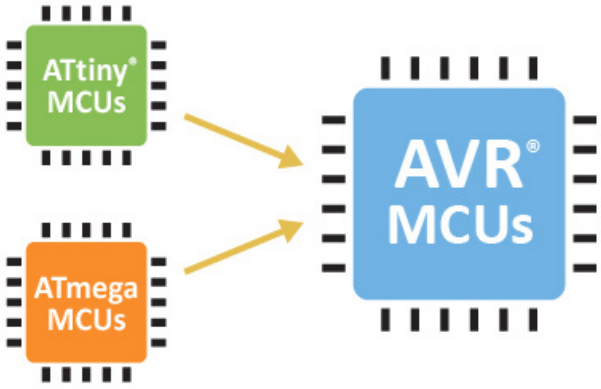
Migrating from ATtiny® and ATmega Devices to Newer AVR® Devices

Introduction

The AVR® family of microcontrollers (MCUs) is evolving, with traditional ATtiny® and ATmega families being transitioned to newer AVR devices that offer significant enhancements in efficiency, power consumption, functionality and ease of development. One of the most notable advancements is the introduction of Core Independent Peripherals (CIPs), which enable peripherals to function autonomously from the Central Processing Unit (CPU) to improve system efficiency and responsiveness.

This guide will assist you in migrating your designs based on ATtiny and ATmega MCUs to the new AVR devices. You can reuse your existing code as is, provided it does not interact directly with specific hardware. For any code that communicates directly with hardware, this guide will help you migrate or replace it to leverage the enhanced hardware capabilities of these newer AVR devices.



New AVR MCU Naming Convention

AVRxxEAyy

AVR:	xx:	EA:	yy:
Architecture or CPU	Flash Memory Size (KB)	Family Designation	Number of Pins

AVR32EA32 → 32 KB Flash, EA series, 32 pinsAVRxxEAyy

Note that before the rebranding of AVR family names, we introduced several families with the ATtiny and ATmega names. The following families are newer devices:

- ATtiny1607 (tinyAVR-0) Family
- ATtiny3217 (tinyAVR-1) Family
- ATtiny3227 (tinyAVR-2) Family
- ATmega4809 (megaAVR-0) Family

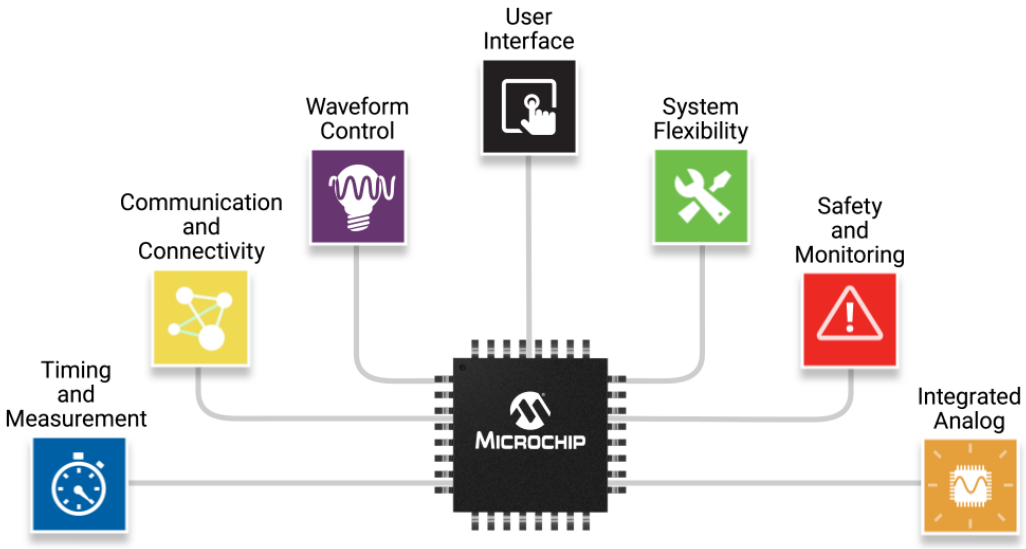
We have introduced a streamlined naming convention that helps you identify suitable MCUs more easily based on your specific requirements. This format indicates the Flash memory size and the number of pins on the device, simplifying product selection and comparison. This naming structure is intended to enable quick identification of devices that meet memory and I/O requirements.

New naming convention: AVRxxEAyy

- **AVR:** Architecture or CPU
- **xx:** Flash memory size (KB)
- **EA:** Family designation
- **yy:** Number of pins

Example: **AVR32EA32 →** 32 KB Flash, EA series, 32 pins

Modern AVR MCUs Implement CIPs



What are CIPs?

CIPs are specialized hardware modules integrated into modern AVR MCUs that can operate independently of the CPU. Unlike traditional peripherals that rely heavily on CPU intervention, CIPs can perform specific tasks such as signal processing, event triggering and data transfer autonomously.

This architecture improves **system efficiency** by:

- Reducing CPU load
- Lowering power consumption
- Enhancing real-time responsiveness

Through direct peripheral to peripheral communication, often via the Event System, CIPs allow the CPU to remain in low power states until needed. This makes them excellent options for energy-conscious and performance-critical applications.

Advantages of CIPs

1. **Faster Processing:** Tasks are handled autonomously by peripherals, allowing the CPU to stay in low-power modes. Example: Frequency detection via Configurable Custom Logic (CCL) can be done completely in hardware without CPU involvement.

2. **Reduced Power Consumption:** Peripherals operate with minimal CPU intervention. Example: The Analog Comparator (AC) monitors temperature, waking the CPU only when thresholds are exceeded.

3. **Simplified Development:** Event-driven communication reduces software complexity. Example: The Event System (EVSYS) detects Timer Overflow to trigger ADC conversion independent of CPU intervention.

4. **Enhanced Real-Time Response:** The system responds promptly to internal and external events. Example: Tasks managed by hardware such as the integrated CCL and Event System (EVSYS) can often outperform similar tasks managed by software because hardware propagation is typically much faster than software latency.

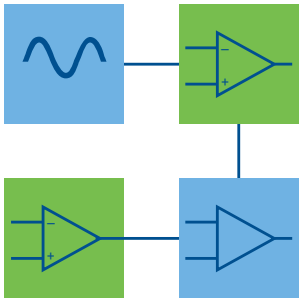
5. **Advanced Analog Features On-Chip:** CIPs deliver sophisticated analog processing with fewer external components. Example: Using the ADC with thresholding and accumulation while in lower power operating modes improves noise resilience.

6. **Lower System Cost:** Integration reduces external parts. Example: Using the ADC with integrated gain or interconnection with the on-chip OPAMP reduces the need for external signal conditioning circuits.

7. **Dynamic Reconfiguration:** More integrated features mean more control at runtime than is available with external components. Example: OPAMPs can be reconfigured at runtime as environmental changes occur, which would be complex when using their external counterparts.

Peripheral Enhancements for Efficient System Design

Many developers will recognize peripheral names such as Timer/Counter, ADC or USART from their experience with ATtiny and ATmega devices. However, it is important to note that these peripherals in newer AVR devices often include added functionality and enhancements that are not available in their legacy counterparts. These improvements not only expand the capabilities of each peripheral but also enable them to execute more tasks independently, reducing the amount of code needed for configuration and operation. These enhanced peripherals enable faster data throughput and can operate in parallel with the CPU or continue functioning in low-power modes when the CPU is inactive. This increased autonomy and performance contribute to more efficient and streamlined system designs.



Intelligent Analog Peripherals

In newer AVR devices, integrated analog is treated as a true peripheral, offering significant flexibility and control. This means that analog modules such as ADCs, DACs, comparators and operational amplifiers can be dynamically reconfigured at runtime, allowing them to adapt to changing application needs. Moreover, these analog peripherals can be interconnected with one another or with digital peripherals via features like Timers or the Event System. This interconnection enables the creation of high-performance, closed-loop control systems and complex analog-digital interactions without CPU intervention. These capabilities result in more efficient processing, reduced code complexity and support for power-saving strategies where peripherals continue operating while the CPU is in a low-power state.

Peripheral	ATtiny®/ATmega	New AVR® Devices
ADC	10-bit, basic	12-bit, faster, noise reduction Computation modes include Thresholding, Averaging and Accumulation Optional event-triggered conversion
DAC	Not available	Integrated, 10-bit
Analog Comparator	Basic	Selectable response time and hysteresis to balance power consumption Windowing function and event generation
Op Amp	Not available	Integrated connection with other peripherals and features Reconfigurable at runtime
Zero-Cross Detector	Not available	Integrated, event generation

Timers

Timers in newer AVR devices have evolved beyond the basic functionality found in older ATtiny and ATmega MCUs. While many developers may be familiar with the Timer/Counter modules from legacy devices, the updated versions in modern AVR devices feature additional operational modes, higher resolution and enhanced flexibility. These improvements allow timers to handle more sophisticated timing, waveform generation and event counting tasks with minimal CPU involvement. Many of these timers can also interconnect with other peripherals via the Event System, enabling them to operate in parallel with the CPU or continue functioning independently in low-power modes. This enhanced autonomy supports real-time control applications and improves system efficiency.

Refer to the following table to help transition from legacy ATtiny and ATmega timers to the newer, more advanced timers and leverage the additional features for improved performance and functionality.

Legacy Timer to New Timer Migration Table

Legacy Timer	Bit Width	Application Requirements	New Timer	Bit Width	Additional Capabilities of New Timer
TC0	8-bit	Basic timer/counter, single compare/capture channel, basic PWM generation	**TCB**	16-bit (can run in 8-bit mode)	Input capture, frequency measurement, noise canceler, synchronized operation
TC1	16-bit	Advanced timer/counter, multiple compare/capture channels, advanced PWM generation	**TCA**	16-bit (can run in 8-bit mode)	Multiple waveform generation modes, split mode for dual 8-bit timers, enhanced waveform generation
TC2	8-bit	Basic timer/counter, single compare/capture channel, asynchronous operation	**TCB**	16-bit (can run in 8-bit mode)	Input capture, frequency measurement, noise canceler, synchronized operation
TC3	16-bit	Advanced timer/counter, multiple compare/capture channels, advanced PWM generation	**TCA**	16-bit (can run in 8-bit mode)	Multiple waveform generation modes, split mode for dual 8-bit timers, enhanced waveform generation

New AVR MCU Timers with Advanced Capabilities

Timer	Bit Width	Application Requirements	Additional Capabilities	Enabled By
TCC	24-bit	Advanced waveform generation, multiple compare/capture channels	Higher resolution (finer timing control)	24-bit counter width
			Dead-time insertion (prevents shoot-through)	Dedicated dead-time control registers
			Fault protection (improves safety)	Built-in fault detection and handling mechanisms
TCD	12-bit	Programmable prescaler, double-buffered compare registers	Asynchronous operation (independent of main clock)	Asynchronous clock source
			Fault handling (enhances reliability)	Integrated fault detection and response features
			Integration with Event System (facilitates event-driven applications)	Direct connection to AVR MCU's Event System
TCE with WEX	16-bit	Multiple compare/capture channels, advanced waveform generation	Glitch-free PWM (smooth and stable signals)	Double-buffered compare registers and advanced PWM control logic
			Amplitude and offset scaling (dynamic adjustment)	Dedicated amplitude and offset control registers
TCF	32-bit	Multiple compare/capture channels, advanced waveform generation	Very high-resolution timing (extremely precise control)	32-bit counter width
			Precision motor control (fine control over speed and position)	High-resolution compare/capture channels and advanced control logic

Enhancements to Communication Peripherals

Communication peripherals in newer AVR devices have been significantly enhanced to support higher data rates, improved protocol handling and more flexible configurations. While familiar interfaces like USART, SPI and I^2C remain available, their updated versions offer expanded capabilities such as advanced error detection, fractional baud rate control and support for the LIN protocol. These enhancements enable more reliable and efficient communication, especially in complex or high-speed applications. Many communication peripherals also support features like double buffering and interrupt-driven operation, which improve data throughput and reduce CPU load. These peripherals can operate autonomously or in conjunction with the Event System, enabling parallel task execution and supporting low-power designs where communication tasks can continue even when the CPU is inactive.

- **USART:** More modules, LIN/IrDA support, precise baud rate control and enhanced error handling
- **SPI:** Faster communication, double-buffered data registers and interrupt-driven modes
- **I^2C/TWI:** Supports Fast Mode Plus and High-Speed Mode, improved multi-host handling and error detection
- **USB:** USB 2.0 support in more models, with endpoint buffering and power management
- **LIN, USART in SPI Mode, CRC:** Newly supported or enhanced functionalities

New Peripherals in AVR Devices

Product Families	Program Memory Size (KB)	Pin Count	10-bit ADC	12-bit ADC	OPAMP /PTC	USB	Automotive	Motor Control	Level Shifters	Touch	Functional Safety
AVR DA MCU	32–128	28–64		✓			✓			✓	✓
AVR DB MCU	32–128	28–64		✓			✓		✓		✓
AVR EA MCU	16–64	28–48		✓							✓
AVR DU MCU	16–64	14–32				✓					✓
AVR DD MCU	16–64	14–32		✓			✓		✓		✓
ATmega328 MCU	4–32	32									
AVR EB MCU	16	14–32		✓	✓			✓		✓	✓
ATtiny2 MCU	4–32	14–24	✓		✓		✓			✓	✓
ATtiny1 MCU	2–32	8–24	✓				✓				✓
ATtiny0 MCU	2–16	8–24		✓	✓		✓			✓	✓

New AVR devices introduce a range of peripherals that were traditionally not available on most legacy ATtiny and ATmega MCUs(1). These peripherals, which are listed below, expand the functional capabilities of the MCUs, enabling tasks that previously required external components or more advanced MCU families.

(1) Exceptions are the ATtiny1607, ATtiny3217, ATtiny3227 and ATmega4809 families.

- **Event System (EVSYS):** Lets peripherals send signals directly to each other without involving the CPU
  - Enables fast, predictable responses in active, idle, and standby modes
  - Events, like interrupts, can be routed in software to connect peripherals such as ADCs, timers and I/O pins
  - Excellent for real-time and low-latency applications
- **Configurable Custom Logic (CCL):** Lets you create custom digital logic circuits inside the MCU
  - Reduces the need for external logic devices
  - Useful for signal conditioning, custom timing, and control logic
- **Peripheral Touch Controller (PTC):** Built-in capacitive touch sensing for buttons, sliders and wheels
  - High sensitivity and low noise
  - Optimal for user interfaces and interactive controls
- **Real-Time Counter (RTC):** Low-power clock and calendar module
  - Supports timekeeping, alarms and scheduled operations
  - Common in battery-powered applications
- **Advanced Analog Peripherals:** Include programmable gain amplifiers (PGAs), high-resolution DACs and integrated OPAMPs
  - Enable precise signal processing, amplification and generation directly on-chip

- **Multi-Voltage I/O (MVIO):** Allows connection to peripherals using different voltage levels
  - No need for external level shifters
  - Simplifies mixed-voltage system design
- **Zero-Cross Detect (ZCD):** Detects the zero-crossing point of AC signals with high precision
  - Useful for AC power control and phase synchronization
- **Direct Memory Access (DMA):** Moves data between peripherals and memory without CPU involvement
  - Improves data throughput and responsiveness
  - Reduces CPU load
- **Waveform Extension (WEX):** Advanced waveform control features
  - Includes dead-time insertion, fault protection and complementary outputs
  - Designed for motor control, power conversion and precise waveform generation

Peripheral Migration Tips

To simplify the transition from ATtiny and ATmega devices to newer AVR devices, follow these step-by-step guidelines:

1. Review Peripheral Functionality

- Read the introductory sections of the peripheral chapters in the new AVR MCU's data sheet. This will help you understand the overall functionality of the peripheral.
- Note new features such as autonomous operation, threshold detection and accumulation.

2. Leverage MPLAB® Code Configurator (MCC) Melody

- Use MCC Melody for graphical setup
- Utilize "Easy View" for quick configuration and "Registers" tab for detailed insight.
- Contextual help provides links to macro and function description, relevant sections of an html version of the data sheet and even common use case examples for the applicable peripheral or feature being configured.

3. Generate Code and API

- **Initialize and control peripherals automatically** using MCC.
- **Generated API functions and macros** simplify common peripheral tasks by abstracting low-level register operations.
- **Intuitive naming conventions and inline comments** make the code easy to read, understand and integrate.
- **Start with production-ready code in most cases,** minimizing the need for modification before deployment.
- **Contextual help icons** in MCC provide quick access to online documentation for each peripheral or feature.

4. Replace Software Algorithms with Hardware Features

- Identify software routines that can now be offloaded to peripherals
  - **Averaging ADC readings:** Use the ADC's built-in accumulation feature.
  - **Threshold checking:** Employ ADC or comparator threshold detection.
  - **Timing loops:** Replace with timers using event-triggered interrupts.
  - **Pulse generation:** Use advanced timers (e.g., TCD, WEX) for precise PWM.
  - **Data transfer loops:** Utilize DMA for high-speed, CPU-free data movement.

Recognizing where integrated peripheral features can replace custom software algorithms allows for reduced code complexity, improved execution speed and lower power consumption. This approach helps you fully exploit the enhanced capabilities of modern AVR MCUs.

Use the "Easy View" interface within MCC Melody to quickly configure peripherals and the Registers tab to understand how configurations map to hardware. The generated API functions and macros can serve as a foundation for your application code, enabling faster development and fewer bugs. When migrating, compare the legacy peripherals you used (e.g., timers, ADCs, UARTs) with their newer counterparts (e.g., TCA, ADC with thresholding, USART with Event System support) to identify new capabilities and simplify your design. Test each peripheral incrementally, validate functionality and optimize performance through iteration. Taking these steps ensures a smoother transition and allows you to take full advantage of the powerful features available in modern AVR devices.

IDE Migration: Atmel Studio to MPLAB X IDE

Migrating from Atmel Studio to MPLAB X Integrated Development Environment (IDE) offers a more modern, flexible and integrated development experience, particularly if you are working with the latest AVR devices and other Microchip MCUs. MPLAB X IDE is cross-platform and supports a unified workflow across AVR, PIC®, PIC32 and SAM devices and dsPIC® DSCs, eliminating the need to learn multiple environments. With advanced debugging features, support for a broader range of tools and integration with MCC for graphical peripheral setup, MPLAB X IDE streamlines development and reduces setup time. Additionally, as we focus our ongoing support and feature updates on MPLAB X IDE, migrating ensures access to the latest device support, tools and ecosystem enhancements, ultimately future-proofing your development process.

Why MPLAB X IDE?

- Unified support for all Microchip MCUs
- Advanced debugging (supports PICkit™, MPLAB ICD and Atmel-ICE debuggers)
- MCC for code generation
- Cross-platform (Windows®, macOS®, Linux® operating systems)
- Plug-in ecosystem

Key Differences

Feature	Atmel Studio	MPLAB® X IDE
Platform	Windows® only	Cross-platform
Compiler	AVR-GCC	MPLAB XC8 Compiler (GCC-based)
Debug Tools	Atmel-ICE	Wide support
Libraries	ASF, START	MCC
Build System	Makefile	Apache® Ant™-based

Migration Steps

- Import your project via the MPLAB X IDE wizard.
- Configure device packs, compilers and debug tools.
- Use MCC to configure peripherals.
- Validate build, debug and refine code.

Conclusion

Migrating from ATtiny and ATmega devices to the newer AVR MCUs offers a significant upgrade in performance, power efficiency and development ease. The advanced peripherals, such as Core Independent Peripherals (CIPs), and the enhanced capabilities of modern AVR devices enable more efficient and streamlined system designs. By leveraging tools like MPLAB X IDE and MCC, you can simplify the migration process, reduce development time and fully exploit the powerful features of these new MCUs. This guide provides the necessary steps and insights to ensure a smooth transition, allowing you to confidently harness the full potential of the latest AVR devices in your applications.

DS00006188A