Programmers by

ASIX®

User's Guide

Address:	ASIX s.r.o. Staropramenna 4 150 00 Prague Czech Republic
E-Mail:	sales@asix.net (sales inquiries, ordering) support@asix.net (technical support)
WWW:	tools.asix.net (development tools) www.asix.net (company website)
Tel.:	+420-257 312 378
Fax:	+420-257 329 116

Table of Contents

Introduction

This manual describes programmers manufactured by ASIX and controlling software for programming.

First chapter describes PRESTO USB programmer, its installation, possible programmer to device wirings and its basic features. In the section about wirings of the programmer there are notes to problems related with the device programing under the pictures.

Second chapter shortly describes the others programmers, their features and possible programmed devices.

Third chapter is about software UP. This software is usable for controlling of all ASIX programmers. The programmers allow ICSP programming. The UP can be interactively controlled using command line, Windows messages and DLL library.

For programming devices using JTAG interface serves program Jtag Player. Its usage is described in the fourth chapter.

Fifth chapter is about Precog. The Precog is used for programming Cyan Technology eCOG1 microcontroller. It allows simple debugging too.

In appendixes there are described positions of configuration words in supported PIC devices and UP_DLL.DLL settings and names. Appendix C is short chapter about ICSP programming. There is information about maximal currents drawn from the programmer, solutions of possible problems related with it and the other information.

1 <u>PRESTO</u>

PRESTO is a very fast and flexible USB programmer for programming and testing of wide variety of popular integrated circuits - microcontrollers, serial EEPROMs, CPLD, FPGA and more. The programmer is optimized to achieve maximal speed of programming yet for affordable price. It also features overcurrent protection on Vpp and Vcc and overvoltage protection on Vcc. The programmer is powered from USB and it is capable of either supplying power to the application circuitry or using the power from application circuitry during the programming.

1.1 Using

PRESTO is designated for programming and testing of the integrated circuits directly in the application circuitry. List of supported parts includes:

- **Microchip PIC** microcontrollers – parts with serial programming (Flash, EPROM and OTP), what are all PICs and dsPICs except of several obsolete parts.
- **Atmel AVR** microcontrollers – all parts supporting "SPI Low Voltage Serial Downloading", e.g. ATtiny12, AT90S8535 or Atmega128.
- **Atmel AVR32** microcontrollers - e.g. AT32UC3A1256
- **Atmel 8051 family** devices – parts supporting ISP programming, e.g. AT89S8253, AT89LP4052, AT89LP216 or AT89S2051
- **eCOG1** microcontrollers by Cyan Technology
- **Texas Instruments** microcontrollers – flash MSP430 and CCxxxx devices
- **Cypress** – PSoC devices
- **Serial EEPROM** - I^2C (24LCxx), Microwire (93LCxx) and SPI (25Cxx)
- Parts with **JTAG** interface, for which a software generating a SVF or XSVF file is available. These include CPLD (e.g. Xilinx XC95xx and CoolRunner), configuration memories for FPGA (e.g. Xilinx XC18Vxx and XCFxxS), microcontrollers (e.g. ATmega128) and more.
- Parts with **ARM** core – programming and debugging of AT91SAM7 microcontrollers is supported with ARMINE utility

1.2 Installation

To install the software the user must possess privileges of local Administrator for both the driver installation and running the software for the first time at which point another driver is installed. Normal user privileges will suffice for further usage of the software.

Install PRESTO drivers **BEFORE** first installation of UP v.2.0 or higher!

1.3 PRESTO Driver installation

Note: With new drivers Windows 98 complains about missing .CAT files during installation. In this case it is recommended to use older drivers "ASIX_PRESTO_USB_DRIVERS_2004-04-06.ZIP" available at http://tools.asix.net/dwnld_presto.htm .

Insert the installation CD-ROM into the drive and connect PRESTO to a USB port. The operating system detects a new device and starts searching for the drivers:

Choose the recommended automatic installation. The operating system finds the driver on the CD-ROM and prompts you for confirmation to install uncertified drivers. At this point choose "Continue Anyway".

Successful installation of the driver is announced by a system notice.
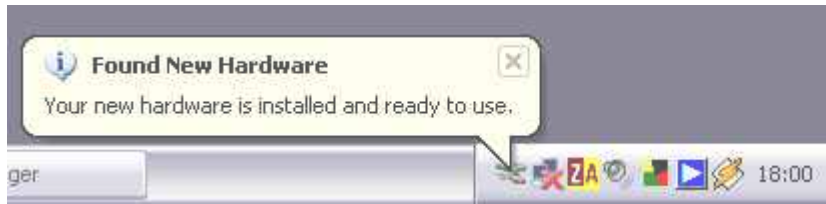
Functionality of the driver may be checked in the hardware manager.

Driver installation under Windows 7

For Windows 7 use the latest driver from web or the driver included on the PRESTO accessory CD - ROM. Under Windows 7 the driver will not install automatically, it must be installed manually consequently: Open Device Manager and find the connected PRESTO programmer there. Open its properties window and select "Update driver". Select a location where the unpacked driver is saved. The Device Manager must be accessed by right mouse button clicking on the Computer icon and selecting properties.

Now, when the PRESTO driver is installed, you can install the software, you will use:

UP - The program UP supports PRESTO as well as other ASIX's programmers. It offers many advanced functions like projects, command line control, Windows message control, workspace setup including user keyboard shortcut definitions, serial number generation by various methods, etc.

JTAG SVF PLAYER – This software is usable for programming devices with JTAG interface

PRECOG – This software is usable for programming of eCOG microcontrollers

ARMINE – ARMINE is a software package providing with FLASH programming and debugging of microcontrollers embedding ARM core. ARMINE package is based on OpenOCD with added support for PRESTO, necessary extensions for FLASH programming and a simple but convenient GUI frontend.

Note: When PRESTO looks to be damaged, the user can test it using software located at http://tools.asix.net/supp_testers.htm .

1.4 Description of the programming connector
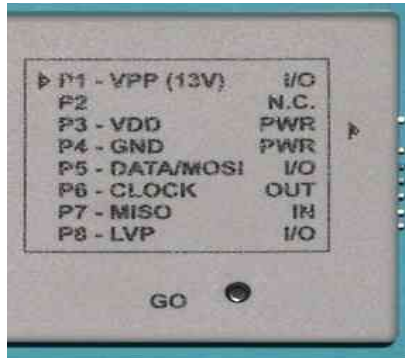
Pin	AVR[3]	AVR TPI	8051 arch.	JTAG	eCOG[7]
P1	Reset	Reset	Reset	USR[4]	CS
P2					
P3	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]
P4	GND	GND	GND	GND	GND
P5	MOSI	TPIDATA	MOSI	TDI	MOSI
P6	SCK	TPICLK	SCK	TCK	CLK
P7	MISO		MISO	TDO	MISO
P8				TMS	LOADB[5]

Pin	PIC ICSP	MSP430	MSP430 SBW	CCxxxx	PSoC	I2C	MicroWire	SPI
P1	V_{PP}	TEST		Reset	XRST		CS	#CS
P2								
P3	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]	V_{CC}[8]
P4	GND	GND	GND	GND	GND	GND	GND	GND
P5	DATA	TDI	SBWTDIO	Debug data	ISSP_SDATA	SDA	DI	SI
P6	CLK	TCK	SBWTCK	Debug clock	ISSP_SCLK	SCK	CLK	SCK
P7		TDO					DO	SO
P8	LVP[2]	TMS					ORG[2]	

1 – build in PullUp resistor in PRESTO
2 – pin may be left unconnected if suitably wired in the application circuitry
3 – crystal oscillator required, if no other clock source is used
4 – selectable function TRST, SCK or user
5 – log.0 / Z, PullUp in application circuitry necessary
6 – log.0 / Z
7 – crystal oscillators of 32.768 kHz and 5.000 MHz required
8 – Build-in PullDown 1 kΩ

Notes:

- **For more information see devices connection examples.**
- **If the target unit is powered from some switched power supply, or if it is not grounded, so there could be very high voltage difference between ground of PRESTO and ground of the target device, it could cause its failure.**
  The "correct" way how to connect PRESTO to the target is to connect PRESTO and target together, than turn on power of the target and than connect PRESTO into USB.
  Much simpler way to prevent this is to ground the board before PRESTO is connected - simply, to have ground pin of PRESTO to be a slightly longer (to make sure it is the first pin which is connected), like USB connectors are having it. (PRESTO is "grounded" to ground of the PC.)

1.5 Examples of the programmer to application wiring

Standalone PIC without application, using HVP (13V) mode
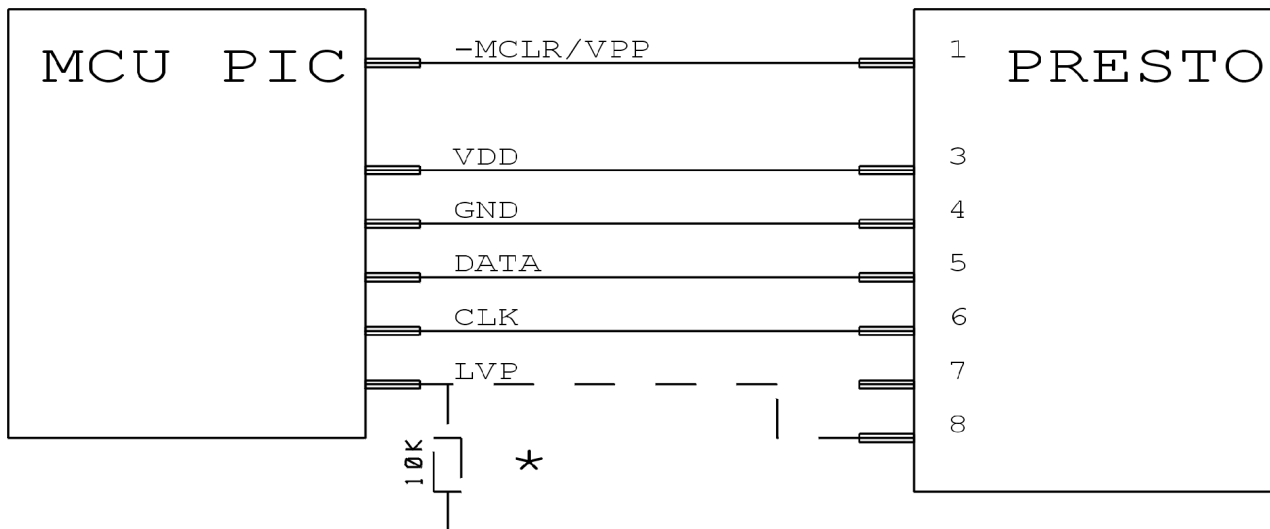
* Connect either to PRESTO or pull-down

See notes to PIC microcontrollers and configuration word address.

Onboard PIC, using LVP (PGM) mode (not 13V), powered from application
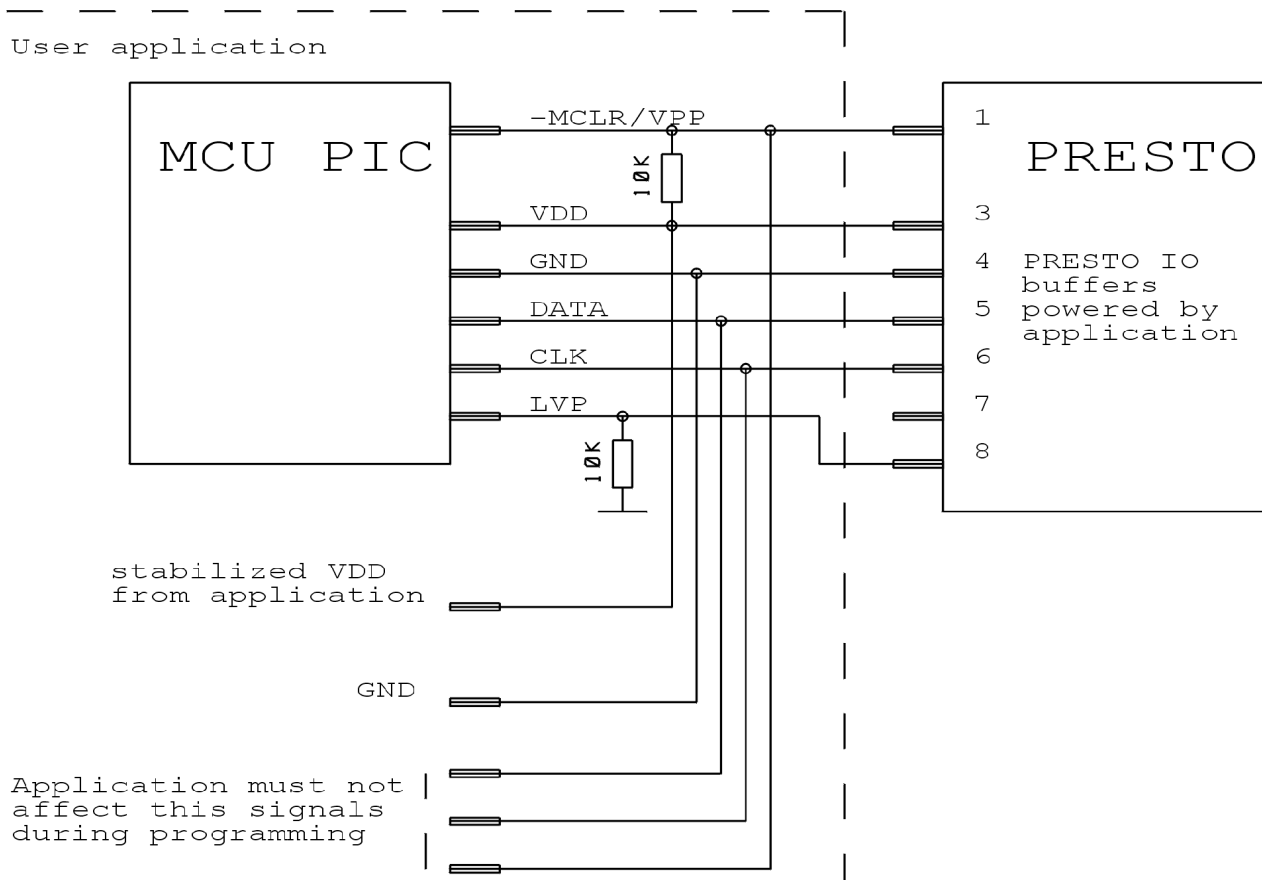
See notes to PIC microcontrollers and configuration word address.

Onboard PIC, using HVP (13V) mode, powered from application
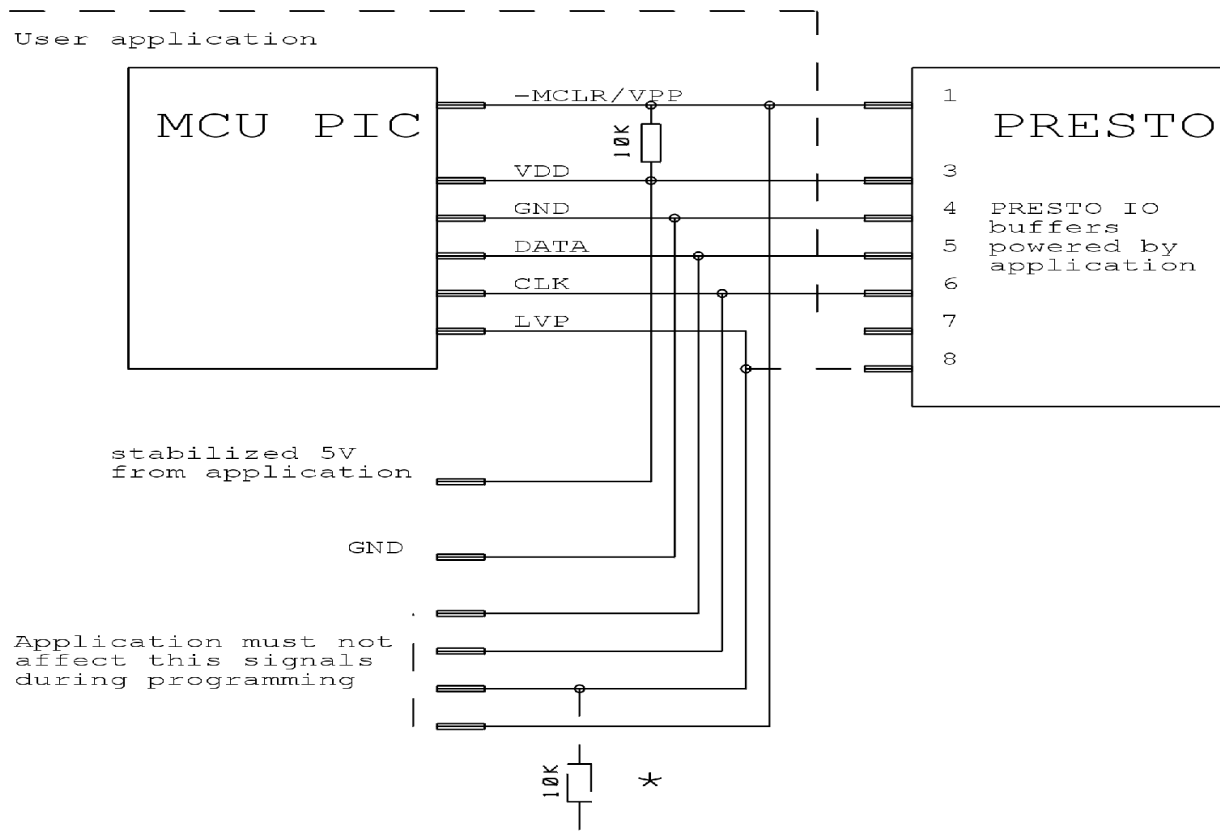
* Connect either to PRESTO or pull-down during programming

See notes to PIC microcontrollers and configuration word address.

Onboard PIC, using LVP (PGM) mode (not 13V), powered from PRESTO
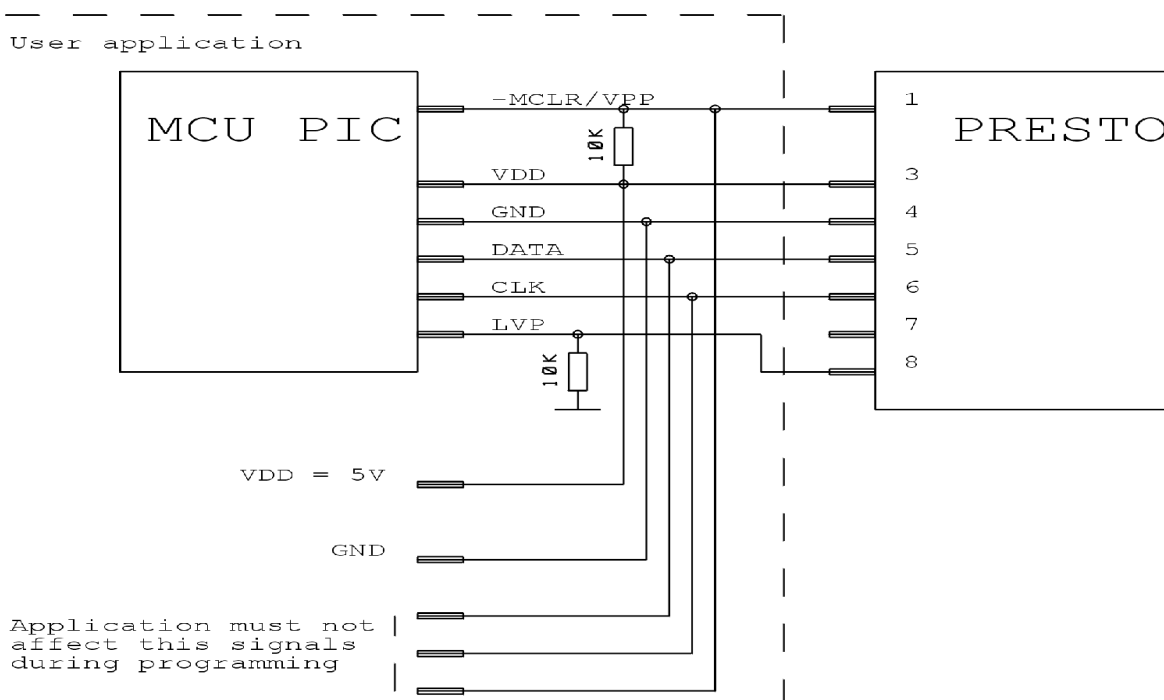
See notes to PIC microcontrollers and configuration word address.

Onboard PIC, using HVP (13V) mode, powered from PRESTO
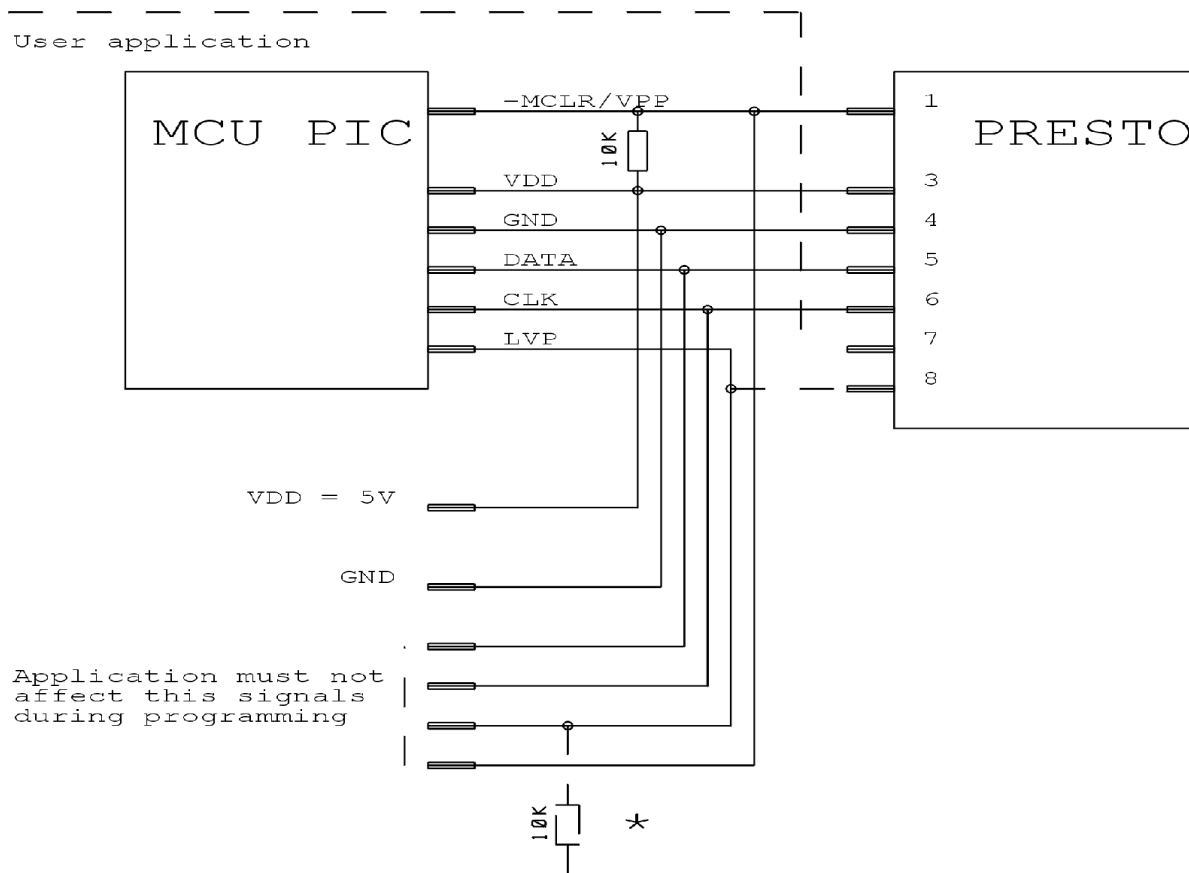
* Connect either to PRESTO or pull-down during programming

See configuration word address.

Notes:

- When the programmed device is PIC18FxxJxx microcontroller, on the MCLR pin will never be 13V, but only 5V level.
- If the firmware or data are protected using CP or CPD, the device must be always entirely erased before each programming.
- The device is not erasable when CP or CPD with supply voltage lower than 5V (for example 3.3V).
- If the processor has more than one VDD and GND pins, all of them must be connected, including AVCC and AGND if processor has some.
- It is necessary to power the 3V target externally, PRESTO is capable to supply the target with 5V.
- Some devices require less maximal voltage than 13V on MCLR pin, but PRESTO will provide 13V. When such device is chosen in the UP, the software gives a notice about it to user. User can limit the voltage by a two resistor divider or using a Zener diode and current limiting resistor.
- During erasing of microcontroller in HVP mode a LVP fuse can be erased too. To program the device in LVP mode the user must set the LVP fuse in HVP mode again.
- PIC32MX devices can be programmed via ICSP interface with external 3V supply voltage.
- Devices with ICPORT fuse must have the dedicated ICSP port disabled for LVP programming.
- With PIC24 and dsPIC33 it's possible to choose programming method using PE CheckBox. The PE means "Programming Executive", this method is usually faster then common method of the programming.
- If the PRESTO programmer is used with some new PIC devices programmed in the HVP mode, an over current error message indicating over current on the programming voltage can appear. If there is not serious mistake in the chip connection, it can be caused by Microchip new production technology used. The newly produced chips behave different way than the older chips, even if they are members of older chip families. The solution of the problem is to connect a 1 nF capacitor between the VPP and GND signals. In the case that the problem still appears, it is possible to add serial 10 Ω resistor to the VPP signal.

Onboard eCOG, always powered from application (VDD=3.3V)

```
User application

eCOG1                        27  EICE_CS            *          |        1    PRESTO

                                 VDD                           3
                                 GND                           4    PRESTO IO
                             24  EICE_MOSI                      5    buffers
                             25  EICE_CLOCK                     6    powered by
                             23  EICE_MISO                      7    application
                             26  EICE_LOAD                      8

   28    29    30    31        VDD  1K
                               100K
                               100K
                               100K
32.768kHz  5.0MHz            GND
```
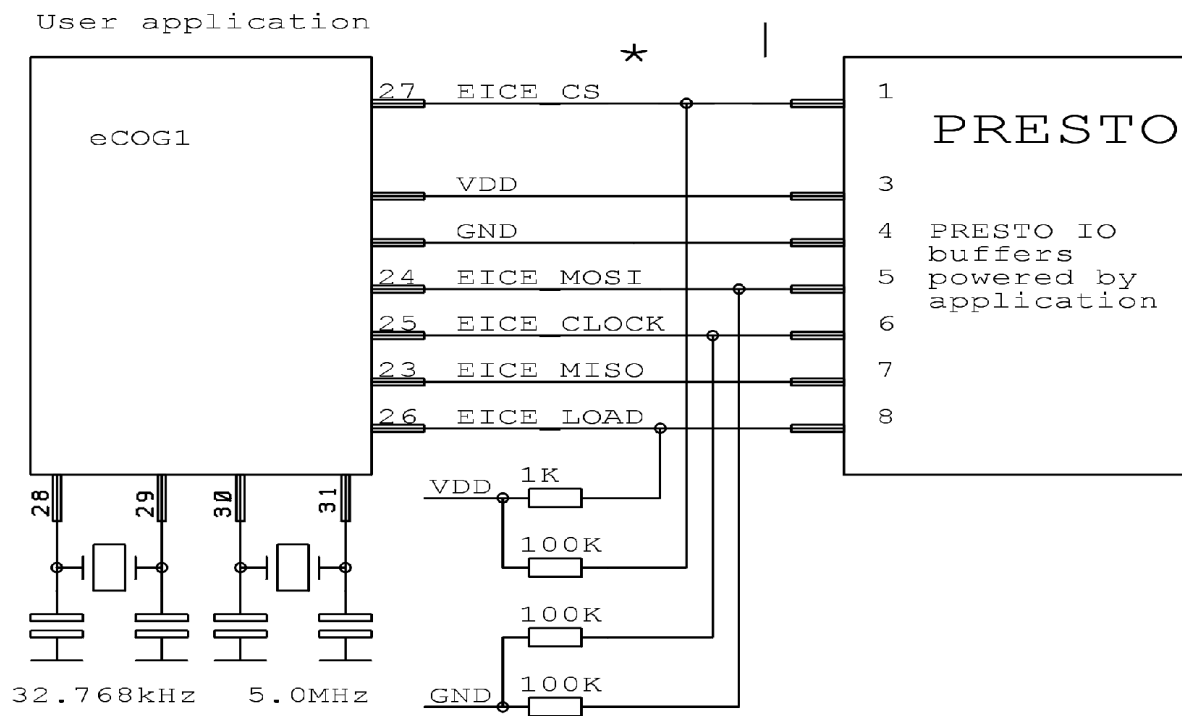
* Connection of EICE_CS to PRESTO is not needed. Stays in log1 during programming.

Notes:
- VCC must be 3.3V, the power must be supplied always from the application.
- To program an eCOG processor an application PreCOG can be downloaded at http://tools.asix.net/ free of charge. It is also available on supplied CD-ROM.

- See chapter about PreCOG.

Onboard AVR, powered from application
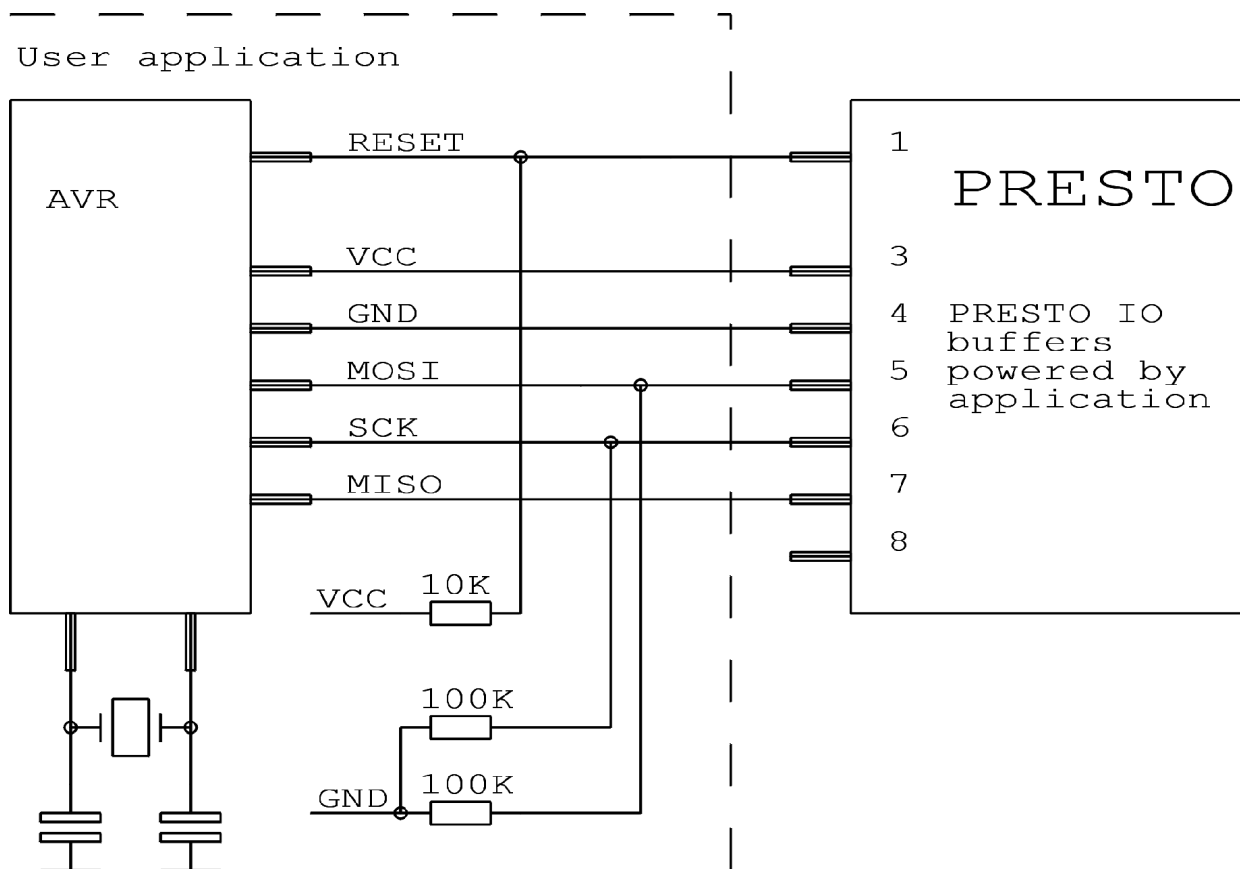
Notes:

● Fuses are by default (from manufacturer) set to the internal oscillator in ATmega and ATtiny microcontrollers. For the first time it is necessary to program the device with "Oscillator frequency" ">750kHz" or less set in the PRESTO programmer settings" window. External crystal is necessary only if you change the fuses of the microcontroller setings to external oscillator during programming.

● Is is not possible to connect the external crystal to every AVR microcontroller (e.g. Attiny13, Attiny15).

● If the fuses of the programmed AVR device are correctly set, it is necessary to click by right button to the **configuration** window and choose **learn fuses** item in the menu. The fuses settings will be saved to the project file so that if a .hex file will be opened next time, the fuses will be set equally. (If the device is programmed using command-line invocation of UP, user has to specify a .ppr file instead of a .hex file to save the fuses.)

● It is possible to mark item **Open .hex file with data memory automatically** in the **File** menu. This command causes that data for the EEPROM memory are loaded in the same time with the program file.

● If it is necessary not to change area of the EEPROM memory in the microcontroller, use EESAVE fuse. If this fuse is in an active state, use **Program all except of data EEPROM** command, else the UP program will report that EEPROM memory is not erased.

● There is useful choice "**No data memory erase before its programming**" in **Options→Program settings...→Programming** menu.

● For 3.3V devices it's possible to use internal supply with HPR3V3 converter.

● For conversion of the ICSP connector of the PRESTO to the Atmel's 10pin ISP connector may be used **HPRAVR** adapter.

● Some AVR devices have ISP interface situated on different pins than the SPI interface. See "Serial downloading" section of the device datasheet.

Atmel AVR with TPI interface (e.g. ATtiny10)
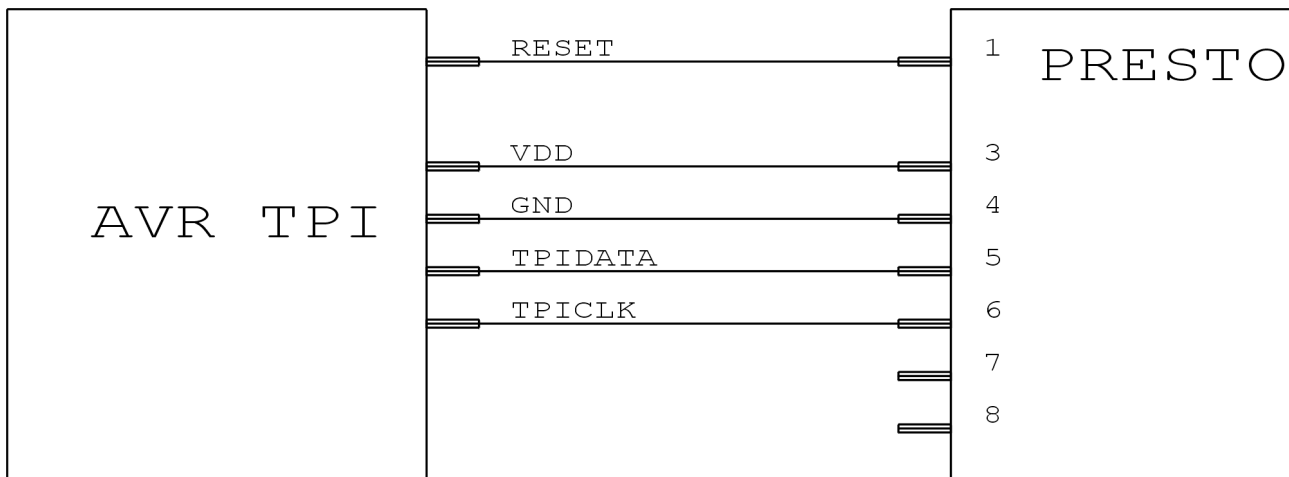
Notes:

- The chip requires 12 V to be connected on its RESET pin, during HVP programming. The programmer can supply it with 13 V only, so there must be connected an external circuit reducing this voltage. The chip needs no additional external components, if it is programmed with standard low voltage method.

Onboard Atmel 8051 microcontroller powered from application

User application

8051

RESET
VCC
GND
MOSI
SCK
MISO
SS *
GND   10K
100K
GND   100K

PRESTO

1
3
4
5
6
7
8

PRESTO IO
buffers
powered by
application
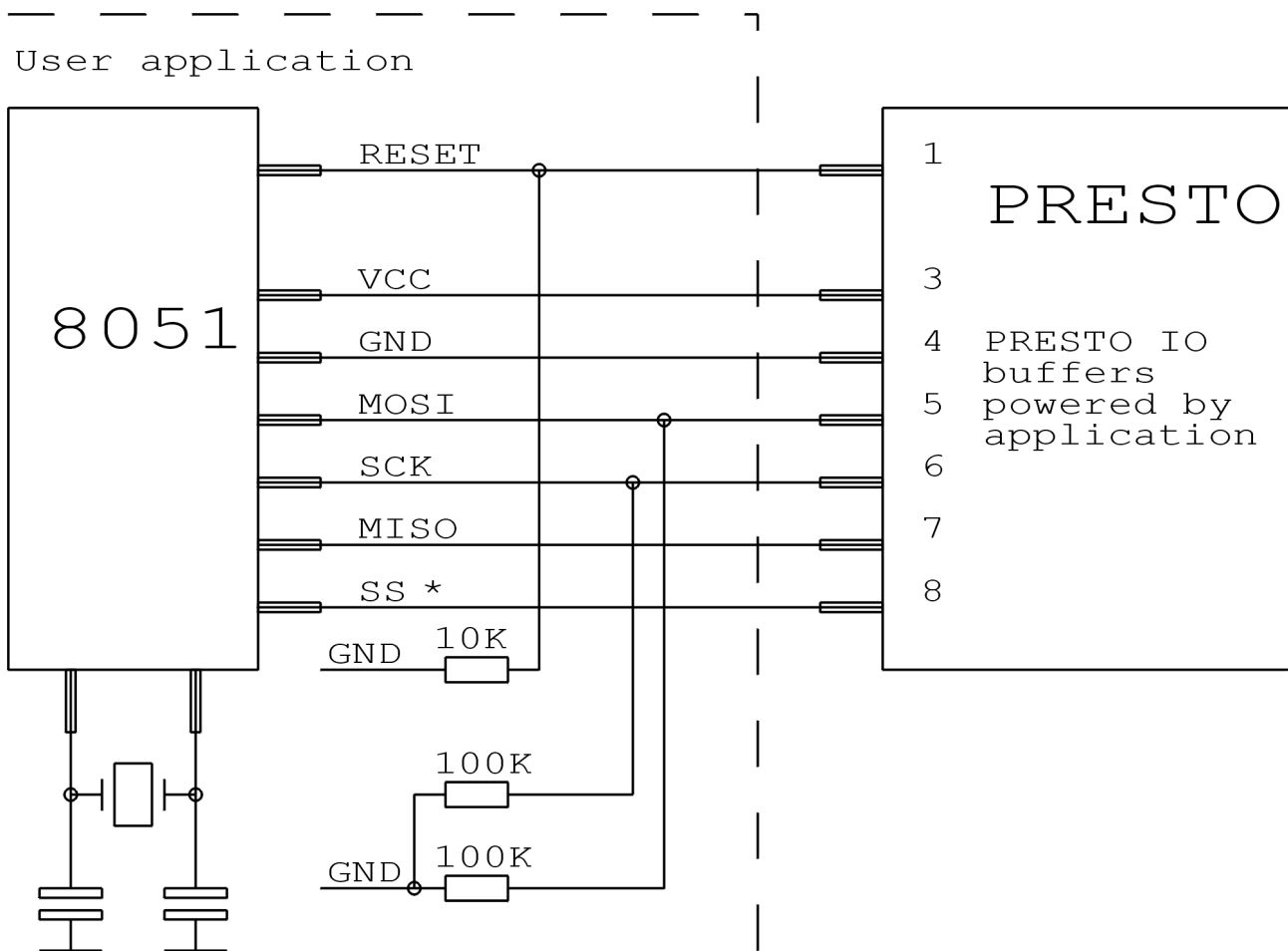
* The SS pin must be connected only for AT89LP2052/4052/213/214/216/428/828/6440/51RD2.

Notes:

● AT89LP213, AT89LP214 and AT89LP216 use inverse reset logic. The resistor on the RESET pin must be connected to VCC and not to GND.

● The parts containing "C" in the name cannot be programmed by the PRESTO programmer, but the parts with "S" in the name are supported and some of them are compatible with the "C" types. For example the AT89C2051 isn't supported, but the AT89S2051 is supported.

● Software supposes that the POL pin of the chip is connected to log. 1 during AT89LP52 programming. In case that the POL pin is connected to log. 0, the "Inverse RESET" CheckBox must be checked off. AT89LP51RD2 has inverze logic of the reset, software supposes the POL pin to be connected to log. 0.

Connection of a PSoC device by Cypress
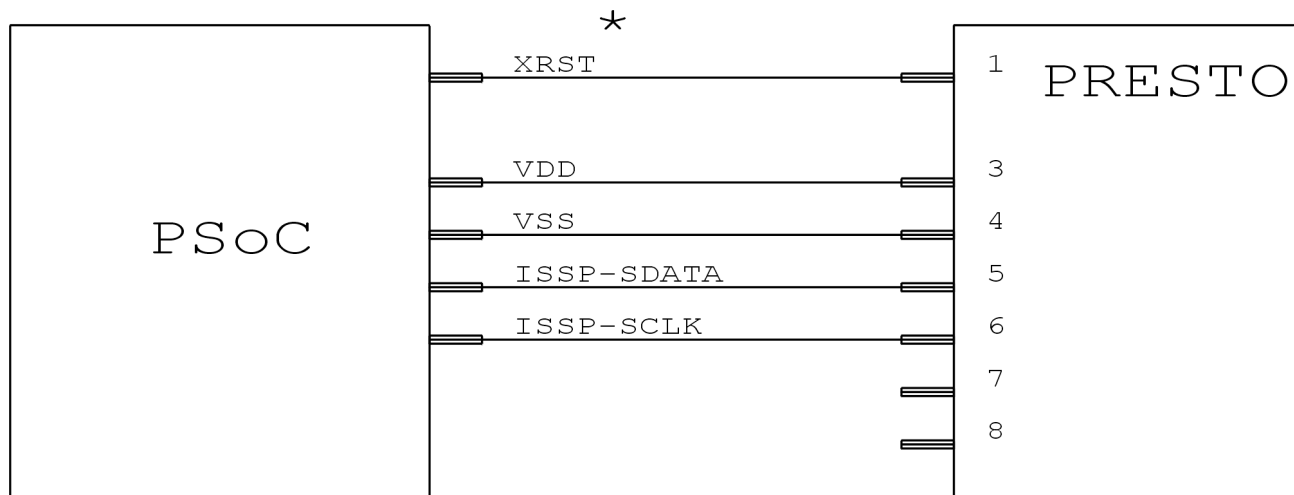
* Not every PSoC device has the XRST pin.

Note:

● User should first select the programming mode initialization method in the „PRESTO programmer settings" window. The devices without the XRST pin can use power-cycling method only. The devices with XRST pin can use both methods, but the method what initializes using reset signal is better, because it can be used also with external power supply.

● The "Programming algorithm" in the "PRESTO programmer settings" window should be set in accordance with the supply voltage used during programming.

Connection of a MSP430 device what has not the SBW (two wires) interface
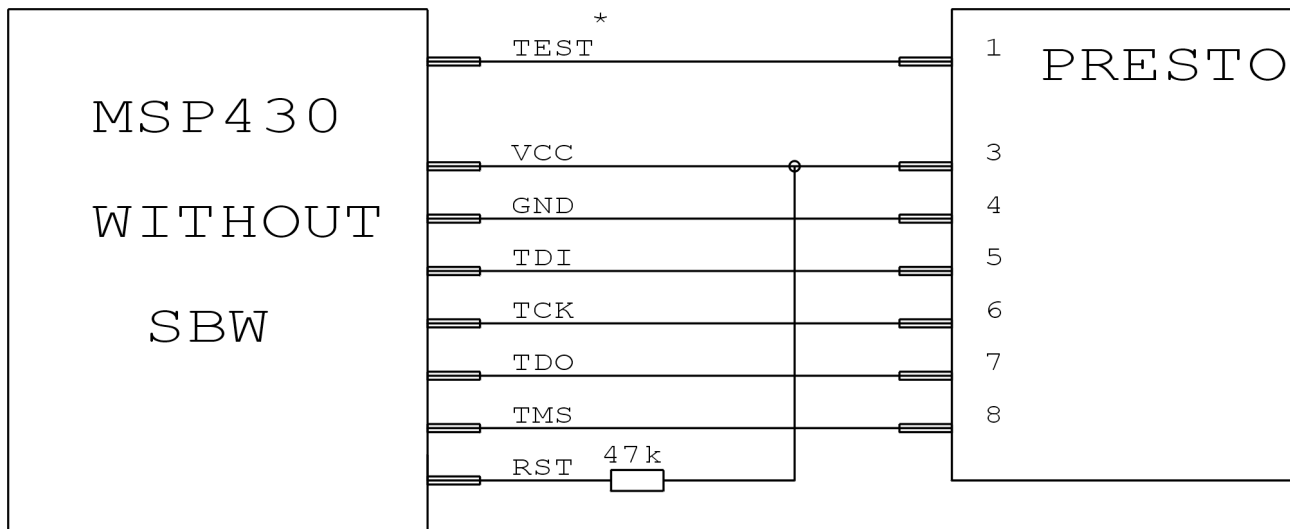
* Not every MSP430 device has the TEST pin.

Notes:

- This interface connection diagram can be used for example with MSP430F1xx, MSP430F4xx, MSP430F21x1 but not with MSP430F20xx or MSP430F22xx.

- If the device has calibration values at the information memory and this memory won't be reprogrammed (erased) during programming, it should be programmed with chosen "**Calibrated internal RC**" oscillator in the "**PRESTO programmer settings**" window. In the other cases it should be "**Not calibrated internal RC**" oscillator.

- That's not possible to blow security fuse of the MSP430 processors using PRESTO programmer.

- For programming of alone MSP430 processors with this interface (non SBW) it's possible to use internal power supply with HPR3V3 converter.

Connection of CC430 or MSP430 device what has the SBW (two wires) interface
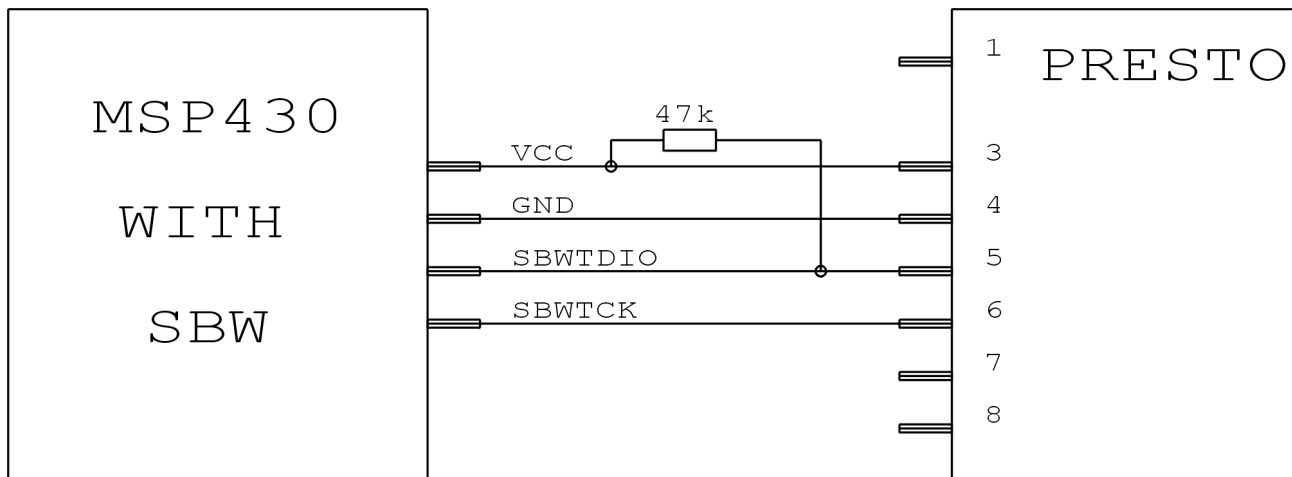
Notes:

- The device what has the SBW interface can be programmed using this interface only. It's for example MSP430F20xx, MSP430F22xx or MSP430F5xxx.

- If the device has calibration values at the information memory and this memory won't be reprogrammed (erased) during programming, it should be programmed with chosen "**Calibrated internal RC**" oscillator in the "**PRESTO programmer settings**" window. In the other cases it should be "**Not calibrated internal RC**" oscillator. The oscillator type cannot be set for MSP430F5xxx and CC430 devices.

- That's not possible to blow security fuse of the MSP430 processors using PRESTO programmer.

- Speed ComboBox in the "PRESTO programmer settings" window is useful when there is some capacitor connected to the RESET pin.

- The chips with calibration constants saved in the information memory can have their information memory erased with or without Segment A. This can be set using "Erase Segment A" setting.

Connection of CCxxxx device by TI (Chipcon)
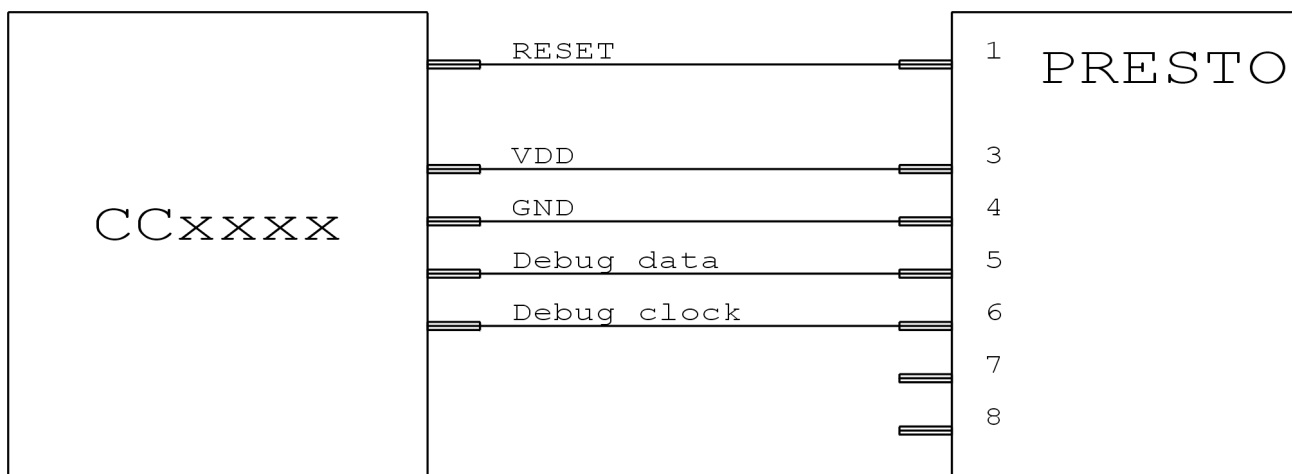
Connection of I²C memory to PRESTO
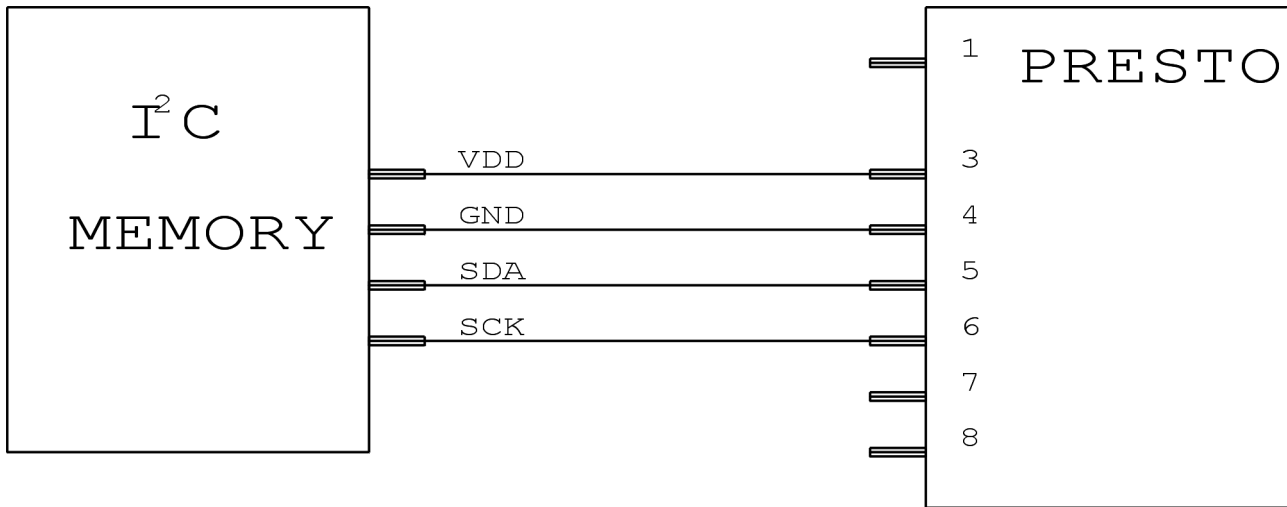
Notes:

● The programmer is using internal pull-up resistor of 2.2kΩ on the data signal (SDA) when working with I²C bus.

● If the programmed device is 24LC(S)21A or 24LC(S)22A, its VCLK pin must be connected to VDD during programming.

● The 34xx02 needs high voltage for the SWP and CSWP write protection commands on the A0 pin. The high voltage is generated on the VPP pin of the programmer. The high voltage from the programmer is 13V, but the chip high voltage should be less than 10V. User must lower the voltage himself. The A0, A1 and A2 pins of the memory must be set manually in accordance with the selected protection mode.

Connection of SPI memory to PRESTO
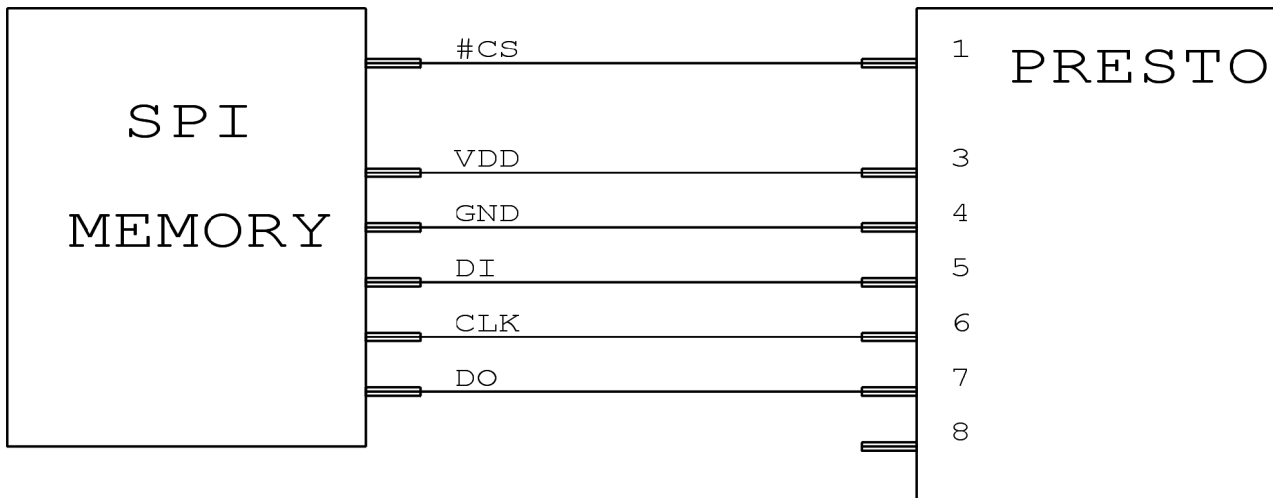
Various manufacturers mark the SPI memory pins with different names. Some of them can be found in the table:

Name in picture	Atmel, SST	ST
DI	SI	D
DO	SO	Q
CLK	SCK	C
CS	CS, CE	S

Notes:

● **Programming SPI memory onboard** - The program enable and / or Hold pins can be parked at proper logic level in application. The programmer needs that all pins during programming are digital inputs or just isolated from other circuit by some multiplexer. Programmer clocks these memories at approx 500-1000kHz (to satisfy timings of "AA" devices), so capacitances of data lines must be slow enough to allow this speed.

● For 3.3V devices it's possible to use internal supply with HPR3V3 converter.

Connection of microwire memory to PRESTO
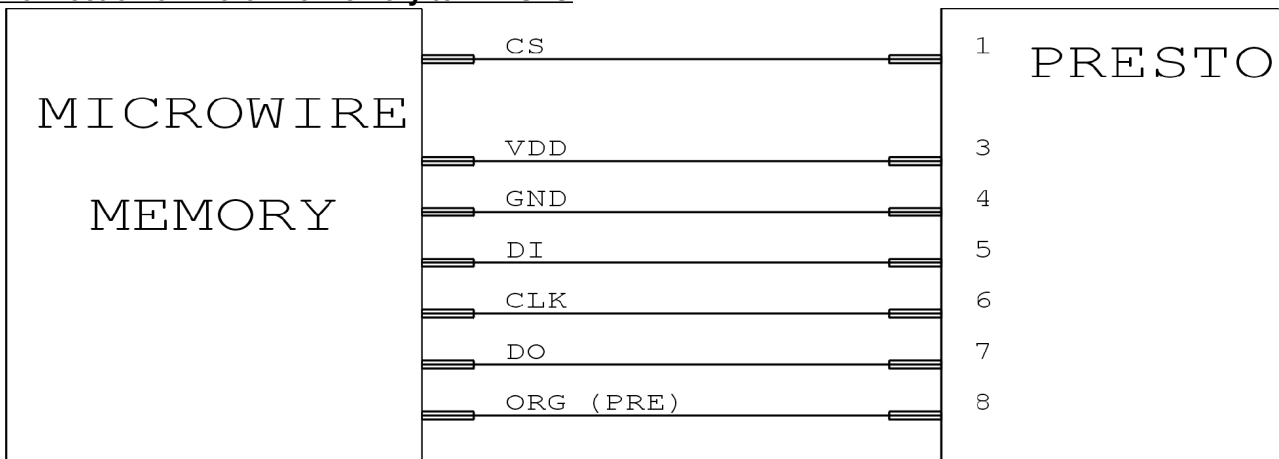
Notes:

● The **LVP** pin (Pin8) selects organisation of memory in either 8-bit 16-bit words. The user selects desired organisation in the software and PRESTO then sets this pin to corresponding logical value. If the memory organisation is hard-wired in the application circuitry or it is not possible to choose it the LVP pin of the programmer will be left unconnected. If the programmed device is M93Sx6 then the LVP pin must be interconnected with the PRE pin of the chip.

Connection of a device programmed over the JTAG interface to PRESTO
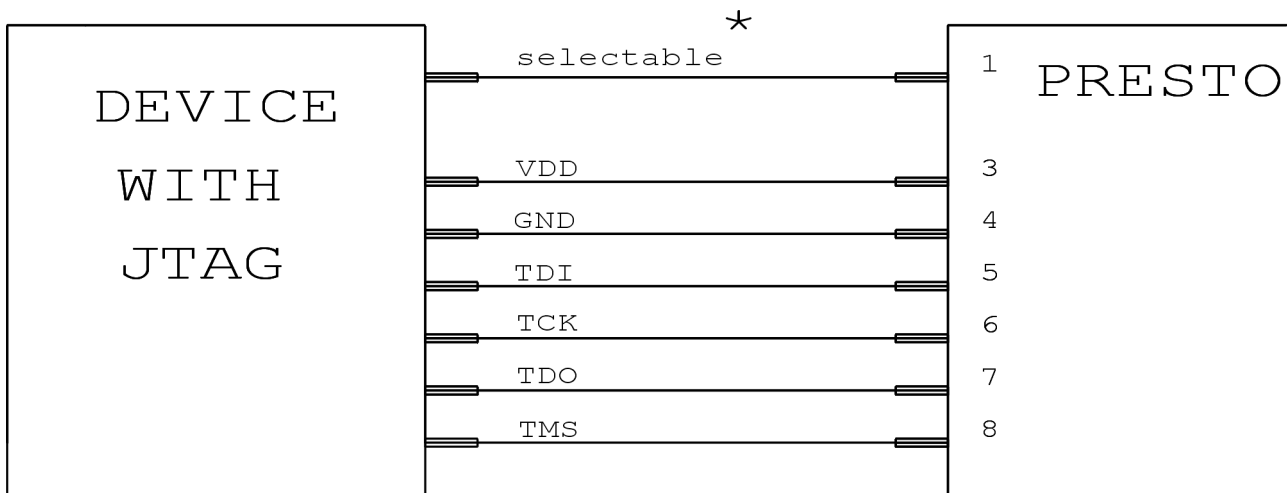
* The VPP signal may act as SCK or TRST described in SVF file or may be in user defined state (log.1, log.0, tristate, different during the programming and after it)

The P1 of the programmer should be connected to the RESET pin when programming AVR32 devices.

Notes:

- The power supply is always taken from the application for JTAG applications often use voltage other than 5V.
- It is possible to program and test parts for which there is a software providing data in SVF or XSVF format. The XSVF format is recommended for CPLD Xilinx XC9500 only, while SVF is recommended for all other parts. The software for using JTAG with PRESTO is available for download at http://tools.asix.net/ and on supplied CD-ROM.
- PRESTO in JTAG mode is not providing voltage to application.
- When the JTAG pins are used in application as I/O pins, processor must be held in Reset during programming. It can be done using yellow (VPP) pin of PRESTO, which can be programmed (in JTAG player options) to do this.
- The JTAG interface is used for programming and debugging of processors with ARM core too. The ARM processors are programmed with **ARMINE** software available on the web. For more information on the ARM devices programming see the ARMINE help.
- The HPR1V2 converter should be used with the PRESTO programmer for programming of devices with less supply voltage than 2.7V.
- The AVR32 MCUs are programmed via JTAG interface with the UP program. The part mustn't be in the reset state during programming.
- The ATxmega MCUs are programmed via JTAG interface with the UP program. The P1 pin isn't required for the programming.The ATxmega devices can be used with the HPR3V3 converter.

1.6 Description of indicators and controls

Green LED (ON-LINE) - PRESTO is connected to the computer
Yellow LED (ACTIVE) - Communication is in progress
Button (GO)   - Triggers the programming process

1.7 Technical specifications

Warning: **Violation of these parameters may cause damage of the programmer or connected computer!**

Maximal Vcc voltage	$U_{VCC\,MAX}$	7.5 V
Maximal voltage for other pins	$U_{IO\,MAX}$	5.5 V
Maximal current drawn from V_{CC}	$I_{VCC\,MAX}$	100 mA
Maximal current drawn from V_{PP}	$I_{VPP\,MAX}$	50 mA
Maximal current drawn from other pins	$I_{IO\,MAX}$	4 mA
Voltage if powered from application	$U_{VCC\,IN}$	3.0 V to 5.0 V $\pm10\%$
Voltage on V_{PP} during programming	U_{VPP}	5 V/13 V
Operating temperature	T_{OP}	0 to 40 °C

Dimensions L x W x H	Approx. 105 x 55 x 25 mm.
Maximal allowed length of the ICSP cable	150mm

2 Other programmers

2.1 PICCOLO

Description of the programmer

PICCOLO is a low cost development programmer for 18-pin Microchip PIC® microcontrollers which are equipped by flash memory. Parts in larger packages (28 and 40) can be programmed using ICSP cable.

PICCOLO perfectly suits beginners, students and amateurs. The programmer conforms to Microchip programming specification (development category).

Supported parts

• All recent PIC® microcontrollers equipped by flash memory, directly in the programmer 18-pin packages only.

2.2 PICQUICK

Description of the programmer

PICQUICK is fast, reliable, yet cheap development tool for PIC® microcontrollers and serial EEPROM memories (Microchip). On of the greatest advantages of this programmer is support for all types of PIC microcontrollers.

Choose any part suitable for your application and you can be sure that PICQUICK supports it. Moreover there id support for Microchip EEPROM memories (I2C and MicroWire). These memories are often used to extend internal memory of the microcontroller. PICQUICK is designed to be flexible to support future parts. Support for practically all new parts which came to the market after PICQUICK was added just by upgrading the software.

Hence there is no need to upgrade firmware nor any hardware changes are required which brings convenience and financial benefits to the user. The software upgrade is available for free to all users. It contains support for new parts, new features and possible changes to programming algorithms, if required by Microchip.

PICQUICK directly supports In-Circuit Serial Programming (ICSP) by dedicated connector and cable, which is included in the package. Current limiting circuitry for both power supply and programming voltage minimizes risk of damage to the part when handled incorrectly.

Supported parts

- all recent PIC® microcontrollers
- serial memories 93Cxx and 24Cxx

2.3 CAPR-PI

Supported parts

- All recent PIC® microcontrollers with flash memory, which do not feature MCLR/Vpp and I/O function on a single pin.
  The restriction applies to e.g. PIC16F627/628.

2.4 PICCOLO Grande

Description of the programmer

PICCOLO GRANDE is a low cost development programmer for 18, 28 and 40-pin Microchip PIC® microcontrollers which are equipped by flash memory. It is possible to program standalone parts using a ZIF socket as well as parts already soldered in an application circuit using ICSP cable (In-Circuit Serial Programming).

PICCOLO perfectly suits beginners, students and amateurs needs. The programmer conforms to Microchip programming specification (development category).

Supported parts

- All recent PIC® microcontrollers equipped by flash memory. Parts in 18, 28 and 40-pin packages can be programmed directly in the programmer.

2.5 PVK Pro

Description of the programmer

PVKPro is development and educational kit with built-in programmer for PIC16F84A on a single board. It is designated for study and educational purposes and helps to get familiar with PIC mirocontrollers, realtime processing, I/O, display multiplexing, button scanning and more.

The board contains all the essentials for typical microcontroller applications.

- power supply circuitry
- oscillator
- reset circuitry
- 4-digit 7-segment LED display
- 8 LEDs
- 8 pushbuttons

All user pins can be configured by DIP switches to be connected to on board peripherals or to be used for external

circuitry. The board connects to the PC using parallel port.

Supported parts

- PIC16F83 / 84 / 84A
- PIC16F627 / 628 with limitations
  The programmer PVK-Pro cannot comply with programming specifications for parts which feature I/O function on the MCLR pin.

PIC16F627/628 microcontrollers must not be programmed or tested in PVK-Pro with their MCLR pin configured as I/O. Doing so could cause damage to the part or the programmer.

Using the -MCLR pin as I/O ns PVK-Pro does not make any sense anyway thus the inability to program such parts is rather not much limiting.

3.1 HPR3V3

HPR3V3 is an optional accessory to the PRESTO programmer for programming of stand-alone 3.3 V devices as for example DataFlash Memories. The PRESTO programmer can supply a programmed device with 5 V. However, some devices require 3.3 V supply voltage and 3.3 V logic levels on their pins. In such case, the HPR3V3 level shifter or an external 3.3 V power supply have to be used.

Usage

The HPR3V3 level shifter can be used very simply. Plug it directly to the pins of the PRESTO programmer (**NOT to the programmed device connector!**). Pin 2 is used as a key, so that is not possible to connect it wrong way. Now connect the programmed device to HPR3V3 using ICSP cable. Connection of the programmed device pins is same as connection of the device to the PRESTO. See wiring examples for PRESTO. Common connection diagram is below.

Notes:

- The HPR3V3 pins are unidirectional, so this level shifter can be used for AVR devices, SPI Flash memories or MSP430 devices without SBW interface, but it's not intended for PIC devices and MSP430 devices with SBW interface.

- Never connect external voltage to the output **3.3 V supply pins!**
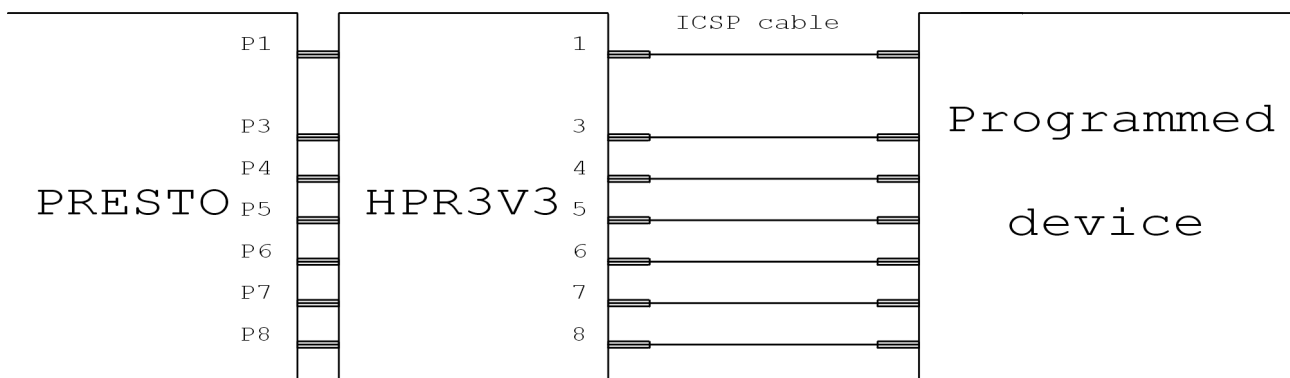
3.2 <u>HPR1V2</u>

HPR1V2 is an optional accessory to the PRESTO programmer for programming of the devices with supply voltage and logical levels between 1.2 and 3.3 V as for example Xilinx CoolRunner-II. The PRESTO programmer can program devices with signal levels between 3 and 5 V ± 10 %, but sometimes it's necessary to program a device with less signal levels. In such case, the HPR1V2 level shifter must be used. The level shifter must be used with external supply voltage from application, it cannot be supplied from the programmer.

Usage

Plug the HPR1V2 level shifter directly to the pins of the PRESTO programmer (**NOT to the programmed device connector!**). Pin2 is used as a key, so that is not possible to connect it wrong way. Now connect the programmed device to HPR1V2 using ICSP cable. Connection of the programmed device pins is same as connection of the device to the PRESTO. Common connection diagram is below.

Notes:

● The HPR1V2 pins are unidirectional, this level shifter can be used for example with JTAG devices, but it's not intended for devices what need bidirectional pins like PIC processors.

● The HPR1V2 must be supplied with external supply voltage from application.

● Never turn on the internal supply voltage from programmer, when the HPR1V2 is connected**!**
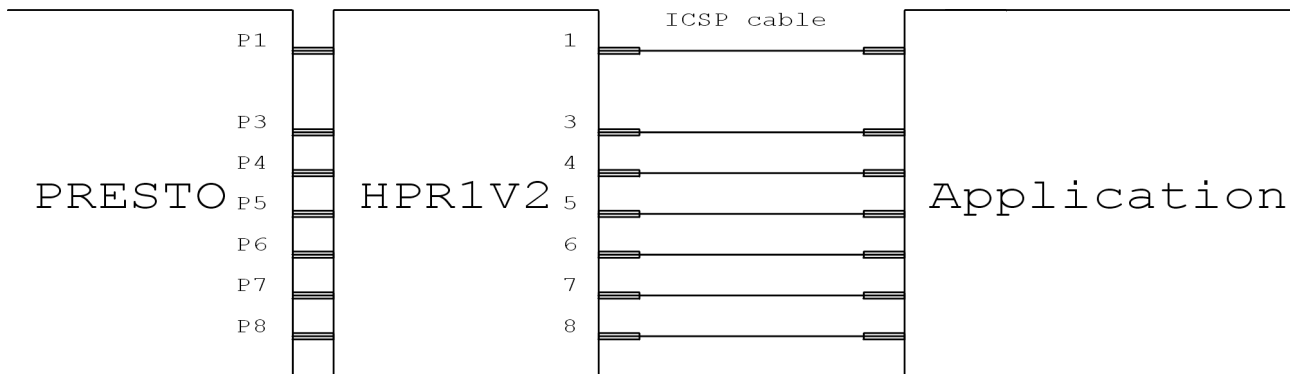
4 <u>Software UP</u>

UP is the software for ASIX programmers. It offers many advanced features and allows detailed user control of device programming process - either interactively or "remotely" controlled using command line, Windows messages and DLL library. It runs under Windows 95/98/ME/NT/2K/XP.

4.1 Installation of UP

The installation procedure is fairly simple: Get the installation package either from supplied CD-ROM or http://tools.asix.net/ (UP_xxx_EN.EXE, substitute xxx by version number) and run it (there is not necessary to close other running applications). The installation takes a few seconds and requires just to press Enter several times. No modification of operating system is performed during the installation and thus it is not necessary to reboot the machine and the application may be started instantly (e.g. by clicking on corresponding icon). Upon first launch the application prompts for language selection (English/Czech), type of the programmer (e.g. PRESTO) and a port which the

programmer connects to (USB in case of PRESTO).

Application removal can be performed in common fashion using the control panel applet or manually by deletion of corresponding directory and shortcuts. There is no necessity to remove previous installation before installing new version of the software. The user settings are preserved when installing new version over previous one. It is recommended to use always the most recent version of the software.

4.2 Device programming

It is recommended to use projects in UP. Project file contains all settings.

New project can be made by clicking **File→New Project** item in the menu, existing project can be opened by clicking **File→Open Project** item.

Before programming it is necessary to choose used programmer and a programmed device type. It can be done by selecting menu items **Options→Select programmer** and **Device→Select device** or by double-clicking on chosen programmer and device name, which are shown in the right up corner. It is useful to change settings of program to customize it to users requirements. Use the **Options→Program settings** menu item to do it. Detailed description can be found in Application menu overview section.

Any time, if PRESTO programmer is chosen, it is shown PRESTO programmer settings window, where should voltage source and some other important settings be configured.

Programming : Use **File→Open** menu item to open a .hex file.. If it is necessary to change fuses settings to modify the programmed device features, it can be done in the **Configuration** window. Changes can be saved by clicking the **File→Save** menu item. (See notes to AVR and PIC programming.)

Programming will start after clicking **Device→Program** menu item or after clicking the **Program** button. The programmer do following operations: Erases, blank checks, programs and verifies whole part. Before taking the action a check on Device ID, and Code/Data Protection is performed.

If it is necessary to program only a part of the device, it can be done by selecting of relevant item in the **Device→Program** menu or by selecting of item under arrow near the **Program** button. Detailed description can be found in Application menu overview section.

Note : If the setting "**load hex file before programming**" in programming options is selected, software reloads the hex file after pressing **program** button. When fuses are not stored in the hex file user must either deselect the option for hex file reloading or select in file loading options not to **erase configuration memory before reading file** (this means fuses which are not exactly stored in hex file are turned to default state).

4.3 Selecting of GO button function

The PRESTO programmer contains GO button. GO button allows user to start device programming comfortably, without need of PC keyboard or mouse. LEDs indicate programmer status - green LED (ON-LINE*)* informs about USB connection and yellow LED (ACTIVE*)* indicates that PRESTO is just working (programming, reading, ...)

Function of the button can be selected according to user's needs in **Device→Key shortcuts** menu under the **Button GO** item.

UP must be running if user wants to use the Go button, but it can be minimized.

4.4 Mass production mode

Menu: **Device→Program→Mass production**

The function is available also from tool panel under "Program" icon.

In the mass production dialog the programming can be started by clicking "Program" button, which is equivalent to "Program All" or "Program All but EEPROM" depending state of "No EEPROM programming" option.

Mass production counter, is also available in this dialog. According to Options it can be also displayed in the status panel. The counter displays total number of programmed parts in both mass production and standard mode.

4.5 Serial numbers

The "**Serial numbers**" function programs serial number or other data sequence into given memory area before or after the programming.

Serial numbers may be:

- *automatic*
  Automatic numbers are always programmed to a single memory location of the device e.g. code memory, EEPROM memory or ID locations. The serial number can be either decimal or hexadecimal and may be encoded either as 4-bit combinations (up to 4 to a single word) or ASCII character (up to 2 to a single word). If code memory is used to store serial number it may be optionally enclosed in RETLW instruction.
- *read from a file*
  In this case a number may be split into several memory areas (e.g. serial number itself in code memory, device address in the EEPROM and, the same serial number in the ID locations to allow reading it even from a protected part).

Note: By one word is meant one position of memory.

In "**Options/Program settings.../Serial numbers**" a file for the serial numbers logging can be chosen. In the case of the automatic (counted by the UP) serial number the serial number itself is logged. In the case of the serial number read from a file the serial number label is logged.

Serial number file format

The serial number file is a text file which can be easily generated by other program. Recommended file extension is .SN or .TXT.

- *White space* is a space, tab or end-of-line (CR+LF).
- *Comment* is any line beginning with semicolon. '**;**'
- *Serial number record* is a text in form
  *label***:** data record, data record, ..., data record; *Comment*
- *Label* is a string identifying serial number. Label is **mandatory**.
- *Data record* is compound of address (hexadecimal) and data fields (hexadecimal) to be programmed to the memory starting at given address,
  e.g. 2100 05 55 54 causes the EEPROM to contain 05h, 55h, 54h at addresses 00 to 02.

  It can be specified memory where the serial number will be saved using word CODE. or PROG. or P. for program memory or DATA. or EE. or E. for data memory or ID. or I. for ID location. These words are followed by address in the specified memory.
  e.g. `EE.00 05 55 54` causes the EEPROM to contain 05h, 55h, 54h at addresses 00 to 02.
  Notes:
  > There is no specifier for fuses, it does not make much sense to place serial number there.

  > For dsPIC parts the CODE address is the address of the 24 bit word, i.e. the internal dsPIC address divided by 2. In case of EEPROM, specify address of 16-bit word.

  > Standalone (I2C, SPI) memories feature just one address space which is treated as CODE.

- *Comment* following serial number record is optional.

Example of serial number file:

; comment at the beginning

sn1:	0000	34 45 56 67,		
	2100	01 02 03 04;	this is serial number 1	
sn2:	0000	45 56 67 78,	2100	02 02 03 04;
sn3:	0000	56 67 78 89,	2100	03 02 03 04;

; a note

sn4:	0000	67 78 89 9A,	2100	04 02 03 04;
sn5:	0000	78 89 9A AB,	2100	05 02 03 04;
sn6:	0000	78 89 9A AB,	2100	06 02 03 04;
sn7:	0000	78 89 9A AB,	2100	07 02 03 04;

sn8:   code.0001 3F00 3F01 3F02 3F03, data.0002 'x' '4' '2';
sn9:   prog.0001 3F00 3F01 3F02 3F03, e.0002 'x' '4' '3';

4.6 Using UP from the command line

The program itself ensures to be run in a single instance. If there is another instance being run, the command line parameters are handed over to the already running instance to be executed.

Parameter overview

up.exe [{**/ask** | **/q**}] [{**/e** *eeprom_file.hex* | [**/noe**]}] [{**/p** | [**/pdiff**] | [**/o**]} *file.hex* | *file.ppr*] [**/part** part_name] [**/erase**] [**/w**[nd] *up_window_class*] [**/cfg**] [**/devid**] [**/blank**] [**/verify** *file*] [**/s** *programmer_SN*]

Legend:
The text in **bold** is to be put on the command line as is.
The text in *italic* is to be replaced by real parameter, e.g. *file_name* is to be replaced by real name of the file to be opened.

Text in curly brackets separated by | (pipes) represents a single choice from the listed items. e.g. { **A** | **B** } means "choose just one of A or B".
Text in [braces] represents optional parameter which can be used, but can also be omitted.

- **/ask** Ask. To be used with **/p**. If the parameter is used, the program always prompts the user whether to program the part, even if this was disabled in the Settings of the program. The confirmation dialog also shows selected part type.
- **/q /quiet** Quiet mode. In this mode the program does not require any user intervention, but rather silently returns to an error code instead of displaying a dialog. See Return error codes.
- **/e file** EEPROM file. Name of a file containing EEPROM data. If the name contains spaces, it is necessary to enclose it by quotes. This parameter can be used together with **/o** or **/p** parameter only.
- **/noe** No EEPROM. Causes the program to skip EEPROM programming (and all operations with EEPROM). If used with the MSP430 devices, the program skip all operations with the information memory.
- **/p file** Program. Programs given file to code memory. If the name contains spaces, it is necessary to enclose it by quotes. (device is erased, code, data, cfg (and ID) memory is programmed and verified.)
- **/pdiff file** - Program differentially. Programs given file. If the name contains spaces, it is necessary to enclose it by quotes.
- **/o file** Open. File with given name will be opened. Optional parameter. If the name contains spaces, it is necessary to enclose it by quotes.
- **/eeonly** - The programmer will do the selected operation for EEPROM memory only, with MSP430 for Information memory only.
- /**part name** Selects the specified part in the UP.
- /**erase** The part will be erased.

- **/wnd class name** Select another window class name. Using this parameter you can open more than one instance of program UP. Each instance must have unique window class name.
- **/cfg** If this parameter is used together with **/p** parameter, only configuration memory is programmed. It's useful for example for AVR devices programming, because the user can configure the chip for faster oscillator first and then to program it much faster.
- **/devid** If this parameter is used together with **/p** parameter, only the Device ID of the chip is checked.
- **/blank** Program will check if the chip is blank, it will return an error code in accordance with the result.
- **/verify file** Does the part verification.
- **/s programmer_SN** - This parameter allows to select the programmer in accordance with its serial number. The serial number can be entered as it is displayed in the UP or printed on the programmer, e.g.016709 or A6016709.

When working on several different projects it is likely that the program is set to use a part or programmer other than the one the user assumed. In such case using project files (`.PPR`), which contain all necessary settings including the file path, is strongly recommended.

Note: Examples, how to use UP from the command line can be found in the UP installation directory in files "read_avr_eeprom.bat" and "set_idle_power_1.bat" .

Opening a file

up.exe *file name*

up.exe file.hex
up.exe "C:\My Documents\Recent Projects\PIC\My latest project\flasher.hex"

Programming the part

```
up.exe /p file name
```

up.exe /p file.hex
up.exe /p "C:\My Documents\Recent Projects\PIC\My latest project\flasher.hex"

Return error codes

0 - No errors.
1 - File error, e.g. file not found, unrecognized file format.
2 - Device error. Communication test failed, error in communication.
3 - Programming preparation error, e.g. failed to erase part.
4 - Programming error.
5 - Verification failed.
6 - User interaction needed.
7 - Device ID error.

Note: In the batch files, you can get the resulting error code by syntax %errorlevel%. See example file "read_avr_eeprom.bat" in installation directory of UP.

4.7 Controlling UP utilizing Windows messages

UP may be also controlled using Windows messages. Running instance of UP executes desired action instantly upon reception of a message.
The messages ought to be sent to Window of "up v1.x" class. Type of the message is always WM_USER. Particular commands are distinguished by wParam, the parameters are taken from lParam.

Commands overview

```
(* Messages to UP
 * These messages should be sent to Window identifiable by its class "up v1.x"
 * Almost all messages responses 0=false=failed=can't; 1=true=done=OK=can
 * WM_USER:
 *    wParam = 0   lParam = 0; does anything, only returns 1
 *            lParam = 1; SetForegroundWindow()
 *            lParam = 2; maximizes and SetForegroundWindow()
 *    wParam = 1   lParam = any; Does programming all contents of file; Result is same as on command line
 *    wParam = 2   lParam = any; Does programming without eeprom;      Result is same as on command line
 *    wParam = 3   Does programming (with erase)              Result is same as on command line
 *            lParam |= 1; of main code memory
 *            lParam |= 2; of data eeprom memory
 *            lParam |= 4; of configuration memory
 *    wParam = 4   Does reading                        Result is 1 - ok
 *            lParam |= 1; of main code memory                 0 - failed
 *            lParam |= 2; of data eeprom memory
 *            lParam |= 4; of configuration memory
 *    wParam = 5   Does differential programming           Result is 1 - ok
 *            lParam |= 1; of main code memory                 0 - failed
 *            lParam |= 2; of data eeprom memory
 *            lParam |= 4; of configuration memory
 *    wParam = 6   Does verification                   Result is 1 - ok
 *            lParam |= 1; of main code memory                 0 - failed
 *            lParam |= 2; of data eeprom memory
 *            lParam |= 4; of configuration memory
 *    wParam = 7   Does erasing                        Result is same as on command line
 *            lParam |= 1; of main code memory
 *            lParam |= 2; of data eeprom memory
 *                wParam = 1,2,3,4,5,6,7 are 'thread blocking'!
 *    wParam = 15  lParam = any;                            Result is always 1
 *            Does the same like Button GO was pressed
 *            (even using another programmer than PRESTO)
 *    wParam = 16  query for capability
 *    wParam = 17  do requested action
 *       wParam = 16 and 17 has same lParam values:
 *            lParam = 0; MCLRControl_Run
 *            lParam = 1; MCLRControl_Stop
 *            lParam = 2; MCLRControl_Reset
 *            lParam = 8; Actual voltage on PRESTO
 *                    0 = Unknown
 *                    1 = 0V
 *                    2 = ~2V
 *                    3 = 5V
 *                    4 = >6V
 *    wParam = 32  reinitialization of UP
 *            lParam |= 1; reload settings (=reload project/ini file or registry)
 *            lParam |= 2; reload language file
 *            lParam |= 4; recreate programmer      (like programmer was changed)
 *            lParam |= 8; reload programmer settings (like port settings)
 *            lParam |=16; reload selected part
 *            lParam |=32; reload hex file
 *            lParam |=64; recreate all dialog windows (adjust their size when reloading part)
 *            lParam == 0x0100; refresh part specific windows
 *            lParam == 0x0200; refresh all editors
 *            lParam == 0x0300; refresh project captions
 *    wParam = 33  lParam |= 1; save all project settings
```

```
*   wParam = 48  actual file save (like Ctrl+S is pressed)
*           lParam |= 1; main code memory will be saved
*           lParam |= 2; data eeprom memory will be saved
*           lParam |= 4; configuration memory
*              (for AVRs |=4 is not possible and (1+2) is not possible)
*   wParam = 56, lParam=0; will return the handle of the UP main form
*)
{=============================================================================}
(*
* WM_CLOSE:     will close the program
*)
```

Example

```
var
 window: HWND;
begin
 window := FindWindow('up v1.x', nil);
 Result := SendMessage(window, WM_USER, 0, 0);
end.
```

Usage of UP_DLL.DLL

Note: UP_DLL is communicating with UP, so UP must be running at the time. **UP_DLL is not standalone "thing".**

String values may be sent to UP by using up_dll.dll library.

unit up_dll;

interface

Function UP_LoadFile   (FileName: PChar; style: integer): integer; stdcall;
```
(*
*  Load File (with extension .hex or .ppr);
*  Loading of .ppr file can result in loading .hex file too;
*  Result codes are same like on command line.
*
*  Style |= 1; UP will be quiet on file load errors
*  Style |= 2; UP will do no previous file saving
*
*)
```
Function UP_GetStrValue(ValueName: PChar; Value: PChar; Size: integer): integer; stdcall;
Function UP_GetIntValue(ValueName: PChar; var Value: integer): LongBool; stdcall;
Function UP_SetStrValue(ValueName: PChar; Value: PChar): LongBool; stdcall;
Function UP_SetIntValue(ValueName: PChar; Value: integer): LongBool; stdcall;

Function UP_LoadFile_Wnd(WndClass:PChar; FileName: PChar; style:integer):integer; stdcall;
Function UP_SetStrValue_Wnd(WndClass:PChar; ValueName: PChar; Value:PChar): BOOL; stdcall;
Function UP_SetIntValue_Wnd(WndClass:PChar; ValueName: PChar; Value:integer): BOOL; stdcall;
Function UP_GetStrValue_Wnd(WndClass:PChar; ValueName: PChar; Value: PChar; Size: integer): integer; stdcall;
Function UP_GetIntValue_Wnd(WndClass:PChar; ValueName: PChar; var Value: integer): LongBool; stdcall;
```
(*
* All these functions are used for changing internal settings of UP in runtime.
* UP_GetIntValue, UP_SetStrValue, UP_SetIntValue returns nonzero if successful
* UP_GetStrValue returns amount of characters to copy into Value string including null terminator
```

```
 *     If Size is less than requied size, no characters are copied.
 *)
implementation

function UP_LoadFile;    external 'up_dll.dll';
function UP_GetStrValue; external 'up_dll.dll';
function UP_GetIntValue; external 'up_dll.dll';
function UP_SetStrValue; external 'up_dll.dll';
function UP_SetIntValue; external 'up_dll.dll';

function UP_LoadFile_Wnd; external 'up_dll.dll';
function UP_SetStrValue_Wnd; external 'up_dll.dll';
function UP_SetIntValue_Wnd; external 'up_dll.dll';
function UP_GetStrValue_Wnd; external 'up_dll.dll';
function UP_GetIntValue_Wnd; external 'up_dll.dll';

end.
```

Description of setting names and values is in appendix.

4.8 Running UP more than once

When user requires to connect more of the PRESTO programmers to a computer, for each of them UP must be running.

In a typical case only one instance of UP is running at a time: another execution of the application results in passing the command line parameters to already running instance.
UP can be run more than once with each instance having unique window class name. Instances with the same class name will communicate with each other. Windows class name can be specified on the command line using **/w** parameter. All command line parameters are described in separate chapter.

Example

First instance of UP can be run in common fashion from the start menu.
Another instance may be run from command line e.g. `up /w "another up"`

4.9 Access to a programmer by more than one instance

Only a single instance of UP can access a programmer at a time (in case of PRESTO this applies also to other utilities handling the programmer). It is up to the user to ensure that no more than one instance is accessing a programmer which connects to parallel port at a time. In case of USB programmer this is task of access right management of the operating system. The operating system will not allow multiple access to a programmer. UP does not allow other software to access PRESTO even if it is idle for it is continuously monitoring state of the button and voltage on the power supply pin.
It is possible to force UP to release the programmer for other application by using dialog for selection of a programmer. Navigate the menu to select a programmer. As long the dialog is displayed, the programmer is released and can be accessed by another application **There will not be any loss of data** after the dialog is canceled.

4.10 Intel HEX File Format used by UP

UP is using Intel HEX Files to read and write data (common extension of such files is `.HEX`).

Supported HEX file variants

- "basic", also called Intel 8-bit HEX File, MPASMWIN produces this type of files when **INHX8M** parameter is issued
- "extended", also called Intel 32-bit HEX File, MPASMWIN produces this type of files when **INHX32** parameter is issued

Intel HEX File format description

Intel HEX Files are text file compound of lines (records) of following structure:
:LLAAAATTDDDD...CC

- **:** Every line begins with colon (0x3A).
- **LL** Length of the record (number of DD fields).
- **AAAA** Address of first byte of the record.
- **TT** Record type:
  - 00 - Data record.
- 01 - End-of-file record. The file must end with this type of record.
- 02 - Extended segment address. (32-bit HEX only)
- 04 - Extended linear address. (32-bit HEX only)
  There are also other record types (03, 05) which UP ignores when reading a file and does not use when writing a file.
- **DD** Record data. The number of bytes must be exactly LL.
- **CC** Checksum. The checksum is binary complement of sum of all values on the line (8-bit byte by byte addition is used).

Data record

Example of a data record with configuration memory of a 14-bit part.
:02400E00413F30

- Length of the record: **02** - Size of the configuration memory is one word = 14 bits = 2 bytes (aligned to whole bytes)
- Record address: **400E** - Address of the configuration memory is 2007h which, addressed by bytes, is 400Eh
- Record type: **00** - Data record
- Record data: **413F** - Configuration word is 3F41h
- Checksum: **30** = 02 + 40 + 0E + 00 + 41 + 3F = xxD0; neg D0 = 30

End of file

The only form of *end of file* record is:
:00000001FF

Extended addressing

This record types are used when more than 64 kB need to be addressed (e.g. for PIC18F microcontrollers the configuration information is stored at 0x30000000).

In such case it is necessary to use extended linear address record which contains upper 16 bits of the address. Lower 16 bits are stored in the data record.
:020000040030CA
This line sets upper 16 bits of address for configuration memory of PIC18F.

In case of extended segment address record, a segment address (i.e. bits 19-4) is stored. The segment address is then added to the addresses of forthcoming data records.

Saving of the part type identification tag to the `.HEX` file

Since confusion between selected part and the part a `.HEX` file was generated for is a common problem, UP offers a possibility to save part identification directly to the `.HEX` file. If Save part type to the file.
option is selected, the program appends after end of file record an identification of the part type: `#PART=.....` Most of the programs working with Intel HEX files ignore this line, however such file cannot be considered as fully complying to Intel HEX File format.

4.11 Support for calibration memory

Working with calibration information when using UV eraser

Before erasing the part it is necessary to save calibration information. The function "Save calibration information..." can be used to perform this task.

Menu: File → Save calibration information...
Menu: File → Read calibration information...

The program also features function for blank checking the part. When this function is invoked, calibration information is displayed.

Working with calibration information of flash memory equipped parts

The calibration memory contents is **preserved** during common chip erasure.

If it is desired to erase calibration information for any reason, function "Erase all, including calibration memory" can be used to perform this task. (Part → Erase → Erase all, including calibration memory).

Warning*: New flash parts with calibration memory (e.g. PIC12F629) feature also bandgap bits, which are part of calibration memory. Thus these bits will get erased along with calibration memory once "Erase all, including calibration memory" function is used.*

4.12 Application menu overview

Application menu is a common part of most of the Windows applications. Using menu the application functions may be selected. Actions can be triggered either by a mouse click on corresponding menu item or using a keyboard by holding the <ALT> key and pressing a key corresponding to highlighted character of a menu item.

The menu is divided to following categories:

- File
- Edit
- View
- Device
- Options
- Help

- PRESTO programmer settings window

- HEX editor windows

File menu

File → New

Keyboard shortcut: Ctrl+N

Creates a new empty file. If the currently open file has not been saved, the user is prompted to save it first.

File → Open...

Keyboard shortcut: Ctrl+O

Opens a file using standard Windows dialog. For supported HEX file types see Description of Intel HEX file format. HEX and A43 files are loaded as HEX, the other are loaded as BIN.

File → Open next file

Imports next HEX or BIN file with selectable offset. This function is useful if a user needs to load second file to the chip memory. HEX and A43 files are loaded as HEX, the other are loaded as BIN.

File → Reload...

Keyboard shortcut: Ctrl+R

Reloads currently open file from the disk. This function is useful to reload a file updated by another application.

If the option "Check for HEX file updates" is turned on (see Options), the user is asked whether to reload the file whenever it is updated.

File → Save

Keyboard shortcut: Ctrl+S

Saves the file to the disk. If you wish to save the file with a different name, use Save as... instead. Unused memory areas may be skipped during the saving process and thus may not get saved according to selected Options.

File → Save as...

Saves the file with a new name using a standard Windows dialog. Unused memory areas may be skipped during the saving process and thus may not get saved according to selected Options.

File → Import data memory...

Provides with importing of EEPROM memory from separate file using a standard windows dialog. The file is read from the zero address regardless of its contents as if it was plain EEPROM data. A file generated by a compiler cannot be imported this way under normal circumstances.
This function is provided for compatibility with older software which saves EEPROM contents to a separate file. This

function is considered to be obsolete since contents of all memories is saved to a single file by recent applications. (According to Microchip's recommendation)

File → Open hex file with data automatically

If this choice is checked off the UP program will automatically load the hex file for the data memory with loading of the file for the code memory. This choice is active only if separate file for the data memory is opened.

File → New project

Keyboard shortcut: Shift+Ctrl+N

Creation of a new project.
Usage of project files is recommended especially when using several different parts or using several programmers. A project file contains all necessary settings and thus provides with convenient way of loading them all at once.

File → Open project

Keyboard shortcut: Shift+Ctrl+O

Opens already existing project file using a standard Windows dialog. If some other file was open within this project, it is loaded too.

File → Save project

Keyboard shortcut: Shift+Ctrl+S

Saves a project using standard Windows dialog **with a new name**. Saving of the file with the same name is performed automatically likewise the program settings.

File → Close project

Keyboard shortcut: Shift+Ctrl+W

Saves currently open project, closes it and restores the state of the program as it was before starting work on the project.

File → Recent projects

Under this menu item there are remembered last 5 used projects. After clicking on the project name, the project will be loaded.

File → Read calibration information...

Opens a file using standard Windows dialog and reads calibration information from it.

File → Save calibration information...

Using standard Windows dialog creates a file with calibration information read from the part in the programmer. After the part is erased it is possible to load the calibration information back using the command **Read calibration information**.

For further information about support for calibration information see separate chapter about calibration memory.

File → Export to binary...

Writes raw binary data from program or EEPROM memory to a selected file. Data alignment of 8 or 16 bits can be chosen.

File → Exit

Standard Windows keyboard shortcut: Alt+F4
Keyboard shortcut: Alt+X

Exits the application.

Notice*: If the currently opened file has been changed, the application prompts the user with an option to save the changes. If the application termination is forced by computer shutdown and there is no user response Windows terminates the application without giving it a chance to save the file or settings after a while.*
When operating with the hardware the application refuses any system request for shutdown and may be reported as an application which is not responding - this is normal.

Edit Menu

Edit → Fill with value...

Fills selected memory area with given value. This function is usually used to blank (all ones) or zero (all zeros) of selected areas, but may be also used for other purposes.

When this function is invoked, the memory type is preselected according to active window.

If there was a memory area selected it is automatically predefined for filling. A memory area can be selected by holding down the **Shift** key and clicking a mouse or moving by cursor keys. For further information see hex editors.

Edit → Insert text...

Inserts a text encoded in ASCII into a memory. The end-of-line character can be encoded as NULL, CR, LF or CR+LF. It is possible to fill in individual bytes or enclose data by RETLW instruction (applies to program memory only).

When this function is invoked the memory type and starting address is predefined according to active window and selected location.

Edit → Enclose by RETLW

Encloses data in selected area by RETLW instruction.
This function can be invoked only if there a hex editor window active, it may be also invoked from the local menu (right mouse click in the hex editor).

A memory area can be selected by holding down the **Shift** key and clicking a mouse or moving by cursor keys. For further information see hex editors.

View menu

View → Code memory

Toggles visibility status of code memory editor. For further information on editors see separate chapter.

View → EEPROM memory

Toggles visibility of EEPROM memory editor. For further information on editors see separate chapter.

View → Configuration memory

Toggles visibility of configuration memory editor.

View → Show code memory

Keyboard shortcut: Alt+F10

Shows code memory editor and brings its window to foreground. For further information on editors see separate chapter.

View → Show EEPROM memory

Keyboard shortcut: Alt+F11

Shows EEPROM memory editor and bring its window to foreground. For further information on editors see separate chapter.

View → Show configuration memory

Keyboard shortcut: Alt+F12

Shows configuration memory editor and brings its window to foreground.

Device menu

Device → Program

Keyboard shortcut: Shift+F5

- **Program All**
  Keyboard shortcut: F5
  Erases, blank checks, programs and verifies whole part. Before taking the action a check on Device ID, and Code/Data Protection is performed.
- **Program all but EEPROM**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Performs task similar to Program All, except of erasing, programming and verification of EEPROM memory.
  This function is not available for parts without EEPROM memory, Program All should be used instead.
  This function may not be available under certain circumstances of Code or Data protection of the part. In such situation the program offers complete erase and reprogramming of the part, including the EEPROM memory.
- **Program code memory**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Erases, blank checks, programs and verifies code memory.
- **Program EEPROM memory**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Erases, blank checks, programs and verifies EEPROM memory.
- **Program configuration memory**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Programs and verifies configuration memory.

- **Differential programming**
  Keyboard shortcut: Ctrl+F5
  Perform differential programming, i.e. reads the part and programs only memory cells value of which is different to desired one. If code or data protection is applied, it makes no sense to perform differential programming, and thus complete reprogramming will be performed.
  Since this function requires support by the programmed part, it may not be available for some of them.
- **Differential program EEPROM**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Perform differential programming of the EEPROM memory, it works same way as differential programming of the code memory. Since this function requires support by the programmed part, it may not be available for some of them. If code or data protection is applied, it makes no sense to perform differential programming, and thus complete reprogramming will be performed.
  This function must be used with AVR devices, if the user wants to program the EEPROM memory only and the memory were not erased before.
- **Mass production**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Provides with easy way of programming of several pieces of part with identical or similar content. (e.g. different serial number). See Mass production mode for details.

Some items may not be available for certain parts.

Device Read

Keyboard shortcut: Shift+F6

- **Read All**
  Keyboard shortcut: F6
  Reads out the whole part.
- **Read all but EEPROM**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Reads out the whole part except of EEPROM memory.
- **Read code memory**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Reads out the code memory.
- **Read EEPROM**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Reads out EEPROM memory.
- **Read configuration memory**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Reads configuration memory.

Some items may not be available for some parts.

Device Verify

Keyboard shortcut: Shift+F7

- **Verify All**
  Keyboard shortcut: F7
  Compares content of the whole part with contents of the editors.
- **Verify All but EEPROM**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Compares content of the part except of EEPROM with contents of the editors.
- **Verify code memory**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Compares content of code memory with content of the editor.
- **Verify EEPROM**
  Keyboard shortcut can be assigned in Options Keyboard shortcuts
  Compares content of EEPROM memory with content of the editor.

- **Verify configuration memory**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Compares content of configuration memory with content of the editor.

Some items may not be available for some parts.

Device → Erase

Keyboard shortcut: Shift+F8

- **Erase All**
  Keyboard shortcut: F8
  Erases the whole part.
- **Erase code memory**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Erases code memory. This function cannot be used if the part is code or data protected.
- **Erase EEPROM**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Erases EEPROM memory and blank checks it. This function cannot be used if the part is code or data protected.

Blank check is performed automatically after erasure. Since in most of the cases the erasure is performed without any problems, it is possible to turn this feature off (*Option → Settings → Programming → …*).

Device → Blank check

Keyboard shortcut: Shift+F9

- **Blank check All**
  Keyboard shortcut: F9
  Checks the whole part whether it is blank.
- **Blank check all but EEPROM**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Checks whether the part, except of EEPROM memory, is blank.
- **Blank check code memory**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Checks whether code memory is blank.
- **Blank check EEPROM**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Checks whether the EEPROM is blank.
- **Blank check configuration memory**
  Keyboard shortcut can be assigned in Options → Keyboard shortcuts
  Checks whether the configuration memory is blank.

Some items may not be available for some parts.

Device → Select part...

Keyboard shortcut: F4

A dialog window for selection of a part pops up. In case of certain types of memories it is necessary to select also data organization after the part is selected.

The dialog window displays only parts supported by selected programmer. If the selected part cannot be programmed by ICSP, the ICSP mode is turned of automatically.
To select a part not supported by the programmer, select different programmer first.

<u>**Options menu**</u>

Options → Program settings...

Keyboard shortcut: Shift+F10

Options → Program settings...; Programming tab

Keyboard shortcut: Shift+F10

All generic options can be set in this window.

Settings related to particular programmer (e.g. communication port, ICSP mode) are available in equipment configuration; To select type of part to be programmed, memory organization, etc. use separate window, see selecting part type.

- **`Reload  .HEX` file before every programming**
  If this option is selected the file is reread any time programming is invoked. If mass production is active, the file is read first and then following serial number is issued.

- **Ask before programming of OTP / programming of Flash/ Code/Data protection programming/ differential program**
  Set of options determining which actions require user confirmation to be taken. If the program has to prompt the user confirmation it does so just once, except of the programming Code/Data protection, for which additional confirmation may be required. If the program prompts the user to announce some additional information (e.g. The part to be programmed has Code protection active already, erase whole part?), and the answer is positive, no further confirmation is required.

- **Show fuses warning messages**
  User can select if he wants the warning messages associated with some fuses to be shown. It's recommended not to disable this messages.

- **Automatically close status window**
  Causes status window to be closed automatically unless there was a warning or an error has occurred during the erasure / programming / verification.

- **Beep after succesful/unsuccesful finishing**
  Causes to program to use standard system "exclamation" sound when there is warning or error or when there is no error. Depends on users choice.

- **Turn off all sounds of the UP**
  If this item is checked off, the UP never beeps.

- **ICSP programming**
  ICSP programming cannot be used for all parts, on the other hand some programmers require programming of certain parts in ICSP mode only.
  When ICSP cable is used to program microcontroller directly in the application circuitry, the "ISCP power up delay" causes a delay after applying power to the microcontroller, e.g to charge filtering capacitors in application circuitry. PICQUICK and PRESTO has built-in overcurrent protection, which measures the current shortly after applying power to the microcontroller. In ISCP mode the actual delay depends on this option, thus increasing this value unnecessarily increases the probability of possible damage to the part when handled incorrectly. For shorter delays the programmer's circuitry can detect the overcurrent soon enough to prevent the damage to the part. The overcurrent limit is about 100 mA for both power supply and programming voltage.

- **Slower switching of voltage with ICSP** - After selecting of this option it's possible to change charging and discharging time of VDD. When there is a capacitor on VDD pin used, the pin changes its voltage level slower. This may cause problems during programming. Solution of this problem is to increase the charging and discharging time. A formula to compute the time what would be selected can be found in "**Using ICSP**" [section](#).

- **Don't any blank check before cfg word programming**
  Most of the rewritable parts are able to overwrite configuration memory without erasing the whole part. To

take the advantage of this, blank checking of configuration memory can be disabled. This does not apply to programming of the whole part when complete erasure is performed. Skipping the blank check after erasure slightly speeds up the programming process which is useful especially during development. Improperly erased part will not be programmed correctly, which shows up a bit later during verification process, on the other hand improper erasure happens during one in several hundreds attempts only.

- **Do not perform blank check after erasing**

  When this item is chosen, programmer don't check, whether the device was properly erased. Programming will be faster, but may occur problem with bad erased device.

- **Do not erase device before programming**

  Device will not be erased before the programming process.

- **No data memory erase before its programming**

  This item is useful for programming the AVR devices, where .hex file have to be loaded separately for EEPROM memory. If user wants not to change EEPROM memory, this choice can be used.

- **Don't verify unprogrammed words at the end of the memory**

  If there is some clear region at the end of the programmed memory, it will not be verified. This function makes the verification faster, the user usually do not mind the content of the clear region at the end of the memory.

- **Do not verify**

  This function allows to skip the verification step during programming. It can be useful for development, but this function must not be used for production programming! If the verification is skipped, we cannot be sure that the chip has been programmed correctly.

Options → Program settings...; Panels tab

Keyboard shortcut: Shift+F10

In this part of menu can be configured view of the application window. User can set where and what controlling component to show.

- **Toolbar** is a bar with speed buttons just under the application menu. If you desire to hide the tool bar completely, turn off both Show captions and Show icons in toolbar options
- **Status bar** (in the bottom of main application window) the status bar can be also disabled, otherwise it is used to displays information about selected programmer, ISCP mode, selected part, file modification status etc. The status bar reacts to double-click and right click of the mouse.
- **Mass production counter** is displayed on the status panel and shows total number of successfully programmed parts.

Options → Program settings...; Files tab

Keyboard shortcut: Shift+F10

- **File save style**
  This option can be used to inhibit writing of particular parts of memory to `.HEX` file.
- **Do automatic check for newer versions of actual .hex file**
  Useful especially during software development - the file will be reread when its creation/modification time changes.
- **Check part number when loading `.hex` file**
  Causes part type identification tag in the `.HEX` (if any) to be compared with selected part and asserts a warning if they do not match.
- **Save part number into `.hex` file**

Part type identification tag will be appended at the end of the file if this option is selected Most of the programs working with Intel HEX files ignore this line, however such file cannot be considered as fully complying with Intel HEX File format. See Intel HEX files for more information.

• **BIN file loading and saving style**
This option allows to select if the program asks how to load or save a BIN file for parts what have more bytes per word. The program can ask before every BIN file loading and saving or it can be loaded like Little Endian or Big Endian.

• **Save unused positions to `.hex` file**
If this option is turned off the resulting file may be shorter, however this may also cause confusion since memory cell is considered to be blank if all its bits are set (i.e. FFFh, 3FFFh etc.) which may also be valid instruction (e.g. 3FFFh represents addlw -1)
The file is saved in blocks of 8 or 16 bytes, thus the probability of omitting of such instruction is pretty low.

• **Clear main / data / ID locations before reading.**
All bits in corresponding memory area are set upon reading the file. This will cause all positions not saved in the file to be clear.

• **Erase configuration memory before reading** – It's useful to deselect this option, when fuses setting is not stored in a .hex file.

• **Read data memory / ID locations from the device**
Causes EEPROM / ID to be read from the part inserted in the programmer rather than a file, which prevents this information to be overwritten during programming.
Warning! This function may cause spontaneous access to the programmer e.g. upon starting the program.

• **Project storing style**

Here can be set the project saving options.

Options → Program settings...; Colors tab

Keyboard shortcut: Shift+F10

Colors of hex editors can be changed to meet users needs and aesthetic feeling.

Options → Program settings...; Editors tab

Keyboard shortcut: Shift+F10

• **Narrow code memory editor**
Causes the code memory editor to display 8 instead of 16 cells per a line. This is useful especially for low resolution displays. This option may change spontaneously when different part is selected.

• **Mask ID positions**
According to specification the ID locations should be masked with leaving (typically) only 4 bits for user data. If this option is selected, program will handle this automatically.

• **Configuration memory editor: show cfg word instead of fuses**
Recommended for advanced users only.
This option provides with direct editing of configuration word.

Options → Program settings...; Serial numbers tab

Keyboard shortcut: Shift+F10

For information on serial numbers and concerning features, see separate chapter.

Options → Program settings...; Others tab

Keyboard shortcut: No shortcut

- **Update check settings**
  The updater options can be set here. There are possibilities that the UP will ask at the start if it can connect to Internet to be able to check for the a program version or it won't ask and will always check for the version or it will never ask and never connect to the Internet.

Options → Language selection...

Keyboard shortcut: Ctrl+L

Select a language file using standard windows dialog. By this a single installation of the program can be used in several language mutations.

Options → Keyboard shortcuts

Keyboard shortcut: Ctrl+K

A dialog window for assignment of keyboard shortcuts comes up after invoking this command.

Help menu

Help → Help on program

Keyboard shortcut: F1

Invokes the help you are currently reading. The help may be invoked from various places, always by pressing F1.

Help →List of supported devices

Shows a list of all supported devices.

Help → ASIX - Tools website

Opens company web pages ASIX s.r.o..

Help → Check Internet for an update

Will access the Internet and check if there is a new version of the program.

Help → About

Shows a window with information about the program.

PRESTO programmer settings window

Idle power supply

- **None / External**: The programmer does not supply any power on VDD pin. It is possible to Run / Stop the application program only if external power supply is available.
- **Internal 5V**: The programmer supplies 5V on VDD pin. The application may be powered from this pin.

Active power supply

- **External 3 to 5V**: The programmer will not supply any power to application, on the contrary it will feed its input/output circuitry from the application.
- **Internal 5V**: The programmer will supply 5V for the part on VDD pin.

Options related PIC microcontrollers

-MCLR pin control

The state of -MCLR pin in idle can be controlled using these buttons.

Programming method

- **HVP**: Classical programming with 13V applied to -MCLR/VPP
- **LVP**: LVP mode, only logical values 0 and 1 are applied to -MCLR/VPP.

Programming algorithm

- **Auto**: Particular algorithm will be selected according to voltage present on the VDD pin.
- **Vcc=5V**: Algorithm for fast 5V programming is selected.
- **Vcc=3 to 5V**: Slow programming algorithm, though working with all voltages will be used.

PE

With PIC24 and dsPIC33 it's possible to choose programming method using PE CheckBox. The PE means "Programming Executive", this method is usually faster then common method of the programming.

Options related to AVR and 8051 microcontrollers

Oscillator frequency

The AVR microcontroller require either external or internal oscillator running during the programming. The maximal possible communication speed depends on the frequency of the used oscillator.

Faster programming with slow clock

After the chip erasing its fuses are programmed so that maximal frequency of internal oscillator is used. Then the programmer can communicate faster with the part. The programmer can reach shorter programming times for parts what use slow clock. This function only works when all the chip programming is selected, because the required fuses values must be back programmed at the end of the process.

Inverse RESET

If this choice is selected, the programmer makes inverse reset signal. It's useful if some reset circuit what needs inverse input level in comparison with the output is used in the application and when the programmer is connected to the input of this circuit.

HVP

If this choice is selected, the programmer will use high voltage during communication with the chip. This allows programming of the chip also if the external RESET signal is switched off.

Options related to I2C memories

I2C bus speed

Choose maximal speed of the I2C bus. PRESTO is using internal pullup resistor of 2.2kΩ. when working with I^2C bus.

I2C Memory Address

Choose address of an I^2C memory to be programmed.

Hex editor windows

To display contents of memories of to be programmed so called hex editors are used. Different colors are used to display hex editor cells according to origin of the data, and their status, thus it can be easily told which cells were really loaded from the file, which were successfully programmed etc. The colors can be adjusted according to user's taste or needs. (especially recommended on workstations with low color displays), see settings, colors.

Selecting an area

An area in a hex editor can be selected by holding down the shift key and moving the cursor. After desired area is selected it is possible to perform filling with a value and enclose to RETLW instruction which are accessible from context menu (by right clicking the mouse).

Code memory editor

Menu: **View → Code memory**

Keyboard shortcut to show the window:	F10
Keyboard shortcut to close the window:	Esc

Code memory editor shows contents of main code memory or, in case of serial EEPROM parts (24xx, 93xx, ...), the contents of the memory itself.

EEPROM editor

Menu: **View → EEPROM memory**

Keyboard shortcut to show the window:	F11
Keyboard shortcut to close the window:	Esc

EEPROM (data memory) editor is used to show content of additional memory of some parts, typically an EEPROM memory. *Since not all of the parts are equipped by such memory, this editor may not be accessible for some parts.*

Configuration memory editor

Menu: **View → Configuration memory**

Keyboard shortcut to show the window:	F12
Keyboard shortcut to close the window:	Esc

Configuration memory editor shows settings to be programmed to the part but do not reside in any of previously mentioned memories. *Since not all of the parts need configuration data, this editor may not be accessible for some parts.*

Tips for advanced users:

Although this configuration memory can be represented as set of options in fact it is nothing more but a memory, which can be also handled cell by cell, thus it is possible to view this memory also this way. This can be achieved by turning on **Options → Program settings → Editors Show configuration memory as raw data**, or by double clicking the configuration memory window.

ID locations of the part (not to be confused with Device ID) can be also found in the configuration memory window. ID locations can be programmed with a value to identify the part, e.g. serial number. The ID locations are **always available for reading**, even if code or data protection is applied to the part.

    According to recommendation of Microchip the ID locations should not be programmed with any value, only certain number of bits (typically 4) of each position should be used to carry identification data while other bits should be programmed with a default value. This can be achieved by turning on **Options → Program settings → Editors Mask ID locations ...**.

5 <u>JTAG SVF PLAYER</u>

This software is used for programming the components with **JTAG** interface by <u>PRESTO</u> programmer. It is possible to program and test parts for which there is a software providing data in **SVF** or **XSVF** format.

For example:

>   CPLD - Xilinx XC9572XL, XCR3256XL, Altera EPM7128AELC, EPM7064SLC, EPM7128SLC and others

>   FPGA configuration Flash PROMs – Xilinx XC3S1000, XC2V1000 and others

>   memory – XC18V02VQ44I, ...

>   Atmel ATmega128, ATmega64, ...

Notes:

- The XSVF format is recommended for CPLD Xilinx XC9500 only, while SVF is recommended for all other parts.
- JTAG player does not support .JED files.
- See <u>example</u> of Altera devices programming.

5.1 Installation

The installation of JTAG PLAYER is very simple. Get the installation program (JTAG_xxx_EN.EXE, substitute xxx by version number) from supplied CD-ROM or <u>http://tools.asix.net/</u> (the newest version will be at <u>http://tools.asix.net/</u> every time) and run it. During the installation choose only the folder, where to install the Player, and the folder name in the start menu.

5.2 Simple Programming / Testing

Create standard **_Serial Vector Format_ (*.svf)** file used for exchanging descriptions of high-level IEEE 1149.1 bus operations.

Serial Vector Format (*.svf) is recommended file format for all testing and programming except Xilinx CPLD XC9500. For programming Xilinx CPLD XC9500 it is recommended to use only *Xilinx Serial Vector Format* (*.xsvf).
See implementation status of <u>SVF</u> and <u>XSVF</u> files.
Connect PRESTO to JTAG port on your application
See pin description <u>above</u>.
Start *jtagplay.exe* utility
Select **Open & Process File** in program menu

5.3 Examples of creating SVF/XSVF File

Programming of AVR ATmega128
Generate SVF File using *avrsvf.exe* utility available <u>ATMEL</u>'s website located in *Tools & Software* of AVR 8-bit RISC.
For example :
avrsvf -datmega128 -s -e -ifmyfile.hex -pf -vf -ovmyfile.svf -mp
will generate SVF file which processing will result in erase, program and verify of file myfile.hex. Run avrsvf -h for more information.

Programming of Xilinx CPLD
Using iMPACT software available from Xilinx website generate XSVF File. In **Operation Mode Selection** which appears after iMPACT startup, select **Prepare Configuration Files ➔ Boundary-Scan File ➔ XSVF File**. Run all

operations (erase, program, verify, test...) like programmer (e.g. Parallel Cable) is connected and close iMPACT. Processing of XSVF file will do all recorded actions.

Using SVF file is strongly discouraged for programming Xilinx CPLD XC9500. Programming algorithm of XC9500(XL) devices cannot be described in SVF file well.

Programming of Lattice CPLD

First create a JED output file, for example with the ispLEVER software. Then create the SVF file using Universal File Writer (UFW). This program is installed with ispVM. The ispLEVER and ispVM can be downloaded from the Lattice's web.

Programming of Altera CPLD

The QUARTUS II software by Altera is able to generate the SVF file, but it must be first set in the program menu. The SVF file as it is generated cannot be programmed to the chip using the PRESTO, because there is wrong Silicon ID of the chip. Altera said, that the file was intended for ATE devices and they are not disposed to make it usable also for other devices. So the file must be fixed manually. It can be done so that the user simply erases or comment the "CHECKING SILICON ID" section in the SVF file.

5.4 SVF File implementation status

SVF file was implemented with document "Serial Vector Format Specification, Revision E" which can be accessed at http://www.asset-intertech.com/support/svf.html with these limitations:

• SVF Commands *PIO* and *PIOMAP* are not implemented (yet)

• *HDR+SDR+TDR* / *HIR+SIR+TIR* length is maximum 2^{31} bits

• Supported maximum TCK frequencies are 3MHz, 1.5MHz, 750kHz and divisions of 1MHz starting at 500kHz

• *RUNTEST MAXIMUM max_time SEC* parameter is ignored

• *RUNTEST run_count* is maximum $2^{31}/3$ (approx. 715milion)

• *RUNTEST min_time SEC* is maximum $2^{31}/3$ ms (approx. 715 seconds)

• *TRST* and *RUNTEST SCK* commands share one configurable PRESTO pin (VPP) and can never be used together

5.5 XSVF File implementation status

XSVF file was implemented with Xilinx document "XAPP503, Appendix B: XSVF File Format" with there limitations:

• XSVF Commands *XSETSDRMASKS*, *XSDRINC* and *XSIR2* are not implemented yet

• Using XSVF file is recommended only for programming and testing of Xilinx XC9500 CPLDs

5.6 Options Explanation

Default TCK signal frequency

This is TCK clock frequency which will use PRESTO after SVF command *FREQUENCY* with "default" value. XSVF file hasn't any appropriate command, this TCK speed is used all along the file. 3MHz is maximum clock speed which PRESTO can generate.

RUNTEST without run_count

SVF file interpreter should stay for at least specified amount of time in specified state and run clocks on TCK. PRESTO can't run clocks on maximum speed for accurate time (can meet only *min_time SEC* parameter), but can run with much better timing accuracy slow clocks (~100kHz). Without any clocks PRESTO can meet *min_time SEC* parameter very closely, but SVF file specification does not allow this, even so many types of programmed parts are not demanding clocks on TCK.

RUNTEST with run_count and no timing

This command should be interpreted as minimum clocks which should PRESTO do on TCK. However, some SVF file

generators are using this command as wait cycle and assuming clock speed is 1MHz, thus recommended setting is "interpret as RUNTEST min_time with scale 1MHz".

VPP PRESTO pin while running test and after test

Selection between function provided on VPP pin: TRST or SCK described in SVF file or user output (suitable for reset signal to keep programmed part in reset during file processing)

RUNTEST timing multiply (in JTAG Player version 1.3 and later)

For exact timing specified in SVF and XSVF file, fill this value with 0% (no additional time). For programming of XC9500(XL) is recommended value 100%, for programming of ATmega128 (and other AVRs) it is recommended 25%.

5.7 Running Player from Command Line

SVF & XSVF player can be started from command line for higher comfort especially when debugging.

jtagplay.exe [-p] [-f filename] [-i inifile] [-c] [-cc] [-s]

-p	automatically process once the file specified with -f filename
-f filename	filename to be processed
-i inifile	inifile where are options
-c	close this utility after file was processed without errors
-cc	close this utility after file was processed *with* errors
-s	the parameter "-s serial_number" will select PRESTO. Serial_number can be for example "1234"

or "011234". The "01" is fixed prefix.

Utility returns these codes:

0	last file processing was without errors
1	last file processing failed
2	last file processing could not start

6 PRECOG

This software is used for programming Cyan Technology eCOG1 microcontroller. It includes basic debug control via eICE interface (Run, Stop, Reset).

6.1 Installation

The installation of PRECOG is very simple. Get the installation program (PRECOG_xxx_EN.EXE, substitute xxx by version number) from supplied CD-ROM or http://tools.asix.net/ (the newest version will be at http://tools.asix.net/ every time) and run it. During the installation choose only the folder, where to install the PRECOG, and the folder name in the start menu.

6.2 Device programming

Connect PRESTO to eCOG1.

Open a data file by clicking **Open** button or **File/Open** menu item. Supported are the files with extension .rom

Press the **Program** button or **Device/Program** menu item to start programming..

6.3 Debugging

Connect PRESTO to a eCOG processor.

Press the **Attach** button or **Device/Attach** menu item.

Now can be used debugging buttons (Run, Stop, Reset) or the same named items in Debug menu.

7 <u>presto.dll library</u>

Functions implemented in the presto.dll enable setting or reading of logical values at single pins of the PRESTO programmer. Various communication protocols can be implemented this way. *QSetPins()* function enables output pins control. *QGetPins()* function enables input pins reading. *QSendByte()* function enables a fast SPI Byte on the data and clock pins to be sent. If also reading is required, then the *QSendByte_OutIn()* can be used. Then there are also functions for the programmer features settings, for supply and programming voltages control and functions for reading of the returned values.
The library can be used with all PRESTO programmers, it does not depend on the version of the programmer.

The functions implemented in the presto.dll library are described in detail in different <u>document</u>.

APPENDIX A: <u>Configuration word addresses in PIC devices</u>

PIC10xxx Configuration word addresses

All PIC10xxx devices have the configuration word at address FFFh.

PIC12xxx Configuration word addresses

FFFh		2007h
PIC16F505	PIC12C509A	PIC12C671
PIC12C508	PIC12F510	PIC12C672
PIC12F508	PIC12CE518	PIC12CE673
PIC12C508A	PIC12CE519	PIC12CE674
PIC12C509	rfPIC12C509Ax	PIC12F629
PIC12F509		PIC12F675
		rfPIC12F675x
		PIC12F635

PIC16xxx Configuration word addresses

PIC16xxx with cofig mem addr. FFFh			
PIC16C54	PIC16C57	PIC16C54-LP	PIC16C56-LP
PIC16C54A	PIC16C57C	PIC16C55-HS	PIC16C57-HS
PIC16C54B	PIC16C58A	PIC16C55-RC	PIC16C57-RC
PIC16C54C	PIC16C58B	PIC16C55-XT	PIC16C57-XT
PIC16C55	PIC16HV540	PIC16C55-LP	PIC16C57-LP
PIC 16C55A	PIC16C54-HS	PIC16C56-HS	PIC16F54
PIC16C56	PIC16C54-RC	PIC16C56-RC	PIC16F57
PIC16C56A	PIC16C54-XT	PIC16C56-XT	PIC16F59
PIC16C505			

Note: All other supported PIC16xxx have the configuration word at address 2007h.

PIC18xxx Configuration word addresses

Microcontroller	Cfg. Mem. Addr.	Microcontroller	Cfg. Mem. Addr.	Microcontroller	Cfg. Mem. Addr.
PIC18F24J10	3FF8h	PIC18F63J11	1FF8h	PIC18F83J11	1FF8h
PIC18LF24J10	3FF8h	PIC18F63J90	1FF8h	PIC18F83J90	1FF8h
PIC18F25J10	7FF8h	PIC18F64j11	3FF8h	PIC18F84J11	3FF8h
PIC18LF25J10	7FF8h	PIC18F64J90	3FF8h	PIC18F84J90	3FF8h
PIC18F44J10	3FF8h	PIC18F65J10	7FF8h	PIC18F85J10	7FF8h
PIC18LF44J10	3FF8h	PIC18F65j90	7FF8h	PIC18F85J11	7FF8h
PIC18F45J10	7FF8h	PIC18F65J15	BFF8h	PIC18F85J90	7FF8h
PIC18LF45J10	7FF8h	PIC18F66J10	FFF8h	PIC18F85J15	BFF8h
		PIC18F66J15	17FF8h	PIC18F86J10	FFF8h
		PIC18F67J10	1FFF8h	PIC18F86J15	17FF8h
				PIC18F87J10	1FFF8h

Note: All other supported PIC18xxx have the configuration word at address 300000h.

dsPIC30xxx Configuration word addresses

All dsPIC30 devices have the configuration word at address 7C0000h.

dsPIC33xxx Configuration word addresses

All dsPIC33 devices have the configuration word at address 7C0000h.

PIC24xxx Configuration word addresses

Microcontroller	Cfg. Mem. Addr.
PIC24FJ16GA0xx	2BFCh
PIC24FJ32GA0xx	57FCh
PIC24FJ48GA0xx	83FCh
PIC24FJ64GA0xx	ABFCh
PIC24FJ96GA0xx	FFFCh
PIC24FJ128GA0xx	157FCh

All PIC24H devices have the configuration word at address 7C0000h.

APPENDIX B: UP_DLL.DLL setting names and values

Look into some example batch files to see how to work with these settings.

This information is provided only for expert users and without any warranty.

types are

  string = is string

  integer = signed 32bit value

  boolean = accessed like integers; 0 is false, other value is true

==

Prog.LoadFileBfgProg

      boolean

      If true, hex file is reloaded every time part is programmed

File.AutoCheck

      boolean

      If true, hex file is periodically tested for changes

File.LoadOnModify

      boolean

      If true, when change is detected, question pops up

FileLoad.ClearData

FileLoad.ClearCfg

FileLoad.ClearID

FileLoad.ClearCode

      boolean

      If true, contents of code memory are erased (in UP memory) before new file is loaded;

      all cells not stored in hex file will have its default (blank) state

Part.Name

      string

      Selected part name

Prog.Name

      string

      Selected programmer name

           values are PICCOLO, PICCOLOG, CAPRPI, PVKPROP, PICQUICK, PREST

Prog.BusSpeed

    integer

    Communication speed

        1 = Accelerated

        2 = Fast

        3 = Medium

        5 = Slow

Prog.ICSP

    boolean

    ICSP settings

Prog.PortBase

    integer

    Base address of used LPT port or serial number of device

LanguageFile

    string

    Relative path to used language file

Project.File

    string

    Project file path

Project.Present

    boolean

Project.Template

    boolean

    If true, user is asked for project name before its saving

HexFile.File

    string

    Opened hex file path

HexFile.Present

    boolean

HexFile.Template

>boolean

>If true, user is asked for name before saving

HexFile.SaveVoid

>boolean

>If true, empty cells are saved too

Prog.QBfrEraseFlash

>boolean

>Question before erasing flash parts

Prog.QBfrProgFlash

>boolean

>Question before programming flash parts

Prog.QBfrProg

>boolean

>Question before programming OTP parts

Prog.QBfrDiffProg

>boolean

>Question before differential programming (of flash parts)

Prog.QBfrProgCP

>boolean

>Warning before programming part with some kind of protection

Prog.CloseStatOnGoodAct

>boolean

>If true, status window is automatically closed after read/verify etc... without errors

Prog.CloseStatOnGoodProg

>boolean

>If true, status window is automatically closed after programming without errors

Prog.SkipBlankForCfg

boolean

If true, no blank check of part is performed before programming configuration space

Prog.SkipBlankCheck

boolean

If true, no blank check of part is performed before programming

Serial

integer

0 = no serial numbers

1 = serial numbers are from external file

2 = serial numbers are computed

Serial.Step

integer

Stepping of serial numbers

Serial.File

string

File name of external file with serial numbers

Serial.File.Next

string

Label of serial number

Serial.Length

integer

If serial number is computed, serial number length (digits)

Serial.Actual

(unsigned) integer

If serial number is computed, actual computed serial number (if decimal, coded as BCD)

Serial.ASCII

boolean

If serial number is computed, If true, serial number is stored to part as ASCII characters

Serial.SaveTo

    integer

    1 = code memory

    2 = data memory

Serial.Retlw

    boolean

    If serial number is computed, If true, memory cells are filled with retlw instructions

Serial.Addr

    integer

    If serial number is computed, address where to save

Serial.CPW

    integer

    If serial number is computed, chars per word

Serial.Base

    integer

    If serial number is computer, base of serial number, can be only 10 or 16

Serial.Succ

    integer

    next serial number is

    0 = same

    1 = incremented

    2 = decremented

    3 = random (LSFR)

Serial.Order

    integer

    0 = HiLo hilo

    1 = hilo HiLo

    2 = LoHi lohi

    3 = lohi LoHi

Serial.Write.BeforeProg

    boolean

    If true, current serial number is "written" into opened hex editors just before programming the part.

Serial.Write.AfterProg

> boolean

> If true, current serial number is "written" into opened hex editors after successful programming.

Serial.Succ.AfterProg

> boolean

> If true, next serial number is generated after successful programming

ICSP.LongTime

> boolean

> If true, longer times for switching Vcc are taken

ICSP.LongTime.Time.SwOn

> integer

> Time to wait after Vcc is switched on in microseconds.

ICSP.LongTime.Time.SwOff

> integer

> Time to wait after Vcc is switched off in microseconds.

SpecSettings.PREST.Power

> integer

> 0 = idle power supply is None / External

> 1 = idle power supply is Internal 5V

SpecSettings.PREST.ProgPower

> integer

> 0 = power supply during programming is External 2 to 5V

> 1 = power supply during programming is Internal 5V

SpecSettings.PREST.i2cSpeed

> integer

> 0 = 100kHz

> 1 = 500kHz

> 2 = 1MHz

> 3 = Maximal

SpecSettings.PREST.i2cAddr

>   integer
>
>   0 = first suitable address or N/A
>
>   1 = second suitable address
>
>   etc...

SpecSettings.PREST.LVP

>   integer
>
>   0 = HVP method
>
>   1 = LVP method

SpecSettings.PREST.PICAlg

>   integer
>
>   0 = automatic selection
>
>   1 = assume VDD = 5V
>
>   2 = assume VDD < 5V

SpecSettings.PREST.AVRXTAL.CLK

SpecSettings.PREST.AVRXTAL.RPT

>   integers
>
>   represent maximum AVR oscillator frequency
>
>   values can be found in *.lng files at item
>
>   MainForm.PRESTSpecForm.ComboAVRXTAL.xxx.Items where xxx is minimum
>
>   divisor of system clock of selected AVR's SPI module. This is 2 for
>
>   new AVRs, 3 and 4 for older AVRs and 24 for Atmel's 8051 arch.
>
>   processors.
>
>   These settings can be found in ini file too at
>
>   [SpecSettings.PREST], XTALRpt and XTALClk

APPENDIX C: Using ICSP

ICSP (In-Circuit Serial Programming) is a programming mode of PIC microcontrollers, which provides with programming of parts soldered on printed circuit board. There are two different programming algorithms available: HVP (+13V applied to Vpp) or LVP (using the LVP pin). LVP programming can be disabled in the configuration word. New microcontrollers have the LVP mode enabled by default thus it is necessary to handle the LVP pin during first programming (The LVP pin must be held in log.0 during HVP programming).

Pins used during the programming

HVP algorithm (+13V applied to Vpp)

- **-MCLR/VPP** pin must be separated from reset circuitry (e.g. by 10kΩ resistor). Programming voltage (VPP) of +13V is applied to this pin during the programming, rising slope and the voltage level must not be affected by the application circuitry.
- **LVP** pin (if the part has any) **must be held in log.0!!**
- **RB6 and RB7** pins must not be affected by application circuitry during the programming.

LVP algorithm (without +13V)

- **RB6, RB7, LVP and -MCLR/VPP** pins must not be affected by application circuitry during the programming. Other pins may be in either logical level.

LVP algorithm is supported only by PRESTO programmer so far.

Maximal ratings of the pins (current drawn from the programmer)

	PRESTO	PICQUICK	PICCOLO	CAPR-PI
CLK & DATA, VPP @ 0V/5V	24mA	4mA	4mA	8mA
Vpp @ 13V	50mA[1]	50mA[1]	cca 1mA	cca 1mA

1) For flash memory parts only. In case of OTP parts the application may draw only 1mA. The programmer provides current of 50mA on the VPP pin, but almost all the current is required for programming with the OTP part.

The frequency on data pins may reach several MHz during the programming and the application circuitry must not significantly affect the slopes of the signals.

Powering options

In all cases it is necessary to interconnect common ground (GND).
The microcontroller to be programmed can be powered either

- **externally** from the application
- **internally** from the programmer (5V)

External power supply from the application **cannot** be used for certain types of microcontrollers, which feature -MCLR/VPP pin configurable as I/O.
Internal power supply may be used only if the application circuitry does not draw too much power from Vcc pin of the programmer.

Programmer	Maximal current drawn from the programmer
PRESTO	90mA[1]
PICQUICK	10mA[1]
PICCOLO	50mA

1) The programmer features software overcurrent protection. If the allowed current is significantly exceeded for certain amount of time (configurable) the programmer turns the power supply off. PICQUICK checks overcurrent upon turning on Vdd and Vpp only, while PRESTO checks overcurrent all the time the power supply is on.

- **PRESTO** programmer features hardware support for external power supply. The programmer is capable of using voltage present on Vdd to power its I/O circuitry. This voltage may be even less than 5V. Please make sure whether the microcontroller you use in your application is not only able to run at voltage less than 5V but that it also can be programmed with such low voltage.
- I case of **PICQUICK and PICCOLO (GRANDE)** it is possible to use external power supply if Vdd pin remains unconnected. Nevertheless the external power supply voltage must be the same as programmer's (that is 5V).
- **CAPR-PI** programmer supports external power supply only, no other option is available. The operational voltage is 5V.

If the application contains capacitors which slow down power supply switching it may be necessary to set longer charge/discharge times in software.

Programmer	Charging current	Discharging current
PRESTO	corresponds to 50Ω	corresponds to 1kΩ
PICQUICK	corresponds to 50Ω	corresponds to 10kΩ
PICCOLO	corresponds to 50Ω	none

Approximate times to be set in the software **t[µs] = 2.5 × C[µF] × R[Ω]** .
E.g. for application containing capacitor of 33µF programmed using PRESTO 2.5×33×50=4125µs is needed to charge and 2.5×33×1000=82.5ms to discharge the capacitor.

Notes:

- Sometimes an error can occur, that UP cannot program calibration word or there are errors when reading device ID or it gave error of Overcurrent limit on VDD or so on. In this case can help to lengthen charging and discharging time about several seconds in Program settings... menu item.

- If it gave error of Overcurrent limit on VPP, try to use shorter ICSP cable (at most 20cm).

ICSP connector

All programmers by ASIX are using unified connector for ICSP programming with pin spacing of 2.54mm. This connector has 6 or 8 pins (depends on the programmer) with 5 (or 7) signals. Extended version of the connector (8 pins) features additional LVP pin, which is used for LVP programming.

Pin number	Signal	Programming connector
1	-MCLR	VPP/-MCLR
2		*not used (key)*
3	VCC	VCC
4	GND	GND
5	RB7	DATA
6	RB6	CLOCK
7		*not used*
8	RB3/RB4/RB5	LVP

Recommended wiring of -MCLR/VPP pin
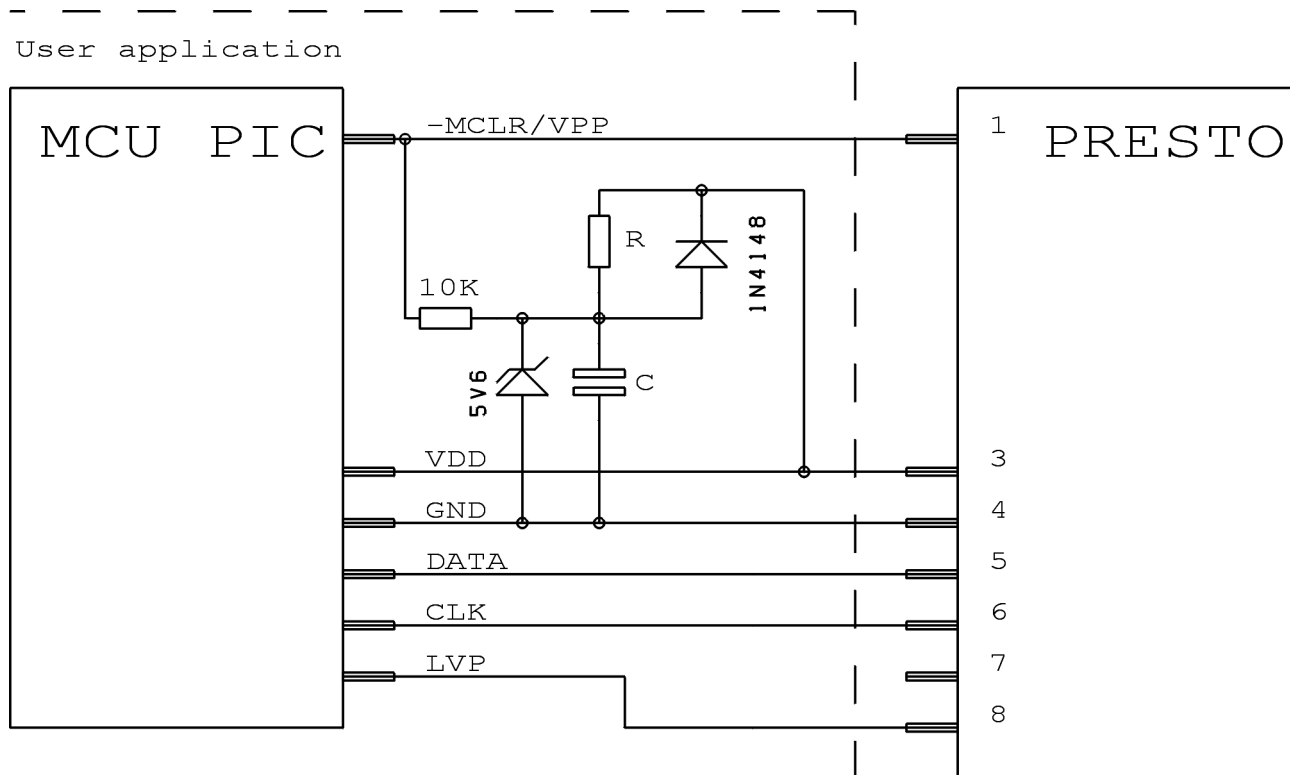
Recommended wiring considering recommendations in Microchip datasheet.
Values of R and C determine time to hold the processor in reset state. The diode causes fast discharge of the capacitor when Vdd is disconnected. Zener diode limits programming voltage (+13V) supplied by the programmer.
The circuitry can be simplified by omitting R,C and diode. Such circuitry has no reset state hold time.

Document Revision History

Date	Revision	Main changes
31-July-2006	1.0	Initial version
20-August-2007	2.0	Added wiring diagrams for MSP430 Modified wiring diagram for 8051 devices Added new notes under the wiring diagrams Added new commandline commands Added information on new supported parts Added new possibilities of serial numbers file commands Added information on command line PRESTO selecting with JTAG Player
2-February-2008	2.1	Added information on HPR1V2 converter Added information on new Windows message and dll functions. Small text fixes
21-April-2008	2.2	Added wiring diagrams for Cypress PSoC devices. Fixed CFG word addresses of PIC24F processors. Added contact information.
7-July-2008	2.3	Added notes on 34xx02 I^{2}C memories. Added new command line parameter /devid and new error code 7. Added new Windows message what causes erasing of the chip. Added new UP Options menu items description.

Date	Revision	Main changes
		Added info how to create the SVF file for Lattice CPLD. Added a note on usage of the /noe parameter with the MSP430.
17-October-2008	2.4	Added a note on PIC32MX.
9-December-2008	2.5	Added information on new "Import next file" function of the UP. Added information on new updater function in the UP Help menu. Added info how the files are loaded in accordance with their extension with File/Open dialog. Added a note for AVR devices. In the Technical specifications specified maximal ICSP cable length. Added info how to create the SVF file for Altera CPLD. Small text fixes.
9-January-2009	2.6	Under the MSP430 SBW picture added info on MSP430F5xxx. Under the MCU 8051 picture added info on supported types.
3-March-2009	2.7	Added a note to LVP programming of the PIC chips with ICPORT fuse. Added information on logging of the serial numbers to a file.
27-March-2009	2.8	Under the JTAG picture added notes on AVR32 programming.
31-July-2009	2.9	Added info on ATxmega devices connections. Added info on Programming Executive. Added info on /blank command line parameter. Added info on Speed ComboBox with MSP430 SBW devices. Small updates and fixes of the text.
23-October-2009	2.10	Added info on M93Sx6 MicroWire memories programming. Added connection diagram for CCxxxx devices by Texas Instruments. Added AT89LP6440 info under the 8051 connection diagram.
15-January-2010	2.11	Added new command line parameters for the UP. The label name "Crystal frequency" changed to "Oscillator frequency". In the "PRESTO options window" section there are new settings. In the Options of the UP there are new settings. In the notes for PIC devices changed value of additional capacitor to 1nF. New parameter in the PRESTO technical specification.
22-January-2010	2.12	Improved note on new PIC chips over current problem and its solution.
14-April-2010	2.13	Added connection diagram for AVR TPI chips. Added notes for PSoC and MSP430 chips.
20-April-2010	2.14	Updated table in the "Description of the programming connector" chapter. In the driver installation chapter added info on installation under Windows 7.
27-April-2010	2.15	Added info on command line parameters /pdiff and /eeonly. Completed information on serial numbers.
14-October-2010	2.16	Added AT89LP52 programming note under 8051 connection diagram. Added a chapter about presto.dll library.
3-March-2011	2.17	Added information on CC430 programming. Links on chips wirings added to contents.
19-May-2011	2.18	In the commandline parameters changed programmer SN format. In the table under SPI memories picture added a description of CS signal. In the menu description added info on "Recent projects" menu item.
20-October-2011	2.19	Added notes for AT89LP51RD2 under 8051 wiring picture. Completed information on the PRESTO driver installation under Windows 7.