	Transistion Delay from Input to PORTx Register	2	—	—	T_{CY}	

Notes:
1. Data in the “Typ.” column are at 3.3V, +25°C unless otherwise stated.
2. Parameters are characterized but not tested in manufacturing.

Figure 42-4. BOR and Master Clear Reset Timing Characteristics

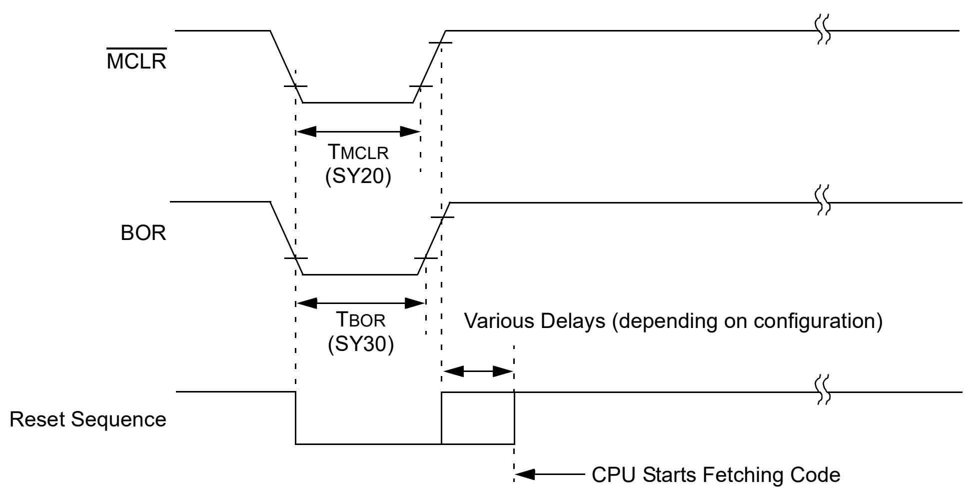


Table 42-26. Reset and Watchdog Timer Timing Requirements

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial** **-40°C ≤ T_A ≤ +125°C for Extended**							
Param eter No.	**Symbol**	**Characteristic[1]**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
SY00	T_{PU}	Power-up Period	—	80	—	µs	BOR trip point to first instruction on FRC
SY13	T_{IOZ}	I/O High-Impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset	—	1	—	µs	
SY20	T_{MCLR}	$\overline{\text{MCLR}}$ Pulse Width (low)	8	—	—	µs	
SY35	T_{FSCM}	Fail-Safe Clock Monitor Delay	—	—	2	µs	Clock fail to BFRC ready
SY40	F_{MIN}	Fail-Safe Clock Monitor Minimum Frequency	4	—	—	MHz	

Note:
1. Parameters are characterized but not tested in manufacturing.

Figure 42-5. High-Speed PWMx Module Fault Timing Characteristics

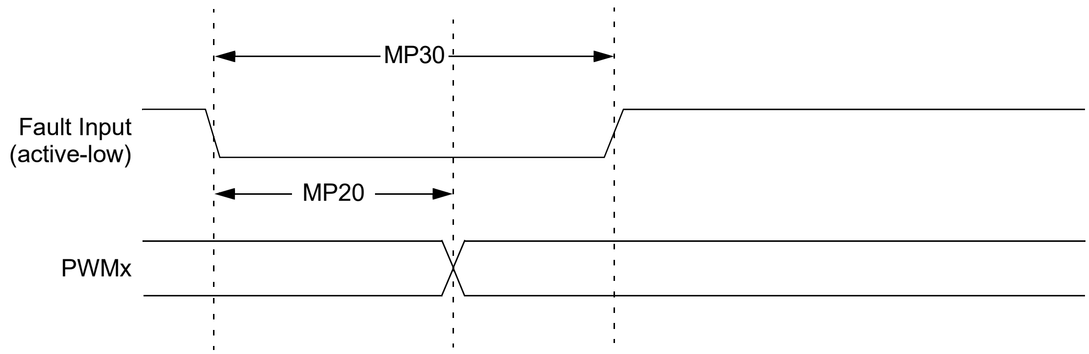


Figure 42-6. High-Speed PWMx Module Timing Characteristics

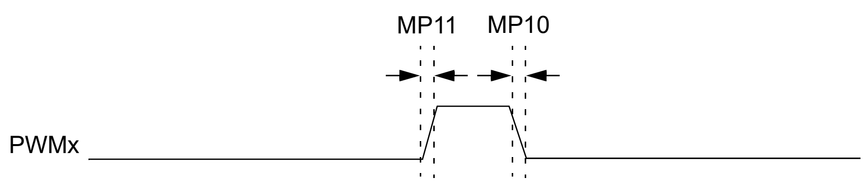


Table 42-27. High-Speed PWMx Module Timing Requirements

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial** **-40°C ≤ T_A ≤ +125°C for Extended**							
Parameter No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
MP10	T_{FPWM}	PWMx Output Fall Time	—	—	—	ns	See Parameter DO32
MP11	T_{RPWM}	PWMx Output Rise Time	—	—	—	ns	See Parameter DO31
MP20	T_{FD}	Fault Input ↓ to PWMx I/O Change[(1)]	—	15	—	ns	PCI Inputs 19 through 22, level triggered
MP30	T_{FH}	Fault Input Pulse Width[(1)]	4	—	—	ns	

Note:

1. These parameters are characterized but not tested in manufacturing.

Figure 42-7. QEI Module Index Pulse Timing Characteristics

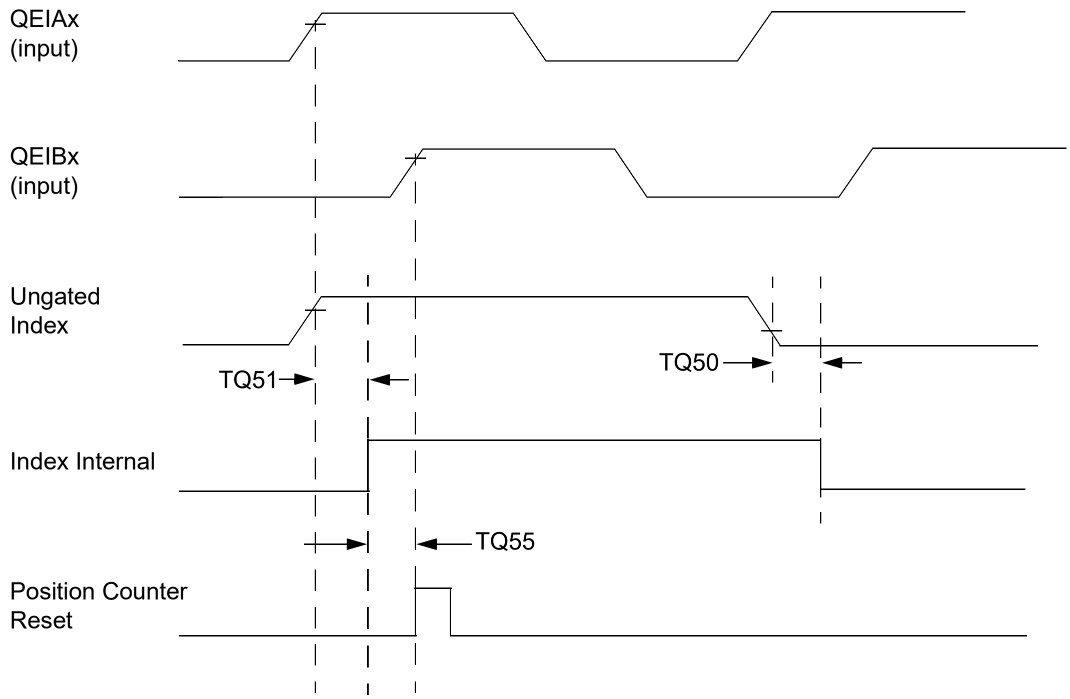


Table 42-28. QEI Index Pulse Timing Requirements

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended						
Param eter No.	Symbol	Characteristic(1)	Min.	Max	Units	Conditions
TQ50	TqIL	Filter Time to Recognize Low with Digital Filter(2)	3 * N * T_{CY}	—	ns	N = 1, 2, 4, 16, 32, 64, 128 and 256
TQ51	TqiH	Filter Time to Recognize High with Digital Filter(2)	3 * N * T_{CY}	—	ns	N = 1, 2, 4, 16, 32, 64, 128 and 256
TQ55	Tqidxr	Index Pulse Recognized to Position Counter Reset (ungated index)	3 T_{CY}	—	ns	

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Alignment of index pulses to QEIA and QEIB is shown for position counter reset timing only. Shown for the forward direction only (QEIA leads QEIB). The same timing applies for the reverse direction (QEIA lags QEIB), but index pulse recognition occurs on the falling edge.

Table 42-29. SPIx Maximum Data/Clock Rate Summary

SPI Host Transmit Only (Half-Duplex)	SPI Host Transmit/ Receive (Full-Duplex)	SPI Client Transmit/ Receive (Full-Duplex)	CKE	Maximum Data Rate (MHz)	Condition
Figure 42-8 Figure 42-9	—	—	0	25	Using PPS
				50	Dedicated Pin
Figure 42-9 Table 42-30	—	—	1	25	Using PPS
				50	Dedicated Pin
—	Figure 42-10 Table 42-31	—	0	25	Using PPS
				50	Dedicated Pin
—	Figure 42-11 Table 42-32	—	1	25	Using PPS
				50	Dedicated Pin
—	—	Figure 42-12 Table 42-33	0	25	Using PPS
				50	Dedicated Pin
—	—	Figure 42-13 Table 42-34	1	25	Using PPS
				50	Dedicated Pin

Figure 42-8. SPIx Host Mode (Half-Duplex, Transmit Only, CKE = 0) Timing Characteristics

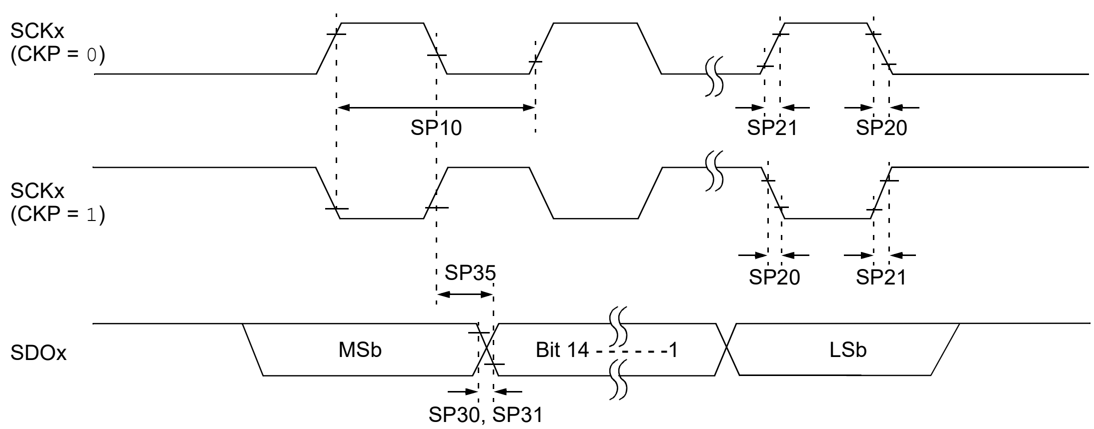


Figure 42-9. SPIx Host Mode (Half-Duplex, Transmit Only, CKE = 1) Timing Characteristics

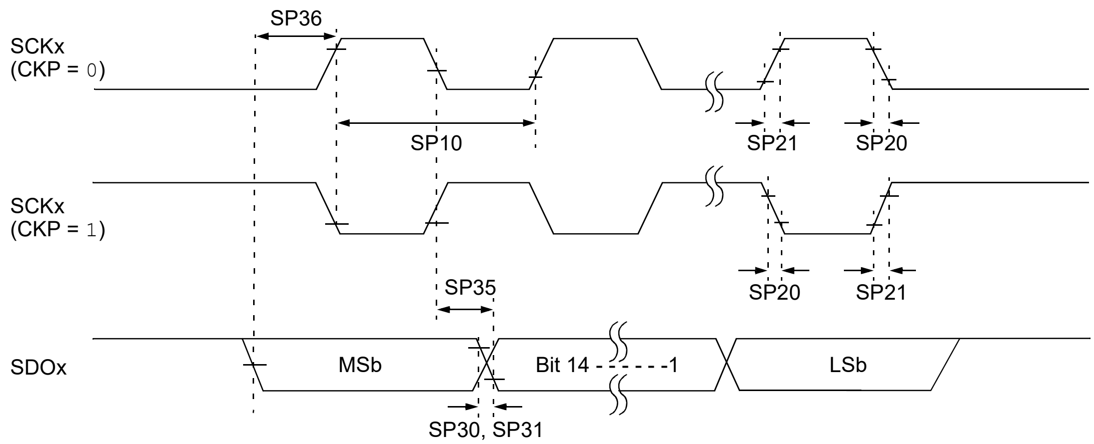


Table 42-30. SPIx Host Mode (Half-Duplex, Transmit Only) Timing Requirements(1)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial** **-40°C ≤ T_A ≤ +125°C for Extended**							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.(2)**	**Max.**	**Units**	**Conditions**
SP10	$F_{SC}P$	Maximum SCKx Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP20	$T_{SC}F$	SCKx Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5		SPI2 dedicated pins
SP21	$T_{SC}R$	SCKx Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$(3)	SDOx Data Output Fall Time	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$(3)	SDOx Data Output Rise Time	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI dedicated pins
SP36	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	SDOx Data Output Setup to First SCKx Edge	10	—	—	ns	Using PPS pins
			1	—	2	ns	SPI2 dedicated pins

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in the “Typ.” column are at 3.3V, +25°C unless otherwise stated.
3. Assumes a 30 pF load on all SPIx pins.

Figure 42-10. SPIx Host Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 1) Timing Characteristics

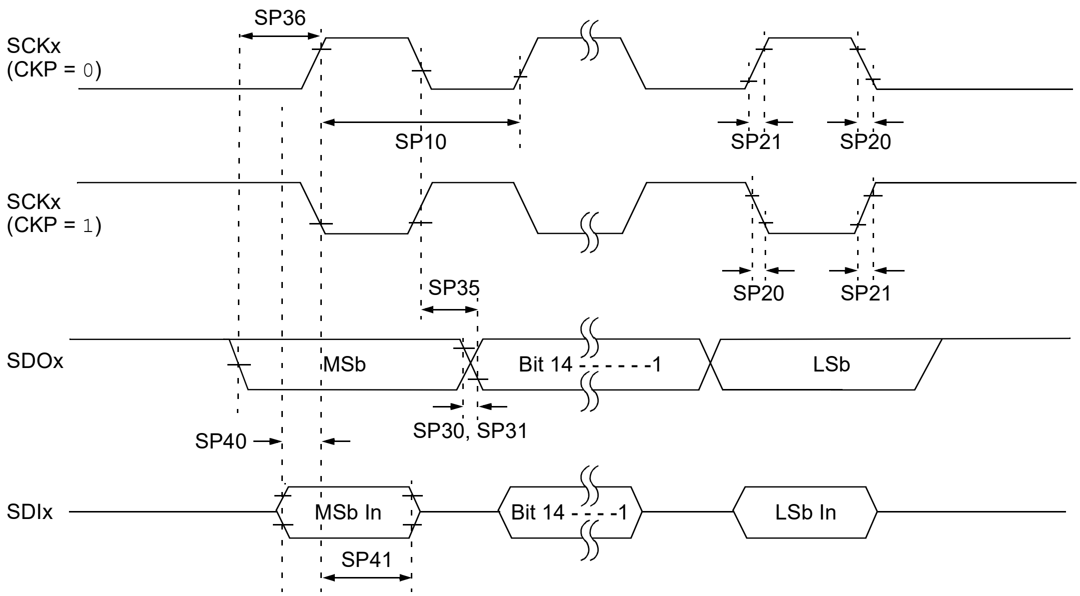


Table 42-31. SPIx Host Mode (Full-Duplex, CKE = 1, CKP = X, SMP = 1) Timing Requirements(1)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.(2)**	**Max.**	**Units**	**Conditions**
SP10	$F_{SC}P$	Maximum SCKx Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP20	$T_{SC}F$	SCKx Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP21	$T_{SC}R$	SCKx Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$	SDOx Data Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$	SDOx Data Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI2 dedicated pins
SP36	$T_{DO}V2_{SC}$, $T_{DO}V2_{SC}L$	SDOx Data Output Setup to First SCKx Edge	10	—	—	ns	Using PPS pins
			1	—	2	ns	SPI2 dedicated pins

Notes:
1. These parameters are characterized but not tested in manufacturing.
2. Data in the “Typ.” column are at 3.3V, +25°C unless otherwise stated.
3. Assumes a 30 pF load on all SPIx pins.

Table 42-31. SPIx Host Mode (Full-Duplex, CKE = 1, CKP = X, SMP = 1) Timing Requirements[1] (continued)

Parameter No.	Symbol	Characteristic	Min.	Typ.[2]	Max.	Units	Conditions
Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial** **-40°C ≤ T_A ≤ +125°C for Extended**							
SP40	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	Setup Time of SDIx Data Input to SCKx Edge	—	—	17	ns	Using PPS pins
			0.1	—	0.6	ns	SPI2 dedicated pins
SP41	$T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$	Hold Time of SDIx Data Input to SCKx Edge	2	—	—	ns	Using PPS pins
			2.5	—	3.3	ns	SPI2 dedicated pins

Notes:
1. These parameters are characterized but not tested in manufacturing.
2. Data in the “Typ.” column are at 3.3V, +25°C unless otherwise stated.
3. Assumes a 30 pF load on all SPIx pins.

Figure 42-11. SPIx Host Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 1) Timing Characteristics

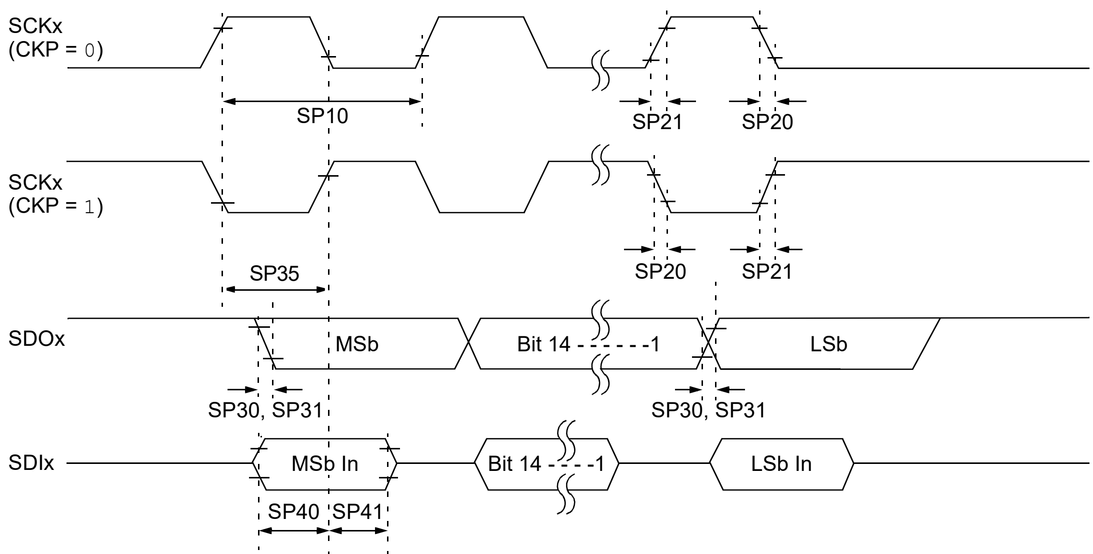


Table 42-32. SPIx Host Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 1) Timing Requirements(1)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristic**	**Min.**	**Typ.(2)**	**Max.**	**Units**	**Conditions**
SP10	$F_{SC}P$	Maximum SCKx Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP20	$T_{SC}F$	SCKx Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP21	$T_{SC}R$	SCKx Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$	SDOx Data Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$	SDOx Data Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI2 dedicated pins
SP40	$TdiV2_{SC}H$, $TdiV2_{SC}L$	Setup Time of SDIx Data Input to SCKx Edge	—	—	17	ns	Using PPS pins
			0.1	—	0.6	ns	SPI2 dedicated pins
SP41	$T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$	Hold Time of SDIx Data Input to SCKx Edge	2	—	—	ns	Using PPS pins
			2.5	—	3.3	ns	SPI2 dedicated pins

Notes:
1. These parameters are characterized but not tested in manufacturing.
2. Data in the "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes a 30 pF load on all SPIx pins.

Figure 42-12. SPIx Client Mode (Full-Duplex, CKE = 0, CKP = x, SMP = 0) Timing Characteristics

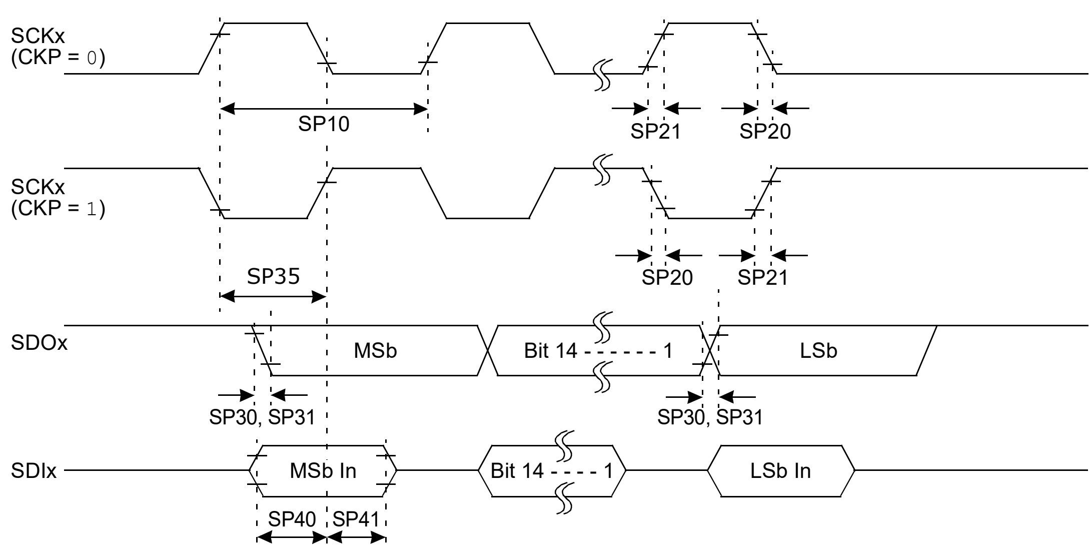


Table 42-33. SPIx Client Mode (Full-Duplex, CKE = `0`, CKP = `x`, SMP = `0`) Timing Requirements[1]

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.[2]**	**Max.**	**Units**	**Conditions**
SP10	$F_{SC}P$	Maximum SCKx Input Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP72	$T_{SC}F$	SCKx Input Fall Time[3]	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP73	$T_{SC}R$	SCKx Input Rise Time[3]	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$	SDOx Data Output Fall Time[3]	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$	SDOx Data Output Rise Time[3]	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI2 dedicated pins
SP40	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	Setup Time of SDIx Data Input to SCKx Edge	—	—	17	ns	Using PPS pins
			1	—	1	ns	SPI2 dedicated pins
SP41	$T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$	Hold Time of SDIx Data Input to SCKx Edge	2	—	—	ns	Using PPS pins
			2.5	—	3.3	ns	SPI2 dedicated pins
SP50	$T_{SS}L2_{SC}H$, $T_{SS}L2_{SC}L$	$\overline{SSx}$ ↓ to SCKx ↑ or SCKx ↓ Input	90	—	—	ns	
SP51	$T_{SS}H2_{DO}Z$	$\overline{SSx}$ ↑ to SDOx Output High-Impedance	5	—	20	ns	
SP52	$T_{SC}H2_{SS}H$, $T_{SC}L2_{SS}H$	$\overline{SSx}$ ↑ After SCKx Edge[3]	1.5 T_{CY} + 40	—	—	ns	

Notes:
1. These parameters are characterized but not tested in manufacturing.
2. Data in the "Typ." column are at 3.3V, +25°C unless otherwise stated.
3. Assumes a 30 pF load on all SPIx pins.

Figure 42-13. SPIx Client Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 0) Timing Characteristics

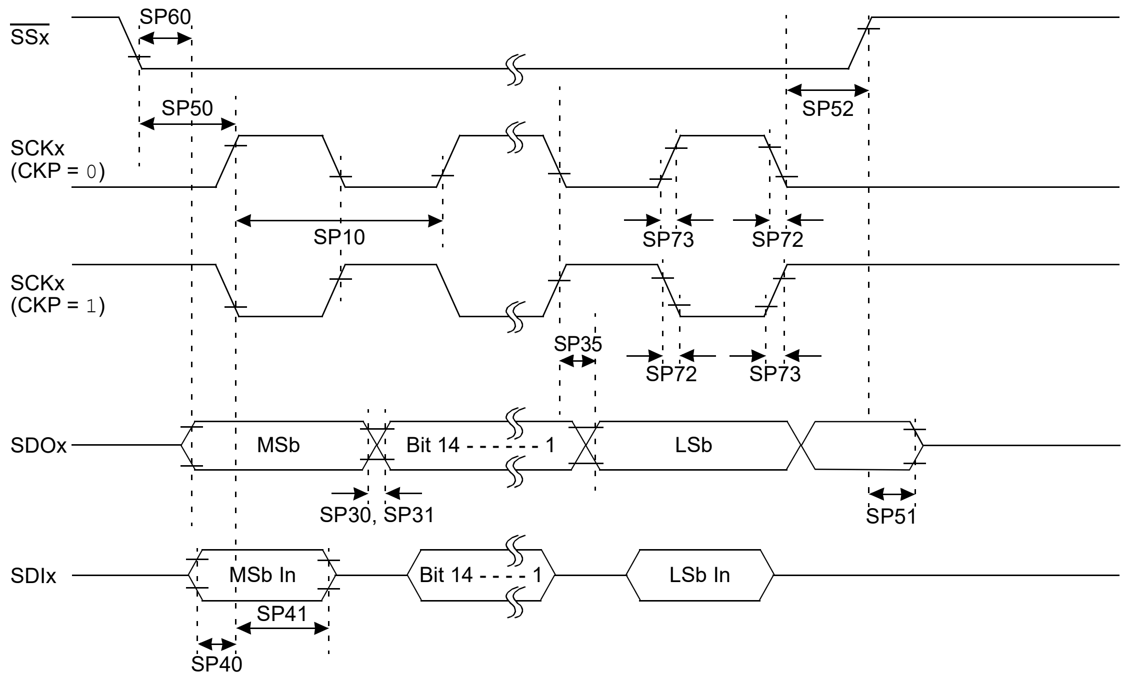


Table 42-34. SPIx Client Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 0) Timing Requirements(1)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) **Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial** **-40°C ≤ T_A ≤ +125°C for Extended**							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.(2)**	**Max.**	**Units**	**Conditions**
SP10	$F_{SC}P$	Maximum SCKx Input Frequency	—	—	25	MHz	Using PPS pins
			—	—	50	MHz	SPI2 dedicated pins
SP72	$T_{SC}F$	SCKx Input Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP73	$T_{SC}R$	SCKx Input Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins
SP30	$T_{DO}F$	SDOx Data Output Fall Time(3)	—	—	5	ns	Using PPS pins
			—	—	2.5	ns	SPI2 dedicated pins
SP31	$T_{DO}R$	SDOx Data Output Rise Time(3)	—	—	6	ns	Using PPS pins
			—	—	3.1	ns	SPI2 dedicated pins

Notes:

1. These parameters are characterized but not tested in manufacturing.
2. Data in the “Typ.” column are at 3.3V, +25°C unless otherwise stated.
3. Assumes a 30 pF load on all SPIx pins.

Table 42-34. SPIx Client Mode (Full-Duplex, CKE = 1, CKP = x, SMP = 0) Timing Requirements[1] (continued)

Parameter No.	Symbol	Characteristic	Min.	Typ.[2]	Max.	Units	Conditions
Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
SP35	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDOx Data Output Valid After SCKx Edge	—	—	7	ns	Using PPS pins
			2.7	—	4.4	ns	SPI dedicated pins
SP40	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	Setup Time of SDIx Data Input to SCKx Edge	—	—	17	ns	Using PPS pins
			0.1	—	0.6	ns	SPI2 dedicated pins
SP41	$T_{SC}H2diL$, $T_{SC}L2_{DI}L$	Hold Time of SDIx Data Input to SCKx Edge	2	—	—	ns	Using PPS pins
			2.5	—	3.3	ns	SPI2 dedicated pins
SP50	$T_{SS}L2_{SC}H$, TssL2scL	$\overline{SSx}$ ↓ to SCKx ↑ or SCKx ↓ Input	90	—	—	ns	
SP51	$T_{SS}H2doZ$	$\overline{SSx}$ ↑ to SDOx Output High-Impedance[3]	5	—	20	ns	
SP52	$T_{SC}H2_{SS}H$, $T_{SC}L2_{SS}H$	$\overline{SSx}$ ↑ After SCKx Edge[3]	1.5 T_{CY} + 40	—	—	ns	
SP60	$T_{SS}L2_{DO}V$	SDOx Data Output Valid After $\overline{SSx}$ Edge	—	—	50	ns	

Notes:
1. These parameters are characterized but not tested in manufacturing.
2. Data in the “Typ.” column are at 3.3V, +25°C unless otherwise stated.
3. Assumes a 30 pF load on all SPIx pins.

Figure 42-14. I2Cx Bus Start/Stop Bits Timing Characteristics (Host Mode)

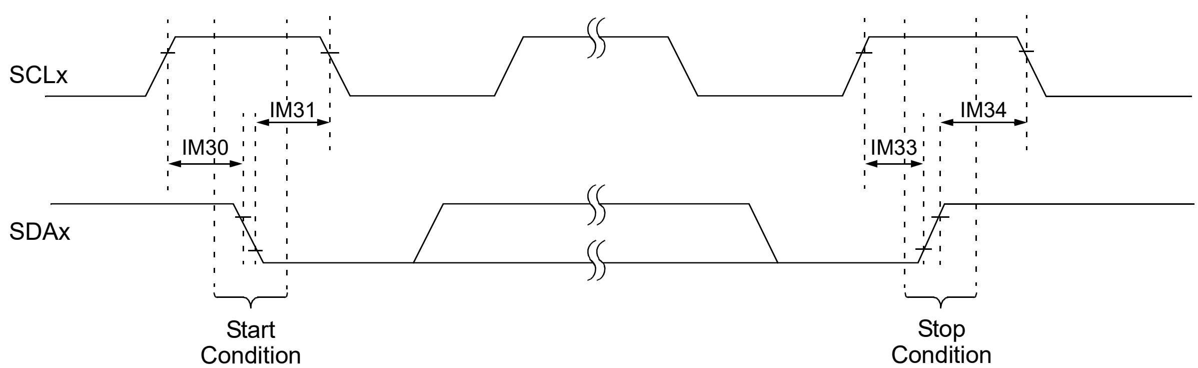


Figure 42-15. I2Cx Bus Data Timing Characteristics (Host Mode)

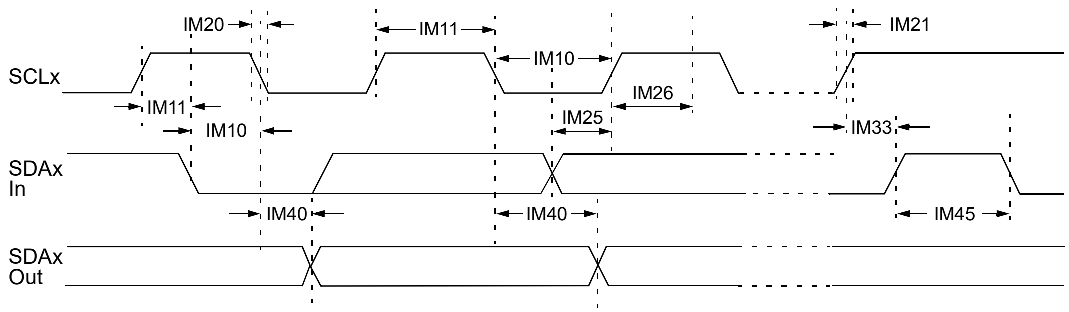


Table 42-35. I2Cx Bus Data Timing Requirements (Host Mode)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic[3]**		**Min.[1]**	**Max.**	**Units**	**Conditions**
IM10	Tlo:scl	Clock Low Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode[2]	T_{CY} (BRG + 1)	—	µs	
IM11	Thi:scl	Clock High Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode[2]	T_{CY} (BRG + 1)	—	µs	
IM20	Tf:scl	SDAx and SCLx Fall Time	100 kHz mode	—	300	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 x (V_{DD}/5.5V)	300	ns	
			1 MHz mode[2]	—	120	ns	
IM21	Tr:scl	SDAx and SCLx Rise Time	100 kHz mode	—	1000	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 Cb	300	ns	
			1 MHz mode[2]	—	120	ns	
IM25	Tsu:dat	Data Input Setup Time	100 kHz mode	250	—	ns	
			400 kHz mode	100	—	ns	
			1 MHz mode[2]	50	—	ns	
IM26	Thd:dat	Data Input Hold Time	100 kHz mode	0	—	µs	
			400 kHz mode	0	0.9	µs	
			1 MHz mode[2]	0	0.3	µs	

Notes:
1. BRG is the value of the I^2C Baud Rate Generator.
2. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
3. These parameters are characterized but not tested in manufacturing.

Table 42-35. I2Cx Bus Data Timing Requirements (Host Mode) (continued)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated)
Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial
-40°C ≤ T_A ≤ +125°C for Extended

Parameter No.	Symbol	Characteristic[3]		Min.[1]	Max.	Units	Conditions
IM30	Tsu:sta	Start Condition Setup Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	Only relevant for Repeated Start condition
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode[2]	T_{CY} (BRG + 1)	—	µs	
IM31	Thd:sta	Start Condition Hold Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	After this period, the first clock pulse is generated
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode[2]	T_{CY} (BRG + 1)	—	µs	
IM33	Tsu:sto	Stop Condition Setup Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode[2]	T_{CY} (BRG + 1)	—	µs	
IM34	Thd:sto	Stop Condition Hold Time	100 kHz mode	T_{CY} (BRG + 1)	—	µs	
			400 kHz mode	T_{CY} (BRG + 1)	—	µs	
			1 MHz mode[2]	T_{CY} (BRG + 1)	—	µs	
IM40	Taa:scl	Output Valid from Clock	100 kHz mode	—	3450	ns	
			400 kHz mode	—	900	ns	
			1 MHz mode[2]	—	450	ns	
IM45	Tbf:sda	Bus Free Time	100 kHz mode	4.7	—	µs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	µs	
			1 MHz mode[2]	0.5	—	µs	
IM50	Cb	Bus Capacitive Loading		—	400	pF	
IM51	T_{PGD}	Input Glitch Filter Delay		65	390	ns	Typical value for this parameter is 130 ns.

Notes:
1. BRG is the value of the I²C Baud Rate Generator.
2. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
3. These parameters are characterized but not tested in manufacturing.

Figure 42-16. I2Cx Bus Start/Stop Bits Timing Characteristics (Client Mode)

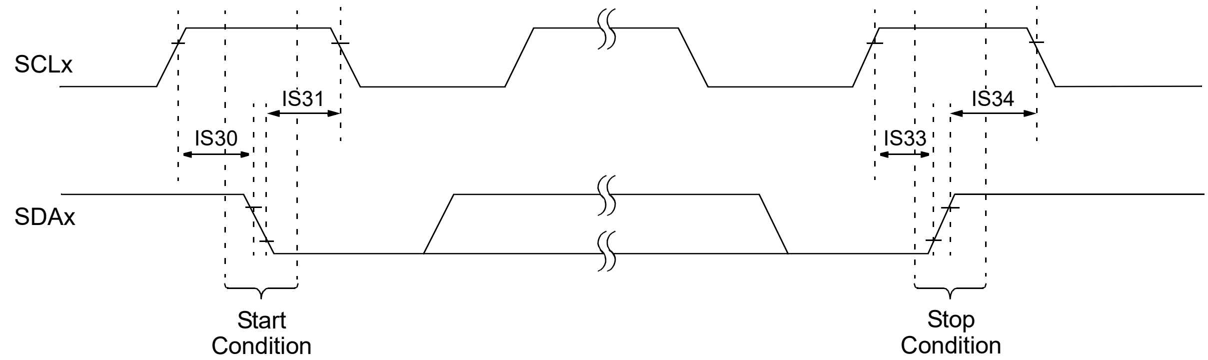


Figure 42-17. I2Cx Bus Data Timing Characteristics (Client Mode)

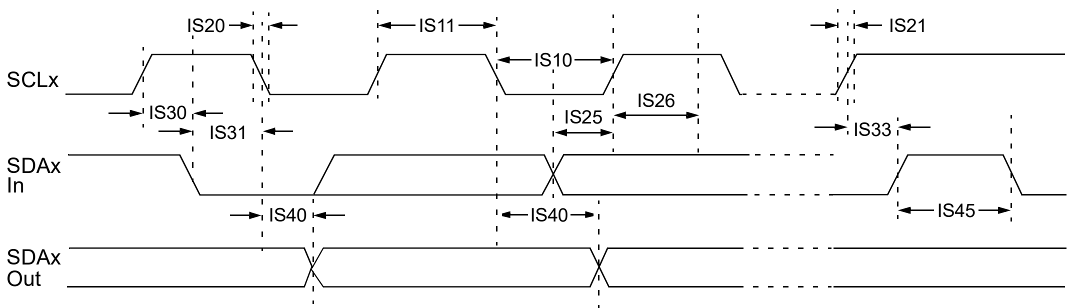


Table 42-36. I2Cx Bus Data Timing Requirements (Client Mode)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic(2)**		**Min.**	**Max.**	**Units**	**Conditions**
IS10	Tlo:scl	Clock Low Time	100 kHz mode	4.7	—	µs	
			400 kHz mode	1.3	—	µs	
			1 MHz mode(1)	0.5	—	µs	
IS11	Thi:scl	Clock High Time	100 kHz mode	4.0	—	µs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	µs	Device must operate at a minimum of 10 MHz
			1 MHz mode(1)	0.28	—	µs	

Notes:
1. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
2. These parameters are characterized but not tested in manufacturing.

Table 42-36. I2Cx Bus Data Timing Requirements (Client Mode) (continued)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	Symbol	Characteristic(2)		Min.	Max.	Units	Conditions
IS20	Tf:scl	SDAx and SCLx Fall Time	100 kHz mode	—	300	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 x (V_{DD}/5.5V)	300	ns	
			1 MHz mode(1)	20 x (V_{DD}/5.5V)	120	ns	
IS21	Tr:scl	SDAx and SCLx Rise Time	100 kHz mode	20 + 0.1 Cb	1000	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	—	300	ns	
			1 MHz mode(1)	—	120	ns	
IS25	Tsu:dat	Data Input Setup Time	100 kHz mode	250	—	ns	
			400 kHz mode	100	—	ns	
			1 MHz mode(1)	50	—	ns	
IS26	Thd:dat	Data Input Hold Time	100 kHz mode	0	—	μs	
			400 kHz mode	0	0.9	μs	
			1 MHz mode(1)	0	0.3	μs	
IS30	Tsu:sta	Start Condition Setup Time	100 kHz mode	4.7	—	μs	Only relevant for Repeated Start condition
			400 kHz mode	0.6	—	μs	
			1 MHz mode(1)	0.26	—	μs	
IS31	Thd:sta	Start Condition Hold Time	100 kHz mode	4.0	—	μs	After this period, the first clock pulse is generated
			400 kHz mode	0.6	—	μs	
			1 MHz mode(1)	0.26	—	μs	
IS33	Tsu:sto	Stop Condition Setup Time	100 kHz mode	4	—	μs	
			400 kHz mode	0.6	—	μs	
			1 MHz mode(1)	0.26	—	μs	
IS34	Thd:sto	Stop Condition Hold Time	100 kHz mode	> 0	—	μs	
			400 kHz mode	> 0	—	μs	
			1 MHz mode(1)	> 0		μs	
IS40	Taa:scl	Output Valid from Clock	100 kHz mode	0	3540	ns	
			400 kHz mode	0	900	ns	
			1 MHz mode(1)	0	400	ns	
IS45	Tbf:sda	Bus Free Time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
			1 MHz mode(1)	0.5	—	μs	
IS50	C_B	Bus Capacitive Loading		—	400	pF	

Notes:

1. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
2. These parameters are characterized but not tested in manufacturing.

Table 42-36. I2Cx Bus Data Timing Requirements (Client Mode) (continued)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended						
Parameter No.	Symbol	Characteristic(2)	Min.	Max.	Units	Conditions
IS51	T_{PGD}	Input Glitch Filter Delay(2)	65	390	ns	Typical value for this parameter is 130 ns.

Notes:

1. Maximum Pin Capacitance = 10 pF for all I2Cx pins (for 1 MHz mode only).
2. These parameters are characterized but not tested in manufacturing.

Table 42-37. I3C Open Drain Timing Parameters

Parameter	Symbol	Min.	Max.	Units	
Low Period of SCL Clock	tLOW_OD	200	–	ns	Figure 42-20
	tDIG_OD_L	tLOW_ODmin + tfDA Odmin	–	ns	Figure 42-20
High Period of SCL Clock (for First Broadcast Address)	tHIGH_INIT	200	–	ns	–
High Period of SCL Clock (for Mixed Bus)	tHIGH	–	41	ns	Figure 42-19
	tDIG_H	–	tHIGH + tCF	ns	Figure 42-19 Figure 42-29
High Period of SCL Clock (for Pure Bus)	tHIGH	24 (tHIGHmin from Push-Pull)	–	ns	Figure 42-19
	tDIG_H	32 (tDIG_Hmin from Push-Pull)	–	ns	Figure 42-19 Figure 42-29
Fall Time of SDA Signal	tfDA_OD	–	12	ns	Figure 42-20
SDA Data Setup Time During Open Drain Mode	tSU_OD	3	–	ns	Figure 42-20
Clock After START (S) Condition	tCAS	38.4 nano	For ENTAS0: 1 µ	seconds	Figure 42-20
			For ENTAS1: 100 µ		–
			For ENTAS2: 2 milli		–
			For ENTAS3: 50 milli		–
Clock Before STOP (P) Condition	tCBP	tCASmin / 2	–	seconds	Figure 42-21
Active Controller to Secondary Controller Overlap time during handoff	tCRHPOverlap	tDIG_OD_Lmin	–	ns	Figure 42-28
Bus Available Condition	tAVAL	1	–	µs	–

Table 42-37. I3C Open Drain Timing Parameters (continued)

Parameter	Symbol	Min.	Max.	Units	
Bus Idle Condition	tIDLE	200	–	µs	–
Time Internal Where New Controller Not Driving SDA Low	tNEWCRLock	tAVALmin	–	µs	Figure 42-28

Table 42-38. I3C Push-Pull Timing Parameters for SDR and HDR-DDR

Parameter	Symbol	Min.	Typ.	Max.	Units	
SCL Clock Frequency	fSCL	0.01	12.5	12.9	MHz	–
SCL Clock Low Period	tLOW	24	–	–	ns	Figure 42-18
	tDIG_L	32	–	–	ns	Figure 42-19
SCL Clock High Period	tHIGH_MIXED	24	–	–	ns	Figure 42-19
(for Mixed Bus)	tDIG_H_MIXED	32	–	45	ns	Figure 42-19
SCL Clock High Period	tHIGH	24	–	–	ns	Figure 42-18
(for Pure Bus)	tDIG_H	32	–	–	ns	Figure 42-19 Figure 42-18
Clock in to Data Out for Target	tSCO	–	–	12	ns	Figure 42-22
SCL Clock Rise Time	tCR	–	–	150e06 * 1 / fSCL (capped at 60)	ns	Figure 42-19
SCL Clock Fall Time	tCF	–	–	150e06 * 1 / fSCL (capped at 60)	ns	Figure 42-22
SDA Signal Data Hold in Push-Pull Mode (Controller)	tHD_PP	tCR + 3 and tCF + 3	–	–	–	
SDA Signal Data Hold in Push-Pull Mode (Target)	tHD_PP	0	–	–	–	Figure 42-23 Figure 42-24 Figure 42-25 Figure 42-26
SDA Signal Data Setup in Push-Pull Mode	tSU_PP	3	–	N/A	ns	Figure 42-23
Clock After Repeated START (Sr) Condition	tCASr	tCASmin / 2	–	N/A	ns	Figure 42-27
Clock Before Repeated START (Sr) Condition	tCBSr	tCASmin / 2	–	N/A	ns	Figure 42-27
Capacitive Load per Bus Line (SDA/SCL)	Cb	–	–	50	pF	–

Figure 42-18. I3C Legacy Mode Timing

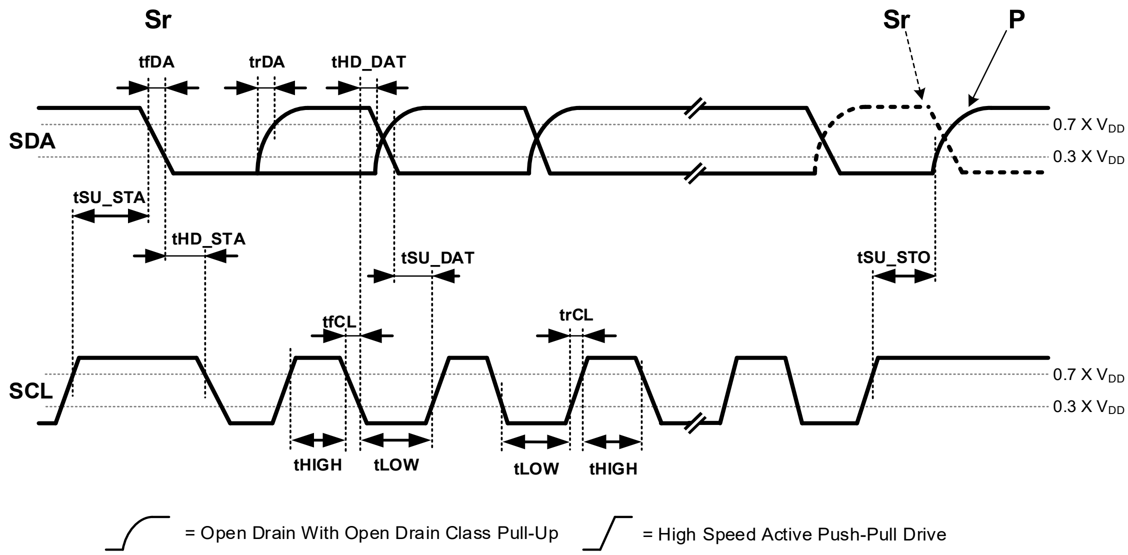


Figure 42-19. t_{DIG_H} and t_{DIG_L}

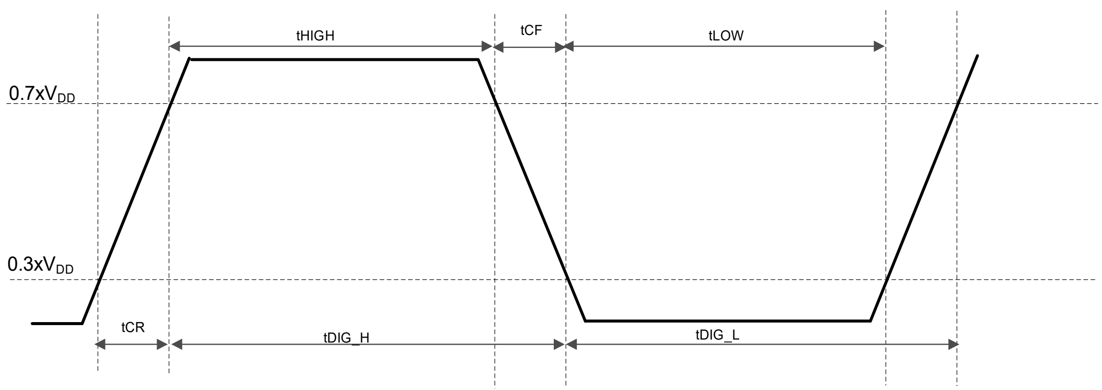


Figure 42-20. I3C START Timing

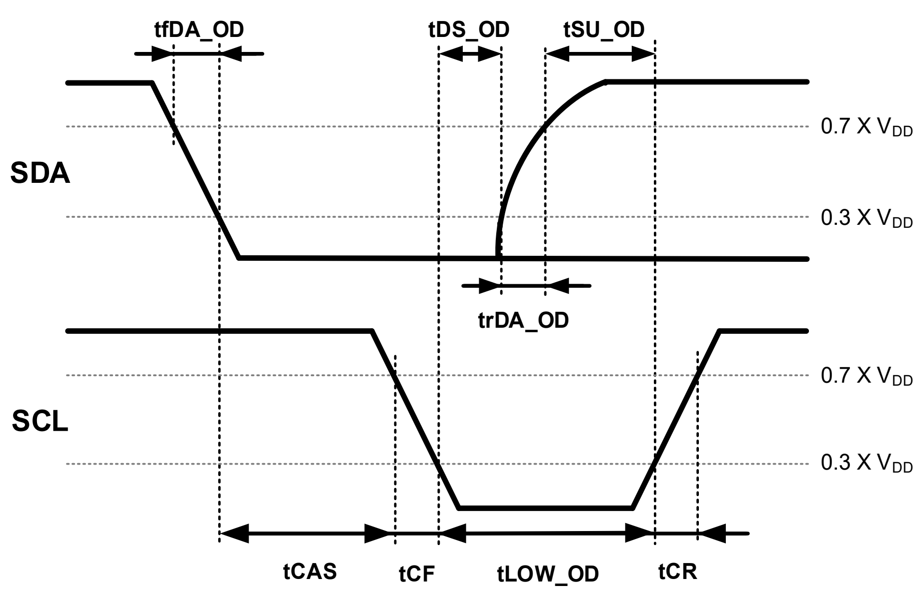


Figure 42-21. I3C STOP Timing

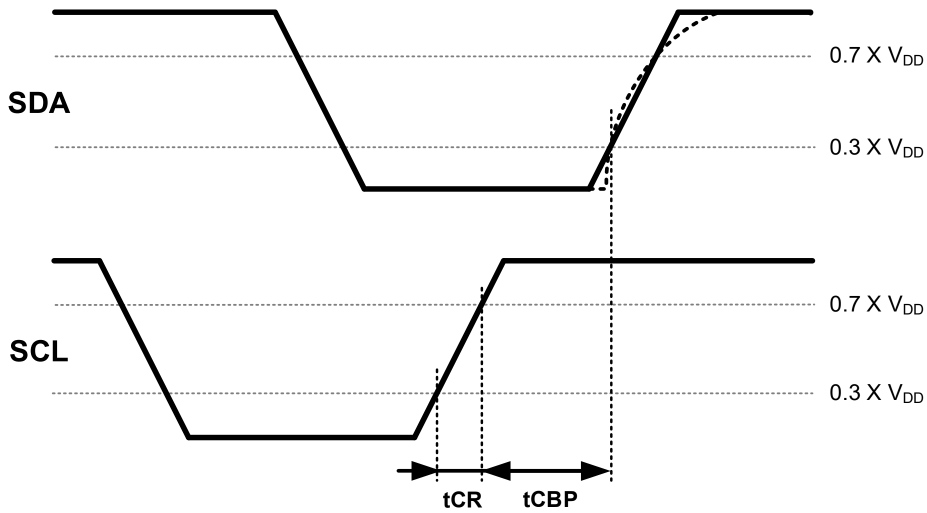


Figure 42-22. I3C SDR Target Out Timing

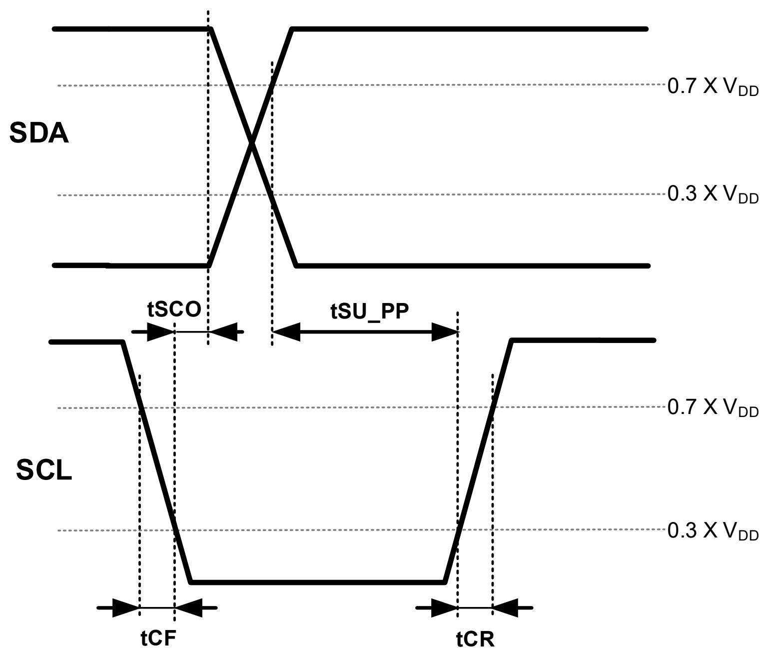


Figure 42-23. Controller SDR Timing

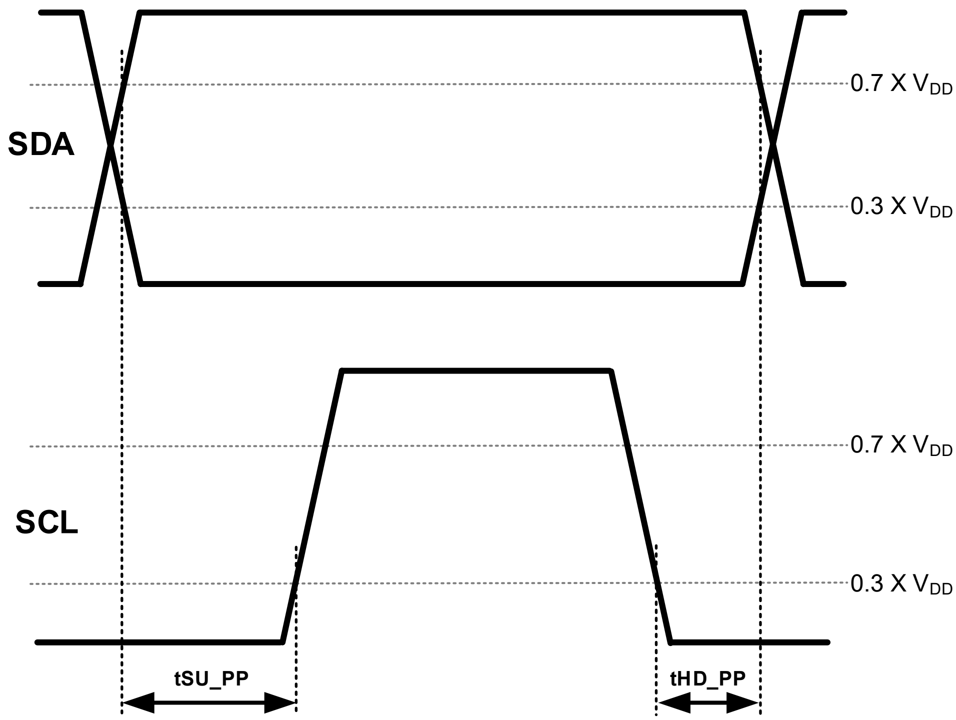


Figure 42-24. Controller DDR Timing

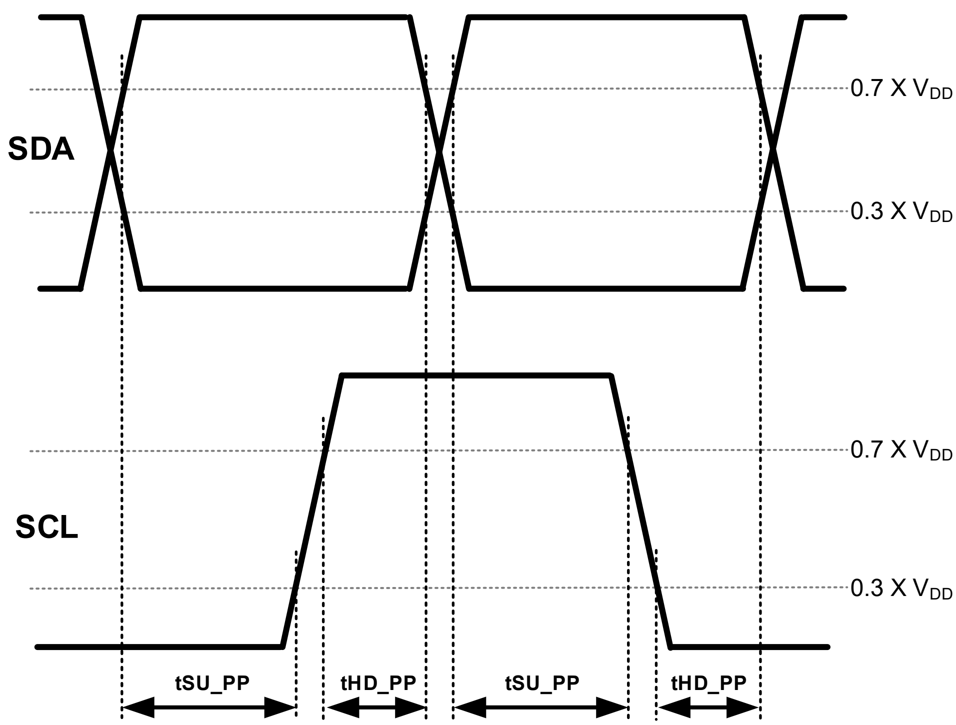


Figure 42-25. T-Bit When Target Ends Read and Controller Generates STOP

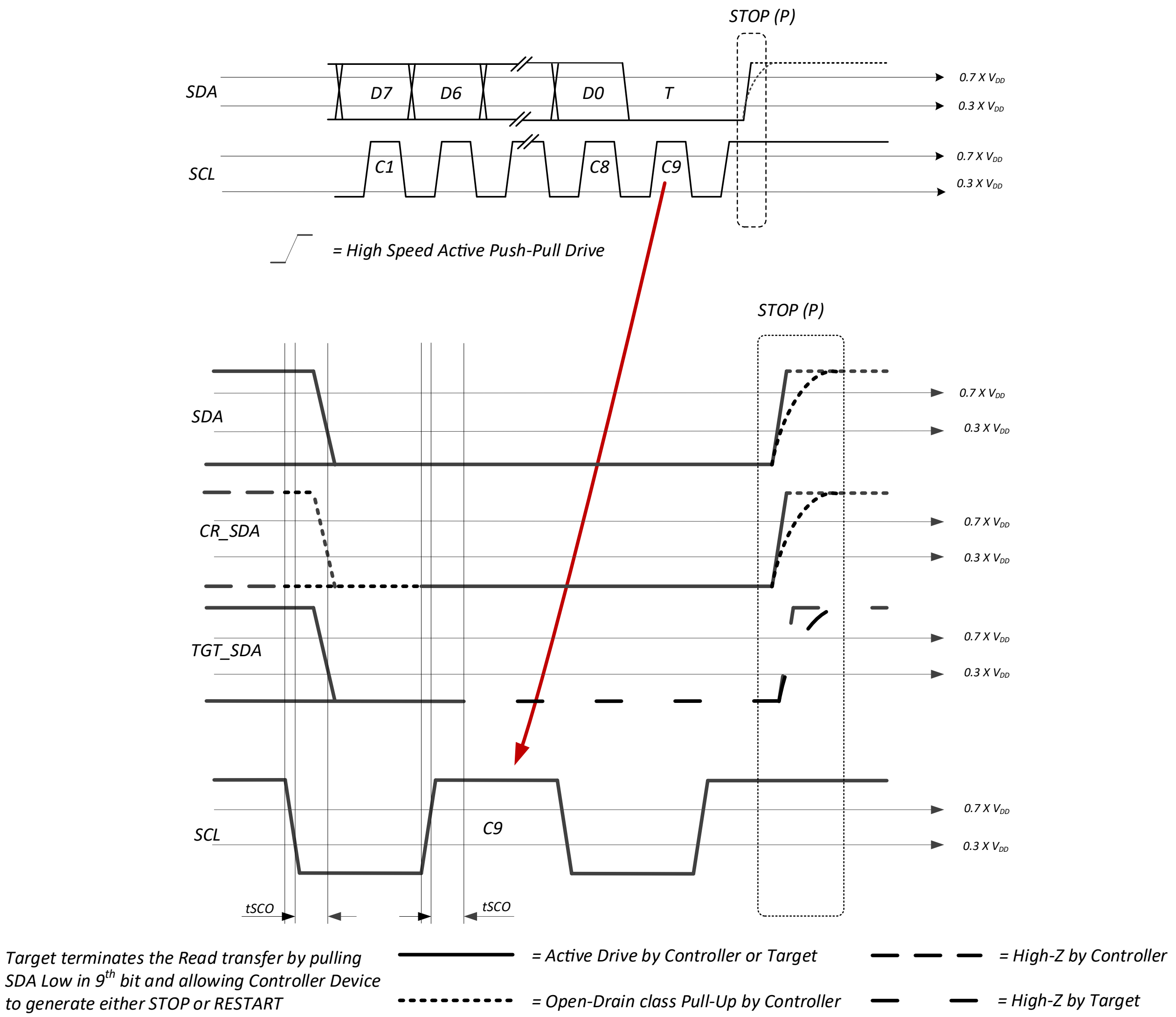


Figure 42-26. T-Bit When Target Ends Read and Controller Generates Repeated START

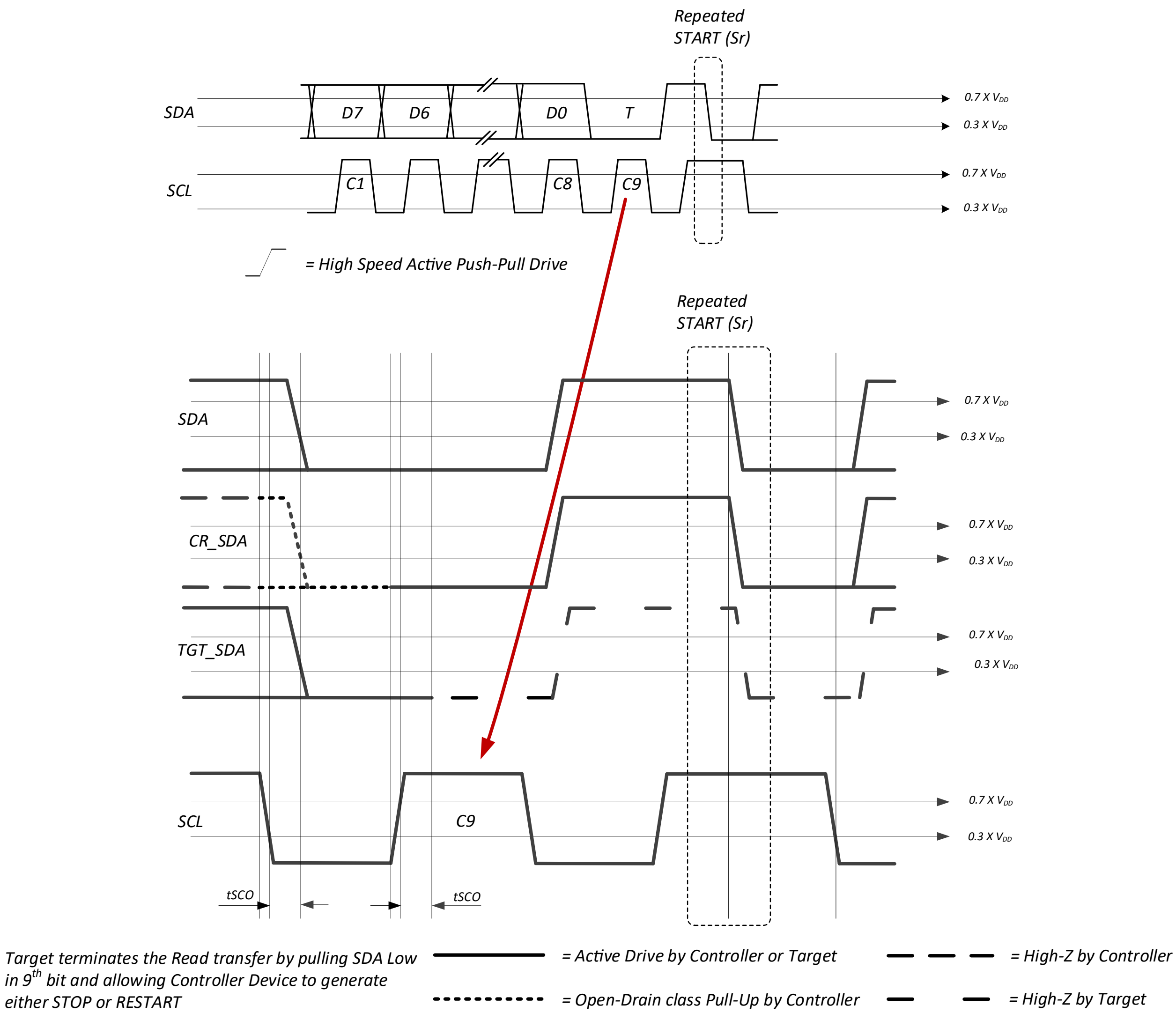


Figure 42-27. T-Bit When Controller Ends Read with Repeated START and STOP

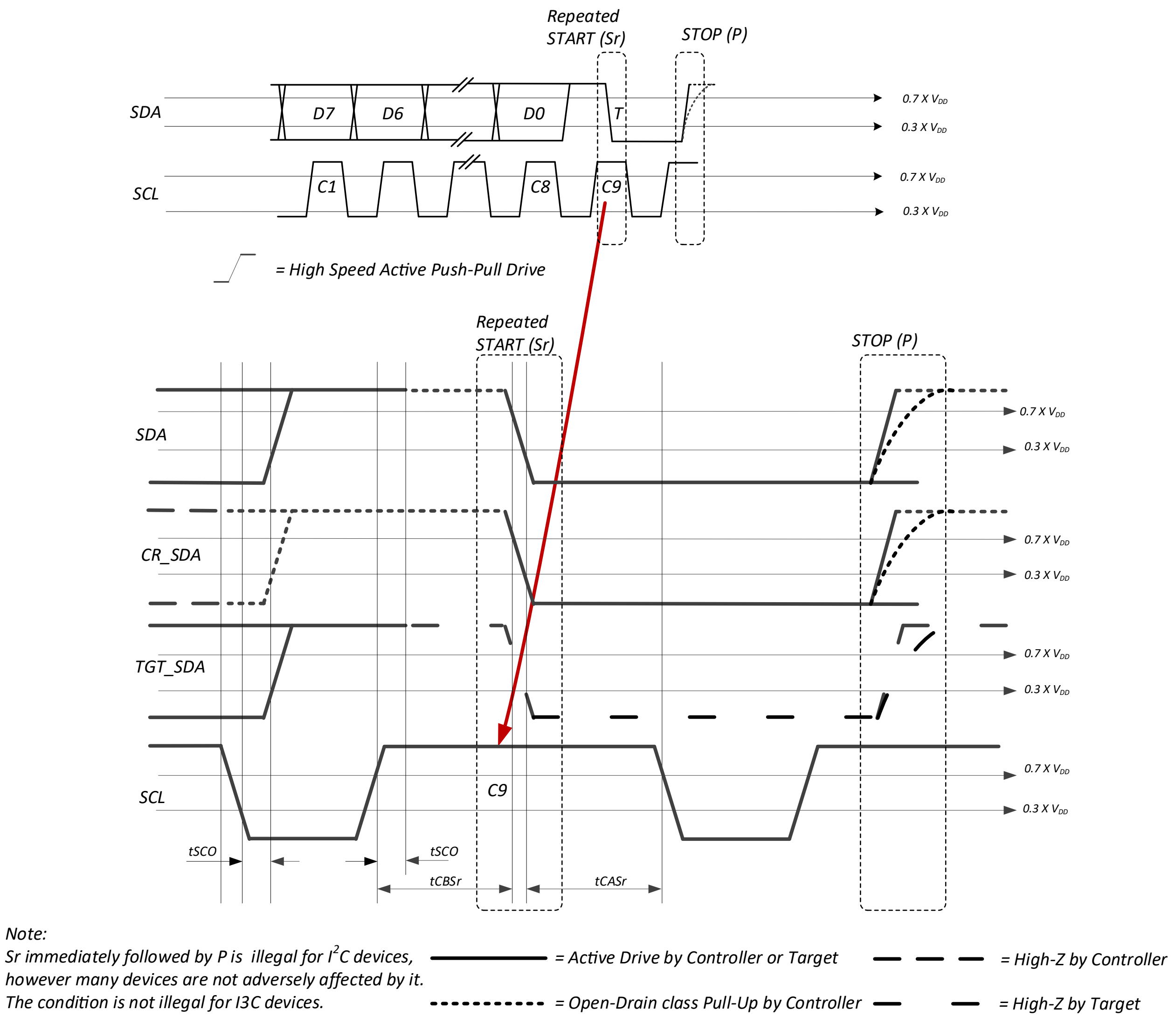


Figure 42-28. Controller to Controller Bus Handoff

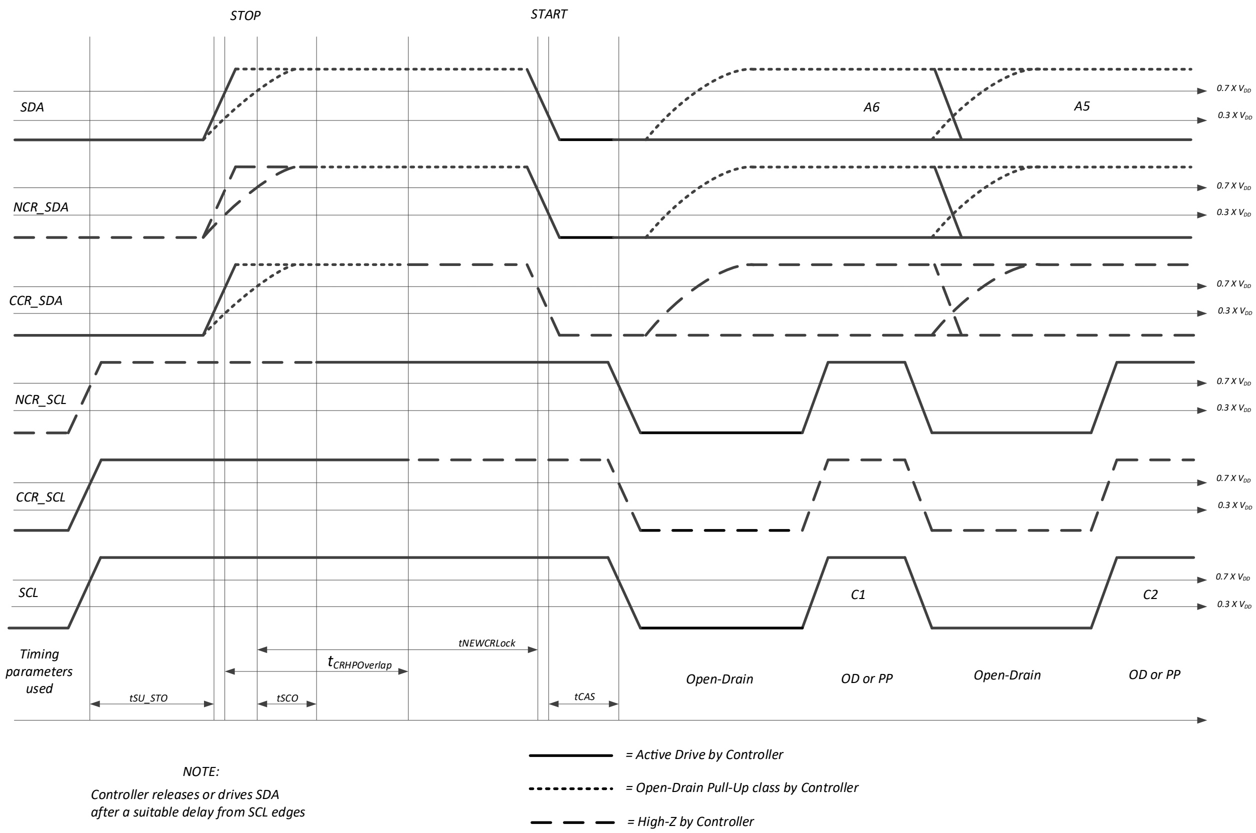


Figure 42-29. I^2C Spike Filter Behavior

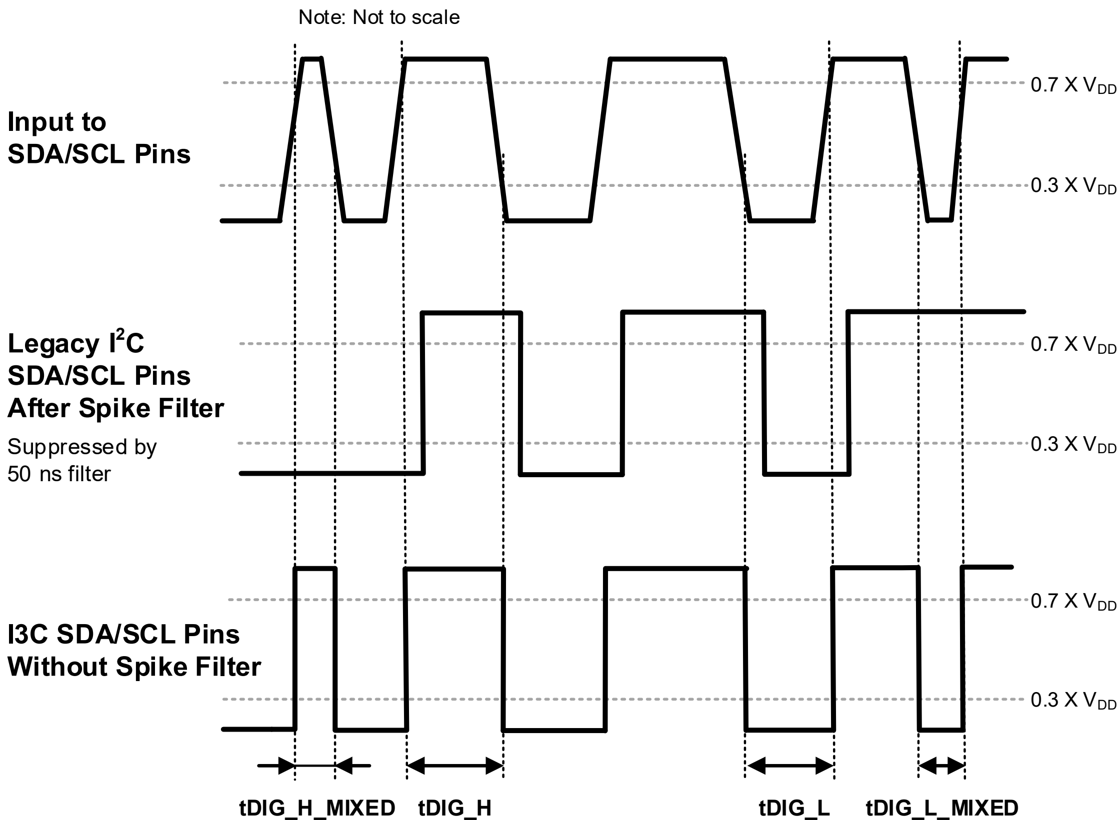


Figure 42-30. UARTx Module I/O Timing Characteristics

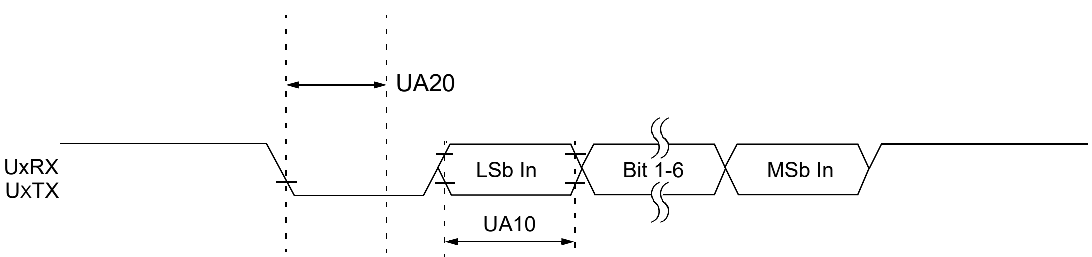


Table 42-39. UARTx Module I/O Timing Requirements(1)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤+85°C for Industrial Operating temperature -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
UA10	T_{BAUD}	UARTx Baud Time	20	—	—	ns	

Note:
1. Parameters are characterized but not tested in manufacturing.

Table 42-39. UARTx Module I/O Timing Requirements[1] (continued)

Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤+85°C for Industrial Operating temperature -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
UA11	F_{BAUD}	UARTx Baud Frequency	—	—	20	Mbps	BRGS = 0,CLKMOD = 0 (16x divide) or CLKMOD = 1 (fractional)
			—	—	50	Mbps	BRGS = 1,CLKMOD = 0 (4x divide)
UA20	T_{SBPM}	Start Bit Pulse Width to Trigger UARTx Wake-up	50	—	—	ns	

Note:
1. Parameters are characterized but not tested in manufacturing.

Table 42-40. ADC Module Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	Symbol	Characteristics	Min.	Typical	Max.	Units	Conditions
Analog Input							
AD12	$V_{INH} – V_{INL}$	Full-Scale Input Span	AV_{SS}	—	AV_{DD}	V	
AD14	V_{IN}	Absolute Input Voltage	AV_{SS} – 0.3	—	AV_{DD} + 0.3	V	
AD60	C_{HOLD}	Hold Capacitor Capacitance[1]	—	1	—	pF	
AD61	C_{PIN}	Pin Capacitance[1]	—	4	—	pF	
AD62	R_{IC}	Input Resistance[1]	—	120	—	Ω	Includes R_{SS}
AD66	V_{BG}	Internal Voltage Reference Source	0.784	0.8	0.816	V	From 3/4 band gap buffer
AD67	V_{REF}	Internal 15/16 AV_{DD} Reference[2]	-10	5	10	LSb	
AD68	T_{REF}	Internal 15/16 AV_{DD}Reference Sampling Time[2]	32	—	—	ns	
AD69	$VL_{\_REF}$	Internal 1/16 AV_{DD}Reference [2]	-10	5	10	LSb	
AD70	$TL_{\_REF}$	Internal 1/16 AV_{DD}Reference Sampling Time[2]	32	—	—	ns	
ADC Accuracy							
AD20	N_R	Resolution	12 data bits			bits	
AD21	INL	Integral Nonlinearity	-3	—	3	LSb	V_{DD} = 3.3V, AV_{DD} = 3.3V Gain error uncompensated
AD22	DNL	Differential Nonlinearity	-1	—	2	LSb	
AD23	GERR	Gain Error	-130	—	25	LSb	
AD24	OERR	Offset Error	-20	-5	5	LSb	

Notes:
1. Parameters are not characterized or tested in manufacturing.
2. Parameters are characterized but not tested in manufacturing.
3. Characterized with a 1 kHz sine wave.
4. Throughput includes 1.5 T_{AD} conversion time.
5. Data in the “Typ” column are at 3.3V, +25°C. Parameters are for design guidance only and are not tested.

Table 42-40. ADC Module Specifications (continued)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Param No.	**Symbol**	**Characteristics**	**Min.**	**Typical**	**Max.**	**Units**	**Conditions**
AD23a	GERR	Gain Error(2)	-10	—	10	LSb	V_{DD} = 3.3V, AV_{DD} = 3.3V Gain error compensated
AD25	—	Monotonicity	—	—	—	—	Guaranteed
Dynamic Performance							
AD34	ENOB	Effective Number of Bits(2,3)	—	10.5	—	bits	
AD50	T_{AD}	ADC Clock Period	12.5	—	125	ns	T_{AD} = FIN/4
AD51	F_{TP}	Throughput Rate(1,4)	—	—	40	Msps	

Notes:
1. Parameters are not characterized or tested in manufacturing.
2. Parameters are characterized but not tested in manufacturing.
3. Characterized with a 1 kHz sine wave.
4. Throughput includes 1.5 T_{AD} conversion time.
5. Data in the “Typ” column are at 3.3V, +25°C. Parameters are for design guidance only and are not tested.

Table 42-41. Die Temperature Diode Specifications(1)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Comments**
TD01	T_{COEFF}	Temperature Coefficient	—	1.5	—	mV/C	
TD02	T_{SAMPLE}	Sampling Time	0.3125	—	—	µS	

Note:
1. Parameters are for design guidance only and are not tested.

Table 42-42. High-Speed Analog Comparator Module Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Comments**
CM10	V_{IOFF}	Input Offset Voltage	-35	—	+35	mV	
CM11	V_{ICM}	Input Common-Mode Voltage Range(1)	AV_{SS}	—	AV_{DD}	V	
CM13	CMRR	Common-Mode Rejection Ratio(2)	60	—	—	dB	
CM14	T_{RESP}	Large Signal Response(2)	—	5	—	ns	V+ input step of 100 mV while V- input is held at $AV_{DD}/2$
CM15	V_{HYST}	Input Hysteresis(2)	15	30	45	mV	Depends on HYSSEL[1:0]

Notes:
1. Parameters are characterized but not tested in manufacturing.
2. Parameters are for design guidance only and are not tested in manufacturing.

Table 42-43. DACx Module Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Comments**
DA02	CV_{RES}	Resolution		12		bits	
DA03	INL	Integral Nonlinearity Error(2)	-25	—	35	LSb	
DA04	DNL	Differential Nonlinearity Error(2)	-5	—	5	LSb	
DA05	EOFF	Offset Error(2)	-10	—	20	LSb	Internal node at comparator input
DA06	EG	Gain Error(2)	-10	—	50	LSb	Internal node at comparator input
DA07	T_{SET}	Settling Time(1)	600	750	2000	ns	Output with 1% of desired output voltage with a 5-95% or 95-5% step
DA08	V_{OUT}	Voltage Output Range(2)	0.165	—	3.135	V	V_{DD} = 3.3V
DA09	T_{TR}	Transition Time(1)	340	—	—	ns	
DA10	T_{SS}	Steady-State Time(1)	550	—	—	ns	

Notes:
1. Parameters are for design guidance only and are not tested in manufacturing.
2. DAC output codes from 5% to 95%. The DAC is operational at values <5% or >95%.

Table 42-44. DACx Output (DACOUTx Pins) Specifications(1)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Comments**
DA11	R_{LOAD}	Resistive Output Load Impedance	10K	—	—	Ohm	
DA11a	C_{LOAD}	Output Load Capacitance	—	—	30	pF	Including output pin capacitance
DA12	I_{OUT}	Output Current Drive Strength	-3	—	3	mA	Sink and source

Note:
1. Parameters are for design guidance only and are not tested in manufacturing.

Table 42-45. Current Bias Generator Specifications[1]

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended						
Parameter No.	Symbol	Characteristic	Min.	Typ.	Max.	Units
CC03	I10SRC	10 μA Source Current	8	—	12	μA
CC04	ISELOUT	0-200 μA Selectable Output Source Current	—	±15	—	%

Note:
1. Parameters are for design guidance only and are not tested in manufacturing.

Table 42-46. Operational Amplifier Specifications

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	Symbol	Characteristic	Min	Typ.	Max.	Units	Comments
OA01	GBWP	Gain Bandwidth Product[1,3]	—	10	—	MHz	Low-Power mode
			—	100	—	MHz	High-Power mode
OA02	SR	Slew Rate[1]	—	10	—	V/μs	Low-Power mode. Measured from 0.5 to 2.5 Volts with a step change in input voltage.
			—	100	—	V/μs	High-Power mode. Measured from 0.5 to 2.5 Volts with a step change in input voltage.
OA03	V_{IOFF}	Input Offset Voltage	-1.5[1]	-1/+1	+1.5[1]	mV	0°C - 85°C Unity gain configuration, High-Power mode
			-5	-1/+1	+5	mV	-40°C - 125°C Unity gain configuration, High-Power mode
OA04	V_{IBC}	Input Bias Current	—	—	—	—	See DI50
OA05	V_{ICM}	Common-Mode Input Voltage Range[1]	AV_{SS}	—	AV_{DD}	V	
OA07	CMRR	Common-Mode Rejection Ratio[1]	—	80	—	dB	
OA08	PSRR	Power Supply Rejection Ratio[1]	—	60	—	dB	At 10 kHz

Table 42-46. Operational Amplifier Specifications (continued)

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended							
Parameter No.	**Symbol**	**Characteristic**	**Min**	**Typ.**	**Max.**	**Units**	**Comments**
OA09	V_{OR}	Output Voltage Range(1)	AV_{SS}	—	AV_{DD}	mV	0.5V input overdrive, no output loading
OA11	C_{LOAD}	Output Load Capacitance(1)	—	—	30	pF	Including output pin capacitance
OA12	I_{OUT}	Output Current Drive Strength(1)	—	10	—	mA	Sink and source
OA13	P_{MARGIN}	Phase Margin(1)	65	—	—	degree	Unity gain
OA14	G_{MARGIN}	Gain Margin(1)	20	—	—	dB	Unity gain
OA15	OLG	Open-Loop Gain(1)	80	90	—	dB	

Notes:
1. Parameters are for design guidance only and are not tested in manufacturing.
2. Parameters are characterized but not tested in manufacturing.
3. Specification for small signal input.

Table 42-47. UREF Module Specification

Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended						
Param No.	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
UR10	UREF Offset	-7	±3	+7	mV	VCM = V_{DD}/2, -40°C - 85°C
UR11	UREF Offset	-9	±5	+9	mV	VCM = V_{DD}/2, -40°C - 125°C

Table 42-48. RDC Module Specification(1)

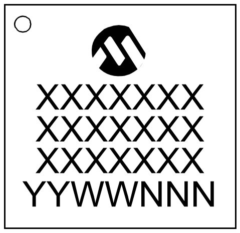
Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +85°C for Industrial -40°C ≤ T_A ≤ +125°C for Extended					
Param No.	**Symbol**	**Description**	**Min**	**Max**	**Units**
RDC01	F_{EXC}	Excitation output clock frequency	2	20	KHz
RDC02	F_{TRIG}	ADC trigger frequency	4	640	KHz
RDC03	F_{RDCCLK}	Excitation input clock frequency	F_{TRIG}*4	—	KHz

Note:
1. Parameters are for design guidance only and are not tested in manufacturing.

43. Packaging Information

43.1. Package Marking Information

36-Lead vQFN (5x5 mm)

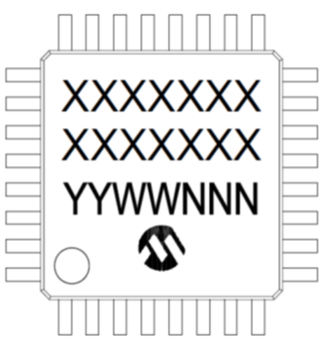


Example

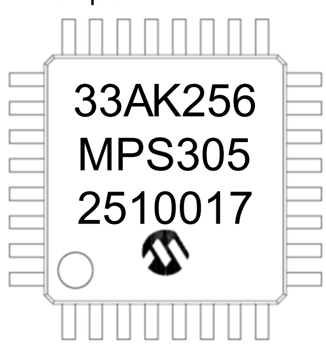
48-Lead VQFN (6x6 mm)

Example

48-Lead TQFP (7x7 mm)



Example

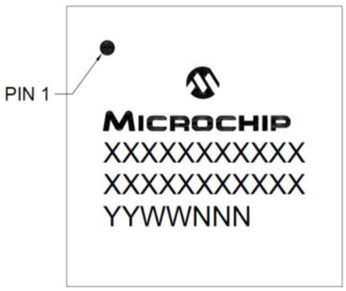


Legend:

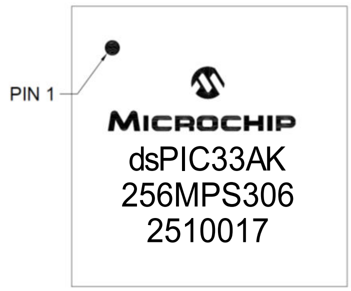
XX...X	Customer-specific information
Y	Year code (last digit of calendar year)
YY	Year code (last 2 digits of calendar year)
WW	Week code (week of January 1 is week ‘01’)
NNN	Alphanumeric traceability code

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

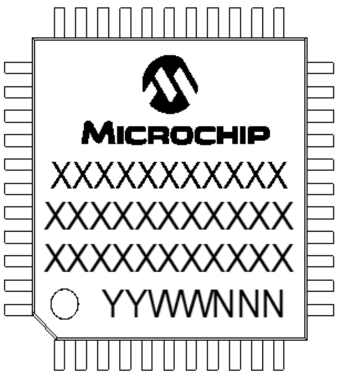
64-Lead vQFN (9x9 mm)



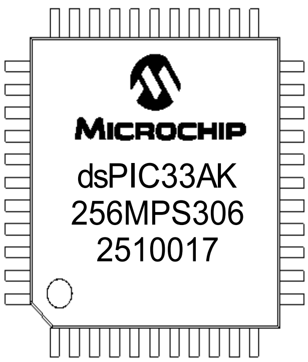
Example



64-Lead TQFP (10x10 mm)



Example



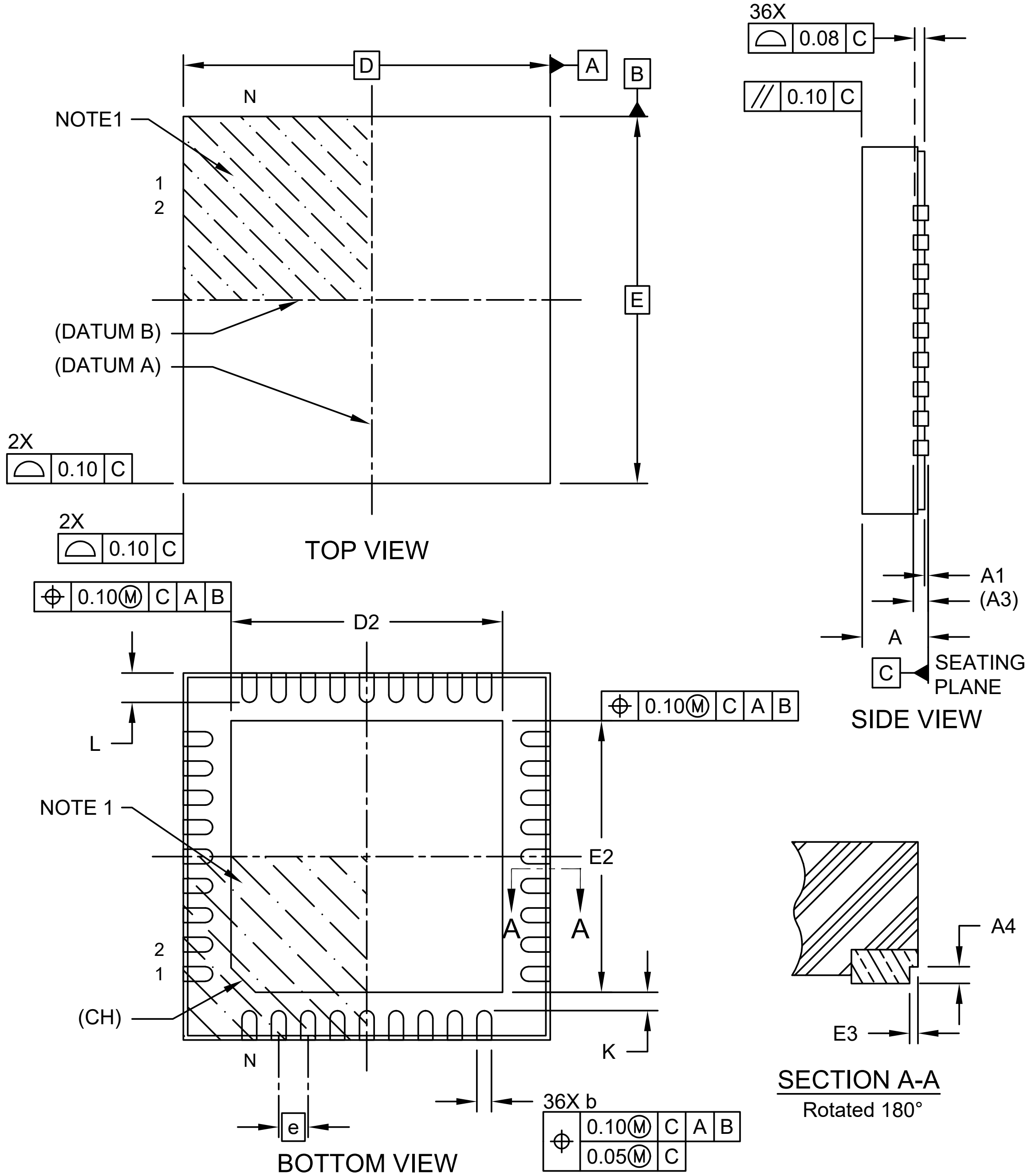
Legend:	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

43.2. Package Details

**36-Lead Very Thin Plastic Quad Flat, No Lead Package (4AW) - 5x5x0.9 mm Body [VQFN]
With 3.7x3.7 mm Exposed Pad and Step Cut Wettable Flanks**

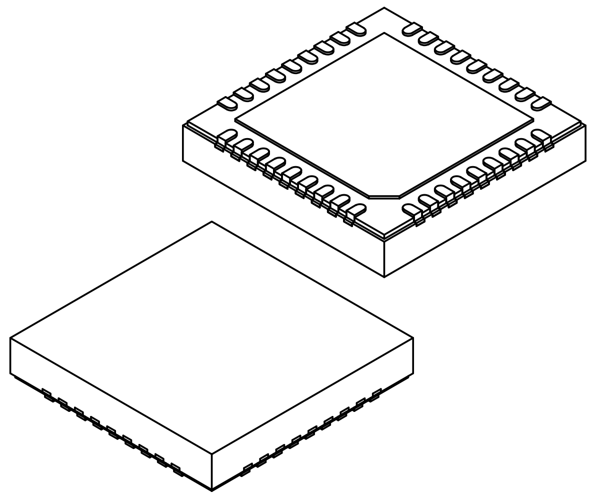
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-579-4AW Rev B Sheet 1 of 2

36-Lead Very Thin Plastic Quad Flat, No Lead Package (4AW) - 5x5x0.9 mm Body [VQFN] With 3.7x3.7 mm Exposed Pad and Step Cut Wettable Flanks

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		36	
Pitch	e		0.40 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		5.00 BSC	
Exposed Pad Length	D2	3.60	3.70	3.80
Overall Width	E		5.00 BSC	
Exposed Pad Width	E2	3.60	3.70	3.80
Exposed Pad Chamfer	CH		0.35 REF	
Terminal Width	b	0.15	0.20	0.25
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K	0.25	-	-
Wettable Flank Step Cut Width	E3	0.035	-	0.085
Wettable Flank Step Cut Depth	A4	0.100	-	0.190

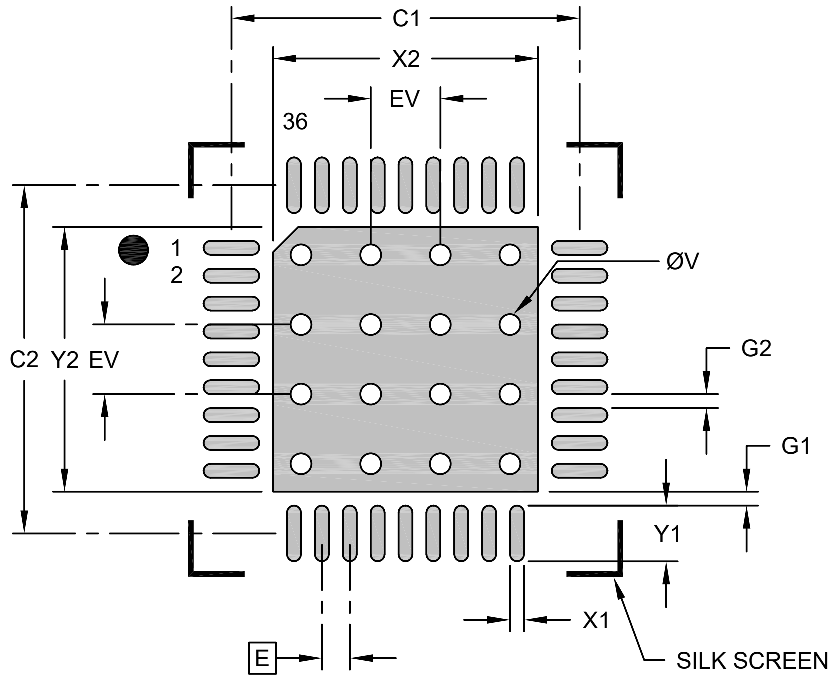
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-579-4AW Rev B Sheet 2 of 2

36-Lead Very Thin Plastic Quad Flat, No Lead Package (4AW) - 5x5x0.9 mm Body [VQFN] With 3.7x3.7 mm Exposed Pad and Step Cut Wettable Flanks

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

		Units	MILLIMETERS		
Dimension Limits			MIN	NOM	MAX
Contact Pitch	E			0.40 BSC	
Center Pad Width	X2				3.80
Center Pad Length	Y2				3.80
Contact Pad Spacing	C1			5.00	
Contact Pad Spacing	C2			5.00	
Contact Pad Width (X36)	X1				0.20
Contact Pad Length (X36)	Y1				0.80
Contact Pad to Center Pad (X36)	G1		0.20		
Contact Pad to Center Pad (X32)	G2		0.20		
Thermal Via Diameter	V			0.30	
Thermal Via Pitch	EV			1.00	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

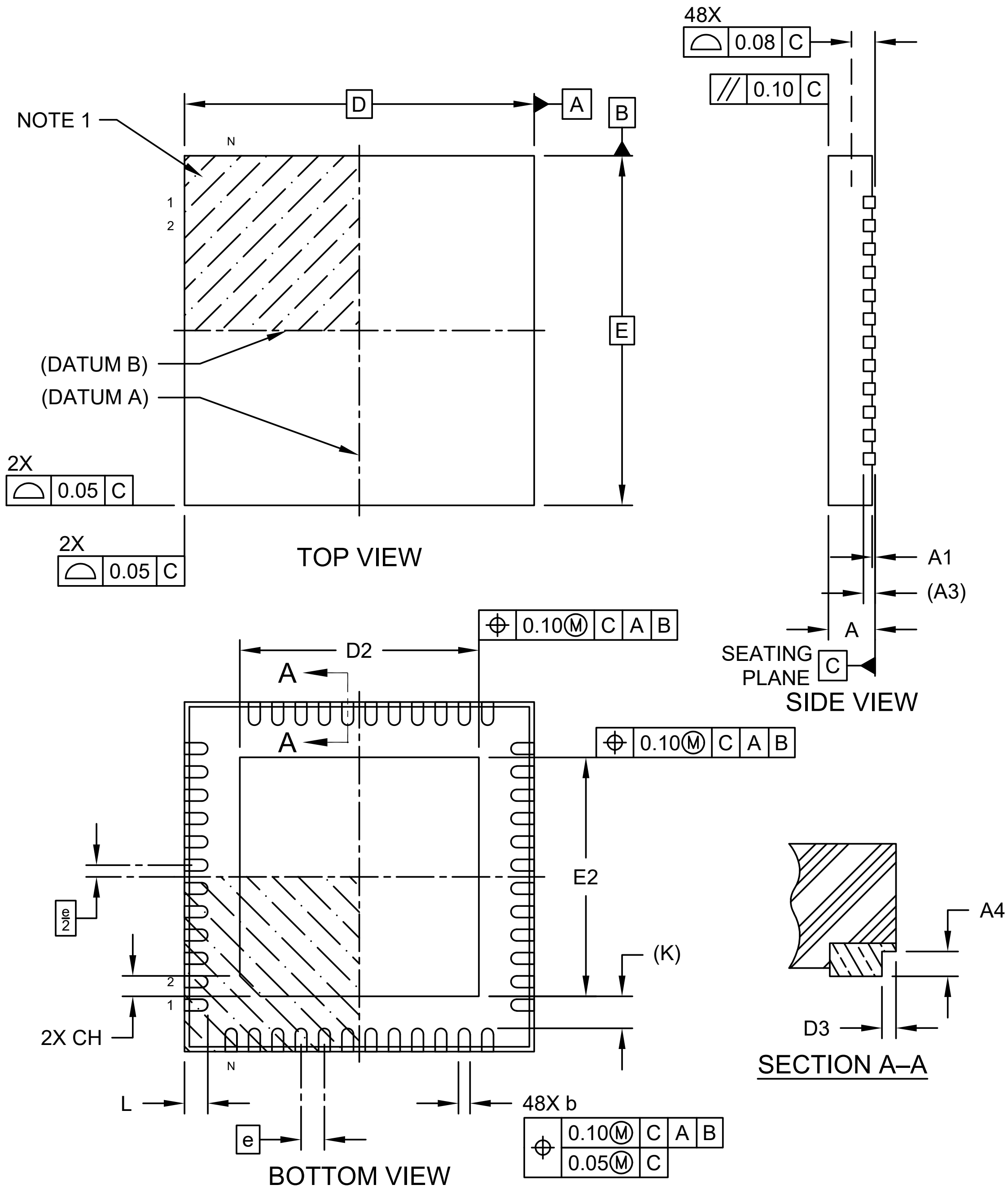
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2579-4AW Rev B

48-Lead Very Thin Plastic Quad Flat, No Lead Package (6MX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad and Stepped Wettable Flanks

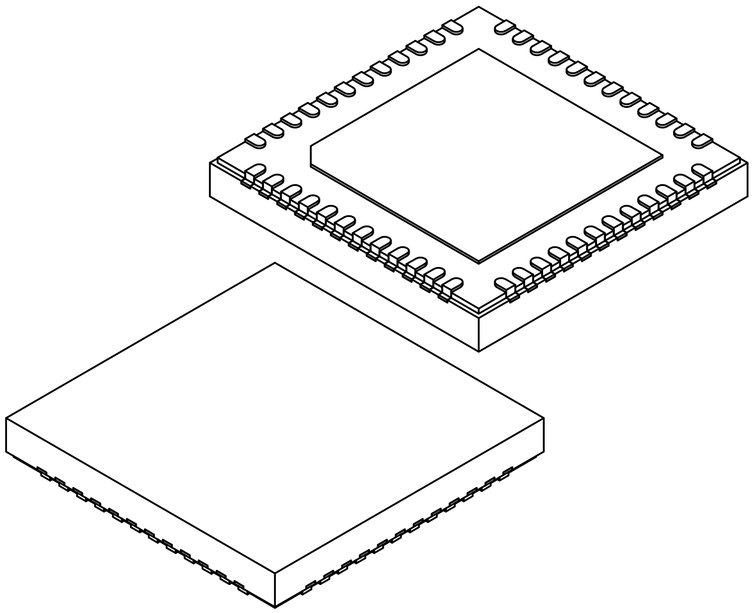
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-504-6MX Rev B Sheet 1 of 2

48-Lead Very Thin Plastic Quad Flat, No Lead Package (6MX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



	Units	MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		48	
Pitch	e		0.40 BSC	
Overall Height	A	0.80	0.85	0.90
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.20 REF	
Overall Length	D		6.00 BSC	
Exposed Pad Length	D2	4.00	4.10	4.20
Overall Width	E		6.00 BSC	
Exposed Pad Width	E2	4.00	4.10	4.20
Exposed Pad Corner Chamfer	CH		0.35 REF	
Terminal Width	b	0.15	0.20	0.25
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K		0.55 REF	
Wettable Flank Step Length	D3	-	-	0.085
Wettable Flank Step Height	A4	0.10	-	0.19

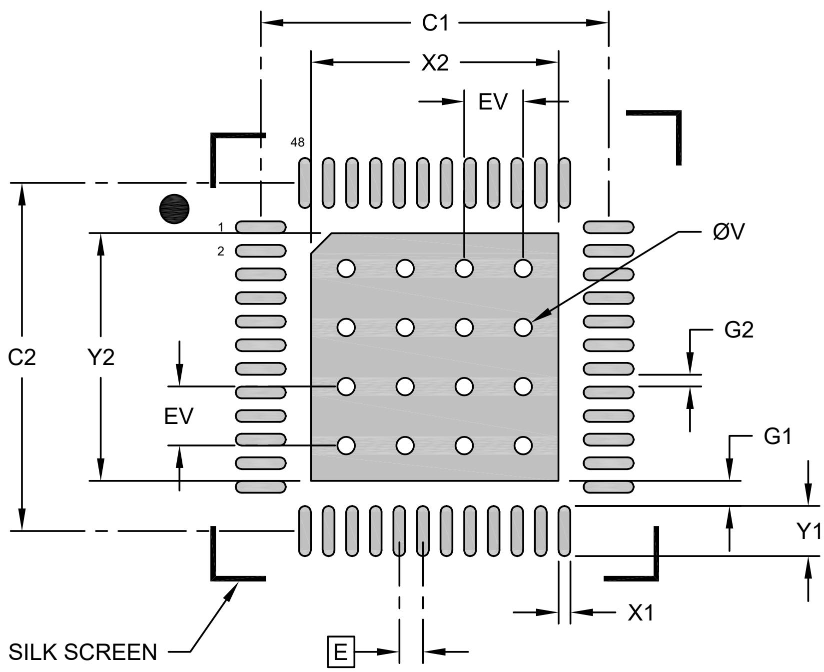
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-504-6MX Rev B Sheet 2 of 2

48-Lead Very Thin Plastic Quad Flat, No Lead Package (6MX) - 6x6 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

	Units	MILLIMETERS		
	Dimension Limits	MIN	NOM	MAX
Contact Pitch	E		0.40 BSC	
Optional Center Pad Width	X2			4.20
Optional Center Pad Length	Y2			4.20
Contact Pad Spacing	C1		5.90	
Contact Pad Spacing	C2		5.90	
Contact Pad Width (X48)	X1			0.20
Contact Pad Length (X48)	Y1			0.85
Contact Pad to Center Pad (X48)	G1	0.20		
Contact Pad to Contact Pad (X44)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

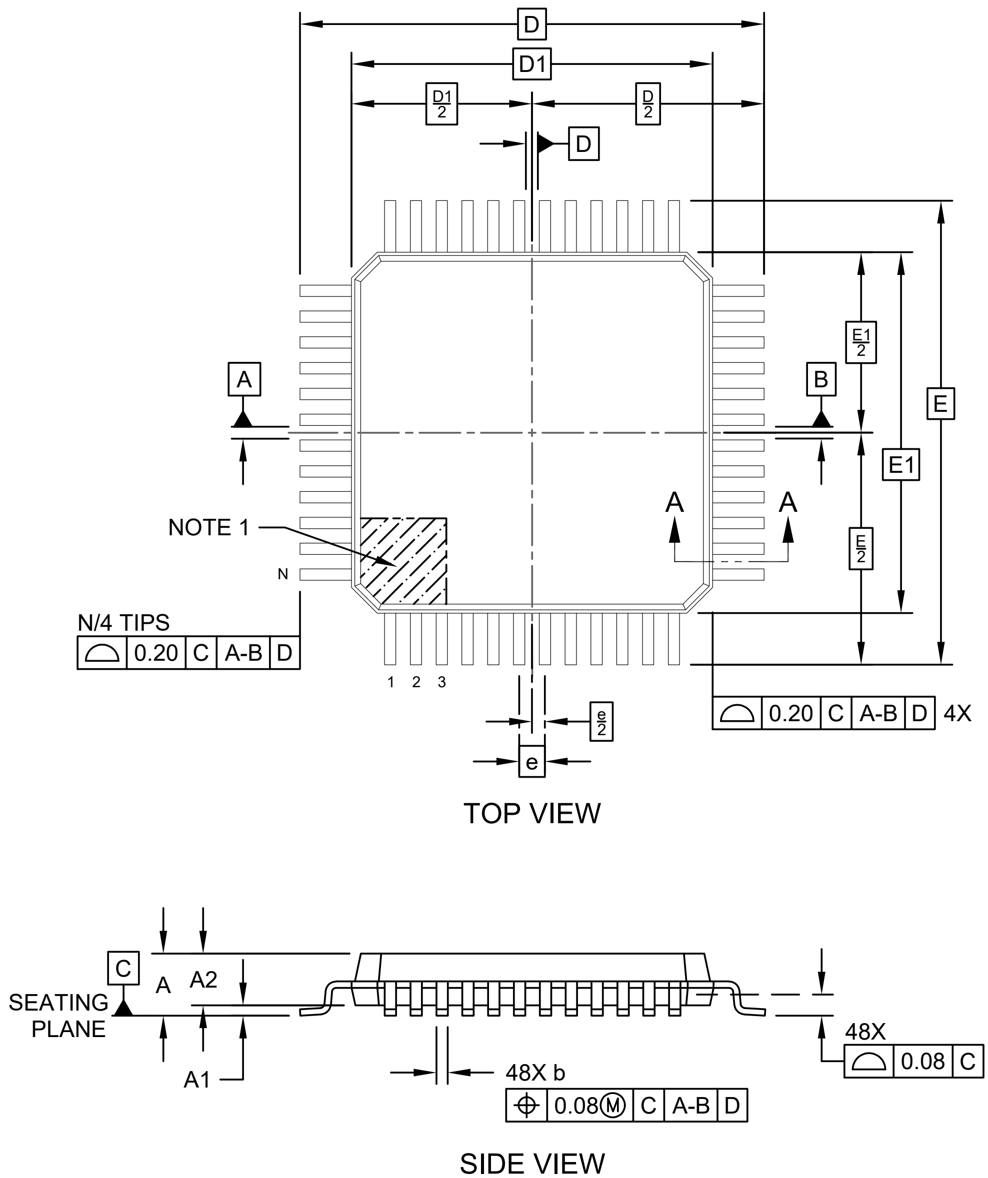
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2504-6MX Rev B

48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

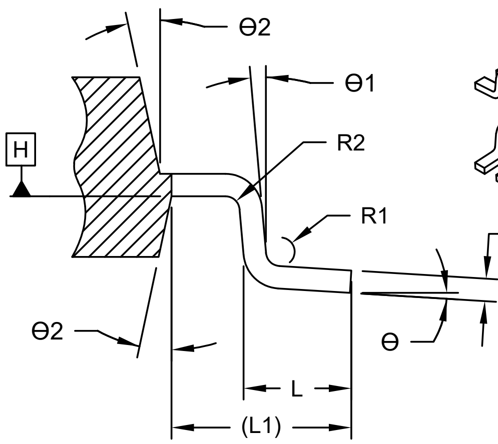
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



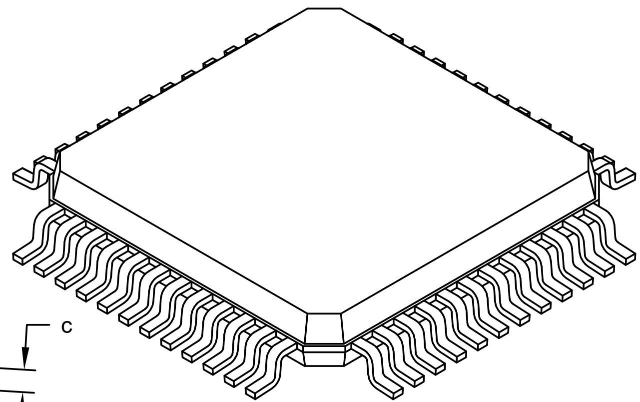
Microchip Technology Drawing C04-00300-Y8X Rev E Sheet 1 of 2

48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



SECTION A-A



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		48	
Pitch	e		0.50 BSC	
Overall Height	A	-	-	1.20
Standoff	A1	0.05	-	0.15
Molded Package Thickness	A2	0.95	1.00	1.05
Overall Length	D		9.00 BSC	
Molded Package Length	D1		7.00 BSC	
Overall Width	E		9.00 BSC	
Molded Package Width	E1		7.00 BSC	
Terminal Width	b	0.17	0.22	0.27
Terminal Thickness	c	0.09	-	0.16
Terminal Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Lead Bend Radius	R1	0.08	-	-
Lead Bend Radius	R2	0.08	-	0.20
Foot Angle	Θ	0°	3.5°	7°
Lead Angle	Θ1	0°	-	-
Mold Draft Angle	Θ2	11°	12°	13°

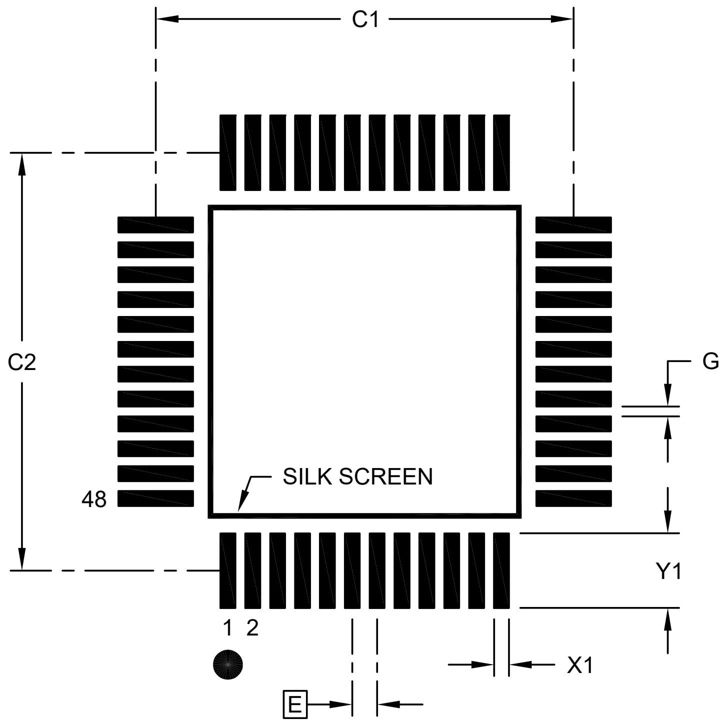
Notes:

1. The Pin 1 visual index feature may vary, but it must be located within the hatched area.
2. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. The theoretically exact value is shown without tolerances.
   REF: Reference Dimension, usually without tolerance, is for information purposes only.

Microchip Technology Drawing C04-00300-Y8X Rev E Sheet 2 of 2

48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Contact Pad Spacing	C1		8.40	
Contact Pad Spacing	C2		8.40	
Contact Pad Width (X48)	X1			0.30
Contact Pad Length (X48)	Y1			1.50
Distance Between Pads	G	0.20		

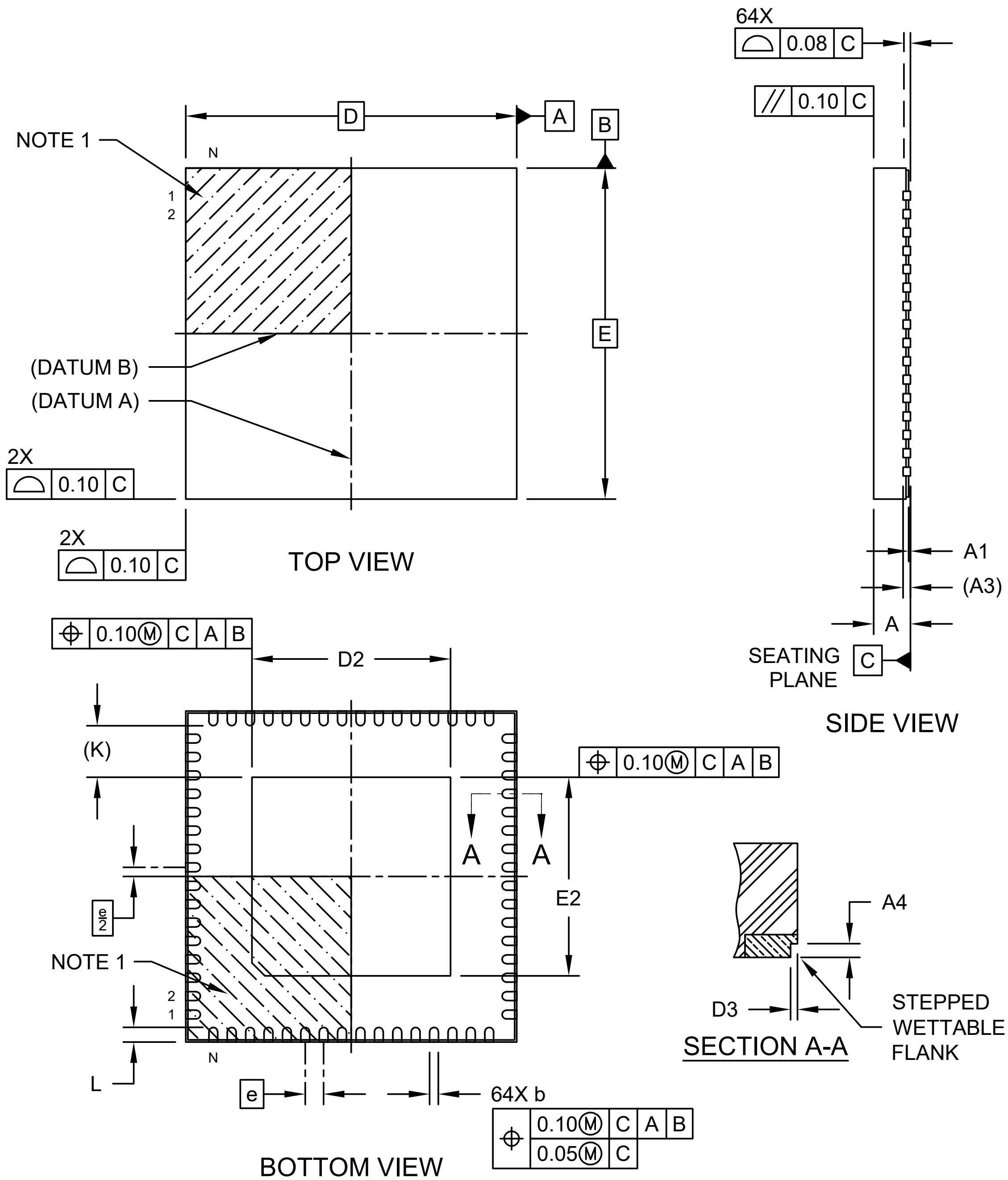
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. The theoretically exact value is shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during the reflow process.

Microchip Technology Drawing C04-02300-Y8X Rev E

64-Lead Very Thin Plastic Quad Flat, No Lead Package (5LX) - 9x9x1.0 mm Body [VQFN] With 5.4 mm Exposed Pad and Stepped Wettable Flanks

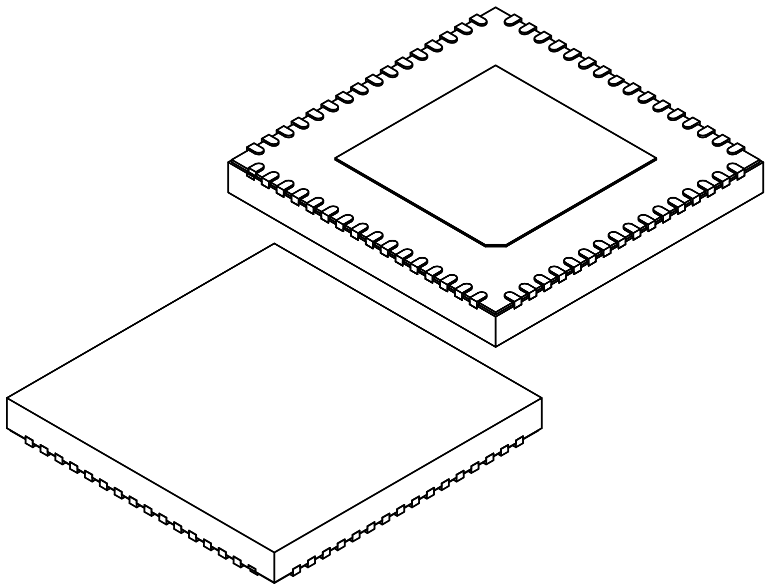
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-483-5LX Rev F Sheet 1 of 2

64-Lead Very Thin Plastic Quad Flat, No Lead Package (5LX) - 9x9x1.0 mm Body [VQFN] With 5.4 mm Exposed Pad and Stepped Wettable Flanks

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



	Units	MILLIMETERS		
	Dimension Limits	MIN	NOM	MAX
Number of Terminals	N		64	
Pitch	e		0.50 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.203 REF	
Overall Length	D		9.00 BSC	
Exposed Pad Length	D2	5.30	5.40	5.50
Overall Width	E		9.00 BSC	
Exposed Pad Width	E2	5.30	5.40	5.50
Terminal Width	b	0.20	0.25	0.30
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K		1.40 REF	
Wettable Flank Step Length	D3	0.035	0.060	0.085
Wettable Flank Step Height	A4	0.10	-	0.19

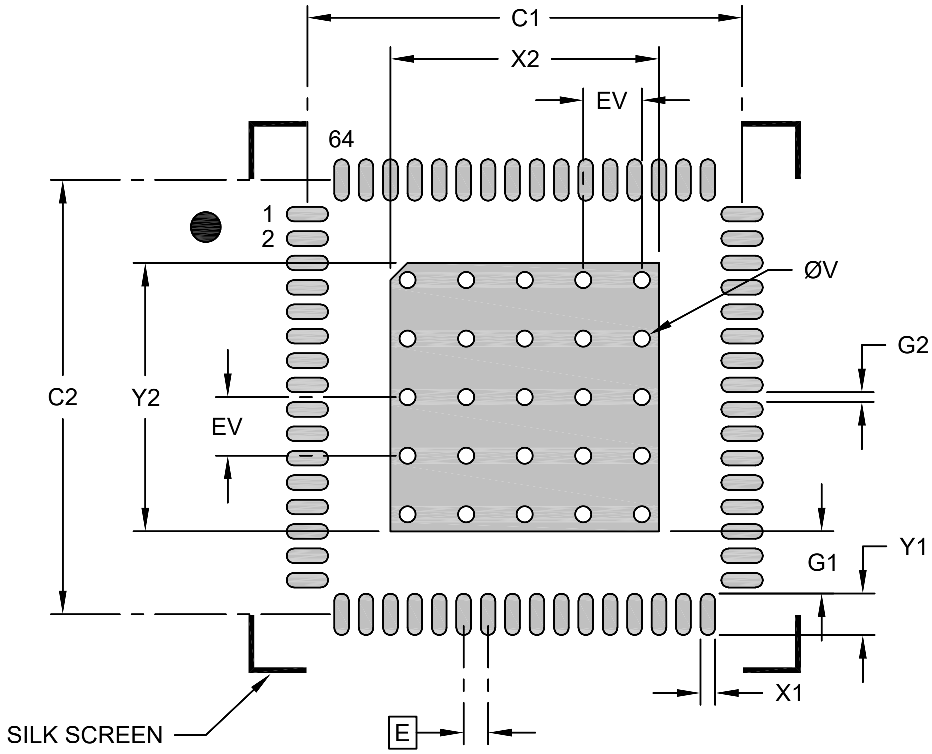
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-483-5LX Rev F Sheet 2 of 2

64-Lead Very Thin Plastic Quad Flat, No Lead Package (5LX) - 9x9x1.0 mm Body [VQFN] With 5.4 mm Exposed Pad and Stepped Wettable Flanks

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	X2			5.50
Optional Center Pad Length	Y2			5.50
Contact Pad Spacing	C1		8.90	
Contact Pad Spacing	C2		8.90	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			0.85
Contact Pad to Center Pad (X64)	G1	1.28		
Contact Pad to Contact Pad (X60)	G2	0.20		
Thermal Via Diameter	V		0.33	
Thermal Via Pitch	EV		1.20	

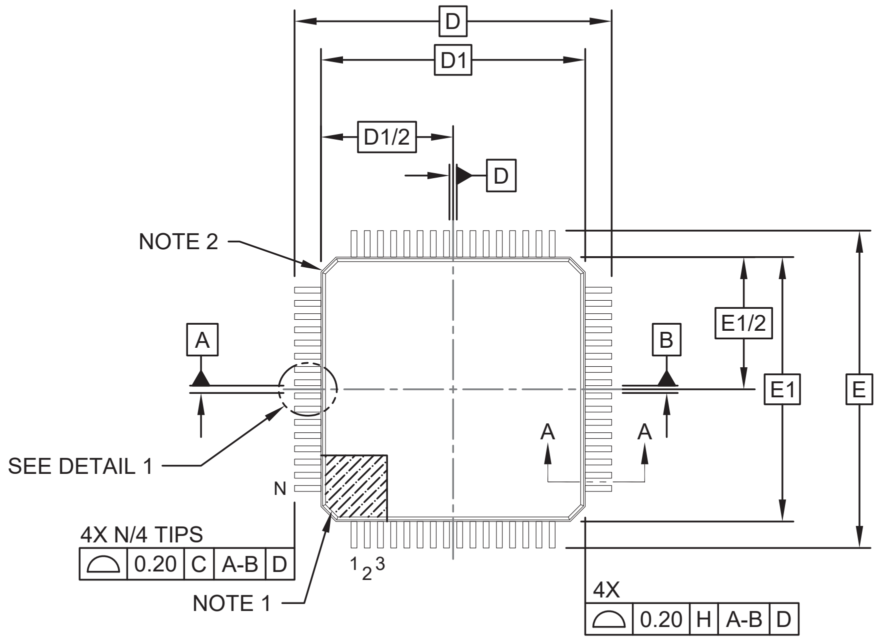
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

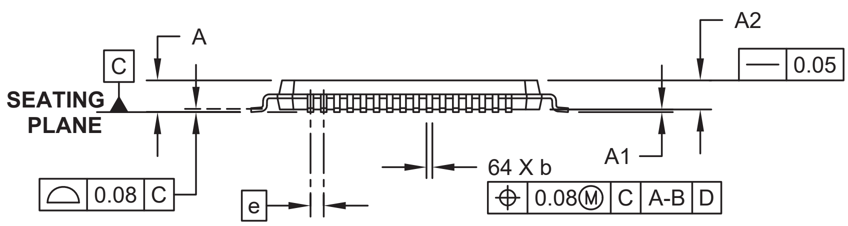
Microchip Technology Drawing C04-2483-5LX Rev F

64-Lead Plastic Thin Quad Flatpack (V2X)-10x10x1 mm Body [TQFP]; With 2.00 mm Footprint

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW

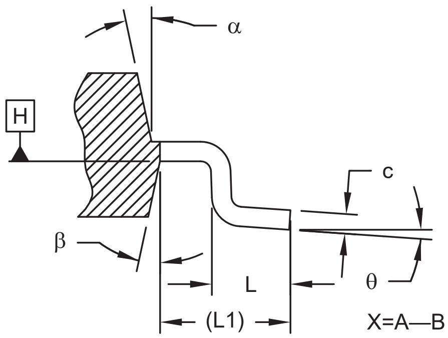


SIDE VIEW

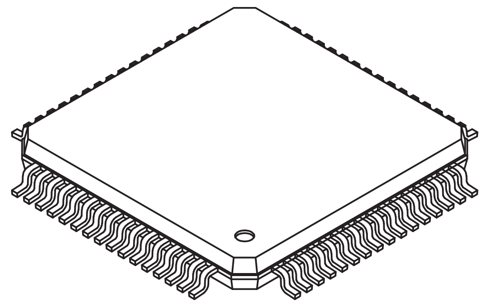
Microchip Technology Drawing C04-00085-V2X Rev F Sheet 1 of 2

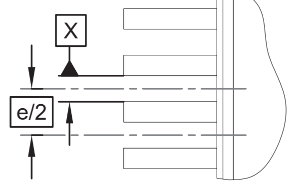
64-Lead Plastic Thin Quad Flatpack (V2X)-10x10x1 mm Body [TQFP]; With 2.00 mm Footprint

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



SECTION A-A





DETAIL 1

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Leads	N		64	
Lead Pitch	e		0.50 BSC	
Overall Height	A	-	-	1.20
Molded Package Thickness	A2	0.95	1.00	1.05
Standoff	A1	0.05	-	0.15
Foot Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Foot Angle	θ	0°	3.5°	7°
Overall Width	E		12.00 BSC	
Overall Length	D		12.00 BSC	
Molded Package Width	E1		10.00 BSC	
Molded Package Length	D1		10.00 BSC	
Lead Thickness	c	0.09	-	0.20
Lead Width	b	0.17	0.22	0.27
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

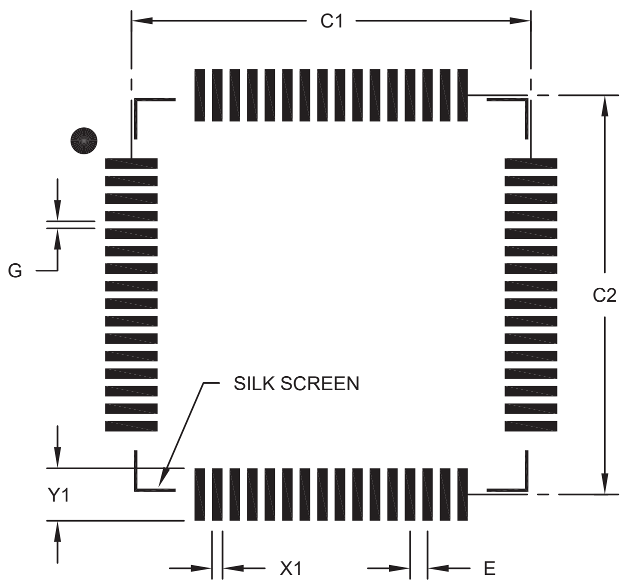
Notes:

1. The Pin 1 visual index feature may vary, but it must be located within the hatched area.
2. Chamfers at the corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. The theoretically exact value is shown without tolerances.
REF: Reference Dimension, usually without tolerance, is for information purposes only.

Microchip Technology Drawing C04-00085-V2X Rev F Sheet 2 of 2

64-Lead Plastic Thin Quad Flatpack (V2X)-10x10x1 mm Body [TQFP]; With 2.00 mm Footprint



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Contact Pad Spacing	C1		11.40	
Contact Pad Spacing	C2		11.40	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			1.50
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. The theoretically exact value is shown without tolerances.

Microchip Technology Drawing C04-02085-V2X Rev F

44. Revision History

Revision A (December 2025)

This is the initial version of the document.

Revision B (March 2026)

This revision incorporates the following updates:

- Sections:
  - Updated **High-Speed Analog-to-Digital Converters**, **Peripheral Features**, **Analog Features**, **Qualification**, **Integrated Touch Controller (ITC)**, **3.3.12.2.1. DSP Multiply Instructions**, **3.4.3.7. TAG Memory Parity**, **3.4.3.7.1. TAG Operation**, **3.4.3.8.1. Cache Mode**, **3.4.4.4. Module Operation When Cache Enabled**, **3.4.4.4.1. ISB Buffers**, **4.1.1.1. Unique Device Identifier (UDID)**, **4.5.1. Execute from RAM**, **8.6. Cryptographic Accelerator Module (CAM)**, **8.6.1.4. Asymmetric Crypto Engine**, **10.6.5. INTTREG**, **10.8. Interrupt Sequence**, **11.4.7. Virtual Output Pins**, **11.4.9. I/O Multiplexing with Multiple Peripherals**, **11.4.10. Change Notice (CN)**, **11.4.10.1. CN Configuration and Operation**, **11.4.11. I/O Integrity Module (IOIM)**, **12.4.3.2. Primary Oscillator Pin Functionality**, **12.4.4. Internal Fast RC (FRC) Oscillator**, **12.4.5. BFRC Oscillator**, **12.4.6. Phase-Locked Loop (PLL)**, **12.4.6.3.1. Setup for Using PLL with the Primary Oscillator (POSC)**, **15. High-Resolution PWM with Fine-Edge Placement**, **15.5.2.2.4. LLC Resonant Converter Mode**, **15.5.2.3.1. Complementary Output Mode**, **16. 40 MSPS Analog-to-Digital Converter (ADC)**, **16.6.3. Windowed Multiple Conversions**, **16.5.4. Integration of the Multiple Samples**, **18.4.1. Excitation Signal Generation**, **18.4.2.1. ADC Input Selection and Coherent Demodulation**, **18.4.2.1.1. ADC Trigger Signals**, **18.4.2.2. Excitation Signal Feedback Delay**, **18.4.2.3. CIC Filter**, **18.4.2.3.2. Heterodyne (Synchronous Demodulation)**, **Operating Modes**, **Register Source Mode**, **External Signal Source Mode**, **18.4.2.3.4. Auto Shift of Filter Output**, **18.4.3. CORDIC Block**, **18.4.3. CORDIC Block**, **18.5. Interrupts**, **20.2. Architectural Overview**, **22. Serial Peripheral Interface (SPI)**, **24.4.5.1.2. Handling GETMXDS CCC**, **24.4.5.1.4. Handling ENTDAA, GETPID and GETDCR** and **45. Product Identification System**.
  - Added **16-Bit Resolution Mode**, **16-bit Conversion Example**, **24. Improved Inter-Integrated Circuit (I3C)**, **10.6.5 Vector Fail Address**, and **15.5.2.6.3 PCI Output Control Priority**.
  - Removed **3.4.3.6.4. Implications of Variable NVM Wait States**, **3.4.3.9.2. Stream Buffers**, **3.4.4.13.2. Cache Coherency After a BOOTSWP Event** and **17. Integrated Touch Controller (ITC)**.
  - Split **40 MSPS Analog-to-Digital Converter (ADC)** and **Integrated Touch Controller (ITC)** into two separate sections.
- Registers:
  - Updated **3.2.5 Core Mode Control Register**, **3.2.6. Modulo Addressing Control Register**, **3.2.8 X AGU Modulo Addressing End Register**, **3.2.10 Y AGU Modulo Addressing End Register**,**3.2.15 Debug Hold PC Register** , **3.4.2.5. Cache RAM Command Register (Address/Control)**, **6.2.22. NVM CRC Seed Register**, **7.1.11. FPED Configuration Register**, **8.2.10. Peripheral Access Control Register 3**, **8.6.2.1. Crypto Accelerator Enable Register**, **9.1.1. Reset Control Register**, **9.4.6.1. Voltage Monitor Control Register**, **9.4.6.3 Voltage Monitor Fault Injection Configuration Register**, **10.4.6. Interrupt Control and Status Register**, **10.4.13. Interrupt Request Flags Register 4**, **10.4.24. Interrupt Enable Register 4**, **10.4.48. Interrupt Priority Register 18**, **10.4.49. Interrupt Priority Register 19**, **12.3.2. Oscillator Configuration Register**, **12.3.5. Clock Generator Control Register**, **12.3.6. Clock Generator Divider Register**, **12.3.7. PLL Control Register**, **12.3.8. PLL Divider Register**, **15.4.1. PWM Clock Control Register**, **15.4.3. Frequency Scaling Minimum**

Period Register, **15.4.4. Master Phase Register**, **15.4.8. Combinational Trigger Register**, **15.4.9. Combinatorial PWM Logic Control Register**, **15.4.10. PWM Event Output Control Register y**, **15.4.11. PWM Generator x Control Register**, **15.4.13. PWM Generator x I/O Control 1 Register**, **15.4.14. PWM Generator x I/O Control 2 Register**, **15.4.15. PWM Generator x Event 1 Register**, **15.4.16. PWM Generator x Event 2 Register**, **15.4.17. PWM Generator x F1 PCI 1 Register**, **15.4.18. PWM Generator x F1 PCI 2 Register**, **15.4.19. PWM Generator x F2 PCI 1 Register**, **15.4.20. PWM Generator x F2 PCI 2 Register**, **15.4.23. PWM Generator x Period Register**, **15.4.27. PWM Generator x Leading-Edge Blanking Register**, **15.4.28. PWM Generator x Phase Register**, **15.4.29. PWM Generator x Duty Cycle Register**, **15.4.30. PWM Generator x Dead-Time Register**, **15.4.32. PWM Generator x Trigger A Register**, **15.4.33. PWM Generator x Trigger B Register**, **15.4.35. PWM Generator x Trigger D Register**, **15.4.36. PWM Generator x Trigger E Register**, **15.4.37. PWM Generator x Trigger F Register**, **18.3.1. RDC Control Register**, **18.3.2. RDC ADC Selection Register**, **18.3.3. RDC Status Register**, **18.3.4. RDC Excitation Signal Control Register**, **18.3.5. RDC Excitation Signal Delay Register**, **18.3.8. RDC CORDIC Block Angle Input Register**, **18.3.13. CIC Status Register**, **18.3.16. CIC Control 2 Register**, **18.3.11. CIC Control 1 Register**, **18.3.12. CIC Length Register**, **18.3.13. CIC Status Register**, **18.3.15. CIC Filter Input Timeout Counter Register**, **18.3.18. CIC Channel x Input Register**, **18.3.19. CIC Channel x Output Register**, **18.3.20. CIC Channel x Decimated Accumulator Value Register**, **19.3.5. DACx Data Register**, **20.3.1. QEI 1 Control Register**, **20.3.2. QEI 1 I/O Control Register**, **20.3.3. QEI 1 Status Register**, **22.3.1. SPIx Control Register 1**, **28.3.2. CCPx Control Register 2**, **37.2.2. Peripheral Module Disable 1 Register** and **37.2.5. Peripheral Module Disable 4 Register**.

  - Added **10.4.27 Interrupt Enable Register 7**, **13.3.15 DMA Channel x Pattern Register**, **PWM Generator x CL PCI 1 Register**, **PWM Generator x CL PCI 2 Register**, **PWM Generator x FF PCI 1 Register**, **PWM Generator x FF PCI 2 Register**, **15.4.25. PWM Generator x SP PCI 1 Register**, **15.4.26. PWM Generator x S PCI 2 Register**, **16.3.57 ADC 2 Channel x Secondary Accumulator Register**, **Peripheral Pin Select Output Register 32**, **Peripheral Pin Select Output Register 33**,**Peripheral Pin Select Output Register 34** and **Peripheral Pin Select Output Register 35**.
  - Removed **12.3.10. Reset Control Register** because it is already in the Resets section.

- Tables:
  - Updated **Table 1. dsPIC33AK256MPS306 Family Device Features**, **Table 2. 36-Pin VQFN Complete Pin Function Descriptions**, **Table 3. 48-Pin VQFN, TQFP Complete Pin Function Descriptions**, **Table 4. 64-Pin VQFN, TQFP Complete Pin Function Descriptions**, **Table 5. Pinout I/O Descriptions**, **Table 9-2. Code Execution Start Time for Various Device Resets**, **Table 11-13. Output Selection for Remappable Pins (RPn)**, **Table 12-3. Clock Generator Clock Resources**, **Table 12-4. PLL Clock Sources**, **Table 12-5. Clock Monitor Clock Resources**, **Table 12-6. Primary Oscillator Modes**, **Table 12-7. Clock Pin Function Selection**, **Table 12-8. PLL Mode Defaults**, **Table 13-7. RELOADS/RELOADD/RELOADC Bits and Data Transfer Modes**, **Table 14-2. CLKSEL Clock Selection bit**,**Table 15-2. MCLKSEL PWM Master Clock Selection**, **Table 16-2. ADC Input Availability**, **Table 16-4. Output Format**, **Table 18-2. CIC Block Summary**, **Table 21-2. UART Clock (FUART) Source Selection bits**, **Table 22-2. SPI Host Clock Source Selection bit**, **Table 26-2. CLKSEL Selection bit**, **Table 27-2. TCS Timer Clock Source Select bit**, **Table 28-2. CLKSEL Time Base Clock Select bits**, **Table 29-4. DS3 Data Selection MUX 3 Signal Selection bits**, **Table 42-2. Operating MHz vs. Voltage**, **Table 42-4. Thermal Packaging Characteristics**, **Table 42-10. DC Characteristics: PWM Delta Current**, **Table 42-12. DC Characteristics: PLL Delta Current**, **Table 42-13. DC Characteristics: ADC Δ Current**, **Table 42-14. DC Characteristics: Comparator + DAC Delta Current**, **Table 42-15. Op Amp Delta Current**, **Table 42-16. I/O Pin Input Specifications**, **Table 42-17. I/O Pin Input Leakage Specifications**, **Table 42-20. Capacitive Loading on Output Pins**, **Table 42-21. External Clock Timing Requirements**, **Table 42-22. PLLn Timing Specifications**, **Table 42-23. Peripheral Input Clock Timing Specifications**, **Table 42-24. Internal FRC Accuracy**,

Table 42-25. I/O Timing Requirements, **Table 42-38. I3C Push-Pull Timing Parameters for SDR and HDR-DDR**, **Table 42-39. UARTx Module I/O Timing Requirements**, **Table 42-40. ADC Module Specifications**, **Table 42-42. High-Speed Analog Comparator Module Specifications** and **Table 42-46. Operational Amplifier Specifications**.

  – Added **Table 42-47. UREF Module Specification** and **Table 42-48. RDC Module Specification**.

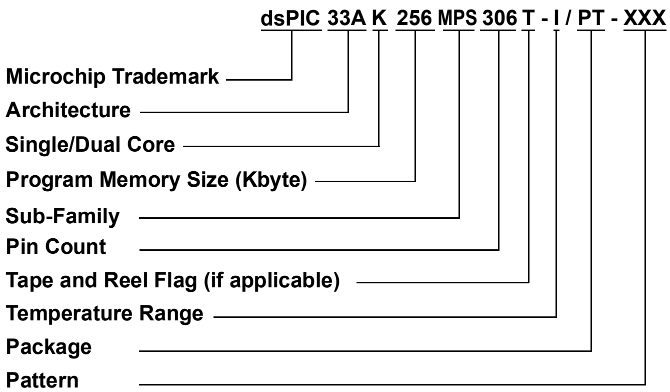
- Figures:
  – Updated **Figure 1-1. dsPIC33AK256MPS306 Family Block Diagram**, **Figure 6-3. Boot Sequence Number**, **Figure 9-3. Window Comp**, **Figure 10-4. Interrupt Latency**, **Figure 12-1. Oscillator Module Block Diagram**, **Figure 12-4. Clock Generator**, **Figure 12-9. PLL Block Diagram**, **Figure 15-15. Override and SWAP Signal Flow, Complementary Mode**, **Figure 12-2. Clock Generator**, **Figure 18-9. CIC Channel Block Diagram** and **Figure 24-1. I3C Hardware Block Diagram**.
- Examples:
  – Updated **Example 11-1. Configuring UART1 Input and Output Functions**, **Example 11-2. Virtual PPS Connection**, **Example 11-3. IOIM Code**, **Example 11-4**, **Example 12-2. Code Example for Using PLL with the Primary Oscillator (POSC)**, **Example 13-5. Code for Fixed to Block Continuous Transfer (Peripheral to Memory)**, **Example 13-6. Null Write Mode**, **Example 18-1. RDC Configuration Code**, **Example 25-5. Short PWM Code (SPC) Support** and **Example 25-6. SENT Reception (SPC Pulse Transmission)**.

  – Removed **Example 6-4. Partition Swap** due to redundancy.
- – Equations:
- Updated **12-8. F_{VCO} Calculation** and **Equation 12-9. F_{PLLO} Calculation**.

Minor grammatical corrections and formatting changes have been made throughout the document.

45. Product Identification System

To order or obtain information, for example, on pricing or delivery, contact Microchip: https://www.microchip.com/en-us/about/contact-us.



Device:	dsPIC33AK128MPS303, dsPIC33AK128MPS305, dsPIC33AK128MPS306, dsPIC33AK256MPS303, dsPIC33AK256MPS305, dsPIC33AK256MPS306, dsPIC33AK128MPS103, dsPIC33AK128MPS105, dsPIC33AK128MPS106, dsPIC33AK256MPS103, dsPIC33AK256MPS105, dsPIC33AK256MPS106	
Architecture:	33A	= 32-bit Digital Signal Controller (DSC)
Single/Dual Core:	K	= Single-Core Device
Sub-Family:	MPS	Motor Control, Power Supply and Sensing and Real-Time Control
Pin Count:	103/303	= 36
	105/305	= 48
	106/306	= 64
Tape & Reel Option:	Blank	= Tube or Tray
	T	= Tape & Reel
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
	H	= -40°C to +150°C (High Temp)
Package:	36 M7	= VQFN (4AW)
	48 M7	= VQFN (6MX)
	48 PT	= TQFP (Y8X)
	64 M7	= VQFN (5LX)
	64 PT	= TQFP (VGX)
Pattern	V_{xx}	= 3-character standard code with a “V” prefix, specifying Automotive Qualified Product

Examples:

- dsPIC33AK256MPS306-I/PT: dsPIC33, 32-Bit Core, 256-Kbyte Program Memory, 64-Pin, Industrial temperature, TQFP package.
- dsPIC33AK256MPS306-I/PTVAO: dsPIC33, 32-Bit Core, 256-Kbyte Program Memory, 64-Pin, Industrial temperature, TQFP package, Automotive QEC-A10x Qualified.

Notes:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.
3. For more information on Automotive-Qualified Products, please see www.microchip.com/en-us/solutions/automotive-and-transportation/recommended-for-automotive, or contact your local Sales Office.

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-2991-4

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.