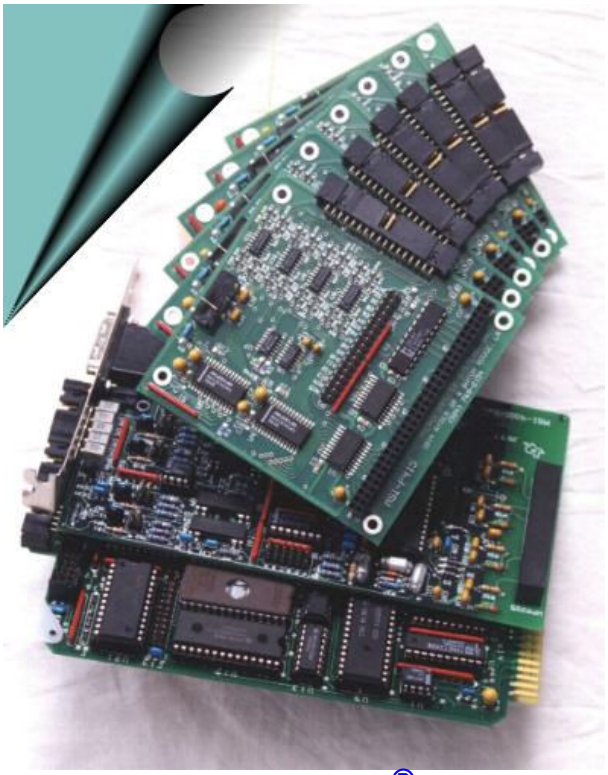
S L O B O M I R P U N I V E R Z I T E T

FAKULTET INFORMACIONIH TEHNOLOGIJA

MIKROPROCESORSKI SISTEMI

Dr Aleksandar Č. ŽORIĆ



PICmicro®

2012.

Autor:
Dr Aleksandar Č. ŽORIĆ, Slobomir P Univerzitet - Fakultet informacionih tehnologija

MIKROPROCESORSKI SISTEMI

Recenzenti:
Dr Milun Jevtić, red. prof. Elektronskog fakulteta Univerziteta u Nišu
Dr Zlatko Bundalo, red. prof. Elektrotehničkog fakulteta Univerziteta u Banja Luci

Izdavač:
Slobomir P Univerzitet - Fakultet informacionih tehnologija, Bijeljina

Računarska obrada teksta i slika:
Dr Aleksandar Č. Žorić

Dizajn korica:
Dr Aleksandar Č. Žorić

Štampa:
MST Gajić, Beograd

Tiraž:
*20*0 primeraka

ISBN 978-86-80893-38-9

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

S A D R Ž A J

P R E D G O V O R

Udžbenik "Mikroprocesorski Sistemi" napisan je sa ciljem da popuni prazninu usled dužeg nedostatka domaće stručne literature koja se odnosi na problematiku Microchip-ovih PIC mikrokontrolera. Knjiga je koncipirana i napisana u skladu sa programom predmeta *Mikroprocesorski Sistemi* koji studenti Fakulteta informacionih tehnologija Slobomir P Univerziteta slušaju na trećoj godini osnovnih akademskih studija. Autor je na osnovu dosadašnjih iskustava siguran da ova knjiga može poslužiti kao udžbenik i studentima elektrotehničkih fakulteta, kao i inženjerima informatike i elektrotehnike u sticanju znanja i veština pri projektovanju ugrađenih sistema. Sadržaj udžbenika je plod višegodišnje nastave koju je autor držao studentima iz predmeta *Mikroprocesorski Sistemi* i obiluje jednostavnijim primerima realizacije i programiranja manje složenih uređaja i sistema zasnovanih na PIC (*Programmable Interface Controller*) MCU (*Micro Controller Unit*). Knjiga je posvećena osmobitnim MCU iz serije PIC16F87X, odnosno, četrdesetopinskom PIC16F877 Microchip-ovom mikrokontroleru koji već izvesno vreme reprezentuje sam vrh popularnih osmobitnih MCU.

Materija u knjizi je organizovana tako da opis arhitekture pomenutog MCU prate i primeri programa pisanih na asemblerskom jeziku i jeziku C. Svi programi su brižljivo napisani i komentarisani sa posebnom pažnjom. Njihova valjanost je proverena na softverskom simulatoru ili na realnom razvojnom sistemu. Najveći broj programa napisan je na C jeziku za šta je korišćen CCS (*Custom Computer Services Inc.*) C kompajler kao integrisano razvojno okruženje.

Udžbeničko štivo je podeljeno na šest poglavlja. Prvo, uvodno poglavlje opisuje esencijalne razlike među pojmovima kao što su mikroprocesor, mikroračunar i mikrokontroler, kriterijume za pravilan izbor MCU-a i postupak razvoja aplikacija na bazi MCU-a.

Drugo poglavlje razmatra softverske razvojne alate PIC mikrokontrolera za razvoj aplikacija, skup i podelu instrukcija, programske jezike i modularno programiranje, rukovanje prekidima i protočnu obradu instrukcija. Sagledane su prednosti modularnog načina programiranja i razmatrani potprogrami, stek i prekidni mehanizam.

Treće poglavlje posvećeno je arhitekturi i organizaciji PIC16F877 MCU. S tim u vezi, najpre su opisane RISC (*Reduced Instruction Set Computer*) i CISC (*Complex Instruction Set Computer*) arhitekture mikrokontrolera i ukratko periferijski moduli a potom detaljno razmatrana organizacija memorijskog prostora i načini adresiranja statičke RAM memorije. Izboru i

konfigurisanju taktnog oscilatora CPU jedinice posvećen je deo poglavlja. Kraj poglavlja odnosi se na organizaciju programskog brojača i konfiguracionu reč MCU.

U četvrtom poglavlju opisani su integrisani sistemi za resetovanje kojima je opremljen MCU. Razmatran je uticaj više različitih vrsta internih i eksternih reset signala na stanja registara mikrokontrolera i izvršavanje programa, kao i ponašanje sistema u toku uspostavljanja napona napajanja. Opisana su, takođe, jednostavna eksterna kola za resetovanje MCU za slučaj kada zbog određenih ograničenja interna integrisana kola ne mogu biti korišćena.

I/O portovi MCU opšte i specijalne namene koji služe za povezivanje MCU sa spoljašnjim okruženjem razmatrani su u petom poglavlju knjige. Detaljno je opisana arhitektura raspoloživih portova i pojedinih namenskih linija portova. Ilustrovana je i programski objašnjena prekidna logika određenih linija porta B MCU-a. Prikazan je i jedan način realizacije matrične tastature 4x3.

Posebno je u okviru ovog poglavlja razmatran port D u radnom režimu paralelnog slave porta (*Parallel Slave Port - PSP*). Data su objašnjenja i ilustracije ciklusa upisa i čitanja PSP potkrepljena programskim rešenjima.

Najobimnije šesto poglavlje ovog udžbenika posvećeno je integrisanim periferijskim podsistemima MCU. Posle uvodnog dela posvećenog tajmerima/brojačima opisani su principi merenja vremena i frekvencije na bazi tajmera/brojača sa primerom realizacije digitalnog frekvencmetra i merača periode periodičnih signala. Od integrisanih standardnih periferijskih uređaja detaljno su razmatrani osmobitni tajmer 0, šesnaestobitni tajmer 1 i osmobitni tajmer 2. Dati su primeri njihove primene u sistemima za upravljanje proizvodnom trakom i mašinom za pakovanje, kao i u sistemu za akviziciju temperature. Tri podmodula (*Capture-Compare-PWM*) u sastavu CCP periferijskog uređaja analizirana su pojedinačno kao i primeri programskih rešenja praktične primene podmodula.

Integrisanom 10-bitnom A/D konvertoru sukcesivnih aproksimacija posvećena je posebna pažnja, zbog činjenice da se radi o važnoj periferiji koja omogućava da se u mikrokontroleru (u digitalnom domenu) prihvataju i obrađuju analogne ulazne veličine. Opisani su važni parametri A/D modula kao što su: brzina konverzije, vreme akvizicije i izbor taktnog generatora.

Kraj ovog poglavlja bavi se analizom integrisanih serijskih komunikacionih interfejsa. Analizirani su asinhroni i sinhroni komunikacioni interfejsi, protokoli i načini sprezanja ovih periferijskih uređaja sa PC računarom i drugim poluprovodničkim uređajima koji su opremljeni određenim komunikacionim interfejsima. Objašnjen je asinhroni serijski prenos podataka koji se realizuje komunikacionim kanalom po standardu RS232 ili RS485, kao i sinhroni serijski prenos zasnovan na integrisanim periferijskim komunikacionim interfejsima IIC (*Inter Integrated Circuit – I^2C*) i SPI (*Serial Peripheral Inteface*).

Gotovo svako izlaganje materije u okvirima potpoglavlja praćeno je sa jednim ili više proverenih primera praktične primene ili realizacije što, prema mišljenju autora, može doprineti jasnoći i boljem razumevanju izloženog štiva.

Autor se srdačno zahvaljuje recenzentima dr Milunu Jevtiću, redovnom profesoru Elektronskog fakulteta Univerziteta u Nišu i dr Zlatku Bundalu, redovnom profesoru Elektrotehničkog fakulteta Univerziteta u Banja Luci na umesnim sugestijama i savetima koji su doprineli kvalitetu ovog dela. Uprkos činjenici da je udžbenik pripreman i realizovan sa velikom pažnjom, autor je svestan mogućih grešaka koje se, gotovo po pravilu, potkradaju i najiskusnijim u ovom poslu. Stoga će svaka dobronamerna i korisna primedba čitalaca biti uzeta u obzir i prihvaćena sa zahvalnošću autora.

Prof. dr Aleksandar Č. ŽORIĆ
Mart 2012

Ova knjiga posvećena je mojoj porodici, supruzi Lidiji i sinovima Čedomiru i Nikoli.
Za godine ljubavi, razumevanja i stalne podrške, kao znak zahvalnosti.

Autor

Poglavlje 1

MIKROPROCESORI, MIKRORAČUNARI ILI MIKROKONTROLERI?

Savremeni integrisani mikroprocesori CPU (Central Processing Unit) generalno su zastupljeni u zahtevnim aplikacijama sofisticiranih performansi gde cena i gabarit dizajna nisu kritični kriterijumi izbora. Velika procesorska snaga modernih mikroprocesora dopušta mogućnost njihove ugradnje u desktop računare i radne stanice gde su softverska kompatibilnost, fleksibilnost, značajna procesorska moć za multiprocesnu obradu, performanse i pouzdanost od značaja. Integrisani blokovi današnjih mikroprocesora (kakav je npr. Pentium IV) pored mikroprocesora opšte namene sadrže i jedinicu za upravljanje memorijskim prostorom, interapt kontroler, matematički procesor, keš memoriju itd. Sve to im omogućava da izvršavaju simultano veći broj kompleksnih i zahtevnih procesa - programa. Sa većim brojem jezgara današnji mikroprocesori uspešno omogućavaju i paralelizam u izvršavanju procesa. Ali ono što njih u osnovi razlikuje od mikrokontrolera je što oni sami ne mogu da funkcionišu kao mikroračunar. Neophodan je dodatni hardver za realizaciju memorijskog prostora - operativne memorije, i brojnih ulazno/izlaznih modula za realizaciju interakcije sa okruženjem - perifernim uređajima.

Termin mikroračunari koristi se za sistem koji minimalno sadrži mikroprocesor, programsku memoriju, memoriju podataka i ulazno-izlazni (I/O) uređaj. Time je omogućeno da mikroračunar realizuje neku korisnu funkciju prema okruženju. Savremeni mikroračunarski sistemi uključuju i

dodatne komponente kao što su tajmeri, brojači, A/D konvertori, komunikacioni interfejsi i slično. Prema tome, pod mikroračunarskim sistemom opšte namene, danas se podrazumeva računar kao što je PC sa perifernim memorijskim uređajima sa magnetnim i optičkim medijumima, periferijama za multimedijalne komunikacije i operativnim sistemom opšte namene. Za razliku od ovih, postoje i mikroračunarski sistemi specijalizovane namene, koji moraju da zadovolje i dodatne stroge zahteve. Takvi su na pimer sistemi za rad u realnom vremenu (*Real Time Systems*) koji mogu da budu i složeniji od PC-a sa specijalizovanim operativnim sistemom za rad u realnom vremenu. Ali, veliki broj specijalizovanih mikroračunarskih sistema se realizuje za konkretnu - jednu namenu, za šta nije neophodan hardver i softver obima PC-a. Za realizaciju ovih namenskih (ugrađenih - *embedded*) mikroračunarskih sistema, od kojih se danas sve više zahteva što manja potrošnja energije, realizovani su mikroračunari na jednom čipu. Integrisan ceo mikroračunar na jednom čipu, kratko nazvan mikrokontroler, danas može biti mikroračunar baziran na 8-bitnom mikroprocesoru sa skromnim memorijskim prostorom i skromnim periferijama (U/I kontrolerima), pa do mikroračunara sa 32- bitnim mikroprocesorom i moćnim pridruženim hardverom.

Ova knjiga posvećena je mikroračunarima integrisanim u jednom silicijumskom čipu, tkzv. mikrokontrolerima MCU (*Micro Controller Unit*) za realizaciju namenskih (ugrađenih) mikroračunarskih sistema. Termin mikro sugeriše veličinu uređaja a kontroler oblast primene (upravljačke aplikacije). Sinonim ugrađeni kontroler upućuje na ugradnju mikrokontrolera u široki spektar moćnih uređaja kojima upravlja, kao što su: kućni elektronski uređaji, automobilski i vojni uređaji, HI-FI audio oprema, celularni telefonski uređaji, bežični kontrolni uređaji, smart kartice i slično. Prosečni automobil danas, poseduje na primer oko 20-tak ugrađenih mikroračunara, (Mercedes S klase iz 1999 godine posedovao je 63 mikrokontrolera a automobil marke BMW iz iste godine 65) dok je tipično srednje elektrificirano domaćinstvo opremljeno sa više od 50-tak mikrokontrolera ugrađenih u različite električne i elektronske uređaje. Američka kompanija *Texas Instruments* proizvela je prvi mikrokontroler serije TMS1000. Četvorobitna centralna procesorska jedinica (CPU) sa dovoljno RAM, ROM memorije i I/O hardverom u jednom čipu ugrađivana je u kalkulatore, mikrotalasne pećnice i industrijske tajmere.

Veliki broj mikrokontrolera različitih proizvođača, različitih arhitektura i performansi, raspoloživ je danas na tržištu. Neki od njih su podesni za krajnje jednostavne aplikacije, dok su drugi namenjeni za upravljanje i nadzor složenih procesa sa zahtevnim obradama podataka. Tako je nastala jedna vrsta mikroračunara na čipu poznata pod nazivom *Data Signal Processors* (DSP) namenjenih obradi analognih električnih signala. Posebna pažnja u ovoj knjizi posvećena je programiranju i projektovanju sistema na bazi PIC serije mikrokontrolera, proizvođača Microchip Technology Inc.

Iz napred rečenog jasno je da su mikroračunari poput PC-a fleksibilniji a mikrokontroleri kompaktniji mikroračunarski sistemi, a da izbor prvenstveno zavisi od polja primene i cene.

1.1. IZBOR MIKROKONTROLERA

Mikroračunar u jednom čipu, poznat pod nazivom mikrokontroler, postaje takav integralni deo života ljudi da se može reći da je 'računarska revolucija' bila najava primene mikrokontrolera u svakodnevnom životu. Naime, na temelju modernog društva, više od pet biliona mikrokontrolera, ugrađenih u široki spektar proizvoda, prodaje se u toku samo jedne godine.

Na tržištu danas, zastupljeno je na stotine raspoloživih mikroprocesora i mikrokontrolera a izbor odgovarajućeg za zadatu aplikaciju može biti ozbiljna poteškoća za projektanta. Uopšteno, polazeći od korisničkih zahteva, funkcionalnosti i karakteristika aplikacije, kao i cene, projektant

je u situaciji da poredi različite mikroračunare raspoložive na tržištu. Međutim, konačan izbor ipak je uslovljen brojnim faktorima kao što su: tržišni trendovi, profil proizvođača, popularnost, ekspertiza lokalnog dizajna i slično.

U tabeli 1.1.1. dat je pregled respektivnih proizvođača popularnih osmobitnih mikrokontrolera najnižih reprezentativnih cena koštanja.

Proizvođač	Uređaj	Memorija u čipu	Ostale karakteristike uređaja
Atmel Corp.	*ATtiny11*	1-kbyte Flash	*8-bitni tajmer, analogni komparator, watchdog tajmer, oscilator u čipu, 1 izvor eksternog prekida,*
Dallas Semi	*DS80C310*	256-byte RAM	*4 takta po instrukcijskom ciklusu, UART, tri 16-bitna tajmera/brojača, dva data pointera, 10 internih/16 eksternih izvora prekida*
Hitachi	*H8/3640*	8-kbyte ROM, 512-byte RAM	*Tri 8-bitna/jedan 16-bitni tajmer, jedan 14-bitni PWM tajmer, watchdog tajmer, dva SCI porta, 8-bitni ADC, 32KHz subtaktni generator*
Infineon	*C501*	8-kbyte ROM, 256-byte RAM	*Serijski komunikacioni interfejs SCI, tri 16-bitna tajmera, 32 I/O porta*
Microchip	*PIC16CR54C*	768-byte ROM, 25-byte RAM	*12 I/O pina velikog strujnog kapaciteta, 8-bitni tajmer, watchdog tajmer, RC oscilator*
Mitsubishi	*M37531M4*	8-kbyte ROM, 256-byte RAM	*2.2V do 5.5V napajanje, tri 8-bitna tajmera, 16-bitni watchdog tajmer, 10-bitni osmokanalni ADC, UART, 1 eksterni izvor prekida, ugrađeni taktni generator*
Motorola	*68HC705KJ1*	1240-byte OTP, 64-byte RAM	*Višefunkcijski tajmer sa 15 stanja, oscilator u čipu, watchdog tajmer, I/O port velikog strujnog kapaciteta*
NEC	*789011*	2-kbyte ROM, 128-byte RAM	*Dva 8-bitna tajmera, UART, 22 I/O porta, dvokanalni serijski interfejs SCI*
Philips	*P87LPC762*	2-kbyte OTP, 128-byte RAM	*Oscilator, UART, watchdog tajmer, 32-bajtna EEPROM za podatke, IIC, analogni komparatori, tajmeri/brojači*
Samsung	*KS860004*	4-kbyte ROM, 208-byte RAM	*Jedan 8-bitni tajmer, jedan 8-bitni tajmer/brojač, RC oscilator, 32 I/O porta, 14 izvora prekida*
Scenix	*SX28AC*	3-kbyte Falsh, 136-byte RAM	*8-bitni tajmer, watchdog tajmer, analogni komparator, programabilni I/O, brown-out detektor*
STMicro	*ST6203CB1*	1-kbyte ROM ili OTP, 64-byte RAM	*8-bitni tajmer, watchdog tajmer, 9 I/O linija visokog strujnog kapaciteta, brown-out kolo*
Toshiba	*TMP87C405AM*	4-kbyte ROM, 256-byte RAM	*9 izvora prekida, programabilni watchdog tajmer, 22 programabilna I/O porta*
Xemics SA	*XE8301*	22-kbyte ROM, 512-byte RAM	*Preskaler taktnog generatora, četiri 8-bitna tajmera sa PWM, watchdog tajmer, UART, kristalni i RC oscilator, 20 programabilnih I/O linija*
Zilog	*Z8E000*	0.5-kbyte OTP, 32-byte RAM	*Jedan 16-bitni tajmer, watchdog tajmer, 13 I/O pina, 4 izvora prekida*

Tabela 1.1.1. *Pregled respektivnih proizvođača popularnih 8-bitnih mikrokontrolera sa najnižom cenom koštanja.*

Kao snažan alat koji projektantu dopušta mogućnost kreiranja sofisticirane manipulacije ulazno-izlaznim podacima pod programskom kontrolom, mikrokontroleri se prema rezoluciji

mogu klasifikovati na 8-bitne, 16-bitne i 32.bitne. Osmobitni mikrokontroleri su najpopularniji i koriste se u najvećem broju aplikacija zasnovanim na mikrokontroleru. Šesnaestobitni i tridesetdvobitni mikrokontroleri su snažniji uređaji veće procesorske snage ali su uobičajeno skuplji i nepotrebni u većem broju aplikacija opšte namene.

Proces projektovanja ugrađenih sistema počinje, kao i većina stvari u životu, od cilja- definicijom krajnjeg proizvoda. Definicija proizvoda pretpostavlja opis ciljnih funkcija i načina rada proizvoda. Kao ključni korak u uspešnom dizajnu bilo kog elektronskog sistema navodi se izrada tehničke dokumentacije. U odnosu na krajnji dizajn dokumentacija treba da sadrži:

- *definiciju zahteva proizvoda,*
- *definiciju funkcionalnih zahteva,*
- *izbor mikrokontrolera,*
- *hardversko-softverske specifikacije,*
- *opis razvoja sistema,*
- *hardverski dizajn,*
- *dizajn softvera,*
- *integraciju,*
- *verifikaciju.*

U toku razvoja sistema često se događa situacija u kojoj se pojavljuje problem sa izborom mikrokontrolera pa se, takođe često, pojedine faze projektovanja moraju ponoviti. Na kraju, proces projektovanja sistema nije uvek moguće deliti. Kada je za razvoj aplikacije izabran mikrokontroler odgovarajuće procesorske moći opredeljenje projektanta za uži izbor proizvođača i odgovarajućeg uređaja treba uskladiti sa:

- *mišljenjem stručne javnosti i popularnošću serije mikrokontrolera,*
- *cenom koštanja,*
- *tehničkim karakteristikama uređaja,*
- *performansama raspoloživog razvojnog alata*
- *potrošnjom uređaja,*
- *gabaritom uređaja,*
- *iskustvom projektanata u radu sa serijom mikrokontrolera.*

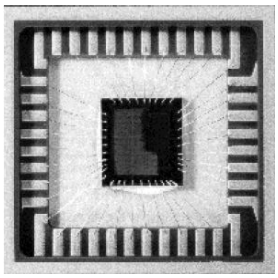
Prema tome, izbor odgovarajućeg mikrokontrolera u funkciji je brojnih činilaca sa gotovo jednakim težinskim udelom i međusobnim uticajem na opredeljenje. Izbor odgovarajućeg mikrokontrolera za datu aplikaciju je uobičajeno i funkcija familijarnosti projektanta sa arhitekturom kontrolera.

1.1.1. Zašto PICmicro®?

U postupku realizacije sofisticiranih elektronskih sistema, odluku o načinu implementacije određenog dela sistema donosi projektant. Dizajneru na raspolaganju stoje diskretna IC kola, PLD kola ili mikroprocesori. Međutim, mnoge aplikacije mogu biti pogodno implementirane primenom mikrokontrolera a mnoštvo njih zasnovano je na PIC familiji Microchip-ovih RISC mikrokontrolera. Napredne karakteristike ovih mikrokontrolera predmet su pažnje narednih poglavlja ove knjige. Međutim, od koristi je na ovom mestu nabrojati neke od značajnih osobina pomenute familije mikrokontrolera kao što su:

1. *RISC arhitektura sa relativno malim brojem instrukcija,*
2. *protočna obrada instrukcija koja povećava brzinu izvođenja instrukcija,*

3. *veliki broj raspoloživih instrukcija koje se izvršavaju u jednom instrukcijskom ciklusu,*
4. *fleksibilni izbor taktnog oscilatora CPU jedinice (interni/eksterni/PLL),*
5. *široki spektar u čipu integrisanih periferija kao što su: A/D konvertor, impulsno-širinski modulator (PWM), tajmeri/brojači, sinhroni i asinhroni serijski komunikacioni interfejsi, USB komunikacioni interfejs, EEPROM memorija, analogni komparatori itd.,*
6. *interna programska memorija (FLASH EEPROM) i memorija podataka (SRAM),*
7. *serijsko programiranje uređaja u sistemu,*
8. *raspoloživi u kućištima od 8 do 40 pinova za prilagođenje širokom spektru aplikacija,*
9. *napon napajanja u opsegu od 2V do 5.5V,*
10. *relativno brzo ovladavanje jednostavnom arhitekturom uređaja.*



Slika. 1.1.1. *Izgled jednog 8-bitnog RISC mikrokontrolera na jedinstvenom supstratu, proizvodnje Microchip Inc.*

1.1.2. Ugrađeni sistemi

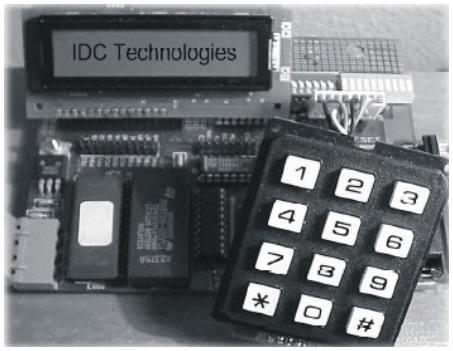
Postoji neverovatno mnogo aplikacija zasnovanih na ugrađenim mikroprocesorskim sistemima, od kojih su neke očigledne a druge manje primetne za većinu ljudi. Ugradnja velikog broja savremenih mikrokontrolera u jednostavne i funkcionalno složene aplikacije, od ekonomično praktičnih do zabavnih i gotovo apsurdnih, uticala je na značajno smanjenje cene ovih uređaja na tržištu. Ipak, cena sistema sa ugrađenim kontrolerima, od slučaja do slučaja, nije uvek razumno prihvatljiva. Prava tržišna revolucija nastala je upravo razvojem i realizacijom ugrađenih mikroprocesorskih sistema pri čemu su mnogi analogni sistemi ubrzo postali deo istorije. Ugrađeni sistemi namenjeni su prvenstveno inteligentnim elektronskim uređajima korišćenim za upravljanje, kao i uređajima za nadzor povezanim sa realnim okruženjem. Primeri takvih sistema su PLC (programabilni logički kontroleri), DCS (distribuirani kontrolni sistemi) i pametni senzori. Savremena elektronska merna instrumentacija poseduje ugrađeni mikroprocesorski sistem kao srce celog sistema.

Mikrokontroleri se generalno koriste za namenske aplikacije a izbor odgovarajućeg dobrim delom je funkcija familijarnosti projektanta sa njegovom arhitekturom. Međutim, razvojni alati za dati mikrokontroler i njihova cena, takođe značajno utiču na opredeljenje projektanta.

1.1.3. Razvoj aplikacija na bazi mikrokontrolera

Jedna od mogućih mapa za dizajniranje aplikacije na bazi mikrokontrolera zasnovana je na sledećim postupcima:

1. *Na prvom mestu definicija korisničkih zahteva.*
2. *Razvoj funkcionalnih specifikacija sistema. Kreiranje dovoljne dokumentacije za podršku zahtevima u opisnoj formi, formi blok dijagrama, dijagrama toka, vremenskih dijagrama itd.*
3. *Pronalaženje podesnog hardvera za podršku neophodnoj funkcionalnosti aplikacije. Ovaj korak bi trebao pomoći projektantu u odluci da li je mikrokontroler neophodan ili nije.*
4. *Identifikacija odgovarajućeg mikrokontrolera kao mozga aplikacije ukoliko je neophodan.*
5. *Kada je izabran odgovarajući mikrokontroler, pažljivo sprovesti dvostruku proveru zadovoljenja zahteva u slučajevima brzine, potrošnje, broja I/O linija, komunikacionih interfejsa mikrokontrolera itd.*
6. *Prikupljanje celokupnog alata za pomoć u razvoju hardvera i softvera. Alat uključuje odgovarajući asembler i/ili kompajler za pisanje i prevođenje programa, odgovrajući simulator za izabrani mikrokontroler, hardverski emulator po potrebi, razvojni sistem, programator i slično.*
7. *Ako je dizajner familijaran sa arhitekturom izabranog mikrokontrolera može se započeti sa asembliranjem prototipa. U suprotnom, dizajner bi morao proučavanjem arhitekture mikrokontrolera, pisanjem programskih primera i testiranjem na razvojnom sistemu ili programskom simulatoru započeti proces familijarizacije.*
8. *Kada dizajner ovlada arhitekturom mikrokontrolera softver treba podeliti na celine koje mogu biti napisane u formi potprograma ili funkcija i testirane nezavisno. Razvoj hardvera može ići paralelno u iteracijama testiranja i otklanjanja grešaka za sve programske i hardverske celine aplikacije. Za kompletiranje pojedinih iteracija potrebno je koristiti opremu za testiranje i razvojne alate kao što su emulator, programator, simulator i slično, i u skladu sa mogućnostima minimizirati broj iteracija.*
   Jedan realističan mikrokontrolerski razvojni sistem prikazan je na slici 1.1.2.



Slika. 1.1.2. *Izgled jednog mikrokontrolerskog razvojnog sistema.*

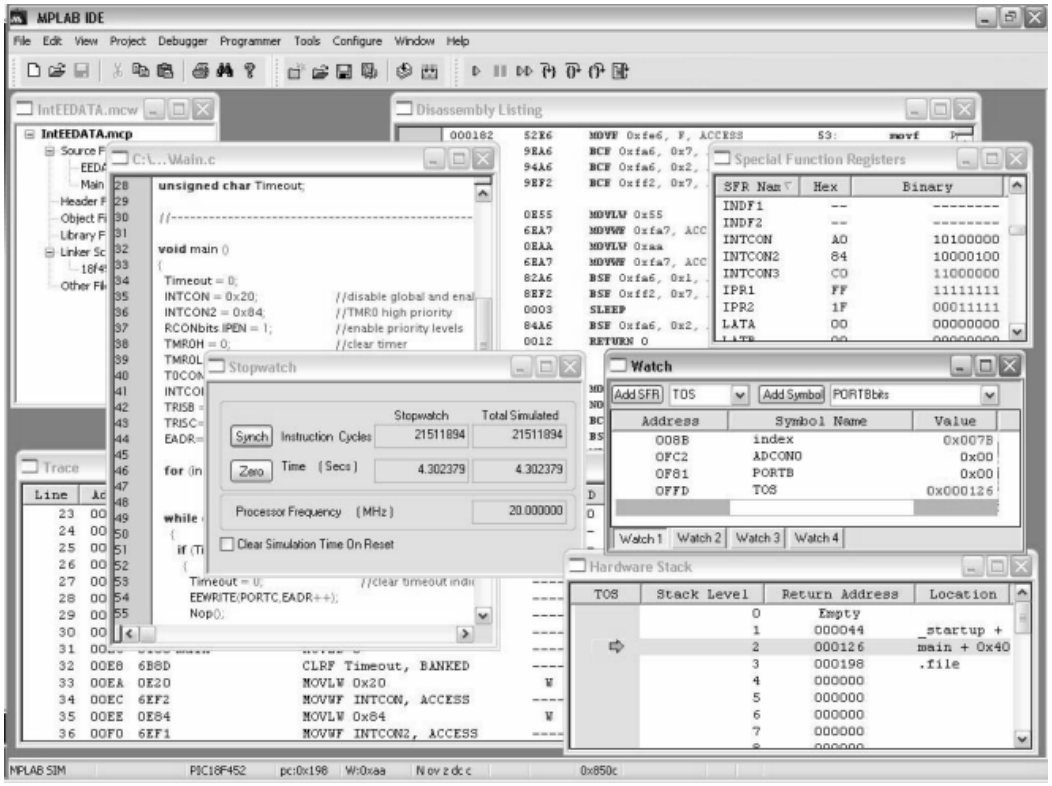
9. *Konačno, integracijom softvera i hardvera potrebno je ponovo izvršiti testiranje i eventualno otklanjanje grešaka kompletne aplikacije.*
10. *U toku pisanja programa i izgradnje hardvera, važna aktivnost koja ne sme biti propuštena je izrada dokumentacije. Dokumentovanje dizajna je ekstremno važno ne samo kao trag projektanta već i za testiranje aplikacije u toku života uređaja i budućih revizija.*
11. *Finalni korak je razvoj sistema u ciljnom okruženju na odgovarajućoj proizvodnoj liniji.*

Poglavlje 2

ALATI ZA RAZVOJ SOFTVERA – PROGRAMIRANJE PIC MCU

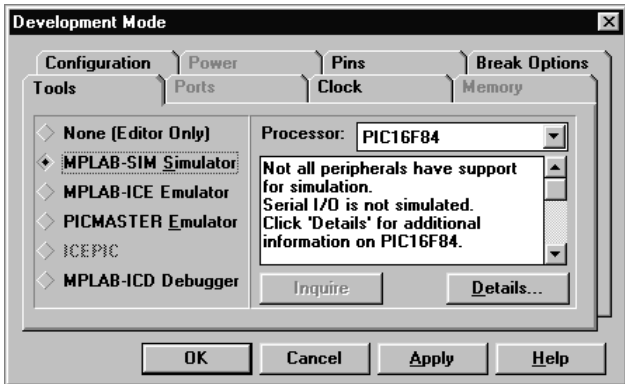
2.1. RAZVOJNI ALATI

Za pisanje i testiranje softvera potreban je dobar razvojni alat. Sa porastom popularnosti PIC familije mikrokontrolera izbor raspoloživog softverskog razvojnog alata je postajao širi i atraktivniji za korisnike. Microchip nudi integrisano razvojno okruženje MPLAB IDE pod Windows operativnim sistemom za PICmicro® MCU sa ugrađenim editorom, asemblerom, linkerom i prevodiocem, mogućnošću integracije višeg programskog jezika, programskim simulatorom aplikacija, mogućnošću pokretanja aplikacije na emulatoru i softverom za programiranje mikrokontrolera. Sve nabrojano u jednom integrisanom okruženju predstavlja izvanredan alat sa mogućnošću da korisnik monitoriše izvođenje aplikacije i odziv pod različitim ulaznim uslovima. Na slici 2.1.1. a) prikazan je izgled više otvorenih prozora paketa MPLAB IDE, dok je na slici 2.1.1. b) dat izgled jednog od pomoćnih prozora za podešavanje moda.

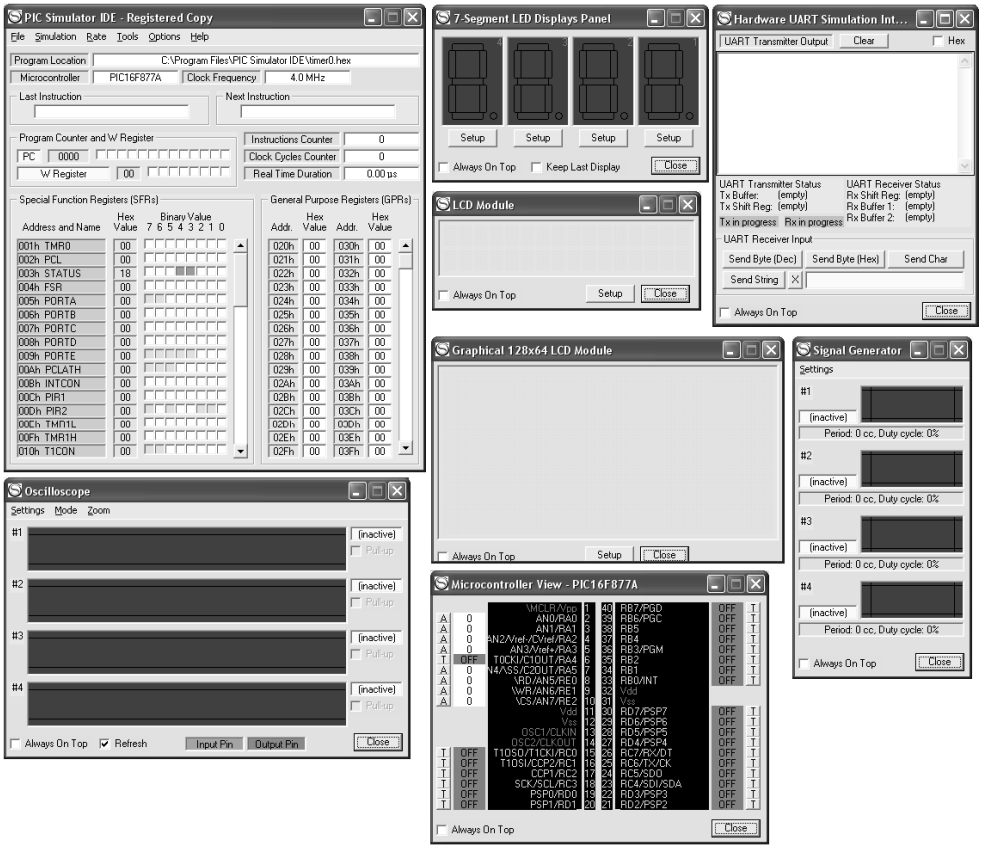


Slika 2.1.1. a). *Integrisano softversko razvojno okruženje za PICmicro® MCU MPLAB IDE (Microchip).*

Osim MPLAB IDE paketa korisniku na raspolaganju stoji mnoštvo drugih softverskih alata kao što su: PIC Simulator IDE - izvanredno integrisano okruženje simulatora namenjeno serijama PIC12FXXX i PIC16FXXX sa BASIC prevodiocem, asemblerom, disasemblerom, generatorom HEX datoteke, interaktivnim asemblerskim editorom i većim brojem simuliranih I/O uređaja, PIC18 Simulator IDE namenjeno seriji mikrokontrolera PIC18FXXXX, CCS C kompajler za brzi razvoj aplikacionog softvera na višem programskom jeziku C, slika 2.1.4., koji uključuje tri odvojena prevodioca - PCB za 12.bitni operacioni kod, PCM za 14-bitni opkod i PCH za 16-bitni opkod PIC mikrokontrolera, kao i softver za programiranje PICmicro® uređaja IC-Prog namenjen većem broju različitih vrsta programatora, slika 2.1.5.

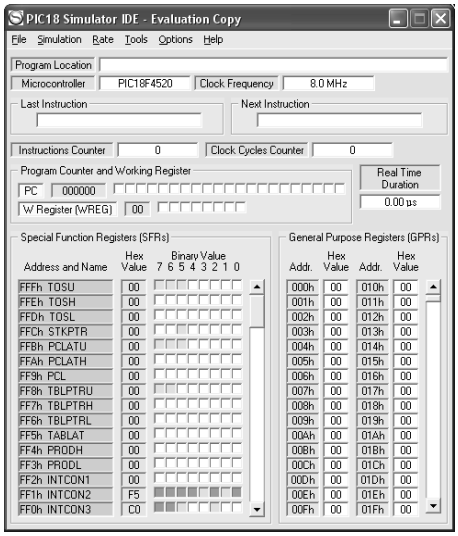


Slika 2.1.1. b). *Izgled prozora razvojnog moda paketa MPLAB IDE za izbor procesora.*

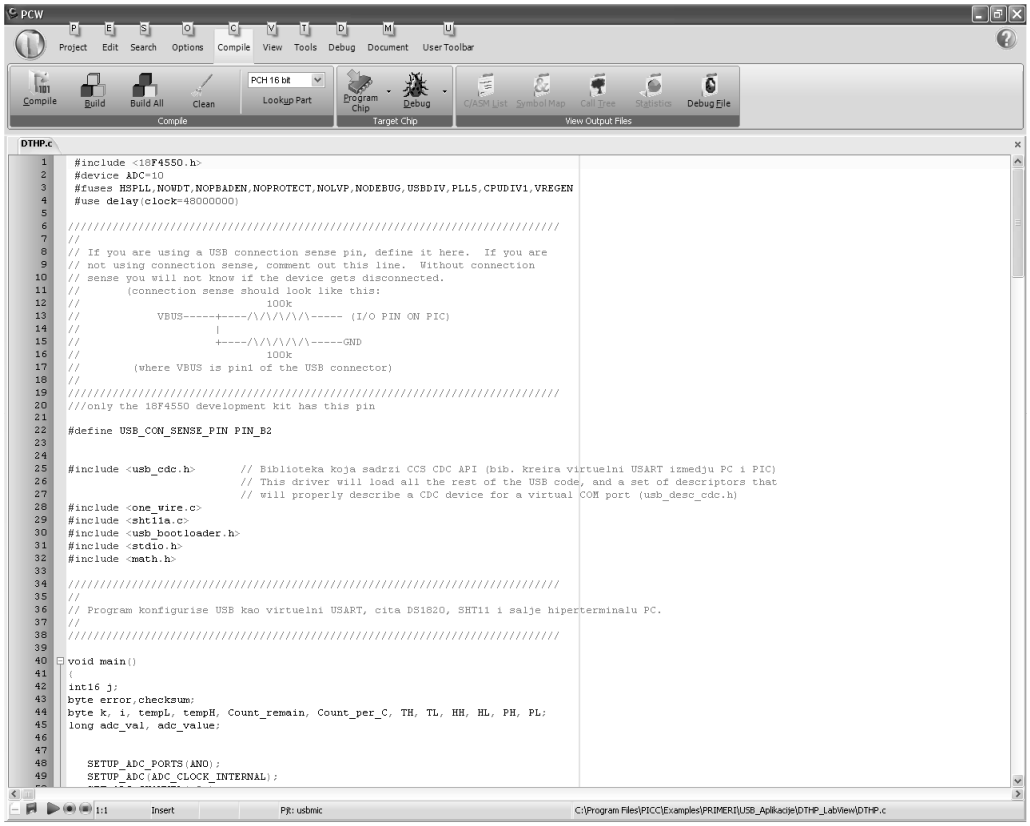


Slika 2.1.2. *PIC Simulator IDE*.

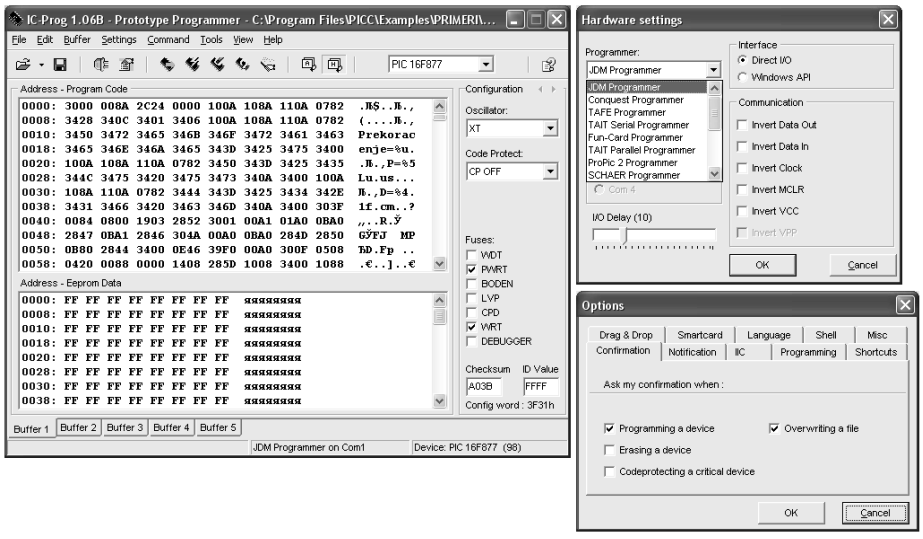
Na slici 2.1.2. prikazan je izgled PIC simulatora IDE sa prozorima simuliranih I/O uređaja i portovima izabranog mikrokontrolera PIC16F877A a na slici 2.1.3. izgled glavnog prozora istog simulatora namenjenog seriji PIC18FXXXX mikrokontrolera.



Slika 2.1.3. *PIC18 Simulator IDE*.



Slika 2.1.4. *Editor i radno okruženje CCS C kompajlera za PICmicro®.*



Slika 2.1.5. *Izgled prozora alata IC-Prog za programiranje PICmicro®.*

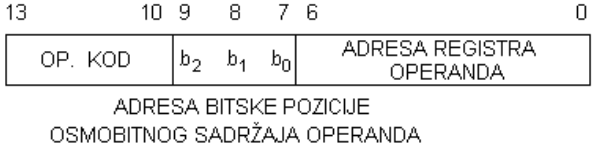
2.2. PROGRAMIRANJE PIC16F877 MCU

2.2.1. Skup i podela instrukcija

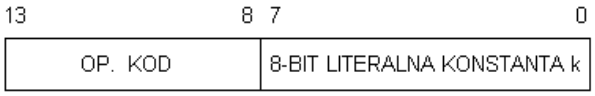
Mikrokontroler PIC16F877 poseduje skup od 35 četrnaestobitnih instrukcija od kojih se većina izvršava u jednom instrukcijskom ciklusu izuzev instrukcija grananja programa za čije izvođenje su potrebna dva ciklusa. Sve instrukcije mogu se podeliti na tri grupe: bajt orijentisane, bit orijentisane i literalne i kontrolne instrukcije/operacije. Na slici 2.2.1. prikazan je generalni format tri grupe instrukcija.

Format bajt-orijentisanih instrukcija

Format bit-orijentisanih instrukcija



Opšti format literalnih i kontrolnih instrukcija



Specifičan format kontrolnih instrukcija CALL i GOTO

Slika 2.2.1. *Formati tri grupe instrukcija MCU PIC16F877.*

Sedam najnižih bita bajt orijentisanih operacija predstavlja adresu registra operanda što znači da se može adresirati 2^7=128 registara. Bit označen sa d je odredišni bit operacije tako da je za d=0 rezultat operacije u akumulatoru (radni registar w), dok je za d=1 rezultat u registru operanda. Podrazumevana vrednost ovog bita u slučaju da se izostavi je d=1. Širina polja operacionog koda bajt orijentisanih instrukcija je šest bita na najvišoj poziciji u instrukcijskoj reči. Prema tome, ukupan broj bajt orijentisanih operacija može biti 2^6=64, od čega je implementirano 18.

Kao i kod grupe bajt orijentisanih operacija, sedam najnižih bita bit orijentisanih operacija predstavlja adresu registra operanda. Sledeća tri bita predstavljaju adresu bitske pozicije (0 do 7) osmobitnog sadržaja registra operanda (2^3=8), dok najviša četiri bita pripadaju operacionom kodu ove grupe instrukcija (maksimalan broj ovih instrukcija može biti 2^4=16).

Literalne i kontrolne operacije su prostijeg formata. Osam najnižih bita ovih operacija predstavlja literalnu konstantu (0 do 255) dok je operacioni kod šestobitni. Izuzetak čine kontrolne operacije koje se odnose na instrukcije poziva potprograma CALL i bezuslovnog skoka GOTO koje imaju nešto drugačiji format. Naime, literalna konstanta ovih instrukcija je 11-bitna i predstavlja simboličku adresu prve instrukcije potprograma ili bezuslovnog skoka, dok je operacioni kod ovih instrukcija trobitni. Budući da je adresa skoka 11-bitna to se ovim operacijama može doseći prostor programske memorije ne veći od 2^{11}=2048 instrukcijskih reći (dve kilo reči) u odnosu na tekuću kontrolnu instrukciju skoka.

U tabeli 2.2.1. prikazan je potpuni skup instrukcija podeljenih u tri grupe. U prvoj koloni prikazane su mnemoničke instrukcije koje sadrže operande i odredišni bit d, dok je u drugoj koloni dat opis svake instrukcije. Broj instrukcijskih ciklusa potrebnih za izvođenje svake od instrukcija dat je u trećoj koloni tabele. Za izvođenje uslovnih instrukcija (npr. DECFSZ f,d) potrebna su jedan ili dva instrukcijska ciklusa u zavisnosti od kondicionalnog testa. Ako je kondicionalni test istinit instrukcija se izvršava u dva ciklusa pri čemu je drugi ciklus NOP operacija. Isto se događa u slučaju modifikacije sadržaja programskog brojača koja je rezultat neke instrukcije (npr. CALL k). Četvrta kolona opisuje 14-bitni operacioni kod instrukcije gde je sa d označen odredišni bit čija vrednost je d=1 ako je odredišni registar operand f a d=0 ako je to akumulator w. Sa f su označena sedam bita adrese operanda dok je x vrednost bita koja može biti proizvoljna 0 ili 1. Sa tri bita označena sa b u skupu bit orijentisanih instrukcija kodira se jedna od osam bitskih pozicija 8-bitnog sadržaja operanda (0 do 7), dok je u skupu literalnih i kontrolnih instrukcija sa k označena 8-bitna ili 11-bitna literalna konstanta.

U poslednjoj petoj koloni tabele navedeni su bitovi statusnog registra CPU na koje neke instrukcije mogu imati uticaj.

Koristeći operacione kodove instrukcija date u tabeli 2.2.1. moguće je direktno, bez posredstva prevodioca, napisati izvršni binarni oblik programa uz poznavanje adresa operanada koje čine registri opšte i posebne namene, kao u sledećem primeru. Neka su heksadecimalne adrese korisničkih promenljivih, za koje je rezervisan prostor u RAM memoriji, S_COPY, P_COPY i W_COPY redom 0x70h, 0x71h i 0x72h a adrese registara posebne namene, kao što

su STATUS i PCLATH redom 0x03h i 0x0Ah. Za asemblerski potprogram sa labelom PUSH, odgovarajući 14-bitni binarni ekvivalenti mnemoničkih instrukcija, dobijeni iz tabele 2.2.1., prikazani su istovremeno.

Kompletan instrukcijski set

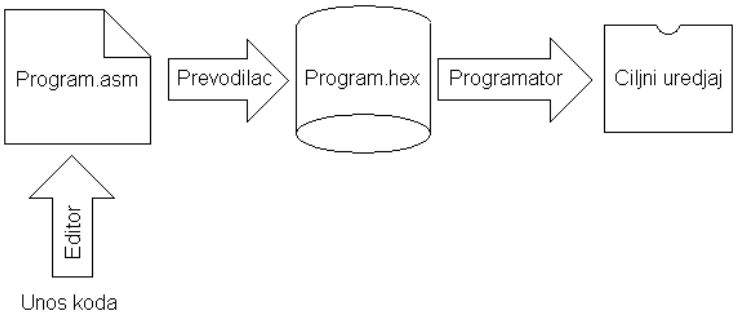
Mnemonik, Operandi		OPIS INSTRUKCIJE	Broj ciklusa	14-bit Opkod MSb			LSb	Uticaj na Stat. Reg.
BAJT-ORIJENTISANE OPERACIJE								
ADDWF	f, d	Sabiranje sadržaja registara W i f	1	00	0111	dfff	ffff	C,DC,Z
ANDWF	f, d	AND operacija nad bitovima registara W i f	1	00	0101	dfff	ffff	Z
CLRF	f	Brisanje sadržaja registra f	1	00	0001	1fff	ffff	Z
CLRW	-	Brisanje sdržaja registra W	1	00	0001	0xxx	xxxx	Z
COMF	f, d	Operacija komplementiranja bitova registra f	1	00	1001	dfff	ffff	Z
DECF	f, d	Dekrementiranje sadržaja registra f	1	00	0011	dfff	ffff	Z
DECFSZ	f, d	Dekrementiranje sadržaja registra f-skok ako je on 0.	1(2)	00	1011	dfff	ffff	
INCF	f, d	Inkrementiranje sadržaja registra f	1	00	1010	dfff	ffff	Z
INCFSZ	f, d	Inkrementiranje sadržaja registra f-skok ako je on 0	1(2)	00	1111	dfff	ffff	
IORWF	f, d	OR operacija nad bitovima registara W i f	1	00	0100	dfff	ffff	Z
MOVF	f, d	Prenos sadržaja registra f	1	00	1000	dfff	ffff	Z
MOVWF	f	Prenos sadržaja registra W u registar f	1	00	0000	1fff	ffff	
NOP	-	Nema operacije izuzev trošenja instrukcijskog ciklusa	1	00	0000	0xx0	0000	
RLF	f, d	Rotiranje sadržaja registra f ulevo kroz Carry bit	1	00	1101	dfff	ffff	C
RRF	f, d	Rotiranje sadržaja registra f udesno kroz Carry bit	1	00	1100	dfff	ffff	C
SUBWF	f, d	Oduzimanje sadržaja registra W od registra f	1	00	0010	dfff	ffff	C,DC,Z
SWAPF	f, d	Zamena polubajtova registra f	1	00	1110	dfff	ffff	
XORWF	f, d	Ekskluzivna OR operacija nad bitovima reg. W i f	1	00	0110	dfff	ffff	Z
BIT-ORIJENTISANE OPERACIJE								
BCF	f, b	Resetovanje bita b registra f	1	01	00bb	bfff	ffff	
BSF	f, b	Setovanje bita b registra f	1	01	01bb	bfff	ffff	
BTFSC	f, b	Testiranje bita b registra f-skok ako je on 0	1 (2)	01	10bb	bfff	ffff	
BTFSS	f, b	Testiranje bita b registra f-skok ako je on 1	1 (2)	01	11bb	bfff	ffff	
LITERALNE I KONTROLNE OPERACIJE								
ADDLW	k	Sabiranje literalne konstante k sa sadržajem registra W	1	11	111x	kkkk	kkkk	C,DC,Z
ANDLW	k	AND op. nad literalnom konst. k i sadržajem registra W	1	11	1001	kkkk	kkkk	Z
CALL	k	Poziv potprograma na simboličkoj adresi k	2	10	0kkk	kkkk	kkkk	
CLRWDT	-	Brisanje registra Watchdog tajmera	1	00	0000	0110	0100	TO,PD
GOTO	k	Bezuslovni skok na simboličku adresu k	2	10	1kkk	kkkk	kkkk	
IORLW	k	OR op. nad literalnom konst. k i sadržajem registra W	1	11	1000	kkkk	kkkk	Z
MOVLW	k	Prenos literalne konstante k u registar W	1	11	00xx	kkkk	kkkk	
RETFIE	-	Povratak iz prekidne rutine sa omogućenim glob. prek.	2	00	0000	0000	1001	
RETLW	k	Povratak iz potprog. sa lit. konst. k u registru W	2	11	01xx	kkkk	kkkk	
RETURN	-	Povratak iz potprograma	2	00	0000	0000	1000	
SLEEP	-	Uvođenje u standby režim rada	1	00	0000	0110	0011	TO,PD
SUBLW	k	Oduzimanje sadržaja registra W od lit. konst. k	1	11	110x	kkkk	kkkk	C,DC,Z
XORLW	k	Ekskluzivna OR op. nad lit. konst. k i sadržajem reg. W	1	11	1010	kkkk	kkkk	Z

Tabela 2.2.1. *Kompletan skup instrukcija mikrokontrolera PIC16F877.*

PUSH:		
MOVWF	W_COPY	00 0000 1111 0010
SWAPF	STATUS, 0	00 1110 0000 0011
MOVWF	S_COPY	00 0000 1111 0000
CLRF	STATUS	00 0001 1000 0011
MOVF	PCLATH, 0	00 1000 0000 1010
MOVWF	P_COPY	00 0000 1111 0001
CLRF	PCLATH	00 0001 1000 1010
RETURN		00 0000 0000 1000

2.2.2. Asemblerski jezik

Asemblerski jezik predstavlja simbolički mašinski jezik koji umesto binarnih ili heksadecimalnih operacionih kodova instrukcija koristi sugestivne mnemoničke instrukcije, jednostavne za pamćenje, čije delove čine i simbolički predstavljeni operandi. Na slici 2.2.2. prikazan je tok generisanja apsolutnog koda kojim se puni programska memorija MCU. Kada je izvorna datoteka asemblirana na ovaj način, sve promenljive i rutine korišćene u izvornoj datoteci moraju biti definisane unutar nje ili u posebnim datotekama koje eksplicitno moraju biti uključene u izvornu. Ako proces prevođenja protekne bez grešaka generiše se hex datoteka koja sadrži izvršni mašinski kod za punjenje programske memorije ciljnog uređaja. Ova datoteka može biti korišćena zajedno sa debagerom za testiranje koda pri izvršenju ili sa programatorom za programiranje ciljnog uređaja.

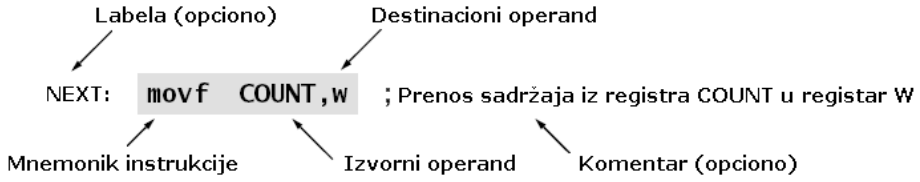


Slika 2.2.2. *Način generisanja apsolutnog koda.*

MPASM asembler/prevodilac tipično je uključen u Microchip-ovo integrisano razvojno okruženje MPLAB IDE. Proces prevođenja izvorne asemblerske datoteke *.asm generiše datoteke tipa: HEX, ERR i LST. Prvom se puni programska memorija mikrokontrolera, druga sadrži spisak greški i upozorenja. Treća je najkorisnija za programera jer sadrži, adrese, programski kod, greške, dodatne informacije poput grafičkog zauzeća memorije itd.
Osnovni elementi asemblerskog jezika su sledeći:

- **labele,**
- **instrukcije,**
- **operandi,**
- **direktive i**
- **komentari.**

Labele su tekstualni opisi nekog dela programa. Obično se koriste kao tačka u programu gde će doći do skoka. Labela mora početi u prvoj koloni sa abecednim znakom ili sa znakom "_", završava se sa ":" a maksimalna dopuštena dužina labele je 32 karaktera.
Instrukcije ili naredbe se pišu za određenu seriju mikrokontrolera po pripadajućim sintaksnim pravilima.
Operandi su delovi instrukcije i označavaju registar, varijablu ili konstantu.
Direktive su asemblerske komande koje se pojavljuju u izvornom kodu ali se uobičajeno ne prevode u operacione kodove. Koriste se za kontrolu asemblerskog programa i kontrolu alokacije podataka. Ne zavise od tipa mikrokontrolera.
Komentari se unose radi jasnoće i dokumentovanja a pišu se iza instrukcija ili u praznom redu sa obaveznim znakom ";" na početku. Ne prevode se.



Slika 2.2.3. *Format tipične linije izvornog koda.*

U nastavku teksta, tabela 2.2.2., opisana je principijelna struktura tipičnog asemblerskog programa.

```
;*****************************************************************
;  Ova datoteka predstavlja šablon asemblerskog programa       *
;  za PICmicro PIC16F877 MCU. Datoteka sadrži osnovne          *
;  asemblerske gradivne blokove.                               *
;                                                              *
;  Ako se prekidi ne koriste, kod prikazan između ORG 0x04     *
;  direktive i labele Main može biti uklonjen. Takođe se mogu  *
;  ukloniti i promenljive w_copy, s_copy i p_copy.             *
;                                                              *
;*****************************************************************
;                                                              *
;   Ime datoteke:         xxx.asm                              *
;   Datum:                                                     *
;   Verzija programa:                                          *
;                                                              *
;   Autor:                                                     *
;   Preduzeće:                                                 *
;                                                              *
;                                                              *
;*****************************************************************
;                                                              *
;   Uključne datoteke:                                         *
;                                                              *
;                                                              *
;*****************************************************************
;                                                              *
;   Komentari:                                                 *
;                                                              *
;*****************************************************************

      LIST     p=16f877          ; listing direktiva za definiciju procesora.
#INCLUDE <p16f877.inc>          ; kontrolna direktiva za uključenje dodatne datoteke-datoteke za deklaraciju simboličkih i fizičkih
                                ; adresa SPR (Special Purpose Registers) registara (promenljivih) i odgovarajućih bitova.

      __CONFIG _CP_OFF & _WDT_ON & _BODEN_ON & _PWRTE_ON & _RC_OSC & _WRT_ENABLE_ON & _LVP_ON &
_DEBUG_OFF & _CPD_OFF
```

```
; '__CONFIG' direktiva za podatke-koristi se za ugradnju konfiguracionih podataka unutar .asm datoteke.
; Labele koje slede direktivu sadržane su u odgovarajućoj .inc datoteci.


;******** Definicija promenljivih ********

W_COPY      EQU   0x70        ; promenljiva W_COPY koja se koristi pri zameni konteksta procesa.
S_COPY      EQU   0x71        ; promenljiva S_COPY koja se koristi pri zameni konteksta procesa.
P_COPY      EQU   0x72        ; promenljiva P_COPY koja se koristi pri zameni konteksta procesa.

; kontrolna direktiva EQU je opšteg oblika <label> equ <expr> i ima sledeće značenje: Vrednost <expr> pridružuje se simbolu
;<label>. U ovom slučaju to su adrese gornje tri korisničke promenljive.


;********* Adresa reset vektora *********

      ORG         0x00        ; kontrolna direktiva označava startnu adresu programa - adresa reset vektora prog. mem.
      CLRF        PCLATH      ; brisanje PCLATH registra obezbeđuje selekciju stranice 0 programske memorije.
      GOTO        MAIN        ; bezuslovni skok na početak glavnog programa.

;************ Skok na adresu prekidnog vektora i kod za spašavanje konteksta glavnog programa **************

      ORG         0x04        ; adresa prekidnog vektora na koju se program grana po pojavi prekidnog zahteva.
      MOVWF       W_COPY      ; sačuvaj aktuelni sadržaj registra w u w_copy (kopija w registra).
      SWAPF       STATUS, W   ; prenesi sadržaj svopovanog status registra u w registar.
      MOVWF       S_COPY      ; sačuvaj aktuelni sadržaj registra w u s_copy (kopija svopovanog status registra).
      MOVF        PCLATH, W   ; prenesi sadržaj pclath registra u w registar.
      MOVWF       P_COPY      ; sačuvaj aktuelni sadržaj registra w u p_copy (kopija pclath registra).

; gornji skup instrukcija koristi se za čuvanje konteksta prekinutog procesa, sadržaja registara w, status i pclath, dok se sadržaj PC
; automatski pamti u steku. Instrukcija swapf koristi se umesto instrukcije movf koja može uticati na Z bit statusnog registra.

;************ Nastavak koda pripada rutini za obradu prekida **************

; ISR:
; izvorni kod prekidne servisne rutine sa labelom ISR započinje na ovom mestu. Kod ISR rutine može biti lociran i na drugom mestu,
; do čije prve instrukcije se može doći naredbom bezuslovnog skoka.
; posle izvođenja koda prekidne servisne rutine sledi niz instrukcija za rekonstrukciju sadržaja registara w, status i pclath kao pre
skoka na adresu prekidnog vektora, dok se sadržaj PC automatski rekonstruiše iz steka.

;************ Kod za rekonstrukciju konteksta prekinutog glavnog programa i povratak iz prekidne rutine *************

      MOVF        P_COPY, W   ; prenesi kopiju PCLATH registra u w.
      MOVWF       PCLATH      ; obnovljeni sadržaj PCLATH registra odgovara onom pre skoka na prekidni vektor.
      SWAPF       S_COPY, W   ; svopuj prethodno svopovanu kopiju STATUS registra i prenesi je u w.
      MOVWF       STATUS      ; obnovljeni sadržaj STATUS registra odgovara onom pre skoka na prekidni vektor.
      SWAPF       W_COPY, F
      SWAPF       W_COPY, W   ; dvostrukom primenom instrukcije swapf koja ne utiče na bitove statusnog registra obnavlja
                              ; se sadržaj registra w koji odgovara onom pre skoka na prekidni vektor.
      RETFIE                  ; povratak iz ISR u glavni program na mesto prekida.

;************ Kod glavnog programa *************

MAIN:

; početak koda glavnog programa.


END                           ; kontrolna direktiva za indikaciju kraja programa.
```

Tabela 2.2.2. *Struktura tipičnog asemblerskog programa.*

MPASM asembler podržava sledeće radix forme (brojne osnove): heksadecimalna, decimalna, oktalna, binarna i ASCII. Podrazumevana radix forma je heksadecimalna. Ova forma radix-a određuje formu dodeljenih konstanti u object fajlu kada radix nije eksplicitno specificiran asemblerskom direktivom. Tabela pokazuje različite radix specifikacije, njihovu sintaksu i primere korišćenja.

Tip	Sintaksa	Primer
Decimalni	D'<dec_cifre>' .<dec_cifre>	D'100' .100
Heksadecimalni	H'<hex_cifre>' 0x<hex_cifre>	H'9f' 0x9f
Oktalni	O'<oktalne_cifre>'	O'777'
Binarni	B'<binarne_cifre>'	B'00111001'
ASCII	A'<karakter>' '<karakter>'	A'C' 'C'

Tabela 2.2.3. *Tabela različitih radix specifikacija.*

Četiri niža bita (4-bit) I I V	*Tri viša bita (3 bit)* --->							
Hex	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	NUL	DLE	Space	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	Bell	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Tabela 2.2.4. *Tabela 7-bitnih ASCII karaktera.*

2.2.3. Viši programski jezici

Za razliku od asemblera, viši programski jezici nude mogućnost brzog, konformnog, preglednog i efikasnog dizajniranja aplikacionog softvera što, u krajnjem, ima uticaj na cenu aplikacije. Primena viših programskih jezika u mikrokontrolerskim aplikacijama podržana je od strane proizvođača koji, u skladu sa povećanjem radne brzine mikrokontrolera, obezbeđuju i veću programsku i RAM memoriju ovih komponenata. Jedan program napisan na asembleru troši tipično 80% od veličine istog programa u C verziji. Savremeni razvojni sistemi za PIC mikrokontrolere u osnovi nude projektantu iste mogućnosti kao PC zasnovano razvojno okruženje, izuzev grafičkih biblioteka. Razvoj proizvoda kombinacija je znanja i iskustva

dizajnera. Programiranje mikrokontrolera na višem programskom jeziku može biti naizgled težak zadatak uz inicijalni trošak za odgovarajući kompajler, emulator i neophodni hardver, što je otežavajuće za procenu cene projekta.

Na tržištu je raspoloživo nekoliko C kompajlera za PIC seriju mikrokontrolera koji imaju veoma slične karakteristike i koji se često koriste u komercijalne, industrijske i edukacione svrhe, kao što su:

- mikroC
- PICC18
- C18
- CCS C

Popularni i snažni mikroC kompajler, proizvod mikroElektronike (*www.mikroe.com*), jednostavan je za učenje, poseduje veliki broj bibliotečkih funkcija i integrisano razvojno okruženje sa ugrađenim simulatorom i debagerom (*In Circuit Debager*-MikroICD). Demo verzija kompajlera sa ograničenjem za veličinu programa do 2K raspoloživa je na sajtu firme mikroElektronika.

PICC18, drugi popularni C kompajler, razvijen je od strane firme HI-Tech Software (*www.htsoft.com*) i raspoloživ u dve verzije: standardna i profesionalna. Snažan simulator i integrisano razvojno okruženje (*Hi-Tide*) glavne su karakteristike kompajlera. PICC18 je podržan od strane PROTEUS simulatora (*www.labcenter.co.uk*) koji može biti korišćen za simulaciju sistema zasnovanih na PIC mikrokontrolerima. Vremenski limitirana demo verzija ovog kompajlera raspoloživa je na sajtu proizvođača.

C18 je proizvod firme Microchip Inc. (*www.microchip.com*), uključuje simulator i hardversko-softverski razvojni alat kao što je emulator (ICE200) i debager (ICD2). Na sajtu proizvođača raspoloživa je vremenski limitirana demo verzija kompajlera kao i verzija sa limitiranom funkcionalnošću koja nije vremenski limitirana.

CCS C kompajler proizvod je firme *Custom Computer Services Inc*. (*www.ccsinfo.com*). Poseduje veliki broj ugrađenih bibliotečkih funkcija i podršku debageru ICD-U40. Ovi alati omogućavaju projektantu brzo i efikasno dizajniranje aplikacionog softvera za Microchip-ove mikrokontrolere u višem programskom jeziku C, istovremeno poboljšavajući preglednost i čitljivost programa. PCB, PCM, PCH i PCD su odvojeni prevodioci za 12-bitne, 14-bitne, 16-bitne i 24-bitne operacione kodove PICmicro®, respektivno. Proizvođač nudi vremenski ograničenu demo verziju ovog kompajlera.

Pomenuti C kompajleri dopuštaju mogućnost uključenja asemblerskih jezičkih instrukcija u C program za npr. realizaciju striktnog programskog kašnjenja i slično.

Osim pomenutih C kompajlera na tržištu su raspoloživi mikroBasic i mikroPascal kompajleri, proizvodi firme mikroElektronika, koji programerima familijarnim sa programskim jezicima Basic i Pascal pružaju mogućnost konformnog programiranja PICmicro® uređaja.

U ovoj knjizi je za razvoj aplikacija korišćen CCS C kompajler, detaljnije diskutovan u sledećim poglavljima.

2.2.4. Modularno programiranje - potprogrami i hardverski stek

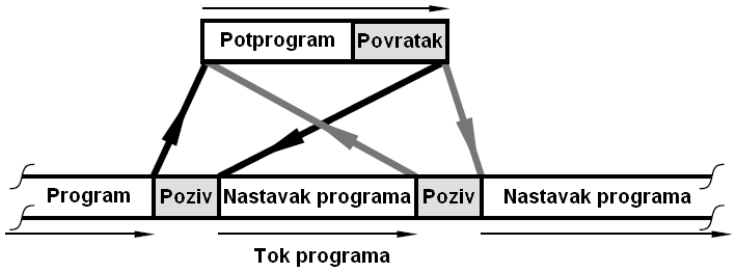
Dobar softver treba da bude konfigurisan kao skup interaktivnih programskih modula a ne kao jedan veliki program koji se izvršava od svog početka pa do kraja. Postoje mnoge prednosti modularnog programiranja, koje posebno dolaze do izražaja kada veličina programa premaši nekoliko stotina linija ili u slučaju timskog razvoja projekta. Modularno programiranje koristi određene principe konstrukcije programa. Može se definisati kao jedan pristup programiranju pri kome su odvojeni logički zadaci programirani odvojeno a povezani docnije. Modularno programiranje zahteva dekompoziciju programskih zahteva i specifikacija na određeni broj zasebnih rutina, odnosno, jasno definisanih kraćih i dobro dokumentovanih zadataka. Ideja,

jedan zadatak po modulu dobro je pravilo koje programera navodi na generalizaciju rešenja i mogućnosti višestrukog korišćenja koda. Prednosti ovakvog načina programiranja slične su prednostima pri realizaciji modularnog hardverskog sistema. Naime, svaki se programski modul, kao osnovni konstruktivni element programa, može testirati, debagovati i održavati zasebno što programsku celinu čini pouzdanijom. Modul se može preuzeti iz npr. standardne biblioteke koja se isporučuje sa prevodiocem, iz specijalizovnih biblioteka ili iz personalnih biblioteka korisnika koje on sam razvija. Ako je rešenje programskog modula dovoljno opšte, može se primeniti u velikom broju programa. Eventualno ažuriranje programske celine olakšano je izmenom samo pojedinih programskih modula.

Programski moduli se mogu pozivati iz glavnog programa, drugih programskih modula ili na bazi hardverski generisanih zahteva. Tipičan primer startovanja izvršenja potprograma na osnovu hardverskog događaja su potprogrami - rutine za obradu prekida. Softverski modul za obradu prekida ima svoje specifičnosti u odnosu na potprograme zbog čega mu treba biti posvećena posebna pažnja.

U programskom jeziku C potprogrami imaju jedan od dva oblika: funkcije i makroi. Funkcija je potprogram koji podrazumeva mehanizam poziv/povratak (call/return) podržan u modernim računarima orijentisanim na rad sa stekom. Funkcije koriste stek da od pozivaoca dobiju ulazne podatke i predaju im vrednost. Programski kod funkcije se u izvršnom programu pojavljuje samo jednom ali se funkcije mogu pozivati proizvoljan broj puta. Makroi su slični funkcijama, to su imena kojima je pridružena tekstualna definicija. U C jeziku makroi imaju jedan od dva oblika, makro kao objekat čije ime se zamenjuje tekstualnom definicijom i makro kao funkcija koji osim imena sadrži i parametre, slično funkciji. Glavna razlika makroa u odnosu na funkciju je to što se njihov programski kod kopira na mestima poziva u programu i izvršava kao niz instrukcija bez podrške mehanizma poziv/povratak koji koriste funkcije.

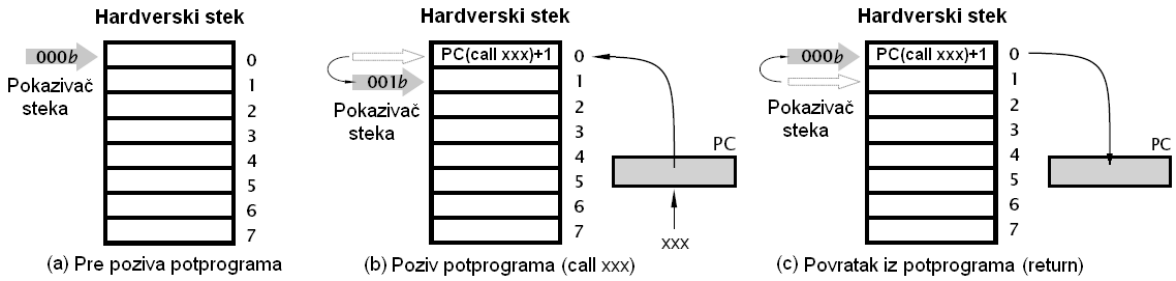
Svako pozivanje funkcije predstavlja dodatno opterećenje jer se deo vremena troši na smeštanje podataka na stek, prenošenje izvršavanja programa na drugo mesto u memoriji i povratak na instrukciju iza poziva funkcije. Makro, sa druge strane, prevodilac razvija u tekst svaki put kada sretne njegovo ime što proizvodi kod koji se izvršava bez dodatnog opterećenja. Međutim, ako se makro poziva više puta u programu, kopiranje koda makroa na više mesta uvećava obim programa i u izvornom i u izvršnom obliku, što je cena veće brzine izvođenja u odnosu na funkciju.



Slika 2.2.4. *Pozivanje potprograma.*

Na slici 2.2.4. je prikazan simbolički tok programa koji u jednom svom delu dva puta poziva potprogram za izvođenje specifičnog zadatka. Na mestu poziva potprograma (*call*), koji se uobičajeno nalazi na drugom mestu u memoriji, kao na slici, tekuća adresa programskog brojača se automatski inkrementira i smešta u hardverski stek tako da ukazuje na povratnu adresu u programu a potom se u programski brojač upisuje adresa prve instrukcije potprograma (simboličku adresu potprograma u izvornom kodu predstavlja njegovo ime). Po povratku iz potprograma (*return*) automatskim mehanizmom se sa vrha hardverskog steka uzima sačuvana povratna adresa koja ukazuje na prvu instrukciju u programu iza poziva potprograma i smešta u

programski brojač. Na taj način omogućen je normalan nastavak izvođenja programa, iz koga je pozvan potprogramski modul.



Slika 2.2.5. *Korišćenje hardverskog steka za čuvanje povratnih adresa.*

Na slici 2.2.5. prikazan je sadržaj steka pre, po pozivu i po povratku iz potprograma. Mikrokontroler PIC16F877 poseduje hardverski stek od osam 13-bitnih registara organizovanih po cirkularnom LIFO (*Last In First Out*) principu. Stek nije u sastavu RAM memorije već predstavlja zasebnu celinu sa pridruženim trobitnim pokazivačem steka koji nije dostupan programeru.

Poziv potprograma (call xxx) inicira redom:

- smeštanje inkrementirane adrese instrukcije poziva (call xxx), koju sadrži 13-bitni programski brojač u registar steka na koji ukazuje pokazivač steka. Inkrementirana adresa instrukcije poziva potprograma (PC(call xxx)+1) predstavlja adresu prve instrukcije koja sledi posle call xxx instrukcije u programu iz koga je pozvan potprogram.
- automatsko inkrementiranje pokazivača steka,
- punjenje programskog brojača adresom prve instrukcije potprograma (xxx) što za posledicu ima grananje programa.

Povratak iz potprograma (return) inicira redom:

- automatsko dekrementiranje pokazivača steka,
- punjenje programskog brojača povratnom adresom koja se čuva u registru steka na koji ukazuje pokazivač steka što za posledicu ima nastavak izvođenja programa iz koga je pozvan potprogram.

Izlazna tačka svakog potprograma mora biti instrukcija return.

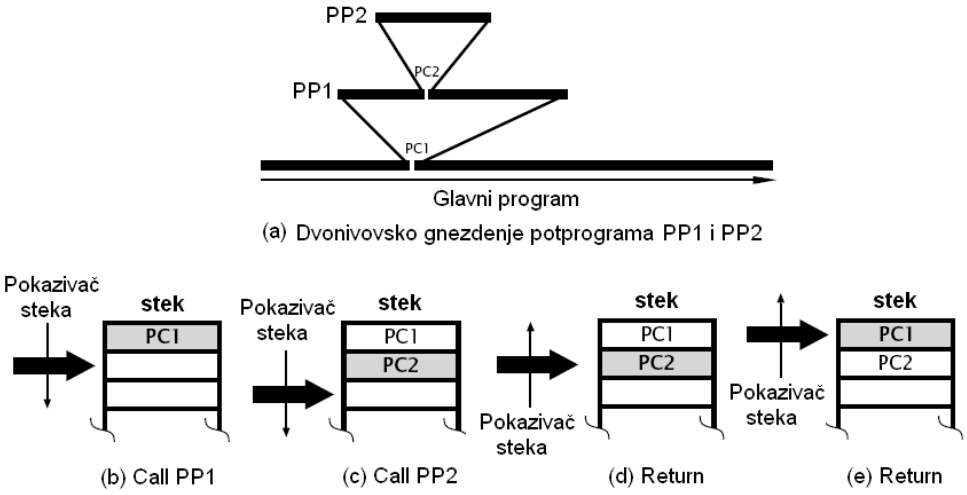
Mehanizam steka podržava tkzv. gneždenje potprograma, odnosno, mogućnost pozivanja jednog potprograma iz drugog potprograma. Na slici 2.2.6. prikazan je simbolički princip dvonivovskog gneždenja, kao i izgled hardverskog steka pri pozivanju i povratku iz potprograma. Iz glavnog programa se najpre poziva potprogram PP1 čija se povratna adresa PC1 smešta na vrh steka na koji ukazuje pokazivač. Potom se iz potprograma PP1 poziva potprogram PP2 čija se povratna adresa PC2 smešta na jedan nivo ispod vrha, imajući u vidu automatsko inkrementiranje pokazivača steka posle smeštanja prve povratne adrese.

Povratak u glavni program događa se suprotnim tokom. Po kompletiranju potprograma PP2 povratnom instrukcijom return se dekrementira pokazivač steka a potom u programski brojač prenosi sačuvana druga povratna adresa PC2. Ova adresa je zapravo adresa prve instrukcije posle instrukcije poziva potprograma PP2 i nalazi se u potprogramu PP1 odakle je pozvan potprogram PP2. Kada se završi izvođenje i potprograma PP1 povratnom instrukcijom RETURN se još jednom dekrementira pokazivač steka a potom u programski brojač prenosi sačuvana prva povratna adresa PC1. Ova adresa je zapravo adresa prve instrukcije posle instrukcije poziva potprograma PP1 i nalazi se u glavnom programu odakle je pozvan potprogram PP1.

Mehanizam steka može rukovati i situacijama u kojima potprogram poziva samog sebe. Takvi potprogrami poznati su pod nazivom rekurzivni potprogrami.

Mehanizam steka se dalje koristi i za rukovanje prekidima. U programima koji koriste i potprograme i prekide, dubina steka od osam nivoa katkad može biti nedovoljna. Naime,

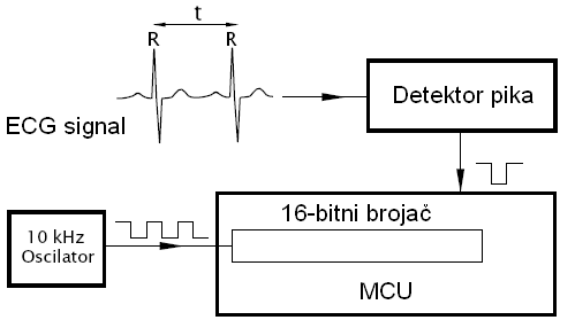
cirkularni LIFO mehanizam steka posle sačuvanih osam povratnih adresa prepisuje devetu adresu preko prve, desetu preko druge itd., čime se gube povratne adrese nekih potprograma ili prekida. Mikrokontroler PIC16F877 ne poseduje statusne bitove za indikaciju praznog i punog steka niti instrukcije PUSH (guraj u stek) i POP (izvuci iz steka) za implementaciju softverskog steka. PUSH i POP akcije se odnose na sadržaj programskog brojača i automatski se događaju izvođenjem instrukcija CALL, RETURN, RETLW i RETFIE kao i grananjem programa na adresu prekidnog vektora.



Slika 2.2.6. *Gnežđenje potprograma i izgled steka pri pozivanju i povratku iz potprograma.*

2.2.5. Rukovanje prekidima

Potprogrami, razmatrani u prethodnoj sekciji, predstavljaju jednostavne predvidljive događaje koje diktira program. U realnim situacijama, gde postoji interakcija procesora sa spoljašnjim fizičkim događajima na čije zahteve procesor treba da odgovori, to, međutim, nije jednostavno. Primer jednog vremenski osetljivog događaja prikazan je na slici 2.2.7.



Slika 2.2.7. *Detekcija i merenje trajanja jednog spoljašnjeg događaja.*

Sistem meri proteklo vreme između dva R talasa signala elektrokardiografa (ECG) pacijenta a zasnovan je na detektoru pika ECG signala i mikrokontroleru MCU. Maksimalno očekivano vreme između dva uzastopna R talasa je 1.5s. Potrebno vremensko razlaganje od 0.1ms

dobijeno je primenom spoljašnjeg oscilatora frekvencije 10KHz. Za merenje proteklog vremena, kao vremenska baza se koristi 16-bitni brojač taktovan impulsima oscilatora. Merenje vremena se zasniva na stalnom praćenju stanja brojača i čitanju njegovog registra u trenucima pojave R talasa. Razlika dva uzastopno pročitana stanja brojača srazmerna je proteklom vremenu. Glavni problem, međutim, jeste detekcija R talasa budući da ECG signal pacijenta nije sinhronizovan sa MCU. Jedan od načina detekcije je stalno čitanje izlazne linije detektora pika povezane na ulaz mikrokontrolera. Takva tehnika poznata je pod nazivom tehnika prozivke (*polling*). Ako se pretpostavi normalni srčani ritam pacijenta od 60 otkucaja/min, tada vreme između dva uzastopna R talasa iznosi 1s. Za detekciju jednog R talasa koji se može dogoditi u ovom vremenu potrebno je, za usvojeno razlaganje vremena, obaviti 10000 prozivki. Prema tome, 99.99% vremena, u kome nema pojave R talasa, procesor troši beskorisno tražeći jedan događaj u 10000.

Alternativni pristup rešenju bio bi korišćenje dodatnog hardvera čiji zadatak bi bio slanje signala procesoru u trenutku detekcije R talasa, što bi moglo biti iskorišćeno za prekidanje tekućeg procesa, čitanje registra brojača i potom nastavak izvođenja prekinutog procesa. Takav pristup daje šansu procesoru da do trenutka detekcije R talasa obavi mnoštvo drugih korisnih akcija, izvršavajući tekući proces, i ne potroši ovo vreme na prozivku. Ovakav mehanizam-mehanizam prekida podržavaju svi savremeni procesori, uključujući i hardver za generisanje prekidnog signala na odgovarajući događaj. Glavni nedostatak prekidom upravljanog praćenja događaja u realnom vremenu je dodatna kompleksnost hardvera.

Tehnika prozivke može biti adekvatna za slučaj događaja koji se retko pojavljuju, kada perioda prozivanja ne mora biti kratka. Međutim, smanjenje brzine prozivanja smanjuje verovatnoću precizne detekcije trenutka pojave asinhronog događaja (preskakanje) ali povećava efikasnost procesora.

Esencijalno, pojava prekidnog signala uzrokuje prekid aktuelnog procesa, pamćenje njegove prekinute pozicije i skok na specijalni potprogram poznat pod imenom prekidna servisna rutina. Ova se rutina izvršava samo na zahtev, odnosno, na odgovarajući događaj koji generiše prekidni signal.

Na slici 2.2.8. je prikazan blok dijagram odziva procesora na prekidni zahtev. Kao što se može videti:

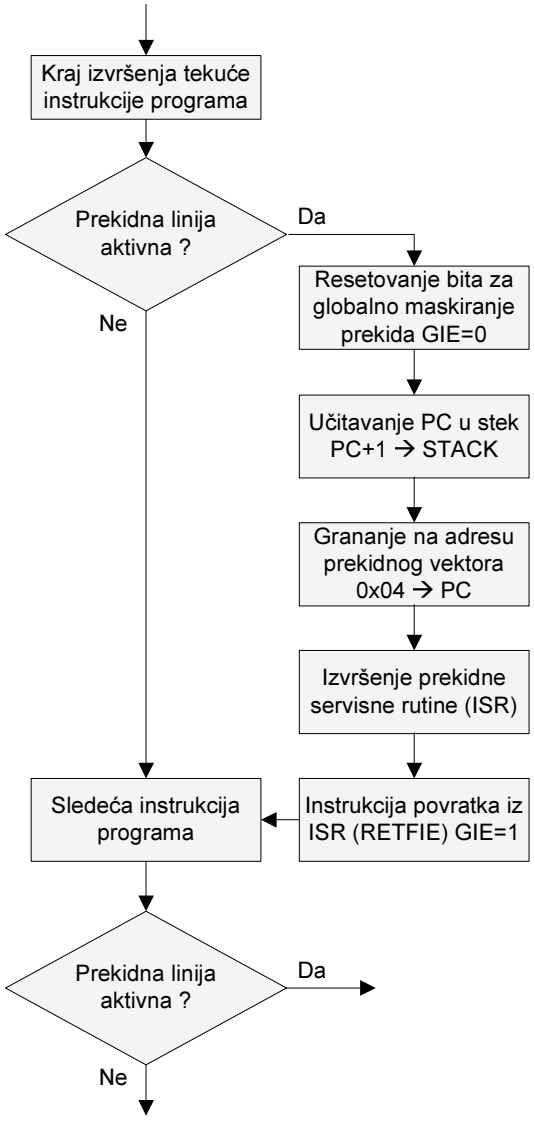
1. Procesor odmerava prekidnu liniju jednom u svakom instrukcijskom ciklusu. Ako je ova linija aktivna, odgovarajući indikator prekida (flag-zastavica), koji predstavlja leč kolo, je setovan, u suprotnom indikator je resetovan. Bez obzira na stanje ove linije tekuća instrukcija se izvršava do kraja. To znači da se instrukcija u vreme izvođenja ne može prekinuti, čak i u slučaju dvociklusnih instrukcija.
2. Ako je prekidna linija neaktivna, procesor jednostavno nastavlja sa izvođenjem sledeće instrukcije posle čega se opisani proces ispitivanja linije prekida ponavlja.
3. Ako je prekidna linija aktivna, u sledeća tri instrukcijska ciklusa kontrola se prebacuje na prekidnu servisnu rutinu. Prvi od tri ciklusa može biti završni ciklus dvociklusne instrukcije ili, u suprotnom, pseudo ciklus, dok su druga dva neophodna za uspostavljanje protočne obrade instrukcija na kojoj bazira mikrokontroler PIC16F877. Opisano kašnjenje od tri do četiri instrukcijska ciklusa, od trenutka pojave prekidnog signala do početka izvođenja prve instrukcije prekidne servisne rutine, poznato je pod nazivom vreme kašnjenja (*latency*). Nemoguće je preciznije odrediti vreme kašnjenja (tri ili četiri instrukcijska ciklusa) zbog slučajne prirode pojave prekidnog zahteva koji se može pojaviti bilo kada, npr odmah posle ispitivanja prekidne linije.
4. U toku trajanja vremena kašnjenja po automatizmu se izvršavaju sledeće operacije:

a) Bit za globalno maskiranje prekida GIE (*Global Interrupt Enable*) se resetuje čime se automatski zabranjuju (maskiraju) svi ostali prekidi. Na ovaj način omogućena je obrada aktuelnog prekida bez mogućnosti njegovog prekidanja.
b) Sadržaj inkrementiranog registra programskog brojača upućuje se na hardverski stek na isti način kao i za instrukciju poziva potprograma (*CALL*). Ovo će omogućiti procesoru da se po izvođenju prekidne servisne rutine instrukcijom povratka vrati na mesto prekida. Pri ovome treba

imati u vidu ograničenu dubinu hardverskog steka od osam nivoa zbog mogućeg ugnježdavanja.

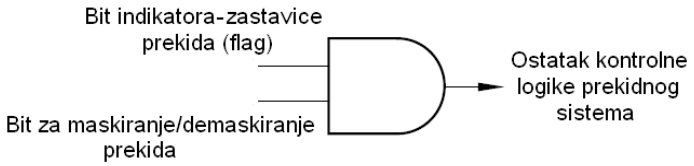
c) Kako je prva instrukcija prekidne servisne rutine uvek na adresi prekidnog vektora 004h programske memorije, završni korak je punjenje registra programskog brojača adresom ove instrukcije. Ako je program za obradu prekida na drugom mestu u programskoj memoriji a ne odmah ispod adrese prekidnog vektora tada se instrukcijom bezuslovnog skoka *GOTO* vrši grananje programa na početak programa za obradu prekida.

5. Kao i potprogram, prekidna servisna rutina mora biti okončana instrukcijom za povratak. Međutim, u ovom slučaju nije dovoljno samo punjenje programskog brojača povratnom adresom sa steka. Neophodno je još i ponovo omogućiti mehanizam prekida, onemogućen resetovanjem GIE bita za maskiranje svih prekida. Relevantna završna instrukcija za ovu situaciju je RETFIE (*RETurn From Interrapt and Enable*). Na taj način, po povratku u prekinuti program bilo koji budući prekidi mogu biti servisirani.



Slika 2.2.8. *Odziv na prekidni zahtev.*

Prekidna servisna rutina se značajno razlikuje od potprograma ne samo po završnoj instrukciji (*RETFIE*) već i po logici prekidnog sistema, kao i pseudoslučajnoj prirodi prekida. Na slici 2.2.9. je prikazan deo te logike za jedan izvor prekida koji ima interakciju sa procesorom preko dva bita kontrolnog registra prekida.



Slika 2.2.9. *Bit flag-a-zastavice prekida i bit za maskiranje/demaskiranje prekida kao deo logike prekidnog sistema.*

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-x
	GIE	**PEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**
	bit 7							bit 0
bit 7	**GIE:** Bit za maskiranje svih prekida (globalni mask bit). 1 = Omogućeni svi nemaskirani prekidi. 0 = Onemogućeni svi prekidi.							
bit 6	**PEIE**: Bit za maskiranje prekida sa svih periferija. 1 = Omogućeni nemaskirani prekidi sa svih periferija. 0 = Onemogućeni prekidi sa svih periferija.							
bit 5	**T0IE**: Bit za omogućenje prekida sa tajmera 0 pri tranziciji FFh→00h. 1 = Omogućen prekid. 0 = Onemogućen prekid.							
bit 4	**INTE**: Bit za maskiranje spoljašnjeg prekida sa RB0/INT linije. 1 = Omogućen prekid. 0 = Onemogućen prekid.							
bit 3	**RBIE**: Bit za maskiranje prekida sa višeg nibla porta B. 1 = Omogućen prekid. 0 = Onemogućen prekid.							
bit 2	**T0IF**: Zastavica prekida sa tajmera 0 pri tranziciji brojačkog registra TMR0 FFh→00h. 1 = TMR0 brojački registar prekoračio osnovu brojanja (bit mora biti programski resetovan). 0 = TMR0 brojački registar nije prekoračio osnovu brojanja.							
bit 1	**INTF**: Zastavica spoljašnjeg prekida sa RB0/INT linije. 1 = RB0/INT spoljašnji prekid se dogodio (bit mora biti programski resetovan). 0 = RB0/INT spoljašnji prekid se nije dogodio.							
bit 0	**RBIF**: Zastavica prekida sa višeg nibla porta B. 1 = Najmanje jedan od četiri bita višeg nibla porta B je promenio stanje u odnosu na stanje zapamćeno u leč registru porta B. Operacija čitanja porta B može ukloniti ovu asimetriju (bit mora biti programski resetovan). 0 = Nijedan od četiri bita višeg nibla porta B nije promenio stanje u odnosu na stanje leč registra porta B.							

Tabela 2.2.5. *Sadržaj registra INTCON za kontrolu tri standardna izvora prekidnog signala.*

Bit zastavice prekida (flag) se automatski setuje kada odgovarajući izvor prekida zahteva servis. Ako je pri tome i mask bit setovan zahtev će biti prosleđen ostatku kontrolne logike prekidnog sistema, kao na slici 2.2.9. Treba primetiti da stanje mask bita ne utiče na stanje zastavice. To znači da se zastavica automatski setuje svaki put kada izvor prekida postavi zahtev, dok se prosleđivanje zahteva ostatku logike vrši samo u slučaju da je mask bit setovan.

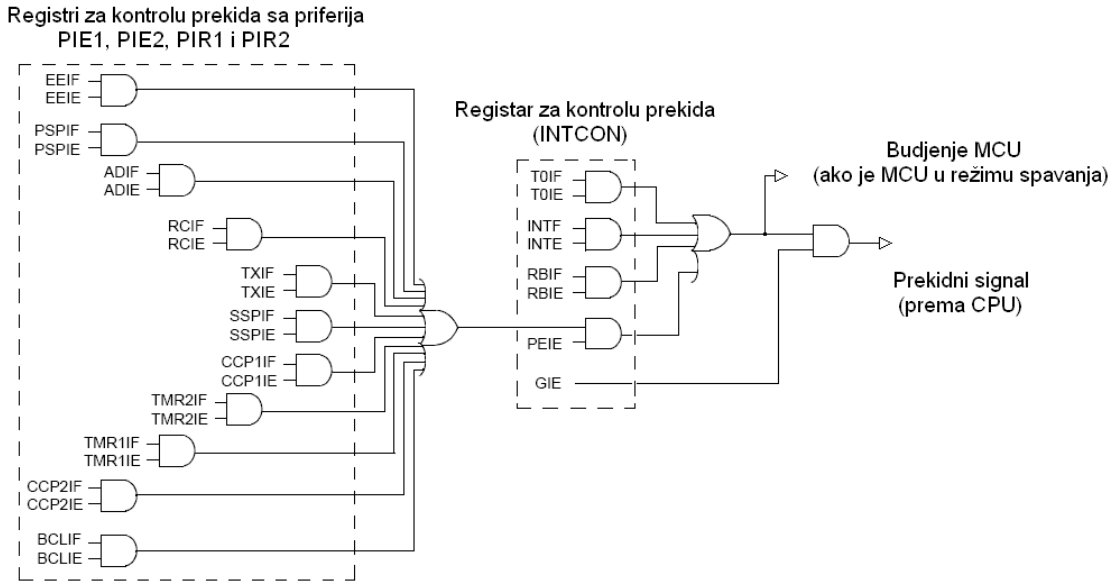
Prema tome, ako je mask bit resetovan, jedino se tehnikom prozivke zastavice prekida može detektovati zahtev koji generiše odgovarajući izvor prekida. Bit zastavice prekida se po automatskom setovanju mora programski resetovati kako bi se mogao detektovati sledeći zahtev izvora prekida, prekidnim mehanizmom ili prozivkom.

Veoma je važno napomenuti da se neposredno pre povratka iz rutine za obradu prekida u prekinuti program bit zastavice prekida mora programski resetovati da bi se mogao detektovati novi zahtev za prekidnim servisom. U suprotnom, dogodila bi se beskrajna serija prekidnih zahteva. Setovana zastavica prekida bila bi znak procesoru da se dogodio novi zahtev pa bi po izlasku iz prekidne servisne rutine program ponovo granao na prekidnu rutinu. Kako neposredno pre izlaska iz prekidne rutine nije resetovana zastavica prekida, opisani ciklus bi se ponavljao neograničeni broj puta.

Individualni (lokalni) bit za maskiranje/demaskiranje prekida, koji se odnosi na odgovarajući izvor prekida, može biti setovan ili resetovan isključivo programski. Posle reseta mikrokontrolera individualni mask bitovi svih izvora prekida, kao i GIE bit, postavljaju se u reset stanje čime su svi individualni prekidi maskirani. Za demaskiranje pojednih prekidnih zahteva treba koristiti asemblersku instrukciju BSF (*bit set file register*), odnosno, setovati mask bit odgovarajućeg izvora prekida u kontrolnom registru prekida ali i bit za globalno omogućenje prekida GIE.

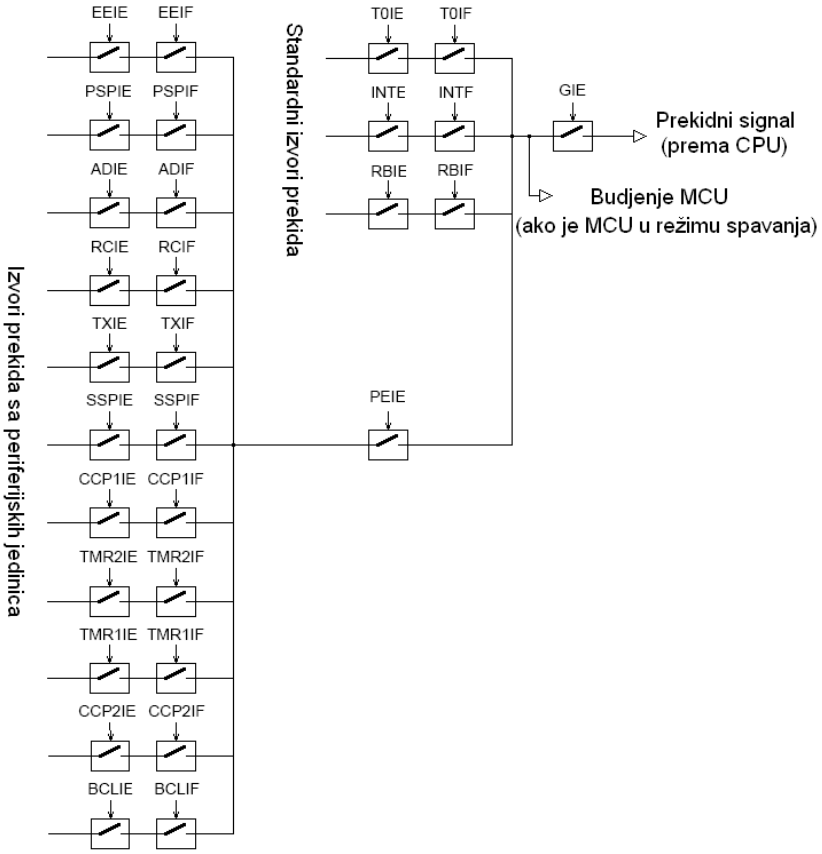
PIC16F87X serija mikrokontrolera poseduje 14 hardverski ugrađenih izvora prekidnog signala, slika 2.2.10. Registar za kontrolu prekida INTCON (*INTerrupt CONtrol*) sadrži individualne bitove za maskiranje/demaskiranje tri standardna prekida (TOIE, INTE, RBIE) i njima odgovarajuće flag bitove, bit za globalno maskiranje/demaskiranje svih vrsta prekida GIE, kao i bit za globalno maskiranje/demaskiranje prekida sa svih periferijskih jedinica PEIE (*PEripheral Interrupt Enable*), tabela 2.2.5.

GIE (G*lobal Interrupt Enable*) bit omogućava (ako je setovan) ili onemogućava (ako je resetovan) sve nemaskirane prekide. Kada je GIE bit setovan zajedno sa individualnim bitom za omogućenje/onemogućenje (maskiranje/demaskiranje) prekida, setovanje odgovarajućeg individualnog flag bita uzrokuje programski skok na adresu prekidnog vektora. Individualni mask bitovi za omogućenje/onemogućenje prekida mogu se setovati bez obzira na stanje GIE bita. Međutim, programski skok na adresu prekidnog vektora se neće dogoditi u slučaju da je GIE bit resetovan. Individualni prekidi se mogu onemogućiti kroz njihove odgovarajuće mask bitove za omogućenje/onemogućenje prekida koji se nalaze u različitim registrima.



Slika 2.2.10. *Kontrolna logika prekidnog sistema mikrokontrolera PIC16F877.*

Tri standardna izvora prekidnog signala, spoljašnji prekid na liniji RB0/INT porta B, spoljašnji prekid na linijama višeg nibla porta B, RB<7:4> i prekid koji generiše tajmer 0 modul usled prekoračenja njegovog osmobitnog brojačkog registra TMR0, kontrolišu se i detektuju u registru INTCON i hardverski su ugrađeni u gotovo svim serijama Microchip-ovih mikrokontrolera.



Slika 2.2.11. *Principijelna ekvivalentna šema prekidne logike mikrokontrolera PIC16F877.*

Kontrolni bitovi prekida sa pojedinih periferija (omogućenje/onemogućenje) sadržani su u registrima specijalne namene PIE1 i PIE2, a njima odgovarajuće zastavice prekida (flag bitovi) u SFR registrima PIR1 i PIR2. Bit za globalno maskiranje/demaskiranje prekida sa svih periferija (PEIE) sadržan je u INTCON registru.

Principijelna ekvivalentna šema prekidne logike mikrokontrolera, slika 2.2.11., jasno pokazuje da je GIE bit glavni prekidač svih prekida, dok je PEIE bit glavni prekidač prekida sa periferijskog podsistema. Takođe se može primetiti da parovi mask bit/bit zastavice individualnih prekida moraju biti setovani da bi se prekid dogodio.

Mikrokontroler PIC16F877 se instrukcijom SLEEP uvodi u stand-by režim niske potrošnje. Izvođenjem ove instrukcije blokira se kvarcni oscilator, kojim se taktuje CPU jedinica, čime se prekida sa daljim izvođenjem instrukcija. Kako je snaga disipacije MCU srazmerna frekvenciji taktovanja

$$P_D = C_T \, f_{CLK} \, V_{DD}^2$$

gde je f_{CLK} frekvencija taktnih impulsa, C_T ekvivalentno koncentrisano kapacitivno opterećenje MCU a V_{DD} napon napajanja MCU, to se za f_{CLK}*=0Hz* snaga disipacije svodi na minimalnu, što

je posledica struja curenja kroz poluprovodnik. Ova osobina mikrokontrolera od esencijalnog je značaja za uređaje sa baterijskim napajanjem. Buđenje mikrokontrolera iz *sleep* režima vrši se resetom MCU ili generisanjem bilo kog prekidnog signala. Sa slika 2.2.10. i 2.2.11. je evidentno da se buđenje MCU prekidnim signalom može obaviti na dva načina: normalnim nastavljanjem izvođenja prve instrukcije posle SLEEP instrukcije ako je pre izvođenja SLEEP instrukcije GIE bit bio resetovan ili grananjem programa na prekidni vektor ako je GIE bit bio setovan.

Budući da postoji samo jedan prekidni vektor na adresi programske memorije 004h i ukupno 14 izvora prekida, prvi zadatak rutine za obradu prekida ISR je određivanje izvora prekidnog signala metodom prozivke individualnih zastavica prekida-flegova. Ovaj deo rutine za obradu prekida poznat je pod nazivom analizator prekida. Po određivanju izvora prekidnog signala prelazi se na obradu odgovarajućeg prekida. Uobičajeno je da se prozivaju samo izvori prekida koji su omogućeni-demaskirani. Deo asemblerskog programa koji pripada prekidnoj servisnoj rutini sa analizatorom prekida za tri standardna izvora prekida u INTCON registru prikazan je sledećom sekvencom instrukcija u tabeli 2.2.6.

```
ISR:                            ; Početak prekidne servisne rutine
BTFSC      INTCON,RBIF          ; Proveri prekid na promenu višeg nibla porta B
GOTO       EXT_PROMENA          ; Ako jeste, skoči na labelu EXT_PROMENA
BTFSC      INTCON,INTF          ; Proveri prekid na RB0 pinu porta B
GOTO       EXT_PIN              ; Ako jeste, skoči na labelu EXT_PIN
BTFSC      INTCON,T0IF          ; Proveri prekid sa Tajmera 0
GOTO       TAJMER0              ; Ako jeste, skoči na labelu TAJMER0

IRQ_IZLAZ:
RETFIE                          ; Povratak iz prekidne rutine sa GIE=1

EXT_PROMENA:
;
; Kod za obradu prekida sa višeg nibla porta B
;
BCF        INTCON,RBIF          ; Brisanje zastavice prekida pre izlaska iz ISR
GOTO       IRQ_IZLAZ

EXT_PIN:
;
; Kod za obradu prekida sa RB0 pina porta B
;
BCF        INTCON,INTF          ; Brisanje zastavice prekida pre izlaska iz ISR
GOTO       IRQ_IZLAZ

TAJMER0:
;
; Kod za obradu prekida sa Tajmera 0
;
BCF        INTCON,T0IF          ; Brisanje zastavice prekida pre izlaska iz ISR
GOTO       IRQ_IZLAZ
```

Tabela 2.2.6. *Asemblerska forma ISR sa analizatorom prekida za tri standardna izvora prekidnog signala.*

Iz date sekvence se vidi da se najpre vrši provera ko je od tri izvora prekida generisao prekidni zahtev i potom prelazi na odgovarajuću obradu. Redosled prozivke (RBIF, INTF, T0IF) definiše prioritet prekida u slučaju da više od jednog prekida koincidira. Jasno je da će u tom slučaju zbog setovanih zastavica prekida, po obradi prioritetnog prekida i izlasku iz prekidne rutine program ponovo ulaziti u nju uzastopno onoliki broj puta koliko preostalih koincidentnih zahteva nižeg prioriteta postoji. Obrada prekidnih zahteva ići će redosledom koji diktira analizator prekida. U gornjem primeru, za slučaj da sva tri zahteva koincidiraju najpre se obrađuje

EXT_PROMENA, po ponovnom ulasku u prekidnu rutinu EXT_PIN i na kraju TAJMER0 podrutina, resetujući pri tome svaki put odgovarajuću zastavicu prekida.

Zbog automatskog hardverskog setovanja zastavica svih izvora prekida nezavisno od stanja njihovih mask bitova, tehnika prozivke zastavica može biti iskorišćena za detekciju odgovarajućih događaja bez primene prekidnog mehanizma. Da bi se mogao detektovati sledeći događaj, kao i u slučaju ISR, mora se izvršiti programsko resetovanje odgovarajuće zastavice. Primena tehnike prozivke umesto prekidnog mehanizma, kao što je već rečeno, značajno umanjuje efikasnost procesora trošeći dragoceno procesorsko vreme na prozivanje, odnosno, čekanje na pojavu događaja.

Interni procesorski registri (registri posebne namene) kao što su tipično W-akumulator, STATUS-statusni registar ALU, PCLATH-upisni bafer za pet viših bitova programskog brojača i PC-programski brojač jesu deljivi resursi. Jedna ISR rutina koristi ove registre na isti način kao i bilo koji potprogram ili deo glavnog programa, tako da je eventualni konflikt zbog izmene sadržaja registara moguć. Na primer, zbog slučajne prirode prekidnog zahteva ne može se predvideti memorijska banka memorije podataka sa čijim registrima operiše program u trenutku pojave prekida. Budući da se program prekida a kontrola prebacuje na rutinu za obradu prekida, u njoj se zavisno od potrebe može selektovati druga memorijska banka. Selekcija odgovarajuće banke vrši se odgovarajućim upisom u bitske pozicije 5 i 6 statusnog registra. Po izlasku iz prekidne rutine lako može doći do konflikta ako aktuelna banka ne odgovara onoj u kojoj je program operisao do pojave prekida. Sličan zaključak važi i za npr. registar akumulatora W čijim sadržajem operiše i tekući program koji se prekida i prekidna rutina.

Mogući konflikti se mogu izbeći pamćenjem sadržaja procesorskih registara od značaja u trenutku pojave prekida, slično hardverskom mehanizmu čuvanja sadržaja PC registra u steku. Posle obrade prekida i neposredno pred povratak u prekinuti program, obnavljanjem sadržaja ovih registara na stanje koje odgovara trenutku pojave prekidnog zahteva moglo bi se nastaviti sa izvođenjem prekinutog programa kao da prekida nije ni bilo.

Opisani postupak pamćenja sadržaja internih registara, prebacivanja kontrole na prekidnu servisnu rutinu, obnavljanja sadržaja registara i ponovnog vraćanja kontrole na prekinuti program poznat je pod nazivom zamena konteksta procesa. Posebna pažnja pri zameni konteksta treba biti posvećena izboru lokacija u memoriji podataka gde će se čuvati zapamćeni sadržaji internih registara. Ove lokacije koriste se za sistemske namene i ne sme ih koristiti niti prekidna rutina za čuvanje rezultata/međurezultata obrade, niti potprogrami ili glavni program.

16 registara opšte namene (GPR-korisnički registri), koji pripadaju najvišem adresnom prostoru bilo koje od četiri memorijske banke mikrokontrolera PIC16F877, mapirani su u svim bankama memorije podataka te predstavljaju njihove zajedničke registre. Otuda je najpogodnije ove memorijske lokacije RAM memorije koristiti za pamćenje i obnavljanje sadržaja internih registara procesora jer ne zahtevaju selekciju memorijske banke ('*banking*'). Izabrane lokacije predstavljaju softverski stek i koriste se u nedostatku instrukcija PUSH i POP, kojima raspolažu neki mikrokontroleri, za pamćenje i obnavljanje sadržaja važnih registara, respektivno. U slučaju nedostatka, ove instrukcije se mogu programski implementirati kao npr. asemblerski makroi. Takav način implementacije PUSH i POP instrukcija povećava fleksibilnost u smislu pamćenja i obnavljanja sadržaja većeg broja registara od značaja ali, nažalost, iziskuje više vremena za izvršavanje.

Primeri asemblerske programske implementacije PUSH i POP instrukcija u formi makroa dati su u tabeli 2.2.7.

```
;Rezervacija prostora (dodela adresa promenljivim)

S_COPY     EQU  0x70          ;16 GPR registara, najvišeg adresnog prostora
P_COPY     EQU  0x71          ;svake banke zajednički su za sve četiri banke
W_COPY     EQU  0x72          ;počev od adrese 0x70 do 0x7F

PUSH MACRO
```

```
    MOVWF       W_COPY          ;Sačuvaj sadržaj W reg. u njegovu kopiju W_COPY.
    SWAPF       STATUS, W       ;Sačuvaj sadržaj STATUS reg. bez uticaja na njegov
    MOVWF       S_COPY          ;sadržaj, primenom instrukcije swapf.
    MOVF        PCLATH, W       ;Sačuvaj sadržaj PCLATH reg. u njegovu kopiju P_COPY
    MOVWF       P_COPY
    ENDM

POP  MACRO
    MOVF        P_COPY, W       ;Obnovi PCLATH reg. i selektuj originalnu stranicu
    MOVWF       PCLATH          ;programske memorije.
    SWAPF       S_COPY, W       ;Obnovi STATUS reg. i selektuj originalnu banku
    MOVWF       STATUS          ;memorije podataka.
    SWAPF       W_COPY, F       ;Obnovi W reg. ne utičući na sadržaj već obnovljenog
    SWAPF       W_COPY, W       ;STATUS reg., dvostrukom primenom instrukcije swapf.
    ENDM
```

Tabela 2.2.7. *Programska implementacija PUSH i POP makroa (potprograma).*

Opšti oblik deklaracije makro definicije u asembleru je `<label> macro [<arg>,...,<arg>]`. Kraj makro definicije završava direktivom ENDM. Kao što je opisano u poglavlju 2.2.4. ove knjige makro je sličan funkciji, može i ne mora imati argumente a na mestu poziva u programu kod makroa se u celini kopira. U gornjem primeru pokazana su dva makroa bez argumenata koja se pozivaju imenima PUSH i POP. Makro PUSH koristi se kao zamena za nedostajuću instrukciju MCU PIC16F877 istog imena. Koristi se za privremeno smeštanje (pamćenje) sadržaja internih registara procesora W, STATUS i PCLATH u odgovarajući adresni prostor memorije podataka sa simboličkim adresama W_COPY, S_COPY i P_COPY.

Može se primetiti da se u oba makroa na određenim mestima koriste instrukcije zamene niblova *swapf* statusnog registra i registra akumulatora. Razlog primene ove instrukcije leži u činjenici da ona ne utiče ni na jedan bit statusnog registra. Razlog za izbegavanje primene naizgled logične instrukcije *movf,* umesto *swapf,* jeste mogućnost uticaja ove instrukcije na Z bit (zero bit) statusnog registra.

Po pravilu, PUSH makro uključuje registre specijalne namene koje kao deljive resurse zajednički koriste ISR, potprogrami ili glavni program i poziva se odmah na ulazu u prekidnu rutinu, odnosno, ispod adrese prekidnog vektora, kao bi se odmah pristupilo 'spašavanju' okoline prekinutog procesa. Nasuprot tome, POP makro se po pravilu poziva na izlazu iz prekidne rutine, odnosno, ispred poslednje instrukcije RETFIE, kako bi se po obradi prekida izvršila rekonstrukcija okoline prekinutog procesa.

Sledeća programska sekvenca prikazuje način implementacije PUSH i POP makroa u prekidnoj rutini, tabela 2.2.8.

```
ORG 0x04                    ; Adresa prekidnog vektora
PUSH                        ; Poziv PUSH makroa navođenjem njegovog imena
GOTO        ISR             ; Skok na ISR za obradu prekida
;
;
ISR:                        ; Početak prekidne servisne rutine
BTFSC       INTCON,RBIF     ; Proveri prekid na promenu višeg nibla porta B
GOTO        EXT_PROMENA     ; Ako jeste, skoči na labelu EXT_PROMENA
BTFSC       INTCON,INTF     ; Proveri prekid na RB0 pinu porta B
GOTO        EXT_PIN         ; Ako jeste, skoči na labelu EXT_PIN
BTFSC       INTCON,T0IF     ; Proveri prekid sa Tajmera 0
GOTO        TAJMER0         ; Ako jeste, skoči na labelu TAJMER0

IRQ_IZLAZ:
POP                 ; Poziv POP makroa navođenjem njegovog imena
RETFIE                      ; Povratak iz prekidne rutine sa GIE=1
```

```
EXT_PROMENA:
;
; Kod za obradu prekida sa višeg nibla porta B
;
BCF INTCON,RBIF      ; Brisanje zastavice prekida pre izlaska iz ISR
GOTO       IRQ_IZLAZ

EXT_PIN:
;
; Kod za obradu prekida sa RB0 pina porta B
;
BCF INTCON,INTF      ; Brisanje zastavice prekida pre izlaska iz ISR
GOTO       IRQ_IZLAZ

TAJMER0:
;
; Kod za obradu prekida sa Tajmera 0
;
BCF INTCON,T0IF      ; Brisanje zastavice prekida pre izlaska iz ISR
GOTO       IRQ_IZLAZ
```

Tabela 2.2.8. *Način pozivanja PUSH i POP makroa unutar ISR.*

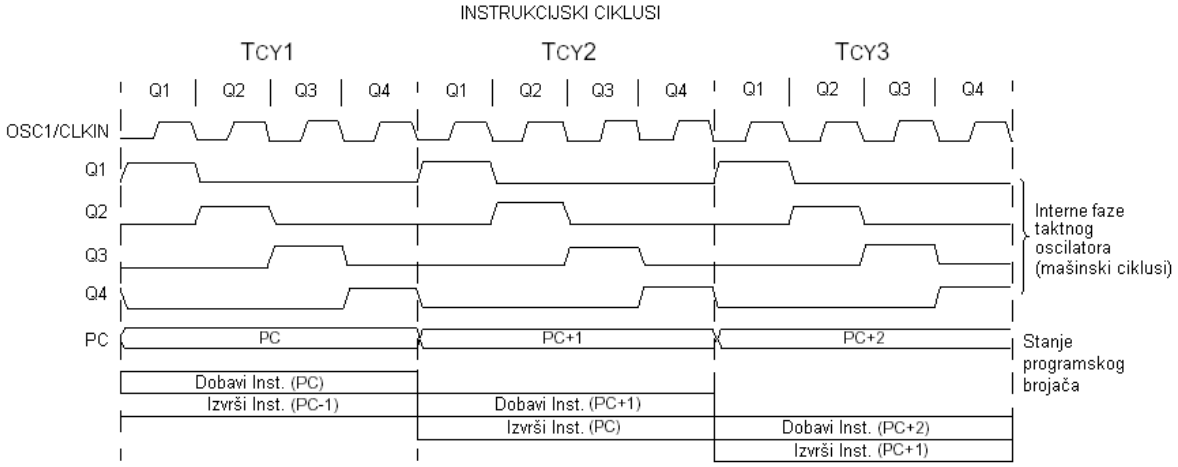
Iz svega napred rečenog može se zaključiti:

- ISR se mora završiti instrukcijom RETFIE,
- sadržaji deljivih registara specijalne namene moraju biti sačuvani na ulazu i obnovljeni na izlazu iz ISR,
- nema prosleđivanja parametara ka ili od ISR. Umesto njih treba koristiti globalne promenljive, ako je potrebno.
- ISR treba biti što je moguće kraća sa minimalnom funkcionalnošću. Ovaj princip obezbeđuje garanciju da drugi prekidni zahtevi ne budu propušteni a od koristi je i pri debagovanju rutine.

2.2.6. Protočna (pipelining) obrada instrukcija

Kao rezultat Harward arhitekture mikrokontrolera PIC16F877, funkcije dobavljanja i izvršenja instrukcija su razdvojene i obavljaju se paralelno sa minimumom interakcije. Konkurentan pristup programskoj memoriji i memoriji podataka dopuštaju mogućnost jednovremenog dobavljanja jedne i izvršenja druge instrukcije što za posledicu ima ubrzanje izvršenja programskog koda.

Na slici 2.2.12. ilustrativno je prikazan princip preklapanja faza dobavljanja i izvršenja instrukcija koji koristi PIC16F877. Instrukcijskim ciklusom T_{CY} se naziva vreme potrebno za izvršenje jedne instrukcije. Kao što se sa slike može primetiti, jedan instrukcijski ciklus čine četiri mašinska ciklusa (Q), odnosno, četiri periode taktnog oscilatora CPU jedinice T_{OSC}. Prema tome, za poznatu i konstantnu frekvenciju oscilovanja generatora takta CPU $f_{OSC}=1/T_{OSC}$, instrukcijski ciklus je takođe konstantan i iznosi $T_{CY}=4T_{OSC}=4/f_{OSC}$. Očigledno je, dakle, da je trajanje instrukcijskog ciklusa četiri puta veće u odnosu na periodu taktnog oscilatora. S tim u vezi, brzina izvršenja instrukcija je četiri puta manja u odnosu na frekvenciju taktnog oscilatora. Budući da je brzina izvršenja instrukcija važan parametar za korisnika, proizvođači MCU definišu njenu maksimalnu vrednost simboličkom oznakom MIPS (*Million Instructions per Second*). Maksimalna frekvencija taktovanja mikrokontrolera PIC16F877, prema specifikaciji proizvođača, iznosi 20MHz. U skladu sa time i maksimalna brzina izvršenja instrukcija iznosi $f_{OSC}/4=5$ MIPS.



Slika 2.2.12. *Preklapanje faza dobavljanja tekuće i izvršenja prethodno dobavljene instrukcije.*

Za realizaciju tehnike protočne obrade instrukcija, instrukcijski registar je dvostruko baferovan (implementiran u dva nivoa-IR1 i IR2) tako da se dobavljanje jedne instrukcije izvodi u jednom instrukcijskom ciklusu. Dobavljanje instrukcije započinje u prvom mašinskom ciklusu Q_1 tekućeg instrukcijskog ciklusa, inkrementiranjem programskog brojača (PC+1) a završava u mašinskom ciklusu Q_4 čitanjem instrukcije iz programske memorije i smeštanjem njenog operacionog koda u instrukcijski registar IR1. U isto vreme, prethodno dobavljena instrukcija, za vrednost programskog brojača (PC), premešta se iz instrukcijskog registra IR1 u registar IR2, koji zapravo predstavlja registar instrukcijskog dekodera.

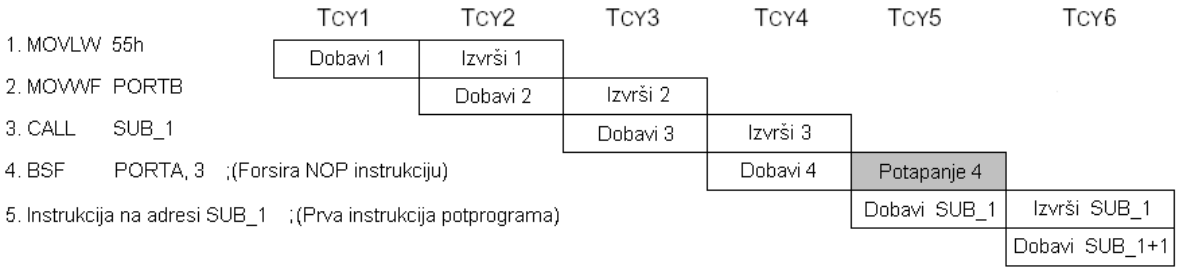
Izvršenje instrukcije smeštene u instrukcijski dekoder započinje u mašinskom ciklusu Q_1 dekodovanjem operacionog koda instrukcije i potom izvršenjem u toku trajanja sledeća tri mašinska ciklusa Q_2, Q_3 i Q_4. Čitanje i dobavljanje operanda iz memorije podataka vrši se u ciklusu Q_2 dok se u ciklusu Q_3 vrši obrada u aritmetičko-logičkoj jedinici ALU. U poslednjem ciklusu Q_4 rezultat obrade se upisuje u memoriju podataka.

Kako su adresne i magistrale podataka programske memorije i memorije podataka fizički odvojene, iz napred izloženog je jasno da se i faza dobavljanja i faza izvršenja instrukcija mogu odvijati sinhrono i jednovremeno jer se dobavljanje vrši iz programske memorije a za izvršavanje koristi memorija podataka. Minimum interakcije obe faze je mašinski ciklus Q_4 u kome se vrši smeštanje pročitanog operacionog koda instrukcije u registar IR1 i istovremeni prenos prethodno dobavljene instrukcije u registar IR2, odnosno, registar instrukcijskog dekodera. Takođe je jasno da se pod takvim okolnostima ne može u isto vreme dobavljati i izvršavati ista instrukcija.

Opisanom protočnom obradom instrukcija postiže se efektivno izvršenje instrukcija u jednom instrukcijskom ciklusu, premda i faza dobavljanja i faza izvršenja troše po jedan. Izuzetak čine instrukcije grananja programa, zbog diskontinualne promene sadržaja programskog brojača. Za kompletiranje izvršenja takvih instrukcija potrebna su dva instrukcijska ciklusa, kao na slici 2.2.13.

Slika 2.2.13. prikazuje protočnu obradu instrukcija programske sekvence date na istoj slici. Dobavljanje instrukcije 1 vrši se u instrukcijskom ciklusu T_{CY1} dok se u narednom ona izvršava a u isto vreme dobavlja operacioni kod instrukcije 2. U trećem ciklusu izvršava se instrukcija 2 a dobavlja instrukcija poziva potprograma SUB_1 kao treća instrukcija. U četvrtom instrukcijskom ciklusu sledi izvršenje instrukcije poziva potprograma na simboličkoj adresi SUB_1. U opštem slučaju potprogram SUB_1 može biti lociran bilo gde u programskoj memoriji, što podrazumeva grananje programa, odnosno, programski skok. U skladu sa tehnikom protočne obrade instrukcija u instrukcijskom ciklusu T_{CY4} dobavlja se i sledeća instrukcija u nizu, instrukcija 4. Ova instrukcija, međutim, ne predstavlja prvu instrukciju pozvanog potprograma koju treba

dobaviti ali sadržaj programskog brojača koji ukazuje na ovu instrukciju predstavlja povratnu adresu. Ova adresa mora biti sačuvana na steku, kako bi tekući program mogao normalno nastaviti sa izvršenjem po povratku iz potprograma. Kada je povratna adresa sačuvana, programski brojač PC se puni adresom prve instrukcije potprograma što predstavlja diskontinualnu promenu sadržaja PC. Budući da dobavljena instrukcija 4 nije prva instrukcija potprograma, u instrukcijskom ciklusu T_{CY5} se neće izvršiti. Umesto nje izvršava se NOP (*No Operation*) instrukcija koja ne proizvodi nikakav efekat izuzev potrošnje jednog instrukcijskog ciklusa. U istom ciklusu T_{CY5}, međutim, dobavlja se prva instrukcija potprograma na simboličkoj adresi SUB_1, na koju ukazuje sadržaj programskog brojača. Dobavljena prva instrukcija potprograma izvršava se u instrukcijskom ciklusu T_{CY6} u kome se, takođe, dobavlja i sledeća, druga instrukcija potprograma SUB_1 itd.



Slika 2.2.13. *Primer protočne obrade jednociklusnih i dvociklusnih instrukcija.*

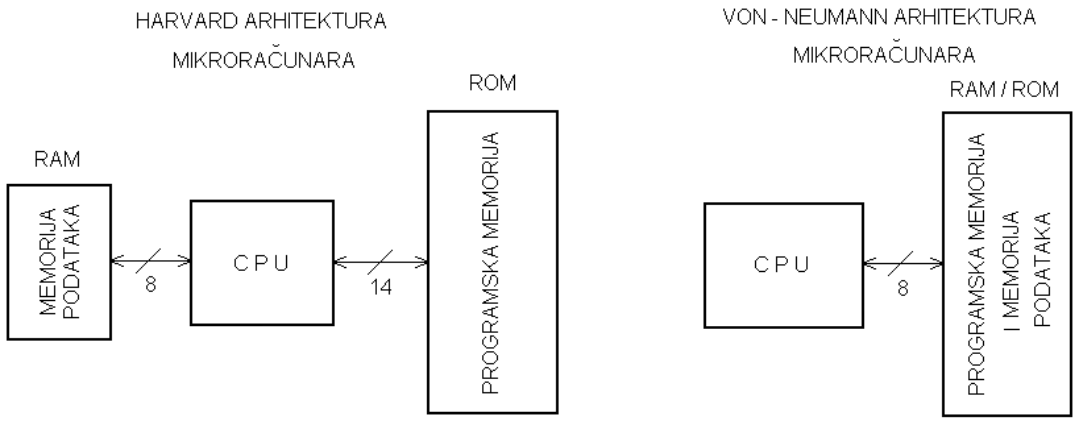
Instrukcija grananja programa, kao što je poziv potprograma CALL, očigledno za izvođenje zahteva dva instrukcijska ciklusa s obzirom na diskontinualnu promenu sadržaja programskog brojača. Slično, i ostale instrukcije grananja kao što su GOTO, RETURN, RETFIE i RETLW takođe troše dvostruko više vremena za izvođenje u odnosu na jednociklusne instrukcije.

Poglavlje 3

ARHITEKTURA I ORGANIZACIJA MCU PIC16F877

3.1. RISC i CISC ARHITEKTURE MIKROKONTROLERA

Dve vrste arhitektura konvencionalne su za današnje mikrokontrolere: Von-Neumann arhitektura na kojoj bazira veći procenat mikrokontrolera i Harvard arhitektura na kojoj je zasnovana PIC serija mikrokontrolera. Na slici 3.1.1. prikazani su principi organizacije pomenutih arhitektura mikrokontrolera.



Slika 3.1.1. *Harvard i Von-Neumann (Princeton) arhitekture mikroračunara.*

Kao što se sa slike može videti, konvencionalna Von-Neumann arhitektura mikrokontrolera poseduje jednu internu magistralu koja povezuje CPU sa programskom (ROM) i memorijom

podataka (RAM). Ovakav koncept povezuje kompletan memorijski prostor sa CPU jednom magistralom tako da podaci i instrukcije koriste istu magistralu.

Harvard arhitektura je noviji koncept u odnosu na Von-Neumann-ovu. Harvard arhitektura mikroračunara bazira na principu fizičkog razdvajanja internih magistrala za programsku memoriju (ROM) i memoriju podataka (RAM). Time je, zbog konkurentnog pristupa programskoj memoriji i memoriji podataka, ukupan protok podataka kroz CPU ubrzan. Rezolucija magistrale podataka osmobitnog mikrokontrolera Harvard arhitekture je osmobitna dok je rezolucija instrukcijske reči (rezolucija programske magistrale) uobičajeno veća od osam bitova (tipično 12, 14 ili 16). Instrukcije i podaci dobavljaju se jednovremeno i izvršavaju u jednom instrukcijskom ciklusu. Tipično za Harvard arhitekturu je da poseduje manje raspoloživih instrukcija nego Von-Neumann-ova, i da se većina instrukcija izvršava u jednom instrukcijskom ciklusu.

Mikrokontroleri Harvard arhitekture nazivaju se uopšteno RISC mikrokontrolerima (*Reduced Instruction Set Computer*) i predstavljaju popularnu arhitekturu modernih procesora. Jedan 8-bitni mikrokontroler ove arhitekture uobičajeno poseduje do nekoliko desetina instrukcija.

Mikrokontroleri Von-Neumann-ove arhitekture nazivaju se CISC mikrokontrolerima (*Complex Instruction Set Computer*). Magistrala podataka i programska magistrala su iste rezolucije, npr. osmobitna, međutim, zbog zauzimanja zajedničke magistrale podaci i instrukcije ne mogu biti dobavljani istovremeno. Tipični osmobitni mikrokontroleri Von-Neuman arhitekture poseduju preko dve stotine instrukcija.

Microchip mikrokontroleri PIC serije, koji su tema ove knjige, zasnovani su na modernoj Harvard arhitekturi. Na slici 3.1.2. prikazan je blok dijagram principijelne organizacije RISC mikrokontrolera PIC16F877.

Kapacitet programske FLASH EEPROM memorije iznosi 8192 14-bitnih reči dok je kapacitet RAM memorije podataka 512 bajta, od čega 368 bajta pripada korisničkom adresnom prostoru a ostatak registrima specijalne namene.

Programski brojač (brojač instrukcija) PC je 13-bitni, tako da se može adresirati ukupan adresni prostor programske memorije od 2^{13}=8192 instrukcijske reči. Magacinska memorija STACK nije u sastavu interne RAM memorije već predstavlja zasebnu celinu od osam nivoa za čuvanje povratnih adresa.

Mikrokontroler podržava dva načina adresiranja RAM memorije podataka, direktni i indirektni. Pri direktnom načinu adresiranja 9-bitna adresa se dobija tako što se na 7-bitnu adresu, dobijenu iz operacionog koda instrukcije, dodaju još dva bita iz STATUS-nog registra CPU. Indirektno adresiranje se izvodi pomoću MSB bita STATUS-nog registra i 8-bitnog registra specijalne namene, FSR (*File Select Register*) registra, koji ima ulogu pokazivača.

8-bitnoj aritmetičko-logičkoj ALU jedinici pridružen je jedan registar akumulatora w (*work register*) i registar stanja (*Status register*).

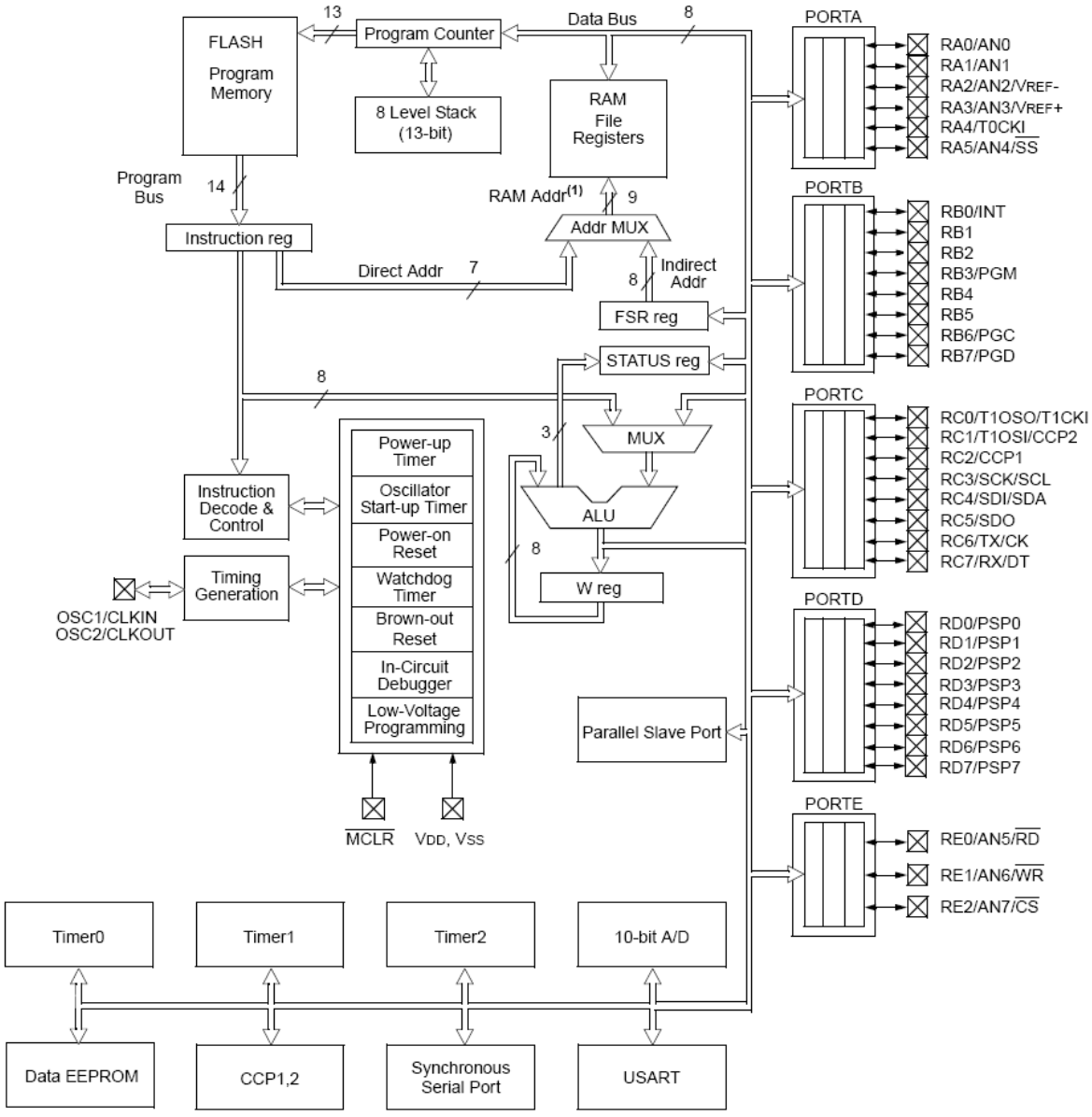
PIC16F877 poseduje određeni broj internih integrisanih programabilnih elektronskih kola za povećanje pouzdanosti sistema, minimizaciju cene razvoja aplikacije eliminisanjem eksternih komponenata, režim niske potrošnje (*Sleep*) i zaštitu koda. Pogodnost je i postojanje dvožičnog internog inteligentnog interfejsa za serijsko programiranje mikrokontrolera u sistemu ISSP (*In System Serial Programming*). Maksimalna frekvencija taktovanja je prema podacima proizvođača 20MHz. Sve instrukcije (35) izvršavaju se u jednom ciklusu, osim instrukcija grananja programa za čije izvršenje su potrebna dva instrukcijska ciklusa.

Za povezivanje sa spoljnim svetom implementirana su pet portova, A do E, od čega su tri 8-bitna, jedan 6-bitni i jedan 3-bitni. Linije pojedinih portova multipleksirane su analognim, digitalnim I/O funkcijama i funkcijama za sinhronu/asinhronu serijsku komunikaciju. Jedan od pet portova se može konfigurisati kao 8-bitni paralelni mikroprocesorski port.

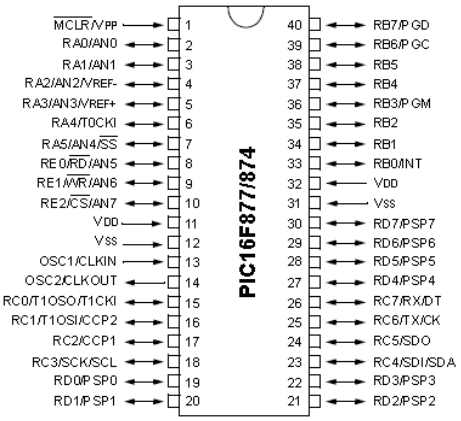
Mikrokontroler PIC16F877 opremljen je nizom periferijskih jedinica koje uređaj čine fleksibilnijim u realizaciji aplikacija. Od standardne opreme ovih komponenata treba pomenuti tajmere/brojače. Integrisana su dva 8-bitna tajmera, tajmer0 i tajmer2 kao i jedan 16-bitni tajmer1 koji ima mogućnost priključenja eksternog kristalnog oscilatora. Još jedna važna integrisana periferija je i 10-bitni A/D konvertor sa registrom sukcesivnih aproksimacija, koji

pripada klasi brzih konvertora, sa osam analognih vremenski multipleksiranih kanala. DATA EEPROM (*Electrical Erasable Programmable Read Only Memory*) memorija, koja takođe pripada periferijskom podsistemu, kapaciteta je 256 bajta. Realizovana je kao poseban fizički prostor a namenjena skladištenju programskih konstanti i/ili nepromenljivih parametara programa. Ova memorija je reprogramabilna postojana memorija kojoj se pristupa preko dva registra specijalne namene, adresnog i registra za podatke. Dva CCP modula namenjena su preciznom merenju/generisanju vremenskih intervala, npr. trajanja impulsa, kao i generisanju impulsno-širinski modulisanog signala.

Sinhroni serijski komunikacioni interfejsi kao što su IIC (*Inter-Integrated Circuit*) i SPI (*Serial Peripheral Interface*) pripadaju periferijskom bloku nadređenog sinhronog serijskog porta. Poslednja integrisana periferija je USART (*Universal Synchronous/Asynchronous Receiver Transmiter*) za specifičnu sinhrono/asinhronu komunikaciju sa spoljašnjim uređajima.



Slika 3.1.2. *Blok dijagram organizacije mikrokontrolera PIC16F877.*



Slika 3.1.3. *Pin konfiguracija mikrokontrolera PIC16F877 u PDIP-40 kućištu.*

Na slici 3.1.3. prikazana je pin konfiguracija četrdesetopinskog osmobitnog CMOS FLASH mikrokontrolera iz serije PIC16F877 u PDIP kućištu, proizvodnje Microchip.

3.2. ORGANIZACIJA MEMORIJSKOG PROSTORA

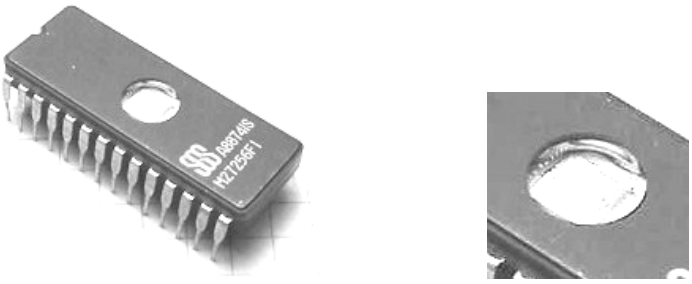
Memorija je, kao i procesor, fundamentalna komponenta savremenih računarskih sistema. Sastoji se od niza memorijskih reči (u najprostijem slučaju to su bajtovi) od kojih svaka ima jedinstvenu adresu. Prilikom izvršavanja programa, procesor na bazi vrednosti programskog brojača PC dobavlja instrukcije iz memorije i potom izvršava. Dobavljene instrukcije dodatno, u toku izvršenja, mogu zahtevati čitanje operanada ili upis podataka na druge memorijske lokacije.

Mikrokontroler PIC16F877 za smeštanje programa, odnosno operacionih kodova instrukcija, koristi tkzv. programsku memoriju iz koje procesor dobavlja instrukcije. Čitanje operanada i upis rezultata obrade instrukcija (podataka) vrši se unutar brze memorije podataka. Obe memorijske komponente ključne su i neophodne za izvršavanje programa a realizuju se različitim tehnologijama.

Generalno, memorije se dele na postojane (*non-volatile*) i ne postojane (*volatile*). Postojane memorije odlikuju osobinom čuvanja upisanih podataka i po ukidanju napona napajanja. Nasuprot njima, nepostojane memorije mogu čuvati podatke samo dok su priključene na napon napajanja.

PIC16C83/4 mikrokontroler, proizveden 1994 godine, je bio prvi PICmicro® koji je koristio EEPROM (*Electrically Erasable Programmable Read Only Memory*) tehnologiju za smeštanje i

čuvanje programa. Premda skuplja u odnosu na dotadašnju postojanu EPROM (*Erasable Programmable Read Only Memory*) memoriju, slika 3.2.1., koja se električnim putem programira a briše ultravioletnom radijacijom, primena postojane EEPROM memorije se pokazala pogodnijom za edukacione svrhe i izradu prototipova ugrađenih sistema. Zajedno sa ovom inovacijom, proizvođač integriše u čipu i periferijski memorijski modul tipa EEPROM koji programeru dopušta mogućnost postojanog skladištenja podataka nezavisno od memorije podataka i eliminiše potrebu za implementacijom spoljašnjeg EEPROM memorijskog modula. PIC16C83/4 i njegov naslednik PIC16F83/4 sa popularnom Flash EEPROM programskom memorijom ostaju jedini članovi porodice mikrokontrolera sa programskom memorijom tipa EEPROM do pojave serije PIC16F87X 1998 godine. Od 2000 godine Microchip se obavezao na opremanje najvećeg broja njegovih standardnih mikrokontrolera programskom memorijom tipa Flash EEPROM koja dopušta mogućnost reprogramiranja u samom sistemu, što ih čini pogodnim za upotrebu. Osim toga ova vrsta memorije se odlikuje velikom gustinom pakovanja (manja površina čipa po ćeliji u odnosu na konvencionalni EEPROM) i posebno, mogućnošću programiranja i brisanja u većim blokovima a ne bajt po bajt, kao u slučaju standardnih EEPROM memorija. Budući da je trajanje ciklusa upisa/brisanja EEPROM memorija tipično 4ms, ova osobina Flash EEPROM memorije je od značaja pri upisu veće količine podataka, zbog značajnog smanjenja vremena potrebnog za upis/brisanje.



Slika 3.2.1. *Izgled EPROM memorije u keramičkom kućištu sa prozorom za UV brisanje.*

Manje popularne memorije tipa EPROM, slika 3.2.1., mogu se naći i u plastičnim kućištima bez prozora za prosvetljavanje, OTP memorije (*One Time Programming*), i ugrađene su u neke serije PIC mikrokontrolera. Budući da se sadržaj takvih memorija ne može brisati, memorija se može programirati samo jednom.

Memorija podataka je brza statička RAM (*Random Access Memory*) nepostojana memorija koja obezbeđuje brzo čitanje operanada i upis rezultata obrade instrukcija u memorijske lokacije. Premda dovoljno brza i reprogramirljiva, glavni nedostaci su nepostojanost, mala gustina pakovanja i relativno visoka cena.

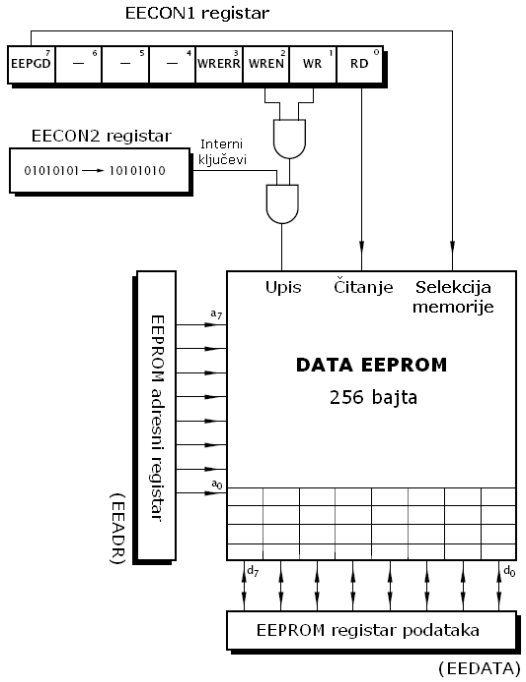
3.2.1. Periferijska Data EEPROM memorija

Jedna od dobrih osobina PIC MCU je posedovanje integrisane matrične memorije podataka tipa EEPROM koja je implementirana kao mikrokontrolerska periferija. Memorija je postojana i namenjena skladištenju nepromenljivih parametara programa ili parametara koji se povremeno menjaju. Tipičan primer primene ovih memorija su pametne kartice sa npr. PIN kodom, brojem računa i bezbednosnim podacima. Neki od ovih parametara, kao što je broj računa, se ne menjaju dok se drugi povremeno menjaju.

Data EEPROM memorija je kapaciteta 256 bajta i nije deo programske niti memorije podataka. Njoj se pristupa indirektno preko četiri registara specijalne namene SPR (*Special*

Purpose Register), mapiranih u RAM memoriji podataka. Adresni registar EEADR koristi se za adresiranje jedne od 256 memorijskih lokacija sa po osam bitova podataka. Registar podataka EEDATA namenjen je čuvanju podatka koji se upisuje u ili čita iz memorije, dok se kontrolni registri, EECON1 i EECON2, koriste za kontrolu operacija upisa i čitanja iz memorije.

Na slici 3.2.2. prikazana je logička organizacija Data EEPROM memorijskog modula mikrokontrolera PIC16F877 sa pripadajućim registrima specijalne namene.



Slika 3.2.2. *Logička organizacija Data EEPROM memorijskog modula PIC16F877 MCU.*

Ključne karakteristike memorijskog modula, od značaja za korisnika su:

- Istrajnost, izražena kroz minimalni broj ciklusa upisa/brisanja svake memorijske ćelije, koja iznosi 10^5 (tipično 10^6).
- Maksimalno vreme trajanja ciklusa upisa/brisanja iznosi 8ms (tipično 4ms).
- Maksimalno vreme čuvanja upisanih podataka iznosi 40 godina.

Adresni registar EEADR je osmobitni registar specijalne namene koji sadrži adresu jedne od 256 osmobitnih memorijskih lokacija. Mapiran je u adresnom prostoru RAM memorije podataka u memorijskoj banci 2 na adresi 10Dh. Koristi se za obe operacije upisa u ili čitanja iz Data EEPROM memorije.

Registar podataka EEDATA takođe je mapiran u banci 2 RAM memorije podataka na adresi 10Ch. Posle operacije čitanja Data EEPROM memorije u njemu se automatski smešta osmobitni podatak sa adresirane lokacije, dok se prilikom upisa u memoriju podatak iz EEDATA registra automatski prenosi u adresiranu memorijsku lokaciju.

Osmobitni kontrolni registri EECON1 i EECON2 mapirani su u banci 3 RAM memorije podataka na adresamai 18Ch i 18Dh, respektivno. EECON1 registar se koristi za kontrolu i monitoring operacija upisa i čitanja iz memorije a njegov sadržaj je prikazan u tabeli 3.2.1. Drugi kontrolni registar EECON2 nije fizički implementiran i čita se kao osmobitni niz nula. Pre startovanja svake operacije upisa u adresiranu memorijsku lokaciju Data EEPROM memorije, uzastopni niz tkzv. ključeva 55h i AAh mora biti upisan u EECON2 kontrolni registar bez prekidanja. Pomenuta sekvenca ključeva koristi se za deblokiranje (otključavanje) upisnog ciklusa i predstavlja zaštitni mehanizam memorije od nenamernog upisa.

Asemblerski potprogram za čitanje bajta podatka iz adresirane lokacije dat je u tabeli 3.2.2. i sadrži sledeće koračne zadatke:

- Kopiranje adrese memorijske lokacije iz korisničkog registra u EEADR registar.
- Selekcija Data EEPROM odredišne memorije.
- Setovanje RD bita EECON1 registra za startovanje operacije čitanja.
- U sledećem instrukcijskom ciklusu RD bit se automatski resetuje a pročitani podatak nalazi u EEDATA registru. Kopiranje pročitanog sadržaja iz EEDATA registra u korisnički registar.

EECON1 REGISTAR

	R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
	EEPGD	**-**	**-**	**-**	**WRERR**	**WREN**	**WR**	**RD**
	bit 7							bit 0
bit 7	**EEPGD:** Bit za selekciju programske/Data EEPROM memorije. 1 = Selektovana programska Flash EEPROM memorija. 0 = Selektovana Data EEPROM memorija. **Komentar:** Ovaj bit ne može biti promenjen u toku operacija upisa ili čitanja.							
bit 6-4	**Neimplementirani bitovi:** Čitaju se kao '0'.							
bit 3	**WRERR**: Bit za detekciju greške pri upisu u memoriju 1 = Operacija upisa prerano okončana. (MCLR Reset ili WDT reset u toku Normalnog Izvršenja instrukcija) 0 = Operacija upisa kompletirana.							
bit 2	**WREN**: Bit za omogućenje upisa u memoriju. 1 = Ciklus upis dopušten. 0 = Ciklus upisa zabranjen.							
bit 1	**WR**: Bit za inicijalizaciju ciklusa upisa u memoriju. 1 = Iniciran jedan ciklus upisa. Bit se automatski resetuje po okončanju ciklusa upisa a softverski može biti jedino setovan. 0 = Ciklus upisa kompletiran.							
bit 0	**RD**: Bit za inicijalizaciju ciklusa čitanja iz memorije. 1 = Iniciran jedan ciklus čitanja. Bit se automatski resetuje po okončanju ciklusa čitanja a softverski može biti jedino setovan. 0 = Nema inicijalizacije ciklusa čitanja.							

Tabela 3.2.1. *Sadržaj kontrolnog registra Data EEPROM i programske Flash EEPROM memorije, EECON1.*

```
      ; ********************************************************************
      ; *Funkcija: Potprogram za čitanje bajta podatka iz Data EEPROM memorije *
      ; *Ulaz: Korisnička promenljiva ADRESA u Banci 0 RAM memorije           *
      ; *Izlaz: Korisnička promenljiva PODATAK u Banci 0 RAM memorije         *
      ; ********************************************************************
EE_RD:                          ;Labela (simbolička adresa) potprograma
BCF       STATUS, RP1           ;
BCF       STATUS, RP0           ;Selekcija Banke 0
MOVF      ADRESA, W             ;Korisnička promenljiva ADRESA sadrži adresu mem. lok.
                                ;i deklarisana je u Banci 0 RAM memorije
BSF       STATUS, RP1           ;Selekcija Banke 2
MOVWF     EEADR                 ;Upis adrese mem. lok., koja se čita, u EEADR registar
BSF       STATUS, RP0           ;Selekcija Banke 3
BCF       EECON1, EEPGD         ;Selekcija odredišne Data EEPROM memorije
BSF       EECON1, RD            ;Start operacije čitanja
BCF       STATUS, RP0           ;Selekcija Banke 2
```

```
MOVF       EEDATA, W            ;Podatak u akumulatoru, EEDATA → W
BCF        STATUS, RP1          ;Selekcija Banke 0
MOVWF      PODATAK              ;Pročitani podatak smešta se u korisničku promenljivu
                                ;PODATAK koja je deklarisana u Banci 0 RAM memorije
RETURN                          ;Povratak u glavni program sa selektovanom Bankom 0
```

Tabela 3.2.2. *Asemblerski potprogram za čitanje bajta iz Data EEPROM memorije.*

Nakon setovanja RD bita kontrolnog registra EECON1<0> podatak je raspoloživ već u sledećem instrukcijskom ciklusu u registru EEDATA, što znači da se može pročitati sledećom instrukcijom. Podatak u EEDATA registru će biti sačuvan do sledećeg čitanja ili upisa u memoriju.

Upis bajta u Data EEPROM memorijsku lokaciju je proceduralno komplikovaniji budući da je neophodno osigurati uzastopni i neprekinuti upis ključeva u kontrolni registar EECON2 kako bi se izbegao nenamerni upis korumpiranog podatka usled greške u softveru ili otkaza hardvera, npr. pojava gliča na liniji napajanja MCU.

Asemblerski potprogram za upis bajta podatka u adresiranu memorijsku lokaciju Data EEPROM memorije prikazan je u tabeli 3.2.3. i sadrži sledeće koračne zadatke:

- Kopiranje adrese memorijske lokacije iz korisničkog registra u EEADR registar.
- Kopiranje podatka iz korisničkog registra u EEADR registar.
- Selekcija Data EEPROM odredišne memorije.
- Setovanje WREN bita u EECON1 registru za omogućenje ciklusa upisa.
- Onemogućenje svih prekida resetovanjem GIE bita.
- Upis ključeva 55h i AAh u kontrolni registar EECON2, respektivno.
- Setovanje WR bita EECON1 registra za startovanje ciklusa upisa.
- Resetovanje bita WREN za onemogućenje slučajnog ciklusa upisa.
- Omogućenje svih prekida setovanjem GIE bita.
- Provera završetka upisa bajta podatka testiranjem WR bita.

```
      ; ************************************************************************
      ; *Funkcija: Potprogram za upis bajta podatka u Data EEPROM memoriju     *
      ; *Ulaz: Korisničke promenljive ADRESA i PODATAK u Banci 0 RAM memorije  *
      ; *Izlaz: Ne postoji (selektovana Banka 0)                               *
      ; ************************************************************************
EE_WR:                          ;Labela (simbolička adresa) potprograma
BCF        STATUS, RP0          ;
BCF        STATUS, RP1          ;Banka 0
MOVF       ADRESA, W            ;Korisnička promenljiva ADRESA mapirana u Banci 0
BSF        STATUS, RP1          ;Banka 2
MOVWF      EEADR                ;Upis adrese mem. lokacije u EEADR registar
BCF        STATUS, RP1          ;Banka 0
MOVF       PODATAK, W           ; Korisnička promenljiva PODATAK mapirana u Banci 0
BSF        STATUS, RP1          ;Banka 2
MOVWF      EEDATA               ;Upis podatka U EEDATA registar
BSF        STATUS, RP0          ;Banka 3
BCF        EECON1, EEPGD        ;Selekcija odredišne Data EEPROM memorije
BSF        EECON1, WREN         ;Omogućenje upisa u Data EEPROM memoriju
                                ;
BCF        INTCON, GIE          ;Onemogućenje svih prekida (ako se koriste)radi
                                ;neprekinutog upisa ključeva u EECON2 registar
MOVLW      0x55                 ;
MOVWF      EECON2               ;Upis prvog ključa 55h u EECON2 registar
MOVLW      0xAA                 ;
MOVWF      EECON2               ;Upis drugog ključa AAh u EECON2 registar
```

```
BSF        EECON1, WR          ;Start operacije upisa u memorijsku lokaciju
                               ;
BCF        EECON1, WREN        ;Onemogućenje upisa u Data EEPROM memoriju
BSF        INTCON, GIE         ;Omogućenje svih prekida (ako se koriste)
                               ;
BTFSC      EECON1, WR          ;
GOTO       $-1                 ;Provera završetka upisa testiranjem WR bita
BCF        STATUS, RP0         ;
BCF        STATUS, RP1         ;Selekcija Banke 0
RETURN                         ;Povratak u glavni program sa selektovanom Bankom 0
```

Tabela 3.2.3. *Asemblerski potprogram za upis bajta u Data EEPROM memoriju.*

Ciklus upisa neće biti iniciran ako sekvenca ključeva za deblokiranje ciklusa nije prosleđena tačno i bez interferencije. Npr. prekidni signal, koji se može dogoditi u vreme trajanja ove sekvence, izazvaće prekidanje ciklusa upisa. Da bi se takva neželjena situacija sprečila potrebno je onemogućiti sve prekide, resetovanjem GIE bita INTCON registra, dok se ne završi inicijalizacija ciklusa upisa. Jedna od mogućih situacija koja može izazvati pogrešan upis podatka je resetovanje MCU pre kompletiranja ciklusa upisa (npr. izlazom Watchdog tajmera ili spoljašnjim resetom). U takvoj situaciji, kada je ciklus upisa prevremeno okončan, bit WRERR EECON1 registra se automatski setuje.

Izbor Data EEPROM memorije vrši se resetovanjem bita EEPGD kontrolnog registra EECON1. U suprotnom, biće selektovana Flash EEPROM programska memorija.

Setovanjem bita WREN EECON1 registra omogućava se upis u izabranu memoriju, posle čega sledi neprekinuti upis sekvence ključeva 55h i AAh, respektivno, u virtuelni kontrolni registar EECON2. Inicijalizacija se završava setovanjem WR bita kontrolnog registra EECON1, čime se startuje operacija upisa bajta podatka. Bit WREN mora strogo biti setovan pre nego se setuje WR bit. S toga se ne sme dopustiti jednovremeno setovanje oba bita. Kako se WREN bit ne resetuje hardverski, potrebno je osigurati programsko resetovanje ovog bita nakon inicijalizacije ciklusa upisa. Resetovanje WREN bita posle iniciranog upisnog ciklusa ne utiče na započeti tekući ciklus upisa i dodatno, sprečava mogućnost slučajnog upisa.

Već je rečeno da maksimalno trajanje ciklusa upisa u memorijsku lokaciju iznosi približno 8ms (tipično 4ms), što je drastično duže trajanje u odnosu na trajanje instrukcijskog ciklusa. Međutim, novi ciklus upisa ne sme biti inicijaliziran pre kompletiranja prethodnog. Budući da se WR bit kontrolnog registra EECON1 automatski hardverski briše po okončanju operacije upisa bajta, to se testiranjem ovog bita u petlji, metodom prozivke, utvrđuje kraj ciklusa upisa pa tek potom napušta potprogram. Takav pristup realizaciji potprograma garantuje da za vreme trajanja ciklusa upisa neće doći do promene sadržaja registara specijalne namene EEDATA, što može biti uzrok pogrešnog ishoda upisnog ciklusa.

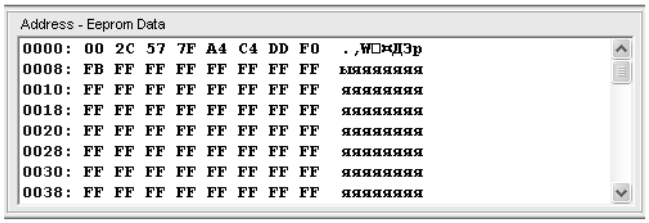
Implementacija metode prozivke u cilju utvrđivanja kraja ciklusa upisa troši dragoceno procesorsko vreme neefikasno, budući da je proces zaposlen čekanjem (*busy waiting*) za vreme trajanja ciklusa upisa od oko 8ms. Ovaj nedostatak se može izbeći korišćenjem prekidnog signala koji Data EEPROM periferijski modul hardverski generiše po kompletiranju ciklusa upisa, pod uslovom da je prekid omogućen. Naime, za povećanje efikasnosti procesora potrebno je, posle setovanja WR bita kontrolnog registra EECON1 (kraj inicijalizacije ciklusa upisa), programski omogućiti prekid sa memorijskog modula i izaći iz potprograma. Za dugo vreme trajanja ciklusa upisa procesor može raditi mnoge korisne operacije do trenutka kada ga memorijski modul prekidnim signalom obavesti o kraju ciklusa upisa.

Kontrolni mask bit za maskiranje/demaskiranje prekida sa Data EEPROM modula EEIE i odgovarajuća zastavica prekida (flag bit) EEIF nalaze se u kontrolnim registrima specijalne namene PIE2<4> i PIR2<4>, mapiranim na adresama RAM memorije podataka 8Dh i 0Dh, respektivno.

Osim promene sadržaja Data EEPROM memorijskog modula pod programskom kontrolom, postoji i mogućnost inicijalizacije stanja modula u toku eksternog programiranja MCU pomoću posebnog uređaja-programatora. Naime, adresni prostor programske memorije koji ne pripada prostoru za smeštanje korisničkog programa odnosi se na specijalno testiranje/konfiguraciju memorijskog prostora i nalazi se u opsegu adresa 2000h – 3FFFh. Ovom se adresnom prostoru pristupa samo u toku eksternog programiranja uređaja. Npr. konfiguraciona reč MCU je na adresi 2007h. Data EEPROM modul takođe pripada ovom prostoru u opsegu adresa 2100h – 21FFh. Za čuvanje, na primer, 10 diskretnih vrednosti funkcije sin(α) za opseg uglova od 0^0 do 90^0 sa korakom promene od 10^0, deo izvornog programskog koda bio bi oblika:

```
Org     2100h     ;početak adresnog prostora Data EEPROM memorijskog modula
SINUS   DE   00h, 2Ch, 57h, 7Fh, 0A4h, 0C4h, 0DDh, 0F0h, 0FBh, 0FFh
```

gde je asemblerskom direktivom DE specificirana lista podataka odvojenih zapetama koji se upisuju u Data EEPROM. Posle programiranja MCU, sadržaj Data EEPROM modula će izgledati kao na slici 3.2.3.



Slika 3.2.3. *Sadržaj prvih 10 lokacija Data EEPROM memorijskog modula mikrokontrolera PIC16F877 koji predstavlja look-up tabelu sinusne funkcije.*

Podaci upisani u Data EEPROM memoriju na ovaj način mogu docnije biti čitani ili modifikovani u toku izvršenja programa. Na primer, za čitanje vrednosti funkcije sin(50) adresira se lokacija 05h i potom čita njen sadržaj C4h ili 196 decimalno (196/256=0.76525).
Čitanje i upis u Data EEPROM memoriju mogu se vršiti pod kontrolom programa koji se izvršava (interno) ili eksterno na bazi uređaja za programiranje mikrokontrolera-programatora.

Data EEPROM memorija poseduje mehanizam za zaštitu njenog koda od eksternog čitanja. Programiranjem tkzv. konfiguracione reči mikrokontrolera, tačnije jednog bita ove reči, bita CPD (*Code Protect Data EEPROM*), može se onemogućiti/omogućiti eksterno čitanje koda Data EEPROM memorije programatorom. Međutim, ovaj postupak ne sprečava mogućnost internog čitanja ili modifikacije podataka u Data EEPROM memoriji pod kontrolom programa. Kada je zaštita koda omogućena (CPD resetovan) spoljašnji pristup memoriji preko ICSP (*In Circuit Serial Programming*), odnosno, pristup memoriji programatorom je onemogućen. Time je sprećena mogućnost neovlašćenog čitanja koda, upisanog u ovu memoriju.

CCS C kompajler poseduje ugrađene funkcije za čitanje i upis bajta podatka u Data EEPROM memoriju oblika:

```
Data=read_eeprom(address)-čitanje bajta podatka sa specificirane lokacije (address).

write_eeprom(address,data)-upis bajta podatka (data) na specificiranu lokaciju
(address). Iz funkcije se izlazi po kompletiranju ciklusa upisa.
```

U tabeli 3.2.4. data je funkcija, pisana na C jeziku, za brojanje događaja i trajno pamćenje u Data EEPROM memoriji. Funkcija je pisana tako da se na poziv najpre pročita aktuelni sadržaj brojača, sačuvan na odgovarajućim lokacijama Data EEPROM memorije, potom se inkrementira (poveća za jedan) pa ponovo upiše na iste memorijske lokacije. Dakle, svaki poziv

funkcije inkrementira aktuelni sadržaj brojača i trajno pamti novu vrednost, što predstavlja funkciju odometra. Brojac je trobajtna promenljiva čija maksimalna vrednost može biti 2^{24}-1=16777215.

```
////////////////////////////////////////////////////////////////////////////
//
// Funkcija: Odometar()
//
// Poziv ove funkcije rezultuje čitanjem trobajtne vrednosti brojčanika u Data
// EEPROM memoriji na lokacijama 0x10, 0x11 i 0x12, respektivno (2^24-1 je
// maksimalni broj koji registruje brojčanik, npr. broj pređenih kilometara),
// inkrementiranjem brojčanika i upisom inkrementirane vrednosti brojčanika u
// postojanu Data EEPROM memoriju na istim lokacijama.
//
////////////////////////////////////////////////////////////////////////////

void Odometar(void)
{
unsigned int brojcanik[3];          //Deklaracija 3 bajtne promenljive(brojcanik)

//Citanje aktuelne vrednosti brojcanika iz Data EEPROM memorije

brojcanik[0] = read_eeprom(0x10);   //Čitanje LSB bajta na adresi 16dec (0x10)
brojcanik[1] = read_eeprom(0x11);   //Čitanje NSB bajta na adresi 17dec (0x11)
brojcanik[2] = read_eeprom(0x12);   //Čitanje MSB bajta na adresi 18dec (0x12)

//Inkrementiranje brojcanika

if(++brojcanik[0] != 0) break;      //Ako je inkrementirani LSB bajt ≠0 izađi
                                    //iz kontrolne naredbe if (kraj), u suprotnom
                                    //inkrementiraj NSB bajt
else if(++brojcanik[1] != 0) break; //Ako je inkrementirani NSB bajt ≠0 izađi
                                    //iz kontrolne naredbe if (kraj), u suprotnom
                                    //inkrementiraj MSB bajt
else brojcanik[2]++;                //inkrementiranje MSB bajta

//Upis inkrementirane vrednosti brojcanika u Data EEPROM memoriju

write_eeprom(0x10, brojcanik[0]);   //Upis novog LSB bajta na adresi 16dec (0x10)
write_eeprom(0x11, brojcanik[1]);   //Upis novog NSB bajta na adresi 17dec (0x11)
write_eeprom(0x12, brojcanik[2]);   //Upis novog MSB bajta na adresi 18dec (0x12)
}
```

Tabela 3.2.4. *C funkcija odometra koja koristi Data EEPROM memoriju.*

3.2.2. Programska Flash EEPROM memorija

Programska memorija, integrisana u čipu MCU, je tipa Flash EEPROM, kapaciteta osam kilo reči (8192 lokacije sa po 14 bita) a namenjena je čuvanju programskog koda glavnog programa, potprograma i prekidnih rutina. Kompletan memorijski prostor adresira 13-bitni programski brojač PC (2^{13}=8192). Reset vektor je na adresi 0000h a vektor prekida na adresi 0004h. Primenjena tehnika straničenja deli memoriju na četiri stranice od po dve kilo reči.

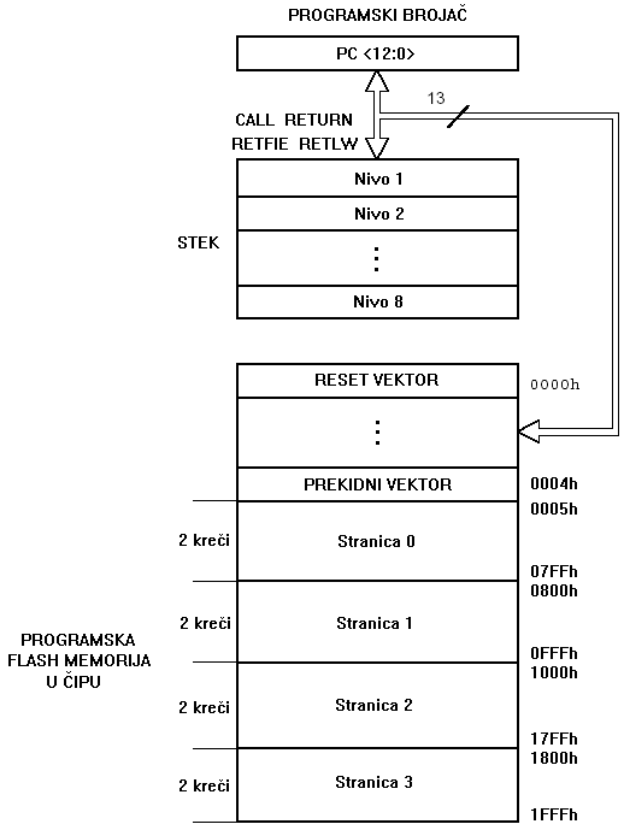
Za čuvanje povratnih adresa prilikom pozivanja i vraćanja iz potprograma ili prekidne rutine koristi se zasebna celina od osam nivoa-stek.

Na slici 3.2.4. prikazana je principijelna organizacija prostora programske memorije PIC16F877 MCU. U odnosu na konvencionalnu Data EEPROM memoriju, minimalni broj ciklusa

upisa/brisanja programske memorije je 100 puta manji i iznosi 10^3 dok je maksimalno vreme upisa u memorijsku lokaciju identično, oko 8ms. U skladu s tim, ova memorija je pogodnija za čuvanje konstantnih podataka, npr. look-up tabela, nego podataka koji zahtevaju čestu modifikaciju. Flash EEPROM memorija ima manju geometriju u odnosu na konvencionalni EEPROM (manja površina čipa po ćeliji) ali glavna prednost ostaje brzina upisa/brisanja, imajući u vidu blokovski pristup.

Ove memorije se proizvode u dve varijante: NAND i NOR Flash memorije. Razlika se sastoji u strukturi veza između memoriskih ćelija, po čemu su varijante memorija i imenovane (NOR-paralelna veza memorijskih ćelija, što podseća na paralelnu vezu tranzistora CMOS NOR logičkog kola i NAND-serijska veza memorijskih ćelija, kao i tranzistora CMOS NAND logičkog kola). Takođe, postoji razlika i u strukturi interfejsa za upis i čitanje memorije. NOR podržava slučajni pristup pri čitanju dok NAND podržava stranični pristup.

Kako serijska veza memorijskih ćelija troši manje fizičkog prostora za realizaciju u odnosu na paralelnu, to je ključni cilj realizacije NAND Flash memorija povećanje kapaciteta za zadati gabarit memorije i time smanjenje cene. Primera radi, NAND Flash memorija određenog kapaciteta zahvata oko 60% površine ekvivalentne NOR Flash memorije. Intencija NAND flash memorija je zamena magnetskih hard diskova ovom vrstom solid-state memorija.

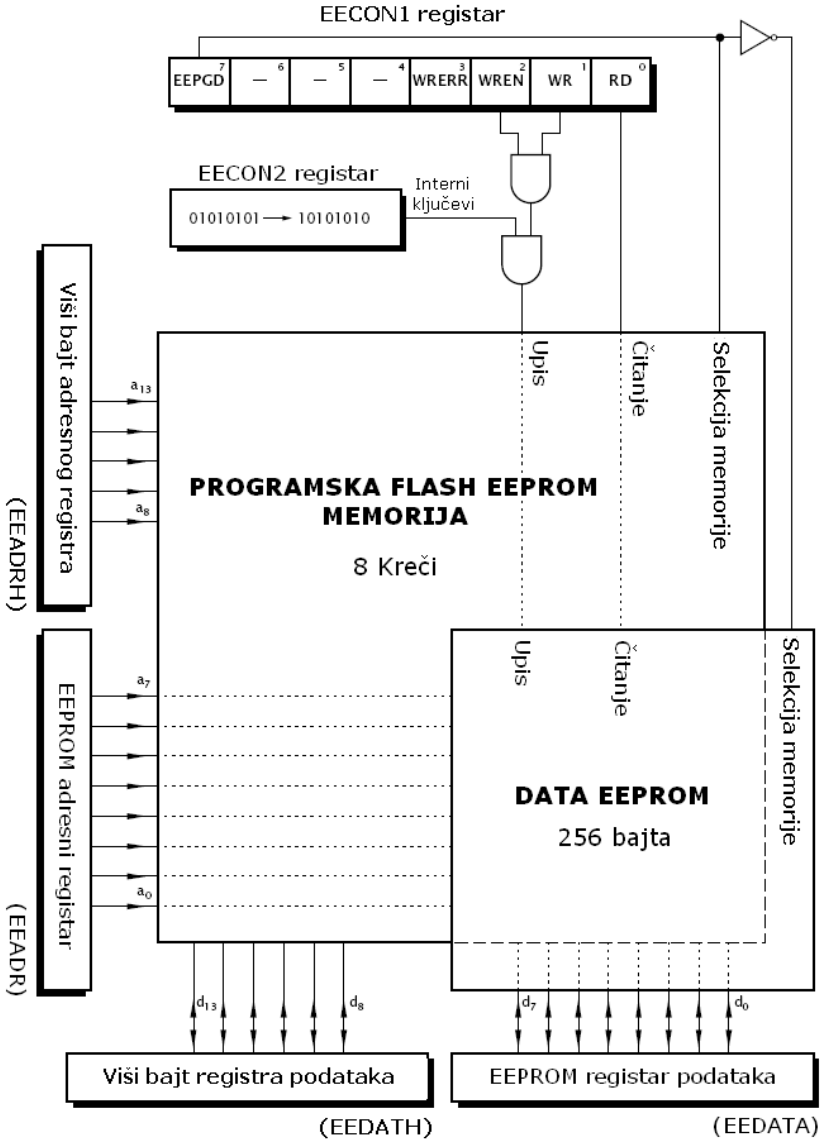


Slika 3.2.4. *Organizacija prostora programske Flash EEPROM memorije PIC16F877 MCU.*

Na slici 3.2.5. prikazana je logička organizacija Data EEPROM memorije sa nadodatom Flash EEPROM programskom memorijom i pripadajućim adresnim, registrima podataka i kontrolnim registrima EECON1 i EECON2. Može se primetiti da su adresni registar EEADR i registar podataka EEDATA kao i oba kontrolna registra, zajednički resursi za jedan i drugi memorijski modul. Zbog razlika u kapacitetu i rezoluciji, programskoj memoriji su pridodata još

dva registra, adresni EEADRH i registar podataka EEDATH, koji predstavljaju više bajtove adresnog i registra podataka Flash programske memorije, respektivno. Izbor ciljnog memorijskog modula vrši se odgovarajućim upisom u MSB bitsku poziciju kontrolnog registra EECON1 (EEPGD bit). Resetovanjem ovog bita pristupa se Data EEPROM memorija dok se setovanjem pristupa Flash EEPROM programskoj memoriji.

Čitanje Flash EEPROM memorijskog modula je slično čitanju Data EEPROM memorije sa jedinom razlikom udvojenih registara za adrese i podatke. Međutim, postoji interakcija sa programskim kodom, koji se čuva u ovoj memoriji, dobavlja iz nje i potom izvršava. S obzirom na ovaj dualizam, u kodu za čitanje programske memorije neophodno je, po setovanju bita RD kontrolnog registra EECON1 (start ciklusa čitanja), umetnuti dve NOP instrukcije kako bi se u sledeća dva instrukcijska ciklusa omogućilo čitanje dva bajta podatka (14-bitna rezolucija jedne reči programske memorije). Potprogram napisan na asembleru za čitanje programske Flash EEPROM memorije dat je u tabeli 3.2.5.



Slika 3.2.5. *Logička organizacija DATA EEPROM i programske Flash memorije PIC16F877 MCU.*

```
    ; ************************************************************************
    ; *Funkcija: Potprogram za čitanje reči iz Flash EEPROM programske memorije  *
    ; *Ulaz: Korisničke promenljive NIŽA_ADR i VIŠA_ADR u Banci 0 RAM memorije   *
    ; *Izlaz: Korisničke promenljive NIŽI_POD i VIŠI_POD u Banci 0 RAM memorije  *
    ; ************************************************************************
Flash_RD:                   ;Labela (simbolička adresa) potprograma
BCF       STATUS, RP1       ;
BCF       STATUS, RP0       ;Selekcija Banke 0

MOVF      NIŽA_ADR, W       ;Korisnička promenljiva NIŽA_ADR deklarisana u Banci 0
BSF       STATUS, RP1       ;Selekcija Banke 2
MOVWF     EEADR             ;Upis nižeg bajta adrese u EEADR registar
BCF       STATUS, RP1       ;Selekcija Banke 0
MOVF      VIŠA_ADR, W       ;Korisnička promenljiva VIŠA_ADR deklarisana u Banci 0
BSF       STATUS, RP1       ;Selekcija Banke 2
MOVWF     EEADRH            ;Upis višeg bajta adrese u EEADRH registar
BSF       STATUS, RP0       ;Selekcija Banke 3

BSF       EECON1, EEPGD     ;Selekcija odredišne Flash EEPROM memorije
BSF       EECON1, RD        ;Start operacije čitanja

NOP                         ;Neophodno kašnjenje od dva instrukcijska ciklusa
NOP                         ;zbog uzastopnog čitanja dva bajta podatka

BCF       STATUS, RP0       ;Selekcija Banke 2
MOVF      EEDATA, W         ;Niži bajt podatka u akumulatoru, EEDATA → W
BCF       STATUS, RP1       ;Selekcija Banke 0
MOVWF     NIŽI_POD          ;Upis u korisničku promenljivu, deklarisanoj u Banci 0
BSF       STATUS, RP1       ;Selekcija Banke 2
MOVF      EEDATH, W         ;Viši bajt podatka u akumulatoru, EEDATH → W
BCF       STATUS, RP1       ;Selekcija Banke 0
MOVWF     VIŠI_POD          ;Upis u korisničku promenljivu, deklarisanoj u Banci 0
RETURN                      ;Povratak u glavni program sa selektovanom Bankom 0
```

Tabela 3.2.5. *Asemblerski potprogram za čitanje reči iz Flash EEPROM programske memorije.*

Upis u programsku Flash memoriju zahteva nešto drugačiji postupak u odnosu na upis u Data EEPROM memoriju. Osim razlike koja se odnosi na udvojene registre za adresiranje i podatke, operacija upisa u Flash programsku memoriju je jedinstvena po tome što mikrokontroler ne izvršava instrukcije za vreme trajanja upisnog ciklusa. Takvo stanje ne predstavlja sleep režim, budući da kristalni taktni oscilator i periferije nastavljaju sa radom. Takođe se detektuju i prekidni događaji (ako su omogućeni) i stavljaju u red čekanja dok se upisni ciklus ne završi. Po kompletiranju ciklusa upisa program će granati na prekidni vektor ako je prekid omogućen i ako su se prekidi dogodili za vreme trajanja ciklusa upisa u Flash EEPROM.

Druga važna razlika u odnosu na upis u Data EEPROM memoriju je bit WRT konfiguracione reči mikrokontrolera koji, kada je resetovan, sprečava upis u programsku memoriju pod kontrolom programa (interni upis). Potprogram napisan na asembleru za upis reči u programsku Flash EEPROM memoriju prikazan je u tabeli 3.2.6.

```
 ; ************************************************************************
 ; *Funkcija: Potprogram za upis reči u Flash EEPROM programsku memoriju      *
 ; *Ulaz: Korisničke promenljive NiŽA_ADR, VIŠA_ADR, NIŽI_POD i VIŠI_POD u Banci 0  *
 ; *Izlaz: Ne postoji (selektovana Banka 0)                                   *
```

```
 ; ********************************************************************************
Flash_WR:                       ;Labela (simbolička adresa) potprograma
BCF       STATUS, RP1           ;
BCF       STATUS, RP0           ;Selekcija Banke 0

MOVF      NIŽA_ADR, W           ;Korisnička promenljiva NIŽA_ADR deklarisana u Banci 0
BSF       STATUS, RP1           ;Selekcija Banke 2
MOVWF     EEADR                 ;Upis nižeg bajta adrese u EEADR registar
BCF       STATUS, RP1           ;Selekcija Banke 0
MOVF      VIŠA_ADR, W           ;Korisnička promenljiva VIŠA_ADR deklarisana u Banci 0
BSF       STATUS, RP1           ;Selekcija Banke 2
MOVWF     EEADRH                ;Upis višeg bajta adrese u EEADRH registar
BCF       STATUS, RP1           ;Selekcija Banke 0

MOVF      NIŽI_POD, W           ;Korisnička promenljiva NIŽI_POD deklarisana u Banci 0
BSF       STATUS, RP1           ;Selekcija Banke 2
MOVWF     EEDATA                ;Upis nižeg bajta podatka u EEDATA registar
BCF       STATUS, RP1           ;Selekcija Banke 0
MOVF      VIŠI_POD, W           ;Korisnička promenljiva VIŠI_POD deklarisana u Banci 0
BSF       STATUS, RP1           ;Selekcija Banke 2
MOVWF     EEDATH                ;Upis višeg bajta podatka u EEDATH registar
BSF       STATUS, RP0           ;Banka 3

BSF       EECON1, EEPGD         ;Selekcija odredišne Flash EEPROM memorije
BSF       EECON1, WREN          ;Omogućenje upisa u Flash EEPROM memoriju
                                ;
BCF       INTCON, GIE           ;Onemogućenje svih prekida (ako se koriste)radi
                                ;neprekinutog upisa ključeva u EECON2 registar
MOVLW     0x55                  ;
MOVWF     EECON2                ;Upis prvog ključa 55h u EECON2 registar
MOVLW     0xAA                  ;
MOVWF     EECON2                ;Upis drugog ključa AAh u EECON2 registar
BSF       EECON1, WR            ;Start operacije upisa u memorijsku lokaciju

NOP                             ;Neophodno kašnjenje od dva instrukcijska ciklusa
NOP                             ;za pripremu operacije upisa u Flash memoriju

BCF       EECON1, WREN          ;Onemogućenje upisa u Flash EEPROM memoriju
BSF       INTCON, GIE           ;Omogućenje svih prekida (ako se koriste)
                                ;
BTFSC     EECON1, WR            ;
GOTO      $-1                   ;Provera završetka upisa testiranjem WR bita
BCF       STATUS, RP0           ;
BCF       STATUS, RP1           ;Selekcija Banke 0
RETURN                          ;Povratak u glavni program sa selektovanom Bankom 0
```

Tabela 3.2.6. *Asemblerski potprogram za upis reči u Flash EEPROM programsku memoriju.*

Instrukciju setovanja bita WR (start upisnog ciklusa) slede dve NOP instrukcije, neophodne za pripremu operacije upisa u Flash memoriju. Mikrokontroler potom blokira izvođenje instrukcija za vreme od oko 4-8ms, koliko traje ciklus upisa. Posle kompletiranja ciklusa upisa mikrokontroler će nastaviti sa izvođenjem instrukcije koja sledi drugu NOP instrukciju potprograma za upis reči u Flash memoriju.

Na kraju ciklusa upisa bit WR se automatski hardverski briše. Pošto mikrokontroler ne izvršava instrukcije u toku trajanja ciklusa upisa, nije nužno proveravati WR bit u cilju

određivanja kraja ciklusa upisa. Značenje bita WRERR kontrolnog registra EECON1 je isto kao i za Data EEPROM memoriju.

Vrednost upisana u programsku Flash memoriju ne mora biti validna instrukcija. Prema tome, 14-bitni podaci mogu biti upisani u lokacije programske memorije kao npr. kalibracioni parametri, serijski brojevi, pakovani 7-bitni ASCII kod i slično. Izvršenje takvih podataka, koji ne predstavljaju validne instrukcije na lokacijama programske memorije, rezultuje izvršenjem NOP (*No Operation*) instrukcije.

Kao i kod upisa u Data EEPROM memorijski modul na kraju ciklusa upisa generiše se prekidni signal, ako je omogućen, i setuje odgovarajuća zastavica prekida. Kontrolni mask bit za maskiranje/demaskiranje prekida sa Flash EEPROM ili Data EEPROM modula EEIE i odgovarajuća zastavica prekida (flag bit) EEIF nalaze se u kontrolnim registrima specijalne namene PIE2<4> i PIR2<4>, mapiranim na adresama RAM memorije podataka 8Dh i 0Dh, respektivno.

Kao i kod Data EEPROM memorije, čitanje i upis podataka u programsku Flash memoriju može se vršiti interno pod kontrolom programa ili eksterno programatorom (ICSP). Generalna zaštita programske memorije vrši se programiranjem dva konfiguraciona bita konfiguracione reči CP0 i CP1 (*Code Protect*), čije kombinacione vrednosti određuju granice zaštićenog/ nezaštićenog adresnog prostora programske memorije, tabela 3.2.7. Međutim, ova dva duplirana bita konfiguracione reči CP0 i CP1 kao i bit WRT (*Write Enable*), za zaštitu koda programske Flash memorije od internog/eksternog upisa/čitanja, imaju različite konkretne efekte na upis/čitanje ove memorije, vidi tabelu 3.2.8. Iz tabele 3.2.8. se može primetiti da bilo koja kombinaciona vrednost ova tri bita nema uticaja na interno čitanje programske memorije (čitanje pod kontrolom programa) koje je uvek omogućeno.

CP1	CP0	Adresni prostor programske Flash EEPROM memorije
0	0	Zaštićen ceo adresni prostor od 0000h do 1FFFh
0	1	Zaštićena samo gornja polovina adresnog prostora od 1000h do 1FFFh
1	0	Zaštićen samo deo gornjeg adresnog prostora (256 mem. reči) od 1F00h do 1FFFh
1	1	Nezaštićen adresni prostor cele programske memorije

Tabela 3.2.7. *Kombinacione vrednosti konfiguracionih bitova CP0 i CP1 i odgovarajući efekti na zaštićen/nezaštićen adresni prostor programske Flash memorije.*

Konfiguracioni bitovi			Lokacije programske memorije	Interno čitanje	Interni upis	ICSP čitanje	ICSP upis
CP1	CP0	WRT					
0	0	x	Cela programska memorija	Da	Ne	Ne	Ne
0	1	0	Nezaštićeni prostor	Da	Ne	Da	Ne
0	1	0	Zaštićeni prostor	Da	Ne	Ne	Ne
0	1	1	Nezaštićeni prostor	Da	Da	Da	Ne
0	1	1	Zaštićeni prostor	Da	Ne	Ne	Ne
1	0	0	Nezaštićeni prostor	Da	Ne	Da	Ne
1	0	0	Zaštićeni prostor	Da	Ne	Ne	Ne
1	0	1	Nezaštićeni prostor	Da	Da	Da	Ne
1	0	1	Zaštićeni prostor	Da	Ne	Ne	Ne
1	1	0	Cela programska memorija	Da	Ne	Da	Da
1	1	1	Cela programska memorija	Da	Da	Da	Da

Tabela 3.2.8. *Uticaj konfiguracionih bitova CP0, CP1 I WRT na interno/eksterno čitanje/upis u programsku Flash memoriju.*

Jednom omogućena zaštita koda Data EEPROM ili Flash EEPROM programske memorije može biti uklonjena samo potpunim brisanjem uređaja u postupku reprogramiranja.

Kao i za Data EEPROM memoriju, CCS C kompajler poseduje ugrađene funkcije za čitanje i upis reči (16-bitna reč) u programsku Flash EEPROM memoriju oblika:

```
data=read_program_eeprom(addr)-čitanje reči sa specificirane lokacije (address).

write_program_eeprom(address,data)-upis reči (data) na specificiranu lokaciju
(address). Iz funkcije se izlazi po kompletiranju ciklusa upisa.
```

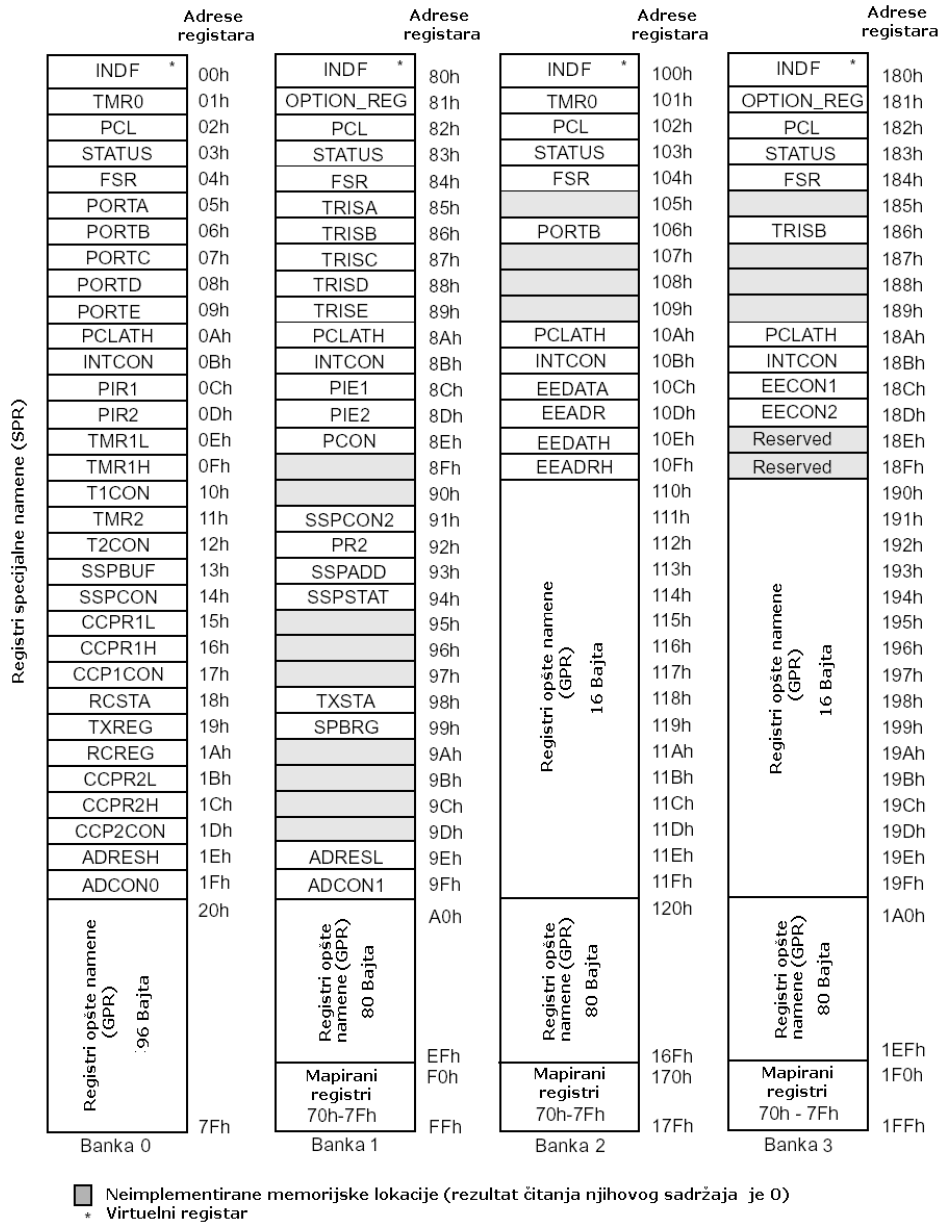
gde je podatak za upis ili čitanje 16-bitni dok je adresa 16-bitna za PIC16Fxxx seriju a 32-bitna za 18Fxxxx seriju mikrokontrolera.

3.2.3. SRAM memorija podataka

Memorija podataka je nepostojana brza statička RAM (*Random Access Memory*) memorija sa direktnim pristupom, kapaciteta 512 bajta. Namenjena je privremenom čuvanju podataka, međurezultata i rezultata obrade u toku izvršenja programa. Od ukupnog adresnog prostora memorije, 368 bajta pripada korisničkom prostoru (*user space*), manji broj osmobitnih registara nije fizički implementiran dok je ostatak rezervisan za sistemske potrebe (*system space*). Ova vrsta memorije je neophodna imajući u vidu da je za izvršenje određenih instrukcija potreban ciklus upis operanada u odgovarajuće memorijske lokacije. Adresni prostor programske Flash EEPROM memorije se za ove svrhe ne može koristiti imajući u vidu nedopustivo dugo vreme potrebno za upis u lokaciju, 4-8ms. Ciklus čitanja memorije nije kritičan budući da je jednako brz kako za Flash EEPROM tako i za SRAM memoriju. Da bi se osigurao veoma brz upis u memorijsku lokaciju u toku trajanja kratkog instrukcijskog ciklusa potrebno je koristiti drugu vrstu memorije kao što je SRAM. Memorija je podeljena u četiri memorijske banke kapaciteta 128 bajta, slika 3.2.6. Za rad sa grupom registara koji pripadaju jednoj memorijskoj banci potrebno je najpre selektovati odgovarajuću banku i potom pristupati registrima. Selekcija banke se vrši programiranjem statusnog registra ALU jedinice STATUS. Kombinacione vrednosti dva bita ovog registra RP0 i RP1 određuju selektovanu memorijsku banku prema tabeli 3.2.9.

RP1	RP0	Selektovana memorijska banka
0	0	Banka 0
0	1	Banka 1
1	0	Banka 2
1	1	Banka 3

Tabela 3.2.9. *Kombinacione vrednosti dva bita STATUS registra za izbor memorijske banke.*



Slika 3.2.6. *Adresni prostor RAM memorije podeljen na četiri memorijske banke sa registrima specijalne (SPR-special purpose registers) i opšte namene (GPR-general purpose registers).*

Imenovani registri na nižim adresama bilo koje od četiri banke, za koje je rezervisan odgovarajući adresni prostor RAM memorije, jesu registri specijalne namene (*SPR-special purpose registers*). Ove registre koriste jezgro CPU (STATUS, PCL, PCLATH, FSR itd) i periferijski moduli integrisani u čipu (TMR0, TMR1, RCSTA, CCPCON itd.) za statusne informacije modula, kontrolu operacija kao i za podatke. U skladu s tim, registri specijalne namene se mogu podeliti u tri grupe:

- Statusni
- Kontrolni
- SPR za podatke

Na primer, statusni registar CPU jedinice ima simboličko ime STATUS dok dva statusna registra USART periferijske jedinice RCSTA i TXSTA pripadaju prijemniku i predajniku integrisanog serijskog komunikacionog interfejsa USART-a, respektivno. Od kontrolnih registara mogu se izdvojiti INTCON za kontrolu prekida sa standardnih izvora prekidnog signala, T1CON za kontrolu tajmerske jedinice 1, ADCON0 i ADCON1 za kontrolu A/D konvertorske jedinice, EECON0 i EECON1 za kontrolu Data EEPROM i programske Flash memorije itd. Registri za podatke uglavnom su namenjeni privremenom čuvanju npr. rezultata 10-bitne A/D konverzije ADRESL i ADRESH, podatka pročitanog iz Data EEPROM jedinice EEDATA, podatka pročitanog ili upisanog na odgovarajući port MCU PORTA, PORTB itd. Prema direkciji registri podataka se mogu podeliti na ulazne (za čitanje), izlazne (za upis) i bidirekcione ili ulazno izlazne (za čitanje i upis).

Neki često korišćeni registri specijalne namene kao što su STATUS, INTCON, PCLATH, PCL, FSR, OPTION_REG itd., mapirani su u više banaka ili čak u sve četiri, slika 3.2.6., što je učinjeno u cilju bržeg pristupa registrima. Eventualna nužna promena memorijske banke može se na taj način izbeći i time redukovati programski kod.

Osmobitne memorijske lokacije na višim adresama predstavljaju korisnički adresni prostor a čine ih tkzv. registri opšte namene (GPR-general purpose registers). Ovaj veći deo ukupnog adresnog prostora RAM memorije namenjen je rezervaciji prostora za programske promenljive, konstante, programski stek i slično, jednom rečju namenjen je programeru.

Svaki registar RAM memorije podataka ima pridruženu jedinstvenu adresu, npr. PORTA je na adresi 05h. Hardver mikrokontrolera PIC16F877 podržava dva načina pristupa registrima RAM memorije, odnosno, dva načina adresiranja RAM memorije - direktni i indirektni.

3.2.4. Načini adresiranja RAM memorije

Jedan od ozbiljnih nedostataka Microchip-ovog mikrokontrolera PIC16F877 je podrška malom broju načina adresiranja RAM memorije – direktnom i indirektnom.

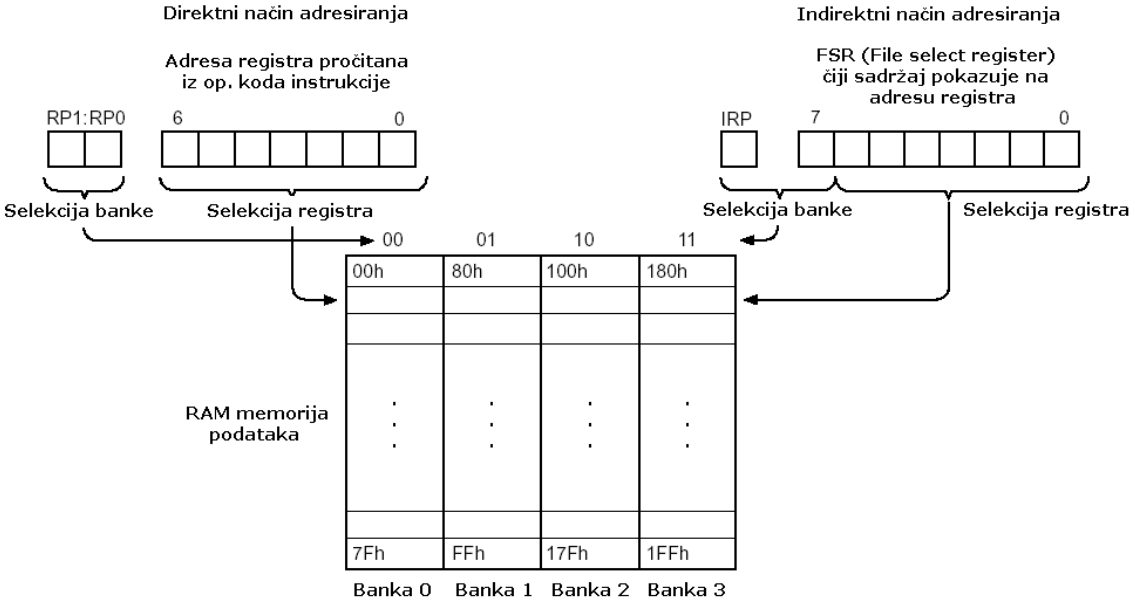
Na slici 3.2.7. ilustrativno su prikazana dva devetobitna načina adresiranja. Direktni način adresiranja koristi sedmobitnu adresu registra operanda, dobijenu iz operacionog koda instrukcije, na koju se dodaju još dva bita RP0 i RP1 STATUS registra za selekciju memorijske banke. Sedam bita adrese registra operanda dopušta adresiranje maksimalno 2^7=128 registara, koliko svaka od memorijskih banaka poseduje. Najviša dva bita, dobijena iz STATUS registra RP0 i RP1, koja grade devetobitnu adresu koriste se za selekciju jedne od četiri banaka. Potpuno je trivijalno da pristup registru operanda u odgovarajućoj banci mora biti propraćen prethodnom selekcijom odgovarajuće memorijske banke. U suprotnom, ako je programskim kodom selektovana druga aktuelna memorijska banka a ne izvrši se odgovarajuća promena banke pre pristupa operandu, pristupiće se zapravo registru koji pripada aktuelnoj banci selektovanoj programskim kodom a ne registru operanda.

Bilo koja instrukcija kojom se čita, modifikuje ili prepisuje sadržaj nekog od SPR ili GPR registara RAM memorije predstavlja primer direktnog adresiranja. U tabeli 3.2.10. dat je primer dela koda koji predstavlja direktni način adresiranja RAM memorije.

```
BSF       STATUS, RP0
BCF       STATUS, RP1     ;selekcija Banke 1. Registar TRISA mapiran je u Banci 1
MOVWF     TRISA           ;adresa TRISA registra sadržana je u op. kodu instrukcije
                          ;movwf TRISA
```

Tabela 3.2.10. *Primer koda koji predstavlja direktni način adresiranja SRAM memorije za podatke.*

Sve tri instrukcije gornjeg primera koriste direktni način adresiranja sa jedinom razlikom što se STATUS registru može pristupati iz bilo koje memorijske banke, budući da je mapiran u sve četiri, dok je TRISA registar mapiran samo u banci 1. S toga se u registar TRISA upisuje sadržaj akumulatora W isključivo posle selekcije memorijske banke 1.



Slika 3.2.7. *Direktni i indirektni načini adresiranja RAM memorije podataka.*

Indirektno adresiranje se izvodi pomoći IRP bita STATUS registra na MSB poziciji i registra specijalne namene FSR (*File Select Register*). MSB bitovi STATUS i FSR registara koriste se za selekciju memorijske banke, kao na slici 3.2.7. Memorijskoj lokaciji se pristupa preko virtuelnog INDF registra koji sadrži adresu na koju ukazuje FSR registar u svojstvu pokazivača. Drugim rečima, bilo koji pristup virtuelnom INDF registru zapravo je pristup registru u RAM memoriji (lokaciji) na adresi na koju ukazuje FSR registar.

Pretpostavimo na primer, da GPR registar na adresi 22h sadrži vrednost F1h. Upisujući vrednost 22h u FSR registar podesiće se pokazivač na adresu RAM memorije 22h. Čitanje INDF registra potom, daje za rezultat F1h što znači da je pročitan sadržaj GPR registra na adresi 22h bez njegovog direktnog adresiranja. S druge strane, upis sadržaja npr. 13h u INDF registar za rezultat će imati punjenje memorijske lokacije na adresi 22h, na koju ukazuje FSR registar, sadržajem 13h.

U tabeli 3.2.11. dat je primer programskog koda za brisanje uzastopnih 16 lokacija RAM memorije, počevši od adrese 20h do adrese 2Fh, indirektnim načinom adresiranja.

```
 ; ***************************************************************************
; *Deo programskog koda za brisanje prvih 16 GPR registara RAM memorije u Banci 0    *
 ; ***************************************************************************

MOVLW 0x20      ;Inicijalizacija pokazivača (FSR pointer)
MOVWF FSR       ;na početak RAM-a (adresa 0x20).
NEXT:           ;Labela programskog skoka.
CLRF INDF       ;Brisanje INDF registra je efektivno brisanje lokacije RAM-a na čiju
                ;adresu ukazuje FSR registar.
INCF FSR,F      ;inkrementiranje pokazivača (za brisanje lokacije na sledećoj adresi).
BTFSS FSR,4     ;ispitivanje bita 4 FSR pokazivača. Kada se njegov sadržaj
                ;inkrementira sa vrednosti 2Fh (00101111)2 na 30h (00110000)2 bit 4 se
```

```
                ;setuje. 30h je adresa sedamnaeste lokacije u odnosu na 20h, koju ne
                ;treba brisati.
GOTO  NEXT      ;ako bit 4 nije setovan briši sledeću lokaciju RAM-a
```

Tabela 3.2.11. *Primer koda za brisanje prvih 16 lokacija RAM memorije indirektnim načinom adresiranja.*

Sadržajem 20h puni se registar FSR tako da ukazuje na lokaciju RAM memorije sa adresom 20h. Potom se u programskoj petlji najpre vrši brisanje registra INDF, čime se efektivno briše lokacija na čiju adresu ukazuje sadržaj FSR registra, i odmah zatim inkrementira sadržaj FSR registra tako da ukazuje na sledeću memorijsku lokaciju. Naredna instrukcija testira stanje bita FSR registra na poziciji 4, čija će se nulta vrednost promeniti na jednicu po inkrementiranju sadržaja FSR sa 2Fh na 30h. U zavisnosti od stanja četvrtog bita FSR registra program grana na labelu NEXT, brišući narednu memorijsku lokaciju, ili preskače instrukciju GOTO NEXT nastavljajući sa daljim izvršavanjem.

CCS C kompajler takođe podržava direktni i indirektni pristup registrima RAM memorije. U tabeli 3.2.12. prikazane su instrukcije CCS C kompajlera kao primeri oba načina adresiranja. Prve tri instrukcije su primeri direktnog pristupa dok poslednje dve predstavljaju primere indirektnog pristupa na bazi indirekcionog operatora * (operatora posrednog pristupa).

```
 ; ********************************************************************************
; *Primeri instrukcija CCS C kompajlera za direktni i indirektni pristup RAM memoriji*
 ; ********************************************************************************

Set_tris_B(0);  //ugrađena f-ja za upis u direkcioni registar porta B (upisom nula
                //svih osam linija porta B konfigurišu se kao digitalni izlazi)
Output_B(0xF0); //ugrađena f-ja za upis podatka (0xF0) na 8-bitni port B
a=0x06;         //dodela vrednosti promenljivoj a (vrednost 0x06 predstavlja adresu
                //porta B)
Data=*a;        //sadržaj promenljive na adresi a dodeljuje se promenljivoj Data
                //što odgovara čitanju porta B i dodeli pročitane vrednosti
                //promenljivoj Data
*0x06=0xFF;     //promenljivoj na adresi 0x06 (adresa porta B) dodeljuje se sadržaj
                //FFh što odgovara upisu sadržaja FFh u portB.
```

Tabela 3.2.12. *Primeri instrukcija CCS C kompajlera za direktni i indirektni pristup RAM memoriji.*

3.3. IZBOR I KONFIGURISANJE TAKTNOG OSCILATORA CPU

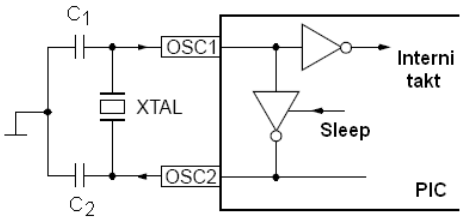
Za taktovanje CPU jedinice mikrokontrolera PIC16F87X i njegovih periferijskih modula predviđena su četiri radna režima kola oscilatora, koje korisnik može birati programiranjem dva bita konfiguracione reči mikrokontrolera FOSC1 i FOSC0. To su:

- LP (Low Power Crystal) za spoljašnje kvarc kristale rezonantne frekvencije do 200KHz, npr. satni kvarc kristal frekvencije rezonanse 32768Hz.

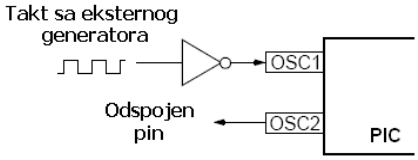
- XT (Crystal/Resonator) za spoljašnje kvarc kristale i keramičke rezonatore frekvencije rezonanse od 200KHz do 4MHz.
- HS (High Speed Crystal/Resonator) za spoljašnje kvarc kristale i keramičke rezonatore frekvencije rezonanse od 4MHz do 20MHz.
- RC (Resistor/Capacitor) za jeftine spoljašnje pasivne vremenske komponente R i C. (Neke serije PIC MCU kao što je PIC12C5XX poseduju integrisane RC komponente i ne raspolažu HS modom oscilatora).

Na slici 3.3.1. prikazana je konfiguracija i način priključenja spoljašnjih komponenata prva tri tipa kristalnih oscilatora LP, XT i HS. Ova konfiguracija obuhvata jedan, u MCU integrisani invertujući pojačavač, čiji izlaz prelazi u stanje visoke impedanse (odspojen izlaz) po izvršenju SLEEP instrukcije, i tri spoljašnje komponente - dva keramička kondenzatora i jedan kvarc kristal, kao na slici 3.3.1. Jedina razlika između prva tri radna režima je u pojačanju invertujućeg pojačavača. U LP modu pojačanje pojačavača je najmanje a potrošnja minimizirana zahvaljujući niskim radnim frekvencijama do oko 200KHz. HS mod se koristi za visoke radne frekvencije, do 20MHz sa najvećim pojačanjem pojačavača ali i potrošnjom kola. U ovom modu postižu se velike brzine izvršenja instrukcija. XT mod je predviđen za umerene radne frekvencije, do 4MHz, što podrazumeva umereno pojačanje pojačavača i potrošnju kola.

Kada je konfigurisan u jednom od prva tri moda, uređaj može biti taktovan i eksternim taktnim generatorom kao na slici 3.3.2. Takav način taktovanja može biti od koristi kada se želi sinhronizacija većeg broja MCU jednim taktnim generatorom. U tom slučaju eksterni oscilator treba priključiti na ulazni pin OSC1 mikrokontrolera, dok izlazna linija OSC2 postaje odspojena (plivajuća). Ova linija, iako plivajuća, treba biti spojena na zajedničku tačku nultog potencijala-masu preko otpornika u cilju redukcije šuma. Periodični Izlazni signal eksternog generatora takta (pravougaona povorka impulsa) mora biti takav da niski naponski nivo njegovih impulsa bude manji od $0.3V_{DD}$ a visoki veći od $0.7V_{DD}$, gde je V_{DD} napon napajanja mikrokontrolera. Jedan od tri moda, LP, XT ili HS, treba konfigurisati u skladu sa izabranom frekvencijom eksternog generatora.



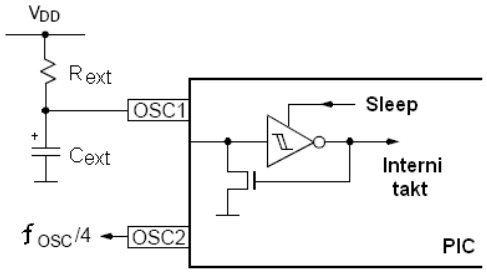
Slika 3.3.1. *Povezivanje spoljašnjih komponenata za konfigurisanje LP, XT ili HS oscilatora.*



Slika 3.3.2. *Taktovanje MCU eksternim generatorom taktnog signala u LP, XT ili HS modovima.*

RC mod je koristan za jeftine aplikacije gde striktni tajming i aktuelna brzina taktovanja nisu od krucijalnog značaja, slika 3.3.3. Brzina taktovanja (frekvencija oscilovanja taktnog generatora) je funkcija izabranih vrednosti otpornosti otpornika R i kapacitivnosti kondenzatora C koji se priključuju spolja, napona napajanja V_{DD} i radne temperature. Za seriju mikrokontrolera PIC16F87X preporučljive vrednosti spoljašnjih pasivnih komponenta su:

$$3k\Omega \le R_{EXT} \le 100k\Omega \text{ i } C_{EXT} > 20pF$$



Slika 3.3.3. *RC mod taktnog oscilatora sa izlazom sistemskog takta.*

Proizvođač u formi grafika i tabela nudi specifikacije R i C komponenta u funkciji frekvencije oscilovanja RC generatora takta. Korisnik, međutim, mora uzeti u obzir i varijacije frekvencije zbog tolerancija vrednosti spoljašnjih R i C komponenata.

U RC modu MCU generiše sistemski takt $f_{OSC}/4$, raspoloživ na pinu OSC2, koji može biti baferovan i primenjen kao sistemski takt za sinhronizaciju drugih komponenata ili mikrokontrolera.

Opseg napona napajanja PIC16F87X serije mikrokontrolera je od 2V do 5.5V, dok je tipična potrošnja struje manja od 0.6mA pri naponu napajanja 3V i frekvenciji taktovanja 4MHz, slika 3.3.4. Tipična potrošnja struje, međutim, pri istom naponu i frekvenciji taktovanja od 32KHz je samo 20μA.

Sve serije PIC mikrokontrolera poseduju tkzv. *sleep* funkcionalnost za prevođenje uređaja u režim rada sa ekstremno niskom potrošnjom, izvršavanjem instrukcije SLEEP. Kao što se sa slika 3.3.1. i 3.3.3. može videti, ovom instrukcijom se blokira kolo kristalnog ili RC oscilatora i time blokira dalje izvršavanje instrukcija. Kao rezultat, potrošnja struje opada na vrednost manju od 1μA koja predstavlja rezultantnu struju curenja kroz poluprovodnik.

Već je rečeno da je snaga disipacije MCU srazmerna frekvenciji taktovanja i data izrazom

$$P_D = C_T \, f_{CLK} \, V_{DD}^2$$

gde je f_{CLK} frekvencija taktnih impulsa, C_T ekvivalentno koncentrisano kapacitivno opterećenje MCU a V_{DD} napon napajanja MCU.

Buđenje mikrokontrolera iz *sleep* režima vrši se:

1. spoljašnjim resetovanjem MCU na $\overline{\text{MCLR}}$ liniji,
2. prethodno omogućenim spoljašnjim prekidima (RB0 linija ili četiri linije višeg nibla porta B) ili prethodno omogućenim unutrašnjim prekidima generisanih sa periferija,
3. izlaznim signalom prethodno omogućenog WDT tajmera.

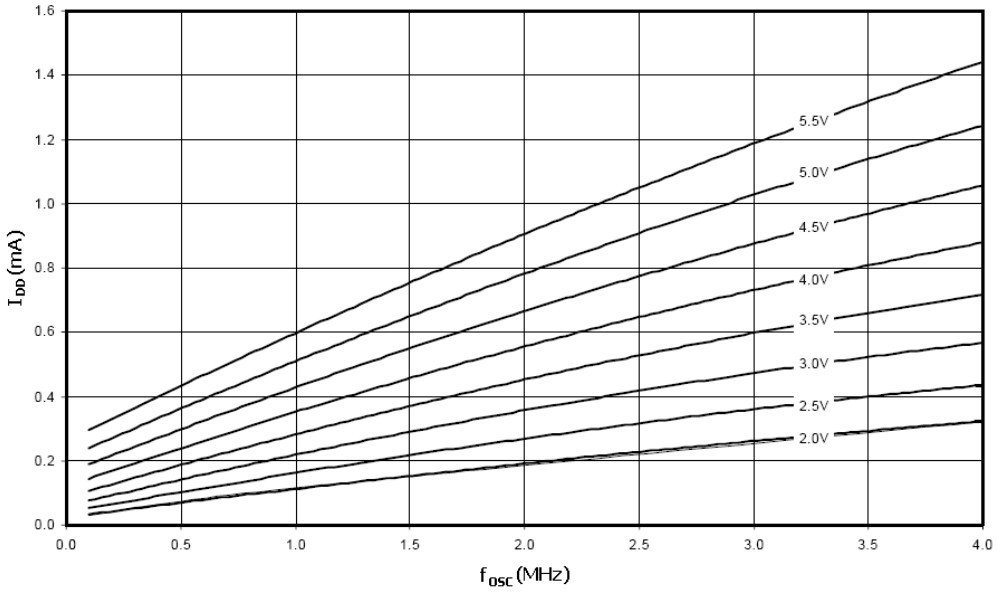
Sledeći periferijski prekidi mogu probuditi MCU iz sleep stanja ako su prethodno bili omogućeni i pod odgovarajućim okolnostima:

1. prekid sa PSP porta (paralelni slave port) pri čitanju ili upisu u port,
2. prekid sa tajmera 1 (TMR1) ako je konfigurisan u asinhronom brojačkom modu,
3. SSP (START/STOP) bit detekcije prekida,
4. SSP prijem ili predaja u slave modu (SPI/IIC),
5. prekid sa USART modula u sinhronom slave režimu rada,
6. prekid po kompletiranju operacije upisa u EEPROM memoriju,
7. prekid sa A/D modula ako je taktovan autonomnim RC taktnim oscilatorom,

Ostale periferije ne mogu generisati prekide zbog blokiranog taktnog oscilatora u sleep režimu MCU.

Buđenje MCU omogućenim prekidnim signalom vrši se na dva načina u zavisnosti od stanja GIE bita pre izvršenja SLEEP instrukcije:

a) normalnim nastavljanjem izvođenja prve instrukcije posle SLEEP instrukcije ako je pre izvođenja SLEEP instrukcije GIE bit bio resetovan
b) izvršenjem prve instrukcije koja sledi SLEEP instrukciju i odmah potom grananjem programa na prekidni vektor ako je GIE bit bio setovan.

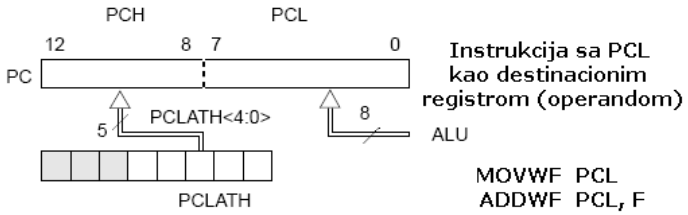


Slika 3.3.4. *Tipične krive potrošnje struje MCU PIC16F877 u XT režimu taktovanja pri različitim vrednostima napona napajanja.*

3.4. PROGRAMSKI BROJAČ (BROJAČ INSTRUKCIJA)

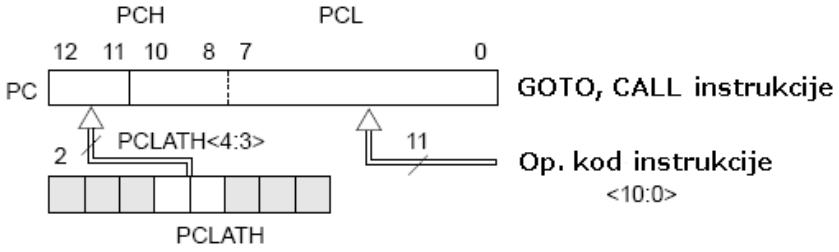
Programski brojač je 13-bitni registar podeljen na dva dela PCL i PCH. Niži bajt programskog brojača PCL lociran je u RAM memoriji podataka i mapiran u sve četiri memorijske banke radi bržeg pristupa (adrese 02h, 82h, 102h i 182h). PCL registar je R/W tipa, što znači da se može čitati ili se u njega može upisivati sadržaj. Viši petobitni deo programskog brojača PCH nije direktno pristupačan programeru što znači da se ne može direktno čitati niti se u njega direktno može upisivati sadržaj. Međutim, ovom delu PC pridružen je PCLATH registar u svojstvu bafera pa se u PCH deo programskog brojača može samo upisivati sadržaj i to posredno preko PCLATH registra. PCLATH registar se automatski briše pri bilo kojoj vrsti reseta mikrokontrolera. Najviša tri bita ovog registra nisu fizički implementirana i čitaju se kao nule. Na slici 3.4.1. prikazan je princip promene sadržaja programskog brojača PC pri upisu u PCL registar.

PCL registru se može pristupati programski kao i bilo kom drugom GPR ili SPR registru RAM memorije podataka. Na primer, ako je sadržaj akumulatora *n* tada instrukcija ADDWF PCL, F sabira aktuelni sadržaj PCL registra sa sadržajem akumulatora *n* i rezultat smešta u PCL registar, što će za posledicu imati skok napred za *n* mesta u odnosu na instrukciju ADDWF PCL, F.



Slika 3.4.1. *Direktna promena sadržaja 13-bitnog PC pri upisu u PCL registar.*

Uprkos činjenici da se instrukcijama za upis u PCL registar direktno menja stanje samo ovog osmobitnog registra, u isto vreme nižih pet bitova PCLATH<4:0> registra automatski se kopira u programeru nedostupan viši deo programskog brojača PCH. Kako se registar PCLATH automatski briše pri bilo kom resetu MCU to će instrukcija za upis u PCL efektivno adresirati prvih 2^8=256 reči programske memorije dok se novom instrukcijom ne izmeni sadržaj PCLATH registra. Prema tome, posebnu pažnju treba posvetiti promeni sadržaja PC instrukcijom upisa u PCL registar, specijalno u slučaju kada novi sadržaj PCL može prekoračiti maksimalnu osmobitnu vrednost.



Slika 3.4.2. *Indirektna promena sadržaja 13-bitnog PC pri izvršenju instrukcija programskog skoka.*

Instrukcije programskog skoka koje indirektno menjaju sadržaj PC kao što su CALL i GOTO, slika 3.4.2., takođe koriste deo sadržaja PCLATH registra. Naime, niži jedanaestobitni deo operacionog koda ovih instrukcija, koji predstavlja adresu programskog skoka, prenosi se na niže bitske pozicije programskog brojača PC<10:0> dok se dva najviša bita PC<12:11> formiraju prenosom sadržaja PCLATH<4:3> registra u isto vreme u dve najviše bitske pozicije programskog brojača PC<12:11>.

Budući da je adresa programskog skoka jedanaestobitna, znači da se instrukcijama skoka može doseći najviše 2^{11}=2K prostora programske memorije koliko iznosi veličina jedne od četiri stranice programske memorije. Za daleke programske skokove sa jedne na drugu stranicu programske memorije koriste se dva najviša bita PCLATH<4:3>. Prema tome, selekcija stranice programske memorije vrši se podešavanjem bitova <4:3> PCLATH registra što mora prethoditi npr., pozivu potprograma koji se nalazi na drugoj stranici programske memorije u odnosu na tekuću, ili bezuslovnom skoku. U tabeli 3.4.1. prikazane su kombinacione vrednosti bitova 4 i 3 PCLATH registra za izbor stranice programske memorije.

PCLATH<4:3>		Selektovana stranica programske memorije
0	0	Stranica 0
0	1	Stranica 1
1	0	Stranica 2
1	1	Stranica 3

Tabela 3.4.1. *Kombinacione vrednosti dva bita STATUS registra za izbor memorijske banke.*

U tabeli 3.4.2. je dat primer ispravnog programskog koda za poziv potprograma sa tekuće nulte stranice, koji je lociran na stranici jedan programske memorije.

```
     ORG 0x500            ;Stranica 0 programske memorije (opseg adresa 000h-7FFh)
     BCF PCLATH,4
     BSF PCLATH,3         ;Selekcija stranice 1 programske memorije(800h-FFFh)
     CALL SUB1_P1         ;Poziv potprograma na stranici 1 (800h-FFFh)
     .
     .                    ;pažnju obratiti na adrese sledećih instrukcija skoka
     .
     ORG 0x900            ;Stranica 1 (opseg adresa 800h-FFFh)
SUB1_P1:                  ;Pozvani potprogram na stranici 1 (800h-FFFh)
     .
     .
     .
     RETURN               ;Povratak na prvu instrukciju posle CALL SUB_P1 na stranici
                          ;0 (000h-7FFh)
```

Tabela 3.4.2. *Primer ispravnog programskog koda udaljenog poziva potprograma.*

Može se primetiti da se pre poziva potprograma SUB1_P1 resetovanjem, odnosno, setovanjem bitova 4 i 3 PCLATH registra, respektivno, najpre pravilno selektuje stranica jedan programske memorije a potom poziva potprogram.

U skladu sa tehnikom protočne obrade instrukcija u prvom instrukcijskom ciklusu izvršavanja dvociklusne instrukcije CALL SUB1_P1 programski brojač se inkrementira tako da ukazuje na sledeću instrukciju ispod instrukcije poziva potprograma, koja se u istom ciklusu i dobavlja. Osim dobavljanja sledeće instrukcije u nizu, u prvom ciklusu se inkrementirana vrednost PC upućuje na LIFO stek, pamteći na taj način povratnu adresu, a potom programski brojač puni adresom prve instrukcije potprograma. Dobavljena instrukcija ispod instrukcije poziva potprograma, međutim, ne predstavlja prvu instrukciju potprograma te se u sledećem instrukcijskom ciklusu neće izvršiti. Umesto nje izvršava se NOP instrukcija a u istom, drugom ciklusu dobavlja prva instrukcija pozvanog potprograma na koju ukazuje sadržaj PC.

Prema tome, u prvom od dva instrukcijska ciklusa dvociklusne instrukcije poziva potprograma najpre se pamti povratna adresa sledeće instrukcije u nizu. Potom se niži deo programskog brojača puni 11-bitnom adresom prve instrukcije potprograma (adresa je sadržana u operacionom kodu instrukcije poziva potprograma) a viši puni prenosom sadržaja PCLATH<4:3> registra u dve najviše bitske pozicije programskog brojača PC<12:11>. Time je pravilno selektovana i stranica programske memorije na kojoj je lociran potprogram.

Po povratku iz potprograma, sadržaj PCLATH registra nije neophodno menjati budući da se sačuvana povratna adresa u steku nalazi na stranici 0 programske memorije. Međutim, bilo koji sledeći poziv potprograma ili bezuslovni skok zahtevaće ponavljanje procedure selekcije memorijske stranice, odnosno, novi upis u PCLATH registar imajući u vidu njegov prethodni sadržaj kojim je selektovana stranica 1.

3.4.1. Čitanje tabele metodom izračunatog skoka

Izračunati programski skok se izvodi dodavanjem ofseta PCL registru programskog brojača (ADDWF PCL), dakle, instrukcijom koja kao destinacioni registar koristi PCL registar. Ova metoda je veoma pogodna za pristup podacima u programskoj memoriji, odnosno, za čitanje tabela zasnovanih na uzastopnom nizu RETLW K instrukcija (*RETurn with Literal k in W*), gde je K osmobitna tabelarna konstanta. Pri izvršenju instrukcije povratka osmobitna konstanta se upisuje u akumulator W. Kako je PCL registar osmobitni, maksimalna veličina tabele može biti 256 redova.

Tabela se realizuje kao potprogram. Prva instrukcija u tabeli zaračunava ofset (pomeraj) dodajući ga PCL registru. Sledstveno tome program grana na odgovarajuću RETLW K instrukciju u nizu takvih. U tabeli 3.4.3. dat je primer izvornog koda za principijelno čitanje tabele u programskoj memorji MCU.

```
.
.
MOVLW  offset   ;Učitaj ofset (pomeraj)u registar akumulatora w
CALL  Table     ;Poziv potprograma (Tabele)
.
.
.
Table:          ;Labela potprograma

ADDWF  PCL,F    ;Dodavanje ofseta sadržaju registra PCL i prepis rezultata u PCL
                ;Generisanje izračunatog programskog skoka
RETLW  'E'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom E u W registru
RETLW  'T'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom T u W registru
RETLW  'F'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom F u W registru
.
.
```

Tabela 3.4.3. *Principijelni način čitanja tabele u programskoj memoriji MCU zasnovane na uzastopnom nizu RETLW K instrukcija.*

Metod je naizgled jednostavan i ispravan, međutim, izvesne predostrožnosti moraju biti uzete u obzir kada se radi o tabeli lociranoj bilo gde u programskoj memoriji. Naime, PCLATH registar se posle reseta MCU automatski briše čime se automatski selektuje stranica nula programske memorije. Ako je tabela locirana na drugoj stranici programske memorije, poziv potprograma Table rezultiraće pogrešnim programskim skokom na nultu stranicu budući da sadržaj najviša dva bita PCLATH<4:3> registra nije izmenjen. Takođe, za pravilno adresiranje, posle svakog uzastopnog bloka od 256 reči programske memorije sadržaj PCLATH registra mora biti inkrementiran budući da se njegov sadržaj prenosi u viši bajt PC. Instrukcija dodavanja ofseta PCL registru ADDWF PCL, F neće izračunati vrednost veću od osmobitne u slučaju prekoračenja. Maksimalna veličina tabele ograničena je na 254 redova pre nego operacija sabiranja ADDWF PCL,F uzrokuje prekoračenje PCL registra i time grešku u adresiranju.

Na primer, kod programa u tabeli 3.4.4. uzrokovaće skok na pogrešnu adresu u slučaju da je vrednost ofseta veća od nule jer se adresa potprograma Table nalazi na kraju drugog bloka od 256 reči na nultoj stranici programske memorije. Posle dodavanja ofseta većeg od nule PCL registru, sledi automatski inkrement programskog brojača u cilju dobavljanja sledeće instrukcije, odnosno, prelaz na treći blok. Promenu bloka, međutim, ne prati i promena sadržaja PCLATH registra što će proizvesti pogrešan programski skok. To znači da bi u slučaju dodavanja ofseta vrednosti npr. jedan PCL registru, vrednost PCL bila FFh a 13- bitna adresa PC 2FFh zbog istovremenog prenosa sadržaja PCLATH registra u viši deo programskog brojača. U istom

instrukcijskom ciklusu sadržaj PC se automatski inkrementira a novi sadržaj PC će biti 200h umesto očekivanog 300h. Izračunati programski skok na adresu 200h je, međutim, pogrešan i neće proizvesti očekivani rezultat.

```
ORG  0x00A           ;Program lociran u nultom bloku programske memorije od 256 reči
MOVLW  HIGH Table
MOVWF  PCLATH        ;Viša petobitna adresa (0x03) tabele u PCLATH
MOVF  offset,W       ;Učitaj ofset (pomeraj)u registar akumulatora w
CALL  Table          ;Poziv potprograma (Tabele)
.
.
.
ORG  0x2FE           ;Tabela locirana na kraju drugog bloka programske memorije od 256
                     ;reči
Table:               ;Labela potprograma

ADDWF  PCL,F         ;Dodavanje ofseta sadržaju registra PCL i prepis rezultata u PCL

RETLW  'E'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom E u W registru
RETLW  'T'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom T u W registru
RETLW  'F'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom F u W registru
.
.
```

Tabela 3.4.4. *Neispravan kod programa za čitanje tabele u programskoj memoriji MCU.*

Jedan od načina za izbegavanje greške u adresiranju je korišćenje direktive ORG (*ORiGin*) za specifikaciju apsolutne adrese potprograma i upis više adrese potprograma Table u PCLATH registar pre njegovog poziva. Direktivom ORG ukazuje se asembleru gde da locira početak potprograma ili bilo kog programskog modula. Microchip-ov asembler poseduje direktive HIGH i LOW koje mogu biti korišćene za razdvajanje 13-bitne adrese programskog brojača na viši i niži deo, respektivno, što olakšava adresnu aritmetiku. Međutim, gornji primer pokazuje da predloženo rešenje nije u potpunosti tačno, za slučaj kada je potprogram lociran pri kraju jednog bloka programske memorije od 256 reči.

Univerzalno rešenje ispravnog koda za čitanje tabele, ma gde bila locirana u prostoru programske memorije MCU, prikazano je u tabeli 3.4.5.

```
.
ORG  0x00A           ;Program lociran na stranici 0 programske memorije

MOVLW  HIGH Table+1
MOVWF  PCLATH        ;Viši deo simboličke adrese Table, koja je prethodno
                     ;inkrementirana, prenosi se u PCLATH registar

MOVLW  LOW Table+1   ;Niži deo inkrementirane simboličke adrese Table u W
ADDWF  offset,W      ;Provera, ima li prekoračenja bloka od 256 reči prog. memorije
BTFSC  STATUS,C
INCF  PCLATH,F       ;Blok od 256 reči prekoračen sabiranjem ofseta i niže adrese.

MOVF  offset,W       ;Učitavanje ofseta u W registar

CALL  Table          ;Poziv potprograma na simboličkoj adresi Table
CLRF  PCLATH         ;Brisanje PCLATH (nulta stranica) po povratku iz tabele
.
.
ORG  0x8FF           ;Prva instrukcija potprograma Table locirana na stranici 1
                     ;programske memorije na adresi 0x8FF
Table:               ;Simbolička adresa potprograma

ADDWF  PCL,F         ;Sabiranje ofseta i sadržaja PCL registra
```

```
RETLW  'F'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom F u W registru
RETLW  'I'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom I u W registru
RETLW  'T'      ;Povratak na instrukciju ispod CALL sa ASCII karakterom T u W registru
.
.
```

Tabela 3.4.5. *Univerzalni kod programa za čitanje tabele u programskoj memoriji MCU.*

CCS C kompajler ima podršku za smeštanje bilo koje strukture podataka u programsku memoriju MCU kao niza konstanti. Na primer, za smeštanje deset konstanti tipa *byte* ili tipa *char* u programsku memoriju koristi se sledeća sintaksa:

```
CONST INT TABLE[10]={9,8,7,6,5,4,3,2,1,0};
CONST CHAR TABLE[10]={'A','B','C','D','E','F','G','H','I','J'};
```

dok se za pristup tabeli koristi:

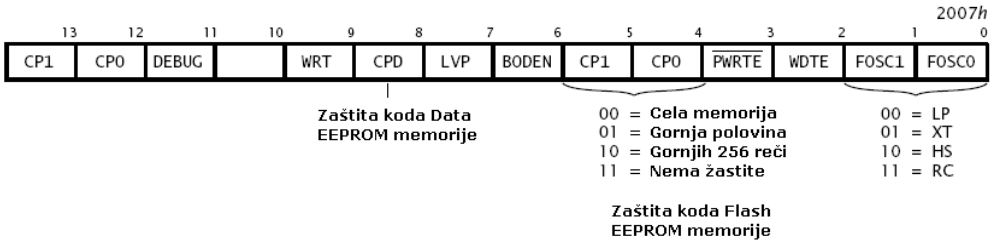
```
x = TABLE [i];
ili npr.
x = TABLE [5];
```

3.5. KONFIGURACIONA REČ MCU PIC16F877

Prostor programske Flash EEPROM memorije izvan korisničkog prostora, u opsegu adresa od 2000h do 3FFFh, odnosi se na prostor namenjen specijalnom testu/konfiguraciji programske memorije i dostupan je samo u toku eksternog programiranja MCU uz pomoć programatora. Ovaj prostor programske memorije nije vidljiv u toku normalnog izvršenja instrukcija, međutim, u režimu programiranja mikrokontrolera programator uređaja može pristupiti odgovarajućim privatnim lokacijama programske memorije. Memorijske lokacije na adresama od 2000h do 2003h označene su kao ID lokacije. Korisnik u ovom adresnom prostoru može čuvati npr. čeksumu ili identifikacione brojeve. Preporučuje se korišćenje samo četiri najmanje značajna bita ID lokacija. PIC16F877 MCU u okviru pomenutog prostora programske memorije, na adresi 2007h, poseduje rezervisanu tajnu lokaciju koja predstavlja tkzv. konfiguracionu reč MCU.

Programiranje svakog bita konfiguracione reči mikrokontrolera će osigurati da u normalnom režimu rada MCU taktni oscilator i drugi resursi budu konfigurisani na odgovarajući način. Na slici 3.5.1. ilustrovan je raspored konfiguracionih bitova konfiguracione reči MCU PIC16F877.

Neprogramirano, preset stanje konfiguracione reči je 3FFFh što znači da će nekonfigurisan MCU PIC16F877 biti u RC modu oscilatora sa omogućenim sigurnosnim Watchdog tajmerom (WDTE), onemogućenim Power-Up tajmerom (PWRTE), bez zaštite koda unutar programske memorije (CP1 CP0), sa omogućenim Brown-Out kolom za resetovanje MCU (BODEN) i omogućenim programiranjem pod niskim naponom na liniji RB3/PGM (LVP), bez zaštite koda u Data EEPROM memoriji (CPD) i sa mogućnošću upisa u nezaštićen prostor programske memorije (WRT), kao i sa onemogućenom debager funkcionalnošću na linijama RB6 i RB7 MCU (DEBUG).

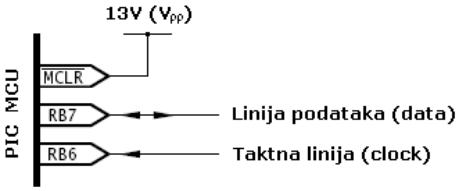


Slika 3.5.1. *Konfiguraciona reč PIC16F877 MCU.*

Udvojeni par konfiguracionih bitova CP1 i CP0, za izbor odgovarajuće zaštite programske memorije, na bitskim pozicijama (4, 5) i (12, 13) mora imati iste vrednosti da bi se postigla odgovarajuća zaštita. Omogućenje Brown-Out kola za resetovanje MCU pomoću konfiguracionog bita BODEN automatski omogućuje Power-Up tajmer bez obzira na stanje odgovarajućeg konfiguracionog bita PWRTE konfiguracione reči.

Veliki broj programatora MCU dopušta operatoru mogućnost direktnog postavljanja konfiguracionih bitova iz menija odgovarajućeg softvera za podršku hardveru programatora. Slika 3.5.2. prikazuje način povezivanja odgovarajućih linija mikrokontrolera u režimu programiranja/verifikacije. Liniju MCLR (*Master CLeaR*) u ovom režimu potrebno je dovesti na visoki napon od 13V-15V što MCU prebacuje u režim programiranja/verifikacije a potom se linije sinhrone serijske komunikacione magistrale *Data* i *Clock* mogu koristiti za ICSP (*In Circuit Serial Programming*) programiranje. Već je rečeno da u ovom režimu programator može pristupati korisničkom adresnom prostoru programske memorije ali i njenim privatnim lokacijama koje u normalnom režimu rada MCU (izvršenje korisničkog programa) nisu vidljive.

Uprkos mogućnosti direktnog postavljanja bitova konfiguracione reči, preporučuje se ugradnja izabrane konfiguracione reči u izvorni aplikacioni kod primenom asemblerske direktive za podatke __CONFIG kako bi se u vreme programiranja MCU na lokaciji sa adresom 2007h izvršilo automatsko postavljanje konfiguracionih bitova konfiguracione reči u skladu sa izabranom.



Slika 3.5.2. *Povezivanje odgovarajućih linija PIC16F877 MCU u režimu programiranja/verifikacije.*

Primer korišćenja direktive za podatke __CONFIG sa simboličkim imenima konfiguracionih bitova na odgovarajućim bitskim pozicijama konfiguracione reči dat je u nastavku teksta. Simbolička imena konfiguracionih bitova kojima su pridružene odgovarajuće adrese, odnosno, bitske pozicije u konfiguracionoj reči definisana su u uključnoj (*include*) datoteci P16F877.inc koja se kontrolnom direktivom #INCLUDE priključuje glavnom programu i kopira na mestu poziva.

```
    __CONFIG    _CP_OFF & _WDT_ON & _BODEN_ON & _PWRTE_ON & _HS_OSC & _WRT_ENABLE_ON &
_LVP_OFF & _DEBUG_OFF & _CPD_ON
```

ili u binarnom obliku

```
    __CONFIG    b'11111001110110'
```

Prvi primer sa simboličkim imenima konfiguracionih bitova je povoljniji sa stanovišta prenosivosti programa budući da različiti procesori mogu imati različiti aranžman bitova konfiguracione reči. U drugom binarnom obliku, promena uključne *.inc datoteke, odnosno procesora, može proizvesti potpuno pogrešnu konfiguracionu reč s obzirom na moguć drugačiji redosled konfiguracionih bitova. Ukoliko se neki od konfiguracionih bitova izostave u direktivi __CONFIG njihove će vrednosti posle programiranja MCU odgovarati neprogramiranim preset vrednostima.

CCS C kompajler poseduje vrlo sličan mehanizam programiranja svakog bita konfiguracione reči, koji koristi direktivu #FUSES na vrhu izvornog programa posle direktive #INCLUDE. Sledeći primer pokazuje primenu ove direktive za konfigurisanje MCU, koji odgovara gornjim asemblerskim direktivama __CONFIG.

```
#include <16F877.h>
#fuses NOPROTECT,WDT,BROWNOUT,PUT,HS,WRT,NOLVP,NODEBUG,CPD
```

U tabeli 3.5.1. je prikazan uticaj stanja konfiguracionih bitova konfiguracione reči na odgovarajuće resurse MCU.

bit 13-12	**CP1:CP0:** Zaštita koda programske FLASH memorije
bit 5-4	11 = Bez zaštite
	10 = 1F00h to 1FFFh zaštićeno 256 reči najvišeg adresnog prostora
	01 = 1000h to 1FFFh zaštićena gornja polovina adresnog prostora
	00 = 0000h to 1FFFh zaštićen ceo adresni prostor
bit 11	**DEBUG:** ICD (In-Circuit Debugger) Mod debagovanja programa
	1 = Debager onemogućen, RB6 i RB7 linije su I/O linije opšte namene
	0 = Debager omogućen, RB6 i RB7 linije koristi kolo debagera.
bit 10	**Neimplementiran bit:** Čita se kao '1'
bit 9	**WRT:** Upis u programsku FLASH memoriju
	1 = Omogućen interni upis u nezaštićeni prostor programske memorije
	0 = Onemogućen interni upis u nezaštićeni prostor programske memorije
bit 8	**CPD:** Zaštita koda Data EEPROM memorije
	1 = Bez zaštite
	0 = Zaštićen kod u Data EEPROM
bit 7	**LVP**: ICSP (In-Circuit Serial Programming) programiranje MCU pod niskim naponom
	1 = RB3/PGM linija ima PGM funkciju, omogućeno LV programiranje MCU
	0 = RB3 linija je digitalna I/O linija, HV na MCLR liniji koristi se za programiranje MCU
bit 6	**BODEN**: Brown-out kolo za resetovanje MCU
	1 = BOR omogućen
	0 = BOR onemogućen
bit 3	$\overline{\textbf{PWRTE}}$: Power-up Tajmer
	1 = PWRT onemogućen
	0 = PWRT omogućen
bit 2	**WDTE**: Watchdog Tajmer
	1 = WDT omogućen
	0 = WDT onemogućen
bit 1-0	**FOSC1:FOSC0**: Izbor oscilatora
	11 = RC oscilator

	10 = HS oscilator
	01 = XT oscilator
	00 = LP oscilator

Tabela 3.5.1. *Bitovi konfiguracione reči MCU PIC16F877.*

Poglavlje 4

INTEGRISANI SISTEMI ZA RESETOVANJE MCU

Za pouzdani početak izvršavanja aplikacionog koda upisanog u programsku memoriju MCU, mikrokontroler mora na pravilan način preći iz stanja odsustva napona napajanja V_{DD} u novo stanje po njegovom priključenju. Zbog konačne vrednosti unutrašnje otpornosti izvora napajanja, ulazne kapacitivnosti MCU na linijama napajanja kao i parazitskih kapacitivnosti metalnih veza linija napajanja, nominalna vrednost napona napajanja ne može biti dostignuta trenutno već nakon prelaznog režima čije trajanje je uglavnom u funkciji vrste izvora za napajanje MCU. Lokalni kristalni oscilator MCU takođe karakteriše prelaznim režim u kome ni amplituda ni frekvencija oscilovanja nisu konstantne. Naime, poznato je da je u kolima oscilatora sa pozitivnom povratnom spregom šum poluprovodničkih komponenata odgovoran za započinjanje oscilacija. Zbog jake pozitivne reakcije u kolu oscilatora, po priključenju napajanja amplituda oscilacija raste sa vremenom do uspostavljanja stacionarne vrednosti. U toku kolebanja amplitude oscilovanja i frekvencija, odnosno, perioda ponavljanja oscilacija je promenljiva. Posle prelaznog režima oscilator ulazi u stacionarno stanje oscilovanja sa stabilnom amplitudom i frekvencijom oscilovanja. Početak izvršavanja koda u vreme trajanja opisanih prelaznih režima predstavljao bi nekorektan radni režim MCU imajući u vidu stalne promene naponskih nivoa na I/O linijama, kao i nestabilnost trajanja instrukcijskog ciklusa, odnosno, tajminga.

U objektivnim okolnostima tokom normalnog izvršavanja koda moguće su tranzijentne pojave padova napona na liniji napajanja ispod dopuštene margine, što ugrožava pozdan rad aplikacije.

U slučaju nepredvidljivog zaglavljivanja programa koji se izvršava, usled npr. trenutnih elektromagnetskih-induktivnih smetnji ili logički loše projektovanog programa, poželjno je osigurati autonomni mehanizam samoresetovanja MCU koji će restartovati aplikaciju pod drugačijim, izmenjenim okolnostima.

Za sinhronizaciju više MCU u paralelnom multiprocesorskom režimu rada, postojanje eksternog reset ulaza značilo bi mogućnost istovremenog starta više CPU i mogućnost manuelnog resetovanja uređaja.

PIC16F877 MCU raspolaže ugrađenim mehanizmima namenjenim povećanju pouzdanosti uređaja, minimizaciji cene razvoja aplikacije eliminisanjem eksternih komponenata, kao i mehanizmima za smanjenje potrošnje, izbor oscilatora i slično što predstavlja specifične karakteristike MCU.

4.1. VRSTE RESETA MCU PIC16F877

Mikrokontroler PIC16F877 poseduje šest ugrađenih zasebnih vrsta reseta CPU jedinice kao što su:

- $\overline{\text{MCLR}}$ reset u toku normalnog režima rada MCU – priključenjem linije $\overline{\text{MCLR}}$ na niski naponski nivo u toku normalnog izvršavanja instrukcija.
- $\overline{\text{MCLR}}$ reset u toku standby režima – priključenjem linije $\overline{\text{MCLR}}$ na niski naponski nivo u toku standby (sleep) režima MCU.
- Power-on Reset (POR) – reset po uključenju napona napajanja MCU - kada napon napajanja dostigne naponski prag između 1.2V i 1.7V).
- WDT reset – resetovanje kola izlazom sigurnosnog WatchDog tajmera zbog isticanja time-out perioda (tipično 17ms) u normalnom režimu rada MCU.
- WDT *Wake-up* reset (buđenje) - resetovanje kola izlazom sigurnosnog WatchDog tajmera zbog isticanja time-out perioda (tipično 17ms) u standby režimu rada MCU.
- Brown-Out Reset (BOR) - reset usled pada napona napajanja ispod dopuštene margine u trajanju dužem od 100μs.

Resetovanje MCU nema uticaja na status jednog malog broja registara specijalne namene. Njihov status posle POR reseta ili nije unapred poznat ili ostaje nepromenjen posle bilo koje druge vrste reseta u odnosu na status pre resetovanja. Veći broj drugih registara specijalne namene, međutim, postavlja se u predefinisano '*reset*' stanje posle pojave jedne od nabrojanih vrsta reseta, izuzev WDT *Wake-up* reseta. Resetovanje izlazom sigurnosnog Watchdog tajmera kada je MCU u sleep režimu predstavlja tkzv. 'buđenje' MCU i može se tretirati kao nastavak normalnog režima rada posle vremena provedenog u režimu niske potrošnje.

4.1.1. $\overline{\text{MCLR}}$ reset u toku normalnog rada MCU i u sleep stanju

Svi PIC mikrokontroleri poseduju $\overline{\text{MCLR}}$ (Master CLeaR) ulaz koji se u sprezi sa spoljašnjim prekidačem može koristiti za manuelno eksterno resetovanje uređaja, slika 3.1.1. Ako je napon na $\overline{\text{MCLR}}$ liniji $V_{MCLR} \leq 0.2V_{DD}$ (niži prag prebacivanja CMOS invertora) za vreme duže od 100ns, doći će do resetovanja MCU. Vrednost pull-up otpornika od 33KΩ je maksimalno dopuštena vrednost otpornosti za koju napon na $\overline{\text{MCLR}}$ liniji neće opasti ispod praga logičke jedinice od

$0.85V_{DD}$ (viši prag prebacivanja CMOS invertora) za slučaj otvorenog prekidača, usled pada napona na otporniku uzrokovanog strujom curenja kroz ovaj pin. Otpornik od 100Ω služi za ograničenje struje kroz unutrašnju diodu za zaštitu od negativnih prenapona na $\overline{\text{MCLR}}$ liniji usled superpozicije šumova. Za napon na $\overline{\text{MCLR}}$ liniji $V_{MCLR} \geq 0.85V_{DD}$ CPU započinje sa normalnim izvršavanjem instrukcija startujući od prve instrukcije na adresi reset vektora 0x00h. Nulti sadržaj programskog brojača koji ukazuje na prvu instrukciju, podrazumeva automatsku selekciju nulte stranice programske memorije dok se nulta memorijska banka memorije podataka takođe selektuje automatskim resetovanjem bitova za izbor banke statusnog registra RP1 i RP0, vidi tabelu 4.1.1.

Ako je eksterno resetovanje MCU izvršeno u toku normalnog radnog režima, tada ova vrsta reseta ne menja stanje bitova statusnog registra $\overline{\text{TO}}$ i $\overline{\text{PD}}$. Dakle, njihova stanja ostaju ista kao pre resetovanja. Sadržaj statusnog registra ALU prikazan je u tabeli 4.1.1.

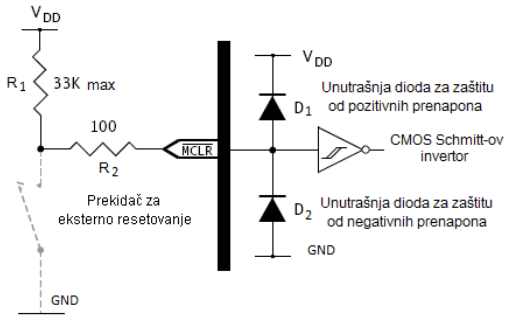
STATUS REGISTAR

	R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
	IRP	**RP1**	**RP0**	**$\overline{\text{TO}}$**	**$\overline{\text{PD}}$**	**Z**	**DC**	**C**
	bit 7							bit 0
bit 7	**IRP:** Bit za selekciju memorijske banke, korišćen za indirektno adresiranje. 1 = Banka 2, 3 (100h - 1FFh). 0 = Banka 0, 1 (00h - FFh).							
bit 6-5	**RP1:RP0**: Bitovi za selekciju memorijske banke, korišćeni za direktno adresiranje. 11 = Banka 3 (180h - 1FFh). 10 = Banka 2 (100h - 17Fh). 01 = Banka 1 (80h - FFh). 00 = Banka 0 (00h - 7Fh). **Komentar:** Svaka banka poseduje 128 osmobitnih registara.							
bit 4	**TO**: Tajm-aut bit. 1 = Posle power-up, `CLRWDT` instrukcije ili `SLEEP` instrukcije. 0 = Watchdod time-out period istekao.							
bit 3	**PD**: Power-down bit. 1 = Posle power-up ili posle `CLRWDT` instrukcije. 0 = Posle egzekucije `SLEEP` instrukcije.							
bit 2	**Z**: Zero bit. 1 = Rezultat aritmetičke ili logičke operacije je nula. 0 = Rezultat aritmetičke ili logičke operacije različit od nule.							
bit 1	**DC**: Digit carry/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instrukcije). 1 = Prenos/pozajmica sa nižeg na viši nibl (sa bitske pozicije 3 na 4) za BCD cifre. 0 = Nema prenosa/pozajmice. **Komentar:** (za operaciju oduzimanja bit pozajmice (borrow) je komplementiran).							
bit 0	**C**: Carry/borrow bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instrukcije). 1 = Prenos/pozajmica sa MSB bitske pozicije (prekoračenje). 0 = Nema prenosa/pozajmice. **Komentar:** Za operaciju oduzimanja bit pozajmice (borrow) ima vrednost 1 kada je rezultat operacije veći ili jednak nuli a vrednost 0 kada je rezultat negativan. Instrukcije rotiranja operanda ulevo ili udesno (RLF, RRF) koriste bit Carry/borrow kao ulazno/izlazni.							

Tabela 4.1.1. *Sadržaj statusnog registra ALU MCU PIC16F877.*

$\overline{\text{MCLR}}$ linija se može koristiti i za eksterno resetovanje MCU u režimu niske potrošnje (sleep stanje). Bitovi statusnog registra $\overline{\text{TO}}$ i $\overline{\text{PD}}$ postavljaju se tada u stanje 1 0, respektivno, pa se njihovim ispitivanjem može odrediti priroda reset signala. Ova vrsta reseta koristi se za tkzv.

'buđenje' MCU, zato što posle promene stanja na $\overline{\text{MCLR}}$ liniji sa logičke nule na jedinicu mikrokontroler započinje sa izvršenjem prve instrukcije na adresi reset vektora 0x00h.



Slika 4.1.1. *Manuelno spoljašnje resetovanje MCU na* $\overline{\text{MCLR}}$ *liniji.*

Kada sistem ne zahteva manuelni reset, $\overline{\text{MCLR}}$ linija može biti direktno povezana sa linijom napajanja V_{DD}. Kod 8-pinskih PIC MCU, kao što je PIC12C5XX familija, $\overline{\text{MCLR}}$ linija može biti konfigurisana i korišćena čak i kao I/O linija opšte namene, programiranjem konfiguracione reči MCU, zbog malog broja raspoloživih I/O linija.

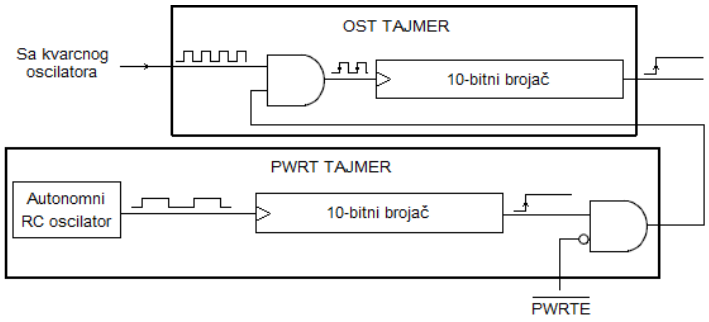
4.1.2. POR (Power-up Reset) reset

Svi PIC mikrokontroleri poseduju interni *Power-up* reset mehanizam za automatsku detekciju spremnosti procesora da izvrši prvu instrukciju programa, posle priključenja napona napajanja. Za ilustraciju ovog mehanizma razmotrimo donekle idealizovanu situaciju prikazanu na slici 4.1.3. po priključenju napona napajanja MCU u trenutku t_0. Ako je početna brzina promene napona napajanja $dV_{DD}/dt \geq 50V/s$, tada se pri naponskom pragu između 1.2V-1.7V generiše interni reset signal POR (*Power-on Reset*) koji inicira sledeće sekvence događaja:

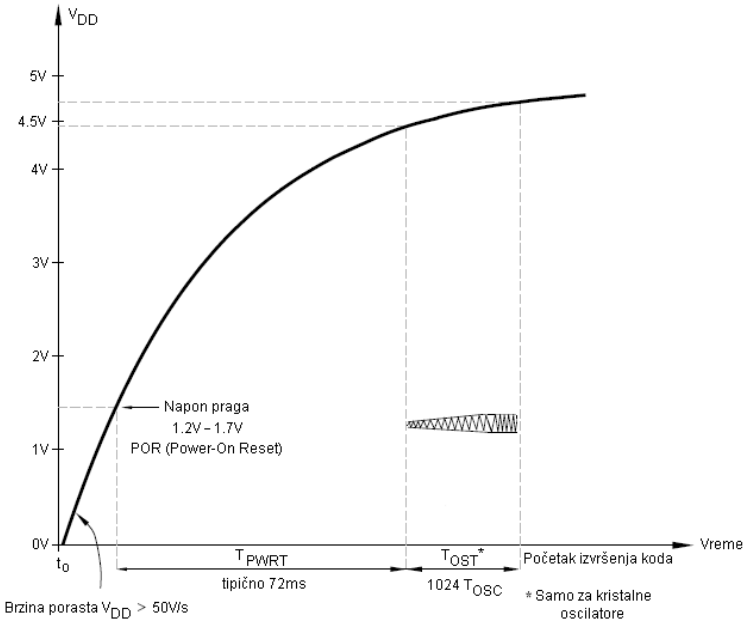
1. Generisanje fiksnog perioda *Power-up* tajmera (PWRT) T_{PWRT}, nominalnog trajanja 72ms, u toku koga je MCU resetovan. *Power-up* tajmer poseduje autonomni RC oscilator kojim se taktuje njegov 10-bitni brojač do trenutka prekoračenja punog ciklusa brojanja od 2^{10}= 1024 taktnih intervala, što tipično iznosi 72ms, vidi sliku 4.1.2. Zadatak ovog tajmera je da za vreme T_{PWRT} MCU drži u stanju reset dok napon napajanja ne dostigne prihvatljiv nivo. Ovo fiksno kašnjenje biće omogućeno ako je konfiguracioni bit $\overline{\text{PWRTE}}$ konfiguracione reči MCU resetovan. Period PWRT tajmera T_{PWRT} varira od čipa do čipa i u funkciji je temperature i napona napajanja.

2. Po isteku trajanja perioda T_{PWRT}, *Power-up* tajmer inicira drugi tajmer tkzv. *Oscillator Start-up* tajmer (OST) u čijem sastavu je takođe 10-bitni brojač koga, međutim, taktuje glavni CPU kristalni oscilator, slika 4.1.2. Period punog ciklusa brojanja ovog tajmera u direktnoj je zavisnosti od rezonantne frekvencije kvarc kristala i iznosi $T_{OST}=2^{10}$ $T_{OSC}=1024T_{OSC}$. Npr. ako je rezonantna frekvencija kvarc kristala f_{OSC}=10MHz, tada je $T_{OSC}=1/f_{OSC}$=0.1μs dok je period OST tajmera T_{OST}=102.4μs. Period koji generiše OST tajmer, za koje vreme je MCU u reset stanju, osiguraće pouzdano i stabilno funkcionisanje taktnog oscilatora pre početka obrade programa. OST tajmer, koji generiše kašnjenje T_{OST}, implementiran je samo za konfiguracije kvarcnih oscilatora u LP, XT i HS modovima ali ne i za RC mod oscilatora.

OST tajmer se aktivira i u slučaju buđenja MCU iz sleep stanja (isključen glavni oscilator) kako bi se omogućio restart glavnog oscilatora i njegovo normalno funkcionisanje pre nastavka obrade programa.



Slika 4.1.2. *Blok šema PWRT i OST tajmera u sprezi.*



Slika 4.1.3. *Sekvence događaja po priključenju MCU na napon napajanja V_{DD}, pre početka izvršenja koda.*

3. Kao i u slučaju eksternog $\overline{\text{MCLR}}$ reseta, izvršenje koda započinje sa adrese reset vektora 0x00h odmah nakon isteka vremenskog perioda T_{PU} koji predstavlja zbir vremena $T_{PU}=T_{PWRT}+T_{OST}=72ms+1024T_{OSC}$, kao na slici 4.1.3. Međutim, za razliku od eksternog reseta, POR reset postavlja oba bita statusnog registra $\overline{\text{TO}}$ i $\overline{\text{PD}}$ u neaktivno stanje, 1 1, respektivno.

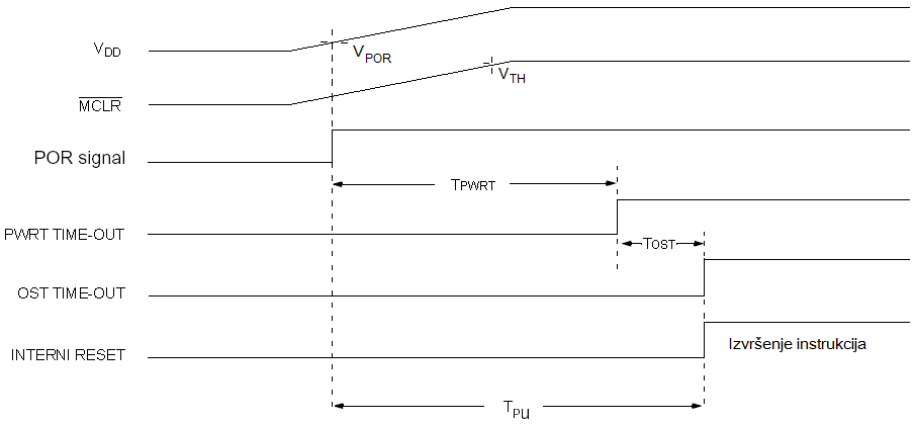
Konfiguracija oscilatora	Power-up		Buđenje iz sleep stanja
	$\overline{\text{PWRTE}}$ =0	$\overline{\text{PWRTE}}$ =1	
LP, XT, HS	72ms+1024T_{OSC}	1024T_{OSC}	1024T_{OSC}
RC	72ms	-	-

Tabela 4.1.2. *MCU Power-up tajming u različitim situacijama.*

U tabeli 4.1.2. prikazane su različite situacije reset tajminga pri uspostavljanju napona napajanja (Power-up) i buđenju MCU iz sleep stanja (*Wake-up*) kada je PWRT tajmer omogućen/onemogućen i za različite konfiguracije taktnog oscilatora. Kao što se može videti, jedino u RC modu taktnog oscilatora nema kašnjenja od $1024T_{OSC}$ neophodnog za stabilizaciju oscilatora, budući da RC oscilator ulazi u stacionarni režim odmah nakon prve periode oscilovanja. U preostala tri moda kristalnog oscilatora, kašnjenje koje unosi OST tajmer (za koje vreme je MCU resetovan) generiše se po automatizmu u cilju stabilizacije kola kvarcnog oscilatora i u toku uspostavljanja napona napajanja i pri buđenju MCU.

Omogućeni PWRT tajmer generisaće dodatno kašnjenje od 72ms samo u toku uspostavljanja napona napajanja i to za bilo koju konfiguraciju oscilatora ali ne i prilikom buđenja MCU. Naime, odlazak MCU u sleep stanje je prelaz iz normalnog radnog režima MCU, sa već uspostavljenom nominalnom vrednošću napona napajanja V_{DD}, u stanje mirovanja. S tim u vezi, prilikom buđenja MCU ne postoji potreba za novim startovanjem PWRT tajmera. Međutim, ovaj zaključak ne važi za OST tajmer. Budući da odlazak u sleep stanje znači isključenje kola kristalnog oscilatora to se pri buđenju, odnosno, novom uključenju oscilatora mora izbeći prelazni režim (period nestabilnog rada oscilatora), trajanja oko $1024T_{OSC}$, držanjem MCU u stanju reset do isteka ovog vremenskog perioda. Ovo se postiže automatskim startovanjem OST tajmera.

Na slici 4.1.4. ilustrovana je *Power-up* sekvenca internih signala MCU kada početna brzina porasta napona napajanja zadovoljava uslov $dV_{DD}/dt \geq 50V/s$ i kada je $\overline{\text{MCLR}}$ linija povezana sa linijom napajanja V_{DD}, kao na slici 4.1.1. Kako se na otpornicima R_1 i R_2 pad napona usled struje curenja može zanemariti, za slučaj otvorenog prekidača, napon na liniji napajanja V_{DD} jednak je naponu na $\overline{\text{MCLR}}$ liniji.



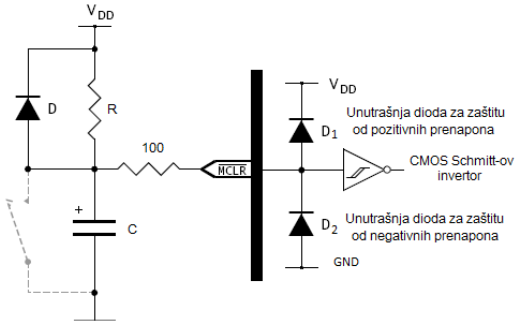
Slika 4.1.4. *Ilustracija Power-up sekvence za $dV_{DD}/dt \geq 50V/s$, kada je $V_{MCLR}=V_{DD}$.*

Na slici 4.1.4 se može uočiti trenutak kada napon napajanja dostiže naponski prag V_{POR} (1.2V-1.7V), odnosno, trenutak inicijalizacije PWRT tajmera sa fiksnom zadrškom od približno 72ms posle čega sledi inicijalizacija OST tajmera. Po isteku vremenske zadrške koju generiše OST tajmer, $T_{OST}=1024T_{OSC}$, MCU započinje regularno izvršavanje koda. Za vreme reset stanja MCU, trajanja $T_{PU}=T_{PWRT}+T_{OST}$, napon napajanja dostiže nominalnu radnu vrednost a kristalni oscilator ulazi u stacionarni režim oscilovanja.

Prethodna razmatranja odnose se na slučajeve kada početna brzina porasta napona napajanja nije manja od 50V/s. Međutim, ako je uspostavljanje napona napajanja V_{DD} sporo, sa početnom brzinom porasta manjom od 50V/s, opisani *Power-up* mehanizam neće biti u stanju da generiše odgovarajući tajming (POR signal kojim se trigeruje PWRT tajmer). Ako se i dogodi da POR signal trigeruje PWRT tajmer ukupno vreme zadrške MCU u reset stanju

$T_{PU}=T_{PWRT}+T_{OST}=72ms+1024T_{OSC}$ će biti nedovoljno za dostizanje nominalne radne vrednosti napona napajanja i stabilizaciju kristalnog oscilatora. Nominalna vrednost napona V_{DD} je generalno, za verzije uređaja sa napajanjem od 5V, oko 4V kada kristalni oscilator MCU nije u HS (*High Speed*) modu i oko 4.5V za MCU sa kristalnim oscilatorom konfigurisanim u HS modu. U takvom slučaju, MCU može započeti obradu programa na jedan nepredvidiv, pogrešan način.

Kada je prisutna neizvesnost u pogledu pouzdanosti *Power-up* mehanizma (spori porast napona V_{DD}) dodatna spoljašnja kola moraju biti korišćena za držanje $\overline{\text{MCLR}}$ linije na niski naponski nivo (MCU u stanju reset) dovoljno dugo da napon napajanja sigurno dostigne nominalnu vrednost, od trenutka priključenja na MCU.



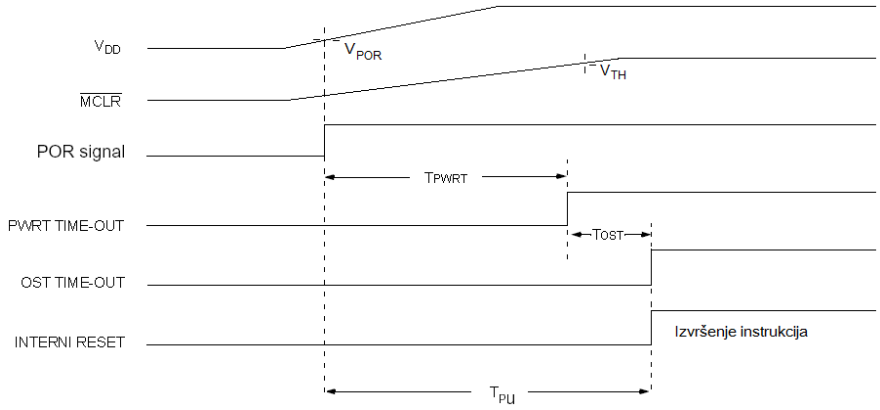
Slika 4.1.5. *Spoljašnje kolo za resetovanje MCU za izvore sa sporim uspostavljanjem napona napajanja.*

Na slici 4.1.5. prikazana je konfiguracija i način priključivanja spoljašnjeg kola za resetovanje MCU za izvore napajanja čija početna brzina porasta napona je manja od 50V/s. Napon na kondenzatoru C, jednak naponu na $\overline{\text{MCLR}}$ liniji, raste sporije od napona napajanja V_{DD} zbog procesa punjenja kondenzatora. Brzina porasta napona na kondenzatoru obrnuto je proporcionalna vremenskoj konstanti kola $\tau=RC$. Prema tome, izborom vremenske konstante spoljašnjeg kola može se uticati na brzinu porasta napona na $\overline{\text{MCLR}}$ liniji koja je, svakako, uvek manja od brzine porasta napona V_{DD}. MCU će biti u stanju reset sve dok napon na $\overline{\text{MCLR}}$ liniji ne dostigne viši prag prebacivanja Schmitt-ovog invertora od $0.85V_{DD}$. Budući da se brzina porasta napona na $\overline{\text{MCLR}}$ liniji može menjati promenom vremenske konstante RC, pravilnim izborom parametara kola može se osigurati normalizacija napona napajanja i stabilizacija oscilovanja kristalnog oscilatora pre nego što napon na $\overline{\text{MCLR}}$ liniji dostigne viši prag prebacivanja invertora, posle čega MCU izlazi iz reset stanja. Poluprovodnička dioda D na slici 4.1.5. koristi se za brzo pražnjenje napunjenog kondenzatora C kroz kolo MCU nakon ukidanja napona napajanja. Napunjeni kondenzator se po ukidanju napona napajanja brzo prazni preko provodne diode koja paralelno vezana sa otpornikom R redukuje ukupnu vremensku konstantu pražnjenja. Time se značajno skraćuje prelazni režim pražnjenja kondenzatora i ubrzava dovođenje RC kola u početno stanje. Po priključenju napona napajanja (proces punjenja kondenzatora) dioda D je inverzno polarizovana i može se ignorisati.

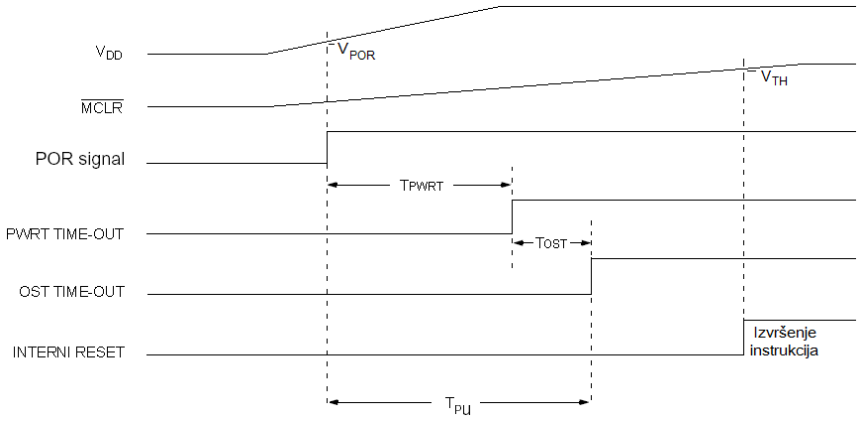
Ako se za pouzdani početak izvršavanja koda koristi spoljašnje kolo za resetovanje MCU, kao na slici 4.1.5., tada su u zavisnosti od vrednosti vremenske konstante RC i brzine porasta napona napajanja moguće tri situacije:

1. Početna brzina porasta napona napajanja $dV_{DD}/dt \geq 50V/s$ i relativno mala vremenska konstanta, tako da u toku vremenskog intervala T_{PU} napon na $\overline{\text{MCLR}}$ liniji dostiže viši prag prebacivanja Schmitt-ovog invertora $V_{TH}=0.85V_{DD}$. Ova situacija ilustrovana je na slici 4.1.6. Kao što se može videti, efekat na početak izvršavanja koda je isti kao i u slučaju odsustva spoljašnjeg RC kola. Naime, MCU neće izaći iz stanja reset do isteka vremenskog intervala T_{PU}.

2. Početna brzina porasta napona napajanja $dV_{DD}/dt \geq 50V/s$ i relativno velika vremenska konstanta, tako da napon na $\overline{\mathbf{MCLR}}$ liniji dostiže viši naponski prag invertora V_{TH} po isteku vremenskog intervala T_{PU}. Situacija je ilustrovana na slici 4.1.7. Može se primetiti da niski naponski nivo na $\overline{\mathbf{MCLR}}$ liniji, ispod $V_{TH}=0.85V_{DD}$, ne dopušta MCU izlazak iz stanja reset i posle isteka vremenskog intervala T_{PU}. Tek nakon što napon na $\overline{\mathbf{MCLR}}$ liniji dostigne viši prag prebacivanja invertora $V_{TH}=0.85V_{DD}$ MCU izlazi iz stanja reset i započinje regularno izvršavanje koda.



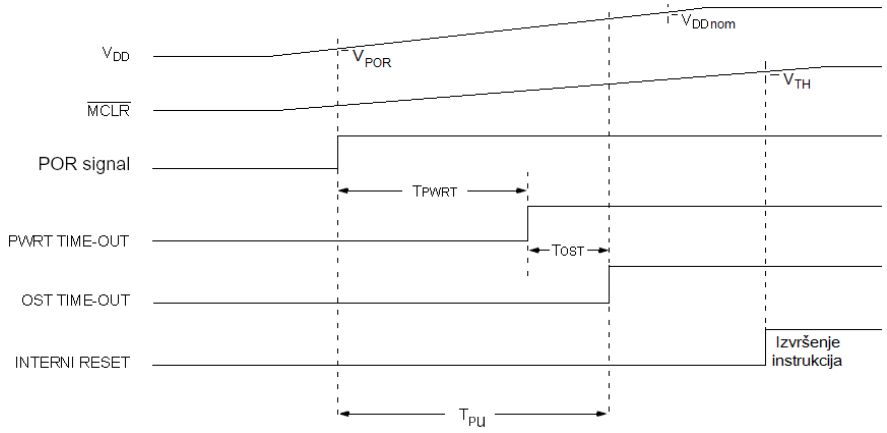
Slika 4.1.6. *Power-up sekvenca za slučaj primene spoljašnjeg RC kola kada napon na* $\overline{\mathbf{MCLR}}$ *liniji dostiže naponski prag* $V_{TH}=0.85V_{DD}$ *u toku vremenskog intervala* T_{PU}.



Slika 4.1.7. *Power-up sekvenca za slučaj primene spoljašnjeg RC kola kada napon na* $\overline{\mathbf{MCLR}}$ *liniji dostiže naponski prag* $V_{TH}=0.85V_{DD}$ *po isteku vremenskog intervala* T_{PU}.

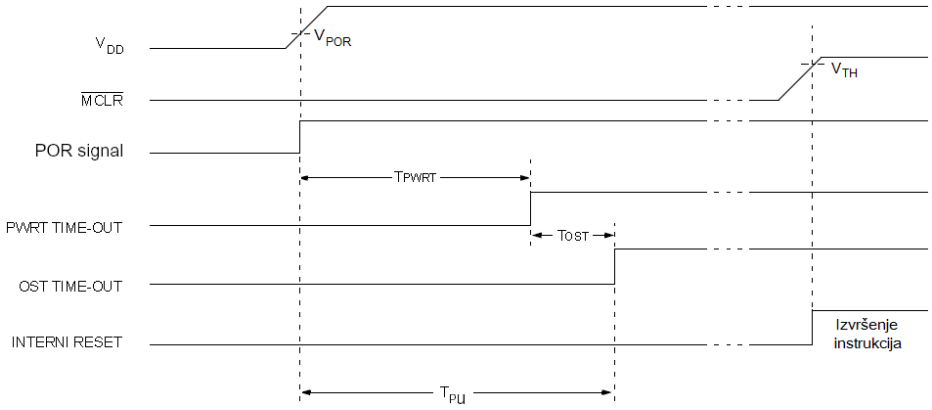
3. Početna brzina porasta napona napajanja $dV_{DD}/dt < 50V/s$ i odgovarajuća vremenska konstanta, tako da napon napajanja dostiže nominalnu vrednost V_{DDnom} po isteku vremenskog intervala T_{PU}, slika 4.1.8. Zbog vremenske konstante RC, odnosno, procesa punjenja kondenzatora, napon na $\overline{\mathbf{MCLR}}$ liniji raste sporije od napona napajanja. Zbog toga će viši prag prebacivanja invertora biti dostignut izvesno vreme posle dostizanja nominalne vrednosti napona napajanja. Ovo vreme direktno zavisi od izbora vremenske konstante kola RC. MCU će izaći iz stanja reset kada napon na $\overline{\mathbf{MCLR}}$ liniji dostigne naponski prag V_{TH}, čime započinje izvršavanje koda. Pravilnim izborom vremenske konstante RC kola može se obezbediti dovoljno dugo vreme držanja MCU u stanju reset

da se posle stabilizacije napona napajanja stabiliše i kristalni oscilator CPU i potom otpočne sa regularnim izvršavanjem koda.



Slika 4.1.8. *Power-up sekvenca za slučaj primene spoljašnjeg RC kola kada i napon napajanja i napon na* $\overline{\text{MCLR}}$ *liniji dostižu nominalnu vrednost* V_{DDnom} *i naponski prag* V_{TH}*, respektivno, po isteku vremenskog intervala* T_{PU}.

Slučaj pod tačkom 3. je praktično situacija u kojoj zbog sporog porasta napona napajanja interna kola ne mogu da zadovolje zahtevani tajming, zbog čega je nužna primena spoljašnjeg reset kola, kao na slici 4.1.5.



Slika 4.1.9. *Držanje MCU u reset stanju i izbor odgovarajućeg trenutka za start aplikacije.*

Slike 4.1.7. i 4.1.8. pokazuju da je MCU u stanju reset sve vreme dok je napon na $\overline{\text{MCLR}}$ liniji niži od višeg praga prebacivanja invertora $V_{TH}=0.85V_{DD}$. Ova se osobina može iskoristiti za držanje MCU u stanju reset, držanjem $\overline{\text{MCLR}}$ linije na niski napon, i potom izabrati odgovarajući trenutak za početak izvršavanja koda na uzlaznoj ivici napona na $\overline{\text{MCLR}}$ liniji, slika 4.1.9. Opisani način startovanja aplikacije koristi se u svrhe testiranja ili sinhronizacije više MCU u paralelnom režimu rada (paralelno procesiranje).

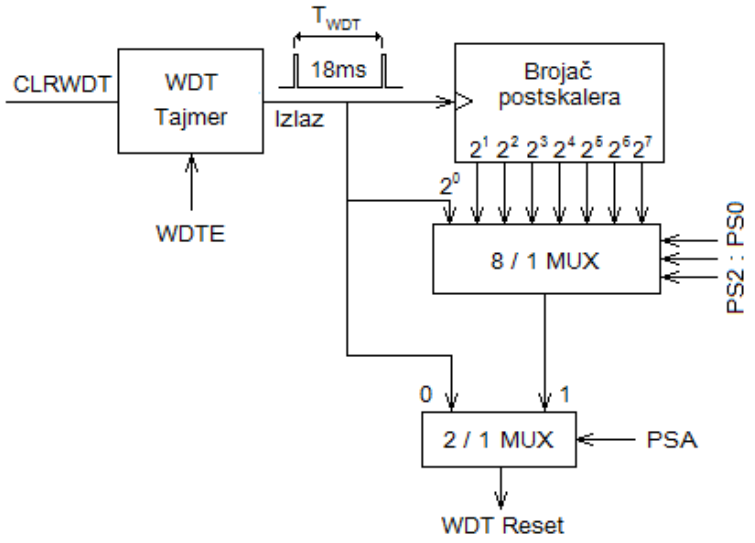
4.1.3. WDT (Watch-Dog Timer) reset u toku normalnog rada MCU i u sleep stanju

WDT (Watch-Dog Timer) tajmer predstavlja tkzv. sigurnosni tajmer namenjen automatskom resetovanju MCU u slučaju zaglavljivanja programa usled prisustva šuma u okruženju ili logički loše projektovanog programa. WDT je integrisani autonomni RC oscilator periodičnih relaksacionih oscilacija koji ne zahteva dodavanje spoljašnjih komponenti. Oscilator će nastaviti sa radom čak i u slučaju blokade taktnog oscilatora CPU, npr. posle izvršenja SLEEP instrukcije. Perioda impulsa sa izlaza WDT tajmera (oscilatora) je temperaturski zavisna veličina i varira od čipa do čipa u opsegu od 7ms do 33ms. Tipična vrednost periode oscilovanja je T_{WDT}=18ms, odnosno, frekvencije oscilovanja $f_{WDT}=1/T_{WDT}$=55.55Hz. WDT tajmeru se programski može dodeliti deljivi resurs-postskaler koga čini 7-bitni binarni brojač. Brojač se koristi kao delitelj frekvencije taktnih impulsa sa izlaza WDT tajmera, slika 4.1.10. Odnos deljenja postskalera se programski može birati iz opsega od $1:2^0$ do $1:2^7$. Zahvaljujući postskaleru perioda WDT tajmera može biti programski umnožena (frekvencija podeljena) do maksimalnih 2^7 T_{WDT} što tipično iznosi 2.304s.

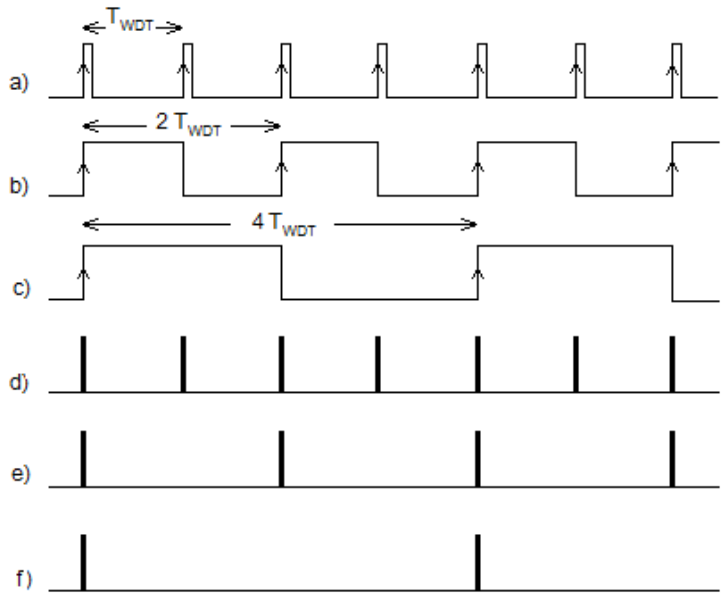
Može se primetiti da je i najkraće trajanje periode WDT tajmera T_{WDT}=18ms mnogostruko veće od trajanja instrukcijskog ciklusa MCU $T_{CY}=4T_{OSC}$, gde je T_{OSC} perioda taktnog oscilatora CPU. Npr. za f_{OSC}=20MHz T_{CY}=200ns.

Sa slike 4.1.10. se jasno vidi da se jednim bitom registra specijalne namene, bit PSA (*PreScaler Assignment*) za dodelu preskalera, može birati jedan od dva kanala dvokanalnog multipleksora 2/1. Npr. ako je PSA=0 izlaz WDT tajmera direktno se spaja sa izlazom WDT reset. Prema tome, brojač postskalera nema uticaja na multiplikaciju periode tajmera. Ako je PSA=1 tada se osnovna perioda impulsa sa izlaza WDT tajmera ili jedna od sedam umnoženih sa jednog od sedam izlaza brojača postskalera propušta kroz izabrani kanal osmokanalnog multipleksora i vodi do izlaza WDT reset. Izbor jednog od osam kanala multipleksora 8/1 vrši se selekcionim bitovima registra specijalne namene PS0, PS1 i PS2.

WDT tajmer se programiranjem konfiguracione reči MCU može omogućiti WDTE=1 ili onemogućiti WDTE=0, vidi sliku 4.1.10.



Slika 4.1.10. *Pojednostavljena blok šema WDT tajmera sa postskalerom.*



Slika 4.1.11. *Primer talasnih oblika signala na izlazima postskalera pri deljenju frekvencije oscilatora WDT tajmera sa 2 i sa 4.*

Na slici 4.1.11. prikazani su talasni oblici signala na izlazu WDT tajmera a) i na izlazima postskalera b), c). Impulsi sa izlaza WDT tajmera taktuju 7-bitni brojač postskalera na čija se prva dva izlaza, označena na slici 4.1.10. sa 2^1 i 2^2, generišu talasni oblici dvostruko i četvorostruko manje frekvencije, odnosno, isto toliko veće periode. Brojač postskalera zasnovan je na ivičnim flip-flopovima tako da se promene stanja na njihovim pravim i komplementarnim izlazima događaju na uzlaznoj ivici taktnog signala. Pogodnim kolima za diferenciranje izdvajaju se potom samo prednje ivice ovih signala veoma kratkog trajanja, d), e), f), kojima se vrši resetovanje MCU. Talasni oblici signala na preostalih pet izlaza postskalera dobijeni su na sličan način kao i prethodna dva sa drugačijim odnosima deljenja $1:2^3$ do $1:2^7$.

Zadatak WDT tajmera je da periodično na svakih T_{WDT}=18ms ili više u slučaju dodele postskalera resetuje MCU izlazom WDT reset. Asembler MCU poseduje instrukciju CLRWDT (*CleaR Watch-Dog Timer*) koja kada se izvrši restartuje oscilator WDT tajmera vračajući ga u početno stanje generisanja jedne periode. Ako se u programu koji se izvršava umetne instrukcija CLRWDT tako da se izvrši pre isteka periode WDT tajmera T_{WDT} do reseta MCU neće doći jer se oscilator tajmera restartuje vraćajući se u početno stanje i ponovno započinjući generisanje periode. Samo u slučaju zaglavljivanja programa do izvršenja instrukcije CLRWDT neće doći, pa time ni do restarta oscilatora WDT tajmera, usled čega po isticanju periode WDT tajmera dolazi do resetovanja MCU. Ako je zaglavljivanje aplikacije rezultat trenutnih smetnji, novi restart aplikacije resetovanjem MCU znači novi početak izvođenja programa pod drugačijim okolnostima (odsustvo smetnji) koje neće dovesti do zaglavljivanja.

Opisani autonomni mehanizam zamenjuje operatera koji bi u slučaju zaglavljivanja procesa morao manuelno spoljašnjim resetom restartovati aplikaciju.

Budući da je oscilator WDT tajmera autonoman, resetovanje MCU WDT tajmerom se može vršiti u toku normalnog izvršenja programa ili u toku sleep režima. Ako se reset dogodi u toku normalnog izvršenja programa procesor će započeti izvršenje programa sa adrese reset vektora resetujući pri tome $\overline{\mathbf{TO}}$ bit i setujući $\overline{\mathbf{PD}}$ bit statusnog registra. Resetovanje izlazom WDT kada je MCU u sleep stanju imaće za posledicu izvršenje prve instrukcije programa koja sledi iza SLEEP instrukcije posle kratkog kašnjenja od T_{OST}=1024T_{OSC} ako je izabran kristalni oscilator. Ova situacija predstavlja prirodni nastavak izvršenja koda posle vremena provedenog u režimu niske potrošnje pri čemu su oba bita statusnog registra $\overline{\mathbf{TO}}$ i $\overline{\mathbf{PD}}$ resetovana.

4.1.4. BOR (Brown-Out Reset) reset

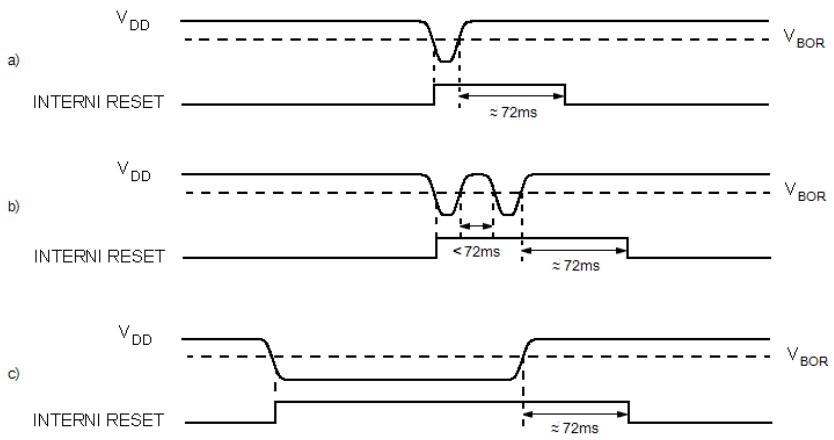
Naprednije serije PIC MCU kao što je PIC16F877 poseduju ugrađeni interni reset mehanizam tkzv. BOR (Brown-Out Reset) reset za povećanje operativne pouzdanosti MCU. Ovaj mehanizam obezbeđuje resetovanje MCU ako napon napajanja u toku normalnog izvršenja programa opadne ispod marginalne vrednosti V_{BOR}, što tipično iznosi 4±0.3V, i ostane ispod ove vrednosti duže od 100μs. Imajući u vidu da se uglavnom radi o tranzijentnim promenama na linijama napajanja MCU, napon napajanja posle izvesnog vremena može prekoračiti marginalnu vrednost V_{BOR} i normalizovati se. Ako je vreme za koje je napon napajanja ispod vrednosti V_{BOR} kraće od 100μs do reseta MCU neće doći.

Ukoliko se BOR reset dogodi uređaj će ostati u stanju reset sve dok napon napajanja ne poraste iznad marginalne vrednosti V_{BOR}. Međutim, povećanje V_{DD} iznad marginalne vrednosti V_{BOR} inicira uzastopno PWRT tajmer koji generiše dodatni reset tajming u trajanju T_{PWRT}=72ms i potom OST tajmer koji će zadržati MCU u stanju reset još za vreme T_{OST}=1024T_{OSC} ako je izabrani taktni oscilator CPU kristalni.

U toku vremenskog intervala T_{PWRT}+T_{OST} može doći do ponovnog pada napona napajanja ispod vrednosti V_{BOR} u trajanu dužem od 100μs što će usloviti restart BOR procesa a kada napon napajanja ponovo naraste iznad V_{BOR} i restart PWRT i OST tajmera uzastopno.

BOR reset kolo će resetovati MCU i kada POR signal ne trigeruje PWRT tajmer zbog eventualnog sporog porasta napona napajanja.

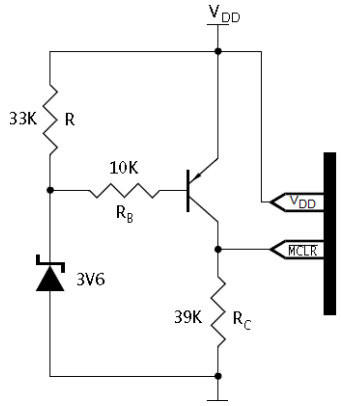
PWRT tajmer će automatski biti omogućen kada je omogućeno kolo za BOR reset (setovanjem bita **BODEN** konfiguracione reči) bez obzira na stanje bita $\overline{\textbf{PWRTE}}$ konfiguracione reči.



Slika 4.1.12. *BOR reset u različitim situacijama.*

Na slici 4.1.12. su ilustrovane tipične opisane situacije pojave i restarta BOR reseta. Slučaj pod a) predstavlja kratkotrajnu tranzijentnu pojavu pada napona napajanja ispod dozvoljene margine u trajanju dužem od 100μs. Interno reset kolo drži MCU u stanju reset do trenutka povećanja napona napajanja iznad praga V_{BOR}. Reset stanje će, međutim, trajati još izvesno vreme zbog uzastopnog trigerovanja PWRT i OST tajmera. Slika 4.1.12. c) ilustruje BOR reset kada pad napona napajanja ispod praga V_{BOR} ima duže trajanje dok slučaj b) predstavlja restart BOR reseta i potom oba tajmera, uzrokovan varijacijom napona napajanja.

Resetovanje MCU usled pada napona napajanja ispod marginalne vrednosti je nužno zbog nekompatibilnosti smanjenih naponskih nivoa na I/O linijama na kojima mogu biti povezani različiti uređaji.



Slika 4.1.13. *Eksterno kolo za BOR reset.*

Ako PIC MCU nije opremljen ugrađenim kolom za BOR reset, eksterno kolo sa slike 4.1.13., čiji je izlaz povezan sa $\overline{\text{MCLR}}$ linijom, može poslužiti za resetovanje MCU pri padu napona napajanja V_{DD} ispod 4.15V. Duboko zasićeni tranzistor pri nominalnoj vrednosti napona napajanja od 5V osigurava visok napon od oko 4.9V na $\overline{\text{MCLR}}$ liniji obezbeđujući normalno izvršavanje instrukcija MCU. Pri naponu $V_{DD}=V_Z+V\gamma\approx3.6V+0.55V=4.15V$ tranzistor prestaje da provodi usled čega napon na $\overline{\text{MCLR}}$ liniji opada na približno 0V i time resetuje MCU. Do resetovanja MCU dolazi, međutim, i pri nešto većoj vrednosti napona napajanja od oko 4.3V, budući da se reset događa pri naponu na $\overline{\text{MCLR}}$ liniji $V_{MCLR}\leq0.2V_{DD}$.

Pojava POR ili BOR reseta se može utvrditi ispitivanjem statusnih bitova POR i BOR kontrolnog/statusnog registra PCON, tabela 4.1.3.

Ostale vrste reseta mogu biti identifikovane ispitivanjem bitova statusnog registra $\overline{\mathbf{TO}}$ i $\overline{\mathbf{PD}}$, u skladu sa tabelom 4.1.4.

PCON REGISTAR

	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-1
	-	-	-	-	-	-	$\overline{\mathbf{POR}}$	$\overline{\mathbf{BOR}}$
	bit 7							bit 0
bit 7-2	**Neimplementirani bitovi:** Čitaju se kao '0'.							
bit 1	$\overline{\mathbf{POR}}$: Power-on Reset statusni bit. 1 = Nije se dogodio POR reset. 0 = Dogodio se POR reset (bit mora biti setovan programski posle POR reseta).							
bit 0	$\overline{\mathbf{BOR}}$: Brown-Out Reset statusni bit. 1 = Nije se dogodio BOR reset. 0 = Dogodio se BOR reset (bit mora biti setovan programski posle BOR reseta).							

Tabela 4.1.3. *Sadržaj kontrolnog/statusnog registra PCON (Power CONtrol).*

$\overline{\mathbf{POR}}$	$\overline{\mathbf{BOR}}$	$\overline{\mathbf{TO}}$	$\overline{\mathbf{PD}}$	**Komentar**
0	x	1	1	Power-on Reset
1	0	1	1	Brown-out Reset
1	1	0	1	WDT Reset u toku normalnog izvršenja instrukcija
1	1	0	0	WDT Reset u sleep stanju (Wake-up)
1	1	u	u	MCLR Reset u toku normalnog izvršenja instrukcija
1	1	1	0	MCLR Reset u sleep stanju ili 'buđenje' iz sleep stanja prekidom

Tabela 4.1.4. *Uticaj različitih vrsta reset signala na statusne bitove.*

Vrste reseta	**PC**	**STATUS**	**PCON**
Power-on Reset	000h	0001 1xxx	-0x
MCLR Reset u toku normalnog izvršavanja inst.	000h	000u uuuu	-uu
MCLR Reset u SLEEP stanju	000h	0001 0uuu	-uu
WDT Reset	000h	0000 1uuu	-uu
WDT Wake-up (u SLEEP stanju)	PC + 1	uuu0 0uuu	-uu
Brown-out Reset	000h	0001 1uuu	-u0
'Buđenje' prekidnim signalom iz SLEEP stanja	PC + 1	uuu1 0uuu	-uu

Tabela 4.1.5. *Uticaj reset uslova na stanja nekih registara specijalne namene.*

U tabeli 4.1.5. su prikazana stanja nekih registara specijalne namene kao što su PC (programski brojač), STATUS i PCON, posle pojave jednog od šest reset uslova. Poslednja vrsta opisuje buđenje MCU iz sleep stanja prekidnim signalom i mogućnost nastavka izvršenja programa sa adrese za jedan veće (PC+1) od adrese SLEEP instrukcije kojim je MCU uveden u standby režim. S obzirom da se buđenje vrši prekidnim signalom moguće je i grananje programa na adresu prekidnog vektora ako je GIE bit pre ulaska u standby režim bio setovan.

Poglavlje 5

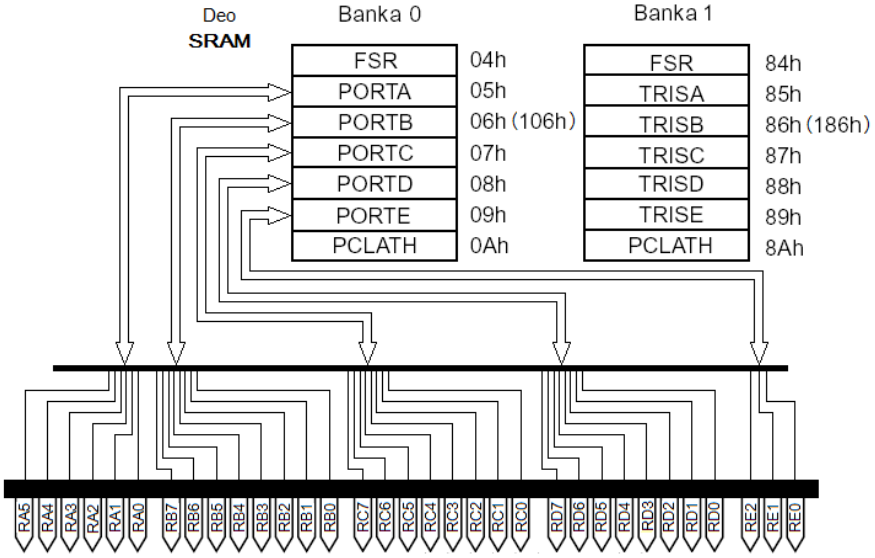
POVEZIVANJE MCU SA OKRUŽENJEM – I/O PORTOVI OPŠTE I SPECIJALNE NAMENE

Ulazno/izlazni (I/O) portovi MCU predstavljaju fundamentalnu direktnu fizičku vezu priferijskih i CPU jedinice MCU sa spoljašnjim svetom. Broj portova kao i broj I/O linija jednog određenog porta varira od serije do serije PIC MCU, npr. od 5 I/O linija za osmopinske do 33 I/O linija za četrdesetopinske PIC MCU uređaje. Veći broj I/O linija, odnosno, portova povećava fleksibilnost primene ali i gabarit MCU. PIC16F877 mikrokontroler poseduje pet I/O portova i to:

- PORTA – 6-bitni
- PORTB – 8-bitni
- PORTC – 8-bitni
- PORTD – 8-bitni
- PORTE – 3-bitni

I/O linije portova su u opštem slučaju multifunkcionalne i programski konfigurabilne, od digitalnih I/O linija opšte namene, digitalnih I/O linija specijalne namene do analognih I/O linija.

Principijelno, svaki I/O port može biti razmatran kao registar čiji sadržaj je vidljiv spoljašnjem svetu.

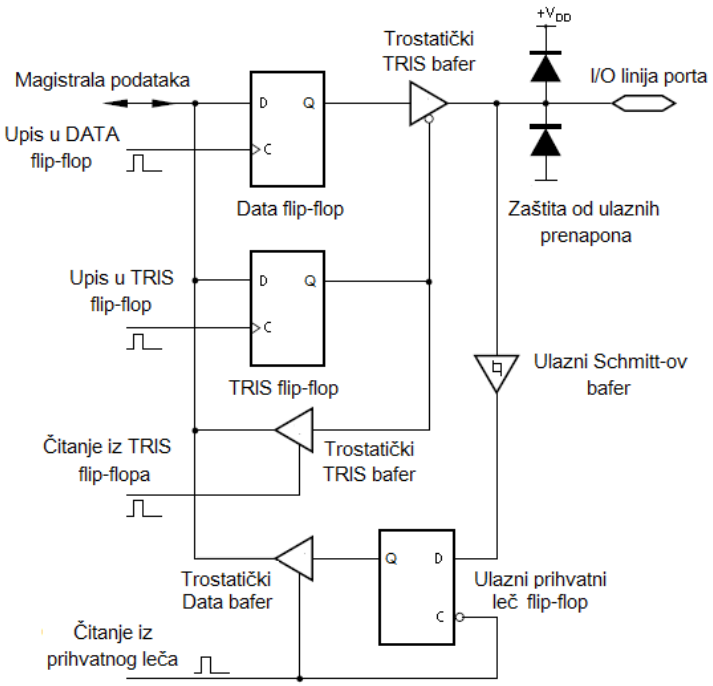


Slika 5.1. *Registri podataka I/O portova i njihovi direkcioni registri PIC16F877 MCU mapirani u SRAM memoriji.*

Pojednostavljeni princip mapiranja svih I/O portova PIC16F877 i njihovih direkcionih registara u deo RAM memorije MCU ilustrovan je na slici 5.1. Prenos podataka na relaciji registar-I/O linije porta je dvosmeran pa se govori o bidirekcionom ili bilateralnom portu pri čemu se svi bitovi porta prenose istovremeno (paralelni port). Registrima svih šest I/O portova pridruženi su odgovarajući TRIS direkcioni registri, kao na slici 5.1. Ovi registri se koriste za programsku selekciju smera prenosa podataka za svaku I/O liniju odgovarajućeg porta pojedinačno. Programskim upisom nule na određenu bitsku poziciju određenog TRIS registra odgovarajuća linija porta kome je pridružen TRIS registar konfiguriše se kao izlazna i obrnuto.

Određeni broj integrisanih periferija MCU koristi odgovarajuće linije I/O portova kao zajednički resurs pa se zbog deljenja resursa stvara privid o većem broju I/O linija. Na primer, PIC16F877 deli pet linija porta A i tri linije porta E na analogne ulaze osmokanalnog A/D konvertora kao integrisane periferije MCU i digitalne I/O linije opšte namene.

Da bi se bolje razumele funkcije i karakteristike I/O porta potrebno je razmotriti pojednostavljenu hardversku strukturu jedne tipične I/O linije porta, prikazane na slici 5.2.



Slika 5.2. *Pojednostavljena hardverska struktura tipične I/O linije porta.*

Upis bita podatka u liniju porta konfigurisanu kao izlaz vrši se taktovanjem D flip-flopa za podatke (data flip-flop) na čijoj magistrali podataka (ulaz D) je postavljena odgovarajuća bitska vrednost. Za konfigurisanje linije porta kao izlazne neophodno je pre opisane procedure sa magistrale podataka taktnim signalom upisati u TRIS flip-flop logičku nulu koja će omogućiti gornji trostatički TRIS bafer i time direktno povezati izlaz flip-flopa za podatke na izlaznu liniju porta.

Čitanju stanja ulazne linije porta prethodi konfigurisanje linije porta kao ulazne upisom logičke jedinice sa magistrale podataka u TRIS flip-flop na odgovarajući taktni signal. Time se izlaz gornjeg trostatičkog TRIS bafera dovodi u stanje visoke impedanse (odspojen izlaz) čime se omogućava prenos ulaznog podatka sa ulazne linije preko Schmitt-ovog bafera za eliminaciju šumova do ulaza prihvatnog leč flip-flopa (ulaz D). Generisanjem taktnog signala za čitanje iz prihvatnog leča ulazni podatak se preko trostatičkog bafera za podatke prenosi na magistralu podataka.

Stanje TRIS flip-flopa (direkcija linije podataka) može se pročitati omogućenjem donjeg trostatičkog TRIS bafera čiji ulaz je direktno povezan sa izlazom TRIS flip-flopa a izlaz sa magistralom podataka. Prema tome, TRIS bit se može čitati iz ili upisivati u TRIS flip-flop sa magistrale podataka. Razmatrajmo primer dela programa za izmenu direkcije linije RB7 porta B napisan na asembleru i C jeziku.

```
BCF TRISB,7          ;Brisanje MSB bita TRISB registra (linija RB7 porta B je izlazna)
```

```
#BYTE TRIS_B=0x86      //Deklaracija adrese TRIS_B registra
BIT_CLEAR(TRIS_B,7)    //Brisanje MSB bita TRIS_B registra
```

BCF je primer instrukcije tipa *read-modify-write* gde procesor najpre čita stanje TRISB registra, modifikuje njegov sadržaj i potom vrši upis u TRISB registar. Kompletna akcija izvršava se u jednom instrukcijskom ciklusu u kome procesor čita, modifikuje sadržaj i potom upisuje u registar specijalne namene-TRISB. Sledeći primer instrukcija takvog tipa su instrukcije za inkrementiranje i dekrementiranje sadržaja registara koje se u realnosti izvršavaju tako što se najpre izvrši transfer sadržaja registra iz SRAM memorije podataka u privremeni procesorski registar (čitanje), potom sadržaj privremenog registra inkrementira ili dekrementira koristeći ALU (modifikacija) i na kraju prenosi u SRAM memoriju na odgovarajuću adresu (upis).

Pokušaj čitanja linije porta sa slike 5.2. kada je ona konfigurisana kao izlazna će u većini situacija efektivno kopirati stanje flip-flopa za podatke (data flip-flop) u CPU registar, međutim to nije uvek slučaj. Naime, ako je izvorna ili ponorna struja kroz izlaznu liniju velika, napon na liniji za koju je povezan ulaz Schmitt-ovog bafera može značajno odstupati od normalnih logičkih nivoa usled povećanog pada napona na izlaznoj otpornosti gornjeg trostatičkog TRIS bafera, slika 5.2. Ovo može dovesti do pogrešnog čitanja pravog stanja flip-flopa za podatke.

Pokušaj upisa na liniji porta koja je konfigurisana kao ulazna će za razliku od prethodno opisane situacije rezultirati pravilnom izmenom stanja flip-flopa za podatke. Budući da je izlaz gornjeg trostatičkog TRIS bafera u stanju visoke impedanse kada je linija porta konfigurisana kao ulazna, slika 5.2., upis u flip-flop za podatke izvršiće se nesmetano bez uticaja na stanje ulazne linije do eventualne docnije izmene direkcije linije porta. Opisana mogućnost upisa u flip-flop za podatke na način nevidljiv spoljašnjem svetu je važna u situaciji reseta PIC MCU. Naime, posle reseta MCU sve linije svih portova automatski se konfigurišu kao digitalni ulazi, odnosno, svi bitovi direkcionih TRIS registara portova su setovani FFh. Za kontrolu spoljašnjeg uređaja priključenog na port program prvo upisuje inicijalno stanje u flip-flopove podataka a potom se samo promenom direkcije linija porta (upis logičke nule u TRIS flip-flop) upisano stanje prenosi na izlazne linije porta jednovremeno.

Za korektan rad sa portovima MCU posebnu pažnju treba posvetiti ograničenjima po pitanju strujnog kapaciteta linija porta. Prema specifikacijama proizvođača data su sledeća ograničenja za dva slučaja:

- Struja ponora izlazne linije porta I_{OL} kada je linija na niskom potencijalu (stanje logičke nule) ne sme premašiti vrednost od 8.5mA za graničnu vrednost napona logičke nule na liniji od $V_{OL}=V_{OLMAX}=0.6V$.
- Izvorna struja izlazne linije porta I_{OH} kada je linija na visokom potencijalu (stanje logičke jedinice) ne sme premašiti vrednost od 3mA za graničnu vrednost napona logičke jedinice na liniji od $V_{OH}=V_{OHMIN}=V_{DD}-0.7V$.

Veće izvorne/ponorne struje od navedenih limita vode degradaciji naponskih nivoa logičke jedinice/nule i dopuštene su ako je degradacija prihvatljiva. Podaci proizvođača govore da maksimalna dopuštena izlazna izvorna/ponorna struja po jednoj I/O liniji porta, za koju vrednost neće doći do oštećenja, iznosi 25mA, respektivno. Međutim, kada se koristi više nego jedna I/O linija globalna ograničenja moraju biti poštovana. Kombinovano korišćenje I/O linija svih portova A, B, C, D i E, ograničava ukupnu maksimalno dopuštenu izvorno/ponornu struju na ±200mA.

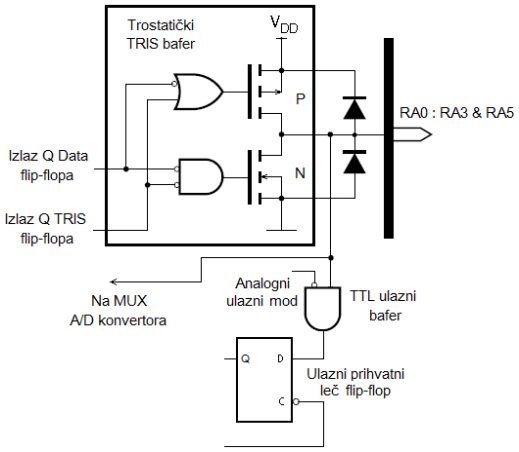
Proizvođač specificira i maksimalnu struju kroz liniju napajanja V_{DD} od I_{DD}=250mA i kroz zajedničku liniju V_{SS}, I_{SS}=300mA.

I/O linije portova konfigurisane kao ulazne sa TTL ulaznim baferima, kao napon logičke nule prepoznaju ulazne napone koji zadovoljavaju uslov V_{IL}≤0.5V a napon logičke jedinice za V_{IH}≥2V. I/O linije portova konfigurisane kao ulazne sa Schmitt-ovim ulaznim baferima imaju ulazne pragove prebacivanja stanja bafera na izlazu od V_{IL}=0.2V_{DD} (1V za V_{DD}=5V) i V_{IH}=0.8V_{DD} (4V za V_{DD}=5V).

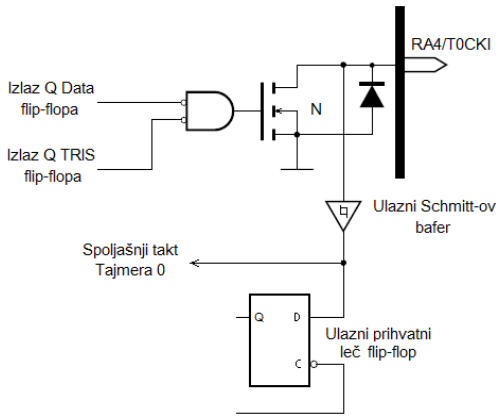
Blok dijagram jedne tipične I/O linije porta, prikazan na slici 5.2., može razlikovati od porta do porta u funkciji namene linija porta i električnih karakteristika.

5.1. ORGANIZACIJA I MULTIPLEKSNE FUNKCIJE PORTA A

Port A je šestobitni bidirekcioni I/O port kome je pridružen TRIS A direkcioni registar. Oba registra mapirana su u SRAM memoriji na adresama 05h i 85h, respektivno. Tipična digitalna logika pet linija porta <RA3:RA0> i RA5 identična je onoj prikazanoj na slici 5.2. sa neznatnim izmenama ilustrovanim na slici 5.1.1. Ozbiljnija izmena jednog dela logike odnosi se samo na liniju RA4, slika 5.1.2.



Slika 5.1.1. *Pojednostavljena struktura višefunkcionalnih I/O linija <RA3:RA0> i RA5 porta A.*



Slika 5.1.2. *Pojednostavljena struktura I/O linije RA4 porta A.*

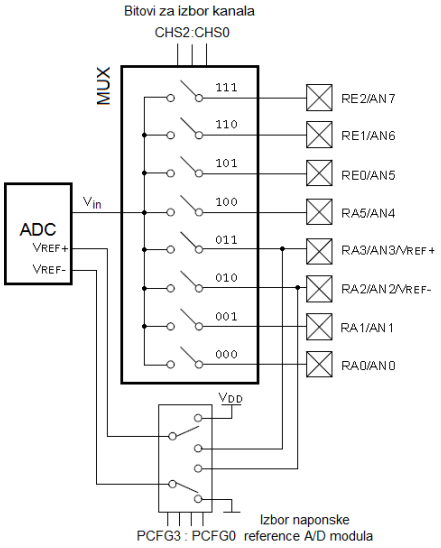
Kao što se sa slike 5.1.1. može videti, trostatički TRIS bafer implementiran je na bazi serijske veze N i P-kanalnog MOS tranzistora. Ulazni bafer navedenih pet linija porta je TTL tipa a svaka od pet linija može biti iskorišćena i kao jedan od osam analognih ulaza integrisane periferije MCU-A/D konvertora. Kanali A/D konvertora su multipleksirani, tako da je svaka analogna ulazna linija povezana sa odgovarajućim ulazom analognog multipleksora (MUX), slika 5.1.3.

Izlaz trostatičkog TRIS bafera je u stanju visoke impedanse kada je TRIS flip-flop setovan jer nijedan od dva MOS tranzistora nije provodan, čime je flip-flop za podatke izolovan od I/O linije. Odgovarajuća linija porta je tada konfigurisana kao ulazna a da li će se ista koristiti kao digitalni ili analogni ulaz zavisiće od izbora radnog režima (analogni ulazni mod na jednom od dva ulaza TTL ulaznog bafera). Kada je TRIS flip-flop resetovan linija je konfigurisana kao digitalni izlaz opšte namene. Stanje izlazne linije tada zavisi od stanja flip-flopa za podatke. Naime, kada je izlaz Q flip-flopa za podatke postavljen u stanje logičke jedinice (flip-flop

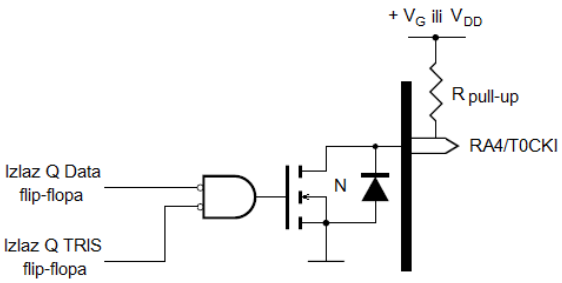
setovan) provodan je samo P-kanalni tranzistor tako da je izlazna linija na visokom potencijalu dok je za stanje logičke nule provodan samo N-kanalni tranzistor zbog čega je linija na niskom potencijalu.

Struktura linije porta A RA4 izmenjena je u odnosu na ostalih pet linija, slika 5.1.2, budući da je u izlaznom delu uklonjen P-kanalni tranzistor a na mesto TTL ulaznog bafera implementiran Schmitt-ov bafer. Takva konfiguracija izlaznog stepena poznata je pod nazivom 'otvoreni drejn' (*open-drain*). Za uspostavljanje naponskog nivoa logičke jedinice u slučaju blokiranog tranzistora (otvoreno kolo) potrebna je ugradnja spoljašnjeg pull-up otpornika između drejna N-kanalnog tranzistora i linije napajanja, slika 5.1.4. Za napajanje pull-up otpornika može se koristiti napon napajanja MCU V_{DD} ili napon posebnog spoljašnjeg generatora V_G čija maksimalna vrednost, u skladu sa preporukama proizvođača, ne sme prekoračiti 8.5V. Otuda naponski nivoi logičkih stanja na liniji RA4 mogu biti različiti u odnosu na ostale linije porta A. Otpornost pull-up otpornika ne sme biti suviše mala, zbog velike struje kroz zatvoreni tranzistorski prekidač i mogućeg oštećenja, niti suviše velika da bi se smanjio uticaj indukovanih elektromagnetnih smetnji spoljašnjih izvora napajanja. Dobar kompromis je izbor vrednosti otpornika iz opsega 10KΩ do 100KΩ.

Linija RA4 se može koristiti kao digitalna I/O linija opšte namene ili kao digitalni ulaz za taktovanje integrisane periferije MCU-tajmera 0. Međutim, ova linija ne može biti korišćena kao analogni ulaz A/D konvertora. Zamenom TTL ulaznog bafera Schmitt-ovim povećan je imunitet linije na šumove i smetnje u slučaju kada se ova linija koristi za spoljašnje taktovanje tajmera 0.



Slika 5.1.3. *Principijelna blok šema osmokanalnog analognog multipleksora (MUX) čiji su ulazi povezani sa osam analognih ulaznih linija.*



Slika 5.1.4. *Povezivanje spoljašnjeg pull-up otpornika na open-drain izlaz linije RA4.*

Posle POR (Power On Reset) reseta sve linije porta A automatski se konfigurišu kao analogni ulazi izuzev RA4 linije koja se konfiguriše kao digitalni ulaz, a čitaju se kao logičke nule.

Sve linije porta A, izuzev linije RA4, multipleksirane su funkcijom analognih ulaza za A/D konverziju a neke i funkcijama priključaka za pozitivni kraj $+V_{REF}$ i negativni kraj $-V_{REF}$ spoljašnje naponske reference A/D konvertora. Vrsta operacije svake linije porta A bira se postavljanjem/brisanjem kontrolnih bitova kontrolnog registra ADCON1 A/D konvertora. Kada se linije porta A koriste kao analogni ulazi, odgovarajući bitovi direkcionog TRIS A registra moraju biti setovani.

Prema tome, za konfigurisanje linija porta A nije dovoljno programirati samo port A i njegov direkcioni TRIS A registar već je potrebno programirati i kontrolni registar A/D konvertora ADCON1, kao u primeru asemblerske programske sekvence prikazane u tabeli 5.1.1.

```
; ********************************************************************************
; *     Tipična asemblerska programska sekvenca za konfigurisanje linija porta A     *
; ********************************************************************************

BCF STATUS, RP0 ;
BCF STATUS, RP1 ; Selekcija Banke 0 (PORTA registar je mapiran u banci 0)
CLRF PORTA      ; Inicijalizacija porta A
BSF STATUS, RP0 ; Sel. Banke 1 (ADCON1 i TRISA registri su mapirani u banci 1)
MOVLW 0x06
MOVWF ADCON1    ; Konfig. svih linija porta A kao digitalnih I/O linija (0x06)
MOVLW 0xCF      ; Inicijalizacija direkcije linija porta A
MOVWF TRISA     ; Konfig. linija RA<3:0> kao dig. ulaza i RA<5:4> kao dig. izlaza
```

Tabela 5.1.1. *Tipična asemblerska programska sekvenca za konfigurisanje linija porta A.*

Tipičan primer za konfiguraciju svih linija porta A kao pet analognih ulaza, kao i konfiguraciju A/D konvertora za čitanje digitalnog ekvivalenta analogne veličine sa jednog od pet ulaznih kanala-kanal 0, prikazan je u tabeli 5.1.2. jednostavnom programskom sekvencom na C jeziku CCS C kompajlera.

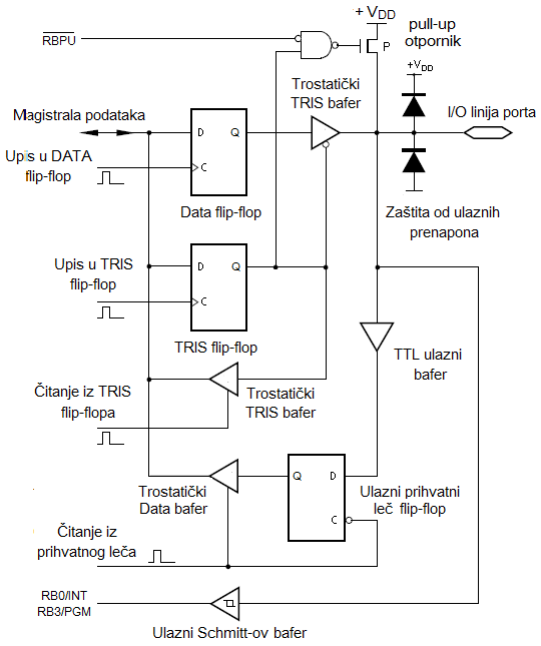
```
     //************************************************************************
     //   C programska sekvenca za konfigurisanje linija porta A kao analognih   *
     //           ulaza i za A/D konverziju ulazne veličine na liniji RA0        *
     //************************************************************************

setup_adc_ports( ALL_ANALOG ); //konfiguracija svih linija porta A kao analognih ulaza
setup_adc(ADC_CLOCK_INTERNAL ); //konfiguracija taktnog generatora A/D konvertora
set_adc_channel( 0 );           //selekcija kanala 0, odnosno, linije RA0/AN0 porta A
delay_us(20);                   //neophodno kašnjenje zbog vremena akvizicije
dig_value = read_adc();  //čitanje 10-bitnog digitalnog ekvivalenta analogne vrednosti
setup_adc( ADC_OFF );    //isključenje A/D modula u cilju smanjenja potrošnje MCU
```

Tabela 5.1.2. *C programska sekvenca za konfigurisanje linija porta A kao analognih ulaza, A/D konverziju ulazne veličine na liniji RA0 i čitanje digitalnog ekvivalenta.*

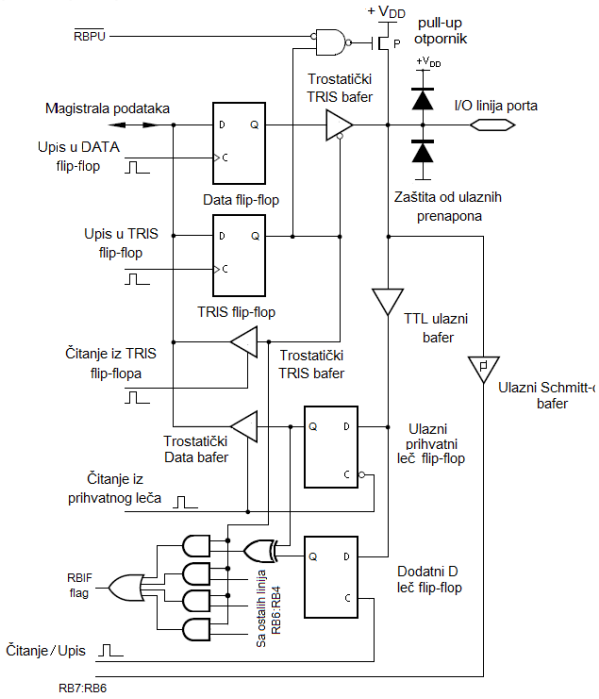
5.2. ORGANIZACIJA I MULTIPLEKSNE FUNKCIJE PORTA B

Port B je osmobitni bidirekcioni I/O port kome je pridružen TRIS B direkcioni registar. Oba registra mapirana su u SRAM memoriji na adresama 06h i 86h kao i 106h i 186h, respektivno. Tipična digitalna logika linija nižeg nibla porta B <RB3:RB0> prikazana je na slici 5.2.1.

Linija RB3 je multipleksirana funkcijom programiranja pod niskim naponom (Low Voltage Programming) dok linija RB0 predstavlja izvor prekidnog signala, kada su konfigurisane kao ulazne. Obe linije su zbog dodatnih funkcija opremljene Schmitt-ovim triger baferima na ulazu za eliminisanje smetnji. Kada se koriste kao digitalne I/O linije opšte namene tip ulaznog bafera je TTL, kao na slici 5.2.1. Linije porta B RB1 i RB2 su digitalne I/O linije opšte namene sa TTL ulaznim baferom i ne poseduju Schmitt-ov bafer. Sve linije porta B opremljene su internim programski kontrolisanim pull-up otpornikom koga čini otpornost kanala između drejna i sorsa P-kanalnog MOS tranzistora, približne vrednosti oko 20KΩ. Posle POR reseta svi tranzistori su blokirani, odnosno, svi pull-up otpornici onemogućeni, iako su sve linije porta B konfigurisane kao ulazne. Pull-up otpornik se automatski isključuje kada se odgovarajuća linija porta B konfiguriše kao izlazna. MSB bit OPTION registra $\overline{\text{RBPU}}$ koristi se za programsko omogućenje/onemogućenje individualnih pull-up otpornika na linijama porta B. Uprkos činjenici da svih osam pull-up otpornika kontroliše jedan $\overline{\text{RBPU}}$ bit OPTION registra, samo linije konfigurisane kao ulazne mogu imati priključen pull-up otpornik.



Slika 5.2.1. *Digitalna logika linija nižeg nibla porta B, <RB3:RB0>.*



Slika 5.2.2. *Digitalna logika linija višeg nibla porta B, <RB7:RB4>.*

Kao što je rečeno, linija RB0 porta B predstavlja izvor spoljašnjeg prekida samo u slučaju kada je konfigurisana kao digitalni ulaz. Digitalna logika ove linije prepoznaje pojavu rastuće ili opadajuće ivice impulsa na liniji RB0 kao prekidni zahtev. Izbor rastuće/opadajuće ivice impulsa za generisanje prekidnog zahteva vrši se programski, setovanjem/resetovanjem bita INTEDG OPTION registra, respektivno. Ova vrsta prekida može biti iskorišćena za buđenje MCU iz sleep

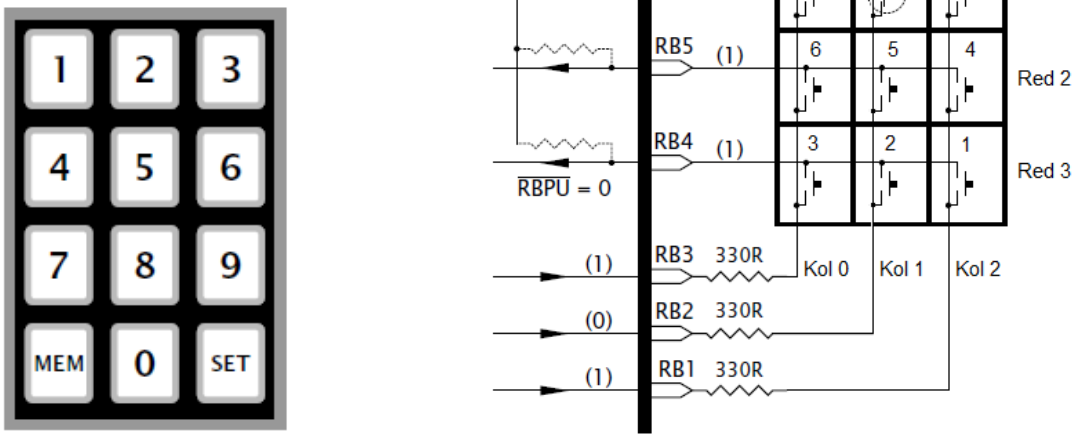
režima ako je bit INTE kontrolnog registra prekida INTCON bio setovan pre odlaska MCU u sleep stanje dok će od statusa GIE bita zavisiti hoće li program granati na adresu prekidnog vektora ili ne. Jedan od najprostijih načina buđenja MCU iz sleep stanja opisanim prekidnim mehanizmom je npr. pritiskanjem ili otpuštanjem tastera povezanog na liniju RB0 pri omogućenom pull-up otporniku na ovoj ulaznoj liniji porta B.

Digitalna logika linija višeg nibla porta B je unekoliko izmenjena u odnosu na niži, slika 5.2.2. Kao što se može primetiti dodatni D flip-flop, u odnosu na sliku 5.2.1., sa kombinacionom mrežom čini digitalnu logiku prekidnog sistema svake od četiri linija višeg nibla porta B. Dodatni D flip-flop radi u paraleli sa ulaznim prihvatnim D flip-flopom ali u protiv fazi. Kada CPU čita port B ulazni prihvatni D flip-flop uzorkuje stanje ulazne linije porta dok u isto vreme dodatni leč flip-flop postaje transparentan. Kada se ciklus čitanja završi dodatni D leč zamrzava i kaptira stanje na ulaznoj liniji kao u vreme čitanja. Evidentno je da se stanje na izlazu prihvatnog leč flip-flopa poredi pomoću EXOR kola sa stanjem na izlazu dodatnog D flip-flopa. Samo u slučaju neslaganja stanja na izlazima oba flip-flopa, generiše se prekidni zahtev, odnosno, setuje zastavica RBIF i to ako neslaganje postoji bar na jednoj ili više ulaznih linija višeg nibla porta B istovremeno.

I ova vrsta spoljašnjeg prekida može biti iskorišćena za buđenje MCU iz režima spavanja (sleep). U prekidnoj servisnoj rutini korisnik može eliminisati uzrok prekida na sledeći način:

- čitanjem ili upisom u port B što će izjednačiti stanja na izlazima oba opisana D leč flip-flopa i time eliminisati neslaganje na jednoj ili više ulaznih linija višeg nibla porta B i potom
- programskim resetovanjem indikatorske zastavice prekida RBIF u INTCON registru.

Ova vrsta prekida preporučuje se za buđenje MCU iz sleep stanja pritiskanjem ili otpuštanjem tastera priključenog na jednu od linija višeg nibla porta B. Ne preporučuje se rad sa portom B metodom programske prozivke ako se koristi opisani prekidni mehanizam. Takođe, opisani prekidni mehanizam zajedno sa programski konfigurabilnim pull-up otpornicima dopuštaju mogućnost lakog povezivanja porta B sa matričnom tastaturom i jednostavno dekodovanje pritisnutog tastera u prekidnoj rutini.



a) b)

Slika 5.2.3. *a) Izgled matrične tastature 4x3, b) povezivanje tastature sa portom B.*

Na slikama 5.2.3. a) i b) prikazani su frontalni izgled matrične tastature 4x3 i način njenog povezivanja sa portom B MCU, respektivno. Tako organizovan dvodimenzionalni niz od dvanaest tastera smanjuje broj priključnih I/O linija tastature sa MCU na svega sedam, dok bi npr. matrična tastatura veličine 8x8 sa 64 tastera za priključivanje zahtevala samo šesnaest I/O linija. Pri normalno otvorenim kontaktima tastera svi redovi su, zbog priključenih pull-up otpornika, na visokom potencijalu (stanje logičke jedinice) bez obzira na logičko stanje kolona. Skeniranje tastature je proces ispitivanja logičkih stanja sva četiri reda (stanje logičke jedinice ili nule) dok je jedna od tri kolona u stanju logičke nule (niski naponski nivo) a preostale dve u stanju logičke jedinice (visoki naponski nivo). Ponavljanjem postupka za svaku od tri kolona uzastopno i uz primenu odgovarajućeg algoritma, utvrđuje se redni broj pritisnutog tastera. Samo pritisnuti taster povezan sa odgovarajućom kolonom, koja se nalazi u stanju logičke nule, dovodi odgovarajući red u stanje logičke nule, takođe, što se može detektovati ispitivanjem logičkih stanja redova. Otpornici od 330Ω ograničiće struju kroz odgovarajuće prekidače ako jedna ili više linija višeg nibla porta B slučajno dođe u stanje logičke nule zbog npr. loše projektovanog softvera.

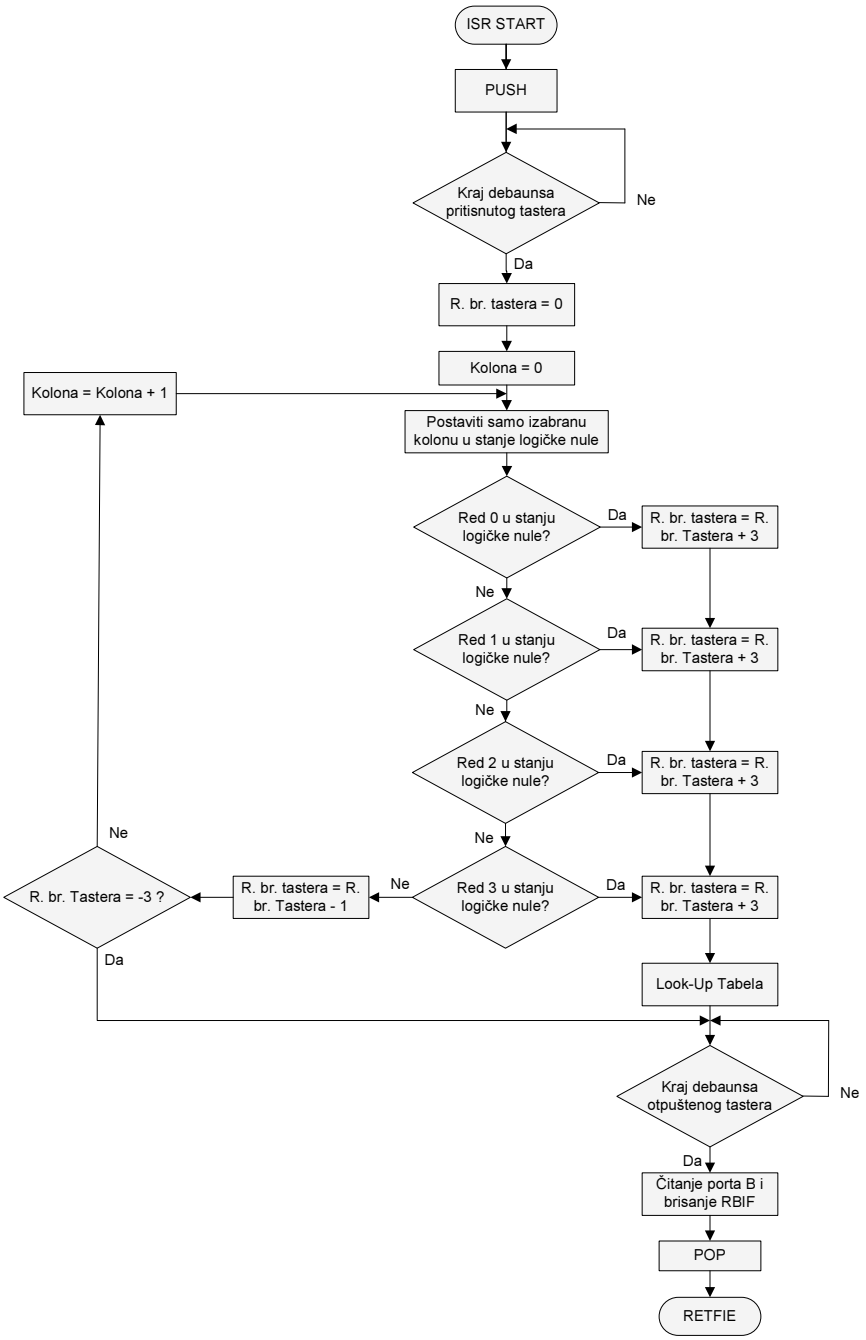
Ako je omogućen prekidni mehanizam višeg nibla porta B tada pritisak na bilo koji taster uzrokuje skok na adresu prekidnog vektora. Obrada prekida i dekodovanje pritisnutog tastera vrši se u prekidnoj rutini prema algoritmu prikazanom na slici 5.2.4. Kao što se može videti, skok na prekidni vektor praćen je najpre pozivom PUSH funkcije za spašavanje konteksta prekinutog procesa pa potom debaunsiranjem pritisnutog tastera. Kratak prelazni režim koji se manifestuje kao višestruko odskakanje realnog mehaničkog kontakta pritisnutog tastera, što je jednako višestrukom pritiskanju/otpuštanju, mora biti premošćen kako bi se čitanje vršilo u stacionarnom stanju. Opisani prelazni režim postoji kako prilikom pritiska tako i pri otpuštanju tastera a njegovo trajanje u funkciji je konstruktivnog rešenja i materijala za realizaciju tastera. Npr. grafitni tasteri imaju loše karakteristike po ovom pitanju ali i nisku cenu dok su tkzv. 'klik' tasteri najpovoljniji. Očigledno je da se čitanje stanja tastera ne sme vršiti u toku trajanja prelaznog režima jer bi moglo biti pogrešno. Jedan od prostih načina rešavanja problema odskakanja kontakata tastera (eng. debouncing) je tajming, odnosno, generisanje dovoljno duge vremenske pauze od trenutka detekcije prekida tako da se prelazni režim sigurno završi i potom čitanje stabilnog stanja tastera. Za većinu od mnogobrojnih vrsta komercijalno raspoloživih tastera maksimalno trajanje prelaznog režima je poznato i kreće se u opsegu od nekoliko ms do nekoliko desetina ms. Tajming metoda debaunsiranja je prosta ali ne i najpouzdanija budući da se trajanje prelaznog režima menja sa vremenom upotrebe tastera, zbog habanja mehaničkih kontakata. Digitalno filtriranje signala dobijenog sa priključka tastera je pogodniji način za eliminaciju fenomena odskakanja kontakata pa se danas uglavnom primenjuje ovo rešenje u programskoj formi.

Kada se debaunsiranje završi nastavlja se sa skeniranjem matrične tastature kako bi se utvrdio redni broj pritisnutog tastera (od 1 do 12). Skeniranje počinje dodelom početne vrednosti tri promenljivoj 'redni broj tastera' i potom nastavlja izborom nulte kolone. Izabrana kolona se postavlja na niski naponski nivo dok su preostale dve na visokom. Za tako izabranu kombinaciju na kolonama proverava se koji od četiri reda je na niskom potencijalu, odnosno, koji od četiri tastera koji pripadaju koloni 0 (tasteri sa rednim brojevima 3, 6, 9, 12) je pritisnut i u skladu sa time izračunava redni broj pritisnutog tastera. Npr. ako je pritisnut taster sa rednim brojem 6, tada će red 2 biti na niskom potencijalu a algoritam pravilno izračunati redni broj tastera.

Ako algoritam utvrdi da niti jedan od četiri reda nije na niskom potencijalu, prelazi na selekciju kolone 1 spuštajući je na niski naponski nivo dok su preostale dve na visokom. Prethodno, algoritam dekrementira početnu vrednost promenljive 'redni broj tastera' tako da sada iznosi dva. Ponavlja se postupak provere koji od četiri reda je na niskom potencijalu i u skladu sa time izračunava redni broj pritisnutog tastera.

Ako i u ovom slučaju algoritam ne pronađe niti jedan red na niskom potencijalu prelazi na poslednji korak skeniranja tastature postavljajući kolonu 2 na niski potencijal a preostale dve na visoki. Proverom redova utvrdiće se koji od poslednja četiri tastera, koji pripadaju koloni 2, je pritisnut.

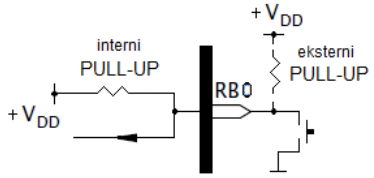
Na slici 5.2.3 b) prikazana je situacija u kojoj je pritisnut taster sa rednim brojem 8 koji pripada koloni 1. Algoritam bi za slučaj kada je kolona 1 na niskom potencijalu utvrdio da je i red 1 na niskom potencijalu i u skladu s tim tačno izračunao redni broj tastera. Izračunatim rednim brojem pritisnutog tastera adresira se potom look-up tabela i iz nje čita odgovarajući karakter pridružen tasteru. U konkretnom slučaju to je karakter 5 koji odgovara tasteru sa rednim brojem osam. Pri kraju, proverava se da li je taster otpušten i ako jeste vrši debaunsiranje a potom eliminiše uzrok prekida čitanjem porta B i resetovanjem zastavice prekida RBIF. Izlasku iz prekidne rutine prethodi rekonstrukcija konteksta prekinutog procesa pozivom funkcije POP i izlazak sa globalno omogućenim prekidom.



Slika 5.2.4. *Algoritam dekodovanja pritisnutog tastera matrične tastature 4x3 u prekidnoj rutini*.

Opisani algoritam za obradu prekidnog zahteva sa matrične tastature 4x3 zasnovan je na ugrađenoj prekidnoj logici višeg nibla porta B i realizovan kao prekidna servisna rutina.

Na slici 5.2.5. prikazan je način povezivanja tastera na prekidnu liniju porta B RB0, koja je konfigurisana kao digitalni ulaz, koristeći unutrašnji pull-up otpornik. Umesto unutrašnjeg moguće je priključiti spoljašnji pull-up otpornik (isprekidano na slici 5.2.5.) čiji bi gornji kraj bio povezan na napon napajanja V_{DD}. U tom slučaju bi programski trebalo onemogućiti unutrašnje pull-up otpornike. Budući da je ova linija porta B izvor prekidnog signala kada je konfigurisana kao ulazna i da se prekid događa na pojavu uzlazne ili silazne ivice naponskog signala na linji RB0, pritisak na taster može biti iskorišćen za ulazak u prekidnu servisnu rutinu gde bi se vršila obrada, npr. odbrojavanje.



Slika 5.2.5. *Povezivanje tastera sa prekidnom linijom RB0 koristeći unutrašnji pull-up otpornik.*

Za taster povezan kao na slici 5.2.5. primetno je da se pritiskom na taster generiše opadajuća (silazna) ivica naponskog signala pa se s tim u vezi linija RB0 mora programski konfigurisati za generisanje prekida na opadajućoj ivici impulsa.

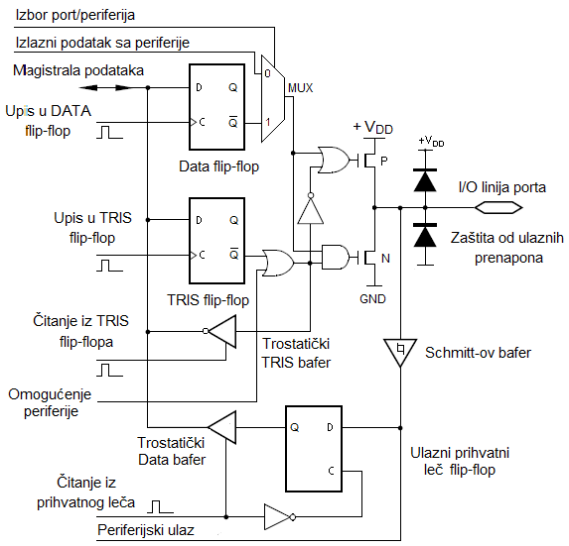
U primeru C koda, tabela 5.2.1., prikazana je i komentarisana glavna funkcija kojom se konfiguriše linija porta B RB0 za prekid na opadajućoj ivici impulsa, omogućuje globalni i odgovarajući individualni prekid i potom ulazi u beskonačnu petlju u isčekivanju prekidnog zahteva. Pritisak na taster povezan sa linijom RB0 uzrokovaće grananje programa glavne funkcije na prekidnu servisnu rutinu u kojoj se nakon debaunsiranja tastera vrši inkrementiranje brojačke promenljive. Brojačka promenljiva odbrojava broj pritisaka na taster.

```
//Primer programa koji pri svakom pritisku na taster linije RB0 grana na prekidnu
//servisnu rutinu, inkrementirajući odgovarajuću promenljivu za odbrojavanje.

#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT
#use delay(clock=20000000)
  long value;               // globalna 16-bitna promenljiva

#INT_EXT                    // Prekidna funkcija za obradu prekida sa RB0 linije
void Brojac_isr(void)
{
   delay_ms(30);            // debaunsiranje tastera metodom tajminga
   value++;                 // inkrementiranje promenljive za brojanje događaja
   while(!input(PIN_B0));   // sačekaj do otpuštanja tastera (RB0 na visokom potenc.)
   delay_ms(30);            // debaunsiranje tastera metodom tajminga
}

void main(void)
{
   set_tris_b(0x01);         // Sve linije porta B izlazne, RB0 ulazna
   port_b_pullups(TRUE);     // Omogućenje pull-up otpornika na ulaznoj RB0 liniji
   ext_int_edge(H_TO_L);           // prekid na opadajućoj ivici impulsa na RB0 lin.
   enable_interrupts(INT_EXT);     // omogućenje individualnog prekida na RB0 liniji
   enable_interrupts(GLOBAL);      // globalno omogućenje svih nemaskiranih prekida
   value=0;                        // inicijalizacija brojačke promenljive
   while(TRUE);                    // beskonačna petlja
}
```

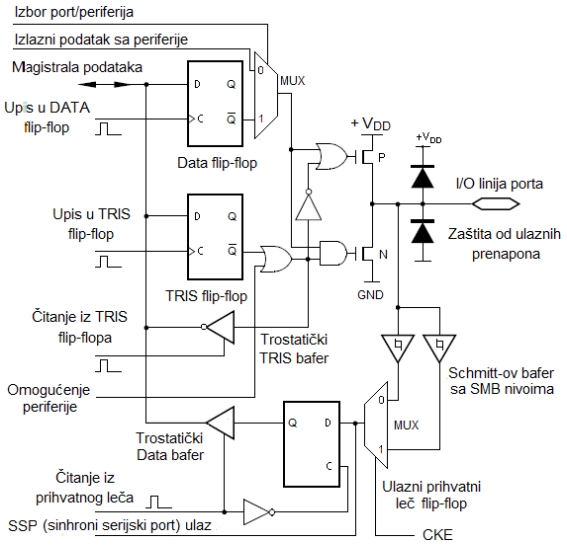
Tabela 5.2.1. *Primer C programa sa prekidnom funkcijom za obradu prekida sa linije RB0.*

5.3. ORGANIZACIJA I MULTIPLEKSNE FUNKCIJE PORTA C

Najkompleksniji port PIC16F877 MCU, sa stanovišta multifunkcionalnosti njegovih I/O linija, je svakako port C. I ovom osmobitnom portu pridružen je osmobitni direkcioni TRIS C registar čiji sadržaj određuje smer prenosa podataka pojedinih I/O linija porta na poznati i opisani način.



Slika 5.3.1. *Digitalna logika linija porta C RC<2:0> i RC<7:5>.*



Slika 5.3.2. *Digitalna logika linija porta C RC<4:3>.*

Na slici 5.3.1. prikazana je digitalna logika šest linija osmobitnog porta C. Sve linije su multipleksirane funkcijama odgovarajućih periferija kao što su: tajmer 1, CCP (*capture/compare/pwm*), USART (*universal synchronous/asynchronous receiver/transmiter*) itd. Slika 5.3.2. prikazuje izgled digitalne logike dve linije porta C RC<4:3>, koja se neznatno razlikuje od ostalih linija. Ove dve linije predstavljaju ujedno i linije sinhronog serijskog porta (SSP) sa I^2C (*inter integrated circuit*) i SPI (*serial peripheral interface*) serijskim komunikacionim interfejsima. Svih osam linija opremljeno je Schmitt-ovim baferima na ulazima.

Dve linije porta C RC<4:3> opremljene su dodatnim Schmitt-ovim baferima, slika 5.3.2., čiji su izlazni naponski nivoi usklađeni sa standardima za serijski prenos informacija po SMB (*System Management Bus*) magistrali.

Pri omogućenju periferijskih funkcija posebnu pažnju treba obratiti na smerove određenih I/O linija porta C, definisanih u direkcionom TRIS C registru. Naime, neke periferije ignorišu odgovarajuće bitove direkcionog TRIS C registra kojima se odgovarajuće linije periferije konfigurišu kao izlazne ili ulazne, postavljajući automatske vrednosti. Ignorisanje odgovarajućih bitova TRIS C registra događa se u vremenu kada se vrši omogućenje određene periferijske funkcije pa s tim u vezi treba izbegavati instrukcije sa TRIS C registrom kao destinacionim.

Mnogobrojni primeri poziva ugrađenih C funkcija za rad sa svim portovima MCU, kao i individualnim linijama portova dati su u tabeli 5.3.1.

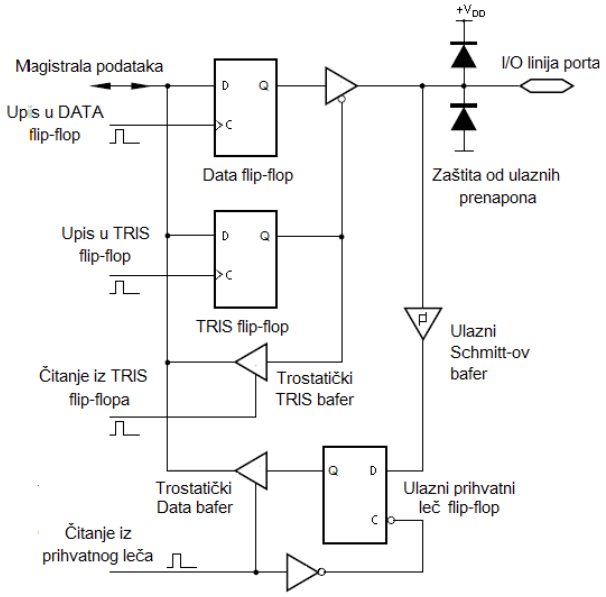
```
//primeri ugrađenih c funkcija za rad sa portovima i individualnim linijama portova

set_tris_b( 0xff );        //sve linije porta B konfigurisane su kao digitalni ulazi
port_b_pullups (true);     //uključeni pull-up otpornici na ulaznim linijama porta B
value = input_c();         //čitanje porta C
set_tris_b( 0x00 );        //sve linije porta B konfigurisane su kao digitalni izlazi
output_b(value);           //upis bajta (value) na port B

output_low(PIN_C0);        //postavljanje linije RC0 na niski naponski nivo
output_high(PIN_A1);       //postavljanje linije RA1 na visoki naponski nivo
output_bit(PIN_B1, 0);     //postavljanje linije RB1 na niski naponski nivo

output_float(pin_D2);      //linija RD2 konfigurisana kao digitalni ulaz
output_drive(pin_A3);      //linija RA3 konfigurisana kao digitalni izlaz
input(PIN_B4);             //čitanje stanja linije RB4 (0-nisko, 1-visoko)
```

Tabela 5.3.1. *Ugrađene C funkcije CCS C kompajlera za rad sa portovima i linijama portova.*

5.4. ORGANIZACIJA I MULTIPLEKSNE FUNKCIJE PORTA D

Port D je osmobitni port sa Schmitt-ovim trigerskim baferima na svim ulazima. Svaka linija ovog porta individualno može biti konfigurisana kao digitalna ulazna/izlazna linija opšte namene. Na slici 5.4.1. prikazana je digitalna logika svih linija porta D kada su konfigurisane kao digitalne I/O linije opšte namene.

Port D se može konfigurisati kao jedan osmobitni mikroprocesorski port PSP (*Parallel Slave Port*) setovanjem kontrolnog bita PSPMODE TRISE<4> registra. Ulazni baferi u ovom režimu rada su TTL tipa.



Slika 5.4.1. *Digitalna logika linija porta D u I/O režimu rada.*

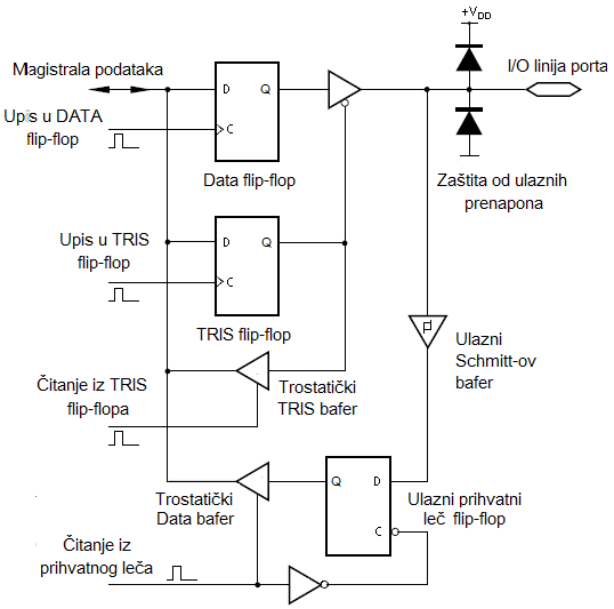
5.5. ORGANIZACIJA I MULTIPLEKSNE FUNKCIJE PORTA E

Port E je trobitni port sa Schmitt-ovim trigerskim baferima ugrađenim na sva tri ulaza kada su linije porta konfigurisane kao digitalni ulazi/izlazi opšte namene, slika 5.5.1. Sve tri linije porta E mogu biti konfigurisane kao kontrolni digitalni ulazi mikroprocesorskog porta (PSP), setovanjem bita PSPMODE TRISE<4> registra i setovanjem tri najniža bita TRISE<2:0> registra, vidi tabelu 5.5.1. U ovom radnom režimu ulazni baferi su TTL tipa. Linije porta E multipleksirane su analognim ulazima A/D modula kao MCU periferije. Da bi se omogućila funkcija analognog ulaza svake linije porta E potrebno je programirati direkcioni TRIS E registar i linije konfigurisati kao ulazne a potom programirati kontrolni registar A/D modula ADCON1 tako da se linije konfigurišu kao analogni ulazi. Posle POR reseta tri linije porta E automatski se konfigurišu kao analogni ulazi a čitanje linija rezultira čitanjem logičkih nula.

TRISE REGISTAR

	R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
	IBF	**OBF**	**IBOV**	**PSPMODE**	**-**	**Bit2**	**Bit1**	**Bit0**
	bit 7							bit 0
bit 7	**IBF:** Statusni bit koji opisuje stanje ulaznog bafera PSP. 1 = Jedan bajt je primljen i čeka na ciklus čitanja od strane CPU. 0 = Nema primljenog bajta.							
bit 6	**OBF**: Statusni bit koji opisuje stanje izlaznog bafera PSP. 1 = Izlazni bafer porta D sadrži upisani bajt podatka. 0 = Izlazni bafer porta D je pročitan od strane mastera.							
bit 5	**IBOV**: Bit za detekciju prekoračenja pri upisu u ulazni bafer. 1 = Dogodio se upis bajta u PSP a da prethodno upisani nije pročitan (bit IBOV se mora brisati softverski). 0 = Nema prekoračenja.							
bit 4	**PSPMODE**: Bit za izbor radnog režima PSP. 1 = PORTD funkcioniše kao PSP. 0 = PORTD funkcioniše kao digitalni I/O port opšte namene.							
bit 3	**Neimplementiran**: Čita se kao logička nula.							
bit 2	**Bit2**: Kontrolni bit za izbor direkcije linije RE2. 1 = Ulazna. 0 = Izlazna.							
bit 1	**Bit1**: Kontrolni bit za izbor direkcije linije RE1. 1 = Ulazna. 0 = Izlazna.							
bit 0	**Bit0**: Kontrolni bit za izbor direkcije linije RE0. 1 = Ulazna. 0 = Izlazna.							

Tabela 5.5.1. *Sadržaj direkcionog registra porta E, TRISE.*



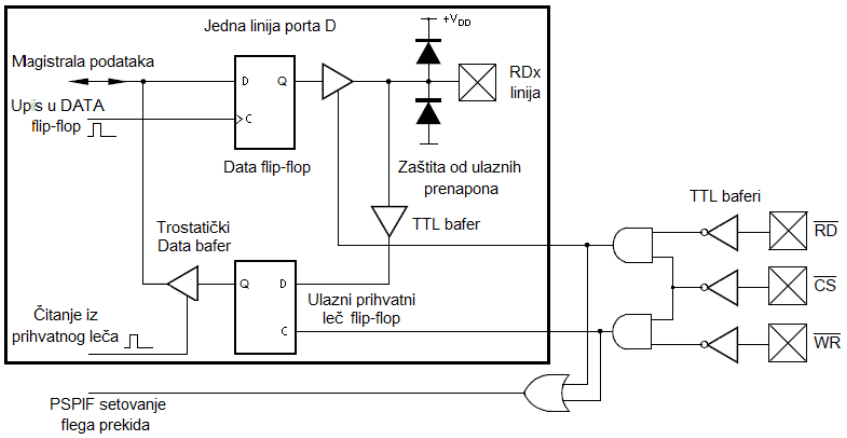
Slika 5.5.1. *Digitalna logika linija porta E u I/O režimu rada.*

5.6. PARALELNI SLAVE PORT

Već je rečeno da PORTD radi kao jedan osmobitni paralelni slave port ili mikroprocesorski port kada je kontrolni bit PSPMODE registra TRISE<4> setovan. U slave modu ovaj port se ponaša kao asinhroni R/W port, gde se operacije upisa ili čitanja jednog bajta podatka kontrolišu spolja preko kontrolnih ulaznih linija porta E. Operaciju upisa $\overline{WR}$ podatka sa linija porta D u leč registar porta D kontroliše linija RE1/$\overline{WR}$/AN6, dok operaciju čitanja $\overline{RD}$ iz leč registra porta D i prenos podatka na linije porta D kontroliše linija RE0/$\overline{RD}$/AN5. PSP može direktno biti povezan na osmobitnu mikroprocesorsku magistralu podataka tako da spoljašnji MCU može čitati iz ili upisivati u osmobitni leč porta D.

Setovanjem bita PSPMODE TRISE<4> registra linija porta E RE0/$\overline{RD}$/AN5 postaje ulaz za kontrolu operacije čitanja porta D $\overline{RD}$, linija porta E RE1/$\overline{WR}$/AN6 postaje ulaz za kontrolu operacije upisa u leč registar porta D $\overline{WR}$, dok linija porta E RE2/$\overline{CS}$/AN7 postaje ulaz $\overline{CS}$ (*Chip Select*) kojim se omogućava/onemogućava rad paralelnog slave porta, slika 5.6.1. Da bi se ova funkcionalnost omogućila konfiguracioni bitovi kontrolnog registra ADCON1<3:0> (PCFG3:PCFG0) moraju biti setovani za konfigurisanje linija porta E PORTE<2:0> kao digitalnih I/O linija. Takođe, odgovarajući bitovi direkcionog TRIS registra TRISE<2:0> moraju biti setovani čime pomenute linije porta E postaju digitalni ulazi.

Port D efektivno sadrži dva 8-bitna leč registra: jedan za ulazne podatke i drugi za izlazne. Bajt podatka upisuje se u leč porta D ili čita sa lečevanih linija porta D. U ovom radnom režimu direkcioni registar porta D TRISD se ignoriše, budući da isključivo spoljašnji uređaj kontroliše smer toka podataka, slika 5.6.1.

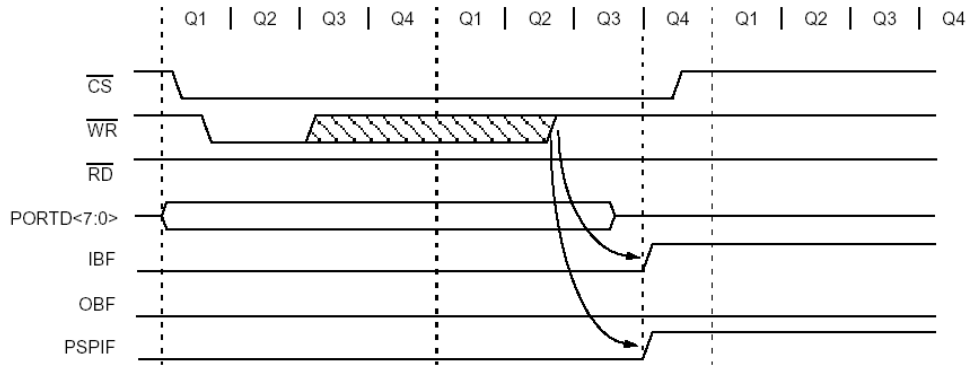


Slika 5.6.1. *Blok dijagram porta D i porta E u PSP režimu rada.*

Upis jednog bajta u PSP događa se kada se prvi put detektuje niski naponski nivo na obe kontrolne linije porta E, $\overline{CS}$ i $\overline{WR}$ slika 5.6.2. Kada bilo koja od ove dve linije promeni stanje i pređe na visoki naponski nivo, IBF (*input buffer full*) statusni bit TRIS E registra će se setovati u poslednjem, četvrtom mašinskom ciklusu Q_4 ako je promena stanja nastala pre kraja mašinskog ciklusa Q_2, kao na slici 5.6.2. U suprotnom, IBF bit će se setovati u četvrtom mašinskom ciklusu Q_4 sledećeg instrukcijskog ciklusa. Setovanje IBF statusnog bita signalizira kompletiranje operacije upisa bajta u PSP. Prekidni flag bit PSPIF PIR1 registra PIR1<7> se takođe setuje u istom Q_4 mašinskom ciklusu kao i bit IBF.

IBF bit se briše jedino čitanjem ulaznog leč registra porta D.

IBOV (*input buffer overflow*) flag bit TRIS E registra TRISE<5> setuje se kada se pokuša sa novim upisom u PSP a da prethodno upisani bajt nije pročitan.



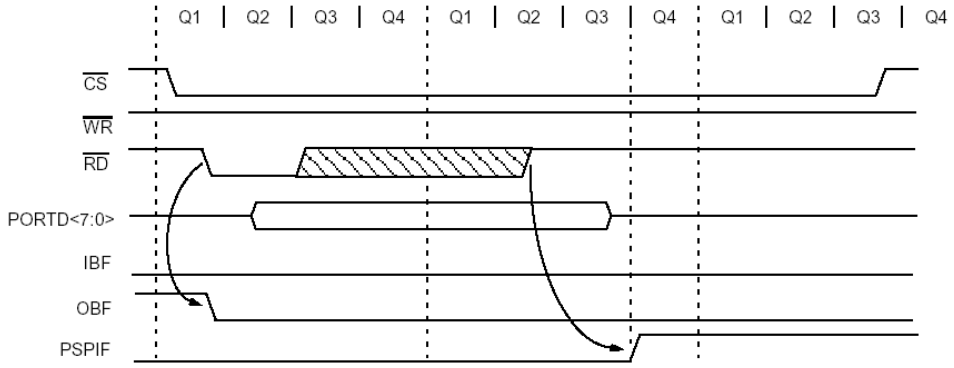
Slika 5.6.2. *Vremenski dijagram ciklusa upisa u PSP.*

Čitanje jednog bajta iz PSP događa se kada se prvi put detektuje niski naponski nivo na obe kontrolne linije porta E, $\overline{CS}$ i $\overline{RD}$, slika 5.6.3. OBF (*output buffer full*) statusni bit TRIS E registra TRISE<6> se istovremeno resetuje, signalizirajući na taj način da je izlazni bafer porta D spreman za čitanje bajta podatka sa spoljašnje magistrale. Kada bilo koja od ove dve kontrolne linije, $\overline{CS}$ ili $\overline{RD}$, promeni stanje i pređe na visoki naponski nivo, prekidni flag bit PSPIF PIR1 registra PIR1<7> će se setovati u poslednjem, četvrtom mašinskom ciklusu Q_4 ako je promena stanja nastala pre kraja mašinskog ciklusa Q_2, kao na slici 5.6.3. U suprotnom, PSPIF flag će se setovati u četvrtom mašinskom ciklusu Q_4 sledećeg instrukcijskog ciklusa.

Setovanje PSPIF bita signalizira kompletiranje operacije čitanja bajta iz PSP. OBF (*output buffer full*) statusni bit ostaje resetovan do novog upisa u PORTD.

Kada PORTD nije konfigurisan kao PSP, IBF i OBF bitovi ostaju u stanju reset. Međutim, ako je IBOV statusni bit TRIS E registra, TRISE<5>, prethodno bio setovan on se mora programski resetovati.

Kao što se može videti sa slika 5.6.2. i 5.6.3. prekidni signal se generiše kada se ciklus upisa ili čitanja PSP kompletira, setujući u oba slučaja bit PSPIF. Ovaj bit mora biti programski resetovan nakon generisanja prekida. Prekid se može onemogućiti resetovanjem bita za maskiranje prekida PSPIE (*parallel slave port interrupt enable*) registra PIE1<7>.



Slika 5.6.3. *Vremenski dijagram ciklusa čitanja iz PSP.*

U tabeli 5.6.1. je dat primer koda na C jeziku za programiranje PSP porta u režimu upisa i čitanja.

```
//Primer C koda za upis i čitanje PSP porta sa odgovarajućim ugrađenim funkcijama

setup_psp(PSP_ENABLED);      //Omogućenje porta D kao PSP porta
 while(psp_output_full());   //čekati dok je izlazni bafer PSP pun
 psp_data=command;           //upis u PSP

 while(!psp_input_full());   //čekati dok je ulazni bafer PSP prazan
 if ( psp_overflow() )
      error=true             //ako postoji pokušaj novog upisa a da prethodni bajt
                             //nije pročitan setuj error flag
 else
      data=psp_data;         //ako ne, pročitaj PSP
```

Tabela 5.6.1. *Primer C koda za upis i čitanje PSP porta primenom odgovarajućih ugrađenih funkcija.*

Poglavlje 6

INTEGRISANI PERIFERIJSKI PODSISTEMI MCU PIC16F877

Povećana funkcionalnost PIC16F877 mikrokontrolera postignuta je integracijom većeg broja periferijskih jedinica različite namene na jedinstvenom supstratu kao što su:

- Tajmer0: 8-bitni tajmer/brojač sa 8-bitnim preskalerom,
- Tajmer1: 16-bitni tajmer/brojač sa preskalerom, koji može biti inkrementiran u sleep režimu spoljašnjim kristalnim oscilatorom ili spoljašnjim taktnim impulsima,
- Tajmer2: 8-bitni tajmer sa 8-bitnim period registrom, preskalerom i postskalerom,
- Dva CCP (*Capture-Compare-PWM*) modula:
  - Modul za hvatanje (*Capture*) je 16-bitni sa maksimalnom rezolucijom od 12.5ns,
  - Modul za poređenje (*Compare*) je 16-bitni sa maksimalnom rezolucijom od 200ns,
  - Impulsno-širinski modulator (*PWM-Pulse Wide Modulation*) sa maksimalnom rezolucijom od 10 bita.
- 10-bitni višekanalni A/D konvertor,
- Sinhroni serijski port (*SSP*) sa SPI (Master mod) i I^2C (Master/Slave mod) serijskim komunikacionim interfejsima,
- Univerzalni sinhrono asinhroni prijemnik predajnik (*USART*) sa detekcijom 9-bitne adrese,
- 8-bitni paralelni slave port (*PSP*) sa spoljašnjom kontrolom $\overline{\text{RD}}$, $\overline{\text{WR}}$ i $\overline{\text{CS}}$ ulaza,
- EEPROM memorija za podatke kapaciteta 256 bajta.

Integracijom nabrojanih periferijskih jedinica omogućena je primena MCU u širokom spektru aplikacija od najprostijih do sofisticiranih, zahtevnih, višefunkcionalnih, bez povećanih troškova realizacije i uz povećanu pouzdanost sistema.

Svi nabrojani periferijski moduli imaju ugrađene prekidne mehanizme kojima se procesor MCU obaveštava o kraju odgovarajuće operacije pod kontrolom određenog modula. Bitovi za maskiranje/demaskiranje individualnih prekida periferijskih modula nalaze se u kontrolnim registrima PIE1 i PIE2, dok registri PIR1 i PIR2 sadrže odgovarajuće zastavice prekida periferijskih jedinica. Tabele 6.1. i 6.2. opisuju sadržaj registara za kontrolu prekida PIE1 i PIE2.

PIE1 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
	PSPIE	**ADIE**	**RCIE**	**TXIE**	**SSPIE**	**CCP1IE**	**TMR2IE**	**TMR1IE**
	bit 7							bit 0
bit 7	**PSPIE:** Bit za maskiranje prekida sa paralelnog slave porta (PSP) porta pri operaciji upisa/čitanja. 1 = Omogućen prekid pri upisu/čitanju PSP porta. 0 = Onemogućen prekid.							
bit 6	**ADIE**: Bit za maskiranje prekida sa A/D konvertora. 1 = Omogućen prekid sa A/D. 0 = Onemogućen prekid sa A/D.							
bit 5	**RCIE**: Bit za maskiranje prekida sa USART prijemnika. 1 = Omogućen prekid. 0 = Onemogućen prekid.							
bit 4	**TXIE**: Bit za maskiranje prekida sa USART predajnika. 1 = Omogućen prekid. 0 = Onemogućen prekid.							
bit 3	**SSPIE**: Bit za maskiranje prekida sa sinhronog serijskog porta (SSP). 1 = Omogućen SSP prekid. 0 = Onemogućen SSP prekid.							
bit 2	**CCP1IE**: Bit za maskiranje prekida sa CCP1 periferije. 1 = Omogućen CCP1 prekid. 0 = Omogućen CCP1 prekid.							
bit 1	**TMR2IE**: Bit za maskiranje prekida sa tajmera 2, pri izjednačenju sadržaja registara TMR2 i PR2. 1 = Omogućen prekid. 0 = Onemogućen prekid.							
bit 0	**TMR1IE**: Bit za maskiranje prekida sa tajmera 1 pri tranziciji FFFFh→0000h. 1 = Omogućen prekid. 0 = Onemogućen prekid.							

Tabela 6.1. *Sadržaj kontrolnog registra PIE1 za maskiranje/demaskiranje prekida sa periferija.*

PIE2 REGISTAR

	U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
	-	**Rezervisan**	**-**	**EEIE**	**BCLIE**	**-**	**-**	**CCP2IE**
	bit 7							bit 0
bit 7	**Neimplementiran:** čita se kao '0'.							
bit 6	**Rezervisan**: Ovaj bit treba uvek držati resetovanim.							
bit 5	**Neimplementiran:** čita se kao '0'.							
bit 4	**EEIE**: Bit za maskiranje prekida sa EEPROM-a pri upisu podatka. 1 = Omogućen prekid. 0 = Onemogućen prekid.							
bit 3	**BCLIE**: Bit za maskiranje prekida pri koliziji na SSP magistrali. 1 = Omogućen prekid. 0 = Onemogućen prekid.							
bit 2-1	**Neimplementirani:** čitaju se kao '0'.							
bit 0	**CCP2IE**: Bit za maskiranje prekida sa CCP2 periferije. 1 = Omogućen CCP2 prekid. 0 = Omogućen CCP2 prekid.							

Tabela 6.2. *Sadržaj kontrolnog registra PIE2 za maskiranje/demaskiranje prekida sa periferija.*

PIR1 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
	PSPIF	**ADIF**	**RCIF**	**TXIF**	**SSPIF**	**CCP1IF**	**TMR2IF**	**TMR1IF**
	bit 7							bit 0
bit 7	**PSPIF:** Zastavica prekida sa PSP porta pri operaciji upisa/čitanja. 1 = Operacija upisa ili čitanja PSP porta završena (mora biti brisan programski). 0 = Nema operacije upisa ili čitanja PSP porta.							
bit 6	**ADIF**: Zastavica prekida sa A/D konvertora. 1 = Kraj A/D konverzije. 0 = A/D konverzija u toku.							
bit 5	**RCIF**: Zastavica prekida sa USART prijemnika. 1 = Prijemni bafer USART-a pun. 0 = Prijemni bafer USART-a prazan.							
bit 4	**TXIF**: Zastavica prekida sa USART predajnika. 1 = Predajni bafer USART-a prazan. 0 = Predajni bafer USART-a pun.							
bit 3	**SSPIF**: Zastavica prekida sa SSP porta. 1 = Jedan od prekidnih uslova SSP se dogodio (mora biti brisan programski). Uslovi koji setuju ovaj bit su: **-SPI mod** Jedna transmisija ili prijem završen. **-I^2C slave mod** Jedna transmisija ili prijem završen. **-I^2C master mod** Jedna transmisija ili prijem završen. Inicirani START uslov kompletiran. Inicirani STOP uslov kompletiran. Inicirani RESTART uslov kompletiran. Inicirani Acknowledge uslov kompletiran. Jedan START uslov se dogodio dok je SSP neaktivan (Multi-Master mod). Jedan STOP uslov se dogodio dok je SSP neaktivan (Multi-Master mod). 0 = Nijedan od prekidnih uslova SSP se nije dogodio.							
bit 2	**CCP1IF**: Zastavica prekida sa CCP1 modula. **Capture mod**. 1 = Sadržaj TMR1 registra prebačen u registar CCP1 modula (mora biti brisan programski). 0 = Nema prenosa sadržaja TMR1 u registar CCP1 modula. **Compare mod** 1 = Sadržaj TMR1 registra izjednačen sa sadržajem regisra CCP1 modula (mora biti brisan programski). 0 = Sadržaj TMR1 registra različit od sadržaja registra CCP1 modula. **PWM mod** Ne koristi se u ovom modu.							
bit 1	**TMR2IF**: Zastavica prekida sa tajmera 2. 1 = Sadržaj TMR1 registra izjednačen sa sadržajem regisra PR2 (mora biti brisan programski). 0 = Sadržaj TMR1 registra različit od sadržaja registra PR2.							
bit 0	**TMR1IF**: Zastavica prekida sa tajmera 1 pri tranziciji FFFFh→0000h. 1 = Prekoračenje osnove brojanja tajmera 1 (mora biti brisan programski). 0 = Nema prekoračenja osnove brojanja tajmera 1.							

Tabela 6.3. *Sadržaj registra PIR1 sa zastavicama prekida određenih periferijskih modula.*

PIR2 REGISTAR

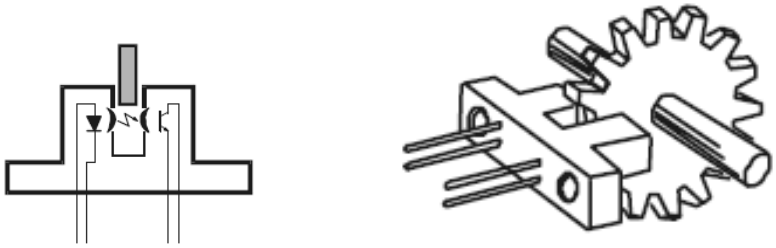
	U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
	-	**Rezervisan**	**-**	**EEIF**	**BCLIF**	**-**	**-**	**CCP2IF**
	bit 7							bit 0
bit 7	**Neimplementiran:** čita se kao ’0’.							
bit 6	**Rezervisan**: Ovaj bit treba uvek držati resetovanim.							
bit 5	**Neimplementiran:** čita se kao ’0’.							
bit 4	**EEIF**: Zastavica prekida pri operaciji upisa u EEPROM. 1 = Ciklus upisa kompletiran (mora biti brisan programski). 0 = Ciklus upisa nije kompletiran ili nije startovan.							
bit 3	**BCLIF**: Zastavica prekida za indikaciju kolizije na SSP magistrali. 1 = Konflikt na SSP portu kada je konfigurisan u I^2C master modu. 0 = Nema konflikta na SSP portu.							
bit 2-1	**Neimplementirani:** čitaju se kao ’0’.							
bit 0	**CCP2IF**: Zastavica prekida sa CCP2 modula. **Capture mod**. 1 = Sadržaj TMR1 registra prebačen u registar CCP2 modula (mora biti brisan programski). 0 = Nema prenosa sadržaja TMR1 u registar CCP2 modula. **Compare mod** 1 = Sadržaj TMR1 registra izjednačen sa sadržajem regisra CCP2 modula (mora biti brisan programski). 0 = Sadržaj TMR1 registra različit od sadržaja registra CCP2 modula. **PWM mod** Ne koristi se u ovom modu.							

Tabela 6.4. *Sadržaj registra PIR2 sa zastavicama prekida određenih periferijskih modula.*

Tabele 6.3. i 6.4. opisuju sadržaj registara PIR1 i PIR2 sa zastavicama prekida svih periferijskih jedinica.

6.1. TAJMERI/BROJAČI

Od krucijalnog značaja za mnoge sisteme predstavlja mogućnost rada u realnom vremenu, kao što je na primer: merenje trajanja nekog događaja, brojanje događaja ili kontrola jednog spoljašnjeg, fizičkog događaja uz poznate periode ponavljanja. Kao primer kontrole, može se navesti merenje vremenskog intervala između dva susedna zuba na nazubljenom rotacionom zamajcu radi kontrole ugaone brzine zamajca, slika 6.1.1.



Slika 6.1.1. *Optički senzor (optocoupler) za merenje ugaone brzine zamajca i način njegove ugradnje.*

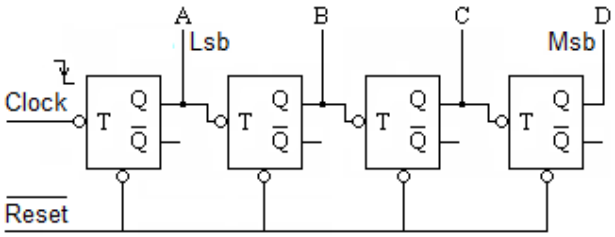
Za precizno merenje vremena uopšte uglavnom se koriste brze sekvencijalne sinhrone mreže, poznate pod nazivom digitalni tajmeri/brojači. Ove komponente se mogu smatrati standardnim periferijskim jedinicama savremenih MCU, različitih rezolucija i brzina. Šta više, moderni MCU objedinjuju više integrisanih brojačkih komponenata različitih rezolucija, čime se značajno širi polje primene i povećava efikasnost sistema uopšte.

Jedan tajmer/brojač karakterišu sledeći parametri:

- Rezolucija
- Stanje brojača
- Osnova brojanja
- Kod brojanja

Rezoluciju brojača predstavlja broj bitova n. Stanje brojača predstavlja stanje njegovih izlaza. U novo stanje brojač prelazi pod dejstvom taktnog impulsa tako da se stanja brojača mogu interpretirati kao niz uzastopnih brojeva. Osnova brojanja je ukupan broj različitih stanja brojača, dok je kod brojanja redosled izmene stanja brojača.

Brojačka sekvencijalna mreža se realizuje od niza ivičnih flip-flopova (flip-flop čija se promena stanja vrši pri kratkotrajnoj pojavi jedne od ivica impulsa) i kombinacionih kola.



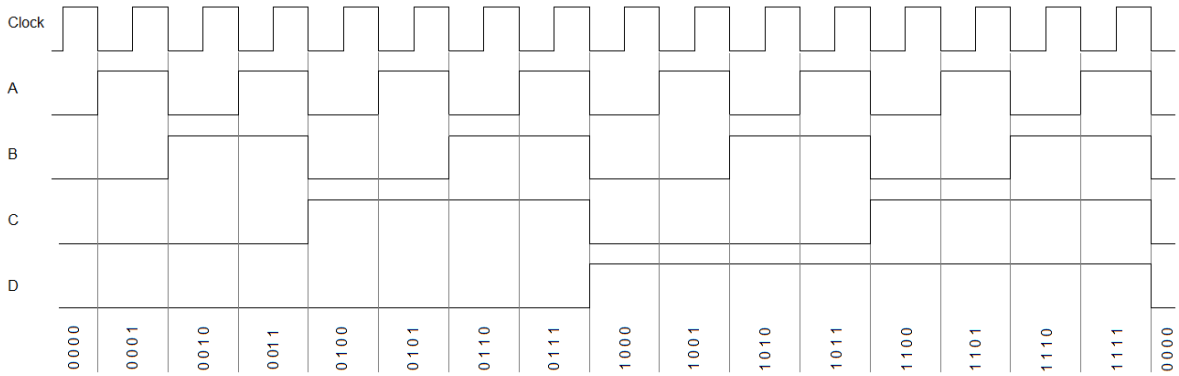
Slika 6.1.2. *Asinhroni četvorobitni binarni brojački registar na bazi T flip-flopova.*

Na slici 6.1.2. prikazana je relativno jednostavna konfiguracija asinhronog (rednog) binarnog brojačkog registra rezolucije 4 bita, realizovanog sa asinhronim ivičnim T flip-flopovima. Brojač broji u težinskom binarnom kodu budući da svaki flip-flop na svom izlazu daje signal dvostruko manje frekvencije u odnosu na frekvenciju signala na svom ulazu (T ulaz).

Prema tome, svaki T flip-flop je u suštini delitelj frekvencije pobudnog signala sa odnosom deljenja dva. Kako izlazni signal jednog stepena predstavlja pobudni za sledeći, to znači da će se na izlazu A (Lsb) generisati signal dvostuko manje frekvencije u odnosu na taktni, na izlazu B četiri puta, izlazu C osam puta i izlazu D (Msb) šesnaest puta manji u odnosu na pobudni taktni signal. Rezolucija brojača može biti proširena dodavanjem novih stepena (razreda) kaskadno na poslednji MSB flip-flop i jednaka je ukupnom broju flip-flopova. Prikazano rešenje brojača primenljivo je za umerene frekvencije taktovanja budući da je maksimalna frekvencija taktovanja limitirana propagacionim kašnjenjem kroz sve flip-flopove u kaskadi. U slučaju većih rezolucija brojača ukupna propagacija je zbir propagacije kroz svaki flip-flop što može ozbiljno degradirati maksimalnu taktnu frekvenciju. Ako se pretpostavi da su propagacije kroz sve flip-flopove jednake, propagacija ukupne konfiguracije je $t_{PU} = N\,t_{PFF}$, gde je *N* broj flip-flopova, pa je maksimalna frekvencija taktnih impulsa data izrazom

$$f_{\max} = \frac{1}{N\,t_{PFF}}$$

Na slici 6.1.3. su prikazani talasni oblici signala na ulazu za taktovanje i pravim izlazima četvorobitnog binarnog brojača za 16 taktnih intervala, odnosno, jedan puni ciklus brojanja. Kao što se može videti, svaki flip-flop se okida opadajućom ivicom signala na svom T ulazu i svaki na svom izlazu generiše signal dvostruko manje frekvencije u odnosu na signal na svom ulazu za trigerovanje T. Početno stanje brojača je 0000 dok je krajnje 1111. Puni ciklus brojanja podrazumeva tranzicije iz jednog u drugo stanje pod uticajem taktnih impulsa, kao na slici 6.1.3, tako da se posle šesnaest taktnih intervala brojač u sedamnaestom ponovo nađe u početnom stanju 0000 nastavljajući sa brojanjem. Brojač se u početno stanje može uvesti resetovanjem brojačkog registra. Za detekciju kraja jednog ciklusa brojanja projektanti ugrađuju digitalnu logiku koja na tranziciji iz stanja 1111 u početno stanje 0000 (pri prekoračenju) generiše impuls kratkog trajanja, koji se u MCU jedinicama koristi kao prekidni zahtev.

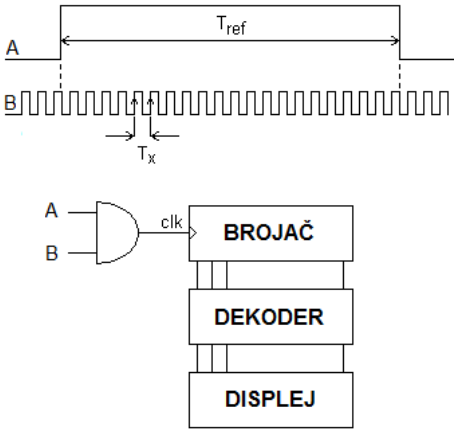


Slika 6.1.3. *Talasni oblici signala na ulazu za taktovanje i pravim izlazima flip-flopova 4-bitnog binarnog brojača i odgovarajuća stanja.*

Tajmeri/brojači su komercijalno zastupljeni kao diskretni sistemi ili integrisani u MCU jedinice, rezolucija osam, šesnaest i tridesetdva bita.

6.2. MERENJA VREMENA I FREKVENCIJE NA BAZI TAJMERA/BROJAČA

Standardne periferijske jedinice kojima su opremljeni svi mikrokontroleri, tajmeri/brojači, veoma su pogodne za precizna merenja vremenskih intervala kao što su širina impulsa i perioda ponavljanja ili precizna merenja broja ciklusa u jedinici vremena-frekvencije. Frekvencija periodičnog ulaznog signala može biti određena tkzv. direktnom mernom metodom, odnosno, prostim brojanjem impulsa ulaznog signala u toku trajanja referentnog vremenskog intervala T_{ref}, koji se može generisati programski ili uz pomoć druge tajmerske jedinice. Na slici 6.2.1. ilustrovan je princip direktne metode merenja frekvencije zasnovan na primeni tajmera/brojača.



Slika 6.2.1. *Principijelna blok šema sistema za merenje frekvencije direktnom mernom metodom na bazi tajmera/brojača.*

Kao što se sa slike 6.2.1. može videti, periodični ulazni signal oblika periodične povorke pravougaonih impulsa, nepoznate periode ponavljanja T_x, dovodi se na jedan ulaz logičkog 'I' kola, ulaz B, čijim se izlazom taktuje brojač. Na drugom ulaznom kraju 'I' kola, ulaz A, priključen je signal striktnog trajanja T_{ref}, oblika kao na slici 6.2.1. Stanje n-bitnog brojača (n-bitni brojački registar) inkrementira se svaki put kada se na njegovom ulazu za taktovanje, ulaz clk, pojavi rastuća ivica impulsa. Prema tome, logičko 'I' kolo (gejt) će za vreme trajanja visokog naponskog nivoa na ulazu A, propuštati impulse nepoznate periode ponavljanja, priključene na ulaz B, do ulaza za taktovanje brojača, clk. Kako je vremenski razmak između dve rastuće ivice impulsa jednak nepoznatoj periodi impulsa T_x, to za vreme trajanja referentnog signala A važi

$$T_{ref} = N T_x ,$$

gde je *N* broj odbrojanih perioda T_x koji pokazuje displej nakon dekodovanja stanja brojačkog registra. Prema tome, iz gornje jednačine sledi da je broj na displeju

$$N = \frac{T_{ref}}{T_x} = T_{ref} f_x ,$$

imajući u vidu da je recipročna vrednost periode jednaka frekvenciji, $f_x = 1/T_x$. Poslednji izraz pokazuje da je broj *N* pročitan sa displeja direktno srazmeran nepoznatoj frekvenciji i striktnom

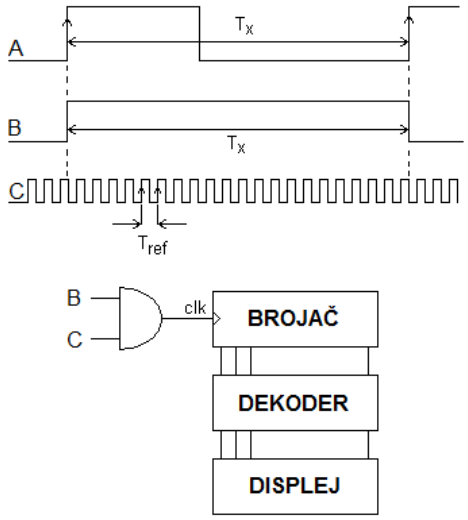
trajanju referentnog vremenskog intervala. Referentni vremenski signal može biti generisan spolja, npr. uz pomoć stabilnog generatora signala, ili češće programski npr. striktnim brojanjem instrukcijskih ciklusa čije trajanje je takođe striktno i poznato.

Budući da se brojač inkrementira jednom od ivica pravougaonih impulsa (npr. rastućom ili opadajućom), jednostavno se može pokazati da je greška merenja u granicama Δ = ±1 impuls, odnosno, Δ = ±1 perioda. Prema tome, pri merenju može doći do pogrešnog odbrojavanja brojača za jedan impuls više ili manje, zbog čega je relativna greška merenja oblika

$$\delta(\%) = \pm\frac{1}{N}100$$

Ova se greška naziva diskretnom greškom merenja i obrnuto je proporcionalna frekvenciji mernog signala. Za male frekvencije ulaznih signala metoda je krajnje nepogodna zbog relativno velike diskretne greške, i obrnuto. Stoga se za relativno male frekvencije ulaznog signala radije koristi tkzv. recipročna metoda merenja frekvencije, odnosno, metoda direktnog merenja periode signala dok se frekvencija preračunava.

Na slici 6.2.2. prikazana je principijelna blok šema sistema za direktno merenje periode ponavljanja signala (recipročna metoda merenja frekvencije) koja bazira na primeni tajmera/brojača.



Slika 6.2.2. *Principijelna blok šema sistema za merenje vremenskog intervala-periode ponavljanja signala na bazi tajmera/brojača.*

Sa slike 6.2.2. se vidi da je princip merne metode identičan prethodno opisanom sa razlikom da se na mesto signala nepoznate periode dovodi signal stabilne referentne periode ponavljanja T_{ref}, odnosno frekvencije $f_{ref} = 1/T_{ref}$, a na mesto signala striktnog trajanja signal nepoznate periode T_x. Konverzija talasnog oblika A u talasni oblik B, koja je neophodna da bi se omogućilo brojanje u toku trajanja cele periode, može se izvršiti jednostavno npr. primenom asinhronog ivičnog T flip-flopa. Na ulaz T flip-flopa za okidanje dovodi se talasni oblik A a na njegovom pravom izlazu dobija talasni oblik B, imajući u vidu da trigerski flip-flop deli frekvenciju ulaznog periodičnog signala sa dva. Pod ovakvim okolnostima može se pisati

$$T_x = NT_{ref},$$

gde je *N* broj odbrojanih perioda referentnog trajanja T_{ref} koji pokazuje displej nakon dekodovanja stanja brojačkog registra. Iz poslednje jednačine sledi da je broj na displeju

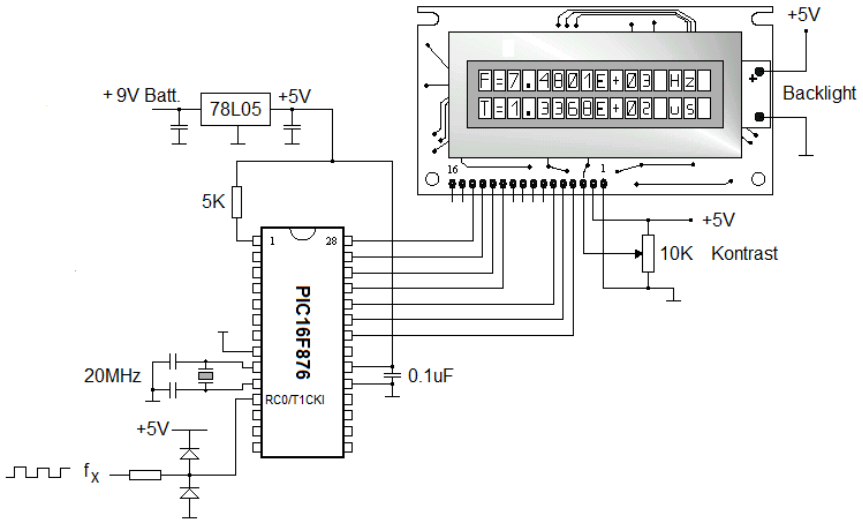
$$N = \frac{T_x}{T_{ref}} = f_{ref}\, T_x = \frac{f_{ref}}{f_x},$$

srazmeran trajanju nepoznate periode T_x i frekvenciji referentnog signala f_{ref}. Kako je diskretna greška merenja i u ovom slučaju $\delta(\%) = \pm\frac{1}{N}100$, primetno je da metoda daje tačniji rezultat merenja pri manjim frekvencijama merenog signala, odnosno, većim periodama ponavljanja. Budući da postoji obrnuta proporcionalnost očitanog broja N sa merenom frekvencijom f_x, to se frekvencija može izračunati na bazi recipročne vrednosti nepoznate periode, po čemu je metoda i dobila ime.

Opisane merne metode kao glavnu jedinicu koriste tajmer/brojač čija rezolucija (broj bitova za kodiranje) ima odlučujući uticaj na tačnost mernog postupka. Greška merenja od ±1 impuls može biti zanemariva na odbrojanih, recimo 10^6 impulsa, dok je maksimalni broj impulsa ograničen rezolucijom brojača i iznosi 2^n za binarne brojače, gde je n broj bitova. Prema tome, ako se osigura dovoljno velika rezolucija brojača i dovoljno velika stabilnost generisanih referentnih intervala vremena, tačnost merenja jednom ili drugom metodom može biti zavidno visoka. Prvi uslov nije teško ispuniti budući da postoje tajmerske komponente rezolucije 32 bita, kao i da rezolucija može biti uvećana i softverski. Uslov stabilnosti generisanih referentnih vremenskih intervala takođe može biti ispunjen jednostavno ako se za generisanje vremenskih intervala koriste oscilatori na bazi kvarc kristala sa temperaturskom kompenzacijom rezonantne frekvencije. Zahvaljujući dugoročno visokoj stabilnosti frekvencije rezonanse kristala kvarca, koja u izvesnoj meri zavisi od temperature, osigurana je visoka stabilnost frekvencije oscilovanja kvarcnog oscilatora.

Zbog svega napred navedenog može se reći da se veličine vremenskog domena, kao što su širina impulsa, faza, perioda ponavljanja i frekvencija mogu meriti sa najvećom tačnošću i bez visokih troškova.

Na slici 6.2.3. prikazano je kolo frekvencmetra/periodmetra zasnovano na 28-pinskom PIC16F876 MCU i integrisanom DMM (*Dot Matrix Module*) dvorednom LCD displeju sa 16 karaktera po redu za vizuelni prikaz frekvencije i periode.



Slika 6.2.3. *Blok šema digitalnog frekvencmetra/periodmetra na bazi PIC MCU.*

```
          /*__ Primer C programa za digitalni frekvencmetar/periodmetar __*/

#include <16F877.h>
#use delay (clock=4000000)           //CPU takt = 4MHz, TCY=1us
#fuses XT,NOWDT,PROTECT,NOLVP,PUT
#include <stdlib.h>
#include "LCD.C"
#bit t1_overflow=0x0C.0              //PIR1,bit0 (TMR1IF)

   int cycles8, cycles;             //Deklaracija promenljivih
   float freq, period;
   long freqc_high, perc_high;
   long freqc_low, perc_low;
                            /*___ F U N K C I J E ___*/

void wait_for_low_to_high()
{
   while(input(PIN_C0)) ; // za visoki nivo signala čekaj do pojave opadajuće ivice
   delay_us(1);           // vreme opadanja
   while(!input(PIN_C0)); // čekaj do pojave rastuće ivice __|--
}

void wait_for_low_to_high_T1Flag() {

while(input(PIN_C0))     // za visoki nivo signala čekaj do pojave opadajuće ivice
{     if (t1_overflow)
      {t1_overflow=0; perc_high++;}
}

delay_us(1);             // vreme opadanja

while(!input(PIN_C0))   // čekaj rastuću ivicu i proveravaj prekoračenje tajmera1
{      if (t1_overflow)
       {t1_overflow=0; perc_high++;}
}
  // obriši flag t1_overflow ako je tajmer1 prekoračio i uvećaj promenljivu perc_high
}
                        /*___ G L A V N A  F U N K C I J A ___*/
void main()
{
      lcd_init();
      lcd_putc('\f');

   while (TRUE) {
      cycles8=0;
      cycles=0;
      freqc_high=0;
      perc_high=0;
      t1_overflow=0;
      set_timer1(0);
      setup_timer_1(T1_EXTERNAL|T1_DIV_BY_1);

              /* ___ Generisanje refrentnog vrem. intervala od 1s ___  */

//start spoljašnje brojačke petlje
      while (cycles!=0xFF) {                  //true=3, false=4
       cycles8=0;                             //1 ciklus
//start unutrašnje brojačke petlje

       while (cycles8!=0xFF) {                //true=3, false=4
         if (t1_overflow)                     //true=2, false=3   //----|
            {t1_overflow=0; freqc_high++;}    //6 ciklusa         //    |__8 ciklusa
         else                                                     //    |
            {delay_cycles(5);}                                    //----|
         delay_cycles(1) ;                  //x=1
```

```
         cycles8++;                          //1

       }                                     //2 ciklusa za skok na vrh unutr. petlje
// kraj unutrašnje brojačke petlje
//(mat: ukupan broj ciklusa unutr. petlje=((3+8+x+1+2)*255 + 4)*255)
//(mat: ako je x=1.363014 ukupan broj ciklusa unutr. petlje je 1 milion)
//(mat: ako je x=1 ukupan broj ciklusa unutr. petlje je 976395, treba dopuniti do
//1000000 što iznosi 23605 ciklusa)

 delay_cycles(85);                    //y=85
 cycles++;                            //1 ciklus
      }                               //2 ciklusa za skok na vrh spolj. petlje
// kraj spoljašnje brojačke petlje
// (mat: ukupan broj ciklusa spolj. petlje=(3+1+y+1+2)*255+4=23605)
// (mat: iz gornje jednacine je y=(23605-4)/255)-(3+1+0+0+1+2))
// (mat: ako je y=85.55294117647 ukupan broj ciklusa spolj. petlje je 23605)
// (mat: ako je y=85 ukupan broj ciklusa spolj. petlje je = 23464, treba dopuniti do
// 23605 sto iznosi 141 ciklus)

      delay_cycles(141);             //z=141

                    /* ___ Kraj generisanja Tref=1 sekunda ___ */

      setup_timer_1(T1_DISABLED);  //TMR1 OFF da bi se procitala prava vrednost
      if (t1_overflow)             //da li je doslo do prekoracenja TMR1?
          freqc_high++;
      freqc_low=get_timer1();                //čitanje 16-bitne vrednosti TMR1
      freq=make32(freqc_high, freqc_low);   //sinteza 32-bitne reči

       /*___ IZBOR: merenje frekvencije direktnom ili reciprocnom metodom ___ */

//___ Do 1000Hz meri se perioda impulsa a frekvencija preračunava F=1/T.
//___ Iznad 1000Hz meri se broj impulsa u striktnom vremenskom intervalu od 1s,
//___ odnosno frekvencija, dok se perioda preračunava T=1/F.
//___ Na taj način je omogućeno merenje frekvencije u opsegu od 0.2328mHz do
//___ 50MHz (granična frekvencija tajmera1) sa približno konstantnom tačnošću.

      if (freq>1000) {
            lcd_putc('\f');
            printf(lcd_putc,"F=%4.4E Hz\n",freq);      //prikaz na displeju
            printf(lcd_putc,"T=%4.4E s\n",1/freq);     //prikaz na displeju
         }
      else
         {
            setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);   // Start timer 1
            wait_for_low_to_high();
            set_timer1(0);
            wait_for_low_to_high_T1Flag();
            setup_timer_1(T1_DISABLED);       //TMR1 OFF da se pročita prava vrednost
            if (t1_overflow)                  //da li je došlo do prekoračenja TMR1?
                perc_high++;
            perc_low=get_timer1();

            period=make32(perc_high, perc_low);  //sinteza 32-bitne reči

            lcd_putc('\f');

            printf(lcd_putc,"T=%4.4E us\n",period);          //prikaz na displeju
            printf(lcd_putc,"F=%4.4E Hz\n",1000000/period);  //prikaz na displeju
          }
   }
}
```

Tabela 6.2.1. *Primer programa na C jeziku za merenje frekvencije/periode periodičnih signala.*

Programski kod na C jeziku, tabela 6.2.1., napisan je tako da kombinuje obe prethodno opisane metode merenja i to: do 1000Hz meri se perioda ponavljanja ulaznih impulsa a frekvencija izračunava kao recipročna vrednost periode dok je za frekvencije iznad 1KHz implementirana direktna metoda merenja frekvencije a perioda se izračunava. Na ovaj način je osigurano merenje frekvencije u širokom opsegu promene od 0.2328mHz ($1/(2^{32}\mu s)$) pa do granične frekvencije taktovanja tajmera1 od 50MHz. Za merenje periode ponavljanja koristi se trajanje instrukcijskog ciklusa T_{CY}=1μs kao vremenska referenca dok se za direktno merenje frekvencije odbrojava 10^6 instrukcijskih ciklusa u dve ugnježdene while strukture kako bi se generisao striktni vremenski interval od 1s. Balansnim kodom za kašnjenje *delay_cycles()* podešen je broj ciklusa tako da bez obzira na programska grananja uvek iznosi striktno 10^6 ciklusa. Za vreme dok se generiše referentni vremenski interval u pozadini se taktuje (inkrementira) 16-bitni tajmer1 modul i programski odbrojavaju prekoračenja njegove osnove brojanja (2^{16}). Broj prekoračenja čuva se u 16-bitnoj promenljivoj *freqc_high* ili *perc_high* tako da je ukupan rezultat merenja 32-bitni. Ugrađena biblioteka LCD.C upravlja LCD modulom koji se koristi za vizuelni prikaz merene frekvencije/periode.

6.3. TAJMER/BROJAČ 0 (TMR0)

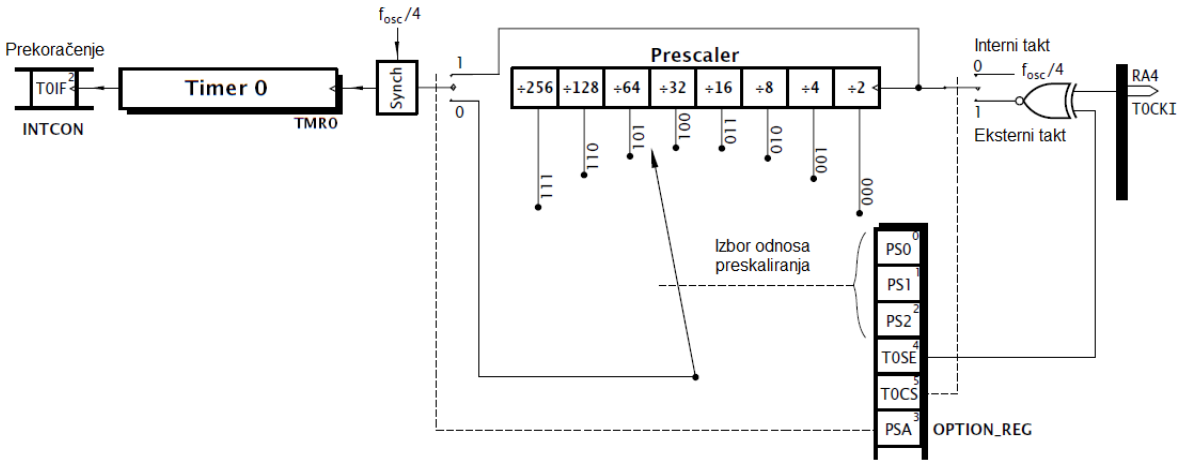
Tajmer 0 je integrisana periferija mikrokontrolera namenjena merenju vremena i frekvencije i brojanju događaja koja poseduje:

- 8-bitni R/W brojački registar TMR0 (SFR registar),
- 8-bitni programabilni preskaler (delitelj frekvencije) koji nije dostupan programeru,
- mogućnost internog ili spoljašnjeg taktovanja,
- izbor ivice impulsa (rastuća ili opadajuća) za spoljašnji takt,
- digitalnu logiku za detekciju prekoračenja tajmera/brojača prilikom tranzicije iz stanja FFh u stanje 00h i generisanje prekidnog zahteva.

Periferija se može naći u dva radna režima: tajmerski ili brojački, u zavisnosti od izvora taktnog signala. Tajmerski (timer) radni režim ovog modula selektuje se resetovanjem bita T0CS (*Timer 0 Clock Select*) OPTION registra OPTION_REG<5>. U ovom radnom režimu 8-bitni brojački registar tajmera 0 TMR0 inkrementira se internim taktnim signalom frekvencije $f_{OSC}/4$, tj. na svaki instrukcijski ciklus (u odsustvu preskalera-delitelja frekvencije). Posle jednog upisa u TMR0 registar, u naredna dva instrukcijska ciklusa inkrement brojačkog registra je onemogućen zbog sinhronizacije.

Brojački (counter) radni režim modula selektuje se setovanjem bita T0CS OPTION registra OPTION_REG<5>. U brojačkom radnom režimu 8-bitni brojački registar TMR0 inkrementira se na svakoj rastućoj ili opadajućoj ivici impulsa spoljašnjeg taktnog signala na liniji RA4/T0CKI. (u odsustvu preskalera). Ivica spoljašnjeg impulsa, kojom se inkrementira brojački registar TMR0, bira se setovanjem/resetovanjem bita T0SE (*Timer 0 Source Edge select*) OPTION registra OPTION_REG<4>. Resetovani bit odgovara rastućoj ivici impulsa i suprotno.

Zbog sinhronizacije spoljašnjeg taktnog signala sa internim taktnim impulsima, kao na slici 6.3.1, neophodno je da trajanje visokog naponskog nivoa taktnog impulsa na RA4/T0CKI ulazu bude najmanje $2T_{OSC}$+20ns i isto toliko niskog naponskog nivoa, za slučaj kada preskaler nije dodeljen tajmeru 0 (odnos preskaliranja je tada 1:1), gde je T_{OSC} perioda kristalnog kvarcnog oscilatora. Kada se koristi preskaler ova minimalna perioda spoljašnjeg taktnog signala od $4T_{OSC}$+40ns treba biti podeljena odnosom preskaliranja.



Slika 6.3.1. *Blok šema tajmer 0 modula PIC16F877 sa preskalerom.*

Tajmer 0 prekid se generiše na prelazu osmobitnog brojačkog registra TMR0 iz stanja FFh u stanje 00h (prekoračenje brojačkog registra), za šta postoji podrška ugrađene digitalne logike. Ovo prekoračenje setuje flag bit T0IF INTCON registra INTCON<2>. Prekid može biti onemogućen (maskiran) resetovanjem bita T0IE INTCON registra INTCON<5>. T0IF flag bit mora biti programski resetovan neposredno pre izlaska iz prekidne rutine kako bi se omogućio novi prekid tajmera 0.

Prekid tajmer 0 modula se ne može iskoristiti za buđenje mikrokontrolera iz sleep režima budući da se brojački registar tajmera 0 taktuje internim taktom, jednakim četvrtini frekvencije kristalnog oscilatora (trajanje instrukcijskog ciklusa), ili je u slučaju spoljašnjeg taktovanja taktni signal sinhronizovan internim taktom koji se blokira po uspostavljanju stanby režima.

8-bitni programabilni preskaler, koji nije dostupan programeru (registar preskalera nije mapiran u RAM memoriji), je deljivi resurs koji programski može biti dodeljen tajmer 0 modulu ili watchdog tajmeru MCU. Suštinski gledano, radi se o osmobitnom binarnom brojaču čiji svaki sledeći razred deli frekvenciju signala sa izlaza prethodnog razreda faktorom dva, vidi sliku 6.3.1. Status kontrolnih bitova OPTION registra OPTION_REG<3:0>, PSA (*Prescaler Assignment*) i PS2:PS0, tabela 6.3.1, određuje dodelu preskalera i odnos preskaliranja, respektivno. Kada je preskaler dodeljen tajmeru 0, tada svaki upis u TMR0 brojački registar (CLRF TMR0, MOVWF TMR0, BSF TMR0, x itd.) briše registar preskalera dok instrukcije čitanja tajmera 0 nemaju uticaja na preskaler. Kada je preskaler dodeljen watchdog tajmeru, tada instrukcija brisanja watchdog tajmera CLRWDT briše i registar preskalera. U oba slučaja, brisanje registra preskalera ne menja dodelu ovog kola tajmer 0 modulu ili watchdog tajmeru.

Imajući u vidu da se brojački registar preskalera, kada je dodeljen watchdog tajmeru, taktuje izlaznim impulsima watchdog tajmera tipične periode ponavljanja (time-out period) 18ms, vidi sliku 4.1.10. poglavlja 4.1.3, on se objektivno u ovom slučaju mora tretirati kao postskaler dok je u slučaju dodele tajmer 0 periferiji to preskaler.

Sa slike 6.3.1. se može videti da se odnos preskaliranja 1:1 za tajmer 0 modul može postići ako se prethodno preskaler dodeli watchdog tajmeru (PSA=1).

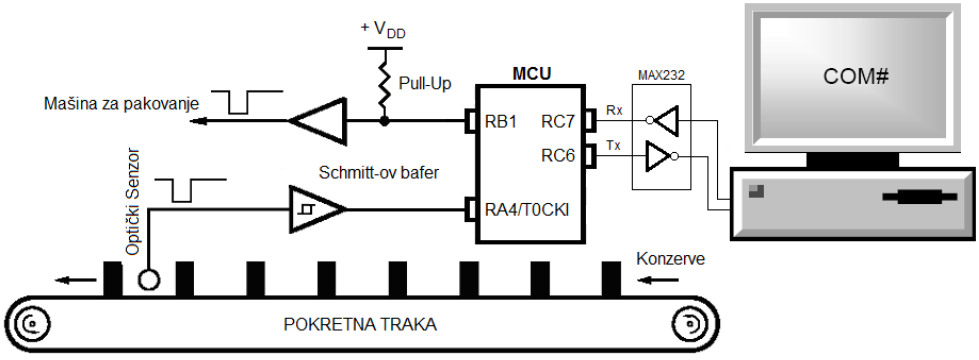
U tabeli 6.3.1. je dat opis kontrolnih bitova 8-bitnog OPTION registra MCU.

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
	RBPU	**INTEDG**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**
	bit 7							bit 0
bit 7	**RBPU:** Bit za omogućenje pull-up otpornika na portu B 1 = PORTB pull-up otpornici onemogućeni 0 = PORTB pull-up otpornici individualno omogućeni na ulaznim linijama porta B							
bit 6	**INTEDG**: Bit za selekciju ivice impulsa spoljašnjeg prekida na RB0/INT liniji porta B 1 = Prekid na rastućoj ivici impulsa na RB0/INT liniji 0 = Prekid na opadajućoj ivici impulsa na RB0/INT liniji							
bit 5	**T0CS**: Bit za izbor taktnog signala tajmera 0 1 = Spoljašnji taktni signal na RA4/T0CKI liniji 0 = Interni taktni signal (inkrement TMR0 registra na svaki instrukcijski ciklus)							
bit 4	**T0SE**: Bit za izbor ivice impulsa spoljašnjeg taktnog signala 1 = Inkrement TMR0 registra na opadajućoj ivici impulsa na RA4/T0CKI liniji 0 = Inkrement TMR0 registra na rastućoj ivici impulsa na RA4/T0CKI liniji							
bit 3	**PSA**: Bit za dodelu preskalera (delitelja frekvencije taktnih impulsa) 1 = Preskaler dodeljen Watchdog tajmeru 0 = Preskaler dodeljen Tajmeru 0							
bit 2-0	**PS2:PS0**: Bitovi za izbor odnosa preskaliranja (deljenja)							
	Vrednost	Za tajmer 0	Za Watchdog tajmer					
	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					

Tabela 6.3.1. *Opcioni registar PIC16F877 MCU.*

Tajmer 0 se uglavnom koristi za npr. brojanje spoljašnjih događaja ili za određivanje perioda pojavljivanja spoljašnjih događaja. Koristi se i za precizno generisanje vremenskih kašnjenja bez učešća procesora. Na slici 6.3.2. prikazana je principijelna šema povezivanja MCU za brojanje proizvoda i pakovanje u sistemima sa pokretnom trakom. Ugrađeni optički senzor kretanja generiše impuls svaki put kada npr. konzerva na pokretnoj traci preseće svetlosni snop



Slika 6.3.2. *Brojanje proizvoda na pokretnoj traci uz pomoć tajmera 0 MCU.*

koji usmereni izvor svetlosti emituje ka foto prijemniku. Impulsi se posle uobličavanja Schmitt-ovim baferom vode na spoljašnji taktni ulaz tajmera 0. Na svakih 24 odbrojanih konzervi MCU generiše impuls približnog trajanja 100ms kojim inicijalizira mašinu za pakovanje. Sistem posredstvom RS232 serijskog komunikacionog interfejsa šalje poruku PC računaru o tekućem broju paketa. Ako iz bilo kojih razloga dođe do prekida napajanja ili zaglavljivanja programa sistem se samo resetuje i generiše poruku o vrsti reseta.

U tabeli 6.3.2. je opisan program, napisan na C jeziku, koji vrši monitoring proizvoda na pokretnoj traci, upravlja radom mašine za pakovanje i serijskim komunikacionim interfejsom šalje poruku PC računaru o tekućem broju paketa.

```
 // Program za monitoring pokretne trake i upravljanje mašinom za pakovanje proizvoda

#include <16F877.h>
#fuses HS,WDT,NOPROTECT,NOLVP,PUT              // Konfiguraciona reč
#use delay(clock=20000000)                     // Taktna frekvencija CPU
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7) // Direktiva za konfigurisanje USART
                                               // serijskog komunikacionog interfejsa
#define  Br_konzervi   -24                     // Početna vrednost TMR0 je -24 ili 232
Long  Br_paketa;                     // Broj paketa sa po 24 konzerve

#int_rtcc                            // Prekidna funkcija (izvršava se svaki put
void count_isr()                     // kada Tajmer 0 ili RTCC (timer0) prekoraci
{                                    // opseg brojanja (255 --> 0)).
   output_low(PIN_B1);               // Upravljački impuls mašine za pakovanje = low
   delay_ms(100);
   output_high(PIN_B1);              // Upravljački impuls mašine za pakovanje = high
   set_timer0(Br_konzervi);          // Presetovanje (postavljanje na predefinisanu
                                     // vrednost) Tajmer 0 modula
   Br_paketa++;                      // Uvećaj broj paketa
}
void main() {
   switch ( restart_cause() )        // Vrednost koja indicira uzrok poslednjeg reseta
   {
      case WDT_TIMEOUT:              // Reset usled isticanja time-out perioda WDT-a
      {
         printf("\r\nZaglavljen program, restart procesora!\r\n");
         break;
      }
      case NORMAL_POWER_UP:          // POR reset (pri uspostavljanju napajanja MCU)
      {
         printf("\r\nPOR reset!\r\n"); // Slanje poruke PC računaru naredbom printf
         break;
      }
   }
   setup_wdt(WDT_2304MS);            // Time-out period WDT-a = 2340ms
   setup_counters(RTCC_EXT_L_TO_H, WDT_2304MS); // Spoljašnji takt tajmera 0 rastućom
                                                // ivicom impulsa (L_TO_H)
   set_timer0(Br_konzervi);                     // Presetovanje Tajmer 0 modula
   enable_interrupts(INT_RTCC);
   enable_interrupts(GLOBAL);
   output_high(PIN_B1);              // Upravljački impuls mašine za pakovanje = high
   Br_paketa=0;

   do
   {
      restart_wdt();                            // Restart WDT tajmera
      printf("%lu Tekući broj paketa je \n\r",Br_paketa);
   }
  while (TRUE);
}
```

Tabela 6.3.2. *Program za monitoring pokretne trake i upravljanje mašinom za pakovanje.*

6.4. TAJMER/BROJAČ 1 (TMR1)

Integrisana periferija MCU, tajmer 1, predstavlja 16-bitni tajmer/brojač koji poseduje dva 8-bitna R/W registra TMR1H (high) i TMR1L (low). Ovaj registarski par čini 16-bitni TMR1 registar sa ukupno 2^{16}=65536 stanja (od 0000h do FFFFh). Tajmer 1 prekid, ako je omogućen, generiše se na prelazu 16-bitnog brojačkog registarskog para TMR1 iz stanja FFFFh u stanje 0000h. Zastavica prekida TMR1IF, u registru PIR1<0>, setuje se bez obzira na to da li je prekid omogućen ili ne. Prekid može biti omogućen/onemogućen setovanjem/resetovanjem bita TMR1IE, respektivno, u registru za maskiranje/demaskiranje prekida sa periferijskih jedinica PIE1<0>.

Ova periferijska jedinica je, između ostalog, projektovana i za povezivanje sa spoljašnjim kristalnim oscilatorom za kontrolu aplikacija u realnom vremenu. Kada je spoljašnji kristalni oscilator tajmera 1 omogućen, setovanjem bita T1OSCEN kontrolnog registra T1CON<3>, linije RC0 i RC1 porta C, na koje se povezuje kvarc kristal, automatski se konfigurišu kao izlazna i ulazna, respektivno, bez obzira na sadržaj direkcionog TRIS C registra.

Tajmer 1 može raditi u jednom od dva režima:

- tajmer - brojački registarski par TMR1 taktuje se internim taktnim signalom frekvencije $f_{OSC}/4$, odnosno, inkrementira na svaki instrukcijski ciklus (u odsustvu preskalera-delitelja frekvencije).
- brojač - taktovanje brojačkog registarskog para TMR1 rastućom ivicom spoljašnjeg izvora taktnog signala.

T1CON REGISTAR

	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	-	**-**	**T1CKPS1**	**T1CKPS0**	**T1OSCEN**	**T1SYNC**	**TMR1CS**	**TMR1ON**
	bit 7							bit 0
bit 7-6	**Neimplementirani:** čitaju se kao '0'.							
bit 5-4	**T1CKPS1 T1CKPS0**: Bitovi za izbor odnosa preskaliranja ulaznog takta tajmera 1. 11=1:8 Vrednost odnosa preskaliranja. 10=1:4 Vrednost odnosa preskaliranja. 01=1:2 Vrednost odnosa preskaliranja. 00=1:1 Vrednost odnosa preskaliranja.							
bit 3	**T1OSCEN**: Kontrolni bit za omogućenje eksternog oscilatora tajmera 1. 1 = Oscilator omogućen. 0 = Oscilator onemogućen isključenjem invertora oscilatornog kola.							
bit 2	**T1SYNC**: Bit za kontrolu sinhronizacije eksternog takta tajmera 1. 1 = Nema sinhronizacije eksternog takta. 0 = Sinhronizacija eksternog takta. Napomena: Kada je selektovan interni takt (TMR1CS=0) ovaj bit je ignorisan.							
bit 1	**TMR1CS**: Bit za selekciju takta tajmera 1. 1 = Eksterni takt sa RC0 pina na rastućoj ivici impulsa. 0 = Interni takt ($f_{OSC}/4$).							
bit 0	**TMR1ON**: Bit za uključenje/isključenje tajmera 1. 1 = Tajmer 1 uključen (omogućen). 0 = Tajmer 1 isključen (onemogućen).							

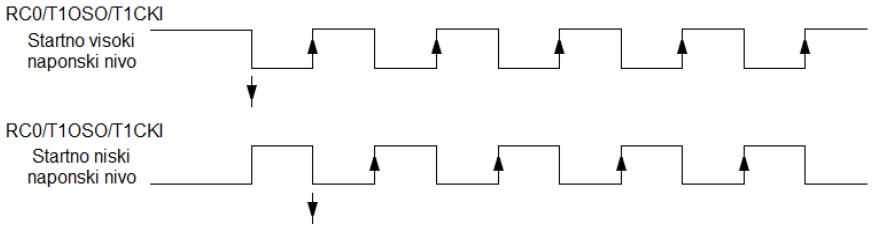
Tabela 6.4.1. *Sardžaj kontrolnog registra tajmer 1 periferijske jedinice PIC16F877 MCU.*

Radni režim se određuje izborom izvora taktnog signala pomoću bita TMR1CS u kontrolnom registru tajmera 1 T1CON<1>, tabela 6.4.1. Rad ove periferijske jedinice može biti omogućen/onemogućen setovanjem/resetovanjem kontrolnog bita TMR1ON, respektivno, kontrolnog registra T1CON<0>. Tajmerski radni režim bira se resetovanjem bita TMR1CS kontrolnog registra T1CON<1>. Brojački registarski par TMR1 inkrementira se internim taktom frekvencije $f_{OSC}/4$, na svaki instrukcijski ciklus, dok stanje sinhronizacionog kontrolnog bita $\overline{\text{T1SYNC}}$ nije od značaja budući da je interni takt u isto vreme i sinhronizacioni taktni signal.

Brojački radni režim tajmera 1 bira se setovanjem bita TMR1CS kontrolnog registra tajmera T1CON<1>. Spoljašnji izvor taktnog signala inkrementira brojački registarski par TMR1 svakom rastućom ivicom spoljašnjeg takta na liniji RC0, kada je kontrolni bit T1OSCEN kontrolnog registra tajmera 1 T1CON<3> resetovan, ili na liniji RC1, kada je isti kontrolni bit setovan. Neposredno po omogućenju ovog radnog režima neophodna je jedna opadajuća ivica taktnog impulsa kako bi se sledećim nizom rastućih brojač uzastopno inkrementirao, kao na slici 6.4.1.

Kada se nađe u brojačkom radnom režimu tajmer 1 se može konfigurisati za rad u

- sinhronom ili
- asinhronom brojačkom modu.



Slika 6.4.1. *Inkrementiranje TMR1 registarskog para na svakoj rastućoj ivici spoljašnjeg takta nakon pojave prve opadajuće ivice.*

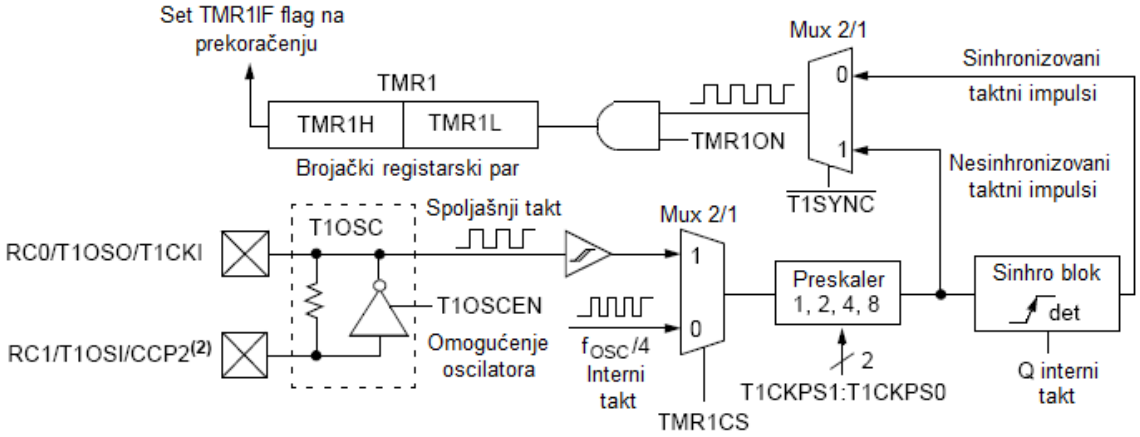
Sinhroni brojački mod, karakteriše sinhronizacija spoljašnjeg taktnog signala jednom fazom internog kada je kontrolni bit $\overline{\text{T1SYNC}}$ resetovan. Sinhronizacija sa internim taktom frekvencije $f_{OSC}/4$ (f_{OSC} – frekvencija rezonanse kvarc kristala) vrši se sa izlaza preskalera, slika 6.4.2, koji je realizovan kao asinhroni kaskadni brojač osnove brojanja tri i koji nije dostupan programeru. U sinhronom modu, u toku trajanja sleep režima, tajmer 1 neće biti inkrementiran čak i u prisustvu spoljašnjeg taktnog signala zbog blokiranog sinhro kola dok se brojački registar preskalera tajmera 1 nastavlja da inkrementira.

Kada je kontrolni bit $\overline{\text{T1SYNC}}$ kontrolnog registra T1CON<2> setovan, spoljašnji taktni signal premošćava sinhro blok kao na slici 6.4.2, a brojač prelazi u asinhroni mod. TMR1 brojački registar će nastaviti da se inkrementira i po ulasku MCU u sleep režim što, u krajnjem, vodi generisanju prekida prilikom tranzicije TMR1 registra iz stanja FFFFh u stanje 0000h. Ovaj se prekid može iskoristiti za buđenje (*wakeup*) MCU iz standby režima. U asinhronom brojačkom režimu, međutim, tajmer 1 ne može biti korišćen kao vremenska baza za operacije hvatanja (*Capture*) i poređenja (*Compare*) CCP periferije.

Posebna pažnja u radu sa tajmer 1 modulom u asinhronom brojačkom režimu treba biti posvećena operacijama čitanja i upisa u TMR1 16-bitni registarski par. Ispravnost operacije čitanja je hardverski zagarantovana. Problem, međutim, može nastati pri uzastopnom čitanju dve osmobitne vrednosti i mogućem prekoračenju brojača između dva čitanja. Pri upisu, preporučuje se jednostavno stopiranje tajmera i potom upis željenog podatka. Upis informacije za vreme inkrementiranja TMR1 registarskog para može proizvesti jednu nepredvidivu 16-bitnu vrednost.

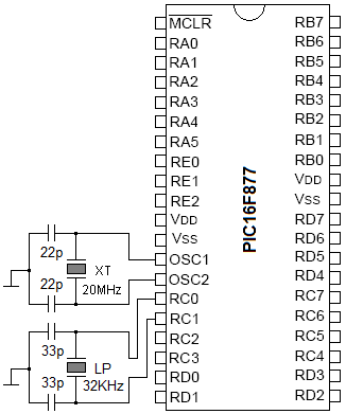
Kao što je napred rečeno, tajmer 1 je projektovan i za povezivanje sa spoljašnjim kristalnim oscilatorom. Naime, deo kola kristalnog oscilatora ugrađen je između linija RC0 i RC1 dok se preostale komponente oscilatora dodaju spolja (kvarc kristal i dva blok kondenzatora), kao na

slici 6.4.3. Kolo oscilatora je projektovano za malu potrošnju (*low power*) i za radne frekvencije do 200kHz. Primarno, kolo oscilatora je namenjeno povezivanju sa satnim (*watch*) kvarc kristalom rezonantne frekvencije 2^{15}=32768Hz. Oscilator može nastaviti sa radom i kada je mikrokontroler u sleep režimu. Da bi se osigurao ispravan start kristalnog oscilatora tajmera 1, mora se obezbediti odgovarajuće softversko kašnjenje u skladu sa trajanjem prelaznog režima oscilatora.



Slika 6.4.2. *Blok šema tajmera 1 sa priključnim linijama porta C za spoljašnji oscilator.*

Jedan od načina striktnog tajminga uz pomoć tajmer 1 periferije je presetovanje samo TMR1H registra posle svakog prekidnog signala tajmera 1, odnosno, u toku prekidne procedure. Na primer, konfigurisanjem preskalera na vrednost 1 i presetovanjem brojačkog registarskog para na vrednosti TMR1H=80H i TMR1L=00H osiguraće se prekid posle tačno jedne sekunde, pri rezonantnoj frekvenciji kvarc kristala od 32768Hz. Nadalje će striktni tajming, bez gubitka i jednog instrukcijskog ciklusa, biti osiguran presetovanjem samo TMR1H registra na vrednost 80H po ulasku u prekidnu rutinu dok u pozadini registar TMR1L nastavlja sa inkrementiranjem.



Slika 6.4.3. *Povezivanje watch kvarc kristala (32768Hz) u kolo oscilatora tajmera 1.*

TMR1H (hex) TMR1L (hex)=00	Vreme brojanja do pojave prekida (s)
80	1
C0	0.5
E0	0.25
F0	0.125

Tabela 6.4.2. *Početne heksa vrednosti TMR1H registra pri TMR1L=0 i odgovarajuća striktna kašnjenja tajmera 1 (do pojave prekida) sa spoljašnjim watch kvarcnim oscilatorom.*

Prema tome, asinhroni rad tajmera 1 sa vlastitim kvarcnim oscilatorom daje mogućnost projektantu da lako upravlja zadacima u realnom vremenu minimizirajući potrošnju i eksternu

logiku. Tabela 6.4.2. prikazuje neke vrednosti TMR1H registra (TMR1L=0) i odgovarajuće zadrške u sekundama pri frekvenciji kvarc kristala tajmer 1 periferije od 32768Hz.

	Frekvencija (KHz)		
Preskaler	*32.768*	*100*	*200*
1	2	0.655	0.327
2	4	1.31	0.655
4	8	2.62	1.31
8	16	5.24	2.62

Tabela 6.4.3. *Striktna kašnjenja tajmera 1 pri brojanju od početne vrednosti 0000H do pojave prekida (puni ciklus brojanja) za različite frekvencije taktovanja spoljašnjeg kristalnog oscilatora.*

Drugi način striktnog tajminga je resetovanje brojačkog registarskog para TMR1 pri čemu se po ulasku u prekidnu rutinu, usled prekoračenja tajmera 1 (puni ciklus brojanja), nastavlja sa inkrementiranjem TMR1 u pozadini bez programskog uticaja na sadržaj oba registra. U tabeli 6.4.3. su date izračunate vrednosti kašnjenja koje proizvodi jedan puni ciklus brojanja tajmera 1 pri različitim vrednostima preskalera i za različite rezonantne frekvencije kvarc kristala.

Tajmer 1 poseduje ugrađeni interni ulaz za resetovanje kojim upravlja ma koji od dva CCP (*Capture/Compare/PWM*) modula (CCP1 ili CCP2). Naime, CCP1 ili CCP2 periferija MCU, kada je konfigurisana u tkzv. Compare modu, generiše specijalni interni okidni signal kojim se resetuje tajmer 1, poređenjem sadržaja 16-bitnih registara CCPR1 ili CCPR2 sa TMR1 registrom. Da bi se reset operacija tajmera 1 CCP periferijama dogodila, tajmer 1 mora biti konfigurisan u tajmerskom ili sinhronom brojačkom modu. U asinhronom brojačkom režimu tajmera 1 opisana reset operacija je onemogućena.

Za slučaj kada jedna operacija upisa u jedan od brojačkih registara tajmera 1 koincidira sa pojavom specijalnog okidnog signala CCP1 ili CCP2 modula, operacija upisa će imati prioritet.

Registarski parovi CCP periferija CCPRxH i CCPRxL (x=1, 2) u ovom radnom režimu efektivno postaju 16-bitni programabilni period registri tajmera 1. Specijalni interni, hardverski okidni signal, resetuje TMR1 registarski par ali nema uticaj na indikator prekida tajmera 1 (TMR1IF flag bit). TMR1 brojački registar se ne resetuje na POR (Power On Reset) ili bilo koju drugu vrstu reseta, osim specijalnim okidnim signalom jedne od dve CCP periferije. Međutim, POR ili BOR (Brown Out Reset) reset resetuju kontrolni registar tajmera 1 T1CON čime se blokira inkrementiranje tajmera 1 (T1CON<0>) a odnos deljenja preskalera postavlja na vrednost 1:1. Inače, bilo koji upis u TMR1H ili TMR1L registre briše brojački registar preskalera.

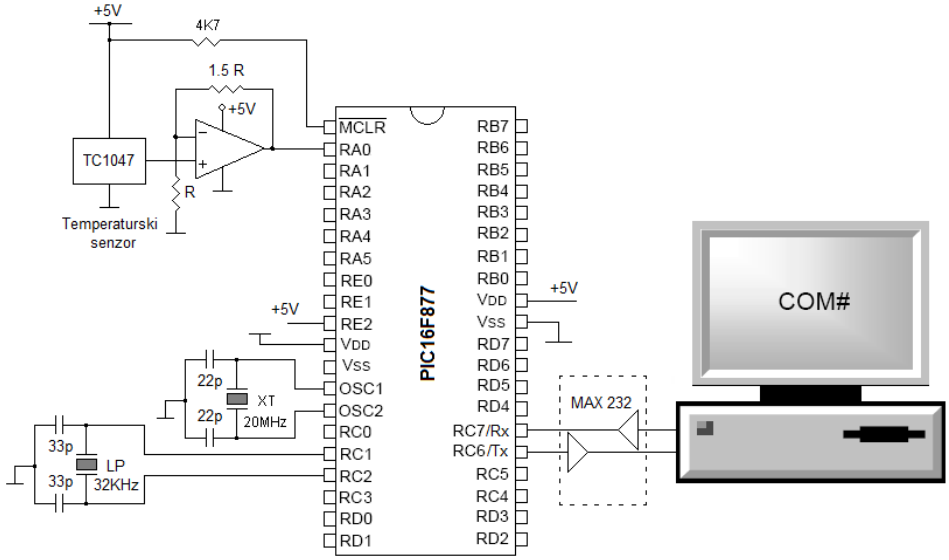
Na slici 6.4.4. je prikazan jednostavan računarski sistem za prikupljanje i obradu podataka o temperaturi merne tačke, zasnovan na analognom integrisanom temperaturskom senzoru TC1047, proizvodnje Microchip. Sistem čini MCU PIC16F877 povezan sa PC računarom preko serijskog komunikacionog interfejsa USART u asinhronom režimu rada. Striktno odmeravanje temperature je postignuto primenom spoljašnjeg kvarc kristal oscilatora tajmera 1, frekvencije rezonanse 32768Hz, kao na slici 6.4.4. Tajmer 1 je konfigurisan u asnhronom brojačkom modu.

Na slici 6.4.5. je prikazana linearna karakteristika prenosa integrisanog analognog senzora temperature TC1047 sa nagibom od 10mV/^{0}C. Prema podacima proizvođača, greška linearnosti je u granicama ±1^{0}C u celom opsegu radnih temperatura a karakteristika prenosa oblika

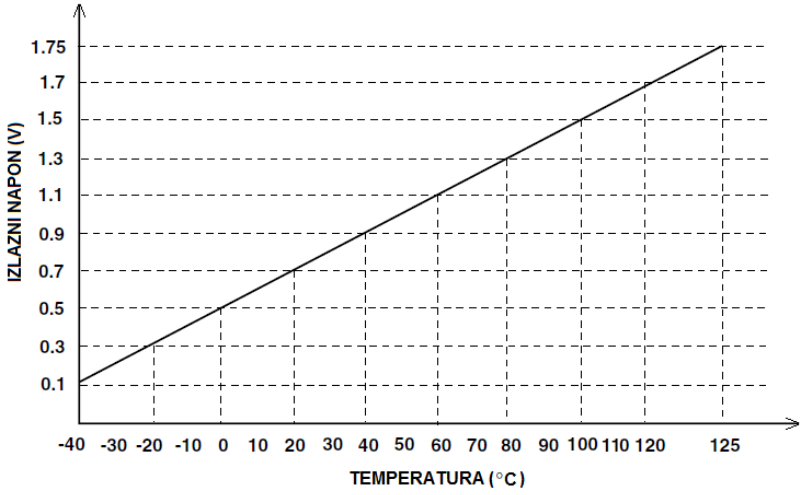
$$V_{OUT} = 0.01T + 0.5$$

gde je T temperatura u stepenima Celsius-ovim a V_{OUT} izlazni napon u voltima. Kao što se može videti na slici 6.4.4, izlaz senzorskog kola je povezan sa ulazom neinvertujućeg

pojačavača, stepena pojačanja 2.5, kako bi se opseg promene temperaturski zavisnog napona približnije uskladio sa opsegom konverzije A/D konvertora (0V - 5V).



Slika 6.4.4. *Primer akvizicionog hardvera na bazi analognog senzora temperature i watch kvarc kristala (32768Hz) tajmera 1 za striktno odmeravanje.*



Slika 6.4.5. *Linearna karakteristika prenosa integrisanog senzora temperature TC1047.*

Svaki 10-bitni digitalni ekvivalent temperature, dobijen A/D konverzijom izlaznog napona neinvertujućeg pojačavača, šalje se PC računaru putem serijskog RS232 porta brzinom 9600 baud-a na dalju obradu. Proces akvizicije treba započeti prijemom bilo kog karaktera na liniji za serijski prijem MCU (Rx), koji se salje od strane PC računara, a stopira se prekidnim signalom po prijemu karaktera 'S'. U tabeli 6.4.4. je prikazan C kod programa, napisan u skladu sa napred postavljenim zahtevima.

```
#include <16F877.h>
#device ADC=10
#fuses HS,NOWDT,PROTECT,NOLVP,PUT
```

```
#use delay (clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)  //RS232 ser. Brzina 9600 Baud.

#bit RCBUFF_FULL=0x0C.5   //PIR1,bit5 (RCIF - setovan kada je prijemni bafer pun)
#bit t1_overflow=0x0C.0   //PIR1,bit0 (TMR1IF) - setovan kada dođe do prekoračenja T1
#byte TMR1H=0x0F          //Adresa TMR1H registra

   char  stopc;           //Deklaracija globalne promenljive

#int_rda                  //prekidna rutina (podatak primljen preko RS232 serijskog
void serial_isr() {       //komunikacionog interfejsa raspoloživ u prijemnom baferu)
   stopc=getch();         //čitanje primljenog karaktera
}

void main()
{
   long value;            //Deklaracija lokalne promenljive

   setup_port_a( ALL_ANALOG );      //Linije porta A konfigurisane kao analogni ulazi
   setup_adc( ADC_CLOCK_INTERNAL ); //Interni RC generator takta A/D sa Tad=2-6us
   set_adc_channel( 0 );            //Izbor kanala 0 A/D konvertora

STP:                                //Start point labela

   RCBUFF_FULL=0;
   t1_overflow=0;
   disable_interrupts(global);
   disable_interrupts(int_rda);
   setup_timer_1(T1_DISABLED);      //Stop tajmer 1

   while (!RCBUFF_FULL);            //Čekaj do prijema bilo kog karaktera (START) bez
                                    //omogućenog prekida
   enable_interrupts(global);       //Omogući globalni prekid
   enable_interrupts(int_rda);      //Omogući prekid po prijemu karaktera

   setup_timer_1(T1_EXTERNAL|T1_DIV_BY_1|T1_CLK_OUT); //Ext. kvarcni oscilator 2^15Hz
   set_timer1(0);                                     //Reset i start tajmera 1

               /*___ PROGRAM ZA OBRADU I SLANJE PODATAKA RACUNARU ___*/
Loop:
      TMR1H=0xC0;              //Prekid na svakih 0.5s striktno, setovanjem
                               //samo višeg bajta TMR1H
      if(stopc=='S')           //STOP karakter je 'S' i prima se u prekidnoj rutini
         {goto  STP;}
      else {
         value = Read_ADC();   //Čitanje dig. ekvivalenta analogne vrednosti na ch0
         printf("%LX",value);  //Slanje PC računaru promenljive value tipa long
T1_CHK:
         if (t1_overflow)      //Provera da li je došlo do prekoračenja tajmera 1
            {t1_overflow=0; goto Loop;}
         else
            {goto T1_CHK;}
          }
}
```

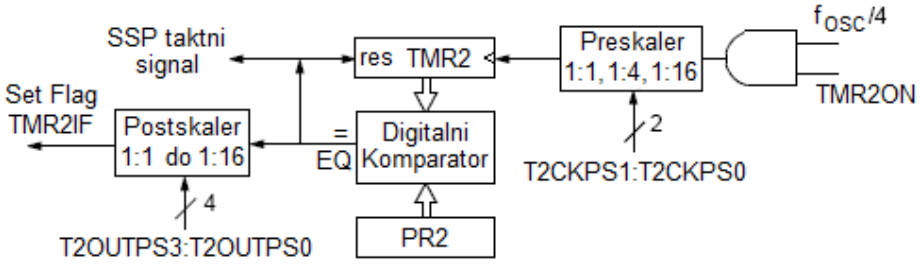
Tabela 6.4.4. *Striktno odmeravanje temperature tajmer 1 periferijom u kolu za akviziciju temperature merne tačke.*

6.5. TAJMER/BROJAČ 2 (TMR2)

Tajmer 2 je integrisana periferija MCU koja sadrži 8-bitni brojački registar TMR2, registre preskalera i postskalera kao i 8-bitni period registar PR2 R/W tipa. Koristi se uglavnom kao vremenska baza u postupku generisanja impulsno-širinski modulisanog signala PWM (*Pulse Width Modulation*) uz pomoć CCP modula. Brojački registar tajmera 2 TMR2 je R/W tipa i briše se posle bilo koje vrste reseta MCU.

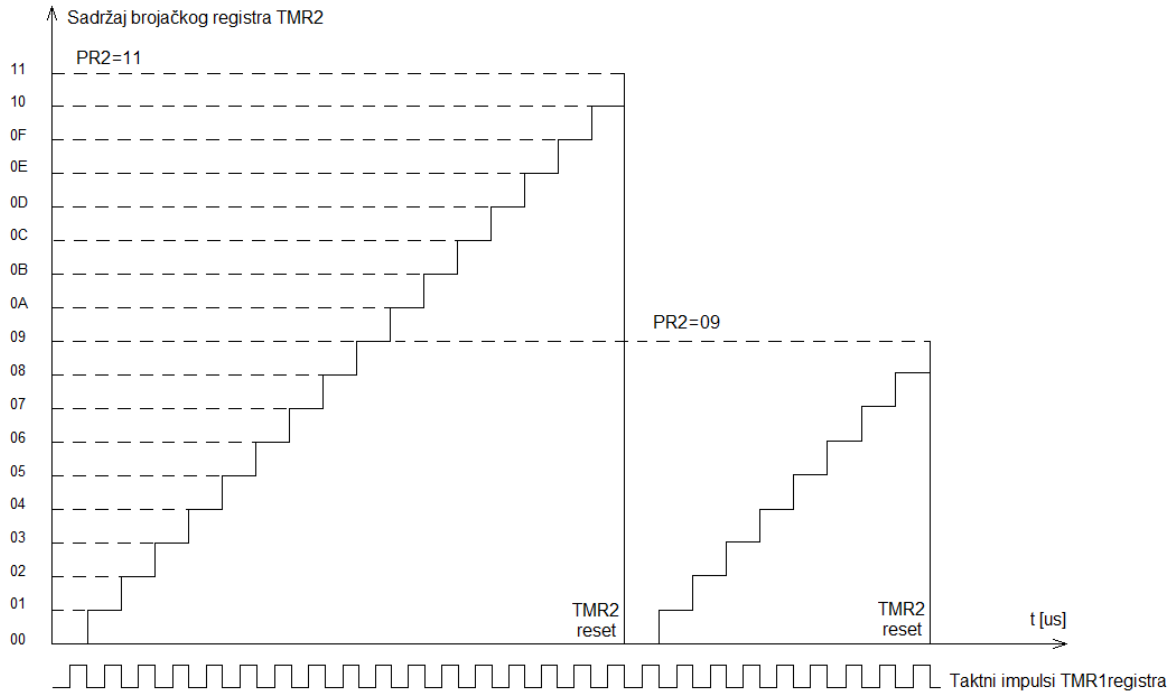
Na slici 6.5.1. je prikazana blok šema tajmer 2 periferije sa koje se vidi da se frekvencija ulaznog taktnog signala $f_{OSC}/4$ (interni takt) najpre deli uz pomoć preskalera odnosom deljenja 1:1, 1:4 ili 1:16 i potom takav signal vodi na ulaz za taktovanje brojačkog registra tajmera 2 TMR2. Promenljivi sadržaj brojačkog registra TMR2, koji se inkrementira posredstvom taktnih impulsa, stalno se upoređuje sa fiksnim sadržajem stacionarnog period registra PR2 uz pomoć 8-bitnog digitalnog komparatora. Izlazni signal komparatora EQ resetuje TMR2 registar u trenutku izjednačenja sadržaja ovog registra sa sadržajem PR2 registra. Isti izlazni signal digitalnog komparatora koristi se nadalje kao taktni signal registra postskalera ali i kao pomerački taktni signal integrisane periferije MCU – sinhronog serijskog porta (SSP). Signal dobijen na izlazu 4-bitnog brojačkog registra postskalera koristi se kao prekidni signal tajmer 2 periferije koji setuje zastavicu prekida TMR2IF u registru PIR1<1>. Kao što se sa slike 6.5.1. može videti, prekidni signal na izlazu postskalera se može generisati pri svakom izjednačenju sadržaja registara TMR2 i PR2, pri svakom drugom izjednačenju, svakom trećem itd., do pri svakom šesnaestom, što zavisi od izabranog odnosa postskaliranja.

8-bitni period registar PR2 se posle bilo koje vrste reseta MCU automatski inicijalizira u stanje FFh. Međutim, on je R/W tipa, mapiran u SRAM memoriji MCU, kao i TMR2 registar, pa se njegov sadržaj može modifikovati.



Slika 6.5.1. *Blok šema tajmer 2 periferije PIC16F877 MCU.*

Na slici 6.5.2. je ilustrovan princip poređenja inkrementirajućeg sadržaja brojačkog registra TMR2 sa sadržajem period registra PR2. Može se primetiti da se vremenski interval između dva reset stanja registra TMR2 menja u skladu sa vrednošću upisanom u PR2 registar. Za veće vrednosti sadržaja PR2 registra trajanja pomenutih intervala su duža i obrnuto. Zbog toga se tajmer 2 i koristi kao vremenska baza pri generisanju PWM signala.



Slika 6.5.2. *Inkrementiranje sadržaja TMR2 registra do izjednačenja sa sadržajem PR2 registra.*

Tajmer 2 modul može biti isključen resetovanjem kontrolnog bita TMR2ON kontrolnog registra T2CON<2>, čime se smanjuje ukupna potrošnja MCU. Sadržaj kontrolnog registra T2CON, tajmer 2 modula, prikazan je u tabeli 6.5.1.

T2CON REGISTAR

	U-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0 R/W-0	
	- TOUTPS3 TOUTPS2 TOUTPS1 TOUTPS0 TMR2ON T2CKPS1 T2CKPS0	
	bit 7	bit 0
bit 7	**Neimplementiran:** čita se kao '0'.	
bit 6-3	**TOUTPS3:TOUTPS0**: Bitovi za izbor odnosa postskaliranja izlaznog signala tajmera 2. 0000=1:1 Vrednost odnosa postskaliranja. 0001=1:2 Vrednost odnosa postskaliranja. 0010=1:3 Vrednost odnosa postskaliranja. . . . 1111=1:16 Vrednost odnosa postskaliranja.	
bit 2	**TMR2ON**: Kontrolni bit za uključenje/isključenje tajmer 2 periferije. 1 = Tajmer 2 uključen. 0 = Tajmer 2 isključen.	
bit 1-0	**T2CKPS1:T2CKPS0**: Bitovi za izbor odnosa preskaliranja ulaznog takta tajmera 2. 00 = Odnos preskaliranja 1:1. 01 = Odnos preskaliranja 1:4. 0x = Odnos preskaliranja 1:16.	

Tabela 6.5.1. *Sadržaj kontrolnog registra tajmer 2 periferijske jedinice PIC16F877 MCU.*

Brojački registri preskalera i postskalera tajmer 2 periferije, koji nisu dostupni programeru, automatski se brišu posle:

- jednog upisa u TMR2 registar,
- jednog upisa u kontrolni registar tajmera 2, T2CON,
- bilo koje vrste reseta MCU (POR, $\overline{\text{MCLR}}$, WDT, BOR).

Upis u kontrolni registar T2CON ne utiče na sadržaj TMR2 brojačkog registra.

6.6. CCP (CAPTURE, COMPARE, PWM) PERIFERIJE

MCU PIC16F877 u svom sastavu ima integrisana dva identična kompozitna CCP periferijska modula, CCP1 i CCP2. Oba modula sadrže po jedan 16-bitni registarski par CCPRx, sastavljen od dva osmobitna registra CCPRxL i CCPRxH, x=1,2, koji može raditi kao:

- 16-bitni prihvatni registar CCPRx, x=1,2 (Capture)
- 16-bitni registar za poređenje CCPRx x=1,2 (Compare)
- PWM (*pulse width modulation*) master/slave registar CCPRx za izbor koeficijenta ispune signala (Duty Cycle)

CCP mod	Korišćeni tajmerski resursi
Capture	Tajmer 1
Compare	Tajmer 1
PWM	Tajmer 2

Tabela 6.6.1. *Tajmerski resursi korišćeni u CCP periferijama.*

Rad oba modula je identičan a izuzetak je operacija generisanja specijalnog okidnog signala. Naime, specijalni okidni signal, koji CCP1 periferijski modul u Compare režimu generiše interno-hardverski, u trenutku izjednačenja sadržaja 16-bitnih registara CCPR1 i TMR1, resetuje TMR1 registar dok modul CCP2 istim signalom resetuje TMR1 registar tajmera 1 i startuje jednu A/D konverziju (ako je A/D periferijski modul omogućen).

Tabela 6.6.1. prikazuje tajmerske resurse koje koriste oba CCP modula konfigurisana u sva tri režima rada dok je u tabeli 6.6.2. prikazan međusobni uticaj-interakcija dva CCP periferijska modula konfigurisana u različitim radnim režimima.

CCP1	CCP2	MEĐUSOBNA INTERAKCIJA MODULA
Capture	Capture	Ista TMR1 vremenska baza za oba modula
Capture	Compare	Modul u režimu Compare treba biti konfigurisan za specijalni okidni događaj koji resetuje TMR1 brojački registarski par
Compare	Compare	Modul u režimu Compare treba biti konfigurisan za specijalni okidni događaj koji resetuje TMR1 brojački registarski par
PWM	PWM	Ista izlazna frekvencija i brzina ažuriranja (prekid tajmera 2) za oba modula
PWM	Capture	Bez interakcije
PWM	Compare	Bez interakcije

Tabela 6.6.2. *Uzajamni uticaj dva CCP periferijska modula.*

Svakom od dva periferijska CCP modula pridružen je po jedan kontrolni registar CCP1CON i CCP2CON, mapirani u SRAM memoriji MCU, čiji su sadržaji prikazani u tabeli 6.6.3.

CCP1CON/CCP2CON REGISTRI

	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	-	**-**	**CCPxX**	**CCPxY**	**CCPxM3**	**CCPxM2**	**CCPxM1**	**CCPxM0**
	bit 7							bit 0
bit 7-6	**Neimplementirani:** čitaju se kao '0'.							
bit 5-4	**CCPxX:CCPxY**: Bitovi najmanje težine (LSB) za PWM režim rada. Capture režim: Ne koriste se. Compare režim: Ne koriste se. PWM režim: Ova dva bita predstavljaju LSB bitove registra za izbor koeficijenta ispune signala (Duty Cycle). Preostala osam MSB bita pripadaju nižem bajtu registarskog para CCP modula CCPRxL (x=1,2)							
bit 3-0	**CCPxM3**:**CCPxM0**: Bitovi za izbor radnog režima CCPx modula (x=1,2). 0000 = CCPx modul onemogućen (modul resetovan). 0100 = Capture režim, svaka opadajuća ivica ulaznog impulsa. 0101 = Capture režim, svaka rastuća ivica ulaznog impulsa. 0110 = Capture režim, svaka četvrta rastuća ivica ulaznog impulsa. 0111 = Capture režim, svaka šesnaesta rastuća ivica ulaznog impulsa. 1000 = Compare režim, setovanje izlazne linije CCPx modula pri izjednačenju sadržaja registarskih parova CCPRx i TMR1 (setovanje CCPxIF zastavice). 1001 = Compare režim, resetovanje izlazne linije CCPx modula pri izjednačenju sadržaja registarskih parova CCPRx i TMR1 (setovanje CCPxIF zastavice). 1010 = Compare režim, generisanje programskog prekida CCPx modula pri izjednačenju sadržaja registarskih parova CCPRx i TMR1 (setovanje CCPxIF zastavice, bez uticaja na stanje izlazne linije). 1011 = Compare režim, generisanje specijalnog okidnog signala pri izjednačenju sadržaja registarskih parova CCPRx i TMR1 (setovanje CCPxIF zastavice, bez uticaja na stanje izlazne linije): CCP1 resetuje TMR1, CCP2 resetuje TMR1 i startuje jednu A/D konverziju (ako je A/D periferijski modul omogućen). 11xx = PWM režim							

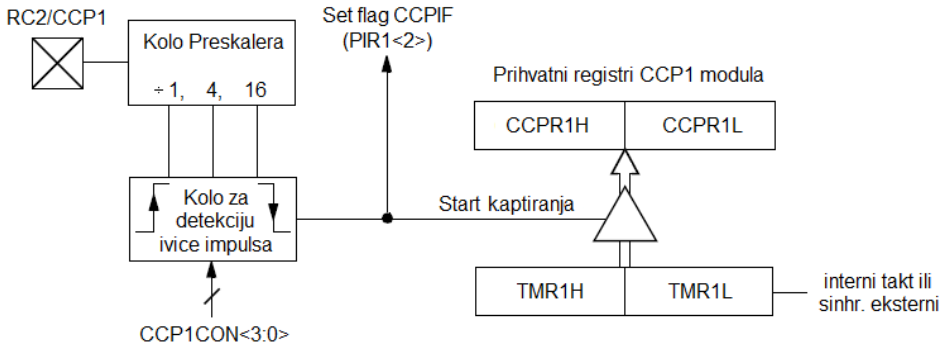
Tabela 6.6.3. *Sadržaj kontrolnih registara CCP1 i CCP2 periferijskih modula.*

6.6.1. CCP u režimu kaptiranja (Capture)

Radni režim kaptiranja (Capture) jednog od dva CCP periferijska modula se koristi za precizno merenje periode ili širine impulsa na ulaznoj liniji modula. Maksimalno razlaganje vremena iznosi 12.5ns pri maksimalno dopustivoj frekvenciji taktovanja MCU od 20MHz, odnosno, pri minimalnom trajanju instrukcijskog ciklusa od 200ns (T_{CY}/Max_Preskaler_Odnos_Deljenja = 200ns/16=12.5ns).

Ulazna linija za CCP1 modul je linija porta C RC2 dok je za CCP2 predviđena linija RC1. Budući da je ovaj radni režim identičan za oba modula, dalja izlaganja odnosiće se samo na CCP1 modul. U ovom radnom režimu 16-bitni registarski par CCP1 modula CCPR1H:CCPR1L kaptira 16-bitnu vrednost registra TMR1 tajmera 1 pri jednom od sledećih događaja na ulaznoj liniji RC2 CCP1 modula:

- Pri pojavi svake opadajuće ivice impulsa na ulaznoj liniji RC2,
- Pri pojavi svake rastuće ivice impulsa na ulaznoj liniji RC2,
- Pri pojavi svake četvrte rastuće ivice impulsa na ulaznoj liniji RC2,
- Pri pojavi svake šesnaeste rastuće ivice impulsa na ulaznoj liniji RC2.



Slika 6.6.1. *Blok šema CCP1 periferije u režimu kaptiranja.*

Izbor jedne od četiri vrste događaja na ulaznoj liniji RC2 CCP1 modula vrši se konfigurisanjem kontrolnih bitova CCP1M3:CCP1M0 kontrolnog registra CCP1CON<3:0>. Kada se dogodi kaptiranje setuje se zastavica prekida CCP1IF registra PIR1<2>, odnosno, generiše prekidni zahtev, slika 6.6.1. Zastavica prekida se mora programski resetovati nakon automatskog setovanja. Ako se novi prenos sadržaja iz TMR1 u CCPR1 registar (novo kaptiranje) dogodi pre nego se pročita prethodni sadržaj CCPR1 registra, nova vrednost će biti prepisana preko prethodne.

Za pravilno konfigurisanje CCP1 periferijskog modula u režimu kaptiranja linija RC2 mora biti konfigurisana kao digitalni ulaz, setovanjem drugog bita direkcionog registra TRISC<2>, dok

tajmer 1 periferijski modul mora biti konfigurisan u tajmerskom ili sinhronom brojačkom režimu. Tajmer 1 konfigurisan u asinhronom brojačkom režimu će onemogućiti rad CCP1 modula u režimu kaptiranja.

Promena režima kaptiranja CCP1 modula može generisati jedan lažni Capture prekidni signal. Da bi se ovo izbeglo preporučuje se onemogućenje prekida resetovanjem mask bita za prekide CCP1IE, koji pripada registru PIE1<2>, pre promene režima rada kao i programsko brisanje zastavice prekida CCP1IF posle promene i potom omogućenje prekida setovanjem mask bita CCP1IE.

Preskaler CCP1 modula u režimu kaptiranja može biti konfigurisan u jednom od četiri moda specificirana kontrolnim bitovima CCP1M3:CCP1M0 kontrolnog registra CCP1CON<3:0>. Kada je CCP1 modul isključen, CCP1M3:CCP1M0 = 0000, ili modul nije u režimu kaptiranja brojač preskalera je resetovan. Takođe, bilo koji reset MCU briše registar preskalera.

Promena odnosa preskaliranja u režimu kaptiranja može bit uzrok generisanja lažnog prekidnog signala. S toga se za promenu odnosa preskaliranja CCP1 modula u režimu kaptiranja preporučuje najpre isključenje CCP1 modula, upis nove vrednosti i potom uključenje modula, kao u sledećem primeru:

```
CLRF CCP1CON                     ; Isključenje CCP1 modula
MOVLW NOVA_CAPT_PRESC_VREDNOST   ; Učitavanje nove vrednosti preskaliranja u W registar
                                 ; Prebacivanje nove vrednosti i uključenje CCP modula
MOVWF CCP1CON                    ; Učitavanje nove vrednosti u kontrolni reg. CCP1CON
```

Asemblerskom programskom sekvencom gornjeg primera briše se brojački registar preskalera CCP1 modula u Capture režimu i onemogućava pojava lažnog prekidnog signala.

U tabeli 6.6.4. je dat primer korišćenja CCP1 i CCP2 modula za precizno merenje širine dolazećeg impulsa.

```
////////////////////////////////////////////////////////////////////////////////
////                                CCP-Capture                                 ////
////                                                                            ////
////          Program meri sirinu jednog impulsa koji se dovodi na              ////
////       kratkospojene ulazne linije PORT-a C RC2 (CCP1) i RC1 (CCP2).        ////
////                                                                            ////
////////////////////////////////////////////////////////////////////////////////

#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

long    prednja, zadnja, sirina_impulsa;

#int_ccp2                    //Prekidna funkcija CCP2 u Capture ili Compare režimu
void isr()
{
   prednja = CCP_1;
   zadnja = CCP_2;

sirina_impulsa = zadnja - prednja;
}
                 //CCP_1 vreme do tranzicije impulsa sa niskog na visoki nivo
                 //CCP_2 vreme do tranzicije impulsa sa visokog na niski nivo

                 //Vreme trajanja impulsa je dato kao: sirina_impulsa/(Fclock/4) u us.
                 //Za ispravno merenje širine impulsa na bazi ovog programa,
                 //trajanje niskog nivoa impulsa mora biti duže od trajanja
                 //ISR rutine. Trajanje ISR rutine iznosi 45 instr. ciklusa ili 9 us.

void main()
```

```
{
 setup_ccp1(CCP_CAPTURE_RE);  //Konfiguracija CCP1 za hvatanje uzlazne ivice impulsa
 setup_ccp2(CCP_CAPTURE_FE);  //Konfiguracija CCP1 za hvatanje opadajuće ivice impulsa
 setup_timer_1(T1_INTERNAL);  //Inicijalizacija tajmera 1

   enable_interrupts(INT_CCP2);   //Prekid na opadajućoj ivici impulsa CCP2 modula
   enable_interrupts(GLOBAL);

   while(TRUE) {
      delay_ms(1000);
      printf("\r%lu us ", sirina_impulsa/5 );  //Slanje rezultata PC-u putem RS232
   }
}
```

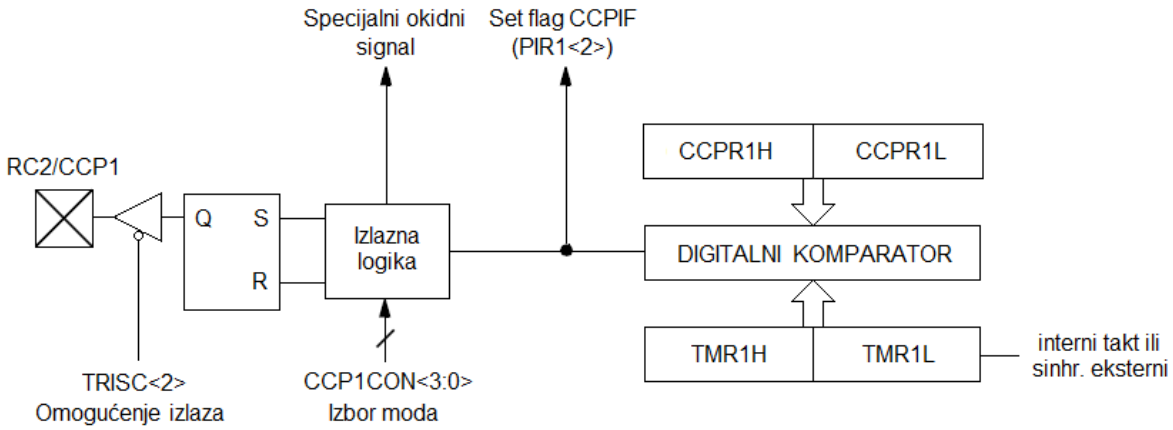
Tabela 6.6.4. *Primer programa za merenje širine impulsa koristeći oba CCP modula u režimu kaptiranja.*

6.6.2. CCP u režimu poređenja (Compare)

Jedan od dva CCP periferijska modula koristi se u ovom radnom režimu kao generator impulsa čije trajanje/pauza može biti precizno programski određeno. Naime, digitalna vrednost upisana u npr. 16-bitni registarski par CCP1 modula, CCPR1H:CCPR1L, stalno se poredi sa promenljivim sadržajem 16-bitnog registra tajmera 1, TMR1. Kada se sadržaji 16-bitnih registara izjednače izlazna linija modula RC2/CCP1

- se setuje (visoki naponski nivo)
- se resetuje (niski naponski nivo)
- stanje izlazne linije ostaje nepromenjeno

Na slici 6.6.2. je prikazana principijelna blok šema CCP1 periferije konfigurisane za režim poređenja. Izbor jednog od tri događaja na izlaznoj liniji RC2/CCP1 porta C vrši se konfigurisanjem bitova CCP1M3:CCP1M0 kontrolnog registra CCP1CON<3:0>. U isto vreme, u trenutku izjednačenja sadržaja, setuje se zastavica prekida CCP1IF registra PIR1<2>. Ovaj bit se programski mora resetovati. Kada je CCP1 periferijski modul konfigurisan u režimu poređenja za generisanje tkzv. programskog prekida modula, CCP1M3:CCP1M0=1010, prekid se događa (setovanje bita CCP1IF) u trenutku izjednačenja sadržaja registarskih parova CCPR1 i TMR1 ako je omogućen ali ne utiče na stanje izlazne linije RC2/CCP1.



Slika 6.6.2. *Blok šema CCP1 periferije u režimu poređenja.*

Linija RC2/CCP1 porta C, za ovaj radni režim CCP1 modula, mora biti konfigurisana kao izlazna, što se postiže resetovanjem drugog bita direkcionog registra TRISC<2>. Brisanje kontrolnog registra CCP1CON resetuje izlazno leč kolo RC2/CCP1 linije (RS flip-flop) uspostavljajući podrazumevani niski naponski nivo na izlaznoj liniji. Pomenuto leč kolo, međutim, ne predstavlja registarski leč podataka I/O porta C.

Kada se CCP modul koristi u režimu poređenja tajmer 1 modul mora biti konfigurisan u tajmerskom ili sinhronom brojačkom režimu. Asinhroni brojački režim tajmer 1 modula će onemogućiti rad CCP modula u režimu poređenja.

Specijalni okidni signal, koji se generiše interno hardverom CCP1 modula u režimu poređenja resetuje TMR1 registarski par tajmera 1. S tim u vezi 16-bitni registarski par CCPR1H:CCPR1L efektivno služi kao programabilni period registar tajmera 1.

Specijalni okidni signal koji generiše CCP2 modul, takođe resetuje TMR1 registar i dodatno startuje jednu A/D konverziju ako je A/D periferijski modul omogućen.

U tabeli 6.6.5. dat je komentarisani primer programskog koda na C jeziku za generisanje impulsa tačno određenog trajanja na zahtev, pritiskom na taster, uz pomoć CCP1 modula.

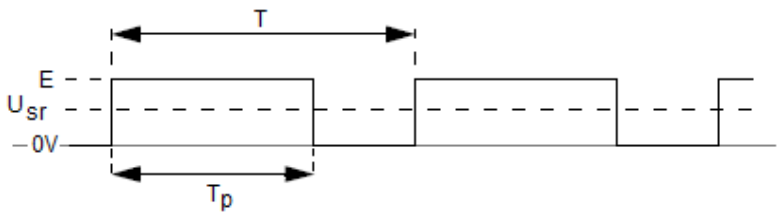
```
         ////////////////////////////////////////////////////////////////////
         ////                        CCP1-COMPARE                        ////
         ////                                                            ////
         ////  Program generiše impuls u trajanju 100us na RC2 liniji    ////
         ////  koristeći CCP1 modul, posle pritiska na taster (opadajuća ////
         ////  ivica impulsa na RB0 liniji).                             ////
         ////                                                            ////
         ////////////////////////////////////////////////////////////////////

#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT
#use delay(clock=20000000)

void main()
{
  setup_ccp1(CCP_COMPARE_CLR_ON_MATCH);  //Konfig. CCP1 modula u COMP. režimu RC2=Low
  setup_timer_1(T1_INTERNAL);    //Tajmerski režim TMR1 (inkr. na svaki instr. ciklus)

 while(TRUE)
 {
  while(input(PIN_B0)) ;               //Čekaj na pritisak tastera na RB0 liniji
```

```
 setup_ccp1(CCP_COMPARE_SET_ON_MATCH); //Konfig. CCP1 u COMP. za setovanje RC2 linije
                                       //u trenutku izjednačenja sadržaja CCP1 sa TMR1
    CCP_1=0;                           //Brisanje 16-bitnog registra CCP1 modula
    set_timer1(0);                     //RC2=High (izjednačeni sadržaji)

    CCP_1 = 500;                       //Upis u CCP1 registar
                              //Impuls trajanja 100us=(4/Fclock)*500= (4/20000000)*500

    setup_ccp1(CCP_COMPARE_CLR_ON_MATCH);      //Konfigurisanje CCP1 u COMPARE
                                               //režimu za resetovanje RC2 linije
                                               //po izjednačenju sadržaja CCP1 sa TMR1
    delay_ms(1000);                    //Debauns tastera
 }
}
```

Tabela 6.6.5. *Primer programa za generisanje impulsa preciznog trajanja na izlaznoj liniji RC2/CCP1, koristeći CCP1 modul u režimu poređenja.*

6.6.3. CCP u režimu generisanja PWM (Pulse Width Modulation) signala

Već je rečeno da integracija većeg broja periferijskih jedinica različite namene na jedinstvenom supstratu MCU povećava fleksibilnost i polje primene kola. Međutim, jedan od ozbiljnih nedostataka PIC16F877 i drugih Microchip-ovih MCU serija predstavlja nedostatak integrisanog neposrednog D/A konvertora za konverziju digitalnih veličina u analogne. Ovaj nedostatak je delimično otklonjen integracijom PWM modulatora koji u osnovi predstavlja posredni D/A konvertor. Koristi se činjenica da se digitalnom kontrolom širine pravougaonog impulsa u okviru periode ponavljanja može kontrolisati srednja (jednosmerna) vrednost napona periodičnog signala pravougaone forme. Na slici 6.6.3. je prikazan izgled periodičnog unipolarnog signala pravougaone forme sa svim karakterističnim parametrima.



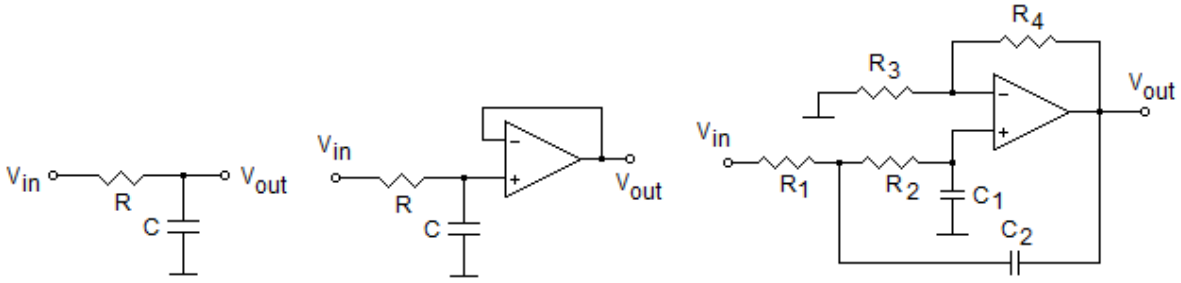
Slika 6.6.3. *Karakteristični parametri periodičnog unipolarnog signala pravougaone forme.*

Srednja vrednost napona periodičnog signala sa slike 5.6.3. može se odrediti iz izraza

$$U_{SR}=\frac{1}{T}\int_0^T u(t)dt=\frac{1}{T}\int_0^{T_P} Edt+\frac{1}{T}\int_{T_P}^{T} 0dt=\frac{T_P}{T}E=T_P f E$$

gde je T_P trajanje impulsa, f=1/T frekvencija a E maksimalna vrednost napona. Poslednji izraz pokazuje da je srednja ili jednosmerna vrednost napona periodičnog signala pravougaone forme direktno proporcionalna širini impulsa, frekvenciji i maksimalnoj vrednosti napona. Ako se frekvencija i maksimalna vrednost napona učine konstantnim tada srednja vrednost napona direktno zavisi samo od širine impulsa. S druge strane, pogodnim projektovanjem PWM generatora, na širinu impulsa u okviru periode ponavljanja može se uticati promenom digitalne reči što predstavlja funkciju D/A konvertora. Budući da je analogni ekvivalent napona konvertovane digitalne reči jednak srednjoj vrednosti napona pravougaone talasne forme to znači da se on može dobiti pomoću niskopropusnog filtra koji se priključuje na izlaz PWM generatora i koji se može realizovati kao pasivno RC kolo (ili više kaskadnih RC ćelija), slika 6.6.4. a) ili aktivno na bazi operacionog pojačavača, slika 6.6.4 c). Ako se vremenska konstanta filtra izabere tako da bude mnogo veća od periode ponavljanja pravougaonog napona $\tau=RC>>T$, rezultujući jednosmerni napon biće proporcionalan vremenu T_P, odnosno, digitalnom ekvivalentu. Da bi se omogućilo priključivanje snažnijih potrošača na izlaz pasivnog NF filtra bez uticaja na vremensku konstantu filtra, ne retko se njegov izlaz razdvaja od potrošača uz pomoć jediničnog pojačavača koji ima ulogu baferskog razdvojnog stepena kao na slici 6.6.4. b). Odnos trajanja impulsa prema periodi ponavljanja, koji figuriše u izrazu za srednju vrednost napona, T_P/T naziva se koeficijentom ispune signala (*duty cycle*).

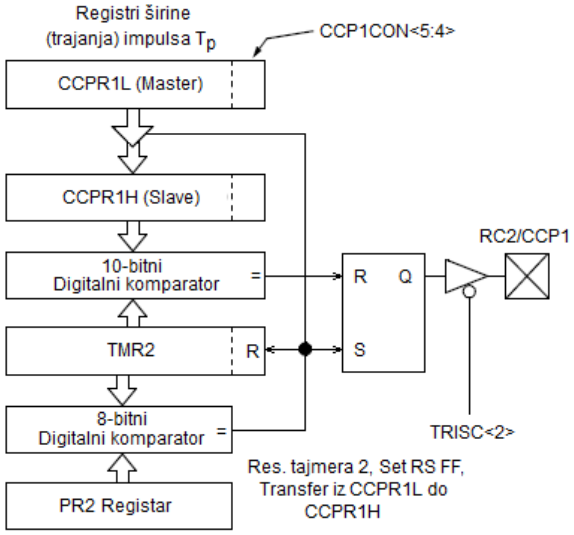
PWM generator ima široku primenu u mernim i telekomunikacionim sistemima, u kolima za kontrolu električne snage i konverziju električne energije u druge oblike, kao i u kolima za upravljanje AC i DC motorima.



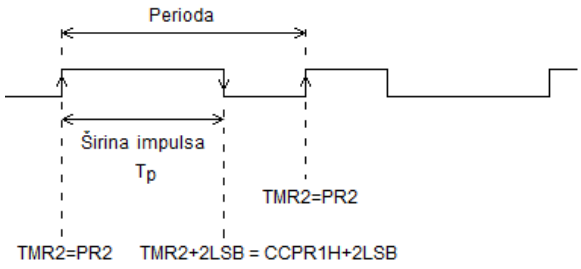
a) b) c)

Slika 6.6.4. *a) RC sekcija pasivnog NF filtra, b) pasivni NF filter sa baferskim izlaznim stepenom, c) bikvadratna sekcija aktivnog NF filtra.*

Dva CCP periferijska modula u sastavu PIC16F877 MCU opremljena su PWM generatorima rezolucije do 10 bita sa PWM signalima na multifunkcionalnim izlaznim linijama RC2/CCP1 i RC1/T1OSI/CCP2 porta C. Za rad u PWM režimu neophodno je pomenute linije porta C konfigurisati kao digitalne izlaze.



Slika 6.6.5. *Blok šema PWM generatora PIC16F877 MCU.*



Slika 6.6.6. *Vremenski parametri i karakteristični trenuci impulsno-širinski modulisanog signala PWM generatora.*

Brisanjem kontrolnog registra CCP1 modula CCP1CON, izlazno leč kolo (RS flip-flop) linije RC2/CCP1, slika 6.6.5, se automatski resetuje a izlazna linija postavlja na podrazumevani niski naponski nivo. Treba, međutim, imati u vidu da ovo leč kolo ne predstavlja izlazni leč podataka I/O porta C.

Na slici 6.6.6. je prikazan izgled impulsno-širinski modulisanog izlaznog signala koga karakterišu dva vremenska parametra: perioda (vremenska baza) i vreme za koje je izlazni signal visok (širina ili trajanje impulsa). Rad kola sa slike 6.6.5. se može lako objasniti ako se pođe od stanja izjednačenja sadržaja registara TMR2 i PR2, vidi sliku 6.6.6. Tada se izlazom 8-bitnog digitalnog komparatora

- resetuje registar TMR2
- setuje izlazno leč kolo (RS flip-flop) i time podiže naponski nivo na izlaznoj liniji PWM generatora RC2/CCP1, osim ako je digitalni ekvivalent širine impulsa jednak nuli,
- sadržaj registara CCPR1L i CCP1CON<5:4> (master), koji predstavlja digitalni ekvivalent širine (trajanja) impulsa, prenosi u registar CCPR1H koji je proširen za dva LSB bita (slave).

Registar CCPR1L sadrži viši osmobitni deo digitalnog ekvivalenta širine impulsa, dok su preostala dva bita najmanje težine dobijena iz kontrolnog registra CCP1CON<5:4>. Na taj način je formirana desetobitna rezolucija za digitalni ekvivalent širine impulsa. Sa slike 6.6.5. se vidi da je desetobitni digitalni ekvivalent širine impulsa dvostruko baferovan kako bi se izbegla pojava gličeva u izlaznom PWM signalu. Naime, upis u registre CCPR1L:CCP1CON<5:4> može se izvršiti u bilo kom trenutku ali se prenos sadržaja u CCPR1H registar koji je proširen za dva LSB bita neće dogoditi pre isteka tekuće periode, to jest pre izjednačenja sadržaja registara TMR2 i PR2 (TMR2=PR2).

Prenosom sadržaja CCPR1L registra u CCPR1H započinje novo poređenje promenljivog, inkrementirajućeg sadržaja tajmera 2 sa konstantnim sadržajem CCPR1H registra. Oba osmobitna registra su proširena sa dva LSB bita pa se izjednačenje njihovih sadržaja detektuje desetobitnim digitalnim komparatorom. U trenutku izjednačenja sadržaja komparatorskim izlazom se resetuje izlazni RS flip-flop, što uzrokuje promenu stanja na izlaznoj liniji RC2/CCP1 sa visokog na niski naponski nivo. Nadalje, tajmer 2 nastavlja sa inkrementiranjem i stalnim poređenjem svog promenljivog sadržaja sa konstantnim sadržajem 8-bitnog PR2 registra uz

pomoć 8-bitnog komparatora do trenutka izjednačenja oba sadržaja TMR2=PR2 posle čega se opisani ciklus ponavlja.

Periodu PWM signala određuje vrednost PR2 registra tajmer 2 periferijskog modula i izračunava se prema izrazu

$$PWM\_period = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{Odnos preskaliranja tajmera 2}) \quad (1)$$

gde je T_{OSC} perioода ponavljanja oscilacija taktnog oscilatora CPU, npr. kvarcnog. Frekvencija PWM signala je obrnuto proporcionalna PWM periodu (f_{PWM}=1/PWM_period) dok se postskaler tajmera 2 ne koristi za generisanje periode PWM signala.

Širinu (trajanje) impulsa PWM signala određuje 10-bitna digitalna reč koju formira osam MSB bitova CCPR1L registra i dva LSB bita kontrolnog registra CCP1CON<5:4>. Širina impulsa PWM signala se izračunava prema sledećem izrazu

$$PWM\_širina\_impulsa = (CCPR1L{:}CCP1CON\langle 5{:}4\rangle) \bullet T_{OSC} \bullet (\text{Odnos preskaliranja tajmera 2}) \quad (2)$$

Za zadatu frekvenciju PWM izlaznog signala, maksimalna rezolucija, izražena brojem raspoloživih bitova za kodiranje, se može izračunati iz izraza

$$rezolucija = \frac{\log \frac{f_{OSC}}{f_{PWM}}}{\log 2} \quad [broj\ bitova]$$

Poslednji izraz pokazuje da rezolucija opada sa povećanjem frekvencije generisanog PWM signala.

Za slučaj kada je vrednost širine impulsa veća od vrednosti periode ponavljanja, izlazna linija odgovarajućeg CCP modula neće biti resetovana, kao u slučaju kada je digitalni ekvivalent širine impulsa jednak nuli zbog čega izlazna linija ne može biti setovana.

Konfigurisanje CCP1 modula za PWM režim rada podrazumeva sledeće neophodne korake:

- izbor vrednosti periode ponavljanja PWM signala upisom u PR2 registar, u skladu sa izrazom (1),
- izbor vrednosti širine impulsa PWM signala upisom u CCPR1L:CCP1CON<5:4> registre, u skladu sa izrazom (2),
- konfiguraciju RC2/CCP1 linije porta C kao digitalnog izlaza resetovanjem bita TRISC<2>,
- izbor odgovarajuće vrednost preskalera tajmera 2 i uključenje tajmer 2 modula upisom odgovarajućeg bajta u kontrolni registar tajmera 2 T2CON,
- konfiguraciju CCP1 modula za PWM režim rada upisom odgovarajućeg bajta u kontrolni registar CCP1 modula CCP1CON.

PWM frekvencija	1.22KHz	4.88KHz	19.53KHz	78.12KHz	156.3KHz	208.3KHz
Tajmer 2 preskaler	16	4	1	1	1	1
Vrednost PR2 registra	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maks. rezolucija	10	10	10	8	7	5.5

Tabela 6.6.6. *Primeri PWM frekvencija i odgovarajuće vrednosti preskalera tajmera 2, period registra PR2 i maksimalne rezolucije generisanog signala.*

U tabeli 6.6.6. je dato nekoliko vrednosti frekvencija PWM izlaznog signala, odgovarajuće vrednosti preskalera tajmera 2 i period registra PR2, kao i njima odgovarajuće rezolucije pri frekvenciji kvarcnog taktnog oscilatora od 20MHz.

```
        ////////////////////////////////////////////////////////////////////
        ////                          CCP1 - PWM                        ////
        ////                                                            ////
        ////  Program čita 8-bitnu vrednost sa porta B koja određuje    ////
        ////  širinu impulsa (trajanje impulsa) u okviru periode ponav- ////
        ////  ljanja. Perioda ponavljanja, odnosno, frekvencija PWM     ////
        ////  signala bira se čitanjem RS232 porta računara, koristeći  ////
        ////  PC hiperterminal.                                         ////
        ////                                                            ////
        ////////////////////////////////////////////////////////////////////

#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT
#use delay(clock=10000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, BRGH1OK)

void main() {
   char selection;
   byte value;

   printf("\r\nFrekvencija:\r\n");         //Slanje konstantnog stringa PC terminalu
   printf("    1)   19.5 khz\r\n");
   printf("    2)   4.9 khz\r\n");
   printf("    3)   1.2 khz\r\n");

   do {
     selection=getc();
   } while((selection<'1')||(selection>'3'));  //do petlja dok se ne primi
                                               // karakter '1' do '3'
   setup_ccp1(CCP_PWM);                   //Konfigurisanje CCP1 za PWM rezim rada

         // Perioda generisanih impulsa je (4/Fclock)*Preskaler TMR2*(PR2+1)
         // U ovom programu je Fclock=10000000 i vrednost PR2=127
         // Za tri moguće selekcije frekvencija generisanih impulsa je:
         // (4/10000000)*1*128 = 51.2 us ili 19.5 kHz
         // (4/10000000)*4*128 = 204.8 us ili 4.9 kHz
         // (4/10000000)*16*128 = 819.2 us ili 1.2 kHz

   switch(selection) {
     case '1' : setup_timer_2(T2_DIV_BY_1, 127, 1);  //(preskaler TMR2, period
                                                     //registar PR2, postskaler TMR2)
                break;
     case '2' : setup_timer_2(T2_DIV_BY_4, 127, 1);
                break;
     case '3' : setup_timer_2(T2_DIV_BY_16, 127, 1);
                break;
   }

  while( TRUE ) {
    value = input_b();      //Čitanje 8-bitne vrednosti porta B koja odredjuje širinu
                            //impulsa PWM signala
```

```
    printf("%2X\r",value);  // Slanje informacije o vrednosti promenljive value PC-u

    set_pwm1_duty(value);   //Trajanje (širina) impulsa u okviru periode ponavljanja:

                            // Ako je value LONG INT tipa trajanje je
                            //            value*(1/Fclock)*Preskaler TMR2
                            // Ako je value INT tipa trajanje je
                            //            value*4*(1/Fclock)*Preskaler TMR2
                //Primer: za value=30 i preskaler TMR2=1 širina impulsa iznosi 12us
  }
}
```

Tabela 6.6.7. *Primer programa za konfigurisanje i generisanje PWM signala.*

U tabeli 6.6.7. je dat primer programa na C jeziku za generisanje PWM signala čija se frekvencija zadaje putem serijske komunikacije MCU sa PC računarom kao jedna od tri moguće opcije, dok širina impulsa zavisi od zadate osmobitne vrednosti na portu B i može biti promenljiva. Program opisuje način konfiguracije CCP1 modula za PWM režim rada.

6.7. A/D KONVERTOR

Najznačajniji integrisani periferijski modul kojim je opremljen MCU PIC16F877 je 10-bitni analogno-digitalni konvertor zbog činjenice da je velika većina fizičkih veličina analognog karaktera i potrebe za digitalnom obradom analognih veličina koja se pokazala celishodnijom od analogne. Desetobitna rezolucija konvertora nije industrijski standard (12-bita) ali je za većinu aplikacija zadovoljavajuća. Konvertor je zasnovan na registru sukcesivnih aproksimacija SAR i pripada klasi srednje brzih konvertora. Brzina konverzije iznosi nekoliko desetina mikrosekundi, uključujući i vreme akvizicije, što je dovoljno za npr. konverziju audio signala i više nego dovoljno za konverziju sporopromenljivih veličina. Međutim, brzina konverzije u izvesnoj meri zavisi od izbora taktnog generatora A/D modula i frekvencije taktovanja CPU jedinice. Osam A/D kanala formirano je na principu vremenskog multipleksa tako da se jedan relativno brzi konvertor može koristiti za konverziju više analognih veličina. Konvertor je unipolaran i može konvertovati analogne vrednosti napona iz opsega 0V do 5V ili manje. Za naponsku referencu A/D modula može se koristiti napon napajanja MCU pod uslovom da je dovoljno stabilan ili se može primeniti spoljašnja stabilna naponska referenca.

Konfgurisanje modula se vrši programiranjem dva kontrolna registra ADCON0 i ADCON1, dok se rezultat konverzije automatski smešta u registre podataka ADRESH i ADRESL.

ADCON0 REGISTAR

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
	ADCS1	**ADCS0**	**CHS2**	**CHS1**	**CHS0**	**GO/DONE**	**-**	**ADON**
	bit 7							bit 0
bit 7-6	**ADCS1:ADCS0:** Bitovi za izbor taktnog oscilatora A/D modula. 11=1:8 Vrednost odnosa preskaliranja. 11=1:4 Vrednost odnosa preskaliranja. 11=1:2 Vrednost odnosa preskaliranja. 11=1:1 Vrednost odnosa preskaliranja.							
bit 5-3	**CHS2:CHS0**: Bitovi za izbor analognog kanala. 000= Kanal 0 (RA0/AN0). 001= Kanal 1 (RA1/AN1). 010= Kanal 2 (RA2/AN2). . . . 110= Kanal 6 (RE1/AN6). 111= Kanal 7 (RE2/AN7).							
bit 2	**GO/DONE**: Statusni bit A/D konverzije. 1 = A/D konverzija u toku (setovanje ovog bita startuje jednu A/D konverziju). 0 = A/D konverzija završena (bit se automatski briše po kompletiranju konverzije).							
bit 1	**Neimplementiran**: Čita se kao '0'.							
bit 0	**ADON**: Bit za omogućenje/onemogućenje A/D modula. 1 = A/D modul uključen. 0 = A/D modul isključen (smanjena potrošnja).							

Tabela 6.7.1. *Sadržaj kontrolnog registra ADCON0 A/D konvertora.*

ADCON1 REGISTAR

	U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADFM	**-**	**-**	**-**	**PCFG3**	**PCFG2**	**PCFG1**	**PCFG0**
	bit 7							bit 0

bit 7 **ADFM:** Bit za izbor formata rezultata A/D konverzije.
1= Desno poravnat. 6 MSB bitova ADRESH registra čitaju se kao nule.
0= Levo poravnat. 6 LSB bitova ADRESL registra čitaju se kao nule.

bit 6-4 **Neimplementirani**: Čitaju se kao '0'.

bit 3-0 **PCFG3:PCFG0**: Kontrolni bitovi za konfiguraciju A/D porta.

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

A – analogni ulaz, D- digitalni I/O

Tabela 6.7.2. *Sadržaj kontrolnog registra ADCON1 A/D konvertora.*

Zahvaljujući autonomnom RC taktnom oscilatoru, A/D konvertor poseduje jedinstvenu mogućnost konverzije u toku sleep režima, dakle, u odsustvu digitalnog šuma.

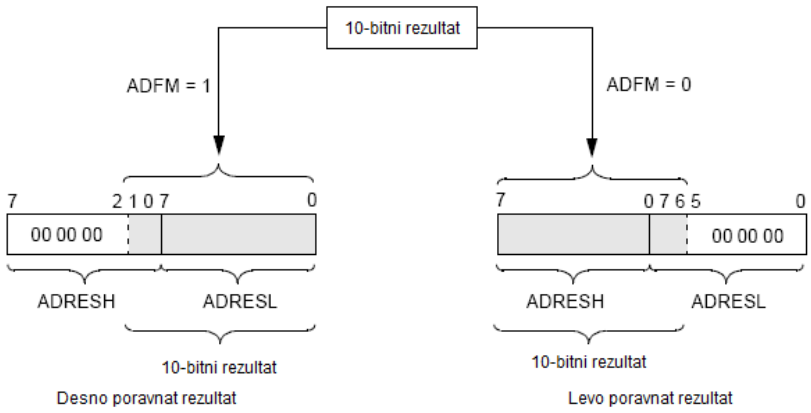
Tabela 6.7.1. prikazuje sadržaj kontrolnog registra ADCON0 koji služi za kontrolu rada A/D konvertorskog modula a tabela 6.7.2 sadržaj drugog kontrolnog registra ADCON1 koji se koristi za konfigurisanje linija porta A i porta E i formatiranje rezultata konverzije.

Registri rezultata konverzije ADRESH i ADRESL sadrže 10-bitni digitalni ekvivalent analognog napona. Na kraju jedne konverzije rezultat se automatski učitava u ovaj registarski par, resetuje bit GO/$\overline{\text{DONE}}$ kontrolnog registra ADCON0<2> i setuje zastavica prekida ADIF. 10-bitni rezultat konverzije sadržan u 16-bitnom registarskom paru ADRESH:ADRESL može biti formatiran na dva načina, kao na slici 6.7.1. Desno poravnat rezultat konverzije dobija se setovanjem kontrolnog bita za izbor formata ADFM registra ADCON1<7>, pri čemu šest najznačajnijih bitova ADRESH registra predstavljaju tkzv. vodeće nule. Levo poravnat rezultat konverzije sa nulama na mestu šest najmanje značajnih bitova ADRESL registra dobija se resetovanjem ADFM kontrolnog bita.

Bitovi direkcionog TRISA i TRISE registra, koji odgovaraju multipleksiranim analognim kanalima A/D konvertora, moraju biti setovani čime se odgovarajuće linije konfigurišu kao ulazne. Konfigurisanje analognih kanala porta A i porta E vrši se konfiguracionim kontrolnim bitovima PCFG3:PCFG0 kontrolnog registra ADCON1. Na primer, ako se želi da sva pet kanala porta A budu analogni ulazi a napon napajanja V_{DD} referentni napon A/D modula, bitove PCFG3:PCFG0 treba podesiti na vrednost 0010, vidi tabelu 6.7.2.

Izbor jednog od osam multipleksiranih analognih kanala vrši se podešavanjem bitova za selekciju odgovarajućeg analognog kanala CHS2:CHS0 kontrolnog registra ADCON0<5:3>. Na primer, ako se želi izbor kanala 1, RA1/AN1, selekcione bitove CHS2:CHS0 treba podesiti na vrednost 001, respektivno.

Jedan od četiri raspoloživa izvora taktnog signala A/D modula bira se podešavanjem kontrolnih bitova ADCS1:ADCS0 kontrolnog registra ADCON0<7:6>. Izbor utiče na brzinu konverzije, kao i na mogućnost izvršenja konverzije u toku sleep režima.



Slika 6.7.1. *Formatiranje 10-bitnog rezultata A/D konverzije.*

Setovanjem GO/$\overline{\text{DONE}}$ kontrolnog bita registra ADCON0<2> startuje se proces konverzije, dok se ovaj bit automatski resetuje po okončanju konverzije. Time se ovaj bit može iskoristiti za ispitivanje kraja konverzije metodom programske prozivke.

Uključenje/isključenje A/D modula se kontroliše bitom ADON registra ADCON0<0>. Kada je isključen modul ne troši energiju izvora za napajanje što smanjuje ukupnu potrošnju uređaja.

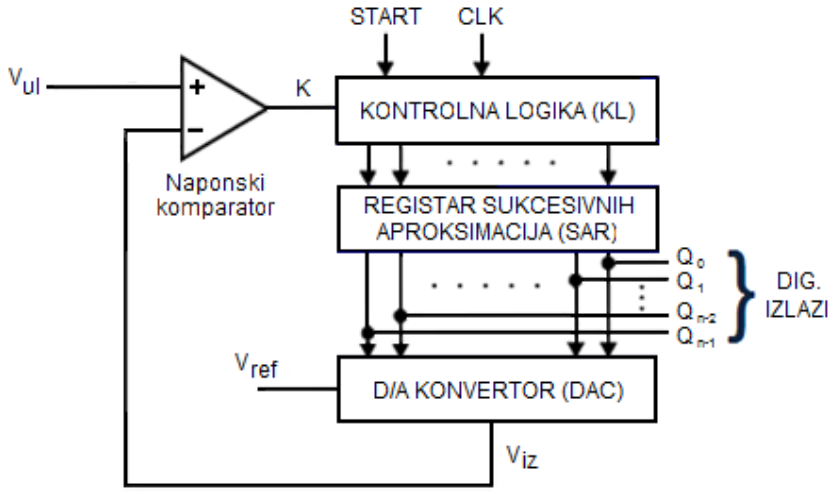
Već je rečeno da A/D modul koristi princip vremenskog multipleksa kako bi se dobilo osam analognih kanala uz primenu jednog srednje brzog SAR A/D konvertora. Na slici 5.1.3. petog poglavlja je prikazana principijelna šema osmokanalnog analognog multipleksora u sastavu A/D

modula. Kontrolnim bitovima za izbor kanala CHS2:CHS0 vrši se uključenje odgovarajućeg elektronskog prekidača multipleksora i time spaja odgovarajući analogni ulaz sa ulazom A/D konvertora. Prema istoj slici, programski se, uz pomoć kontrolnih bitova PCFG3:PCFG0, bira naponska referenca A/D konvertora kao eksterna naponska referenca sa linija porta A (RA2 i RA3) ili napon napajanja mikrokontrolera V_{DD}. Za eksternu naponsku referencu važe uslovi $V_{REF+} \leq V_{DD}$ i $V_{REF-} \geq 0V$.

6.7.1. A/D konvertor sa registrom sukcesivnih aproksimacija

A/D konvertori sa registrom sukcesivnih aproksimacija (SAR ADC) predstavljaju čest izbor konvertora za aplikacije sa malom potrošnjom kao što su baterijski napajani instrumenti, uređaji za industrijsku kontrolu, akviziciju podataka itd. SAR A/D konvertori pripadaju klasi srednje brzih konvertora sa brzinama konverzije od nekoliko μs do nekoliko desetina μs i rezolucijama od 8 do 16 bita. Kao što je rečeno, mikrokontroler PIC16F877 je opremljen 10-bitnim A/D konvertorom ove vrste. Principijelna blok šema A/D konvertora sa registrom sukcesivnih aproksimacija data je na slici 6.7.2.

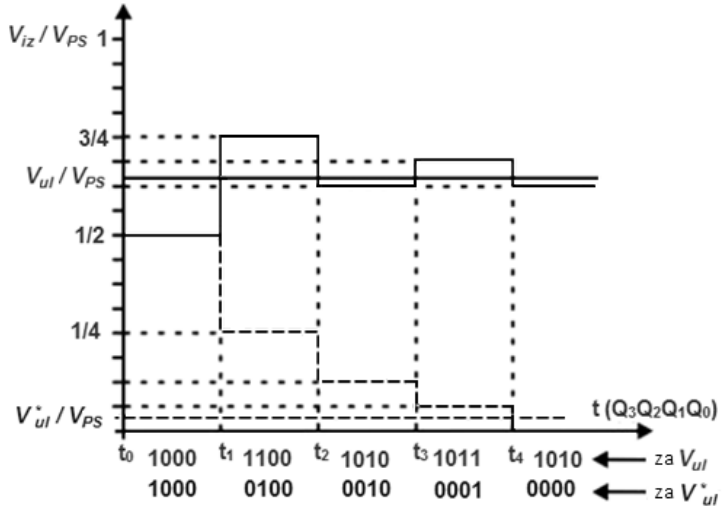
Maksimalni ulazni napon koji se može priključiti na konvertor je V_{ulmax} i jednak je naponu pune skale V_{ps} tj. $V_{ulmax}=V_{ps}$. Maksimalni napon na izlazu D/A konvertora V_{iz} je za 1 LSB manji od napona V_{ps}. Pod navedenim uslovima digitalno-analogni konvertor će, kada je MSB bit setovan ($Q_{n-1}=1$) a svi ostali bitovi resetovani, generisati napon jednak polovini pune skale $V_{iz}=V_{ps}/2$. Sledeći bit ($Q_{n-2}=1$) ima težinu $V_{ps}/4$, odnosno, pošto se radi o binarnom D/A konvertoru svaki sledeći bit ima težinu upola manju od prethodnog.



Slika 6.7.2. *Blok šema A/D konvertora sa registrom sukcesivnih aproksimacija.*

Konverzija počinje tako što kontrolna logika setuje bit najveće težine registra sukcesivnih aproksimacija Q_{n-1}. D/A konvertor tada generiše napon na svom izlazu vrednosti $V_{iz}=V_{ps}/2$ koji

se poredi sa ulaznim naponom V_{ul} uz pomoć analognog naponskog komparatora. Ako je $V_{ul}>V_{ps}/2$, MSB bit digitalnog ekvivalenta napona V_{ul} ostaje setovan, u suprotnom ovaj se bit resetuje. Kontrolna logika, na osnovu izlaza komparatora, resetuje (ako je k=0), ili ne resetuje (ako je k=1) MSB flip-flop registra sukcesivnih aproksimacija, čime je definisan bit najveće težine Q_{n-1} izlazne informacije. U sledećem taktnom intervalu kontrolna logika setuje bit Q_{n-2}. Komparator upoređuje novu vrednost izlaznog napona D/A konvertora V_{iz} (koja iznosi $V_{ps}/4$ ili $3V_{ps}/4$ u zavisnosti od rezultata poređenja u prethodnom taktnom intervalu) sa naponom na ulazu V_{ul}. Na bazi izlaza komparatora kontrolna logika resetuje (k=0) ili ne resetuje (k=1) flip-flop bita Q_{n-2} čime je definisana i vrednost bita drugog po težini (do MSB bita). Proces poređenja i setovanja/resetovanja sledećih flip-flopova se nastavlja sve do bita najmanje težine Q_0.



Slika 6.7.3. *Promena napona na izlazu D/A konvertora u toku A/D konverzije za dve različite vrednosti analognog napona na ulazu.*

U poslednjem n+1 taktnom intervalu (n-rezolucija konvertora), zavisno od stanja izlaza komparatora, flip-flop bita najmanje težine će se resetovati ili ostati setovan što predstavlja kraj procesa konverzije. Iz opisanog se vidi da se konverzija vrši po principu bit po bit i ima trajanje od n+1 taktnih intervala, gde je n rezolucija konvertora.

Kao primer, na slici 6.7.3. je prikazan vremenski dijagram izlaznog napona D/A konvertora V_{iz} idealnog četvorobitnog A/D konvertora sa registrom sukcesivnih aproksimacija. Na slici 6.7.3. je pretpostavljen ulazni napon $11/16>V_{ul}/V_{ps}>5/8$. Konverzija počinje upisom koda "1000" u SAR (registar sukcesivnih aproksimacija). Izlazni napon D/A konvertora se postavlja na $V_{iz}=V_{ps}/2$. Pošto je $V_{ul}>V_{iz}$, na osnovu k=1, u trenutku t_1 kontrolna logika ne resetuje Q_3, čime je određen MSB bit izlazne informacije. Potom kontrolna logika setuje bit Q_2=1, tako da je kod upisan u registar sukcesivnih aproksimacija (SAR) "1100", a izlazni napon D/A konvertora $V_{iz}=3V_{ps}/4$. Pod ovakvim okolnostima je $V_{ul}<V_{iz}$ pa kontrolna logika u trenutku t_2, na osnovu k=0, resetuje Q_2 čime je konačno određeno i stanje bita Q_2. U sledećem taktnom intervalu bezuslovno se setuje bit Q_1=1. SAR je postavljen u stanje "1010", a V_{iz} na $V_{iz}=5V_{ps}/8$. Pošto je $V_{ul}>V_{iz}$, a k=1 kontrolna logika neće resetovati bit Q_1. U trenutku t_3 kontrolna logika setuje i poslednji LSB bit Q_0=1, tako da je novo stanje SAR registra "1011". U trenutku t_4 se na osnovu k=0 resetuje Q_0 tako da je konačan rezultat konverzije u registru sukcesivnih aproksimacija $Q_3Q_2Q_1Q_0$="1010", što je najbliži digitalni ekvivalent ulaznog napona.

Na istom vremenskom dijagramu na slici 6.7.3. je isprekidanom linijom prikazan tok konverzije ulaznog napona za $V^*_{ul}<V_{ps}/16$. U pet koraka sukcesivnih aproksimacija, kao rezultat konverzije se dobija $Q_3Q_2Q_1Q_0$="0000".

Na osnovu sprovedene analize funkcionisanja A/D konvertora sa registrom sukcesivnih aproksimacija mogu se definisati uslovi za sintezu kontrolne logike (KL) i registra sukcesivnih aproksimacija (SAR):

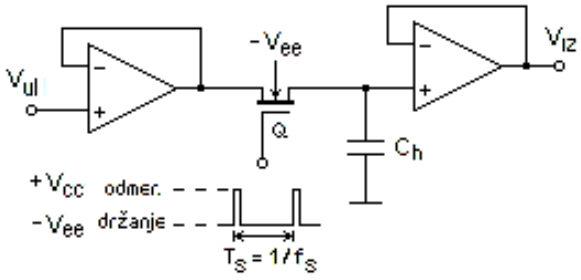
- Konverzija analognog signala u digitalni binarno kodovan broj od n cifara se obavlja se u n+1 taktnom intrvalu, od kojih prvi (t_0) inicijalizuje sekvencu sukcesivnih aproksimacija, poslednji (t_n) označava kraj konverzije.
- Start konverzije se zadaje asinhrono u odnosu na taktni impuls A/D konvertora.
- Perioda taktnih impulsa mora biti duža od zbira vremena postavljanja D/A konvertora, vremena postavljanja izlaza komparatora i propagacije kroz kontrolnu logiku.
- Korišćeni n-bitni D/A konvertor mora imati monotono rastuću karakteristiku prenosa (diferencijalnu linearnost bolju od ±0.5 LSB).
- Flip-flipovi registra sukcesivnih aproksimacija (SAR registra) treba da imaju mogućnost pojedinačnog setovanja i resetovanja.

Ulazni napon V_{ul} ne sme da se menja u toku konverzije s obzirom da se flip-flopovi postavljaju bit po bit a već postavljeni flip-flopovi, u slučaju promene napona na ulazu V_{ul}, ne mogu da promene stanje. Da bi se osigurala nepromenljivost ulaznog napona u toku konverzije, koji inače može biti promenljiv u vremenu, uvodi se princip tkzv. odmeravanja i držanja (praćenja i pamćenja) ulaznog napona. Na slici 6.7.4. je prikazano kolo za odmeravanje i držanje (S/H–*Sample and Hold*) koje se koristi u kolima A/D konvertora. Analogni ulazni signal uvodi se u A/D konvertor preko bloka za odmeravanje u kome se vrši vremensko kvantovanje signala. Naime, u pravilnim vremenskim razmacima T_S (uniformno odmeravanje), pod kontrolom impulsa za odmeravanje, formiraju se odgovarajući amplitudski odmerci analognog signala, slika 6.7.5. Ovako odabrani odmerci predstavljaju diskretizovan kontinualni signal i u suštini su analogni podaci.

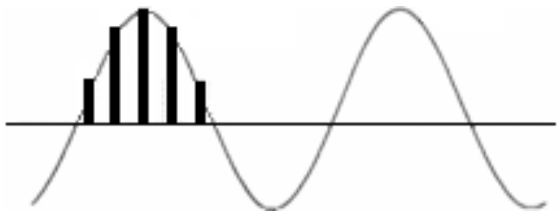
Frekvencija odmeravanja f_S zavisi od frekvencijskog spektra kontinualnog ulaznog signala. Da bi se iz spektra povorke odmeraka mogao da regeneriše kontinualni signal pomoću niskopropusnog filtra, najniža frekvencija odmeravanja mora zadovoljiti uslov

$$f_{S\min} \geq 2f_g$$

gde f_g predstavlja najvišu spektralnu komponentu (harmonik) ulaznog analognog signala.



Slika 6.7.4. *S/H (Sample & Hold) kolo za odmeravanje i držanje.*



Slika 6.7.5. *Odmeravanje prostoperiodičnog napona.*

Kolo za odmeravanje treba da obezbedi prenos trenutne vrednosti signala sa ulaza na izlaz samo u toku kratkog trajanja impulsa za odmeravanje, kao na slici 6.7.4. U odsustvu impulsa (u toku trajanja pauze) otvoreni prekidač treba da osigura dobru izolovanost memorijskog kondenzatora C_h od generatora ulaznog signala. Najjednostavnije takvo kolo sastoji se od analognog tranzistorskog prekidača Q sa npr. unipolarnim NMOS tranzistorom. Prekidač je zatvoren kada je na gejtu tranzistora uspostavljen visoki napon V_{CC} čija vrednost mora biti veća od zbira maksimalne vrednosti ulaznog napona i napona indukcije kanala. $V_G=V_{CC}>V_{ulmax}+V_T$. U oblasti malih napona V_{DS} prekidač se ponaša kao mala linearna otpornost čija vrednost zavisi

od napona V_{GS}. Za negativni napon na gejtu $V_G=-V_{EE}$ tranzistorski prekidač je otvoren sa vrlo velikom otpornošću između drejna i sorsa.

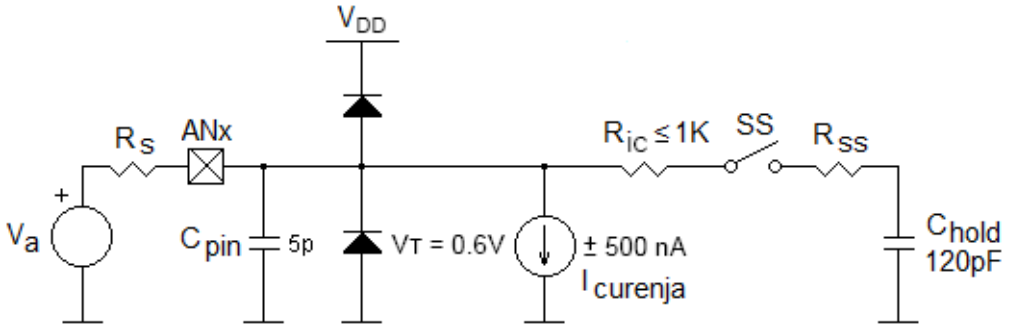
Memorijsku ćeliju u kolu na slici 6.7.4. čini kondenzator C_h dok operacioni pojačavači, čija su pojačanja jednaka jedinici, imaju ulogu razdvojnih stepena tj. ulogu konvertora impedanse. Operacioni pojačavač na ulazu S/H kola predstavlja neznatno opterećenje (velika ulazna otpornost) za izvor analognog napona V_{ul} dok izlazni pojačavač predstavlja neznatno opterećenje za memorijski kondenzator. S druge strane mala izlazna otpornost pojačavača na ulazu omogućava brzo punjenje memorijskog kondenzatora preko zatvorenog tranzistorskog prekidača budući da je vremenska konstanta punjenja kondenzatora $\tau=C_h(R_{IZ}+R_{TR})$, gde je R_{IZ} izlazna otpornost ulaznog pojačavača reda nekoliko desetina oma a R_{TR} otpornost kanala provodnog tranzistora reda nekoliko oma. Mala vremenska konstanta punjenja memorijskog kondenzatora je od značaja zbog mogućnosti skraćivanja trajanja impulsa za odmeravanje. U periodu između impulsa, formirano naelektrisanje u kondenzatoru se održava približno konstantnim imajući u vidu veliku otpornost otvorenog tranzistorskog prekidača i veliku ulaznu otpornost izlaznog razdvojnog pojačavača. Pomenuti period između dva impulsa mora biti najmanje jednakog ili trajanja dužeg od vremena konverzije analognog signala.

6.7.2. Vreme akvizicije A/D kanala

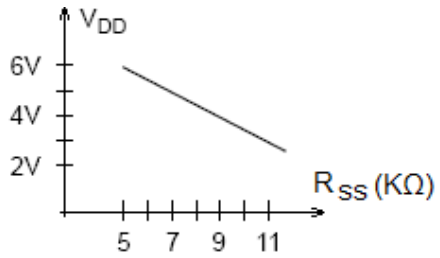
Za postizanje specificirane tačnosti A/D konverzije ±½LSB, pre starta svakog konverzionog procesa se prethodno mora osigurati dovoljno dugo vreme punjenja kondenzatora C_{HOLD} u kolu za odmeravanje i držanje (*sample and hold*) tako da se on napuni na vrednost ulaznog napona. Drugim rečima, posle selekcije odgovarajućeg ulaznog analognog kanala potrebno je osigurati da protekne određeni vremenski period (vreme akvizicije-T_{ACQ}) posle koga se može startovati A/D konverzija. U suprotnom, pojava grubih grešaka rezultata konverzije je sasvim izvesna.

Na slici 6.7.6. je prikazana principijelna ekvivalentna šema jednog ulaznog analognog kanala i kola za odmeravanje i držanje S/H. Ulazni kanal je modelovan parazitskom kapacitivnošću na ulaznoj liniji porta C_{PIN}=5pF, otpornošću međuveze ulaznog kanala i S/H kola $R_{IC}\leq$1kΩ i ulaznom strujom curenja $I_{LEAKAGE}=\pm$500nA koja je posledica prisustva različitih PN spojeva kao što su prenaponske zaštitne diode i elektronski prekidač SS, prikazani na istoj slici. Sa V_T=0.6V je označen napon praga provođenja silicijumskog PN spoja.

Kapacitivnost kondenzatora C_{HOLD} je 120pF dok otpornost elektronskog prekidača R_{SS} zavisi od napona napajanja uređaja i prikazana je na slici 6.7.7.



Slika 6.7.6. *Modelovanje ulaznog analognog kanala.*



Slika 6.7.7. *Zavisnost otpornosti elektronskog prekidača S/H kola od napona napajanja.*

Unutrašnja otpornost analognog naponskog generatora na ulazu R_S, otpornost međuveze R_{IC} i otpornost prekidača R_{SS} direktno utiču na vreme potrebno za punjenje kondenzatora C_{HOLD}. Maksimalna preporučena unutrašnja otpornost R_S analognog generatora na ulazu je 10kΩ imajući u vidu vrednost struje curenja. Za manje otpornosti R_S vreme punjenja kondenzatora je kraće, i suprotno.

Vreme akvizicije je dato kao zbir vremena potrebnog za odziv ulaznog pojačavačkog kola (*Amplifier settling time*), vremena punjenja kondenzatora C_{HOLD} i vremenskog intervala u funkciji temperaturskog koeficijenta vremenske konstante ulaznog kanala i S/H kola (C_{HOLD} (R_{IC} + R_{SS} + R_S)), to jest:

T_{ACQ} = Vreme odziva pojač. + Vreme punjenja C_{HOLD} + Temp. koef. vrem. konst.

Ili izraženo analitički

$$T_{ACQ} = T_{AMP} + T_C + T_{COEFF}$$

Vreme odziva pojačavačkog kola na ulazu je T_{AMP} = 2μs, dok za specificiranu tačnost A/D konvertora ±½LSB pri 10-bitnoj rezoluciji maksimalno vreme punjenja kondenzatora C_{HOLD} iznosi prema slici 5.7.6.

$$T_C = C_{HOLD} (R_{IC} + R_{SS} + R_S) \ln(1023.5/(½)) = 120pF (1kΩ + 7kΩ + 10kΩ) \ln(2047) = 16.47μs$$

Za izračunavanje vremenskog intervala u funkciji temperaturskog koeficijenta vremenske konstante ulaznog kanala i S/H kola T_{COEFF} treba uzeti najgori slučaj tj. maksimalnu pozitivnu razliku aktuelne (radne) i sobne temperature MCU.

Ukupno vreme akvizicije na visokoj radnoj temperaturi MCU od 50°C iznosi

$$T_{ACQ} = 2μs + 16.47μs + [(50°C - 25°C)(0.05μs/°C)] = 19.72μs$$

Prema tome, posle izbora odgovarajućeg analognog kanala programer mora obezbediti vremensku zadršku od maksimalno 20μs za najgori slučaj (maksimalna dopuštena unutrašnja otpornost analognog naponskog izvora i na maksimalnoj radnoj temperaturi) posle koje može startovati proces konverzije, kako bi se osigurala specificirana tačnost konverzije.

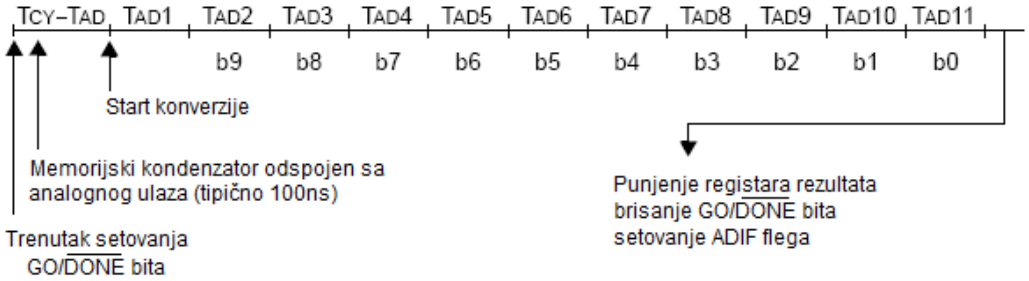
Trenutak izbora kanala je i trenutak zatvaranja elektronskog prekidača SS S/H kola, odnosno, početak punjenja kondenzatora C_{HOLD}, dok start konverzije predstavlja trenutak otvaranja prekidača SS posle čega se vrednost analognog napona na ulazu, 'zapamćena' na kondenzatoru C_{HOLD} konvertuje u digitalnu reč.

Posle završene jedne konverzije, neophodno je obezbediti vremensku zadršku od dve periode taktnog signala $2T_{AD}$ pre nego započne novi proces akvizicije. U toku ove vremenske zadrške kondenzator C_{HOLD} nije povezan sa izabranim ulaznim analognim kanalom. Po isteku zadrške akvizicija izabranog kanala automatski započinje.

6.7.3. Izbor taktnog generatora A/D modula

Četiri vrste generatora taktnog signala A/D modula, koje se programski mogu birati, stoje na raspolaganju korisniku. Ako se sa T_{AD} definiše vreme A/D konverzije po bitu tada ovo vreme predstavlja i periodu taktnog signala A/D modula budući da se u jednom taktnom intervalu uspostavlja vrednost jednog bita rezultata konverzije. Ukupno vreme potrebno za jednu konverziju iznosi maksimalno $12T_{AD}$ za 10-bitnu rezoluciju konvertora.

Na slici 6.7.8. je ilustrovan broj potrebnih T_{AD} ciklusa za izvođenje jedne regularne konverzije. Posle starta konverzije, setovanjem GO bita, trajanje prvog vremenskog segmenta iznosi minimalno jedan instrukcijski ciklus T_{CY} a maksimalno jedan taktni interval T_{AD}. Još jedanaest taktnih intervala potrebno je za konverziju analognog napona u desetobitni digitalni ekvivalent. Kao što se sa slike 6.7.8. može videti, $12T_{AD}$ vremenskih intervala je neophodno za kompletiranje jedne konverzije.



Slika 6.7.8. *Potreban broj taktnih intervala za izvođenje jedne regularne konverzije.*

Za korektnu A/D konverziju trajanje taktnog intervala ne sme biti kraće od T_{ADmin}=1.6μs, što odgovara maksimalnoj frekvenciji taktnih impulsa od 625kHz.

Kontrolnim bitovima ADCS1:ADCS0 kontrolnog registra ADCON0<7:6> može se izabrati jedna od četiri opcija za taktovanje A/D modula, i to:

- $T_{AD}=2T_{OSC}$
- $T_{AD}=8T_{OSC}$
- $T_{AD}=32T_{OSC}$
- Interni RC oscilator A/D modula (T_{AD}=4μs tipično)

gde je $T_{OSC}=1/f_{OSC}$ perioda kvarcnog oscilatora.

Perioda taktnog signala koji generiše interni RC oscilator A/D modula varira između 2μs i 6μs a tipična vrednost iznosi 4μs. Ako se, na primer, za izvor taktnog signala izabere druga opcija $T_{AD}=8T_{OSC}$ tada, uz pomenuti uslov korektne konverzije, frekvencija kvarcnog oscilatora ne sme biti veća od

$$f_{OSCmax}=1/T_{OSC}=8/T_{ADmin}=5MHz$$

Drugim rečima, ako je npr. frekvencija kvarcnog oscilatora f_{OSC}=4MHz, prva od gornjih opcija za izbor taktnog signala se ne bi smela birati kada se ima u vidu da je

$$T_{AD}=2T_{OSC}=2/f_{OSC}=0.5\mu s \leq T_{ADmin}=1.6\mu s$$

Ostale tri opcije zadovoljavaju uslov korektne konverzije.

Resetovanje GO/$\overline{\text{DONE}}$ bita u toku trajanja konverzije predstavlja nasilni prekid tekućeg procesa konverzije. Sadržaj registarskog para ADRESH:ADRESL neće biti promenjen u odnosu na rezultat poslednje regularne konverzije. Posle prevremenog prekida tekuće konverzije, takođe je potrebno obezbediti zadršku od $2T_{AD}$ pre starta sledeće akvizicije, koja automatski startuje po izboru kanala.

6.7.4. Rad A/D modula u sleep režimu

Kao što je rečeno, A/D modul može raditi i u toku sleep režima mikrokontrolera pod uslovom da je za generator taktnog signala izabran interni RC oscilator modula (ADCS1:ADCS0 = 11). A/D konverzija u toku sleep režima je povoljnija sa stanovišta digitalnog šuma koji se na ovaj način eliminiše.

Za izvođenje konverzije u toku sleep režima, pored izbora internog RC oscilatora, potrebno je izvršiti SLEEP instrukciju odmah nakon setovanja GO/$\overline{\text{DONE}}$ bita, odnosno, starta konverzije. Ako je omogućen, prekidni signal A/D modula će probuditi (*wake-up*) uređaj iz sleep stanja i nastaviti sa normalnim izvođenjem programa, u suprotnom, A/D modul se posle izvršene konverzije isključuje premda ADON bit ostaje setovan.

Pokušaj da se A/D konverzija izvrši u sleep režimu kada izvor taktnog signala A/D modula nije interni RC oscilator, rezultuje prekidom započete konverzije i isključivanjem A/D modula premda, i u ovom slučaju, ADON bit ostaje setovan. Isključivanjem A/D modula značajno se smanjuje njegova sopstvena potrošnja.

Reset mikrokontrolera isključuje A/D modul, prekida eventualno započetu konverziju i rekonfiguriše sve analogne linije portova A i E kao analogne ulaze. Posle POR reseta stanje ADRESH:ADRESL registarskog para je neodređeno, dok posle $\overline{\text{MCLR}}$ i WDT reseta stanje registarskog para ostaje nepromenjeno u odnosu na stanje pre reseta.

```
////////////////////////////////////////////////////////////////////////
////                              AD_Statistika                     ////
////                                                                ////
////  Program prikazuje min, max i srednju vrednost (mean) 30       ////
////  uzastopnih odmeraka A/D konverzije internog A/D konvertora    ////
////  MCU (10-bitni konvertor) sa nultog kanala. Preko RS232 interfejsa ////
////  rezultat se šalje hiperterminalu računara.                    ////
////                                                                ////
////////////////////////////////////////////////////////////////////////

#include <16F877.h>
#device ADC=10
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

void main() {
```

```
   int i;
   long value, min, max;
   float mean=0;

   printf("Uzimanje odmeraka:");

   setup_port_a( ALL_ANALOG );         // Sve linije porta A izuzev RA4 su analogne
   setup_adc( ADC_CLOCK_INTERNAL );    // Interni RC takt A/D modula (vreme konverzije
                                       // po bitu 2-4 us)
   set_adc_channel( 0 );               // Izbor kanalo 0 (RA0)

   do {
      min=0xffff;
      max=0;
      for(i=0; i<=30; ++i) {
         delay_us(20);
         value = Read_ADC();          // Čitanje digitalne reci A/D modula sa kanala 0
         if(value<min)
            min=value;
         if(value>max)
            max=value;

    mean=mean+value;
      }
      mean=mean/30;
      printf("\n\rMin: %4LX  Max: %4LX Mean: %2F\r\n",min,max,mean);

   } while (TRUE);
}
```

Tabela 6.7.3. *Primer programa na C jeziku za rad sa A/D konvertorom.*

U tabeli 6.7.3. je prikazan program napisan na C jeziku koji konfiguriše kanal nula A/D konvertora i potom vrši 30 uzastopnih konverzija izračunavajući pri tome minimalnu, maksimalnu i srednju vrednost rezultata konverzije. Dobijene vrednosti se šalju računaru putem RS232 komunikacionog interfejsa.

6.8. UNIVERZALNI SINHRONO/ASINHRONI PRIJEMNIK/PREDAJNIK (USART)

USART je jedan od dva integrisana serijska komunikaciona interfejsa kojima raspolaže PIC16F877 mikrokontroler (*SCI – Serial Communication Inteface*). Ova periferija može biti konfigurisana u tkzv. full duplex asinhronom režimu za komunikaciju sa spoljašnjim uređajima kao što su personalni računar, monitor terminali sa katodnom cevi itd., ili u tkzv. half duplex sinhronom modu za komunikaciju sa eksternim uređajima kao što su integrisani A/D i D/A konvertori, serijske EEPROM memorije itd.

Kao što je napred rečeno, USART može biti konfigurisan u jednom od sledećih radnih režima:

- Asinhroni (full duplex)
- Sinhroni Master (half duplex)
- Sinhroni Slave (half duplex)

Modul takođe poseduje mogućnost multiprocesorske komunikacije koristeći 9-bitnu detekciju adrese.

Full-duplex režim prenosa podataka znači mogućnost slanja i primanja podataka u isto vreme preko Tx i Rx linije, respektivno. Kada je konfigurisan, USART modul kontroliše prijemnu Rx i predajnu Tx liniju mikrokontrolera tako da ove linije ne mogu biti korišćene kao I/O linije opšte namene.

Sinhroni master mod je zasnovan na tkzv. half-duplex prenosu (predaja i prijem podataka nisu istovremeni). Konfiguriše se setovanjem bita SYNC statusnog i kontrolnog registra predajnika TXSTA<4>, vidi tabelu 6.8.1. Dodatno, setovanjem bita SPEN statusnog i kontrolnog registra prijemnika RCSTA<7> konfigurišu se RC6/TX/CK i RC7/RX/DT I/O linije kao CK (clock) i DT (data) linije, respektivno, tabela 6.8.2.

U sinhronom master modu procesor generiše taktni signal na CK liniji. Master mod se konfiguriše setovanjem bita CSRC registra TXSTA<7>.

U sinhronom slave modu takt generiše spoljašnji uređaj što dopušta MCU da prenese ili primi podatak dok je u sleep režimu. Slave mod se konfiguriše resetovanjem bita CSRC registra TXSTA<7>.

TXSTA REGISTAR

	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
	CSRC	**TX9**	**TXEN**	**SYNC**	**-**	**BRGH**	**TRMT**	**TX9D**
	bit 7							bit 0
bit 7	**CSRC:** Bit za izbor izvora taktnog signala. **Asinhroni mod rada:** Nema uticaja **Sinhroni mod rada:** 1 = Master mod (takt generiše interno BRG generator). 0 = Slave mod (takt generiše spoljašnji izvor taktnog signala).							
bit 6	**TX9**: Bit za omogućenje devetobitne transmisije. 1 = Devetobitna transmisija. 0 = Osmobitna transmisija.							
bit 5	**TXEN**: Bit za omogućenje/onemogućenje predajnika. 1 = Predajnik omogućen. 0 = Predajnik onemogućen.							
bit 4	**SYNC**: Bit za selekciju radnog režima USART-a. 1 = Sinhroni mod. 0 = Asinhroni mod.							
bit 3	**Neimplementiran**: Čita se kao nula.							
bit 2	**BRGH**: Bit za izbor brzine serijskog prenosa. **Asinhroni mod:** 1 = Veće brzine prenosa. 0 = Manje brzine prenosa. **Sinhroni mod:** Ne koristi se u ovom radnom režimu.							
bit 1	**TRMT**: Statusni bit pomeračkog registra predajnika. 1 = Pomerački registar predajnika prazan. 0 = Pomerački registar predajnika pun.							
bit 0	**TX9D**: Deveti bit serijskog podatka, može biti bit parnosti.							

Tabela 6.8.1. *Sadržaj statusnog i kontrolnog registra predajnika USART periferijskog modula.*

RCSTA REGISTAR

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
	SPEN	**RX9**	**SREN**	**CREN**	**ADDEN**	**FERR**	**OERR**	**RX9D**
	bit 7							bit 0
bit 7	**SPEN:** Bit za omogućenje/onemogućenje serijskog porta. 1 = Serijski port omogućen (konfiguriše RC7/Rx i RC6/Tx linije kao linije serijskog porta). 0 = Serijski port onemogućen.							
bit 6	**RX9**: Bit za omogućenje devetobitnog prijema. 1 = Devetobitni prijem. 0 = Osmobitni prijem.							
bit 5	**SREN**: Bit za omogućenje/onemogućenje prijema jednog bajta. **Asinhroni mod:** Nema uticaja. **Sinhroni mod - master:** 1 = Prijem omogućen. 0 = Prijem onemogućen. **Sinhroni mod - slave:** Nema uticaja.							
bit 4	**CREN**: Bit za omogućenje/onemogućenje kontinuiranog prijema podataka. **Asinhroni mod:** 1 = Kontinuirani prijem omogućen. 0 = Kontinuirani prijem onemogućen. **Sinhroni mod - master:** 1 = Kontinuirani prijem omogućen (omogućava kontinuirani prijem dok se CREN bit ne resetuje, CREN bit poništava uticaj SREN bita). 0 = Kontinurani prijem onemogućen.							
bit 3	**ADDEN**: Bit za omogućenje/onemogućenje detekcije adrese. **Asinhroni mod 9-bitni (RX9=1):** 1 = Omogućena detekcija adrese, omogućen prekid i punjenje prijemnog bafera kada je RSR<8> setovan. 0 = Onemogućena detekcija adrese, svi bajtovi su primljeni a deveti bit može biti korišćen kao bit parnosti.							
bit 2	**FERR**: Bit za detekciju greške prijema okvira podatka. 1 = Greška pri prijemu okvira (može biti ažurirana čitanjem RCREG prijemnog registra i prijemom sledećeg validnog bajta). 0 = Nema greške pri prijemu okvira.							
bit 1	**OERR**: Bit za detekciju greške prekoračenja. 1 = Greška prekoračenja (bit može biti resetovan resetovanjem CREN bita). 0 = Nema greške prekoračenja.							
bit 0	**RX9D**: Deveti bit primljenog podatka (može biti bit parnosti ali se mora izračunavati pomoću korisničkog programa.							

Tabela 6.8.2. *Sadržaj statusnog i kontrolnog registra prijemnika USART periferijskog modula.*

USART modulu MCU su pridruženi registri za kontrolu predaje i prijema 8-bitnog podatka, registar za upis podatka koji se šalje, registar za čitanje primljenog podatka, jedan registar za podešavanje brzine serijskog prenosa i registri za kontrolu prekida koje generiše ovaj modul.

Registri za kontrolu transmisije i prijema		***Registri za upis i čitanje podataka***	
Ime registra	*Opis*	*Ime registra*	*Opis*
TXSTA	Statusni i kontrolni registar predajnika	TXREG	Registar podataka za transmisiju

RCSTA	Statusni i kontrolni registar prijemnika	RCREG	Registar podataka za prijem

Registri za kontrolu prekida	
Ime registra	*Opis*
INTCON	Kontrolni registar prekida
PIR1	Registar flegova periferijskih prekida
PIE1	Registar za omogućenje prekida sa periferija

Registar za podešavanje brzine serijskog prenosa	
Ime registra	*Opis*
SPBRG	Registar Baud Rate generatora

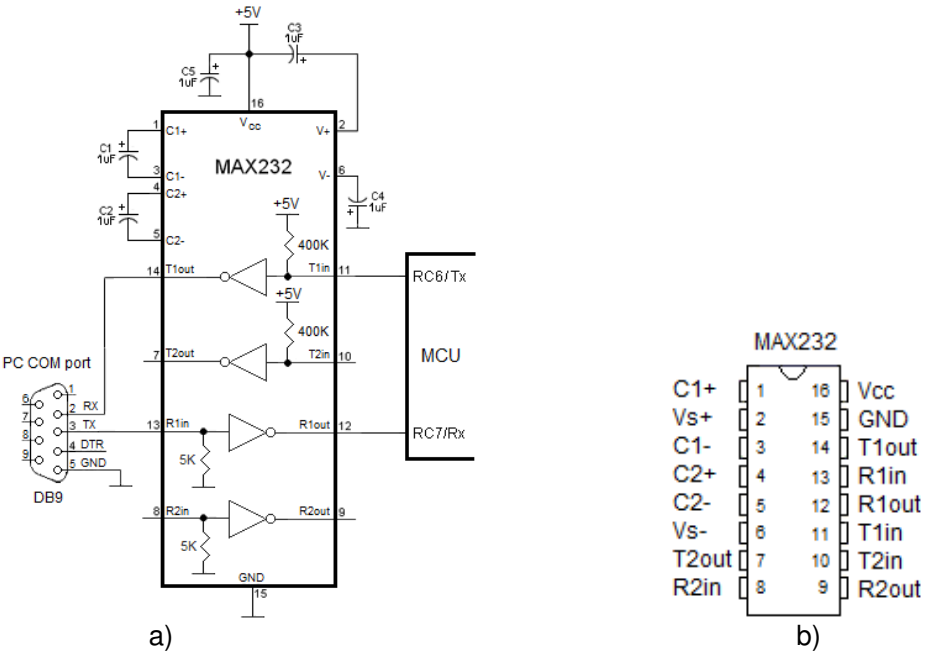
6.8.1. Asinhrona serijska komunikacija

Serijska komunikacija je jednostavan način prenosa podataka između mikrokontrolera i računara ili mikrokontrolera i periferija kao što su A/D konvertori, serijski EEPROM i programabilni uređaji zasnovani na mikroračunaru. Većina PC računara raspolaže sa jednim ili više serijskih portova zbog čega je ova forma komunikacije popularna, uključujući i činjenicu da se prenos informacija može vršiti na relativno dužoj distanci između računara i periferije. Takođe, veliki broj savremenih mernih, akvizicionih i komunikacionih uređaja opremljen je serijskim portom.

Osnovni nedostaci serijske komunikacije su relativno mala brzina prenosa kao i činjenica da serijski port po RS-232 protokolu može komunicirati samo sa jednom periferijom. Brzina prenosa podataka, međutim, u određenim aplikacijama ne mora biti ograničavajući faktor.

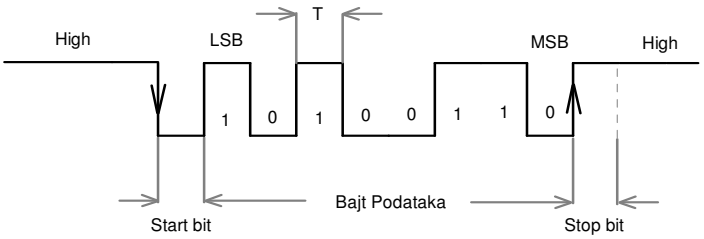
Za komunikaciju sa serijskim COM portom PC računara (RS-232 komunikacija) linijski RS-232 interfejs mora biti korišćen za prilagođenje naponskih nivoa logičke jedinice i nule na strani mikrokontrolera (+5V i 0V) i računara (+/-12V), respektivno. U čipu MAX232 je u tom cilju implementiran naponski udvajač (kolo za udvostručavanje napona sa 5V na 10V) i kolo naponskog invertora za generisanje negativnog napona od -10V.

Na slici 6.8.1 a) je prikazan način međusprege I/O linija USART modula mikrokontrolera PIC16F877 i PC računara primenom integrisanog driver/receiver-a MAX232 čija pin konfiguracija je data na istoj slici pod b).



Slika. 6.8.1. a) *Povezivanje MAX232 integrisanog driver/receiver-a sa USART modulom mikrokontrolera i PC COM portom, b) pin konfiguracija MAX232 čipa.*

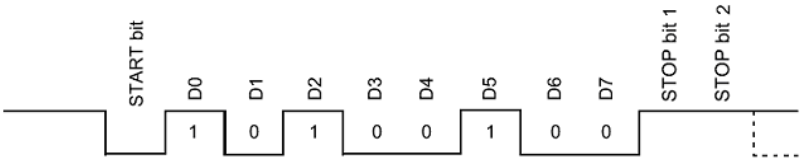
Komunikacioni protokol zasnovan je na standardnom NRZ (*Non Return to Zero*) formatu serijskog prenosa prikazanom na slici 6.8.2. Budući da je prenos asinhron (u odsustvu taktnog signala), sinhronizacioni parametar prenosa predstavlja brzina prenosa koja mora biti jednaka za predajnik i prijemnik. Serijski prenos započinje spuštanjem linije za transmisiju podataka od strane predajnika na niski naponski nivo u trajanju jedan bitski interval T, slika 6.8.2. Ovaj prvi bit predstavlja tkzv. startni bit kojim predajnik inicira prijemnik za prenos jednog bajta. Posle startnog bita sledi osam bitova podatka, dok kraj prenosa prijemnik detektuje na rastućoj ivici desetog bitskog intervala – stop bit. Interval između opadajuće i rastuće ivice impulsa na slici 6.8.2. predstavlja jedan okvir.



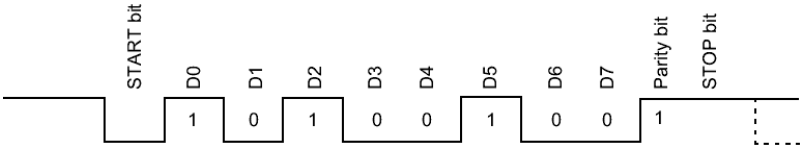
Slika 6.8.2. *Standardni NRZ (Non Return to Zero) format asinhronog serijskog prenosa (START bit + 8 bitova podatka + STOP bit).*

Format prenosa može biti i drugačiji. Naime, ponekad se umesto jednog stop bita koriste dva uzastopna, slika 6.8.3, dok se u drugom slučaju ispred jednog stop bita umeće bit parnosti, slika 6.8.4. Bit parnosti (parna ili neparna parnost) se koristi za kontrolu ispravnosti prenosa podataka i otkrivanje greške u prenosu koja nastaje na jednoj bitskoj poziciji. Npr., bit parnosti može biti setovan za slučaj kada je broj logičkih jedinica bajta za prenos neparan. Na prijemnoj strani se proverava stanje bita parnosti da bi se detektovala eventualna greška pri prenosu na

jednoj bitskoj poziciji. Odgovarajućim algoritmima je docnije moguće ispravljanje greške na prijemnoj strani.



Slika 6.8.3. *Asinhroni serijski prenos bajta podatka u formatu START bit + 8 bitova podatka + dva STOP bita.*



Slika 6.8.4. *Serijski prenos devetobitnog podatka u formatu START bit + bajt podatka + bit parnosti + STOP bit.*

Brzina serijskog prenosa obrnuto je proporcionalna trajanju bitskog intervala T dok je trajanje prenosa jednog bajta podatka, u zavisnosti od izabranog formata, proizvod trajanja bitskog intervala i ukupnog broja bitova, uključujući start, stop i eventualno bit parnosti. Npr., za zadatu brzinu serijskog prenosa Baud Rate = 9600 bita/s trajanje bitskog intervala T je:

$$T = 1 / \text{Baud Rate} = 1/ 9600 = 104.16\ \mu s$$

Za zadati format prenosa, npr., start bit + 8 bitova podatka + stop bit, trajanje prenosa jednog okvira iznosi 10 x T = 1.0416 ms.

Trajanje bitskog intervala određuje namenski BRG (Baud Rate Generator) generator koji podržava oba radna režima USART-a, asinhroni i sinhroni. Koristi se za dobijanje standardnih brzina serijskog prenosa podataka. BRG generator predstavlja 8-bitni tajmer sa slobodnim oscilatorom kao izvorom takta. Period 8-bitnog tajmera kontroliše se uz pomoć 8-bitnog SPBRG kontrolnog registra. U asinhronom radnom režimu, bit BRGH registra TXSTA<2> takođe kontroliše brzinu prenosa dok se u sinhronom modu ovaj bit ignoriše.

U tabeli 6.8.3. su dati izrazi za izračunavanje brzine serijskog prenosa za različite radne režime USART-a.

Sync	BRGH=0 (Za manje brzine prenosa)	BRGH=1 (Za veće brzine prenosa)
0	(Asinhroni) Baud Rate=$F_{OSC}/(64(x+1))$	Baud Rate=$F_{OSC}/(16(x+1))$
1	(Sinhroni) Baud Rate=$F_{OSC}/(4(x+1))$	-

Tabela 6.8.3. *Izračunavanje brzine serijskog prenosa za različite radne režime USART-a.*

Parametar X u tabeli 6.8.3. predstavlja celobrojnu vrednost (0-255) upisanu u SPBRG registar BRG generatora. Iz gornjih jednačina se jednostavno izračunava celobrojna vrednost SPBRG registra X za dva slučaja:

$X = (f_{OSC}/(16*\text{Baud Rate})) - 1$ za BRGH = 1 (high speed)

$X = (f_{OSC}/(64*\text{Baud Rate})) - 1$ za BRGH = 0 (low speed)

Za zadatu vrednost brzine prenosa (Baud Rate), iz gornjih formula se izračunava brojna vrednost X SPBRG registra koja se zaokružuje na najbliži ceo broj, čime se dobija realna brzina prenosa. Eventualna razlika realne i zadate brzine predstavlja grešku. Da bi se postigla veća fleksibilnost u odnosu na razliku realne i zadate brzine, izborom vrednosti bita BRGH (0 ili 1) omogućeno je izračunavanje brojne vrednosti X na dva načina od kojih se usvaja onaj za koji je pomenuta razlika u brzinama manja. Na primer, ako je frekvencija rezonanse kvarcnog oscilatora f_{OSC} = 4 MHz a zadata brzina serijskog prenosa Baud Rate = 9600 baud

Za BRGH = 1

X = 4000000/(16 x 9600) - 1 = 25.04

Za BRGH = 0

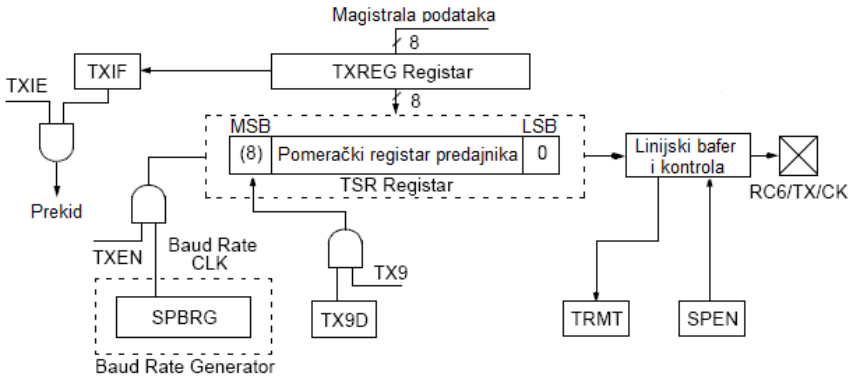
X = 4000000/(64 x 9600) - 1 = 5.51

Bolji izbor predstavlja slučaj BRGH = 1, X = 25, zbog manje razlike zadate i realne brzine.

Asinhroni radni režim se selektuje resetovanjem SYNC bita kontrolnog registra TXSTA<4>, tabela 6.8.1., a stopira u toku sleep moda mikrokontrolera. Predajnik i prijemnik su funkcionalno nezavisni, izuzev što koriste isti format podatka i brzinu prenosa. BRG generator generiše taktni signal multipliciran 16 ili 64 puta (zavisno od stanja bita BRGH) što odgovara određenoj brzini pomeranja podatka u pomeračkim registrima predajnika i prijemnika. Predaja ili prijem podatka počinje prenosom LSB bita podatka kao prvog bita. Bit parnosti nije podržan hardverski ali se može implementirati softverski i uskladištiti kao deveti bit podatka. Podatak na prijemnoj liniji RC7/RX/DT odmerava se tri puta u svakom bitskom intervalu pomoću kola za većinsku detekciju kako bi se pouzdano utvrdilo prisustvo visokog ili niskog naponskog nivoa na liniji.

6.8.2. Asinhroni predajnik USART-a

Blok dijagram asinhronog predajnika USART modula prikazan je na slici 6.8.5. Srce predajnika predstavlja pomerački registar TSR. Programski upisan podatak u registar predajnog bafera TXREG transferuje se u TSR registar u jednom instrukcijskom ciklusu, posle čega se setuje zastavica prekida TXIF registra (PIR1<4>) indicirajući na taj način pražnjenje registra TXREG. Punjenje registra predajnog bafera TXREG novim sadržajem neće biti moguće dok se ne završi prenos poslednjeg stop bita tekuće transmisije. Zastavica prekida TXIF će se setovati bez obzira na stanje mask bita TXIE i ne može biti resetovana softverski. Ona se automatski resetuje po upisu novog sadržaja u registar TXREG. Zastavica prekida TXIF se, takođe, automatski setuje posle setovanja bita za omogućenje/onemogućenje asinhronog predajnika TXEN.



Slika 6.8.5. *Pojednostavljena blok šema dela USART-a za asinhroni prenos podataka.*

TSR – (*Transmit shift register*) pomerački registar predajnika (nije mapiran u memoriji podataka i nije dostupan korisniku).
SPEN – bit za omogućenje/onemogućenje serijskog porta.
TRMT – statusni bit pomeračkog registra predajnika (TRMT=1-TSR prazan, TRMT=0-TSR pun).
SPBRG – registar BGR generatora za izbor brzine transfera podataka.
TXEN – bit za omogućenje/onemogućenje serijskog predajnika.
TXREG – registar podataka predajnog bafera.

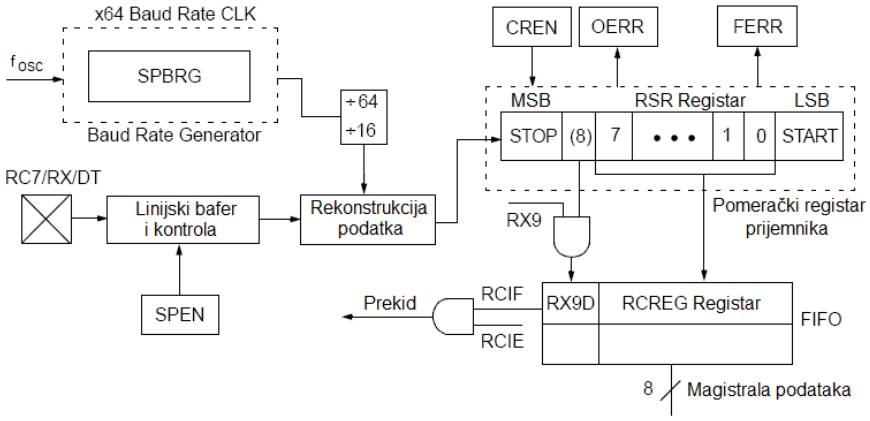
Transmisija se omogućava setovanjem bita TXEN (TXSTA<5>) (pod uslovom da je omogućen asinhroni serijski port) ali neće započeti do trenutka upisa podatka u TXREG. Posle toga sledi prenos sadržaja u pomerački registar TSR i potom serijsko pomeranje bitova uz pomoć BRG generatora takta. Transmisija takođe može startovati ako se podatak najpre upiše u TXREG i potom setuje TXEN bit. Resetovanje bita TXEN u toku transmisije podatka rezultuje prekidom transmisije, resetom predajnika i postavljanjem izlazne linije predajnika RC6/TX/CK u stanje visoke impedanse. Moguć je i uzastopni prenos većeg broja reči (*back to back*) ako se osigura uzastopno i pravovremeno punjenje registra TXREG imajući u vidu da se on automatski prazni transferom njegovog sadržaja u pomerački TSR registar.

Dok TXIF zastavica indicira status registra predajnog bafera TXREG, drugi bit TRMT (TXSTA<1>) ukazuje na status pomeračkog registra predajnika TSR. Ovaj bit se automatski setuje kada je TSR registar prazan. Nema prekidne logike u vezi s ovim bitom tako da korisnik prozivkom TRMT bita može utvrditi kada je TSR registar prazan, odnosno, detektovati kraj transmisije.

Za izbor 9-bitnog prenosa bit TX9 (TXSTA<6>) mora biti setovan a deveti bit upisan u TX9D bitsku poziciju registra TXSTA<0>, slika 6.8.5., pre upisa bajta podatka u registar TXREG. Budući da se sadržaj upisan u registar TXREG transferuje u pomerački registar TSR u jednom instrukcijskom ciklusu a potom započinje prenos to bi naknadni upis devetog bita posle završenog transfera bio nekorektan.

6.8.3. Asinhroni prijemnik USART-a

Blok dijagram asinhronog prijemnika USART modula prikazan je na slici 6.8.6.



Slika 6.8.6. *Pojednostavljena blok šema dela USART-a za asinhroni prijem podataka.*

RSR – (*Receive shift register*) pomerački registar prijemnika (nije mapiran u memoriji podataka i nije dostupan korisniku).
CREN – bit za omogućenje/onemogućenje kontinuiranog prijema podataka.
OERR – (*Overrun error*) bit za detekciju greške prekoračenja.
FERR – (*Framing error*) bit za detekciju greške prilikom prijema okvira.
SPEN – bit za omogućenje/onemogućenje serijskog porta.
SPBRG – registar BGR generatora za izbor brzine transfera podataka.
RCREG – registar podataka prijemnog bafera.

Asinhroni prijem se omogućava setovanjem bita CREN (RCSTA<4>), pod uslovom da je setovanjem bita SPEN bio omogućen asinhroni serijski port. Serijski podatak, koji prihvata ulazna linija RC7/RX/DT, rekonstruiše se u pomeračkom registru velike brzine, na slici 6.8.6. označenom kao blok za rekonstrukciju podatka (*data recovery*).

Srce prijemnika je pomerački registar prijemnika, RSR registar. Posle primljenog STOP bita (kraj prijema) 8-bitni podatak u RSR registru transferuje se u registar prijemnog bafera RCREG (ako je prazan) a bit zastavice prekida RCIF setuje (PIR1<5>). Ovaj samočitljivi bit resetuje se hardverski posle čitanja registra RCREG, odnosno, posle njegovog pražnjenja.

Registar prijemnog bafera RCREG je realizovan kao dvostruko baferovan FIFO registar što dopušta mogućnost prijema i transferovanja RCREG FIFO registru dva bajta podatka, kao i prijem trećeg bajta u RSR pomerački registar. Ako je, međutim, po detekciji STOP bita trećeg primljenog bajta registar RCREG još uvek pun (nije pročitan) generiše se greška prekoračenja setovanjem bita *overrun error* OERR (RCSTA<1>). Rezultat pojave ove vrste greške je gubljenje podatka u RSR registru. Dva bajta podatka u FIFO registru RCREG mogu biti dobijena dvostrukim uzastopnim čitanjem ovog registra. OERR bit se briše softverski resetovanjem prijemne logike, odnosno, resetovanjem bita CREN i potom njegovim setovanjem. Inače, setovani bit OERR sprečava transfer sadržaja pomeračkog registra RSR u RCREG, kao i dalji prijem podataka. S toga je od krucijalne važnosti postupak brisanja ovog bita, u slučaju njegovog setovanja.

Greška prijema okvira podatka rezultuje setovanjem bita *framing error* FERR (RCSTA<2>), a nastaje u slučaju kada se stanje poslednjeg STOP bita detektuje kao stanje logičke nule (niski

naponski nivo). S obzirom da se čitanjem registra RCREG i prijemom novog validnog podatka, bitske pozicije FERR i RX9D pune novim vrednostima, čitanje statusnog i kontrolnog registra prijemnika RCSTA pre nego što se pročita FIFO registar RCREG onemogućiće gubljenje starih vrednosti ova dva bita.

CCS C kompajler koristi direktivu #use rs232 za konfigurisanje USART modula, kojom se specificiraju linije MCU za prijem Rx i predaju Tx serijskog podatka. Normalne C I/O funkcije, kao što je printf() ili puts(), koriste ove linije kao njihovu vezu sa standardnim I/O kanalima. Ako se npr. za linije Tx i Rx specificiraju pinovi PIN_C6 i PIN_C7, respektivno, tada će one biti korišćene kao I/O linije hardverski ugrađenog USART periferijskog modula. U slučaju primene MCU koji nema integrisan USART modul pomenute linije će predstavljati I/O linije softverski kreiranog USART-a. Prema tome, CCS C kompajler nudi mogućnost korišćenja hardverski ugrađenog ili softverski kreiranog USART-a tako da korisnik može raspolagati i većim brojem asinhronih komunikacionih interfejsa. Naravno, glavna prednost hardverskog modula je postojanje namenskog hardvera koji autonomno upravlja obostranim prenosom podataka dok se u slučaju softverske realizacije prenos vrši pod kontrolom programa, tj. uz učešće procesora.

```
            ;*********** POTPROGRAM ZA INICIJALIZACIJU USART-a ***********
Serial_Setup:
       Bank1                    ;selekcija banke 1
       Movlw    .129            ;baud rate = 9.6k za 20Mhz taktne frekvencije
       Movwf    SPBRG           ;upis vrednosti 129 u SPBRG registar
       Movlw    H'24'           ;upis H'24' u TXSTA reg.
       Movwf    TXSTA           ;inicijalizacija TXSTA reg. (BRGH=1, SYNC=0, TXEN=1)
       Bank0                    ;selekcija banke 0
       Movlw    H'90'           ;upis H'90' u RCSTA reg.
       Movwf    RCSTA           ;inicijalizacija RCSTA reg. (SPEN=1, CREN=1)
       Return
           ;********* POTPROGRAMI ZA PRIJEM i PREDAJU BEZ PREKIDA *********
Serial_Receive:
        Bank0
        Btfss   PIR1,RCIF       ;proveriti da li je primljen podatak?
        Goto    $-1             ;ne, proveravati do prijema podatka.
        Movf    RCREG,W         ;prenos primljenog podatka u W reg. i autom. brisanje
        Return                  ;RCIF zastavice prekida

Serial_Transmit:
        Bank0
        Btfss   TXSTA,TRMT      ; proveriti da li je pom. reg. predajnika TSR prazan?
        Goto    $-1            ;ne, proveravati dok se ne isprazni.
        Return                 ;TSR prazan (podatak poslat).

;Programska sekvenca za omogućenje prekida pri prijemu i slanju podatka.

       Movlw    0xC0           ;globalno omogućenje prekida (GIE) i periferijskih
       Movwf    INTCON         ;prekida (PEIE)
       Bank1
       Movlw    0x30           ;omogućenje prekida pri prijemu Rx i slanju podatka Tx
       Movwf    PIE1           ;upisom 0x30 u PIE1 registar.
       Bank0
```

Tabela 6.8.4. *Asemblerske programske sekvence za inicijalizaciju i rad sa USART-om.*

U tabeli 6.8.4. prikazani su jednostavni potprogrami za inicijalizaciju USART periferijskog modula, prijem i predaju podataka metodom prozivke, kao i asemblerska programska sekvenca za omogućenje prekida pri predaji i prijemu podatka.

USART modul u asinhronom režimu rada može rukovati 8-bitnom ili 9-bitnom dužinom podatka. U najvećem broju aplikacija koristi se 8-bitni prenos. Postoji nekoliko razloga zbog čega se u nekim slučajevima koristi 9-bitni prenos:

- raspoloživi podatak je dužine 9 bitova,
- podatak je 8-bitni a zahtevaju se dva stop bita,
- podatak je 8-bitni a zahteva se bit pariteta,
- podatak je 8-bitni a zahteva se 9-ti bit za detekciju adrese u slučaju multiprocesorske komunikacije (kada je 9-ti bit setovan šalje se adresa a kada je resetovan šalje se podatak)

TX9 bit u TXSTA registru i RX9 bit u RCSTA registru moraju biti setovati da bi se omogućila devetobitna transmisija i prijem. Redosled čitanja i upisa podataka je veoma važan kada se radi sa devetobitnim operacijama. Deveti bit uvek treba upisivati ili čitati pre preostalih 8 bitova.

Program prikazan u tabeli 6.8.5. predstavlja primer C programa za asinhronu serijsku komunikaciju. Deklarisana je prekidna funkcija za prijem podataka, ključne reči sa kojom se poredi zadati string. U slučaju jednakosti PC računaru se šalje isti string pročitan s desno u levo.

```
#include <16F877.h>                                 // header fajl (include datoteka)
#fuses XT,NOWDT,NOPROTECT,NOLVP,PUT                 // konfiguraciona reč MCU
#use delay(clock=4000000)                           // rezon. frekvencija kvarc kristala
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)  // direktiva za konfig. USART-a MCU

#include <stdio.h>                        // standardna bibl. za rad sa RS232
#include  <string.h>                     //  stand.  bibl.  za  rad  sa  stringovnim  prom.

#define  STRING_SIZE    6               // Broj karaktera ključne reči

                                 // deklaracija stringovne konstante za ključnu reč

  char KEY[STRING_SIZE] = {'D','O','N','G','L','E'};
  char s_password[6];                          // deklaracija prom. za unos password-a

#int_rda                  // deklaracija prekidne f-je asinh. prijema ser. podatka
void serial_isr() {       // prekidna funkcija
   gets(s_password);      // čitanje niza karaktera u stringovnu promenljivu s_password
}

void main()                        // glavna funkcija
{
   enable_interrupts(global);   // omogućenje globalnog prekida
   enable_interrupts(int_rda);  // omogućenje prekida po prijemu serijskog podatka

STP:                              // start point

if(strcmp(s_password, KEY))     // provera da li je s_password="DONGLE"
      printf("ELGNOD\r");       // ako jeste, pošalji string "ELGNOD" PC-u ser. portom
   else
      goto  STP;                // ako ne, skoči na STP
}
```

Tabela 6.8.5. *Programski primer primene USART periferijskog modula u asinhronom režimu.*

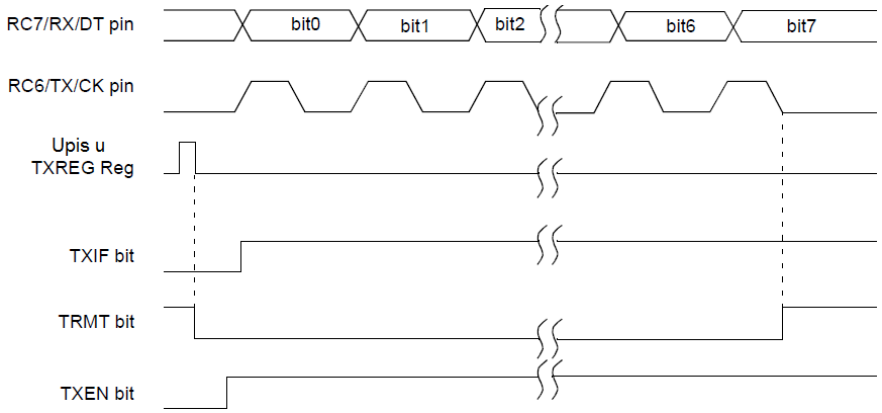
Asinhroni RS232 standard je tkzv. nebalansirani standard ili standard prenosa podataka sa jednim krajem (*single ended*), što znači da prijemnik meri razliku potencijala signalne linije i referentnog kraja. Dužina vodova koji spajaju predajnik sa prijemnikom kod RS232 standarda je relativno mala i zavisi od usvojene brzine prenosa, npr. do 15m pri brzini prenosa od 20 kbaud-a. Iako su referentni krajevi predajnika i prijemnika uobičajeno povezani povratnim transmisionim vodom, na većim distancama je moguća pojava značajne razlike potencijala između referentnih krajeva predajnika i prijemnika koja degradira imunitet prenosa na šum. Tako će bilo koji spoljašnji šum različito uticati na signalnu liniju u odnosu na refrentne krajeve zbog njihovih nebalansiranih električnih karakteristika.

6.8.4. USART u sinhronom master modu

U sinhronom master modu prenos podataka se vrši u polu dupleks (*half duplex*) režimu. Kada se podatak šalje prijem je onemogućen i obrnuto. Sinhroni mod se konfiguriše setovanjem bita SYNC registra TXSTA<4>. Dodatno, setovanjem bita za omogućenje serijskog porta SPEN registra RCSTA<7>, I/O linije USART-a RC6/Tx/CK I RC7/Rx/DT konfigurišu se kao taktna i linija podataka, respektivno. U sinhronom master modu takt na izlaznoj CK liniji generiše interni BRG generator USART modula setovanjem bita CSRC statusnog i kontrolnog registra predajnika TXSTA<7>.

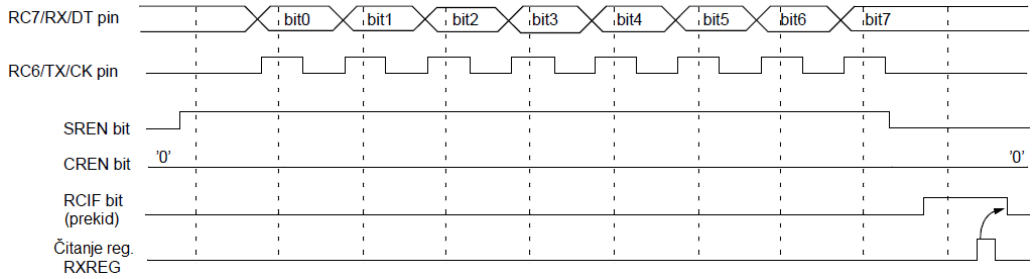
Transmisija se omogućava setovanjem bita TXEN registra TXSTA<5> (ako je omogućen sinhroni serijski port, setovan bit za sinhroni prenos SYNC i setovan bit za izbor taktnog signala CSRC). Sa izuzetkom inicijalne konfiguracije predajnika i sinhronizacije svakog prenetog bita taktnim intervalima, postupak slanja bajta podatka identičan je opisanom asinhronom načinu slanja. Pomeranje bitova na izlaznoj liniji podataka DT sinhronizovano je rastućom ivicom taktnog signala dok je podatak stabilan na opadajućoj ivici sinhro takta.

Na slici 6.8.7. prikazani su talasni oblici karakterističnih signala u toku sinhronog slanja jednog bajta podatka.



Slika 6.8.7. *Vremenski dijagram sinhronog slanja podatka USART modulom u master modu.*

Kada je selektovan sinhroni mod, prijem podatka je omogućen setovanjem bilo bita SREN registra RCSTA<5> ili bita CREN istog registra RCSTA<4>, kao i bita za izbor internog taktnog signala CSRC. Bit na liniji podataka DT odmerava se na opadajućoj ivici taktnog impulsa. Setovanjem bita SREN omogućen je prijem jednog bajta, dok se setovanjem CREN bita omogućava kontinuirani sinhro prijem bajtova do resetovanja istog bita. U slučaju setovanja oba bita SREN i CREN bit CREN ima prednost. Posle sinhronog pomeranja poslednjeg bita, podatak primljen u pomerački registar prijemnika RSR transferuje se u registar RCREG, ako je prazan, istovremeno setujući zastavicu prekida RCIF. Registar RCREG je dvostruko baferovan registar. Sa razlikom u inicijalnoj konfiguraciji prijemnika i sinhronizovanog prenosa podataka, opisana procedura asinhronog prijema podataka identična je sinhronoj. Na slici 6.8.8. prikazani su talasni oblici karakterističnih signala u toku prijema jednog bajta podatka.



Slika 6.8.8. *Vremenski dijagram sinhronog prijema podatka USART modulom u master modu.*

6.8.5. USART u sinhronom slave modu

Glavna razlika sinhronog slave moda u odnosu na master sastoji se u činjenici da je u slave modu sinhronizacioni pomerački taktni signal generisan spolja a ne interno BRG generatorom kao u slučaju master moda. Ovakav način taktovanja dopušta mogućnost da uređaj šalje ili prima podatke i u sleep režimu (stand-by). Izbor slave sinhronog moda omogućen je resetovanjem bita CSRC registra TXSTA<7>.

Rad predajnika u sinhronom master i slave modu je identičan, izuzev u slučaju sleep režima. Ako su npr. dva bajta upisana u TXREG i potom izvršena SLEEP instrukcija tada će:

- prvi upisani bajt biti transferovan u pomerački TSR registar i poslat prijemniku,
- drugi bajt ostaje u registru TXREG,
- zastavica prekida TXIF neće biti setovana,
- kada se pošalje poslednji bit prvog bajta iz pomeračkog TSR registra drugi bajt će biti transferovan iz registra TXREG u TSR registar a zastavica prekida TXIF setovana,
- ako je setovan mask bit TXIE prekid će probuditi MCU iz sleep režima i ako je pri tome i bit za globalno omogućenje/onemogućenje prekida GIE setovan program grana na adresu prekidnog vektora (0x04).

Kada je reč o sinhronom slave prijemu, nema razlike u odnosu na master mod izuzev u sleep režimu. Kontrolni bit SREN registra RCSTA<5> nema uticaja na rad USART modula u sinhronom slave modu. Ako je prijem omogućen setovanjem CREN bita kontrolnog i statusnog registra RCSTA<4> pre izvršenja SLEEP instrukcije, tada jedan bajt može biti primljen u toku sleep režima MCU. Po prijemu poslednjeg bita podatka u pomerački registar prijemnika RSR sadržaj će biti transferovan u dvostruko baferovani registar RCREG. Ako je pri tome setovan bit za maskiranje prekida RCIE, generisani prekidni signal će probuditi MCU iz sleep stanja. Program će granati na adresu prekidnog vektora jedino u slučaju ako je i GIE bit setovan. U suprotnom, program nastavlja sa izvršenjem prve sledeće instrukcije posle SLEEP instrukcije.

6.9. SINHRONI SERIJSKI PORT U MASTER MODU (MSSP)

Ovaj integrisani periferijski modul mikrokontrolera PIC16F877 predstavlja sinhroni serijski interfejs, pogodan za komunikaciju sa spoljašnjim uređajima kao što su: serijske EEPROM memorije, pomerački registri, displej drajveri, A/D i D/A konvertori, digitalni senzori, memorijske SD kartice i slično. MSSP modul može raditi u jednom od dva režima:

- SPI (Serial Peripheral Interface - Motorola)
- I^2C (Inter Integrated Circuit - Philips)

6.9.1. SPI sinhroni serijski interfejs

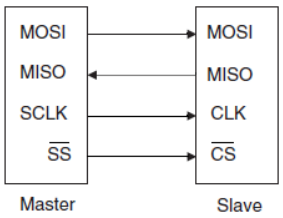
SPI (*Serial Peripheral Interface*) predstavlja trožični/četvorožični sinhroni serijski bus standard (Motorola) koji radi u tkzv. full duplex režimu. Uređaji na SPI magistrali rade u odnosu master/slave, gde master uređaj inicira transfer podataka, selektuje jedan slave uređaj i generiše taktni signal za sve slave uređaje prisutne na magistrali. SPI magistrala može raditi sa jednim master i jednim ili više slave uređaja. Selektovani slave uređaj generiše odziv kao znak prisutnosti na magistrali i potom razmenjuje podatke sa master uređajem sinhrono na svakom taktnom intervalu. Ovaj jednostavan komunikacioni interfejs se često naziva i četvorožičnim komunikacionim interfejsom budući da magistrala sadrži četiri linije. Gledano sa strane master uređaja, linije SPI magistrale su:

- MOSI (master out slave in) ili SDO (serial data out) izlazna linija podataka
- MISO (master in slave out) ili SDI (serial data in) ulazna linija podataka
- SCLK (serial clock) izlazna linija za taktovanje
- SS (slave select) ili CS (chip select) izlazna linija za izbor slave uređaja

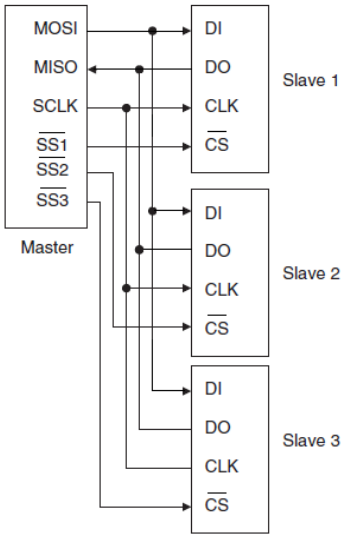
Svaki slave uređaj poseduje iste linije SPI komunikacionog interfejsa kao i master ali je njihov smer suprotan u odnosu na smer linija interfejsa master uređaja. Na slici 6.9.1. je prikazan osnovni način povezivanja jednog master i jednog slave uređaja SPI magistralom a na slici 6.9.2. SPI magistrala sa jednim master i više slave uređaja. Kao što se sa slike 6.9.1. može videti, master uređaj šalje serijski podatak na liniji MOSI a prima na liniji MISO. Pre početka transfera slave uređaj mora biti selektovan uz pomoć dodatne linije SS.

Sa slike 6.9.2. se može videti da se svaki slave uređaj selektuje od strane mastera individualno. Npr., ruka robota može imati MCU jedinice za kontrolu svakog zgloba koje komuniciraju sa master procesorom. Premda svi slave uređaji prisutni na magistrali primaju taktni signal jednovremeno, odaziva se samo jedan selektovani uređaj. Ne selektovani slave uređaji postavljaju svoje izlazne linije podataka DO u stanje visoke impedanse čime se izbegava bilo kakva interferencija sa selektovanim slave uređajem na magistrali. Ne selektovani slave uređaji prelaze u tkzv. režim 'osluškivanja' na magistrali čekajući selekciju. Prenos podataka se npr. vrši od slave uređaja ka masteru i od mastera ka slave uređaju istovremeno i sinhrono sa taktnim impulsima koje generiše master uređaj. Na početku komunikacije master uređaj spušta odgovarajuću liniju za selekciju slave uređaja na niski naponski nivo, čime selektuje jedan od više prisutnih slave uređaja. Potom master započinje sa generisanjem taktnih impulsa u toku čijeg trajanja se vrši full duplex sinhroni prenos podataka. Kada se završi sa prenosom podataka master uređaj stopira taktovanje slave uređaja. Glavne prednosti SPI serijskog komunikacionog interfejsa su:

- jednostavan komunikacioni protokol,
- full duplex komunikacija,
- veoma jednostavan hardver interfejsa.



Slika 6.9.1. *SPI magistrala sa jednim master i jednim slave uređajem.*



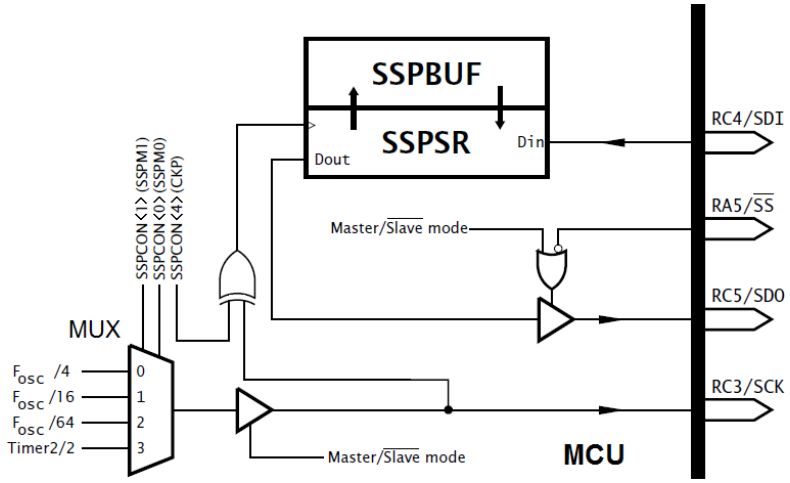
Slika 6.9.2. *SPI magistrala sa jednim master i više slave uređaja.*

Glavni nedostaci SPI interfejsa su:

- najmanje tri i po jedna selekciona linija za svaki slave uređaj, što predstavlja relativno veliki broj žičanih veza za realizaciju protokola,
- nema hardverske kontrole toka (*flow control*),
- nema generisanja signala potvrde od strane slave uređaja.

Treba pomenuti, takođe, da ne postoje SPI standardi regulisani od strane međunarodnog komiteta, tako da danas postoji više SPI implementacija. Na primer, u nekim SPI

implementacijama MOSI i MISO linije podataka su kombinovane u jednu tako da je broj I/O linija redukovan na tri. Neke druge implementacije imaju liniju za selekciju uređaja SS koja je aktivna na visokom naponskom nivou a ne na niskom ili dve taktne linije i slično.



Slika 6.9.3. *Principijelna blok šema MSSP porta MCU PIC16F877.*

Na slici 6.9.3. prikazana je principijelna blok šema sinhronog serijskog porta MCU. Srce SSP porta predstavlja registar specijalne namene SSPBUF. Bajt upisan u ovaj registar automatski se transferuje u pomerački registar SSPSR porta i potom sinhrono šalje na izlaznu liniju RC5/SDO. U isto vreme, osam bitova podatka pomera se sa ulazne linije RC4/SDI puneći pomerački registar porta SSPRS. Kada se ova transakcija kompletira, novi bajt se automatski transferuje u SSPBUF registar odakle se može pročitati. Ovaj transfer se signalizira setovanjem bita BF (*Buffer Full*) statusnog registra porta SSPSTAT<0> i setovanjem SSPIF zastavice prekida u registru prekida PIR1<3>. Bit BF se automatski resetuje čitanjem registra SSPBUF dok se zastavica prekida SSPIF mora programski resetovati ukoliko se koristi prekid.

Ugrađeni četvorokanalni multipleksor MUX, vidi sliku 6.9.3., omogućava programski izbor frekvencije taktovanja pomeračkog registra SSP porta SSPSR internim taktnim signalima na četiri različita načina. Naime, bitovima za kontrolu moda kontrolnog registra SSP porta SSPCON<1:0> SSPM1 i SSPM0, koji predstavljaju adresne ulaze multipleksora, bira se jedan od četiri internih generatora frekvencije taktovanja pomeračkog registra. Tri od četiri raspoloživa izvora taktne frekvencije generišu takt dobijen deljenjem frekvencije glavnog oscilatora CPU jedinice. Npr., ako je frekvencija rezonanse kristalnog oscilatora CPU jedinice 20MHz, mogući izbor taktnih frekvencija SSP porta je 5MHz, 1.25MHz ili 312.5KHz (200ns, 800ns ili 3.2μs). Četvrti izvor taktnog signala je tajmer 2 modul. Frekvencija taktovanja koju generiše ovaj modul jednaka je polovini frekvencije signala koji generiše tajmer 2 modul na bazi poređenja njegovih TMR2 i PR2 osmobitnih registara. Ova opcija se uglavnom koristi za veoma male brzine pomeranja podatka.

Polaritet taktnog signala SCK, takođe se može birati programski setovanjem ili resetovanjem bita CKP kontrolnog registra porta SSPCON<4>. Ako je npr. CKP bit resetovan, podatak postaje validan (odmeravanje) na opadajućoj ivici taktnog signala.

Četiri najniža bita kontrolnog registra SSP porta SSPCON<3:0> određuju radni režim porta, tabela 6.9.1. Dve od šest kombinacija konfigurišu SSP port za radni režim SPI slave. Za razliku od situacije u kojoj uređaj u SPI master modu generiše taktni sinhronizacioni signal, uređaj u SPI slave modu je taktovan spoljašnjim taktom, uobičajeno generisanim od strane udaljenog master uređaja. Slave uređaj čija je kontrolna linija SS na visokom naponskom nivou je ne selektovani uređaj koji se nalazi u tkzv režimu 'osluškivanja'.

SSPM<3:0> bitovi za izbor moda SSP porta	
0000	SPI master mod sa frekvencijom taktovanja $f_{OSC}/4$
0001	SPI master mod sa frekvencijom taktovanja $f_{OSC}/16$
0010	SPI master mod sa frekvencijom taktovanja $f_{OSC}/64$
0011	SPI master mod sa frekvencijom taktovanja *tajmer2/2*
0100	SPI slave mod, SS kontrolna linija omogućena
0101	SPI slave mod, SS kontrolna linija onemogućena

Tabela 6.9.1. *Radni režimi SPI serijskog komunikacionog interfejsa*.

Bez obzira na protokol SSP port se omogućava setovanjem bita SSPEN (*SSP ENable*) kontrolnog registra porta SSPCON<5>. U suprotnom, SSP port je onemogućen a linije porta mogu biti korišćene kao digitalne I/O linije opšte namene. Kada je SSP port omogućen pažnju treba obratiti na smer linija porta koji se bira programiranjem direkcionog TRISC registra.

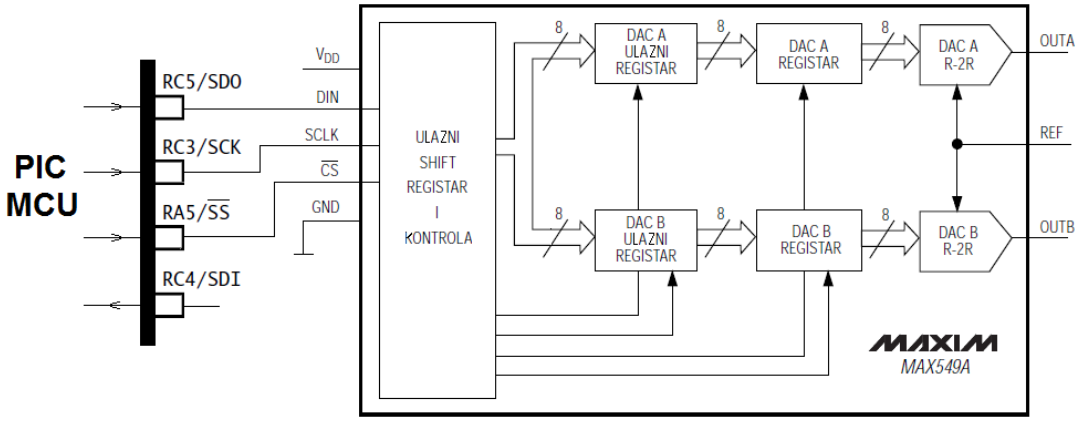
MSB bit kontrolnog registra SSP porta WCOL (*Write COLlision*) se automatski setuje ako program pokuša sa upisom u SSPBUF registar porta a da prenos prethodno upisanog bajta nije kompletiran. Za resetovanje ovog upozoravajućeg mehanizma bit WCOL mora biti resetovan programski.

SPI protokol može biti realizovan i programski ali su glavne prednosti hardverske implementacije povećana operativna brzina i jednostavan upravljački kod. Kada je brzina od esencijalnog značaja treba dodatno koristiti i ugrađeni prekidni mehanizam SSP porta. Prekidni signal porta može biti iskorišćen za buđenje PIC MCU iz sleep režima.

SPI transakcije mogu biti kodirane na C jeziku korišćenjem specifičnih ugrađenih funkcija za rad sa SPI serijskim komunikacionim interfejsom. Na primer, CCS C kompajler koristi sledeće konstrukcije za rad sa SSP portom u SPI modu:

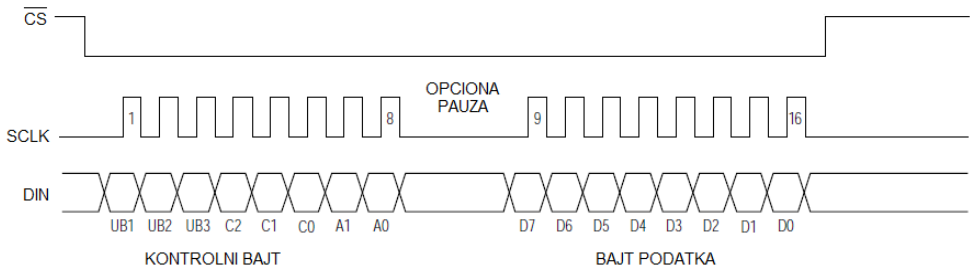
```
spi_write(DATA_OUT);    //f-ja slanja bajta od strane SPI uređaja.
DATA_IN=spi_read();     //f-ja vraća vrednost pročitanu od strane SPI.
setup_spi(spi_master|spi_l_to_h|spi_clk_div_4); //konfig. SSP kao SPI master . . .
spi_data_is_in();       //f-ja vraća ne nultu vrednost ako je pod. primljen preko SPI
RES=spi_xfer(TRANS);  //f-ja prenosa pod. ka i čitanja pod. od jednog SPI uređaja.
```

Ilustracije radi, razmatrajmo primer povezivanja PIC MCU SPI master komunikacionog interfejsa sa SPI slave interfejsom integrisanog osmobitnog dualnog D/A konvertora MAX549A. Na slici 6.9.4. je prikazana principijelna blok šema osmobitnog D/A konvertora MAX549A povezanog sa SPI interfejsom PIC MCU.



Slika 6.9.4. *Blok šema integrisanog dualnog D/A konvertora MAX549A sa SPI slave interfejsom i način povezivanja sa SPI master interfejsom PIC MCU.*

Dvostruko baferovani digitalni podaci prenose se komandom kontrolne logike iz ulaznih prihvatnih registara u izlazne stacionarne registre D/A konvertora i potom konvertuju uz pomoć R/2R lestvičaste otporne mreže u analogni ekvivalent. Kontrolna digitalna logika konvertora opremljena je SPI kompatibilnim, nestandardnim komunikacionim slave interfejsom koji radi isključivo u režimu upisa podataka. Osmobitna kontrolna reč digitalne logike konvertora omogućava različite kombinacije za izbor kanala, upisa podatka u ulazni prihvatni registar, transfer u izlazne stacionarne registre konvertora i slično. Na slici 6.9.5. je prikazan tajming dijagram serijskog SPI interfejsa konvertora MAX549A. Kao što se sa slike može videti, upisu bajta podatka prethodi upis kontrolnog bajta (komande) za programiranje interfejsa. U katalogu proizvođača integrisanog D/A konvertora MAX549A dat je skup kontrolnih reči za programiranje interfejsa.



Slika 6.9.5. *Tajming dijagram SPI interfejsa MAX549A D/A konvertora.*

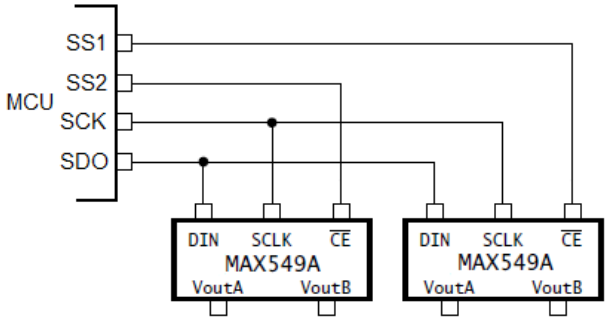
Posle upisa kontrolnog bajta sledi upis podatka za konverziju koji u zavisnosti od kontrolne reči može bit upisan u ulazni prihvatni registar jednog ili drugog kanala, transferovan iz ulaznog u izlazni stacionarni registar i slično. U intervalu između dva upisa moguća je opciona pauza u odsustvu taktnih impulsa.

U tabeli 6.9.2. dat je C kod funkcije za programiranje serijskog interfejsa D/A konvertora MAX549A. Funkcija ima tri argumenta, selekciona linija, podatak A i podatak B, i ne vraća vrednost. Zadatak funkcije je upis digitalnih podataka u oba izlazna stacionarna registra odgovarajućeg D/A integrisanog kola kako bi se na analognim izlazima dobili odgovarajući analogni ekvivalenti.

```
//Funkcija slanja kontrolnog bajta i bajta podatka registrima dualnog D/A
//konvertora MAX549A.

void MAX549A(uint8 CS, uint8 data_A, uint8 data_B)
{
CS=0;               //Selekcija čipa
spi_write(0x01);    //Kontrolni bajt. Punjenje DAC A ulaznog registra. Sadržaji
                    //DAC izlaznih stacionarnih registara nepromenjeni.
spi_write(data_A);  //Upis digitalnog podatka u DAC A ulazni reg.
CS=1;               //Deselekcija čipa

CS=0;               //Selekcija čipa
spi_write(0x0A);    //Kontrolni bajt. Punjenje DAC B ulaznog registra i ažuriranje
                    //sadržaja oba DAC izlazna stacionarna registra.
spi_write(data_B);  //Upis digitalnog podatka u DAC B ulazni registar i prenos
                    //sadržaja iz oba ulazna u oba izlazna DAC registra.
CS=1;               //Deselekcija čipa
}
```

Tabela 6.9.2. C *funkcija za upis podataka u registre DAC A i DAC B D/A konvertora MAX549A.*

Povezivanje dva D/A konvertorska kola MAX549A na SPI magistralu MCU prikazano je na slici 6.9.6. Budući da su selekcione linije SS1 i SS2 odvojene, funkcija data u tabeli 6.9.2. čiji jedan argument je adresa selekcione linije može biti iskorišćena za programiranje oba čipa posredstvom SPI magistrale.



Slika 6.9.6. *Povezivanje dva dualna MAX549A D/A konvertora na SPI magistralu MCU.*

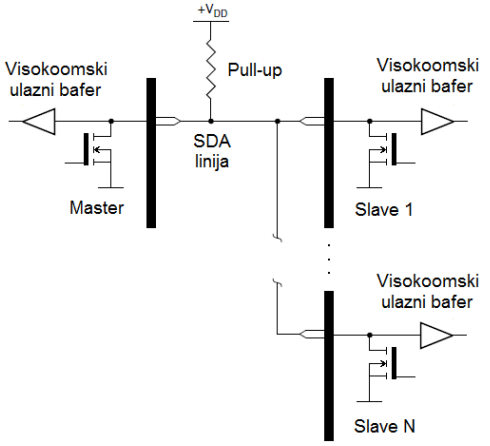
6.9.2. IIC sinhroni serijski interfejs

Premda je SPI protokol relativno brz on ipak za realizaciju zahteva, kao što je napred rečeno, najmanje tri linije i po jednu selekcionu za svaki slave uređaj. Dodavanje novog uređaja na SPI magistralu povećava troškove proširenja jer iziskuje određene hardverske modifikacije dizajna. IIC (*Inter Integrated Circuit*) protokol, razvijen od strane kompanije Philips ranih osamdesetih godina, uvodi novi komunikacioni koncept dvožičnog bidirekcionog prenosa adrese i podatka u jednom serijskom toku redukujući broj linija na dve.

IIC protokol je relativno kompleksan a njegova puna specifikacija dostupna je na sajtu kompanije Philips. Magistralu čine dve linije od kojih je jedna linija taktnog signala za sinhronizaciju transfera SCL a druga linija podataka SDA. Taktna linija SCL je bidirekciona, što dopušta mogućnost da više nego jedan master uređaj preuzme kontrolu nad magistralom u različito vreme. Originalna IIC specifikacija definiše gornju granicu frekvencije taktovanja od 100KHz, ili 100Kbit/s, ali je specifikacija proširena 1993 godine bržim modom sa graničnom frekvencijom od 400KHz ili 400Kbit/s, što je trenutno standard. 1998 godine IIC specifikacija je proširena vrlo brzim modom sa graničnom frekvencijom od 3.4Mbit/s. Bidirekciona linija podataka SDA dopušta tok podataka u jednom ili drugom smeru, odnosno, od master ka slave uređaju i suprotno. Šta više, bidirekcionalnost dopušta mogućnost prijemniku da signalizira svoj status, vraćajući informaciju predajniku na kraju svakog prenetog bajta.

Kada nema podataka za prenos obe linije trebaju biti na viskom naponskom nivou, što predstavlja neaktivno stanje. Master uređaj koji želi da preuzme kontrolu nad neaktivnom magistralom mora spustiti SDA liniju na niski naponski nivo za vreme visokog napona na taktnoj SCL liniji, što je poznato pod nazivom START uslov. Da bi master uređaj, koji generiše startni uslov, mogao da spusti SDA liniju na niski naponski nivo potrebno je da svi ostali uređaji prikačeni na ovu liniju budu u stanju visoke impedanse. Linija SDA se u takvim okolnostima može podići na visoki naponski nivo uz pomoć eksternog pull-up otpornika prikačenog za ovu liniju. Prema tome, za implementaciju SDA linije a takođe i taktne SCL linije izlazi uređaja moraju biti sa otvorenim kolektorom (*open-collector*) ili otvorenim drejnom (*open-drain*). To znači da uređaji moraju generisati logičku nulu da bi liniju spustili na niski naponski nivo ili izlaze

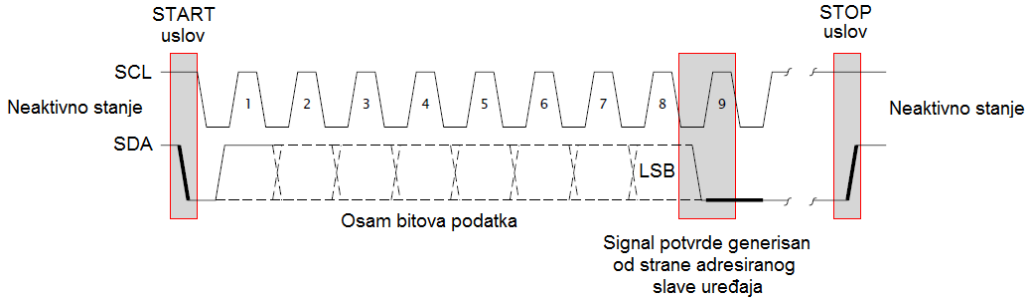
uvesti u stanje visoke impedanse da bi preko spoljašnjeg pull-up otpornika liniju podigli na visoki naponski nivo. Na slici 6.9.7. prikazan je opisani način povezivanja master uređaja sa više slave uređaja na dvožičnoj magistrali.



Slika 6.9.7. *Povezivanje IIC uređaja na IIC magistralu.*

Svaku transakciju na SDA liniji inicira master uređaj generisanjem START uslova, posle čega šalje slave uređajima osam bitova informacije sinhronizovanih taktnim impulsima koje takođe generiše on. Tokom ovog perioda, bilo koja promena bita mora biti izvršena za vreme dok je taktna linija SCL na niskom naponskom nivou. Serijsko pomeranje informacije ka slave uređajima vrši se na rastućoj ivici taktnih impulsa. Osmobitna informacija poslata slave uređajima posle generisanja START uslova predstavlja adresni ili kontrolni bajt.

Posle transfera osmog bita kontrolnog bajta, u devetom taktnom intervalu adresirani slave uređaj zauzima SDA liniju magistrale spuštajući je na niski naponski nivo. Time signalizira master uređaju svoje prisustvo na magistrali. Da bi adresirani slave uređaj zauzeo SDA liniju magistrale master uređaj je mora otpustiti postavljajući svoj izlaz u stanje visoke impedanse. Budući da je linija povezana na napon napajanja preko pull-up otpornika, adresirani slave uređaj uključuje izlazni tranzistor sa otvorenim drejnom, vidi sliku 6.9.7., i time spušta SDA liniju na niski naponski nivo. Provodni tranzistor ima malu otpornost kanala u odnosu na otpornost pull-up otpornika pa formirani razdelnik napona na SDA liniji proizvodi niski napon koji odgovara stanju logičke nule. Signal koji generiše slave uređaj poznat je pod nazivom signal potvrde (*acknowledge*) kojim slave potvrđuje svoje prisustvo na magistrali. Ako ovaj signal u devetom taktnom intervalu izostane, master uređaj prekida transfer i uobičajeno posle određene pauze pokušava sa novom transakcijom. Opisana situacija ilustrovana je vremenskim dijagramima na slici 6.9.8.

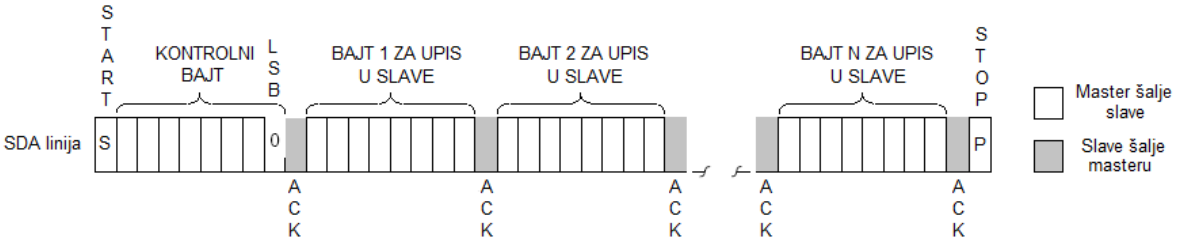


Slika 6.9.8. *Transfer podataka na IIC magistrali.*

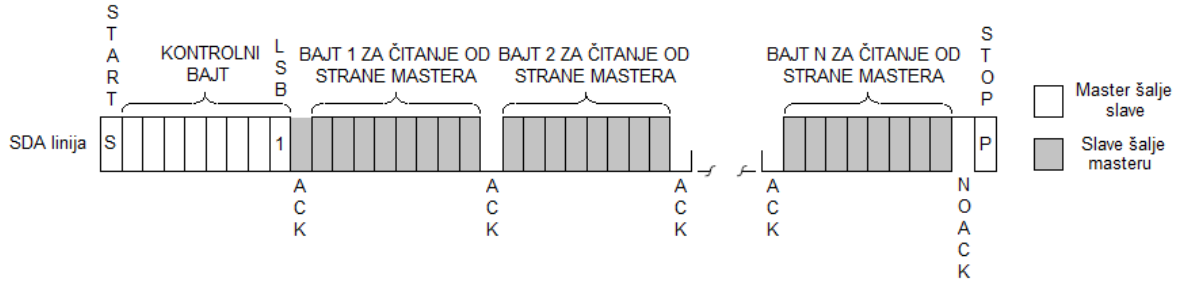
Posle prenosa kontrolnog bajta i odziva slave uređaja može se dogoditi prenos više bajtova u jednom ili drugom smeru pre nego što master uređaj generiše tkzv. STOP uslov kojim završava transakciju. STOP uslov master generiše postavljajući SDA liniju na visoki naponski nivo za vreme dok je taktni signal na visokom naponu. Sledeća transakcija mora započeti ponovnim generisanjem START uslova od strane master uređaja i završiti STOP uslovom. Slave uređaj razlikuje specijalne START i STOP situacije koje na magistrali generiše master.

Očigledno je da je svaka transakcija uokvirena parom uslova START i STOP a tok podataka između može biti od mastera ka slave uređaju (master upisuje) ili obratno (master čita). Tok podataka određen je stanjem LSB bita kontrolnog bajta koji master šalje slave uređaju posle generisanog START uslova. Naime, poslednji poslati bit kontrolnog osmobitnog podatka je LSB bit, slika 6.9.8. Operacija čitanja slave uređaja od strane mastera kodirana je sa LSB=1 a operacija upisa u slave sa LSB=0 (R/$\overline{W}$). Preostalih sedam viših bitova kontrolnog bajta može poslužiti delom u svrhe adresiranja i drugim delom u svrhe kodovanja vrste IIC uređaja.

U slučaju izbora operacije upisa, posle svakog poslatog bajta od strane master uređaja, koji sleduju iza slanja kontrolnog bajta, slave uređaj signalizira prijem signalom potvrde (*acknowledge*) spuštajući magistralu na niski napon, slika 6.9.9. U slučaju odsustva signala potvrde posle bilo kog poslatog bajta master prekida transakciju koju može ponoviti posle određene pauze. Na slici 6.9.9. neosenčene površine odgovaraju aktivnosti master uređaja (master šalje slave uređaju) a osenčene aktivnosti slave uređaja (slave šalje master uređaju) na liniji prenosa podataka SDA. Radi jednostavnosti izostavljen je talasni oblik taktnih impulsa na SCL liniji budući da se može pretpostaviti sinhronizacija svakog prenetog bita taktnim impulsima. Broj bajtova za upis u slave uređaj zavisi od konkretnog uređaja. Npr. ako je slave uređaj osmobitna serijska EEPROM memorija tada prvi bajt za upis predstavlja viši deo 16-bitne adrese memorijske lokacije, drugi bajt predstavlja niži deo adrese dok je treći bajt osmobitni podatak koji se upisuje u selektovanu lokaciju. Posle preneta tri bajta i odziva slave memorije master emituje STOP uslov čime uspešno završava transakciju.



Slika 6.9.9. *Protokol slanja paketa od N bajta slave uređaju.*



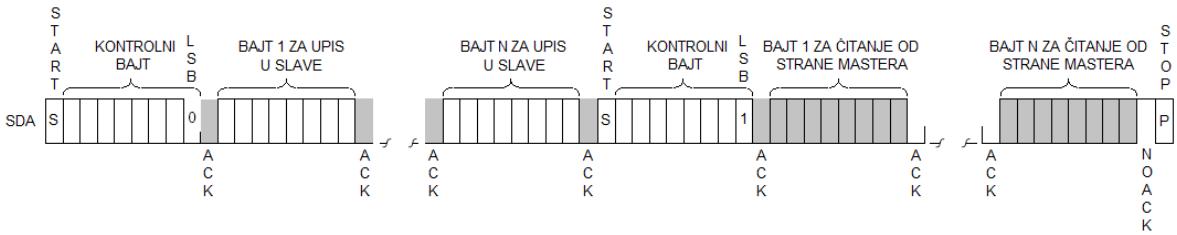
Slika 6.9.10. *Protokol prijema paketa od N bajta od strane master uređaja.*

Kao što je rečeno, operacija čitanja slave uređaja se bira slanjem kontrolnog bajta slave uređaju sa bitom LSB=1 posle generisanog start uslova, slika 6.9.10. Odmah nakon zauzimanja magistrale i slanja signala potvrde (*acknowledge*), slave uređaj nastavlja sa slanjem bajta na

SDA liniji sinhrono sa taktnim impulsima koje generiše master. Posle svakog primljenog bajta, u devetom taktnom intervalu, slave uređaj otpušta SDA liniju koju zauzima master spuštajući je na niski naponski nivo kao signal potvrde primljenog bajta (*acknowledge*). Posle primljenog poslednjeg N-tog bajta, u devetom taktnom intervalu, master neće zauzeti magistralu a zbog postojanja pull-up otpornika napon na SDA liniji postaje visok. Takvu situaciju slave uređaj tumači kao odsustvo signala potvrde od strane mastera (*noacknowledge)* zbog čega otpušta SDA liniju dopuštajući masteru da generiše STOP uslov i time završi transakciju.

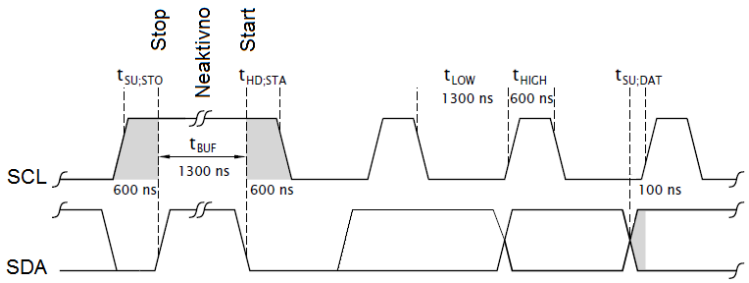
Broj bajtova koje slave uređaj šalje masteru zavisi od konkretnog slave uređaja i algoritma kontrole prenosa od strane mastera. Npr. čitanje osmobitne serijske EEPROM memorije može biti okončano posle prenosa jednog bajta ako master ne generiše signal potvrde posle primljenog bajta (*noacknowledge).* Međutim, zahvaljujući mehanizmu pokazivača na adresu memorijske lokacije, koji se automatski inkrementira posle upisanog ili pročitanog bajta, memorija može biti u celosti pročitana sekvencijalno ako master stalno posle svakog primljenog bajta generiše signal potvrde (*acknowledge)* do prijema poslednjeg bajta.

Još jedna karakteristična situacija na IIC magistrali ilustrovana je na slici 6.9.11. Radi se o situaciji u kojoj master započinje i vrši upis u slave uređaj nakon čega menja smer prenosa podataka. Umesto STOP uslova, master bez prekidanja transakcije šalje drugi START uslov (RESTART) i potom odgovarajući kontrolni bajt sa izborom operacije čitanja. Opisana situacija je moguća npr. prilikom čitanja IIC serijske EEPROM memorije sa tačno određene memorijske lokacije direktnim pristupom. Naime, master uređaj upisom dva bajta 16-bitne adrese u slave memorijski uređaj selektuje jednu od 2^{16} memorijskih lokacija sa koje posle RESTART uslova isčitava bajt podatka. Takođe se može vršiti uzastopno čitanje više memorijskih lokacija počev od adrese lokacije selektovane upisom dva adresna bajta.



Slika 6.9.11. *IIC protokol kontinuiranog uzastopnog upisa i čitanja podataka.*

Već je rečeno da sedam viših bitova kontrolnog bajta predstavlja adresu slave uređaja. Esencijalno za IIC protokol predstavlja zahtev da svaka vrsta slave uređaja ima jednu adresu. Ova se adresa dodeljuje od strane proizvođača IIC interfejsa i fabrički se programira. Da bi dva ili više uređaja iste vrste mogla deliti istu magistralu dopušta se mogućnost da dva do četiri bita ove adrese budu postavljena lokalno od strane dizajnera, obično povezivanjem adresnih pinova uređaja na odgovarajuće logičke nivoe. Posle START uslova svi slave uređaji prisutni na magistrali ispituju i porede prvih sedam primljenih bitova sa njihovom personalnom adresom. Ako ne postoji slaganje, ostatak transakcije se ignoriše do sledećeg START uslova.



Slika 6.9.12. *Minimalni tajming za brzu IIC magistralu (400KHz).*

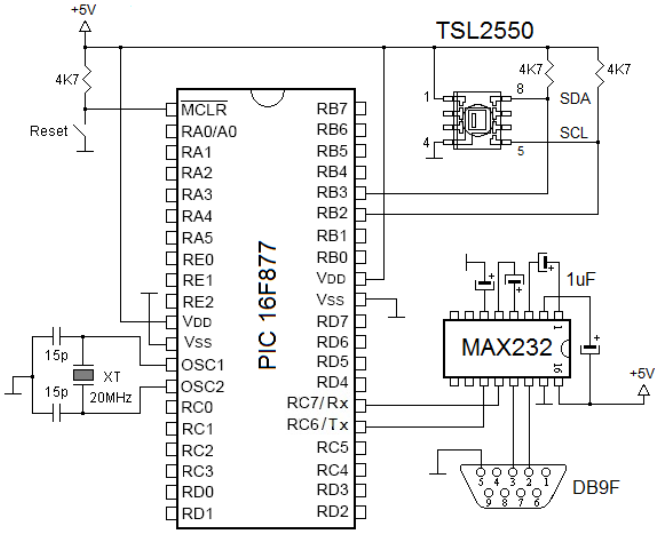
Slika 6.9.12. prikazuje minimalna potrebna karakteristična vremena na SCL i SDA linijama standardne, brze IIC magistrale (f_{CLKMAX}=400KHz). Npr., posle generisane opadajuće ivice impulsa na SDA liniji SCL linija mora ostati na visokom naponu za vreme $t_{HD;STA}$≥0.6μs da bi se ispravno generisao START uslov. Slično prethodnom, generisanje jednog STOP uslova zahteva da linija takta SCL bude najmanje $t_{SU;STO}$=0.6μs na visokom naponu pre nego se generiše rastuća ivica impulsa na SDA liniji. Minimalno potrebno vreme između jednog STOP i sledećeg START uslova, odnosno, minimalno vreme neaktivnosti magistrale je prema slici 6.9.12. t_{BUF}=1.3μs. Minimalna trajanja niskog i visokog napona na SCL liniji iznose 1.3μs i 0.6μs, respektivno. Promena bita podatka vrši se za vreme niskog napona na taktnoj SCL liniji i mora biti kompletirana za vreme ne manje od $t_{SU;DAT}$=100ns pre pojave rastuće ivice takta.

Na slici 6.9.12. nisu prikazana maksimalna vremena porasta i opadanja ivica impulsa na SCL i SDA liniji čija trajanja ne smeju biti veća od 300ns pri maksimalnoj parazitskoj kapacitivnosti linija magistrale od 400pF. Za poštovanje ovih tranzicionih restrikcija potrebno je usvojiti vrednosti pull-up otpornika na obe linije ne veće od 1.8KΩ. Međutim, za kraće magistralne linije i manji broj slave uređaja na magistrali, vrednosti pull-up otpornika mogu biti i do deset puta veće kako bi se smanjila disipacija za vreme trajanja niskog napona na linijama SDA i SCL.

Ugrađene biblioteke CCS C kompajlera za rad sa integrisanom IIC periferijom PIC MCU dopuštaju mogućnost korišćenja IIC hardverskog interfejsa i mogućnost programske realizacije IIC protokola na bilo kojim I/O linijama portova. Ako se koristi programski IIC protokol pažnju treba posvetiti činjenici da najveći broj linija I/O portova nije tipa otvoreni drejn, izuzev linije porta A RA4. Stanje logičke jedinice na izlaznoj liniji nije stanje visoke impedanse kako to zahteva IIC protokol. Međutim, moguće je liniju porta rekonfigurisati kao ulaznu (velika ulazna otpornost) i tako simulirati njeno odspajanje sa magistrale. Stanje logičke jedinice formira se zahvaljujući pull-up otporniku na liniji IIC magistrale.

CCS C kompajler koristi sledeće konstrukcije za rad sa SSP portom u IIC modu:

```
i2c_start();        //f-ja generiše START uslov u master modu
i2c_stop();         //f-ja generiše STOP uslov u master modu
data = i2c_read(x); //f-ja za čitanje bajta iz slave uređaja (x=1 ACK, x=0 NOACK)
i2c_write(data);    //f-ja za upis bajta u slave uređaj (f-ja vraća ACK bit)
i2c_poll();         //f-ja vraća vrednost 1 ako je hardver primio bajt u baferu
#use I2C(master, sda=PIN_C3, scl=PIN_C4, fast=400000) //direktiva za konfig. IIC porta
```



Slika 6.9.13. *Povezivanje TSL2550 digitalnog senzora osvetljaja IIC magistralom sa PIC MCU.*

Slika 6.9.13. ilustruje način povezivanja modernog digitalnog senzora ambijentalnog osvetljaja TSL2550 (TAOS) sa ugrađenim serijskim IIC interfejsom i PIC16F877 MCU. Kolo je opremljeno MAX232 linijskim drajverom za komunikaciju sa PC računarom i slanje rezultata monitoru serijskog porta. Digitalni senzor TSL2550, proizvodnje TAOS, je senzor ambijentalnog osvetljaja koji kombinuje dve foto diode, jednu osetljivu na vidljivu i infracrvenu svetlost (kanal 0) i drugu primarno osetljivu na infracrvenu (kanal 1), jedan A/D konvertor i serijski komunikacioni IIC interfejs na jedinstvenom supstratu. Digitalni izlaz kanala 1 koristi se za kompenzaciju efekta infracrvene komponente ambijentalne svetlosti kanala 0. Digitalni izlazi oba kanala koriste se za dobijanje vrednosti osvetljaja u Lux-ima koja aproksimira odziv ljudskog oka. Senzor je prvenstveno namenjen za kontrolu pozadinskog svetla displeja laptop računara, PDA uređaja, GPS prijemnika i slično.

Senzor može raditi u dva režima, standardnom i proširenom. Vreme konverzije za svaki kanal u standardnom modu iznosi 400ms, odnosno, 800ms za oba kanala. Prošireni mod dopušta senzoru da radi u uslovima većih osvetljaja sa pet puta kraćim vremenom konverzije u odnosu na standardni mod.

TSL2550 sadrži 8-bitni komandni registar u koji se može upisivati ili se on može čitati preko IIC interfejsa. Ovaj registar kontroliše celokupan rad integrisanog senzora. Postoje i dva samočitljiva registra koja sadrže poslednje konvertovane vrednosti svakog od dva ADC kanala. Sedmobitna adresa senzora kao slave uređaja hardverski je ožičena interno u postupku fabrikacije senzora i ima binarnu vrednost 0111001. Budući da je adresa nepromenljiva, na magistralu se može povezati samo jedan senzor TSL2550.

Sadržaji komandnog registra senzora i odgovarajuće operacije prikazane su u tabeli 6.9.3.

Komanda	Operacija
0x00	Uspostavljanje stand-by režima rada sa smanjenom potrošnjom senzora
0x03	Buđenje senzora/čitanje sadržaja komandnog registra
0x1D	Komanda za izbor proširenog radnog režima
0x18	Komanda za resetovanje ili povratak na standardni radni režim
0x43	Čitanje ADC kanala_0
0x83	Čitanje ADC kanala_1

Tabela 6.9.3. *Sadržaji komandnog registra senzora i odgovarajuće operacije.*

Po priključenju na napon napajanja, senzor osvetljaja TSL2550 automatski prelazi u stand-by režim rada sa vrednošću komandnog registra 0x00. Komanda 0x03 ima dve namene: koristi se za buđenje uređaja iz stand-by režima ili nakon upisa u komandni registar može biti iskorišćena za proveru ispravnosti komunikacije. Naime, vrednost vraćena u toku ciklusa čitanja komandnog registra treba biti 0x03 ako je komunikacija ispravna.

Senzor sadrži dva osmobitna ADC registra podataka (kanal 0 i kanal 1), svaki podeljen na dve tetrade CHORD (viši polubajt) i STEP (niži polubajt). Ove dve tetrade se koriste za određivanje logaritamske ADC vrednosti u skladu sa uputstvom i specifikacijom proizvođača. MSB bit svakog ADC registra je bit validnosti koji ukazuje na to da je A/D konvertor upisao podatak u registar odgovarajućeg kanala, uzimajući u obzir konačno vreme konverzije konvertora. Više detalja o značenju i upotrebi tetrada i načinu izračunavanja osvetljaja može se naći u uputstvu proizvođača senzora TSL2550.

```
#include <16F877.h>
#device ADC=10
#fuses HS,NOWDT,PUT,NOPROTECT,NOLVP,NODEBUG
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#define TSL_SDA  PIN_B3       //RB3
```

```
#define TSL_SCL  PIN_B2       //RB2

#use I2C(master, sda=TSL_SDA, scl=TSL_SCL, slow=100000)

long const step_value[8] = {1,2,4,8,16,32,64,128};   // Lookup tabela za step_value
long const chord_value[8] = {0, 16, 49, 115, 247, 511, 1039, 2095};  // Lookup tabela
                                                                     // za chor_value
#include <stdio.h>
#include <math.h>

void main()  {

int1 wake=0;
int8 cmd=0x1D, Chord_0, Chord_1, Step_0, Step_1, channel_0, channel_1;
long ADC_0, ADC_1;
float Light_Level;

      output_float(TSL_SDA);     // Inicijalizacija IIC magistrale
      output_float(TSL_SCL);
      delay_us(20);

// IIC Rutina za budjenje TSL2550

  if (!wake) {
      i2c_start();
      i2c_write(0x72);    // Adresa TSL slave uredjaja i operacija upisa (kontr. bajt)
      i2c_write(0x03);    // Upis komande 0x03 u komandni reg. za budjenje TSL2550
      i2c_stop();
      wake=1;

// IIC Rutina za izbor moda

      i2c_start();
      i2c_write(0x72);   // Adresa TSL slave uredjaja i operacija upisa (kontr. bajt)
      i2c_write(cmd);    // Upis komande 0x1D za izbor proširenog radnog moda
      i2c_stop();
      }

// IIC Rutine za citanje registara ADC konvertora

RPTR:
      i2c_start();
      i2c_write(0x72);   // Adresa TSL slave uredjaja i operacija upisa u komand. reg.
      i2c_write(0x43);   // U komandni reg. upisana komanda čitanja kanala_0
      i2c_start();       // Restart uslov
      i2c_write(0x73);   // Adresa TSL slave uredjaja i operacija čitanja kanala_0
      channel_0=i2c_read(0);        // NOACK
      i2c_stop();


      i2c_start();
      i2c_write(0x72);  // Adresa TSL slave uredjaja i operacija upisa u komand. reg.
      i2c_write(0x83);  // U komandni reg. upisana komanda čitanja kanala_1
      i2c_start();      // Restart uslov
      i2c_write(0x73);  // Adresa TSL slave uredjaja i operacija čitanja kanala_1
```

```
      channel_1=i2c_read(0);        // NOACK
      i2c_stop();

////////////////////////////////// OBRADA ////////////////////////////////////
//Ispitivanjem MSB bita kanala_0 i kanala_1 utvrđuje se kraj konverzije (MSB=1)

    if( bit_test(channel_0,7) && bit_test(channel_1,7) ) {

         Step_0=channel_0 & 0x0f; Step_1=channel_1 & 0x0f; Chord_0=(channel_0>>4 &
         0x07); Chord_1=(channel_1>>4 & 0x07);
         ADC_0=chord_value[Chord_0] + step_value[Chord_0] * Step_0;
         ADC_1=chord_value[Chord_1] + step_value[Chord_1] * Step_1;
      if ((ADC_0 | ADC_1)==0)
         Light_Level = 0;
         else
         Light_Level=5.0*((float)ADC_0*0.46)*exp(-3.13*((float)ADC_1/(float)ADC_0));
         }
         else
             goto RPTR;    //Ako je MSB=0 ponovi čitanje kanala_0 i kanala_1
    printf("Nivo osvetljaja je: %4.1f Lux\n", Light_Level);  //Slanje PC računaru
}
```

Tabela 6.9.4. *Primer korišćenja ugrađenih funkcija CCS C kompajlera za rad sa IIC serijskim komunikacionim interfejsom.*

U tabeli 6.9.4. je dat opis C programa za kontrolu rada kola sa slike 6.9.13. Program pokazuje da se može ostvariti jednostavna interakcija PIC MCU i IIC slave uređaja, kao što je digitalni senzor osvetljaja TSL2550, korišćenjem specijalizovanih ugrađenih funkcija CCS C kompajlera za rad sa SSP portom u IIC modu.

I pored dobrih osobina kao što su mali broj žičanih veza i jednostavan protokol, IIC komunikacioni interfejs ima ključnu manu – relativno malu brzinu prenosa podataka što za zahtevnije aplikacije može biti ograničavajući faktor.

ZAHVALNOST

Rad na knjizi "Integrisani Računarski Sistemi - PICmicro®" podržan je od strane Ministarstva za Nauku i Tehnologiju Republike Srbije u okviru projekta broj III47016.

LITERATURA

[1] D. Živković, M. Popović, *Impulsna i Digitalna Elektronika*, Nauka, Elektrotehnički fakultet, Beograd, 1993.

[2] Vujo Drndarević, *Personalni Računari u Sistemima Merenja i Upravljanja*, Akademska misao, Elektrotehnički fakultet, Saobraćajni fakultet, Beograd, 2003.

[3] Sid Katzen, *The Quintessential PIC Microcontroller*, Springer Verlag, NewYork, 2000.

[4] Dogan Ibrahim, *Advanced PIC Microcontroller Projects in C*, Elsevier Ltd. USA, 2008.

[5] Stuart R. Ball, *Analog Interfacing to Embedded Microprocessors*, Newnes, USA, 2001.

[6] Ken Arnold, *Embedded Controller Hardware Design*, LLH Technology Publishing, USA, 2000.

[7] Julio Sanchez & Maria P. Canton, *Microcontroller Programming-The Microchip PIC*, CRC Press, USA, 2007.

[8] L. D. Jasio, T. Wilmshurst, D. Ibrahim, J. Morton, M. P. Bates, J. Smith, D. W. Smith, C. Hellebuyck, *PIC Microcontrollers*, Elsevier Inc. USA, 2008.

[9] Myke Predko, *Programming and Customizing PICmicro Microcontrollers*, McGraw-Hill, USA, 2000.

[10] D.W. Smith, *PIC in Practice.* Newnes, USA, Oxford, 2002.