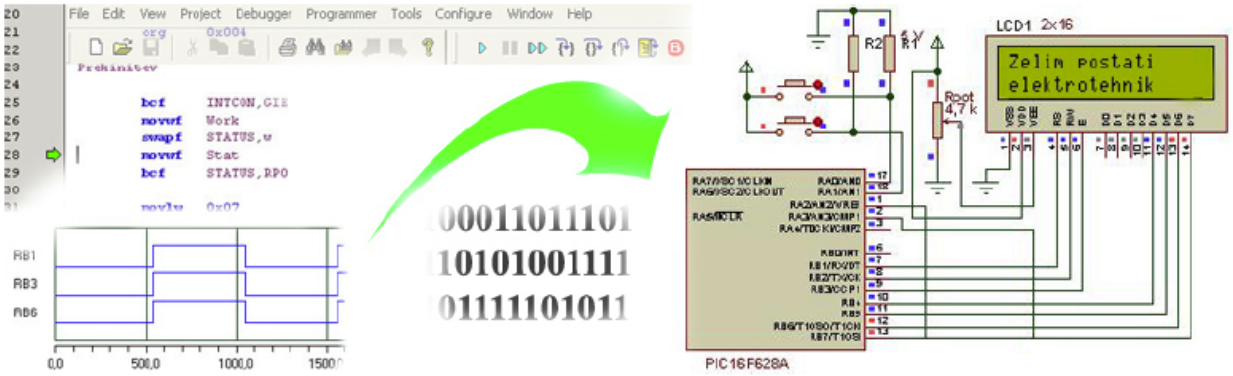
KONZORCIJ ŠOLSKIH CENTROV
REPUBLIKA SLOVENIJA
MINISTRSTVO ZA ŠOLSTVO IN ŠPORT
Naložba v vašo prihodnost
OPERACIJO DELNO FINANCIRA EVROPSKA UNIJA
Evropski socialni sklad

Elektrotehnik

UPORABA MIKROPROCESORSKIH NAPRAV



Milan Ivič

www.bodiprofi.si

munus 2

Uvajanje novih izobraževalnih programov v srednjem poklicnem in strokovnem izobraževanju s področja tehnike za obdobje 2008-2012.
Operacijo delno sofinancira Evropska unija iz Evropskega socialnega sklada in Ministrstvo za šolstvo in šport.

SPLOŠNE INFORMACIJE O GRADIVU

Izobraževalni program: Elektrotehnik

Ime modula: Uporaba mikroprocesorskih naprav – OIM7

Naslov učnega gradiva: Uporaba mikroprocesorskih naprav

Naslov učnih tem ali kompetenc, ki jih obravnava učno gradivo:
Zgradba, delovanje in uporaba mikroprocesorskih vezij.
Programiranje v zbirnem jeziku.
Izdelava in uporaba algoritmov.
Uporaba razvojnega okolja za mikrokontroler.
Programiranje mikrokontrolerjev.
Izdelava mikroprocesorskih vezij.

Avtor: Milan Ivič

Recenzent: Martin Škorjanc
Lektorica: mag. Klementina Podvršnik

CIP - Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

Ivič, M.
MUNUS2 [Elektronski vir] : Uporaba mikroprocesorskih naprav / Milan Ivič. - El. knjiga. -
Kranj : Konzorcij šolskih centrov, 2011.

Način dostopa (URL): http://munus2.tsc.si. - Projekt MUNUS 2

ISBN xxxxxxxxxxxxx
xxxxxxxxxx

Izdajatelj: Konzorcij šolskih centrov Slovenije v okviru projekta MUNUS 2
Slovenija,julij 2011

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Povzetek

Gradivo Programiranje in uporaba mikroprocesorskih naprav je namenjeno dijakom 4. letnika SSI – smer elektrotehnik in dijakom 2 letnika PTI – smer elektrotehnik. Zajema vsebinski del, naveden v katalogu znanja. Za programski jezik je izbran zbirni jezik. Primeri programov, ki so izdelani v razvojnem okolju MPLAB IDE v8.70, so namenjeni Microchipovemu mikrokontrolerju PIC. Gradivo poleg opisa instrukcij zbirnega jezika, razhroščevanja in predstavitve razvojnega okolja vsebuje primere uporabe mikrokontrolerjev s komentarji in potrebnimi načrti. Na koncu poglavij so vaje, namenjene utrjevanju znanja in razvijanju algoritmičnega razmišljanja. Primeri programov, ki so napisani v gradivu, so le ena od možnih rešitev s poudarkom na razumevanju obravnavanih tem.

Ključne besede: mikrokontroler, zbirni jezik, algoritem, diagram poteka, programiranje mikrokontrolerja, mikroprocesorska vezja, vhodno-izhodne enote mikrokontrolerja, prekinitve v mikrokontrolerju, instrukcije, parametri, pulzno širinska modulacija.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kazalo

Kazalo vsebine:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kazalo slik:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kazalo tabel:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Kazalo enačb:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

UČNI CILJI

Dandanes si življenja brez avtomatizacije ne predstavljamo. Skoraj vsako napravo, ki jo dnevno uporabljamo, krmili kateri od mikrokontrolerjev, ki skrbi za njeno pravilno delovanje. Če pogledamo samo avtomobile spoznamo, da programirani procesi neprestano skrbijo za varnost voznikov in drugih udeležencev v prometu, pa tudi za lažje in učinkovitejše upravljanje naprav v njih. V proizvodnih procesih se stremi k zmanjševanju stroškov dela in k čim večji kakovosti izdelkov, zato se vsaka delovna operacija najde na seznamu potreb po avtomatizaciji. Zaradi tega sta postala poznavanje osnov programiranja mikroprocesorskih naprav in znanje algoritmičnega razmišljanja osnovni sestavini funkcionalne pismenosti elektrotehnika, ki se uporablja pri:

- branju navodil in postopkov za uporabo posamezne elektrotehnične naprave;
- pri vzdrževanju in popravilu elektrotehniških in drugih naprav;
- naročanju in izbiranju posameznih elektrotehniških elementov in naprav;
- zamenjavi množice elementov v posameznih napravah z mikroprocesorji, ki nadomestijo njihovo vlogo;
- uporabi vrste senzorjev v napravah za krmiljenje in regulacijo procesov;
- načrtovanju novih rešitev elektrotehniških in drugih naprav;
- popravilu programov mikroprocesorjev;
- zamenjavi mikroprocesorskih programov z novimi.

CILJI

- spoznavanje zgradbe in delovanja mikroprocesorskih vezij;
- poznavanje in uporaba razvojnega okolja MPLAB IDE in izdelava programov;
- načrtovanje in izdelava mikroprocesorskih vezij;
- razvijanje algoritmičnega razmišljanja;
- načrtovanje in izdelava programov za mikroprocesorje;
- programiranje mikroprocesorskih vezij;
- izvajanje krmiljenja, zajemanja in regulacije z mikroprocesorskimi vezji;
- uporaba znanih rešitev na novih primerih programov za mikroprocesorje;
- razvijanje sposobnosti ustvarjanja in predstavitev dela v skupini;
- iskanje informacij za obstoječe in nove rešitve v programih.

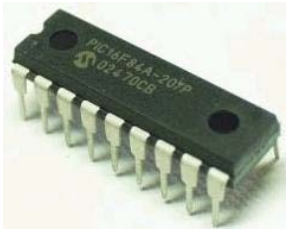
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

MIKROKONTROLER PIC

PIC (Peripheral Interface Controller) je družina mikrokontrolerjev proizvajalca Microchipa. V tej družini obstaja veliko mikrokontrolerjev z različnimi zmogljivostmi. Najenostavnejši so zgrajeni v 8-pinskem ohišju (ohišju z 8 nožicami, priključki), zmogljivejši pa so zgrajeni v 40-pinskem ohišju. Na enem integriranem vezju vsebujejo centralno procesno enoto (CPE), programski in podatkovni pomnilnik ter različne vhodno-izhodne naprave. Njihovo uporabnost določa program, ki ga vanje vpišemo. Pri programiranju bomo uporabili predvsem mikrokontroler PIC16F628A, zato moramo poznati njegove osnovne karakteristike.

Mikrokontroler PIC16F628A

Mikrokontroler PIC16F628A je Microchipov mikrokontroler, ki ga dobimo v 18-pinskem DIL-ohišju.



Slika 1: PIC16F628A

Njegove najpomembnejše značilnosti so:

- Ima 16 vhodno-izhodnih pinov, ki jih lahko programsko krmilimo.
- Napetost napajanja Vdd je 5 V.
- Vsebuje 2 k programskega pomnilnika Flash, 224 bajtov pomnilnika RAM in 128 bajtov podatkovnega pomnilnika EEPROM.
- Vsebuje časovnike TMR0, TMR1 in TMR2 in pozna 10 načinov sprožitev prekinitev.
- Centralna procesna enota (CPE) je 8-bitna.
- Maksimalna frekvenca zunanjega oscilatorja je 20 MHz.
- Vsebuje notranja 4 MHz- in 48 kHz-oscilatorja.
- Vsebuje dva analogna komparatorja.
- Omogoča PWM (**P**ulse **w**idth **m**odulation), pulzno širinsko modulacijo.
- Pozna 35 instrukcij za programiranje v zbirnem jeziku.
- Centralna procesna enota CPE izvaja instrukcije. Vgrajen ima 8-bitni delovni register, v katerega se shranjujejo podatki po izvedbi posameznih instrukcij.
- Programski pomnilnik je sestavljen iz 2048 lokacij. V vsako lokacijo lahko vpišemo 14-bitni podatek. Instrukcija zavzame eno lokacijo. Posebno vlogo imata prva in peta

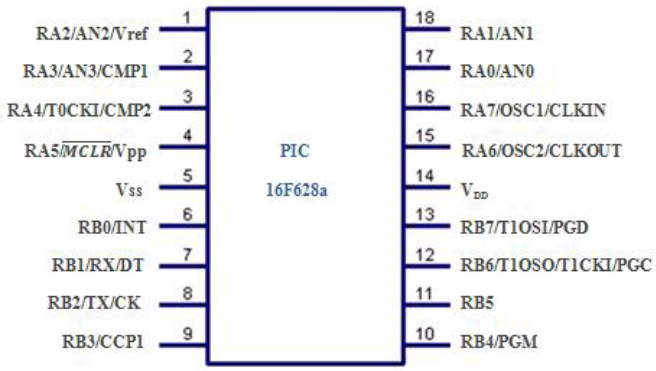
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

lokacija. V prvi se nahaja instrukcija oziroma ukaz, ki se bo izvedel takoj po vklopu mikrokontrolerja. V peti se začne del programa, ki se izvede ob prekinitvi.

- Podatkovni pomnilnik RAM je razdeljen na štiri banke: banko 0, banko 1, banko 2 in banko 3. Prvih 32 lokacij je namenjenih SFR-registrom (**S**pecial **f**unction **r**egisters), ki služijo za komuniciranje z ostalimi enotami mikrokontrolerja. Na naslednjih 96 lokacijah je 224 bajtov GPR-registrov (**G**eneral **p**urpose **r**egisters). Na zadnjih 16 lokacijah se GPR-registri prelivajo v vse štiri banke.

Podrobnejše podatke najdemo na spletni strani proizvajalca mikrokontrolerja PIC16F628A.

Opis osnovne vloge posameznih priključkov mikrokontrolerja PIC16F628A:

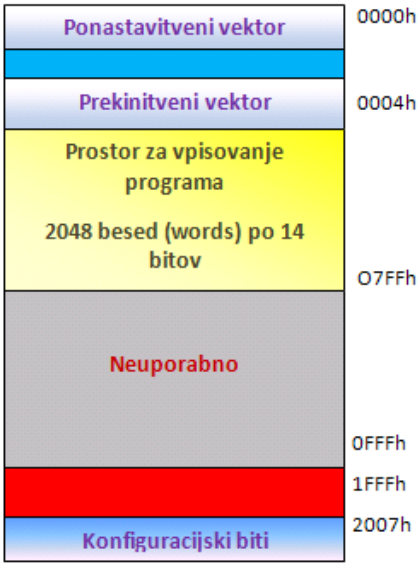


Slika 2: Oznake priključkov mikrokontrolerja PIC16F628A

- OSC1 – vhod zunanjega kristalnega oscilatorja (določanje takta).
- OSC2 – izhod zunanjega kristalnega oscilatorja.
- MCLR – "master clear", ponastavitveni vhod (reset). Ponastavljamo s stanjem logične 0, delovanje programa v stanju logične 1.
- RA0 do RA7 – vhodno-izhodni pini na porta.
- RA4/T0CKI – pin je tipa "open drain". Pozna dve stanji: maso, kadar je pin v stanju logične 1, in veliko impedanco, ko je v stanju logične 0. Če vežemo svetlečo diodo med +5 V in pin RA4, se bo vklopila, če bomo ta pin postavili v stanje logične 0. Če pa vežemo svetlečo diodo med pin RA4 in maso, se ne bo nikoli vklopila. Pin RA4 lahko uporabimo za štetje impulzov s časovnikom TMR0.
- RB0/INT – pin RB0 je lahko izbran za zunanji vir prekinitev. Lahko ga določimo kot vhodni ali kot izhodni pin mikrokontrolerja.
- RB1 do RB7 – vhodno-izhodni pini. Pini RB4 do RB7 so lahko izbrani za zunanji vir prekinitev.
- Vss – masa.
- V_{DD} – napetost napajanja +5 V.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Programski pomnilnik Flash:
Programski pomnilnik Flash uporabljamo za vpisovanje programa v mikrokontroler. Vanj lahko shranimo 2048 besed (word), dolgih 14 bitov od naslova 0000h do 07FFh. Vsebina programskega pomnilnika Flash se ne izbriše, če mikrokontrolerju izklopimo napajanje. Po podatkih proizvajalca se vsebina ohrani najmanj 40 let. Programski pomnilnik Flash omogoča večkratno vpisovanje programa, po podatkih proizvajalca do 100.000 vpisov. Ponastavitveni (reset) vektor se nahaja na naslovu 0000h, prekinitveni vektor pa na naslovu 0004h.

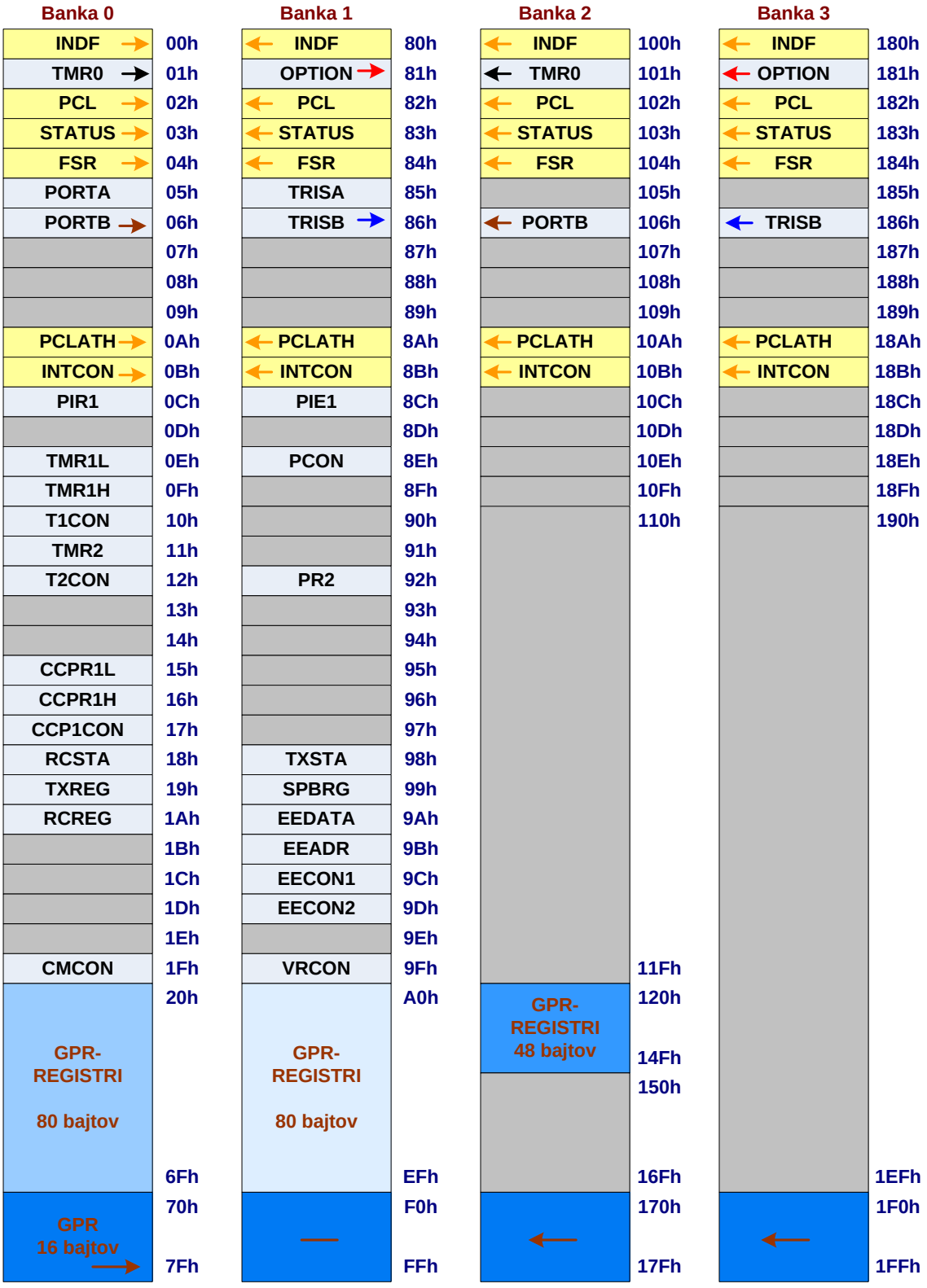


Slika 3: Programski pomnilnik Flash

Podatkovni pomnilnik RAM:
Podatkovni pomnilnik RAM (**R**andom **a**ccess **m**emory) je razdeljen na štiri banke: banko 0, banko 1, banko 2 in banko 3. Vsaka banka ima 128 bajtov spominskega prostora. Vsebuje SFR- in GPR-registre. SFR-registri zavzemajo prvih 32 lokacij v vsaki banki. Preko njih komuniciramo z ostalimi enotami mikrokontrolerja. Na sliki 4 vidimo, da se nekateri SFR-registri in njihove vrednosti prelivajo v druge banke. Tako se na primer register PORTB iz banke 0 (naslov 06h) preliva z registrom PORTB v banki 2 (naslov 106h).
GPR-registri zavzemajo 224 bajtov podatkovnega pomnilnika RAM, od katerih se jih 16 bajtov na zadnjih lokacijah preliva v vse štiri banke. V banki 0 zavzemajo GPR-registri še 80 bajtov (od 20h do 6Fh), v banki 1 tudi 80 bajtov (od A0h do EFh), v banki 2 pa še 48 bajtov (od 120h do 14Fh) podatkovnega pomnilnika RAM. Vsebine GPR-registrov se izgubijo, ko mikrokontrolerju izklopimo napajanje.
Če želimo brati ali pisati v register podatkovnega pomnilnika RAM, se moramo nahajati v banki, v kateri je želeni register.

Programski števec:
V mikrokontrolerju PIC16F628A je programski števec 13-biten register. V njem se vedno nahaja naslov naslednje instrukcije, ki se bo v programu izvedla. Dosegljiv je preko PCLATH- in PCL-registrov. PCL-register vsebuje spodnjih 8 bitov programskega števca. Lahko ga poljubno beremo ali pa vanj vpisujemo. Zgornjih 5 bitov programskega števca pa najdemo v registru PCLATH, ki ni direktno dosegljiv.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.



Legenda:

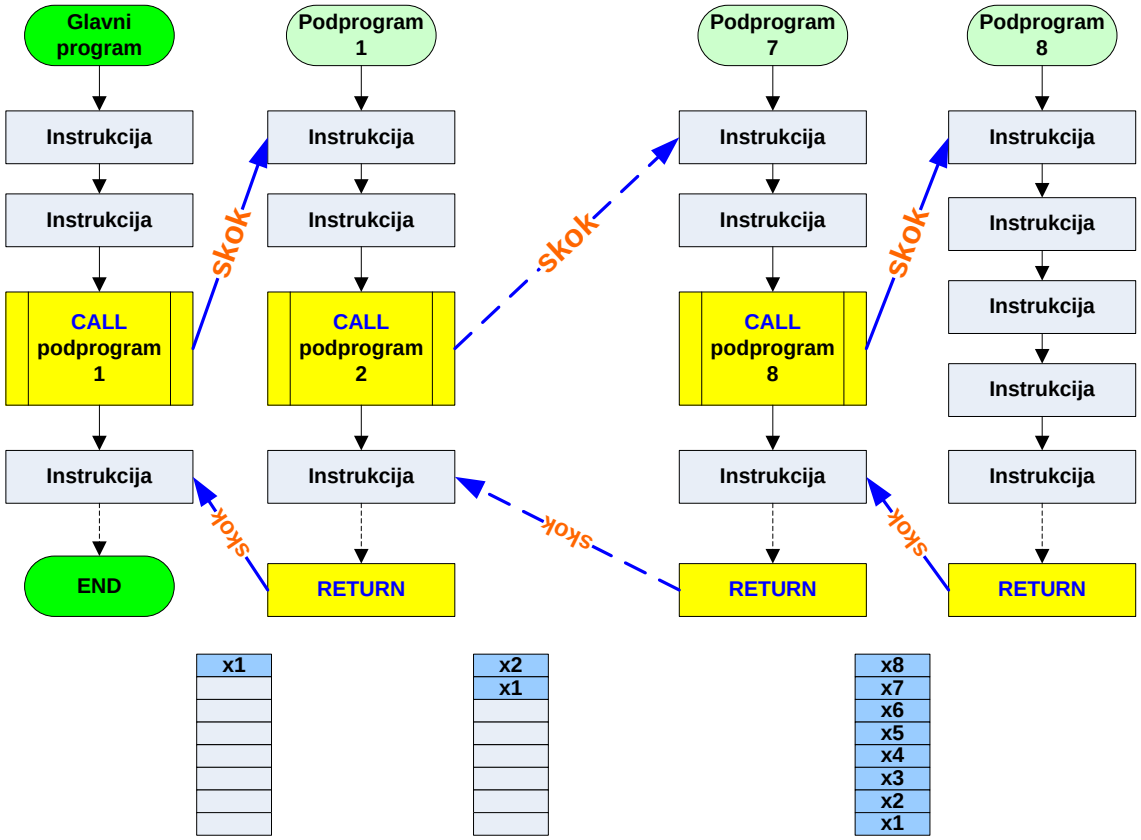
: **Naslovi oz. lokacije v mikrokontrolerju PIC16F628a niso podprte.**

: **Vrednosti registrov, kjer so puščice enake barve, se prelivajo iz ene banke v drugo (druge).**

Slika 4: Organizacija podatkovnega pomnilnika RAM

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Podprogrami:
Podprogram je del programa, ki je ločen od glavnega programa. Pokličemo ga po potrebi. Če želimo, da se neki del programa izvede večkrat, za ta del programa izdelamo podprogram. Podprogram iz glavnega programa pokličemo z instrukcijo CALL. V parametru te instrukcije navedemo labelo oziroma naslov, ki pomeni ime podprograma. Podprogram vedno začnemo z labelo, zaključimo pa z instrukcijo RETURN, ki poskrbi, da se iz podprograma vrnemo v glavni program na instrukcijo, ki sledi instrukciji CALL.
Instrukcija CALL povzroči skok na podprogram z labelo, ki je zapisana v parametru te instrukcije. Podprogrami se nahajajo na koncu glavnega programa, pred direktivo END. Če bi ga zapisali kar med glavnim programom, bi se izvedel tudi takrat, ko ga ne bi poklicali.
Podprograme lahko pokličemo tudi iz podprograma, vendar le do meje 8 podprogramov v globino. Delovanje podprogramov prikazuje slika 5.



Slika 5: Delovanje podprogramov

Aritmetična logična enota:
Za izvajanje vseh instrukcij v mikrokontrolerju PIC skrbi 8-bitna aritmetična logična enota ALU (**A**rithmetic **l**ogic **u**nit). Lahko sešteva in odšteva 8-bitna števila ter z njimi izvaja nekatere logične operacije (AND, OR, XOR, NOT …). Po izvršitvi vsake instrukcije se v registru STATUS postavijo trije biti na ustrezne vrednosti. Iz njihovih vrednosti lahko ugotovimo, kakšen je dobljeni rezultat. Ta je lahko pozitiven, negativen, enak 0 ali napačen. Te tri bite imenujemo zastavice.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

SFR-registri:
V gradivu so opisani tisti SFR-registri, ki jih bomo uporabljali pri izdelavi vaj. Podrobnejše opise najdemo v kataloških podatkih proizvajalca mikrokontrolerja PIC16F628A.

Register STATUS (RAM naslov: 03h, 83h, 103h, 183h)

Bit 7	6	5	4	3	2	1	Bit 0
IRP	**RP1**	**RP0**	**/T0**	**/PD**	**Z**	**DC**	**C**

Tabela 1: Register STATUS

- Bit 0, C (**C**arry bit): Zastavica C se pri seštevanju postavi na 1, če nastane prenos na bitu 8 (najpomembnejši bit) dobljenega rezultata. Zastavica C se pri seštevanju postavi na 0, če ni prenosa na bitu 8 dobljenega rezultata.
- Bit 1, DC (**D**igit **c**arry): Zastavica DC se pri seštevanju postavi na 1, če nastane prenos med 4. in 5. bitom dobljenega rezultata. Zastavica DC se pri seštevanju postavi na 0, če ni prenosa med 4. in 5. bitom dobljenega rezultata.
- Bit 2, Z (**Z**ero): Zastavica Z se postavi na 1, če je rezultat zadnje instrukcije število 0. Zastavica Z se postavi na 0, če rezultat zadnje instrukcije ni enak številu 0.
- Bit 3, /PD (**P**ower-**d**own bit): 1 => po vklopu, z instrukcijo clrwdt. 0 => v režimu delovanja SLEEP.
- Bit 4, /TO (**T**ime-**o**ut bit): 1 => po vklopu, z instrukcijo clrwdt, z instrukcijo sleep. 0 => WDT (**W**atch**d**og **t**imer) je prekoračil vrednost.
- Bit 5-6, RP: bita za izbiro banke podatkovnega pomnilnika RAM pri direktnem naslavljanju.
  00 => banka 0 (00h–7Fh)
  01 => banka 1 (80h–FFh)
  10 => banka 2 (100h–17Fh)
  11 => banka 3 (180h–1FFh)
- Bit 7: IRP: bit za izbiro banke podatkovnega pomnilnika RAM pri indirektnem naslavljanju.
  1 => banki 2, 3 (100h – 1FFh)
  0 => banki 0, 1 (00h – FFh)

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Register OPTION (naslov RAM: 81h, 181h)

Bit 7	6	5	4	3	2	1	Bit 0
/RBPU	**INTEDG**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**

Tabela 2: Register OPTION

- Bit 0-2: Biti PS0, PS1 in PS2 se uporabljajo za nastavitev preddelilnika. Z njihovo vrednostjo določimo, kolikšen bo faktor deljenja preddelilnika (slika 9). Kateremu časovniku bo služil preddelilnik, določa bit 3 (PSA) v registru OPTION.

PS2	PS1	PS0	TMR0	WDT
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

Tabela 3: Preddelilnik

- Bit 3, PSA (**Pres**caler **a**ssignment): 1 => preddelilnik dodeli stražnemu mehanizu. 0 => preddelilnik dodeli časovniku.
- Bit 4, T0SE (**T**imer **0** **s**ource **e**dge select): 1 => časovnik se bo povečal pri prehodu iz visokega logičnega nivoja na pinu RA4 v nizkega. 0 => časovnik se bo povečal pri prehodu iz nizkega logičnega nivoja na pinu RA4 v visokega.
- Bit 5, T0CS (**T**imer **0** **c**lock **s**ource select): 1 => časovnik dobiva impulze za proženje iz pina RA4. 0 => časovnik dobiva impulze za proženje iz strojnega takta.
- Bit 6, INTEDG (**Int**errupt **edg**e select): 1 => prekinitev na pinu RB0 se sproži ob prehodu signala iz nizkega logičnega nivoja v visokega. 0 => prekinitev na pinu RB0 se sproži ob prehodu signala iz visokega logičnega nivoja v nizkega.
- Bit 7, RBPU (PO**RT**B **p**ull-**u**p enable bit): 1 => izključi upore, ki ohranjajo priključke PORTB na potencialu logične 1. 0 => vključi upore, ki ohranjajo priključke PORTB na potencialu logične 1.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Register INTCON (RAM naslov: 0Bh, 8Bh, 10Bh, 18Bh)

Bit 7	6	5	4	3	2	1	Bit 0
GIE	**PEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**

Tabela 4: Register INTCON

- Bit 0, RBIF (PO**R**T**B** change **i**nterrupt **f**lag): 1 => nastala je prekinitev na pinih RB4−RB7. 0 => ni prekinitve na pinih RB4−RB7.
- Bit 1, INTF (RB0/**INT** interrupt **f**lag): 1 => nastala je prekinitev na pinu RB0. 0 => ni prekinitve na pinu RB0.
- Bit 2, T0IF (**T**MR**0** overflow **i**nterrupt **f**lag): 1 => nastala je prekinitev na časovniku TMR0. 0 => ni prekinitve na časovniku TMR0.
- Bit 3, RBIE (PO**R**T**B** change **i**nterrupt **e**nable): 1 => omogočeno je proženje prekinitev s spremembo logičnega nivoja na pinih RB4−RB7. 0 => proženje prekinitev s spremembo logičnega nivoja na pinih RB4−RB7 je onemogočeno.
- Bit 4, INTE (RB0/**INT** interrupt **e**nable): 1 => omogočeno je proženje prekinitev s spremembo logičnega nivoja na pinu RB0. 0 => proženje prekinitev s spremembo logičnega nivoja na pinu RB0 je onemogočeno.
- Bit 5, T0IE (**T**MR**0** overflow **i**nterrupt **e**nable): 1 => omogočeno je proženje prekinitev na časovniku TMR0. 0 => proženje prekinitev na časovniku TMR0 je onemogočeno.
- Bit 6, PEIE (**Pe**ripheral **i**nterrupt **e**nable): 1 => omogočimo odzivanje mikrokontrolerja na prekinitve, ki jih določimo v registru PIE 1 (**P**eripheral **i**nterrupt **e**nable). 0 => onemogočimo odzivanje mikrokontrolerja na prekinitve, ki jih določimo v registru PIE 1.
- Bit 7, GIE (**G**lobal **i**nterrupt **e**nable): 1 => omogočeno je odzivanje mikrokontrolerja na prekinitve. 0 => onemogočeno je odzivanje mikrokontrolerja na prekinitve.

Povzetek

Spoznali smo osnovne značilnosti mikrokontrolerja PIC16F628A. Na enem integriranem vezju vsebuje centralno procesno enoto, programski in podatkovni pomnilnik ter različne vhodno-izhodne naprave. Njihovo vlogo v elektronskem vezju določa program, ki ga vanj vpišemo. Mikrokontroler PIC16F628A vsebuje 18 vhodno-izhodnih pinov, ki jih lahko programsko krmilimo. Centralna procesna enota je 8-bitna, izvaja instrukcije ter tako obdeluje in prenaša podatke. V ta namen ima vgrajen 8-bitni delovni register. Vsebuje 2 k programskega pomnilnika Flash, ki omogoča večkratno vpisovanje programa. Med dodatnimi enotami najdemo še tri časovnike, dva komparatorja, pulzno širinsko modulacijo in 10 načinov proženja prekinitev. Podatkovni pomnilnik RAM je razdeljen na štiri banke ter vsebuje SFR- in GPR-registre. Preko SFR-registrov komuniciramo z ostalimi enotami na čipu, GPR-registri pa so nam na voljo za shranjevanje podatkov. Če želimo brati ali pisati v določen register podatkovnega pomnilnika RAM, se moramo nahajati v banki, v kateri je želeni register. Nekateri SFR- in GPR-registri se prelivajo iz ene banke v drugo.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Vaje

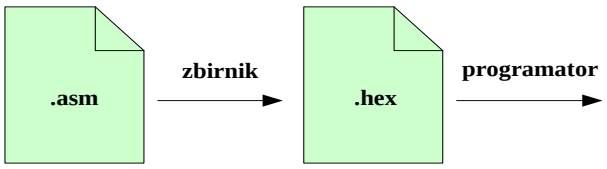
1. Na spletni strani proizvajalca mikrokontrolerjev PIC poišči karakteristične podatke za mikrokontroler PIC16F628A.
2. V karakterističnih podatkih poišči vseh 10 virov proženja prekinitev in jih komentiraj.
3. Določi vrednost registra STATUS tako, da imaš dostop bo SFR-registra TRISA, ki se nahaja v banki 2 podatkovnega pomnilnika RAM.
4. Določi vrednost registra INTCON tako, da se bo mikrokontroler PIC16F628A odzival na prekinitve, ki jih bo prožil časovnik TMR0. Katera zastavica se bo postavila na vrednost 1, ko bo časovnik TMR0 sprožil prekinitev? Ali se po proženju prekinitve zastavica sama postavi na vrednost 0? Pomagaj si s karakterističnimi podatki mikrokontrolerja.
5. Naslove SFR-registra OPTION (banka 1 in banka 3) in PORTB (banka 0 in banka 2) zapiši v binarnem in decimalnem številskem sistemu.
6. Opiši razliko med analognimi in digitalnimi signali.

PROGRAMSKO OKOLJE MPLAB IDE

MPLAB IDE je razvojno okolje za mikrokontrolerje PIC in je brezplačno na voljo na internetu, na proizvajalčevi strani http://www.microchip.com/. Vsebuje tudi simulator, ki nam pomaga pri iskanju napak v programu.
MPLAB IDE pozna poleg projektov tudi delovne prostore – workspacee. Projekt vsebuje datoteke z izvorno kodo, knjižnice in ostale datoteke, ki spadajo k programu. Delovni prostor pa vsebuje nastavitve mikrokontrolerja, prevajalnika in urejevalnika.
Za izdelavo projekta imamo na voljo čarovnika za projekte (Project Wizard). Če programiramo v zbirnem jeziku, moramo pri shranjevanju projekta datoteki dodati končnico .asm. Program pišemo v datoteki, ki smo jo ustvarili. Razvojno okolje MPLAB IDE nam sproti, med pisanjem programa obarva izvorno kodo in jo s tem napravi pregledno in čitljivo. Ko program napišemo, ga moramo prevesti. Preden pa ga prevedemo, moramo nastaviti razhroščevalnik (debugger). Ker bomo pri vajah uporabljali simulator, kliknemo na *Debugger* > *Select Tool* > *MPLAB SIM*. Nastavitve simulatorja se nahajajo v *Debugger* > *Settings*. V okno pod jezičkom *OSC/Trace* moramo vpisati frekvenco kristalnega oscilatorja, ki bo priključen na mikrokontroler. Pod jezičkom *Animation/Realtime Updates* imamo na voljo spreminjanje hitrosti izvajanja programa med simulacijo. Če hočemo simulirati spremembe signalov na vhodnih pinih mikrokontrolerja, to naredimo v oknu *Debugger* > *Stimulus* > *New Workbook*. Potek spreminjanja vhodnih pinov in odziv na ustreznih izhodnih pinih mikrokontrolerja, kar je seveda odvisno od izdelanega programa, ki se izvaja, lahko spremljamo v oknu, ki se odpre ob kliku na *View* > *Simulator Logic Analyzer*. V tem oknu dodajamo tiste pine mikrokontrolerja (jeziček *Channels*), na katerih želimo opazovati signale.
Program prevedemo z ukazom *Project* > *Build All*. Po končanem prevajanju se bo pojavilo novo okno, v katerem se bodo med drugim izpisala opozorila in morebitne napake v programu. Te se pojavijo vsaka v svoji vrstici. Ob dvokliku na posamezno vrstico z napako se pojavi okno z

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

izvorno kodo in na vrstico z napako v izvorni kodi na levi strani kaže zelena puščica. Če napak po koncu prevajanja ni, se poročilo o prevajanju konča z "BUILD SUCCEEDED".
Projekte v okolju MPLAB IDE odpiramo s *Project* > *Open* in jih zapiramo s *Project* > *Close*. Okno *Watch* je eno najpomembnejših orodij pri razhroščevanju. V njem lahko med izvajanjem programa v simulatorju spremljamo vrednosti posameznih registrov (SFR) in spremenljivk (GPR). Okno Watch odpremo z *View* > *Watch*. Spremljamo lahko štiri različne skupine registrov in spremenljivk (Watch1−Watch4). V seznam lahko s pomočjo vrstice nad seznamom dodajamo nove spremenljivke in registre. Gumb Add SFR je namenjen dodajanju SFR-registrov, gumb Add Symbol pa dodajanju spremenljivk. V oknu Watch lahko opazujemo tudi lokacije v pomnilniku, ki nimajo svojega imena. To storimo tako, da dvokliknemo na začetek prazne vrstice in vpišemo heksadecimalni naslov spremenljivke, ki jo želimo opazovati. Izberemo lahko tudi obliko izpisa vrednosti posameznih registrov in spremenljivk (binarno, decimalno, heksadecimalno …). To storimo tako, da izberemo register oziroma spremenljivko, nanjo kliknemo z desnim miškinim gumbom in nato v meniju, ki se odpre, kliknemo Properties ter izberemo med ponujenimi opcijami.
Ostali meniji imajo ukaze, ki za začetek niso tako pomembni. Po potrebi jih bomo spoznali pri vajah.
Ko program prevedemo, se v mapi, kjer je projekt shranjen, ustvari datoteka s končnico .hex. S programatorjem jo prenesemo v programski pomnilnik mikrokontrolerja. Pri tem lahko uporabimo programator PICkit 2, ki ga dobimo v prodajalnah za dokaj nizko ceno. Vsebuje tudi logični analizator, na računalnik pa ga lahko priključimo preko USB-vhoda. Programator ne potrebuje zunanjega napajanja. Pred uporabo tega programatorja na računalnik namestimo ustrezni program: PICkit 2 v2.61.



Slika 6: Prevajanje programa

Povzetek

Spoznali smo programsko okolje za programiranje Microchipovih mikrokontrolerjev. Za vsak nov projekt si izberemo mapo, v katero se bodo shranjevale datoteke novega projekta in datoteka s projektom. V projektu moramo določiti tip mikrokontrolerja, izbrati prevajalnik za zbirni programski jezik in določiti frekvenco kristalnega oscilatorja, ki ga bomo priklopili na mikrokontroler. Ko program napišemo, ga za vpis v mikrokontroler prevedemo v datoteko s končnico .hex, za kar uporabimo ustrezen programator. V programskem okolju lahko simuliramo delovanje izdelanega programa. Razhroščevalnik nam pomaga pri odpravljanju morebitnih napak v programu in opazovanje dogajanja v registrih.

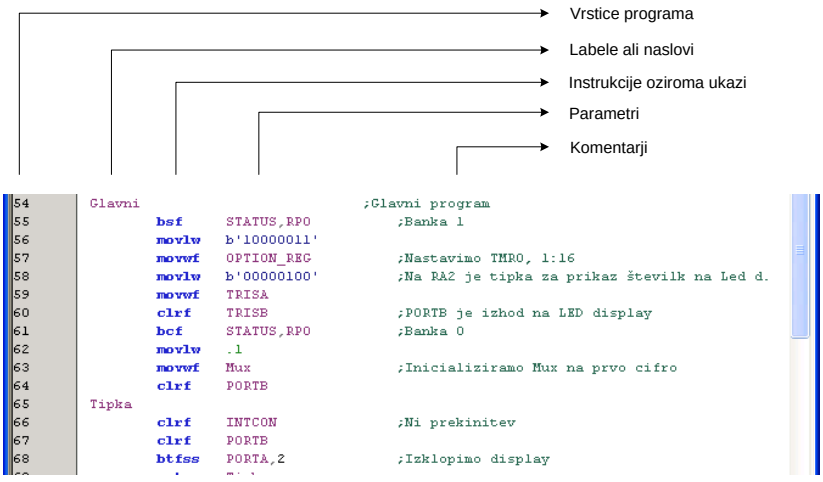
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Vaje

1. Na računalnik si namesti zadnjo verzijo programske opreme MPLAB IDE za programiranje Microchipovih mikrokontrolerjev PIC, ki jo najdeš na proizvajalčevi strani http://www.microchip.com/.
2. V programskem okolju MPLAB IDE odpri nov projekt in določi pot do mape, kamor se bo projekt shranjeval. Novemu projektu dodeli ime Vaja 1. V projektu izberi tip mikrokontrolerja PIC16F628A, določi frekvenco kristalnega oscilatorja 4 MHz ter izberi prevajalnik za zbirni programski jezik.
3. Na računalnik si namesti programsko opremo za programator PICkit 2.

ZBIRNI JEZIK

Zbirnik je programski jezik, ki ga lahko uvrstimo tik nad strojni jezik. Je jezik kratic, ki so ga uvedli programerji, da bi poenostavili uporabo strojnega jezika. Namesto da bi bili strojne kode strojnih ukazov prepisovali iz tabel, so ukaze poimenovali s kraticami, ki si jih je laže zapomniti kot številke. Zbirni jezik je predstavljen v razumljivejši obliki z mnemoniki ali smiselnimi okrajšavami. Zbirni jezik strojne ukaze oziroma instrukcije predstavlja v simbolični obliki in se ne more izvajati neposredno, saj ga je treba prej prevesti v strojni jezik.
Vsaka od 35 instrukcij mikrokontrolerja PIC16F628A ima v zbirnem jeziku svojo okrajšavo, ki jo imenujemo mnemonik. Zbirniški prevajalnik prevaja program, napisan v zbirnem jeziku po vrsticah tako, da mnemonike prevede v strojno kodo ukaza, ki jo razume centralna procesna enota mikrokontrolerja. Strojno kodo vpisuje v datoteko s končnico .hex, ki jo nato s programatorjem prenesemo v programski pomnilnik mikrokontrolerja.
Program v zbirnem programskem jeziku je sestavljen iz stolpca instrukcij, njihovih parametrov in komentarjev. Komentarji niso obvezni, vendar je program s komentarji preglednejši, saj nam ti opisujejo njegovo delovanje. V veliko pomoč so tudi pri odpravljanju napak.



Slika 7: Program v zbirnem jeziku

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Labele ali naslove pišemo skrajno levo. Postavimo jih lahko kamor koli pred instrukcijo ali v novo vrstico med programom. Instrukcije ali ukaze pišemo s tabulatorsko tipko odmaknjene od levega roba, sledijo pa jim parametri. Če je parametrov več, jih moramo ločiti z vejicami. Komentarje lahko pišemo na konec katere koli vrstice ali v svojo vrstico, vedno pa jih moramo začeti s podpičjem.

Instrukcije mikrokontrolerja

- **addlw** k — **Add l**iteral and **w**ork. Vrednost delovnega registra w prištej vrednosti konstante k. Rezultat shrani v delovni register. Instrukcija porabi 1 urin cikel.
- **addwf f** — **Add w**ork and **f**ile. Vrednosti registra f prištej vrednost delovnega registra w. Instrukcija porabi 1 urin cikel.
- **andlw** k — **AND l**iteral with **w**ork. Izvedi logično operacijo IN med delovnim registrom in konstantno vrednostjo k. Rezultat operacije shrani v delovni register. Instrukcija porabi 1 urin cikel.
- **andwf** f — **AND w**ork with **f**ile. Izvedi logično operacijo IN med vrednostjo delovnega registra in vrednostjo registra, zapisanega v parametru. Instrukcija porabi 1 urin cikel.
- **bcf** f,n — **B**it **c**lear **f**ile. V registru f postavi bit n na 0. Instrukcija porabi 1 urin cikel.
- **bsf** f,n — **B**it **s**et **f**ile. V registru f postavi bit n na 1. Instrukcija porabi 1 urin cikel.
- **btfss** f,n — **B**it **t**est **f**ile, **s**kip if **s**et. Testiraj bit n v registru f, preskoči naslednjo instrukcijo, če je 1. Instrukcija porabi 2 urina cikla pri preskoku naslednje instrukcije, če ne preskoči, pa enega.
- **btfsc** f,n — **B**it **t**est **f**ile, **s**kip if **c**lear. Testiraj bit n v registru f, preskoči naslednjo instrukcijo, če je 0. Instrukcija porabi 2 urina cikla pri preskoku naslednje instrukcije, če ne preskoči, pa enega.
- **call** a — **Call** sobroutine. Pokliči podprogram na naslovu a. Instrukcija porabi 2 urina cikla.
- **clrf** f — **Clear f**ile. V registru f postavi vse bite na vrednost 0. Instrukcija porabi 1 urin cikel.
- **clrw** — **Clear w**ork. V delovnem registru w postavi vse bite na vrednost 0. Instrukcija porabi 1 urin cikel.
- **clrwdt** — **Clear w**atch**d**og **t**imer. Vrednost časovnika stražnega mehanizma postavi na 0. Instrukcija porabi 1 urin cikel.
- **comf** f — **Com**plement **f**ile. Negiraj vrednost registra f. Instrukcija porabi 1 urin cikel.
- **decf** f — **Dec**rement **f**ile. Zmanjšaj vrednost registra f za 1. Instrukcija porabi 1 urin cikel.
- **decfsz** f — **Dec**rement **f**ile, skip if **z**ero. Zmanjšaj vrednost registra f za 1, preskoči naslednjo instrukcijo, če je vrednost registra f enaka 0. Instrukcija porabi 1 urin cikel, če pogoj ni izpolnjen, če je izpolnjen, pa 2 urina cikla.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

- **goto** a Unconditional branch. Nadaljuj izvajanje programa na naslovu a. Instrukcija porabi 2 urina cikla.
- **incf** f **Inc**rement **f**ile. Povečaj vrednost registra f za 1. Instrukcija porabi 1 urin cikel.
- **incfsz f** **Inc**rement file, skip if zero. Povečaj vrednost registra f za 1, preskoči naslednjo instrukcijo, če je vrednost registra f enaka 0. Instrukcija porabi 1 urin cikel, če pogoj ni izpolnjen, če je izpolnjen, pa 2 urina cikla.
- **iorlw** k **I**nclusive **OR** **l**iteral with **w**ork. Izvedi logično operacijo ALI med delovnim registrom w in konstanto k. Rezultat operacije shrani v delovni register w. Instrukcija porabi 1 urin cikel.
- **iorwf** f **I**nclusive **OR** **w**ork with **f**ile. Izvedi logično operacijo ALI med delovnim registrom w in registrom f. Instrukcija porabi 1 urin cikel.
- **movlw** k **Mov**e **l**iteral to **w**ork. Vrednost konstante k prestavi v delovni register w. Instrukcija porabi 1 urin cikel.
- **movf** f,d **Mov**e **f**ile. Vrednost registra f prestavi na mesto, določeno s parametrom d. Instrukcija porabi 1 urin cikel.
- **movwf** f **Mov**e **w**ork to **f**ile. Podatek iz delovnega registra w prestavi v register f. Instrukcija porabi 1 urin cikel.
- **nop** **No** **op**eration. Ne naredi ničesar. Instrukcija porabi 1 urin cikel.
- **retfie** **Ret**urn **f**rom **i**nt**e**rrupt. Vrni se iz prekinitvenega podprograma. Instrukcija porabi 2 urina cikla.
- **rlf** f **R**otate **l**eft **f**ile through carry. Vsebino registra f rotiraj v levo skozi zastavico C. Instrukcija porabi 1 urin cikel.
- **return** **Return** from sobroutine. Vrni se iz podprograma. Instrukcija porabi 2 urina cikla.
- **rrf** f **R**otate **r**ight **f**ile through carry. Vsebino registra f rotiraj v desno skozi zastavico C. Instrukcija porabi 1 urin cikel.
- **sleep** **Sleep**. V časovnik stražnega mehanizma vpiši vrednost 0 in postavi mikrokontroler v režim delovanja sleep.
- **sublw** k **Sub**tract **w**ork from **l**iteral. Vrednost delovnega registra w odštej od številske konstante k. Rezultat shrani nazaj v delovni register w. Instrukcija porabi 1 urin cikel.
- **subwf** f **Sub**tract **w**ork from **f**ile. Vrednost delovnega registra odštej od vrednosti registra f. Instrukcija porabi 1 urin cikel.
- **swapf** f **Swap** nibbles in **f**ile. Zamenjaj zgornji in spodnji polbajt v registru f. Instrukcija porabi 1 urin cikel.
- **xorlw** k E**x**clusive **OR** **l**iteral with **w**ork. Izvedi ekskluzivno operacijo ALI med vrednostjo delovnega registra in konstanto k. Rezultat te operacije shrani v delovni register. Instrukcija porabi 1 urin cikel.
- **xorwf** f E**x**clusive **OR** **w**ork with **f**ile. Izvedi ekskluzivno operacijo ALI med vrednostjo delovnega registra in vrednostjo registra f. Instrukcija porabi 1 urin cikel.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Direktive in konfiguracijski biti

Direktive so ukazi v programu, ki se ne prevedejo v strojno kodo. Prevajalniku povejo, kako naj prevaja posamezne dele programa.
Direktiva list p=16F628a na začetku programa določi tip uporabljenega mikrokontrolerja, v našem primeru PIC16F628A.
Direktiva #include <p16f628a.inc> vključi v program datoteko p16f628a.inc, ki vsebuje imena registrov v mikrokontrolerju.
Direktiva org določi naslov programskega pomnilnika, kamor se bo vpisoval program. Naslov je podan kot parameter tej direktivi.
Z direktivo end zaključimo program.

S konfiguracijskimi biti lahko izbiramo med zunanjim in notranjim oscilatorjem, vrsto oscilatorja, vlogo ponastavitvenega vezja, stabilizacijo oscilatorja pred izvajanjem programa, ko mikrokontroler priključimo na napajalno napetost, vključimo oziroma izključimo časovnik stražnega mehanizma, zaščitimo program pred kopiranjem, določimo pa lahko še nekatere druge parametre, ki jih v naših vajah ne bomo uporabljali. Vsi parametri so za posamezne tipe mikrokontrolerjev opisani v karakterističnih podatkih proizvajalca Microchipa.
Konfiguracijske bite nastavimo z določitvijo vrednosti posameznih bitov v registru CONFIG. Do tega registra, njegov specialni naslov v podatkovnem pomnilniku je 2007h, lahko dostopamo le med programiranjem.

Register CONFIG (RAM naslov: 2007h)

Bit 13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
/CP	_	_	_	_	/CPD	LVP	BOREN	MCLRE	FOSC2	/PWRTE	WDTE	FOSC1	FOSC0

Tabela 5: Register CONFIG

Nastavitev konfiguracijskih bitov za vaje:

- Bit 0 (FOSC0), bit 1 (FOSC1) in bit 4 (FOSC2) določajo izbiro oscilatorja. Na mikrokontroler bomo priključili kristalni oscilator s frekvenco 4 MHz. Priključili ga bomo med pin 15 (RA6/OSC2/CLKOUT) in pin 16 (RA7/OSC1/CLKIN). Za takšno izbiro določimo vrednost teh treh bitov: bit 0 = 1, bit 1 = 0 in bit 4 = 0.
- Bit 2 (WDTE) določa delovanje stražnega mehanizma. Vlogo tega časovnika bomo spoznali v nadaljevanju. Pri vajah bomo izklopili delovanje tega časovnika, zato bo vrednost bita 2 = 0.
- Bit 3 (/PWRTE) določa vklop ali izklop stabilizacije kristalnega oscilatorja, preden začne mikrokontroler izvajati program. Če je stabilizacija vklopljena, ta traja 72 ms. Mi bomo stabilizacijo omogočili, zato bo vrednost bita 3 = 1.
- Bit 5 (MCLRE) določa vlogo ponastavitvenega vezja, priključenega na pin 4 (RA5//MCLR/Vpp) mikrokontrolerja. Če je vrednost bita 5 = 1, omogočimo delovanje ponastavitvenega vezja. V tem primeru mikrokontroler izvaja program, ko je na njegovem pinu 5 napetost +5 V (nivo logične 1). Mikrokontroler ponastavimo, če ta pin

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

povežemo z maso (nivo logične 0). Po ponastavitvi se prične program izvajati na naslovu 000h podatkovnega pomnilnika. Če je vrednost bita 5 = 0, onemogočimo delovanje ponastavitvenega vezja, hkrati pa določimo pin RA5 kot vhodni pin mikrokontrolerja.

- Bit 6 (BOREN) omogoča delovanje posebnega vezja v mikrokontrolerju, ki poskrbi za ponastavitev programa, če napetost V_{DD} za daljši čas (72 ms) pade pod določen napetostni nivo. Tega delovanja ne bomo omogočili, zato bo vrednost bita 6 = 0.
- Bit 7 (LVP) omogoča nizko napetostno programiranje, ki se v tem tipu mikrokontrolerja ne uporablja. Če je vrednost bita 7 = 0, lahko uporabimo pin RB4 kot vhodni ali kot izhodni pin mikrokontrolerja.
- Bit 8 (/CPD) omogoča zaščito programskega pomnilnika pred kopiranjem, če je njegova vrednost enaka 0. Programskega pomnilnika ne bomo zaščitili pred kopiranjem, zato bo vrednost bita 8 = 1.
- Bit 13 (/CP) omogoča zaščito programa v pomnilniku Flash (od naslova 0000h do 07FFh) pred kopiranjem, če je njegova vrednost enaka 0. Programske kode v pomnilniku Flash ne bomo zaščitili, zato bo vrednost bita 13 = 1.
- Biti od 9 do 12 v mikrokontrolerju PIC16F628A niso podprti in se berejo kot 0.

Konfiguracijske bite lahko določimo na začetku programa, takoj za direktivo #include na dva načina. Prvi način opisuje imena konfiguracijskih bitov in njihovo stanje, drugi način pa vrednosti vseh konfiguracijskih bitov, najpogosteje zapisanih v heksadecimalnem številskem sistemu. Ker že poznamo direktive, zapišimo prve tri vrstice programa:

```
        list        p=16f628a               ; Tip mikrokontrolerja
        #include    <p16f628a.inc>          ; Vključi v program datoteko p16f628a.inc,
                                            ; ki vsebuje imena registrov v mikrokontrolerju.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC
```

Konfiguracijski biti določajo izklop zaščite pred kopiranjem, izklop stražnega mehanizma, vklop stabilizacije oscilatorja, omogočena sta ponastavljanje in uporaba pina RB4 ter uporaba zunanjega kristalnega oscilatorja.
Za opisane nastavitve je vrednost registra CONFIG zapisana v binarni obliki:

```
        __CONFIG        b'10000100101001'
```

Zapišimo še vrednost registra CONFIG v heksadecimalnem zapisu:

```
        __CONFIG        0x2129
```

Povzetek

Mikrokontrolerje PIC bomo programirali v zbirnem programskem jeziku, ki ga lahko uvrstimo tik nad strojni jezik. Vsaka od 35 instrukcij mikrokontrolerja PIC ima v zbirnem jeziku svojo okrajšavo, ki jo imenujemo mnemonik. Program v zbirnem programskem jeziku je sestavljen iz

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

stolpca instrukcij, njihovih parametrov in komentarjev. Zbirniški prevajalnik mnemonike prevede v strojno kodo ukaza, ki jo razume centralna procesna enota mikrokontrolerja CPE.
Poleg instrukcij sestavljajo program v zbirnem jeziku še direktive, ki prevajalniku povejo, kako naj prevaja posamezne dele programa. Nastavitev delovanja posameznega tipa mikrokontrolerja pa nastavimo s konfiguracijskimi biti.

Vaje

1. V registru STATUS želimo bit 5 z imenom RP0 postaviti na vrednost 1. Napiši ustrezno instrukcijo, parametra in komentar tako, da bo po izvršitvi instrukcije vrednost bita 5 v registru STATUS enaka 1.
2. Vrstica programa v zbirnem jeziku vsebuje naslednjo instrukcijo in parametra:

```
btfss       PORTB,3
```

3. Napiši komentar, iz katerega bo razviden pomen te instrukcije.
4. Napiši del programa, s katerim boš vsem osmim bitov registra TRISB določil vrednost 0.
5. Vrstica programa v zbirnem jeziku vsebuje naslednjo instrukcijo in parameter:

```
movlw       0x3F
```

6. Kakšna je vrednost posameznih bitov delovnega registra po izvršitvi te instrukcije?
7. Iz glavnega programa želimo narediti skok v podprogram z imenom Zakasnitev. Ko se podprogram Zakasnitev izvrši, se mora program izvajati naprej v glavnem programu. Napiši instrukcijo z ustreznim parametrom za skok v podprogram in za vrnitev v glavni program.
8. Kaj določa direktiva:

```
org         0x004
```

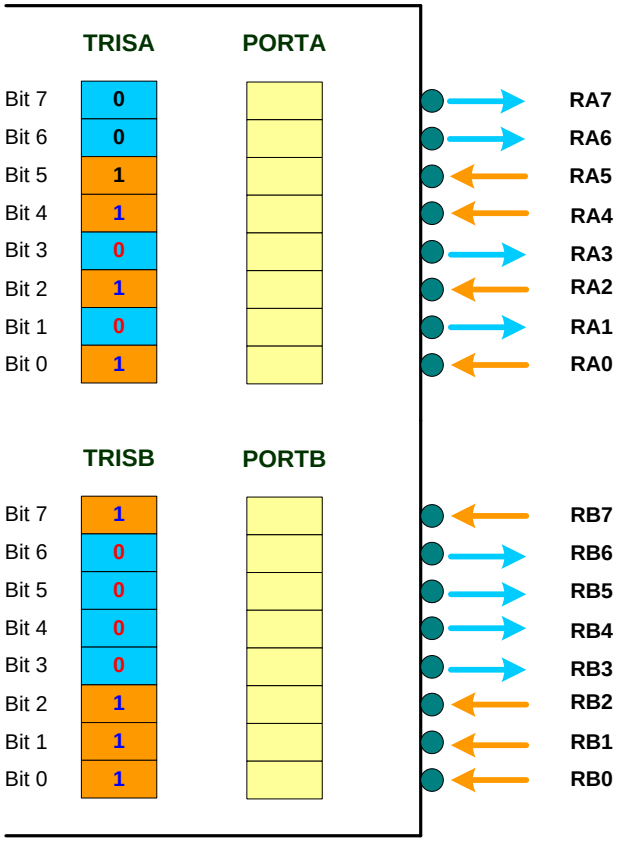
9. Na mikrokontroler PIC16F628A bomo priključili zunanji kristalni oscilator 4 MHz. Časovnik stražnega mehanizma bomo izklopili, namesto ponastavitvene tipke (reset) pa bomo uporabljali pin RA5. Vključili bomo stabilizacijo kristalnega oscilatorja, preden začne mikrokontroler izvajati program. Vsebino programskega pomnilnika želimo zaščititi pred kopiranjem. Na spletu poišči karakteristične podatke za ta tip mikrokontrolerja in na podlagi podatkov določi konfiguracijske bite, ki bodo omogočili nastavitev mikrokontrolerja po gornjih zahtevah. Konfiguracijske bite zapiši opisno, v binarni in v heksadecimalni obliki.
10. Opiši nastavitve mikrokontrolerja PIC16F628A, če je določena vrednost registra CONFIG:

```
__CONFIG 0x013B
```

11. Kakšne so vrednosti posameznih bitov registra CONFIG?

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Mikrokontrolerju PIC16F628A določimo nekatere pine kot vhodne in nekatere kot izhodne zaradi vezja, s katerim deluje. Glavna naloga mikrokontrolerja je namreč ta, da na posamezne pine pošilja napetost +5 V (1) ali napetost 0 V (0) in preverja, ali je na določenih pinih napetost +5 V (1) ali je ni (0). V odvisnosti od vezja, s katerim bo deloval, določimo ustrezne pine kot vhodne oziroma kot izhodne.



Slika 8: Določanje vhodno-izhodnih pinov mikrokontrolerja

Mikrokontroler PIC16F628A ima 16 vhodno-izhodnih pinov, ki so razdeljeni na port A in port B. Programsko so pini združeni v register PORTA in register PORTB. Smer vsakega pina (ali bo deloval kot vhodni ali kot izhodni pin) določimo z vrednostjo bitov v registru TRISA za PORTA in v registru TRISB za PORTB. Če ima določen pin v registru TRIS vrednost 0, bo deloval kot izhodni pin, če pa ima vrednost 1, bo deloval kot vhodni pin. Na sliki 8 vidimo, da so pini RA0, RA2, RA4, RA5, RB0, RB1, RB2 in RB7 določeni kot vhodni pini, drugi pa kot izhodni pini.
Od vseh 16 pinov mikrokontrolerja PIC16F628A imajo nekateri posebno vlogo.

- Pine RA0, RA1, RA2 in RA3 lahko uporabimo kot vhode analognih komparatorjev, pina RA3 in RA4 sta v tem primeru izhoda komparatorjev. Lahko pa pine RA0–RA3 uporabimo kot digitalne vhode oziroma izhode. Izbiro nastavimo z določitvijo vrednosti bitov registra CMCON (Comparator control register).
- Pin RA5 lahko uporabimo le kot digitalni vhod ali pa kot ponastavitvni vhod. V tem primeru nam služi za ponastavitev mikrokontrolerja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

- Pin RA4 lahko ob ustrezni nastavitvi mikrokontrolerja uporabimo kot števni vhod časovnika TMR0. V tem primeru deluje TMR0 kot števec impulzov.
- Na pina RA6 in RA7 priključimo zunanji kristalni oscilator, ki mikrokontrolerju določa takt, s katerim ta izvršuje instrukcije. Če pa uporabljamo notranji oscilator, lahko ta dva pina uporabljamo kot digitalna vhoda oziroma izhoda.
- Pin RB0 nam lahko ob pravilni nastavitvi mikrokontrolerja proži prekinitve vsakič, ko se na njem pojavi ustrezna sprememba signala. Prekinitve nam lahko prožijo tudi ustrezne spremembe signala na pinih RB4, RB5, RB6 in RB7.
- Pin RB3 lahko uporabimo tudi kot vhod oziroma izhod pulzno širinsko moduliranega signala (PWM).

Program za določitev vhodno-izhodnih pinov, kot je prikazano na sliki 8, je:

```
clrf    PORTA           ;Inicializacija PORTA
clrf    PORTB           ;Inicializacija PORTB
movlw   b'00000111'
movwf   CMCON           ;Omogočimo vhodno-izhodne pine.
bcf     STATUS,RP1
bsf     STATUS,RP0      ;Banka 1
movlw   b'00110101'
movwf   TRISA           ;Pini RA0, RA2, RA4 in RA5 so vhodni, drugi so izhodni.
movlw   b'10000111'
movwf   TRISB           ;Pini RB0, RB1, RB2 in RB7 so vhodni, drugi so izhodni.
```

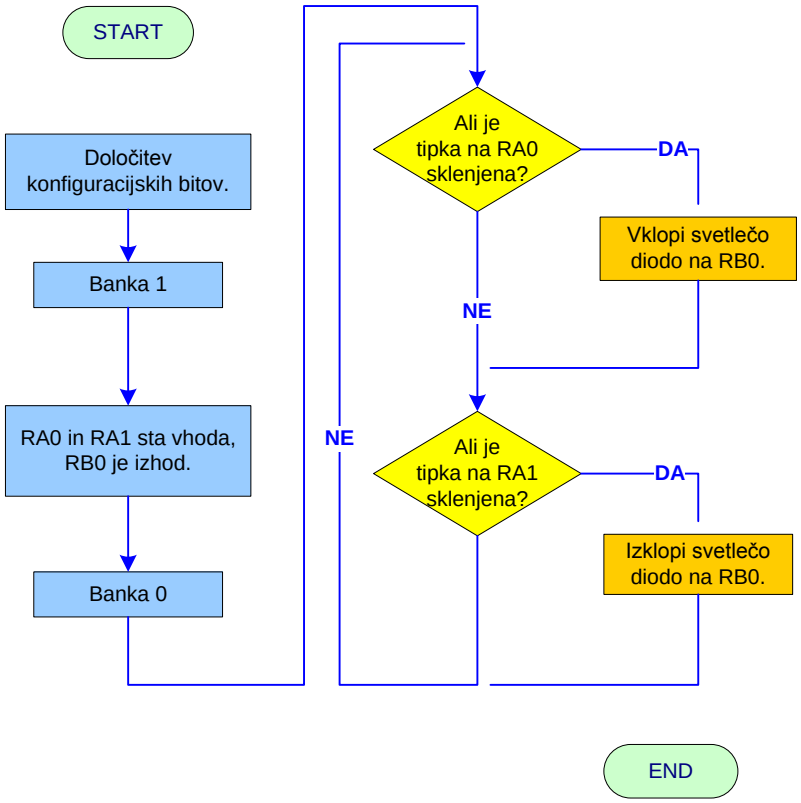
Izdelajmo sedaj prvi program v zbirnem jeziku za krmiljenje svetleče diode.

PROGRAM 1, KRMILJENJE SVETLEČE (LED) DIODE

Program naj deluje po naslednjih zahtevah.
S tipko T_1, priključeno na pin RA0, vklopimo svetlečo diodo, ki je priključena na pin RB0. Svetleča dioda sveti, dokler je s tipko T_2, priključeno na pin RA2, ne izklopimo.
Pri priključitvi tipk na pine mikrokontrolerja moramo upoštevati dejstvo, da mora biti logično stanje na uporabljenem pinu vedno določljivo. V ta namen je potreben upor, ki ohranja uporabljen priključek na nivoju logične 1 ali na nivoju logične 0. To je odvisno od tega, ali bo ob pritisku na tipko na tistem pinu, kjer je tipka priključena, stanje logične 0 ali stanje logične 1. Program bomo napisali tako, da bo pritisk na tipko zaznan kot logična 1.
Pri pritisku na tipko in pri njeni sprostitvi nastanejo na kontaktnih površinah motnje, ki lahko povzročijo težave pri branju stanja tipke. Motnje povzroča odbijanje kontaktov tipke, ki jih mikrokontroler navadno zazna kot večkratni pritisk na tipko. Te motnje lahko odpravimo na različne načine, kar bomo spoznali v nadaljevanju.
Svetlečo diodo, priključeno na izhodni pin mikrokontrolerja, moramo zaščititi z ustreznim zaščitnim uporom, saj jo lahko v nasprotnem primeru uničimo. Paziti moramo tudi, da posameznih pinov mikrokontrolerja ne preobremenimo. Obremenimo jih lahko z največ 20 mA toka v obe smeri. Svetlečo diodo bomo priključili tako, da se bo vklapljala z logično 1.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Najprej izdelamo diagram poteka. Ta predstavlja opis poteka operacij programa. Prikazuje zaporedje operacij, ki jih program pri obdelavi izvede. Je eden izmed načinov zapisa algoritma.



Slika 9: Diagram poteka za program 1

Program za krmiljenje svetleče diode s tipkama v zbirnem jeziku:

```
;-------------------------------------------------------------------------------
;Program 1, Krmiljenje svetleče diode s tipkama
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list    p=16f628a               ;Tip mikrokontrolerja
        #include <p16f628a.inc>         ;Vključi v program datoteko p16f628a.inc,
                                        ;ki vsebuje imena registrov v mikrokontrolerju.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC

        org     0x000
        goto    Glavni                  ;Nadaljuj izvajanje programa na naslovu Glavni.
        org     0x004

Glavni
        bcf     STATUS,RP0              ;Banka 0
        movlw   b'00000111'
        movwf   CMCON                   ;Omogočimo vhodno-izhodne pine na PORTA.
        bsf     STATUS,RP0              ;Banka 1
        movlw   b'00000011'
        movwf   TRISA                   ;RA0 in RA1 sta vhodna pina.
```

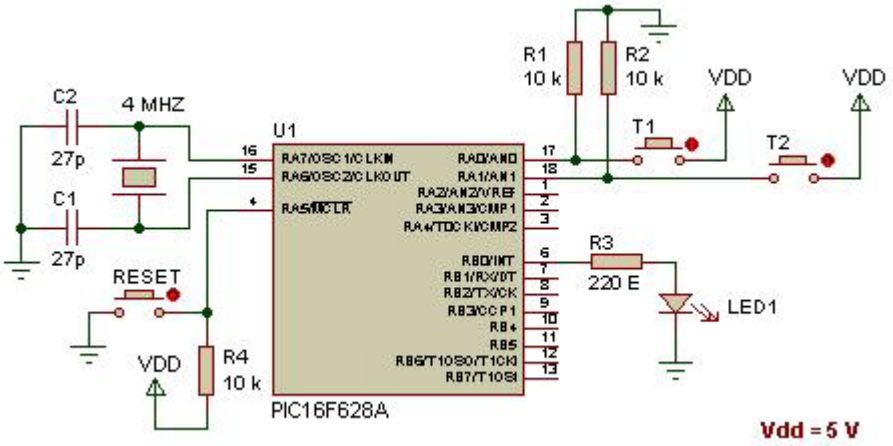
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        movlw   b'11111110'
        movwf   TRISB                   ;RB0 je izhodni pin.
        bcf     STATUS,RP0              ;Banka 0
        clrf    PORTB                   ;Inicializacija PORTB

;************** Ugotavljanje stanja tipk ********************

Ali_je_T1_sklenjena
        btfss   PORTA,0                 ;Ali je tipka T1, priključena na pin RA0, sklenjena?
        goto    Ali_je_T2_sklenjena     ;Ni sklenjena, preveri tipko T2.
        bsf     PORTB,0                 ;Je sklenjena, vklopi svetlečo diodo, priključeno na pin RB0.
Ali_je_T2_sklenjena
        btfss   PORTA,1                 ;Ali je tipka T2, priključena na pin RA1, sklenjena?
        goto    Ali_je_T1_sklenjena     ;Ni sklenjena, ponovno preveri tipko T1.
        bcf     PORTB,0                 ;Je sklenjena, izklopi svetlečo diodo, priključeno na pin RB0.
        goto    Ali_je_T1_sklenjena     ;Ponovno preveri stanje tipke T1.

        end
```

V prvih treh vrsticah programa smo izbrali tip mikrokontrolerja, vključili v program datoteko p16f628a.inc in določili konfiguracijske bite, ki bodo pri naših programih vedno enaki.
Po vklopu mikrokontrolerja in po ponastavitvi program vedno začne na naslovu 0x000. Na naslovu 0x004, to je peta lokacija programskega pomnilnika, se začne prekinitvena rutina ali prekinitveni program. Da se po vklopu ali ponastavitvi mikrokontrolerja ne začne izvajati prekinitveni program, smo pred direktivo **org** 0x004 z instrukcijo **goto** povzročili skok v programu na oznako oziroma labelo, ki ji sledi v parametru. Za zdaj prekinitev še ne bomo uporabljali, pa tudi programsko jih nismo vključili. Če bi omogočili odzivanje mikrokontrolerja na prekinitve in bi pri izvajanju glavnega programa nastala prekinitev, bi program nadaljeval v prekinitveni rutini in se nato iz prekinitvene rutine vrnil v glavni program na mesto, kjer je pred prekinitvijo končal. Instrukcija **goto** s parametrom Glavni torej pošlje izvajanje programa mimo prekinitvene rutine na naslov Glavni. S tem povzroči, da se izvajanje programa nadaljuje za labelo Glavni. Labele ali naslovi naj ne vsebujejo šumevcev in presledkov.
Instrukcija **bcf** postavi na 0 določen bit v registru, ki ga določimo s prvim parametrom. Kateri bit bomo postavili na 0, določa drugi parameter. V našem primeru smo izklopili peti bit z imenom RP0 v registru STATUS. S tem smo prešli v banko 0 podatkovnega pomnilnika RAM, kjer se nahaja register CMCON. Ker imamo sedaj dostop do tega registra, lahko prvemu, drugemu in tretjemu bitu določimo vrednost 1. S tem smo omogočili vhodno-izhodne pine na PORTA. Na pina RA0 in RA1 sta namreč priključeni tipki. Ta dva pina moramo zato določiti kot vhodna, pin RB0, kamor je priključena svetleča dioda, pa kot izhodni pin mikrokontrolerja. Vhodno-izhodne pine mikrokontrolerja določimo z vrednostjo bitov registra TRISA in TRISB. Ker se nahajata v banki 1, smo z instrukcijo **bsf** in parametroma STATUS,RP0 omogočili prehod iz banke 0 v banko 1.
Prvemu in drugemu bitu (bitu 0 in bitu 1) registra TRISA smo določili vrednost 1. S tem smo določili pina RA0 in RA1 kot vhodna pina mikrokontrolerja. Pin RB0 smo določili kot izhodni pin, tako da smo prvemu bitu (bitu 0) registra TRISB določili vrednost 0. Kako se bodo obnašali ostali pini mikrokontrolerja na PORTA in PORTB, nas ne zanima, saj nanje ne bomo priklopili nobenega elementa razen kristalnega oscilatorja in ponastavitvene tipke. S konfiguracijskimi biti smo namreč določili uporabo zunanjega kristalnega oscilatorja (pina RA6 in RA7) in možnost ponastavitve mikrokontrolerja (pin RA5).
Določitvi vloge pinov mikrokontrolerja sledi ugotavljanje stanja tipk T_1 in T_2. Stanje pina RA0 preverjamo z instrukcijo **btfss** (testiraj bit v registru, preskoči naslednjo instrukcijo, če je setiran) in parametroma PORTA,0. Ta instrukcija je vejitvena, saj ob izpolnitvi določenega pogoja

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

spremeni tok programa (glej diagram poteka). Instrukcija pregleda stanje prvega bita (bit 0) registra PORTA. Če je vrednost testiranega bita enaka 1, bo mikrokontroler preskočil eno instrukcijo in izvedel naslednjo, če pa je vrednost testiranega bita enaka 0, se bo izvajanje programa nemoteno izvajalo naprej. Če pritisnemo na tipko T_1, priključeno na pin RA0, bo na tem pinu napetost +5 V. Mikrokontroler bo to zaznal kot vrednost logične 1 na pinu. Bit 0 v registru PORTA se bo postavil na 1. Mikrokontroler bo preskočil naslednjo instrukcijo in izvedel naslednjo, **bsf** PORTB,0. Na pin RB0 bo poslal +5 V, svetleča dioda, ki je priključena na ta pin, se bo vklopila. Enako preverjamo stanje tipke T_2, priključene na pin RA1. S pritiskom na tipko T_2 torej izklopimo svetlečo diodo. Z direktivo end zaključimo program.
Pri priključenem kristalnem oscilatorju 4 MHz mikrokontroler vsakih šest µs preveri stanje posamezne tipke, če ne pritisnemo na nobeno od tipk. Instrukcija btfss namreč pri izpolnjenem pogoju porabi 2 urina cikla, pri neizpolnjenem pa enega. Ker deluje mikrokontroler s četrtino frekvence kristalnega oscilatorja, znaša en urin cikel 1 µs.



Slika 10: Priklop elementov na mikrokontroler za program 1

Na mikrokontroler priključimo elemente, kot je prikazano na sliki 10. Dokler tipka T_1 oziroma T_2 ni pritisnjena, je na vhodnem pinu RA0 oziroma RA1 potencial logične 0, saj sta preko uporov 10 kΩ povezana z maso. Ko pritisnemo na eno od tipk, ustrezni vhodni pin priklopimo na napajalno napetost 5 V. Pull-up upora R_1 in R_2 sta potrebna zato, da je stanje vhodnih pinov vedno določljivo. Svetleča dioda porabi okoli 15 mA toka, padec napetosti na njej pa je okoli 1,8 do 2 V. Zaščitni upor R3 izračunamo z enačbo:

$$R_3 = \frac{V_{dd} - U_{Led}}{15 \cdot 10^{-3}} = \frac{5\,V - 1{,}9\,V}{15\,mA} = 206\,\Omega$$

Enačba 1: Izračun zaščitnega upora za svetlečo diodo

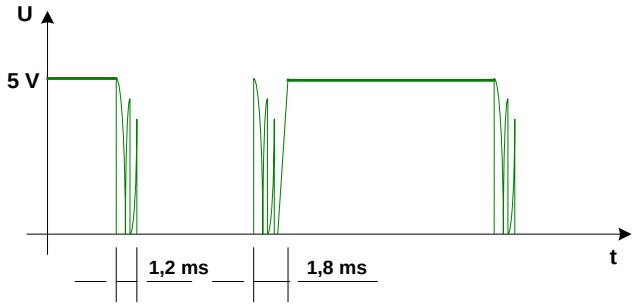
Kondenzatorja C_1 in C_2 skrbita za stabilnost kristalnega oscilatorja. Priporočena vrednost teh dveh kondenzatorjev za kristalni oscilator vrednosti 4 MHz je od 15 pF do 30 pF.
Vezje vsebuje tudi tipko za ponastavljanje (resetiranje) mikrokontrolerja. Ponastavljanje programsko omogočimo z določitvijo konfiguracijskih bitov. Mikrokontroler izvaja program, če je ponastavitveni vhod (pin 4) priključen na napajalno napetost Vdd. Zato na ta pin priključimo napajalno napetost preko upora R_4. Njegova vrednost je 10 kΩ. Isti pin preko tipke **že** mo na maso, tako da nam ta tipka služi kot tipka za ponastavitev. Po pritisku nanjo se pin 4 postavi na

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

potencial mase in s tem ponastavi mikrokontroler. Program se začne izvajati od začetka, od naslova 0x000 naprej.

Programsko odpravljanje motenj odbijanja kontaktov tipk

Pri pritisku na tipko in pri njeni sprostitvi nastanejo na kontaktih motnje, ki jih imenujemo odbijanje kontaktov. Zaradi elastičnega trka kontaktov se ti nekajkrat razklenejo in zopet sklenejo, dokler kontakta nista popolnoma sklenjena. Odbijanje kontaktov traja pri sklenitvi navadno dalj časa kot pri sprostitvi tipke. Čas odbijanja kontaktov je odvisen od kontaktnega gradiva, njegove mase, kontaktnih sil in kontaktne razdalje. Te motnje povzročajo težave pri ugotavljanju stanja tipke. Mikrokontroler te motnje zazna kot večkratni pritisk na tipko. Odbijanje kontaktov se pojavlja tudi pri vseh elektromehanskih relejih na njihovih kontaktnih sistemih.



Slika 11: Odbijanje kontaktov

Motnje zaradi odbijanja kontaktov bomo odpravili programsko. Po prvi zaznavi pritisnjene tipke počakamo določen čas, npr. 2 ms in nato še enkrat preverimo stanje tipke. Zato program za 2 ms spravimo v zanko. Po izhodu iz zanke je kontakt zanesljivo sklenjen. Program lahko v zanki čaka tudi na sprostitev tipke, saj se tudi takrat pojavijo motnje na kontaktih.
V naslednjem programu lahko vidimo, kako programsko odpravimo motnje, ki nastanejo na kontaktih pri sklenitvi tipke.

```
;-------------------------------------------------------------------------------
;Program 1 a, Krmiljenje svetleče diode s tipkama. Odprava motenj zaradi odbijanja kontaktov pri sklenitvi tipke.
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list        p=16f628a               ;Tip mikrokontrolerja
        #include    <p16f628a.inc>          ;Vključi v program datoteko p16f628a.inc,
                                            ;ki vsebuje imena registrov v mikrokontrolerju.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC

Temp1   equ     0x20                        ;Spremenljivka za zakasnitev
Temp2   equ     0x21                        ;Spremenljivka za zakasnitev

        org     0x000
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        goto    Glavni          ;Nadaljuj izvajanje programa na naslovu Glavni.
        org     0x004

Glavni
        bcf     STATUS,RP0      ;Banka 0
        movlw   b'00000111'
        movwf   CMCON           ;Omogočimo vhodno-izhodne pine na PORTA.
        bsf     STATUS,RP0      ;Banka 1
        movlw   b'00000011'
        movwf   TRISA           ;RA0 in RA1 sta vhodna pina.
        movlw   b'11111110'
        movwf   TRISB           ;RB0 je izhodni pin.
        bcf     STATUS,RP0      ;Banka 0
        clrf    PORTB           ;Inicializacija PORTB

;************* Ugotavljanje stanja tipk ********************

Ali_je_T1_sklenjena
        btfss   PORTA,0         ;Ali je tipka T1, priključena na pin RA0, sklenjena?
        goto    Ali_je_T2_sklenjena ;Ni sklenjena, preveri tipko T2.
        call    Zakasnitev      ;Je sklenjena, počakaj 2 ms (odbijanje kontaktov).
        bsf     PORTB,0         ;Vklopi svetlečo diodo, priključeno na pin RB0.
Ali_je_T2_sklenjena
        btfss   PORTA,1         ;Ali je tipka T2, priključena na pin RA1, sklenjena?
        goto    Ali_je_T1_sklenjena ;Ni sklenjena, ponovno preveri tipko T1.
        call    Zakasnitev      ;Je sklenjena, počakaj 2 ms (odbijanje kontaktov).
        bcf     PORTB,0         ;Izklopi svetlečo diodo, priključeno na pin RB0.
        goto    Ali_je_T1_sklenjena ;Ponovno preveri stanje tipke T1.

;************ Podprogram za zakasnitev, po kateri bo kontakt zanesljivo sklenjen **********

Zakasnitev
        movlw   .150            ;Prva konstanta za zakasnitev 2 ms
        movwf   Temp1           ;Prestavi konstanto iz delovnega registra v spremenljivko Temp1.
        movlw   .3              ;Druga konstanta za zakasnitev 2 ms
        movwf   Temp2           ;Prestavi konstanto iz delovnega registra v spremenljivko Temp2.
Se_zmanjsaj
        decfsz  Temp1,f         ;Zmanjšaj Temp1 za 1, preskoči naslednjo instrukcijo, če je 0.
        goto    Se_zmanjsaj     ;Temp1 še ni enak 0. Še zmanjšuj spremenljivko Temp1.
        decfsz  Temp2,f         ;Temp1 = 0. Zmanjšaj Temp2 za 1. Če je Temp2 = 0, preskoči naslednjo
                                ;instrukcijo, sicer ponovno zmanjšuj Temp1.
        goto    Se_zmanjsaj
        nop                     ;Ne stori ničesar, porabi le en urin cikel.
        return                  ;Vrni se iz podprograma.
;*******************************************************************************
        end
```

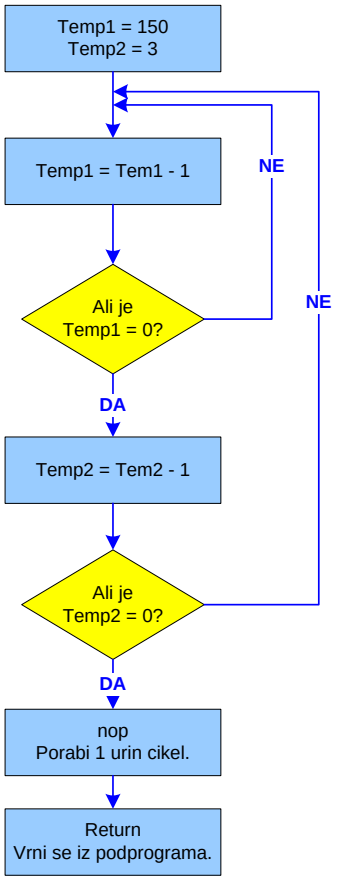
Prva novost v programu sta dve vrstici, ki vsebujeta besedo **equ**. To ni instrukcija mikrokontrolerja, temveč direktiva zbirnika, s katero lahko definiramo konstante. V programu smo konstanti Temp1 in Temp2 uporabili kot spremenljivki, saj smo jima dali za vrednost kar naslov lokacije v podatkovnem pomnilniku RAM, kamor se bodo shranjevale njune vrednosti. Pri tem pa moramo paziti, da sta naslova lokacije v področju GPR-registrov. Pri mikrokontrolerju PIC16F628A je to od naslova 0x20 do 0x6F. Kjerkoli v programu se bosta pojavili imeni Temp1 in Temp2, ju bo prevajalnik zamenjal z vrednostjo, ki smo jima ju priredili. Konstanti priredimo vrednost tako, da na začetku vrstice napišemo njeno ime, nato zapišemo direktivo equ, sledi pa ji vrednost konstante.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Konstanti Temp1 smo določili vrednost 150, konstanti Temp2 pa vrednost 3. Pika pred številko 150 in številko 3 pomeni, da je njuna vrednost desetiška. Vrednosti lahko v programu navedemo v različnih številskih sistemih, vendar moramo uporabljeni sistem sporočiti prevajalniku z ustrezno predpono. Vrednost konstante Temp1 bi lahko določili tudi z dvojiškim ali šestnajstiškim številskim sistemom:

```
movlw   .150            ;Desetiški številski sistem, lahko tudi d'150'

movlw   b'10010110'     ;Dvojiški številski sistem
movlw   0x96            ;Šestnajstiški številski sistem
```

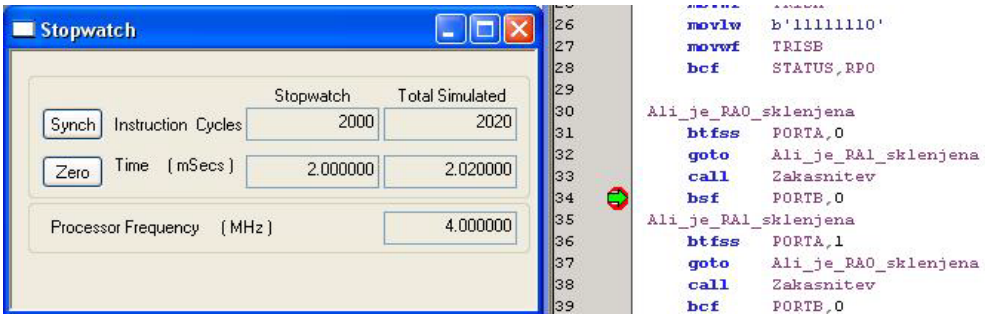
V programu navadno zapišemo vrednosti v različnih številskih sistemih, saj nam naredijo posamezne dele programa preglednejše. Če bi pri določitvi vhodno-izhodnih pinov porta B uporabili desetiški ali šestnajstiški številski sistem, na prvi pogled ne bi videli, kateri pini so določeni kot vhodi in kateri kot izhodi.
Zakasnitev 2 ms se izvede v podprogramu z labelo Zakasnitev. Podprogram moramo zaključiti z instrukcijo return, ki ne potrebuje nobenih parametrov. Podprogrami se vedno nahajajo na koncu glavnega programa. Če bi podprogram zapisali med glavnim programom, bi ga mikrokontroler izvedel tudi takrat, kadar ga ne bi poklicali, saj bi bil zanj le del glavnega programa. Da je program čitljivejši in preglednejši, podprograme ustrezno označimo s komentarji.
Instrukcija **decfsz** zmanjšuje vrednost registra, zapisanega v prvem parametru, za 1 in nato preveri, ali je vrednost registra enaka 0. Če je enaka 0, preskoči naslednjo instrukcijo, v nasprotnem primeru nadaljuje izvajanje programa. Drugi parameter je f, kar pomeni, da se bo rezultat operacije shranil nazaj v register, napisan v prvem parametru.



Slika 12: Diagram poteka za podprogram Zakasnitev

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

V podprogramu Zakasnitev določimo vrednost spremenljivkama Temp1 in Temp2. Ker smo omejeni z registri enega bajta, lahko z eno zanko napravimo največ 255 ponovitev. Najprej zmanjšujemo Temp1 za 1. Ko je enaka 0, zmanjšamo Temp2 za 1 in ponovno zmanjšujemo Temp1. Ko se vrednost spremenljivke zmanjša na 0, se začne ponovno zmanjševati od vrednosti 255. Druga zanka dvakrat ponovi prvo zanko, ta pa napravi najprej 150 ponovitev in nato še dvakrat 255 ponovitev. Skupaj torej dobimo 660 ponovitev. Ena ponovitev porabi 3 urine cikle, instrukcija decfsz enega, če ni izpolnjen pogoj, instrukcija goto pa dva. Čeprav pri podprogramu za zakasnitev motenj na kontaktih zaradi odbijanja ni potrebna točno določena zakasnitev, smo mi določili zakasnitev natančno 2 ms. Ta zakasnitev traja od trenutka, ko mikrokontroler zazna prvo sklenitev kontakta pri pritisku na tipko, do trenutka, ko se vrnemo iz podprograma na instrukcijo, ki sledi instrukciji call v glavnem programu. Da bo zakasnitev trajala natanko 2 ms, smo v podprogramu dodali instrukcijo **nop**, ki ne naredi ničesar, porabi le en urin cikel. Pri izračunu upoštevamo, da deluje mikrokontroler PIC s četrtino frekvence kristalnega oscilatorja, ki ima v našem primeru vrednost 4 MHz. En urin cikel torej traja 1 µs.



Slika 13: Meritev časa zakasnitve v okolju MPLAB

Trajanje zakasnitve lahko izmerimo v okolju MPLAB. S prekinitveno točko (breakpoint) označimo, kje naj se izvajanje programa ustavi. Nato enostavno odčitamo, kako dolgo se je program izvajal v izbranem delu programa. Vrednost izbranega oscilatorja določimo v meniju *Debugger* > *Settings…* > *Osc / Trace*, čas pa odčitamo v oknu *Stopwatch*, ki ga odpremo s klikom na *Debugger* > *Stopwatch*. V tem oknu se izpišeta število urinih ciklov in čas izvajanja izbranega dela programa. Pred meritvijo moramo števec postaviti na nič s pritiskom na gumb Zero.

Povzetek

Izdelali smo prvi program v zbirnem jeziku za mikrokontroler PIC. Zahteve delovanja smo upoštevali pri načrtovanju diagrama poteka, ki je eden izmed načinov zapisa algoritma. Diagram poteka nam je pomagal pri načrtovanju programa, ki naj bo napisan pregledno, čitljivo in pravilno, da ga lahko prevedemo v strojno kodo, ki jo razume centralna procesna enota.
Ugotovili smo, da moramo okoli mikrokontrolerja v vsakem vezju nanizati določene elemente, ki zagotavljajo njegovo pravilno delovanje. Ti elementi so oscilator, ponastavitveno (reset) vezje in napetostno napajanje. Oscilator skrbi za določanje takta, s katerim mikrokontroler izvršuje instrukcije, s ponastavitvenim vezjem pa lahko izvajanje programa kadarkoli ponastavimo.
Naučili smo se določati posamezne pine mikrokontrolerja kot vhode ali kot izhode in nanje ustrezno priključiti tipke oziroma svetlele diode. Za nemoteno delovanje krmilnega vezja smo

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

programsko odpravili motnje, ki nastanejo na kontaktih tipk. V ta namen smo glavnemu programu dodali podprogram, v katerem se izvede načrtovana časovna zakasnitev.
Z zbirniško direktivo znamo v programu definirati konstante in spremenljivke. Njihove lokacije pravilno izberemo v področju GPR-registrov.
Vrednosti registrov lahko določamo v različnih številskih sistemih. Uporabimo lahko desetiški, dvojiški ali šestnajstiški številski sistem, pred uporabljenim pa postavimo ustrezno predpono.

Vaje

1. Na mikrokontroler PIC16F628A bomo priključili tri tipke in tri svetleče diode. Tipko T_1 bomo priključili na pin RA1, tipko T_2 na pin RA2 in tipko T_3 na pin RA4. Prvo svetlečo diodo priključimo na pin RB3, drugo na pin RB5 in tretjo na pin RB7. Določi vhodno-izhodne pine mikrokontrolerja.
2. V programu smo določili vhodno-izhodne pine mikrokontrolerja PIC16F628A z določitvijo vrednosti registra TRISB:

```
movlw   0x9B        ;Prestavi podatek v delovni register.
movwf   TRISB       ;Iz delovnega registra prestavi podatek v register TRISB.
```

3. Kateri pini mikrokontrolerja so določeni kot vhodi in kateri kot izhodi?
4. Na mikrokontroler PIC priključimo zunanji kristalni oscilator, ki ima vrednost 20 MHz. S kakšnim taktom mikrokontroler izvršuje instrukcije v programu?
5. Izdelaj program, ki naj deluje po naslednjih zahtevah. S pritiskom na tipko T_1, priključeno na pin RB0, vklopimo svetleči diodi, priključeni na pina RB5 in RB6. S pritiskom na tipko T_2, priključeno na pin RB1, izklopimo svetlečo diodo, ki je priključena na pin RB6. S pritiskom na tipko T_3, priključeno na pin RB2, izklopimo drugo svetlečo diodo, ki je priključena na pin RB5. Načrtuj diagram poteka in izdelaj program v zbirnem jeziku, ki bo deloval po podanih zahtevah. V oknu *Stopwatch* izmeri čas, v katerem program preveri stanje posamezne tipke, če nobenega od njih ne sklenemo. Na mikrokontroler PIC16F628A je priključen zunanji kristalni oscilator vrednosti 4 MHz.
6. Z digitalnim osciloskopom smo izmerili čas odbijanja kontaktov uporabljene tipke, ki znaša 3 ms. Spremeni podprogram iz gradiva (Program 1 a) tako, da bo po izhodu iz podprograma kontakt zanesljivo sklenjen.
7. V kateri banki podatkovnega pomnilnika RAM se moramo nahajati, da imamo dostop do registra CMCON? Kakšna je vloga prvih treh bitov v tem registru?

ZAKASNITVE V MIKROKONTROLERJU

Pri izvajanju programov od mikrokontrolerja velikokrat zahtevamo določene časovne zakasnitve. Najenostavneje jih napravimo z zanko, ki ji določimo, kolikokrat naj se ponovi. Vsaka ponovitev zahteva določen čas, zato je skupni čas zakasnitve približno enak številu ponovitev zanke, pomnoženo s trajanjem ene ponovitve. Navadno se poleg zanke izvede še kakšna instrukcija, ki

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

vpliva na skupni čas zakasnitve. Trajanje zakasnitve najlaže izmerimo v oknu *Stopwatch* v okolju MPLAB. Če želimo določiti natančen čas zakasnitve, po potrebi dodamo v program eno ali več instrukcij nop, ki ne naredijo nič, vsaka pa porabi en urin cikel. Lahko pa na spletu poiščemo enega od programov, ki nam za zahtevane čase izpiše ustrezne podprograme. Podprograme za zakasnitev pišemo vedno na koncu glavnega programa pred direktivo end. Za določanje ponovitev v zanki potrebujemo spremenljivke, ki jih definiramo z direktivo equ zbirnika MPLAB. Naslovimo jih v področje GPR-registrov podatkovnega pomnilnika RAM. Spremenljivkam bomo dali ime Temp z zaporedno številko. V naslednjem programu lahko vidimo krmiljenje svetlečih diod na pet različnih načinov.

```
;-------------------------------------------------------------------------------
;Zakasnitve
;Program 2, Krmiljenje svetlečih diod na pet različnih načinov
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 10 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list        p=16f628a               ;Tip mikrokontrolerja
        #include    <p16f628a.inc>          ;Vključi v program datoteko p16f628a.inc,
                                            ;ki vsebuje imena registrov v mikrokontrolerju.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC

Temp1   equ     0x20                    ;Spremenljivka za zakasnitev 500 µs
Temp2   equ     0x21                    ;Spremenljivka za zakasnitev 1 ms
Temp3   equ     0x22                    ;Spremenljivka za zakasnitev 100 ms
Temp4   equ     0x23                    ;Spremenljivka za zakasnitev 500 ms
Temp5   equ     0x24                    ;Spremenljivka za zakasnitev 1 s

;********* Definiranje vzorcev utripanja, 1 => Svetleča dioda sveti, 0 => Svetleča dioda ne sveti. *********

;********* Vzorec 1 *********
P00     equ     b'11111110'
P01     equ     b'11111101'
P02     equ     b'11111011'
P03     equ     b'11110111'
P04     equ     b'11101111'
P05     equ     b'11011111'
P06     equ     b'10111111'
P07     equ     b'01111111'

;********* Vzorec 2 *********
P10     equ     b'01111111'
P11     equ     b'10111111'
P12     equ     b'11011111'
P13     equ     b'11101111'
P14     equ     b'11110111'
P15     equ     b'11111011'
P16     equ     b'11111101'
P17     equ     b'11111110'

;********* Vzorec 3 *********
P20     equ     b'01111110'
P21     equ     b'10111101'
P22     equ     b'11011011'
P23     equ     b'11100111'
P24     equ     b'11011011'
P25     equ     b'10111101'
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
P26     equ     b'01111110'

;********* Vzorec 4 *********
P30     equ     b'11111110'
P31     equ     b'11111101'
P32     equ     b'11111010'
P33     equ     b'11110101'
P34     equ     b'11101010'
P35     equ     b'11010101'
P36     equ     b'10101010'
P37     equ     b'01010101'
P38     equ     b'10101011'
P39     equ     b'01010111'
P3A     equ     b'10101111'
P3B     equ     b'01011111'
P3C     equ     b'10111111'
P3D     equ     b'01111111'

;********* Vzorec 5 *********
P40     equ     b'00000000'
P41     equ     b'11111111'
P42     equ     b'00000000'
P43     equ     b'11111111'
P44     equ     b'00000000'
P45     equ     b'11111111'
P46     equ     b'00000000'
P47     equ     b'11111111'
P48     equ     b'00000000'

        org     0x000                   ;Ponastavitveni (reset) vektor
        goto    Glavni                  ;Nadaljuj izvajanje programa na naslovu Glavni.
        org     0x004                   ;Prekinitveni vektor

;********************** Glavni program **********************

Glavni
        bcf     STATUS,RP0              ;Banka 0
        movlw   b'00000111'
        movwf   CMCON                   ;Omogočimo vhodno-izhodne pine na PORTA.
        bsf     STATUS,RP0              ;Banka 1
        movlw   b'00011111'
        movwf   TRISA                   ;RA0 do RA4 so vhodni pini.
        clrf    TRISB                   ;Vsi pini na PORTB so izhodni.
        bcf     STATUS,RP0              ;Banka 0
        movlw   0xFF
        movwf   PORTB                   ;Inicializacija PORTB, izklopimo vse svetleče diode.

;**************** Ugotavljanje stanja tipk ****************
Stanje_tipk
        btfss   PORTA,0         ;Ali je tipka na RA0 sklenjena?
        call    Vzorec1         ;Je sklenjena, pokliči Vzorec1.
        btfss   PORTA,1         ;Ali je tipka na RA1 sklenjena?
        call    Vzorec2         ;Je sklenjena, pokliči Vzorec2.
        btfss   PORTA,2         ;Ali je tipka na RA2 sklenjena?
        call    Vzorec3         ;Je sklenjena, pokliči Vzorec3.
        btfss   PORTA,3         ;Ali je tipka na RA3 sklenjena?
        call    Vzorec4         ;Je sklenjena, pokliči Vzorec4.
        btfss   PORTA,4         ;Ali je tipka na RA4 sklenjena?
        call    Vzorec5         ;Je sklenjena, pokliči Vzorec5.
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        goto    Stanje_tipk     ;Ponovno preveri stanje tipk.

;********************** Podprogrami **********************

;----------- Podprogrami za različne vzorce utripanja svetlečih diod -----------

Vzorec1
        movlw   P00             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P01             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P02             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P03             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P04             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P05             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P06             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P07             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   0xFF            ;Izklopi vse svetleče diode.
        movwf   PORTB
        call    Cas_100ms       ;Zakasnitev 100 ms
        return                  ;Vrni se iz podprograma.

Vzorec2
        movlw   P10             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P11             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P12             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P13             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P14             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P15             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P16             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P17             ;Podatek vzorca
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   0xFF            ;Izklopi vse svetleče diode.
        movwf   PORTB
        call    Cas_100ms       ;Zakasnitev 100 ms
        return                  ;Vrni se iz podprograma.

Vzorec3
        movlw   P20             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P21             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P22             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P23             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P24             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P25             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P26             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   0xFF            ;Izklopi vse svetleče diode.
        movwf   PORTB
        call    Cas_100ms       ;Zakasnitev 100 ms
        return                  ;Vrni se iz podprograma.

Vzorec4
        movlw   P30             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P31             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P32             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P33             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P34             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P35             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P36             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P37             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        movlw   P38             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P39             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P3A             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P3B             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P3C             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   P3D             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_100ms       ;Zakasnitev 100 ms
        movlw   0xFF            ;Izklopi vse svetleče diode.
        movwf   PORTB
        call    Cas_100ms       ;Zakasnitev 100 ms
        return                  ;Vrni se iz podprograma.

Vzorec5
        movlw   P40             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   P41             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   P42             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   P43             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   P44             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   P45             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   P46             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   P47             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   P48             ;Podatek vzorca
        movwf   PORTB           ;Prestavi podatek vzorca na PORTB.
        call    Cas_1s          ;Zakasnitev 1 s
        movlw   0xFF            ;Izklopi vse svetleče diode.
        movwf   PORTB
        call    Cas_1s          ;Zakasnitev 1 s
        return                  ;Vrni se iz podprograma.

;------------ Podprogrami za zakasnitve, uporabljen je oscilator 10 MHz. ------------

;----- Podprogram za zakasnitev 1 ms -----
```

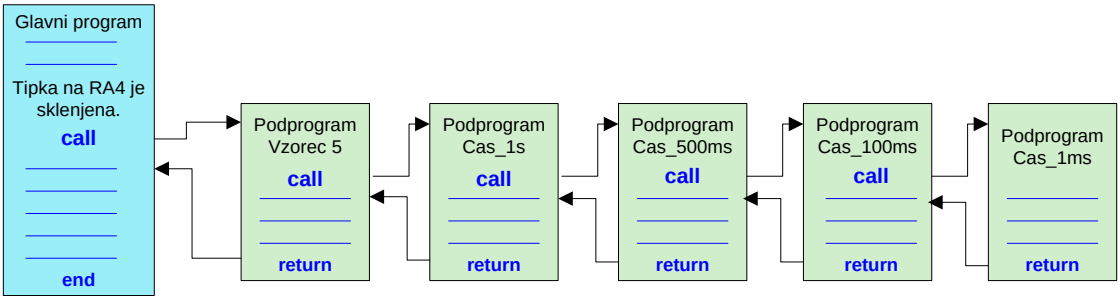
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
Cas_1ms
        movlw   .2
        movwf   Temp2           ;Temp2 = 2
zanka1
        movlw   .249
        movwf   Temp1           ;Temp1 = 249
zanka2
        nop                     ;Vsaka instrukcija nop porabi 0,4 µs v vsaki ponovitvi.
        nop                     ;Fosc = 10 MHz, 10 MHz/4 = 2,5 MHz, T = 1/2,5 MHz = 0,4 µs v eni ponovitvi
        decfsz  Temp1,f         ;Temp1 = Temp1 – 1. Podatek shrani nazaj v register Temp1.
        goto    zanka2          ;Temp1 še ni enak 0.
        decfsz  Temp2,f         ;Temp2 = Temp2 – 1. Podatek shrani nazaj v register Temp2. Temp2 = 0?
        goto    zanka1          ;Temp2 še ni enak 0.
        return                  ;Temp2 = 0, vrni se iz podprograma. Zakasnitev = 2501 x 0,4 µs = 1, 0004 ms

;----- Podprogram za zakasnitev 1 ms -----
Cas_100ms
        movlw   .100
        movwf   Temp3           ;Temp3 = 100 => 100-krat pokliče podprogram Cas_1ms.
zanka3
        call    Cas_1ms         ;Pokliči podprogram za zakasnitev 1 ms.
        decfsz  Temp3,f         ;Ali se je podprogram za zakasnitev 1 ms že 100-krat izvedel?
        goto    zanka3          ;Ne, Temp3 še ni enak 0.
        return                  ;Temp3 = 0, vrni se iz podprograma. Zakasnitev = 100 x 1,0004 ms = 100,04 ms

;----- Podprogram za zakasnitev 500 ms -----
Cas_500ms
        movlw   .5
        movwf   Temp4           ;Temp4 = 5 => 5-krat pokliče podprogram Cas_100 ms.
zanka4
        call    Cas_100ms       ;Pokliči podprogram za zakasnitev 100 ms.
        decfsz  Temp4,f         ;Ali se je podprogram za zakasnitev 100 ms že 5-krat izvedel?
        goto    zanka4          ;Ne, Temp4 še ni enak 0.
        return                  ;Temp4 = 0, vrni se iz podprograma. Zakasnitev = 5 x 100,04 ms = 500,2 ms

;----- Podprogram za zakasnitev 1 s -----
Cas_1s
        movlw   .2
        movwf   Temp5           ;Temp5 = 2 => 2-krat pokliče podprogram Cas_500ms.
zanka5
        call    Cas_500ms       ;Pokliči podprogram za zakasnitev 500 ms.
        decfsz  Temp5,f         ;Ali se je podprogram za zakasnitev 500 ms že 2-krat izvedel?
        goto    zanka5          ;Ne, Temp5 še ni enak 0.
        return                  ;Temp5 = 0, vrni se iz podprograma. Zakasnitev = 2 x 500,2 ms = 1,0004 s

        end
```

Z direktivo equ lahko ustvarimo konstanto z izbranim imenom, ki bo imela neko vrednost. Kjerkoli v programu se bo pojavilo ime te konstante, ga bo prevajalnik zamenjal z vrednostjo, ki smo jo konstanti priredili. Ime konstante mora biti na začetku vrstice. V programu smo prvih pet konstant (od Temp1 do Temp5) uporabili kot spremenljivke, saj smo jim dali za vrednost naslov lokacije podatkovnega pomnilnika RAM, kamor se bodo shranjevale njihove vrednosti. Naslednje konstante za posamezne vzorce vsebujejo vrednosti bitov PORTB. Na priključke PORTB so namreč priključene svetleče diode. Program je s tem prilagodljivejši in preglednejši. Če bi želeli katerega od vzorcev utripanja svetlečih diod spremeniti, ne bi bilo treba spreminjati vrednosti bitov po celem programu, temveč bi spremenili le vrednosti posameznih konstant, seveda pod pogojem, da ostanejo svetleče diode priključene na isti port. Vrednosti konstant so

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

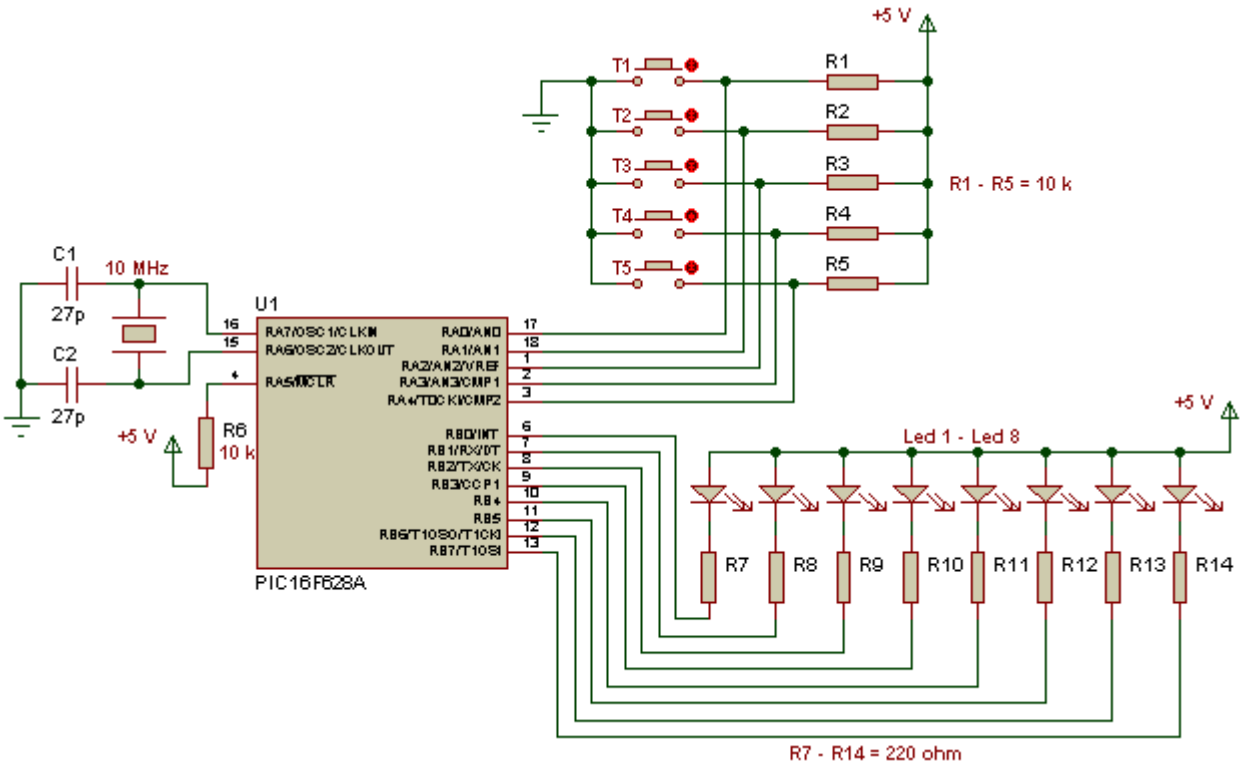
podane v dvojiškem številskem sistemu, da se posamezni vzorci vidijo na prvi pogled. Ker so svetleče diode priključene proti napajalni napetosti, bo posamezna dioda svetila takrat, ko bo na pinu, na katerega je priključena, napetost potenciala logične 0.
Po pritisku na eno od tipk program nadaljuje v podprogramu z določenim vzorcem utripanja svetlečih diod, ki smo ga določili s konstantami. Kateri vzorec se bo izvedel, je odvisno od tega, katero tipko smo sklenili. Iz podprograma nadaljuje v podprogramih za zakasnitev. Vsak vzorec ima namreč več kombinacij, med katerimi je pri prvih štirih vzorcih presledek 100 ms, pri zadnjem vzorcu pa 1 s. Podprogramov za različne časovne zakasnitve je več. Osnovni je namenjen časovni zakasnitvi 1 ms. Podprogram za časovno zakasnitev 100 ms stokrat kliče podprogram za časovno zakasnitev 1 ms, podprogram za časovno zakasnitev 500 ms petkrat kliče podprogram za časovno zakasnitev 100 ms, podprogram za časovno zakasnitev 1 s pa dvakrat kliče podprogram za časovno zakasnitev 500 ms. Podprogrami se torej izvajajo v globino, saj kličemo neki podprogram iz drugega podprograma. Paziti moramo le, da ne presežemo meje osmih podprogramov v globino. Klicanje podprogramov ob sklenitvi tipke, priključene na pin RA4, prikazuje slika 14.



Slika 14: Klicanje podprogramov v globino 5

Vhodni pini od RA0 do RA4 so preko 10 kΩ uporov, ki priključujejo ohranjajo na potencialu logične 1, povezani na napajalno napetost 5 V. Ti ohranjajo pine na potencialu logične 1, kadar tipke niso sklenjene. Ko katero od tipk sklenemo, je ustrezni pin na potencialu logične 0, na posameznem uporu pa je padec napetosti 5 V. Programsko odpravljanje motenj na kontaktih tipk, ki nastanejo zaradi odbijanja kontaktov, v tem programu ni izvedeno, saj ne vpliva na delovanje programa in izdelanega elektronskega vezja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 15: Priklop elementov na mikrokontroler za program 2 RAM – zakasnitve

Povzetek

Zakasnitev v programu najenostavneje napravimo z zanko, v kateri se program izvaja določen čas. Če hočemo izvajanje programa za določen čas poslati v zanko, moramo deklarirati spremenljivko v področju GPR-registrov. Ker smo omejeni z registri dolžine 1 bajta, se lahko v zanki z eno spremenljivko izvrši največ 255 ponovitev. Za daljše čase zakasnitev potrebujemo več spremenljivk, s katerimi izdelamo več zank. Čas zakasnitve je odvisen tudi od oscilatorja, ki določa takt, s katerim mikrokontroler izvršuje instrukcije. Za zakasnitve navadno izdelamo podprograme, ki jih kličemo po potrebi oziroma po zahtevi za zakasnitev. Če je zahtev po časovnih zakasnitvah več, izdelamo podprogram za osnovno zakasnitev, za daljše čase pa določimo, kolikokrat se bo osnovna zakasnitev ponovila.

Vaje

1. Program zakasnitve spremeni tako, da bo zakasnitev pri vzorcu 1 trajala med kombinacijami 200 ms, pri vzorcu 2 naj traja 300 ms, pri vzorcu 5 pa med kombinacijami 800 ms.
2. Vezju za zakasnitve zamenjamo oscilator 10 MHz z oscilatorjem 4 MHz. V oknu *Stopwatch* v okolju MPLAB izmeri čas trajanja vseh zakasnitev.
3. V programu zakasnitve poljubno spremeni vzorce utripanja svetlečih diod. Preizkusi delovanje na testni ploščici.

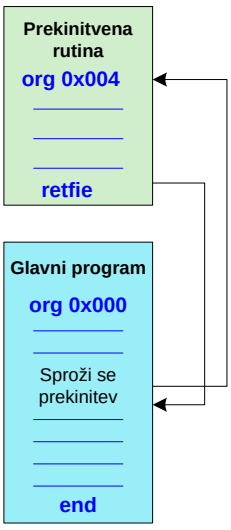
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

4. Izdelaj program, ki bo deloval po naslednjih zahtevah: mikrokontrolerju PIC16F628A priključimo kristalni oscilator vrednosti 10 MHz. S pritiskom na tipko T_1, priključeno na pin RA1, vklopimo svetlečo diodo, ki je priključena na pin RB3. Svetleča dioda naj se po 30 sekundah izklopi. Simulacijo delovanja preveri s simulatorjem MPLAB, čas pa izmeri v oknu *Stopwatch*. Izdelaj tudi diagram poteka.
5. Izdelaj opomnik za svetlobno (svetleča dioda) in zvočno (piskač) opominjanje. Opomnik vklopimo s stikalom, kadar želimo, da nas opomni po 10 minutah od časa vklopa. Opominjanje naj traja 5 sekund. Uporabi mikrokontroler PIC16F628A in kristalni oscilator 4 MHz. Vhodno-izhodne pine, na katere boš priključil potrebne elemente vezja, določi sam. Preizkusi delovanje opomnika na testni ploščici.

PREKINITVE, ČASOVNIK V MIKROKONTROLERJU

Prekinitve

Prekinitve so pomembna lastnost mikrokontrolerja. Ob prekinitvi mikrokontroler preneha izvajati glavni program in skoči v prekinitveno rutino, imenovano interrupt. Ko se ta zaključi, mikrokontroler nadaljuje izvajanje glavnega programa tam, kjer ga je pred prekinitvijo končal. Prekinitvena rutina se vedno nahaja v programskem pomnilniku na naslovu 0x004. Pisanje na naslov 0x004 dosežemo z direktivo org. Prekinitvena rutina se mora končati z instrukcijo **retfie** (return from interrupt). Da se prekinitvena rutina ali prekinitveni program ne začne izvajati tudi takrat, ko v mikrokontrolerju ni prekinitve, se mora na naslovu 0x000 nahajati instrukcija goto, ki pošlje mikrokontroler v glavni program mimo prekinitvene rutine. Kako pa lahko sprožimo prekinitev? Mikrokontroler PIC16F628A pozna 10 različnih načinov proženja prekinitev. V tem poglavju bomo spoznali štiri največkrat uporabljene načine, druge pa bomo po potrebi obravnavali pri posameznih programih.



Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Slika 16: Prikaz delovanja prekinitev

Obravnavali bomo naslednje štiri načine proženja prekinitev v mikrokontrolerju:

- pri prekoračenju vrednosti časovnikovega registra TMR0,
- pri spremembi signala na pinu RB0,
- pri spremembi signala na pinih RB4, RB5, RB6 in RB7,
- ob zaključku vpisovanja v podatkovni pomnilnik EEPROM.

Na delovanje prekinitev vpliva register INTCON (glej tabelo 4), ki se nahaja v vseh štirih bankah podatkovnega pomnilnika RAM. Odzivanje mikrokontrolerja na prekinitve omogočimo tako, da sedmi bit (GIE) tega registra postavimo na vrednost 1. Omogočimo ali onemogočimo lahko vsak vir prekinitve posebej, tako da postavimo ustrezni bit registra INTCON na vrednost 1 oziroma na vrednost 0. Če želimo vsak vir prekinitve v prekinitveni rutini obravnavati posebej, moramo vedeti, kaj je vir posamezne prekinitve oziroma zakaj je v določenem trenutku nastala prekinitev. To odčitamo iz vrednosti zastavic T0IF, INTF in RBIF. Zastavica T0IF (tretji bit v registru INTCON) se postavi na 1, če je nastala prekinitev na časovniku TMR0, zastavica INTF (drugi bit v registru INTCON) se postavi na 1, če je prekinitev nastala na pinu RB0, zastavica RBIF (prvi bit v registru INTCON) pa se postavi na 1, če se je prekinitev pojavila zaradi spremembe signala na enem izmed pinov od RB4 do RB7. Zastavica EEIF za ugotavljanje prekinitve po koncu vpisa v pomnilnik EEPROM se nahaja v SFR-registru PIR1 (osmi bit tega registra). V registru PIR1 so tudi zastavice za nekatere druge vire prekinitev, ki jih v tem poglavju ne bomo obravnavali. Ob prekinitvi moramo torej v prekinitveni rutini najprej pogledati, kje je izvor prekinitve. Ko to ugotovimo, skočimo na tisti del prekinitvene rutine, ki je določen temu izvoru. Preden pa se iz prekinitvene rutine vrnemo v glavni program z instrukcijo retfie, moramo zastavico, ki nam je sporočila vir prekinitve, postaviti na vrednost 0.
Postopek v prekinitveni rutini:

1. Izklopimo bit GIE, da se med izvajanjem prekinitvene rutine ne pojavi nova prekinitev.
2. Vrednosti registrov Work in STATUS shranimo posebej v pomnilnik RAM, ker se lahko v prekinitveni rutini spremenita.
3. Nastavimo ustrezno banko, da imamo dostop do registrov z zastavicami.
4. Ugotovimo vir prekinitve s testiranjem bitov (zastavic).
5. Napišemo ustrezen del prekinitvene rutine za posamezni vir prekinitve.
6. Ustrezni bit (zastavico), ki nam je sporočil vir prekinitve, postavimo na vrednost 0.
7. Povrnemo vrednost registrov Work in STATUS.
8. Izvedemo instrukcijo retfie, ki nas vrne v glavni program. Instrukcija retfie avtomatsko vklopi in postavi na vrednost 1 bit GIE in s tem ponovno omogoči odzivanje mikrokontrolerja na prekinitve.

Poglejmo si tako rutino:

```
;-------------------------------------------------------------------------------
;Primer prekinitvene rutine
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list        p=16f628a               ;Tip mikrokontrolerja
        #include    <p16f628a.inc>          ;Vključi v program datoteko p16f628a.inc,
                                            ;ki vsebuje imena registrov v mikrokontrolerju.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC
```

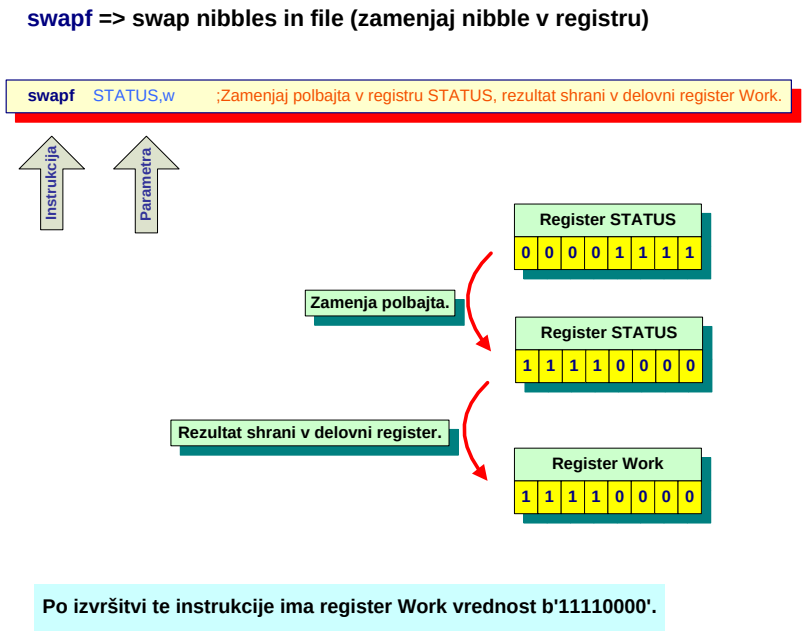
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
Delovni equ     0x20            ;V to spremenljivko bomo začasno shranili vrednost registra Work.
Status  equ     0x30            ; V to spremenljivko bomo začasno shranili vrednost registra STATUS.

        org     0x000           ;Ponastavitveni (reset) vektor
        goto    Glavni          ;Nadaljuj izvajanje programa na naslovu Glavni.
        org     0x004           ;Prekinitveni vektor

;********************** Prekinitvena rutina **********************

        bcf     INTCON,GIE      ;Onemogočimo odzivanje mikrokontrolerja na prekinitve.
        movwf   Delovni         ;Shranimo vrednost registra Work v Delovni.
        swapf   STATUS,w        ;Zamenjamo polbajta v registru STATUS, rezultat shranimo v Work.
        movwf   Status          ;Shranimo vrednost registra STATUS v Status.
        bcf     STATUS,5        ;Prehod v banko 0

    ;---------- Ugotovimo izvor prekinitve. ----------

        btfsc   INTCON,T0IF     ;Ali je prekinitev nastala na časovniku TMR0?
        goto    Timer_TMR0      ;Da, izvedi program za to prekinitev.
        btfsc   INTCON,INTF     ;Ali je prekinitev nastala zaradi spremembe signala na RB0?
        goto    RB0_pin         ;Da, izvedi program za to prekinitev.
        btfsc   INTCON,RBIF     ;Ali je prekinitev nastala zaradi spremembe signala na RB4-RB7?
        goto    RB4_RB7         ;Da, izvedi program za to prekinitev.
        btfsc   PIR1,EEIF       ;Ali je prekinitev nastala zaradi EEPROM-a?
        goto    EEPROM          ;Da, izvedi program za to prekinitev.

   ;---------- Programi za posamezne prekinitve ----------

Timer_TMR0
        ;Tukaj pišemo prekinitveni program za časovnik TMR0.
        bcf     INTCON,T0IF     ;Zastavico T0IF postavimo na 0.
        goto    Zakljuci        ;Konec prekinitvene rutine, namenjene TMR0

RB0_pin
        ;Tukaj pišemo prekinitveni program za RB0.
        bcf     INTCON,INTF     ;Zastavico INTF postavimo na 0
        goto    Zakljuci        ;Konec prekinitvene rutine, namenjene RB0

RB4_RB7
        ;Tukaj pišemo prekinitveni program za RB4-RB7.
        bcf     INTCON,RBIF     ;Zastavico RBIF postavimo na 0.
        goto    Zakljuci        ;Konec prekinitvene rutine, namenjene RB4-RB7

EEPROM
        ;Tukaj pišemo prekinitveni program za EEPROM.
        bcf     PIR1,EEIF       ;Zastavico EEIF postavimo na 0.
        goto    Zakljuci        ;Konec prekinitvene rutine, namenjene EEPROM-u

Zakljuci
        swapf   Status,w
        movwf   STATUS          ;Vrnemo vrednost registra STATUS.
        swapf   Delovni,f
        swapf   Delovni,w       ;Vrnemo vrednost registra Work.
        retfie                  ;Izhod iz prekinitvene rutine

;********************** Glavni program **********************

Glavni
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
;Tukaj se nahaja glavni program.

    end
```

Če v programu uporabljamo le en vir prekinitve, ga v prekinitveni rutini ni treba ugotavljati. Moramo pa zbrisati zastavico, saj tega mikrokontroler ne naredi sam. V nasprotnem primeru bo mikrokontroler neprenehoma klical prekinitveno rutino.
Nova instrukcija v programu je **swapf**, ki zamenja polbajte v registru, zapisanem v prvem parametru, in jih shrani na mesto, ki ga določimo z drugim parametrom. Če je drugi parameter f, to pomeni, da se bo rezultat shranil v isti register.



Slika 17: Prikaz delovanja instrukcije swapf

Polbajt (nibble) je računalniški pojem za skupino 4 bitov. En bajt ima torej dva polbajta, zgornjega in spodnjega. V programu smo instrukcijo swapf uporabili za shranitev registrov STATUS in Work, ker ne spreminja nobenih zastavic. Register STATUS vsebuje tri zastavice, ki se postavijo glede na rezultat zadnje operacije. Da se zastavice ne spremenijo, register STATUS shranjujemo z instrukcijo swapf, saj ta nikoli ne spreminja nobene zastavice. Tudi delovni register Work shranjujemo zato, da se njegova vrednost v prekinitveni rutini ne spremeni. Če se namreč prekinitev zgodi ravno takrat, ko želimo v glavnem programu uporabiti vrednost delovnega registra in v prekinitveni rutini njegovo vrednost spremenimo, se bo po vrnitvi v glavni program v njem nahajala napačna vrednost.
Ob pravilni nastavitvi mikrokontrolerja PIC16F628A, bo pin RB0 sprožil prekinitev ob vsaki ustrezni spremembi signala. Prekinitev lahko sproži ob vsakem prehodu signala z 0 na 1 ali nasprotno. Kakšen prehod signala bo sprožil prekinitev, določimo z bitom INTEDG v registru OPTION (tabela 2). Če bomo prožili prekinitve s pinom RB0, ga moramo določiti kot vhodni pin mikrokontrolerja. Prekinitve lahko prožimo tudi s spremembo signala na kateremkoli pinu od RB4 do RB7. Tudi te pine moramo v tem primeru določiti kot vhode. Za odzivanje mikrokontrolerja na prekinitve pri spremembi signala na pinu RB0 moramo v registru INCTON postaviti na 1 bita GIE in INTE, za odzivanje na pinih od RB4 do RB7 pa bita GIE in RBIE.
Ker bomo prekinitve ob koncu vpisa v pomnilnik EEPROM in prekinitve na časovniku obravnavali posebej, si sedaj oglejmo primer naslednjega programa:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
;-------------------------------------------------------------------------------
;Prekinitve na RB0 in RB5
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list      p=16f628a             ;Tip mikrokontrolerja
        #include  <p16f628a.inc>        ;Vključi v program datoteko p16f628a.inc.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC

Delovni equ     0x20                    ;V to spremenljivko bomo začasno shranili vrednost registra Work.
Status   equ    0x30                    ;V to spremenljivko bomo začasno shranili vrednost registra STATUS.
Temp1   equ     0x22
Temp2   equ     0x25
Temp3   equ     0x27                    ;Spremenljivke za zakasnitev 50 s

        org     0x000                   ;Ponastavitveni vektor
        goto    Glavni                  ;Nadaljuj na naslovu Glavni.
        org     0x004                   ;Prekinitveni vektor

;********************** Prekinitvena rutina **********************

        bcf     INTCON,GIE              ;Onemogočimo odzivanje mikrokontrolerja na prekinitve.
        movwf   Delovni                 ;Shranimo vrednost registra Work v Delovni.
        swapf   STATUS,w                ;Zamenjamo polbajta v registru STATUS, rezultat shranimo v Work.
        movwf   Status                  ;Shranimo vrednost registra STATUS v Status.
        bcf     STATUS,5                ;Prehod v banko 0

    ;---------- Ugotovimo izvor prekinitve. ----------

        btfsc   INTCON,INTF             ;Ali je prekinitev nastala zaradi spremembe signala na RB0?
        goto    RB0_pin                 ;Da, izvedi program za to prekinitev.
        btfsc   INTCON,RBIF             ;Ali je prekinitev nastala zaradi spremembe signala na RB4-RB7?
        goto    RB5_pin                 ;Da, izvedi program za to prekinitev.

   ;---------- Programi za posamezne prekinitve ----------

RB0_pin
        bsf     PORTB,1                 ;Vklopi razsvetljavo, priključeno na pin RB1.
        call    Zakasnitev              ;Pokliči podprogram za zakasnitev.
        bcf     PORTB,1                 ;Izklopi razsvetljavo, na pin RB1 pošlji nivo logične 0.
        bcf     INTCON,INTF             ;Zastavico INTF postavimo na 0.
        goto    Zakljuci                ;Konec prekinitvene rutine, namenjene prekinitvi na RB0

RB5_pin
        movlw   b'00000100'             ;Maska za preklapljanje svetleče diode na pinu RB2
        xorwf   PORTB,f                 ;Operacija XOR med masko in PORTB
        bcf     INTCON,RBIF             ;Zastavico RBIF postavimo na 0.
        goto    Zakljuci                ;Konec prekinitvene rutine, namenjene prekinitvi na RB5

Zakljuci
        swapf   Status,w
        movwf   STATUS                  ;Vrnemo vrednost registra STATUS.
        swapf   Delovni,f
        swapf   Delovni,w               ;Vrnemo vrednost registra Work.
        retfie                          ;Izhod iz prekinitvene rutine
```

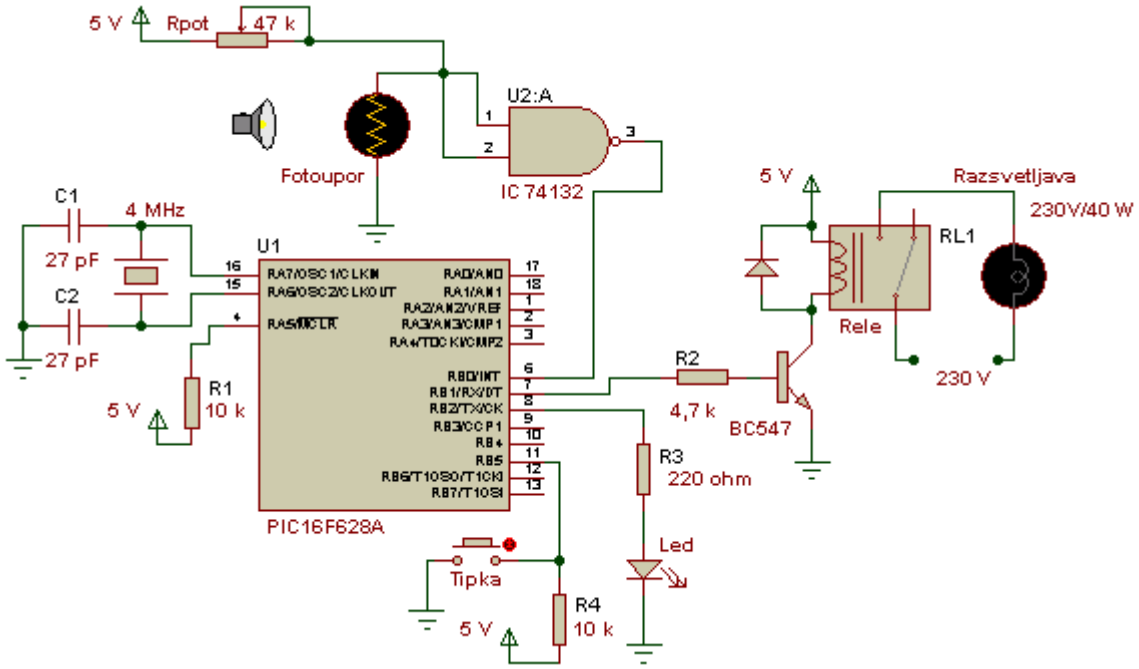
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
;********************** Glavni program **********************

Glavni
        bcf     STATUS,RP0              ;Banka 0
        movlw   b'00000111'
        movwf   CMCON                   ;Omogočimo vhodno-izhodne pine na PORTA.
        bsf     STATUS,RP0              ;Banka 1
        clrf    TRISA                   ;Vsi pini PORTA so izhodni.
        movlw   b'00100001'
        movwf   TRISB                   ;RB0 in RB5 sta vhoda.
        bcf     STATUS,RP0              ;Banka 0
        clrf    PORTB                   ;Inicializacija PORTB
        movlw   b'10011000'
        movwf   INTCON                  ;Omogočene so prekinitve na RB0 in RB5.
        clrf    PORTB                   ;Inicializacija porta B
Zanka
        goto    Zanka           ;Program je v zanki, pri spremembi na RB0 in RB5 se bo sprožila prekinitev.

;************** Podprogram za zakasnitev 50 sekund **********************

Zakasnitev
        movlw   .100                    ;Prva konstanta za zakasnitev 50 s
        movwf   Temp1                   ;Prestavi konstanto iz delovnega registra v spremenljivko Temp1.
        movlw   .167                    ;Druga konstanta za zakasnitev 50 s
        movwf   Temp2                   ;Prestavi konstanto iz delovnega registra v spremenljivko Temp2.
        movlw   .254                    ;Tretja konstanta za zakasnitev 50 s
        movwf   Temp3                   ;Prestavi konstanto iz delovnega registra v spremenljivko Temp3.
Se_zmanjsaj
        decfsz  Temp1,f                 ;Zmanjšaj Temp1 za 1, preskoči naslednjo instrukcijo, če je 0.
        goto    Se_zmanjsaj             ;Temp1 še ni enak 0.
        decfsz  Temp2,f                 ;Temp1 = 0. Zmanjšuj Temp2 za 1.
        goto    Se_zmanjsaj
        decfsz  Temp3,f                 ;Temp2 = 0. Zmanjšuj Temp3 za 1.
        goto    Se_zmanjsaj
        nop                             ;Ne stori ničesar, porabi le en urin cikel.
        return                          ;Vrni se iz podprograma.

        end
```

Prekinitve v programu prožita spremembi signala na pinu RB0 in RB5. Prekinitev na pinu RB0 se bo sprožila pri prehodu signala iz visokega nivoja v nizkega. Registra OPTION namreč nismo vključili v program, zato imajo vsi njegovi biti, med njimi bit INTEDG, vrednost 0. V glavnem programu smo določili vrednost registra INTCON tako, da se lahko mikrokontroler odziva na opisane prekinitve. Zatem smo program spravili v zanko, saj ne uporabljamo nobenih dodatnih vhodnih pinov. Kljub temu da je program v zanki, delujejo prekinitve v ozadju. Ob prekinitvi program skoči v prekinitveno rutino, kjer poleg obveznih vrstic programa, ki smo jih že opisali, preverja vzrok prekinitve.
Če je prekinitev nastala na pinu RB0, program vklopi pin RB1 in takoj zatem pokliče podprogram za 50-sekundno zakasnitev. Po vrnitvi iz podprograma izklopi pin RB1 in zatem še indikator prekinitve, zastavico INTF. Na pin RB0 bomo priklopili vezje s svetlobnim senzorjem. Ko bo senzor zaznal temo, se bo vklopila razsvetljava, ki jo bomo preko releja priključili na pin RB1. Po 50 sekundah se bo razsvetljava izklopila.
Vsaka prekinitev, ki jo sproži sprememba signala iz nizkega nivoja v visokega ali nasprotno na pinu RB5, preklopi svetlečo diodo, priključeno na pin RB2. Pri prvi prekinitvi jo vklopi, pri drugi prekinitvi na pinu RB5 se svetleča dioda izklopi. V ta namen v prekinitveni rutini uporabimo novo instrukcijo **xorwf**. Ta instrukcija izvrši operacijo xor nad enakoležečimi biti

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

med delovnim registrom in registrom, zapisanim v prvem parametru. Drugi parameter določi, kam se bo rezultat operacije shranil. V našem primeru se bo shranil nazaj v isti register. Operacija xor se bo torej vsakič izvršila med registrom PORTB in delovnim registrom, ki smo mu določili vrednost s tako imenovano masko.

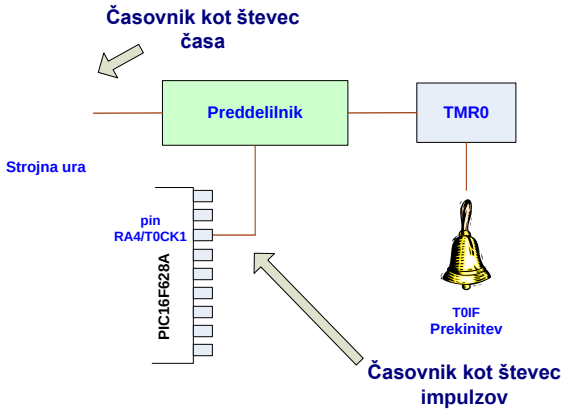


Slika 18: Priklop elementov na mikrokontroler, prekinitve na RB0 in RB5

Časovnik v mikrokontrolerju

Mikrokontroler PIC16F628A vsebuje več časovnikov: TMR0, TMR1, TMR2 in časovnik stražnega mehanizma. Njihove vloge bomo spoznali pri posameznih programih v nadaljevanju. Najprej spoznajmo časovnik 0, ki ga predstavlja SFR-register z imenom TMR0. Najdemo ga v banki 0 in banki 2 podatkovnega pomnilnika RAM na naslovih 01h in 101h, njegova vrednost pa se iz ene banke preliva v drugo. Je 8-bitni register, katerega vrednost se poveča za 1 ob vsakem izpolnjenem pogoju proženja. Ko doseže maksimalno vrednost 255 (FFh), se vrne na 0 (00h) in se povečuje naprej. Pravimo, da je časovnik prekoračil svojo vrednost ali s tujko, da je nastal overflow na časovniku. Ob prekoračitvi svoje vrednosti časovnik TMR0 sproži prekinitev, če smo jo seveda omogočili. Časovnikova vrednost se poveča vsakič, ko mine en strojni cikel (en strojni cikel traja četrtino frekvence oscilatorja), ali pa vsakič, ko se na pinu RA4 spremeni signal. V prvem primeru časovnik deluje kot števec časa, saj se povečuje v točno določenih časovnih intervalih, v drugem primeru pa deluje kot števec impulzov, ki se pojavijo na pinu RA4. Mikrokontroler PIC16F628A vsebuje tudi preddelilnik, ki deli prožilne impulze, namenjene časovniku.

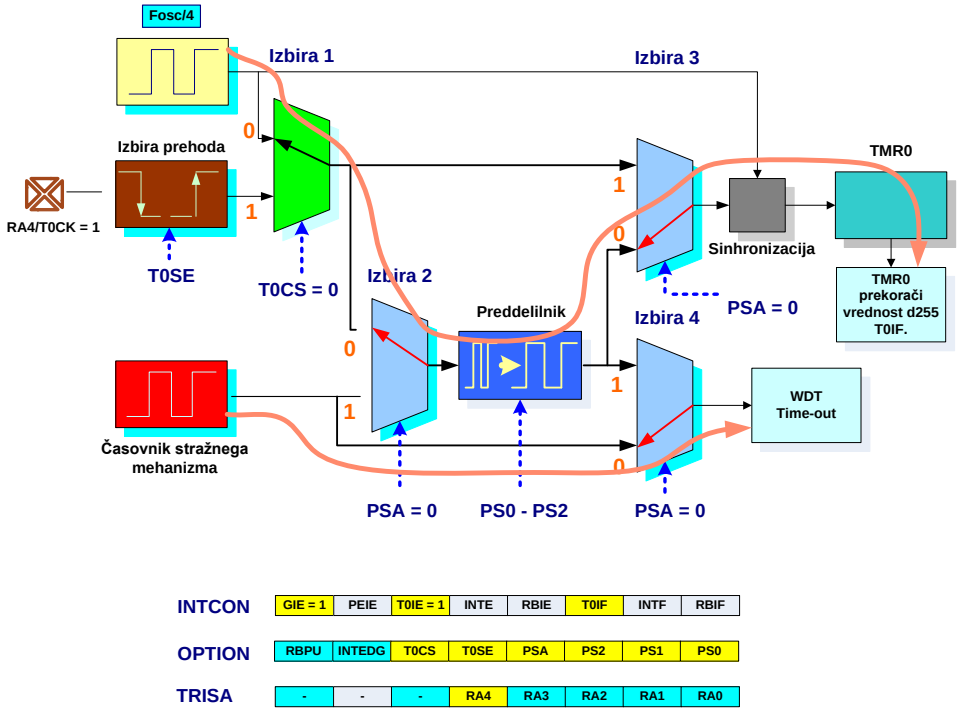
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 19: Shema delovanja časovnika TMR0

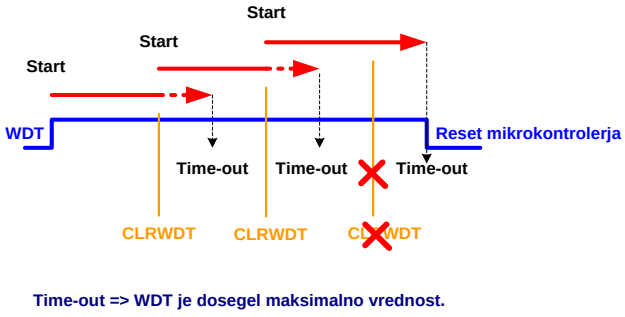
S preddelilnikom lahko dosežemo, da se časovnik TMR0 poveča za 1 vsakih 64 strojnih ciklov ali po vsakih 64 impulzih na pinu RA4, če ga uporabljamo kot števec impulzov. Preddelilnik je torej priročen, kadar želimo s časovnikom meriti daljše časovne intervale. Brez preddelilnika bi na časovniku nastala prekoračitev (overflow) pri 4 MHz oscilatorju v času 255 µs. S prvimi tremi biti registra OPTION (glej tabelo 3) lahko nastavimo osem različnih stopenj preddelitve. Ali bo časovnik deloval kot števec časa ali kot števec impulzov, določimo z vrednostjo bita T0CS, to je šesti bit registra OPTION. Če ga uporabljamo kot merilnik impulzov, lahko s petim bitom istega registra (bit T0SE) izbiramo, ali se bo povečeval pri prehodu iz nizkega nivoja v visokega ali pri prehodu iz visokega nivoja signala na pinu RA4 v nizkega.
V register TMR0 lahko poljubno vpisujemo podatke in tako določimo točen čas, kdaj bo nastala prekinitev, če ga prožimo s taktom strojne ure. Za odzivanje mikrokontrolerja na časovnikove prekinitve moramo vklopiti bit GIE in bit T0IE v registru INTCON. Če vpišemo neko vrednost v register TMR0, se začne njegova vrednost povečevati čez čas dveh strojnih ciklov. Za zakasnitev 200 µs (pri oscilatorju 4 MHz) bi morali v register TMR0 vpisati vrednost 57, saj bo za to zakasnitev potrebnih poleg začetnih dveh korakov še 198 korakov, da bo presegel vrednost 255.
Slika 20 prikazuje delovanje časovnika TMR0. Na njegovo delovanje vplivajo nastavitve registrov OPTION, INCTON in TRISA. Vpliv prvih dveh smo že opisali, v registru TRISA pa moramo določiti pin RA4 kot vhod, če hočemo časovnik TMR0 uporabiti kot števec impulzov. Na prikazanem primeru vidimo, da je preddelilnik dodeljen časovniku TMR0 (bit PSA ima vrednost 0), ki deluje kot števec časa (bit T0CS ima vrednost 0). Sinhronizacija traja dva strojna cikla, če v register vpišemo neko vrednost. Ko časovnikov register TMR0 prekorači vrednost 255 (overflow), se sproži prekinitev. Indikator prekinitve, zastavica T0IF, se postavi na vrednost 1. Na sliki je prikazan tudi časovnik stražnega mehanizma, ki poskrbi, da napaka v programu ne ustavi izvajanja mikrokontrolerja.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 20: Primer delovanja časovnika TMR0

Časovnik stražnega mehanizma (WDT) ima svoj notranji RC-oscilator, ki skrbi za neprestano povečevanje njegove vrednosti. Zunanji kristalni oscilator ne vpliva na njegovo delovanje. Ko doseže maksimalno vrednost (time-out), povzroči ponastavitev (reset) mikrokontrolerja in program se začne izvajati od začetka. Če je mikrokontroler v stanju mirovanja (sleep mode) ali je v programu napaka, stražni mehanizem poskrbi, da mikrokontroler ne ustavi izvajanja programa. Brez uporabe preddelilnika doseže maksimalno vrednost v času 18 ms, z uporabo največjega preddelilnega faktorja pa znaša ta čas več kot 2,3 s.



Slika 21: Prikaz delovanja časovnika stražnega mehanizma

Če ga z nastavitvijo konfiguracijskih bitov nismo izklopili, moramo v programu njegov register redno počistiti, njegovo vrednost spraviti na 0, preden doseže časovno omejitev (time-out). To dosežemo z instrukcijo clrwdt, ki ne potrebuje nobenega parametra. Časovnika stražnega mehanizma v naših programih ne uporabljamo, zato ga izklopimo pri določitvi konfiguracijskih bitov (__CONFIG _WDT_OFF).
Naslednji program uporablja prekinitev na časovniku TMR0. Prekinitve se prožijo s sklenitvijo tipke T_1, priključene na pin RA0. Na pinih RB0, RB2, RB4 in RB7 so priključene svetleče diode.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Če je tipka T_1 sklenjena, svetleče diode utripajo. PORTB inicializiramo s tipko T_2, priključeno na pin RA1.

```
;-------------------------------------------------------------------------------
;Prekinitve na časovniku TMR0
;Prekinitve na časovniku TMR0 prožimo s pritiskom na tipko, priključeno na pin RA0.
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list    p=16f628a               ;Tip mikrokontrolerja
        #include <p16f628a.inc>         ;Vključi v program datoteko p16f628a.inc.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC

Delovni equ     0x20                    ;V to spremenljivko bomo začasno shranili vrednost registra Work.
Status  equ     0x30                    ;V to spremenljivko bomo začasno shranili vrednost registra STATUS.
Temp1   equ     0x22
Temp2   equ     0x25
Temp3   equ     0x27                    ;Spremenljivke za zakasnitev

        org     0x000                   ;Ponastavitveni (reset) vektor
        goto    Glavni                  ;Nadaljuj na naslovu Glavni.
        org     0x004                   ;Prekinitveni vektor

;********************** Prekinitvena rutina **********************

        bcf     INTCON,GIE              ;Onemogočimo odzivanje mikrokontrolerja na prekinitve.
        movwf   Delovni                 ;Shranimo vrednost registra Work v Delovni.
        swapf   STATUS,w                ;Zamenjamo polbajta v registru STATUS, rezultat shranimo v Work.
        movwf   Status                  ;Shranimo vrednost registra STATUS v Status.
        bcf     STATUS,5                ;Prehod v banko 0
Vklopi_utripanje
        movlw   b'01010101'
        xorwf   PORTB,f                 ;Preklapljanje svetlečih diod na pinih RB0, RB2, RB4 in RB6
Zakasnitev
        movlw   .221
        movwf   Temp1
        movlw   .133
        movwf   Temp2
        movlw   .3
        movwf   Temp3
Se
        decfsz  Temp1
        goto    Se
        decfsz  Temp2
        goto    Se
        decfsz  Temp3
        goto    Se                      ;Zanke za zakasnitev med preklopom svetlečih diod
        bcf     INTCON,T0IF             ;Indikator  prekinitve na TMR0 postavimo na 0.

        swapf   Status,w
        movwf   STATUS                  ;Vrnemo vrednost registra STATUS.
        swapf   Delovni,f
        swapf   Delovni,w               ;Vrnemo vrednost registra Work
        retfie                          ;Izhod iz prekinitvene rutine

;********************** Glavni program **********************
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
Glavni
        bcf     STATUS,RP0          ;Banka 0
        movlw   b'00000111'
        movwf   CMCON               ;Omogočimo vhodno-izhodne pine na PORTA.
        bsf     STATUS,RP0          ;Banka 1
        clrf    TRISB               ;Vsi pini PORTB so izhodni.
        movlw   b'00000011'
        movwf   TRISA               ;RA0 in RA1 sta vhoda.
        movlw   b'10000011'
        movwf   OPTION_REG          ;Preddelilnik dodeljen časovniku TMR0, preddelitev 1 : 16
        bcf     STATUS,RP0          ;Banka 0
        clrf    PORTB               ;Inicializacija PORTB
        movlw   b'10100000'
        movwf   INTCON              ;Omogočene so prekinitve na časovniku TMR0.
        clrf    PORTB               ;Inicializacija porta B

;********** Branje tipk  **********

Ali_je_RA0_sklenjena
        btfsc   PORTA,0
        goto    Ali_je_RA0_sklenjena ;Tipka na RA0 je sklenjena, proži prekinitve na časovniku TMR0.
        clrf    TMR0                ;Tipka na RA0 ni sklenjena, onemogoči prekoračitev na TMR0.
        goto    Ali_je_RA1_sklenjena
Ali_je_RA1_sklenjena
        btfsc   PORTA,1
        goto    Ali_je_RA0_sklenjena
        clrf    PORTB               ;Tipka na RA1 je sklenjena, izklopi vse svetleče diode.
        goto    Ali_je_RA0_sklenjena

        end
```

Odzivanje mikrokontrolerja na prekinitve ob prekoračenju vrednosti časovnikovega registra TMR0 smo omogočili z določitvijo vrednosti registra INTCON v glavnem programu, vklopili smo njegov osmi (GIE) in šesti bit (T0IE). Časovniku smo v registru OPTION določili preddelitev 1 : 16, zato se njegova vrednost poveča za 1 po vsakem šestnajstem strojnem ciklu. Dokler tipka T_1 ni sklenjena, časovnik ne doseže vrednosti 255, saj se njegov register po instrukciji clrf in parametru TMR0 med programom za branje tipk programsko postavi na 0. Ob sklenitvi te tipke pa se ta instrukcija ne izvede, zato časovnik nemoteno povečuje svojo vrednost. Preklapljanje svetlečih diod na PORTB smo dosegli z instrukcijo xor med masko, ki smo jo vpisali v delovni register in PORTB. Delovanje prikazuje slika 22. Program za zakasnitev med preklapljanjem svetlečih diod smo zapisali kar v prekinitveni rutini. Frekvenca preklapljanja je odvisna od te zakasnitve in od števila strojnih ciklov, ki se porabijo za izvršitev nekaterih instrukcij, odvisno od tega, kje se je izvajal program, ko smo sklenili tipko T_1. Da svetleče diode ne bi svetile, ko sprostimo tipko T_1, PORTB inicializiramo s tipko T_2. Namesto tipke T_1 lahko uporabimo ustrezni senzor, na primer senzor gibanja. Ko zazna gibanje, se kot indikator vklopi utripanje svetlečih diod. Paziti moramo le na to, da je na vhodnem pinu ustrezni napetostni nivo pri stanju logične 0 in pri stanju logične 1.
Pred izhodom iz prekinitvene rutine ne smemo pozabiti izbrisati indikatorja prekinitve, zastavice T0IF.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Prvič v prekinitveni rutini:

	0 1 0 1 0 1 0 1	**Vrednost delovnega registra, maska.**
xor	**0 0 0 0 0 0 0 0**	**Vrednost registra PORTB, svetleče diode ne svetijo.**
	0 1 0 1 0 1 0 1	**Rezultat operacije xor med w in PORTB. Rezultat shrani v register PORTB, svetleče diode svetijo.**

Drugič v prekinitveni rutini:

	0 1 0 1 0 1 0 1	**Vrednost delovnega registra, maska.**
xor	**0 1 0 1 0 1 0 1**	**Vrednost registra PORTB, svetleče diode svetijo.**
	0 0 0 0 0 0 0 0	**Rezultat operacije xor med w in PORTB. Rezultat shrani v register PORTB, svetleče diode ne svetijo.**

Slika 22: Uporaba operacije xor za preklapljanje svetlečih diod

Povzetek

Spoznali smo odzivanje mikrokontrolerja na prekinitve, ki jih pri programiranju zelo pogosto uporabljamo. Ob prekinitvi mikrokontroler iz glavnega programa skoči v prekinitveno rutino. Ko jo obdela, nadaljuje izvajanje glavnega programa tam, kjer je prek prekinitvijo končal. Mikrokontroler PIC16F628A pozna 10 načinov proženja prekinitev. Vse načine lahko poljubno vklapljamo in jim določamo parametre. Vir prekinitve preprosto ugotovimo iz vrednosti zastavic. V prekinitveni rutini moramo poskrbeti za to, da se vrednosti registrov STATUS in Work ne spremenita.
Časovnik TMR0 lahko v mikrokontrolerju uporabimo kot števec časa ali kot števec impulzov. Njegova vrednost se poveča za 1 vsakič, ko mine en strojni cikel, če ga uporabljamo kot števec časa, oziroma pri vsaki spremembi signala na pinu RA4, če ga uporabljamo kot števec impulzov. Ko doseže maksimalno vrednost 255, se vrne na 0 in šteje dalje, ob tem pa sproži prekinitev, če smo jo programsko omogočili.
Stražni mehanizem deluje neodvisno od zunanjega kristalnega oscilatorja, saj ga proži notranji RC-oscilator. Ko doseže maksimalno vrednost, povzroči ponastavitev mikrokontrolerja, zato moramo v programu vrednost njegovega registra redno in pravočasno postavljati na nič. V naših programih časovnika stražnega mehanizma ne bomo uporabljali, zato ga pri nastavitvi konfiguracijskih bitov izklopimo.

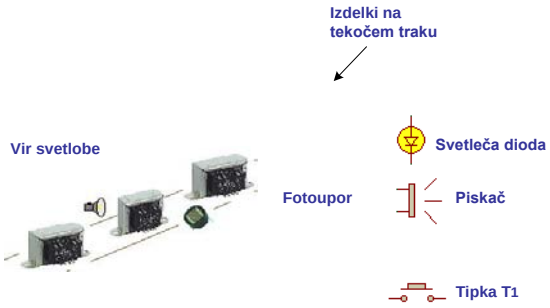
Vaje

1. Določi vrednost registra OPTION za naslednje zahteve. Preddelilnik je dodeljen časovniku TMR0, preddelitev je 1 : 64, časovnik TMR0 se uporablja kot števec impulzov, njegova vrednost naj se povečuje pri prehodu signalov na RA4 z visokega logičnega nivoja na nizkega, pull-up upori so vključeni.
2. Mikrokontrolerju PIC16F628A želimo omogočiti odzivanje na prekinitve. Prekinitev naj se sproži, ko časovnik TMR0 preseže svojo vrednost, in vsakič, ko se spremeni signal na

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

pinu RB7. Kakšna mora biti vrednost registra INTCON? Kateri zastavici nam sporočata vir prekinitve?

3. Napiši program v zbirnem jeziku za mikrokontroler PIC16F628A, ko bo po vsakem pritisku na tipko, priklopljeno na pin RB0, vklopil svetlečo diodo na pinu RA2 za 5 sekund. Po tem času se mora svetleča dioda izklopiti. Pritisnjena tipka pomeni logično stanje 0 na pinu RB0. V oknu *Stopwatch* natančno izmeri čas vklopa svetleče diode in število strojnih ciklov, izvedenih v tem času. Program prevedi, ga s programatorjem PICkit 2 vpiši v mikrokontroler in preizkusi delovanje na preizkusni ploščici. Mikrokontrolerju določa takt zunanji kristalni oscilator vrednosti 10 MHz.
4. V register TMR0 lahko poljubno vpisujemo podatke in tako določimo točen čas, kdaj bo prekoračena njegova vrednost. Katero vrednost moramo vpisati v ta register, da nastane prekoračitev po času 200 µs? Preddelilnika ne uporabljamo, takt določa oscilator vrednosti 4 MHz. Kolikšna je ta vrednost, če mikrokontrolerju določa takt oscilator z vrednostjo 10 MHz?
5. Z mikrokontrolerjem PIC16F628A želimo vklopiti opozorilo, namenjeno kontroli izdelkov na tekočem traku v proizvodnji. Ko kontrolor pritisne na tipko T_1, se po 20. izdelku za 2 sekundi vklopita svetleča dioda in piskač. Ko kontrolor izdelek odda v kontrolo in pritisne na tipko T_1, se opozorilo ponovi po 20. izdelku. Izdelaj diagram poteka in napiši program, ki bo deloval po navedenih zahtevah. Izdelke na tekočem traku zazna svetlobni senzor, fotoupor. Ko je izdelek med virom svetlobe in fotouporom, je napetost na fotouporu 4 V, če izdelka med virom svetlobe in fotouporom ni, pa znaša napetost 0,2 V.



Slika 23: Shematični prikaz izdelkov na tekočem traku

Opozorilo naj se vklopi v prekinitveni rutini. Spremembo signala na fotouporu zazna pin RA4. Tipka T_1 je priključena na pin RA0, svetleča dioda na pin RA1 in piskač na pin RA2. Mikrokontrolerju določa takt kristalni oscilator z vrednostjo 4 MHZ. Program prevedi, ga s programatorjem PICkit 2 vpiši v mikrokontroler in preizkusi delovanje na preizkusni ploščici.

6. Program prekinitve na časovniku spremeni tako, da se bodo svetleče diode preklopile po vsaki 125. prekinitvi na časovniku. V ta namen deklariraj novo spremenljivko Stevec, ki šteje prekinitve. Preddelitev časovnika TMR0 je (pri 4 MHz oscilatorju) nastavljena tako, da se prekinitev pojavi približno vsake 4 ms. V prekinitveni rutini naj se spremenljivka Stevec zmanjšuje za 1, in ko je njena vrednost enaka 0, se morajo svetleče diode preklopiti. Po 125 prekinitvah se bo to zgodilo približno vsake pol sekunde.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

PRIKAZOVALNIKI, MATRIČNA TIPKOVNICA

7-segmentni LED-prikazovalnik

7-segmentni prikazovalnik priključimo na mikrokontroler tako, da vsak segment povežemo z lastnim pinom mikrokontrolerja preko zaščitnega upora. Za vklapljanje posameznih segmentov prikazovalnika z logično 1 uporabljamo take s skupno katodo, za vklapljanje z logično 0 pa prikazovalnike s skupno anodo.

Segment	a	b	c	d	e	f	g	dp
Pin	RB5	RB7	RB1	RB0	RB2	RB4	RB6	RB3

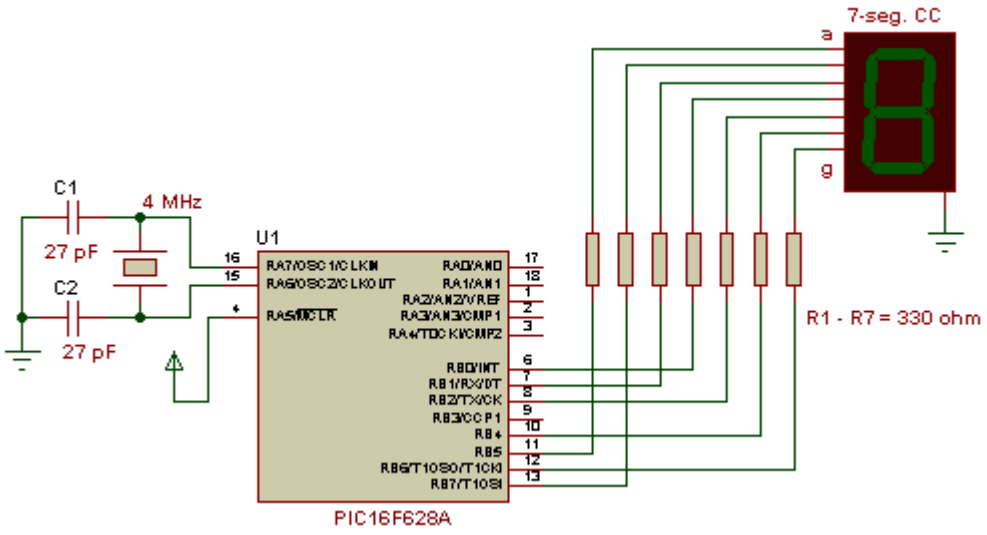
Tabela 6: Priključitev 7-segmentnega LED-prikazovalnika na mikrokontroler

Zaradi enostavnosti vezja, npr. konstruiranja v okolju Eagle, bomo priključili segmente prikazovalnika na pine mikrokontrolerja, kot je prikazano v tabeli 6. Ne smemo pozabiti na zaščitne upore, vsak segment prikazovalnika povežemo z lastnim pinom mikrokontrolerja preko upora 330 Ω. Poglejmo si program, ki na prikazovalniku s skupno katodo prikazuje štetje od 0 do 9 v intervalih 1 s:

```
;-------------------------------------------------------------------------------
;7-segmentni prikazovalnik
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list        p=16f628a               ;Tip mikrokontrolerja
        #include    <p16f628a.inc>          ;Vključi v program datoteko p16f628a.inc.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC

Stevilo  equ        0x20                    ;Število za izpis
Stevec   equ        0x30                    ;Števec za zakasnitev

        org         0x000                   ;Ponastavitveni vektor
        goto        Glavni                  ;Nadaljuj na naslovu Glavni.
        org         0x004                   ;Prekinitveni vektor

;********************** Glavni program **********************
Glavni
        bcf         STATUS,5                ;Banka 0
        movlw       b'00000111'
        movwf       CMCON                   ;Omogočimo vhodno-izhodne pine na PORTA.
        bsf         STATUS,RP0              ;Banka 1
        clrf        TRISB                   ;Vsi pini PORTB so izhodni.
        movlw       b'10000011'
        movwf       OPTION_REG              ;Preddelilnik dodeljen časovniku TMR0, preddelitev 1 : 16
        bcf         STATUS,5                ;Banka 0
        clrf        PORTB                   ;Inicializacija PORTB
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        clrf    INTCON          ;Onemogočimo odzivanje na prekinitve.
        clrf    Stevilo         ;Inicializacija, Stevilo = 0
Prikazuj
        call    Izpisi          ;Izpiši število na 7-segmentni prikazovalnik.
        call    Cakaj           ;Zakasnitev 1 s
        incf    Stevilo,f       ;Povečaj Stevilo za 1.
        movlw   .10
        subwf   Stevilo,w
        btfsc   STATUS,Z        ;Ali smo že prišli do številke 9?
        clrf    Stevilo         ;Da, postavi Stevilo na 0.
        goto    Prikazuj        ;Ne, prikaži naslednjo številko.

;********************** Podprogram za zakasnitev 1 s **********************
Cakaj
        clrf    Stevec          ;Stevec = 0
        clrf    TMR0            ;TMR0 = 0
Se
        btfss   INTCON,T0IF     ;Ali je preteklo 4,1 ms? Ali je TMR0 prekoračil svojo vrednost?
        goto    Se              ;Ne
        clrf    INTCON          ;Da, postavi zastavico T0IF na 0.
        incf    Stevec,f        ;Povečaj Stevec za 1.
        movlw   .244            ;Vrednost delovnega registra je 244.
        subwf   Stevec,w        ;Vrednost delovnega registra odštej od vrednosti Stevec.
        btfss   STATUS,Z        ;Z = 1? Ali je pretekla 1 sekunda? (4,1 ms x 244 ~ 1 s)
        goto    Se              ;Ne še
        return                  ;Da

;********************** Tabela znakov za prikazovalnik **********************
Znaki   addwf   PCL,f
        DT      0xB7, 0x82, 0xE5, 0xE3, 0xD2, 0x73, 0x77, 0xA2, 0xF7, 0xF3

;********* Podprogram za izpis znaka na 7-segmentni prikazovalnik *********
Izpisi
        movf    Stevilo,w       ;Kopiraj vrednost registra Stevilo v delovni register.
        call    Znaki           ;V tabeli izberi ustrezni znak.
        movwf   PORTB           ;Izpiši število na 7-segmentni prikazovalnik.
        return

        end
```



Slika 24: Priklop 7-segmentnega prikazovalnika na mikrokontroler

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

V glavnem programu nastavimo časovnik TMR0 tako, da deluje kot števec časa s preddelilnikom 1 : 16. Časovnik smo uporabili zato, da z njim odmerimo približno 1 sekundo za zakasnitev med spreminjanjem vrednosti na 7-segmentnem prikazovalniku. Časovnik je nastavljen tako, da povečuje svojo vrednost od 0 do 255 v 4,1 ms pri 4 MHz oscilatorju. Podprogram Cakaj se izvaja toliko časa, dokler časovnik 244-krat ne prekorači svoje vrednosti (overflow). V ta namen preverjamo zastavico T0IF v registru INTCON, ki se ob prekoračitvi na časovniku postavi na 1. Za preverjanje števila prekoračitev časovnikovega registra smo uporabili novo instrukcijo **subwf** (vrednost registra Work odšteje od vrednosti registra f, ki je v našem primeru Stevec). Pred to instrukcijo smo delovnemu registru določili vrednost 244, ker želimo 244 ponovitev. Spremenljivka Stevec se povečuje za 1 vsakokrat, ko časovnik prekorači svojo vrednost. Ko časovnik 24-krat prekorači svojo vrednost, je vrednost spremenljivke Stevec enaka 244. Ker je rezultat odštevanja 244 – 244 = 0, se zastavica Z v registru STATUS (njegov tretji bit) postavi na 1, kar je pogoj, da se iz podprograma za zakasnitev vrnemo v glavni program. Enako v glavnem programu preverjamo, ali smo že prešteli do 9.
V zanki v glavnem programu povečujemo vrednost spremenljivke Stevec in njeno vrednost vsakič spremenimo v kodo znaka, ki ga želimo prikazati na prikazovalniku. Pretvorbo napravimo s tabelo v programskem pomnilniku. Če želimo na primer na 7-segmentni prikazovalnik izpisati številko 7, shranimo v delovni register vrednost 7 in nato pokličemo tabelo, ki ima na osmem mestu (šteje se od mesta 0 naprej) kodo znaka 7 (0xA2). To kodo pošljemo na PORTB, kamor je priključen 7-segmentni prikazovalnik s skupno katodo. Če kodo 0xA2 pretvorimo v dvojiški številski sistem, dobimo b'10100010', kar pomeni, da bo na priključkih RB1, RB5 in RB7 napetost 5 V. Ker so na te priključke povezani segmenti c, a in b 7-segmentnega prikazovalnika, bo prikazoval številko 7.
Za shranjevanje vrednosti spremenljivke Stevilo v delovni register smo uporabili novo instrukcijo **movf**. Ta kopira vrednost prvega parametra, ki je v našem primeru Stevilo, na mesto, določeno z drugim parametrom, v našem primeru w.

Tabele podatkov

Programski pomnilnik mikrokontrolerja pogosto uporabljamo za shranjevanje podatkov, napisanih v obliki tabele. Ta ima obliko podprograma. Preden tabelo pokličemo z instrukcijo call, moramo v delovni register vpisati zaporedno številko (odmik) podatka, ki ga želimo dobiti iz tabele. Prvi element tabele ima odmik 0. Ko se tabela izvrši, dobimo v delovnem registru vrednost izbranega podatka v tabeli. Tak način pridobivanja podatkov iz tabele nam omogoči direktiva DT (**d**efine **t**able – definiraj tabelo).Ta direktiva nadomesti vsak 8-bitni podatek za njo od leve proti desni. Podatek je lahko številka, znak ali beseda:

```
DT      .12, 0xA2, 'K', "Mikrokontroler".
```

Podatke ločimo z vejico. Paziti moramo, da pri klicu ne prekoračimo konca tabele. Tako lahko v tabelo zapišemo največ 256 8-bitnih podatkov, saj je PCL 8-bitni register. Mikrokontroler vsebuje poseben register, imenovan programski števec. V njem se vedno nahaja naslov naslednje instrukcije, ki se mora v programu izvesti. Pri mikrokontroleju PIC16F628A je ta register 13-biten, kar zadošča za 8 k besed programskega pomnilnika. Ker ima PIC16F628A na voljo le 2 k programskega pomnilnika Flash, ostaneta zgornja dva bita programskega števca neuporabna. Tako nam ostane 11-biten programski števec, ki lahko naslovi 2 k (2048) besed pomnilnika. Programski števec je dosegljiv preko dveh registrov SFR: PCLATH in PCL, ki se nahajata v vseh štirih bankah. Register PCL vsebuje spodnjih 8 bitov programskega števca in ga lahko poljubno beremo ali pa vanj vpisujemo. Zgornjih 5 bitov programskega števca pa se nahaja v registru PCLATH, ki ni direktno dosegljiv.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

Poglejmo primer tabele iz prejšnje vaje, ko je 7-segmentni prikazovalnik izpisoval številke od 0 do 9:

```
;********************** Tabela znakov za prikazovalnik **********************
Znaki  addwf    PCL,f
       DT       0xB7,  0x82,  0xE5,  0xE3,  0xD2,  0x73,  0x77,  0xA2,  0xF7,  0xF3
```

Preden smo z instrukcijo **call** Znaki poklicali tabelo, smo delovnemu registru določili odmik podatka, ki smo ga želeli dobiti iz tabele. Prva instrukcija tabele **addwf** PCL,f prišteje registru PCL vrednost delovnega registra w. To pomeni, da se bo izvršila instrukcija, ki je na novem naslovu programskega pomnilnika. Če damo v delovni register w vrednost 0 in pokličemo tabelo, bomo po vrnitvi iz tabele dobili v delovnem registru w vrednost prvega podatka (od leve proti desni) tabele (0xB7). Če damo v register w vrednost 5 in pokličemo tabelo, bomo po vrnitvi iz tabele dobili v delovnem registru w vrednost šestega podatka tabele (0x73), saj se šteje od mesta 0 naprej. V prejšnjem programu se je vrednost delovnega registra w povečevala po vrsti od 0 do 9.

Povzetek

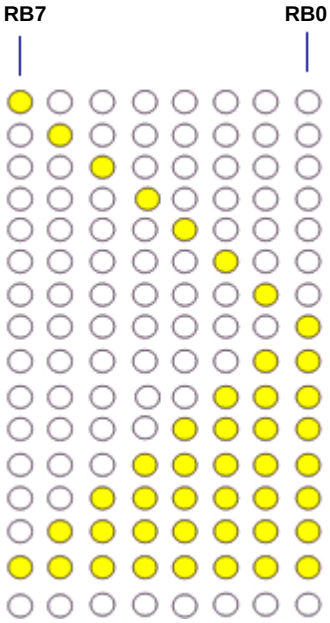
Programski pomnilnik mikrokontrolerja PIC je v prvi vrsti namenjen shranjevanju programa. Pogosto pa ga uporabljamo za shranjevanje podatkov v tabelah, ki jih v programski pomnilnik vpišemo skupaj s programom. Tabele definiramo z direktivo DT. V tabele lahko zapisujemo podatke v obliki znakov, besed ali števil v različnih številskih sistemih, zapišemo pa lahko največ 256 8-bitnih podatkov. Za vrnitev iz tabele s pravim podatkom poskrbi programski števec, v katerem se vedno nahaja naslov naslednje instrukcije, ki se mora v programu izvesti.
Na mikrokontroler smo priključili 7-segmentni prikazovalnik s skupno katodo. Za zaščito segmentov prikazovalnika smo vsakega posebej preko 330-ohmskih uporov povezali z ustreznim priključkom mikrokontrolerja, skupne katode vseh segmentov pa z maso.
V programu smo časovnik TMR0 uporabili kot števec časa. Z dodelitvijo in nastavitvijo preddelilnika smo mu določili čas, v katerem doseže svojo maksimalno vrednost 255. Število želenih prekoračitev časovnika smo preverjali z zastavico registra STATUS, ki se postavi na vrednost 1, ko je rezultat zadnje operacije enak 0.

Vaje

1. Program 7-segmentni prikazovalnik spremeni, da bo prikazoval štetje od 0 do 9 na 7-segmentnem prikazovalniku s skupno anodo. Izvajanje programa simuliraj s simulatorjem MPLAB, pri tem pa v *Watch* oknu opazuj spreminjanje vrednosti registrov GPR Stevilo in Stevec ter registrov SFR PORTB, WREG, STATUS, TMR0 in PCL.
2. Program 7-segmentni prikazovalnik spremeni, da bo prikazoval štetje od 4 do 8 v presledkih 0,5 sekunde na 7-segmentnem prikazovalniku s skupno katodo. Zakasnitev izvedi s časovnikom TMR0 in preddelilnikom. S programatorjem PICkit 2 programiraj mikrokontroler PIC16F628A, ki mu takt določa kristalni oscilator vrednosti 4 MHz. Prikaži delovanje na preizkusni ploščici.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

3. Na PORTB mikrokontrolerja PIC16F628A priključi 8 svetlečih diod. Te naj se vklapljajo po vzorcu, prikazanem na sliki 25 v presledku 500 ms. Podatke vzorcev zapiši v šestnajstiškem številskem sistemu v tabelo programskega pomnilnika.



Slika 25: Vzorec vklapljanja svetlečih diod

Mikrokontrolerju določa takt kristalni oscilator vrednosti 10 MHz. Program prevedi in ga s programatorjem PICkit 2 prenesi v mikrokontroler. Vezje sestavi na preizkusni ploščici in prikaži delovanje.

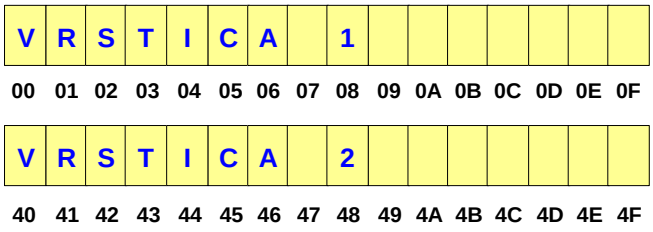
4. Izdelaj program za mikrokontroler PIC16F628A, ki bo deloval po naslednjih zahtevah. S pritiskom na tipko T_1, povezano na priključek RA0 mikrokontrolerja, sprožimo prikaz štetja na 7-segmentnem prikazovalniku s skupno katodo od 0 do 9 v presledkih 500 ms. S tipko T_2, povezano na priključek RA1 mikrokontrolerja, pa sprožimo prikaz štetja od 9 do 0 v enakih presledkih. Mikrokontrolerju določa takt kristalni oscilator vrednosti 4 MHz. Program prevedi in ga s programatorjem PICkit 2 prenesi v mikrokontroler. Vezje sestavi na preizkusni ploščici in prikaži delovanje.
5. Izdelaj program za mikrokontroler PIC16F628A, ki bo na 7-segmentnem prikazovalniku s skupno katodo v presledku 1 s prikazoval črke A, C, E, F, G, H, I, L, O, P, S in U. Mikrokontrolerju določa takt kristalni oscilator vrednosti 4 MHz. Program prevedi in ga s programatorjem PICkit 2 prenesi v mikrokontroler. Prikaži delovanje na preizkusni ploščici.
6. Izdelaj program za mikrokontroler PIC16F628A, ki bo na 7-segmentnem prikazovalniku s skupno katodo prikazoval število zasedenih parkirnih mest na parkirišču. Ko voznik avtomobila s pritiskom na tipko T_1, povezano na priključek RA1 mikrokontrolerja, sproži odpiranje parkirne zapornice pri vhodu na parkirišče, se število zasedenih parkirnih mest poveča za 1. Ko pa voznik zapušča parkirišče, s pritiskom na tipko T_2, povezano na priključek RA2 mikrokontrolerja, sproži odpiranje parkirne zapornice pri izhodu s

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

parkirišča in število zasedenih parkirnih mest se zmanjša za 1. Parkirišče vsebuje 8 parkirnih mest.

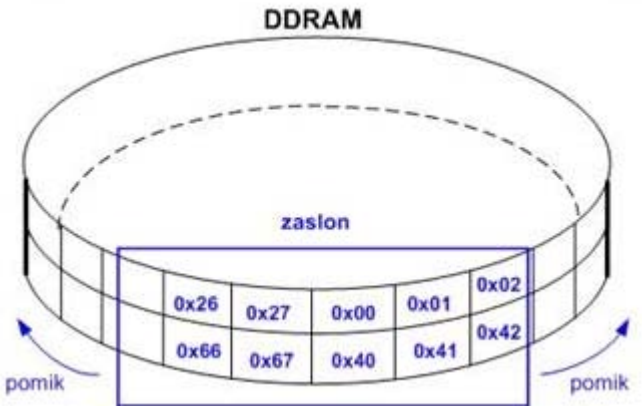
LCD-prikazovalnik

LCD-prikazovalniki (**L**iquid **c**ristal **d**isplay – prikazovalniki na tekoče kristale) omogočajo izpisovanje črk, številk in ostalih znakov. Uporabljamo jih kot vmesnike med uporabnikom in elektronsko napravo. V vajah bomo uporabili dvovrstične LCD-prikazovalnike z lastno osvetlitvijo, ki imajo v vsaki vrstici po 16 znakov (DEM 16216 SYH-PY). Vsebujejo lasten kontroler (KS0070B-00), ki skrbi za vklapljanje ustreznih pik na zaslonu. Vsak znak je sestavljen iz 5 x 8 pik. LCD-kontroler KS0070B-00 ima 80 bajtov zaslonskega pomnilnika RAM, imenovanega DDRAM (Display data RAM). Vsebuje še znakovni pomnilnik ROM s podatki o simbolih (imenovanega CGROM) in 64 bajtov zaslonskega pomnilnika RAM (imenovanega CGRAM), namenjenega izdelavi lastnih simbolov. CGROM je pomnilnik, ki ga ne moremo spreminjati. Vsebuje podatke o tem, kako so sestavljeni znaki, ki jih LCD pozna: vse črke angleške abecede, številke, ločila in še ostale znake, ki so odvisni od proizvajalca, osnoven nabor znakov ASCII pa je vedno enak.



Slika 26: LCD 2 x 16, naslovi DDRAM – šestnajstiške vrednosti

Vsak znak na LCD-prikazovalniku ima svoj naslov (slika 26) v DDRAM-u. Začetni naslov skrajnega levega znaka v prvi vrstici je 0x00, skrajnega levega znaka v drugi vrstici pa 0x40. Vsaka vrstica je navidezno dolga po 40 znakov, kar znaša skupaj 80 znakov. Vsak znak zavzame 1 bajt pomnilnika DDRAM. Od vseh 40 znakov v vsaki vrstici jih trenutno vidimo na zaslonu le 16. Tega lahko poljubno premikamo po celotnem prikazovalniku in tako določimo, kaj bo v določenem trenutku prikazano na zaslonu.



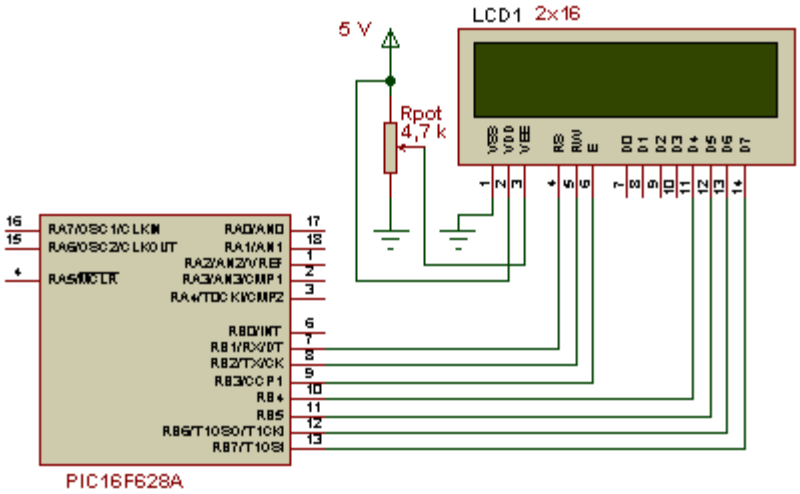
Slika 27: Kar v resnici vidimo na LCD-ju, je le del celotnega DDRAM-a.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

CGRAM ima enako vlogo kot CGROM, le da lahko vanj sami vpisujemo podatke oziroma jih iz njega beremo. Če delamo z znaki velikosti 5 x 8 pik, lahko ustvarimo do 8 novih znakov, ki imajo kode od 0x00 do 0x07. Vsak znak vsebuje 8 naslovov, vrednosti znaka pa so odvisne od novega simbola oziroma znaka, ki smo ga ustvarili.

Priklop LCD-prikazovalnika na mikrokontroler

LCD- prikazovalniki brez možnosti osvetlitve zaslona imajo 14 priključkov, taki z možnostjo osvetlitve zaslona pa imajo 2 priključka več, torej 16 priključkov. Funkcija posameznega priključka je standardna, vedeti moramo le, kje se nahaja prvi priključek. Če priključki za določen tip LCD-ja niso označeni, poiščemo podatke na spletu.



Slika 28: Priklop LCD-ja na mikrokontroler

Priključki od 7 do 14 LCD-prikazovalnika so namenjeni podatkovnim linijam (od D0 do D7). Preko njih se prenašajo podatki od mikrokontrolerja do LCD-ja, če pišemo v LCD, in od LCD-ja do mikrokontrolerja, če beremo iz LCD-ja. LCD-prikazovalniki lahko delujejo v 8- ali v 4-bitnem načinu. Mi bomo izbrali 4-bitni način delovanja, saj s tem privarčujemo pri priključkih mikrokontrolerja. Ker iz LCD-ja ne bomo ničesar brali, bomo priključek 5 (R/W) povezali z maso in s tem pridobili dodaten prost priključek na mikrokontrolerju. Podatki se pri 4-bitnem načinu delovanja prenašajo po 4 bite hkrati, zato moramo poslati obe polovici bajta posebej, pošiljamo pa jih po podatkovnih linijah od D4 do D7. Tretji priključek LCD-ja služi nastavitvi kontrasta. Kontrast reguliramo z napetostjo na tem priključku. V ta namen uporabimo potenciometer kot delilnik napetosti.
LCD-prikazovalnik krmilimo tako, da mu pošiljamo ukaze ali podatke. Če postavimo priključek 4 (RS) v stanje logične 0, bo LCD sprejel podatek kot ukaz, če pa ga postavimo na nivo logične 1, bo LCD sprejel navaden podatek. Priključek 6 (E) je namenjen vklopu prikazovalnikove logike. Ko pošiljamo neki podatek na LCD, ga moramo nekako obvestiti, da je na podatkovnih linijah nov podatek. To storimo tako, da ta priključek vklopimo (postavimo na 1) in izklopimo (postavimo na 0). Ob tem prehodu bo LCD sprejel nov podatek. Pri 4-bitnem načinu delovanja pošljemo najprej zgornje 4 bite podatka na priključke 11–14 (D4–D7), vmes vklopimo in izklopimo priključek 6 (E) in nato na iste priključke pošljemo še spodnje 4 bite podatka. Zatem zopet vklopimo in izklopimo priključek 6 (E). Vsak vpis ukaza ali podatka vzame LCD-ju nekaj časa, da ga obdela. V tem času, podajajo ga proizvajalci LCD-jev, mu ne smemo pošiljati

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

naslednjega ukaza ali podatka, zato uporabimo metodo zakasnitve, ki bo v našem primeru ca. 5 ms. Časa zakasnitve ni treba točno nastaviti, le manjši od predpisanega ne sme biti.

Št. priključka	Simbol priključka	Funkcija priključka
1	GND	masa
2	Ucc	napetost napajanja +5 V
3	Uk	nastavitev kontrasta
4	RS	preklop podatkovni/ukazni način delovanja
5	R/W	preklop bralno/pisalni način delovanja
6	E	vklop krmilne logike
7	D0	podatkovne linije (linije D0–D3 niso uporabljene pri 4-bitnem načinu delovanja)
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	LED- (K)	napetostno napajanje za osvetlitev zaslona, +5 V, 33 mA
16	LED+ (A)	

Tabela 7: Funkcije priključkov LCD 2 x 16

LCD-prikazovalnik moramo pred uporabo inicializirati. To pomeni, da moramo izvesti določeno zaporedje operacij, da ga postavimo v želeni način delovanja. Uporabili bomo inicializacijo za delo v 4-bitnem načinu. Opis korakov, potrebnih pri inicializaciji, si poglejmo v programu, ki bo v prvi vrstici LCD-ja zapisal "Zelimo vam", v drugi vrstici pa "varno voznjo", kot kaže slika 29.

Slika 29: Prikaz napisa na LCD-ju

```
;-------------------------------------------------------------------------------
; Prikaz napisa na LCD-ju
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list        p=16f628a               ;Tip mikrokontrolerja
        #include    <p16f628a.inc>          ;Vključi v program datoteko p16f628a.inc.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC


;********************** Deklaracija spremenljivk **********************
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        Cblock  0x20            ;Blok konstant in začasnih spremenljivk
                Temp1
                Temp2           ;Začasni spremenljivki
                Naslov          ;Nov naslov DDRAM-a
                Stevec          ;Števec za izpis niza v prvi vrstici
                Stevec1         ;Števec za izpis niza v drugi vrstici
        endc

;******************** Definiranje novih oznak ********************
        #define RS      PORTB,1         ;RS-linija LCD-ja
        #define RW      PORTB,2         ;RW-linija LCD-ja
        #define E       PORTB,3         ;E-linija LCD-ja

;*************** Definiranje konstant za LCD ukaze ***************
DDRAM   equ     0x80            ;Maska za vpis v DDRAM
Kurzor  equ     0x0C            ;Izklop kazalčka

;*************** Začetek ****************

        org     0x000                   ;Ponastavitveni vektor
        goto    Glavni_zacetek          ;Nadaljuj na naslovu Glavni_zacetek.
        org     0x004                   ;Prekinitveni vektor

;********************** Glavni program **********************
Glavni_zacetek
        bsf     STATUS,5                ;Banka 1
        clrf    TRISA
        clrf    TRISB                   ;Vsi pini PORTA in PORTB so izhodni.
        movlw   b'10000001'
        movwf   OPTION_REG              ;Preddelilnik dodeljen časovniku TMR0, preddelitev 1 : 4.
        bcf     STATUS,5                ;Banka 0
        clrf    PORTB                   ;Inicializacija PORTB
        clrf    INTCON                  ;Onemogočimo odzivanje na prekinitve.
        call    LCD_Init                ;Inicializacija LCD
        bsf     RS                      ;Podatkovni način, pisanje znakov
        call    LCD_Niz1                ;Izpišemo niz na LCD (prva vrstica).
        clrf    Stevec                  ;Spremenljivka Stevec = 0
        movlw   0x40                    ;Inicializacija LCD
        call    DDRAM_n                 ;Pišemo na začetek druge vrstice LCD (naslov 0x40).
        bsf     RS                      ;Podatkovni način, pisanje znakov
        call    LCD_Niz2                ;Izpišemo niz na LCD (druga vrstica).
        clrf    Stevec1                 ;Spremenljivka Stevec1 = 0
Delaj   goto    Delaj                   ;Neskončna zanka

;********************** Podprogrami **********************
;******* Izpis na LCD v 4-bitnem načinu *********************
LCD_pisi
        movwf   Temp1                   ;Podatek iz W v Temp1
        movlw   0x0F                    ;b'00001111' v register W
        andwf   PORTB,f                 ;Zgornje 4 bite registra PORTB izbrišemo.
        movf    Temp1,w                 ;Registru W povrnemo podatek iz Temp1.
        andlw   0xF0                    ;Izbrišemo spodnje štiri bite.
        iorwf   PORTB,f                 ;Pošljemo zgornje štiri bite podatka.
        bsf     E
        bcf     E                       ;Obvestimo LCD o novem podatku.
        swapf   Temp1,f                 ;Zamenjamo polbajte v Temp1.
        movlw   0xF0                    ;b'11110000' v register W
        andwf   Temp1,f                 ;Spodnje 4 bite v Temp1 izbrišemo.
        movlw   0x0F                    ;b'00001111' v register W
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        andwf   PORTB,f         ;Zgornje 4 bite v registru PORTB izbrišemo.
        movf    Temp1,w         ;Podatek iz Temp1 v register W
        iorwf   PORTB,f         ;Pošljemo spodnje štiri bite podatka.
        bsf     E
        bcf     E               ;Obvestimo LCD o novem podatku.
        call    Cakaj_155_us    ;Zakasnitev 155 µs
        return

;****************** Inicializacija LCD-ja  *********************
LCD_Init
        bcf     RW
        bcf     RS              ;Ukazni način, vpisovanje podatkov v LCD
        movlw   .34
        call    Cakaj_ms        ;ca. 35 ms pavze
        movlw   0x30
        movwf   PORTB           ;Na D7–D4 pošljemo 0x30.
        bsf     E
        bcf     E               ;Obvestimo LCD o novem podatku.
        movlw   .5
        call    Cakaj_ms        ;ca. 5 ms pavze
        movlw   0x30
        movwf   PORTB           ;Na D7–D4 pošljemo 0x30
        bsf     E
        bcf     E               ;Obvestimo LCD o novem podatku.
        call    Cakaj_155_us    ;Zakasnitev 155 µs
        movlw   0x30
        movwf   PORTB           ;Na D7–D4 pošljemo 0x30.
        bsf     E
        bcf     E               ;Obvestimo LCD o novem podatku.
        movlw   .5
        call    Cakaj_ms        ;ca. 5 ms pavze
        movlw   0x20
        movwf   PORTB           ;Na D7–D4 pošljemo 0x20.
        bsf     E
        bcf     E               ;Obvestimo LCD o novem podatku.
        call    Cakaj_155_us    ;Zakasnitev 155 µs
                                ;Sedaj smo v 4-bitnem načinu.
        movlw   0x28            ;LCD 2 x 16, 4-bitni način, velikost znakov 5 x 8 pik
        call    LCD_pisi        ;Pošljemo 0x28 na LCD.
        movlw   0x08            ;Izklop zaslona
        call    LCD_pisi        ;Pošljemo 0x08 na LCD.
        movlw   0x01            ;Brisanje DDRAM-a
        call    LCD_pisi        ;Pošljemo 0x01 na LCD, izbrišemo zaslon.
        movlw   .2
        call    Cakaj_ms        ;Zahtevana zakasnitev, večja od 2 ms
        movlw   0x06            ;Pisanje v desno brez pomikanja zaslona
        call    LCD_pisi        ;Pošljemo 0x06 na LCD
        movlw   0x0F            ;Vklop zaslona, veliki kazalček
        call    LCD_pisi        ;Pošljemo 0x0F na LCD
        return

;****************** Nastavitev trenutnega naslova DDRAM-a  *********************
                                ;Naslov DDRAM-a vpišemo v register W pred klicem podprograma.
DDRAM_n
        bcf     RS              ;Ukazni način
        iorlw   DDRAM           ;Z OR-operacijo ustvarimo ukaz.
        call    LCD_pisi        ;Vpis ukaza v LCD
        bsf     RS              ;Podatkovni način
        return
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
;****************** Zakasnitev 155 µs  *********************
Cakaj_155_us
        movlw   .50             ;Število ponovitev zanke prestavimo v register W
        movwf   Temp2           ;Vrednost registra W v Temp2
Se      decfsz  Temp2,f         ;Temp2 = Temp2 - 1. Temp2 = 0?
        goto    Se              ;Temp2 še ni enak 0. Še zmanjšuj Temp2.
        return                  ;Temp2 = 0. Vrni se iz podprograma.

;********* Zakasnitev ca. 1 ms krat vrednost registra W *********
Cakaj_ms
        movwf   Temp2           ;W v Temp2 (iz LCD_Init je vrednost W enaka 34 oziroma 5)
        clrf    TMR0            ;Inicializacija registra TMR0
Cak1    btfss   INTCON,T0IF     ;Ali je TMR0 prekoračil svojo vrednost (255)?
        goto    Cak1            ;Ne še
        bcf     INTCON,T0IF     ;Da, postavi zastavico T0IF na 0.
        decfsz  Temp2,f         ;Temp2 = Temp2 - 1
        goto    Cak1            ;Temp2 še ni enak 0.
        return                  ;Temp2 = 0. Vrni se iz podprograma.

;*************** Naša niza, napisana sta v tabelah ***************
Tabela1
        addwf   PCL,f
        DT      "Zelimo vam"    ;Tabela1 z nizom
Tabela2
        addwf   PCL,f
        DT      "varno voznjo"  ;Tabela2 z nizom

;*************** Niz1 za izpis v prvi vrstici LCD-ja ***************
LCD_Niz1
        clrf    Naslov          ;Naslov = 0
        bsf     RS              ;Podatkovni način
        movlw   .10
        movwf   Stevec          ;Niz 1 ima 10 znakov (9 črk in 1 presledek: Zelimo vam).
Delaj1
        movf    Naslov,w        ;Odmik = 0, začnemo s prvim znakom niza iz tabele 1.
        call    Tabela1
        call    LCD_pisi        ;Podatek iz tabele 1 v LCD
        incf    Naslov,f        ;Naslednji podatek iz tabele 1
        decfsz  Stevec,f        ;Ali se vpisani vsi podatki iz tabele 1 v LCD?
        goto    Delaj1          ;Ne še
        bcf     RS              ;Da, preklopi na ukazni način.
        return

;*************** Niz2 za izpis v drugi vrstici LCD-ja ***************
LCD_Niz2
        clrf    Naslov          ;Naslov = 0
        bsf     RS              ;Podatkovni način
        movlw   .12
        movwf   Stevec1         ;Niz 2 ima 12 znakov (11 črk in 1 presledek: varno voznjo).
Delaj2
        movf    Naslov,w        ;Odmik = 0, začnemo s prvim znakom niza iz tabele 1
        call    Tabela2
        call    LCD_pisi        ;Podatek iz tabele 1 v LCD
        incf    Naslov,f        ;Naslednji podatek iz tabele 1
        decfsz  Stevec1,f       ;Ali se vpisani vsi podatki iz tabele 1 v LCD?
        goto    Delaj2          ;Ne še
        bcf     RS              ;Da, preklopi na ukazni način.
        movlw   Kurzor          ;Izklopimo kazalček (koda ukaza 0x0C v register W)
        call    LCD_pisi
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        return
        end
```

Na začetku programa najdemo novo direktivo **Cblock**. Ta direktiva počne isto kot equ, le da nam nekoliko olajša delo. Z njo damo besedam vrednosti, ki so lahko naslovi spremenljivk. Prvi parameter poleg direktive (0x20) pomeni vrednost, ki jo bo imela prva oznaka na seznamu, ki tej direktivi sledi. V našem primeru je prva oznaka na seznamu Temp1, ki ima torej vrednost 0x20. To vrednost smo določili zato, ker se na tem naslovu prične podatkovni pomnilnik RAM, namenjen splošni uporabi pri mikrokontrolerju PIC16F628A. Vsaka naslednja oznaka na seznamu bo dobila vrednost, za 1 večjo od vrednosti predhodne oznake v seznamu.
Naslednja nova direktiva v programu je **#define**. Ta direktiva tekst v prvem parametru zamenja s tekstom v drugem parametru. Kadarkoli se npr. v programu pojavi beseda E, jo prevajalnik zamenja s tekstom PORTB,3, kamor tudi povežemo priključek 6 LCD-ja (E) na mikrokontroler (priključek RB3).
V podprogramu za inicializacijo LCD-ja najprej poskrbimo za zakasnitev 35 ms, da se lahko LCD-jeva elektronika stabilizira. Nato sledi natančno določeno zaporedje korakov, ki so potrebni za inicializacijo in delo v 4-bitnem načinu. Ti koraki so zapisani in komentirani v podprogramu LCD_Init.
V glavnem programu smo časovniku TMR0 dodelili preddelilnik 1 : 4, zato preseže svojo vrednost 255 v času ca. 1 ms pri 4 MHz oscilatorju. S TMR0 v podprogramu Cakaj_ms odmerimo zakasnitev, ki je odvisna od vrednosti delovnega registra W, preden se začne podprogram izvajati. Prvič je vrednost W enaka 34, drugič pa 5. V prvem primeru TMR0 34-krat preseže vrednost 255, v drugem primeru pa 5-krat, preden mikrokontroler zapusti podprogram. Da se pri prekoračenju TMR0 ne sproži prekinitev, smo registru INTCON izklopili njegov sedmi bit GIE. Zakasnitev 155 µs smo dosegli z vrtenjem programa v zanki v podprogramu Cakaj_155_us.
Poglejmo vlogo podprograma LCD_pisi bolj natančno. Podatek za vpis na LCD podprogram prebere iz registra W. Register W se namreč iz obeh tabel vsakič vrne z vrednostjo odmika, ki je v našem primeru črka ali presledek, in ga začasno shrani v Temp1. Kakšen bo odmik, je odvisno od vrednosti spremenljivke Naslov. Če je njegova trenutna vrednost 5, se bo register W iz tabele vrnil z vrednostjo šestega znaka v nizu. Spremenljivki Stevec in Stevec1 štejeta odmike v nizih, ki sta zapisana v obeh tabelah in skrbita, da tabele ne prekoračimo. Preden smo začeli iz posamezne tabele brati, smo postavili vrednost spremenljivke Naslov na 0, spremenljivkama Stevec in Stevec1 pa smo določili vrednost 10 oziroma 12, saj je v prvi tabeli 10 znakov (šteje tudi presledek), v drugi pa 12. V tem podprogramu smo uporabili nove instrukcije. Instrukcija **andlw** napravi logično operacijo AND med posameznimi enakoležečimi biti registra W in konstanto, podano v parametru, ki je v našem primeru 0xF0. Rezultat operacije se vpiše nazaj v register W. Ker imajo zgornji štirje biti vrednost 1 (F0 šestnajstiško je 11110000 dvojiško), instrukcija andlw pobriše priključke RB4–RB7 0, da se lahko nanje vpiše nov podatek, hkrati pa ohrani vrednosti priključkov RB0 – RB3, kamor so povezani RS-, R/W- in E-priključki LCD-ja. Instrukcija **andwf** napravi v našem programu logično operacijo AND med registroma W in PORTB, rezultat operacije pa shrani nazaj v register PORTB. Instrukcija **iorwf** pa napravi logično operacijo OR med registrom W in registrom, zapisanim v prvem parametru, v našem primeru je to PORTB. Rezultat operacije shrani v register PORTB. Instrukcijo swapf že poznamo, v programu pa jo uporabimo zato, da v začasni spremenljivki Temp1, kjer je shranjena trenutna koda znaka iz tabele, zamenja spodnje 4 bite z zgornjimi 4 biti. To storimo zato, da pošljemo na LCD enkrat zgornje 4 bite podatka in drugič spodnje 4 bite, saj delamo v 4-bitnem načinu. Poglejmo, kakšen je postopek vpisa črke Z, ki je prvi znak v tabeli 1. Koda znaka Z v CGROM-u je $01011010_{(2)}$. S to vrednostjo se iz tabele 1 vrne register W in jo shrani v Temp1. Nato registru W določimo vrednost $00001111_{(2)}$ (0Fh), da lahko z instrukcijo andwf pobrišemo

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

zgornje 4 bite registra PORTB. Zatem registru W povrnemo kodo znaka Z, ki je bila shranjena v Temp1. Z masko $11110000_{(2)}$ (F0h) in instrukcijo andlw pobrišemo spodnje 4 bite kode znaka Z, ki ima sedaj vrednost $01010000_{(2)}$. Da pri prenosu tega podatka na PORTB ne spremenimo spodnjih 4 bitov registra PORTB, uporabimo instrukcijo iorwf. Na podatkovnih linijah od D4 do D7 LCD-ja (povezani so na RB4 do RB7) so sedaj pripravljeni zgornji štirje biti kode znaka Z. LCD moramo samo še obvestiti, da je pripravljen nov podatek, zato vklopimo in izklopimo njegov priključek E. Po tem prehodu je LCD podatek prejel. Poslati mu moramo še spodnje 4 bite kode znaka Z. Z instrukcijo swapf zamenjamo polbajta v Temp1. Ta ima sedaj vrednost $10100101_{(2)}$. Spodnji 4 biti kode znaka Z so zamenjali mesto z zgornjimi 4 biti. Postopek enako kot prej ponovimo in po prehodu priključka E z 1 na 0 prejme LCD drugo polovico kode znaka Z. Pred pošiljanjem drugega znaka iz tabele na LCD počakamo ca. 155 µs. Zakasnitve, ki jih moramo upoštevati, so za posamezne tipe LCD-jev različne. Najdemo jih v proizvajalčevih katalogih. Za naš tip LCD-ja so podatki na spletni strani http://pdf1.alldatasheet.com/datasheet-pdf/view/110299/ETC/DEM16216SYH-PY.html
V katalogu so opisani tudi ukazi LCD-prikazovalnika in potrebne zakasnitve, ki jih posamezni ukaz potrebuje, da ga LCD obdela. Vseh je namreč preveč, da bi jih obravnavali v tem gradivu. Vsak ukaz je koda dolžine enega bajta, ki jo pošljemo LCD-ju. Mednje sodijo:

- Clear Display: Počisti zaslon in postavi DDRAM-a na 0x00.
- Return Home: Pomakne vidni del zaslona na začetni položaj, DDRAM se ne spremeni.
- Entry Mode Set: Znake lahko izpisujemo od leve proti desni ali nasprotno, vključimo ali izključimo lahko avtomatsko pomikanje vidnega polja zaslona.
- Display ON/OFF Control: Omogoča izklop LCD-ja. Izbiramo lahko tudi obliko kazalčka in ali bo viden ali ne.
- Cursor or Display Shift: Vidni del zaslona pomika po DDRAM-u. Izbiramo lahko med desnim in levim pomikom. Omogoča tudi premikanje kazalčka.
- Function Set: Izbiramo lahko med 4- in 8-bitnim delovanjem LCD-ja, število vrstic na LCD-ju in velikost znakov.
- Set CGRAM Address: Nastavimo naslov CGRAM-a, od katerega dalje se bodo vpisovali poslani podatki.
- Set DDRAM Address: Nastavimo naslov DDRAM-a, od katerega dalje se bodo vpisovali poslani podatki.
- Read Busy Flag & Address: Vsak vpis ukaza ali podatka vzame LCD-ju nekaj časa, da ga obdela. S stanjem zastavice ugotovimo, ali je LCD zaključil delo.

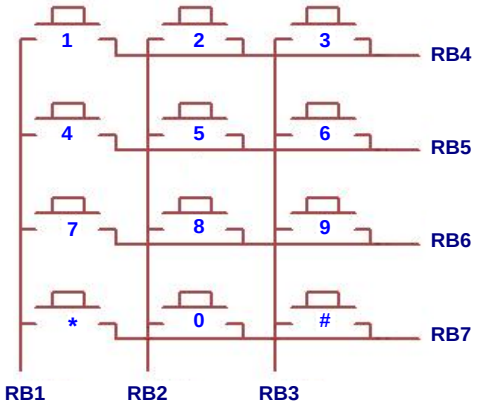
V katalogu najdemo tudi naslove vseh znakov DDRAM-a, ki jih je LCD zmožen prikazati. Poleg teh pa lahko sami izdelamo do osem svojih znakov. Katere vrednosti in naslove CGRAM-a imajo, je prikazano v katalogu.

Priklop matrične tipkovnice na mikrokontroler

Matrična tipkovnica vsebuje 12 tipk, ki so razporejene v 3 stolpce in 4 vrstice. Ugotoviti moramo, katera tipka je trenutno sklenjena. To storimo tako, da v določenih časovnih presledkih priklapljamo napetost na posamezne stolpce, pri tem pa preverjamo stanja vseh štirih vrstic. Ker v vsakem trenutku vemo, kateri stolpec je pod napetostjo, iz prebranega stanja vrstic ugotovimo,

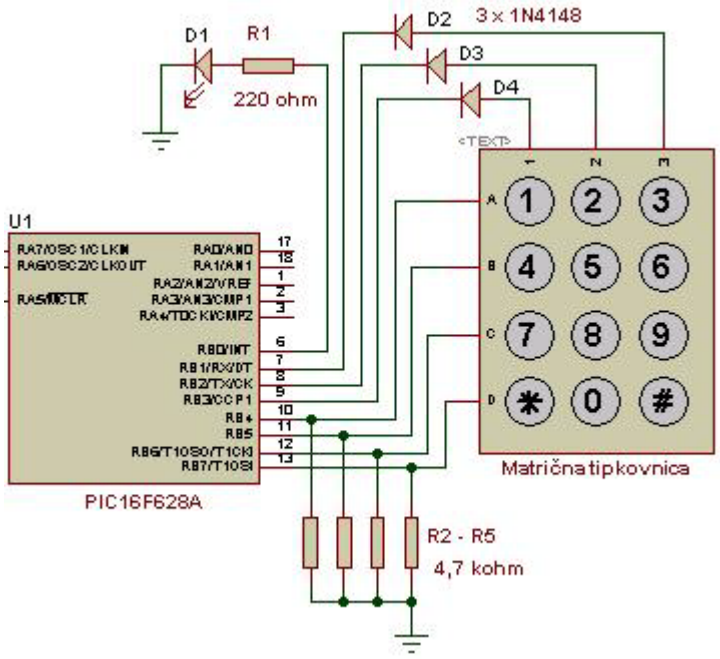
Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

katera tipka je sklenjena. Upoštevati moramo odbijanje kontaktov tipk pri sklenitvi, kar nam lahko povzroča težave. Zato programsko odstranimo te motnje z zakasnitvijo, kar že poznamo. Tipkovnico bomo priključili na mikrokontroler tako, da bo sklenjena tipka povzročila logično 1 na priključku mikrokontrolerja.



Slika 30: Zgradba matrične tipkovnice

Matrično tipkovnico bomo priključili na mikrokontroler, kot je prikazano na sliki 30. Izdelali bomo program, ki bo vklopil svetlečo diodo le, če pritisnemo na tipko 3.



Slika 31: Priklop matrične tipkovnice na mikrokontroler

```
;-------------------------------------------------------------------------------
; Priklop matrične tipkovnice
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list        p=16f628a               ;Tip mikrokontrolerja
        #include    <p16f628a.inc>          ;Vključi v program datoteko p16f628a.inc.
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC

;********************** Deklaracija spremenljivk **********************
        Cblock  0x20
                Temp
                Tipka
                Stevec
                Stevec1
        endc

Tipka3  equ     b'00010010'             ;Koda tipke 3
;*************** Začetek ****************

        org     0x000                   ;Ponastavitveni (reset) vektor
        goto    Glavni                  ;Nadaljuj na naslovu Glavni_zacetek.
        org     0x004                   ;Prekinitveni vektor

;********************** Glavni program **********************
Glavni
        bsf     STATUS,5        ;Banka 1
        movlw   b'11110000'
        movwf   TRISB           ;RB0–RB3 so izhodi, RB4–RB7 so vhodi.
        bcf     STATUS,5        ;Banka 0
        clrf    INTCON          ;Prekinitve so onemogočene.
        clrf    PORTB           ;Inicializacija PORTB
Delaj
        call    Zakasnitev      ;Zakasnitev zaradi odbijanja kontaktov
        call    Beri            ;Beremo tipkovnico.
        movf    Tipka,w
        movwf   Temp            ;Shranimo vrednost registra Tipka v Temp.
        call    Zakasnitev      ;Zakasnitev zaradi odbijanja kontaktov
        call    Beri            ;Ponovno beremo tipkovnico.
        movf    Tipka,w         ;Nova vrednost registra Tipka, shrani v W.
        subwf   Temp,w          ;Odštejemo vrednost w od vrednosti Temp.
        btfss   STATUS,Z        ;Primerjamo vrednost Tipka z vrednostjo Temp.
        goto    Delaj           ;Vrednosti nista enaki, zastavica Z ni enaka 1, preverjajmo naprej.
        movlw   b'11110000'
        andwf   Tipka,w         ;Pobrišemo spodnje 4 bite registra Tipka, rezultat operacije shranimo v W.
        btfsc   STATUS,Z        ;Ali je bila pritisnjena tipka?
        goto    Delaj           ;Ne, še preverjajmo.
        movlw   Tipka3
        subwf   Tipka,w         ;Primerjamo vrednost registra Tipka s Tipka3.
        bcf     PORTB,0         ;Izklopimo svetlečo diodo na RB0.
        btfsc   STATUS,Z        ;Ali je bila pritisnjena tipka 3?
        bsf     PORTB,0         ;Da, vklopi svetlečo diodo na RB0.
        goto    Delaj           ;Neskončna zanka

;********** Podprogram za branje matrične tipkovnice **********
Beri
        movlw   b'00000001'     ;Začnemo s prvim stolpcem.
        movwf   Tipka
Se
        bcf     STATUS,C
        rlf     Tipka,f         ;Premaknemo se na naslednji stolpec (levo).
        movlw   b'11110001'     ;Maska
        andwf   PORTB,f         ;RB1, RB2 in RB3 postavimo na 0.
        movf    Tipka,w         ;Vrednost Tipka v W
        iorwf   PORTB,f         ;Na ustrezni stolpec pošljemo 1 (5 V).
```

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
        btfsc   Tipka,4         ;Ali smo obdelali vse stolpce?
        Return                  ;Da
        movlw   b'11110000'     ;Ne
        andwf   PORTB,w         ;Pobriše spodnje 4 bite registra PORTB.
        btfsc   STATUS,Z        ;Ali je pritisnjena tipka?
        goto    Se              ;Ne, preverjaj naprej.
        iorwf   Tipka,f         ;V registru Tipka je koda tipke 3 matrične tipkovnice.
        return

;********** Podprogram za zakasnitev 20 ms **************
Zakasnitev
        movlw   .247
        movwf   Stevec
        movlw   .26
        movwf   Stevec1
Se1
        decfsz  Stevec,f
        goto    Se1
        decfsz  Stevec1,f
        goto    Se1
        return

        end
```

Svetleča dioda je povezana s priključkom mikrokontrolerja RB0. Vklopljena bo toliko časa, dokler bo sklenjena tipka 3 na matrični tipkovnici. Ali je tipka 3 sklenjena, preverja podprogram za branje matrične tipkovnice. Če je sklenjena, se iz podprograma vrne s spremenljivko Tipka, ki je enake vrednosti kot koda tipke 3 (Tipka3 b'00010010'). Prvi bit spremenljivke Tipka predstavlja priključek svetleče diode, naslednji trije biti označujejo stolpec pritisnjene tipke, zgornji štirje biti pa vrstico pritisnjene tipke. Če ni pritisnjena nobena tipka, je vrednost zgornjih 4 bitov enaka 0. Priključke posameznih stolpcev vklapljamo postopoma s pomikanjem. To smo dosegli z instrukcijo rlf, ki vsebino registra rotira za 1 bit v levo. Za vsakega izmed treh stolpcev (drugi, tretji in četrti bit) nato preberemo, ali je v kateri vrstici pritisnjena tipka. Ko pridemo do stolpca, kjer je tipka sklenjena, shranimo vrednost PORTB v spremenljivko Tipka. Tako dobimo kodo tipke 3 v spremenljivki Tipka. Ker so vrednosti enake, je rezultat odštevanja 0, zato se zastavica Z v registru STATUS postavi na 1. Izvede se naslednja instrukcija, ki vklopi svetlečo diodo, priključeno na RB0. Ta ostane vklopljena, dokler je tipka 3 sklenjena. Ker dobimo obe vrednosti enaki, tudi če nobena tipka ni sklenjena, moramo preveriti tudi stanje zgornjih 4 bitov. Če so vsi štirje enaki 0, ni bila pritisnjena nobena tipka. Če pa je kateri od teh bitov vklopljen, pomeni, da je bila pritisnjena ena od tipk. Zato smo na začetku programa definirali konstanto s kodo tipke 3. Če je v programu prebrana vrednost enaka vrednosti kode (Tipka3), potem je bila res pritisnjena tipka 3 matrične tipkovnice.
Zaradi motenj pri odbijanju kontaktov tipk matrične tipkovnice smo dodali podprogram za zakasnitev 20 ms. Dokler kontakta posamezne tipke nista zanesljivo sklenjena (kar po 20 ms zanesljivo sta) mikrokontroler pošljemo v podprogram za zakasnitev.

Povzetek

Na mikrokontroler smo priključili dvovrstični LCD-prikazovalnik, ki služi kot vmesnik med uporabnikom in elektronskim vezjem. Vsebuje lasten kontroler, ki skrbi za vklapljanje ustreznih

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

pik na zaslonu, medtem ko mi le pošiljamo ukaze na njegov vhod in tako določimo, kaj želimo na zaslonu videti. Deluje lahko v 8-bitnem in 4-bitnem načinu. Krmilimo ga tako, da mu po podatkovnih linijah pošiljamo podatke ali ukaze. Pred uporabo ga moramo inicializirati, saj v nasprotnem primeru ne bo deloval. Poleg nabora znakov, ki jih določi proizvajalec, lahko prikazuje tudi znake, ki jih izdelamo sami.
Ker pri krmilnih napravah navadno potrebujemo več tipk, smo na mikrokontroler priključili matrično tipkovnico, ki vsebuje 12 tipk, razporejenih v tri stolpce in štiri vrstice. Taka tipkovnica potrebuje 7 linij, da lahko z mikrokontrolerjem ugotovimo, katera od tipk je trenutno sklenjena.

Vaje

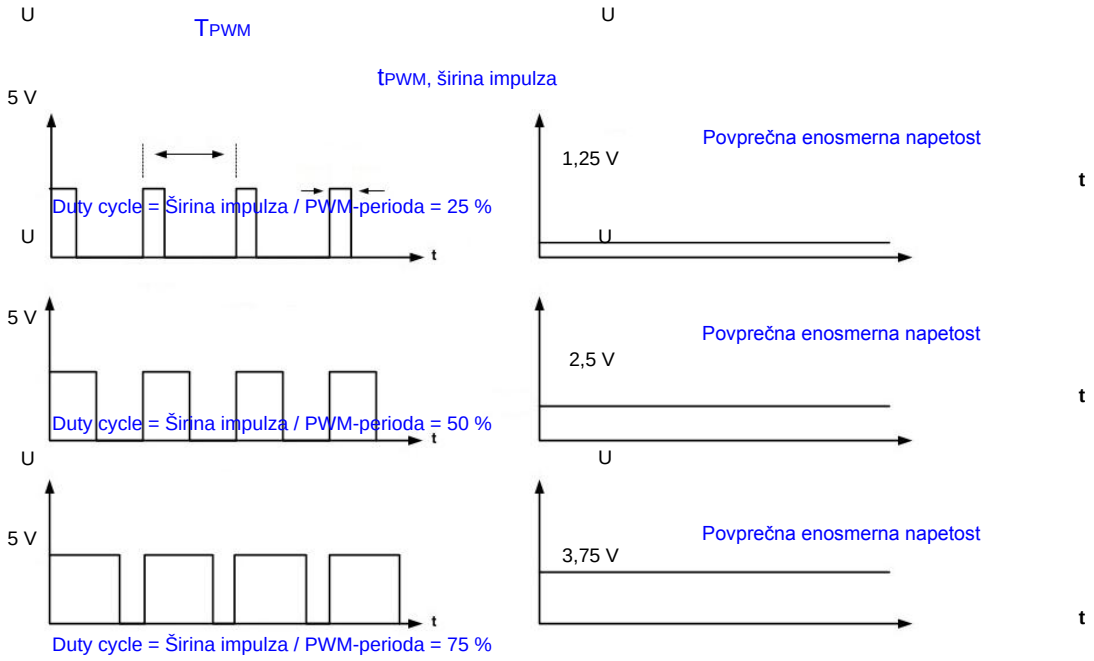
1. Program Prikaz napisa na LCD-ju spremeni tako, da bosta napisa v vrsticah zamenjana. V prvi vrstici naj bo prikazan napis *Varno voznjo*, v drugi vrstici pa *vam zelimo*. Sestavi vezje na preizkusni ploščici, program prenesi v mikrokontroler PIC16F628A in prikaži delovanje. Upoštevaj, da mikrokontrolerju določa takt kristalni oscilator 4 MHz.
2. Program Prikaz napisa na LCD-ju spremeni tako, da bo za celoten napis uporabljena le ena tabela. Delovanje preizkusi na vezju, ki si ga sestavil na preizkusni ploščici.
3. Na spletu poišči karakteristične podatke za LCD DEM 16216 SYH-PY. Iz kataloga ugotovi kode v CGROM-u za naslednje znake: Ω, μ, ?, F, v in V.
4. Program Prikaz napisa na LCD-ju spremeni tako, da bo napis v obeh vrsticah pomaknjen za dve mesti v desno. Delovanje preizkusi na vezju, ki si ga sestavil na preizkusni ploščici.
5. Program Prikaz napisa na LCD-ju spremeni tako, da bo na koncu besedila viden utripajoč kazalček. Pomagaj si z ukazi, napisanimi v kataloških podatkih za LCD. Delovanje preizkusi na vezju, ki si ga sestavil na preizkusni ploščici.
6. Izdelaj program, ki bo deloval po naslednjih zahtevah. S pritiskom na tipko T_1, povezano na priključek RA1, vklopimo napis v prvi vrstici LCD-ja, s pritiskom na tipko T_2, povezano na priključek RA2, pa vklopimo napis v drugi vrstici LCD-ja. Ko pritisnemo na tipko T_2, naj se napis v prvi vrstici izbriše. Po sklenitvi ponastavitvene tipke se napis izbriše. Vsebino napisov določi sam. Delovanje preizkusi na vezju, ki si ga sestavil na preizkusni ploščici.
7. Izdelaj program, ki bo deloval po naslednjih zahtevah. Po vklopu mikrokontrolerja naj LCD prikazuje napis *Pritisni na eno od tipk*. S pritiskom na tipko T_1, povezano na priključek RA1, vklopimo napis *Pritisnil si na desno tipko*. S pritiskom na tipko T_2, povezano na priključek RA1, pa vklopimo napis *Pritisnil si na levo tipko*. Delovanje preizkusi na vezju, ki si ga sestavil na preizkusni ploščici.
8. Izdelaj program, ki bo deloval po naslednjih zahtevah. Na priključek RA0 mikrokontrolerja ustrezno priključi vezje s fotouporom. Pri dnevni svetlobi naj bo na LCD-ju prikazan napis *Razsvetljava izklopljena*. Če fotoupor zatemnimo, se mora vklopiti svetleča dioda, povezana na priključek RB0, na LCD-ju pa naj bo prikazan napis *Razsvetljava vklopljena*. Delovanje preizkusi na vezju, ki si ga sestavil na preizkusni ploščici.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

9. Izdelaj dva lastna znaka velikosti 5 x 8 pik, ki naj ju prikazuje LCD.
10. Izdelaj program, ki bo vklopil svetlečo diodo, povezano na priključek RB0, samo takrat, ko pritisnemo na tipko 8 matrične tipkovnice. Po sprostitvi tipke naj se svetleča dioda izklopi. Delovanje preizkusi na vezju, ki si ga sestavil na preizkusni ploščici.
11. Program iz prejšnje vaje spremeni, da bo svetleča dioda ob sklenitve tipke 8 utripala s frekvenco ca. 2 Hz. Delovanje preizkusi na vezju, ki si ga sestavil na preizkusni ploščici.

PULZNO ŠIRINSKA MODULACIJA

Pulzno širinska modulacija (PWM – **P**ulse **w**idth **m**odulation) je tehnika nadzora oziroma krmiljenja energije, ki jo pošiljamo električnim porabnikom v obliki pravokotnih impulzov. Periodi pravokotnih impulzov bomo rekli PWM-perioda. Duty cycle, lahko bi ga poimenovali obratovalni ciklus, je razmerje med širino impulza in PWM-periodo. Povprečno vrednost enosmernega pulzirajočega signala lahko pri enaki frekvenci impulzov spreminjamo s spreminjanjem širine impulzov (slika 32).



Slika 32: Časovni diagram pulzno širinske modulacije

Če znaša napetost pravokotnih impulzov 5 V, lahko s spreminjanjem širine impulza in pri enaki frekvenci pulzirajočega signala dosežemo katerokoli napetost med 0 V in 5 V. S pulzno moduliranim signalom lahko npr. krmilimo hitrost vrtenja enosmernega motorja, določamo položaj in smer zasuka servomotorja ali pa krmilimo svetilnost svetlečih diod. Z mikrokontrolerjem PIC16F628A bomo generirali PWM-signal za krmiljenje svetilnosti svetleče diode. Njegov priključek RB3 ima dodatno ime CCP1, ki nam pove, da ima ta priključek poleg vhodno-izhodne funkcije še neko drugo. Pri pravilni nastavitvi mikrokontrolerja dobimo na tem priključku pulzno moduliran signal.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

```
;-------------------------------------------------------------------------------
; Pulzno širinska modulacija PWM
;Okolje MPLAB IDE 8.7, prevajalnik MPASM Assembler V5.39, oscilator 4 MHz
;Avtor: Milan Ivič, junij 2011
;-------------------------------------------------------------------------------

        list    p=16f628a               ;Tip mikrokontrolerja
        #include <p16f628a.inc>         ;Vključi v program datoteko p16f628a.inc.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _MCLRE_ON & _LVP_OFF & _XT_OSC


        org     0x000                   ;Ponastavitveni vektor
        goto    Glavni                  ;Nadaljuj na naslovu Glavni_zacetek.
        org     0x004                   ;Prekinitveni vektor

;********************** Glavni program **********************
Glavni
        bsf     STATUS,RP0              ;Banka 1
        bcf     TRISB,3                 ;Priključek RB3/CCP1 je izhod.
        movlw   .199                    ;Določimo frekvenco PWM-signala.
        movwf   PR2
        bcf     STATUS,RP0              ;Banka 0
        movlw   b'00001010'             ;Določimo širino impulzov PWM-signala (8 zgornjih bitov).
        movwf   CCPR1L
        movlw   b'00000100'             ;Vključimo TMR2 s postavitvijo tretjega bita registra T2CON na 1.
        movwf   T2CON
        movlw   b'00001100'             ;Vključimo PWM (tretji in četrti bit morata biti na 1).
        movwf   CCP1CON                 ;Peti in šesti bit (spodnja dva bita 10-bitnega podatka) sta 0.
Zanka
        goto    Zanka                   ;Neskončna zanka

        end
```

Za generiranje PWM-signala bomo uporabili časovnik TMR2, ki se nahaja v banki 0 in deluje skupaj z 8-bitnim registrom PR2. TMR2 je 8-bitni register in mu lahko dodelimo preddelitev 1 : 1, 1 : 4 ali 1 : 16. Njegovo delovanje vključimo s postavitvijo tretjega bita registra T2CON na 1, preddelitev pa mu določimo z določitvijo prvega in drugega bita v tem registru. Njegova vrednost se pri nastavljeni vrednosti preddelitve 1 : 1 povečuje s četrtino frekvence priključenega oscilatorja. Mi imamo postavljena ta dva bita na 0, zato je izbrana preddelitev 1 : 1. Časovnik TMR2 deluje z 8-bitnim registrom PR2. Frekvenca PWM-signala je odvisna od hitrosti povečevanja časovnika TMR2 in od nastavljene vrednosti registra PR2. Ko TMR2 doseže vrednost, ki je nastavljena v registru PR2, se ponastavi in začne ponovno naraščati od vrednosti 0. Čas trajanja ene periode PWM-signala izračunamo po enačbi 2.

$$T_{PWM} = [(PR2) + 1] \cdot 4 \cdot T_{osc} \cdot TMR2_{faktor\ preddelitve}$$

Enačba 2: Izračun periode PWM-signala

Pri priključenem 4 MHz oscilatorju, nastavljeni vrednosti PR2 = 199 in izbrani preddelitvi 1 : 1 znaša ta čas T_{PWM} = 200 µs. Frekvenca PWM signala, ki ga dobimo na priključku RB3, znaša:

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

$$f_{PWM} = \frac{1}{T_{PWM}} = 5\ kHz$$

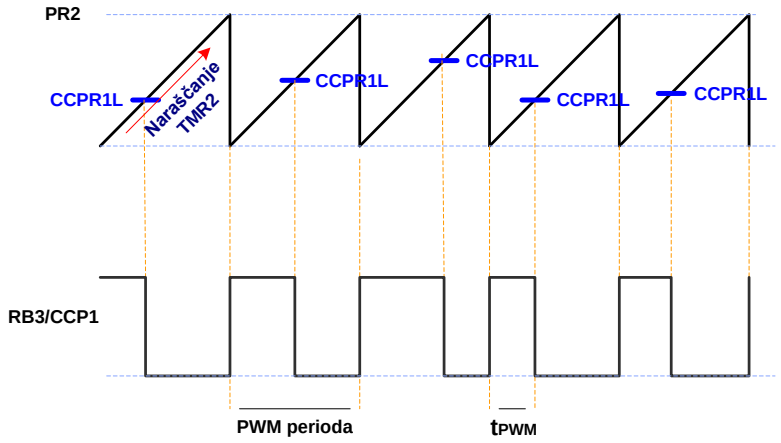
Enačba 3: Izračun frekvence PWM-signala

Z nastavitvijo mikrokontrolerja smo v našem programu določili čas trajanja periode PWM-signala, ki znaša 200 µs. Kako pa določimo širino impulza (t_{PWM}), ki lahko v našem primeru traja od 0 do 200 µs? Širina impulza je določena z vrednostjo vseh osmih bitov registra CCPR1L ter vrednostjo petega in šestega bita registra CCP1CON. Na voljo imamo torej 10 bitov, izmed katerih je 8 zgornjih bitov v registru CCPR1L in dva spodnja v registru CCP1CON (peti in šesti bit). V našem programu smo določili vrednost CCPR1L, ki znaša $00001010_{(2)}$, vrednosti petega in šestega bita registra CCP1CON pa sta $00_{(2)}$. Vseh 10 bitov ima torej vrednost $0000101000_{(2)}$ ali 40, če pretvorimo v desetiško število.

$$t_{PWM} = (CCPR1L_{8\ bitov}\ ter\ CCP1CON_{2\ bita}) \cdot t_{osc} \cdot TMR2_{faktor\ preddelitve}$$

Enačba 4: Izračun širine PWM-impulzov

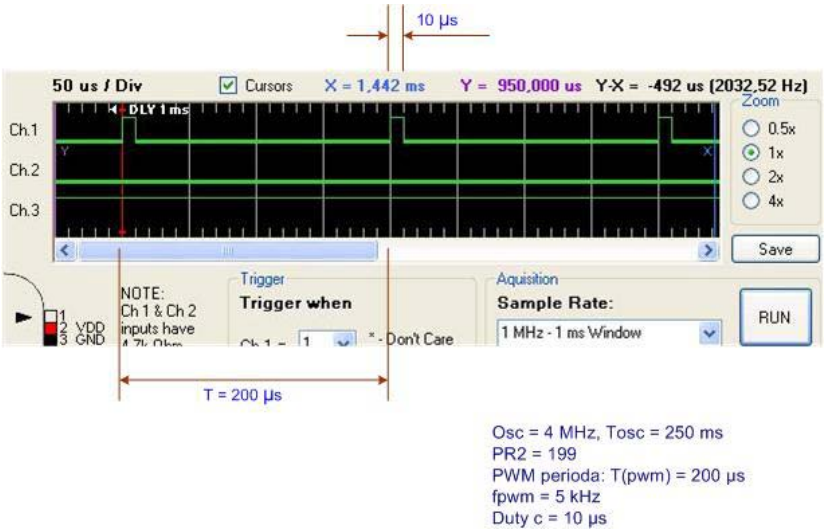
Čas trajanja vsakega impulza pulzno moduliranega signala, ki ga dobimo na priključku RB3, izračunanega po enačbi 4, znaša 10 µs.



Slika 33: Prikaz PWM-izhoda

Na sliki 33 vidimo prikaz delovanja PWM v mikrokontrolerju PIC16F628A. Hitrost naraščanja vrednosti TMR2 je odvisna od oscilatorja, ki daje takt mikrokontrolerju, in od preddelilnika, ki mu je dodeljen. Ko doseže nastavljeno vrednost registra PR2, se ponastavi, hkrati vklopi priključek RB3/CCP1, njegova vrednost pa začne ponovno naraščati od 0. Mikrokontroler stalno preverja vrednost TMR2 z vrednostjo registra CCPR1L (dejansko tudi z vrednostjo dveh spodnjih bitov, ki sta v registru CCP1CON). Ko TMR2 doseže vrednost registra CCPR1L, mikrokontroler izklopi priključek RB3/CCP1. Na koncu poglejmo še oscilogram, ki smo ga dobili kar s programatorjem PICkit 2. Slika 44 prikazuje PWM-signal na priključku RB3/CCP1, generiranem na podlagi našega programa.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.



Slika 34: Oscilogram PWM-izhoda za program Pulzno širinska modulacija PWM

Povzetek

Z mikrokontrolerjem PIC16F628A lahko generiramo pulzno širinsko modulirani signal (PWM). Z njim lahko krmilimo energijo, ki jo pošiljamo električnim porabnikom. Pri enaki frekvenci spreminjamo širino impulzov, od katerih je odvisna povprečna vrednost enosmerne napetosti. Spoznali smo registre mikrokontrolerja, ki vplivajo na PWM-način delovanja. Na podlagi izdelanega programa in programatorja PICkit 2 smo izračunali in izmerili PWM-signal, dobljen na ustreznem izhodnem priključku mikrokontrolerja.

Vaje

1. Program Pulzno širinska modulacija PWM spremeni, da bo frekvenca PWM-signala na priključku RB3/CCP1 znašala 20 kHz. Program prenesi v mikrokontroler PIC16F628A in z osciloskopom izmeri frekvenco PWM-signala. Mikrokontrolerju priključi oscilator vrednosti 4 MHz.
2. Koliko znašata najmanjša in koliko največja dobljena frekvenca PWM-signala, če na mikrokontroler priključimo oscilator 4 MHz? Koliko znašata ti mejni frekvenci, če na mikrokontroler priključimo oscilator 20 MHz? Upoštevaj faktor preddelilnika.
3. V programu Pulzno širinska modulacija PWM spremeni vrednost registra CCPR1L na 254 desetiško. Izračunaj čas trajanja enega impulza PWM-signala. Program prenesi v mikrokontroler PIC16F628A, priključi oscilator 4 MHz in z osciloskopom izmeri ta čas. Izmeri čas tudi s programatorjem PICkit 2. Primerjaj rezultata, dobljena z različnima vrednostma registra CCPR1L. Na priključek RB3 poveži svetlečo diodo z ustreznim zaščitnim uporom. Primerjaj svetilnost svetleče diode pred spremembo vrednosti registra CCPR1L in po njej. Komentiraj dobljeni rezultat.

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.

4. Želimo, da svetleča dioda, priključena na RB3, počasi spreminja svetilnost od najmanjše do največje. Kaj bi morali v programu spremeniti?
5. Servomotor RS-2 ima to lastnost, da se zavrti v smeri urnega kazalca, če ga krmilimo z impulzi širine od 0,7 ms do 1 ms. Če ga krmilimo z impulzi širine od 1,7 ms do 2 ms, se bo zavrtel v nasprotno smer, pri impulzih širine 1,5 ms pa bo v izhodiščnem položaju. Ali je možno krmiliti ta servomotor z mikrokontrolerjem PIC16F628A?
6. Izdelaj program, ki bo spreminjal svetilnost petim svetlečim diodam. Od najmanjše do največje svetilnosti naj preteče največ 10 sekund. Postopek naj se neprestano ponavlja. Komentiraj program, ki si ga izdelal. Uporabi mikrokontroler PIC16F628A, oscilator izberi sam.

VIRI

Mikeln, J. (2004). *Programirajmo mikrokontrolerje*. Ljubljana: AX elektronika, d. o. o.

Splet: Pridobljeno 2. 6. 2011 iz http://www.jaycar.com.au/images_uploaded/40044D.pdf

Splet: Pridobljeno 2. 6. 2011 iz http://www.microchip.com/

Splet: Pridobljeno 2. 6. 2011 iz http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805

Splet: Pridobljeno 17. 6. 2011 iz http://pdf1.alldatasheet.com/datasheet-pdf/view/110299/ETC/DEM16216SYH-PY.html

Učno gradivo je nastalo v okviru projekta Munus 2. Njegovo izdajo je omogočilo sofinanciranje Evropskega socialnega sklada Evropske unije in Ministrstva za šolstvo in šport.