Osmibitové mikrořadiče Microchip

PIC16F887

SPŠE Dobruška, učební text, 2009
Ing Josef Hloušek

Tento učební text je určen pro výuku předmětu Mikropočítačové systémy ve 4. ročníku oboru Elekrotechnika. Cílem výuky je použití osmibitového mikrořadiče s pamětí Flash v jednoduché aplikaci. Pro praktickou část výuky je používáno profesionální vývojové prostředí Microchip MPLAB IDE s projektovým manažerem, textovým editorem, assemblerem a simulátorem. Hardwarová část je založena na debuggeru / programátoru PICKit 2 s demonstrační deskou osazenou mikrořadičem PIC16F887.
Text předpokládá znalost číslicové techniky, logických obvodů, binární a hexadecimální číselné soustavy. Výhodou při studiu je znalost technické angličtiny.

1. Osmibitové mikrořadiče Microchip

1.1. Základní sortiment mikrořadičů PIC

PIC 16F a PIC 18F jsou mikrořadiče s architekturou Harvard, vyráběné firmou **Microchip** od roku 1975 na základě mikrořadičů General Instruments. Původně originální, ale nepříliš ambiciózní PIC (Programmable Interface Controller), byly velmi rychle vylepšovány a cca od roku 2001 je firma Microchip vedoucí světovou firmou na trhu osmibitových mikrořadičů. Do roku 2008 firma Microchip vyrobila 6 000 000 000 mikrořadičů. Mikrořadiče PIC byly první, které v roce 1993 obsahovaly na čipu paměť EEPROM. Zvláštností mikrořadičů PIC je délka programovacího slova, které je 12, 14 nebo 16 bitů, každé slovo obsahuje vždy kompletní instrukci. Výpočetní výkon je až 10 MIPS. Jasnou výhodou je velmi široká nabídka nejrůznější variant pro velký okruh zákazníků, široký sortiment periférií včetně sběrnic USB a CAN a řadičů displejů LCD.

V České republice jsou nejrozšířenější tři řady mikrořadičů PIC :
Základní řada (Base-Line) s pamětí OTP nebo Flash až 2KB, instrukční soubor má 33 instrukcí, jako periferie jsou na čipu osmibitové čítače, osmibitový A/D převodník a komparátor. Výpočetní výkon je 5 MIPS.
Střední řada (Mid-Range) s pamětí OTP nebo Flash až 8KB, instrukční soubor má 35 instrukcí, jako periferie jsou na čipu osmibitové a šestnáctibitové čítače, desetibitový A/D převodník, komparátor, sériové komunikační kanály UART, SPI, I2C. Všechny typy mají rozhraní ICSP pro sériové programování osazených mikrořadičů na desce tištěného spoje a rozhraní ICD pro ladění programů v paměti flash. Výpočetní výkon je 5 MIPS.
Elitní řada (High-End) s pamětí Flash až 64KB, instrukční soubor má 83 instrukcí, jako periferie jsou na čipu osmibitové a šestnáctibitové čítače, desetibitový A/D převodník, komparátor, sériové komunikační kanály UART, SPI, I2C. Všechny typy mají rozhraní ICSP pro sériové programování osazených mikrořadičů na desce tištěného spoje a rozhraní ICD pro ladění programů v paměti flash. Výpočetní výkon je 10 MIPS.

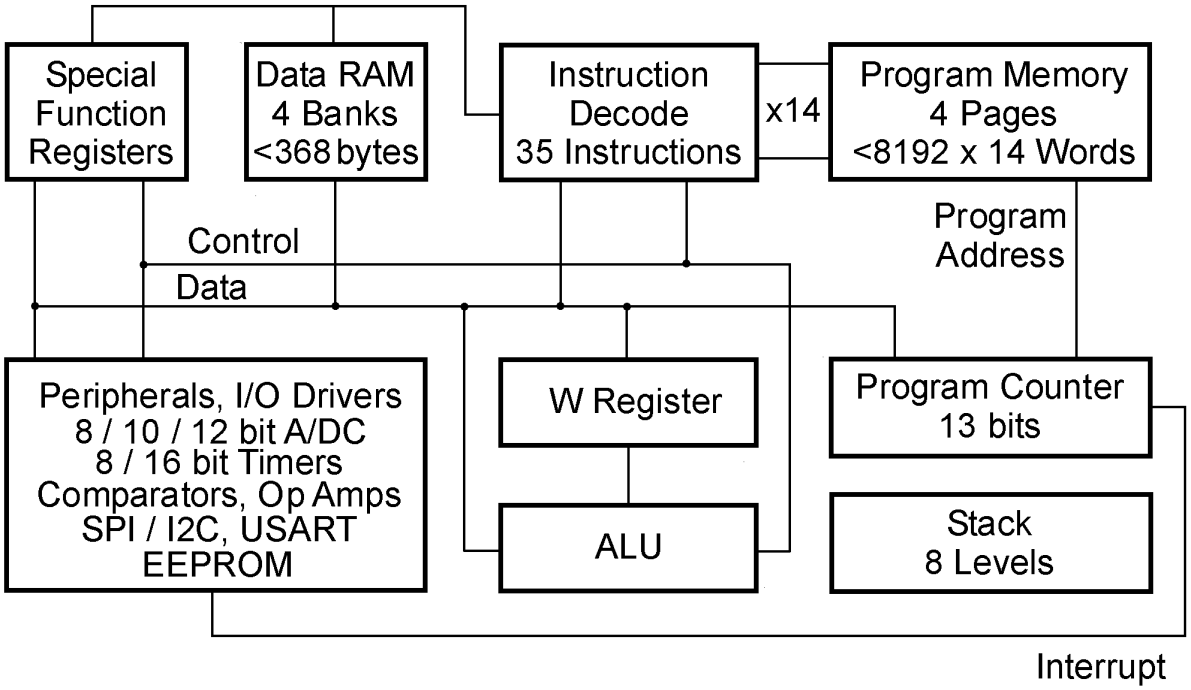
Porovnání hlavních parametrů osmibitových mikrořadičů Microchip :

řada	Flash	RAM	instrukce	výkon	čítače	ADC	UART	I / O	f
základní	2 KB	72 B	33	5 MIPS	1	8 bit	--	32	20 MHz
střední	8 KB	368 B	35	5 MIPS	3	10/12 bit	1	53	20 MHz
elitní	64 KB	4 KB	83	10 MIPS	5	10/12 bit	2	70	40 MHz

Každá řada obsahuje desítky typů a variant, všechny uvedené hodnoty jsou nejvyšší, které každá řada nabízí.

Rozsah nabídky mikrořadičů ilustruje ukázka pouzder.

1.2. Střední řada PIC16F, funkční schéma



Základní vlastnosti:

ALU (Arithmetical Logic Unit), osmibitová aritmeticko-logická jednotka, maximální taktovací kmitočet 20 MHz, výpočetní výkon 5MIPS, 35 instrukcí

W Register (Working Register), střadač, není součástí sady registrů RAM

Program Counter, 13bitový čítač programu, může vytvořit až 2^{13} = 8192 adres

Program Memory, paměť Flash pro uložení programu. Obsahuje až 8K slov, každé o délce 14 bitů. Na každé z paměťových míst je možno uložit úplnou instrukci sestávající z kódu instrukce a až dvou operandů.

Data RAM, až 512 osmibitových registrů rozdělených do čtyř bank po 128 registrech. Některé registry jsou využívány procesorem a periferiemi (SFR – Special Function Registers), ostatní registry nemají přesné určení a mohou být využity programem (GPR – General Purpose Registers).

Stack, zásobník, osm hardwarových 14bitových registrů pro uložení adres při větvení programu. Registry nejsou součástí ani paměti RAM ani Flash a nejsou přístupné programátorovi.

Peripherals, periferní hardwarové obvody, jejich sortiment je závislý na konkrétním typu mikrořadiče. Datové a konfigurační registry periférií jsou součástí sady registrů RAM.

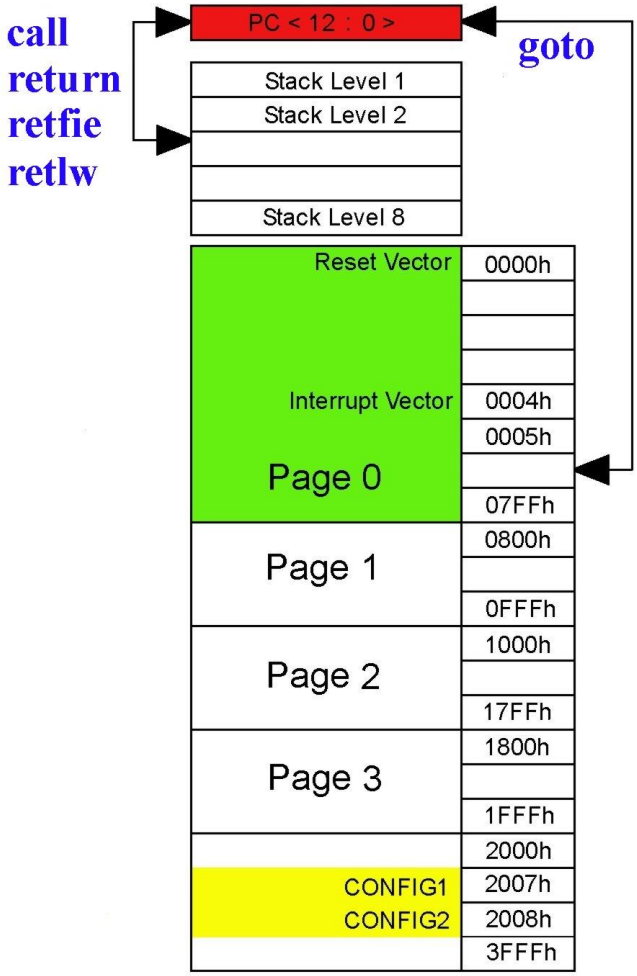
Vybrané typy mikrořadičů střední řady a jejich základní parametry:

řada	Flash	RAM	EEPROM	I /O	čítače	ADC	komp.	serial	piny
PIC12F 6xx	2 KB	128 B	256 B	6	2 + 1	4 x 10b	1	--	8
PIC16F 6xx	4 KB	256 B	256 B	18	2 + 1	12x 10b	2	USART,I2C,SPI	14~28
PIC16F 7xx	8 KB	368 B	--	36	2 + 1	14x 10b	2	USART,I2C,SPI	28~44
PIC16F 88x	8 KB	368 B	256 B	35	2 + 1	14x 10b	2	USART,I2C,SPI	28~44

2. Paměť pro program

2.1. Organizace paměti Flash

Paměť pro uložení programu je organizována jako sada 8192 registrů po14 bitech.



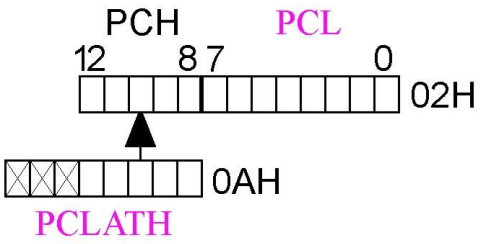
Mikrořadiče PIC16F mají 13bitový čítač programu (PC, Program Counter), který může adresovat až 8KB 14bitových slov. Všechny instrukce mají délku jednoho slova a v paměti tedy může být uloženo až 8K instrukcí.
Paměť je dělena do čtyř stránek po 2K. Při skocích mezi stránkami paměti musí být nastaveno pět nejvyšších bitů registru PC zápisem do registru PCLATH (Program Counter Latch High). Při postupném vykonávání instrukcí obsah registru PC přejde přes hranice stránek samovolně. Paměť menší než 2K nevyžaduje stránkování.
Registr PC je složen z registrů PCL (Program Counter Least Significant Byte) a **PCH**. Detaily o registru PC viz článek 2.2.
Některé typy mikrořadičů mají menší paměť než 8K. Je-li v takovém mikrořadiči adresována paměť, která není na čipu implementována, je adresována paměť se „shodnou“ adresou na nižší stránce, např. místo adresy 17FFH je adresována paměť 7FFH.
První instrukce uživatelského programu by měla být uložena na adrese 0005H nebo vyšší.

Signál RESET vždy nastaví obsah PC na hodnotu 0000H, na této adrese je uložen Reset Vector.

Potvrzené přerušení nastavuje obsah PC na adresu 0004H, na této adrese je uložen Interrupt Vector. O požadavcích na přerušení a jejich obsloužení viz článek 9.Přerušovací systém.

Stack (zásobník) je tvořen osmi 13bitovými hardwarovými registry. Zásobník je používán při instrukcích CALL a RETURN pro uložení a vyzvednutí návratových adres. Zásobník není přístupný uživateli, instrukční soubor neobsahuje instrukce POP a PUSH. Funkce zásobníku je popsána v článku 2.3. Zásobník.

2.2. Čítač programu (PC, Program Counter)



Čítač programu má délku 13 bitů a je složen z 8bitového registru PCL (Program Counter Least Significant Byte) a 5bitového hardwarového registru **PCH**.
Registr PCL je součástí sady registrů (viz paměť RAM), má adresu 02H a je určen pro zápis i čtení. Jakýkoliv zápis do registru PCL zapíše také obsah registru PCLATH do registru **PCH**.

Registr **PCH** není uživateli přístupný, jeho obsah se mění zápisem do registru PCLATH
Registr PCLATH (Program Counter Latch High) je součástí sady registrů RAM, adresa je 0AH.

Příklad pro nastavení hodnoty registru PCLATH :

```
bsf     PCLATH, 3    ; nastaví třetí bit registru PCLATH na hodnotu 1
```

Instrukce **bsf** (Bit Set f) nastaví označený bit registru **f** na hodnotu 1.
Třetí bit registru PCLATH se zapíše jako jedenáctý bit registru PC, obsah registru PC bude tedy 0800H a vyšší což je stránka 1 (Page 1) paměti Flash.

Obsah registru PC je nulován signálem RESET.

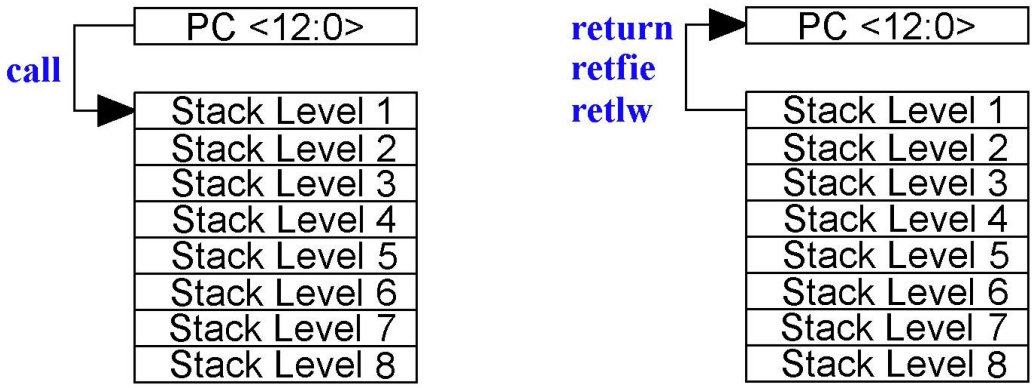
Obsah registru PC je ukládán na vrchol zásobníku (Stack Level 1) při instrukci **call**, tj. při volání podprogramu.

Obsah registru PC je naplněn z vrcholu zásobníku při instrukcích **return** , **retfie** a **retlw**, tj. při návratu z podprogramu nebo z obslužné rutiny přerušení.

Nižších 11 bitů registru PC je naplněno 11bitovou adresou, kterou obsahují instrukce **call** pro volání podprogramu a **goto** pro nepodmíněný skok. Tak jsou umožněny skoky programu uvnitř jedné stránky o velikosti 2K. Pro skoky na jinou stránku je nutno nastavit i nejvyšší dva bity registru PC zápisem bitů 4 a 5 do registru PCLATH.

2.3. Zásobník (Stack)

Zásobník je tvořen osmi 13bitovými hardwarovými registry, které nejsou přístupné uživateli. Zásobník umožňuje až osm větvení programu instrukcemi **call** nebo obsluhou přerušení.



Do zásobníku je ukládán obsah čítače programu PC jako návratová adresa při instrukci **call**, obsahem zásobníku je naplněn čítač programu při instrukcích **return** , **retfie** a **retlw**. Do zásobníku je možno zapsat nejvýše osm adres, devátý zápis přepíše adresu, která byla zapsána jako první.
Uživatel (programátor) nemá k dispozici žádné instrukce typu PUSH nebo POP pro uložení do zásobníku nebo pro vyzvednutí ze zásobníku.

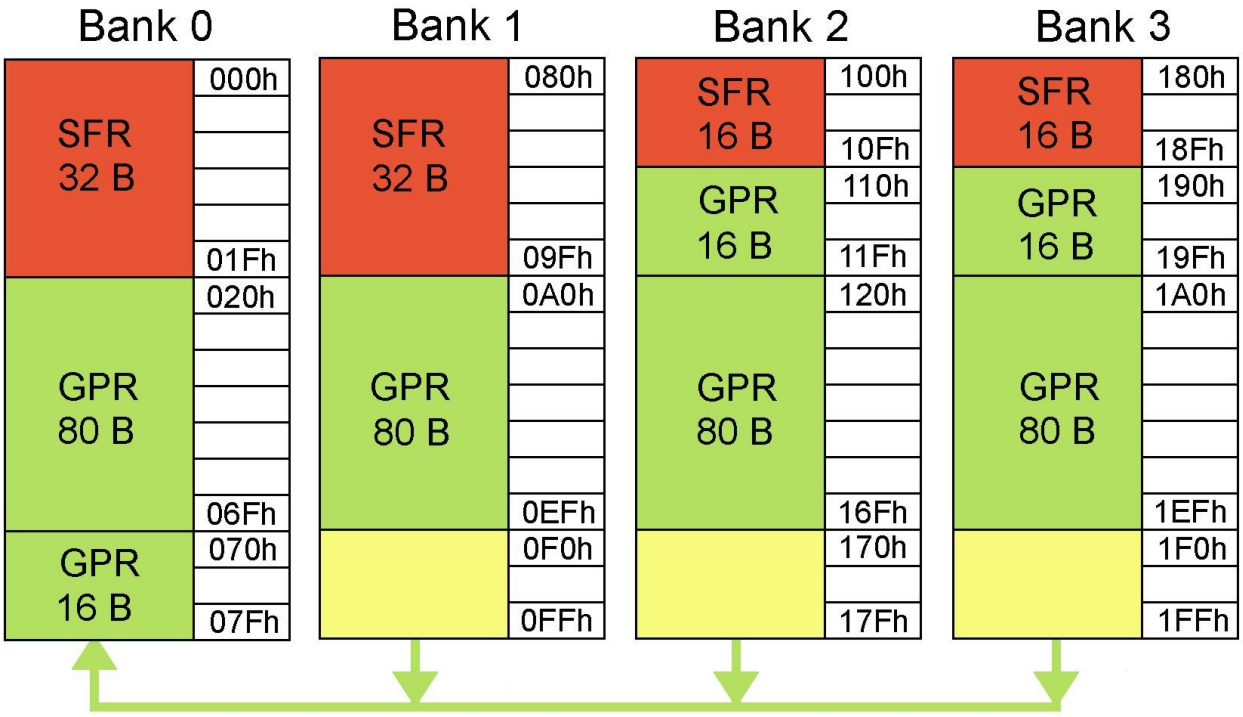
Mikrořadiče PIC16F nemají k dispozici žádnou informaci (např. jako status bit) o naplnění zásobníku. Programátor musí zajistit, aby do zásobníku bylo zapisováno nejvýše osmkrát, poté musí následovat čtení zásobníku. Instrukce, které do zásobníku zapisují, musejí být vždy následovány shodným počtem instrukcí, které ze zásobníku čtou. Nedodržení tohoto pravidla má za následek chybnou funkci programu.

3. Paměť RAM

3.1. Paměť RAM, organizace

Paměť pro uložení dat je implementována jako statická RAM. Paměť může obsahovat až 512 osmibitových registrů. Operandy instrukcí však mohou obsahovat nejvýše 7 adresních bitů. Paměť je proto rozdělena do čtyř bloků (Bank 0, 1, 2 a 3) po 128 registrech.
V paměti jsou dva typy registrů:
Registry SFR (Special Function Register) jsou využívány procesorem a periferiemi a jsou implementovány na nejnižších 32 nebo 16 adresách v každé bance. Některé registry SFR jsou signálem RESET uvedeny do definovaného stavu.
Registry GPR (General Purpose Register) jsou k dispozici uživateli pro manipulaci s daty. V bance může být nejvýše 96 GPR registrů, jsou implementovány na vyšších adresách v bance. Signál RESET nemění obsah registrů GPR, po zapnutí napájecího napětí je tedy stav registrů GPR náhodný.
Šestnáct registrů s nejvyšší adresou v bankách 1, 2 a 3 je možné adresovat, z adresy je ale respektováno pouze 7 nejnižších bitů takže jsou vždy adresovány registry s adresami 070h až 07Fh v Bance 0. Registry s adresami 070h až 07FH jsou tak přístupné ze všech Bank.

Umístění nejvyššího možného počtu registrů v paměti RAM :



3.2. Registry SFR (Special Function Register)

Bank 0

adresa	symbol	název	stav po reset
00h	INDF	Indirect Addressing	x x x x x x x x
01h	TMR0	Timer 0 Module Register	x x x x x x x x
02h	PCL	Program Counter Least Significant Byte	0 0 0 0 0 0 0 0
03h	STATUS	Status Register	0 0 0 1 1 x x x
04h	FSR	File Select Register	x x x x x x x x
05h	PORTA	Port A Register	x x x x x x x x
06h	PORTB	Port B Register	x x x x x x x x
07h	PORTC	Port C Register	x x x x x x x x
08h	PORTD	Port D Registers	x x x x x x x x
09h	PORTE	Port E Register	- - - - x x x x
0Ah	PCLATH	Program Counter Latch High	- - - 0 0 0 0 0
0Bh	INTCON	Interrupt Control Register	0 0 0 0 0 0 0 x
0Ch	PIR1	Peripheral Interrupt Request Register 1	- 0 0 0 0 0 0 0
0Dh	PIR2	Peripheral Interrupt Request Register 2	0 0 0 0 0 0 - 0
0Eh	TMR1L	Timer 1 Module Low byte	x x x x x x x x
0Fh	TMR1H	Timer 1 Module High byte	x x x x x x x x
10h	T1CON	Timer 1 Control Register	0 0 0 0 0 0 0 0
11h	TMR2	Timer 2 Module register	0 0 0 0 0 0 0 0
12h	T2CON	Timer 2 Control Register	- 0 0 0 0 0 0 0
13h	SSPBUF	Synchronous Seriál Port Buffer Register	x x x x x x x x
14h	SSPCON	Synchronous Seriál Port Control Register	0 0 0 0 0 0 0 0
15h	CCPR1L	Capture / Compare / PWM 1 Low byte	x x x x x x x x
16h	CCPR1H	Capture / Compare / PWM 1 High byte	x x x x x x x x
17h	CCP1CON	Capture / Compare / PWM 1 Control Register	0 0 0 0 0 0 0 0
18h	RCSTA	Receive Status and Control Register	0 0 0 0 0 0 0 x
19h	TXREG	EUSART Transmit Data Register	0 0 0 0 0 0 0 0
1Ah	RCREG	EUSART Receive Data Register	0 0 0 0 0 0 0 0
1Bh	CCPR2L	Capture / Compare / PWM 2 Low byte	x x x x x x x x
1Ch	CCPR2H	Capture / Compare / PWM 2 High byte	x x x x x x x x
1Dh	CCP2CON	Capture / Compare / PWM 2 Control Register	- - 0 0 0 0 0 0
1Eh	ADRESH	A/D Result Register High byte	x x x x x x x x
1Fh	ADCON0	A/D Control Register 0	0 0 0 0 0 0 0 0

0 = hodnota bitu po Reset je 0
1 = hodnota bitu po Reset je 1
x = hodnota bitu po Reset není definována, může být 0 nebo 1
- = bit není využíván

Bank 1

adresa	symbol	název	stav po reset
80h	INDF	Indirect Addressing	x x x x x x x x
81h	OPTION_REG	Option Register	1 1 1 1 1 1 1 1
82h	PCL	Program Counter Least Significant Byte	0 0 0 0 0 0 0 0
83h	STATUS	Status Register	0 0 0 1 1 x x x
84h	FSR	File Select Register	x x x x x x x x
85h	TRISA	Port A Tri-State Register	1 1 1 1 1 1 1 1
86h	TRISB	Port B Tri-State Register	1 1 1 1 1 1 1 1
87h	TRISC	Port C Tri-State Register	1 1 1 1 1 1 1 1
88h	TRISD	Port D Tri-State Registers	1 1 1 1 1 1 1 1
89h	TRISE	Port E Tri-State Register	- - - - 1 1 1 1
8Ah	PCLATH	Program Counter Latch High	- - -0 0 0 0 0
8Bh	INTCON	Interrupt Control Register	0 0 0 0 0 0 0 x
8Ch	PIE1	Peripheral Interrupt Enable Register 1	- 0 0 0 0 0 0 0
8Dh	PIE2	Peripheral Interrupt Enable Register 2	0 0 0 0 0 0 - 0
8Eh	PCON	Power Control Register	- - 0 1 - - q q
8Fh	OSCCON	Oscillator Control Register	- 1 1 0 q 0 0 0
90h	OSCTUNE	Oscillator Tuning Register	- - - 0 0 0 0 0
91h	SSPCON2	Synchronous Seriál Port Control Register 2	0 0 0 0 0 0 0 0
92h	PR2	Timer 2 Period Register	1 1 1 1 1 1 1 1
93h	SSPADD	Synchronous Seriál Port Addres Register	0 0 0 0 0 0 0 0
94h	SSPSTAT	Synchronous Seriál Port Status Register	0 0 0 0 0 0 0 0
95h	WPUB	Weak Pull-Up Port B Register	1 1 1 1 1 1 1 1
96h	IOCB	Interrupt –On- Change Port B Register	0 0 0 0 0 0 0 0
97h	VRCON	Voltage Reference Control Register	0 0 0 0 0 0 0 0
98h	TXSTA	Transmit Status and Control Register	0 0 0 0 0 0 1 0
99h	SPBRG	Set Period Baud Rate Generator	0 0 0 0 0 0 0 0
9Ah	SPBRGH	Set Period Baud Rate Generator Hihg byte	0 0 0 0 0 0 0 0
9Bh	PWM1CON	PWM 1 Control Register	0 0 0 0 0 0 0 0
9Ch	ECCPAS	Enhanced Capture/Compare/PWM Auto-Shutdown	0 0 0 0 0 0 0 0
9Dh	PSTRCON	Pulse Steering Control Register	- - - 0 0 0 0 1
9Eh	ADRESL	A/D Result Register Low byte	x x x x x x x x
9Fh	ADCON1	A/D Control Register 1	0 - 0 0 - - - -

0 = hodnota bitu po Reset je 0
1 = hodnota bitu po Reset je 1
x = hodnota bitu po Reset není definována, může být 0 nebo 1
q = hodnota bitu po Reset je závislá na podmínkách
- = bit není implementován, čtení bitu vrací hodnotu 0

Bank 2

adresa	symbol	název	stav po reset
100h	INDF	Indirect Addressing	x x x x x x x x
101h	TMR0	Timer 0 Module Register	x x x x x x x x
102h	PCL	Program Counter Least Significant Byte	0 0 0 0 0 0 0 0
103h	STATUS	Status Register	0 0 0 1 1 x x x
104h	FSR	File Select Register	x x x x x x x x
105h	WDTCON	Watch Dog Control Register	- - - 0 1 0 0 0
106h	PORTB	Port B Register	x x x x x x x x
107h	CM1CON0	Comparator 1 Control Register 0	0 0 0 0 - 0 0 0
108h	CM2CON0	Comparator 2 Control Register 0	0 0 0 0 - 0 0 0
109h	CM2CON1	Comparator 2 Control Register 1	0 0 0 0 - - 1 0
10Ah	PCLATH	Program Counter Latch High	- - - 0 0 0 0 0
10Bh	INTCON	Interrupt Control Register	0 0 0 0 0 0 0 x
10Ch	EEDAT	EEPROM Data Register	0 0 0 0 0 0 0 0
10Dh	EEADR	EEPROM Address Register	0 0 0 0 0 0 0 0
10Eh	EEDATH	EEPROM Data High byteRegister	- - 0 0 0 0 0 0
10Fh	EEADRH	EEPROM Address High byteRegister	- - - - 0 0 0 0

Bank 3

adresa	symbol	název	stav po reset
180h	INDF	Indirect Addressing	x x x x x x x x
181h	OPTION_REG	Option Register	1 1 1 1 1 1 1 1
182h	PCL	Program Counter Least Significant Byte	0 0 0 0 0 0 0 0
183h	STATUS	Status Register	0 0 0 1 1 x x x
184h	FSR	File Select Register	x x x x x x x x
185h	SRCON	SR Latch Control Register	0 0 0 0 0 0 - 0
186h	TRISB	Port B Tri-State Register	1 1 1 1 1 1 1 1
187h	BAUDCTL	Baud Rate Control Register	0 1 - 0 0 - 0 0
188h	ANSEL	Analog Select Register	1 1 1 1 1 1 1 1
189h	ANSELH	Analog Select Register High byte	- - 1 1 1 1 1 1
18Ah	PCLATH	Program Counter Latch High	- - - 0 0 0 0 0
18Bh	INTCON	Interrupt Control Register	0 0 0 0 0 0 0 x
18Ch	EECON1	EEPROM Control Register 1	x - - - x 0 0 0
18Dh	EECON2	EEPROM Control Register 2	- - - - - - - -
18Eh		reserved	
18Fh		reserved	

0 = hodnota bitu po Reset je 0
1 = hodnota bitu po Reset je 1
x = hodnota bitu po Reset není definována, může být 0 nebo 1
- = bit není implementován, čtení bitu vrací hodnotu 0

3.3. STATUS registr

STATUS Status Register								adresa
R / W	R / W	R / W	R	R	R	R	R	03 H Bank 0
IRP	RP1	RP0	NOT_TO	NOT_PD	Z	DC	C	
0	0	0	1	1	x	x	x	
7	6	5	4	3	2	1	0	bit

IRP **Indirect Register Pointer (**Bank Select Bit, použit při nepřímém adresování)
0 = Bank 0,1 (000H – 0FFH)
1 = Bank 2,3 (100H – 1FFH)
RP1 , RP0 **Register Pointer** (Bank Select, použity při přímém adresování)
00 = Bank 0 (000H – 07FH)
01 = Bank 1 (080H – 0FFH)
10 = Bank 2 (100H – 17FH)
11 = Bank 3 (180H – 1FFH)
NOT_TO **Time-Out** Status Bit
0 = naplněn interval časovače WDT (Watch Dog Timer)
1 = po zapnutí napájení nebo po instrukcích **clrwdt** nebo **sleep**
NOT_PD **Pover Down** Status Bit
0 = po instrukci **sleep**
1 = po zapnutí napájení nebo po instrukci **clrwdt**
Z **Zero** Flag Bit
0 = výsledek logické nebo aritmetické operace není 0
1 = výsledek logické nebo aritmetické operace je 0
DC **Decimal Carry** Flag Bit
0 = výsledek operace bez přenosu ze 4 bitu
1 = výsledek operace s přenosem ze 4 bitu
Pro instrukce **addwf** , **addlw** , **subwf** , **sublw**. Při výpůjčce (Borrow) je bit DC reverzní.
C **Carry** Flag Bit
0 = výsledek operace bez přenosu z nejvyššího bitu
1 = výsledek operace s přenosem z nejvyššího bitu
Pro instrukce **addwf** , **addlw** , **subwf** , **sublw**. Při výpůjčce (Borrow) je bit C reverzní.

STATUS registr může být použit jako cílový registr v mnoha instrukcích. Výsledek zápisu může být však neočekávaný. Např. instrukce **clrf** STATUS zapíše 0 do bitů IRP, RP1 a RP0 a nastaví příznak Z na hodnotu 1. Stav registru po instrukci je potom 0 0 0 u u 1 u u (u= hodnota bitu se nemění). Proto se doporučuje používat pro nastavení registru STATUS jen instrukce **bcf** , **bsf** , **movwf** a **swapf** .

Příklad přepínání Bank:

```
                                ; po Resetu je RP0 = RP1 = 0, je nastavena Banka 0
        :
        bsf     STATUS, RP0     ; nastaví bit RP0 = Banka 1
        :
        bcf     STATUS, RP0     ; nuluje bit RP0 = Banka 0
        :
```

3.4. Registry PCL a PCLATH

PCL Program Counter Least Significant Byte								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	02 H Bank 0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
0	0	0	0	0	0	0	0	

PCLATH Program Counter Latch High								adresa
--	--	--	R / W	R / W	R / W	R / W	R / W	0A H Bank 0
--	--	--	bit 4	bit 3	bit 2	bit 1	bit 0	
--	--	--	0	0	0	0	0	

Registr PCL slouží jako nejnižších 8 bitů čítače programu.
Registr PCLATH slouží pro zápis do horních 5 bitů čítače programu.
Funkce obou registrů je popsána v článku 2.2.

Příklad volání podprogramu na stránce 1 ze stránky 0 :

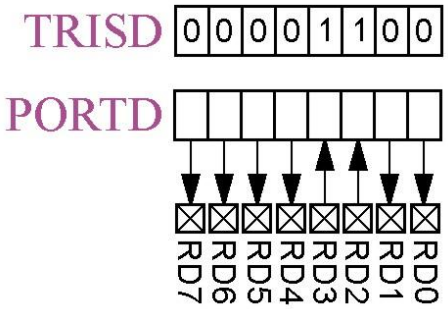
```
        org     500         ; stránka 0
        bsf     PCLATH, 3   ; nastaví bit 3 v registru PCLATH (= bit 11 v registru PC)
                            ; stránka 1 (0800H – 0FFFH)
        call    Sub1_P1     ; volá podprogram Sub1_P1 na stránce 1
        :
        org     900         ; stránka 1
Sub1_P1                     ; návěští (Label) volaného podprogramu
        :
        return              ; návrat z podprogramu
```

4. I / O porty

Mikrořadiče PIC16F88x mají čtyři osmibitové porty a jeden čtyřbitový vstupně / výstupní port, pro aplikaci je k dispozici až 36 vstupů / výstupů. Některé z I/O pinů jsou používány vnitřními periferiemi (čítače, A/D převodníky) a pokud jsou tyto periferie povoleny, jim přiřazené piny nemohou být použity pro jiný účel.

4.1. Port D

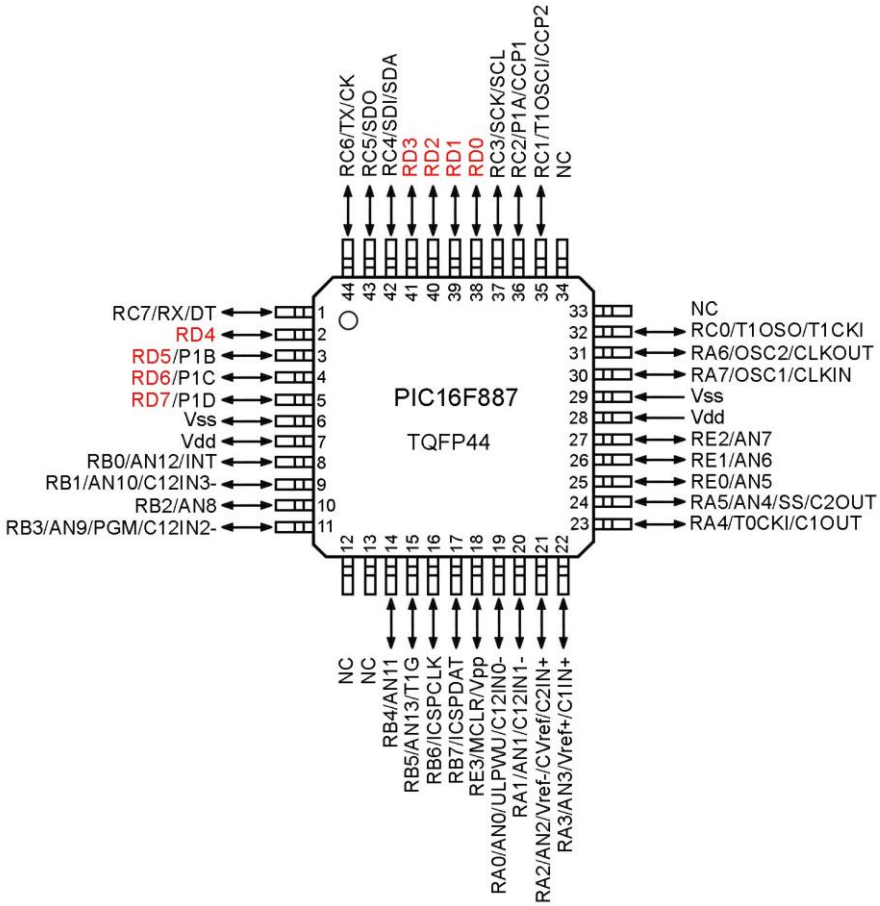
Port D je osmibitový obousměrný port. Směr přenosu dat nastavuje registr TRISD.
Nastavením bitu v registru TRISD na hodnotu 1 je odpovídající pin portu D nastaven jako vstup a hodnota pinu je uložena do odpovídajícího bitu registru PORTD. Po signálu Reset jsou všechny piny portu D nastaveny jako vstupy.
Nastavením bitu v registru TRISD na hodnotu 0 je odpovídající pin portu D nastaven jako výstup a hodnota odpovídajícího bitu registru PORTD je vyvedena na pin.



Konfigurace portu D

```
banksel   PORTD
clrf      PORTD
banksel   TRISD
movlw     B'00001100'   ; nastaví piny 3 a 2 jako vstupy  a piny 7 : 4 , 1 : 0 jako výstupy
movwf     TRISD
```

Piny portu D
v pouzdře TQFP44



PORTD Port D Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	08 H Bank 0
RD7	RD 6	RD 5	RD 4	RD 3	RD 2	RD 1	RD 0	
x	x	x	x	x	x	x	x	

TRISD Tri-State D Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	88 H Bank 1
TRISD7	TRISD 6	TRISD 5	TRISD 4	TRISD 3	TRISD 2	TRISD 1	TRISD 0	
1	1	1	1	1	1	1	1	

TRISD **< 7 : 0 > Port D Tri-State Control Bit**
0 = pin konfigurován jako výstup
1 = pin konfigurován jako vstup

Piny RD5, RD6 a RD7 mohou být konfigurovány pro výstup signálu PWM v registru PSTRCON. Funkce není v tomto učebním textu využívána.

4.2. Port A

Port A je osmibitový obousměrný port. Směr přenosu dat nastavuje registr TRISA.
Nastavením bitu v registru TRISA na hodnotu 1 je odpovídající pin portu A nastaven jako vstup a hodnota pinu je uložena do odpovídajícího bitu registru PORTA. Po signálu Reset jsou všechny piny portu A nastaveny jako **analogové** vstupy. Pro nastavení digitálních vstupů musí být nakonfigurován registr ANSEL
Nastavením bitu v registru TRISA na hodnotu 0 je odpovídající pin portu A nastaven jako výstup a hodnota odpovídajícího bitu registru PORTA je vyvedena na pin. Pro nastavení digitálních výstupů musí být nakonfigurován registr ANSEL

Konfigurace portu A jako digitální I/O

```
        banksel     PORTA
        clrf        PORTA       ;nuluje PORTA
        banksel     ANSEL
        clrf        ANSEL       ;digitální I/O
        banksel     TRISA
        movlw       0C          ;nastaví piny 3 a 2 jako vstupy
        movwf       TRISA       ;a piny 5 : 4 , 1 : 0 jako výstupy
```

PORTA Port A Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	05 H Bank 0
RA7	RA 6	RA 5	RA 4	RA 3	RA 2	RA 1	RA 0	
x	x	x	x	x	x	x	x	

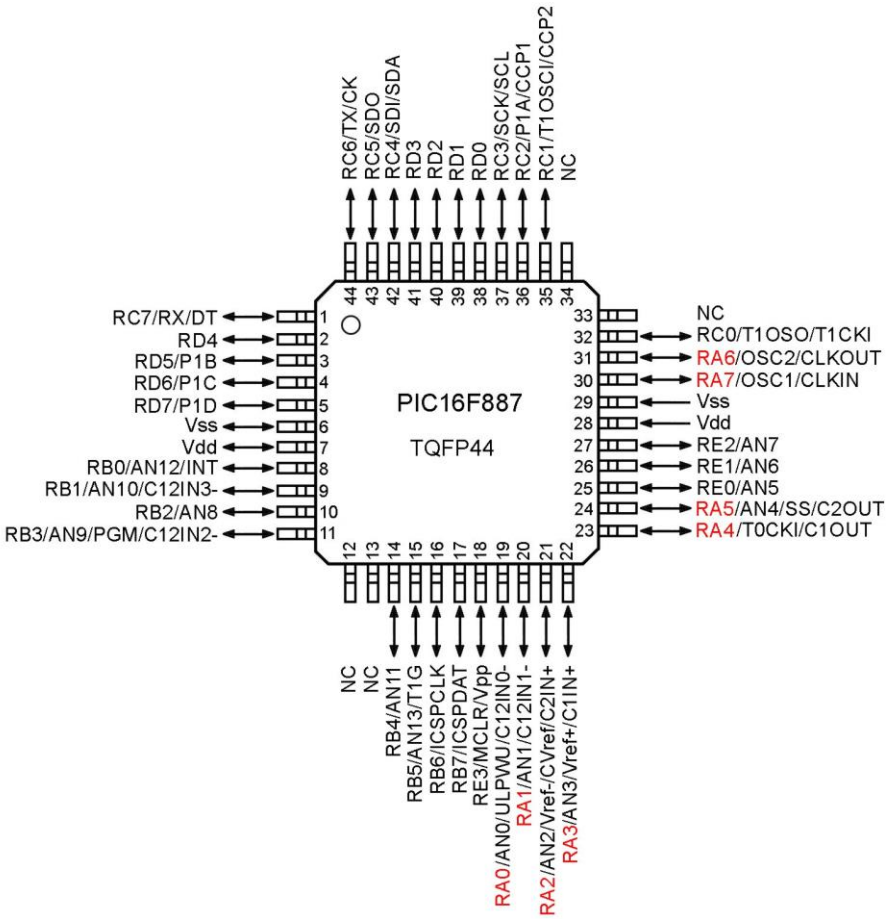
TRISA Tri-State A Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	85 H Bank 1
TRISA7	TRISA 6	TRISA 5	TRISA 4	TRISA 3	TRISA 2	TRISA 1	TRISA 0	
1	1	1	1	1	1	1	1	

TRISA **< 7 : 0 > Port A Tri-State Control Bit**
0 = pin konfigurován jako výstup
1 = pin konfigurován jako vstup

ANSEL Analog Select Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	188 H Bank 3
ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	
1	1	1	1	1	1	1	1	

ANS7 : ANS0 **Analog Select Bit**
0 = pin konfigurován jako digitální vstup / výstup
1 = pin konfigurován jako analogový vstup, bit registru TRISA musí být nastaven na 1

Piny portu A
v pouzdře TQFP44



Piny portu A mohou být konfigurovány na řadu speciálních funkcí. Konfigurace je popsána v příslušných kapitolách.
Analogové vstupy A/D převodníku jsou využity v praktickém cvičení na PICKit2 kde je detailně vysvětlena potřebná konfigurace..
Konfigurace pinů portu A pro analogové komparátory a další funkce není v tomto učebním textu využívána.

4.3. Port B

Port B je osmibitový obousměrný port. Směr přenosu dat nastavuje registr TRISB.
Nastavením bitu v registru TRISB na hodnotu 1 je odpovídající pin portu B nastaven jako vstup a hodnota pinu je uložena do odpovídajícího bitu registru PORTB. Po signálu Reset jsou piny < 5 : 0 > portu B nastaveny jako **analogové** vstupy. Pro nastavení digitálních vstupů musí být nakonfigurován registr ANSELH
Nastavením bitu v registru TRISB na hodnotu 0 je odpovídající pin portu B nastaven jako výstup a hodnota odpovídajícího bitu registru PORTB je vyvedena na pin. Pro nastavení digitálních výstupů musí být nakonfigurován registr ANSELH

PORTB Port B Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	06 H Bank 0
RB7	RB 6	RB 5	RB 4	RB 3	RB 2	RB 1	RB 0	
x	x	x	x	x	x	x	x	

TRISB Tri-State B Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	86 H Bank 1
TRISB7	TRISB 6	TRISB 5	TRISB 4	TRISB 3	TRISB 2	TRISB 1	TRISB 0	
1	1	1	1	1	1	1	1	

TRISB **< 7 : 0 > Port B Tri-State Control Bit**
0 = pin konfigurován jako výstup
1 = pin konfigurován jako vstup

ANSELH Analog Select High Register								adresa
--	--	R / W	R / W	R / W	R / W	R / W	R / W	189 H Bank 3
--	--	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8	
--	--	1	1	1	1	1	1	

ANS13 **:** ANS8 **Analog Select Bit**
0 = pin konfigurován jako digitální vstup / výstup
1 = pin konfigurován jako analogový vstup, bit registru TRISB musí být nastaven na 1

Na všechny piny portu B jsou připojeny vnitřní Pull-Up rezistory, které při použití pinu jako vstup definují hodnotu 1 na nepřipojeném vstupu např. při použití spínacího tlačítka. Pull-Up rezistory jsou konfigurovány v registru WPUB.

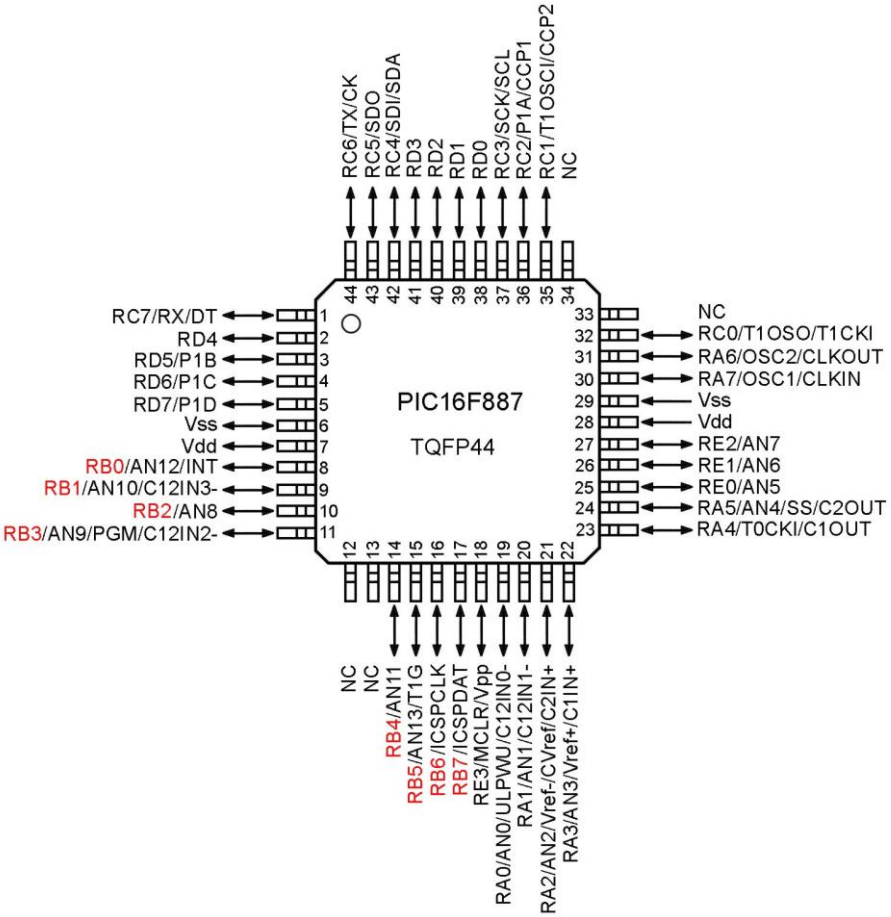
WPUB Weak Pull-Up Port B Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	95 H Bank 1
WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	
1	1	1	1	1	1	1	1	

WPUB7 **:** WPUB0 **Weak Pull-Up Port B Bit**
0 = Pull-Up rezistor odpojen
1 = Pull-Up rezistor připojen
Pull-Up rezistory jsou odpojeny při konfiguraci pinů jako výstup.

Pin **RB0** může být konfigurován jako vstup **INT** pro vstup vnějšího signálu přerušení. Tato funkce je využívána v praktickém cvičení na PICKit2 kde je detailně vysvětlena potřebná konfigurace.

Piny **RB7** a **RB6** portu B jsou využívány pro signály rozhraní **ICSP** (In Circuit Seriál Programing) při ladění programu.

Piny portu B
v pouzdře TQFP44



5. Konfigurační bity

Mikrořadiče PIC16F887 obsahují dva šestnáctibitové registry, jejichž obsah nastavuje některé parametry mikrořadiče. Registry _CONFIG1 a _CONFIG2 jsou umístěny v paměti Flash na adresách 2007H a 2008H (viz. čl. 2.1.), nejsou běžně dostupné a jejich obsah je možné změnit jen při programování paměti Flash.
Ve cvičných programech na PICKit2 je obsah registrů nastavován direktivou __config.

_CONFIG1 Config 1 Register								adresa
15	14	13	12	11	10	9	8	2007 H Flash
--	--	DEBUG	LVP	FCMEN	IESO	BOREN1	BOREN0	
1	1	1	1	1	1	1	1	

_CONFIG1 Config 1 Register								adresa
7	6	5	4	3	2	1	0	2007 H Flash
CPD	CP	MCLRE	PVRTE	WDTE	FOSC2	FOSC1	FOSC0	
1	1	1	1	1	1	1	1	

DEBUG In-Circut Debugger Mode bit – povolení ladicího režimu
1 = ladicí režim zakázán, piny RB6 a RB7 mohou být použity jako I/O piny
0 = ladicí režim povolen, piny RB6/ICSPCLK a RB7/ICSPDAT používá debugger
Ve cvičných projektech na PICKIT2 je ladicí režim povolen vývojovým prostředím MPLAB a nemusí být nastavován v programu.

LVP Low Voltage Programming Enable bit – povolení režimu programování nízkým napětím
1 = pin RB3 je určen pro programování, programování nízkým napětím povoleno
0 = pin RB3 je digitální I/O, standardní programování na pinu MCLR
Ve cvičných projektech na PICKIT2 je použito standardní programování, ***bit LVP musí být 0***

FCMEN Fail-Safe Clock Monitor Enabled bit – povolení záložního oscilátoru
1 = záložní oscilátor povolen
0 = záložní oscilátor zakázán
Ve cvičných projektech na PICKIT2 není bit FCMEM programován a mikrořadič je tak automaticky přepnut na vnitřní RC oscilátor s kmitočtem 4 MHz.
V pozdějších projektech je bit FCMEN nulován a je nastaven vnitřní RC oscilátor pomocí konfiguračních bitů FOSC2 : FOSC0

IESO Internal / External Switchover bit – přepínání vnitřní / vnější oscilátor
1 = přepínání oscilátorů povoleno
1 = přepínání oscilátorů zakázáno
Přepínání oscilátorů je využíváno v aplikacích s častým vstupem mikrořadiče do úsporného režimu (Sleep Mode), ve kterém je vypínán vnější oscilátor. Probuzení z režimu Sleep pak probíhá taktováním z vnitřního oscilátoru.
Ve cvičných projektech na PICKIT2 není tento režim použit a bit IESO má být nulován.

BOREN1 : BOREN0 Brown-out Reset bit – Reset při poklesu napájecího napětí
11 = Brown-out Reset povolen
10 = Brown-out Reset povolen v režimu Run, zakázán v režimu Sleep
01 = Brown-out Reset nastaven bitem SBOREN v registru PCON
00 = Brown-out Reset zakázán
Ve cvičných projektech na PICKIT2 není Brown-out Reset používán, oba bity mají být nulovány.

CPD Code Protection Data bit – ochrana paměti pro data proti zápisu
1 = paměť dat není chráněna proti přepsání
0 = paměť dat je chráněna proti přepsání
Ve cvičných projektech na PICKIT2 je paměť přepisována, ***bit CPD musí být 1****.*

CP Code Protection bit – ochrana paměti pro program proti zápisu
1 = paměť programu není chráněna proti přepsání
0 = paměť programu je chráněna proti přepsání
*Ve cvičných projektech na PICKIT2 je paměť přepisována****, bit CP musí být 1****.*

MCLRE MCLR pin function bit – funkce pinu RE3 / MCLR
1 = pin RE3 je konfigurován jako vstup vnějšího přerušení
0 = pin RE3 je digitální vstup
Ve cvičných projektech na PICKIT2 není přerušení využíváno, bit MCLRE má být 0.

PWRTE Power-Up Timer Enable bit – zpoždění programu při zapnutí napájení
1 = zpoždění zakázáno
0 = zpoždění povoleno
Při zapnutí napájení může být program spuštěn se zpožděním cca 64ms než se napájecí napětí ustálí. Ve cvičných projektech na PICKIT2 není hodnota bitu PWRTE důležitá.

WDTE Watchdog Timer Enable bit – povolení časovače WDT
1 = časovač WDT povolen
0 = časovač WDT zakázán
Ve cvičných projektech na PICKIT2 nesmí běžet časovač WDT, ***bit WDTE musí být 0***

FOSC2 : FOSC0 Oscillator Selection bits – výběr oscilátoru
111 = RC oscilátor, RC člen na pinu RA7, výstup oscilátoru na pinu RA6
110 = RCIO oscilátor, RC člen na pinu RA7, pin RA6 jako vstup / výstup
101 = INTOSC oscilátor, pin RA7 jako vstup / výstup, výstup oscilátoru na pinu RA6
100 = INTOSCIO oscilátor, pin RA7 jako vstup / výstup, pin RA6 jako vtup / výstup
011 = vnější oscilátor, vstup na pinu RA7, pin RA6 jako vstup / výstup
010 = HF krystalový oscilátor 1 MHz až 20 MHz, krystal na pinech RA7 a RA6
001 = XT oscilátor 100 kHz až 4 MHz, krystal nebo keramický rezonátor na pinech RA7 a RA6
000 = LP oscilátor 32 kHz, krystal na pinech RA7 a RA6
Ve cvičných projektech na PICKIT2 je používán vnitřní RC oscilátor, ***bity FOSC musí být 100***

Pokud není registr _CONFIG1 naprogramován, je hodnota všech jeho bitů rovna 1. V úvodních projektech na PICKit2 požaduje integrované prostředí MPLAB nulování bitů LVP a WDTE a tento požadavek musí být potvrzen.
V dalších projektech mají být konfigurační bity nastaveny v úvodu programu direktivou __config.

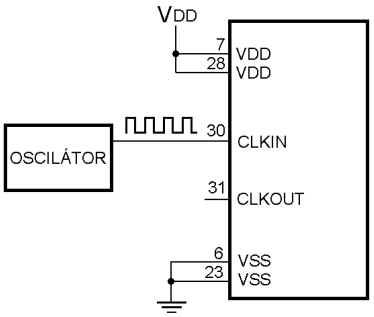
```
__config  _CONFIG1, 2FF4        ; nastaví LVP=0, zakáže programování LVP
                                ; nastaví WDTE=0, zakáže časovač WDT
                                ; nastaví vnitřní RC oscilátor

__config  _CONFIG1, 20C4        ; nastaví konfigurační bity shodně se vzorovými projekty Microchip
```

6. Oscilátor

Mikrořadiče PIC16F 887 mohou jako zdroj taktovacího kmitočtu používat:

6.1. Oscilátor vnější

Zdrojem taktovacího kmitočtu je úplný vnější oscilátor. Obdélníkový signál je přiveden na pin CLKIN. Dovolený kmitočet je 0 až 20MHz.

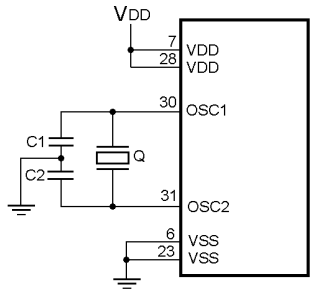


6.2. Oscilátor na čipu s vnějšími součástkami

Zdrojem taktovacího kmitočtu je oscilátor, jehož podstatná část je implementována na čipu mikrořadiče. Kmitočet oscilátoru je určen vnějšími součástkami.

6.2.1. Oscilátor řízený krystalem nebo keramickým rezonátorem

Oscilátor může být konfigurován ve třech módech podle tabulky

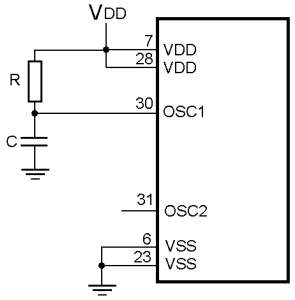


mód	kmitočet	
	krystal	keram.rezonátor
LP	32.768 kHz	--
XT	100 kHz – 4MHz	
HS	1 MHz – 20 MHz	

:

Krystal nebo keramický rezonátor mají být specifikovány pro sériovou rezonanci, sériový odpor krystalu má být co nejmenší. Hodnota kondenzátorů je stanovena v technických podmínkách použitého krystalu. Zvětšování kapacity zlepšuje stabilitu oscilátoru a zabraňuje kmitání na vyšších harmonických základního kmitočtu. Snižování kapacity zlepšuje náběh oscilací při zapnutí napájecího napětí.

6.2.2. Oscilátor řízený RC článkem



Vnější odpor má mít hodnotu mezi 3 kΩ a 100 kΩ, doporučená hodnota kondenzátoru je 20 pF. Pin OSC2 může být nakonfigurován jako výstup taktovacího kmitočtu pro synchronizaci dalších obvodů.

6.3. Oscilátory vnitřní.

Čip mikrořadiče PIC16F887 obsahuje dva úplné RC oscilátory.
Oscilátor HFINTOSC má základní kmitočet 8 MHz, který může být dále dělen až na 125 kHz. .
Oscilátor je kalibrován až na přesnost ± 2% a může být dále doladěn v registru OSCTUNE
Oscilátor LFINTOSC má základní kmitočet 32 kHz, není kalibrován a jeho kmitočet se může vlivem výrobních tolerancí, teploty a napájecího napětí pohybovat od 15 kHz do 45 kHz.
Piny OSC1 a OSC2 jsou uvolněny pro využití jako I/O piny PA6 a PA7 Portu A.

6.4. Volba zdroje taktovacího kmitočtu, registr OSCCON

Zdroj taktovacího kmitočtu určuje bit SCS (Systém Clock Select) v registru OSCCON

OSCCON Systém Clock Select Register								adresa
U	R / W	R / W	R / W	R	R	R	R / W	8F H Bank 1
--	IRCF2	IRCF1	IRCF0	OSTS	HTS	LTS	SCS	
0	1	1	0	1	0	0	0	

IRCF2 : IRCF1 : IRCF0 **Internal Oscillator Frequency Select Bits**
111 = 8 MHz
110 = 4 MHz (po Reset)
101 = 2 MHz
100 = 1 MHz
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 32 kHz (LFINTOSC)
OSTS **Oscillator Status Bit**
0 = vnitřní oscilátor (HFINTOSC nebo LFINTOSC) je zdrojem takt. kmitočtu
1 = oscilátor podle hodnoty **konfiguračních bitů FOSC** < 2 : 0 > registru _CONFIG1
HTS **HFINTOSC Status Bit**
0 = HFINTOSC není stabilní
1 = HFINTOSC je stabilní
LTS **LFINTOSC Status Bit**
0 = LFINTOSC není stabilní
1 = LFINTOSC je stabilní
SCS **Systém Clock Select Bit**
0 = vybrán oscilátor podle hodnoty **konfiguračních bitů FOSC** < 2 : 0 > registru _CONFIG1
1 = vybrán vnitřní oscilátor (HFINTOSC nebo LFINTOSC)

FOSC<2:0> Oscillator Selection bits, registr _CONFIG1 (paměť Flash)
111 = RC oscilátor, RC na pinu OSC1, výstup CLKOUT na pinu OSC2
110 = RC oscilátor, RC na pinu OSC1, pin RA6/OSC2 jako I /O pro všeobecné použití
101 = vnitřní oscilátor, výstup CLKOUT na pinu OSC2, pin RA7/OSC1 jako I/O
100 = vnitřní oscilátor, piny RA6/OSC2 a RA7/OSC1 jako I /O pro všeobecné použití
011 = vnější oscilátor, CLKIN na pinu RA7/OSC1, pin RA6/OSC2 jako I /O
010 = HS oscilátor, krystal nebo rezonátor na pinech OSC2 a OSC1
001 = XT oscilátor, krystal nebo rezonátor na pinech OSC2 a OSC1
000 = LP oscilátor, krystal 32.768 kHz na pinech OSC2 a OSC1

Pokud není registr _CONFIG1 naprogramován, je hodnota všech jeho bitů rovna 1, jako zdroj taktovacího kmitočtu je konfigurován vnější RC oscilátor.

6.5. Taktování mikrořadiče při výpadku vnějšího oscilátoru, registr _CONFIG1

Mikrořadiče PIC16F887 obsahují detekční obvod, který trvale sleduje vstup taktovacích impulzů z vnějšího oscilátoru. Jestliže dojde k výpadku taktovacích impulzů na dobu cca 2ms, je jako zdroj taktovacího kmitočtu připojen vnitřní RC oscilátor. Kmitočet vnitřního RC oscilátoru je standardně 4MHz.
Přepnutí vnějšího oscilátoru na vnitřní RC oscilátor povoluje bit FCMEN v registru _CONFIG1

FCMEN Fail Clock Monitor Enabled Bit, registr _CONFIG1 (paměť Flash)
0 = záložní RC oscilátor zakázán
1 = záložní RC oscilátor povolen

Registr _CONFIG1 je šestnáctibitový registr v paměti Flash (adresa 2007H). Adresa registru je mimo oblast přístupnou uživateli. Obsah registru může být změněn pouze direktivou __config při ukládání programu do paměti Flash.

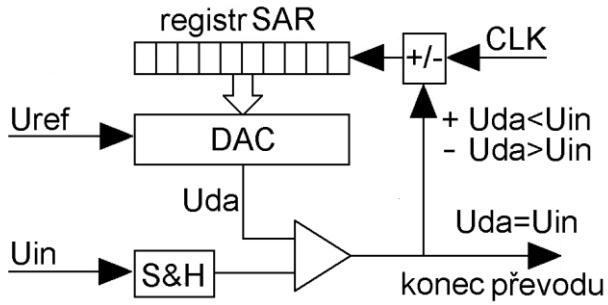
Pokud není registr _CONFIG1 naprogramován, je hodnota všech jeho bitů rovna 1, přepnutí na vnitřní RC oscilátor je povoleno.

7. AD převodník

Mikrořadiče PIC16F887 obsahují 10 bitový, 14 kanálový, komparační (SAR = Successive Approximation Register) analogově / číslicový převodník

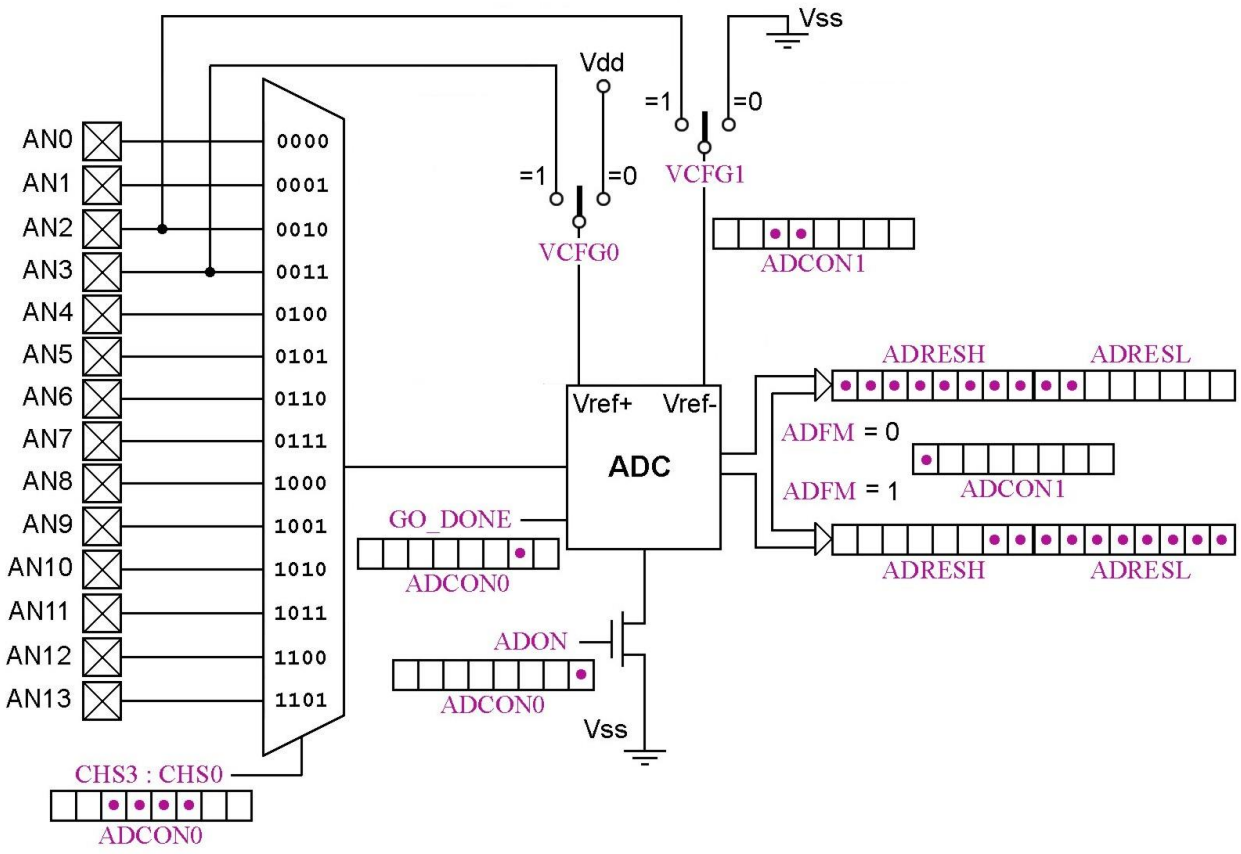
7.1. Princip SAR AD převodníku

Vstupní analogové napětí je porovnáváno na analogovém komparátoru s analogovým napětím, které je výsledkem DA převodu digitálního slova, uloženého v registru SAR. Je-li vstupní napětí menší, je obsah registru SAR zvětšen. Je-li vstupní napětí větší, je obsah registru SAR zmenšen. AD převod je ukončen když obě napětí na vstupu komparátoru jsou shodná, Výsledek převodu je uložen v registru SAR.



Uin – vstupní analogové napětí
Uref – referenční napětí, volí se buď napájecí napětí nebo vnější napěťová reference
Uda – výstupní analogové napětí AD převodníku
CLK – taktovací kmitočet, perioda musí být delší než je doba převodu (cca 2μs**)**
S&H – vzorkovací obvod (Sample and Hold)

7.2. Blokové schema AD převodníku



7.3. Konfigurace AD převodníku

Před použitím musí být AD převodník zkonfigurován :
- konfigurací vstupního Portu
- výběrem vstupního kanálu
- volbou referenčního napětí
- nastavením taktovacího kmitočtu
- formátováním výsledku

7.3.1. Konfigurace vstupního Portu

Jako vstupy analogového napětí jsou určeny Port A (kanály 0 až 7) a Port B (kanály 8 až 13).
Jako analogové vstupy musí být porty konfigurovány v registrech TRISA, TRISB, ANSEL a ANSELH.
Nastavením bitu v registru TRISA na hodnotu 1 je odpovídající pin portu A nastaven jako vstup.
Po signálu Reset jsou všechny piny Portu A i Portu B nastaveny jako analogové vstupy.

TRISA Tri-State A Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	85 H Bank 1
TRISA7	TRISA 6	TRISA 5	TRISA 4	TRISA 3	TRISA 2	TRISA 1	TRISA 0	
1	1	1	1	1	1	1	1	

TRISA **< 7 : 0 > Port A Tri-State Control Bit**
0 = pin konfigurován jako výstup
1 = pin konfigurován jako vstup

ANSEL Analog Select Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	188 H Bank 3
ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	
1	1	1	1	1	1	1	1	

ANS7 **:** ANS0 **Analog Select Bit**
0 = pin konfigurován jako digitální vstup / výstup
1 = pin konfigurován jako analogový vstup, bit registru TRISA musí být nastaven na 1

TRISB Tri-State B Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	86 H Bank 1
TRISB7	TRISB 6	TRISB 5	TRISB 4	TRISB 3	TRISB 2	TRISB 1	TRISB 0	
1	1	1	1	1	1	1	1	

TRISB **< 7 : 0 > Port B Tri-State Control Bit**
0 = pin konfigurován jako výstup
1 = pin konfigurován jako vstup

ANSELH Analog Select High Register								adresa
--	--	R / W	R / W	R / W	R / W	R / W	R / W	189 H Bank 3
--	--	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8	
--	--	1	1	1	1	1	1	

ANS13 **:** ANS8 **Analog Select Bit**
0 = pin konfigurován jako digitální vstup / výstup
1 = pin konfigurován jako analogový vstup, bit registru TRISB musí být nastaven na 1

7.3.2. Výběr vstupního kanálu

Kanál, ze kterého je analogové napětí přivedeno na vstup AD převodníku, je volen bity CHS3 : CHS0 v registru ADCON0.

ADCON0 A/D Control Register 0								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	1F H bank 0
ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO_DONE	ADON	
0	0	0	0	0	0	0	0	

CHS3:CHS2:CHS1:CHS0 Analog Channel Select – kanál A/D převodníku
xxxx = binary numbered analog channel – binární číslo kanálu (0000=AN0, 0011=AN3)

7.3.3. Volba referenčního napětí

Při AD převodu musí být připojen zdroj referenčního napětí. Tím může být buď napájecí napětí mikrořadiče (Vdd a Vss) nebo vnější referenční napětí, přivedené na piny AN2 a AN3. Volba se provádí nastavením bitů VCFG1 a VCFG0 v registru ADCON1.

ADCON1 A/D Control Register 1								adresa
R / W	U	R / W	R / W	U	U	U	U	9F H bank 1
ADFM	--	VCFG1	VCFG0	--	--	--	--	
0	0	0	0	0	0	0	0	

VCFG1 Voltage Reference – zdroj referenčního napětí
1 = Vref pin – vnější referenční napětí
0 = VSS – napájecí napětí
VCFG0 Voltage Reference – zdroj referenčního napětí
1 = Vref pin – vnější referenční napětí
0 = VDD – napájecí napětí

7.3.4. Nastavení taktovacího kmitočtu AD převodníku

AD převodník potřebuje čas nejméně 1.6μs na převedení 1 bitu výsledku. Taktovací kmitočet převodníku musí být menší než 600 kHz. Pokud je jako zdroj taktovacího kmitočtu pro AD převodník používán oscilátor mikrořadiče, musí být dělen na požadovanou hodnotu nastavením bitů ADCS1 a ADCS0 v registru ADCON0.

ADCON0 A/D Control Register 0								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	1F H bank 0
ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO_DONE	ADON	
0	0	0	0	0	0	0	0	

ADCS1: ADCS0 A/D Conversion Clock – taktovací kmitočet A/D převodu
00 = Fosc / 2
01 = Fosc / 8
10 = Fosc / 32
11 = FRC (internal RC oscillator)

Ve cvičných projektech na PICKit2 je oscilátor mikrořadiče nastaven na kmitočet 4 MHz. Dělením tohoto kmitočtu osmi je dosaženo vyhovujícího kmitočtu 500 kHz, který je použit jako taktovací kmitočet AD převodníku.

7.3.5. Formátování výsledku AD převodu

Výsledek AD převodu je ukládán do registrů ADRESH (vyšší bity) a ADRESL (nižší bity). Formát výsledku (zarovnání vlevo / vpravo) se nastavuje bitem ADSM v registru ADCON1

ADCON1 A/D Control Register 1								adresa
R / W	U	R / W	R / W	U	U	U	U	9F H bank 1
ADFM	--	VCFG1	VCFG0	--	--	--	--	
0	0	0	0	0	0	0	0	

ADFM A/D Conversion Result Format – formát výsledku A/D převodu
0 = Left Justified – výsledek zarovnán vlevo

ADRESH A/D Result Register High byte								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	1E H bank 0
MSB								
x	x	x	x	x	x	x	x	

ADRESL A/D Result Register Low byte								adresa
R / W	R / W	U	U	U	U	U	U	9E H bank 1
	LSB	--	--	--	--	--	--	
x	x	x	x	x	x	x	x	

ADFM A/D Conversion Result Format – formát výsledku A/D převodu
1 = Right Justified - výsledek zarovnán vpravo

ADRESH A/D Result Register High byte								adresa
U	U	U	U	U	U	R / W	R / W	1E H bank 0
--	--	--	--	--	--	MSB		
x	x	x	x	x	x	x	x	

ADRESL A/D Result Register Low byte								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	9E H bank 1
							LSB	
x	x	x	x	x	x	x	x	

7.4. Povolení AD převodníku

Obvod AD převodníku je po signálu Reset odpojen od napájecího napětí. Tím je snížen příkon mikrořadiče. Před použitím musí být AD převodník povolen nastavením bitu ADON v registru ADCON0.

ADCON0 A/D Control Register 0								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	1F H bank 0
ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO_DONE	ADON	
0	0	0	0	0	0	0	0	

ADON ADC Enable
1 = ADC Enabled – AD převodník povolen
0 = ADC Disabled – AD převodník zakázán

7.5. Spuštění AD převodu

Vstupní analogové napětí je přiváděno na vzorkovací obvod (Sample & Hold), ve kterém je nabíjen kondenzátor s kapacitou 10pF. Před spuštěním každého AD převodu musí být tento kondenzátor nabit na plné napětí. Čas potřebný pro nabití kondenzátoru je ovlivněn zejména hodnotou vstupního rezistoru (max10kΩ), požadovanou přesností převodu (standardně ½ LSB), velikostí napájecího napětí (standardně 5 V) a teplotou mikrořadiče (standardně 50°C).
Pro uvedené hodnoty je čas pro nabití kondenzátoru cca 4.6μs. Tento čas se nejsnáze vytvoří opakováním instrukcí **nop**, při taktovacím kmitočtu 4MHz je doba vykonání jedné instrukce rovna 1μs, postačuje tedy 5 instrukcí **nop**.

Vlastní AD převod je spuštěn nastavením bitu GO_DONE v registru ADCON0 na hodnotu 1

ADCON0 A/D Control Register 0								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	1F H bank 0
ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO_DONE	ADON	
0	0	0	0	0	0	0	0	

Bit GO_DONE nemá být nastavován v jedné instrukci spolu s bitem ADON.

7.6. Ukončení AD převodu

O ukončení převodu informuje AD převodník požadavkem na přerušení (když je povoleno) nastavením bitu ADIF v registru PIR1 a nulováním bitu GO_DONE v registru ADCON0. Výsledek převodu je uložen do registrů ADRESH a ADRESL.

Není-li využíváno přerušení, je nutné ukončení AD převodu zjistit testováním bitu GO_DONE

```
bsf     ADCON0, GO_DONE     ; start AD převodu
btfsc   ADCON0, GO_DONE     ; převod ukončen?
goto    $-1                 ; ne, opakuj test
```

Před startem nového převodu je nutné uklidit výsledek převodu kopií registrů ADRESH a ADRESL do některého GPR registru. Dalším převodem je výsledek přepsán.

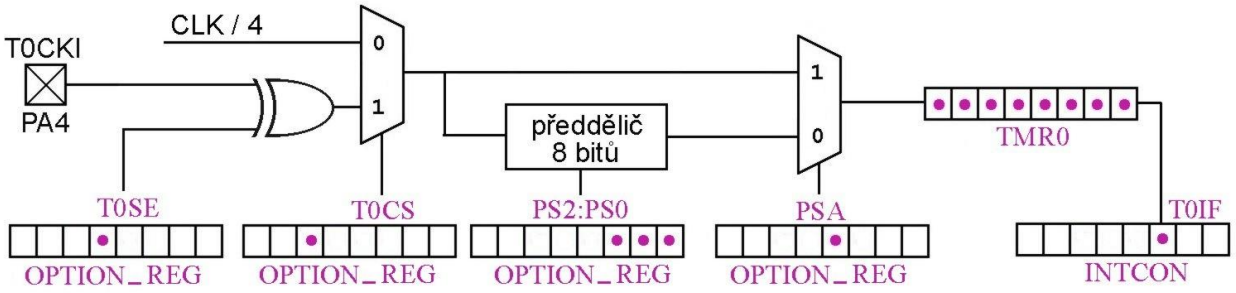
Konfigurace AD převodníku a zobrazení výsledku převodu je obsaženo v **projektu Ctyri**.
Použití výsledku AD převodu pro nastavení délky časové smyčky je obsaženo v **projektu Pet.**

8. Čítače a časovače

Mikrořadiče PIC16F887 obsahují tři moduly čítačů / časovačů:
- modul TIMER0, osmibitový čítač s osmibitovým předděličem
- modul TIMER1, šestnáctibitový čítač
- modul TIMER2, osmibitový čítač se čtyřbitovým předděličem

8.1. Čítač / časovač TIMER0

Modul TIMER0 obsahuje osmibitový čítač vpřed tvořený registrem TMR0. Nastavením bitu T0CS v registru OPTION_REG může být čítač inkrementován buď taktovacím signálem mikrořadiče při vykonání každé instrukce nebo vnějším signálem, přivedeným na pin T0CKI (RA4). Vnější signál může čítač inkrementovat buď nástupnou nebo sestupnou hranou podle hodnoty bitu T0SE v registru OPTION_REG.
Před čítač může být bitem PSA připojen osmibitový předdělič, jehož dělicí poměr může být nastaven na hodnoty 1 : 2 až 1 : 256 třemi bity PS2 : PS0 v registru OPTION_REG.
Při změně obsahu čítače (registru TMR0) z hodnoty FFH na 00H je nastaven bit T0IF v registru INTCON.



OPTION_REG Option Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	81 H bank 1
NOT_RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	
1	1	1	1	1	1	1	1	

PS2: PS0 Prescaler Rate Select bits – dělící poměr předděliče
000 = 1 : 2
001 = 1 : 4
010 = 1 : 8
011 = 1 : 16
100 = 1 : 32
101 = 1 : 64
110 = 1 : 128
111 = 1 : 256

PSA Prescaler Assigment bit – vřazení předděliče
0 = předdělič je připojen
1 = předdělič není připojen (je připojen k modulu WDT)

T0SE TIMER0 Source Edge select bit – volba hrany vstupního signálu
0 = čítač je inkrementován **sestupnou** hranou signálu T0CKI na pinu PA4
1 = čítač je inkrementován **nástupnou** hranou signálu T0CKI na pinu PA4

T0CS TIMER0 Clock Source select bit – volba vstupního signálu
0 = čítač je inkrementován taktovacím signálem mikrořadiče při vykonání každé instrukce
1 = čítač je inkrementován signálem T0CKI na pinu PA4

TMR0 Timer0 Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	01 H bank 0
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
x	x	x	x	x	x	x	x	

Registr TMR0 může být čten kdykoliv, čítač po čtení pokračuje v inkrementaci.
Do registru TMR0 může být zapsáno kdykoliv, inkrementace čítače je zakázána po dva instrukční cykly následující po zápisu.

INTCON Interrupt Control Register								adresa
R / W	R / W	R / W	R / W	R / W	R	R	R	0B H bank 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	
0	0	0	0	0	0	0	x	

Při přeplnění registru TMR0 (při změně hodnoty FFH na hodnotu 00H) je nastaven bit T0IF (TIMER0 Interrupt Flag) v registru INTCON. Je-li přerušení povoleno nastavením bitu T0IE (TIMER0 Interrupt Enable) program pokračuje obslužnou rutinou přerušení. Když přerušení využíváno není, musí program bit T0IF testovat. Bit T0IF musí být nulován programem aby mohlo být rozeznáno další přeplnění čítače.

Předdělič není možné číst ani do něho zapisovat. Jestliže PSA = 0, předdělič je vložen před čítač a při každém zápisu do registru TMR0 je předdělič vynulován.

9. Přerušovací systém PIC16F88x

9.1. Zdroje přerušení

Chod programu může být přerušen logickým signálem z některého ze zdrojů přerušení:
a. Vnější signál přivedený na pin RB0
b. Přeplnění čítače Timer0
c. Změna úrovně signálu na některém pinu Portu B
d. Periferní obvody (AD převodník, komparátor, sériové kanály)

9.2. Obsluha přerušení

Každý zdroj přerušení má přidělen jeden bit (Flag) v řídícím registru. Požadavek na přerušení nastavuje příslušný Flag na hodnotu 1. V každém instrukčním cyklu procesor testuje hodnotu všech požadavků. Pokud je některý z požadavků nastaven a přerušení je povoleno pak :
1. Bit GIE v registru INTCON je nulován, všechna přerušení jsou zakázána
2. Obsah čítače programu (návratová adresa) je uložen do zásobníku.
3. Do čítače programu je vložena hodnota 0004H a program pokračuje obslužnou rutinou přerušení, jejíž první instrukce musí být na této adrese. Obslužná rutina musí:
- Otestovat všechny požadavky na přerušení a určit zdroj přerušení
- Vynulovat Flag který přerušení způsobil a umožnit tak přijetí nového požadavku na přerušení
- Skončit instrukcí **retfie**
4. Instrukce **retfie**
- Vyzvedne návratovou adresu ze zásobníku a uloží ji do čítače programu
- Nastaví bit GIE v registru INTCON na hodnotu 1 a povolí všechna přerušení

Do zásobníku je ukládána pouze návratová adresa. Obsah důležitých registrů (W, STATUS) je vhodné uložit do paměti RAM na adresy od 70H, které jsou přístupné ve všech bankách.

Příklad obslužné rutiny přerušení od čítače Timer0

```
        cblock 70               ; blok adres RAM pro dočasné uložení registrů W a STATUS
W_Save
STATUS_Save
        endc

        org     4
ISR:                                    ; Interrupt Service Routine
        movwf   W_Save          ; dočasné uložení registru W do W_Save
        movf    STATUS, W
        movwf   STATUS_Save     ; dočasné uložení registru STATUS do STATUS_Save

        btfsc   INTCON, T0IF    ; test bitu T0IF, požadavek na přerušení od čítače Timer0
        goto    ServiceTimer0
        goto    ExitISR
ServiceTimer0:
        :
        bcf     INTCON, T0IF    ; nuluje požadavek na přerušení, umožňuje další požadavek
ExitISR:
        movf    STATUS_Save, W
        movwf   STATUS          ; přesun registru STATUS_Save zpět do STATUS
        swapf   W_Save, F       ; přesun registru W_Save zpět do W
        swapf   W_Save, W       ; swapf nemá vliv na příznaky STATUS, movf by mohlo mít
        retfie
```

9.3. Vnější přerušení

Signál vnějšího přerušení musí být přiveden na pin RB0, bit INTEDG v registru OPTION_REG určuje, jestli požadavek na přerušení vyvolá nástupná nebo sestupná hrana. Platný požadavek nastaví bit INTF v registru INTCON. Vnější přerušení může být zakázáno nulováním bitu INTE.

INTCON Interrupt Control Register								adresa
R / W	R / W	R / W	R / W	R / W	R	R	R	0B H bank 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	
0	0	0	0	0	0	0	x	

GIE General Interrupt Enable – povolení všech přerušení
0 = všechna přerušení zakázána
1 = všechna přerušení povolena

INTE Interrupt Enable – povolení vnějšího přerušení
0 = vnější přerušení zakázáno
1 = vnější přerušení povoleno

INTF Interrupt Flag – příznak vnějšího přerušení
0 = žádný požadavek vnějšího přerušení
1 = požadavek vnějšího přerušení nastaven

OPTION_REG Option Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	81 H bank 1
NOT_RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	
1	1	1	1	1	1	1	1	

INTEDG Interrupt Edge – hrana vnějšího přerušení
0 = požadavek vnějšího přerušení na sestupnou (falling) hranu
1 = požadavek vnějšího přerušení na nástupnou (rising) hranu

9.4. Přerušení od čítače Timer 0

Při přeplnění registru TMR0 (při změně hodnoty FFH na hodnotu 00H) je nastaven bit T0IF v registru INTCON. Přerušení může být zakázáno nulováním bitu T0IE.

INTCON Interrupt Control Register								adresa
R / W	R / W	R / W	R / W	R / W	R	R	R	0B H bank 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	
0	0	0	0	0	0	0	x	

GIE General Interrupt Enable – povolení všech přerušení
0 = všechna přerušení zakázána
1 = všechna přerušení povolena

T0IE Timer0 Interrupt Enable – povolení přerušení od čítače Timer 0
0 = přerušení Timer0 zakázáno
1 = přerušení Timer0 povoleno

T0IF Timer0 Interrupt Flag – příznak přerušení od čítače Timer 0
0 = žádný požadavek přerušení od Timer 0
1 = požadavek přerušení od Timer 0 nastaven

9.5. Přerušení od změny logické úrovně na pinech Portu B

Při změně logické úrovně na pinech Portu B je nastaven bit RBIF v registru INTCON. Přerušení může být zakázáno nulováním bitu RBIE. Jednotlivé piny portu B mohou být konfigurovány v registru IOCB.

INTCON Interrupt Control Register								adresa
R / W	R / W	R / W	R / W	R / W	R	R	R	0B H bank 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	
0	0	0	0	0	0	0	x	

GIE General Interrupt Enable – povolení všech přerušení
0 = všechna přerušení zakázána
1 = všechna přerušení povolena

RBIE PortB Interrupt Enable – povolení přerušení od Portu B
0 = přerušení PortB zakázáno
1 = přerušení PortB povoleno

RBIF PortB Interrupt Flag – příznak přerušení od Portu B
0 = žádný požadavek přerušení od Portu B
1 = požadavek přerušení od Portu B nastaven

IOCB Interrupt On-Change Port B Register								adresa
R / W	R / W	R / W	R / W	R / W	R / W	R / W	R / W	96 H bank 1
IOCB 7	IOCB 6	IOCB 5	IOCB 4	IOCB 3	IOCB 2	IOCB 1	IOCB 0	
0	0	0	0	0	0	0	0	

IOCB7 : IOCB0 Interrupt On-Change PortB – povolení přerušení od pinů Portu B
0 = přerušení od změny úrovně na pinu Portu B zakázáno
1 = přerušení od změny úrovně na pinu Portu B povoleno

9.6. Přerušení od periférií

Všechny periferie na čipu mikrořadičů PIC16F mohou žádat o přerušení nastavením příslušných bitů v registrech PIR1 a PIR2. Každé přerušení je možno individuálně povolit nebo zakázat nastavením nebo nulováním bitů v registrech PIE1 a PIE2.
Všechna přerušení od periferií je možno povolit nebo zakázat nastavením nebo nulováním bitu PEIE v registru INTCON.

INTCON Interrupt Control Register								adresa
R / W	R / W	R / W	R / W	R / W	R	R	R	0B H bank 0
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	
0	0	0	0	0	0	0	x	

GIE General Interrupt Enable – povolení všech přerušení
0 = všechna přerušení zakázána (včetně přerušení od periferií)
1 = všechna přerušení povolena

PEIE Peripheral Interrupt Enable – povolení přerušení od periferií
0 = přerušení od periferií zakázána
1 = přerušení od periferií povolena

Detailní popis přerušení od periferií přesahuje určení tohoto učebního textu.

10. Instrukční soubor mikrořadičů PIC16F

10.1. Struktura instrukce

Instrukční soubor má 35 instrukcí, každá má délku 14 bitů. Každá instrukce obsahuje **kód instrukce** a až dva **operandy**.

```
kód          zdrojový     cílový
instrukce    operand      operand
nop                                   ; prázdná instrukce
```
příklad instrukce bez operandu

```
clrw                                  ; vynuluje W registr
```
instrukce bez zdrojového operandu, cílový operand (W registr) je součástí kódu instrukce

```
movlw        05                       ; uloží konstantu 05h do W registru
```
cílový operand (W registr) je součástí kódu instrukce

```
movwf                TRISD            ; zkopíruje W regitr do registru TRISD
```
zdrojový operand (W registr) je součástí kódu instrukce

```
movf         PORTD,  0                ; zkopíruje PORTD do W registru
```
cílový operand (W registr) je určen hodnotou 0

```
decf         Counter, 1               ; uloží hodnotu Counter - 1  do Counter
```
cílový operand (Counter registr) je určen hodnotou 1

Operace přesunů mezi registry a uložení konstanty do registru musí být provedeny přes registr W (Working register), musí být vykonány pomocí dvou instrukcí.

Všechny instrukce které mají jako operand registr (file) pracují v módu Read – Modify – Write, tedy obsah registru je čten, obsah je změněn a výsledek je zapsán do cílového operandu.

Většina instrukcí je vykonána v jednom instrukčním cyklu, který je složen ze čtyř taktů hodinového kmitočtu. Při taktovacím kmitočtu 4MHz je instrukce vykonána za 1μs.
Instrukce s podmíněným skokem a instrukce, které mění obsah čítače programu jsou vykonány ve dvou instrukčních cyklech.

Instrukce jsou rozděleny do tří skupin

- **bytově orientované** operace, operandem je celý byte (registr)

```
    incf         Counter, 1           ; inkrementuje Counter, výsledek uloží do Counter
```

- **bitově orientované**, operandem je určený bit určeného byte (registru)

```
    bsf          STATUS,  RP0         ; nastaví bit RP0 registru STATUS
```

- operace s **přímým operandem** (konstantou) a řídící operace

```
    movlw        5A                   ; uloží konstantu 5Ah do registru W
    goto         Start                ; nepodmíněný skok na návěští Start
    nop                               ; prázdná instrukce
```

10.2. Přehled instrukcí

mnemonika	funkce	cykly	14 bitů instrukce	status
Bytově orientované instrukce				
addwf f, d	Add W and f	1	00 0111 d f f f f f f f f	C, DC, Z
andwf f, d	AND W with f	1	00 0101 d f f f f f f f f	Z
clrf f	Clear f	1	00 0001 1 f f f f f f f f	Z
clrw	Clear W	1	00 0001 0 xxx xxxx	Z
comf f, d	Complement f	1	00 1001 d f f f f f f f f	Z
decf f, d	Decrement f	1	00 0011 d f f f f f f f f	Z
decfsz f, d	Decrement f, Skip if 0	1 (2)	00 1011 d f f f f f f f f	
incf f, d	Increment f	1	00 1010 d f f f f f f f f	Z
incfsz f, d	Increment f, Skip if 0	1 (2)	00 1111 d f f f f f f f f	
iorwf f, d	Inclusive OR W with f	1	00 0100 d f f f f f f f f	Z
movf f, d	Move f	1	00 1000 d f f f f f f f f	Z
movwf f	Move W to f	1	00 0000 1 f f f f f f f f	
nop	No Operation	1	00 0000 0 xx0 0000	
rlf f, d	Rotate Left f through Carry	1	00 1101 d f f f f f f f f	C
rrf f, d	Rotate Right f through Carry	1	00 1100 d f f f f f f f f	C
subwf f, d	Subtract W from f	1	00 0010 d f f f f f f f f	C, DC, Z
swapf f, d	Swap nibbles in f	1	00 1110 d f f f f f f f f	
xorwf f, d	Exclusive OR W with f	1	00 0110 d f f f f f f f f	Z
Bitově orientované instrukce				
bcf f, b	Bit Clear f	1	01 00bb b f f f f f f f f	
bsf f, b	Bit Set f	1	01 01bb b f f f f f f f f	
btfsc f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb b f f f f f f f f	
btfss f, b	Bit Test f, Skip if Set	1 (2)	01 11bb b f f f f f f f f	
Instrukce s přímým operandem (konstanta) a řídící instrukce				
addlw k	Add literal and W	1	11 111x kkkk kkkk	C, DC, Z
andlw k	AND literal with W	1	11 1001 kkkk kkkk	Z
call k	Call subroutine	2	10 0 kkk kkkk kkkk	
clrwdt	Clear Watchdog Timer	1	00 0000 0110 0100	/TO, /PD
goto k	Go to address	2	10 1 kkk kkkk kkkk	
iorlw k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z
movlw k	Move literal to W	1	11 00xx kkkk kkkk	
retfie	Return from interrupt	2	00 0000 0000 1001	
retlw k	Return with literal in W	2	11 01xx kkkk kkkk	
return	Return from Subroutine	2	00 0000 0000 1000	
sleep	GO into standby mode	1	00 0000 0110 0011	/TO, /PD
sublw k	Subtract W from literal	1	11 110x kkkk kkkk	C, DC, Z
xorlw k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z

f 7-bitová adresa registru, může adresovat 128 registrů tj. jedna banka paměti RAM
d = 0 cílový operand je W
d = 1 cílový operand je f
b 3-bitová adresa bitu, může adresovat 8 bitů (tj. bity 0 až 7)
k 8-bitová hodnota přímého operandu (konstanta) tj. hodnota 0 až 255
k 11-bitová adresa (**call** a **goto**), může adresovat 2K instrukcí tj. 1 stránka paměti Flash

10.3. Formát instrukcí

Bytově orientované instrukce

13					8	7	6						0
kód instrukce						**d**	**f**	**f**	**f**	**f**	**f**	**f**	**f**

d = 0 cílový operand je W
d = 1 cílový operand je f
f 7-bitová adresa, může adresovat 128 registrů tj. jedna banka paměti RAM

Bitově orientované instrukce

13			10	9		7	6						0
kód instrukce				**b**	**b**	**b**	**f**	**f**	**f**	**f**	**f**	**f**	**f**

b 3-bitová adresa bitu
f 7-bitová adresa, může adresovat 128 registrů tj. jedna banka paměti RAM

Instrukce s přímým operandem

13					8	7							0
kód instrukce						**k**	**k**	**k**	**k**	**k**	**k**	**k**	**k**

k 8-bitová hodnota přímého operandu (konstanta) tj. hodnota 0 až 255

Instrukce CALL a GOTO

13		11	10										0
kód instrukce			**k**	**k**	**k**	**k**	**k**	**k**	**k**	**k**	**k**	**k**	**k**

k 11-bitová adresa, může adresovat 2K instrukcí tj. 1 stránka paměti Flash

10.4. Popis instrukcí

addlw — Add Literal and W

Syntaxe: **addlw k** $0 \le k \le 255$
Operace: W + k → W
Status: C, DC, Z
Popis: obsah registru **W** je přičten k osmibitové konstantě **k** a výsledek je uložen do **W**
Příklad:

addlw 15	před instrukcí	W = 10h
	po instrukci	W = 25h
addlw Myreg	před instrukcí	W = 10h
		adresa registru Myreg = 37h
	po instrukci	W = 47h

addwf — Add W and f

Syntaxe: **addwf f, d** $0 \le f \le 127$ d = 0 nebo 1
Operace: W + f → destination
Status: C, DC, Z
Popis: obsahy registrů **W** a **f** jsou sečteny, výsledek je uložen do **W** když **d = 0**
obsahy registrů **W** a **f** jsou sečteny, výsledek je uložen do **f** když **d = 1**
Příklad:

addwf FSR, 0	před instrukcí	W = 17h, FSR = C2h
	po instrukci	W = D9h, FSR = C2h
addwf INDF, 1	před instrukcí	W = 17h, obsah FSR = 20h
	po instrukci	W = 17h, obsah FSR = 37h

andlw — AND Literal with W

Syntaxe: **andlw k** $0 \le k \le 255$
Operace: W AND k → W
Status: Z
Popis: logická operace AND obsahu registru **W** s konstantou **k,** výsledek je uložen do **W**
Příklad:

andlw 5F	před instrukcí	W = A3h = 1010 0011b
		k = 5Fh = 0101 1111b
	po instrukci	W = 03h = 0000 0011b

andwf — AND W with f

Syntaxe: **andwf f, d** $0 \le f \le 127$ d = 0 nebo 1
Operace: W AND f → destination
Status: Z
Popis: logická operace AND obsahu registrů **W** a **f**, výsledek do **W** když **d = 0**
logická operace AND obsahu registrů **W** a **f**, výsledek do **f** když **d = 1**
Příklad:

andwf FSR, 1	před instrukcí	W = 17h = 0001 0111b
		FSR = C2h =1100 0010b
	po instrukci	W = 17h
		FSR = 02h = 0000 0010b

bcf — Bit Clear f

Syntaxe: **bcf f, b** 0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operace: 0 → f <b>
Status: --
Popis: bit **b** v registru **f** je vynulován
Příklad: **bcf** STATUS, 6 před instrukcí STATUS = E7h = 1110 0111b
po instrukci STATUS = A7h = 1010 0111b

bsf — Bit Set f

Syntaxe: **bsf f, b** 0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operace: 1 → f <b>
Status: --
Popis: bit **b** v registru **f** je nastaven na hodnotu 1
Příklad: **bsf** STATUS, 5 před instrukcí STATUS = 0Ah = 0000 1010b
po instrukci STATUS = 2Ah = 0010 1010b

btfsc — Bit Test, Skip if Clear

Syntaxe: **btfsc f, b** 0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operace: skip if f <b> = 0
Status: --
Popis: když bit **b** v registru **f** je 0 pak následující instrukce je vykonána jako **nop**
Příklad: Here: **btfsc** Flag, 4 před instrukcí PC = Here, Flag = xxx0 xxxxb
False: **goto** $ po instrukci PC = True
True: xx

btfss — Bit Test, Skip if Set

Syntaxe: **btfss f, b** 0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operace: skip if f <b> = 1
Status: --
Popis: když bit **b** v registru **f** je 1 pak následující instrukce je vykonána jako **nop**
Příklad: Here: **btfss** Flag, 4 před instrukcí PC = Here, Flag = xxx0 xxxb
False: **goto** $ po instrukci PC = False
True: xx

call — Call Subroutine

Syntaxe: **call** **k** 0 ≤ k ≤ 2047
Operace: PC + 1 → Top of Stack (TOS)
k → PC<10:0>
PCLATH <4:3> → PC<12:11>
Status: --
Popis: 13-bitová návratová adresa PC+1 je uložena do zásobníku
11-bitová adresa **k** je uložena do PC<10:0>
nejvyšší dva bity PC jsou nataženy z PCLATH<4:3>
Příklad: Here: **call** There před instrukcí PC = Here
po instrukci PC = There , TOS = Here + 1

clrf — Clear f

Syntaxe: **clrf** **f** 0 ≤ f ≤ 127
Operace: 00h → f
1 → Z
Status: Z
Popis: obsah registru **f** je vynulován, příznak Z je nastaven na hodnotu 1
Příklad: **clrf** TRISD před instrukcí TRISD = 5Ah
po instrukci TRISD = 00h, Z = 1

clrw — Clear W

Syntaxe: **clrw**
Operace: 00h → W
1 → Z
Status: Z
Popis: obsah registru **W** je vynulován, příznak Z je nastaven na hodnotu 1
Příklad: **clrw** před instrukcí W = 5Ah
po instrukci W = 00h, Z = 1

clrwdt — Clear Watchdog Timer

Syntaxe: **clrwdt**
Operace: 00h → WDT
0 → WDT předdělička
1 → NOT_TO
1 → NOT_PD
Status: NOT_TO, NOT_PD
Popis: časovač **WDT i předdělička** vynulovány, příznaky NOT_TO a NOT_PD nastaveny
Příklad: **clrwdt** před instrukcí WDT = xx
po instrukci WDT = 00, prescaler count= 0

comf — Complement f

Syntaxe: **comf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: NON f → destination
Status: Z
Popis: obsah registru **f** je invertován po bitech, výsledek do **W** když **d = 0**
obsah registru **f** je invertován po bitech, výsledek do **f** když **d = 1**
Příklad: **comf** Reg1, 0 před instrukcí Reg1 = 13h = 0001 0011b
po instrukci W = ECh = 1110 1100b
Reg1 = 13h = 0001 0011b

decf — Decrement f

Syntaxe: **decf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: f - 1 → destination
Status: Z
Popis: obsah registru **f** je dekrementován (zmenšen o 1), výsledek do **W** když **d = 0**
obsah registru **f** je dekrementován (zmenšen o 1), výsledek do **f** když **d = 1**
Příklad: **decf** Count, 1 před instrukcí Count = 01h
po instrukci Count = 00h

decfsz — Decrement f, Skip if 0

Syntaxe: **decfsz f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: f - 1 → destination, skip if□□ f-1= 0
Status: --
Popis: obsah registru **f** je dekrementován (zmenšen o 1), výsledek do **W** když **d = 0**
obsah registru **f** je dekrementován (zmenšen o 1), výsledek do **f** když **d = 1**
když výsledek je 0 pak následující instrukce je vykonána jako **nop**
Příklad:
```
Loop:  decfsz Count, 1
       goto   Loop
Continue:
```
před instrukcí PC = Loop, Count = 01h
po instrukci PC = Continue, Count = 00h

goto — Go to address

Syntaxe: **goto k** 0 ≤ k ≤ 2047
Operace: k → PC<10:0>
PCLATH<4:3> → PC<12:11>
Status: --
Popis: nepodmíněný skok na adresu **k**
11-bitová adresa **k** je uložena do PC<10:0>
nejvyšší dva bity PC jsou nataženy z PCLATH<4:3>
Příklad:
```
Here:  goto 200
```
před instrukcí PC = Here
po instrukci PC = 200h

incf — Increment f

Syntaxe: **incf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: f + 1 → destination
Status: Z
Popis: obsah registru **f** je inkrementován (zvětšen o 1), výsledek do **W** když **d = 0**
obsah registru **f** je inkrementován (zvětšen o 1), výsledek do **f** když **d = 1**
Příklad: **incf** Count, 1 před instrukcí Count = FFh
po instrukci Count = 00h

incfsz — Increment f, Skip if 0

Syntaxe: **incfsz f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: f + 1 → destination, skip if f+1= 0
Status: --
Popis: obsah registru **f** je inkrementován (zvětšen o 1), výsledek do **W** když **d = 0**
obsah registru **f** je inkrementován (zvětšen o 1), výsledek do **f** když **d = 1**
když výsledek je 0 pak následující instrukce je vykonána jako **nop**
Příklad:
```
Loop:  incfsz  Count, 1
       goto    Loop
Continue:
```
před instrukcí PC = Loop, Count = FFh
po instrukci PC = Continue, Count = 00h

iorlw — Inclusive OR Literal with W

Syntaxe: **iorlw k** 0 ≤ k ≤ 255
Operace: W OR k → W
Status: Z
Popis: logická operace OR obsahu registru **W** s konstantou **k,** výsledek je uložen do **W**
Příklad: **iorlw** 35 před instrukcí W = 9Ah = 1001 1010b
k = 35h = 0011 0101b
po instrukci W = BFh = 1011 1111b

iorwf — Inclusive OR W with f

Syntaxe: **iorwf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: W OR f → destination
Status: Z
Popis: logická operace OR obsahu registrů **W** a **f**, výsledek do **W** když **d = 0**
logická operace OR obsahu registrů **W** a **f**, výsledek do **f** když **d = 1**
Příklad: **iorwf** Result, 0 před instrukcí W = 91h = 1001 0001b
Result = 13h = 0001 0011b
po instrukci W = 93h = 1001 0011b
Result = 13h = 0001 0011b

movlw — Move Literal to W

Syntaxe: **movlw k** 0 ≤ k ≤ 255
Operace: k → W
Status: --
Popis: osmibitová konstanta **k** je uložena do **W**
Příklad: **movlw** 5A
před instrukcí W = xxxx
k = 5Ah
po instrukci W = 5Ah

movf — Move f

Syntaxe: **movf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: f → destination
Status: Z
Popis: obsah registru **f** je přesunut do **W** když **d = 0**
obsah registru **f** je přesunut do **f** když **d = 1**
Příklad: **movf** FSR, 0
před instrukcí W = 00h
FSR = C2h
po instrukci W = C2h
FSR = C2h
movf FSR, 1
před instrukcí W = 00h
FSR = 43h
po instrukci W = 00h
FSR = 43h

movwf — Move W to f

Syntaxe: **movwf f** 0 ≤ f ≤ 127
Operace: W → f
Status: --
Popis: obsah registru **W** je přesunut do **f**
Příklad: **movwf** Reg1
před instrukcí W = 47h
Reg1 = FFh
po instrukci W = 47h
Reg1 = 47h

nop — No Operation

Syntaxe: **nop**
Operace: No operation
Status: --
Popis: No operation
Příklad: Here: **nop**
před instrukcí PC = Here
po instrukci PC = Here + 1

retfie — Return from Interrupt

Syntaxe: **retfie**

Operace: TOS → PC (Top of Stack to Program Counter)
1 → GIE (General Interrupt Enable)

Status: --

Popis: 13-bitová adresa ze zásobníku je uložena do PC
GIE, sedmý bit registru INTCON je nastaven na hodnotu 1, přerušení povoleno

Příklad:
```
retfie          před instrukcí  PC = xxxx
                po instrukci    PC = TOS, GIE = 1
```

retlw — Return with Literal in W

Syntaxe: **retlw k** 0 ≤ k ≤ 255

Operace: k → W
TOS → PC (Top of Stack to Program Counter)

Status: --

Popis: osmibitová konstanta **k** je uložena do **W**
13-bitová adresa ze zásobníku je uložena do PC

Příklad:
```
Here:   call    Table   ; W obsahuje počáteční adresu tabulky
                                        po instrukci    W = 8
Table:  addwf   PCL,0                   PC = TOS = Here + 1
        retlw   1       ; začátek tabulky
        xx
        retlw   8       ; konec tabulky
```

return — Return from Subroutine

Syntaxe: **return**

Operace: TOS → PC (Top of Stack to Program Counter)

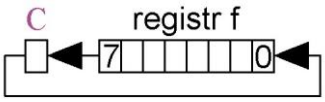
Status: --

Popis: 13-bitová adresa ze zásobníku je uložena do PC

Příklad:
```
return          před instrukcí  PC = xxxx
                po instrukci    PC = TOS
```

rlf — Rotate Left f through Carry

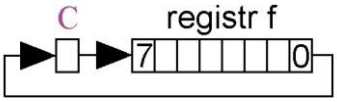
Syntaxe: **rlf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: viz příklad
Status: C
Popis: obsah registru **f** rotuje o 1 bit vlevo včetně příznaku C, výsledek do **W** když **d = 0**
obsah registru **f** rotuje o 1 bit vlevo včetně příznaku C, výsledek do **f** když **d = 1**
Příklad: **rlf** Reg1, 0 před instrukcí Reg1 = E6h = 1110 0110b
C = 0



po instrukci Reg1 = E6h = 1110 0110b
W = BBh = 1100 1100b
C = 1

rrf — Rotate Right f through Carry

Syntaxe: **rrf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: viz popis
Status: C
Popis: obsah registru **f** rotuje o 1 bit vpravo včetně příznaku C, výsledek do **W** když **d = 0**
obsah registru **f** rotuje o 1 bit vpravo včetně příznaku C, výsledek do **f** když **d = 1**
Příklad: **rrf** Reg1, 1 před instrukcí Reg1 = E6h = 1110 0110b
C = 0



po instrukci Reg1 = 73h = 0111 0011b
C = 0

sleep — Sleep

Syntaxe: **sleep**
Operace: 00h → WDT
0 → WDT předdělička
1 → NOT_TO
0 → NOT_PD
Status: NOT_TO, NOT_PD
Popis: mikrořadič je uveden do stavu Sleep, oscilátor je zastaven
časovač **WDT i předdělička** vynulovány, NOT_PD nulován, NOT_TO nastaven
Příklad: **sleep**

sublw — Subtract W from Literal

Syntaxe: **sublw k** 0 ≤ k ≤ 255
Operace: k - W → W
Status: C, DC, Z
Popis: obsah registru **W** je odečten od osmibitové konstanty **k** a výsledek je uložen do **W**

Příklad:				
	sublw	02	před instrukcí	W = 01h
			po instrukci	W = 01h, C = 1, Z = 0
	sublw	02	před instrukcí	W = 02h
			po instrukci	W = 00h, C = 1, Z = 1
	sublw	02	před instrukcí	W = 03h
			po instrukci	W = FFh, C = 0, Z = 0
	sublw	Myreg	před instrukcí	W = 10h
				adresa registru Myreg = 37h
			po instrukci	W = 27h, C = 1

subwf — Subtract W from f

Syntaxe: **subwf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: f - W → destination
Status: C, DC, Z
Popis: obsah registru **W** je odečten od registru **f ,** výsledek je uložen do **W** když **d = 0**
obsah registru **W** je odečten od registru **f ,** výsledek je uložen do **f** když **d = 1**

Příklad:				
	subwf	Reg1, 1	před instrukcí	W = 02h
				Reg1 = 03h
			po instrukci	W = 02h
				Reg1 = 01h

swapf — Swap Nibbles in f

Syntaxe: **swapf f, d** 0 ≤ f ≤ 127 d = 0 nebo 1
Operace: f<3:0> → destination <7:4>
f<7:4> → destination <3:0>
Status: --
Popis: spodní a horní půlbyty registru **f** jsou zaměněny**,** výsledek do **W** když **d = 0**
spodní a horní půlbyty registru **f** jsou zaměněny**,** výsledek do **f** když **d = 1**

Příklad:				
	swapf	Reg1, 1	před instrukcí	Reg1 = A5h
			po instrukci	Reg1 = 5Ah

xorlw — Exclusive OR Literal with W

Syntaxe: **xorlw k** $0 \le k \le 255$

Operace: W XOR k $\rightarrow$ W

Status: Z

Popis: logická operace XOR obsahu registru **W** s konstantou **k,** výsledek je uložen do **W**

Příklad: **xorlw** 0AF

před instrukcí W = B5h = 1011 0101b
k = AFh = 1010 1111b

po instrukci W = 1Ah = 0001 1010b

xorwf — Exclusive OR W with f

Syntaxe: **xorwf f, d** $0 \le f \le 127$ d = 0 nebo 1

Operace: W XOR f $\rightarrow$ destination

Status: Z

Popis: logická operace XOR obsahu registrů **W** a **f**, výsledek do **W** když **d = 0**
logická operace XOR obsahu registrů **W** a **f**, výsledek do **f** když **d = 1**

Příklad: **xorwf** Reg1, 1

před instrukcí W = B5h = 1011 0101b
Reg1 = AFh = 1010 1111b

po instrukci Reg1 = 1Ah = 0001 1010b

11. Assembler MPASM

Assembler je strojově orientovaný programovací jazyk, ve kterém jsou psány zdrojové soubory xxx**.asm** aplikačních programů pro mikrořadiče PIC16F.

11.1. Syntaktická pravidla assembleru MPASM

11.1.1. Příkazový řádek

Základním elementem zdrojového textu je příkazový řádek, který je rozdělen do čtyř polí, jejichž pořadí a pozice na řádku musí být dodrženy

- label (návěští)
- mnemonics (instrukce, direktiva, makro)
- operands (operandy)
- commentary (komentář)

Příklad :

```
label   mnemonics   operands      commentary
        #include    p16f887.inc

Dest:   equ         0B            ; definice proměnné Dest

        org         00            ; reset vector
        goto        Start

        org         20            ; začátek programu

Start :
        movlw       0A            ; uloží konstantu 0Ah to W
        movwf       Dest          ; kopíruje W (0Ah) do proměnné Dest
        bcf         Dest, 3       ; instrukce se dvěma operandy, nuluje třetí bit proměnné Dest
        goto        Start         ; smyčka
        end
```

11.1.2. Label (návěští)

Návěští (label) je pojmenování jednoho nebo více řádků a slouží pro **větvení programu**. Pravidla pro návěští jsou :

- začíná na prvním znaku řádku
- může obsahovat až 32 znaků
- jako poslední znak má být mezera, Tab, „:“ (dvojtečka) nebo konec řádku (Enter)
- nesmí být použita vyhrazená slova tj. instrukce a direktivy
- jsou Case sensitive, tj. rozlišují se malá a velká písmena, Delay a DELAY jsou různá návěští

11.1.3. Mnemonics (instrukce, direktivy)

Instrukce jsou vyhražená slova (**add**, **call**, **movwf**) která jsou překládána do strojového kódu.
Direktivy jsou příkazy pro způsob překladu a nejsou překládány.
Pravidla:

- **začínají na druhém** nebo dalším znaku řádku
- pokud je label na stejném řádku s instrukcí, musí být oddělen znakem „:“ (dvojtečka)
- nejsou Case sensitive, **MOVWF** a **movwf** jsou shodné instrukce

11.1.4. Operandy

Operandy obsahují data, s kterými jsou prováděny operace a adresy uložení instrukcí. Každý řádek může obsahovat až dva operandy.

Pravidla:

- operandy následují za instrukcí
- mezi instrukcí (direktivou) a operandem musí být jedna nebo více mezer
- dva operandy musí být odděleny znakem „ , “ (čárka)
- jsou Case sensitive, JEDNA a jedna jsou dva různé symboly operandů

11.1.5. Komentář

Komentář (poznámky) jsou texty, které vysvětlují funkci programu. Doporučuje se opatřit každý zdrojový text dostatečným počtem komentářů. Komentář není překládán.

Pravidla:

- komentář následuje za operandy, může začínat na kterékoliv pozici řádku včetně první
- komentář začíná znakem „ ; “ (středník) a končí znakem konec řádku (Enter)

11.2. Konstanty

Assembler podporuje následující aritmetické konstanty

- **hexadecimální**	H‘9F‘	nebo	h‘9F‘	nebo	0x9F
- **dekadické**	D‘100‘	nebo	d‘100‘	nebo	.100
- **binární**	B‘0100‘	nebo	b‘0100‘		

Ve standardních projektech v MPLAB IDE jsou ve výchozím nastavení konstanty interpretovány jako hexadecimální, je tedy možné psát 9F místo H‘9F‘

Výchozí nastavení konstant je možné změnit direktivou radix.

Aritmetická konstanta musí začínat číslicí. Před hexadecimální konstanty začínající znaky A až F musí být zapsána číslice 0, např 0A0 nebo 0FF.

11.3. Oprerátory

Aritmetické operátory mohou být použity spolu s direktivami a jejich proměnnými :

operátor	popis	příklad
$	obsah čítače programu	**goto** $+3
low	nižší byte adresy	**movlw** low Table
high	vyšší byte adresy	**movlw** high Table
*	násobení	a = b * c
/	dělení	a = b / c
=	rovná se	index = 0

Assembler MPASM podporuje cca 40 operátorů, uvedeny jsou jen ty nejčastěji použité ve vzorových programech a při cvičení.

11.4. Direktivy

Direktivy jsou příkazy assembleru, které jsou součástí zdrojového textu ale nejsou překládány do spustitelného souboru. Direktivy určují postup překladu, nastavují vstupy, výstupy a umístění dat.
Většina direktiv má více jmen a formátů kvůli podpoře starších verzí assembleru. Dále jsou uvedeny jen direktivy a jejich formát použitý ve vzorových programech a cvičeních.
Direktivy nejsou Case Sensitive, END, End a end jsou totožné direktivy.

banksel *symbol*

Nastaví assembler na banku RAM, která obsahuje *symbol*. Tento symbol musí být dříve definován.

```
        #include    p16f887.inc    ; obsahuje standardní definice symbolů pro PIC16F887

        banksel     TRISB          ; nastavuje Banku 1 ve které je umístěn registr TRISB

        clrf        TRISB          ; nuluje TRISB, nastavuje PORTB jako výstupy
        banksel     PORTB          ; nastavuje banku 0 ve které je umístěn registr PORTB

        movlw       55
        movwf       PORTB          ; ukládá hodnotu 55h na Port D
        goto $
        end                        ; každý program musí být ukončen direktivou end
```

cblock **address**

Definuje seznam symbolů a určuje jejich adresy v paměti RAM.
address nastavuje adresu prvního symbolu seznamu, musí být umístěna do pole registrů GPR (pro všeobecné použití). První adresa GPR v bance 0 mikrořadiče PIC16F887 je 20h. Pokud není definováno jinak je symbolu přiřazen jeden registr (jeden byte).
Seznam symbolů končí direktivou **endc**.

```
        cblock      20
Delay1                      ; definuje registr Delay1 na adrese 020h
Delay2                      ; definuje registr Delay2 na adrese 021h
        endc
```

__config ***výraz***

Nastavuje konfigurační bity procesoru, který musí být předtím definován direktivou list nebo v projektu MPLAB IDE volbou **Configure / Select Device**
Direktiva má být umístěna na začátku zdrojového textu. Výraz má obsahovat symboly bitů použité v hlavičkovém souboru **.inc** pro daný typ mikrořadiče. Více bitů může být spojeno znakem & (logický AND).
Direktiva začíná znaky __ **(dvojí podtržení)**, pro řadu PIC18F má být používán formát config.

```
#include p16F887.inc
        __config    _CONFIG1, _BOR_OFF & _MCLRE_OFF & _PWRTE_ON
```

end

Konec zdrojového textu. Při překladu jednoho zdrojového textu musí být obsažena právě jedna direktiva end. Pokud zdrojový text obsahuje soubor .inc, nesmí tento soubor obsahovat end.

```
        #include        p16f887.inc
                :                       ; instrukce programu
                :
        end                             ; konec zdrojového textu
```

endc

Konec seznamu symbolů začínající direktivou cblock. Direktivy cblock a endc musí být vždy v páru.

```
        cblock          20
Delay1                          ; Definuje dva registry
Delay2
        endc
```

symbol equ výraz

Hodnota **výraz** je přiřazena k symbol
Používá se zejména k přiřazení symbolu k adrese registru SFR

```
STATUS          equ     03      ; místo adresy 03h může program používat symbol STATUS
TRISD           equ     88      ; místo adresy 88h může program používat symbol TRISD
PORTD           equ     08      ; místo adresy 08h může program používat symbol PORTD
PORTC           equ     07
PCLATH          equ     0A
```

#include *file.inc*

#include *"c:\ Program Files \ myfile.inc"*

Specifikovaný soubor **file.inc** je čten jako součást zdrojového textu na místě direktivy jako by celý soubor byl skutečně na místě direktivy napsán.
Pokud je použit vlastní soubor .inc nebo pokud název souboru obsahuje mezery, musí být uzavřen v uvozovkách.

```
        #include        p16F887.inc     ; standardní include soubor
```

org *adresa*

Definuje první adresu paměti Flash na kterou je uložen přeložený strojový kód. Obvykle používané adresy pro mikrořadiče PIC16F jsou:

00h	reset vektor
04h	vektor rutiny pro ošetření přerušení
05h nebo vyšší	začátek uživatelského programu

```
        #include    p16f887.inc     ; vložení standardní definice symbolů

        org     00                  ; následující přeložená instrukce bude uložena na adresu 0h
        goto    Main                ; skok na adresu s návěštím Main

        org     04                  ; následující přeložená instrukce bude uložena na adresu 04h
        goto    int_routine         ; skok na adresu s návěštím int_routine

        org     10                  ; následující přeložená instrukce bude uložena na adresu 10h
Main
        ;                           ; hlavní program
        goto    Main                ; smyčka na návěští Main

        org     100                 ; následující přeložená instrukce bude uložena na adresu 100h
int_routine
        ;                           ; obslužná rutina přerušení
        retfie                      ; návrat z přerušení
        end
```

radix *hex / dec / oct*

Nastavuje radix (základ číselné soustavy) pro interpretaci dat. V projektech v MPLAB IDE je výchozí nastavení **hex** a není ho nutné nastavovat.

```
        movlw       D‘50‘   ; 50 je deklarováno jako dekadické číslo
        movlw       O‘50‘   ; 50 je deklarováno jako oktalové (osmičkové) číslo
        movlw       50      ; 50 není deklarováno a je ve výchozím nastavení jako hexadecimální

        radix       dec
        movlw       H‘50‘   ; 50 je deklarováno jako hexadecimální číslo
        movlw       O‘50‘   ; 50 je deklarováno jako oktalové (osmičkové) číslo
        movlw       50      ; 50 není deklarováno a je direktivou radix nastaveno jaho dekadické
```

12. Projekty na PICKit2 Debugg Expres

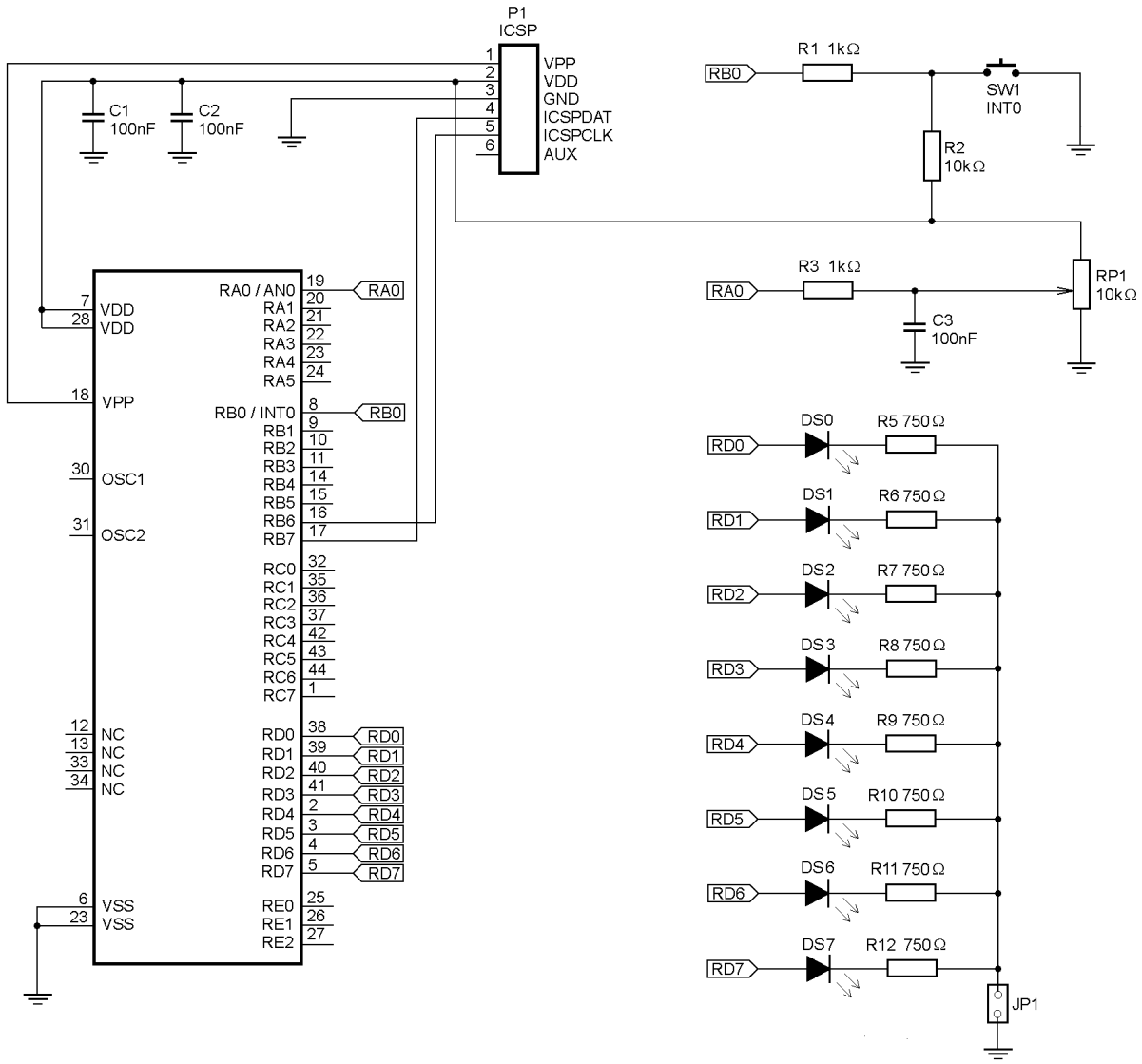
12.1. PICKit2

PICKit2 je hardwarový debugger, který podporuje ladění programů na na mikrořadičích PIC16F. Sestava „Debugg Express“ obsahuje demonstrační desku, osazenou mikrořadičem PIC16F887 v pouzdře TQFP44.

PICKit2 umožňuje :
- psaní programů v assembleru MPASM v prostředí MPLAB IDE
- překlad a odladění programů na reálném procesoru
- uložení programu do paměti Flash

Demonstrační deska umožňuje ladění a chod aplikačních programů se zaměřením na :
- konfigurace PortuD, ovládání osmi LED programem
- softwarová zpožďovací smyčka, blikání LED
- konfigurace PortuA jako digitální vstup, ovládání LED tlačítkem
- konfigurace portuA jako analogový vstup, AD převodník pro měření napětí
- použití AD převodníku pro řízení doby zpožďovací smyčky
- čítač Timer 0 pro měření času
- programy složené z výše uvedených aplikací

12.2. Schema demonstrační desky s mikrořadičem PIC16F887



P1	konektor ICSP (In Circuit Serial Programming) pro připojení debuggeru
DS0 – DS7	LED diody připojené na port D
JP1	spojka (jumper) pro odpojení LEDs od napájení
RP1	potenciometr jako zdroj analogového napětí pro úlohy s A/D převodníkem
SW1	spínací tlačítko, zdroj přerušení
VDD	plus pól napájecího napětí 5V
VSS, GND	mínus pól napájecího napětí, zem
VPP	programovací napětí
ICSPDAT	datový vodič sběrnice ICSP
ICSPCLK	taktovací vodič sběrnice ICSP