Navodila za delo z modulom MPU-PIC16F876

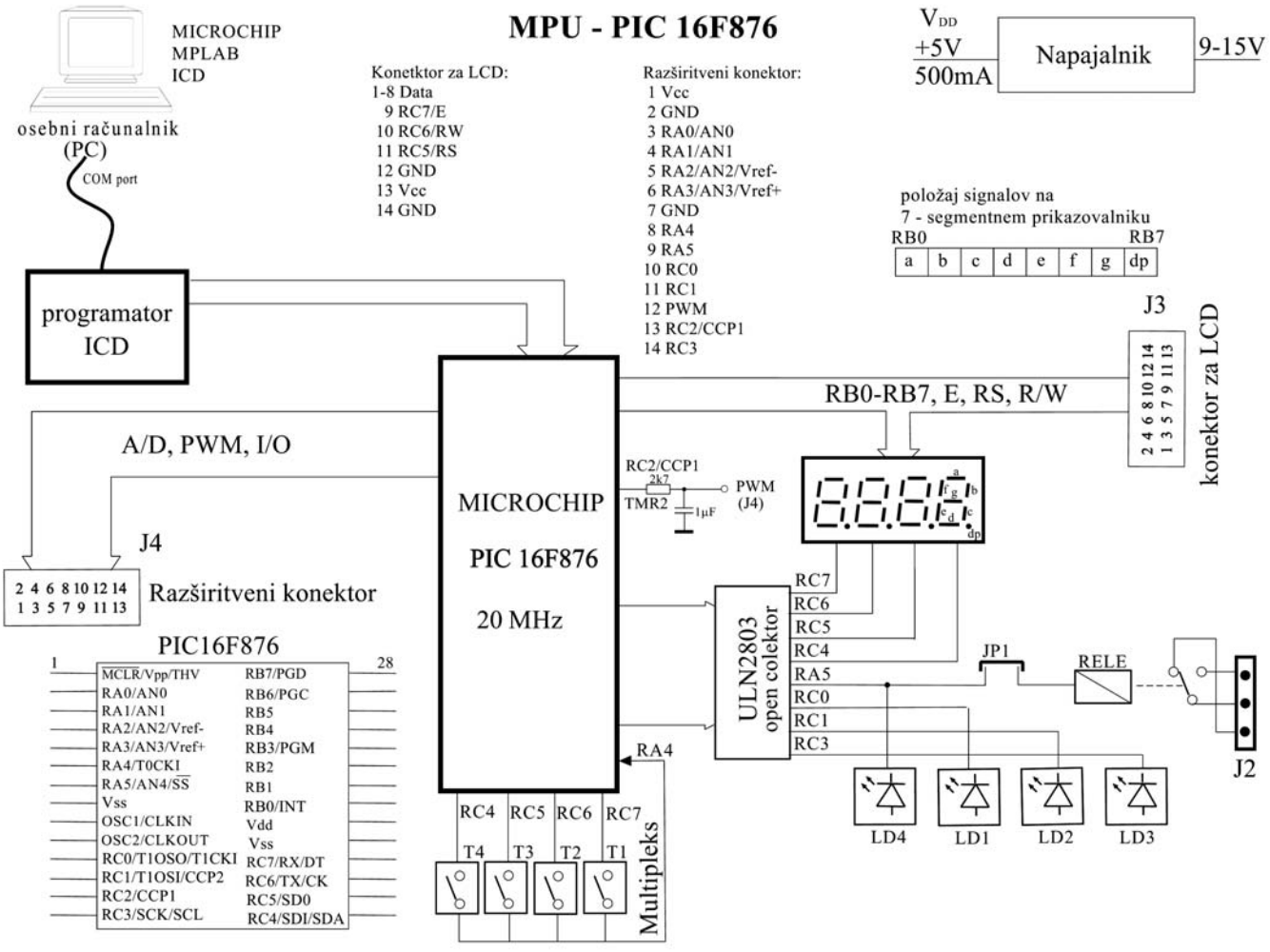
1. Uvod

Modul **MPU-PIC16F876** je namenjen uporabi v pedagoškem procesu. Zgrajen je z uporabo mikrokrmilnika **Microchip PIC16F876**; na kartici pa poleg le-tega najdemo še:

- 4 enote - LED,
- 4 enote - 7-segmentni LED prikazovalnik,
- 4 tipke,
- rele in
- razširitveni konektor (priključki za analogne vhode, PWM izhod, časovniki, ...).

Za programiranje in razhroščanje (kontrolirano izvajanje programa) je predvidena uporaba **ICD(1) modula**.

Zgradbo modula in konfiguracijo ter priključitev elementov predstavlja spodnja slika:



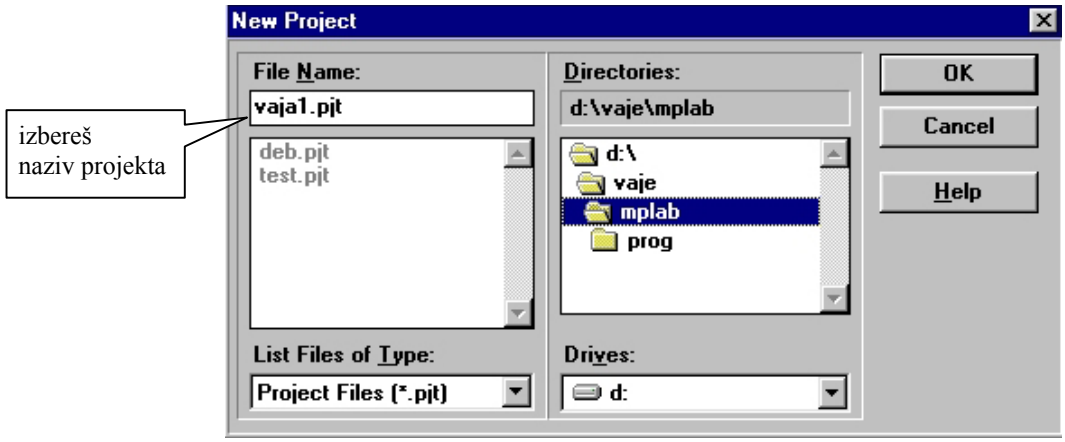
Mikrokrmilnik PIC16F876 programiramo z uporabo programskega orodja ***Microchip MPLAB*** (priporoča se V5.70 za uporabo skupaj z ICD1) v zbirnem jeziku (Microchip Assembler) in/ali v C jeziku (potrebna je instalacija ene od različic prevajalnika).

2. Uporaba programskega orodja Microchip MPLAB

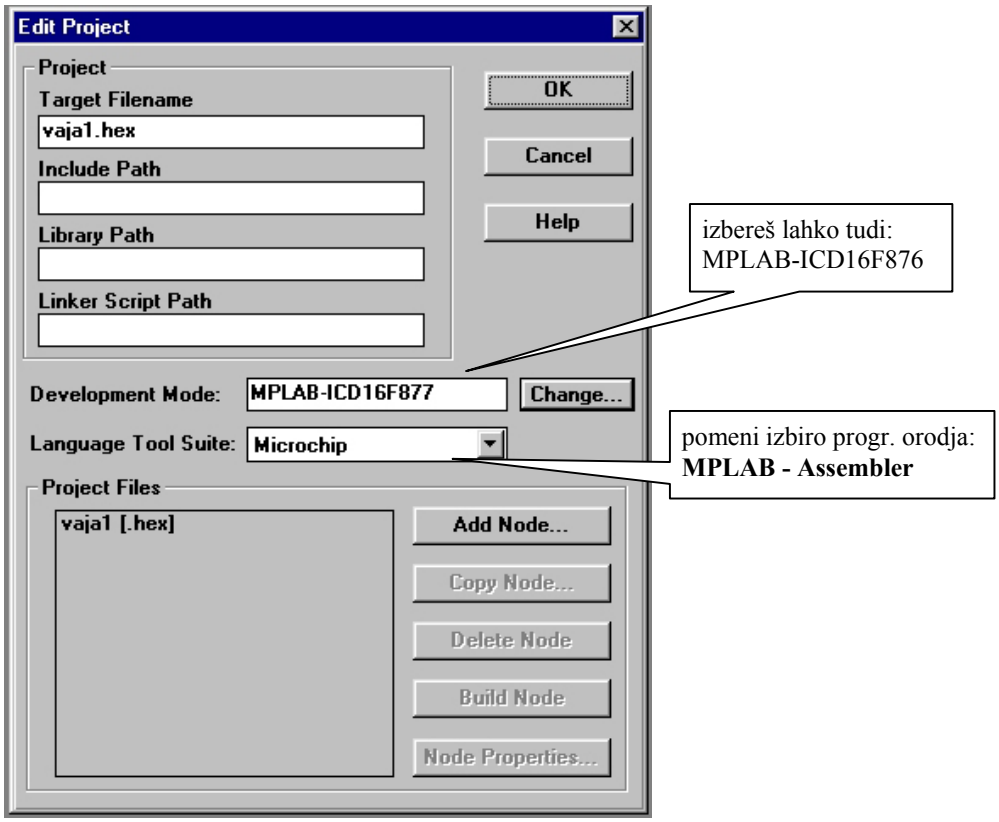
Modul MPU-PIC16F876 poveži z modulom ICD. Pazi, da je kabel ustrezno priključen (glej napise!). Nato modul ICD poveži z računalnikom preko serijskega vodila (COM1 ali COM2, lahko tudi COM4). Nazadnje priključi še napajanje (AC ali DC, 7,5 – 15V).

Poženi program *Microchip MPLAB* (***Start → Programs → Microchip MPLAB → MPLAB***).

Odpri novi projekt (***Project → New Project*** ...), projekt imenuješ poljubno, obvezna končnica **.pjt,** pot (direktorij) odpreš v svojem delovnem imeniku:

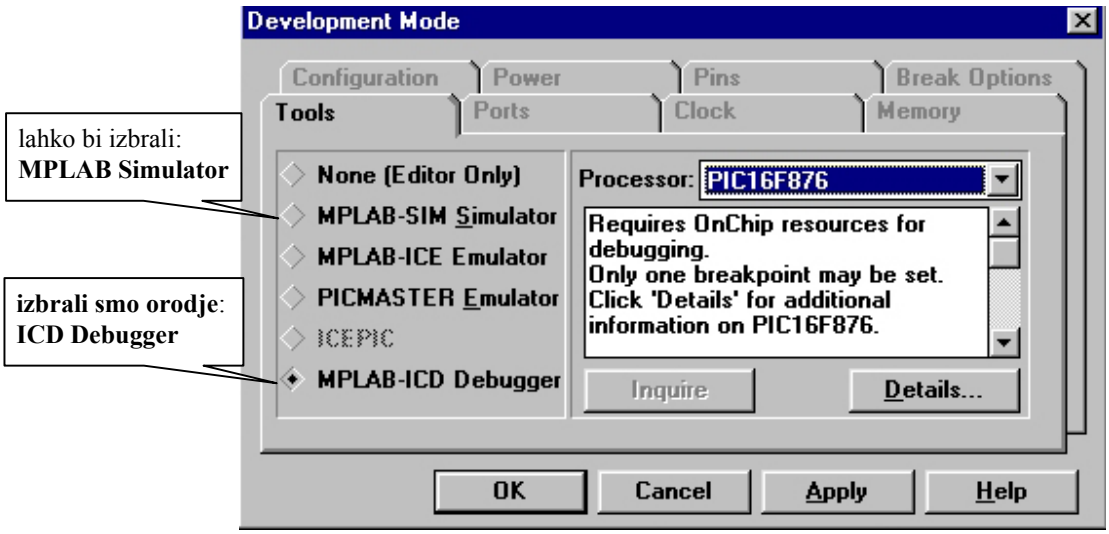


Pritisni OK, odpre se okno ***Edit Project***:



Ustvari se datoteka s končnico **.hex**, z imenom projekta npr.: vaja1.hex (tovrstna datoteka predstavlja preveden program).

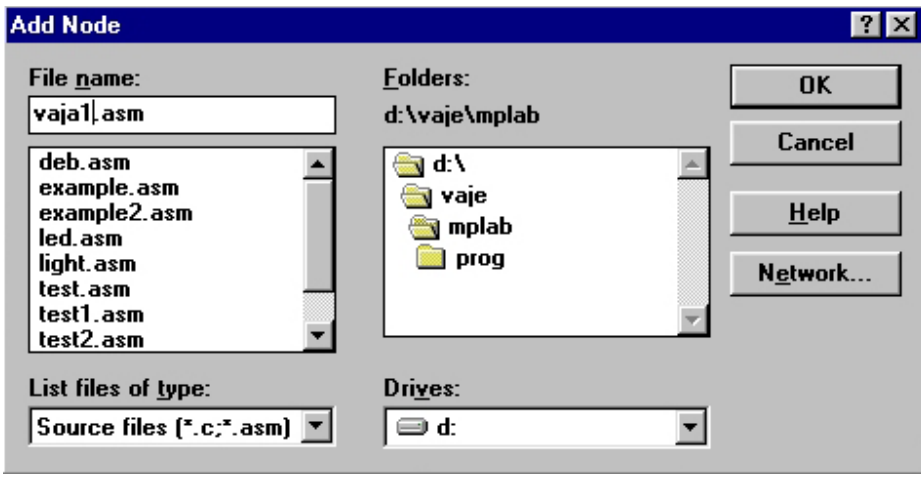
Klikni ***Change*** ..., odpre se okno ***Development Mode***, kjer nastaviš opcije, kot so prikazane na spodnji sliki:



Klikneš **OK** in prikaže se okno ***MPLAB-ICD***:

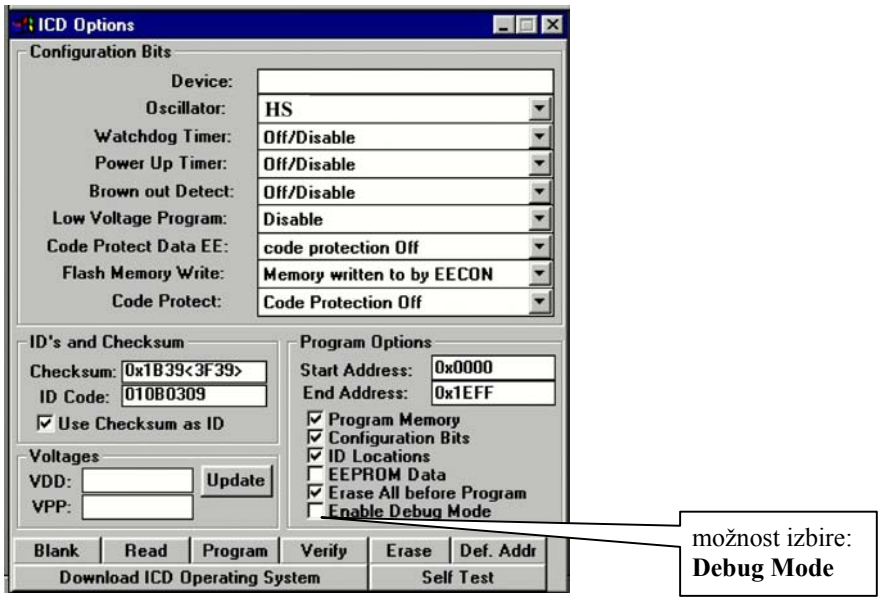
Opomba: Na namiznih osebnih računalnikih so to običajno vrata COM1 ali COM2; na prenosnih računalnikih (brez vgrajenih serijskih vmesnikov) z **dodanim USB vmesnim pretvornikom so to običajno vrata COM4.**

V oknu ***Edit Project*** klikneš še ***Add Node*** in vključiš datoteko, v kateri bo tvoj program s končnico **.asm** (izvorni program v zbirnem jeziku, npr.: vaja1.asm, le-to lahko kasneje urejaš v vključenem urejevalniku MPLAB):



Nato v oknu ***Edit Project*** klikneš **OK** (s tem se zapre).

V oknu *MPLAB-ICD* klikneš ***Option**s* ... in v oknu ***ICD Options*** nastaviš opcije (v desni polovici), kot to kaže spodnja slika.

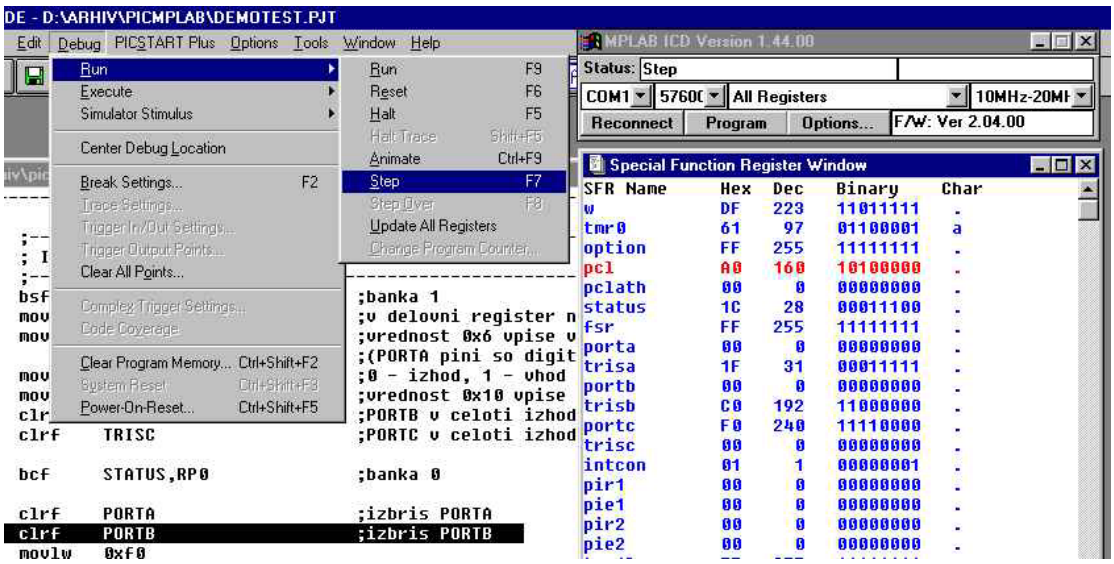


Okno ***ICD Options*** lahko zapreš (desni kot zgoraj).

Zdaj odpreš novo datoteko (***File → New***), **napišeš (ali spremeniš) izvorni program v zbirnem jeziku** (Assembler) ter ga shraniš s ***File → Save as*** ... (v našem primeru **vaja1.asm**).
Program bo tako napisan v zbirnem jeziku (Assembler) in vsebuje obvezno končnico ***.asm**.

1. **Prevajenje** izvedeš tako, da v menuju ***Project*** izbereš opcijo ***Build All***.
2. **Program naložiš** tako, da v oknu ***MPLAB-ICD*** klikneš ***Reconnect*** in nato ***Program***.
3. **Program poženeš** tako, da še enkrat sprožiš prevajanje (***Project → Build All***).
4. **Delovanje programa prekineš** tako, da v oknu *MPLAB-ICD* klikneš ***Reconnect.***

Opomba: Če izbereš možnost ***Enable Debug Mode*** (v oknu ***ICD Options***), **ne izvajaš točk 3 in 4**, kajti MPLAB omogoča kontrolirano izvajanje programa preko ikon ali menijev (***Debug → Run → ...***).
Možnosti: start programa (**Run** F9), ponovni start (**Reset** F6), ustavitev (**Halt** F5), izvajanje po korakih (**Step** F7), vstavljanje prekinitvenih točk (**Break Settings** F2), sledenje registrov (**SFR**) in spremenljivk (**Watch Windows**).



Opozorilo: **Razhroščevalni način** izvajanja programa (***Debug Mode)*** ima **določene pomanjkljivosti**: zasede se del RAM in ROM pomnilnika, za komunikacijo med ICD1 modulom in MPU-PIC16F876 se uporabljata **liniji RB6 in RB7** (pike in črtice na LED prikazovalnikih se naključno spreminjajo, **program ne teče v realnem času**!).

3. Programiranje PIC16F876

Programiranje PIC16F876 lahko poteka v **zbirnem jeziku** (Assembler, integriran v MPLAB) ali v **programskem jeziku C** (potrebno ga je dodatno naložiti).
Nabor ukazov zbirnega jezika obsega **35 ukazov** v skladu z RISC arhitekturo. Opisani so v nadaljevanju.

Pomen simbolov :

k - konstanta
W - delovni register (Work register)
f - pomnilniška lokacija (File register)
<b> - bit (mesto v registru)
d - zastavica za izbiro cilja (W - 0 ali f - 1)
PC - programski števec
TOS - števec sklada
WDT - Watchdog Timer
[labela] - simbolna označba vrstice (v 1. koloni)
C, DC, Z, N – zastavice v STATUS registru (Carry, Decimal Carry, Zero, Negative)

Opis ukazov

ADDLW	**Seštej konstanto in W**
Sintaksa:	[labela] ADDLW k
Operandi:	$0 \le k \le 255$
Operacija:	$(W) + k \rightarrow (W)$
Spreminja status:	C, DC, Z
Opis:	Vsebini registra W se prišteje osem bitna konstanta 'k', rezultat se shrani v register W.

ANDLW	**Konstanta IN W**
Sintaksa:	[labela] ANDLW k
Operandi:	$0 \le k \le 255$
Operacija:	(W) .IN. $k \rightarrow (W)$
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni IN z osem bitno konstanto 'k', rezultat se shrani v register W.

ADDWF	**Seštej W in f**
Sintaksa:	[labela] ADDWF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	$(W) + (f) \rightarrow (cilj)$
Spreminja status:	C, DC, Z
Opis:	Vsebini registra W se prišteje vsebina registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

ANDWF	**W IN f**
Sintaksa:	[labela] ANDWF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(W) .IN. $(f) \rightarrow (cilj)$
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni IN z vsebino registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

BCF	**Zbriši bit b v f**
Sintaksa:	[labela] BCF f,b
Operandi:	$0 \le f \le 127$,
	$0 \le b \le 7$
Operacija:	$0 \rightarrow$ (f<b>)
Spreminja status:	Nobenega
Opis:	Bit 'b' v registru 'f' je zbrisan.

BSF	**Postavi bit b v f**
Sintaksa:	[labela] BSF f,b
Operandi:	$0 \le f \le 127$,
	$0 \le b \le 7$
Operacija:	$1 \rightarrow$ (f<b>)
Spreminja status:	Nobenega
Opis:	Bit 'b' v registru 'f' je postavljen.

BTFSS	**Testiraj bit, preskoči če je '1'**
Sintaksa:	[labela] BTFSS f,b
Operandi:	$0 \le f \le 127$,
	$0 \le b \le 7$
Operacija:	preskoči, če je (f<b>) = 1
Spreminja status:	Nobenega
Opis:	Če je bit 'b' v registru 'f' enak '0', se izvede naslednji ukaz. Če je bit 'b' enak '1', je naslednji ukaz preskočen in se namesto njega izvede NOP, kar povzroči, da se ukaz izvaja dva urina cikla.

BTFSC	**Testiraj bit, preskoči če je '0'**
Sintaksa:	[labela] BTFSC f,b
Operandi:	$0 \le f \le 127$,
	$0 \le b \le 7$
Operacija:	preskoči, če je (f<b>) = 0
Spreminja status:	Nobenega
Opis:	Če je bit 'b' v registru 'f' enak '1', se izvede naslednji ukaz. Če je bit 'b' enak '0', je naslednji ukaz preskočen in se namesto njega izvede NOP, kar povzroči, da se ukaz izvaja dva urina cikla.

CALL	**Klic podprograma**
Sintaksa:	[labela] CALL k
Operandi:	$0 \le f \le 2047$
Operacija:	(PC) + 1 $\rightarrow$ TOS, k $\rightarrow$ PC<10:0>, (PCLATH<4:3>)$\rightarrow$ (PC<12:11>)
Spreminja status:	Nobenega
Opis:	Kliče podprogram. Najprej je naslov povratka iz podprograma (PC+1) shranjen v sklad. Enajst bitni naslov je naložen v bite PC <10:0>. Zgornja bita PC sta naložena iz PCLATH. Ukaz CALL traja dva cikla.

CLRF	**Zbriši f**
Sintaksa:	[labela] CLRF f
Operandi:	$0 \le f \le 127$
Operacija:	00h $\rightarrow$ (f), 1 $\rightarrow$ Z
Spreminja status:	Z
Opis:	Vsebina registra 'f' je zbrisana in bit (Z) v statusnem registru je postavljen na 1.

CLRW	**Zbriši W**
Sintaksa:	[labela] CLRW
Operandi:	Nima
Operacija:	00h → (W), 1 → Z
Spreminja status:	Z
Opis:	Vsebina registra W je zbrisana in bit (Z) v statusnem registru je postavljen na 1.

CLRWDT	**Zbriši Watchdog Timer**
Sintaksa:	[labela] CLRWDT
Operandi:	Nima
Operacija:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{TO}$ 1 → $\overline{PD}$
Spreminja status:	$\overline{TO}$, $\overline{PD}$
Opis:	Ukaz CLRWDT resetira Watchdog Timer. Resetira tudi njegov prescaler. Statusna bita $\overline{TO}$ in $\overline{PD}$ sta postavljena.

COMF	**Komplement f**
Sintaksa:	[labela] COMF f,d
Operandi:	0 ≤ f ≤ 127, d ∈ [0,1]
Operacija:	$(\overline{f})$ → (cilj)
Spreminja status:	Z
Opis:	Izračun komplementa registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

DECF	**Zmanjšaj f za 1**
Sintaksa:	[labela] DECF f,d
Operandi:	0 ≤ f ≤ 127, d ∈ [0,1]
Operacija:	(f) - 1 → (cilj)
Spreminja status:	Z
Opis:	Zmanjšaj register 'f' za 1. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

DECFSZ	**Zmanjšaj f za 1, preskoči če je 0**
Sintaksa:	[labela] DECFSZ f,d
Operandi:	0 ≤ f ≤ 127, d ∈ [0,1]
Operacija:	(f) - 1 → (cilj); preskoči, če je rezultat enak 0
Spreminja status:	Nobenega
Opis:	Zmanjšaj register 'f' za 1. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'. Če je bit 'b' v registru 'f' enak 1, se izvede naslednji ukaz. Če je rezultat enak 0, je naslednji ukaz preskočen in se namesto njega izvede NOP, kar povzroči, da se ukaz izvaja dva urina cikla.

GOTO	**Brezpogojni skok**
Sintaksa:	[labela] GOTO k
Operandi:	0 ≤ f ≤ 2047
Operacija:	k → PC<10:0>, (PCLATH<4:3>)→ (PC<12:11>)
Spreminja status:	Nobenega
Opis:	GOTO sproži brezpogojni skok (vejitev). Enajst bitni naslov je naložen v bite PC <10:0>. Zgornja bita PC sta naložena iz PCLATH. Ukaz GOTO traja dva cikla.

INCF	**Povečaj f za 1**
Sintaksa:	[labela] INCF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(f) + 1 → (cilj)
Spreminja status:	Z
Opis:	Povečaj register 'f' za 1. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

INCFSZ	**Povečaj f za 1, preskoči če je 0**
Sintaksa:	[labela] INCFSZ f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(f) + 1 → (cilj);
	preskoči, če je rezultat enak 0
Spreminja status:	Nobenega
Opis:	Poveča register 'f' za 1. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'. Če je bit 'b' v registru 'f' enak 1, se izvede naslednji ukaz. Če je rezultat enak 0, je naslednji ukaz preskočen in se namesto njega izvede NOP, kar povzroči, da se ukaz izvaja dva urina cikla.

IORLW	**Konstanta ALI W**
Sintaksa:	[labela] IORLW k
Operandi:	$0 \le k \le 255$
Operacija:	(W) .ALI. k → (W)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni ALI z osem bitno konstanto 'k', rezultat se shrani v register W.

IORWF	**W ALI f**
Sintaksa:	[labela] IORWF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(W) .ALI. (f) → (cilj)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni ALI z vsebino registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

MOVF	**Shrani f**
Sintaksa:	[labela] MOVF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(f) → (cilj)
Spreminja status:	Z
Opis:	Vsebina registra f je pomaknjena v ciljni register. Če je d=0, se rezultat shrani v register W, če pa je d=1, se rezultat shrani v register 'f'. Ukaz z d=1 je običajno uporabljen za testiranje registra f, saj vpliva na statusno zastavico Z.

MOVLW	**Shrani k v W**
Sintaksa:	[labela] MOVLW k
Operandi:	$0 \le k \le 255$
Operacija:	k → (W)
Spreminja status:	Nobenega
Opis:	Osem bitna konstanta 'k' se shrani v register W.

MOVWF	**Shrani W v f**
Sintaksa:	[labela] MOVWF f
Operandi:	$0 \le k \le 127$
Operacija:	$(W) \rightarrow (f)$
Spreminja status:	Nobenega
Opis:	Vsebina registra W se shrani v register f.

NOP	**Ni operacije**
Sintaksa:	[labela] NOP
Operandi:	Nima
Operacija:	Ni operacije
Spreminja status:	Nobenega
Opis:	Ni operacije.

RETFIE	**Vrni se iz interrupta**
Sintaksa:	[labela] RETFIE
Operandi:	Nima
Operacija:	TOS $\rightarrow$ PC, 1 $\rightarrow$ GIE
Spreminja status:	Nobenega
Opis:	Vrnitev iz prekinitvene rutine (interrupta).

RETLW	**RETURN s konstanto v W**
Sintaksa:	[labela] RETLW k
Operandi:	$0 \le k \le 255$
Operacija:	k $\rightarrow$ (W), TOS $\rightarrow$ PC
Spreminja status:	Nobenega
Opis:	V register W se naloži konstanta 'k'. V programski števec se naloži vrednost z vrha sklada (naslov za vrnitev iz podprograma). Operacija traja dva cikla.

RETURN	**Vrnitev iz podprograma**
Sintaksa:	[labela] RETURN
Operandi:	Nima
Operacija:	TOS $\rightarrow$ PC
Spreminja status:	Nobenega
Opis:	Vrnitev iz podprograma. V programski števec se naloži vrednost z vrha sklada (naslov za vrnitev iz podprograma). Operacija traja dva cikla.

RLF	**Rotiraj f v levo skozi Carry**
Sintaksa:	[labela] RLF f,d
Operandi:	$0 \le f \le 127$, $d \in [0,1]$
Operacija:	
Spreminja status:	C
Opis:	Vsebina registra f je pomaknjena za eno mesto v levo skozi zastavico statusnega registra Carry. Če je d=0, se rezultat shrani v register W, če pa je d=1, se rezultat shrani v register 'f'.

RRF	**Rotiraj f v desno skozi Carry**
Sintaksa:	[labela] RRF f,d
Operandi:	$0 \le f \le 127$, $d \in [0,1]$
Operacija:	
Spreminja status:	C
Opis:	Vsebina registra f je pomaknjena za eno mesto v desno skozi zastavico statusnega registra Carry. Če je d=0, se rezultat shrani v register W, če pa je d=1, se rezultat shrani v register 'f'.

SLEEP	**SLEEP način delovanja**
Sintaksa:	[labela] SLEEP
Operandi:	Nima
Operacija:	00h → WDT, 0 → WDT prescaler, 1 → $\overline{TO}$ 0 → $\overline{PD}$
Spreminja status:	$\overline{TO}$, $\overline{PD}$
Opis:	Statusni bit $\overline{PD}$, ki označuje status napajanja je zbrisan, statusni bit $\overline{TO}$, ki označuje pretek časa, je postavljen. Watchdog Timer in njegov prescaler sta zbrisana. Procesor se preklopi v način delovanja SLEEP z zaustavljenim oscilatorjem.

SUBLW	**Odštej W od konstante**
Sintaksa:	[labela] SUBLW k
Operandi:	$0 \le k \le 255$
Operacija:	k - (W) → (W)
Spreminja status:	C, DC, Z
Opis:	Vsebina registra W se odšteje (po metodi dvojiškega komplementa) od osem bitne konstante 'k', rezultat se shrani v register W.

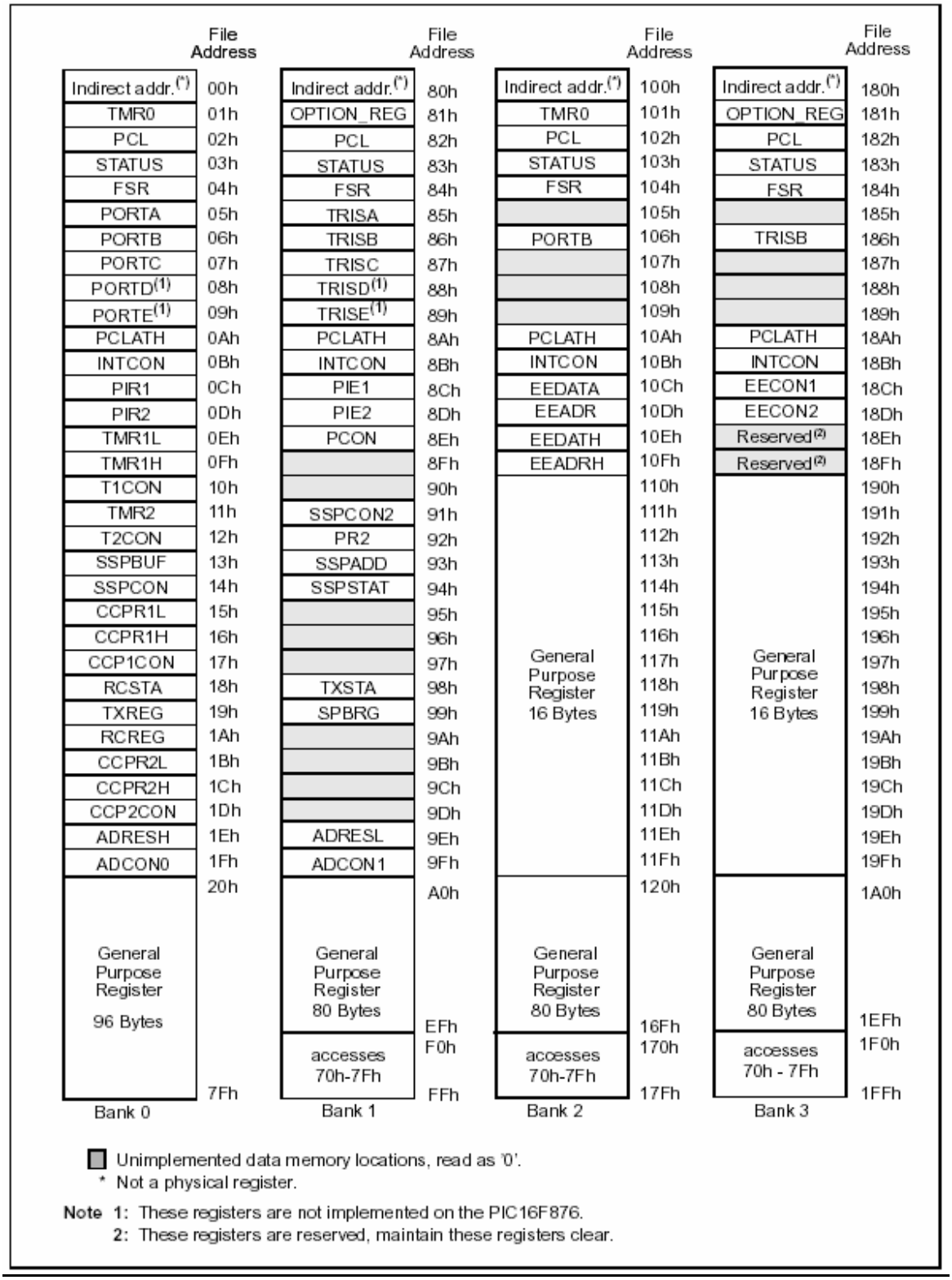
SUBWF	**Odštej W od f**
Sintaksa:	[labela] SUBWF f,d
Operandi:	$0 \le f \le 127$, $d \in [0,1]$
Operacija:	(f) - (W) → (cilj)
Spreminja status:	C, DC, Z
Opis:	Vsebina registra W se odšteje (po metodi dvojiškega komplementa) od vsebine registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

SWAPF	**Zamenjaj nibbla v f**
Sintaksa:	[labela] SWAPF f,d
Operandi:	$0 \le f \le 127$, $d \in [0,1]$
Operacija:	(f<3:0>) → (cilj<7:4>), (f<7:4>) → (cilj<3:0>)
Spreminja status:	C, DC, Z
Opis:	Zgornji in spodnji nibble (štirje biti) sta zamenjana. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

XORLW	**Konstanta XOR W**
Sintaksa:	[labela] XORLW k
Operandi:	$0 \le k \le 255$
Operacija:	(W) .XOR. k → (W)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni XOR (ekskluzivni ali) z osem bitno konstanto 'k', rezultat se shrani v register W.

XORWF	**W IN f**
Sintaksa:	[labela] XORWF f,d
Operandi:	$0 \le f \le 127$, $d \in [0,1]$
Operacija:	(W) .XOR. (f) → (cilj)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni XOR (ekskluzivni ali) z vsebino registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

4. Organizacija pomnilnika (posebni in splošni registri)



Statusni register

Statusni register vsebuje aritmetični status aritmetične logične enote (ALU), resetni status in bite za izbiro segmenta (banke) v pomnilniku. Statusni register lahko uporabimo kot cilj v katerem koli ukazu, če je ukaz takšen, da vpliva na bite Z, C ali DC, le-teh ni mogoče spreminjati, bitov **$\overline{\text{TO}}$** in **$\overline{\text{PD}}$** pa ni mogoče prepisati.

Naslov	Ime	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Vrednost pri: POR,BOR	Vrednost pri vseh drugih resetih
03h	STATUS	IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	0001 1xxx	000q quuu

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	TO	PD	Z	DC	C
bit 7							bit 0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bit 7: **IRP:** Bit za izbiro bank registrov **(**Register Bank Select bit, uporabljen za posredno naslavljanje)
- 1 = Banka 2, 3 (100h - 1FFh)
- 0 = Banka 0, 1 (00h - FFh)

bit 6-5: **RP1:RP0**: Bit za izbiro banke registrov **(**Register Bank Select bits, uporabljen za neposredno naslavljanje)
- 11 = Banka 3 (180h - 1FFh)
- 10 = Banka 2 (100h - 17Fh)
- 01 = Banka 1 (80h - FFh)
- 00 = Banka 0 (00h - 7Fh)

Vsaka banka ima maksimalno 128 bytov.

bit 4: **$\overline{\text{TO}}$** : Bit za prekoračitev časa (Time-out bit)
- 1 = Po vklopu napajanja, ukazu CLRWDT ali SLEEP
- 0 = Pojavila se je prekoračitev časa WDT

bit 3: **$\overline{\text{PD}}$** : Bit za izklop napajanja (Power-down bit)
- 1 = Po vklopu ali z ukazom CLRWDT
- 0 = Z izvedbo ukaza SLEEP

bit 2: **Z**: Ničelni bit (Zero bit)
- 1 = rezultat aritmetične a li logične operacije je nič
- 0 = rezultat aritmetične a li logične operacije ni nič

bit 1: **DC**: Bit za prenos med nibbloma (Digit carry/ **$\overline{\text{borrow}}$** bit, ukazi ADDWF, ADDLW, SUBLW, SUBWF)

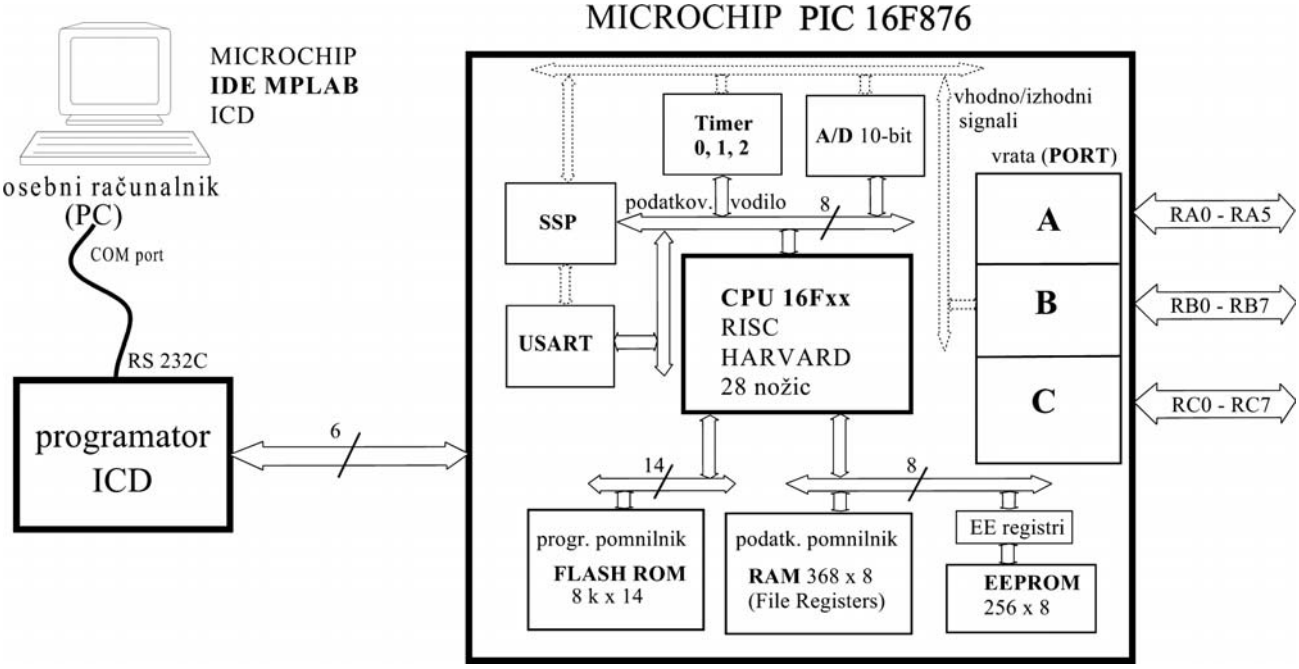
(za **$\overline{\text{borrow}}$** je polariteta obrnjena)
- 1 = Pojavil se je prenos s četrtega bita.
- 0 = Prenos s četrtega bita se ni pojavil.

bit 0: **C**: Bit za prenos/ $\overline{\text{izposojo}}$ (Carry/ **$\overline{\text{borrow}}$** bit, ukazi ADDWF, ADDLW, SUBLW, SUBWF)
- 1 = Prišlo je do prenosa iz najvišjega bita (MSB)
- 0 = Ni prišlo do prenosa iz najvišjega bita (MSB)

Opomba: Za **$\overline{\text{borrow}}$** je polariteta obrnjena. Odštevanje je izvedeno s prištevanjem dvojiškega komplementa drugega operanda. Za ukaze za rotacije (RRF, RLF) se ta bit naloži z bodisi najvišjim ali najnižjim bitom registra, ki ga rotiramo.

5. Vhodno/izhodne enote

Mikrokrmilnik PIC16F876 ima tri vhodno/izhodna vrata - porte: PORTA, PORTB in PORTC, preko katerih so dostopni vhodno/izhodni signali vsebovanih vmesnikov. Vrednosti na njih lahko beremo oz. postavljamo s postavljanjem vrednosti istoimenskih registrov, konfiguriramo pa jih z nastavitvijo vrednosti v registrih TRISA, TRISB in TRISC.



PORTA:

PORTA ima 6 vhodno/izhodnih priključkov, katerih obnašanje (logični vhod ali izhod) nastavljamo z registrom TRISA. Postavitev ustreznega **bita TRISA na 1** bo iz priključka na PORTA **naredila vhod**, **postavitev na 0 pa izhod**. Priključek RA4 je multipleksiran z vhodom časovnika Timer0. Ostali priključki PORTA so multipleksirani z analognimi vhodi in analognim vhodom ter analognim vhodom VREF. Njihovo delovanje je določeno z nastavitvijo/brisanjem ustreznih bitov v registru ADCON1. Register TRISA določa smer delovanja RA priključkov tudi, ko so le-ti konfigurirani kot analogni vhodi, torej je treba zagotoviti, da so priključki tudi v tem primeru konfigurirani kot vhodi (ustrezni biti v TRISA so postavljeni na 1).
Ob vklopu se priključki konfigurirajo kot analogni vhodi in imajo vrednost 0.

Funkcije PORTA:

Ime	Bit#	Buffer	Funkcija
RA0/AN0	bit0	TTL	Vhod/izhod ali analogni vhod
RA1/AN1	bit1	TTL	Vhod/izhod ali analogni vhod
RA2/AN2	bit2	TTL	Vhod/izhod ali analogni vhod
RA3/AN3/VREF	bit3	TTL	Vhod/izhod ali analogni vhod ali VREF
RA4/TOCKI	bit4	ST	Vhod/izhod ali vhod za zunanji takt za Timer0 Izhod je z odprtim ponorom
RA5/$\overline{SS}$/AN4	bit5	TTL	Vhod/izhod ali analogni vhod

Legenda: TTL = TTL vhod, ST = vhod s Schmittovim Triggerjem

S PORTA povezani registri:

Naslov	Ime	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Vrednost ob POR, BOR	Vrednost ob drugih resetih
05h	PORTA	-	-	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	-	-	PORTA register smeri podatkov						--11 1111	-- 11 1111
9Fh	ADCON1	ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legenda: x = neznan, u = nespremenjen, - = neuporabljene lokacije, beri kot '0', PORTA ne uporablja osenčenih lokacij

PORTB:

PORTB ima 8 vhodno/izhodnih priključkov, katerih obnašanje (logični vhod ali izhod) nastavljamo z registrom TRISB. Postavitev ustreznega bita TRISB na 1 bo iz priključka na PORTB naredila vhod, postavitev na 0 pa izhod. Trije biti so multipleksirani s funkcijo za programiranje pri nizki napetosti (RB3/PGM, RB6/PGC in RB7/PGD). Priključek RB0 je multipleksiran z zunanjim interruptom.

Funkcije PORTB:

Ime	Bit#	Buffer	Funkcija
RB0/INT	bit0	TTL/ST	Vhod/izhod ali zunanji priključek za interrupt
RB1	bit1	TTL	Vhod/izhod
RB2	bit2	TTL	Vhod/izhod
RB3/PGM	bit3	TTL	Vhod/izhod ali priključek za programiranje pri nizki napetosti
RB4	bit4	TTL	Vhod/izhod
RB5	bit5	TTL	Vhod/izhod
RB6/PGC	bit6	TTL/ST	Vhod/izhod ali priključek za programiranje pri nizki napetosti
RB7/PGD	bit7	TTL/ST	Vhod/izhod ali priključek za programiranje pri nizki napetosti

Legenda: TTL = TTL vhod, ST = vhod s Schmittovim Triggerjem

S PORTB povezani registri:

Naslov	Ime	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Vrednost ob POR, BOR	Vrednost ob drugih resetih
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB register smeri podatkov								1111 1111	1111 1111
81h, 181h	OPTION_REG	$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legenda: x = neznan, u = nespremenjen, - = neuporabljene lokacije, beri kot '0', PORTB ne uporablja osenčenih lokacij

PORTC:

PORTC ima 8 vhodno/izhodnih priključkov, katerih obnašanje (logični vhod ali izhod) nastavljamo z registrom TRISC. Postavitev ustreznega bita TRISC na 1 bo iz priključka na PORTC naredila vhod, postavitev na 0 pa izhod.

Funkcije PORTC:

Ime	Bit#	Buffer	Funkcija
RC0/T1OSO/T1CKI	bit0	ST	Vhod/izhod ali izhod oscilatorja za Timer1 / vhod takta za Timer1
RC1/T1OSI/CCP2	bit1	ST	Vhod/izhod ali vhod oscilatorja za Timer1 ali vhod Capture2 / izhod Compare 2 / **izhod PWM2**
RC2/CCP1	bit2	ST	Vhod/izhod ali vhod Capture1 / izhod Compare 1 / **izhod PWM1**
RC3/SCK/SCL	bit3	ST	Vhod/izhod ali sinhronski serijski CLOCK za SPI in I^2C
RC4/SDI/SDA	bit4	ST	Vhod/izhod ali podatkovni vhod SPI / I^2C
RC5/SDO	bit5	ST	Vhod/izhod ali sinhronski serijski podatkovni izhod
RC6/TX/CK	bit6	ST	Vhod/izhod ali USART asinhronski Transmit ali sinhronski clock
RC7/RX/DtT	bit7	ST	Vhod/izhod ali USART asinhronski Receive ali sinhronski podatki

Legenda: TTL = TTL vhod, ST = vhod s Schmittovim Triggerjem

S PORTC povezani registri:

Naslov	Ime	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Vrednost ob POR, BOR	Vrednost ob drugih resetih
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC register smeri podatkov								1111 1111	1111 1111

Legenda: x = neznan, u = nespremenjen

Primer konfiguracije vrat in uporabe vhodov/izhodov:

```
        include <p16f876.inc>       ; vključitev datoteke z definicijami simbolov registrov

        list    p=16f876            ; izbira tipa čipa

        bsf     STATUS,RP0          ; kodna stran 1
        movlw   0x00                ; PORTB in PORTC - sami izhodi
        movwf   TRISB               ; PORTB – RB(0:7) : izhodi
        movwf   TRISC               ; PORTC – RC(0:7) : izhodi
        bsf     STATUS,RP0          ; BANK 1
        movlw   B'00011011'         ; RA(2, 5): izhodi, RA(0, 1, 3, 4): vhodi
        movwf   TRISA
                                    ; za PORTA je potrebno v ADCON1 vpisati B'10000100' ali 0x84
        movlw   B'10000100'         ; B'10000100'
        movwf   ADCON1              ; ADCON1 = B'10000100' : desno poravnan zapis, RA(0,1,3) : analogni vhodi
        bcf     STATUS,RP0          ; kodna stran 0
        movlw   B'11111111'         ; nastavitev stanja na izhodih PORTC
        movwf   PORTC               ; na PORTC
        movlw   B'00001111'         ; nastavitev stanja na izhodih PORTB
        movwf   PORTB               ; na PORTB

label1  movlw   0xef                ;
        movwf   PORTB               ; zapiši na PORTB
        bsf     PORTA,5             ; postavi bit 5 PORTA na 1
        goto label1                 ; izvajaj zanko, skoči na label1
        end
```

Primer programa v zbirnem jeziku

```
;-------------------------------------------------------------------------------
; Demo program za vklop LED diod: LD1 na RC0 in LD3 na RC3
;-------------------------------------------------------------------------------

        list    p=16f876  ;izbira tipa cipa
        include <p16f876.inc>  ;vkljucitev datoteke z definicijami simbolov
;-------------------------------------------------------------------------
; Nastavitev vektorjev

        org     0x00      ;RESET vektor
        goto    Main      ;skok na glavni program Main
        org     0x05      ;zacetek programske kode
Main
;--------- Inicializacija registrov vhodno/izhodnih vmesnikov
        bsf     STATUS,RP0          ;banka 1
        movlw   0x06      ;v delovni register nalozi vrednost 0x06
        movwf   ADCON1    ;ne uporabljamo analognih vhodov (00000110)
        movlw   0x10      ;0 - izhod, 1 – vhod, b'00010000'
        movwf   TRISA     ;vsi pini porta A so izhodni, razen RA4

        clrf    TRISB     ;vsi pini porta B so izhodni
        clrf    TRISC     ;vsi pini porta C so izhodni
        bcf     STATUS,RP0          ;banka 0
        clrf    INTCON    ;prekinitev ne bomo uporabljali
        clrf    PORTC     ;izhode porta C postavi na 0

;-------- jedro programa
Zanka
        bsf     PORTC,0   ;vklop LED diode LD1 na RC0
        bsf     PORTC,3   ;vklop LED diode LD3 na RC3
        goto    Zanka     ;skok na zacetek

        end               ;psevdo ukaz za konec programa
```

Primer programa v C-jeziku

```
/* Demo program za vklop LED diod: LD1 na RC0 in LD3 na RC3 */

#include <pic1687x.h> // vkljucitev datoteke z definicijami simbolov

void main(void)
{

/* --------- Inicializacija registrov vhodno/izhodnih vmesnikov */
ADCON1=0x06;   // ne uporabljamo analognih vhodov, 0b00000110
TRISA=0x10;    // vsi pini porta A so izhodni, razen RA4, 0b00010000)
TRISB=0;       // vsi pini porta B so izhodni
TRISC=0;       // vsi pini porta C so izhodni
PORTC=0;    // izhode porta C postavi na 0
INTCON=0;   // prekinitev ne bomo uporabljali

/* -------- jedro programa */

        while (1)   // neskoncna zanka
        {
        RC0=1;     // vklop LED diode LD1 na RC0
        RC3=1;     // vklop LED diode LD3 na RC3
        }

}
```

Analogni vhodi

Opis analognih vhodov bo omejen na primer uporabe. Podrobnosti najdete v obširnejšem priročniku.
A/D pretvornik je 10-bitni, vsebuje 5-kanalni multiplekser, modul pa ima tudi pozitivni in negativni referenčni vhod.

Registri A/D modula so:
- register višjih bitov rezultata A/D pretvorbe (A/D Result High Register, **ADRESH**)
- register nižjih bitov rezultata A/D pretvorbe (A/D Result Low Register, **ADRESL**)
- A/D kontrolni register 0 (A/D Control Register 0, **ADCON0**)
- A/D kontrolni register 1 (A/D Control Register 1, **ADCON1**)

ADCON0 nadzoruje delovanje A/D modula, **ADCON1** pa nastavi funkcije priključkov. Registra **ADRESH:ADRESL** vsebujeta 10-bitni rezultat pretvorbe. Ko je A/D pretvorba zaključena, se rezultat naloži v ta registra, bit **GO/$\overline{\text{DONE}}$** (ADCON0<2>) je postavljen na '0', bit ADIF se postavi na '1'.

Uporaba pretvornika je opisana v naslednjih točkah:

1. **Konfiguriraj A/D modul:**
   - Konfiguriraj analogne priključke / napetostne reference in digitalne vhode/izhode (**ADCON1**)
   - Izberi vhodni kanal A/D pretvornika (**ADCON0**)
   - Izberi takt A/D pretvorbe (**ADCON0**)
   - Vključi A/D modul (**ADCON0**)
2. Po potrebi konfiguriraj prekinitve, ki jih proži A/D pretvornik
   - Zbriši bit **ADIF**
   - Postavi bit **ADIE**
   - Postavi bit **GIE**
3. Počakaj, da se inicializacija izvede
4. **Poženi pretvorbo**
   - postavi bit **GO/$\overline{\text{DONE}}$** (**ADCON0**)
5. **Počakaj, da se A/D pretvorba konča, obstajata dva načina:**
   - **Čakaj, da se zbriše bit GO/$\overline{\text{DONE}}$ ali**
   - Čakaj na A/D prekinitev
6. **Preberi rezultat v registrih (ADRESH:ADRESL), po potrebi zbriši bit ADIF.**
7. Za naslednjo pretvorbo se vrni v korak 1 ali 2 (kateri ustreza).

Register ADCON0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	**ADCS0**	**CHS2**	**CHS1**	**CHS0**	**GO/** $\overline{\text{DONE}}$	**—**	**ADON**
bit7	6	5	4	3	2	1	bit0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bit 7-6: **ADCS1:ADCS0: bita za izbiro takta A/D pretvorbe**
- 00 = FOSC/2
- 01 = FOSC/8
- 10 = FOSC/32
- 11 = FRC (zunanji RC oscilator)

bit 5-3: **CHS2:CHS0: biti za izbiro analognega kanala**
- 000 = kanal 0, (RA0/AN0)
- 001 = kanal 1, (RA1/AN1)
- 010 = kanal 2, (RA2/AN2)
- 011 = kanal 3, (RA3/AN3)
- 100 = kanal 4, (RA5/AN4)

bit 2: **GO/$\overline{\text{DONE}}$: Statusni bit A/D pretvorbe**

Če je ADON = 1
- 1 = A/D pretvorba poteka (postavitev tega bita sproži A/D pretvorbo)
- 0 = A/D pretvorba ne poteka (ta bit se avtomatsko postavi na '0' ko je A/D pretvorba končana)

bit 1: **Neuporabljen**: beri kot '0'

bit 0: **ADON**: A/D On bit
- 1 = A/D pretvornik deluje
- 0 = A/D pretvornik je izključen

Register ADCON1

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	**-**	**-**	**-**	**PCFG3**	**PCFG2**	**PCFG1**	**PCFG0**
bit7							bit0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bit 7: **ADFM: Izbira formata rezultata A/D pretvorbe**
- 1 = **Desno poravnan**. 6 najbolj pomembnih (most significant) bitov **ADRESH** se prebere kot '0'.
- 0 = **Levo poravnan**. 6 najmanj pomembnih (least significant) bitov **ADRESL** se prebere kot '0'.

bit 6-4: **Neuporabljeni: Beri kot '0'**
bit 3-0: **PCFG3:PCFG0:** Biti za konfiguracijo A/D Porta (Configuration Control bits)

PCFG3: PCFG0	AN7 (1) RE2	AN6(1) RE1	AN5 (1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN / Refs (2)
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	**D**	**D**	**D**	**D**	**A**	**D**	**A**	**A**	**VDD**	**VSS**	**3/0**
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	**D**	**D**	**D**	**D**	**D**	**D**	**D**	**D**	**VDD**	**VSS**	**0/0**
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

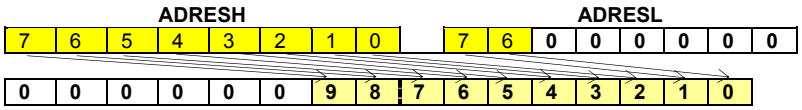
Legenda: A = analogni vhod, D = digitalni vhod/izhod

Opomba 1: Ti kanali na čipih z 28 priključki (16F876) niso na voljo, ampak pri PIC mikrokrmilnikih s 40 nožicami (16F877)
2: Stolpec označuje število analognih vhodov in število analognih vhodov, ki jih uporabljamo kot vhode za referenčno napetost

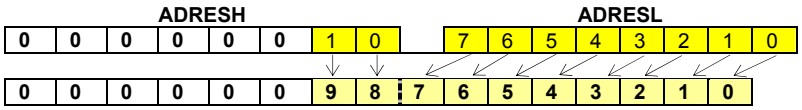
Vkolikor se zadovoljimo z **8-bitno vrednostjo rezultata (0 do 255)**, izberemo levo poravnavo in rezultat prenesemo iz registra **ADRESH** v 8-bitno spremenljivko. Pri tem je ločljivost A/D pretvorbe enaka 19,53 mV (pri območju 5,0 V).

Če je pomemben popoln **10-bitni rezultat (0 do 1023)**, je primernejša izbira **desne poravnave**, postopek shranjevanja pa je prikazan spodaj. Pri tem je ločljivost A/D pretvorbe enaka 4,88 mV (pri območju 5,0 V).

Levo poravnan rezultat shrani v 16-bitno spremenljivko:



Desno poravnan rezultat shrani v 16-bitno spremenljivko:



Primer programa v zbirnem jeziku

```
;-------------------------------------------------------------------------------------------------
; Naziv programa: Demo program za AD pretvorbo: AN3 shrani v RezH in
;  RezL ;   v casu AD pretvorbe vklopljena LD4 na RA5 izhodu
;-------------------------------------------------------------------------------------------------

            list        p=16f876   ;izbira tipa cipa
            include <p16f876.inc>  ;vkljucitev datoteke z definicijami simbolov
;-------------------------------------------------------------------------------------------------
;--- sprememnljivke
RezH        equ         0x20       ;zgornjih 8 bitov rezultata
RezL        equ         0x21       ;spodnjih 8 bitov rezultata
;--- Nastavitev vektorjev
            org         0x00                    ;RESET vektor
            goto        Main                    ;skok na glavni program Main
            org         0x05                    ;zacetek programske kode
Main
;--------- Inicializacija registrov vhodno/izhodnih vmesnikov
            bsf         STATUS,RP0              ;banka 1
            movlw       b'00011011'             ;0 - izhod, 1 - vhod
            movwf       TRISA                   ;RA5-izhod, RA4-vhod,
RA3/AN3-vhod, RA2-izhod, RA1/AN1-vhod, RA0/AN0-vhod
            clrf        TRISB                   ;vsi pini porta B so izhodni
            clrf        TRISC                   ;vsi pini porta c so izhodni
;--- inicializacija A/D - AN3
            movlw       b'10000100'             ;desna poravnava, vhod AN3 je
                                                analogni, tudi AN1 in AN0
            movwf       ADCON1                  ;bit7=1 -> desna poravnava 10-
                                                bitnega rezultata
            bcf         STATUS,RP0              ;nazaj v banko 0
            movlw       b'10011001'             ;takt (f/32,takt=20MHz), izbira
                                    anal. kan. (AN3), AD pretv. je omogocen
            movwf       ADCON0
;--------------------
            clrf        INTCON                  ;prekinitev ne bomo uporabljali
            clrf        PORTC                   ;izhode porta C postavi na 0

;-------- jedro programa
Zanka
            bsf         PORTA,5                 ;vklopi LD4 na RA5
;--- Start AD pretvorbe in cakanje na rezultat
            bsf         ADCON0,2                ;start AD pretvorbe (postavi bit
                                                2 (GO_DONE) na 1
pocakaj     btfsc       ADCON0,2 ;testiraj GO_DONE
            goto        pocakaj                 ;cakanje na izvedbo AD
                                                pretvorbe (pribl. 20 us)
;--- pretvorba je koncana in rezultat pripravljen
            bcf         PORTA,5                 ;izklopi LD4 na RA5
            movf        ADRESH,w                ;zgornjih 8-bitov (dejansko le 2
                                                bita!) rezultata -> W
            movwf       RezH                    ;premik visjih osmih bitov iz
                                        delovnega registra w v sprem RezH
            bsf         STATUS,RP0              ;v banko1, kjer je ADRESL
            movf        ADRESL,w                ;spodnjih 8-bitov rezultata -> w
            bcf         STATUS,RP0              ;nazaj v banko0
            movwf       RezL                    ;premik nizjih 8-bitov v   RezL

Test        nop                                 ;«Test« simbol za Breakpoint

            goto        Zanka                   ;ponovitev AD pretvorbe

            end                         ;psevdo ukaz za konec programa
```

Primer programa v C-jeziku

```
/* Demo program za AD pretvorbo vhoda AN3, rezultat shrani v 16-bit sprem.
Rez ; v casu AD pretvorbe je vklopljena LD4 na RA5 izhodu */

#include <pic1687x.h>  /* vkljucitev datoteke z definicijami simbolov */

unsigned int Rez;   /* sprem. za rezultat pretvorbe */

void main(void)
{

/* --------- Inicializacija registrov vhodno/izhodnih vmesnikov */

TRISA=0x1b;             /*b'00011011' RA5-izhod, RA4-vhod, RA3/AN3-vhod,
                        RA2-izhod, RA1/AN1-vhod, RA0/AN0-vhod */
TRISB=0;                /* vsi pini porta B so izhodni */
TRISC=0;                /* vsi pini porta c so izhodni */
/*--- inicializacija A/D - AN3 */
ADCON1=0x84;            /* b'10000100' desna poravnava, vhod AN3 je
                        analogni, tudi AN1 in AN0 */
ADCON0=0x99;            /* b'10011001' takt(f/32,takt=20 MHz), izbira anal. kan.
                        (AN3) */

PORTC=0;                /* izhode porta C postavi na 0 */
INTCON=0;               /* prekinitev ne bomo uporabljali */

/* -------- jedro programa */

        while (1) /* neskoncna zanka */
        {
        RA5=1;    /* vklop LD4 */
        /*--- Start AD pretvorbe in cakanje na rezultat */
        ADGO=1;   /* start AD pretvorbe, postavi bit 2 (GO_DONE) na 1 */
        while (ADGO==1); /* cakanje na izvedbo AD pretvorbe (~ 20 us) */

        RA5=0;    /* izklop LD4 */
        Rez = ADRESL|(ADRESH<<8);    /* rezultat AD pretvorbe */

        asm("test nop");   /* moznost nastav. Breakpoint na simb. "test" */


        }               /* ponovitev AD pretvorbe */

}
```

6. Časovnik Timer0 in periodično generinje prekinitev

Modul **Timer0** je 8-bitni časovnik/števec z naslednjimi lastnostmi:

- vsebuje 8-bitni programsko izbirljiv preddelilnik (prescaler PSA);
- ima možnost izbire med notranjim (urin takt, npr.: 20 MHz) in zunanjim taktom (možnost izbire prehoda);
- sproži zastavico (**T0IF**) in/ali prekinitveno zahtevo (**Interrupt**) ob napolnitvi števca (prehod iz FFh na 00);
- standardna enota vseh Microchip PIC mikrokrmilnikov (npr.: 16F84) srednje kategorije (Mid Range).

Z modulom **Timer0** so povezani registri:

- **TMR0** (01h) – 8-bitni števni register (vpišemo lahko 8-bitno število: (256-vrednost), register vedno **prišteva** prehode takta do napolnitve, zatem se register inkrementira od vrednosti 00 oz. vpisane vrednosti);
- **OPTION_REG** (81h) – register z biti za nastavitev preddelilnika in kontrolnimi biti timerja;
- **INTCON** (0Bh) – register s kontrolnimi in statusnimi biti prekinitvenih zahtev.

Register OPTION_REG (na naslovu 81h oz 01h v segmentu Bank1)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	**INTEDG**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**
bit7	6	5	4	3	2	1	bit0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bit 7: **$\overline{\text{RBPU}}$**: **PORTB** "Pull-up" upori - izbira (1 – upori niso vključeni, 0 – upori so vključeni)

bit 6: **INTEDG**: izbira prehoda signala za prekinitev preko RB0/INT vhoda (1 – naraščajoči, 0 – padajoči)

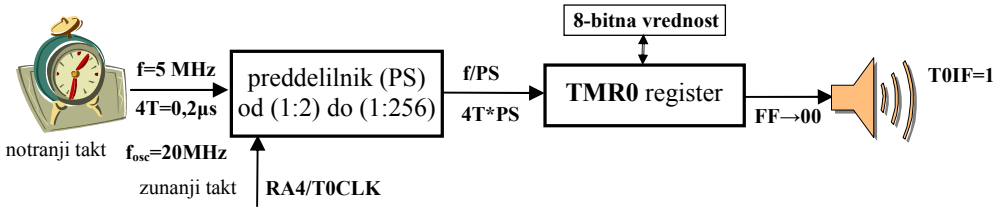
bit 5: **T0CS**: izbira takta za **TMR0** (1 – zunanji takt na RA4/T0CKL, **0** – notranji takt iz CLKOUT)

bit 4: **T0SE**: izbira prehoda signala za **TMR0** (1 – padajoči, **0** – naraščajoči na RA4/T0CKL zunanjem taktu)

bit 3: **PSA**: dodelitev preddelilnika (1 – **WDT** "Watch Dog Timer" časovni stražnik1, **0** – **Timer0**)

biti 2-0: **PS2:PS0**: biti za izbiro preddelilnika

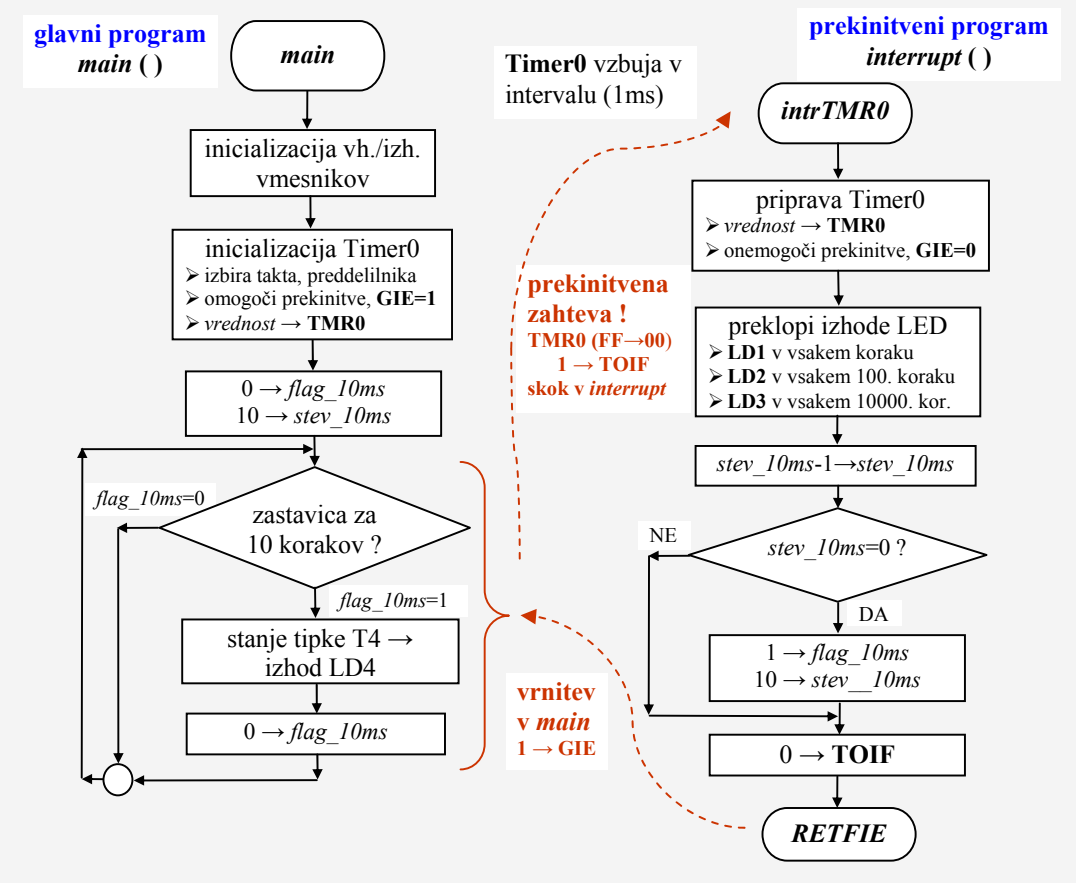
PS2,PS1,PS0	TMR0 (PS)	WDT vrednost
0 0 0	1:2	1:1
0 0 1	1:4	1:2
0 1 0	1:8	1:4
0 1 1	1:16	1:8
1 0 0	1:32	1:16
1 0 1	1:64	1:32
1 1 0	1:128	1:64
1 1 1	1:256	1:128



Največji možni doseg (časovni interval) časovnika **TMR0** je: 256*256*0,2us=13,1ms (pri izbiri preddelilnika 1:256, začetni vrednosti TMR0=0 in taktu 20 MHz). Večjo ločljivost nastavitve časovnega intervala dosežemo pri čim manjši vrednosti preddelilnika.

Ob vsaki napolnitvi števca se samodejno aktivira zastavica (**T0IF**=1) in hkrati tudi sproži prekinitvena zahteva (če je omogočena). V prekinitvenem strežnem programu je potrebno programsko zbrisati zastavico (**T0IF**=0), preden ponovno omogočimo prekinitve.

$Timer0_{Interval}$ = (256-**TMR0**) * PS * 4 * $T_{osc,,}$ pri čemer je T_{osc} = 50 ns, če je frekvenca urinega takta 20 MHz.



Ob začetku prekinitvenega strežnega programa je priporočljivo (vkolikor ne uporabljamo drugih prekinitvenih virov) **zbrisati bit za omogočanje prekinitev** (**GIE=0**), zato da so med izvajanjem prekinitvenega programa vse prekinitve blokirane in s tem ni nevarnosti ponovnega aktiviranja prekinitvene zahteve, še preden se strežni program konča. Le-ta mora biti **vedno zaključen z ukazom RETFIE**, ki povzroči vrnitev v prej prekinjeni program in ponovno omogočenje prekinitev (ob tem se samodejno postavi bit **GIE=1**).

Register INTCON (na naslovu 0Bh v vseh segmentih)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	**PEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**
bit7	6	5	4	3	2	1	bit0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bit 7: **GIE**: Globalni prekinitveni bit (**1** – omogoči vse nemaskirane prekinitve, **0** – onemogoči vse prekinitve)

bit 6: **PEIE**: bit za omogočanje prekinitev periferije (1 – omogoči vse nemaskirane prekinitve periferije, **0** – onemogoči)

bit 5: **T0IE**: omogočanje prekinitev **TMR0** (**1** – omogoči TMR0 prekinitev, 0 – onemogoči ali maskiraj)

bit 4: **INTE**: omogoči zunanjo prekinitev RB0/INT (1 – omogoči, **0** – onemogoči ali maskiraj)

bit 3: **RBIE**: omogoči prekinitve ob spremembi na RB vratih (1 – omogoči, **0** – maskiraj)

bit 2: **T0IF**: zastavica prekoračitve števca **TMR0** (**1** – prekoračitev ali prehod iz FFh na 0 dosežena, 0 – ni prekoračitve)

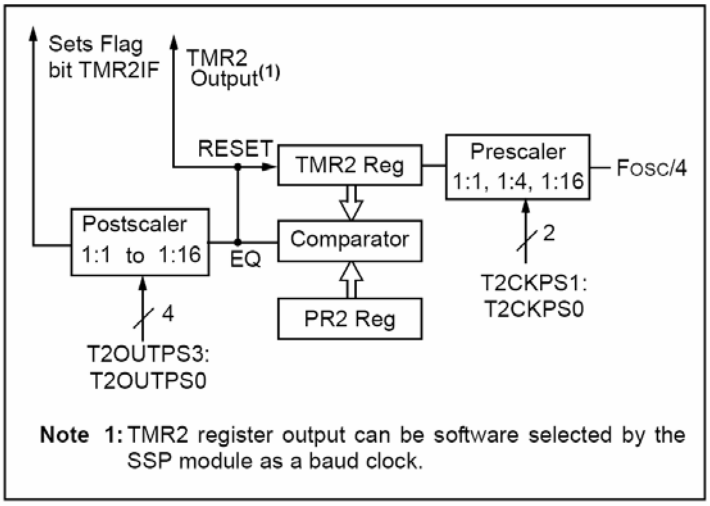
bit 1: **INTF**: zastavica prekinitve zaradi RB0/INT vhoda (1 – se je zgodila, 0 – ni)

bit 0: **RBIF**: zastavica prekinitve na RB4 - RB7 vhodih (1 – se je zgodila, 0 – ni)

7. Časovnik Timer2 in generiranje pulzno-širinskih signalov (PWM)

Modul **Timer2** je 8-bitni časovnik/števec z vgrajenim preddelilnikom (Prescaler) in izhodnim delilnikom (Postscaler). Lahko se uporablja tudi kot PWM časovna baza za generiranje dveh PWM signalov (z enako periodo) v okviru CCP podsklopa.

Zgradba modula:

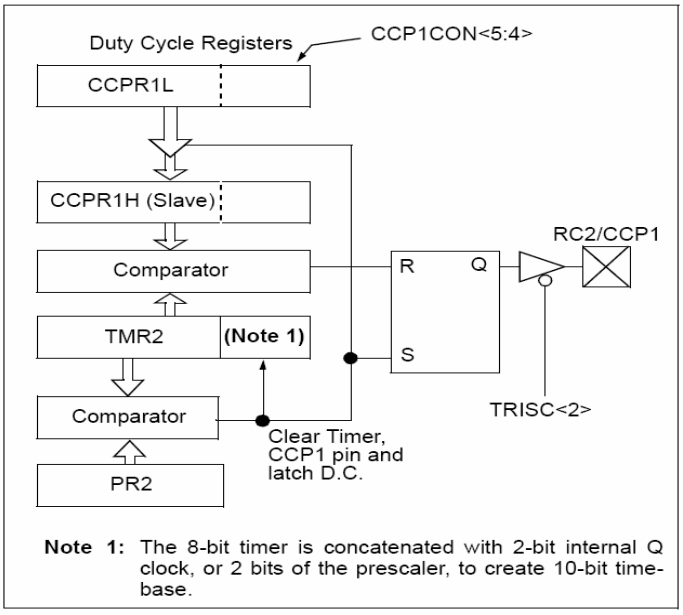


CAPTURE/COMPARE/PWM (**CCP1** in **CCP2**) je podsklop modula **Timer2**, ki vsebuje 16-bitne registre in omogoča delovanje v naslednjih načinih:

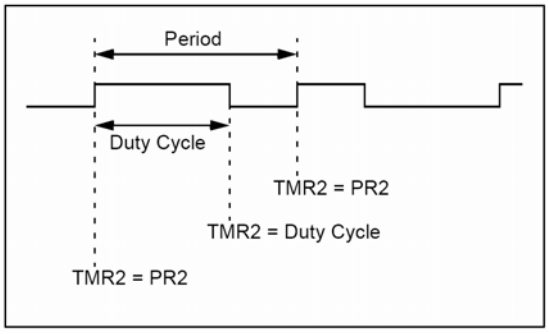
- 16-bitni zajemalni (Capture) register
- 16-bitni primerjalni (Compare) register
- **PWM register širine pulza** (Master/Slave Duty cycle)

PWM način delovanja omogoča, da na izhodih **CCP1** in **CCP2** programsko generiramo pulznoširinska signala z nastavljivo (10-bitno) širino pulzov pri vnaprej izbrani frekvenci (le-ta je enaka za oba sklopa).

Poenostavljena shema **PWM1** sklopa:



PWM izhodni signal:



Izhod za **PWM1**: priključek **RC2/CCP1**

Izhod za **PWM2**: priključek **RC1/CCP2**

Periodo PWM signala nastavimo z vpisom 8-bitne vrednosti v register PR2. Izračunamo jo po naslednji formuli:

$PWM_{Perioda}$ = (**PR2**+1) * 4 * T_{osc} * $TMR2_{Prescaler}$, pri čemer je T_{osc} = 50 ns, če je frekvenca urinega takta 20 MHz.

Frekvenco (obratna vrenost $PWM_{perioda}$) lahko izbiramo od nekaj 100 Hz do nekaj 100 kHz (spodnja tabela), pri čemer se zmanjša ločljivost nastavitve širine pulza pod 10-bitov, če je frekvenca večja od 20 KHz.

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12kHz	156.3 kHz	208.3 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFFh	0xFFh	0xFFh	0x3Fh	0x1Fh	0x17h
Maximum Resolution (bits)	10	10	10	8	7	5.5

Širino pulza (Duty cycle) nastavimo z vpisom višjih 8-bitov v register **CCPR1L** in spodnjih dveh bitov 10-bitne vrednosti v register **CCP1CON** (bita 5 in 4) v skladu z naslednjo formulo:

$PWM_{Duty\ cycle}$ = (**CCPR1L:CCP1CON**<5:4>) * T_{osc} * $TMR2_{Prescaler}$

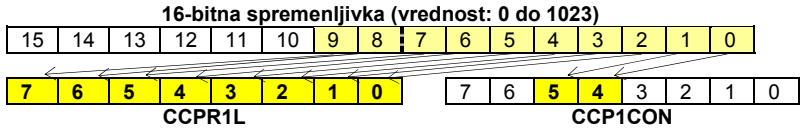
Postopek konfiguriranja CCP modula za PWM način delovanja:

1. Nastavitev PWM periode (frekvence) z vpisom **8-bitne vrednosti** v register **PR2**;
2. Nastavitev PWM širine pulza z vpisom višjih 8-bitov v register **CCPR1L** in spodnjih dveh bitov **10-bitne vrednosti** v register **CCP1CON** (bita 5 in 4)
3. Nastavitev **CCP1 priključka kot izhod** z brisanjem bita 2 (RC2=0) v registru **TRISC**;
4. Nastavitev **TMR2** preddelilnika in omogočitev Timer2 z vpisom ustrezne vrednosti (npr.: 04h) v register **T2CON**;
5. Konfiguriranje CCP1 modula za PWM način delovanja z vpisom (npr.: 0Ch) v register **CCP1CON**;
6. Ponavljanje točke 2, ko želimo spremeniti širino pulza.

Navedeni postopek velja za **PWM1** sklop, ki ima izhod na priključku **CCP1/RC2**. Če želimo uporabiti tudi **PWM2** sklop, dobimo izhodni signal na priključku **CCP2/RC1**, registra v točki 2 in 5 (oz. formuli) pa nadomestimo z **CCPR2L** in **CCP2CON**.

Vkolikor smo zadovoljni z **8-bitno ločljivostjo**, je dovolj da 8-bitno vrednost (med 0 in 255) prenesemo v register **CCPR1L**, medtem ko bita 5 in 4 v registru CCP1CON pustimo na izhodiščni vrednosti (0).

Če želimo generirati PWM signal s polno **10-bitno ločljivostjo** (med 0 in 1023), naložimo vrednost iz 16-bitne spremenljivke **v oba registra:**



Opis registrov CCP1CON oz. CCP2CON (na naslovih 17h oz. 1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	-	**CCPxX**	**CCPxY**	**CCPxM3**	**CCPxM2**	**CCPxM1**	**CCPxM0**
bit7	6	5	4	3	2	1	bit0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bita 7-6: neuporabljena

bita 5-4: **CCPxX: CCPxY**: **spodnja dva bita 10-bitne vrednosti širine pulza PWM sklopa**

biti 3-0: **CCPxM3- CCPxM0**: izbira načina delovanja CCPx sklopa, za **PWM način: 11xx**

Opis registra T2CON (na naslovu 12h)

u-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	**TOUTPS3**	**TOUTPS2**	**TOUTPS1**	**TOUTPS0**	**TMR2ON**	**T2CKPS1**	**T2CKPS0**
bit7	6	5	4	3	2	1	bit0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bit 7: neuporabljen

biti 6-3: **TOUTPS3: TOUTPS0**: delilnik frekvence za izhod Timer2 (od 0000 - 1:1 do 1111 - 1:16) – **nima vpliva na PWM**

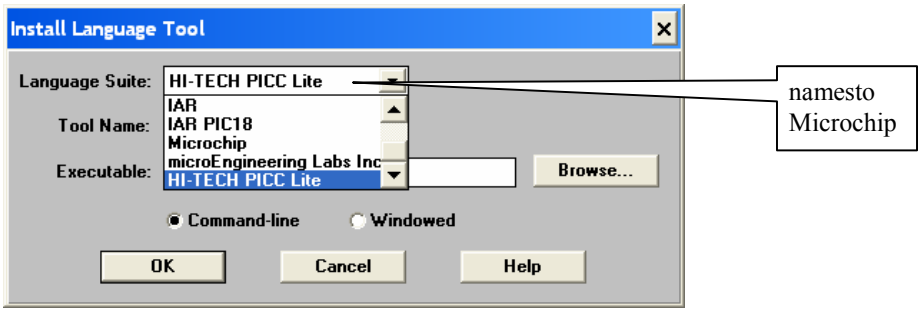
biti 2: **TMR2ON**: **omogočitev Timer2** (**1 - Timer2 aktiven**, 0 - neaktiven)

bita 1-0: **T2CKPS1- T2CKPS0**: **izbira predelilnika** za Timer2 v **PWM načinu: 00 – 1x,** 01 – 4x, 10 ali 11 – 16x

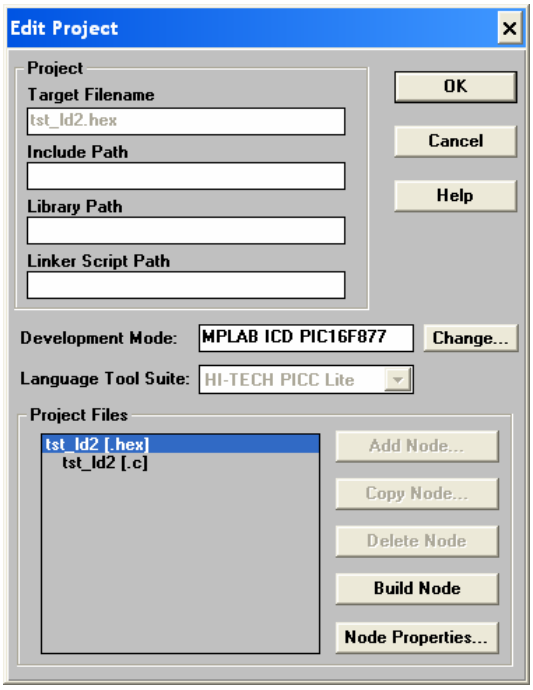
8. Uporaba programskega jezika C

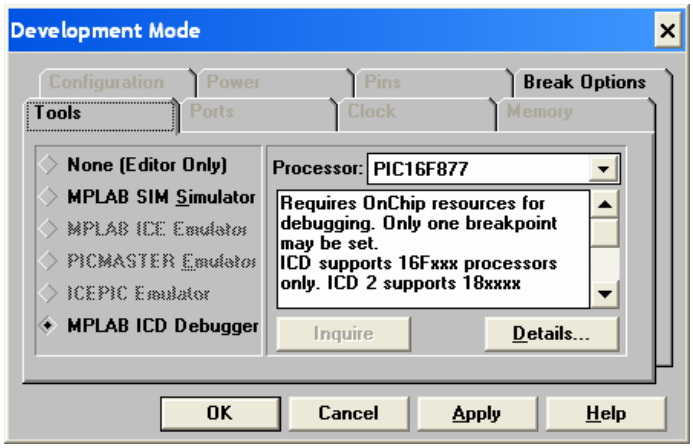
Na voljo so mnoge različice C prevajalnikov različnih proizvajalcev. Za nekomercialno uporabo je zelo primeren HI-TEC C prevajalnik **PICC Lite** (v nadaljevanju bo opisana različica: *HI-TECH PICC Lite Compiler Release notes v8.02PL1*).
To je popolnoma delujoč C prevajalnik z nekaj (manj pomembnimi) omejitvami: omogoča le 16F877 ali 16F84, dolžina programa največ 2 kW, možnost uporabe pomnilnika za spremenljivke samo v prvih dveh segmentih, izpisi števil v *float* formatu niso mogoči.

Program se po instalaciji naloži (privzeto) na disk C: v imenik **C:\PICCLITE**. Samodejno se integrira v IDE okolje MPLAB. To preverimo s: (***Project → Install Language Tool***).



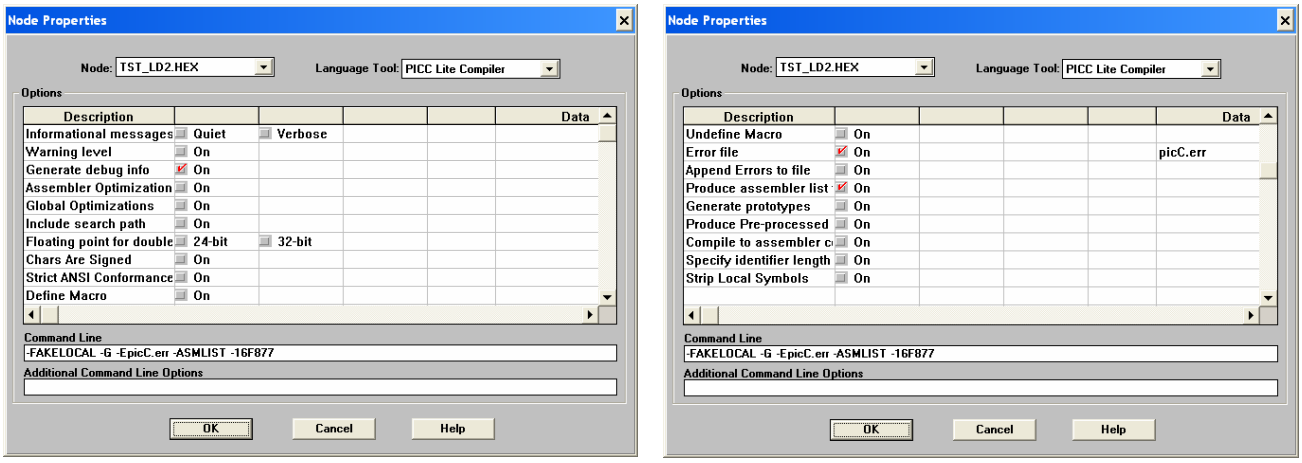
V meniju (***Project → New...*** *ali* ***Edit Project...***) odpremo ali modificiramo projekt ***.pjt**, pri čemer **najprej** izberemo **Language Tool Suite: HI-TECH PICC Lite**. Zatem **obvezno izberemo PIC16F877** (izbiramo lahko sicer med **ICD** razvojnim okoljem in med **MPLAB SIM** simulatorjem).





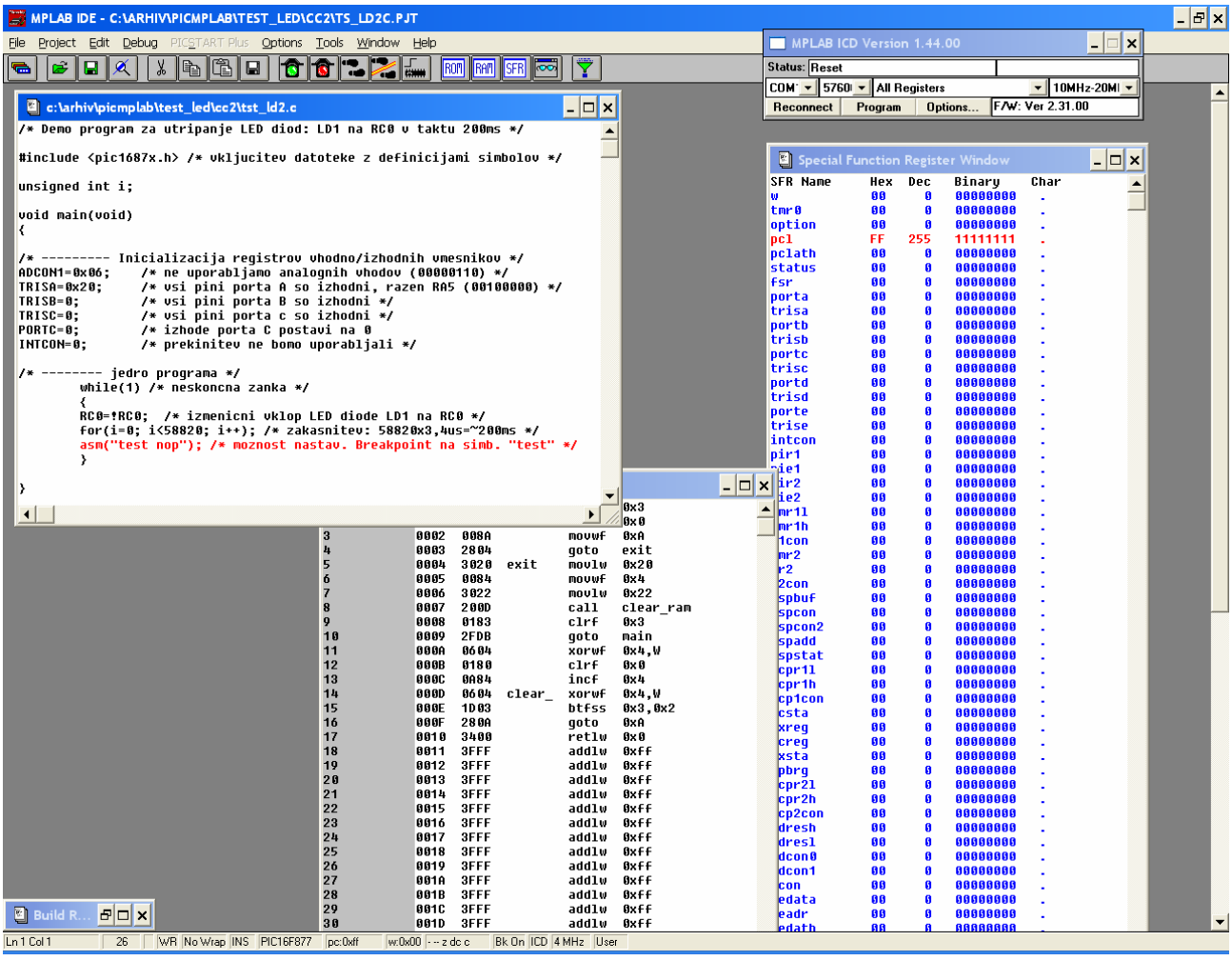
Označimo datoteko s končnico [.hex] in odpremo okno **Node Properties**….
V oknu lahko izberemo **Generate Debug Info** (koristno pri testiranju z **ICD** v **Debug Mode** ali z **MPLAB SIM**, prav tako lahko izberemo: **Produce assembler listening** (izpis prevedenega C – programa v različici za zbirni jezik).

Obvezno izberemo rubriko **Error File** in v koloni **Data** vpišemo ime datoteke (npr.: *picC.err*), kamor se bodo po prevajanju vpisovala sporočila o morebitnih napakah.



Zatem v oknu **Edit project** izberemo **Add node…**, ter vključimo ime izvornega programa v C jeziku npr.: *tst_ld2.c*. Temu se takoj prilagodi tudi ime datoteke z izvršilno kodo: *tst_ld2.hex*.

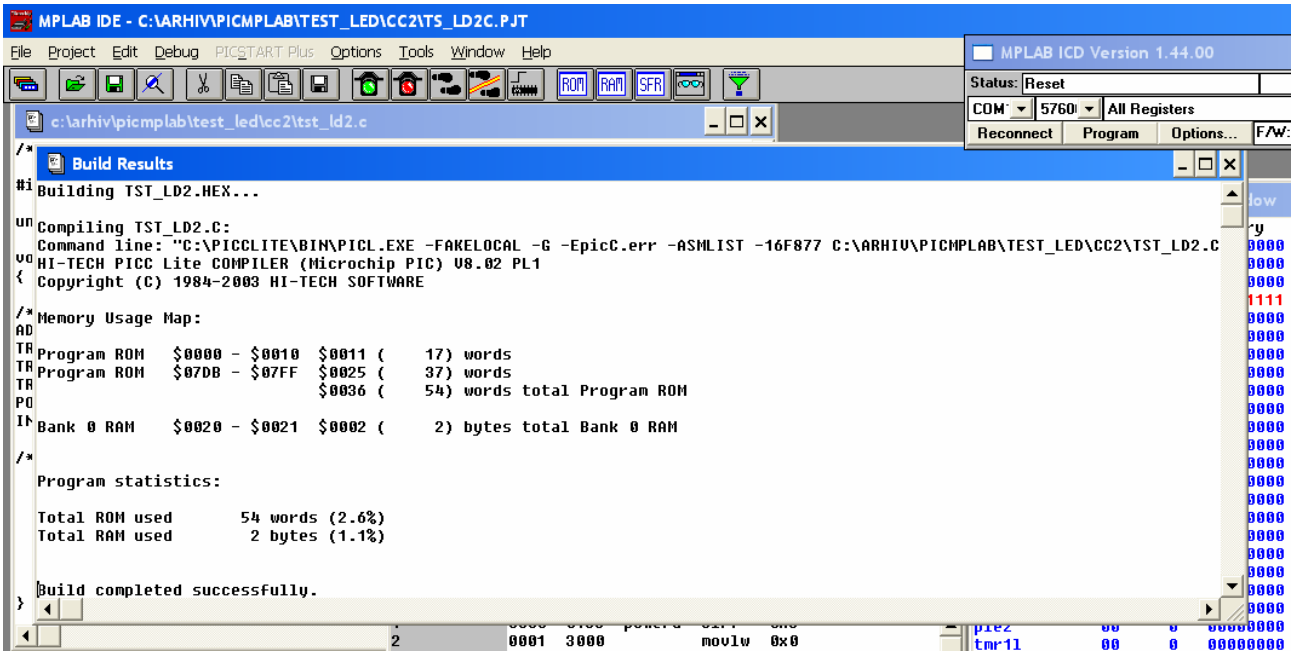
Primer izpisa izvornega programa ***.c** (levo), okno vsebine programskega pomnilnika s programom v strojni kodi (inverzni zbirnik) in okno s prikazom vsebine **SFR** registrov (desno).



Program v C jeziku prevedemo z ukazom: (***Project → Build all***).

Vkolikor ni sintaktičnih napak, se izpiše sporočilo o lastnostih prevedenega programa z zaključno vrstico: **Build completed successfully**. Nadaljujemo z izvajanjem programa (simulator) ali z nalaganjem (download) v Flash ROM ciljnega mikrokrmilnika (enako kot pri programiranju z Microchip zbirnikom) s pomočjo **ICD** (***Reconnect, Program***). Če v oknu **Options...** nimamo izbran **Debug Mode**, še enkrat sprožimo ***Build all***, kar povzroči start programa (sprostitev Reseta).

Vkolikor se izpiše sporočilo **Build failed**, je potrebno odpreti in proučiti datoteko (npr.: *picC.err*) s sporočili o napakah, le-te odpraviti in ponavljati zgornji postopek, dokler nismo odpravili vseh sintaktičnih napak v izvorni kodi.



Prekinitvena rutina (ogrodje)

```
;-------------------------------------------------------------------------------------------------------------------------------
list    p=16f876                        ; izbira tipa čipa
        include <p16f876.inc>           ; vključitev datoteke z definicijami simbolov registrov
        #define PC          0x2         ; definicija naslova PC

BCKW    equ     0x20                    ; definicija naslova spremenljivke - rezerva (backup za W)
BCKST   equ     0x21                    ; definicija naslova spremenljivke - rezerva (backup za STATUS)

;-------------------------------------------------------------------------------------------------------------------------------
; Tukaj nastavim reset vektor
;-------------------------------------------------------------------------------------------------------------------------------
        org     0x0                     ; RESET vektor
        goto    Main                    ; skok na glavni program, Main
        org     0x4                     ; prekinitveni vektor, Interrupt
        goto    Int                     ; skok v prekinitveno rutino - Int
        org     0x5                     ; začetek programske kode

;-------------------------------------------------------------------------------------------------------------------------------
; Prekinitveni program
;-------------------------------------------------------------------------------------------------------------------------------
RefDisp
        ;-----------------------------------------------------------------------------------------------------------------------
                                        ; Vpiši program, ki se izvede ob prekinitvi - časovno zahtevne operacije
        ;-----------------------------------------------------------------------------------------------------------------------
        return

;-------------------------------------------------------------------------------------------------------------------------------
; Prekinitvena rutina
;-------------------------------------------------------------------------------------------------------------------------------
Int                                     ; Shranjevanje stanja delovnega in statusnega registra
        movwf   BCKW                    ; backup za delovni register
        swapf   STATUS, W               ;
        clrf    STATUS                  ; brisanje statusnega registra
        movwf   BCKST                   ; backup za status
        ;-----------------------------------------------------------------------------------------------------------------------
        ; Prekinitveni program
        ;-----------------------------------------------------------------------------------------------------------------------
        call    RefDisp                 ; skok v rutino prekinitvenega programa
        bcf     INTCON, T0IF            ; prekinitveni program je bil izveden
        ;-----------------------------------------------------------------------------------------------------------------------
                                        ; Vzpostavitev starega stanja
        swapf   BCKST, W                ;
        movwf   STATUS                  ; vzpostavi stari status
        swapf   BCKW, F                 ;
        swapf   BCKW, W                 ; vzpostavi staro stanje delovnega registra
        retfie

;-------------------------------------------------------------------------------------------------------------------------------
; Glavni program
;-------------------------------------------------------------------------------------------------------------------------------
Main
        ;-----------------------------------------------------------------------------------------------------------------------
                                        ; Vpiši ukaze za inicializacijo (vhodno/izhodna enota, ...)
        ;-----------------------------------------------------------------------------------------------------------------------
        ;------------------------------; Konfiguracija OPTION_REG (za prekinitve)
        bcf     STATUS,RP1              ;
        bsf     STATUS,RP0              ; BANK 1
        movlw   0x3                     ; preddelilnik 1/16, pull-up vklopljeni, notranji pulzi
        movwf   OPTION_REG
        ;-----------------------------------------------------------------------------------------------------------------------
        movlw   B'10100000'             ; nastavitev prekinitev ( GIE  = 1 , T0IE = 1 )
        movwf   INTCON
        bcf     STATUS,RP0              ; BANK 0
        ;-----------------------------------------------------------------------------------------------------------------------
        ;-----------------------------------------------------------------------------------------------------------------------
                                        ; Vpiši glavni progam - časovno nezahtevne operacije
        ;-----------------------------------------------------------------------------------------------------------------------
        end                             ; konec programa (psevdoukaz)
;-------------------------------------------------------------------------------------------------------------------------------
```