

# User manual for



## BeeHive304

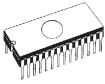
Ultra speed universal 4x 64-pindrive production multiprogrammer, design focused on high-capacity memories programming.



## BeeProg3

Ultra speed universal 64-pindrive programmer, design focused on high-capacity memories programming.





This document is copyrighted by Elnec s.r.o., Presov, Slovakia. 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 Elnec s.r.o.

The control program is copyright Elnec s.r.o., Presov, Slovakia. 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 consult manual on [www.elnec.com](http://www.elnec.com).

Elnec s.r.o. assumes no responsibility for misuse of this manual.

Elnec s.r.o. 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. Elnec s.r.o. respects those trademarks.

COPYRIGHT © 1991 – 2023

Elnec s.r.o.  
Presov, Slovakia

2<sup>nd</sup> November 2023  
ZLI-0330

## ***How to use this manual***

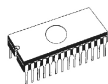
This manual explains how to install the control program and how to use your programmer. It is assumed that the user has some experience with PCs and installation of software. Once you have installed the control program we recommend you consult the context sensitive HELP within the control program rather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

***Dear customer,***

*thank you for purchasing one of the **Eltec**  
programmer.*

---

*Please, download actual version of manual  
from Eltec WEB site ([www.eltec.com](http://www.eltec.com)),  
section Support/ Download, if current one will  
be out of date.*



## Table of contents

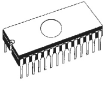
How to use this manual .....	3
<b>Introduction</b> .....	<b>6</b>
Products configuration .....	7
PC requirements .....	7
Software .....	9
Additional services: .....	9
<b>Quick Start</b> .....	<b>10</b>
<b>Detailed description</b> .....	<b>13</b>
<b>BeeHive304</b> .....	<b>14</b>
Introduction .....	15
BeeHive304 elements .....	19
Connecting BeeHive304 to the PC .....	21
Manipulation with the programmed device .....	21
Selftest and calibration check .....	22
Technical specification .....	23
<b>BeeProg3</b> .....	<b>29</b>
Introduction .....	30
BeeProg3 elements .....	33
Connecting BeeProg3 to the PC .....	34
Manipulation with the programmed device .....	35
Multiprogramming by BeeProg3 .....	35
Selftest and calibration check .....	35
Technical specification .....	37
<b>Setup</b> .....	<b>43</b>
Software setup .....	44
Hardware setup .....	48
<b>PG4UW</b> .....	<b>55</b>
PG4UW the programmer software .....	56
File .....	59
Buffer .....	67
Device .....	75
Programmer .....	105
Options .....	111
Help .....	125
<b>PG4UWMC</b> .....	<b>128</b>
Help .....	153
<b>Common notes</b> .....	<b>155</b>
Maintenance .....	156
Software .....	157
Hardware .....	163
Other .....	163
<b>Troubleshooting and warranty</b> .....	<b>164</b>
Troubleshooting .....	165
If you have an unsupported target device .....	166
Warranty terms .....	166

## ***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:***

<b><i>Device</i></b>	any kind of programmable integrated circuits or programmable devices
<b><i>ZIF socket</i></b>	Zero Insertion Force socket used for insertion of target device
<b><i>PMI</i></b>	Programming Module Interface – connectors used for insertion of programming module to programmer
<b><i>Buffer</i></b>	part of memory or disk, used for temporary data storage
<b><i>Printer port</i></b>	type of PC port (parallel), which is primarily dedicated for printer connection.
<b><i>USB port</i></b>	type of PC port (serial), which is dedicated for connecting portable and peripheral devices.
<b><i>HEX data format</i></b>	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*

---

This user manual covers Elnec programmers **BeeHive304** and **BeeProg3**.

**BeeHive304** is a desktop programmer but may be used as the core for automated programmers and automatic test equipments (ATE) too. It is ultra speed universal 4x64-pindrive **concurrent multiprogramming system** designed for high volume production programming of high capacity memories. The chips are programmed at near theoretical maximum programming speed.

**BeeProg3** is a desktop programmer but may be used as the core for automated programmers and automatic test equipments (ATE) too. It is ultra speed universal programmer with 64 powerful pindrivers designed for low volume production programming.

Advanced design, including protection circuits, original brand components and careful manufacturing allows us to provide a **three-year warranty** for BeeHive304 and BeeProg3 on parts and labor for the programmers. This warranty terms are valid for customers, who purchase a programmer directly from Elnec company. The warranty conditions of Elnec sellers may be differ depending on the target country law system or Elnec seller's warranty policy.

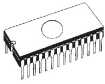
## Products configuration

	programmer	external power supply	internal power supply	power cord 1,2m	USB cable 1,8m	AP3 diagnostic POD	programming module fixating screw	screw with washers for ground connection	M4x70 DIN912 screw with washer	tie and tie mount for fixating cables	CD with software and user manual	sticker register your programmer	sticker Elnec programmer inside	leaflet Notes about ESD	antistatic set	vacuum handling tool kit	transport case
BeeHive304	•	•		•	•	1x	4x	•	6x	3x	•	•	2x	•	•	•	•
BeeProg3	•		•	•	•	•	•				•	•		•			•

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

## PC requirements

These PC requirements are valid for 3.38 version of PG4UW (issued 2.2018) and above.



## Minimal PC requirements

	OS - Windows	CPU	RAM [MB]	free disk space [MB]	USB 2.0 high speed	LAN
2x BeeHive304	XP	C2Quad	2000	500	•	1Gb
BeeHive304	XP	C2D	2000	500	•	1Gb
BeeProg3	XP	C2D	1000	500	-	100Mb

1024 x 768 is minimal monitor resolution.

The "Minimal PC requirements" mean that the device programmer and SW will run at these conditions, but not with fully enjoyable experience.

## Recommended PC requirements

	OS - Windows	CPU	RAM [MB]	free disk space [MB]	USB 2.0 high speed	2x USB 2.0 high speed controllers	LAN
2x BeeHive304	Win 7	Core i5 <sup>*1</sup>	4000 <sup>*2</sup>	2000 <sup>*2</sup>		•	1Gb
BeeHive304	Win 7	Core i3 <sup>*1</sup>	2000 <sup>*2</sup>	2000 <sup>*2</sup>	•		1Gb
BeeProg3	Win 7	Core i3 <sup>*1</sup>	2000 <sup>*2</sup>	1000 <sup>*2</sup>	•		1Gb

We recommended to use higher monitor resolution as 1024 x 768.

\*1 – or better

\*2 – or more

We recommend up-to-date Windows 11, but if you're happy with older version of Windows OS then we don't require upgrading your OS and our programming tools will work reliable also with your OS.

If two programmers are to be connected to a single PC, then we strongly recommend connecting each programmer to separate USB 2.0 High speed controller (USB EHCl). For more information see "Hardware setup" chapter.

**Warning:** It is not allowed to connect BeeHive304 or BeeProg3 programmer directly to PC via LAN cable. BeeHive304 or BeeProg3 may be connected via LAN cable using switch or router only.

Free disk space requirement depends also on used IC device size and number of attached programming sites. For large devices the required free space on disk will be approximately  $1000MB + 2x \text{ Device size} \times \text{number of programming sites}$  attached to this PC.

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



## Software

Elneec provides common software for all device programmers. Regular and OnDemand versions of software are available. Regular version is released usually every 3-4 weeks, OnDemand versions are released based on customer request - for hot new devices support and bug fixes - often daily. Elneec provides download of updated software version without any fees for all programmers.

### ***Why is it important to use the latest version of the control program?***

- Semiconductor manufacturers continuously introduce new devices with new package types, manufactured by new technologies in order to support the need for flexibility, quality and speed in product design and manufacturing. To keep pace and to keep you up-to-date, we usually implement more than 7000 new devices into the control program within a year.
- Furthermore, a typical programmable device undergoes several changes during its lifetime in an effort to maintain or to improve its technical characteristics and process yields. These changes often impact with the programming algorithms, which need to be upgraded (the programming algorithm is a set of instructions that tells the programmer how to program data into a particular target device). Using the newest algorithms in the programming process is the key to obtaining high quality results. In many cases, while the older algorithm will still program the device, they may not provide the level of data retention that would be possible with an optimal algorithm. Failure to not use the most current algorithm can decrease your programming yields (more improper programmed target devices), and may often increase programming times, or even affect the long term reliability of the programmed device.

*Our commitment is to implement support for these new or modified parts before or as soon as possible after their release, so that you can be sure that you are using latest and/or optimal programming algorithms that were created for this new device.*

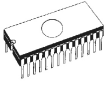
**Free software updates** are available from our Internet address [www.elnec.com](http://www.elnec.com).

## ***Additional services:***

- free technical support (phone/fax/e-mail).
- free lifetime software update via Web site.

We also offer the following new services in our customer support program:

- **Keep-Current** is a service by which Elneec ships to you the latest version of the control program for programmer and the updated user documentation. A Keep-Current service is your hassle-free guarantee that you always have access to the latest software and documentation, at minimal cost. For more information see [www.elnec.com](http://www.elnec.com).
- **AlgOR** (Algorithm On Request) service allows you to receive from Elneec software support for programming devices not yet available in the current device list. For more information see [www.elnec.com](http://www.elnec.com).



---

## *Quick Start*

---

## ***Installing programmer hardware***

- connect the USB port of programmer to a USB port of PC using supplied cable
- connect the connector of the power supply adapter (power cord) to the programmer and turn on programmer by switch

## ***Installing the programmer software***

Run the installation program from the CD (setup.exe) and follow the on-screen instructions. Please, for latest information about the programmer hardware and software see [www.elnec.com](http://www.elnec.com).

## ***Run the control program***



Double click on

After start, control program PG4UW automatically scans all existing ports and searches for any connected Elnec programmer. Program PG4UW is common for all Elnec programmers hence PG4UW will try to find all supported programmers.

Menu **File** is used for source files manipulation, settings and viewing directory, changes drives, changes start and finish address of buffer for loading and saving files and loading and saving projects.

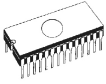
Menu **Buffer** is used for buffer manipulation, block operation, filling a part of buffer with string, erasing, checksum and of course editing and viewing with other items (find and replace string, printing...).

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







Menu **Programmer** is used for work with programmer.

Menu **Options** is used to view and change various default settings.

Menu **Help** is used for view supported devices and programmers and information about program version.



## Programming a device

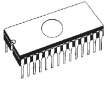
1. select device: click on 
2. load data into buffer:
  - a) from file: click on 
  - b) from device: insert device to AP3 programming module ZIF and click on 
3. insert target device to AP3 programming module ZIF
4. check, if the device is blank: click on 
5. program device: click on 
6. additional verify of device: click on 



---

## *Detailed description*

---



---

# BeeHive304

---





## Introduction

**BeeHive304** is next member of Elnec **concurrent universal multiprogrammer** series, built to meet the strong demand for an **extremely fast and reliable** multiprogrammer for high-capacity memories.

Designed with great emphasis on technical perfection and speed of hardware, this programmer perfectly **fits for high-demand desktop programming** as well as for **automated programming systems** and ATE machines, where ensures the highest quality and overall yield.

**BeeHive304** consists of four independent isolated **universal programming sites**, based on the **BeeProg3** programming core hardware. Therefore the programming sites can run asynchronously (**concurrent multiprogramming programming mode**). Each programming site starts programming at the moment the chip is detected to be inserted in the socket properly - independently on the status of other programming sites. As a result, three programming sites work, while you are replacing the programmed chip at the fourth site.

**BeeHive304** multiprogrammer supports **as many chips as the BeeProg3 programmer** and without obvious decrease of programming speed because each programming site works independently. Also each programming site can program a different chip, if necessary.

Because the **BeeProg3 programming core** is based on a state of the art **FPGA**, powerful **ARM** processor powered by Linux OS and internal **SSD**, **BeeHive304** is ready to program devices at theoretically possible speeds. The achieved ultra fast programming speed - more than 22.5 MB/s continuously - is actually higher, than real devices supported so far can utilize. This is reflected in extremely short programming times. For example, the **2 GB eMMC NAND** Flash could be done in **less than 100 sec** - if programmed memory allows that speed.

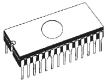
Tests show, that **BeeHive304** is currently faster than all competitors in this price category, and for many chips it is the fastest at all.

**BeeHive304** supports **all kinds** of types and silicon technologies of today's and tomorrows **programmable devices**. It partially supports also devices "from yesterday". You can be sure that the next devices support will require the software update only, at the most (if necessary) simple programming module, therefore **the cost of ownership is minimized**.

Modular construction of hardware - the programming sites works independently - allows for continuous operation when a part of the programming sites becomes inoperable. It also makes service quick and easy.

The sensing circuits detect proper placement of device in the socket of programming module, which allows beginning **programming immediately upon insertion of a chip**. Operator merely removes the finished chip and inserts a new chip. Operator training is therefore minimized.

**BeeHive304** interfaces with any IBM PC compatible personal computers, running MS Windows OS, through **USB** (2.0 High Speed) port or **LAN** port (via switch or router).



Banana jack for ESD wrist straps connection to easy-to-implement the ESD protection control and screw with washers for earth wire.

With its very competitive price together with excellent hardware design for reliable programming, it is probably the best "value for money" programmer in this class.

The 64-pin rich-features, **precise and powerful pindriver** of **BeeProg3** programming core deliver high speed, accurate and clean waveform signals to the device by eliminating noise, ground bounce and overshoot, which maximizes programming yield and guarantees long data retention. This also allows the reliable support of virtually any nonvolatile technology used for programmable devices - (E)EPROM, Flash, MRAM, PCM, ... - by a single device programmer.

**FPGA based** totally reconfigurable TTL pindrivers provide H/L/pull\_up/pull\_down and read capability for each pin of the device. The dual H/L drivers enables to provide two different H levels for both core signals and I/O signals of programmed device without additional logic. Programmer pindrivers operate down to 0.8V therefore the programmer is ready to program the full range of today's and tomorrow's advanced very-low-voltage devices.

**Extremely fast programming**, achieved by using FPGA based state machine, fast processor and SSD allow execution of all time-critical routines and data transfers inside of the programmer.

The programmer performs device **insertion test** based on the check of proper signal path between the programmer and programmed device before it programs each device. In dependence on programming configuration it identifies missed or poor contact between programmed device and the ZIF socket of the programming module, missed or poor contact between the programming module and the programmer and it's also able to indicate wrong position of device in the ZIF socket or the programming module (moved, rotated, backward oriented). These capabilities, supported by **overcurrent protection** and **signature-byte check** help prevent chip damage due to operator error.

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

**Built-in protection circuits** eliminate damage of programmer and/or programmed device due to environment or operator failure. All the inputs of the BeeProg3 programming core - pins of programming module interface (pindriver signals and also supporting signals), connection to PC and power supply input, are **protected against ESD** up to 15kV.

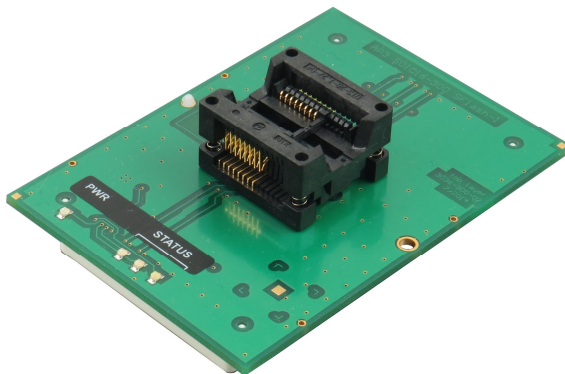
For the proper and reliable programming of ultra-fast memories the BeeHive304 utilizes **specialized modules, optimally designed** for specific device families, exactly according to the needs of programmed devices. But if it is possible, then universal programming modules, dedicated for IC package type, are used. The programming modules are identical for all programmers based on the BeeProg3 programming core.

Programming modules construction is **designed for perfect stability** at the top of the programmer, to be tough enough for insert/replace chips by mechanical arm and also allows keeping identical position of ZIF socket also after replacing of the module.

The **programming modules** are available for devices in PDIP, 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.



**Note:** orientation of ZIF socket and device reference pin are set with effort to achieve perfect functionality and ergonomic, therefore may be different at different AP3 programming modules.

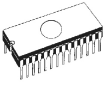
**BeeHive304** programmer is designed as desktop programmer, but it has a special construction that is suitable for use in the automated programming systems. Dimensions of the programmer are reduced to technical minimum for this features & quality hardware in intent to minimize overhead of the handler's arm movement. The construction is mechanically stable to be immune against vibration during operation. The case of BeeHive304 is prepared to be fastened from top or from bottom of programmer body into automated programmer working place.

**BeeHive304** can be implemented into automated programmer (as a replacement of obsolete programmer for example) or into some handler using more ways:

- if one or two BeeHive304 programmers are enough to perform all jobs, then it is enough to use one control PC for whole programming system, for both machine control and also programmer software. The BeeHive304 can be attached to this computer using USB or LAN network.
- if more than two BeeHive304 programmers are required to implement into automated programmer, then more computers are needed to use. Whole system is scalable up to 64 programming sites in one automated programming system by multiplying of BeeHive304 programmers and the control PCs. Maximal configuration comprise of 16x BeeHive304 programmers. One control computer in the system serves as a master unit. Here run also multiprogramming control software, serialization engine and interface to the host system. Interfacing of control PCs in whole network is done over standard LAN network.

Implementation of BeeHive304 into software of **automated programmers** and **handlers** is by using of simple **remote control** of the PG4UWMC control software. There exist examples of implementation for standard programming languages and of course we are ready to help customer with this task.

The programmer is driven by a **comfortable and easy-to-use** control program with pull-down menu, hot keys and on-line help. The software runs on all versions of MS Windows from Windows XP to Windows 11 (32bit and 64-bit).



- **PG4UWMC** is control software for production mode  
This part of the software is focused on the easy monitoring of high-volume production operations.  
Operator-friendly control software combines many powerful functions with ease of use. Graphic user interface provide overview of all important activities result without burden of operator with non-important details.  
There is used a **project file** to control the BeeHive304 multiprogramming system. Project file contains user data, chip programming setup information, chip configuration data, auto programming command sequence, etc. Therefore the operator error is minimized, because the project file is normally created and proofed by engineering and then given to the operator. The optional protected mode can be set for project file to avoid unwanted changes of the project file.  
Each chip may be programmed with different data such as serial number, configuration and calibration information using rich-features serialization system.
- **PG4UW** is control software for engineering mode or programming site driver  
This part of the software is focused to the quick and easy preparation of the project file for usage in the production mode control software.  
Each programming site is driven by a **comfortable and easy-to-use** control program with pull-down menu, hot keys and on-line help. It is the same years-proven software, as is used for all other El nec single-site programmers.  
Selecting of device is performed by its class, by manufacturer or simply by typing a fragment of 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 of programming immediately after insertion of chip into socket).  
All known data formats are supported. Automatic file format detection and conversion during loading of file are performed.  
The software also provides - in the Device information section - lots of information about the programmed device. As a special, the **drawings of all available packages** are provided. The software also provides **explanation of chip marking** (the meaning of prefixes and suffixes at the chips) for each supported chip.  
The rich-featured **serialization 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 a file and program it to programmed device.  
**Jam files** of JEDEC standard JESD-71 are interpreted by **Jam Player**. Jam files are generated by design software which is provided by the manufacturer of respective programmable device. Chips are programmed through JTAG (IEEE 1149.1 Joint Test Action Group) interface.  
**VME files** are interpreted by VME Player. VME file is a compressed binary variation of SVF file and contains high-level IEEE 1149.1 bus operations. SVF files are interpreted by SVF Player. SVF file (Serial Vector Format) contains high-level IEEE 1149.1 bus operations. SVF files are generated by design software which is provided by manufacturer of respective programmable device. Chips are programmed through JTAG interface. VME files are generated by design software which is provided by manufacturer of respective programmable device.

It is important to remember, that a support of most of the new devices requires **only a software update**, because the **BeeHive304** is truly a universal programmer. With our prompt

service, support for the new device can be added within hours. See **AlgOR** (Algorithm On Request) service and **OnDemand** software for details.

This service is almost in all cases free. Please note, however, that we can ask the customer to share the costs, if development and manufacturing costs are too high.

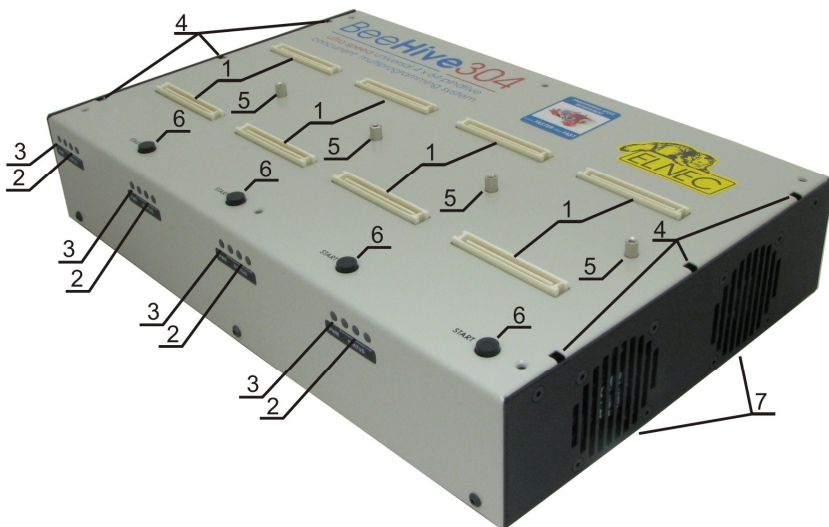
Advanced design of the **BeeHive304** universal programmer, including protective circuits, original brand components, careful manufacturing and burning-in allow us to provide a **three-year warranty** on parts and workmanship of the programmer.

Elneec provides **free shipping of programmer repaired under warranty back to customers worldwide**. The warranty is valid from the date of purchase.

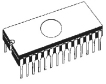
Registration of the product will speed up every repair request. Registration should be performed within 60 days from the date of purchase

## BeeHive304 elements

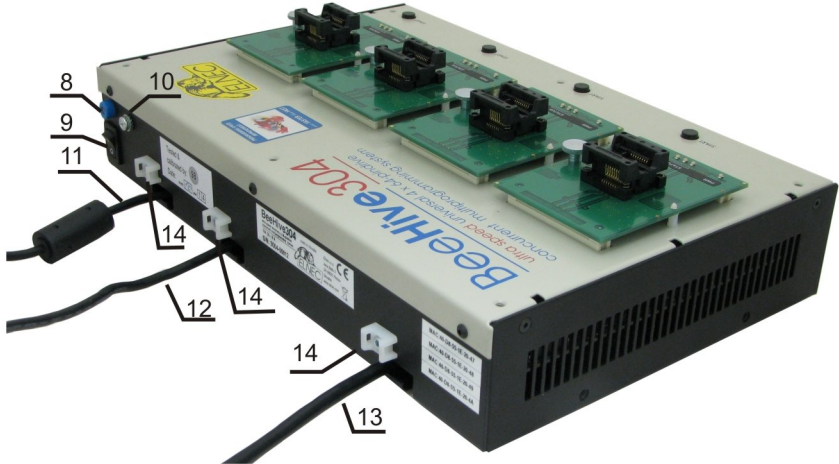
- 1) Programming Module Interface (PMI) connectors
- 2) work result LEDs of site
- 3) power/sleep LED of site
- 4) 6 x 4,2mm holes for fastening BeeHive304 to bottom or top plate
- 5) programming module fixing screws
- 6) button START
- 7) temperature controlled cooling fans



Front and top view to BeeHive304

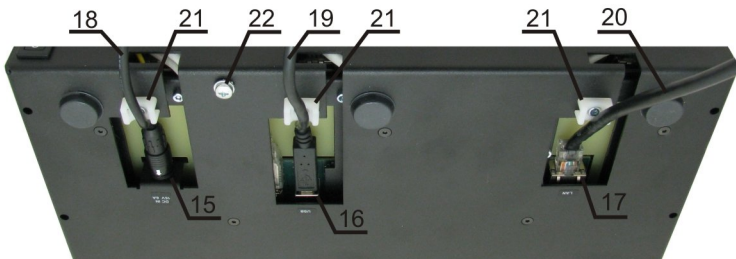


- 8) banana jack for ESD wrist strap connection
- 9) ON/OFF switch
- 10) screw with washers for ground connection on the rear side
- 11) 15V DC power cable connected to programmer from rear side
- 12) USB cable connected to programmer from rear side
- 13) LAN cable connected to programmer from rear side
- 14) rear side cable tie mounts for fixing power, USB and LAN cables



Rear and top view to BeeHive304

- 15) power connector, 15V 6A
- 16) type B USB connector for PC ↔ BeeHive304 communication cable
- 17) LAN connector for network ↔ BeeHive304 communication cable
- 18) 15V DC power cable connected to programmer from bottom side
- 19) USB cable connected to programmer from bottom side
- 20) LAN cable connected to programmer from bottom side
- 21) bottom side cable tie mount for fixing power, USB or LAN cable
- 22) screw with washers for ground connection on the bottom side



Bottom view of BeeHive304

## Connecting BeeHive304 to the PC

### Using USB port

Recommendation for connecting programmer to PC:

1. make ground connection between programmer and PC or other ground
2. connect programmer with PC via USB cable
3. connect power supply to programmer and turn on it by switch
4. run PG4UW control program and connect programmer

### Using LAN port

Recommendation for connecting programmer to PC:

1. make ground connection between programmer and PC or other ground
2. connect programmer with Cat5e Ethernet cable to nearest network device (switch, hub or router)
3. ensure, that DHCP server in your network is configured
4. connect power supply to programmer and turn on it by power switch
5. run PG4UW control program and connect programmer

**Warning:** *Beeprog3 contains SSD as buffer and ARM powered by Linux OS, therefore we recommended shut down programmer by following steps:*

1. turn off PG4UW control software
2. turn off programmer by power switch

*Otherwise when you start PG4UW control software and connect this not correctly power down programmer long scanning of internal memory (SSD) occur.*

*When work result LED blinking don't turn off programmer to avoid programmer damage!!!*

## Manipulation with the programmed device

At control program select menu **Device / Select device (Alt+F5)**. Select your desired device by typing full or part of name. At right column you see name of AP3 module which is needed for work with your desired device.

Insert AP3 programming module to Programming Module Interface (PMI) connectors.

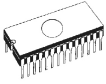
Then you can insert desired device into the open ZIF socket (push the top of ZIF) and close socket (release the top of ZIF). The correct orientation of the programmed device in ZIF socket is shown on the picture near the ZIF socket.

#### **Note:**

*Programmer need not to be switched off and also the software may be running when:*

- *Inserting / removing AP3 programming module to / from Programming Module Interface (PMI) connectors*
- *Inserting / removing the target device to / from the AP3 programming module ZIF socket but none operation with target device can be active (LED BUSY light off).*

*Programmer's protection electronics protect the target device and the programmer itself against either short or long-term power failures and, partly, also against a PC failure. However,*



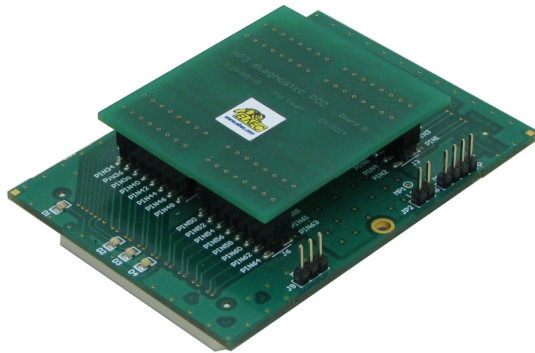
*it is not possible to grant the integrity of the target device due to incorrect, user-selected programming parameters. Target device may be not destroyed by forced interruption of the control program (reset or switch-off PC), by removing the physical connection to the programmer, but the content of actually programmed cell may remains undefined.*

## **Selftest and calibration check**

If you feel that your programmer does not perform according to your expectation, please run the programmer **Selftest** with using AP3 diagnostic POD enclosed in the standard delivery package.

### **Selftest of programmer site**

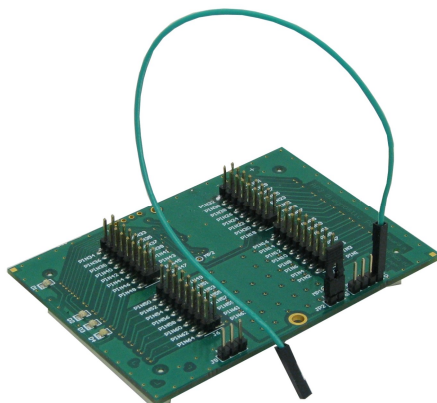
- Put together Board A with Board B of AP3 diagnostic POD
- Insert **AP3 diagnostic POD** into Programming Module Interface (PMI) connectors of the programmer site.
- Run selftest of programmer in PG4UW (menu Programmer / Selftest).



AP3 diagnostic POD for Selftest plus

### **Calibration test**

- Disconnect **Board A** from **Board B** of **AP3 diagnostic POD**
- Insert **Board A** of **AP3 diagnostic POD** into Programming Module interface (PMI) connectors of the programmer site.
- Run calibration test of programmer in PG4UW (menu Programmer / Calibration test).



AP3 diagnostic POD for Calibration test

## Technical specification

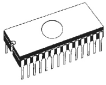
### HARDWARE

#### Base unit, DACs

- USB 2.0 high-speed compatible port, up to 480 Mb/s transfer rate
- 100Mbit LAN port
- on-board intelligence: powerful processor (ARM9 400MHz) and FPGA based state machine (basic clock 50 MHz plus PLL)
- built-in mSATA SSD as internal buffer (1282GB, upgradeable to higher capacity) (\*)
- three D/A converters for VCC1, VCC2, and VPP, controllable rise and fall time
- VCC1, VCC2 range 0.8V..7V/1A (step 10mV)
- VPP range 0V..25V/1A (step 25mV)
- Temperature controlled fans
- selftest capability
- protection against surge and ESD on power supply input, USB and LAN port and pins of Programming Module Interface (PMI)(IEC1000-4-2: 15kV air, 8kV contact)
- banana jack for ESD wrist straps connection
- screw with two washers for connection to ground

**(\*) Note:**

- *the data in the buffer are stored in fragments and also partially compressed, therefore the buffer can keep standard data also for bigger size devices, than current size of the buffer*
- *this upgrade is mechanically complicated and can be done at Elnec only*
- *the programmers up to s/n 3004-00041(including) were supplied with 32GB SSD as internal buffer*
- *the 128GB SSD can store as a buffer roughly 110 GB of the random data (28 GB in case of 32 GB SSD) for the bigger sizes SSD, proportionally. If you plan to work/copy eMMC/NAND*



device, which might contain random data in full range of the device size, we recommend to have equal capacity buffer in the programmer.

## **Pindriver (available on the programming module interface - connectors for programming modules)**

- pindrivers: 64 universal
- VCC1/VCC2 and VPP can be connected to each pin
- perfect ground for each pin
- 2 independent FPGA based TTL driver provides H, L, CLK, pull-up, pull-down on all pindriver pins, logic level 0,75V - 5V (IOL and IOH current 20mA)
- logic signals frequency: up to 125MHz (3.3V), 80MHz (5V)
- analog pindriver output level selectable from 0.8 V up to 25V
- current limitation, overcurrent shutdown, power failure shutdown
- continuity test: each pin is tested before every programming operation

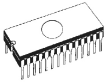
## **DEVICE SUPPORT**

### **Programmer, using programming modules:**

- NAND FLASH: Samsung K9xxx, KFxxx, SK Hynix (ex Hynix) HY27xxx, H27xxx, Toshiba TC58xxx, TH58xxx, Micron MT29Fxxx, (ex Numonyx ex STM) NANDxxx, Spansion S30Mxxx, S34xxx, 3D-Plus 3DFNxxx, ATO Solution AFNDxxx, Fidelix FMNDxxx, Eon Silicon Sol. EN27xxx, ESMT F59xxx, LBA-NAND Toshiba THGVNxxx
- serial NAND FLASH: Micron MT29Fxxx, GigaDevice GD5Fxxx
- eMMC: Hynix H26Mxxxxxxx, Kingston KE44B-xxxx/xxx, Micron MTFCxxxxxxx, Numonyx NANDxxxxxxx, Phison PSM4A11-xx, Samsung KLMxxxxxxx, SanDisk SDINxxx-xx, Toshiba THGBMxxxxxxx
- Multi-chip devices: NAND+RAM, NOR+RAM, NOR+NOR+RAM, NAND+NOR+RAM
- Serial Flash: standard SPI, high performance Dual I/O SPI and Quad I/O SPI (25Bxxx, 25Dxxx, 25Exxx, 25Fxxx, 25Lxxx, 25Mxxx, 25Pxxx, 25Qxxx, 25Sxxx, 25Txxx, 25Uxxx, 25Vxxx, 25Wxxx, 25Xxxx, 26Vxxx, 45PExx, MX66Lxxx, S70FLxxx), DataFlash (AT45Dxxx, AT26Dxxx)
- parallel NOR Flash: 28Fxxx, 29Cxxx, 29Fxxx, 29GLxxx, 29BVxxx, 29LVxxx, 29Wxxx, 49Fxxx series, Samsung's K8Fxxxx, K8Cxxxx, K8Sxxxx, K8Pxxxx series, ...
- EPROM: NMOS/CMOS, 27xxx and 27Cxxx series
- EEPROM: NMOS/CMOS, 28xxx, 28Cxxx, 27EExxx series, 3D Plus 3DEExxxxxxx
- mDOC H3: SanDisk (ex M-Systems) SDED5xxx, SDED7xxx, MD2533xxx, MD2534xxx, Hynix HY23xxx
- FRAM: Ramtron
- MRAM: Everspin MRxxxxx8x, 3D Plus 3DMRxxxxxxx
- NV RAM: Dallas DSxxx, SGS/Inmos MKxxx, SIMTEK STKxxx, XICOR 2xxx, ZMD U63x series
- Serial E(E)PROM: Serial E(E)PROM: 11LCxxx, 24Cxxx, 24Fxxx, 25Cxxx, 30TSExxx, 34Cxxx, 34TSxx, 59Cxxx, 85xxx, 93Cxxx, NVM3060, MDAxxx series, full support for LV series, AT88SCxxx
- Serial FRAM: Cypress(Ramtron): FM24xxxxxx, FM25xxxxxx, Fujitsu: MB85RCxxxx, MB85RSxxxx, Lapis(OKI, Rohm): MR44xxxxx, MR45xxxxx
- Serial MRAM: Everspin MH20xxx, MH25xxx



- Configuration (EE)PROM: XCFxxx, XC17xxxx, XC18Vxxx, EPCxxx, EPCSxxx, AT17xxx, AT18Fxxx, 37LVxx
- 1-Wire E(E)PROM: DS1xxx, DS2xxx
- PLD Altera: MAX 3000A, MAX 7000A, MAX 7000B, MAX 7000S, MAX7000AE, MAX II/G/Z, MAX V
- 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
- PLD: Xilinx: XC9500, XC9500XL, XC9500XV, CoolRunner XPLA3, CoolRunner-II
- SPLD/CPLD series: AMD, AML, Atmel, Cypress, Gould, ICT, Lattice, National Semicond., Philips, STMicroelectronics, TI (TMS), Vantis, VLSI
- FPGA: FPGA: Microsemi(Actel): ProASIC3, IGL00, Fusion, ProASICplus, SmartFusion
- FPGA: Lattice: MachXO, MachXO2, LatticeXP, LatticeXP2, ispXPGA
- FPGA: Xilinx: Spartan-3AN
- Clocks: TI(TMS), Cypress
- Special chips: Atmel Tire Pressure Monitoring ATA6285N, ATA6286N; PWM controllers: Ziker Labs, Analog Devices; Multi-Phase ICs: IR(Chil Semiconductor); Gamma buffers: AUO, Maxim, TI, ...
- Microcontrollers MCS51 series: 87Cxxx, 87LVxx, 89Cxxx, 89Sxxx, 89Fxxx, 89LVxxx, 89LSxxx, 89LPxxx, 89Exxx, 89Lxxx, all manufacturers, Philips LPC series
- Microcontrollers Atmel ARM. AT91SAM7Sxx, AT91SAM7Lxx, AT91SAM7Xxx, AT91SAM7XCxx, AT91SAM7SExx series;
- Microcontrollers Atmel ARM9: AT91SAM9xxx series;
- Microcontrollers ARM Cortex-M3: ATSAM3Axxx, ATSAM3Uxxx, ATSAM3Nxxx, ATSAM3Sxxx, ATSAMD20, ATSAM3Xxxx series
- Microcontrollers ARM Cortex-M4: ATSAM4Sxxx series
- Microcontrollers Atmel AVR 8bit/16bit: AT90Sxxxx, AT90pwm, AT90can, AT90usb, ATtiny, ATmega, ATxmega series
- Microcontrollers Atmel AVR32: AT32UC3xxxx, ATUCxxxD3/D4/L3U/L4U series
- Microcontrollers TI (Chipcon): CC11xx, CC24xx, CC25xx, CC85xx series
- Microcontrollers Coreriver: Atom 1.0, MiDAS1.0, 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 Generalplus: GPM8Fxxx series
- Microcontrollers GreenPeak: GPxxx series
- Microcontrollers Infineon(Siemens): XC800, C500, XC166, C166 series
- Microcontrollers MDT 1xxx and 2xxx series
- Microcontrollers Megawin: MG87xxx, MPC82xxx series
- Microcontrollers Microchip PICmicro: PIC10xxx, PIC12xxx, PIC16xxx, PIC17Cxxx, PIC18xxx, PIC24xxx, dsPIC, PIC32xxx series
- Microcontrollers Motorola/Freescale: HC05, HC08, HC11, HC12, HCS08, RS08, S12, S12X, MC56F, MCF51, MCF52 series, Kinetis (K,L), Qorivva/5xxx Power Architecture
- Microcontrollers Myson 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 Flash and OTP series
- Microcontrollers Nuvoton ARM Cortex-Mx: NUC1xx, M05x, Mini51, Nano1xx series
- Microcontrollers Nuvoton (Winbond): N79xxx, W77xxx, W78xxx, W79xxx, W83xxx series
- Microcontrollers NXP (Philips) ARM Cortex-Mx: LPC11xx, LPC11Cxx, LPC11Dxx, LPC11Uxx, LPC12xx, LPC12Dxx, LPC13xx, LPC17xx, LPC11Axx, LPC11Exx, LPC11xxLV, LPC18xx, LPC43xx, LPC8xx, EM7xx, series
- Microcontrollers NXP (Philips) UOC series: UOCIII, UOC-TOP, UOC-Fighter (TDA1xxxx) series
- Microcontrollers NXP (Philips) ARM7: LPC2xxx, MPT6xx, PCD807xx, SAF7780xxx series
- Microcontrollers NXP (Philips) ARM9: LPC31xx series
- Microcontrollers Pasat: TinyModule DIL40, DIL50 series
- Microcontrollers Scenix (Ubicom): SXxxx series
- Microcontrollers Syntek: STK6xxx series
- Microcontrollers Renesas: R8C/Tiny series, RX series, uPD70Fxxx, uPD78Fxxx series, RL78 series, R32C series
- Microcontrollers SyncMOS: SM39xxx, SM59xxx, SM73xxx, SM79xxx, SM89xxx series
- Microcontrollers & Programmable System Memory STMicroelectronics: uPSD, PSD 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 Silicon Laboratories(Energy Micro): EFM32Gxx, EFM32GGxx, EFM32LGxx, EFM32TGxx, EFM32WGxx series
- Microcontrollers Silicon Laboratories: SiM3Cxxx, SiM3Lxxx, SiM3Uxxx series
- Microcontrollers Texas Instruments: MSP430 series, MSC12xx series, TMS320F series, CC430 series,
- Microcontrollers Texas Instruments (ex Luminary Micro): LM3Sxxx, LM3Sxxxx series, LM4Fxxxx series, TM4C series
- Microcontrollers ZILOG: Z86/Z89xxx and Z8Fxxxx, Z8FMCxxxxx, Z16Fxxxx, ZGP323xxxxxxx, 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, QIXIN, Signetic, Tekmos, Weltrend, Amic, Cyrod Technologies, Ember, Ramtron, Nordic Semiconductor, Samsung, ABOV Semiconductor...

**Notes:**

- For all supported devices see actual **Device list** on [www.elnec.com](http://www.elnec.com)

## Package support

- 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..

## Programming speed

**Notes:**

- It is important to say, that we always use random numbers data pattern for programming speed testing. Some our competitors use "sparse" data pattern, where only small amount of non-blank data are programmed or there are used data with only few 0 bits (FE, EF, etc.).

*This cheating approach can "decrease" programming time considerably. If you plan to compare, always ask which pattern they use.*

- *The programming speed practically doesn't depend on PC type because data for programming and main part of programming algorithm are stored internally inside of the programmer.*
- *All devices mentioned below, including both NAND Flash, are programmed at their maximal speed, the programming time can not be shorter.*
- *We're sorry, but there exist not very much devices, where the 20+MB/s BeeHive304 programming speed can be utilized*

Device	Size [bits]	Operation	Time
JS28F00AM29EWH (parallel NOR Flash)	4000080hx16 (1 Giga)	programming and verify	10 sec
MT29F1G08ABAEAWP (parallel NAND Flash)	8400000hx8 (1 Giga)	programming and verify	19.5 sec
SDIN7DP2-8G (eMMC NAND FLASH)	1D2000000hx8 (64Giga)	programming *1	342 sec
S25FL164K (serial Flash)	800300hx8 (64 Mega)	programming and verify	30.7 sec
AT89LP51RD2 (microcontroller)	10000hx8	programming and verify	5.2 sec
PIC32MX360F512L (microcontroller)	80000hx8	programming and verify	8.9 sec

**Conditions:** Core2 Duo, 3.16 GHz, 1G RAM, USB 2.0 HS, Windows 7. Version of SW: 3.09, 10/2014.

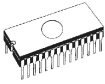
\*1 Verification of programming is done by internal controller of eMMC device. The device receives a block of data plus CRC, if it all matches, the internal controller confirm the proper programming.

## SOFTWARE

- **Algorithms:** only manufacturer approved or certified algorithms are used. Custom algorithms are available at additional fee.
- **Algorithm updates:** software updates are available regularly, approx. every 4 weeks, free of charge. **OnDemand** version of software is available for highly needed chips support and/or bugs fixes. Available nearly daily, depending on request.
- **Main features:** revision history, session logging, on-line help, device and algorithm information

## Device operations

- **standard:**
  - intelligent device selection by device type, manufacturer or typed fragment of part name
  - automatic ID-based selection of EPROM/Flash EPROM
  - blank check, read, verify
  - program
  - erase
  - configuration and security bit program
  - 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 (automatic start immediately after device insertion)
  - multi-project mode
  - lot of serialization modes (more type of incremental modes, from-file mode, custom generator mode)
  - statistic
  - count-down mode

## **Buffer operations**

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

## **File load/save**

- 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
- 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**

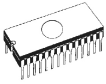
- external power supply unit: operating voltage 100-240V AC rated, 90-264 VAC max., 47-63Hz. Output voltage: 15V, 6A, output cable length 1200mm (47.2 inch)
- power consumption max. 90W active
- **dimensions of BeeHive304** programmer: 320,5 x 205 x 58,4 mm (12,6 x 8,1 x 2,3 inch). *Dimensions were measured without programming module inserted and does not include projections. Total height of BeeHive304 programmer with programming module(s) inserted depends on ZIF socket height and can vary between 76-88mm.*
- weight of programmer without programming modules: 3.6kg (7,9 lb)
- operating temperature: 5°C ÷ 40°C (41°F ÷ 104°F)
- operating humidity: 20%..80%, non condensing

---

# BeeProg3

---





## Introduction

**BeeProg3** is next member of Eltec **universal programmer** series, built to meet the strong appeal for an **extremely fast and reliable programmer** for high-capacity memories.

**BeeProg3** is designed with great emphasis on technical perfection and speed of hardware, this programmer perfectly fits for **high-demanding desktop programming** as well as for **automated programming systems** and ATE machines, where ensures the highest quality and overall yield.

**BeeProg3 programmer** core is based on a state of the art **FPGA**, powerful **ARM** processor powered by Linux OS and internal SSD to be ready to program devices at theoretically possible speeds. The achieved **ultra fast programming speed** - more than 22,5 MB/s continuously - is actually higher, than real devices supported so far can utilize. This is reflected in extremely short programming times. For example, the 2 GB eMMC NAND Flash could be done in **less than 100 sec** - if programmed memory allow that speed.

Tests show, the BeeProg3 is currently faster than all competitors in this price category (status: 10/2014), for many chips it the fastest at all.

**BeeProg3** supports all kinds of types and silicon technologies of today's and tomorrows **programmable devices**. It partially supports also devices "from yesterday". You can be sure that the next devices support will require the software update only, at the most (if necessary) simple programming module, therefore **the cost of ownership is minimized**. You have a freedom to choose the optimal device for your design.

For the proper and reliable programming of ultra-fast memories the BeeProg3 utilizes **specialized modules, optimally designed** for specific device families, exactly according to the needs of programmed devices. But if it is possible, then universal programming modules, dedicated for IC package type, are used. The programming modules are identical for all programmers based on the BeeProg3 programming core (BeeHive304 for example).

Programming modules construction is designed for perfect stability at the top of the programmer, to be tough enough for insert/replace chips by mechanical arm and also allows keeping identical position of ZIF socket also after replacing of the module.

**BeeProg3** interfaces with any IBM PC compatible personal computers, running MS Windows OS, through USB (2.0 High Speed) port or 100 Mb LAN port (via switch or router).

With its very competitive price together with excellent hardware design for reliable programming, it is probably the best "value for money" programmer in this class.

The 64-pin rich-features, **precise and powerful pindriver of BeeProg3** deliver high speed, accurate and clean waveform signals to the device by eliminating noise, ground bounce and overshoot, which maximizes programming yield and guarantees long data retention. This also allows the reliable support of virtually any nonvolatile technology used for programmable devices - (E)EPROM, Flash, MRAM, PCM, ... - by a single device programmer.

**FPGA based** totally reconfigurable TTL pindrivers provide H/L/pull\_up/pull\_down and read capability for each pin of the device. The dual H/L drivers enables to provide two different H levels for both core signals and I/O signals of programmed device without additional logic.

Programmer pindrivers operate down to 0.8V, therefore the programmer is ready to program the full range of today's and tomorrow's advanced very-low-voltage devices.

**Extremely fast programming**, achieved by using FPGA based state machine, fast processor and SSD, allow execution of all time-critical routines and data transfers inside of the programmer.

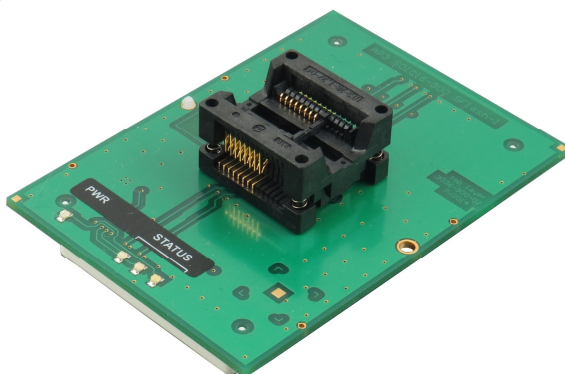
The programmer performs device **insertion test** based on the check of proper signal path between the programmer and programmed device before it programs each device. In dependence on programming configuration it identifies missed or poor contact between programmed device and the ZIF socket of the programming module, missed or poor contact between the programming module and the programmer and it's also able to indicate wrong position of device in the ZIF socket or the programming module (moved, rotated, backward oriented). These capabilities, supported by **overcurrent protection** and **signature-byte check** help prevent chip damage due to operator error.

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

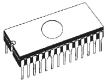
**Built-in protection circuits** eliminate damage of programmer and/or programmed device due to environment or operator failure. All the inputs of the BeeProg3 programmer, including the pindriver signals, connection to PC and power supply input, are **protected against ESD** up to 15kV.

Banana jack for ESD wrist straps connection to easy-to-implement the ESD protection control.

The programming modules are available for devices in PDIP, 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.



**Note:** orientation of ZIF socket and device reference pin are set with effort to achieve perfect functionality and ergonomic, therefore may be different at different AP3 programming modules.



The programmer is driven by a **comfortable and easy-to-use** control program with pull-down menu, hot keys and on-line help. The software runs on all versions of MS Windows from Windows XP to Windows 11 (32bit and 64-bit).

Selecting of device is performed by its class, by manufacturer or simply by typing a fragment of 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 of programming immediately after insertion of chip into socket).

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

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

The software also provides - in the Device information section - lots of information about the programmed device. In addition, the **drawings of all available packages** are provided. The software also provides **explanation of chip marking** (the meaning of prefixes and suffixes at the chips) for each supported chip.

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

**VME files** are interpreted by VME Player. VME file is a compressed binary variation of SVF file and contains high-level IEEE 1149.1 bus operations. SVF files are interpreted by SVF Player. SVF file (Serial Vector Format) contains high-level IEEE 1149.1 bus operations. SVF files are generated by design software which is provided by manufacturer of respective programmable device. Chips are programmed through JTAG interface. VME files are generated by design software which is provided by manufacturer of respective programmable device.

By attaching more BeeProg3 programmers to the same PC (through USB port) one can create a **powerful multiprogramming system**, which **supports as many chips, as the BeeProg3 programmer** and without obvious decrease of programming speed. It is important to know, that there is concurrent multiprogramming - each programmer works independently and each programmer can program a different chip, if necessary.

Advanced design of the **BeeProg3** universal programmer, including protective circuits, original brand components, careful manufacturing and burning-in allow us to provide a **three-year warranty** on parts and workmanship of the programmer.

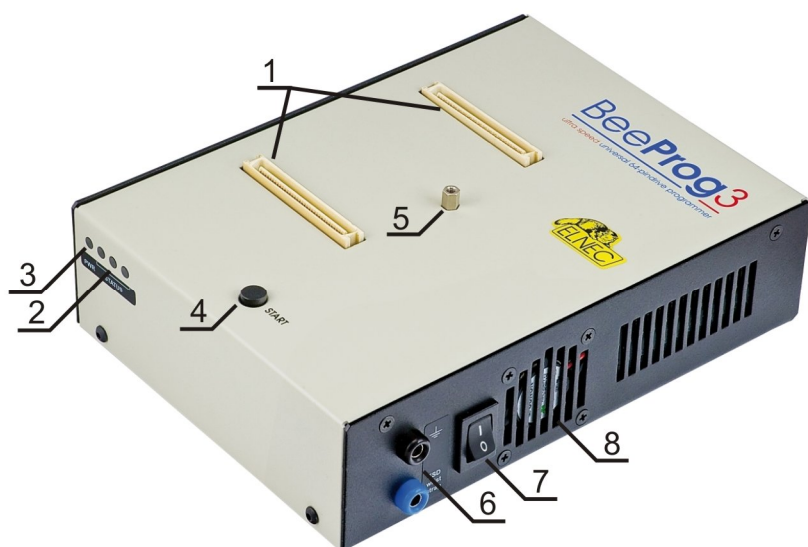
Eltec provides **free shipping of programmer repaired under warranty back to customers worldwide**. The warranty is valid from the date of purchase.

Registration of the product will speed up every repair request. Registration should be performed within 60 days from the date of purchase

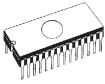


## BeeProg3 elements

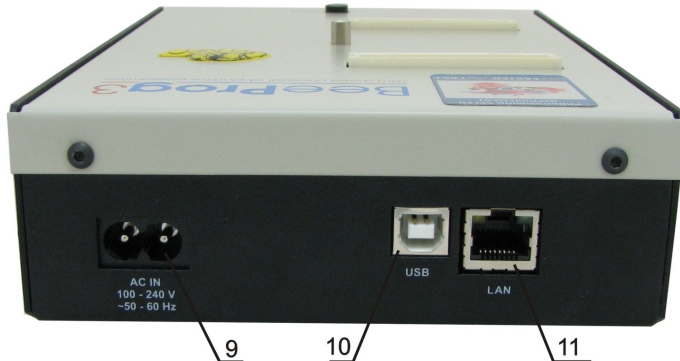
- 1) Programming Module Interface (PMI) connectors
- 2) work result LEDs
- 3) power/sleep LED of site
- 4) button START
- 5) programming module fixing screw
- 6) "GND" banana jack can be used for grounding of the programmer  
"ESD wrist strap" banana jack is place for attaching of ESD wrist strap
- 7) power switch
- 8) temperature controlled cooling fan



Right top view to BeeProg3



- 9) power supply connector
- 10) type B USB connector for PC ↔ BeeProg3 communication cable
- 11) LAN connector for network ↔ BeeProg3 communication cable



Rear view to BeeProg3

## Connecting BeeProg3 to the PC

### Using USB port

Recommendation for connecting programmer to PC:

1. make ground connection between programmer and PC or other ground
2. connect programmer with PC via USB cable
3. connect power supply to programmer and turn on it by switch
4. run PG4UW control program and connect programmer

### Using LAN port

Recommendation for connecting programmer to PC:

1. make ground connection between programmer and PC or other ground
2. connect programmer with Cat5e Ethernet cable to nearest network device (switch, hub or router)
3. ensure, that DHCP server in your network is configured
4. connect power supply to programmer and turn on it by power switch
5. run PG4UW control program and connect programmer

**Warning:** *Beeprog3 contains SSD as buffer and ARM powered by Linux OS, therefore we recommended shut down programmer by following steps:*

1. *turn off PG4UW control software*
2. *turn off programmer by power switch*

*Otherwise when you start PG4UW control software and connect this not correctly power down programmer long scanning of internal memory (SSD) occur.*

*When work result LED blinking don't turn off programmer to avoid programmer damage!!!*

## Manipulation with the programmed device

At control program select menu **Device / Select device (Alt+F5)**. Select your desired device by typing full or part of name. At right column you see name of AP3 module which is needed for work with your desired device.

Insert AP3 programming module to Programming Module Interface (PMI) connectors.

Then you can insert desired device into the open ZIF socket (push the top of ZIF) and close socket (release the top of ZIF). The correct orientation of the programmed device in ZIF socket is shown on the picture near the ZIF socket.

### **Note:**

*Programmer need not to be switched off and also the software may be running when:*

- *Inserting / removing AP3 programming module to / from Programming Module Interface (PMI) connectors*
- *Inserting / removing the target device to / from the AP3 programming module ZIF socket but none operation with target device can be active (LED BUSY light off).*

*Programmer's protection electronics protect the target device and the programmer itself against either short or long-term power failures and, partly, also against a PC failure. However, it is not possible to grant the integrity of the target device due to incorrect, user-selected programming parameters. Target device may be not destroyed by forced interruption of the control program (reset or switch-off PC), by removing the physical connection to the programmer, but the content of actually programmed cell may remains undefined.*

## Multiprogramming by BeeProg3

During installation of PG4UW at Select Additional Tasks window you check, if it is allowed to install BeeProg3 multiprogramming control support.

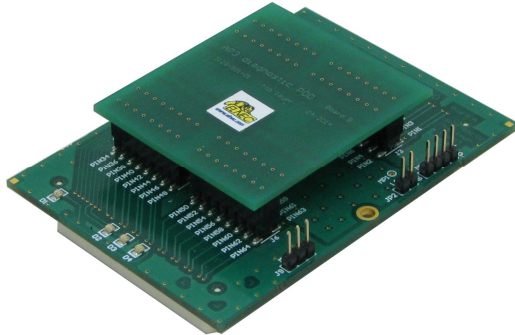
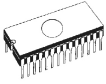
For start of BeeProg3 multiprogramming is necessary run special control program **pg4uwmc.exe**. At this program user assign BeeProg3 to control programs, may load projects for all BeeProg3 and run PG4UW for every connected and assigned BeeProg3.

## Selftest and calibration check

If you feel that your programmer does not perform according to your expectation, please run the programmer **Selftest** with using AP3 diagnostic POD enclosed in the standard delivery package.

### **Selftest of programmer**

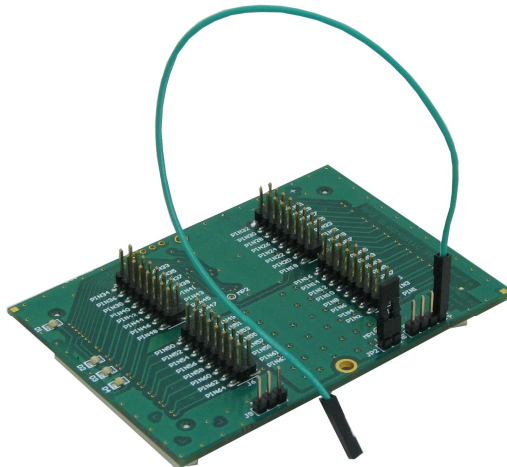
- Put together Board A with Board B of AP3 diagnostic POD
- Insert **AP3 diagnostic POD** into Programming Module Interface (PMI) connectors of the programmer site.
- Run selftest of programmer in PG4UW (menu Programmer / Selftest).



AP3 diagnostic POD for Selftest plus

### **Calibration test**

- Disconnect **Board A** from **Board B** of **AP3 diagnostic POD**
- Insert **Board A** of **AP3 diagnostic POD** into Programming Module Interface (PMI) connectors of the programmer.
- Run calibration test of programmer in PG4UW (menu Programmer / Calibration test).



AP3 diagnostic POD for Calibration test

# Technical specification

## HARDWARE

### Base unit, DACs

- USB 2.0 high-speed compatible port, up to 480 Mb/s transfer rate
- 100Mbit LAN port
- on-board intelligence: powerful processor (ARM9 400MHz) and FPGA based state machine (basic clock 50 MHz plus PLL)
- built-in mSATA SSD as internal buffer (128GB, upgradeable to higher capacity) (\*)
- three D/A converters for VCC1, VCC2, and VPP, controllable rise and fall time
- VCC1, VCC2 range 0.8V..7V/1A (step 10mV)
- VPP range 0V..25V/1A (step 25mV)
- Temperature controlled fan
- selftest capability
- protection against surge and ESD on power supply input, USB and LAN port and pins of Programming Module Interface (PMI)(IEC1000-4-2: 15kV air, 8kV contact)
- banana jack for ESD wrist straps connection
- banana jack for connection to ground

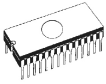
**(\*) Note:**

- *the data in the buffer are stored in fragments and also partially compressed, therefore the buffer can keep standard data also for bigger size devices, than current size of the buffer*
- *this upgrade is mechanically complicated and can be done at ElneC only*
- *the programmers up to s/n 1200-00053 (including) were supplied with 32GB SSD as internal buffer*

*the 128GB SSD can store as a buffer roughly 110 GB of the random data (28 GB in case of 32 GB SSD) for the bigger sizes SSD, proportionally. If you plan to work/copy eMMC/NAND device, which might contain random data in full range of the device size, we recommend to have equal capacity buffer in the programmer*

### **Pindriver (available on the programming module interface - connectors for programming modules)**

- pindrivers: 64 universal
- VCC1/VCC2 and VPP can be connected to each pin
- perfect ground for each pin
- 2 independent FPGA based TTL driver provides H, L, CLK, pull-up, pull-down on all pindriver pins, logic level 0,75V - 5V (IOL and IOH current 20mA)
- logic signals frequency: up to 125MHz (3.3V), 80MHz (5V)
- analog pindriver output level selectable from 0.8 V up to 25V
- current limitation, overcurrent shutdown, power failure shutdown
- continuity test: each pin is tested before every programming operation

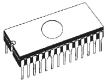


## DEVICE SUPPORT

### ***Programmer, using programming modules:***

- NAND FLASH: Samsung K9xxx, KFxxx, SK Hynix (ex Hynix) HY27xxx, H27xxx, Toshiba TC58xxx, TH58xxx, Micron MT29Fxxx, (ex Numonyx ex STM) NANDxxx, Spansion S30Mxxx, S34xxx, 3D-Plus 3DFNxxx, ATO Solution AFNDxxx, Fidelix FMNDxxx, Eon Silicon Sol. EN27xxx, ESMT F59xxx, LBA-NAND Toshiba THGVNxxx
- serial NAND FLASH: Micron MT29Fxxx, GigaDevice GD5Fxxx
- eMMC: Hynix H26Mxxxxxxx, Kingston KE44B-xxx/xxx, Micron MTFCxxxxxx, Numonyx NANDxxxxxxx, Phison PSM4A11-xx, Samsung KLMxxxxxxx, SanDisk SDINxxx-xx, Toshiba THGBMxxxxxxx
- Multi-chip devices: NAND+RAM, NOR+RAM, NOR+NOR+RAM, NAND+NOR+RAM
- Serial Flash: standard SPI, high performance Dual I/O SPI and Quad I/O SPI (25Bxxx, 25Dxxx, 25Exxx, 25Fxxx, 25Lxxx, 25Mxxx, 25Pxxx, 25Qxxx, 25Sxxx, 25Txxx, 25Uxxx, 25Vxxx, 25Wxxx, 25Xxxx, 26Vxxx, 45PExx, MX66Lxxx, S70FLxxx), DataFlash (AT45Dxxx, AT26Dxxx)
- parallel NOR Flash: 28Fxxx, 29Cxxx, 29Fxxx, 29GLxxx, 29BVxxx, 29LVxxx, 29Wxxx, 49Fxxx series, Samsung's K8Fxxx, K8Cxxx, K8Sxxx, K8Pxxx series, ...
- EPROM: NMOS/CMOS, 27xxx and 27Cxxx series
- EEPROM: NMOS/CMOS, 28xxx, 28Cxxx, 27EExxx series, 3D Plus 3DEExxxxxxx
- mDOC H3: SanDisk (ex M-Systems) SDED5xxx, SDED7xxx, MD2533xxx, MD2534xxx, Hynix HY23xxx
- FRAM: Ramtron
- MRAM: Everspin MRxxxxx8x, 3D Plus 3DMRxxxxxxx
- NV RAM: Dallas DSxxx, SGS/Inmos MKxxx, SIMTEK STKxxx, XICOR 2xxx, ZMD U63x series
- Serial E(E)PROM: Serial E(E)PROM: 11LCxxx, 24Cxxx, 24Fxxx, 25Cxxx, 30TSExxx, 34Cxxx, 34Tsx, 59Cxxx, 85xxx, 93Cxxx, NVM3060, MDAxxx series, full support for LV series, AT88SCxxx
- Serial FRAM: Cypress(Ramtron): FM24xxxxxx, FM25xxxxxx, Fujitsu: MB85RCxxx, MB85RSxxx, Lapis(OKI, Rohm): MR44xxxxxx, MR45xxxxxx
- Serial MRAM: Everspin MH20xxx, MH25xxx
- Configuration (EE)PROM: XCFxxx, XC17xxxx, XC18Vxxx, EPCxxx, EPCSxxx, AT17xxx, AT18Fxxx, 37LVxx
- 1-Wire E(E)PROM: DS1xxx, DS2xxx
- PLD Altera: MAX 3000A, MAX 7000A, MAX 7000B, MAX 7000S, MAX7000AE, MAX II/G/Z, MAX V
- PLD Lattice: ispGAL22V10x, ispLSI1xxx, ispLSI1xxxEA, ispLSI2xxx, ispLSI2xxxA, ispLSI2xxxE, ispLSI2xxxV, ispLSI2xxxVE, ispLSI2xxxVL, LC4xxxB/C/N/ZC/ZE, M4-xx/xx, M4A3-xx/xx, M4A5-xx/xx, M4LV-xx/xx, ispCLOCK, Power Manager/II, ProcessorPM
- 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: FPGA: Microsemi(Actel): ProASIC3, IGLOO, Fusion, ProASICplus, SmartFusion
- FPGA: Lattice: MachXO, MachXO2, LatticeXP, LatticeXP2, ispXPGA
- FPGA: Xilinx: Spartan-3AN
- Clocks: TI(TMS), Cypress

- Special chips: Atmel Tire Pressure Monitoring ATA6285N, ATA6286N; PWM controllers: Ziker Labs, Analog Devices; Multi-Phase ICs: IR(Chil Semiconductor); Gamma buffers: AUO, Maxim, TI, ...
- Microcontrollers MCS51 series: 87Cxxx, 87LVxx, 89Cxxx, 89Sxxx, 89Fxxx, 89LVxxx, 89LSxxx, 89LPxxx, 89Exxx, 89Lxxx, all manufacturers, Philips LPC series
- Microcontrollers Atmel ARM. AT91SAM7Sxx, AT91SAM7Lxx, AT91SAM7Xxx, AT91SAM7XCxx, AT91SAM7SExx series;
- Microcontrollers Atmel ARM9: AT91SAM9xxx series;
- Microcontrollers ARM Cortex-M3: ATSAM3Axxx, ATSAM3Uxxx, ATSAM3Nxxx, ATSAM3Sxxx, ATSAM3D20, ATSAM3Xxxx series
- Microcontrollers ARM Cortex-M4: ATSAM4Sxxx series
- Microcontrollers Atmel AVR 8bit/16bit: AT90Sxxxx, AT90pwm, AT90can, AT90usb, ATtiny, ATmega, ATmega series
- Microcontrollers Atmel AVR32: AT32UC3xxxx, ATUCxxxD3/D4/L3U/L4U series
- Microcontrollers TI (Chipcon): CC11xx, CC24xx, CC25xx, CC85xx series
- Microcontrollers Coreriver: Atom 1.0, MiDAS1.0, 1.1, 2.0, 2.1, 2.2, 3.0 series
- Microcontrollers Cypress: CY7Cxxxxx, CY8Cxxxx
- Microcontrollers ELAN: EM78Pxxx
- Microcontrollers EPSON: S1C17 series
- Microcontrollers Explore Microelectronic: EPF01x, EPF02x series
- Microcontrollers Generalplus: GPM8Fxxx series
- Microcontrollers GreenPeak: GPxxx series
- Microcontrollers Infineon(Siemens): XC800, C500, XC166, C166 series
- Microcontrollers MDT 1xxx and 2xxx series
- Microcontrollers Megawin: MG87xxx, MPC82xxx series
- Microcontrollers Microchip PICmicro: PIC10xxx, PIC12xxx, PIC16xxx, PIC17Cxxx, PIC18xxx, PIC24xxx, dsPIC, PIC32xxx series
- Microcontrollers Motorola/Freescale: HC05, HC08, HC11, HC12, HCS08, RS08, S12, S12X, MC56F, MCF51, MCF52 series, Kinetis (K,L), Qorivva/5xxx Power Architecture
- Microcontrollers Myson 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 Flash and OTP series
- Microcontrollers Nuvoton ARM Cortex-Mx: NUC1xx, M05x, Mini51, Nano1xx series
- Microcontrollers Nuvoton (Winbond): N79xxx, W77xxx, W78xxx, W79xxx, W83xxx series
- Microcontrollers NXP (Philips) ARM Cortex-Mx: LPC11xx, LPC11Cxx, LPC11Dxx, LPC11Uxx, LPC12xx, LPC12Dxx, LPC13xx, LPC17xx, LPC11Axx, LPC11Exx, LPC11xxLV, LPC18xx, LPC43xx, LPC8xx, EM7xx, series
- Microcontrollers NXP (Philips) UOC series: UOCIII, UOC-TOP, UOC-Fighter (TDA1xxxx) series
- Microcontrollers NXP (Philips) ARM7: LPC2xxx, MPT6xx, PCD807xx, SAF7780xxx series
- Microcontrollers NXP (Philips) ARM9: LPC31xx series
- Microcontrollers Pasat: TinyModule DIL40, DIL50 series
- Microcontrollers Scenix (Ubicom): SXxxx series
- Microcontrollers Syntek: STK6xxx series
- Microcontrollers Renesas: R8C/Tiny series, RX series, uPD70Fxxx, uPD78Fxxx series, RL78 series, R32C series



- Microcontrollers SyncMOS: SM39xxx, SM59xxx, SM73xxx, SM79xxx, SM89xxx series
- Microcontrollers & Programmable System Memory STMicroelectronics: uPSD, PSD series
- Microcontrollers STM (ex SGS-Thomson): ST6xx, ST7xx, ST10xx, STR7xx, STR9xx, STM32F/LW, STM8A/S/L series, SPC5 (Power Architecture)
- Microcontrollers Silicon Laboratories(Cygnal): C8051 series
- Microcontrollers Silicon Laboratories(Energy Micro): EFM32Gxx, EFM32GGxx, EFM32LGxx, EFM32TGxx, EFM32WGxx series
- Microcontrollers Silicon Laboratories: SiM3Cxxx, SiM3Lxxx, SiM3Uxxx series
- Microcontrollers Texas Instruments: MSP430 series, MSC12xx series, TMS320F series, CC430 series,
- Microcontrollers Texas Instruments (ex Luminary Micro): LM3Sxxx, LM3Sxxxx series, LM4Fxxxx series, TM4C series
- Microcontrollers ZILGO: Z86/Z89xxx and Z8Fxxxx, Z8FMCxxxxx, Z16Fxxxx, ZGP323xxxxxxx, 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, QIXIN, Signetic, Tekmos, Weltrend, Amic, Cyrod Technologies, Ember, Ramtron, Nordic Semiconductor, Samsung, ABOV Semiconductor...

**Notes:**

- For all supported devices see actual **Device list** on [www.elnec.com](http://www.elnec.com)

## Package support

- 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..

## Programming speed

**Notes:**

- It is important to say, we always use random numbers data pattern for programming speed testing. Some our competitors use "sparse" data pattern, where only small amount of non-blank data are programmed or there are used data with only few 0 bits (FE, EF, etc.). This cheating approach can "decrease" programming time considerably. If you plan to compare, ask always which pattern they use.
- The programming speed practically doesn't depend on PC type because data for programming and main part of programming algorithm are stored internally inside of the programmer.
- All devices mentioned below, including both NAND Flash, are programmed at their maximal speed, the programming time can not be shorter.
- We're sorry, but there exist not very much devices, where the 20+MB/s BeeProg3 programming speed can be utilized

Device	Size [bits]	Operation	Time
JS28F00AM29EWH (parallel NOR Flash)	4000080hx16 (1 Giga)	programming and verify	10 sec
MT29F1G08ABAEAWP (parallel NAND Flash)	8400000hx8 (1 Giga)	programming and verify	19.5 sec
SDIN7DP2-8G (eMMC NAND FLASH)	1D2000000hx8 (64Giga)	programming *1	342 sec
S25FL164K (serial Flash)	800300hx8 (64 Mega)	programming and verify	30.7 sec
AT89LP51RD2 (microcontroller)	10000hx8	programming and verify	5.2 sec
PIC32MX360F512L (microcontroller)	80000hx8	programming and verify	8.9 sec



**Conditions:** *Core2 Duo, 3.16 GHz, 1G RAM, USB 2.0 HS, Windows 7. Version of SW: 3.09, 10/2014.*

*\*1 Verification of programming is done by internal controller of eMMC device. The device receives a block of data plus CRC, if it all matches, the internal controller confirm the proper programming.*

## SOFTWARE

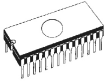
- **Algorithms:** only manufacturer approved or certified algorithms are used. Custom algorithms are available at additional fee.
- **Algorithm updates:** software updates are available regularly, approx. every 4 weeks, free of charge. **OnDemand** version of software is available for highly needed chips support and/or bugs fixes. Available nearly daily, depending on request.
- **Main features:** revision history, session logging, on-line help, device and algorithm information

## Device operations

- **standard:**
  - intelligent device selection by device type, manufacturer or typed fragment of part name
  - automatic ID-based selection of EPROM/Flash EPROM
  - blank check, read, verify
  - program
  - erase
  - configuration and security bit program
  - 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 (automatic start immediately after device insertion)
  - multi-project mode
  - lot of serialization modes (more type of incremental modes, from-file mode, custom generator mode)
  - statistic
  - count-down mode

## Buffer operations

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



## **File load/save**

- automatic file type identification

## **Supported file formats**

- unformatted (raw) binary
- HEX: Intel, Intel EXT, Motorola S-record, MOS, Exoramax, Tektronix, ASCII-SPACE-HEX,, ASCII 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**

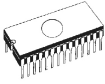
- operating voltage 100-240V AC rated, 90-264 VAC max., 47-63Hz
- power consumption max. 25W active
- **dimensions of BeeProg3** programmer: 139,5 x 189,5 x 48,4 mm (5,5 x 7,5 x 1,9 inch). *Dimensions were measured without programming module inserted and does not include projections. Total height of BeeProg3 programmer with programming module inserted depends on ZIF socket height and can vary between 66-78mm.*
- weight of programmer without programming modules: 1,25kg (2,7 lb)
- operating temperature: 5°C ÷ 40°C (41°F ÷ 104°F)
- operating humidity: 20%..80%, non condensing



---

# *Setup*

---



The programmer package contains a CD with the control program, useful utilities and additional information. The permission to freely copy the content of the CD is granted in order to demonstrate how Elnec programmers work.

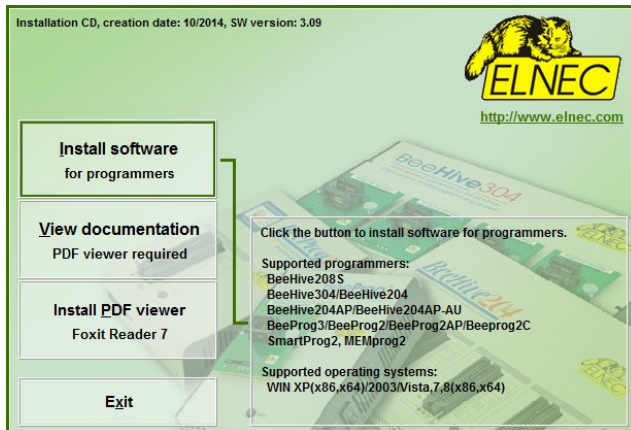
For programmers connected through USB port, control program requires correctly installed USB driver

**We recommended installing software before connecting programmer to PC to avoid unwanted complication during installation.**

## Software setup

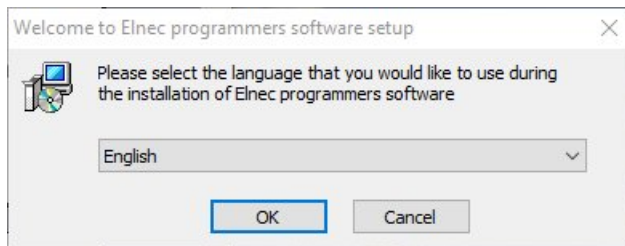
Insert delivered CD to your CD drive and install program starts automatically (if not, run setup.exe). Install program will guide you through the installation process and will do all the necessary steps before you can first run the control program.

### Step 1.



Click on "Install software for programmers" button.

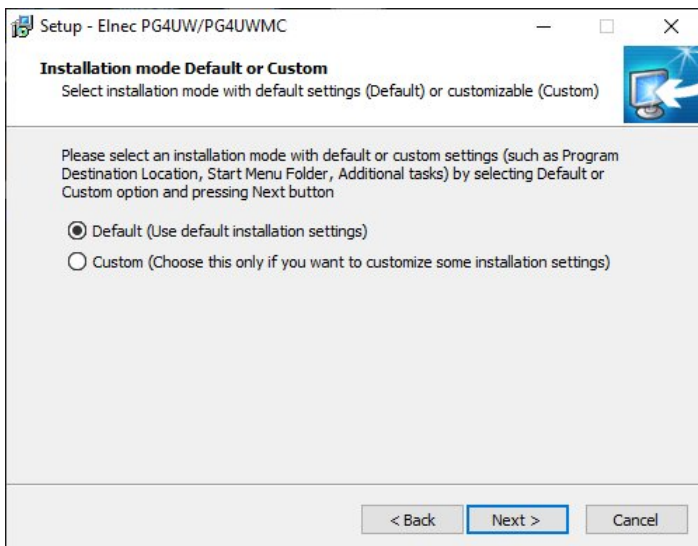
### Step 2.



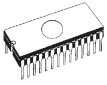
Select language and then click on "OK" button.

**Step 3.**

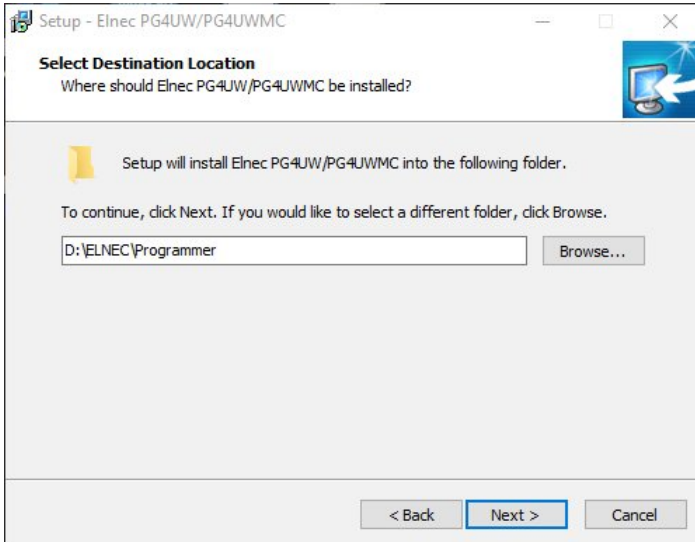
Click on "Next" button

**Step 4.**

For default setting you click on "Next" button. Setup will be continuing with Step 6. For change default setting you click on "Custom" and then on "Next" button.

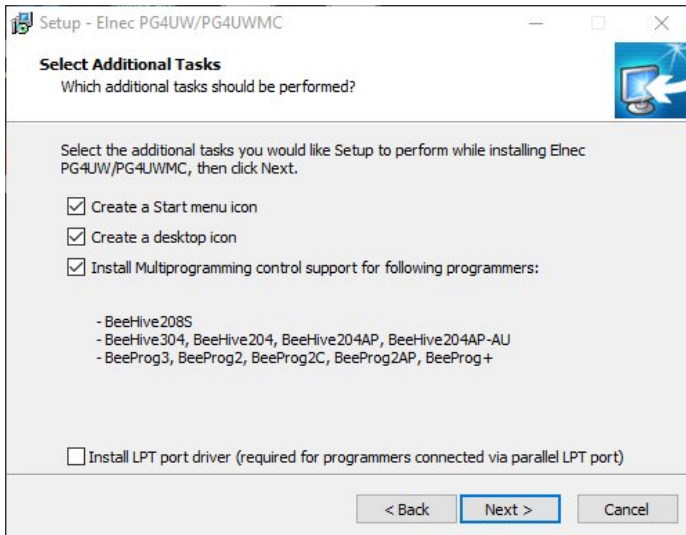


**Step 5.**

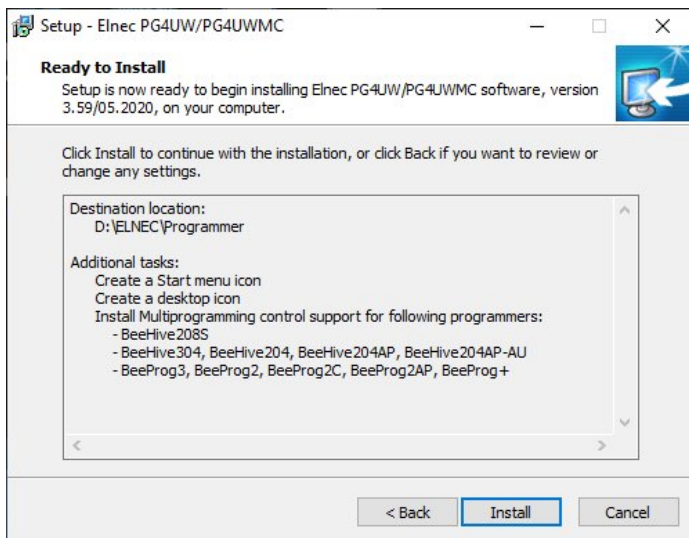


To change default folder click on “Browse” button, select the destination folder.  
Then click on “Next” button

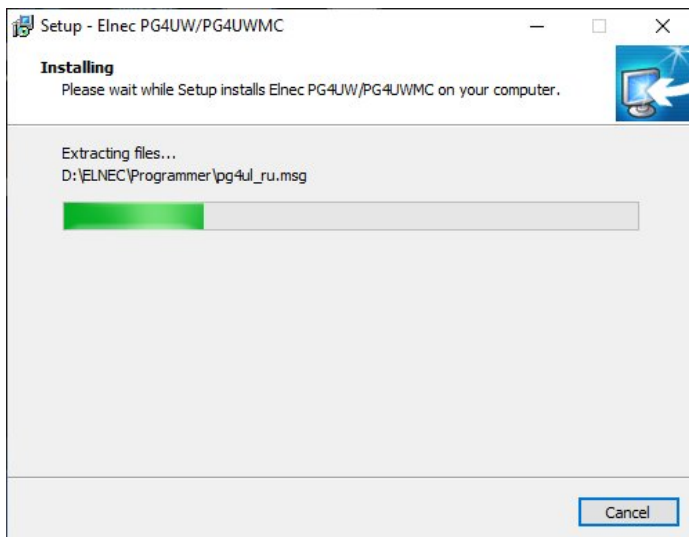
**Step 6.**



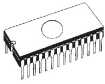
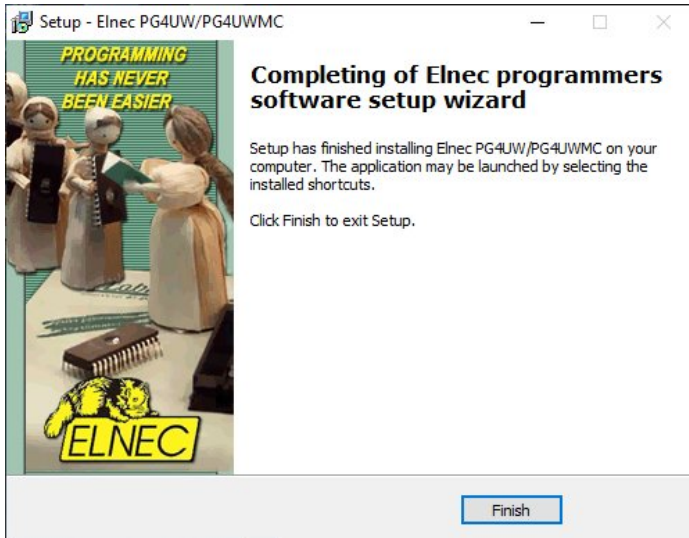
Check if “Install Multiprogramming control support” is selected.  
Change default setting, if you want. Then click on “Next” button

**Step 7.**

Check your settings and then click on "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 programmer we recommend using the latest version of PG4UW. You may download the latest version of programmer software (file pg4uwarc.exe) from our Internet site [www.elnec.com](http://www.elnec.com), part download.

Copy pg4uwarc.exe to a temporary directory, disconnect programmer from PC and then launch it. Setup will start with **Step 2** from previous chapter.

## ***Hardware setup***

**Warning:** *Because of high programmer's communication traffic, we recommend to connect each programmer to separated USB 2.0 High speed controller (USB EHCI). Most of new PC motherboards have two or more EHCI controller integrated in chipset. If not, you can use PCI (PCI-E) USB add-on card (Renesas USB chipset is recommended). If the EHCI integrated in motherboard chipset is used, consult the motherboards manual or motherboard manufacturer tech support for USB ports mapping so you will be able connect each programmer to separated EHCI. In generally, we also recommend connect the programmers directly to PC's USB ports (without USB HUB) and preferable to the USB ports mounted on the motherboard directly (mostly located on the rear side of the PC).*

When the programmer is connected to USB port before control program was installed, Windows will detect new hardware and ask user to select driver installation method: automatically or manually. To detect programmer correctly, control program installation CD must be inserted to computer's CD-ROM drive and following steps have to be done:



**Step 1.**

Directly connect USB cable to type B USB port on programmer.

**Step 2.**

Directly connect USB cable to type A USB2.0 port on PC.

**Step 3.**

Connect connectors of power supply cable to appropriate connectors on programmer and wall plug.

**Step 4.**

Turn on programmer. At this time all 'work result' LEDs will be blinking approximately 30 sec (OS startup) and then LEDs will switch off.

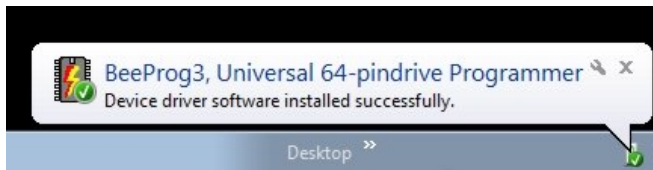
**For Windows 7 and Windows 8.**

**Step 5.**

In the notification area at task bar (mainly at lower right corner) you will see following notification bubble:

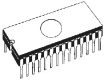


After successfully installed driver for programmer you will see



**Note:** if another programmer will be connected to PC (maybe to the same USB port) "Installing device driver software" will launch again. If the same programmer will be connected to other USB port, there is no needed for any additional driver installation.

**For Windows 10:** No information or bubble is displayed.

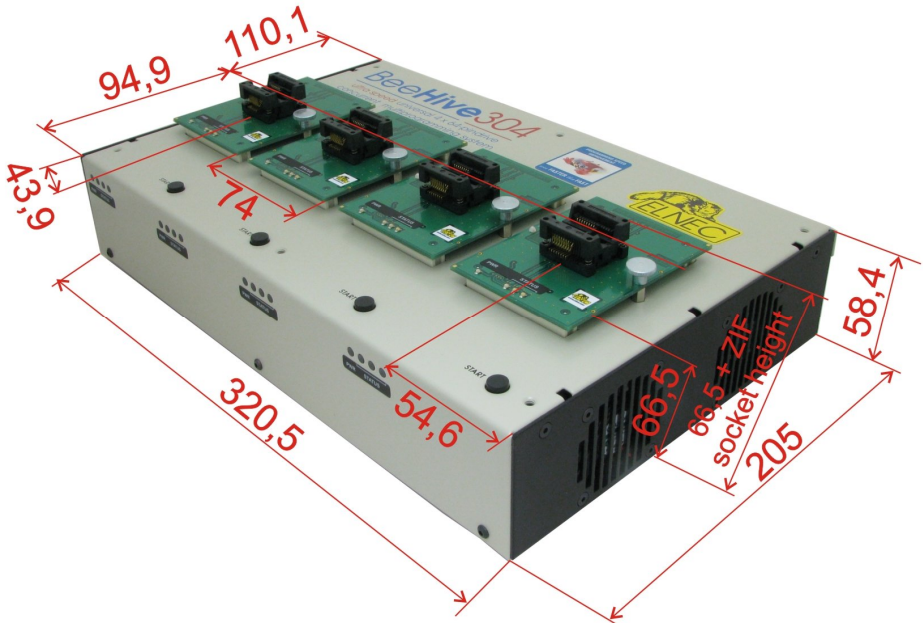


## Installation of the BeeHive304/BeeProg3 programmer into the target system

This chapter contains mechanical drawing of BeeHive304 and BeeProg3 and other information needed for installation BeeHive304 and BeeProg3 to target system.

### BeeHive304

In the picture below are overall dimensions of BeeHive304 with programming modules. Total height of programmer with installed programming modules depends on programming module ZIF socket height. Total height can be determined by **66,5mm + ZIF socket height**.



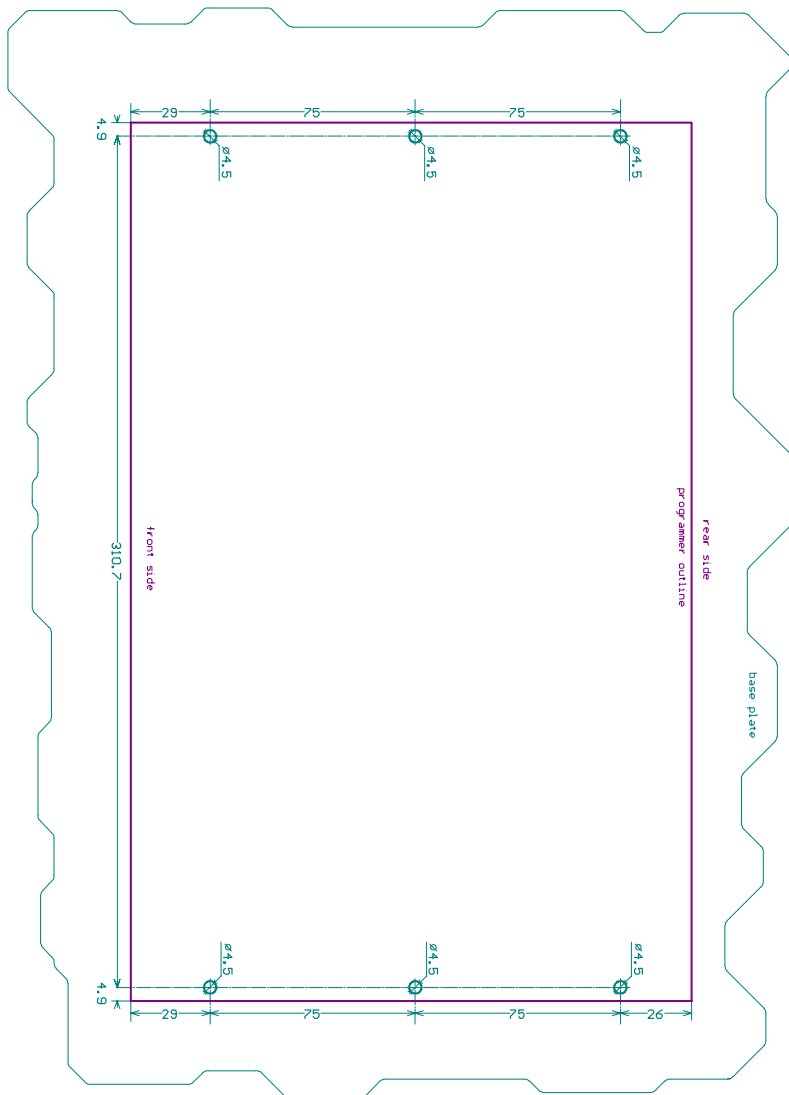
Right top view to BeeHive304 with dimensions

**Note:** Minimal distance between BeeHive304 and other object is 2cm. It is necessary for good cooling of BeeHive304 and air ventilation around it.

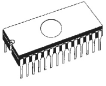
When you need replace used or worn AP3 module by other (but same type), position of ZIF center is same (with 0,1mm tolerance) only if AP3 module is fixed by fixating screw. A center of ZIF is same with center of programming module at X and Y axis.

**Warning:** If you integrated BeeHive304 to automated programming machine and BeeHive304 is mounted on moving part (programmer is not static) then use high flexibility USB cable for connect programmer to PC. External programmer power supply must be mounted on same part as programmer and connected by high flexibility power cord set to mains.

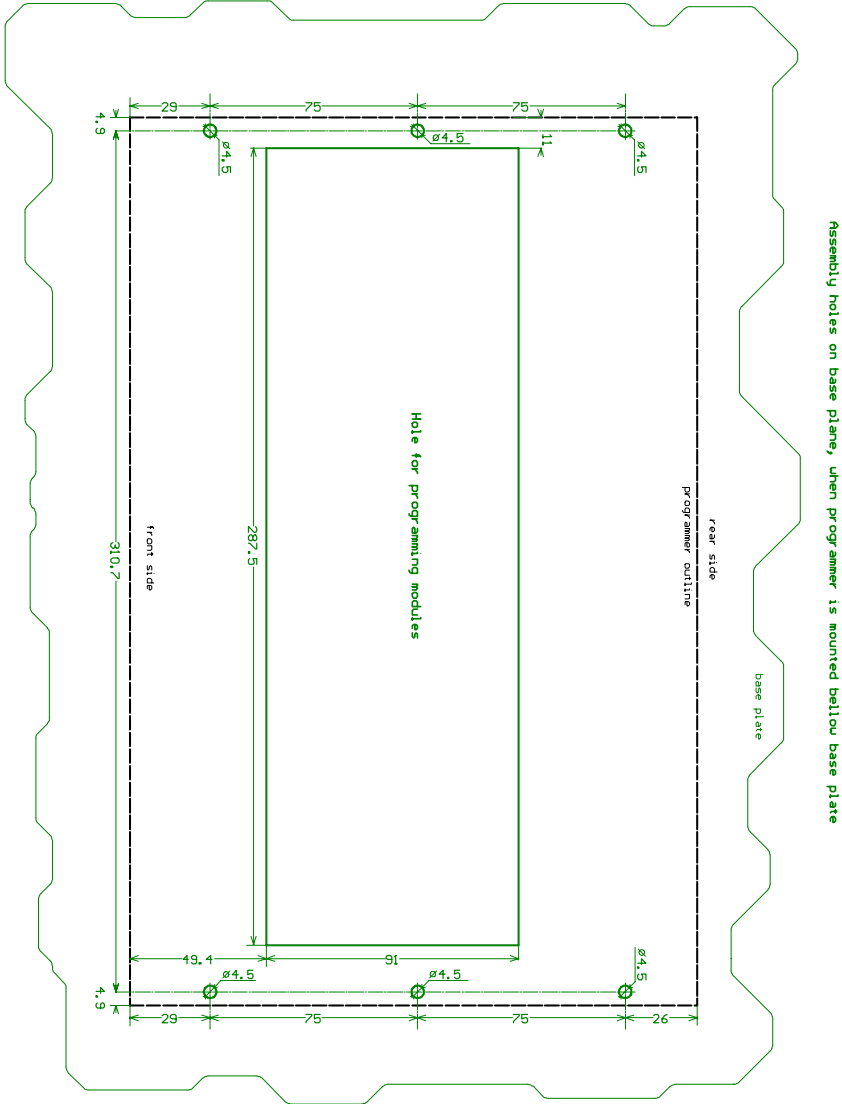
**BeeHive304** can be mounted on base plate by 6 screws M4 with nuts. Length of screw depends on base plate thickness. On base plate can be  $\varnothing 4,5$ mm holes replaced with M4 internal threads. In this case you may use six M4x70 screws which are at standard delivery. Drawing for mounting programmer on base plate:



Assembly holes on base plate, when programmer is mounted on base plate

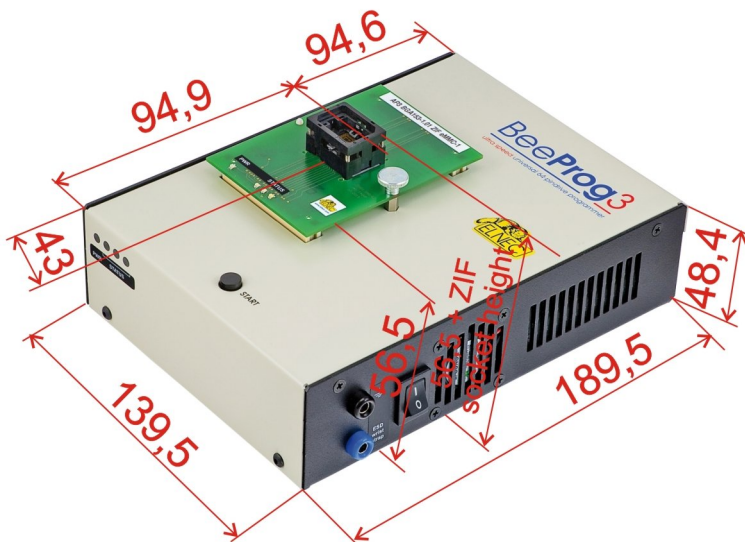


**BeeHive304** can be mounted below base plate 6 screws M4 with nuts. Length of screw depends on base plate thickness. On base plate can be  $\varnothing 4,5$ mm holes replaced with M4 internal threads. In this case you may use six M4x70 screws which are at standard delivery. Drawing for mounting programmer below base plate:



## BeeProg3

In the picture below are overall dimensions of BeeProg3 with programming module. Total height of programmer with installed programming modules depends of programming module ZIF socket height. Total height can be determined by **56,5mm + ZIF socket height**.

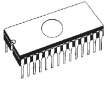


Right top view to BeeProg3 with dimensions

**Note:** Minimal distance between BeeProg3 and other object is 2cm. It is necessary for good cooling of BeeProg3 and air ventilation around it.

When you need replace used or worn AP3 module by other (but same type), position of ZIF center is same (with 0,1mm tolerance) only if AP3 module is fixed by fixating screw. A center of ZIF is same with center of programming module at X and Y axis.

**Warning:** If you integrated BeeProg3 to automated programming machine and BeeProg3 is mounted on moving part (programmer is not static) then use high flexibility USB cable for connect programmer to PC. External programmer power supply must be mounted on same part as programmer and connected by high flexibility power cord set to mains.

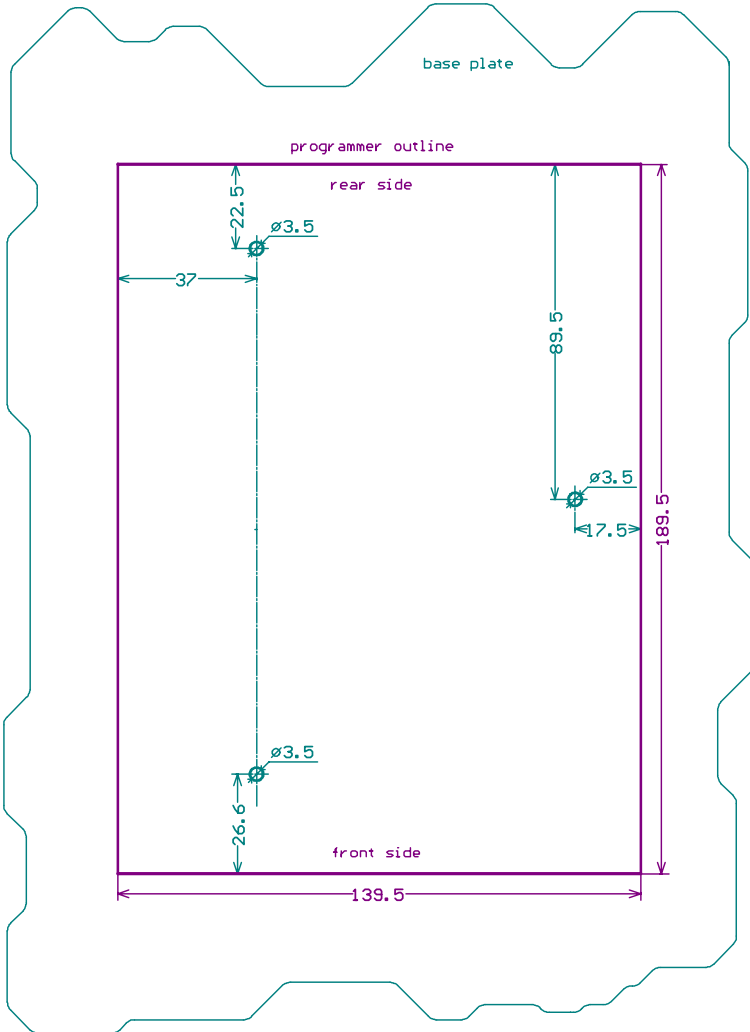


**BeeProg3** can be mounted on base plate by 3 screws M3. Length of screw depends on base plate thickness.

**Attention:** For proper mounting of programmer on base plate, mounting screws must be minimal 3mm depth in programmer. To avoid damage of PCB in programmer mounting screws must be maximum 8mm depth in programmer.

Drawing for mounting programmer on base plate:

Assembly holes on base plane, when programmer is mounted on base plate

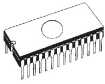




---

***PG4UW***

---



## ***PG4UW the programmer software***

Program PG4UW is common control program for all Elnec programmers, which works with all versions of MS Windows from Windows XP to Windows 11, 32-bit and 64-bit.

### ***Using the programmer software***

*The control program delivered by Elnec, included on the CD in your package, is granted to be free from any viruses at the moment of delivery. To increase their safety our programs include a special algorithm for detecting possible virus infections.*

### ***Run the control program***



In Windows environment: double click on icon PG4UW.

After start, control program PG4UW automatically scan all existing ports and search for the connected any Elnec programmer. Program PG4UW is common for all the Elnec programmers, hence program try to find all supported programmers.

**Note:** *When PG4UW is started, program is checked for its integrity. Than the program display a 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 error code and description of possible reasons (disconnected programmer, bad connection, power supply failure, incompatible printer port...). Eliminate the error source and press any key. If error condition still exists, the program resumes its operation in the demo mode and access to the programmer is not possible. If you cannot find the cause of the error, follow the instructions in **Troubleshooting** section. In addition, the control program checks communication with programmer prior to any operation with the programmed device.

### ***Buffer synchronization for programmer with internal data disk***

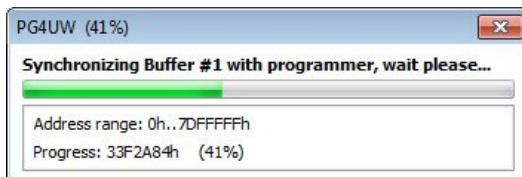
Buffer synchronization operation is automatically performed for advanced programmers with internal data disk in following situations:

- before the first device operation
- after device read operation
- after selecting new device
- after loading project
- after (re)finding of programmer in PG4UW control program

**Buffer synchronization** copies required data for device from PC (host computer running PG4UW) buffer to internal programmer disk (or from programmer disk to PC after device reading). This feature significantly increases performance of device operations Program/Verify, because no data are transferred during device operations, whole programming/verifying



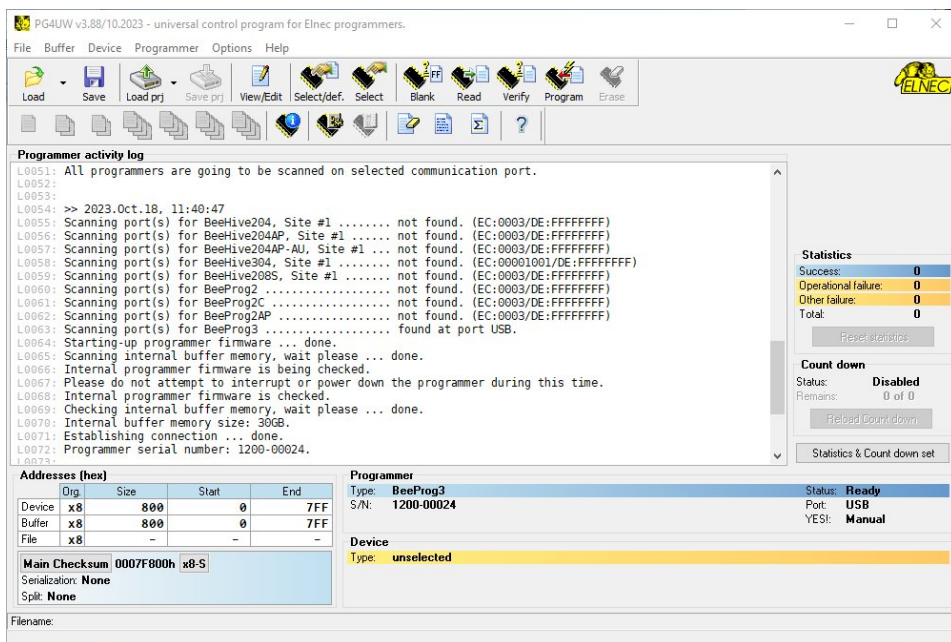
operation is running inside programmer. Buffer synchronization is indicated in PG4UW control program by message **"Synchronizing buffers with programmer..."**.



Progress of the data transfer is showed in separate window. Transfer can be canceled by pressing key <Esc> or pressing the window's exit button.

## Description of the user screen

Windows program PG4UW



PG4UW v3.88/10.2023 - universal control program for Elnecc programmers.

File Buffer Device Programmer Options Help

Load Save Load prj Save prj View/Edit Select/def. Select Blank Read Verify Program Erase

**Programmer activity log**

```

L0051: ALL programmers are going to be scanned on selected communication port.
L0052:
L0053:
L0054: >> 2023.Oct.18, 11:40:47
L0055: Scanning port(s) for BeeHive204, Site #1 ..... not found. (EC:0003/DE:FFFFFFFF)
L0056: Scanning port(s) for BeeHive204AP, Site #1 ..... not found. (EC:0003/DE:FFFFFFFF)
L0057: Scanning port(s) for BeeHive204AP-AU, Site #1 ... not found. (EC:0003/DE:FFFFFFFF)
L0058: Scanning port(s) for BeeHive304, Site #1 ..... not found. (EC:00001001/DE:FFFFFFFF)
L0059: Scanning port(s) for BeeHive208S, Site #1 ..... not found. (EC:0003/DE:FFFFFFFF)
L0060: Scanning port(s) for BeeProg2 ..... not found. (EC:0003/DE:FFFFFFFF)
L0061: Scanning port(s) for BeeProg2C ..... not found. (EC:0003/DE:FFFFFFFF)
L0062: Scanning port(s) for BeeProg2AP ..... not found. (EC:0003/DE:FFFFFFFF)
L0063: Scanning port(s) for BeeProg3 ..... found at port USB.
L0064: Starting-up programmer firmware ... done.
L0065: Scanning internal buffer memory, wait please ... done.
L0066: Internal programmer firmware is being checked.
L0067: Please do not attempt to interrupt or power down the programmer during this time.
L0068: Internal programmer firmware is checked.
L0069: Checking internal buffer memory, wait please ... done.
L0070: Internal buffer memory size: 36GB.
L0071: Establishing connection ... done.
L0072: Programmer serial number: 1200-00024.
L0073:
    
```

**Statistics**

Success: 0  
Operational failure: 0  
Other failure: 0  
Total: 0

Reset statistics

**Count down**

Status: Disabled  
Remains: 0 of 0

Reload Count-down

Statistics & Count down set

**Addresses (hex)**

	Org.	Size	Start	End
Device	x8	800	0	7FF
Buffer	x8	800		7FF
File	x8	-	-	-

Main Checksum 0007F800h x8-S  
Serialization: None  
Split: None

Filename:

**Programmer**

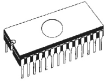
Type: BeeProg3  
S/N: 1200-00024  
Status: Ready  
Port: USB  
YES!: Manual

**Device**

Type: unselected

## Toolbars

Under main menu are placed toolbars with button shortcuts of frequently used menu commands. Toolbars are optional and can be turned off by menu command **Options / View**.



### Log window

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

Operation can be:

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

Content of Log window can be saved to file concurrently while information is written to Log window. This option can be set by menu **Options / General options** (and tab *Log file* in dialog *General options*).

### Panel Addresses

Panel Addresses contains information about actual address ranges of currently selected device, loaded file and buffer start-end address settings. Some devices allow modifying default device and buffer address ranges by menu command **Device / Device options / Operation options**.

Panel Addresses also contains some advanced information about current status of Split, Serialization and buffer checksum. For more information about each of the options, please look at:

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

### Panel Programmer

Panel Programmer contains information about currently selected programmer.

The information includes

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

For more details please look at **Programmer / Automatic YES!**.

### Panel Device

It contains information about currently selected device.

The information includes

- device name (type) and manufacturer
- device adapter needed to use with currently selected programmer
- reference to detailed *Device info* dialog, available also by menu **Device / Device info**



- reference to *Advanced device options* - this is available for some types of devices only

### Panel Statistics

It contains statistics information about currently selected device.

The information includes

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

Statistics and count-down options are available by menu command **Device / Device options / Statistics** or by mouse right click on panel *Statistics* and select item *Statistics* from popup menu

### Panel File

The panel is placed on the bottom of PG4UW main window. Panel shows 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 buffer
<F5>	Select/default	Target-device selection from 10 last selected devices list
<Alt+F5>	Select/manual	Target-device selection by typing device/vendor name
<F6>	Blank	Blank check
<F7>	Read	Reads 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 saving settings too
<Ctrl+F1>		Displays additional information about current device
<Ctrl+F2>	Erase	Fill's the buffer with a given value
<Ctrl+Shift+F2>		Fill's the buffer with random values.

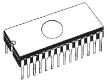
## File

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

### File / Load

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

The reserved key <F3> will bring out this menu from any menu and any time.



## File formats description:

### 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 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 an 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

Very simple hex file format similar as ASCII HEX without 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 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

Very simple hex file format similar as ASCII HEX without 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 other data bytes.

Here is an example of 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 minor modification of Intel HEX format, therefore in the software is recognized and indicated as Intel HEX file format.

### Note for special x16 formats:

*Intel HEXx16 is Intel Hex file format with 16 bits data word for TMS320F devices.*

*Motorola HEXx16 is Motorola file format with 16 bits data word for TMS320F devices.*

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

When the check box **Automatic file format recognition** is unchecked program allows user to manually select wished file format from list of available file formats on panel **Selected file**



**format.** Default set is from **Options / General options** in panel **Load file format** at tab **File options**.

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

### Panel Additional operation

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

- **Autodetect** buffer will be erased (filled) with default "blank" value for recently selected device and buffer. Recent default value is displayed after "Autodetect" text in brackets, for example: "Autodetect (FFh)", "Autodetect (00h)"
  - **Custom defined** buffer will be erased (filled) with user specified Byte value
- Buffer erase is performed immediately before reading file content to buffer and it is functional for binary and all HEX file formats. Using this one-shot setting disables current setting of **Erase buffer before loading** option in menu **Options / General options** at tab **Hex file options**.

If the checkbox **Swap bytes** is displayed, the user can activate function of swapping bytes within 16bit words (or 2-byte words) during reading of file. This feature is useful especially when loading files with Motorola representation of byte order in file (big endian). Standard load file is using 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). Number 4F52H is stored in memory:*

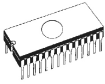
Address	big endian system	little endian system
1000H	4FH	52H
1001H	52H	4FH

The **View/Edit buffer** in 16-bit mode show real view to 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 load to relevant position in buffer (dependent on selected device).

### Panel Buffer offset for loading

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



Available offset options are:

- None** this setting means, no offset is applied for loading data from file to 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** mode has two options:  
**Negative offset** and **Automatic negative offset** - set by two ways: manual or automatic.  
For manual set use option Negative offset and put wished offset value to its edit box.  
For automatic offset detection use option Automatic negative offset. This value is subtracted from current address for save data to buffer.

Negative offset value (manually defined or automatically detected) is subtracted from current buffer address for store data to buffer.

Negative offset is applied only for all HEX file formats and is using always x8 format. Negative offset settings are ignored for binary files and other non-HEX files.

#### Notes:

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

#### Example for negative offset using:

A file contains data by Motorola S - format.

A data block started at address FFFF0H.

It is a S2 format with length of address array of 3 bytes.

For all data reading you can set Negative offset option and value of negative offset to FFFF0H.

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

#### List of file format codes and error codes

There are some errors can occur during file download in some of supported formats. The error is written to LOG window in face "Warning: error #xyy in line rrr", xx is file format code, y is error code and rrr is 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. The file format of saved file can be chosen from supported formats list box. There can be also entered the Buffer start and Buffer end addresses which exactly specify part of buffer to save to 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 function of swapping bytes within 16bit words (or 2-byte words) during writing to file. This feature is useful especially when saving files with Motorola representation of byte order in file (big endian). Standard save file operation is using little endian byte order.

The reserved key <F2> will bring out this menu from any menu and any time.

## File / Load project

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

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

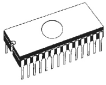
Project information consists of:

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

**Note:** for projects with serialization turned on

*Serialization is read from project file by following procedure:*

1. *Serialization settings from project are accepted*
2. *Additional serialization file search is performed. If the file is found it will be read and serialization settings from the additional file will be accepted. Additional serialization file is*



always associated to the specific project file. When additional serialization file settings are accepted, project serialization settings are ignored.

Name of additional serialization file is derived from project file name by adding extension ".sn" to project file's name.

Additional serialization file is always placed to the directory "serialization\" into 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*

Additional serialization file is created and refreshed after successful device program operation. The only requirement for creating additional serialization file is load project with serialization turned on.

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

#### **Enter job identification dialog**

The dialog will be showed when loading protected project files.

It contains two editable fields:

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

**Note:** Dialog Enter job identification is not password dialog. Values of Operator identification and Job ID have informative purpose only; they will be included in Job Report. It does not relate to protected and/or encrypted project passwords.

## **File / Save project**

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

#### **Description of actually selected project in file list box**


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


#### **Description of project being saved**

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



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

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

Checkbox **Set Protected mode of software after loading of this project file** is used to save project in special mode called **Protected mode**. After clicking the button with key , password dialog appears, which is used to specify Protected mode password for project being saved, and another security options (disable other project loading, device operations restriction) to prevent operator's 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 by **label Protected mode** in right top corner of Programmer activity log.

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

Checkbox **Require project file checksum before first programming** when active, software asks user for entering correct project file unique ID, before allowing to start the first device programming after load project. This feature is recommended for additional check, that correct project file is recently loaded. There is also recommended using this checkbox along with active **Protected mode**. When the request of project file unique ID is active, the software indicates this by label **(ID)** next to project file name in bottom status line in control program main window.

**Note:** *Option **Require project file unique ID before first programming** is replacement of former **Require project file checksum before first programming**. **Unique ID** advantage over generic **checksum** is that **unique ID** is calculated not just from main device buffer data, but also from secondary buffers data used by device and available device settings. When the request of project file checksum is active, the software indicates this by label **(Csum)** next to project file name in bottom status line in control program main window. This option is no longer available in **Save project dialog**, but it can be activated after loading of older project file, that has the checksum request set on.*

## File / Reload file

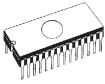
Choose this option to reload a recently used file.

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 of them. Lastly used files are listed before files used far off.

To Reload a file:

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

**Note:** *When reloading a file the file format is used, by which the file was lastly loaded/saved.*



## ***File / Reload project***

Choose this option to reload a recently used file from **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 of them. Lastly used files are listed before files used far off.

To Reload a project:

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

## ***File / Project options***

This option is used for display/edit project options of actually loaded project. Project options mean basic description of project including 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 more detailed project description

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

## ***File / Load encryption table***

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

## ***File / Save encryption table***

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

## ***File / Exit without save***

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

## ***File / Exit and save***

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

## Buffer

Menu **Buffer** is used for buffer manipulation, block operation, filling a part of buffer with string, erasing, checksum 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 buffer (for viewing in DUMP mode only). Use arrow keys for select the object for edit. Edited data are signified by color. The data in buffer outside of area where are located data for the selected chip are shown using gray background.

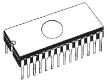
You can use <F4> hot key also.

### View/Edit Buffer

This dialog is used to **view** (view mode) or **edit** (edit mode) data in buffer. Use arrow keys for select data for edit. The data in buffer outside of area where are located data for the selected chip are shown using gray background.

Following commands are available for editing buffer data. Please note, that not all commands are available for all situations. It depends on selected device and buffer(s) used for device.

<b>F1</b>	display help of actual window
<b>F2</b>	fill buffer block specified by start and end addresses by requested hex (or ASCII) string.
<b>Ctrl+F2</b>	erase buffer with specified blank value
<b>Ctrl+Shift+F2</b>	fill buffer with random data
<b>Shift+F2</b>	save buffer data to 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 main menu "File" and also by buttons Load, Save in main application window.
<b>F3</b>	copy specified block of buffer data at new address. Target address needn't be out from source block addresses.
<b>Shift+F3</b>	load data from binary file to buffer. This command is available for secondary buffers only. For more information see notes for save buffer data command (Shift+F2) above.
<b>F4</b>	move block is used to move specified block of data in current buffer on new address. Target address needn't be out from source block addresses. Source address block (or part) will be filled by topical blank character.
<b>F5</b>	swap bytes command swaps a high- and low- order of byte pairs in current buffer block. This block must start on even address and must have an even number of bytes. If these conditions do not fulfill, the program modifies addresses itself (start address is moved on lower even address and/or end address is moved on higher odd address).
<b>F6</b>	print buffer
<b>F7</b>	find string (max. length 16 ASCII characters)
<b>F8</b>	find and replace string (max. 16 ASCII chars.)
<b>F9</b>	change current address



<b>F10</b>	change mode view / edit
<b>F11</b>	switch the mode of buffer data view between 8 bit and 16 bit view. It can be also do by mouse clicking on the button to the right of View/Edit mode buffer indicator. This button indicates 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.
<b>F12</b>	checksum dialog allows count checksum of selected block of buffer change mode view / edit
<b>Arrow keys</b>	move cursor up, down, right and left
<b>Home/End</b>	jump on start / end current line
<b>PgUp/PgDn</b>	jump on previous / next page
<b>Ctrl+PgUp/PgDn</b>	jump on start / end current page
<b>Ctrl+Home/End</b>	jump on start / end current device
<b>Shift+Home/End</b>	jump on start / end current buffer
<b>Backspace</b>	move cursor one position left (back)

**Note:** characters 20H - FFH (mode ASCII) and numbers 0...9, A...F (mode HEX) immediately changes content of edit area.

Not all commands are available for all situations. It depends on selected device and buffer(s) used for device.

**Warning:** Editing of ASCII characters for word devices is disabled.

## Print buffer

This command allows write selected part of buffer to printer or to file. Program uses at it an external text editor in which selected block of buffer is displayed and can be printed or saved to file, too. By default is set simple text editor **notepad.exe**, which is standard part of all versions of Windows.

In Print buffer dialog are following options:

### Block start

Defines start address of selected block in buffer.

### Block end

Defines end address of selected block in buffer.

### External editor

This item defines path and name of external program, which has to be used as text viewer for selected block of buffer. By default is set simple text editor notepad.exe, which is standard part of all versions of Windows. User can define any text editor for example wordpad.exe, which is able to work with large text files. In user defined text editor user can print or save to file selected block of buffer.

The external editor path and name is saved automatically to disk.

## Find dialog box

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

**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 buffer to the end of the buffer) is the default. **Backward** searches toward the beginning. In view mode searches all buffer.

Origin specifies where the search should start.

## Find dialog box

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

**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 buffer to the end of the buffer) is the default. **Backward** searches toward the beginning. In view mode searches all 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 **Options** box you can select prompt on replace: if program finds instance you will be asked before program change 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 buffer to the end of the buffer) is the default. **Backward** searches toward the beginning. In view mode searches all buffer.

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

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

**Yes** replaces found item and finds next

**No** finds 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 buffer data view between 1 bit and 8 bit view. It can be also doing by mouse clicking on the button to the right of View/Edit mode buffer indicator. This button indicates actual data view mode (1 bit or 8 bit), too.

**Arrow keys** move cursor up, down, right and left

**Home/End** jump on start / end current line

**PgUp/PgDn** jump on previous / next page

**Ctrl+PgUp/PgDn** jump on start / end current page

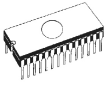
**Ctrl+Home/End** jump on start / end edit area

**Backspace** move cursor one position left (back)

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

## Buffer / Fill block

Selecting this command causes filling selected block of buffer by requested hex (or ASCII) string.



Selecting option "Allow address history logging" activates saving of recently confirmed values. These are saved for each device separately, count is limited to 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.

Selecting option "Maintain last inserted values" causes that for the next time you open this dialog, previously confirmed values will be reloaded as default.

## **Buffer / Copy block**

This command is used to copy specified block of data in current buffer on new address. Target address needn't be out from source block addresses.

## **Buffer / Move block**

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

Selecting option "Allow address history logging" activates saving of recently confirmed values. These are saved for each device separately, count is limited to 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.

Selecting option "Maintain last inserted values" causes that for the next time you open this dialog, previously confirmed values will be reloaded as default.

## **Buffer / Swap block**

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

Following swap modes are available, user can select from:

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

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

Swap bytes operation from Start address 0 to End address N modifies data in buffer by 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 original buffer byte values from addresses 0, 1, 2...

b0n, b1n, b2n... means nibble-swapped original bytes b0, b1, b2... by following 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 option **"Allow address history logging"** activates saving of recently confirmed values. These are saved for each device separately, count is limited to 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.

Selecting option **"Maintain last inserted values"** causes that for the next time you open this dialog, previously confirmed values will be reloaded as default.

## Buffer / Erase block

If this command is selected, the selected range of buffer will be filled with topical blank character.

Selecting option "Allow address history logging" activates saving of recently confirmed values. These are saved for each device separately, count is limited to 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.

Selecting option "Maintain last inserted values" causes that for the next time you open this dialog, previously confirmed values will be reloaded as default.

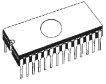
The reserved key <Ctrl+F2> will bring out this menu from any menu and any time.

## Buffer / Fill random data

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

Selecting option "Allow address history logging" activates saving of recently confirmed values. These are saved for each device separately, count is limited to 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.

Selecting option "Maintain last inserted values" causes that for the next time you open this dialog, previously confirmed values will be reloaded as default.

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

## Buffer / Duplicate buffer content

This command performs duplicate buffer content in range of source EPROM to range of destination EPROM. This procedure is suitable if there is used for example 27C512 EPROM to 27C256 EPROM position.

**Note:** *The procedure always uses buffer start address 00000h.*

## Buffer / Checksum

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

PG4UW contains following functions related to checksum:

- Tab **Checksum calculator**, this is on-demand **checksum calculator** that can calculate and display various types of checksums of various data blocks in buffer. (\*1)
- Tab **Main checksum options** contains options for **Automatic checksum calculator** with **Main checksum** value displayed in main window of PG4UW in table **Addresses** and in **Programmer activity log of PG4UW** as "Data checksum of selected buffers:".

Tab **Checksum calculator** contains following controls:

- Group **Buffers included to checksum calculator** contains:
  - Checkboxes to select, which buffers have to be used (included) for checksum calculator. Data of selected buffers are sequentially processed by checksum calculation as one data stream, buffer by buffer, in order as listed in 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 buffer. So there is problem to check the checksum of buffer, when data on some addresses were changed by serialization engine before each device programming. If part of buffer (data block) used for serialization is excluded from checksum calculation, the checksum of buffer data will not be changed by serialization data changes. One or more excluded blocks can be specified.
- Fields displaying values of calculated **checksum types**: see description of types at the bottom.
- Column marked as **STRAIGHT** is result of checksum calculation without additional adjustments.
- Column marked as **NEGATED** is a negation of checksum so, that  $SUM + NEG. = FFFFH$ .
- Column marked as **SUPPLEMENT** is complement of checksum so, that  $SUM + SUPPL. = 0$  (+ carry).
- **Insert checksum options** box - this box contains following options for **Calculate & insert** operation:
  - **Insert checksum** Kind of checksum that is written into the buffer when, the Calculate & insert operation was executed.



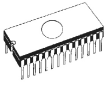
- **Insert at address** Address in buffer where a result of chosen checksum is written, when the Calculate & insert was executed. Address can not be specified inside the range <From address> to <To address>. Address is always defined as Byte address.
- **Size** Size of chosen checksum result, which will be written into the buffer. A size of inserted checksum may be Byte (8-bit), Word (16-bit) or DWORD (32-bit). If size is smaller than selected checksum size, only lower byte(s) of checksum value will be written into the buffer.  
**Note:** *If Word size was selected, a low byte of checksum value will be written on address specified in box Insert address and a high byte will be written on address incremented by one. Similarly it is for DWORD.*
- **Calculate button** - click on the button Calculate starts calculating checksums for selected block in buffer. No writes into the buffer are executed.
- **Calculate & insert button** - click on the button Calculate & insert starts calculating checksums for selected block in the buffer and writes the chosen checksum into the buffer on address specified by Insert address. This function is available for Byte, Word, CRC-CCITT and CRC-XMODEM checksums.
- **Close button** - closes dialog Checksum.

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

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

**Note:**

- *When selecting new device by Select device dialog (F5), **Checksum calculator settings** are set to defaults, as well as **Main Checksum settings**.*
- ***Checksum calculator settings** are not saved to .ini or project file(s), they are temporary only. **Main Checksum settings** are saved to .ini and project files.*
- *When starting PG4UW or loading project, initial settings for **Checksum calculator** for recently selected device from .ini or .epj file are not defaults, but they are taken (copied)*



from **Main Checksum** settings. So **Checksum calculator** options are equal to recent **Main Checksum** options after starting software or loading project.

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

### Word sum Little Endian (x16)

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

### Word sum Big Endian (x16)

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

### CRC-CCITT

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

### CRC-XMODEM

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

### CRC-16

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

### CRC-32

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

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$   
(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 complement of checksum so, that SUM + SUPPL. = 0 (+ carry).

**Device dependent checksum** - applies for some devices, e.g. STMicroelectronics's STM8 family. The checksum modes for **main checksum** can be set in pop-up menu by clicking on label checksum in main program window or by 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.

**DWORD** is 32-bit word.

Abbreviated form of Main Checksum forms listed in main window of PG4UW and in 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 a work with selected programmable devices - device select, read data from device, device blank check, device program, device verify and device erase.

### **Device / Select from default devices**

This window allows selecting the desired type of the device from list of default devices. This one is a cyclic buffer in which are stored recently selected devices including their device options. This list is saved to disk by command **File / Exit and save**.

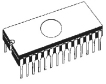
If you wish display additional information about the current device, use an **<Ctrl+F1>** key. This command provides a size of device, organization, programming algorithm and a list of programmers (including auxiliary modules) that supported this device. You can find here package information and other general information about current device too.

Use a **<Del>** key for delete of current device from list of default devices. There isn't possible to empty this list, if you repeat this access. The last device stays in buffer and the **<Del>** key isn't accepted.

### **Device / Select device...**

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

- Enter **search query** into search field to filter device database. Query will be divided to **fragments** which will be considered as indivisible. Use **space** character " " to separate query fragments. To include **space** character into **fragment**, use **double quotes** to surround this "space containing fragment".
- Query **fragments** are compared to **device name**, **manufacturer**, **programming adapter name** and **ordering number**, strings are **not case sensitive**.



- Fragments of query are by default compared using **AND** logic, but there is an option to switch to **OR** logic if needed.
- Matching results have computed **relevancy** according to how close the match is (with respect to AND/OR logic). By default, results are ordered by **relevancy**.
- **Search query** field has history of 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 separated list as optional.

**Note 1:** *The names of the programmable devices in software don't contain all characters, shown at the top of the chip or mentioned in the datasheet section part numbering. The names contain all characters necessary to identification of the device, but don't contain such codes, that have none influence to the programming, for example temperature code, speed code, packing type code, etc.. If such code letter is at the end of the name, is omitted, if such code letter is in the middle of name, then is replaced by 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 **S29GL064NxxTxx01**

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

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

- searching **Am27C512-150**, **Am27C512-200** and **Am27C512-250** gives **Am27C512**
- searching **S29GL064N11TF1010** gives **S29GL064NxxTxx01**
- searching **??27C512** gives **Am27C512** and also **AT27C512**

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

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

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

Selected device is automatically saved to buffer of default devices. This buffer is accessible with **Device / Select from default devices** command.

## Device / Select EPROM /Flash by ID

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

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

**Warning:** *The control program only support this time EPROM's and Flash with 28 and 32 pins. Any of programmers determines pins number automatically. For other programmers you must enter this number manually.*

*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 into the socket a device that is not an EPROM or Flash. It may be damaged when the programmer applies the high voltage.*

*We don't recommend apply this command to:*

- 1) 2764 and 27128 EPROM types, because most of them ID not supports*
- 2) Flash memories with non-standard pinout (e.g. Firmware Hub Flash)*
- 3) Flash memories, which don't accept Vid voltage at A9 pin*
- 4) low voltage EPROM and Flash memories*

**Note:** *The procedure is not available if ISP programmable device is selected i.e. in ISP mode of the programmer. Select any device programmable in ZIF socket of the programmer.*

## **Device / Device options**

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

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

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

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

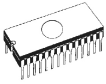
#### **List of commonly used items:**

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 special mode of buffer when programming or reading device. Using split options is particularly useful when using 8-bit data memory devices in 16-bit or 32-bit applications.

Following table describes 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 following: (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 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 chip, if it has proper connection to the ZIF socket (continuity test). The programmer is able to identify the wrong contact, bad inserted chip and also (partially) back inserted chip.

**Device ID check error terminates the operation** (default ENABLE)

Programmer provides ID check before each selected action. It compares read ID codes from device with ID codes defined by device manufacturer. In case of ID error, control program behaves as follows:

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

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

**Note 1:** *Some old chips don't carry electronic ID.*

**Note 2:** *In some special cases, several microcontrollers don't provide ID, if copy protection feature in the chip is set, even if device ID check setting in 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% nominal VCC +/-10% VCCmin - VCCmax)

group **Target system power supply parameters**

This group is available in ISP mode for some types of devices. It contains following settings:

**Enable target system power supply** - enables supplying of target system from programmer. Supply voltage for target system is switched on before action with programmed device and is switched off after action finished. If Keep ISP signals at defined level after operation is enabled, then programmer will switch off supply voltage after pull-up/pull-down resistors are deactivated.

**Voltage** - supply voltage for target system. Supply voltage range is from 2V to 6V.

**Note:** *The voltage value given to target system depends also on current flowing to target system. To reach exact voltage supply for target system, the proper Voltage and Max. current values have to be defined. The Max. current value specified has to be as exact as possible equal to real current consumption of target system.*

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

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

**Target supply settle time** - determines time, after which must be supply voltage in target system stabilized at set value and target system is ready to any action with programmed device.

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

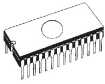
**Power down time** - determines time after switch off target system power supply within target system keeps residual supply voltage (e.g. from charged capacitor). After this time elapsed target system has to be without supply voltage and can be safely disconnected from programmer.

group **Target system parameters**

This group is available in ISP mode for some types of devices. It contains following settings:

**Oscillator frequency** (in Hz) - oscillator's frequency of device (in target system). Control program sets programming speed by its, therefore is necessary set correct value.

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



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

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

**Inactive level of ISP signals** - this parameter determine level of ISP signals after finishing access to target device. Signals of ISP connector can be set to Pull-up (they are tied through 22k resistors to supply voltage) or Pull-down (they are tied through 22k resistors to ground).

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

#### group **Programming parameters**

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

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

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

There are three types of serialization:

- Incremental mode
- From file mode
- Custom generator mode

Dialog **Serialization** contains also settings for associated serialization position files that are used with project files with serialization turned on. For more detailed information about using serialization in project files, look at Serialization and projects.

#### **Basic rules of serialization:**

- serialization is associated with recently selected device only. If a new device is selected, the serialization settings will be reset (serialization will be set to disabled)
- serialization settings for recent device are saved along with other settings of the device to project file or to configuration file when application is closed
- serialization engine calls request for new (next) serial number before each device programming is started
- used serial number is indicated by (\*) after serial number value. Used serial number means, the next device programming will use next serial number



**Note:** Calling of new serial number request before programming can be suppressed in case of previous unsuccessful device programming result by option **Serial number usage if programming action fails**:

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

Serialization can work with control program's main buffer or extended buffers available for some types of devices, for example Microchip PIC16Fxxx devices with Data EEPROM Memory. The selection which buffer use by serialization routine is available in dialog Serialization. If 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 to assign individual serial numbers to each programmed device. A starting number entered by user will be incremented by specified step for each device program operation and loaded in selected format to specified buffer address prior to programming of each device. Options available in Incremental mode serialization allow also set equivalent serialization to Microchip SQTP used for Microchip PICmicro® devices.

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

#### **S / N size**

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

#### **Address**

Address option specifies the buffer address, where serial value has to be written. Note that address range must be inside the device start and device end addresses. Address must be correctly specified so the last (highest or lowest) byte of serial value must be inside device start and device end address range.

#### **Start value**

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

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

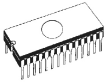
#### **Step**

Step options specify the increment step of serial value incrementation.

#### **S / N mode**

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

- **ASCII** means the serial number is written to buffer as ASCII string. For example number \$0528CD is in ASCII mode written to 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 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 „Save to buffer“ item.

### Style

Style option defines 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 in hexadecimal number, i.e. each nibble must have value from 0 to 9. Values A to F are not allowed as nibbles of BCD numbers.

**Note:** *Select the base in Style options before entering numbers of serial start value and step.*

### Save to buffer

Save to buffer option specifies the serial value byte order to write to buffer. This option is used for 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 serial number to the lowest address in buffer.
- **MSByte first** (used by Motorola processors) will place the Most Significant Byte first to the lowest address in buffer.

### Split serial number

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

Following split options are available:

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

### Example 1:

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

To make above described serialization following settings have to be set in 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 xx xx 00 00 00 16

The 2nd device

Address Data

007FFF0 xx xx 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 xx xx 02 00 00 16

etc.

"xx" mean user data programmed to device

Serial numbers are written to device from address 7FFFCH to address 7FFFFH because serial number size is 4 bytes.

### Example 2:

Following example shows usage of SQTP serialization mode when serial number is split into RETLW instructions for Microchip PIC16F628 devices.

**Note:** *Serial quick turn programming (SQTP) is Microchip specified standard for serial programming of Microchip PIC microcontrollers. Microchip PIC devices allows 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 Incremental mode serialization or From file mode serialization.*

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

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

### Example 2a:

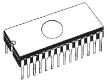
*Use of serialization split with RETLW instructions for Microchip PIC16F628 devices.*

Device PIC16F628 has 14 bit wide instruction word. Instruction RETLW has 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 k are data bits, i.e. serial number byte

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



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

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

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

Address	Data
0000040	ABCD 1234 xxxx xxxx xxxx xxxx xxxx

We want to use RETLW instruction so buffer has to be:

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

We can do this by following steps:

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

The buffer content before starting device program will look for example as following:

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 following:

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 split of serial number by bytes to buffer at every second byte. The correct serial number is set tightly before device programming operation starts.

The buffer content of 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

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

### Example 2.b

*Use of serialization split with NOP instructions for Microchip PIC24FJ256 devices*

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

We can do this by following steps:

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

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

Address	Data
0000800	00 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 following:

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)

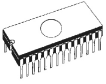
Split settings described above mean split of serial number into fragments with 16 bit (2 bytes) size to buffer with gap of 2 bytes between fragments. The correct serial number is set tightly before device programming operation starts.

The buffer content of 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



Next devices will have same format of serial number incremented by 1 for each device.

**Example 3:**

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

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

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

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 effects of serialization options, there is possible to test the real serial number, which will be written to buffer. The test can be made by following steps:

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

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

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

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

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

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

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

Dialog Serialization offers following options for From-file serialization:

#### **File name**

File name option specifies the serialization file name from which serial addresses and values will be read. The input file for Classis From file serialization must have special format, which is described in From-file serialization file format below.

#### **Start label**

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

#### **Comment used lines (serial records) in main serialization file**

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

**Example:** When we have following line (serial record) in file before reading its value:

```
[label123] 20000 01 02 03 04 05 06 07 08
```

The format of line (after reading its value and commenting it) will be:

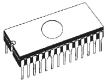
```
;label123] 20000 01 02 03 04 05 06 07 08
```

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

##### **Additional operation with used files**

The group box contains three types of operation. User can select one of the operations to do with used serialization data files in Playlist From-file mode. Following operations are available:

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



- **Delete used file:** Program deletes used serialization data file(s)

**Directory**

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

**File format used for data files**

The option allows user to select file format of serialization data files. Following options are available:

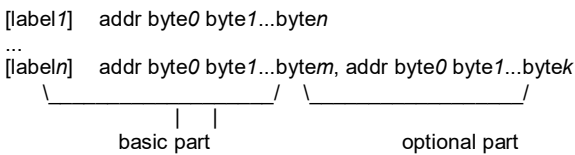
- Binary
- Intel HEX
- ASCII Space
- Autodetect - let's serialization engine to automatically determine file format. Automatic recognition is supported for file formats Binary, Intel Hex, ASCII Space, and Motorola.

**Note:** Size of main From-file serialization file is limited to 1000MB. Recommended maximal number of serial records (items) in one serialization file is 100000 records. More records may cause slower operation when reading 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 buffer addresses and data to write to buffer. Input file has text type format, which structure is:



; Comment

Meaning is:

**basic part**

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

**optional part**

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

**label1, labeln - labels**

Labels are identifiers for each line of input file. They are used for addressing each line of file. The labels should be unique within the file. Addressing lines of file means, the required start label entered by user defines line in input file from which serial values reading starts.



**addr -**

Addr defines buffer address to write data following the address.

**byte0...byten, byte0...bytem, byte0...bytek -**

Bytes arrays `byte0...byten`, `byte0...bytem` and `byte0...bytek` are defining data, which are assigned to write to buffer. Maximum count of bytes in one data field following the address is 64 bytes. Data bytes are written to buffer from address `addr` to `addr+n`.

The process of writing particular bytes to buffer is:

`byte0` to `addr`  
`byte1` to `addr + 1`  
`byte2` to `addr + 2`  
....  
`byten` to `addr + n`

**Optional part** is delimited from the first data part by 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 delimiters basic part and optional part of data

“;” - the semicolon character means the beginning of a comment. All characters from “;” to the end of line are ignored. Comment can be on individual line or in the end of definition line.

**Notes:**

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

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

```
[nav1] A7890 78 89 56 02 AB CD; comment 1
```

```
[nav2] A7890 02 02 04 06 08 0A
```

```
[nav3] A7890 08 09 0A 0B A0 C0; comment2
```

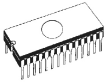
```
[nav4] A7890 68 87 50 02 0B 8D
```

```
[nav5] A7890 A8 88 59 02 AB 7D
```

;next line contains also second definition

```
[nav6] 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 buffer on address \$A7890. All values have size 6 bytes. The line with „nav6“ label has also second value definition, which is written to buffer on address \$FFFF6 and has size 10 bytes, i.e. the last byte of this value will be written to address \$FFFFF.

**Note:** *Address for Serialization is always assigned to actual device organization and buffer organization that control program is using for current device. If the buffer organization is byte org. (x8), the Serialization Address will be byte address. If the buffer organization is wider than byte, e.g. 16 bit words (x16); the Serialization Address will be 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 special header at the first no empty line of file. The header is text line in format

```
FILETYPE=PG4UW SERIALIZATION PLAYLIST FILE
```

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

*labelx* - represents label

Labels are identifiers for each no-empty line of input file. They are used for addressing each line of file. The labels should be unique within the file. Addressing lines of file means, that the required start label entered by user defines line in input file from which serial values reading starts.

*datafilename* - defines name of data file, which contains serialization data. When serialization requires new serial value, the data file will be loaded by standard PG4UW "Load file" procedure to PG4UW buffer. File format can be binary or Hex file (Intel Hex etc.). The auto-recognition system recognizes proper file format and forces load of file in the right file format. Data filename is relative to parent (playlist) serialization 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 the examples included with PG4UW, placed in installation directory in Examples\ subdirectory as following:

```
<PG4UW_inst_dir>\Examples\Serialization\fromfile_playlist_example\
```

The typical path can look like this:

C:\Program Files\Elnec\_sw\Programmer\Examples\Serialization\fromfile\_playlist\_example\

You can test the serialization by following steps:

1. start PG4UW
2. you need to have our programmer connected and correctly found in PG4UW
3. select wished device, the best are devices with erasable memory, (not OTP memory)
4. select dialog from menu Device / Device Options / Serialization
5. Set the From-file mode and in the panel From-file mode options select our example serialization file fromfile\_playlist.ser
6. click the OK button to accept the new serialization settings
7. run "Program" device operation

You can see at the serialization indicating labels in the main window of PG4UW and also in info progress window during device programming and repeating of programming.

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

Custom generator serialization mode provide maximum flexible serialization mode, because the user have serialization system fully in his hands.

When Custom generator mode of serialization is selected, serial numbers are generated by user made program "on-the-fly" before each device is programmed in PG4UW or PG4UWMC. Custom generator mode serialization allows user to generate unique sequence of serial numbers desired. Serial numbers can be incremented as a linear sequence or completely non-linear sequence. The user made serial number generator program details are described later 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 following:*

*<PG4UW\_inst\_dir>\Examples\Serialization\customgenerator\_example\*

*The typical path can look like this:*

*C:\Program*

*Files\Elnec\_sw\Programmer\Examples\Serialization\customgenerator\_example\*

There are following options for **Custom generator serialization** in PG4UW control software:

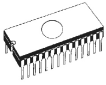
In dialog Serialization select in **Mode** panel option Custom generator mode. The following options will be displayed:

#### **Serialization data file**

Specifies the path and name for the data file that will contain the current serial number. When device is to be programmed, the PG4UW software calls user made serial number generator that updates the data file. The recommended extension of data file is .dat.

Because many of our customers use also BP Microsystems programmers, they ask us for possibility to be used the same serialization software. Therefore the Serialization data file is compatible with .dat files of "Complex serialization" system, available in BP Micro software,

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



### Serialization generator

Specifies the path and name for the executable file which will generate 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 custom generator serialization program. The format of serial number can be like hexadecimal number or like any string that does not contain spaces, tabulators or CRLF characters.

### Last serial number

This option specifies the maximum value of serial number allowed. If the value is no empty string or is "0", it will be passed to serialization generator program. The generator is responsible for testing the value of last serial number and generate serial .dat file with appropriate error information in the serialization .dat file in case of current serial number greater then last serial number. If the value of Last serial number is not specified or is zero "0", the value will not be passed to generator program.

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

This new option has special purpose. If there is requirement to call custom generator with special parameter -RESULT, the check box should be checked. Otherwise it has to be unchecked (the default state is unchecked). If checked, custom generator is called by PG4UW control program after each device operation is completed, no matter the result of device operation is OK or Error. Parameters for generator are created by PG4UW serialization engine. Two parameters are used:

`-RESULT[n]=TRUE | FALSE`

where *n* is optional Programmer Site order number, if multiprogramming is used.

TRUE means that device operation was finished OK. FALSE means, that device operation was finished with error.

*N* parameter contains serial number, for which the -RESULT is called.

For example call of

`generator.exe -RESULT=TRUE -N1234`

means operation was successfully finished for serial number 1234.

Button **Call generator and test consequent data file syntax** allows user to test current setting by calling generator and analyse .dat file. It is useful to test settings before using them.

### Custom generator program

Custom generator program or serialization generator is program that will generate the unique sequence of serial numbers and write the serial data to serialization .dat file. This program is made by user. The path and name of the serialization program must be specified in the Serialization options dialog in Custom generator mode options.

The program will be called from PG4UW every time the new serial data have to be generated. This is usually made before each device programming operation. PG4UW control program passes command line parameters to serialization program and serialization program generates serialization .dat file which is read by PG4UW control program. Following command line parameters are used:

**-N<serial number>** Specifies current serial number.

**N<serial number>** Specifies current serial number, that generator has to use to create proper serial data file. Generator must return the same value at line T01:<serial number> in data file. More information about serialization .dat file format is available in the next section, few lines lower.

**Note:** If PG4UW software detects difference between serial number passed to generator by **-N** command line parameter and serial number specified in T01:... record in data file created by generator, serialization error will be reported.

**E<serial number>** specifies ending (or last) serial number. The parameter is only passed when value of Last serial number specified in dialog Serialization in PG4UW software is no zero. The serialization program should return error record T06 in the serialization .dat file, if the current serial number is greater than ending serial number. For details look at section Serialization .dat file format.

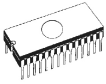
The name of executable file generator (generator.exe) can be replaced by any suitable user specified name for generator application.

### Serialization .dat file format

Serialization .dat file generated by serialization generator must meet following text format. Serialization .dat file consists of records and serial data section.

Record is line, which begin with one of Txx prefixes as described below. Value of "xx" represents the record type code. Records are used to inform PG4UW software about serialization status (current and last serial numbers, serialization data and data format, errors, etc.). Required records are records T01, T02, T03 and T04. Other records are optional.

- T01:<serial number>** Contains current serial number value passed to generator by command line parameter **-N<serial number>**. The format of serial numbers can be like hexadecimal number or like any string that does not contain spaces, tabulators or CRLF characters.
- T02:<serial number>** Contains next serial number value, that PG4UW will use in next serialization cycle. This value is generated by serialization generator and informs PG4UW, which serial number will follow after current serial number. The format of serial numbers can be like hexadecimal number or like any string that does not contain spaces, tabulators or CRLF characters.
- T03:<data format code>** Specifies the serialization data format. Following formats are supported now:  
T03:50 or T03:55 ASCII Space data format  
T03:99 Intel Hex data format  
T03:87 Motorola Exormax
- T04:** indicates the serialization data will follow from next line to the end of file. Serialization data are stored in one of standard ASCII data file formats, for example Intel Hex, ASCII Space and so on. The format used for data must be specified by record T03.



**Example:** Typical serialization data file:

```
T01:000005
T02:001006
T03:99
T04:
:030000000096B89
:0300030000005F5
:02000C005A0197
:01003F004F71
:0000001FF
```

*The file consists of following information:*

*line T01 - current serial number 000005h*

*line T02 - ending (last) serial number 001006h (this value will be used as new current serial number in next serialization cycle)*

*line T03 - serialization data format after line T04 is Intel Hex*

*line T04 - serialization data, which will be loaded to buffer of PG4UW before programming device, data are represented in Intel Hex format*

**Optional records are:**

*T05:<message>* Warning or error message. This record causes the serialization is stopped and warning or error message is displayed in PG4UW software.

*T06:* Current serial number greater than limit

This record causes the serialization is stopped and warning or error message is displayed in PG4UW software. The reason of turning serialization off is the current serial number is greater then allowed maximum ending serial number. This record can be used when -E command line parameter is specified, it means no zero Last serial value in dialog Serialization is specified.

*T11:<message>* Less important warning or message. The serialization will not be interrupted.

Flowchart of device programming with custom-generator serialization

When Custom-generator serialization is used, it means, that before each device programming is started, serialization engine calls serialization generator executable, to generate serial .dat file. PG4UW serialization engine manages proper command line parameters for calling of serialization generator. The data from .dat file are immediately read to internal programmer buffer and used as data for programming device. Also next serial number information (record T02) is remembered in PG4UW.

**Typical flowchart of device programming is following:**

1. Start of programming batch
2. Device insertion test
3. Serialization sequence, consists from four steps:
  - call of serialization generator with proper command line parameters to generate serialization .dat file
  - waiting for serialization .dat file to be available
  - reading of serialization .dat file data to programmer buffer (the data will be used for programming device)
  - delete serialization .dat file after reading of data from it

4. Device programming
5. Device verification
6. Operation result check.

This is fully managed by PG4UW control program. Serialization generator does not have to do any operation according to operation result. Control program will call serialization generator with required command line parameters.

**OK** - PG4UW makes request for next serial number. Next serial number was read from .dat file in step 3. Call of serialization generator will have next serial number specified in command line.

**ERROR** - PG4UW does not make request for new serial number. Recent serial number will be used for next device. Next call of serialization generator will have recent serial number specified in command line.

7. Repeat programming with next device?  
Yes        goto step 2.  
No         continue at step 8.
8. End of programming batch

#### **Notes:**

- *In case of error programming result, recent serial number is used, but generator will be called at step 3. anyway, even if the same number is used as for previously programmed device.*
- *If error of serialization .dat file is detected, program PG4UW reports serialization error and stops continuing of programming batch immediately.*
- *In dialog Serialization settings there is performed no validity test of settings for Custom generator mode, when the settings are confirmed (by OK button). Test button Call generator and test consequent data file syntax can be used for testing the settings before final confirmation of the settings by OK button.*

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

Statistics gives the information about actual count of device operations, which were proceeded on selected type device. If one device is corresponding to one device operation, e.g. programming, the number of device operations will be equal to number of programmed devices.

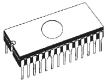
The next function of statistics is **Count down**. Count down allows checking the number of device operations, and then number of devices, on which device operations have to be done. After each successful device operation the value of count down counter is decremented. Count down has user defined start number of devices to do. When count down value reach zero, it means, specified number of devices is complete and user message about complete count down will be displayed.

**Statistics** dialog contains following options:

Check boxes **Program**, **Verify**, **Blank**, **Erase** and **Read** define operations, after which statistics values increment.

Any selected and performed device operation will increment the **Total** counter and one of **Success** or **Failure** counters depending on device operation result (success or failure).

A combination of partial operations is counted as one operation only. For example, a Read operation including Verify after Read is one operation. A Program operation including Erase and/or Verify operations is counted as one operation.



Check box **Count down** sets Count down activity (enable or disable). Edit box following the Count down check box defines initial number of count down counter, from which count down starts.

**Statistics** dialog can be also opened by pressing right mouse button on **Statistics** panel and clicking displayed item **Statistics**.

**Statistics dialog** contains seven statistics values – **Success**, **Operational failure**, **Adapter test failure**, **Insertion test failure**, **ID check failure**, **Other failure (prog. SW, HW)** and **Total**.

Meaning of the values is:

<b>Success</b>	number of operations which where successfully completed
<b>Operational failure</b>	number of operations which where not successfully completed due to error of device
<b>Adapter test failure</b>	number of operations which where not successfully completed due to incorrect programming adapter
<b>Insertion test failure</b>	number of operations which where not successfully completed due to incorrect position of device in programming adapter
<b>ID check failure</b>	number of operations which where not successfully completed due to incorrect ID code read from device
<b>Other failure (prog. SW, HW)</b>	number of operations which where not successfully completed due to hardware error of programmer or control software error
<b>Total</b>	number of all operations

Actual statistics values are displaying in main window of control program in **Statistics** panel.

**Statistics panel** contains four statistics values – **Success**, **Operational failure**, **Other failure**, **Total** and two Count down information values **Count down** and **Remains**.

Meaning of the values is:

<b>Success</b>	number of operations which where successfully completed
<b>Operational failure</b>	number of operations which where not successfully completed due to error of device
<b>Other failure</b>	number of operations which where not successfully completed due to other reason than device error
<b>Total</b>	number of all operations
<b>Count down</b>	informs about Count down activity (Enabled or Disabled)
<b>Remains</b>	informs about remaining number of device operations to do

**Note:** When new device type is selected, all statistics values are set to zero and **Count down** is set to **Disabled**. For multiprogramming PG4UWMC software, 'Reset statistics' button is disabled until any action is running on sites. To stop action on sites press button 'Stop All' **Reset** button in **Statistics** panel reset statistics values.

**Reload Count down** button in **Statistics** panel reloads initial value to **Count down**.

For PG4UW software, the statistics information is saved to Log window when closing PG4UW.



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

### ***Device / Device options / Associated file***

This command is used for setting associated file with current device. This is a file, which can be automatic loaded to buffer after device is selected from default devices select list or by start control program.

You can edit the associated file name in file name box, put a full pathname. The control program checks the present of this file on the disk. Also is possible enabling or disabling automatic load of this file.

You can save both settings i.e. associated file and enabling of automatic load of this file to disk by command **File / Exit and save**.

### ***Device / Device options / Special options***

The special terms used here are exactly the terms used by manufacturer of respective chip. Please read the documentation to the chip you want to program for explanation of all used terms.

If the name of this menu item is starting by "View/Edit ...", then the Read device command will read the content of the chip configuration and it can be viewed and edited by this menu command.

### ***Device / Blank check***

This command will verify if device is "Blank" and can be programmed like **virgin/unused** device.

The control program reports a result of this action by messages in Info window and in Programmer activity log.

If needed to test, the device content is "like new" (virgin, fresh from manufacturer), perform Device/Read of virgin device (or erase buffer/settings) and then run Device/Verify operation of tested device.

If needed to do it before programming (instead of "Blank check before programming" operation), use multiproject feature.

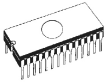
The menu command **Device / Device options / Operation options** of PG4UW allows to customize available operation options.

### ***Device / Read***

This command allows reading all device or its part into the buffer. The read procedure can also read the content of the chip configuration (if it exists and is readable). The special device configuration areas can be viewed or edited in dialogs available by menu **View / Edit buffer** and menu **Device / Device options / Special options** (Alt+S).

The control program reports a finish of Read action by writing a message to INFO window and LOG.

After finished read procedure will be the buffer synchronization process started. The read data will be transferred from programmer's internal SSD disk to PC. Progress of the data transfer is showed in separate window. Transfer can be canceled by pressing key <Esc> or pressing the window's exit button.



The menu command **Device / Device options / Operation options** allows to set another working area as the standard. Setting an option **Verify data after reading** in this menu command means a higher reliability for device reading. In this case are the read data stored in programmer for verification used. Buffer synchronization will be performed after the verification procedure.

## Device / Verify

This command compares content of the whole device including its available special areas with data in buffer stored in internal storage disk of programmer. Therefore are the data from data buffer transferred from PC into the programmer. Progress of the buffer synchronization is in separate window shown. Synchronization process can be canceled by pressing of key **<Esc>**, or pressing window's exit button. The control program reports a result of verify operation to Info window and Log window.

The menu command **Device / Device options / Operation options** of PG4UW allows to customize available operation options.

Settings in dialog **General options** (menu **Options / General options**) of PG4UW in tab **Errors** allows to control, how to write the found errors to user specified report file. Also first 45 found errors are written to Programmer activity log.

### Notes:

- *Verify operation compares content of the whole chip with the data in the software, therefore it might happen - in case of incomplete programmed chip - the verification after programming shows none error, but solo verify operation does not pass.*
- *Verify operation can report errors also in case of protected devices, which have active read protection of data.*
- *Generally, "whole device" means the device range between Device start and Device end addresses set in dialog Device / Device options / Operation options. Not all devices have support to customize Start-End device addresses. Some devices (for example NAND FLASH) have customizable sector or page count/range, and the "whole device" means range of device specified by these options.*

## Device / Program

This command executes device programming. The control program reports a result of this action by messages in INFO window and LOG.

The menu command **Device / Device options / Operation options** of PG4UW allows to customize areas to be programmed, and set other operation options.

Before programming procedure will be data stored in PC transferred into internal storage disk of programmer. Progress of the buffer synchronization is in separate window shown. Synchronization process can be canceled by pressing of key **<Esc>**, or pressing window's exit button.

## Device / Erase

This command executes device erase. The control program reports a result of this action by messages in INFO window and LOG.

The menu command **Device / Device options / Operation options** of PG4UW allows to customize available operation options.

After Erase, if device (chip) doesn't support erase verify command, the blank check operation takes place to verify successfulness of Erase operation.

## Device / Test

This command executes a test of device selected from list of supported devices (for example static RAM) on programmers, which support this test.

The sRAM test is done in 3 basic steps:

- **Test of data drivers functionality.**

Drivers test ... test of D0..D7 signals reaction on CE\, OE\ and WE\:

- in first cycle write data 0x55 to the address 0x0 (CE/=L WE/=L OE/=H) and compare with data read from same address (CE/=L WE/=H OE/=L), data have to be valid.
- then other combination control pins (CE/=L WE/=H OE/=H), (CE/=H WE/=H OE/=L), ..., is set and data have to be not valid - data bus driver have to be inactive.

- **sRAM test, basic part.**

Programmer here writes random data to sRAM device and then verifies the 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:

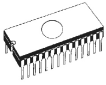
- *It is possible to select a delay between write operation and succeeding verify of programmed data (at condition the device is supplied) in intent to detect 'leak' of the bits.*
- *Programmer hasn't capability to detect errors like too big current on the signal pins or such "analog" errors*
- *All tests are done at low frequency (meant compared with maximal speed of tested device), therefore usage of such test is limited.*

### Conclusions:

- the device programmer can provide only basic answer about health of the sRAM
- if you need test sRAM more deeply use please specialized sRAM tester.

## Device / IC test

This command activates a test section for ICs, mainly Standard Logic IC. The ICs are sorted by type of technology to groups/libraries.



First select an appropriate library, wished device and then a mode for test vectors run (LOOP, SINGLE STEP). Control sequence and test results are displayed to Programmer activity log. In case of need, it is possible to define the test vectors directly by user. Detailed description of syntax and methods of creation testing vectors is described in example\_e.lib file, which is in programs installation folder.

**Notes:**

- *Testing of IC is done using test vectors at some (pretty low) speed. The tests by test vectors can not detect all defects of the chip. In other words, if IC test report is "FAIL", then device is defective. But if test report is "PASS", it means the chip passed our tests, but still might not pass the tests, that check other - mainly dynamic - parameters of the tested IC.*
- *Because the rising/falling edges of programmers are tuned for programming of chips, sometimes the test of some chips fails, although the chips aren't defective (counters for example).*

## 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 programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 Joint Test Action Group (JTAG) interface. Device can be programmed or verified, but Jam STAPL does not generally allow other functions such as reading a device.

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

The Jam Composer is a program, generally written by a programmable logic vendor, that generates a Jam file (.jam) containing the user data and programming algorithm required to program a design into a device.

The Jam Player is a program that reads the Jam file and applies vectors for programming and testing of devices in a JTAG chain.

The devices can be programmed in ZIF socket of the programmer or in target system through ISP connector. It is indicated by [PLCC44] (Jam) or (ISP-Jam) suffix after name of selected device in control program. Multiple devices are possible to program and test via JTAG chain: JTAG chain (ISP-Jam)

More information on the website: <http://www.altera.com>

See please application notes:

"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"

and related application notes for details.

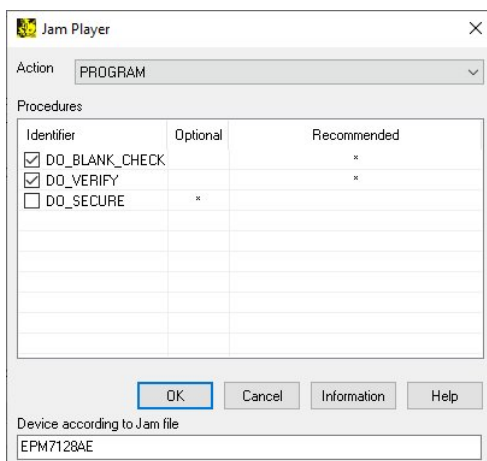
**Software tools:**

**Altera:** MAX+plus II, Quartus II, SVF2Jam utility (converts a serial vector file to a Jam file), LAT2Jam utility (converts an ispLSI3256A JEDEC file to a Jam file);

**Xilinx:** Xilinx ISE Webpack or Foundation software (generates STAPL file or SVF file for use by utility SVF2Jam);

**Actel:** Actel Libero® Integrated Design Environment (IDE) (generates STAPLE file and/or PDB file), Actel FlashPro (converts a PDB file to STAPLE file).

## JAM player dialog



Jam Player (see Action and Procedures controls)

### Action

Select desired action for executing.

Jam file consists of actions. Action consists of calling of procedures which are executed.

### Procedures

Program flow executes statements from each procedure. Procedures may be optional and recommended. Recommended procedures are marked implicitly. You can enable or disable procedures according to your needs. Jam Player executes only marked procedures. Other procedures are ignored. Number of procedures is different, it depends on Jam file.

### OK

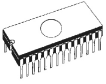
Accept selected action with appropriate procedures which are marked.

### Information

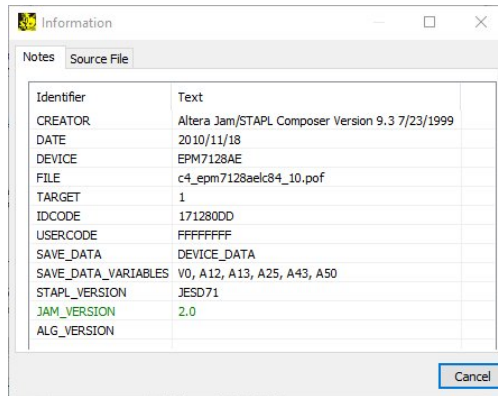
It displays information about Jam file. You can preview Notes and source file in dialog.

### Device according to Jam file

File is made for a specific device. Device name is found in Jam file in part NOTE identifier DEVICE. Device name must be identical with name of the device selected in dialog Select device. When devices are different, software will indicate this situation by warning message during start of the Jam Player.



## JAM file information dialog



**Notes:** statements are used to store information about the Jam file. The information stored in NOTE fields may include any type of documentation or attributes related to the particular Jam program.

**Source file** contains a program in Jam language. Jam program consists of a sequence of statements. Jam statement consists of a label, which is optional, an instruction, and arguments, and terminates with a semicolon (;). Arguments may be literal constants, variables, or expressions resulting in the desired data type (i.e., Boolean or integer). Each statement usually occupies one line of the Jam program, but this is not required. Line breaks are not significant to the Jam language syntax, except for terminating comments. An apostrophe character (') can be used to signify a comment, which is ignored by the interpreter. The language does not specify any limits for line length, statement length, or program size. More information can be found on the website: <http://www.altera.com>

Jam file with extension .jbc is Jam STAPL Byte code format which is not visible.

### **Information about converting JED file to Jam STAPLE file for XILINX devices:**

- install Xilinx Integrated Software Environment (ISE) 6.3i software free download: WebPACK\_63\_fcfull\_i.exe + 6\_3\_02i\_pc.exe (315MB or so)
- run Xilinx ISE 6/Accessories/iMPACT
- in dialog "Operation Mod Selection: What do you want to do first?" choose: "Prepare Configuration Files",
- in dialog "Prepare Configuration Files: I want create a:" choose: "Boundary-Scan File",
- in dialog "Prepare Boundary-Scan File: I want create a:" choose: "STAPL File",
- in dialog "Create a New STAPL File" write name of Jam file with extension .stapl,
- in dialog "Add Device" select JED file with extension .jed,
- in the created jtag chain select device e.g.: XC2C32A (left mouse button) and select sequence operation (e.g.: Erase, Blank, Program, Verify; right mouse button),
- in menu select item "Output/Stapl file/Stop writing to Stapl file"
- run PG4UW, select device e.g.: Xilinx XC2x32A [QFG32](Jam), load Jam file (Files of type: select STAPL File)

- choose "Device operation option Alt+O" press button "Jam configuration". Warning "Select device from menu "Select Devices" and Jam file is probably different! Continue?" choose Yes. (Xilinx sw. does not include line: NOTE "DEVICE" "XC2x32A"; in Jam file). In dialog "Jam player" select action and procedures, finish dialogs, press button "Play Jam" from toolbar and read Log window

### **Information about ACTEL device programming using STAPLE file**

Actel's flash FPGA programming in PG4UW program is performed using Actel Jam player. This programming solution results in special content toolbar button – Play STAPL, which replace all common operations icon (program, erase, verify...) associated with non Jam programming device.

Operation (program, erase, verify...) with Actel device consists of several following steps.

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

Load the appropriate STAPLE file (generated for example by Actel design software LIBERO IDE) clicking on main toolbar "Load" icon. The STAPLE file contains the user data and programming algorithm required to program a design into a device.

#### **Selecting an action**

After successful loading the STAPLE file, select an intending operation action in Device operation options (Alt+O short key)/STAPL configuration. For device programming select PROGRAM from action list. List of all the actions for the programming file with describe can be found in ACTEL FlashPro User's Guide on <<http://www.actel.com>>.

#### **Running an action**

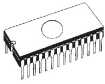
Click Play STAPL button to run selected action. Successful operation (e.g. programming) is terminated with printout "Exit Code = 0... Success" to log window.

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

Actel PDB file is a proprietary file format that can be supported by Actel programmers only, e.g. FlashPro programmer. PG4UW control program can program Actel devices only with Jam STAPL file. Therefore a file conversion between PDB and STAPL is necessary.

Converting PDB file to Jam STAPL file for ACTEL devices:

- install software tool FlashPro (component of Actel Libero tool suite or can be downloads from <<http://www.actel.com>> as a standalone version)
- run FlashPro
- click the New Project button or from the File menu choose New Project and type in the name of your project in the Project Name field. Select desired programming mode – single device or chain and click OK
- from the Configuration menu, choose Load Programming File and select corresponding \*.pdb file to convert
- from the File menu, choose Export/Export Single Device STAPL File... Type in file name and click Save button for export STAPL file to the directory you specified
- Conversion of PDB file to STAPL has finish and created \*.stp file can be used for programming Actel device.



## Frequently asked questions about Actel

**Q:** How can be identify/verify already programmed Actel device?

**A:** There are several possible options to get this done. Each option(action) is varying from each other in method of comparing already programmed Actel device with loaded STAPL file. There are the following appropriate mentioned actions in a STP file:

- **DEVICE\_INFO:** read and among other things display to log window also the checksum of the programming environment programmed into the device. This value can be manually compared by user with the value in the header of the STAPL file (can be viewed in Information window). **Caution:** Value of the programmed device checksum isn't counting from existing (maybe corrupted) device data content however this value is stored during programming to special memory localization and is only reading!
- **VERIFY\_DEVICE\_INFO:** similar options as previous with difference in automatic comparison of programmed device checksum and STAPLE file checksum. The result of comparison can be either success or error window message.
- **VERIFY:** the safest but the slowest (~tens of seconds depends on device capacity against ~1 second in options 1 and 2) option for data compare programmed device content with content of STAPLE file. Comparison selected family features (FPGA Array, targeted FlashROM pages, security setting...) is executing bit by bit and verification process can be early terminated if data mismatch occurs with writing error message to log window.

**Q:** Is it possible to program Actel device with two different STAPLE file in one program action in PG4UW?

**A:** Yes, it is possible. PG4UW control program has built-in multi-project solution for mentioned situation. As an example can be programming data content (first STAPL file) together with security encryption key (second STAPL file).

## The IspVM Virtual Machine

**The IspVM Virtual Machine** is a Virtual Machine that has been optimized specifically for programming devices which are compatible with the IEEE 1149.1 Standard for Boundary Scan Test. The IspVM EMBEDDED tool combines the power of Lattice's IspVM Virtual Machine™ with the industry-standard Serial Vector Format (SVF) language for Boundary Scan programming and test.

The IspVM System software generates VME files supporting both ispJTAG and non-Lattice JTAG files which are compliant to the IEEE 1149.1 standard and support SVF or IEEE 1532 formats. The VME file is a hex coded file that takes the chain information from the IspVM System window. The devices can be programmed in ZIF socket of the programmer or in target system through ISP connector. It is indicated by [PLCC44] (VME) or (ISP-VME) suffix after name of selected device in control program. Multiple devices are possible to program and test via JTAG chain: JTAG chain (ISP-VME).

More information 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 is designed for programming Microsemi's PolarFire, RTG4, IGLOO2, SmartFusion2, ProASIC3 (including ProASIC3 nano), IGLOO (including IGLOO nano), SmartFusion and Fusion devices. The DirectC requires a programming data binary file that can be generated by Libero SoC. More information on the website: <https://www.microsemi.com>

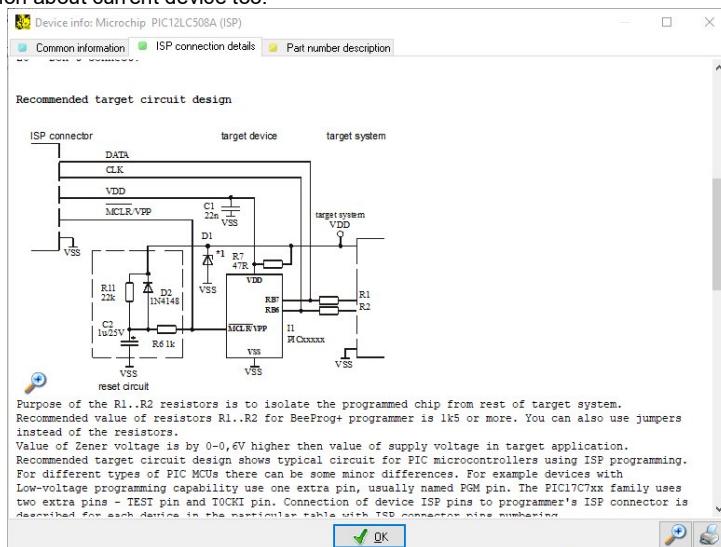


The DirectC Player is a program that reads the DAT file and applies vectors for programming and testing of devices in a JTAG chain. The devices can be programmed in ZIF socket of the programmer or in target system through ISP connector.

Appropriate programming operation can be selected in **Device operation options** dialog by pressing button **DirectC configuration...** and selecting allowed Action.

## Device / Device info

The command provides additional information about the current device - size of device, organization, programming algorithm and a list of programmers (including auxiliary modules), that supported this device. You can find here package information and other general information about current device too.



Recommended target circuit design

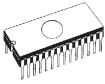
Purpose of the R1..R2 resistors is to isolate the programmed chip from rest of target system. Recommended value of resistors R1..R2 for BeeProg+ programmer is 1k5 or more. You can also use jumpers instead of the resistors.

Value of Zener voltage is by 0-0,6V higher then value of supply voltage in target application. Recommended target circuit design shows typical circuit for PIC microcontrollers using ISP programming. For different types of PIC MCUs there can be some minor differences. For example devices with Low-voltage programming capability use one extra pin, usually named PGM pin. The PIC17C7xx family uses two extra pins - TEST pin and T0CKI pin. Connection of device ISP pins to programmer's ISP connector is described for each device in the particular cable with ISP connector pins numbering.

The reserved key <Ctrl+F1> will bring out this menu from any menu and any time immediately.

## Programmer

Menu Programmer includes commands used for work with programmers.



## Programmer / Find programmer

Select programmer and communication parameters in PG4UW software.

**Programmer** in this section is shown list of all programmers available in current version of PG4UW software. Select the programmer which you want to connect. If option **Search all** is selected, then PG4UW will try to find any of supported programmers and first found programmer will be connected.

**Port** in this section is shown list of all communication ports available for selected programmer. Select communication port where the programmer is physically connected. If you aren't sure with type of communication port, where the programmer is connected, select **All ports**.

**Connect** click this button to start scanning selected communication port(s) for selected programmer. Upon connecting selected programmer which is different than programmer connected previously, the list of default devices will be cleared, except last device if it's supported on new programmer.

**DEMO** click this button to set PG4UW to demo mode with selected programmer. In this case the PG4UW software doesn't scan the communication ports for selected programmer. Upon setting software to demo mode with selected programmer which is different than previous programmer, the list of default devices will be cleared, except the last device if it's supported on new programmer.

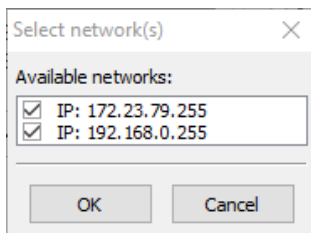
**Cancel** click this button to close Find programmer dialog without changes.

### Connecting the programmer via USB:

- Select the programmer with USB connection capability, e.g. BeeProg2. This will enable USB port settings.
- Select USB port and click Find button.

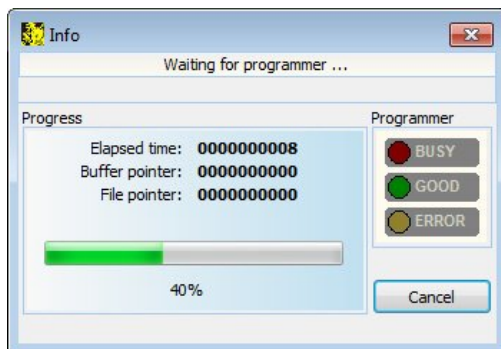
### Connecting the programmer via LAN port:

- Select the programmer with LAN connection capability, e.g. BeeProg3. This will enable LAN ports settings.
- It is possible to input the IP address of desired programmer , or scanning network for available programmers. Automatic scan process will obtain network's properties at first. If multiple networks are present, it is possible to select which network should be scanned for programmers. After the scanning process will be the table of available programmers shown. Only inactive programmers are present in the table. For connection with desired programmer, select programmer in table and press button **OK**.



- Click **Find** button.

Time of connection establishment with selected programmer may vary. During the connection the programmer's firmware is verified and it can happen that the programmer will be restarted . In this case will be the progress of the programmer's reboot in separate window shown.

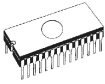


During the reboot sequence will all LEDs of the programmer blinking. If the reboot sequence ends, only the Power LED poorly lit. Part of the connection process is the testing of internal data storage, SSD disk, of the programmer. Time of the test process may vary depending on the disk capacity, amount of file system errors ... Progress of internal buffer testing is shown in separate window.

PG4UW

**Checking internal buffer memory, wait please ... --  
(this may take a few minutes)**

This setting is saved to disk by command **Options / Save options**.



## ***Programmer / Refind programmer***

This menu command is used to re-find (reestablish communication with) currently selected programmer.

To select other type of programmer, programmer communication parameters and to establish communication with newly selected programmer use menu **Programmer / Find programmer**.

## ***Programmer / Handler***

In dialog **Handler** a Handler type and Handler communication parameters can be set. Handler is an external device for special control of device operations in control program. When **None** Handler is selected, this means default state of control program, i.e. device operations are controlled directly by user otherwise control program is in special mode, when device operations are controlled automatically with co-operation with Handler.

Dialog **Handler** contains following items:

- Selected Handler**    select wished Handler type.
- Search at port**     select a COM port, which will be scanned for a requested Handler.

Pressing key **<Enter>** or button **OK** initiates scanning for Handler by set parameters. If selected Handler type is **None**, no Handler scanning will be processed. Current Handler settings are saved to configuration file by command **Options / Save options** or when control program is closed.

Handler is not available for sale.

## ***Programmer / Module options***

This option is used for multiple socket programmers for defining **MASTER** socket and activity of each socket. **MASTER socket** group box allows user to set socket which is preferentially used for device reading operation. **Enable/Disable socket** checkbox array allows user to set enabling and disabling of each socket individually. Disabled sockets are ignored for any device operation.

## ***Programmer / Automatic YES!***

This command is used for setting **Automatic YES!** mode. In this mode you just take off the programmed device, then put new device into ZIF socket and a last operation will be repeated automatically. Program automatically detects an insertion of a new device and runs last executed operation without pressing any key or button. An insertion of device into ZIF is displayed on the screen. Repeated operation executing will be cancelled by pressing key **<ESC>** during waiting for insert/remove a device to/from ZIF.

After an operation with a device is executed, one of the OK or ERROR (status) LEDs on the programmer will light in dependence on the result of an operation and the BUSY LED will blink.

If the program detects removal of a device, then status LED will be switched off, but the BUSY LED will still be blinking to indicate readiness of the program to repeat last operation with new device.

After the program indicates one or more pins of (new) device in the ZIF socket, the BUSY LED will go to light continually. From this the program will wait a requested time for insert the rest pins of new device. If a requested time (Device insertion complete time) overflows and a device is not correctly inserted, the program will light the ERROR LED to indicate this state. After new device was inserted correctly, the program will switch off all status LEDs, except BUSY, and will start an operation with new device.

This mode may be enabled or disabled by item **Automatic YES!** mode. If a new programmer is selected **Options / Find programmer**, this mode will be disabled.

The **Response time** is interval between insertion of the chip into the ZIF socket and the start of selected device operation. If longer positioning of the chip in the ZIF socket is necessary select **elongated** response time.

**Programming adapter used** shows name of adapter used with currently selected device.

**Pins of ZIF socket excluded from sensing** contains list of pins that will be ignored from testing by Automatic YES!. The reason to ignore the pins is mostly - capacitors connected to these pins.

Button **Automatic YES! parameters setting wizard** will run wizard that can detect permanently connected pins (pins with capacitors) and set these pins to list of pins excluded from sensing. After selecting of device, list of excluded pins contains default excluded pins for selected device adapter. If other bypass capacitors to universal programmer and/or device adapter are added by customer, there is necessary to run **Automatic YES! parameters setting wizard** to override default parameters and detect other pins with capacitors.

Button **Reset Automatic YES! parameters to default state**, click this button to discard all parameters set by **Automatic YES! parameters setting wizard**. Default parameters preset in PG4UW software will be used.

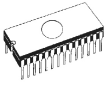
In **Device removal hold off time** is time period between you removed device from the ZIF socket and the time when software starts to check the socket for new device inserted. This interval is in seconds and must be from 1 to 120 (default value is 2 seconds).

In **Device insertion complete time** is possible to set a time within all pins of the device have to be properly inserted after a first pin(s) detected so that the program will not detect incorrectly inserted device. This interval is in seconds and must be from 1 to 120 (default value is 5 seconds).

The **Suspend on error** defines if the Automatic YES! function will be temporary disabled on error to see result of operation or will go on without suspension.

The options are set to defaults after new device is selected by **Device / Select device**

This setting is saved by command **Options / Save options** and is saved also into the project file by **File / Save project** command. The Automatic YES! settings saved in project file can be then used in multiprogramming control program PG4UWMC, when option "**Use settings according to the last loaded project file**" is selected in **General options / Automatic YES!** tab.



**Note:** When using device socket adapters with some passive or active parts, for example capacitors for bypassing supply voltage, the Automatic YES! function may need to know these pins, it could be done by running Automatic YES! parameters setting wizard. This is necessary to make Automatic YES! function working properly. Otherwise Automatic YES! function will "think" the pins are still connected and it will not allow user to insert new device and start new programming.

## Programmer / Selftest

Executes selftest of connected programmer.

First a selftest with Diagnostic pod is executed, Diagnostic pod is included in programmer's delivery package. If no error occurs, then a selftest without Diagnostic pod will be executed.

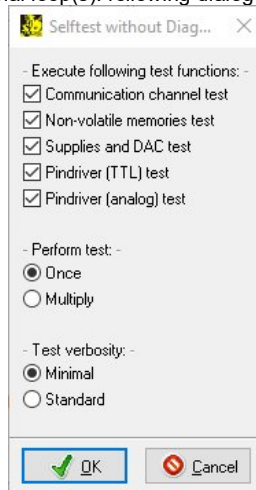
Results of particular test functions as well as overall result are written to Programmer activity log.

If complete selftest is successfully done, then dialog window, shown below, will be displayed. In this dialog window is possible to run selected selftest additional loop(s) or return to application's main screen.



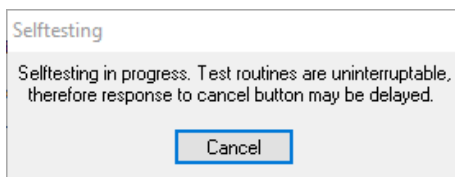
- **Selftest without Diagnostic pod:** selects selftest without Diagnostic pod additional loop(s).
- **Selftest with Diagnostic pod:** selects selftest with Diagnostic pod additional loop(s).
- **Execute:** clicking this button will start executing selected selftest.
- **Cancel:** clicking this button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

Prior to executing selftest additional loop(s), following dialog is displayed.



- In section **Execute following test function** is possible to select parts of the programmer that shall be tested in additional loop(s). Content in this section depends on tested programmer.
- In section **Perform test** is possible to select if selected test functions will run **Once** or **Multiply** times.
- In section **Test verbosity** is possible to select verbosity of writes to Programmer activity log during executing selftest functions.
- **OK** - clicking this button will start executing selected selftest functions.
- **Cancel** - clicking this button will finish selftest and return to main screen.

Selftest of the programmer can be cancelled anytime by clicking Cancel button in the window displayed during running selftest, see picture below.



## ***Programmer / Calibration test***

Command executes test of programmer's calibration values using **AP3 diagnostic POD** (Board A only), which is included in standard delivery of programmer.

There are tested voltage levels of TTL drivers, VCCP, VPP1 and VPP2 voltages on each pin of ZIF socket. Result of the Calibration Test can be saved into file and/or printed (for next use).

## ***Options***

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

### ***Options / General options***

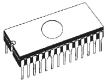
General options dialog allows user to control and set variety of PG4UW program options. The options can be saved to PG4UW configuration file when closing PG4UW application, or anytime by command **Options / Save options**.

#### ***File options***

File options page allows you to set options for erase buffer before loading, auto-reload of current file and recognition method of file formats for loaded files.

**Erase buffer before loading** option sets **erasing** main buffer (with desired value) automatically before loading of data file. There are two options to specify erase value:

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



Main buffer erase is performed immediately before reading file content to buffer and it is functional for binary and all HEX file formats. Main buffer means the first buffer available for viewing/editing in dialog View/Edit buffer (menu Buffer / View/Edit...).

In group **When current file is modified by another process** can be set mode of reloading of actually loaded (current) file. There are three choices:

- Prompt before reloading file
- Reload automatically

- Ignore change scanning of current file

There are three situations when file modification is tested:

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

**Load file format** allows setting mode of file format recognition for loading files. When automatic file format is selected, program analyses format of loading file and test file for each of supported formats that are available in program. If file format matches one of supported formats, the file is read to buffer in detected format.

Manual file format allows user to select explicitly wished file format from list of supported file formats. File may be loaded no completely or incorrectly, if file format does not match to user selected format.

Check box **Show "Load recent project" dialog on program start** sets the dialog to appear on application PG4UW start. Dialog **Load recent project** contains list of recent projects (project history). User can quickly select and load any of the project from list, or close the dialog without loading of project file.

## File extensions

File extensions page allows you to set file masks.

**File format masks** is used for setting file-name masks to use as a filter for file listing in **File / Save** and **File / Load** file window for all file formats. Mask must contain one of wildcards (\*, ?) at least and must be applied correctly by syntax.

**Note:** *More masks can be specified for each file format. Semicolon is used as delimiter for extensions.*

**Example:**

*Motorola:                                   \*.MOT;\*.S19*

*Defines two file masks   \*.MOT and \*.S19 - for Motorola file format.*

**Project file default extension** is used for setting project files-extension used as default extension in **File / Load** project and **File / Save** project dialogs.

## Buffer

Checkbox **Erase buffer before selecting of new device** allows setting automatically erase main buffer when user selecting new device. This can be useful for some kind of special devices, which require exact type of data at certain addresses, and the data are not part of data file loaded to main buffer for this device.

Main buffer can be erased (filled) with default "blank" value for selected device or with custom-defined value. This can be controlled by selection group box **Erase value** and **Custom erase value** edit field. Main buffer means the first buffer available for viewing/editing in dialog View/Edit buffer (menu **Buffer / View/Edit...**).



**Note:** The setting is saved to PG4UW configuration .ini file. It is not saved to project file.

## Language

This page allows you to select another language for user interface such as menu, buttons, dialogs, information and messages. It also allows selecting wished help file in another language. For another language support of user interface the language definition file is required.

## Sound

Panel **Sound settings** page allows user to select the sound mode of program. Program generates sounds after some activities, e.g. activities on device (programming, verifying, reading, etc.). Program generates sound also when warning or error message is displayed. User can now select sound from Windows system sound (required installed sound card), PC speaker or none sound.

Panel **Allow sound for following actions** contains following options:

- Check box **Successful operation**  
When checked, sound will be generated after device operation successfully completed.  
When unchecked, no sound will be generated after successful device operation.
- Check box **In case of error**  
When checked, sound will be generated after device operation is finished with error.  
When unchecked, no sound will be generated after device operation finished with error.

In the panel **Programmer internal speaker sound settings** is possible to set sound options for some programmers with built-in internal speaker. Sound beeps are then generated from internal programmer speaker after each device operation for indicating device operation result – good or bad result.

## Colors and LEDs

**Colors of the work result LEDs of programmer:**

- **Standard color scheme** (ERROR=red, BUSY=yellow)
- **Former color scheme** (ERROR=yellow, BUSY=red)

**Note:** These settings are available only for newer types of programmers. If you can't see mentioned settings in menu, or menu is not enabled for editing, your programmer doesn't support LED color scheme customization.

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

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

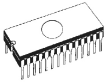
**Note:** These settings are available only for older types of programmers.

**LED brightness** allows customization of LED light intensity. **Note:** This setting is available only for new types of programmers.

**Label style** allows highlighting of important labels (Programmer type, Device type, Statistics, Checksum, etc.) in main window of application

## Errors

This option allows to set a device verify errors saving to file. When verify errors occur, first 45 differences are written to Log window. If user wants to save the verify errors (data differences) to file, he can set options in section **Save device verify errors to file** to one of two methods: cumulate errors from all verify actions to the same file or save errors to file just from last verify



action. Verify errors will be saved to file with name specified by **Error file name** edit box. Following error report file options are available:

- option **No** (default) verify errors saving to file is disabled. Errors are displayed just on screen
- option **New** save verify errors to file just from last verify action. Before first write of new verify action is file deleted and created as new one
- option **Append** verify errors from all verify actions are cumulated into the same file. If file does not exist, the new file will be created

Box **Error report file size limit** contains settings that allow setting max. number of verify errors saved to file. It contains following options:

- Check box **Stop verification after max. number of errors reached**  
If checked, verify action will finish after **Max. number of errors** will be written in file.  
If not checked, all verify errors are saved to the file.
- Edit box **Max. number of errors** specifies number of verify errors, that can be written to error file in one verify operation.

### Log file

This options associates with using of **Log window**. All reports for Log window can be written into the Log file too. The Log file name is "Report.rep" as default. The control program creates this file with name and directory specified in **Log file name** edit box. Following Log file options are available:

- **Create none** content of Log window is not copied to Log file, i.e. all reports will be displayed to Log window only
- **Rewrite** deletes old Log file and creates new one during each start of control program
- **Append** default, adds Log window reports into existing Log file. If file does not exist, the new file will be created

Checkbox **Add date information to Log file name (to create new log file each new day)** allows user to have different Log file for each day. When this option is set, SW adds the information about current date into Log file name, specified by user in Log file name edit box. The SW automatically adds current date string (year/month/day) into user specified Log file name by the following rules:

If user specified log file name has format:

**<user\_log\_file\_name>.<log\_file\_extension>**

The name with added date will be:

**<user\_log\_file\_name><-yyyy-mmm-dd>.<log\_file\_extension>**

The new part representing of 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 added date will look like this (have a date November, 7th, 2006):  
c:\logs\myfile-2006-nov-07.log*

If do you wish to have log file name without any prefix before date information, you can specify the log file name as:

**.<log\_file\_extension>** - dot is the first in file name

**Example:** User specifies Log file name: `c:\logs\log`

The final log file name with added date will look like this (have a date November, 7th, 2006):  
`c:\logs\2006-nov-07.log`

Advanced options about Log file size limit are available too.

- option **Use Log file text truncating when file size limit is reached** - when checked, the Log file size limit is on. It means, that when Log file size reaches specified value, the part of text included in Log file will be truncated. When the option is unchecked, the size of Log file is unlimited, respectively is limited by free disk space only.
- option **Maximum Log file size** specifies the maximum size of Log file in kB.
- option **Amount of truncated text** specifies the percentage of Log file text, which will be truncated after Maximum Log file size is reached. The higher value means more text will be truncated (removed) from Log file.

Drop down list **Date format style in the log** allows select desired date format for all date information written to Programmer activity log.

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

## Job Report

Job Report represents the summary description of operation recently made on device. Job is associated with project file and it means the operation starting with Load project until loading of new project or closing program PG4UW.

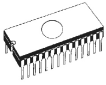
Job Report contains following information:

- project name
- project date
- Protected mode status
- PG4UW software version
- programmer type and serial number
- start time of executing the Job (it means time when Load project operation was performed)
- end time of executing the Job (time of creating the Job Report)
- device name
- device type
- checksum
- device operation options
- serialization information
- statistics information

Job Report is generated in following cases:

- user command Load project is selected
- closing or disconnecting programmer sites is selected
- closing the PG4UW
- device Count down counter reaches 0 (finished status)
- manually by user, when menu "File | Job Report" is used

The Job Report is generated for recently loaded project file, only when statistics value of Total is greater than 0. It means, at least one device operation (program, verify,...) must be performed.



Following options are available for Job Report:

Checkbox **Enable Job Report function** - when checked, the Job Report function is active (enabled). Otherwise the Job Report function is disabled. The possibility of create Job report manually is still available from menu File.

Checkbox **Automatically save Job Report file** - when checked, the Job Report will be saved automatically to directory specified in edit field Job Report directory and with file name created as following:

```
job_report_<ordnum>_<prjname>.jrp
```

where

<ordnum> is decimal order of the file. If there are any report files with the same name, then order for new report file is incremented about order of existing files.

<prjname> is project file name of recently used project, and without the project file name extension.

#### **Example 1:**

*Let's use the project file c:\myproject.eprj and directory for Job Report set to d:\job\_reports\.*  
*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.*  
*Name of this file is d:\job\_reports\job\_report\_000\_myproject.jrp*

*The final Job Report file name of new report will be:*

```
d:\job_reports\job_report_001_myproject.jrp
```

**Note:** *The order inside file name is incremented by 1.*

When **Automatically save Job Report file** setting is set, no Job Report dialogs appears when generating Job Report. Newly generated Job Report is saved to file without any dialogs or messages (if no error occurs while saving to file).

If the checkbox **Automatically save Job Report file** is unchecked, the PG4UW will show Job Report dialog every time needed.

In the Job Report dialog user can select operation to do with Job Report. If user selects no operation (Close button), the Job Report will be written to PG4UW Log Window only.

### **Automatic YES!**

Allows user to set mode of indication for the state when the programmer and the software wait for withdrawing programmed device and a new one in active Automatic YES! mode.

**By LED Busy blinking** - the programmer indicates the state when a device is programmed and the programmer with software wait for inserting a new device mode by blinking with the LED Busy. After an operation with a device is done, one of the status LEDs (OK or Error) lights, in dependence on the result of previous operation and the LED Busy is blinking. If the program detects removal of a device from ZIF socket, then the status LED goes off, but the

LED Busy is still blinking to indicate readiness of the program to repeat last operation with new device. After the program indicates one or more pins of (new) device in the ZIF socket, the LED Busy goes light continually. From this point the program waits a requested time for insertion of the rest pins of new device. If a requested time (Device insertion complete time) overflows and a device is not correctly inserted, the program will light the LED Error to indicate this state. When new device is inserted correctly, the status LED goes off and a new operation with device is started.

**Not indicated (quiet mode)** – the programmer does not indicate the state when a device is programmed and the programmer with software wait for inserting a new device. After an operation with a device only one of the status LEDs Error or OK lights, in dependence on the result of previous operation. This LED goes off immediately after detecting removal of a device from the ZIF socket.

### **Remote control**

Remote control of PG4UW control program allows controlling some functions of PG4UW application by other application. This is very suitable feature for integrating device programmer to mass-production handler system or other useful application.

Remote application that controls PG4UW acts as Server. Program PG4UW acts as Client. Communication between PG4UW and remote control program is made via TCP protocol - this allows the PG4UW to be installed on one computer and remote control application to be installed on another computer and these computers will be connected together via network.

Default TCP communication settings for remote control are:

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

Address setting applies for PG4UW (Client) only. Port setting applies for PG4UW (Client) and also for Server application.

Default settings allow using remote control on one computer (address localhost). PG4UW (Client) and remote control Server have to be installed on the same computer.

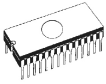
**Note:** *If firewall is installed on system, firewall can display warning message when remote control Server or Client is starting. When firewall is showing warning with question asking to allow or deny network access for remote Server or Client, please select 'Allow' option, otherwise remote control will not work. Of course you can specify in firewall options more strict rights to allow remote Server/Client access on specified address and port only.*

For more information about remote control of PG4UW and demonstration remote control applications, please see the application note **remotemanual.pdf** placed in subdirectory \RemoteCtrl which is in the directory, where PG4UW is installed. Manual for remote control is available also from Windows Start / Programs menu link to Remote manual, created during PG4UW installation.

### **Save options**

Page allows you to select the program options saving when exiting program. Three options are available here:

<b>Don't save</b>	don't save options during quitting program and don't ask for saving options
<b>Auto save</b>	save options during quitting program without asking for saving options



**Prompt for save** program asks user for saving options before quitting program. User can select to save or not to save options

### **Other**

Page **Other** allows user to manage other program settings.

Panel **Application priority** allows user to set the priority of the program. Priority settings can affect performance of programmer (device programming time), especially if there are running more demanding applications in the system. Please note that setting application priority level to Low can significantly slow down the program.

In the panel **Tool buttons**, hint display options on toolbar buttons in main program window can be modified. In the panel **Start-up directory** can be selected mode of selecting directory when program starts. **Default start-up directory** means directory, from which program is called. **Directory in which program was lastly ended** means the last current directory when program was lastly ended. This directory assumes the first directory from directory history list.

## **Options / View**

Use the View menu commands to display or hide different elements of program environment such as toolbars.

Following toolbars are available now:

### **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 Device options before device operation is confirmed.

## **Options / Protected mode**

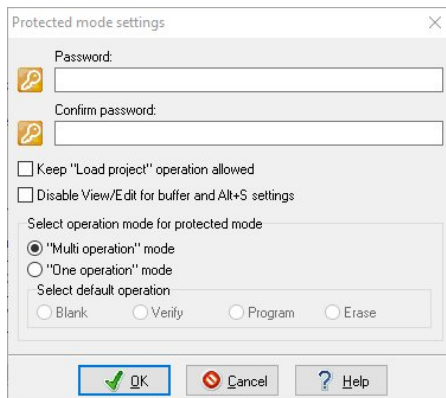
**Protected mode** is special mode of program. When program is in Protected mode, there are disabled certain program operations (reset statistics...) and commands related to buffer content (load/save data to/from buffer, edit buffer...), device settings (edit device options, special options...) or device operations (device read...). **Protected mode** is used to prevent operator from modify program or device settings due to insignificance. Protected mode is suitable for programming of large amount of devices which are of the same type.

Protected mode function is available independently in single programming control software PG4UW and in multiprogramming control software **PG4UWMC**.

### **Protected mode in PG4UW**

There are two ways how to switch program to Protected mode:

1. by using menu command **Options / Protected mode**. This command displays password dialog. User has to enter password twice to confirm the password is correct. After password confirmation program switches to Protected mode. The entered password is then used to switch off Protected mode.
2. by reading project, that was previously saved in Protected mode. For details see **File / Save project**.



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 Protected mode is active. If the option is enabled (checked), the Load project operation button and menu will be allowed in Protected mode.

Checkbox **Disable View/Edit for buffer and Alt+S settings** is set to inactive state by default - it means the View/Edit buffer button and menu will be enabled when Protected mode is active. This allows you and others to view content of buffer, but not edit (due to active Protected mode).

Activate this option, if you wish to disable also viewing of buffer content in Protected mode. In this case, we recommend that you activate also option Encrypt project file (with password). For details see File / Save project.

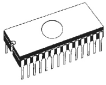
### Select operation mode for protected mode

Options **"Multi operation" mode** - represents basic form of protected mode, where all available device operations (blank, verify, program, erase) are enabled, except read. This provides certainty, that operator cannot modify buffer data by accidental or intentional read operation. It's useful when you want to have all supported device operations enabled in one project (multi-project).

Options **"One operation" mode** - represents **enhanced form of protected mode**, where **only one operation** from all available **is enabled**. Provides better certainty, because prevents operator from executing wrong type of device operation.

By **building** more projects saved in "One operation" mode using Multi-project Wizard you can put together also non-standard **flow of device operations** of control SW (e.g. Program + Verify + Verify + Verify). Please, see examples of use and differences between operation modes.

To switch program from Protected mode to **Normal mode**, use the menu command **Options / Normal mode**. The **"Password required"** dialog appears. User has to enter the same password as the password entered during switch to Protected mode.



Other way to **cancel Protected mode** of program is closing of program. Next time the program starts in Normal (standard) mode (the only exception is case of project loaded by command line parameter with name of project which was saved in Protected mode).

When Protected mode is active, the software indicates this by **label Protected mode** in right top corner of Programmer activity log.

If you search information about another option **Require project file unique ID before first programming**, indicated by **label (ID)** next to project file name in bottom status line of control program, please take a look at chapter File / Save project.

### **Protected mode in PG4UWMC**

Administrator Mode and Operator Mode in PG4UWMC (former Protected mode)

Program PG4UWMC is set by default to Administrator Mode. It means no operation blocking for user is applied. But in production, there is suitable to block some menu commands, to ensure, user does not modify important program settings or configuration. **Operator Mode** is used for this purpose.

PG4UWMC has **Operator Mode** very similar to **Protected mode** of program PG4UW. The difference is, that Operator mode can be activated by menu command but cannot be activated by Project file. Another difference is that Operator mode settings of PG4UWMC are saved to configuration .ini file of PG4UWMC while program PG4UWMC is closed. During next start of application PG4UWMC the recent Operator mode settings obtained from .ini file are used.

There is menu command - **Options / Switch to Operator Mode...** that allows using Operator mode in application PG4UWMC. After selecting the menu **Options / Switch to Operator Mode...**, password dialog appears. User has to enter password twice to confirm the password is correct. After successful password confirmation program switches 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.

To switch program from Operator mode back to Administrator mode, use the menu command **Options / Switch to Administrator Mode....** The „**Password required**“ dialog appears. User has to enter the same password as the password entered during switch to Operator mode.

When **Administrator Mode** is active, the label Administrator Mode is visible in right top corner of Programmers activity log.

When **Operator Mode** is active, the label Operator Mode is visible in right top corner of Programmers activity log.

**Note:** *Sometimes when Administrator Mode is switched from Operator Mode, some commands (for example command "Load project") may remain disabled. This can be resolved by clicking on button Stop ALL.*

During **load of project** operation (menu "File / Load project") these **restrictions will be accepted** if the project is **protected** and has defined **default operation**. Device operations in PG4UWMC will be enabled or disabled accordingly. Also "Load project" operation can be affected this way. This **is applied only** when option "Use Site #1 project for all Sites" is selected. If the option is turned off, each Site will have its own project file, and no menu operations blocking forced by project settings will be performed. All operation protection settings from protected projects will be **ignored**.

## Multi-projects

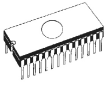
**Multi-project** is special feature which provides possibility to run **any sequence** of operations **with any device**, based on informations saved during creation of **sub-projects** and **multi-project** itself.

In practice, using Multi-projects you are able to:

- comfortably program multi-chip devices
- configure and run any sequence of device operations (e.g. Program + Verify + Verify + Verify) with one device

Basic terms related to Multi-project:

- **Multi-project file** is special file that contains all **Multi-project** information. Multi-project file can include one or more projects. Projects included in Multi-project (file) are also called **sub-projects**.
- **Sub-project** means classic project file which has been included into Multi-project file during Multi-project file build.
- **Project file** - a special type of file that combines buffer data, device operation options, special options and some level of safety features. It completely defines the way how to treat with the device. Once saved, it can be reloaded anytime and the operation can be repeated exactly.
- **Multi-chip device** is device with two or more independent chips (of the same or various types) in single package.



- **Sub-device** - an individual part of multichip device. Sub-device is selectable from PG4UW device list. Once selected, you can work with respective chip in fully manner. You can define, test and save the project file for the partial chip.
- **Master device** - a multichip device unit, consists of sub-devices. Master-device is selectable from PG4UW device list, too. Once selected, you can use Multi-project Wizard to build-up the Multi-project file from individual project files and save/load/execute it. Master device is not defined if the Multi-project is built up from Single-chip devices.
- **Device operation** - each operation executable directly selecting via menu, clicking on toolbar button or callable via remote command (Blank, Read, Verify, Program, Erase). Some of these operations (especially Program and Erase) may contain embedded sub-operations, editable via Menu/Device/Device options.
- **Multi-project Wizard** - an assistant for Multi-project file building. The Wizard allows user to select projects that have to be included in Multi-project and save them to one Multi-project file. Process of saving selected project files to one Multi-project file is called Multi-project file building. The Wizard also allows to start device operation according to projects (sub-devices) included in Multi-project. More information about.
- **Multi-project checksum** includes checksums of all data and settings from all sub-projects saved in multi-project file. Multi-project checksum also acts as unique ID of each multi-project file that means no two multi-projects have the same multi-project checksum. Even if saving the same multi-project more times without any changing of its content, the multi-project checksum will be different.

### **Multi-project Wizard**

Multi-project device operation requires Multi-project file, which contains partial sub-projects associated to sub-devices (chips) of Master device. Multi-project file can be created in Multi-project Wizard. The Wizard has following main functions:

- Select of sub-projects and build final Multi-project file
- Load of existing Multi-project file \*1
- Start device operation of recent Multi-project

**Note \*1:** Existing Multi-project file can be loaded from main menu of PG4UW using menu *File / Load project* or from Multi-project Wizard by *Load multi-prj* command.

#### **Multi-project Wizard contains following controls:**

- Button **Load multi-prj** is used for load of existing Multi-project file.
- Button **Build Multi-project** is used for build of new Multi-project file, which uses projects listed in table Sub-projects.
- **Table 1: Sub-projects** contains list of projects that are included in recent Multi-project.
- Button **Add project** is used for adding of new project file(s) to list of project files in Table1.
- Button **Remove project** is used for removing of selected project file from list of project files in Table 1.
- Buttons **Move up** and **Move down** are used for moving of selected project in Table 1 one position up or down. Projects are processed in specified sequence order, the upper-most (#1) as first.
- Button **Help** show this help.
- Buttons of device operation **Blank, Verify, Program, Erase, or Run** are used for running of selected device operation on all chips (sub-devices) listed in table Sub-projects.
  - In **"Multi operation" mode**, one of all available operations can be run at a time (the same operation on each sub-device).

- In "**One operation**" mode, only one operation can be run (the same operation on each sub-device) or each subproject can run it's own (one) operation, depending on projects the Multi-project consists of.

**Following two basic actions have to be performed when using Multi-project to program Multi-chip devices (similar also for Single-chip devices)::**

- Making (building) of Multi-project (or Multi-project file)
- Using of Multi-project for running of device operation

**Making (building) of Multi-project (or Multi-project file)**

Following steps are recommended when making Multi-project file:

- Create "classic" projects, one project for each sub-device of multichip device. Projects are created in the same way as projects for generic devices:
  - select sub-device according to required chip of multichip device \*1
  - set device parameters, settings, and load required device data to buffer by **Load file** command in PG4UW
  - optionally make test of device operation by running the device operation on device
  - if everything is OK, the project file can be created by **Save project** command
- Select Master multichip device, the Multi-project has to be used for. After selection of multichip device, Multi-project Wizard is automatically opened.
- In Multi-project Wizard add required projects by **Add project button**. Each project represents one sub-device of multichip device.
- After completing of sub-project selection, use button **Build Multi-project** to create final Multi-project file. Program will prompt for name of new Multi-project file. Final Multi-project file will contain all sub-projects listed in Table 1: Sub-projects.

**Note:** *There is possible to create Multi-project from any classic project files. So association with Master device is not mandatory. It is only on user's consideration how to combine correct sub-devices (sub-projects) into one Multi-project. This feature can be especially useful when using ISP programming of devices in JTAG chain with different projects defined.*

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

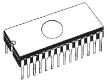
- selecting of Master multichip device from **Select Device** dialog in PG4UW
- loading of created **Multi-project file**
- opening dialog **Multi-project Wizard** directly from PG4UW menu **Options / Multi-project Wizard**

\*1 Convention for Master-device and Sub-device part names in 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 of Multi-project for running of device operation

Typical usage of existing Multi-project file has following order.

For single programming in PG4UW:

- Load created Multi-project by **File / Load project** menu command in PG4UW main window or **Load multi-prj** button in Multi-project Wizard. After successful loading of Multi-project, Multi-project Wizard is opened automatically.
  - In Wizard run wished device operation using one of available device operation buttons (Blank, Verify, Program, Erase), mostly Program device operation is used. Selected device operation is executed as sequence of sub-project loading and consequent sub-device programming for each sub-device defined in Multi-project. And this is main purpose of Multi-project - to automate running sequence of device operations for each chip of multichip device. The side effect of this concept is, that device progress indicators are reset to 0 at beginning of each sub-device operation, so it looks like progress bar is "jumping" to 0 few times while multichip operation is running.
  - After programming of all sub-devices is completed (or error occurs), standard "Repeat" dialog is displayed. Programmed device can be removed from programmer socket and new device can be inserted. Pressing Yes button in dialog Repeat or YES! button on programmer, will start multichip device programming sequence again.
- \* If Automatic YES! function is turned on, no Repeat dialog is displayed after device operation is completed, but Automatic YES! window will appear. The window shows status of programmer socket and notice about removing of programmed device and inserting of new device to programmer socket. After inserting of new device, multichip device operation sequence will start automatically. For more details about Automatic YES! function, please take a look at **Programmer / Automatic YES!**.

For multiprogramming by PG4UWMC or standalone programmer:

- Load Multi-project by Load project menu.
  - Run wished device operation by one of available device operation buttons (Blank, Verify, Program, Erase), mostly Program device operation is used. Selected device operation is executed as sequence of sub-project loading and consequent sub-device programming for each sub-device defined in Multi-project. The side effect of this concept is, that device progress indicators are reset to 0 at beginning of each sub-device operation, so it looks like progress bar is "jumping" to 0 few times while multichip operation is running.
  - After programming of all sub-devices is completed (or error occurs), information with result of device operation is displayed in PG4UWMC. Programmed device can be removed from programmer socket and new device can be inserted. Pressing operation button for the Site or YES! button on programmer Site, will start multichip device programming sequence again.
- \* If Automatic YES! function is turned on, sequence of device operation is started again automatically after removing of programmed and inserting of new device to programmer socket. For more details about Automatic YES! function, please take a look at **Programmer / Automatic YES!**.

### Notes:

- *serialization is not supported in multiprogramming mode (only single programming supports serialization)*
- *count-down function is not supported now*

## Options / Save options

This command saves all settings that are currently supported for saving, even if auto-save is turned off. Following options are saved:

- settings of recently used programmer
- settings of recently selected device - device options and special device options, if available (no device data buffers are saved)
- list of 20 last selected devices with their actual device options settings (no data buffers are saved)
- options under the Options menu (General options and also current configuration of main window toolbars)
- recent files/projects list (names of up to 10 recently used files and names of up to 10 recently used projects)
- main program window and some sub-windows - position and size

## Help

Menu **Help** contains commands that let you view supported devices and programmers and information about program version too.

Pressing the <F1> key accesses the Help. When you are selecting menu item and press <F1>, you access context-sensitive help. If PG4UW is executing an operation with the programmer <F1> generates no response.

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.

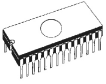
Detailed information on individual menu commands can be found in the integrated on-line Help.

**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 manual on [www.elnec.com](http://www.elnec.com).*

*Since the Help system is continuously updated together with the control program, it may contain information not included in this manual.*

## Help / Supported devices

This command displays list of all devices supported by at least one type of all supported programmers. It is useful especially when user wants to find any device supported by at least one type of programmers.



## **Help / Supported programmers**

This command displays information about programmers, where supported this program.

## **Help / Device list (current programmer)**

This command makes a list of all devices supported by current programmer and saves it to **?????DEV.txt** text file and **?????DEV.htm** HTML file in the directory where control program is run from. Marks **?????** are replaced by abbreviated name of current programmer, the device list is generated for.

## **Help / Device list (all programmers)**

This command makes device lists for all programmers and saves them to **?????DEV.TXT** text files and **?????DEV.HTM** HTML files in the directory where control program is running from. Characters **?????** are replaced by abbreviated name of programmers, the device lists are generated for.

**Note:** *The control program loses all information about current device after this command is executed. Reselect wished device again by any of select methods in menu **DEVICE**.*

## **Help / Device list (cross reference)**

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

- one main HTML file **TOP\_DEV.htm** with supported device manufacturers listed
- partial HTML files with list of supported devices for each device manufacturer

Main HTML file is placed to directory where this control program for programmers is located.

Partial HTML files are placed to subdirectory **DEV\_HTML** placed to the directory where control program for programmers is located.

## **Help / Create problem report**

Command **Create problem report** is used for writing more particular diagnostic information to Log window and consequently for creating **a Problem report information file** from **Log window**. The created file can be opened and content can be read by any text editor.

Problem report file is useful when problem appears during device programmer usage and kind of the error is so complicated, that user can not resolve it oneself and he must contact programmer manufacturer. When customer in this case sends only insufficient information about his problem to manufacturer, it might be not enough for detection of problem reason. Therefore it is recommended to send also Problem report file, which helps manufacturer to localize the reason of error and resolve it sooner.

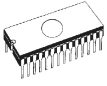


There are two options/buttons for **Create problem report**:

- **Button Yes, last days logs.** This option creates smaller size Problem report .zip file, containing log files from last week. Log files included into .zip are all truncated to maximum file size about 1MB.
- **Button Yes, all days logs available.** This option creates complete Problem report, it means all available PG4UW/PG4UWMC log files are included into Problem report .zip file, and no file truncate is applied. Resulting Problem report .zip file could be much more bigger, than in Last days logs case, but it offers complete overview of operations

## **About**

When you choose the Info command from the menu, a window appears, showing copyright and version information.



---

# ***PG4UWMC***

---



Program **PG4UWMC** is used for fully parallel concurrent device multiprogramming on more programmers or on multiprogramming capable programmers connected to USB/LAN ports of one or more computers.

**PG4UWMC** is focused to the easy monitoring of high-volume production operations. Operator-friendly user interface of PG4UWMC combines many powerful functions with ease of use and provides overview of all important activities and operation results without burden of operator with non-important details.

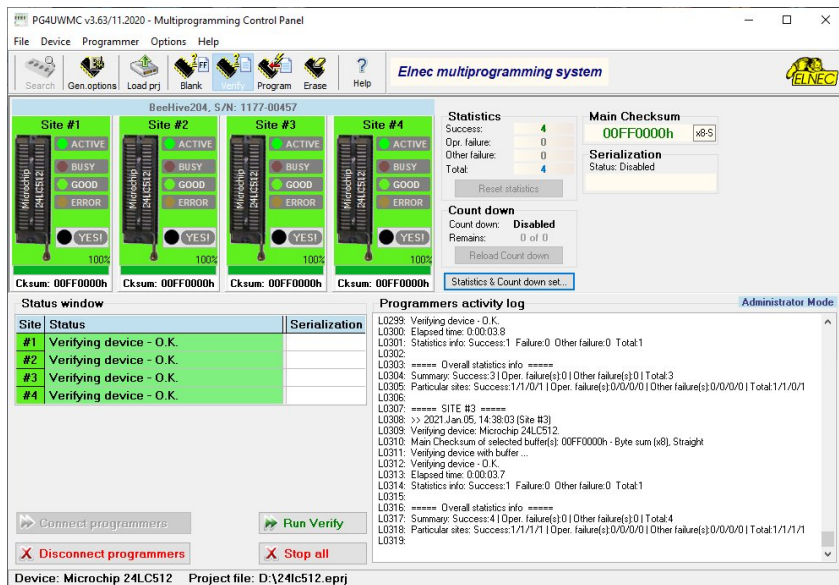
**PG4UWMC** is using a project file to control the multiprogramming system. Project file contains user data, chip programming setup information, chip configuration data, auto programming command sequence, etc. Therefore the operator error is minimized, because the project file is normally created and verified by engineers and then given to the operator. The optional Operator mode can be set for PG4UWMC to avoid unwanted/unintended operations. Each chip may be programmed with different data such as serial number, configuration and calibration information.

Program PG4UWMC consists of following main windows:

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

For more details about multiprogramming and how to use PG4UWMC, please refer to user manual for any of our USB interfaced concurrent multiprogramming system (programmer).

## The basic description of the main parts of PG4UWMC



PG4UWMC v3.63/11.2020 - Multiprogramming Control Panel

File Device Programmer Options Help

Search Gen. options Load prj Blank Verify Program Erase Help Elnec multiprogramming system

BeeHive204, S/N: 1177-00457

Site #1 Site #2 Site #3 Site #4

ACTIVE ACTIVE ACTIVE ACTIVE

BUSY BUSY BUSY BUSY

GOOD GOOD GOOD GOOD

ERROR ERROR ERROR ERROR

100% 100% 100% 100%

Cksun: 00FF0000h Cksun: 00FF0000h Cksun: 00FF0000h Cksun: 00FF0000h

Statistics

Success: 4

Op. failure: 0

Other failure: 0

Total: 4

Reset statistics

Main Checksum

00FF0000h x8-S

Serialization

Status: Disabled

Count down

Count down: Disabled

Remains: 0 of 0

Reload Count down

Statistics & Count down set...

Status window

Site	Status	Serialization
#1	Verifying device - O.K.	
#2	Verifying device - O.K.	
#3	Verifying device - O.K.	
#4	Verifying device - O.K.	

Programmers activity log

L0299: Verifying device - O.K.

L0300: Elapsed time: 0:00:03.8

L0301: Statistics info: Success:1 Failure:0 Other failure:0 Total:1

L0302: ----- Overall statistics info -----

L0304: Summary: Success:3 | Oper. failure(s):0 | Other failure(s):0 | Total:3

L0305: Particular sites: Success:1/1/0/1 | Oper. failure(s):0/0/0/0 | Other failure(s):0/0/0/0 | Total:1/1/0/1

L0306: ----- SITE #3 -----

L0308: >> 2021 Jan.05, 14:38:03 (Site #3)

L0309: Verifying device: Microchip 24LC512

L0310: Main Checksum of selected buffer(s): 00FF0000h - Byte sum (x8), Straight

L0311: Verifying device with buffer ...

L0312: Verifying device - O.K.

L0313: Elapsed time: 0:00:03.7

L0314: Statistics info: Success:1 Failure:0 Other failure:0 Total:1

L0315: ----- Overall statistics info -----

L0316: Summary: Success:4 | Oper. failure(s):0 | Other failure(s):0 | Total:4

L0318: Particular sites: Success:1/1/1/1 | Oper. failure(s):0/0/0/0 | Other failure(s):0/0/0/0 | Total:1/1/1/1

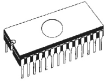
L0319:

Connect programmers Run Verify

Disconnect programmers Stop all

Device: Microchip 24LC512 Project file: D:\24lc512.eprj

PG4UWMC main window



Main window of PG4UWMC consists of following parts:

### Menu and tool buttons

Menu and tool buttons allow access to most of PG4UWMC functions.

### Tool button Settings

Button is used to open PG4UWMC Settings dialog. Settings dialog is described below.

### Panels Site #1, Site #2,...

Panels are used to inform about:

- Programmer Site selected
- Programmer Site activity
- current device operation status and/or result

Each panel also contains button **Run** or button **YES!** used to start device operation.

### Box Statistics

Box Statistics informs about number of programmed devices and number of good and failed devices.

**Note:** *'Reset statistics' button is disabled until any action is running on sites. To stop action on sites press button 'Stop All'.*

### Checksum

Checksum is showing simple checksum of data loaded from current project file.

### Panel Status window

Panel Status window informs about current state of each Site. State can be

**Blank** Site is no active

**Ready** Site is active and ready to work. Programmer is connected. No device operation is running.

**and other information** currently running device operation, result, programmer connection state and so on

**Log window** on the right side of Status window

Log window contains information about connecting/disconnecting programmers, device operation results and other information.

### Button Connect programmers

Button is used to connect all selected Programmer Sites (start PG4UW control program for each Site). This button is usually used as the first step after starting PG4UWMC.

### Button Disconnect programmers

Button is used to disconnect all connected Programmer Sites and close Programmer Sites control programs. The button will apply only if no device operation is currently running on any connected programmer.

### Button Run <operation>

Button is used to start device operations on all connected programmers at the same time.

The value of <operation> can be one of following types: Program, Verify, Blank check, Erase.

### Button Stop ALL

Button is used to stop currently running device operations on all connected Programmer Sites.

### Button Help

Button is used to display this help.

### Button Start remote control of programmer(s)

The button is available for automated programmers only and only if **Use Site #1 project for all Sites** in PG4UWMC Settings dialog/Multiprogramming/Project options is checked. It is used to start remote control of PG4UWMC interface.

After PG4UWMC Remote Control starts, a modules detection on sites takes place (according to projects loaded on sites), and a window called PG4UWMC remote control appears on screen.



Its purpose is to provide operation progress information, emergency stop button and logo of programmer's manufacturer company. Every element of this window can be customized, except logo, which has to be always shown. To display menu with options, use right-click on logo, please.

## File / Load project

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

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

Project information consists of:

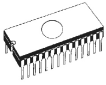
- manufacturer and name of the first device selected in the project
- date and time of project creation
- user written description of project (it can be arbitrary text, usually author of project and some notes)

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

*Serialization is read from project file by following procedure:*

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

*Name of additional serialization file is derived from project file name by adding extension ".sn" to project file's name.*



Additional serialization file is always placed to the directory "serialization\" into 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*

Additional serialization file is created and refreshed after successful device program operation. The only requirement for creating additional serialization file is load project with serialization turned on.

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

**Enter job identification dialog**

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

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

**Note:** *Dialog Enter job identification is not password dialog. Values of Operator identification and Job ID have informative purpose only and they will be included in Job Report. It does not relate to protected and/or encrypted project passwords.*

## File / Make Job report

**Job report** represents the summary description of operation recently made on device. Job is associated with project file and it means the operation starting with **Load project** until loading of new project or closing program PG4UW.

**Job report** contains following information:

- project name
- project date
- Protected mode status
- PG4UW software version
- programmer type and serial number
- start time of executing the Job (it means time when Load project operation was performed)
- end time of executing the Job (time of creating the Job Report)
- device name
- device type
- checksum
- device operation options
- serialization information

- statistics information

**Job report** is generated in following cases:

- user command Load project is selected
- closing or disconnecting programmer sites is selected
- closing the PG4UW
- device Count down counter reaches 0 (finished status)
- manually by user, when menu "File / Job Report" is used

The **Job report** is generated for recently loaded project file, only when statistics value of Total is greater than 0.

It means, at least one device operation (program, verify, etc...) must be performed.

## ***File / Exit without save***

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

## ***File / Exit and save***

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

## ***Device / Select device & Create project***

There is necessary to close PG4UWMC before starting single PG4UW in Engineering mode, which can be used to select device and create (save) project file. See help for PG4UW software: Device / Select device and File / Save project.

## ***Device / Blank check***

This command executes device blank check. The control program reports a result of this action by messages in INFO window and LOG.

The menu command **Device / Device options / Operation options** of PG4UW allows to customize available operation options during creating a project.

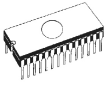
## ***Device / Verify***

This command compares content of the whole device, including special areas and settings (if available), with data stored in the PC (buffer and/or other storage areas) or in the programmer itself (depends on the model of the programmer).The control program reports a result of verify operation to Info window and in the Programmer activity log.

The menu command **Device / Device options / Operation options** of PG4UW allows to customize available operation options during creating a project.

### **Notes:**

- *Verify operation compares content of the whole chip with the data in the software, therefore it might happen - in case of incomplete programmed chip - the verification after programming shows none error, but solo verify operation does not pass.*



- *Verify operation can report errors also in case of protected devices, which have active read protection of data.*
- *Generally, "whole device" means the device range between Device start and Device end addresses set in dialog Device / Device options / Operation options. Not all devices have support to customize Start-End device addresses. Some devices (for example NAND FLASH) have customizable sector or page count/range, and the "whole device" means range of device specified by these options.*

## **Device / Program**

This command executes device programming. The control program reports a result of this action by messages in INFO window and LOG.

The menu command **Device / Device options / Operation options** of PG4UW allows to customize areas to be programmed, and set other operation options during creating a project.

## **Device / Erase**

This command executes device erase. The control program reports a result of this action by messages in INFO window and LOG.

The menu command **Device / Device options / Operation options** of PG4UW allows to customize available operation options during creating a project.

After Erase, if device (chip) doesn't support erase verify command, the blank check operation takes place to verify successfulness of Erase operation.

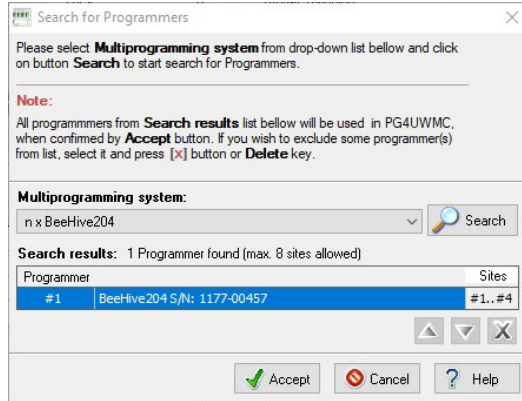
## **Programmer / Search for Programmers**

### **Search on local computer**

This mode of programmers searching is active after installation of PG4UWMC by default. If you prefer to operate with programmers connected to different computers via network, try Network mode.

This command is used for searching and managing multiprogramming systems connected to one PC. It is possible connect a few same type single site programmer or multiprogramming system up to 64 programming sites to one PC.

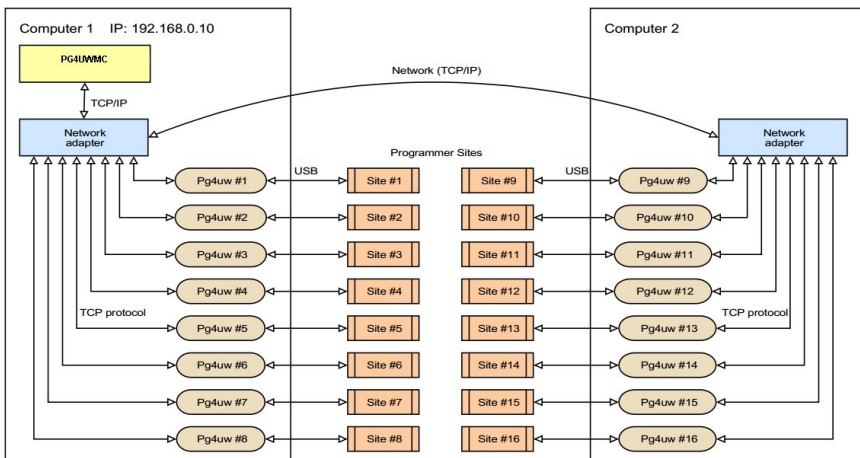
- Select multiprogramming system connected to your PC.
- Press button search.
- Using arrows button organize founded connecting programmer to programming sites.
- Press Accept button



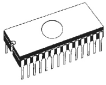
**Note:** Red colored programmers indicate that there are some sites which are expected to be present, but cannot be found. These sites are listed in "Not found" column. Otherwise the column is hidden.

### Search in defined Programmers group on network.

PG4UWMC, when switched to Network mode, allows to search, start, control and monitor instances of PG4UW on network computers. Communication between PG4UWMC and PG4UW is realized through PG4UWMC Network Agent, which is running on each computer. All PG4UWs, PG4UWMC Network Agents on network and controlling PG4UWMC must be of same (thus compatible) version. This feature is available only for automated programmers and is intended to be used mainly with handler machines.

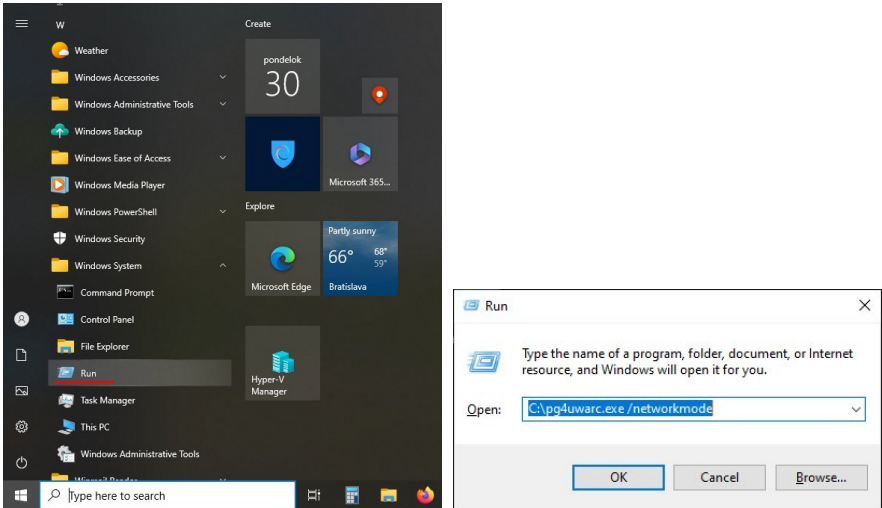


Typical configuration of remotely controlled multiprogramming system running on two computers



## Installation

During installation, the Network Mode feature will not be installed by default. You have to activate it by **executing installation procedure with command-line parameter /networkmode** (e.g. press Win+R and write C:\pg4uwarc.exe /networkmode).

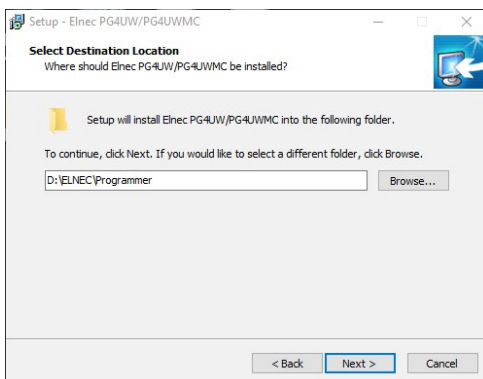
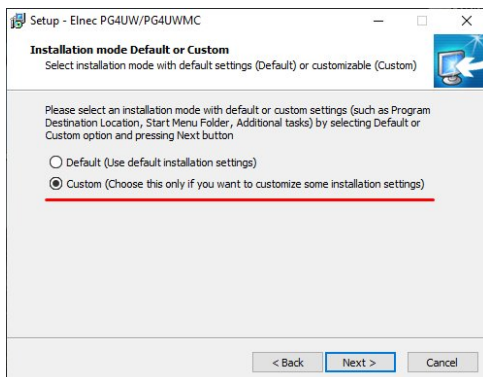


After some initial screens, an option to include installation of PG4UWMC Network Agent and selection of Programmers Group will appear. Please **define name of Programmers group**, which this installed computer will belong to. PG4UWMC Network Agent will be configured to start with windows.

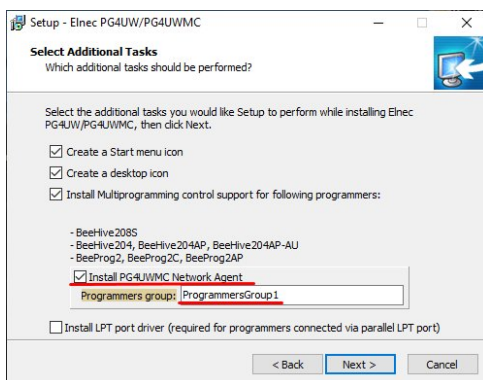
Installation procedure with command-line parameter /networkmode



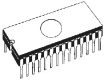




Installation procedure – customized

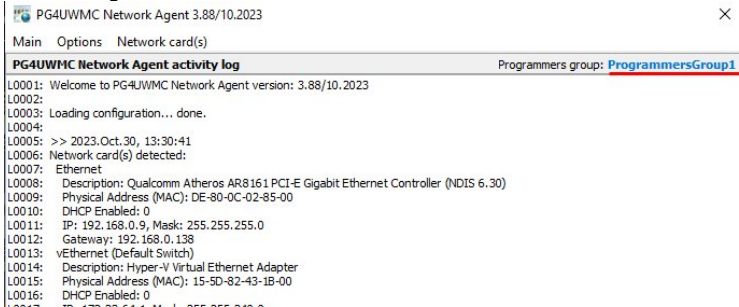


Installation procedure with checked Installation of PG4UWMC Network Agent and selected name of Programmers group



This way should PG4UW be installed on each computer on network which is considered to work in Programmers group.

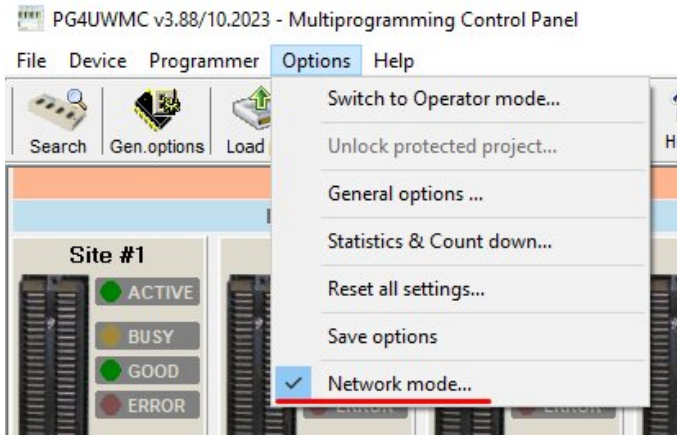
Each computer in Programmers group must have PG4UWMC Network Agent running in background. If PG4UWMC Network Agent is not running after installation, please, run it from Start menu / All Programs ...



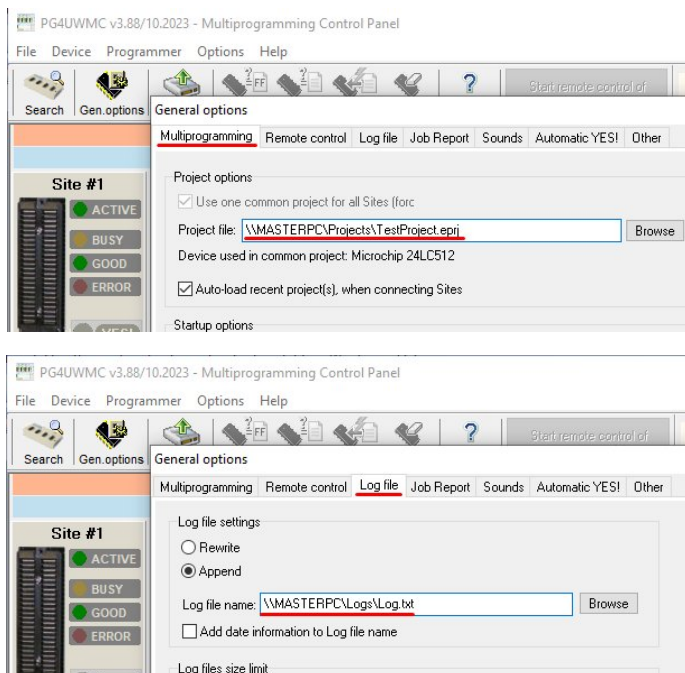
Once the installation is done on each computer, we can proceed to initial configuration of PG4UWMC.

### Configuration

Run PG4UWMC on computer which will control whole programming process. In Menu / Options check Network mode.



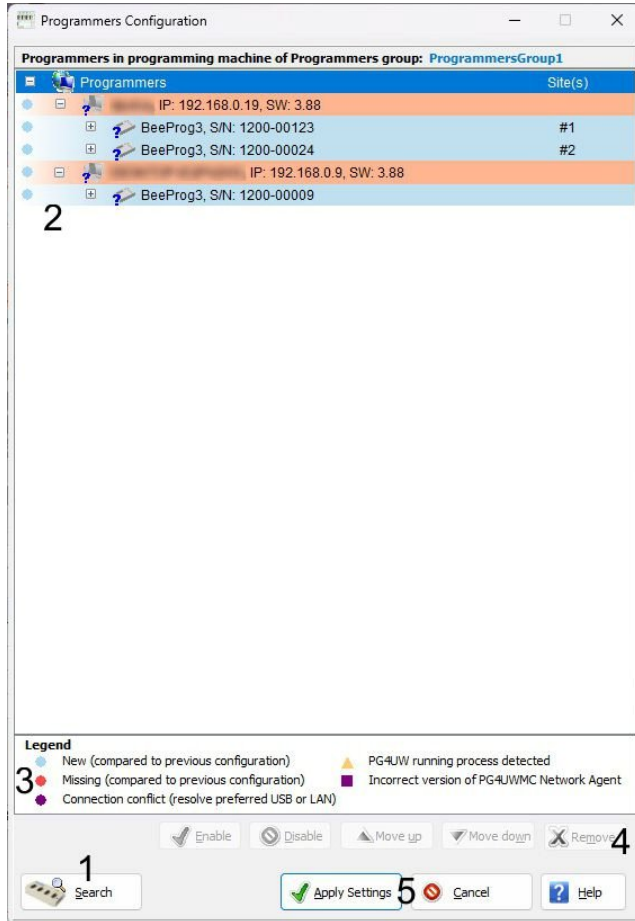
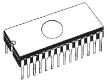
We are on network, thus we need to set network path to project file, and log file.



Configuring PG4UWMC read project from network, save logs to network paths.

Now we can proceed to first Search on network in defined Programmers group.

- Search for programmers
- Evaluate what was found
- Check the legend (for help what to do)
- Resolve problems to meet restrictions
- Enable, Disable, Move, Remove programmers as you desire
- Apply changes or Cancel.



### Search in Programmers group on network

From this point, working with PG4UWMC should be as usual.

#### Troubleshooting

If searching programmers does not finish as expected, please check following:

- each computer in Programmers group must run PG4UWMC Network Agent with same Programmers group
- your firewall settings may block network communication, please check firewall rules or temporarily disable firewall.

## **Programmer / Credit box info**

The menu item Credit box info provide all necessary information about Credit box (boxes) attached to the PC: type, serial number, information about activation and about credits available. If more Credit boxes attached to the PC, software shows also information about all credits available.

Availability of Credit box (boxes) and also information about amount of credits in attached Credit box (boxes) is also shown at main window, it is 'Credit box' button below the 'Statistics' and 'Count down' section. This button appears dynamically, when the device that belongs to 'Paid ISP support' is selected and if at least one Credit box is present.

The bar-graph at the bottom of 'Credit box' button also indicates status of credits on attached Credit box (boxes) as next:

green bargraph	85% of total credits available
yellow bargraph	credit box button 50% of total credits available
red bargraph	credit box button 10%, of total credits available
no bargraph	credit box button 0% of total credits available (depleted)

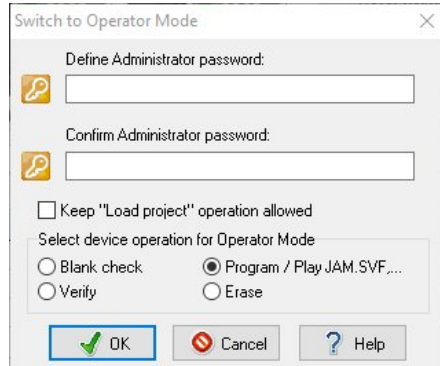
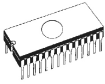
Information about available credits is periodically updated (also during opened Credit box info window).

## **Options / Switch to operator mode (administrator mode)**

Program PG4UWMC is set by default to **Administrator mode**. It means that no operation blocking for user is applied. But in production, there is suitable to block some menu commands, to ensure, user does not modify important program settings or configuration. **Operator mode** is used for this purpose.

PG4UWMC has **Operator mode** very similar to **Protected mode** of program PG4UW. The difference is, that Operator mode **can be activated by menu command** but **cannot be activated by Project file**. Another difference is that **Operator mode** settings of PG4UWMC are **saved to configuration .ini file** of PG4UWMC while program PG4UWMC is closed. During next start of application PG4UWMC the recent Operator mode settings obtained from .ini file are used. If you wish to see more information about **Protected mode** of single application PG4UW, please look at Options / Protected mode.

There is menu command - **Options / Switch to Operator mode...** that allows use Operator mode in PG4UWMC. After selecting the menu, Switch to Operator mode dialog appears. User has to enter password twice to confirm the password is correct. After successful password confirmation program switches to Operator mode.



**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.

To switch program from Operator mode back to Administrator mode, use the menu command **Options / Switch to Administrator Mode...** The „Password required" dialog appears. User has to enter the same password as the password entered during switch to Operator mode.

When Administrator mode is active, the label **Administrator mode** is visible in right top corner of **Programmers activity log**.

When Operator mode is active, the label **Operator mode** is visible in right top corner of **Programmers activity log**.

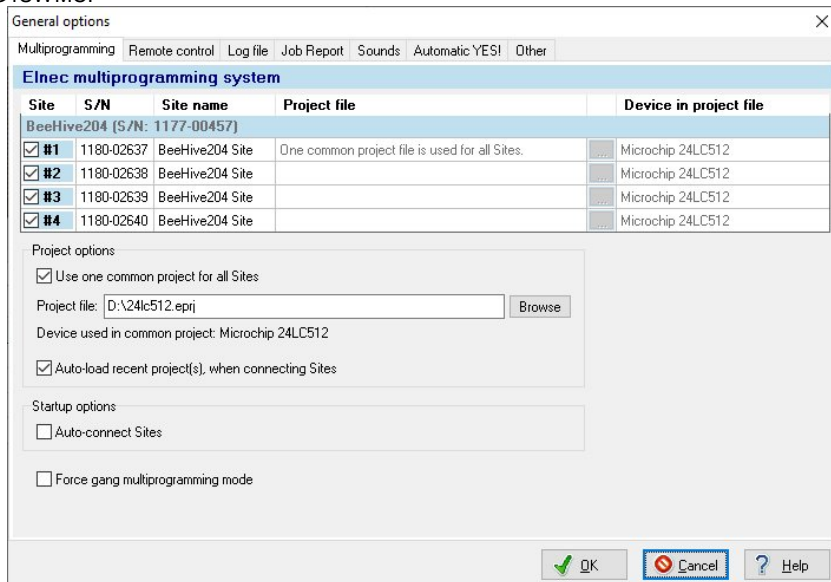
#### Notes:

*Sometimes when Administrator mode is switched from Operator mode, some commands (for example command "Load project") may remain disabled. This can be resolved by clicking on button Stop ALL.*

*If the project is **protected** and has defined **default operation**, during **load of project operation** (menu "File / Load project"), these **restrictions will be accepted** and device operations in PG4UWMC will be enabled or disabled accordingly. Also "Load project" operation can be affected this way. This is **applied only** when option "**Use Site #1 project for all Sites**" is selected. If the option is turned off, each Site will have its own project file, and no menu operations blocking forced by project settings will be performed. All operation protection settings from protected projects will be **ignored**.*

## Options / General options

PG4UWMC General options dialog is used to set or display configuration options of PG4UWMC.



Site	S/N	Site name	Project file	Device in project file
BeeHive204 (S/N: 1177-00457)				
<input checked="" type="checkbox"/> #1	1180-02637	BeeHive204 Site	One common project file is used for all Sites.	Microchip 24LC512
<input checked="" type="checkbox"/> #2	1180-02638	BeeHive204 Site		Microchip 24LC512
<input checked="" type="checkbox"/> #3	1180-02639	BeeHive204 Site		Microchip 24LC512
<input checked="" type="checkbox"/> #4	1180-02640	BeeHive204 Site		Microchip 24LC512

Project options

Use one common project for all Sites

Project file:

Device used in common project: Microchip 24LC512

Auto-load recent project(s), when connecting Sites

Startup options

Auto-connect Sites

Force gang multiprogramming mode

### panel Multiprogramming

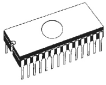
There is table which contains following columns:

- column **Sites** contains checkboxes with Site numbers #1, #2, #3, #4 used to enable/disable using individual Programmer Site with specified Site number
- column **Serial number S/N** contains information about serial numbers for Programmer Sites
- column **Site name**
- column **Project file** contains edit lines Project: #1, Project: #2 ...Project: #4 for setting individual projects to be loaded after running each PG4UW. Project file names can be entered manually or by dialog **Select project file**, which can be opened for each Site by clicking on button "...". If the project name edit line is blank, the automatic project load will not be performed.
- column **Device in project file**

Checkbox **Use one common project for all Sites** is placed under the Sites numbers and Projects table.

When there is requirement to program the same device types with the same data, the checkbox should be checked.

- If the checkbox is checked, one common project file will be used for all Programmer Sites. In this mode all Sites are using the same shared buffer of project data and program the same device type.



- If the checkbox is not checked, each Site will use its own project file defined by name in table of Sites in column **Project file**. In this mode each Site is using its own buffer of project data, which allows to program different data to different types of devices at the same time in each Site.

#### Checkbox **Auto-load recent project(s), when connecting Sites**

When checked, recent project(s) are automatically loaded every time programmer Sites (their PG4UWs) are started and connected to PG4UWMC. Sites connecting to PG4UWMC can be performed by two ways:

- automatically, after starting of PG4UWMC (see information about checkbox **Auto-connect Sites** below)
- manually by clicking button **Connect programmers** in main windows of PG4UWMC

#### Checkbox **Auto-connect Sites**

When checked, Sites are automatically connected after PG4UWMC start (user does not need to press Connect programmers button after starting of PG4UWMC).

#### Checkbox **Force gang multiprogramming mode**

Standard mode of multiprogramming operation on our multiprogrammers is **concurrent multiprogramming mode**, when each programming site works independently and operator can re-load programmed device while other programming sites are running. In **gang multiprogramming mode** predefined **operation starts on all programming sites at a time** by pressing any YES! button.

#### Notes:

- *Works for all active (present and enabled) sites.*
- *Start is blocked while any site is busy.*
- *In this mode Automatic YES! is disabled.*

#### Panel **Log file** are used to set mode of using Log file report

General options

Multiprogramming Remote control Log file Job Report Sounds Automatic YES! Other

Log file settings:

Rewrite

Append

Log file name: zers\Peter\AppData\Roaming\EItec\Pg4uw\reportmc.log Browse

Add date information to Log file name

Log files size limit

Use Log files text truncating when file size limit is reached

Maximum Log file size: 20000 kB (200 kB..200000 kB)

Amount of truncated text: 50%

Date format style in the log

yyyy.mm.dd (2017.Oct.11)

OK Cancel Help





Log file is text file containing information about PG4UWMC control program operation flow, which means information about loading project files, device operation types and device operation results. Multiprogramming system generates few of Log files. One main Log file of program PG4UWMC and Log files for each of running Programmer Sites. Each Site has its own one Log file. The name of Site's Log file has the same prefix as the name of Log file specified in edit box Log file. The file name prefix is followed by the number of Site in the form of `_#<Snum>`.

**Example:**

*The Log file name specified by user is: "report.log". Then names of 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...*

Following options can be set for Log file creation

- option **Append Log file** sets usage of Log file on. Log file will be created after the first restart of PG4UWMC. For all other next starts of PG4UWMC, the existing Log file will be preserved and new data will be appended to the existing Log file.
- option **Rewrite Log file** sets usage of Log file on. Log file will be created after the first restart of PG4UWMC. For all other next starts of PG4UWMC, the existing Log file will be rewritten and new Log file will be created. Data from previous Log file will be deleted.

Checkbox **Add date information to Log file name** allows user to set date information into Log file name specified by user in **Log file name** edit box. When the checkbox is checked, program automatically adds current date string into user specified Log file name by the following rules:

If user specified log file name has format:

**<user\_log\_file\_name>.<log\_file\_extension>**

The name with added date will be:

**<user\_log\_file\_name><-yyyy-mmm-dd>.<log\_file\_extension>**

The new part representing of 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 added date will look like this (have a date November, 7th, 2006):*

*c:\logs\myfile-2006-nov-07.log*

If do you wish to have log file name without any prefix before date information, you can specify the log file name as:

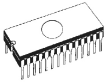
**<log\_file\_extension> - dot is the first in file name**

**Example:** *User specifies Log file name: c:\logs\log*

*The final log file name with added date will look like this (have a date November, 7th, 2006):*

*c:\logs\2006-nov-07.log*

**Advanced options about Log file size limit are available too:**



- **option Use Log file text truncating when file size limit is reached** - when checked, the Log file size limit is on. It means that when Log file size reaches specified value, the part of text included in Log file will be truncated. When the option is unchecked, the size of Log file is unlimited, respectively is limited by free disk space only.
- option **Maximum Log file size** specifies the maximum size of Log file in kB
- option **Amount of truncated text** specifies the percentage of Log file text, which will be truncated after Maximum Log file size is reached. The higher value means more text will be truncated (removed) from Log file.

**Note:** *Lines start with '+' are shown in the log file, but not in the log at screen to keep better overview of the on-screen log.*

#### **Common information:**

**Index of Programmer Site** is integer number from 1 to 8 which defines unambiguously each running Programmer Site.

**Serial number of Programmer Site** defines unambiguously the programmer or programmer site used. Instance will search all programmers connected on USB Bus until it finds programmer (site) with desired serial number. Programmers or Programmer Sites with different serial numbers will be ignored. If the PG4UWMC does not find desired Programmer Site, the Programmer Site will be set to Demo mode with status set to "Not found".

On one computer, 8 Programmer Sites can be run at the same time.

**Panel Job Report** settings are used to set mode of using Job Report.

Job Report represents the summary description of operation recently made on device. Job is associated with project file and it means the operation starting with Load project until loading of new project or closing program PG4UWMC.

**Job Report** contains following information:

- project name
- project date
- Protected mode status
- PG4UWMC software version
- programmer type and serial number
- start time of executing the Job (it means time when Load project operation was performed)
- end time of executing the Job (time of creating the Job Report)
- device name
- device type
- checksum
- device operation options
- serialization information
- statistics information

Job Report is generated in following cases:

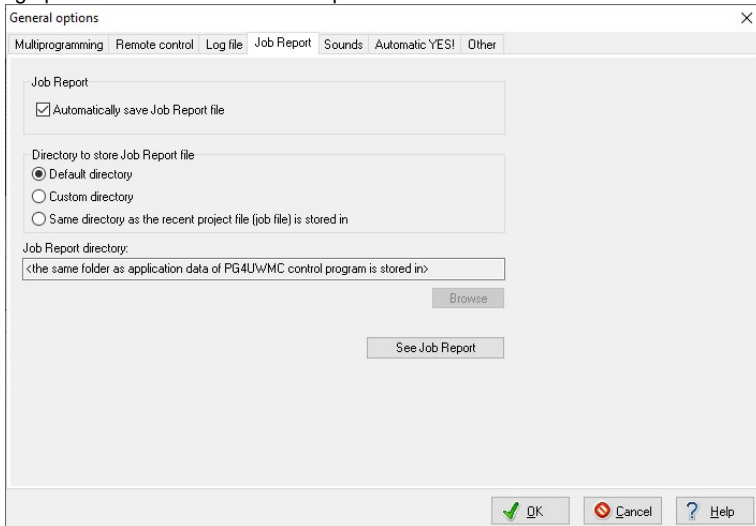
- user command Load project is selected
- closing or disconnecting programmer sites is selected
- closing the PG4UWMC
- device Count down counter reaches 0 (finished status)
- manually by user, when menu "File / Job Report" is used

The Job Report is generated for recently loaded project file, only when statistics value of Total is greater than 0.

It means, at least one device operation (program, verify...) must be performed.

Job Report dialog settings are in dialog PG4UWMC Settings (menu Options / Settings) in tab Job Report.

Following options are available for Job Report:



When the checkbox **Automatically save Job Report file** is checked, the Job Report will be saved automatically to directory specified in edit field Job Report directory and with file name created as following:

*job\_report\_<ordnum>\_<prjname>.jrp*

where

<ordnum> is decimal order of the file. If there are any report files with the same name, then order for new report file is incremented about order of existing files.

<prjname> is project file name of recently used project, and without the project file name extension.

**Example 1:** Let's use the project file c:\myproject.eprj and directory for Job Report set to d:\job\_reports\

There are no report files present in the Job Report directory.

The final Job Report file name will be:

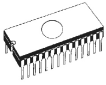
*d:\job\_reports\job\_report\_000\_myproject.jrp*

**Example 2:** Let's use the conditions from Example 1, but assume there is already one report file present.

Name of this file is d:\job\_reports\job\_report\_000\_myproject.jrp

The final Job Report file name of new report will be:

*d:\job\_reports\job\_report\_001\_myproject.jrp*

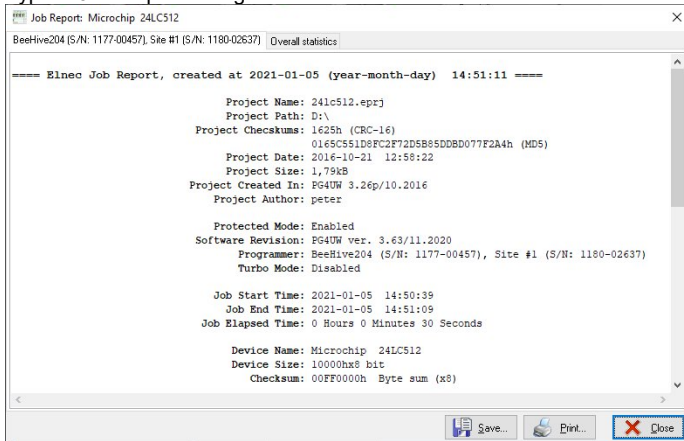


**Note:** The order inside file name is incremented by 1.

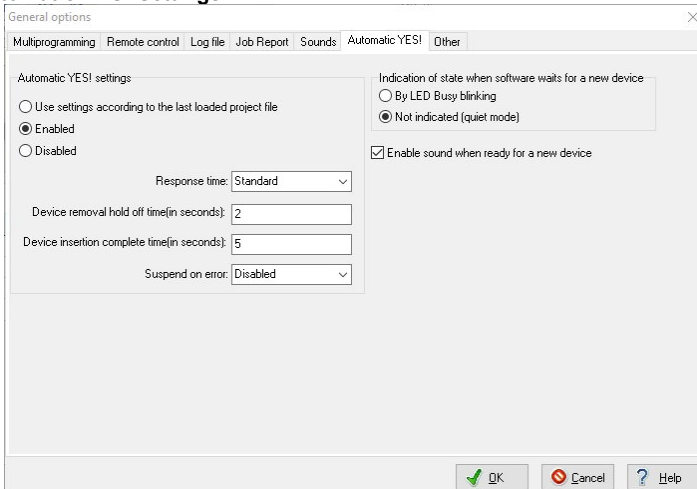
When **Automatically save Job Report file** setting is set, no Job Report dialogs appears when generating Job Report. Newly generated Job Report is saved to file without any dialogs or messages (if no error occurs while saving to file).

If the checkbox **Automatically save Job Report file** is unchecked, the PG4UWMC will show Job Report dialog every time needed.

In the Job Report dialog user can select operation to do with Job Report. If user selects no operation (Close button), the Job Report will be written to PG4UWMC Log Window only. Example of typical Job Report dialog is shown below:



### Panel Automatic YES! Settings



In this mode you just take off the programmed device, then put new device into ZIF socket and a last operation will be repeated automatically. Program automatically detects an insertion of a

new device and runs last executed operation without pressing any key or button. An insertion of device into ZIF is displayed on the screen. Repeated operation executing will be canceled by pressing key <Esc> during waiting for insert/remove a device to/from ZIF.

This feature may not be available for some types of programmers.

**Use settings according to the last loaded project file** - Automatic YES! options is set by the settings in project file. One of the setting's items of Automatic YES! is 'Pins of programmer's ZIF excluded from sensing', which is depend on used programming adapter. Because it is possible that different programmers use different programming adapters for same device, this setting will be ignored in this case and in the log window you can find following sentence: "None connected pins setting was not accepted due to different programming adapter. Please use automatic YES wizard again." If this case occur go on master programming site (if you run PG4UWMC with option 'Use Site #1 project for all Sites') or on programming site which wrote previous mentioned sentence to log and click on the button 'Setting Automatic YES! parameters' in Programmer / Automatic YES! options.

Settings of 'Indication of state when software waits for a new device' and 'Enable sound when ready for a new device' is not stored in project file.

**Enabled** - Automatic YES! function is enabled on all connected programming site with parameters set by PG4UWMC.

**Disabled** - Automatic YES! function is disabled on all connected programming site. Use this setting if you need to use button YES! for starting a next operation with the programmed device.

**Response time** - interval between insertion of the chip into the ZIF socket and the start of selected device operation. If longer positioning of the chip in the ZIF socket is necessary select elongated response time.

**Device removal hold off time** - time period between you removed device from the ZIF socket and the time when software starts to check the socket for new device inserted. This time is in seconds and must be from 1 to 120 (default value is 2 seconds).

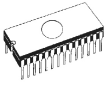
**Device insertion complete time** - time within all pins of the device have to be properly inserted after a first pin(s) detected so that the program will not detects incorrectly inserted device. This interval is in seconds and must be from 1 to 120 (default value is 5 seconds).

**Suspend on error** - defines if the Automatic YES! function will be temporary disabled on error to see result of operation or will going on without suspension.

Indication of state when software waits for a new device:

**Not indicated (quiet mode)** - the programmer, regardless of the number of ZIF sockets of the programmer, does not indicate the state when a device is programmed and the programmer with software wait for inserting a new device. After an operation with a device only one of the status LEDs Error or OK lights, in dependence on the result of previous operation. This LED goes off immediately after detecting removal of a device from the ZIF socket.

**By LED Busy blinking** - the programmer, regardless of the number of ZIF sockets of the programmer, indicates the state when a device is programmed and the programmer with



software wait for inserting a new device mode by blinking with the LED Busy. After an operation with a device is done, one of the status LEDs (OK or Error) lights, in dependence on the result of previous operation and the LED Busy is blinking. If the program detects removal of a device from ZIF socket, then the status LED goes off, but the LED Busy is still blinking to indicate readiness of the program to repeat last operation with new device. After the program indicates one or more pins of (new) device in the ZIF socket, the LED Busy goes light continually. From this point the program waits a requested time for insertion of the rest pins of new device. If a requested time (Device insertion complete time) overflows and a device is not correctly inserted, the program will light the LED Error to indicate this state. When new device is inserted correctly, the status LED goes off and a new operation with device is started.

**Enable sound when ready for a new device** - when checked, sound will be generated if SW detects complete empty ZIF socket and is ready to accept new device into ZIF socket.

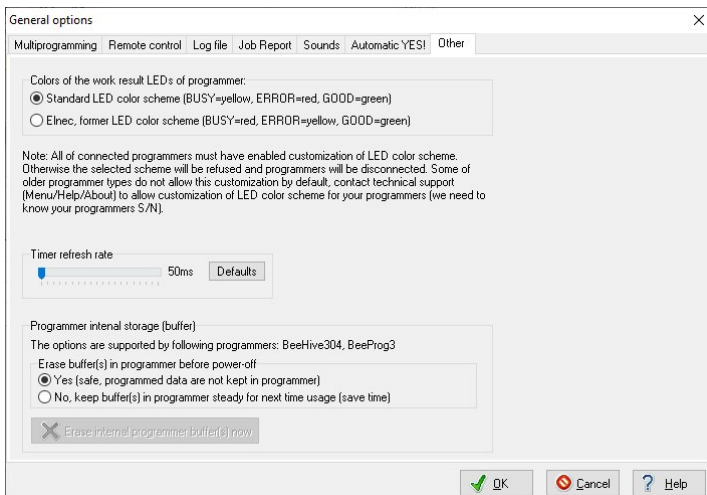
When any of previous options is selected and confirmed by OK button, PG4UWMC send selected settings to all connected programming site. Also if you set Automatic YES! parameters on master programming site these settings will be send to all connected slave programming site sites and to PG4UWMC.

For more details about Automatic YES! feature see Programmer / Automatic YES!.

### Panel Other

Colors of the work result LEDs of programmer:

- Standard LED color scheme (ERROR=red, BUSY=yellow)
- Former LED color scheme (ERROR=yellow, BUSY=red)



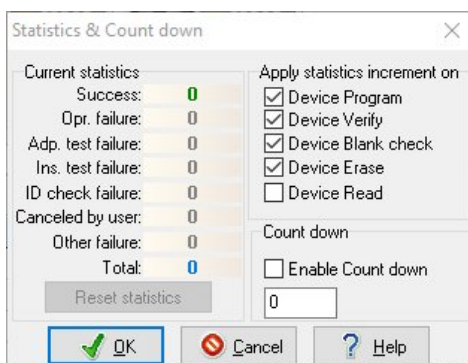
**Note:** These settings are available only for some types of programmers. If you can't see mentioned settings in menu, or menu is not enabled for editing, your programmer doesn't support LED color scheme customization.

**Timer refresh rate** defines how often the PG4UWMC program will request status information from running Programmer Sites. Status information means current device operation type, progress, result and so on. Current status information is displayed in main window of PG4UWMC. The default timer refresh rate value is 200ms. If you wish faster refresh of status information displayed in Operation panel of PG4UWMC, select shorter refresh interval. If you notice the system performance slow down, when using faster refresh, select higher refresh value to make refresh less often. On the Pentium 4 computers there is almost no performance penalty depending on timer refresh rate but on slower computers it is sometimes useful to select longer (less often) timer interval.

**Programmer internal storage (buffer)** Please take a look at General options/Buffer, at radio group Erase buffer(s) in programmer before power-off part help page.

## Options / Statistics & Count down

This command shows Statistics & Count down settings.



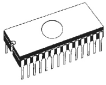
**Statistics** gives the information about actual count of device operations, which were proceeded on selected type device. If one device is corresponding to one device operation, e.g. programming, the number of device operations will be equal to number of programmed devices.

The next function of statistics is **Count down**. Count down allows checking the number of device operations, and then number of devices, on which device operations have to be done. After each successful device operation the value of count down counter is decremented. Count down has user defined start number of devices to do. When count down value reach zero, it means, specified number of devices is complete and user message about complete count down will be displayed.

Statistics & Count down contains following options:

Check boxes **Program**, **Verify**, **Blank**, **Erase** and **Read** define operations, after which statistics values increment.

Any selected and performed device operation will increment the **Total** counter and one of **Success** or **Failure** counters depending on device operation result (success or failure).



A combination of partial operations is counted as one operation only. For example, a Read operation including Verify after Read is one operation. A Program operation including Erase and/or Verify operations is counted as one operation.

Check box **Enable Count down** sets Count down activity (enable or disable). Edit box following the Count down check box defines initial number of count down counter, from which count down starts.

Statistics & Count down dialog can be also opened by pressing right mouse button on Statistics panel or Count down panel and clicking displayed item Statistics & Count down.

Statistics dialog contains seven statistics values – **Success**, **Operational failure**, **Adapter test failure**, **Insertion test failure**, **ID check failure**, **Cancelled by user**, **Other failure** (programmer SW or HW) and **Total**.

Meaning of the values is:

<b>Success</b>	number of operations which where successfully completed
<b>Operational failure</b>	number of operations which where not successfully completed due to error of device
<b>Adapter test failure</b>	number of operations which where not successfully completed due to incorrect programming adapter
<b>Insertion test failure</b>	number of operations which where not successfully completed due to incorrect position of device in programming adapter
<b>ID check failure</b>	number of operations which where not successfully completed due to incorrect ID code read from device
<b>Canceled by user</b>	number of operations which where not successfully completed due to canceled by user
<b>Other failure</b>	number of operations which where not successfully completed due to hardware error of programmer or control software error
<b>Total</b>	number of all operations

Actual statistics values are displaying in main window of control program in **Statistics & Count down** panel.

**Notes:**

**Reset statistics** button is disabled until any action is running on sites. To stop action on sites press button 'Stop All'

Reset button in Statistics panel clears statistics values to value 0.

The statistics information is saved to Job Summary report.

## Options / Reset all settings

**Reset all settings** dialog is used to reset all PG4UWMC settings to default state. It also deactivates Administrator and/or Protected modes recently used in PG4UWMC.

User is prompted to enter "Captcha", to confirm reset of all PG4UWMC settings. Changes are applied after restarting of PG4UWMC.



## Help

Menu **Help** contains commands that let you view supported devices and programmers and information about program version too.

Pressing the <F1> key accesses the Help. When you are selecting menu item and press <F1>, you access context-sensitive help. If PG4UW is executing an operation with the programmer <F1> generates no response.

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 manual on [www.elnec.com](http://www.elnec.com).*

*Since the Help system is continuously updated together with the control program, it may contain information not included in this manual.*

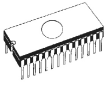
### Help / Supported programmers

The list of currently supported programmers can be displayed in PG4UWMC by menu Help | Supported programmers. Generally, supported programmers in PG4UWMC are 48-pindrive and 64-pindrive universal programmers with USB or LAN interface. Also all of our USB connected multiprogramming systems are supported. PG4UWMC can handle from 1 to 16 programmer sites. One programmer site means one ZIF socket module

### Help / Create problem report

Command **Create problem report** is used for writing more particular diagnostic information to Log window and consequently for creating a **Problem report information file** from **Log window**. The created file can be opened and content can be read by any text editor.

Problem report file is useful when problem appears during device programmer usage and kind of the error is so complicated, that user can not resolve it oneself and he must contact programmer manufacturer. When customer in this case sends only insufficient information about his problem to manufacturer, it might be not enough for detection of problem reason. Therefore it is recommended to send also Problem report file, which helps manufacturer to localize the reason of error and resolve it sooner.



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

- **Button Yes, last days logs.** This option creates smaller size Problem report .zip file, containing log files from last week. Log files included into .zip are all truncated to maximum file size about 1MB.
- **Button Yes, all days logs available.** This option creates complete Problem report, it means all available PG4UW/PG4UWMC log files are included into Problem report .zip file, and no file truncate is applied. Resulting Problem report .zip file could be much more bigger, than in Last days logs case, but it offers complete overview of operations

## About

When you choose the Info command from the menu, a window appears, showing copyright and version information.

## Programmers supported by PG4UWMC

The list of currently supported programmers can be displayed in PG4UWMC by menu Help / Supported programmers. Generally, supported programmers in PG4UWMC are 48-pin drive and 64-pin drive universal programmers with USB or LAN interface. Also all of our USB connected multiprogramming systems are supported. PG4UWMC can handle from 1 to 8 programmer sites. One programmer site means one ZIF socket module.

## Troubleshooting

### Serial numbers

For successful using of multiply programmers, correct serial numbers must be specified for each used programmer in panel Serial numbers. If there is empty field for serial number, application PG4UW for the programmer Site won't start.

When PG4UWMC application is searching for connected programmers in "Search for programmers" dialog, serial numbers of programmers are detected automatically. User does not need (and can not) specify serial numbers by himself.

### Communication error(s) while searching for programmers

If some kind of communication error(s) occurs, please close all PG4UW applications and PG4UWMC and then start PG4UWMC and click button "Connect programmers" to start PG4UW applications for each Site and connect programmers.

### All programmers are connected correctly but unstable working

If communication with programmers is lost randomly during device operation (for example device programming), please close other programs, especially programs which consumes large amount of system resources (multimedia, CAD, graphic applications and so on).

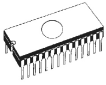
**Note:** We also recommend using computer USB ports placed on back side of computer and directly connected to motherboard, because computer USB ports connected to computer motherboard indirectly - via cable, may be unreliable when using high speed USB 2.0 transfer modes. This recommendation is valid not only for programmers, but also for other devices.



---

## *Common notes*

---



## **Maintenance**

We recommend following the instructions and precautions herein to achieve high reliability of the programmer for a long period of time.

The programmer maintenance depends on character and amount of its use. Regardless, the following recommendations are generally accepted:

- Do not use and store the programmer in dusty places.
- Humidity accelerates sedimentation of debris and dust in ZIF socket and Programming Module Interface (PMI) connectors.
- After end the job cover the ZIF socket and Programming Module Interface (PMI) connectors.
- Do not expose the programmer to direct sunlight or position near a source of heat.

### ***Intensive daily use (programming centre, production)***

#### **Daily maintenance**

Check the ZIF sockets of the programming module for their condition and wear. Remove debris, dust and grime from the ZIF sockets with clean, dry and compressed air. Clean the ZIF sockets both in closed and opened state.

#### **Weekly maintenance**

Perform the Selftest for every programmer or programming site.

#### **Quarterly maintenance**

Gently clean the surface of the programmer with isopropyl alcohol or technical alcohol on a soft cloth.

Remove debris, dust and grime from the Programming Module Interface (PMI) connectors with clean, dry and compressed air.

Perform the calibration test.

### ***Daily use (developing laboratory, office)***

#### **Daily maintenance**

After end of the job cover the ZIF socket of the programming module. It is also recommended protecting the ZIF socket of programming modules from dust and grime.

#### **Weekly maintenance**

Check the ZIF socket of the programming modules for their condition and wear. Remove debris, dust and grime from the ZIF sockets with clean, dry and compressed air. Clean the ZIF sockets both in closed and opened state.

#### **Quarterly maintenance**

Perform the Selftest for every programmer or programming site.

#### **Biannual maintenance**

Gently clean the surface of the programmer with isopropyl alcohol or technical alcohol on a soft cloth.

Remove debris, dust and grime from the Programming Module Interface (PMI) connectors with clean, dry and compressed air.

Perform the calibration test.

## Occasional use

### Daily maintenance

After end of the job cover the ZIF socket of the programming module. It is also recommended protecting the ZIF socket of programming modules from dust and grime.

### Quarterly maintenance

Check the ZIF socket of the programming modules for their condition and wear. Remove debris, dust and grime from the ZIF sockets with clean, dry and compressed air. Clean the ZIF sockets both in closed and opened state.

### Biannual maintenance

Perform the Selftest for every programmer or programming site.

### Annual maintenance

Gently clean the surface of the programmer with isopropyl alcohol or technical alcohol on a soft cloth.

Remove debris, dust and grime from the Programming Module Interface (PMI) connectors with clean, dry and compressed air.

Perform the calibration test.

### Warning:

*The ZIF socket of the programming modules is considered as consumables. The guaranteed mechanical life cycle of the programming module ZIF socket given by manufacturer of the ZIF socket is generally from 5.000 to 10.000 mechanical cycles (actuations), even the life cycle of some specialized BGA ZIF sockets can be about 500.000 mechanical cycles. Programmed devices, environment and ZIF socket maintenance have direct influence to actual electrical lifetime of ZIF socket (it means that ZIF socket does not cause programming failures yet). Keep fingers away from contacts of ZIF socket, because contacts of the ZIF socket fouled by smear and grime from fingers may cause the programming failures. Change the ZIF socket or the socket converter if you noticed increased number of programming failures.*

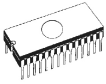
*The warranty does not apply to the ZIF sockets that are wear or grimy and which cause large amount of failures during working with programmer.*

## Software

PG4UW is common control program for all of the ElneC programmers. During work with this SW you can find some features and menu items that are not available for currently selected programmer.

### Command line parameters

We recommend using special utility **pg4uwcmd.exe** to make command line parameter control of PG4UW. For backward compatibility there is possible to use some command line parameters also directly with **pg4uw.exe**, but better way is to use **pg4uwcmd.exe**, which has support of more command line commands and also it has capability to return ExitCode (or ErrorLevel) value indicating success or error result of executing command line parameters. For more information about using **pg4uwcmd.exe** command line controller for **PG4UW**, please take a look at **Remote command line control of PG4UW**.



---

**Command line parameters which can be used directly with pg4uw.exe**

<code>/Prj:&lt;file_name&gt;</code>	forces project load when program is starting or even if program is already running, <file_name> means full or relative project file path and name
<code>/Loadfile:&lt;file_name&gt;</code>	forces file load when program is starting or even if program is already running, <file_name> means full or relative path to file that has to be loaded, file format is detected automatically
<code>/Saveproject:&lt;file_name&gt;</code>	the command is used to save currently selected device type, buffer contents and configuration to project file. Command <code>/Saveproject:...</code> is equivalent to user selected command <code>Save project</code> in PG4UW control program.

Please note, the file name Windows conventions must be fulfilled. It means also, that when file name contains spaces, the command line parameter must have the file name bounded inside quotation marks.

**Examples:**

```
/prj:c:\myfile.eprj
```

Load project file with name `c:\myfile.eprj`.

```
/loadfile:"c:\filename with spaces.bin"
```

Load file `"c:\filename with spaces.bin"` to buffer.

<code>/Program[:switch]</code>	forces start of "Program device" operation automatically when program is starting, or even if program is already running, also one of following optional switches can be used:
switch 'noquest'	forces start of device programming without question
switch 'noanyquest'	forces start of device programming without question and after operation on device is completed, program doesn't show "Repeat" operation dialog and goes directly into main program window

**Examples:**

```
1. /Program
```

```
2. /Program:noquest
```

```
3. /Program:noanyquest
```

<code>/Close</code>	this parameter has sense together with <code>/Program</code> parameter only, and makes program to close automatically after device programming is finished successfully
<code>/Close: always</code>	this parameter has sense together with <code>/Program</code> parameter only, and makes program to close automatically after device programming is finished, no matter if device operation was successful or not.

Basic rules for using of executive command line parameters:

- command line parameters are not case sensitive
- command line parameters can be used when first starting of program or when program is already running
- if program is already running, then any of command line operation is processed only when program was not busy (no operation was currently executing in program). Program must

be in basic state, i.e. main program window focused, no modal dialogs displayed, no menu commands opened or executed.

- order of processing command line parameters when using more parameters together is defined firmly as following:
  1. Load project (/Prj:...)
  2. Load file (/Load file:...)
  3. EPROM/Flash select by ID
  4. Program device (/Program[:switch])
  5. Close of control program (/Close only together with parameter /Program)

### Available command line parameters for starting program PG4UW in demo mode

Demo mode is useful in situations, when no programmer device is available. Demo mode can be used by clicking button Demo in dialog Find programmer or by command line parameter /demo. Recommended usage of the parameter is:

```
pg4uw.exe /demo /<programmer name>
```

where <programmer name> has to be replaced by name of wished programmer as it is used in PG4UW control program.

## Remote command line control of PG4UW

PG4UW can accept set of commands from the command line (command line parameters). The remote control can be achieved also by these command line parameters, but more efficient way is to use special tool **pg4uwcmd.exe**, which has many advantages. The main advantage is size of the **pg4uwcmd**, which result the calling of **pg4uwcmd** results a much faster response than calling of PG4UW directly.

Program pg4uwcmd.exe can be used to:

1. start PG4UW application with specified command line parameters
2. force command line parameters to PG4UW that is already running

Very good feature of pg4uwcmd.exe is its return code according to command line parameters operation result in PG4UW.

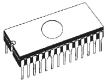
### Return values of pg4uwcmd.exe

If the command line parameters processed in PG4UW were successful, the ExitCode (or ErrorLevel) of pg4uwcmd.exe is zero. Otherwise the ExitCode value is number 1 or more.

Return value of program pg4uwcmd.exe can be tested in batch files.

### Following executive command line parameters are available to use with pg4uwcmd.exe

/Prj:<file_name>	Loads project file. Parameter <file_name> means full or relative project file path and name.
/Loadfile:<file_name>	Loads file. Parameter <file_name> means full or relative path to file that has to be loaded. File format is detected automatically
/Program[:switch]	Forces start of "Program device" operation automatically when program is starting, or even if program is already running. Also one of following optional switches can be used:
switch 'noquest'	forces start of device programming without question



switch 'noanyquest' forces start of device programming without question and after operation on device is completed, program doesn't show "Repeat" operation dialog and goes directly into main program window

**Examples:**

```
/Program  
/Program:noquest  
/Program:noanyquest
```

/Close This parameter has sense together with /Program parameter only, and makes program PG4UW to close automatically after device programming is finished (no matter if operation was successful or not).

/Saveproject:<file\_name> The command is used to save currently selected device type, buffer contents and configuration to project file. Command /Saveproject... is equivalent to user selected command Save project in PG4UW control program.

/writebuffer:ADDR1:B11,B12,B13,B14,...,B1N[:ADDR2:B21,B22,B23,B24,...,B2M]...  
Command /writebuffer is used to write block of Bytes to PG4UW main buffer at specified address. Write buffer command has one block of data required and other block(s) of data (marked with [...]) optional. Please do not use spaces or tabs in the command.  
Buffer address is always defined as Byte address, it means, that for buffer organization x16, the address AAAAx16 in buffer has to be specified in command /writebuffer as 2\*AAAA (x8).

**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 looks like following:  
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 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 looks like following:  
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]..`

Command `/writebufferex` is used to write block of Bytes to PG4UW main buffer at specified address. The command is very similar to command `/writebuffer`, except one more parameter – INDEX.

The INDEX parameter specifies the order of buffer, where data will send. The main buffer has index '1'. The first secondary buffer has index '2', etc. Please note, the secondary buffer(s) is(are) available for some kinds of devices only (e.g. Microchip PIC16F628). The kind of buffer indexed by parameter `buffindex` depends on order of buffer in application PG4UW in dialog View/Edit buffer. For example device Microchip PIC16F628 has additional buffer with label "Data EEPROM". This buffer can be accessed for data write(s) by this function when `buffindex = 2` is specified.

#### Example 1:

```
/writebufferex:1:7FF800:12,AB,C5,D4,7E,80
```

The command is equivalent to 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
```

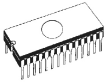
The command writes 6 Bytes 12H ABH C5H D4H 7EH 80H to secondary buffer with index "2" at address 2FH. The addressing looks like following:

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. program `pg4uwcmd.exe` must be located in the same directory as program `pg4uw.exe`
2. if `pg4uw.exe` is not running when `pg4uwcmd.exe` is called, it will be automatically started
3. command line parameters are not case sensitive
4. command line parameters can be used when first starting of program or when program is already running
5. if program is already running, then any of command line operation is processed only when program was not busy (no operation was currently executing in program). Program must be in basic state, i.e. main program window focused, no modal dialogs displayed, no menu commands opened or executed
6. order of processing command line parameters when using more parameters together is defined firmly as following:
  - step1 Load file (`/Loadfile:...`)
  - step2 Load project (`/Prj:...`)
  - step3 EPROM/FLASH autoselect
  - step4 Program device (`/Program[:switch]`)
  - step5 Close of control program (`/Close` only together with parameter `/Program`)

**Example 1:**

```
pg4uwcmd.exe /program:noanyquest /loadfile:c:\empfile.hex
```

Following operations will perform:

1. start pg4uw.exe (if not already running)
2. load file c:\empfile.hex
3. start program device operation without questions
4. pg4uwcmd.exe is still running and periodically checking status of pg4uw.exe
5. when device programming completes, pg4uwcmd.exe is closed and is returning ExitCode depending on load file and device programming results in pg4uw.exe. When all operations were successful, pg4uwcmd.exe returns 0, otherwise returns value 1 or more.

**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:**

Let's assume the PG4UW control program is running, and has user selected device. We need to load required data to PG4UW device buffer and save the selected device settings and buffer content to project file. Data required for device are stored in file c:\15001-25001\file\_10.bin. Project file will be stored at c:\projects\project\_10.eprj.

Following command line parameters should be specified to realize wished operation:

```
pg4uwcmd.exe /loadfile:c:\15001-25001\file_10.bin /saveproject:c:\projects\project_10.eprj
```

When PG4UW receives the commands, it will do following procedures:

1. loads data file c:\15001-25001\file\_10.bin
2. saves the currently selected device settings and buffer data to project file c:\projects\project\_10.eprj

If the result of operations performed is OK, PG4UWcmd application will return ExitCode (or ErrorLevel) value 0.

If there are some errors (can not load file or save to project file), PG4UWcmd application will return ExitCode value equal or greater than 1.

**Note:** When using the above commands, user must be sure the PG4UW is not performing any device operation, for example device programming. If the PG4UW is busy, it will refuse the commands and returns error status (ExitCode equal or greater than value 1).

## Command line parameters PG4UWMC

Program PG4UWMC supports following command line parameters:

### **/prj:<file\_name>**

Loads project file. Parameter <file\_name> means full or relative project file path and name. There is also available to make Load project operation from command line by entering project file name without prefix /prj...

### **Example:**

```
pg4uwmc.exe c:\projects\myproject.eprj  
Makes load project file "c:\projects\myproject.eprj".
```

### **/autoconnectsites**

The command forces PG4UWMC to connect programmer Sites (start control program PG4UW for each Site) during start of PG4UWMC, for all Site-s that were used, when PG4UWMC was recently closed. There is also equivalent option "Auto-connect Sites" available in "Settings" dialog of PG4UWMC.

## Hardware

If you find none solution, please document the situation, i.e., provide us an accurate description of your PC configuration, including some other circumstances bearing on the problem in question, and advise the manufacturer of your problem. Don't forget please enter of PC type, manufacturer, speed, operation system, resident programs; your parallel port I/O manufacturer and type. Use please **Device problem report** form for this purpose.

### **Warning:**

*Class A ITE notice*

*Devices described at this manual are class A products. In domestic environment this products may cause radio interference in which case the user may be required to take adequate measures.*

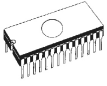


*Because BeeProg3 has internal power supply, follow these special precautions:*

- *Circuit breakers (overcurrent protection) must be a part of building electrical installation.*
- *Pull out power cord plug from outlet to disconnect programmer from mains. The outlet must be placed near to programmer and must be easy accessible.*

## Other

Please don't move **Info** application window while BUSY LED is on - supervisor circuit can be active and switch the programmer into safe mode as in case of communication PC-programmer error.



---

## *Troubleshooting and warranty*

---

## Troubleshooting

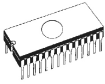
We really want you to enjoy our product. Nevertheless, problems can occur. In such cases please follow the instructions below.

- It might be your mistake in proper operation of the programmer or its control program PG4UW.
  - Please read carefully all the enclosed documentation again. Probably you will find the needed answer right away.
  - Try to install programmer and PG4UW on another computer. If your system works normally on the other computer you might have a problem with the first one PC. Compare differences between both computers.
  - Ask your in-house guru (every office has one!).
  - Ask the person who already installed programmer.
- If the problem persists, please call the local dealer, from whom you purchased the programmer, or call Elnec direct. Most problems can be solved by phone, e-mail or fax. If you want to contact us please use following methods:
  - **Mail/fax** - Copy the "**DEVICE PROBLEM REPORT**" form and fill it in following the instructions at the end of the form. Write everything down that you consider being relevant about the programmer, software and the target device. Send the completed form by mail or fax to Elnec (fax number in the control program, menu **Help / About**) or to your local dealer. If you send the form by fax please use black ink, a good pen and large letters!
  - **E-mail** - Use "**DEVICE PROBLEM REPORT**" form on the CD or from our Internet site and fill it in following the instructions at the end of the form. Use standard ASCII editor. Write everything down that you consider being relevant about the programmer, software and the target device. Send the completed form by e-mail to your local dealer or to Elnec ((nospam version) **elnec at elnec dot com**).
  - **Phone** - Copy "**DEVICE PROBLEM REPORT**" form and fill it in following the instructions at the end of the form. Write everything down that you consider being relevant about the programmer, software and the target device. Send the completed form by mail or fax to Elnec (fax number in the control program, menu **Help / About**) or to your local dealer. If you send the form by fax please use black ink, a good pen and large letters easily to read. Then call your local dealer or Elnec customer support center (phone number in the control program, menu **Help / About**). Please keep your manual, the programmer and the completed "**DEVICE PROBLEM REPORT**" form (just faxed) available, so that you can respond quickly to our questions.
- If your programmer is diagnosed as defective, consult your local dealer or Elnec about the pertinent repair center in your country. Please carefully include the following items in the package:
  - defective product
  - completed "**DEVICE PROBLEM REPORT**" form
  - photocopy of a dated proof of purchase

***Without all these items we cannot admit your programmer to repair.***

**Note:**

You may find the "**DEVICE PROBLEM REPORT**" form at our Internet site ([www.elnec.com](http://www.elnec.com)), section **Support / Problem report**.



## ***If you have an unsupported target device***

If you need to operate on a target device not supported by the control program for programmer, please do not despair and follow the next steps:

- Look in the device list of the latest version of the control program on our Internet site (section Download, file corresponding to your programmer). Your new target device might already be included in this version! If yes, download the file pg4uwarc.exe and install the new version of the control program.
- Contact Elnec direct, filling up a "Device Problem Report" form following the instructions at the end of this form. We may need detailed data sheets of your target device and, if possible, samples. The samples will be returned to you after we include your target device in a new version of PG4UW.

## ***Warranty terms***

The manufacturer, Elnec s.r.o. Presov, Slovakia, gives a guarantee on failure-free operating of the programmer and all its parts, materials and workmanship for **three-year** (BeeHive304 and BeeProg3) from the date of purchase. If the product is diagnosed as defective, Elnec s.r.o. or the authorized repair center will repair or replace defective parts at no charge. Parts used for replacement and/or whole programmer are warranted only for the remainder of the original warranty period.

For repair within the warranty period, the customer must prove the date of purchase.

This warranty terms are valid for customers, who purchase a programmer directly from Elnec company. The warranty conditions of Elnec sellers may differ depending on the target country law system or Elnec seller's warranty policy.

The warranty does not apply to products that are of wear and tear or mechanically damaged. Equally, the warranty does not apply to products opened and/or repaired and/or altered by personnel not authorized by Elnec, or to products that have been misused, abused, accidentated or that were improperly installed.

For unwarrantable repairs you will be billed according to the costs of replacement materials, service time and freight. Elnec or its distributors will determine whether the defective product should be repaired or replaced and judge whether or not the warranty applies.

Elnec has used its best efforts to develop hardware and software that is stable and reliable. Elnec does not guarantee that the hardware and software are free of "bugs", errors or defects. Elnec's liability is always limited to contract's net value paid by a buyer.

***Manufacturer:***

✉: **Elnec s. r. o.**, Jana Bottu 5, SK - 08001 Presov, Slovakia

☎: +42151/77 34 328, 77 31 007, fax 77 32 797

[www.elnec.com](http://www.elnec.com), e-mail (nospam version): elnec at elnec dot com

Elnec is not liable for:

**Damage caused by inappropriate use or handling of products.**

- Damage caused by users or third parties modifying or trying to modify products.
- damage caused by viruses, root kits, etc.
- Any further damage or consequent damage caused by hardware errors or software “bugs”.

**For example:** *lost profits, lost savings, damages arose from claims of third parties against a client, damage or loss of recorded data or files, renown, loss caused by impossibility to use etc.*

**Elnec s.r.o.**

Jana Bottu 5  
SK – 080 01 Presov  
Slovakia

[www.elnec.com](http://www.elnec.com)

ZLI-0330