SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
SVEUČILIŠNI PREDDIPLOMSKI STUDIJ ELEKTROTEHNIKE

Filip Vorkapić

RAZVOJNI SUSTAV ZA PIC MIKROKONTROLERE

ZAVRŠNI RAD

Rijeka, 2011.

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
SVEUČILIŠNI PREDDIPLOMSKI STUDIJ ELEKTROTEHNIKE

Filip Vorkapić

RAZVOJNI SUSTAV ZA PIC MIKROKONTROLERE

ZAVRŠNI RAD

Student: Filip Vorkapić
JMBAG: 0036435821
Mentor: doc. dr. sc. Viktor Sučić
Kolegij: Digitalna logika
Smjer: Automatika

Rijeka, 2011.

(this page is intentionally left blank)

(ubaci original zadatka završnog rada)

(this page is intentionally left blank)

(ubaci posvetu)

SADRŽAJ

1. Uvod

Kroz nekoliko poglavlja koja slijede biti će govora o sljedećim temama: općenito o mikrokontrolerima, razvojnim sustavima za PIC mikrokontrolere te o osnovama rada sa mikrokontrolerima, vrste programatora i prijedlog programatorskog rješenja za PIC mikrokontrolere.

1.1. Mikrokontroler

Mikrokontroleri su elektronički uređaji građeni u obliku čipa, a najbolje ih opisuje izraz „maleno računalo“ jer po svojim mogućnostima oni to upravo i jesu. Njihovu strukturu čine mikroprocesorska jedinica, RAM i ROM memorija, flash memorija te ovisno o modelu i do nekoliko različitih modula za komunikaciju sa računalima i ostalim uređajima, a među njima su moduli za: serijsku komunikaciju, paralelnu komunikaciju, USB komunikaciju, Ethernet komunikaciju, itd. Primjena mikrokontrolera je vrlo raznovrsna: od paljenja i gašenja LED dioda, upravljanja sustavom automatizirane („pametne“) kuće, pa sve do upravljanja kompletnim automatiziranim proizvodnim procesom nekog postrojenja. Svaki model mikrokontrolerskog čipa koji se proizvede nije isti po karakteristikama i mogućnostima jer se oni izrađuju upravo po zahtjevima procesa za koji su namjenjeni, pa tako primjerice industrijski mikrokontroleri imaju puno veću memoriju i brzinu obrade podataka za razliku od nekih mikrokontrolera koji se koriste za edukacijske svrhe na kojima se izvršavaju puno jednostavniji programi i upravljaju jednostavnije komponente poput LED dioda, sedam segmentnih ekrana ili sl.

1.2. PIC mikrokontroler

PIC mikrokontroler (engl. Programmable Intelligent computer), originalni proizvod tvrtke Microchip, vrlo je popularan među inžinjerima i elektroničkim početnicima, a dolazi u puno različitih izvedbi, svaka sa posebnim komponentama i mogućnostima. Mnogi elektronički projekti mogu se vrlo lako izvesti pomoću mikrokontrolera iz PIC obitelji, poput digitalnih satova, vrlo jednostavnih video igrica, robota, servo kontrolera i mnogih drugih. Njihova primjena je vrlo široka, a njihove mogućnosti i varijante su nebrojene, a može ih se nabaviti po vrlo pristupačnim cijenama što ih čini još popularnijima. Upravo zbog svoje

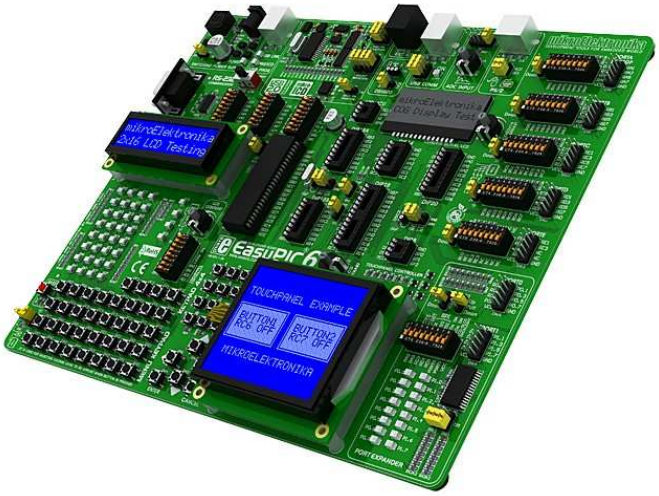
raznovrsnosti, ali ponajviše zbog prilično jednostavnog programiranja, PIC mikrokontroleri imaju dosta veliku primjenu u edukaciji, kako srednjoškolskoj tako i fakultetskoj, u digitalnoj logici, digitalnoj elektronici, automatici i drugim srodnim područjima.

1.3. Ideja korištenja PIC mikrokontrolera u edukacijske svrhe

Osobno smatram da su PIC mikrokontroleri jako bitni u edukaciji studenata elektrotehnike koje imalo zanima automatika ili elektronika jer programiranje PIC-eva nije zahtjevno, a studenti mogu puno naučiti iz rada sa PIC mikrokontrolera. Ipak nemjerljivo je važno da se, nakon naučene teorije o nekom području, znanje podkrijepi praktičnim primjerima u koje se more uložiti mala količina truda da bi se zadaci izvršili, ali količina iskustva i znanja koja se iz tih primjera dobije je zamjetljiva, u najmanju ruku.

2. Razvojni sustav za mikrokontrolere

Da bi se mogao objasniti rad sa mikrokontrolerima najbolji način je da se prije samog rada objasni razvojna okolina u kojoj su se vodila testiranja mikrokontrolera. Najčešće korišteni i najpraktičniji razvojni sustav za mikrokontrolere su razvojne pločice (*engl. development board*). Ova je razvojna okolina u širokoj primjeni, i često je koriste i početnici u svom radu, ali i napredni korisnici. Osnovna prednost ovakve razvojne okoline je njezina jednostavnost za upotrebu, dok je njezina mana što ne udovoljava svim uvjetima koji su postavljeni za kategorizaciju. Tipično, ovakva razvojna okolina nudi krajnjem korisniku mogućnost jednostavne promjene načina spajanja perifernih jedinica sa korištenim mikrokontrolerom, stvaranje novih perifernih jedinice za korištenje u dizajnu mikrokontrolerskih sustava te njihovo spajanje na razvojnu okolinu, te donekle dozvoljava promjene načina spajanja korištenog mikrokontrolera sa osnovnim modulima koji su potrebni za rad samog mikrokontrolera.



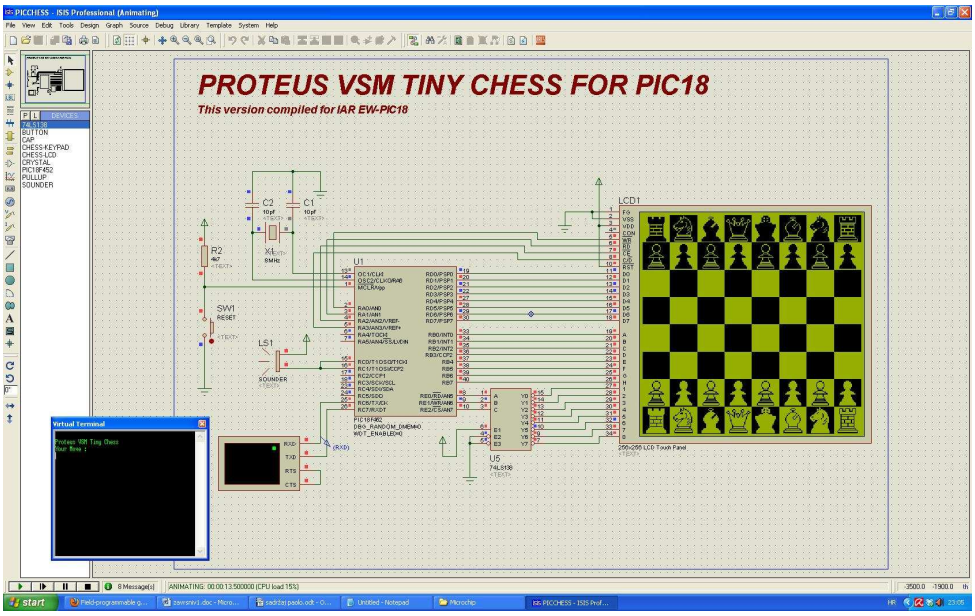
Slika 2.1 Razvojna pločica za mikrokontrolerske sustave – mikroElektronika [S1]

Promjena načina spajanja perifernih jedinica mikrokontrolera izvedena je pomoću sklopki smještenih na samoj razvojnoj pločici, dok je promjena osnovnih modula potrebnih za rad mikrokontrolera djelomično moguća i to promjenom samih komponenti na razvojnoj pločici.

Osim gore prikazane razvojne pločice na tržištu je moguće pronaći i razvojne pločice drugih proizvođača poput Microchip-ovih razvojnih pločica. Osnovne razlike između svih tih sustava jesu u njihovoj sveobuhvatnosti; neke su pločice namijenjene razvoju cjelokupnih sustava, dok su neke orijentirane na specifične dijelove mikrokontrolerskih sustava.

Slika 2.2 Specijalizirana razvojna pločica – Microchip [S2]

U skupinu programskih razvojnih okolina svrstavaju se svi oni programski paketi koji podržavaju projektiranje i testiranje rada sklopovlja mikrokontrolerskih sustava. Ovakve razvojne okoline moraju biti u mogućnosti ponuditi krajnjem korisniku razvoj mikrokontrolerskih sustava od osnovnih modula do perifernih jedinica (Slika 2.3).



Slika 2.3 Sučelje programske razvojne okoline ISIS Proteus

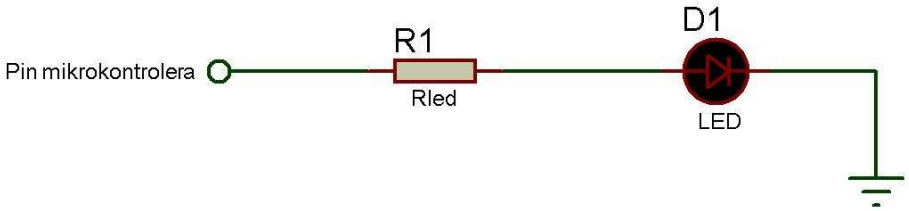
Općenito gledajući, ovakve su razvojne okoline vrlo pogodne za rad jer nude krajnjem korisniku izuzetnu fleksibilnost i neograničen broj iteracija razvoja sustava bez potrebe za izradom sklopovskih prototipa, što može uštediti znatan dio budžeta namijenjenog za izvedbu projekta. Također, prikazana razvojna okolina (Slika 2.3) nudi simulaciju mnogih standardnih perifernih jedinica, što projektantu uvelike olakšavao posao razvoja. Osnovni nedostatak ovakvih sustava, kao i svih simulacijskih sustava, je razlika između simulacijskih podataka i podataka prikupljenih mjerenjem veličina na istom implementiranom sustavu, međutim napretkom simulacijskog programa taj nedostatak sve više iščezava.

2.1. Periferne jedinice mikrokontrolera

Periferne jedinice se mogu podijeliti na ulazne, izlazne i ulazno-izlazne periferne jedinice, po smjeru protoka podataka između njih i mikrokontrolera, dok se po njihovoj funkciji mogu podijeliti na periferije za digitalni prikaz stanja, za digitalne ulaze, za grafički i alfanumerički prikaz podataka, za komunikaciju s drugim mikrokontrolerima, računalima ili sustavima, za pohranu podataka, za mjerenje vremena, te za vezu s analognim sklopovima. U daljnjem tekstu periferne su jedinice podijeljene upravo po njihovoj funkciji.

2.1.1. LED diode

Led (*engl. Light emitting diode*) diode koriste se za prikazivanje digitalnog stanja pojedinog izvoda (*engl. pin*) mikrokontrolera. Tipično se spajaju na način da je dioda svijetli kada je na izvodu mikrokontrolera stanje logičke jedinice, a ne svijetli kada je izvod u stanju logičke nule ili stanju visoke impedancije.

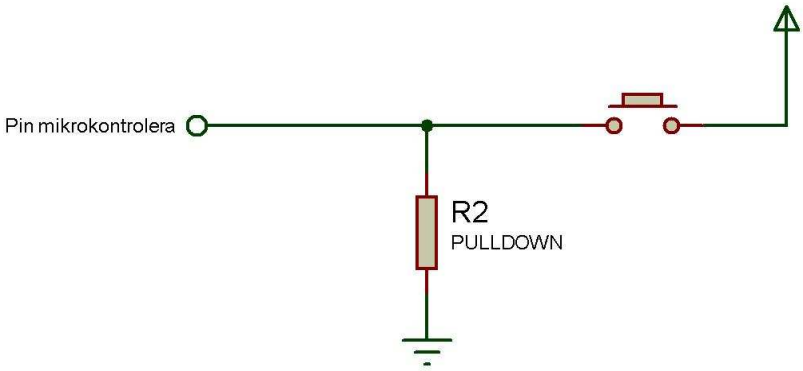


Slika 2.4 Tipični spoj LED dioda za prikaz stanja

Pri spajanju LED dioda na mikrokontroler potrebno je paziti da dioda ne zahtjeva više od 20 mA struje, što je tipična mogućnost izvoda mikrokontrolera. Ukoliko struja prelazi navedenu vrijednost moguće je iskoristiti LED diodu drugačijih karakteristika ili između mikrokontrolera i diode postaviti pojačalo.

2.1.2. Tipkala i sklopke

Tipkala i sklopke služe za davanje ulaznih podataka u mikrokontroler (Slika 2.5). Najčešće se koriste za korisničke naredbe (tipkala za kretanje po izborniku, start/stop tipkala itd.) i kao senzori digitalnih stanja (krajnje sklopke, mehanički brojači itd.).



Slika 2.5 Tipični spoj tipkala

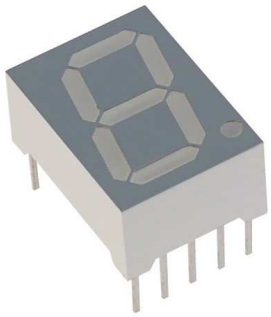
Otpornik prikazan na shemi služi da bi pin mikrokontrolera postavio u stanje logičke nule kada tipkalo nije pritisnuto (*engl. pull-down resisitor*).

2.1.3. Ekrani

Ekrane korištene s mikrokontrolerima možemo podijeliti na LED i LCD (*engl. Liquid crystal display*) po njihovoj unutarnjoj strukturi, odnosno na alfanumeričke i grafičke po njihovim mogućnostima prikaza.

2.1.3.1. Sedam segmenti LED ekrani

Sedam segmentni LED ekrani koriste se za prikaz znamenki 0-9 u dekadskom brojevnom sustavu, te za prikaz znamenki 0-F u heksadekadskom brojevnom sustavu.

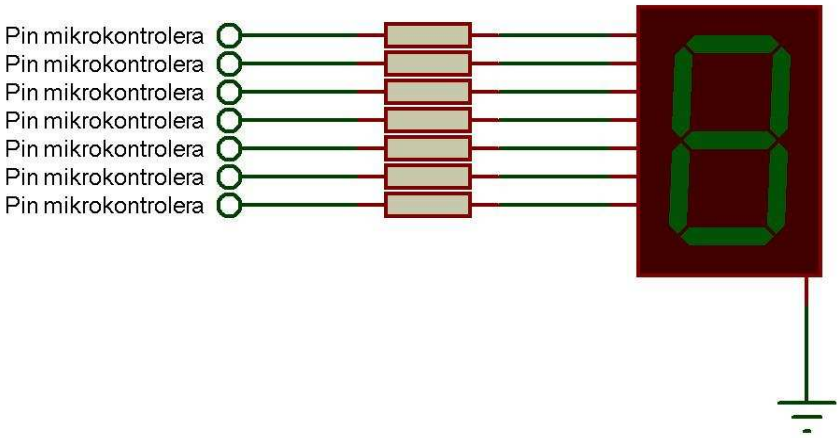


Slika 2.6 Sedam segmentni ekran [S4]

Mogu se koristiti kao samostalne periferije za spajanje na mikrokontroler (Slika 2.6) (svaki ekran se spaja zasebno) ili kao grupa ekrana za prikaz višeznamenkastih brojeva (više ekrana se spaja u multipleksirani način rada).

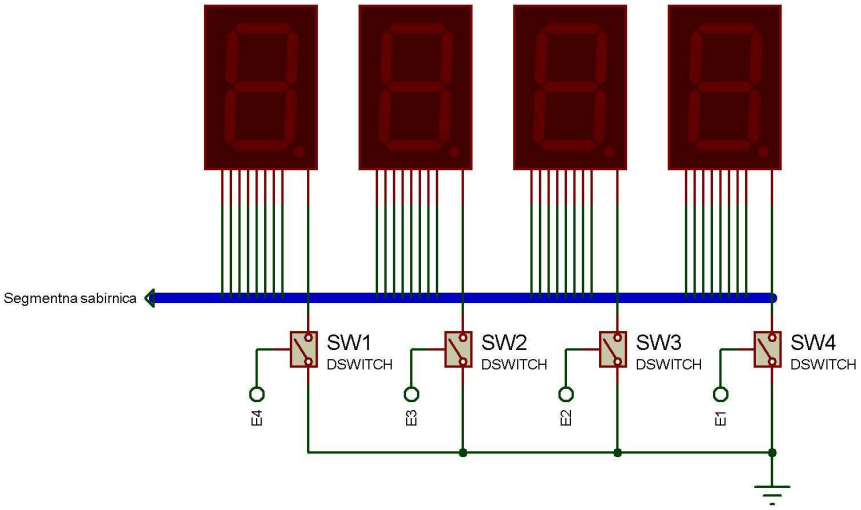
Slika 2.7 Sedam segmenti ekrani u multipleksiranom načinu rada [S5]

Spajanje svakog ekrana zasebno (Slika 2.8) izvodi se tako da se svaki izvod sedam segmentnog ekrana spoji preko predotpornika na izvod mikrokontrolera, dok je zajednička katoda spojena na masu (kod ekrana sa zajedničkom anodom anoda bi bila spojena na napona napajanja).



Slika 2.8 Spoj jednog sedam segmentnog ekrana

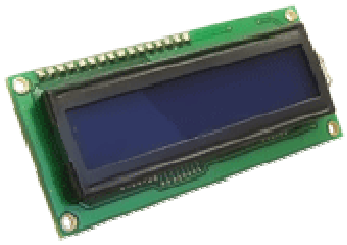
Kod multipleksiranog načina rada izvodi segmenata spojeni na segmentnu sabirnicu, a zajedničke anode spajaju se na sklopke koje služe kako bi se moglo mikroprocesorom odrediti u kojim će se trenutcima uključiti odnosno isključiti pojedini ekrani u skupini. Ovakav način rada iskorištava tromost ljudskog oka za prikaz i time štedi broj korištenih izvoda mikrokontrolera (umjesto 28 koristi se 11 izvoda za rad s 4 ekrana).



Slika 2.9 Sedam segmentni ekrani u multipleksiranom načinu rada

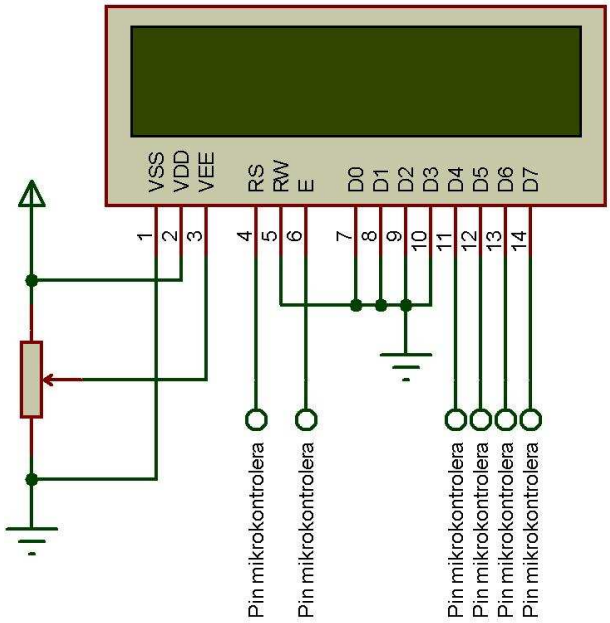
2.1.3.2. Alfanumerički LCD ekrani

Alfanumerički LCD ekrani (Slika 2.10) vrlo su jednostavni za korištenje te su stoga vrlo često korišteni za prikaz podataka u mikrokontrolerskim sustavima. Također, pogodni su jer mogu prikazivati više znakova, pa olakšavaju korisniku sustava lakšu interakciju. Postoje izvedbe LCD ekrana s dva ili četiri reda, te osam, šesnaest ili dvadeset znakova u svakom retku.



Slika 2.10 Alfanumerički LCD ekran [S6]

Postoje izvedbe s paralelnom i serijskom komunikacijom s mikrokontrolerom, no paralelna je u najširoj upotrebi. Osim prikaza slova i brojeva moguće je definirati i prikaz posebnih znakova stvorenih od strane korisnika. Način spajanja na mikrokontroler prikazan je na donjoj shemi (Slika 2.11).



Slika 2.11 LCD ekran s paralelnom komunikacijom

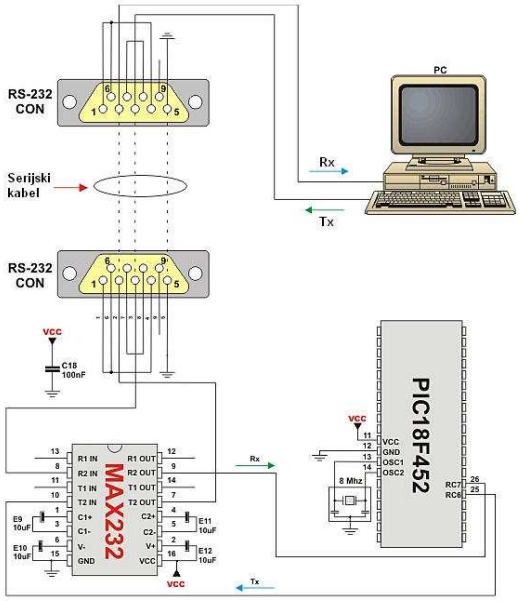
2.1.4. Komunikacijske periferije

Pod pojmom komunikacijskih periferija podrazumijevamo sve one komponente i/ili skupine komponenata koji omogućavaju mikrokontroleru da komunicira s drugim uređajima poput osobnog računala, mobilnog uređaja itd. različitim protokolima komunikacije. Danas se koristi mnogo različitih vrsta komunikacije poput serijske, paralelne, „One-wire“, SPI, I2C, Can i Ethernet komunikacije. Svaki od tih načina pogodan je za različite udaljenosti komunikacije, podatke koje šaljemo komunikacijskim kanalom, brzine prijenosa itd.

2.1.4.1. Serijska komunikacija – RS232

Serijska komunikacija je u prošlosti bila najčešće korištena komunikacija. Najveća je prednost ove komunikacije što iziskuje samo dva vodiča za komunikaciju, jedan od kojih je vodič kojim mikrokontroler šalje podatke priključenoj periferiji s kojom komunicira (*engl. Tx line, Transmit line*), a drugi vodič prenosi podatke od periferije do mikrokontrolera koji prima podatke (*engl. Rx line, Recieve line*). Za komunikaciju mikrokontrolera i osobnog računala potrebno je između mikrokontrolera i računala postaviti integrirani sklop za prilagodbu naponskih razina, jer računalo daje signale od 12V dok mikrokontroler tipično daje signale od

5V. U tu se svrhu najčešće koristi integrirani sklop MAX232. Shema spajanja prikazana je na donjoj slici (Slika 2.12).

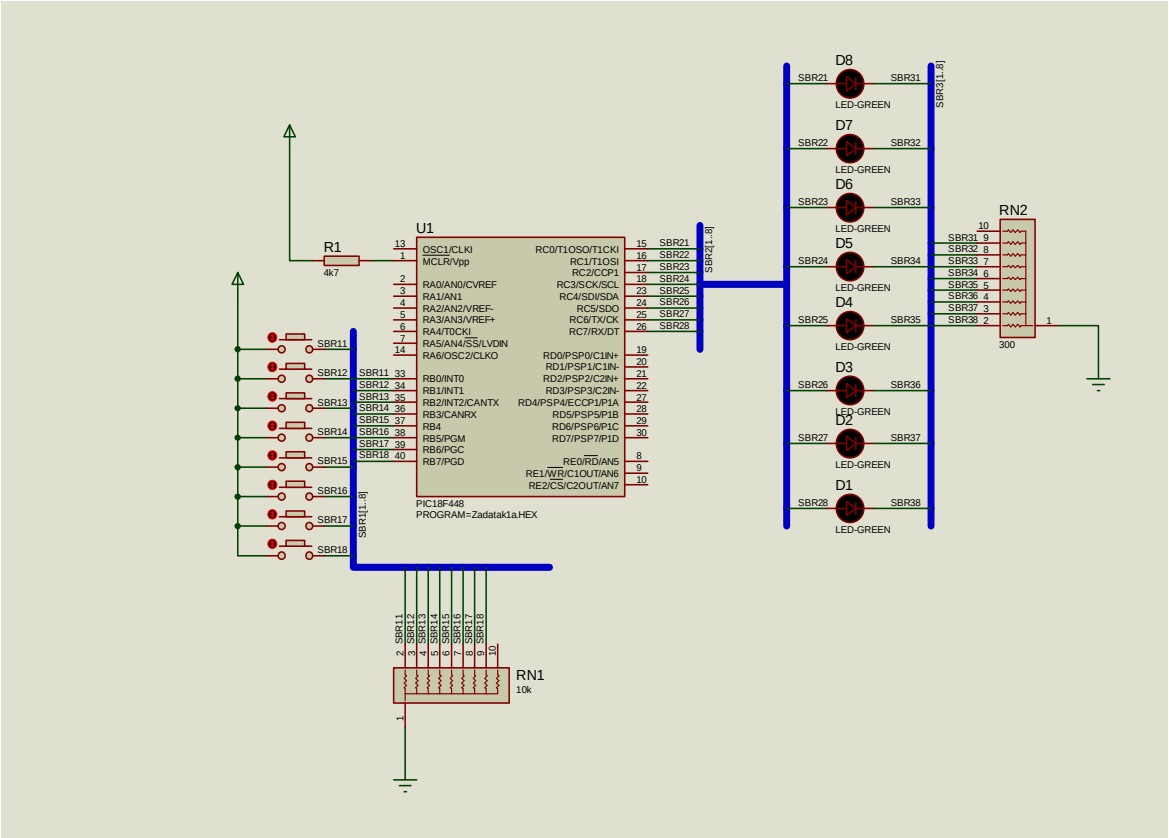


Slika 2.12 Serijska komunikacija mikrokontrolera i osobnog računala [S8]

3. Rad sa PIC mikrokontrolerima

3.1. Digitalni ulazno-izlazni portovi mikrokontrolera

Potrebno je napraviti mikrokontrolerski sustav koji će prikazati osnovne mogućnosti PIC mikrokontrolera u području djelovanja njegovih portova kao digitalnih ulaza i digitalnih izlaza. Neka jedan port mikrokontrolera bude ulaznog tipa i jedan port izlaznog tipa. Kao ulazne komponente spojite tipkala koja rade u pozitivnoj logici (kad su pritisnuta daju stanje logičke jedinice). Kao izlazne komponente spojite LED diode.



Slika 3.1: *Proteus design* rješenja mikrokontrolerskog sustava iz prvog zadatka

Opis Slike 3.1: Za dizajniranje ovog, a i svih ostalih, mikrokontrolerskih sustava korišten je PIC18F448 mikrokontroler. Na port B (koji je kasnije postavljen kao port za ulazne signale) spojeno je 8 digitalnih tipkala (svako tipkalo ima svoj izvod na mikrokontroleru) koja funkcioniraju zasebno preko *pull-down* otpornika što omogućava da u slučaju da nije pritisnuto pojedino tipkalo na portu B je stanje logičke “0”, a ako je pritisnuto je stanje logičke “1”. Na port C (koji je kasnije

postavljen kao port za izlazne signale) spojeno je 8 LED dioda za prikazivanje stanja na ulazu odnosno na portu B. Elementi RN1 i RN2 predstavljaju *"resistor networks"* (otporničke mreže) koji su nam vrlo povoljne za koristiti u praksi jer nam fizički zahtjevaju manje prostora na pločici od 10 zasebnih otpornika. Debele, plave linije predstavljaju sabirnice, a da bi mikrokontroler izvršavao svoje zadatke potrebno je master clear izvod mikrokontrolera spojiti preko R1 (otpornik od 4,7 kΩ) na napon od 5 V (predstavljen u dizajnu sa strelicom prema gore), jer u protivnom bi mikrokontroler konstantno bio u stanju reset. Slika 1 se odnosi na kompletni prvi zadatak (1., 2., 3. i 4. podzadatak).

3.1.1. Rad sa LED diodama – 1. zadatak

Potrebno je napraviti program koji će stanja na tipkalima prikazati na LED diodama (ako je pritisnuto i-to tipkalo mora svijetliti i-ta dioda).

MicroPascal kod:

```
program Zadatak1a;              //naziv programa
begin                           //početak programa
  trisb := 255;                 //postavljanje vrijednosti trisB registra
  trisc := 0;                   //postavljanje vrijednosti trisC registra
repeat                          //početak repeat-until petlje
portc := portb;                 //dodijeljivanje vrijednosti portC-u
until 1=2;                      //uvjet repeat-until petlje
end.                            //kraj programa
```

Objašnjenje koda: Tris registri koji se nalaze unutar mikrokontrolera nam služe za određivanje toka podataka (ulaz ili izlaz) pojedinog porta mikrokontrolera. Naredba "trisb := 255" određuje da svaki izvod porta B mikrokontrolera radi kao ulazni, a "trisc := 0" naredba određuje izvode porta C kao izlazne. Beskonačna repeat-until petlja omogućava da se operacija dodjeljivanja vrijednosti ulaza na izlaz izvrši više od jednog puta.

3.1.2. Rad sa LED diodama – 2. zadatak

Potrebno je napraviti program koji će prikazati binarno brojanje na LED diodama od 0 do 255.

MicroPascal kod:

```
program Zadatak1b;                  //naziv programa
var brojac: byte;                   //deklaracija varijable brojač
begin                               //početak programa
trisc := 0;                         //postavljanje vrijednosti trisC registra
 repeat                             //početak repeat-until funkcije
  for brojac := 0 to 255 do         //deklaracija parametara for petlje
   begin                            //početak for petlje
    portc := brojac;                //dodijeljivanje vrijednosti portC-u
    delay_ms(100);                  //određivanje kašnjenja
   end;                             //kraj for petlje
 until 1=2;                         //uvjet repeat-until petlje
end.                                //kraj programa
```

Objašnjenje koda: Da bi mogli simulirati brojač preko LED dioda potrebno je iskodirati neku vrstu brojača koja će se dodjeljivati na izlaz. Jedan od načina je preko “for-petlje”. Prvi korak nam je da deklariramo varijablu brojač koja je tipa byte, a potom odredimo parametre brojača unutar for-petlje. Ugrađena funkcija delay_ms() nam omogućuje, kao što i sam njen naziv kaže, kašnjenje signala.

3.1.3. Rad sa LED diodama – 3. zadatak

Potrebno je napraviti program koji će prikazati binarno brojenje na LED diodama od 255 do 0.

MicroPascal kod:

```
program Zadatak1c;                  //naziv programa
var brojac: byte;                   //deklaracija varijable brojač
begin                               //početak programa
 trisc := 0;                        //početak repeat-until petlje
  repeat                            //postavljanje vrijednosti trisC registra
   for brojac := 255 downto 0 do    //deklaracija parametara for petlje
    begin                           //početak for petlje
     portc := brojac;               //dodijeljivanje vrijednosti portC-u
     delay_ms(100);                 //određivanje kašnjenja
    end;                            //kraj for petlje
  until 1=2;                        //uvjet repeat-until petlje
end.                                //kraj programa
```

Objašnjenje koda: Postupak je vrlo sličan kao i u b) podzadatku samo što u zadavanju parametara brojača unutar for-petlje mijenjamo smjer brojanja sa ključnom riječi downto (u prošlom slučaju je bilo od 0 do 255, sada ide od 255 do 0).

3.1.4. Rad sa LED diodama – 4. zadatak

Potrebno je napraviti program koji će na LED diodama pokazati komplement stanja tipkala.

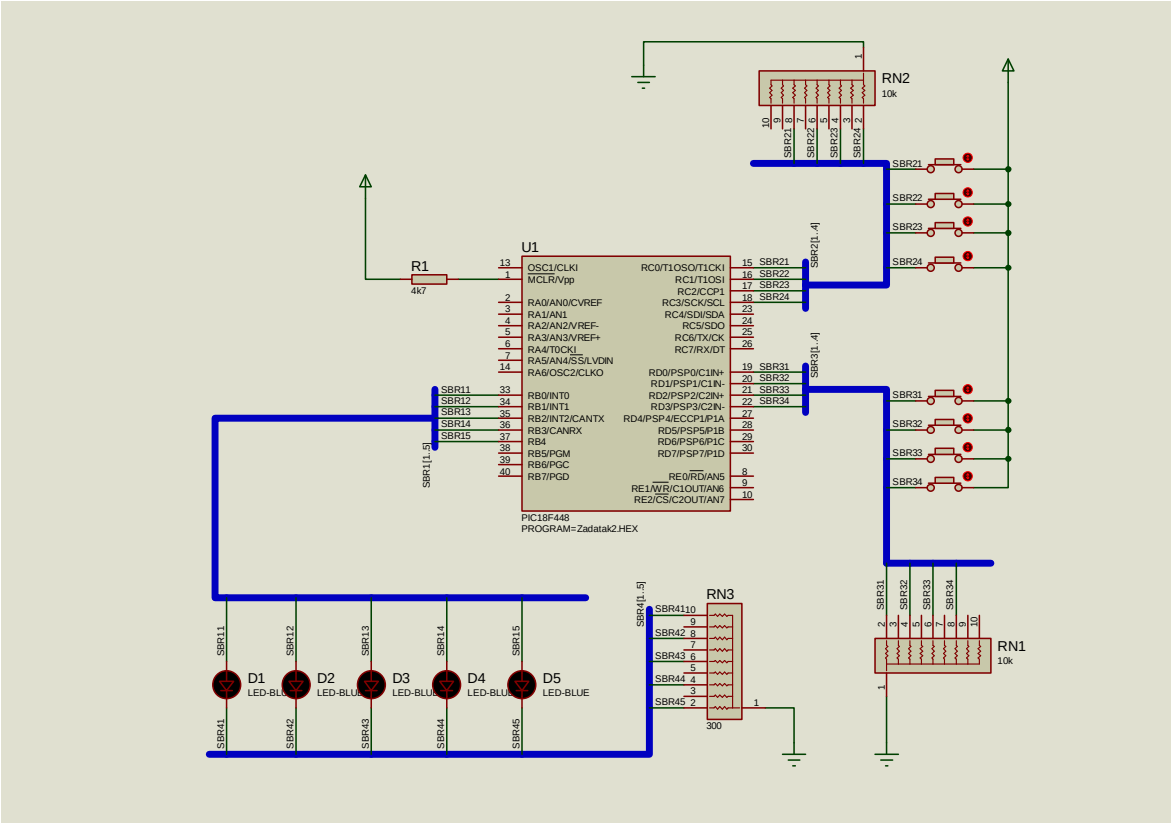
MicroPascal kod:

```
program Zadatak1d;              //naziv programa
begin                           //početak programa
 trisb := 255;                  //postavljanje vrijednosti trisB registra
 trisc := 0;                    //postavljanje vrijednosti trisC registra
  repeat                        //početak repeat-until petlje
   portc := not portb;          //dodijeljivanje vrijednosti portC-u
  until 1=2;                    //uvjet repeat-until petlje
end.                            //kraj programa
```

Objašnjenje koda: Komplement ulaza na izlazu postižemo korištenjem ugrađene funkcije *not.*

3.2. Izvedba 4-bitnog potpunog zbrajala

Potrebno je napraviti mikrokontrolerski sustav za zbrajanje dvaju četverobitnih brojeva. Kao ulaze uzmite tikala u pozitivnoj logici, a kao izlaze LED diode (sami odredite koliko vam tipkala i LED dioda treba kako biste prikazali sve potrebne ulazne kombinacije i sve potrebne zbrojeve).



Slika 3.2: *Proteus design* rješenja mikrokontrolerskog sustava iz drugog zadatka

Opis Slike 3.2: U ostvarivanju sustava korišten je PIC18F448 mikrokontroler kao i u prethodnom primjeru. Kao ulazne portove mikrokontrolera postavljeni su port C i port D, a kao izlazni port B. Izvod MCLR („Master Clear“) spojen je preko otpornika od 4,7 kΩ na napon od 5V potreban za normalan rad mikrokontrolera. Dva ulazna 4-bitna broja predstavljena su zasebno sa po 4 digitalna tipkala, čiji je rad u „pozitivnoj logici“ (kad je tipkalo pritisnuto daje logičku „1“) ostvaren preko „pull down“ otpornika (RN1 i RN2) spojenih na masu. Izlazni podatak biti će prikazan preko pet LED-dioda (toliko LED dioda je potrebno jer je najveći zbroj dva 4-bitna broja jednak 30, a može se prikazati sa najmanje sa 5-bitnim brojem); LED diode spojene su na masu preko otporničke mreže RN3.

MicroPascal kod:

```
program Zadatak2;                  //naziv programa
begin                              //početak programa
 trisb := 0;                       //postavljanje vrijednosti trisB registra
 trisc := 255;                     //postavljanje vrijednosti trisC registra
 trisd := 255;                     //postavljanje vrijednosti trisD registra
 cmcon := 0x07;                    //postavljanje vrijednosti cmcon registra
  repeat                           //početak repeat-until petlje
   portb:=portd + portc;           //dodijeljivanje vrijednosti portB-u
  until 1=2;                       //uvjet repeat-until petlje
end.                               //kraj programa
```

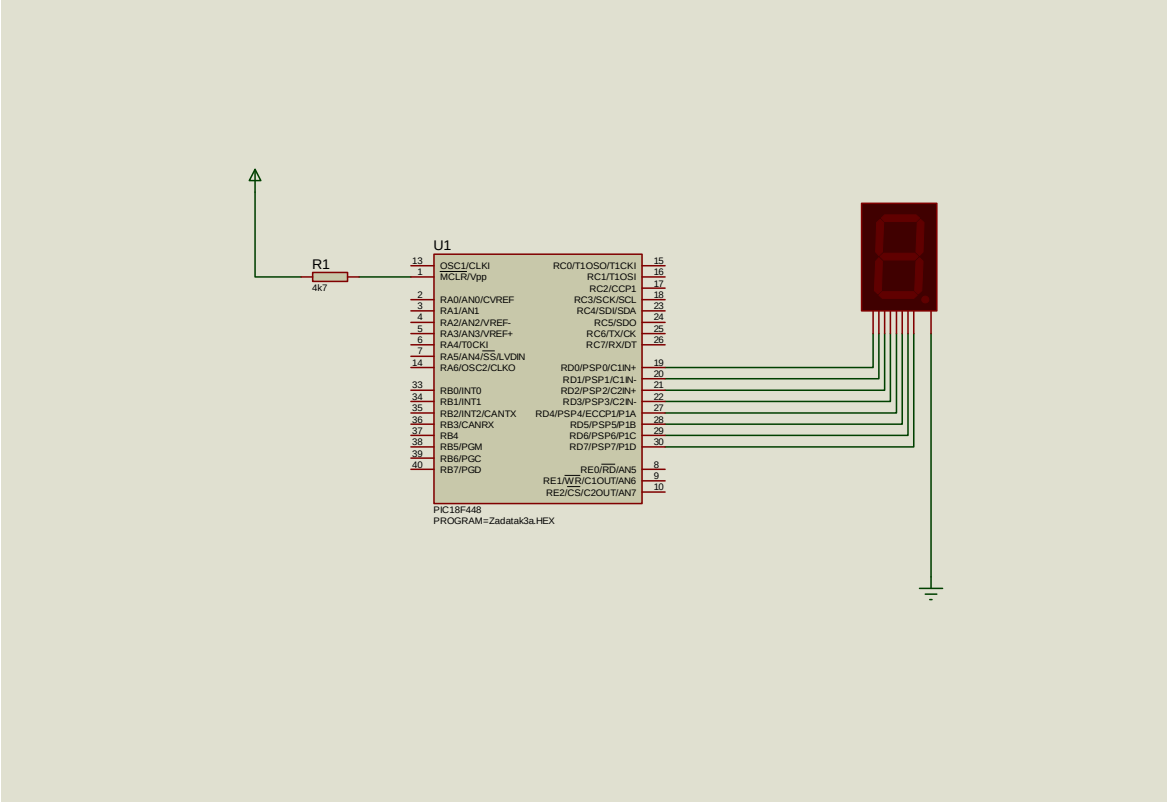
Objašnjenje koda: Ključnom riječi *tris* određujemo karakter pojedinog port-a mikrokontrolera, ulazni ili izlazni. Registar cmcon upravlja radom komparatora unutar mikrokontrolera, a ako mu se dodijeli heksadekadska vrijednost 0x07 (kao što je učinjeno u kodu) komparator se isključuje te se tako mogu koristiti svi izvodi porta D (u suprotnom se nebi mogli!). Operaciju zbrajanja koju mora obavljati mikrokontroler izvršavamo preko jednostavne jednadžbe, a da bi se ta operacija konstantno izvršavala stavljena je unutar besnonačne petlje *repeat..until 1=2;*

3.3. Upravljanje radom jednog sedam segmentnog ekrana

Potrebno je napraviti mikrokontrolerski sustav koji će prikazati mogućnosti PIC mikrokontrolera da upravlja jednim sedam segmentnim ekranom.

3.3.1 Ispis heksadecimalnog skupa znamenki

Potrebno je napraviti program koji će na ekranu ispisivati brojeve od 0-F (hex).



Slika 3.3.1: *Proteus design* rješenja mikrokontrolerskog sustava iz trećeg zadatka (podzadatak a)

Opis Slike 3.3.1: Sve što je potrebno da bi mikrokontrolerski sustav iz ovog zadatka izvršavao svoju funkciju je PIC18F448 mikrokontroler, sedam segmentni ekran i otpornik od 4,7 kΩ. Sedam segmentni ekran je sastavljen od 8 LED dioda koje su tako postavljene da bi mogle pokazivati znamenke. Svakom zasebnom LED diodom možemo upravljati davajući signale na pojedine izvodove ekrana (preko sedam izvodova se upravlja diodama za prikaz znamenaka, jedan izvod

nam koristi za upravljanje diodom koja prestavlja točku, a posljednji deveti izvod se spaja na masu da bi ekran mogao raditi). Radi upravljanja sedam segmentnim ekranom port D postavljen je kao izlazni i pojedini njegovi izvodovi spojeni su na 8 upravljačkih izvoda ekrana.

MicroPascal kod:

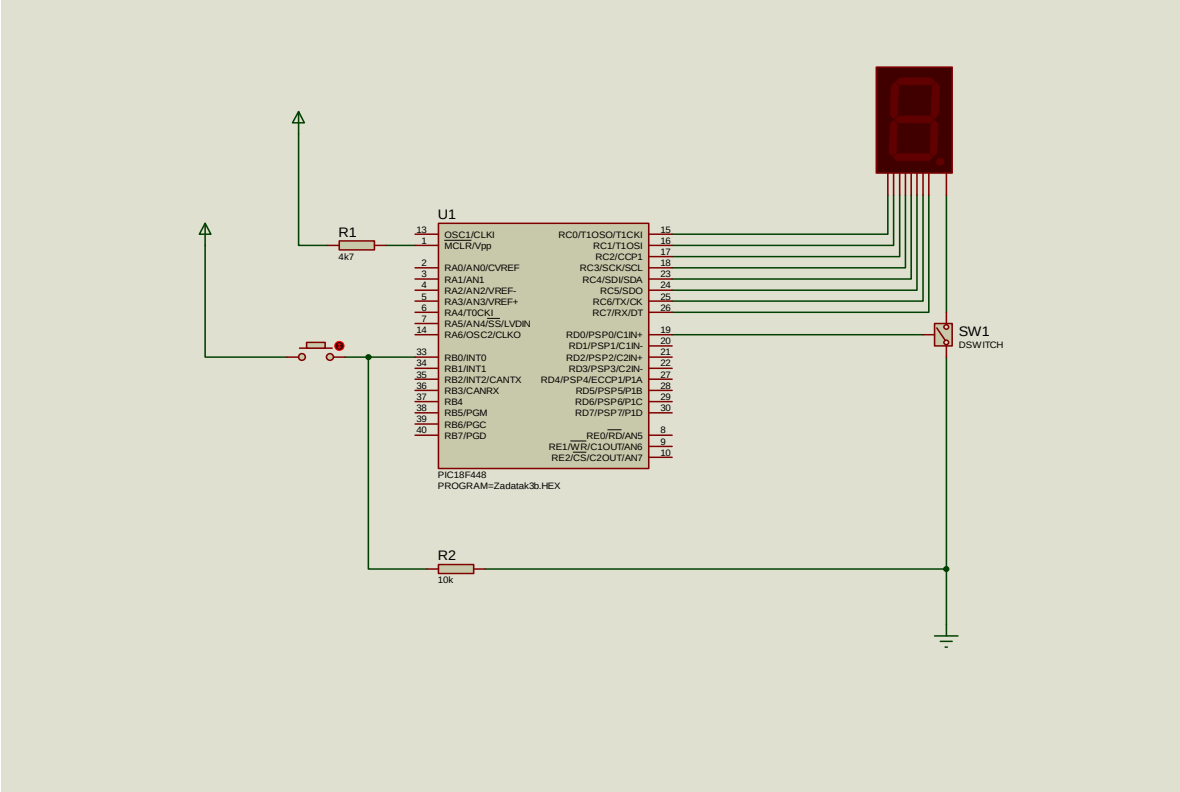
```
program Zadatak3a;                  //naziv programa
var brojac: byte;                   //deklaracija varijable brojač
begin                               //početak programa
 trisd := 0;                        //postavljanje vrijednosti trisD registra
  repeat                            //početak repeat-until petlje
   for brojac := 0 to 15 do         //deklaracija parametara for petlje
    begin                           //početak for petlje
     case brojac of                 //početak case funkcije
      0: portd:= 63;                //1. slučaj: dodijeljivanje vrijednosti portu D
      1: portd:= 6;                 //2. slučaj: dodijeljivanje vrijednosti portu D
      2: portd:= 91;                //3. slučaj: dodijeljivanje vrijednosti portu D
      3: portd:= 79;                //4. slučaj: dodijeljivanje vrijednosti portu D
      4: portd:= 102;               //5. slučaj: dodijeljivanje vrijednosti portu D
      5: portd:= 52;                //6. slučaj: dodijeljivanje vrijednosti portu D
      6: portd:= 125;               //7. slučaj: dodijeljivanje vrijednosti portu D
      7: portd:= 7;                 //8. slučaj: dodijeljivanje vrijednosti portu D
      8: portd:= 127;               //9. slučaj: dodijeljivanje vrijednosti portu D
      9: portd:= 111;               //10. slučaj: dodijeljivanje vrijednosti portu D
      10: portd:= 119;              //11. slučaj: dodijeljivanje vrijednosti portu D
      11: portd:= 124;              //12. slučaj: dodijeljivanje vrijednosti portu D
      12: portd:= 57;               //13. slučaj: dodijeljivanje vrijednosti portu D
      13: portd:= 94;               //14. slučaj: dodijeljivanje vrijednosti portu D
      14: portd:= 121;              //15. slučaj: dodijeljivanje vrijednosti portu D
      15: portd:= 113;              //16. slučaj: dodijeljivanje vrijednosti portu D
     end;                           //kraj case funkcije
    delay_ms(200);                  //određivanje kašnjenja
   end;                             //kraj for petlje
  until 1=2;                        //uvjet repeat-until petlje
end.                                //kraj programa
```

Objašnjenje koda: Zadatak je riješen uporabom *for*-petlje i *case* funkcije. Deklarirana je varijabla brojac, tipa *byte*, radi izvršavanja *for*-petlje. Postavljanjem vrijednosti trisD registra u „0“ određujemo ulazni karakter portD-a. Program funkcionira na sljedeći način: U svakom od 15 ciklusa for-petlje redni broj ciklusa („brojac“) se ubacuje u case funkciju i u odnosu na njegov iznos se pridodaje karakteristična dekadska vrijednost portD-u. Karakteristične dekadske vrijednosti određuju se po posebnom pravilu po kojem se LED diode u 7segmentnom ekranu uključuju i isključuju (svaka dioda predstavlja jedan bit u upravljačkom signalu koji

se dovodi na ekran). Kašnjenje je dodano radi usporavanja promjene ciklusa i prikladnijeg prikaza na ekranu.

3.3.2. Brojač objekata na proizvodnoj traci – 1. zadatak

Potrebno je napraviti program koji će simulirati brojač na proizvodnoj traci. Imate infracrveni senzor koji detektira prolaz objekta na traci. Svaki put kad objekt prodje senzor vam daje stanje log. jedinice duljine 150 ms. Svaki put kada senzor da signal povecajte brojac i prikazite to na 7seg ekranu. Ako brojac predje 9 neka krene opet od nule. Senzor simulirajte tipkalom.



Slika 3.3.2: *Proteus design* rješenja mikrokontrolerskog sustava iz trećeg zadatka (podzadatak b)

Opis Slike 3.3.2: Za ostvarenje mikrokontrolerskog sustava korištene su sljedeće komponente: PIC18F448 mikrokontroler, tipkalo, sklopka, otpornik od 4,7 kΩ i 10kΩ . Master Clear izvod mikrokontrolera spojen je na ulazni napon od 5V bez kojeg mikroprocesor ne bi radio. Na portC spojen je jedan 7segmentni ekran, dok je prvi izvod portD-a iskorišten za slanje upravljačkog signala prema digitalnoj

sklopci koja predstavlja virtualni „ON-OFF“ tipku za ekran. Na prvi izvod portB-a spojeno je digitalno tipkalo koje je dalje preko 10k otpornika spojeno na masu („pull down“ otpornik).

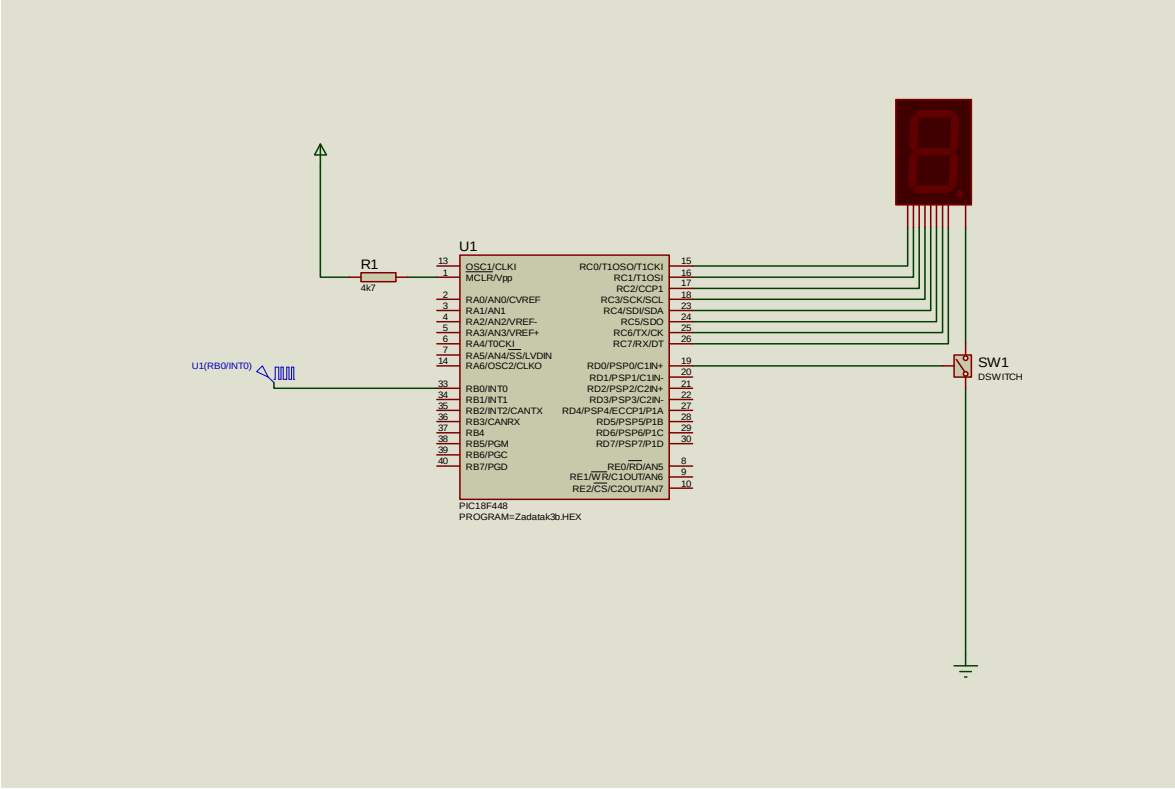
MicroPascal kod:

```
program Zadatak3b;                      //naziv programa
var brojac : byte;                      //deklaracija varijable brojac
  function mask(broj:byte): byte;       //određivanje parametara funkcije mask
            begin                       //početak funkcije mask
              case broj of              //početak case funkcije
                0: result:= 63;         //1. slučaj: dodijeljivanje vrijednosti parametru result
                1: result:= 6;          //2. slučaj: dodijeljivanje vrijednosti parametru result
                2: result:= 91;         //3. slučaj: dodijeljivanje vrijednosti parametru result
                3: result:= 79;         //4. slučaj: dodijeljivanje vrijednosti parametru result
                4: result:= 102;        //5. slučaj: dodijeljivanje vrijednosti parametru result
                5: result:= 109;        //6. slučaj: dodijeljivanje vrijednosti parametru result
                6: result:= 125;        //7. slučaj: dodijeljivanje vrijednosti parametru result
                7: result:= 7;          //8. slučaj: dodijeljivanje vrijednosti parametru result
                8: result:= 127;        //9. slučaj: dodijeljivanje vrijednosti parametru result
                9: result:= 111;        //10. slučaj: dodijeljivanje vrijednosti parametru result
             end;                       //kraj case funkcije
           end;                         //kraj funkcije mask
begin                                   //početak glavnog programa
 trisb:=1;                              //postavljanje vrijednosti tris B registra
 trisc:=0;                              //postavljanje vrijednosti tris C registra
 trisd:=0;                              //postavljanje vrijednosti tris D registra
 brojac:=0;                             //dodijeljivanje vrijednosti varijabli brojac
  repeat                                //početak repeat until petlje
   if portb.0 = 1 then                  //zadavanje parametara if funkcije
   if brojac <= 9 then                  //zadavanje parametara if funkcije
    begin                               //početak if funkcije
     brojac := brojac + 1;              //povećavanje vrijednosti varijable brojac
    end;                                //kraj if funkcije
   if brojac > 9 then brojac := 0;      //zadavanje parametara if funkcije
    portc := mask(brojac);              //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
   repeat                               //početak repeat until petlje
    portd.0 := 1;                       //dodijeljivanje vrijednosti prvom izvodu porta D
    delay_ms(4);                        //kašnjenje
    portd.0 := 0;                       //dodijeljivanje vrijednosti prvom izvodu porta D
   until portb.0 = 0;                   //kraj repeat until petlje
   if portb.0 = 0 then                  //zadavanje parametara if funkcije
    portc := mask(brojac);              //dodijejivanje vrijednosti portC-u pomoću funkcije mask
   repeat                               //početak repeat until petlje
    portd.0 := 1;                       //dodijeljivanje vrijednosti prvom izvodu porta D
    delay_ms(4);                        //kašnjenje
    portd.0 := 0;                       //dodijeljivanje vrijednosti prvom izvodu porta D
   until portb.0 = 1;                   //kraj repeat until petlje
  until 1=2;                            //kraj repeat until petlje
end.                                    //kraj programa
```

Objašnjenje koda: Radi lakšeg upravljanja sedam segmentnim ekranom uvodi se nova funkcija „mask“ preko koje će se u daljnjem kodu dodijeljivati vrijednost portu C. Preko tris registara određuju se karakteri pojedinih portova i to port C kao izlazni (njime upravljamo sedam segmentni ekran), port D kao izlazni (njime upravljamo digitalnu sklopku) i port B kao ulazni (na njega je spojeno digitalno tipkalo kojim upravljamo sustavom). Program funkcionira na sljedeći način: Ukoliko je pritisnuto tipkalo odnosno na prvom izvodu porta B je „1“, provjerava se vrijednost brojača te ako je manja od 9 se uvećava za 1, a ako je jednaka 9 se ponovno postavlja na 0. Pomoću funkcije mask vrijednost brojača se transformira i dodijeljuje portu C. Da bi se trenutna brojka prikazivala konstantno na ekranu koristi se inteligentan način iskorištavanja latencije ljudskog oka i to na način da svake 4 milisekunde uključujemo i isključujemo sklopku spojenu preko portD-a, a to se sve odvija dok je tipkalo pritisnuto, a kad ono nije pritisnuto vrijednost na prvom izvodu porta B je „0“, ne mijenja se vrijednost brojača te se ta nepromjenjena vrijednost brojača prikazuje na ekranu ponovno koristeći onaj isti inteligentan način iz prethodnog slučaja.

3.3.3. Brojač objekata na proizvodnoj traci – 2. zadatak

Potrebno je napraviti program koji će simulirati brojač na proizvodnoj traci. Imate infracrveni senzor koji detektira prolaz objekta na traci. Svaki put kad objekt prodje senzor vam daje stanje log. jedinice duljine 150 ms. Svaki put kada senzor da signal povecajte brojac i prikazite to na 7seg ekranu. Ako brojac predje 9 neka krene opet od nule. Senzor simulirajte naponskim izvorom koji će svakih 1 s dati signal od 150 ms.



Slika 3.3.3: *Proteus design* rješenja mikrokontrolerskog sustava iz trećeg zadatka (podzadatak c)

Opis Slike 3.3.3: Mikrokontrolerski sustav funkcionira na isti princip kao i sutav u prethodnom zadatku, jedina razlika je u tome što se u prošlom zadatku sustavom upravljalo digitalnim tipkalom, a u ovom zadatku se to radi generiranim pulsnim signalom. Za ostvarenje mikrokontrolerskog sustava korištene su sljedeće komponente: PIC18F448 mikrokontroler, generator digitalnog signala, digitalna dvopozicijska sklopka te otpornik od 4,7 kΩ. Master Clear izvod mikrokontrolera spojen je na ulazni napon od 5V bez kojeg mikroprocesor ne bi radio. Na portC spojen je jedan 7segmentni ekran, dok je prvi izvod portD-a iskorišten za slanje

upravljačkog signala prema sklopci koja predstavlja virtualni „ON-OFF“ tipku za ekran. Na prvi izvod porta B spojen je pulsni naponski izvor. Mikrokontrolerski sustav funkcionira na isti princip kao i sustav iz prethodnog zadatka.

MicroPascal kod:

```
program Zadatak3c;                      //naziv programa
var brojac : byte;                      //deklaracija varijable brojac
   function mask(broj:byte): byte;      //određivanje parametara funkcije mask
              begin                     //početak funkcije mask
                case broj of            //početak case funkcije
                   0: result:= 63;      //1. slučaj: dodijeljivanje vrijednosti parametru result
                   1: result:= 6;       //2. slučaj: dodijeljivanje vrijednosti parametru result
                   2: result:= 91;      //3. slučaj: dodijeljivanje vrijednosti parametru result
                   3: result:= 79;      //4. slučaj: dodijeljivanje vrijednosti parametru result
                   4: result:= 102;     //5. slučaj: dodijeljivanje vrijednosti parametru result
                   5: result:= 109;     //6. slučaj: dodijeljivanje vrijednosti parametru result
                   6: result:= 125;     //7. slučaj: dodijeljivanje vrijednosti parametru result
                   7: result:= 7;       //8. slučaj: dodijeljivanje vrijednosti parametru result
                   8: result:= 127;     //9. slučaj: dodijeljivanje vrijednosti parametru result
                   9: result:= 111;     //10. slučaj: dodijeljivanje vrijednosti parametru result
                end;                    //kraj case funkcije
              end;                      //kraj funkcije mask
begin                                   //početak glavnog programa
 trisb:=1;                              //postavljanje vrijednosti trisB registra
 trisc:=0;                              //postavljanje vrijednosti trisC registra
 trisd:=0;                              //postavljanje vrijednosti trisD registra
 brojac:=0;                             //dodijeljivanje vrijednosti varijabli brojac
  repeat                                //početak repeat until petlje
   if portb.0 = 1 then                  //zadavanje parametara if funkcije
   if brojac <= 9 then                  //zadavanje parametara if funkcije
    begin                               //početak if funkcije
     brojac := brojac + 1;              //povećavanje vrijednosti varijable brojac
    end;                                //kraj if funkcije
   if brojac > 9 then brojac := 0;      //zadavanje parametara if funkcije
    portc := mask(brojac);              //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
   repeat                               //početak repeat until petlje
    portd.0 := 1;                       //dodijeljivanje vrijednosti prvom izvodu porta D
    delay_ms(4);                        //kašnjenje
    portd.0 := 0;                       //dodijeljivanje vrijednosti prvom izvodu porta D
   until portb.0 = 0;                   //kraj repeat until petlje
   if portb.0 = 0 then                  //zadavanje parametara if funkcije
    portc := mask(brojac);              //dodijejivanje vrijednosti portC-u pomoću funkcije mask
   repeat                               //početak repeat until petlje
    portd.0 := 1;                       //dodijeljivanje vrijednosti prvom izvodu porta D
    delay_ms(4);                        //kašnjenje
    portd.0 := 0;                       //dodijeljivanje vrijednosti prvom izvodu porta D
   until portb.0 = 1;                   //kraj repeat until petlje
  until 1=2;                            //kraj repeat until petlje
end.                                    //kraj programa
```

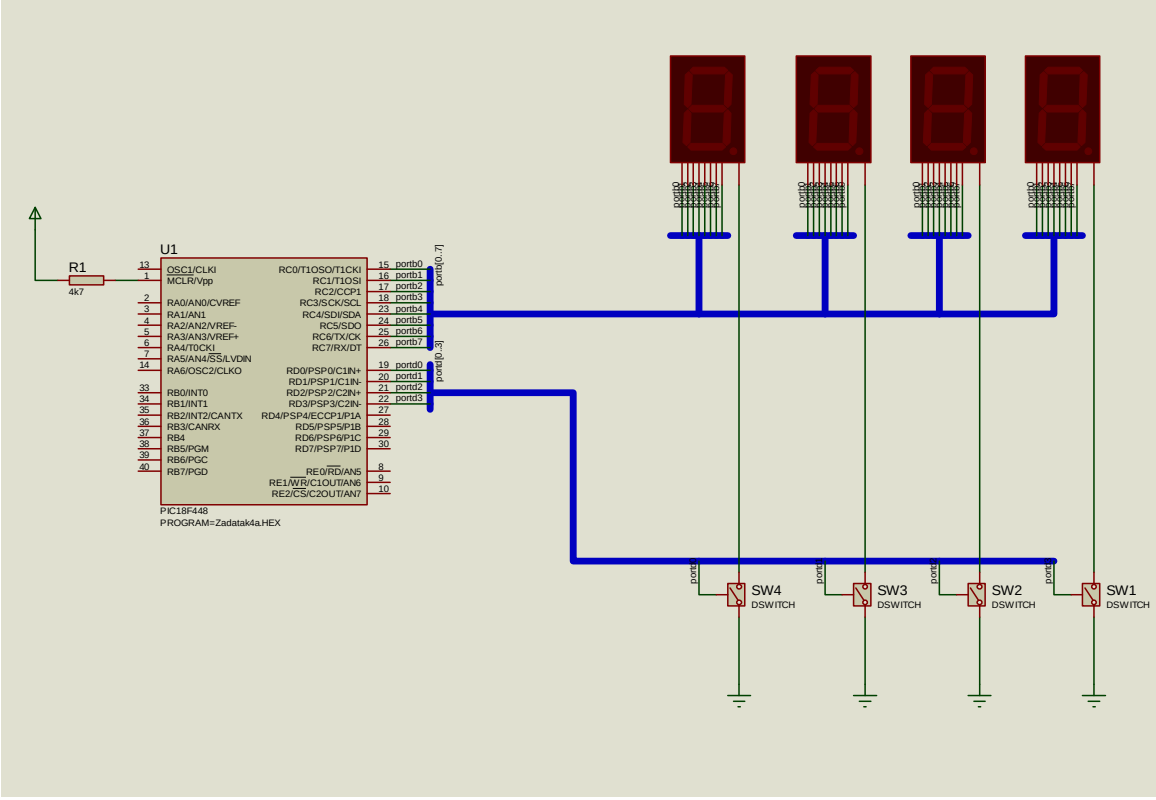
Objašnjenje koda: Radi lakšeg upravljanja sedam segmentnim ekranom uvodi se nova funkcija „mask“ preko koje će se u daljnjem kodu dodijeljivati vrijednost portu C. Preko tris registara određuju se karakteri pojedinih portova i to port C kao izlazni (njime upravljamo sedam segmentni ekran), port D kao izlazni (njime upravljamo digitalnu sklopku) i port B kao ulazni (na njega je spojeno pulsni izvor napona kojim upravljamo sustavom). Program funkcionira na sljedeći način: Ukoliko je na prvom izvodu porta B „1“, provjerava se vrijednost brojača te ako je manja od 9 se uvećava za 1, a ako je jednaka od 9 se ponovno postavlja na 0. Pomoću funkcije mask vrijednost brojača se transformira i dodijeljuje portu C. Da bi se trenutna brojka prikazivala konstantno na ekranu koristi se inteligentan način iskorištavanja latencije ljudskog oka i to na način da svake 4 milisekunde uključujemo i isključujemo sklopku spojenu preko porta D, a to se sve odvija dok je stanje na prvom izvodu porta B u „1“, a kad je ono „0“, ne mijenja se vrijednost brojača te se ta nepromjenjena vrijednost brojaca prikazuje na ekranu ponovno koristeći onaj isti inteligentan način iz prethodnog slučaja.

3.4. Upravljanje radom više multipleksiranih sedam segmentnih ekrana

Potrebno je napraviti mikrokontrolerski sustav koji će prikazati mogućnosti PIC mikrokontrolera da kontrolira s više sedam segmentnih ekrana.

3.4.1. Ispis statičkog teksta

Potrebno je napraviti program koji će na 4 sedam segmentna ekrana ispisati neki broj po želji.



Slika 3.4.1: *Proteus design* rješenja mikrokontrolerskog sustava iz četvrtog zadatka (podzadatak a)

Opis Slike 3.4.1: Za ostvarivanje mikrokontrolerskog sustava korištene su sljedeće komponente: PIC18F448 mikrokontroler, 4 sedam segmentna ekrana, 4 digitalne sklopke te otpornik od 4,7 kΩ. Preko otpornika od 4,7 kΩ je spojen Master Clear izvod mikrokontrolera na napon od 5V bez kojeg mikrokontroler nebi

izvršavao svoje funkcije. Port C određen je kao izlazni port mikrokontrolera i koristi se za paralelno upravljanje sa svih 4 sedam segmentna ekrana, dok se preko porta D, koji je također izlazni port, zasebno upravlja sa 4 sklopke koje nam služe kao „ON-OFF“ tipke za sedam segmentne ekrane.

MicroPascal kod:

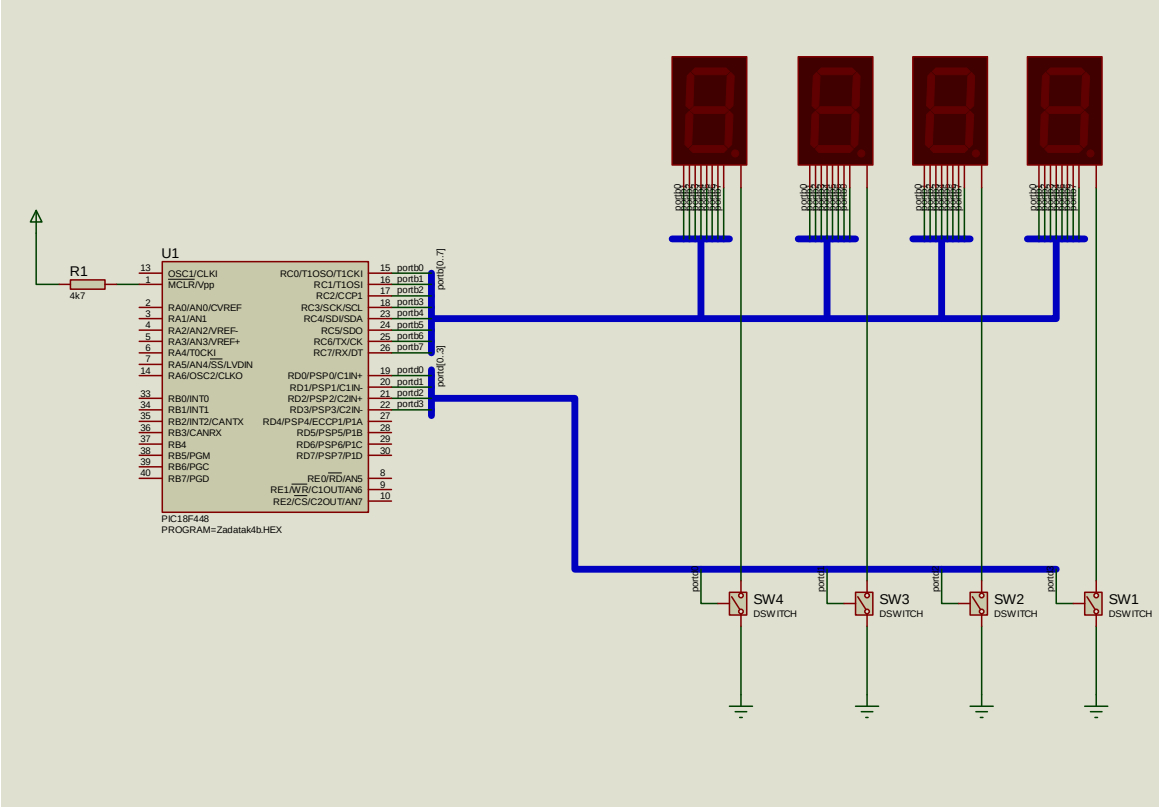
```
program Zadatak4a;                  //naziv programa
 function mask(broj:byte): byte;    //određivanje parametara funkcije mask
  begin                             //početak funkcije mask
   case broj of                     //početak case funkcije
        0: result:= 63;             //1.slučaj: dodijeljivanje vrijednosti varijabli result
        1: result:= 6;              //2.slučaj: dodijeljivanje vrijednosti varijabli result
        2: result:= 91;             //3.slučaj: dodijeljivanje vrijednosti varijabli result
        3: result:= 79;             //4.slučaj: dodijeljivanje vrijednosti varijabli result
        4: result:= 102;            //5.slučaj: dodijeljivanje vrijednosti varijabli result
        5: result:= 109;            //6.slučaj: dodijeljivanje vrijednosti varijabli result
        6: result:= 125;            //7.slučaj: dodijeljivanje vrijednosti varijabli result
        7: result:= 7;              //8.slučaj: dodijeljivanje vrijednosti varijabli result
        8: result:= 127;            //9.slučaj: dodijeljivanje vrijednosti varijabli result
        9: result:= 111;            //10.slučaj: dodijeljivanje vrijednosti varijabli result
  end;                              //kraj case funkcije
 end;                               //kraj funkcije mask
begin                               //početak programa
 trisc:=0;                          //dodijeljivanje vrijednosti trisC registru
 trisd:=0;                          //dodijeljivanje vrijednosti trisD registru
  repeat                            //početak repeat until petlje
   portc:=mask(1);                  //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
   portd.3:=1;                      //dodijeljivanje vrijednosti četvrtom izvodu porta D
   delay_ms(4);                     //kašnjenje
   portd.3:=0;                      //dodijeljivanje vrijednosti četvrtom izvodu porta D
    portc:=mask(1);                 //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
    portd.2:=1;                     //dodijeljivanje vrijednosti trećem izvodu porta D
    delay_ms(4);                    //kašnjenje
    portd.2:=0;                     //dodijeljivanje vrijednosti trećem izvodu porta D
     portc:=mask(1);                //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
     portd.1:=1;                    //dodijeljivanje vrijednosti drugom izvodu porta D
     delay_ms(4);                   //kašnjenje
     portd.1:=0;                    //dodijeljivanje vrijednosti drugom izvodu porta D
      portc:=mask(1);               //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
      portd.0:=1;                   //dodijeljivanje vrijednosti prvom izvodu porta D
      delay_ms(4);                  //kašnjenje
      portd.0:=0;                   //dodijeljivanje vrijednosti prvom izvodu porta D
  until 1=2;                        //kraj repeat until petlje
end.                                //kraj programa
```

Objašnjenje koda: Isto kao i u prethodnom zadatku radi lakšeg upravljanja sedam segmentnim ekranom uvodi se nova funkcija „mask“ preko koje će se u daljnjem kodu dodijeljivati vrijednost portu C. Preko tris registara određuju se karakteri pojedinih portova i to port C kao izlazni (njime upravljamo sedam

segmentne ekrane) i port D kao izlazni (njime upravljamo digitalne sklopke). Program funkcionira na sljedeći način: Unutar beskonačne „repeat until“ petlje prvo se određuje vrijednost porta C, a zatim se onim istim načinom konstantnog prikazivanja iz prethodnog zadatka prikazuje ta vrijednost, no taj cjelokupni proces se mora ponoviti onoliko puta koliko je sedam segmentnih ekrana u sustavu (u ovom slučaju 4), te je u skladu s tim potrebno i isto toliko upravljačkih izvoda za pojedinu digitalnu sklopku (portd.0 – portd.3).

3.4.2. Izvedba četveroznamenkastog brojača

Potrebno je napraviti program koji će na 4 sedam segmentna ekrana ispisivati stanja brojača koji broji od 0 do 9999.



Slika 3.4.2: *Proteus design* rješenja mikrokontrolerskog sustava iz četvrtog zadatka (podzadatak b)

Opis Slike 3.4.2: Za ostvarivanje mikrokontrolerskog sustava korištene su sljedeće komponente: PIC18F448 mikrokontroler, 4 sedam segmentna ekrana, 4 digitalne sklopke te otpornik od 4,7 kΩ. Preko otpornika od 4,7 kΩ je spojen Master Clear izvod mikrokontrolera na napon od 5V bez kojeg mikrokontroler nebi izvršavao svoje funkcije. Port C određen je kao izlazni port mikrokontrolera i koristi se za paralelno upravljanje sa svih 4 sedam segmentna ekrana, dok se preko porta D, koji je također karakteriziran kao izlazni port, zasebno upravlja sa 4 digitalne sklopke koje nam služe kao „ON-OFF“ tipke za sedam segmentne ekrane.

MicroPascal kod:

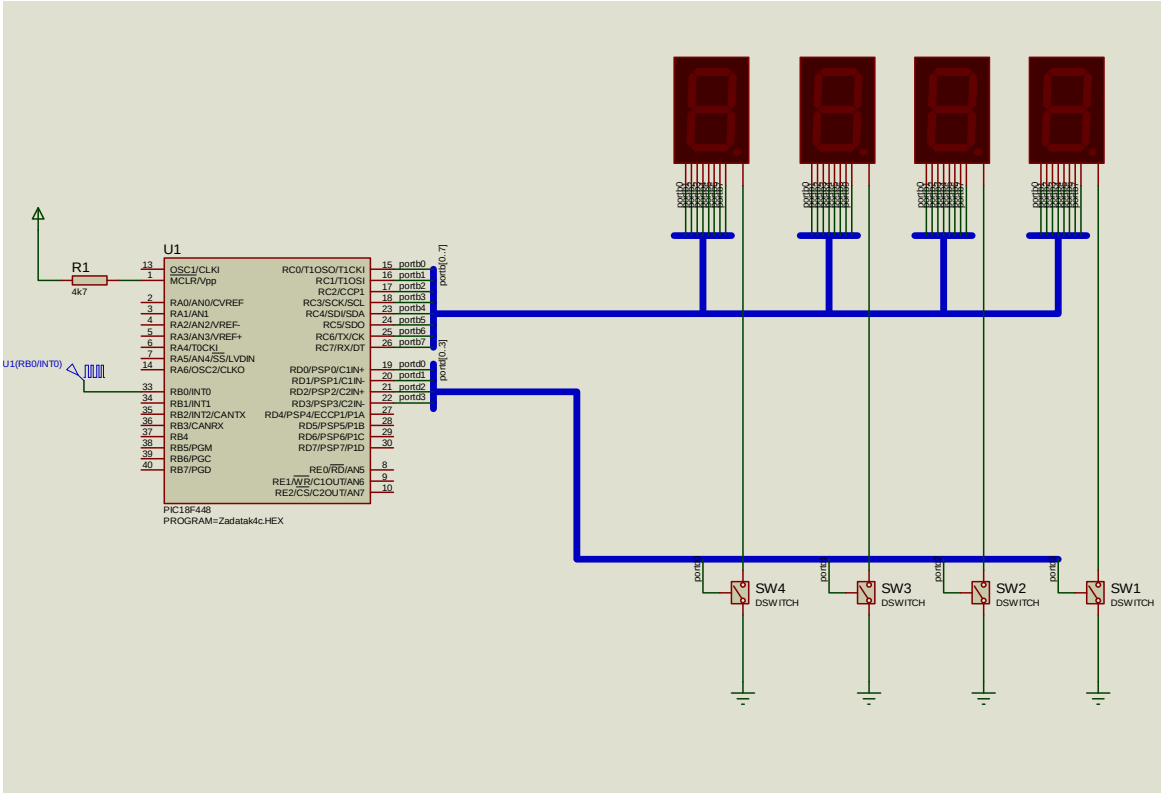
```
program Zadatak4b;                  //naziv programa
 function mask(broj:byte): byte;    //određivanje parametara funkcije mask
  begin                             //početak funkcije mask
   case broj of                     //početak case funkcije
        0: result:= 63;             //1.slučaj: dodijeljivanje vrijednosti varijabli result
        1: result:= 6;              //2.slučaj: dodijeljivanje vrijednosti varijabli result
        2: result:= 91;             //3.slučaj: dodijeljivanje vrijednosti varijabli result
        3: result:= 79;             //4.slučaj: dodijeljivanje vrijednosti varijabli result
        4: result:= 102;            //5.slučaj: dodijeljivanje vrijednosti varijabli result
        5: result:= 109;            //6.slučaj: dodijeljivanje vrijednosti varijabli result
        6: result:= 125;            //7.slučaj: dodijeljivanje vrijednosti varijabli result
        7: result:= 7;              //8.slučaj: dodijeljivanje vrijednosti varijabli result
        8: result:= 127;            //9.slučaj: dodijeljivanje vrijednosti varijabli result
        9: result:= 111;            //10.slučaj: dodijeljivanje vrijednosti varijabli result
  end;                              //kraj case funkcije
 end;                               //kraj funkcije mask
var tisucice: byte;                 //deklariranje varijable tisucice tipa byte
var stotice: byte;                  //deklariranje varijable stotice tipa byte
var desetice: byte;                 //deklariranje varijable desetice tipa byte
var jedinice: byte;                 //deklariranje varijable jedinice tipa byte
begin                               //početak programa
 trisc:=0;                          //dodijeljivanje vrijednosti trisC registru
 trisd:=0;                          //dodijeljivanje vrijednosti trisD registru
    for tisucice:= 0 to 9 do        //određivanje parametara for petlje
     begin                          //početak for petlje
      for stotice:= 0 to 9 do       //određivanje parametara for petlje
       begin                        //početak for petlje
        for desetice:= 0 to 9 do    //određivanje parametara for petlje
         begin                      //početak for petlje
          for jedinice:= 0 to 9 do  //određivanje parametara for petlje
           begin                    //početak for petlje
  portc:=mask(jedinice);            //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
  portd.3:=1;                       //dodijeljivanje vrijednosti četvrtom izvodu porta D
  delay_ms(4);                      //kašnjenje
  portd.3:=0;                       //dodijeljivanje vrijednosti četvrtom izvodu porta D
```

```
   portc:=mask(desetice);          //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
   portd.2:=1;                     //dodijeljivanje vrijednosti trećem izvodu porta D
   delay_ms(4);                    //kašnjenje
   portd.2:=0;                     //dodijeljivanje vrijednosti trećem izvodu porta D
    portc:=mask(stotice);          //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
    portd.1:=1;                    //dodijeljivanje vrijednosti drugom izvodu porta D
    delay_ms(4);                   //kašnjenje
    portd.1:=0;                    //dodijeljivanje vrijednosti drugom izvodu porta D
     portc:=mask(tisucice);        //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
     portd.0:=1;                   //dodijeljivanje vrijednosti prvom izvodu porta D
     delay_ms(4);                  //kašnjenje
     portd.0:=0;                   //dodijeljivanje vrijednosti prvom izvodu porta D
   end;                            //kraj for petlje
    end;                           //kraj for petlje
      end;                         //kraj for petlje
       end;                        //kraj for petlje
end.                               //kraj programa
```

Objašnjenje koda: Radi lakšeg upravljanja sedam segmentnim ekranom uvodi se nova funkcija „mask“ preko koje će se u daljnjem kodu dodijeljivati vrijednost portu C. Preko tris registara određuju se karakteri pojedinih portova i to port C kao izlazni (njime upravljamo sedam segmentne ekrane) i port D kao izlazni (njime upravljamo digitalne sklopke). Program funkcionira na sljedeći način: Unutar četiri for petlje (od kojih svaka predstavlja zasebnu znamenku broja kojeg prkazujemo) prvo se određuje vrijednost porta C, a zatim se onim istim načinom konstantnog prikazivanja iz prethodnog zadatka prikazuje ta vrijednost, no taj cijelokupni proces se mora ponoviti onoliko puta koliko je sedamsegmentnih ekrana u sustavu (u ovom slučaju 4), te je u skladu s tim potrebno i isto toliko upravljačkih izvodova za pojedinu digitalnu sklopku (portd.0 – portd.3).

3.4.3. Izvedba zapornog sata pomoću četiri sedam segmentnih ekrana – 1. zadatak

Potrebno je napraviti program koji će ispisivati na 4 sedam segmentnih ekrana koliko je sekundi proteklo od njegovog pokretanja s korištenjem vanjskog signala za generiranje takta od jedne sekunde.



Slika 3.4.3: *Proteus design* rješenja mikrokontrolerskog sustava iz četvrtog zadatka (podzadatak c)

Opis Slike 3.4.3: Uspoređujući ovaj sa mikrokontrolerskim sustavom i prethodnog primjera može se zamijetiti samo jedna razlika, a to je prisustvo još jedne nove komponente, pulsnog generatora napona koji se koristi kao vanjski signal takta zahtjevan u zadatku. Dakle, na port C, koji je karakteriziran kao izlazni port, spojeni su sedam segmentni ekrani, na port D, također izlazni port, spojene su digitalne sklopke („ON-OFF“ sklopke), a na port B, ulazni port, spojen je pulsni generator napona. Naravno Master Clear izvod mikrokontrolera spojen je preko otpornika od 4,7 kΩ na napon od 5V kao i u svim dosadašnjim sustavima što je neophodno za rad mikrokontrolera.

MicroPascal kod:

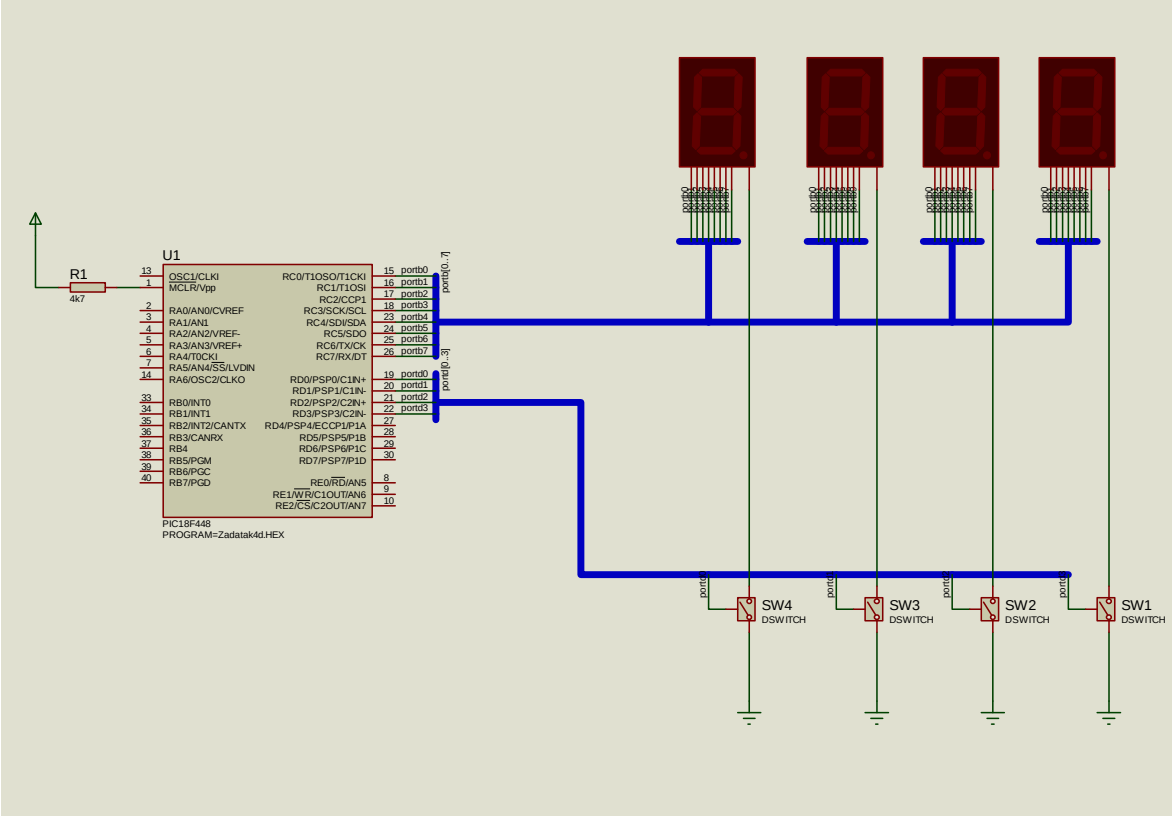
```
program Zadatak4c;                    //naziv programa
 function mask(broj:byte): byte;      //određivanje parametara funkcije mask
  begin                               //početak funkcije mask
   case broj of                       //početak case funkcije
        0: result:= 63;               //1.slučaj: dodijeljivanje vrijednosti varijabli result
        1: result:= 6;                //2.slučaj: dodijeljivanje vrijednosti varijabli result
        2: result:= 91;               //3.slučaj: dodijeljivanje vrijednosti varijabli result
        3: result:= 79;               //4.slučaj: dodijeljivanje vrijednosti varijabli result
        4: result:= 102;              //5.slučaj: dodijeljivanje vrijednosti varijabli result
        5: result:= 109;              //6.slučaj: dodijeljivanje vrijednosti varijabli result
        6: result:= 125;              //7.slučaj: dodijeljivanje vrijednosti varijabli result
        7: result:= 7;                //8.slučaj: dodijeljivanje vrijednosti varijabli result
        8: result:= 127;              //9.slučaj: dodijeljivanje vrijednosti varijabli result
        9: result:= 111;              //10.slučaj: dodijeljivanje vrijednosti varijabli result
  end;                                //kraj case funkcije
 end;                                 //kraj funkcije mask
var tisucice: byte;                   //deklariranje varijable tisucice tipa byte
var stotice: byte;                    //deklariranje varijable stotice tipa byte
var desetice: byte;                   //deklariranje varijable desetice tipa byte
var jedinice: byte;                   //deklariranje varijable jedinice tipa byte
begin                                 //početak programa
 trisc:=0;                            //dodijeljivanje vrijednosti trisC registru
 trisd:=0;                            //dodijeljivanje vrijednosti trisD registru
 trisb:=1;                            //dodijeljivanje vrijednosti trisB registru
jedinice:=0;                          //postavljanje vrijednosti varijabli jedinice
desetice:=0;                          //postavljanje vrijednosti varijabli desetice
stotice:=0;                           //postavljanje vrijednosti varijabli stotice
tisucice:=0;                          //postavljanje vrijednosti varijabli tisucice
repeat                                //početak repeat-until petlje
   if portb.0 = 0 then                //zadavanje parametara if funkcije
     begin                            //početak if funkcije
  repeat                              //početak repeat until petlje
   portc:=mask(jedinice);             //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
   portd.3:=1;                        //dodijeljivanje vrijednosti četvrtom izvodu porta D
   delay_ms(4);                       //kašnjenje
   portd.3:=0;                        //dodijeljivanje vrijednosti četvrtom izvodu porta D
    portc:=mask(desetice);            //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
    portd.2:=1;                       //dodijeljivanje vrijednosti trećem izvodu porta D
    delay_ms(4);                      //kašnjenje
    portd.2:=0;                       //dodijeljivanje vrijednosti trećem izvodu porta D
     portc:=mask(stotice);            //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
     portd.1:=1;                      //dodijeljivanje vrijednosti drugom izvodu porta D
     delay_ms(4);                     //kašnjenje
     portd.1:=0;                      //dodijeljivanje vrijednosti drugom izvodu porta D
      portc:=mask(tisucice);          //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
      portd.0:=1;                     //dodijeljivanje vrijednosti prvom izvodu porta D
      delay_ms(4);                    //kašnjenje
      portd.0:=0;                     //dodijeljivanje vrijednosti prvom izvodu porta D
     until portb.0 = 1;                                   //kraj repeat until petlje
              end;                                        //kraj if funkcije
   if portb.0 = 1 then                                    //zadavanje parametara if funkcije
    begin                                                 //početak if funkcije
      jedinice := jedinice + 1;                           //povećavanje vrijednosti varijabli jedinice za 1
      if jedinice > 9 then begin desetice := desetice + 1; jedinice := 0;end;      //zadavanje parametara if funkcije, povećavanje desetica za 1
      if desetice > 9 then begin stotice := stotice + 1; desetice := 0;end;        //zadavanje parametara if funkcije, povećavanje stotica za 1
```

```
    if stotice > 9 then begin tisucice := tisucice + 1; stotice := 0;end;          //zadavanje parametara if funkcije, povećavanje tisucica za 1
    if tisucice > 9 then begin tisucice := 0;end;                                  //zadavanje parametara if funkcije
    repeat                                                                         //početak repeat until petlje
   portc:=mask(jedinice);              //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
   portd.3:=1;                         //dodijeljivanje vrijednosti četvrtom izvodu porta D
   delay_ms(4);                        //kašnjenje
   portd.3:=0;                         //dodijeljivanje vrijednosti četvrtom izvodu porta D
   portc:=mask(desetice);              //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
   portd.2:=1;                         //dodijeljivanje vrijednosti trećem izvodu porta D
   delay_ms(4);                        //kašnjenje
   portd.2:=0;                         //dodijeljivanje vrijednosti trećem izvodu porta D
    portc:=mask(stotice);              //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
    portd.1:=1;                        //dodijeljivanje vrijednosti drugom izvodu porta D
    delay_ms(4);                       //kašnjenje
    portd.1:=0;                        //dodijeljivanje vrijednosti drugom izvodu porta D
     portc:=mask(tisucice);            //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
     portd.0:=1;                       //dodijeljivanje vrijednosti prvom izvodu porta D
     delay_ms(4);                      //kašnjenje
     portd.0:=0;                       //dodijeljivanje vrijednosti prvom izvodu porta D
  until portb.0 = 0;                   //kraj repeat until petlje
 end;                                  //kraj if funkcije
until 1=2;                             //kraj repeat until petlje
          end.                         //kraj programa
```

Objašnjenje koda: Rad programa se može podijeliti u dva dijela, a segmenti su određeni sa promjenom vanjskog pulsnog izvora takta. Dok je puls takta u „nuli“ prikazuje se trenutno stanje brojila (zasebno se prikazuje svaka od varijabli: jedinice, desetice, stotice i tisucice). Kada puls poraste u „1“ vrijednost brojača (ponovno se svaka znamenka zasebno povećava dok ne dođe do vrijednosti 9 kada se ponovno resetira na 0) se povećava pa se novo stanje brojila prikazuje na sedam segmentnim ekranima.

3.4.4. Izvedba zapornog sata pomoću četiri sedam segmentnih ekrana – 2. zadatak

Potrebno je napraviti program koji će ispisivati na 4 sedam segmentna ekrana koliko je sekundi proteklo od njegovog pokretanja bez korištenja vanjskog signala za generiranje takta od jedne sekunde.



Slika 3.4.4: *Proteus design* rješenja mikrokontrolerskog sustava iz četvrtog zadatka (podzadatak d)

Opis Slike 3.4.4: Mikrokontrolerski sustav iz iz ovg primjera je jednak onome u prvom i drugom podzadatku.

MicroPascal kod:

```
program Zadatak4c;                  //naziv programa
 function mask(broj:byte): byte;    //određivanje parametara funkcije mask
  begin                             //početak funkcije mask
   case broj of                     //početak case funkcije
       0: result:= 63;              //1.slučaj: dodijeljivanje vrijednosti varijabli result
```

```
      1: result:= 6;                    //2.slučaj: dodijeljivanje vrijednosti varijabli result
      2: result:= 91;                   //3.slučaj: dodijeljivanje vrijednosti varijabli result
      3: result:= 79;                   //4.slučaj: dodijeljivanje vrijednosti varijabli result
      4: result:= 102;                  //5.slučaj: dodijeljivanje vrijednosti varijabli result
      5: result:= 109;                  //6.slučaj: dodijeljivanje vrijednosti varijabli result
      6: result:= 125;                  //7.slučaj: dodijeljivanje vrijednosti varijabli result
      7: result:= 7;                    //8.slučaj: dodijeljivanje vrijednosti varijabli result
      8: result:= 127;                  //9.slučaj: dodijeljivanje vrijednosti varijabli result
      9: result:= 111;                  //10.slučaj: dodijeljivanje vrijednosti varijabli result
  end;                                  //kraj case funkcije
 end;                                   //kraj funkcije mask
var tisucice: byte;                     //deklariranje varijable tisucice tipa byte
var stotice: byte;                      //deklariranje varijable stotice tipa byte
var desetice: byte;                     //deklariranje varijable desetice tipa byte
var jedinice: byte;                     //deklariranje varijable jedinice tipa byte
var brojac: byte;                       //deklariranje varijable jedinice tipa byte
begin                                   //početak programa
 trisc:=0;                              //dodijeljivanje vrijednosti trisC registru
 trisd:=0;                              //dodijeljivanje vrijednosti trisD registru
  repeat                                //dodijeljivanje vrijednosti prvom izvodu porta D
   jedinice := jedinice + 1;                                        //povećavanje vrijednosti varijabli jedinice
za 1
    if jedinice > 9 then begin desetice := desetice + 1; jedinice := 0;end;     //zadavanje parametara if funkcije, povećavanje
desetica za 1
    if desetice > 9 then begin stotice := stotice + 1; desetice := 0;end;       //zadavanje parametara if funkcije, povećavanje
stotica za 1
    if stotice > 9 then begin tisucice := tisucice + 1; stotice := 0;end;       //zadavanje parametara if funkcije, povećavanje
tisucica za 1
    if tisucice > 9 then begin tisucice := 0;end;                       //zadavanje parametara if funkcije
   repeat                                                               //početak repeat until petlje
       for brojac := 0 to 50 do             //dodijeljivanje vrijednosti prvom izvodu porta D
           begin                            //dodijeljivanje vrijednosti prvom izvodu porta D
            portc:=mask(jedinice);          //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
           portd.3:=1;                      //dodijeljivanje vrijednosti četvrtom izvodu porta D
           delay_ms(4);                     //kašnjenje
           portd.3:=0;                      //dodijeljivanje vrijednosti četvrtom izvodu porta D
           portc:=mask(desetice);           //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
           portd.2:=1;                      //dodijeljivanje vrijednosti trećem izvodu porta D
           delay_ms(4);                     //kašnjenje
           portd.2:=0;                      //dodijeljivanje vrijednosti trećem izvodu porta D
           portc:=mask(stotice);            //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
           portd.1:=1;                      //dodijeljivanje vrijednosti drugom izvodu porta D
           delay_ms(4);                     //kašnjenje
           portd.1:=0;                      //dodijeljivanje vrijednosti drugom izvodu porta D
           portc:=mask(tisucice);           //dodijeljivanje vrijednosti portC-u pomoću funkcije mask
           portd.0:=1;                      //dodijeljivanje vrijednosti prvom izvodu porta D
           delay_ms(4);                     //kašnjenje
           portd.0:=0;                      //dodijeljivanje vrijednosti prvom izvodu porta D
           end;                             //dodijeljivanje vrijednosti prvom izvodu porta D
       until 1=2;                           //dodijeljivanje vrijednosti prvom izvodu porta D
       end.                                 //dodijeljivanje vrijednosti prvom izvodu porta D
```

Objašnjenje koda: Budući da se u zadatku traži da mikrokontrolerski sustav broji sekunde, ali bez vanjskog pulsa takta, bilo je potrebno unutar koda prilagoditi izvršavanje procesa tako da se povećanje brojila izvrši svakih 1 sekundu. To je učinjeno tako da se pomoću for petlje iterativno 50 puta prikazivalo trenutno stanje

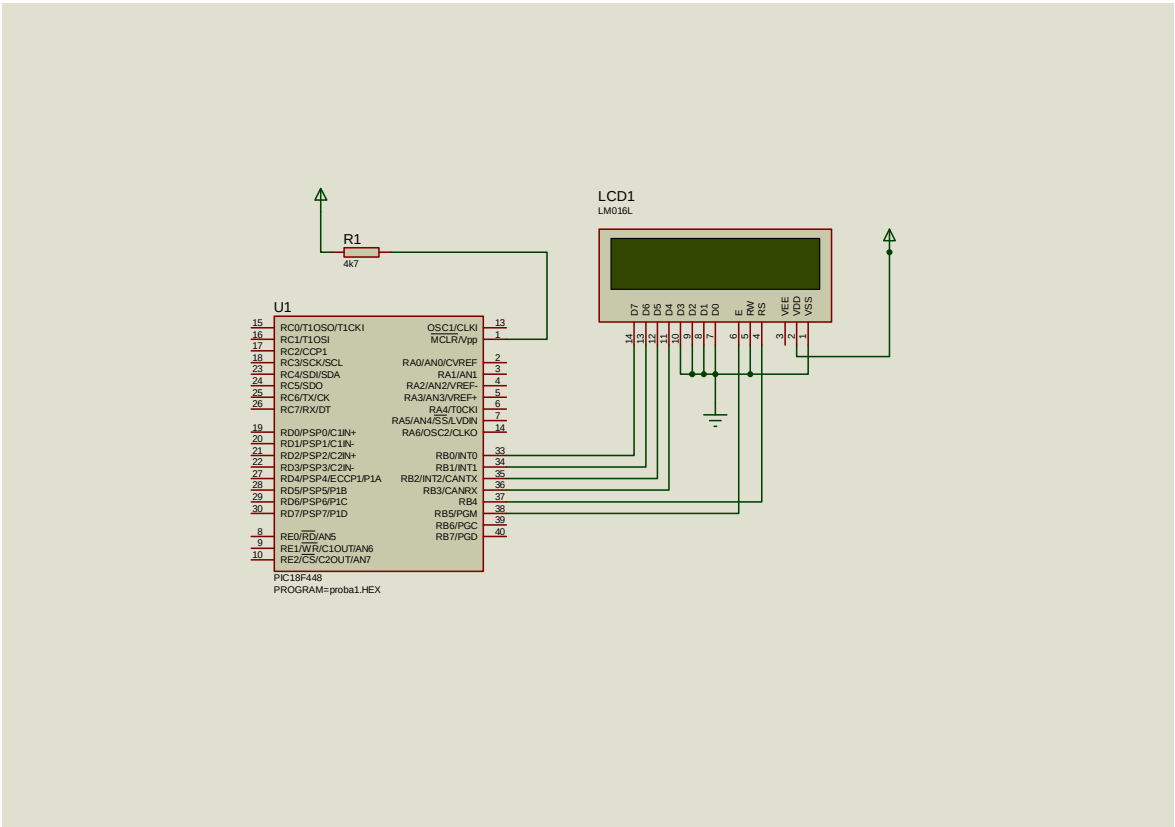
brojača (postupak koji traje približno 12ms), a vrijeme potrebno da se sve operacije izvrše je približno jednako 1 sekundi.

3.5. Upravljanje radom LCD ekrana

Potrebno je napraviti mikrokontrolerski sustav koji će prikazati mogućnosti PIC mikrokontrolera da upravlja radom alfanumeričkog LCD ekrana dimenzija 16x2.

3.5.1. Ispis statičkog teksta na LCD ekranu

Potrebno je napraviti program koji će ispisati neki proizvoljan statički tekst na LCD ekranu.



Slika 3.5.1: *Proteus design* rješenja mikrokontrolerskog sustava iz petog zadatka (podzadatak a)

Opis Slike 3.5.1: Komponente koje su potrebne da bi se mikrokontrolerski sustav iz podzadatka a) ostvario su PIC18F448 mikrokontroler, otpornik od 4,7 kΩ te alfanumerički (prikazuje brojke i slova) LCD ekran. Master Clear izvod je naravno preko otpornika od 4,7 kΩ spojen na napon od 5V, a LCD ekranom upravlja se preko porta B koji je u ovom slučaju karakterom izlazni port. Način na koji se spaja LCD ekran na mikrokontroler je proizvoljan, a način na koji se upravlja sa LCD ekranom pročitan je iz tvorničkih podataka za konkretni LCD ekran.

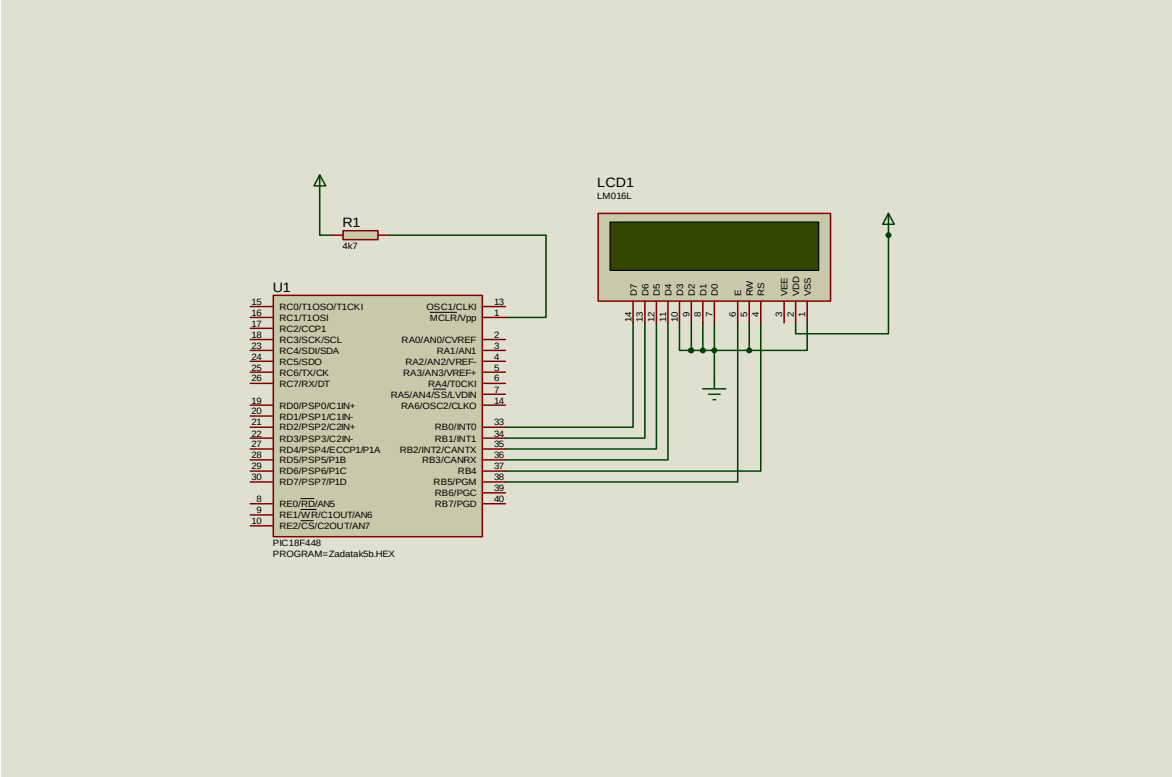
MicroPascal kod:

```
program Zadatak5a;                                  //naziv programa
var text: array[20] of char;                        //deklaracija varijable text tipa array of char
begin                                               //početak programa
  Lcd_Config(PORTB,0,1,2,3,PORTB,4,7,5);            //određivanje načina spajanja LCD ekrana
  Lcd_Cmd(LCD_CURSOR_OFF);                          //isključivanje kursora na LCD ekranu
  text := 'mikro';                                  //dodijeljivanje sadržaja varijabli text
  Lcd_Out(1, 1, text);                              //prikazivanje varijable text na LCD ekranu
end.                                                //kraj programa
```

Objašnjenje koda: Na LCD ekranu se mogu prikazivati varijable isključivo tipa string i char, pa je u skladu s tim pravilom deklarirana varijabla text koja je tipa polja char-ova i može „u sebe“ spremiti 20 znakova (bilo slova ili brojki). Naredbom Lcd_config daju se upute mikrokontroleru na koji način je spojen pojedini upravljački izvod LCD ekrana. Proizvoljni tekst na ekranu se vrlo lako moze ispisati i to na način da se poruka koja se želi ispisati prvo dodijeli varijabli text (imajući na umu njen „kapacitet“), a zatim se uz pomoć naredbe Lcd_out ispisuje zadana poruka na ekranu.

3.5.2. Izvedba zapornog sata pomoću LCD ekrana

Potrebno je napraviti program koji će na LCD ekranu ispisivati brojač od 0 do 9999 s povećanjem svakih 500 ms.



Slika 3.5.2: *Proteus design* rješenja mikrokontrolerskog sustava iz petog zadatka (podzadatak b)

Opis Slike 3.5.2: Mikrokontrolerski sustav je identičan onome iz prethodnog podzadatka odnosno za njegovo ostvarenje korišten je PIC18F448 mikrokontroler, otpornik od 4,7 kΩ te alfanumerički LCD ekran. Način spajanja LCD ekrana je identičan kao i u prošlom primjeru, dok je isto tako Master Clear izvod preko otpornika od 4,7 kΩ spojen na napon od 5V.

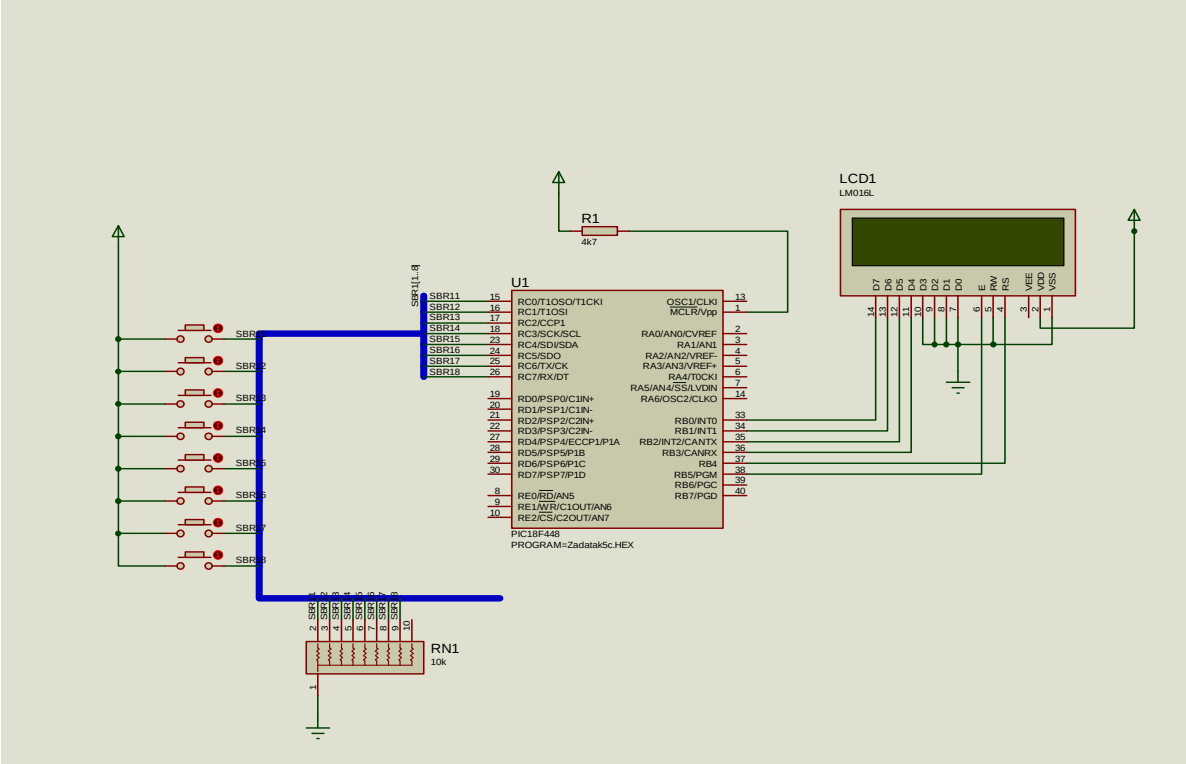
MicroPascal kod:

```
program Zadatak5b;                          //naziv programa
var text: string[6];                        //deklaracija varijable text tipa string
var brojac: integer;                        //deklaracija varijable brojac tipa integer
begin                                       //početak programa
  Lcd_Config(PORTB,0,1,2,3,PORTB,4,7,5);    //određivanje načina spajanja LCD ekrana
  Lcd_Cmd(LCD_CURSOR_OFF);                  //isključivanje kursora na LCD ekranu
  for brojac := 0 to 9999 do                //određivanje parametara for petlje
  begin                                     //početak for petlje
  IntToStr(brojac,text);                            //konverzija integer u string
  Lcd_Out(1, 1, text);                      //prikazivanje varijable text na LCD ekranu
  delay_ms(500);                            //kašnjenje
  end;                                      //kraj for petlje
end.                                        //kraj programa
```

Objašnjenje koda: U ovom slučaju je zbog mogućnosti konverzije integer u string varijabla brojac deklarirana kao integer, varijabla pomoću koje će se prikazivati podaci na ekranu je tipa string. Pomoću Lcd_config naredbe dajemo instrukcije mikrokontroleru na koji način su spojeni pojedini izvodi LCD ekrana. Brojač simuliramo pomoću for-petlje, a da bi se brojanje prikazalo na ekranu iskorištena je ugrađena funkcija IntToStr koja radi konverziju varijable brojač tipa integer i sprema ju u varijablu text tipa string koju je moguće prikazati na LCD ekranu. Poruka se na ekran ispisuje pomoću naredbe Lcd_out(). Kašnjenje od 500 ms je dodano zbog zahtjeva u zadatku.

3.5.3. Heksadecimalni ispis vrijednosti ulaznog porta na LCD ekran

Potrebno je napraviti program koji će na ekranu ispisivati stanje jednog odabranog ulaznog porta ali u heksadekadskom obliku, na ulazni port u Proteusu stavite tipkala.



Slika 3.5.3: *Proteus design* rješenja mikrokontrolerskog sustava iz petog zadatka (podzadatak c)

Opis Slike 3.5.3: Mikrokontrolerski sustav u ovom slučaju je u biti nadograđen u odnosu na dva sustava iz prethodnih podzadataka. Na port C mikrokontrolera spojeno je 8 digitalnih tipkala (port C je u ovom slučaju ulazni port) koja služe kao ulazni podatak koji će se prikazivati na LCD ekranu. Tipkala su spojena na masu preko „otporničke mreže“ koji obnaša funkciju „pull down“ otpornika. Izlazna komponenta (LCD ekran) identično je spojena kao i u prethodnim slučajevima kao i Master Clear izvod mikrokontrolera.

MicroPascal kod:

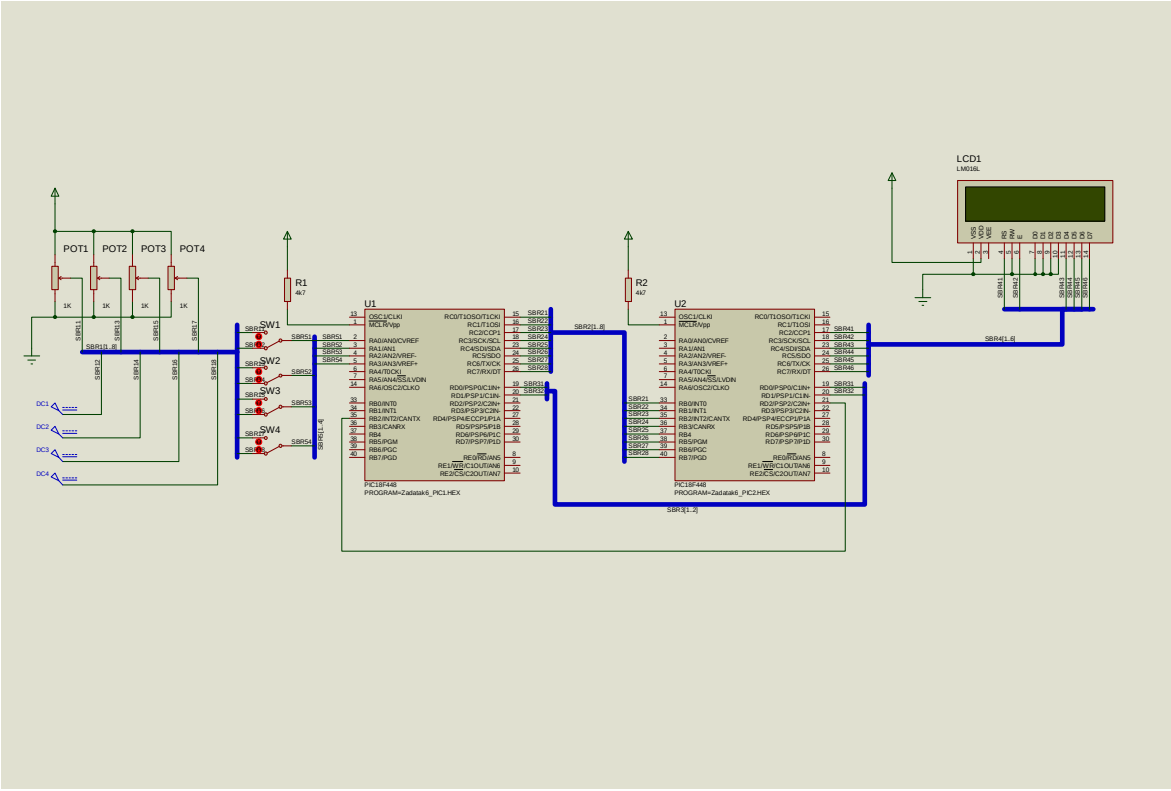
```
program Zadatak5c;                                          //naziv programa
var text: string[6];                                        //dekalracija varijable text tipa string
var broj1,broj2: byte;                                      //deklaracija varijabli broj1 i broj2 tipa byte
var broj1a,broj2a: string[6];                               //deklaracija varijabli broj1a i broj2a tipa
string
begin                                                       //početak programa
  trisc:=255;                                               //određivanje vrijednosti trisC registra
  Lcd_Config(PORTB,0,1,2,3,PORTB,4,7,5);                    //određivanje načina spajanja LCD ekrana
  Lcd_Cmd(LCD_CURSOR_OFF);                                  //isključivanje kursora na LCD ekranu
repeat                                                      //početak repeat until petlje
  broj1 := portc.0 * 1 + portc.1 * 2 + portc.2 * 4 + portc.3 * 8;   //dodijeljivanje vrijednosti varijabli broj1
  broj2 := portc.4 * 1 + portc.5 * 2 + portc.6 * 4 + portc.7 * 8;   //dodijeljivanje vrijednosti varijabli broj2
   case broj1 of                                            //početak case funkcije
     0: broj1a := '0';                                      //1.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     1: broj1a := '1';                                      //2.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     2: broj1a := '2';                                      //3.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     3: broj1a := '3';                                      //4.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     4: broj1a := '4';                                      //5.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     5: broj1a := '5';                                      //6.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     6: broj1a := '6';                                      //7.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     7: broj1a := '7';                                      //8.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     8: broj1a := '8';                                      //9.slučaj: dodijeljivanje vrijednosti varijabli
broj1a
     9: broj1a := '9';                                      //10.slučaj: dodijeljivanje vrijednosti varijabli broj1a
     10: broj1a := 'A';                                     //11.slučaj: dodijeljivanje vrijednosti varijabli broj1a
     11: broj1a := 'B';                                     //12.slučaj: dodijeljivanje vrijednosti varijabli broj1a
     12: broj1a := 'C';                                     //13.slučaj: dodijeljivanje vrijednosti varijabli broj1a
     13: broj1a := 'D';                                     //14.slučaj: dodijeljivanje vrijednosti varijabli broj1a
     14: broj1a := 'E';                                     //15.slučaj: dodijeljivanje vrijednosti varijabli broj1a
     15: broj1a := 'F';                                     //16.slučaj: dodijeljivanje vrijednosti varijabli broj1a
end;                                                        //kraj case funkcije
  case broj2 of                                             //početak case funkcije
     0: broj2a := '0';                                      //1.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
     1: broj2a := '1';                                      //2.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
     2: broj2a := '2';                                      //3.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
     3: broj2a := '3';                                      //4.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
     4: broj2a := '4';                                      //5.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
     5: broj2a := '5';                                      //6.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
     6: broj2a := '6';                                      //7.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
     7: broj2a := '7';                                      //8.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
```

```
    8: broj2a := '8';                          //9.slučaj: dodijeljivanje vrijednosti varijabli
broj2a
    9: broj2a := '9';                          //10.slučaj: dodijeljivanje vrijednosti varijabli broj2a
    10: broj2a := 'A';                         //11.slučaj: dodijeljivanje vrijednosti varijabli broj2a
    11: broj2a := 'B';                         //12.slučaj: dodijeljivanje vrijednosti varijabli broj2a
    12: broj2a := 'C';                         //13.slučaj: dodijeljivanje vrijednosti varijabli broj2a
    13: broj2a := 'D';                         //14.slučaj: dodijeljivanje vrijednosti varijabli broj2a
    14: broj2a := 'E';                         //15.slučaj: dodijeljivanje vrijednosti varijabli broj2a
    15: broj2a := 'F';                         //16.slučaj: dodijeljivanje vrijednosti varijabli broj2a
end;                                           //kraj case funkcije
  Lcd_Out(1, 1, broj1a);                       //prikazivanje varijable broj1a na LCD
ekranu
 Lcd_Out(2,1, broj2a);                         //prikazivanje varijable broj2a na LCD
ekranu
until 1=2;                                     //kraj repeat until petlje
end.                                           //kraj programa
```

Objašnjenje koda: Program funkcionira na sljedeći način: Budući da stanja na tipkalima se očituju kao 8-bitni binarni broj na izvodovima portC-a prvo je potrebno taj ulazni podatak „podijeliti“ u dva 4-bitna binarna broja, dobiti njihove dekadske vrijednosti (broj1 i broj 2 varijable tipa byte), a zatim ih pomoću case funkcije „prebaciti“ u heksadekadski oblik i u nove varijable u koje se mogu zapisivati heksadekadski brojevi (broj1a i broj2a varijable tipa string) što je i učinjeno gore navedenim kodom. Novonastali heksadekadski brojevi se prikazuju pomoću naredbe Lcd_out.

3.6. Paralelna komunikacija dva mikrokontrolera

Potrebno je napraviti mikrokontrolerski sustav koji će pokazati osnovne mogćnosti komunikacije dva mikrokontrolera u paralelnom režimu rada. Komunikacija je jednosmjerna. Podatkovni dio sabirnice sastoji se od 8 bitova. Prvi mikrokontroler mjeri analogne veličine na četiri ulazna kanala (analogne ulaze simulirati potenciometrima, voditi računa o maksimalnim ulaznim vrijednostima mikrokontrolera). Drugi mikrokontroler prikazuje dobivene podatke o sva četiri mjerenja na LCD ekranu. Mjerenja se šalju svakih 5 sekundi. Veličinu upravljačkog dijela sabirnice odrediti samostalno u skladu s osmišljenim protokolom komunikacije.



Slika 3.6: *Proteus design* rješenja mikrokontrolerskog sustava iz šestog zadatka

Opis Slike 3.6: Za ostvarivanje mikrokontrolerskog sustava sa slike potrebne su sljedeće komponente: dva PIC18F448 mikrokontrolera, alfanumerički LCD ekran, 2 otpornika od 4,7 kΩ te 4 izvora analognog signala (na slici su izvedbe izvora pomoću DC izvora i potenciometara, a njihovo korištenje se kontrolira sa sklopkama). Sustav funkcionira tako da prvi mikrokontroler očitava i pretvara

analogne signale u digitalni oblik koristeći ugrađeni modula za A/D pretvorbu te zatim podatke o mjerenjima pravilno adresira i šalje na drugi mikrokontroler koji je zadužen za prikazivanje podataka na LCD ekranu. Signal takta je simuliran u prvom mikrokontroleru i njime se određuje tijek odrade pojedinih akcija.

MicroPascal kod (Mikrokontroler 1):

```
program Zadatak6_PIC;                          //naziv programa
 var tmp : word;                               //deklaracija varijable tmp tipa word
  begin                                        //početak programa
   trisc := 0;                                 //dodijeljivanje vrijednosti trisC registru
   trisa := 255;                               //dodijeljivanje vrijednosti trisA registru
   trisd := 0;                                 //dodijeljivanje vrijednosti trisD registru
   trisb := 0;                                 //dodijeljivanje vrijednosti trisB registru
   adcon1 := 0;                                //dodijeljivanje vrijednosti adcon1 registru
   cmcon := 0x07;                              //dodijeljivanje vrijednosti cmcon registru
    repeat                                     //početak repeat until petlje
     delay_ms(1000);                           //kašnjenje
     tmp := Adc_read(0);                       //očitavanje i digitalna pretvorba analognog signala
     tmp := tmp/4;                             //skaliranje vrijednosti varijable tmp
     portc := tmp;                             //postavljanje vrijednosti portC-a
     portd.0 := 0;                             //postavljanje vrijednosti nultom izvodu
portD-a
     portd.1 := 0;                             //postavljanje vrijednosti prvom izvodu
portD-a
     delay_ms(5);                              //kašnjenje
     portb.2 := 1;                             //postavljanje vrijednosti drugom izvodu
portB-a
     delay_ms(50);                             //kašnjenje
     portb.2 := 0;                             //postavljanje vrijednosti drugom izvodu
portB-a
     tmp := Adc_read(1);                       //očitavanje i digitalna pretvorba analognog signala
     tmp := tmp/4;                             //skaliranje vrijednosti varijable tmp
     portc := tmp;                             //postavljanje vrijednosti portC-a
     portd.0 := 1;                             //postavljanje vrijednosti nultom izvodu
portD-a
     portd.1 := 0;                             //postavljanje vrijednosti prvom izvodu
portD-a
     delay_ms(5);                              //kašnjenje
     portb.2 := 1;                             //postavljanje vrijednosti drugom izvodu
portB-a
     delay_ms(50);                             //kašnjenje
     portb.2 := 0;                             //postavljanje vrijednosti drugom izvodu
portB-a
     tmp := Adc_read(2);                       //očitavanje i digitalna pretvorba analognog signala
     tmp := tmp/4;                             //skaliranje vrijednosti varijable tmp
     portc := tmp;                             //postavljanje vrijednosti portC-a
     portd.0 := 0;                             //postavljanje vrijednosti nultom izvodu
portD-a
     portd.1 := 1;                             //postavljanje vrijednosti prvom izvodu
portD-a
     delay_ms(5);                              //kašnjenje
```

```
    portb.2 := 1;                                   //postavljanje vrijednosti drugom izvodu
portB-a
    delay_ms(50);                                   //kašnjenje
    portb.2 := 0;                                   //postavljanje vrijednosti drugom izvodu
portB-a
    tmp := Adc_read(3);                             //očitavanje i digitalna pretvorba analognog signala
    tmp := tmp/4;                                   //skaliranje vrijednosti varijable tmp
    portc := tmp;                                   //postavljanje vrijednosti portC-a
    portd.0 := 1;                                   //postavljanje vrijednosti nultom izvodu
portD-a
    portd.1 := 1;                                   //postavljanje vrijednosti prvom izvodu
portD-a
    delay_ms(5);                                    //kašnjenje
    portb.2 := 1;                                   //postavljanje vrijednosti drugom izvodu
portB-a
    delay_ms(50);                                   //kašnjenje
    portb.2 := 0;                                   //postavljanje vrijednosti drugom izvodu
portB-a
    delay_ms(4000);                                 //kašnjenje
  until 1=2;                                        //kraj repeat until petlje
 end.                                               //kraj programa
```

Objašnjenje koda: Kod programiranja prvog mikrokontrolera vrlo je bitno da se adcon1 registar postavi u 0 jer on određuje karakter analognih ulaza na portu A. Zatim očitavanje i A/D pretvorba se jednako vrši za sva 4 analogna ulaza na sljedeći način: pomoću adc_read funkcije se zapisuje digitalni ekvivalent analognog signala (10-bitni broj) na pojedinom izvodu u varijablu unutar koda, zatim se ta varijabla skalira na 8-bitni broj (dijeli se sa 4) i šalje na izlazni port C. Pomoću izvoda portD.0 i portD.1 se adresira poslani podatak, a pomoću portB.2 izvoda se simulira signal takta. Registar cmcon se postavlja u heksadekadsku vrijednost 0x07 da bi se isključio komparator unutar mikrokontrolera i slobodno koristio portD kao izlazni port (u suprotnom sustav nebi obavljao zadanu funkciju).

MicroPascal kod (Mikrokontroler 2):

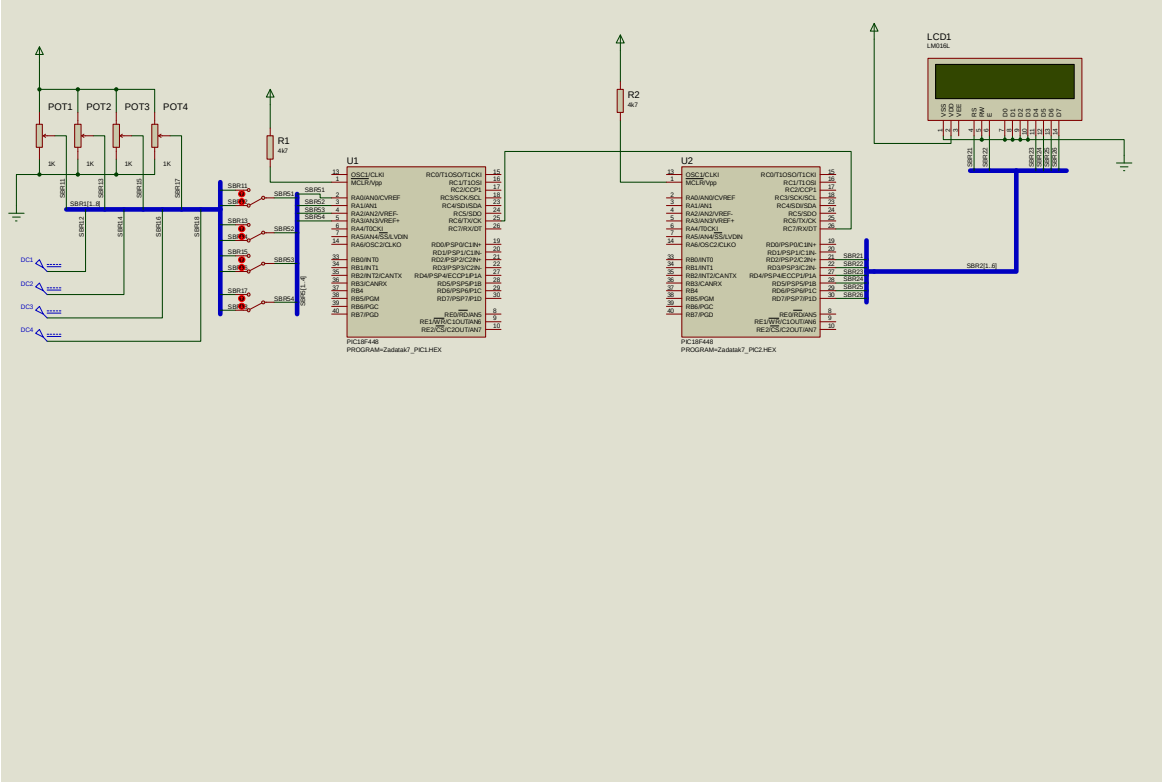
```
program Zadatak6_PIC2;                              //naziv programa
 var br1,br2,br3,br4 : string[6];                   //deklaracija varijabli br1,br2,br3 i br4 tipa
string
 var tmp,op : byte;                                 //deklaracija varijabli tmp i op tipa byte
  begin                                             //početak programa
   trisc := 0;                                      //dodijeljivanje vrijednosti trisC registru
   trisb := 255;                                    //dodijeljivanje vrijednosti trisB registru
   trisd := 255;                                    //dodijeljivanje vrijednosti trisD registru
   cmcon := 0x07;                                   //dodijeljivanje vrijednosti cmcon registru
   Lcd_init(portc);                                 //inicijalizacija alfanumeričkog LCD
ekrana
   Lcd_Cmd(LCD_CURSOR_OFF);                         //isključivanje kursora na LCD ekranu
    repeat                                          //početak repeat until petlje
```

```
    tmp := portb;                                                          //dodijeljivanje vrijednosti varijabli tmp
    op := portd.0 + portd.1 * 2;                                           //dodijeljivanje vrijednosti varijabli op
     if portd.2 = 1 then                                                   //određivanje parametara if funkcije
      begin                                                                //početak if funkcije
       if op = 0 then begin ByteToStr(tmp,br1); delay_ms(10); end;         //1. uvjetovana akcija
       if op = 1 then begin ByteToStr(tmp,br2); delay_ms(10); end;         //2. uvjetovana akcija
       if op = 2 then begin ByteToStr(tmp,br3); delay_ms(10); end;         //3. uvjetovana akcija
       if op = 3 then begin ByteToStr(tmp,br4); delay_ms(10); end;         //4. uvjetovana akcija
      end;                                                                 //kraj if funkcije
        Lcd_out(1,1,'1:');                                                 //prikazivanje na LCD ekranu
        Lcd_out(1,3,br1);                                                  //prikazivanje na LCD ekranu
        Lcd_out(1,9,'2:');                                                 //prikazivanje na LCD ekranu
        Lcd_out(1,11,br2);                                                 //prikazivanje na LCD ekranu
        Lcd_out(2,1,'3:');                                                 //prikazivanje na LCD ekranu
        Lcd_out(2,3,br3);                                                  //prikazivanje na LCD ekranu
        Lcd_out(2,9,'4:');                                                 //prikazivanje na LCD ekranu
        Lcd_out(2,11,br4);                                                 //prikazivanje na LCD ekranu
   until 1=2;                                                              //kraj repeat until petlje
end.                                                                       //kraj programa
```

Objašnjenje koda: Drugi mikrokontroler funkcionira na sljedeći način: sa ulaznog porta (portB) zapisuje se vrijednost koja je poslana sa prvog mikrokontrolera (izmjerena vrijednost) u varijablu tmp,a zatim se adresa koja je takodjer poslana sa prvog mikrokontrolera zapisuje u varijablu op u dekadskom obliku te uspoređuje varijablu op sa zadanim adresarom i u skladu s tim prikazuje podatak na točno određenom mjestu na LCD ekranu. Mikrokontroler šalje upute za prikaz vrijednosti LCD ekranu tek kad je na njegovom portD.2 izvodu “1” (signal takta).

3.7. Serijska komunikacija između dva mikrokontrolera

Ostvariti mikrokontrolerski sustav iz prethodnog zadatka koristeći serijsku komunikaciju između dva mikrokontrolera.



Slika 3.7: *Proteus design* rješenja mikrokontrolerskog sustava iz sedmog zadatka

Opis Slike 3.7: Mikrokontrolerski sustav iz prethodnog i ovog zadatka se razlikuju u tome što je u ovom slučaju za komunikaciju između dva mikrokontrolera koristi samo jedna žica (serijska komunikacija), dok se u prethodnom koristilo čak njih 11. Ulazni analogni signala su simulirani na isti način (potenciometrima i DC izvorima), a rezultati mjerenja se prikazuju također na alfanumeričkom LCD ekranu.

MicroPascal kod (Mikrokontroler 1):

```
program Zadatak7_PIC1;                                  //naziv programa
var tmp: word;                                          //deklaracija varijable tmp tipa word
var op,ch: byte;                                        //dekalracija varijabli op i ch tipa byte
begin                                                   //početak programa
  usart_init(9600);                                     //aktivacija usart modula za serijsku
komunikaciju
  adcon1:=0;                                            //dodijeljivanje vrijednosti adcon1 registru
  trisa:=255;                                           //dodijeljivanje vrijednosti trisA registru
  trisc:=0;                                             //dodijeljivanje vrijednosti trisC registru
  op:=252;                                              //dodijeljivanje vrijednosti varijabli op
  ch:=0;                                                //dodijeljivanje vrijednosti varijabli ch
  repeat                                                //početak repeat until petlje
    begin
      repeat                                            //početak repeat until petlje
        begin
          usart_write(op);                              //slanje vrijednosti varijable op preko usart modula
          delay_ms(10);                                 //kašnjenje
          tmp:=adc_read(ch);                            //očitavanje i digitalna pretvorba analognog signala
          tmp:=tmp/4;                                   //skaliranje varijable tmp
          if tmp > 251 then tmp := 251;                 //skaliranje varijable tmp
          usart_write(tmp);                             //slanje vrijednosti varijable tmp preko usart modula
          delay_ms(10);                                 //kašnjenje
          op := op+1;                                   //povećavanje vrijednosti varijabli op za 1
          ch := ch+1;                                   //povećavanje vrijednosti varijabli ch za 1
        end;                                            //kraj repeat until petlje
     until ch = 4;
      op := 252;                                        //dodijeljivanje vrijednosti varijabli op
      ch := 0;                                          //dodijeljivanje vrijednosti varijabli ch
       delay_ms(4000);                                  //kašnjenje
    end;                                                //kraj repeat until petlje
  until 1 = 2;
end.                                                    //kraj programa
```

Objašnjenje koda: Prvi mikrokontroler je zadužen za mjerenje analognih signala te adresiranje i slanje istih na drugi mikrokontroler da bi se pravilno prikazali na LCD ekranu. Za serijsku komunikaciju se koristi Usart modul unutar mikrokontrolera koji se aktivira naredbom Usart_init() (unutar zagrada se upisuje brzina prijenosa podataka u serijskoj vezi). Najjednostavniji način adresiranja mjerenja je taj da se mjerenja najprije skaliraju na 8-bitne brojeve, pa opet na vrijednosti u intervalu od 0 do 251, a posljednja četiri broja (252, 253, 254 i 255) se koriste kao adrese za pojedino mjerenje. U skladu s tim pravilima radi ovaj program . Dakle prvo se pošalje adresa mjerenja (varijabla op odnosno jedna od vrijednosti 252 - 255), zatim se izvrši samo mjerenje i skaliranje istog te se i ono šalje, a na kraju se povećava vrijednost adrese i kanala s kojeg se mjeri analogni signal (na portu A) i nastavlja se na sljedeće mjerenje. Nakon što su izvršena

mjerenja za sva četiri analogna ulaza, vrijednosti varijabli adrese (op) i kanala (ch) se resetiraju, na 252 odnosno 0 i nakon 5 s (koje je simulirano kašnjenjem) se ponovno vrše mjerenja.

MicroPascal kod (Mikrokontroler 2):

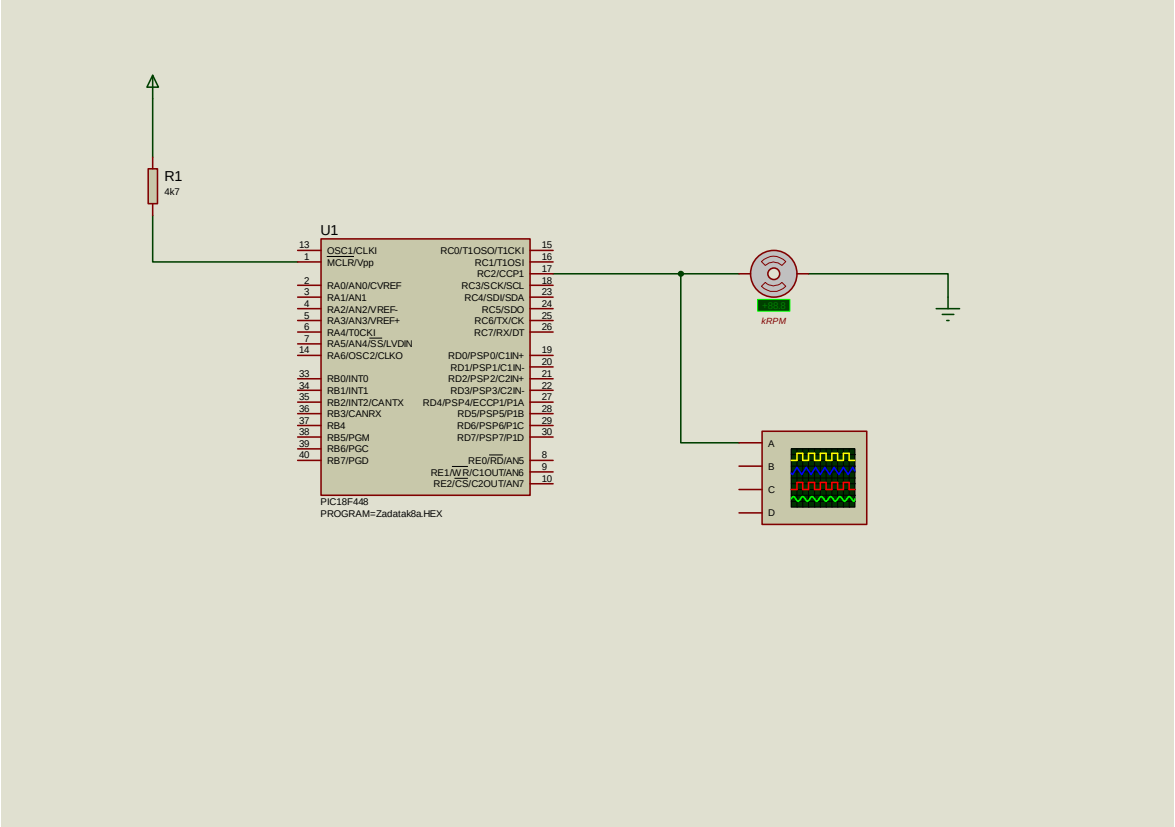
```
program Zadatak7_PIC2;                                  //naziv programa
var adresa,data: byte;                                  //deklaracija varijabli adresa i data tipa
byte
var tekst: string[6];                                   //dekalracija varijable tekst tipa string
begin                                                   //početak programa
  trisc:=255;                                           //postavljanje vrijednosti trisC registra
  Usart_Init(9600);                                     //aktivacija usart modula za serijsku
komunikaciju
  Lcd_Init(portd);                                      //inicijalizacija LCD ekrana
  Lcd_Cmd(LCD_CURSOR_OFF);                              //isključivanje kursora na LCD ekranu
  cmcon:=0x07;                                          //postavljanje vrijednosti cmcon registra
  trisd:=0;                                             //postavljanje vrijednosti trisD registra
  while true do                                         //početak while petlje
   begin
    if Usart_Data_Ready() = 1 then                      //određivanje parametara if funkcije
      begin                                             //početak if funkcije
        adresa:=usart_read;                             //dodijeljivanje vrijednosti varijabli adresa
        if adresa=252 then                              //određivanje parametara if funkcije
          begin                                         //početak if funkcije
            repeat                                      //početak repeat until petlje
            until usart_data_ready() = 1;               //kraj repeat until petlje
            if Usart_Data_Ready() = 1 then              //određivanje parametara if funkcije
              begin                                     //početak if funkcije
               data:=usart_read;                        //dodijeljivanje vrijednosti varijabli data
               bytetostr(data,tekst);                   //zapisivanje varijable data u varijablu
tekst
               lcd_out(1,1,'1:');                       //prikazivanje na LCD ekranu
               lcd_out(1,3,tekst);                      //prikazivanje na LCD ekranu
            end;                                        //kraj if funkcije
          end;                                          //kraj if funkcije
        if adresa=253 then                              //određivanje parametara if funkcije
          begin                                         //početak if funkcije
            repeat                                      //početak repeat until petlje
            until usart_data_ready() = 1;               //kraj repeat until petlje
              if Usart_Data_Ready() = 1 then            //određivanje parametara if funkcije
                begin                                   //početak if funkcije
                 data:=usart_read;                      //dodijeljivanje vrijednosti varijabli data
                 bytetostr(data,tekst);                 //zapisivanje varijable data u varijablu
tekst
                 lcd_out(1,9,'2:');                     //prikazivanje na LCD ekranu
                 lcd_out(1,11,tekst);                   //prikazivanje na LCD ekranu
                end;                                    //kraj if funkcije
          end;                                          //kraj if funkcije
        if adresa=254 then                              //određivanje parametara if funkcije
          begin                                         //početak if funkcije
```

```
            repeat                                                  //početak repeat until petlje
              until usart_data_ready() = 1;                         //kraj repeat until petlje
              if Usart_Data_Ready() = 1 then                        //određivanje parametara if funkcije
                begin                                               //početak if funkcije
                 data:=usart_read;                                  //dodijeljivanje vrijednosti varijabli data
                 bytetostr(data,tekst);                             //zapisivanje varijable data u varijablu
tekst
                 lcd_out(2,1,'3:');                                 //prikazivanje na LCD ekranu
                 lcd_out(2,3,tekst);                                //prikazivanje na LCD ekranu
                end;                                                //kraj if funkcije
            end;                                                    //kraj if funkcije
          if adresa=255 then                                        //određivanje parametara if funkcije
            begin                                                   //početak if funkcije
              repeat                                                //početak repeat until petlje
              until usart_data_ready() = 1;                         //kraj repeat until petlje
              if Usart_Data_Ready() = 1 then                        //određivanje parametara if funkcije
                begin                                               //početak if funkcije
                 data:=usart_read;                                  //dodijeljivanje vrijednosti varijabli data
                 bytetostr(data,tekst);                             //zapisivanje varijable data u varijablu
tekst
                 lcd_out(2,9,'4:');                                 //prikazivanje na LCD ekranu
                 lcd_out(2,11,tekst);                               //prikazivanje na LCD ekranu
              end;                                                  //kraj if funkcije
            end;                                                    //kraj if funkcije
        end;                                                        //kraj if funkcije
    end;                                                            //kraj while petlje
end.                                                                //kraj programa
```

Objašnjenje koda: Drugi mikrokontroler po pravilima osmišljene komunikacije, prvo učitava adresu mjerenja koje stiže preko komunikacijskog porta, zatim uspoređuje vrijednost adrese sa adresarom te prikazuje na točno određenom mjestu na LCD ekranu podatak koji mu je takodjer nakon adrese poslan preko komunikacijskog porta. Pomoću ugrađene funkcije Usart_data_ready() mikrokontroler provjerava dali mu je poslan ikoji podatak, ako mu funkcija vrati rezultat “1” znači da je podatak poslan te on nastavlja sa izvršavanjem svojih akcija. Vrlo je bitno kod inicijalizacije Usart modula da je brzina prijenosa podataka serijskom komunikacijom između dva mikrokontrolera jednaka odnosno mora upisan isti broj u funkciji Usart_init().

3.8. Upravljanje brzinom vrtnje motora

Mikrokontrolerom generirajte upravljačke signale dobivene pulsno-širinskom modulacijom (PWM) za pogon istosmjernog motora. Istosmjerni motor simulirajte odgovarajućom komponentom u Proteus simulacijskom paketu. Generirajte upravljačke signale tako da motor ubrzava od početne brzine 0 do njegove maksimalne brzine. Osciloskopom prikazati dobivene valne oblike za slučaj da motor vrti sa 60% maksimalne brzine.



Slika 3.8.1: *Proteus design* rješenja mikrokontrolerskog sustava iz osmog zadatka

Opis Slike 3.8.1: Za ostvarivanje mikrokontrolerskog sustava korišten je PIC18F448 mikrokontroler, otpornik od 4,7 kΩ, DC motor iz knjižnice komponenti unutar programa Proteus te digitalni osciloskop. Mikrokontroler preko izvoda portC.2 šalje upravljački signal prema DC motoru, a valni oblik upravljačkog signala se može pogledati na ekranu digitalnog osciloskopa.

MicroPascal kod (1):

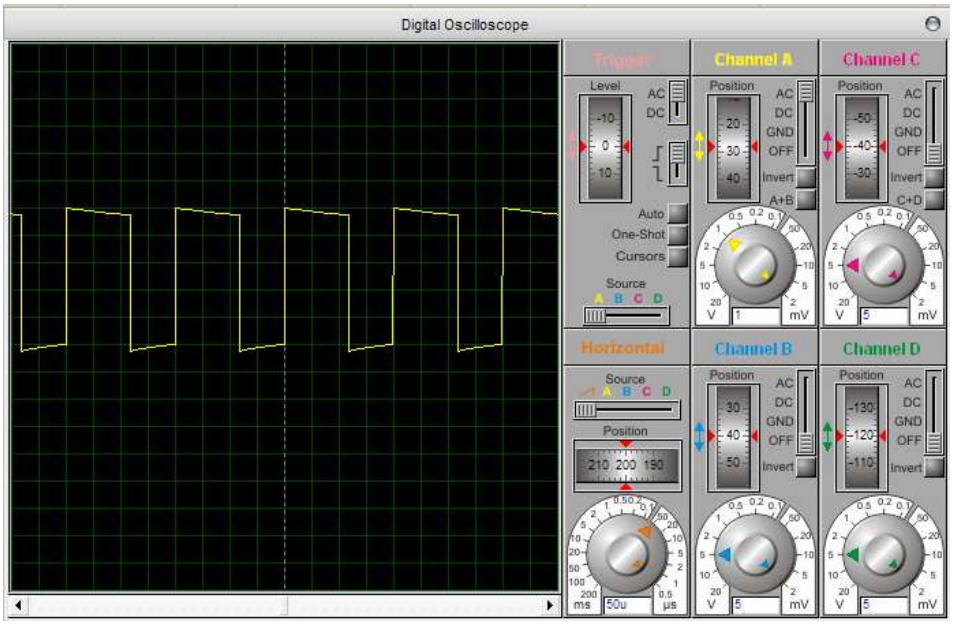
```
program Zadatak8;                  //naziv programa
var brojac: byte;                  //deklaracija varijable brojac tipa  byte
begin                              //početak programa
portc:=0;                          //dodijeljivanje vrijednosti portu C
Pwm1_Init(5000);                   //određivanje frekvencije pulsno širinske modulacije
Pwm1_Start();                      //inicijalizacija pulsmo širinske modulacije
repeat                             //početak repeat until petlje
  for brojac:= 0 to 255 do         //određivanje parametara for petlje
    begin                          //početak for petlje
     Pwm1_Set_Duty(brojac);        //određivanje parametara pulsno širinske modulacije
     Delay_ms(1);                  //kašnjenje
    end;                           //kraj for petlje
   delay_ms(1000);                 //kašnjenje
until 1=2;                         //kraj repeat until petlje
end.                               //kraj programa
```

Objašnjenje koda: Ugrađenom funkcijom Pwm1_init() se određuje frekvencija pulsno širinske modulacije, a funkcijom Pwm_start() se inicijalizira pulsno širinska modulacija. Korištenjem for petlje, konkretnije brojača unutar nje, te funkcije Pwm1_set_duty() upravljački signal se modulira te tako ubrzava DC motor od nule do maks vrijednosti.

MicroPascal kod (2):

```
program Zadatak8a;                 //naziv programa
begin                              //početak programa
portc:=0;                          //dodijeljivanje vrijednosti portC-u
Pwm1_Init(5000);                   //određivanje frekvencije pulsno širinske modulacije
Pwm1_Start();                      //inicijalizacija pulsno širinske modulacije
repeat                             //početak repeat until petlje
    Pwm1_Set_Duty(60*255/100);     //postavljanje parametara pulsno širinske modulacije
until 1=2;                         //kraj repeat until petlje
end.                               //kraj programa
```

Objašnjenje koda: Da bi se DC motor vrtio na 60% maksimalne brzine unutar funkcije Pwm1_set_duty() se mora postaviti vrijednost koja je ekvivalentna 60% maksimalne vrijednosti modulacije (255).

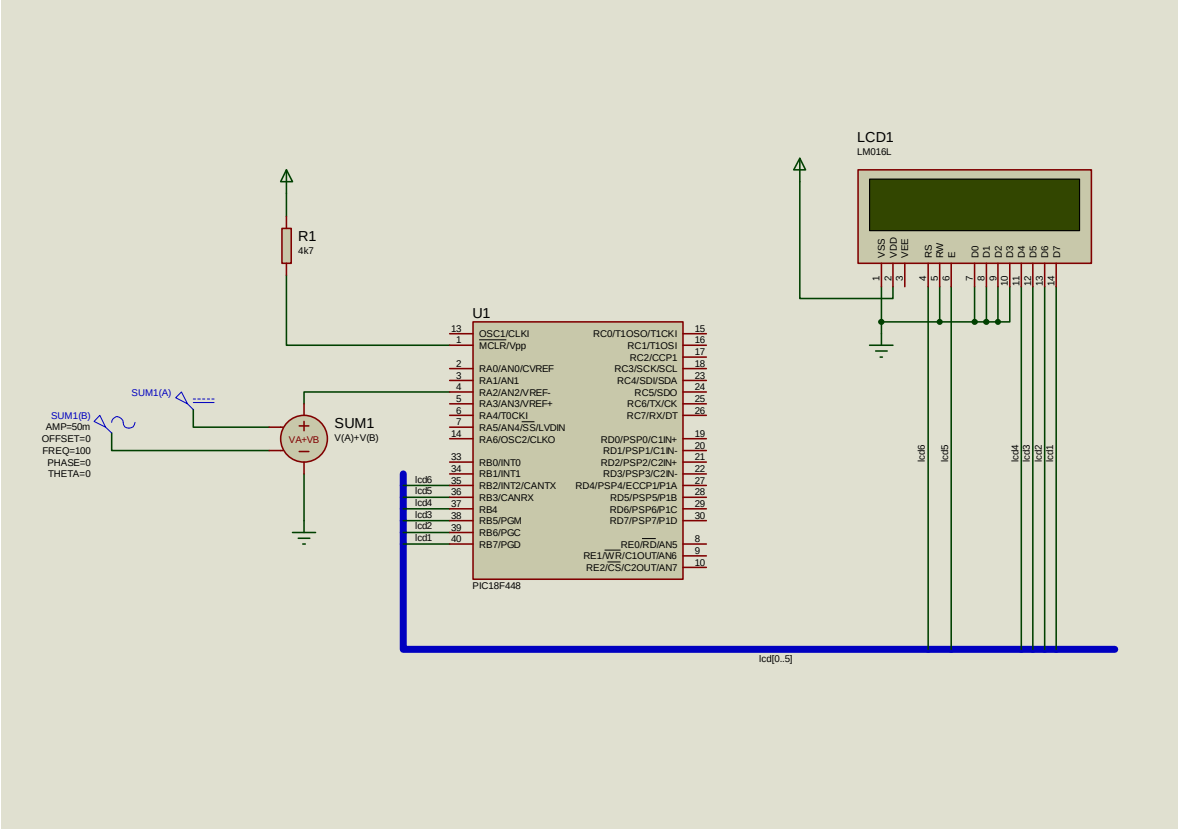


Slika 3.8.2: Prikaz valnog oblika upravljačkog signala

Opis Slike 3.8.2: Na slici je prikazan valni oblik upravljačkog signala za slučaj da se motor vrti sa 60% maksimalne brzine.

3.9. Mjerenje analogne veličine metodom usrednjavanja

Potrebno je realizirati mikrokontrolerski sustav koji će kompenzirati smetnje nastale prilikom mjerenja analognih veličina pomoću metode usrednjavanja. Mjerenja se odvijaju svakih 100ms, a ukupni rezultat mikrokontrolera se mora prikazati na priključenom LCD ekranu iterativno nakon prikupljanja 50 ulaznih podataka. Ulazni signal je istosmjernog karaktera sa superponiranom smetnjom sinusnog oblika. Frekvencija smetnje i amplitudu odaberite proizvoljno imajući u vidu cilj zadatka.



Slika 3.9: *Proteus design* rješenja mikrokontrolerskog sustava iz devetog zadatka

Opis Slike 3.9: Da bi se simulirao istosmjerni ulazni signal sa sinusnom smetnjom korištena su generator sinusnog signala, DC izvor i sumator čiji je izlaz nakraju spojen na ulazni izvod mikrokontrolera. Također su korišteni PIC18F448 mikrokontroler, otpornik od 4,7 kΩ te alfanumerički LCD ekran.

MicroPascal kod:

```
program Zadatak9;                                          //naziv programa
 var tmp: word;                                            //deklaracija varijable tmp tipa word
 var suma,brojac,tren,prosjek : word;                      //deklaracija varijabli suma,brojac,tren i prosjek tipa
byte
 var txt: string[6];                                       //deklaracija varijable txt tipa string
  begin                                                    //početak programa
   trisa:=255;                                             //postavljanje vrijednosti trisA registra
   adcon1:=0;                                              //postavljanje vrijednosti adcon1 registra
   lcd_init(portb);                                        //inicijalizacija LCD ekrana
   lcd_cmd(lcd_cursor_off);                                //isključivanje kursora na LCD ekranu
    repeat                                                 //početak repeat until petlje
      suma := 0;                                           //postavljanje vrijednosti varijabli suma
       for brojac := 1 to 50 do                            //određivanje parametara for petlje
        begin                                              //početak for petlje
          tmp := Adc_Read(2);                              //očitavanje i digitalna pretvorba analognog signala
          tren := tmp/4;                                   //skaliranje varijable tmp
          suma := suma + tren;                             //određivanje vrijednosti varijable suma
          delay_ms(100);                                   //kašnjenje
        end;                                               //kraj for petlje
        prosjek := suma/50;                                //određivanje vrijednosti varijabli prosjek
        WordToStr(prosjek, txt);                           //zapisivanje varijable prosjek u varijablu
txt
         lcd_out(1,1,'REZ:');                              //prikazivanje na LCD ekranu
         lcd_out(1,6, txt);                                //prikazivanje na LCD ekranu
  until 1=2;                                               //kraj repeat until petlje
 end.                                                      //kraj programa
```

Objašnjenje koda: Program vrlo jednostavno funkcionira. Korištenjem for petlje sa 50 iteracija se simulira traženih 50 mjerenja, dok se ukupna suma mjerenja sa svakom iteracijom povećava. Nakon što se izvrše svih 50 mjerenja ukupna suma se dijeli sa 50 i prikazuje na LCD ekranu kao porsječna ulazna vrijednost analognog signala, vrijednost sume se resetira i ponovno se vrše mjerenja.

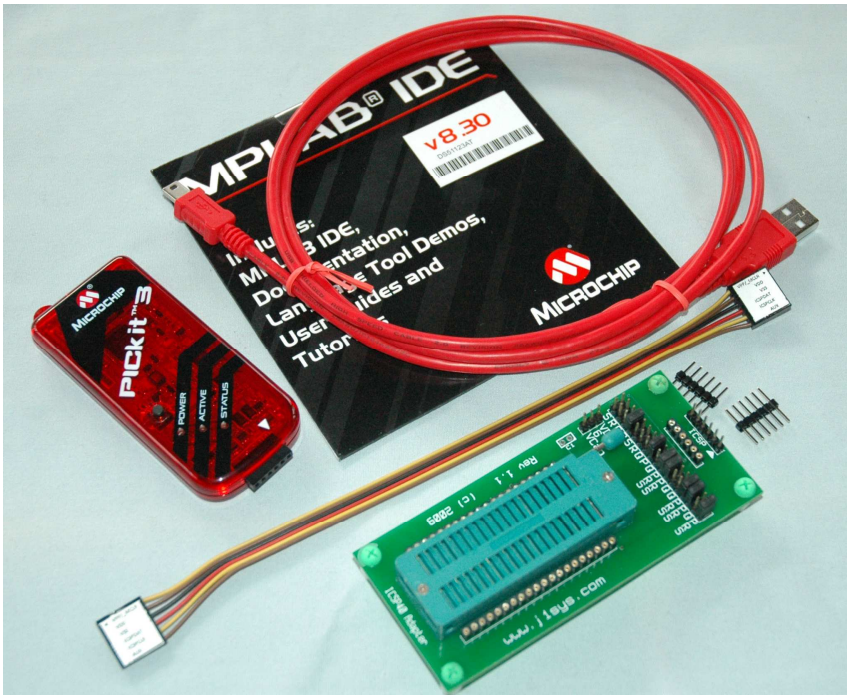
4. PIC programator

PIC programator je električni krug koji uspostavlja vezu između osobnog računala i mikrokontrolera koristeći paralelni, serijski ili USB port računala. Može zapisivati podatke u mikrokontroler i čitati ih nazad zbog njihove verifikacije. Funkcija PIC programatora je prevođenje razina digitalne logike sa osobnog računala u primjerene logičke razine mikrokontrolera – mnoge razine su u redu, ali za normalno (visokonaponsko) programiranje PIC mikrokontrolera potreban je maksimalni napon od 13,5 V na MCLR izvodu. To dodatno komplicira strujni krug sučelja budući da su naponi iz serijskog ili USB porta dosta niži od 13,5 V. Tipične razine digitalne logike su nominalno postavljene na 5 V zato uobičajeno ovi programatori zahtjevaju korištenje vanjskog izvora energije za generiranje višeg napona. Serijski port generira više napone i tu činjenicu koriste programatori kao što je JDM PIC programator. Njegov potpuni naziv je JDM serial port PIC programator i on pametno koristi serijski port za dobivanje viših napona i iz tog razloga ne iziskuje vanjski izvor energije. Jeftin je i lako se izrađuje, ali njegov najveći nedostatak je što se nemože integrirati u ploču razvojnog sustava. Svi PIC programatori rade na isti način – generiraju serijski tok podataka koristeći dvije signalna voda clock-a i podataka. Još jedan izvor kontrolira napon pri kojem se izvršava programiranje (MCLR izvod) i još dva izvoda koja kontroliraju izvor energije i masu. Aplikacija pokrenutana PC-ju (softver za programiranje) odabranu HEX datoteku generiranu iz compiler-a prevodi u serijski tok podataka. To je usmjereno u programator kroz određeno sučelje (serijski, paralelni ili USB port). Od tamo programator prenosi signale mikrokontroleru. Nakon što su svi podaci poslani, šalje se serijski konfiguracijski Word i mikrokontroler je programiran i spreman za upotrebu.

Metoda koja se koristi na novim PIC uređajima, a naziva se LVP (low volt programming) i kako joj ime govori ne zahtijeva visoki napon kod programiranja mikrokontrolera. Njezin nedostatak je specifičan način programiranja zbog kojeg je jedan izvod mikrokontrolera rezerviran i neupotrebljiv, što je u redu kod uređaja sa 40 izvoda, ali kod onih sa 18 izvoda može stvarati probleme. Najbitnije svojstvo PIC programatora je da on posjeduje PIC ICSP (in circuit serial programmer) konekciju. To je skup konekcija koje dozvoljavaju programiranje PIC mikrokontrolera dok se on i dalje nalazi unutar strujnog kruga. Zbog toga takav

način programiranja pogoduje razvoju prototipova budući da sav prije spojeni hardver ostaje nepromjenjen čak i kod programiranja mikrokontrolera. Vodove ICSP-a moguće je spojiti i na protoboard bez lemljenja pa je ovaj način jedan od najlakših za razvijanje električnih krugova i stvaranje prototipova.

4.1.Komercijalni PIC programatori – PICkit3 programator



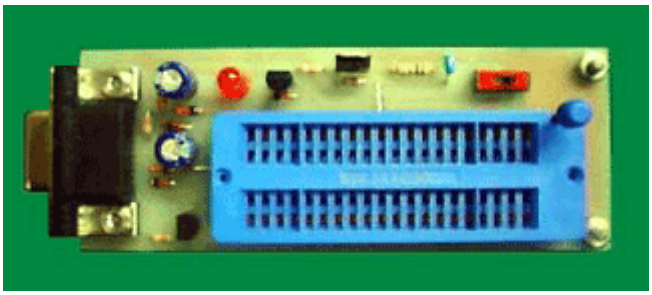
Slika 4.1: PICkit3 programator [S9]

Jedan od najviše korištenih PIC mikrokontrolera je PICkit 3 tvrtke Microchip. To je komercijalni programator / debugger koji je vrlo jednostavan, niske je cijene i ima mogućnost ICSP programiranja mikrokontrolera, a njime se kontrolira sa MPLAB IDE softverom koji se pokreće na računalu isključivo na Windows platformi. Sustav debuggera izvršava programski kod kao i realni uređaj te za to koristi uređaj sa ugrađenim sklopovljem za emulaciju umjesto zasebnog debuggerskog čipa. Svim mogućnostima datog uređaja može se pristupiti interaktivno te one mogu biti postavljene i modificirane kroz MPLAB IDE sučelje.

Neke od mogućnosti PICkit 3 programatora su:

- podrška za usb prijenos podataka pri punoj brzini koristeći standardne Windows drivere
- real-time izvršavanje
- procesori rade na maksimalnim brzinama
- ugrađen sklop za sprječavanje kratkog spoja, odnosno previsokog napona
- niski naponi programiranja do 5 V (raspon od 1,8 do 5 V)
- dijagnostičke LED diode (power, active, status)
- mogućnost čitanja i pisanja programskih kodova i podataka u/iz memorije mikrokontrolera
- mogućnost brisanja svih memorijskih tipova (EEPROM, ID, konfiguracijska i programska memorija) uz potvrdu

4.2. Besplatni PIC programatori - Multi PIC programator (JDM programator)

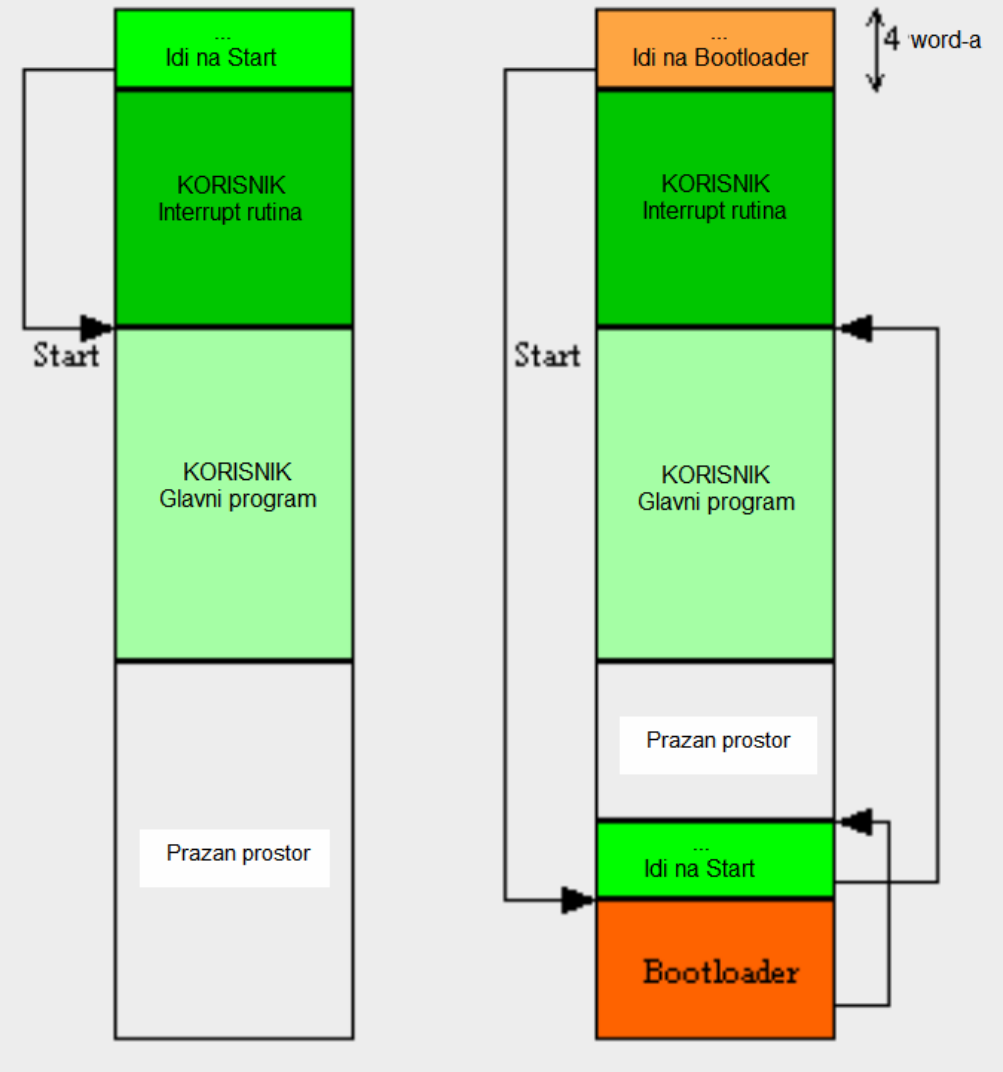


Slika 4.2: Multi PIC programator [S10]

No, komercijalni programatori nisu jedini način da bi se programiralo PIC mikrokontrolere, na Internetu postoje mnogi projekti za izradu PIC programatora koji zahtjevaju vrlo malo uloženog novca i truda, a daju iste mogućnosti kao PICkit3 programatori. Primjer jednog takvog je Multi PIC programator koji može programirati PIC mikrokontrolere sa od 8 do 40 izvoda, a koristi samo jedan ZIF socket. Nije mu potreban vanjski izvor energije i budući da je sastavljen od vrlo malo komponenti lako se izrađuje. Po svom načinu rada ovaj programator je JDM programator. U mogućnosti je programirati PIC mikrokontrolere obitelji 12F, 16F i 18F (ne sve modele) i nema točno određen softver s kojim radi stoga bi se za njegovu uporabu trebalo dodatno potražiti besplatni softver koji bi služio kao sučelje Multi PIC programatoru.

5. Bootloader

Da bi bolje razumjeli ulogu bootloader-a jedini pravi način da ju opišemo preko primjera računala. Centralni procesor koji se nalazi unutar računala može pokretati programske kodove isključivo ako se oni nalaze u ROM (Read-only memory) ili RAM (Random access memory) memoriji računala. Moderni operativni sustavi i aplikacije svoje programske kodove pohranjuju u takozvanu stalnu memoriju (to je memorija koja čuva podatke i kada nije uključena u struju) kao npr. HDD, CD, DVD, SD memorijske kartice, USB memorije itd. Kada se računalo koje je isključeno ponovno upali da bi se pokrenuo operativni sustav i sve aplikacije zajedno sa njim, najprije se moraju svi podaci potrebni za njihovo pokretanje prebaciti u RAM memoriju iz , primjerice, hard diska, a to se odvija pod kontrolom malenog programa koji se nalazi u ROM memoriji računala. Taj se program naziva bootloader i njegova svrha je da pokrene postupak popunjavanja (engl. loading) RAM memorije sa podacima za podizanje operativnog sustava. Sličan proces se mora izvršiti unutar PIC mikrokontrolera da bi on mogao izvršavati programske kodove koji su uneseni u njegovu Flash memoriju. Metoda korištenja bootloader-a je jedino moguća na novijim mikrokontrolerima koji imaju mogućnost re-programiranja dijelova vlastite flash memorije. Bootloader unutar PIC mikrokontrolera omogućava prijenos programskog koda između računala i PIC-a preko RS232 port-a (serijska komunikacija) ili USB port-a. Bootloader tipično nadgleda status ulazno-izlaznog (RX) izvoda svog USART modula (modul za serijsku komunikaciju) i aktivira se ako postoji bilo kakva aktivnost na izvodu unutar određenog vremenskog perioda nakon reseta mikrokontrolera npr. jedna sekunda od pokretanja. Nakon aktivacije, bootloader prima podatke sa RX izvoda te ih koristi za programiranje ostatka flash memorije (pritom zaobilazeći dio memorije gdje se sam bootloader nalazi). Nakon što sve podatke rasporedi po memoriji, odnosno isprogramira, započne sa izvršavanjem novog programa. Ukoliko na RX izvodu nema podataka za prijenos, bootloader nastavlja sa izvršavanje programskog koda koji je otprije pohranjen u flash memoriji mikrokontrolera.



Slika 5: Prikaz mapiranja flash memorije PIC mikrokontrolera sa i bez Bootloader-a [S11]

5.1. Prednosti i nedostaci

Prednosti:

- brži je od serijskog programiranja pomoću ICSP (In Circuit Serial Programmer) PIC programatora
- omogućuje unos programskog koda u potpuno izgrađenom sustavu
- dopušta korištenje više različitih sučelja za unos koda (RS232, USB, mrežno sučelje, infracrveno sučelje, itd.)
- njegova osnovna upotreba je u uređajima u kojima je potrebno osvježavanje (engl. update) softvera u mikrokontrolerima, a da pritom nije

potrebno otvarati uređaj i uključivati obični PIC programator; uređaji koji uobičajeno koriste RS232 port mogu se bootloadati kroz taj port
- nema hardvera za programiranje što omogućuje jednostavnije programiranje

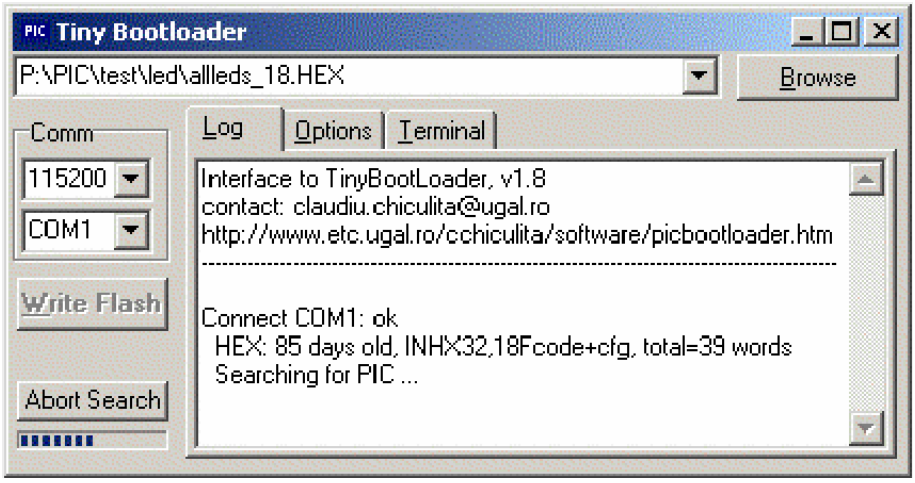
Nedostaci:

- budući da je smješten unutar flash memorije, mikrokontroler ima manje memorijskog prostora za glavni program
- upotrebljava se isključivo s mikrokontrolerima koji mogu reprogramirati vlastitu flash memoriju

5.2. Tiny PIC bootloader

Primjer besplatnog bootloader-a kojega je moguće implementirati u pozamašan broj modela PIC mikrokontrolera naziva se Tiny PIC bootloader. to je bootloader za Microchip PIC mikrokontrolere koji zauzima nešto manje od 100 word-ova programskog prostora što ga čini najmanjim poznatim bootloader-om. Podržava 16F, 18F, dsPIC obitelji PIC uređaja, odnosno sve novije uređaje koji podržavaju samoprogramiranje. Može zapisivati u flash, eeprom i konfiguracijske bajtove (18F). Postavljen je tako da nakon reseta mikrokontrolera pričeka 1 sekundu (naravno omogućeno je podešavanje te vrijednosti) poruku sa PC-a, te ako je ne dobije pokreće već postojeću aplikaciju.

Naravno, Tiny PIC bootloader ima svoj prilagođeni PC softver za komunikaciju. Njegove mogućnosti su: automatska detekcija HEX sadržaja i modela PIC mikrokontrolera, pamćenje zadnjih postavki, u slučaju greške samostalno pokušava resinkronizaciju s PIC-om, postavke komunikacije su u potpunosti podesive, pa se, primjerice, može odabrati bilo koji COM port ili željena brzina prijenosa podataka.

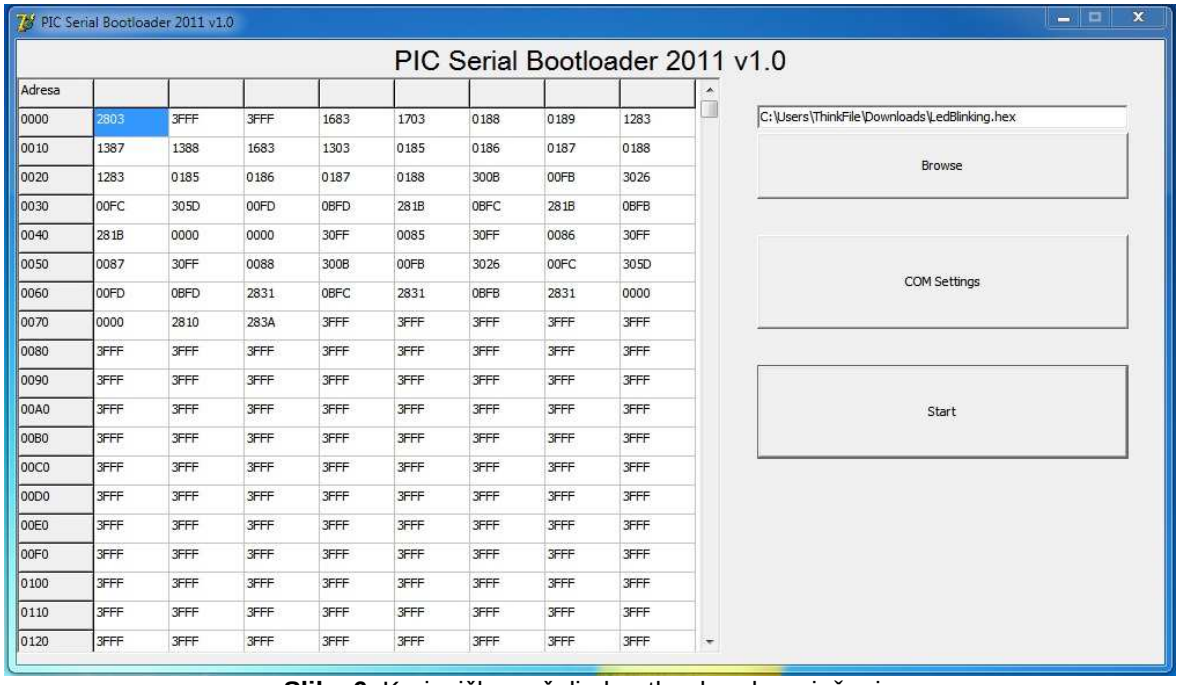


Slika 5.2: Korisničko sučelje Tiny PIC Bootloader-a [S12]

6. PIC Serial Bootloader 2011

Posljednji zadatak mog završnog rada bio je predložiti rješenje za programator PIC mikrokontrolera, a vodeći se po zaključcima dobivenim iz vlastitog istraživanja za ovaj rad odlučio sam se za izradu Bootloader-a jer mi se čini kao mnogo jednostavnije i efikasnije rješenje od klasičnih programatora. Projekt je nazvan PIC Serial Bootloader 2011 upravo zato jer komunicira sa računalom preko RS232 serijskog port-a. Aplikacija je u potpunosti izrađena u Delphi 7 razvojnom paketu.

6.2. Princip rada



Slika 6: Korisničko sučelje bootloader-skog rješenja

Kao što se vidi na gornjoj slici (Slika 6) korisničkog sučelja aplikacija zahtjeva HEX datoteku (dobivenu iz nekog od dostupnih razvojnih alata za programska rješenja za mikrokontrolere), zatim nudi opciju podešenja COM port-a preko kojeg se komunicira sa PIC mikrokontrolerom te naposljetku pritiskom na tipku Start slaže podatke očitane iz HEX datoteke u tablicu koja realno predstavlja memorijski prostor u mikrokontroleru u kojem ce se HEX kod pohraniti. nakon razvrstavanja po tablici word-ovi se šalju jedan po jedan u mikrokontroler i u skladu sa podacima o adresama bootloader u mikrokontroleru slaže te word-ove u svoju flash memoriju (adrese svakog pojedinog word-a se iščitavaju također iz tablice).

6.3. Programski kod

Kao što sa već napomenuo aplikacija je izrađena koristeći Delphi programski jezik, a u nastavku je kompletan kod PIC Serial Bootloader-a 2011.

Delphi kod:

```
// PROCEDURA KOD PRITISKA TIPKE ZA ODABIR HEX DATOTEKE
procedure TForm1.BrowseClick(Sender: TObject);
 begin
if PromptForFileName(select, 'Hex File (*.hex)|*.hex', '', 'Select your project file', 'C:\', False) then
  abc.Text := select
 end;
// PROCEDURA KOD PRITISKA TIPKE START
procedure TForm1.StartClick(Sender: TObject);
var Hex: textfile;
    buffer: string;
    strlength: integer;
    data,tmp: string;
    i,j,redak,pocetak,reps,counter,broj,brojac,bytecount,adress: integer;
    br: string;
begin
AssignFile(Hex, select);
Reset(Hex);
broj := 0;
tablica.Cells[0,0]:='Adresa';
  for i := 1 to 1024 do
    begin
      br := inttohex(broj,4);
      Tablica.Cells[0,i] := br;
      broj := broj + 16;
    end;
  for i := 1 to 8184 do
    for j := 1 to 8 do
      begin
      tablica.Cells[j,i] := '3FFF';
      end;
   counter := 1;
   while not Eof(Hex) do
   begin
        ReadLn(Hex, buffer);
        Delete(buffer, 1, 1);
        if not (buffer = '00000001FF') then
         begin
          strlength := Length(buffer);
          tmp := copy(buffer, 1, 2);
          bytecount := strtoint('$'+tmp); bytecount := bytecount div 2;
          tmp:=copy(buffer, 3, 4);
          adress := strtoint('$'+tmp);
          data := copy(buffer, 9, (strlength - 10));
          redak := adress div 16 + 1;
          pocetak :=  adress mod 16 div 2 + 1;
        if adress = 0 then
         begin
          for i := 1 to bytecount do
           begin
            tablica.Cells[i,redak] := copy(data, counter, 4);
```

```
          counter := counter + 4;
          tablica.Cells[i,redak] := copy(tablica.Cells[i,redak], 3, 2)
          + copy(tablica.Cells[i,redak], 1, 2);
         end;
        end;
         if not (adress = 0) then
         begin
         i := pocetak;
         reps := bytecount;
         brojac := 1;
         while (brojac <= reps) do
          begin
           if(i = 9 )then begin i := 1; redak := redak + 1; end;
             tablica.Cells[i,redak] := copy(data, counter, 4);
             counter := counter + 4;
             tablica.Cells[i,redak] := copy(tablica.Cells[i,redak], 3, 2)
             + copy(tablica.Cells[i,redak], 1, 2);
             i := i + 1;
             brojac := brojac + 1;
          end;
         end;
         end;
       if (buffer = '00000001FF') then
        begin
         showmessage('Loading complete!');
        end;
       counter := 1;
   end;
   CloseFile(Hex);
  comport1.open;
  for j:=1 to 1024 do
  for i:=1 to 8 do
   begin
    comport1.WriteStr(tablica.Cells[i,j]);
   end;
 comport1.close;
end;

// PROCEDURA KOD PRITISKA TIPKE COM SETTINGS
procedure TForm1.SettingsClick(Sender: TObject);
begin
comport1.ShowSetupDialog;
end;
end.
```

7. Zaključak

Kao osvrt na kompletan završni rad moram napisati da su me zaista zainteresirali mikrokontroleri, smatram da su oni jedan vrlo zanimljivi dio elektronike koji bi se većini elektrotehničkih entuzijasta i inžinjera svidio zbog svoje jednostavnosti i edukacije provedene kroz praktične primjere. Posebno sam zadovoljan, sada na kraju, jer sam rad sa mikrokontrolerima opisao pomoću skupa zadataka za koje sam sam pronaša idejna rješenja i implementirao ih, a navedene činjenice o programatorima, odnosno bootloader-ima potkrijepio vlastitim rješenjem bootloader-a za PIC mikrokontrolere. Nadam se da mi ovo nije posljednji put da ću raditi sa mikrokontrolerima, a ja ću se sigurno potruditi da to ne bude te da se moje znanje o njima dodatno proširi.

8. Literatura

- Paolo Zenzerović: «Udaljeni laboratorij za razvijanje mikroprocesorskih sustava: Idejno rješenje i projektiranje razvojne okoline», Završni rad

[S1] Slika 2.2 Razvojna pločica za mikrokontrolerske sustave – mikroElektronika http://www.mikroe.com/img/development-tools/pic/easypic6/gallery/easypic6_550_3.jpg

[S2] Slika 2.3 Specijalizirana razvojna pločica – Microchip http://embedded-system.net/embedded-system/images/embedded/2009/03/microchip-picdem-lab-development-kit.jpg

[S3] Slika 2.5 Sučelje Basic razvojne okoline tvrtke mikroElektronika http://www.mikroe.com/img/compilers/mikropascal/pro/pic/gallery/compiler_ide_01.png

[S4] Slika 2.9 Sedam segmentni ekran http://www.electrofun.biz/catalog/images/7Segments.jpg

[S5] Slika 2.10 Sedam segmenti ekrani u multipleksiranom načinu rada http://www.mikroe.com/img/development-tools/accessory-boards/display/serial-7-segment-1/gallery/serial7seg1_550_2.jpg

[S6] Slika 2.13 Alfanumerički LCD ekran http://www.mikroe.com/img/development-tools/compontents/2x16-lcd/lcd_thumb.gif

[S7] Slika 2.15 Grafički LCD ekran http://www.mikroe.com/img/development-tools/compontents/glcd/glcd_thumb.gif

[S8] Slika 2.17 Serijska komunikacija mikrokontrolera i osobnog računala mikroElektronika, mikroPascal ofr PIC, help

[S9] Slika 4.1: PICkit3 programator http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en538340&redirects=pickit3

[S10] Slika 4.2: Multi PIC programator http://feng3.nobody.jp/en/pg5v2.html

[S11] Slika 5: Prikaz mapiranja flash memorije PIC mikrokontrolera sa i bez Bootloader-a http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm

[S12] Slika 5.2: Korisničko sučelje Tiny PIC Bootloader-a http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm