Frekvenčni generator z mikrokontrolerjem

Avtor: mag. Vladimir Mitrović
E-pošta: svijet.elektronike@kc.htnet.hr

Frekvenčni generatorji generirajo signale različnih frekvenc in valovnih oblik. Odvisno od zanesljivosti, frekvenčnega razpona in drugih možnosti, ki jih nudijo, so lahko te naprave tudi precej drage. V tem poglavju bomo analizirali, kaj lahko dosežemo z mikrokontrolerji in Bascomom. Zaradi enostavnosti se bomo ukvarjali le s pravokotno obliko signala. Omenimo še to, da so za izdelavo takšne naprave bolj primerni AVR mikrokontrolerji; pri mikrokontrolerjih iz družine 8051 se časovnika ne da konfigurirati tako, da bi generiral spremembe stanja na nekem od pinov brez programske kontrole. Vendar je to zato tudi toliko večji izziv za Bascom programerja!

Poglejmo si najprej, kaj lahko dosežemo s "standardnimi" za to namenjenimi Bascom rutinami. Gre za ukaz Sound, o katerem v Helpu piše naslednje:

```
SOUND  pin, duration, frequency
pin        Kateri koli V/I pin, npr. P1.0
duration   Število pulzov, 1 - 32768
frequency  Trajanje stanja 0" ali "1" na
           izbranem pinu
```

Izbrani pin se zadrži v stanju "0" ali "1" frequency mikrosekund. Zanka se izvrši duration-krat.

Toda ni vse tako, kot piše! Analizirali smo Sound ukaz enako, kot to tudi sicer počnemo, ko nekaj "zaškripa": pred in za ukazom vpišemo kakšnih deset "nop" ukazov,

```
nop
...
nop
Sound P3.5 , 32000 , 1
nop
nop
```

nakar dobljeno heksa kodo prevedemo in disasembliramo... rezultat je prikazan v Tabeli 1.

Heksa koda	ukaz	trajanje
	ZANKA:	
12 00 2E	LCALL 002E	2+7
A8 02	MOV R0,R2	2
D2 90	SETB P3.5	1
D8 FE	DJNZ R0,*-0	2
C2 90	CLR P3.5	1
A8 02	MOV R0,R2	2
D8 FE	DJNZ R0,*-0	2
70 EF	JNZ ZANKA	2

Tabela 1: Asemblerski prevod ukaza Sound

Tabela 1 prikazuje samo najpomembnejši del rutine; preskočili smo uvodni del, v katerem se dani parametri prenesejo v registre R6, R7 (duration, število impulzov) oziroma R2 (frequency, trajanje posameznega stanja). Prav tako ni prikazan podprogram na naslovu 002E, v katerem parameter duration zmanjšamo za 1. Poglejmo si podrobneje naslednji segment:

```
MOV R0,R2
SETB P3.5
DJNZ R0,*-0
```

Tukaj se najprej želeno trajanje iz registra R2 prekopira v R0 in postavi pin P3.5 v stanje "1", nato pa se vsebina registra R0 zmanjšuje za 1 (DJNZ), vse dokler ne postane enaka 0. Iz tega lahko takoj opazimo bistvo problema: ker izvedba DJNZ ukaza traja 2 takta, je trajanje stanja "1" dvakrat daljše kot piše v Help-u! Če temu prištejemo še en takt, ki ga potrebujemo za izvršitev ukaza SETB, postane napaka še večja – če bi, kot smo to naredili v našem primeru, nastavili parameter frequency na 1, bi namesto pričakovane 1µs stanje "1" trajalo cele 2 x 1 + 1 = 3 µs.

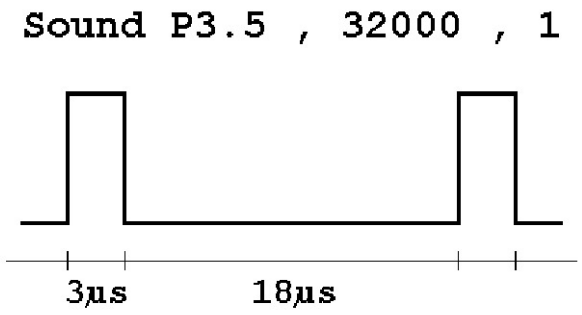
Nadaljujmo z analizo kode iz tabele 1. Sledi podoben segment, v katerem Bascom generira "0":

```
CLR P3.5
MOV R0,R2
DJNZ R0,*-0
```

Problem je podoben, le da se je tukaj pojavil tudi ukaz MOV, ki povzroča dodatno podaljšanje omenjene polperiode za 2 takta. Vendar to še ni vse! Za ukazom DJNZ sledi še skok na začetek zanke in klicanje podprograma na naslovu 2EH, v katerem se preverja, ali je zanka izvršena duration-krat (ker gre za 16-bitno število, ne moremo uporabiti enostavnega DJNZ ukaza). To k trajanju polperiode prinese še dodatne takte... v kratkem: stanje "0" traja dvakrat dlje + 16 µs, kar je veliko več, kot smo pričakovali. Ker svoji analizi enostavno nismo mogli verjeti, smo ukaz Sound vpisali v zanko

```
Do
   Sound P3.5 , 32000 , 1
Loop
```

in opazovali dobljeni signal z osciloskopom. Namesto pričakovanega signala frekvence 500 kHz, je osciloskop pokazal signal frekvence okoli 48 kHz (!), kot na sliki 1: stanje "1" je trajalo 3 µs, stanje "0" pa celih 18 µs, točno tako, kot je pokazala analiza. Moramo pa povedati, da gre za najslabši možni primer - če bi frekvenco zmanjševali, bi bila asimetrija signala vse manjša in manjša (vendar bi še naprej obstajala razlika med trajanjem "1" in "0"), frekvenca pa bi postala enaka približno polovici zadane. Ukaz Sound torej proizvaja nekakšen signal, na čigar frekvenco lahko vplivamo z njenimi parametri, vendar pa ne na način, ki je dokumentiran.



Slika 1: Signal, ki ga proizvaja ukaz Sound, precej odstopa od pričakovanega; v tem primeru smo pričakovali 500 kHz z enakim trajanjem "1" in "0".

Ali smo lahko natančnejši?

Ali lahko sami napišemo natančnejšo Sound rutino? Vsekakor! Za to sta nam na voljo dve možnosti: programiranje v asemblerju ali uporaba timerjev mikrokontrolerja. Oba pristopa imata tako svoje prednosti kot tudi pomanjkljivosti. Najprej bomo videli, kaj lahko storimo s čistim programiranjem; poglejmo si rutino Tone (tabela 2).

Rutina Tone je zelo podobna kodi, ki jo generira ukaz Sound, le da smo tukaj več pozornosti namenili pravilnemu timingu. Tudi tukaj je trajanje enega stanja ("0" ali "1") določeno z majhno zanko, v kateri se dekrementira vsebina registra R0; ukaz DJNZ R0 se ponavlja vse dokler vsebina registra R0 ni enaka 0. V našem primeru bo eno stanje trajalo 248 (začetna vrednost v R0) x 2 (trajanje enega DJNZ ukaza) = 496 µs. K temu je treba dodati še 4 µs, kolikor skupno potrebujejo ostali ukazi v "veliki" zanki (TONE1: ... DJNZ R1), zato bo skupno trajanje ene polperiode v našem primeru znašalo 500 µs. Z drugimi besedami, generirali bomo ton frekvence točno 1 kHz, signal pa bo popolnoma pravilen, ker se obe polperiodi generirata znotraj iste zanke. Število impulzov, ki jih generiramo, določa začetna vrednost registra R1 in bo v našem primeru 100 (polovica vpisane vrednosti).

Generirali smo torej ton frekvence 1 kHz v trajanju točno 100 milisekund! To je dosti bolj natančno od tistega, kar lahko dosežemo z ukazom Sound... Poglejmo, kako izračunamo potrebne parametre:

heksa koda	ukaz	trajanje
	TONE:	
79 C8	MOV R1,#200	1
	TONE1:	
78 F8	MOV R0,#248	1
B2 B5	CPL P3.5	1
	TONE2:	
D8 FE	DJNZ R0,.TONE2	2
D8 F8	DJNZ R1,.TONE1	2

Tabela 2: Analiza rutine Tone

- v R1 vpišemo število dvakrat večje od števila impulzov, ki jih želimo generirati;
- v R0 vpišemo vrednost, ki jo izračunamo na naslednji način: od trajanja ene polperiode v mikrosekundi odštejemo 4 in dobljeno vrednost delimo z 2.

Ali:
R1 = število_impulzov x 2
R0 = (trajanje_ene_polperiode_v_µs - 4) / 2

Kakšne so omejitve rutine Tone?

- **a)** rutina je napisana za osnovni takt frekvence 12 MHz (kot tudi vse ostale časovne rutine Bascoma); če uporabljamo kvarc druge frekvence, bodo dobljene vrednosti sorazmerno odstopale od pričakovanih;
- **b)** prekinitve katerega koli izvora (Timero ali 1, Into ali 1, Serial) "kradejo" procesorske cikle in s tem vplivajo na točnost generirane frekvence;
- **c)** maksimalna vrednost, ki jo lahko vpišemo v R0 in R1 je 255 (točneje 0; 0 = 256); ta določa minimalno frekvenco, ki jo lahko generiramo (R0 = 0: fmin ~ 969 Hz, R0 = 1: fmax ~ 83 kHz) ter maksimalno trajanje (R1 = 0: 128 impulzov);
- **d)** celo tistih frekvenc, ki se nahajajo znotraj "dovoljenega" obsega, ne moremo vedno natančno določiti; npr. za f = 20 kHz je trajanje ene polperiode 25 µs, zato bi v R0 morali vpisati (25 - 4) / 2 = 10,5; potrebno se je odločiti ali za R0 = 10 (f = 20833 Hz) ali pa za R0 = 11 (f = 19230 Hz);
- **e)** način vpisa parametrov je precej neroden, razen v primeru, ko želimo generirati manjše število vnaprej definiranih tonov.

Na omejitve pod a) in d) ne moremo vplivati, ker so pogojene z načinom delovanja mikrokontrolerja. Omejitvam pod b) se lahko izognemo, če med izvajanjem rutine Tone onemogočimo prekinitve. S preostalima dvema pripombama pa se lahko spoprimemo: večji razpon frekvenc in trajanja lahko dosežemo, če namesto registrov za vpis parametrov uporabimo spremenljivke tipa word, način vpisa parametrov pa lahko približamo standardnemu Bascom načinu delovanja (ne pa tudi popolnoma izenačimo s standardnimi Bascom ukazi).

Tone ali Sound?

Izboljšana Tone rutina je zapisana v datoteki TONE$SUB.BAS in jo najdete na priloženem CD-ju. Deloma je programirana v asemblerju, deloma pa v Bascomu. Tukaj je ne bomo podrobno analizirali; povejmo le, da je zasnovana na opisanih principih, osnovna razlika pa je v tem, da so parametri 16-bitni (word), kar jim omogoča veliko širši nastavitveni razpon. Prav tako je predviden uvodni del za izračun vrednosti parametrov, tako da se jih lahko vnaša v naravni obliki (ni več potrebno "ročno" računanje).

```
TONE trajanje , število_impulzov
trajanje trajanje enega impulza v us,
          36 - 65535
(byte ali word konstan  ta ali spremenljivka)
število_impulzov
    število impulzov (0-65535)
(byte ali word konstanta ali spremenljivka)
```

Opombe:
- frekvenca oscilatorja = 12 MHz
- če je trajanje število deljivo s 4, bo dobljeni signal pravilen
- če je trajanje sodo število, ki ni djeljivo s 4, se bo trajanje "1" in "0" razlikovalo za 2 µs
- če je trajanje liho število, ga bo rutina preračunala v prvo manjše sodo število
- število_impulzov = 0 pomeni, da se bo rutina izvajala v neskončnost
- Željeno frekvenco lahko preračunate v impulzni čas (trajanje enega impulza) po formuli

$$\text{trajanje} = \frac{1.000.000}{f}\,[\mu s]$$

nakar dobljeno vrednost zaokrožite na najbližje celo sodo število. Če želite, da namesto vas to opravi Bascom, namesto rutine Tone pokličite rutino Frek:

```
FREQ frekvenca , število_impulzov
frekvenca frekvenca v Hz, 16 - 25000
    byte ali word konstanta ali
    spremenljivka)
število_impulzov
    število impulzov (0-65535)
    byte ali word konstanta ali
    spremenljivka)
```

Opombe:
- rutina Frek preračuna zadano frekvenco v impulzni čas ter nato pokliče rutino Tone
- vseh frekvenc ni možno "okroglo" preračunati v čas, zato so možna tudi manjša odstopanja med zadano in generirano frekvenco
- če generirana frekvenca odstopa od zadane, se postavi Err bit
- veljajo vse opombe iz opisa rutine Tone

Seznam nekaterih frekvenc, ki se jih da natančno generirati: 16 Hz, 20 Hz, 25 Hz, 32 Hz, 40 Hz, 50 Hz, 80 Hz, 100 Hz, 200 Hz, 400 Hz, 500 Hz, 1 kHz, 2 kHz, 4 kHz, 5 kHz, 10 kHz, 20 kHz, 25 kHz, ...

Tone in Frek lahko štejemo za nova Bascom ukaza... pa vendarle, ker nista sestavni del Bascoma, se je za njihovo uporabo potrebno ustrezno pripraviti:
- na začetku programa vpišite ukaze

```
Dim Tone$us As Word , Tone$cy As Word
Tone$pin Alias Px.y
Declare Sub Tone(tone$us , Tone$cy)
Declare Sub Freq(tone$us , Tone$cy)
(Px.y je pin V/I vrat, na katerih že-
lite generirati signal, npr. P3.5)
```

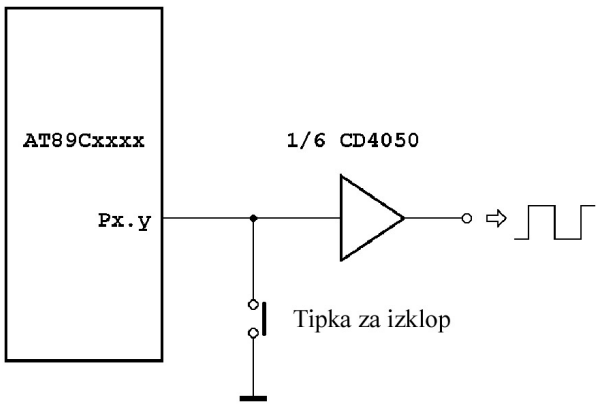
- na koncu programa (za ukazom End) dodajte vsebino datoteke Tone$sub.bas (mora biti v isti mapi/direktoriju, v katerem se nahaja glavni program)

```
$include Tone$sub.bas
```

Sedaj lahko ukaza Tone in Frek uporabljate tako, kot je zgoraj opisano. Sledi nekaj primerov:

```
Tone 60 , 1000    ' 1000 impulzov,
                  ' T = 60us, f = 16667Hz
' ...
Tone$pin = 0      ' če želimo, da se niz
                  začne z "1",
                  ' pred klicanjem Tone/Frek
                  postavimo
                  ' Tone$pin na "0" in obratno
Freq ff , us      ' ff, us: byte/word
                  spremenljivki
If Err = 1 Then   ' če f ni = ff, ...
' ...
Freq 1000 , 0     ' f = 1 kHz, stalno
```

Zadnji primer moram še malo razložiti... Kot že rečeno, če je parameter število_impulzov = 0, se bo trajno generiral



Slika 2:Princip frekvenčnega generatorja, ki uporablja rutini Time in Freq v neskončni zanki.

signal željene frekvence. To nam lahko pride prav, kadar s pomočjo mikrokontrolerja želimo narediti frekvenčni generator ali kaj podobnega. Vendar s tem program spravimo v neskončno zanko, iz katere se lahko 'reši' samo z resetom. To pa s programskega stališča ni najbolj posrečena rešitev! Zato smo v rutini Tone in Freq dodali še en majhen dodatek: če trajanje signala ni časovno omejeno, lahko izvajanje rutine prekinemo tako, da izhodni pin (Tone$pin) kratkotrajno vežemo na maso; obe rutini pričakujeta ta dogodek in ko ga odkrijeta, prekineta svoje izvajanje ter vrneta kontrolo glavnemu programu... sedaj lahko spremenimo frekvenco ali naredimo karkoli drugega, pač odvisno od tega, kam nas logika programa zanese.

Slika 2 prikazuje idejno rešitev frekvenčnega generatorja, ki uporablja opisano možnost. Reset tipka prekine izvajanje programa in povrne kontrolo glavnemu programu... Poglejmo, kako je to realizirano programsko v rutini Tone:

```
TONE:
   ...
   SETB TONE$PIN
   ...
   JB TONE$PIN,.TONE
RETURN
```

Enostavno, mar ne? Namesto ukaza DJNZ, ki smo ga uporabljali za štetje, ko smo hoteli proizvesti samo določeno število impulzov, smo tukaj uporabili ukaz JB, ki nas vrača na začetek zanke vse dokler je Tone$pin = 1. Seveda se testiranje pogoja dogaja v trenutku, ko bi Tone$pin moral biti "1"; če ni, pomeni, da je pin od zunaj kratko sklenjen na maso, zato se namesto vrnitve na začetek zanke izvede naslednji ukaz - Return.

Timer in Sound

Kako lahko s pomočjo timerja generiramo zvočni signal? Poglejmo si enostaven primer:

```
Tone$bit Alias P3.5
Tone$stop Alias P3.0
Config Timer0 = Timer , Gate = Internal ,
                              Mode = 2
On Timer0 Tim0 Nosave
Enable Timer0
Enable Interrupts
 ...
Load Timer0 , 100
Start Timer0
Petlja:
   jb tone$stop,.zanka
   Stop Timer0
 ...
Tim0:
   Tone$bit = Not Tone$bit
Return
```

Tukaj uporabljamo Timero v načinu 2, v katerem se začetna vrednost (=100) samodejno obnovi vsakič, ko timer prešteje zadano število procesorskih ciklov (to delo se opravlja na hardverskem nivoju). Istočasno se izvaja prekinitvena rutina Timo, znotraj katere invertiramo stanje izhodnega pina Tone$bit. V našem primeru se bo to zgodilo vsakih 100 µs, kar pomeni, da bo generiran signal frekvence 5 kHz.

Opazimo, da se vse to izvaja znotraj zanke, v katero program vstopi po definiranju začetne vrednosti (Load Timero) in zagonu timerja (Start Timero). Zanka je sestavljena iz enega samega asemblerskega ukaza (jb tone$stop,.zanka), ki se bo izvajal, vse dokler ne bomo spojili pin Tone$stop na maso.

Katere so prednosti ali pomanjkljivosti take rešitve? Človek bi pričakoval, da je hardverska rešitev vedno boljša in hitrejša od softverskega ekvivalenta. To je res, saj se stanje timerja spremeni vsako mikrosekundo, torej dvakrat hitreje kot je to bilo možno doseči s pomočjo ukaza djnz v rutinah Sound ali Tone. Vendar s tem še nismo rešili našega problema: ker program čaka na prekinitev v zanki, kjer je trajanje jb, kakor tudi vseh drugih ukazov za skok 2 µs, nam pri tem ne moreta pomagati niti Bascom niti asembler! Torej nismo dosegli praktično nobene izboljšave - točnost je še naprej na nivoju dveh mikrosekund. Nekatere stvari so celo nekoliko ušle nadzoru, ker se bo, ne glede na to, ali se je prekinitev zgodila v prvi ali drugi mikrosekundi izvedbe ukaza za skok, pridružena prekinitvena rutina izvršila šele, ko se bo ukaz za skok končal. To lahko pri določenih frekvencah povzroči rahlo nesimetrijo generiranega signala.

Izboljšanja se moramo torej lotiti na drug način. Poglejmo si naslednjo idejo:

```
Zanka:
   Idle
   jb tone$stop,.zanka
   Stop Timer0
```

Od prvega primera se razlikuje le po ukazu Idle, ki je tukaj dodan znotraj zanke. Bistvo spremembe je v tem, da sedaj program ne bo čakal v zanki (ki se izvede vsaki 2 µs) temveč bo "zaspal" pri ukazu Idle. Ker lahko prekinitev prekine Idle vsako mikrosekundo, smo s tem dosegli dvakrat večjo natančnost nastavitve frekvence! Ampak lahko opazimo, da tudi ukaza Idle in jb trajata nekaj časa (k čemur je potrebno dodati še skok v prekinitveno rutino Timo, njeno izvajanje in vrnitev; skupno 9 µs); vendar ne pozabimo, da se ti programski koraki izvajajo paralelno z delom timerja. Edini predpogoj je, da je začetna vrednost timerja (Load Timero) vsaj 9 ali več, da bi se teh nekaj instrukcij utegnilo izvršiti; če vnesemo vrednost med 1 in 8, ne bomo dosegli pričakovanega trajanja. (Zanimivo je, da je tudi pri Load Timero, 10, generator nerazložljivo kazal občasno nestabilnost v delovanju.)

Največja vrednost, ki jo lahko vpišemo v autoreload načinu (mode 2), znaša 255 (pravzaprav, 256: 256 = 0). Ta omejitev izhaja iz arhitekture mikrokontrolerja in ne iz Bascoma. Zato imamo na voljo sorazmerno majhno število frekvenc, ki jih lahko generiramo. Ali pa tudi ne?

Poskusimo simulirati autoreload s timerjem postavljenim v način 1. Potrebovali bomo eno spremenljivko tipa word, v katero bomo shranili začetno stanje timerja, ki jo bomo v preki-

nitveni rutini ponovno vpisovali v register timerja. Poglejmo si, kako bi naj to izgledalo.

```
Dim Trajanje As Word
Tone$stop Alias P3.0
Tone$bit Alias P3.7
Config Timer1 = Timer , Gate = Internal,
                                Mode = 1
On Timer1 Tim1 Nosave
Enable Timer1
Enable Interrupts
Trajanje = 10000          ' = Load Timer1,
                                    10000
Trajanje = 12 - Trajanje  ' razlaga v tekstu
Gosub Tim1
Zanka:
   Idle
   jb tone$stop,.zanka
Stop Timer1
 ...
 ...
Tim1:
   Tone$bit = Not Tone$bit
   Counter1 = Trajanje    ' prog. simulacija
   Start Timer1           ' načina 2
Return
```

Primer je zelo podoben prejšnjemu; tukaj smo uporabili Timer1 (kar niti ni toliko pomembno), ki je postavljen v način 1, način 2 pa je simuliran programsko. Najprej smo v spremenljivko Trajanje naložili želeno začetno vrednost in nato malo računali. Ta izračun je potreben zato, ker timer vedno šteje navzgor; zato je namesto 10000, kot v našem primeru, potrebno vpisati 65536 - 10000 = 55536 (ta izračun pri načinu 2 neopazno opravi namesto nas ukaz Load). 65536 je za 1 preveliko število za spremenljivko tipa word, vendar lahko Bascom tudi malo prevaramo. Tako lahko želeni izračun dosežemo z ukazoma:

```
Trajanje = 10000
Trajanje = 0 - Trajanje
```

Pri tem o uspešno zamenja 65536. Vendar zakaj smo v primeru računali z 12? Za razlago poglejmo, kaj se dogaja v rutini Tim1. Po invertiranju pina Tone$bit, zaradi česar smo v rutino tudi prišli, moramo "ročno" napolniti register timerja in ga nato ponovno zagnati. V načinu 2 se je vse to dogajalo na hardverskem nivoju, vendar smo bili omejeni na razpon 1-256. V trenutku, ko ponovno zaženemo timer, je le-ta že preštel 12 ciklov – zato smo morali začetno vrednost korigirati na omenjeni način. Da gre ravno za 12, lahko ugotovimo z natančno analizo generirane ali z meritvijo.

Opisani postopek traja določeno število ciklov, zato bo pravilno deloval samo, če bo začetna vrednost spremenljivke Trajanje 17 ali več (pri prvi rešitvi je bila 9). No, to nas ne sme preveč motiti: "natančnost" je tudi tukaj 1 µs, možno pa je celo uspešno kombinirati oba postopka ter z malo programerske spretnosti tudi uporabljati isti timer v obeh primerih. Tako bomo lahko realizirali frekvence v razponu med 7,63 Hz (65536 µs) in 55,55 kHz (9 µs). "Navzgor" se lahko premikamo na dva načina: s povečanjem frekvence kvarca na 24 MHz ali z zankami, vendar lahko na ta način realiziramo le nekaj "okroglih" frekvenc. "Navzdol" se lahko premikamo z zmanjšanjem frekvence kvarca (kar istočasno upočasnjuje celoten odziv), z zunanjim delilnikom, vendar tudi s spretno uporabo samega mikrokontrolerja. Preiščimo podrobneje to zadnjo možnost.

Konfigurirajmo Timero tako, da bo v načinu 2 štel mikroprocesorske cikle in v prekinitveni rutini vsakih, recimo 50 ciklov, invertiral stanje pina P3.5.

```
Config Timer0 = Timer , Gate = Internal ,
                                 Mode = 2
On Timer0 Tim0 Nosave
Priority Set Timer0
Enable Timer0
Enable Interrupts
 ...
 ...
Load Timer0 , 50
Start Timer0
 ...
 ...
Tim0:
   P3.5 = Not P3.5
Return
```

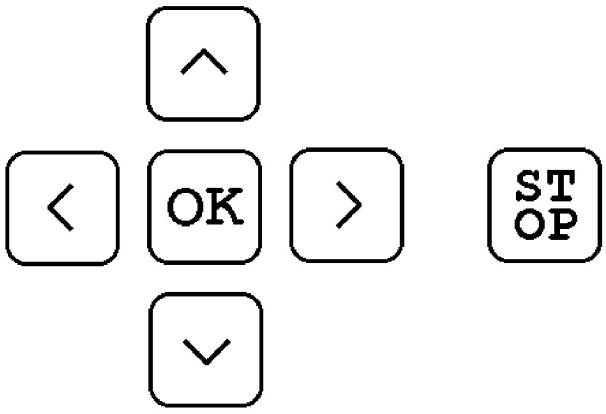
S tem smo na pinu P3.5 dobili signal frekvence, ki je 100-krat nižja od frekvence internega takta (1 MHz). Deljenja s faktorjem manjšim od 9 ne moremo realizirati (o tem smo že pisali), potrebno pa je biti pazljiv tudi s faktorji manjšimi od 20 (ker je tedaj mikroprocesor več časa v prekinitveni rutini kot pa v glavnem programu). To zmanjšano frekvenco lahko sedaj uporabimo kot vhod v Timer1, če ga konfiguriramo tako, da deluje kot števec (vhod je ravno pin P3.5):

```
Config Timer1 = Counter , Ga-
te = Internal , Mode = 1
On Timer1 Tim1 Nosave
```

Timer1 bomo uporabili na že opisan način, s primerno izbiro faktorja deljenja pa lahko precej znižamo spodnjo mejno frekvenco. Veliko sem eksperimentiral s timerji, tako da vem, da ta ideja zagotovo deluje - potrebno jo je samo nekoliko prilagoditi konkretni uporabi, vendar to prepuščam vam.

Frekvenčni generator

Končno smo prišli do cilja! Ob uporabi prej opisanih princi-



Slika 3: Tipke za navigacijo skozi programske menije.

pov in z njihovim uvrščanjem v ustrezni program bomo naredili natančni frekvenčni generator. Pravzaprav gre za dva programa - G_FREK_24.BAS in G_FREK_18.BAS – ki se nahajata na priloženem CD-ju in ju uporabite v originalu ali pa kot osnovo za modifikacije glede na lastne potrebe. Tukaj se ne bomo spuščali v podrobno opisovanje programske kode, ker smo vse osnovne principe že preanalizirali, temveč bomo nekaj prostora namenili programski zasnovi in načinu uporabe. Povejmo le, da sta oba programa rešena na principu štetja, torej brez uporabe timerja. Za to smo se odločili, ko nam je uspelo sestaviti zelo enostaven programski trik, s katerim se natančnost generiranja frekvence z dveh popravi na en cikel mikrokontrolerja in s tem izenači z možnostmi, ki nam jih omogočajo timerji; zainteresirane bralce vabim, da sami analizirajo rutini Frek_1 in Frek_2 programa F_gen_24.bas.

Predvideno je, da generator proizvaja določeno število fiksno zadanih frekvenc (okoli 50 frekvenc) v področju med 1 Hz in 500 kHz. Program je interaktiven in za komunikacijo z uporabnikom uporablja alfanumerični prikazovalnik (16*1) ter vrsto navigacijskih tipk, kot je nazorno prikazano na sliki 3. S tipkama "gor" in "dol" se pomikamo po nivojih menija:

```
... 7 <-> 1 <-> 2 <-> 3 <-> 4 <-
> 5 <-> 6 <-> 7 <-> 1 ...
```

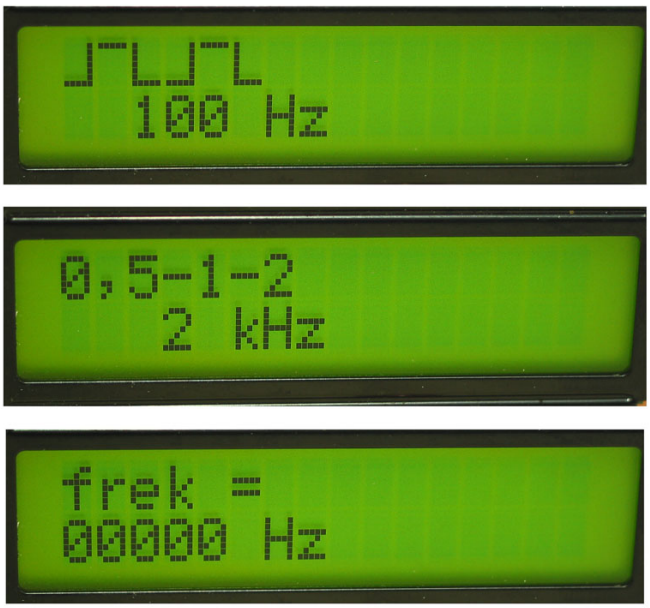
Posamezni nivoji omogočajo naslednjo izbiro frekvenc:

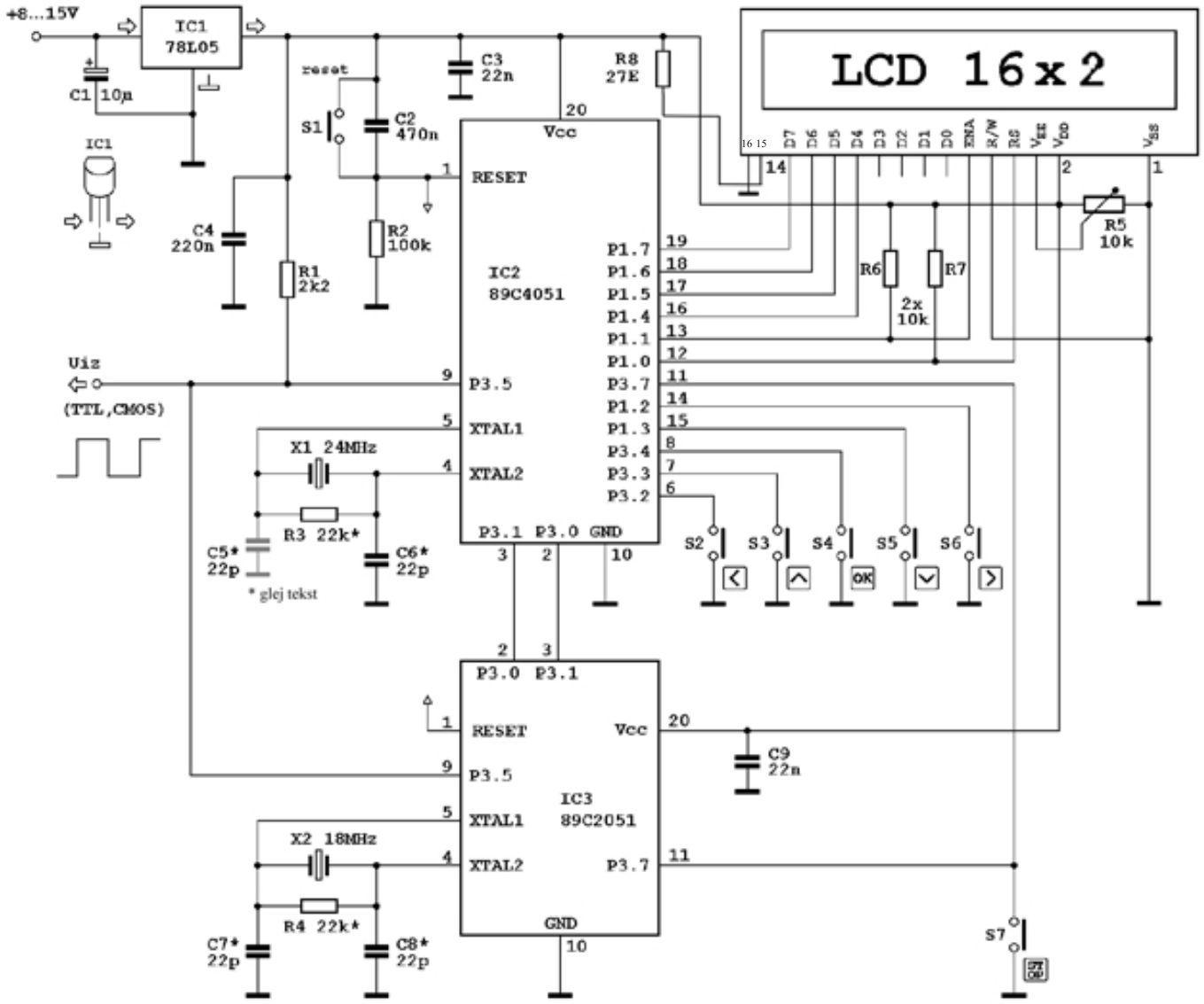
- **1.** nivo: raster 1 : 2 : 5
  - prikaz na prikazovalniku: 1-2-5 1 kHz
  - generirane frekvence: 1 - 2 - 5 - 10 - 20 - 50 - 100 - 200 - 500 Hz, 1 - 2 - 5 - 10 - 20 - 50 - 100 - 200 - 500 kHz
- **2.** nivo: oktavni raster 0,5 : 1 : 2
  - prikaz na prikazovalniku: 0,5-1-2 1 kHz
  - generirane frekvence: 15,63 - 31,25 - 62,5 - 125 - 250 - 500 Hz - 1 - 2 - 4 - 8 - 16 kHz
- **3.** nivo: dekadni raster 1 : 10
  - - prikaz na prikazovalniku: 1-10-100 1 kHz
  - - generirane frekvence: 1 - 10 - 100 Hz - 1 - 10 - 100 kHz
- **4.** nivo: dekadni raster 2 : 20
  - prikaz na prikazovalniku: 2-20-200 2 kHz
  - generirane frekvence: 2 - 20 - 200 Hz - 2 - 20 - 200 kHz
- **5.** nivo: dekadni raster 5 : 50
  - prikaz na prikazovalniku: 5-50-500 5 kHz
  - generirane frekvence: 5 - 50 - 500 Hz - 5 - 50 - 500 kHz
- **6.** nivo: fini raster
  - prikaz na prikazovalniku: 1-2-3-4 1 kHz
  - generirane frekvence: 1 - 1,5 - 2 - 3 - 4 - 5 - 6 - 8 - 10 - 15 - 20 - 30 - 40 - 50 - 60 - 80 - 100 - 150 - 200 - 300 - 400- 500 - 600 - 800 Hz - 1 - 1,5 - 2 - 3 - 4 - 5 - 6 - 8 - 10 - 15 - 20 - 30 - 40 - 50 - 60 - 80 - 100 - 150 - 200 - 300 - 400 500 kHz
- **7.** nivo: prosto izbiranje frekvence
  - prikaz na prikazovalniku: frek = 00000 Hz
  - frekvenčni razpon: 1 Hz - 80 kHz

Na vsakem nivoju s tipkama "levo" in "desno" izbiramo eno od ponujenih frekvenc. S pritiskom na tipko "OK" zaženemo, s tipko "STOP" pa prekinemo generiranje signala.

Izjema je sedmi nivo (prosto izbiranje frekvence). S prvim pritiskom na tipko "OK" dobimo možnost vpisa petmestnega števila, ki predstavlja želeno frekvenco: s tipkama "levo" in "desno" se premikamo po njegovih cifrah, s tipkama "gor" in "dol" pa korigiramo vrednost trenutno izbrane cifre. S ponovnim pritiskom na tipko "OK" se začenja generiranje izbrane frekvence. S pritiskom na tipko "STOP" prekinemo generiranje signala in omogočimo spremembo nivoja (tipki "gor" in "dol" ponovno dobita svojo staro funkcijo sprehajanja po nivojih).

Lahko opazimo, da gre na nivojih 1-6 za nabor enakih frekvenc, ki so grupirane na različne načine. Tako so na nivoju 1





Slika 4: Shema frekvenčnega generatorja: vezje je zelo enostavno, ker mikrokontrolerja opravita vse, kar je potrebno.

frekvence razvrščene s korakom 1 : 2 : 5, na nivoju 2 s korakom 0,5 : 1 : 2 (oktavno), na nivojih 3 - 5 z dekadnim korakom 1 : 10 : 100 itn. To nam omogoča hitro in enostavno izbiranje želene frekvence in njeno spreminjanje, pač odvisno od tega, kaj želimo doseči. Naj vas spomnim, da so glede na "digitalni" način generiranja (trajanje ene periode je mnogokratnik trajanja enega mikroprocesorskega cikla) frekvence vseh generiranih signalov zelo natančne (natančnost je odvisna od natančnosti kvarčnega kristala).

To je istočasno tudi razlog, zaradi katerega ni možno generirati vsake želene frekvence na nivoju 7. Približno do frekvence 1 kHz je možno izbrati vsako frekvenco (korak 1 Hz), pri čemer bo odstopanje med zadano in dejansko frekvenco manjše od 0,1 %. Pri frekvencah višjih od 1 kHz se korak postopoma povečuje, medtem ko pri frekvencah nad 10 kHz raste hitro in doseže nekaj kHz. V program je vgrajena rutina za preverjanje natančnosti, ki skrbi za nastalo odstopanje. Če npr. izberemo frekvenco 3000 Hz, je najbližja točna frekvenca, ki jo lahko generiramo, 3003,003 Hz. Ko s tipko "OK" sprožimo delovanje generatorja, bo program izpisal, da generiramo 3003 Hz in ne želenih 3000 Hz.

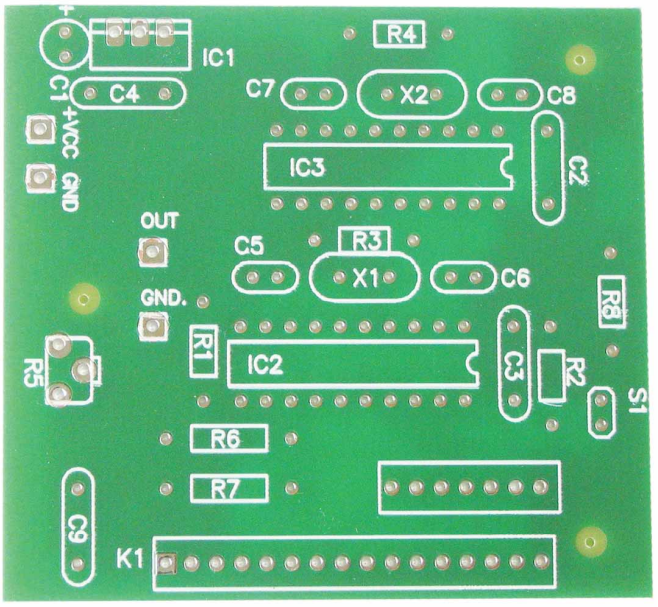
Ta problem je samo na sedmem nivoju, na katerem prosto izbiramo frekvenco. Na vseh ostalih nivojih smo prisiljeni izbirati med omejenim številom razpoložljivih frekvenc, vendar so zato vse frekvence zelo natančne. Da bi to dosegli, smo uporabili dva trika:

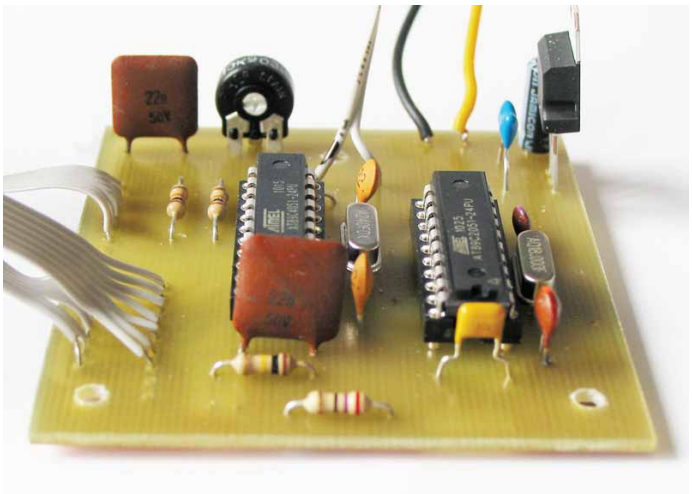
- za nekatere frekvence (16 kHz, 80 kHz, 500 kHz...) je napisana posebna rutina; generirani signal je nekoliko "nesimetričen" (trajanje "1" in "0" se razlikuje za 1 - 3 µs), vendar je frekvenca točna;
- mikrokontroler deluje na dveh frekvencah: 24 in 18 MHz; delovanje na 18 MHz omogoča natančno generiranje frekvenc 15 Hz, 30 Hz, 60 Hz in njihovih dekadnih mnogokratnikov, medtem ko lahko vse ostale frekvence realizi-

ramo s 24 MHz kvarcom.
Vezje je zelo enostavno, pravzaprav minimalistično (slika 4) - poleg dveh mikrokontrolerjev so uporabljeni samo še stabilizator napetosti, alfanumerični prikazovalnik in šest navigacijskih tipk. Vse potrebne funkcije se izvajajo znotraj samih mikrokontrolerjev. Lahko bi vse skupaj realizirali tudi s samo enim mikrokontrolerjem, vendar se je, ko se je pojavila potreba po dveh delovnih frekvencah, pojavil tudi problem preklapljanja dveh kvarčnih kristalov. Po več poskusih sem ugotovil, da je najenostavnejša rešitev uporaba drugega mikrokontrolerja. S tem smo dobili tudi nekaj dodatnega programskega prostora, zaradi česar je lahko program elegantnejši za uporabo ter natančnejši. Mimogrede naj omenim, da so Bascom rutine za delo z long spremenljivkami precej 'zapravljive', tako da je skupna programska koda večja od 4 kB. Mikrokontroler, ki deluje s 24 MHz, opravlja večino dela: upravlja z navigacijskimi tipkami in prikazovalnikom, generira večino frekvenc, upravlja z delom drugega mikrokontrolerja. Drugi, pomožni mikrokontroler proizvaja samo kakšnih 15 kritičnih frekvenc. Komunikacija med mikrokontrolerji je standardna serijska, za kar imamo v Bascom vgrajene potrebne rutine. Glede na velikost programa moramo kot glavni mikrokontroler (IC2) uporabiti AT89C4051, kot pomožni (IC3) pa bo zadostoval tudi AT89C1051 ali AT89C2051.

Generirani signal je pravokotne oblike, simetričen, amplitude 5 Vpp. Pull-up upor R1 pospešuje prehod iz stanja "0" v stanje "1". Takšen signal je tako TTL kot tudi CMOS kompatibilen ter se ga lahko direktno uporabi v različnih digitalnih vezjih. Seveda je možna njegova uporaba tudi v analognih vezjih, kolikor to dopušča pravokotna oblika signala. Takrat bi morali predvideti tudi ustrezni regulator amplitude. Preoblikovanje pravokotnega signala v sinusni ni ravno enostavno. Če gre za eno ali le nekaj frekvenc, lahko to dosežemo s pazljivim filtriranjem. Specialna analogna integrirana vezja – funkcijski generatorji –, kot so ICL8038, XR2206 ali v zadnjem času zelo popularni MAX038, rešujejo ta problem na drugačen način: pri njih pravokotni signal ni osnovna valovna oblika, temveč je to trikotna, ki se jo potem s tranzistorsko-uporovnim limiterjem preoblikuje v signal sinusne valovne oblike. Naš generator tega ne zmore, vendar zato proizvaja signal zelo natančne frekvence. Lahko ga uporabljamo npr. za kalibriranje nekega analognega generatorja ali osciloskopa ali za merjenja na različnih napravah.

Razložiti moramo še vlogo posameznih komponent, razporejenih okoli kvarčnih kristalov X1 in X2. Kot sem že razložil, sem se odločil za dve delovni frekvenci: 24 in 18 MHz. Vendar ko sem prvič poskusil delovanje na 24 MHz, se stvari niso odvijale tako, kot sem pričakoval: namesto s 24, je mikrokontroler zaosciliral z 8 MHz. Radioamaterji in vsi ostali, ki so se ukvarjali z VF oscilatorji, zagotovo poznajo razlog: kvarčne kristale za višje frekvence včasih obdelujejo tako, da ne delujejo na osnovni frekvenci, ampak na nekem izmed harmonikov. Ti overtonski kristali so veliko cenejši, zato je možnost, da boste na njih naleteli v trgovini, tudi toliko bolj verjetna. Takšen kvarc bo torej osciliral na osnovni frekvenci (8 MHz), na tretjem harmoniku (24 MHz) in morda še pri kateri frekvenci, pač odvisno od tega, kako ga bomo vzbujali. Rešitev problema sem našel na Bascomovi medmrežni skupini: svetovali so mi, da spojim paralelno kvarcu upor in ustrezno nastavim vrednosti kondenzatorjev v krogu oscilatorja. Po nekaj poskusih je moj kvarc veselo zanihal na želeni frekvenci. Ker nisem imel pri roki originala, mi je celo uspelo prisiliti "navadni" 6 MHz kvarc, da je zaosciliral na drugi zame zanimivi frekvenci -18 MHz. Možno je, da bo za nek drug kvarc potrebno nekoliko prilagoditi vrednosti R3, R4, C6 in C8. Po mojih izkušnjah bo najbolje, če v tem primeru C5 in C7 izpustite. Če vam uspe nabaviti kristale, ki imata osnovni frekvenci 18 MHz in 24 MHz, celotna opisana zgodba odpade, kristale pa boste vezali na "klasičen" način s C5 oziroma C7 in brez R3 in R4. V tem primeru boste lahko z rahlo korekcijo vrednosti kondenzatorjev C5 - C8 "dosegli" delovno frekvenco točno želene vrednosti - za to potrebujemo kakovosten merilnik frekvenc. Vendar je tudi brez uglaševanja natančnost za amaterske potrebe več kot zadovoljiva. Pri overtonskih kristalih povzročata R3 in R4 manjša odstopa-



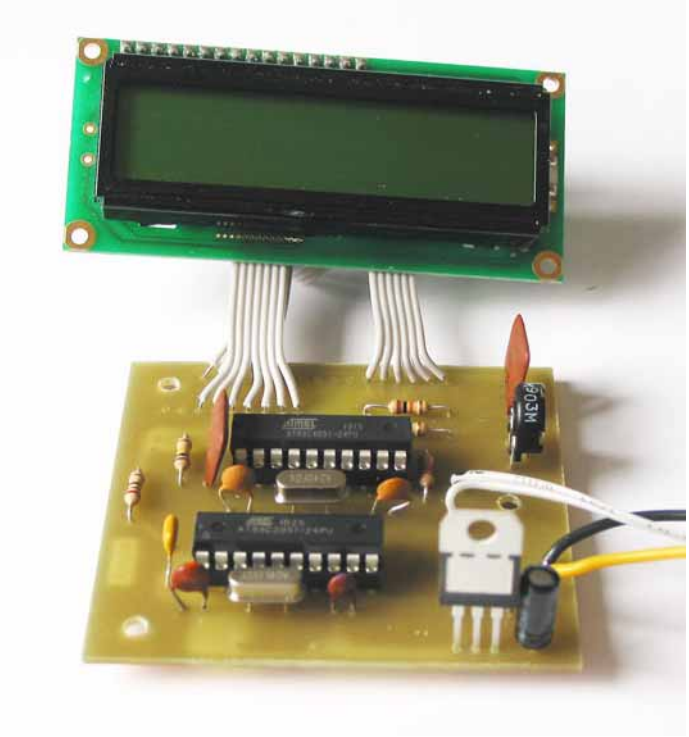


Slika 5: Razpored elementov za na tiskano vezje

nja delovne frekvence od želene vrednosti, sam oscilator pa je nekoliko kritičnejši - v tem primeru s kondenzatorji ne moremo vedno doseči točne frekvence. Pa vendarle odstopanja niti v tem primeru ne presegajo 0,1 %.

Načrt tiskanega vezja je prikazan na sliki 5. Celoten frekvenčni generator je nameščen na enem tiskanem vezju. Tiskano vezji je projektirano za miniaturne impulzne tipke, zato boste, če boste hoteli uporabiti tipke drugačnih izvedb. Ali pa enostavno namestite tipke na univerzalno tiskano vezje v želenem razporedu in jih povežite s koščki žice. Razen kvarčnih kristalov, o katerih smo že govorili, ostale komponente niso kritične; vsi večji kondenzatorji (C2 - C4, C9) so lahko z dvojnim (RM5) ali trojnim (RM7,5) razmakom priključkov.

Opomba: F_gen_24 in F_gen_18 sta napisana za BascomLT, zato v Bascom-8051 ne bosta dobro delovala. Razlika je pravzaprav v načinu, kako ta dva programa shranjujeta spremenljivke tipa long v pomnilnik mikrokontrolerja. Ker posamezne byte long spremenljivk uporabljamo neposredno iz asemblerja, smo se pač moral odločiti za eno izmed verzij programa.•



NASTAVITVE | IDENTI

Kosovnica

Podjetje AX, d.o.o.
Špruha 33
1236 TRZIN

Uporabnik SG - SAMO GREGORČIČ
Datum izpisa 5.7.2016, 13:22:20
Ident 5KIT0060 - KIT FREKVENČNI GENERATOR_M, vse za tiskanino
Cena 1 Nabavna
Datum tečaja od 5.7.2016
Vrsta identov 100, 200, 201, 300, 400, 500, 505, 600, 700, 701

svet ELEKTRONIKE

Navodila so v virtualni trgovini!

Nivo	Ident	Opombe	Koda	EM
5KIT0060	**KIT FREKVENČNI GENERATOR_M, vse za tiskanino**		**1,0000**	**KOS**
1	DISPLAY 2x16 OSVETLITEV LCD (HY_1602F)		1ELD0071	KOS
1	TIPKA DTS6-visoka	S1	1CON0112	KOS
1	CPU AT89C2051-24PU - DIL	Program M2	5ELU0142	KOS
1	CPU AT89C4051-24PU - DIL	Program M4	5ELU0143	KOS
2	PODNOŽJE DIL20		1CON0033	KOS
1	KRISTAL 18.000 MHz		1ELQ0004	KOS
1	KRISTAL 24.000 MHz		1ELQ0006	KOS
1	IC LM7805-TO ohišje		1ELC0055	KOS
1	KOND.EL. 10uF/50V		1ELC0006	KOS
2	KOND.KER. 22N		1ELC0039	KOS
1	KOND.KER. 220N		1ELC0034	KOS
1	KOND.KER. 470n		1ELC0009	KOS
4	KOND.KER. 22p	* beri tekst	1ELC0094	KOS
1	UPOR 2K2, 1/4W		1ELR0077	KOS
1	UPOR 100K, 1/4W		1ELR0010	KOS
2	UPOR 22K, 1/4W	* beri tekst	1ELR0066	KOS
2	UPOR 10K, 1/4W		1ELR0200	KOS
1	UPOR 27E, 1/4W		1ELR0073	KOS
1	UPOR TRIM 10K stoječi		1ELR0191	KOS
1	TIV FREKVENČNI GENERATOR_M		1TIV0087	KOS
1	SPONKE STV ZA TIV 2-pola		1CON0039	KOS
40	LETEV ENO. 40 PIN - M - 1 pin	7 za tipke, 2 x 16 za LCD	1CON0057	KOS
0,3	KABEL FLET 10 ŽILNI, 0.14, AWG28	2x10 cm za LCD, 10 cm za tipke	1KAB0017	MET