JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
PEDAGOGICKÁ FAKULTA

Studijní obor:

Měřící a výpočetní technika

BAKALÁŘSKÁ PRÁCE

KONSTRUKCE A REALIZACE PROGRAMÁTORU MIKROKRAŘADIČŮ A JEHO APLIKACE

Vedoucí bakalářské práce:
doc. PaedDr. Petr Adámek, Ph. D.

Vypracoval:
Ladislav Musel

Anotace:

Práce se zabývá konstrukcí a realizací programátoru mikrořadičů a jeho aplikací. Zařízení bude realizováno s pomocí mikrořadiče PIC18F2550. Součástí práce je detailní popis zařízení ze strany hardwaru i softwaru včetně dokumentace. Práce také objasňuje důvody vzniku tohoto konkrétního řešení a ukazuje postupně jeho realizaci.

Abstract:

The work deals with construction and realization of the programmer of the Microcontroller and its Application. The device will be realized with the PIC18F2550 microcontroller. One part of this work contains description of device's hardware and software including detailed documentation. The work also explains reasons of this specific solution and shows subsequently its realization.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Mlýnech dne 20. 4. 2009

Chtěl bych poděkovat vedoucímu mé bakalářské práce, panu doc. PaedDr. Petr Adámek, Ph. D., za přípravné konzultace a odborné vedení při vypracování této bakalářské práce na téma Konstrukce a realizace programátoru mikrořadičů a jeho aplikace.

Obsah

1 ÚVOD A CÍLE PRÁCE

Mikrořadiče se dnes široce používají všude tam, kde je potřeba řídit procesy různé úrovně a složitosti. Poskytují jednoduché a efektivní technické řešení, dříve realizovatelné s velkým počtem pasivních součástek a velkým počtem integrovaných obvodů. Které měli též velkou spotřebu [1].

Celá práce je rozdělena do několika kapitol. První kapitola je věnována principu, rozdělení a funkci mikrořadičů PIC®. Z důvodu obsáhlosti tématu je v určitých směrech zaměřena na nejběžnější typy, s kterými můžeme přijít do styku při realizaci projektů. Tato kapitola obsahuje ještě celý instrukční soubor 8-bitového mikrořadiče a nástin programovacího jazyka Assembler.

Do druhé části práce je zařazen návrh řešení a volba konstrukce programátoru mikrořadičů PIC®. Na kterou navazuje další kapitola s praktickou realizací programátoru. Byl zvolen programátor, který se připojuje k PC přes rozhraní USB. Pro jeho stavbu byl použit programátor komunikující přes rozhraní RS232. K ovládání programátoru je použit open source software.

Poslední kapitola je zaměřena na aplikaci programátoru a mikrořadiče na výukové úloze typu světelného hada z LED diod. Úloha je vypracována s podrobným návodem jak postupovat při její realizaci.

Hlavním cílem práce je návrh a praktická realizace programátoru mikrořadičů PIC®. Pro splnění hlavního cíle bylo nejprve se seznámit s funkcí a strukturou mikrořadičů. Dále navrhnout technické řešení programátoru. Pro ověření jeho funkce se jevilo jako nejvhodnější praktická ukázka jeho aplikace na konkrétní úloze, pro kterou byl vytvořen popis.

2 POPIS PRINCIPŮ MIKROŘADIČŮ PIC®

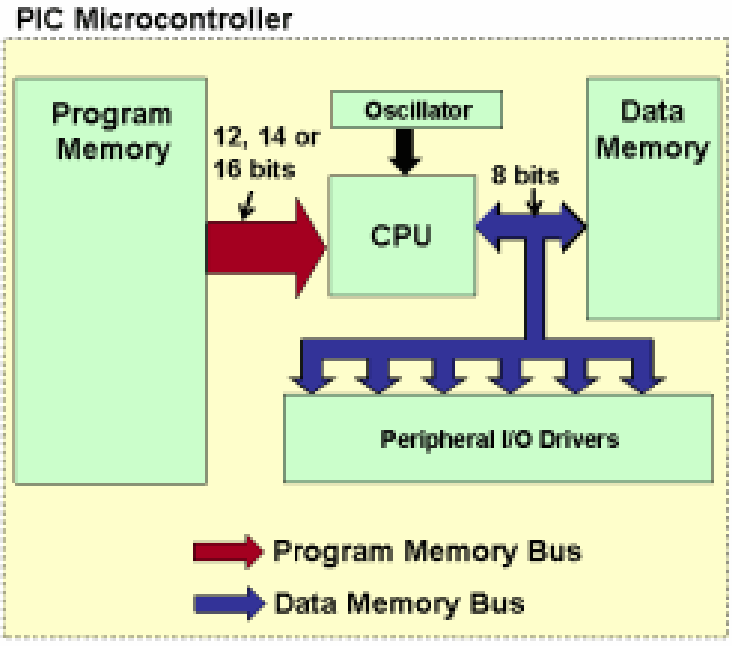
2.1 Co je mikrořadič PIC®

Mikrořadiče PIC® [1, 2] jsou programovatelné polovodičové součástky vyráběné firmou Microchip Technology Inc. sídlící v USA. Pro jejich označení se vžilo hned několik názvů např. mikrokontrolér (angl. microcontroller) nebo jednočipový mikropočítač. Vzhledově připomínají klasický integrovaný obvod, který však skrývá nepřeberné množství funkcí.

2.1.1 Harvardská architektura

Mikrořadiče pracují na principu Harvardské architektury [3] (obr. 1). Tato technologie byla poprvé použita u počítače Harvard Mark I, odtud byl také převzat její název. Hlavní výhodou této architektury je oddělení pamětí pro data a pro program (instrukce). Toto uspořádání paměti skýtá hned několik výhod. Například paměti programu a paměti dat můžou být úplně odlišné. Rozdíly mohou být v různé délce slova, časování, technologii a způsobu adresování.

Hlavní znakem mikrořadiče PIC® je, že pracuje s redukovanou instrukční sadou RISC. Což znamená, že mikropočítač je vybaven menším počtem instrukcí než u CISC. Instrukce nedostupné v redukované instrukční sadě je možné nahradit sekvencí dostupných instrukcí [1, 3, 4].



Obr. 1 Architektura 8-bitového mikrořadiče PIC® [2]

2.1.2 Přehled mikrořadičů PIC® [1, 2, 3, 4]

Výrobce rozděluje mikrořadiče PIC® do tří skupin 8-bitové, 16-bitové a 32-bitové, které lze dále rozdělit dle typového označení. Pro představu jsou zde nastíněny charakteristiky jednotlivých typových řad. Jak šel časový vývoj, tak také mikrořadiče doznaly mnoho změn. Hlavně co se týká použití typu pamětí. Dnes se nejvíce používají, pro jejich snadnou programovatelnost, paměti typu FLASH a EEPROM.

8-bitové mikrořadiče PIC®

- **PIC10:**
  - Nejjednodušší řada.
  - Čtyři I/O.
  - Pouzdro s šesti vývody.
  - Některé typy mají A/D převodník nebo komparátor.
  - Používají paměť typu FLASH.
- **PIC12:**
  - Obsahují paměť FLASH a EEPROM.
  - Interní oscilátor a časovač.
  - Šest I/O.
  - Pouzdro s osmi vývody.
- **PIC16:**
  - Obsahují paměť FLASH a EEPROM.
  - A/D převodník, komparátor.
  - Podporují komunikační rozhraní UART ,SPI a I^2C.
  - Až 368 paměti RAM.
  - Pouzdro s 14 až 64 vývody.
- **PIC18:**
  - Nástupce zaniklé řady PIC17.
  - Podporují kolem 70 instrukcí.
  - FLASH a EEPROM.
  - Podporují komunikační rozhraní USB 2.0, ETHERNET, CAN.
  - Některé obsahují bootloader.
  - Taktovací kmitočet až 64 MHz.

16 bitové mikrořadiče PIC®

- **PIC24F:**
  - Maximální rychlost 16 MIPS.
  - Pouzdro s 24 až 100 vývody.
  - Napájecí napětí 3 - 3,6 V.
  - USB 2.0, Ethernet 10/100.
- **PIC24H:**
  - Maximální rychlost až 40 MIPS.
  - Napájecí napětí 3 - 3,6 V.
  - Až 16 kB paměti RAM.
  - Přímý přístup do paměti (DMA).
  - Pouzdro s 18 až 100 vývody.
- **dsPIC30F:**
  - Digitální signálové kontroléry.
  - Větší rozsah napájecího napětí 2,5 - 5 V.
  - Maximální rychlost až 40 MIPS.
  - Pouzdro s 18 až 80 vývody.
- **dsPIC33F:**
  - Digitální signálové kontroléry.
  - Maximální rychlost až 40 MIPS.
  - Napájecí napětí 3 - 3,6 V.
  - Až osm vyhrazených kanálů.
  - Pouzdro s 18 až 100 vývody.

32-bitové mikrořadiče PIC®

- **PIC32MX:**
  - Novinka představená koncem roku 2007.
  - Maximální rychlost 80 MIPS.
  - 32 až 512 kB program paměti FLASH.
  - 8 až 32 kB paměti RAM.
  - Až 16 desetibitovými A/D převodník.
  - Dva vyhrazené DMA kanály, USB 2.0.
  - Pouzdro s 18 až 100 vývody.

2.2 Popis Mikrořadiče PIC16F628A [1, 5]

PIC16F628A a PIC16F6xx (tab. 1) patří do skupiny 8-bitových mikrořadičů. Jsou založené na instrukční sadě RISC, která obsahuje 35 jednoduchých instrukcí. Všechny instrukce se vykonají během jediného cyklu, s výjimkou pro instrukce větvení programu, které probíhají ve dvou cyklech. Taktovací frekvenci je možno nastavit pomocí externího oscilátoru a to na hodnotu od 0 – 20MHz. Mikrořadič disponuje dvěma interními oscilátory 4MHz s přesností ±1% a nízkonapěťovým 48 kHz. Dále je možné například krystal nebo RC člen připojit jako externí oscilátor.

Pro adresaci slouží přímé, nepřímé a relativní adresování. V mikrořadiči je zabudován osmi úrovňový zásobník, který se používá při volání instrukce CALL nebo přerušení.

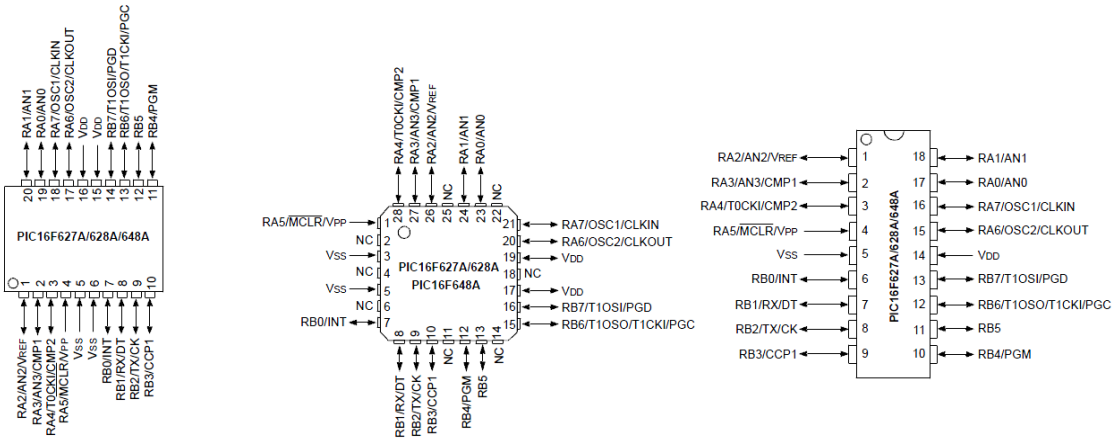
Pro uchování dat je mikrořadič vybaven pamětí Flash a EERPOM (obr. 2). Pro zajímavost u paměti EEPROM je možnost až miliónu zápisů při zachování dat nad 40 let.

Rozsah pracovního napětí se pohybuje od 2,0 - 5,5 V. Odebíraný proud se odvíjí od použitého zařízení (oscilátor, Timer1, Watchdog atd.) a stavu v kterém pracuje (Sleep). Řádově se jedná o µA a v případě vyšších frekvencí nad 1MHz stoupá odebíraný proud do stovek µA. Odběr se ve stavu Sleep se pohybuje okolo 100 nA při napájecím napětí 2 V.

		PIC16F627A	PIC16F628A	PIC16F648A	PIC16LF627A	PIC16LF628A	PIC16LF648A
Clock	Maximum Frequency of Operation (MHz)	20	20	20	20	20	20
Memory	Flash Program Memory (words)	1024	2048	4096	1024	2048	4096
	RAM Data Memory (bytes)	224	224	256	224	224	256
	EEPROM Data Memory (bytes)	128	128	256	128	128	256
Peripherals	Timer module(s)	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2
	Comparator(s)	2	2	2	2	2	2
	Capture/Compare/ PWM modules	1	1	1	1	1	1
	Serial Communications	USART	USART	USART	USART	USART	USART
	Internal Voltage Reference	Yes	Yes	Yes	Yes	Yes	Yes
Features	Interrupt Sources	10	10	10	10	10	10
	I/O Pins	16	16	16	16	16	16
	Voltage Range (Volts)	3.0-5.5	3.0-5.5	3.0-5.5	2.0-5.5	2.0-5.5	2.0-5.5
	Brown-out Reset	Yes	Yes	Yes	Yes	Yes	Yes
	Packages	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN

Tab. 1 Přehled základních parametrů mikrořadičů řady16F6xx [2]

Mikrořadič PIC16F628A je konstruován ve třech typech pouzder (obr. 2) SSOP (18 pin), QFN (28 pin) a PDIP (18 pin).



Obr. 2 Pouzdra mikrořadiče PIC16F628A (zleva) SSOP, QFN a PDIP [1]

Speciální vlastnosti mikrořadiče:

- Reset při zapnutí napájecího napětí (POR).
- Detekční obvod výpadku napájecího napětí (BOR).
- Časovač pro zpoždění resetu.
- Konfigurovatelný vývod MCLR.
- Individuálně konfigurovatelné odpory Pull-up.
- Časovač zapnutí oscilátoru (OST).
- Watchdog (WDT).
- Programovatelná ochrana kódu.
- Sériové programování v zapojení (ICSP) pomocí dvou vývodů.
- Nízký odběr a vysoká rychlost CMOS FLASH technologie.

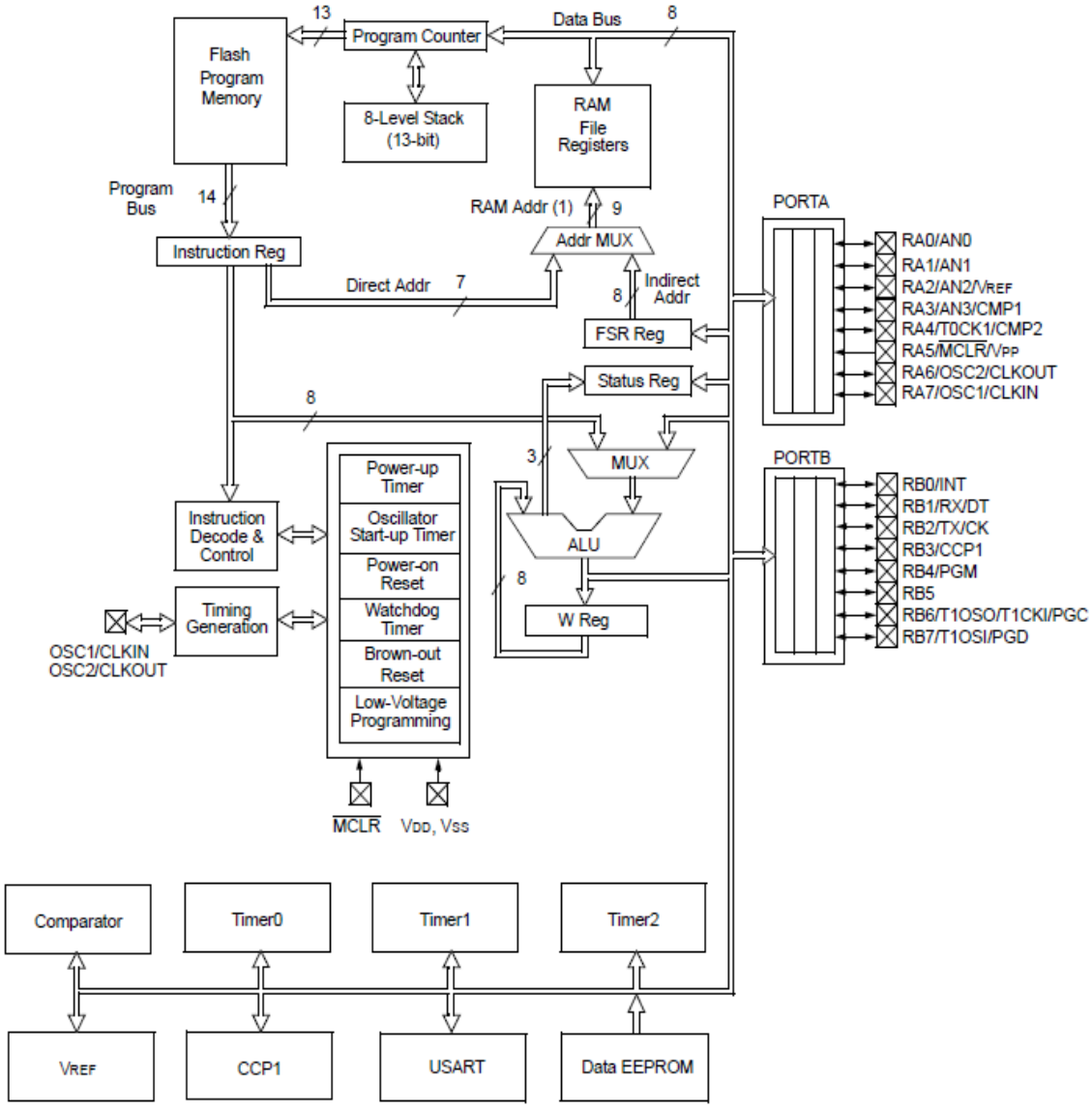
Možnosti periférií mikrořadiče:

- 16 vývodů s možností individuální konfigurace.
- Dva analogové komparátory.
- Vstupy jsou přepínatelné na různé vývody mikropočítače.
- Výstup komparátoru je přístupný na vývodu mikropočítače.
- 8-bitový čítač/časovač TMR0.
- Programovatelná předdělička.

- 16-bitový čítač/časovač TMR1 s předděličkou.
- Timer1 použít pomaloběžný oscilátor LP zapojený.
- Univerzální synchroní/asynchroní příjmač/vysílač USART/SCI.

2.2.1 Aritmeticko-logická jednotka ALU (Arithmetic/Logic Unit)

V blokovém schématu mikrořadiče (obr. 3) jsou popsány nejdůležitější části, ALU je jednou z nich. Vykonává všechny příkazy a matematické operace. Pracuje s daty umístněnými v registru W a daty umístněnými v libovolném registru pole registrů. Podle některých operací nastavuje v registru SWR příznakové bity C, DC a Z [4, 6].



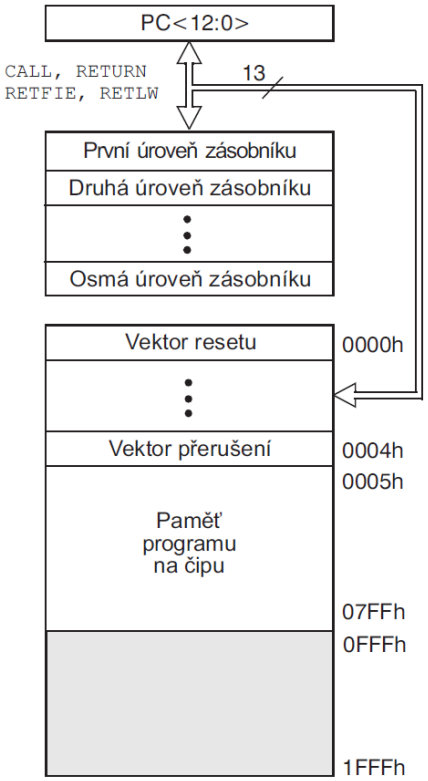
Obr. 3 Blokové schéma mikrořadiče PIC16F628A [1]

2.2.2 Pracovní registr W (Working register)

Jak název sám napovídá, jedná se o registr, který plní roli mezi-paměti, přes kterou se provádí většina operací. Hodnoty mířící na výstupy procházejí nejdříve vždy W registrem a až potom míří na výstupní registr. Dále uchovává druhý operand .při provádění dvouoprandových instrukcí [1, 4, 6].

2.2.3 Paměť programu (Program Memory)

Paměť programu [1, 4, 7,] (obr. 4) slouží pro uložení vlastního strojového kódu (programu). Popisovaný mikrořadič využívá paměť typu FLASH, kterou je možno vícenásobně programovat. Kapacita paměti je 8K x 14bitů, při čemž fyzicky implementovaných je prvních 2K v rozsah 0000h až 07FFh. Při adresaci nad fyzicky implementovanou paměť dojde k přetečení PC a návrat na 0000h.



Obr. 4 Uspořádání paměti a zásobník STACK [5]

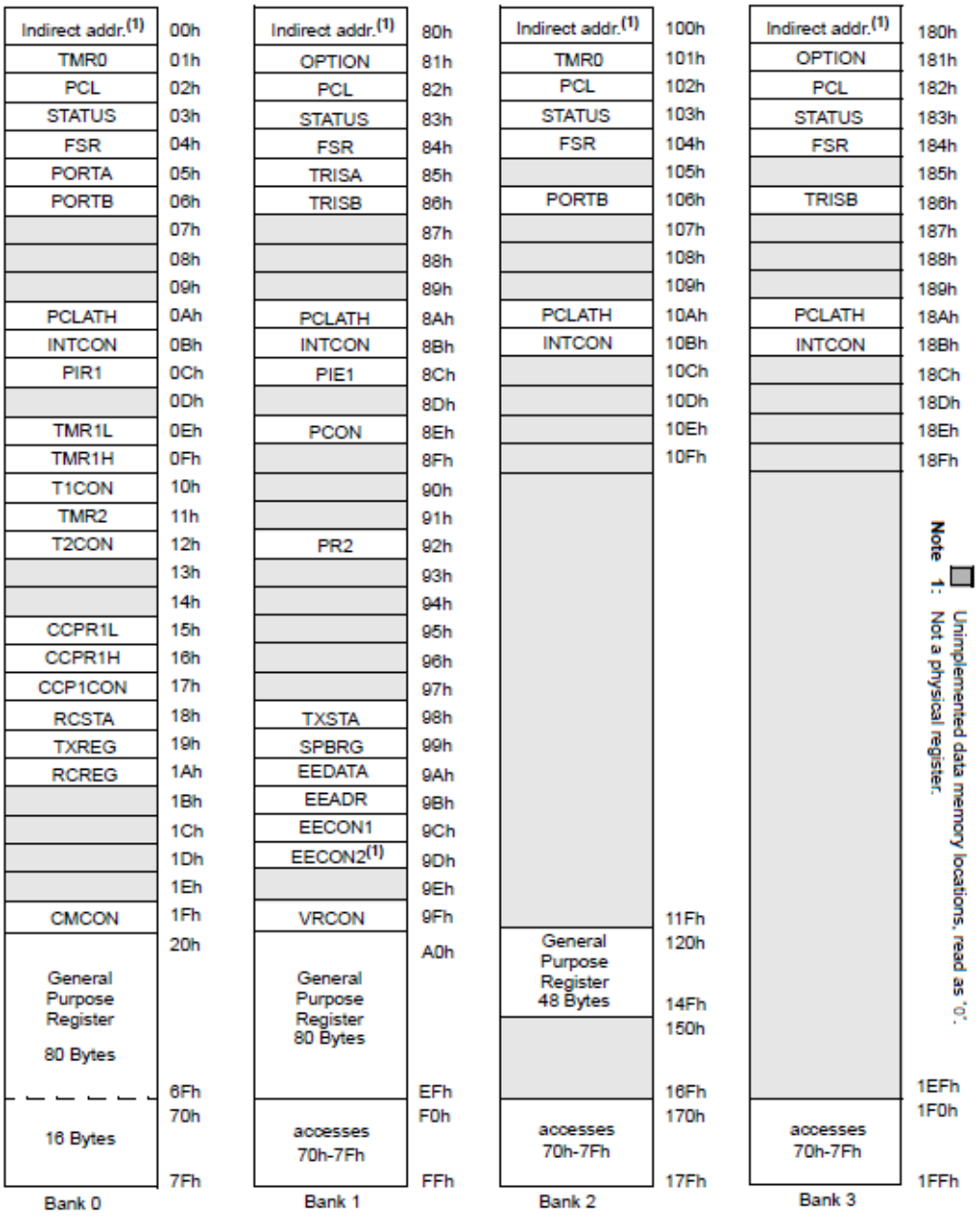
Třinácti-bitový **Programový čítač PC** (PROGRAM COUNTER) odkazuje na adresy v paměti programu, ze které se postupně načítají a vykonávají instrukce. Po resetu začíná PC na adrese 0000h označovanou jako vektor pro RESET. Při vyvolání přerušení začíná PC na adrese 0004h [1].

Vyvoláním přerušení nebo voláním instrukce CALL dochází k uchování návratové adresy, aby bylo možno pokračovat ve vykonávání programu. Návratovou adresu si PC ukládá do zásobníku (STACK). Jde o hardwarový osmi úrovňový 13-bitový zásobník pracující na principu kruhového registru. Po naplnění všech osmi úrovní zásobníku dochází k přetečení, které nelze zjistit pomocí žádného příznakového bitu. Tudíž devátá návratová adresa přepíše adresu na prvním místě v zásobníku, dále pak desátá přepíše druhé místo atd. Pro přístup

k zásobníku slouží návratové instrukce RETURN, RETLW a RETFIE. Do zásobníku lze ukládat pouze návratové adresy [1, 4].

2.2.4 Paměť dat

Paměť dat (obr. 5) je rozdělena na dvě oblasti registrů. První obsahuje **univerzální** registry sloužící jako paměť RAM, někdy také označována jako paměť pro všeobecné použití. Slouží pro ukládání libovolných dat, kterými mohou být výsledky matematických operací nebo námi nadefinované hodnoty. Velikost paměti RAM je 224 bajtů.



Obr. 5 Uspořádání paměti dat 16F628A [1]

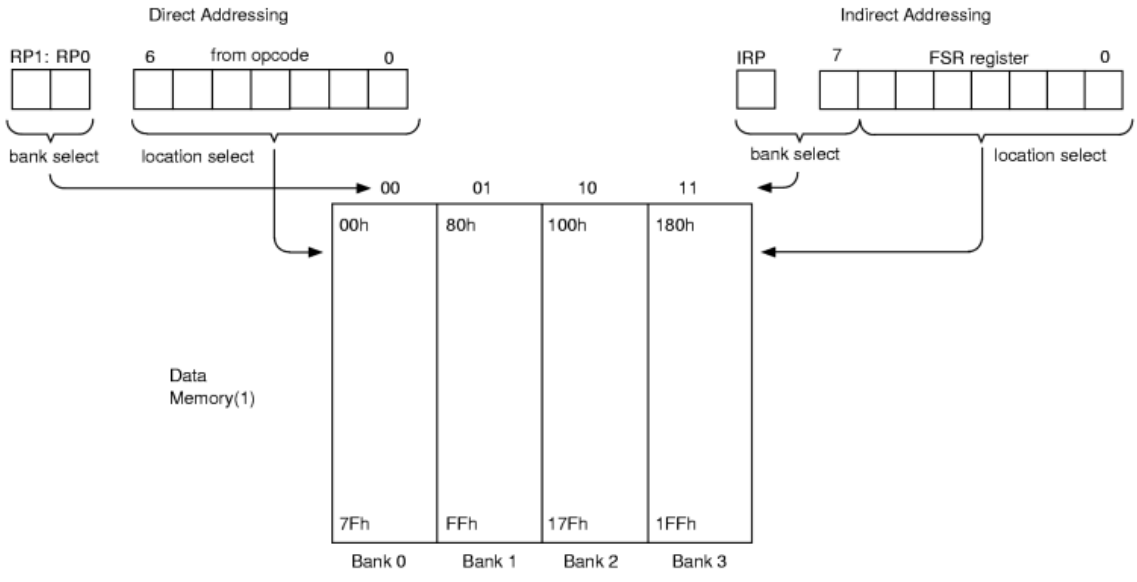
Druhá část paměti dat obsahuje speciální funkční registry (SFR), které umožňují přístup a nastavení jednotlivých periferií mikrořadiče. Na rozdíl od universálních registrů mají striktně definované funkce a nelze je měnit. Tyto registry jsou tvořeny jako statická paměť RAM [4, 6, 8].

V předchozím odstavci bylo uvedeno rozdělení paměti dat dle oblasti registrů, to ovšem není jediné možné rozdělení. Paměť dat se dále rozděluje fyzicky a to na čtyři Banky (stránky) po 128 bajtech. Některé ze speciálních registrů jsou zrcadleny ve více bankách (např STATUS, PCL) [8].

2.2.5 Speciální funkční registry (SFR)

2.2.5.1 Nepřímé adresování dat, registry INDF a FSR [1, 4, 5]

Nejdříve si vysvětlíme dva způsoby adresovaní registrů (obr. 6) datové paměti. Prvním způsobem je **přímé adresování,** kdy instrukce přímo uvádí odkud nebo kam se mají data přenést. Pro adresaci datové paměti je za potřebí devíti bitová adresa. U přímého adresování se skládá se ze 7 bitové adresy určené v instrukci a dvou bitů RP0,RP1 registru STATUS. Bity RP0 a RP1 nastavují požadovanou banku (stránku).



Obr. 6 Způsoby adresování registrů [1]

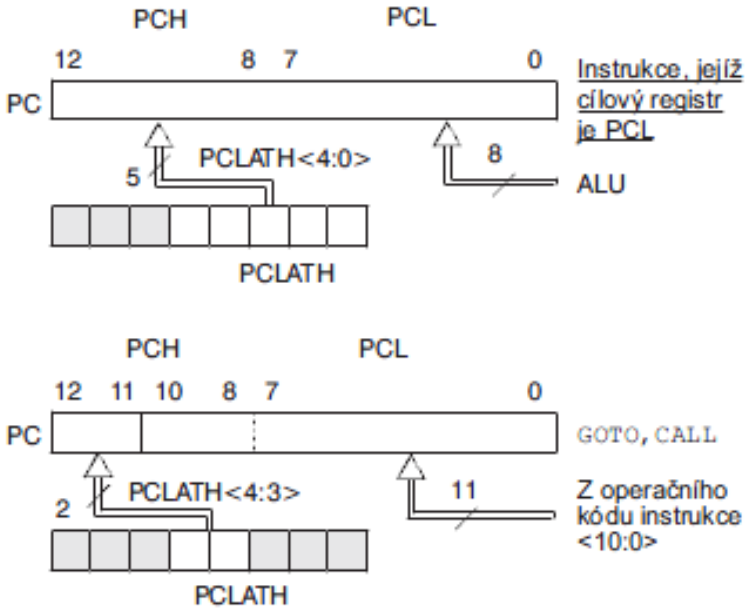
Nepřímé adresování pracuje tak, že se nejdříve přistoupí na adresu registru a až zde se nachází požadovaná adresa. K tomu to slouží funkční registr FSR a INDF. Registr INDF není fyzicky implementován, ale přistupuje se

k němu stejně jako k jakémukoliv jinému registru. Při zapisování nebo čtení, s ním pracujeme, jako s registrem jehož adresa je uložena v registru FSR. U nepřímého adresování se devíti bitová adresa skládá z 8 bitů registru FSR a sedmého bitu IPR z registru STATUS. Bit IPR a nejvyšší bit FSR určují výběr banky (stránky).

2.2.5.2 PCL a PCLACHT registry

Registry PCL a PCLACHT slouží pro přístup do programového čítače (PC), který má šířku 13 bitů. Registr PCL je nižší bajt, z něhož lze číst i do něj zapisovat. Vyšší bajt PCH nelze číst vůbec, zapisovat do něj lze nepřímo pomocí PCLATH.

Na obrázku (obr. 7) jsou uvedeny dvě možnosti přesunu dat v PC. V horní ukázce jde o přesun dat při zápisu do PCL. Zde dochází k automatickému přesunu dat z PCLATH do horní části PC. Spodní část obrázku popisuje situaci při vykonání instrukce GOTO nebo CALL. Při resetu jakéhokoli druhu dojde k vynulování celého programového čítače [1, 4, 5].

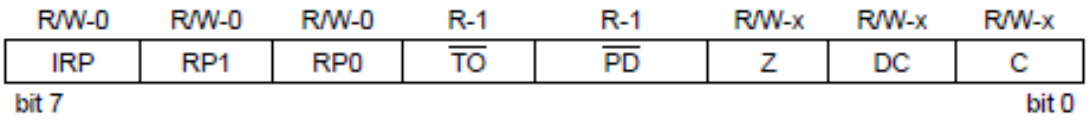


Obr. 7 PCL a PCLACHT registry [5]

Každý zápis do PCL způsobí přepsání horních bitů PC ze záchytného registru PCLATH. Je nezbytné před každým skokem nastavit do PCLATH adresu požadované stránky paměti programu do které chceme skočit [4].

2.2.5.3 STATUS registr

Tento registr (obr. 8) obsahuje stavy ALU, příznaky událostí a bity pro nastavení adresy banky paměti dat. R = bit pro čtení, W = bit pro zápis, x = neznámá hodnota bitu.



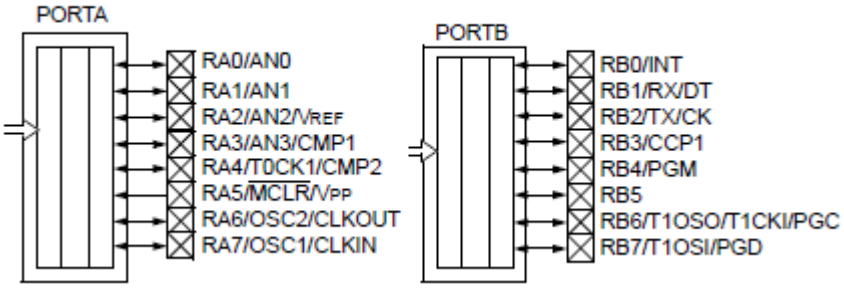
Obr. 8 Registr STATUS [1]

Bit 7 IRP se uplatňuje při výběru banky registrů RAM (pouze při nepřímém adresování). V logické nule je vybrána banky 0, 1 a při logické jedničce banky 2, 3. **Bit 6 RP1 a bit 5 RP0** se naopak používá pouze pro výběr při přímém adresování registrů RAM. Výběr banky 0 provedeme nastavením obou bitů do nuly. Pro výběr banky 1 tedy platí nastavení RP1 = 0, RP0 = 1. Dále analogicky banka 2 RP1 = 1, RP0 = 0 a banka 3 RP1 = 1, RP0 = 1. **Bit 3 $\overline{\text{TO}}$ a bit 4 $\overline{\text{PD}}$** jsou určeny pouze pro čtení a po zapsání hodnoty do registru STATUS se jejich obsah nemění do doby příchodu další události. **Bit 4 $\overline{\text{TO}}$** je příznak přetečení časovače WDT (Watchdog). Tento bit se nastavuje na logickou nulu po resetu způsobeným přetečením WDT. Logická jednička se nastaví po zapnutí napájení vykonáním instrukce CLRWDT nebo SLEEP. **Bit 3 $\overline{\text{PD}}$** je příznak úsporného režimu SLEEP. Logická nula se nastavuje po vykonání instrukce SLEEP. Jednička označuje zapnutí napájení nebo vykonání instrukce CLRWDT. **Bit 2 Z** (ZERO) jedná se o příznak nuly zaznamenávající výsledek aritmetické nebo logické operace. Je-li výsledek nula je nastaven bit do logické jedničky a naopak. **Bit 1 DC** (DIGIT CARRY/BOROW) je plněn výpůjčkou při odčítání a přenosem při sčítání. Jedná se ovšem o výpůjčku a přenos z nižších 4 bitů do vyšších. Využívá se především v operacích s desítkovými čísly. **Bit 0 C** je příznak přenosu při sčítání a výpůjčky při operacích odčítání. Další využití nachází při rotaci bitu s instrukcí RRF a RLF [1, 4, 5, 6].

2.2.5.4 PORTA, PORTB registry

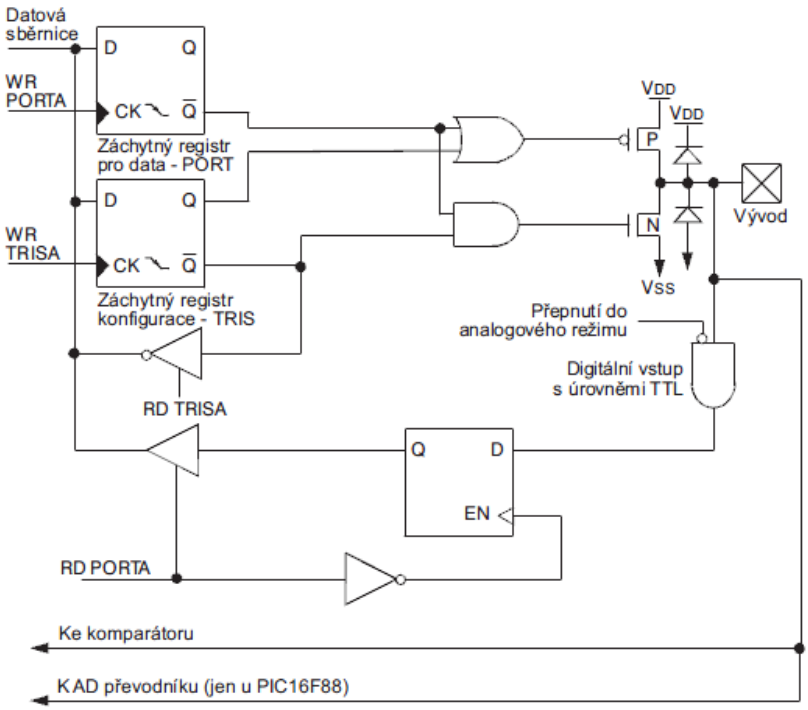
PORTA a PORTB jsou obousměrné 8-bitové porty, které se mohou nastavit jako digitálně vstupně výstupní vývody, ale mohou sloužit také pro činnost některé z periferií. Z obrázku (obr.9) je patrné, že nastavení jednotlivých

pinů je nepřeberné. A skoro každý má své vlastní hardwarové zapojení. Pro příklad je na obrázku (obr. 10) uvedeno blokové schéma vývodů RA0/AN0. Pro podrobné vnitřní zapojení portů je třeba vždy vyhledat datasheet daného mikrořadiče [4].



Obr.9 PORTA, PORTB registry [1]

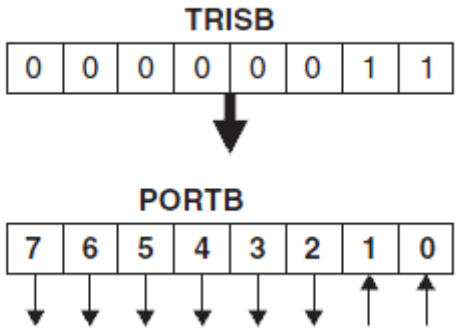
Registry PORTA a PORTB jsou záchytné pracovní registry portů a každý obsahují 8 funkčních bitů. Pokud chceme zapsat nějaké hodnoty na výstup, zapíšeme je do těchto funkčních bitů. Hodnoty zde zůstanou do doby jejich dalšího přepsání.



Obr. 10 Vnitřní blokové schéma vývodu RA0/AN0 [5]

2.2.5.5 TRISA, TRISB registry

Pomocí registru TRISA nastavujeme, zda bude daný pin vstup nebo výstup portu A, analogicky to platí pro TRISB a port B, příklad je uveden na obrázku (obr.11). Logická nula na bitu nastaví port na výstupní funkci a naopak logická jednička na vstupní funkci. Pokud některý z vývodů nastavíme jako výstupní, má z počátku hodnotu logické jedničky. To lze změnit zápisem do registrů PORTA a PORTB. Vývod nastavený jako vstupní se chová jako by se odpojil. Důvodem této funkce je jeho vysoká impedance [1, 4, 9].



Obr. 11 Nastavení PORTUB [3]

2.2.5.6 OPTION registr

OPTION registr (obr. 12) obsahuje řídící bity pro ovládání a konfiguraci programovatelné předděličky, vnějšího přerušovacího vstupu, čítače/časovače Timer0 a pull-up rezistorů brány PORTB. Všechny bity lze číst a rovněž všechny bity je možné libovolně nastavit [1, 4].

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RAPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Obr. 12 OPTION registr [1]

2.2.5.7 Ostatní speciální funkční registry

ITCON registr obsahuje bity pro práci s přerušovacím systémem mikropočítače. **PIE1** registr obsahuje bity pro individuální povolení jednotlivých druhů přerušení od periferií. **PIR1** registr obsahuje příznakové bity pro

přerušení od periferií. **PIE2** registr obsahuje bity pro individuální povolení přerušení od modulu oscilátoru, dokončení zápisu do EEPROM a komparátoru. **PIR2** registr obsahuje příznakové bity pro přerušení od modulu oscilátoru, dokončení zápisu do EEPROM a komparátoru.

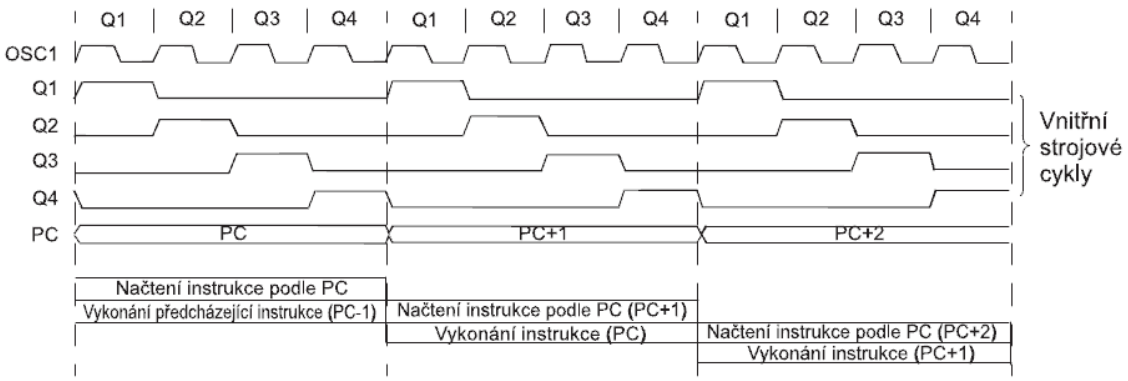
PCON jsou příznaky přerušení a nastaví se vždy, když jsou splněny podmínky, které mohou vyvolat přerušení. To nastane tehdy, pokud žádost o přerušení není přijata, např. přerušení nejsou povolena nebo jsou přerušení maskována. Uživatel musí zajistit programově jejich nulování před povolením přerušení [1, 4, 5].

2.3 Instrukční cyklus

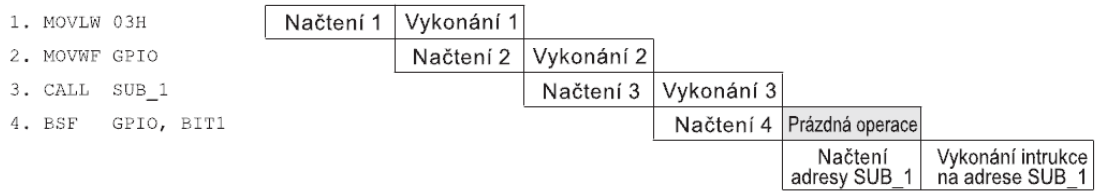
„Taktovací frekvence mikropočítače je interně vydělena čtyřmi. Tím vzniknou čtyři vzájemně fázově posunuté strojové cykly označované jako takt Q1, Q2, Q3, Q4. Tyto čtyři takty tvoří instrukční cyklus. V taktu Q1 dochází k inkrementaci programového čítače PC a během taktu Q4 dochází k uložení operačního kódu následující instrukce do záchytného registru instrukce. Tato instrukce je vykonána během následujících čtyř taktů Q1 – Q4“ (obr.13). [8]

Instrukční cyklus [4, 8] se skládá ze 4 taktů (Q1, Q2, Q3, Q4). Provádění instrukcí (obr. 14) je následovné. Zatímco je jedna instrukce dekódována a vykonávána, je současně vyjmuta z paměti a umístněna do instrukčního cyklu.

Pokud dochází k programovému skoku, tedy změně obsahu PC, nelze vykonat již načtenou instrukci. Dochází k novému načtení operačního kódu dle obsahu PC, již načtená instrukce je vykonána jako NOP a doba dokončení celé instrukce skoku jsou dva instrukční cykly.



Obr. 13 Instrukční cyklus a frekvence oscilátoru [5]



Obr. 14 Průběh zpracování instrukcí [5]

2.4 Instrukční soubor mikrořadiče PIC16F628A

Instrukční soubor [4, 6, 8] je sada všech instrukcí, které mikrořadič zná a umí je vykonat. Každá instrukce je aplikovatelná na kterékoliv paměťové místo. Většina instrukcí je vykonána v průběhu jednoho instrukčního cyklu. Dva cykly provádějí pouze instrukce, které souvisí s větvením programu nebo v případech kdy dochází k zápisu do čítače registrů PC.

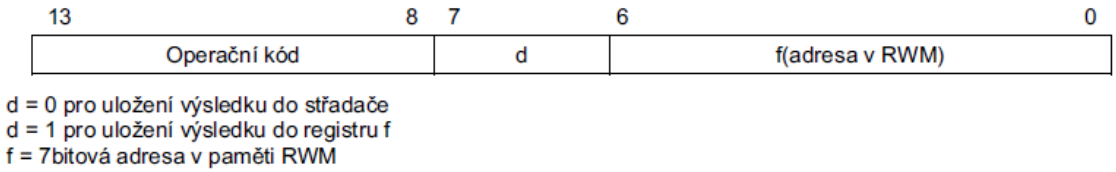
Počet instrukcí se liší dle typu mikrořadiče. Zvolený mikrořadič PIC16F628A obsahuje 35 jednoslovných instrukcí. Ty jsou dále rozděleny logicky do tří základních skupin bajtově orientované instrukce, bitově orientované instrukce a řídící instrukce.

Pro popis instrukce se využívají tyto parametry a označení:

- **Instrukce** - jedná se syntaxi pro psaní instrukce v Assembleru. Nejčastěji se skládá z několika písmen, které naznačují funkci instrukce.
- ***Instrukční kód*** - zápis instrukce ve strojovém kódu, který bychom použili pouze v případě programování bez assembleru. Délka toho to kódu je 14 bitů.
- ***Počet cyklů*** - informace za kolik cyklů se instrukce vykoná.
- ***Operace*** - zjednodušený zápis funkce pro rychlou orientaci.
- ***Stavové bity*** - ovlivňované příznaky v registru stavového slova.
- ***Interval operandu*** - udává možný rozsah konstanty, adresy bitu a bajtu.

2.4.1 Bajtové orientované instrukce [1, 4]

Bajtově orientované (obr. 15) instrukce používají pro označení adresy písmeno f, které určuje registr, s kterým se pracuje. Jedná se o sedmi-bitovou adresu v paměti RAM. Parametr d ovlivňuje místo uložení výsledku operace. Když je d rovno logické nule, výsledek se ukládá zpět do pracovního registru W. Při d rovnající se logické jedničce putuje výsledek na definovanou adresu f.



Obr. 15 Bajtové orientované instrukce [1]

Instrukce :	**ADDWF f,d**
Instrukční kód :	00 0111 dfff ffff
Počet cyklů :	1
Operace :	(W) + (f) → (d)
Stavové bity :	C, DC, Z
Interval operandu :	$0 \leq f \leq 127$, $d \in [0, 1]$

Instrukce ADDWF aritmeticky sečte data pracovního registru W s daty registru f. Parametr d určuje místo uložení výsledku. Když je hodnota d rovna logické nule, výsledek se zapíše zpět do registru W. Je-li hodnota d rovna logické jedničce , výsledek je uložen do registru f.

Instrukce :	**ANDWF f,d**
Instrukční kód :	00 0101 dfff ffff
Počet cyklů :	1
Operace :	(W) .AND.(f) → (d)
Stavové bity :	Z
Interval operandu :	$0 \leq f \leq 127$, $d \in [0, 1]$

Instrukce ANDWF provede logický součin dat pracovního registru W s daty registru f. Parametr d určuje místo uložení výsledku. Když je hodnota d rovna logické nule, výsledek se zapíše zpět do registru W. Je-li hodnota d

rovna logické jedničce, výsledek je uložen do registru f.

Instrukce : **CLRF f**
Instrukční kód : 00 0001 1fff ffff
Počet cyklů : 1
Operace : 00h → (f), 1 → Z
Stavové bity : Z
Interval operandu : 0 ≤ f ≤ 127

Vynuluje se obsah registru f a bit Z se nastaví do logické jedničky.

Instrukce : **COMF f,d**
Instrukční kód : 00 1001 dfff ffff
Počet cyklů : 1
Operace : $(\bar{f}) \rightarrow$ (d)
Stavové bity : Z
Interval operandu : 0 ≤ f ≤ 127, d ∈ [0, 1]

Instrukce provede komplement (negace) obsahu registru ‘f‘. To znamená, že dojde k záměně logických nul za jedničky a jedniček za nuly. Je-li d rovno logické nule, výsledek se uloží do registru W, při d rovno logické jedničce výsledek bude uložen do registru f.

Instrukce : **DECF f,d**
Instrukční kód : 00 0011 dfff ffff
Počet cyklů : 1
Operace : (f) - 1 → (d)
Stavové bity : Z
Interval operandu : 0 ≤ f ≤ 127, d ∈ [0, 1]

Od obsahu registru f je odečtena logická jednička. Je-li d rovno logické nule, výsledek se uloží do registru W, při d rovno logické jedničce bude výsledek uložen do registru f.

Instrukce : **DECFSZ f,d**
Instrukční kód : 00 1011 dfff ffff

Počet cyklů : 1 - 2

Operace : (f) - 1 → (d), skok jeli výsledek „0“

Interval operandu : $0 \leq f \leq 127$, $d \in [0,1]$

Od obsahu registru f je odečtena logická jednička. Když není výsledek roven logické nule, proběhne následující instrukce. Pokud odečtením získáme logickou nulu, přeskočí se následující instrukce a provede se místo ní instrukce NOP. Tento druhý případ zabere dva instrukční cykly. Dále je-li hodnota d rovna logické nule, výsledek se uloží do registru W, při d rovno logické jedničce bude výsledek uložen do registru f.

Instrukce : **INCF f,d**

Instrukční kód : 00 1010 dfff ffff

Počet cyklů : 1

Operace : (f) + 1 → (d)

Stavové bity : Z

Interval operandu : $0 \leq f \leq 127$, $d \in [0, 1]$

K obsahu registru f je přičtena logická jednička. Je-li hodnota d rovna logické nule, výsledek se uloží do registru W, při d rovno logické jedničce bude výsledek uložen do registru f.

Instrukce : **INCFSZ f,d**

Instrukční kód : 00 1111 dfff ffff

Počet cyklů : 1 - 2

Operace : (f) + 1 → (d), skok jeli výsledek „0“

Interval operandu : $0 \leq f \leq 127$, $d \in [0, 1]$

K obsahu registru f je přičtena logická jednička. Když není výsledek roven logické nule, proběhne následující instrukce. Pokud přičtením získáme nulu, přeskočí se následující instrukce a provede se místo ní instrukce NOP. Doba pro vykonání instrukce zabere dva instrukční cykly. Dále je-li hodnota d rovna logické nule, výsledek se uloží do registru W, při d rovno logické jedničce bude výsledek uložen do registru f.

Instrukce : **IORWF f,d**

Instrukční kód : 00 0100 dfff ffff

Počet cyklů : 1

Operace : (W).OR.(f) → (d)

Stavové bity : Z

Interval operandu : $0 \leq f \leq 127$, $d \in [0, 1]$

Instrukce provede logický součin mezi pracovním registrem W a registrem f. Je-li hodnota d rovna logické nule výsledek se uloží do pracovního registru W, při d rovno logické jedničce je výsledek uložen do registru f.

Instrukce : **MOVF f,d**

Instrukční kód : 00 1000 dfff ffff

Počet cyklů : 1

Operace : (f) → (d)

Stavové bity : Z

Interval operandu : $0 \leq f \leq 127$, $d \in [0, 1]$

Obsah registru f je přesunut podle hodnoty d. Rovná-li se d logické nule, výsledek se přesune do registru W. Pokud se d rovná logické jedničce, je uložen zpět do registru f a zároveň je ovlivněn bit Z ze stavového registru.

Instrukce : **MOVWF f**

Instrukční kód : 00 0000 1fff ffff

Počet cyklů : 1

Operace : (W) → (f)

Interval operandu : $0 \leq f \leq 127$

Obsah registru W je přesunut do registru f.

Instrukce : **NOP**

Instrukční kód : 00 0000 0xx0 0000

Počet cyklů : 1

Operace : žádná operace

Interval operandu : $0 \leq k \geq 127$

Prázdná operace při které se nic neprovede, pouze proběhne jeden strojní cyklus.

Instrukce :	**RLF f,d**
Instrukční kód :	00 1101 dfff ffff
Počet cyklů :	1
Operace :	levá rotace bitů registru **f**
Stavové bity :	C
Interval operandu :	$0 \leq f \leq 127$, $d \in [0, 1]$

K obsahu registru f je připsán zleva bit C stavového registru. Po-té je zprava z registru f přesunut bit do C. Dochází takzvané rotaci bitů. Dále platí je-li d rovno logické nule, výsledek se uloží do registru W, při d rovno logické jedničce bude výsledek uložen do registru f.

Instrukce :	**RRF f,d**
Instrukční kód :	00 1100 dfff ffff
Počet cyklů :	1
Operace :	pravá rotace bitů registru **f**
Stavové bity :	C
Interval operandu :	$0 \leq f \leq 127$; $d \in [0, 1]$

K obsahu registru f je připsán zprava bit C stavového registru. Po-té je zleva z registru f přesunut bit do C. Dochází takzvané rotaci bitů. Dále platí je-li d rovno logické nule výsledek se uloží do registru W, při d rovno logické jedničce bude výsledek uložen do registru f

Instrukce:	**SUBWF f,d**
Instrukční kód:	00 0010 dfff ffff
Počet cyklů:	1
Operace:	(f) - (W) $\rightarrow$ (d)
Stavové bity:	C, DC, Z
Interval operandu:	$0 \leq f \leq 127$; $d \in [0, 1]$

Instrukce aritmeticky odečte data pracovního registru W s od dat registru f. Pokud je hodnota d rovna nule, výsledek se zapíše zpět do registru W. Je-li hodnota d rovna jedné, výsledek je uložen do registru f.

Instrukce : **SWAPF f,d**

Instrukční kód : 00 1110 dfff ffff

Počet cyklů : 1

Operace : (f<3:0>) → (d <7:4>), (d<7:4>) → (f<3:0>)

Interval operandu : 0 ≤ f ≤ 127

Vzájemné prohození čtyř nižších a čtyř vyšších bitů registru f. Pokud je hodnota d rovna nule, výsledek se zapíše zpět do registru W. Je-li hodnota d rovna jedné, výsledek je uložen do registru f.

Instrukce : **XORWF f,d**

Instrukční kód : 00 0110 dfff ffff

Počet cyklů : 1

Operace : (W) XOR (f) → (d)

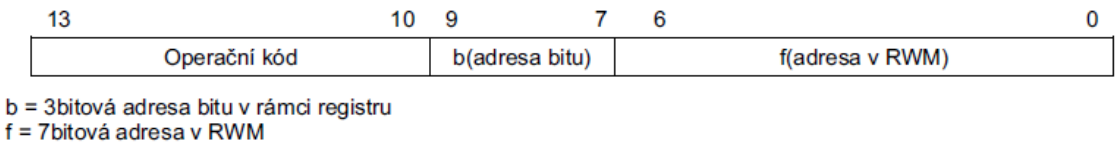
Stavové bity : Z

Interval operandu : 0 ≤ f ≤ 127, d ∈ [0, 1]

Instrukce provede logickou operaci nonekvivalent mezi pracovním registrem W a registrem f. Pokud je hodnota d rovna nule, výsledek se zapíše zpět do registru W. Je-li hodnota d rovna jedné, výsledek je uložen do registru f.

2.4.2 Bitově orientované instrukce [1, 4]

U bitově orientovaných instrukcí (obr.16) adresa f určuje registr, se kterým se pracuje a parametr b určuje pozici bitu ve vybraném registru. Pracuje se tedy pouze s jedním adresovaným bitem.



Obr. 16 Bitově orientované instrukce [5]

Instrukce : **BCF f,b**

Instrukční kód : 01 00bb bfff ffff

Počet cyklů : 1

Operace : 0 → (f<b>)

Interval operandu : 0 ≤ f ≤ 127, 0 ≤ b ≤ 7

Bit b registru f je vynulován.

Instrukce : **BSF f**

Instrukční kód : 01 01bb bfff ffff

Počet cyklů : 1

Operace : 0 → (f<b>)

Interval operandu : 0 ≤ f ≤ 127; 0 ≤ b ≤ 7

Bit b registru f je vynulován.

Instrukce : **BTFSC f,b**

Instrukční kód : 01 00bb bfff ffff

Počet cyklů : 1 - 2

Operace : přeskoč, jestliže f(b) = 0

Interval operandu : 0 ≤ f ≤ 127; 0 ≤ b ≤ 7

Pokud bude bit b registru f roven logické nule, bude přeskočena následující instrukce a provede se instrukce NOP.

Instrukce : **BTFSS f,b**

Instrukční kód : 01 11bb bfff ffff

Počet cyklů : 1-2

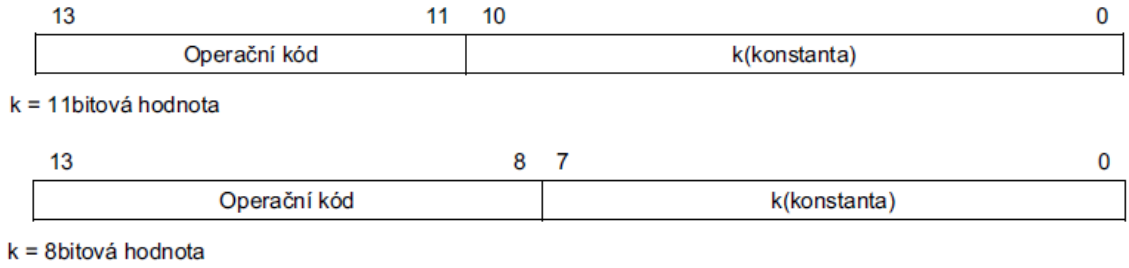
Operace : přeskoč, jestliže f(b) = 1

Interval operandu : 0 ≤ f ≤ 127; 0 ≤ b ≤ 7

Pokud bude bit b registru f roven logické jedničce, bude přeskočena následující instrukce a provede se instrukce NOP.

2.4.3 Řídící instrukce [1, 4]

Řídící instrukce (obr. 17) mění buď tok programu nebo způsob, jakým mikrořadič funguje. Používají zápis k pro určeni osmi-bitové nebo 11-bitové konstanty v případě instrukcí CALL a GOTO.



Obr. 17 Řídící instrukce [5]

Instrukce :	**ADDLW k**
Instrukční kód :	11 111x kkkk kkkk
Počet cyklů :	1
Operace :	(W) + k → (W)
Stavové bity :	C, DC, Z
Interval operandu :	0 ≤ k ≤ 255

Popis :

Data pracovního registru W se algebraicky sečtou s osmi bitovou konstantou k a uloží se zpět do pracovního registru W.

Instrukce :	**ANDLW k**
Instrukční kód :	11 1001 kkkk kkkk
Počet cyklů :	1
Operace :	(W).AND.(k) → (W)
Stavové bity :	Z
Interval operandu :	0 ≤ k ≤ 255

Popis :

Instrukce provede operaci logického součinu mezi daty pracovního registru W a osmi bitovou konstantou k.

Instrukce : **CALL k**

Instrukční kód : 10 0kkk kkkk kkkk

Počet cyklů : 2

Operace : (PC) + 1 → TOS , k → PC<10:0>,
(PCLATH<4:3>) → PC<12:11>

Interval operandu : 0 ≤ k ≤ 2047

Popis :

Instrukce volá podprogram z adresy k v paměti programu**.** Tato adresa podprogramu je vložena do programového čítače PC. Horní dva bity PC jsou doplněny ze záchytného registr programu čítače PCLATCH. V tomto případě se neberou v potaz tři nižší bity z PCLATCH.

Instrukce : **CLRWDT**

Instrukční kód : 00 0000 0110 0100

Počet cyklů : 1

Operace : 00h → WDT , 0 →WDT prescaler ;
1→ $\overline{\textbf{PD}}$, 1→ $\overline{\textbf{TO}}$

Stavové bity : $\overline{\textbf{TO}}$, $\overline{\textbf{PD}}$

Popis :

Instrukce vynuluje WDT a předělič ku pokud je připojena k WDT. Dále se nastaví bity $\overline{\textbf{TO}}$ a $\overline{\textbf{PD}}$ do logické jedničky.

Instrukce : **GOTO k**

Instrukční kód : 10 1kkk kkkk kkkk

Počet cyklů : 2

Operace : k → PC <10:0> ,
(PCLATH<4:3>) → PC<12:11>

Interval operandu : 0 ≤ k ≤ 2047

Popis :

Provede se nepodmíněný skok. Adresa k je vložena do programového čítače PC. Horní dva bity PC jsou doplněny ze záchytného registr programu čítače PCLATCH. V tomto případě se neberou v potaz tři nižší bity z PCLATCH.

Instrukce : **IORLW k**

Instrukční kód : 11 1000 kkkk kkkk

Počet cyklů : 1

Operace : (W).OR.k → (W)

Stavové bity : Z

Interval operandu : 0 ≤ k ≤ 255

Instrukce provede logický součet mezi pracovním registrem **W** a konstantou **k**. Výsledek je uložen do pracovního registru **W.**

Instrukce : **MOVLW k**

Instrukční kód : 11 00xx kkkk kkkk

Počet cyklů : 1

Operace : k → (W)

Interval operandu : 0 ≤ k ≤ 255

Osmi bitová konstanta **k** je vložena do pracovního registru **W**.

Instrukce : **RETFIE**

Instrukční kód : 00 0000 0000 1001

Počet cyklů : 2

Operace : (PC) → STACK, 1 → GIE

Návrat z přerušení. Naplní hodnotu **PC** ze zásobníku a povolí přerušení nastavením bitu GIE do logické jedničky.

Instrukce : **RETLW k**

Instrukční kód : 11 01xx kkkk kkkk

Počet cyklů : 2

Operace : k → (W), TOS → PC

Interval operandu : 0 ≤ k ≤ 255

Návrat z podprogramu. Naplní PC ze zásobníku a registr W naplní konstantou k.

Instrukce : **RETURN**

Instrukční kód : 00 0000 0000 1000

Počet cyklů : 2

Operace : TOS → PC

Návrat z podprogramu. Naplní hodnotu PC ze zásobníku.

Instrukce : **SLEEP**

Instrukční kód : 00 0000 0110 0011

Počet cyklů : 1

Operace : 00h → WDT , 0 →WDT prescaler ;

1→$\overline{\mathbf{PD}}$, 0→$\overline{\mathbf{TO}}$

Stavové bity : $\overline{\mathbf{PD}}$, $\overline{\mathbf{TO}}$

Vynuluje power-down bit $\overline{\mathbf{PD}}$, nastaví time-out bit $\overline{\mathbf{TO}}$, vynuluje čítač Watchdog a jeho předděličku. Procesor přejde do stavu SLEEP, oscilátor je vypnut.

Instrukce : **SUBLW k**

Instrukční kód : 11 110x kkkk kkkk

Počet cyklů : 1

Operace : k - (W) → (W)

Stavové bity : C,DC,Z

Interval operandu : 0 ≤ k ≤ 255

Pomocí dvojkového doplňku se odečte pracovní registr W od konstanty k. Výsledek se uloží do pracovního registru W.

Instrukce : **XORLW k**

Instrukční kód : 11 1010 kkkk kkkk

Počet cyklů : 1

Operace : (W).OR.k → (W)

Stavové bity : Z

Interval operandu : 0 ≤ k ≤ 255

Provede nonekvivalneci (**XOR**) obsah registru **W** s konstantou **k**, výsledek uloží do registru **W**.

2.5 Programovací jazyk Assembler

Pro mikrořadiče PIC® se používá programovací jazyk Assembler a jeho následný překlad do hexadecimálního souboru. Jedná se o jazyk symbolických adres podobný strojovému kódu, tzv. nízkoúrovňový jazyk. Vyznačuje se velkou rychlostí oproti vysokoúrovňovým systémovým jazykům (C, C++, Pascal, atd.) [assembler,c, prog jazyky].

Znaky Assembleru:

- V assembleru se programujeme za pomocí instrukcí.
- Řádkový překladač - na jednom řádku je jeden nebo žádný příkaz,
- Je možné kteroukoliv část vynechat, ale i tak je nutné použít oddělovač.
- Syntaxe: **`Label: příkaz P1,P2 ; komentář (to co je za středníkem překladač ignoruje)`**
    - Label - návěští.
    - P1,P2 - operandy, parametry.
- Pseudoinstrukce – moc se nepoužívají, např. movfw = movf f,0
- Makroinstrukce
- Direktivy - nejpoužívanější rezervovaná slova:
    - `list` - nastavení předvoleb výstupního souboru, zadání typu procesoru
    - `#include` - zdrojový kód z externího souboru
    - `end` - direktiva, která je na konci programu a to co je za ní, jako by nebylo
    - `org` - touto direktivou můžeme ovlivnit oblast paměti, kam (od které) budou ukládány proměnné
    - `equ` - definice konstant.
    - `set` - definice proměnné
    - `_config` - nastavení konfiguračních slov mikrořadiče
    - `cblock, endblock` - definice bloku konstant
    - `banksel` - kód pro výběr datové paměti

3 VOLBA KONSTRUKCE PROGRAMÁTORU

3.1 Problematika programového vybavení

Pro programování mikrořadičů PIC® je potřeba softwarové a hardwarové vybavení, jejichž výběr je navzájem ovlivněn svými parametry. Výběr programového vybavení je třeba zvážit z hlediska jeho využitelnosti. Pro jednoduché programování není potřeba pořizovat drahé vývojové kity.

Nejznámější firmy zabývající se výrobou programového vybavení jsou Microchip, Asix, Elnec a Xeltek. Tyto firmy nabízejí pro svoje produkty plnou softwarovou a hardwarovou podporu, která je však vykoupena cenou v řádech tisíců korun. Jsou tedy v hodné spíše pro profesionální firemní řešení. Jen část jich je cenově přijatelná začínajícím programátorům [2, 10, 11, 12].

3.2 Softwarové vybavení

Mezi hlavní softwarové vybavení řadíme editor pro psaní programu, debugger, překladač, ovládací program programátoru a simulátor. To vše a další nástroje se nachází ve vývojovém prostředí MPLAB [2] od firmy Microchip Technology Inc., některé nástroje mají svá licenční omezení.

Volbu softwarové výbavy z části omezuje operační systém osobního počítače a typ licence použitého softwaru. Nejvíce softwaru je pro operační sytém MS Windows. Distribuce Linuxu v této oblasti nemají takové zastoupení. Pro základní programovací úkony lze na internetu najít dostatek freeware programů.

Jako jednoduchý editor programu může sloužit notepad.exe, vyskytující se standardně v MS Windows. Do něj se zapisuje vytvořený program v jazyce Assembler a ukládá se jako soubor s koncovkou asm. Překladačem MPASM je tento soubor přeložen do hexadecimálního kódu, soubor s koncovkou hex.

Dalším možností pro vytvoření programu je použití vyšších programovacích jazyků C, C++, Pascal atd., které jsou následně kompilovány do asembleru, potažmo hexadecimálního souboru [2, 4].

Získaný hexadecimální soubor se načte do ovládacího programu programátoru a ten se naprogramuje (vypállí) programátorem do mikrořadiče

Debugger slouží k ladění programu a odstranění chyb. Toho se docílí tak, že je program vykonáván postupně po jednotlivých řádkách. Simulátor umí

v počítači softwarově simulovat chování mikrořadiče. Simulátor a debugger zjednodušují a urychlují práci, nejsou nutně potřební pro programování [10].

3.3 Hardwarové vybavení

Hardwarové vybavení může mít několik podob. Například se může jednat o jednoduchý programátor se sériovým portem, emulátor nebo vysoce dimenzovaný vývojový kit. Tyto zařízení nejčastěji pracují s počítačem. Výjimku tvoří vícenásobné programátory schopné pracovat samostatně. Jsou schopny data načítat z paměťových karet.

3.3.1 Komunikační rozhraní s PC

Osobní počítač určuje typ komunikačního rozhraní s programátorem. S rozvojem výpočetní techniky se měnilo komunikační rozhraní. V počátcích byly používány pro komunikaci rozhraní LPT (paralelní port) a R232 (sériový port). Dnes se tyto rozhraní přestávají osazovat do základních desek počítačů. Důvodem jejich zániku je masivní rozšíření modernějšího a rychlejšího rozhraní USB (Universal Serial Bus) [13]. Při nedostupnosti potřebného rozhraní lze využít redukci nebo v případě notebooku dokovací stanici s příslušným rozhraním.

3.3.2 Programátory PIC® [2, 10, 11, 12]

Programátory PIC® slouží k jednoduchému naprogramování, čtení, ověření a mazaní FLASH a EEPROM paměti mikrořadiče. Výjimka nastává při mazání mikrořadiče s typem paměti EPROM. Zde se používá mazačka pracující na principu osvitu okénka mikrořadiče ultrafialovým zářením.

Pro zápis (vypalování) dle typu mikrořadiče PIC® se používají tři způsoby:

- **ICSP** (In-Circuit Serial Programming) způsob programování, který umožňuje programovat součástky, které jsou už osazeny na desce plošných spojů. Programovací napětí (Vpp) je +13 V. Pro naprogramování IO je důležitých pouze pět vodičů: napájecí napětí Vdd a Vss, programovací napětí Vpp přivedené na vývod MCLR, signál clock, přivedený na vývod RB6 a signál data přivedený na vývod RB7. Tyto signály jsou vyvedeny na konektor ICSP a umožňují naprogramovat obvod

přímo v zapojení (In Circuit Serial Programmining) nebo s adaptérem programovat další typy sériově programovatelných obvodů. Signály na ostatních vývodech mikrokontroléru jsou ignorovány [7, 10, 13].

- **ISP** (In-Systém Programming) způsob programování v patici programátoru.
- **LVP** (Low Voltage Programming). způsob programování "nízkým" napětím, např. 3,3V.

Nejběžnější programátory :

- MPLAB ICD 3 USB programátor pro většinu PIC® (Microchip).
- PRESTO - USB programátor pro většinu PIC® (Asix).
- SuperPro 5004GP – vícenásobný univerzální programátor (Xeltek).
- JDM - RS232 programátor pro 8 bitové PIC® [15].
- USBPICPROG – USB programátor (open source projekt) [16].

3.3.3 Emulátor [2, 10]

Emulátor je hardwarový prostředek umožňující napodobovat činnost mikrořadiče. Umožňuje krokování (vykonávání programu po jedné instrukci), za plného běhu lze zobrazovat i měnit hodnoty a v každém okamžiku jsou dostupné všechny údaje.

Nejběžnější emulátory:

- MPLAB REAL ICE – emulátor pro většinu PIC® (Microchip)
- MU Beta - bohatě vybavený hardwarový real-time emulátor (Asix).

3.3.4 Vývojové prototypové a výukové desky [2, 10]

Vývojové, prototypové a výukové desky (kity) představují hotová řešení, připravená k okamžitému použití. Zaměřují se na práci se softwarem, protože hardware je zde pevně daný. Hardware obsahuje dle typu různé periferie např LED indikaci, LCD, klávesnici a různé komunikační rozhraní. Příklady, které se k deskám dodávají, usnadňují a zrychlují osvojení problematiky programování.

Nejběžnější vývojové kity:

- PICkit™ 2, 3 - vývojový kit určený pro začátečníky (Microchip),
- PICDEM 2 Plus - vývojový kit se zaměřený na řadu 18Fxxxx (Microchip),

- PVK40 – - vývojový kit se zaměřený na řadu 16Fxx, 18fxx a 18Fxx (Asix).

3.4 Požadavky na programovací vybavení

Programovací vybavení bylo zvoleno tak, aby neporušovalo žádné licenční podmínky svých produktů. Složitější programátory totiž obsahují naprogramovaný mikrořadič PIC®, který zajišťuje komunikaci s programovým vybavením v počítači. Na internetu se nachází mnoho „clonů“, které kopírují komerční provedení programátorů. A to jak z hardwarového hlediska tak i softwarového (firmware).

Zvolený programátor by měl vycházet z licence open source projektu nebo z možnosti použití freeware softwaru volně dostupného na internetu. Výhodou by byla možnost použití pro dva operační systémy Linux a MS Windows.

Podpora programování 8 bitových mikročipů PIC® a to řad 10Fxxx, 12Fxxx, 16Fxx, 16Fxxx, 18Fxxx, 18Fxxxx. Zápis bude prováděn přes ICSP rozhraní programovacím napětím (Vpp) +13V bez nutnosti instalace dalšího externího napájecího zdroje. Potřebné programovací napětí musí být získáno z počítačového rozhraní USB nebo RS 232 [1].

Musí být nabídnuto rozumné alternativní řešení problému „slepice, vejce“, tj. nutnost naprogramování řídícího obvodu jiným programátorem.

Hardwarové zapojení musí být spolehlivé a ovladače pro něj snadno nainstalovatelné na PC.

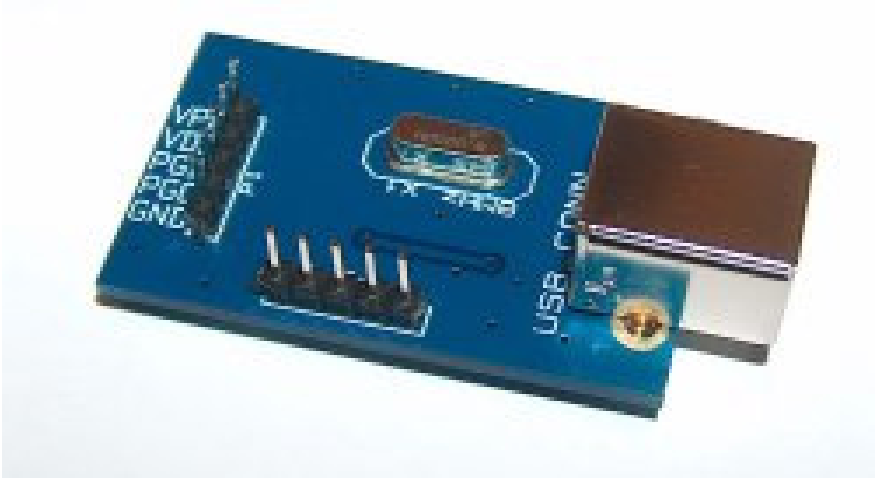
Shrnutí požadavků:

- Zapojení neporušující licenční podmínky
- Obslužný software volně dostupný (freeware)
- Možnost programování většiny 8 bitových mikročipů PIC®.
- Zápis přes ICSP rozhraní
- Programátor bez externího zdroje
- Připojení programátoru přes rozhraní USB
- Vyřešení problému „slepice a vejce“
- Spolehlivé zapojení a nenáročná obsluha.

3.5 Zvolené programovací vybavení

Zadané požadavky na programovací zařízení splňuje open source projekt Usbpicprog 0.2.0 (obr. 18, 19) osazeny PIC18F2550, který bude naprogramován pomocí jednoduchého JDM programátoru připojeného k počítači přes rozhraní RS232. K programátoru je „přibalen“ i software s podporou MS Windows, Mac OS X a několika distribucí Linuxu [16].

Obr.18 Usbpicprog 0.2.0 (BOTTOM)[15]

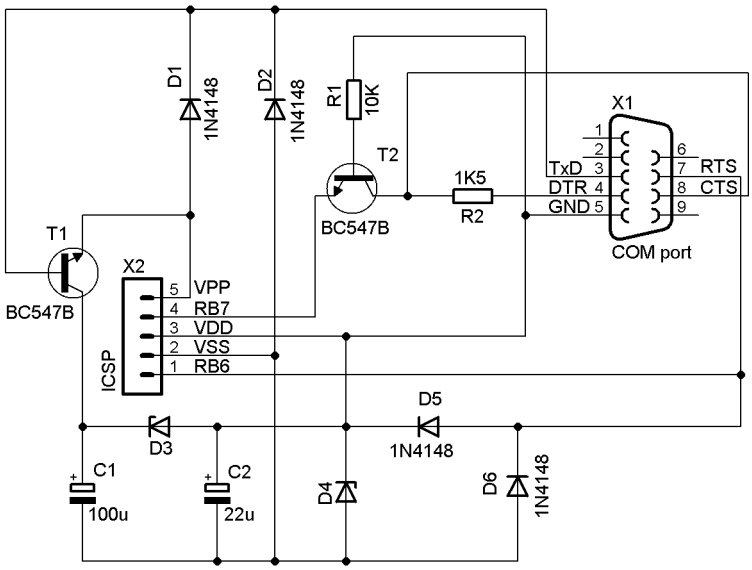


Obr.19 Usbpicprog 0.2.0 (TOP) [15]

4 REALIZACE PROGRAMÁTORU

4.1 JDM programátor [15]

Konstrukce JDM programátoru (obr 20) je velice jednoduchá a dá se zvládnout pomocí součástek ze „šuplíkových zásob“. Konstrukci můžeme provést na nepájivém poli nebo na univerzální desce plošného spoje. Tento programátor zvládá naprogramovat většinu 8 bitových mikrořadičů PIC® podporující programovací rozhraní ICSP.



Obr. 20 Zapojení JDM programátoru

4.1.1 Popis zapojení

Problém potřeby 13V programovacího napětí (Vpp) byl vyřešen v programátoru JDM následovně. Sériový port je napájen z počítače teoretickým napětím ±12 V, u notebooku se používá zdvojené napětí 5V. V praxi je pak napětí u počítače cca ±11,5V a notebooku ±9,5V. To ovšem není dostatečné programovací napětí (Vpp) a proto je kladné napájecí napětí Vdd spojeno se zemí PC. Dále získané napájecí napětí (Vss) ze sériového portu je omezeno a stabilizováno diodami D4 - D6 na -5,1V. Záměnou uzemnění (GND) za záporné napětí (Vss) -5,1V o proti programovacímu napětí (Vpp) získáme dostatečné programovací napětí (Vpp). Z toho vyplívá, že potřebné napájecí napětí z počítače je nyní 8V. Napájecí napětí z RTS tedy omezíme diodou D3 na asi

8,2 V a proti napájecímu napětí (Vss) -5,1 V získáme potřebné 13,3 V programovací napětí (Vpp) [15, 17, 18].

Vývod RTS má dvě funkce a to jako zdroj napájecího napětí a zdroj taktovacích impulsů pro programovaný IO. V době impulsu na vývodu RTS slouží kondenzátor C2 jako zdroj napájecího napětí. Protože se jako zdroj napájecího napětí využívá i signál TxD musí se jeho výchylky hlídat diodami D1 a D2 [15].

Tranzistor T1 pracuje jako spínač a tranzistor T2 jako obousměrný převodník napěťových úrovní.

4.1.2 Konstrukce programátoru

Konstrukce byla upravena proti původní verzi změnou konektoru CANNON25 na u počítačů dostupnější verzi CANNON9F. Dále byla vypuštěna nepotřebná patice pro integrované obvody. Součástky programátoru (tab.2) stačí zapájet do DPS a programátor je připraven na oživení.

R1	10 kΩ
R2	1,5 kΩ
C1	100 uF/16 V
C2	22 uF/16 V
D4	Zenerova dioda 5,1 V
D3	Zenerova dioda 8,2 V
D1, D2, D5, D6	1N4148
T1, T2	BC547B
X1 konektor CANNON9F do DPS	
DPS JDM_PROG	

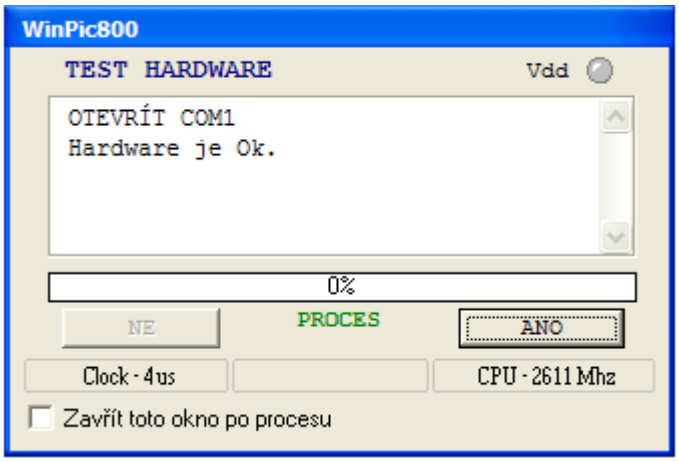
Tab.2 Seznam součástek JDM programátoru

4.1.3 Oživení programátoru a ovládací program

Ovládací program Winpic800 (obr. 21) [19] je volně dostupný na Internetu. Umožňuje načíst data k programování ze souborů v několika formátech, číst, mazat a zapisovat program do mikrořadiče, editovat data a mnoho dalšího nastavení. Program podporuje hned několik hardwarových programátorů a proto je nutné po prvním spuštění změnit (v Nastavení→Hardware) programátor na JDM Programmer.

Obr. 21 Program WinPIC800

Po sestavení programátoru a jeho připojení do sériového portu ověříme pomocí ovládacího programu jeho funkčnost (obr. 22). Nyní je programátor připraven a plně funkční. Pro programování stačí nastavit daný typ mikrořadiče a připojit pomocí „kšandy“ přes rozhraní ICSP.



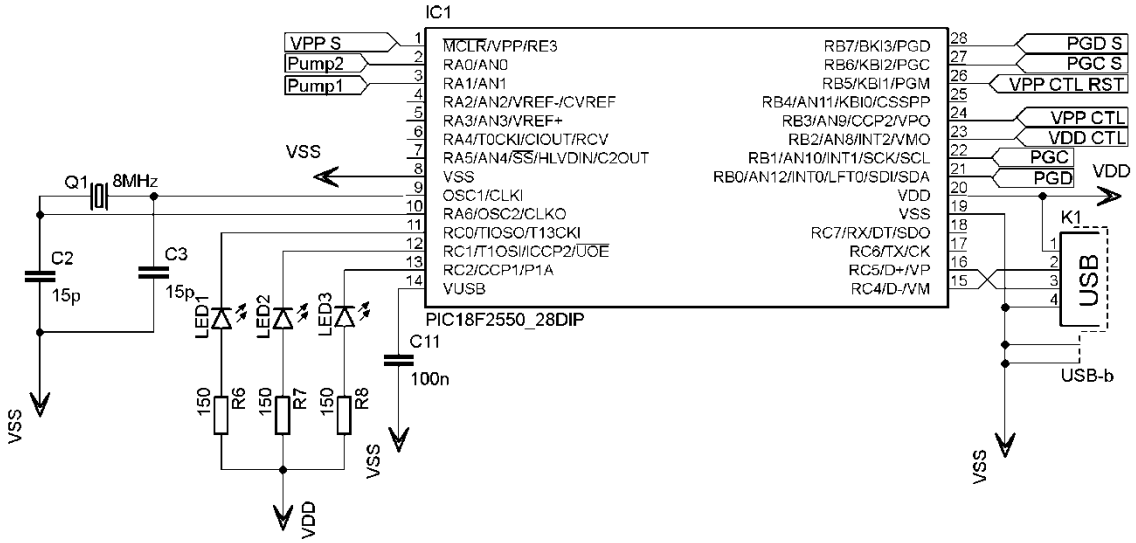
Obr. 22 Ověření programátoru ve WinPIC800

4.2 Usbpicprog 0.2.0 programátor

Usbpicprog [16] je open source projekt programátoru pro Microchip PIC® s komunikačním rozhraním USB. Samotný projekt je rozdělen do tří částí Hardware, Firmware a PC Software. Tento projekt je nadále rozvíjen, a proto se můžeme dočkat rozrůstající databáze podporovaných integrovaných obvodů.

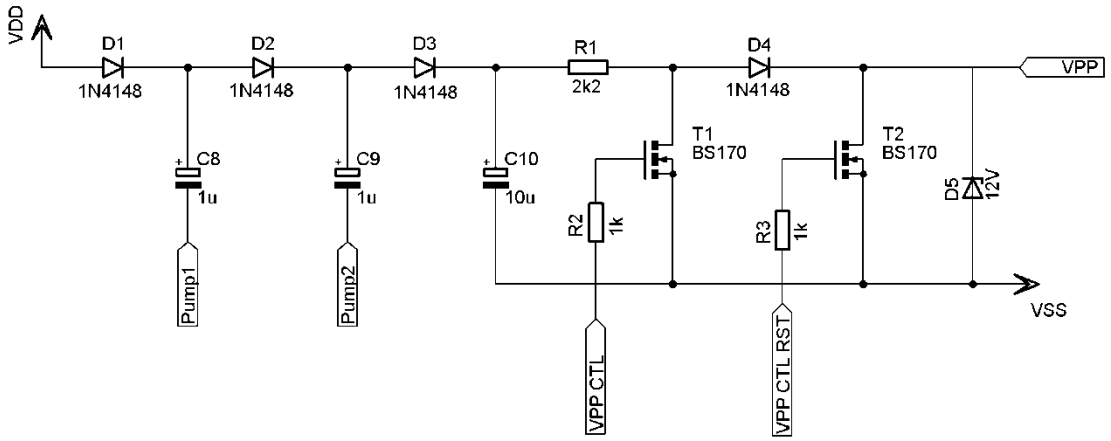
4.2.1 Hardware [16]

Základem zapojení je již zmíněný procesor PIC18F2550 IC1 (obr.23) ovládající všechny funkce programátoru. Takt obstarává externí oscilátor skládající se z dvojice kondenzátorů C2, C3 a krystalu Q1. Indikaci stavu zajišťuje trojice LED diod. Přes konektor USB-b se připojujeme k počítači.



Obr. 23 Zapojení mikrořadiče

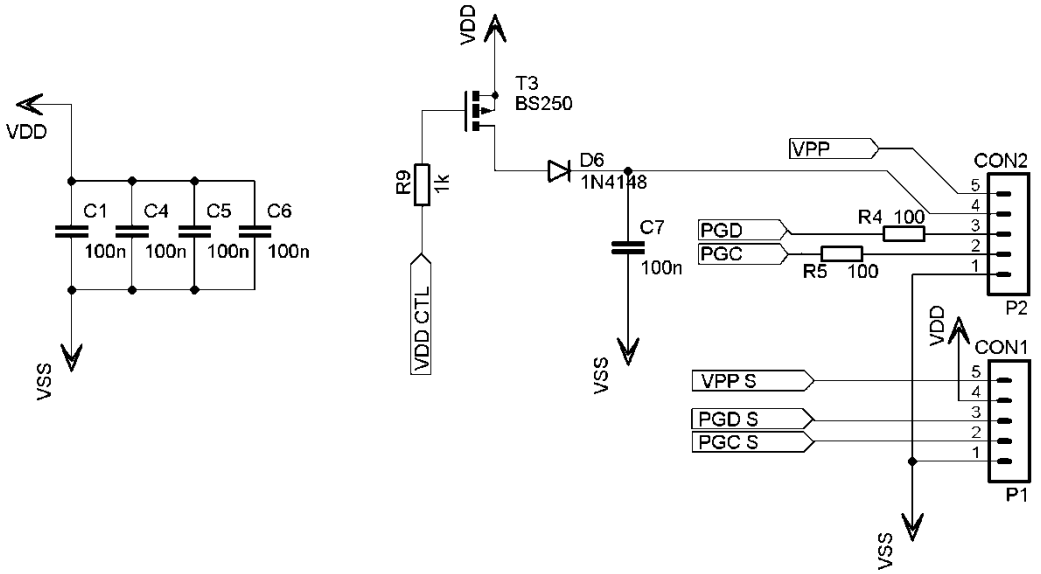
Programovací napětí 13V (Vpp) generuje nábojová pumpa (obr. 24) tvořená diodami kondenzátory C8, C9, C10 a diodami D1, D2, D3. řízení této pumpy obstarává přímo mikrořadič z pinů RA0 a RA1. K ovládání programovací napětí slouží tranzistory T1 a T2.



Obr. 24 Nábojová pumpa

Pro naprogramování samotného jádra PIC18F25550 slouží konektor

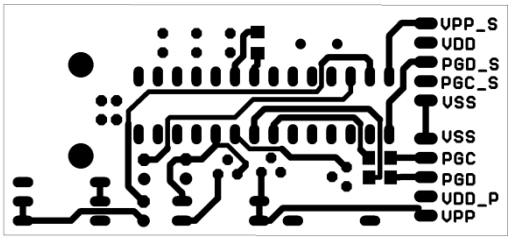
CON1 (obr. 25), který je po té uzavřen JUMPERY. CON2 je pak ICSP rozhraní pro programování mikrořadičů. Kondenzátory C1, C4, C5, C6 filtrují napětí.

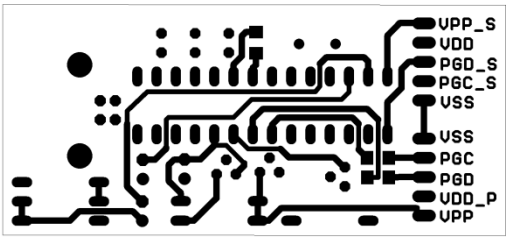


Obr. 25 Zapojení konektorů programátoru

4.2.1.1 Konstrukce

Konstrukce programátoru se skládá z oboustranného plošného spoje. Schémata byly vytvořeny v programu Eagle 5.5 light a následně vytištěny na průhlednou fólii určenou pro inkoustové tiskárny. Předlohy (obr. 26, 27) byly přiloženy a zajištěny na DPS s fotocitlivou vrstvou. Dále proběhlo krátké osvícení UV světlem a následné vyvolání DPS ve vývojce. Následovalo vyleptání DPS v leptacím roztoku $FeCl_3$. Po očištění následovalo nanesení ochranného pájitelného laku. Dalším krokem po zaschnutí laku bylo vyvrtání děr. Posledním krokem bylo zapájení součástek. Velikost desky je 31x67 mm. Seznam součástek viz příloha C.





Obr.26 DPS Usbpicprog (TOP) *Obr.27 DPS Usbpicprog (BOTTOM)*

4.2.2. Firmware

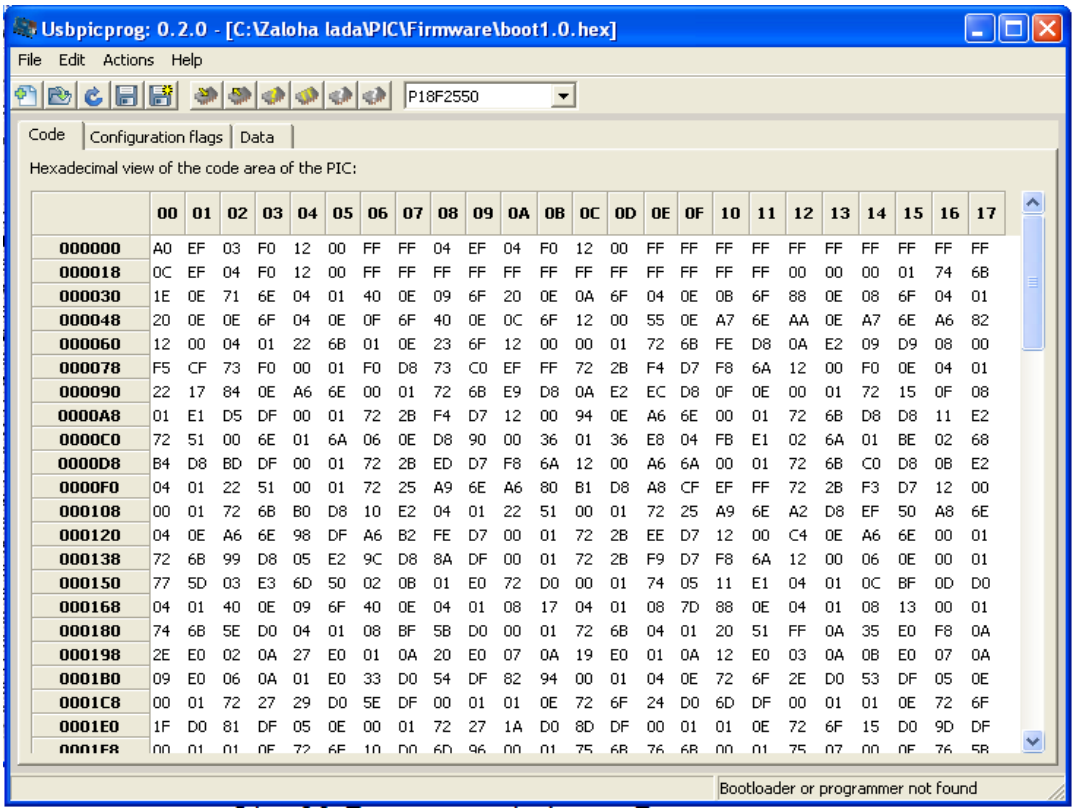
K programátoru jsou dostupné dva hexadecimální soubory Boot.hex a firmware-0.2.0.hex. Mikrořadič PIC18F2550 obsahuje zařízení podporující Bootloader. To je schopnost programátoru po nahrání Boot.hex do spodní části paměti mikrořadičů samostatně nahrávat naše vlastní programy (firmware-0.2.0.hex). Z toho vyplývá, že pro daný typ PIC® stačí pouze jeden zápis z jiného programátoru. Bootloader se využívá především pro možnost aktualizace firmware.

4.2.3 PC Software [16]

Instalace usbpicprog-0.2.0-Setup.exe obsahuje program (obr. 28) pro základní operace s mikrořadičem. Program obsahuje autodetekci programátoru s bootloader nebo plným firmwarem.

Pro zpřístupnění programátoru se používají dva ovladače Microchip FS USB.inf a usbpicprog.inf pro windows.

Software libusb-win32-filter-bin-0.1.12.1.exe slouží pro přehled a kontrolu o připojených zařízení k USB.



Obr. 28 Program usbpicprog [16]

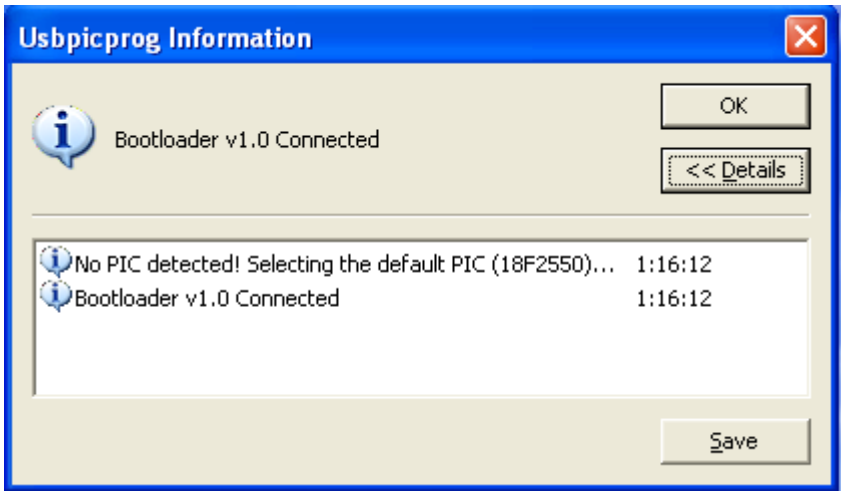
4.3 Oživení programátoru

Do programátoru vložíme PIC18F2550. Nyní připojíme JDM programátor na konektor usbpicprog CON1 (VPP->VPP_S, VDD->VDD, RB7->PGD_S, RB6->PGC_S a VSS->VSS). V programu Winpic800 nastavíme PIC18F2550 a dáme načíst Boot.hex. Nyní můžeme spustit zápis (vypalování) a následné ověření zápisu. JDM programátor můžeme odpojit.

V programátoru usbpicprog máme uložen program pro bootloader, takže můžeme přejít k naprogramování jádra programátoru. Spojíme jumperem pin 5 a 4 na CON1. K počítači připojíme přes kabel usb (k tiskárně) programátor usbpicprog. Instalace ovladačů proběhne v Plug and Play režimu. Systém nás vyzve k zadání cesty k ovladačům. Po nainstalování zapneme program usbpicprog.exe a ten detekuje námi zapsaný Bootloader (boot.hex) v mikrořadiči.

V programu usbpicprog nastavíme PIC18F2550 a načteme do něj firmware-0.2.0.hex. Teď stačí vypálit firmware do mikrořadiče a vypojit z USB. Abychom zabránily jeho dalšímu nechtěnému přepsání, spojíme ještě jumperem piny 1, 2 na CON1

Programátor můžeme nyní připojit k USB a spustit program. Ten detekuje firmware mikrořadiče (obr. 29) a po potvrzení je připraven k práci.



Obr. 29 Detekce programátoru Usbpicprog 0.2.0

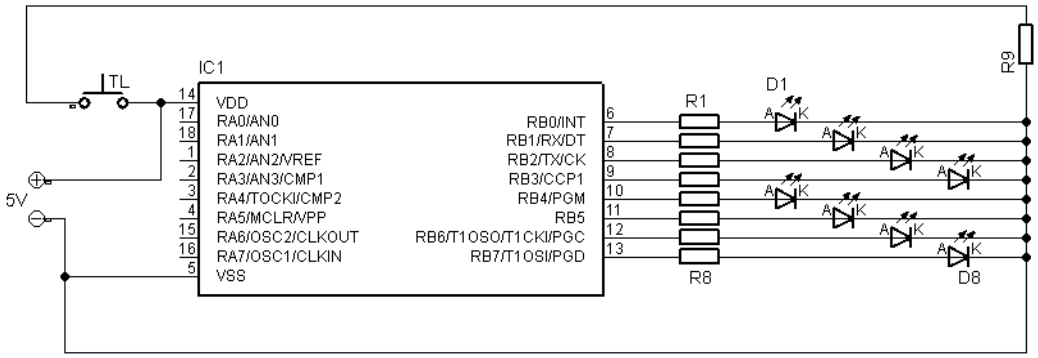
5 REALIZACE ÚLOHY

5.1 Zadání

1. Sestrojte Světelného hada z osmi LED diod (obr. 30), které budou ovládané pomocí mikrořadiče 16F628A. Jako výstupy použijte PORTB a jako vstup PORTA.
2. Přidejte další dva stavy blikání LED diod přepínané pomocí jednoho tlačítka.

Tato úloha je zaměřena na seznámení s mikrořadičem PIC16F628A a je v hodná pro výuku na středních školách a vysokých školách. Na zapojení ověříme funkčnost programátoru Usbpicprog. Seznámíme s nástinem postupu při realizaci projektu Světelného hada [7, 9, 20].

Jedná se o. jednoduché zapojení osmi LED diod, které jsou připojené přes rezistory R1 až R8(560Ω) na PORTB mikrořadiče. Tlačítkem připojeným na PORTA,0 budeme přepínat jednotlivé stavy.



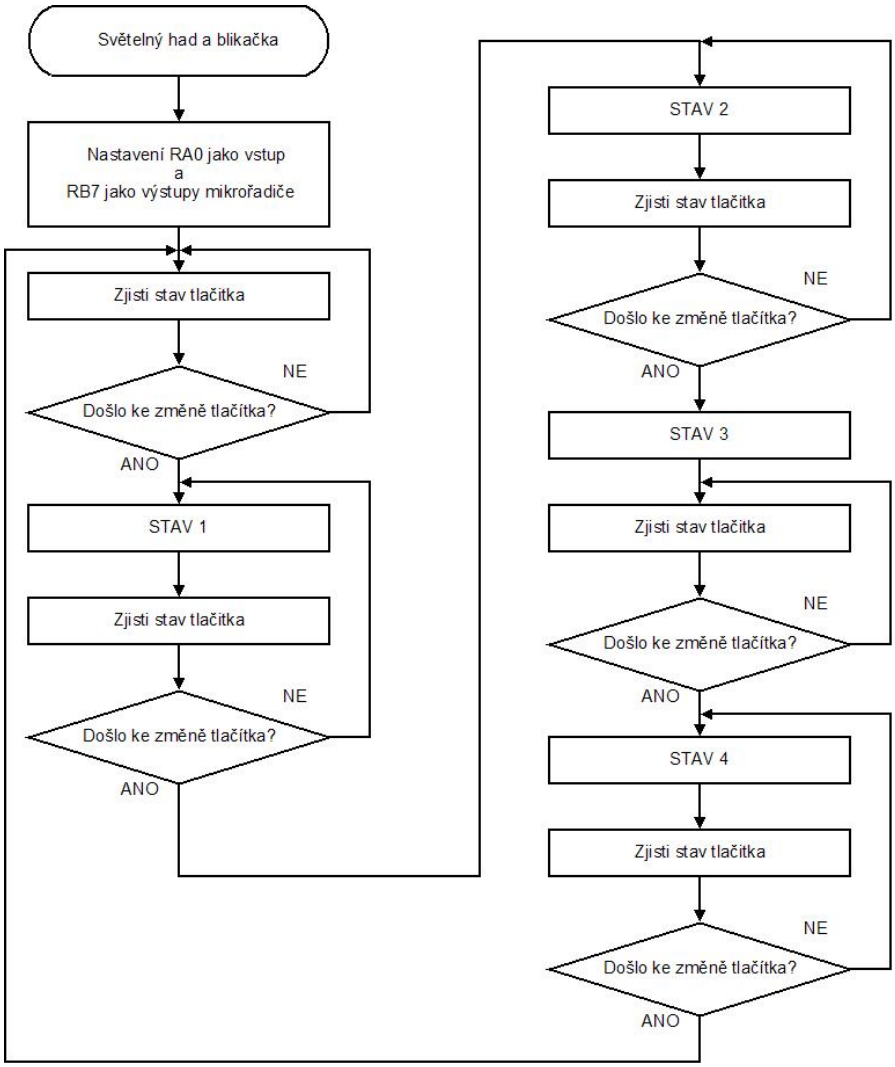
Obr. 30 Zapojení LED diod

5.2 Vypracování

Na začátku si musíme nejprve sestrojit diagram (obr. 31) našeho budoucího programu podle zadaného úkolu. Diagram nám zjednoduší naši práci při vytváření programu v asembleru.

V prvním bloku diagramu je název programu nebo funkce, které chceme dosáhnout. Následuje přiřazení vstupu a výstupů mikrořadiče dle zadání.

V dalším kroku přichází na řadu zjištění stavu tlačítka., z kterého vyplívá následné rozvětvení dle výsledku. Pokud není tlačítko zmáčknuto tak se vrátí program zpět před blok „Zjistěte stav tlačítka“. To probíhá do doby, dokud není tlačítko zmáčknuto. Po-té dokola běží stav „STAV 1“, který probíhá do doby dalšího zmáčknutí tlačítka. Tímto přejdeme na „STAV2“. Analogicky následují další dva stavy a po¨nich návrat za blok nastavení portů [7, 9].



Obr. 31 Diagram programu

Po sestavení diagramu přistoupíme k realizaci programu. Vysvětlivky k programu jsou napsány za středníky. Stavy blikání se analogicky opakují, proto jsou napsány pouze dva stavy. Plná verze progamu světelného hada se nachází v příloze C.

```
;*************************************************************
;            SVETELNY HAD;
;                                          Author : Musel Ladislav
;*************************************************************
      list p=16f628A         ;určuje typ použitého mikrořadiče
      include<p16f628A.inc>  ;potřebné údaje o mikrořadiči
      __config _pwrte_on & _wdt_off & _mclre_off & _boden_off &
_lvp_off & _intrc_osc_noclkout ;nastavení mikrořadiče

cislo equ   20h         ; přirazeni názvu registrům
cisloa      equ   21h

#define     tlac  porta,0

      org   00h         ; na první pozici v paměti bude goto start
      goto  start

Cekani  clrf            cislo   ;podprogram cekani
         movlw    180          ;toto číslo nastavuje rychlost blikání
         movwf    cisloa       ;délku časové smyčky
cekani2 incfsz    cislo,1
        goto      cekani2
        decfsz    cisloa,
        goto      cekani2
        return                 ;návrat z podprogramu

start   movlw     b'00000111'  ;typ komparátoru (off)
        movwf     cmcon
        bsf       status,rp0   ;nastaveni in / out
        movlw     b'00000001'
        movwf     trisa
        clrf      trisb
        bcf       status,rp0

;********************  prvni stav (tma) **********************

stav1z      clrf  portb
       btfsc      tlac ;zjišťuje stav tlačítky, při nule přeskočí řádek
       goto $-1         ;způsobí vrácení procesoru o jeden řádek víš

stav1  movlw      b'00000000'  ;uloží na port samé nuly (nic nesvítí)
       movwf      portb
       call       cekani    ;vyvolá se podprogram cekani(blikání)
       btfsc      tlac ;zjišťuje stav tlačítky, při nule přeskočí řádek
       goto       stav2z    ;skočí na druhý stav
       goto       stav1     ;vrátí se na začátek prvního stavu

;********************  druhy stav (vse sviti) ****************
stav2z clrf       portb
       btfsc      tlac ;zjišťuje stav tlačítky, při nule přeskočí řádek
       goto       $-1    ;způsobí vrácení procesoru o jeden řádek víš

stav2  movlw      b'11111111'  ;uloží na port samé nuly (vše svítí)
       movwf      portb
       call       cekani
       btfsc      tlac
       goto       stav3z
       goto       stav2
```

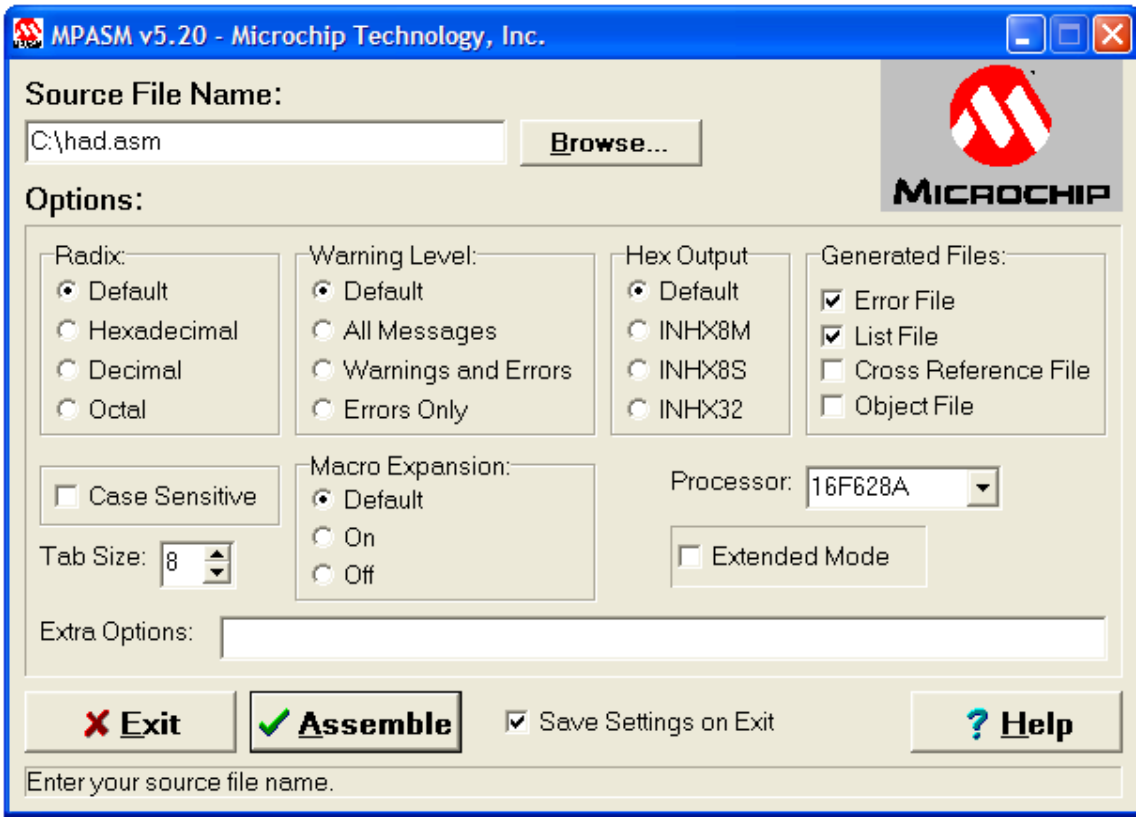
```
;********************  treti stav (had) ****************

stav3z      clrf  portb
      btfsc tlac
      goto  $-1

.
.
.
.
.
       goto stav1z ;vrátí se na stav1

end                        ;konec programu
```

Po dopsání programu v textovém editoru uložíme soubor s názvem had.asm. Následuje překlad do hexadecimálního souboru (had.hex), který provedeme v programu MPASM.exe (obr. 32). Tento program je volně ke stažení na stránkách firmy Microchip®. Nyní stačí pouze naprogramovat mikrořadič 16F628A a vložit ho do zapojení [7, 9, 20].



Obr. 32 Nastavení v programu Mspasm.exe

6 ZÁVĚR

Bakalářská práce obsahuje přehled základních informací o mikrořadičích PIC®. Protože toto téma je velice obsáhlé, je práce soustředěna na jeden konkrétní 8-bitový mikrořadič 16F628A. Díky tomu že, 8-bitové mikrořadiče jsou postavené na stejném základu, lze pracovat i s ostatními mikrořadiči této řady na stejném principu.

Dalším z úkolů této práce bylo sestrojit programátor mikrořadičů PIC® a ověřit jeho funkci. Pro představu jsem uvedl několik typů zařízení používaných v rámci programování mikrořadičů. Následně byli z konstruovány dva programátory. První jednoduchý JDM programátor (Příloha A) pracující přes rozhraní RS232. Jeho funkce byla ověřena při naprogramování druhého programátoru usbpicprog 0.2.0 (Příloha B). Tento druhý programátor komunikuje s počítačem přes rozhraní USB 2.0. Mezi jeho plusy patří především rychlost a podpora více operačních systémů.

Ověření funkce tohoto programátoru zajistil poslední úkol a to realizace světelného hada. K tomuto úkolu byl vypracován podrobný návod jak postupovat při jeho realizaci. Celý program v Assemleru se nachází v příloze C.

Programátor usbpicprog je určený pro programování přes rozhraní ICSP, které podporují převážně jen osmi bitové typy mikrořadičů PIC®. Pro rozšíření stačí upravit zapojení a PC software tak, aby bylo podporováno programovací rozhraní LVP. Touto úpravou je možnost získat programátor s podporu 16 a 32 bitových mikrořadičů PIC®.

Použitá literatura

[1] Microchip®, katalogové listy.

[2] URL: < http://www.microchip.com>, 20.4.2009 8:00.

[3] PREDKO M.: PROGRAMMING AND CUSTOMIZING THE PIC® MICROCONTROLLER, Tab Electronics, 2008, 3.vydání.

[4] HRBÁČEK, J.: Mikrořadiče PIC16CXX a vývojový kit PICSTART, BEN technická literatura, Praha 1995 -1997, 4.vydání

[5] URL : < http://www.copsu.cz/mikrop/>, 20.4.2009 8:00

[6] VACEK, V.,VACKOVÁ, V.: Učebnice programování PIC, BEN technická literatura, Praha 2000

[7] HRBÁČEK, J.: Moderná učebnica programovania mikrokontrolérov PIC, BEN technická literatura, Praha 2004.

[8] PEROUTKA, O.: Mikrokontroléry PIC16F87X a důležité rozdíly mezi řadou PIC 16F87X a PIC 16F87XA, BEN technická literatura, Praha 2005.

[9] HRBÁČEK, J.: Moderní učebnice programování mikrokontrolérů PIC 2, BEN technická literatura, Praha 2007.

[10] URL : < http://www.asix.com/>, 20.4.2009 8:00.

[11] URL : < http://www.elnec.com/>, 20.4.2009 8:00.

[12] URL : < http://www.xeltek.cz/>, 20.4.2009 8:00.

[13] KAINKA, B.: Měření, řízení a regulace pomocí sběrnice USB. Praha, BEN, 2002.

[14] URL : <http://www.belza.cz>, 20.4.2009 8:00.

[15] URL : <http://www.jdm.homepage.dk/newpics.htm>, 20.4.2009 8:00.

[16] URL : <http://usbpicprog.org>, 20.4.2009 8:00.

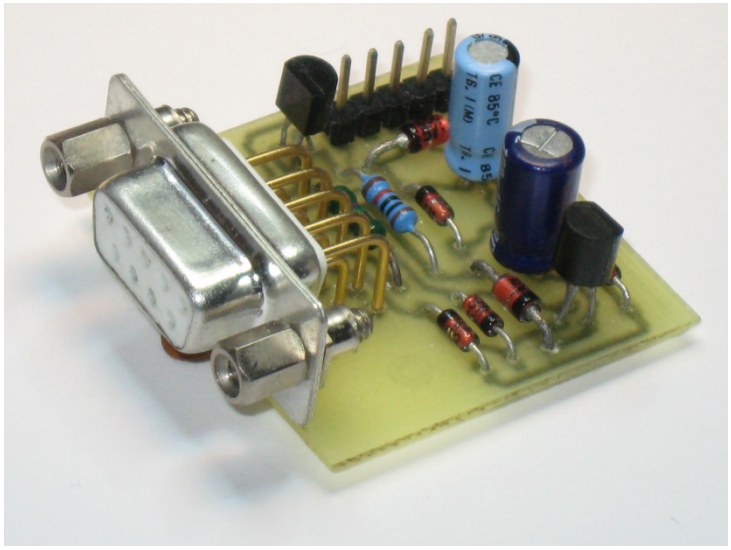
[17] VEDRAL, J.: KUBÍČEK, M.: Elektrické obvody měřicích přístrojů. Praha, Vydavatelství ČVUT, 1993.

[18] ČERNOCH, H., VALENTA, V.: Aplikovaná elektronika. Praha, ČVUT v Praze, 1992.

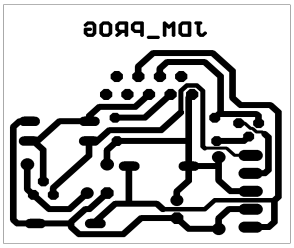
[19] URL : http://winpic800.com, 20.4.2009 8:00.

[20] URL : < http://pandatron.cz/>, 20.4.2009 8:00.

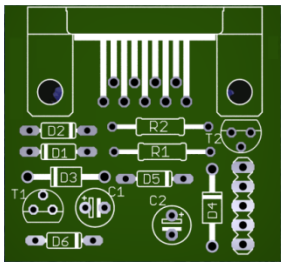
Příloha A



Obr. 33 JDM programátor

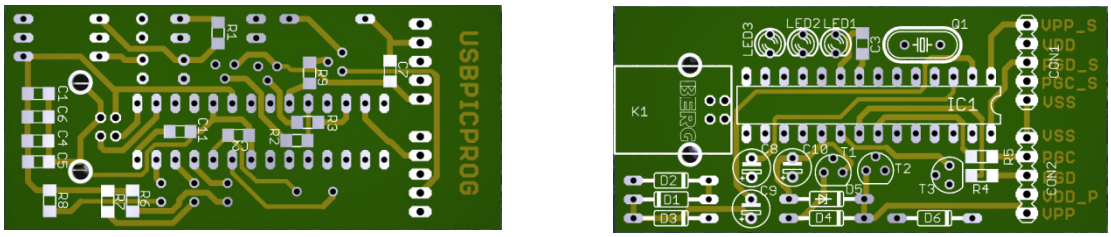


Obr. 34 DPS JDM programátoru

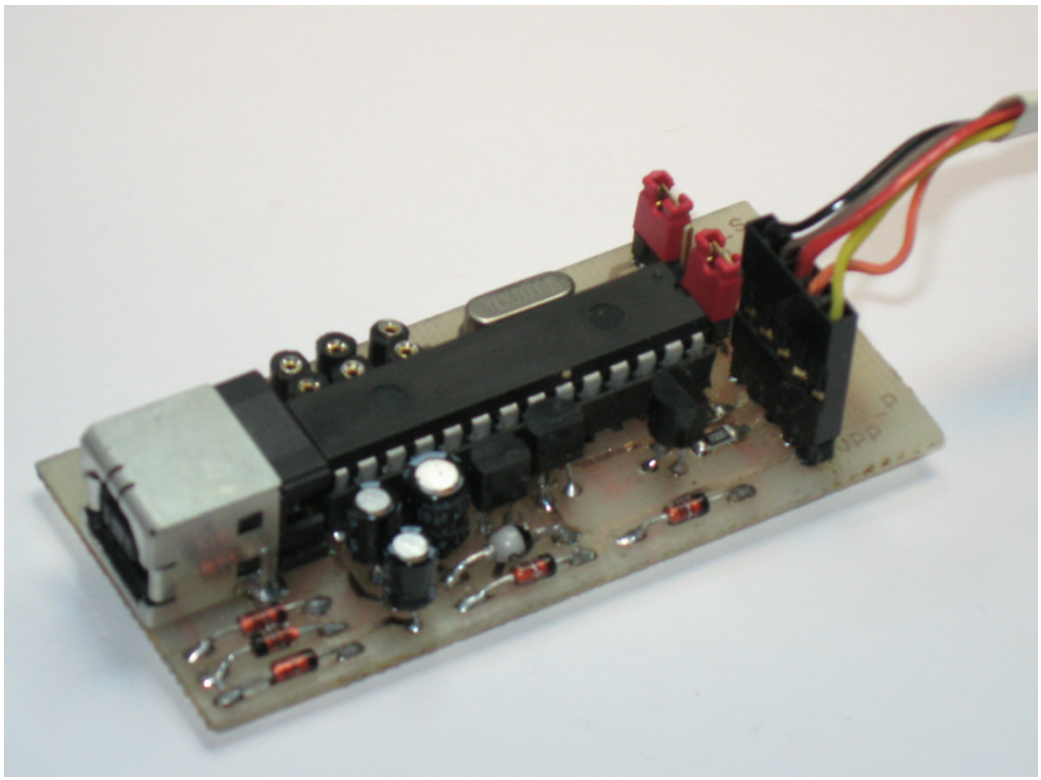


Obr. 35 Rozmítnění součástek na DPS

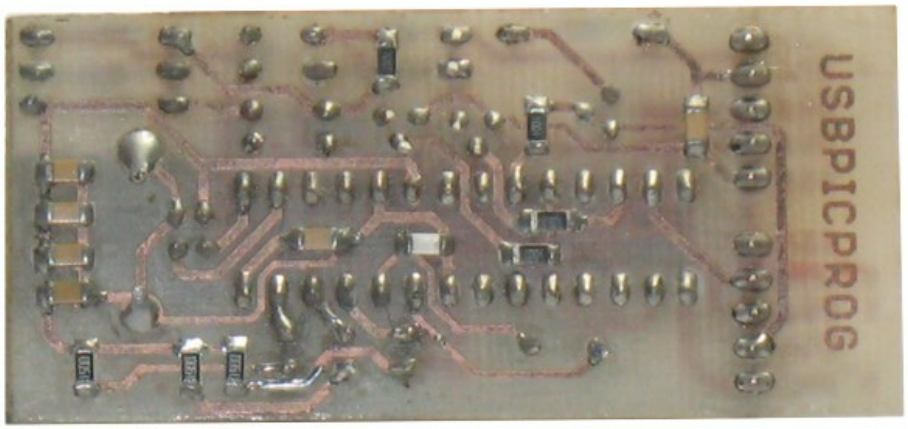
Příloha B



Obr. 36 Rozmítnění součástek na DPS usbpicprog (zleva BOTTOM,TOP)



Obr. 37 Porgramátor usbpicprog (TOP)



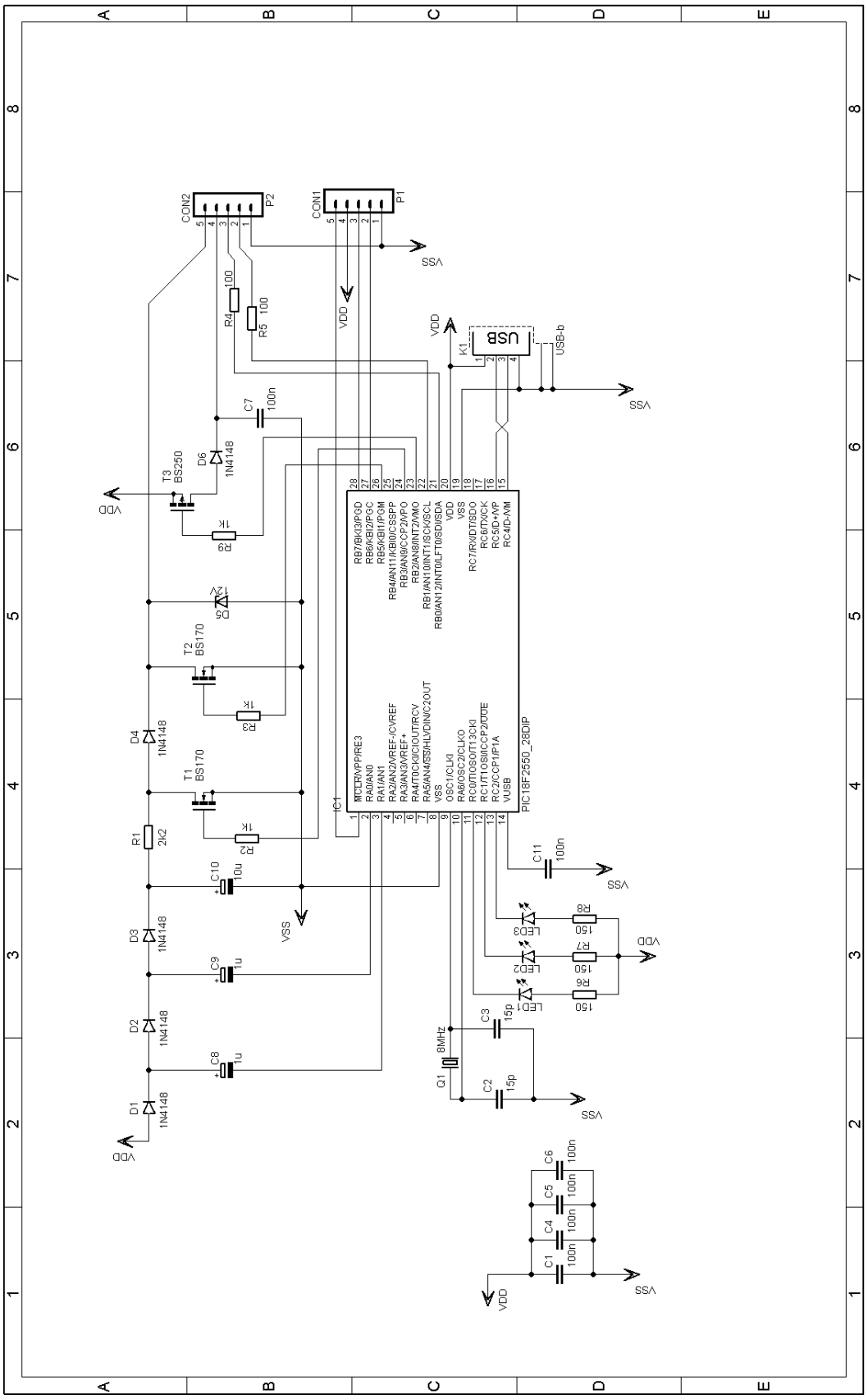
Obr. 38 Programáto usbpicprog (BOTTOM)

Seznam použitých součástek

```
EAGLE Version 5.4.0 Copyright (c) 1988-2009 CadSoft

Part     Value             Device              Package   Library

C1       100n              C-EUC1206           C1206     rcl
C2       15p               C-EUC1206           C1206     rcl
C3       15p               C-EUC1206           C1206     rcl
C4       100n              C-EUC1206           C1206     rcl
C5       100n              C-EUC1206           C1206     rcl
C6       100n              C-EUC1206           C1206     rcl
C7       100n              C-EUC1206           C1206     rcl
C8       1u                CPOL-EUE2.5-5       E2,5-5    rcl
C9       1u                CPOL-EUE2.5-5       E2,5-5    rcl
C10      10u               CPOL-EUE2.5-5       E2,5-5    rcl
C11      100n              C-EUC1206           C1206     rcl
CON1     P1                MA05-1              MA05-1    con-lstb
CON2     P2                MA05-1              MA05-1    con-lstb
D1       1N4148DO35-10     1N4148DO35-10       DO35-10   diode
D2       1N4148DO35-10     1N4148DO35-10       DO35-10   diode
D3       1N4148DO35-10     1N4148DO35-10       DO35-10   diode
D4       1N4148DO35-10     1N4148DO35-10       DO35-10   diode
D5       12V               ZENER-DIODEDO35Z10  DO35Z10   diode
D6       1N4148DO35-10     1N4148DO35-10       DO35-10   diode
IC1      PIC18F2550_28DIP  PIC18F2550_28DIP    DIL28-3   microchip
K1       USB-b             PN61729-S           PN61729-S con-berg
LED1                       LED3MM              LED3MM    led
LED2                       LED3MM              LED3MM    led
LED3                       LED3MM              LED3MM    led
Q1       8MHz              CRYSTALHC49U-V      HC49U-V   crystal
R1       2k2               R-EU_R1206          R1206     rcl
R2       1k                R-EU_R1206          R1206     rcl
R3       1k                R-EU_R1206          R1206     rcl
R4       100               R-EU_R1206          R1206     rcl
R5       100               R-EU_R1206          R1206     rcl
R6       150               R-EU_R1206          R1206     rcl
R7       150               R-EU_R1206          R1206     rcl
R8       150               R-EU_R1206          R1206     rcl
R9       1k                R-EU_R1206          R1206     rcl
T1       BS170             BS170               SOT54E    transistor-small-signal
T2       BS170             BS170               SOT54E    transistor-small-signal
T3       BS250              S250               SOT54E    transistor-small-signal
Kabel USB (k tiskárně)
```



Obr. 39 Celkové schéma Usbpicprog 0.2.0

Příloha C

```
;*************************************************************
;            SVETELNY HAD;
;                                          Author : Musel Ladislav
;*************************************************************
      list p=16f628
      include<p16f628.inc>
      __config _pwrte_on & _wdt_off & _mclre_off & _boden_off
& _lvp_off & _intrc_osc_noclkout

cislo equ   20h
cislo2      equ   21h

#define     tlac  porta,0

org   00h
goto  start

;*************************************************************
;        ZPOZDENI
;*************************************************************
cekani      clrf  cislo
      movlw   200
      morf  cislo2
cekani2     incfsz      cislo,1
      goto  cekani2
      decfsz      cislo2,1
      goto  cekani2
      return

start movlw b'00000111'
      morf  cmcon
      bsf   status,rp0
      movlw b'00000001'
      morf  trisa
      clrf  trisb
      bcf   status,rp0
;*************************************************************
;        NASTAVENI PETI STAVU BLIKANI LED
;*************************************************************
;********************  prvni stav (tma) **********************
stav1z       clrf     portb
             btfsc    tlac
             goto     $-1

stav1        movlw    b'00000000'
             morf     portb
             call     cekani
             btfsc    tlac
             goto     stav2z
             goto     stav1
;********************  druhy stav (vse sviti) ****************
stav2z       clrf     portb
             btfsc    tlac
             goto     $-1

stav2        movlw    b'11111111'
             morf     portb
             call     cekani
             btfsc    tlac
```

```
            goto    stav3z
            goto    stav2
;********************  treti stav (vse sviti) ****************
stav3z      clrf    portb
            btfsc   tlac
            goto    $-1
stav3       movlw   b'00000001'
            morf    portb
            call    cekani
            btfsc   tlac
            goto    stav4

            movlw   b'00000010'
            morf    portb
            call    cekani
            btfsc   tlac
            goto    stav4

            movlw   b'00000100'
            morf    portb
            call    cekani
            btfsc   tlac
            goto    stav4

            movlw   b'00001000'
            morf    portb
            call    cekani
            btfsc   tlac
            goto    stav4

            movlw   b'00010000'
            morf    portb
            call    cekani
            btfsc   tlac
            goto    stav4

            movlw   b'00100000'
            morf    portb
            call    cekani
            btfsc   tlac
            goto    stav4

            movlw   b'01000000'
            morf    portb
            call    cekani
            btfsc   tlac
            goto    stav4

            movlw   b'10000000'
            morf    portb
            call    cekani
            btfsc   tlac
            goto    stav4z
            goto    stav3
;********************  ctvrty stav (vse sviti) ****************
stav4z      clrf    portb
            btfsc   tlac
            goto    $-1
stav4       movlw   b'01010101'
            morf    portb
            call    cekani
```

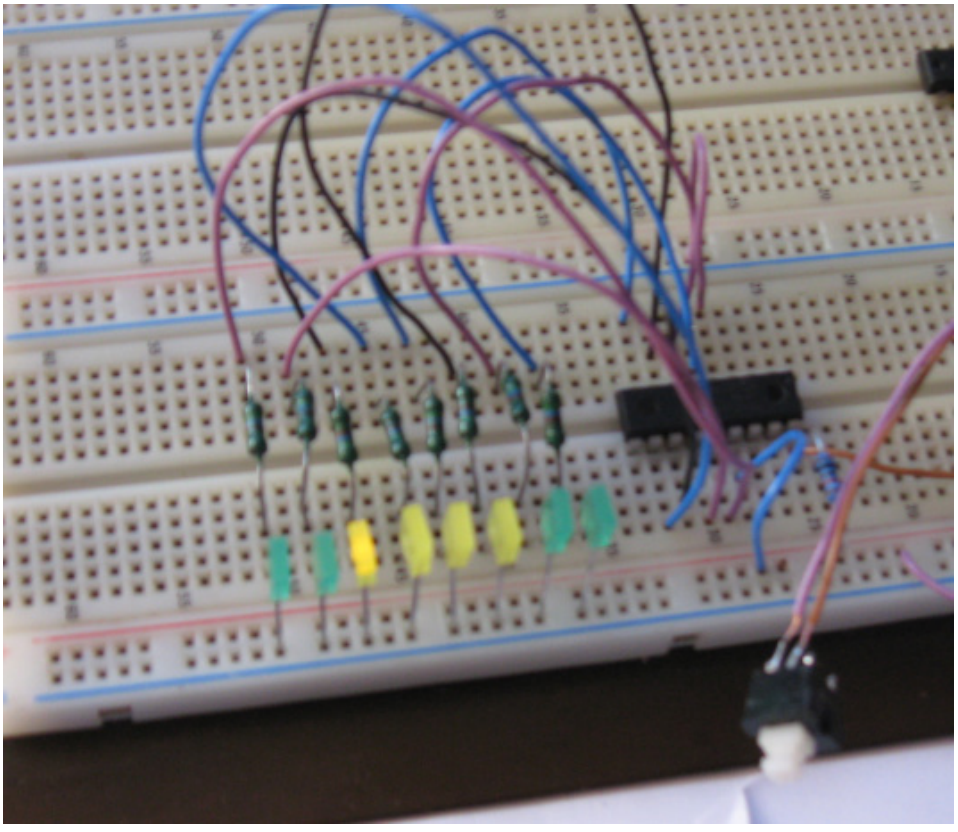
```
                btfsc    tlac
                goto     stav5

                movlw    b'10101010'
                morf     portb
                call     cekani
                btfsc    tlac
                goto     stav5z
                goto     stav4
;********************  paty stav (vse sviti) ****************
stav5z          clrf     portb
                btfsc    tlac
                goto     $-1

stav5           movlw    b'01100110'
                morf     portb
                call     cekani
                btfsc    tlac
                goto     stav1

                movlw    b'10011001'
                morf     portb
                call     cekani
                btfsc    tlac
                goto     stav1z
                goto     stav5

end
```



Obr. 40 Světelný had z LED diod za použití PIC16F628A