User manual for

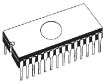
# DATAMAN-48PRO4

Advanced universal 48-pindrive programmer with USB interface and ISP capability

# DATAMAN-48PRO4C

Cost effective version of the DATAMAN-48PRO4 programmer
Universal 48-pindrive programmer with USB interface and ISP capability

20th April 2026



COPYRIGHT © 2017-2026
Dataman Programmers Ltd

This document is copyrighted by Dataman Programmers Ltd, United Kingdom. All rights reserved. This document or any part of it may not be copied, reproduced or translated in any form or in any way without the prior written permission of Dataman Programmers Ltd.

The control program is copyrighted by Dataman Programmers Ltd. The control program or any part of it may not be analyzed, disassembled or modified in any form, on any medium, for any purpose.

Information provided in this manual is intended to be accurate at the moment of release, but we continuously improve all our products. Please check for an updated manual on our website at **www.dataman.com**.

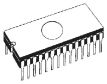
Dataman Programmers Ltd assumes no responsibility for misuse of this manual.

Dataman Programmers Ltd reserves the right to make changes or improvements to the product described in this manual at any time without notice. This manual contains names of companies, software products, etc., which may be trademarks of their respective owners. Dataman Programmers Ltd respects those trademarks.

# *How to use this manual*

This manual explains how to install the control program and how to use your programmer. It is assumed that the user has some experience with PCs and installation of software. Once you have installed the control program we recommend you consult the context sensitive HELP within the control program rather than the User manual. Revisions are implemented in the context sensitive help before the User manual.

*We continuously update our manual. You may find the latest version on our website (**www.dataman.com/pages/downloads**).*



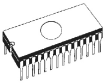
# Table of contents

Conventions used in the manual

References to the control program functions are in bold, e.g. **Load**, **File**, **Device**, etc. References to control keys are written in brackets **<>**, e.g. **<F1>.**

Terminology used in the manual:

| | |
|---|---|
| ***Device*** | any kind of programmable integrated circuits or programmable devices |
| ***ZIF socket*** | Zero Insertion Force socket used for insertion of target device |
| ***Buffer*** | part of memory or disk, used for temporary data storage |
| ***USB port*** | type of PC port (serial), which is dedicated for connecting portable and peripheral devices. |
| ***HEX data format*** | format of data file, which may be read with standard text viewers; e.g. byte 5AH is stored as characters '5' and 'A', which mean bytes 35H and 41H. One line of this HEX file (one record) contains start address and data bytes. All records are secured with checksum. |



# Introduction

**DATAMAN-48PRO4** is a USB interfaced universal programmer and logic IC tester with 48 powerful pindrivers. Using a built-in ISP connector the programmer is able to program ISP capable chips in circuit. This design allows us to easily add new devices to the device list.

**DATAMAN-48PRO4C** is a cost effective version of the **DATAMAN-48PRO4** programmer (without certain special devices). If you need to program some of the mentioned devices, see the DATAMAN-48PRO4 programmer.

**DATAMAN-48PRO4 / DATAMAN-48PRO4C** work with almost any Windows® based compatible, portable or desktop personal computers.

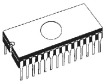
**DATAMAN-48PRO4 / DATAMAN-48PRO4C** is driven by an **easy-to-use, control program** with pull-down menus, hot keys and online help. The control program is common for all DATAMAN programmers.

Advanced design, including protection circuits, original brand components and careful manufacturing allows us to provide a **three-year warranty** for DATAMAN-48PRO4 / DATAMAN-48PRO4C on parts and workmanship of the programmers (limited 25,000 cycle warranty on the ZIF socket). These warranty terms are valid for customers who purchase a programmer directly from Dataman. The warranty conditions of Dataman distributors may be different and depending on the law or the reseller's warranty policy.

# *Products configuration*

Before installing and using your programmer, please carefully check that your package includes all of the following parts. If you find any discrepancy with the respective parts list and/or if any of these items are damaged, please contact your distributor immediately.

| | DATAMAN-48PRO4 | DATAMAN-48PRO4C |
|---|---|---|
| Programmer | • | • |
| USB cable 1,8m | • | • |
| ISP cable 0,2m | • | • |
| Power cordset 1,2m | • | • |
| External power supply | • | • |
| 48 pins diagnostic pod – type I | • | • |
| Diagnostic pod for ISP connectors #2 | • | • |
| ZIF anti-dust cover | • | • |
| Quick Guide | • | • |
| Transport case | • | • |



# *PC requirements*

Minimum PC requirements

| | DATAMAN-48PRO4 | DATAMAN-48PRO4C |
|---|---|---|
| OS - Windows | Windows 10 | Windows 10 |
| CPU | Core2Duo | Core2Duo |
| RAM [MB] | 1000 | 1000 |
| Free disk space [MB] | 500 | 500 |
| USB 2.0 high speed | • | • |

*The "Minimum PC requirements" mean that the device programmer and SW will run at these conditions, but not with fully enjoyable experience. 1024 x 768 is the minimum monitor resolution.*

Optimal PC requirements

| | DATAMAN-48PRO4 | DATAMAN-48PRO4C |
|---|---|---|
| OS - Windows | Windows 10*1 | Windows 10*1 |
| CPU | Core i3*2 | Core i3*2 |
| RAM [MB] | 4000*3 | 4000*3 |
| Free disk space [MB] | 2000*3 | 2000*3 |
| USB 2.0 high speed | • | • |

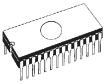
*We recommend using a higher monitor resolution such as 1024 x 768.*
**1 – or newer* **2 – or better* **3 – or more*
These PC requirements are valid for v4.11 of the PG4UW (issued 4.2026) and above.

Free disk space requirement depends on the selected IC device size and the number of attached programming sites. For large devices the required free space on disk will be approximately *1000MB* + 2x *Device size* x *number of programming sites* attached to this PC.

A very easy indication, if your PC hardware/software configuration is good enough for the current software version and the current situation with PG4UW/PG4UWMC, is to run Windows task manager (Ctrl+Alt+Del) and see the performance folder. It should be max. 80% of CPU usage at full operation of the programming system.

# *Additional services:*

- Free technical support (e-mail / phone) **www.dataman.com/pages/contact-us.**
- Free lifetime software updates via our website **www.dataman.com/pages/downloads.**



# Quick Start

Installing the programmer software

Download the latest version of the software for your programmer from **www.dataman.com/pages/downloads**.
Run the install program with the programmer disconnected or powered off, which will guide you through the installation process and undertake all the necessary steps before you can first run the control program.

Installing programmer hardware

Connect the USB port of the programmer to the USB port of the host PC using the enclosed USB cable.
Connect the power supply to the programmer.

Run the control program

Double click on the  icon on your desktop.

After the control software starts, PG4UW automatically scans all existing ports and searches for any connected DATAMAN programmers. The PG4UW control software is common for all of the programmers mentioned above; PG4UW will try to find all of the supported programmer types.

**File** menu is used for source file selection, settings and directory selection, changing drives, changing the start and finish addresses of the buffer when loading and saving files and the loading and saving of projects.

**Buffer** menu is used for viewing/editing data in the buffer, block operations, filling part of the buffer with a string, erasing the buffer, checksum calculations and other operations with the data (find and replace string, printing buffer contents...).

**Device** menu is used for functions with the selected programmable device: select, read, blank check, program, verify, erase and setting of the programming process, serialization and associated file control.

**Programmer** menu is used for functions relating to the programmer.

**Options** menu is used to view and change various default settings in the software.

**Help** menu is used for viewing the supported devices and programmers and information about the program version.

Programming a device

1. select device: click on 
2. load data into buffer:
   a) from file: click on 
   b) from device: insert device to ZIF and click on 

3. insert target device to the ZIF
4. check, if the device is blank: click on 

5. program device: click on 
6. additional verify of device: click on 

# Detailed description

# DATAMAN-48PRO4 / DATAMAN-48PRO4C

# *Introduction*

**DATAMAN-48PRO4** is a USB interfaced universal programmer built to meet the strong demand of the small manufacturing and developer's community for a fast and reliable universal programmer.

**DATAMAN-48PRO4C** is a cost effective version of the **DATAMAN-48PRO4** programmer. **DATAMAN-48PRO4C** programmer has hardware identical to the **DATAMAN-48PRO4** programmer; the differences are only in software. **DATAMAN-48PRO4C** has the following limitations against the **DATAMAN-48PRO4**:

- unsupported obsolete bipolar chips (PROMs and PLDs)
- unsupported obsolete EPROMs with programming voltage 21V and above
- unsupported obsolete 2708 EPROMs
- unsupported obsolete microcontrollers MCS48 series
- unsupported obsolete microcontrollers 8751/8752 with programming voltage 21V and above
- unsupported some very obsolete chips and special implementations

If you need to program some of the mentioned devices, see **DATAMAN-48PRO4** programmer.

**DATAMAN-48PRO4 / DATAMAN-48PRO4C** Supports all current and future silicon technologies and programmable device types without requiring family-specific modules.
. You have the freedom to choose the optimal device for your design. Using the built-in in-circuit serial programming (**ISP**) connector, the programmer is able to program ISP capable chips in circuit.

**DATAMAN-48PRO4 / DATAMAN-48PRO4C** aren't only programmers, but also **testers** of TTL/CMOS logic ICs and memories. Furthermore, it allows generating user-definable test pattern sequences.

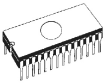
**DATAMAN-48PRO4 / DATAMAN-48PRO4C** provides **fast programming** due to high-speed FPGA driven hardware and execution of time-critical routines inside of the programmer. As a result, when used in manually operated production, this one-socket-programmer in most cases waits for the operator.

**DATAMAN-48PRO4 / DATAMAN-48PRO4C** interfaces with the IBM PC compatible personal computers running MS Windows OS through USB (2.0 high speed).

**DATAMAN-48PRO4 / DATAMAN-48PRO4C** provides a banana jack for ESD wrist strap connection for easy-to-implement ESD protection control and also another banana jack for the earth wire.

**DATAMAN-48PRO4 / DATAMAN-48PRO4C** as a FPGA based totally reconfigurable 48 powerful TTL pindrivers, which provide H/L/pull_up/pull_down and read capability for each pin of the socket. Advanced pindrivers incorporate **high-quality high-speed** circuitry to deliver signals without overshoot or ground bounce for all supported devices. Improved pindrivers operate down to 1.8V so you'll be ready to program the full range of today's advanced low-voltage devices.

**DATAMAN-48PRO4 / DATAMAN-48PRO4C** performs a **device insertion test** based on checking the proper signal path between the programmer and programmed device before it programs each device. Independent of programming configuration it identifies missed or poor



contact between the programmed device and the ZIF socket of the programming adapter (or the programmer directly), missed or poor contact between the programming adapter and the programmer and it is also able to indicate a **wrong position** of the device in the ZIF socket of the programmer / the programming adapter (moved, rotated, backwards oriented). These capabilities, supported by **overcurrent protection** and **signature-byte check** help prevent chip damage due to operator error.

The selftest capability allows running a diagnostic part of software to thoroughly check the health of the programmer.

Built-in **protection circuits** eliminate damage of the programmer and/or programmed device due to the environment or operator failure. All the inputs of the **DATAMAN-48PRO4 / DATAMAN-48PRO4C** programmer, including the ZIF socket, ISP connector, connection to PC and power supply input, are **protected against ESD** up to 15kV.

When the programming specification requires it, the **DATAMAN-48PRO4 / DATAMAN-48PRO4C** programmer performs programming verification at the marginal level of supply voltage, which, obviously, improves programming yield, and guarantees long data retention.

Various **programming adapters** are available to handle devices in PLCC, JLCC, SOIC, SDIP, SOP, PSOP, SSOP, TSOP, TSOPII, TSSOP, QFP, PQFP, TQFP, VQFP, QFN (MLF), SON, BGA, EBGA, FBGA, VFBGA, UBGA, FTBGA, LAP, CSP, SCSP, LQFP, MQFP, HVQFN, QLP, QIP and other packages..

**DATAMAN-48PRO4 / DATAMAN-48PRO4C** programmer is driven by an **easy-to-use** control program with pull-down menu, hot keys and on-line help. Selecting of the device is performed by its class, by manufacturer or simply by typing a fragment of the vendor name and/or part number.

**Standard** device-related commands (read, blank check, program, verify, erase) are boosted by some **test functions** (insertion test, signature-byte check), and some **special functions** (autoincrement, production mode - start immediately after insertion of chip into the socket).

All known data formats are supported. Automatic file format detection and conversion during loading of the file.

The rich-featured **autoincrement function** enables one to assign individual serial numbers to each programmed device - or simply increments a serial number, or the function enables one to read serial numbers or any programmed device identification signatures from an external file.

The software also provides a lot of information about programmed devices. The **drawings of all available packages**, explanation of **chip labeling** (the meaning of prefixes and suffixes on the chips) for each supported chip are provided.

The software provides full information for ISP implementation: Description of the ISP connector pins for the currently selected chip, recommended target design around the in-circuit programmed chip, and other necessary information.

The **remote control** feature allows the PG4UW software flow to be controlled by another application – either using .BAT file commands or using DLL file. DLL file, examples (C/PAS/VBASIC/.NET) and a manual are part of standard software delivery.

**Jam files** of JEDEC standard JESD-71 are interpreted by the **Jam Player**. Jam files are generated by design software which is provided by the manufacturer of the respective programmable device. Chips are programmed in ZIF or through the ISP connector (IEEE 1149.1 Joint Test Action Group (JTAG) interface).

**VME files** are interpreted by the VME Player. SVF (Serial Vector Format) files are interpreted by the SVF Player. A VME file is a compressed binary variation of an SVF file and both contain high-level IEEE 1149.1 bus operations. SVF and VME files are generated by design software which is provided by the manufacturer of the respective programmable device. Chips are programmed in ZIF or through the ISP connector (IEEE 1149.1 Joint Test Action Group (JTAG) interface).

Multiple devices are possible to program and test via JTAG chain: JTAG chain (ISP-Jam) or JTAG chain (ISP-VME).

Attaching more DATAMAN-48PRO4 / DATAMAN-48PRO4C programmers to the same PC (through the USB port) achieves a **powerful multiprogramming system**, which **supports as many chips as are supported by DATAMAN-48PRO4 / DATAMAN-48PRO4C programmer** and without obvious decrease of **programming speed**. It is important to know, there is concurrent multiprogramming - each programmer works independently and each programmer can program a different chip, if necessary.

It is important to remember that in most cases new devices require **only a software update** due to the DATAMAN-48PRO4 / DATAMAN-48PRO4C being a truly universal programmer. With our prompt service you can have new devices added to the current list within hours!

Advanced design including protection circuits, original brand components and careful manufacturing and burning-in allows us to provide a **three-year warranty** on parts and workmanship of the programmer **DATAMAN-48PRO4 / DATAMAN-48PRO4C** (limited 25,000-cycle warranty on ZIF DIL48 socket).

# *DATAMAN-48PRO4 / DATAMAN-48PRO4C elements*

1. 48 pin ZIF socket
2. Work result LEDs
3. Power/sleep LED
4. YES! Button
5. ISP connector


6. "ESD wrist strap" connector is place for attaching an ESD wrist strap
7. "GND" connector can be used for grounding of the programmer
8. Power switch
9. Power supply connector
10. USB connector for PC ↔ DATAMAN-48PRO4 / DATAMAN-48PRO4C communication cable

# Connecting DATAMAN-48PRO4 / DATAMAN-48PRO4C to the PC by USB port

Recommendation for connecting the programmer to the PC:

1. make the ground connection between programmer and PC or another ground
2. connect programmer with PC via USB cable
3. connect power supply to programmer and turn on using power switch “8”.
4. run PG4UW control program and find programmer

Problems related to the DATAMAN-48PRO4 / DATAMAN-48PRO4C ⇔ PC interconnection, and their troubleshooting

If you have any problems with DATAMAN-48PRO4 / DATAMAN-48PRO4C ⇔ PC interconnection, see section ***Common notes***.

# Manipulation with the programmed device

After selection of the desired device for your work, you can insert it into the open ZIF socket (the lever is up) and close the socket (the lever is down). The correct orientation of the programmed device in the ZIF socket is shown on the picture near ZIF socket on the programmer's cover. The programmed device can only be inserted or removed from the socket when the BUSY LED light is off.

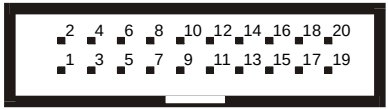
**Note:** *The programmer’s internal electronics protect both the programmer and the target device from short or long-term power outages and most PC failures. However, we cannot guarantee the integrity of the target device if incorrect programming parameters are selected by the user.*
*While the target device will not be damaged by a forced interruption (such as a PC reset, power-off, or disconnection), the data in the specific memory cell being programmed at that moment may become corrupted or undefined.* ***Do not remove the device from the ZIF socket while the "BUSY" LED is lit.***

# In-system serial programming with DATAMAN-48PRO4 / DATAMAN-48PRO4C

***For general definitions, recommendations, and instructions regarding ISP, please refer to the "Common notes / ISP" section.***

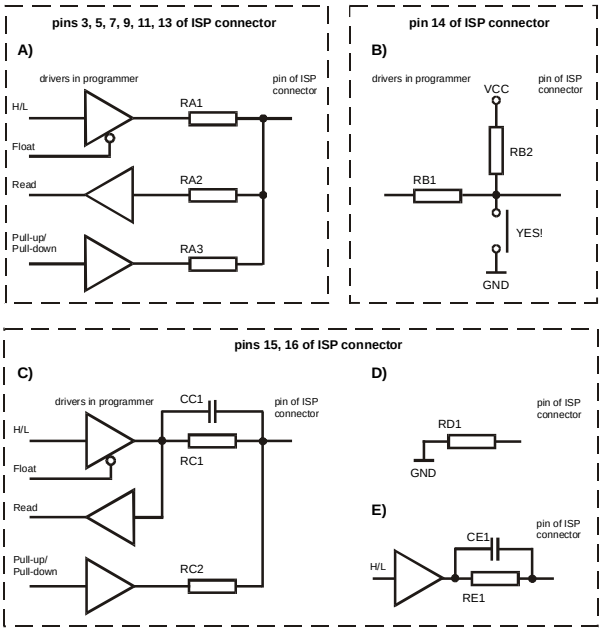
***Description of ISP connector***

The programmer uses a 20-pin, dual-row ISP connector with a 2.54mm (0.1”) pitch, such as the TE Connectivity 5103310-5 or an equivalent compatible model.




Front view of ISP connector on the programmer.

H/L/read driver




RA1 180R, RA2 1k3, RA3 22k,
RB1 10k, RB2 10k,
CC1 1n, RC1 1k3, RC2 22k,
RD1 22k, CE1 1n, RE1 1k3,

*Comments on the diagram above:*

- B) Pin 14 is an input pin. Pulse to logical L has the same function as pressing the Yes! button on top of the programmer,
- C) Connection of pins 15 and 16 when configured as a logical signal needed for ISP programming,
- D) E) When pins 15 and 16 are configured to indicate "LED OK" and "LED ERROR" status, they function as output pins.
- D) Before first action with desired ISP device,
- E) After first action with desired ISP device.

**Notes:**

- *When LED OK or LED ERROR ON (shine), this status is presented as logical H, level of H is 1,8V - 5V depending on the H level of the desired ISP device.*
- *When LED OK or LED ERROR OFF (not shine), this status is presented as logical L, level of L is 0V - 0,4V.*
- *The above mentioned values are provided to understand (and also to exactly calculate) the value of resistors, which isolate (separate) the programmed chip and target system.*
- *These signals may stay unconnected if you don't use them.*

Specification of the ISP connector pins depends on the device you want to program. You can find this information in the control SW for the programmer (PG4UW), menu **Device / Device Info (Ctrl+F1)**. Be aware, the ISP programming of the respective device must be selected. It is indicated by the (ISP) suffix, after the name of the selected device.
These specifications correspond with application notes published by the device manufacturers.
**Notes:**

- *Pin no. 1 is signed by a triangle mark on the ISP cable connectors*.
- *The ISP cable uses a 20-position, dual-row connector with a 2.54mm (0.1") pitch, such as the Harting 09185207813 or an equivalent compatible model.*

DATAMAN-48PRO4 / DATAMAN-48PRO4C ISP cable

**Warnings:**

- ***When** you use DATAMAN-48PRO4 / DATAMAN-48PRO4C as an **ISP programmer**, **do not insert** any device into the **ZIF socket**.*
- ***When** you program devices in the **ZIF socket**, **do not insert** the ISP cable into the **ISP connector**.*
- *Use only the provided **ISP cable**. When you use another ISP cable (other material, length…), unreliable programming may occur.*
- ***DATAMAN-48PRO4 / DATAMAN-48PRO4C can supply** the programmed device (pin 1 of ISP connector) and the target system (pins 19 and 20 of ISP connector) with certain limitations (see Technical specification / ISP connector).*
- ***DATAMAN-48PRO4 / DATAMAN-48PRO4C** applies the programming voltage to the target device and checks the value (the target system can modify the programming voltage). If the programming voltage is different than expected, no action with the target device will be executed.*

# *Multiprogramming using DATAMAN-48PRO4 / DATAMAN-48PRO4C*

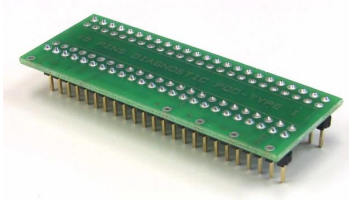
The default installation of PG4UW includes the multiprogramming control support for **DATAMAN-48PRO4 / DATAMAN-48PRO4C**.
To start the multiprogramming with **DATAMAN-48PRO4 / DATAMAN-48PRO4C** it is necessary to run the special control program **pg4uwmc.exe**. Using this program the user is able to assign one or more **DATAMAN-48PRO4 / DATAMAN-48PRO4C** to the control program, load projects for all connected **DATAMAN-48PRO4 / DATAMAN-48PRO4C** and run an instance of PG4UW for every connected and assigned **DATAMAN-48PRO4 / DATAMAN-48PRO4C**.

# *Selftest*

If you feel that your programmer is not working correctly, please run the programmers (ISP connector) selftest using the Diagnostic pod (Diagnostic pod for ISP connectors #2), included with the standard delivery package.

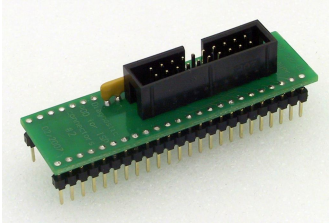
***Selftest of programmer***

- Insert the **48 pin diagnostic pod - type I** into the ZIF socket of the programmer. The **48 pin diagnostic pod - type I** must be inserted as a 48 pin device.
- Run the selftest of the programmer in PG4UW (menu Programmer / Selftest).



***Selftest of ISP connector***

- Insert the **Diagnostic pod for ISP connectors #2** into the ZIF socket of the programmer. The **Diagnostic pod for ISP connectors #2** must be inserted as a 48 pin device.
- Interconnect the 20 pin connector of the **Diagnostic pod for ISP connectors #2** with the ISP connector of the programmer using the ISP cable included with the programmer. Make sure that the pins are interconnected properly (i.e. 1-1, 2-2 ... 20-20).
- Run the selftest of the ISP connector in PG4UW (menu Programmer / Selftest ISP connector…).



# *Technical specification*

HARDWARE

Base unit, DACs

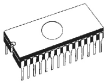
- USB 2.0 high-speed compatible port, up to 480 Mb/s transfer rate
- on-board intelligence: powerful microprocessor and FPGA based state machine
- three D/A converters for VCCP, VPP1, and VPP2, controllable rise and fall time
- VCCP range 0...8V/1A
- VPP1, VPP2 range 0...26V/1A
- selftest capability
- protection against surge and ESD on power supply input, USB port connection
- banana jack for ESD wrist strap connection
- banana jack for connection to ground

Socket, pindriver

- 48-pin DIL ZIF (Zero Insertion Force) socket accepts both 300/600 mil devices up to 48-pins
- pindrivers: 48 universal drivers
- VCCP / VPP1 / VPP2 can be connected to each pin
- perfect ground for each pin
- FPGA based TTL driver provides H, L, CLK, pull-up, pull-down on all pindriver pins
- analog pindriver output level selectable from 1.8 V up to 26V
- current limitation, overcurrent shutdown, power failure shutdown
- ESD protection on each pin of the socket (IEC1000-4-2: 15kV air, 8kV contact)
- continuity test: each pin is tested before every programming operation

ISP connector

- 20-pin male type connector with miss-insertion lock
- 6 TTL pindrivers, provide H, L, CLK, pull-up, pull-down; H level selectable from 1.8V up to 5V to handle all (low-voltage) devices.
- 1x VCCP voltage (range 2V...7V/100mA), can be applied to two pins
- programming chip voltage (VCCP) with both source/sink capability and voltage sense
- and 1x VPP voltage (range 2V...25V/50mA), can be applied to six pins
- Target system power supply voltage (range 2V...6V/250mA)
- ESD protection on each pin of the ISP connector (IEC1000-4-2: 15kV air, 8kV contact)
- For ISP devices only: two output signals, which indicate state of work result = LED OK and LED Error (active level: min 1.8V)
- input signal, YES! switch equivalent (active level: max 0.8V)



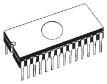
DEVICE SUPPORT

**DATAMAN-48PRO4C** has the following limitations against the **DATAMAN-48PRO4**:

- unsupported obsolete bipolar chips (PROMs and PLDs)
- unsupported obsolete EPROMs with programming voltage 21V and above
- unsupported obsolete 2708 EPROMs
- unsupported obsolete microcontrollers MCS48 series
- unsupported obsolete microcontrollers 8751/8752 with programming voltage 21V and above
- unsupported some very obsolete chips and special implementations

***Programmer, in ZIF socket***

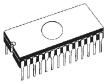
- ***Parallel NAND Flash: 3D Plus 3DFNxxx, All-Flash AFAxxx, Alliance Memory AS9Fxxx, AMD AM30LVxxx, ATO Solution AFNDxxx, MST5xxx, Axia Memory Tech. AX20xxx, Chipsip CT49xxx, Deutron Electr. ASUxxx, Dosilicon FMNxxx, DSNDxxx, Eon Silicon Sol. EN27xxx, ESIGMA ES27xxx, ES71xxx, ESMT F59xxx, FM6xxx, Flexxon FHNDxxx, Fidelix FMNxxx, FORESEE FS33xxx, FSNxxx, Fudan Microelect. FM29xxx, GigaDevice GD9xxx, MD5Nxxx, GSTO GSSxxx, Hana Micron HM34xxx, HeYangTek HYNxxx, ICMAX IMSxxx, ISSI IS34xxx, IS35xxx, Intel JS29Fxxx, Jeju Semicoductor JS27xxx, JSFxxx, Kingston Techn. KSLxxx, KIOXIA (ex Toshiba) TC58xxx, TH58xxx, TY8xxx, TY9xxx, Macronix MX30xxx, MX60xxx, MX63xxx, Micron MT29Axxx, MT29Cxxx, MT29Fxxx, MT29Gxxx, MT29Rxxx, Micron (ex Numonyx ex STMicroelectronics) NANDxxx, MIRA P1Uxxx, PSUxxx, NANYA NM1xxx, NeuMem NM9Fxxx, Paragon PN61xxx, PowerFlash A1Uxxx, ASUxxx, FSUxxx, Samsung K5xxx, K9xxx, KM29Nxxx, KFGxxx, KFMxxx, SK Hynix H27xxx, HY27xxx, H8xxx, H9xxx, HYCxxx, HYDxxx, HYGxxx, SkyHigh (ex Cypress ex Spansion) S34Mxxx, Spansion S19MNxxx, S30Mxxx, S76xxx, Spectek FxxL41Axx, FxxL41Bxx, FxxL51Axxx, FxxL52Axxx, FxxM29Bxxx, FxxM40Axxx, FxxM49Axxx, UnilC SCNxxx, VDATA VDxxx, WeForce CT49xxx, Winbond W29Nxxx, W71xxx, XTX PN61xxx, XT27xxx, XT61xxx, YuXin Semicond. YX21xxx, Zentel A5Uxxx, Zetta ZDNDxxx, LBA-NAND Toshiba THGVNxxx***
- ***Serial NAND flash: All-Flash AFAxxx, Alliance Memory AS5Fxxx, ATO Solution ATOxxx, Auton Technology 830Axxx, BIWIN BWxxx, Dosilicon DS35xxx, Etron Technology EM73xxx, EM78xxx, ESMT F50xxx, Fidelix FM35xxx, FORESEE FS35xxx, Fudan Microelect. FM25xxx, GigaDevice GD5Fxxx, GSTO GSSxxx, GTSxxx, Fudan Microelect. FM25xxx, HeYangTek HY25xxx, HYFxxx, HYLxxx, HUYANG 5Fxxx, ISSI IS37xxx, Jeju Semicoductor JS28xxx, Macronix MX31xxx, MX35xxx, MEMORITEK MK60xxx, Micron MT29Fxxx, MK Founder MKSxxx, Paragon PN26xxx, SiliconGo SGMxxx, SkyHigh S35xxx, SS SS25xxx, UnilC SCNxxx, UNIM UM19xxx, TX25xxx, Winbond W25Nxxx, XINCUN XCSPxxx, XTX PN26xxx, XT26xxx, YuXin Semicond. YX25xxx, Zentel ANVxxx, Zetta ZD35xxx, ZTE ZX29xxx***
- ***HyperFlash: Cypress/Spansion/ISSI S26KL, S26KS series, Cypress/Spansion S26HL, S26HS series***
- ***eMMC: Flexxon FEMCxxxxxx, Greenliant GLS85VMxxxxx, ISSI IS22xxxx, Kingston EMMCxxx, KIOXIA (ex Toshiba) THGBMxxxxxxxxx, Micron MTFCxxxxxx, Phison PTExxxxx, Samsung KLMxxxxxxx, SanDisk SDINxxxx, SK Hynix (ex Hynix) H26Mxxxxxxxx, Swissbit SFEMxxxxxxxxxxxxxxxxx series***
- ***eMCP: Micron MT29xZZZxxxxxxxx***
- ***eSD: XTX Technology XTSDxxGLGxxx, All-Flash SDVxxxMxxxxx series***
- ***Memory Cards: MMC, SD, SDHC, SDXC series***
- ***Multi-chip devices: NAND+RAM, NOR+RAM, NOR+NOR+RAM, NAND+NOR+RAM***
- ***Serial NOR Flash: standard SPI, Multi I/O - Dual I/O, Quad I/O, Dual-Quad I/O, Octa I/O (25Bxxx, 25Dxxx, 25Exxx, 25Fxxx, 25Hxxx, 25Lxxx, 25Mxxx, 25Pxxx, 25Qxxx, 25Rxxx, 25Sxxx, 25Txxx, 25Uxxx, 25Vxxx, 25Wxxx, 25Xxxx, 25LW/UW/LM/UMxxx, XPxxx, 26Vxxx, 28HS/HLxxx, 35Xxxx, 45PExx, 55LX/Xxxx, 65U/Lxxx, 66L/LM/UM/UWxxx, 70FL/FSxxx, 74xxx, 77xxx, 78xxx, 79FLxxx, 98FLxxx series), DataFlash (AT45Dxxx, AT26Dxxx series)***
- ***Parallel NOR Flash: 28Fxxx, 29Cxxx, 29Fxxx, 29BVxxx, 29LVxxx, 29ALxxx, 29GLxxx, 29JLxxx, 29PLxxx, 29Wxxx, 49Fxxx series, Samsung K8Fxxxx, K8Cxxxx, K8Sxxxx, K8Pxxxx series, ...***
- ***EPROM: NMOS/CMOS, 27xxx and 27Cxxx series***
- ***EEPROM: NMOS/CMOS, 28xxx, 28Cxxx, 27EExxx series, 3D Plus 3DEExxxxxxxx***



- ***mDOC H3: SanDisk (ex M-Systems) SDED5xxx, SDED7xxx, MD2533xxx, MD2534xxx, Hynix HY23xxx***
- ***FRAM: Ramtron***
- ***MRAM: Everspin MRxxxxx8x, 3D Plus 3DMRxxxxxxxx***
- ***NV RAM: Dallas DSxxx, SGS/Inmos MKxxx, SIMTEK STKxxx, XICOR 2xxx, ZMD U63x series***
- ***Serial EEPROM: 11LCxxx, 24Cxxx, 24Fxxx, 25Cxxx, 30TSExxx, 34Cxxx, 34TSxx, 59Cxxx, 85xxx, 93Cxxx, NVM3060, MDAxxx series, SPD5118, full support for LV series, AT88SCxxx***
- ***Serial FRAM: Infineon(Cypress, Ramtron): FM25xxxxxx, RAMXEED(Fujitsu): MB85RCxxxx, MB85RSxxxx, Lapis(OKI, Rohm): MR44xxxxx, MR45xxxxx, Smart Memories: SF24Cxxx, SF25Cxxx***
- ***Serial MRAM: Everspin: MH20xxx, MH25xxx , Avalanche Tech.: AS100xxxxx, AS100xxxxx, Renesas(IDT): M30xx204xxxx***
- ***Serial NV RAM: Infineon(Cypress): CY14Mxxxx, Anvo-Systems ANV32xxxx***
- ***Serial ReRAM: RAMXEED(Fujitsu): MB85ASxxx***
- ***Configuration (EE)PROM: XCFxxx, XC17xxxx, XC18Vxxx, EPCxxx, EPCSxxx, EPCQxxx, AT17xxx, AT18Fxxx, 37LVxx***
- ***1-Wire E(E)PROM: DS1xxx, DS2xxx***
- ***Optical Transceivers: SFP, SFP+, QSFP+, XFP***
- ***PLD Altera: MAX 3000A, MAX 7000A, MAX 7000B, MAX 7000S, MAX7000AE, MAX II/G/Z, MAX V, MAX 10***
- ***PLD Intel: MAX II/G/Z, MAX V, MAX 10***
- ***PLD Lattice: ispGAL22V10x, ispLSI1xxx, ispLSI1xxxEA, ispLSI2xxx, ispLSI2xxxA, ispLSI2xxxE, ispLSI2xxxV, ispLSI2xxxVE, ispLSI2xxxVL, LC4xxxB/C/V/ZC/ZE, M4-xx/xx, M4A3-xx/xx, M4A5-xx/xx, M4LV-xx/xx, ispCLOCK, Power Manager/II, ProcessorPM, iCE40 UltraPlus, iCE40 Ultra/UltraLite, iCE40 LP/HX, CrossLink, CrossLinkPlus, Platform Manager/2, L-ASC10***
- ***PLD: Xilinx: XC9500, XC9500XL, XC9500XV, CoolRunner XPLA3, CoolRunner-II***
- ***SPLD/CPLD series: AMD, AMI, Atmel, Cypress, Gould, ICT, Lattice, National Semicond., Philips, STMicroelectronics, TI (TMS), Vantis, VLSI***
- ***FPGA: Anlogic SALELF2, SALELF3***
- ***FPGA: Intel/Altera, non-volatile key: Cyclone V, Arria 10***
- ***FPGA: Gowin LittleBee, Arora***
- ***FPGA: JingZhong Wily***
- ***FPGA: Lattice: MachXO, MachXO2, LatticeXP, LatticeXP2, ispXPGA, MachXO3, MachXO3D, MachXO5-NX***
- ***FPGA: Lattice, non-volatile key: ECP2 S-Series***
- ***FPGA: Microsemi(Actel): ProASIC3, IGLOO, Fusion, ProASICplus, SmartFusion***
- ***FPGA: Pango Compa***
- ***FPGA: Xilinx: Spartan-3AN***
- ***Clocks: Cypress, Renesas (IDT), Silicon Laboratories, Skyworks Solutions, TI(TMS)***
- ***Special chips:***
- ***Tire Pressure Monitoring: Atmel: ATA6285N, ATA6286N, Infineon: SP3x***
- ***Digital Multiphase/Power controllers/Power modules: Infineon(IR, CHiL Semiconductor): XDPE1xxxxx, TDA38xxx, IR3xxxx, CHL8xxxx, MPS: MP2xxxx, MPM54xxx, Analog Devices (Linear Technology, Maxim): ADM1xxx, ADP10xx, LTC2xxx, LTC3xxx, LTC7xxx, LTM4xxx, MAX20xxx, MAX34xxx, TI (TMS) TPS5xxxx, TPS6xxxx, UCD3xxxxx, UCD9xxxxx, Renesas(Intersil): RAA228xxx, RAA229xxx,***

***ISL68xxx, ISL69xxx, Maxlinear(Exar): XR771xx, XRP77xx, PWM controllers: Zilker Labs ...***

- ***Capacitive Sensors: Azoteq IQSxxxx***
- ***Gama buffers: AUO, Maxim, TI, ...***
- ***SoC and modules Espressif: ESP32, ESP32-C2, ESP32-C3, ESP32-C6, ESP32-S2, ESP32-S3, ESP32-H2 series***
- ***Microcontrollers MCS51 series: 87Cxxx, 87LVxx, 89Cxxx, 89Sxxx, 89Fxxx, 89LVxxx, 89LSxxx, 89LPxxx, 89Exxx, 89Lxxx, all manufacturers, Philips LPC series***
- ***Microcontrollers Intel 196 series: 87C196 KB/KC/KD/KT/KR/...***
- ***Microcontrollers Artery: AT32F403xxx, AT32F421xxx***
- ***Microcontrollers Atmel ARM. AT91SAM7Sxx, AT91SAM7Lxx, AT91SAM7Xxx, AT91SAM7XCxx, AT91SAM7SExx series;***
- ***Microcontrollers Atmel ARM9: AT91SAM9xxx series;***
- ***Microcontrollers Atmel ARM Cortex-M3: ATSAM3Axxx, ATSAM3Uxxx, ATSAM3Nxxx, ATSAM3Sxxx, ATSAMD20, ATSAM3Xxxx series***
- ***Microcontrollers Atmel ARM Cortex-M4: ATSAM4Sxxx series***
- ***Microcontrollers Atmel/Microchip AVR 8bit/16bit: AT90Sxxxx, AT90pwm, AT90can, AT90usb, ATtiny, ATmega, ATxmega series***
- ***Microcontrollers Atmel/Microchip AVR32: AT32UC3xxxx, ATUCxxxD3/D4/L3U/L4U series***
- ***Microcontrollers Atmel/Microchip ARM Cortex-M0+: ATSAMD2x, ATSAMDA1, ATSAMC2x, ATSAMD09x, ATSAMD10x, ALSAML2x, ATSAMHAx, ATSAMR2x, ATSAMR3x series***
- ***Microcontrollers Atmel/Microchip ARM Cortex-M4: ATSAM4Sxxx, ATSAM4Ex, ATSAM4Cx, ATSAMG5x, ATSAM4LCx, ATSAM4LSx series***
- ***Microcontrollers Coreriver (Gencore): Atom 1.0, MiDAS 1.0, 1.1, 2.0, 2.1, 2.2, 3.0 series***
- ***Microcontrollers Cypress: CY7Cxxxxx, CY8Cxxxxx***
- ***Microcontrollers ELAN: EM78Pxxx***
- ***Microcontrollers EPSON: S1C17 series***
- ***Microcontrollers Explore Microelectronic: EPF01x, EPF02x series***
- ***Microcontrollers Geehy: APM32A0xx, APM32A1xx, APM32A4xx, APM32E0xx, APM32E1xx, APM32F00x, APM32F0xx, APM32F1xx, APM32F4xx***
- ***Microcontrollers Generalplus: GPM8Fxxx series***
- ***Microcontrollers GigaDevice: GD32A1xx, GD32A5xx, GD32C1xx, GD32E1xx, GD32E2xx, GD32E5xx, GD32F1xx, GD32F2xx, GD32F3xx, GD32F4xx***
- ***Microcontrollers GreenPeak: GPxxx series***
- ***Microcontrollers Infineon(Siemens): XC800, C500, XC166, C166, XC2000, XE164, TC2xx, TC3xx, TC17xx series***
- ***Microcontrollers ITE Tech. Inc.: IT5xxx, IT6xxx, IT8xxxx***
- ***Microcontrollers MDT 1xxx and 2xxx series***
- ***Microcontrollers Megawin: MG87xxx, MPC82xxx series***
- ***Microcontrollers Microchip PICmicro: PIC10xxx, PIC12xxx, PIC16xxx, PIC17Cxxx, PIC18xxx, PIC24xxx, dsPIC, PIC32xxx series***
- ***Microcontrollers Microchip AVR 8bit: AVR DA, DB, DD, DU, EA, EB, SD series***
- ***Microcontrollers Microchip ARM Cortex-M0: PIC32CM-MC, PIC32CM-JH, PIC32CM-GV series***
- ***Microcontrollers Microchip ARM Cortex-M23: PIC32CM-LE series***
- ***Microcontrollers Microchip ARM Cortex-M4: PIC32CX series***
- ***Microcontrollers MindMotion MM32F0020xxx, MM32F031xxx***

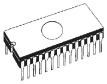


- ***Microcontrollers Motorola/Freescale/NXP: HC05, HC08, HC11, HC12, HCS08, RS08, S12, S12X, MC56F, MCF51, MCF52 series, Kinetis (K, KL, KE, KV, KM, KEA), Qorivva/Power Architecture xPC5xxx, S32K, MWCTxxxx***
- ***Microcontrollers Myson Technology (Myson Century): MTV2xx, 3xx, 4xx, 5xx, CS89xx series***
- ***Microcontrollers National: COP8xxx series***
- ***Microcontrollers NEC: uPD70Fxxx, uPD78Fxxx series***
- ***Microcontrollers Novatek: NT68xxx series***
- ***Microcontrollers Nordic Semiconductor: nRF24LExxx, nRF24LUxxx, nRF315xx, nRF51xxx, nRF52xxx, nRF54xxx, nRF91xxx Flash and OTP series***
- ***Microcontrollers Nuvoton ARM Cortex-Mx: NUC1xx, NUC2xx, M0A2xxx, M03x, M05x, M25x, M26x, M45x, M48x, Mini51, NM1xxx, NANO1xx, NUC029xxx, NDA1xx, N5xx, NPCXxxx series***
- ***Microcontrollers Nuvoton (Winbond): MS51xxx, ML51xxx, N76xxx, N78xxx, N79xxx, W77xxx, W78xxx, W79xxx, W83xxx series***
- ***Microcontrollers Nuvoton: KM101, KM103, MN1M7 series***
- ***Microcontrollers NXP (Philips) ARM Cortex-Mx: LPC11xx, LPC11xxLV, LPC11Axx, LPC11Cxx, LPC11Dxx, LPC11Exx, LPC11Uxx, LPC12xx, LPC12Dxx, LPC13xx, LPC15xx, LPC17xx, LPC18xx, LPC40xx, LPC43xx, LPC8xx, EM7xx, LPC51xx, LPC54xxx, LPC55xx, MCXAxxx series***
- ***Microcontrollers NXP (Philips) UOC series: UOCIII, UOC-TOP, UOC-Fighter (TDA1xxxx), PCF79xxx series***
- ***Microcontrollers NXP (Philips) ARM7: LPC2xxx, MPT6xx, PCD807xx, SAF7780xxx series***
- ***Microcontrollers NXP (Philips) ARM9: LPC31xx series***
- ***Microcontrollers Panasonic: MN101, MN103, MN1M7 series***
- ***Microcontrollers Pasat: TinyModule DIL40, DIL50 series***
- ***Microcontrollers Puya PY32F0xx, PY32F4xx***
- ***Microcontrollers Realtek: RTL875x, RTL876x, RTL872x, RTS54xx***
- ***Microcontrollers Scenix (Ubicom): SXxxx series***
- ***Microcontrollers Syntek: STK6xxx series***
- ***Microcontrollers Renesas: Microcontrollers Renesas: R8C/Tiny series, RX series, uPD70Fxxx, uPD78Fxxx series, RL78 series, R32C series, H8 series, H8S series, H8 Tiny series, SH series, M16C series, Synergy series, RA series***
- ***Microcontrollers SyncMOS: SM39xxx, SM59xxx, SM73xxx, SM79xxx, SM89xxx series***
- ***Microcontrollers & Programmable System Memory STMicroelectronics: uPSD, PSD series***
- ***Microcontrollers STM: STM32 SFI (secure firmware install): STM32L562RET [LQFP64](SFI), STM32U575OIYxQ [WLCSP90](SFI)***
- ***Microcontrollers STM (ex SGS-Thomson): BlueNRG, ST6xx, ST7xx, ST10xx, STR7xx, STR9xx, STM32C/F/G/H/L/U/W, STM8A/S/L series, SPC5 (Power Architecture), STM32F0xx, STM32F1xx, STM32F2xx, STM32F3xx, STM32F4xx, STM32F7xx, STM32L0xx, STM32L1xx, STM32L4xx, STM32W108***
- ***Microcontrollers Silicon Laboratories(Cygnal): C8051 series***
- ***Microcontrollers Silicon Laboratories(Energy Micro): EFM32PGxx, EFM32ZGxx, EFR32BGxx, EFR32FGxx, EFR32MGxx, EFR32ZGxx, EFM8xx, EFM32Gxx, EFM32GGxx, EFM32LGxx, EFM32TGxx, EFM32WGxx series***
- ***Microcontrollers Silicon Laboratories: SiM3Cxxx, SiM3Lxxx, SiM3Uxxx series***
- ***Microcontrollers Telink Semicond.: TLSR82xxx, TLSR83xxx, TLSR92xxx***

- ***Microcontrollers TI (Chipcon): CC11xx, CC13xx, CC24xx, CC25xx, CC26xx, CC85xx series***
- ***Microcontrollers Texas Instruments: MSP430, MSC12xx, MSPM0Cxxxx, MSPM0Gxxxx, MSPM0Lxxxx, RM4xLxxx, TMS320F, TMS570LSxxxx, CC430 series.***
- ***Microcontrollers Texas Instruments (ex Luminary Micro): LM3Sxxx, LM3Sxxxx series, LM4Fxxxx series, TM4C series***
- ***Microcontrollers Toshiba: TX00, TX03, TX04, TXZ3, TXZ3+, TXZ4 series***
- ***Microcontrollers ZILOG: Z86/Z89xxx and Z8Fxxxx, Z8FMCxxxxx, Z16Fxxxx, Z16FMCxx, ZGP323xxxxxx, ZLF645xxxxxxx, ZLP12840xxxxx, ZLP323xxxxxxx series***
- ***Microcontrollers other: EM Microelectronic, Spansion(Fujitsu), Goal Semiconductor, Hitachi, Holtek, Novatek, Macronix, Princeton, Winbond, Samsung, Toshiba, Mitsubishi, Realtek, M-Square, ASP, Coreriver, Gencore, EXODUS Microelectronic, Topro, TinyARM, VersaChips, SunplusIT, M-Square, NSING (Nations Technologies), QIXIN, Signetic, Tekmos, Weltrend, Amic, Cyrod Technologies, Ember, Ramtron, Nordic Semiconductor, Samsung, ABOV Semiconductor, Iflytek, Telink Semiconductor, Lapis, Qorvo, Luminary Micro, Energic Technology, Trident Microsystem, Fresco Logic, Parade, Ene Technology Inc, HangZhou Nano IC, ETA CHIPS, Melexis, Yuntu ...***
- ***EPROM: 2708 *1***
- ***PLD: Bipolar PALxxx *2***

***Programmer, through ISP connector***

- Serial EEPROM: IIC series, MW series, SPI series, KEELOQ series, PLD configuration memories, UN I/O series
- 1-Wire EEPROM: DS1xxx, DS2xxx
- Serial NOR Flash: standard SPI (25xxx), DataFlash (AT45Dxxx, AT26Dxxx)
- Configuration (EE)PROM: AT17xxx, AT18Fxxx, EPCxxx, EPCSxxx, EPCQxxx, XCFxxx, XC18Vxxx
- Serial FRAM: Cypress(Ramtron): FM24xxxxxx, FM25xxxxxx, Fujitsu: MB85RCxxxx, MB85RSxxxx, Lapis(OKI, Rohm): MR44xxxxx, MR45xxxxx
- Microcontrollers Atmel/Microchip: AT89Cxxx, AT89Sxxx, AT89LSxxx, AT89LPxxx, AT90pwm, AT90can, AT90usb, AT90Sxxxx, ATtiny, ATmega, ATxmega series
- Microcontrollers Atmel/Microchip AVR32: AT32UC3xxxx, ATUCxxxD3/D4/L3U/L4U series
- Microcontrollers Atmel ARM7: AT91SAM7Sxx, AT91SAM7Xxx, AT91SAM7XCxx, AT91SAM7SExx series;
- Microcontrollers Cypress: CY8C2xxxx
- Microcontrollers Elan: EM78Pxxx, EM6xxx series
- Microcontrollers EM Microelectronic: 4 and 8 bit series
- Microcontrollers Microchip PICmicro: PIC10xxx, PIC12xxx, PIC16xxx, PIC17xxx, PIC18xxx, PIC24xxx, dsPIC, PIC32xxx series
- Microcontrollers Mitsubishi: M16C
- Microcontrollers Motorola/Freescale: HC08 (both 5-wire, All-wire), HC11, HC12, HCS08, S12, S12X, MC56F, MCF52, Kinetis K series
- Microcontrollers Nordic Semiconductor: nRF24LExxx, nRF24LUxxx, nRF315xx, nRF5xxxx Flash and OTP series
- Microcontrollers Nuvoton ARM Cortex-Mx: NUC1xx, NUC2xx, M0A2xxx, M03x, M05x, M25x, M26x, M45x, M48x, Mini51, NM1xxx, NANO1xx, NDA1xx series
- Microcontrollers Nuvoton (Winbond): MS51xxx, ML51xxx series
- Microcontrollers NXP (Philips) ARM7: LPC2xxx, MPT6xx series



- Microcontrollers NXP (Philips) ARM Cortex-Mx: LPC11xx, LPC11xxLV, LPC11Axx, LPC11Cxx, LPC11Dxx, LPC11Exx, LPC11xxLV, LPC11Uxx, LPC12xx, LPC12Dxx, LPC13xx, LPC17xx, LPC18xx, LPC40xx, LPC43xx, EM7xx, LPC8xx series
- Microcontrollers NEC: uPD7xxx series
- Microcontrollers Philips (NXP): LPCxx series, 89xxx series
- Microcontrollers Renesas: R8C/Tiny series, uPD7xxx series, RL78 series, RH850 series
- Microcontrollers Realtek, M-Square
- Microcontrollers Samsung: ICPZBSxxx series
- Microcontrollers Scenix (Ubicom): SXxxx series
- Microcontrollers Silicon Laboratories(Energy Micro): EFM32Gxx, EFM32GGxx, EFM32LGxx, EFM32TGxx, EFM32WGxx series
- Microcontrollers Silicon Laboratories: EFM32PGxx, EFM32ZGxx, EFR32BGxx, EFR32FGxx, EFR32MGxx, EFR32ZGxx, EFM8xx, SiM3Cxxx, SiM3Lxxx, SiM3Uxxx series
- Microcontrollers STM (ex SGS-Thomson): ST6xx, ST7xx, ST10xx, STR7xx, STR9xx, STM32F/L/W, STM8A/S/L series, SPC5 (Power Architecture)
- Microcontrollers Silicon Laboratories(Cygnal): C8051 series
- Microcontrollers & Programmable System Memory STMicroelectronics: uPSD, PSD series
- Microcontrollers TI (Chipcon): CC11xx, CC13xx, CC24xx, CC25xx, CC26xx, CC85xx series
- Microcontrollers TI: MSP430 series (both JTAG and BSL), MSC12xxx series, CC430 series, LM4F series, TM4C series, MSPM0Cxxxx, MSPM0Gxxxx, MSPM0Lxxxx series
- Microcontrollers ZILOG: Z8Fxxxx, Z8FMCxxxxx, Z16Fxxxx series, ZLF645x0xx
- Microcontrollers other: Realtek, M-Square, Luminary Micro, Qorvo
- Various PLD (also by Jam/VME/SVF/STAPL/... Player/JTAG support):
- Altera: MAX 3000A, MAX 7000A, MAX 7000B, MAX 7000S, MAX 9000, MAX II/G/Z, MAX V
- Xilinx: XC9500, XC9500XL, XC9500XV, CoolRunner XPLA3, CoolRunner-II
- PLD Lattice: ispGAL22xV10x, ispLSI1xxxEA, ispLSI2xxxE, ispLSI2xxxV, ispLSI2xxxVE, ispLSI2xxxVL, M4-xx/xx, M4LV-xx/xx, M4A3-xx/xx, M4A5-xx/xx, LC4xxxB/C/V/ZC/ZE, ispCLOCK, Power Manager/II, ProcessorPM
- FPGA: Microsemi(Actel): ProASIC3, IGLOO, Fusion, ProASICplus, SmartFusion
- FPGA: Lattice: MachXO, MachXO2, LatticeXP, LatticeXP2, ispXPGA

**Notes:**

- *2708 memories (*1) are programmed by using 2708 additional module*
- *Devices marked *2 are obsolete and some of them might also require the additional PLD-1 module for programming*
- *Complicated devices with laborious implementation might belong to the 'Paid ISP support' category*
- *For all supported devices see the actual **Device list** on **www.dataman.com**.*

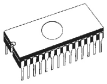
***I.C. Tester***

- TTL type: 54,74 S/LS/ALS/H/HC/HCT series
- CMOS type: 4000, 4500 series
- static RAM: 6116 ... 624000
- user definable test pattern generation

Package support

- support all devices in DIP with the default socket
- package support includes DIP, SDIP, PLCC, JLCC, SOIC, SOP, PSOP, SSOP, TSOP, TSOPII, TSSOP, QFP, PQFP, TQFP, VQFP, QFN (MLF), SON, BGA, EBGA, FBGA, VFBGA, UBGA, FTBGA, LAP, CSP, SCSP, LQFP, MQFP, HVQFN, QLP, QIP etc.

- Devices in non-DIP packages up to 48 pins are supported by universal programming adapters - if it is technically possible on the **DATAMAN-48PRO4 / DATAMAN-48PRO4C** programmer. That means if the reliability of operations when using the universal programming adapter is below the industrial standard, then specialized programming adapters are necessary.
- programmer is compatible with third-party adapters for non-DIP support



Programming speed

***Note:***

- *It is important to know, we always use a random data pattern for programming speed testing. Some our competitors use a "sparse" data pattern, where only a few non-blank data are programmed or there is used data with only a few 0 bits (FE, EF, etc.). This methodology significantly reduces the programming time. If you plan to compare, always ask which data pattern they use.*
- *The programming speed depends only slightly on the host PC speed, of course at the condition the CPU usage is below 100%.*

| Device | Size [bits] | Operation | Time |
|---|---|---|---|
| K8P6415UQB (parallel NOR Flash) | 400100hx16 bit (64 Mega) | programming and verify | 16.5 sec |
| MT29F1G08ABAEAWP (parallel NAND Flash) | 8400000Hx8 (1 Giga) | programming and verify | 46.2 sec |
| THGBM3G4D1FBAIG (eMMC NAND Flash) | 1872 MB x8 (16 Giga) | programming *1 | 339 sec |
| QB25F640S33 (serial Flash) | 800200Hx8 (64 Mega) | programming and verify | 30.9 sec |
| AT89LP51RD2 (microcontroller) | 10000Hx8 | programming and verify | 7.4 sec |
| PIC32MX360F512L (microcontroller) | 80000Hx8 | programming and verify | 10 sec |

**Conditions:** *PC type: Intel Core i7-12700, 2.1GHz, 32GB RAM, USB 2.0 HS, Win 11, software PG4UW v4.11.*

*1 *Implementation is the same as in card readers. Verification of programming is performed by the internal controller. The internal controller confirms the proper programming using a status register.*

SOFTWARE

- **Algorithms:** only manufacturer approved or certified algorithms are used. Custom algorithms are available at an additional fee.
- **Algorithm updates:** software updates are available regularly, approx. every 4 weeks, free of charge. **OnDemand** versions of software are available for urgent chip support and/or bug fixes. Available on a nearly daily basis.
- **Main features:** revision history, session logging, on-line help, device and algorithm information

Device operations

- **standard:**
  - intelligent device selection by device type, manufacturer or fragment of the part name
  - automatic ID-based selection of certain EPROM/Flash EPROM devices.
  - blank check, read, verify
  - program
  - erase
  - configuration and security bit programming
  - illegal bit test
  - checksum
  - interpret the Jam Standard Test and Programming Language (STAPL), JEDEC standard JESD-71
  - interpret the VME files, compressed binary variation of SVF files
  - interpret the SVF files (Serial Vector Format)
  - interpret the Actel STAPL Player files

- **security**
  - insertion test
  - contact check
  - ID byte check
- **special**
  - production mode (automatically start immediately after device insertion)
  - multi-project mode
  - lot of serialization modes (several incremental modes, from-file mode, custom generator mode)
  - statistics
  - count-down mode

Buffer operations

- view/edit, find/replace
- fill/copy, move, byte swap, word/dword split
- checksum (byte, word)
- print

File load/save

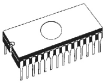
- no download time because the programmer is PC controlled
- automatic file type identification

Supported file formats

- unformatted (raw) binary
- HEX: Intel, Intel EXT, Motorola S-record, MOS, Exormax, Tektronix, ASCII-SPACE-HEX,, ASCII HEX, Renesas Consolidated HEX
- Altera POF, JEDEC (ver. 3.0.A), e.g. from ABEL, CUPL, PALASM, TANGO PLD, OrCAD PLD, PLD Designer ISDATA, etc.
- JAM (JEDEC STAPL Format), JBC (Jam STAPL Byte Code), STAPL (STAPL File) JEDEC standard JESD-71
- VME (ispVME file VME2.0/VME3.0)
- SVF (Serial Vector Format revision E)
- STP (Actel STAPL file)

GENERAL

- external power supply unit: operating voltage 100-240V AC rated, 90-264V AC max., 47-63Hz. Output: 15V DC.
- operating voltage 15-20V DC, max. 1.6A
- power consumption max. 24W active, about 3W sleep
- dimensions 146x100x64 mm (5.7x3.9x2.5 inch)
- weight 0.6kg (1.32 lb)
- operating temperature 5°C ÷ 40°C (41°F ÷ 104°F)
- operating humidity 20%...80%, non condensing



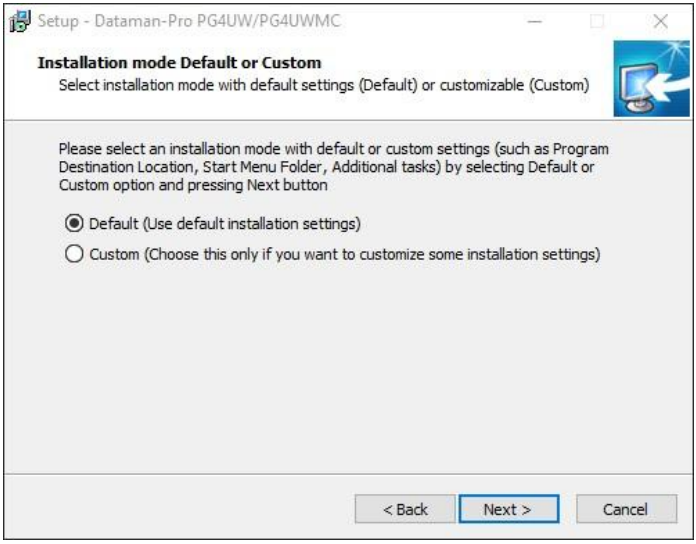
# Setup

# *Software setup*

**Step 1.**
Download the latest version of the software for your programmer from **www.dataman.com/pages/downloads**.
Copy dataman_control_software.exe to a temporary directory, disconnect programmer from PC and then launch it.
**Step 2.**

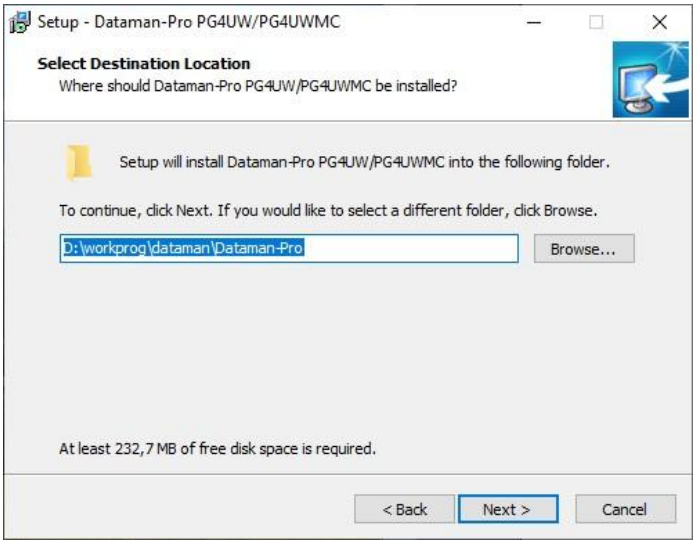

Click on “Next” button

**Step 3.**



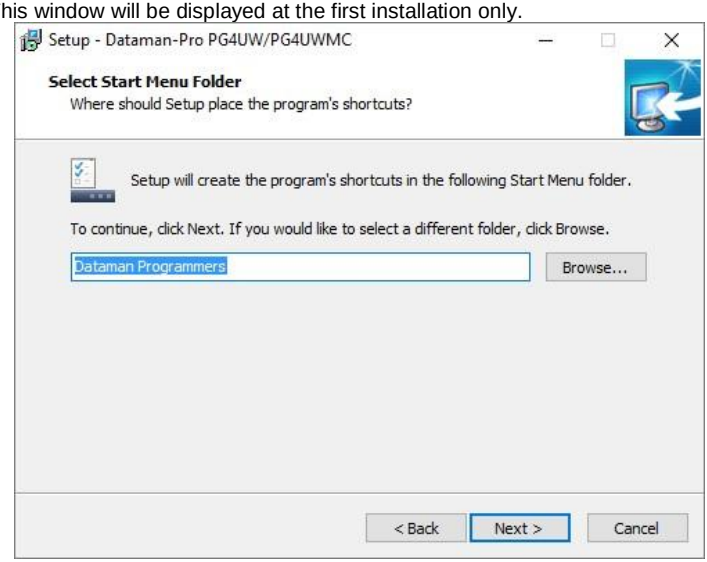

For default settings click on the “Next” button. Setup will continue with Step 5. To change default settings you can click on “Custom” and then “Next” button.

**Step 4.**



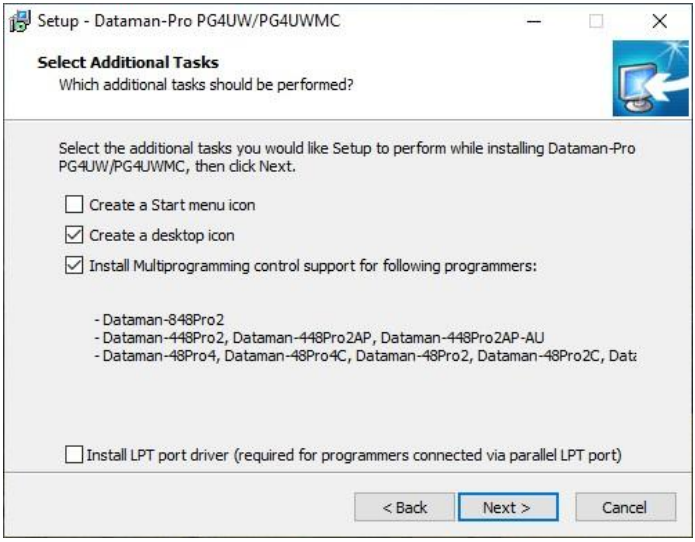

To change the default folder click “Browse” button, select the destination folder. Then click “Next” button

**Step 5.** This window will be displayed at the first installation only.



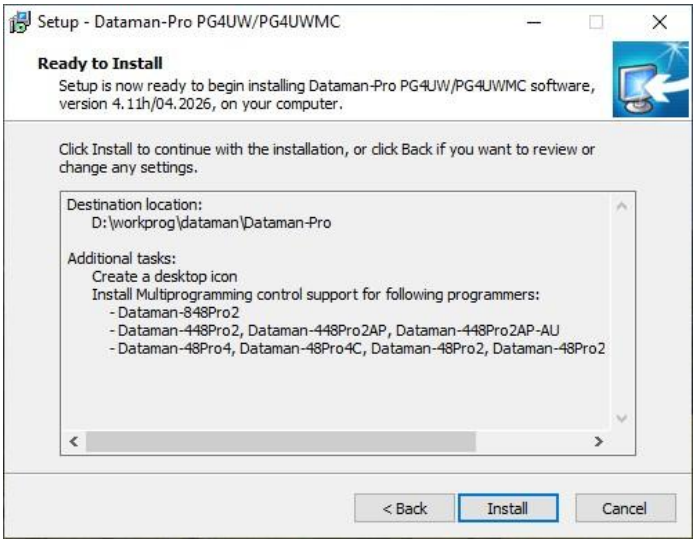

To change the default folder, click on “Browse” button, select the destination folder. Then click “Next” button

**Step 6.**



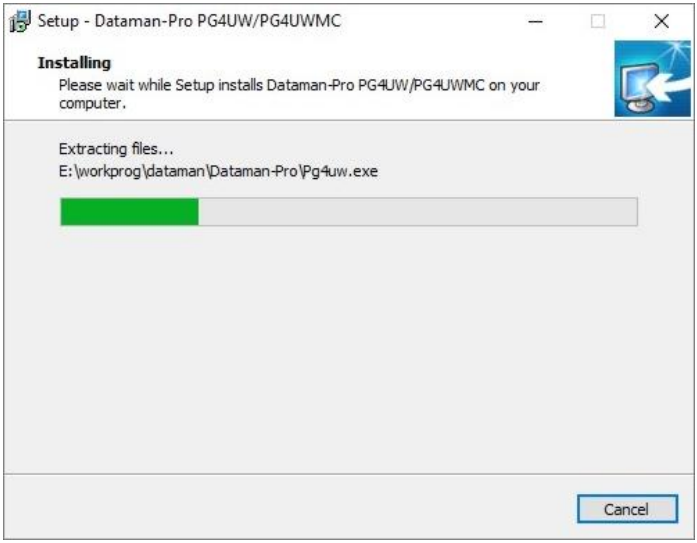

Check if “Install Multiprogramming control support” is selected. Change the default setting, if you want. Then click “Next” button

**Step 7.**




Check your settings and then click “Install” button

**Step 8.**



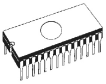

Installation process will start.

**Step 9.**


Click “Finish” button to finish setup.

New versions of programmer software

In order to exploit all the capabilities of your programmer, we recommend using the latest version of PG4UW. You may download the latest version of the programmer software (file dataman_control_software.exe) from our website **www.dataman.com/pages/downloads**.
Copy dataman_control_software.exe to a temporary directory, disconnect the programmer from the PC and then launch it. Setup will start with **Step 2** from the previous chapter.



# *Hardware setup*

**Warning:** *In general, we recommend to connect the programmer(s) directly to the PC's USB ports (without a USB HUB) and preferably to the USB ports mounted directly on the motherboard (usually located on the back of the PC).*

**Step 1.**
Directly connect the USB cable to the type B USB port on the programmer.

**Step 2.**
Directly connect the USB cable to the type A USB port on the PC (high-speed port recommended).

**Step 3.**
Connect the power supply cable to the appropriate connectors on programmer and wall.

**Step 4.**
Turn on the programmer. At this point all the 'work result' LEDs light up successively and then the LEDs switch off.

# PG4UW

# *PG4UW-programmer software*

The PG4UW program is the common control program for all DATAMAN programmers, it works with MS Windows 10 and Windows 11, 32-bit and 64-bit.

Using the programmer software

***Run the control program***

In the Windows environment: double click the desktop icon PG4UW. 

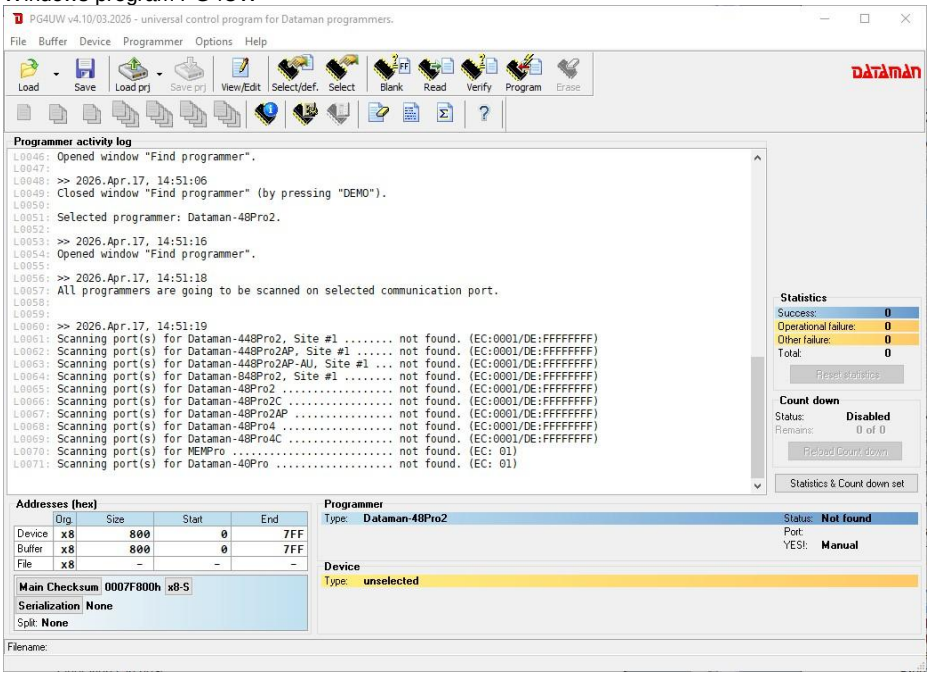
After starting, the control program PG4UW will automatically scan all existing ports and search for any connected DATAMAN programmer. The PG4UW program is common for all of the DATAMAN programmers, hence the program will try to find all supported programmers.

**Note:** *When PG4UW is started, the program is checked for its integrity. Then the program display the standard user menu and waits for your instructions.*

If the control program cannot communicate with the programmer, an error message appears on the screen including an error code and description of the possible reasons (disconnected programmer, bad connection, power supply failure...). Eliminate the source of the error and press any key. If the error condition still exists, the program resumes its operation in DEMO mode and access to the programmer is not possible. If you cannot find the cause of the error, follow the instructions in the **Troubleshooting** section. In addition, the control program checks communication with the programmer prior to any operation with the programmed device.

Description of the user screen

Windows program PG4UW




**Toolbars**

Under the main menu toolbars are placed with buttons for shortcuts to frequently used menu commands. Toolbars are optional and can be turned off through the menu command **Options / View**.

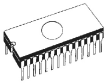
**Log window**

Log window contains the flow-control progress information about almost every operation made in PG4UW.

An operation can be:

- starting of PG4UW
- programmer search
- file/project load/save
- selection of a device
- device operations (device read, blank check, programming, ...)
- remote control application connection and disconnection
- and others

The content of the Log window can be saved to a file concurrently while information is being written to the Log window. This option can be set in the menu **Options / General options** (tab, *Log file* in the *General options* dialog).



**Addresses Panel**
Addresses Panel contains information about the actual address ranges of the currently selected device, loaded file and buffer start-end address settings. Some devices allow modifying the default device and buffer address ranges using the menu command **Device / Device options / Operation options**.
Addresses Panel also contains some advanced information about the current status of the Split, Serialization and buffer checksum features. For more information about each of the options, please see:

- Split - menu **Device / Device options / Operation options**
- Serialization - menu **Device / Device options / Serialization**
- Checksum - menu **Buffer / Checksum** section: Checksum displayed in main window

**Programmer Panel**
Programmer Panel contains information about the currently selected programmer.
The information includes

- programmer type
- port via which the programmer is connected to the computer
- programmer status, can be one of the following:
  - Ready – programmer is successfully found, connected and ready to work
  - Not found - programmer is not found
  - Demo - when the user selects the option (button) DEMO in the Find programmer dialog
- YES! mode - some types of programmer allow the use of special modes for starting the next device operation in one of the following ways:
  - manually by the control program dialog Repeat
  - manually by the YES! button, placed directly on the programmer
  - automatically – the programmer automatically detects device removal and insertion of a new device

For more details please see the **Programmer / Automatic YES!** chapter.

**Device Panel**
It contains information about the currently selected device.
The information includes:

- device name (type) and manufacturer
- device adapter needed for use with the currently selected programmer
- reference to a detailed *Device info* dialog, available also through menu **Device / Device info**
- reference to *Advanced device options* - this is only available for some types of devices

**Statistics Panel**
Contains statistical information about the currently selected device.
The information includes:

- number of successful, failed and total device operations
- count-down status indicating number of remaining devices

Statistics and count-down options are available through menu command **Device / Device options / Statistics** or by right clicking on the *Statistics* panel and selecting Statistics from popup menu.

**File Panel**
The panel is placed at the bottom of the PG4UW main window. The panel shows the currently loaded file or project name, size and date.

List of hot keys

| | | |
|---|---|---|
| **<F1>** | Help | Calls Help |
| **<F2>** | Save | Save file |
| **<F3>** | Load | Load a file into the buffer |
| **<F4>** | Edit | Viewing/editing of the buffer |
| **<F5>** | Select/default | Target-device selection from the 10 last selected devices |
| **<Alt+F5>** | Select/manual | Target-device selection by typing device/vendor name |
| **<F6>** | Blank | Blank check |
| **<F7>** | Read | Reads the device's content into the buffer |
| **<F8>** | Verify | Compares contents of the target device with the buffer |
| **<F9>** | Program | Programs target device |
| **<Alt+Q>** | Exit without save | Terminates the PG4UW |
| **<Alt+X>** | Exit and save | Terminates the PG4UW and saves settings |
| **<Ctrl+F1>** | | Displays additional information about the current device |
| **<Ctrl+F2>** | Erase | Fill's the buffer with a specified value |
| **<Ctrl+Shift+F2>** | | Fill's the buffer with random values. |

# *File*

**File** menu is used for source file manipulation, settings and viewing directory, changing drives, changing the start and finish address of the buffer for loading and saving files using **binary**, **MOTOROLA**, **MOS Technology**, **Intel (extended) HEX**, **Tektronix, ASCII space**, **JEDEC**, and **POF** formats. The menu commands for loading and saving projects are located in this submenu too.

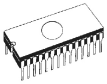
File / Load

Analyses the file format and loads the data from the specified file into the buffer. You can choose the desired format (**binary**, **MOTOROLA**, **MOS Technology**, **Tektronix**, **Intel (extended) HEX**, **ASCII space**, **ASCII HEX**, **Straight HEX**, **JEDEC** and **POF**). The control program stores the last valid mask for file listing. You can save the mask into the configuration file using the command **Options / Save options**.

The reserved key **<F3>** will bring up this menu from any menu and at any time.

***File format descriptions:***

**ASCII HEX format**
Each data byte is represented as 2 hexadecimal characters, and is separated by white space from all other data bytes. The address for the data bytes is set by using a sequence of $Annnn, characters, where nnnn is the 4-hex characters of the address. The comma is required. Although each data byte has an address, most are implied. Data bytes are addressed sequentially unless an explicit address is included in the data stream. Implicitly, the file starts at address 0 if no address is set before the first data byte. The file begins with a STX (Control-B) character (0x02) and ends with an ETX (Control-C) character (0x03).
**Note:** *The checksum field consists of 4 hex characters between the $S and comma characters. The checksum immediately follows an end code.*



Here is an example of ASCII HEX file. It contains the data "Hello, World" to be loaded at address 0x1000:

^B $A1000,
48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 0A ^C
$S0452,

**ASCII SPACE format**
A very simple hex file format similar to ASCII HEX without the checksum field, without start (STX) and end (ETX) characters. Each data byte is represented as 2 hexadecimal characters, and is separated by white space from all other data bytes. The address field is separated by white space from the data bytes. The address is set by using a sequence of 4-8 hex characters.

Here is an example of ASCII SPACE file. It contains the data "Hello, World" to be loaded at address 0x1000:
0001000 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 0A

**Straight HEX format**
A very simple hex file format similar to ASCII HEX without the address and checksum fields, without start (STX) and end (ETX) characters. Each data byte is represented as 2 hexadecimal characters, and is separated by white space from all the other data bytes.

Here is an example of a Straight HEX file. It contains the data "Hello, World":
48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 0A

**Samsung HEX format**
Samsung HEX file format is a slight modification of the Intel HEX format, therefore in the software it is recognized and indicated as Intel HEX file format.

**Notes for special x16 formats:**
- *Intel HEXx16 is an Intel Hex file format with a 16 bit data word for TMS320F devices.*
- *Motorola HEXx16 is a Motorola file format with a 16 bit data word for TMS320F devices.*

Checking the **Automatic file format recognition** check box tells the program to detect the file format automatically. When the program can't detect the file format from one of the supported formats, binary file format is assumed.

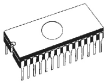
When the **Automatic file format recognition** check box is unchecked, the program allows the user to manually select the desired file format from a list of available file formats on the **Selected file format** panel. Default is set from **Options / General options** in the panel **Load file format** on the tab **File options**.

**Attention:** The p*rogram doesn't recognize files in ASCII Hex format automatically, it recognizes them as binary. So, download files in ASCII Hex format with the option for automatic file format recognition disabled.*

**Additional operation Panel**

Checking the check box **Erase buffer before loading** tells the program to erase all data from the recently selected buffer. There are two options to specify erase values:

- **Autodetect** buffer will be erased (filled) with the default "blank" value for the recently selected device and buffer. Recent default value is displayed after "Autodetect" text in brackets, for example: "Autodetect (FFh)", "Autodetect (00h)"
- **Custom defined** the buffer will be erased (filled) with user specified Byte value



Buffer erase is performed immediately before reading any file content to the buffer and it is functional for binary and all HEX file formats. Using this one-shot setting disables the current setting of **Erase buffer before loading** option in the menu **Options / General options** on tab **Hex file options**.
If the checkbox **Swap bytes** is displayed, the user can activate the function of swapping bytes within 16bit words (or 2-byte words) during reading of the file. This feature is useful especially when loading files with Motorola representation of byte order in file (big endian). The standard load file uses the little endian byte order.

**Note:** *Big-endian and little-endian are terms that describe the order in which a sequence of bytes are stored in the byte-wide file or byte-wide computer memory. Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first. For example, in a big-endian computer, the two bytes required for the hexadecimal number 4F52 would be stored as 4F52H in storage address 1000H as: 4FH is stored at storage address 1000H, and 52H will be at address 1001H. In a little-endian system, it would be stored as 524FH (52H at address 1000H, and 4FH at address 1001H).*
*How the value 4F52H is stored in memory:*

| *Address* | *Big endian system* | *Little endian system* |
|---|---|---|
| *1000H* | *4FH* | *52H* |
| *1001H* | *52H* | *4FH* |

The **View/Edit buffer** in 16-bit mode shows the real view of the word. For example the 1001 0000 1111 0110 binary is shown a 90F6 hex - D15=1 and D0=0.

**Add blank spare area** - (for NAND Flash devices) if checked, adds blank spare area data during file loading to the relevant position in the buffer (dependent on the selected device).

**Buffer offset for loading Panel**

**Buffer offset for loading** panel contains a one-shot offset setting for loading data from the file to the buffer. The setting is used to specify an optional offset of the data loaded into the buffer. When the Load file dialog window is opened, the offset always has the default setting, None. It means no offset is used to store the read data in the buffer.

Available offset options are:

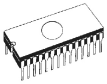
**None** this setting means that no offset is applied when loading data from a file to the buffer.

**Positive offset** set of offset value, which is added to current address to store data to buffer. This offset is available for all formats and is used in x8 format, if current buffer organization is x8, or in x16 format, if current buffer organization is x16.

**Negative offset** has two options:

**Negative offset** and **Automatic negative offset** - set in two ways: manual or automatic.
For manual setting use the option Negative offset and put the desired offset value in the edit box.
For automatic offset detection use the option Automatic negative offset. This value is subtracted from the current address to save data into the buffer.

Negative offset value (manually defined or automatically detected) is subtracted from the current buffer address to store the data into the buffer.
Negative offset is only applied for all HEX file formats and always uses x8 format. Negative offset settings are ignored for binary files and other non-HEX files.



**Notes**:

- *Since the value of the negative offset is subtracted from the real address, the result of subtraction can be negative number. Therefore, make sure you set this value correctly!*
- *We recommend automatic setting of the negative offset only in special cases. This option contains a heuristic analysis, which can treat some data in the file incorrectly. There are especially critical files which contain a fragmented addresses range and which exceeds the size of the selected device - some blocks can be ignored.*
- *Automatic negative offset option is not available for some kinds of special devices which require HEX files with exactly specified blocks - for example Microchip PICmicro devices. For these special devices, there are only manual offset settings available (None, Positive offset, Negative offset)*.

**Example of negative offset use:**
A file contains data in Motorola S - format.
A data block started at address FFFF0H.
It is an S2 format with a length of address array of 3 bytes.
For all data read you can set a Negative offset option and the value of negative offset to FFFF0H.

It means that the offset will be subtracted from the current real addresses and the data will be written from buffer address 0.

**List of file format codes and error codes**
There are some errors which can occur during file loading in certain supported file formats. Any error is written to the LOG window in the format "Warning: error #xxy in line rrr", xx is the file format code, y is the error code and rrr is the line number in decimal.

File format codes:
#00y - binary
#10y - ASCII Space
#20y - Tektronix
#30y - Extended Tektronix
#40y - Motorola
#50y - MOS Technology
#60y - Intel HEX

Load file error codes:
#xx1 - bad first character - header
#xx2 - bad character in current line
#xx3 - bad CRC
#xx4 - bad read address
#xx5 - bad length of current line
#xx6 - too big negative offset
#xx7 - address is out of buffer range
#xx8 - bad type of selected file format
#xx9 - the file wasn't loaded all

File / Save

This command saves data in the buffer which has been created, modified or read from a device onto a specified disk drive. The file format of the saved file can be chosen from the supported formats list box. You can also enter the Buffer start and Buffer end addresses which

exactly specify the part of the buffer to save to the file. Supported file formats now are **binary**, **MOTOROLA**, **MOS Technology**, **Tektronix**, **Intel (extended) HEX**, **ASCII space**, **JEDEC** and **POF**.

If the checkbox **Swap bytes** is displayed, the user can activate the function of swapping bytes within 16bit words (or 2-byte words) during writing to the file. This feature is useful especially when saving files with Motorola representation of the byte order in the file (big endian). Standard save file operation uses the little endian byte order.

The reserved key **<F2>** will bring up this menu from any menu and at any time.

File / Load project

This option is used for loading a project file which contains device configuration, buffer data and user interface configuration.

The standard dialog **Load project** contains an additional window - **Project description** - placed at the bottom of the dialog. This window is for displaying information about the currently selected project file.
Project information consists of:

- manufacturer and name of the first device selected in the project
- date and time of project creation
- user written description of the project (it can be arbitrary text, usually author of the project and some notes)

**Note:** *For projects with serialization turned on.*

*Serialization is read from the project file using the following procedure:*

- *Serialization settings from the project are accepted*
- *Additional serialization file search is performed. If the file is found it will be read and the serialization settings from the additional file will be accepted. The additional serialization file is always associated with the specific project file. When the additional serialization file settings are accepted the project serialization settings are ignored.*

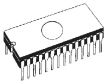
*The name of the additional serialization file is derived from the project file name by adding the extension ".sn" to project file's name.*
*The additional serialization file is always placed in the directory "serialization\" in the control program's directory.*

**Example:**

*Project file name: my_work.prj*
*Control program's directory: c:\Program Files\Programmer*

*The additional serialization file will be:*
*c:\Program Files\Programmer\serialization\my_work.prj.sn*

*The additional serialization file is created and refreshed after a successful device program operation. The only requirement for creating an additional serialization file is loading the project with serialization turned on.*



*Command **File / Save project** deletes the additional serialization file, if the file exists, and is associated with the currently saved project.*

**Enter job identification dialog**

The dialog will be shown when loading protected project files.

It contains two editable fields:

- Operator identification — this parameter will be used to identify the programmer's operator. The operator ID must be at least 3 chars. The user has to enter the Operator identification value, because it is a mandatory parameter when creating a Job Report for a protected project.
- Enter Job ID — identification of the current job.

**Note:** *The dialog Enter job identification is not a password dialog. The values for Operator identification and Job ID have an informative purpose only; they will be included in the Job Report. This does not relate to the protected and/or encrypted project password.*

File / Save project

This dialog is used for saving a **project file** which contains the settings for the device configuration and the buffer data. Data saved to a project file can be restored anytime by using the menu command **File / Load project**.

**Description of the selected project in the file list box**

Displays information about the existing project file currently selected in the Save project dialog. This box is only for information and is not editable.

**Description of project being saved**

Upper half displays information about the actual program configuration including the currently selected device, program mode, date and time, etc., and is not editable. These actual program settings are used for the creation of the project description header.

Bottom half is user editable and contains the project description (arbitrary text) which usually consists of the project author and some notes.

Checkbox **Encrypt project file (with password)** is used to save a project in a special format using an encryption algorithm. This prevents loading of the project file into the software without knowledge of the password. After clicking the key button  , the password dialog appears, which is used to specify the encryption password for the project being saved.

Checkbox **Set Protected mode of software after loading of this project file** is used to save a project in a special mode called **Protected mode**. After clicking the key button  , a password dialog appears, which is used to specify the Protected mode password for the project being saved, and another security options (disable other project loading, device operations restriction) to prevent operator mistakes. Projects saved with active **Protected mode** are special projects called **Protected mode projects**. For more detailed information about Protected mode projects see **Options / Protected mode**. When Protected mode is active, the software indicates this with a label **Protected mode** in the top right corner of the Programmer activity log.

**Recommendation:** *passwords for **Encrypt project files (with password)** and **Set Protected mode of software after loading of this project file** should not be the same.*

Checkbox **Require project file unique ID before first programming** when active, the software asks the user to enter the correct project file unique ID, before allowing you to start the first device programming operation after loading the project. This feature is recommended as an additional check that the correct project file is being loaded. It is also recommended to use this checkbox along with the active **Protected mode**. When the request for the project file unique ID is active, the software indicates this using a (**ID)** label next to the project file name in the bottom status line of the control program main window.
**Note:** *Option Require project file unique ID before first programming is a replacement of the former Require project file checksum before first programming. The Unique ID advantage over a generic checksum is that a unique ID is calculated not just from the main device buffer data, but also from secondary buffer data used by the device and the available device settings. When the request for a project file checksum is active, the software indicates this by a (CSum) label next to project file name in the bottom status line in the control program main window. This option is no longer available in the Save project dialog, but it can be activated after loading of an older project file that has the checksum request set to ON.*

File / Reload file

Choose this option to reload a recently used file from the **Reload file** list.

When you use a file (load or save), it is automatically added to the **Reload file** list (up to 10 file names are stored in the list). Files are listed in order depending on time of use. Recently used files are listed before files used previously.

To Reload a file:

1. From the File menu, choose Reload file.
2. List of recently used files is displayed. Click the file you want to reload.

**Note:** *When reloading a file the file format which the file was previously loaded/saved as is used.*

File / Reload project

Choose this option to reload a recently used file from the **Reload project** list.

When you use a project (load or save), it is automatically added to the **Reload project** list (up to 10 project names are stored in the list). Projects are listed in order depending on time of use. Recently used files are listed before files used previously.

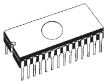
To Reload a project:

1. From the File menu, choose Reload project.
2. List of recently used projects is displayed. Click the project you want to reload.

File / Project options

This option is used to display/edit project options of the currently loaded project. Project options mean a basic description of the project including the following project data:

- device name and manufacturer
- project creation date



- user defined project description (arbitrary text), e.g. project author and other text data for a more detailed project description.

User can directly edit the user defined project description only. The device name, manufacturer, and project date are generated automatically by the program.

File / Load encryption table

This command loads the data from a binary file on the disk and it saves them into the part of the memory reserved for an encryption (security) table.

File / Save encryption table

This command writes the content of the memory reserved for an encryption table, into the file on the disk as a binary data.

File / Exit without save

The command deallocates the software heap, cancels the buffer created on disk (if it exists) and returns back to the operation system.

File / Exit and save

The command deallocates the software heap, cancels the buffer created on the disk (if it exists), saves the current setting of any recently selected devices to disk, and returns back to the operation system.

# *Buffer*

Menu **Buffer** is used for buffer manipulation, block operation, filling part of the buffer with a string, erasing, checksum calculation and of course editing and viewing with other items (find and replace string, printing...).

Buffer / View/Edit

This dialog is used to **view** (view mode) or **edit** (edit mode) data in the buffer (for viewing in DUMP mode only). Use the arrow keys to select the object to edit. Edited data is signified by its color. The data in the buffer outside of the area where data for the selected chip is located is shown using a gray background.

You can use the **<F4>** hot key also.

***View/Edit Buffer***

| | |
|---|---|
| **F1** | display help for the current window |
| **F2** | fill buffer block specified by start and end addresses with the requested hex (or ASCII) string. |
| **Ctrl+F2** | erase buffer with specified blank value |
| **Ctrl+Shift+F2** | fill buffer with random data |
| **Shift+F2** | save buffer data to a binary file. This command is available for secondary buffers only. Secondary buffers are special areas used for some devices, for example Data EEPROM for Microchip PICmicro devices. Commands for Load/Save data to/from Main buffer are available in the main menu "File" and also with the Load, Save buttons in the main application window. |
| **F3** | copy specified block of buffer data to a new address. Target address needn't be outside of the source block addresses. |

| | |
|---|---|
| **Shift+F3** | load data from a binary file to the buffer. This command is available for secondary buffers only. For more information see notes for the save buffer data command (Shift+F2) above. |
| **F4** | move block is used to move the specified block of data in the current buffer to a new address. Target address needn't be outside of the source block addresses. Source address block (or part of it) will be filled by the topical blank character. |
| **F5** | swap bytes command swaps a high- and low- order of byte pairs in the current buffer block. This block must start on even address and must have an even number of bytes. If these conditions are not met, the program modifies addresses itself (start address is moved to a lower even address and/or the end address is moved to a higher odd address). |
| **F6** | print buffer |
| **F7** | find string (max. length 16 ASCII characters) |
| **F8** | find and replace string (max. 16 ASCII chars.) |
| **F9** | change current address |
| **F10** | change mode view / edit |
| **F11** | switch the mode of the buffer data view between 8 bit and 16 bit view. It can also be done with a mouse click on the button to the right of the View/Edit mode buffer indicator. This button indicates the actual data view mode (8 bit or 16 bit), too. When 16 bit view mode is used, Bytes of data inside 16 bit Words are displayed in Little-endian order. |
| **F12** | checksum dialog allows calculation of the checksum for the selected block in the buffer, change mode view / edit |
| **Arrow keys** | move cursor up, down, right and left |
| **Home/End** | jump to start / end of current line |
| **PgUp/PgDn** | jump to previous / next page |
| **Ctrl+PgUp/PgDn** | jump to start / end of current page |
| **Ctrl+Home/End** | jump to start / end of current device |
| **Shift+Home/End** | jump to start / end of current buffer |
| **Backspace** | move cursor one position left (back) |

**Note:** *Characters 20H - FFH (mode ASCII) and numbers 0...9, A...F (mode HEX) immediately changes the content of the edited area.*
*Not all commands are available for all situations. It depends on the selected device and buffer(s) used for the device.*
**Warning:** *Editing ASCII characters for word devices is disabled.*

***Print buffer***

This command allows you to write the selected part of the buffer to a printer or to a local file. The Program uses an external text editor in which the selected block from the buffer is displayed and can again be printed or saved to a file. By default the simple text editor is **notepad.exe**, which is a standard part of all versions of Windows.

In the Print buffer dialog there are the following options:

**Block start**

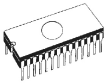
Defines the start address of the selected block in the buffer.

**Block end**

Defines the end address of the selected block in the buffer.

**External editor**

This item defines the path and name of the external program which should be used as the text viewer for the selected block of the buffer. By default the simple text editor notepad.exe is set,



which is a standard part of all versions of Windows. The user can define any text editor for example wordpad.exe, which is able to work with large text files. In the user defined text editor the user can print or save to a file the selected block of the buffer.
The external editor path and name is saved automatically to disk.

***Find dialog box***

Enter the search string to **Find** into the text input box and choose **<Find>** to begin the search or choose **<Cancel>** to exit.

**Direction** box specifies which way you want to search, starting from the current cursor position (In edit mode). **Forward** (from the current position or start of the buffer to the end of the buffer) is the default. **Backward** searches toward the beginning. In view mode it searches the whole buffer.
**Origin** specifies where the search should start.

***Find & Replace dialog box***

Enter the search string in the **Text to find** string input box and enter the replacement string in the **Replace with** input box.

In the **Options** box you can select prompt on replace: if the program finds an instance, you will be asked before the program changes it.

**Origin** specifies where the search should start.

**Direction** box specifies which way you want to search, starting from the current cursor position (In edit mode). **Forward** (from the current position or start of the buffer to the end of the buffer) is the default. **Backward** searches toward the beginning. In view mode it searches the whole buffer.

Press **<Esc>** or click the **Cancel** button to close the dialog window.

By pressing the **Replace** button the dialog box is closed and a Question window is displayed. This window contains the following choices:

| | |
|---|---|
| **Yes** | replaces the found item and finds the next instance |
| **No** | finds the next item without replacing current one |
| **Replace All** | replaces all found items |
| **Abort search** | aborts this command |

***View/Edit buffer for PLD***

| | |
|---|---|
| **Ctrl+F2** | erase buffer with specified blank value |
| **Ctrl+Shift+F2** | fill buffer with random data |
| **F9** | go to address... |
| **F10** | change mode view / edit |
| **F11** | switch the mode of the buffer data view between 1 bit and 8 bit view. This can be also done by mouse clicking on the button to the right of the View/Edit mode buffer indicator. This button indicates the actual data view mode (1 bit or 8 bit), too. |
| **Arrow keys** | move cursor up, down, right and left |
| **Home/End** | jump to start / end of current line |
| **PgUp/PgDn** | jump to previous / next page |
| **Ctrl+PgUp/PgDn** | jump to start / end current page |
| **Ctrl+Home/End** | jump to start / end of the edit area |
| **Backspace** | move cursor one position left (back) |

**Note:** *Characters 0 and 1 immediately changes the content of the edit area.*

Buffer / Fill block

Selecting this command causes filling of the selected block in the buffer using the specified hex (or ASCII) string.

Selecting the option "Allow address history logging" activates logging of the recently confirmed values. These are saved for each device separately; count is limited to the last 15 items.
**Note:** *Address history values are common for all buffer data manipulation dialogs.*

Default address ranges are set based on the selected device's buffer.
If "Maintain last inserted values" is selected, the dialogue will automatically populate with your previously confirmed settings upon reopening.

Buffer / Copy block

This command is used to copy the specified block of data in the current buffer to a new address. Target address needn't be outside of the source block addresses.

Buffer / Move block

This command is used to move the specified block of data in the current buffer to a new address. Target address needn't be outside of the source block addresses. Source address block (or part of) will be filled with the topical blank character.

Selecting the option "Allow address history logging" activates the logging of recently confirmed values. These are saved for each device separately; count is limited to the last 15 items.
**Note:** *Address history values are common for all buffer data manipulation dialogs.*

Default address range is set according to the buffer range of the selected device.
If "Maintain last inserted values" is selected, the dialogue will automatically populate with your previously confirmed settings upon reopening.

Buffer / Swap block

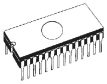
This command swaps a high- and low- order of byte pairs, foursomes, nibbles inside bytes or bits inside bytes depending on the swap mode selected by user. Swap operation is performed on the buffer block specified by the Start and End addresses. This block must start on an even address and must have an even number of bytes. If the conditions are not met, the program modifies the addresses itself (start address is moved to a lower even address and/or the end address is moved to a higher odd address).

User can select from the following swap modes:

| | |
|---|---|
| 1. Swap 2-bytes inside 16-bit words | swap byte pairs inside 16-bit words. |
| 2. Swap 4-bytes inside 32-bit words | swap byte foursomes inside 32-bit words. |
| 3. Swap nibbles inside bytes | swap high- and low- nibbles inside each byte. |
| 4. Mirror bits inside bytes | mirror bits inside each byte |

**Examples of swap operation in the buffer:**

The swap bytes operation from Start address 0 to End address N modifies data in the buffer as shown in the following tables:



| Address | Original Data | Swap 2-bytes inside 16-bit words | Swap 4-bytes inside 32-bit words | Swap nibbles inside bytes | Mirror bits inside bytes |
|---|---|---|---|---|---|
| 0000h | b0 | b1 | b3 | b0n | b0m |
| 0001h | b1 | b0 | b2 | b1n | b1m |
| 0002h | b2 | b3 | b1 | b2n | b2m |
| 0003h | b3 | b2 | b0 | b3n | b3m |
| 0004h | b4 | b5 | b7 | b4n | b4m |
| 0005h | b5 | b4 | b6 | b5n | b5m |
| 0006h | b6 | b7 | b5 | b6n | b6m |
| 0007h | b7 | b6 | b4 | b7n | b7m |

b0, b1, b2 ... means the original buffer byte values from addresses 0, 1, 2...
b0n, b1n, b2n... means the nibble-swapped original bytes b0, b1, b2... follow these rules:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Original Byte bits | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| Nibble-swapped Byte Bits | bit 3 | bit 2 | bit 1 | bit 0 | bit 7 | bit 6 | bit 5 | bit 4 |
| | | | | | | | | |
| Original Byte bits | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
| Mirrored Byte Bits | bit 0 | bit 1 | bit 2 | bit 3 | bit 4 | bit 5 | bit 6 | bit 7 |

Selecting the option **"Allow address history logging"** activates logging of the recently confirmed values. These are saved for each device separately; count is limited to the last 15 items.
**Note:** *Address history values are common for all buffer data manipulation dialogs.*

Default address range is set according to buffer range of selected device.
If "Maintain last inserted values" is selected, the dialogue will automatically populate with your previously confirmed settings upon reopening.

Buffer / Erase block

If this command is selected, the selected range of the buffer will be filled with the topical blank character.
Selecting the option "Allow address history logging" activates logging of the recently confirmed values. These are saved for each device separately; count is limited to the last 15 items.
**Note:** *Address history values are common for all buffer data manipulation dialogs.*

Default address range is set according to buffer range of selected device.
If "Maintain last inserted values" is selected, the dialogue will automatically populate with your previously confirmed settings upon reopening.

The reserved key **<Ctrl+F2>** will bring up this menu from any menu and at any time.

Buffer / Fill random data

If this command is selected, the selected range of the buffer will be filled with random data.

Selecting option "Allow address history logging" activates logging of the recently confirmed values. These are saved for each device separately; count is limited to the last 15 items.
**Note:** *Address history values are common for all buffer data manipulation dialogs.*

Default address range is set according to the buffer range of the selected device.
If "Maintain last inserted values" is selected, the dialogue will automatically populate with your previously confirmed settings upon reopening.

The reserved key **<Shift+Ctrl+F2>** will bring up this menu from any menu and at any time.

Buffer / Duplicate buffer content

This command performs a duplicate of the buffer content in the range of the source EPROM to the range of the destination EPROM. This procedure is suitable if there is, for example, 27C512 EPROM to 27C256 EPROM position.
**Note:** *The procedure always uses the buffer start address 00000h.*

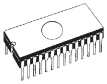
Buffer / Checksum

Checksum of the data stored in the buffer of PG4UW is useful to verify that the buffer data is correct.

PG4UW contains the following functions related to checksum:

- Tab **Checksum calculator**, this is an on-demand **checksum calculator** that can calculate and display various types of checksums of various data blocks in the buffer.
- Tab **Main checksum options** contains options for **Automatic checksum calculator** with the **Main checksum** value displayed in the main window of the PG4UW in the table **Addresses** and in the **Programmer activity log of PG4UW** as **"Data checksum of selected buffers:"**.

Tab **Checksum calculator** contains the following controls:
- Group **Buffers included to checksum calculator** contains:
  - Checkboxes to select which buffers should be used (included) for the checksum calculator. Data in the selected buffers are sequentially processed by the checksum calculation as one data stream, buffer by buffer, in the order as listed in the group **Buffers included to checksum calculator**.
  - Address range for each buffer. Addresses are always defined as Byte addresses.
  - Customizable **Excluded blocks** for each buffer. Excluded blocks can be useful for example for **serialization**. Serialization usually modifies data at specified addresses in the buffer. So there is a problem to use the checksum of the buffer when data at some addresses was changed by the serialization engine before programming each device. If the part of the buffer (data block) used for serialization is excluded from the checksum calculation, the checksum of the buffer data will not be changed by the serialization data changes. One or more excluded blocks can be specified.
- Fields displaying the values of the calculated **checksum types**: see the description of the types at the bottom.
- Column marked as **STRAIGHT** is the result of a checksum calculation without additional adjustments.
- Column marked as **NEGATED** is a negation of the checksum so that, SUM + NEG. = FFFFH.
- Column marked as **SUPPLEMENT** is a complement of the checksum so that, SUM + SUPPL. = 0 (+ carry).
- **Insert checksum options** box - this box contains the following options for the **Calculate & insert** operation:



- **Insert checksum** — Type of checksum that is written into the buffer when the Calculate & insert operation was executed.
- **Insert at address** — Address in the buffer where the result of the chosen checksum is written, when the **Calculate & insert** was executed. The address can not be specified inside the range <**From address**> to <**To address**>. The address is always defined as a Byte address.
- **Size** — Size of the chosen checksum result which will be written into the buffer. The size of the inserted checksum may be **Byte** (8-bit), **Word** (16-bit) or **DWORD** (32-bit). If the size is smaller than the selected checksum size, only the lower byte(s) of the checksum value will be written into the buffer.
  **Note:** *If Word size was selected, the low byte of the checksum value will be written to the address specified in the box **Insert address** and a high byte will be written to the address incremented by one. Similarly, for a DWORD.*

- **Calculate button** - clicking on the button Calculate starts calculating checksums for the selected block in the buffer. No writes into the buffer are executed.
- **Calculate & insert button** - clicking on the button **Calculate & insert** starts calculating checksums for the selected buffers and writes the chosen checksum into the main buffer (the first buffer in the dialog View/Edit buffer) at the address specified by the Insert address. This function is available for Byte, Word, CRC-CCITT and CRC-XMODEM checksums.
- **Close button** - closes the Checksum dialog.

Tab **Main Checksum options** contains the following options:

- Group **Buffers included to Main Checksum calculation** contains:
  - Checkboxes to select which buffers should be used (included) for the **Main Checksum** calculation. Data in the selected buffers is sequentially processed by the checksum calculation as one data stream, buffer by buffer, in the order as listed in the group **Buffers included to Main Checksum calculation**.
  - Address range for each buffer. Addresses are always defined as Byte addresses.
  - Customizable **Excluded blocks** for each buffer. Excluded blocks can be useful for example for serialization. Serialization usually modifies data at specified addresses in the buffer. So there is problem to use the checksum of the buffer, when data at some addresses was changed by the serialization engine before programming each device. If part of the buffer (data block) used for serialization is excluded from the checksum calculation, the checksum of the buffer data will not be changed by the serialization data changes. One or more excluded blocks can be specified.
- Selection group **Checksum type** allows you to select the type of checksum to be used for the **Main Checksum**. More information about Checksum types can be found at the bottom of this page.
- Field **Checksum** contains the actual value of the recently calculated checksum.
- Button **Apply** is used to confirm the checksum settings from the **Main Checksum options**. Please note, that once the button is pressed, any previous checksum settings are lost.
- Button **Close** is used to close the Checksum dialog. If you made some changes in the settings, they won't take effect until you press **Apply**.

**Note:**

- *When selecting a new device using the Select device dialog (F5), the **Checksum calculator** settings are set to the defaults, as well as the **Main Checksum** settings.*

- ***Checksum calculator*** *settings are not saved to the .ini or project file(s), they are only temporary.* ***Main Checksum*** *settings are saved to both .ini and project files.*
- *When starting PG4UW or loading a project, the initial* ***Checksum Calculator*** *settings for the selected device are not set to defaults. Instead, they are copied directly from the* ***Main Checksum*** *settings. Consequently, after the software starts or a project is loaded, the* ***Checksum Calculator*** *options will match the most recent* ***Main Checksum*** *configuration.*
- *Dialog* ***Checksum options*** *also offers useful functions such as* ***Reset to defaults****, or Copy* ***Checksum calculator*** *settings to* ***Main Checksum*** *settings, and vice versa.*

***Checksum types***

**Byte sum (x8)**

Buffer data is summed byte-by-byte irrespective of the current buffer view mode (x8/x16/x1) organization. Any carry bits exceeding 32-bits are neglected. This checksum mode is indicated by the string (x8) displayed after the checksum value in the main program window.

**Word sum Little Endian (x16)**

Buffer data is summed word-by-word irrespective of the current buffer view mode organization. Any carry bits exceeding 32-bits are neglected. This checksum mode is indicated by the string (x16 LE) displayed after the checksum value in the main program window. The term Little Endian means the buffer checksum is calculated from words read from the buffer in Little Endian mode.

**Word sum Big Endian (x16)**

Buffer data is summed word-by-word irrespective of the current buffer view mode organization. Any carry bits exceeding 32-bits are neglected. This checksum mode is indicated by the string (x16 BE) displayed after the checksum value in the main program window. The term Big Endian means that the buffer checksum is calculated from words read from the buffer in Big Endian mode.

**CRC-CCITT**

Buffer data is summed by bytes to Word using the polynomial x^16+x^12+x^5+1 (1021h), init value FFFFh, reflections in/out are off, XOR out 0

**CRC-XMODEM**

Buffer data is summed by bytes to Word using the polynomial x^16+x^12+x^5+1 (1021h), init value 0h, reflections in/out are off, XOR out 0

**CRC-16**

Buffer data is summed by bytes to sum by bytes to WORD using the standard CRC-16 algorithm with the polynomial x^16+x^15+x^2+1 (0x8005), init value 0, and XOR out 0

**CRC-32**

Buffer data is summed by bytes to DWORD using the standard CRC-32 algorithm with the polynomial:

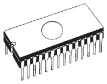
x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1 (0x04C11DB7),

initial value 0xFFFFFFFF and XOR out 0xFFFFFFFF

**MD5**

an MD5 hash expressed as a sequence of 32 hexadecimal digits (128 bits)

**SHA-1**



"Secure Hash Standard" expressed as a sequence of 40 hexadecimal digits (160 bits)

***Checksum forms***

**Straight** checksum without additional adjustments.
**Negated** negation of checksum so that, SUM + NEG. = FFFFH.
**Supplement** is a complement of the checksum so that, SUM + SUPPL. = 0 (+ carry).
**Device dependent checksum** - applies for certain devices, e.g. STMicroelectronics's STM8 family. The checksum modes for the **main checksum** can be set in the pop-up menu by clicking on the checksum label in the main program window or by using the menu shortcuts
**Shift+Ctrl+1** for Byte sum (x8),
**Shift+Ctrl+2** for Word sum Little Endian (x16)
**Shift+Ctrl+3** for Word sum Big Endian (x16) etc...
**Word** is 16-bit word.
**DWORD** is 32-bit word.

The abbreviated form of the Main Checksum forms listed in the main window of PG4UW and in the Programmer activity log are:

Byte sum (x8): x8
Word sum (x16) Little Endian: x16 LE
Word sum (x16) Big Endian: x16 BE
Straight: -S
Negated: -N
Supplement: -U

# *Device*

Menu **Device** includes functions for work with the selected programmable device - device select, read data from device, device blank check, device program, device verify and device erase.

Device / Select from default devices

This window allows you to select a device type from the list of default devices. This list functions as a cyclic buffer, storing recently selected devices and their specific options. The list is automatically saved to disk when using the **File / Exit and save** command.

To display more information about the current device, press **<Ctrl+F1>**. This command shows the device size, organization, and programming algorithm, along with a list of compatible programmers and auxiliary modules. You can also find package details and other general information here.

Press the <Del> key to remove the current device from the default devices list. Note that you cannot empty the list entirely; if you attempt to delete the final item, the command will be ignored and the last device will remain.

Device / Select device...

This window allows you to select the desired device from our database of all the devices supported by the currently selected programmer. It is possible to choose a device by **name**, by **type** or by **manufacturer**.

- Enter a **search query** into the search field to filter the device database. The query will be divided to **fragments** which will be considered as indivisible. Use the **space** character " " to separate query fragments. To include the **space** character into a **fragment**, use **double quotes** to surround this: "space containing fragment".
- Query **fragments** are compared to **device name**, **manufacturer**, **programming adapter name** and **ordering number**, search strings are **not case sensitive**.
- Fragments of the query are by default compared using **AND** logic, but there is an option to switch to **OR** logic if needed.
- Search results are assigned a **relevancy** score based on how closely they match your criteria (using AND/OR logic). By default, results are sorted by this **relevancy**.
- **Search query** field has history of the last few items which are saved with global options.
- If there are less than 10 results matching your query, the **Search suggestions** algorithm (if enabled) tries to help you find more results and offers them in a separated list as optional.

**Note 1:** *The names of the programmable devices in the software do not contain all characters shown on the top of the chip or mentioned in the related datasheet. The names contain all characters necessary for identification of the device, but do not contain codes that have no influence on the programming, for example the temperature code, speed code, packing type code, etc... If such a code (letter) is at the end of the name, it is omitted, if such a code (letter) is in the middle of the device name, then is replaced by the character 'x'.*

**Examples:**
- *Devices **Am27C512**-150, **Am27C512**-200 and **Am27C512**-250 are shown in the software only once, as **Am27C512***
- ***S29GL064N11TF1010** device is shown in the software as **S29GL064N**xx**T**xx**01***

The option **Tolerant search and "x" character replacement** switches to a less strict method of string comparison which allows you to find results that do not exactly match the query. This method also helps to cover previously mentioned part numbering issues. As a "wildcard character" you use the question mark "?".

Examples **of tolerant search queries** and **results**:

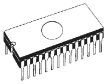
- searching **Am27C512**-150, **Am27C512**-200 and **Am27C512**-250 gives **Am27C512**
- searching **S29GL064N11TF1010** gives **S29GL064N**xx**T**xx**01**
- searching **??27C512** gives Am**27C512** and also **AT27C512**

**Note 2:** *If a certain device is listed twice and the second time with the suffix **x16**, **Dual**, **Quad**, **etc**. it means that that programming algorithm provides faster access modes.*

If you wish to display additional information about the current device, use the **Device info** button or the **<Ctrl+F1>** shortcut key.

The currently displayed device list can be saved to a text file by pressing the **Save currently displayed list to file** button.

The selected device is automatically saved to the buffer of default devices. This buffer is accessible using the **Device / Select from default devices** command.



Device / Select EPROM /Flash by ID

Use this command to autoselect an EPROM or Flash device as the active device by reading the device ID. The programmer can automatically identify certain devices by reading the manufacturer and the device-ID that are burnt into the chip. This only applies to EPROM or Flash devices that support this feature. If the device does not support the chip ID and manufacturer's ID, a message will be displayed indicating this is an unknown or not supported device.

If more than one device with an identical chip ID and manufacturer's ID are detected, the list of all these devices will be displayed. A corresponding device can be chosen from this list by selecting its part name (or manufacturer name) from the list and pressing **<Enter>** (or click the **OK** button). Press the **<Esc>** key or click the **Cancel** button at any time to cancel the device selection without affecting the currently selected device.

**Warning:** *The control program only supports at this time, EPROM's and Flash devices with 28 and 32 pins.*

*The programmer applies a high voltage to the appropriate pins on the socket. This is necessary to enable the system to read the device ID. Do not insert any device that is not an EPROM or Flash into the socket. It may be damaged when the programmer applies the high voltage!*

*We don't recommend using this function with:*

- *2764 and 27128 EPROM types, because most of them do not support ID*
- *Flash memories with non-standard pinout (e.g. Firmware Hub Flash)*
- *Flash memories which don't accept Vid voltage on the A9 pin*
- *low voltage EPROM and Flash memories*

**Note:** *The procedure is not available if a ISP programmable device is selected i.e. you are in ISP mode with the programmer. You would need to select any device which is programmable in the ZIF socket of the programmer.*

Device / Device options

All settings in this menu are used for the programming process, serialization and associated file control.

***Device / Device options / Operation options***

All settings for this command are used for the programming process control. This is a flexible environment which contains items associated with the current device and programmer type. Items which are valid for the current device but aren't supported by the current programmer, are disabled. These settings are saving to disk along with the associated device using the **File / Exit and save** command.

The commonly used terms are also explained in the user manual for the programmer. The special terms used here are exactly the terms as used by the manufacturer of the respective chip. Please read the documentation for the chip you want to program, for the full explanation of all terms used.

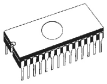
**List of commonly used items:**

group **Addresses**:

device start address (default 0)
device end address (default device size-1)
buffer start address (default 0)
**Split** (default none)

This option allows setting of a special mode for the buffer when programming or reading a device. Using the split options is particularly useful when using 8-bit data memory devices in 16-bit or 32-bit applications.

The following table describes the buffer-to-device and device-to-buffer data transfer:



| Split type | Device | Buffer Address assignment |
|---|---|---|
| None | Device [ADDR] | Buffer [ADDR] |
| Even | Device [ADDR] | Buffer [2*ADDR] |
| Odd | Device [ADDR] | Buffer [1+ (2*ADDR)] |
| 1/4 | Device [ADDR] | Buffer [4*ADDR] |
| 2/4 | Device [ADDR] | Buffer [1+(4*ADDR)] |
| 3/4 | Device [ADDR] | Buffer [2+(4*ADDR)] |
| 4/4 | Device [ADDR] | Buffer [3+(4*ADDR)] |

Real addressing will be as follows: (all addresses are hexadecimal)

| Split type | Device addresses | Buffer addresses |
|---|---|---|
| None | 00 01 02 03 04 05 | 00 01 02 03 04 05 |
| Even | 00 01 02 03 04 05 | 00 02 04 06 08 0A |
| Odd | 00 01 02 03 04 05 | 01 03 05 07 09 0B |
| 1/4 | 00 01 02 03 04 05 | 00 04 08 0C 10 14 |
| 2/4 | 00 01 02 03 04 05 | 01 05 09 0D 11 15 |
| 3/4 | 00 01 02 03 04 05 | 02 06 0A 0E 12 16 |
| 4/4 | 00 01 02 03 04 05 | 03 07 0B 0F 13 17 |

Terms explanation:
Access to device address ADDR is written as Device [ADDR].
Access to buffer address ADDR is written as Buffer [ADDR].
ADDR value can be from zero to the device size (in bytes).
All addresses are byte oriented addresses.

group **Insertion test:**

**Insertion test** (default ENABLE)
If enabled, the programmer checks all pins of the programmed device, if they have proper connection to the ZIF socket (continuity test). The programmer is able to identify poor contact, mis inserted chip and also (partially) a reversely inserted chip.
**Device ID check error terminates the operation** (default ENABLE)
Programmer performs an ID check before each selected action. It compares the read ID codes from the device with ID codes defined by the device manufacturer. In the case of an ID error, the control program behaves as follows:

- if this item is set to ENABLE, the selected action is finished
- if the item is set to DISABLE, the selected action continues. The control program just writes a warning message about the ID error to the LOG window.

If enabled, the programmer checks the electronic ID of the programmed chip.

**Note 1**: *Some old chips don't carry an electronic ID.*
**Note 2:** *In some special cases, several microcontrollers do not provide an ID if the copy protection feature in the chip is set. Even if the device ID check setting in the control program is set to "Enable".*

group **Command execution:**

| | |
|---|---|
| blank check before programming | (default DISABLE) |
| erase before programming | (default DISABLE) |
| verify after reading | (default ENABLE) |
| verify | (ONCE, TWICE) |
| verify options | (nominal VCC +/-5%<br>nominal VCC +/-10%<br>VCCmin…VCCmax) |

group **Target system power supply parameters**

This group is available in ISP mode for some types of devices. It contains the following settings:
**Enable target system power supply** - enables the supplying of the target system from the programmer. The supply voltage for the target system is switched on before action with the programmed device and is switched off after the action finished. If Keep ISP signals at defined level after operation is enabled, the programmer will switch off the supply voltage after the pull-up/pull-down resistors are deactivated.

**Voltage** - supply voltage for the target system. Supply voltage range is from 2V to 6V.
**Note:** *The voltage value given to the target system also depends on the current flowing to the target system. To reach the exact voltage supply for the target system, the proper Voltage and Max. current values must be defined. The Max. current value specified has to be, as exact as possible, equal to real current consumption of the target system.*

**Max. current** - maximum current consumption of the powered target system. Current consumption range is from 0 to 300mA

**Voltage rise time** – determines the skew rate of the rising edge of the target system power supply voltage (switch on supply voltage).

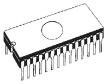
**Target supply settle time** – determines the time, after which the supply voltage in target system must stabilize at the set value for the target system to be ready for any action with programmed device.

**Voltage fall time** – determines the skew rate of the falling edge of the target system power supply voltage (switch off supply voltage).

**Power down time** - determines the time after switch off of the target system the power supply within the target system keeps a residual supply voltage (e.g. from a charged capacitor). After this time has elapsed, the target system must be without any supply voltage and can be safely disconnected from the programmer.

group **Target system parameters**

This group is available in ISP mode for some types of devices. It contains the following settings:
**Oscillator frequency** (in Hz) – the oscillator's frequency for the device (in target system). The control program sets the programming speed by this, therefore it is necessary to set the correct value.



**Supply voltage** (in mV) - supply voltage in the target system. The control program checks or sets (it depends on programmer type) the entered supply voltage in the target system before every action with the device.

**Disable test supply voltage** - disables the measurement and checking of the supply voltage for the programmed device, set in the Supply voltage edit box, before each action with the device.

**Delay after reset active** - this parameter determines the delay after the Reset signal is active before starting action with the device. This delay depends on the values of the devices used in the reset circuit of the device and can be chosen from these values: 10ms, 50ms, 100ms, 500ms or 1s.

I**nactive level of ISP signals** - this parameter determines the level of the ISP signals after finishing access to the target device. The signals of the ISP connector can be set to Pull-up (tied through 22k resistors to the supply voltage) or Pull-down (tied through 22k resistors to ground).

**Keep ISP signals at defined level after operation** - enables the keeping of the set level of ISP signals after access to the target device is finished. The control program indicates activated pull-up/pull-down resistors by displaying a window with a warning. After the user closes this window, the control program will deactivate the resistors.

group **Programming parameters**

This group is available for some types of devices. It contains settings for which device parts or areas have to be programmed.

group **Erase parameters**

This group is available for some types of devices. It contains special settings for the erase modes of the selected device.

***Device / Device options / Serialization***

Serialization is a special mode of the program. When a serialization mode is activated, a specified value is automatically inserted into a predefined address in the buffer before programming each device. When subsequent devices are programmed one by one, the serial number value is changed for each new device automatically and inserted into the buffer before programming, so each device has a unique serial number.

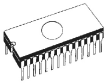
There are three types of serialization:

- Incremental mode
- From file mode
- Custom generator mode

**Serialization** dialog also contains settings for the associated serialization position files that are used with project files where serialization is enabled. For more detailed information about using serialization in project files, look at the Serialization and projects section.

**Basic rules of serialization:**

- serialization is associated with the currently selected device only. If a new device is selected, the serialization settings will be reset (serialization will be set to disabled).
- serialization settings for the current device are saved along with other settings for the device to the project file or to the configuration file when the application is closed.



- serialization engine calls for the new (next) serial number before each device programming operation is started.
- the used serial number is indicated by (*) after the serial number value. The used serial number means that the next device programming operation will use the next serial number after this indicator.

**Note:** *The request for a new serial number prior to programming may be suppressed following an unsuccessful programming attempt by enabling the option:* ***Serial number usage if programming action fails:***

- when **Reuse generated serial number for next programmed device** option is selected, the request for a new (next) serial number is suppressed in the case of an unsuccessful device operation result. It means that the used serial number is used again, and remains same until a successful device programming operation is completed.
- when **Throw away (use the serial number only once, regardless result of the programming)** is selected, the request for a new serial number is performed before each device operation, regardless of the result of the previous programming operation.

Serialization can work with the control program's main buffer or the extended buffers that are available for some types of device, for example Microchip PIC16Fxxx devices with Data EEPROM Memory. The selection for which buffer is use by the serialization routine is available in the Serialization dialog. If the Buffer settings box is not visible, the current serialization mode does not support extended buffers.

## *Device / Device options / Serialization / Incremental mode & SQTP*

The **Incremental mode & SQTP** enables you to assign individual serial numbers to each programmed device. A starting number entered by the user will be incremented by a specified step for each device program operation and loaded in the selected format to the specified buffer address, prior to programming each new device. Options available in the Incremental mode serialization also allow you to set an equivalent serialization scheme to the Microchip SQTP, used for Microchip PICmicro® devices.

There are the following options that the user can modify for incremental mode:

**S / N size**

S / N size option defines the number of bytes of the serial number value which will be written to the buffer. For Bin (binary) serialization modes, the values 1-8 are valid for the S / N size and for ASCII serialization modes, the values 1-16 are valid for the S / N size.

**Address**

Address option specifies the buffer address where the serial number value should be written. Note that the address range must be inside the device's start and end addresses. The address must be correctly specified so the last (highest or lowest) byte of the serial value is inside the device's start and end address range.

**Start value**

Start value option specifies the initial value from which the serialization will start. Generally, the max. value for serialization is $1FFFFFFF in a 32 bit long word.

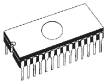
When the actual serial value exceeds the maximum value, the three most significant bits of the serial number are set to zero. After this action, the number is always inside the range 0...$1FFFFFFF (this is a basic style of overflow handling).

**Step**

Step options specify the increment step of the serial value incrementation.

**S / N mode**

S / N mode option defines the form in which serial value is to be written to the buffer. Two options are available:



- **ASCII** means the serial number is written to the buffer as an ASCII string. For example the number $0528CD in ASCII mode is written to the buffer as 30h 35h 32h 38h 43h 44h ('0' '5' '2' '8' 'C' 'D'), i.e. six bytes.
- **Bin** means the serial number is written directly to the buffer. If the serial number has more than one byte length, it can be written in one of two possible byte orders. The byte order can be changed in the **Save to buffer** item.

**Style**
Style option defines the serial number base. There are two options:
- **Decimal** numbers are entered and displayed using the characters '0' through '9'.
- **Hexadecimal** numbers also use characters 'A' through 'F'.

The special case is Binary Dec, which means BCD number style. BCD means the decimal number is stored as a hexadecimal number, i.e. each nibble must have a value from 0 to 9. Values A to F are not allowed as nibbles of BCD numbers.
**Note:** *Select the base in the Style options before entering numbers for the serial start value and step.*

**Save to buffer**
Save to buffer option specifies the serial value byte order to write to the buffer. This option is used for the Bin S / N mode (for ASCII mode, it has no effect).
Two options are available:
- *LSByte first* (used by Intel processors) will place the Least Significant Byte of the serial number at the lowest address in the buffer.
- *MSByte first* (used by Motorola processors) will place the Most Significant Byte first at the lowest address in the buffer.

**Split serial number**
This option allows you to divide serial number into individual fragments (mostly bytes) and place the bytes at each Nth address of the buffer. This feature is particularly useful for SQTP serialization mode for Microchip PIC devices, when the device serial number can be a part of the program memory as group of RETLW or NOP instructions. For more information see Example 2 shown in the Examples section below.

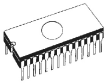
Following split options are available:
- Check box **Split serial number** – turns on/off the split function
- **Split gap** – specifies the number of bytes placed between the split serial number fragments
- **S/N fragment size** – serial number is split into fragments with the size specified by this option

**Example 1:**
Write serial numbers to AT29C040 devices at address 7FFFAH, size of serial number is 4 bytes, start value is 16000000H, incremental step is 1, the serial number form is binary and the least significant byte is placed at the lower address of the serial number in the device.

To make the above described serialization, the following settings have to be set in the Serialization dialog:

| | |
|---|---|
| Mode: | Incremental mode |
| S/N size: | 4 bytes |
| S/N mode: | Bin |
| Style: | Hex |

| | |
|---|---|
| Save to buffer: | LS Byte first |
| Address: | 7FFFCh |
| Start value: | 16000000h |
| Step: | 1 |



Split serial number: unchecked (empty box)

Following values will be written to device:
The 1st device
Address Data
007FFF0 xx xx xx xx xx xx xx xx xx xx xx 00 00 00 16
The 2nd device
Address Data
007FFF0 xx xx xx xx xx xx xx xx xx xx xx 01 00 00 16
The 3rd device
Address Data
007FFF0 xx xx xx xx xx xx xx xx xx xx xx 02 00 00 16
etc.
"xx" means user data programmed to the device
Serial numbers are written to the device from address 7FFFCH to address 7FFFFH because the serial number size is 4 bytes.

**Example 2:**
Following example shows usage of the SQTP serialization mode where the serial number is split into RETLW instructions for Microchip PIC16F628 devices.

**Note:** *Serial Quick Turn Programming (SQTP) is a Microchip specified standard for serial programming of Microchip PIC microcontrollers. Microchip PIC devices allow you to program a unique serial number into each microcontroller. This number can be used as an entry code, password, or ID number.*

*Serialization is done by using a series of RETLW (Return Literal W) instructions, with the serial number bytes as the literal data. To serialize, you can use the Incremental mode serialization or the From file mode serialization.*

*Incremental serialization offers the serial number Split function. Serial number split allows usage of incremental numbers separated into even or odd bytes and between each byte of the serial number a RETLW instruction code is inserted.*

*From file serialization is uses a proprietary serial numbers file. This file can consist of various serial numbers. The numbers can have a format suitable for SQTP, that means number RETLW b1 RETLW b2 and so on. Note that a PG4UW serial file format is not compatible with a SQTP serial file generated by Microchip MPLAB.*

**Example 2a:**
*Use of serialization split with RETLW instructions for Microchip PIC16F628 devices.*

Device PIC16F628 has a 14 bit wide instruction word. Instruction RETLW has a 14-Bit Opcode:

| Description | | MSB | 14-Bit word | | LSB |
|---|---|---|---|---|---|
| RETLW | Return with literal in W | 11 | 01xx | kkkk | kkkk |

where xx can be replaced by 00 and kkkk are the data bits, i.e. serial number byte

The opcode of the RETLW instruction is hexadecimal 34KKH where KK is the data Byte (serial number byte)

Let's assume we want to write the serial number 1234ABCDH as part of four RETLW instructions to the PIC device. The highest Byte of the serial number is the most significant Byte. We want to write the serial number to the device program memory at address 40H. Serial number split is very useful in this situation. Serialization without the serial number split will write the following number to the buffer and device:

Address Data
0000080 CD AB 34 12 xx xx xx xx xx xx xx xx xx xx xx xx

**Note:** *Address 80H is because the buffer has byte organization and the PIC has word organization, so it has the equivalent program memory address of 40H. When the buffer has word organization x16, the address will be 40H and the number 1234ABCDH will be placed in the buffer as follows:*

Address Data
0000040 ABCD 1234 xxxx xxxx xxxx xxxx xxxx xxxx

We want to use the RETLW instruction so the buffer has to be:
Address Data
0000040 34CD 34AB 3434 3412 xxxx xxxx xxxx xxxx

We can do this by following these steps:

**A)** Write four RETLW instructions at address 40H to the main buffer (this can be done by hand editing the buffer or by loading a file with the proper content). The bottom 8 bits of each RETLW instruction are not important now because the serialization will write the correct serial number bytes at the bottom 8 bits of each RETLW instruction.

The buffer content before starting a device programming operation will look for example as follows:

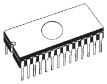
Address Data
0000040 3400 3400 3400 3400 xxxx xxxx xxxx xxxx

8 bits of each RETLW instructions are zeros, they can have any value.

**B)** Set the serialization options as follows:

| | |
|---|---|
| S/N size: | 4 Bytes |
| Address: | 40h |
| Start value: | 1234ABCDh |
| Step: | 1 |
| S/N mode: | BIN |
| Style: | HEX |
| Save to buffer: | LS Byte first |
| Split serial number: | checked |
| Split gap: | 1 byte(s) |
| S/N fragment size: | 1 byte(s) |

Split settings described above mean splitting the serial number by bytes to the buffer at every second byte. The correct serial number is set before the device programming operation starts.



The buffer content of the serial number when programming the first device will be:

| Address | Data |
|---|---|
| 0000040 | 34CD 34AB 3434 3412 xxxx xxxx xxxx xxxx |

The second device will have:
Address Data
0000040 34CE 34AB 3434 3412 xxxx xxxx xxxx xxxx

The following devices will have the same format of the serial number, of course incremented by 1 for each device.

**Example 2.b**
*Use of the serialization split with NOP instructions for Microchip PIC24FJ256 devices*

Device PIC24FJ256 has 24 bit wide instruction word. The Instruction NOP has the code 00xxxxh. Let's assume we want to use serialization in the same manner as SQTP serialization specified in the Microchip MPLAB®:

We can do this with the following steps:
**A)** Write the NOP instructions (00xxxxh) at address 800h to the main buffer of PG4UW. This can be done by hand editing the buffer or by loading a file with the proper content. The address 800h in the PG4UW buffer is equivalent to the PIC24Fxxx Program memory address 200h. For more details you should look at the Device information in PG4UW for the PIC24FJ256 device.

The buffer content with NOPs at address 800h before starting the device program operation should look for example as follows:

Address Data
0000800 00 00 00 00 00 00 00 00 xx xx xx xx xx xx xx xx

xx – means any byte value

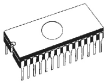
**B)** Set the serialization options as follows:

| | |
|---|---|
| S/N size: | 3 bytes |
| Address: | 800h |
| Start value: | 123456h |
| Step: | 1 |
| S/N mode: | BIN |
| Style: | HEX |
| Save to buffer: | LS byte first |
| Split serial number: | checked |
| Split gap: | 2 byte(s) |
| S/N fragment size: | 2 byte(s) |

Split settings described above mean splitting the serial number into fragments with 16 bit (2 bytes) size to the buffer with a gap of 2 bytes between the fragments. The correct serial number is set before the device programming operation starts.
The buffer content for the serial number when programming the first device will be:

Address Data
0000800 56 34 00 00 12 00 00 00 xx xx xx xx xx xx xx xx

The second device will have:
Address Data
0000800 57 34 00 00 12 00 00 00 xx xx xx xx xx xx xx xx



Additional devices will have the same format for the serial number but will be incremented by 1 for each device.

**Example 3:**
Following example uses the same serialization options as Example 2a; instead the serial number Split gap is set to 2 and 3.

When the Split gap is set to 2 bytes, the buffer content will look as follows:

Byte buffer organization:
Address Data
0000080 CD xx xx AB xx xx 34 xx xx 12 xx xx xx xx xx xx

Word16 buffer organization:
Address Data
0000040 xxCD ABxx xxxx xx34 12xx xxxx xxxx xxxx

When the Split gap is set to 3 bytes, the buffer content will look as follows:

Byte buffer organization:
Address Data
0000080 CD xx xx xx AB xx xx xx 34 xx xx xx 12

Word16 buffer organization:
Address Data
0000040 xxCD xxxx xxAB xxxx xx34 xxxx xx12 xxxx

**Note:** *When you are not sure about the effects of the serialization options, it is possible to test the actual serial number which will be written to the buffer. The test can be made using the following steps:*

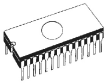
1. *select the desired serialization options in the Serialization dialog and confirm these with the OK button*
2. *in the Device operation options dialog, set the Insertion test and Device ID check (if available) to Disabled*
3. *check there is no device in the programmer's ZIF socket*
4. *run the Device Program operation (for some types of device it is necessary to select the programming options before programming will start)*
5. *after completing the programming operation (with some errors because a device is not present) look at the main buffer (View/Edit buffer) at the address where the serial number should be placed*

**Note:** The a*ddress for Serialization is assigned to the current buffer organization that the control program PG4UW is using for the current device. If the buffer organization is byte org. (x8), the Serialization Address will be a byte address. If the buffer organization is wider than a byte, e.g. 16 bit words (x16); the Serialization Address will be a word address.*

## *Device / Device options / Serialization / From file mode*

Using the From-file method, serial number values are read from the user specified input file(s) and written as serialization data to the buffer at the specified addresses.

There are two basic kinds of From-file serialization depending on the format of the serialization file used.



- **"Classic" From-file mode:** the serialization file has serial number values directly included. Serialization data is then read directly from the serialization file to the buffer address specified in the file. Classic From-file mode is indicated in the main window and info window of the PG4UW control program on the "Serialization" panel as "From-file" serialization. Description of the "classic" From-file serialization file is listed in the "Classic From-file serialization file format" chapter.
- **From-file mode** from a "playlist" file: the serialization file does not have serial number values directly included in it. The file contains a list of external files that contain the serialization data. Serialization data is then read from these external data files, each file means one serialization step (one device programmed). Playlist From-file mode is indicated in the main window and the info window of the PG4UW control program on the "Serialization" panel as "From-file-pl" serialization. Description of the "playlist" serialization file is listed in the "Playlist From-file serialization file format" chapter.

The PG4UW software selects the proper From-file serialization mode automatically, depending on the format of the user specified serialization file.

Serialization dialog offers the following options for From-file serialization:
**File name**
File name option specifies the serialization file name from which the serial number addresses and values will be read. The input file for the Classis From file serialization must have a special format, which is described in the From-file serialization file format below.

**Start label**
Start label defines the start label in the input file. The reading of the serial values from the file starts from the defined start label (or from the first uncommented label in the file, when the option **Comment used lines (serial records) in the main serialization file** is checked).

**Comment used lines (serial records) in the main serialization file**
When this checkbox is checked, the special "used lines comment mode" from the From-file serialization is used. In this mode, the serialization engine comments the line with the recent serialization record immediately after reading the serial value from the record. A comment means replacing the first character on the line with the ";" character. A commented line will never be used for serialization again, because the line will be determined by the serialization engine as a generic comment line. In this mode, there is no need to remember the last position (last serial number) used, because the serialization engine searches the main serialization file for the first uncommented line (record), each time the request for the next serial number occurs.

**Example:** When we have the following line (serial record) in the file before reading its value:
[label123] 20000 01 02 03 04 05 06 07 08
The format of the line (after reading its value and commenting it) will be:
;label123] 20000 01 02 03 04 05 06 07 08

**Advanced options for Playlist From-file serialization**

**Additional operation with used files**
The group box contains three types of operation. The user can select one of the operations to perform with the serialization data files in the Playlist From-file mode. The following operations are available:

- **Do nothing:** Program does not make any operation with the serialization data file(s)
- **Move used file to specified directory:** Program moves the serialization data files to a user specified directory for used serialization file(s)
- **Delete used file:** Program deletes the serialization data file(s)

**Directory**
This option is available in the playlist From-file serialization mode when the option "**Move used file to specified directory**" is selected. The user can specify the target directory into which the serialization data files will be moved.

**File format used for data files**
This option allows the user to select the file format of the serialization data files. The following options are available:

- Binary
- Intel HEX
- ASCII Space
- Autodetect - let's the serialization engine automatically determine the file format. Automatic recognition is supported for the file formats Binary, Intel Hex, ASCII Space, and Motorola.

**Note:** *The size of the main From-file serialization file is limited to 1000MB. Recommended maximum number of serial records (items) in one serialization file is 10,0000 records. More records may cause slower operation when reading the serial number before each device programming cycle.*

**Classic From-file mode**

**File format**
Classic From-file serialization input file has text format. The file includes addresses and arrays of bytes defining the buffer addresses and data to write to the buffer. The input file has a text type format, with the following structure:

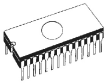
[label*1*] addr byte*0* byte*1*...byte*n*
...
[label*n*] addr byte*0* byte*1*...byte*m*, addr byte*0* byte*1*...byte*k*
_____/ _____/
| |
basic part optional part

; Comment

Meaning is:

**basic part**
Basic part defines the buffer address and the array of bytes to write to the buffer. The basic part must be always defined after the label in the line.

**optional part**
Optional part defines the second array of bytes and the buffer address to write to the buffer. One optional part can be defined after the basic part of the data.



**label*1*, label*n*** - labels
Labels serve as unique identifiers for each line in the input file, allowing them to be specifically addressed. By entering a required start label, the user defines the exact line where the reading of serial values begins.

**addr -**
Addr defines the buffer address to write the data to.

**byte*0*...byte*n*, byte*0*...byte*m*, byte*0*...byte*k*** -
Bytes arrays byte*0*...byte*n*, byte*0*...byte*m* and byte*0*...byte*k* are defining the data which is assigned to write to the buffer. The maximum count of bytes in one data field, following the address is 64 bytes. Data bytes are written to the buffer from address addr to addr+n.
The process of writing particular bytes to the buffer is:
byte*0* to addr
byte*1* to addr + *1*
byte*2* to addr + *2*
....
byte*n* to addr + *n*

**Optional part** is delimited from the first data part by the character "**,**" (comma) and its structure is the same as in the first data part, i.e. address and following array of data bytes.

**Characters with special use:**

[ ] - labels must be defined inside square brackets
"**,**" – character which delimits the basic part and optional part of the data
"**;**" - the semicolon character means the beginning of a comment. All characters from the "**;**" to the end of line are ignored. A comment can be on an individual line or at the end of a definition line.
**Notes:**

- *Label names can contain all characters except "[" and "]". The label names are analyzed as non case sensitive, i.e. the character "a" is same as "A", "b" is the same as "B" etc..*
- *All address and byte number values in the input file are hexadecimal.*
- *Allowed address value size is from 1 to 4 bytes.*
- *Allowed size of data arrays in one line is in the range from 1 to 64 bytes. When there are two data arrays in one line, the sum of their size in bytes can be a maximum of 80 bytes.*
- *Be careful to set the correct addresses. The address must be defined inside the device start and end address range. In the case where the address is out of range, a warning window appears and serialization is set to disabled (None).*
- *Address for Serialization is always assigned to the actual device organization and buffer organization that the control program is using for the current device. If the buffer organization is byte org. (x8), the Serialization Address will be a byte address. If the buffer organization is wider than a byte, e.g. 16 bit words (x16); the Serialization Address will be a word address.*

**Example of typical input file for Classic From file serialization:**

[nav*1*] A7890 78 89 56 02 AB CD; comment*1*
[nav*2*] A7890 02 02 04 06 08 0A
[nav*3*] A7890 08 09 0A 0B A0 C0; comment*2*
[nav*4*] A7890 68 87 50 02 0B 8D
[nav*5*] A7890 A8 88 59 02 AB 7D

;next line contains a second definition
[nav*6*] A7890 18 29 36 42 5B 6D, FFFF6 44 11 22 33 99 88 77 66 55 16

; this is last line - end of file

In the example file, six serial values with labels „nav1“, „nav2“, ...“nav6“ are defined. Each value is written to the buffer at address $A7890. All values have a size of 6 bytes. The line with the „nav6“ label also has a second value definition which is written to the buffer at address $FFFF6 and has a size of 10 bytes, i.e. the last byte of this value will be written to address $FFFFF.

***Note:*** *The serialization address is always assigned based on the device and buffer organization used by the control program. If the buffer is byte-organized (x8), a byte address is used. If the buffer is wider (e.g., 16-bit words (x16), the serialization address will be a word address.*

**Playlist From-file serialization file format**
From-file serialization playlist file includes list of filenames which contain serialization data. The file format is similar to classic serialization file format. Following file format differences are for playlist files:

1. The playlist file must have a special header on the first non-empty line. This header is a text line in the following format:
*FILETYPE=PG4UW SERIALIZATION PLAYLIST FILE*

2. each serial data batch is represented by a separate line in the following format
*[label x]* datafilename

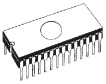
*labelx* - represents a label
Labels are identifiers for each non-empty line of the input file. They are used to address each line. Labels must be unique within the file. Addressing a line means the user enters a start label to define exactly where serial value reading begins.

*Datafilename* - This text specifies the name of the data file containing serialization data. When serialization requires a new value, the standard PG4UW "Load file" procedure loads the data file into the PG4UW buffer. The file can be in binary or Hex (such as Intel Hex) format. The auto-recognition system identifies the correct format and ensures the file loads properly. The data file name is relative to the parent (playlist) file.

**Example of playlist serialization file:**
;---- following file header is required ----------
*FILETYPE=PG4UW SERIALIZATION PLAYLIST FILE*

*;----- references to serialization data files*
*[nav1] file1.dat*
*[nav2] file2.dat*
*[nav3] file3.dat*
*...*

*[label n] filex.dat*
*;-------- end of file ----------*



For detailed and fully functional examples of “From-file serialization” files, look at the examples included with PG4UW, placed in the installation directory in the Examples\ subdirectory as follows:
<PG4UW_inst_dir>\Examples\Serialization\fromfile_playlist_example\

The typical path can look like this:
C:\Program
Files\DATAMAN_sw\Programmer\Examples\Serialization\fromfile_playlist_example\

You can test the serialization using the following steps:
3. start PG4UW
4. you need to have our programmer connected and correctly found in PG4UW
5. select the desired device, the best are devices with erasable memory, (not OTP memory)
6. select the dialog from the menu Device / Device Options / Serialization
7. Set the From-file mode and in the panel From-file mode options select our example serialization file fromfile_playlist.ser
8. click the OK button to accept the new serialization settings
9. run the "Program" device operation

You can view the serialization labels in the main window of PG4UW. You can also view them in the progress information window during device programming and repeated programming operations.

## *Device / Device options / Serialization / Custom generator mode*

The custom generator serialization mode provides maximum flexibility because the user has complete control over the serialization system.

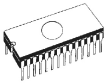
When custom generator mode is selected, serial numbers are generated "on the fly" by a user-created program before each device is programmed in PG4UW or PG4UWMC. This mode allows the user to generate any desired sequence of unique serial numbers. Serial numbers can be incremented as a linear sequence or a completely non-linear sequence. Details regarding the user-created serial number generator program are described in the following section **Custom generator program**.

**Examples:**
*There are also example .exe and C/C++ source files available. The files are placed in the PG4UW installation directory in Examples\ subdirectory as follows:*
*<PG4UW_inst_dir>\Examples\Serialization\customgenerator_example*
*The typical path can look like this:*
*C:\Program*
*Files\DATAMAN_sw\Programmer\Examples\Serialization\customgenerator_example*

The PG4UW control software offers the following options for custom generator serialization:

In the Serialization dialogue, select the Custom generator mode option in the Mode panel to display the following options:

**Serialization data file**
This text specifies the path and name for the data file that will contain the current serial number. When a device is ready for programming, the PG4UW software calls the user-created serial number generator to update the data file. The recommended extension for this data file is.dat.
Because many of our customers also use BP Microsystems programmers, they requested the ability to use the same serialization software. Therefore, this serialization data file is compatible with the .dat files used in the "Complex serialization" system available in the BP Microsystems software.



**Note:** *The data file is completely and periodically overwritten during device programming with serialization. Be sure to enter the correct name of the desired .dat file. Example: "c:\serial_files\serial.dat".*

**Serialization generator**

Specifies the path and name for the executable file which will generate the serialization data file, for example c:\serialization\generator.exe.

**First serial number**

This option is required to specify the initial serial number that will be passed to the custom generator serialization program. The format of the serial number can be a hexadecimal number or like any string that does not contain spaces, tabulators or CRLF characters.

**Last serial number**

This option specifies the maximum allowed value for the serial number. If the value is not an empty string or is "0", it will be passed to the serialization generator program. The generator is responsible for verifying the value of the last serial number and generating a serialization .dat file with appropriate error information if the current serial number exceeds the last serial number. If the value of the Last serial number is not specified or is zero ("0"), it will not be passed to the generator program.

**Check box Call generator with -RESULT parameter after device operation completed**

This new option serves a specific purpose. Check this box if you need to call a custom generator using the specialized **-RESULT** parameter. Otherwise, leave it unchecked (the default setting).

When enabled, the PG4UW control program calls the custom generator after every device operation, regardless of whether the operation succeeded or failed. The PG4UW serialization engine automatically creates the parameters for the generator. Two parameters are used:

**-RESULT[n]=TRUE / FALSE** and **-N <s/n>**

- **[n]**: An optional Programmer Site order number, used during multiprogramming.
- **TRUE / FALSE**: Indicates the operation outcome. **TRUE** means the device operation finished successfully. **FALSE** means the device operation failed with an error.
- **-N**: Contains the specific serial number targeted by the **-RESULT** call.

**Example**

A command line call of generator.exe -RESULT=TRUE -N1234 indicates that the operation was successfully completed for serial number 1234.

**The "Call generator and test consequent data file syntax" Button**

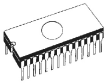
This button allows you to test your current settings by calling the generator and analysing the generated .dat file. Use this feature to verify your settings before applying them to production.

**Custom Generator Program**

A custom generator program (or serialization generator) is a user-created application. It generates a unique sequence of serial numbers and writes the data to a serialization .dat file. You must specify the path and file name of your program in the **Custom generator mode options** within the **Serialization options** dialogue box.

The PG4UW control program calls this application every time it needs to generate new serial data, typically before each device programming operation. PG4UW passes specific command-

line parameters to the program. The program then generates the .dat file, which PG4UW reads. The following command-line parameters are used:



*generator.exe - N<serial number> - E<serial number>*

- ***-N**: Specifies the current serial number the generator must use to create the proper serial data file. The generator must return this exact same value on line T01:<serial number> in the data file. For more information about the serialization .dat file format, please refer to the next section.*
    - *Note: If the PG4UW software detects a difference between the serial number passed to the generator via the -N parameter and the serial number specified in the T01:... record of the created data file, a serialization error will be reported.*
- ***-E**: Specifies the ending (or last) serial number. This parameter is only passed if the **Last serial number** value specified in the PG4UW **Serialization** dialogue box is non-zero. The serialization program should return error record T06 in the serialization .dat file if the current serial number exceeds this ending serial number. See the **Serialization .dat file format** section for complete details.*

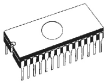
*The name of the executable file (generator.exe) can be replaced by any valid application name specified by the user.*

**Serialization .dat File Format**

The serialization .dat file created by your generator must strictly adhere to the following plain text format. The file consists of a records section and a serial data section.
Each record occupies a single line and begins with one of the Txx prefixes described below, where xx represents the specific record type code. Records pass critical serialization status information to the PG4UW software, such as current and last serial numbers, serialization data, data formats, and error codes.
Required records are records T01, T02, T03 and T04. Other records are optional.

- ***T01:<serial number>***
    - ***Description**: Contains the current serial number value passed to the generator by the command-line parameter -N<serial number>.*
    - ***Format**: Can be a hexadecimal number or any character string. It must not contain spaces, tabs, or carriage return/line feed (CRLF) characters.*
- ***T02:<serial number>***
    - ***Description**: Contains the next serial number value that PG4UW will use in the subsequent serialization cycle. This value is created by the generator to inform PG4UW which serial number follows the current one.*
    - ***Format**: Can be a hexadecimal number or any character string. It must not contain spaces, tabs, or CRLF characters.*
- ***T03:<data format code>***
    - ***Description**: Specifies the serialization data format. The following formats are currently supported:*
        - *T03:50 or T03:55 – ASCII Space data format*
        - *T03:99 – Intel Hex data format*
        - *T03:87 – Motorola Exormax data format*
- ***T04:***
    - ***Description**: Indicates that the actual serialization data will follow on the lines immediately after this record, continuing to the end of the file.*

- ***Format****: The data must be stored in one of the standard ASCII file formats (such as Intel Hex or ASCII Space) and must match the specific format declared in the* ***T03*** *record.*



**Example:** Typical serialization data file:

*T01:000005*
*T02:001006*
*T03:99*
*T04:*
*:0300000000096B89*
*:03000300000005F5*
*:02000C005A0197*
*:01003F004F71*
*:00000001FF*

*This file contains the following information:*

- ***Line T01**: The current serial number is 000005h.*
- ***Line T02**: The ending (last) serial number is 001006h. This value will be applied as the new current serial number in the next serialization cycle.*
- ***Line T03**: Specifies that the serialization data following line T04 is in Intel Hex format.*
- ***Line T04**: Indicates the actual serialization data. This data will be loaded into the PG4UW buffer before programming the device, using the specified Intel Hex format.*

***Optional Records***

- ***T05:<message>***
    - ***Description**: Warning or error message. This record immediately stops the serialization process and displays the specified warning or error message within the PG4UW software.*
- ***T06:***
    - ***Description**: Indicates that the current serial number exceeds the allowed limit. This record stops the serialization process and displays an error message in the PG4UW software. Use this record when the -E command-line parameter is specified (meaning a non-zero **Last serial value** was set in the **Serialization** dialogue box).*
- ***T11:<message>***
    - ***Description**: Low-priority warning or informational message. This record does not interrupt the serialization process.*

**Flowchart of Device Programming with Custom-Generator Serialization**

When using custom-generator serialization, the serialization engine calls the custom generator's executable file before starting each device programming operation to generate the .dat file. The PG4UW serialization engine automatically manages and passes the required command-line parameters to the generator.

The data from the generated .dat file is immediately read into the internal buffer of the programmer and applied as the active data for the target device. Additionally, PG4UW remembers the next serial number information (provided by record T02) for the subsequent cycle.

***The typical flowchart for device programming is as follows:***

1. ***Start of programming batch***
2. ***Device insertion test***
3. ***Serialization sequence (consists of four steps):***
    - ***Call the serialization generator with the proper command-line parameters to generate the serialization .dat file.***
    - ***Wait for the serialization .dat file to become available.***
    - ***Read the serialization .dat file data into the programmer buffer (this data will be used for programming the device).***
    - ***Delete the serialization .dat file after reading the data from it.***
4. ***Device programming***
5. ***Device verification***
6. ***Operation result check***
   ***This process is fully managed by the PG4UW control program. The serialization generator does not need to perform any operations based on the result. The control program will automatically call the serialization generator with the required command-line parameters:***
    - ***OK: PG4UW requests the next serial number. The next serial number was read from the .dat file in step 3. The next call to the serialization generator will have this new serial number specified in the command line.***
    - ***ERROR: PG4UW does not request a new serial number. The recent (current) serial number will be reused for the next device. The next call to the serialization generator will have this same serial number specified in the command line.***
7. ***Repeat programming with the next device?***
    - ***Yes: Go back to step 2.***
    - ***No: Continue to step 8.***
8. ***End of programming batch***

***Notes:***

- ***If a programming error occurs, the recent serial number is reused. However, the generator will still be called in step 3 anyway, even though it is using the same number as the previously failed device.***
- ***If a serialization .dat file error is detected, the PG4UW program immediately reports a serialization error and halts the programming batch.***
- ***The Serialization settings dialogue box does not perform a validity test of the custom generator mode settings when you click the OK button. Use the test button Call generator and test consequent data file syntax to verify your settings before final confirmation.***

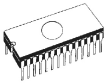
***Device / Device options / Statistics***

The statistics feature provides information on the actual count of device operations performed on the selected device type. If each physical device undergoes exactly one operation (e.g., programming), the total number of device operations will equal the total number of programmed devices.

**Countdown Function**

The countdown function allows you to track the remaining number of device operations—and therefore the number of devices—yet to be processed.

- The user defines a starting number of devices to be completed.



- After each successful device operation, the countdown counter automatically decrements by one.
- When the countdown value reaches zero, the specified quantity of devices is complete, and a user notification will be displayed.

---

The Statistics dialogue contains the following options:

- Operation Check Boxes (Program, Verify, Blank, Erase, and Read): Define which device operations will cause the statistics counters to increment.
- Success and Failure Counters: Any selected and performed device operation will increment the Total counter, as well as either the Success or Failure counter depending on the outcome.
    - *Note: A combination of partial operations is counted as a single operation. For example, a "Read" operation that includes a "Verify after Read" is one operation. A "Program" operation that includes "Erase" and/or "Verify" operations is also counted as one operation.*
- Count down Check Box: Enables or disables the countdown activity.
- Count down Edit Box: Defines the initial starting value for the countdown counter.

*Tip: The Statistics dialogue can also be opened by right-clicking the Statistics panel and selecting the displayed Statistics item.*

Statistics Dialogue Values

The Statistics dialogue displays seven distinct values:

- Success: Number of operations successfully completed.
- Operational failure: Number of operations that failed due to a device error.
- Adapter test failure: Number of operations that failed due to an incorrect programming adapter.
- Insertion test failure: Number of operations that failed due to the device being incorrectly positioned in the programming adapter.
- ID check failure: Number of operations that failed due to an incorrect ID code being read from the device.
- Other failure (prog. SW, HW): Number of operations that failed due to a programmer hardware error or control software error.
- Total: The total sum of all operations performed.

Statistics Panel

The active statistics values are displayed directly in the main window of the control program within the Statistics panel.

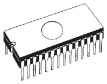
The Statistics panel contains four counter values (Success, Operational failure, Other failure, and Total) and two countdown values (Count down and Remains).

- Success: Number of operations successfully completed.
- Operational failure: Number of operations that failed due to a device error.
- Other failure: Number of operations that failed due to any reason other than a device error.
- Total: The total sum of all operations performed.
- Count down: Displays whether the countdown feature is currently enabled or disabled.

- Remains: Displays the remaining number of device operations to be completed.

Notes:

- When a new device type is selected, all statistics values reset to zero, and the countdown feature is automatically disabled.
- For the multiprogramming PG4UWMC software, the Reset statistics button is disabled while operations are actively running on any site. To stop active operations on all sites, press the Stop All button.
- The Reset button in the Statistics panel resets all statistics values to 0.
- The Reload Count down button in the Statistics panel restores the countdown counter to its initial user-defined starting value.
- For the PG4UW software, all statistics information is automatically saved to the Log window when you close the application.



For the multiprogramming PG4UWMC software, the statistics information is saved to the **Job Summary** report.

**Device / Device options / Associated file**
This command links an associated file with the current device. This file can be automatically loaded into the buffer when the device is selected from the default device selection list or when the control program launches.
You can edit the associated file name by entering the full file path in the file name box. The control program checks the disk to verify that this file exists. You can also enable or disable the automatic loading of this file.
Both settings—the associated file path and the automatic load setting—can be saved to the disk using the **File / Exit and save** command.

**Device / Device options / Special options**
The specialized terms used here are taken directly from the documentation provided by the manufacturer of the respective chip. Please refer to the manufacturer's documentation for the specific chip you intend to program for explanations of all used terms.
If the name of this menu item begins with "View/Edit ...", executing the **Read device** command will read the content of the chip configuration. It can then be viewed and edited using this menu command.

**Device / Blank check**
This command executes a device blank check. The control program reports the result of this action via messages in the **INFO** window and the **LOG**.
You can customize the available operation options by navigating to **Device / Device options / Operation options** in PG4UW.

**Device / Read**
This command allows you to read the entire device or a specific part of it into the buffer. The read procedure can also extract the contents of the chip configuration (if it exists and is readable). The specific device configuration areas can be viewed or edited in the dialogue boxes available under **View / Edit buffer** and **Device / Device options / Special options** (Alt+S).
The control program reports the completion of the Read action by writing a message to the **INFO** window and the **LOG**.
The **Device / Device options / Operation options** menu command allows you to define a non-standard working area. Enabling the **Verify data after reading** option in this menu command ensures higher reliability for device reading.

**Device / Verify**
This command compares the contents of the entire device, including special areas and settings (if available), with the data stored on the PC (in the buffer and/or other storage areas) or in the programmer itself (depending on the model). The control program reports the result of the verify operation to the **Info** window and the **Programmer activity log**.
You can customize the available operation options by navigating to **Device / Device options / Operation options** in PG4UW

The settings in the **Errors** tab of the **General options** dialogue box (accessible via **Options / General options** in PG4UW) allow you to control how detected errors are written to a user-specified report file. Additionally, the first 45 detected errors are automatically written to the **Programmer activity log**.

**Notes:**

- The **Verify** operation compares the contents of the entire chip with the data stored in the software. Consequently, if a chip is incompletely programmed, a verification run immediately after programming might show zero errors, while a standalone verify operation fails.
- The **Verify** operation may also report errors on protected devices that have active read protection enabled.
- Generally, the "whole device" refers to the range between the **Device start** and **Device end** addresses set in the **Device / Device options / Operation options** dialogue box. Note that not all devices support customized Start-End addresses. Some devices (such as NAND FLASH) have customizable sector or page counts/ranges instead; for these, the "whole device" refers to the range specified by those specific options.

**Device / Program**
This command executes device programming. The control program reports the result of this action via messages in the **INFO** window and the **LOG**.
You can customize the specific areas to be programmed and adjust other operation parameters by navigating to **Device / Device options / Operation options** in PG4UW.

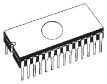
**Device / Erase**
This command executes a device erase. The control program reports the result of this action via messages in the **INFO** window and the **LOG**.
You can customize the available operation options by navigating to **Device / Device options / Operation options** in PG4UW.

*Note: After an Erase operation on a device that does not support a dedicated erase verify command, a blank check operation takes place automatically to verify the success of the erase.*

**Device / Test**
This command executes a test on the device selected from the supported devices list (such as static RAM), provided you are using a programmer that supports this test.
The **sRAM test** is executed in three basic steps:

- **Test of data drivers functionality**: The driver test checks the reaction of the D0 through D7 signals against the CE\, OE\, and WE\ control signals:
    - In the first cycle, the system writes the data 0x55 to address 0x0 (CE/=L, WE/=L, OE/=H) and compares it with the data read from that same address (CE/=L, WE/=H, OE/=L). The data must be valid.
    - Other combinations of control pins are then tested (e.g., CE/=L, WE/=H, OE/=H and CE/=H, WE/=H, OE/=L). For these states, the data must be invalid, indicating that the data bus driver is inactive.
- **sRAM test (basic part)**: The programmer writes random data to the sRAM device and then verifies the written content

.



- **RAM test, advanced (optional).**

"Walking one" and "Walking zero" are common terms, who need explanation can study:
http://www.google.com/search?q=memory+test+walking+one
http://www.google.com/search?q=memory+test+walking+zero

**Notes**:

- You can select a delay between the write operation and the succeeding verification of programmed data (provided the device remains supplied) to detect bit 'leaks'.
- The programmer cannot detect hardware errors such as excessive current on the signal pins or similar "analog" faults.
- All tests are executed at a low frequency compared to the maximum speed of the tested device; therefore, the application of this test is limited.

Conclusions:

- The device programmer can only provide basic feedback regarding the overall health of the sRAM.
- To test sRAM more thoroughly, please use a specialized sRAM tester.

**Device / IC test**

This command activates the test section for ICs, primarily Standard Logic ICs. The ICs are sorted by technology type into specific groups and libraries.
First, select the appropriate library and the desired device. Then, choose a mode for the test vectors to run (e.g., LOOP or SINGLE STEP). The control sequence and test results are displayed in the Programmer activity log. If needed, users can define the test vectors directly. A detailed description of the syntax and methods for creating test vectors is provided in the example_e.lib file, located in the program's installation folder.

Notes:

- IC testing is performed using test vectors at a relatively low speed. Tests executed by test vectors cannot detect all defects within the chip. In other words, if an IC test report reads "FAIL", the device is defective. However, if the test report reads "PASS", it simply means the chip passed our specific tests; it may still fail tests that check other dynamic parameters of the tested IC.
- Because the rising and falling edges of the programmer are tuned specifically for programming chips, testing on certain chips (such as counters) may occasionally fail even if the chips are not actually defective.

**Device / Jam/VME/SVF/STAPL/mDOC ... Player**

Jam STAPL was created by Altera® engineers and is supported by a consortium of programmable logic device (PLD) manufacturers, programming equipment makers, and test equipment manufacturers.

The Jam™ Standard Test and Programming Language (STAPL), JEDEC standard JESD-71, is a standard file format for ISP (In-System Programming) purposes. Jam STAPL is a freely licensable open standard. It supports the programming or configuration of programmable devices and the testing of electronic systems using the IEEE 1149.1 Joint Test Action Group (JTAG) interface. A device can be programmed or verified, but Jam STAPL generally does not support other operations such as reading a device.

The Jam STAPL programming solution consists of two components: the Jam Composer and the Jam Player.

- **The Jam Composer** is an application—generally provided by a programmable logic vendor—that generates a Jam file (.jam). This file contains the user data and the programming algorithm required to load a design onto a device.
- **The Jam Player** is an application that reads the Jam file and applies the required vectors for programming and testing devices within a JTAG chain.

**Device Programming Options**
Devices can be programmed directly in the Zero Insertion Force (ZIF) socket of the programmer or in the target system via an In-System Programming (ISP) connector.

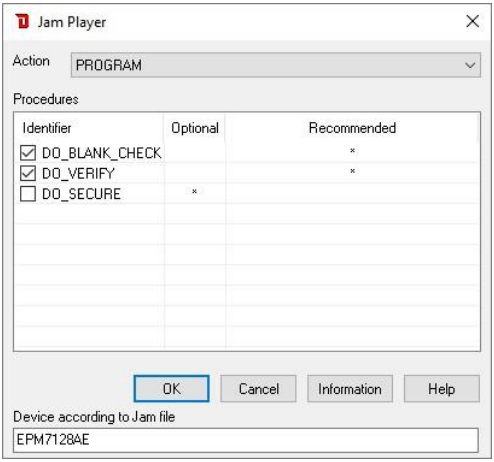
- A suffix such as **[PLCC44] (Jam)** or **(ISP-Jam)** will appear after the name of the selected device in the control program to indicate the mode.
- To program and test multiple devices via a JTAG chain, select **JTAG chain (ISP-Jam)**.

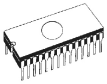
For more information, please visit the Altera website. You can also refer to the following application notes for detailed guidance:http://www.altera.com

- **AN 425**: Using the Jam Player to Program Altera Devices
- **AN 100**: In-System Programmability Guidelines
- **AN 122**: Using Jam STAPL for ISP & ICR via an Embedded Processor

**Software tools:**
***Altera: MAX+plus II, Quartus II, SVF2Jam utility (converts a serial vector file to a Jam file), and LAT2Jam utility (converts an ispLSI3256A JEDEC file to a Jam file).***
***Xilinx: Xilinx ISE Webpack or Foundation software (generates a STAPL file or SVF file for use by the SVF2Jam utility).***
***Actel: Actel Libero® Integrated Design Environment (generates a STAPLE file and/or PDB file), and Actel FlashPro (converts a PDB file to a STAPLE file).***

***JAM player dialog***







**Action**
Select the desired action for execution.
A Jam file consists of multiple actions, and each action consists of calling specific procedures that will be executed by the system.

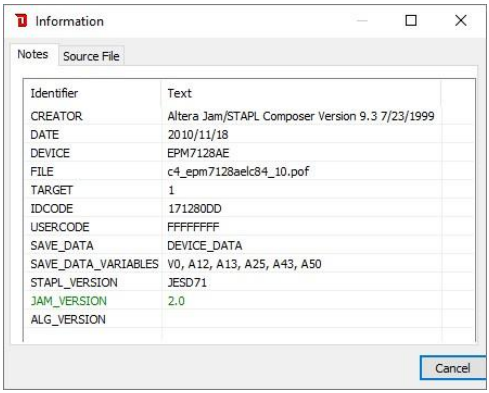
**Procedures**
The program flow executes specific statements from each procedure. Procedures can be either optional or recommended. Recommended procedures are marked automatically by default. You can manually enable or disable procedures based on your needs. The Jam Player will only execute the procedures you have marked; all other procedures are ignored. The total number of available procedures will vary depending on the specific Jam file.

**OK**
Clicking this button accepts the selected action along with all of your marked procedures.

**Information**
This displays detailed information regarding the active Jam file. You can also preview the associated notes and the source file directly within this dialogue box.

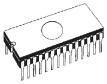
**Device According to Jam File**
Each file is created for a specific hardware device. The target device name is stored within the Jam file under the NOTE identifier DEVICE section. This device name must be identical to the name of the device selected in the **Select device** dialogue box. If the devices do not match, the software will display a warning message when the Jam Player launches.

JAM file information dialog




**NOTE Statements**
Statements are used to store attributes and miscellaneous information about the Jam file. The information stored in NOTE fields may include any type of documentation or specific parameters related to that particular Jam program.

**Source Files**
A source file contains a program written in the Jam language, consisting of a sequence of statements. A Jam statement consists of an optional label, an instruction, and arguments, and it always terminates with a semicolon (;).
Arguments may be literal constants, variables, or expressions resulting in the desired data type (such as Boolean or integer). While each statement usually occupies a single line of the Jam program, this is not required. Line breaks are not significant to the Jam language syntax, except when terminating comments.



An apostrophe character (') signifies a comment, which is completely ignored by the interpreter. The language does not specify any rigid limits for line length, statement length, or total program size. More information can be found on the Altera website: http://www.altera.com

***Jam files with the .jbc extension are stored in the Jam STAPL Bytecode format and are not viewable as plain text.***

***Converting a JED File to a Jam STAPL File for Xilinx Devices***
***To create a Jam STAPL file using Xilinx software, follow these steps:***

1. ***Install the Software: Download and install the free Xilinx Integrated Software Environment (ISE) 6.3i software (including files like WebPACK_63_fcfull_i.exe and 6_3_02i_pc.exe, totalling roughly 315 MB).***
2. ***Launch iMPACT: Open Xilinx ISE 6 and navigate to Accessories > iMPACT.***
3. ***Select Mode: In the "Operation Mode Selection: What do you want to do first?" dialogue box, choose Prepare Configuration Files.***
4. ***Choose File Category: In the "Prepare Configuration Files: I want to create a:" dialogue box, select Boundary-Scan File.***
5. ***Choose File Type: In the "Prepare Boundary-Scan File: I want to create a:" dialogue box, select STAPL File.***
6. ***Name the File: In the "Create a New STAPL File" dialogue box, enter your desired file name using the .stapl extension.***
7. ***Add Device: In the "Add Device" dialogue box, select your source .jed file.***
8. ***Set Operations: In the generated JTAG chain, left-click on the target device (e.g., XC2C32A). Then, right-click to select your sequence of operations (e.g., Erase, Blank, Program, Verify).***
9. ***Finalise the File: Navigate to the menu and select Output > Stapl file > Stop writing to Stapl file.***
10. ***Load into PG4UW: Open the PG4UW software. Select your target device (e.g., Xilinx XC2x32A [QFG32](Jam)) and load your newly created file (set the file type dropdown to STAPL File).***
11. ***Configure and Play:***
    - ***Press Alt+O or navigate to Device operation options and click the Jam configuration button.***
    - ***A warning will appear stating: "Select device from menu 'Select Devices' and Jam file is probably different! Continue?" Click Yes. (This occurs because Xilinx software does not automatically include the specific NOTE "DEVICE" "XC2x32A"; identifier line in its Jam files).***
    - ***In the Jam player dialogue box, select your desired actions and procedures.***
    - ***Close the dialogue box, click the Play Jam button on the toolbar, and monitor the process in the Log window.***

---

***Information About Actel Device Programming Using a STAPL File***
*Actel's flash FPGA programming in the PG4UW software is performed using the Actel Jam player. This specialized programming solution introduces a unique toolbar button—**Play STAPL**—which replaces all of the standard operation icons (such as Program, Erase, and Verify) that are normally associated with non-Jam programming devices.*

*Executing operations (such as programming, erasing, or verifying) on an Actel device consists of the following steps:*

**Loading the *.stp (STAPLE file)**

Load the appropriate STAPLE file (generated by design software like Actel Libero IDE) by clicking the **Load** icon on the main toolbar. This file contains the user data and the programming algorithm required to program a design onto a device.

**Selecting an Action**

After successfully loading the STAPLE file, select your intended operation under **Device Operation Options** (or use the **Alt+O** shortcut) in the STAPL configuration menu. To program the device, select **PROGRAM** from the action list. You can find a complete list of available actions and their descriptions in the Actel FlashPro User's Guide http://www.actel.com.

**Running an action**

***Click the Play STAPL button to run your selected action. A successful operation (such as programming) will conclude with the message "Exit Code = 0... Success" displayed in the log window.***

***Information about converting PDB file to JAM STAPLE for ACTEL devices***

Actel PDB files are proprietary and only supported by Actel programmers, such as FlashPro. Because the PG4UW control program can only program Actel devices using Jam STAPL files, you must convert PDB files to the STAPL format before proceeding.

**Steps to Convert a PDB File to a Jam STAPL File:**

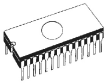
- **Install FlashPro:** Get the FlashPro software tool. It is included in the Actel Libero tool suite or available as a standalone download.
- **Launch the software:** Open the **FlashPro** application.
- **Create a project:** Click the **New Project** button (or go to **File > New Project**). Enter a name for your project, select your desired programming mode (single device or chain), and click **OK**.
- **Load the file:** Go to the **Configuration** menu, select **Load Programming File**, and choose the .pdb file you wish to convert.
- **Export the file:** Go to the **File** menu and select **Export > Export Single Device STAPL File...**. Enter a file name and click **Save** to export the STAPL file to your specified directory.
- **Finalise:** Once the conversion is complete, you can use the newly created .stp file to program your Actel device.
-

***Frequently asked questions about Actel***

**Q: How can I identify or verify an Actel device that has already been programmed?**

**A:** There are several ways to accomplish this. Each available action uses a different method to compare the loaded STAPL file against the data on the programmed Actel device. You can use the following actions found in a .stp file:

- **DEVICE_INFO:** This reads and displays device data in the log window, including the checksum of the programmed environment. You can manually compare this displayed value with the checksum found in the header of the STAPL file (viewable in the Information window). **Caution:** This checksum value is not calculated from the actual device data content; instead, it is stored in a specific memory location during programming and simply read back.



- **VERIFY_DEVICE_INFO:** This option is similar to the previous one, but it automatically compares the programmed device checksum with the STAPL file checksum. The system will display either a success message or an error window based on the result.
- **VERIFY:** This is the safest but slowest option to compare the programmed device content with the STAPL file. While the first two options take about a second, this method can take tens of seconds depending on the device capacity. It performs a bit-by-bit comparison of selected family features (such as the FPGA Array, targeted FlashROM pages, and security settings). If a data mismatch occurs, the verification process terminates immediately and logs an error message.

**Q: Is it possible to program an Actel device with two different STAPLE files in a single programming action in PG4UW?**
**A:** Yes, it is. The PG4UW control program features a built-in multi-project solution for this exact scenario. For example, you can program the actual data content (using the first STAPL file) and the security encryption key (using the second STAPL file) at the same time.

***The IspVM Virtual Machine***

**The IspVM Virtual Machine** is a specialized environment designed for programming devices that meet the IEEE 1149.1 Boundary Scan standard. By combining Lattice's IspVM Virtual Machine™ with the standard Serial Vector Format (SVF) language, the IspVM Embedded tool provides a powerful solution for Boundary Scan programming and testing.

Using the IspVM System software, users can generate VME files (hex-coded files containing JTAG chain data) that support both ispJTAG and non-Lattice JTAG devices. These files comply with IEEE 1149.1 standards and support SVF or IEEE 1532 formats. Devices can be programmed either via a programmer's ZIF socket or directly in-system using an ISP connector. In the control program, these methods are identified by the suffixes [PLCC44] (VME) or (ISP-VME). Additionally, the "JTAG chain (ISP-VME)" setting allows for the simultaneous programming and testing of multiple devices within a JTAG chain.
More information can be found on the website: http://www.latticesemi.com
**Software tools:**
**Lattice:** ispLEVER, IspVM System ISP Programming Software, PAC-Designer Software, svf2vme utility (converts a serial vector file to a VME file)

***Introduction to DirectC***

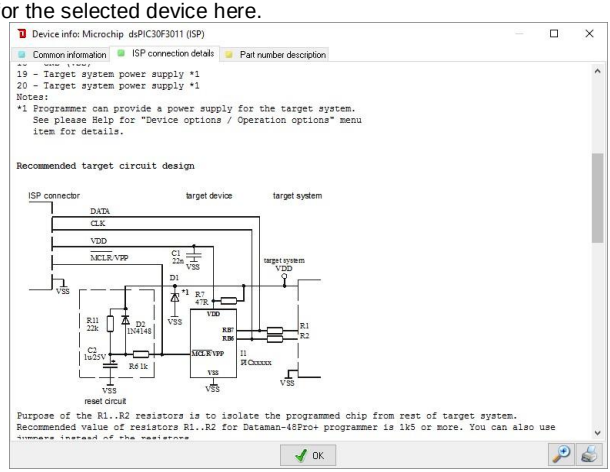
The DirectC system is designed to program a wide range of Microsemi devices, including the PolarFire, RTG4, IGLOO2, SmartFusion2, ProASIC3 (including ProASIC3 nano), IGLOO (including IGLOO nano), SmartFusion and Fusion devices. To function, DirectC requires a binary programming data file generated by Libero SoC; further details can be found at microsemi.com.

The DirectC Player executes the programming by reading DAT files and applying vectors to devices within a JTAG chain. Devices can be programmed using either the programmer's ZIF socket or directly in the target system via an ISP connector. You can configure these operations by clicking the "**DirectC configuration...**" button in the **Device Operation Options** dialogue and selecting the desired action.

Device / Device info

This command provides detailed information about the current device, including its size, internal structure, and programming algorithm. It also lists the specific programmers and

auxiliary modules that support it. Additionally, you can find package details and other general specifications for the selected device here.



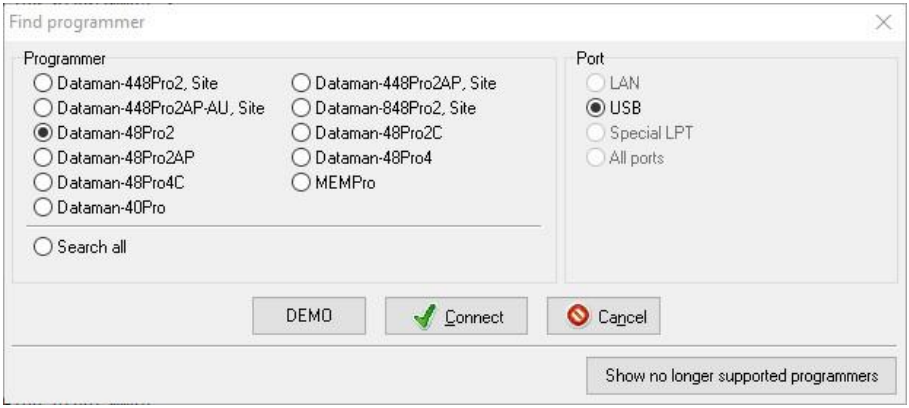

The reserved key **<Ctrl+F1>** will bring up this menu from any menu and at any time.

# *Programmer*

The Programmer menu contains commands for managing and operating your programming hardware.

Programmer / Find programmer

Select the programmer and communication parameters in the PG4UW software.




**Programmer** This section lists all programmers supported by the current version of the PG4UW software. Simply select the programmer you wish to connect. If you choose the **Search all** option, PG4UW will scan for any supported hardware and automatically connect to the first one it identifies.

**Port** This section displays a list of all available communication ports for the selected programmer. Select the specific port where the programmer is physically connected. If you are unsure which port to use, select **All ports**.

**Connect** Click this button to scan the selected communication port(s) for the chosen programmer. If the newly connected programmer differs from the previous one, the default device list will be cleared, though the most recently selected device will be retained if the new hardware supports it.

**DEMO** Click this button to enter demo mode for the selected programmer. In this mode, the PG4UW software will skip the communication port scan. If you switch to a different programmer while entering demo mode, the default device list will be cleared, though the last selected device will be kept if it is compatible with the new hardware.

**Cancel** click this button to close the Find programmer dialog without making changes.

**Connecting the programmer via USB:**

- Select a programmer with USB capability (e.g., DATAMAN-48PRO4 / DATAMAN-48PRO4C) to enable the USB port settings.
- Select the USB port and click the **Find** button.

The settings from the Find programmer dialogue are saved to the disk using the Options / Save options command.

Programmer / Refind programmer

This command is used to re-establish communication with the currently selected programmer.

To select a different programmer model, adjust communication parameters, or establish a connection with a new device, use the Programmer / Find programmer menu.

Programmer / Handler

In the Handler dialogue, you can select the handler type and configure its communication parameters. A handler is an external device used to automate device operations within the control program. Selecting "None" keeps the program in its default state, where the user manually controls all operations. Otherwise, the program enters a specialized mode where operations are performed automatically in coordination with the handler.

The **Handler** dialog contains the following items:

**Selected Handler** select the desired Handler type.
**Search at port** select a COM port which will be scanned for the requested Handler.

Pressing Enter or clicking OK will begin scanning for the handler based on your specified parameters. If the handler type is set to "None," the scanning process will be skipped. Your current handler settings are saved to the configuration file when you use the Options / Save options command or when the program is closed.

Please note that the handler is not available for purchase.

Programmer / Credit box info

The **Credit box info** menu item provides all the details regarding Credit boxes connected to your PC, including the model type, serial number, activation status, and current credit balance. If multiple boxes are connected, the software will also display the total combined credit available.

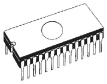
The status and balance of your Credit boxes are also visible in the main window via the **Credit box** button, located below the Statistics and Count Down sections. This button only appears when a device requiring "Paid ISP support" is selected and at least one Credit box is detected. The bar graph at the bottom of this button provides a quick visual indicator of your remaining credits.

green bargraph 85% of total credits available

yellow bargraph 50% of total credits available

red bargraph 10%, of total credits available

no bargraph 0% of total credits available (depleted)



Information about the available credits is periodically updated (also when the Credit box info window is open).

Working with devices that belong to the 'Paid ISP support' category

As customer needs evolve, programmable devices are becoming increasingly complex and diverse. Consequently, we must allocate significantly more development resources to support these new devices than in the past—both due to their technical complexity and the sheer volume of devices requiring implementation.

For offline programming (using a ZIF socket and adapter), these development costs are partially offset by adapter sales. However, for In-System Programming (ISP), customers only purchase the programmer, leaving ongoing support and software update costs unfunded.

To keep software updates free for ISP-mode devices, we have introduced a small fee for devices that require extensive development time or are rarely used.

The system is straightforward: to program these specific devices, a Credit box must be connected to your PC. This USB dongle comes pre-loaded with credits (ranging from 25,000 to 500,000 depending on the model). A small "micropayment" is deducted from your credit balance each time you program a supported device. Credit costs are very low, starting at approximately $0.01 per credit for the CB-25k model and dropping to about $0.003 per credit for the CB-500k version.

Each Credit box is valid for 10 regular software versions. Once activated during its first use, it remains compatible with the current version and the next nine subsequent releases (including all OnDemand versions). For example, a box activated with version 3.01 will work through version 3.10.

You can monitor your Credit box status directly in the main window of the PG4UW and PG4UWMC software. For more details, please refer to the Usage of Credit box for devices in 'Paid ISP support' category application note on our website.

Programmer / Module options

This option is used with multiple-socket programmers to define the MASTER socket and the status of each individual socket.

MASTER socket group box: Allows you to designate a preferred socket for device reading operations.

Enable/Disable socket checkboxes: Lets you manually enable or disable each socket. Disabled sockets are ignored during all device operations.

Programmer / Automatic YES!

This command enables Automatic YES! mode. In this mode, the software automatically repeats the last operation whenever a new device is detected in the ZIF socket, eliminating the need to press any buttons. You can cancel this process by pressing  while the program is waiting for a device to be inserted or removed.

**Operation and Indicators:**

**Post-Operation**: Once an operation finishes, either the OK or ERROR LED will light up based on the result, and the BUSY LED will blink.
**Device Removal**: When you remove the device, the status LED turns off, but the BUSY LED continues to blink, signaling that the system is ready for the next device.
**Device Insertion:** Once the programmer detects the first pins of a new device, the BUSY LED will glow steadily. The program then waits for a set duration (Device insertion complete time) for the remaining pins to be seated.
**Finalization:** If the device is not seated correctly within this timeframe, the ERROR LED will light up. If the insertion is successful, all status LEDs will turn off (except BUSY), and the operation will begin automatically.

You can enable or disable this feature via the **Automatic YES!** mode menu item. Note that selecting a new programmer through **Options / Find programmer** will automatically disable this mode.
**Response Time:** This is the delay between inserting a chip and the start of the operation. Increase this value if you need more time to properly position the chip in the socket.
**Programming Adapter Used:** This field displays the specific adapter required for the currently selected device.
**Excluded Pins:** This list identifies ZIF socket pins that will not be sensed by Automatic YES! (typically due to connected capacitors).
**Automatic YES! Parameters Setting Wizard:** Run this wizard to detect pins with bypass capacitors and add them to the exclusion list. While the software provides default exclusions for specific adapters, you should rerun the wizard if you add your own capacitors to the programmer or adapter.
**Reset Automatic YES! Parameters:** Click this button to discard wizard-generated settings and return to the software's default parameters.
**Device Removal Hold Off Time:** This is the wait time (1–120 seconds, default 2s) after removing a device before the software begins scanning for the next insertion.
**Device Insertion Complete Time:** This is the window (1–120 seconds, default 5s) allowed to finish seating a device after the first pins are detected. If the device isn't fully seated within this time, the software will report an insertion error.
**Suspend on Error:** This setting determines whether Automatic YES! pauses after an error so you can review the result, or continues to the next device without stopping.
The options are set to defaults after new device is selected by **Device / Select device**

These settings are preserved using the **Options / Save options** command and are also included in project files via **File / Save project**. Automatic YES! settings stored in a project file can be utilized in the **PG4UWMC** multiprogramming software by selecting "Use settings according to the last loaded project file" within the **General options / Automatic YES!** tab.

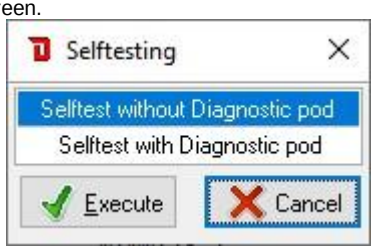
**Note:** When using socket adapters that contain passive or active components (such as bypass capacitors), you must run the Automatic YES! parameters setting wizard. This ensures the software identifies these pins correctly. If this is not done, the system may incorrectly detect a device as still being present, preventing you from starting a new programming cycle.

Programmer / Selftest

Executes a selftest of the connected programmer.

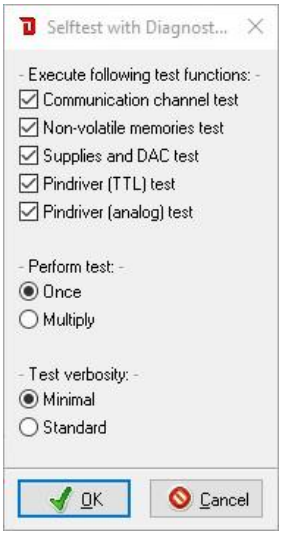
First, a self-test is performed using the Diagnostic POD included in the programmer's delivery package. If no errors are found, a second self-test is run without the POD. The results for each specific function, as well as the overall outcome, are recorded in the Programmer Activity Log.

Once the full self-test completes successfully, the dialogue window shown below will appear. From this window, you can choose to run additional loops of the test or return to the application's main screen.



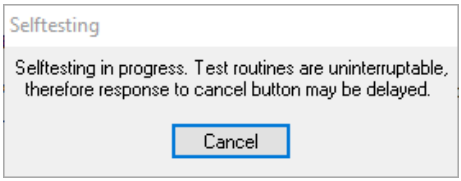

- **Selftest without Diagnostic pod:** selects selftest without the Diagnostic pod (or module) additional loop(s).
- **Selftest with Diagnostic pod:** selects selftest with the Diagnostic pod (or module) additional loop(s).
- **Execute:** clicking this button will start executing the selected selftest.
- **Cancel:** clicking this button will finish the selftest and return to application's main screen.

After selecting the selftest the following dialog is displayed prior to executing additional selftest loop(s).

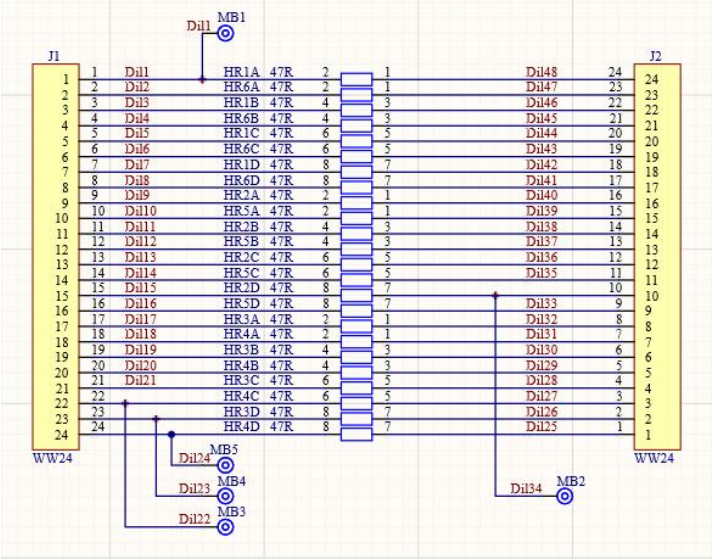


- In the Execute section, you can select specific programmer components to test in additional loops. The options available in this section vary depending on the programmer being tested.
- In the Perform test section, you can choose whether the selected test functions will run once or multiple times.
- In the Test verbosity section, you can select how much detail is recorded in the Programmer Activity Log during the self-test.
- **OK** - clicking this button will start executing the selected selftest functions.
- **Cancel** - clicking this button will finish the selftest and return to the main screen.

You can cancel the programmer's self-test at any time by clicking the Cancel button in the progress window, as shown in the picture below.

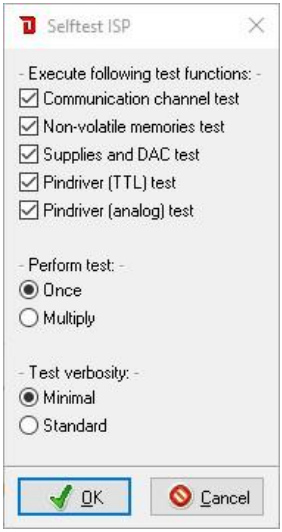


Schematic of the 48 Pins Diagnostic pod - Type I (if you are in a hurry):



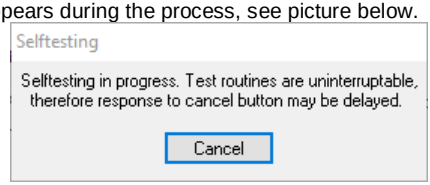

Programmer / Selftest ISP connector

This command executes a selftest of the ISP connector for the current programmer using the diagnostic pod for ISP connectors.



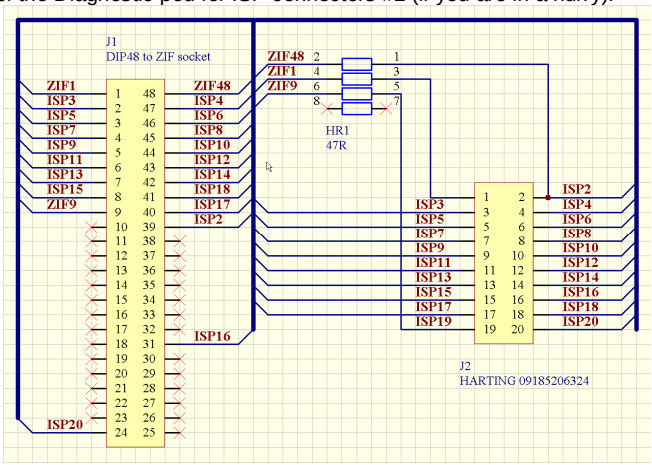
- In the Execute section, you can select specific programmer components to test in additional loops. The options available in this section vary depending on the programmer being tested.
- In the Perform test section, you can choose whether the selected test functions will run once or multiple times.
- In the Test verbosity section, you can select how much detail is recorded in the Programmer Activity Log during the self-test.
- **OK** - clicking this button will start executing the selected selftest functions.
- **Cancel** - clicking this button will finish the selftest and return to the main screen.

You can cancel the ISP connector self-test at any time by clicking the Cancel button in the window that appears during the process, see picture below.



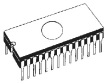

**Diagnostic pod for ISP connectors #2** is used for testing the 20 pin ISP connector of the programmers. The **Diagnostic pod for ISP connector #2** is available as a standard accessory for the DATAMAN-48PRO4 / DATAMAN-48PRO4C.

Schematics of the Diagnostic pod for ISP connectors #2 (if you are in a hurry):




*Sequence for testing the 20 pin ISP connector:*

1. Insert the Diagnostic POD for ISP connectors #2 into the programmer's ZIF socket. Ensure it is inserted as a 48-pin device.
2. Interconnect the 20-pin connector on the Diagnostic POD for ISP connectors #2 with the programmer's ISP connector using the provided ISP cable. Ensure all pins are properly aligned (e.g., 1-1, 2-2... 20-20).



3. Run the selftest of the ISP connector in the PG4UW (Programmer / Selftest ISP connector…).

We recommend to run this test every 6 months.

# *Options*

The Options menu contains commands that let you view and change various default settings.

Options / General options

The General options dialog allows you to control and configure various PG4UW program settings. These options can be saved to the configuration file when closing the application or at any time using the **Options / Save options** command.

***File options***

The File options page allows you to configure settings for erasing the buffer before loading, automatically reloading the current file, and choosing the recognition method for file formats.

The **Erase buffer before loading** option automatically clears the main buffer with a specified value before a data file is loaded. You can specify the erase value using one of two options:

- **Autodetect: the** buffer will be erased (filled) with the default "blank" value for the recently selected device and buffer
- **Custom defined:** The buffer will be erased and filled with a user-specified byte value.

This erase occurs immediately before the file content is read into the buffer and is compatible with binary and all HEX file formats. The main buffer refers to the primary buffer accessible via the **View/Edit buffer** dialogue (**Buffer / View/Edit**... menu).

In the When current file is modified by another process group, you can set the reload mode for the currently loaded file. There are three choices:

- Prompt before reloading file
- Reload automatically
- Ignore change scanning of current file

There are three situations when the file modification is checked:

- switching to the control program from another application
- selecting the device operation **Verify** or **Program**
- when a repeat of the last device operation is selected in the "Repeat?" dialog.

The **Load file format** setting determines how the program identifies file types during the loading process. When Automatic is selected, the program analyzes the file and tests it against all supported formats. If a match is found, the file is read into the buffer using the detected format. When Manual is selected, the user must explicitly choose the desired format from a list. If the selected format does not match the actual file type, the file may load incorrectly or incompletely.

The Show **Load recent project** dialog on program start checkbox enables this dialog to appear whenever PG4UW is launched. The Load recent project window displays a list of your project history, allowing you to quickly select and load a project or close the dialog without loading a file.

***File extensions***

The file extensions page allows you to set file masks.

**File format masks** are used to set filename filters for all file formats in the **File / Save** and **File / Load** windows. Each mask must include at least one wildcard (* or ?) and follow the correct syntax.

**Note:** Multiple masks can be specified for each file format. Use a semicolon as a delimiter to separate different extensions.
**Example:**
*Motorola:* **.MOT;*.S19*
*Defines two file masks* **.MOT and *.S19 - for Motorola file format.*
**Project file default extension** is used for setting the project files-extension used as the default in the **File / Load** project and **File / Save** project dialogs.

***Buffer***

The **Erase buffer before selecting new device** checkbox automatically clears the main buffer whenever a new device is chosen. This is useful for specific devices that require exact data at certain addresses which may not be included in the loaded data file.
The main buffer can be filled with the selected device's default "blank" value or a custom-defined value, configured in the **Erase value** and **Custom erase value** fields. As a reminder, the **main buffer** refers to the primary buffer accessible via the **Buffer / View/Edit...** menu.

Note: This setting is saved to the PG4UW configuration (.ini) file, not to individual project files.

***Language***

This page allows you to select a different language for the user interface, including menus, buttons, dialogues, and messages. You can also choose a help file in your preferred language. Please note that a **language definition file** is required to support additional interface languages.

***Sound***

The Sound settings page allows you to configure the program's audio notifications. Sounds are generated following specific activities, such as device operations (programming, verifying, reading, etc.), or when a warning or error message appears. You can choose to use Windows system sounds (requires a sound card), the PC speaker, or no sound at all.

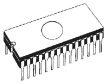
The Allow sound for the following actions panel includes the following options:
**Successful operation** check box
- When checked, a sound will play after a device operation completes successfully. When unchecked, no sound will be generated for successful operations.
- **In case of error** check box
  When checked, a sound will play if a device operation fails. When unchecked, no sound will be generated if an error occurs.

In the **Programmer internal speaker sound settings** panel, you can configure audio options for programmers equipped with a built-in speaker. The internal speaker will emit beeps after each operation to indicate whether the result was successful or unsuccessful.

***Colors and LEDs***

**Colors of the work result LEDs of the programmer:**
- **Standard color scheme** (ERROR=red, BUSY=yellow)
- **Former color scheme** (ERROR=yellow, BUSY=red)

**Note:** These settings are available only for newer programmer models. If these options are missing from the menu or are disabled, your programmer does not support LED color scheme customization.



**Colors of the work result indication in the software:**

- **Standard color scheme** (ERROR=red, BUSY=yellow)
- **According to the LEDs on the** programmer (ERROR=yellow, BUSY=red)

**Note:** *These settings are only available for older types of programmers.*

**LED brightness** allows you to customize the intensity of the LED lights. **Note:** This setting is only available for newer programmer models.

**Label style** allows you to highlight important labels (such as Programmer type, Device type, Statistics, Checksum, etc.) in the application's main window.

***Errors***

This option allows you to save device verification errors to a file. When verification fails, the first 45 differences are displayed in the Log window. If you wish to save these data differences to a file, you can choose between two methods in the **Save device verify errors to file** section: appending errors from all actions to a single file or saving only the errors from the most recent action. The errors will be saved using the filename specified in the **Error file name** field.

The following error report file options are available:

| | |
|---|---|
| option **No** (default) | Verification error logging to a file is disabled. Errors will only be displayed on the screen. |
| option **New** | Save verify errors to a file only from the last verification action. Before writing the results of a new action, the existing file is deleted and replaced with a new one. |
| option **Append** | Verification errors from all verification actions are cumulated into the same file. If the file does not already exist, a new one will be created. |

The **Error report file size limit** box allows you to set the maximum number of verification errors saved to the file. It includes the following options:

- **Stop verification after max. number of errors reached** checkbox. If checked, the verification process will stop once the Max. number of errors has been written to the file. If unchecked, all verification errors will be saved.
- The **Max. number of errors** edit box specifies the maximum number of verification errors that can be recorded in the error file during a single operation.

***Log file***

These options control the **Log window** and allow all reports to be mirrored to a **Log file**. By default, the log file is named "Report.rep." The program will create this file using the name and directory specified in the **Log file name** field.
The following Log file options are available:

| | |
|---|---|
| • **Create none** | The content of the Log window will not be copied to the Log file; all reports will be displayed in the Log window only. |
| • **Rewrite** | This option deletes the old Log file and creates a new one each time the control program is launched. |
| • **Append** | By default, this adds Log window reports to the existing Log file. If the file does not exist, a new one will be created. |

The **Add date information to Log file name** checkbox allows you to maintain a separate log file for each day. When enabled, the software automatically appends the current date (year/month/day) to the filename specified in the **Log file name** field, following these rules:

If the user specified log file name has the format:
**<user_log_file_name>.<log_file_extension>**

The name with the added date will be:
**<user_log_file_name><-yyyy-mmm-dd>.<log_file_extension>**

The new part representing the date consists of: yyyy - year, mmm - month and dd - day.

**Example:** *User specifies Log file name: c:\logs\myfile.log*
*The final log file name with the added date will look like this (having a date of November, 7th, 2026):*
*c:\logs\myfile-2026-nov-07.log*

If do you wish to have a log file name without a prefix before the date information, you can specify the log file name as:
**<log_file_extension>** - dot is first in the file name

**Example:** *User specifies the Log file name: c:\logs\.log*
*The final log file name with the added date will look like this (have a date November, 7th, 2026):*
*c:\logs\2026-nov-07.log*

Advanced options for managing the Log file size limit are also available

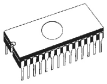
- **Use Log file text truncating when file size limit is reached** option: When checked, the Log file size limit is active. Once the file reaches the specified size, older text will be truncated. If unchecked, the Log file size is unlimited, subject only to available disk space.
- **Maximum Log file size** specifies the maximum size of the Log file in kB.
- **Amount of truncated text** specifies the percentage of text to be truncated once the Maximum Log file size is reached. A higher value means more text will be removed from the log file.

The Date format style in the log drop-down list allows you to select the preferred date format for all timestamped entries in the Programmer Activity Log.

The Log file settings can be saved to disk using the command **Options / Save options**.

***Job Report***

Job Report provides a summary of the most recent device operation. A "Job" is associated with a specific project file and covers all activity from the moment a project is loaded until a new project is opened or the PG4UW application is closed.



The Job Report contains the following information:
- project name
- project date
- Protected mode status
- PG4UW software version
- programmer type and serial number
- job execution start time (the time the Load project operation was performed).
- job execution end time (the time the Job Report was generated).device name
- device type
- checksum
- device operation options
- serialization information
- statistics information

The Job Report is generated in following cases:
- the user selects the Load project command.
- closing or disconnecting of programmer sites is selected
- closing the PG4UW software
- the device Count down counter reaches 0 (finished status)
- manually by the user, when the menu command "File / Job Report" is used

The Job Report is generated for the currently loaded project only if the Total statistics value is greater than zero. This means at least one device operation (such as program or verify) must have been performed

The following options are available for the Job Report:

The **Enable Job Report function** checkbox activates the Job Report feature when checked. If unchecked, the function is disabled, though you can still manually create a Job Report from the File menu.

The **Automatically save Job Report file** checkbox enables the automatic saving of reports to the directory specified in the **Job Report directory** field. The file name will be generated as follows:

job_report_<ordnum>_<prjname>.jrp

where
<ordnum> is the sequential file number. If report files with the same name already exist, the number for the new file is incremented based on the existing count.
<prjname> is the name of the most recently used project, excluding the file extension.

**Example 1:**
*For example, if you use the project file **c:\myproject.eprj** and set the Job Report directory to *d:\job_reports* (with no existing reports in that folder), the final file name will be:*

*d:\job_reports\job_report_000_myproject.jrp*

**Example 2:**
*Using the conditions from Example 1, assume there is already one report file present named d:\job_reports\job_report_000_myproject.jrp.*
*The final file name for the new Job Report will be: d:\job_reports\job_report_001_myproject.jrp*

**Note:** *The sequence number in the filename is incremented by one for each new file.*

When the **Automatically save Job Report file** setting is enabled, no dialogue windows will appear during the report generation process. The new Job Report is saved directly to the file without prompts or messages, provided no errors occur during the saving process.

If the **Automatically save Job Report file** checkbox is unchecked, PG4UW will display the **Job Report** dialogue whenever necessary.
In this dialogue, you can choose how to handle the Job Report. If you choose not to perform an operation (by clicking the **Close** button), the report will only be written to the PG4UW **Log Window**

***Automatic YES!***

This option allows you to set the indication mode for when the programmer and software are waiting for a programmed device to be replaced with a new one while in **Automatic YES!** mode.

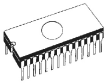
**LED Busy blinking** - The programmer indicates it is waiting for a new device by blinking the **Busy LED**. Once an operation is complete, either the **OK** or **Error** LED will light up (depending on the result) while the Busy LED continues to blink.When the program detects that the device has been removed from the ZIF socket, the status LED turns off, but the Busy LED keeps blinking to show it is ready for the next device.
Once the program detects one or more pins of a new device, the Busy LED stays on steadily. The program then waits for a set duration (Device insertion complete time) for the remaining pins to be seated. If the device is not correctly inserted before this time expires, the Error LED will light up. If inserted correctly, the status LED turns off and the next operation begins automatically.

**Not indicated (quiet mode)** – The programmer does not provide a specific indication while waiting for a new device to be inserted. After an operation, only the **OK** or **Error** LED will light up, depending on the result. This LED turns off as soon as the device is removed from the ZIF socket.

***Remote control***

The Remote Control feature allows external applications to manage certain PG4UW functions. This is ideal for integrating the device programmer into mass-production handler systems or other custom applications.

In this setup, the remote application controlling PG4UW acts as the Server, while the PG4UW program acts as the Client. Communication is handled via TCP protocol, which allows PG4UW and the remote control application to run on separate computers connected over a network.



Default TCP communication settings for remote control are:

Port: **telnet** Address: **127.0.0.1** or **localhost**

The Address setting applies only to the PG4UW (Client), while the Port setting applies to both the PG4UW (Client) and the Server application.

The default settings enable remote control on a single machine (using the localhost address). In this configuration, both the PG4UW (Client) and the remote control Server must be installed on the same computer.

**Note:** *If a firewall is installed on your system, it may display a warning when the remote control Server or Client starts. When prompted to allow or deny network access, please select 'Allow'; otherwise, the remote control will not function. Alternatively, you can configure your firewall settings more strictly to allow access only for a specific address and port.*

For more information on the PG4UW remote control feature and demonstration applications, please refer to the **remotemanual.pdf file located in the \RemoteCtrl subdirectory of the PG4UW installation folder**. You can also access the remote control manual via the Remote manual link in the Windows **Start / Programs** menu, which is created during installation.

***Save options***

This page allows you to choose how program settings are saved when you exit the application. There are three options available:

| | |
|---|---|
| **Don't save** | Closes the program without saving options and without prompting the user. |
| **Auto save** | Automatically saves all options when exiting the program without a confirmation prompt. |
| **Prompt for save** | The program asks whether you would like to save your settings before quitting. You can then choose to save or discard the changes. |

***Other***

The **Other** page allows the user to manage other program settings.

The **Application priority panel** allows you to set the program's priority level. These settings can affect programmer performance (specifically device programming time), particularly if other demanding applications are running on the system. Please note that setting the priority level to Low can significantly slow down the program.

In the **Tool buttons** panel, you can modify how hints are displayed for toolbar buttons in the main window. In the **Start-up directory** panel, you can choose which directory the program opens upon launch.

The **Default start-up directory** is the folder from which the program is executed. The **Directory in which program was lastly ended** refers to the last active directory used before the program was closed, which is the first entry in the directory history list.

Options / View

Use the **View** menu commands to show or hide various elements of the program environment, such as toolbars.

The following toolbars are currently available:

***Options / View / Main toolbar***
Choose this command to show or hide the Main toolbar.
***Options / View / Additional toolbar***
Choose this command to show or hide the Additional toolbar.
***Options / View / Device options before device operation***
Choose this command to enable/disable display of the Device options before device operation is confirmed.

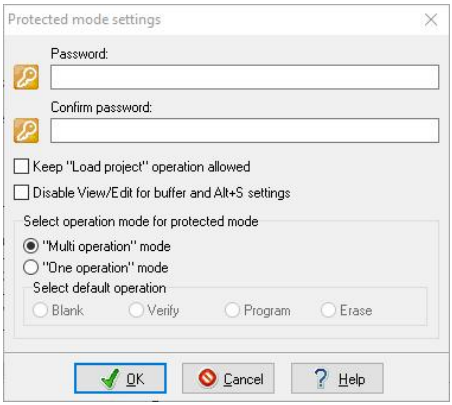
Options / Protected mode

**Protected Mode** is a restricted operating state designed to prevent unauthorized or accidental modifications to program settings and device configurations. In this mode, certain commands—such as resetting statistics, editing buffer content, loading/saving data, and reading devices—are disabled.

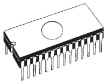
This mode is ideal for mass-production environments where the same device type is programmed repeatedly, ensuring an operator cannot unintentionally alter the setup. **Protected Mode** is available independently in both the **PG4UW** single-programming software and the **PG4UWMC** multi-programming software.

***Protected mode in PG4UW***
There are two ways to switch the program to Protected Mode:

1. By using the menu command **Options / Protected mode**. This command opens a password dialogue where you must enter and confirm a password. Once confirmed, the program switches to **Protected Mode**. This same password is required to switch the program back to **Normal Mode**.
2. By loading a project that was previously saved in protected status (a **Protected mode project**). For further details, please refer to the **File / Save project** section.




The **Keep "Load project" operation allowed** checkbox is disabled by default, meaning the **Load project button and menu command are unavailable while in Protected mode**. If this option is enabled (checked), users will still be able to load projects even while the software is in Protected mode.



The **Disable View/Edit for buffer and Alt+S** settings checkbox is inactive by default. This means the **View/Edit buffer** button and menu remain enabled in Protected Mode, allowing users to view—but not edit—the buffer content and settings.

Enable this option if you want to completely block the viewing of buffer content while in Protected Mode. If you choose this, we also recommend enabling the **Encrypt project file (with password)** option for added security. For more details, see the **File / Save project** section.

**Select operation mode for protected mode**
**Multi-operation mode** is the standard form of Protected Mode. It enables all available device operations—such as blank check, verify, program, and erase—while disabling the read function. This ensures an operator cannot intentionally or accidentally modify buffer data via a read operation. It is ideal for projects where you need access to all supported device operations within a single multi-project setup.

The **"One operation" mode** is an enhanced form of Protected Mode that enables only a single operation at a time. This provides greater security by preventing operators from accidentally executing the wrong device operation.

By using the **Multi-project Wizard** to combine multiple projects saved in "One operation" mode, you can create custom, non-standard operation sequences (e.g., Program + Verify + Verify + Verify). Please refer to the examples for further details on the differences between these operation modes.

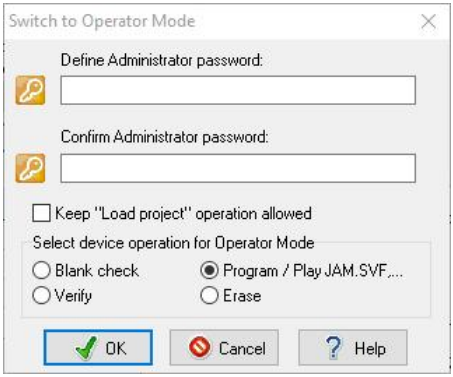
To switch the program from **Protected Mode** back to **Normal Mode**, use the menu command **Options / Normal Mode**. When the "Password required" dialogue appears, enter the same password used to activate Protected Mode.
Alternatively, you can cancel Protected Mode by closing the program. The software will launch in **Normal Mode** the next time it is opened (unless a protected project is loaded via a command-line parameter).
When Protected Mode is active, a "**Protected Mode**" label will appear in the top-right corner of the Programmer Activity Log.
For information regarding the **Require project file unique ID before first programming** option—indicated by the **(ID)** label next to the project name in the bottom status line—please refer to the **File / Save project** chapter.

***Protected mode in PG4UWMC***
Administrator and Operator Modes in PG4UWMC (formerly Protected Mode)

By default, PG4UWMC runs in **Administrator Mode**, which provides full access to all operations and settings. In a production environment, however, it is often necessary to block certain commands to prevent users from modifying critical configurations. **Operator Mode** is designed for this purpose.
The Operator Mode in PG4UWMC is very similar to the Protected Mode in PG4UW, with two key differences:

- It can only be activated via a menu command and cannot be triggered by a project file.
- The Operator Mode settings are saved to the PG4UWMC configuration (.ini) file upon closing the application. This ensures that the software automatically resumes in the previously selected mode the next time it is launched.

The menu command **Options / Switch to Operator Mode...** activates Operator Mode in PG4UWMC. After selecting this command, a password dialogue will appear. You must enter the password twice to confirm it; once successfully confirmed, the program will switch to **Operator Mode..**



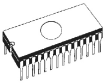

Checkbox **Keep "Load project" operation allowed** is set to inactive state by default - it means the Load project operation button and menu will be disabled when Operator mode is active. If the option is enabled (checked), the Load project operation button and menu will be allowed in Operator mode.

Command **"Load project"** in Operator Mode can also be disabled automatically, by loading of protected project file that has set Protected mode option not to allow another project load. To enable Load project command again, Administrator Mode of PG4UWMC must be entered.

The menu command **Options / Switch to Operator Mode...** activates Operator Mode in PG4UWMC. After selecting this command, a password dialogue will appear. You must enter the password twice to confirm it; once successfully confirmed, the program will switch to **Operator Mode**.

When **Administrator Mode** is active, the "Administrator Mode" label is displayed in the top-right corner of the Programmers Activity Log. Similarly, the "Operator Mode" label is displayed when **Operator Mode** is active.

Note: Occasionally, when switching from Operator Mode back to Administrator Mode, certain commands (such as "Load project") may remain disabled. To resolve this, simply click the **Stop ALL** button.



When loading a project (**File / Load project**), any restrictions defined within a protected project—such as default operations—will be applied, and device operations in PG4UWMC will be enabled or disabled accordingly. This can also affect the **Load project** command itself.

Please note that these restrictions only apply when the **Use Site #1 project for all Sites** option is selected. If this option is turned off, each site will use its own project file, and any menu blocking or operation protection settings forced by the project files will be ignored.

Multi-projects

**Multi-project** is a specialized feature that allows you to execute any sequence of operations on any device, based on information saved within both the sub-projects and the multi-project itself.

In practice, using Multi-projects allows you to:

- Easily program multi-chip devices
- Configure and run any **custom sequence** of operations (e.g., Program + Verify + Verify + Verify) on a single device.

Basic terms related to Multi-project:

- A **Multi-project** file is a specialized file that stores all relevant data for multiple projects. It can contain one or more individual projects, which are also referred to as **sub-projects**.
- A **sub-project** is a standard project file that was integrated into a multi-project file during the build process.
- A **project file** is a specialized file type that integrates buffer data, device operation settings, advanced options, and essential safety features. It serves as a comprehensive set of instructions for the device, allowing saved operations to be reloaded and repeated with exact precision at any time.
- A **Multi-chip device** consists of two or more independent chips—either of the same or different types—housed within a single package.
- A **Sub-device** is an individual component within a multi-chip device. It can be selected from the PG4UW device list, allowing you to work with that specific chip independently. You can define, test, and save project files tailored to each individual chip.
- A **Master device** is a multi-chip unit composed of multiple sub-devices. Like individual chips, the master device can be selected from the **PG4UW device list**. Once selected, you can use the Multi-project Wizard to assemble a multi-project file from individual project files, which can then be saved, loaded, or executed. Note that a master device is not defined if the multi-project is created using single-chip devices.
- A **device operation** is any task that can be executed via the menu, a toolbar button, or a remote command (such as Blank, Read, Verify, Program, or Erase). Certain operations, specifically **Program** and **Erase**, may include embedded sub-operations that can be configured through the **Device Options** menu.
- The **Multi-project Wizard** is an assistant designed to help build multi-project files. It allows users to select individual projects and save them into a single multi-project file, a process known as **multi-project file building**. Additionally, the wizard can initiate device operations based on the specific projects or sub-devices included in the file.
- A **Multi-project checksum** includes the checksums of all data and settings from every sub-project saved within a multi-project file. This checksum serves as a unique ID for each file, ensuring that no two multi-project files are identical. Even if you save the same multi-project

multiple times without changing its content, the multi-project checksum will remain unique for each instance.

The **Multi-project Wizard** is used to create a multi-project file, which is required for multi-project device operations. This file contains individual sub-projects associated with the sub-devices (chips) of a master device. The wizard provides the following main functions:

- Select sub-projects and build the final multi-project file
- Loading existing Multi-project files ***1**
- Initiate the device operation for the most recent multi-project

****1 Note:** An existing multi-project file can be loaded through the main PG4UW menu via **File / Load project**, or within the **Multi-project Wizard** using the **Load multi-prj** command.*

The **Multi-project Wizard contains the following controls:**

- The **Load multi-prj** button is used to load an existing multi-project file.
- The **Build Multi-project** button is used to create a new multi-project file using the projects listed in the **Sub-projects** table.
- **Table 1: Sub-projects** contains a list of all projects included in the current multi-project.
- The **Add project** button is used to add one or more new project files to the list in Table 1.
- The **Remove project** button is used to remove a selected project file from the list in Table 1.
- The **Move up** and **Move down** buttons are used to shift a selected project in Table 1 one position higher or lower. Projects are processed in the specified sequence, starting with the top-most entry (#1).
- The **Help** button displays this help documentation.
- The **Blank, Verify, Program, Erase, and Run** device operation buttons are used to perform the selected operation on all chips (sub-devices) listed in the **Sub-projects** table.
  - In **Multi operation mode**, only one type of operation can be run at a time, applying the same action to every sub-device.
  - In **One operation mode**, either a single operation can be run across all sub-devices, or each sub-project can run its own specific operation, depending on the projects within the multi-project.

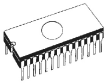
**The following two basic actions must be performed when using a multi-project to program multi-chip devices (the process is similar for single-chip devices):**

- Creating or building the multi-project file.
- Using a multi-project to run device operations.

**Building a multi-project or multi-project file**

The following steps are recommended when creating a multi-project file:

- Create "classic" projects, with one project for each sub-device of the multi-chip device. These projects are created in the same way as those for generic devices:
  - Select the sub-device that corresponds to the required chip of the multi-chip device *1
  - Set the device parameters and settings, then load the required device data to the buffer using the **Load file** command in PG4UW.
  - Optionally test the device operation by running it on the device.
  - If everything is correct, the project file can be created using the **Save project** command.
- Select the master multi-chip device for which the multi-project will be used. After the multi-chip device is selected, the **Multi-project Wizard** opens automatically.
- In the **Multi-project Wizard**, add the necessary projects using the **Add project** button. Each project represents a single sub-device of the multi-chip device.



- Once you have finished selecting the sub-projects, use the **Build Multi-project** button to create the final multi-project file. The program will prompt you for a file name. The resulting multi-project file will contain all sub-projects listed in Table 1: **Sub-projects**.

**Note:** *It is possible to create a multi-project from any standard project files, meaning association with a master device is not mandatory. The user is responsible for ensuring the correct combination of sub-devices and sub-projects within a single multi-project. This feature is particularly useful for ISP programming of devices in a JTAG chain that have different defined projects.*

The Multi-project Wizard can be opened by performing one of the following actions*:*

- *Select the master multi-chip device from the **Select Device** dialog in PG4UW.*
- *Load a previously created multi-project file*
- *Open the **Multi-project Wizard** dialogue directly from the PG4UW menu via **Options / Multi-project Wizard***

**1 Convention for Master-device and Sub-device part names in the PG4UW device list:*
*Master-device: Multichip_original_part_name [package_type]*
*Sub-devices: Multichip_original_part_name [package_type] (part1)*
*Multichip_original_part_name [package_type] (part2)*
*...*
*Multichip_original_part_name [package_type] (part n)*
*Example:*
*Master-device: TV0057A002CAGD [FBGA107]*
*Sub-devices #1 TV0057A002CAGD [FBGA107] (NAND)*
*#2 TV0057A002CAGD [FBGA107] (NOR)*

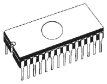
**Using a multi-project to run device operations**

A typical workflow for using an existing multi-project file follows this order.

For single programming in PG4UW:

- Load the created multi-project using the **File / Load project** menu command in the PG4UW main window or the **Load multi-prj** button in the Multi-project Wizard. Once the multi-project is successfully loaded, the Multi-project Wizard will open automatically.
- In the Wizard, run the desired device operation by using one of the available buttons (Blank, Verify, Program, or Erase), with the **Program** operation being the most common choice. The selected operation is executed as a sequence of loading sub-projects and programming each sub-device defined in the multi-project. This is the primary purpose of a multi-project: to automate the sequence of operations for every chip in a multi-chip device. A side effect of this process is that the progress indicators reset to 0% at the start of each sub-device operation, which causes the progress bar to "jump" back to the beginning multiple times during the overall process.
- Once all sub-devices have been programmed or an error occurs, the standard **Repeat** dialog is displayed. You can then remove the programmed device from the socket and insert a new one. Pressing the **Yes** button in the dialog, or the **YES!** button on the programmer, will restart the multi-chip device programming sequence.
- If the **Automatic YES!** function is enabled, the Repeat dialog will not be displayed once the device operation is complete. Instead, an **Automatic YES!** window will appear, showing the status of the programmer socket and a prompt to replace the programmed device with a new

one. Once the new device is inserted, the multi-chip device operation sequence will start automatically. For more details on this feature, please refer to **Programmer / Automatic YES!**.



For multiprogramming using PG4UWMC or a standalone programmer:

- Load the multi-project through the **Load project** menu.
- Run the desired device operation using one of the available buttons (Blank, Verify, Program, or Erase), though **Program** is most commonly used. The selected operation executes as a sequence of loading sub-projects and programming each sub-device defined in the multi-project. Because of this, progress indicators reset to zero at the start of each sub-device operation, causing the progress bar to "jump" back to the beginning several times during the multi-chip process.
- Once all sub-devices have been programmed or an error occurs, the operation results are displayed in PG4UWMC. You can then remove the programmed device from the socket and insert a new one. Pressing the operation button for the Site, or the **YES!** button on the programmer Site, will restart the multi-chip device programming sequence.
- If the **Automatic YES!** function is enabled, the device operation sequence restarts automatically once the programmed device is removed and a new one is inserted into the socket. For further details on this feature, please refer to **Programmer / Automatic YES!**.

**Notes:**

- ***Serialization*** *is not supported in multiprogramming mode; only single programming supports serialization.*
- *The* ***count-down function*** *is currently not supported.*

Options / Save options

This command saves all currently supported settings, even if auto-save is disabled. The following options are saved:

- The settings of the most recently used programmer
- The settings of the most recently selected device, including device options and special device options, if available (note that device data buffers are not saved)
- The list of the 20 most recently selected devices and their current device option settings (data buffers are not saved)
- Options under the **Options** menu, including general options and the current configuration of the main window toolbars
- The **recent files/projects list**, which includes the names of up to 10 recently used files and up to 10 recently used projects
- The position and size of the main program window and various sub-windows

# *Help*

The **Help** menu contains commands that allow you to view supported devices and programmers, as well as information regarding the software version.

Pressing **F1** opens the Help documentation. If you highlight a menu item and press **F1**, you will access context-sensitive help. Please note that if PG4UW is currently executing an operation with the programmer, the **F1** key will not respond.

The following Help items are highlighted:

- words describing the keys referred to by the current Help
- all other significant words
- current cross-references; click on this cross-reference to obtain further information.

Detailed information on individual menu commands can be found in the integrated on-line Help.

**Note:** *The information provided in this manual is intended to be accurate at the time of release; however, we continuously improve all our products. Please consult the manual on* **www.dataman.com/pages/downloads**.

*Because the help system is continuously updated alongside the control program, it may contain information not included in this manual.*

Help / Supported devices

This command displays a list of all devices supported by at least one type of programmer. It is particularly useful for users who need to confirm if a specific device is supported by any of the available programmer models.

Help / Supported programmers

This command displays information about the programmers supported by this program.

Help / Device list (current programmer)

This command generates a list of all devices supported by the current programmer and saves it as both a text file (?????DEV.txt) and an HTML file (?????DEV.htm). These files are saved in the directory from which the control program is run. The "????? " placeholders are replaced by the abbreviated name of the specific programmer for which the device list was generated.

Help / Device list (all programmers)

This command generates device lists for all programmers and saves them as text files (?????DEV.TXT) and HTML files (?????DEV.HTM) in the directory from which the control program is running. The placeholders (?????) are replaced by the abbreviated names of the programmers for which the lists were generated.

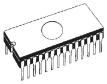
Note: *The control program clears all information about the current device after this command is executed. Please reselect the desired device using any of the selection methods in the* ***DEVICE*** *menu.*

Help / Device list (cross reference)

This command generates a cross-reference list of all devices supported by the programmers available on the market and compatible with this control program. The resulting list is in HTML format and consists of the following files:

- one main HTML file **TOP_DEV.htm** with the supported device manufacturers listed
- partial HTML files with a list of the supported devices for each device manufacturer

The main HTML file is placed in the directory where the programmer control program is located.



Partial HTML files are placed in a subdirectory named **DEV_HTML**, which is located in the same directory as the programmer control program.

Programmer / Create problem report

The **Create problem report** command writes detailed diagnostic information to the Log window and generates a problem report file. This file is a standard text format that can be opened and read with any text editor.

A problem report file is essential when a complex error occurs that cannot be resolved independently. In such cases, providing only a brief description of the issue may not be enough for the manufacturer to identify the cause. Sending the problem report file alongside your inquiry allows the manufacturer to locate the source of the error and resolve it more quickly.

There are two options/buttons for **Create problem report**:

- The **Yes, last days logs** button creates a smaller Problem report (.zip file) containing log files from the past week. All log files included in the .zip are truncated to a maximum size of approximately 1MB.
- The **Yes, all days logs available** button creates a complete problem report by including all available PG4UW/PG4UWMC log files in the .zip file without truncating them. While the resulting file may be significantly larger than the "last days logs" version, it provides a comprehensive overview of all operations.

About

When you select the **Info** command from the menu, a window appears displaying copyright and version information.

# PG4UWMC

The **PG4UWMC** software is designed for fully parallel, concurrent multi-programming across multiple programmers or multi-programming-capable devices connected via USB to one or more computers.

It is specifically tailored for monitoring high-volume production. The operator-friendly interface combines powerful functionality with ease of use, providing a clear overview of essential activities and results without distracting the operator with unnecessary details.

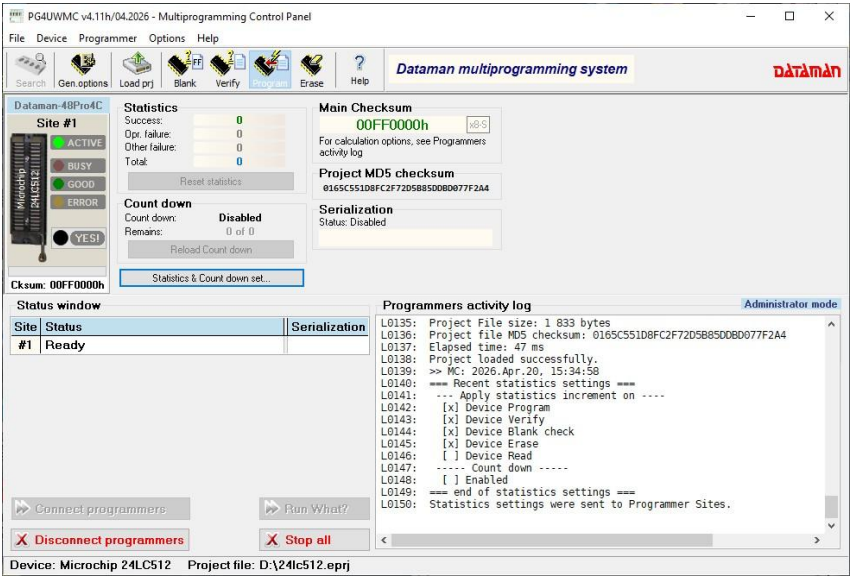
**PG4UWMC** uses a project file to manage the multi-programming system. This file contains user data, programming setups, chip configuration, and automated command sequences. This approach minimises operator error, as engineers typically create and verify the project file before providing it to the operator. Additionally, an optional **Operator mode** can be enabled to prevent unintended changes. Each chip can also be programmed with unique data, such as serial numbers, configuration settings, or calibration info.

The PG4UWMC program consists of the following main windows:

- main window
- settings dialog window
- "Search for Programmers" dialog window

For more details regarding multiprogramming and the use of PG4UWMC, please refer to the user manual for any of our USB-interfaced concurrent multiprogramming systems.

A basic description of the primary components of PG4UWMC




PG4UWMC main window

The main window of the PG4UWMC program consists of the following parts:

**Menu and tool buttons**
The menu and tool buttons provide access to most PG4UWMC functions.

**Tool button Settings**
This button is used to open the **PG4UWMC Settings** dialogue, which is described in detail below.

**Panels Site #1, Site #2...**
These panels are used to inform the user about:
- Programmer Site that is selected
- Programmer Site activity
- current device operation status and/or result

Each panel also includes a **Run** or **YES!** button, which is used to initiate the device operation.

**Statistics Box**
The **Statistics** box provides information on the total number of programmed devices, as well as the count of successful and failed units.

Note: *The **Reset statistics** button remains disabled while any action is in progress on the sites. To stop all active site operations, press the **Stop All** button.*

**Checksum**
The **Checksum** displays a simple checksum of the data loaded from the current project file.

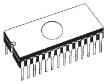
The **Status window** panel
The **Status window** panel provides information on the current state of each site. Possible states include:

| | |
|---|---|
| **Blank** | Site is not active |
| **Ready** | Site is active and ready to work. Programmer is connected. No device operation is running. |
| **and other information** | the currently running device operation, result, programmer connection state and so on. |

**Log window** on the right side of the Status window
The **Log window** displays information regarding the connection and disconnection of programmers, device operation results, and other relevant status updates.

**Connect programmers Button**
The button is used to connect all selected programmer sites by launching the PG4UW control program for each one. This is typically the first step taken after starting PG4UWMC.

**Disconnect programmers Button**
The button is used to disconnect all connected programmer sites and close their respective control programs. This action can only be performed if no device operations are currently running on any of the connected programmers.



**Run <operation> Button**
The button is used to simultaneously initiate device operations on all connected programmers. The available operations include **Program**, **Verify**, **Blank check**, and **Erase**.

**Stop ALL Button**
The button is used to stop all device operations currently running on connected programmer sites.

**Help Button**
The button is used to display this help.

**Start remote control of programmer(s) Button**
This button is available exclusively for automated programmers, and only when the option **Use Site #1 project for all Sites** is enabled in the **PG4UWMC Settings (Multiprogramming / Project options)**. It is used to initiate remote control of the PG4UWMC interface.

Once **PG4UWMC Remote Control** begins, the system detects modules on the sites based on the loaded projects, and a window titled **PG4UWMC Remote Control** appears.

This window provides operation progress updates and an emergency stop button. While the manufacturer's logo must remain visible, all other elements of this window can be customised. To access the options menu, please right-click on the logo.

File / Load project

This option is used to load a project file, which includes the saved device configuration, buffer data, and user interface settings.
The standard **Load project** dialog includes an additional **Project description** window at the bottom. This window displays information about the currently selected project file, including:

- The manufacturer and name of the primary device selected in the project.
- The date and time the project was created.
- A user-written description (any text, though typically the author's name and relevant notes).

*Note: For projects with serialization enabled.*

The following procedure is used to read serialization from the project file:

- Serialization settings from the project are accepted
- An additional serialization file search is performed. If the file is found, it will be read, and its serialization settings will be applied. Because additional serialization files are always associated with a specific project, their settings will take priority and override the original project serialization settings.

The name of the additional serialization file is derived from the project file name by adding a **.sn** extension. This file is always located in the **serialization** subdirectory within the control program's directory.

**Example:**

*Project file name: my_work.prj*
*Control program's directory: c:\Program Files\Programmer*

*The additional serialization file will be:*
*c:\Program Files\Programmer\serialization\my_work.prj.sn*

An additional serialization file is created and updated after each successful device programming operation. The only requirement for generating this file is to load a project with serialization enabled.

The **File / Save project** command deletes any existing additional serialization file associated with the project being saved.

**Enter job identification dialog**

The dialog will be shown when loading protected project files.
It contains two editable fields:

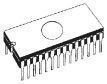
- **Operator identification** is used to identify the programmer's operator. The Operator ID must be at least three characters long. Because this is a mandatory parameter for creating a Job Report for a protected project, the user must enter an identification value.
- **Enter Job ID** - identification of the current job.

**Note:** *The **Enter job identification** dialog is not a password prompt. The **Operator identification** and **Job ID** values are for informational purposes only and will be included in the Job Report. These fields are unrelated to protected or encrypted project passwords.*

File / Make Job report

A **Job Report** provides a summary of the operations recently performed on a device. A "job" is tied to a specific project file, covering all activity from the moment the project is loaded until a new project is opened or the PG4UW program is closed.

A **Job report** contains the following information:
- project name
- project date
- Protected mode status
- PG4UW software version
- programmer type and serial number
- job execution start time (the moment the **Load project** operation was performed)
- job execution end time (the moment the **Job Report** was created)
- device name
- device type
- checksum
- device operation options
- serialization information
- statistics information



The **Job report** is generated in the following cases:
- command Load project is selected
- closing or disconnecting programmer sites is selected
- closing the PG4UW program
- device Count down counter reaches 0 (finished status)
- manually by the user, when the "File / Job Report" menu is used

The **Job Report** is generated for the currently loaded project file only if the **Total** statistics value is greater than zero. This means that at least one device operation, such as program or verify, must be performed.

File / Exit without save

This command deallocates the heap, cancels the buffer on disk (if it exists), and returns to the operating system.

File / Exit and save

This command deallocates the heap, cancels the buffer on the disk (if it exists), saves the current settings of recently selected devices to the disk, and returns to the operating system.

Device / Select device & Create project

You must close **PG4UWMC** before starting a single instance of **PG4UW** in Engineering mode, which is used to select a device and create or save a project file. For further details, refer to the PG4UW software help sections: **Device / Select device** and **File / Save project**.

Device / Blank check

This command performs a **device blank check**. The control program reports the outcome via messages in the **INFO** and **LOG** windows.

The **Device / Device options / Operation options** menu command in PG4UW allows you to customize the available operation settings when creating a project.

Device / Verify

This command compares the entire content of the device—including special areas and settings—with the data stored in the PC or the programmer. The control program reports the results of the verify operation in the **Info window** and the **Programmer activity log**.

The **Device / Device options / Operation options** menu command in PG4UW allows you to customise available operation settings when creating a project.

**Notes:**
- *The Verify operation compares the entire chip's content with the data in the software. Because of this, an incompletely programmed chip might pass the verification check immediately following programming but fail during a standalone Verify operation.*
- *Verify may also report errors on protected devices where read protection is active.*
- *Generally, "**whole device**" refers to the range between the **Device start** and **Device end** addresses defined in the **Device / Device options / Operation options** dialog. Note that not all devices allow for customized start/end addresses. For certain devices (such as NAND FLASH), where sector or page counts are adjustable, "whole device" refers to the range specified by those specific settings.*

Device / Program

This command initiates **device programming**. The control program reports the outcome via messages in the **INFO** and **LOG** windows.

The **Device / Device options / Operation options** menu command in PG4UW allows you to customise the specific areas to be programmed and configure other operation settings while creating a project.

Device / Erase

This command initiates a **device erase**. The control program reports the outcome via messages in the **INFO** and **LOG** windows.

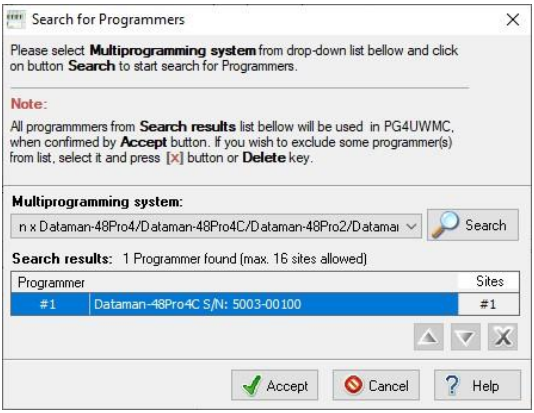
The **Device / Device options / Operation options** menu command in PG4UW allows you to customise available operation settings when creating a project.

If the device (chip) does not support a dedicated erase verify command, a **blank check** operation is automatically performed after the erase to confirm the action was successful.

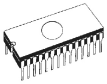
Programmer / Search for Programmers

This command is used to search for and manage multiprogramming systems connected to a single PC. You can connect several single-site programmers of the same type or a multiprogramming system with up to 64 programming sites to one computer.

- Select multiprogramming system connected to your PC.
- Press search button.
- Use the **arrow buttons** to organize the discovered programmers and assign them to specific programming sites.
- Press the Accept button




**Note:** *Programmers highlighted in **red** indicate that certain expected sites could not be found. These sites are listed in the **"Not found"** column, which remains hidden if all sites are accounted for.*



Programmer / Credit box info

The **Credit Box Info** menu item provides all necessary details regarding any Credit Boxes attached to the PC. This includes the type, serial number, activation status, and the number of available credits. If multiple Credit Boxes are connected, the software will also display the total number of available credits across all devices.

The main window also displays the availability of Credit Boxes and the number of credits they contain. You can find this information on the **Credit Box** button located below the **Statistics** and **Count Down** sections. This button appears dynamically whenever at least one Credit Box is present and a device requiring Paid ISP support is selected.

The bar graph at the bottom of the Credit Box button indicates the credit status of the attached boxes as follows:

| | |
|---|---|
| green bargraph | 85% of total credits available |
| yellow bargraph | 50% of total credits available |
| red bargraph | 10%, of total credits available |
| no bargraph | 0% of total credits available (depleted) |

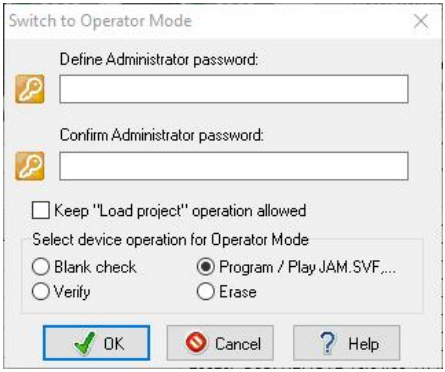
Information about available credits is updated periodically, or while the Credit Box Info window is open.

Options / Switch to operator mode (administrator mode)

By default, the PG4UWMC program is set to Administrator mode, which means no operations are blocked for the user. In a production environment, however, it is often appropriate to block certain menu commands to ensure that users do not modify important program settings or configurations. **Operator mode** is designed for this purpose.

The **Operator mode** in PG4UWMC is very similar to the **Protected mode** in the PG4UW program. One difference is that **Operator mode** is activated through a menu command rather than a project file. Additionally, **Operator mode** settings for PG4UWMC are saved to the configuration .ini file when the program is closed. When PG4UWMC is restarted, it applies the most recent **Operator mode** settings from that .ini file. For more information regarding the Protected mode in the single PG4UW application, please refer to the **Options** and **Protected mode** section.

The menu command **Options / Switch to Operator mode...** allows you to use **Operator mode** in PG4UWMC. After you select this menu item, the **Switch to Operator mode** dialog will appear. The user must enter a password twice to confirm it is correct. Once the password is confirmed, the program will switch to **Operator mode**.




The **Keep "Load project" operation allowed** option is inactive by default. This means the Load project button and menu command are disabled when Operator mode is active. If this option is enabled, the Load project button and menu remain available in Operator mode.

The **Load project** command can also be disabled automatically in Operator mode by loading a protected project file that restricts loading other projects. To re-enable the Load project command, you must return PG4UWMC to Administrator mode.

To switch from Operator mode back to Administrator mode, use the menu command "Options / Switch to Administrator Mode..." A "Password required" dialog will appear. The user must enter the same password used when switching to Operator mode.

When Administrator mode is active, the label Administrator mode is visible in the right top corner of the **Programmers activity log**.

When Operator mode is active, the label Operator mode is visible in the right top corner of the **Programmers activity log**.

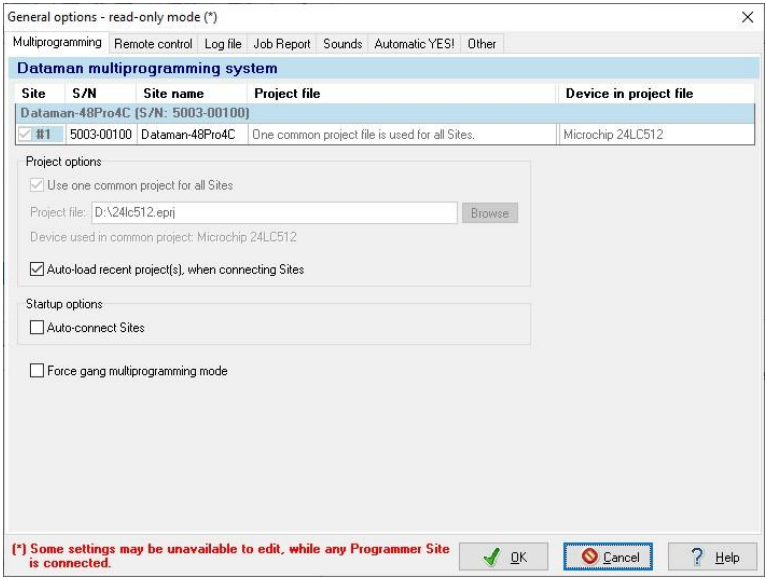
**Notes:** *Sometimes, when switching from Operator mode to Administrator mode, certain commands such as "Load project" may remain disabled. This can be resolved by clicking the "Stop ALL" button.*

*If a project is protected and has a defined default operation, those restrictions are applied when the project is loaded via the "File / Load project" menu. Device operations in PG4UWMC will then be enabled or disabled accordingly. The "Load project" command itself may also be affected. These restrictions only apply when the "Use Site #1 project for all Sites" option is selected. If this option is turned off, each site uses its own project file, and any menu operation blocks forced by project settings are ignored.*

Options / General options

The PG4UWMC General options dialog is used to set or display configuration options for PG4UWMC.

**Multiprogramming panel**




The table contains the following columns:

- **Sites** column contains checkboxes with Site numbers #1, #2, #3, #4 used to enable/disable use of individual Programmer Site with the specified Site number
- **Serial number S/N** column contains information about the serial numbers for the Programmer Sites
- **Site name** column
- **Project file** column contains edit lines for Project #1 through Project #4. These allow you to set individual projects to be loaded automatically after each PG4UW instance runs. You can enter project file names manually or select them via the **Select project file** dialog. This dialog is opened by clicking the "..." button located on the right side of each project edit line. If a project name edit line is left blank, no automatic project load will occur.
- **Device in project file** column

The checkbox **Use one common project for all Sites** is located below the Sites numbers and Projects table.

If you need to program the same device types with the same data, you should check this box.

- If the checkbox is checked, a single common project file will be used for all programmer sites. In this mode, all sites use the same shared buffer of project data and program the same device type.
- If the checkbox is not selected, each site will use the specific project file named in the Project File column of the site table. In this mode, each site uses its own buffer of project data. This allows different data to be programmed to different device types simultaneously at each site.

**Auto-load recent project(s), when connecting Sites** checkbox

When this option is selected, the most recent projects are automatically loaded every time the programmer sites and their corresponding PG4UW instances are started and connected to PG4UWMC. There are two ways to connect sites to PG4UWMC:

- The connection occurs automatically after PG4UWMC starts. For more details, please see the information regarding the **Auto-connect Sites** checkbox below
- Alternatively, you can connect manually by clicking the **Connect programmers** button in the main PG4UWMC window.

The **Auto-connect Sites** checkbox ensures that sites are automatically connected when PG4UWMC starts. This eliminates the need for the user to press the **Connect programmers** button after launching the application.

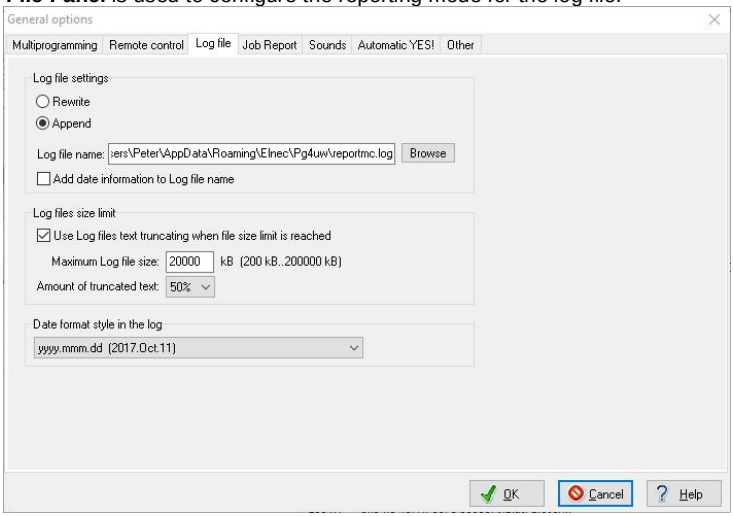
**Force gang multiprogramming mode** checkbox

The standard multiprogramming operation for our multiprogrammers is **concurrent mode**. In this mode, each programming site works independently, allowing the operator to reload a programmed device while other sites are still running. In contrast, **gang multiprogramming** mode starts a predefined operation on all programming sites simultaneously when any **YES!** button is pressed.

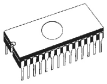
**Notes:**

- *Works for all active (present and enabled) sites.*
- *Start is unavailable while any site is busy.*
- *In this mode Automatic YES! is disabled.*

The **Log File Panel** is used to configure the reporting mode for the log file.




The log file is a text file that contains information about the PG4UWMC control program operation flow, including details on loading project files, device operation types, and operation results. The multiprogramming system generates several log files: one main log file for the PG4UWMC program and individual log files for each active programmer site. Each site



maintains its own log file. The name of a site log file uses the same prefix as the name specified in the Log File edit box, followed by the site number in the format_#<Snum>.

**Example:**
*The Log file name specified by the user is: "report.log". Then the names of the Log files will be:*
*PG4UWMC main Log file name - "report.log"*
*Site's #1 Log file name - "report_#1.log"*
*Site's #2 Log file name - "report_#2.log"*
*Site's #3 Log file name - "report_#3.log"*
*and so on...*

The following options can be set for the Log file creation

- The "Append Log File" option enables log file usage. The system creates this file after the first time you restart PG4UWMC. For all subsequent restarts, the software preserves the existing log file and adds new data to it.
- The "Rewrite Log File" option enables log file usage. The system creates a log file after the first time you restart PG4UWMC. For every subsequent restart, the software deletes the previous data and creates a new log file.

The "Add date information to Log file name" checkbox allows you to include the date in the filename specified in the edit box. When you select this option, the program automatically adds the current date to your chosen filename according to specific rules:

If the user specified log file name has the format:
**<user_log_file_name>.<log_file_extension>**

The name with the added date will be:
**<user_log_file_name><-yyyy-mmm-dd>.<log_file_extension>**
The new part representing the date consists of yyyy - year, mmm - month and dd - day.

**Example:** *User specifies the Log file name: c:\logs\myfile.log*

*The final log file name with added date will look like this (lets use the date November, 7th, 2026):*
*c:\logs\myfile-2026-nov-07.log*

If you want the log file name to appear without a prefix before the date information, you can format it as:
**<log_file_extension>  - dot is the first in file name**

**Example:** The u*ser specifies the Log file name: c:\logs\.log*

*The final log file name with the added date will look like this (lets use the date November, 7th, 2026):*
*c:\logs\2026-nov-07.log*

**Advanced options for Log file size limits are also available:**

- The "Use Log file text truncating when file size limit is reached" option enables a size limit for the log file. When this is selected, the program truncates a portion of the text once the file reaches a specified size. If you deselect this option, the log file size remains unlimited, restricted only by available disk space.

- The option **Maximum Log file size** specifies the maximum size of the Log file in kB
- The option **Amount of truncated text** specifies the percentage of the Log file text which will be truncated after the Maximum Log file size is reached. A higher value means more text will be truncated (removed) from Log file.

**Note:** *Lines that start with '+' are shown in the log file, but not in the log on the screen, to keep a better overview of the on-screen log.*

**Common information:**
The **Programmer Site Index** is a whole number from 1 to 8 that uniquely identifies each active Programmer Site.
**Serial number of Programmer Site** uniquely identifies the specific programmer or programmer site in use. The application searches all programmers connected via USB until it locates the site with the matching serial number. It ignores all other programmers or sites with different serial numbers. If the software cannot find the requested site, it sets that site to Demo mode with a "Not found" status.

On one computer, 8 Programmer Sites can be run at the same time.

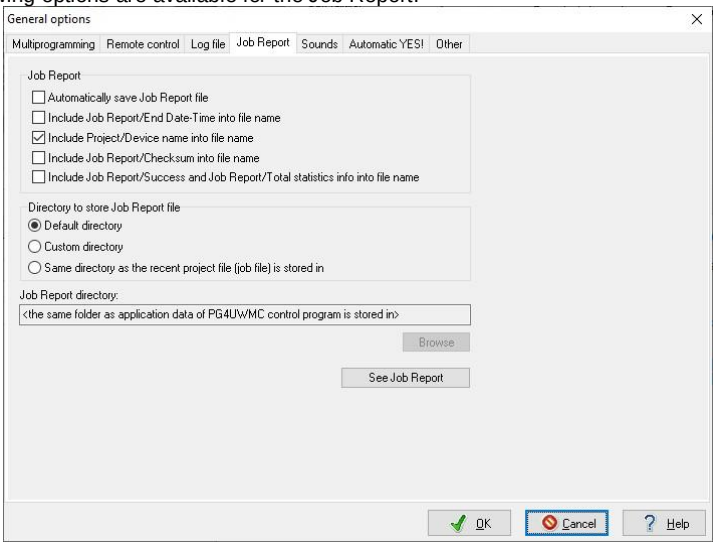
The **Job Report Panel** settings determine how the Job Report is utilized. A Job Report provides a summary of the operations recently performed on a device. A job is associated with a project file and encompasses all operations from the time a project is loaded until a new project is opened or the PG4UWMC program is closed.

A **Job Report** contains the following information:
- project name
- project date
- Protected mode status
- PG4UWMC software version
- programmer type and serial number
- job execution start time (the moment the **Load project** operation was performed)
- job execution end time (the moment the **Job Report** was created)
- device name
- device type
- checksum
- device operation options
- serialization information
- statistics information

The **Job report** is generated in the following cases:
- command Load project is selected
- closing or disconnecting programmer sites is selected
- closing the PG4UW program
- device Count down counter reaches 0 (finished status)
- manually by the user, when the "File / Job Report" menu is used

The Job Report is generated for the currently loaded project file only when the Total statistics value is greater than zero.
This means you must perform at least one device operation, such as programming or verifying, for the report to be created.

The Job Report dialog settings are located in the PG4UWMC Settings dialog (found under the Options / Settings menu) within the Job Report tab.
The following options are available for the Job Report:




When the **Automatically save Job Report file** checkbox is selected, the program automatically saves the Job Report to the folder specified in the "Job Report directory" field. The system generates the filename according to the following rules:

*job_report_<ordnum>_<prjname>.jrp*

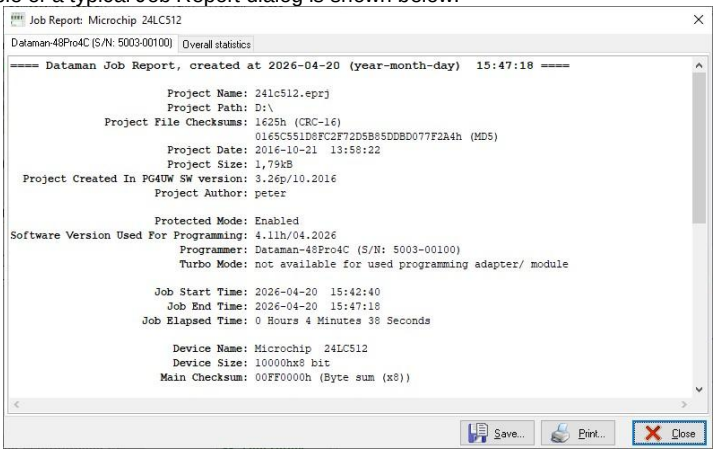
where
*<ordnum>* represents the decimal sequence number of the file. If report files with the same name already exist, the system increments the number for the new report file based on the existing files.
*<prjname>* refers to the name of the most recently used project file, excluding the file extension..
**Example 1:** Let's use the project file c:\myproject.eprj and the directory for the Job Report is set to d:\job_reports\
Where there are no report files present in the Job Report directory.
The final Job Report file name will be:
*d:\job_reports\job_report_000_myproject.jrp*

**Example 2:** Let's use the conditions from Example 1, but assume there is already one report file present.
The name of this file is d:\job_reports\job_report_000_myproject.jrp
The final Job Report file name for the new report will be:
*d:\job_reports\job_report_001_myproject.jrp*

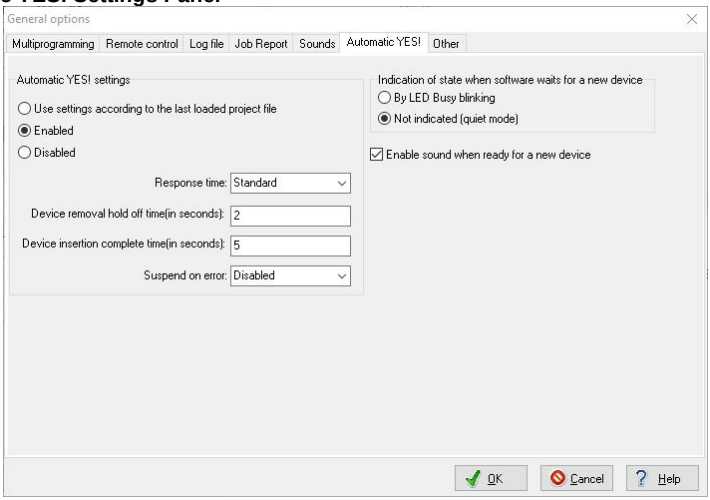
**Note:** *The sequence number within the file name increases by an increment of one*.

When the **Automatically save Job Report file** setting is enabled, no dialog boxes appear during the generation of the Job Report. Provided that no errors occur during the saving process, the system saves the newly generated report to a file without displaying any dialogs or messages).

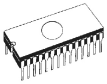
If you deselect the **Automatically save Job Report file** checkbox, PG4UWMC will display the Job Report dialog whenever necessary. Within this dialog, you can choose how to handle the Job Report. If you do not select an operation and instead click the Close button, the software will only write the Job Report to the PG4UWMC Log Window.
An example of a typical Job Report dialog is shown below:




**Automatic YES! Settings Panel**




In this mode, you simply remove the programmed device and insert a new one into the ZIF socket to automatically repeat the previous operation. The program detects the insertion of the



new device and executes the last command without requiring any keystrokes or button clicks. The screen will indicate when a device is inserted into the ZIF socket. You can cancel the repeated operation by pressing the key while the system is waiting for a device to be inserted or removed.

This feature may not be available for all programmer models.

When the **Use settings according to the last loaded project file** option is selected, Automatic YES! functions are determined by the project file settings. One specific item, "Pins of programmer's ZIF excluded from sensing," depends on the programming adapter being used. Since different programmers may use different adapters for the same device, the system will ignore this setting if the adapters do not match. In such cases, the log window will display the following message:
"None connected pins setting was not accepted due to different programming adapter. Please use automatic YES wizard again."
If this occurs, go to the master programming site (if running PG4UWMC with the "Use Site #1 project for all Sites" option) or the specific site that generated the log message. Then, click the "Setting Automatic YES! parameters" button under the **Programmer / Automatic YES!** options menu.

The settings for "Indication of state when software waits for a new device" and "Enable sound when ready for a new device" are not stored in the project file.

**Enabled** - The Automatic YES! function is enabled on all connected programming sites using the parameters specified by PG4UWMC.

**Disabled** - The Automatic YES! function is disabled on all connected programming sites. Use this setting if you prefer to use the YES! button to manually start the next operation for the programmed device.

**Response time** - This setting defines the interval between inserting the chip into the ZIF socket and the start of the selected device operation. If the chip requires more time for positioning within the ZIF socket, select the elongated response time.

**Device removal hold off time** - This value defines the time period between removing a device from the ZIF socket and when the software begins checking the socket for a new insertion. This duration is measured in seconds and must be between 1 and 120, with a default value of 2 seconds

**Device insertion complete time** - This value defines the time within which all pins of the device must be properly inserted after the first pins are detected. This prevents the program from incorrectly identifying a device as improperly seated. This interval is measured in seconds and must be between 1 and 120, with a default value of 5 seconds

**Suspend on error** - This setting determines whether the Automatic YES! function will be temporarily disabled following an error to allow you to view the operation results, or if it will continue without interruption.

Indication of state when the software is waiting for a new device:

**Not indicated (quiet mode)** - Regardless of how many ZIF sockets the programmer has, it does not provide a specific status indication while waiting for a new device to be inserted. After

a device operation, only the "Error" or "OK" LED will light up, depending on the previous result. This LED turns off immediately once the system detects that the device has been removed from the ZIF socket.

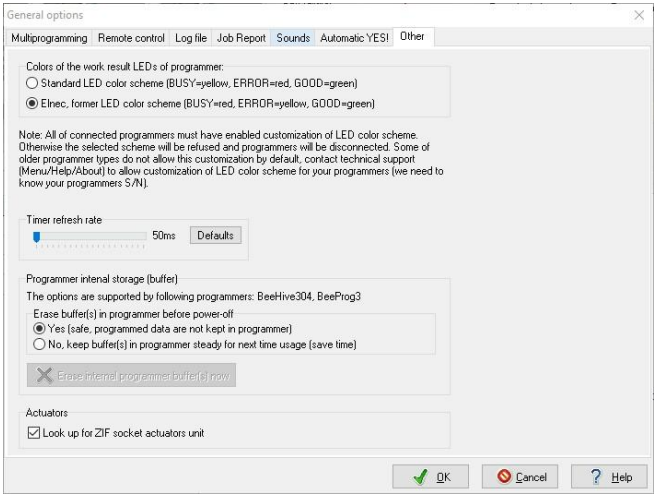
**By LED Busy blinking** - Regardless of the number of ZIF sockets, the programmer indicates it is ready for a new device by flashing the Busy LED. Once an operation finishes, either the OK or Error LED will light up while the Busy LED continues to blink. When the program detects that the device has been removed, the status LED turns off, but the Busy LED continues blinking to show it is ready to repeat the operation. When a new device is detected, the Busy LED stays on steadily while the program waits for all pins to be seated. If the "Device insertion complete time" expires before the device is correctly inserted, the Error LED will light up. Once the device is inserted correctly, the status LED turns off and the next operation begins.

**Enable sound when ready for a new device** - When this option is selected, the software will generate a sound notification once it detects a completely empty ZIF socket and is ready for the next device insertion.

When you select any of the previous options and click the OK button, PG4UWMC sends the chosen settings to all connected programming sites. Similarly, if you configure Automatic YES! parameters on the master site, these settings are distributed to all connected slave sites and to PG4UWMC.

For more details regarding the Automatic YES! feature, please refer to the Programmer / Automatic YES! section.

**Other Panel**
Colors for the programmer's result LEDs:
- Standard LED color scheme (ERROR=red, BUSY=yellow)
- Former LED color scheme (ERROR=yellow, BUSY=red)



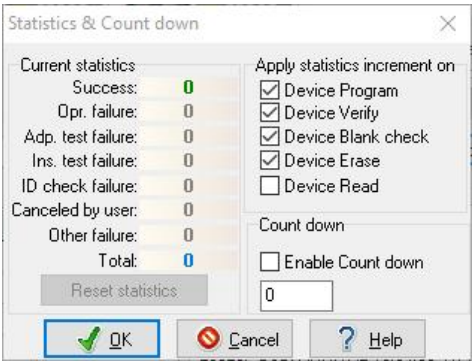
**Note:** *These settings are available only for certain types of programmers. If you cannot see these options in the menu, or if the menu is disabled for editing, your programmer does not support LED colour scheme customization.*

The **Timer refresh rate** defines how frequently the PG4UWMC software requests status updates from active Programmer Sites. This status information includes the current operation type, progress, and results, which the system displays in the PG4UWMC main window. The default refresh rate is 200ms. You can select a shorter interval if you prefer more frequent updates in the Operation panel. However, if system performance slows down, you should choose a higher value to reduce the refresh frequency. While Pentium 4 computers experience almost no performance impact from this setting, choosing a longer interval may be beneficial on slower machines.

**Programmer internal storage (buffer)** Please refer to the help page section for **General options/Buffer**, specifically the **Erase buffer(s) in programmer before power-off** radio group.

Options / Statistics & Count down

This command shows the Statistics & Count down settings.




**Statistics** provides information on the total number of operations performed on a selected device type. If each device undergoes a single operation, such as programming, the number of operations will equal the total number of programmed devices.

The Count Down function tracks the remaining number of operations and devices required to complete a task. After each successful operation, the countdown value decreases. You can define the starting number of devices; once the value reaches zero, the specified quantity is complete, and a notification message will appear.

The Statistics and Count Down section includes the following options:

The check boxes **Program**, **Verify**, **Blank**, **Erase** and **Read** determine which specific operations will trigger an increase in the statistics values.

Any selected and completed device operation will increase the **Total** counter, as well as either the **Success** or **Failure** counter based on the result. A combination of partial tasks is counted

as a single operation. For example, a Read operation that includes a subsequent Verify is treated as one operation, as is a Program operation that includes Erase or Verify steps.

The "Enable Count Down" checkbox toggles the count down function on or off. The edit box next to it allows you to define the initial value where the count down begins.

You can also open the Statistics and Count Down dialog by right-clicking on either the Statistics or Count Down panel and selecting the "Statistics & Count Down" option from the menu.

The Statistics dialog displays seven distinct values: Success, Operational failure, Adapter test failure, Insertion test failure, ID check failure, Cancelled by user, Other failure (software or hardware), and Total.

The meaning of the values are:

| | |
|---|---|
| **Success** | number of operations which were successfully completed |
| **Operational failure** | number of operations which were not successfully completed due to an error of the device |
| **Adapter test failure** | number of operations which were not successfully completed due to an incorrect programming adapter |
| **Insertion test failure** | number of operations which were not successfully completed due to an incorrect position of the device in the programming adapter |
| **ID check failure** | number of operations which were not successfully completed due to an incorrect ID code read from the device |
| **Canceled by user** | number of operations which were not successfully completed due to being canceled by the user |
| **Other failure** | number of operations which were not successfully completed due to a hardware error of the programmer or a control software error |
| **Total** | total number of all operations |

The current statistics values are displayed in the main program window within the **Statistics and Count Down panel**.

**Notes:**
*The Reset statistics button remains disabled while any action is running on the sites. To halt site activity, press the "Stop All" button. Clicking the Reset button in the Statistics panel clears all values to zero. This statistical information is then saved to the Job Summary report.*

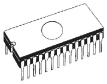
Options / Reset all settings

The **Reset All Settings** dialog restores all PG4UWMC settings to their default state. It also deactivates any active Administrator or Protected modes.

The system prompts you to enter a "Captcha" to confirm the reset. These changes will take effect after you restart PG4UWMC.

# *Help*

The Help menu provides commands for viewing supported devices and programmers, as well as information regarding the software version.



Pressing the **<F1>** key opens the Help menu. If you highlight a menu item and press **<F1>**, you will access context-sensitive help. However, the **<F1>** key will not respond while PG4UW is executing a programmer operation.

The following Help items are highlighted:

- words describing the keys referred to by the current Help
- all other significant words
- current cross-references; click on this cross-reference to obtain further information

Detailed information on individual menu commands can be found in the integrated on-line Help.

**Note:** *Information provided in this manual is intended to be accurate at the moment of release, but we continuously improve all our products. Please consult the manual at* **www.dataman.com/pages/downloads**.

> Since the Help system is updated continuously alongside the control program, it may contain information that is not included in this manual.

Help / Supported programmers

You can view the list of currently supported programmers in PG4UWMC by navigating to the **Help | Supported programmers** menu. Generally, PG4UWMC supports 48-pindrive and 64-pindrive universal programmers with USB or LAN interfaces, as well as all of our USB-connected multiprogramming systems. The software can manage between 1 and 16 programmer sites, where each site represents a single ZIF socket module.

Help / Create problem report

The **Create problem report** command writes detailed diagnostic information to the Log window, which is then used to generate a problem report file. This file can be opened and read with any standard text editor.

A problem report file is essential when a complex issue occurs that a user cannot resolve independently, requiring contact with the manufacturer. In these cases, providing only a brief description may be insufficient to identify the cause. Therefore, we recommend including the problem report file to help the manufacturer locate and resolve the error more quickly

There are two options or buttons available for creating a problem report:

- The "Yes, last days logs" button creates a smaller Problem report in a .zip format containing log files from the previous week. All log files included in this .zip file are truncated to a maximum size of approximately 1MB.
- **Button Yes, all days logs available.** This option creates a complete problem report by including all available PG4UW and PG4UWMC log files in the .zip file without any text truncation. While the resulting .zip file may be significantly larger than the "Last days logs" version, it provides a comprehensive overview of all operations.

About

When you select the Info command from the menu, a window appears that displays the copyright and version information.

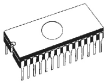
Troubleshooting

**Serial numbers**
To use multiple programmers successfully, you must specify the correct serial number for each one in the Serial Numbers panel. If a serial number field is left empty, the PG4UW application for that programmer site will not start.
When the PG4UWMC application searches for connected hardware in the "Search for programmers" dialog, it detects the serial numbers automatically. You do not need to specify these serial numbers manually, nor are you able to do so.

**Communication error(s) while searching for programmers**
If a communication error occurs, please close all PG4UW applications and the PG4UWMC. Then, restart PG4UWMC and click the "Connect programmers" button to launch the PG4UW applications for each site and reconnect the programmers.

**All programmers are connected correctly but unstable working**
If communication with the programmers is lost during a device operation, such as programming, please close other active programs. This applies especially to software that consumes large amounts of system resources, including multimedia, CAD, and graphics applications

**Note:** *We also recommend using USB ports located on the back of the computer that are connected directly to the motherboard. Ports connected indirectly via a cable may be unreliable when using high-speed USB 2.0 transfer modes. This recommendation applies to both programmers and other peripheral devices.*



# Common notes

# *Maintenance*

We recommend following these instructions and precautions to ensure the long-term reliability of the programmer.

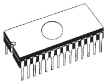
Programmer maintenance depends on the nature and frequency of its use. However, the following general recommendations are typically advised:

- Do not use or store the programmer in dusty environments.
- Humidity accelerates the accumulation of debris and dust within the ZIF socket.
- After finishing a job, protect the programmer's ZIF socket with the provided dust cover.
- Do not expose the programmer to direct sunlight or place it near a heat source.

***Intensive daily use (programming centre, production)***
**Daily Maintenance**
Inspect the programmer's ZIF sockets and socket converters for wear and general condition. Use clean, dry compressed air to remove debris, dust, and grime from the ZIF sockets. Ensure you clean the sockets in both the open and closed positions.
**Weekly Maintenance**
Run a self-test on every programmer or programming module.
**Quarterly Maintenance**
Gently clean the programmer's surface using a soft cloth dampened with isopropyl or technical alcohol. If your programmer supports it, perform a calibration test.

**Daily Use (Development Laboratory or Office)**
**Daily Maintenance**
Cover the programmer's ZIF socket with the provided dust cover after finishing your work. We also recommend protecting the socket converters from dust and grime.
**Weekly Maintenance**
Inspect the programmer's ZIF sockets and socket converters for wear and general condition. Use clean, dry compressed air to remove debris, dust, and grime from the sockets, ensuring you clean them in both the open and closed positions.
**Quarterly Maintenance**
Run a self-test on every programmer or programming module.
**Biannual Maintenance**
Gently clean the surface of the programmer with a soft cloth and isopropyl or technical alcohol. Additionally, perform a calibration test if your programmer supports this feature.

**Occasional Use**
**Daily Maintenance**
After finishing your work, protect the programmer's ZIF socket with the provided dust cover. It is also recommended to protect the ZIF sockets of any socket converters from dust and grime.
**Quarterly Maintenance**
Inspect the programmer's ZIF sockets and socket converters for wear and general condition. Use clean, dry compressed air to remove debris, dust, and grime from the ZIF sockets, cleaning them in both the open and closed positions.
**Biannual Maintenance**
Run a self-test for every programmer or programming module.



**Annual Maintenance**
Gently clean the surface of the programmer using a soft cloth and isopropyl or technical alcohol. If your programmer supports it, perform a calibration test.

***Warning:***
*The programmer's ZIF socket and socket converters are considered consumables. The ZIF socket typically has a lifespan of approximately 25,000 mechanical cycles. Socket converters generally last between 5,000 and 10,000 cycles, though some specialised BGA ZIF sockets can reach about 500,000 cycles. The actual electrical lifetime, or the point at which a socket begins to cause programming failures, is directly influenced by the devices being programmed, the operating environment, and regular maintenance. Avoid touching the ZIF socket contacts, as oil and grime from fingers can lead to errors. You should replace the ZIF socket or socket converter if you notice an increase in programming failures.*
*The warranty does not cover ZIF sockets that are worn or dirty and consequently cause frequent operational failures.*

***Programmer ZIF socket replacement***
The programmer ZIF socket is inserted into a precision socket, making replacement straightforward.
**Procedure:**
Grasp the ZIF socket with both hands from the sides, placing your fingers at the front. Gently push forward and upward, but not all the way. Repeat this slight rocking or "bending" motion several times until the ZIF socket slips out.
To install the new socket, place it over the opening and ensure the leads align with the connector holes. Press down firmly on both the top and bottom sections simultaneously until it is fully seated.

# *Software*

PG4UW is the standard control software for all DATAMAN programmers. While using this program, you may encounter certain features and menu items that are not available for your currently selected programmer modelCommand line parameters PG4UW

We recommend using the special utility **pg4uwcmd.exe** for command-line control of PG4UW. While some parameters can be used directly with **pg4uw.exe** for backward compatibility, **pg4uwcmd.exe** is the superior option. It supports a wider range of commands and can return an ExitCode (or ErrorLevel) to indicate whether a command was successful. For more information on using the controller, please refer to the **Remote command line control of PG4UW** section.
The following command-line parameters can be used directly with **pg4uw.exe**

| | |
|---|---|
| /Prj:<file_name> | This command forces a project to load when the program starts or while it is already running. The variable <file_name> represents the full or relative path and name of the project file |
| /Loadfile:<file_name> | This command forces a file to load when the program starts or while it is already running. The variable <file_name> represents the full or relative path to the file, and the software detects the file format automatically. |
| /Saveproject:<file_name> | This command saves the currently selected device type, buffer contents, and configuration to a project file. Using the |

**/Saveproject:...** command is equivalent to selecting the "Save Project" command within the PG4UW control program.

Please note that Windows file naming conventions must be followed. This means that if a filename contains spaces, the command-line parameter must be enclosed in quotation marks.

**Examples:**
/prj:c:\myfile.eprj
Load the project file with name c:\myfile.eprj.

/loadfile:"c:\filename with spaces.bin"
Load file "c:\filename with spaces.bin" to buffer.

| | |
|---|---|
| /Program[:switch] | This command automatically initiates the "Program device" operation when the program starts or while it is already running. You may also use one of the following optional switches: |
| switch 'noquest' | This command initiates device programming automatically without prompting the user for confirmation |
| switch 'noanyquest' | This command initiates device programming without a confirmation prompt. Once the operation is complete, the program bypasses the "Repeat" dialog and returns directly to the main window. |

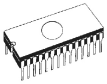
**Examples:**
/Program
/Program:noquest
/Program:noanyquest

| | |
|---|---|
| /Close | This parameter is only effective when used alongside the **/Program** parameter. It causes the program to close automatically once the device programming has successfully finished. |
| /Close: always | This parameter is only effective when used alongside the **/Program** parameter. It causes the program to close automatically once device programming is complete, regardless of whether the operation was successful. |

The following are the basic rules for using executive command-line parameters:

- command line parameters are not case sensitive
- You can use command-line parameters both when first launching the program and while the program is already running.
- If the program is already running, command-line operations will only process if the software is idle and not currently executing a task. The program must be in its default state, meaning the main window is in focus with no modal dialogs, menu commands, or operations active.
- When using multiple command-line parameters simultaneously, the system processes them in the following fixed order:

1. Load project (/Prj:...)
2. Load file (/Load file:...)
3. EPROM/Flash select by ID
4. Program device (/Program[:switch])
5. Close the control program (the /Close parameter is only valid when used with /Program)



**The following command-line parameters are available for starting the PG4UW program in demo mode:**

Demo Demo mode is useful when a programmer device is not available. You can enter Demo mode by clicking the **Demo** button in the **Find Programmer** dialog or by using the **/demo** command-line parameter. The recommended syntax for this parameter is:

**pg4uw.exe /demo /**

In this command, replace with the exact name of the desired programmer as it appears in the PG4UW control software.

Remote command line control of PG4UW

PG4UW can accept a variety of commands from the command line. While you can achieve remote control using these parameters, it is more efficient to use the specialized tool **pg4uwcmd.exe**. The primary advantage is its small file size, which allows **pg4uwcmd.exe** to respond much faster than calling PG4UW directly

Program pg4uwcmd.exe can be used to:
1. start the PG4UW application with the specified command line parameters
2. force command line parameters to an instance of PG4UW that is already running

A key feature of **pg4uwcmd.exe** is its ability to return a specific exit code based on the result of the command-line operation within PG4UW.

If the command-line parameters are processed successfully in PG4UW, the ExitCode (or ErrorLevel) for **pg4uwcmd.exe** is zero. If an error occurs, the ExitCode value will be 1 or higher. You can test the return value of **pg4uwcmd.exe** within batch files

**Following executive command line parameters are available to use with pg4uwcmd.exe**

| | |
|---|---|
| /Prj:<file_name> | This command loads a project file. The parameter <file_name> represents either the full or the relative path and name of the project file. |
| /Loadfile:<file_name> | This command loads a file. The parameter <file_name> represents the full or relative path to the file you wish to load. The software detects the file format automatically. |
| /Program[:switch] | This command automatically initiates the "Program device" operation when the software starts or while it is already running. You can also use one of the following optional switches: |
| switch 'noquest' | This command initiates device programming immediately without prompting the user for confirmation. |
| switch 'noanyquest' | This command initiates device programming without a confirmation prompt. Once the operation is complete, the software bypasses the "Repeat" dialog and returns directly to the main program window. |

**Examples:**
/Program
/Program:noquest
/Program:noanyquest

/Close This parameter is only effective when used with the **/Program** parameter. It causes the PG4UW program to close automatically

once device programming is complete, regardless of whether the operation was successful./Saveproject:<file_name>

This command saves the currently selected device type, buffer contents, and configuration settings to a project file. Using the **/Saveproject...** command is equivalent to selecting "Save Project" within the PG4UW control program.

/writebuffer:ADDR1:B11,B12,B13,B14,...,B1N[::ADDR2:B21,B22,B23,B24,...,B2M]...

The **/writebuffer** command writes a block of bytes to the main PG4UW buffer at a specified address. This command requires at least one data block, while additional blocks are optional. Ensure there are no spaces or tabs within the command.
The buffer address is always defined as a byte address. For a x16 buffer organisation, an address of AAAAx16 must be entered as 2*AAAA (x8) in the command.

**Example 1:**
/writebuffer:7FF800:12,AB,C5,D4,7E,80
Writes 6 Bytes 12H ABH C5H D4H 7EH 80H to buffer at address 7FF800H.
The addressing is structured as follows:
the first Byte at the lowest address

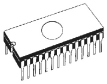
| Buffer Address | Data |
|---|---|
| 7FF800H | 12H |
| 7FF801H | ABH |
| 7FF802H | C5H |
| 7FF803H | D4H |
| 7FF804H | 7EH |
| 7FF805H | 80H |

**Example 2**:
/writebuffer:7FF800:12,AB,C5,D4,7E,80::FF0000:AB,CD,EF,43,21
_____/ _____/
the first block of data the second block of data
Writes two blocks of data to the buffer.
The first block of data - 6 Bytes 12H ABH C5H D4H 7EH 80H are written to buffer at address 7FF800H in the same way as in Example 1.
The second block of data - 5 Bytes ABH CDH EFH 43H 21H are written to buffer at address FF0000H.
The addressing is structured as follows:
the first Byte at the lowest address

| Buffer Address | Data |
|---|---|
| FF0000H | ABH |
| FF0001H | CDH |
| FF0002H | EFH |
| FF0003H | 43H |
| FF0004H | 21H |

/writebufferex:INDEX:ADDR1:B11,B12,B13,B14,...,B1N[::ADDR2:B21,B22,B23,B24,...,B2M]..

The **/writebufferex** command writes a block of bytes to the main PG4UW buffer at a specified address. It functions similarly to the **/writebuffer** command but includes an additional parameter called **INDEX**.



The **INDEX** parameter identifies which buffer will receive the data. The main buffer is assigned index "1," while the first secondary buffer is index "2," and so on. Please note that secondary buffers are only available for specific devices, such as the Microchip PIC16F628. The specific buffer identified by the index corresponds to the order shown in the **View/Edit buffer** dialog. For example, since the Microchip PIC16F628 includes an additional buffer labeled "Data EEPROM," you can write to it by setting the index to 2

**Example 1:**
/writebufferex:1:7FF800:12,AB,C5,D4,7E,80
The command is equivalent to the command
/writebuffer:1:7FF800:12,AB,C5,D4,7E,80
described in section about command /writebuffer.

**Example 2:**
/writebufferex:2:2F:12,AB,C5,D4,7E,80
The command writes 6 Bytes 12H ABH C5H D4H 7EH 80H to secondary buffer with index “2” at address 2FH. The addressing looks as follows:
the first Byte at the lowest address

| Buffer Address | Data |
|---|---|
| 00002FH | 12H |
| 000030H | ABH |
| 000031H | C5H |
| 000032H | D4H |
| 000033H | 7EH |
| 000034H | 80H |

**Basic rules for using of executive command line parameters:**

1. The **pg4uwcmd.exe** program must be located in the same directory as **pg4uw.exe**.
2. If **pg4uw.exe** is not active when **pg4uwcmd.exe** is called, it will start automatically.
3. Command line parameters are not case sensitive.
4. You can use command line parameters both at the initial launch of the program and while it is already running.
5. If the program is currently running, it will only process command line operations when it is idle. The software must be in its basic state, meaning the main window is focused, no modal dialogs are visible, and no menu commands are active.
6. When using multiple parameters together, the program processes them in this strict order:
    - **Step 1:** Load file (/Loadfile:...)
    - **Step 2:** Load project (/Prj:...)
    - **Step 3:** EPROM/FLASH autoselect
    - **Step 4:** Program device (/Program[:switch])
    - **Step 5:** Close the control program (/Close, valid only with the /Program parameter)

**Example 1:**
pg4uwcmd.exe /program:noanyquest /loadfile:c:\empfile.hex
Following operations will be performed:

1. start pg4uw.exe (if not already running)
2. load the file c:\empfile.hex
3. start the program device operation without questions
4. pg4uwcmd.exe is still running and periodically checking the status of pg4uw.exe

5. when device programming finishes, pg4uwcmd.exe closes and returns an ExitCode based on the file loading and programming results in pg4uw.exe. If all operations are successful, pg4uwcmd.exe returns 0; otherwise, it returns a value of 1 or higher..

**Example 2:**
pg4uwcmd.exe /program:noanyquest /prj:c:\emproject.eprj
The operations are the same as in Example 1; just Load file operation is replaced by Load project file c:\emproject.eprj command.

**Example 3:**
Using pg4uwcmd.exe in batch file and testing return code of pg4uwcmd.exe.

```
rem ------- beginning of batch -------------------------
@echo off
rem Call application with wished parameters
pg4uwcmd.exe /program:noanyquest /prj:c:\emproject.eprj
rem Detect result of command line execution
rem Variable ErrorLevel is tested, value 1 or greater means the error occurred
if ErrorLevel 1 goto FAILURE
echo Command line operation was successful
goto BATCHEND
:FAILURE
echo Command line operation error(s)
:BATCHEND
echo.
echo This is end of batch file (or continue)
pause
rem ------- end of batch -------------------------
```

**Example 4:**
Assume the PG4UW control program is running with a device already selected. To load the required data into the buffer and save the configuration to a project file, follow this example. The data file is located at **c:\15001-25001\file_10.bin** and the project will be saved as **c:\projects\project_10.eprj**. To perform this operation, use the following command line parameters:
**pg4uwcmd.exe /loadfile:c:\15001-25001\file_10.bin /saveproject:c:\projects\project_10.eprj**
Once PG4UW receives these commands, it will execute the following procedures:

1. loads data file c:\15001-25001\file_10.bin
2. saves the currently selected device settings and buffer data to the project file c:\projects\project_10.eprj

If the operations are completed successfully, the PG4UWcmd application returns an ExitCode or ErrorLevel value of 0. If errors occur, such as a failure to load the file or save the project, the application returns an ExitCode of 1 or higher.

Note: Before using these commands, ensure that PG4UW is not currently performing a device operation, such as programming. If the software is busy, it will reject the commands and return an error status with an ExitCode of 1 or higher.

Command line parameters PG4UWMC

The Program PG4UWMC supports the following command line parameters:

**/prj:<file_name>**
This command loads a project file, using the <file_name> as either the full or relative path. You can also load a project from the command line by entering the file name directly, without the **/prj:** prefix

**Example:**
*pg4uwmc.exe c:\projects\myproject.eprj*
*Makes load project file "c:\projects\myproject.eprj".*
**/autoconnectsites**
This command forces PG4UWMC to connect programmer sites upon startup for all sites that were active when the software was last closed. This will launch the PG4UW control program for each of those sites automatically. An equivalent "Auto-connect Sites" option is also available in the PG4UWMC "Settings" dialog.

# *Hardware*

**Warning:**

*Class A ITE notice*

The devices described in this manual are Class A products. In a domestic environment, these products may cause radio interference, which may require the user to take appropriate corrective measures.

# *ISP (In-System Programming)*

Definition

**In-system programming (ISP)** enables the programming and reprogramming of a device while it is installed in its final system. Through a simple interface, the ISP programmer communicates serially with the device to update its non-volatile memory. This approach eliminates the need to physically remove chips, saving both time and money during laboratory development and field updates.
In this context:

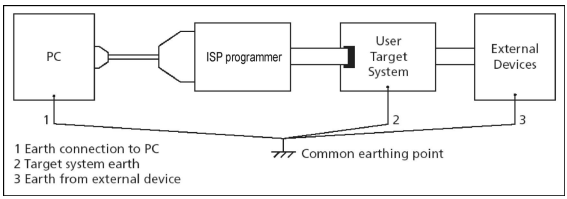
- The Target Device is the component (such as a microcontroller or PLD) to be programmed.
- The Target System is the physical PCB that houses the target device.
- The ISP Programmer is a device equipped with in-system programming capabilities, such as the DATAMAN-48PRO4 or DATAMAN-48PRO4C**.**

General rules for in-system programming

To avoid damaging your PC, ISP programmer, target device, or target system, we recommend following these general rules for in-system programming:

- Establish a **common earth point** for the target system, ISP programmer, and PC.

- If using a laptop or PC that lacks a direct connection to a common earth point, create a **hard-wired connection** to one (using the VGA connector, for example).
- Ensure that any other equipment connected to the target system is also linked to the **common earth point** too**.**




Connecting a Dataman ISP programmer to a target system involves linking two active electrical devices. Improper connection procedures can result in damage to both the programmer and the system.

**Note:** *If the programmer is damaged during in-system programming because these instructions were not followed, it will be considered damage due to improper handling and will* ***not be covered under warranty****.*

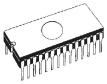
1. Turn off both the ISP programmer and the target device.
2. Ensure all devices share the same GND potential (e.g., by connecting their GND points with a wire).
3. Plug one end of the ISP cable into the programmer, then turn on the programmer and launch the control software.
4. Select the target device and desired operation options within the software.
5. Initiate the action for the target device (e.g., Read or Program).
6. When prompted by the software, connect the other end of the ISP cable to the target system and power it on.
7. When prompted by the software, disconnect the ISP cable from the target system and power the system off.
8. To perform another action on the target device, return to step 5.

**Design Recommendations for Target Systems**

The target system must be designed so that all signals required for in-system programming connect directly to the ISP programmer via the ISP connector. If the system uses these signals for other functions, they must be isolated to ensure the system does not interfere with them during programming. Dataman programmers are designed to work alongside manufacturer application notes; following these notes exactly is essential for successful in-system programming.

# *Other*

Do not move the **Info window** while the **BUSY LED** is illuminated. Doing so may trigger the watchdog circuit, placing the programmer into a safe state as if a PC-to-programmer communication error had occurred.



# Troubleshooting and warranty

# Troubleshooting

We want you to have the best experience with our product. However, if problems occur, please follow these troubleshooting steps:

- **Review Your Process:** Ensure the programmer and the PG4UW control software are being operated correctly.
- **Consult Documentation:** Re-read the provided documentation; you may find the solution immediately. For the most up-to-date information, use the **context-sensitive help** within the PG4UW control program.
- **Test on Another PC:** Try installing the programmer and software on a different computer. If it works there, the issue likely lies with the original PC.
- **Seek Internal Support:** Consult with a technical lead or someone in your office who has experience with the installation.

If the issue persists:

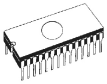
- **Contact Support:** Reach out to the local dealer where you purchased the programmer or contact Dataman Programmers directly.
- **Repairs:** If the unit is diagnosed as defective, consult your dealer or the Dataman contact page to find a repair centre in your country.
- **Shipping for Repair:** When sending a product for repair, ensure the package includes:
  - The defective product.
  - A completed **"DEVICE PROBLEM REPORT"** form
  - A photocopy of your **dated proof of purchase**.

***Without all these items we cannot admit your programmer to repair.***

# If you have an unsupported target device

If you need to work with a target device that the current control program doesn't support, please don't worry—just follow these steps:

- **Check our website:** View the device list for the latest software version at dataman.com/pages/downloads. Your device may already be supported! If it is, download **dataman_control_software.exe** and install the update.
- ***Contact Dataman directly:*** *Complete a "Device Problem Report" form, following the instructions provided at the end of the document. We may require detailed data sheets and, if possible, physical samples of the device. Any samples provided will be returned to you once support for the device has been added to a new version of PG4UW*



# *Warranty terms*

The manufacturer, Dataman Programmers Ltd, provides a three-year guarantee from the date of purchase covering the programmer (models DATAMAN-48PRO2, 48PRO2, 48PRO2C, and 40PRO), including all parts, materials, and workmanship. This warranty is limited to 25,000 cycles for DIL ZIF sockets and 10,000 cycles for all other ZIF socket types. If a product is confirmed to be defective, Dataman or an authorised repair centre will repair or replace the parts free of charge. Replacement parts or units are warranted for the remainder of the original warranty period only.
To claim a warranty repair, the customer must provide proof of the purchase date.
These terms apply to customers who purchase directly from Dataman. Please note that warranty conditions from third-party sellers may vary based on local laws or specific dealer policies.
The warranty does not cover:

- Standard wear and tear or mechanical damage.
- Products opened, altered, or repaired by unauthorised personnel.
- Misuse, abuse, accidental damage, or improper installation.

For repairs not covered by the warranty, you will be billed for materials, labour, and shipping.
Dataman or its distributors reserve the right to decide between repair or replacement and to determine warranty eligibility.
Please also refer to the **Troubleshooting** section for further assistance.

*Manufacturer:*
✉: Dataman Programmers Ltd
Unit 2 Newton Hall, Dorchester Road
Maiden Newton
Dorset
DT2 0BD
United Kingdom
☎: + 44 (0) 1300 320719
www.dataman.com, sales@dataman.com

Dataman has made every effort to develop stable and reliable hardware and software. However, Dataman does not guarantee that these products are entirely free of "bugs," errors, or defects. Dataman's liability is strictly limited to the net contract value paid by the buyer.

## Dataman is not liable for:

- **Misuse or Mishandling**: Any damage resulting from the inappropriate use or handling of the products.
- **Unauthorised Modifications**: Damage caused by users or third parties who modify or attempt to modify the hardware or software.
- **Consequential Damage**: Any further or secondary damage caused by hardware errors or software "bugs."

**For example:** *lost profits, lost savings, damages arising from claims of third parties against a client, damage or loss of recorded data or files, renown, loss caused by impossibility to use etc.*