Univerzitet u Banjoj Luci
Elektrotehnički fakultet

REALIZACIJA MIKROPROCESORSKOG DALJINSKI UPRAVLJANOG POZICIONERA ANTENE

DIPLOMSKI RAD

Kandidat: **Čedomir Zeljković**
Mentor: **prof. dr Slobodan Vukosavić**

Banja Luka, juni 2003. godine

SADRŽAJ

1 UVOD

1.1. ZAŠTO SVE OVO?

Malo je koji diplomski rad donio neko "epohalno otkriće". Ni ovaj rad, naravno, ne predstavlja ništa spektakularno i dosad neviđeno. Međutim, postoji nekoliko činjenica koje daju težinu i vrijednost ovome projektu.

Prvenstveno, zadatak konstrukcije pozicionera antene je sveobuhvatan. To je jedan zaokružen projekat od idejnog rješenja do praktične realizacije. Od konstruktora se očekuje da inženjerski razmišlja i racionalno organizuje čitavi proces realizacije. Podrazumijeva se sposobnost generisanja i debagovanja programskog kôda, praktično programiranje mikrokontrolera, realizacija štampanih ploča,… Znanje potrebno za projektovanje hardvera i pripadajućeg mu soſtvera leži u mnoštvu oblasti koje se izučavaju na elektrotehničkom fakultetu. Tu prevashodno treba izdvojiti procesne računare, digitalnu i energetsku elektroniku, elektromotorne pogone, a neki dijelovi projekta dodiruju oblasti automatike i prenosa signala. Tako, jedan ovakav zadatak predstavlja svojevrsan izazov kreatoru da pokaže svoju spremnost da se i u budućnosti može uhvatiti u koštac sa novim praktičnim problemima.

1.2. PROJEKTNI ZADATAK

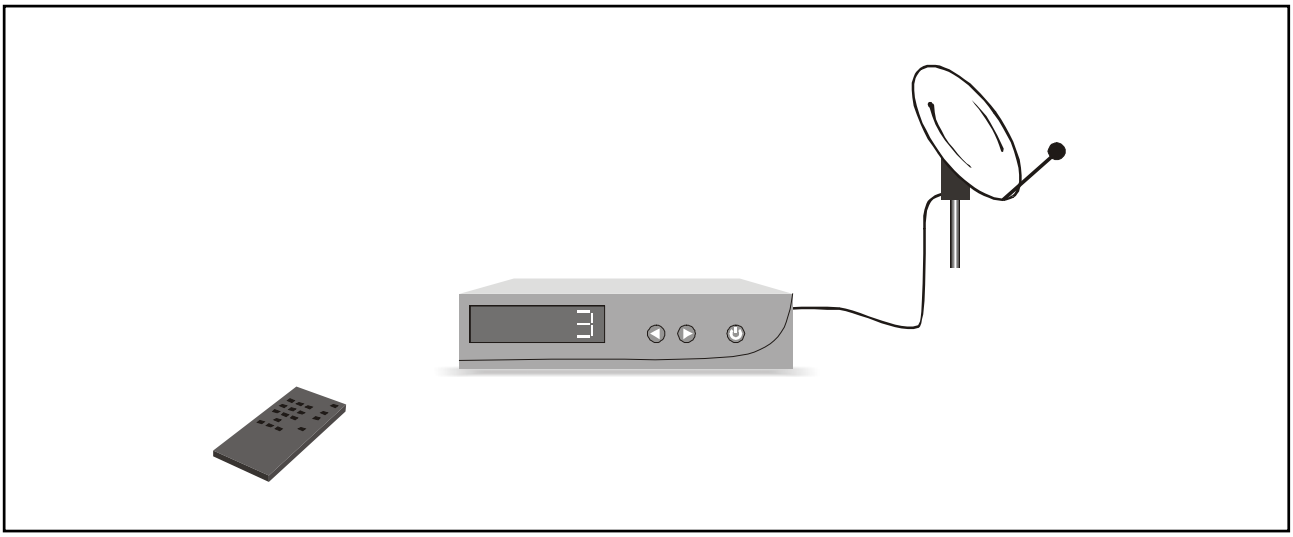
1.2.1. Struktura sistema za pozicioniranje antene

Antenski pozicioner je zamišljen kao jednostavan kućanski uređaj. S tom svrhom potrebno je što više pojednostaviti njegovu konstrukciju i smanjiti cijenu, a da ne dođe do gubitka osnovne funkcionalnosti.

Centralna komponenta čitavog sistema je mikroprocesorski sklop u zasebnom kućištu. Napajan je mrežnim naponom 220V / 50Hz i zato naravno opskrbljen odgovarajućim AC/DC pretvaračem. Potrebno je da taj uređaj posjeduje elemente korisničkog interfejsa, dakle tastere za upravljačke akcije i displej za vizuelnu kontrolu novonastalih efekata. Takođe, potrebno je implementirati infracrveno daljinsko upravljanje što bi funkcionalnost uređaja podiglo na još jedan nivo više.

Drugi dio ovog pozicionog sistema je mehanička konstrukcija za rotiranje antene, opremljena pogonskim koračnim motorom i davačem krajnjeg položaja. Očekuje se, naravno, da će ovaj dio sistema biti montiran u eksterijeru, pa se predviđa kablovska veza od motora i davača ka centralnoj jedinici. Ne planira se direktna veza između motora i osovine antene, nego treba upotrijebiti mehanički reduktor realizovan na principu pužnog prenosa. To će donijeti dva vrlo značajna poboljšanja u čitavom projektu. Prvo poboljšanje se manifestuje kroz povećanje preciznosti pozicioniranja, jer reduktor posjeduje značajan stepen prenosa. Drugi dobitak je smanjenje spoljnih uticaja na ugaoni položaj antene, zbog kočenja puža, pa nije potrebno da motor bude stalno uključen. Druga strana medalje je povećanje vremena potrebnog da se dostigne željena pozicija antene. Planirana rezolucija pozicionera antene je 0,1°, a sve brže od 1/10 obrtaja antene u sekundi bila bi za ovaj uređaj prihvatljiva brzina pozicioniranja.

Slika 1.1: KOMPONENTE ANTENSKOG POZICIONERA



Osnovna namjena opisanog mikroprocesorskog pozicionera je rotiranje zemaljske ili kućne satelitske antene. To daje opravdanje da se predviđa samo jedan stepen slobode i to promjena azimutalnog ugla antene. U osnovnoj realizaciji nije potrebno djelovati na elevacioni ugao, jer se zemaljske antene montiraju paralelno tlu, a svi komercijalni sateliti su u jednoj liniji, pa je dovoljno jednom fiksirati odgovarajuću elevaciju i ne minjati je. Tako bi jedna od mogućih ekstenzija ovog uređaja bila realizacija višedimenzionalnog pozicionera za neku napredniju namjenu, eventualno sa pozicioniranjem spregnutim sa jačinom primanog signala.

1.2.2. Upravljanje pozicionerom

Dva su osnovna stanja uređaja: mod uključenog uređaja i mod isključenja (*stand by mode*).

Po dovođenju napajanja uređaj je isključen. Na displeju je prikazana samo jedna crtica i uređaj je neosjetljiv na korisničke komande. Jedino je aktivna komanda *Power* koja dovodi do uključenja. Pretvarač za upravljanje motorom je onemogućen i motor je sigurno isključen.

Kada se uređaj uključi, postaje sposoban da prima i ostale komande za upravljanje. Planirano je deset kanala numerisanih od 0 do 9 za memorisanje deset različitih pozicija antene. Inicijalno je aktivan onaj kanal na kojem je uređaj posljednji put isključen, a to je omogućeno memorisanjem u interni EEPROM mikrokontrolera. U toku normalnog funkcionisanja uređaja broj trenutno aktivnog kanala prikazan je na cifri jedinica četvorocifrenog displeja. Izbor novog aktivnog kanala vrši se upotrebom odgovarajućeg numeričkog tastera na daljinskom upravljaču. Motor tada kreće u cilju zauzimanja memorisane pozicije tog kanala. Prebacivanje na neposredno sljedeći kanal ostvaruje se tasterom *channel up*, dok se neposredno prethodni dostiže tasterom *channel down*. Predviđeno je da ovi tasteri postoje kako na daljinskom upravljaču, tako i na prednjem panelu same centralne jedinice sistema. Poseban mod rada pozicionera je mod postavljanja nove željene pozicije (*set mode)*, gdje se pomoću numeričkih tastera na daljinskom upravljaču unosi vrijednost nove ugaone pozicije. Iz moda podešavanja se izlazi bilo memorisanjem nove pozicije, bilo odustajanjem kada bi se antena trebala vratiti u prethodno memorisani položaj.

Novo djelovanje na taster *Power* dovešće do isključenja i čuvanja posljednjeg statusa uređaja u internoj EEPROM memoriji.

2 KONCEPCIJA SISTEMA

2.1. HARDVERSKI BLOK DIJAGRAM

Zahvaljujući činjenici da *Microchip*-ov mikrokontroler PIC16F877 predstavlja kvalitetnu integraciju centralne procesne jedinice (CPU), memorijskih registara i periferija, ne zahtijeva se složen eksterni hardver da bi se kompletirao mikroprocesorski sistem. Na blok-dijagramu sa slike 2.1 može se propratiti struktura sistema.

Od ulaznih jedinica uočavaju se prijemnik infracrvenih komandi, jednostavna tastatura sa tri osnovna tastera, te davač krajnjeg položaja antene.

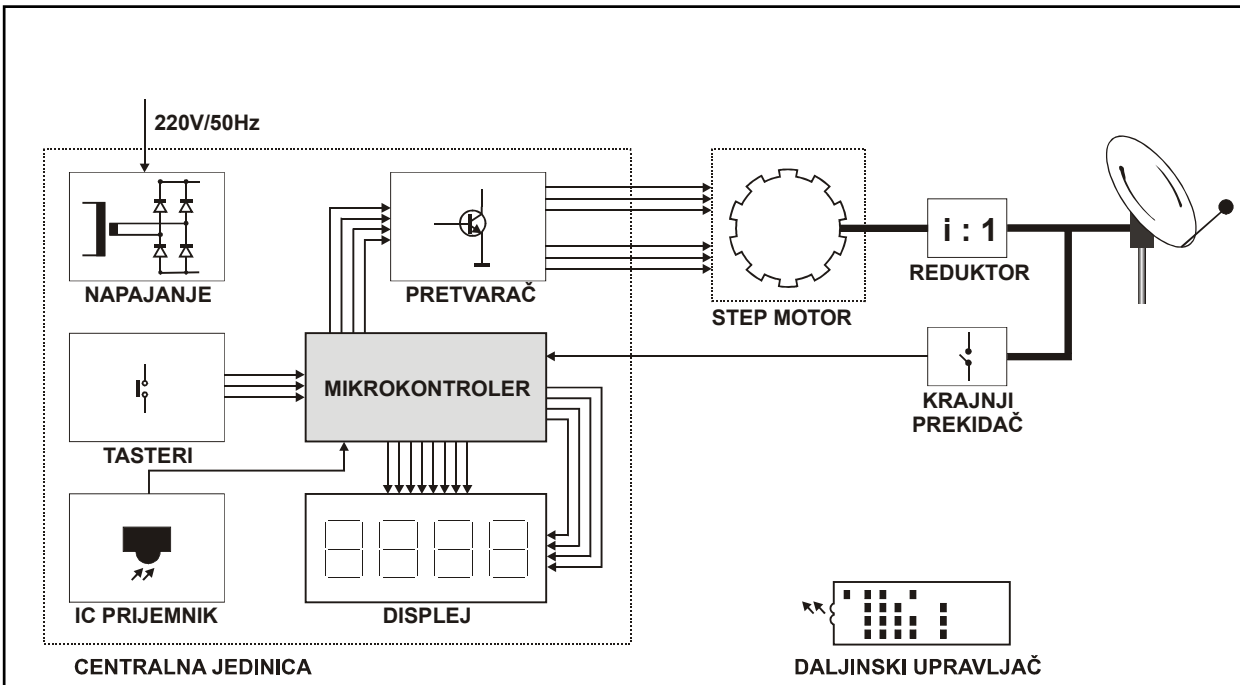
U izlazne jedinice koje opslužuje mikrokontroler spadaju tranzistorski pretvarač za upravljanje koračnim motorom i četvorocifreni sedmosegmentni LED displej, koji služi za vizuelnu kontrolu uređaja za vrijeme korisničkih akcija.

Naravno, blok koji je neophodan za funkcionisanje čitavog sistema je blok napajanja, gdje se od mrežnog napona 220V/50Hz dobijaju filtrirani jednosmjerni naponi +5V i +12V. Napon +5V, koji služi za napajanje mikrokontrolera, dodatno je stabilizovan.

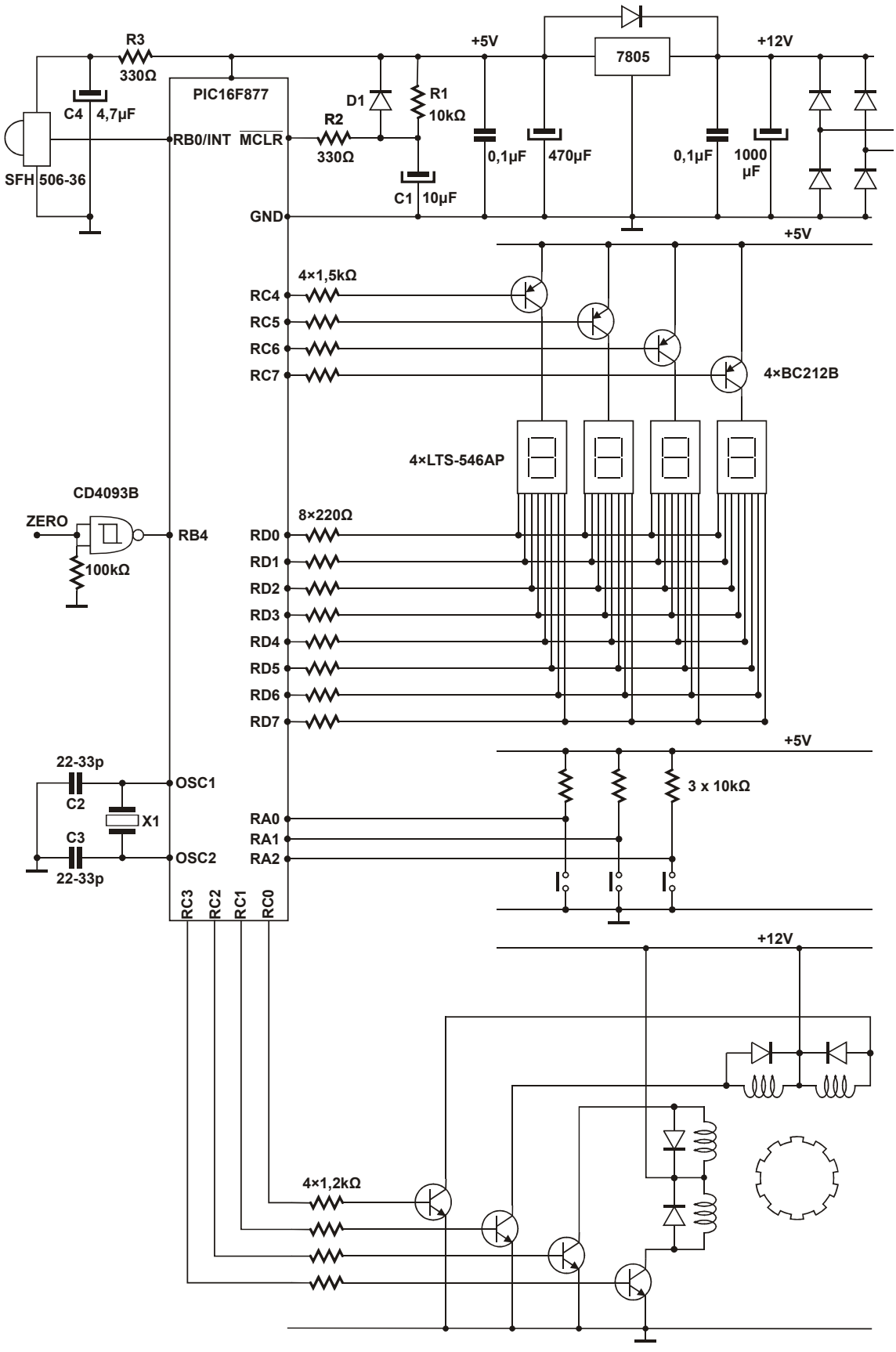
Na slici 2.2 data je detaljna šema osnovne ploče uređaja. Dominantni dio je mikrokontroler PIC16F877 sa nekolicinom eksternih komponenti neophodnih za njegovo ispravno funkcionisanje. Tu spadaju reset-kolo (D1, R1, R2, C1), oscilatorske komponente (X1, C2, C3) i napajanje. Integrisana komponenta SFH 506-36 služi kao prijemnik infracrvenih daljinskih komandi, a par R3, C4 predstavlja njeno filter-kolo. Impulsi sa davača krajnje pozicije uvode se preko CMOS Šmitovog trigera CD4093B, a još tri ulazna pina mikrokontrolera okupirana su tasterima *Power*, *Channel Up* i *Channel Down*. Konačno, vidljiva je veza četiri sedmosegmentna LED displeja sa zajedničkom anodom u sklopu sa multipleksiranim osvježavanjem, gdje tranzistori BC212B služe kao displej-drajveri s obzirom na ograničenost izlazne struje porta mikrokontrolera na 25 mA.

Detaljni opis funkcija pojedinih hardverskih komponenti, kao i njihovo sadejstvo sa softverom mikrokontrolera, daje se u nastavku rada u specijalizovanim poglavljima. O modovima rada uređaja i upravljanju govori osmo poglavlje.

Slika 2.1: BLOK DIJAGRAM HARDVERA



Slika 2.2: DETALJNA ŠEMA OSNOVNE PLOČE



2.2. ORGANIZACIJA SOFTVERA

Da bi se ovaj antenski pozicioner mogao svrstati u kategoriju *low cost* uređaja, hardver je dosta jednostavan. Međutim, to daje neophodnost softverske implementacije željenih funkcija. Naravno, takvo rješenje je prihvatljivije kad god je moguće da softver odgovori na sve postavljene zahtjeve. Tada je fleksibilnost velika, a i cijena uređaja rapidno opada sa brojem proizvedenih primjeraka.

Softversko rješenje ovog slučaja nema neku sebi svojstvenu formu, nego je oblikovano u stilu većine programa za *embedded* aplikacije. Tri su glavna dijela programa:

- Inicijalizacija
- Glavna programska petlja
- Prekidne (*interrupt*) rutine

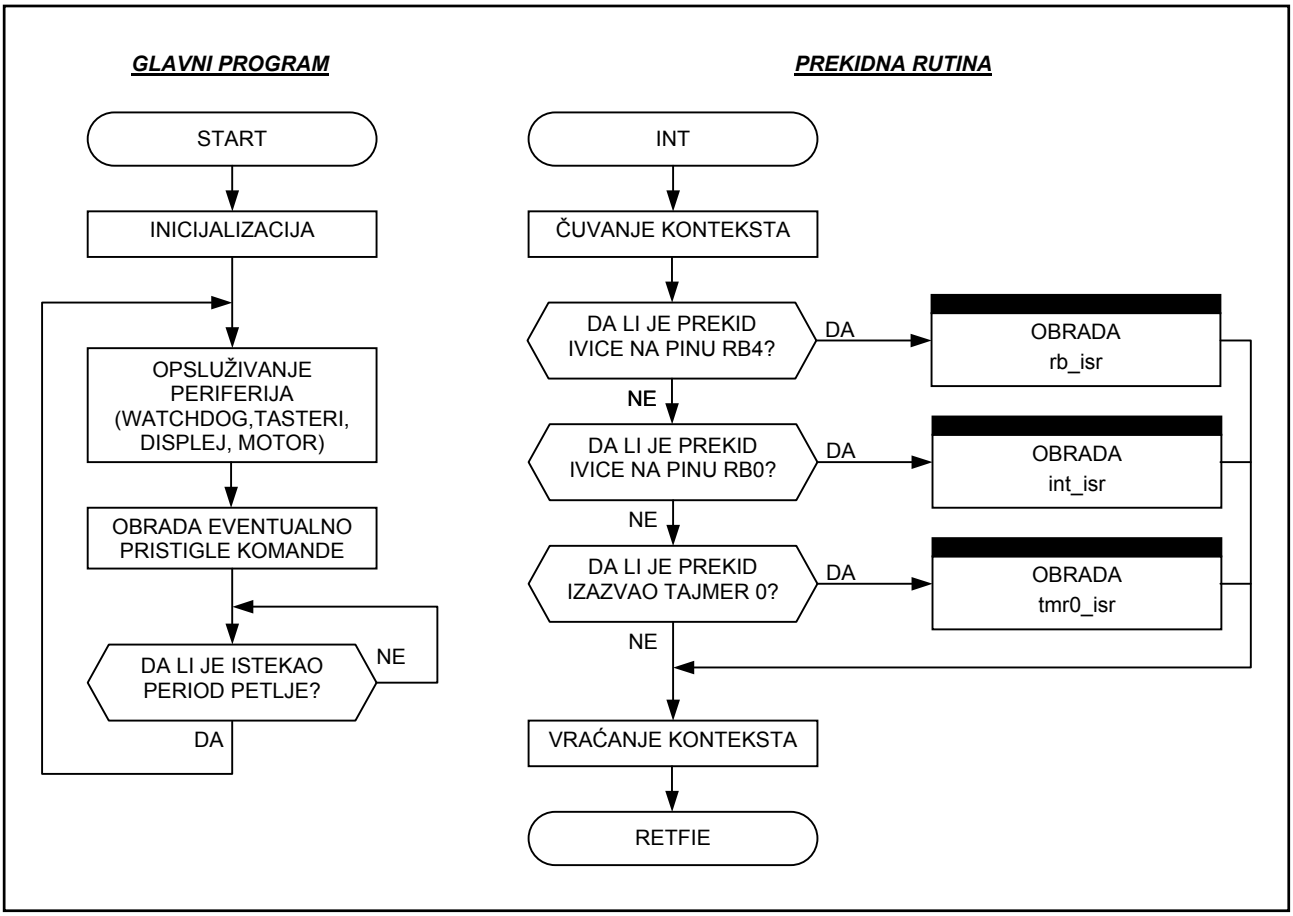
Inicijalizacioni dio programa namijenjen je da pripremi uređaj za ispravan rad. To se obavlja punjenjem nekolicine registara konfiguracionim bajtovima. Sadržaj ostalih registara odgovara po resetu, pa ih nije potrebno dirati. U inicijalno konfigurisanje prvenstveno spadaju određivanje smijera linija na I/O portovima, te podešavanje osnovnih periferija preko registra `OPTION_REG`. Koji će prekidi u startu biti omogućeni određuju vrijednosti upisane u registre `INTCON` i `PIE1`. Na koncu, vrši se inicijalizacija korisničkih registara u RAM-u. Primjer je postavljanje uređaja u *stand by mode* odmah po uključenju i prikaz samo jedne crtice na mjestu posljednje cifre displeja.

Uvid u glavnu programsku petlju može se steći posmatranjem programskog blok-dijagrama sa slike 2.4. Trajanje petlje je 1 ms, a pošto se u svakom prolasku ispiše po jedna cifra četvorocifrenog LED displeja, dobije se da je frekvencija osvježavanja displeja 250 Hz. Takođe, u glavnoj petlji vrši se skeniranje tastera i upravljanje pozicijom aktuatorskog motora. Konačno, tu se vrši i procesiranje komandi koje su eventualno pristigle od daljinskog upravljača.

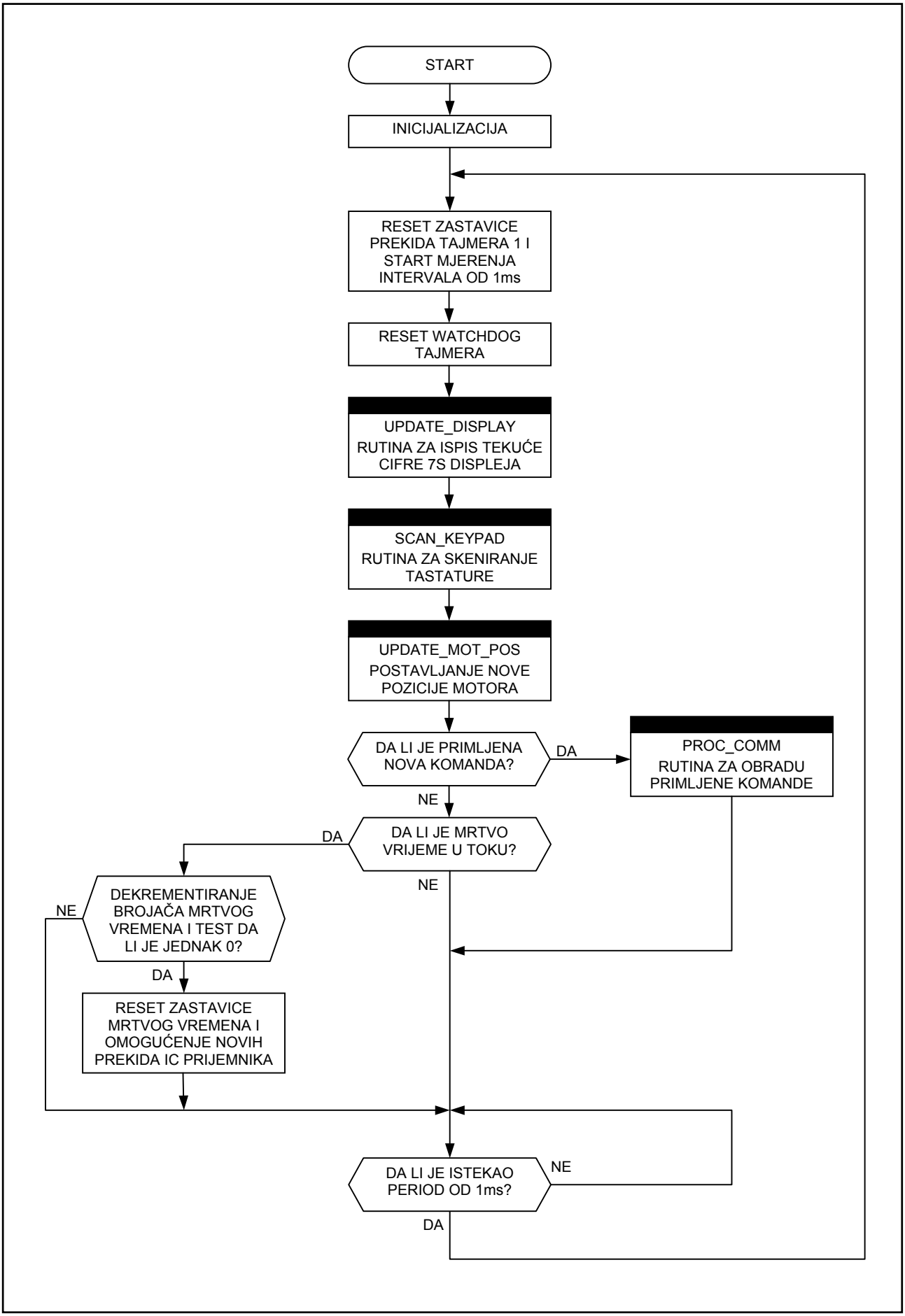
Pošto su primanja impulsa od daljinskog upravljača i davača krajnje pozicije asinhroni događaji, rutine za njihovu obradu pozivaju se interaptom mikrokontrolera. Posebna poglavlja namijenjena su objašnjenju njihovog načina funkcionisanja.

Kao mjera zaštite od slučaja da program "iskoči" iz glavne petlje i "zaluta" u nekom neželjenom pravcu upotrijebljen je *watchdog*. To je brojač sa vlastitim, internim, RC oscilatorom, čiji *overflow* izaziva reset mikrokontrolera. Glavna petlja programa obavezno mora sadržavati instrukciju koja nulira ovaj brojač (`clrwdt`). Zato, dokle god se ispravno vrti glavna petlja neće doći do nasilnog reseta, a ako se za duži period ne izvrši instrukcija `clrwdt`, nastupiće ponovna inicijalizacija mikrokontrolera.

Slika 2.3: STRUKTURA SOFTVERA MIKROKONTROLERA



Slika 2.4: DETALJAN BLOK-DIJAGRAM GLAVNOG PROGRAMA MIKROKONTROLERA



3 CENTAR SVIJETA - MIKROKONTROLER

3.1. OSNOVNE KARAKTERISTIKE PIC16F877

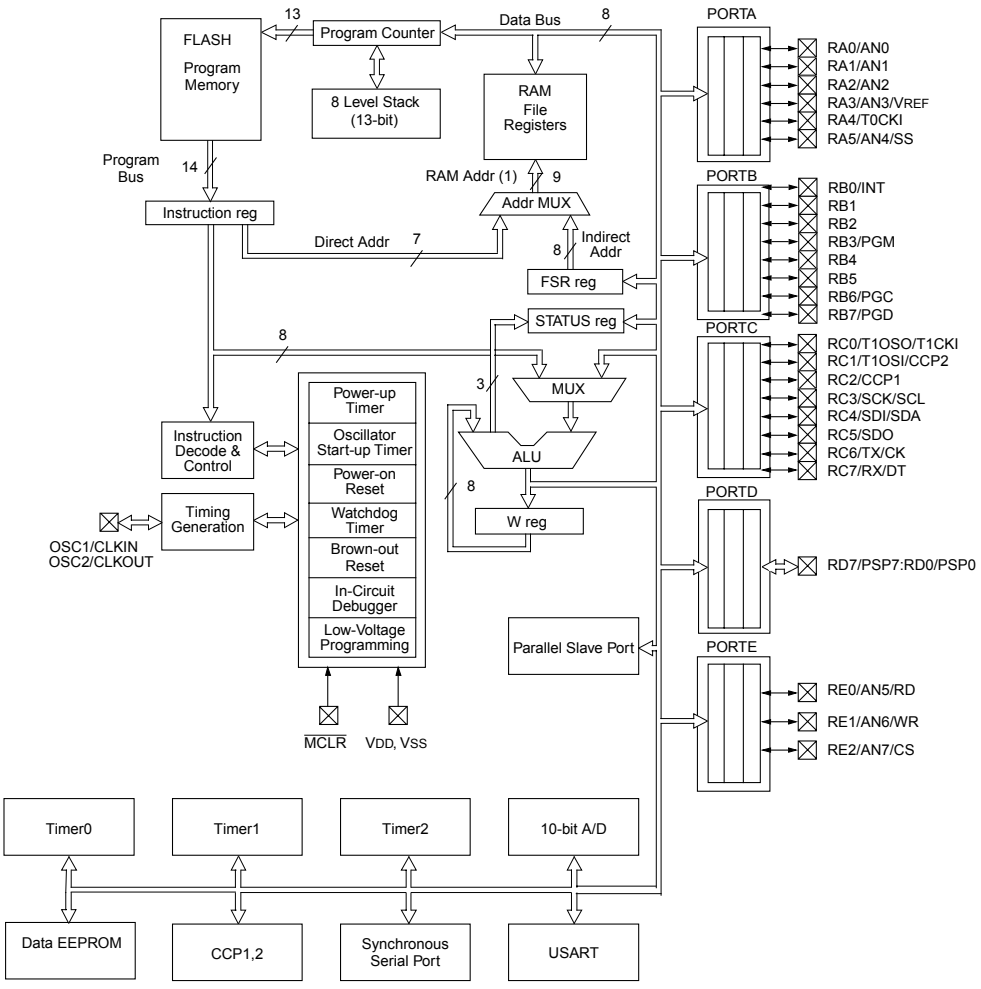
Riječ je o *Microchip*-ovom 8-bitnom CMOS mikrokontroleru zasnovanom na *flash* tehnologiji. To znači da na samom čipu postoji programska memorija koja se može upisivati i brisati električnim putem, što je naprednije od UV brisanja uređaja sa EPROM-om.

RISC arhitektura omogućuje odvojene magistrale 8-bitnih podataka i 14-bitne programske memorije. Tako je moguće da se pribavlja naredna instrukcija dok se izvršava tekuća (*pipelining*). Sve instrukcije traju jednako (osim u slučaju grananja programa) i završe se za četiri ciklusa oscilatora. Dakle, ako je oscilator konfigurisan na maksimalnih 20 MHz, dobije se da ciklus instrukcije iznosi zavidnih 200 ns.

Sa blok dijagrama kojeg daje *Microchip* (slika 3.1) može se ustanoviti da se koncepcija ovog mikrokontrolera ne razlikuje mnogo od koncepcije RISC mikrokontrolera drugih proizvođača prisutnih na tržištu. Uočavaju se standardne komponente:

- Flash programska memorija – 8 kword
- RAM (*File Registers*) – 368 bajtova
- Aritmetičko-logička jedinica (ALU)
- Akumulator (*Working Register*)
- Hardverski stek (*Stack*) sa 8 nivoa
- EEPROM memorija podataka – 256 bajtova
- Razne periferne jedinice (portovi, tajmeri, A/D konvertor, USART,...)

Slika 3.1: ARHITEKTURA MIKROKONTROLERA



PIC16F877 podržava tehniku prekida (*interrupts*). Postoji ukupno 14 izvora prekida, što vanjskih, što unutrašnjih. Svaki prekid nema sopstveni interapt-vektor, nego postoji jedinstvena adresa (0x0004) od koje se nastavlja izvršavati program kada se dogodi bilo koji od njih. Tada je na programeru da prozivanjem zastavica pojedinih prekida (*interrupt flags polling*) ustanovi ko traži prekid i uputi program na izvršavanje odgovarajuće rutine za obradu. Adresa na koju se program treba vratiti po obradi prekida čuva se automatski u hardverskom steku i u programski brojač vraća izvršavanjem instrukcije RETFIE.

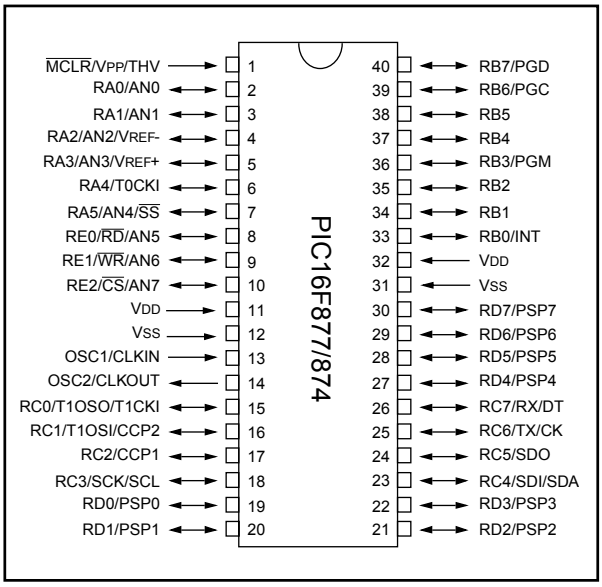
Ono što mikrokontrolere izdvaja od ostalih procesora jeste prisustvo raznih specijalnih kola koja se trebaju izboriti sa potrebama *real-time* aplikacija. Ovaj mikrokontroler posjeduje niz osobina potrebnih da se maksimizuje pouzdanost sistema, minimizuje cijena kroz eliminaciju eksternih komponenti, omoguće režimi niske potrošnje energije, te pruže zaštitu programskog kôda. Upotreba navedenih resursa definiše se u programatorskom softveru upisom odgovarajuće konfiguracione riječi u registar CONFIG. Taj registar je dostupan samo u toku upisa programa u mikrokontroler i drugačije se ne može adresirati. Među ponuđenim karakteristikama stoje mogućnost odabira četiri tipa oscilatora, mogućnost upotrebe *Power-up* (PWRT) i *Oscilator Start-up* (OST) tajmera, te eventualno korišćenje *Power-on* (POR) i/ili *Brown-out* (BOR) reseta. Povećanju pouzdanosti kroz sprečavanje zalaska programa u mrtve petlje pomaže upotreba *Watchdog* tajmera (WDT). Zaštitu kôda od neželjenog iščitavanja pruža opcija *code protection*.

Za uređaje sa baterijskim napajanjem interesantna mogućnost je *Sleep Mode*, režim sa smanjenom potrošnjom energije, kada se program ne izvršava, ali rade neke periferne jedinice koje mogu "probuditi" mikrokontroler. Tada je potrošnja mikokontrolera ispod 1 μA.

Jezgro mikrokontrolera PIC16F877 pakuje se u 40-pinsko DIP pakovanje ili u 44-pinska kućišta QFP i PLCC. Na slici 3.2 data je 40-pinska varijanta pakovanja procesora.

Napajanje se dovodi na pinove V_{DD} i V_{SS}. Nožice OSC1 i OSC2 služe za priključenje oscilatorskih komponenti (RC-kolo ili rezonator), odnosno priključenje eksternog oscilatora kao bolje, ali skuplje varijante. Osim za reset-kolo pin 1 (MCLR/V_{PP}) ima ulogu u toku procesa programiranja mikrokontrolera. Ostalih 33 pina predstavljaju I/O linije. Oni su grupisani u pet portova (PORTA – PORTE) i svaki od njih je individualno konfigurabilan kao izlaz ili kao ulaz. Osim opšte namjene većina pinova ima i specifičnu svrhu koju dobija u slučaju korišćenja nekih specijalnih periferija mikrokontrolera (brojača, serijske komunikacije, A/D konvertora i dr.).

Slika 3.2: PIN-DIJAGRAM



Na kraju ovog uvoda treba napomenuti da su *Microchip* PIC mikrokontroleri dominantni u odnosu na konkurentne relativno visokom strujom koju može propustiti I/O pin (25 mA). Takođe, ova familija mikrokontrolera posjeduje dosta širok opseg napona napajanja koji se proteže od 2,0 V do 5,5 V.

3.2. MEMORIJSKA MAPA MIKROKONTROLERA

Strukturu memorije kod PICmicro™ mikrokontrolera čine tri odvojena bloka:

- Programska memorija
- Memorija podataka
- EEPROM memorija podataka

Odvojeno od navedenih memorijskih blokova egzistira zasebna struktura stek (*stack*), koja se sastoji od 8 13-bitnih registara. Stek pointer se ne može čitati i upisivati. Prilikom izvršenja instrukcije `CALL` ili prilikom poziva prekida mikrokontrolera, adresa sljedeće instrukcije se stavlja na stek. Ponovno vraćanje starog sadržaja programskog brojača izaziva izvršenje instrukcije `RETURN`, `RETFIE` ili `RETLW`. Stek radi na principu cirkularnog bafera, što znači da se u njega može staviti osam različitih adresa, a da pokušaj unošenja devete izaziva brisanje prve i tako redom. Programski se ne može utvrditi da li je došlo do prepunjavanja steka.

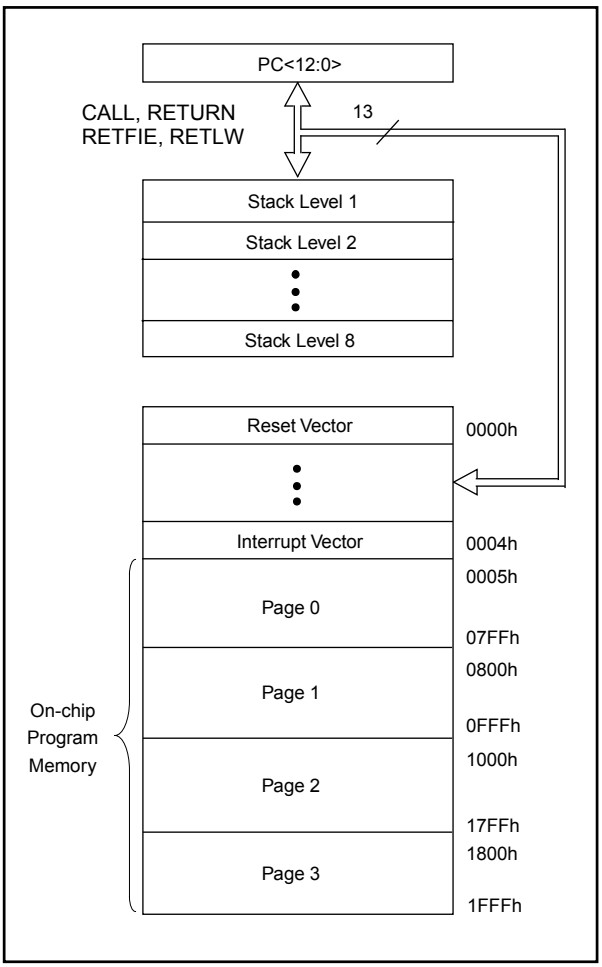
3.2.1. Organizacija programske memorije

PIC16F877 mikrokontroleri imaju 13-bitni programski brojač (PC) koji je u mogućnosti da adresira memorijski prostor od 8k programskih riječi od 14 bita.

Reset vektor je 0x0000 i od njega počinje izvršavanje programa. Interapt vektor je 0x0004.

Mapa programske memorije i stek prikazani su dijagramom na slici 3.3.

Slika 3.3: MAPA PROGRAMSKE MEMORIJE I STEK



3.2.2. Organizacija memorije podataka

Memorija za podatke je izdijeljena u više cjelina – banki (*banks*), a sastoji se od registara opšte namjene (*General Purpose Registers*) i registara specijalne funkcije (*Special Function Registers*). U jednom od specijalnih registara, tzv. STATUS registru postoje dva bita RP1 i RP0 koji služe za odabir željene banke podataka po principu:

RP1	RP0	Bank
0	0	Bank0
0	1	Bank1
1	0	Bank2
1	1	Bank3

Svaka banka može da sadrži do 128 registara (0x7F). Niže lokacije u banci zauzimaju specijalni registri, a ostatak prostora popunjavaju registri opšte namjene implementirani kao statički RAM. Neki specijalni registri koji se često koriste mapirani su u sve banke da bi se omogućio brži pristup i redukcija kôda.

Mapa registara procesora PIC16F877 prikazana je na slici 3.4. Nekoliko specijalnih registara su registri jezgra, usko povezani sa funkcionisanjem CPU. Ostali registri su vezani za periferne module i služe njihovom upravljanju i kontroli statusa.

Slijedi opis nekoliko najvažnijih i najčešće korišćenih registara jezgra.

Programski brojač (PC) određuje adresu instrukcije u programskom flešu koja će sljedeća biti pribavljena. Riječ je o 13-bitnom registru. Simboličko ime nižeg bajta je PCL. To je registar koji se može i upisivati i iščitavati. Težih pet bita programskog brojača smješteni su u izolovani registar PCH kojem se pristupa samo preko leča PCLATH mapiranom u internom RAM-u na adresi 0x0A.

STATUS registar je veoma bitan i zato je predviđeno da se može adresirati iz bilo koje banke. On pokazuje status aritmetičko-logičke jedinice, reset status mikrokontrolera i sadrži bite za selekciju banki internog RAM-a. Od navedenih flegova posebno treba izdvojiti *Zero* bit (Z) koji se postavlja kad je rezultat aritmetičke operacije jednak nuli i bit prenosa/pozajmice *Carry* (C).

Registar **OPTION_REG** se koristi za konfiguraciju preskalera za tajmer 0 ili *Watchdog*, za upravljanje tajmerom 0, selekciju ivice okidanja eksternog interapta, te za omogućenje *Pull-up* otpornika na portu B.

INTCON je registar za manipulisanje sistemom prekida mikrokontrolera. Pored bita za omogućenje svih prekida (GIE) i bita za omogućenje perifernih prekida (PEIE), u ovom registru su interapt-flegovi i biti omogućenja prekida tajmera 0, vanjskog prekida na pinu RB0/INT i prekida porta B na promjenu stanja. Osim ova tri osnovna prekida postoji još 11 periferijskih prekida. Bitovi za njihovo omogućenje nalaze se u registrima PIE1 i PIE2, a korespondentni flegovi, vijesnici interapta, u registrima PIR1 i PIR2. Ovi se flegovi setuju čim se ispuni uslov interapta bez obzira na stanje njihovog bita omogućenja, a po izvršenju servis rutine potrebno ih je softverski resetovati.

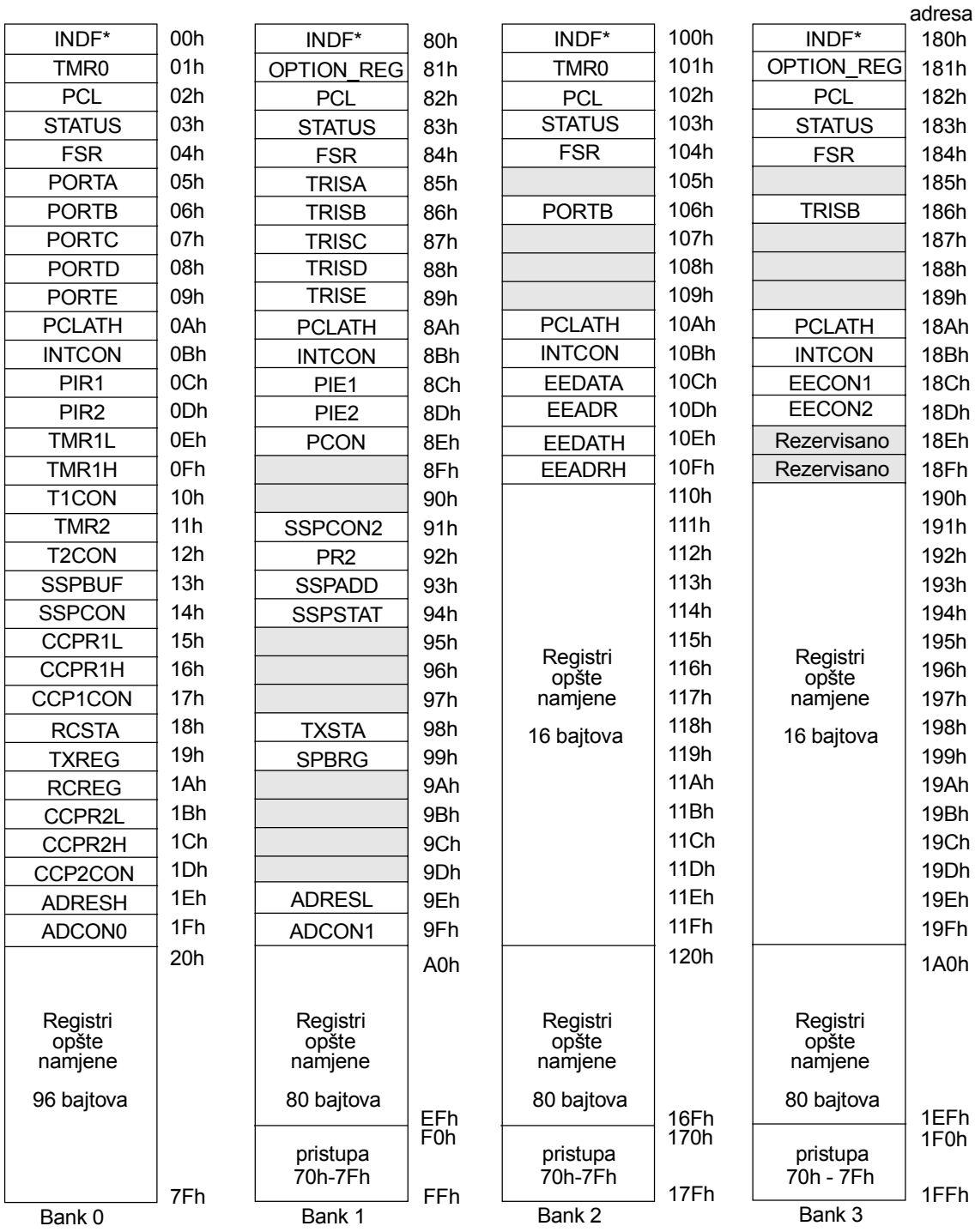
Kada je riječ o registrima jezgra ne treba zaboraviti par **FSR** (*File Select Register*) i **INDF** (*Indirect File*), koji služe za indirektno adresiranje memorije podataka. Bilo koja instrukcija koja se obraća INDF registru ustvari indirektno pristupa onoj lokaciji internog RAM-a čija je adresa trenutno u registru FSR.

Konačno, treba spomenuti registar **PCON** (*Power Control Register*). U PIC16F877 on sadrži samo dva bita. Pomoću bita POR detektuje se razlika između *Power-on* reseta i reseta izazvanih drugim uzrokom. Drugi bit (BOR) služi kao indikacija *Brown-out* stanja (nepropisne naponske prilike u napajanju mikrokontrolera), zbog kojeg se takođe može desiti reset.

3.2.3. Interni EEPROM za podatke

Ova memorija sadrži 256 bajtova. Ako je potrebno neke podatke sačuvati i po ukidanju napajanja mikrokontrolera, treba ih prethodno zapisati u interni EEPROM.

Slika 3.4: REGISTARSKA MAPA MIKROKONTROLERA



Neimplementirani dio RAM-a. Čita se kao '0'.

* Nije fizički registar.

3.3. OPIS PERIFERNIH JEDINICA

3.3.1. I/O portovi

Za vezu mikrokontrolera sa svijetom oko sebe zaduženi su ulazno/izlazni portovi. Ima ih pet i označeni su slovima od A do E. Nejednake su širine, pa tako port E čine samo tri pina, port A šest, a ostala tri porta su osmopinski.

Neki pinovi I/O porta su povezani sa nekom perifernom funkcijom mikrokontrolera. Jedino ako je odgovarajuća periferna jedinica neupotrijebljena, moguće je njen pin koristiti kao I/O liniju opšte namjene. Konfiguracija smijera portova vrši se upisom konfiguracionog bajta u pripadajući TRIS registar po pravilu da nula čini pin izlaznim, a jedinica ulaznim.

Svaki port posjeduje odgovarajući registar (PORTX) preko kojeg se programski pristupa I/O pinovima. Upis u neki od tih registara iniciraće upis u leč tog porta, a njegovo čitanje rezultiraće čitanjem logičkih stanja direktno sa pinova. Sve instrukcije upisivanja su tzv. *read-modify-write* instrukcije. To znači da se pri upisu u port prvo očitaju stanja pinova, izvrši modifikacija, a potom ispravljena vrijednost pohrani u leč porta.

Nema velike razlike u električnoj konstrukciji navedenih pet portova. Port B se od ostalih razlikuje interesantnom opcijom koju nude četiri njegova viša bita. Ukoliko se setuje bit RBIE u registru INTCON, svaka promjena stanja na ovim pinovima generisaće prekid mikrokontrolera.

3.3.2. Tajmeri

Ove se periferije koriste za mjerenje vremena i brojanje eksternih događaja. Ugrađena su tri tajmerska modula (TMR 0,1,2) od kojih svaki ima svoje specifičnosti.

Tajmer0 je jednostavni 8-bitni brojač koji generiše interapt pri prelasku sa 0xFF na 0x00 (*overflow*). Posjeduju ga svi niži PICmicro™ procesori i ovdje je zadržana kompatibilnost s njima. Izvor takta za tajmer0 može biti bilo interni sistemski sat (Fosc/4), bilo vanjski generator takta spojen na pin RA4/T0CKI. Moguće je podesiti da se brojač inkrementira na rastuću ili opadajuću ivicu vanjskog signala.

U sadejstvu sa modulom tajmera 0 može se koristiti programabilni preskaler (djelilac frekvencije) sa odnosima dijeljenja od 1:2 do 1:256. Ukoliko dijeljenje nije potrebno preskaler se dodijeli *watchdog* tajmeru.

Kada je preskaler u upotrebi maksimalna frekvencija eksternog izvora iznosi 50 MHz što je veće od maksimalne frekvencije samog mikrokontrolera.

Tajmer1 je 16-bitni i takođe je osposobljen da radi kao brojač ili mjerač vremena. Ima tri izvora takta: sistemski sat (Fosc/4), vanjski takt ili vanjski kristal. Brojač eksternih događaja može se sinhronizovati sa internim oscilatorom, a postoji i asinhroni način rada koji omogućuje da se brojač inkrementira i u *sleep* režimu. Preskaliranje je upotrebljivo sa vrijednostima dijeljenja 1:1, 1:2, 1:4 i 1:8.

Tajmer2 je 8-bitni tajmer sa programabilnim preskalerom i postskalerom. Može biti tajmer/brojač opšte namjene. Međutim, potreban je CCP modulu prilikom generisanja PWM signala, te modulu za sinhronu serijsku komunikaciju (SSP) kao *Baud Rate* generator. U kooperaciji sa tajmerom 2 koristi se registar PR2 (*Period Register*). Kada se vrijednost brojača izjednači sa vrijednošću upisanom u registar PR2, generiše se odgovarajući interapt.

3.3.3. CCP moduli

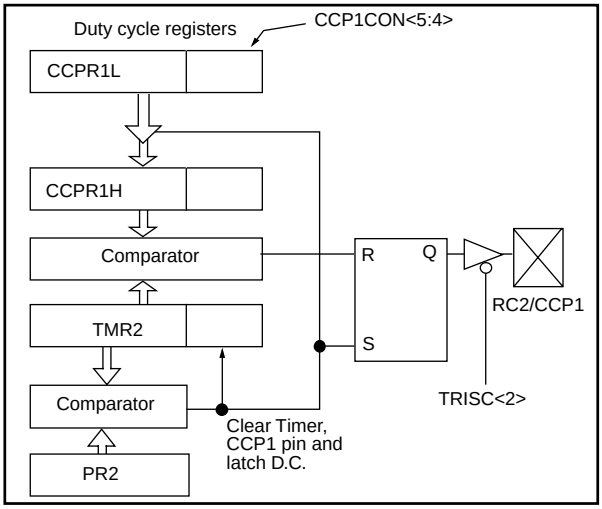
CCP je skraćenica za *Compare/Capture/PWM*. PIC16F877 posjeduje dva ovakva modula koji mu pomažu da se lakše nosi sa raznim zahtjevima *real-time* aplikacija.

U modu *Capture* kada se desi događaj na pinu RC2/CCP1 16-bitna vrijednost tajmera 1 preslikava se u registre CCPR1H:CCPR1L. Istovremeno se setuje korespondentni interapt fleg u registru PIR1. Da bi ova operacija bila uspješna tajmer1 mora raditi u tajmerskom modu ili modu sinhronizovanog brojanja.

U modu *Compare* 16-bitna vrijednost registra CCPR1 se stalno poredi sa vrijednošću para registara tajmera 1. Kada dođe do izjednačenja moguće je da se pin RC2/CCP1 setuje, resetuje ili ostane nepromijenjen, što se ranije podesi odgovarajućim konfiguracionim bajtom upisanim u registar CCP1CON. I u ovom slučaju se setuje interapt fleg CCP1IF.

U modu *PWM* (*Pulse Width Modulation*) pin RC2/CCP1 proizvodi širinsko-impulsni modulisani signal rezolucije do 10 bita. Pojednostavljeni blok-dijagram PWM modula dat je na slici 3.5.

Slika 3.5: POJEDNOSTAVLJENI BLOK-DIJAGRAM PWM MODULA



Period PWM signala određuje se upisom u PR2 registar po formuli:

$$PWM\_period = [(PR2)+1] \cdot 4 \cdot Tosc \cdot TMR2\_preskaler$$

Vrijeme ispune (*Duty Cycle Time*) mijenja se upisom u registar CCPR1L i dva bita registra CCP1CON (biti 5 i 4). Time je omogućena maksimalno 10-bitna rezolucija PWM izlaza. Formula za proračun je:

$$PWM\_Duty\_Cycle = (CCPR1L:CCP1CON<5:4>) \cdot Tosc \cdot TMR2\_preskaler$$

3.3.4. Osvrt na ostale periferije

Mikrokontroler PIC16F877 posjeduje još nekoliko korisnih periferijskih modula koji će u ovom odjeljku biti samo kratko spomenuti.

S obzirom na kontinualnost pojava u spoljašnjem svijetu, teško je upravljati bilo kojim procesom bez digitalizacije analognih veličina. Na većinu zahtijeva može odgovoriti 10-bitni, 8-kanalni A/D konvertor konstruisan na principu sukcesivnih aproksimacija.

Mikrokontroler obično nije usamljen, nego je dio mreže uređaja koji trebaju međusobno komunicirati i razmjenjivati podatke. U tu svrhu, on je opremljen sa tri hardverska komunikaciona modula.

Prvi od njih je SSP modul (*Synchronous Serial Port*), koji služi za komunikaciju sa serijskim EEPROM-ima, pomjeračkim registrima, displej-drajverima, A/D konvertorima, itd. Ovaj modul može raditi u jednom od dva moda:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)

Drugi serijski komunikacioni modul je USART (*Universal Synchronous Asynchronous Receiver Transmiter*). On uglavnom služi za povezivanje sa personalnim računarom, ali to nije njegova jedina mogućnost primjene. USART se može konfigurisati u neki od sljedećih modova rada:

- Asinhroni rad (*full duplex*)
- Sinhroni *master* rad (*half duplex*)
- Sinhroni *slave* rad (*half duplex*)

Osim serijskih, postoji i jedan paralelni komunikacioni modul. Riječ je o modulu PSP (*Parallel Slave Port*). On služi da se PIC16F877 direktno poveže na 8-bitnu magistralu podataka drugog mikroprocesora. Eksterni procesor tada koristeći linije *Read* (RD) i *Write* (WR) može čitati i upisivati PORTD registar kao svaki drugi 8-bitni leč.

3.4. PROGRAMIRANJE

3.4.1. Set instrukcija

U duhu Harvardske RISC arhitekture procesor ima samo 35 instrukcija. One su 14-bitne i sastoje se od operacionog kôda i jednog ili više operanada. Slika 3.6 prikazuje opšti format implementiranih instrukcija, a kompletan pregled instrukcionog seta daje tabela 3.1. Instrukcije su podijeljene u tri kategorije:

- *Byte-oriented* (operacije sa čitavim registrima)
- *Bit-oriented* (operacije sa pojedinim bitovima)
- *Literal & Control* (operacije sa konstantama i upravljačke)

Izvršavanje svake instrukcije traje jedan mašinski ciklus, osim kod grananja programa. Kod potvrdnog odgovora u nekom testu, pribavljena instrukcija se proglašava nevažećom i traži se nova, što rezultuje trajanjem od dva mašinska ciklusa. Dok se u drugom ciklusu pribavlja nova instrukcija, praktično se izvršava naredba NOP.

Jedan mašinski ciklus traje četiri perioda oscilatora što garantuje njegovu dužinu od 1μs pri frekvenciji oscilatora od 4 MHz.

Slika 3.6: OPŠTI FORMAT ZA INSTRUKCIJE

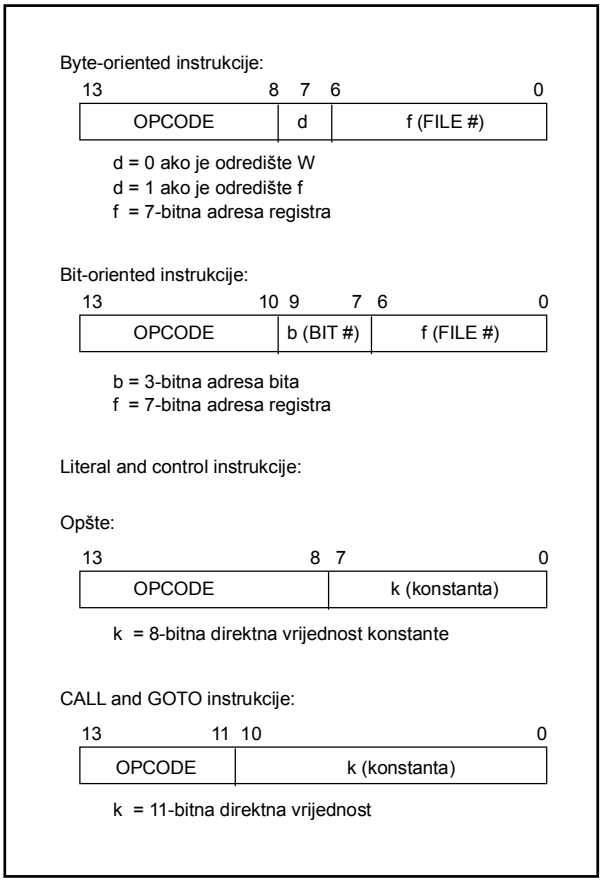


Tabela 3.1: PREGLED SETA INSTRUKCIJA

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Napomene (*notes* u posljednjoj koloni tabele):

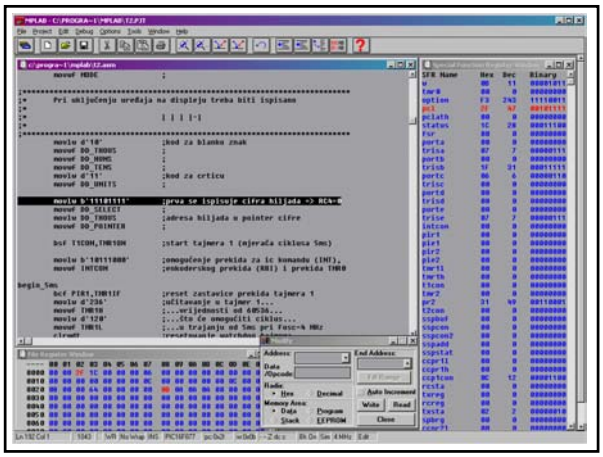
1. Kada se vrši modifikacija I/O registra, a u instrukciji se koristi prethodno njegovo stanje (npr. `MOVF PORTB,1`), kao operand se koristi stanje pročitano sa pinova porta. Tako je moguće da se u leč pina konfigurisanog kao ulaz i postavljenog na nulu preko spoljašnjih elektronskih komponenti, poslije izvršenja instrukcije upiše nula.
2. Ako se ova instrukcija izvrši nad registrom TMR0 (i ako je d=1, gdje je to moguće), preskaler će biti resetovan ako je dodijeljen tajmerskom modulu.
3. Ako je modifikovan programski brojač (PC) ili je rezultat logičkog testa pozitivan, za izvršenje instrukcije biće potrebna dva mašinska ciklusa. U drugom ciklusu praktično se izvršava naredba NOP (*No operation*).

3.4.2. Razvojni softver MPLAB IDE

Microchip besplatno nudi vrlo funkcionalan i stabilan softver za razvoj programa za PIC mikrokontrolere. MPLAB IDE je aplikacija koja se izvršava pod *Microsoft*® *Windows*® operativnim sistemima i koja sadrži:

- Editor kôda (podržani su asembler i C)
- Simulator sa debagerom
- Projekt menadžer

Slika 3.7: GLAVNI PROZOR RAZVOJNOG ALATA MPLAB IDE



3.4.3. Prenos programa u mikrokontroler

Izvorni kôd programa, bilo da je pisan u asembleru ili C jeziku, potrebno je prije upisa u mikrokontroler prevesti u mašinski jezik. To se može uraditi u navedenom *Microchip*-ovom razvojnom softveru MPLAB IDE aktiviranjem komande *Build*, naravno nakon što se otklone sve eventualne greške.

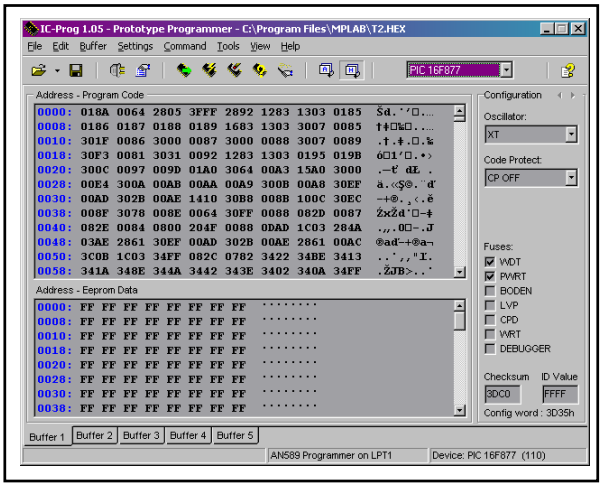
Kao rezultat ove operacije pojaviće se objektni fajl u heksadecimalnom formatu (*.hex) koji nosi niz 14-bitnih instrukcija namijenjenih mikrokontroleru.

Microchip je u startu objelodanio protokol za prenos *hex* fajla u internu programsku memoriju mikrokontrolera, pa su na tržištu prisutni razni programatori i nije obavezna upotreba originalnih. Programator koji je korišćen pri realizaciji ovog projekta je posebno napajani uređaj povezan sa personalnim računarom preko paralelnog (LPT1) porta. Opremljen je odgovarajućim podnožjima za podržane mikrokontrolere. Komunikacija sa nadređenim računarom je sinhrona serijska. To znači da na mikrokontroleru postoje jedan serijski I/O pin (RB7 – pin40) i jedan taktni ulaz (RB6 – pin39). Za generisanje takta PC koristi D1 bit (pin3), a za podatke D0 bit (pin2) paralelnog porta. Sa paralelnog porta potreban je još jedan bit (D3 – pin5) za kontrolu napona programiranja V_{PP}, koji treba da bude prisutan na MCLR ulazu mikrokontrolera za vrijeme programiranja i kreće se u granicama od +12V do +14V.

Kao što je to slučaj sa hardverom, takođe i izbor softvera za programiranje nije ultimativan. Postoji dosta kvalitetnih programa koji posjeduju rutine za prenos mašinskih instrukcija iz *hex* fajla u mikrokontrolersku programsku memoriju. Besplatni program *IC prog* je jedan od njih. Veoma je moćan i osim sa linijom proizvoda PICmicro™, uspješno komunicira sa još dosta programabilnih čipova (uglavnom serijskih EEPROM-a i *flash* mikrokontrolera).

Njegova osnovna funkcija je programiranje – prenos podataka iz PC-a u programabilni uređaj. Međutim, *IC prog* posjeduje još nekoliko vrlo važnih opcija. Moguće je iščitati trenutno stanje memorije mikrokontrolera ili je potpuno izbrisati. Takođe, moguće je izvršiti verifikaciju – provjeru pravilnog upisa podataka poređenjem bafera u računaru sa aktuelnim stanjem memorije mikrokontrolera.

Slika 3.8: POGLED NA INTERFEJS PROGRAMA IC PROG



4 DISPLEJ

4.1. KOMPROMIS U VIDU MULTIPLEKSIRANOG OSVJEŽAVANJA

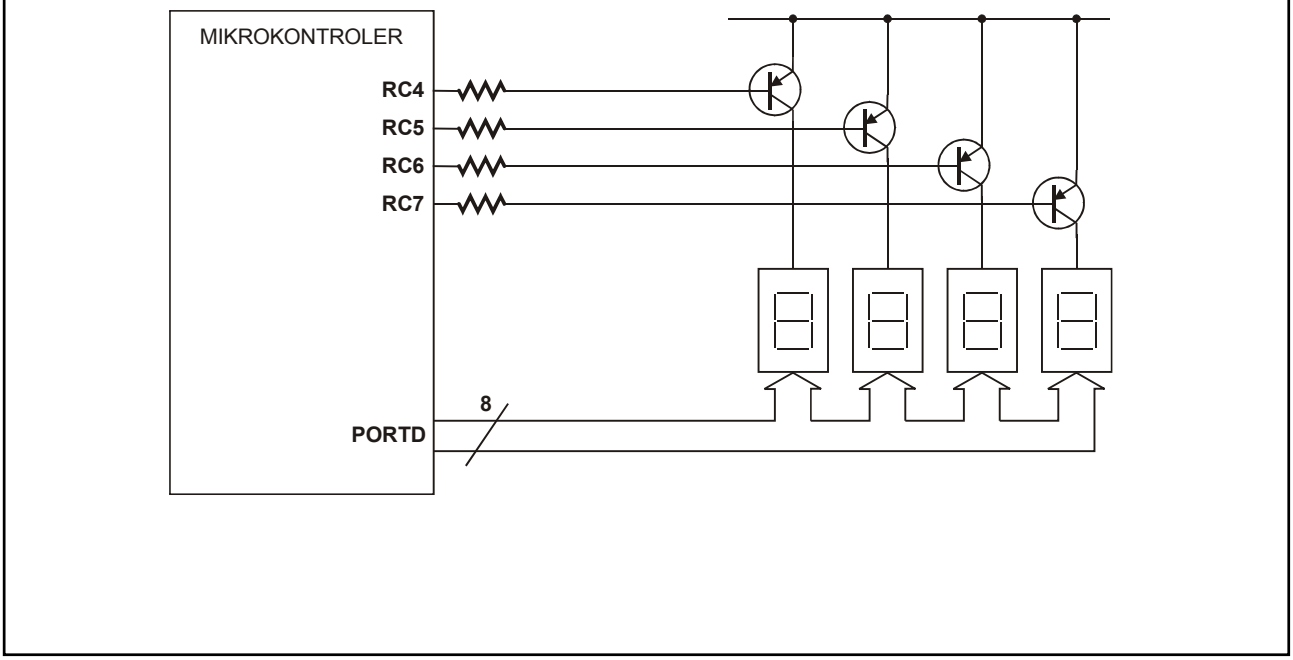
S obzirom da je ovaj antenski pozicioner zamišljen kao kućanski uređaj sa daljinskim upravljanjem, najveći efekat u vizuelnoj kontroli ostvarila bi upotreba LED displeja. Njegova potrošnja je znatnija od ostalih tipova, ali mu je vidljivost najveća.

Imajući u vidu rezoluciju pozicionera, te nepotrebnost da se ispisuje tekst, dobija se da sve potrebe može zadovoljiti četvorocifreni sedmosegmentni displej.

Jasno je da mikrokontroler ne može posebno upravljati svakom od četiri displejske cifre, jer bi mu to okupiralo 32 pina (4×8). Zato je potrebno pribjeći nekoj tehnici povezivanja koja koristi manje komunikacionih linija. Najveću uštedu bi vjerovatno obezbijedilo korišćenje serijske veze sa pomjeračkim registrima čiji bi izlazni lečevi napajali pojedine cifre. Međutim, ovdje je ta mogućnost odbačena, jer je u suprotnosti sa konceptom minimizacije hardvera. Kao neko kompromisno rješenje odabrano je povezivanje u formi multipleksiranog osvježavanja. Tada su sve četiri displejske cifre paralelno povezane na osmobitnu magistralu mikrokontrolera, a posebne četiri linije dovođenjem napajanja na zajedničku anodu određuju samo jednu cifru koja je trenutno aktivna (slika 4.1). Ako se dovoljno brzo izmjenjuju aktivne cifre da se iskoristi inercija ljudskog oka, postiže se utisak stalnog sjaja sve četiri cifre. Obično se savjetuje da frekvencija osvježavanja bude umnožak od 50 Hz zbog raznih izbijanja sa drugim uređajima napajanim iz mreže (npr. fluorescentnim osvjetljenjem). Ovdje je izabrana vrijednost od 250Hz.

Od eksternog hardvera potrebni su otpornici za ograničenje struje i četiri PNP tranzistora male snage. Za lijepu vidljivost LED displej je potrebno napajati sa strujom od oko 5 mA. To znači da u multipleksiranom režimu, kad svaka cifra svijetli četvrtinu vremena, potrebna struja iznosi 20 mA. Toliku struju može direktno dati pin mikrokontrolera i nema potrebe za korišćenjem eksternih drajvera. Međutim, zajednička anoda može "povući" i do 160 mA, pa eto razloga za upotrebu drajverskih tranzistora. Uz sve prehodno navedeno, ograničavajući otpornici na portu D biraju se od 220 Ω, a otpornici u bazi tranzistora od 1,5 kΩ.

Slika 4.1: HARDVER ZA MULTIPLEKSIRANI POGON DISPLEJA



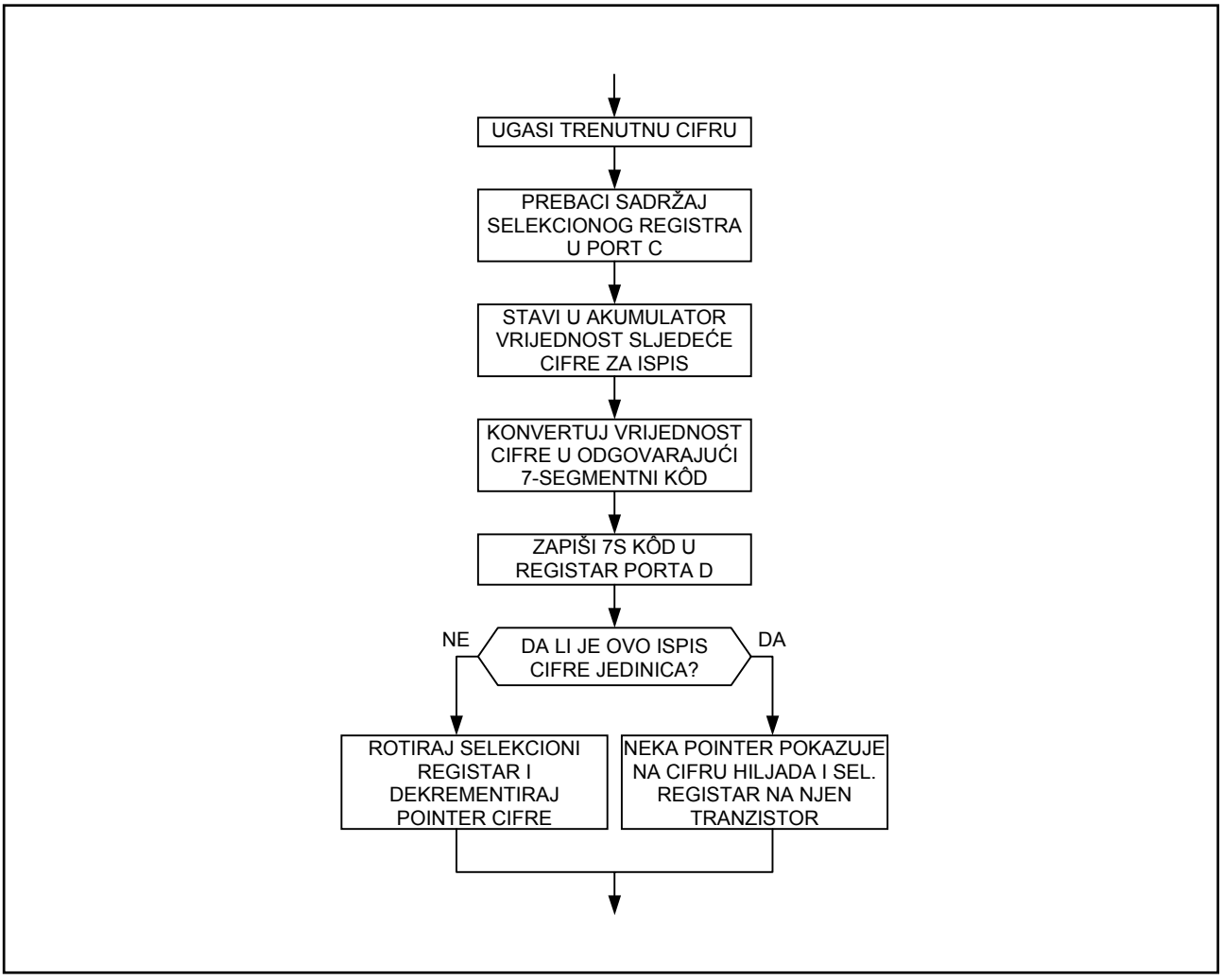
4.2. PROGRAMSKI KÔD

Hardverom opisanom u prethodnom odjeljku upravlja rutina `update_display`. Ona je dio glavne petlje programa, pa se dakle izvršava svaku 1 ms po jednom. Predstavlja jedno veoma racionalno rješenje, jer zauzima svega 46 lokacija u programskom flešu, a izvršava se za tridesetak mašinskih ciklusa mikrokontrolera. Upotrebljava 7 registara opšte namjene iz internog RAM-a (tabela 4.1), a takođe pristupa registrima za indirektno adresiranje. Blok-dijagram rutine za osvježavanje displeja prikazan je na slici 4.2.

Tabela 4.1: UPOTRIJEBLJENI REGISTRI

Simbol. ime	Adresa	Opis namjene
DD_UNITS	0x28	Vrijednost cifre jedinica
DD_TENS	0x29	Vrijednost cifre desetica
DD_HUNS	0x2A	Vrijednost cifre stotica
DD_THOUS	0x2B	Vrijednost cifre hiljada
DD_DIGIT	0x2C	Privremeni registar
DD_SELECT	0x2D	Selektor aktivne cifre
DD_POINTER	0x2E	Pokazivač aktivne cifre

Slika 4.2: BLOK-DIJAGRAM RUTINE UPDATE_DISPLAY



Slika 4.3: LISTING KÔDA RUTINE UPDATE_DISPLAY

```
;**********************************************************************
;**                                                                   **
;**  DISPLAY UPDATE                                                   **
;**  Svako novo izvršenje ovog dijela koda bira sljedeću cifru,       **
;**  postavlja njenu vrijednost na port d i aktivira korespondentni   **
;**  tranzistor za napajanje displeja.                                **
;**                                                                   **
;**********************************************************************
update_display
       movlw 0xFF           ;gašenje trenutne cifre
       movwf PORTD          ;...
       movf PORTCLATCH,w    ;učitaj temp leč porta C
       andlw b'00001111'    ;sačuvaj samo nibl za pobudu motora
       movwf PORTCLATCH     ;vrati u leč novo stanje
       movf DD_SELECT,w     ;učitaj displejske selekcione bite
       andlw b'11110000'    ;maskiraj niži nibl
       iorwf PORTCLATCH,w   ;spoji niblove za motor i displej
       movwf PORTC          ;izbaci na port C
       movwf PORTCLATCH     ;memoriši u temp leč porta C
       movf DD_POINTER,w    ;prebaci pointer trenutne cifre u...
       movwf FSR            ;...registar za ind. adresiranje
       movf INDF,w          ;vrijednost trenutne cifre u akumulator
       call get_7s_code     ;konverzija dec. vrijednosti u 7s kôd
       movwf PORTD          ;dovođenje 7s kôda na segmente
       rlf DD_SELECT,f      ;pomijeranje selekcionog registra
       btfss STATUS,C       ;da li je ispisana zadnja cifra (UNITS)?
       goto new_unit        ;da...priprema novog ciklusa
       decf DD_POINTER,f    ;ne...dekr. pointera trenutne cifre
       goto scan_keypad     ;izlaz
new_unit
       movlw b'11101111'    ;neka je 0 u sel. registru na mjestu...
       movwf DD_SELECT      ;...hiljada (prva cifra u nizu)
       movlw DD_THOUS       ;neka pointer trenutne cifre...
       movwf DD_POINTER     ;...pokazuje na adresu cifre hiljada
       goto scan_keypad     ;izlaz

;**********************************************************************
;**                                                                   **
;**  GET_7S_CODE                                                      **
;**  Podprogram koji binarnu vrijednost iz akumulatora pretvara u kod **
;**  za 7S displej sa zajedničkom katodom.                            **
;**  Raspored na displeju: |a|b|g|f|e|d|dp|c|                         **
;**                                                                   **
;**********************************************************************
get_7s_code
       movwf DD_DIGIT       ;pričuvaj sadržaj iz w
       sublw d'14'          ;oduzmi max vrijednost za prikaz
       btfss STATUS,C       ;da li je došlo do prenosa
       retlw b'11111111'    ;da...vrati praznu vrijednost
       movf DD_DIGIT,w      ;ne...nastavi ispis korektne vrijednosti
       addwf PCL,f          ;dodaj sadržaj iz w programskom brojaču
       retlw b'00100010'    ;kod za '0'
       retlw b'10111110'    ;kod za '1'
       retlw b'00010011'    ;kod za '2'
       retlw b'00011010'    ;kod za '3'
       retlw b'10001110'    ;kod za '4'
       retlw b'01001010'    ;kod za '5'
       retlw b'01000010'    ;kod za '6'
       retlw b'00111110'    ;kod za '7'
       retlw b'00000010'    ;kod za '8'
       retlw b'00001010'    ;kod za '9'
       retlw b'11111111'    ;kod za ' '
       retlw b'11011111'    ;kod za '-'
       retlw b'10001010'    ;kod za 'Y'
       retlw b'00100110'    ;kod za 'N'
       retlw b'01100011'    ;kod za 'C'
```

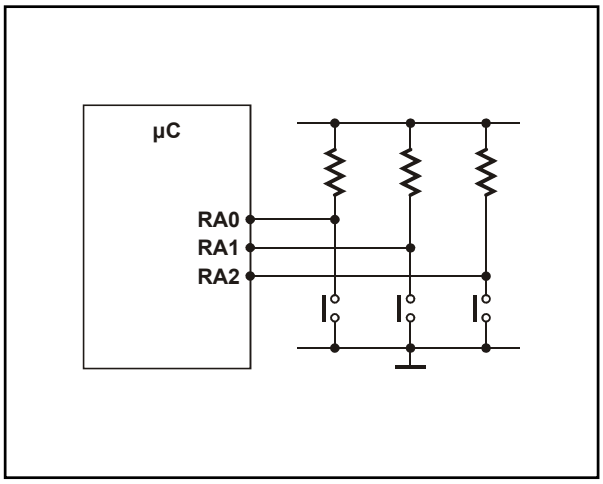
5 TASTATURA

5.1. TEK DA SE NAĐE...

Predviđeno je da uređaj odgovara na dosta korisničkih akcija (unošenje željene pozicije, pozicioniranje, memorisanje,...), pa je očigledna potreba za petnaestak tastera, numeričkih i funkcijskih. Zaista nema realnog opravdanja da se tolika tastatura smjesti na prednji panel uređaja. Mnogo je bolje upravljačke akcije povjeriti daljinskom upravljaču.

Međutim, ne bi bilo loše omogućiti da se na već podešenom i memorisanom uređaju neke akcije mogu izvršiti i direktno. Usvojeno je rješenje da korisnik može proći kroz deset predefinisanih pozicija antene, pritiskanjem tastera *channel up* ili *channel down*. Predviđen je i treći taster – *power*, koji je namijenjen za uključenje/isključenje uređaja.

Slika 5.1: POVEZIVANJE TASTERA SA MIKROKONTROLEROM



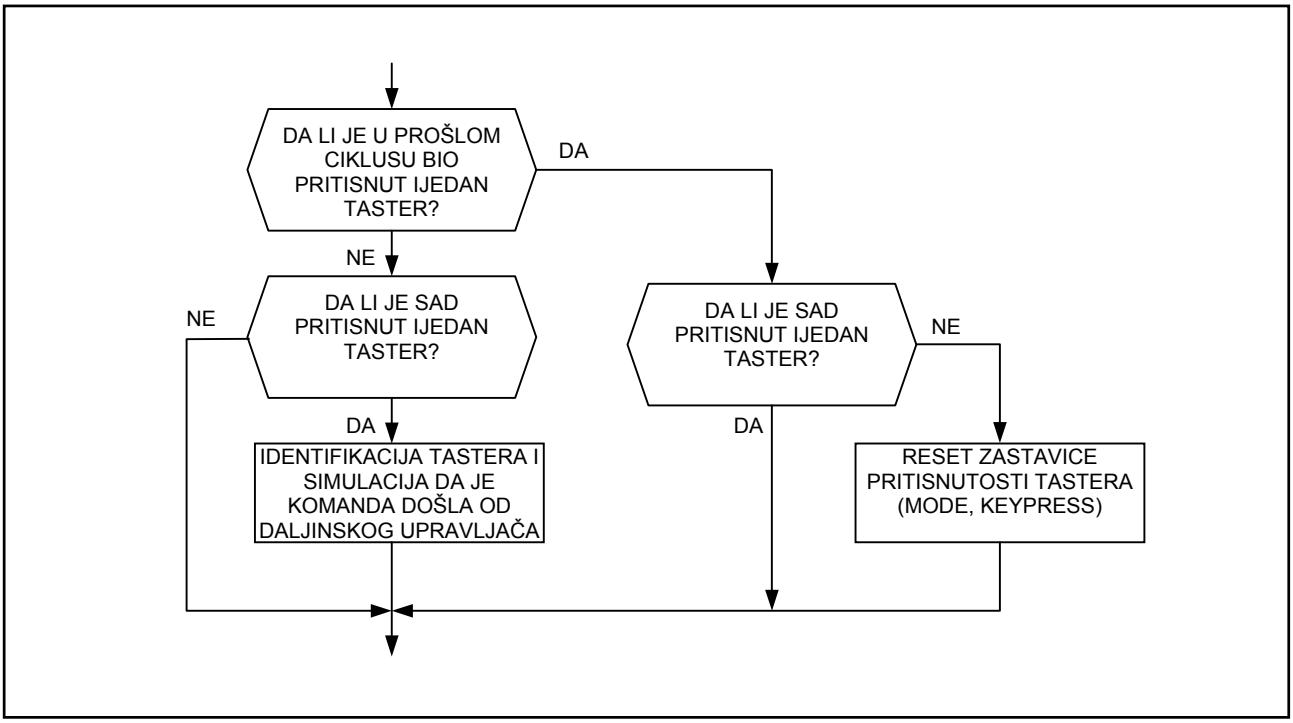
Hardver tastature je zaista jednostavan što pokazuje šema sa slike 5.1. Za svaki taster potreban je samo jedan otpornik, koji služi da sigurno definiše naponski nivo na ulaznom pinu u trenucima kad taster nije pritisnut.

5.2. RUTINA ZA SKENIRANJE TASTERA

Poput rutine za osvježavanje displeja i dio programa koji prati stanja tastera `scan_keypad` smješten je u glavnu programsku petlju. S obzirom da je trajanje pritiska tastera sigurno dosta duže od perioda petlje (1 ms), to je potpuno opravdano i sigurno je da programu neće promaknuti nijedan takav događaj. Međutim, tada se javlja opasnost da korisnik duže vrijeme drži pritisnut taster, pa da program mnogo puta prođe kroz glavnu petlju i ovaj događaj prepozna kao višestruke pritiske. Ovaj problem se rješava upotrebom jednog kontrolnog flega nazvanog KEYPRESS i smještenog u registar MODE zajedno sa ostalim flegovima stanja uređaja. Već pri prvoj detekciji pritisnutosti nekog od tastera, ovaj se fleg setuje i odgovarajuća komanda izvršava. U narednim prolascima kroz rutinu `scan_keypad`, pritisak tastera se ignoriše dok se ne ustanovi da su svi tasteri otpušteni. Tada se fleg KEYPRESS resetuje i tek tada je program spreman za prijem nove komande.

Rutina `scan_keypad` je jednostavna i optimizovana, pa troši izuzetno malo procesorskog vremena. Njen blok-dijagram prikazuje slika 5.2, a kompletan asemblerski listing slika 5.3.

Slika 5.2: BLOK-DIJAGRAM RUTINE SCAN_KEYPAD



Slika 5.3: LISTING KÔDA RUTINE SCAN_KEYPAD

```
;**********************************************************************
;**                                                                   **
;**  SCAN_KEYPAD                                                      **
;**  Rutina za skeniranje tastera spojenih na tri niža pina porta A.  **
;**  Ako je novi taster pritisnut, dalje se simulira kao da je        **
;**  odgovarajuća komanda došla od daljinskog upravljača              **
;**                                                                   **
;**********************************************************************
scan_keypad
    btfsc MODE,KEYPRESS   ;da li je bio pritisnut taster?
    goto unpress          ;da...da vidimo da li je sad
    movlw d'00'           ;pp da nijedan taster nije pritisnut
    btfss PORTA,0         ;da li je pritisnut taster Channel Down?
    movlw d'33'           ;da...njegov kôd u akumulator
    btfss PORTA,1         ;da li je pritisnut taster Power?
    movlw d'12'           ;da...njegov kôd u akumulator
    btfss PORTA,2         ;da li je pritisnut taster Channel Up?
    movlw d'32'           ;da...njegov kôd u akumulator
    andlw b'11111111'     ;samo da se eventualno setuje bit Z
    btfsc STATUS,Z        ;da li je sad pritisnut ijedan taster?
    goto update_mot_pos   ;ne...izlaz
    movwf COMMANDCODE     ;ostavljanje kôda komande u odg. registar
    bsf RC_STAT,VCR       ;simulacija da je komanda došla od daljinskog
    bsf MODE,KEYPRESS     ;postavljanje zastavice pritiska tastera
    goto update_mot_pos   ;izlaz
unpress
    comf PORTA,w          ;komplement PORTA u akumulator
    andlw b'00000111'     ;maskiranje nevažnih bita
    btfss STATUS,Z        ;da li je sad pritisnut ijedan taster?
    goto update_mot_pos   ;da...čekaj otpuštanje
    bcf MODE,KEYPRESS     ;ne...reset zastavice pritiska tastera
    goto update_mot_pos   ;izlaz
```

6 DALJINSKO UPRAVLJANJE

6.1. UVOD U PROBLEMATIKU

I bez ulaženja u sitne detalje, brzo se nameće zaključak da bi daljinsko upravljanje mnogo donijelo na fleksibilnosti i funkcionalnosti uređaja. Jedan razlog za to je, između ostalog, i nemogućnost prednjeg panela da primi obilje tastera, a ako bi se na njima uštedilo, lakoća upravljanja bi izašla iz okvira koji bi bili prihvatljivi za prosječnog korisnika. Visoku fleksibilnost garantuje praktično totalna softverska realizacija, pa su performanse uređaja ograničene uglavnom inventivnošću konstruktora, a bez neke potrebe za izmjenom hardvera.

Daljinsko upravljanje je nešto što se kod savremenih uređaja već odavno podrazumijeva, te se zato pri njegovom dizajniranju treba imati u vidu da korisnik vjerovatno već ima određene navike, pa nebi trebalo odstupiti od postojećih šablona i procedura koje korisnik intuitivno očekuje.

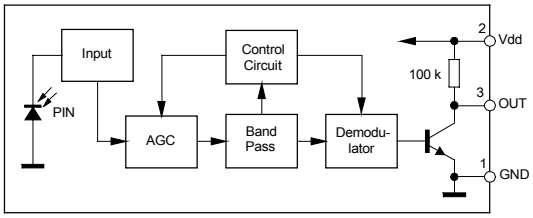
Takođe, iako nije teško ustanoviti sopstveni protokol za komunikaciju, mnogo je pametnije iskoristiti neki od standardnih, provjerenih načina prenosa daljinskih komandi. U ovom slučaju izbor je pao na RC5, veoma popularan i dosta korišćen protokol daljinske komunikacije kojeg je ustanovio *Philips*.

6.2. HARDVER ZA DALJINSKU KOMUNIKACIJU

Njemački proizvođač *Siemens* je omogućio da je u kompletnoj priči o daljinskom upravljanju, hardver zaista najjednostavniji dio. *Siemens*-ova komponenta SFH 506-36 (sl. 6.1) sadrži sve što je potrebno za prijem

Slika 6.1: SFH 506-36

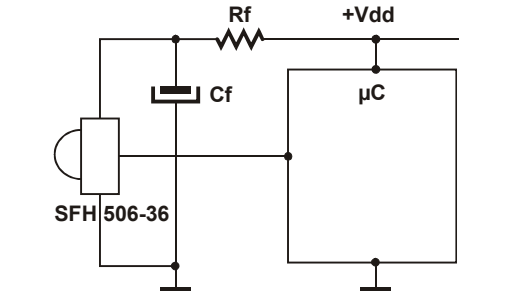
Slika 6.2: INTERNA STRUKTURA SFH 506-36



daljinskih komandi, a sa mikrokontrolerom moguće ju je direktno spregnuti.

Na slici 6.2 moguće je vidjeti internu strukturu SFH 506-36. Fotodioda prima infracrvenu svjetlost talasne dužine od oko 950 nm. Napon koji se na njoj pojavi, pojačava se u bloku sa automatskom kontrolom pojačanja i potom prosljeđuje na izlaz. Evidentno je da izlazni tranzistor uzrokuje inverziju naponskih nivoa, tj. u nepobuđenom stanju na izlazu je logička jedinica, a dolazak infracrvenog snopa uslovljava pad izlaznog napona. TTL naponski nivoi su zagarantovani kada se komponenta napaja sa +5V. Kada je brum napajanja

Slika 6.3: SPREGA IC PRIJEMNIKA SA MIKROKONTROLEROM



mali dodatne pasivne komponente su nepotrebne. U suprotnom pomoći će jedno filter kolo sastavljeno od R_f i C_f kao što prikazuje slika 6.3.

Da bi se obezbijedila veća imunost na smetnje prouzrokovane ambijentalnim svjetlom, predajni signal se moduliše nosiocem 36 kHz. SFH 506-36 posjeduje posebne filtere kako optički za 950 nm, tako u uskopropusni električni filter centralne frekvencije 36 kHz.

6.3. SOFTVERSKO RJEŠENJE

6.3.1. Osvrt na RC5 protokol

U svijetu protokola daljinske komunikacije *Philips*-ov RC5 protokol zauzima značajno mjesto. To je jedan od najpopularnijih i najčešće korišćenih načina kodovanja daljinskih komandi kako za kontrolu kućanskih elektronskih uređaja tako i u industriji za daljinska očitavanja, telemetriju i sl.

Princip komunikacije se sastoji u opštenju preko 14-bitnih riječi. Struktura riječi je naravno strogo precizirana, a organizovana je u obliku:

- dva start bita
- jedan kontrolni (tzv. *toggle*) bit
- pet adresnih bita (kôd uređaja)
- šest komandnih bita (kôd komande)

Vrijednosti start bitâ su uvijek logičke jedinice. Njihova svrha je kalibracija petlje automatske kontrole pojačanja u optičkom prijemniku, a mogu se koristiti za sinhronizaciju pri prijemu, utvrđivanjem perioda komande kao vrijeme proteklo od prijema ivice prvog do prijema ivice drugog start bita.

Sljedeći, kontrolni bit, mijenja svoju vrijednost pri svakom novom pritisku na taster daljinskog upravljača. U slučaju dužeg držanja nekog od tastera dolazi do ponavljanog slanja iste upravljačke riječi i tada se kontrolni bit ne mijenja.

Pet adresnih bita čine kôd odredišnog uređaja. Pošto u isto vrijeme više uređaja može koristiti RC5 protokol, moglo bi doći do neželjenih aktiviranja, pa je zato potrebno da svaki od njih ima zasebnu adresu. Tabela 6.1 pokazuje standardizovane adrese nekih kućanskih uređaja, a u svrhu sprečavanja kolizije sa nekim od njih, za antenski pozicioner izabrana je slobodna adresa 30 (0x1E).

Konačno, posljednjih šest bita predstavljaju kôd same komande. S obzirom na broj komandnih bita, moguće je poslati 64 različite komande po jednom adresiranom uređaju. I komande su takođe standardizovane, a neke najčešće su navedene u tabeli 6.2.

Slika 6.4: KARAKTERISTIKE RC5 PROTOKOLA

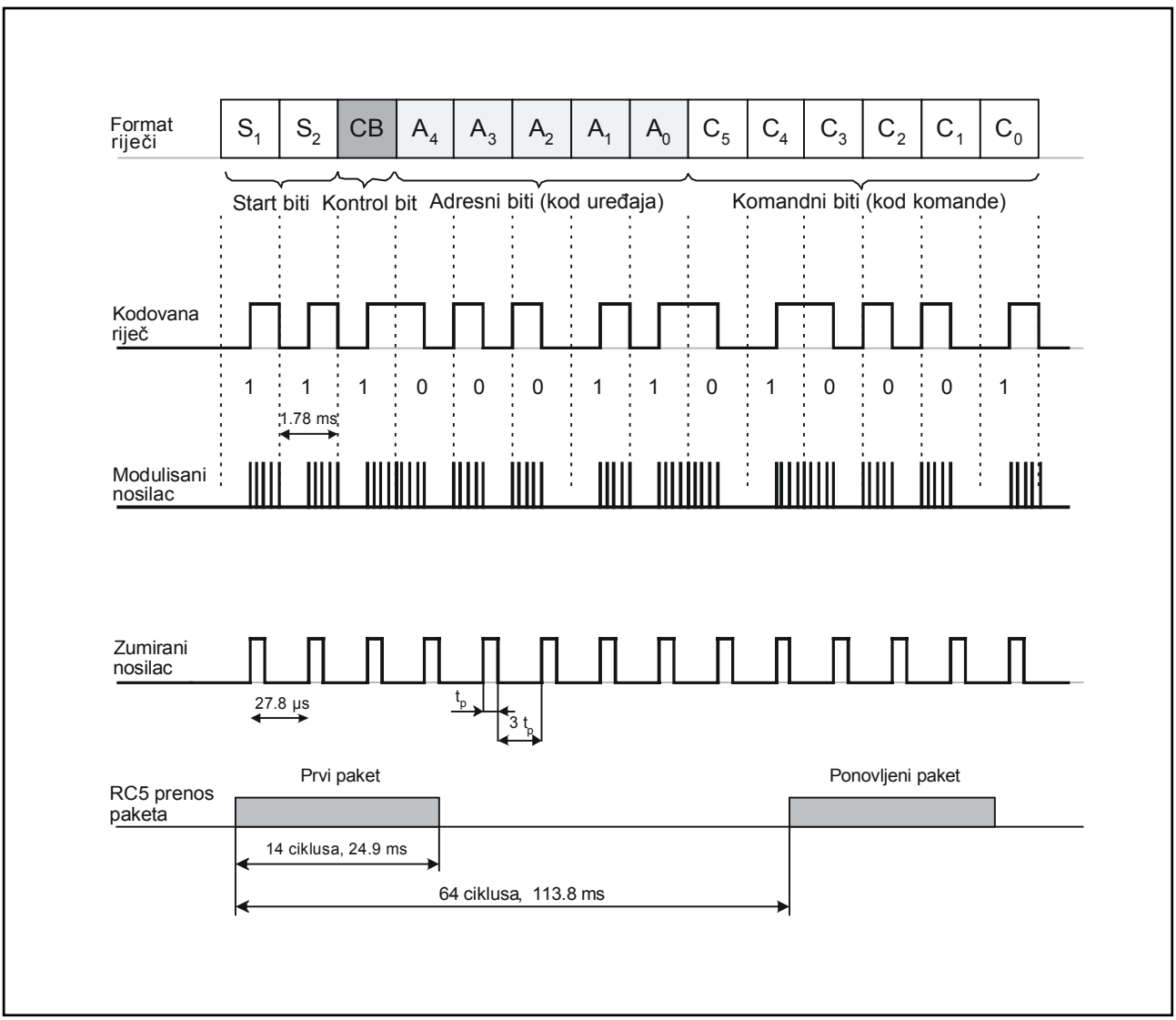
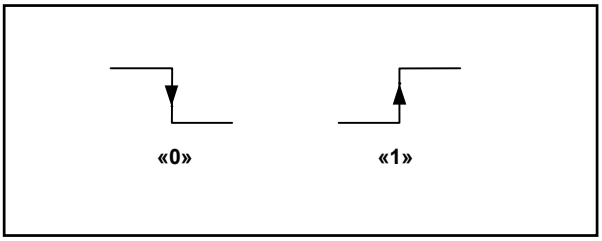


Tabela 6.1: ADRESE ODREDIŠNIH UREĐAJA

Adresa	Odredišni uređaj
0	TV1
1	TV2
5	VCR1
6	VCR2
17	Tuner
20	CD Player
30	Antenski pozicioner

Biti se šalju dvofazno kodovani (tzv. Mančester kôd), gdje je nula predstavljena opadajućom, a jedinica rastućom ivicom signala (slika 6.5). Trajanje jednog bita iznosi 2×889 μs, a nakon slanja prvog paketa od 14 bita, ukoliko je taster i dalje pritisnut, paket identičnog sadržaja se i dalje šalje svakih 113,8 ms.

Slika 6.5: MANČESTER KÔD



6.3.2. Opis softvera za dekodovanje

Nakon što infracrvena svjetlost prođe kroz odgovarajući filter i SFH 506-36 na svom izlazu proizvede povorku digitalnog signala, na softveru mikrokontrolera je da uhvati i dekoduje poslanu mu informaciju. Primjenjeno rješenje je dosta kvalitetno, a neke od njegovih pozitivnih osobina su:

- Baziranost na interaptu, što povlači nekorišćenje prozivanja u petlji (*polling*) i razna čekanja.
- Štednja resursa mikrokontrolera (upotrijebljen je samo jedan osmobitni tajmer kojim se mjere intervali između prekida).
- Fleksibilnost realizacije, što znači mogućnost jednostavne adaptacije za prelazak na druge sisteme daljinske komunikacije kao što su npr. fazno modulisanje ili kodovanje različitim rastojanjem između impulsa.

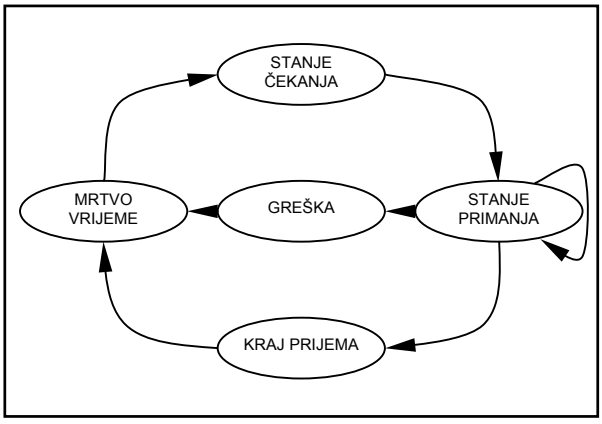
Većinu vremena ovaj softverski prijemnik provodi u stanju čekanja komande. Po nailasku prvog start bita, tj. po detekciji prve rastuće ivice na pinu RB0/INT mikrokontrolera, prijemnik prelazi u mod primanja (*receiving*). U tom modu prijemnik ostaje sve dok ne primi sve bite. Ukoliko se u toku prijema desi neki problem proglašava se greška i poništava prijem tekuće komande. U suprotnom, provjerava se da li je drugi start bit jednak "1" i u slučaju potvrdnog odgovora završava sa prijemom, dostavljajući glavnoj rutini kôd novoprimljene komande. Prije povratka u stanje čekanja, bez obzira da li je došlo do greške ili ne, prijemnik

Tabela 6.2: STANDARDNI KÔDOVI DALJINSKIH KOMANDI

Kôd komande	Komanda
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
12	Stand by
13	Mute/Demute
16	Volume +
17	Volume -
18	Brightness +
19	Brightness -
20	Colour saturation +
21	Colour saturation -
22	Bass +
23	Bass -
24	Treble +
25	Treble -
26	Balance right
27	Balance left
28	Contrast +
29	Contrast -
32	Channel/Program +
33	Channel/Program -
48	Pause
49	Erase
50	Rewind
52	Wind
53	Play
54	Stop
55	Recording

se nalazi u mrtvom vremenu (60 ms) kada je neosjetljiv na nove eksitacije.

Slika 6.6: DIJAGRAM STANJA PRIJEMNIKA DALJ. KOMANDE



Algoritam detekcije je baziran na mjerenju vremenskih intervala između susjednih opadajućih ivica u RC5 signalu. Pošto SFH 506-36 invertuje logičke nivoe procesor radi, ustvari, sa rastućim ivicama i potrebno je u programu podesiti da se interapt generiše dolaskom rastuće ivice na pin RB0/INT. Prvi bit je sigurno jedinica. Sljedeći bitovi se mogu detektovati analizirajući izmjerena vremena prema izrazima:

$$\Delta t_j \in \left[T_P(1-\alpha), T_P(1+\alpha)\right] \Rightarrow b_{j+1} = b_j$$

$$\Delta t_j \in \left[1.5T_P(1-\alpha), 1.5T_P(1+\alpha)\right] \Rightarrow$$
$$b_{j+1} = 1 \text{ i } b_{j+2} = 0 \text{ za } b_j = 1$$
$$b_{j+1} = 1 \text{ za } b_j = 0$$

$$\Delta t_j \in \left[2T_P(1-\alpha), 2T_P(1+\alpha)\right] \Rightarrow b_{j+1} = 1 \text{ i } b_{j+2} = 0$$

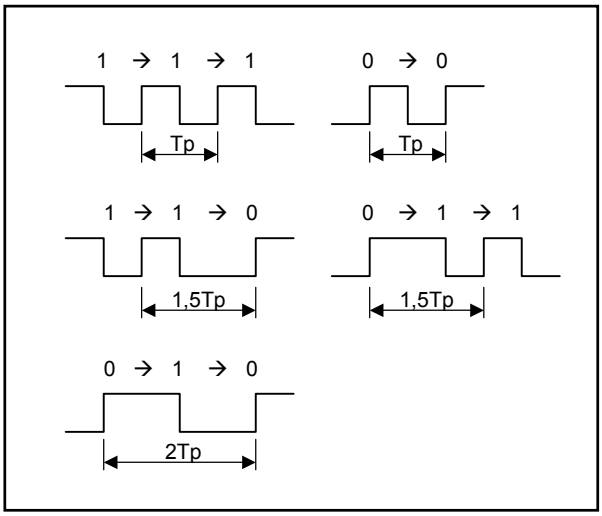
T_P je vrijeme slanja jednog bita čija je nominalna vrijednost po RC5 standardu 1778 μs; Δt_j je izmjereno vrijeme između susjedne dvije opadajuće ivice signala; α je faktor tolerancije koji je u ovom konkretnom primjeru odabran da bude 0,1; b_j je j-ti bit u RC5 14-bitnoj riječi.

Navedeni izrazi se direktno postavljaju nakon analize o kombinacijama koje se javljaju u poretku bita u riječi (slika 6.7).

Kao što je naglašeno ranije detekcioni soft̃ver je potpuno baziran na tehnici interapta, pa je sav kôd vezan za daljinsko upravljanje smješten u interapt servis rutinu (ISR) mikrokontrolerskog programa. U njoj se ispituje trenutno stanje prijemnika, vrši upravljanje tajmerom, ispituje da li su mjerena vremena u potrebnim intervalima i preduzimaju odgovarajuće akcije. Po uspješnom prijemu komande, takođe u ovoj rutini, primljena riječ se potom razbija na toggle bit, kôd uređaja i kôd komande i memoriše respektivno u registre TOGGLECODE, DEVICECODE i COMMANDCODE čije će sadržaje, kada za to dođe vrijeme, iskoristiti glavni program. Princip rada interapt servis rutine dat je blok-dijagramom prikazanom na slici 6.9.

Kôd za obradu prekida tajmera sastoji se samo od jedne linije – skoka na proglašenje greške u `int_isr`, jer do *overflow*-a tajmera u normalnom režimu rada prijemnika ne može doći.

Slika 6.7: KOMBINACIJE U PORETKU BITA



Slika 6.8: UPOTRIJEBLJENI REGISTRI

RC_STAT 0x20 (statusni registar prijemnika daljinske komande)

Bit0	LASTBIT	(vrijednost posljednjeg primljenog bita)
Bit1	RECEIVING	(prijem u toku)
Bit2	TIMEMATCH	(dužina mjerenog vremena odgovara)
Bit3	DEADTIME	(mrtvo vrijeme u toku)
Bit4	NEWBIT	(vrijednost novog bita)
Bit5	TEMP	(privremeni registar bita)
Bit6	VCR	(fleg za ispravni prijem podatka)

RC_BITCOUNT 0x21 (brojač bita)

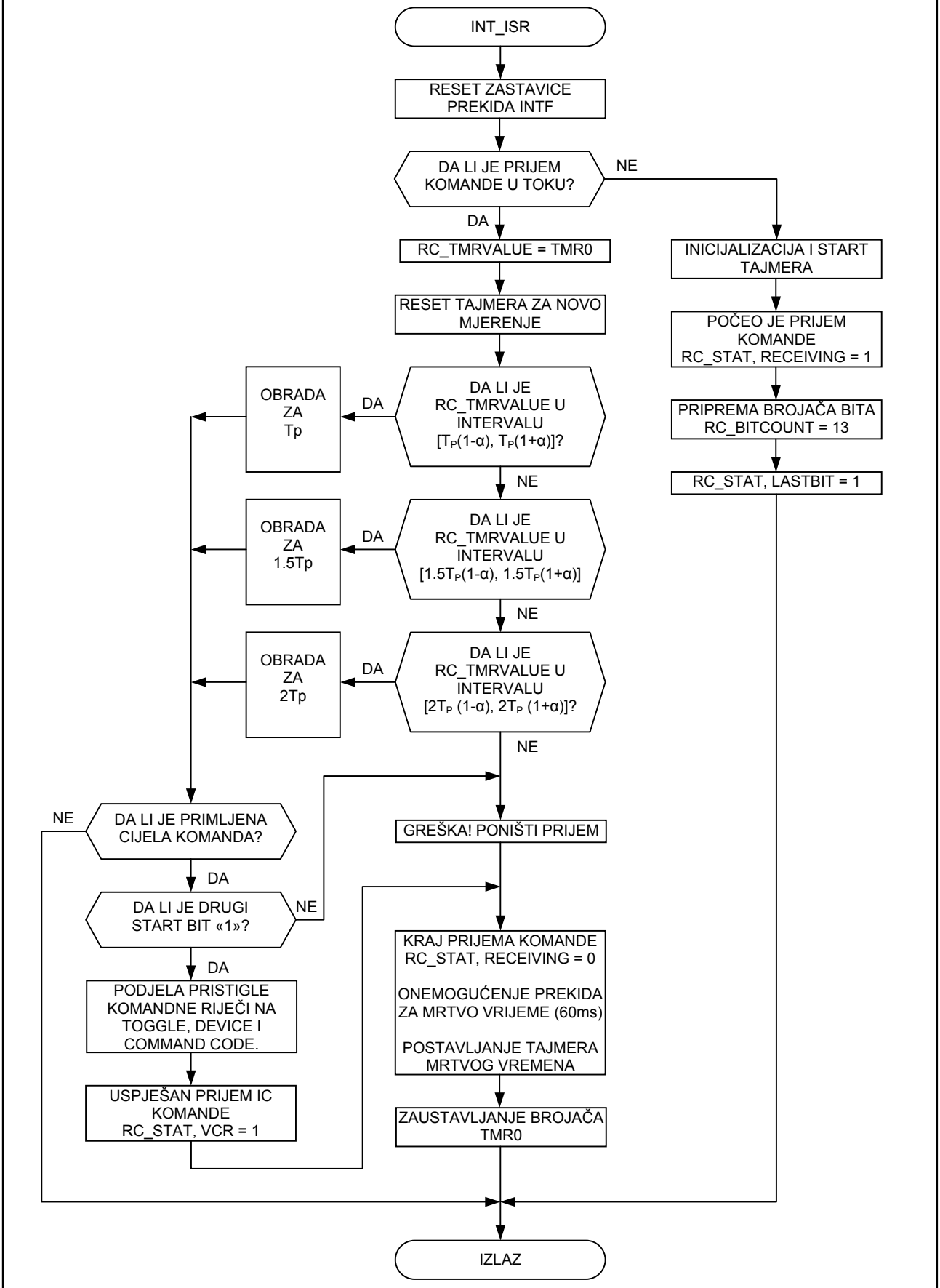
RC_TMRVALUE 0x22 (posljednji izmjereni period između dva interapta)

RC_DEADTIME 0x23 (tajmer mrtvog vremena)

RC_HCOMMAND 0x24 (viši bajt primljene komande)

RC_LCOMMAND 0x25 (niži bajt primljene komande)

Slika 6.9: BLOK-DIJAGRAM RUTINE VANJSKOG PREKIDA (INT_ISR)



7 POZICIONIRANJE

7.1. IZBOR AKTUATORSKOG MOTORA

Kada su sve komponente u upravljačkom dijelu uređaja koliko-toliko usklađene po cijeni i mogućnostima, jasno je da se sa izborom aktuatorskog motora ne smije narušiti taj balans.

Po toj logici odmah treba odbaciti upotrebu asinhronog motora zbog enormno teškog upravljanja pozicijom njegovog rotora. Razvoj regulacionog softvera i konstrukcija pretvarača prevazišli bi višestruko sve ostale aktivnosti i mnogostruko poskupili uređaj.

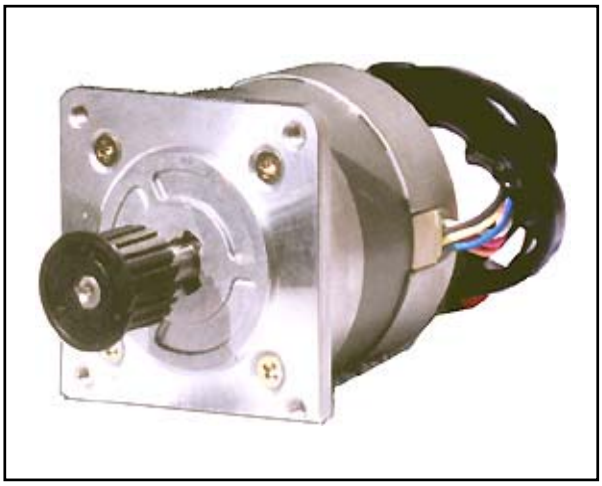
Bolje rješenje bi bilo pozicioniranje antene pomoću jednosmjernog motora. Odgovarajući pretvarači su daleko jednostavniji i lakše upravljivi. Ni cijena takvog motora nije pretjerana. Međutim, da bi u svakom trenutku mikrokontroler raspolagao podatkom o tačnoj poziciji rotora (pa i antene), potrebno je imati nekakav davač pozicije. Dosta veliku rezoluciju mogli bi istjerati korišćenjem optičkog enkodera, ali njegova upotreba je ekonomski neopravdana. Nešto jeftinije rješenje bilo bi mjerenje pozicije potenciometarskim davačem i A/D konvertorom.

Rješenje koje u sijenku baca sva prethodno navedena je pozicioniranje antene pomoću koračnog (*step*) motora. Njime se vrlo jednostavno upravlja, sa nekoliko elektronskih prekidača, bez potrebe za ikakvim vidom D/A konverzije. S obzirom da se pomijeranje rotora step motora vrši u diskretnim koracima, upravljanje se može vršiti bez povratne sprege (*open loop*), jer je dovoljno da mikroprocesor pri svakoj novoj eksitaciji motora osvježi registar u kojem čuva vrijednost trenutne pozicije rotora. Konačno, cijena ovakvih motora nije faktor koji ih stavlja van konkurencije sa ostalima.

Nakon što je step motor izabran kao najprihvatljivije rješenje, potrebno se još odlučiti za njegov tip s obzirom da nisu svi jednoliki nego se proizvode kao motori promjenjive reluktanse i motori sa permanentnim magnetom. Mogućnost pravljenja velikih koraka (od 15 stepeni pa naviše) koju obezbjeđuju motori promjenjive reluktanse više je od koristi u pogonima gdje treba ostvarivati veće brzine. Za predmet ovog rada interesantnije su osobine motora sa permanentnim magnetom: dosta mali ugao koraka i zamašniji moment. Takođe ovi motori posjeduju značajan zadržavajući moment i onda kada kroz namotaje ne protiče struja, pa je moguće održavati željenu poziciju i bez potrošnje el. energije.

Step motor ne treba direktno spregnuti sa antenom, nego preko reduktora sa pužnim prenosom. Tako će sa sprežnim brojem doći do proporcionalnog povećanja rezolucije i smanjenja potrebne snage pogonskog motora. Takođe, zabilježiće se smanjenje spoljnjih uticaja na pomijeranje antene. Gubitak koji stvara uvođenje redukcije je sporije pozicioniranje, ali i to je dio *low cost* kompromisa.

Slika 7.1: STEP MOTOR

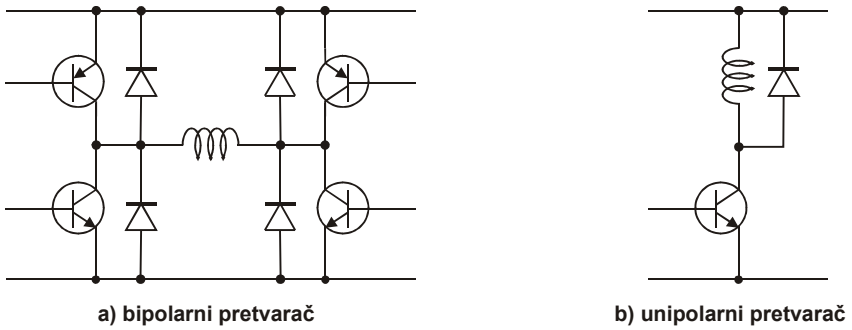


7.2. KAKO UPRAVLJATI STEP MOTOROM?

Step motori se često izrađuju kao dvofazni. To znači da oni na statoru imaju dva odvojena namotaja pomoću kojih se stvara magnetopobudna sila. Četiri su moguće kombinacije u pobuđivanju, zavisno od smjerova struja u svakom od namotaja. Međutim, činjenica da kroz namotaje treba mijenjati smijer struje povlači potrebu za korišćenjem tranzistorskih H-mostova. Dakle, po jednoj fazi statora, u osnovi su potrebna četiri tranzistora, četiri diode i nekoliko elemenata u pobudnom kolu. Dodatni problemi se javljaju sa galvanskim odvajanjem baznih kola tranzistora priključenih na pozitivnu sabirnicu.

Složenost pobudnih kola bila je motiv da proizvođači motora na tržište izbace bifilarno motane modele. Kod takvih motora promjena smijera magnetopobudne sile postiže se puštanjem struje kroz komplementarni namotaj koji je motan oko istih polova, ali u suprotnom smijeru. Zato je umjesto četiri tranzistora u mostnoj konfiguraciji ovdje dovoljan samo jedan elektronski prekidač po fazi statora (slika 7.2).

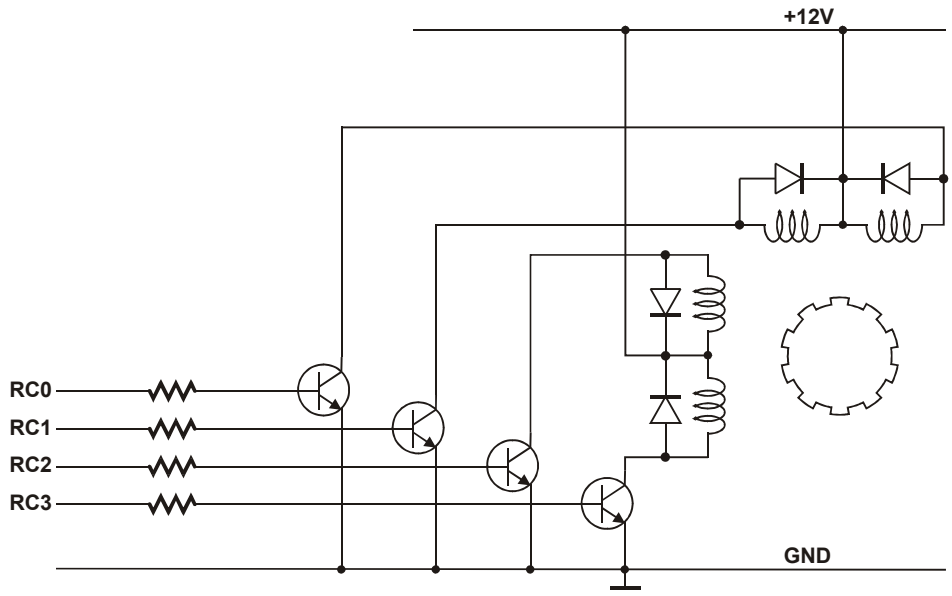
Slika 7.2: UPRAVLJANJE STRUJOM FAZE STATORA



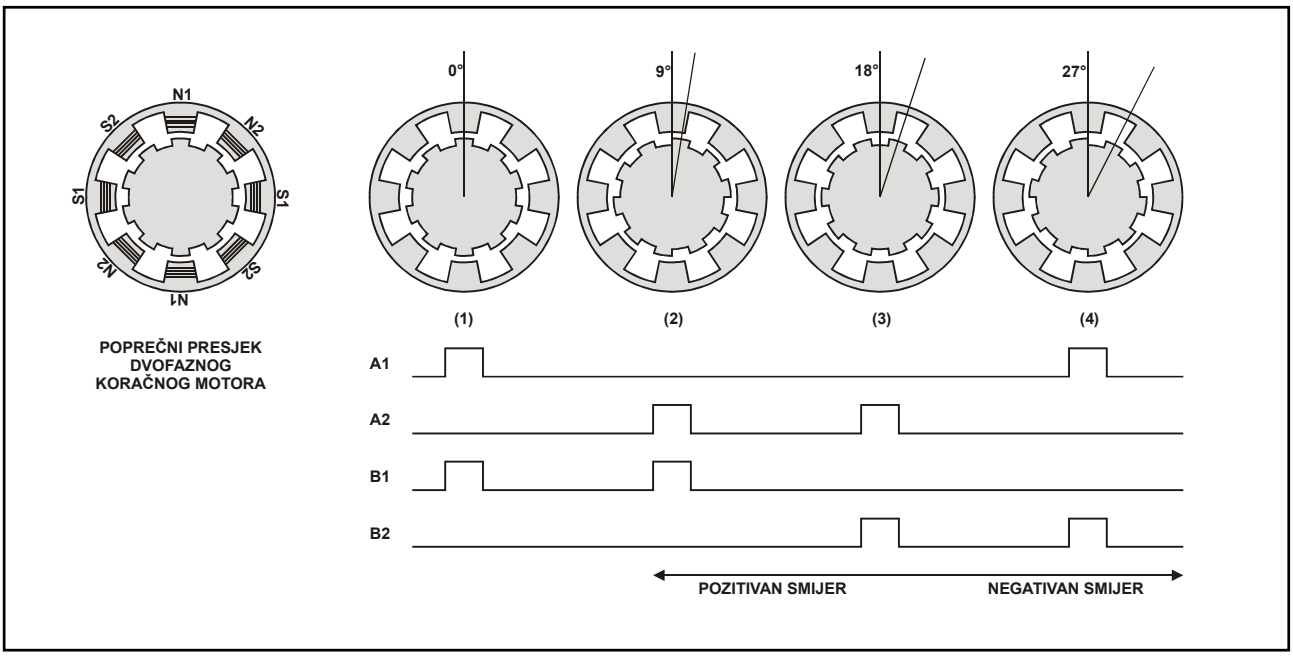
Međutim, nije sve ružičasto kada su u pitanju bifilarni step motori. Jasno je da je potrebno utrošiti duplu količinu bakra za istu snagu motora. Osim toga, konfiguracija H-mosta omogućuje da se pri prekidanju struje kroz namotaj, magnetna energija akumulisana u zavojima ponovo vrati u izvor. Pretvarač za bifilarne step motore to ne omogućuje, pa se nagomilana energija pretvara u toplotu u kolu *freewheeling* diode koja je spojena antiparalelno faznom namotaju. Pa ipak, dokazano je da za manje snage više ima smisla koristiti bifilarno namotane (dakle unipolarne) koračne motore, te je to urađeno i u ovom slučaju. Upotrijebljen je standardni četvorofazni unipolarni motor sa rezolucijom od 200 koraka po obrtu, što će reći da najmanji korak koji se njime postiže iznosi 1,8°. To znači da, ako se želi postići rezolucija pozicionera od 0,1°, prenosni odnos pužnog reduktora treba da iznosi 1,8 : 0,1 = 18.

Kompletnu šemu pretvarača potrebnog za pogon četvorofaznog unipolarnog step motora prikazuje slika 7.3. Kontrola struje kroz statorske faze povjerena je bipolarnim tranzistorima koji rade u prekidačkom režimu. Njima preko ograničavajućih otpornika direktno upravlja mikrokontroler. Četiri su pina neophodna za ovu svrhu, a prirodno se nameće da to budu četiri niža bita porta C, jer su viša četiri upotrijebljena za pogon displejskih cifara. Diode koje su spojene antiparalelno namotajima motora služe kao alternativni put strujama statora pri isključivanju tranzistora, jer se struje kroz induktivnosti ne mogu trenutno prekinuti. Tako drajverski tranzistori bivaju zaštićeni od opasnih prenapona. Umjesto diskretnih tranzistora, bolje je upotrijebiti neko integrisano kolo, npr. SMA4033 koje osim četiri darlingtona posjeduje i odgovarajuće četiri *freewheeling* diode.

Slika 7.3: PRETVARAČ ZA ČETVOROFAZNI UNIPOLARNI STEP MOTOR



Slika 7.4: POBUDNI CIKLUS KORAČNOG MOTORA



Za lakše razumijevanje rada motora koji se pobuđuje pravougaonim naponskim signalima može poslužiti slika 7.4. Na njoj je prikazan step motor sa deset isturenih polova na rotoru i osam isturenih polova na statoru. Na statoru su dvije faze namotane na po četiri nesusjedna pola. Takva konstrukcija obezbjeđuje ugaoni pomjeraj od 9°. U prvom dijelu pobudnog ciklusa pobuđuju se namotaji A1 i B1 i tada se maksimalni fluks pojavljuje na polu N1; rotor zauzima položaj N1 u kome je par njegovih isturenih polova postavljen naspram statorskih polova N1. U drugom dijelu ciklusa pobuđuju se namotaji A2 i B1; maksimalni fluks se pojavljuje na polovima N2 i rotor tada izvrši ugaoni pomjeraj (korak) od 9°. U trećem dijelu ciklusa pobuđuju se namotaji A2 i B2, a u četvrtom namotaji A1 i B2. U svakom dijelu pobudnog ciklusa rotor izvršava pomak od 9°. Na taj način cijelom pobudnom ciklusu će odgovarati ukupan ugaoni pomjeraj od 36°, što važi za motor koji na rotoru ima deset isturenih polova. Analogno se dešava i kod motora sa korakom od 1,8° koji je upotrijebljen u antenskom pozicioneru. Dokle god se motor pobuđuje povorkom impulsa 12341234… kao na slici 7.4 trajaće i rotacija u smijeru kazaljke na časovniku. Obrnutim redoslijedom pobuđivanja 43214321… bi se postiglo obrtanje rotora u suprotnom smijeru. Odgovarajuće pobude tranzistora, tj. vrijednost nižeg nibla porta C mikrokontrolera za oba smijera rotacije date su u tabeli 7.1.

Tabela 7.1: VRIJEDNOSTI POBUDNOG NIBLA PORTA C

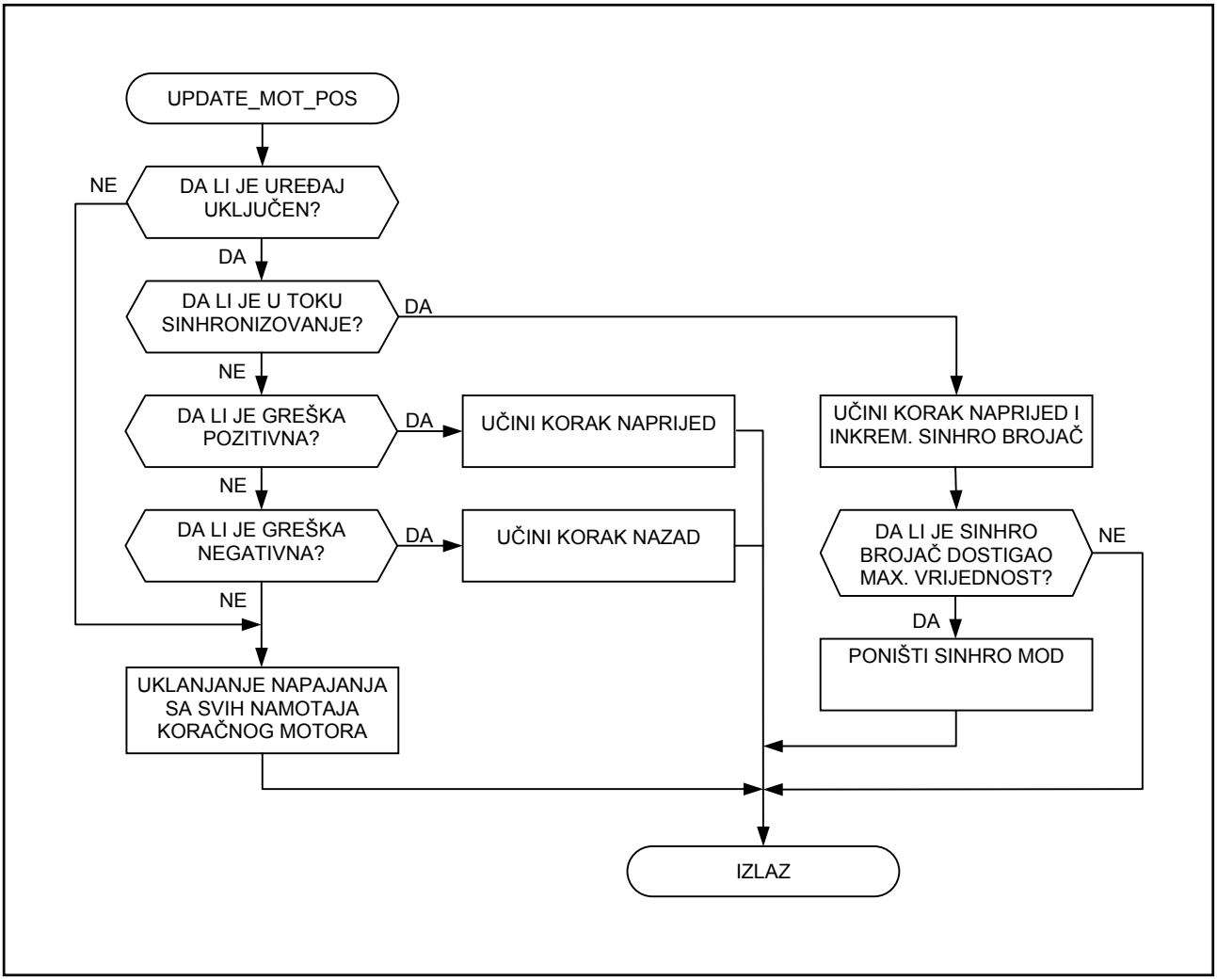
faza koraka	neg. smijer	poz. smijer
1	0101	0101
2	0110	1001
3	1010	1010
4	1001	0110

7.3. SOFTVERSKA PODRŠKA

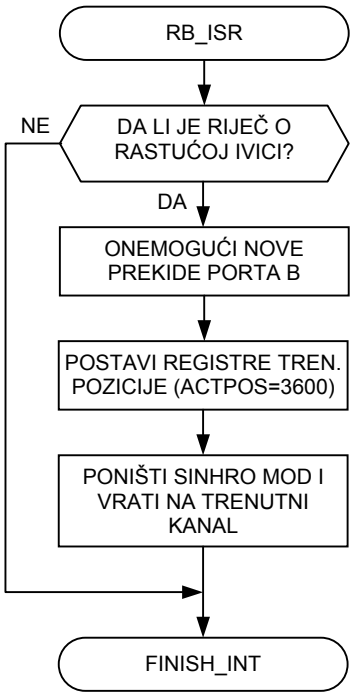
Generisanje pobudnih sekvenci vrši naravno mikrokontroler. Procedura za upravljanje motorom (`update_mot_pos`) dio je glavne programske petlje, pa se tako periodično izvršava. Međutim, pošto je predviđeni period koraka 2 ms, a period glavne programske petlje iznosi 1ms, potrebno je preskočiti svaki drugi ciklus. U okviru te procedure mikrokontroler ispituje uključenost uređaja i u slučaju potvrdnog odgovora poredi vrijednost željene pozicije sa trenutnom. Ako greška u poziciji postoji izvršava se korak motora ka njenom smanjenju. Ako je greška anulirana ili ako je uređaj u *stand by* modu uklanja se napajanje sa svih namotaja koračnog motora.

Poseban režim ovog uređaja predstavlja sinhronizovanje (*synchro mode*). On nastaje kada korisnik na daljinskom upravljaču izabere odgovarajuću komandu (*Sync*). Tada se inicira obrtanje motora ka krajnjoj poziciji (360,0°) i čekanje impulsa od krajnjeg prekidača postavljenog na referentnom položaju. Po dolasku ivice signala sa prekidača sinhronizovanje se završava. Registri trenutne pozicije (`ACTPOSH:ACTPOSL`) tada se pune tako da pokazuju na 360,0°. U sljedećim izvršenjima rutine `update_mot_pos` motor opet nastavlja "potjeru" za traženom pozicijom. Razlog za implementiranje sinhronizacije je u tome što se motorom upravlja bez povratne sprege. Ako iz bilo kojeg razloga mikrokontroler izgubi tačnu trenutnu poziciju antene, pritiskom na samo jedan taster, za kratko vrijeme on je ponovo hvata. Kao dodatna mjera opreza protiv mogućnosti da otkaže krajnji prekidač, te da se uspostavi beskonačno okretanje, uveden je sinhro brojač koji broj koraka unaprijed ograničava na 3840, što je dovoljno jer to iznosi nešto više od punog kruga.

Slika 7.5: BLOK-DIJAGRAM RUTINE ZA POSTAVLJANJE NOVE POZICIJE MOTORA (UPDATE_MOT_POS)



Slika 7.6: BLOK-DIJAGRAM PREKIDNE RUTINE EKSTERNOG PREKIDA PORTA B (RB_ISR)



8 MODOVI RADA I UPRAVLJANJE UREĐAJEM

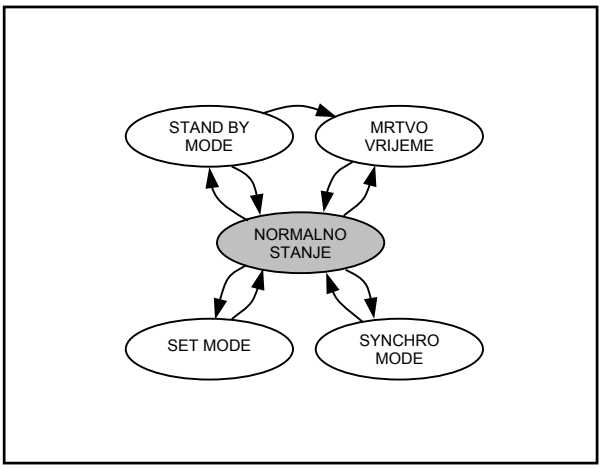
8.1. UVOD

U prva dva poglavlja ukratko su opisani projektni zadatak i koncepcija uređaja. U ostalim poglavljima razmatrane su pojedine komponente pozicionog sistema. Ovdje se govori o načinu sadejstva pojedinih dijelova u finalnoj praktičnoj realizaciji, dakle jedan globalni pogled na funkcionisanje uređaja u cjelini.

8.2. MODOVI RADA UREĐAJA

Pod modovima se podrazumijevaju stanja u kojima se uređaj može naći tokom svog ispravnog funkcionisanja. Korišćenje tehnike modova olakšava programeru kontrolu nad kodom. Za svaki mod vezuju se odgovarajuće mu osobine i mogućnosti, pa se tako smanjuje mogućnost nepredvidivog ponašanja uređaja.

Slika 8.1: DIJAGRAM STANJA UREĐAJA



Registar MODE (slika 8.2) sadrži zastavice (*flags*) pomoću kojih program u svakom trenutku može prepoznati trenutno stanje uređaja. Po prelasku iz jednog u drugi mod obavezno se ažurira ovaj registar. Bit POWER je namijenjen da se razlikuje mod uključenog uređaja od *Stand by* moda. Bitovi SET i SYNCHRO pokazuju boravak u istoimenim modovima. Jedino bit KEYPRESS nema odgovarajući mod nego samo predstavlja indikaciju da je neki od tastera na prednjem panelu uređaja trenutno pritisnut.

Nakon što se uređaj priključi na napajanje i počne izvršavanje programa mikrokontrolera, neposredno se dolazi u *Stand by* mod. U njemu je motor isključen, na displeju je prikazana samo jedna crtica i uređaj ne reaguje na korisničke komande. Jedino je aktivna komanda za uključenje (*Power*), koja nakon kratkog mrtvog vremena (250 ms) uređaj prevodi u normalno stanje. Mrtvo vrijeme je potrebno da se prijemnik infracrvenih signala dovede u oblast stabilnog funkcionisanja. Osim neposredno poslije *Startup*-a, uređaj boravi u mrtvom vremenu i poslije prijema nove IC komande. Predviđeno je da to vrijeme bude dosta kratko (60 ms), ali ono je dovoljno da se izbjegnu neke smetnje, te da se odbace svi biti poslije četrnaestog koji su inače prisutni u verzijama protokola za daljinsko upravljanje nadograđenih na RC5 standard.

U normalnom stanju uređaj provodi većinu vremena. Povremeno korisnik svojim akcijama izaziva prelaske u mod podešavanja pozicije (S*et mode*), a po potrebi nekad i u mod sinhronizovanja sa referentnim položajem (*Synchro mode*).

Slika 8.2: SASTAV REGISTRA MODE

-	-	-	-	SYNCHRO	SET	KEYPRESS	POWER

Bit0	POWER	(indikacija uključenosti uređaja)
Bit1	KEYPRESS	(znak da je pritisnut neki od tastera)
Bit2	SET	(pokazuje da je uređaj u modu unošenja nove pozicije)
Bit3	SYNCHRO	(pokazuje da je uređaj u modu sinhronizovanja sa referentnim položajem)
Bit4:7	-	(nisu iskorišćeni)

8.3. UPRAVLJAČKE KOMANDE

8.3.1. Prijem daljinske komande

Gledano na vremenskoj osi, primanja komandi od daljinskog upravljača su u principu dosta rijetki događaji koji imaju stohastički karakter. Tako je kao najbolje rješenje usvojeno da ivica naponskog signala sa infracrvenog prijemnika SFH506-36, kad je to potrebno, generiše prekid kod mikrokontrolera, a da se inače ne troši procesorsko vrijeme na stalno provjeravanje. Kako je to opisano u poglavlju 6 koje govori o daljinskom upravljanju, po ispravnom prijemu komande njen se kôd stavlja u registar `COMMANDCODE`, a potom se postavlja zastavica za ispravan prijem (bit `VCR` u registru `RC_STAT`). Tada je već moguće da mikrokontroler pređe na izvršenje operacije koju mu je korisnik zadao.

8.3.2. Obrada primljene komande

Predviđeno je da u okviru glavne programske petlje u svakom prolasku mikrokontroler testira stanje bita za ispravan prijem daljinske komande (`VCR`). Kada se ustanovi da je taj bit postavljen na logičku jedinicu, vrši se grananje programa odlaskom na izvršenje rutine za procesiranje primljene komande (`proc_comm`). Kôd komande, koji leži u opsegu od 0 do 63, dodaje se kao ofset programskom brojaču i tako se dalje bira odgovarajuća podprocedura. Od moguće 64 komande, koliko obezbjeđuje RC5 protokol daljinskog upravljanja, u ovom uređaju iskorišćeno je 17. Računajući na to da je za obradu komande nekog od deset numeričkih tastera predviđena jedinstvena procedura `exec_digit`, broj posebnih rutina sa 17 sveden je na 8 (tabela 8.1).

Tabela 8.1: SPISAK PROCEDURA ZA OBRADU KOMANDI

Komanda	Procedura	Kôd
Uključenje/isključenje uređaja	`exec_power`	12
Numerički tasteri (0-9)	`exec_digit`	0-9
Dekrementiranje broja kanala	`exec_cnlup`	32
Inkrementiranje broja kanala	`exec_cnldown`	33
Ulazak u môd podešavanja	`exec_set`	54
Pokretanje motora	`exec_go`	53
Memorisanje pozicije	`exec_mem`	55
Iniciranje sinhronizacije	`exec_sync`	50

U daljem tekstu opisuju se načini izvršavanja pojedinih komandi. Na slikama su prikazani neki od karakterističnih blok-dijagrama.

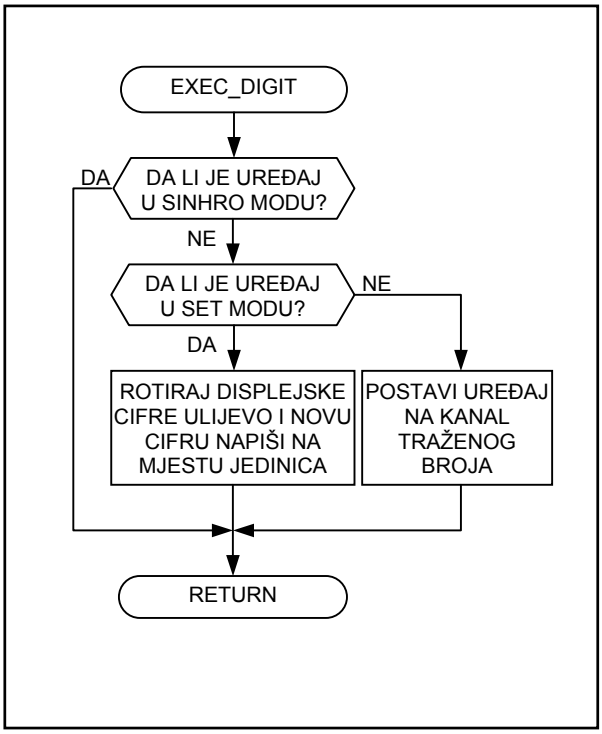
Komanda *Power* preko `MODE` registra ispituje stanje uređaja, pa za isključen uređaj vrši njegovo uključenje i obrnuto. Za vrijeme sinhronizacije sa referentnim položajem ili za vrijeme unošenja nove željene pozicije ova komanda je onemogućena. Ako se vrši uključivanje uređaja, iz internog EEPROM-a mikrokontrolera se iščitava broj aktivnog kanala i trenutna pozicija antene snimljene prilikom zadnjeg isključenja. Analogno se, u slučaju isključivanja, vrši memorisanje aktivnog kanala i trenutne pozicije u interni EEPROM. Tako je osigurano da i u slučaju totalnog ukidanja napajanja vitalni podaci budu sačuvani. Čitanje iz EEPROM-a je jednostavno i izgleda kao čitanje iz internog RAM-a, međutim za upisivanje je potrebno ispoštovati posebnu proceduru koju propisuje *Microchip*, pa je nešto sporije i složenije.

Slika 8.3: OBAVEZNI DIO KÔDA ZA UPIS U INTERNI EEPROM

```
    BANK2
    movf PODATAK     ;učitaj podatak
    movwf EEDATA     ;prebaci ga u registar
                     ;podatka EEPROMa
    bsf STATUS,RP0   ;bank 3
    bcf EECON1,EEPGD ;potvrda da se piše u
                     ;DATA memoriju
    bsf EECON1,WREN  ;omogućenje upisa
    bcf INTCON,GIE   ;onemogućenje prekida
    movlw 0x55       ;upisivanje…
    movwf EECON2     ;sigurnosnog koda 0x55
    movlw 0xAA       ;upisivanje…
    movwf EECON2     ;sigurnosnog koda 0xAA
    bsf EECON1,WR    ;iniciranje upisa
    BANK0
write_wait
    btfss PIR2,EEIF  ;kraj upisa u EEPROM?
    goto write_wait  ;ne...čekaj još
```

Ako se na daljinskom upravljaču izabere neki od cifarskih tastera i mikrokontroler ispravno primi komandu, poziva se procedura `exec_digit`. Kako to prikazuje blok-dijagram sa slike 8.4, a zavisno u kojem modu je uređaj tada, implementira se odgovarajuće djelovanje.

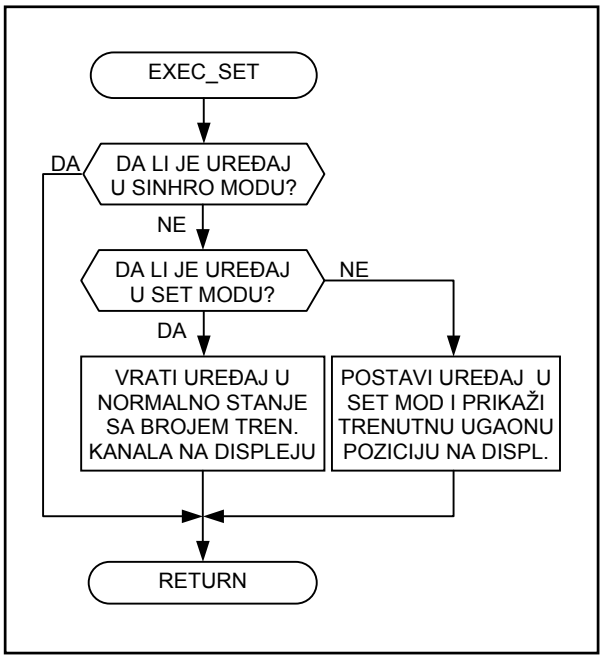
Slika 8.4: BLOK-DIJAGRAM PROCEDURE EXEC_DIGIT



Funkcije *Channel down* i *Channel up* nemaju samostalna rješenja nego se djelimično oslanjaju na proceduru `exec_digit`. Po prijemu jedne od ove dvije komande računa se broj kanala na koji treba prebaciti, a potom se pozove navedena procedura za obradu cifre.

Komanda *Set* služi da se iz normalnog stanja uređaj prevede u mod unosa nove pozicije antene i obrnuto. Slično drugim komandama i ova će komanda biti ignorisana ako je uređaj u sinhro modu.

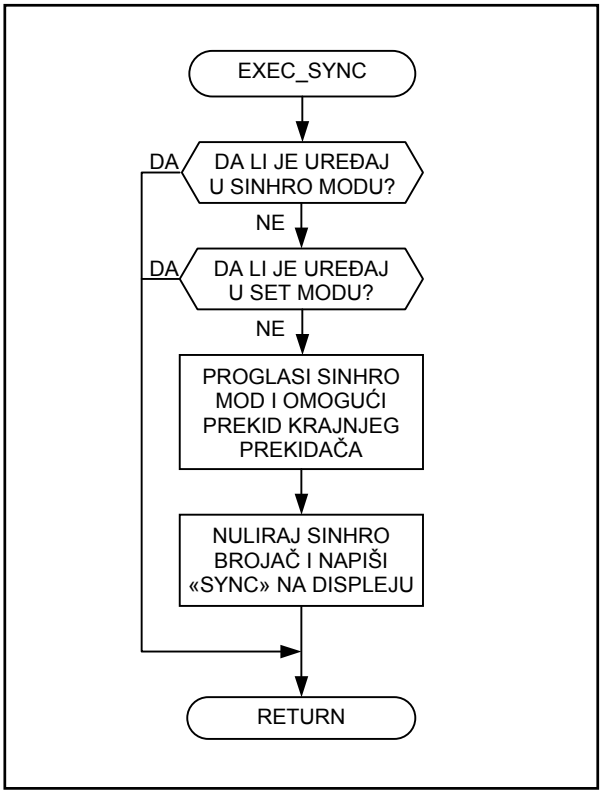
Slika 8.5: BLOK-DIJAGRAM PROCEDURE EXEC_SET



Kada je uređaj u set modu, pomoću cifarskih tastera korisnik unosi novu željenu poziciju. Rezolucija iznosi 0,1°, pa je moguće unijeti uglove od 0° do 359,9°. Tada su moguća tri izbora. Aktivira li se funkcija *Go*, mikrokontroler sa displeja očitava broj, stavlja ga u registre tražene pozicije (`REQPOSH:REQPOSL`) i time automatski inicira rotaciju antene ka željenom ugaonom položaju. Funkcija *Mem* će uraditi isto što i funkcija *Go*, ali će osim toga izvršiti i memorisanje tražene pozicije na trenutno aktivni kanal. Konačno, moguće je ponovo pokrenuti komandu *Set* kada se uređaj vraća u normalno stanje i antenu vraća na poziciju koja je bila posljednja memorisana na tom kanalu.

O potrebama za sinhronizovanjem sa referentnim položajem bilo je riječi u prethodnom poglavlju. Većina stvari vezano za hvatanje referentne pozicije odvija se u okviru rutine `update_mot_pos` zadužene za upravljanje motorom. Međutim, za iniciranje kompletnog procesa zadužena je procedura `exec_sync` koju izaziva pritisak na *Sync* taster na daljinskom upravljaču. Kako to prikazuje slika 8.6, ako uređaj već nije u nekom od specijalnih modova, proglašava se sinhro mod i vrši inicijalizacija sinhronizovanja.

Slika 8.6: BLOK-DIJAGRAM PROCEDURE EXEC_SYNC



9 ZAKLJUČAK

9.1. OSNOVNI CILJ JE ISPUNJEN

Proces projektovanja pozicionera antene stalno je bio praćen praktičnim provjeravanjem putem detaljnih eksprerimenata, što predstavlja osnovnu vrijednost ovoga rada. Tokom realizacije elektronskog sklopa autor je imao priliku da ustanovi koje je uslove potrebno ispuniti da se obezbijede ispravan start i pravilno izvršavanje programa mikrokontrolera, kako imati stabilne oscilacije, koje komponente je neophodno hladiti i kako, na koji način napajati pojedine dijelove uređaja,…

Iako je finalna verzija programa mikrokontrolera sačinjena od oko 800 asemblerskih linija, tokom razvoja je napisano i testirano bar 3000. Za vrijeme kodiranja od velike koristi je bio simulator koji je u sastavu *Microchip*-ovog razvojnog soſtvera MPLAB IDE, mada je dosta kôda bilo testirano direktno na hardveru nakon upisa u mikrokontroler. Tokom eksperimentisanja na raspolaganju nije bio nikakav *In-circuit debugger*, koji bi omogućio otklanjanje grešaka u kôdu dok mikrokontroler radi u ciljnom uređaju. To je svakako otežavajuća okolnost.

No, uprkos restrikcijama vezanim za laboratorijske uslove, električni dio planiranog antenskog pozicionera uspješno je projektovan i praktično realizovan. U finalnom proizvodu evidentna je stabilnost u radu kako mikrokontrolera, tako i popratnih integrisanih i diskretnih komponenata. Program je na kraju uspješno očišćen od bagova i posjeduje ugrađene mehanizme protiv slučajnih ili namjernih nelogičnih korisničkih akcija. Takođe, postoji nekoliko zaštitnih mjera u slučaju otkaza nekih komponenata sistema.

Među vrlo uspješno realizovane treba ubrojiti i podsistem za daljinsko upravljanje – kako njegov hardverski tako i soſtverski dio. Ispostavilo se da je prijemnik infracrvenih signala zaista kvalitetna komponenta, koja je bilježila domet i preko dvadeset metara uz visoku imunost na ambijentalno svjetlo. Prekidna rutina za prijem RC5 daljinskih komandi radi besprijekorno i nikakve greške nisu primjećene tokom eksperimentisanja sa univerzalnim daljinskim upravljačem.

Multipleksirani pogon četvorocifrenog LED displeja je takođe uspješno riješen, ali nasuprot tvrdnji proizvođača, ispostavilo se da 5 mA možda i nije dovoljna jačina struje da se dobije dobra vidljivost. Tako bi ipak trebalo koristiti nekakav drajver između pinova porta i displejskih segmenata koji će moći obezbijediti zadovoljavajući pogon.

Tri tastera predviđena za prednji panel uređaja zbog manje važnosti nisu ni montirana u praktičnoj realizaciji.

Integrisano kolo SMA4033 namijenjeno za pogon step motora ispostavilo se kao odlično rješenje. Četiri darlington tranzistora posjeduju dosta visok koeficijent strujnog pojačanja, pa su izlazni pinovi mikrokontrolera duboko podopterećeni. Pošto je pri testiranju korišćen motor koji povlači 400 mA po fazi, a garantovana kolektorska struja izlaznog tranzistora iznosi 2 A, pokazalo se da kolu nije potreban dodatni hladnjak. Hlađenje jedino zahtijeva stabilizator napona za +5V (LM7805) zbog velike naponske razlike ulaz-izlaz.

9.2. MOGUĆE EKSTENZIJE UREĐAJA

Osnovna zamisao da se konstruiše potpun uređaj sa jasno definisanim funkcijama i namjenom je materijalizovana. Međutim, naravno da je riječ o jednostavnoj spravici koja bi mogla biti baza za razvoj nekog mnogo složenijeg i korisnijeg sklopa. Osim što bi se mogli ispraviti uočeni nedostaci, takođe bi se moglo razmisliti o uvođenju novih funkcija. Pretežno soſtverska implementacija, koja je i ovdje slučaj, omogućuje laku nadgradnju.

Od hardverskih promjena moglo bi se poraditi na ubrzavanju pozicioniranja. To bi se uradilo, recimo, uvođenjem step motora sa većim brojem koraka po krugu i manjim stepenom redukcije ili upotrebom regulisanog pogona sa DC motorom i inkrementalnim enkoderom kao davačem položaja. Tako bi se, naravno, finalni proizvod dodatno poskupio.

Još jedna od mogućih ekstenzija ovog uređaja pomenuta je ranije u radu. Riječ je o konstrukciji višedimenzionog pozicionera koji neće samo pozicionirati po azimutalnom uglu. Tu bi se mogla dodati mogućnost "pametnog" pozicioniranja, spregnutog sa jačinom primanog signala.

10 DOKUMENTACIJA

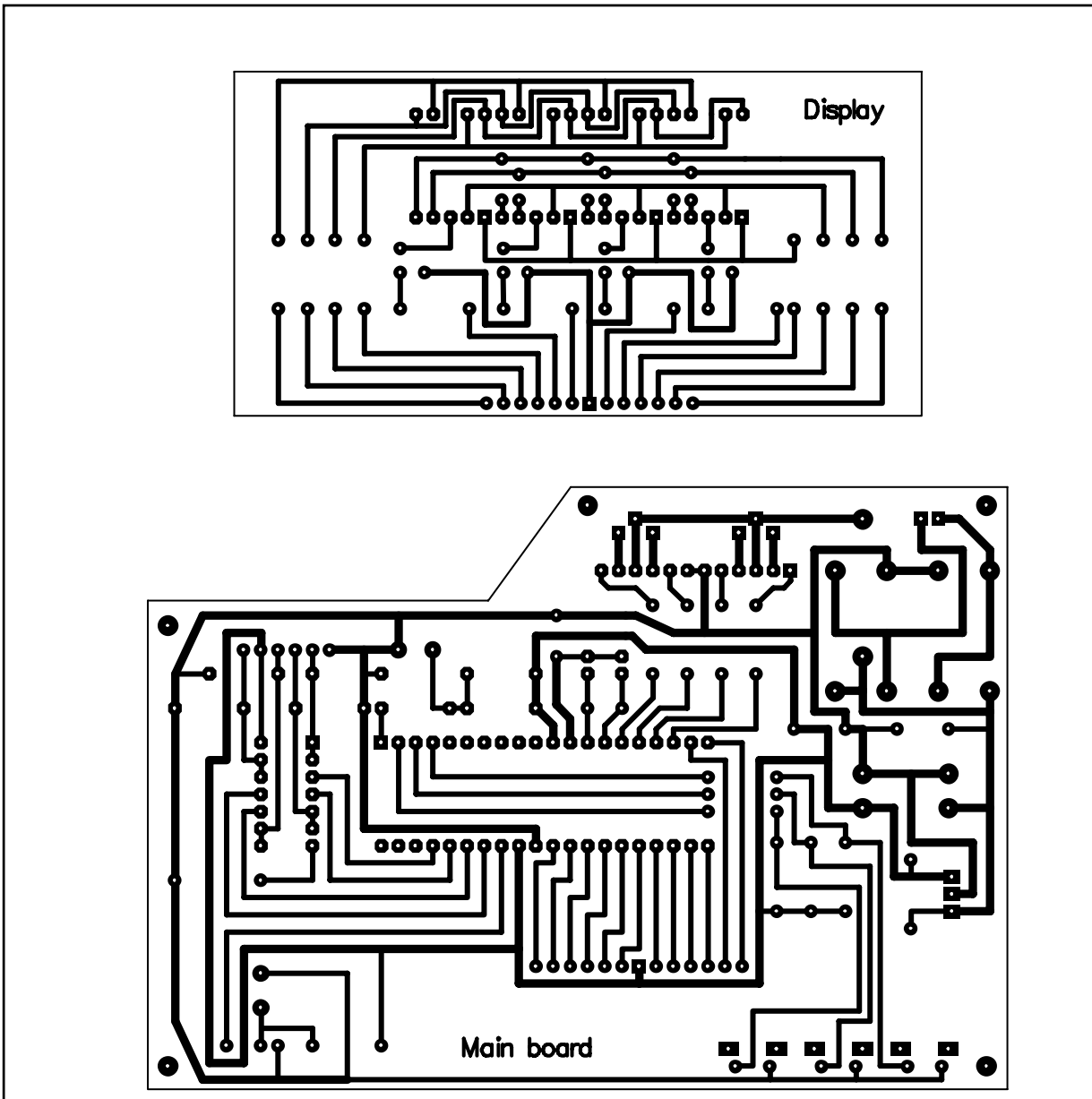
10.1. ŠTAMPANE PLOČE

Neizbježni dio procesa od ideje do finalne realizacije uređaja je i projektovanje štampanih ploča za elektronske komponente. Štampano kolo je podijeljeno na dva zasebna dijela. Većina komponenata (među njima i mikrokontroler) smiješta se na horizontalno montiranu matičnu ploču, a četiri displejske cifre zajedno sa svojim pobudnim elementima na njoj okomitu ploču displeja.

Alat koji je upotrijebljen u ovom slučaju je probna verzija programa Protel 98. Iako je riječ o dosta moćnom softveru, ploče su projektovane kao jednostavne i jednoslojne. Nije predviđeno korišćenje komponenti za površinsku montažu (SMD).

Slika 10.1 prikazuje ploče u prirodnoj veličini.

Slika 10.1: POGLED ODOZDO NA GLAVNU ŠTAMPANU PLOČU I PLOČU DISPLEJA



10.2. POPIS UPOTRIJEBLJENOG MATERIJALA

Komponenta	Vrijednost	Količina
Koračni motor	12V / 200 koraka po obrtaju	1
Energetski transformator	220V / 12V @50Hz	1
Mikrokontroler	PIC16F877-04/P	1
Integrisani IC prijemnik	SFH506-36	1
CMOS NAND Šmitov triger	CD4093B	1
Integrisani stabilizator za napon +5V	LM7805	1
Integrisani 4× bipolarni NPN darlington	SMA4033	1
Sedmosegmentni LED displej (zajednička anoda)	LTS546AP	4
Bipolarni PNP tranzistor	BC212B	4
Kristal	TQG 4MHz	1
Dioda	1N5408	4
Dioda	1N4003	1
Dioda	1N4148	1
Kondenzator	1000 µF	1
Kondenzator	470 µF	1
Kondenzator	10 µF	1
Kondenzator	4,7 µF	1
Kondenzator	0,1 µF	2
Kondenzator	22 pF	2
Otpornik	220 Ω	8
Otpornik	330 Ω	2
Otpornik	1,2 kΩ	4
Otpornik	1,5 kΩ	4
Otpornik	10 kΩ	4
Otpornik	100 kΩ	1

10.3. LISTING PROGRAMA

```
;*********************************************************************
;**                                                                 **
;**  program: t2                                                    **
;**                                                                 **
;**  opis:    Program za mikroprocesorski                           **
;**           daljinski upravljani                                  **
;**           pozicioner antene                                     **
;**                                                                 **
;**  autor:   Čedomir Zeljković                                     **
;**           cedomir@blic.net                                      **
;**                                                                 **
;*********************************************************************

#include "p16f877.inc"

;**  KONFIGURACIONA RIJEČ

        __CONFIG 0x3D35         ;_CP_OFF & _WDT_OFF & _PWRTE_ON &
                                ;_XT_OSC & _LVP_OFF & _WRT_ENABLE_OFF &
                                ;_CPD_OFF & _DEBUG_OFF & _BODEN_OFF

;**  DEFINICIJA MEMORIJSKIH LOKACIJA

#define RC_STAT         0x20
#define RC_BITCOUNT     0x21
#define RC_TMRVALUE     0x22
#define RC_DEADTIME     0x23
#define RC_HCOMMAND     0x24
#define RC_LCOMMAND     0x25
#define RC_TEST         0x26

#define DD_UNITS        0x28
#define DD_TENS         0x29
#define DD_HUNS         0x2A
#define DD_THOUS        0x2B
#define DD_DIGIT        0x2C
#define DD_SELECT       0x2D
#define DD_POINTER      0x2E

#define MP_STEPCOUNT    0x30
#define MP_SYNCCOUNTL   0x31
#define MP_SYNCCOUNTH   0x32
#define MP_PRESCALER    0x33

#define CHANNEL         0x40
#define ACTPOSL         0x41
#define ACTPOSH         0x42

#define REQPOSL         0x50
#define REQPOSH         0x51
#define REG0            0x52
#define REG1            0x53
#define REG2            0x54
#define REG3            0x55

#define UNITS           0x60
#define TENS            0x61
#define HUNS            0x62
#define THOUS           0x63
#define MODE            0x64
#define COMMAND         0x65
#define CURSOR          0x66
#define BLOCK_CNT       0x68
#define COMMANDCODE     0x69
#define DEVICECODE      0x6A
#define TOGGLECODE      0x6B
#define RC_OLDTOGGLE    0x6C

#define W_KEEP          0x70
#define STATUS_KEEP     0x71
#define PCLATH_KEEP     0x72
#define REQPOSL_TEMP    0x73
#define REQPOSH_TEMP    0x74
#define PORTCLATCH      0x75
```

```
;DEFINICIJA BITOVA U REGISTRIMA

;*****************************************************************************
;*      Registar RC_STAT
;*      Status prijemnika daljinske komande
;*****************************************************************************
#define LASTBIT         0       ;posljednji primljeni bit
#define RECEIVING       1       ;prijem u toku
#define TIMEMATCH       2       ;dužina mjerenog vremena odgovara
#define DEADTIME        3       ;mrtvo vrijeme u toku
#define NEWBIT          4       ;vrijednost novog bita
#define TEMP            5       ;privremeni registar bita
#define VCR             6       ;flag za ispravni prijem podatka
;*****************************************************************************

;*****************************************************************************
;*      Registar MODE
;*      Trenutni mod rada pozicionera
;*****************************************************************************
#define POWER           0       ;Uređaj uključen
#define KEYPRESS        1       ;Taster pritisnut
#define SET             2       ;Unošenje pozicije u toku
#define SYNCHRO         3       ;Sinhronizovanje u toku
;*****************************************************************************

;**  DEFINICIJA KONSTANTI

#define BLOCK_ADR       0x40
#define BLOCK_LENGTH    3

#define MIN_1T          0x64    ;decimalno 100
#define MAX_1T          0x7A    ;decimalno 122
#define MIN_15T         0x96    ;decimalno 150
#define MAX_15T         0xB7    ;decimalno 183
#define MIN_2T          0xC8    ;decimalno 200
#define MAX_2T          0xF4    ;decimalno 244

#define MAX_SYNCCOUNTH  0x0F    ;decimalno 15
#define MOTORPRESCALER  0x02    ;decimalno 2

;**  DEFINICIJA MAKROA

BANK0 MACRO
        bcf STATUS,RP0
        bcf STATUS,RP1
        ENDM
BANK1 MACRO
        bsf STATUS,RP0
        bcf STATUS,RP1
        ENDM
BANK2 MACRO
        bcf STATUS,RP0
        bsf STATUS,RP1
        ENDM
BANK3 MACRO
        bsf STATUS,RP0
        bsf STATUS,RP1
        ENDM
```

```
;**  START PROGRAMA

        org 0x0000
        clrf PCLATH             ;
        clrwdt                  ;
        goto init               ; skok na inicijalizacionu rutinu

;**  INTERAPT VEKTOR

        org 0x0004
        goto isr;               ; skok na prekidnu rutinu

;**  INICIJALIZACIONA RUTINA

        org 0x0005

init
        BANK0
        clrf PORTA              ;
        clrf PORTB              ;
        clrf PORTC              ;
        clrf PORTD              ;
        clrf PORTE              ;
        clrf PORTCLATCH         ;
        clrf MP_STEPCOUNT       ;
        clrf REQPOSL            ;
        clrf REQPOSH            ;

;*************************************************************************
;*      Inicijalizacija smijera I/O pinova:
;*************************************************************************
        BANK1
        movlw b'00000111'       ;RA0, RA1, RA2 tasterski ulazi
        movwf TRISA             ;
        movlw b'00011111'       ;RB0 ic-ulaz, RB4 ulaz od kraj. prekidača
        movwf TRISB             ;
        movlw b'00000000'       ;RC0:RC3 stepper drive, RC4:RC7 disp.drive
        movwf TRISC             ;
        movlw b'00000000'       ;RD0:RD7 izlazi za displejske segmente
        movwf TRISD             ;

;*************************************************************************
;*      Inicijalizacija registra OPTION_REG:
;*      ----------------------------------------------------------------
;*      - Onemogućeni pull-up otpornici
;*      - Tajmer 0 inicijalno isključen
;*      - Interapt RB0/INT se dešava na rastuću ivicu signala
;*      - Prescaler tajmera 0 podešen na 1:16
;*************************************************************************
        movlw b'11110011'       ;
        movwf OPTION_REG        ;

        BANK0
;*************************************************************************
;*      Inicijalizacija tajmera 1:
;*      ----------------------------------------------------------------
;*      - Tajmer 1 inicijalno uključen
;*      - Prescaler podešen na 1:1
;*      - Izvor takta je unutrašnji (Fosc/4)
;*************************************************************************
        movlw b'00000001'       ;
        movwf T1CON             ;

;*************************************************************************
;*      Inicijalizacija registra MODE:
;*      ----------------------------------------------------------------
;*      - POWER=0       - uređaj inicijalno isključen
;*      - KEYPRESS=0    - nijedan taster nije inicijalno pritisnut
;*      - SET=0         - uređaj nije već u set modu
;*      - SYNCHRO=0     - uređaj nije u modu sinhronizovanja
;*************************************************************************
        movlw b'00000000'       ;
        movwf MODE              ;
```

```
;**********************************************************************
;*      Pri uključenju uređaja na displeju treba biti ispisano
;*
;*                              | | | |-|
;*
;**********************************************************************
        movlw d'10'             ;kod za blanko znak
        movwf DD_THOUS          ;
        movwf DD_HUNS           ;
        movwf DD_TENS           ;
        movlw d'11'             ;kod za crticu
        movwf DD_UNITS          ;

        movlw b'11101111'       ;prva se ispisuje cifra hiljada => RC4=0
        movwf DD_SELECT         ;
        movlw DD_THOUS          ;adresa hiljada u pointer cifre
        movwf DD_POINTER        ;

;**********************************************************************
;*      Inicijalizacija registra INTCON:
;*      ---------------------------------------------------------------
;*      - Prekid INT za IC komandu je omogućen
;*      - Prekid RB za krajnji prekidač je onemogućen
;*      - Prekid tajmera 0 je omogućen
;*      - Postavljen je bit za globalno omogućenje prekida
;**********************************************************************
        movlw b'10110000'       ;
        movwf INTCON            ;

;**********************************************************************
;*      Inicijalizacija prescalera MP_PRESCALER:
;*      ---------------------------------------------------------------
;*      Prescaler se tovari vrijednošću konstante MOTORPRESCALER
;*      On određuje period koraka motora tako što je period umnožak
;*      vrijednosti prescalera i dužine perioda glavne petlje
;**********************************************************************
        movlw MOTORPRESCALER    ;
        movwf MP_PRESCALER      ;

;**********************************************************************
;*      Inicijalizacija registra RC_STAT:
;*      ---------------------------------------------------------------
;*      Svi flegovi su inicijalno resetovani osim bita DEADTIME.
;*      Time je omogućeno da po dovođenju napajanja na SFH506 ne dođe do
;*      lažnog interapta na mikrokontroleru.
;**********************************************************************
        movlw b'00001000'       ;
        movwf RC_STAT           ;

;**********************************************************************
;*      Inicijalizacija registra RC_DEADTIME:
;*      ---------------------------------------------------------------
;*      Početno mrtvo vrijeme za prijem komandi iznosi 250 perioda
;*      glavne petlje.
;*
;**********************************************************************
        movlw d'250'            ;
        movwf RC_DEADTIME       ;

;**********************************************************************
;*      Početak glavne petlje
;**********************************************************************
begin_1ms
        bcf PIR1,TMR1IF         ;reset zastavice prekida tajmera 1
        movlw d'252'            ;učitavanje u tajmer 1...
        movwf TMR1H             ;...vrijednosti od 64536...
        movlw d'24'             ;...što će omogućiti ciklus...
        movwf TMR1L             ;...u trajanju od 1ms pri Fosc=4 MHz
        clrwdt                  ;resetovanje watchdog tajmera
```

```
;*************************************************************************
;**                                                                     **
;**  DISPLAY UPDATE                                                     **
;**  Svako novo izvršenje ovog dijela koda bira sljedeću cifru,         **
;**  postavlja njenu vrijednost na port d i aktivira korespondentni     **
;**  tranzistor za napajanje displeja.                                  **
;**                                                                     **
;*************************************************************************
update_display
        movlw 0xFF              ;gašenje trenutne cifre
        movwf PORTD             ;...
        movf PORTCLATCH,w       ;učitaj temp leč porta C
        andlw b'00001111'       ;sačuvaj samo nibl za pobudu motora
        movwf PORTCLATCH        ;vrati u leč novo stanje
        movf DD_SELECT,w        ;učitaj displejske selekcione bite
        andlw b'11110000'       ;maskiraj niži nibl
        iorwf PORTCLATCH,w      ;spoji niblove za motor i displej
        movwf PORTC             ;izbaci na port C
        movwf PORTCLATCH        ;memoriši u temp leč porta C
        movf DD_POINTER,w       ;prebaci pointer trenutne cifre u...
        movwf FSR               ;...registar za ind. adresiranje
        movf INDF,w             ;vrijednost trenutne cifre u akumulator
        call get_7s_code        ;konverzija dec. vrijednosti u 7s kôd
        movwf PORTD             ;dovođenje 7s kôda na segmente
        rlf DD_SELECT,f         ;pomijeranje selekcionog registra
        btfss STATUS,C          ;da li je ispisana zadnja cifra (UNITS)?
        goto new_unit           ;da...priprema novog ciklusa
        decf DD_POINTER,f       ;ne...dekr. pointera trenutne cifre
        goto scan_keypad        ;izlaz
new_unit
        movlw b'11101111'       ;neka je 0 u sel. registru na mjestu...
        movwf DD_SELECT         ;...hiljada (prva cifra u nizu)
        movlw DD_THOUS          ;neka pointer trenutne cifre...
        movwf DD_POINTER        ;...pokazuje na adresu cifre hiljada
        goto scan_keypad        ;izlaz

;*************************************************************************
;**                                                                     **
;**  GET_7S_CODE                                                        **
;**  Podprogram koji binarnu vrijednost iz akumulatora pretvara u kod   **
;**  za 7S displej sa zajedničkom katodom.                              **
;**  Raspored na displeju: |a|b|g|f|e|d|dp|c|                           **
;**                                                                     **
;*************************************************************************
get_7s_code
        movwf DD_DIGIT          ;pričuvaj sadržaj iz w
        sublw d'14'             ;oduzmi max vrijednost za prikaz
        btfss STATUS,C          ;da li je došlo do prenosa
        retlw b'11111111'       ;da...vrati praznu vrijednost
        movf DD_DIGIT,w         ;ne...nastavi ispis korektne vrijednosti
        addwf PCL,f             ;dodaj sadržaj iz w programskom brojaču
        retlw b'00100010'       ;kod za '0'
        retlw b'10111110'       ;kod za '1'
        retlw b'00010011'       ;kod za '2'
        retlw b'00011010'       ;kod za '3'
        retlw b'10001110'       ;kod za '4'
        retlw b'01001010'       ;kod za '5'
        retlw b'01000010'       ;kod za '6'
        retlw b'00111110'       ;kod za '7'
        retlw b'00000010'       ;kod za '8'
        retlw b'00001010'       ;kod za '9'
        retlw b'11111111'       ;kod za ' '
        retlw b'11011111'       ;kod za '-'
        retlw b'10001010'       ;kod za 'Y'
        retlw b'00100110'       ;kod za 'N'
        retlw b'01100011'       ;kod za 'C'
```

```
;**********************************************************************
;**                                                                   **
;**  SCAN_KEYPAD                                                      **
;**  Rutina za skeniranje tastera spojenih na tri niža pina porta A.  **
;**  Ako je novi taster pritisnut, dalje se simulira kao da je        **
;**  odgovarajuća komanda došla od daljinskog upravljača              **
;**                                                                   **
;**********************************************************************
scan_keypad
        btfsc MODE,KEYPRESS     ;da li je bio pritisnut taster?
        goto unpress            ;da...da vidimo da li je sad
        movlw d'00'             ;pp da nijedan taster nije pritisnut
        btfss PORTA,0           ;da li je pritisnut taster Channel Down?
        movlw d'33'             ;da...njegov kod u akumulator
        btfss PORTA,1           ;da li je pritisnut taster Power?
        movlw d'12'             ;da...njegov kod u akumulator
        btfss PORTA,2           ;da li je pritisnut taster Channel Up?
        movlw d'32'             ;da...njegov kod u akumulator
        andlw b'11111111'       ;samo da se eventualno setuje bit Z
        btfsc STATUS,Z          ;da li je sad pritisnut ijedan taster?
        goto update_mot_pos     ;ne...izlaz
        movwf COMMANDCODE       ;ostavljanje koda komande u odg. registar
        bsf RC_STAT,VCR         ;simulacija da je komanda došla od daljinskog
        bsf MODE,KEYPRESS       ;postavljanje zastavice pritiska tastera
        goto update_mot_pos     ;izlaz
unpress
        comf PORTA,w            ;komplement PORTA u akumulator
        andlw b'00000111'       ;maskiranje nevažnih bita
        btfss STATUS,Z          ;da li je sad pritisnut ijedan taster?
        goto update_mot_pos     ;da...čekaj otpuštanje
        bcf MODE,KEYPRESS       ;ne...reset zastavice pritiska tastera

;**********************************************************************
;**                                                                   **
;**  UPDATE_MOT_POS                                                   **
;**  Svaki n-ti put (n=MP_PRESCALER) u okviru glavne petlje vrši se   **
;**  update pozicije step motora. Ako postoji razlika između trenutne **
;**  i tražene pozicije izvrši se korak motora ka njenom anuliranju.  **
;**                                                                   **
;**********************************************************************
update_mot_pos
        decfsz MP_PRESCALER     ;dec. brojač i vidi da li je dostigao nulu
        goto test_comm          ;ne...izlaz iz rutine
        movlw MOTORPRESCALER    ;učitaj konstantu prescalera
        movwf MP_PRESCALER      ;i njome napuni prescalerov registar
        btfss MODE,POWER        ;da li je uređaj uključen?
        goto stop_motor         ;ne...ugasi motor
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        goto sync_motor         ;da...rutina za sinhronizovanje
        movf ACTPOSH,w          ;učitaj u W viši bajt trenutne pozicije
        subwf REQPOSH,w         ;REQPOSH-ACTPOSH -> W
        btfss STATUS,C          ;da li je REQPOSH > ACTPOSH?
        goto step_back          ;ne...učini korak nazad
        btfss STATUS,Z          ;da li je REQPOSH = ACTPOSH?
        goto step_forward       ;ne...učini korak naprijed
        movf ACTPOSL,w          ;učitaj u W niži bajt trenutne pozicije
        subwf REQPOSL,w         ;REQPOSL-ACTPOSL -> W
        btfss STATUS,C          ;da li je REQPOSH > ACTPOSH?
        goto step_back          ;ne...učini korak nazad
        btfss STATUS,Z          ;da li je REQPOSL = ACTPOSL?
        goto step_forward       ;ne...učini korak naprijed
stop_motor
        bcf PORTCLATCH,3        ;--0----/\/\/\---|<
        bcf PORTCLATCH,2        ;--0----/\/\/\---|<
        bcf PORTCLATCH,1        ;--0----/\/\/\---|<
        bcf PORTCLATCH,0        ;--0----/\/\/\---|<
        movf PORTCLATCH,w       ;učitaj pripremljenu vrijednost...
        movwf PORTC             ;...i izbaci na PORT C
        goto test_comm          ;ne...izlaz iz rutine

step_forward
        incf MP_STEPCOUNT,f     ;povećanje brojača koraka
        movlw b'11110000'       ;učitaj masku
        andwf PORTCLATCH,f      ;maskiraj niži nibl
        movf MP_STEPCOUNT,w     ;učitaj brojač koraka
        andlw b'00000011'       ;ostavi samo dva najlakša bita
        call get_step_code      ;uzmi odgovarajući pobudni nibl
        iorwf PORTCLATCH,f      ;ubaci u niži nibl za PORT C
```

```
        incf ACTPOSL,f          ;inkrementiraj niži bajt tren.pozicije
        btfsc STATUS,Z          ;da li je došlo do prenosa?
        incf ACTPOSH,f          ;da...inkrementiraj i viši bajt
        goto test_comm          ;ne...izlaz iz rutine

step_back
        decf MP_STEPCOUNT,f     ;povećanje brojača koraka
        movlw b'11110000'       ;učitaj masku
        andwf PORTCLATCH,f      ;maskiraj niži nibl
        movf MP_STEPCOUNT,w     ;učitaj brojač koraka
        andlw b'00000011'       ;ostavi samo dva najlakša bita
        call get_step_code      ;uzmi odgovarajući pobudni nibl
        iorwf PORTCLATCH,f      ;ubaci u niži nibl za PORT C
        decf ACTPOSL,f          ;dekrementiraj niži bajt tren.pozicije
        comf ACTPOSL,w          ;komplement od ACTPOSL -> W
        btfsc STATUS,Z          ;da li je potrebna pozajmica?
        decf ACTPOSH,f          ;da...dekrementiraj i viši bajt
        goto test_comm          ;ne...izlaz iz rutine

get_step_code
        addwf PCL,f             ;redni br. koraka je ofset za PC
        retlw b'00000101'       ;pobuda tranzistora u prvom koraku
        retlw b'00001001'       ;pobuda tranzistora u drugom koraku
        retlw b'00001010'       ;pobuda tranzistora u trećem koraku
        retlw b'00000110'       ;pobuda tranzistora u četvrtom koraku

sync_motor
        incf MP_STEPCOUNT,f     ;povećanje brojača koraka
        movlw b'11110000'       ;učitaj masku
        andwf PORTCLATCH,f      ;maskiraj niži nibl
        movf MP_STEPCOUNT,w     ;učitaj brojač koraka
        andlw b'00000011'       ;ostavi samo dva najlakša bita
        call get_step_code      ;uzmi odgovarajući pobudni nibl
        iorwf PORTCLATCH,f      ;ubaci u niži nibl za PORT C
        incf ACTPOSL,f          ;inkrementiraj niži bajt tren.pozicije
        btfsc STATUS,Z          ;da li je došlo do prenosa?
        incf ACTPOSH,f          ;da...inkrementiraj i viši bajt
        incf MP_SYNCCOUNTL,f    ;povećaj niži bajt sinhro brojača
        btfsc STATUS,Z          ;da li ima prenosa?
        incf MP_SYNCCOUNTH,f    ;da...povećaj i viši bajt sinhro brojača
        movf MP_SYNCCOUNTH,w    ;učitaj vrijednost višeg bajta s. brojača
        sublw MAX_SYNCCOUNTH    ;poredi sa max dozvoljenom vrijednošću
        btfsc STATUS,C          ;da li je došlo do prekoračenja?
        goto test_comm          ;ne...izlaz iz rutine
finish_sync
        bcf MODE,SYNCHRO        ;poništi sinhro mod
        bcf INTCON,RBIE         ;onemogući nove prekide krajnjeg prekidača
        movf CHANNEL,w          ;učitaj broj trenutnog kanala
        movwf COMMANDCODE       ;prebaci u kôd primljene komande...
        bsf RC_STAT,VCR         ;...i simuliraj da je došla od daljinskog

test_comm
        btfsc RC_STAT,VCR       ;ne...da li je primljena nova komanda?
        goto proc_comm          ;da...obradi komandu
dead_key
        btfsc RC_STAT,DEADTIME  ;da li smo u mrtvom vremenu?
        goto dead_wait          ;da...
loop_1ms
        btfss PIR1,TMR1IF       ;da li je istekao ciklus od 1ms?
        goto loop_1ms           ;ne...čekaj još
        goto begin_1ms          ;da...započni novi ciklus

dead_wait
        decfsz RC_DEADTIME,f    ;dekrementiraj brojač mrtvog vremena
        goto loop_1ms           ;još nije isteklo mrtvo vrijeme, pa čekaj još
        bcf RC_STAT,DEADTIME    ;vrijeme je isteklo, pa resetuj zastavicu
        bsf INTCON,INTE         ;omogući nove interapte IC komande
        goto loop_1ms           ;

;*************************************************************************
;**                                                                     **
;**  RUTINA ZA OBRADU PRIMLJENE KOMANDE                                 **
;**                                                                     **
;*************************************************************************
proc_comm
        movf COMMANDCODE,w      ;učitaj kod komande
        movwf COMMAND           ;...i memoriši u registar komande
        btfsc MODE,POWER        ;da li je uređaj uključen
```

```
        goto do_proc_comm       ;da...bezuslovno izvrši komandu
        sublw d'12'             ;ne...
        btfss STATUS,Z          ;da li je tražena komanda POWER?
        goto end_proc_comm      ;ne...ignoriši komandu
        movlw d'12'             ;da...vrati u W kod za komandu POWER
do_proc_comm
        bsf PCLATH,3            ;postavljanje page bita za page1
        bcf PCLATH,4            ;...
        call exec_comm          ;poziv rutini za izvršenje komande
        bcf PCLATH,3            ;vraćanje page bita za page0
        bcf PCLATH,4            ;...
end_proc_comm
        bcf RC_STAT,VCR         ;komanda izvršena => poništi zastavicu VCR
        goto loop_1ms           ;


;*************************************************************************
;**                                                                     **
;**  INTERAPT RUTINA                                                    **
;**                                                                     **
;*************************************************************************
isr     movwf W_KEEP            ;čuvanje sadržaja iz akumulatora
        swapf STATUS,w          ;čuvanje sadržaja statusnog bajta
        clrf STATUS
        movwf STATUS_KEEP       ;...
        movf PCLATH,w           ;čuvanje sadržaja registra PCLATH
        movwf PCLATH_KEEP       ;...
        clrf PCLATH
        btfsc INTCON,RBIF       ;ne...da li je tražen interapt porta B?
        goto rb_isr             ;da...obrađuj
        btfsc INTCON,INTF       ;ne...da li je tražen RB0/INT?
        goto int_isr            ;da...obrađuj
        btfsc INTCON,T0IF       ;da li TMR0 izaziva interapt?
        goto tmr0_isr           ;da...obrađuj
finish_int
        movf PCLATH_KEEP,w      ;vraćanje registra PCLATH
        movwf PCLATH            ;
        swapf STATUS_KEEP,w     ;vraćanje statusnog bajta
        movwf STATUS            ;
        swapf W_KEEP,f          ;vraćanje sadržaja akumulatora
        swapf W_KEEP,w          ;...
        retfie                  ;

;*************************************************************************
;**                                                                     **
;**  RUTINA ZA OBRADU PREKIDA TAJMERA TMR0                              **
;**                                                                     **
;*************************************************************************
tmr0_isr
        bcf INTCON,T0IF         ;reset zastavice prekida tajmera 0
        goto rec_error          ;nije smjelo doći do overflow-a

;*************************************************************************
;**                                                                     **
;**  RUTINA KOJA SE POZIVA PO PRIJEMU RASTUĆE IVICE NA RB0/INT          **
;**  Signal koji dolazi na pin RB0/INT proizvod je dekodovanja          **
;**  IC snopa iz RC5 daljinskog upravljača.                             **
;**                                                                     **
;*************************************************************************
int_isr
        bcf INTCON,INTF         ;reset zastavice prekida INT
        btfss RC_STAT,RECEIVING ;da li je prijem komande u toku?
        goto rec_init           ;ne...inicijalizacija novog prijema
        movf TMR0,w             ;da...pročitaj vrijednost tajmera
        movwf RC_TMRVALUE       ;...i pohrani u registar tajmera
        clrf TMR0               ;reset tajmera za novo mjerenje
        call test_1t            ;rutina za provjeru dužine perioda od Tp
        btfsc RC_STAT,TIMEMATCH ;da li je period u intervalu oko Tp?
        goto rec_1t             ;da...prijem bita
        call test_15t           ;rutina za provjeru dužine perioda od 1,5Tp
        btfsc RC_STAT,TIMEMATCH ;da li je period u intervalu oko 1,5Tp?
        goto rec_15t            ;da...prijem bita (bitova)
        call test_2t            ;rutina za provjeru dužine perioda od 2Tp
        btfsc RC_STAT,TIMEMATCH ;da li je period u intervalu oko 2Tp?
        goto rec_2t             ;da...prijem bitova
rec_error
start_deadtime
        bcf RC_STAT,RECEIVING   ;izlazak iz moda prijema podataka
```

```
        bsf RC_STAT,DEADTIME    ;postavljanje zastavice mrtvog vremena
        bcf INTCON,INTE         ;onemogućenje prekida u mrtvom vremenu
        movlw d'60'             ;
        movwf RC_DEADTIME       ;punjenje brojača mrtvog vremena (60 T)
        BANK1
        bsf OPTION_REG,T0CS     ;zaustavljanje tajmera TMR0
        BANK0
        goto finish_int         ;
rec_init
        clrf TMR0               ;inicijalizacija tajmera
        BANK1
        bcf OPTION_REG,T0CS     ;startovanje tajmera TMR0
        BANK0
        bsf RC_STAT,RECEIVING   ;označi početak prijema nove komande
        movlw d'13'             ;
        movwf RC_BITCOUNT       ;priprema brojača bita
        bsf RC_STAT,LASTBIT     ;primljena je jedinica (prvi start bit)
        goto finish_int         ;

test_1t
        bcf RC_STAT,TIMEMATCH   ;greška - dok se ne dokaže suprotno :-)
        movf RC_TMRVALUE,w      ;
        sublw MIN_1T            ;
        btfsc STATUS,C          ;da li je period kraći od minimalnog?
        return                  ;da...izlaz
        movf RC_TMRVALUE,w      ;ne...dalje
        sublw MAX_1T            ;
        btfss STATUS,C          ;da li je period duži od maksimalnog?
        return                  ;da...izlaz
        bsf RC_STAT,TIMEMATCH   ;ne...period je u intervalu oko Tp
        return                  ;

test_15t
        bcf RC_STAT,TIMEMATCH   ;greška - dok se ne dokaže suprotno
        movf RC_TMRVALUE,w      ;
        sublw MIN_15T           ;
        btfsc STATUS,C          ;da li je period kraći od minimalnog?
        return                  ;da...izlaz
        movf RC_TMRVALUE,w      ;ne...dalje
        sublw MAX_15T           ;
        btfss STATUS,C          ;da li je period duži od maksimalnog?
        return                  ;da...izlaz
        bsf RC_STAT,TIMEMATCH   ;ne...period je u intervalu oko Tp
        return                  ;

test_2t
        bcf RC_STAT,TIMEMATCH   ;greška - dok se ne dokaže suprotno
        movf RC_TMRVALUE,w      ;
        sublw MIN_2T            ;
        btfsc STATUS,C          ;da li je period kraći od minimalnog?
        return                  ;da...izlaz
        movf RC_TMRVALUE,w      ;ne...dalje
        sublw MAX_2T            ;
        btfss STATUS,C          ;da li je period duži od maksimalnog?
        return                  ;da...izlaz
        bsf RC_STAT,TIMEMATCH   ;ne...period je u intervalu oko Tp
        return                  ;

rec_1t
        bsf RC_STAT,NEWBIT      ;predpostavka da je novi bit jedinica
        btfss RC_STAT,LASTBIT   ;da li je prošli bit jedinica?
        bcf RC_STAT,NEWBIT      ;ne...nije onda ni novi
        call add_new_bit        ;rutina koja memoriše novi bit
        goto rec_continue       ;

rec_15t
        bsf RC_STAT,TEMP        ;čuvanje posljednjeg bita u TEMP registru
        btfss RC_STAT,LASTBIT   ;...
        bcf RC_STAT,TEMP        ;...
        bsf RC_STAT,NEWBIT      ;prvi bit je svakako jedinica
        call add_new_bit        ;rutina koja memoriše novi bit
        bcf RC_STAT,NEWBIT      ;
        btfsc RC_STAT,TEMP      ;da li je prošli bit jedinica
        call add_new_bit        ;da...memoriši još jednu nulu
        goto rec_continue       ;

rec_2t
        bsf RC_STAT,NEWBIT      ;prvo je stigla jedinica
```

```
        call add_new_bit        ;rutina koja memoriše novi bit
        bcf RC_STAT,NEWBIT      ;...a potom i nula
        call add_new_bit        ;rutina koja memoriše novi bit
        goto rec_continue       ;

rec_continue
        movf RC_BITCOUNT,f      ;nop koji samo djeluje na Z statusni bit
        btfss STATUS,Z          ;da li je primljen zadnji bit?
        goto finish_int         ;ne...
        btfss RC_HCOMMAND,6     ;da...da li je drugi start bit "1"?
        goto rec_error          ;ne...proglasi grešku
        clrf TOGGLECODE         ;izdvajanje toggle bita
        btfsc RC_HCOMMAND,5     ;...
        bsf TOGGLECODE,0        ;...
        movf RC_HCOMMAND,w      ;izdvajanje koda uređaja
        andlw b'00011111'       ;...
        movwf DEVICECODE        ;...
        movf RC_LCOMMAND,w      ;izdvajanje koda uređaja
        andlw b'11111100'       ;...
        movwf COMMANDCODE       ;...
        bcf STATUS,C            ;...
        rrf COMMANDCODE,f       ;...
        bcf STATUS,C            ;...
        rrf COMMANDCODE,f       ;...
        movf TOGGLECODE,w       ;učitaj vrijednost toggle bita
        addwf RC_OLDTOGGLE,w    ;...i dodaj staru vrijednost
        andlw b'00000001'       ;poništi sve bite osim nultog
        btfsc STATUS,Z          ;da li je praktično novi toggle različit?
        goto rec_error          ;ne...ignoriši pridošlu komandu
        movf TOGGLECODE,w       ;učitaj vrijednost toggle bita
        movwf RC_OLDTOGGLE      ;novi toggle je stari za sljedecu komandu
        bsf RC_STAT,VCR         ;postavljanje zastavice ispravnog prijema
        goto start_deadtime     ;rutina starta mrtvog vremena

;*************************************************************************
;**                                                                     **
;**  PODPROGRAM ZA PRIJEM BITA IC KOMANDE                               **
;**  Bit se prima sa pina PORTB.0 (RB0/INT).                            **
;**  Komanda je 14-bitna (2 start bita + 12 bita podatka), pa su        **
;**  potrebna dva osmobitna registra za njeno memorisanje:              **
;**  RC_HCOMMAND i RC_LCOMMAND.                                         **
;**                                                                     **
;*************************************************************************
add_new_bit
        rlf RC_HCOMMAND,f       ;rotiranje višeg registra komande ulijevo
        bcf RC_HCOMMAND,0       ;
        rlf RC_LCOMMAND,f       ;rotiranje nižeg registra komande ulijevo
        btfsc STATUS,C          ;
        bsf RC_HCOMMAND,0       ;
        bcf RC_LCOMMAND,2       ;
        btfsc RC_STAT,NEWBIT    ;
        bsf RC_LCOMMAND,2       ;
        bcf RC_STAT,LASTBIT     ;NEWBIT je LASTBIT za sljedeći ciklus
        btfsc RC_STAT,NEWBIT    ;
        bsf RC_STAT,LASTBIT     ;
        decf RC_BITCOUNT,f      ;
        return                  ;

;*************************************************************************
;**                                                                     **
;**  RUTINA ZA OBRADU PREKIDA PORTA B                                   **
;**  Poziva se kada se rastave kontakti krajnjeg prekidača              **
;**                                                                     **
;*************************************************************************
rb_isr
        bcf INTCON,RBIF         ;reset zastavice prekida porta B
        btfss PORTB,4           ;da li je riječ o rastućoj ivici signala?
        goto finish_int         ;ne...završi prekid
        bcf INTCON,RBIE         ;onemogući nove prekide
        movlw d'16'             ;low_byte(3600)=16
        movwf ACTPOSL           ;punjenje nižeg registra trenutne pozicije
        movlw d'14'             ;hi_byte(3600)=14
        movwf ACTPOSH           ;punjenje višeg registra trenutne pozicije
        bcf MODE,SYNCHRO        ;poništi sinhro mod
        movf CHANNEL,w          ;učitaj broj trenutnog kanala
        movwf COMMANDCODE       ;prebaci u kôd primljene komande...
        bsf RC_STAT,VCR         ;...i simuliraj da je došla od daljinskog
        goto finish_int         ;završi prekid
```

```
;*************************************************************************
;**                                                                     **
;**  RUTINA ZA IZVRŠENJE KOMANDE TASTATURE ILI DALJINSKOG UPRAVLJAČA    **
;**  Kôd komande koji leži u opsegu 0-63 smješten je u akumulatoru.     **
;**  On se kao ofset dodaje programskom brojaču što stvara skok         **
;**  na odgovarajuću proceduru obrade komande.                          **
;**                                                                     **
;*************************************************************************
        org 0x800
exec_comm
        addwf PCL,f             ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_digit         ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_power         ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_cnlup         ;
        goto exec_cnldown       ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_sync          ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_go            ;
        goto exec_set           ;
        goto exec_mem           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
        goto exec_nop           ;
```

```
;**********************************************************************
;**                                                                   **
;**  RUTINA ZA OBRADU KOMANDE TASTERA CIFRE                           **
;**                                                                   **
;**********************************************************************
exec_digit
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        return                  ;da...nema modifikacije
        btfss MODE,SET          ;da li je uređaj u SET modu?
        goto set_channel        ;ne...prebaci na traženi kanal
        movf CURSOR,w           ;da...ispitivanje na kojoj cifri je kursor
        andlw b'00000011'       ;
        btfsc STATUS,Z          ;da li je kursor na cifri jedinica?
        goto erase              ;da...rutina za brisanje teže tri cifre
        movf DD_HUNS,w          ;ne...shift cifara ulijevo
        movwf DD_THOUS          ;
        movf DD_TENS,w          ;
        movwf DD_HUNS           ;
        movf DD_UNITS,w         ;
        movwf DD_TENS           ;
        incf CURSOR,f           ;
        goto end_digit          ;
erase   movlw d'10'             ;kod za blanko znak
        movwf DD_THOUS          ;
        movwf DD_HUNS           ;
        movwf DD_TENS           ;
        incf CURSOR,f           ;
        goto end_digit          ;
set_channel
        bcf STATUS,C            ;očisti bit prenosa pred rotaciju
        rlf COMMAND,w           ;2xBrojKanala -> W
        BANK2
        movwf EEADR             ;
        bcf STATUS,C            ;očisti bit prenosa pred rotaciju
        rlf EEADR,f             ;ponovo množenje sa 2 (svaka 4. adresa)
        bsf STATUS,RP0          ;Bank 3
        bcf EECON1,EEPGD        ;potvrda da se čita iz DATA memorije
        bsf EECON1,RD           ;iniciranje čitanja iz EEPROMa
        bcf STATUS,RP0          ;Bank 2
        movf EEDATA,w           ;W = EEDATA
        bcf STATUS,RP1          ;Bank 0
        movwf UNITS             ;
        movwf DD_UNITS          ;privremeno, do konverzije u REQPOS
        bsf STATUS,RP1          ;Bank 2
        incf EEADR,f            ;sljedeća adresa je za desetice
        bsf STATUS,RP0          ;Bank 3
        bcf EECON1,EEPGD        ;potvrda da se čita iz DATA memorije
        bsf EECON1,RD           ;iniciranje čitanja iz EEPROMa
        bcf STATUS,RP0          ;Bank 2
        movf EEDATA,w           ;W = EEDATA
        bcf STATUS,RP1          ;Bank 0
        movwf TENS              ;
        movwf DD_TENS           ;privremeno, do konverzije u REQPOS
        bsf STATUS,RP1          ;Bank 2
        incf EEADR,f            ;sljedeća adresa je za stotice
        bsf STATUS,RP0          ;Bank 3
        bcf EECON1,EEPGD        ;potvrda da se čita iz DATA memorije
        bsf EECON1,RD           ;iniciranje čitanja iz EEPROMa
        bcf STATUS,RP0          ;Bank 2
        movf EEDATA,w           ;W = EEDATA
        bcf STATUS,RP1          ;Bank 0
        movwf HUNS              ;
        movwf DD_HUNS           ;privremeno, do konverzije u REQPOS
        bsf STATUS,RP1          ;Bank 2
        incf EEADR,f            ;sljedeća adresa je za hiljadice
        bsf STATUS,RP0          ;Bank 3
        bcf EECON1,EEPGD        ;potvrda da se čita iz DATA memorije
        bsf EECON1,RD           ;iniciranje čitanja iz EEPROMa
        bcf STATUS,RP0          ;Bank 2
        movf EEDATA,w           ;W = EEDATA
        bcf STATUS,RP1          ;Bank 0
        movwf THOUS             ;
        movwf DD_THOUS          ;privremeno, do konverzije u REQPOS
        clrf REQPOSL            ;
        clrf REQPOSH            ;
        call take_reqpos        ;DD_*** -> REQPOS
        movlw d'10'             ;kod za blanko znak
        movwf DD_THOUS          ;
```

```
        movwf DD_HUNS           ;
        movwf DD_TENS           ;
        movf COMMAND,w          ;i zvanično postavi kanal na traženi
        movwf CHANNEL           ;
end_digit
        movf COMMAND,w          ;
        movwf DD_UNITS          ;
        return                  ;

;*************************************************************************
;**                                                                     **
;**  RUTINA ZA OBRADU KOMANDE CHANNEL DOWN                              **
;**  Dekrementiranje broja kanala                                       **
;**                                                                     **
;*************************************************************************
exec_cnldown
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        return                  ;da...nema modifikacije
        bcf MODE,SET            ;poništi SET mode
        decf CHANNEL,w          ;
        sublw d'255'            ;
        btfss STATUS,Z          ;da li se dekrementira sa kanala 0
        sublw d'246'            ;ne...
        addlw d'09'             ;da...
        movwf COMMAND           ;
        goto exec_digit         ;

;*************************************************************************
;**                                                                     **
;**  RUTINA ZA OBRADU KOMANDE CHANNEL UP                                **
;**  Inkrementiranje broja kanala                                       **
;**                                                                     **
;*************************************************************************
exec_cnlup
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        return                  ;da...nema modifikacije
        bcf MODE,SET            ;poništi SET mode
        incf CHANNEL,w          ;
        sublw d'10'             ;
        btfss STATUS,Z          ;da li se inkrementira sa kanala 9
        sublw d'10'             ;ne...
        movwf COMMAND           ;da...
        goto exec_digit         ;

;*************************************************************************
;**                                                                     **
;**  RUTINA ZA OBRADU KOMANDE POWER                                     **
;**  Uključenje/isključenje uređaja                                     **
;**                                                                     **
;*************************************************************************
exec_power
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        return                  ;da...nema modifikacije
        btfsc MODE,SET          ;da li je uređaj u SET modu?
        return                  ;da...nema modifikacije
        btfsc MODE,POWER        ;da li je uređaj uključen?
        goto poweroff           ;da...skok na rutinu za isključenje
        movlw BLOCK_LENGTH      ;učitaj dužinu bloka memorije...
        movwf BLOCK_CNT         ;... i stavi u brojač mem. lokacija
        movlw BLOCK_ADR         ;učitaj početnu adresu bloka memorije
        movwf FSR               ;... i stavi u registar za ind.adresiranje
        BANK2
        movwf EEADR             ;...te stavi u registar EEPROM adrese
block_read
        BANK3
        bcf EECON1,EEPGD        ;potvrda da se čita iz DATA memorije
        bsf EECON1,RD           ;iniciranje čitanja iz EEPROMa
        bcf STATUS,RP0          ;Bank 2
        movf EEDATA,w           ;W = EEDATA
        bcf STATUS,RP1          ;Bank 0
        movwf INDF              ;
        bsf STATUS,RP1          ;Bank 2
        incf EEADR,f            ;inkrementiranje pokazivača EEADR
        incf FSR,f              ;inkrementiranje pokazivača FSR
        bcf STATUS,RP1          ;Bank 0
        decfsz BLOCK_CNT,f      ;dekr. brojača lokacija. Da li je on nula?
        goto block_read         ;ne...čitaj sljedeću lokaciju
        movf CHANNEL,w          ;učitaj broj željenog kanala
```

```
        movwf COMMAND           ;stavi u registar komande
        bsf MODE,POWER          ;postavljanje zastavice uključenosti
        goto exec_digit         ;simuliraj dolazak komande za prebacivanje
                                ;na željeni kanal
poweroff
        movlw BLOCK_LENGTH      ;učitaj dužinu bloka memorije...
        movwf BLOCK_CNT         ;... i stavi u brojač mem. lokacija
        movlw BLOCK_ADR         ;učitaj početnu adresu bloka memorije
        movwf FSR               ;... i stavi u registar za ind.adresiranje
        BANK2
        movwf EEADR             ;...te stavi u registar EEPROM adrese
block_write
        BANK2
        movf INDF,w             ;podatak sa ind.adrese u W
        movwf EEDATA            ;...a potom u registar podatka EEPROMa
        bsf STATUS,RP0          ;bank 3
        bcf EECON1,EEPGD        ;potvrda da se piše u DATA memoriju
        bsf EECON1,WREN         ;omogućenje upisa
        bcf INTCON,GIE          ;onemogućenje prekida
        movlw 0x55              ;
        movwf EECON2            ;upisivanje sigurnosnog koda 0x55
        movlw 0xAA              ;
        movwf EECON2            ;upisivanje sigurnosnog koda 0xAA
        bsf EECON1,WR           ;iniciranje upisa u EEPROM
        BANK0
write_wait
        btfss PIR2,EEIF         ;da li je završen proces upisa u EEPROM
        goto write_wait         ;ne...čekaj još
        bcf PIR2,EEIF           ;da...resetuj zastavicu prekida
        bsf INTCON,GIE          ;omogući nove prekide
        BANK3
        bcf EECON1,WREN         ;onemogućenje novog upisa
        bcf STATUS,RP0          ;bank2
        incf EEADR,f            ;inkrementiranje pokazivača EEADR
        incf FSR,f              ;inkrementiranje pokazivača FSR
        bcf STATUS,RP1          ;Bank 0
        decfsz BLOCK_CNT,f      ;dekr. brojača lokacija. Da li je on nula?
        goto block_write        ;ne...piši na sljedeću lokaciju
        movlw d'10'             ;kod za blanko znak
        movwf DD_THOUS          ;
        movwf DD_HUNS           ;
        movwf DD_TENS           ;
        movlw d'11'             ;kod za crticu
        movwf DD_UNITS          ;
        bcf MODE,POWER          ;reset zastavice uključenosti
        return

;*************************************************************************
;**                                                                     **
;**  RUTINA ZA OBRADU KOMANDE SET                                       **
;**  Uključenje/isključenje moda za unos nove pozicije antene           **
;**                                                                     **
;*************************************************************************
exec_set
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        return                  ;da...nema modifikacije
        btfsc MODE,SET          ;da li je uređaj već u SET modu?
        goto unset              ;da...rutina za izlazak iz SET moda
        movf UNITS,w            ;
        movwf DD_UNITS          ;
        movf TENS,w             ;
        movwf DD_TENS           ;
        movf HUNS,w             ;
        movwf DD_HUNS           ;
        movf THOUS,w            ;
        movwf DD_THOUS          ;
        bsf MODE,SET            ;
        clrf CURSOR             ;
        return
unset
        bcf MODE,SET            ;
        movf CHANNEL,w          ;
        movwf COMMAND           ;
        goto exec_digit         ;
```

```
;**********************************************************************
;**                                                                   **
;**  RUTINA ZA OBRADU KOMANDE SYNC                                    **
;**  Iniciranje sinhronizovanja antene sa referentnim položajem       **
;**                                                                   **
;**********************************************************************
exec_sync
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        return                  ;da...nema modifikacije
        btfsc MODE,SET          ;da li je uređaj u SET modu?
        return                  ;da...nema modifikacije
        bsf MODE,SYNCHRO        ;proglasi sinhro mod
        bcf INTCON,RBIF         ;očisti zastavicu prekida porta B
        bsf INTCON,RBIE         ;omogući prekid krajnjeg prekidača
        clrf MP_SYNCCOUNTL      ;reset nižeg bajta sinhro brojača
        clrf MP_SYNCCOUNTH      ;reset višeg bajta sinhro brojača
        movlw d'05'             ;kôd za S (faktički 5)
        movwf DD_THOUS          ;... na 1. mjestu displeja
        movlw d'12'             ;kôd za Y
        movwf DD_HUNS           ;... na 2. mjestu displeja
        movlw d'13'             ;kôd za N
        movwf DD_TENS           ;... na 3. mjestu displeja
        movlw d'14'             ;kôd za C
        movwf DD_UNITS          ;... na 4. mjestu displeja
        return

;**********************************************************************
;**                                                                   **
;**  RUTINA ZA OBRADU KOMANDE GO                                      **
;**  Iniciranje rotacije antene ka željenoj poziciji                  **
;**                                                                   **
;**********************************************************************
exec_go
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        return                  ;da...nema modifikacije
        btfss MODE,SET          ;da li je uređaj u SET modu?
        return                  ;ne...povratak iz rutine
        call take_reqpos        ;poziv rutini za uzimanje tražene pozicije
        return                  ;povratak u glavnu petlju

;**********************************************************************
;**                                                                   **
;**  RUTINA ZA OBRADU KOMANDE MEM                                     **
;**  Memorisanje željene pozicije na trenutni kanal i                 **
;**  iniciranje rotacije ka njoj                                      **
;**                                                                   **
;**********************************************************************
exec_mem
        btfsc MODE,SYNCHRO      ;da li je u toku sinhronizovanje?
        return                  ;da...nema modifikacije
        btfss MODE,SET          ;da li je uređaj u SET modu?
        return                  ;ne...povratak iz rutine
        call take_reqpos        ;poziv rutini za uzimanje tražene pozicije
        andlw 0xFF              ;
        btfsc STATUS,Z          ;da li je uspjesno uzeta traž. pozicija
        return                  ;ne...povratak iz rutine
        movlw d'04'             ;učitaj dužinu bloka memorije...
        movwf BLOCK_CNT         ;... i stavi u brojač mem. lokacija
        movlw DD_UNITS          ;učitaj početnu adresu bloka memorije
        movwf FSR               ;... i stavi u registar za ind.adresiranje
        movf CHANNEL,w          ;učitaj broj trenutnog kanala
        BANK2
        movwf EEADR             ;...i stavi u registar EEPROM adrese
        bcf STATUS,C            ;očisti bit prenosa pred rotaciju
        rlf EEADR,f             ;EEADR *= 2
        bcf STATUS,C            ;očisti bit prenosa pred rotaciju
        rlf EEADR,f             ;EEADR *= 2
pos_write
        BANK2
        movf INDF,w             ;podatak sa ind.adrese u W
        movwf EEDATA            ;...a potom u registar podatka EEPROMa
        bsf STATUS,RP0          ;bank 3
        bcf EECON1,EEPGD        ;potvrda da se piše u DATA memoriju
        bsf EECON1,WREN         ;omogućenje upisa
        bcf INTCON,GIE          ;onemogućenje prekida
        movlw 0x55              ;
        movwf EECON2            ;upisivanje sigurnosnog koda 0x55
        movlw 0xAA              ;
```

```
        movwf EECON2            ;upisivanje sigurnosnog koda 0xAA
        bsf EECON1,WR           ;iniciranje upisa u EEPROM
        BANK0
pos_write_wait
        btfss PIR2,EEIF         ;da li je završen proces upisa u EEPROM
        goto pos_write_wait     ;ne...čekaj još
        bcf PIR2,EEIF           ;da...resetuj zastavicu prekida
        bsf INTCON,GIE          ;omogući nove prekide
        BANK3
        bcf EECON1,WREN         ;onemogućenje novog upisa
        bcf STATUS,RP0          ;bank2
        incf EEADR,f            ;inkrementiranje pokazivača EEADR
        incf FSR,f              ;inkrementiranje pokazivača FSR
        bcf STATUS,RP1          ;Bank 0
        decfsz BLOCK_CNT,f      ;dekr. brojača lokacija. Da li je on nula?
        goto pos_write          ;ne...piši na sljedeću lokaciju
        bcf MODE,SET            ;isključi set mode
        movf CHANNEL,w          ;
        movwf COMMAND           ;
        goto exec_digit         ;

;*************************************************************************
;**                                                                     **
;**  TAKE_REQPOS                                                        **
;**  Rutina koja treba da očita četvorocifrenu vrijednost iz registara  **
;**  DD_UNITS do DD_THOUS i konvertuje je u dvobajtnu binarnu           **
;**  vrijednost u registrima REQPOSH:REQPOSL                            **
;**  Ako je broj u dozv. opsegu i konverzija uspije, po završetku       **
;**  u akumulatoru je broj različit od nule.                            **
;**  Rutina koristi registre REG0,.., REG3, REQPOSH_TEMP i REQPOSL_TEMP **
;**                                                                     **
;*************************************************************************
take_reqpos
        movf DD_UNITS,w         ;
        movwf REG0              ;
        sublw d'09'             ;
        btfss STATUS,C          ;da li je broj veći od 9 (možda blanko)?
        clrf REG0               ;da...smatraj ga nulom
        movf DD_TENS,w          ;
        movwf REG1              ;
        sublw d'09'             ;
        btfss STATUS,C          ;da li je broj veći od 9 (možda blanko)?
        clrf REG1               ;da...smatraj ga nulom
        movf DD_HUNS,w          ;
        movwf REG2              ;
        sublw d'09'             ;
        btfss STATUS,C          ;da li je broj veći od 9 (možda blanko)?
        clrf REG2               ;da...smatraj ga nulom
        movf DD_THOUS,w         ;
        movwf REG3              ;
        sublw d'09'             ;
        btfss STATUS,C          ;da li je broj veći od 9 (možda blanko)?
        clrf REG3               ;da...smatraj ga nulom

        clrf REQPOSH_TEMP       ;
        movf REG0,w             ;
        movwf REQPOSL_TEMP      ;
add10
        clrw                    ;
        addwf REG1,w            ;
        btfsc STATUS,Z          ;
        goto add100             ;
loop10
        movlw d'10'             ;
        addwf REQPOSL_TEMP,f    ;
        decfsz REG1,f           ;
        goto loop10             ;
add100
        clrw                    ;
        addwf REG2,w            ;
        btfsc STATUS,Z          ;
        goto add1000            ;
loop100
        movlw d'100'            ;
        addwf REQPOSL_TEMP,f    ;
        btfsc STATUS,C          ;
        incf REQPOSH_TEMP,f     ;
        decfsz REG2,f           ;
```

```
        goto loop100            ;
add1000
        clrw                    ;
        addwf REG3,w            ;
        btfsc STATUS,Z          ;
        goto num_val            ;
loop1000
        movlw d'03'             ;
        addwf REQPOSH_TEMP,f    ;
        movlw d'232'            ;
        addwf REQPOSL_TEMP,f    ;
        btfsc STATUS,C          ;
        incf REQPOSH_TEMP,f     ;
        decfsz REG3,f           ;
        goto loop1000           ;

;*************************************************************************
;*  Provjera da li je traženi ugao unutar opsega 0.0 - 359.9
;*  Ako nije u REQPOS se vraćaju stare vrijednosti i odustaje od pomjeraja
;*************************************************************************
num_val
        movf REQPOSH_TEMP,w     ;
        sublw d'14'             ;
        btfss STATUS,C          ;
        retlw 0x00              ;povratak sa 0 u akumulatoru
        btfss STATUS,Z          ;
        goto finish_take        ;
        movf REQPOSL_TEMP,w     ;
        sublw d'15'             ;
        btfss STATUS,C          ;
        retlw 0x00              ;povratak sa 0 u akumulatoru
finish_take
        movf REQPOSL_TEMP,w     ;uzmi privremenu vrijednost REQPOSL
        movwf REQPOSL           ;...i proglasi je važećom
        movf REQPOSH_TEMP,w     ;vrati staru vrijednost od REQPOSH
        movwf REQPOSH           ;...i proglasi je važećom
        retlw 0x01              ;povratak sa 1 u akumulatoru

;*************************************************************************
;**                                                                     **
;**  RUTINA ZA OBRADU KOMANDE NOP                                       **
;**  Izvršava se kada se traži neka neimplementirana naredba            **
;**                                                                     **
;*************************************************************************
exec_nop
        return

        end
```

LITERATURA

- M. R. Stojić DIGITALNI SISTEMI UPRAVLJANJA, Nauka Beograd '90.
- J. Bubalo, R.Ilić, V. Veljković OPTOELEKTRONIKA, Narodna biblioteka Srbije '99.
- S. N. Vukosavić PREDAVANJA IZ PREDMETA PROCESNI RAČUNARI
- S. N. Vukosavić PREDAVANJA IZ PREDMETA MIKROPROCESORSKO UPRAVLJANJE EL. POGONIMA
- S. Frank INTELLIGENT REMOTE POSITIONER, Microchip Application Note AN531
- B. L. Dokić ENERGETSKA ELEKTRONIKA, ETF Banjaluka 2000.
- V. Vučković ELEKTRIČNI POGONI, ETF Beograd '97.
- V. Kremin RC5 CODEC, Cypress MicroSystems Application Note AN2091 2/2003.
- D. Andrić PROGRAMATOR PIC MIKROKONTROLERA, časopis MikroElektronika br.3 7&8/98.
- S. N. Vukosavić MIKROPROCESORSKO UPRAVLJANJE EL. POGONIMA, skripta
- Microchip datasheet DS30292A: PIC16F87X – 8BIT CMOS FLASH MICROCONTROLLERS
- D. Andrić STEP MOTOR & PIC, časopis MikroElektronika br.6 2/99.
- P. P. Acarnley STEPPING MOTORS: A GUIDE TO MODERN THEORY AND PRACTISE, Peter Peregrinus Ltd., London, UK, '84