MIKRORAČUNARSKI SISTEMI
II deo računskih vežbi

Asistenti:

Vladimir Simić

Nenad Petrovic

Uvod

- PIC (***P**eripheral **I**nterface **C**ontroller*) je familija mikrokontrolera razvijena od strane kompanije Microchip Technology
- Naziv su dobili po modelu PIC1650, razvijenog sredinom sedamdesetih godina proslog veka
- Kasnije, promenjeno značenje u ***P**rogrammable **I**ntelligent **C**omputer*
- Karakteristike:
  - niska cena
  - dobra podrška
  - velika zajednica korisnika

Istorija

- Nastao za potrebom da se odvoje I/O operacije od CPU 1975. godine
- Originalno je bio 8-bit
- Kreiran da radi zajedno sa 16-bit CPU – CP1600
- Koristio ROM memoriju za čuvanje koda
- Jedan od prvih RISC-baziranih uređaja
- 1993. se javlja PIC16C84, prvi sa EEPROM
- 2001. se javljaja PIC sa flash memorijom
- Do 2013. godine je u upotrebi preko 12 milijardi primeraka mikrokontrolera iz ove familije

Familije PIC

- Različte širine instrukcija i memorije podataka
- Memorija podataka
  - 8, 16, 32-bit
- Instrukcije
  - 12, 14, 16 i 24-bit
- Različite dubine poziva steka
- Različit broj U/I portova
- Različit broj registara i memorijskih reči
- Naprednije verzije poseduju dodatne module
  - Serijska komunikacija: USART, I2C, SPI, USB
  - ADC/DAC
  - Ethernet

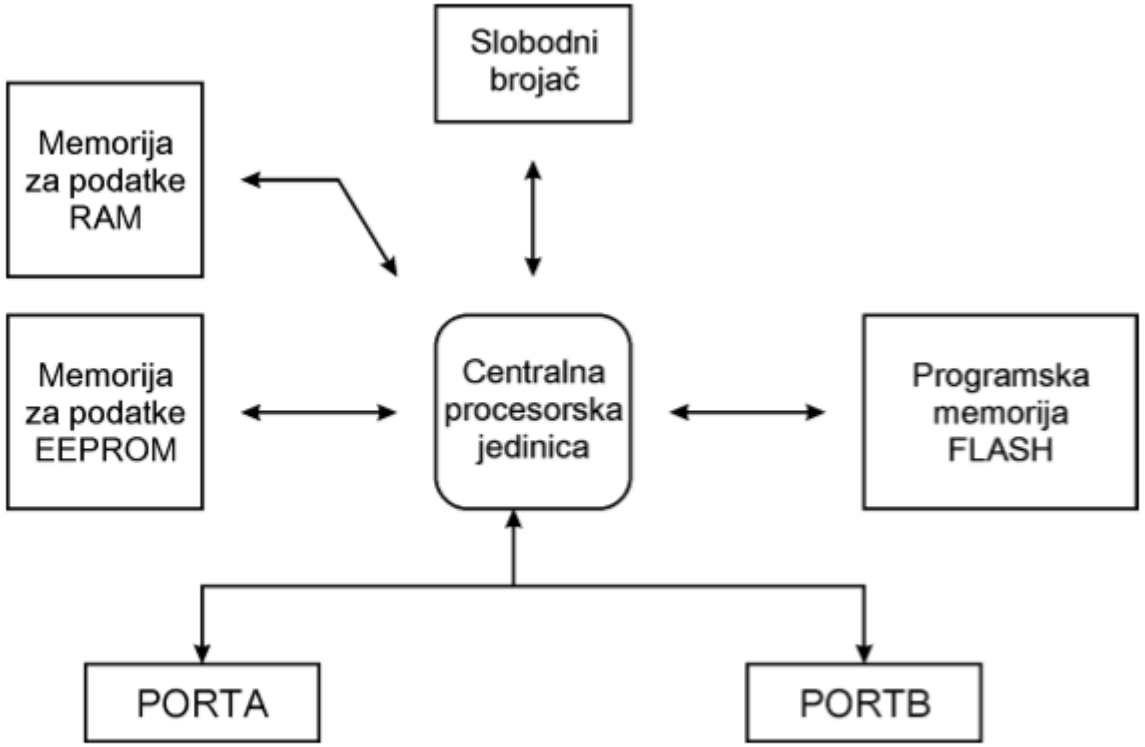
Pregled PIC familija

- PIC10 i PIC12
  - 12bit instrukcije, 2 nivoa steka, nema HW mul/div
- PIC16
  - 14bit instrukcije, 8 nivoa steka
- PIC17
  - 16bit instrukcije, 16 nivoa steka, HW množenje (8b x 8b)
- PIC18
  - 16bit instrukcije, 31 nivo steka, rad sa USB
- PIC24
  - 24bit
  - Hardversko množenje u 1 ciklusu (16b x 16b)
  - deljenje u 19 ciklusa (32b / 16b)
  - DSP operacije
  - DMA pristup periferijama
- PIC32
  - MIPS-baziran, USB OTG, Ethernet

Razvojna okruženja za PIC

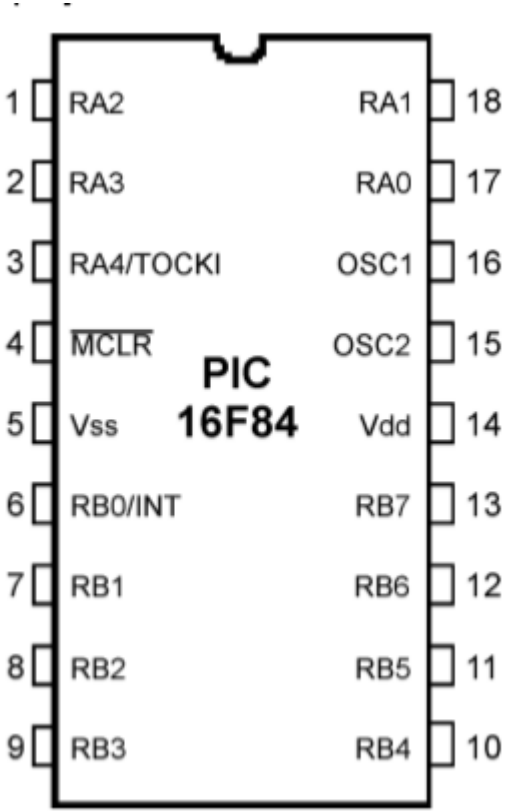
- Freeware MPLAB X razvojno okruženje
  - Razvijeno od strane proizvođača
  - Integriše assembler, linker, softverski simulator i debugger
- Različite mogućnosti pisanja koda
  - asembler
  - C (XC8, MikroC)
  - C++
  - Basic
  - Pascal
  - PICBASIC

Blok šema PIC16F84

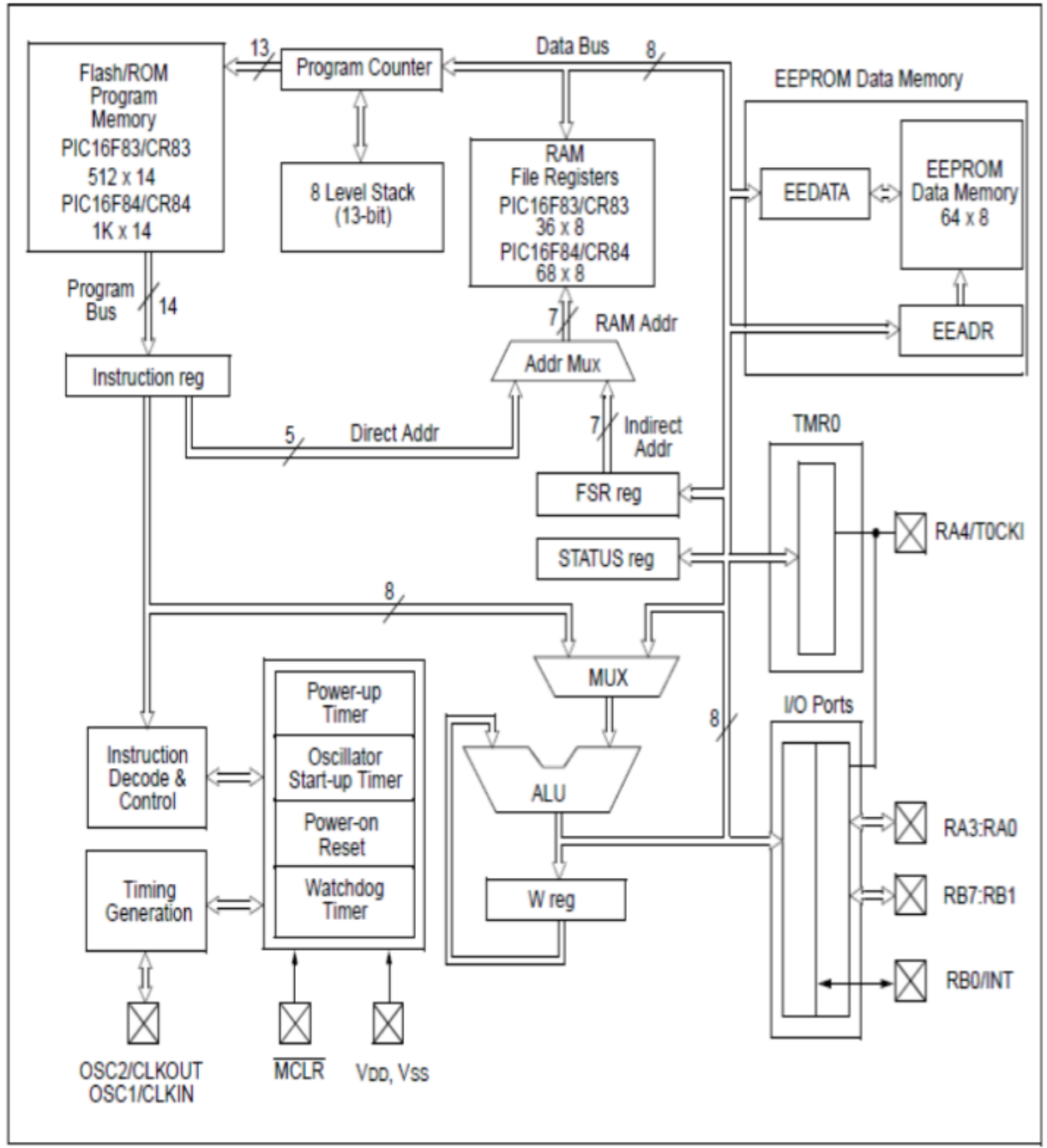


Sl. 2.1 Blok šema mikrokontrolera PIC16F84

Pinovi PIC16F84

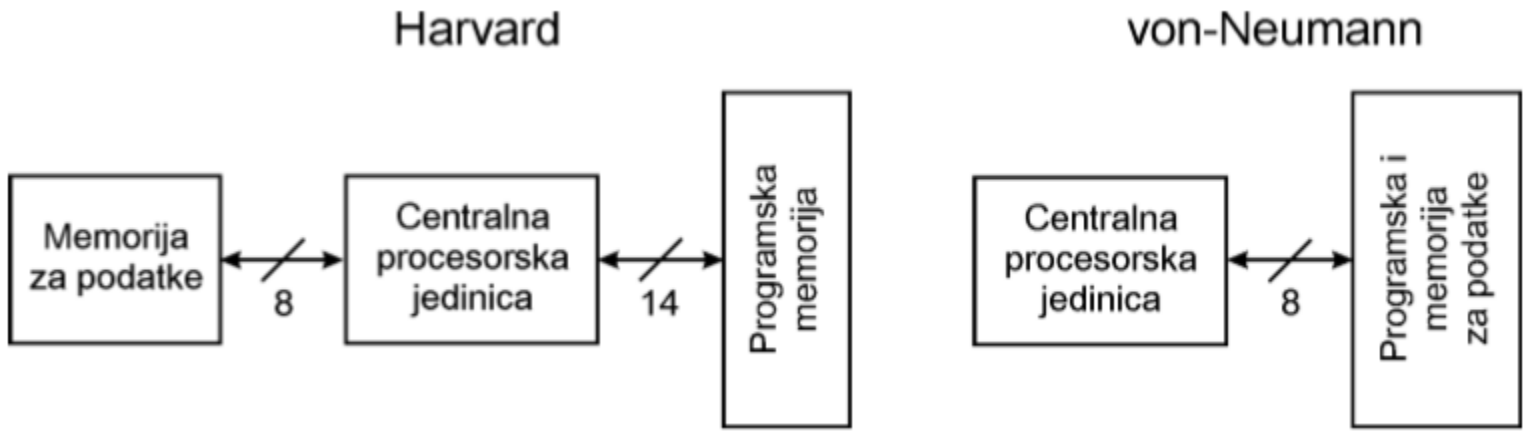


Pin br. 1	**RA2**	Drugi pin porta A. Nema nikakvu dodatnu ulogu.
Pin br. 2	**RA3**	Treći pin porta A. Nema nikakvu dodatnu ulogu.
Pin br. 3	**RA4**	Četvrti pin porta A. Na njemu se takođe nalazi i TOCK1 koji ima tajmersku f-ju.
Pin br. 4	**MCLR**	Reset ulaz i Vpp napon programiranja mikrokontrolera
Pin br. 5	**Vss**	Napajanje, masa.
Pin br. 6	**RBO**	Nulti pin porta B. Dodatna funkcija je interaptni ulaz.
Pin br. 7	**RB1**	Prvi pin porta B. Nema nikakvu dodatnu ulogu.
Pin br. 8	**RB2**	Drigi pin porta B. Nema nikakvu dodatnu ulogu.
Pin br. 9	**RB3**	Treći pin porta B. Nema nikakvu dodatnu ulogu.
Pin br. 10	**RB4**	Četvrti pin porta B. Nema nikakvu dodatnu ulogu.
Pin br. 11	**RB5**	Peti pin porta B. Nema nikakvu dodatnu ulogu.
Pin br. 12	**RB6**	Šesti pin porta B. 'Clock' linija u programskom modu.
Pin br. 13	**RB7**	Sedmi pin porta B. 'Data' linija u programskom modu.
Pin br. 14	**Vdd**	Pozitivan pol napajanja.
Pin br. 15	**OSC2**	Pin namenjen spajanju sa oscilatorom.
Pin br. 16	**OSC1**	Pin namenjen spajanju sa oscilatorom.
Pin br. 17	**RAO**	Drugi pin prta A. Nema nikakvu dodatnu ulogu.
Pin br. 18	**RA1**	Prvi pin porta A. Nema nikakvu dodatnu ulogu.


Flash/ROM
Program
Memory
PIC16F83/CR83
512 x 14
PIC16F84/CR84
1K x 14
13
Program Counter
Data Bus
8
EEPROM Data Memory
8 Level Stack
(13-bit)
RAM
File Registers
PIC16F83/CR83
36 x 8
PIC16F84/CR84
68 x 8
EEDATA
EEPROM
Data Memory
64 x 8
EEADR
Program
Bus
14
7
RAM Addr
Addr Mux
Instruction reg
5
Direct Addr
7
Indirect
Addr
TMR0
FSR reg
RA4/T0CKI
STATUS reg
8
MUX
I/O Ports
Power-up
Timer
Instruction
Decode &
Control
Oscillator
Start-up Timer
ALU
8
Power-on
Reset
RA3:RA0
Timing
Generation
Watchdog
Timer
W reg
RB7:RB1
RB0/INT
OSC2/CLKOUT
OSC1/CLKIN
MCLR
VDD, VSS

Harvard RISC arhitektura

- Harvard arhitektura omougćava istovremeno čitanje instrukcije i čitanje/upis podataka

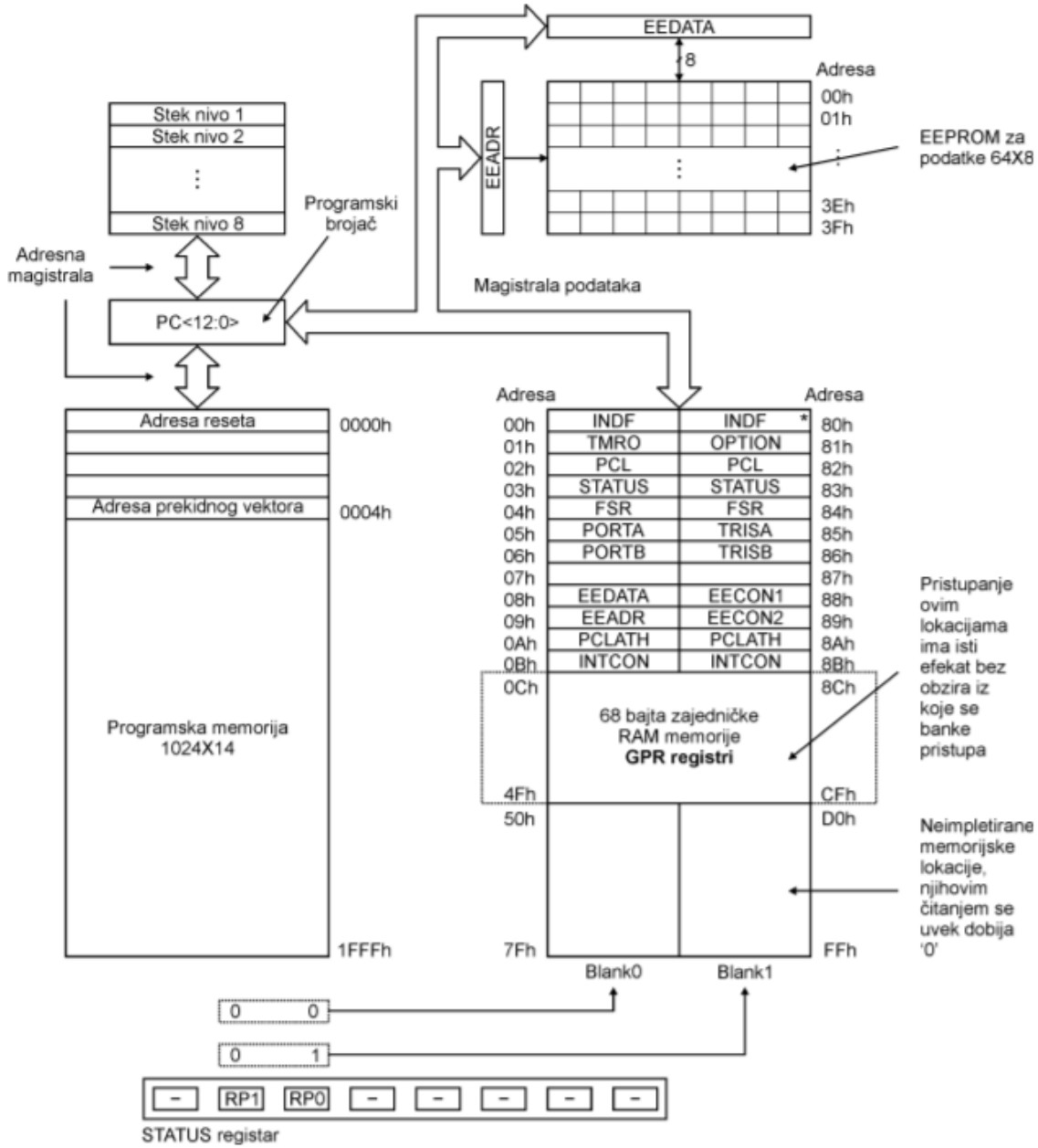


Primene PIC16F84

- Elektronske brave
- Sigurnosni uređaji
- Automobilska industrija
- Kućni aparati

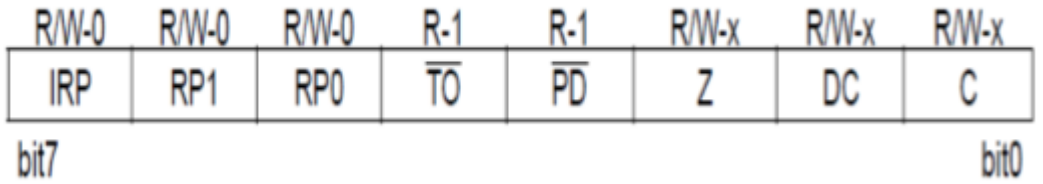
Organizacija memorije

- Programski blok
  - FLASH :1Kx14b
- Blok za podatke
  - EEPROM: 64x8b
  - RAM
    - SFR (registri specijalne namene):
      - Dve memorijske banke
      - prvih 12 lokacija u banci 0 i 1
    - GPR (registri opšte namene): 68x8b lokacija, pristup nezavisno od banke



Sl. 2.15. Memorijska organizacija mikrokontrolera 16F84

Statusni registar



bit 7	Ne koristi se
bit 6-5	bitovi za selekciju banke kod direktnog adresiranja RP0 = 0 BANKA0 (00h – 7Fh) RP0 = 1 BANKA1 (80h – FFh) bit 6 (RP1) se ne koristi kod 16F8X
bit 4	TO: (time out) 1=nakon dovođenja napajanja, CLRWDT ili SLEEP naredbe 0=dogodio se time out reset od WDT (watch dog timer)
bit 3	PD: (power down) 1=nakon dovođenja napajanja ili CLRWDT naredbe 0=nakon izvršenja SLEEP naredbe
bit 2	Z:(zero) indikator (flag) za '0' 1=rezultat aritmetičke ili logičke operacije je 0 0= rezultat aritmetičke ili logičke operacije nije 0
bit 1	DC: Digital carry (decimalni prenos ili pozajmica kod BCD operacija) 1=biloje prenosa iz niže u višu tetradu 0= nije biloje prenosa iz niže u višu tetradu
bit 0	C: (carry/borrow) prenos ili pozajmica iz bit naviše pozicije 1=dogodio se prenos iz bita najviše pozicije 0=nije bilo prenosa iz bita najviše pozicije **Napomena:** kod oduzimanja je vrednost pozajmice komplemnirana, tj bit je 1 kad nije bilo pozajmice

Vrste prekida kod PIC16F84

PORTB

- Spoljašnji prekid na RB0/INT pinu mikrokontrolera
- Prekid prilikom promene na pinovima 4, 5, 6 i 7 porta B
- Prekid prilikom prekoračenja TMR0 brojača
- Prekid prilikom završetka upisivanja u EEPROM

INTCON registar

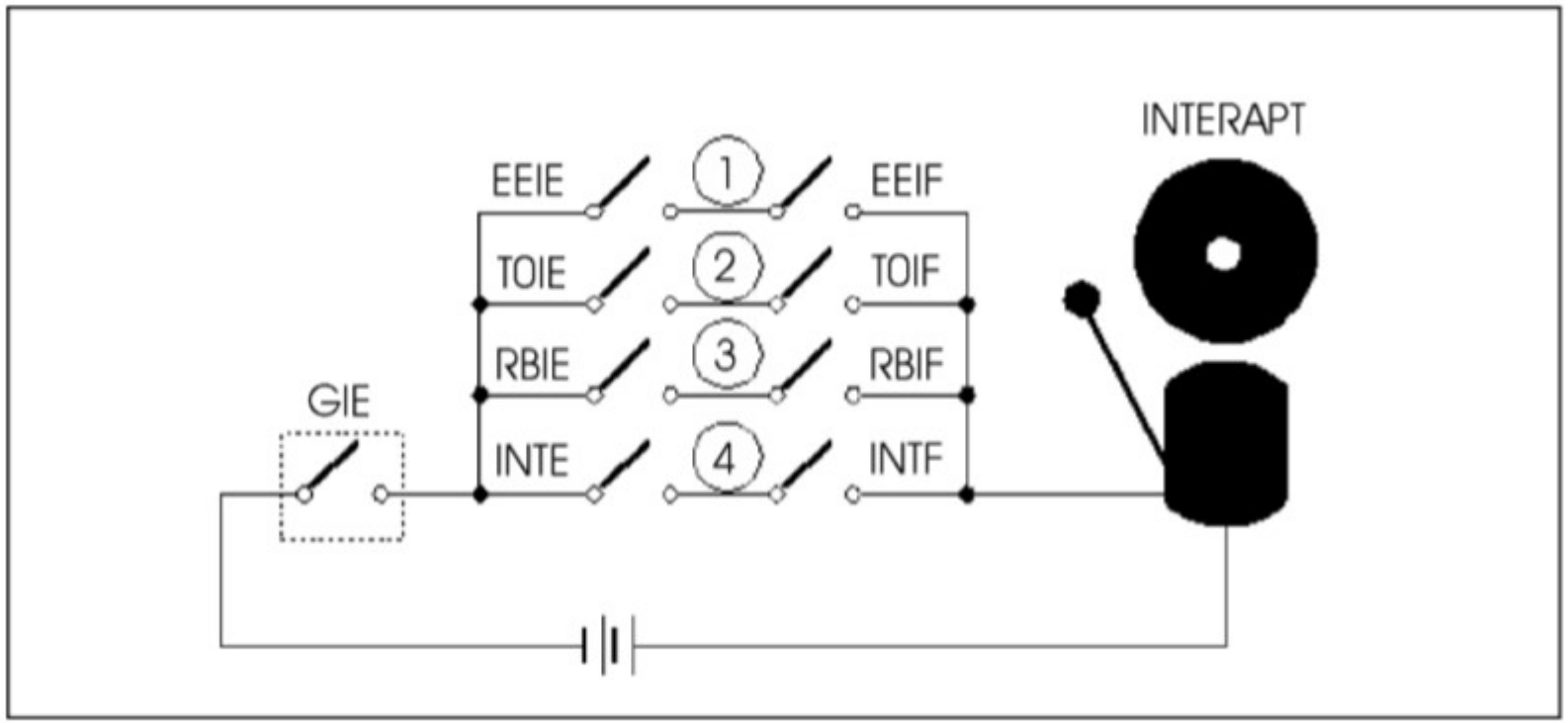
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

LEGENDA:

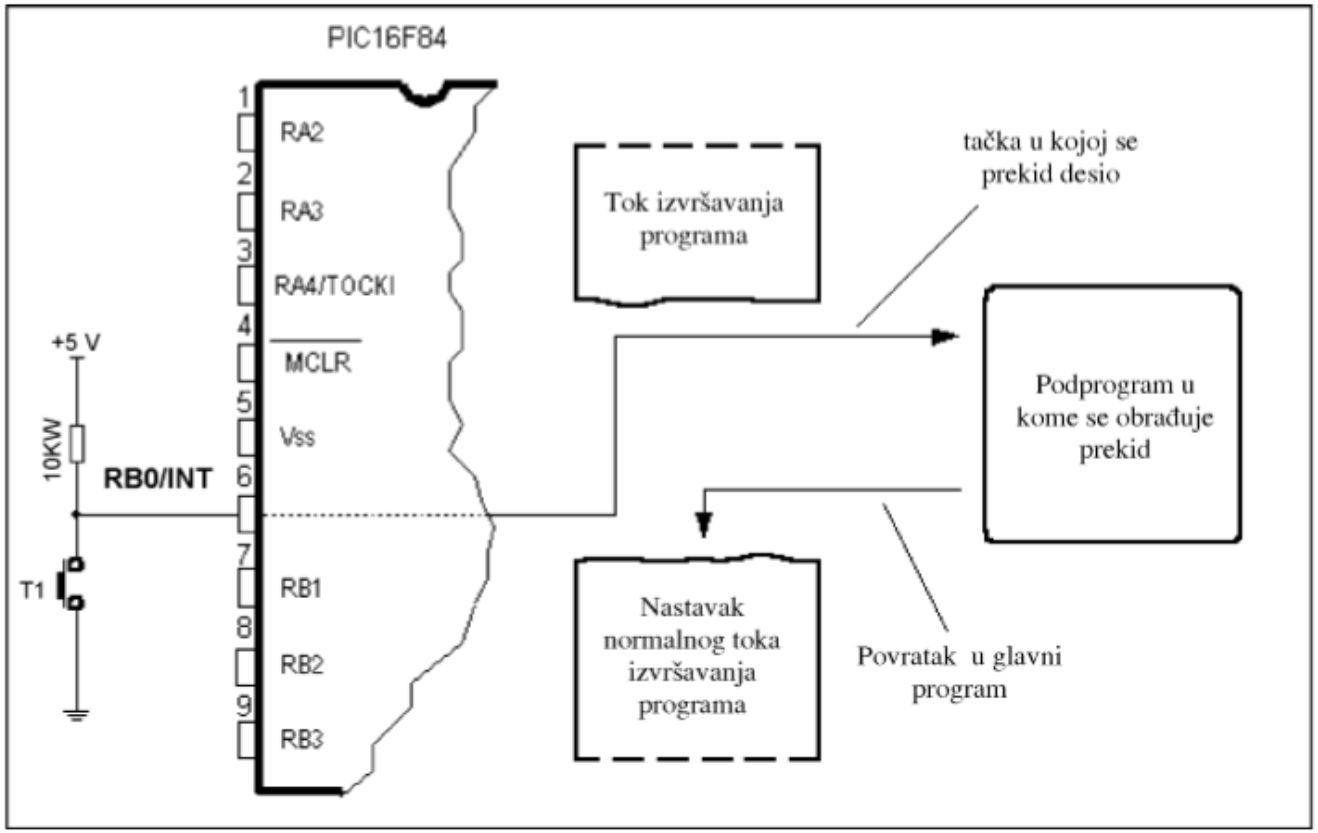
R = bit koji se može čitati
U = neiskorišćen bit, čita se kao "0"
W = bit koji se može upisati
n = vrednost posle reseta

bit 7	**GIE:** bit globalne dozvole prekida 1=dozvoljeni su svi prekidi 0=zabranjeni su sviprekidi
bit 6	**EEIE** dozvola prekida na završetku upisa u data EEPROM 1=dozvoljen prekid na završetku upisa u data EEPROM 0=zabranjen prekid na završetku upisa u data EEPROM
bit 5	**T0IE:** dozvola prekida od tajmerao (TMR0) 1=dozvoljen prekid od TMR0 0=zabranjen prekid od TMR0
bit 4	**INTE:** dozola prekida na RB0/INT priključku 1=dozoljen prekid na RB0/INT priključku 0=zabranjen prekid na RB0/INT priključku
bit 3	**RBIE:** dozvola prekida na promenu na priključcima RB4-RB7 1=dozvoljen prekid na promenu na priključcima RB4-RB7 0=zabranjen prekid na promenu na priključcima RB4-RB7
bit 2	**T0IF:** indikator prekida od TMR0 1=dogodio se prelaz TMR0 a 255 na 0 0=nije se dogodio prelaz TMR0 a 255 na 0
bit 1	**INTF:** indikator prekida na RB0/INT priključku 1=dogodio se prekid na RB0/INT priključku 0=nije se dogodio prekid na RB0/INT priključku
bit 0	**RBIF:** indikator prekida na promenu na priključcima RB4-RB7 1=dogodio se prekid na na promenu na priključcima RB4-RB7 0=nije se dogodio prekid na na promenu na priključcima RB4-RB7

Šema prekida



Prekidi


PIC16F84
1
RA2
2
RA3
3
RA4/T0CKI
4
MCLR
5
Vss
6
RB0/INT
7
RB1
8
RB2
9
RB3
+5 V
10KW
T1
Tok izvršavanja programa
tačka u kojoj se prekid desio
Podprogram u kome se obrađuje prekid
Nastavak normalnog toka izvršavanja programa
Povratak u glavni program

Struktura prekidne rutine

- 1. Čuva se W registar bez obzira na tekuću banku
- 2. Čuva se STATUS registar u bank0.
- 3. Čuvaju se ostali registri
- [Test interrput flegova]
- [Izvršava se prekidna rutina za obradu detektovanog prekida (ISR)]
- [Čišćenje flega obrađenog prekida]
- 4. Restaurišu se ostali registri
- 5. Restauriše se STATUS registar
- 6. Restauriše se W registar

Option registar

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	TOSE	PSA	PS2	PS1	PS0
							bit 0

bit 7	**Not RBPU**: Otpornici vezani na Vcc uključeni/isključeni 0=otpornici na PORTB uključeni 1= otpornici na PORTB isključeni
bit 6	**INTEDG**: ivica za spoljašnji prekid 1= prekid na rastućoj ivici signala na RB0/INT priključku 0= prekid na opadajućoj ivici signala na RB0/INT priključku
bit 5	**T0CS**: izbor takta za TMR0 1= sa RA4/T0CKI priključka 0= interni takt kojim se izvršavaju naredbe (CLKOUT)
bit 4	**T0SE**: izbor ivice takta za TMR0 1 = Uvećava se na prelazu sa visoko na nisko na RA4/T0CKI priključku 0 = Uvećava se na prelazu sa nisko na visoko na RA4/T0CKI priključku
bit 3	**PSA**: dodela preskalera 1= preskaler je dodeljen WDT tajmeru 0= preskaler je dodeljen TMR0
bit 2-0	PS2:PS0: bitovi za izbor odnosa deljenja preskalera

PS2:PS0	TMR0 učestanost	WDT učestanost
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Preskaler

- Preskaler je naziv za deo mikrokontrolera koji deli instrukcijski ciklus pre nego što on dođe do logike koja povećava stanje brojača. Ovim se dobija mogućnost merenja dužih vremenskih perioda
- Preskaler se može pridružiti jednom od dva brojača pomoću PSA bita
  - TMR0
  - Watchdog timer
- PS2,PS1,PS0 definišu vrednost preskalera

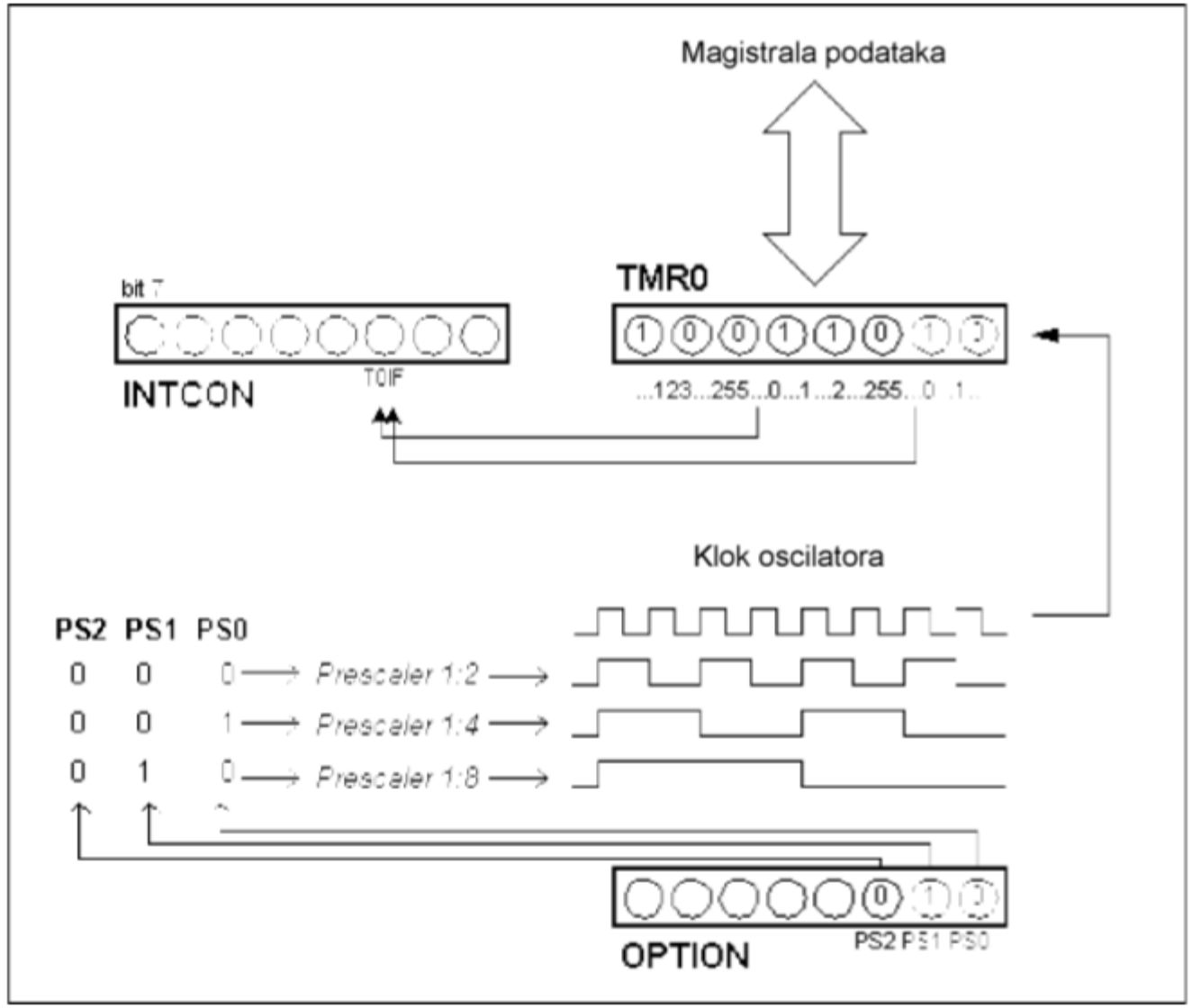
Bitovi	TMR0	WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:18
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

TMR0 (Slobodni brojač)

- Pomoću njih se stvara relacije između realne veličine kao što je vreme, sa promenljivom koja predstavlja stanje brojača unutar mikrokontrolera
- PIC16F84 ima 8-bitni brojač
- Nakon svakog odbrojavnja do 255 brojač resetuje svoju vrednost na nulu i kreće sa novim ciklusom brojanja do 255. Prilikom svakog prelaska sa 255 na nulu, setuje se bit T0IF u INTCON registru. Ukoliko je dozvoljena pojava prekida, ovo se može iskoristiti za generisanje prekida i obradu prekidne rutine.
- Na programeru je da resetuje bit T0IF u prekidnoj rutini, kako bi se mogao detektovati novi prekid, tj. novo prekoračenje

Watchdog timer (Sigurnosni brojač)

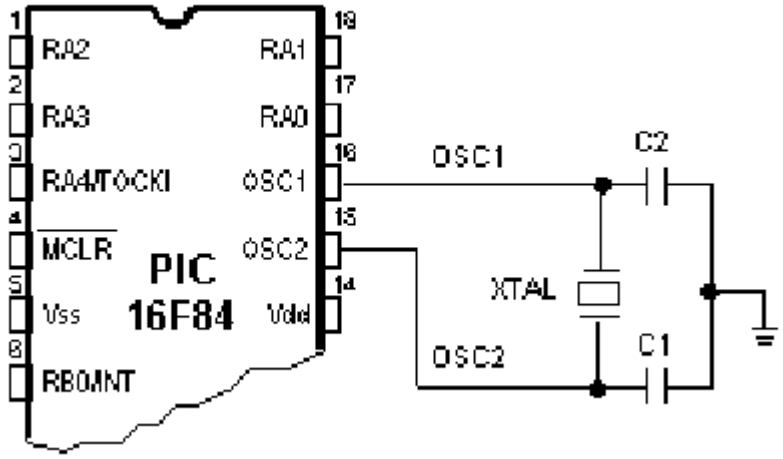
- Mehanizam mikrokontrolera kojim se mikrokontroler brani od zaglavljivanja programa. Kao i kod svakog elektronskog kola, i kod mikrokontrolera može doći do kvara ili nepravilnog rada.
- Kada se to desi mikrokontroler će stati sa radom i ostati u tom stanju sve dok ga neko ne resetuje.
- Zbog toga je uveden mehanizam watchdog koji, nakon određenog vremena, resetuje mikorokontroler (mikrokontroler sam sebe resetuje).
- Watchdog radi na jednostavnom principu: ako doše do prekoračenja brojača mikrokontroler se resetuje i kreće sa izvršavanjem programa iz početka.
- Sledeći korak je sprečavanje reseta u slučaju pravilnog rada, što se postiže upisom nule u WDT registar (instrukcijom CLRWDT) svaki put kada se približi svom prekoračenju, čime će program sprečiti reset sve dok se pravilno izvršava.
- Jednom, kada dođe do zaglavljivanja, nula neće biti upisana, doći će do prekoračenja WDT brojača i desiće se reset koji će mikorokontroler ponovo vratiti u pravilan rad.



Sl. 2.21. Odnos brojača TMR0 i preskalera

Oscilatori

- Procesorski takt predstavlja ¼ špoljašnjeg takta oscilatora
- Najčešće se koriste dve vrste oscilatora
  - Kristalni oscilator (XT, HS, LP)
  - Sa otpornikom i kondenzatorom (RC)



Primer podešavanja preskalera

- Neka je radni takt mikrokontrolera 3.2768Mhz. Potrebno je podesiti vrednost OPTION registra tako da se svaki od displej osvežava frekvencijom od po 200Hz, pri čemu imamo 2 displeja.

$$\frac{2^{15}\times 100\,Hz}{2^{2}\times 2^{8}\times 2^{n+1}} = 400\,Hz, \;\; => n \; = 2$$

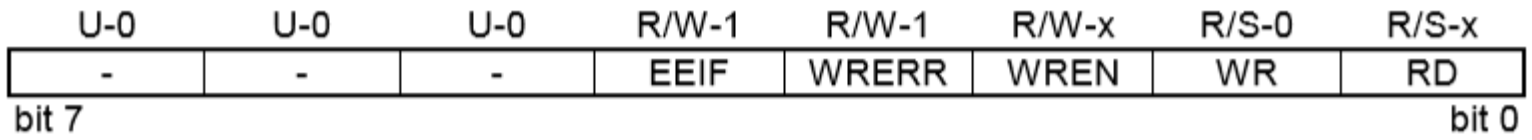
```
OPTION_REG = 1XX0 0010 = 1000 0010; (nisu uključeni otpornici na PORTB)
MOVLW 0x82
                MOVWF OPTION_REG  ; kraj primera
```

EEPROM memorija

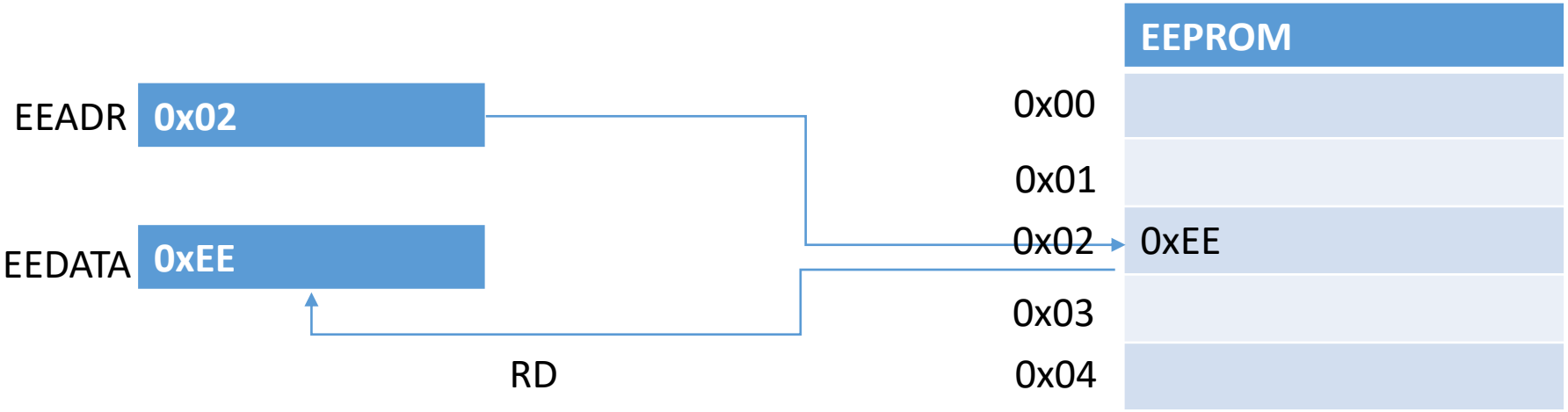
- EEPROM memorija se nalazi u posebnom memorijskom prostoru i pristupa joj se preko specijalnih registara.
- Registri za rad sa EEPROM memorijom su:
  - EEDATA
    - sadrži podatak koji je pročitan ili koga treba upisati.
  - EEADR
    - sadrži adresu EEPROM lokacije kojoj se pristupa.
  - EECON1
    - sadrži kontrolne bitove.
  - EECON2
    - ovaj registar ne postoji fizički i služi da zaštiti EEPROM od slučajnog upisa.

EECON1 registar

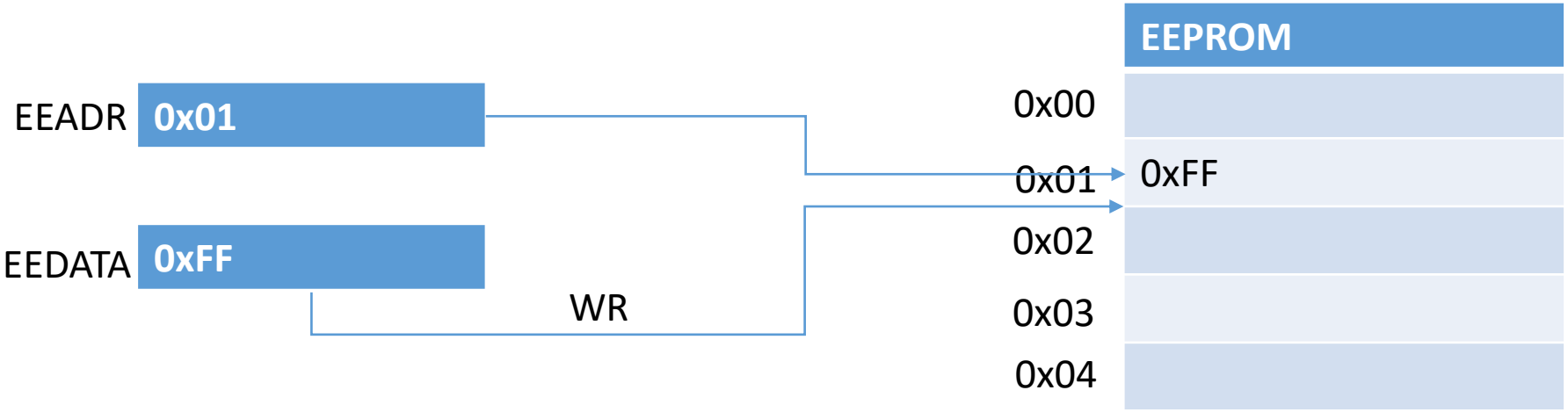
- EEIF
  - Interrupt flag za detekciju prekid nakon upisa u EEPROM
- WRERR
  - Greška pri upisu
- WREN
  - Omogućiti upis
- WR
  - Izvršiti upis
- RD
  - Izvršiti čitanje



Čitanje EEPROM memorije



Upis u EEPROM memoriju



Prekid prilikom završetka upisa u EEPROM

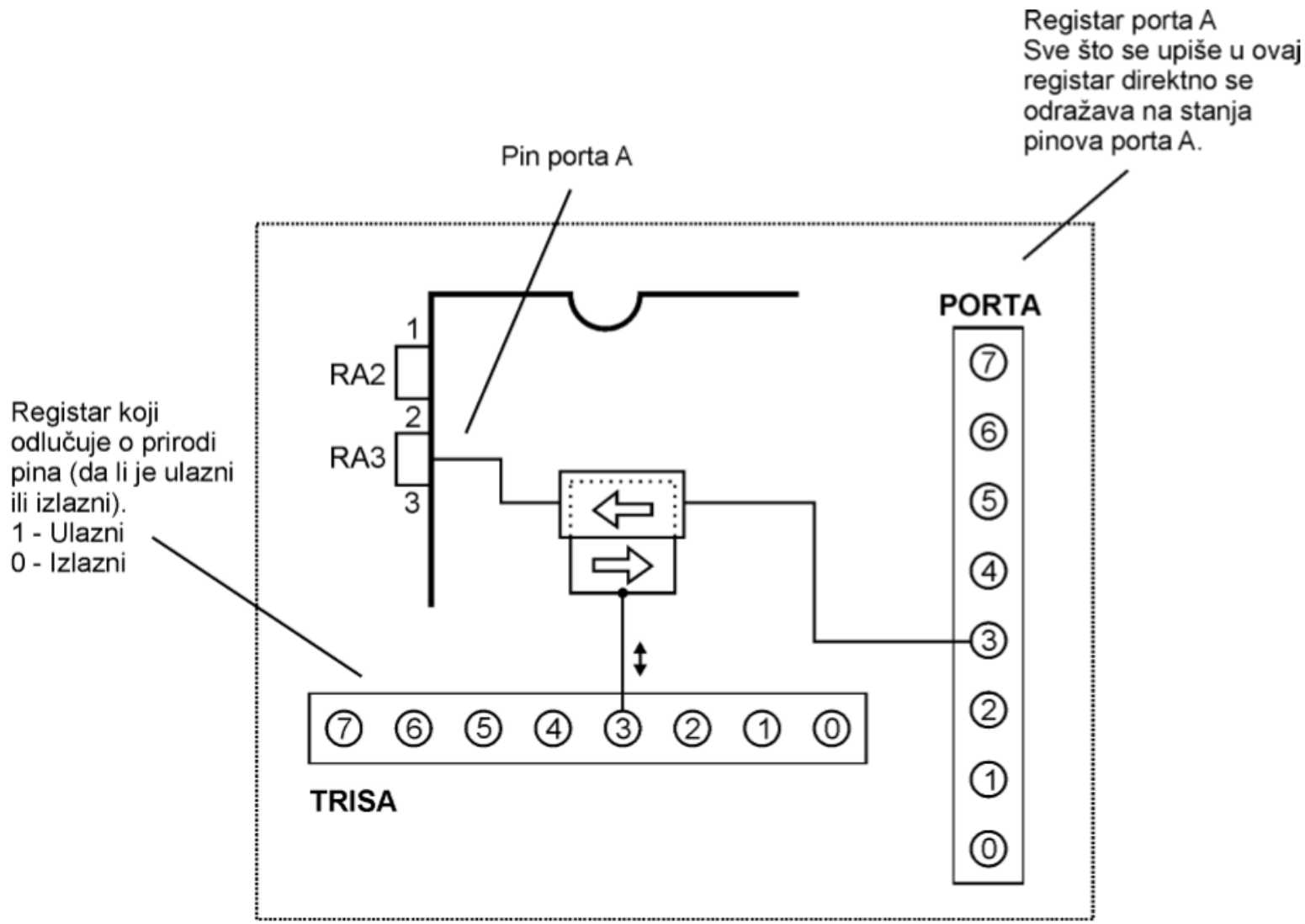
- Prekid može biti uključen/isključen setovanjem/resetovanjem EEIE bita u INTCON registru.
- Ovaj prekid je čisto praktične prirode. Kako upis u jednu lokaciju EEPROM-a traje oko 10ms (što je za pojmove mikrokontrolera veoma dugo), to se mikrokontroleru ne isplati da čeka da se taj upis završi, već je dodat mehanizam prekida po kome on može da nastavi sa izvršenjem glavnog programa, dok se u pozadini vrši upis u EEPROM.
- Kada se upis završi, prekid obaveštava mikrokontroler da je upis gotov. Bit EEIF, kojim se ovo obaveštenje vrši, nalazi se registru EECON1. Pojava prekida može biti onemogućena resetovanjem EEIE bita u INTCOM registru.

Portovi

- Fizička veza centralne procesorske jedinice sa spoljnim svetom.
- Svi pinovi portova mogu se definisati kao ulazni ili kao izlazni, već prema potrebama uređaja koji se razvija.
- PIC16F84 ima dva porta: PORTA i PORTB.
- Da bi definisali pin kao ulazni ili kao izlazni mora se u registar TRIS upisati odgovarajuća kombinacija nula i jedinica. Ako je na odgovarajućem mestu u TRIS registru upisana logička jedinica "1" onda je taj pin ulazni, u obratnom slučaju pin je izlazni.
- Svaki port ima svoj odgovarajući TRIS registar. Tako port A ima TRISA a port B TRISB. Promena smera pinova se može vršiti u toku rada što je posebno pogodno za slučaj komunikacije preko jedne linije gde se menja smer prenosa podataka.
- Registri stanja portova PORTA i PORTB se nalaze u banci 0 dok se registri za smer pinova TRISA i TRISB nalaze u banci 1.

PORTA

- Ima 5 pinova koji su mu pridruženi
- Odgovarajući registar za smer podataka je TRISA
- RA4 pin
  - Na tom pinu se još nalazi i spoljni ulaz za brojač TMR0.
  - Da li se ovaj pin bira kao standardni ulazni pin ili kao ulaz brojača koji se vodi na TMR0 zavisi od bita T0CS (TMR0 Clock Source Select bit).
  - Ovaj bit omogućava da brojač TMR0 uvećava svoje stanje ili iz internog oscilatora ili preko spoljnih impulsa na pinu RA4/TOCKI.



Sl. 2.14. Odnos TRISA i PORTA registra

PORTB

- PORTB ima 8 pinova koji su mu pridruženi.
- Odgovarajući registar za smer podataka je TRISB
- Svaki pin na PORTB ima slabi interni pull-up otpornik (otpornik koji definiše liniju na logičku jedinicu) koji se može aktivirati resetovanjem sedmog bita RBPU u registru OPTION.
- Ovi ''pull-up'' otpornoci se automatski isključuče kada je pin porta konfigurisan kao izlaz. Pri uključenju mikrokontrolera pull-up -ovi su onemogućeni.
- RB7:RB4 mogu izazvati prekid (interapt) koji se dešava kada se stanje na njima promeni iz logičke jedinice u logičku nulu i obratno.
- RB0 se koristi takođe kao izvor prekida (rastuća ili opadajuća ivica)

Mašinski jezik mikrokontrolera PIC16F84

- RISC, redukovan skup instrukcija
- 14-bit instrukcije
- U zavisnosti od tipa, mogu sadržati sledeće elemente
  - f-Memorijski (fajl) registar (7-bit)
  - d-Destinacioni bit (1-bit)
    - d=0 :destinacija akumulator (w)
    - d=1 :destinacija memorijski registar
  - b-kod odgovarajućeg bita u registru (3-bit)
  - k-konstanta (literal)
    - 8-bit za operacije za rad sa memorijskim registrima
    - 11-bit za adrese bezulsovnih skokova

Tipovi mašinskih istrukcija PIC16F84

- Tri grupe:
  - Bajt-orijentisane za rad sa memoriskim registrima
  - Bit-orijentisane za rad sa memoriskim registrima
  - Kontrolne instrukcije i rad sa literalima
- Sve instrukcije se izvršavaju u jednom taktu procesora, izuzev instrukcija koje se izvršavaju u 2 takta
  - Uslovni i bezuslovni skokova
  - Poziv i povratak iz potprograma

Bajt-orijentisane naredbe za rad sa memorijskim registrima

13 8	7	6 0
Operacioni kod	Destinacija (d)	Fajl registar (f)

- MOVF f,d
  - d=0->w:=f
  - d=1->f:=f (korisno za testiranje ZERO FLAG)
- SWAPF f,d
  - Niža i viša četvorka osmobitnog REGISTRA menjaju mesta
- ADDWF, ANDWF, SUBWF, IORWF, XORWF
  - Za SUBWF f,d recimo, kada je d=0 efekat je w:=f-w, a za d=1 f:=f-w
- COMF REGISTAR,d
  - d=0->w:=komplement(REGISTAR)
  - d=1->REGISTAR:=komplement(REGISTAR)
- INCF f,d/DECF f,d
- RRF f,d
  - Sadržaj registra se rotira za jedan bit udesno kroz CARRY FLAG.
- RLF f,d
  - Sadržaj registra se rotira za jedan bit ulevo kroz CARRY FLAG.

Bit-orijentisane naredbe za rad sa memorijskim registrima

13 10	9 7	6 0
Operacioni kod	Bit kod (b)	Fajl registar (f)

- BTFSS f,b
  - Testira bit b u registru f. Preskače narednu instrukciju ako je f[b]==1.
- BTFSC f,b
  - Testira bit b u registru f. Preskače narednu instrukciju ako je f[b]==0.
- BSF f,b
  - Postavlja b-ti bit registra na vrednost 1.
- BCF f,b
  - Postavlja b-ti bit registra na vrednost 0.

Rad sa literalima

13	8	7	0
Operacioni kod		Literal (k)	

- k je u ovom slučaju 8-bit konstanta
- MOVLW k
  - w:=k
- ADDWF k, ANDWF k, SUBWF k, IORWF k, XORWF k
  - w:=w+k/w and k/…/…/…/…/
- RETLW k
  - Povratak iz interapta uz w:=k

Rad sa literalima

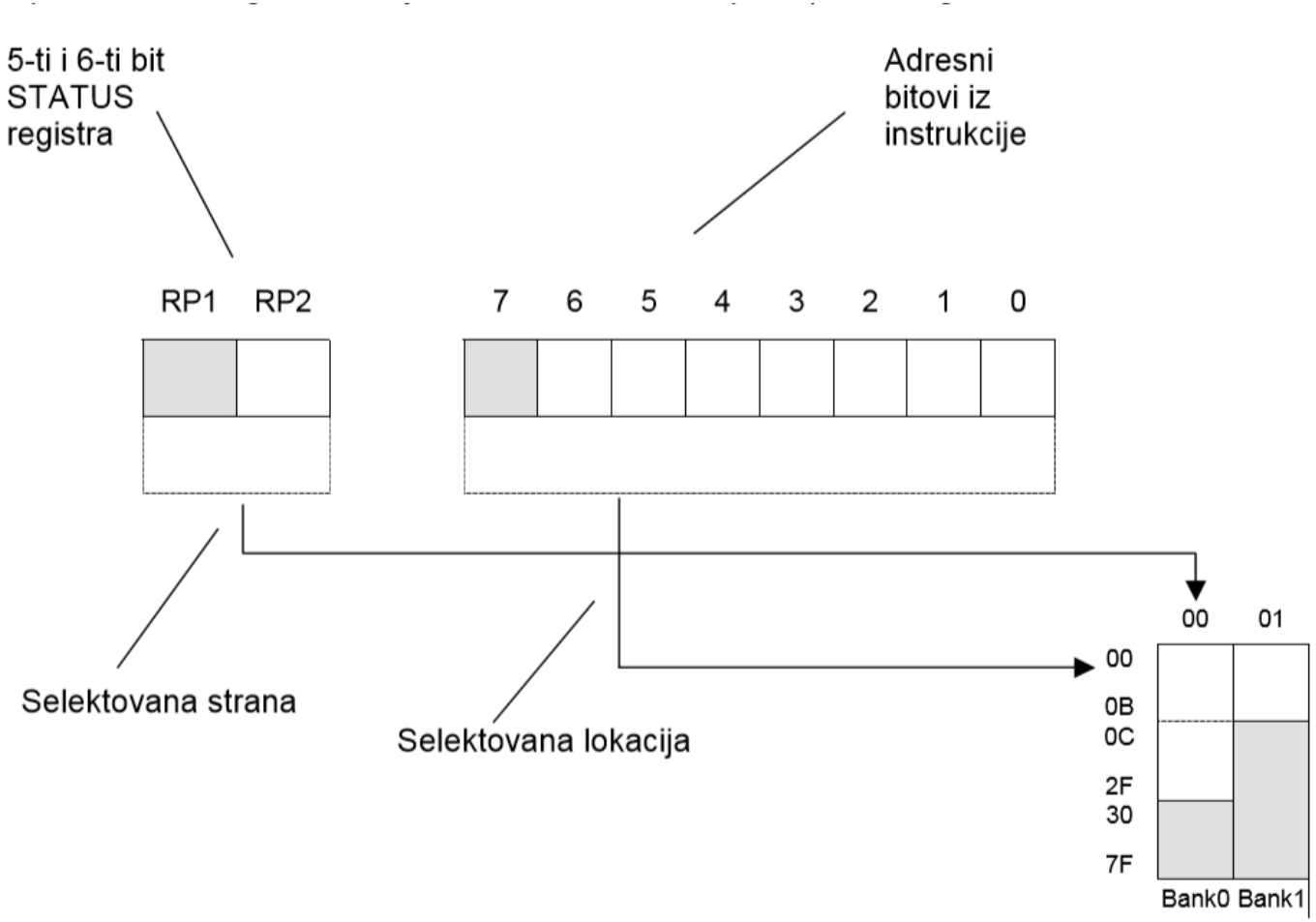
13	11	10	0
Operacioni kod		Apsolutna adresa skoka (k)	

- k je u ovom slučaju 11-bit konstanta
- GOTO k
  - w:=k
  - Bezuslovni skok
- CALL k
  - Poziv potprograma

Adresiranje

- direktno
- indirektno
- relativno

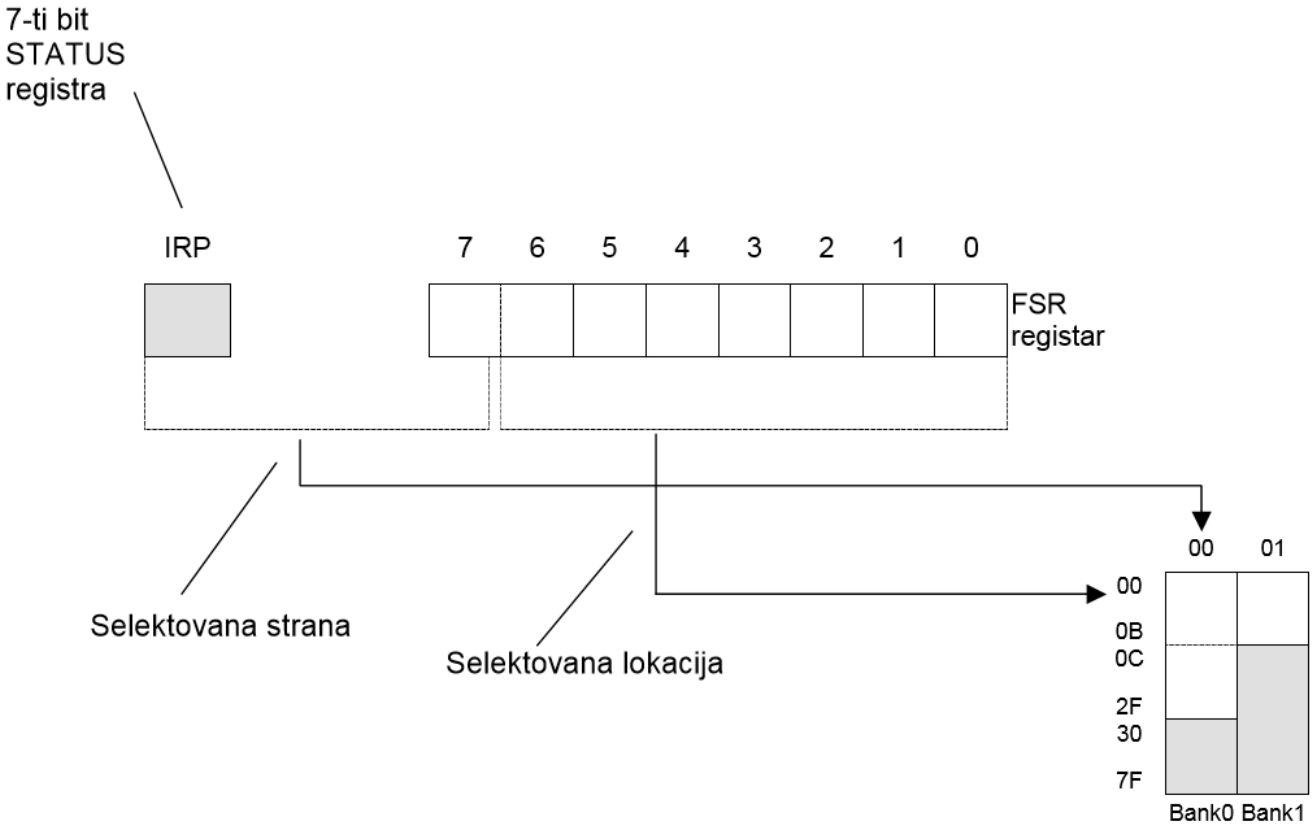
Direktno adresiranje



Selekcija memorijske banke

- Za pristup specijalnim registrima potrebno je pored navođenja imena registra u okviru instrukcije prethodno selektovati odgovarajuću banku.
- Selekcija banke se može vršiti i pomoću direktive banksel posle koje se navodi ime registra kome se pristupa. Na ovaj način nema potrebe da se pamti koji je registar u kojoj banci.
  - banksel TRISB ; Pistupi banci u kojoj je TRISB clrf TRISB ; Izvrsi operaciju nad registrom TRISB
  - banksel POTRB ; Pristupi banci u kojo je PORTB clrf PORTB ; Izvrsi operaciju nad registrom PORTB
- Primer: bcf STATUS,RPO
  - Efekat: Instrukcija BCF resetuje bit RPO (RP0=0) u STATUS registru i time selektuje banku 0, koja ostaje selektovana sve dok se RP= ne postavi na jedan.
- Primer: bsf STATUS,RPO Instrukcija BSF setuje bit RP0
  - Efekat: Instrukcija BSF setuje bit RP0 (RP0=1) u STATUS registru i time selektuje adresiranje registara iz banke1.

Indirektno adresiranje

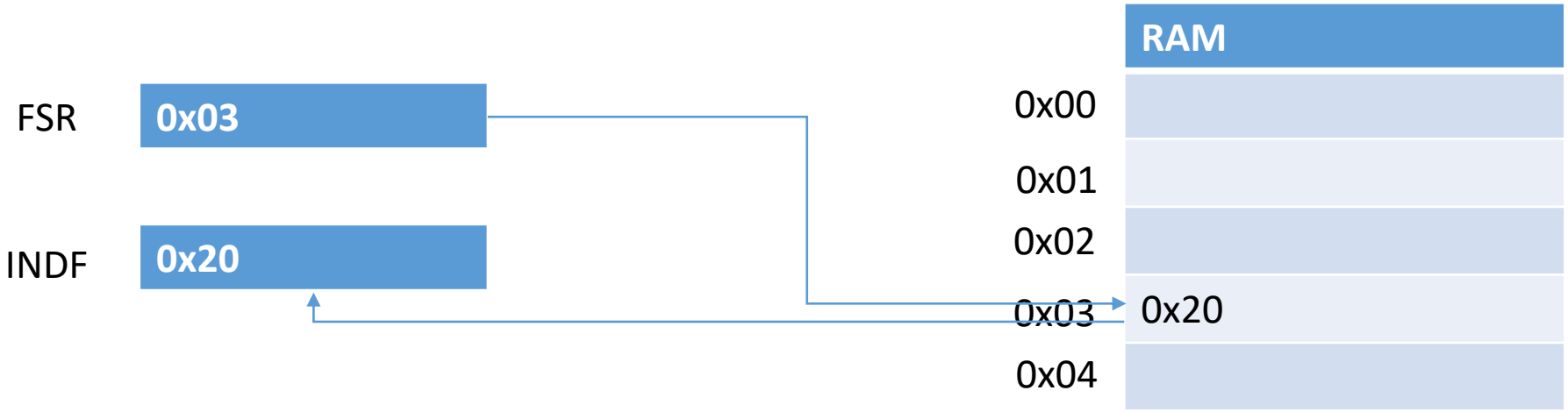


Sl. 2.17. Indirektno adresiranje

FSR i INDF registri

- Indirektno adresiranje se ostvaruje pomoću INDF i FSR registra
- FSR čuva adresu
- Korišćenjem INDF dobijamo vrednost na adresi iz FSR
- Primer: Ako na adresi 0Fh imamo vrednost 20, upisom vrednosti 0Fh u registar FSR dobićemo pokazivač na registar na adresi 0Fh, a čitanjem iz registra INDF dobijamo vrednost 20, što znači da smo iz prvog registra pročitali njegovu vrednost a da mu nismo direktno pristupili (već preko FSR i INDF).
- Indirektno adresiranje je veoma pogodno kada se vrše operacijama sa nizovima podataka koji su smešteni u okviru GPR registra. U tom slučaju je na početku potrebno inicijalizovati registar FSR na vrednost adrese prvog člana niza, a zatim se narednim članovima niza pristupa uvećanjem registra FSR.

Indirektno adresiranje



Stek

- 13-bitni stek (Stack) sa 8 nivoa ili drugim rečima, grupisanih 8 memorijskih lokacija širine 13 bita sa posebnom namenom.
- Njegova osnovna uloga je da sačuva vrednost programskog brojača nakon što se iz glavnog programa skoči na adresu podprograma koji se izvršava.
- Da bi program znao da se vrati na mesto odakle je pošao, mora sa steka da vrati vrednost programskog brojača. Pri prelasku iz programa u podprogram, programski brojač se potiskuje na stek (Primer je instrukcija CALL), a pri izvršenju instrukcija kao što su RETURN, RETLW ili RETFIE koje se izvršavaju na kraju podprograma, vraća sa steka da bi program mogao da nastavi tamo gde je stao pre nego što je bio prekinut.
- Ove operacije stavljanja i vraćanja sa steka programskog brojača u žargonu se nazivaju PUSH i POP po instrukcijama koje pod istim imenom postoje na nekim većim mikrokontrolerima.

Programski brojač

- Programski brojač (PC) je 13-to bitni registar koji sadrži adresu instrukcije koja se izvršava.
- Fizički se realizuje pomoću petobitnog registra PCLATH koji predstavlja pet viših bitova adrese i osmobitnog registra PCL koji predstavlja nižih osam bita adrese.
- Njegovim uvaćanjem ili promenom (npr. u slučaju skoka) mikrokontroler izvršava jednu po jednu instrukciju programa.

Incijalizacija prekida

```
clrf   INTCON  ; svi prekidi onemogućeni
movlw   B'00010000' ; omogućen samo spoljni prekid
movwf   INTCON
bsf   INTCON,  GIE ; dozvoljena pojava prekida
```

Čuvanje sadržaja važnih registara

- Za vreme prekida, samo se povratna vrednost programskog brojača čuva na steku (pod povratnom vrednošću programskog brojača podrazumeva se adresa instrukcije koja je trebala da se izvrši, ali nije se izvršila jer se prekid desio)
- Procedura snimanja važnih registara pre odlaska u prekidnu rutinu u žargonu se naziva ''puš'' (PUSH), dok se procedura vraćanja snimljenih vrednosti naziva ''pop'' (POP). PUSH i POP su instrukcije kod nekih drugih mikorokontrolera (Intel), ali su toliko prihvaćene da se po njima naziva čitava operacija. PIC16F84 nema instrukcija kao što su PUSH i POP i one se moraju programski napraviti
- Za razmenu podataka između registara koristi se instrukcija SWAPF umesto MOVF jer ona ne utiče na stanje bitova STATUS registra
- SWAP f,d : zamena više i niže tetrade u registru f

```
Prekidni_program:
      movwf       WREG_TEMP   ;save WREG
      movwf       WREG_TEMP   ;save WREG
      swapf       STATUS,W    ;store STATUS in WREG
      clrf        STATUS      ;select file register bank0
      movwf       STATUS_TEMP ;save STATUS value

;test interrupt flags here
  ;-------------------------------
      ;resave routine
EndInt:
      swapf       STATUS_TEMP,W ;get saved STATUS value
      movwf       STATUS      ;restore STATUS
      swapf       WREG_TEMP,F ;prepare    WREG    to    be
restored
      swapf       WREG_TEMP,W ;restore    WREG    without
affecting STATUS
      retfie      ;return from interrupt
      end
      ;kraj primera
```

Čitanje EEPROM memorije

- Setovanje bita RD inicira prenos podataka sa adrese koja se nalazi u registru EEADR u EDATA registar.
- Kako za čitanje podataka nije potrebno vreme kao za upis, preuzeti podatak iz EEDATA registra može se već u narednoj instrukciji koristiti dalje.
- Primer čitanja sadržaja EEPROM memorije
  - bcf STATUS, RPO ;bank0, jer je EEDAR na 09h
  - movlw 0x00 ;adresa lokacije koja se čita
  - movwf EEADR ;adresa se prebacuje u EEADR
  - bsf STATUS, RPO ;bank1 jer je EECON1 na 88h
  - bsf EECON1, RD ;čitanje uz EEPROM-a
  - bcf STATUS, RPO ;Bank0 jer je EEDATA na 08h
  - movf EEDATA, W ;W <-- EEDATA
- Nakon poslednje programske instrukcije, sadržaj sa adrese nula EEPROM se nalazi u radnom registru w.

Upis u EEPROM memoriju

- Da bi upisao podatke u EEPROM lokaciju, programer mora prvo da upiše adresu u EEADR registar i podatak u EEDATA registar. Tek tada ima svrhu setovati bit WR, koji pokreće celu akciju. Bit WR će, nakon upisa biti resetovan a EEIF setovan, što se može iskoristiti za obradu prekida.
- Vrednosti 55h i AAh su prvi i drugi ključ koji onemogućuju da dođe do slučajnog upisa u EEPROM. Ove dve vrednosti se upisuju u EECON2 koji služi samo tome da prihvati ove dve vrednosti i time spreči svaki slučajan upis u EEPROM memoriju. Programske linije označene sa 1, 2, 3, 4 i 5 moraju biti izvršene tim redosledom u pravilnom vremenskom razmaku tako da je od velike važnosti isključiti prekide koji bi mogli da poremete potreban vremenski redosled izvršavanja insrtukcija.
- Nakon upisa, prekidi se mogu na kraju ponovo omogućiti.
- Primer upisa u EEPROM memoriju
  - bcf STATUS, RPO ;bank0, jer je EEADR na 09h
  - movlw 0x00 ;adresa lokacije u koju se piše
  - movwf EEADR ;adresa se prebacuje u EEADR
  - movlw 0xEE ;upisujemo vrednost 0xEE
  - movwf EEDATA ;podatak ide u EEDATA registar
  - bsf STATUS, RPO ;Bank1 jer je EEADR na o9h
  - bcf INTCON, GIE ;svi prekidi se onemogućuju
  - bsf EECON1, WREN ;omogućuje se upis
  - 1) movlw 55h
  - 2) movwf EECON2 ;prvi ključ 55h --> EECON2
  - 3) movlw AAh
  - 4) movwf EECON2 ;drugi ključ AAh --> EECON2
  - 5) bsf EECON1, WR ;inicira upis
  - bsf INTCON, GIE ;prekidi se omogućuju

CONFIGURATION WORD

REGISTER 6-1: PIC16F84A CONFIGURATION WORD

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{\text{PWRTE}}$	WDTE	F0SC1	F0SC0
bit13													bit0

bit 13-4 **CP:** Code Protection bit
1 = Code protection disabled
0 = All program memory is code protected

bit 3 **$\overline{\text{PWRTE}}$**: Power-up Timer Enable bit
1 = Power-up Timer is disabled
0 = Power-up Timer is enabled

bit 2 **WDTE**: Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Zadatak 1

- Napisati program na asemblerskom jeziku za PIC16F84 koji upravlja sadrzajem 7s displeja.
- Pin RB7 povezan je na taster a linije RB0-RB6 na segmente displeja. Pritiskom na taster treba inkrementirati sadrzaj prikazan na displeju.
- Kada vrednost prikazna na displeju bude 9 nakon inkrementiranja postaje 0.
- Napisati proceduru za prikaz cifre na displeju a tablicu definicija cifara realizovati programski.
- Početna vrednost prikazana na displeju je nula.

Rešenje a)

```
#include <p16f84a.inc>

#define BANKA0 bcf 3,5
#define BANKA1 bsf 3,5

CIFRA EQU 0x30

    ORG 0x0000
    goto start

start:
    BANKA1
    clrf TRISA
    clrf TRISB
    bsf TRISB,7
    BANKA0
    clrf CIFRA
    call prikazi

petlja:
    btfss PORTB,7
    call uvecajprikazi
    goto petlja

prikazi:
    movf CIFRA, W
    call dekodiranje
    movwf PORTB
    return

uvecajprikazi:
    incf CIFRA, F
    movlw 10
    subwf CIFRA, W
    btfsc STATUS, Z
    clrf CIFRA
    call prikazi
otpusti:
    btfsc PORTB, 7
    return
    goto otpusti

dekodiranje:
    addwf PCL, F
    retlw 0x7E
    retlw 0x30
    retlw 0x6d
    retlw 0x79
    retlw 0x33
    retlw 0x5b
    retlw 0x5f
    retlw 0x7d
    retlw 0x7f
    retlw 0x7b

    END
```

Rešenje b)

```
#include <p16f84a.inc>

#define BANKA0 bcf 3,5
#define BANKA1 bsf 3,5

CIFRA EQU 0x30
PRESSED EQU 0x31

     ORG 0x0000
     goto start

start:
     BANKA1
     clrf TRISA
     clrf TRISB
     bsf TRISB,7
     BANKA0
     clrf CIFRA
     clrf PRESSED
     call prikazi

petlja:
     btfss PORTB,7
     goto uvecajprikazi
     bcf PRESSED, 0
     goto petlja

uvecajprikazi:
     btfsc PRESSED, 0
     goto petlja
     bsf PRESSED, 0
     incf CIFRA, F
     movlw 10
     subwf CIFRA, W
     btfsc STATUS, Z
     clrf CIFRA
     call prikazi
goto petlja

prikazi:
     movf CIFRA, W
     call dekodiranje
     movwf PORTB
     return
dekodiranje:
     addwf PCL, F
     retlw 0x7E
     retlw 0x30
     retlw 0x6d
     retlw 0x79
     retlw 0x33
     retlw 0x5b
     retlw 0x5f
     retlw 0x7d
     retlw 0x7f
     retlw 0x7b

     END
```

XC8

```
#include <stdint.h>
//#include <xc.h>
#include <htc.h>

#define _XTAL_FREQ 3276800

// Configuration Byte
#pragma config CP = OFF, WDTE = OFF, PWRTE = ON, FOSC = HS
// Flash Program Memory Code Protection bit (Code protection off)
// Watchdog Timer Enable bit (WDT disabled)
// Power-up Timer Enable bit (PWRT disabled)
// Oscillator Selection bits (HS oscillator)

void prikazi(uint8_t cifra){
    static const uint8_t codes[] = {0x7E, 0x30, 0x6d, 0x79, 0x33, 0x5b, 0x5f, 0x7d, 0x7f, 0x7b};
    if (cifra<10)
        PORTB = codes[cifra];
}

void main(void) {
    uint8_t cifra = 0;
    uint8_t pressed = 0;

    TRISB = 0;
    TRISB |= 0x80;//TRISBbits. TRISB = TRISB | 0x80
    // TRISBbits.TRISB7 = 1;
    //TRISA = 0x00;
    prikazi(cifra);

    for(;;){// while(1)
        if (PORTBbits.RB7 == 0){
                __delay_ms(10);
                if (PORTBbits.RB7 == 0){
                      if (pressed == 0){
                            pressed = 1;
                            cifra+=1;
                            if (cifra == 10)
                                  cifra = 0;
                            prikazi(cifra);
                      }
                }
        }
        else{
                __delay_ms(10);
                if (PORTBbits.RB7 != 0){
                      pressed = 0;
                }
        }
    }
}
```

Zadatak 2

- Na asemblerskom jeziku napisati program za mikrokontroler PIC16F84A koji kontroliše dva 7s displeja.
- Pinove RB7 i RA2 mikrokontrolera povezati na tastere.
- Glavni deo programa treba da obavlja inkrementiranje i dekrementiranje sadržaja displeja u zavisnosti od toga koji je taster pritisnut.
- Taster povezan na RB7 obavlja inkrementiranje a taster povezan na RA2 obavlja dekrementiranje. Napisati zatim prekidnu proceduru koja na displejima prikazuje sadržaj cifara sa lokacija 0x30 i 0x31 tehnikom osvežavanja.
- Tablicu definicija cifara realizovati programski. Početna vrednost prikazana na

displejima je 00.

- Sa vrednosti 99 prelazi se na 00 nakon dekrementiranja i obrnuto. Frekvencija oscilatora je 3.2768MHz a displej osvezavati sa 50Hz.
- RA0 i RA1 kontrolni tranzistori.

```
#include <p16f84a.inc>
    errorlevel -302
    __CONFIG   _CP_OFF & _WDT_OFF & _PWRTE_ON & _HS_OSC  ;ovo
je za PIC16F84

    CBLOCK 0X30
    CIFRA1
    CIFRA2
    BROJAC
    INDEKS
    WREG_TEMP           ;storage for WREG during interrupt
    STATUS_TEMP         ;storage for STATUS during interrupt
    PCLATH_TEMP         ;storage for PCLATH during interrupt
    FSR_TEMP            ;storage for FSR during interrupt
    ENDC

    ORG 0x0000
    goto ResetCode
    ORG     0x0004            ;place code at interrupt vector
      goto InterruptCode

ResetCode:
    clrf    PCLATH            ;select program memory page 0
      goto  Main        ;go to beginning of program

Main:


    banksel TRISA
    clrf TRISA
    bsf TRISA, 2
    clrf TRISB
    bsf TRISB, 7

      banksel PORTA
    clrf PORTA
    clrf PORTB

    movlw b'00000100'
    movwf OPTION_REG
    movlw b'10100000'
    movwf INTCON

    clrf BROJAC
    clrf INDEKS
    clrf CIFRA1
    clrf CIFRA2

    banksel PORTB

petlja:
    btfss PORTB, 7
    call uvecaj
    btfss PORTA, 2
    call umanji
    goto petlja
```

```
uvecaj:
    clrf BROJAC
    call delay
    btfsc PORTB, 7
    return
    incf CIFRA2, F
    movlw .10
    subwf CIFRA2, W
    btfss STATUS, Z
    goto kraj
    clrf CIFRA2
    incf CIFRA1, F
    movlw .10
    subwf CIFRA1, F
    btfsc STATUS, Z
    clrf CIFRA1
kraj:
    btfss PORTB, 7
    goto kraj
    clrf BROJAC
    call delay
    btfss PORTB, 7
    goto kraj
    return
```

```
umanji:
    clrf BROJAC
    call delay
    btfsc PORTA, 2
    return
    movf CIFRA2, F
    btfss STATUS, Z
    goto dek1
    movlw .9
    movwf CIFRA2
    movf CIFRA1, F
    btfss STATUS, Z
    goto dek2
    movwf CIFRA1
    goto kraj2
dek1:
    decf CIFRA2, F
    goto kraj2
dek2:
    decf CIFRA1, F
kraj2:
    btfss PORTA, 2
    goto kraj2
    clrf BROJAC
    call delay
    btfss PORTA, 2
    goto kraj2
    return
```

```
delay:
    btfss BROJAC, 2
    goto delay
    return

dekodiranje:
    addwf PCL, F
    retlw 0x7e
    retlw 0x30
    retlw 0x6d
    retlw 0x79
    retlw 0x33
    retlw 0x5b
    retlw 0x5f
    retlw 0x70
    retlw 0x7f
    retlw 0x7b
```

```
InterruptCode:
    movwf   WREG_TEMP   ;save WREG
    swapf   STATUS,W    ;store STATUS in WREG
    clrf    STATUS            ;select file register bank0
    movwf   STATUS_TEMP ;save STATUS value
    movf    PCLATH,W    ;store PCLATH in WREG
    movwf   PCLATH_TEMP ;save PCLATH value
    clrf    PCLATH            ;select program memory
page0
    movf    FSR,W       ;store FSR in WREG
    movwf   FSR_TEMP    ;save FSR value

    ;test interrupt flags here
    btfss INTCON, 2
    goto EndInt

    incf INDEKS, F
      movf CIFRA1, W
    btfsc INDEKS, 0
    movf CIFRA2, W

    call dekodiranje
    movwf PORTB

    btfsc INDEKS, 0
    goto cifra2
    bsf PORTA, 0
    bcf PORTA, 1
    goto kraj2
cifra2:
    bcf PORTA, 0
    bsf PORTA, 1
kraj3:
    incf BROJAC, F
    bcf INTCON, T0IF

;-----------------------------------
;End of interrupt routine restores context

EndInt:
    bcf     3,5         ;select bank 0
    movf    FSR_TEMP,W  ;get saved FSR value
    movwf   FSR         ;restore FSR
    movf    PCLATH_TEMP,W     ;get saved PCLATH value
    movwf   PCLATH            ;restore PCLATH
    swapf   STATUS_TEMP,W     ;get saved STATUS value
    movwf   STATUS            ;restore STATUS
    swapf   WREG_TEMP,F ;prepare WREG to be restored
    swapf   WREG_TEMP,W ;restore WREG without affecting
STATUS
    retfie              ;return from interrupt

    end
```

XC8 REŠENJE

```
#include <stdio.h>
#include <stdlib.h>

#include <xc.h>

#define _XTAL_FREQ 3276800

// Configuration Byte
#pragma config CP = OFF         // Flash Program Memory Code Protection bit
(Code protection off)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = ON      // Power-up Timer Enable bit (PWRT disabled)
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)

#define DISPLAY PORTB
#define BUTTON1 !PORTBbits.RB7
#define BUTTON2 !PORTAbits.RA2

unsigned char index = 0;
unsigned char pressed = 0;// cuva stanje tastera
unsigned char D1 = 0;
unsigned char D2 = 0;

unsigned char get_code(unsigned char cifra) {
    static const unsigned char codes[10] = {0x7D, 0x30, 0x6E, 0x7A, 0x33, 0x5B,
0x5F, 0x70, 0x7F, 0x7B};
    return codes[cifra];
}
```

```
void main(int argc, char** argv) {
    TRISB = 0;
    TRISBbits.TRISB7 = 1;

    TRISA = 0;
    TRISAbits.TRISA2 = 1;
    DISPLAY = 0;
      PORTA = 0;

    OPTION_REG = 0x04;
    INTCON = 0xA0;// GIE, T0IE = 1
    INTCONbits.T0IF = 0;

    while (1) {
        if (BUTTON1) {
            __delay_ms(20);
            if (BUTTON1) {
                if (!pressed) {
                    pressed = 1;

                    if (++D2 == 10) {
                        D2 = 0;
                        if (++D1 == 10)
                            D1 = 0;
                    }
                }
            }
        } else {
            __delay_ms(20);
            if (!BUTTON1)
                pressed = 0;
        }

        if (BUTTON2) {
            __delay_ms(20);
            if (BUTTON2) {
                if (!pressed) {
                    pressed = 1;

                    if (D2 == 0) {
                        D2 = 9;
                        if (D1 == 0)
                            D1 = 9;
                        else
                            D1--;
                    } else
                        D2--;

                }
            }
        } else {
            __delay_ms(20);
            if (!BUTTON2)
                pressed = 0;
        }
    }
}
```

```
void interrupt intcode() {
    if (INTCONbits.T0IF == 1)
{
        if (index++ & 1) {
            DISPLAY =
get_code(D1);
            PORTAbits.RA0 = 1;
            PORTAbits.RA1 = 0;
        } else {
            DISPLAY =
get_code(D2);
            PORTAbits.RA1 = 1;
            PORTAbits.RA0 = 0;
        }

        INTCONbits.T0IF = 0;
    }
}
```

Vremenske petlje

- Ponekad je neophodno da se implementira fiksno kašnjenje u PIC programima
- To se postiže izvođenjem “beskorisnih” instrukcija, pri čemu su često organizovane u vidu ugnježdenih petlji
- Za ovu svrhu koristi se decfsz instrukcija, čiji je efekat da dekrementira sadržaj registra i preskoči sledeću instrukciju ako je rezultat nakon dekrementiranja nula
- Takođe, koristi se i NOP da dopuni kašnjenje do određene vrednosti

Primer petlje bez ugnježđavanja

```
                        ;counter cnt_1 is given a value
                        ;before the execution of the delay loop
lbl:    decfsz  cnt_1   ;decrement cnt_1 skip next instruction if result was 0
        goto    lbl     ;loop

                        ;counter cnt_1 is zero at this point
```

```
                ;in this example cnt_1 = 4
decfsz  cnt_1   ; cnt_1 = 3,  microcyles =  1 } 3
goto            ;       = 3,              =  2
decfsz  cnt_1   ;       = 2,              =  1 } 3
goto            ;       = 2,              =  2
decfsz  cnt_1   ;       = 1,              =  1 } 3
goto            ;       = 1,              =  2
decfsz  cnt_1   ;       = 0,              =  2
                                    total = 11
```

$$t = 2 + 3(cnt_1 - 1)$$

Primer sa ugnježđenim petljama

```
                        ;counters cnt_1, cnt_2 are given values
                        ;before the execution of the delay loops

lbl:    decfsz  cnt_1   ;decrement cnt_1 skip next instruction if result was 0
        goto    lbl     ;loop
        decfsz  cnt_2   ;decrement cnt_2 skip next instruction if result was 0
        goto    lbl     ;loop

                        ;counters cnt_1, cnt_2 are zero at this point
```

```
<t0_(n-1)>      ; cnt_n =  00, microcycles = t0_(n-1)
decfsz  cnt_1   ;       = 255,             =  1
goto            ;       = 255,             =  2
<t0_(n-1)>      ; cnt_n = 255, microcycles = t0_(n-1)
decfsz  cnt_1   ;       = 254,             =  1
goto            ;       = 254,             =  2
     .
     .
     .
<t0_(n-1)>      ; cnt_n =  01, microcycles = t0_(n-1)
decfsz  cnt_1   ;       =  01,             =  1
goto            ;       =  01,             =  2
decfsz  cnt_1   ;       =  00,             =  2
```

$t = t_{n-1}^0 + 3$

$t = t_{n-1}^0 + 3$

$t = t_{n-1}^0 + 3$

$$t_n^0 = 2 + \left(t_{n-1}^0 + 3\right)(256 - 1) = 255\, t_{n-1}^0 + 767$$

Proizvoljan početni indeks

```
                ;counters have arbitrary values at start
                ;for this example cnt_n = 4

<t1_(n-1)>      ; cnt_n =  00, microcycles = t1_(n-1)
decfsz  cnt_1   ;       =  03,             =  1
goto            ;       =  03,             =  2
<t0_(n-1)>      ; cnt_n =  03, microcycles = t0_(n-1)
decfsz  cnt_1   ;       =  02              =  1
goto            ;       =  02,             =  2
<t0_(n-1)>      ; cnt_n =  02, microcycles = t0_(n-1)
decfsz  cnt_1   ;       =  01,             =  1
goto            ;       =  01,             =  2
<t0_(n-1)>      ; cnt_n =  01, microcycles = t0_(n-1)
decfsz  cnt_1   ;       =  00,             =  2
```

$t = t_{n-1}^{0} + 3$

$t = t_{n-1}^{0} + 3$

$t = t_{n-1}^{0} + 3$

$$t_n^1 = t_{n-1}^1 + \left(cnt_n - 1\right)\left(t_{n-1}^0 + 3\right) + 2$$

Primer

- Koliko traje kašnjenje prouzrokovano petljom, ako se koristi oscilator 4MHz?

```
movlw .203
movwf C1
movlw .8
movwf C2
loop:
    decfsz C1,1
    goto loop
    decfsz C2,1
    goto loop
```

Rešenje

- Potrebno nam je t_2^1
- cnt1=203, cnt2=8
- $t_2^1 = t_1^1 + (cnt2 - 1) * (t_1^0 + 3) + 2$
- $t_1^0 = 255t_0^0 + 767 = 767\ , jer\ je\ (t_0^0 = 0)$
- $t_1^1 = t_0^1 + (cnt1 - 1)(t_0^0 + 3) + 2 = (203 - 1) * 3 + 2 = 608\ , jer\ je\ (t_0^1 = 0)$
- t_2^1= 608+(8-1)(767+3)+2=6000
- S obzirom da je F=4Mhz, a jedan ciklus traje 4 takta, pri čemu je T=1/F, onda je konačno rešenje
  - $T_{ukupno} = 4T * t_2^1 = \frac{4}{F} * t_2^1$ =6000μs=6ms

Zadatak 3

- Projektovati sistem sa šest 7s displeja baziran na procesoru Microchip PIC16F84a.
- Sistem realizovati bez korišćenja pomoćnih logičkih kola na sledeći način:
  - Linije B0-B6 sa mikrokontrolera iskoristiti za kontrolu pojedinačnih segmenata na displejima (a-g).
  - Linije B7 i A0-A4 iskoristiti za kontrolu tranzistora koji su povezani na displeje.
  - Napisati proceduru koja realizuje tehniku osvežavanja displeja.
  - Cifre koje se ispisuju nalaze se na adresama 35h-3Ah.
- U glavnom delu programa je potrebno ciklično rotirati poruku EF2019 ulevo za jednu poziciju brzinom jedne rotiacije u sekundi. Tablicu def. cifara realizovati programski. Takt procesora je 3.2768MHz, a Displej je potrebno osvežavati sa 50Hz.

XC8 rešenje

```
#include <stdio.h>
#include <stdlib.h>

#include <xc.h>
#include <htc.h>
#define _XTAL_FREQ 3276800

// Configuration Byte
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = ON      // Power-up Timer Enable bit (PWRT disabled)
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)

#define DISPLAY PORTB

void osvezi();
void rotiraj();

unsigned char brojac = 0;

unsigned char poruka [] = {'E','F','2','0','1','9'};

unsigned char kod(unsigned char karakter){
    static unsigned char kodovi10[10] = {0x7D, 0x30, 0x6D, 0x79, 0x33, 0x5B, 0x5F, 0x70, 0x7F, 0x7B};
      static unsigned char kodoviaf[6]={0x77, 0x1F, 0x1E, 0x3D, 0x4F, 0x0F};

      if(karakter>='0' && karakter<='9')
            return kodovi10[karakter-'0'];
      else if(karakter>='A' && karakter<='F')
      return kodoviaf[karakter-'A'];
}
```

```
void main(int argc, char** argv) {
    TRISB = 0; // port B output
    TRISA = 0; // RA0 RA1 output

    DISPLAY = 0;

    OPTION_REG = 7;
    TMR0 = 245;
      //256-11
    INTCON = 0x00;

    while (1){
        while (!INTCONbits.T0IF);
        osvezi();
        if (brojac == 50)
            rotiraj();
    }
}
```

```
void osvezi(){
    static unsigned char displej = 1;

    if (displej > 6)
        displej = 1;

      PORTBbits.RB7 = 0;
    switch (displej){
        case 1:
            brojac++;
            DISPLAY = kod(poruka[0]);
            PORTBbits.RB7 = 1;
            PORTA = 0;
            break;
        case 2:
            DISPLAY = kod(poruka[1]);
            PORTA = 1;
            break;
        case 3:
            DISPLAY = kod(poruka[2]);
            PORTA = 2;
            break;
        case 4:
            DISPLAY = kod(poruka[3]);
            PORTA = 4;
            break;
        case 5:
            DISPLAY = kod(poruka[4]);
            PORTA = 8;
            break;
        case 6:
            DISPLAY = kod(poruka[5]);
            PORTA = 16;
            break;
        default:
            break;
    }

    displej++;
}
```

```
void rotiraj(){
    unsigned char pm = poruka[0];
    for (int i=0; i<5; i++)
        poruka[i] = poruka[i+1];
    poruka[5] = pm;
}
```

ASM

```
#include <p16f84a.inc>

    errorlevel -302
    __CONFIG   _CP_OFF & _WDT_OFF & _PWRTE_ON & _HS_OSC  ;ovo
je za PIC16F84

    CBLOCK 0x20
    CODE1
    CODE2
    CODE3
    CODE4
    CODE5
    CODE6
    ENDC

    CBLOCK 0x35
    CIFRA1
    CIFRA2
    CIFRA3
    CIFRA4
    CIFRA5
    CIFRA6
    PROLAZ
    OSVCIF
    PCIFRA
    BROJAC
    WREG_TEMP           ;storage for WREG during interrupt
    STATUS_TEMP         ;storage for STATUS during interrupt
    PCLATH_TEMP         ;storage for PCLATH during interrupt
    FSR_TEMP            ;storage for FSR during interrupt
    ENDC

    ORG 0x0000
    goto ResetCode

ResetCode:
    clrf    PCLATH            ;select program memory page 0
      goto  Main        ;go to beginning of program
```

```
Main:
    banksel TRISA
    clrf TRISA
    clrf TRISB
    movlw b'00000111'
    movwf OPTION_REG
    banksel PORTA
    clrf PORTA
    clrf PORTB
    clrf INTCON
      ;cuvanje potrebnih kodova u RAM-u,
      ;RB6-a...RB0-g
    movlw b'01111110' ; '0'
    movwf CODE1
    movlw b'00110000' ; '1'
    movwf CODE2
    movlw b'01101101' ; '2'
    movwf CODE3
    movlw b'01011011' ; '9'
    movwf CODE4
    movlw b'01001111' ; 'E'
    movwf CODE5
    movlw b'01000111' ; 'F'
    movwf CODE6
      ;IND.  012345
      ;CODE: 0129EF

      ;EF2019
      ;452013

    movlw .4
    movwf CIFRA1
    movlw .5
    movwf CIFRA2
    movlw .2
    movwf CIFRA3
    movlw .0
    movwf CIFRA4
    movlw .1
    movwf CIFRA5
    movlw .3
    movwf CIFRA6
    clrf OSVCIF
    bsf OSVCIF, 7 ;RB'7'
    clrf PROLAZ
```

```
petlja:
    movlw .245
    movwf TMR0
cekaj:
    btfss INTCON, T0IF
    goto cekaj


    call osvezi
    bcf INTCON, T0IF

    movlw .50
    xorwf PROLAZ, W
    btfsc STATUS, Z
    call pomeri
    goto petlja
```

```
osvezi:
    ;10000000
    btfsc OSVCIF, 7
    goto c1
      ;00000001
    btfsc OSVCIF, 0
    goto c2
      ;00000010
    btfsc OSVCIF, 1
      goto c3
      ;00000100
    btfsc OSVCIF, 2
    goto c4
    ;00001000
    btfsc OSVCIF, 3
    goto c5
c6:
    movf CIFRA6, W
    goto lportb
c5:
    movf CIFRA5, W
    goto lportb
c4:
    movf CIFRA4, W
    goto lportb
c3:
    movf CIFRA3, W
    goto lportb
c2:
    movf CIFRA2, W
    goto lportb
c1:
      incf PROLAZ, F
    movf CIFRA1, W
```

```
lportb:
    addlw 0x20
    movwf FSR
    movf INDF, W
    movwf PORTB

    movf OSVCIF, W
    movwf PORTA
    ;1000 0000
      andlw 0x80
    iorwf PORTB, F

    rlf OSVCIF, F
    btfsc STATUS, C
    bsf OSVCIF, 0
    btfss OSVCIF, 5
    return
    bcf OSVCIF, 5
    bsf OSVCIF, 7
    return
```

```
pomeri:
    clrf PROLAZ
    movf CIFRA1, W
    movwf PCIFRA
    movf CIFRA2, W
    movwf CIFRA1
    movf CIFRA3, W
    movwf CIFRA2
    movf CIFRA4, W
    movwf CIFRA3
    movf CIFRA5, W
    movwf CIFRA4
    movf CIFRA6, W
    movwf CIFRA5
    movf PCIFRA, W
    movwf CIFRA6
    return

    END
```

PIC 18 i XC8 primeri

PIC 18 – sličnosti sa familijom PIC 16

- Slično pakovanje i pinovi (veći broj: 28 i 40, kod PIC18F452)
- Slične uloge i nazivi SFR
- Slični razvojni alati
- PIC 16F instrukcije su podskup PIC 18F instrukcija

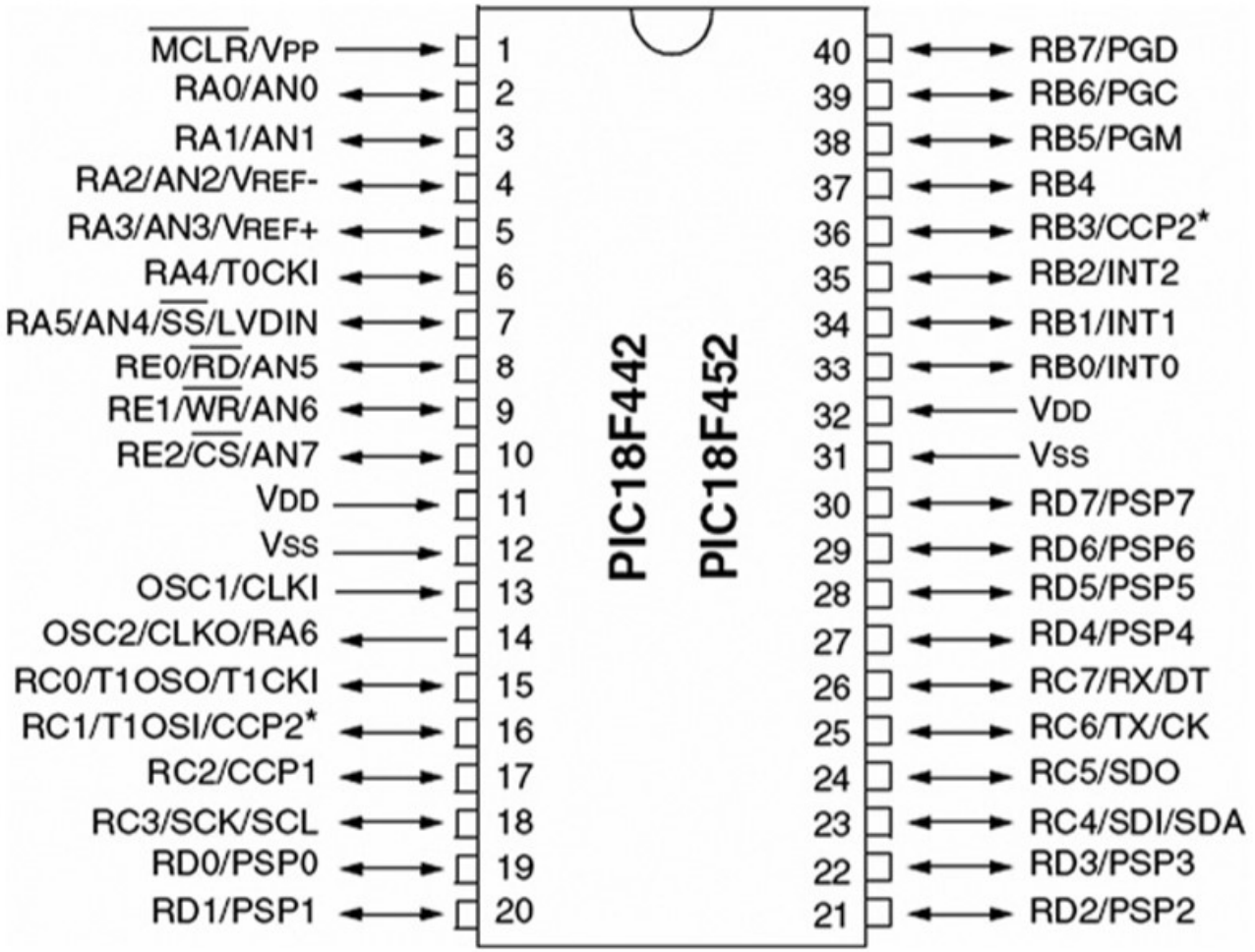
PIC 18 – glavne karakteristike i prednosti u odnosu na PIC 16

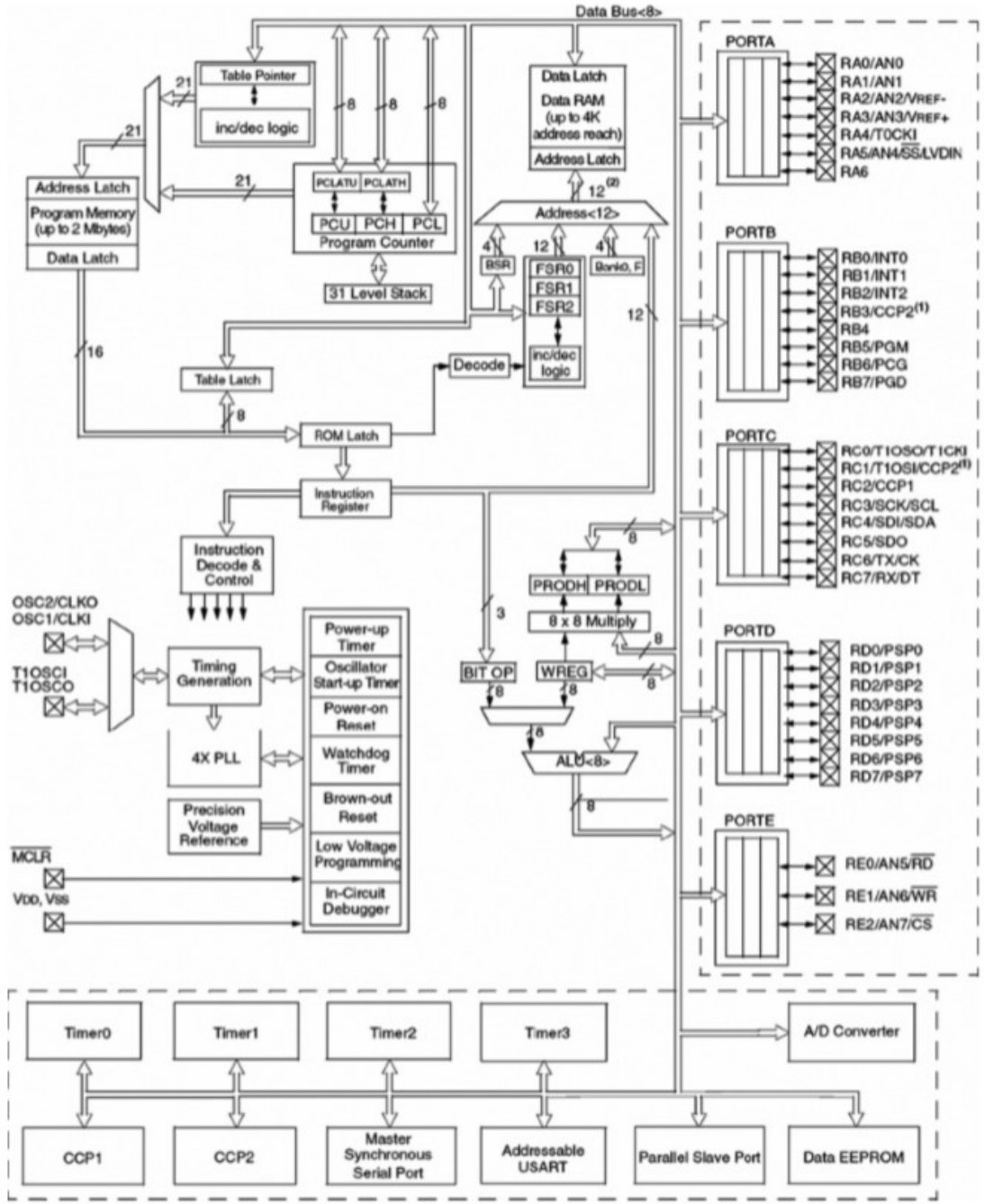
- Kompatibilnost sa PIC 16 source kodom
- Duplo veći broj instrukcija
- 16-bit instrukcijska reč, 8-bit linija podataka
- Hardverski 8 x 8 množač (dva 8-bit registra: PRODH i PRODL)
- Podrška za 16-bit brojače (maksimalno 3) i 8-bit brojače (maksimalno 2)
- Do 5 komparatorskih modula
- Više eksternih interaptova (do 4)
- Prioriteti interaptova
- Veći stack
- Veća programska i memorija podataka
- Skup konfiguracionih registara
- Veća brzina
- Napredniji STATUS registar
- Poboljšana arhitektura i veći broj I/O portova
- 10 bit A/D konvertor

PIC18 – dodatne funkcije

- Real-time OS (RTOS)
  - Aplikacije napisane u C-u
- Komunikacioni protokoli
  - TCP/IP
  - CAN
  - USB
  - ZigBee

40-pin kućište PIC18F452




Data Bus<8>
Table Pointer
inc/dec logic
21
8
Data Latch
Data RAM
(up to 4K
address reach)
Address Latch
12(2)
Address Latch
Program Memory
(up to 2 Mbytes)
Data Latch
PCLATU
PCLATH
PCU
PCH
PCL
Program Counter
31 Level Stack
Address<12>
4
BSR
12
FSR0
FSR1
FSR2
Bank0, F
inc/dec
logic
Decode
Table Latch
16
ROM Latch
Instruction
Register
Instruction
Decode &
Control
PRODH
PRODL
8 x 8 Multiply
3
BIT OP
WREG
ALU<8>
OSC2/CLKO
OSC1/CLKI
T1OSCI
T1OSCO
Timing
Generation
4X PLL
Precision
Voltage
Reference
Power-up
Timer
Oscillator
Start-up Timer
Power-on
Reset
Watchdog
Timer
Brown-out
Reset
Low Voltage
Programming
In-Circuit
Debugger
MCLR
VDD, VSS
PORTA
RA0/AN0
RA1/AN1
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/SS/LVDIN
RA6
PORTB
RB0/INT0
RB1/INT1
RB2/INT2
RB3/CCP2(1)
RB4
RB5/PGM
RB6/PGC
RB7/PGD
PORTC
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2(1)
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
PORTD
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
PORTE
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
Timer0
Timer1
Timer2
Timer3
A/D Converter
CCP1
CCP2
Master
Synchronous
Serial Port
Addressable
USART
Parallel Slave Port
Data EEPROM

Paralelni portovi

- 5 paralelnih portova
  - PORTA (7-bit)
    - Može se koristiti kao analogni ulaz
  - PORTB (8-bit)
  - PORTC (8-bit)
  - PORTD (8-bit)
  - PORTE (3-bit)
- Svaki PORT ima
  - Registar podataka (PORTA)
  - Registar smera (TRISA)
  - Latch registar (LATA)

Tajmeri

- 4 tajmera
- Timer0
  - 8 – ili 16- bit režim (TMR0L i TMR0)
  - 8-bit prescaler
  - Generisanje interapta
  - Eksterni ili interni izvor takta
  - Konfiguracija - T0CON registar
- Timer1
  - 16-bit
  - T1CON
- Timer2
  - 8-bit TMR2
  - 8-bit registar perioda PR2
  - Interrupt kada se TMR2 podudara sa PR2
  - prescaler
  - postscaler
- Timer3
  - Isto kao Timer2

T0CON registar

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT:** Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS:** Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

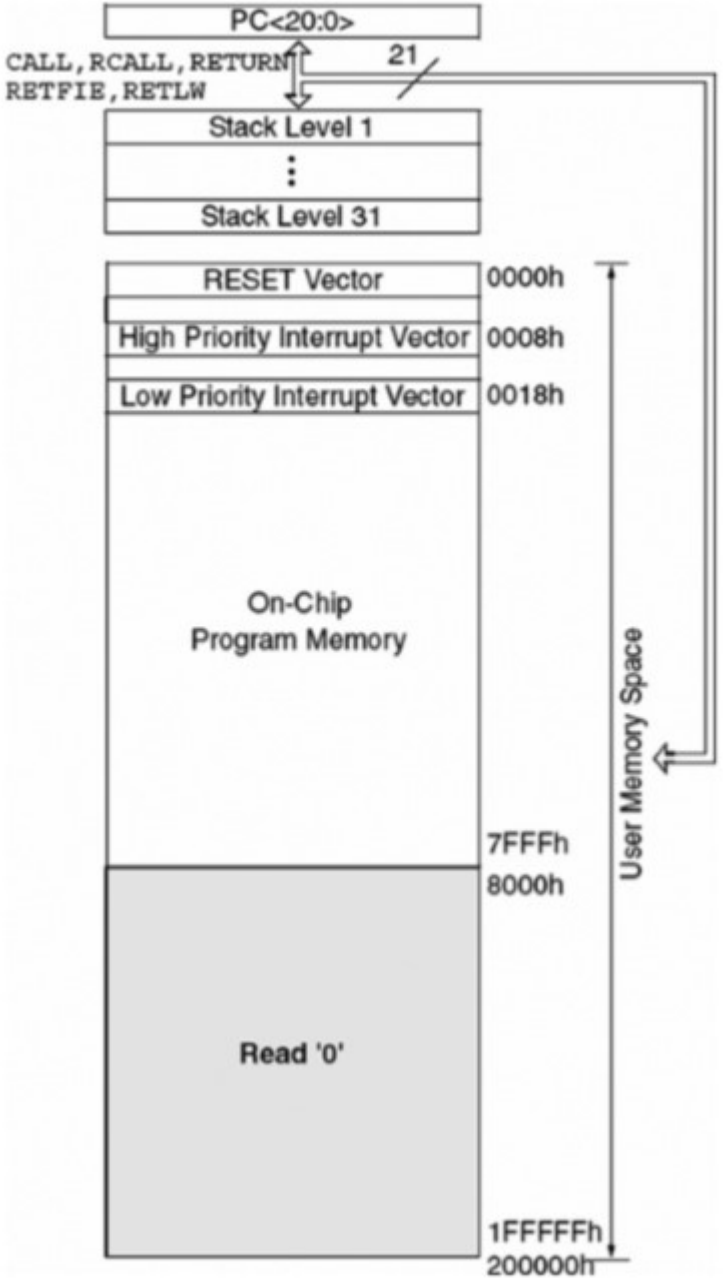
bit 4 **T0SE:** Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0:** Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

PIC18F452 - memorija

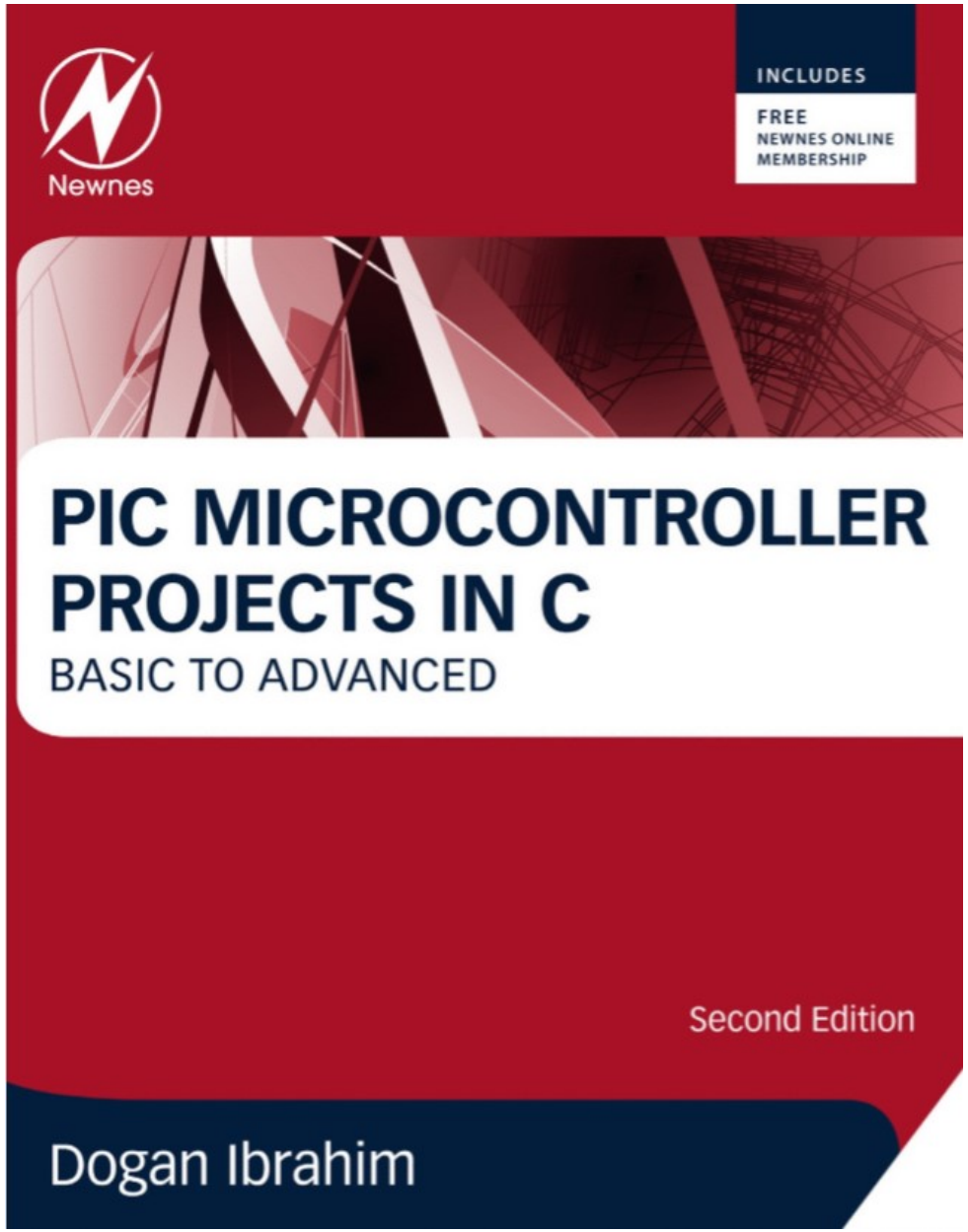
- 21-bit adrese programske memorije
  - 15 bit u upotrebi, ostalih 6 se ne koriste
- 31 nivo magacina, 5-bit stack pointer
- Dve interrupt adrese
  - 0008h za viši prioritet
  - 0018 za niži prioritet



Memorija podataka

- 12 bit magistrala za adresiranje podataka
- 16 banaka po 256 byte
- Samo 6 banaka u upotrebi (6 x 256=1536 bytes)
- Kada se koristi XC8, nema potrebe da se ručno biraju banke

Primeri za XC8+PIC18



Primer 5.2

- LED dioda povezana na RC0
  - 3 treptaja sa 200ms pauzom
  - 2s ugašena

In this project an LEDs is connected to port pin RC0 of a PIC18F45K22 microcontroller and the the microcontroller is operated from an 8 MHz crystal.

The program flashes the LED continuously with the following pattern:

3 flashes with 200 ms delay between each flash
2 s delay

```
Author:     Dogan Ibrahim
Date:       August 2013
File:       XC8-LED2.C
*******************************************************************************/
#include <xc.h>
#pragma config MCLRE = EXTMCLR, WDTEN = OFF, FOSC = HSHP
#define _XTAL_FREQ 8000000

//
// This function creates milliseconds delay. The argument specifies the delay time.
// The delay can be 1 to 65535 ms
//
void DelayMs(unsigned int ms)
{
  unsigned int j;

  for(j = 0; j < ms; j++)__delay_ms(1);
}


void main()
{
   unsigned char i;

   ANSELC = 0;                          // Configure PORTC as digital
   TRISC = 0;                           // Configure PORTC as output
   for(;;)                              // Endless loop
   {
     for(i = 0; i < 3; i++)             // Do 3 times
     {
       PORTCbits.RC0 = 1;               // LED ON
       DelayMs(200);                    // 200 ms delay
       PORTCbits.RC0 = 0;               // LED OFF
       DelayMs(200);                    // 200 ms delay
     }
     DelayMs(2000);
   }
}
```

Figure 5.10: MPLAB XC8 Program Listing.

Primer 5.3

- 8 LED dioda
- Nasumične diode svetle svake sekunde

In this project 8 LEDs are connected to PORTC of a PIC18F45K22 microcontroller and the microcontroller is operated from an 8 MHz crystal.

The program uses a pseudorandom number generator to generate a number between 0 and 32767. This number is then divided by 128 to limit it between 1 and 255. The resultant number is sent to PORTC of the microcontroller. This process is repeated every second.

```
Author:     Dogan Ibrahim
Date:       August 2013
File:       XC8-LED3.C
*******************************************************************************/
#include <xc.h>
#include <stdlib.h>
#pragma config MCLRE = EXTMCLR, WDTEN = OFF, FOSC = HSHP
#define _XTAL_FREQ 8000000

//
// This function creates seconds delay. The argument specifies the delay time in seconds.
//
void Delay_Seconds(unsigned char s)
{
  unsigned char i,j;

  for(j = 0; j < s; j++)
  {
    for(i = 0; i <  100; i++)__delay_ms(10);
  }
}


void main()
{
  unsigned int p;

  ANSELC = 0;                          // Configure PORTC as digital
  TRISC = 0;                           // Configure PORTC as output
  srand(10);                           // Initialize random number seed

  for(;;)                              // Endless loop
  {
    p = rand();                        // Generate a random number
    p = p/128;                         // Number between 1 and 255
    PORTC = p;                         // Send to PORTC
    Delay_Seconds(1);                  // 1 s delay
  }
}
```

Figure 5.13: MPLAB XC8 Program Listing.

Primer 5.4

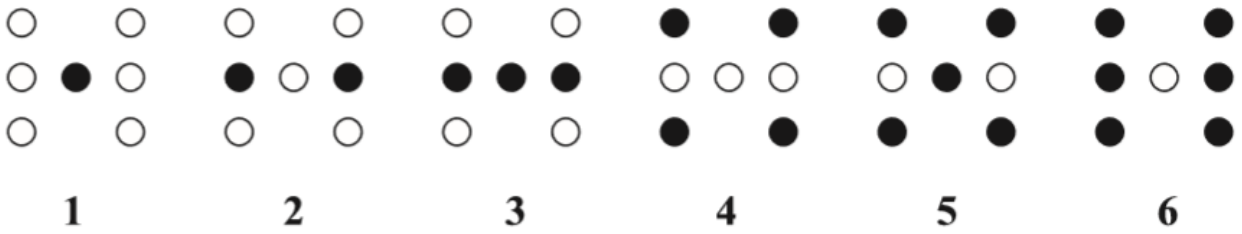
- RC0 – probni ulaz
- RC6 – crvena LED
- RC7 – zelena LED

```
/*********************************************************************************
                                LOGIC PROBE
                                ===========

This is a logic probe project. In this project 2 colored LEDs are connected to PORTC pins RC6
(RED) and RC7 (GREEN). In addition, RC0 is used as the probe input.

If the logic probe is at logic 0 then the RED LED is turned ON. Otherwise, the GREEN LED is
turned ON.

Author:     Dogan Ibrahim
Date:       August 2013
File:       XC8-LED4.C
*********************************************************************************/
#include <xc.h>
#pragma config MCLRE = EXTMCLR, WDTEN = OFF, FOSC = HSHP
#define _XTAL_FREQ 8000000

#define PROBE PORTCbits.RC0
#define RED_LED PORTCbits.RC6
#define GREEN_LED PORTCbits.RC7


void main()
{
   ANSELC = 0;                          // Configure PORTC as digital
   TRISCbits.RC0 = 1;                   // Configure RC0 as input
   TRISCbits.RC6 = 0;                   // Configure RC6 as output
   TRISCbits.RC7 = 0;                   // Configure RC7 as output

   for(;;)                              // Endless loop
   {
     if(PROBE == 0)                     // If the signal is LOW
     {
      GREEN_LED = 0;                     // Turn OFF GREEN LED
      RED_LED = 1;                      // Turn ON RED LED
     }
     else
     {
      RED_LED = 0;                      // Turn OFF RED LED
      GREEN_LED = 1;                    // Turn ON GREEN LED
     }
   }
}
```

Figure 5.17: MPLAB XC8 Program Listing.

Primer 5.5

- Pritiskom na RB0 se prikazuje broj:



- Prikaz stoji 3s, nakon čega se gasi

Required Number	LEDs to be Turned on
1	D4
2	D2, D6
3	D2, D4, D6
4	D1, D3, D5, D7
5	D1, D3, D4, D5, D7
6	D1, D2, D3, D5, D6, D7

Required Number	PORTC Data (Hex)
1	0 x 08
2	0 x 22
3	0 x 2A
4	0 x 55
5	0 x 5D
6	0 x 77

```
/*****************************************************************************
                                SIMPLE DICE
                                ===========

In this project 7 LEDs are connected to PORTC of a PIC18F45K22 microcontroller and the
microcontroller is operated from an 8 MHz crystal. The LEDs are organized as the faces
of a real dice. When a push-button switch connected to RB0 is pressed a dice pattern is
displayed on the LEDs. The display remains in this state for 3 s and after this period
the LEDs all turn OFF to indicate that the system is ready for the button to be pressed again.


Author:         Dogan Ibrahim
Date:           August 2013
File:           XC8-LED5.C
*****************************************************************************/

#include <xc.h>
#pragma config MCLRE = EXTMCLR, WDTEN = OFF, FOSC = HSHP
#define _XTAL_FREQ 8000000

#define Switch PORTBbits.RB0
#define Pressed 0

//
// This function creates seconds delay. The argument specifies the delay time in seconds.
//
void Delay_Seconds(unsigned char s)
{
  unsigned char i,j;

  for(j = 0; j < s; j++)
  {
    for(i = 0; i <  100; i++)__delay_ms(10);
  }
}


void main()
{
   unsigned char J = 1;
   unsigned char Pattern;
   unsigned char DICE[] = {0,0x08,0x22,0x2A,0x55,0x5D,0x77};

   ANSELC = 0;                          // Configure PORTC as digital
   ANSELB = 0;                          // Configure PORTB as digital
   TRISC = 0;                           // Configure PORTC as outputs
   TRISB = 1;                           // Configure RB0 as input
   PORTC = 0;                           // Turn OFF all LEDs

   for(;;)                              // Endless loop
   {
```

Figure 5.26: MPLAB XC8 Program Listing.

Primer 5.5 (nastavak)

```
        if(Switch == Pressed)                  // Is switch pressed ?
        {
         Pattern = DICE[J];                    // Get LED pattern
         PORTC = Pattern;                      // Turn on LEDs
         Delay_Seconds(3);                     // Delay 3 s
         PORTC = 0;                            // Turn OFF all LEDs
         J = 0;                                // Initialize J
        }
        J++;                                   // Increment J
        if(J == 7) J = 1;                      // Back to 1 if  >6
    }
}
```

Figure 5.26
cont'd

Primer 5.7

- Brojanje od 0 do 9
- Inkrementiranje svake sekunde

```
/*****************************************************************************
                                7-SEGMENT DISPLAY
                                =================

In this project a common anode 7-segment LED display is connected to PORTC of a PIC18F45K22
microcontroller and the microcontroller is operated from an 8 MHz crystal. The program displays
numbers 0 to 9 on the display with a 1 s delay between each output.

Author:         Dogan Ibrahim
Date:           August 2013
File:           XC8-LED8.C
*****************************************************************************/

#include <xc.h>
#pragma config MCLRE = EXTMCLR, WDTEN = OFF, FOSC = HSHP
#define _XTAL_FREQ 8000000


//
// This function creates seconds delay. The argument specifies the delay time
// in seconds.
//
void Delay_Seconds(unsigned char s)
{
  unsigned char i,j;

  for(j = 0; j < s; j++)
  {
    for(i = 0; i <  100; i++)__delay_ms(10);
  }
}


void main()
{
  unsigned char Pattern, Cnt = 0;
   unsigned char SEGMENT[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

   ANSELC = 0;                                  // Configure PORTC as digital
   TRISC = 0;                                   // Configure PORTC as outputs

   for(;;)                                      // Endless loop
   {
     Pattern = SEGMENT[Cnt];                     // Number to send to PORTC
     Pattern = ~Pattern;                        // Invert bit pattern
     PORTC = Pattern;                           // Send to PORTC
     Cnt++;                                     // Increment Cnt
     if(Cnt == 10) Cnt = 0;                     // Cnt is between 0 and 9
     Delay_Seconds(1);                          // 1 s delay
   }
}
```

Figure 5.47: MPLAB XC8 Program Listing.

Primer 6.7

- Merenje frekvencije
  - Metoda I: Time window
  - Metoda II: Merenje vremena između dve uzastopne ivice
- Metoda I: Time window 1s->rezolucija 1Hz
  - Pogodno jer direktno dobijamo vrednost frekvencije

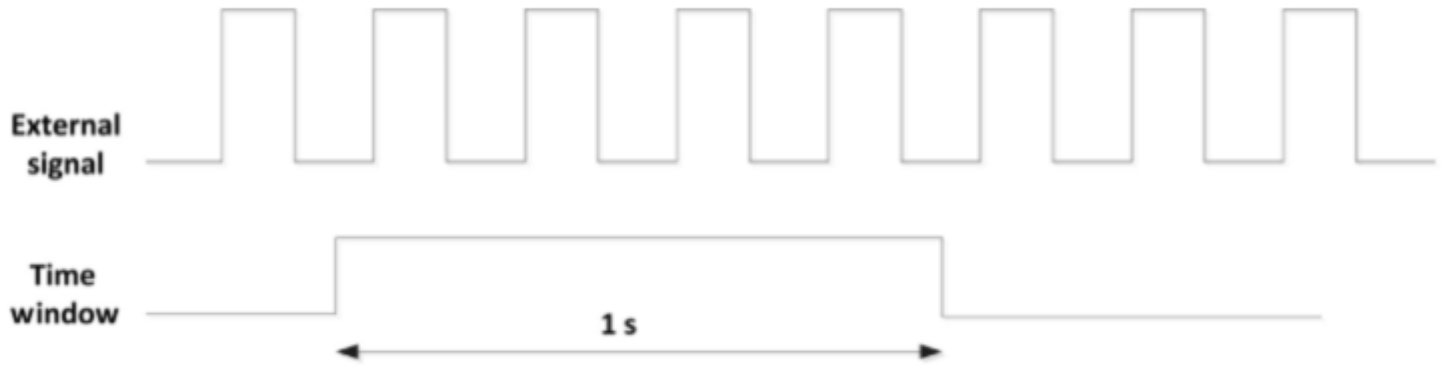
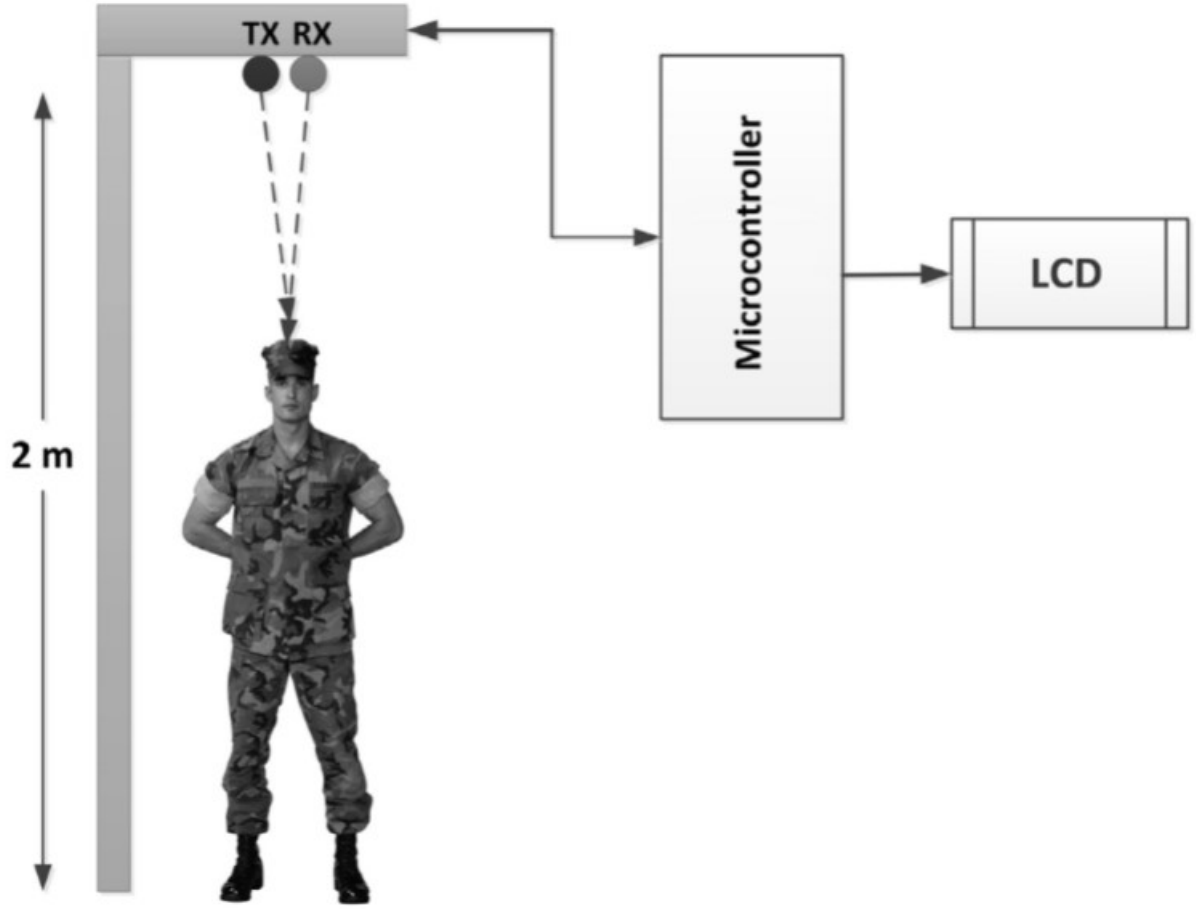


Figure 6.58: Frequency Measurement—Method I.

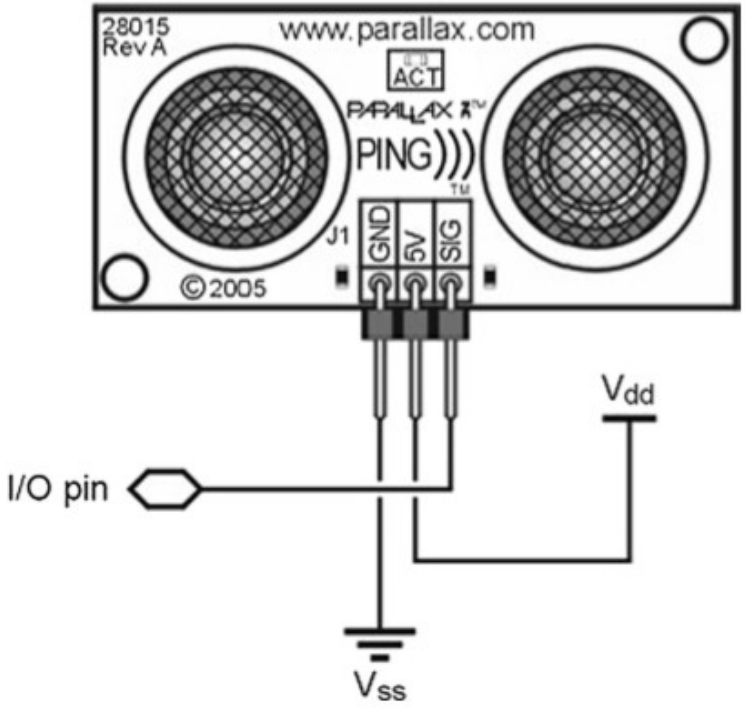
Primer 6.5 – Ultrazvučni merač visine

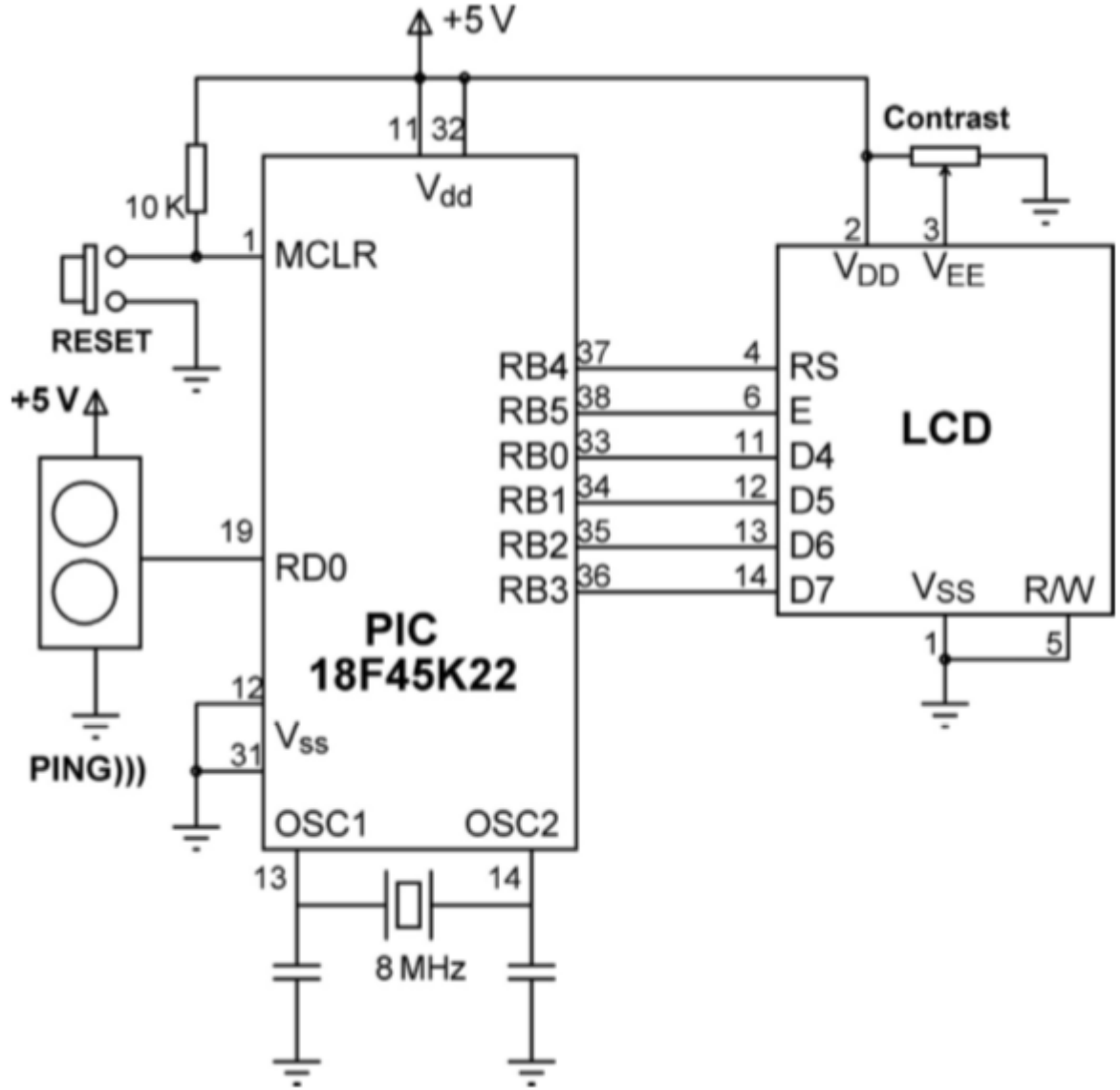


- T=Tm/2 (vreme do objekta u µs)
- Brzina zvuka u vazduhu
  - 340 m/s
  - 34 cm/ms
  - 0,034 cm/µs
- h=0.034* T (s=v*t, h je dato u cm)
- h=34* T/1000 (long int aritemtetika)
- Visina osobe=H-h

Ultrazvučni transducer za merenje visine

- 40kHz radna frekvencija
- 5V, 30mA
- Samo jedan pin za vezu sa mikrokontrolerom
- Meri visinu do 2-3m
- Sa TX se emituju ultrazvučni talasi
- Na RX se prima eho nakon odbijanja

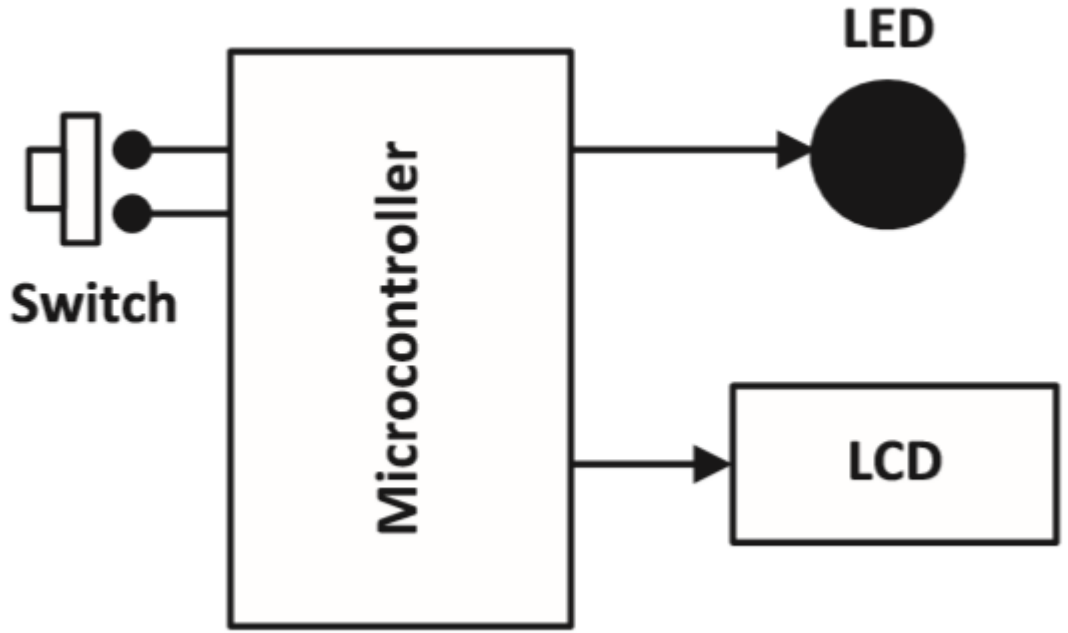



+5 V
11
32
Vdd
10 K
1
MCLR
RESET
+5 V
19
RD0
12
Vss
31
PING)))
OSC1
OSC2
13
14
8 MHz
PIC
18F45K22
RB4
RB5
RB0
RB1
RB2
RB3
37
38
33
34
35
36
Contrast
2
3
VDD
VEE
4
6
11
12
13
14
RS
E
D4
D5
D6
D7
LCD
Vss
R/W
1
5

```
BEGIN
    Define the LCD connections
    Define PING))) connection
    Configure PORTD as digital
    Configure PORTB as digital
    Initialize LCD
    Send start-up message to the LCD
    Configure Timer0 as 16-bit counter with count time of 1 µs
    DO FOREVER
        Send a pulse to the ultrasonic module
        Start timer
        Wait until echo pulse is received
        Calculate the elapsed time
        Divide the time by 2 to find the time to the object
        Calculate the distance to the object (h)
        Calculate the height of the person (H–h)
        Display the height of the person on the LCD
        Wait 1 s
    ENDDO
END
```

Primer 6.8

- Ovaj projekat služi za testiranje bryine reakcije osobe. LED dioda se nakon nasumičnog kašnjenja (1-10s). Na LCD se prikazuje koliko je vremena osobi potrebno da reaguje u milisekundama.



```
BEGIN
    Define interface between the LCD and microcontroller
    Configure PORTB and PORTC as digital
    Configure RC0 as output and RC7 as input
    Initialize LCD
    Configure Timer0 in 16-bit,prescaler 256
    Turn OFF LED
    Initialize random number seed
    DO FOREVER
        Generate random number 1 to 10
        Clear timer registers
        Turn ON LED
        Turn ON timer
        Wait until switch is pressed
        Stop timer
        Read timer count
        Convert count to milliseconds
        Convert count to string
        Display count on LCD
        Turn OFF LED
        Delay 2 s
        Clear LCD
    ENDDO
END
```