UVOD V PROGRAMIRANJE MIKROKRMILNIKOV

Učni vodnik in navodila za vaje

Janez Pogorelc

UM-FERI, 2005

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko

http://www.feri.uni-mb.si

Inštitut za Robotiko

http://www.ro.feri.uni-mb.si

http://www.hipulab.uni-mb.si/

CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004.42(076.1)

POGORELC, Janez
Uvod v programiranje mikrokrmilnikov : učni vodnik in navodila za vaje / Janez Pogorelc. - 1. izd. - [Maribor] : Fakulteta za elektrotehniko, računalništvo in informatiko : Inštitut za robotiko, 2005

ISBN 86-435-0695-8

COBISS.SI-ID 54574081

ISBN 86-435-0695-8
9 788643 506950

Naslov: **UVOD V PROGRAMIRANJE MIKROKRMILNIKOV, Učni vodnik in navodila za vaje**
Prva izdaja, marec 2005
Avtor: višji predavatelj mag. Janez Pogorelc, univ. dipl. inž.
Strokovni recenzent: doc. dr. Martin Terbuc, univ. dipl. inž., Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko
Vrsta učnega gradiva: zbrana gradivo za predavanja
Uredil in oblikoval: Janez Pogorelc
Slike: Janez Pogorelc
Oblikovanje naslovnice za CD zgoščenko: Janez Pogorelc
Naklada: 11 izvodov CD zgoščenk
Obseg: 31 strani
Izdala: Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Inštitut za robotiko
Tisk: Janez Pogorelc
ISBN 86-435-0695-8

Kazalo

Seznam slik

Seznam tabel

Seznam programov

Namen in cilji učnega gradiva

Učno gradivo predstavlja dopolnitev »zbranega gradiva za predavanje« z naslovom **Uvod v programiranje mikrokrmilnikov** in sicer kot učni vodnik z vprašanji in odgovori za utrjevanje ter priročnik za uporabo programskih orodij Microchip MPLAB in C-prevajalnika za programiranje 8-bitnih PIC mikrokrmilnikov srednje kategorije. Namenjeno je predvsem študentom 2. letnika visokošolskega strokovnega programa Elektrotehnika, smer avtomatika pri podpori vajam za predmeta Mikroelektronika in Mikroračunalniški sistemi.

V uvodu je na kratko predstavljen mikrokrmilniški učni komplet **MPU-PIC16F876**, ki je bil razvit na Inštitutu za robotiko UM-FERI za podporo pedagoškemu procesu. Sledijo tabela z opisom vseh ukazov zbirnega jezika in tabela s prikazom datotečnih registrov.

V nadaljevanju sledi podroben opis dela z integriranim programirnim okoljem Microchip **MPLAB V5.70** v kombinaciji z razvojnim kompletom **ICD(1)** ter napotki za konfiguriranje orodja za programiranje bodisi v zbirnem jeziku, bodisi v C-jeziku.

Del gradiva je namenjen tudi »učnemu vodniku«, napotkom za iskanje dodatnih virov ter vprašanjem za utrjevanje po poglavjih, kot je to širše opisano v »zbranem gradivu za predavanje« z naslovom **Uvod v programiranje mikrokrmilnikov.**

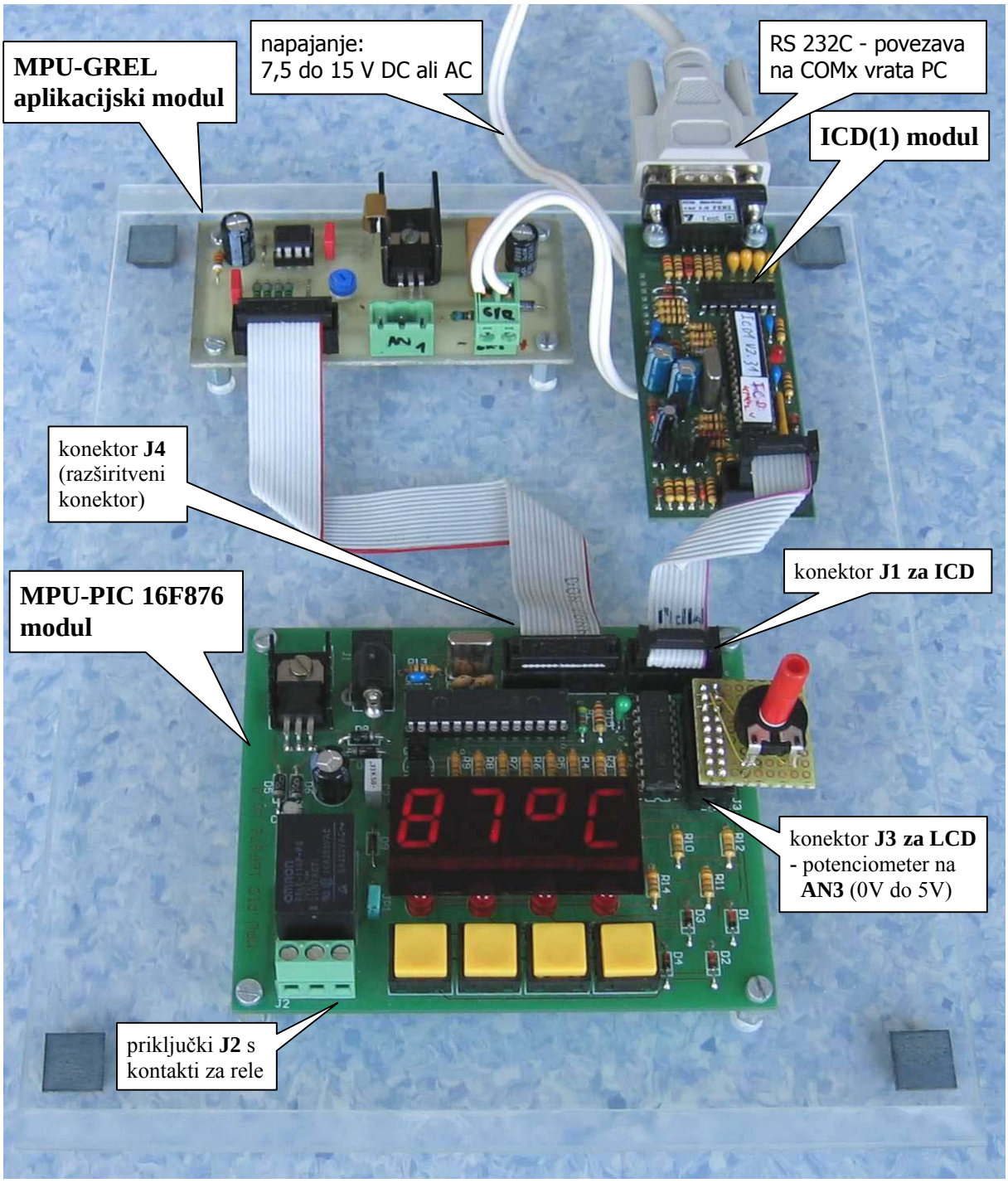
Za nevešče uporabnike programskega jezika C, je priporočljiv predhodni obisk tečaja iz osnov programiranja v ANSI C jeziku, ali da se naučijo osnov iz ustreznih priročnikov.

Zbrano gradivo je primerno tudi za tiste, ki uporabljajo druge tipe Microchip PIC mikrokrmilnikov srednje kategorije (s 14-bitno programsko besedo), npr.: 16F84, 16F877A in drugi.

V učnem gradivu je izpis popolnega (delujočega) programa v C jeziku. Več zgledov kot tudi opisov študentskih projektov, ki temeljijo na uporabi PIC mikrokrmilnikov, je na voljo uporabnikom na spletnih straneh predmetov Mikroelektronika in Mikroračunalniški sistemi v okviru spletišča Inštituta za robotiko (http://www.ro.feri.uni-mb.si/izobrazevanje/).

Pričujoče učno gradivo je bilo oblikovano »po merilih za e-učna gradiva« iz **Poslovnika založniške dejavnosti v e-izobraževanju na Univerzi v Mariboru** (aktivno kazalo, nadpovezave na poglavja, slike, tabele in HTML reference).

1. Predstavitev MPU-PIC16F876 učnega kompleta



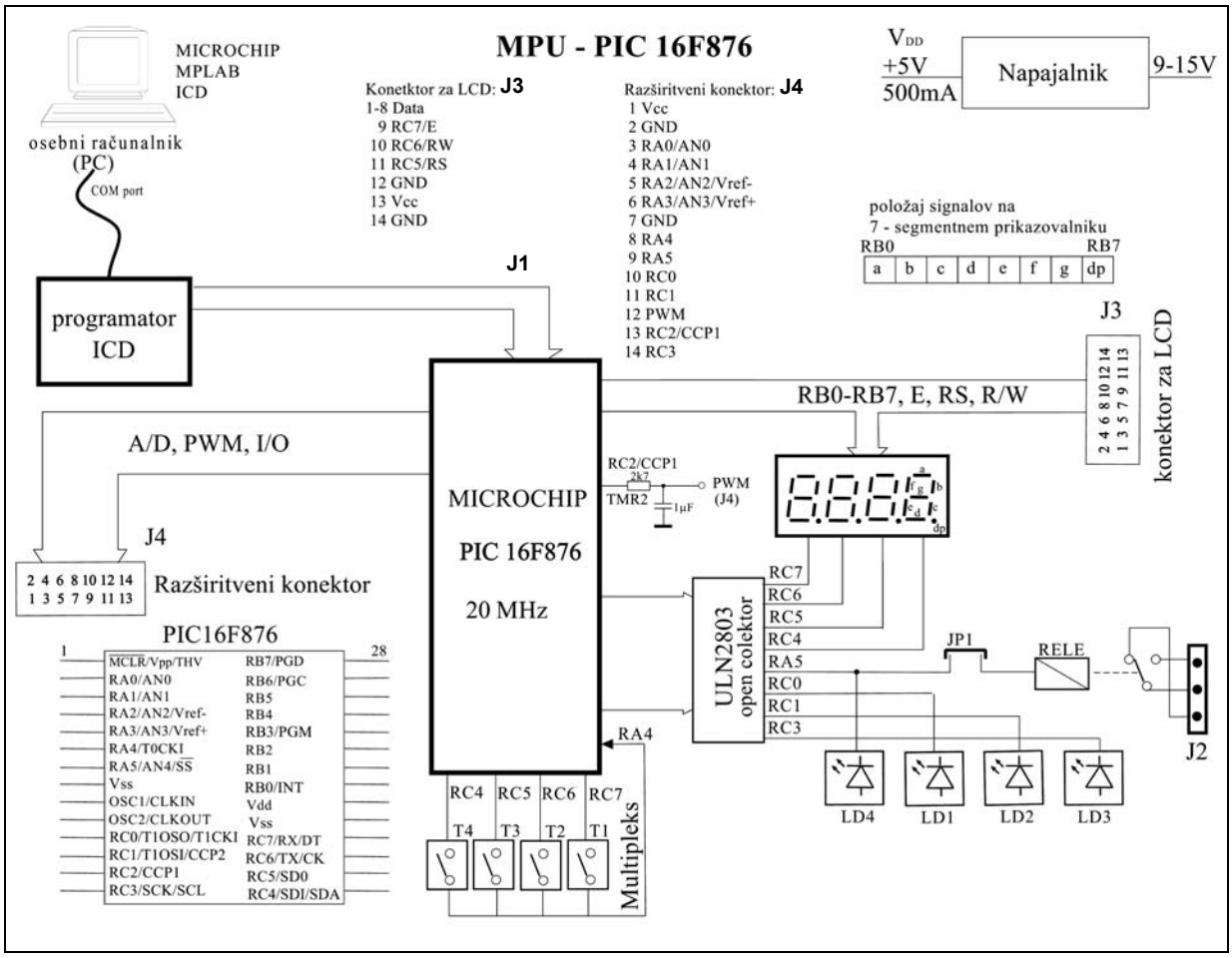
Slika 1-1: Fotografija MPU PIC 16F876 kompleta

Mikrokrmilniški učni komplet MPU-PIC 16F876 je sestavljen iz treh modulov (Slika 1-1):

- **MPU-PIC16F876** modul
- **ICD**(1) modul
- **MPU-GREL** aplikacijski modul

1.1. Modul MPU-PIC16F876

Zgradbo modula in povezavo vhodno/izhodnih naprav na priključke vrat prikazuje blokovna shema (Slika 1-2):



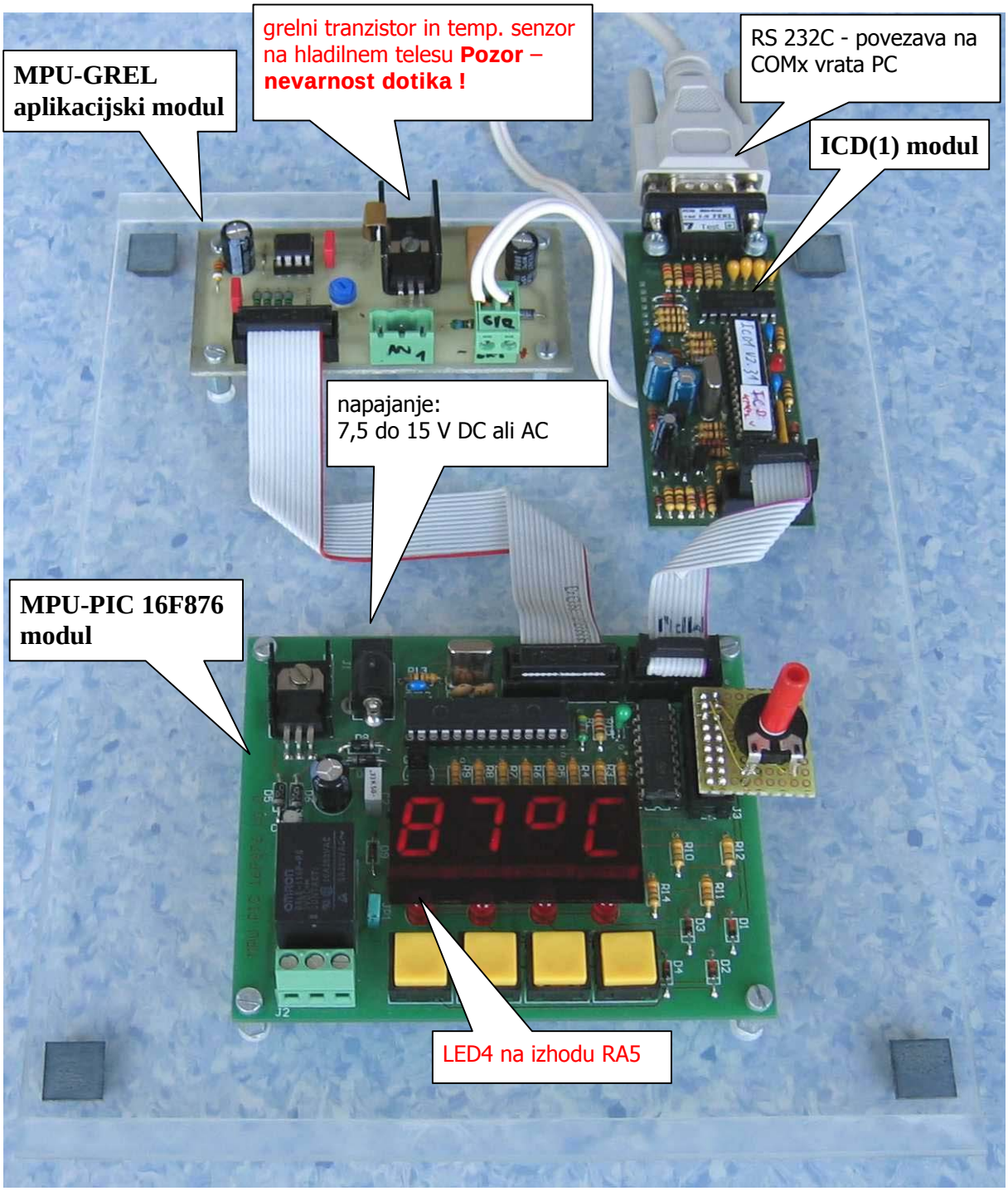
Slika 1-2: Blokovna shema MPU PIC 16F876 modula

Modul **MPU-PIC16F876** je bil razvit na UM-FERI, Inštitutu za robotiko za potrebe učenja osnov programiranja in uporabe PIC mikrokrmilnikov [4]. Zgrajen je na osnovi mikrokrmilnika **Microchip PIC16F876.** Na kartici so na voljo tudi:

- 4 enote LED indikatorjev;
- 4 enote - 7-segmentni LED prikazovalnik;
- 4 tipke;
- rele z delovnim in mirovnim kontaktom;
- 14-polni razširitveni konektor (priključki za analogne vhode, PWM izhod, časovniki, ...) (J4);
- 10-polni konektor za priključitev ICD(1) modula (J1);
- konektor za povezavo z osebnim računalnikom (RS232C oz. COMx vrata);
- 14-polni konektor za priključitev LCD prikazovalnika (J3).

1.2. Opozorilo o možnosti pregrevanja in nevarnosti dotika

OPOZORILO: *Izhod **RA5 (LD4)** je priključen na krmilni vhod grelnega tranzistorja. Če je **LED4 vključena več kot minuto,** lahko temperatura na hladilnem telesu naraste preko 60 °C. V takšnem primeru je najbolje **prekiniti program** (Reconnect na ICD) ali **izklopiti napajanje**.*



Slika 1-3: Možnost pregrevanja MPU PIC 16F876 kompleta

1.2.1. Kratek opis ukazov mikrokrmilnika PIC 16F87x

ukaz, operand	opis ukaza	STATUS	cikli
UKAZI, KI SO POVEZANI Z ZLOGI (BYTE)			
ADDWF f, d	seštej **W** in register **f**	C, DC, Z	1
ANDWF f, d	»logična in« operacija med **W** in registrom **f**	Z	1
CLRF f	register **f** naj dobi vrednost 0	Z	1
CLRW -	register **W** naj dobi vrednost 0	Z	1
COMF f, d	eniški komplement registra **f** (zamenjava 0 in 1)	Z	1
DECF f, d	zmanjšaj za ena vsebino registra **f**	Z	1
DECFSZ f, d	zmanjšaj vsebino reg. **f** za ena in preskoči naslednji ukaz, če **f**=0	-	1(2)
INCF f, d	povečaj za ena vsebino registra **f**	Z	1
INCFSZ f, d	povečaj vsebino reg. **f** za ena in preskoči naslednji ukaz, če **f**=0	-	1(2)
IORWF f, d	»logična ali« operacija med **W** in registrom **f**	Z	1
MOVF f, d	kopiraj vsebino registra f (d=0 ->**W**; d=1 ->**f**)	Z	1
MOVWF f	kopiraj vsebino **W** v register **f**	-	1
NOP -	ne naredi ničesar	-	1
RLF f, d	krožno premakni vse bite v registru **f** za eno mesto v levo	C	1
RRF f, d	krožno premakni vse bite v registru **f** za eno mesto v desno	C	1
SUBWF f, d	odštej vsebino reg. **W** od vsebine v reg. **f** (d=**f**-**W**)	C, DC, Z	1
SWAPF f, d	zamenjaj zgornje in spodnje 4 bite v registru **f**	-	1
XORWF f, d	»izključno ali« logična operacija med **W** in reg. **f**	Z	1
UKAZI, KI SO POVEZANI Z BITI			
BCF f, b	briši bit b v registru **f**	-	1
BSF f, b	postavi bit b v registru **f**	-	1
BTFSC f, b	testiraj bit b v registru **f** in preskoči naslednji ukaz, če je bit b=0	-	1(2)
BTFSS f, b	testiraj bit b v registru **f** in preskoči naslednji ukaz, če je bit b=1	-	1(2)
SISTEMSKI IN UKAZI, KI SO POVEZANI S KONSTANTAMI			
ADDLW k	registru **W** prištej konstanto **k**	C, DC, Z	1
ANDLW k	»logična in« operacija med **W** in konstanto **k**	Z	1
CALL k	klic podprograma	-	2
CLRWDT -	vsebina »časovnega stražnika« (**W**atch **D**og **T**imer) naj bo enaka nič	TO, PD	1
GOTO k	skoči na naslov k (11-bitni naslov)	-	2
IORLW k	»logična ali« operacija med **W** in konstanto **k**	Z	1
MOVLW k	vpiši konstantno vrednost **k** v **W** (**W**=**k**)	-	1
RETFIE -	vrni se iz podprograma za strežbo prekinitvene zahteve	-	2
RETLW k	vrni se iz podprograma s konstantno vrednostjo **k** v **W**	-	2
RETURN -	vrni se iz podprograma	-	2
SLEEP -	postavi mikrokrmilnik v stanje mirovanja - »sleep« način	TO, PD	1
SUBLW k	odštej vsebino **W** od konstantne vrednosti k (**W**=k-**W**)	C, DC, Z	1
XORLW k	»izključni ali« logična operacija med **W** in konstanto **k**	Z	1

Tabela 1-1: Tabela ukazov s kratkimi opisi

f - ime ali naslov datotečnega registra (**f**ile register)

k - konstantna (8-bitna) vrednost med 0 in 255

d - cilj hrambe rezultata operacije (d=0 - shrani se v **W**, d=1 – shrani se v **f**)

b – pozicija ali oznaka bita med 0 in 7

1(2) – trajanje ukaza je 1 (ali 2) ukazna cikla (pri 20 MHz taktu je trajanje enega ukaznega cikla (4*50) ns = 0,2 µs)

1.2.2. Podroben prikaz segmentne organizacije podatkovnega pomnilnika

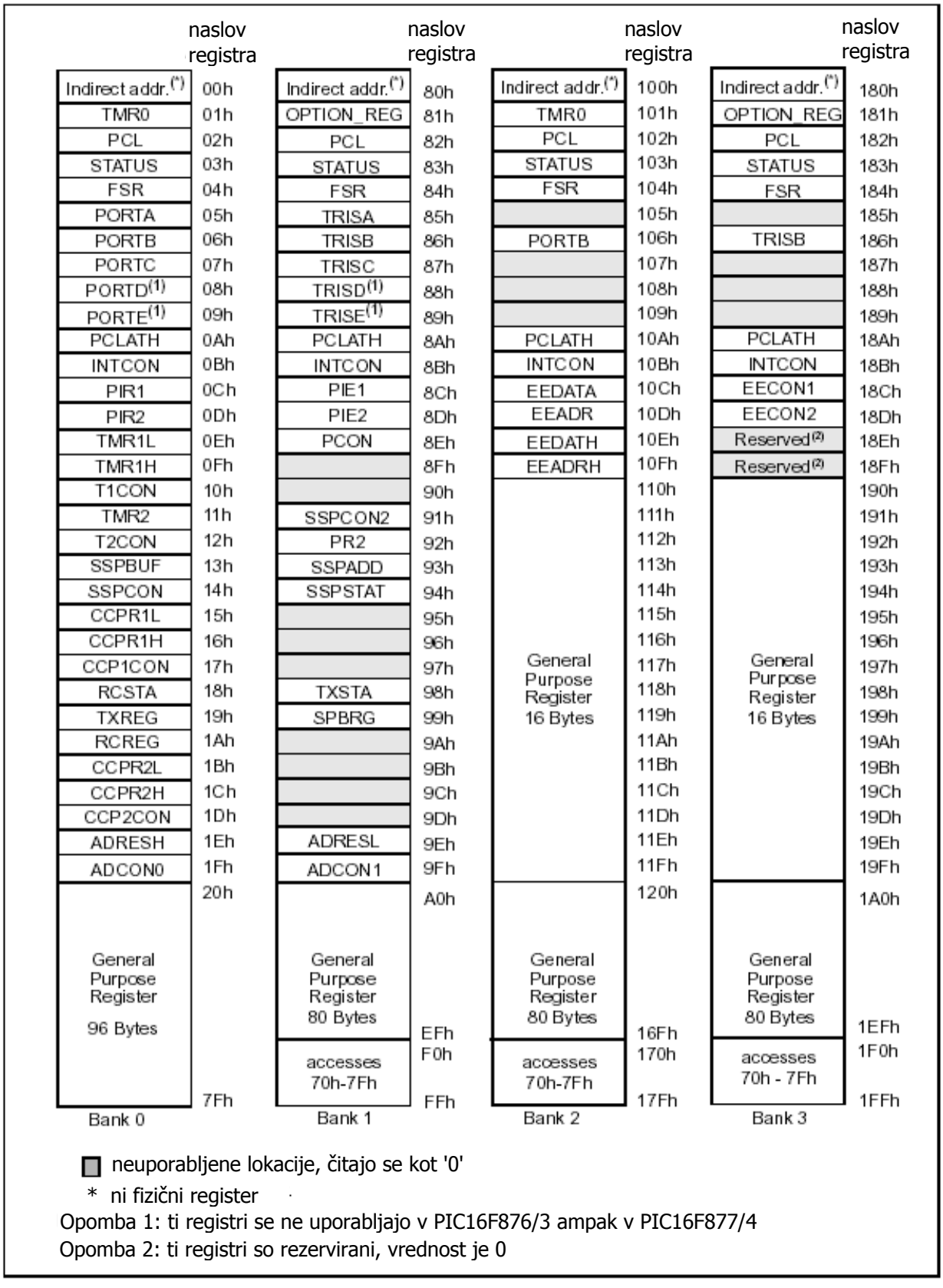


Tabela 1-2: Razporeditev registrov po pomnilniških segmentih

2. Uporaba programskega orodja Microchip MPLAB

Spoznali boste programska orodja integriranega programskega okolja Microchip MPLAB. Naučili se boste kreirati projekt, konfigurirati ICD ali Simulator ter sprožiti postopek generiranja izvršljivega programa za PIC mikrokrmilnik.

Opomba: Opis se nanaša na programsko orodje **Microchip MPLAB V5.70** (zadnje različice, ki še omogoča delo v kombinaciji z **ICD(1)**. V kolikor ima uporabnik možnost dela z **ICD2** (ali naprednejšim razvojnim sistemom), je priporočljiva uporaba novejše različice **MPLAB** iz serije **V6.X**.

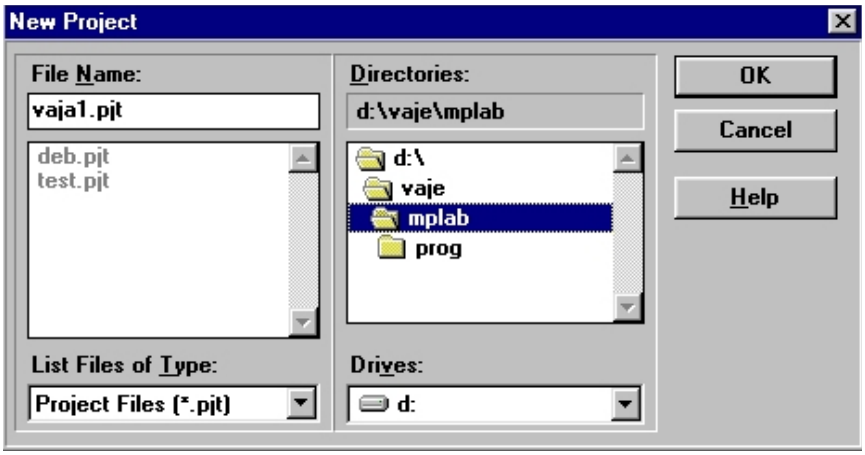
V nadaljevanju bo podrobno opisan postopek od kreiranja projekta do stopnje, ko lahko program izvajamo na ciljnem mikrokrmilniku (npr.: MPU-PIC16F876) ali kar na osebnem računalniku z orodjem MPLAB Simulator.

Modul **MPU-PIC16F876** povežite z modulom **ICD**. Pazite, da bo kabel ustrezno priključen (glejte napise!). Nato modul **ICD** povežite z računalnikom preko serijskega vodila (COM1 ali COM2, lahko tudi COM4). Nazadnje priključite še napajanje (AC ali DC, 7,5 – 15 V).

Poženite program [10][9][16] ***Microchip MPLAB*** (***Start*** → ***Programs*** → ***Microchip MPLAB*** → ***MPLAB***) in prikaže se osnovno okno **MPLAB** programirnega okolja.

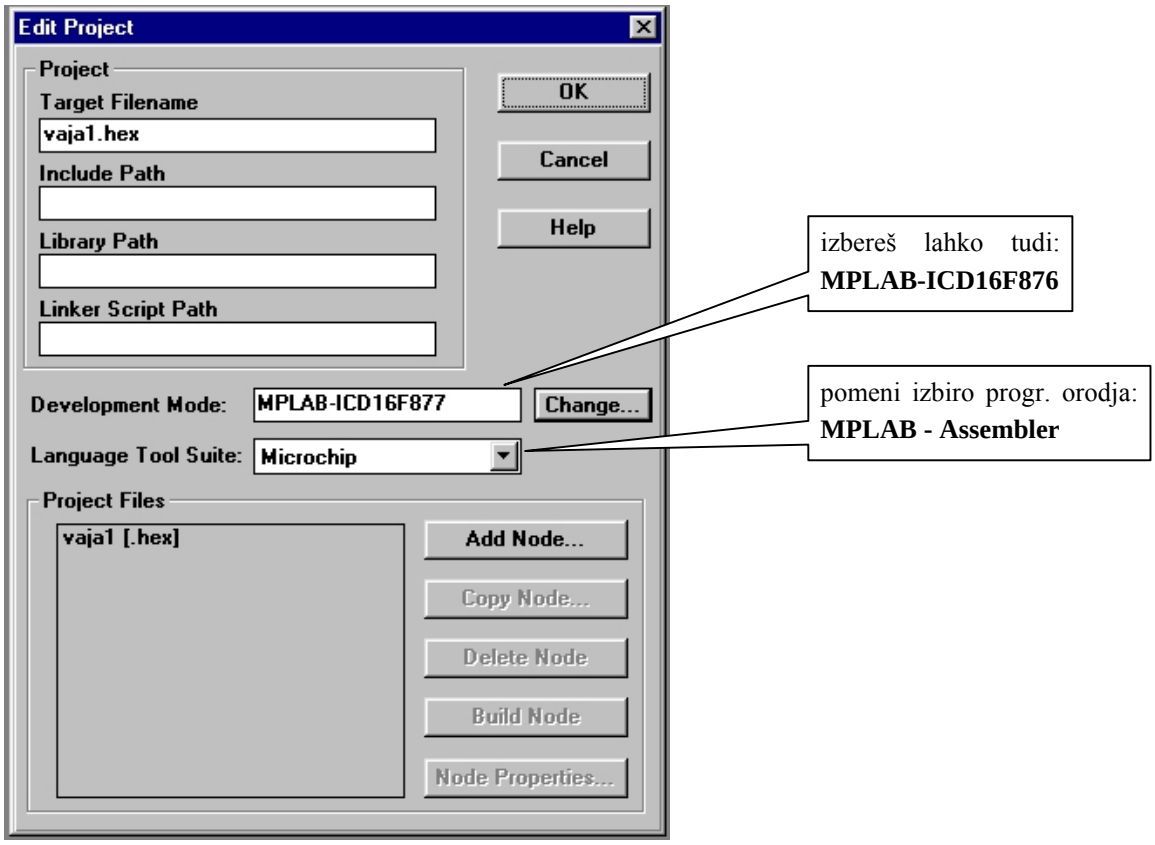
2.1. Konfiguriranje MPLAB programskega okolja

Odprite (Slika 2-1) novi projekt (***Project*** → ***New Project*** ...); projekt imenujte poljubno; obvezna je končnica **.pjt,** pot (direktorij) odprete v svojem delovnem imeniku:



Slika 2-1: Okno za »nov projekt«

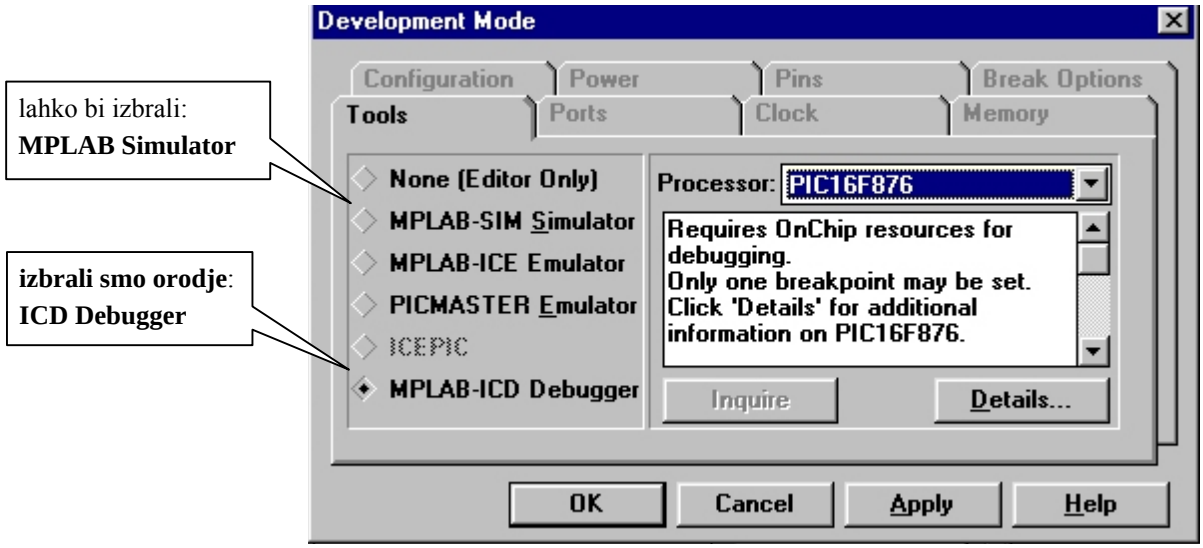
Pritisnite OK, odpre se (Slika 2-2) okno ***Edit Project***:



Slika 2-2: Okno »urejanje lastnosti projekta«

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

Kliknite ***Change*** ..., odpre se (Slika 2-3) okno ***Development Mode***, kjer nastavite opcije, kot so prikazane na spodnji sliki:



Slika 2-3: Okno za izbiro razvojnega okolja

Kliknete **OK** in prikaže se (Slika 2-4) okno ***MPLAB-ICD***:

Slika 2-4: Okno za izbiro COM vrat ICD modula

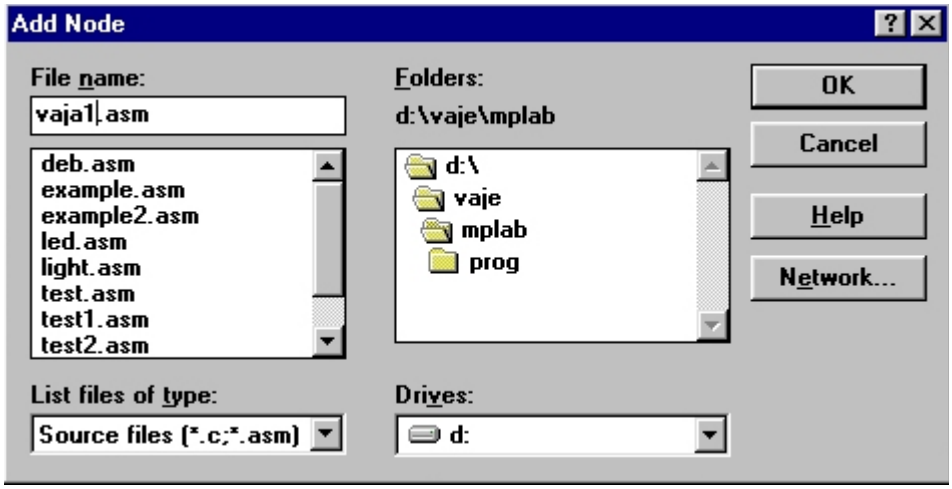
To okno naj bo ves čas prisotno na zaslonu, prestavimo ga lahko npr.: desno zgoraj.

Po potrebi izberemo ustrezna COMx vrata. Če komunikacija med **ICD** modulom in **MPLAB** okoljem ni vzpostavljena, utripa LED indikator na **ICD**. Komunikacijo poskušamo vzpostaviti s klikom na gumb ***Reconnect***. Če LED indikator še vedno utripa, je potrebno izbrati druga COMx vrata.

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

2.2. Nalaganje programa v ciljni sistem in testiranje

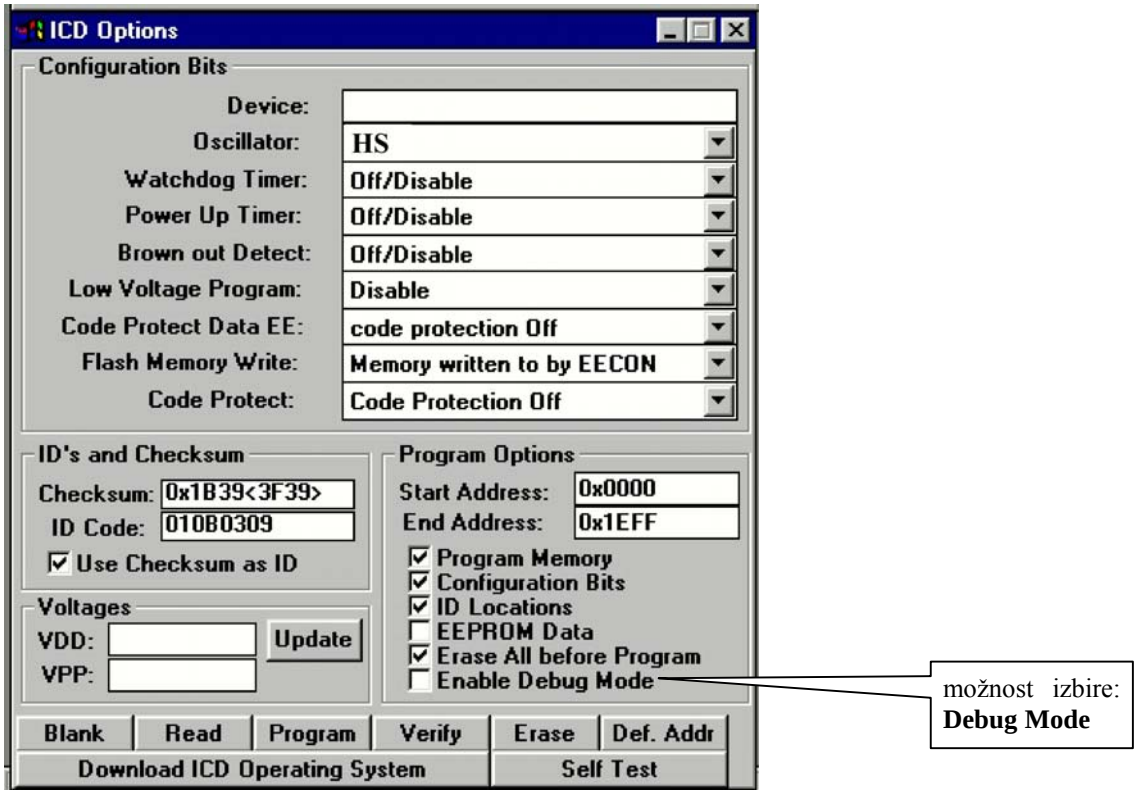
V oknu ***Edit Project*** kliknete še ***Add Node*** in vključite datoteko (Slika 2-5), v kateri se nahaja program s končnico **.asm** (izvorni program v zbirnem jeziku, npr.: vaja1.asm, le-to lahko kasneje urejate v vključenem urejevalniku **MPLAB Editor**):



Slika 2-5: Okno za izbiro/vpis imena izvorne datoteke

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

V oknu ***MPLAB-ICD*** kliknete ***Options*** ... in v oknu ***ICD Options*** nastavite opcije (v desni polovici), kot to kaže (Slika 2-6).



Slika 2-6: Okno za nastavitev možnosti delovanja ICD

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

Zatem odprete novo datoteko (***File*** → ***New***), **napišete (ali modificirate) izvorni program v zbirnem jeziku** (Assembler) ter ga shranite s ***File*** → ***Save as*** ... (v našem primeru **vaja1.asm**).

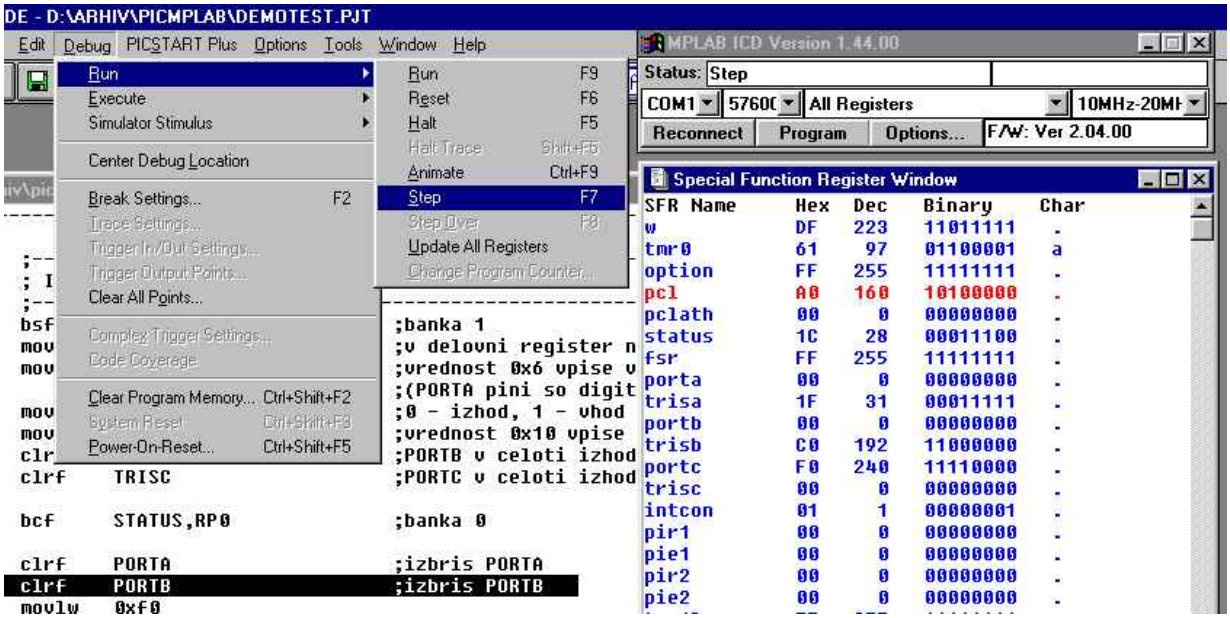
Program bo tako napisan v zbirnem jeziku (Assembler) in vsebuje obvezno končnico ***.asm**.

1. **Prevajanje (**aktiviranje **zbirnika)** izvedete tako, da v osnovnem oknu **MPLAB** menuju ***Project*** izberete opcijo ***Build All.***
2. **Program naložite** tako, da v oknu ***MPLAB-ICD*** kliknete ***Reconnect*** in nato ***Program***.
3. **Program poženete** (sprostite RESET) tako, da **še enkrat** sprožite prevajanje (***Project*** → ***Build All***).
4. **Delovanje programa prekinete** tako, da v oknu ***MPLAB-ICD*** kliknete ***Reconnect.***

Opomba: Če izberete možnost ***Enable Debug Mode*** (v oknu ***ICD Options***), **ne izvajate točk 3 in 4**, kajti MPLAB omogoča kontrolirano izvajanje programa preko ikon ali menijev v **MPLAB** okolju (***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**).



Slika 2-7: Okno za »razhroščanje« (kontrolirano izvajanje programa

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 **MPU-PIC16F876** modula se naključno spreminjajo, vsebina prikazanih registrov ne ustreza vedno dejanskemu stanju, **program ne teče v realnem času**!).

Za razumevanje tematike je potrebno tudi predznanje iz osnov programiranja v zbirnem jeziku, kar je na voljo med drugim v [2][4][6][9]. **Pomembna lastnost** Microchip MPLAB programskega okolja je, da lahko **programe izvajamo** (z določenimi omejitvami) tudi v **simulacijskem načinu** (brez uporabe ciljnega sistema) zgolj na osebnem računalniku povprečnih zmogljivosti [10].

Vprašanja za utrjevanje:

1. *Kakšno končnico ima datoteka - izvorni program v zbirnem jeziku?*
2. *Kakšno končnico ima datoteka - strojni program v zbirnem jeziku?*
3. *S katerim orodjem pretvarjamo izvorne programe v zbirnem jeziku v strojne programe ?*
4. *Kam se naloži program pri uporabi MPLAB-ICD modula?*
5. *Kakšne možnosti pri testiranju programa imamo, če izberemo «ICD Debug Mode«?*
6. *Kakšne omejitve pri testiranju programa imamo, če izberemo «ICD Debug Mode«?*

3. Uporaba programskega jezika C v MPLAB okolju

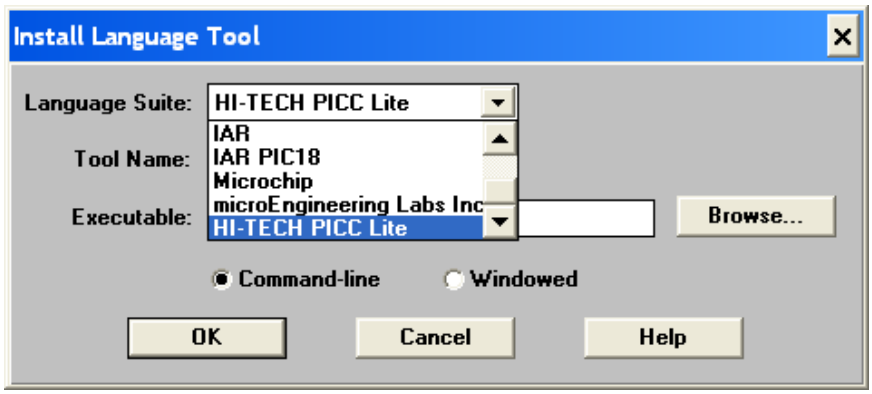
Spoznali boste možnost uporabe C-prevajalnika v programskem okolju Microchip MPLAB. Naučili se boste kreirati projekt, konfigurirati ICD ali Simulator ter sprožiti postopek generiranja izvršljivega programa za PIC mikrokrmilnik.

Na voljo je **več različic C prevajalnikov** različnih proizvajalcev. Za nekomercialno uporabo je zelo primeren **HI-TEC C** [11] prevajalnik **PICC Lite** (v nadaljevanju bo opisana različica: [13][14] *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 ter izpisi števil v *float* in *long* formatu niso možni.

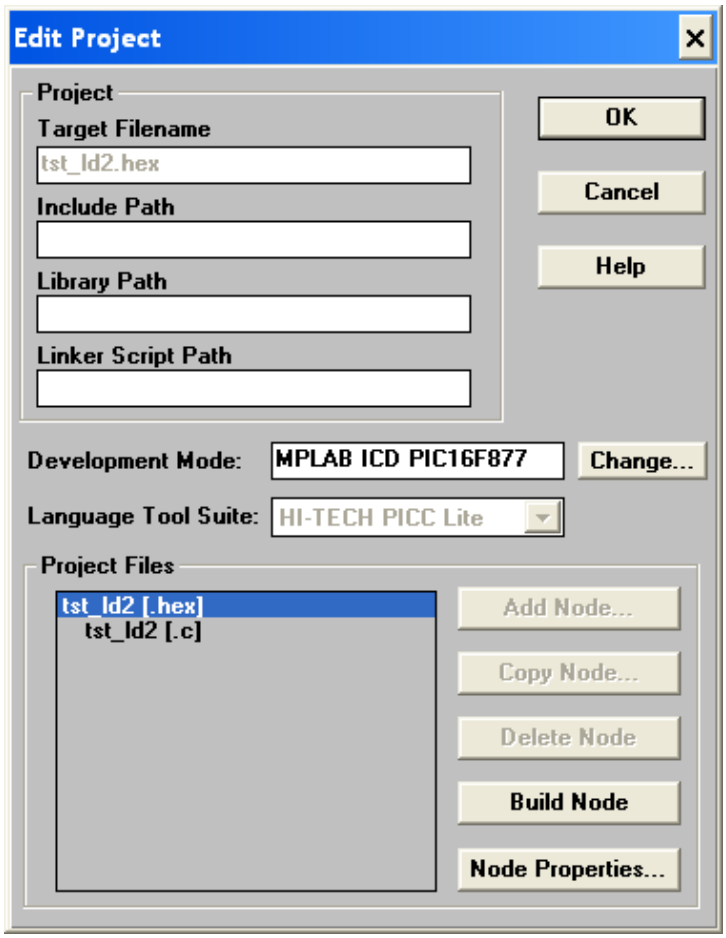
3.1. Konfiguriranje C-prevajalnika

Program se po instalaciji naloži (privzeto) na disk C: v imenik **C:\PICCLITE**. Samodejno se integrira v **IDE okolje MPLAB**. To preverimo (Slika 3-1) v oknu: (***Project → Install Language Tool***).

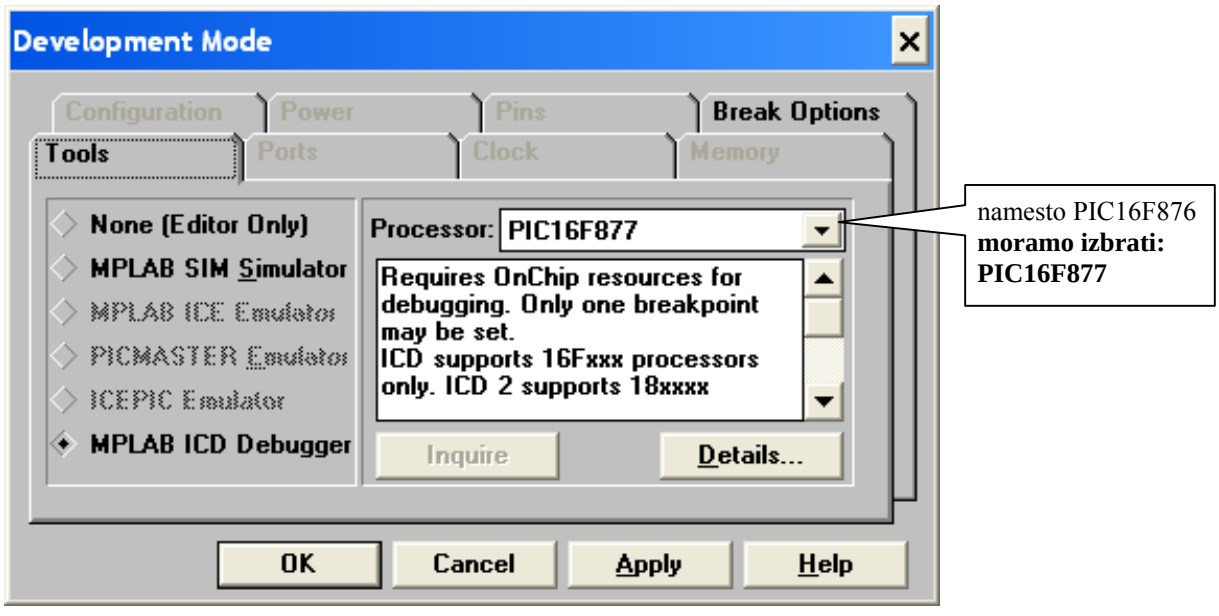


Slika 3-1: Okno za vključitev HI-TECH PICC Lite prevajalnika

V meniju (***Project → New...*** *ali* ***Edit Project...***) odpremo (Slika 3-2) ali modificiramo projekt ***.pjt**, pri čemer **najprej** izberemo **Language Tool Suite: HI-TECH PICC Lite**. Zatem **obvezno izberemo PIC16F877.** V okviru razvojnega okolja (Slika 3-3), lahko izbiramo med **ICD** razvojnim okoljem in med **MPLAB SIM simulatorjem**).

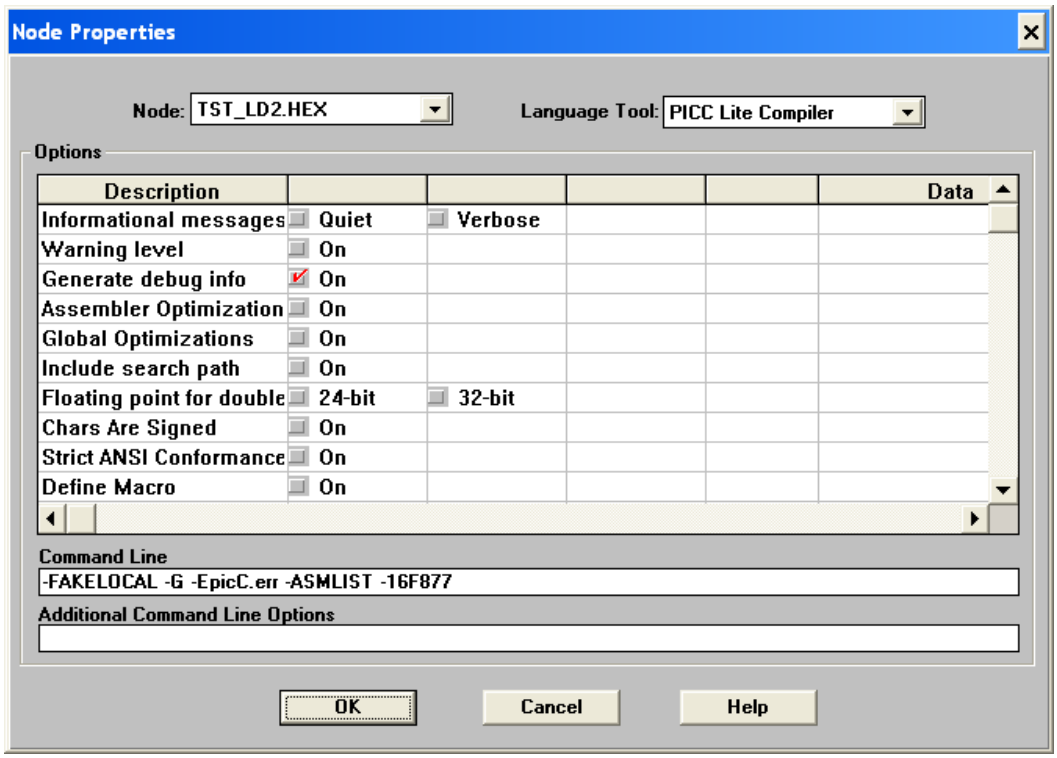


Slika 3-2: Okno za pripravo projekta



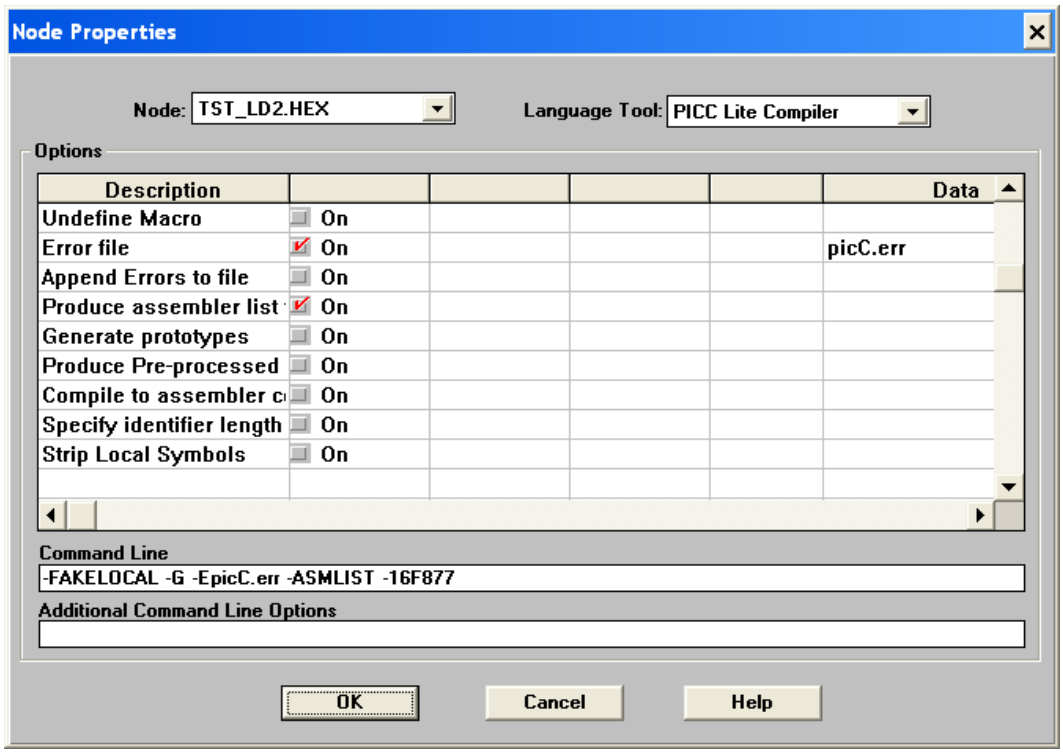
Slika 3-3: Okno za izbiro ICD orodja in čip PIC16F877

Označimo datoteko s končnico [.hex] in odpremo okno (Slika 3-4) **Node Properties**.



Slika 3-4: Okno za nastavitev opcij C-prevajalnika (1. del)

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).



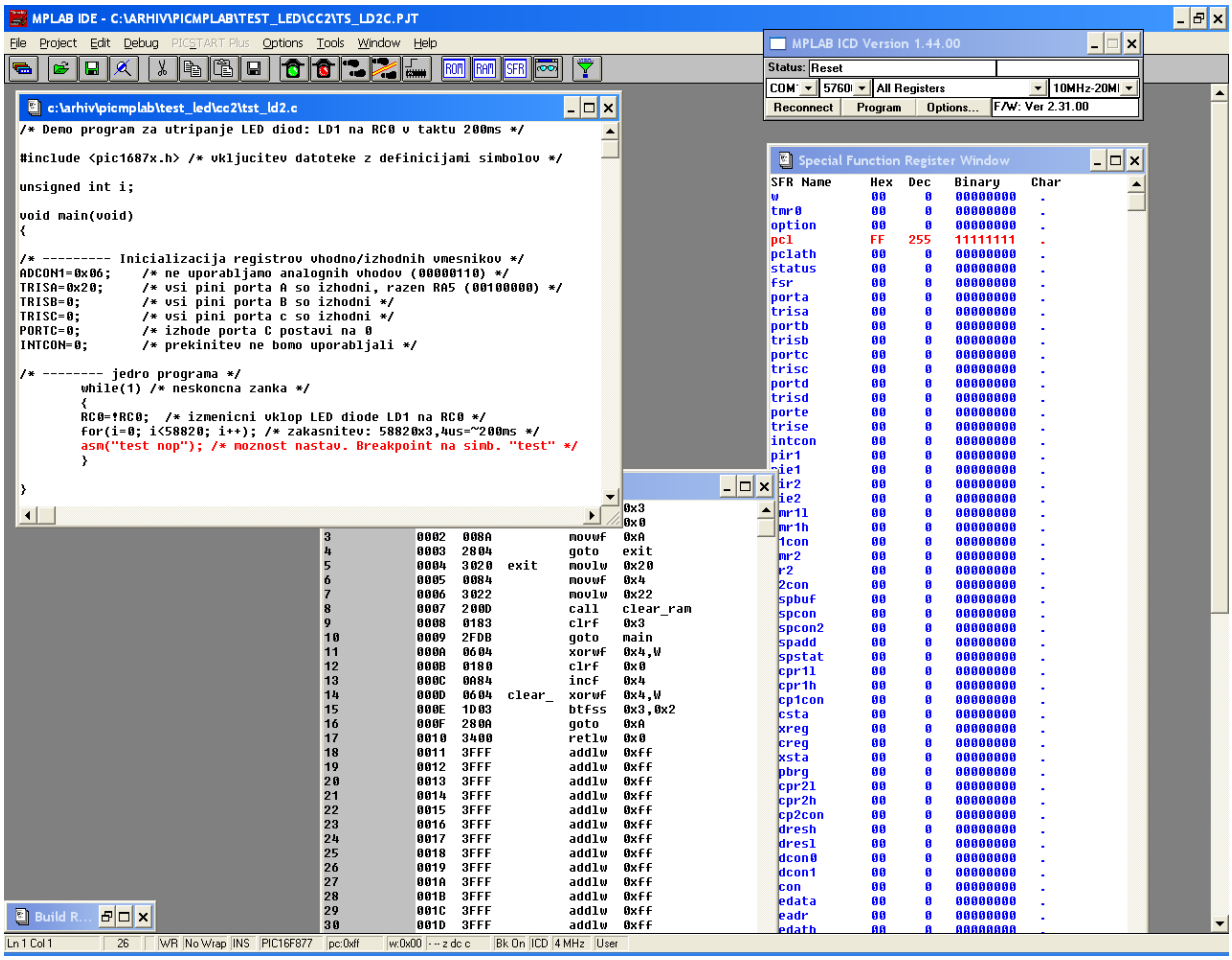
Slika 3-5: Okno za nastavitev opcij C-prevajalnika (2. del)

Obvezno izberemo rubriko **Error File** (Slika 3-5) 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 izvršljivo kodo: *tst_ld2.hex*.

3.2. Prevajanje in testiranje programa

Primer izpisa (Slika 3-6) izvornega programa ***.c** (levo), okno vsebine programskega pomnilnika s programom v strojni kodi (inverzni zbirnik) in okno s prikazom vsebine **SFR** registrov (desno).

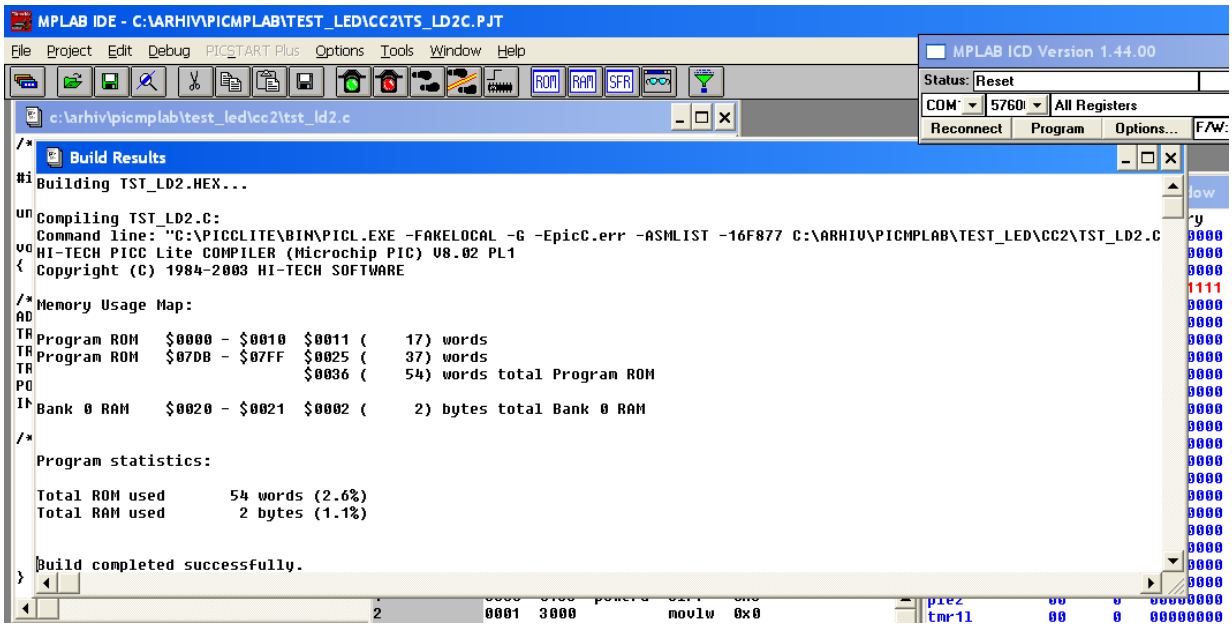


Slika 3-6: Okno, kjer testno izvajamo program

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

V kolikor ni sintaktičnih napak, se izpiše sporočilo o lastnostih prevedenega programa (Slika 3-7) 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).

Če 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.



Slika 3-7: Okno z izpisom statusa C-prevajalnika

Za razumevanje tematike je potrebno tudi predznanje iz osnov programiranja v ANSI C-jeziku, kar je na voljo med drugim v [11][5][13][18]. **Pomembna lastnost** Microchip MPLAB programskega okolja je, da lahko **programe izvajamo** (z določenimi omejitvami) tudi v **simulacijskem načinu** (brez uporabe ciljnega sistema) zgolj na osebnem računalniku povprečnih zmogljivosti [10].

Vprašanja za utrjevanje:

1. *Kakšno končnico ima datoteka - izvorni program v C-jeziku?*
2. *Kakšno končnico ima datoteka - strojni program (preveden iz C-jezika)?*
3. *S katerim orodjem pretvarjamo izvorne programe iz C-jezika v strojne programe ?*
4. *Kam se naloži program pri uporabi MPLAB-ICD modula?*
5. *Kaj predstavlja (pri uporabi HI-TEC PICC Lite) izbira opcije »Generate Debug Info«?*
6. *Kateri čip moramo izbrati (pri uporabi HI-TEC PICC Lite), če imamo v ciljnem sistemu mikrokrmilnik »16F876«?*

4. Vprašanja za utrjevanje in učni vodnik

Vsebina poglavja predstavlja dopolnitev »zbranega gradiva za predavanje« z naslovom **Uvod v programiranje mikrokrmilnikov** in sicer kot učni vodnik z vprašanji in odgovori za utrjevanje, razdeljenimi po poglavjih ter napotki za potrebna znanja s sugestijo za iskanje v dodatnih virih.

4.1. Predstavitev MPU-PIC16F876 učnega kompleta

Spoznali boste učni komplet MPU-PIC16F876 in osnovne lastnosti mikrokrmilnika PIC 16F876. Naučili se boste identificiranja vhodnih oziroma izhodnih priključkov mikrokrmilnika.

Za popolno razumevanje tematike je potrebno tudi predznanje iz osnov digitalne elektronike in gradnikov mikroprocesorskih sistemov, kar je na voljo med drugim v [3]. Dobri opisi zgradbe in uporabe sistemov s PIC mikrokrmilniki so tudi v [2] in [6].

Vprašanja za utrjevanje:

1. *Kateri moduli tvorijo MPU-PIC16F876 komplet ?*
2. *Kakšen tip procesne enote je vsebovan v mikrokrmilniku PIC16F876 ?*
3. *Kakšni pomnilniški elementi so vsebovani v mikrokrmilniku PIC16F876 ?*
4. *Koliko vhodno/izhodnih priključkov ima mikrokrmilnik PIC16F876 ?*
5. *Katere funkcije opravlja ICD modul ?*
6. *Katere vhodne in izhodne signale vsebuje modul MPU-GREL ?*

4.2. Programiranje PIC16F87x v zbirnem jeziku

Naučili se boste oblikovanja programa v zbirnem jeziku (splošni format) ter osnovne programske ukaze in njihov pomen. Ko boste obvladali nekatere podrobnosti, boste v nadaljevanju lahko uporabljali zgolj zgoščeno tabelo nabora vseh 35 ukazov.

Za razumevanje tematike je potrebno tudi predznanje iz osnov digitalne elektronike (številski sistemi, logične operacije) [3] in osnov programiranja v zbirnem jeziku: [2] in [8]. Dobri opisi programskih ukazov PIC mikrokrmilnikov srednje kategorije so na voljo v [2] in [6] ter na spletnem portalu: http://www.interq.or.jp/japan/se-inoue/e_pic.htm.

Vprašanja za utrjevanje:

S katerim ukazom postavimo (na 1) bite 0, 1 in 2 v delovnem registru W ?

Rešitev:

```
IORLW  b'00000111' ;'ALI' log. oper. med istoleznimi biti, visjih 5 bitov v W se ohrani,
                   ;spodnji trije biti v W se postavijo na 1
```

1. *Koliko različnih ukazov je na voljo ?*
2. *Kakšna polja vsebuje programska vrstica v zbirnem jeziku ?*
3. *S katerim ukazom zbrišemo bit 3 v registru f na naslovu 0x2F ?*
4. *S katerim ukazom opravimo logični IN (AND) med registrom W in f na naslovu 0x22 ?*
5. *S katerim ukazom zaključimo podprogram ?*
6. *S katerim ukazom zbrišem bite 5, 6, in 7 v registru W ?*

4.3. Organizacija pomnilnika

*Spoznali boste model in organizacijo podatkovnega pomnilnika (segmenti) in model programskega pomnilnika ter v zvezi s tem najpomembnejši datotečni register **STATUS**. Naučili se boste ločevati med podatkovnim pomnilnikom (RAM) in programskim pomnilnikom (FLASH ROM).*

Za razumevanje tematike je potrebno tudi predznanje iz osnov digitalne elektronike in gradnikov mikroprocesorskih sistemov (pomnilniški elementi), kar je na voljo med drugim v [3]. Dobri opisi zgradbe in opisa pomnilniškega modela PIC mikrokrmilnikov so tudi v [2] in [6] ter na spletnem portalu: http://www.interq.or.jp/japan/se-inoue/e_pic.htm.

Vprašanja za utrjevanje:

Napišite zaporedje ukazov za izbiro segmenta Bank1

Rešitev:

```
BCF     STATUS,RP1     ; 0 → RP1 (brisi bit 6 v reg. STATUS)
BSF     STATUS,RP0     ; 1 → RP0 (postavi bit 5 v reg. STATUS)
```

1. *Napišite zaporedje ukazov za izbiro segmenta Bank2*
2. *Kakšen pomen imajo zastavice v registru STATUS ?*
3. *Kateri pomnilniški segment (Bank0, 1, 2, 3) se največ uporablja ?*
4. *Koliko bitne so celice v programskem pomnilniku ?*
5. *Koliko nivojev ima sklad (stack) ?*
6. *S katerega naslova v programskem pomnilniku se začne izvajati program ?*
7. *Kateri datotečni registri se pojavijo v vseh štirih segmentih ?*

4.4. Osnovne vhodno/izhodne enote

Spoznali boste osnovne vhodno/izhodne enote – digitalne ali logične vhode/izhode. Naučili se boste konfigurirati posamezne priključke vrat A, B in C ter programsko krmiliti izhode in ugotavljati stanja vhodov.

Za razumevanje tematike je potrebno tudi predznanje iz osnov programiranja. Dodatna znanja iz programiranja v zbirnem jeziku so v: [2][4][6][7]. Za razumevanje programov, ki so kodirani v C-jeziku, je potrebno poznavanje osnov ANSI C strukturnega programiranja: [15][11][13][5]. Veliko rešitev praktičnih nalog z uporabo PIC mikrokrmilnikov je na spletnem portalu http://www.interq.or.jp/japan/se-inoue/e_pic6.htm.

Vprašanja za utrjevanje:

Napišite podprogram za inicializacijo vseh linij vrat PORTB kot izhodi.

Rešitev:

```
PBizh   bsf     STATUS,RP0      ; 1 -> RP0 , izbira Bank1 za dostop do TRISB
        clrf    TRISB           ; 0 -> TRISB, vsi izhodi
        bcf     STATUS,RP0      ; 0 -> RP0 , izbira Bank0 za dostop do PORTB
        return                  ; konec podprograma
```

1. *Napišite podprogram za inicializacijo vseh linij vrat PORTC kot vhodi.*
2. *Kakšen pomen imajo linije vrat PORTA ?*
3. *Napišite programček za ugotavljanje stanj na vhodih: RA0, RA1 in RA3 ?*
4. *Stanje stikala na vhodu RA4 prenesi (programsko) na izhod RC3.*
5. *Izhode na RC0 in RC1 preklapljaj izmenoma vsakih (pribl.) 100 ms.*

4.5. Analogni vhodi

Spoznali boste vmesnik mikrokrmilnika, ki omogoča analogne vhode mikrokrmilnika. Naučili se boste konfigurirati posamezne priključke vrat PORTA za izbiro analognih vhodov. Naučili se boste uporabljati 10 ali 8-bitni rezultat in ločevati med desno oz. levo poravnavo ter interpretirati številčni rezultat AD pretvorbe.

Za razumevanje tematike je potrebno tudi predznanje iz osnov programiranja in številskih sistemov. Dodatna znanja iz programiranja v zbirnem jeziku so v: [2][4][6][7]. Za razumevanje programov, ki so kodirani v C-jeziku, je potrebno poznavanje osnov ANSI C strukturnega programiranja: [15][11][13][5]. Veliko rešitev praktičnih nalog z uporabo PIC mikrokrmilnikov, je na spletnem portalu http://www.interq.or.jp/japan/se-inoue/e_pic6.htm. Podrobnejši opis delovanja A/D pretvornika s primeri uporabe v PIC mikrokrmilnikih najdemo tudi v: [8][16][17][18].

Vprašanja za utrjevanje:

Napišite podprogram z imenom InAD1 za inicializacijo AD pretvornika (vhod AN1) z desno poravnavo rezultata.

Rešitev:

```
;--------- Inicializacija registrov vhodno/izhodnih vmesnikov
InAD1   bsf     STATUS,RP0      ; 1 -> RP0, Bank1 za dostop do TRISA in ADCON1
        bsf     TRISA,1         ; 1 -> TRISA.1, RA1/AN1- vhod
;--- inicializacija A/D – AN1
        movlw   b'10000100'     ;priprava vredn. za ADCON1, vhod AN1-analogni, tudi AN3 in AN0
        movwf   ADCON1          ;bit7=1 -> desna poravnava 10-bitnega rezultata
        bcf     STATUS,RP0      ;0 ->RP0, nazaj v banko 0
        movlw   b'10001001'     ;takt (f/32,Fosc=20MHz), izbira anal. kan. AN1, AD  omogocen
        movwf   ADCON0          ;konec inicializacije
        return                  ;konec podprograma
```

1. *Napišite podprogram z imenom InAD3 za inicializacijo AD pretvornika (vhod AN3) z levo poravnavo rezultata.*
2. *Napišite podprogram (ki se navezuje na točko 1) za Start AD pretvorbe vhoda AN3 in shranjevanje 8-bitnega rezultata v register W.*
3. *Kolikšna napetost je na analognem vhodu, če smo pri 8-bitni pretvorbi dobili rezultat 7Fh ?*
4. *Kolikšna napetost je na analognem vhodu, če smo pri 10-bitni pretvorbi dobili rezultat 300h in je V_{ref+} = 4,00 V ?*
5. *Kolikšen je čas AD pretvorbe (približno) in po kolikem času lahko sprožimo ponovno AD pretvorbo ?*

4.6. Časovnik Timer0 in periodično generiranje prekinitev

Spoznali boste osnovni 8-bitni časovnik. Naučili se ga boste konfigurirati in generirati natančne časovne intervale s pomočjo prekinitev.

Za razumevanje tematike je potrebno tudi predznanje iz osnov programiranja in številskih sistemov. Dodatna znanja iz programiranja v zbirnem jeziku so v: [2][4][6][7]. Za razumevanje programov, ki so kodirani v C-jeziku, je potrebno poznavanje osnov ANSI C strukturnega programiranja: [15][11][13][5]. Veliko rešitev praktičnih nalog z uporabo PIC mikrokrmilnikov je na spletnem portalu http://www.interq.or.jp/japan/se-inoue/e_pic6.htm. Podrobnejši opis delovanja časovnika Timer0 s primeri uporabe v PIC mikrokrmilnikih je na voljo v: [8][16][17][18].

Vprašanja za utrjevanje:

Napišite podprogram z imenom InTMR0 za inicializacijo časovnika Timer0, ki bo periodično prožil prekinitve na 0,2 ms.

Rešitev:

```
InTMR0 bsf    STATUS,RP0     ; 1 -> RP0 , izbira Bank1 za dostop do OPTION_REG
       movlw  b'00000001'    ;priprava vrednosti 0x01 za OPTION_REG
       movwf  OPTION_REG     ;preddelilnik casovnika Timer0 nastavimo na 4:1
       bcf    STATUS,RP0     ; 0 -> RP0, nazaj v Bank0
       movlw  b'10100000'    ;priprava vrednosti 0xA0 za INTCON, GIE=1, T0IE=1
       movwf  INTCON         ;Omogocimo prekinitve, ki jih sproza casovnik Timer0
       movlw  d'6'           ;priprava vrednosti (desetisko) 6 za TMR0
       movwf  TMR0           ;vrednost intervala: 0,2*(256-6)*4=200us=0,2ms
       return                ;konec podprograma
```

1. *Napišite podprogram z imenom InTMR0_2 za inicializacijo časovnika Timer0, ki bo periodično prožil prekinitve na 2,0 ms.*
2. *Koliko bitni je števec TMR0 in kakšne vrednosti preddelilnika lahko izbiramo ?*
3. *Kakšna je funkcija časovnega stražnika (Watch Dog Timer) ?*
4. *Ali lahko Timer0 uporabljamo, ne da bi prožil prekinitve ?*
5. *Kakšen interval bi dosegli v zgornjem rešenem zgledu, če ne bi vpisali nobene vrednosti v register TMR0 ?*

4.7. Generiranje pulzno-širinskih signalov (PWM)

Spoznali boste vmesnik mikrokrmilnika, ki omogoča generiranje pulznoširinskih signalov (PWM). Naučili se boste konfigurirati registre za izbiro periode (frekvence) in širine pulza vmesnikov PWM1 in PWM2.

Za razumevanje tematike je potrebno tudi predznanje iz osnov programiranja in številskih sistemov. Dodatna znanja iz programiranja v zbirnem jeziku so v: [2][4][6][7]. Za razumevanje programov, ki so kodirani v C-jeziku, je potrebno poznavanje osnov ANSI C strukturnega programiranja: [15][11][13][5]. Veliko rešitev praktičnih nalog z uporabo PIC mikrokrmilnikov je na spletnem portalu http://www.interq.or.jp/japan/se-inoue/e_pic6.htm. Podrobnejši opis delovanja časovnika Timer2 in PWM vmesnika s primeri uporabe v PIC mikrokrmilnikih je v: [16][17].

Vprašanja za utrjevanje:

Napišite podprogram z imenom InPWM1 za inicializacijo izhoda PWM1 s frekvenco 2,00 kHz.

Rešitev:

```
InPWM1 bsf    STATUS,RP0     ; 1 -> RP0 , izbira Bank1 za dostop do TRISC in PR2
       bcf    TRISC,2        ; RC2/CCP1 port za PWM1 definiramo kot izhod
       movlw  d'155'         ; desetisko 155 -> Za nastavitev frekvence signala 2kHz
       movwf  PR2            ; Perioda=(155+1)0.2us*16 = 499,2us -> f=1/499,2 = 2,003KHz
       bcf    STATUS,RP0     ; Nazaj v banko0
       clrf   CCPR1L         ; zacetna vrednost sirine pulza: CCPR1L = 0
       bsf    CCP1CON,3      ; 1 -> CCP1CON.3
       bsf    CCP1CON,2      ; 1 -> CCP1CON.2, bit3 in 2 na 1 v CCP1CON-PWM nacin delovanja
       movlw  b'00000111'    ; vrednost za T2CON, bit1,0 -> 1, preddelilnik je 1:4
       movwf  T2CON          ; 1 -> bit2, start casovnika TMR2
       return                ; konec podprograma
```

Napišite podprogram z imenom InPWM_2 za inicializacijo časovnika izhoda PWM2 s frekvenco 5,00 kHz.

1. *Koliko bitno vrednost lahko vpišemo kot vrednost za širino pulza ?*
2. *Kakšne vrednosti preddelilnika lahko izbiramo ?*
3. *Kakšna je ločljivost nastavitve širine pulza (najkrajša in najdaljša širina pulza v ns) pri frekvenci 20 kHz ?*
4. *Ali PWM vmesnik proži prekinitve ?*
5. *Kakšna je najmanjša širina pulza (razen 0) v zgornjem (rešenem) primeru ?*

5. Dodatek

5.1. Primer popolnega zgleda programa v C-jeziku za MPU-16F876

```
//-------------------------------------------------------------------------------------------------------
// Naziv programa: Regulacija temperature na MPU-PIC16F876
//
// Avtor(ji), verzija, datum: Darjan Leskovar, Sašo Jesenicnik, Janez Pogorelc, 3.6.2004
//
// Opis programa: Program dvotockovno regulira temperaturo grelnega telesa.
//-------------------------------------------------------------------------------------------------------
#include <pic1687x.h>
#include <stdlib.h>             //stdlib.h potrebujemo, da lahko vkljucimo funkcijo abs()
#include <stdio.h>

        const unsigned char stevila[13]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x63,0x39,0x00};
        const unsigned char linija[4]={0x80, 0x40, 0x20, 0x10};
        unsigned int histereza, stevec, test1, test2, t1, t2, w, x, y, z;
        unsigned int temperatura, zeljena_temp, meja, zg_meja, sp_meja;
        unsigned char i=0;
        unsigned char toleranca;
        int razlika;
        int podatki[7];

static void interrupt prekinitev()      //podprogram, ki se izvede ob prekinitvi
{
        stevec++;               //spremenljivka za dolocanje zakasnitev;
        w++;                    //spremenljivka za hitrost izpisovanja na display
        if(w>=100)              //na vsako stoto prekinitev se vrednosti na zaslonu osvezijo
        {
                podatki[0]=podatki[4];
                podatki[1]=podatki[5];
                podatki[2]=podatki[6];
                w=0;
        }
        PORTC=(PORTC&0x0F)|linija[i];   //Izbira i-tega LED displaya
        PORTB=stevila[podatki[i]];      //zapis stevila v izbran display
        if(i==1)
                PORTB=(PORTB|0x80);     //pri drugi stevilki vkljucimo decimalno piko (RB7=1)
        i++;                            //izbira naslednjega displaya in stevilke
        if(i>3)
                i=0;
        T0IF=0;
}
//----------
void zapis(x)           //podprogram za razstavitev neke vrednosti na cifre in njihov vpis v spremenljivke
za izpis
{                       // na display.
        podatki[4]=x/100;       //desetice
        z=x%100;        //v spremenljivko si shranimo ostanek deljenja z sto
        podatki[5]=z/10; //enice
        podatki[6]=z%10; //prva decimalka
        if(podatki[4]==0)
                podatki[4]=12;  //ce je desetica nic, je ne izpišemo, display ostane prazen
        return;
}
//-----------
unsigned int branje_senz()      //podprogram za branje senzorja(dejanske temperature)
{
        ADCON0=0x81;            //(0b10000001)izberemo kanal AN0 (senzor)
        ADRESH=0;
        ADRESL=0;
        temperatura=0;                  //spremenljivko postavimo na nic, da bo vsota pravilna
        for(t1=0; t1<16; t1++)          //sestnajstkrat izmerimo temperaturo in sestejemo vse vrednosti
```

```
        {
                ADGO=1;
                while(ADGO==1);
                temperatura=(temperatura+(ADRESL|(ADRESH<<8)));    //merimo v 10-bitni locljivosti
        }
        temperatura=temperatura/16;       //izracunamo povprecje izmerjenih vrednosti
        return temperatura;               //vrnemo vrednost v spremenljivki temperatura
}
void branje_potenc()              //podprogram za branje iz potenciometra
{
        ADCON0=0x99;              //(0b10011001) izbira kanala AN3 (potenciometer)
        ADRESH=0x00;
        ADRESL=0x00;
        ADGO=1;                   //start AD pretvorbe
        while(ADGO==1);           //cakanje na izvedbo AD pretvorbe
        return;
}
//-------------------------------------------
void nastavitev()         //podprogram za nastavljanje željene temperature
{
        RC0=1;            //vklop signalizacijske diode
        while(1)
        {

                do        //zanka za nastavitev temperature, izvedla se bo vsaj enkrat

                {
                        branje_potenc();
                        zeljena_temp=(ADRESL|(ADRESH<<8));
                        zeljena_temp=(zeljena_temp*42)/43;        //vrednosti 0-1023 pretvorimo v
vrednosti od 0-999
                        zapis(zeljena_temp);
                        meja=((histereza*zeljena_temp)/100);
                        sp_meja=(zeljena_temp-meja);
                        zg_meja=(zeljena_temp+meja);
                        branje_potenc();
                        test1=ADRESL|(ADRESH<<8);         //prva vrednost AD pretvorbe
                        stevec=0;
                        while(stevec<31);         //zakasnitev 0.1 sekunde
                        branje_potenc();
                        test2=ADRESL|(ADRESH<<8);         //druga vrednost AD pretvorbe
                        razlika=abs(test1-test2);
                }
                while(razlika>toleranca);         //Zanko ponavljamo, dokler premikamo potenciometer
                branje_potenc();
                test1=ADRESL|(ADRESH<<8);
                y=0;
                for (t1=0; t1<=200; t1++)//zakasnitev dveh sekund, med katero 200 krat preverimo
                {                                 // spremembo položaja potenciometra
                        for(t2=0; t2<3100; t2++)
                                asm("nop");
                        branje_potenc();
                        test2=ADRESL|(ADRESH<<8);
                        razlika=0;
                        razlika=abs(test1-test2);
                        if(razlika>toleranca)
                                break;
                        else
                                y++;
                }
                if(y>=200)
                        break;            //prekinemo zanko while(1)
        }
        RC0=0;            //izklop signalizacijske diode
        return;
}
```

```
void main()                //zacetek glavnega programa
{
        podatki[3]=11;             //Konstanta -> crka 'C'
        toleranca=5;               //Za toleranco izberemo neko majhno vrednost, recimo 5.
        histereza=10;              //zacetna vrednost histereze (%)
        TRISA=0x19;                //(0b00011001) RA0(AN0) in RA3(AN3) ter RA4 definirani kot vhodi, RA5 izhod
        TRISB=0x00;                //PORTB in PORTC definiramo kot izhode
        TRISC=0x00;
        PORTA=0x00;                //Resetiramo vse izhode
        PORTB=0x00;
        PORTC=0x00;
        OPTION=0x05;               //(0b00000101 )Preddelilnik casovnika Timer0 nastavljen na 1:64
        INTCON=0xA0;               //(0b10100000)Prekinitve casovnika Timer0 omogocene
        ADCON1=0x84;               //(0b10000100)AN3 in AN0 analogna vhoda, desna razporeditev bitov
                                   // rezultata A/D pretvorbe
        //-----------------------------------------------------------------------------
        nastavitev();      //Funkcija za nastavljanje željene temperature
        while(1)           //neskoncna zanka
        {
                RC1=1;
                temperatura=branje_senz();  //temperatura je vrednost, ki jo vrne funkcija branje_senz()
                temperatura=(temperatura*42)/43;      //skaliranje vrednosti v obmocje 0-999
                if(temperatura > zg_meja)
                        RA5=0;                        //temperatura visja od zgornje meje -> izklopimo grelec
                if(temperatura < sp_meja)
                        RA5=1;                        //temperatura nizja od spodnje meje -> vklopimo grelec
                zapis(temperatura);                   //temperaturo vpisemo v spremenljivke za izpis na display
                //--------------------
                if(((RA4)&&(RC5))==1)                 //tipka T3 -> prikaz in vecanje histereze
                {
                        RC1=0;
                        histereza++;
                        if(histereza>10)
                                histereza=0;      //histereza je lahko najvec 10%.
                        zapis(histereza);  //za dve sekundi prikažemo trenutno vrednost histereze
                        stevec=0;          //stevec zakasnitev postavimo na nic, da lahko štejemo prekinitve.
                        while(stevec<610); //zakasnitev 2 sekundi
                        nastavitev();              //histereza se je spremenila, zato izracunamo nove meje
                }
                if(RC7==1&RA4==1)          //tipka T1 -> prikaz nastavljene temperature
                {
                        RC1=0;
                        nastavitev();
                }
                if(RC4==1&RA4==1)                     //tipka T4 -> prikaz spodnje meje temperature
                {
                        zapis(sp_meja);
                        RC1=0;
                        stevec=0;
                        while(stevec<610); //2 sec zakasnitev,vrednost ostane na displayu izpisana 2 sekundi
                }
                if(RC6==1&RA4==1)          //tipka T2 -> prikaz zgornje meje temperature
                {
                        zapis(zg_meja);
                        RC1=0;
                        stevec=0;
                        while(stevec<610);
                }
        }
}
```

Program 5-1: Popoln zgled programa v C-jeziku – dvotočkovna histerezna regulacija temperature

6. Literatura

[1] James M. Sibigtroth: Understanding small microcontrollers; Prentice Hall, 1993; ISBN 0-13-089129-0, 250 strani

[2] Nebojša Matić: PIC mikrokontroleri; Mikroelektronika, Beograd 2002; ISBN 86-902189-4-7; 276 strani

[3] Matjaž Colnarič: Osnove digitalne tehnike v računalništvu; UM-FERI, Maribor, 2002; ISBN 86-435-0435-8; 138 strani

[4] Miran Rodič, Janez Pogorelc: Navodila za delo z modulom MPU-PIC16F876 (interno gradivo), [http://www.ro.feri.uni-mb.si/predmeti/mikro_el/nav_mpu_pic.pdf]; UM-FERI, Maribor, 2002

[5] Darko Dužanec: Programiranje PIC-ev v C-ju (serija člankov), Svet elektronike, maj 2004 do julij 2004, številke 109-111, ISSN 1318-4679

[6] Jernej Škvarč: PIC od začetka (serija 9 člankov), Svet elektronike, marec 2003 do januar 2004, številke 96-105, ISSN 1318-4679

[7] Microchip: spletna stran proizvajalca PIC mikrokrmilnikov, [http://www.microchip.com/]

[8] Microchip: Priročnik za mikrokrmilnike PIC16F87x (angl.), [http://www.ro.feri.uni-mb.si/predmeti/mikro_el/FTP/PIC16F876_30292b.pdf], 2000, 200 strani

[9] Microchip: Priročnik za MPLAB ICD (angl.), [http://www.ro.feri.uni-mb.si/predmeti/mikro_el/FTP/ICD_51184d.pdf], 2000, 104 strani

[10] Microchip: MPLAB V5.70 integrirano programsko okolje (program), [http://www.ro.feri.uni-mb.si/predmeti/mikro_el/FTP/mp57full.zip], 2003, 13 MB

[11] Brian W. Kernicham, Dennis M. Ritchie: Programski jezik C (slovenski prevod), 1990, UM-FERI Ljubljana, 240 strani

[12] HI-TECH Software: spletna stran proizvajalca C-prevajalnika, [http://www.htsoft.com/]

[13] HI-TECH Software: priročnik C-prevajalnika HI-TECH PICC Lite (angl.), [http://www.htsoft.com/files/demo/piccliteman.zip], 2002, 358 strani, 1,4 MB

[14] HI-TECH Software: C-prevajalnik HI-TECH PICC Lite (program), [http://www.htsoft.com/files/demo/picclite-setup.exe], 2,4 MB

[15] Miran Rodič: Uvod v programski jezik C, (interno gradivo) [http://www.ro.feri.uni-mb.si/predmeti/sis_meh/vaje/UvodC.pdf], UM-FERI, 2001, 14 strani

[16] Hobby Electronics: spletna navodila za uporabo PIC mikrokrmilnikov (angl.), [http://www.interq.or.jp/japan/se-inoue/e_pic.htm]

[17] Hobby Electronics: spletni primeri uporabe PIC mikrokrmilnikov (angl.), [http://www.interq.or.jp/japan/se-inoue/e_pic6.htm]

[18] MicrochipC.com: spletni primeri C-programov za PIC-e (angl.), [http://www.microchipc.com/]

[19] MicrochipC.com: Bootloader – najpreprostejši programator PICev (angl.), [http://www.microchipc.com/PIC16bootload/]

[20] A. Tetičkovič, B. Brečko, B. Gračner: Mobilni robot za sledenje črti, Uvodni seminar skupine študentov 3.l. Mehatronika, [http://www.ro.feri.uni-mb.si/predmeti/skup_sem/projektu/MobRob_Tet_Bre_Gra.pdf], UM-FERI, 2003

[21] J. Horvat, I. Kodrič: Poročilo o izdelavi mobilnega robota, Uvodni seminar skupine študentov 3.l. Mehatronika, [http://www.ro.feri.uni-mb.si/predmeti/skup_sem/projektu/protokol Kodric_meha-rudi.pdf], UM-FERI, 2003, 14 strani

[22] Tekmovanje RoboT200X: spletni članki z opisi mini mobilnih robotov na osnovi PIC mikrokrmilnikov, [http://www.ro.feri.uni-mb.si/tekma/dodatne_informacije_nasveti.htm]