UNIVERZA V LJUBLJANI
FAKULTETA ZA ELEKTROTEHNIKO

Matjaž Lotrič

TIMER Z μPROCESORJEM PIC16F84A

Seminarska naloga

Pri predmetu
Elektronska vezja

Železniki, februar 2003

1. UVOD

Naprave, ki delujejo le omejen čas, najelegantneje krmilimo z vezji, katerim lahko nastavimo časovni interval v katerem naprave delujejo. Ta vezja se imenujejo timerji. Izvedba timerjev je lahko različna, vendar je najelegantnejša izvedba z mikroprocesorji. Danes že najcenejši mikroprocesorji omogočajo izvedbo mnogih funkcij, tako da lahko poleg osnovne funkcije dodamo še dodatne sklope, ne da bi bilo potrebno uporabiti dodatna integrirana vezja.
Sam sem uporabil Microchip-ov procesor PIC16F84A. Dobra lastnost tega procesorja je, da je relativno poceni, poleg tega pa je podprt z preprostim hardwerom in softwerom za sprogramiranje. Vezje sestavljajo trije sklopi katere krmili procesor: to so LED displeji (3), tipke (3+1 za reset) in izhodni del (triak in optokopler). Najpomembnejši del vezja pa je seveda program, ki pove kako naj vse deluje. Vse te sklope in njihovo delovanje bom predstavil v nadaljevanju.

2. DELOVANJE VEZJA

Kakor vidimo iz sheme lahko vezje razdelimo na dva dela:

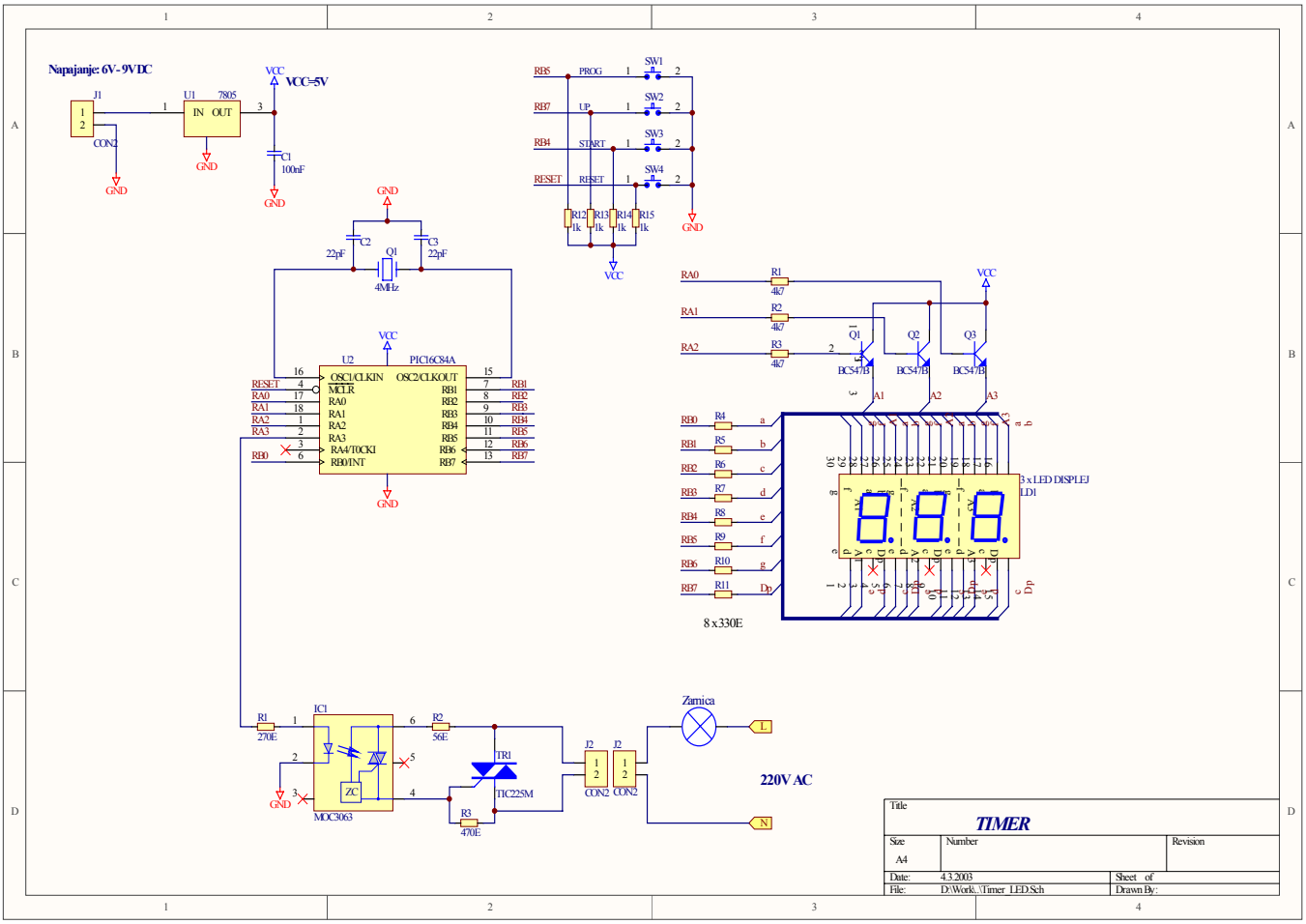
- en del vezja predstavlja mikroprocesor in sklopi okoli njega
- drug del pa vsebuje del, ki skrbi za vklop in izklop luči

Srce vezja je vsekakor mikroprocesor iz Microchip-ove družine 16F84A. Gre za 18 pinski 8 bitni FLASH RISC procesor katerega glavne značilnosti so:

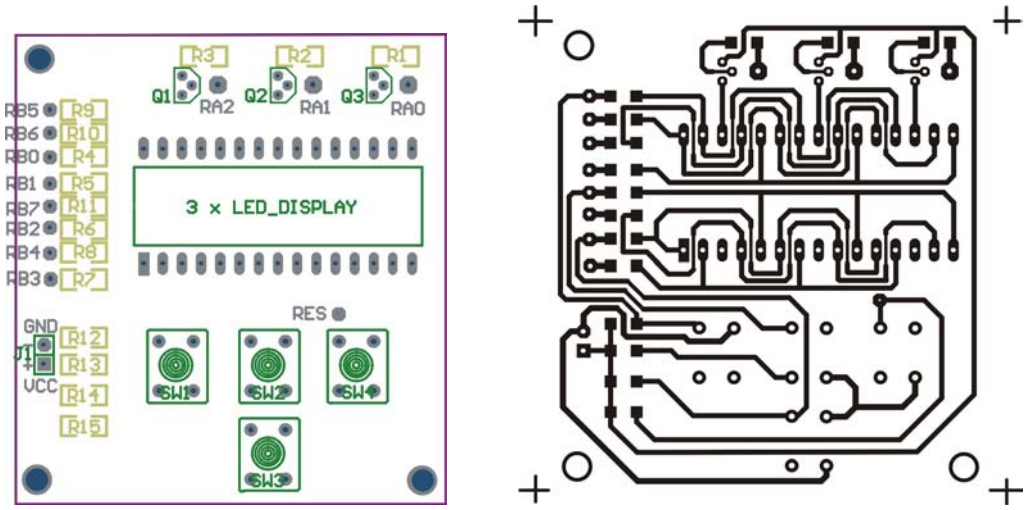
- samo 35 ukazov
- 1K x 14 programskega pomnilnika (FLASH)
- 64 bajtov podatkovnega pomnilnika (EEPROM)
- max. hitrost ure je 20MHz
- 13 vhodno / izhodnih pinov
- 4 prekinitve
- možnost programiranja v vezju oz. ISP

Mikroprocesor skrbi za vse opravila, ki jih vezje izvaja. Iz sheme (*Slika 1*) vidimo, da so na PortB priključeni displeji ter tri tipke, ki jih uporabimo za nastavljanje časa ter za štartanje odštevanja. Četrta tipka je reset (*RST*) in nam resetira delovanj programa. Stanje »reset« se pojavi ob priklopu vezja na napajalno napetost, kasneje pa po priključitvi signala $\overline{\textbf{MCLR}}$ (pin 4) na nizek nivo, kar se ob pritisku na tipko *RST* tudi zgodi. Displeje z skupno anodo prižigamo s pomočjo NPN tranzistorjev (BC 547B), ki se ob krmilnem signalu iz PIC-a odpro in povežejo anodo z napajanjem. (V_{CC} = 5V) Signali za krmiljenje tranzistorjev so priključeni na PortA (RA0 … RA2) PIC mikroprocesorja. Izhod PortA RA3 pa skrbi za prižiganje in ugašanje luči. Signal ni vezan direktno na triak, ki skrbi za vklapljanje oz. izklapljanje porabnika, temveč je nanj povezan indirektno preko optokoplerja (MOC 3063). Le ta nam optično loči del vezja, ki je priklopljen na omrežno napetost od ostalega dela vezja. Porabnik priključimo na vijačne sponke J2. Čip IC3 je stabilizator napetosti 7805 in poskrbi, da usmerjeno napetost, ki jo pripeljemo na sponke J1 stabilizira na 5V in jo uporabljamo za napajanje procesorja in ostalega vezja.

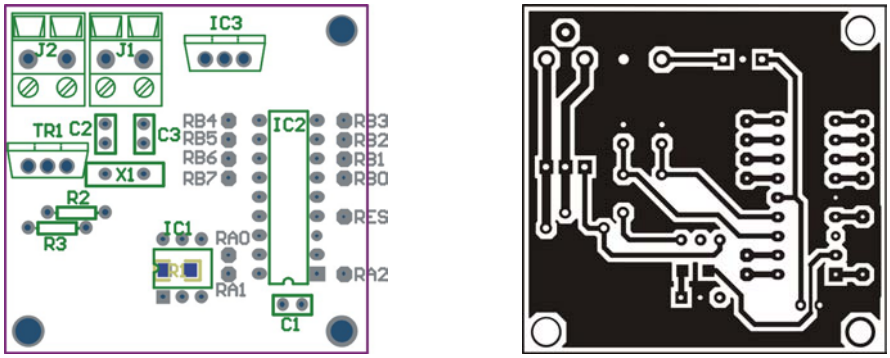
Tiskano vezje sem izdelal na dveh ploščicah s programskim paketom Protel 99 SE. Ploščici sem povezal v sendvič tako, da je ploščica s tipkami in displeji pritrjena na čelno ploskev, vse ostalo s procesorjem in triakom pa je na drugi ploščici. (*Slika 2* in *Slika 3*)



SLIKA 1: Električna shema vezja



SLIKA 2: TIV z displeji in tipkami



SLIKA 3: TIV z procesorjem in triakom

3. OPIS PROGRAMA

Za pravilno delovanje vezja skrbi PIC mikroprocesor, ki je srce vezja. Ker pa je le ta brez programa le neumen košček silicija v plastičnem ohišju, je bilo potrebno napisati program. Program je bil napisan v asemblerju in je sestavljen iz večih delov:

- glavnega programa oz. *MAIN*
- prekinitvene rutine oz. *INT* in
- več podprogramov kot so: *OSVEZI, UP_KEY, SETUP, TIPKE, ...*

Na samem začetku programa sem najprej definiral tip procesorja, imena spremenljivk, reset vektor za začetek programa in prekinitveni vektor, konstante, definicije posameznih registrov oz. bitov za lažje razumevanje in delo ter datoteko z makro programi. Nato si v programu sledijo prekinitvena rutina, podprogrami in isto na koncu še glavni program.

3.1. GLAVNI PROGRAM

V glavnem programu najprej s pomočjo makro programa obrišemo vse splošno namenske registre (od 0x0C do 0x30). Sledi inicializacija vhodov in izhodov, s katero povemo kateri pini so vhodi in kateri so izhodi. Vrednosti vseh I/O pinov nastavimo na nič. Ker sem uporabil TMR0, mu je potrebno v OPTION registru nastaviti kakšna bo frekvenca ure, ki ga poganja. Sam sem mu notranjo uro (1MHz) zmanjšamo v razmerju 1:16. Sledi nastavitev kaj bo prikazano na displejih na samem začetku, ob vklopu vezja. Nato moramo omogočiti prekinitev, ki jo sproži TMR0. TMR0 je v bistvu 8 bitni števec, ki sproži prekinitev ko gre iz stanja 0xFF → 0x00. Urin signal mu lahko določamo s pomočjo preddelilnika, tako kot nam ustreza. Prekinitev TMR0 omogočimo z postavitvijo bitov GIE in T0IE na logično enico. Bita se nahajata v INTCON registru. Na koncu sem program zaciklal v neskončni zanki, vsa ostala opravila se prožijo iz prekinitvene rutine.

3.2. PREKINITVENA RUTINA

Prekinitvena rutina predstavlja najbolj zapleten del programa. Le ta zagotavlja, da se prekinitve prožijo v pravih časovnih intervalih za zagotavljanje točnosti ure. Za zagotavljanje točnosti je najbolje uporabiti PIC-ov notranji timer in njegovo prekinitev TMR0. Kot sem že omenil je TMR0 števec, ki šteje ali zunanje impulze ali pa ga poganjamo z notranjo uro ($f_N = f_{OSC} / 4$). Časovni interval za proženje prekinitev sem nastavil na 4ms in prekinitev se je morala izvesti natanko 250 krat da sem dobil 1s. (4ms * 250 = 1000ms = 1s) Ob nastavljenemu preddelilniku na 1:16 se prekinitev sproži po 4,096ms, kar pa ni sprejemljivo, zato je potrebna programska korekcija. V prekinitveni rutini zato v TMR0

vpišemo vrednost 7 in dodamo potrebno število ukazov, ki ne vplivajo na izvajanje (*nop*), le podaljšajo nam kodo.

Poglejmo preprost izračun:

$$f_{OSC} = \mathbf{4MHz} \rightarrow f_N = \mathbf{1MHz} \rightarrow f_N / \mathbf{16} = f_{ŠT} = \mathbf{62{,}5kHz} \rightarrow T_{ŠT} = \mathbf{16\mu s}$$

Števec TMR0 šteje od 0 do 255:

$$T_{CIK} = 256*16\mu s = \mathbf{4{,}096ms} \Rightarrow \textbf{NAPAKA}!!$$

Programska korekcija: v TMR0 vpišemo vrednost 7, števec sedaj šteje od 7 do 255 $\Rightarrow$ to je:

$$T_{CIK} = (256-7)* 16\mu s = \mathbf{3{,}984ms}$$

Vidimo da nam do točnih **4ms** manjka še **16µs**. Ker je pri f_{OSC} = 4MHz vsak ukazni cikel dolg **1µs** samo preštejemo koliko ciklov je že preteklo do potrebne korekcije in razliko od potrebnih 16 dopolnimo z ukazi ***NOP*** *(no operation)*.

Med samo prekinitvijo sem klical podprograme, ki so skrbeli za osveževanje displejev, skeniranje tipk, nastavljanje ure in nazadnje še za vklop porabnika. Na začetku prekinitvene rutine je potrebno izklopiti možnost, da med izvajanjem TMR0 prekinitve ne vpade še kakšna nova prekinitev. Prav tako je na začetku potrebno shraniti vrednost delovnega (W) ter STATUS-nega registra na pomožne lokacije v pomnilniku. Na koncu pa ne smemo pozabiti obnoviti vrednosti teh dveh registrov. To storita makro ukaza ***PUSH*** in ***PULL***.

3.3. PODPROGRAMI

Podprogrami ali subrutine skrbijo za pravilno delovanje podsklopov vezja.Uporaba subrutin je elegantnejša, saj je pisanje programa hitrejše in razumljivejše. Tudi ob večkratnem ponavljanju dela programske kode je smotrnejši zapis le tega kot subrutino.

Vezje omogoča prikaz minut (0 … 4) in sekund (0 … 59) na treh sedem-segmentnih LED displejih. Displeji delujejo v multipleksnem načinu, kar pomeni, da je v vsakem trenutku aktiven le eden, ostala dva pa sta ugasnjena. Pri tem moramo paziti le, da je frekvenca osveževanja dovolj velika, da ne vidimo utripanja displejev. Za osveževanje je poskrbljeno tako, da se ob prekinitvi, ki se izvrši vsakih 4ms osveži en displej. Vsak displej se torej osveži 83,3 krat na sekundo, kar pa več kot zadošča.

Tipke program skenira ob vsaki prekinitvi in spremembo stanja zapiše v register. Ob skeniranju tipk se displeji ugasnejo, saj se za tipke uporabljajo isti pini porta PortB kot za podatke za LED displeje. Ko je tipka pritisnjena se postavi v kontrolnem registru zastavica, ki prepreči, da bi se en pritisk na tipko zaradi preklopnih motenj upošteval večkrat. S tem preprečimo nestabilnosti v delovanju. Bit se zbriše, ko se v nadaljevanju programa pritisk tipke registrira in upošteva.

Nastavljanje ure poteka tako, da se seštevajo sekunde. To pomeni, da se pri nastavljanju minut za vsak pritisk na tipko *UP* k skupni vsoti prišteje vrednost 60, za desetice sekund 10 in za enice sekund 1. Vsota se shranjuje v 8 bitni register, tako da je maksimalna vrednost, ki zagotavlja pravilno delovanje 255. Od tu tudi izhaja omejitev, da je maksimalen čas, ki ga lahko nastavimo 4 minute in 15 sekund. Seveda se ta čas lahko poveča z razširitvijo registra namenjenega za vsoto iz 8 na 16 bitov.

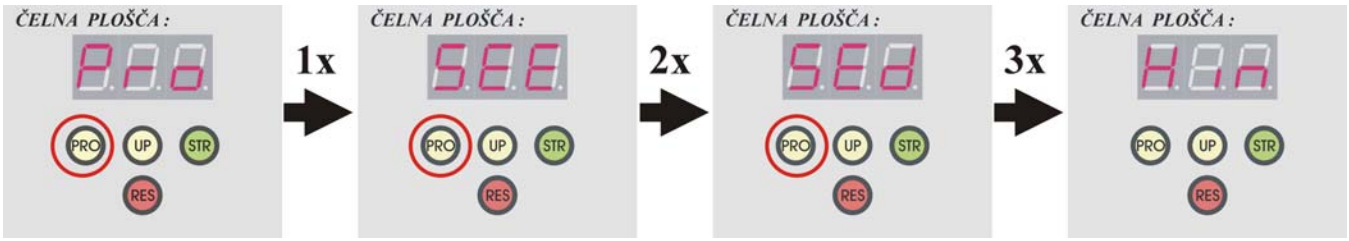
Ob pritisku na tipko *STR* se vsako sekundo ta vsota zmanjša za 1, nova vrednost pa se preračuna v podatke, ki jih prikazujemo na zaslonu. Ob pritisku na STR se postavi na logično enico tudi izhod RA3, ki povzroči odprtje triaka in s tem tudi priklop porabnika na omrežje. Ko je vsota enaka nič, se RA3 postavi na ničlo in triak se zapre.

4. NASTAVLJANJE TIMERJA

Ob vklopu vezja se nam na displejih izpiše **Pro**, kar pomeni, da je vezje pripravljeno za nastavitev timerja. (*Slika 4*). Ob pritisku na tipko *PRO*, se nam na displejih pokaže kaj nastavljamo: (*Slika 5*):

- če tipko *PRO* pritisnemo 1x nastavljamo enice sekund
- če tipko *PRO* pritisnemo 2x nastavljamo desetice sekund
- če tipko *PRO* pritisnemo 3x nastavljamo minute

SLIKA 4: Pogled na čelno ploščo ob priklopu napetosti

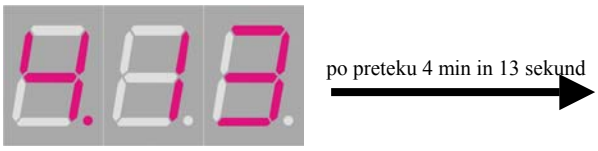


SLIKA 5: Prikaz na zaslonu ob večkratnem pritisku tipke *PRO*

Nastavljanje časa poteka v treh korakih s pomočjo tipk *PRO* ter *UP* in sicer: najprej nastavimo enice sekund, nato desetice sekund in nazadnje še minute. Maksimalen čas, ki ga lahko nastavimo, da vezje pravilno dela je 4:15.
Z tipko *UP* nastavimo želen čas tako, da pritisnemo nanjo tolikokrat, da se na zaslonu prikaže želena številka.

Primer nastavitve časa 4 min 13 sekund:

1. Tipko *PRO* pritisnemo enkrat (displej: ***SEE***) in nato pritisnemo tipko *UP* 3x
   (na displeju: **0.03**)
2. Še enkrat pritisnemo tipko *PRO* (displej: ***SEd***) in nato tipko *UP* 1x
   (na displeju: **0.13**)
3. Če pritisnemo tipko *PRO* še enkrat (displej: ***Min***) lahko nastavimo minute. Za 4 minute pritisnemo tipko *UP* 4x (*Slika 6*)
   (na displeju: **4.13**)



SLIKA 6: Nastavljen čas

SLIKA 7: Displej po preteku časa

Ko je čas nastavljen, s tipko *STR* (*Slika 8*) sprožimo odštevanje časa. V trenutku pritiska se prižge luč oz. poljuben porabnik in je priklopljen dokler ne pride števec do nič (*Slika 7*). Ko pride do nič se porabnik avtomatsko izklopi.

SLIKA 8: S tipko STR vklopimo odštevanje

5. ZAKLJUČEK

Opisan timer uporabljam pri izdelavi tiskanih vezij v osvetljevalni komori. Timer mi prižiga in ugaša dve neonski UV žarnici, ki sta potrebni za izdelavo vezij. Čas osvetljevanja je pri izdelavi vezij po fotopostopku pomemben faktor, saj vsaka sekunda preveč ali premalo pomeni, da je ploščica preveč ali premalo osvetljena. To pa vsekakor poveča čas in posledično tudi stroške izdelave, saj je v vsakem primeru potrebno nanos fotolaka na ploščici ponovno nanesti.
Točnost timerja sem kontroliral z štoparico in ugotovil, da se nastavljeni čas ujema z časom, ki ga kaže štoparica. Seveda bi za točnejše meritve potreboval osciloskop, s katerim bi pomeril čas trajanja impulza, ki drži triak odprt. Med testiranjem vezja sem opazil, da mrzli neonski žarnici potrebujeta nekaj sekund da se prižgeta, zato je priporočljivo, če pred pričetkom osvetljevanja nastavimo čas nekaj sekund, da se neonki ogrejeta, šele nato pa začnemo z osvetljevanjem. Tu se kaže tudi možnost za izboljšavo vezja, saj bi z dodan senzor UV svetlobe šele ob konstantni UV svetlobi sprožil začetek odštevanja.
Druga omejitev pri opisanem vezju pa je maksimalen čas, ki ga lahko nastavimo. Ker mi 4 min in 15s zadošča, nisem programa napisal tako, da bi bil maksimalen čas večji. Seveda se to lahko hitro popravi, saj je v bistvu treba dodati le še en register, ki poveča 8 bitni register v 16 bitnega.

6. LITERATURA

1. Datasheet za PIC16F84A (http://www.microchip.com/)
2. Application Note AN-3004: Applications of Zero Voltage Crossing Optically Isolated Triac Drivers (http://www.fairchildsemi.com/)
3. Datasheet za MOC3061 (http://www.fairchildsemi.com/)