FORTE

High-Speed USB programátor

Referenční příručka

ASIX s.r.o.
Na Popelce 38/17
150 00 Praha 5 - Košíře

www.asix.cz

podpora@asix.cz

obchod@asix.cz

ASIX s.r.o. si vyhrazuje právo změny tohoto dokumentu, jehož aktuální podobu naleznete na Internetu.

ASIX s.r.o. nenese žádnou zodpovědnost za škody způsobené použitím produktu firmy ASIX s.r.o.

13.5.2025

Obsah

1 Úvod

Tento manuál popisuje High-Speed USB programátor FORTE vyrobený firmou ASIX a jeho ovládací software. V první kapitole najdete návod jak rychle začít s programátorem pracovat, příklady připojení programátoru k aplikaci a technické specifikace programátoru.

Druhá kapitola pojednává o instalaci ovladačů a aktualizaci software.

Třetí kapitola je o programu UP. Tento software se používá pro ovládání všech programátorů firmy ASIX. Naleznete zde postup, jak nastavit programátor před programováním, jak součástku naprogramovat či verifikovat. Je zde také popis použití programu z příkazového řádku a nebo DLL knihovny.

Čtvrtá kapitola popisuje JTAG SVF Player, program který slouží k programování součástek s rozhraním JTAG a používajících soubory typu SVF (.svf/.xsvf).

V páté kapitole jsou rady a tipy, jak postupovat v případě potíží s programováním součástky.

Jazykový koutek:

Jsme si vědomi toho, že manuál obsahuje po stránce českého jazyka některé nedostatky. Jedná se především o slovosled, kde jsme občas v rámci zažitých termínů byli nuceni použít slovosled z jazyka anglického. Také vázání předložek s podstatnými jmény není ideální, v tomto ohledu s námi odmítl spolupracovat program, ve kterém je tento manuál napsán. Všem s jazykovým citem se proto tímto omlouváme.

1.1 Použité zkratky a termíny

HVP	(High Voltage Programming) je režim programování, kdy na pin „P“ je na začátku programování přivedeno napětí vyšší, než je napětí napájecí
ICSP	(In-Circuit Serial Programming). V tomto manuálu je význam ICSP totožný s významem ISP (In System Programming), tedy programování součástky přímo v systému
LVP	(Low Voltage Programming) je režim programování, kdy na žádném z pinů není napětí vyšší, než je napětí napájecí
PDI	„Program and Debug Interface“
SBW	„SPY-BI-WIRE“ rozhraní mikrokontroléru MSP430
SWD	„Serial Wire Debug“ rozhraní mikrokontroléru ARM
TPI	„Atmel Tiny Programming Interface“
VCC	pokud je v textu uvedeno VCC nebo VDD, je tím myšleno napájecí napětí na pinu „VDD“, které může být vzhledem k aplikaci vstupní i výstupní
VPP	pokud je v textu uvedeno VPP, je tím myšleno programovací napětí na pinu „P“ u součástek s HVP

Soubor – v textu je souborem chápán obecně soubor s daty. Pokud se jedná o konkrétní typ souboru, je toto dále specifikováno.

Označením .hex je myšlen soubor typu Intel-HEX, označením .bin pak binární soubor.

2

FORTE

Děkujeme Vám za zakoupení programátoru FORTE firmy ASIX s.r.o., učinili jste správné rozhodnutí. V případě jakýchkoli dotazů nebo nejasností nás neváhejte kontaktovat na naší technické podpoře.

2.1 Obsah balení

Součástí balení je:

- programátor FORTE
- kabel ICSPCAB16
- kabel ICSPCAB8
- kabel USB (typ A - B)
- příbalový leták

2.2 Přehled vlastností

FORTE je rychlý a flexibilní High-Speed USB In-System programátor využitelný k programování různých součástek, jako jsou např. mikrokontroléry, sériové paměti EEPROM a Flash, CPLD, FPGA a další.

- velmi rychlý programátor (30 MHz out, 15 MHz in/out)
- rozhraní USB 2.0 High-Speed (480 Mbps), napájení programátoru z USB
- embedded procesor pro komplexní funkce
- synchronní i asynchronní programování, podpora rozhraní JTAG
- programuje při napětí od 1,8 V[1] do 5,5 V
- napájení aplikace napětím od 1,8 V do 5,5 V
- programovací rozhraní 8x I/O + individuálně konfigurovatelné pull-up/down rezistory
- zabudovaná rychlá HW ochrana vstupů/výstupů nezávislá na stavu PC
- nadproudová ochrana na zdrojích VDD a VPP
- přepěťová ochrana na pinu VDD
- tlačítko GO pro rychlou volbu funkce
- více programátorů na 1 PC najednou, podpora příkazového řádku, zpráv Windows a DLL
- Windows XP a novější
- kompaktní provedení

[1] typicky je možné programovat sníženou rychlostí i při napětí 1,2 V

2.3 Rychlý start

Před prvním použitím programátoru FORTE je nutné nainstalovat ovladače a program UP.

2.3.1 Windows

Pro instalaci a pro první spuštění programu UP musí mít uživatel administrátorská práva. Při dalším použití budou dostačující práva běžného uživatele.

Nejprve nainstalujte program UP. Instalátor tohoto obslužného software nainstaluje také USB ovladač programátoru FORTE. Instalátor programu UP najdete na www.asix.tech/prg_up_cz.html.

Po dokončení instalace připojte programátor FORTE k počítači.

Ovladač obsažený v instalátoru programu UP je určen pro Windows 7 a novější. Pro starší verze operačního systému Windows je třeba stáhnout ovladač z webu www.asix.cz, z download sekce programátoru a někam ho rozbalit. Po připojení programátoru se operační systém zeptá na ovladač, v dialogu Nalezen nový hardware je třeba nastavit cestu k rozbalenému ovladači.

Po úspěšné instalaci se rozsvítí zelená ON-LINE LED na programátoru a ve Správci zařízení bude možné najít programátor jako správně nainstalovaný.

2.4 Použití

FORTE je rychlý a flexibilní High-Speed USB programátor využitelný k programování různých součástek jako jsou např. mikrokontroléry, sériové paměti EEPROM a Flash, CPLD, FPGA a další. Programátor má nadproudovou ochranu na zdrojích VDD a VPP a přepěťovou ochranu na pinu VDD.

Programátor je napájený z USB a může buď napájet programovanou aplikaci napětím od 1,8 V do 5,5 V nebo může během programování používat externí napájecí napětí z aplikace.

Programátor může být použit pod Windows XP a novějšími.

2.4.1 Mnoho podporovaných součástek

Seznam podporovaných součástek obsahuje např.:

- Mikrokontroléry **Microchip PIC** – součástky se sériovým programováním, což jsou všechny součástky PIC a dsPIC kromě několika zastaralých součástek.
- Mikrokontroléry s jádrem **ARM** – např. ATSAM3N2A nebo LPC2148.
- Mikrokontroléry **Atmel AVR** – všechny součástky podporující "SPI Low Voltage Serial Downloading", jako např. ATtiny12, AT90S8535 nebo ATmega128.
- Mikrokontroléry **Atmel ATxmega** mikrokontroléry – součástky programované prostřednictvím rozhraní JTAG nebo PDI, např. ATxmega32D4 nebo ATxmega128B1.
- Mikrokontroléry **Atmel AVR32** např. AT32UC3A1256.
- Mikrokontroléry **Atmel 8051** – součástky podporující ISP programování, např. AT89S8253, AT89LP4052, AT89LP216 nebo AT89S2051.
- Mikrokontroléry **Texas Instruments** – 16-bitové MSP430, CC430 a CCxxxx s flash pamětí včetně programování ochranné pojistky.
- Mikrokontroléry **Cypress** PSoC.
- Paměti **Serial EEPROM** a **Flash** - I2C (24LCxx), Microwire (93LCxx) a SPI (25Cxx).

- Součástky s rozhraním **JTAG**, pro které je možné vytvořit soubor typu SVF nebo XSVF. Mezi takové patří CPLD (např. Xilinx XC95xx a CoolRunner), konfigurační paměti pro FPGA (např. Xilinx XC18Vxx a XCFxxS), mikrokontroléry (např. ATmega128) a další.

Tím však nejsou jeho možnosti vyčerpány - podle zájmu zákazníků jsou průběžně doplňovány další typy. Na webu jsou zdarma k dispozici nové verze programu.

2.4.2 Vysoká rychlost

Oproti programátoru PRESTO disponuje FORTE embedded procesorem, umožňujícím provádět komplexní operace s programovanou součástkou, a značně rychlejším výstupním rozhraním. To umožňuje programovat součástky na hranici jejich teoretických možností.

2.4.3 Připojení k USB

Programátor FORTE je řízen a napájen prostřednictvím sběrnice USB. Komunikuje v režimu High-Speed (480 Mbps) při připojení k portu USB 2.0 nebo Full-Speed při připojení k portu USB 1.1 . Připojení programátoru je tedy rychlé a snadné, stačí jediný kabel.

2.4.4 Programování osazených součástek

ISP (In-System Programming) nebo pro mikrokontroléry PIC speciálně ICSP (In-Circuit Serial Programming) v současné době vytlačuje klasické programování, kdy se součástka nejprve naprogramuje, a teprve poté se osadí na desku. Pomocí ISP se snadno programují i SMD součástky s velmi malou roztečí pinů a je umožněn upgrade firmware v již hotovém zařízení.

2.4.5 Programování samostatných součástek

Ti, kdo se neobejdou bez nutnosti programovat i samostatné tj. na desce neosazené součástky, mohou použít doplněk ISP2ZIF, který je vybaven paticí s nulovou zasouvací silou.

2.4.6 Programovací rozhraní

Programátor FORTE disponuje 8 nezávislými I/O piny s individuálně konfigurovatelnými pull-up/down rezistory.

K připojení programované součástky slouží 16-pinový konektor ISP, který je shora kompatibilní s 8-pinovým konektorem ICSP pro mikrokontroléry PIC. VPP je nastavitelné v rozsahu 6,5 V až 17 V.

Připojení vzdálenější aplikace je při použití plochého kabelu s 16 žilami možné díky prokladu jednotlivých signálů zemí.

2.4.7 Možnosti napájení aplikace

Vývod VDD může být vstupem, kdy se pro napájení výstupních bufferů programátoru použije napětí z aplikace, ale také může být výstupem a naopak napětí aplikaci poskytovat.

Programátor FORTE umí napájet aplikaci napětím od 1,8 V do 5,5 V. Pro rychlejší přechody mezi jednotlivými fázemi programování, kdy je nutné vybít kondenzátory v aplikaci až na 0 V, je zabudován řízený vybíjecí rezistor.

Programátor FORTE obsahuje zabudovaný „voltmetr“ na pinech P i VDD.

2.4.8 Nadproudová ochrana

Programátor FORTE má zabudovanou rychlou hardwarovou ochranu proti nadproudu na pinech P a VDD, která je nezávislá na stavu ovládacího PC.

2.4.9 Uživatelské rozhraní

Stav programátoru je přehledně indikován pomocí dvou LED. ON-LINE (zelená LED) informuje o připojení k USB, ACTIVE (dvoubarevná žlutá/červená LED) signalizuje aktivitu nebo chybový stav na programovacím rozhraní.

Pohodlí obsluhy při opakovaném programování značně zvyšuje tlačítko GO, které spouští programování nebo jiné uživatelem definované příkazy.

2.4.10 Software

Základním softwarovým prostředkem pro práci s programátorem FORTE je program UP, který je zároveň určen také pro programátor PRESTO.

Kromě běžných příkazů poskytuje program UP řadu nadstandardních funkcí, které rozšiřují možnosti použití programátoru a ulehčují jeho obsluhu. Jde např. o možnost definování projektů, parametry při spouštění z příkazového řádku umožňující bezobslužné použití programátoru při rutinním programování, nastavení prostředí včetně klávesových zkratek, automatické generování sériových čísel, apod.

Program UP je určen pro Windows XP a novější.

Pro komunikaci po USB se používají modifikované ovladače firmy FTDI.

Součástky s rozhraním JTAG, pro které je možné vytvořit soubor typu SVF nebo XSVF, je možné programovat pomocí programu JTAG SVF Player.

Pro větší pohodlí a kontrolu funkce je v programu UP neustále zobrazováno napětí na pinu VDD. Stav resetovacího signálu lze snadno ovládat pomocí tlačítka v programu.

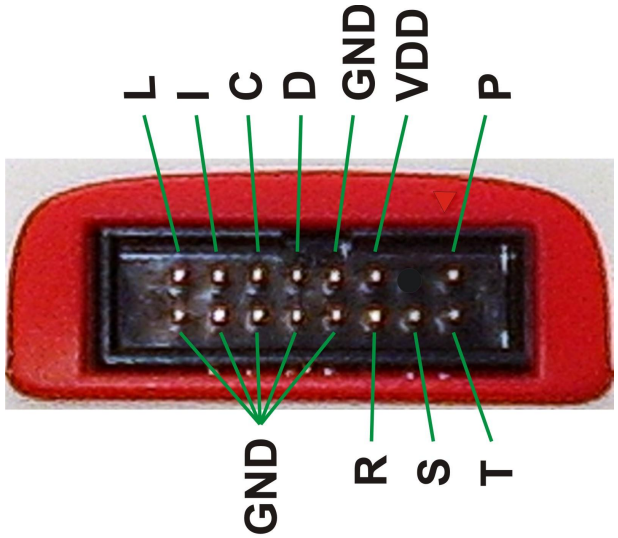
2.5 Ovládací prvky a konektory

Obr. 2: Programátor FORTE

Programátor FORTE obsahuje dvě diody LED, tlačítko, konektor pro připojení k USB a programovací konektor.

2.5.1 Programovací konektor

Programovací konektor je tvořen dvouřadým 16 pinovým konektorem s roztečí 2,54 mm, kde je pin č. 3 (při standardním číslování) vynechán.



Obr. 3: Programovací konektor

Pin	Typ	Popis
P	I/O, VPP	logický vstup/výstup nebo výstup VPP
VDD	PWR	napájení vstup/výstup
GND	PWR	signálová zem
D, C, I, L, T, S, R	I/O	log. vstup/výstup

Tab. 1: Programovací konektor

2.5.2 Tlačítko GO

Tlačítko usnadňuje práci s programovanou aplikací. Umožňuje spustit programování součástky nebo jinou přednastavenou funkci. Více viz kapitola Nastavení funkce tlačítka GO.

2.5.3 Indikátory LED

ON-LINE

Zelená dioda signalizuje, že programátor FORTE je připojen k počítači a zároveň si počítač s programátorem rozumí, tedy ovladače jsou správně nainstalovány.

ACTIVE

Dvoubarevná dioda signalizuje stav zařízení.

Žlutá barva znamená, že právě probíhá komunikace programátoru se součástkou.

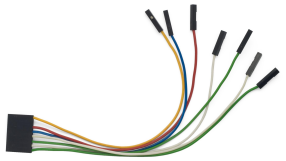
Červená barva signalizuje chybový stav.

2.5.4 USB konektor

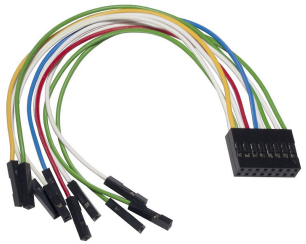
Pro připojení k počítači je určen standardní konektor USB, typ B. Programátor využívá pro komunikaci USB High-Speed (480 Mbps) rozhraní.

2.6 Popis propojení s aplikací

Programátor se k programované aplikaci připojuje pomocí kabelu ICSPCAB8 nebo ICSBCAB16. Kabely jsou vhodné pro konektory s roztečí 2,54 mm.



Obr. 4: ICSPCAB8



Obr. 5: ICSPCAB16

Specifikace použitých konektorů je v následující kapitole.

2.6.1 Zákaznický propojovací kabel

Pokud by programovaná aplikace měla nekompatibilní typ konektoru pro připojení programátoru, může zákazník zhotovit vlastní programovací kabel. Jeho délka by neměla překročit 15 cm.

Zde se s výhodou může uplatnit proklad jednotlivých signálů pomocí GND na 16 žilovém plochém kabelu. Aby se proklad zemí uplatnil, musejí být na straně aplikace aktivně zapojeny všechny signály GND.

V následující tabulce jsou uvedeny příklady vhodných konektorů od firmy FCI pro výrobu zákaznického kabelu

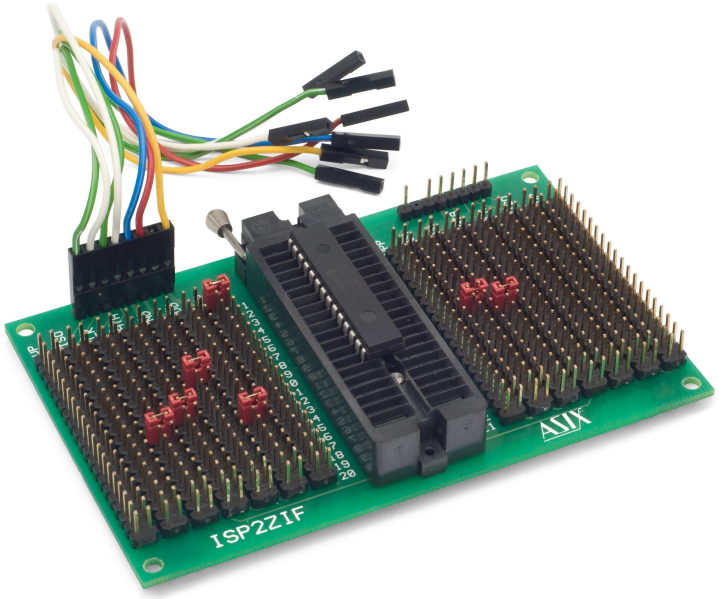
značení FCI	Popis
65039-036LF	housing, 1 pin
65039-029LF	housing, 1 x 8 pinů
65043-029LF	housing, 2 x 8 pinů
47217-000LF	pin

Tab. 2: ICSP kabel - materiálový list

Pro výrobu propojovacího kabelu je vhodné použít kabel s průřezem cca 0,1 až 0,3 mm^2.

2.6.2 Programování v patici ZIF

Pokud je vyžadováno programování samotných obvodů, které se teprve poté použijí v aplikaci, je to možné pomocí volitelného příslušenství ISP2ZIF.



Obr. 6: ISP2ZIF

ISP2ZIF obsahuje patici s nulovou zasouvací silou a konektor ICSP pro připojení programátoru, který zároveň může zajistit napájení programovaného obvodu.

2.6.3 Postup připojení

Správný způsob, jak připojit programátor, je nejprve spojit FORTE a cílovou aplikaci, potom připojit FORTE do USB a nakonec zapnout napájení aplikace.

Je nutné dbát na to, aby GND aplikace, programátoru a USB byly propojené jako první a teprve poté signály a napájení.

Důležité upozornění

Pokud je programovaná aplikace napájená ze spínaného zdroje nebo není uzemněná, může být mezi zemí programátoru a aplikace velký napěťový rozdíl, který může způsobit zničení programátoru.

Nejjednodušší způsob, jak propojit GND dříve než ostatní signály, je uzemnit aplikaci před připojením programátoru a to např. tak, že bude pin GND konektoru ICSP v aplikaci delší než ostatní piny. Tak bude zajištěno, že se nejprve propojí země.

Tabulka propojení

Pin	AVR	AVR TPI	ATxmega PDI	8051
P	RESET	RESET		RESET
-				
VDD	VCC	VCC	VCC	VCC
GND	GND	GND	GND	GND
D	MOSI	TPIDATA	PDI_DATA	MOSI
C	SCK	TPICLK	PDI_CLK	SCK
I	MISO			MISO
L				SS
T				
S				
R				

Tab. 3: Popis propojení č.1

Pin	PIC	MSP430	MSP430 SBW	TI CCxxxx
P	MCLR	TEST / VPP	VPP	RESET
-				
VDD	VDD	VCC	VCC	VDD
GND	VSS	VSS	VSS	GND
D	PGD	TDI	SBWTDIO	Debug_data
C	PGC	TCK	SBWTCK	Debug_clock
I		TDO		
L	LVP	TMS		
T				
S				
R		RESET		

Tab. 4: Popis propojení č.2

Pin	I²C	SPI	QUAD SPI[1]	Microwire	UNI/O
P		- CS	- CS	CS	
-					
VDD	VDD	VDD	VDD	VDD	VCC
GND	GND	GND	GND	GND	VSS
D	SDA	SI	SIO0	DI	SCIO
C	SCL	SCK	SCK	CLK	
I		SO	SIO1	DO	
L			SIO2	ORG/(PRE)	
T					
S			SIO3		
R					

Tab. 5: Popis propojení č.3

Pin	PSoC	1-Wire	ARM SWD	LPC UART
P	XRST	IO[2]	NRST	RESET
-				
VDD	VDD	VDD	VDD	VDD
GND	VSS	GND	GND	VSS
D	ISSP-DATA		SWDIO	RXD
C	ISSP-SCLK		SWCLK	TXD
I				ISP
L				
T				
S				
R				

Tab. 6: Popis propojení č.4

Pin	JTAG	HCS08	HCSxxx	C2
P	USR	RESET		
-				
VDD	VDD	VDD	VDD	VDD
GND	GND	VSS	VSS	GND
D	TDI	BKGD	Data	C2D
C	TCK		Clock	C2CK
I	TDO			
L	TMS			
T				
S				
R				

Tab. 7: Popis propojení č.5

Pin	RL78	RX600	AT89C51CC01UA
P	RESET	RES#	RESET
-			
VDD	VDD	VCC	VCC
GND	VSS	VSS	VSS
D	TOOL0	RXD1	RxD
C		TXD1	TxD
I		MD	EA
L		PC7/UB	PSEN
T			
S			
R			ALE

Tab. 8: Popis propojení č.6

Pin	AVR HVP	AVR UPDI	AVR UPDI+RST
P	RESET	UPDI / RESET	RESET
-			
VDD	VCC	VDD	VDD
GND	GND	GND	GND
D	SDI		UPDI
C	SDO		
I	SII		
L	SCI		
T			
S			
R			

Tab. 9: Popis propojení č.7

Pin	CH32V003	SPD5118	Z8F
P	NRST		RESET
-			
VDD	VDD	VDDSPD	VDD
GND	VSS	GND	VSS
D	SWIO	HSDA	DBG
C		HSCL	
I			
L		HSA	
T			
S			
R			

Tab. 10: Popis propojení č.7

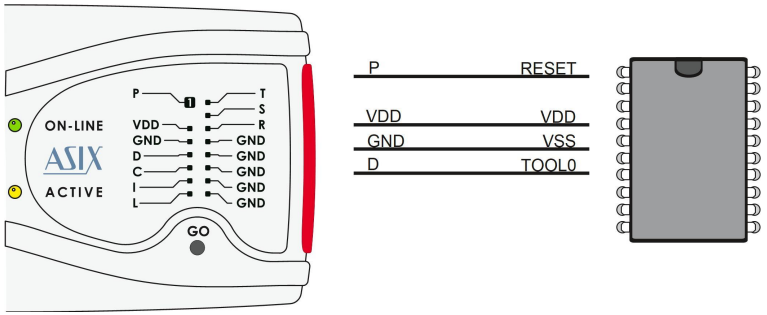
[1] Paměť je vhodné připojit plochým kabelem a připojit všechny jeho vodiče GND.

[2] Pro 1-Wire součástky je potřeba připojit externí pull-up rezistor, případně i Schottkyho diodu, viz kapitolu Rozhraní 1-Wire.

2.6.4 Příklady propojení

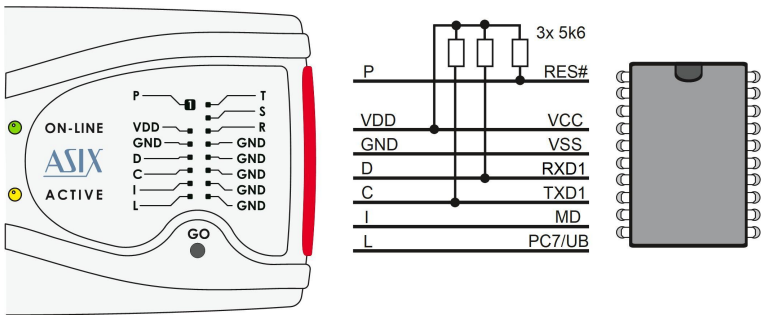
V následujícím textu naleznete příklady, jak propojit programátor FORTE a programovanou součástku. Na straně programované součástky používáme značení výrobce tak, jak je uvedeno v katalogovém listu dané součástky.

Připojení mikrokontrolérů RL78



Obr. 7: Mikrokontrolér Renesas RL78

Připojení mikrokontrolérů RX600

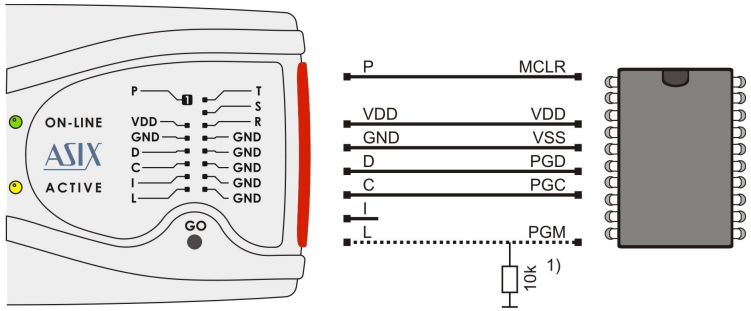


Obr. 8: Mikrokontrolér Renesas RX600

1) Mikrokontroléry RX600 se programují v Boot mode.

2) Pro naprogramování OSIS (ID) a zamčení součástky je potřeba použít nastavení v okně "Nastavení programátoru FORTE".

Připojení mikrokontrolérů PIC



Obr. 9: Mikrokontrolér PIC

1) Ne všechny součástky mají pin PGM. Pin PGM může být připojen buď k pinu L programátoru nebo prostřednictvím pull-down rezistoru připojen na VSS pro programování HVP nebo pull-up rezistorem na VDD pro programování LVP.

2) Pokud je paměť programu nebo datová paměť chráněna pojistkou CP nebo CPD, musí být celá součástka před programováním smazána.

3) Některé součásteky není možné smazat při naprogramované pojistce CP nebo CPD s napájecím napětím nižším než 5 V.

4) Pokud má mikrokontrolér více napájecích pinů VDD nebo VSS, všechny musejí být zapojené, včetně pinů AVDD a AVSS.

5) Při mazání mikrokontroléru v HVP módu může být smazána i pojistka LVP. Aby bylo možné opět programovat mikrokontrolér v LVP módu, musí být v HVP módu pojistka LVP znovu naprogramována.

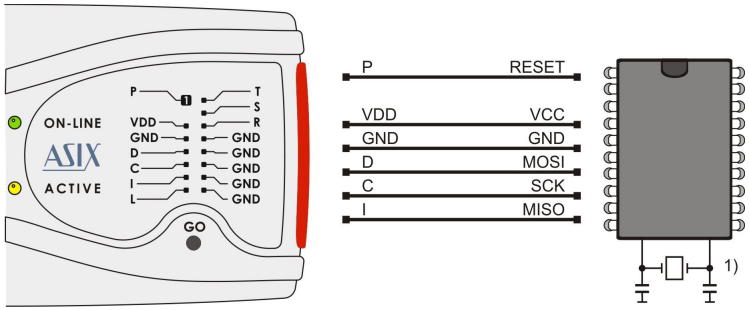
6) Programování součástek PIC32 je podporováno prostřednictvím rozhraní ICSP.

7) Součástky s pojistkou ICPORT musí mít pro LVP programování dedikovaný ICSP port vypnutý.

8) Součástky PIC24 a dsPIC33 mohou být programované s použitím PE (Programming Executive) nebo obvyklou metodou. Programování pomocí PE je obvykle rychlejší.

9) Pokud má součástka pojistkou zapnutý debug režim, programátor s ní nemůže komunikovat.

Připojení mikrokontrolérů AVR



Obr. 10: Mikrokontrolér AVR

1) Zdroj hodinového signálu, který je nastaven v součástce nebo který bude nastaven pojistkami během programování, musí být k součástce připojen. Krystal musí být připojen, pokud je nastavený jako zdroj hodin.

2) Pojistky součástky jsou od výrobce nastaveny na interní oscilátor o frekvenci 1 MHz. Při prvním programování je třeba programovat součástku s nastavením “Frekvence oscilátoru“ na “>750 kHz“ nebo nižší v okně ***Nastavení programátoru FORTE***.

3) K některým mikrokontrolérům AVR není možné připojit krystal (např. ATtiny13, ATtiny15).

4) Po správném nastavení pojistek součástky je potřeba kliknout pravým tlačítkem myši do okna ***Konfigurace*** a vybrat volbu ***Zapamatovat pojistky***. Tím dojde k uložení pojistek do souboru up.ini nebo do projektu, pokud se používá. Soubory .hex pro mikrokontroléry AVR totiž konfigurační pojistky neobsahují. Pokud je součástka programována z příkazového řádku, je potřeba použít soubor projektu .ppr s uloženými pojistkami.

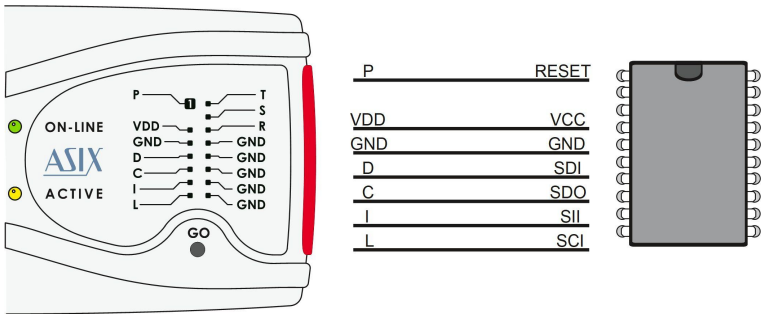
5) Zaškrtnutí volby ***Otevřít soubor s datovou pamětí automaticky*** v menu ***Soubor*** způsobí, že se data pro datovou paměť načtou současně s daty pro paměť programu.

6) Pokud je potřeba zachovat obsah datové paměti v mikrokontroléru, použijte pojistku EESAVE. Pokud je tato pojistka aktivní, naprogramujte mikrokontrolér příkazem ***Naprogramovat vše kromě datové paměti***, v opačném případě bude program UP hlásit chybu smazání datové paměti.

7) Pro konverzi mezi konektorem ICSP programátoru FORTE a 10 pinovým konektorem ISP firmy Atmel může být použit konvertor HPRAVR.

8) Některé součástky AVR mají ISP rozhraní vyvedené na jiných pinech než SPI rozhraní. Více informací je v datasheetu součástky v kapitole “Serial downloading”.

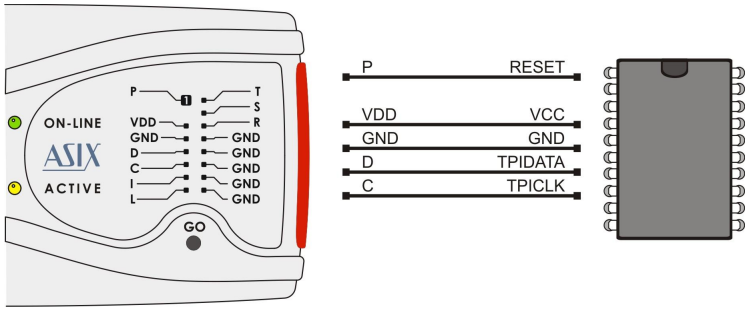
AVR v módu HVP



Obr. 11: AVR v módu HVP (např. ATtiny841)

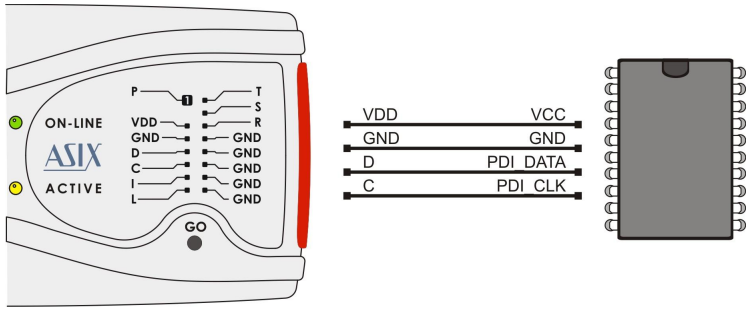
1) U některých součástek je potřeba zapojit ještě další piny na GND, to lze zjistit v datasheetu součástky na obrázku v kapitole "High-voltage Serial programming".

AVR s rozhraním TPI (např. ATtiny10)



Obr. 12: Mikrokontrolér AVR, rozhraní TPI

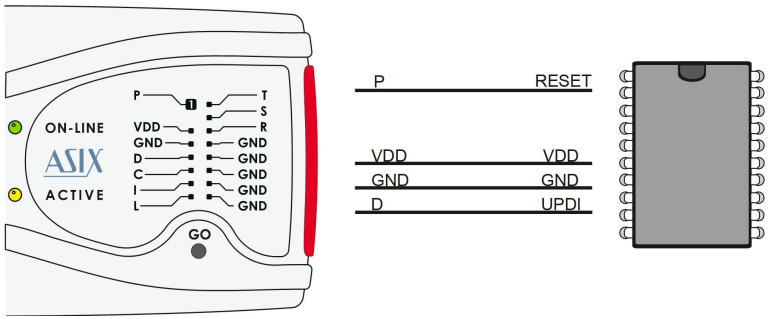
ATxmega s rozhraním PDI



Obr. 13: Mikrokontrolér ATxmega s rozhraním PDI

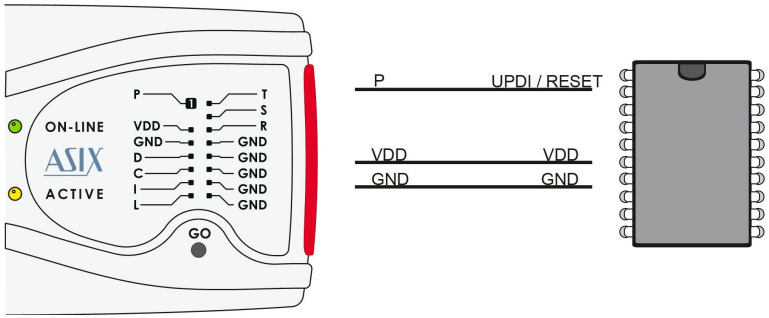
1) Pro programování prostřednictvím rozhraní JTAG musejí být součástky připojené, jak je popsáno v části Rozhraní JTAG.

AVR s piny UPDI a RESET



Obr. 14: Mikrokontrolér AVR s oddělenými piny UPDI a RESET

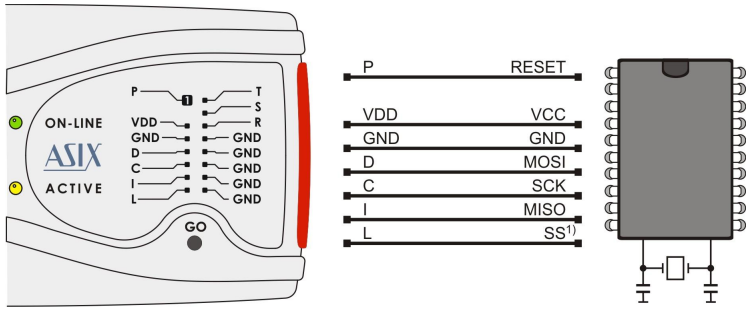
AVR s rozhraním UPDI



Obr. 15: Mikrokontrolér AVR s UPDI na stejném pinu jako RESET

1) ATtiny s UPDI vstupuje do programovacího módu s použitím 12 V pulsu na pinu UPDI.
2) Pokud součástka vypadne z programovacího módu nebo je během programování rozhraní UPDI zakázáno, je pro další komunikaci potřeba udělat power-on reset součástky.
3) Pro AVR s UPDI se očekávají soubory s paměťmi mapovanými od nuly a pojisky se ukládají do projektu, stejně jako u starších AVR, protože Atmel Studio takto soubory vytváří.

Atmel 8051



Obr. 16: Mikrokontrolér Atmel 8051

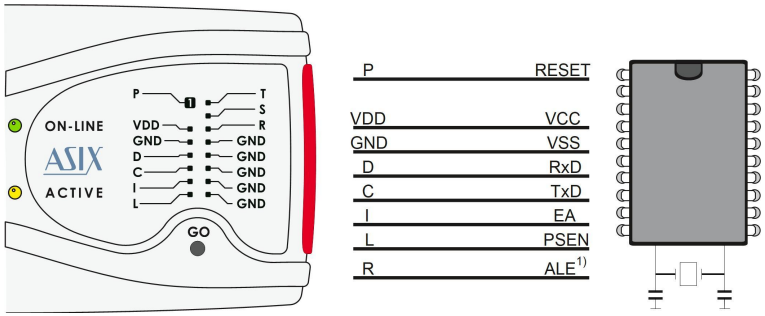
1) Pin SS musí být připojen pouze pro AT89LP2052 / 4052 / 213 / 214 / 216 / 428 / 828 / 6440 / 51RD2 / 51ED2 / 51ID2 / 51RB2 / 51RC2 / 51IC2.

2) AT89LP213, AT89LP214 a AT89LP216 mají inverzní RESET. Případný rezistor na pinu RESET musí být proto zapojený na VCC a ne na GND.

3) Programátor FORTE 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.

4) 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“. Pro AT89LP51RD2, AT89LP51ED2, AT89LP51ID2, AT89LP51RB2, AT89LP51RC2, AT89LP51IC2 software předpokládá pin POL v logické 0.

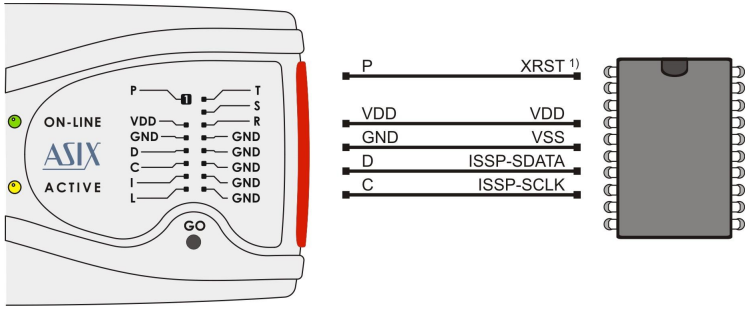
AT89C51CC01UA



Obr. 17: Mikrokontrolér AT89C51CC01UA

1) Pin ALE nemusí být připojen, pokud je na součástce nezapojený nebo je v log.1.

Cypress PSoC



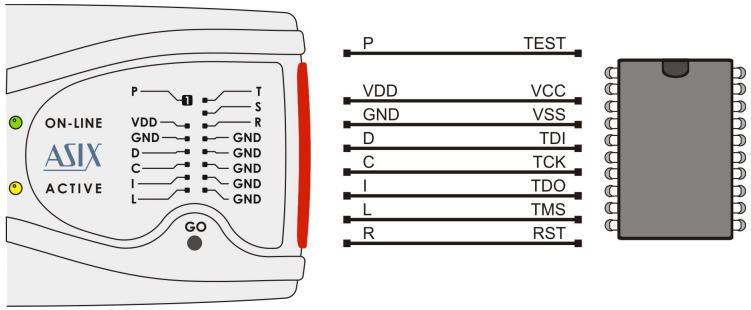
Obr. 18: Mikrokontrolér Cypress PSoC

1) Způsob inicializace programovacího módu je možné vybrat v okně ***Nastavení programátoru FORTE***.

Součástky bez XRST pinu mohou použít pouze inicializaci power-on resetem (napájením). Součástky s pinem XRST mohou použít obě metody, ale metoda inicializace signálem reset je lepší, neboť je možné použít ji i s externím napájením.

2) Položka ***Algoritmus programování*** v okně ***Nastavení programátoru FORTE*** by měla být nastavena podle použitého napájecího napětí.

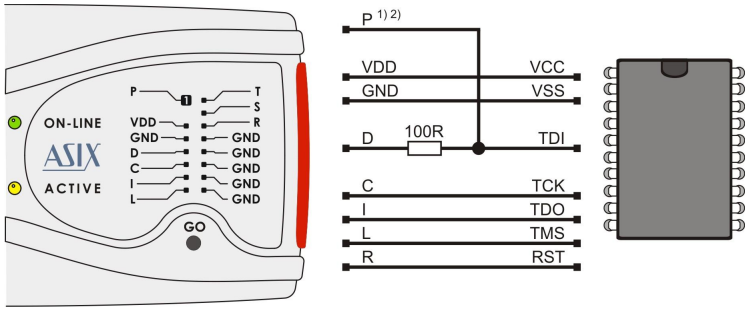
MSP430 / CC430 s pinem TEST, rozhraní JTAG



Obr. 19: Mikrokontrolér MSP430 / CC430, TEST pin a rozhraní JTAG

1) Pokud jsou kalibrační hodnoty oscilátoru uloženy v informační paměti a tato paměť nebude přeprogramována (smazána) během programování, součástka by měla být programována s vybranou volbou ***Interní kalibrovaný RC oscilátor*** v okně ***Nastavení programátoru FORTE***. V ostatních případech by měla být zvolena položka ***Interní nekalibrovaný RC oscilátor***.

MSP430 / CC430 bez pinu TEST, rozhraní JTAG



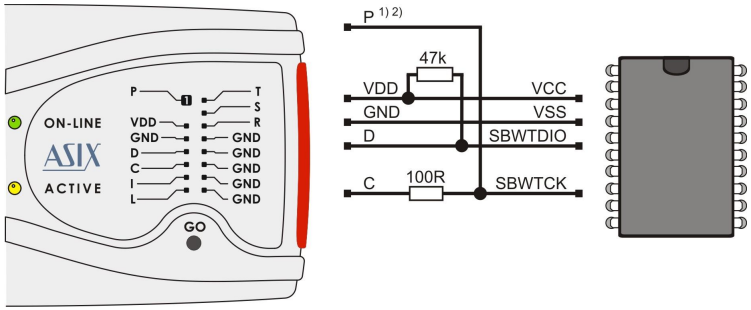
Obr. 20: Mikrokontrolér MSP430 / CC430 bez pinu TEST, rozhraní JTAG

1) Pin P dodává součástce 6,5 V během programování pojistky. V případě, že pojistka nebude programována, tento signál nemusí být zapojen.

2) Součástky MSP430F5xxx a CC430 se zamykají jiným způsobem, signál P zůstane nezapojený. V tomto případě může být vynechán i rezistor 100 R.

3) Pokud jsou kalibrační hodnoty oscilátoru uloženy v informační paměti a tato paměť nebude přeprogramována (smazána) během programování, součástka by měla být programována s vybranou volbou ***Interní kalibrovaný RC oscilátor*** v okně ***Nastavení programátoru FORTE***. V ostatních případech by měla být zvolena položka ***Interní nekalibrovaný RC oscilátor***.

MSP430 / CC430 s rozhraním SBW



Obr. 21: Mikrokontrolér MSP430 / CC430, rozhraní SBW

1) Pin P dodává součástce 6,5 V během programování pojistky. V případě, že pojistka nebude programována, tento signál nemusí být zapojen.

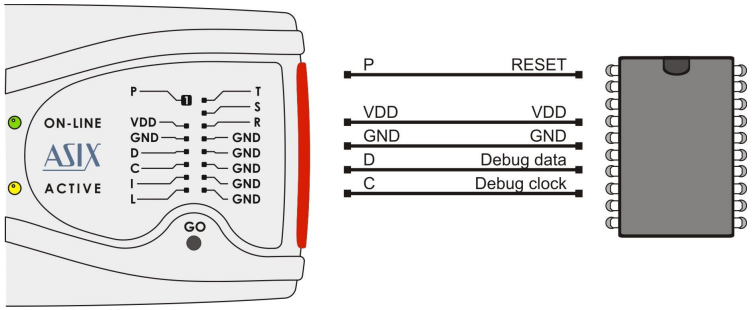
2) Součástky MSP430F5xxx a CC430 se zamykají jiným způsobem, signál P zůstane nezapojený. V tomto případě může být vynechán i rezistor 100 R.

3) Pokud jsou kalibrační hodnoty oscilátoru uloženy v informační paměti a tato paměť nebude přeprogramována (smazána) během programování, součástka by měla být programována s vybranou volbou ***Interní kalibrovaný RC oscilátor*** v okně ***Nastavení programátoru FORTE***. V ostatních případech by měla být zvolena položka ***Interní nekalibrovaný RC oscilátor***. Pro MSP430F5xxx a CC430 se oscilátor nenastavuje.

4) Volba ***Rychlost*** v okně ***Nastavení programátoru FORTE*** umožňuje zpomalit komunikaci v případě, že je na pinu RESET kondenzátor.

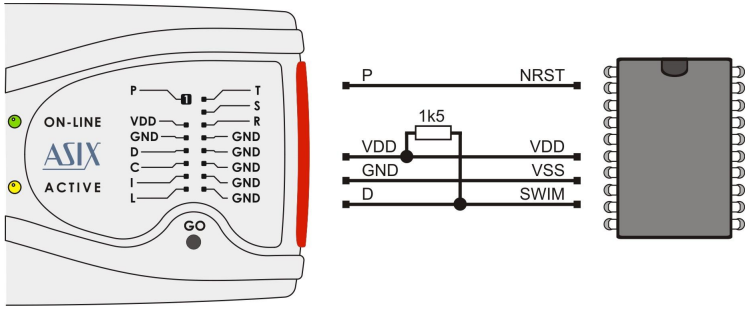
5) Součástky s kalibračními konstantami oscilátoru uloženými v informační paměti mají možnost pomocí volby ***Smazat Segment A*** vybrat, zda se bude mazat i sektor A informační paměti.

TI (Chipcon) CCxxxx



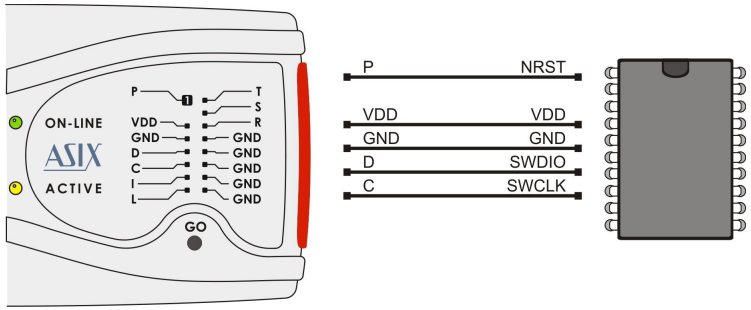
Obr. 22: Mikrokontrolér TI (Chipcon) CCxxxx

STM8



Obr. 23: Mikrokontroléry STM8

Mikrokontroléry ARM s rozhraním SWD

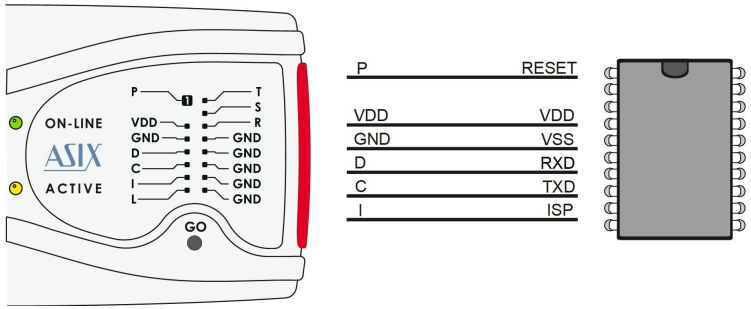


Obr. 24: Mikrokontroléry ARM s rozhraním SWD

1) Mikrokontroléry ARM, které nemají rozhraní SWD, se programují prostřednictvím rozhraní JTAG.

2) Mikrokontroléry Renesas s jádrem ARM a pojistkou OSIS se naprogramováním změněné hodnoty OSIS zamknou. Při vyplnění správné hodnoty OSIS u zamčené součástky lze se součástkou komunikovat. V případě smazání s vyplněnou správnou hodnotou se zámek OSIS zruší.

Mikrokontroléry LPCxxxx, rozhraní UART



Obr. 25: Mikrokontroléry LPCxxxx, rozhraní UART

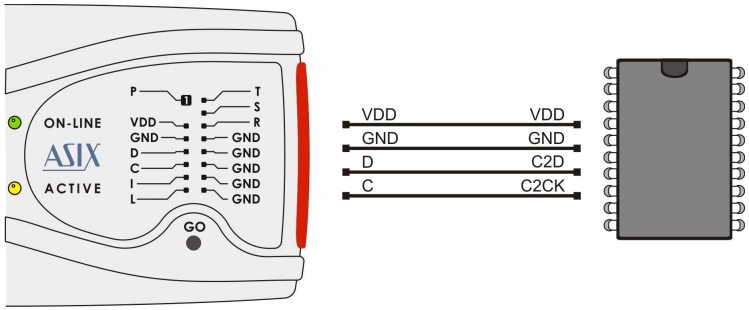
1) Podpora programování přes UART je implementována pro některé mikrokontroléry LPCxxxx jako alternativa k rozhraní SWD.

2) Signály používané jako ISP popisuje tabulka:

Mikrokontrolér	Signál ISP	Další signály
LPC1102	-	-
LPC11xx	PIO0_1	-
LPC11Cxx	PIO0_1	PIO0_3 = Log.1
LPC11Uxx, LPC13xx	PIO0_1	PIO0_3 = Log.0
LPC15xx	ISP_0	ISP_1 = Log.0
LPC17xx	P2.10	-
LPC18xx	P2_7	-
LPC5411x	PIO0_31	PIO0_4 = Log.1, PIO1_6 = Log.0

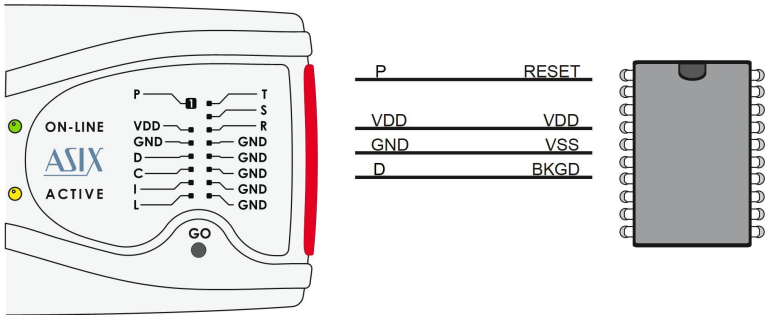
Tab. 11: Signál ISP

Mikrokontroléry C8051 a EFM8 s rozhraním C2



Obr. 26: Mikrokontroléry C8051 a EFM8 s rozhraním C2

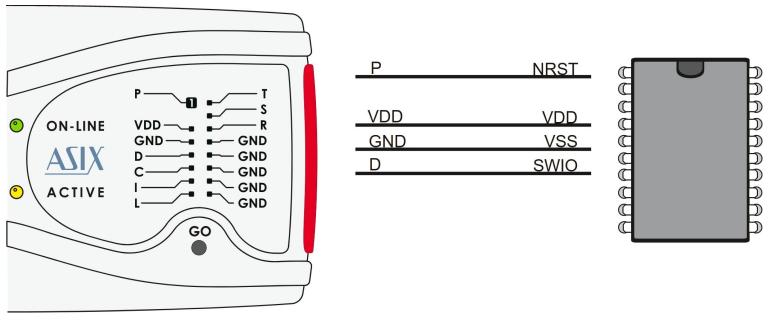
Mikrokontroléry HCS08



Obr. 27: Mikrokontroléry HCS08

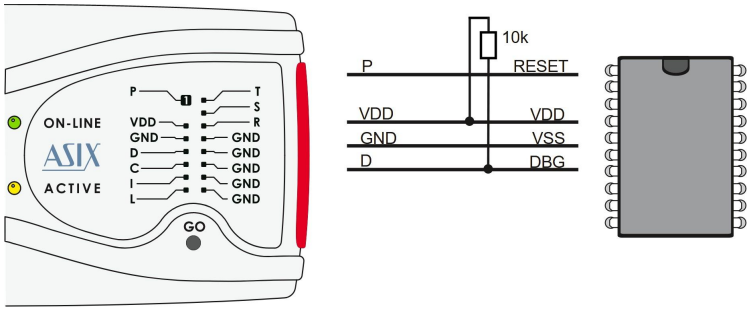
1) U součástek, kde nejde do programovacího módu vstoupit s využitím signálu RESET, programátor využívá pro vstup do programovacího módu power-on reset. Power-on reset nelze automaticky provést při ovládání z příkazového řádku, pokud uživatel používá externí napájení. V tom případě je třeba, aby uživatel zajistil, že během připojovaní napájecího napětí k součástce bude na pinu BKGD log. 0.

CH32V003



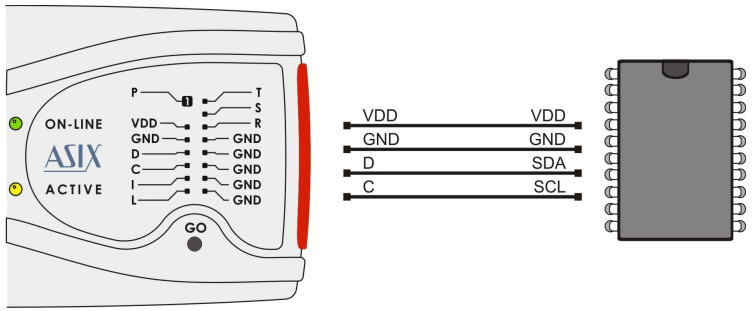
Obr. 28: Mikrokontrolér CH32V003

Z8F



Obr. 29: Mikrokontrolér Z8F

Paměti I2C



Obr. 30: Paměti I2C

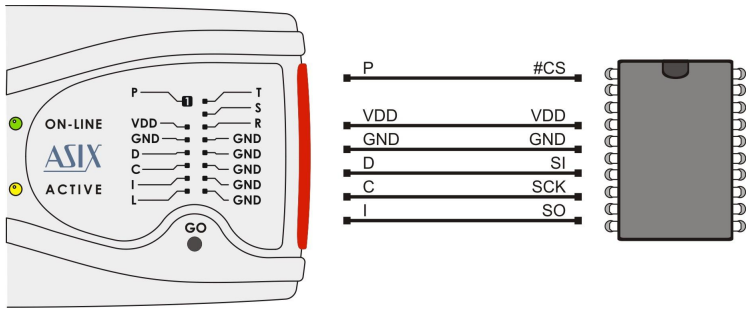
1) Programátor používá na datovém vodiči (SDA) interní Pull-Up rezistor 2,2 kΩ, když pracuje se součástkou komunikující po I^2C. Tento interní rezistor lze vypnout v okně "Nastavení programátoru FORTE".

2) Pokud je programovaná součástka 24LC(S)21A nebo 24LC(S)22A, její VCLK pin musí být v průběhu programování připojen na VDD.

3) Paměti 34xx02 potřebují na pinu A0 „vysoké“ napětí pro příkazy ochrany proti zápisu SWP a CSWP. Toto napětí je generováno na pinu P, který se musí v tomto případě na pin A0 připojit. Piny paměti A0, A1 a A2 musejí být zapojeny manuálně podle zvoleného módu ochrany.

4) Paměť 34AA04 potřebuje na pinu A0 „vysoké“ napětí pro programování konfigurační paměti. Toto napětí je generováno na pinu P, který se musí v tomto případě na pin A0 připojit.

Paměti SPI



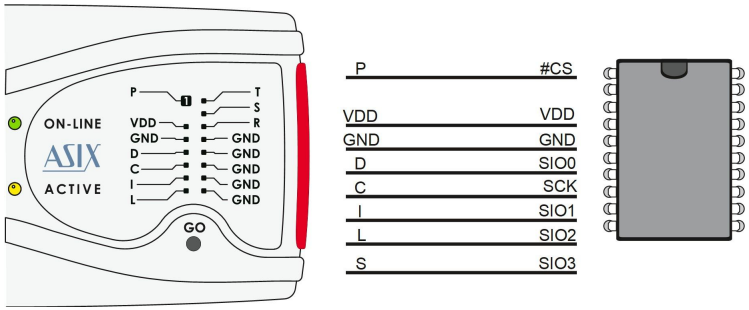
Obr. 31: Paměti SPI

1) Některé součástky mají piny WP, HOLD nebo RESET, všechny tyto piny musejí být zapojeny na potřebnou logickou úroveň tak, aby neblokovaly komunikaci nebo programování součástky.

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

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

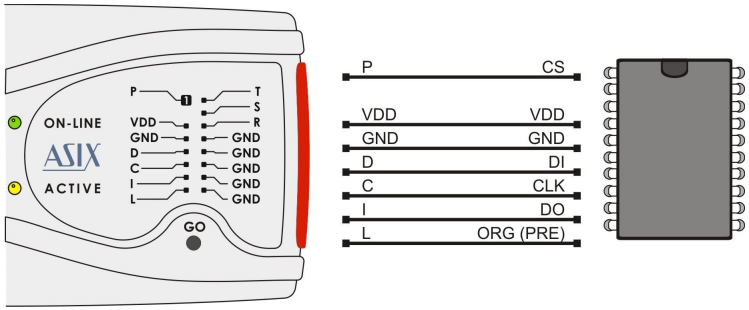
Paměť SPI, rozhraní QUAD



Obr. 32: Paměť SPI, rozhraní QUAD

1) Paměť je vhodné připojit plochým kabelem a připojit všechny jeho vodiče GND.

Paměti Microwire

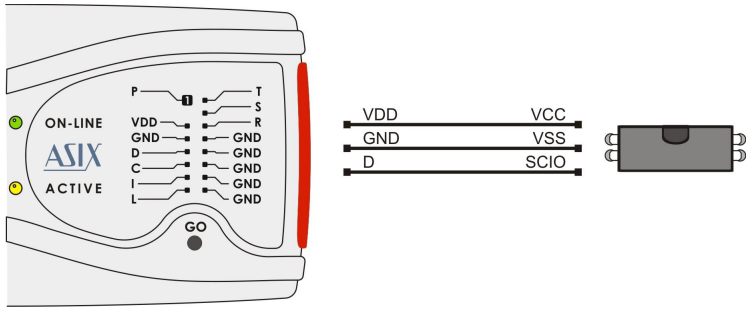


Obr. 33: Paměti Microwire

1) Pin L vybírá organizaci paměti jako buď 8 bitů nebo 16 bitů na slovo. Uživatel vybere organizaci v programu UP a programátor FORTE potom nastaví tento pin na příslušnou logickou úroveň. Pokud je tento pin paměti pevně zapojený v aplikaci na patřičnou logickou úroveň, pin L programátoru zůstane nezapojený.

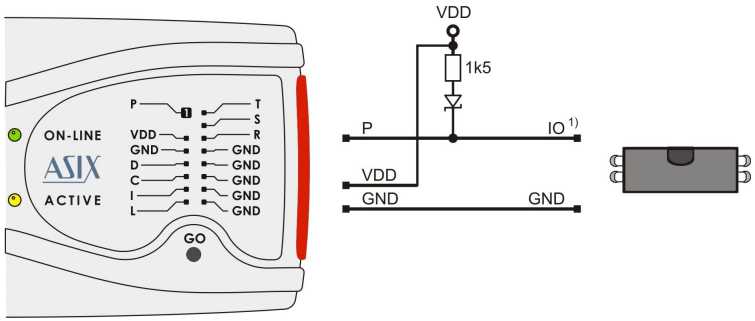
2) V případě použití s pamětí M93Sx6 je nutné pin L připojit na pin PRE součástky, a pak slouží k výběru Protection registru.

Paměti UNI/O



Obr. 34: Sériové paměti UNI/O

Rozhraní 1-Wire



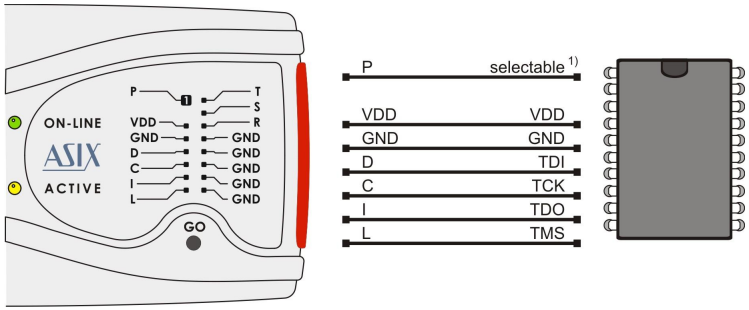
Obr. 35: Součástky s rozhraním 1-WIRE

1) Schottkyho dioda je nutná jen u součástek, kde je pro programování potřeba vyšší než napájecí napětí, např. DS2505 nebo DS2406. Pull-up rezistor je potřeba vždy.

2) Pro DS1821: Pokud je součástka v módu termostat, její pin VDD musí být připojen na pin D programátoru, ale externí napájecí napětí nesmí být připojeno na pin D programátoru, smí být připojeno pouze na pin VDD. V tomto případě smí být součástka programována pouze samostatně!

3) Pro DS28E05: Pro součástku je potřeba použít menší hodnotu pull-up rezistoru, např. 560R.

Rozhraní JTAG



Obr. 36: Součástky s rozhraním JTAG

1) V JTAG Playeru je pin P konfigurovatelný, může být nastaven tak, že během programování drží součástku v resetu, to je potřeba např. pro součástky ATmega.

2) **Programátor vždy používá externí napájecí napětí při programování souborů typu SVF nebo XSVF utilitou JTAG Player**.

3) Mikrokontroléry AVR32 se programují prostřednictvím rozhraní JTAG programem UP. Během programování nesmí být součástka v resetu.

4) Mikrokontroléry ATxmega, které mají rozhraní JTAG, je možné programovat prostřednictvím tohoto rozhraní programem UP. Pin P není k programování potřeba.

5) Mikrokontroléry s jádrem ARM je možné programovat prostřednictvím rozhraní JTAG programem UP.

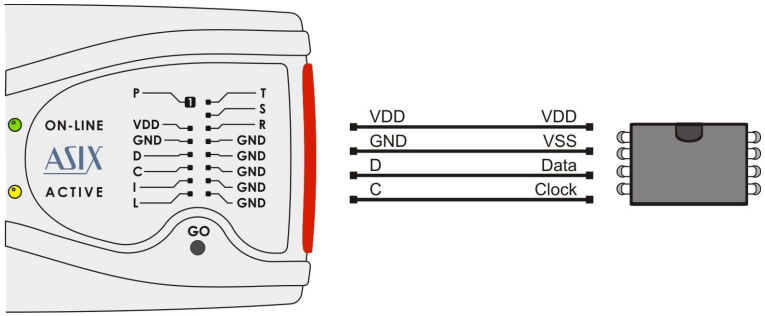
a) Součástky, které mají rozhraní SWD, např. většina součástek s jádrem Cortex-M3, se programují prostřednicvím tohoto rozhraní, viz zapojení.

b) V okně ***Nastavení programátoru FORTE*** je třeba vybrat zdroj hodinového signálu, který je k součástce připojený a nastavit správně jeho frekvenci.

c) Pin P není pro programování potřeba, lze ho připojit na pin NRST součástky, aby mohla být po naprogramování resetována a program se rozeběhl.

d) *Mikrokontroléry NXP LPC2xxx:* Pro správnou funkci je potřeba spojit piny /RESET a /TRST a přivést je na pin P programátoru FORTE. Dále je nutné připojit externí pull-down rezistor 4,7 kΩ až 10 kΩ na pin /RTCK. Mikrokontroléry LPC2xxx po resetu spouští bootloader. Bootloader poté kontroluje pin P0.14 nebo P2.10 (podle typu součástky, viz datasheet součástky), podle jeho stavu rozhodne, zda po resetu spustit uživatelskou aplikaci (log. 1 pro uživatelskou aplikaci, log. 0 pro bootloader). Kvůli této vlastnosti je potřeba na tento pin připojit externí pull-up rezistor, aby byla spuštěna uživatelem naprogramovaná aplikace.

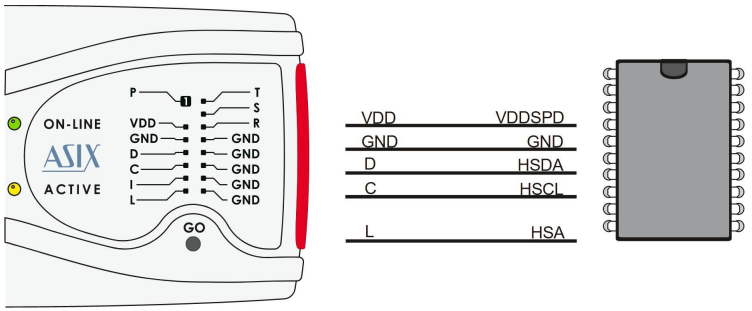
HCSxxx



Obr. 37: Součástky HCSxxx

1) Signál Clock je obvykle na pinu S2, signál Data na pinu PWM.

SPD5118



Obr. 38: SPD5118

1) V offline mode musí být použito interní napájení z programátoru, protože je nutné udělat POR s pinem HSA připojeným na GND. Pro vstup do tohoto módu lze připojit pin HSA součástky na pin L programátoru. V normálním módu nemusí být pin HSA k programátoru připojen. Ochrana proti zápisu může být zrušena pouze v offline mode v registrech MR12, MR13.

2.7 Technická specifikace

2.7.1 Mezní hodnoty

Pracovní teplota	min. 0 °C	max. +55 °C
Skladovací teplota	min. -40 °C	max. +85 °C
Napětí libovolného pinu[1]	min. -0,5 V	max. 6,5 V
Maximální proud I/O pinu	50 mA	
ESD ochrana (HBM model)	±4 kV kontakt	
	±8 kV vzduch	

Tab. 12: Mezní hodnoty

[1] Pin "P" nakonfigurovaný jako výstup poskytuje napětí v rozsahu +6,5 V až +17 V

2.7.2 Provozní specifikace

Důležité upozornění

Při nedodržení zde specifikovaných parametrů může dojít ke zničení programátoru nebo připojeného počítače.

Napájecí napětí VDD dodané z programátoru	1,8 V až 5,5 V
Napájecí napětí VDD při napájení z aplikace	1,8 V až 5,5 V
Externí napájecí napětí VDD pro komunikaci omezenou rychlostí[1]	1,2 V až 5,5 V
Maximální proud odebíraný z VDD	100 mA
Maximální proud odebíraný z VPP	100 mA při 7 V
	10 mA při 17 V
Maximální proud odebíraný z I/O pinu	4 mA @ VDD = 1,8 V
	16 mA @ VDD = 4,5 V
Maximální proud odebíraný ze všech I/O pinů zároveň	100 mA
Dovolené vstupní napětí na pinech	0 až 5,5 V
Výstupní napětí pinu P	Nastavitelné od 6,5 V do 17 V nebo logické úrovně
Vstupní napětí VIL	max. VDD x 0,3 V
Vstupní napětí VIH	min. VDD x 0,7 V
Výstupní napětí VOL	max. 0,55 V při VDD=4,5 V a I=4 mA
	typ. 0,1 V
Výstupní napětí VOH	min. 3,8 V při VDD=4,5 V a I=4 mA
	typ. VDD - 0,1 V
Odolnost na zkrat	trvalá
Operační systém	Windows[2] 32/ 64-bit

USB kompatibilita	USB 2.0 High-Speed (480 Mbps)
USB konektor	typ B
Rozměry programátoru	112 x 64 x 22 mm
Hmotnost programátoru	60 g
Celková hmotnost balení	230 g

Tab. 13: Provozní specifikace

[1] Programátor typicky dokáže omezenou rychlostí komunikovat již při nižším napětí aplikace
[2] Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Windows 11

2.7.3 Prohlášení o shodě a RoHS

Dokumenty prohlášení o shodě a o splňované direktivě RoHS jsou ke stažení na www.asix.tech v sekci Dokumenty.

3

OVLADAČE

V této kapitole se budeme zabývat instalací a aktualizací ovladačů.

3.1 Instalace ovladačů

3.1.1 Operační systém Windows

Pro instalaci a pro první spuštění programu UP musí mít uživatel administrátorská práva. Při dalším použití budou dostačující práva běžného uživatele.

Ovladače pro programátor FORTE se nainstalují během instalace programu UP.

Windows 7 a novější

Nejprve nainstalujte program UP. Instalátor tohoto obslužného software nainstaluje také USB ovladač programátoru FORTE. Instalátor programu UP najdete na www.asix.cz/dwnld_up.htm.

Po dokončení instalace připojte programátor FORTE k počítači. Po chvíli by se měla rozsvítit zelená ON-LINE LED a ve Správci zařízení bude možné najít programátor jako správně nainstalovaný.

Starší podporované verze Windows

Ovladač obsažený v instalátoru programu UP je určen pro Windows 7 a novější.

Pro starší verze operačního systému Windows je třeba stáhnout ovladač z webu www.asix.cz, z download sekce programátoru a někam ho rozbalit. Po připojení programátoru se operační systém zeptá na ovladač, v dialogu "Nalezen nový hardware" je třeba nastavit cestu k rozbalenému ovladači.

Během instalace se operační systém zeptá, zda má nainstalovat software, který neprošel testem Microsoftu pro operační systém Windows. Instalaci je třeba odsouhlasit.

Obr. 39: Dialog Test kompatibility

Po úspěšné instalaci se rozsvítí zelená ON-LINE LED na programátoru a ve Správci zařízení bude možné najít programátor jako správně nainstalovaný.

3.2 Aktualizace ovladačů

Programátor FORTE komunikuje s PC prostřednictvím USB obvodu firmy FTDI (www.ftdichip.com), která pro tyto obvody vyvíjí také ovladače.

Aktuální ovladače jsou vždy součástí instalačního balíku programu UP.

Pro operační systém Windows jsou ovladače již dlouhou dobu stabilní a většinou není nutná jejich aktualizace, pokud si to nevyžadují další aplikace používající obvody FTDI na daném PC.

Pokud přesto potřebujete ovladač aktualizovat, nejjednodušším způsobem, jak to udělat, je aktualizace programu UP.

Z webu si stáhněte nejnovější instalátor programu UP a bez rizika že přijdete o pracně vytvořené nastavení programu či jednotlivých projektů nainstalujete novou verzi programu, která původní verzi nahradí.

4

Použití pod OS Linux

Podpora produktů ASIX pod OS Linux byla ukončena.

Poslední verze, kde jsme naše produkty testovali je UBUNTU 20.04.2 LTS a Wine 5.0.

Podle informací od zákazníků lze naše produkty úspěšně používat ještě do Wine 6.0, ve vyšších verzích už ne.

Programy pro programátory mohou běžet v operačním systému Linux pod Wine. Pro přístup k USB zařízením lze použít libftd2xx.

Krok 1: Instalace libftd2xx a libftchipid

Vždy instalujte 32-bitové verze libftd2xx a libftchipid od FTDI a to i když používáte 64-bitový kernel. Aplikace jsou 32-bitové a proto potřebují ke svému běhu 32-bitové knihovny.

Ovladač lze nalézt na webu firmy FTDI v sekci "Drivers/ D2XX Drivers".

- Rozbalte *libftd2xx.so.1.1.0* (v případě novější verze místo 1.1.0 uveďte číslo aktuální verze) a *libftchipid* a zkopírujte soubory *libftd2xx.1.1.0.so* a *libftchipid0.1.0* to adresáře s 32-bitovými knihovnami (typicky */usr/lib/i386-linux-gnu/*).
- *ln -s libftd2xx.so.1.1.0 /usr/lib/i386-linux-gnu/libftd2xx.so.1* (obvykle stačí zavolat ldconfig)
- *ln -s libftd2xx.so.1.1.0 /usr/lib/i386-linux-gnu/libftd2xx.so.0* (musí být provedeno ručně)
- *ln -s libftchipid.so.0.1.0 /usr/lib/i386-linux-gnu/libftchipid.so.0* (obvykle stačí zavolat ldconfig)
- Knihovny hledají zařízení v */dev/bus/usb*. Zkontrolujte prosím, zda v adresáři */dev/bus/usb* se skutečně vyskytují zařízení pro přístup k USB.
- Zkontrolujte, zda je programátor rozpoznaný v systému (použijte příkaz *lsusb*).
- Zkontrolujte přístupová práva k příslušným souborům v */dev/bus/usb* (příkaz *ls -la /dev/bus/usb/*). Pravděpodobně bude pro vašeho uživatele chybět právo přístupu r+w.
- Pokud vám chybějí práva a používáte udev:

Vytvořte nový soubor v adresáři s pravidly udev */etc/udev/rules.d* nebo */lib/udev/rules.d* (Podle zvyku vaší distribuce). Vhodné jméno pro soubor je například 51-asix_tools.rules. Do souboru vložte následující řádky :

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="f1a0", MODE:="0666", SYMLINK+="asix_presto"
SUBSYSTEMS=="usb", ATTRS{idVendor}=="a600", ATTRS{idProduct}=="a000", MODE:="0666", SYMLINK+="asix_sigma"
SUBSYSTEMS=="usb", ATTRS{idVendor}=="a600", ATTRS{idProduct}=="a003", MODE:="0666", SYMLINK+="asix_forte"
SUBSYSTEMS=="usb", ATTRS{idVendor}=="a600", ATTRS{idProduct}=="a004", MODE:="0666", SYMLINK+="asix_omega"
```

Hodnoty VID a PID jsou přidělovány výrobci a seznam připojených zařízení zjistit příkazem lsusb.

Step 2: Instalace wine

Je potřeba nainstalovat 32-bitovou verzi wine (například *wine-1.4:i386*).

Verze Wine nad 6.0 nejsou podporované.

Step 3: Instalace lin_ftd2xx

Knihovna lin_ftd2xx je dostupná na webu firmy ASIX.

Zkontrolujte hodnotu proměnné prostředí *WINEDLLPATH*. Měla by obsahovat cestu, kde jsou 32-bitová wine DLL, typicky ***/usr/lib/i386-linux-gnu/wine***. Knihovnu lin_ftd2xx nainstalujte do tohoto adresáře.

Je doporučeno nainstalovat také Microsoft™ TrueType core fonts. Tyto fonty lze nainstalovat pomocí balíčku msttcorefonts z repozitáře Ubuntu.

Poznámka:

Knihovna libftd2xx vyžaduje během otevírání zařízení programátoru nebo logického analyzátoru též přístupová práva ke všem ostatním zařízením s čipem FTDI, aby se ujistil, že otevírá to správné zařízení.

5

PROGRAM UP

UP je řídicí software pro programátory firmy ASIX. Program nabízí mnoho pokročilých funkcí a umožňuje ovládání programovacího procesu jak z prostředí programovacího software, tak i vzdáleně z příkazového řádku, pomocí zpráv Windows a knihovny DLL.

5.1 Použité zkratky

Menu → položka	tučnou kurzívou se znakem → jsou uvedeny odkazy na konkrétní položku v menu nebo názvy karty (záložky) konkrétního okna

5.2 Instalace programu UP

Instalace je velmi jednoduchá. Instalační program lze nalézt na www.asix.cz. Spusťte instalátor (UP_xxx_CZ.EXE, kde xxx znamená 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. FORTE) 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.

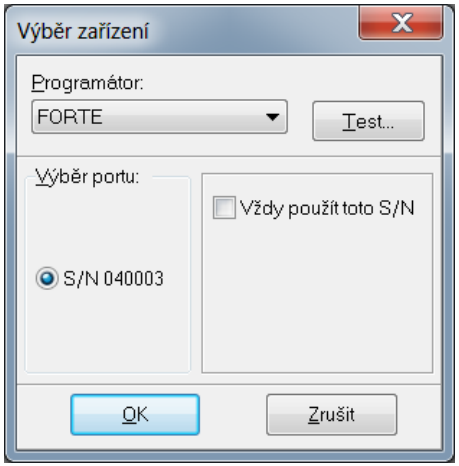
5.3 Programování součástky

V následujícím textu popíšeme způsob jak naprogramovat součástku a upozorníme na co si dát během programování pozor.

5.3.1 Volba programátoru

Dříve než můžeme začít programovat součástku, musíme vybrat programátor, kterým budeme součástku programovat. V současnosti je možné vybrat buď programátor PRESTO nebo programátor FORTE.

Volbu programátoru provedeme pomocí ***Nastavení → Výběr zařízení a portu*** nebo dvojklikem na jméno vybraného programátoru, který je zobrazen v pravém horním rohu okna programu. Zobrazí se následující dialog:



Obr. 40: Volba programátoru

Pokud je programátor připojený k počítači a nekomunikuje s ním jiný program, je zobrazeno jeho

sériové číslo. Tlačítkem **Test** můžeme otestovat komunikaci s vybraným programátorem.

Pokud je zaškrtnuta volba **Vždy použít toto S/N**, použije se aktuálně vybraný programátor i při použití souboru projektu, ve kterém je uloženo sériové číslo jiného programátoru. Před touto volbou má prioritu sériové číslo definované na příkazovém řádku.

5.3.2 Projekty

V programu UP je doporučené používat pro programování součástek projekty.

Do projektů se ukládají veškerá nastavení, která bezprostředně souvisejí s programováním vybrané součástky jako například typ součástky, požadované napětí, způsob verifikace, jméno souboru s programovanými daty a mnoho dalších důležitých informací.

Nový projekt může být vytvořen volbou ***Soubor → Nový projekt***.

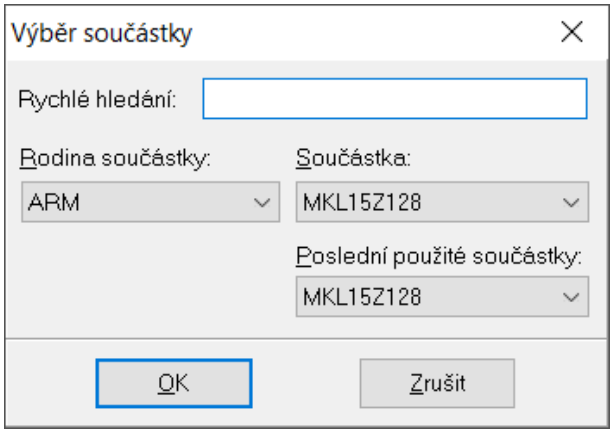
Následuje výběr typu součástky a souboru s daty pro programování, volba nastavení konfiguračního slova součástky, volba použitého napětí a dalších důležitých parametrů.

Po dokončení všech požadovaných nastavení je třeba projekt uložit volbou ***Soubor → Uložit projekt***.

Existující projekt může být otevřen kliknutím na ***Soubor → Otevřít projekt***.

5.3.3 Výběr typu součástky

Typ součástky vybereme volbou ***Součástka → Výběr součástky*** nebo dvojklikem na jméno součástky, které je zobrazeno v pravém horním rohu okna programu.



Obr. 41: Výběr součástky

K dispozici je filtr zobrazení po rodinách součástek a filtr rychlého hledání, kam stačí napsat jen významnou část jména součástky, a tím podstatně zúžit a zrychlit výběr součástky.

Filtr rychlého hledání umožňuje část jména nahradit otazníkem jako např. PIC18?20.

K dispozici je také seznam 10 posledních použitých součástek, ze kterých je možné jednu vybrat.

5.3.4 Nastavení programu

Aby chování programu vyhovovalo potřebám uživatele, je vhodné přizpůsobit jeho nastavení volbou ***Nastavení → Nastavení programu***.

Možností nastavení je velké množství. Pokud si nejste jisti, že všechna nastavení jsou v pořádku, je vhodné použít volbu ***Základní nastavení***, která vrátí všechna nastavení do počátečního stavu.

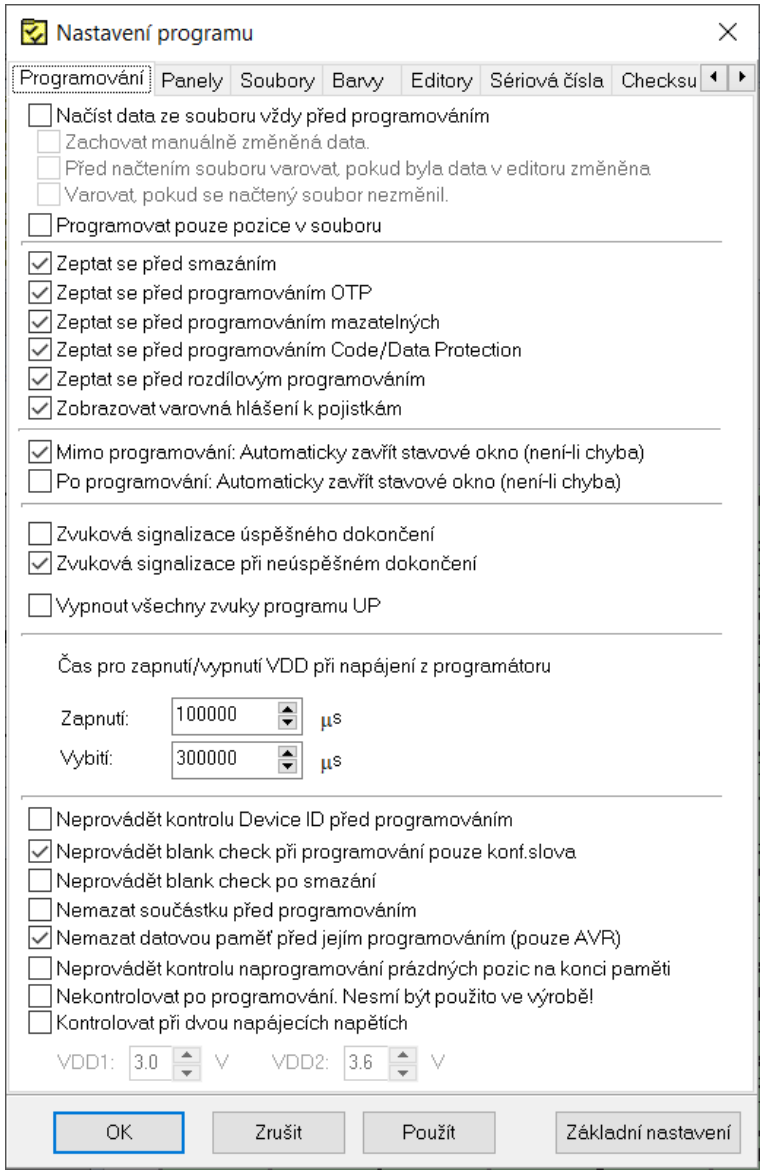
Pokud aktualizujete program UP na novou verzi, zůstávají všechna doposud provedená nastavení zachována.

Detailní popis všech nastavení je v kapitole Menu programu UP, zde se zmíníme jen o některých důležitých nastaveních.

Čas pro zapnutí/vypnutí VDD při napájení z programátoru

Pokud používáte napájení aplikace během programování z programátoru a program kvůli nabíjení kondenzátorů v aplikaci hlásí nadproud na napájecím napětí, může být užitečné prodloužit čas zapnutí v nastavení ***Nastavení programu → Programování → Čas pro zapnutí/ vypnutí VDD při napájení z programátoru*** tak, aby programátor stihl nabít velké kondenzátory v aplikaci.

Na stejném místě je možné nastavit čas pro vypnutí interního napájení, pokud při vypnutí program hlásí, že napětí nebylo dostatečně vybito.



Obr. 42: Nastavení programování

Nastavení pro produkční programování

Pokud chcete pro produkci dokonale ověřit, že součástka je správně naprogramovaná, je možné použít volbu ***Nastavení programu → Programování → Kontrolovat při dvou napájecích napětích*** a jako meze použít nejnižší a nejvyšší dovolené napětí součástky. Někteří výrobci tuto kontrolu pro produkční programování doporučují. Tato funkce je dostupná pouze při napájení z programátoru FORTE.

Užitečnou pomůckou může být ***Nastavení programu → Panely → Zobrazit čítač sériové výroby***, který přehledně monitoruje počty úspěšně či neúspěšně naprogramovaných součástek. Čítač je možné vynulovat volbou ***Součástka → Programovat → Sériová výroba → Vynulovat čítače***.

Dalším výborným pomocníkem může být ***Nastavení programu → Sériová čísla → Zaznamenávat do souboru***. Použitím této volby se do vybraného souboru začnou ukládat informace o průběhu programování jednotlivých součástek.

Pokud si přejete během produkčního programování automaticky vkládat sériové číslo, máte velké množství možností, jak bude sériové číslo vypadat a kde v paměti bude umístěno. Všechna potřebná nastavení sériových čísel najdete v panelu ***Sériová čísla***.

Více informací k sériovým číslům je v kapitole Sériová čísla.

Nastavení pro programování během vývoje

Pokud často měníte obsah programovaných dat např. během vývoje aplikace, máte možnost zpříjemnit si práci použitím tlačítka **GO**, které standardně slouží k naprogramování a verifikaci celého obsahu paměti. Funkce tlačítka je volitelná a nastavuje se pomocí menu ***Nastavení → Klávesové zkratky*** v části ***Tlačítko GO***.

Použití tlačítka GO pro programování by mělo doprovázet ***Nastavení → Nastavení Programu → Programování → Načíst data ze souboru vždy před programováním***.

Nastavení → Nastavení Programu → Programování → Zachovat manuálně změněná data způsobí, že manuálně změněná data nejsou přepsána automatickým načtením souboru před programováním.

Nastavení → Nastavení Programu → Programování → Před načtením souboru varovat, pokud byla data v souboru změněna vyvolá varování, pokud byla data v některém z editorů změněna a je současně aktivní volba pro automatické načtení souboru.

Nastavení → Nastavení Programu → Programování → Varovat, pokud se načtený soubor nezměnil vyvolá varování, se před programováním opakovaně načetl soubor se stejným obsahem.

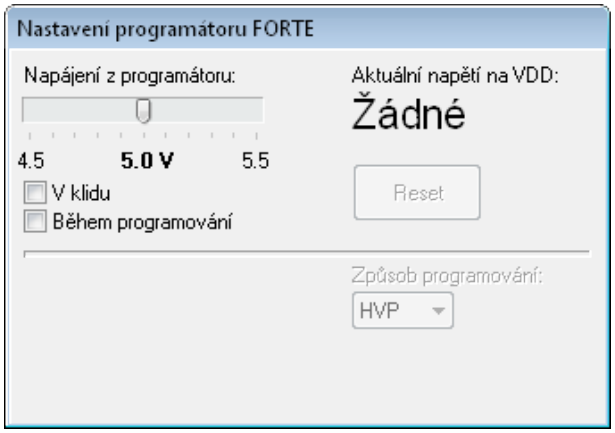
Vhodné může být také ***Nastavení Programu → Programování → Po programování: automaticky zavřít stavové okno, není-li chyba***.

V některých speciálních případech nemají vývojáři k dispozici napájecí pin programované součástky v aplikaci, která je napájená z externího napětí. Aby byly výstupní obvody programátoru napájené, je nutné v tomto případě zapnout napájení z programátoru a nastavit jeho velikost na stejnou hodnotu, jaká je použitá v aplikaci. Protože ale přes programovací piny do programátoru pronikne nějaké napětí z aplikace, programátor vidí, že je napětí přítomné a brání se možnosti aktivovat svoje výstupní napětí. Pro tento případ slouží ***Nastavení Programu → Ostatní → Dovolit kolizi interního napájecího napětí s externím***.

VAROVÁNÍ: Pokud by tato volba byla použita v jiném případě, hrozí zničení programátoru!

Nastavení programátoru

Kdykoli je vybrán konkrétní programátor (PRESTO nebo FORTE), je zobrazeno také okno Nastavení programátoru, kde je možné nastavit používaný zdroj napětí a některé další důležité volby pro programování.



Obr. 43: Nastavení programátoru

Pokud používáte externí napájení aplikace je nutné zrušit nastavení napájení ***Během programování***.

Aktuální velikost napětí se neustále zobrazuje v pravém horním rohu tohoto okna.

Pokud vznikne nějaká chyba nebo přetížení, zobrazí se na tomto místě výstražné upozornění.

Měnit velikost výstupního napětí v klidu, pokud je zapnuté, je možné pouze pokud je to dovoleno v ***Nastavení → Nastavení Programu → Ostatní*** (platí pouze pro FORTE).

Pokud je na pinu VDD přítomno napájení, můžete tlačítkem ***Reset*** běžící aplikaci zastavit nebo dalším stiskem tlačítka umožnit její spuštění.

Pojistky a práce s nimi

Vlastnosti programované součástky (pojistky) mohou být nastaveny v okně ***Konfigurace***. Změny pojistek mohou být uloženy vybráním ***Soubor → Uložit*** nebo ***Soubor → Uložit projekt***.

Pokud soubor .hex dané součástky definuje také její pojistky, ukládají se pojistky do tohoto souboru.

U mikrokontrolérů Atmel a u pamětí se pojistky ukládají do projektu. V tomto případě se musí po nastavení pojistek pravým tlačítkem kliknout na okno ***Konfigurace*** a použít volbu ***Zapamatovat pojistky***.

Pokud je použita volba ***Nastavení → Nastavení Programu → Programování → Načíst data ze souboru vždy znovu před programováním***, software znovu načte soubor po stisku tlačítka ***Programovat***. Pokud pojistky v souboru uloženy nejsou a zároveň není vytvořen projekt, ze kterého by se pojistky mohly připojit, dojde v tomto případě k inicializaci konfigurační paměti do výchozího stavu před každým programováním. Tomu je možné předejít zrušením volby ***Nastavení programu → Soubory → Inicializovat konfigurační paměť před čtením ze souboru***.

Mnoho součástek má specifické požadavky na nastavení pojistek. Více informací, jak pojistky správně nastavit, naleznete v katalogovém listu programované součástky.

5.3.5 Programování

Výběr souboru který chcete programovat provedete volbou ***Soubor → Otevřít***.

Po načtení souboru vidíte aktuální paměť programu, paměť dat a konfigurační paměť (pojistky). Pokud tato okna nejsou vidět, je možné je zapnout v menu ***Zobrazit***.

Kdykoliv je možné programovanou paměť ručně modifikovat prostým označením požadovaného místa a přepsáním hodnoty pomocí klávesnice. Pokud si přejete

modifikaci uložit, použijte ***Soubor → Uložit***, ***Soubor → Uložit jako*** nebo ***Soubor → Export datové paměti***.

Důležité upozornění

Před vlastním programováním je vhodné zkontrolovat nastavení programátoru a nastavení pojistek, protože chyba v tomto případě může znamenat zničení programované součástky nebo dokonce programátoru.

Programování začne po kliknutí na ***Součástka → Programovat*** nebo po kliknutí na tlačítko ***Programovat***.

Před programováním je u některých součástek zkontrolováno Device ID a Code/Data protection bity (elektronický podpis součástky). Pokud nesouhlasí ID s vybraným typem součástky, je vypsáno chybové hlášení.

Velmi často se stane, že se toto chybové hlášení objeví, pokud je nějaká chyba v propojení programované součástky a programátoru.

Pokud je vše v pořádku, udělá programátor následující operace: Smaže součástku, zkontroluje smazání, naprogramuje a zkontroluje naprogramování celé součástky.

Pokud je potřeba programovat pouze některou z pamětí mikrokontroléru, může to být provedeno vybráním příslušné položky v menu ***Součástka → Programovat*** nebo kliknutím na rozbalovací šipku u tlačítka Programovat na liště tlačítek. K dispozici je dle typu použité součástky programování paměti programu, dat a konfigurační paměti nebo programování celé součástky.

Rozdílové programování

Pokud to programovaná součástka podporuje, je v menu ***Součástka → Programovat*** možnost použít rozdílové programování, kdy se nejprve přečte stávající obsah paměti a potom se programují jen buňky, které se liší.

Rozdílové programování je vhodné během vývoje, kdy se často a zároveň nepatrně mění obsah programovaných dat. Protože se zapisují jen ty buňky které jsou změněné je rozdílové programování výhodné u součástek s malým počtem cyklů zápisu. Také může být rychlejší než klasický zápis celého obsahu paměti.

5.4 Další možnosti programu UP

V následujícím textu probereme některé další funkce programu UP, které jsou při programování součástek k dispozici.

5.4.1 Nastavení funkce tlačítka GO

Programátory firmy ASIX obsahují tlačítko GO, které umožňuje uživateli spouštět programování bez potřeby myši nebo klávesnice.

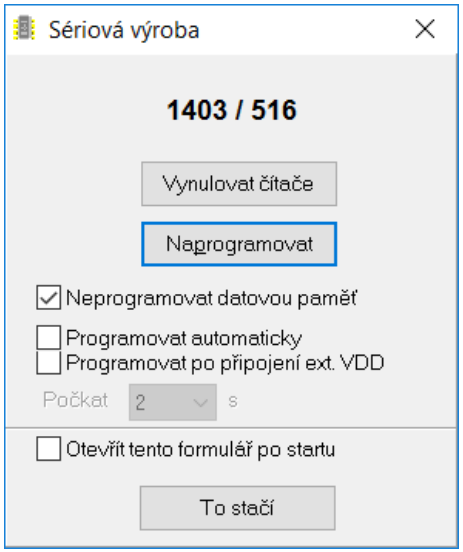
Funkce tlačítka GO může být nastavena podle potřeb uživatele v menu ***Nastavení → Klávesové zkratky*** pod položkou ***Tlačítko GO***.

Pokud chce uživatel používat tlačítko GO, program UP musí být vždy spuštěn, ale může být minimalizovaný.

Některá další nastavení související s tlačítkem GO naleznete v kapitole Nastavení pro programování během vývoje.

5.4.2 Sériová výroba

Funkce sériové výroby je dostupná v menu ***Součástka → Programovat → Sériová výroba***. Tato funkce je dostupná také na panelu funkcí pod tlačítkem Programovat.



Obr. 44: Sériová výroba

Z dialogu ***Sériová výroba*** může být programování spuštěno kliknutím na tlačítko ***Naprogramovat***.

Funkce tohoto tlačítka je ekvivalentní použití “Naprogramovat vše“ nebo “Naprogramovat vše kromě datové paměti“ v závislosti na stavu volby “Neprogramovat datovou paměť“.

V tomto dialogu je zobrazen čítač naprogramovaných součástek. Podle nastavených vlastností programu může být čítač zobrazen také na stavovém panelu. Více o nastavení je v kapitole Nastavení pro produkční programování.

Čítač zobrazuje počet naprogramovaných součástek jak v módu sériové výroby tak ve standardním módu programu.

Tlačítkem Vynulovat čítače vynulujete všechny čítače sériové výroby. Tuto operaci není možné vzít zpět.

Volba "Programovat automaticky" způsobí, že je součástka naprogramována po jejím připojení, kontrola připojení se provádí čtením Device ID součástky.

Programování může být zahájeno v okamžiku připojení externího napájecího napětí (VDD).

Zaškrtnutím položky "Otevřít tento formulář po startu" se formulář sériové výroby otevře po startu programu nebo po otevření souboru projektu, je-li použit.

5.4.3 Sériová čísla

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

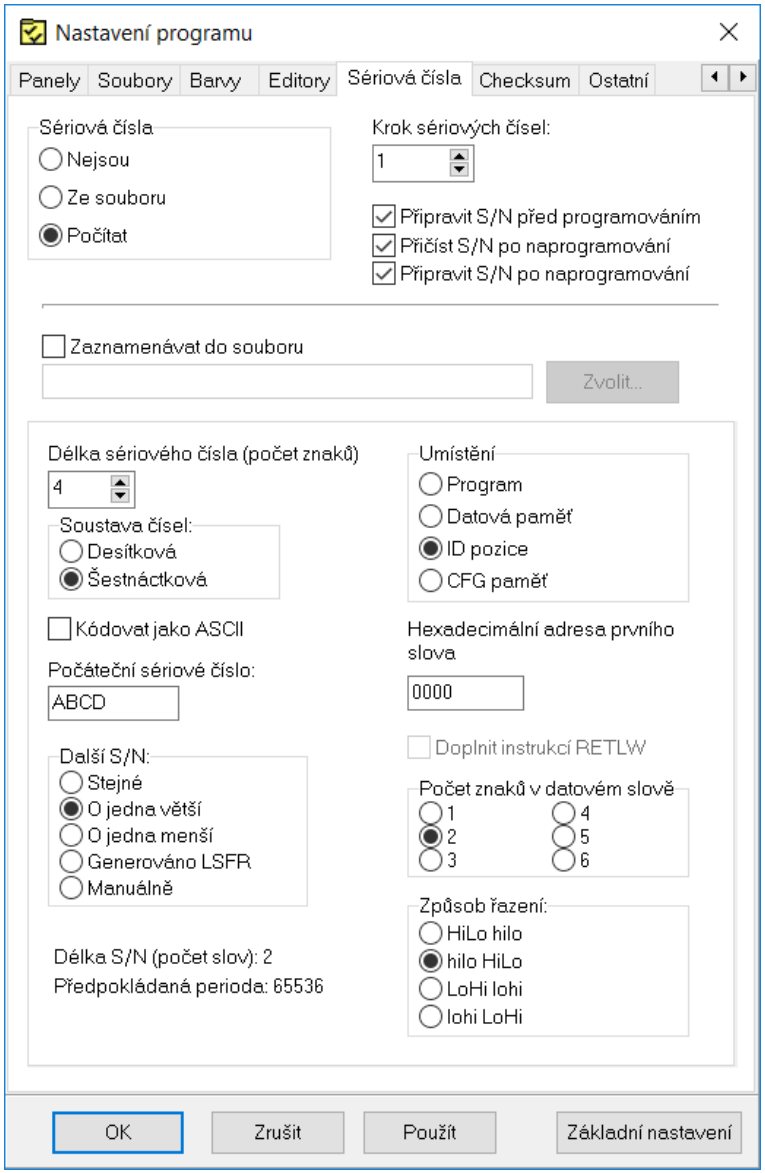
Obr. 45: Sériová čísla

Po zapnutí sériových čísel a nastavení jejich typu v ***Nastavení → Nastavení programu → Sériová čísla*** se zobrazí okno s informacemi o aktuálním sériovém čísle a možností ručního zapsání sériového čísla do HEX editoru paměti, kam se bude sériové číslo ukládat, či přechodu na další číslo v pořadí.

Sériová čísla mohou být:

- **počítaná**
  Počítaná sériová čísla mohou být vkládána vždy pouze na jedno zvolené místo v součástce, jako např. do paměti programu, datové paměti 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 znacích do jednoho slova) nebo ASCII znak (jeden nebo dva ASCII znaky do slova). Při používání paměti programu u mikrokontrolérů Microchip lze pro uložení sériového čísla zvolit instrukce RETLW, kdy se na adresu v paměti programu vloží instrukce RETLW s parametrem odpovídajícím sériovému číslu.

- **vkládaná ze souboru**
  Jedno sériové číslo může být rozmístěné do více částí součástky. (např. vlastní sériové číslo přímo v programu, adresa zařízení v datové paměti a znovu sériové číslo uložené v ID pozicích pro možnost přečtení sériového čísla ze zamknuté součástky)



Obr. 46: Nastavení sériových čísel

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 návěstí sériových čísel - viz Formát souboru se sériovými čísly.

Kromě sériových čísel se do souboru zapisuje datum, čas a výsledek programování. U součástek, kde je tato funkce podporována, se zapisuje i revize programované součástky.

Formát souboru se sériovými čísly

Soubor s definicí sériových čísel je textový a velmi snadno vytvořitelný jiným programem. Doporučená přípona souboru je *.SN nebo *.TXT.

Záznam sériového čísla má tvar

```
[komentář] návěstí: datový záznam, datový záznam, ..., datový záznam;
```

Středník na konci záznamu je povinný.

- **Komentář** je jakýkoli řetězec, který neobsahuje dvojtečku ':'. Komentář je nepovinný. V případě, že celý záznam sériového čísla neobsahuje dvojtečku, je ignorován (je brán pouze jako komentář). Řádek lze také zakomentovat dvěma lomítky '//'.

```
toto je pouhy komentar;
// i toto je komentar
```

- **Bílé znaky** jsou mezera, tabelátor, konec řádku (CR +LF).
- **Návěstí** je řetězec identifikující sériové číslo. Tento řetězec je povinný. Návěstí nesmí obsahovat bílé znaky, dvojtečku, středník.

Datový záznam

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.

Například b'10101010' znamená totéž co h'AA', d'170' nebo jen samotné AA. 'A' znamená totéž co d'65' (samotný ASCII znak).

Příklad záznamu pro PIC16F628A:

2100 05 55 54 znamená uložit do datové paměti na adresy 00 až 02 data 05h, 55h, 54h.

Programová paměť, kam se uloží sériové číslo, může být také specifikována slovem „CODE.“ nebo „PROG.“ nebo jen „P.“.

Datovou paměť specifikujeme slovem „DATA.“ nebo „EE.“ nebo zkráceně jen „E.“.

Pro paměť ID pozic se používá „ID.“ nebo jen „I.“

Tato slova jsou vždy následována adresou ve specifikované paměti.

Příklad:

EE.00 05 55 54 znamená do datové 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 datové paměti (EEPROM) se zadává adresa 16-bitového slova, tzn. tak jak jdou v mikrokontroléru jedna po druhé.

- Samostatné paměti (I^2C, SPI) mají jen paměť CODE, v případě specifikace neexistující paměti bude hlášena chyba.

Příklad souboru se sériovými čísly

```
komentar na zacatek;
sn1:    0000 34 45 56 67,
        2100 01 02 03 04; 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';
```

5.4.4 Podpora kalibrační paměti

Některé součástky obsahují kalibrační paměť s továrně přednastavenou kalibrací součástky. Ztráta jejího obsahu může způsobit vadu funkce součástky, a proto pro tyto případy máme nástroje, jak s kalibrační pamětí pracovat.

Práce s kalibrační pamětí při mazání součástky v UV mazačce

Před mazáním součástky je potřeba zaznamenat si kalibrační informaci. K tomuto účelu je možné použít funkce ***Soubor → Uložit kalibrační informaci***...

Opětovné načtení zajistíme pomocí ***Soubor → Načíst kalibrační informaci*** ...

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

Práce s kalibrační pamětí u součástek s pamětí Flash

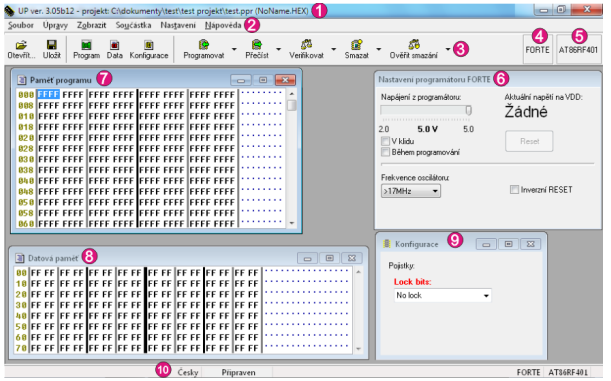
Při smazání těchto součástek se obsah kalibrační paměti automaticky zachovává.

Pokud z nějakého důvodu chcete kalibrační paměť skutečně smazat, lze toto provést funkcí ***Součástka → Smazání → Smazat vše (i kalibrační paměť)***.

Upozornění

Nové Flash součástky s kalibrační pamětí (např. PIC12F629) obsahují i tzv. bity bandgap, které jsou též součástí kalibrace součástky. Tyto bity se vyskytují v konfiguračním slově a při použití funkce ***Součástka → Smazání → Smazat vše (i kalibrační paměť)*** se také smažou!

5.5 Ovládací prvky programu UP



Obr. 47: Ovládací prvky programu UP

1) Záhlaví s jménem aktuálního projektu nebo souboru
2) Menu
3) Panel nástrojů
4) Aktuálně vybraný programátor
5) Aktuálně vybraná součástka
6) Okno nastavení programátoru
7) Okno HEX editoru programové paměti
8) Okno HEX editoru datové paměti
9) Okno konfigurace
10) Stavový panel

5.5.1 Panel nástrojů

Panel nástrojů je lišta s tlačítky rychlé volby pod výběrem menu programu (viz kapitola Ovládací prvky programu UP).

Pokud chcete lištu odstranit, jednoduše zrušte možnosti zobrazit nápisy a ikony na panelu nástrojů pomocí nastavení popsaném v kapitole Nastavení programu.

5.5.2 Stavový panel

Stavový panel je panel ve spodní části okna (viz kapitola Ovládací prvky programu UP). Jsou v něm zobrazeny informace o programátoru, součástce, změně souboru od posledního uložení, apod.

Jednotlivé položky stavového panelu reagují na dvojité kliknutí a nabídku na pravém tlačítku myši.

Pokud chcete stavový panel odstranit nebo znovu zobrazit, použijte volbu ***Nastavení programu → Panely → Ve spodní části okna zobrazovat stavový panel***.

5.5.3 Menu programu UP

V následujícím textu detailně popíšeme jednotlivé položky menu programu UP.

Akce menu mohou být vyvolány kliknutím myší na příslušnou položku menu nebo pomocí klávesnice stisknutím klávesy společně s klávesovou zkratkou zvýrazněnou v menu.

Menu je rozděleno do následujících kategorií:

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

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 formáty souborů jsou popsány v kapitole Příloha C Formát Intel-HEX souboru. Soubory s příponou .hex a .a43 jsou načítány jako Intel-HEX soubory, ostatní jako binární soubory. K dispozici jsou také filtry umožňující otevřít všechny soubory jako .hex nebo jako .bin.

Soubor → Otevřít další soubor

Importuje další soubor .hex nebo .bin 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 Intel-HEX, ostatní jako binární.

Takto je možné složit více .hex souborů dohromady (např. bootloader + program).

Pokud v dialogu **Otevřít další soubor** uživatel zaškrtne volbu **Otevřít příště automaticky**, soubor se bude automaticky načítat po hlavním datovém souboru. Jméno souboru je vypisováno v menu **Soubor → Otevřít další:**. Zde lze také opětovné načítání zrušit.

Tímto způsobem je možné automaticky načítat jeden soubor.

Soubor → Otevřít další:

Pokud je volba aktivní, indikuje, že je nastaveno automatické načítání dalšího souboru, kliknutím na tuto volbu lze jeho automatické načítání zrušit.

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 programu → Soubory → Kontrolovat změny v souboru***, 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, viz kapitola 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 ze souboru...

Pomocí standardního dialogového okna Windows umožní přečíst obsah datové paměti (EEPROM) z jiného souboru. Tím je umožněno načtení datové paměti, pokud nebyla uložena ve stejném souboru jako paměť programu (to se týká např. mikrokontrolérů ATmega8).

Důležité upozornění

Tento soubor, bez ohledu na jeho obsah, je čtený od adresy nula, jako kdyby obsahoval pouze datovou paměť (EEPROM). Soubor normálně vygenerovaný překladačem tedy takto nelze korektně načíst.

Soubor → Otevřít soubor s datovou pamětí automaticky

Pokud je zvolena tato volba, program UP současně s načítáním souboru pro paměť programu automaticky načte 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 byl s projektem otevřen některý další soubor, je tento načten také.

Soubor → Uložit projekt

Klávesová zkratka: Shift+Ctrl+S

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

V dialogu **Uložit projekt** lze volbou **Načítat projekt odemčený** nastavit, že po načtení bude projekt odemčený.

Volbou **Přidat poznámku**, lze do projektu uložit poznámku, která se zobrazí po jeho otevření.

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 otevřením projektu.

Soubor → Poslední projekty

Pod touto položkou je zapamatováno posledních 10 otevřených projektů, kliknutím na jméno projektu se projekt načte.

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 Podpora kalibrační paměti.

Soubor → Export do bin...

Pomocí této funkce lze do vybraného souboru zapsat binární data z paměti programu, datové paměti (EEPROM), konfigurační paměti nebo ID pozic.

Pro zapisovaná data lze zvolit šířku slova po 16 nebo 8 bitech.

Soubor → Ukončení programu

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

Tímto příkazem se program ukončí. Pokud byl otevřený soubor změněn, program se při ukončení dotáže, zda má změny uložit.

Důležité upozornění

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 tak 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 hodnotou nebo náhodnými daty.

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. Více informací o označení oblasti je v kapitole Okna HEX editorů.

Úpravy → Vložení textu...

Umožňuje vložit na zvolené místo paměti text v ASCII nebo hexadecimálním formátu. Konce řádků lze kódovat jako znaky NULL, CR, LF nebo CR+LF.

Lze vyplňovat jednotlivé byty nebo ukládat do instrukcí RETLW (týká se jen paměti programu u mikrokontrolérů Microchip). Pozn.: Instrukce RETLW u mikrokontrolérů Microchip je instrukce návratu s konstantou v pracovním registru – často používaná pro vytváření tabulek.

Při zvolení funkce ***Vložení textu...*** program přednastaví vybranou paměť a počáteční buňku podle aktuálního okna a vybrané buňky.

Úpravy → Vybranou oblast doplnit instrukcí RETLW

Funkce u mikrokontrolérů Microchip doplní vybranou oblast paměti na instrukci RETLW.

Funkci lze použít pouze z otevřeného HEX editoru, 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 informací o označení oblasti je v kapitole Okna HEX editorů.

Menu Zobrazit

Zobrazit → Paměť programu

Zobrazí nebo skryje okno HEX editoru paměti programu. Více o HEX editorech viz Okna HEX editorů.

Zobrazit → Datová paměť

Zobrazí nebo skryje okno HEX editoru datové paměti. Více o HEX editorech viz Okna HEX editorů.

Zobrazit → Boot paměť

Zobrazí nebo skryje okno HEX editoru datové paměti. Více o HEX editorech viz Okna HEX editorů.

Zobrazit → Konfigurační paměť

Zobrazí nebo skryje okno konfigurační paměti. Více o HEX editorech viz Okna HEX editorů.

Zobrazit → Konzole

Zobrazí nebo skryje konzoli, kam UP může vypsat podrobnosti o programování.

Zobrazit → Zobrazení paměti programu

Klávesová zkratka: Alt+F10

Zobrazí HEX editor paměti programu. Pokud je HEX editor již zobrazen, je přesunut na popředí. Více o HEX editorech viz Okna HEX editorů.

Zobrazit → Zobrazení datové paměti

Klávesová zkratka: Alt+F11

Zobrazí HEX editor datové paměti. Pokud je HEX editor již zobrazen, je přesunut na popředí. Více o HEX editorech viz Okna HEX editorů.

Zobrazit → Zobrazení konfigurační paměti

Klávesová zkratka: Alt+F12

Zobrazí editor konfigurační paměti. Pokud je editor již zobrazen, je přesunut na popředí. Více o HEX editorech viz Okna HEX editorů.

Zobrazit → Zobrazení formuláře programátoru

Klávesová zkratka: Ctrl+P

Formulář programátoru je vždy zobrazen. Funkce zobrazení formuláře programátoru přesune formulář programátoru do popředí.

Menu Součástka

Součástka → Programovat

Klávesová zkratka: Shift+F5

Otevře další menu s volbami pro programování součástky. Některé položky mohou být pro určité typy součástek nedostupné.

▸ Naprogramovat vše

Klávesová zkratka: F5

Smaže, zkontroluje smazání, naprogramuje a zkontroluje naprogramování celé součástky. Před operací je provedena kontrola Device ID a Code/Data Protection. Chování funkce je ovlivněno nastavením pro programování, viz kapitola Nastavení programu.

▸ Naprogramovat vše kromě datové paměti

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***.

Provede totéž, jako ***Naprogramovat vše*** s výjimkou mazání, programování a kontroly datové paměti.

U součástek bez datové 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, potom program nabízí možnost ***Smazat celou součástku a naprogramovat i datovou 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 naprogramování programové paměti.

▸ Naprogramovat datovou paměť

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Smaže, zkontroluje smazání, naprogramuje a zkontroluje naprogramování datové paměti.

▸ Naprogramovat konfigurační paměť

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Naprogramuje a zkontroluje naprogramování konfigurační paměti a ID pozic, pokud je součástka obsahuje.

▸ 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.

Tuto funkci musí programovaná součástka podporovat, proto není dostupná pro všechny typy součástek.

Pokud má součástka aktivní Code/Data Protection, rozdílové programování nemá smysl, a místo něj program provede kompletní programování se smazáním součástky.

▸ Naprogramovat rozdílově datovou paměť

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 typy součástek.

Pokud má součástka aktivní Code/Data Protection, rozdílové programování nemá smysl, a místo něj program provede kompletní programování se smazáním součástky.

Rozdílové programování datové paměti je nutné použít u mikrokontrolérů 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.). Více informací je v kapitole Sériová výroba.

Součástka → Čtení

Klávesová zkratka: Shift+F6

Otevře další menu s volbami pro čtení součástky. Některé položky mohou být pro určité typy součástek nedostupné.

▸ Přečíst vše

Klávesová zkratka: F6

Přečte obsah celé součástky.

▸ Přečíst vše kromě datové paměti

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Přečte celou součástku kromě datové paměti.

▸ Přečíst paměť programu

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Přečte paměť programu.

▸ Přečíst datovou paměť

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Přečte datovou paměť.

▸ Přečíst konfigurační paměť

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Přečte konfigurační paměť a ID pozice, pokud je součástka obsahuje..

▸ Přečíst adresu

Funkce umožňuje přečíst data z uživatelem zvolené adresy, podporuje MCU ARM přes rozhraní SWD.

Součástka → Ověření

Klávesová zkratka: Shift+F7

Otevře další menu s volbami pro kontrolu obsahu paměti součástky. Některé položky mohou být pro určité typy součástek nedostupné.

▸ Zkontrolovat vše

Klávesová zkratka: F7

Porovná obsah pamětí součástky s aktuálním obsahem HEX editorů.

▸ Zkontrolovat vše kromě datové paměti

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Porovná obsah pamětí součástky kromě datové paměti s obsahem HEX editorů.

▸ Zkontrolovat paměť programu

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Porovná obsah paměti programu s obsahem HEX editoru programové paměti.

▸ Zkontrolovat datovou paměť

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Porovná obsah datové paměti s obsahem HEX editoru datové paměti.

▸ Zkontrolovat konfigurační paměť

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Porovná obsah konfigurační paměti a ID paměti, pokud jí součástka obsahuje, s daty v okně „Konfigurace“.

Součástka → Smazání

Klávesová zkratka: Shift+F8

Otevře další menu s volbami pro mazání paměti součástky.

Po funkci mazání se automaticky provádí kontrola smazání. V menu ***Nastavení → Nastavení programu → Programování → Neprovádět blank check po smazání*** lze vynutit vynechání této kontroly, což může u některých součástek ušetřit čas.

▸ 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 datovou paměť

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Smaže datovou paměť a zkontroluje ji. Pokud je aktivní Code/Data Protection, nelze tuto funkci použít.

Součástka → Kontrola smazání

Klávesová zkratka: Shift+F9

Otevře další menu s volbami pro kontrolu smazání součástky. Některé položky mohou být pro určité typy součástek nedostupné.

▸ Kontrola smazání všeho

Klávesová zkratka: F9

Ověří, zda je součástka správně smazaná.

▸ Kontrola smazání všeho kromě datové paměti

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Ověří, zda je součástka kromě datové 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 datové paměti

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Ověří, zda je datová paměť správně smazaná.

▸ Kontrola smazání konfigurační paměti

Klávesová zkratka může být přiřazena v ***Nastavení → Klávesové zkratky***

Ověří, zda jsou konfigurační paměť a ID pozice, pokud je součástka obsahuje, správně smazané.

Součástka → Výběr součástky...

Klávesová zkratka: F4

Dialogové okno ***Výběr součástky*** slouží k výběru typu programované součástky. U některých typů pamětí je po vybrání jejich typu nutné zvolit ještě organizaci dat.

V dialogovém okně pro výběr součástky jsou zobrazeny pouze ty součástky, které podporuje aktuálně vybraný programátor. Pokud chcete vybrat součástku, kterou daný programátor nepodporuje, je třeba nejprve zvolit jiný typ programátoru.

Více informací o výběru součástky je v kapitole Výběr typu součástky.

Součástka → Informace o součástce

Zobrazí okno s informacemi o připojení vybrané součástky k programátoru.

Menu Nastavení

Klávesová zkratka: Shift+F10

V menu ***Nastavení*** jsou veškerá nastavení programu UP. Možností nastavení je velké množství. Pokud si nejste jisti, že všechna nastavení jsou v pořádku, je vhodné použít volbu ***Základní nastavení***, která vrátí všechna nastavení do počátečního stavu.

Důležité upozornění

Tlačítko ***Nastavení programu → Základní nastavení*** obnoví stav všech nastavení na všech kartách, tedy např. i nastavení barev.

Nastavení → Nastavení programu → Programování

Klávesová zkratka: Shift+F10

V tomto okně lze nastavit veškerá obecná nastavení programování.

Nastavení týkající se výběru programátoru a komunikačního portu je popsáno v kapitole Volba programátoru.

Pro nastavení typu programované součástky je zvláštní okno Výběr součástky

▸ Načíst soubor vždy znovu před programováním

Při zapnutém nastavení ***Načíst soubor vždy znovu před programováním*** program vždy před jakýmkoli požadavkem pro programování součástky přečte aktuální soubor s programovanými daty z disku.

Pokud je zároveň nastaveno používání sériových čísel a

zápis čísel vždy před programováním, jako první se soubor přečte, a teprve pak se připíše aktuální sériové číslo.

▸ Zachovat manuálně změněná data

Pokud je tato volba aktivní, načtením souboru před programováním se nepřepíší manuálně změněná data v programu UP. Tato volba ovlivňuje jen chování funkce načítání souboru před programováním.

▸ Před načtením souboru varovat, pokud byla data v editoru změněna

Pokud je před načtením souboru před programováním zjištěno, že data v některém editoru byla změněna, program zobrazí varování.

▸ Varovat, pokud se načtený soubor nezměnil

Program zobrazí varování, pokud se obsah načteného souboru od předchozího programování nezměnil.

▸ Programovat pouze pozice v souboru

Programování změní jen pozice obsažené v souboru. Nejprve se vyčte obsah součástky, poté je načten obsah souboru a nahradí se jím data načtená ze součástky. Na adresách, které v souboru nejsou obsaženy, zůstanou data vyčtená ze součástky. Nakonec je paměť přeprogramována.

▸ Zeptat se před smazáním

Pokud je aktivní toto nastavení, vyžádá si program UP před smazáním součástky potvrzení.

▸ 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. Součástka má aktivní Code nebo Data Protection. Bude potřeba ji smazat celou. Pokračovat?), nebude po zodpovězení tohoto dialogu již dále vyžadovat potvrzení programování.

▸ Zobrazovat varovná hlášení k pojistkám

Uživatel může zvolit, zda se budou zobrazovat varovné hlášení přiřazené k některým pojistkám. Doporučuje se tuto volbu ponechat zapnutou.

▸ Mimo programování: Automaticky zavřít stavové okno

Toto nastavení způsobí, že stavové okno bude zavřeno, pokud nenastane chyba během mazání, kontroly mazání, kontroly naprogramování nebo čtení.

▸ Po programování: Automaticky zavřít stavové okno

Toto nastavení způsobí, že stavové okno bude zavřeno, pokud nenastane chyba během programování nebo následné verifikace.

▸ Zvuková signalizace úspěšného dokončení

Při zapnutém nastavení program vyvolá standardní "systémový výkřik", pokud operace (např. mazání, programování atd.) proběhla v pořádku.

▸ Zvuková signalizace neúspěšného dokončení

Při zapnutém nastavení program vyvolá standardní "systémový výkřik", pokud se během operace (např. mazání, programování atd.) objeví chyba nebo varování.

▸ Vypnout všechny zvuky programu UP

Pokud je zaškrtnutá tato volba, program UP nebude vydávat žádné zvuky.

▸ Čas pro zapnutí/vypnutí VDD při napájení z programátoru

Pro programování pomocí ICSP kabelu přímo v osazené DPS je důležité nastavení ***Čas pro zapnutí/vypnutí napětí při napájení z programátoru***, které určuje délku časové prodlevy při připojování a odpojování napětí od součástky.

Programátory PRESTO a FORTE poskytují nadproudovou ochranu. Po časové prodlevě dané nastavením doby ***Zapnutí*** se po připojení napájení součástky provede test na nadměrný proud velikosti cca 100 mA na napájecím napětí.

Nastavením doby ***Vybití*** se určuje po jaké době od vypnutí napájecího napětí proběhne kontrola, zda už na pinu není přítomno žádné napětí.

Pokud je na napájecích pinech součástky připojen blokovací kondenzátor (doporučeno), napětí na pinu se mění pomaleji. To může způsobovat problémy během programování, jejichž řešením je právě prodloužení nabíjecího a vybíjecího času.

Delší než potřebný čas zvyšuje pravděpodobnost zničení součástky při nesprávném zapojení, při kratším čase mohou ještě obvody programátoru detekovat nadměrný proud tekoucí do kondenzátorů aplikace. Vzorec k přibližnému určení potřebného času je možné nalézt v kapitole Příloha B - Použití ICSP.

▸ Neprovádět kontrolu Device ID před programováním

Tímto nastavením lze vypnout kontrolu Device ID před programováním.

▸ Neprovádět blank check při programování pouze konf. slova

Konfigurační slovo umí většina přepisovatelných součástek přepsat, aniž by musela být celá součástka smazána. Přeskočením kontroly smazání (blank check) konfiguračního slova se této vlastnosti využívá, takže program bude nesmazané slovo ignorovat.

Toto nastavení se netýká programování celé součástky, kdy se součástka maže kompletně celá, ale pouze přepisování konfiguračního slova.

▸ Neprovádět blank check po smazání

Přeskakování blank check po smazání je nastavení, které lehce urychlí programování, a proto 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.

▸ Nemazat součástku před programováním

Součástka nebude před programováním smazána.

Nemazat datovou paměť před jejím programováním

Tato položka má vliv pouze na programování součástek Atmel AVR, např. ATmega8, při programování pouze datové paměti. Datová paměť této rodiny součástek nevyžaduje mazání před programováním. Pokud tato volba nebude zaškrtnutá bude před programováním pouze datové paměti součástka smazána včetně všech ostatních pamětí.

▸ Neprovádět kontrolu naprogramování prázdných pozic na konci paměti

Pokud je na konci programované paměti oblast, která obsahuje jen výchozí hodnoty, nebude se verifikovat.

Tato funkce umožňuje zrychlit verifikaci naprogramované paměti, protože 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, že obsah je správně naprogramován.

▸ Kontrolovat při dvou napájecích napětích

Tato funkce je dostupná pouze pro programátor FORTE. Je použitelná pouze s interním napájením z programátoru. Umožňuje provádět verifikaci při dvou napájecích napětích definovaných uživatelem.

Někteří výrobci součástek pro produkční programování doporučují verifikovat obsah při dvou napájecích napětích odpovídajících povolenému rozsahu napájecích napětí součástky.

Nastavení → Nastavení programu → Panely

Klávesová zkratka: Shift+F10

V této části menu může být nastaven vzhled aplikace. Uživatel může nastavit, kde a jak budou některé ovládací komponenty zobrazeny.

▸ Vybranou součástku zobrazit na panelu nástrojů

Vybraná součástka bude kromě stavového panelu zobrazena také na panelu nástrojů.

▸ Vybraný programátor zobrazit na panelu nástrojů

Vybraný programátor bude kromě stavového panelu zobrazen také na panelu nástrojů.

▸ Ve spodní části okna zobrazit stavový panel

Toto nastavení určuje, zda bude zobrazen stavový panel či nikoliv.

▸ Zobrazit ikony na tlačítkách panelu nástrojů

Na panelu nástrojů budou zobrazeny ikony jednotlivých nástrojů.

▸ Zobrazit nápisy na tlačítkách panelu nástrojů

Na panelu nástrojů budou u jednotlivých nástrojů zobrazeny názvy nástrojů.

Pokud se použije volba ***Nápisy na tlačítkách panelu nástrojů vpravo od ikon***, bude celková výška panelu snížena na polovinu.

▸ Zobrazit čítač sériové výroby ve stavové liště

Počítadlo sériové výroby se zobrazuje na stavovém panelu použitím této volby. Počítadlo ukazuje počet programovaných součástek a počet úspěšně naprogramovaných součástek.

Pokud se použije volba ***Počítat všechny akce***, započítá se do celkového počtu každá akce provedená se součástkou (tedy např. vyčtení, smazání, programování datové paměti atd.).

Volba ***Zobrazit i popis hodnot*** zapne pro větší přehlednost vysvětlivky k jednotlivým položkám počítadla.

Pokud nechcete, aby se do správných operací započítávaly ty, které skončily varováním, použijte volbu ***Akci skončenou varováním brát jako nepovedenou***.

Volba ***Při otevření projektu vynulovat čítače*** způsobí, že čítače jsou vynulovány při jakémkoliv načtení projektu včetně automatického načtení projektu při spuštění programu.

Styl počítání umožňuje vybrat, jestli se bude počítadlo zobrazovat ve formátu dobré / špatné nebo dobré / celkem.

K nulování čítačů slouží tlačítko ***Reset počítadla***, v poli ***Přednastavení čítačů*** je možné nastavit jejich počáteční hodnoty.

Nastavení → Nastavení programu → Soubory

Klávesová zkratka: Shift+F10

V nastavení souborů jsou veškerá nastavení pro čtení a ukládání dat do souborů.

▸ Způsob ukládání souborů

Panel ***Způsob ukládání souborů*** slouží k možnosti neukládat vždy všechny oblasti všech editorů do souboru, ale pouze některé. Program UP bude podle nastavení způsobu ukládání zobrazovat dotazy při ukládání jednotlivých editorů projektu.

▸ Kontrolovat změny v souboru

Tato volba slouží, zejména při ladění programu, k opětovnému přečtení souboru po detekci změny data posledního uložení souboru.

▸ Neptat se a neukládat změny do datového souboru

Pokud je tato volba aktivní a data v editoru byla změněna, před otevřením jiného souboru, před zavřením programu atp. se program nebude ptát na uložení a ani ukládat aktuálně otevřený soubor.

▸ Kontrolovat typ součástky při čtení souboru .hex

Pokud byl do souboru .hex uložen i typ součástky, a ten se neshoduje s aktuálně vybraným typem součástky, program na tuto neshodu upozorní.

▸ Ukládat typ součástky do souboru .hex

Za konec Intel-HEX souboru se 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 informací o formátu Intel-HEX naleznete v kapitole Příloha C Formát Intel-HEX souboru.

▸ Varovat, pokud načtený soubor neobsahuje data pro CFG paměť

Zobrazí varování v případě, že načtený soubor neobsahuje data pro konfigurační paměť a u vybraného typu součástky jsou konfigurační data v souboru očekávána.

▸ Varovat, pokud načtený HEX soubor není zarovnaný na velikost slova.

Zobrazí varování v případě, že načtený HEX soubor není zarovnaný na velikost slova paměti součástky.

▸ Způsob načítání a ukládání binárního souboru

V tomto panelu může být nastaveno, jak budou načítány a ukládány soubory .bin, pokud je zvolena součástka s více byty na slovo.

Jsou možnosti, že se program bude vždy před načtením nebo uložením souboru .bin ptát (***Ptát se, zda načíst / uložit .bin jako Big nebo Little Endian***) nebo že program vždy bez ptaní načte soubor jako Little Endian (***Nikdy se neptat, načíst/uložit jako Little Endian***) nebo jako Big Endian (***Nikdy se neptat, načíst/uložit 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 obsahuje samé jedničky (tedy FFFh, 3FFFh atd...), což může ale být i smysluplná instrukce (např. 3FFFh je instrukce addlw -1 mikrokontrolérů PIC).

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

Pokud uživatel používá inicializaci paměti před načtením souboru .hex, což je doporučeno, bude správně naprogramován i program načtený ze zkráceného souboru .hex, protože chybějící instrukce se „automaticky“ vytvoří.

▸ Inicializovat paměť programu / datovou paměť / ID pozice před čtením ze souboru

Před čtením souboru se tato oblast vyplní jedničkami a pak se teprve soubor načte. Takto se smažou všechny pozice, které v souboru .hex nejsou uloženy.

Tato volba je důležitá také v případě, že nejsou v souboru .hex uloženy prázdné pozice (viz ▸ Do souboru .hex ukládat prázdné pozice).

▸ Inicializovat konfigurační paměť před čtením ze souboru

Použitím této volby dojde k inicializaci konfigurační paměti před načtením souboru pro paměť programu.

Pokud v souboru nejsou pojistky uložené, je užitečné tuto volbu zrušit. Uživatel na začátku práce nastaví obsah konfigurační paměti a následně i při opětovném načtení souboru nebude nutné pojistky znovu nastavovat.

▸ Přečíst datovou paměť ze součástky místo čtení ze souboru

Pokud chcete mít jistotu, že nebude přepsán obsah datové paměti, používejte funkci ▸ Naprogramovat vše kromě datové paměti. Kdyby ale omylem byla použita funkce ▸ Naprogramovat vše např. nechtěným stiskem tlačítka GO, k smazání obsahu datové paměti by došlo.

Pro tyto případy je zde funkce ***Přečíst datovou paměť ze součástky místo čtení ze souboru***. Program danou oblast vyplní obsahem paměti součástky připojené k programátoru.

Důležité upozornění

Tato funkce může způsobit nečekanou práci programátoru, například při zapnutí programu UP.

▸ Přečíst ID pozice ze součástky místo čtení ze souboru

Pokud chcete zachovat obsah ID pozic, použijte funkci ***Přečíst ID pozice ze součástky místo čtení ze souboru***. Program danou oblast před vlastním programováním vyplní obsahem uloženým v paměti součástky připojené k programátoru.

Důležité upozornění

Tato funkce může způsobit nečekanou práci programátoru, například při zapnutí programu UP.

▸ Ukládat pojistky v programu UP místo datového souboru

Tato funkce umožňuje ukládání pojistek do ini souboru programu nebo do projektu programu UP i pro součástky, jejichž pojistky se standardně ukládají do datového souboru.

▸ Způsob ukládání projektů

Zde mohou byt nastaveny vlastnosti ukládání projektů při ukončování programu UP. K dipozici je automatické uložení, dotaz na uložení a automatické zachování původního souboru.

▸ Načíst poslední projekt při startu

Touto volbou lze nastavit, zda se při příštím startu programu UP otevře projekt, který byl otevřený při zavírání programu.

Nastavení → Nastavení programu → Barvy

Klávesová zkratka: Shift+F10

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

K dispozici je možnost změnit barvu popředí, pozadí a použitý font vybraného textového prvku.

Nastavení → Nastavení programu → Editory

Klávesová zkratka: Shift+F10

▸ V editoru paměti kódu zobrazit slova po bytech

U součástek s délkou slova 16 bitů je možné zobrazovat jednotlivá slova po bytech.

▸ Editor paměti kódu široký 8 slov

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 typu součástky.

Stejnou funkci pro ostatní paměti mají nastavení:

▸ Editor datové paměti široký 8 slov

▸ Editor boot paměti široký 8 slov

▸ Zobrazovat ASCII překlad pouze nejnižšího bytu slova

Po vybrání této volby se zobrazuje ASCII překlad pouze nejnižšího bytu slova, což je výhodné především při použití mikrokontrolérů PIC.

▸ Maskovat ID pozice při čtení ze součástky, souboru a pod.

Podle specifikací některých výrobců je doporučeno do ID pozic většinou ukládat pouze maskovaná data, kde jsou využitelné pouze čtyři bity. Při povolení tohoto nastavení bude program požadovanou bitovou masku zavádět vždy, když odněkud čte ID pozice.

▸ Maskovat ID pozice při přímém zadání uživatelem

Při povolení tohoto nastavení bude program zavádět bitovou masku při každé modifikaci ID pozic uživatelem. Více informací naleznete v kapitole ▸ Maskovat ID pozice při čtení ze součástky, souboru a pod..

▸ V okně konfigurační paměti zobrazit místo pojistek přímo konf. slova

Toto nastavení je doporučeno pouze pro pokročilé uživatele. Z bezpečnostních důvodů se neukládá do konfiguračního souboru programu UP.

Přímou editací pojistek se rozumí přímé vepsání hodnoty konfiguračního slova v hexadecimálním tvaru.

Při zadání "nepřeložitelného" konfiguračního slova program nerozpoznané položky nechá nezměněné, pokud je uživatel sám nezmění. Většinou se to týká pojistek CP, které mají několik bitů, ale pouze dvě hodnoty.

Nastavení → Nastavení programu → Sériová čísla

Klávesová zkratka: Shift+F10

Více informací o použití sériových čísel naleznete v kapitole Sériová čísla.

▸ Sériová čísla

Na tomto panelu se vybírá, zda budou sériová čísla použita a pokud ano, zda se načítají ze souboru a nebo jsou počítaná a na příslušnou pozici vkládána automaticky.

▸ Připravit S/N před programováním

Toto nastavení způsobí, že se do vybrané pozice v paměti připraví sériové číslo před programováním součástky.

▸ Přičíst S/N po naprogramování

Po úspěšném naprogramování dojde k přechodu na další sériové číslo. Pokud není tato volba použita, je možné na další sériové číslo přejít kliknutím na tlačítko **Další* v okně Sériová čísla - viz kapitola Sériová čísla.

▸ Připravit S/N po naprogramování

Toto nastavení způsobí, že se do vybrané pozice v paměti připraví sériové číslo po naprogramováním součástky.

▸ Krok sériového čísla

Zde se definuje, o kolik větší nebo menší bude následující sériové číslo.

▸ Zaznamenávat do souboru

Zapnutím této volby a výběrem souboru se začnou do souboru ukládat informace o správně i špatně naprogramovaných součástkách a času programování.

▸ Po načtení projektu nastavit aktuální SN podle posledního v souboru

Když je tato funkce zapnutá, poslední programované SN je přečteno z logovacího souboru. Pokud bylo poslední programování zaznamenáno jako bezchybné, je podle této hodnoty vytvořeno další sériové, v opačném případě je znovu nastaveno poslední SN.

Logovací soubor musí existovat a jeho poslední záznam musí obsahovat sériové číslo.

Tato funkce může být užitečná, pokud jsou zapnutá sériová čísla a je zakázané ukládání projektů programu UP, v takovém případě se při zavření programu nebude ukládat aktuální SN do souboru projektu.

▸ Délka sériového čísla (počet znaků)

Zde se nastavuje, kolik znaků má sériové číslo. Např. 4 znaky umožní sériová čísla 0001 až 9999 v desítkové soustavě.

▸ Soustava čísel

Slouží k definici, v jaké soustavě je znak sériového čísla. K dispozici je desítková a šestnáctková soustava.

▸ Kódovat jako ASCII

Použitím této volby se dosáhne toho, že sériové číslo bude čitelné jako ASCII znaky.

▸ Počáteční sériové číslo

Zde se definuje počáteční hodnota od které se začnou sériová čísla počítat.

▸ Další S/N

V poli ***Další S/N*** se definuje následovník sériového čísla.

Pokud je zvoleno ***Stejné***, nebude se sériové číslo měnit. Při volbě ***O jedno větší*** bude následovat číslo o nastavení z odstavce ▸ Krok sériového čísla větší, při volbě ***O jedno menší*** se bude naopak o krok zmenšovat.

Volbou ***Generováno LSFR*** docílíme toho, že čísla budou odpovídat pseudonáhodné sekvenci, která jako startovní podmínku použije ***Počáteční sériové číslo*** a při stejném ***Kroku sériového čísla*** bude vždy stejná.

Volba ***Manuálně*** zobrazí v okně sériových čísel pole, kde bude možné před programováním zadat sériové číslo v šestnáctkové soustavě manuálně. Při této volbě lze také zadat sériové číslo na příkazovém řádku parametrem /sn.

▸ Umístění

V poli ***Umístění*** se určuje, do jaké oblasti paměti se bude sériové číslo ukládat. K dispozici jsou programová paměť, datová paměť a také ID pozice.

▸ Hexadecimální adresa prvního slova

Tato adresa definuje, kde ve vybraném typu paměti bude začínat první slovo sériového čísla.

▸ Doplnit instrukcí RETLW

U mikrokontrolérů Microchip je možné jednotlivá slova sériového čísla doplňovat do instrukce RETLW. Volba je možná pouze při umístění sériového čísla v programové paměti.

▸ Počet znaků v datovém slově

Zde se určuje, kolik znaků sériového čísla tvoří jedno slovo. K dispozici jsou 1 až 4 znaky do slova.

▸ Způsob řazení

Zde se nastavuje, jak budou jednotlivé znaky a slova sériového čísla řazena.

Pokud máme např. sériové číslo v desítkové soustavě se čtyřmi znaky a organizací 2 znaky do slova a máme sériové číslo 1234, bude toto číslo uloženo následovně:

```
HiLo hilo: 12 34
hilo HiLo: 34 12
LoHi lohi: 21 43
lohi LoHi: 43 21
```

Nastavení → Nastavení programu → Checksum

▸ Zobrazovat checksum ve stavové liště

Po zapnutí tohoto nastavení se bude na stavové liště zobrazovat checksum paměti programu.

Pokud je zvolen výpočet checksumu algoritmem MD5, checksum se zobrazí po najetí myši na stavovou lištu v místě textu MD5.

Pozn.: Dvojklik na zobrazené hodnotě checksumu na stavové liště vyvolá přepočítání hodnoty.

▸ Zapisovat checksum do logovacího souboru

Po zapnutí tohoto nastavení se bude do logovacího souboru zapisovat i checksum načítaného datového souboru.

Pozn.: Checksum se přepočítá při načtení souboru, aby se v logu checksum zobrazil, musí být tato funkce zapnuta dříve než je soubor načten.

▸ Algoritmus checksumu

Touto volbou lze vybrat algoritmus, jak se bude checksum počítat.

Nastavení → Nastavení programu → Ostatní

Klávesová zkratka: Shift+F10

▸ Nastavení kontroly nové verze programu UP

Zde se nastavuje, zda se bude program při každém spuštění dotazovat na povolení připojit se k Internetu a kontrolovat přítomnost novější verze na webu, či nikoliv.

Více informací naleznete v kapitole Aktualizace programu UP.

▸ Dovolit kolizi interního napájecího napětí s externím

Varování
Kolize napájecích napětí může způsobit zničení programátoru nebo programované aplikace!

Pokud povolíte kolizi napájecích napětí, umožní programátor připojit na pin VDD interní napětí ve chvíli, kdy na něm vidí přítomné nějaké napětí. Tím může dojít ke zničení programátoru nebo programované aplikace.

Toto nastavení je určeno pro použití ve velmi specifických

případech. Jedním z nich může být požadavek na programování aplikace, ze které není vyveden pin VDD. V tomto případě je aplikace napájena z vlastního zdroje, ale výstupní budiče programátoru musejí být napájeny naopak z USB. Protože však napětí na datových signálech aplikace proniká přes ochranné diody do výstupní sekce programátoru, vidí programátor přítomné jisté malé napětí na VDD a brání se připojení interního napětí na VDD.

▸ Nezobrazovat varování při zapnutí interních 5 V pro 3,3 V součástku

Varování

Potlačením varování o příliš velkém napětí na součástce hrozí zničení programátoru nebo programované aplikace.

Touto volbou potlačíte varování o zapnutí vyššího napětí na výstup VDD, než je pro programovanou součástku povoleno.

Pokud je v aplikaci integrován napěťový konvertor mezi programovanou součástkou a programovacím rozhraním, může být užitečná možnost programovat při vyšším napětí, než na které je programovaná součástka určena.

Tato volba je dostupná pouze pro programátor PRESTO.

▸ Dovolit změnu velikosti napájecího napětí, když je zapnuté

Aby se u programátoru FORTE předešlo nechtěnému zničení připojené aplikace, je standardně zakázáno měnit velikost výstupního napětí, pokud je toto napětí právě do aplikace připojeno. Povolením této volby bude možné výstupní napětí kdykoliv měnit v rozsahu, který je pro danou součástku specifikován.

▸ Dovolit externí napájení u součástek vyžadujících VPP před VCC

U některých součástek výrobce vyžaduje, aby bylo programovací napětí připojeno před napájecím napětím součástky. Pokud je použito externí napájecí napětí, nelze tento požadavek splnit. V tom případě se zobrazí varování, které lze trvale vypnout touto volbou.

Vypnutí tohoto varování by měl uživatel dobře zvážit, toto varování se zobrazuje u součástek, kde použitím externího napájecího napětí dojde k porušení specifikace výrobce součástky.

▸ Při použití Windows Messages nezobrazovat ostatní varování

Při ovládání programu pomocí Windows Messages nezobrazuje žádná varování, podobně jako ve quiet módu na příkazovém řádku.

▸ Pin T během programování

Touto volbou lze nastavit logickou úroveň na pinu T během programování, zvolená úroveň bude na pinu viditelná v době, kdy bude na pinu VDD přítomné napájecí napětí.

V případě využití pinu T pro programování, nebude tato funkce dostupná.

Tato volba je dostupná pouze pro programátor FORTE.

▸ Pin T po programování

Touto volbou lze nastavit logickou úroveň, která se objeví na pinu T po skončení programování, zvolená úroveň bude na pinu viditelná pouze pokud bude na pinu VDD přítomné napájecí napětí.

V případě využití pinu T pro programování, nebude tato funkce dostupná.

Tato volba je dostupná pouze pro programátor FORTE.

Nastavení → Výběr zařízení a portu

Nastavení týkající se výběru programátoru a komunikačního portu je popsáno v kapitole Volba programátoru.

Nastavení → Výběr jazyka

Klávesová zkratka: Ctrl+L

Pomocí standardního dialogového okna lze vybrat jiný soubor s jazykovou 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 klávesové zkratky pro většinu akcí, které programátor umožňuje.

Zároveň se zde nastavuje požadované chování tlačítka ***GO***.

Nastavení → Zamknout projekt

Po načtení je projekt zamčený proti neúmyslným změnám. K odemčení nebo opětovnému zamčení lze použít tuto funkci.

V případě, že v dialogu uložení projektu bylo zvoleno **Načítat projekt odemčený**, projekt bude po načtení odemčený.

Menu Nápověda

Nápověda → Nápověda k programu

Klávesová zkratka: F1

Tímto příkazem se vyvolá nápověda, kterou právě čtete.

Nápověda → Seznam podporovaných součástek

Tato volba zobrazí seznam součástek podporovaných aktuální verzí programu UP.

Nápověda → Zkontrolovat aktualizace na Internetu

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

Nápověda → ASIX s.r.o. na Internetu

Pomocí této volby otevřete internetové stránky ASIX s.r.o. www.asix.cz, kde jsou k dispozici aktuální ovladače a manuály produktů firmy ASIX s.r.o..

Nápověda → Informace o programu

Základní informace o programu a kontakt na technickou podporu zobrazíte volbou ***Informace o programu***.

5.5.4 Okno nastavení programátoru

V okně nastavení programátoru jsou přehledně zobrazena všechna důležitá nastavení týkající se programátoru a programované aplikace.

Vzhled okna závisí na použitém programátoru a na typu programované součástky.

Okno nastavení programátoru FORTE

Napájení z programátoru

Tento prvek umožňuje uživateli nastavit velikost napájecího napětí přiváděného z programátoru do aplikace.

V klidu

Pokud je tato položka zaškrtnutá, programátor bude napájet aplikaci v době, kdy se neprogramuje.

Během programování

Pokud je tato položka zaškrtnutá, během programování bude použito napájení z programátoru.

Reset

Toto tlačítko umožňuje přepínat úrovně na pinu reset součástky mezi úrovní požadovanou pro reset a vysokou impedancí.

Tlačítko ***Reset*** je aktivně k dispozici, pokud je na napájecím pinu přítomno napětí.

Nastavení spojená s mikrokontroléry RX600

▸ Zamknout pomocí ID

Pokud je tato volba aktivní, během programování Konfigurační paměti se naprogramuje i OSIS(ID), čímž lze zamknout přístup k bootloaderu součástky.

▸ Umožnit použití Configuration Clearing (maže TM, ID)

Funkce Configuration Clearing maže oblast TM (Trusted Memory). Pokud se TM používá, je třeba použití této volby dobře zvážit.

Použití této funkce je také jedinou možností jak smazat OSIS(ID).

▸ Baud Rate

Touto volbou lze nastavit rychlost komunikace mezi programátorem a součástkou.

Nastavení spojená s mikrokontroléry PIC

▸ Způsob programování

- **HVP** Bude použito klasické programovaní s napětím v rozsahu 8 V až 13 V na pinu P
- **LVP** Bude použito programování pomocí LVP pinu mikrokontroléru, na pinu P programátoru jsou přítomny pouze logické hodnoty 0 a 1.

▸ PE

Pro součástky PIC24 a dsPIC33 je možné volbou ***PE*** zvolit metodu programování. PE znamená Programming Executive, tato metoda je obvykle rychlejší. Programming Executive dělá po programování verifikaci, proto jí program UP po programování pomocí PE neprovádí.

Nastavení spojená s mikrokontroléry AVR a 8051

▸ Frekvence oscilátoru

Během programování mikrokontrolérů AVR musí být připojený externí oscilátor nebo funkční interní oscilátor. Zde nastavená frekvence musí odpovídat frekvenci, na které oscilátor programované součástky skutečně běží, za případnými děličkami. Maximální rychlost komunikace s mikrokontrolérem je pak závislá na frekvenci tohoto oscilátoru.

▸ Zrychlené programování s pomalými hodinami

Při programování součástky s použitým pomalejším oscilátorem tato volba umožňuje dosáhnout kratších časů programování.

Při použití této volby budou po smazání součástky pojistky naprogramovány tak, že je nastavena maximální frekvence interního oscilátoru. Tím se docílí toho, že programátor může se součástkou komunikovat vyšší rychlostí. Na konci programování se naprogramuje požadovaná hodnota konfigurační paměti včetně rychlosti hodin.

Tato volba má vliv pouze při programování celé

součástky.

▸ 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 mikrokontroléru, a programátor je připojený přes tento resetovací obvod.

▸ Zapsat RC osc Adjustment

Pokud je tato volba zaškrtnutá, programátor zapíše do pojistky "RC osc Adjustment" hodnotu z okna Konfigurace. V opačném případě programátor do pojistky přepíše hodnotu vyčtenou ze součástky.

▸ HVP

Pokud je tato volba zaškrtnutá, programátor při komunikaci se součástkou použije “vysoké“ napětí na pinu P.

To umožňuje programovat součástku s vypnutým externím signálem RESET.

Nastavení spojená s CH32V003

▸ Fast mode

Pokud je tato volba zaškrtnutá, programátor se součástkou komunikuje dvojnásobnou rychlostí.

Nastavení spojená s paměťmi I2C

▸ Rychlost sběrnice I2C

Zvolte maximální možnou rychlost I^2C sběrnice. Programátor během práce na I^2C sběrnici zapíná interní pull - up o velikosti 2,4 kΩ.

▸ Adresa paměti I2C

Zvolte adresu programované I^2C paměti na sběrnici.

Nastavení spojená s paměťmi SPI Flash

▸ První adresa

Nastavuje adresní rozsah pro práci s pamětí. ***První adresa*** nastavuje první adresu v paměti, kde se bude vykonávat zvolená operace.

▸ Poslední adresa

Nastavuje adresní rozsah pro práci s pamětí. ***Poslední adresa*** nastavuje poslední adresu v paměti, kde se bude vykonávat zvolená operace.

Okno nastavení programátoru PRESTO

V klidu

Pokud je tato položka zaškrtnutá, programátor bude napájet aplikaci v době, kdy se neprogramuje.

Během programování

Pokud je tato položka zaškrtnutá, během programování bude použito napájení z programátoru.

Nastavení spojená s mikrokontroléry PIC

Ovládání pinu -MCLR

Tlačítky ***Spustit***, ***Zastavit***, ***Třetí stav*** a ***Reset*** je možné ovládat logickou hodnotu přítomnou na pinu P1 (VPP) během klidu, pokud je přítomno napájení.

Tlačítko Reset vygeneruje resetovací puls.

▸ Způsob programování

- **HVP** Bude použito klasické programovaní s přítomnými 13 V na P1 (VPP).
- **LVP** Bude použito programování pomocí LVP pinu mikrokontroléru, na pinu P1 (VPP) programátoru jsou přítomny pouze logické hodnoty 0 a 1.

▸ Algoritmus programování

- **Auto** Algoritmus bude vybrán podle aktuálně přítomného napětí na VDD.
- **Vcc=5 V** Bude použit vždy algoritmus pro rychlé 5 V programování.
- **Vcc=2,7 to 5,5 V** Bude použit vždy algoritmus pro pomalé programování pracující ale při všech napájecích napětích.

▸ PE

Pro součástky PIC24 a dsPIC33 je možné volbou ***PE*** zvolit metodu programování. PE znamená Programming Executive, tato metoda je obvykle rychlejší. Programming Executive dělá po programování verifikaci, proto jí program UP po programování pomocí PE neprovádí.

Programování boot paměti

Umožňuje vybrat, která oblast boot paměti se má programovat nebo kontrolovat.

Nastavení spojená s mikrokontroléry AVR a 8051

▸ Frekvence oscilátoru

Během programování mikrokontrolérů AVR musí být připojený externí oscilátor nebo funkční interní oscilátor. Zde nastavená frekvence musí odpovídat frekvenci, na které oscilátor programované součástky skutečně běží, za případnými děličkami. Maximální rychlost komunikace s mikrokontrolérem je pak závislá na frekvenci tohoto oscilátoru.

▸ Zrychlené programování s pomalými hodinami

Při programování součástky s použitým pomalejším oscilátorem tato volba umožňuje dosáhnout kratších časů programování.

Při použití této volby budou po smazání součástky pojistky naprogramovány tak, že je nastavena maximální frekvence interního oscilátoru. Tím se docílí toho, že programátor může se součástkou komunikovat vyšší rychlostí. Na konci programování se naprogramuje požadovaná hodnota konfigurační paměti včetně rychlosti hodin.

Tato volba má vliv pouze při programování celé součástky.

▸ HVP

Pokud je tato volba zaškrtnutá, programátor při komunikaci se součástkou použije “vysoké“ napětí na pinu P.

To umožňuje programovat součástku s vypnutým externím signálem RESET.

▸ 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 mikrokontroléru, a programátor je připojený přes tento resetovací obvod.

Nastavení spojená s paměťmi I2C

▸ Rychlost sběrnice I2C

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

▸ Adresa paměti I2C

Zvolte adresu programované paměti I^2C na sběrnici.

Nastavení spojená s paměťmi SPI Flash

▸ První adresa

Nastavuje adresní rozsah pro práci s pamětí. ***První adresa*** nastavuje první adresu v paměti, kde se bude vykonávat zvolená operace.

▸ Poslední adresa

Nastavuje adresní rozsah pro práci s pamětí. ***Poslední adresa*** nastavuje poslední adresu v paměti, kde se bude vykonávat zvolená operace.

5.5.5 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 navíc doplnit na instrukci RETLW. Tyto volby jsou dostupné z kontextového menu (po kliknutí pravým tlačítkem myši).

Editor paměti 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 datové paměti (EEPROM)

Menu: ***Zobrazit → Zobrazení datové paměti***

Klávesová zkratka k zobrazení okna: F11
Klávesová zkratka k zavření okna: Esc

Editor datové paměti (EEPROM) se používá k zobrazení obsahu přídavné paměti některých součástek, typicky paměti EEPROM.

Ne všechny součástky obsahují přídavnou paměť, pro některé součástky nemusí být proto tento editor dostupný.

Editor konfigurační paměti

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.

Obsah editoru konfigurační paměti je závislý na zvoleném typu programované součástky. K bližšímu vysvětlení jednotlivých voleb tohoto okna je nutné seznámit se s katalogovým listem programované součástky.

Ne všechny součástky potřebují konfigurační data, pro

některé součástky nemusí proto být tento editor dostupný.

Tipy pro pokročilé uživatele

Ačkoliv 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.

Toho může 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 měly být naprogramovány výchozí hodnotou.

Toho může být docíleno zapnutím volby ***Nastavení → Nastavení programu → Editory → Maskovat ID pozice...***.

5.6 Použití programu UP z příkazového řádku

Program UP je možné ovládat i z příkazového řádku.

Program sám ošetří, aby byl spuštěn vždy pouze v jedné instanci (s jedním jménem třídy). Pokud je spuštěna druhá instance programu (se stejným jménem třídy), parametry z příkazového řádku se předají první instanci a ta je provede. Běžící instanci se předávají jen parametry /p, /pdiff, /blank, /verify, /erase, /read, /noe, /eeonly, /noboot, /boot, /code, /cfg. Program je možné spustit ve více instancích s parametrem */wnd* viz přehled parametrů.

Na příkazovém řádku je doporučeno používat soubor projektu, který obsahuje nastavení potřebná pro programování. V případě potřeby lze některá nastavení z projektu upravit parametry příkazového řádku. Parametry příkazového řádku mají větší prioritu než nastavení z projektu definovaném na příkazovém řádku.

5.6.1 Přehled parametrů

UP.EXE [{**/ask** | **/q** | **/q1**}]
[{**/e** ***soubor_s_eeprom.hex*** | [**/noe**]}] [{**/p** |[**/pdiff**]| [**/o**]} ***soubor.hex*** | ***soubor.ppr***] [**/df** ***soubor.hex***] [**/part** ***jméno_součástky***] [**/eeonly**] [**/erase**][**/w**[**nd**] ***up_window_class***] [**/cfg**] [**/devid**] [**/blank**] [**/verify** ***soubor***] [**/read** ***soubor***] [**/s** ***SN_programátoru***][**/progname** ***jméno***] [**/noboot**] [**/boot**] [**/code**] [**/getpartrev**] [**/sn** ***sériové_číslo***] [**/conf** ***soubor***]

Legenda

- Text, který je **tučně**, se píše přímo na příkazový řádek tak, jak je zde uveden.
- 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. Např. { A | B } znamená "zvolte buď A nebo B".
- Text v hranatých závorkách [] označuje nepovinný parametr - může být uveden, ale také nemusí.
- Text v uvozovkách „“ nabízí mnemotechnickou pomůcku

/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. Externí aplikace může sledovat průběh programování čtením hodnoty ProgressBaru, viz Sledování průběhu operace.

/q1 „Quiet“ mód 1. Funguje stejně jako parametr /q, jen během programování zobrazí status formulář, který se po programování bez ohledu na chyby zavře. S parametrem /q1 UP neposkytuje hodnotu ProgressBaru jako s parametrem /q.

/e soubor „EEPROM“ soubor. Zadání případného souboru s daty pro datovou paměť součástky. Pokud má jméno souboru v těle mezery, je třeba jméno souboru uzavřít uvozovkami.

/noe „No EEPROM“. Přeskočí programování datové paměti součástky. Pokud je tento parametr použit při programování mikrokontroléru MSP430, je vynecháno programování a mazání informační paměti.

/p soubor „Programovat“. Zadaný soubor bude naprogramován. Za parametrem může být datový soubor, např. soubor hex, nebo soubor projektu programu UP. Pokud jméno souboru obsahuje mezery, je třeba jej uzavřít mezi uvozovky.

/pdiff soubor Programovat rozdílově. Zadaný soubor se naprogramuje rozdílovým algoritmem. Pokud jméno souboru obsahuje mezery, je třeba jej uzavřít mezi uvozovky.

/o soubor Otevřít. Zadaný soubor je otevřen. Může jít o datový soubor, např. soubor hex, nebo soubor projektu programu UP. Jedná se o nepovinný parametr. Pokud jméno souboru obsahuje mezery, je třeba jej uzavřít mezi uvozovky.

/df soubor Pokud je za parametrem /p nebo /o specifikován soubor projektu a uživatel chce tento projekt použít a jen změnit soubor s daty, použije parametr /df, za kterým definuje jméno nového datového souboru. Pokud jméno souboru obsahuje mezery, je třeba jej uzavřít mezi uvozovky.

/eeonly Provede zvolenou operaci jen s datovou pamětí (EEPROM), u MSP430 jen s Informační pamětí.

/part jméno Vybere v programu UP zvolenou součástku.

/erase Smaže součástku.

/wnd jméno třídy Jiné jméno třídy okna. Pomocí tohoto parametru lze spustit program UP ve více instancích (vícenásobně). 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 mikrokontrolérů 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.

/read soubor Přečte součástku a přečtený obsah uloží do zadaného souboru.

/s SN_programátoru Umožňuje vybrat programátor podle sériového čísla. Sériové číslo se zadává tak, jak je zobrazeno v programu UP nebo natištěno na programátoru. Např. 016709 nebo A6016709. Pokud je místo sériového čísla použit znak *, použije se jakýkoliv dostupný programátor vybraného typu.

/progname jméno Umožňuje zvolit typ programátoru podle jména, např. PRESTO nebo FORTE.

/noboot Přeskočí programování boot paměti MSP430.

/boot Provede zvolenou operaci jen s boot pamětí MSP430.

Poznámka

Mikrokontroléry PIC32 mají také boot paměť, která ale používá pro programování proměnné a formulář datové paměti, takže funguje s parametry pro datovou paměť a nikoliv s parametry pro boot paměť. Podobně je to i u informační paměti mikrokontrolérů MSP430 a CC430.

/code Provede zvolenou operaci jen s pamětí programu nebo s hlavní pamětí součástky.

/getpartrev Pouze přečte revizi součástky a jako návratový kódu vrátí revizi + 0x10000.

/sn sériové_číslo Umožňuje zadat hodnotu sériového čísla, které je vloženo na adresu podle nastavení sériových čísel. V nastavení sériových čísel musí být nastavena sériová čísla počítaná, manuálně. Sériové číslo se zadává v šestnáctkové soustavě, např. 1234ABCD.

/conf soubor Do zadaného souboru se při ukončení programu uloží obsah konzole, pouze při spuštění operace z příkazového řádku.

Poznámka: Pokud není použit parametr vyžadující nějakou operaci, UP se spustí a zůstane běžet. Pokud je použit parametr vyžadující nějakou operaci a software neběží, operace se vykoná a software se ukončí, pokud nepotřebuje interakci uživatele.

Použití projektového souboru

Při programování různých součástek se může stát, že je program nastaven jinak než uživatel předpokládal. **Kdykoliv je to možné, je doporučeno používat projektové soubory** (.PPR), ve kterých jsou uložena veškerá nastavení programu potřebná k programování a cesty k datovým souborům.

Příklady použití

Otevření souboru

```
up.exe project.ppr

up.exe "C:\My Documents\Recent Projects\PIC
\My latest project\project.ppr"
```

Naprogramování součástky

```
up.exe /p project.ppr

up.exe /p "C:\My Documents\Recent Projects\PIC
\My latest project\project.ppr"
```

5.6.2 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.

8 Nepodporováno.

9 Chyba sériového čísla zadávaného parametrem /sn.

10 Chyba, součástka zamčená.

0x10000 + konstanta Revize součástky přečtená s parametrem /getpartrev.

Poznámka: V dávkových souborech je možné získat návratovou hodnotu programu z proměnné *%errorlevel%* .

5.6.3 Sledování průběhu operace

Pokud je aplikace spuštěna na příkazovém řádku ve quiet módu (s parametrem /q), lze z externí aplikace sledovat průběh práce čtením hodnoty hlavního ProgressBaru, kterou program průběžně ukládá do Named Shared Memory.

Program UP sdílí proměnnou o velikosti jeden Integer, k proměnné lze přistupovat použitím funkcí Windows OpenFileMapping a MapViewOfFile. Jméno sdílené proměnné je jméno třídy okna programu UP doplněné o řetězec _Progress. Například, pokud je parametrem /w zvoleno jméno up1, jméno sdílené proměnné bude up1_Progress.

V instalačním adresáři programu UP, v podadresáři example_ProgressBar, lze nalézt příklad čtení hodnoty ProgressBaru v jazyce C.

5.7 Ovládání programu UP pomocí zpráv Windows

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

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

Příkaz se identifikuje podle wParam, parametry podle lParam.

5.7.1 Přehled příkazů

Pokud není specifikováno jinak, vrací příkaz návratovou hodnotu:

0 chyba, nepodařilo se

1 vše provedeno v pořádku

Příkazy s wParam 1, 2, 3, 4, 5, 6, 7 a 24 jsou blokující (thread blocking).

wParam	lParam	Description
0	0	does not perform anything, returns 1
	1	SetForegroundWindow()
	2	Maximize, SetForegroundWindow()
1	any	programs everything, same return code as from the command line
2	any	programming excl. data memory, same return code as from the command line

3	see below	programming incl. erasing, same return code as from the command line
	bit 0 = 1	code/main memory
	bit 1 = 1	data memory (EEPROM)
	bit 2 = 1	configuration memory
	bit 3 = 1	boot memory (MSP430, CC430)
4	see below	reading
	bit 0 = 1	code/main memory
	bit 1 = 1	data memory (EEPROM)
	bit 2 = 1	configuration memory
	bit 3 = 1	boot memory (MSP430, CC430)
5	see below	differential programming
	bit 0 = 1	code/main memory
	bit 1 = 1	data memory (EEPROM)
	bit 2 = 1	configuration memory
	bit 3 = 1	boot memory (MSP430, CC430)
6	see below	verification
	bit 0 = 1	code/main memory
	bit 1 = 1	data memory (EEPROM)
	bit 2 = 1	configuration memory
	bit 3 = 1	boot memory (MSP430, CC430)
7	see below	erasing, same return code as from the command line
	bit 0 = 1	code/main memory
	bit 1 = 1	data memory (EEPROM)
	bit 3 = 1	boot memory (MSP430, CC430)
8	see below	BlankCheck of memory, same return code as from the command line
	bit 0 = 1	code/main memory
	bit 1 = 1	data memory (EEPROM)
	bit 2 = 1	configuration memory
	bit 3 = 1	boot memory (MSP430, CC430)

15	any	presses the GO button, returns 1
16	see below	asks if the programmer supports the function
	0	MCLRControl_Run
	1	MCLRControl_Stop
	2	MCLRControl_Reset
	8	current voltage at pin VDD
17	see below	performs the programmer function
	0	MCLRControl_Run
	1	MCLRControl_Stop
	2	MCLRControl_Reset
	8	current voltage at pin VDD **return value PRESTO:** 0 – unknown level 1 – 0 V 2 – approx. 2 V 3 – approx. 5 V 4 – more than 6 V **return value FORTE:** -1 - measuring error xx - measured voltage × 10 example: 33 means 3,3 V
24	address	reads data from address, returns error code, data are returned with wParam = 25
25	any	returns last data read with wParam of 24
32	see below	UP program initialization
	bit 0 = 1	reload settings (reload project/ini file or registry)
	bit 1 = 1	reload language file
	bit 2 = 1	recreate programmer (like programmer was changed)
	bit 3 = 1	reload programmer settings (like port settings)
	bit 4 = 1	reload selected device
	bit 5 = 1	reload current file (.hex, .bin, ...)

	bit 6 = 1	recreate all dialog windows (adjust their size when reloading device)
	0x 0100	refresh device specific windows
	0x 0200	refresh all editors
	0x 0300	refresh project captions
33	1	save all project settings
48	see below[1]	saves current file (same as Ctrl+S), returns 0 when there was no error else returns a nonzero value
	bit 0 = 1	code/main memory
	bit 1 = 1	data memory (EEPROM)
	bit 2 = 1	configuration memory
	bit 3 = 1	boot memory (MSP430, CC430)
56	0	returns the handle of the UP main form

Tab. 14: Parametry zpráv WM_USER

Zpráva typu WM_CLOSE zavře program UP.

Příklad použití

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

[1] Only parameter lParam=1 or lParam=2 are allowed for AVR microcontrollers.

5.8 Knihovna UP_DLL.DLL

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

Knihovna UP_DLL.DLL komunikuje s programem UP, takže UP musí být při použití této knihovny spuštěn. UP_DLL nemůže pracovat samostatně.

```
unit up_dll;
interface

Function UP_LoadFile (FileName: PChar; style: in
teger): integer; stdcall;
(*
* Load File (with extension .hex or .ppr);
* Loading of .ppr file can result in loading .he
x 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 Va
lue: 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; FileNam
e: PChar; style:integer):integer; stdcall;

Function UP_SetStrValue_Wnd(WndClass:PChar; Valu
```

```
eName: PChar; Value:PChar): BOOL; stdcall;

Function UP_SetIntValue_Wnd(WndClass:PChar; Valu
eName: PChar; Value:integer): BOOL; stdcall;

Function UP_GetStrValue_Wnd(WndClass:PChar; Valu
eName: PChar; Value: PChar; Size: integer): inte
ger; stdcall;

Function UP_GetIntValue_Wnd(WndClass:PChar; Valu
eName: PChar; var Value: integer): LongBool; std
call;

(*
* 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.
*)

Function UP_GetChecksum(Recalculate: integer; Ch
ecksum: PChar; Size: integer): BOOL; stdcall;

Function UP_GetChecksum_Wnd(WndClass:PChar; Reca
lculate: integer; Checksum: PChar; Size: integer
): BOOL; stdcall;

(*
* These functions return checksum value from the
 UP, where:
* Recalculate = 0; only returns the checksum
* Recalculate = 1; checksum will be recalculated
 first
* Checksum; the returned checksum as string
* Size; the length of the Checksum array, the ca
ller have to fill
* WndClass; the UP window class name
*)

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.dl
l';

function UP_SetIntValue_Wnd; external 'up_dll.dl
l';

function UP_GetStrValue_Wnd; external 'up_dll.dl
l';
function UP_GetIntValue_Wnd; external 'up_dll.dl
l';

function UP_GetChecksum; external 'up_dll.dll';
function UP_GetChecksum_Wnd; external 'up_dll.dl
l';

end.
```

Více informací naleznete v kapitole Příloha A - UP_DLL.DLL.

5.9 Spuštění více instancí programu UP

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

Program UP může být při běžném používání spuštěn

pouze jednou. Každé další spuštění programu pouze pošle parametry z příkazového řádku již běžícímu programu nebo ho nějakým jiným způsobem zviditelní.

Program UP lze spustit vícekrát s použitím parametru **/w** pod jiným názvem třídy okna. Komunikovat spolu budou ty programy které budou mít stejný název třídy okna.

Parametry programu při spouštění z příkazového řádku jsou popsány v kapitole Použití programu UP z příkazového řádku.

Příklad použití

První instance programu UP může být spuštěna běžným způsobem ze start menu.

Další instance může být spuštěna z příkazové řádky např jako **up.exe /w "another up"**.

5.10 Přístup více programů k jednomu programátoru

K jednomu programátoru může přistupovat vždy nejvýše jeden program nebo utilita.

O přístupová práva programátorů připojených k rozhraní USB se stará operační systém.

Pokud máme spuštěný program UP, nedovolí operační systém jinému programu přístup ke zvolenému programátoru, protože program UP neustále musí kontrolovat stav tlačítka a stav napětí na napájecím pinu.

Programu UP lze přístup k programátoru dočasně zakázat a uvolnit programátor pro jinou aplikaci pomocí dialogu „Výběr zařízení“. Zvolte ***Nastavení → Výběr zařízení a portu***. Po dobu zobrazení tohoto dialogu může jiný program přistupovat k programátoru. Při zrušení dialogu nedojde ke ztrátě neuložených dat v HEX editorech.

5.11 Aktualizace programu UP

Pokud je na webu nová verze programu, je doporučeno stávající verzi aktualizovat, protože mohou být odstraněny některé známé problémy nebo může být zefektivněn programovací algoritmus konkrétní součástky.

V nových verzích také bývají doplňovány nově podporované typy součástek.

Aktualizace je zdarma a je velmi snadná. Stáhněte si z webové adresy https://www.asix.tech/prg_up_cz.html novou verzi programu a jednoduše ji přeinstalujte přes původní verzi tím, že instalátor spustíte a pokračujete tlačítkem ***Další*** až do konce.

Nemusíte mít obavu ze ztráty nastavení, která máte v aktuální verzi programu. Veškerá nastavení zůstanou zachována.

Automatické zjišťování nových verzí na webu se nastavuje v ***Nastavení → Nastavení programu → Ostatní***.

5.12 Příloha A - UP_DLL.DLL

V této příloze jsou popsána jména nastavení a hodnoty funkcí knihovny UP_DLL.DLL.

Pro lepší pochopení se prosím podívejte na ukázkové dávkové soubory v instalačním adresáři programu UP.

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

5.12.1 Datové typy

string	is string
integer	signed 32bit value

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

5.12.2 Seznam proměnných programu UP

Prog.LoadFileBfgProg
boolean

If true, current file is reloaded before device (or its part) is programmed.

Prog.LoadFileBfgProgWarnMod
boolean

If true, the program warns when it is set to reload current file before programming and data in some editor have been manually modified.

Prog.LoadFileBfgProgWarnOnNoChange
boolean

If true, the program warns when it is set to reload current file before programming and the content of the loaded file has not changed from the previous programming.

File.AutoCheck
boolean

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

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

Part.Name
string

Selected device name

Prog.Name
string

Selected programmer name

Possible values are PREST, FORTE.

Prog.PortBase
integer

Serial number of programmer

Prog.UseOnlyThisSN
boolean

If true, the SN saved in the up.ini is always used regardles of the SN contained in a project file.

LanguageFile
string

Relative path to used languange file

Project.File
string

Project file path

Project.Present

boolean

Project.Template
boolean

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

HexFile.File
string

Opened current file path

HexFile.Present
boolean

HexFile.IHex
boolean

If true, the file with extension different from .HEX or .BIN is loaded as HEX file.

HexFile.Template
boolean

If true, the user is asked for name before saving.

HexFile.SaveVoid
boolean

If true, empty cells are saved, too.

HexFile.AskForBinEndian
boolean

If true, the user is asked if the loaded BIN file should be loaded as Little or Big endian.

HexFile.BinLittleEndian
boolean

If true, the loaded BIN file is loaded as Little endian.

HexFile.LoadDataAuto
boolean

If true, with file is loaded also a separate file for data memory.

HexFile.LastDataFile
string

Path of a separate file for data memory.

HexFile.DataIHex
boolean

If true, the file for data memory with extension different from .HEX or .BIN is loaded as HEX file.

Prog.QBfrEraseFlash
boolean

Question before erasing flash devices

Prog.QBfrProgFlash
boolean

Question before programming flash devices

Prog.QBfrProg
boolean

Question before programming OTP devices

Prog.QBfrDiffProg
boolean

Question before differential programming (of flash devices)

Prog.QBfrProgCP
boolean

Warning before programming device with some kind of protection

Prog.CloseStatOnGoodAct
boolean

If true, status window will be automatically closed after

read/verify etc... without errors.

Prog.CloseStatOnGoodProg
boolean

If true, status window will be automatically closed after programming without errors.

Prog.BeepOnGoodProg
boolean

If true, program makes a sound on successful programming.

Prog.BeepOnBadProg
boolean

If true, program makes a sound on unsuccessful programming.

Prog.SoundsOff
boolean

If true, all sounds of the program are switched off.

Prog.SkipBlankForCfg
boolean

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

Prog.SkipBlankCheck
boolean

If true, no blank check of device is performed before device programming.

Prog.SkipErase
boolean

If true, no erasing is performed before device programming.

Prog.SkipEraseData
boolean

If true, no erasing is performed before only data memory programming.

Prog.SkipLastFFVerify
boolean

If true, verification of empty FF positions at the end of the programmed memory is not preformed.

Prog.SkipVerify
boolean

If true, no verification is performed after programming.

Prog.DoubleVerify
boolean

If true, verification on two supply voltages is performed after programming. Supported by FORTE with internal supply voltage only.

Prog.DoubleVerifyV1
integer

Defines supply voltage size for the first verification, the value is in volts x 10.

Prog.DoubleVerifyV2
integer

Defines supply voltage size for the second verification, the value is in volts x 10.

Serial
integer

0	no serial numbers
1	serial numbers are from external file
2	serial numbers are calculated

Serial.Step
integer

Step 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 device as ASCII characters.

Serial.SaveTo
integer

1	code/main memory
2	data memory
4	ID positions

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 computed, 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 device.

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.

Serial.LogSN
boolean

If true, the result of programming is logged to a selected file.

Serial.LogFile
string

File name of a file where the result of the programming will be logged.

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 5 V

SpecSettings.PREST.ProgPower
integer

0	power supply during programming is External 2.7 to 5.5 V
1	power supply during programming is Internal 5 V

SpecSettings.PREST.i2cSpeed
integer

0	100 kHz
1	500 kHz
2	1 MHz
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 = 5 V
2	assume VDD < 5 V

SpecSettings.PREST.UsePE
boolean

If true, PE (programming executive) is used for programming of PIC MCUs.

SpecSettings.PREST.PIC32MZ_BootProg
integer

0	Lower Boot alias
1	Boot Flash 1 and 2

SpecSettings.PREST.PSoCAlg
integer

0	automatic selection
1	assume VDD > 3.6 V
2	assume VDD =< 3.6 V

SpecSettings.PREST.PSoCRSTInit
integer

0	Reset mode
1	Power cycling

SpecSettings.PREST.MSP430osc
integer

0	Calibrated internal RC oscilator
1	Not calibrated internal RC oscilator

SpecSettings.PREST.MSP430speed
integer

0	Normal
1	Slow
2	Slowest

SpecSettings.PREST.MSP430EraseSegmentA
boolean

If true, MSP430 Segment A is erased.

SpecSettings.PREST.SPI_Flash_Freq
integer

0	3 MHz
1	1.5 MHz
2	750 kHz

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.

SpecSettings.PREST.AVRXTAL.AutoClk
boolean

If true, faster programming with slow clock is used for AVR MCUs.

SpecSettings.PREST.AVRRSTInverse
boolean

If true, the reset signal polarity is assumed to be inverse in comparison with standard polarity of the reset signal of the selected chip. It is supported with AVR and 8051 devices.

SpecSettings.PREST.AVRHVP
boolean

If true, high voltage method is used for AVR MCUs, it is supported for AVR TPI interface only.

SpecSettings.FORTE.UVCCLevel
integer

Defines the size of the internal supply voltage provided by FORTE programmer. The value is in volts x 10.

SpecSettings.FORTE.Power
boolean

If true, idle power supply is provided by FORTE programmer.

SpecSettings.FORTE.ProgPower
boolean

If true, the supply voltage during programming is provided by FORTE programmer.

SpecSettings.FORTE.LVP
integer

0	HVP method
1	LVP method

SpecSettings.FORTE.UsePE
boolean

If true, PE (programming executive) is used for programming of PIC MCUs, when programmed by FORTE programmer.

SpecSettings.FORTE.PIC32MZ_BootProg
integer

0	Lower Boot alias
1	Boot Flash 1 and 2

SpecSettings.FORTE.PSoCAlg
integer

0	automatic selection
1	assume VDD > 3.6 V
2	assume VDD =< 3.6 V

SpecSettings.FORTE.PSoCRSTInit
integer

0	Reset mode
1	Power cycling

SpecSettings.FORTE.MSP430osc
integer

0	Calibrated internal RC oscilator
1	Not calibrated internal RC oscilator

SpecSettings.FORTE.MSP430speed
integer

0	Normal
1	Slow
2	Slowest

SpecSettings.FORTE.MSP430EraseSegmentA
boolean

If true, MSP430 Segment A is erased.

SpecSettings.FORTE.MSP430Interface
integer

0	JTAG
1	SBW

SpecSettings.FORTE.ARMFreq
integer

The value specifies the frequency of connected oscillator or crystal. This value is used for programming of ARM7TDMI.

SpecSettings.FORTE.ARMOscType

integer

0	Crystal
1	Ext. Clock
2	Int.RC 32 kHz

SpecSettings.FORTE.ARMCommFreq
integer

Selects communication frequency between FORTE programmer and the programmed chip. The frequencies are same as stated in the "Communication frequency" ComboBox in UP. 0 means the highest frequency, 1 is lower and so on.

SpecSettings.FORTE.SPI_Flash_Freq
integer

Selects communication frequency between FORTE programmer and the programmed chip. The frequencies are same as stated in the "Communication frequency" ComboBox in UP. 0 means the highest frequency, 1 is lower and so on.

SpecSettings.FORTE.AVRXTAL.DELAY
integer

represents maximum AVR oscillator frequency

value can be found in *.lng files at item

MainForm.PRESTSpecForm.ComboAVRXTALPresto2.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.

This setting can be found in ini file too at [SpecSettings.FORTE], XTALClk.

SpecSettings.FORTE.AVRXTAL.AutoClk
boolean

If true, faster programming with slow clock is used for AVR MCUs.

SpecSettings.FORTE.AVRRSTInverse
boolean

If true, the reset signal polarity is assumed to be inverse in comparison with standard polarity of the reset signal of the selected chip. It is supported with AVR and 8051 devices.

SpecSettings.FORTE.AVRHVP
boolean

If true, high voltage method is used for AVR MCUs, it is supported for AVR TPI interface only.

SpecSettings.FORTE.ATxmegaInterface
integer

0	PDI
1	JTAG

SpecSettings.FORTE.i2cSpeed
integer

0	100 kHz
1	400 kHz
2	1 MHz
3	Maximal

SpecSettings.FORTE.i2cAddr
integer

0	first suitable address or N/A
1	second suitable address

etc...

5.13 Příloha B - Použití ICSP

ICSP (In-Circuit Serial Programming) je způsob programování mikrokontrolérů PIC, který umožňuje programovat součástky již osazené na desce plošných spojů.

Pro programování mikrokontrolérů PIC lze použít dva různé algoritmy: **HVP** (s programovacím napětím na pinu -MCLR/VPP programované součástky) nebo LVP (s použitím pinu LVP).

Programovací algoritmus **LVP** lze zakázat v konfiguračním slově součástky. Mikrokontroléry mají z výroby povolen algoritmus LVP, proto je nutné při prvním programování ošetřit i vstup PGM součástky (po dobu programování pomocí algoritmu HVP musí být vstup PGM v log.0).

Pozn.: Ne všechny mikrokontroléry PIC mají pin PGM.

5.13.1 Piny použité během programování

V této kapitole popíšeme jak ošetřit piny v režimu ICSP programování v závislosti na programovacím algoritmu.

Algoritmus HVP

- -MCLR/VPP musí být oddělen od resetovacích obvodů (např. rezistorem 10 kΩ). Během programování je na tento pin P(VPP) přivedeno programovací napětí. Náběžná hrana a napěťová úroveň VPP nesmí být aplikací ovlivněna. Programátor PRESTO podporuje pouze pevné napětí 13 V na pinu P(VPP). Programátor FORTE dovoluje nastavit napětí na pinu P(VPP) v rozmezí 6,5 V až 17 V. V případě použití programátoru PRESTO prosím zkontrolujte maximální dovolené napětí na pinu -MCLR/VPP programované součástky.
- Pin PGM (pokud součástka má pin PGM) musí být držen v log.0.
- Piny RB6 a RB7 nesmí aplikace během programování ovlivňovat.

Algoritmus LVP (bez VPP)

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

Zatěžování jednotlivých pinů programátoru

Maximální proud odebíraný z výstupů programátoru, z pinu P(VPP) a z pinu VDD naleznete v kapitole Technická specifikace.

Součástky OTP (One-Time Programmable) mají na pinu P(VPP) výrazně větší spotřebu než součástky s pamětí FLASH, proto by aplikace na pinu P(VPP) u OTP neměla mít další přidanou spotřebu.

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

5.13.2 Možnosti napájení

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

- externí z aplikace
- interní z programátoru

Externí napájení z aplikace nelze použít u některých vybraných typů mikrokontrolérů, které mají pin -MCLR/VPP konfigurovatelný i jako I/O.

Interní napájení lze použít pouze v případě, pokud aplikace nebude z napájecího pinu (VDD) programátoru odebírat příliš velký proud. Maximální povolená velikost odebíraného proudu je uvedena v technické specifikaci.

Programátory PRESTO i FORTE obsahují nadproudovou ochranu. V případě programátoru PRESTO se jedná o softwarovou ochranu, kdy odpojení výstupu zajistí běžící program UP po detekci přetížení. V případě programátoru FORTE se jedná o hardwarovou ochranu, která není na stavu obslužného software závislá.

Při znatelném překročení maximálního zatížení po dobu delší než nastavenou programátor vypíná napájení. Programátor PRESTO kontroluje stav přetížení po celou dobu zapojeného napájení.

U obou programátorů PRESTO i FORTE je podpora pro externí napájení zabudovaná přímo v hardware. Programátor napájí vstupní a výstupní obvody napětím, které je připojeno na pinu VDD. Napětí může být i nižší než 5 V.

Upozornění

Prosíme, věnujte při návrhu zvýšenou pozornost tomu, které typy programovaných součástek lze při napětí nižším než 5 V nejen provozovat, ale i programovat.

Kapacity na napájení v aplikaci

Pokud jsou v aplikaci na napájení přítomny kapacity, které zpomalují zapnutí a vypnutí napájení a programuje se s napájecím napětím z programátoru, je nutno, aby v ovládacím programu UP byly nastaveny delší předpokládané časy nabití a vybití. V opačném případě bude program hlásit chybu nadproudu na napájecím napětí.

Orientační doba, která by měla být nastavená v programu, je přibližně

t[µs] = 2,5 × C[µF] × R[Ω].

Velikost ekvivalentního odporu je v následující tabulce.

Programátor	Nabíjecí proud	Vybíjecí proud
PRESTO	odpovídá 50 Ω	odpovídá 1 kΩ
FORTE	odpovídá 50 Ω	odpovídá 27 Ω

Tab. 15: Ekvivalentní odpor pinu VDD

Příklad

Pro aplikaci s kondenzátorem 33 µF programovanou programátorem PRESTO je potřebná doba pro nabíjení

2,5 × 33 × 50 = 4125 µs
a pro vybíjení 2,5 × 33 × 1000 = 82,5 ms.

Více informací o nastavení je v kapitole ▸ Čas pro zapnutí/vypnutí VDD při napájení z programátoru.

Poznámky

- Někdy může nastat chyba, kdy UP nemůže programovat kalibrační slovo nebo se vyskytují chyby během čtení device ID nebo UP upozorňuje na nadproud na VDD, atp. V těchto případech může pomoci prodloužení nabíjecích a vybíjecích časů až na několik sekund v menu Čas pro zapnutí/vypnutí VDD při napájení z programátoru.
- Pokud UP upozorňuje na chybu nadproudu na VPP, může pomoci použití kratšího ICSP kabelu (maximální délka je 15 cm).

5.13.3 Konektor ICSP

Všechny programátory ASIX používají pro programování algoritmem ICSP jednotný konektor s piny v rastru 2,54 mm.

Tento konektor má 8 pinů se 7 signály. Pro vlastní programování nejsou vždy potřeba všechny signály.

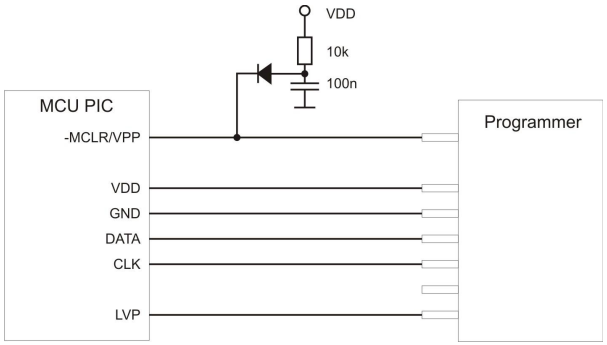
Více informací o zapojení programované součástky k programátoru je v kapitole Popis propojení s aplikací

Číslo pinu	Signál	Programovací konektor
1	-MCLR	P / VPP

2		not used (key)
3	VCC	VDD
4	GND	GND
5	RB7	D / DATA
6	RB6	C / CLOCK
7		not used
8	RB3/RB4/RB5	L / LVP

Tab. 16: Konektor ICSP

Následující obrázek ukazuje doporučené zapojení pinu -MCLR/VPP mikrokontrolérů Microchip s ohledem na programování součástky pomocí ICSP rozhraní.



Obr. 48: Doporučené zapojení pinu -MCLR/VPP

5.14 Příloha C Formát Intel-HEX souboru

V této příloze je popsán formát souborů Intel-HEX používaných programem UP ke čtení i ukládání dat. Běžná přípona takového souboru je .hex.

5.14.1 Podporované varianty souborů Intel-HEX

Program UP podporuje 2 základní varianty souborů Intel-HEX:

- "obyčejný", někdy též 8-bit Intel-HEX File (například MPASMWIN firmy Microchip generuje tento soubor při parametru INHX8M).
- "rozšířený", někdy též 32-bit Intel-HEX File (například MPASMWIN firmy Microchip generuje tento soubor při parametru INHX32).

5.14.2 Popis formátu souboru Intel-HEX

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 Intel-HEX)
	04 - Rozšířená lineární adresa. (pouze 32-bit Intel-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
```

02	Délka záznamu. Velikost konfigurační paměti je jedno slovo = 14 bit = 2 byte (zarovnáno na celé byty)
400E	Adresa záznamu. Adresa konfigurační paměti je slovo 2007h, adresováno po bytech tedy 400Eh
00	Typ záznamu. datový záznam
413F	Data záznamu. Konfigurační slovo je 3F41h
30	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.

Například procesory rodiny PIC18F mají uloženou konfigurační paměť na adrese 0x 30 00 00 00.

Pokud je třeba použít tuto adresu, je nutné do souboru .hex vložit řádek s rozšířenou lineární adresou, který obsahuje horních 16 bitů adresy. Dolních 16 bitů je načteno z řádku s datovým záznamem.

Například tímto řádkem se vybírá konfigurační paměť u součástek rodiny PIC18F.

```
:020000040030CA
```

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 souboru .hex

Velmi často se stává, že dojde k záměně mezi vybraným typem součástky a typem součástky, pro kterou byl soubor .hex 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 takto modifikovaný soubor nelze považovat za vyhovující formátu Intel-HEX.

6

Knihovna up_control.dll

Knihovna up_control.dll umožňuje uživateli ovládat software UP v ní obsaženými funkcemi. Knihovna obsahuje základní programovací funkce.

Z pohledu uživatele program UP běží neviditelně.

Více informací o knihovně je dostupných v samostatném dokumentu.

7

Knihovna FORTE.DLL

Funkce implementované v knihovně forte.dll umožňují na jednotlivých pinech programátoru FORTE nastavovat logické úrovně a číst jejich stav, takto lze vytvářet různé komunikační protokoly.

Kromě funkcí umožňujících ovládání jednotlivých pinů knihovna obsahuje také funkce připravené pro komunikaci po sběrnicích SPI, I^2C a 1-Wire, funkce umožňující ovládat napájecí a programovací napětí programátoru a funkce, kterými lze číst velikost napájecího napětí a stav tlačítka GO na programátoru.

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

8

JTAG PLAYER

JTAG PLAYER je utilita sloužící k programování součástek s rozhraním JTAG pomocí programátorů PRESTO a FORTE.

Tato utilita není součástí instalace programu UP a je možné ji stáhnout samostatně z adresy https://www.asix.tech/prg_jtag-svf-player_cz.html

Utilita JTAG PLAYER je lokalizována pouze v anglickém jazyce.

8.1 Programování JTAG součástky

Po spuštění utility JTAG PLAYER vyberte programátor pomocí menu ***Options → Select Programmer***.

Pokud v zobrazeném dialogu nevidíte připojený programátor, zkontrolujte, zda na programátoru svítí zelená LED „ON-LINE“, a pokud ano, zda s programátorem právě nekomunikuje jiný program.

Nyní pomocí menu ***File → Open&Process*** proveďte naprogramování součástky. Programovaný soubor musí být ve formátu SVF/XSVF. Více informací o formátu souboru naleznete v kapitole Soubor typu SVF a Soubor typu XSVF.

Pokud došlo během programování k nějaké chybě, zkontrolujte propojení programátoru a programované aplikace. V případě chyby také zkontrolujte přítomnost napájecího napětí.

Důležité upozornění
Během programování pomocí rozhraní JTAG musí být přítomno externí napájecí napětí. Programátor v tomto režimu nenapájí programovanou aplikaci.

8.1.1 Soubor typu SVF

Soubor typu SVF (Serial Vector Format) (*.svf) je soubor používaný pro popis vysokoúrovňových operací sběrnice IEEE 1149.1.

SVF je doporučený formát souboru pro všechna testování a programování, kromě Xilinx CPLD XC9500. Pro programování Xilinx CPLD XC9500 je doporučený formát XSVF (Xilinx Serial Vector Format) s příponou *.xsvf.

Příklady jak vytvořit soubor typu SVF

V této kapitole popíšeme způsoby vytvoření souboru typu SVF pro různé typy součástek.

Programování Atmel AVR (např. ATmega128)

Vygenerujte soubor typu SVF použitím programu **avrsvf.exe** dostupného na stránkách firmy ATMEL v sekci Tools & Software of AVR 8-bit RISC MCUs.

Příklad:

```
avrsvf -datmega128 -s -e -ifmyfile.hex -pf -vf -ovmyfile.svf -mp
```

Tento příklad ukazuje, jak ze souboru myfile.hex vytvořit soubor typu SVF, který jtagplay.exe použije pro mazání, programování a verifikaci. Pro více informací spusťte **avrsvf -h**.

Během programování musí být součástka v resetu.

Poznámky

Některé součástky AVR nepodporují programování po stránkách paměti. V tom případě musí být soubor typu SVF vytvořen bez parametru -mp.

Programování Lattice CPLD

Soubor typu SVF může být vytvořen ze souboru .JED použitím programu ispVM System. Tento program je součástí prostředí ispLEVER Classic, který je dostupný na stránkách firmy Lattice.

Programování Altera CPLD

Program QUARTUS II firmy Altera umí generovat soubor typu SVF – nutné nastavit v menu.

Avšak tento soubor .svf nemůže být použit tak, jak je, kvůli špatnému Silicon ID. Podle vyjádření firmy Altera je soubor .svf určen pouze pro Automatic Test Equipment (ATE) programátory a Altera neuvažuje podporu jiných.

Nicméně soubor .svf může být pro naše potřeby manuálně opraven. Pro tuto opravu, prosím, smažte nebo zakomentujte sekci "CHECKING SILICON ID" v souboru .svf.

Stav implementace SVF souborů

Podpora souborů typu SVF byla implementována podle "Serial Vector Format Specification, Revision E" dostupné na www.asset-intertech.com/support/svf.html s těmito omezeními:

- SVF příkazy PIO a PIOMAP nebyly implementovány.
- HDR+SDR+TDR / HIR+SIR+TIR délka je limitována na 2^31 bitů.
- Maximální podporované frekvence TCK jsou 3 MHz, 1,5 MHz, 750 kHz a zlomky 1 MHz začínající na 500 kHz pro programátor PRESTO. Programátor FORTE přidává možnosti 15 MHz, 10 MHz a 5 MHz.
- RUNTEST MAXIMUM max_time SEC parametr je ignorován.
- RUNTEST run_count je limitován na 2^31/3 (přibližně 715 milionů).
- RUNTEST min_time SEC je limitován na 2^31/3 ms (přibližne 715 sekund).
- TRST a RUNTEST SCK - příkazy sdílí stejný konfigurovatelný pin P / VPP programátoru PRESTO i programátoru FORTE, a nikdy nemohou být použity společně

8.1.2 Soubor typu XSVF

Soubor typu XSVF (Xilinx Serial Vector Format) (*.xsvf) je soubor používaný pro popis vysokoúrovňových operací sběrnice IEEE 1149.1. s rozšířením firmou Xilinx.

XSVF je doporučený formát souboru pro programování Xilinx CPLD XC9500.

Příklady jak vytvořit soubor typu XSVF

Programování Xilinx CPLD

K vytvoření souboru typu XSVF (.xsvf) použijte program iMPACT dostupný na stránkách firmy Xilinx.

V dialogu ***Operation Mode Selection***, který se objeví na začátku po startu programu iMPACT, zvolte ***Prepare Configuration Files → Boundary-Scan File → XSVF File***. Spusťte všechny operace (Erase, Program, Verify, Test, ...) stejným způsobem jako když je připojený nějaký programátor (e.g. Xilinx Parallel Cable). Potom nový soubor uložte a zavřete iMPACT. Provedení tohoto souboru vykoná zaznamenané operace.

Pro programování rodiny Xilinx XC9500 **nedoporučujeme** použití souboru (.svf), protože programovací algoritmus součástek XC9500/XL/XV nemůže být v SVF souboru správně popsán.

Stav implementace souborů typu XSVF

Podpora souborů typu XSVF byla implementována podle specifikace "XAPP503, Appendix B: XSVF File Format" dostupné na stránkách firmy Xilinx s těmito omezeními:

- XSVF příkazy XSETSDRMASKS, XSDRINC a XSIR2 nebyly implementovány.
- Pro programování rodin Xilinx XC9500/XV/XL je doporučeno používat jen soubory formátu XSVF.

Pro všechny architektury, kromě XC9500/XV/XL, důrazně doporučujeme použít soubor typu SVF. Pro programování XC9500/XV/XL doporučujeme použít soubor typu XSVF, v souboru typu SVF chybí příkaz XREPEAT, který je pro programování XC9500/XV/XL nutný.

Varování: Provádění souboru s příkazem XREPEAT může být velmi pomalé.

8.1.3 Programovací konektor

V následující tabulce popíšeme význam pinů programátorů PRESTO a FORTE během programování pomocí JTAG rozhraní.

PRESTO	FORTE	Popis funkce
VPP	P	SCK (System Clock) / Uživatelsky definovaný stav během vykonávání souboru a po něm.
-	-	nezapojený pin (klíč)
VDD	VDD	napájení I/O budičů
GND	GND	zem I/O budičů
MOSI	D	JTAG TDI (Test Data In)
CLOCK	C	JTAG TCK (Test Clock)
MISO	I	JTAG TDO (Test Data out)
LVP	L	JTAG TMS (Test Machine State)

Tab. 17: Programovací konektor JTAG

8.2 Popis nastavení

V této kapitole popíšeme význam jednotlivých nastavení menu ***Options → Program Options***.

8.2.1 Default TCK signal frequency

Tato frekvence TCK hodin bude použita do doby než JTAG Player narazí na první příkaz FREQUENCY v souboru typu SVF nebo až do příkazu FREQUENCY s hodnotou "default".

Formát XSVF nepodporuje příkaz FREQUENCY default, proto je frekvence daná parametrem ***Default TCK signal frequency*** použita pro všechny operace.

Maximální frekvence pro programátor PRESTO je 3 MHz; limit pro programátor FORTE je 15 MHz.

Pokud je vybrána volba ***Ignore FREQUENCY commands***, programátor použije pouze frekvenci nastavenou uživatelem a příkazy FREQUENCY budou ignorovány.

8.2.2 Fast Clocks Option (FORTE only)

Volba je dostupná pouze pro programátor FORTE.

Podle JTAG specifikace je signál na TDI vzorkován na náběžnou hranu TCK. Pokud je však vyžadována vyšší frekvence (asi 5 MHz a vyšší), může být užitečné změnit okamžik vzorkování z náběžné hrany na sestupnou hranu posunutím okamžiku vzorkování o 1/2 periody TCK. Pro tuto funkci zvolte ***Fast Clock Option***.

8.2.3 RUNTEST without run_count (SVF only)

Při vykonávání souboru .svf by měl programátor zůstat po specifikovanou dobu ve specifikovaném stavu a přitom generovat hodinový signál na TCK.

Specifikovaný čas může být překročen, ale tím se zpomaluje programování. Ačkoliv to není podporováno specifikací SVF, mnoho programovatelných součástek umožňuje zastavit hodiny na TCK během této doby.

Je třeba zvážit schopnost programátoru dodržet přesný čas. Vysoká přesnost nemůže být dosažena pokud je použita maximální frekvence (programátor může pouze zaručit dodržení min_time SEC parametr). S pomalými hodinami (~100 kHz) může být dosaženo lepší přesnosti. Pokud se hodinový signál nepoužije vůbec, programátor může dodržet min_time SEC parametr téměř přesně.

Vzhledem k těmto faktům jsou dostupné tři možnosti:

- no clock on TCK
- slow clock on TCK (~100 kHz)
- default speed clock on TCK

Příklad: "RUNTEST 3E-3 SEC;" znamená "Generuj hodiny na TCK minimálně po dobu 3 ms".

8.2.4 RUNTEST timing multiply (both SVF and XSVF)

Doporučené hodnoty:

- pro přesné časování specifikované v souboru .svf/.xsvf: 0% (žádný přidaný čas)
- pro rodinu XC9500(XL): 100% nebo více
- pro Atmel AVR (např. ATmega128): 25%

8.2.5 RUNTEST with run_count and no timing (both SVF and XSVF)

Tento příkaz by měl být interpretován jako minimální frekvence na TCK.

Avšak některé generátory souborů typu SVF (např. Xilinx iMPACT) používají tento příkaz jako dobu čekání a předpokládají frekvenci 1 MHz. V takovém případě je doporučené nastavení "interpret as RUNTEST min_time with scale 1 MHz".

Chování JTAG Playeru, když narazí na příkaz RUNTIME se specifikací MINTIME

(Týká se pouze souborů typu SVF, protože u varianty RUNTEST použitelné v XSVF XRUNTEST run_count není možné specifikovat čas.)

- RUNTEST příkaz s run_count a specifikovaným min_time je vykonán na současné TCK frekvenci. Proto může příkaz trvat mnohem delší čas specifikovaný min_time.

- RUNTEST příkaz s run_count a specifikovaným max_time je vykonán na současné TCK frekvenci. Programátor nemůže respektovat “deadline” specifikovaný parametrem max_time, tento parametr je ignorován.

8.2.6 VPP PRESTO / P FORTE pin usage while running test (file) / after test completion

Volba funkcí pinu VPP / P: TRST nebo SCK popsané v souboru nebo uživatelsky volitelné výstupní úrovně (Vhodné pro držení součástky ve stavu reset během vykonávání souboru.).

8.2.7 Default Settings

V JTAG Playeru je připraveno několik výchozích nastavení, tato nastavení jsou primárně určena pro použití s programátorem FORTE, ne však pro programátor PRESTO. Neváhejte, prosím, změnit tato nastavení, pokud nevyhovují Vaší aplikaci.

Default Settings for FPGAs

Default TCK frequency: 15 MHz; Ignore FREQUENCY commands

Fast Clock Option (FORTE only): 5 MHz and above

RUNTEST without run_count (SVF only): default speed clock on TCK

RUNTEST timing multiply (both SVF and XSVF): 0%

RUNTEST with run_count and no timing (both SVF and XSVF): interpret as RUNTEST min_time with scale 1 MHz

VPP PRESTO / P FORTE pin usage while running test (file): Tristate

VPP PRESTO / P FORTE pin usage after test completion: Tristate

Default Settings for XC9500

Default TCK frequency: 5 MHz; Ignore FREQUENCY commands

Fast Clock Option (FORTE only): 5 MHz and above

RUNTEST without run_count (SVF only): slow clock on TCK (~100 kHz)

RUNTEST timing multiply (both SVF and XSVF): 100%

RUNTEST with run_count and no timing (both SVF and XSVF): interpret as RUNTEST min_time with scale 1 MHz

VPP PRESTO / P FORTE pin usage while running test (file): Tristate

VPP PRESTO / P FORTE pin usage after test completion: Tristate

Default Settings for AVR:

Default TCK frequency: 1 MHz; Ignore FREQUENCY commands

Fast Clock Option (FORTE only): 5 MHz and above

RUNTEST without run_count (SVF only): default speed clock on TCK

RUNTEST timing multiply (both SVF and XSVF): 25%

RUNTEST with run_count and no timing (both SVF and XSVF): interpret as RUNTEST min_time with scale 1 MHz

VPP PRESTO / P FORTE pin usage while running test (file): Tristate

VPP PRESTO / P FORTE pin usage after test completion: Tristate

8.3 Spuštění JTAG Playeru z příkazového řádku

Utilita JTAG Player může být, pro větší komfort především během ladění, spouštěn z příkazového řádku s následujícími parametry:

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

-p	automaticky vykoná soubor specifikovaný parametrem -f filename
-f *filename*	specifikuje soubor .svf/.xsvf, který má být vykonán
-i *inifile*	soubor s počátečními nastaveními
-c	zavře program, pokud byl soubor vykonán bez chyb
-cc	zavře program, dokonce pokud byl soubor vykonán s chybami
-s *serial*	použije programátor PRESTO nebo FORTE se specifikovaným sériovým číslem, pokud je místo sériového čísla uveden znak "-" (pomlčka), použije se jakýkoliv připojený programátor bez ohledu na sériové číslo
-forte	použije programátor FORTE, ne PRESTO

Návratové kódy

Program jtagplay.exe vrací tyto chybové kódy:

0	vykonávání posledního souboru bylo bez chyb
1	během vykonávání posledního souboru se vyskytla chyba
2	vykonávání posledního souboru nemohlo být spuštěno

9

MultiUP

MultiUP je utilita sloužící k programování až 4 programátory najednou. MultiUP používá ke své funkci program UP.

Utilita je součástí instalace programu UP.

MultiUP snadno jedním tlačítkem spustí programování vybrané úlohy, která je definována projektem programu UP a dalšími nastaveními.

Pro zvýšení rychlosti program načítá data pro programování jen jednou, je vhodný pro použití ve výrobě.

9.1 Nastavení programování

V menu "Nastavení/Nastavení programu" lze zvolit programátory, které se použijí při programování, projekt programu UP, který budou programovat, zda budou programovat běžným způsobem nebo rozdílově a paměti, které se mají programovat.

Dále je pro zlepšení přehlednosti možné pro každý programátor zvolit vlastní jméno.

Každý z programátorů může programovat jiný projekt nebo mohou být všechny programované projekty shodné.

Všechny programátory nemusí být stejného typu (PRESTO, FORTE), ale je potřeba, aby vybraný projekt programu UP používal shodný typ programátoru.

V nastavení lze také zvolit klávesové zkratky pro spouštění práce jednotlivých programátorů a nastavit ukládání výpisu konzole při ukončení programu.

Parametry zvolené v nastavení lze použít jen jednorázově nebo je uložit do souboru typu MPPR pro další použití.

9.2 Programování

Pro programování předdefinovaných úloh je vhodné otevřít projekt utility MultiUP v menu "Soubor/Otevřít soubor MPPR". Projekt lze vytvořit v menu "Nastavení".

Následně stačí jednotlivá programování spouštět buď samostatně tačítkem "Spustit" na panelu příslušného programátoru nebo všechna aktivní programování najednou tlačítkem "Spustit vše", případně přiřazenými klávesovými zkratkami.

9.3 Příkazový řádek

Jako parametr na příkazovém řádku lze utilitě MultiUP zadat cestu k projektu MPPR, který se po spuštění načte.

Příklad:

multiup.exe C:\data\project.mppr

10

JAK POSTUPOVAT PŘI POTÍŽÍCH

V následující kapitole popíšeme, jak postupovat v případě potíží s programováním součástky. Popíšeme také testovací utility pro programátor FORTE, které slouží ke zjištění, zda není programátor poškozen. Všechny rady a tipy je vhodné projít ještě dříve, než pošlete programátor na opravu do servisu.

Testovací utility nejsou součástí instalace programu UP a je možné je stáhnout samostatně z adresy https://www.asix.tech/prg_testers_cz.html. Lokalizovány jsou pouze v anglickém jazyce.

10.1 Rady a tipy

V případě potíží s programováním součástky je vhodné ověřit následující body:

- Zkontrolovat zapojení mikrokontroléru a programátoru dle popisu propojení v kapitole Popis propojení s aplikací. Byť se to zdá banální, je vhodné toto zapojení zkontrolovat 2x a v druhém případě již předpokládat, že ani vodič vždy nevede nebo vede na více míst, než bychom si přáli.
- Pokud má být pro programování přítomen krystal či další podpůrné součástky (typicky pull-down nebo pull-up rezistory), zkontrolujte, zda jsou hodnoty i zapojení v pořádku.
- Je vhodné a často i nutné, aby na napájení u programované součástky byly blokovací kondenzátory.
- Pokud má programovaná součástka více napájecích pinů, je nutné, aby všechny byly skutečně napájeny. Užitečné bývá zkontrolovat napájecí napětí voltmetrem.
- Je doporučeno používat aktuální verzi programu UP. Pokud máte starší verzi, je doporučeno ji aktualizovat.
- Programujete novou součástku nebo již byla předtím programována? Pokud programována byla, můžou být potíže způsobeny nevhodně nastavenými pojistkami v programované součástce.
- Délka ICSP kablíku by neměla překročit 15 cm.
- Je nutné, aby na datových pinech používaných k programování nebyla přítomna dodatečná kapacita. Zvažte, zda obvody na programovacích pinech nezpůsobují příliš velkou zátěž pro programování.
- Je důležité, aby u součástek, kde se nastavuje rychlost krystalu či komunikace, bylo toto nastavení v souladu se skutečností.
- Některé součástky je možné programovat více různými způsoby. Potíže s programováním mohou být způsobeny i volbou jiného programovacího rozhraní nebo režimu programování, než na který je aplikace připravena.

Pokud žádná z uvedených rad nepomohla problém odstranit a je vyloučené, že programovaná součástka je vadná, může se jednat o vadný programátor.

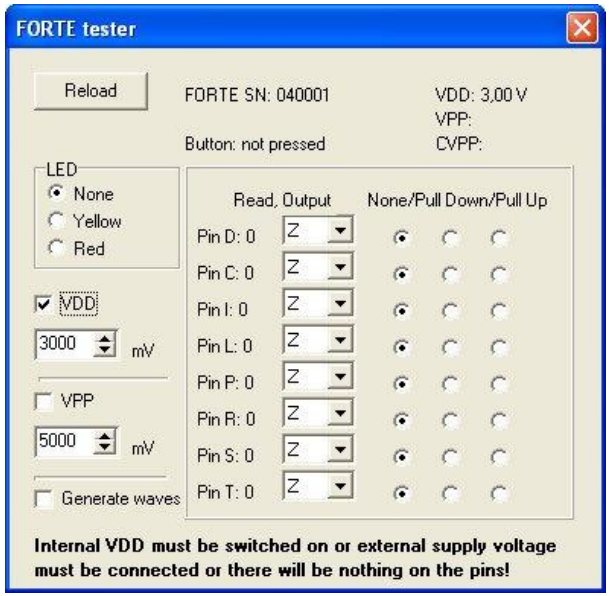
K jednoduchému zjištění, zda je programátor v pořádku, slouží testovací utility popsané v následujících kapitolách.

10.2 FORTE Tester

Utilita FORTE Tester slouží k jednoduchému otestování funkčnosti programátoru FORTE přímo u zákazníka.

Aby bylo možné ji úspěšně spustit, musí být k PC připojen právě jeden programátor FORTE a jeho ovladače musejí být správně nainstalovány. To se nejlépe pozná podle

toho, že svítí zelená dioda ON-LINE.



Obr. 49: Tester FORTE

Po spuštění testeru je nutné zajistit, aby výstupní budiče programátoru byly napájeny. Buď je možné zapnout napájení ***VDD*** a zvolit požadované napětí nebo připojit externí napájecí napětí na pin VDD. Velikost detekovaného napětí se vypisuje v horní části okna.

Základní komunikaci s programátorem snadno ověříme rozsvícením diody ***Active***. Můžeme zvolit žlutou nebo červenou barvu svícení.

V horní části okna je vypsáno také sériové číslo programátoru, které je nutné pro komunikaci se servisem. Toto číslo je zároveň uvedeno na etiketě s čárovým kódem na spodní straně programátoru.

Pod sériovým číslem je místo, kde se vypisují případné chybové hlášení o detekci nadproudu nebo přepětí.

Na dalším řádku vidíme informaci o stisku tlačítka GO.

Velikost programovacího napětí na pinu P ověříme zapnutím volby ***VPP*** a nastavením požadované úrovně napětí. Detekovaná velikost napětí se zobrazuje v pravém horním rohu okna. Hned pod napětím je vypsána velikost detekovaného proudu na pinu P. Protože výstup P obsahuje zabudovaný dělič o celkové velikosti odporu cca 5 kΩ, je zobrazována nenulová hodnota proudu i při odpojeném výstupu pinu P, způsobená právě odběrem děliče.

Pokud jsou výstupy ve stavu vysoké impedance (Z), jsou zobrazovány informace o vstupních logických úrovních na těchto pinech. Zapojením pull-up rezistoru na vybraný pin se začne číst log. 1, v případě pull-down rezistoru se přečte log. 0.

Výstupy je možné nastavit individuálně do log. 1 a log. 0. Kvalitu výstupního signálu můžeme ověřit pomocí voltmetru.

Pomocí volby ***Generate waves*** můžeme posílat na všechny výstupy zároveň periodický signál s frekvencí cca 1 kHz.

Čtení vstupů ani nastavení výstupů nefunguje, pokud není přítomno napájení na pinu VDD.

Pokud se vše zdá být v pořádku, ale přesto se nedaří součástku programovat, je vhodné zkontrolovat propojení s aplikací dle návodu na připojení mikrokontroléru k programátoru.

V případě nejasností, zda je programátor vadný nebo zda je funkční, neváhejte prosím kontaktovat technickou podporu.

11

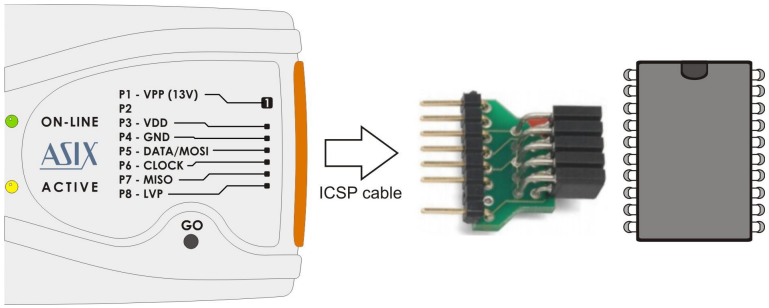
Adaptér HPRAVR

HPRAVR je volitelné příslušenství k programátorům PRESTO a FORTE pro programování mikrokontrolérů AVR v aplikaci se standardním konektorem ISP10PIN na straně programované součástky a s kabelem ICSPCAB8 na straně programátoru. Konektor ISP10PIN se obvykle používá na deskách s mikrokontroléry typu AVR, jako je např. STK500.

11.1 Použití

Připojte adaptér HPRAVR ke konektoru v aplikaci. Ujistěte se, že pin 1 redukce HPRAVR je připojen na pin 1 konektoru ISP10PIN aplikace (Pin 1 je na HPRAVR označený červenou tečkou, na připojené aplikaci může být označen různě - viz informace v příslušném návodu).

Nyní propojte redukci a PRESTO ICSP kablíkem. Pin 2 kabelu je použit jako klíč. Typické propojení programátoru PRESTO a HPRAVR s aplikací je na obrázku níže.



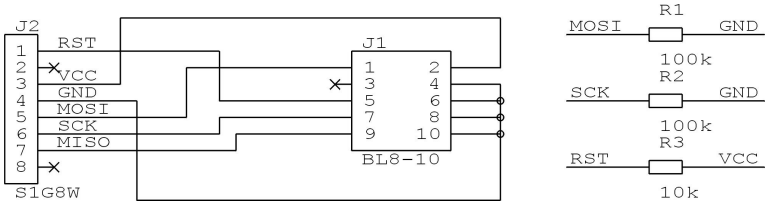
Obr. 50: Použití HPRAVR

Důležité upozornění

Pin č.1 redukce HPRAVR je označen červenou barvou. Zkontrolujte prosím, kde je pin č. 1 v programované aplikaci, protože v případě špatného připojení redukce hrozí zničení programované aplikace.

MOSI	1	2	VCC
NC	3	4	GND
RST	5	6	GND
SCK	7	8	GND
MISO	9	10	GND

Obr. 51: ISP10PIN, pohled shora



Obr. 52: Schéma adaptéru HPRAVR

12

Historie dokumentu

Revize dokumentu	Provedené úpravy
3.7.2013	Dokument byl nově vytvořen.
18.12.2013	Provedeny drobné opravy v textu
25.2.2014	Do příkladů propojení přidána rodina mikrokontrolérů STM8
28.2.2014	Do kapitoly Rozhraní JTAG přidána rodina mikrokontrolérů NXP LPC2xxx
	Přidán parametr [/read] příkazového řádku
1.7.2014	Do manuálu přidáno doporučené zapojení rozhraní SWD mikrokontrolérů ARM.
	Certifikát CE nahrazen prohlášením o shodě.
23.9.2014	Odstraněna poznámka o resetu pro MCU ARM SWD.
	Popsána nová volba - Zobrazovat ASCII překlad pouze nejnižšího bytu slova
21.11.2014	Aktualizován seznam proměnných pro up_dll.dll
25.11.2014	Doplněna nová nastavení programu UP.
3.2.2015	Doplněn popis zobrazení konzole.
	Doplněn popis proměnné PIC32MZ_BootProg.
	Drobné opravy textu.
12.3.2015	Změněn název položky Čas pro zapnutí/vypnutí VDD
	Přidán popis nastavení výpočtu checksumu
7.4.2015	Aktualizován postup jak vytvořit SVF soubor pro Lattice.
19.5.2015	Přidána kapitola o forte.dll.
	Přidáno doporučené zapojení rozhraní C2 mikrokontrolérů C8051 a EFM8.
	Doplněna definice vstupního napětí na pinech programátoru.
	Opraven popis parametru w=32 pro Windows messages.
5.8.2015	Upraven popis způsobu instalace ovladače programátoru.
	Doplněn nový parametr /getpartrev.
	Doplněny chybové kódy pro příkazový řádek.
18.9.2015	Doplněn popis nastavení varování, když soubor není zarovnaný na velikost slova.
19.11.2015	Doplněn popis nastavení pinu T během a po programování.
22.1.2016	Doplněn popis funkce logování sériových čísel.
8.3.2016	Doplněn popis funkce zobrazení formuláře programátoru.
	Doplněn popis nastavení umožňující vypnout varování pro algoritmus VPP před VCC.
20.5.2016	Doplněn popis funkce PE.
	Doplněn popis nastavení počáteční a konečné adresy SPI Flash paměti.
10.8.2016	Doplněn popis zapojení HCSxxx.
10.10.2016	Doplněn popis parametru /sn pro příkazový řádek a popis manuálních sériových čísel.
	Doplněn popis chybového kódu 9 vraceného na příkazovém řádku.
15.12.2016	Přidán popis použití v OS Linux.
2.2.2017	Doplněn popis automatického programování v okně Sériová výroba.
	Doplněn popis funkce Zachovat manuálně změněná data.
10.2.2017	Kapitola o instalaci ovladače v Linuxu byla doplněna.
13.3.2017	Doplněn odkaz na lin_ftd2xx.
23.3.2017	Přidán popis parametru /q1 pro příkazový řádek.
19.4.2017	Doplněn zmizelý obrázek zapojení CCxxxx.
9.5.2017	Doplněn popis funkce "Načíst poslední projekt při startu".

29.6.2017	Přidána kapitola o utilitě MultiUP.
30.10.2017	Doplněn popis zapojení AVR UPDI.
	Doplněn popis funkce "Při použití Windows Messages nezobrazovat ostatní varování".
	Doplněn popis funkce "Zapisovat checksum do logovacího souboru".
22.11.2017	Doplněn popis zapojení RL78.
19.2.2018	Přesunuta kapitola o HPRAVR.
28.3.2018	Doplněn popis funkce "Informace o součástce".
17.5.2018	Přidán popis zapojení RX600.
	Doplněn popis nastavení pro Renesas RX600.
7.6.2018	Přidán popis zapojení AT89C51CC01UA.
30.7.2018	Přidána poznámka k pojistce Debug MCU PIC.
	Přidána poznámka k mapování souborů pro AVR UPDI.
10.8.2018	Přidán popis, kde lze nalézt FTDI driver pro Linux.
	Přidána poznámka k parametrům pro příkazový řádek.
25.1.2019	Přidán popis funkce ukládání pojistek do ini.
	Přidán popis funkce Zapsat RC osc Adjustment.
31.1.2019	Přidán popis předávání hodnoty ProgressBaru ve quiet módu.
26.2.2019	Přidána poznámka pro DS28E05.
	Přidán popis funkce Přečíst adresu
	Přidán popis parametrů 24 a 25 zpráv Windows
23.4.2019	Přidán popis parametru /df, rozšířen popis parametrů /p, /o.
26.6.2019	Aktualizován popis parametru /s.
	Přidán popis funkce "Zamknout projekt".
	Přidán popis nastavení aktuálního sériového čísla z logovacího souboru.
26.7.2019	Přidán popis připojení HCS08.
20.9.2019	Změněna adresa firmy.
24.10.2019	Upravena poznámka u zapojení HCS08.
	Přidána kapitola o up_control.dll
18.12.2019	Doplněn popis zapojení pro AVR HVP.
10.1.2020	Přidán popis automatického načítání dalšího souboru.
31.3.2020	Přidán popis parametru /conf.
	Přidán seznam parametrů, které se předávají běžící aplikaci.
	Přidán popis programování pouze pozic v souboru.
	Přidán popis nastavení vypnutí kontroly Device ID.
26.5.2020	Přidán popis funkce "Načítat projekt odemčený".
	Přidán popis použití projektu na příkazovém řádku.
18.8.2020	Upraveno zapojení AVR UPDI.
11.1.2021	Aktualizace popisu instalace ovladačů.
	Byla popsána možnost vypnout interní pull-up na SDA.
	Aktualizace popisu instalace programu UP.
	JTAG Player, doplněn popis parametru -s
25.2.2021	Opraven popis Windows Message w=48.
	Přidán popis seznamu posledních použitých součástek.
23.12.2021	Byl doplněn obrázek a popis připojení rozhraní QUAD SPI.
7.1.2022	Byla přidána poznámka ohledně 34AA04.
	Upraveno nastavení práv v instalaci pro Linux.
14.1.2022	Doplněn popis komentářů v souboru sériových čísel.
23.5.2022	Přidán popis zapojení MCU LPC přes UART.
28.6.2022	Upraven popis projektů u příkazového řádku.
14.9.2022	Přidán popis zapojení MCU AVR s oddělenými piny UPDI a RESET.
12.1.2023	Přidán popis pojistky OSIS pro Renesas ARM.
	Upravena kapitola o CE a RoHS.
	Přidána poznámka ohledně verze Wine.
13.3.2023	Přidána poznámka k zobrazování checksumu algoritmu MD5.
	Přidán popis funkcí UP_GetChecksum a UP_GetChecksum_Wnd v up_dll.dll.

15.5.2023	Přidán popis zapojení CH32V003.
24.8.2023	Přidán popis funkce Fast mode pro CH32V003.
2.11.2023	Přidán popis funkce "Neptat se a neukládat změny do datového souboru"
6.11.2023	Přidána informace o příkladu čtení ProgressBaru.
6.12.2023	Přidána poznámka, že podpora pod Linuxem byla ukončena.
14.12.2023	Přidán popis zapojení SPD5118.
5.1.2024	Přidán popis zapojení pro Zilog Z8F.
16.2.2024	Obrázek zapojení Z8F byl opraven.
22.8.2024	Přidán návratový kód pro příkazový řádek při zamčené součástce.
20.2.2025	Přidán popis funkce "Přidat poznámku" pro projekt.
28.3.2025	Přidána tabulka ISP signálů pro LPCxxxx.
	V kapitole Popis propojení s aplikací přidán odkaz na kapitolu se specifikací konektorů.
13.5.2025	Do tabulky ISP signálů doplněny LPC5411x.