Univerza v Ljubljani
Fakulteta za elektrotehniko

Marko Novak

Krmilnik koračnih motorjev

Seminarska naloga

pri predmetu
Elektronska vezja

Na Jesenicah, december 2007

Uvod

Motivacija, katera me je gnala pri tem projektu je bila ta, da se kot prvo naučim programirati mikrokontrolerje (real time) v assemblerski kodi in da celoten izdelek predstavim v neki zaključeni celoti, kot delujoč, v nekaterih aplikacijah tudi uporaben, kajti uporaba koračnih motorjev je v industriji zelo velika.

Kot glavni element v vezju je mikrokomtroler PIC16f84A , okoli katerega so razporejeni tudi drugi elementi, kot so stabilizator napetosti 78L05, ki skrbi za stabilizacijo napetosti 5V, nekaj kondenzatorjev, uporov, potenciometer za spreminjanje hitrosti vrtenja, tipke s katerimi krmilimo smer vrtenja ter izhodna stopnja katero sestavlja integrirano vezje ULM2003. To integrirano vezje predstavlja močnostni del krmilnika, katero ojači izhodne tokove iz mikrokontrolerja. V uvodu moram povedati še da celotna aplikacija deluje na dveh napetostnih nivojih, in sicer digitalni(logični) del deluje na napetosti 5V, napajalni del koračnega motorja pa deluje na napetosti 12V.

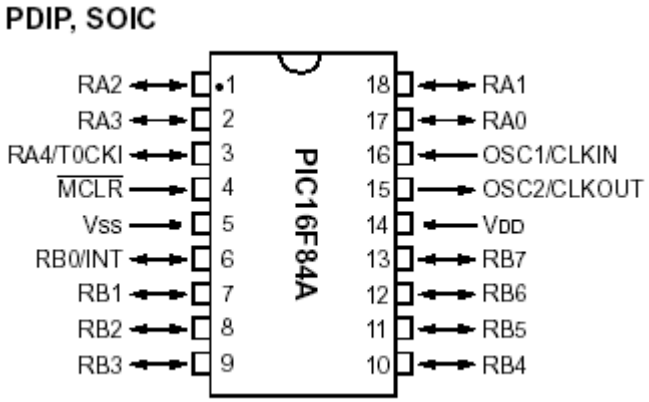
Mikrokontroler PIC16F84a

Kot sem že v uvodu povedal, aplikacijo sestavlja več integriranih elementov in podvezji.
Srce celotne aplikacije je mikrokontroler iz družine PIC in sicer 18 pinski z oznako 16f84a. Družina PIC16fxx je družina 'low cost' visoko zmogljivih mikrokontrolerjev. Danes jih najdemo v tisočih miškah, telefonih, scanerjih, avtomobilih… Uporabljajo pa se tudi krmilni(kot v mojem primeru), regulacijski in merilni tehniki. PIC-i vsebujejo zraven hitrega RISC (reduced instruction set computer) procesorja vse komponente kompletnega mikrokontrolerskega sistema kot so EEPROM ali PROM, RAM, 13 oz. 21 digitalnih vhodov/izhodov, ki so obremenljivi do 25 mA, timer/števec, watchdog timer, powre on reset in oscilator start-up timer, kar vse zmanjša zahtevo po zunanjih elementih. Zaradi majhne porabe je možna uporaba tudi v baterijsko napajanih vezjih. Ker imajo maksimalno 35 ukazov je šas uvajanja za delo s to serijo sorazmerno kratek.

Osnovne lastnosti PIC16f84: visoko zmogljiv RISC CPU, samo 35 enobesednih 14 bitnih ukazov, vsi ukazi so opravljeni v enem ukaznem ciklu, razen programske vejitve v dveh, delovna hitrost 20MHz, 1024 x14 on chip EEprom programski pomnilnik, 36 x 8 registrov(SRAM), 15 specialnih hardverskih registrov, 64 x 8 EEPROM podatkovnega pomnilnika, osem nivojski hardverski sklad, direktno, indirektno in relativno naslavljanje, štiri prekinitve, 13 individualno krmiljenih vhodno/izhodnih pinov, TMR0: 8 bitni real time clock/cunter z 8 bitnim programabilnim delilnikom,

power on reset, power up timer, oscilator stert up timer watch dog timer z lastnim on chip oscilatorjem, security EEPROM fuse za zaščito programa, varčni sleep način delovanja, možnost izbire različnih oscilatorjev kot so RC, crystal XT(kot v mojem primeru) in še drugi.

Položaj pinov:



Arhitektura: PIC16f84 uporablja Harvard arhitekturo z ločenim vodilom za program in podatke, kar je izboljšava z Von-Neumanovo arhitekturo. Zaradi te delitve so ukazi drugačne širine kot 8 bitne besede, lahko so 14 bitne.

Programski pomnilnik: ima 13 bitni programski števec, ki omogoča naslavljanje 8Kx14 programskega pomnilnika, vendar je le prvih 1Kx14 fizično prisotnih. Poleg tega ima EEPROM programski pomnilnik omejitev(na približno 1000) brisalno/pisalnih ciklov, zato ni primeren za aplikacije, kjer se program pogosto spreminja.

Podatkovni pomnilnik, Register File: PIC kontrolerji imajo notranji podatkovni register poimenovan register file. Organiziran je kot 128 x 8 bitov. Dostopen je prek registra FSR, ki je prav tako podatkovna pomnilniška lokacija. Med različnimi stranmi file-a registra se preklapla glede na dva bita v STATUS registra. 16f84 ima dve strani (Bank0, Bank1). Prvih 12 registrov je specialnih (v vsaki banki 12, nekateri se pojavljajo v obeh, nekateri samo v eni, zato je treba pri programiranju biti pazljiv v kateri banki se trenutno nahajamo.) registrov in ti so: **Indirect data addresing** nam daje možnost indirektnega naslavljanja. Vsak ukaz, v katerem se uporablja INDF kot file register, se dejansko nanaša na register, na katerega kaže vsebina File select refistra FSR. **Real time clock/counter register (RTCC)**, pri katerem se njegova vsebina povečuje glede na taktni signal, ki je lahko zunanji prek RTCC vhoda ali notranji prek notranjega oscilatorja.**Program counter (PC) register** je 13 biten in generira naslov

programskega pomnilnika. V normalnem teku programa se v vsakem urinem ciklu poveča za ena, razen če nek ukaz ne vpliva na njegovo vsebino. **Status register (STATUS)** vsebuje status aritmetično logične enote, reset status in bite za izbiranje strani podatkovnega registra (Bank0, Bank1). **Vhodno/izhodna registra** (**PORTA, PORTB**), sta registra dveh portov in sicer porta A, kateri ima razpoložljivih samo 5 vhodno/izhodnih pinov in registra B, kateri im 8 vhodno/izhodnih pinov. **TRISA in TRISB registra** služita za upravljanje registrov PORTA in PORTB. Povesta nam kateri pini PORTAA in PORTAB so vhodni in kateri so izhodni. Pri TRIS registrih moramo biti pazljivi pri izbiri banke saj se nahajata v banki1. **Option register (OPTION)** je bralno/pisalni register, ki vsebuje kontrolne bite za konfiguracijo programskega preddelilnika, fronto zunanje prekinitve, TMR0 in šibke Pull Up upore na PORTB. Biti je treba pazljiv saj se nahaja v Bank1.

Sklad: PIC16f84 ima 8 nivojski 13 bitni hardware sklad, ki ni del programskega ali podatkovnega pomnilnika. Vsebine kazalca sklada (stack pointer) ni možno ne brati ne pisati, le ta se poveča v primeru call ukaza ali ko se zazna ena od prekinitev., istočasno se vrednost programskega števca PC potisne na sklad.

Working register: je 8 bitni register in je primerljiv z akumulatorskim registrom ostalih mikrokontrolerjev. Uporablja se za aritmetične logične operacije, ni pa nujno cilj teh operacij, kar j majhna posebnost.

Splošno namenski registri: so uporabniški registri in so na lokacijah od 0Ch do 2Fh. Njihova vsebina je ob 'Powre up resetu' nedifinirana oz. se ob drugih resetih ne spremeni.

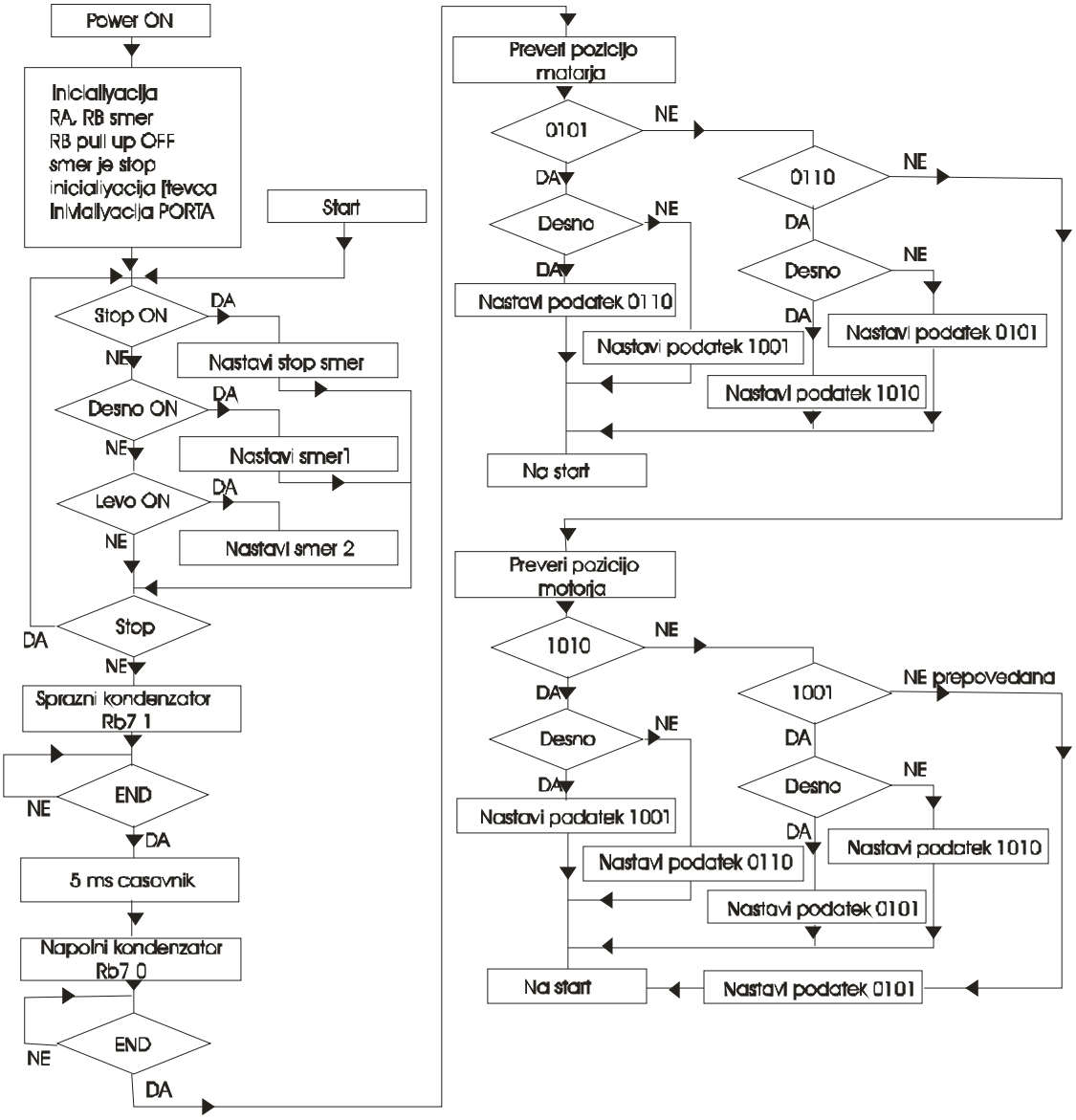
Prekinitve: INT prekinitev ali zunanja prekinitev na pinu RB0/INIT lahko deluje na dvigajočo ali na padajočo fronto. KO se pojavi ustrezna fronta , se postavi bit INTF, ki mora biti zbrisan, preden se ponovno omogoči ta prekinitev. **TMR0 prekinitev** je generirana, kadar TMR0 counter/timer prekorači vrednost iz 0ffh na 00h. Ta prekinitev ne more prebuditi procesorja iz 'Sleep' načina, ker pri tem načinu ne deluje. **Port RB prekinitev** se izvrši ob spremembi vsaj enega od vhodov RB<7,4>. **EEPROM write prekinitev** se postavi, kadar se vpis v eprom konča.

Watchdog Timer (WDT): je števec, ki mora biti programsko zbrisan preden preseže maksimalno vrednost. Če se to ne zgodi v določenem času pride do t.i. WDT – Timeouta, ki povzroči RESET procesorja. Z WDT lahko preverjamo pravilno delovanje programa. Delujoči program mora v določenih presledkih ves čas brisati vsebino WDT-ja. Pri programiranju je treba paziti, da se ukaz za brisanje WDT postavi na mesta, kjer ima program definirano stanje. Nikakor pa se ta ukaz ne sme pojaviti v prekinitvenih rutinah.

Power Down Mode (Sleep): varčevalni način delovanja, ki se aktivira z ukazom SLEEP. Če WDT vključen, se le ta zbriše, vendar še vedno teče. Bit PD se zbriše, TO bit STATUS registra se postavi in oscilator je izključen. I/O biti obdržijo vrednost, ki so jo imeli pred ukazom Sleep.

Programiranje mikrokontrolerja

Programska shema programa za krmiljenje koračnega motorja:



Izvorna assemblerska koda v moji aplikaciji katera je vpisana v mikrokontroler:

```
;*****************************************************
;
;               krmilnik koračnega motorja
;
;
```

```
;*****************************************************

                                                    list        p=16f84a

p=16f84a
                                                    #include    'p16f84a.ing'

#include

"p16f84a.inc"


__CONFIG

_WDT_OFF & __CONFIG     WDT_OFF & _CP_OFF & _PWRTE_ON & _XT_OSC

      errorlevel    -302    ;eliminacija bank opozorila


      cblock  h'0c'
smer                  ;smer vrtenja
                     ;0=stop 1=right 2=left
stevec1                ;števec za čakanje 5ms
stevec2                ;števec za čakanje 1ms
      endc

rb0    equ    0        ;RB0 za PORTB
rb1    equ    1        ;RB1 za PORTB
rb2    equ    2        ;RB2 za PORTB
rb5    equ    5        ;RB5 za PORTB
rb7    equ    7        ;RB7 za PORTB

;***************  Začetek programa  **********************
      org    0              ;Reset
      goto   init
      org    4              ;prekinitev
      clrf   INTCON         ;počisti prekinitveni reg

;***************  inicializacija  ********************
init
      bsf    STATUS,RP0      ;Banka 1
      clrf       TRISA       ;nastavi PORTA vsi izhodi
      movlw    b'00100111'    ;RB0,1,2.5=vhod RB7=izhod
      movwf    TRISB          ;nastavi PORTB
      movlw     b'10000000'   ;RBPU=1 Pull up ni uporabljen
```

```
        movwf   OPTION_REG      ;nastavi OPTION_REG
        bcf     STATUS,RP0      ;Banka 0
        clrf    smer            ;nastavi smer = stop
        clrf    stevec1         ;zbriši števec
        clrf    stevec2         ;zbriši števec
        movlw   b'00000101'     ;nastavi PORTA začetno vrednost
        movwf   PORTA           ;vpiši PORTA
        bsf     PORTB,rb7       ;nastavi RB7 = 1
        btfsc   PORTB,rb5       ;RB5 = 0 ?
        goto    $-1             ;ne. počakaj

start
;*************  preverba stanja stikal  *****************
        btfsc   PORTB,rb1       ;RB1(stop tipka) = ON ?
        goto    preveri1        ;ne. naslednja
        clrf    smer            ;da. nastavi stop smer
        goto    krmilje         ;ne. skoči na krmilje motorja
preveri1
        btfsc   PORTB,rb2       ;RB2(desna tipka) = ON ?
        goto    preveri2        ;ne. naslednja
        movlw   d'1'            ;da. nastavi desno smer
        movwf   smer            ;shrani smer
        goto    krmilje         ;ne. skoči na krmilje motorja
preveri2
        btfsc   PORTB,rb0       ;RB0(leva tipka) = ON ?
        goto    krmilje         ;ne. skoči na krmilje motorja
        movlw   d'2'            ;da. nastavi levo smer
        movwf   smer            ;shrani smer

;********************  krmilje motorja  *********************
krmilje
        movf    smer,w          ;beri smer
        bz      start           ;smer = stop
        bsf     PORTB,rb7       ;nastavi RB7 = 1
        btfsc   PORTB,rb5       ;RB5 = 0 ?
        goto    $-1             ;ne. čakaj
        movlw   d'5'            ;nastavi števec1(5msec)
        movwf   stevec1         ;shrani števec1
skoci   call    casovnik        ;čakaj 1msec
        decfsz  stevec1,f       ;stevec - 1 = 0 ?
        goto    skoci           ;ne. nadaljuj
        bcf     PORTB,rb7       ;nastavi RB7 = 0
        btfss   PORTB,rb5       ;RB5 = 1 ?
        goto    $-1             ;ne. čakaj
        movf    PORTA,w         ;beri PORTA
```

```
        sublw    b'000000101'    ;preveri pozicijo motorja
        bnz      krmilje2        ;naujemanje
        movf     smer,w          ;beri smer
        sublw    d'1'            ;desno ?
        bz       krmilje1        ;da. desno
        movlw    b'00001001'     ;ne. nastavi levi podatek
        goto     krmilje_konec   ;skoči na PORTA pisanje
krmilje1
        movlw    b'00000110'     ;nastavi desni podatek
        goto     krmilje_konec   ;skoči na PORTA pisanje
;-------
krmilje2
        movf     PORTA,w         ;beri PORTA
        sublw    b'000000110'    ;preveri pozicijo motorja
        bnz      krmilje4        ;naujemanje
        movf     smer,w          ;beri smer
        sublw    d'1'            ;desno ?
        bz       krmilje3        ;da. desno
        movlw    b'00000101'     ;ne. nastavi levi podatek
        goto     krmilje_konec   ;skoči na PORTA pisanje
krmilje3
        movlw    b'00001010'     ;Set Right data
        goto     krmilje_konec   ;skoči na PORTA pisanje
;-------
krmilje4
        movf     PORTA,w         ;beri PORTA
        sublw    b'000001010'    ;preveri pozicijo motorja
        bnz      krmilje6        ;naujemanje
        movf     smer,w          ;beri smer
        sublw    d'1'            ;desno ?
        bz       krmilje5        ;da. desno
        movlw    b'00000110'     ;ne. nastavi levi podatek
        goto     krmilje_konec   ;skoči na PORTA pisanje
krmilje5
        movlw    b'00001001'     ;Set Right data
        goto     krmilje_konec   ;skoči na PORTA pisanje
;-------
krmilje6
        movf     PORTA,w         ;beri PORTA
        sublw    b'000001001'    ;preveri pozicijo motorja
        bnz      krmilje8        ;naujemanje
        movf     smer,w          ;beri smer
        sublw    d'1'            ;desno ?
        bz       krmilje7        ;da. desno
        movlw    b'00001010'     ;ne. nastavi levi podatek
```

```
        goto        krmilje_konec   ;skoči na PORTA pisanje
krmilje7
        movlw       b'00000101'     ;Set Right data
        goto        krmilje_konec   ;skoči na PORTA pisanje
;-------
krmilje8
        movlw       b'00000101'

krmilje_konec
        movwf       PORTA           ;pisanje PORTA
        goto        start           ;skoči start

;*************  1msec časovna subroutina  *****************
casovnik
        movlw       d'200'          ;nastavi števec2
        movwf       stevec2         ;shrani števec2
skoci1  nop                         ;časovna prilagoditev
        nop                         ;časovna prilagoditev
        decfsz      stevec2,f       ;števec - 1 = 0 ?
        goto        skoci1          ;ne. nadaljuj
        return                      ;da. konec odštevanja


        end
```

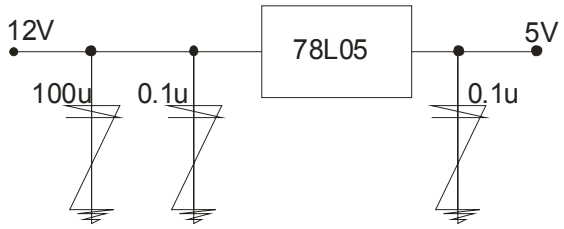
Izvedba programiranja, prevajanja v šestnajstiško kodo ter opisovanja peograma v mikrokontroler: program sem napisal v assemblerski kodi in sicer v programu MP LAB 6.2 s katerim sem ga tudi prevedel. Program se da dobiti na internetu in je na voljo vsem, namenjen pa je programiranju mikrokontrolerjev firme Microchip. Po hitrem privajanju na program, sem začel s programiranjem. Imel sem kar dosti težav, da sem osvojil vse ukaze in registre kateri se nahajajo v različnih bankah. Ko mi je program uspelo prevesti, sem uporabil MP LAB-ov razhroščevalnik, da sem preberjal ali mi program deluje tako kot sem si ga zamislil.

Za vpis programa v mikrokontroler pa sem uporabil program ICPROG, katerega sem prav tako dobil na internetu. Programator sem si izdelal sam, saj je sorazmerno enostaven, priključi pa se ga na osebni računalnik preko COM vrat. Tudi program je zelo enostaven, saj samo odpremo šestnajstiško datoteko, ki smo jo ustvarili z MP LAB-om in jo vpišemo v napravo. Prav tako lahko nastavimo tudi nekaj parametrov kot so tip oscilatorja, WDT in druge.

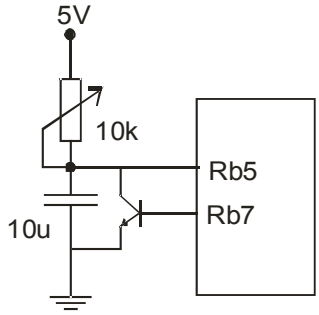
Podsklopi aplikacije

Poleg mikrokontrolerja v aplikaciji nastopajo tudi napajalno vezje, vezje ki določa čas koraka koračnega motorja močnostna izhodna stopnja, oscilator s 4MHz kristalom in tipke s katerim določamo smer vrtenja.

Napajalno vezje je sestavljeno iz gladilnih kondenzatorje in stabilizatorja napetosti 5V iz serije 78L05. To je integriran stabilizator, kateri ima dostopanje od nazivne napetosti +-5%. Izhodni tok je omejen na 100mA. Prav tako ima tudi termično zaščito in zaščito proti kratkemu stiku. Napajalna napetost stabilizatorja je 12V, pred stabilizatorjem pa je nameščen še gladilni kondenzator.

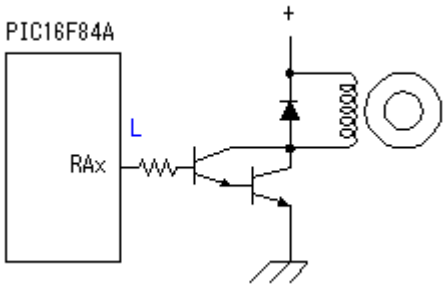


Vezje, ki določa čas koraka je v bistvu RC člen. Sestavljen je iz predupora, potenciometra, s katerim krmilimo število obratov, To počnemo tako, da spreminjamo RC konstanto člena, zato se kondenzator hitreje ali počasneje polni ali prazni. Napetost na kondenzatorju odčitavamo na RB5 vhodu mikrokontrolerja, praznimo ga pa s pomočjo tranzistorja, kateremu je baza priključena na izhod RB7 mikrokontrolerja. Hitrost koraka bi lahko spreminjali tudi s pomočjo spremenljivega kondenzatorja, lahko pa tudi programsko.

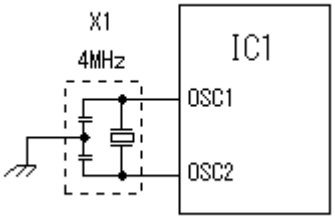


Močnostna izhodna stopnja je sestavljena iz integriranega vezja ULN 2003. Vsebuje z darlingtonovo vezavo povezane tranzistorje, ki lahko preklapljajo tokove do 500mA, pri napetosti do 50V. Slednje deluje kot stikalo, s katerim preklapljamo med navitji koračnega motorja. Tu moramo uporabiti tudi diodo, s katero preprečujemo, da bi inducirana napetost na motorju uničila integrirano vezje. Močnostni del vezja za razliko

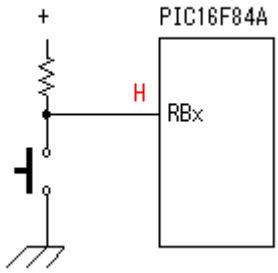
od logičnega dela deluje na napetosti 12V, saj je napetost 5V prenizka da bi z njo napajali koračni motor.



Oscilator je zgrajen iz 4Mhz quartz kristala in dveh kondenzatorjev s kapacitivnostjo 33pF. 4Mhz je izbran zaradi tega, ker ta aplikacija ne potrebuje hitrejšega takta.

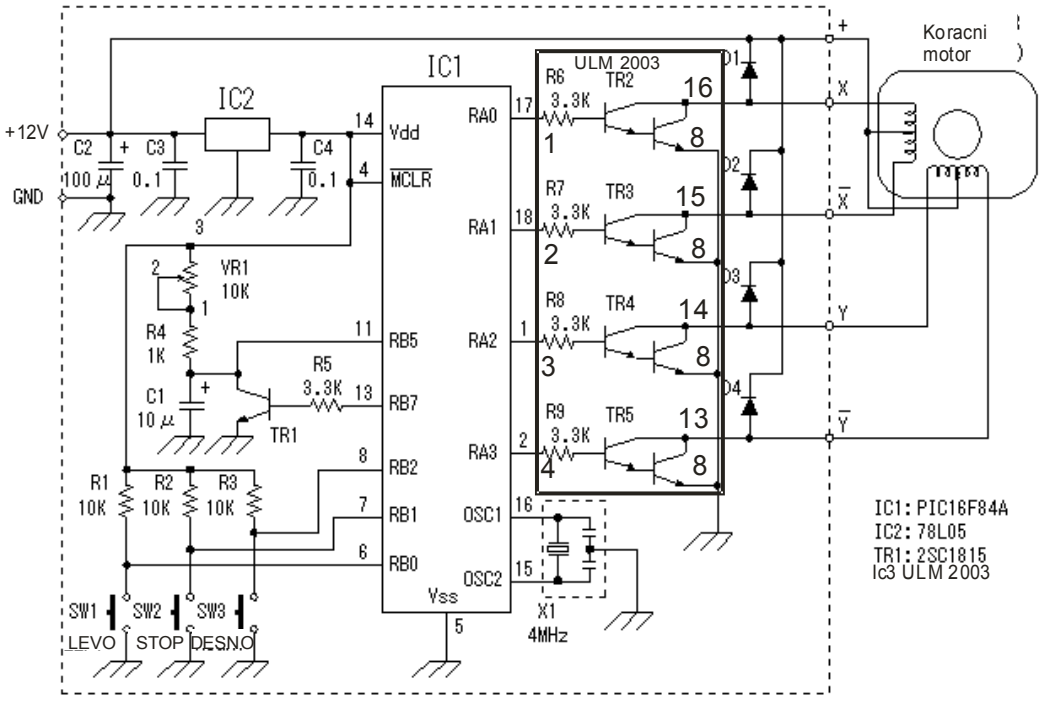


Tipke, s katerimi določamo smer vrtenja, to je levo, desno in stoj. Poleg tipk so tu uporabljeni tudi pull up upori, kateri povežejo vhod mikrokontrolerja na visoki nivo.



Shema, fizična izvedba in ekonomsko poročilo

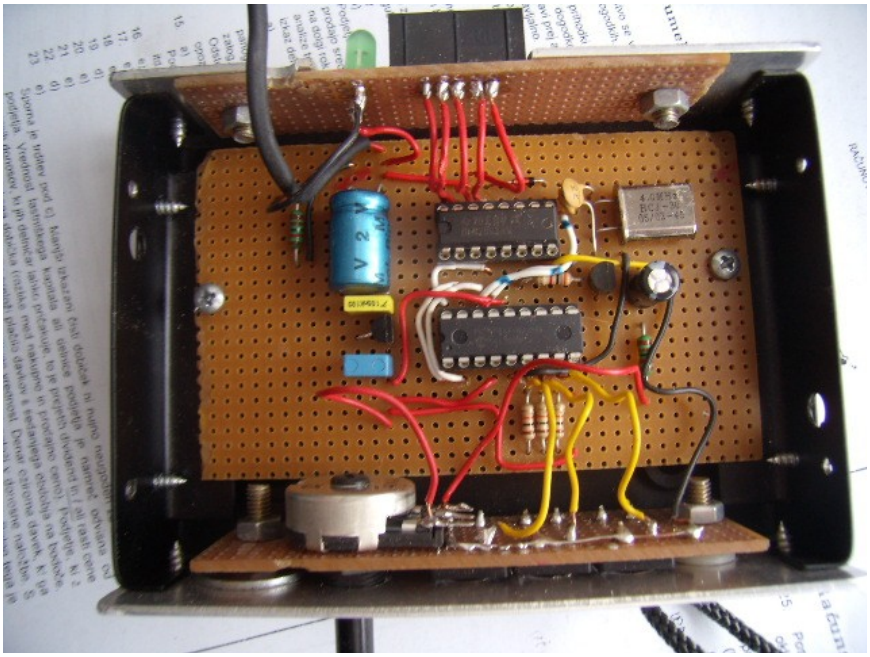
Kompletna shema vezja je predstavljena na naslednji sliki.



Fizična izvedba naprave

Vezje sem izdelal na univerzalni ploščici, vezave elementov sem naredil po spodnjem delu ploščice, nekatere tudi po zgornjem. Osnovne elemente vezja sem namestil horizontalno na ohišje, na prvo malo večjo ploščico. Stikala sem namestil na drugo ploščico katero sem namestil vertikalno glede na osnovno ploščico, to pa zato, da so stikala lepo nameščena na robu ohišja in so zato lahko dostopna za upravljalca naprave. Prav tako kot stikala sem namestil tudi potenciometer za upravljanje hitrosti vrtenja. Nato sem izdelal še eno ploščico, ki ja prav tako postavljena vertikalno, vendar na drugi rob ohišja. Ta pa je namenjena za pritrditev konektorja koračnega motorja, napajanje in signalno LED diodo, katera nam pove da je vezje v delujočem stanju. Vse tri ploščice sem namestil v aluminijasto ohišje dimenzij 110mm x 90mm x 30mm, katero se mi je zdelo ravno pravšnje za tako število elementov in velikosti le teh. Samo napravo pa bi fizično lahko izboljšal na ta način, da bi izdela tiskano vezje, vendar doma nisem imel vseh potrebščin za izdelavo le tega.

Pogled v notranjost naprave



Ekonomsko poročilo

Material sem nabavil v trgovini IC elektronika nekaj pa v trgovini Lotos v Kranju. Vse cene zajemajo 20% DDV.

1. IC PIC16F84A	DIP-18	1kos	9,26EUR
2. Preizkusna plošča	160 x100	1kos	2.86EUR
3. IC ULN 2003	DIP-16	1kos	0.76EUR
4. IC 78L05	TO 5	1kos	0.59EUR
5. Potenciometer	10k lin	1kos	1.99EUR
6. Kondenzator elco	100u	1kos	0.62EUR
7. Kondenzator elco	10u	1kos	0.40EUR
8. Kondenzator poliester	100n	2kos	0.58EUR
9. Konektor flat 10p ž		1kos	0.40EUR
10. Konektor flat 10p m		1kos	0.40EUR
11. Upori različni		4kos	0.28EUR
12. Kondenzator 33p		2kos	0.06EUR
13. LED dioda zelena		1kos	0.14EUR
14. Oscilator 4MHz		1kos	2.90EUR
15. Podnožje 16pin		1kos	0.38EUR
16. Podnožje 18pin		1kos	0.43EUR
17. Tipke		3kos	2.04EUR

18. Tranzistor 2SC1815	1kos	0.07EUR
19. Ohišje Al 110 x 90 x 30	1kos	13.80EUR
20. Banana KB6	2kos	2.98EUR
21. Potrošni material	1kos	3.76EUR

Cena vseh komponent: 43.72 EUR

Zaključek

Za konec te seminarske naloge lahko povem le to, da s samo izdelavo vezja nisem imel težav, več težav pa je nastopilo pri programiranju. Tu sem si pomagal s članki iz revije svet elektronike iz leta 2003, v katerih so opisani osnovni koraki pri programiranju s PIC mikrokontrolerjem. Programiranje mi je vzelo sorazmerno dosti časa, medtem ko sem fizični del opravil sorazmerno hitro.

Kazalo