Programátory

ASIX®

Referenční příručka

Adresa:	ASIX s.r.o. Staropramenná 4 150 00 Praha - Smíchov
E-Mail:	asix@asix.cz (všeobecné dotazy) obchod@asix.cz (poptávky a objednávky zboží)
WWW:	www.asix.cz www.asix.net (anglicky - hlavní stránka) tools.asix.net (anglicky - vývojové prostředky)
Telefon:	257 312 378
Fax:	257 329 116

Obsah

Úvod

Tento manuál popisuje programátory vyrobené firmou ASIX a jejich ovládací software.

První kapitola popisuje USB programátor PRESTO, jeho instalaci, příklady připojení programátoru k programovaným součástkám a základní vlastnosti PRESTA.

Druhá kapitola se stručně zabývá ostatními programátory a jejich vlastnostmi.

Třetí kapitola je o programu UP. Tento software se používá pro ovládání všech programátorů ASIX. Program UP může být interaktivně ovládán z příkazové řádky, zprávami Windows a prostřednictvím dll knihovny.

Pro programování součástek s JTAG rozhraním slouží program JTAG SVF Player, který je popsaný ve čtvrté kapitole.

Pátá kapitola pojednává o programu PRECOG, který může být použit pro programování procesoru eCOG1 firmy Cyan Technology. Tento program umožňuje také jednoduché ladění programu.

V přílohách jsou popsány pozice konfiguračních slov v podporovaných mikrokontrolérech PIC a jména proměnných, jejichž hodnoty lze nastavit pomocí knihovny up_dll.dll. Příloha C je krátká kapitola o ICSP programování, jsou tam informace o maximálních proudech odebíraných z programátorů, řešení případných problémů spojených s In-System programováním a další informace.

1 PRESTO

PRESTO je velmi rychlý a flexibilní USB programátor využitelný k programování různých součástek – mikrokontroléry, sériové paměti EEPROM a Flash, CPLD, FPGA a další. Programátor je optimalizovaný k dosažení maximální rychlosti programování. Programátor je napájený z USB a může buď během programování napájet programovanou součástku z interního zdroje nebo použít externí napájení z aplikace. Na pinech VDD a VPP je nadproudová ochrana a na VDD pinu je ještě navíc ochrana přepěťová.

1.1 Použití

Programátor PRESTO je navržen pro programování integrovaných obvodů přímo v aplikaci. Seznam podporovaných součástek obsahuje např.:

- **Microchip PIC** mikrokontroléry – součástky se sériovým programováním (Flash, EPROM a OTP), což jsou všechny PICy a dsPICy kromě několika zastaralých součástek
- **Atmel AVR** mikrokontroléry – všechny součástky podporující "SPI Low Voltage Serial Downloading", jako např. ATtiny12, AT90S8535 nebo Atmega128.
- **Atmel AVR32** mikrokontroléry - např.AT32UC3A1256
- **Atmel 8051** mikrokontroléry – součástky podporující ISP programování jako např. AT89S8253, AT89LP4052, AT89LP216 nebo AT89S2051
- **eCOG1** mikrokontroléry firmy Cyan Technology
- **Texas Instruments** mikrokontroléry – 16-bitové MSP430 s pamětí Flash a mikrokontroéry CCxxxx
- **Cypress** – procesory PSoC
- **Sériové EEPROM a Flash paměti** - I^2C (24LCxx), Microwire (93LCxx), SPI (25Cxx)
- **součástky s rozhraním JTAG**, pro které je možné vytvořit SVF soubor. Mezi takové součástky patří CPLD (např. Xilinx XC95xx a CoolRunner), konfigurační paměti pro FPGA (např. Xilinx XC18Vxx and XCFxxS), nebo mikrokontroléry (např. ATmega128) a další
- **součástky s jádrem ARM** – programování a debugování mikrokontrolérů AT91SAM7 umožňuje utilita ARMINE

1.2 Instalace

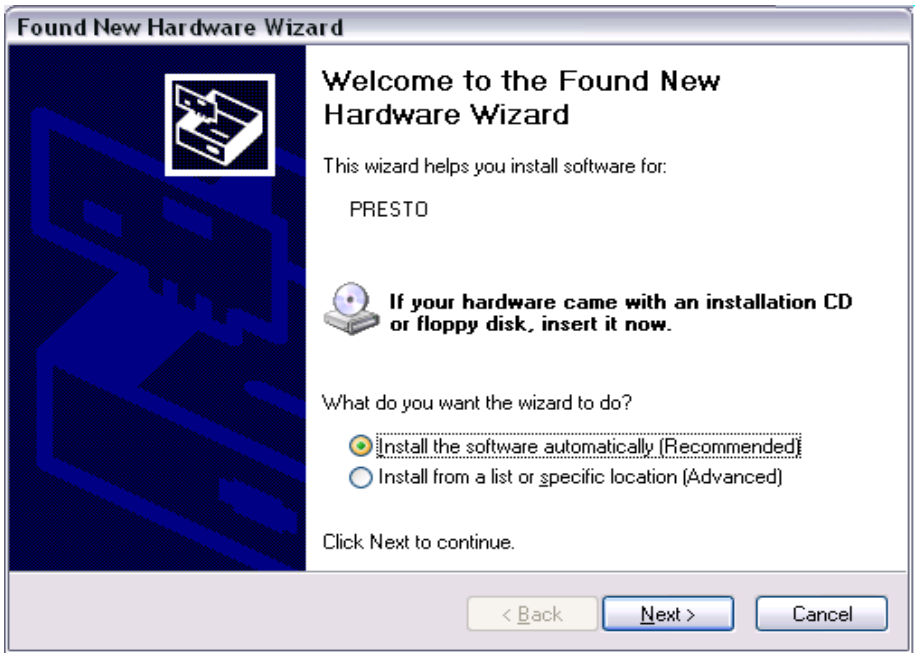
Pro instalaci software musí mít uživatel administrátorská práva a to jak pro instalaci driveru, tak i instalaci a první spuštění obslužného software. Při dalším použití software budou dostačující práva běžného uživatele.

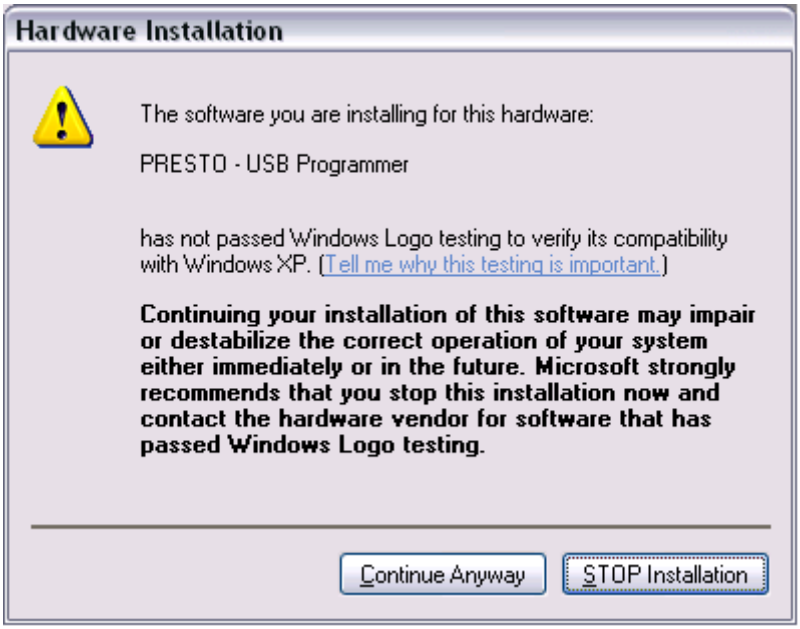
Před první instalací programu UP verze 2.0 nebo vyšší, je nutné nainstalovat ovladače programátoru PRESTO!

1.3 Instalace ovladače pro PRESTO

Poznámka:Při použití nových ovladačů na Windows 98 operační systém v průběhu instalace upozorňuje na chybějící .CAT soubory. V tomto případě je doporučeno použít starší ovladače "ASIX_PRESTO_USB_DRIVERS_2004-04-06.ZIP" dostupné na http://www.asix.cz/dwl_presto.htm .

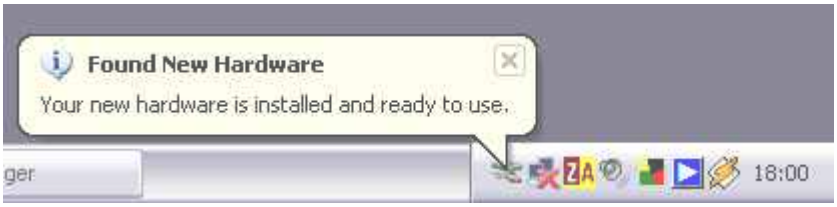
Instalace je jednoduchá. Vložte instalační CD-ROM do mechaniky a připojte PRESTO do USB portu. Operační systém detekuje nové zařízení a začne hledat ovladače.



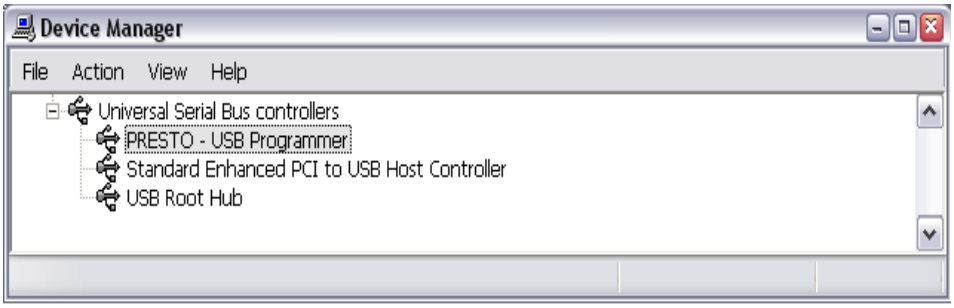


Vyberete doporučenou automatickou instalaci. Operační systém nalezne ovladač na disku CD-ROM a vyzve uživatele k potvrzení instalace necertifikovaných ovladačů. V tomto místě vyberete "Continue Anyway".

Úspěšná instalace ovladačů bude oznámena závěrečným upozorněním.



Funkčnost ovladače může být ověřena ve správci zařízení v ovládacích panelech.



Instalace ovladače pod Windows 7

Pod operačním systémem Windows 7 použijte nejnovější ovladač z webových stránek nebo z CD dodaného s programátorem. Pod tímto operačním systémem se však ovladač nenainstaluje automaticky, je nutné jej nainstalovat ručně následujícím postupem: Ve Správci zařízení najít PRESTO, otevřít jeho vlastnosti, zvolit aktualizaci ovladače a ukázat operačnímu systému cestu k umístění rozbaleného ovladače.

Nyní, když je ovladač programátoru PRESTO nainstalovaný, můžete nainstalovat software :

UP – Program UP podporuje programátor PRESTO stejně jako ostatní programátory ASIX. Program nabízí mnoho pokročilých funkcí jako projekty, ovládání z příkazové řádky, ovládání pomocí zpráv Windows, možností nastavit si vlastní klávesové zkratky, generováním sériových čísel různými metodami atd.

JTAG SVF PLAYER – Tento program se používá pro programování součástek s rozhraním JTAG.

PRECOG – Tímto programem je možné programovat a debugovat mikrokontroléry eCOG

ARMINE – ARMINE je software umožňující programování paměti Flash a debugování mikrokontrolérů s jádrem ARM. Software ARMINE je založený na OpenOCD s přidanou podporou programátoru PRESTO, potřebnými rozšířeními pro programovaní paměti FLASH a jednoduchým rozhraním GUI.

Note: Pokud se zdá, že je PRESTO poškozeno, uživatel ho může otestovat s pomocí testovacího software, který je k nalezení na http://www.asix.cz/supp_testers.htm .

1.4 Popis programovacího konektoru

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]	

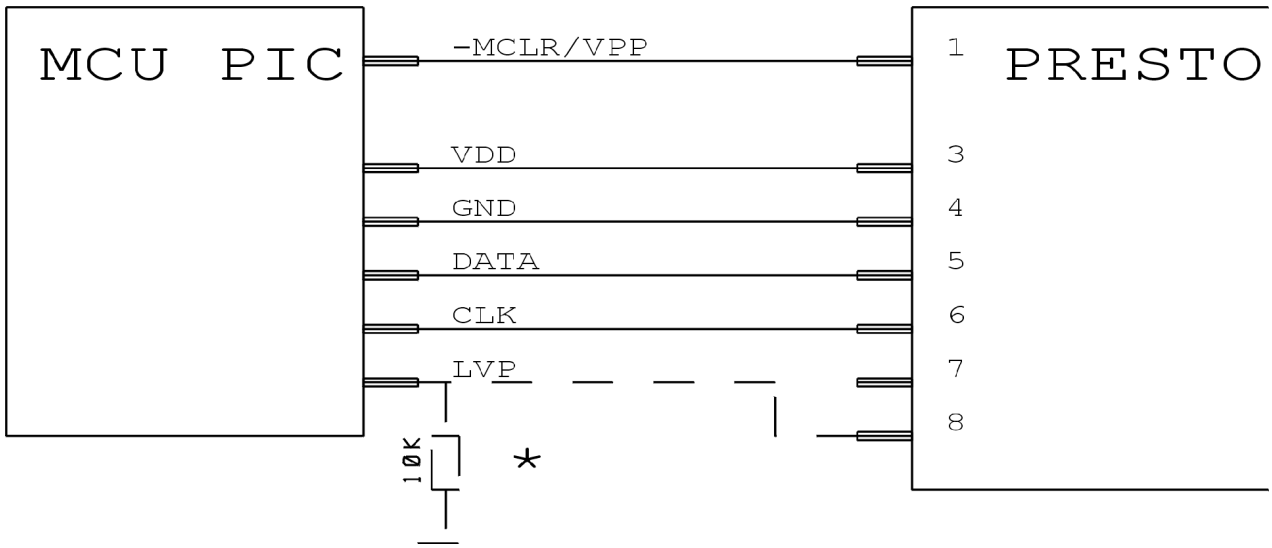
1 – zabudovaný PullUp resistor v PRESTU
2 – pin může zůstat nezapojený, pokud je vhodně zapojený v aplikaci
3 – krystalový oscilátor je nutný, pokud součástka nepoužívá jiný zdroj hodinového signálu
4 – volitelná funkce TRST, SCK nebo uživatelská
5 – log.0 / Z, PullUp v aplikaci je nutný
6 – log.0 / Z
7 – krystalové oscilátory 32.768 kHz a 5.000 MHz jsou nutné
8 – zabudovaný PullDown rezistor1 kΩ

Poznámky:

- **Pro podrobnější informace viz příklady připojení součástek.**
- **Pokud je programovaná aplikace napájená ze spínaného zdroje nebo není uzemněná, může být mezi zemí PRESTA a aplikace velký napěťový rozdíl, což může způsobit zničení PRESTA.**
  Správný způsob jak připojit PRESTO k aplikaci je spojit PRESTO a aplikaci, potom zapnout napájení aplikace a nakonec připojit PRESTO do USB.
  Mnohem jednodušší způsob je uzemnit aplikaci před připojením programátoru a to např. tak, že bude GND pin ICSP konektoru delší než ostatní (tak bude jisté, že se nejprve propojí země), podobně jako je to u USB konektoru (PRESTO je uzemněno na zem počítače).

1.5 Příklady propojení programátoru a aplikace

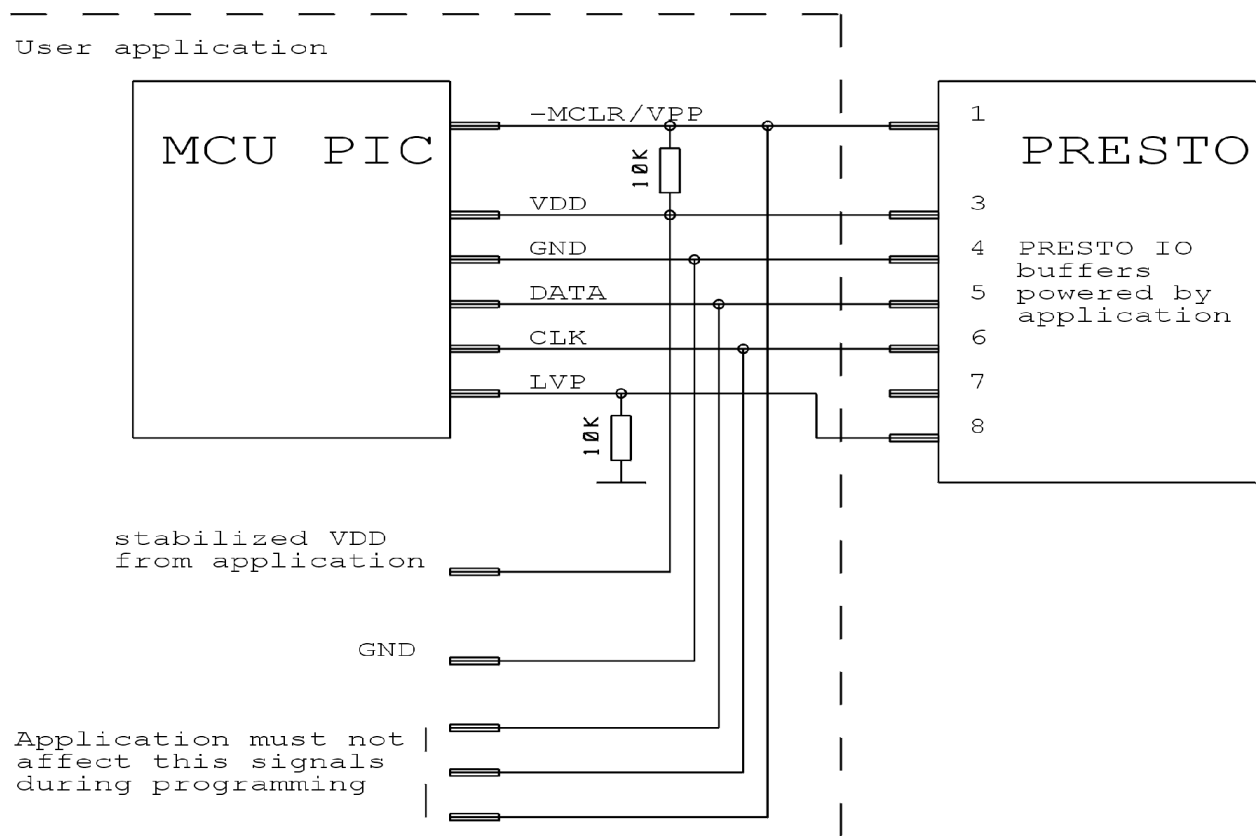
Samostatný PIC bez aplikace, použití HVP (13V) módu



* Připojte buď PRESTO nebo pull-down rezistor

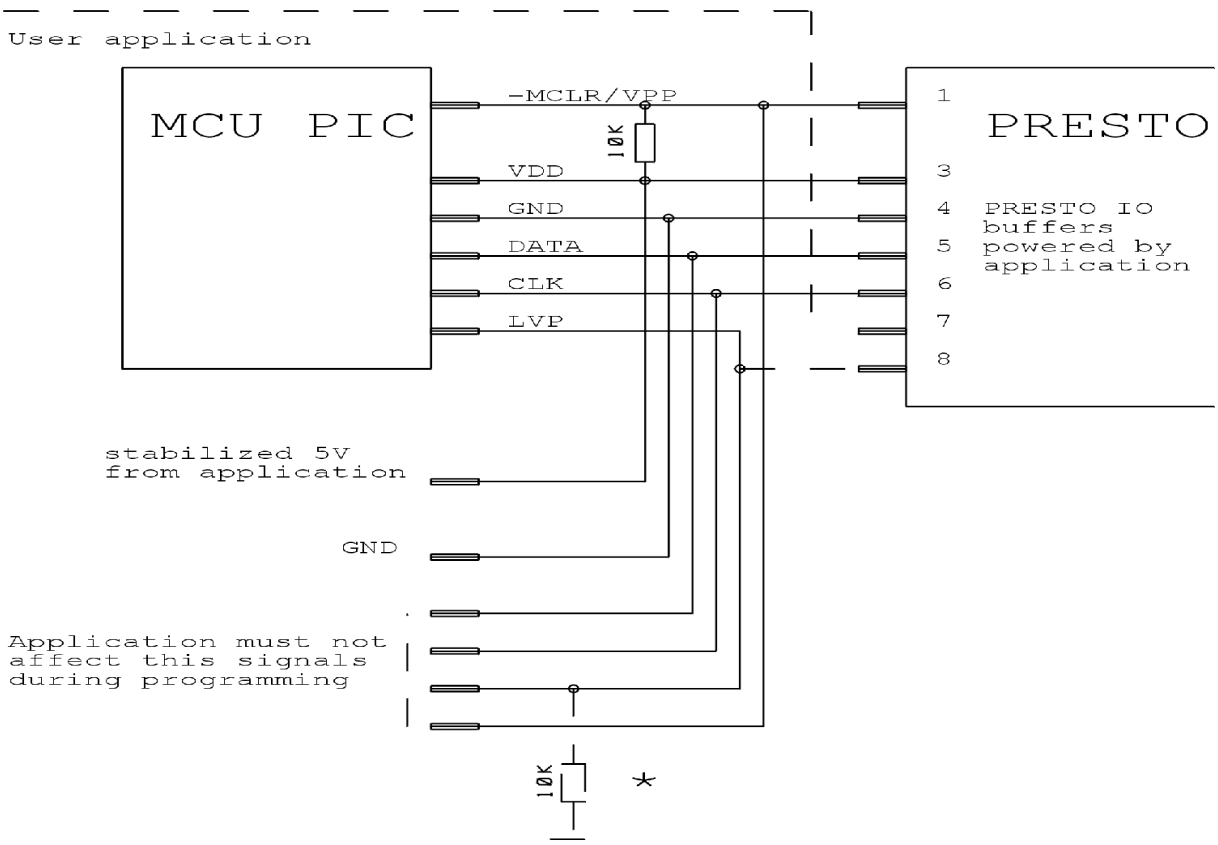
Viz poznámky k mikrokontrolérům PIC a adresy konfiguračního slova.

PIC v aplikaci, použití LVP módu (bez 13V), napájení z aplikace



Viz poznámky k mikrokontrolérům PIC a adresy konfiguračního slova.

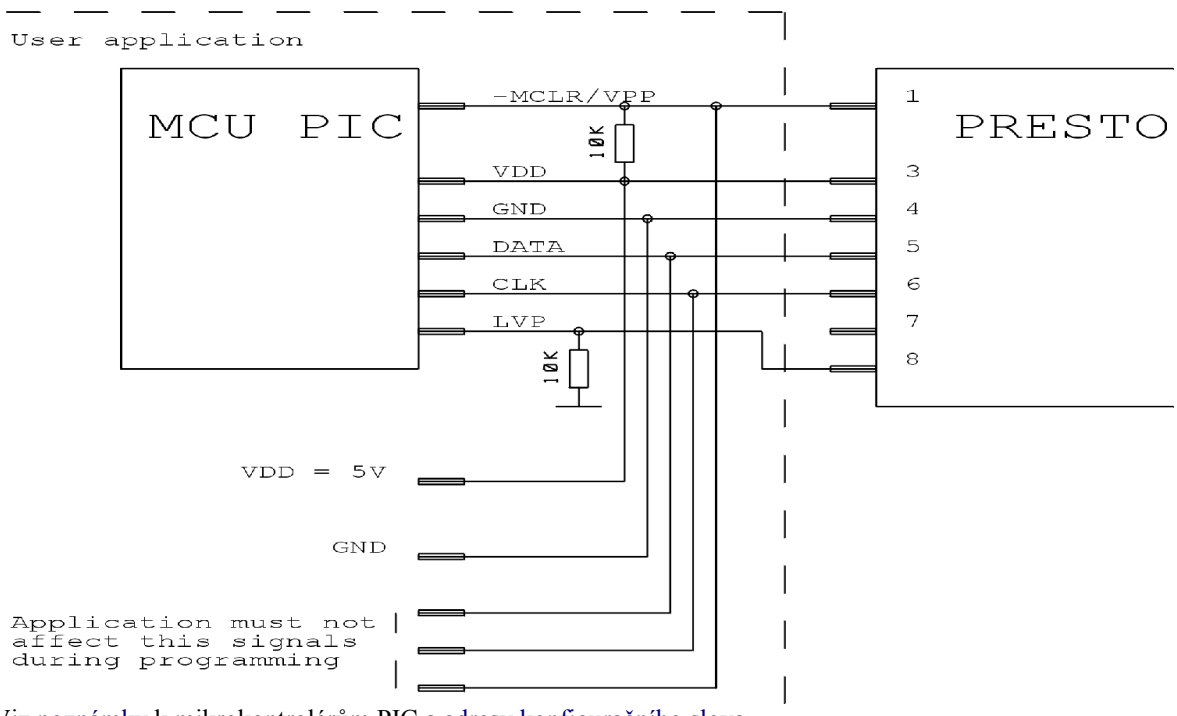
PIC v aplikaci, použití HVP (13V) módu, napájení z aplikace



* Připojte buď PRESTO nebo pull-down rezistor

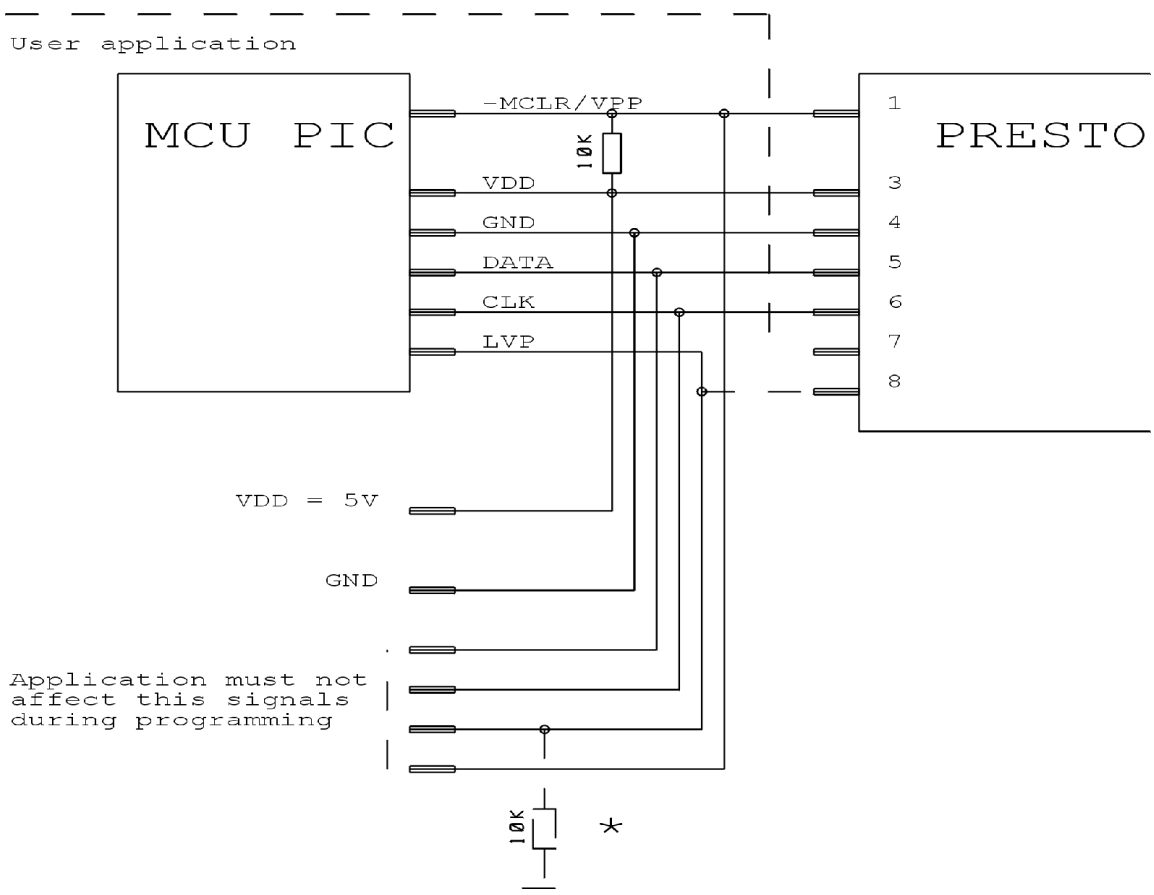
Viz poznámky k mikrokontrolérům PIC a adresy konfiguračního slova.

PIC v aplikaci, použití LVP modu (bez 13V), napájení z PRESTA



Viz poznámky k mikrokontrolérům PIC a adresy konfiguračního slova.

PIC v aplikaci, použití HVP (13V) módu, napájení z PRESTA

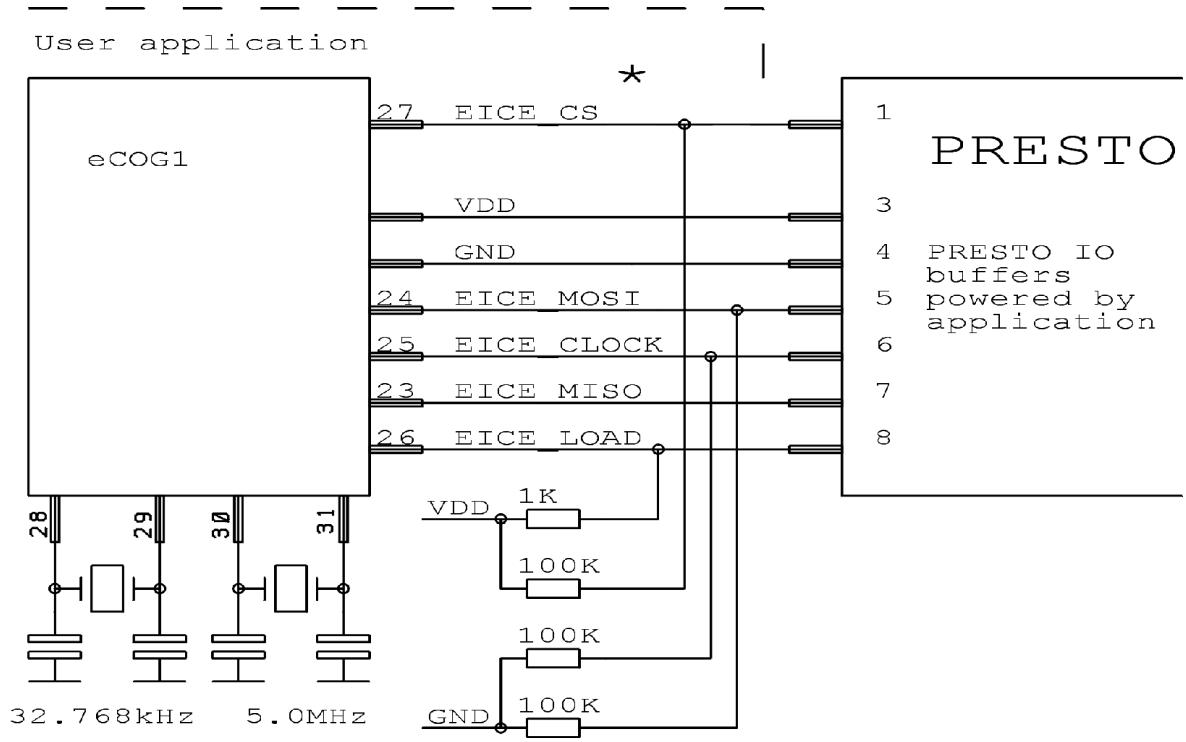


Viz adresy konfiguračního slova.

Poznámky:

- Při programování součástek PIC18FxxJxx, nebude na pinu MCLR 13V, ale maximálně 5V úrovně.
- Pokud je paměť programu nebo datová paměť chráněna pojistkou CP nebo CPD, musí být celá součástka před programováním smazána.
- Většinu součástek není možné smazat při naprogramované pojistce CP nebo CPD s nižším napájecím napětím než 5V.
- Pokud má procesor více VDD a GND pinů, všechny musí být zapojeny, včetně AVCC a AGND pokud procesor nějaké má.
- Součástky, které je nutné napájet 3V je nutné napájet z externího zdroje, PRESTO může dodávat pouze 5V napájecí napětí.
- Některé součástky vyžadují nižší programovací napětí než 13V na pinu MCLR, PRESTO však poskytne 13V. Pokud je vybrána taková součástka v programu UP, program na tuto skutečnost upozorní varovnou hláškou. Uživatel pak může omezit napětí odporovým děličem nebo Zenerovou diodou a rezistorem.
- Při mazání mikrokontroléru v HVP módu může být smazána i pojistka LVP. Aby bylo možné opět programovat procesor v LVP módu, musí být pojistka LVP znovu naprogramována v HVP módu.
- PIC32MX je možné programovat pomocí ICSP rozhraní s použitím externího 3V napájení.
- Součástky s pojistkou ICPORT, musí mít dedikovaný ICSP port vypnutý pro LVP programování.
- Pro součástky PIC24 a dsPIC33 je možné v okně “Nastavení programátoru PRESTO“ zvolit metodu programování pomocí CheckBoxu PE. PE je Programming Executive, tato metoda bývá rychlejší.
- Při použití programátoru PRESTO s některými novými součástkami PIC programovanými v HVP módu se může objevit chybová zpráva indikující nadproud na programovacím napětí. Pokud tato zpráva neindikuje skutečnou chybu v zapojení obvodu, může být způsobena tím, že Microchip změnil technologii výroby součástek. Nově vyráběné obvody i přesto, že se jedná o starší rodiny součástek, se chovají odlišně oproti starším součástkám. Řešením je připojit kondenzátor o kapacitě 1 nF mezi VPP a GND signály. V případě přetrvávajících problémů je možné ještě do signálu VPP sériově zapojit rezistor 10 Ω.

Procesor eCOG v aplikaci, vždy napájený z aplikace (VDD=3.3V)

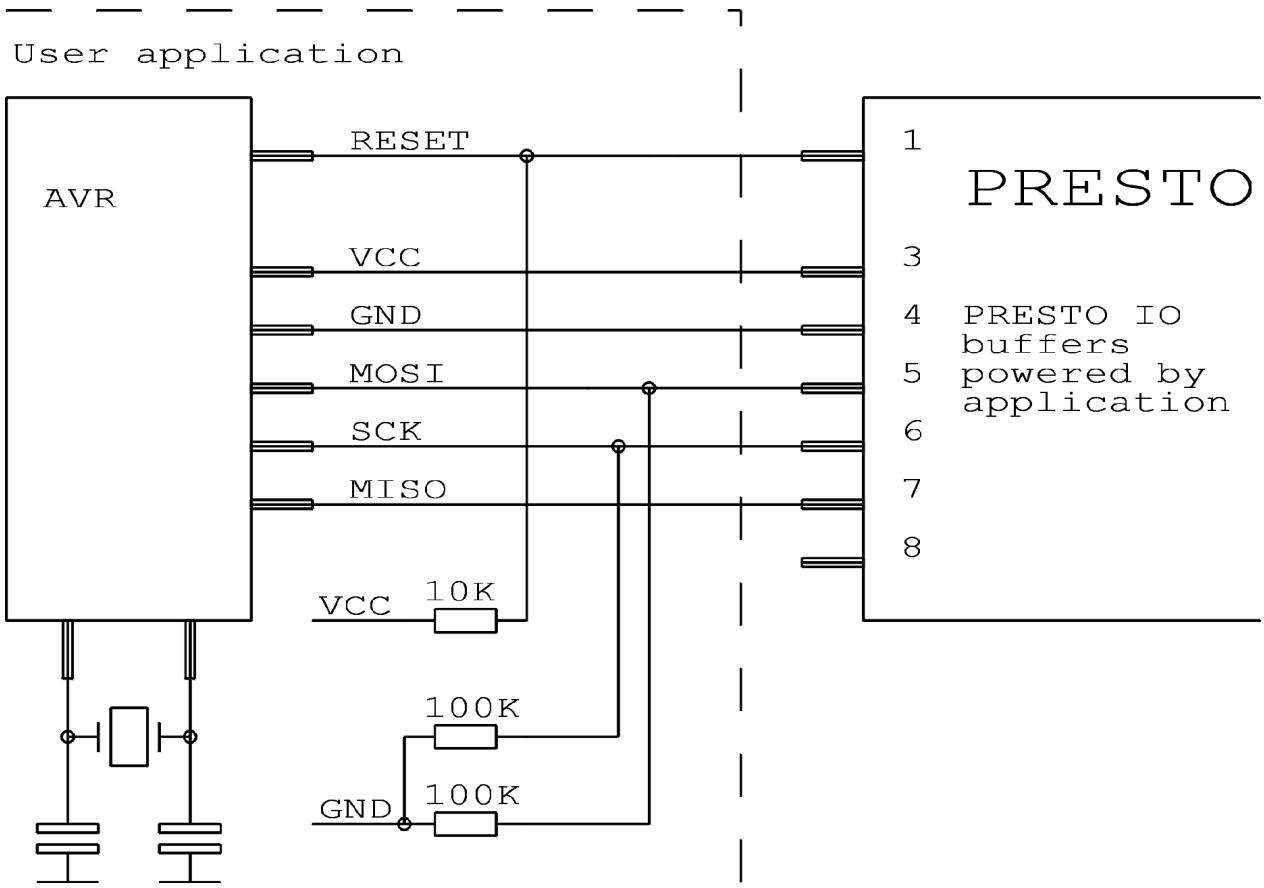


* Připojení pinu EICE_CS k PRESTU není nutné, během programování zůstává v log1.

Notes:

- VDD musí být 3,3V napájení musí být vždy dodáváno z aplikace.
- Aplikace PreCOG určená k programování procesorů eCOG může být stažena z http://www.asix.cz/ nebo je dostupná na přiloženém disku CD-ROM.
- Viz kapitolu o programu PreCOG.

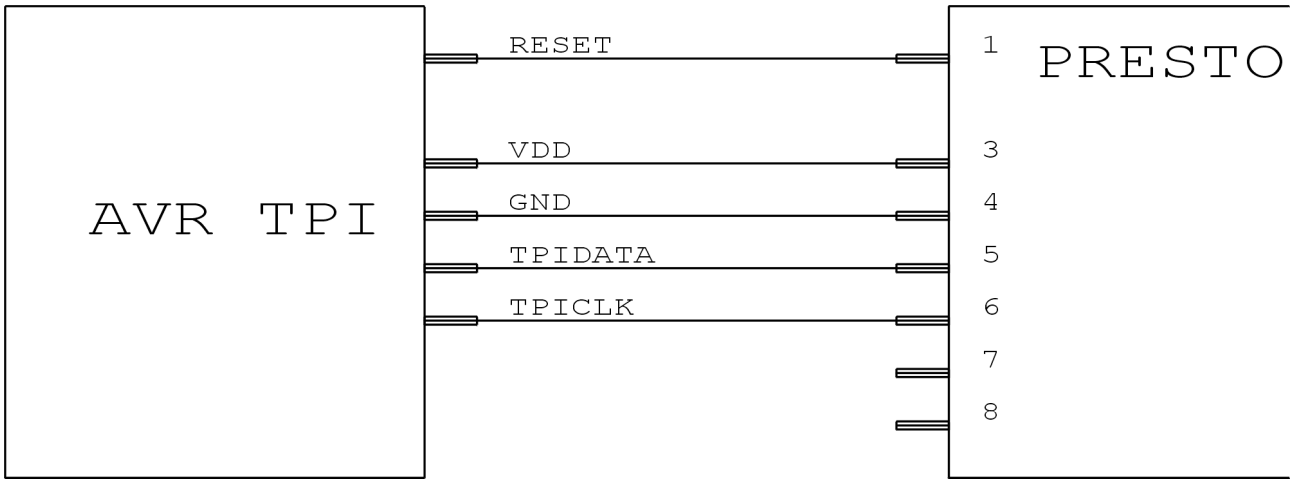
Procesor AVR v aplikaci, napájení z aplikace



Poznámky:

- Pojistky jsou u procesorů ATmega a ATtiny defaultně (od výrobce) nastaveny na interní oscilátor o frekvenci 1MHz, při prvním programování je nutné nastavit položku **Frekvence oscilátoru** v okně **Nastavení programátoru PRESTO** na **>750kHz** nebo nižší. Externí krystal je při programování nutný jen v případě, že jsou pojistky naprogramovány na externí krystalový oscilátor.
- Ke všem AVR procesorům není možné připojit krystal (např. Attiny13, Attiny15).
- Pokud jsou pojistky programovaného procesoru správně nastaveny v okně **Konfigurace**, je třeba kliknout pravým tlačítkem myši do tohoto okna a vybrat položku **Zapamatovat pojistky**. Nastavení pojistek bude uloženo do ini souboru nebo do souboru projektu. Při příštím načtení hex souboru budou pojistky nastaveny podle uložených hodnot. (Pokud je součástka programována spuštěním UPu z příkazové řádky, uživatel musí specifikovat .ppr soubor s uloženými pojistkami místo souboru .hex.)
- V menu **Soubor** je možné zvolit položku **Otevřít hex soubor s datovou pamětí automaticky.** To způsobí, že soubor pro paměť EEPROM bude načten současně se souborem pro paměť programu.
- Pokud je potřeba zachovat obsah paměti EEPROM v mikrokontroléru, použijte pojistku EESAVE. Pokud je tato pojistka aktivní, naprogramujte procesor příkazem **Naprogramovat vše kromě paměti EEPROM**, v opačném případě bude program UP hlásit chybu smazání datové paměti.
- V menu **Nastavení→Nastavení programu...→Programování** je užitečná volba **Nemazat datovou paměť před jejím programováním**.
- Pro napájení 3,3V AVR procesorů je možné použít interní zdroj programátoru s konvertorem HPR3V3.
- Pro konverzi mezi ICSP konektorem PRESTA a 10ti pinovým ISP konektorem firmy Atmel může být použit konvertor **HPRAVR**.
- Některé AVR procesory mají ISP rozhraní vyvedené na jiných pinech než rozhraní SPI. Pro více informací viz kapitolu “Serial downloading“ datasheetu součástky.

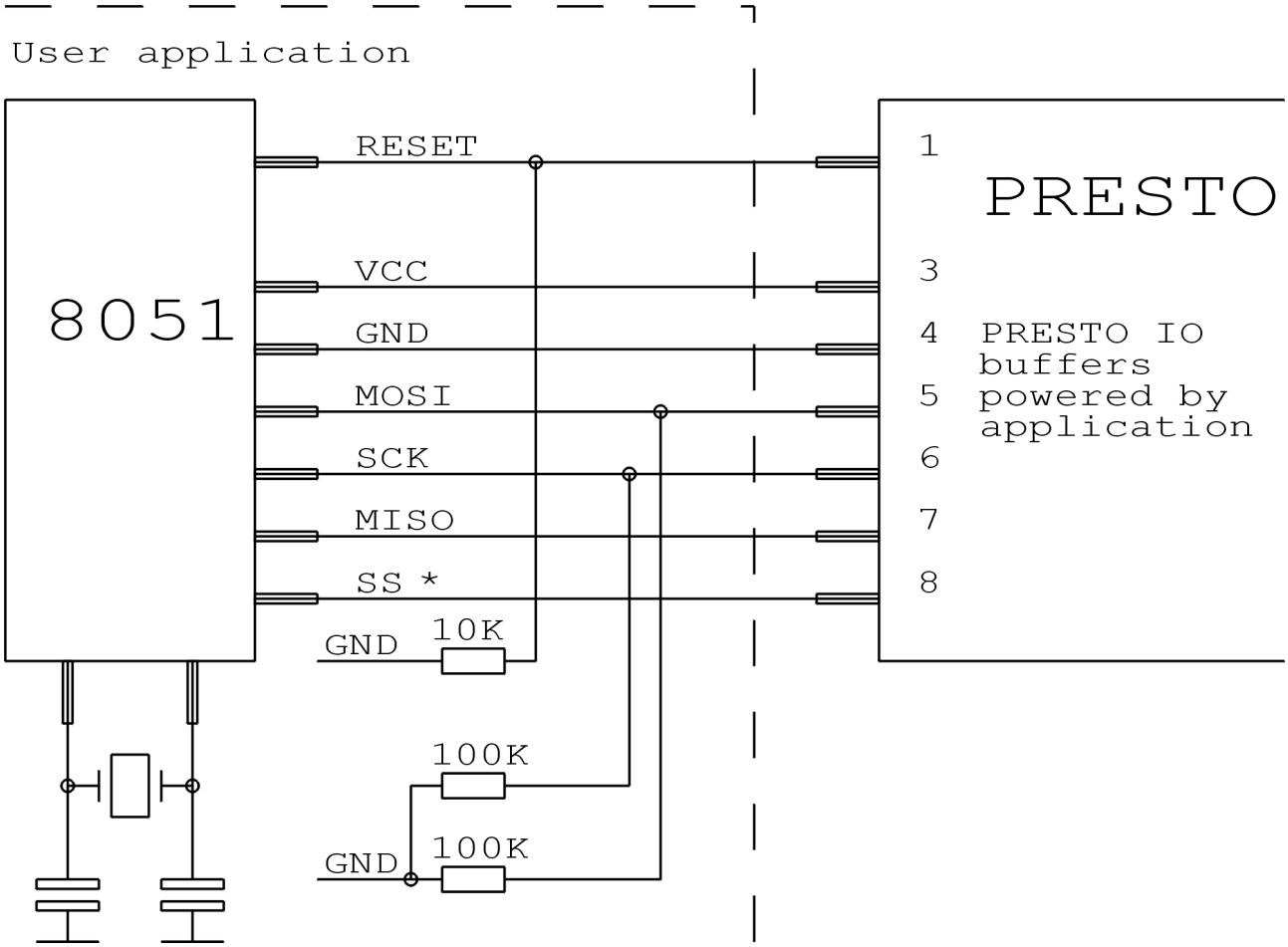
Mikrokontrolér Atmel AVR s rozhraním TPI (např. ATtiny10)



Poznámky:

- Při HVP programování součástky je potřeba na reset pinu 12 V, programátor může dodat pouze 13 V, proto je v tomto režimu potřeba externím obvodem omezit napětí dodané na pinu P1 programátoru. Při standardním low voltage programování není nutná žádná externí úprava.

Mikrokontrolér Atmel 8051 v aplikaci, napájení z aplikace

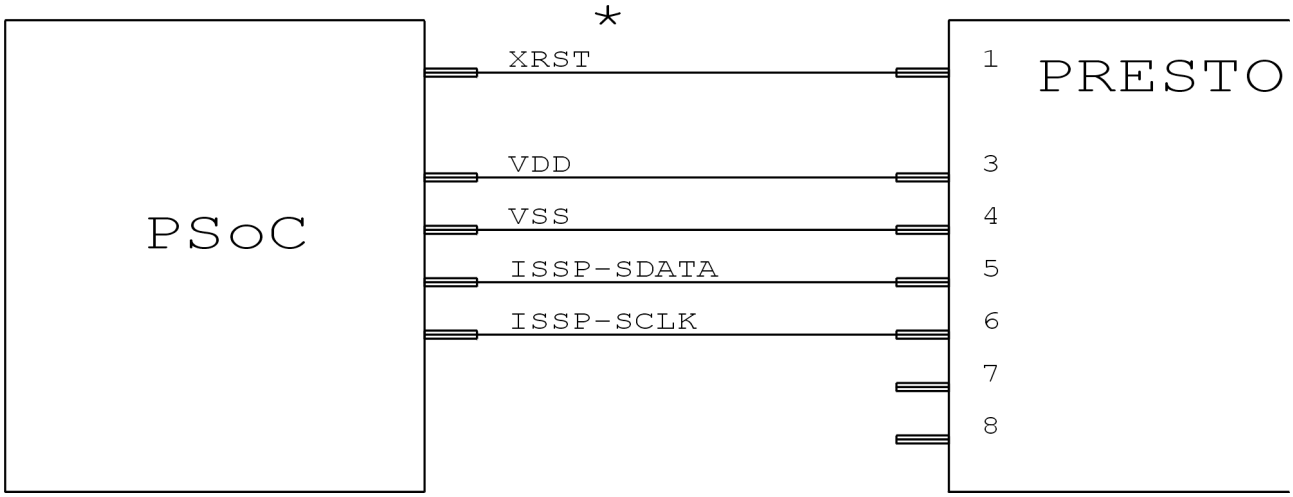


* Pin SS musí byt zapojený pouze u procesorů AT89LP2052/4052/213/214/216/428/828/6440.

Poznámky:

- AT89LP213, AT89LP214 a AT89LP216 mají inverzní RESET. Rezistor na pinu RESET musí být zapojený na VCC a ne na GND.
- Programátor PRESTO nemůže programovat součástky obsahující “C“ v názvu, podporuje však součástky s “S“ v názvu, z nichž některé jsou kompatibilní s “C“ typy. Např. AT89C2051 není podporován, ale AT89S2051 podporován je.
- Software předpokládá, že při programování AT89LP52 je pin POL součástky v logické 1. Pokud je POL v logické nule, je třeba v programu zaškrtnout volbu “Inverzní RESET“.

Zapojení procesoru PSoC od firmy Cypress

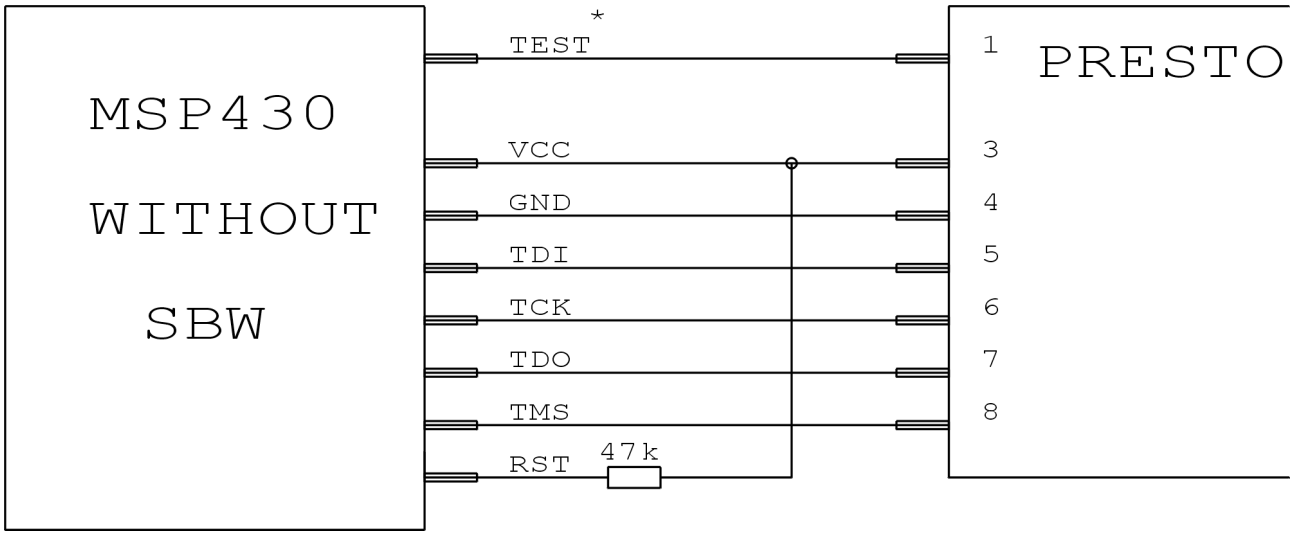


* Ne každý procesor PSoC má pin XRST.

Poznámka:

- Uživatel by měl nejprve vybrat způsob inicializace programovacího módu v okně „Nastavení programátoru PRESTO“. Součástky bez pinu XRST mohou vstoupit do programovacího režimu pouze použitím power-on resetu. Součástky s XRST pinem mohou použít obě metody, ale metoda inicializující signálem reset je lepší neboť je jí možné použít i s externím napájením.
- Položka “Algoritmus programování“ v okně “Nastavení programátoru PRESTO“ by měla být nastavena podle použitého napájecího napětí.

Zapojení procesoru MSP430, který nemá SBW (dvoudrátové) rozhraní

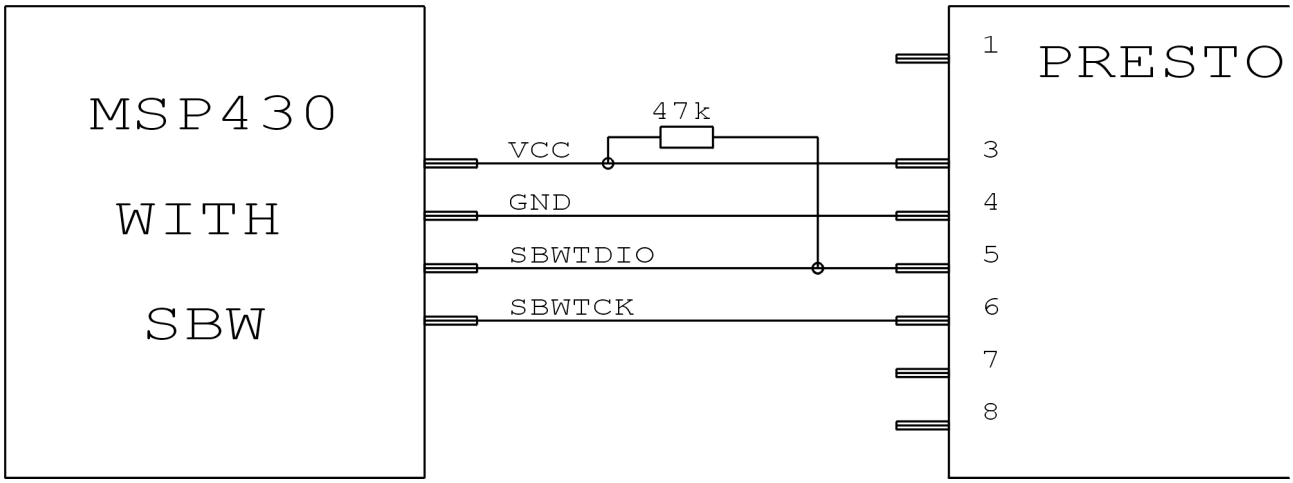


- Ne každý procesor MSP430 má TEST pin.

Poznámky:

- Toto schéma zapojení může být použito např. s procesory MSP430F1xx, MSP430F4xx, MSP430F21x1, ale ne s MSP430F20xx nebo MSP430F22xx.
- Pokud jsou kalibrační hodnoty oscilátoru uloženy v informační paměti a tato paměť nebude přeprogramována (smazána) během programování, procesor by měl být programován s vybranou volbou **Interní kalibrovaný RC oscilátor** v okně **Nastavení programátoru PRESTO**. V ostatních případech by měla být zvolena položka **Interní nekalibrovaný RC oscilátor**.
- Programátorem PRESTO není možné přepálit bezpečnostní pojistku na JTAG rozhraní.
- Pro programování samostatného procesoru s tímto rozhraním (bez SBW) je možné použít interní zdroj programátoru s konvertorem HPR3V3.

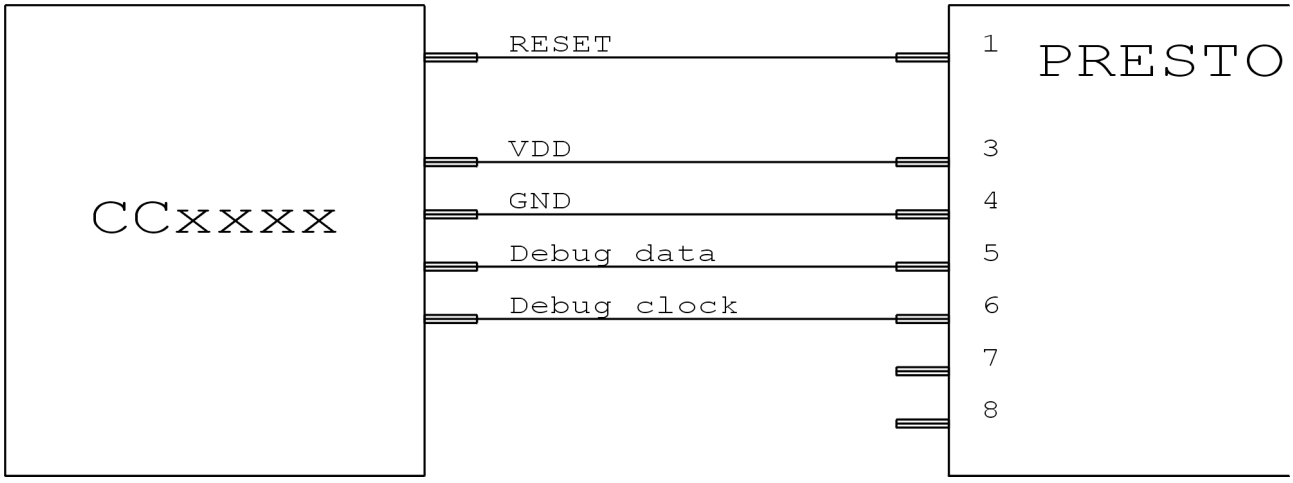
Zapojení procesoru CC430 nebo MSP430, který má SBW (dvoudrátové) rozhraní



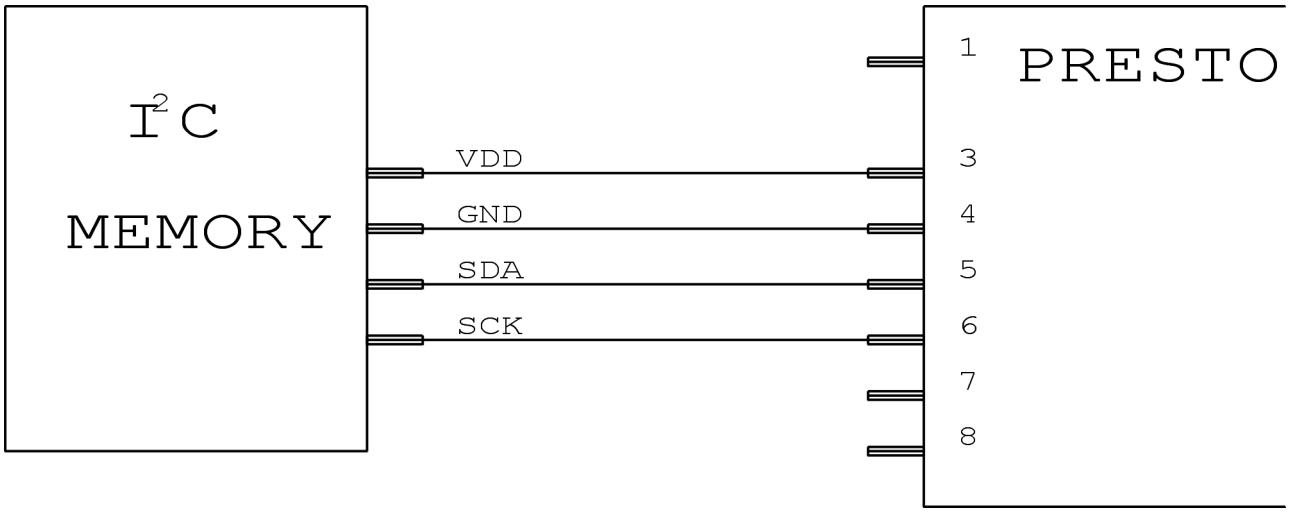
Poznámky:

- Procesor, který má SBW rozhraní, může být programován jen s použitím tohoto rozhraní. Je to např. MSP430F20xx, MSP430F22xx nebo MSP430F5xxx.
- Pokud jsou kalibrační hodnoty oscilátoru uloženy v informační paměti a tato paměť nebude přeprogramována (smazána) během programování, procesor by měl být programován s vybranou volbou **Interní kalibrovaný RC oscilátor** v okně **Nastavení programátoru PRESTO**. V ostatních případech by měla být zvolena položka **Interní nekalibrovaný RC oscilátor**. Pro MSP430F5xxx a CC430 se oscilátor nenastavuje.
- Programátorem PRESTO není možné přepálit bezpečnostní pojistku na JTAG rozhraní.
- Volba "Rychlost" v okně "Nastavení programátoru PRESTO" umožňuje zpomalit komunikaci v případě, že je na pinu RESET kondenzátor.
- Součástky s kalibračními konstantami oscilátoru uloženými v informační paměti mají možnost pomocí volby "Smazat Segment A" vybrat, zda se bude mazat i sektor A informační paměti.

Zapojení procesoru CCxxxx od TI (Chipcon)



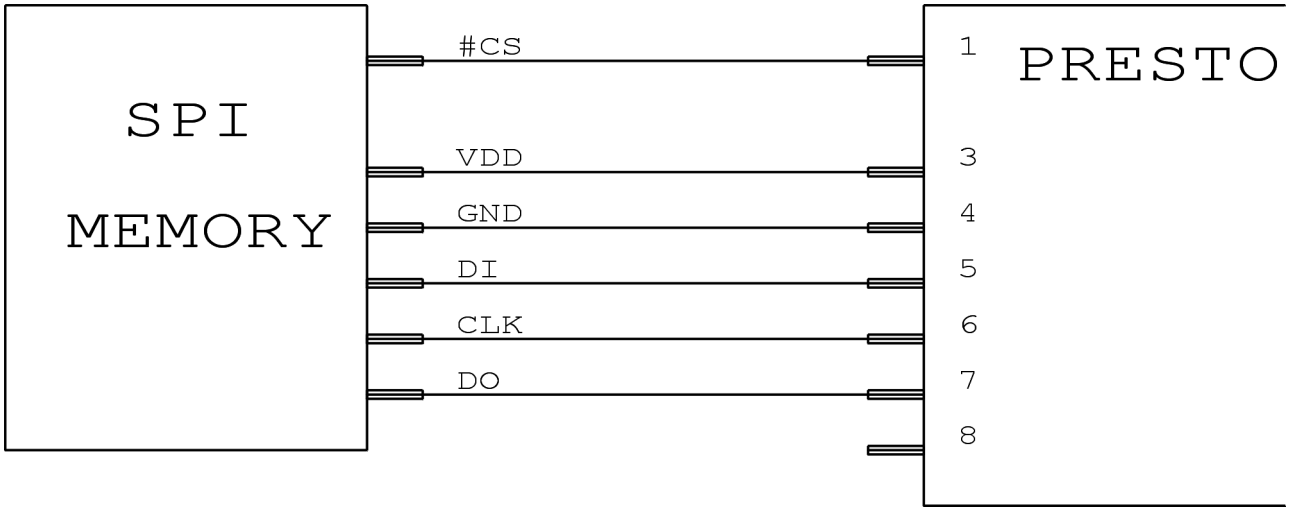
Připojení I²C paměti k PRESTU



Notes:

- Programátor používá na datovém vodiči (SDA) interní Pull-Up rezistor 2,2kΩ, když pracuje s I²C součástkou.
- Pokud je programovaná součástka 24LC(S)21A nebo 24LC(S)22A, její VCLK pin musí být v průběhu programování připojen na VDD.
- Paměti 34xx02 potřebují na pinu A0 vysoké napětí pro příkazy ochrany proti zápisu SWP a CSWP. Vysoké napětí je generováno na pinu VPP programátoru. Napětí z programátoru je 13V, ale vysoké napětí součástky by mělo být menší než 10V, uživatel musí upravit velikost tohoto napětí. Piny paměti A0, A1 a A2 musí být zapojeny manuálně podle zvoleného módu ochrany.

Připojení SPI paměti k PRESTU



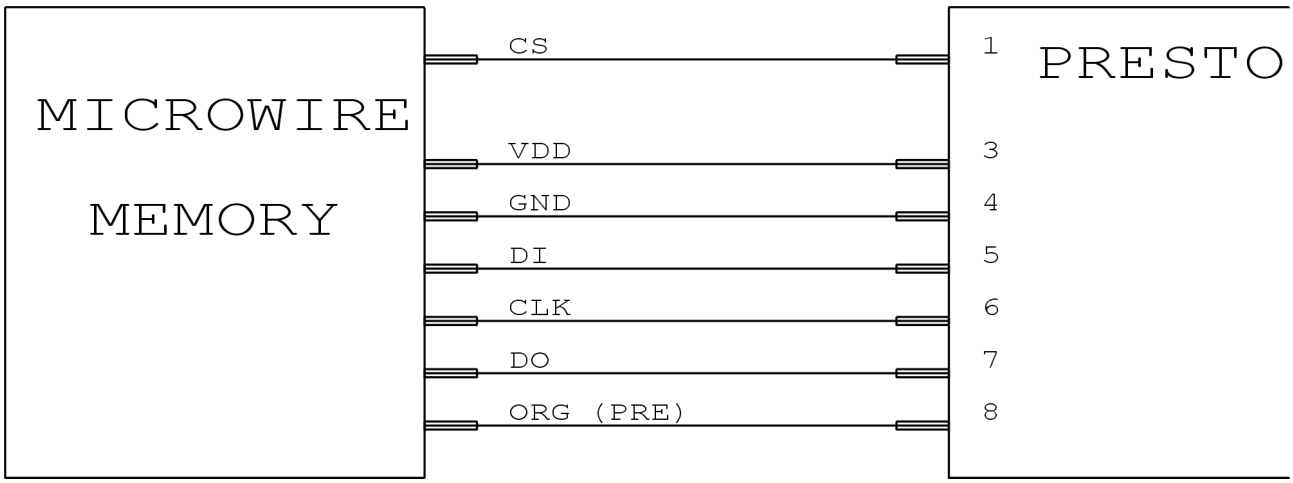
Různí výrobci označují piny SPI pamětí různými jmény. Některá označení jsou uvedena v tabulce níže:

Jméno na obrázku	Atmel, SST	ST
DI	SI	D
DO	SO	Q
CLK	SCK	C

Poznámky:

- **Programování SPI paměti v aplikaci** – Piny Write Enable a Hold mohou být v aplikaci připojeny na patřičnou logickou úroveň. Je třeba, aby všechny piny připojené současně s programátorem na piny programované paměti byly během programování nakonfigurovány jako digitální vstupy nebo odděleny nějakým multiplexorem. Programátor taktuje tyto paměti frekvencí asi 500 - 1000kHz, kapacity datových vodičů musí být dostatečně pomalé, aby umožnily tuto rychlost.
- 3,3V paměti je možné napájet z interního zdroje programátoru s konvertorem HPR3V3.

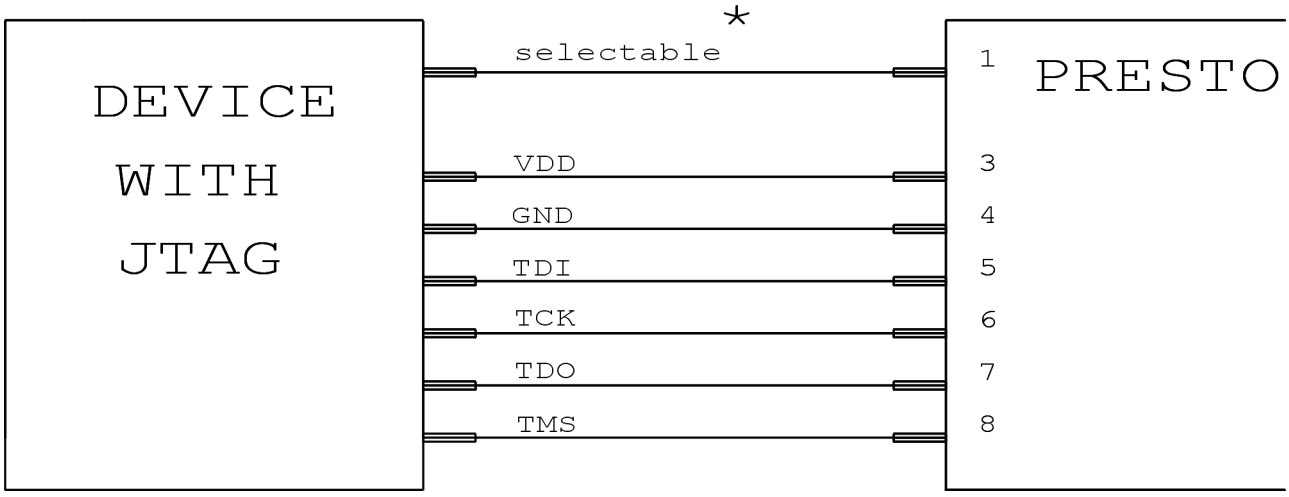
Připojení Microwire paměti k PRESTU



Poznámky:

- **LVP** pin (Pin8) vybírá organizaci paměti jako buď 8-bitů nebo 16-bitů na slovo. Uživatel vybere organizaci v programu UP a programátor PRESTO potom nastaví tento pin na příslušnou logickou úroveň. Pokud je tento pin paměti pevně zapojený v aplikaci na patřičnou logickou úroveň, LVP pin programátoru zůstane nezapojený. V případě použití s pamětí M93Sx6 je nutné pin LVP připojit na pin PRE součástky a slouží k výběru Protection registru.

Připojení součástky programované přes JTAG rozhraní k PRESTU



* Signál VPP se může chovat jako SCK nebo TRST popsaný v SVF souboru nebo může být definovaný uživatelem (log.1, log.0, třetí stav, různý během programování a po něm)

Pro AVR32 je třeba připojit pin P1 programátoru na RESET signál procesoru.

Poznámky:

- Napájecí napětí je vždy přivedeno z externího zdroje, protože součástky s JTAG rozhraním často používají jiné napájecí napětí než 5V.
- Přes JTAG rozhraní je možné programovat všechny součástky pro které je možné vytvořit soubor v SVF formátu. Program pro programování součástek s JTAG rozhraním je dostupný na http://www.asix.cz/ nebo na přiloženém disku CD-ROM.
- PRESTO v JTAG módu neposkytuje napájecí napětí do aplikace.
- Pokud jsou piny JTAG portu použity v aplikaci jako I/O piny, součástka musí být během programování držena v resetu. K tomu může být použit žlutý (VPP) vodič PRESTA, jehož funkci lze nastavit v menu programu JTAG Player.
- Rozhraní JTAG je také použito pro programování a debugování procesorů s jádrem ARM. Procesory ARM se programují programem **ARMINE**, který je ke stažení na webu. Pro více informací o programování součástek s jádrem ARM viz nápovědu programu ARMINE.
- Při programování součástek s nižším napájecím napětím než 2,7V, by měl být s PRESTEM použit konvertor HPR1V2.
- Procesory AVR32 se programují prostřednictvím rozhraní JTAG programem UP. Během programováni nesmí být součástka v resetu.
- Procesory ATxmega se programují prostřednictvím rozhraní JTAG programem UP, pin P1 programátoru zůstane nezapojený. Tyto součástky je možné použít s konvertorem HPR3V3 pro napájení z PRESTA.

1.6 Popis indikátorů a ovládacích prvků

Zelená LED (ON-LINE) - PRESTO je připojeno k počítači
Žlutá LED (ACTIVE) - Probíhá komunikace
Tlačítko (GO) - Spustí programování nebo jinou přednastavenou funkci

1.7 Technická specifikace

Varování: Při nedodržení zde specifikovaných parametrů může dojít ke zničení programátoru nebo připojeného počítače!

Maximální napětí na pinu V_{CC}	$U_{VCC\ MAX}$	7,5 V
Maximální napětí na ostatních pinech	$U_{IO\ MAX}$	5,5 V
Maximální proud odebíraný z V_{CC}	$I_{VCC\ MAX}$	100 mA
Maximální proud odebíraný z V_{PP}	$I_{VPP\ MAX}$	50 mA
Maximální proud odebíraný z ostatních pinů	$I_{IO\ MAX}$	4 mA
Napájecí napětí při napájení z aplikace	$U_{VCC\ IN}$	3,0 V až 5,0 V ±10 %
Napětí na V_{PP} v průběhu programování	U_{VPP}	5 V/13 V
Provozní teplota	T_{OP}	0 až 40 °C

Rozměry L x W x H	Přibližně 105 x 55 x 25 mm.
Maximální dovolená délka ICSP kabelu	150mm

2 Ostatní programátory

2.1 PICCOLO

Popis programátoru

PICCOLO je velmi levný vývojový programátor mikrokontrolérů Microchip PIC® s pamětí Flash v pouzdře s 18 vývody. Větší součástky (s 28 a 40 piny) lze programovat pomocí konektoru a kabelu ICSP (programování součástek osazených na desce plošných spojů). PICCOLO je ideální programátor zvláště pro začátečníky, studenty a amatéry. Je určen pro všechny, kteří se nehodlají zdržovat konstrukcí programátoru ať už podle vlastního návrhu, nebo převzatého z některé z publikací, ale chtějí se rovnou věnovat užitečné činnosti, a mít přitom jistotu, že při nákladech srovnatelných s materiálem budou mít programátor stoprocentně vyhovující programovacím specifikacím firmy Microchip (kategorie Development) s komfortním obslužným programem.

Podporované součástky

- všechny používané procesory PIC® s pamětí Flash, přímo v programátoru pouze typy v pouzdře s 18ti vývody

2.2 PICQUICK

Popis programátoru

PICQUICK je velmi rychlý, levný a spolehlivý vývojový programátor firmy ASIX pro mikrokontroléry PIC® a sériové paměti EEPROM (Microchip). Jednou z největších výhod programátoru PICQUICK je podpora všech řad a typů mikrokontrolérů PIC.

Zvolte si součástku, která nejlépe odpovídá Vaší aplikaci a můžete si být jisti, že PICQUICK ji bude podporovat. Navíc

je implementována podpora EEPROM pamětí Microchip (s rozhraním I2C a MicroWire). Tyto paměti jsou často používány k rozšíření paměti systému s mikrokontrolérem. PICQUICK byl navržen jako velmi flexibilní zařízení tak, aby bylo možné přidat podporu nových součástek. Od jeho uvedení na trh byly prakticky všechny nové součástky přidávány pouze pomocí upgrade softwaru.

Nejsou tedy zapotřebí upgrade firmware ani žádné nákladné zásahy do hardware, což uživateli přináší další cenové výhody ve srovnání s jinými programátory. Upgrade software je pro všechny uživatele k dispozici zdarma. Obsahuje podporu nových součástek, nové funkce a případné změny programovacích algoritmů, pokud jsou firmou Microchip vyžadovány.

PICQUICK přímo podporuje In-Circuit Serial Programming (ICSP) pomocí vyhrazeného konektoru a kabelu, který je v ceně jako standardní součást dodávky. Proudové omezení pro napájecí a programovací napětí minimalizuje riziko poškození programované součástky při chybě obsluhy.

Podporované součástky

- všechny typy v současné době používaných procesorů PIC®
- sériové paměti 93Cxx a 24Cxx

2.3 CAPR-PI

Podporované součástky

- Všechny používané procesory PIC® s pamětí Flash, které nemohou mít na pinu MCLR/Vpp výstupní I/O funkci.
  Omezení se týká procesorů např. PIC16F627/628.

2.4 PICCOLO Grande

Popis programátoru

PICCOLO GRANDE je velmi levný vývojový programátor mikrokontrolérů Microchip PIC® s pamětí Flash v pouzdrech s 18, 28 a 40 vývody. Programovat lze jak neosazené součástky (patice nejsou typu ZIF), tak i zapájené na plošném spoji (ICSP - In-Circuit Serial Programming).

PICCOLO GRANDE je ideální programátor zvláště pro začátečníky, studenty a amatéry. Je určen pro všechny, kteří se nehodlají zdržovat konstrukcí programátoru ať už podle vlastního návrhu, nebo převzatého z některé z publikací, ale chtějí se rovnou věnovat užitečné činnosti, a mít přitom jistotu, že při srovnatelných materiálových nákladech budou mít programátor stoprocentně vyhovující programovacím specifikacím firmy Microchip (kategorie Development) a s komfortním obslužným programem.

Podporované součástky

- všechny používané procesory PIC® s pamětí Flash, přímo v programátoru pouze typy v pouzdře s 18, 28 a 40 vývody

2.5 PVK Pro

Popis programátoru

PVKPro je vývojový a výukový kit s programátorem pro PIC16F84A na jedné desce. Je určen pro studijní a výukové účely a pro úvodní seznámení s problematikou PIC - práce v reálném čase, I/O, multiplex displeje, snímání tlačítek apod.

Deska obsahuje vše, co je potřeba pro typické aplikace mikrokontroléru:

- obvody napájení
- oscilátor
- resetovací obvod
- čtyřmístný sedmisegmentový displej LED
- 8 LED diod
- 8 tlačítek

Všechny uživatelské piny je možno pomocí DIP přepínačů propojit s periferiemi na desce nebo externě - signály procesoru jsou vyvedeny na konektor. Deska se připojuje k počítači prostřednictvím paralelního portu.

Podporované součástky

- PIC16F83 / 84 / 84A
- s omezením PIC16F627 / 628
  Programátor PVK-Pro nemůže vyhovovat programovacím specifikacím procesorů, u kterých může pin MCLR pracovat jako I/O.

Nastavit pin -MCLR jako I/O je pro použití v této desce bezúčelné, proto nemožnost programovat takto nastavené procesory není příliš omezující.

3.1 HPR3V3

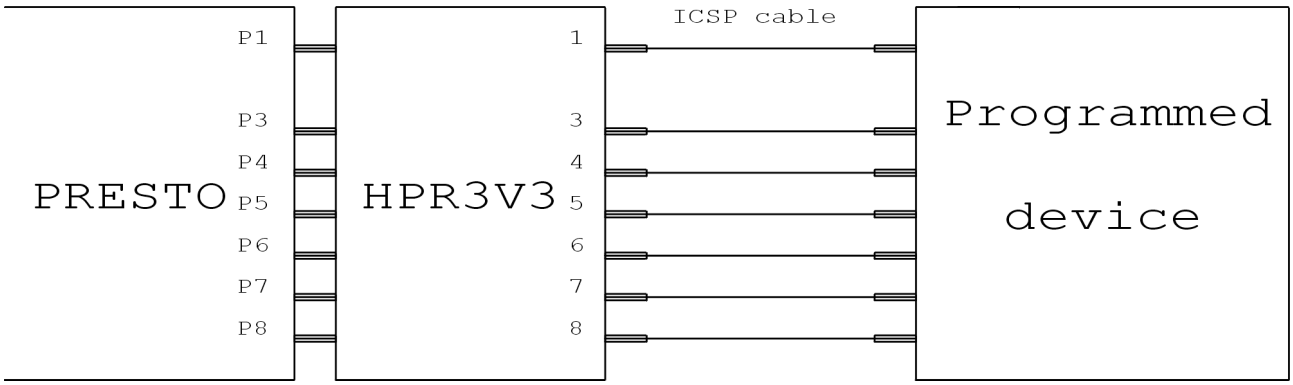
HPR3V3 je volitelné příslušenství k programátoru PRESTO pro programování 3,3 V součástek mimo aplikaci, na příklad DataFlash pamětí. Programátor PRESTO může napájet programovanou součástku 5 V z interního zdroje, některé součástky však vyžadují 3,3 V napájecí napětí a 3,3 V úrovně logických signálů. V takovém případě musí být s PRESTEM použit konvertor úrovní HPR3V3 nebo externí 3,3 V napájení.

Použití

Použití HPR3V3 je velmi jednoduché. Konvertor zapojte přímo na piny PRESTA (**V žádném případě ho nezapojujte na konektor u programované součástky!**). Pin 2 je použitý jako klíč, konvertor tedy není možné zapojit špatně. Nyní ICSP kablíkem propojte programovanou součástku a piny HPR3V3. Zapojení pinů programované součástky je stejné jako by bylo zapojení vzhledem k PRESTU. Běžné schéma propojení PRESTA s HPR3V3 a programovanou součástkou je uvedeno níže. Viz příklady připojení programovaných součástek k PRESTU.

Poznámky:

- Piny HPR3V3 jsou jednosměrné, konvertor může být tedy použit pro procesory AVR, SPI Flash paměti nebo procesory MSP430 bez SBW rozhraní, ale nemůže být použit pro procesory PIC nebo MSP430 se SBW rozhraním.
- **Nikdy nepřipojujte externí napětí k výstupním 3,3 V napájecím pinům!**



3.2 HPR1V2

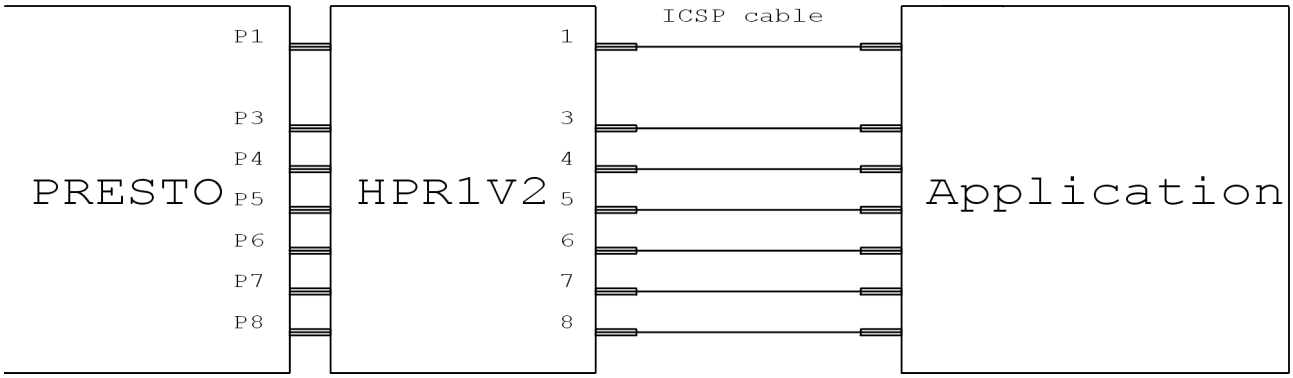
HPR1V2 je volitelné příslušenství k programátoru PRESTO pro programování součástek s napájecím napětím a logickými úrovněmi mezi 1,2 a 3,3 V jako např. Xilinx CoolRunner-II. PRESTO může programovat součástky se signálovými úrovněmi mezi 3 a 5 V ± 10 %, ale někdy je potřeba programovat součástky s nižšími úrovněmi signálů. V tom případě musí být použit konvertor HPR1V2. Konvertor se napájí externím napětím z aplikace, nemůže být napájen z interního zdroje programátoru.

Použití

Připojte HPR1V2 přímo k pinům PRESTA. (**V žádném případě nepřipojujte destičku konvertoru na konektor v aplikaci!**). Pin2 je použitý jako klíč, takže není možné připojit konvertor chybně. Nyní s použitím ICSP kablíku propojte HPR1V2 a programovanou součástku. Zapojení pinů programované součástky je stejné jako by bylo zapojení součástky vzhledem k PRESTU. Běžné propojení PRESTA a HPR1V2 s programovanou součástkou je na obrázku níže.

Poznámky:

- Piny HPR1V2 jsou jednosměrné, konvertor může být použit např. se součástkami programovatelnými přes rozhraní JTAG, není však vhodný pro použití se součástkami využívajícími obousměrnou komunikaci jako např. procesory PIC.
- HPR1V2 musí být napájen externě z aplikace.
- **Nikdy nezapínejte interní napájecí napětí z programátoru, pokud je připojen konvertor HPR1V2!**



4 Program UP

UP je řídicí software pro programátory ASIX. Program nabízí mnoho pokročilých funkcí a umožňuje ovládání programovacího procesu jak z prostředí programovacího software, tak i vzdáleně z příkazové řádky, pomocí zpráv Windows a knihovny DLL. Program je možné použít pod Windows 95/98/ME/NT/2K/XP.

4.1 Instalace programu UP

Instalace je velmi jednoduchá. Instalační program lze nalézt buď na disku CD-ROM dodaném společně s programátorem nebo na www.asix.cz . Spusťte instalátor (UP_xxx_CZ.EXE, za xxx dosaďte číslo verze), není nutné zavírat ostatní aplikace. Instalace trvá jen několik sekund a vyžaduje jen několikrát stisknout klávesu Enter. Během instalace se neprovádí žádná modifikace operačního systému a není tedy nutné počítač restartovat a program může být ihned po instalaci spuštěn (např. kliknutím na příslušnou ikonu). Při prvním spuštění se program zeptá na jazyk, který se má použít (Angličtina/Čeština), programátor (např. PRESTO) a port kam je programátor připojený.

V případě potřeby může být program odstraněn běžným způsobem použitím ikony v ovládacích panelech nebo ručně smazáním příslušného adresáře a zástupců. Před instalací nové verze není potřeba odstraňovat předchozí verzi programu. Je doporučeno používat vždy nejnovější verzi programu.

4.2 Programování součástky

V programu UP je doporučené používat projekty, kde jsou uložena nastavení programu.

Nový projekt může být vytvořen kliknutím na položku menu **Soubor→Nový projekt**, existující projekt může být otevřen kliknutím na **Soubor→Otevřít projekt**.

Před programováním je potřeba vybrat programátor a jméno programované součástky, což může být provedeno v menu **Nastavení→Výběr zařízení a portu** a **Součástka→Výběr součástky** nebo dvojklikem na jméno vybraného programátoru a součástky, která jsou zobrazena v pravém horním rohu okna programu. Aby chování programu

vyhovovalo potřebám uživatele, je vhodné přizpůsobit jeho nastavení v menu **Nastavení→Nastavení programu**. Detailní popis může být nalezen v kapitole Menu programu UP.

Kdykoliv je vybrán programátor PRESTO, je zobrazeno také okno Nastavení programátoru PRESTO, kde je možné nastavit používaný zdroj napětí a některé další důležité volby.

Programování: Použijte položku **Soubor→Otevřít** k otevření HEX souboru, který chcete programovat. Vlastnosti programované součástky (pojistky) mohou být nastaveny v okně **Konfigurace**. Změny mohou být uloženy vybráním **Soubor→Uložit**. (Viz poznámky k programování procesorů AVR a PIC)

Programování začne po kliknutí na **Součástka→Programovat** nebo po kliknutí na tlačítko **Programovat**. Programátor udělá následující operace: Smaže součástku, zkontroluje smazání, naprogramuje a zkontroluje naprogramování celé součástky. Před programováním je zkontrolováno Device ID a Code/Data protection bity.

Pokud je potřeba programovat pouze část procesoru, může to být provedeno vybráním příslušné položky v menu **Součástka→Programovat** nebo kliknutím na šipku u tlačítka **Programovat** na liště tlačítek. Detailní popis může být nalezen v kapitole Menu programu UP.

Poznámka: Pokud je v nastavení programu vybrána (zaškrtnuta) položka "**Načíst hex soubor vždy znovu před programováním**", software znovu načte HEX soubor po stisku tlačítka **Programovat**. Pokud nejsou pojistky uloženy v HEX souboru, uživatel musí buď zrušit tuto volbu nebo zrušit volbu "**Inicializovat konfigurační paměť před čtením ze souboru**" v nastavení programu na záložce **Soubory.** (Pokud je tato položka zaškrtnutá a v HEX souboru nejsou uložené pojistky, bude konfigurační paměť při načtení souboru uvedena do defaultního stavu).

4.3 Nastavení funkce tlačítka GO

Programátor PRESTO obsahuje tlačítko GO, které umožňuje uživateli spouštět programování bez potřeby myši nebo klávesnice. Během programování je stav programátoru indikován dvěma LED indikátory – zelená LED (ON-LINE) indikuje stav připojení programátoru k USB a žlutá LED (ACTIVE) indikuje zda PRESTO právě pracuje (programuje, čte, ...)

Funkce tlačítka GO může být nastavena podle potřeb uživatele v menu **Nastavení→Klávesové zkratky** pod položkou **Tlačítko GO**.

Program UP musí být vždy spuštěn, pokud chce uživatel používat tlačítko GO, ale může být minimalizovaný.

4.4 Sériová výroba

Menu:**Součástka→Programovat→Sériová výroba**

Tato funkce je dostupná také na panelu funkcí pod tlačítkem **Programovat**.

Programování může být spuštěno z dialogu "**Sériová výroba**" kliknutím na tlačítko **Programovat**. Funkce tohoto tlačítka je ekvivalentní použití "**Naprogramovat vše**" nebo "**Naprogramovat vše kromě paměti EEPROM**" v závislosti na stavu volby "**Neprogramovat datovou EEPROM**".

V tomto dialogu je zobrazen čítač naprogramovaných součástek. Podle nastavených vlastností programu může být čítač zobrazen také na stavovém panelu. Čítač zobrazuje počet naprogramovaných součástek jak v módu sériové výroby tak ve standardním módu programu.

4.5 Sériová čísla

Funkce "**Sériová čísla**" naprogramuje sériové číslo nebo jinou sekvenci znaků na vybranou paměťovou pozici.

Sériová čísla mohou být:

- *počítaná*
  Počítaná sériová čísla mohou být vkládána vždy pouze na jedno zvolené místo v součástce, např. do paměti programu, datové EEPROM nebo do ID pozic. Sériové číslo je vždy chápáno jako číslo v desítkové nebo šestnáctkové soustavě a může být kódováno jako 4-bitová kombinace (po jednom až čtyřech do jednoho slova) nebo ASCII znak (jeden nebo dva do slova), případně při používání paměti programu lze zvolit zarovnání do RETLW xx instrukce.
- *vkládaná ze souboru*
  Jedno sériové číslo může být rozmístěné do více částí součástky. (např. vlastní sériové číslo přímo v programu, adresa zařízení v datové EEPROM paměti a znovu sériové číslo uložené v ID pozicích pro možnost přečtení sériového čísla ze zamknuté součástky)

Poznámka: Jedním slovem je míněna jedna pozice paměti.

V "**Nastavení/Nastavení programu.../Sériová čísla**" je možné zvolit soubor pro logování programovaných sériových čísel. V případě počítaných sériových čísel se do souboru zapisují přímo tato čísla, v případě čtení sériových čísel ze souboru se do souboru zapisují labely sériových čísel.

Formát souboru se sériovými čísly

Soubor je textový a velmi snadno vytvořitelný jiným programem. Doporučená přípona souboru je `*.SN` nebo `*.TXT`.

- *Bílé znaky* jsou mezera, tabelátor, konec řádku (CR+LF).
- *Komentář* jakýkoli řetězec, který neobsahuje dvojtečku '**:**' nebo středník '**;**'
- *Záznam sériového čísla* má tvar
  `komentář label: datový záznam, datový záznam, ..., datový záznam;`
- *Label* je řetězec identifikující sériové číslo. Tento řetězec je **povinný**. Label nesmí obsahovat bílé znaky, dvojtečku, středník.
- *Datový záznam* je složen z adresy a datových položek obsažených postupně za touto adresou.
  Každá položka může být zapsána v hexadecimálním tvaru (např. 2100), nebo může být explicitně zadána číselná soustava ve které je číslo zadáno (např. b'10101010' znamená totéž co h'AA', d'170' nebo jen samotné AA) nebo i jako samotný ASCII znak (pak d'65' a 'A' znamená totéž)
  např. `2100 05 55 54` znamená do datové EEPROM paměti uložit na adresy 00 až 02 data 05h, 55h, 54h.

  Paměť kam se uloží sériové číslo může být také specifikována slovem CODE. nebo PROG. nebo P. pro programovou paměť nebo DATA. nebo EE. nebo E. pro datovou paměť nebo ID. nebo I. pro ID pozici. Tato slova jsou následována adresou ve specifikované paměti.
  např. `EE.00 05 55 54` znamená do datové EEPROM paměti uložit na adresy 00 až 02 data 05h, 55h, 54h
  Poznámky:
  Pro konfigurační paměť není žádný specifikátor, nemělo by to ani žádný smysl
  dsPIC - zadává se adresa 24-bitového slova pro všechny adresy (tzn. interní dsPIC adresa 24h je zde 12h), u EEPROM se zadává adresa 16-bitového slova, tzn. tak jak jdou v procesoru jedna po druhé.
  Samostatné paměti (I2C, SPI) mají jen paměť CODE, v případě specifikace neexistující paměti bude hlášena chyba.

- *Komentář* je nepovinný.
- V případě, že celý záznam sériového čísla neobsahuje dvojtečku, je ignorován (je brán pouze jako komentář)
  `toto je pouhy komentar;`

Příklad souboru se sériovými čísly:

komentar na zacatek;

```
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;
poznamka

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 Použití programu UP z příkazové řádky

Program sám ošetří, aby byl spuštěn vždy pouze v jedné instanci. Pokud je spuštěna druhá instance programu, parametry z příkazové řádky se předají první instanci a ta je provede.

Přehled parametrů

up.exe [{**/ask** | **/q**}] [{**/e** *soubor_s_eeprom.hex* | [**/noe**]}] [{**/p** |[**/pdiff**]| [**/o**]} *soubor.hex* | *soubor.ppr*] [**/part** *jméno_součástky*] [**/eeonly**] [**/erase**] [**/w**[**nd**]*up_window_class*] [**/cfg**] [**/devid**] [**/blank**] [**/verify** *soubor*] [**/s** *SN_programátoru*]

Legenda:
Text, který je **tučně** se píše přímo na příkazovou řádku tak, jak je napsán.
Text, který je *kurzívou* je třeba nahradit odpovídajícím parametrem. Např. *jméno souboru* se nahradí skutečným jménem souboru, který má být otevřen.

Text uzavřený ve složených závorkách oddělený znakem | označuje výběr pouze jedné z uvedených možností, tj. { **A** | **B** } znamená "zvolte buď A nebo B".
Text v [hranatých závorkách] označuje nepovinný parametr - může být uveden, ale také nemusí.

- **/ask** Používá se ve spojení s **/p**. Program se před programováním součástky vždy zeptá, jestli má pokračovat, i když je v nastavení programu požadováno, aby se neptal. Zároveň je v dialogu oznámen i vybraný typ součástky.
- **/q; /quiet** Quiet mód. Program se na nic neptá, v případě potřeby zobrazení dialogu skončí chybou. Viz návratové kódy programu.
- **/e soubor** EEPROM soubor. Zadání případného souboru s daty EEPROM paměti. Pokud má jméno souboru v těle mezery, je třeba jméno souboru uzavřít uvozovkami.
- **/noe** No EEPROM. Přeskočí programování EEPROM. Pokud je tento parametr použit při programování procesoru MSP430, je vynecháno programování a mazání informační paměti.
- **/p soubor** Programovat. Zadaný soubor se naprogramuje. Pokud jméno souboru obsahuje mezery, je třeba jej uzavřít mezi uvozovky.
- **/pdiff soubor** - Programovat rozdílově. Zadaný soubor se naprogramuje. Pokud jméno souboru obsahuje mezery, je třeba jej uzavřít mezi uvozovky.
- **/o soubor** Otevřít. Zadaný soubor je otevřen, nepovinný parametr. Pokud jméno souboru obsahuje mezery, je třeba jej uzavřít mezi uvozovky.
- **/eeonly** - Provede zvolenou operaci jen s pamětí EEPROM, u MSP430 jen s Informační pamětí.
- **/part jméno** Vybere v UPu zvolenou součástku.

- **/erase** Smaže součástku.
- **/wnd jméno třídy** Jiné jméno třídy okna. Pomocí tohoto parametru lze spustit program UP najednou více než jednou. Každá nová instance spouštěného programu UP musí mít jiné jméno třídy.
- **/cfg** Pokud je tento parametr použit společně s parametrem **/p**, naprogramuje se pouze konfigurační paměť. To je užitečné např. při programování procesorů AVR, které je možné nejprve přepnout na rychlejší oscilátor a následně naprogramovat mnohem rychleji.
- **/devid** Pokud je tento parametr použit společně s parametrem **/p**, pouze se zkontroluje Device ID součástky.
- /**blank** Provede kontrolu smazání součástky a podle výsledku vrátí chybový kód.
- /**verify soubor** Provede verifikaci součástky.
- /**s SN_programátoru** Umožňuje vybrat programátor podle sériového čísla. Zadává se v hexadecimálním vyjádření nebo převedené do desítkové soustavy (0x016709 nebo 91913).

Při několika rozpracovaných projektech se může stát, že program je nastaven na jinou součástku (nebo i programátor), než uživatel předpokládal. Pak je vhodné používat projektové soubory, (`.PPR`), ve kterých je uloženo veškeré nastavení programu a cesta k pracovnímu souboru.

Poznámka: V instalačním adresáři programu UP jsou ukázkové dávkové soubory “read_avr_eeprom.bat“ a “set_idle_power_1.bat“ ukazující použití parametrů příkazové řádky.

Otevření souboru

up.exe *jméno_souboru*

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

Programování součástky

`up.exe /p` *`jméno_souboru`*

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

Návratové kódy programu

0 - Vše proběhlo bez problémů.
1 - Chyba souboru. Např. soubor nenalezen, soubor má špatný formát.
2 - Chyba zařízení. Test komunikace byl neúspěšný, chyba při komunikaci.
3 - Chyba při přípravě programování. Nelze smazat součástku, apod.
4 - Chyba při programování.
5 - Chyba při verifikaci.
6 - Programování neproběhlo z důvodu potřeby interakce s uživatelem.
7 - Chyba Device ID.

Poznámka: V dávkových souborech je možné získat návratovou hodnotu programu z proměnné %errorlevel%. Viz ukázkový dávkový soubor “read_avr_eeprom.bat“ v instalačním adresáři programu UP.

4.7 Ovládání programu UP pomocí zpráv Windows

Program UP může být ovládaný pomocí zpráv systému Windows. Spuštěná instance programu UP vykoná požadovanou akci okamžitě po přijetí zprávy.

Zprávy musí být posílány oknu třídy `"up v1.x"`. Typ zprávy je vždy WM_USER.

Příkaz se identifikuje podle wParam, parametry podle lParam.

Přehled příkazů

```
(* 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
 *)
```

Příklad

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

Použití knihovny UP_DLL.DLL

Poznámka: Knihovna UP_DLL.DLL komunikuje s programem UP, takže UP musí při použití této knihovny běžet. **UP_DLL nemůže pracovat samostatně.**

Pomocí knihovny up_dll.dll lze se spuštěným programem UP vyměňovat řetězce.

```
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.
```

Popis jmen a hodnot jednotlivých nastavení je v příloze.

4.8 Spuštění více než jednoho programu UP

Pokud uživatel potřebuje připojit více programátorů k jednomu počítači, pro každý programátor musí být spuštěn samostatný program UP.

Program UP při běžném používání může být spuštěn pouze jednou: každé další spuštění programu pouze pošle parametry z příkazové řádky nebo nějakým jiným způsobem zviditelní předchozí spuštěný program UP.
Program UP lze spustit vícekrát pod jiným názvem třídy okna - ty programy, které budou mít stejný název třídy okna spolu budou komunikovat. Jméno třídy okna se programu sdělí pomocí příkazové řádky parametrem **/w**. Parametry programu při spouštění z příkazové řádky jsou popsány v samostatné kapitole.

Příklad

První instance programu UP může být spuštěna běžným způsobem ze start menu.
Další instance může být spuštěna z příkazové řádky např jako `up /w "another up"`

4.9 Přístup více programů k jednomu programátoru

K jednomu programátoru může přistupovat vždy nejvýše jeden program (v případě programátoru PRESTO jsou k dispozici i různé utility). U programátorů připojených k paralelnímu portu počítače je pouze na uživateli, aby ochránil dva programy od společného přístupu k jednomu programátoru. V případě programátorů připojených k rozhraní USB se o přístupová práva stará operační systém. V případě spuštěného programu UP operační systém nedovolí jinému programu přístup ke zvolenému programátoru. Program UP nedovoluje jinému software přístupu k programátoru PRESTO , např. proto, že neustále kontroluje stav tlačítka a stav napětí na napájecím pinu.
Programu UP lze přístup k programátoru zakázat a uvolnit programátor pro jinou aplikaci pomocí dialogu **výběr programátoru**. V menu zvolte výběr programátoru. Po dobu, co je dialog zobrazen, může jiný program přistupovat k programátoru. Při zrušení dialogu **nedojde** ke ztracení rozpracovaných dat.

4.10 Formát souborů Intel HEX používaných programem UP

Program UP používá soubory Intel HEX ke čtení a ukládání dat (běžná přípona takového souboru je .HEX).

Podporované varianty HEX souboru

- "obyčejný", někdy též Intel 8-bit HEX File, MPASMWIN generuje tento soubor při parametru **INHX8M**
- "rozšířený", někdy též Intel 32-bit HEX File, MPASMWIN generuje tento soubor při parametru **INHX32**

Popis formátu Intel HEX souboru

Intel HEX jsou textové soubory, které se skládají z řádků.
Každý řádek má následující strukturu: `:LLAAAATTDDDD...CC`

- **:** Tímto znakem (dvojtečka, 0x3A) musí začínat každý řádek souboru.
- **LL** Délka záznamu (počet políček DD).
- **AAAA** Adresa prvního byte záznamu.
- **TT** Typ záznamu. Typy mohou být:
  - 00 - Datový záznam.
  - 01 - Záznam Konec souboru. Každý soubor musí končit tímto záznamem.
  - 02 - Rozšířená segmentová adresa. (pouze 32-bit HEX)
  - 04 - Rozšířená lineární adresa. (pouze 32-bit HEX)

  Existují i jiné typy, 03 a 05, které program UP při načítání ignoruje a při ukládání souboru nepoužívá.
- **DD** Data záznamu. Počet bytů musí být přesně LL.
- **CC** Kontrolní součet. Kontrolní součet je počítán jako dvojkový doplněk k součtu všech hodnot na řádku.

Datový záznam

Jako příklad poslouží řádek s uloženou konfigurační pamětí 14-bitové součástky.
`:02400E00413F30`

- Délka záznamu: **02** - Velikost konfigurační paměti je jedno slovo = 14 bit = 2 byte (zarovnáno na celé byty)
- Adresa záznamu: **400E** - Adresa konfigurační paměti je slovo 2007h, adresováno po bytech tedy 400Eh
- Typ záznamu: **00** - Datový záznam
- Data záznamu: **413F** - Konfigurační slovo je 3F41h
- Kontrolní součet: **30** = 02 + 40 + 0E + 00 + 41 + 3F = xxD0; neg D0 = 30

Konec souboru

Jedinou možnou variantou řádku *Konec souboru* je:
`:00000001FF`

Rozšířená lineární adresa

Tento řádek obsahují pouze soubory, které potřebují adresovat více než 64 kB adresového prostoru. Na příklad procesory rodiny PIC18F mají uloženou konfigurační paměť na adrese 0x 30 00 00 00.

Pokud je třeba použít tuto adresu, je nutné do HEX souboru vložit řádek s rozšířenou lineární adresou, který obsahuje horních 16 bitů adresy. Dolních 16 bitů je načteno z řádku s datovým záznamem.
`:020000040030CA`
Tímto řádkem se vybírá konfigurační paměť u součástek rodiny PIC18F.

U rozšířených segmentových záznamů se udává segment, tedy bity 19-4 adresy (segment), které se pak přičítají k adresám z datových záznamů (offset).

Ukládání typu součástky do .HEX souboru

Velmi často se stává, že dojde k záměně mezi vybraným typem součástky a typem součástky, pro kterou byl Intel HEX soubor uložen. Proto program UP obsahuje funkci ukládání typu součástky do souboru.
Program zapíše za konec souboru ještě řádek `#PART=.....`. Drtivá většina programů pracujících s Intel HEX soubory takovýto řádek ignoruje, avšak takovýto soubor nelze považovat za vyhovující formátu Intel HEX.

4.11 Podpora kalibrační paměti

Práce s kalibrační pamětí při mazání součástky v UV mazačce

Před mazáním součástky je většinou potřeba si zaznamenat kalibrační informaci. K tomuto účelu je možné použít funkce "Uložit kalibrační informaci..." a "Načíst kalibrační informaci ...".

Menu: Soubor -> Uložení kalibrační informace...
Menu: Soubor -> Načtení kalibrační informace...

Program obsahuje funkci pro kontrolu správného smazání součástky. Při použití této funkce program zobrazí informace z kalibrační paměti.

Práce s kalibrační pamětí u součástek s pamětí flash

Při smazání se obsah kalibrační paměti **zachovává**.

Pokud z nějakého důvodu chcete smazat kalibrační paměť, lze toto provést funkcí "Smazat vše, i kalibrační paměť" (Součástka -> Smazání -> Smazat vše, i kalibrační paměť).

Upozornění: *Nové Flash součástky s kalibrační pamětí (např. PIC12F629) obsahují i tzv. bandgap bity, které jsou též součástí kalibrace součástky. Tyto bity se vyskytují v konfiguračním slově a při funkci "Smazat i kalibrační paměť" se také smažou!*

4.12 Menu programu UP

Menu je součástí téměř každého programu pro Windows. Promocí menu lze vybírat funkce, které program nabízí. Akce menu mohou být vyvolány kliknutím myší na příslušnou položku menu nebo pomocí klávesnice stisknutím klávesy <ALT> společně s klávesovou zkratkou zvýrazněnou v menu.

Menu je rozděleno do následujících kategorií:

- Soubor
- Úpravy
- Zobrazit
- Součástka
- Nastavení
- Nápověda

- Okno Nastavení programátoru PRESTO
- Okna hexeditorů

Menu Soubor

Soubor → Nový

Klávesová zkratka: Ctrl+N

Založí nový prázdný soubor. Pokud právě používaný soubor nebyl uložen, program nejprve nabídne jeho uložení.

Soubor → Otevřít...

Klávesová zkratka: Ctrl+O

Pomocí standardního dialogového okna Windows otevře existující soubor na disku. Podporované soubory viz popis formátu Intel HEX souborů. Soubory s příponou HEX a A43 jsou načítány jako HEX, ostatní jako BIN.

Soubor → Načíst soubor znovu...

Klávesová zkratka: Ctrl+R

Opětovně načte právě otevřený soubor z disku. Funkci je vhodné použít, pokud víte, že soubor na disku byl změněn a chcete tyto změny načíst do programu.

Pokud používáte nastavení "Kontrolovat změny v hex souboru" (viz nastavení programu), program na změnu otevřeného souboru upozorní sám a nabídne jeho opětovné načtení.

Soubor → Uložit

Klávesová zkratka: Ctrl+S

Uloží soubor na disk. Pokud chcete uložit soubor pod jiným názvem, než pod jakým byl otevřen, použijte funkci Uložit soubor jako.... Program může při ukládání přeskakovat nevyužité oblasti paměti a také může některé zvolené oblasti neukládat, viz nastavení programu.

Soubor → Uložit jako...

Pomocí standardního dialogového okna Windows uloží otevřený soubor na disk pod novým jménem. Program může při ukládání přeskakovat nevyužité oblasti paměti, a také může některé zvolené oblasti neukládat, viz nastavení programu.

Soubor → Import datové paměti z hex...

Pomocí standardního dialogového okna Windows umožní přečíst obsah datové EEPROM paměti z jiného souboru. Tento soubor, bez ohledu na jeho obsah, je čtený od adresy nula, jako kdyby obsahoval pouze datovou EEPROM paměť. Soubor normálně vygenerovaný překladačem tedy takto nelze korektně načíst.
Tuto funkci program obsahuje pouze z důvodů kompatibility se starším software, který ukládal obsah datové EEPROM paměti do vedlejšího souboru. Z dnešního pohledu je tato funkce zbytečná, protože obsah všech pamětí se má (dle doporučení firmy Microchip) ukládat pouze do jediného souboru.

Soubor → Otevřít hex soubor s datovou pamětí automaticky

Pokud je zvolena tato volba, program UP současně s načítáním HEX souboru pro paměť programu automaticky načte HEX soubor pro datovou paměť. Tato volba je aktivní pouze pokud je načtený samostatný soubor pro datovou paměť.

Soubor → Nový projekt

Klávesová zkratka: Shift+Ctrl+N

Funkce vytvoří nový projekt.
Používání projektových souborů je vhodné zejména pokud často střídáte programování několika typů součástek, nebo používáte několik různých programátorů. Projektový soubor obsahuje všechna tato nastavení a umožňuje tak jejich hromadné načtení.

Soubor → Otevřít projekt

Klávesová zkratka: Shift+Ctrl+O

Pomocí standardního dialogového okna Windows otevře již existující projekt z disku. Pokud s projektem byl otevřený některý další soubor, je tento načten také.

Soubor → Otevřít další soubor

Importuje další HEX nebo BIN soubor s volitelným offsetem. Tato funkce je užitečná, pokud uživatel potřebuje načíst do paměti součástky další soubor. Soubory s příponou HEX a A43 jsou načítány jako HEX, ostatní jako BIN.

Soubor → Uložit projekt

Klávesová zkratka: Shift+Ctrl+S

Uloží projekt pomocí standardního dialogového okna Windows pod novým názvem. Ukládání samotného projektu pod stejným názvem se provádí automaticky, stejně jako např. ukládání nastavení programu.

Soubor → Zavřít projekt

Klávesová zkratka: Shift+Ctrl+W

Ukončí práci s aktuálně otevřeným projektem, uloží projektový soubor na disk a program se vrátí do stavu, ve kterém byl před vytvořením nového projektu.

Soubor → Načtení kalibrační informace...

Pomocí standardního dialogového okna Windows otevře soubor s kalibrační informací a načte tuto informaci do paměti.

Soubor → Uložení kalibrační informace...

Pomocí standardního dialogového okna Windows program vytvoří soubor s kalibrační informací součástky, kterou přečte ze součástky vložené v programátoru. Tuto kalibrační informaci lze po smazání součástky znovu nahrát pomocí příkazu načtení kalibrační informace.

Pro více informací o podpoře programu UP pro práci s kalibrační pamětí viz samostatnou kapitolu o kalibrační paměti.

Soubor → Export do bin...

Pomocí této funkce lze do vybraného souboru zapsat holá binární data z paměti programu nebo datové EEPROM paměti. Pro zapisovaná data lze zvolit zarovnávání po 16 nebo 8 bitech na slovo.

Soubor → Ukončení programu

Standardní klávesová zkratka Windows: Alt+F4
Klávesová zkratka: Alt+X

Tímto příkazem se program ukončí.

Upozornění: *Pokud byl otevřený soubor změněn, program se při ukončení dotáže, zda má změny uložit. Pokud je ukončení programu vynuceno příkazem vypnout počítač a program nedostane potvrzení od uživatele, systém jej po určité době násilně ukončí bez možnosti uložit otevřený soubor nebo nastavení.*
Pokud program právě pracuje s hardware, odmítá všechny systémové požadavky na vypnutí a může být systémem označen jako program, který neodpovídá.

Menu Úpravy

Úpravy → Vyplnění hodnotou...

Vyplní oblast paměti zadanou hodnotou. Funkce se používá zejména pro vymazání (samé jedničky) či vynulování (samé nuly) dané oblasti, lze však vyplňovat libovolnou zadanou nebo náhodnou hodnotou.

Při zvolení funkce Vyplnit hodnotou, program přednastaví vybranou paměť podle aktivního okna. Pokud byla před zvolením funkce Vyplnit hodnotou označena nějaká oblast paměti, program tuto oblast přednastaví pro vyplnění. Oblast paměti může být vybrána držením klávesy **Shift** a klikáním myší nebo pohybem kurzorovými klávesami. Pro označení oblasti viz hexeditory.

Úpravy → Vložení textu...

Umožňuje vložit na zvolené místo paměti text kódovaný v kódu ASCII. Konce řádků lze kódovat jako znaky NULL, CR, LF nebo CR+LF.
Lze vyplňovat jednotlivé byte, nebo ukládat do instrukcí RETLW (týká se jen paměti programu).

Při zvolení funkce Vyplnit hodnotou, program přednastaví vybranou paměť a počáteční buňku podle aktuálního okna a vybrané buňky.

Úpravy → Vybranou oblast doplnit instrukcí RETLW

Vybranou oblast paměti doplní na instrukci RETLW.
Funkci lze použít pouze z otevřeného hexeditoru, funkce je též dostupná v místní nabídce (pravé tlačítko myši) editoru.

Oblast paměti je možné označit přidržením klávesy **Shift** spolu s kliknutím myši nebo posunem pomocí kurzorových kláves. Více viz hexeditory.

Menu Zobrazit

Zobrazit → Paměť programu

Zobrazí nebo skryje okno hexeditoru paměti programu. Více o hexeditorech viz samostatná kapitola.

Zobrazit → Paměť EEPROM

Zobrazí nebo skryje okno hexeditoru datové EEPROM paměti. Více o hexeditorech viz samostatná kapitola.

Zobrazit → Konfigurační paměť

Zobrazit nebo skryje okno konfigurační paměti. Více o hexeditorech viz samostatná kapitola.

Zobrazit → Zobrazení paměti programu

Klávesová zkratka: Alt+F10

Zobrazí hexeditor paměti programu. Pokud je hexeditor již zobrazen, je přesunut na popředí. Více o hexeditorech viz samostatná kapitola.

Zobrazit → Zobrazení paměti EEPROM

Klávesová zkratka: Alt+F11

Zobrazí hexeditor datové EEPROM paměti. Pokud je hexeditor již zobrazen, je přesunut na popředí. Více o hexeditorech viz samostatná kapitola.

Zobrazit → Zobrazení konfigurační paměti

Klávesová zkratka: Alt+F12

Zobrazí editor konfigurační paměti. Pokud je editor již zobrazen, je přesunut na popředí. Více o hexeditorech viz samostatná kapitola.

Menu Součástka

Součástka → Programovat

Klávesová zkratka: Shift+F5

- **Naprogramovat vše**
  Klávesová zkratka: F5
  Smaže, zkontroluje smazání, naprogramuje a zkontroluje celou součástku. Před operací je provedena kontrola Device ID, a Code/Data Protection
- **Naprogramovat vše kromě paměti EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Provede totéž, jako Naprogramovat vše s výjimkou mazání, programování a kontroly datové EEPROM paměti. U součástek bez datové EEPROM paměti, není tato funkce dostupná a programování se provádí pomocí funkce Programovat Vše.
  V některých případech při použití Code nebo Data Protection není možné tuto funkci použít. V takovém případě program nabízí možnost smazat celou součástku a naprogramovat i datovou EEPROM paměť (daty, která jsou aktuálně v editoru)
- **Naprogramovat paměť programu**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Smaže, zkontroluje smazání, naprogramuje a zkontroluje programovou paměť.
- **Naprogramovat paměť EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Smaže, zkontroluje smazání, naprogramuje a zkontroluje datovou EEPROM paměť.

- **Naprogramovat konfigurační paměť**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Naprogramuje a zkontroluje konfigurační paměť.
- **Naprogramovat rozdílově**
  Klávesová zkratka: Ctrl+F5
  Tato funkce naprogramuje součástku rozdílově, to znamená, že součástku vyčte a přeprogramuje pouze buňky, kde se obsah součástky a editoru neshoduje. Pokud má součástka aktivní Code/Data Protection, rozdílové programování nemá smysl, a místo něj program provede kompletní programování se smazáním součástky. Tuto funkci musí programovaná součástka podporovat, proto není dostupná pro všechny součástky.
- **Naprogramovat rozdílově paměť EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Tato funkce naprogramuje datovou paměť rozdílově, funkce této položky je stejná jako u rozdílového programování paměti programu. Tuto funkci musí programovaná součástka podporovat, proto není dostupná pro všechny součástky. Pokud má součástka aktivní Code/Data Protection, rozdílové programování nemá smysl, a místo něj program provede kompletní programování se smazáním součástky.
  Rozdílové programování paměti EEPROM je nutné použít u procesorů AVR, pokud uživatel potřebuje přeprogramovat pouze datovou paměť bez předchozího mazání součástky.
- **Sériová výroba**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Zobrazí okno pro jednoduché programování několika kusů součástek stejným nebo velmi podobným programem (až na sériové číslo atp.). Viz serializace výroby.

Některé položky mohou být pro určité typy součástek nedostupné.

Součástka → Čtení

Klávesová zkratka: Shift+F6

- **Přečíst vše**
  Klávesová zkratka: F6
  Přečte obsah celé součástky.
- **Přečíst vše kromě paměti EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Přečte obsah celé součástky kromě paměti EEPROM.
- **Přečíst paměť programu**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Přečte obsah paměti programu.
- **Přečíst paměť EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Přečte paměť EEPROM.
- **Přečíst konfigurační paměť**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Přečte konfigurační paměť.

Některé položky mohou být pro určité typy součástek nedostupné.

Součástka → Ověření

Klávesová zkratka: Shift+F7

- **Zkontrolovat vše**
  Klávesová zkratka: F7
  Porovná celou součástku s obsahem editorů.
- **Zkontroluje vše kromě paměti EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Porovná součástku kromě EEPROM paměti s obsahem editorů.
- **Zkontrolovat paměť programu**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Porovná paměť programu s obsahem editoru programové paměti.

- **Zkontrolovat paměť EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Porovná datovou EEPROM paměť s obsahem editoru datové EEPROM paměti.
- **Zkontrolovat konfigurační paměť**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Porovná konfigurační paměť s obsahem editoru konfigurační paměti.

Některé položky mohou být pro určité typy součástek nedostupné.

Součástka → Smazání

Klávesová zkratka: Shift+F8

- **Smazat vše**
  Klávesová zkratka: F8
  Smaže celou součástku.
- **Smazat paměť programu**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Smaže paměť programu. Pokud je aktivní Code/Data Protection, nelze tuto funkci použít.
- **Smazat paměť EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Smaže datovou EEPROM paměť a zkontroluje ji. Pokud je aktivní Code/Data Protection, nelze tuto funkci použít.

Po funkci mazání se automaticky provádí kontrola smazání. Protože ale mazání součástky je v drtivé většině případů bezchybné, v nastavení programu lze vynutit vynechání této kontroly (*Nastavení -> Nastavení programu -> Programování -> ...*).

Součástka → Kontrola smazání

Klávesová zkratka: Shift+F9

- **Kontrola smazání všeho**
  Klávesová zkratka: F9
  Ověří, zda je celá součástka správně smazaná.
- **Kontrola smazání všeho kromě paměti EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Ověří, zda je celá součástka mimo datové EEPROM paměti správně smazaná.
- **Kontrola smazání paměti programu**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Ověří, zda je paměť programu správně smazaná.
- **Kontrola smazání paměti EEPROM**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Ověří, zda je datová paměť EEPROM správně smazaná.
- **Kontrola smazání konfigurační paměti**
  Klávesová zkratka může být přiřazena v Nastavení → Klávesové zkratky
  Ověří, zda je konfigurační paměť správně smazaná.

Některé položky mohou být pro určité typy součástek nedostupné.

Součástka → Výběr součástky...

Klávesová zkratka: F4

Dialogové okno slouží k výběru součástky. U některých typů pamětí je po vybrání typu nutné zadat organizace dat.

V dialogovém okně pro výběr součástky jsou zobrazeny pouze ty součástky, které podporuje vybraný programátor. Pokud vybranou součástku nelze programovat ve vybraném režimu pomocí ICSP, režim ICSP se automaticky vypne. Pro výběr součástky, kterou daný programátor nepodporuje, je třeba nejprve zvolit jiný programátor.

Menu Nastavení

Nastavení → Nastavení programu...

Klávesová zkratka: Shift+F10

Nastavení → Nastavení programu...; záložka Programování

Klávesová zkratka: Shift+F10

V tomto okně lze nastavit veškerá obecná nastavení programování.

Nastavení týkající se jednotlivých programátorů (např. nastavení komunikačního portu, režim ICSP apod.) jsou v nastavení programátoru. Pro nastavení typu programované součástky, nebo u pamětí výběr organizace, je zvláštní okno pro výběr součástky.

- **`Načíst hex soubor vždy znovu před programováním`**
  Při zapnutém tomto nastavení program vždy před jakýmkoli požadavkem pro programování součástky přečte aktuální soubor z disku. Pokud je zároveň nastaveno používání sériových čísel se zápisem čísel vždy před programováním, jako první se soubor přečte a teprve pak se připíše aktuální sériové číslo.
- **Zeptat se před programováním OTP / mazatelných/ Code/Data protection/ rozdílovým programováním**
  Sada nastavení ovlivňující, které potvrzovací dialogy bude program vyžadovat a které nikoliv. Program se ptá pouze jednou, kromě případu programování Code/Data protection. Pokud se program před programováním musí uživatele zeptat na doplňující informaci (např. Vložená součástka má aktivní Code Protection, chcete ji smazat celou?), na případné další potvrzování programování již nečeká.
- **Zobrazovat varovné hlášky k pojistkám**
  Uživatel může zvolit, zda se budou zobrazovat varovné hlášky přiřazené k některým pojistkám. Doporučuje se tuto volbu ponechat zapnutou.
- **Automaticky zavřít stavové okno**
  Způsobí, že bude stavové okno zavřeno pokud nenastane chyba během mazání / programování / kontroly naprogramování.
- **Zvuková signalizace úspěšného dokončení/při neúspěšném dokončení**
  Při zapnutém nastavení program vyvolá standardní "systémový výkřik" pokud se během programování objeví chyba nebo varování nebo v případě bezproblémového programování. Záleží na nastavení.
- **Vypnout všechny zvuky programu UP**
  Pokud je zaškrtnutá tato volba, program UP nebude nikdy vydávat žádné zvuky.
- **Programování pomocí ICSP kabelu**
  Programování pomocí ICSP kabelu nelze použít na všechny typy součástek, naopak některé typy programátorů vyžadují programování některých součástek jen pomocí ICSP kabelu.
  Pro programování pomocí ICSP kabelu přímo v osazené DPS je nastavení **Delší čas pro zapojení napětí při ICSP**, které počítá s delší časovou prodlevou při připojování a odpojování napětí od součástky.
  U programátoru PICQUICK nebo PRESTO se provádí nadproudová ochrana, tzn. po určité časové prodlevě se provede test na nadměrný proud. Tato časová prodleva je při režimu ICSP řízena přímo touto nastavenou hodnotou. Delší než potřebný čas zvyšuje pravděpodobnost zničení součástky při nesprávném zapojení, při kratším čase mohou ještě obvody programátoru detekovat nadměrný proud. Detekce nadměrného proudu je cca 100mA na napájecím i programovacím napětí.
- **Delší čas pro zapojení napětí při ICSP**
  Po vybrání této volby je možné změnit nabíjecí a vybíjecí čas VDD. Pokud je na napájecích pinech připojen blokovací kondenzátor (doporučeno), napětí na pinu se mění pomaleji. To může způsobovat problémy během programování, řešením je prodloužit nabíjecí a vybíjecí čas VDD. Vzorec k přibližnému určení potřebného času je možné nalézt v kapitole **Použití ICSP**.
- **Neprovádět blank check při programování pouze konf. slova**
  Konfigurační slovo umí většina přepisovatelných součástek přepsat, aniž by musela být celá součástka smazána. Přeskočením blank check konfiguračního slova se této vlastnosti využívá, takže program bude nesmazané slovo ignorovat. Toto nastavení se netýká programování celé součástky, kde se součástka maže kompletně celá, ale pouze přepisování konfiguračního slova. Přeskakování blank check po smazání je nastavení, které lehce urychlí programování a je vhodné zejména při ladění. Špatně smazaná součástka se také

špatně naprogramuje a chyba se odhalí pouze o trochu později - na druhou stranu, součástka se špatně smaže jednou za stovky pokusů.

- **Neprovádět blank check po mazání**
  Pokud je nastavená tato položka, programátor nekontroluje zda byla součástka správně smazána. Programování bude rychlejší, ale může nastat problém se špatně smazanou součástkou.
- **Nemazat součástku před programováním**
  Součástka nebude před programováním smazána.
- **Nemazat datovou paměť před programováním**
  Tato položka je užitečná pro programování procesorů AVR, kde je HEX soubor načítán samostatně pro paměť EEPROM. Pokud nechce uživatel měnit obsah paměti EEPROM, může použít tuto volbu.
- **Neprovádět kontrolu naprogramování prázdných pozic na konci paměti**
  Pokud je na konci programované paměti oblast, která obsahuje jen defaultní hodnoty, nebude se verifikovat. Tato funkce umožňuje zrychlit verifikaci naprogramované paměti, na obsahu prázdné paměti na konci obvykle nezáleží.
- **Nekontrolovat po programování**
  Tato volba umožňuje zcela vypnout verifikaci naprogramované součástky. Vypnutím verifikace lze dosáhnout značného zrychlení programovacího procesu při vývoji. Volba nesmí být použita ve výrobě, při vypnuté verifikaci nelze zaručit správnost naprogramovaného obsahu.

Nastavení → Nastavení programu...; záložka **Panely**

Klávesová zkratka: Shift+F10

V této části menu může být nastaven vzhled aplikace. Uživatel může nastavit kde budou které ovládací komponenty zobrazeny.

- **Panel nástrojů** je lišta s tlačítky rychlé volby pod výběrem menu programu. Pokud chcete lištu odstranit, možnosti **zobrazit nápisy** i **zobrazit ikony na tlačítkách toolbaru** zrušte.
- **Stavový panel** (panel ve spodní části okna) lze odstranit. Jsou v něm zobrazeny informace o programátoru, ICSP režimu, součástce, změně souboru od posledního uložení apod. Stavový panel reaguje na dvojité kliknutí a nabídku na pravém tlačítku myši.
- **Počítadlo sériové výroby** se zobrazuje na stavovém panelu a ukazuje počet programovaných součástek a počet úspěšně programovaných součástek.

Nastavení → Nastavení programu...; záložka **Soubory**

Klávesová zkratka: Shift+F10

- **Styl ukládání souborů**
  Slouží k možnosti neukládat vždy všechny oblasti do `.HEX` souboru, ale pouze některé.
- **Kontrolovat změny v hex souboru**
  Slouží zejména při ladění programu k opětovnému přečtení souboru po detekci změny data vytvoření souboru.
- **Kontrolovat typ součástky při čtení hex souboru**
  Pokud byl do `.HEX` souboru uložen i typ součástky a typ součástky v souboru a vybrané se neshodují, program na tuto neshodu upozorní.
- **Ukládat typ součástky do hex souboru**
  Za konec Intel HEX souboru připíše ještě jeden řádek s typem vybrané součástky, pro kterou byl soubor uložen. Takovýto soubor nevyhovuje formátu Intel HEX, avšak většina programů pracujících s Intel HEX formátem tento řádek ignoruje. Více viz popis Intel HEX souborů.
- **Způsob načítání a ukládání BIN souboru**
  Zde může být nastaveno jak budou načítány a ukládány BIN soubory pokud je zvolena součástka s více byty na slovo. Jsou možnosti, že se bude program vždy před načtením nebo uložením BIN souboru ptát nebo že program vždy bez ptaní načte soubor jako Little Endian nebo jako Big Endian.
- **Do souboru hex ukládat prázdné pozice**
  Pokud se nebudou ukládat všechny pozice, výsledný soubor bude menší, ale může dojít k nepříjemnostem, protože za "prázdnou" pozici se považuje taková buňka, která má obsah samé jedničky (tedy FFFh, 3FFFh atd...) což může být i smysluplná instrukce (např. 3FFFh je `addlw -1`)

Protože ale program ukládá soubory vždy po větších blocích, osmi nebo šestnácti bytech, výpadek uložení takovéto instrukce je nižší.

- **Inicializovat paměť programu / EEPROM / ID pozice před čtením ze souboru.**
  Před čtením souboru se tato oblast vyplní jedničkami a pak se začne soubor číst. Takto se smažou všechny pozice, které v souboru nejsou uloženy.
- **Inicializovat konfigurační paměť před čtením ze souboru**
  Pokud nejsou pojistky uložené v hex souboru, je užitečné tuto volbu zrušit.
- **Přečíst paměť EEPROM/ID pozice ze součástky místo čtení ze souboru**
  Program danou oblast pro eliminování chyby přepsání dané oblasti raději vyplní obsahem součástky vložené v programátoru.
  Upozornění! Tato funkce může způsobit při práci s programátorem v nečekané chvíli, např. zapnutí programu.
- **Způsob ukládání projektů**

  Zde mohou byt nastaveny vlastnosti ukládání projektů.

Nastavení → Nastavení programu...; záložka **Barvy**

Klávesová zkratka: Shift+F10

Zde mohou být změněny barvy hex editorů, tak, aby vyhovovaly potřebám uživatele a jeho estetickému cítění.

Nastavení → Nastavení programu...; záložka **Editory**

Klávesová zkratka: Shift+F10

- **Zúžený editor paměti kódu**
  Nastavení způsobí zúžení editoru z původních šestnácti buněk pouze na osm. Nastavení je vhodné zejména pro malé monitory. Nastavení se může samo změnit při změně součástky.
- **Maskovat ID pozice**
  Podle specifikací je doporučeno do ID pozic dávat pouze maskovaná data, kde jsou využitelné většinou pouze čtyři bity. Při povolení tohoto nastavení bude program požadovanou bitovou masku zavádět.
- **V okně konfigurační paměti zobrazit místo pojistek přímo konf. slova**
  Doporučeno pouze pro pokročilé uživatele.
  Přímou editací pojistek se rozumí přímé vepsání hodnoty konfiguračního slova. Při zadání "nepřeložitelného" slova program nerozpoznané položky nechá nezměněné, pokud je uživatel sám nezmění. (Většinou se týká pojistek CP, které mají několik bitů, ale pouze dvě hodnoty.)

Nastavení → Nastavení programu...; záložka **Ostatní**

Klávesová zkratka: Shift+F10

- **Nastavení kontroly nové verze programu UP**
  Umožňuje nastavit zda se bude program při každém spuštění dotazovat na povolení připojit se k Internetu a zkontrolovat zda je na webu novější verze programu. Další možné volby jsou nikdy se neptat a vždy kontrolovat nebo nikdy se neptat a nikdy se k Internetu nepřipojovat.

Nastavení → Nastavení programu...; záložka **Sériová čísla**

Klávesová zkratka: Shift+F10

Pro informace o sériových číslech a jejich vlastnostech viz samostatnou kapitolu.

Nastavení → Výběr jazyka...

Klávesová zkratka: Ctrl+L

Pomocí standardního dialogového okna lze vybrat nový soubor s lokalizací, což umožňuje používat jednu instalaci programu v různých jazykových mutacích.

Nastavení → Klávesové zkratky

Klávesová zkratka: Ctrl+K

Pomocí tohoto dialogového okna lze měnit či definovat nové klávesové zkratky.

Menu Nápověda

Nápověda → Nápověda k programu

Klávesová zkratka: F1

Tímto příkazem se vyvolá nápověda, kterou právě čtete. Nápovědu lze vyvolávat z různých míst programu, vždy stiskem klávesy F1.

Nápověda → Seznam podporovaných součástek

Zobrazí seznam součástek podporovaných touto verzí programu UP.

Nápověda → Zkontrolovat aktualizace na Internetu

Program se připojí k Internetu a zkontroluje zda používáte aktuální verzi programu.

Nápověda → ASIX s.r.o. na Internetu

Otevře internetové stránky ASIX s.r.o..

Nápověda → Informace o programu

Zobrazí okno s informacemi o programu.

Okno nastavení programátoru PRESTO

Napájení v klidu

- **Žádné/Externí**: Programátor na VDD pinu neposkytuje žádné napětí. Spouštět / zastavit program lze pouze při externím napájení z připojené aplikace.
- **Interní 5V**: Programátor na VDD pinu poskytuje 5V. Aplikace může být napájena z tohoto pinu.

Napájení během programování

- **Externí 3 až 5V**: Programátor nebude dodávat žádné napětí do aplikace, naopak bude napájet svoje vstupně výstupní obvody z napětí aplikace.
- **Interní 5V**: Programátor na VDD pinu poskytuje 5V pro programovanou součástku.

Nastavení spojená s procesory PIC

Ovládání pinu -MCLR

Těmito tlačítky lze ovládat logickou hodnotu přítomnou na pinu -MCLR během klidu, pokud je přítomno napájení.

Způsob programování

- **HVP**: Bude použito klasické programovaní s přítomnými 13V na -MCLR/VPP
- **LVP**: Bude použito programování pomocí LVP pinu, na -MCLR/VPP jsou přítomny pouze logické hodnoty 0 a 1.

Algoritmus programování

- **Auto**: Algoritmus bude vybrán podle aktuálně přítomného napětí na VDD.
- **Ucc=5V**: Bude použit vždy algoritmus pro rychlé 5V programování.
- **Ucc=3 to 5V**: Bude použit vždy algoritmus pro pomalé programování pracující ale při všech napájecích napětích.

PE

Pro součástky PIC24 a dsPIC33 je možné zvolit metodu programování pomocí CheckBoxu PE. PE je Programming Executive, tato metoda bývá rychlejší.

Nastavení spojená s procesory AVR

Frekvence oscilátoru

Během programování procesorů AVR musí být připojený externí oscilátor nebo funkční interní oscilátor. Maximální rychlost komunikace s procesorem je pak závislá na frekvenci oscilátoru.

Zrychlené programování s pomalými hodinami

Po smazání součástky jsou naprogramovány pojistky tak, že je nastavena maximální frekvence interního oscilátoru, poté může programátor se součástkou komunikovat vyšší rychlostí. Při programování součástky s použitým pomalejším oscilátorem tato volba umožňuje dosáhnout kratších časů programování. Tato volba má vliv pouze při programování celé součástky neboť na konci programování je třeba naprogramovat požadované hodnoty konfigurační paměti.

Inverzní reset

Pokud je tato volba zaškrtnutá, programátor generuje inverzní signál reset. To je vhodné pokud je v aplikaci použitý resetovací obvod, který potřebuje na vstupu inverzní signál oproti výstupnímu signálu přivedenému k procesoru a programátor je připojený přes tento resetovací obvod.

HVP

Pokud je tato volba zaškrtnutá, programátor při komunikaci se součástkou použije "vysoké" napětím na pinu RESET. To umožňuje programovat součástku se vypnutým externím RESET signálem.

Nastavení spojená s I2C paměťmi

Rychlost I2C sběrnice

Zvolte maximální možnou rychlost I2C sběrnice. PRESTO během práce na I^2C sběrnici zapíná interní pull - up o velikosti 2.2kΩ.

Adresa I2C paměti

Zvolte adresu programované I^2C paměti na sběrnici.

Okna Hex editorů

K zobrazení obsahu pamětí, které mají být programovány jsou použity tzv. hex editory. Pro odlišení stavu jednotlivých buněk jsou v hex editorech použity různé barvy, takže lze snadno poznat, které buňky byly načteny ze souboru, které byly úspěšně naprogramovány atp. Jednotlivé barvy mohou být zvoleny podle potřeb uživatele. To je zvláště doporučeno pro pracovní stanice s displeji zobrazujícími malé množství barev.

Výběr oblasti

Oblast v hex editoru může být vybrána držením klávesy **shift** a pohybem kurzorovými klávesami. Poté co je požadovaná oblast vybrána, je ji možné vyplnit zvolenou hodnotou a hodnoty doplnit na instrukci RETLW, tyto volby jsou dostupné z kontextového menu (po kliknutí pravým tlačítkem myši).

Editor paměť programu

Menu: **Zobrazit → Zobrazení paměti programu**

Klávesová zkratka k zobrazení okna: F10

Klávesová zkratka k zavření okna: Esc

Editor programové paměti zobrazuje obsah paměti kódu nebo, v případě sériových pamětí EEPROM (24xx, 93xx,...), obsah samotné paměti.

Editor EEPROM

Menu: **Zobrazit → Zobrazení paměti EEPROM**

Klávesová zkratka k zobrazení okna: F11

Klávesová zkratka k zavření okna: Esc

EEPROM (datová paměť) editor se používá k zobrazení obsahu přídavné paměti některých součástek, typicky paměti EEPROM.. *Ne všechny součástky obsahují přídavnou paměť, pro některé součástky nemusí být tento editor dostupný.*

Editor konfigurační paměť

Menu: **Zobrazit → Zobrazení konfigurační paměti**

Klávesová zkratka k zobrazení okna: F12

Klávesová zkratka k zavření okna: Esc

Editor konfigurační paměti zobrazuje nastavení, které má být naprogramováno do součástky, ale není součástí žádné z dříve zmíněných pamětí. *Ne všechny součástky potřebují konfigurační data, pro některé součástky nemusí být tento editor dostupný.*

Tipy pro pokročilé uživatele:

Ačkoliv tato konfigurační paměť může být reprezentována jako sada nastavení, ve skutečnosti to není nic víc než paměť, ke které může být přistupováno buňku po buňce a proto je možné zobrazit paměť také tímto způsobem. Může toho být docíleno zapnutím volby **Nastavení → Nastavení programu → Editory V okně konfigurační paměti zobrazit místo pojistek přímo konf. slova,** nebo dvojklikem na okno konfigurační paměti.

V okně konfigurační paměti mohou být také nalezeny ID pozice součástky (neplést s Device ID). ID pozice mohou být naprogramovány hodnotou identifikující součástku, jako např. sériové číslo. ID pozice je **vždy možné číst**, i když je součástka zamčená proti čtení.

Podle doporučení firmy Microchip by ID pozice neměly být programovány jakoukoliv hodnotou, pouze jistý počet bitů (typicky 4) by měl nést data pro identifikaci, zatímco ostatní bity by mely být naprogramovány defaultní hodnotou. Toho může být docíleno zapnutím volby **Nastavení → Nastavení programu → Editory Maskovat ID pozice...**.

5 JTAG SVF PLAYER

Tento program se používá k programování součástek s rozhraním **JTAG** programátorem PRESTO. Je možné programovat a testovat součástky, pro které existuje software poskytující data ve formátu **SVF** nebo **XSVF**.

Na příklad:

CPLD - Xilinx XC9572XL, XCR3256XL, Altera EPM7128AELC, EPM7064SLC, EPM7128SLC a další

FPGA konfigurační Flash PROM paměti – Xilinx XC3S1000, XC2V1000 a další

paměti – XC18V02VQ44I, ...

Atmel ATmega128, ATmega64, ...

Poznámky:

- Formát XSVF je doporučený pouze pro CPLD Xilinx XC9500, zatímco formát SVF je doporučený pro všechny ostatní součástky.
- JTAG player nepodporuje soubory .JED.
- Viz příklad programování CPLD firmy Altera.

5.1 Instalace

Instalace programu JTAG PLAYER je velmi jednoduchá. Získejte instalační program (JTAG_xxx_CZ.EXE, za xxx dosaďte číslo verze) z přiloženého disku CD-ROM nebo z http://www.asix.cz/ (aktuální verze bude vždy na http://www.asix.cz/) a spusťte jej. Během instalace pouze vyberte adresář, kam bude program nainstalován a jméno adresáře ve start menu.

5.2 Jednoduché programování / Testování

Vytvořte standardní ****.svf (Serial Vector Format)*** soubor používaný pro popis vysokoúrovňových operací sběrnice IEEE 1149.1.

Serial Vector Format (*.svf) je doporučený formát souboru pro všechna testování a programování, kromě Xilinx CPLD XC9500. Pro programování Xilinx CPLD XC9500 je doporučený formát *Xilinx Serial Vector Format* (*.xsvf).
Viz stav implementace SVF a XSVF souborů.
Připojte PRESTO k JTAG portu ve vaší aplikaci.
Viz doporučené zapojení výše.
Spusťte program *jtagplay.exe*.
V menu programu zvolte ***Open & Process File***.

5.3 Příklady vytváření SVF/XSVF souboru

Programování procesoru AVR ATmega128

Vygenerujte SVF soubor použitím utility *avrsvf.exe,* která je dostupná na webových stránkách firmy ATMEL v sekci *Tools & Software* 8bitových RISC procesorů AVR.
Např. :
avrsvf -datmega128 -s -e -ifmyfile.hex -pf -vf -ovmyfile.svf -mp
vygeneruje SVF soubor, jehož vykonáním bude procesor smazán, následně naprogramován a ověřeno naprogramování souboru myfile.hex. Pro více informací spusťte avrsvf -h.

Programování Xilinx CPLD

Programem iMPACT dostupným na webu firmy Xilinx vytvořte XSVF soubor. V dialogu **Operation Mode Selection**, který se zobrazí po spuštění iMPACTu, vyberte **Prepare Configuration Files → Boundary-Scan File → XSVF File**. Spusťte všechny operace (erase, program, verify, test...) jako by byl připojený programátor a zavřete iMPACT. Zpracováním XSVF souboru budou vykonány všechny nahrané akce.
Použití SVF souboru není pro programování Xilinx CPLD XC9500 doporučeno.

Programování Lattice CPLD

Nejprve vytvořte výstupní JED soubor, např. programem ispLEVER. Poté z něj pomocí programu Universal File Writer vytvořte (UFW) SVF soubor. Tento program se nainstaluje při instalaci ispVM. Programové balíky ispLEVER a ispVM jsou dostupné na stránkách firmy Lattice.

Programování Altera CPLD

Software QUARTUS II od firmy Altera umí vygenerovat SVF soubor, je třeba to nejprve nastavit v menu programu. SVF soubor jak je vygenerovaný programem nemůže být PRESTEM naprogramován do součástky, protože obsahuje špatnou hodnotu Silicon ID. Altera se k tomu vyjádřila tak, že SVF soubor byl zamýšlen pro "ATE" programátory a není ochotna generovat soubor použitelný také pro ostatní programátory. Proto musí být SVF soubor upraven ručně. V souboru je třeba smazat nebo zakomentovat sekci "CHECKING SILICON ID".

5.4 Stav implementace SVF souboru

SVF soubor byl implementovaný podle dokumentu "Serial Vector Format Specification, Revision E", který může být nalezen na http://www.asset-intertech.com/support/svf.html s těmito omezeními:

- SVF příkazy *PIO* a *PIOMAP* nejsou (zatím) implementovány
- *HDR+SDR+TDR* / *HIR+SIR+TIR* délka je maximálně 2^{31} bitů
- Maximální podporované TCK frekvence jsou 3MHz, 1.5MHz, 750kHz a podíly 1MHz začínající na 500kHz
- *RUNTEST MAXIMUM max_time SEC* parametr je ignorován
- *RUNTEST run_count* je maximálně $2^{31}/3$ (průměrně 715milionů)
- *RUNTEST min_time SEC* je maximálně $2^{31}/3$ ms (průměrně 715 sekund)
- *TRST* a *RUNTEST SCK* příkazy sdílí jeden konfigurovatelný pin PRESTA (VPP) a nemohou být použity společně

5.5 Stav implementace XSVF souboru

XSVF soubor byl implementován podle dokumentu Xilinx "XAPP503, Appendix B: XSVF File Format" s těmito omezeními:

- XSVF příkazy *XSETSDRMASKS*, *XSDRINC* a *XSIR2* nejsou (zatím) implementovány
- Použití XSVF souboru je doporučeno pouze pro programování a testování Xilinx XC9500 CPLD.

5.6 Popis nastavení programu

Default TCK signal frequency

Toto je nastavení frekvence TCK hodin, kterou PRESTO použije po SVF příkaz *FREQUENCY* s hodnotou "default". XSVF soubor nemá žádný takový příkaz, tato frekvence TCK se používá během vykonávání celého XSVF souboru. Maximální hodinová frekvence, kterou může PRESTO generovat je 3MHz.

RUNTEST without run_count

Interpreter souboru SVF by měl zůstat nejméně po specifikovanou dobu ve specifikovaném stavu a generovat hodiny na TCK. PRESTO nemůže generovat hodiny maximální rychlostí přesně dlouhou dobu (může dodržet pouze *min_time SEC* parametr), ale s mnohem lepší přesností časování může běžet na pomalých hodinách (~100kHz). Bez hodin může PRESTO dodržet *min_time SEC* parametr velmi přesně, avšak specifikace SVF souboru toto nedovoluje, třebaže mnoho součástek nevyžaduje hodiny na TCK.

RUNTEST with run_count and no timing
Tento příkaz by měl být proveden jako minimum taktů, které PRESTO dělá na TCK. Avšak některé generátory SVF souborů používají tento příkaz jako čekací cyklus předpokládajíc frekvenci hodin 1MHz, proto doporučené nastavení je "interpret as RUNTEST min_time with scale 1MHz".
VPP PRESTO pin while running test and after test
Výběr mezi funkcemi pinu VPP: TRST nebo SCK popsaný v SVF souboru nebo uživatelský výstup (vhodný pro reset signál, který drží součástku v resetu během vykonávání souboru).
RUNTEST timing multiply (in JTAG Player version 1.3 and later)
Pro přesné časování specifikované v SVF a XSVF souboru, vyplňte tuto hodnotu 0% (žádný přídavný čas). Pro programování XC9500(XL) je doporučená hodnota 100%, pro programování procesoru ATmega128 (a ostatních AVR) je doporučeno 25%.

5.7 Spouštění JTAG PLAYERu z příkazové řádky

SVF & XSVF player může být spuštěn z příkazové řádky pro vyšší komfort zvláště při debugování.
jtagplay.exe [-p] [-f filename] [-i inifile] [-c] [-cc] [-s]

-p	automaticky provede soubor specifikovaný za parametrem -f filename
-f filename	filename je jméno souboru, který se má vykonat
-i inifile	inifile kde jsou nastavení programu
-c	ukončí program, poté co byl soubor vykonán bez chyb
-cc	ukončí program, poté co byl soubor vykonán s chybami
-s	parametr "-s serial_number" vybere PRESTO podle sériového čísla. Serial_number může být na příklad "1234" nebo "011234". "01" je pevný prefix.

Program vrací tyto kódy:

0	poslední soubor byl vykonán bez chyb
1	vykonání posledního souboru bylo neúspěšné
2	vykonávání posledního souboru nemohlo začít

6 PRECOG

Tento program se používá pro programování mikrokontroléru Cyan Technology eCOG1. Obsahuje základní podporu pro ladění aplikací přes rozhraní eICE (Run, Stop, Reset).

6.1 Instalace

Instalace programu PRECOG je jednoduchá. Získejte instalační program (PRECOG_xxx_CZ.EXE, za xxx dosaďte číslo verze) z přiloženého disku CD-ROM nebo z http://www.asix.cz/ (aktuální verze bude vždy na www.asix.cz) a spusťte ho. V průběhu instalace vyberte pouze adresář kam bude PRECOG nainstalován a jméno adresáře v menu start.

6.2 Programování

Připojte PRESTO k procesoru eCOG1.
Otevřete soubor, který chcete naprogramovat, kliknutím na tlačítko **Open** nebo na položku v menu **File/Open**. Jsou podporovány soubory s příponou .rom.

Stiskněte tlačítko **Program** nebo položku menu **Device/Program** pro spuštění programování.

6.3 Ladění

Připojte PRESTO k procesoru eCOG.
Stiskněte tlačítko **Attach** nebo položku menu **Device/Attach**.
Nyní mohou být použita tlačítka pro ladění (**Run**, **Stop**, **Reset**) nebo stejně pojmenované položky menu v sekci **Debug**.

7 Knihovna presto.dll

Funkce implementované v presto.dll umožňují na jednotlivých pinech programátoru nastavovat logické úrovně dle potřeby nebo číst jejich stav, takto lze vytvářet různé komunikační protokoly. Pro ovládání všech pinů, které umožňují výstup, je tu funkce QSetPins(), pro čtení pinů s možností vstupu funkce QGetPins(). Funkcí QSendByte() je možné rychle poslat SPI Byte na pinech data a clock, pokud je současně potřeba i číst, použije se funkce QSendByte_OutIn(). Dále jsou tu funkce pro nastavení vlastností programátoru, ovládání napětí a funkce pro čtení návratových hodnot. Tuto knihovnu je možné použít se všemi programátory PRESTO bez ohledu na verzi hardware.

Funkce implementované v knihovně presto.dll jsou podrobně popsány v samostatném dokumentu věnovaném této knihovně.

Příloha A: Adresy konfiguračního slova procesorů PIC

PIC10xxx Adresy konfiguračního slova

Všechny PIC10xxx mají konfigurační slovo na adrese FFFh.

PIC12xxx Adresy konfiguračního slova

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

PIC16xxx Adresy konfiguračního slova

PIC16xxx s kofig. slovem na adrese 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: Všechny ostatní podporované PIC16xxx mají konfigurační slovo na adrese 2007h.

PIC18xxx Adresy konfiguračního slova

Mikrokontrolér	Cfg. Mem. Adr.	Mikrokontrolér	Cfg. Mem. Adr.	Mikrokontrolér	Cfg. Mem. Adr.
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: Všechny ostatní podporované PIC18xxx mají konfigurační slovo na adrese 300000h.

dsPIC30xxx Adresy konfiguračního slova

Všechny dsPIC30 mají konfigurační slovo na adrese 7C0000h.

dsPIC33xxx Adresy konfiguračního slova

Všechny dsPIC33 mají konfigurační slovo na adrese 7C0000h.

PIC24xxx Adresy konfiguračního slova

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

Všechny PIC24H mají konfigurační slovo na adrese 7C0000h.

Příloha B: UP_DLL.DLL jména nastavení a hodnoty

Podívejte se na ukázkové dávkové soubory, pro příklad, jak pracovat s těmito nastaveními.

Tyto informace jsou poskytovány pouze pro zkušené uživatele a bez jakékoliv záruky.

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 languange 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

Příloha C: Použití ICSP

ICSP (In-Circuit Serial Programming) je způsob programování mikrokontrolérů PIC, který umožňuje programovat součástky, které jsou už osazeny na desce plošných spojů. Pro programování procesorů PIC lze použít dva různé algoritmy: HVP (s +13V na Vpp) nebo LVP (s použitím pinu LVP). Programovací algoritmus LVP lze zakázat v konfiguračním slově součástky. Procesory mají z výroby povolen algoritmus LVP, proto při prvním programování je nutno ošetřovat i vstup LVP (po dobu programování pomocí algoritmu HVP musí být vstup LVP v log.0).

Piny použité během programování

HVP algoritmus (+13V applikováno na Vpp)

- -MCLR/VPP musí být oddělen od resetovacích obvodů (např. rezistorem 10kΩ). Během programování je na tento pin přivedeno programovací napětí (VPP) +13V, náběžná hrana a napěťová úroveň Vpp nesmí být aplikací ovlivněna.
- **Pin LVP** (pokud ho součástka má) **musí být držený v log.0!!**
- **Piny RB6 a RB7** nesmí aplikace během programování ovlivňovat.

LVP algoritmus (bez +13V)

- **Piny RB6, RB7, LVP a -MCLR/VPP** nesmí být během programování aplikací ovlivněny. Všechny piny jsou během programování v různých logických úrovních.

Algoritmus LVP podporuje zatím pouze programátor PRESTO.

Maximální zatěžování jednotlivých pinů programátoru aplikací (proud odebíraný z programátoru)

	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) Platí pouze pro součástky s flash pamětí. U součástek OTP může aplikace zatěžovat max. 1mA. Programátor na pinu VPP poskytuje proud 50mA, ale v případě OTP součástky je téměř všechen tento proud potřeba pro programování součástky.

Na datových pinech se průběhy mění rychlostí i několika MHz a aplikace nesmí žádným výrazným způsobem ovlivňovat rychlosti průběhů.

Možnosti napájení

Ve všech případech je samozřejmě nutné zapojit společnou datovou a napájecí zem (GND).
Napájení programovaného procesoru může být

- **externí** z aplikace
- **interní** z programátoru 5V

Externí napájení z aplikace **nelze** použít u některých typů procesorů, které mají pin -MCLR/VPP konfigurovatelný i jako I/O.
Interní napájení lze použít pouze v případě, pokud aplikace nebude z napájecího pinu (Vcc) programátoru odebírat příliš velký proud.

Programátor	Max. odběr aplikace z programátoru
PRESTO	90mA[1)]
PICQUICK	10mA[1)]
PICCOLO	50mA

1) Programátor obsahuje softwarovou nadproudovou ochranu. Při znatelném překročení maximálního zatížení po dobu delší než velkou (nastavitelná) programátor napájení vypíná. Programátor PICQUICK kontroluje přetížení pouze při

zapnutí napětí Vdd a Vpp, programátor PRESTO kontroluje stav přetížení po celou dobu zapojeného napájení.

- U programátoru **PRESTO** je podpora pro externí napájení zabudována přímo v hardware. Programátor napájí vstupní a výstupní obvody napětím, které je připojeno na pinu Vdd. Napětí může být i nižší než 5V. Prosíme, věnujte však při návrhu zvýšenou pozornost tomu, které typy lze při nižším než 5V napájení, nejen provozovat, ale i programovat.
- U programátorů **PICQUICK a PICCOLO (GRANDE)** lze externí napájení použít trikem, kdy pin Vdd na programátoru zůstane nezapojen. Je však nutné dodržet, aby napájecí napětí v aplikaci a v programátoru bylo shodné (=5V).
- Programátor **CAPR-PI** je stavěn pouze pro podporu externího napájení a jiné zapojení není možné. Pracuje pouze při napájecím napětí 5V.

Pokud jsou v aplikaci na napájecím pinu přítomny kapacity, které brzdí rychlost zapnutí a vypnutí napájení, je nutno, aby v ovládacím programu UP byly nastaveny delší předpokládané časy nabití a vybití.

Programátor	Nabíjecí proud	Vybíjecí proud
PRESTO	odpovídá 50Ω	odpovídá 1kΩ
PICQUICK	odpovídá 50Ω	odpovídá 10kΩ
PICCOLO	odpovídá 50Ω	žádný

Orientační doba, která by měla být nastavená v programu je přibližně **t[µs] = 2.5 × C[µF] × R[Ω]** .
Např. pro aplikaci s kondenzátorem 33µF programovanou programátorem PRESTO je potřebná doba pro nabíjení 2.5×33×50=4125µs a pro vybíjení 2.5×33×1000=82.5ms.

Poznámky:

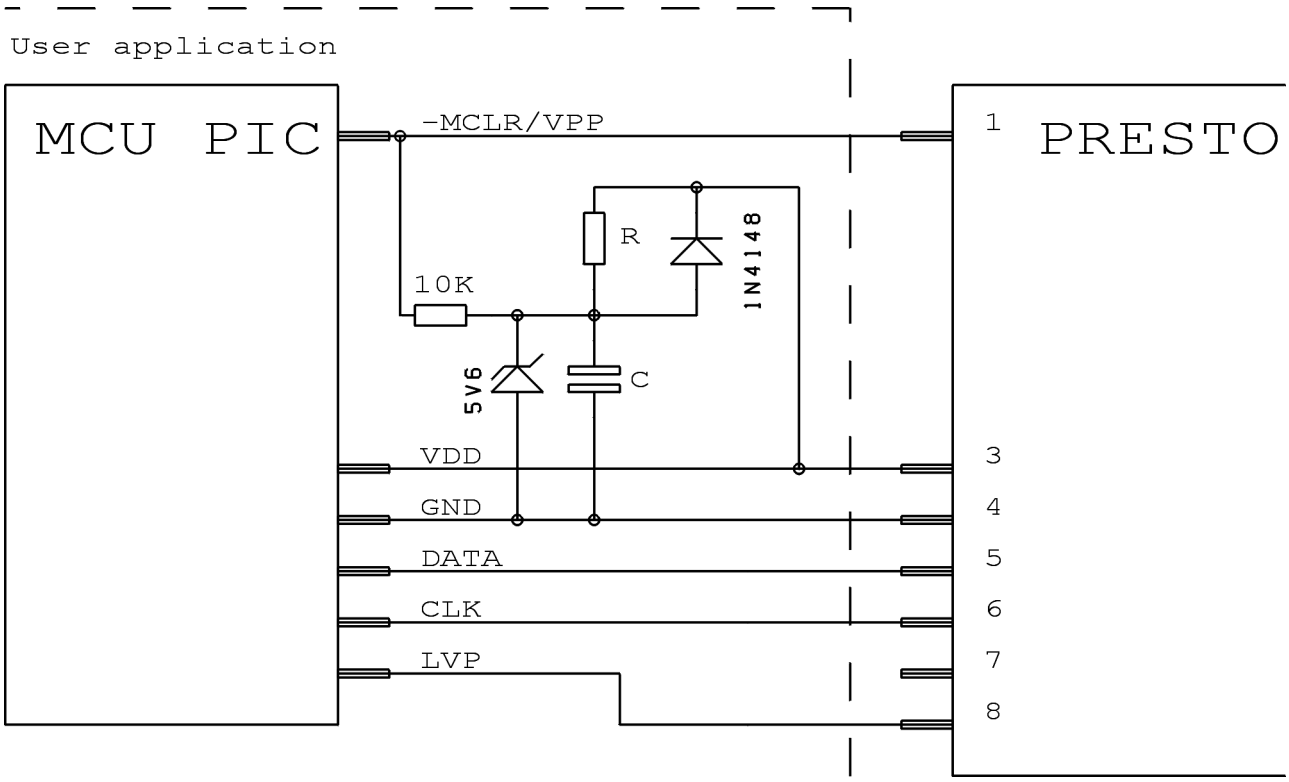
- Někdy může nastat chyba, že UP nemůže programovat kalibrační slovo nebo se vyskytují chyby během čtení device ID nebo UP upozorňuje na nadproud na VDD atp. V tomto případě může pomoci prodloužení nabíjecích a vybíjecích časů až na několik sekund v menu Nastavení programování... .
- Pokud UP upozorňuje na chybu nadproudu na VPP, použijte kratší ICSP kabel (maximálně 20cm).

ICSP konektor

Všechny programátory ASIX používají pro programování algoritmem ICSP jednotný konektor s piny v rastru 2.54mm. Tento konektor má 6 nebo 8 pinů (podle typu programátoru) s 5 resp. 7 signály. Rozšířená verze konektoru (8 pinů) obsahuje navíc signál LVP, který je použit při programování algoritmem LVP.

Číslo pinu	Signál	Programovací konektor
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

Doporučené zapojení -MCLR/VPP pinu



Doporučené zapojení bere v úvahu doporučení firmy Microchip.
Velikostí R a C lze nastavit dobu držení procesoru v resetu. Diodou se dosáhne rychlého vybití kondenzátoru C při vypojení napětí Vdd. Zenerovou diodou se omezuje působnost programovacího napětí +13V z programátoru. Odstraněním součástek R, C a diody lze dosáhnout jednoduššího zapojení, které nemá žádné zpoždění.

Historie dokumentu

Datum	Verze	Hlavní změny
1-února-2008	1.0	Počáteční verze – vznikla překladem anglického dokumentu verze 2.1
21-dubna-2008	1.1	Přidáno doporučené zapojení procesorů PSoC. Oprava adres CFG slova PIC24F v příloze A. Přidány kontaktní informace.
7-července-2008	1.2	Přidány poznámky k I^2C pamětem 34xx02. Přidán nový parametr pro příkazovou řádku /devid a nový error kód 7. Přidána nová zpráva Windows, která smaže součástku. Přidán popis nových nastavení programu UP. Přidána informace jak vytvořit SVF soubor pro CPLD od Lattice. Přidána poznámka k použití parametru /noe s MSP430.
17-října-2008	1.3	Přidána poznámka k programování PIC32MX.
9-prosince-2008	1.4	Přidána informace o funkci “Import dalšího souboru“. Přidána informace o “updateru“ v menu Help. Přidána informace jak jsou soubory načítány podle jejich přípony v dialogu Soubor/Otevřít.

Datum	Verze	Hlavní změny
		Přidána poznámka pro AVR procesory. V Technických specifikacích doplněna maximální délka ICSP kabelu. Přidána informace jak lze vytvořit SVF soubor pro CPLD firmy Altera. Drobné opravy textu.
9-ledna-2009	1.5	Pod obrázek MSP430 SBW doplněna informace o MSP430F5xxx. Pod obrázek MCU 8051 doplněna informace o podporovaných typech.
3-března-2009	1.6	Přidána poznámka o programování součástek PIC s pojistkou ICPORT. Přidána informace o logování sériových čísel do souboru.
27-března-2009	1.7	Pod obrázek zapojení JTAG součástek přidána poznámka k AVR32.
31-července-2009	1.8	Přidána informace o zapojení ATxmega. Přidána informace o novém parametru /blank pro příkazovou řádku. Přidána informace o možnosti nastavit rychlost pro MSP430 SBW. Přidáno vysvětlení co je Programming Executive. Drobná doplnění a aktualizace textu.
23-října-2009	1.9	U MicroWire pamětí doplněny informace o programování ST M93Sx6. Doplněn obrázek připojení CCxxxx součástek od Texas Instruments. U obrázku zapojení 8051 součástek doplněn AT89LP6440.
15-ledna-2010	1.10	Doplněny parametry programu UP pro příkazovou řádku. U AVR změna názvu Frekvence krystalu na Frekvence oscilátoru. V sekci Okno programátoru PRESTO doplněn popis dalších voleb. V sekci nastavení programu UP doplněna nová nastavení. V poznámkách pro PIC změněna hodnota pomocné kapacity na 1nF. Doplněny parametry v technických specifikacích programátoru PRESTO.
22-ledna-2010	1.11	Doplněna poznámka o nových součástkách PIC a řešení problému nadproudu.
14-dubna-2010	1.12	Přidán obrázek a poznámky k propojení programátoru a AVR s rozhraním TPI. Doplněny poznámky k součástkám PSoC a MSP430.
20-dubna-2010	1.13	Upravena a doplněna tabulka v kapitole “Popis programovacího konektoru“. V kapitole o instalaci ovladače doplněna sekce o instalaci pod Windows 7.
27-května-2010	1.14	Doplněny informace o parametrech pro příkazovou řádku /pdiff a /eeonly.
14-října-2010	1.15	Pod obrázkem zapojení 8051 přidána informace k programování AT89LP52. Přidána kapitola o knihovně presto.dll. Drobné jazykové úpravy.
2.března-2011	1.16	Doplněny informace o CC430. Do obsahu přidány odkazy na zapojení jednotlivých součástek.