# Mikroračunala na lak način

## (1. Dio)

### Uvod

Priručnik je namijenjen razumijevanju osnovnog tečaja programiranja mikrokontrolera. Može se upotrijebiti za izvođenje laboratorijskih vježbi iz predmeta mikroračunala i sličnih predmeta.

Svi navedeni primjeri već su više puta isprobani i provjereni u praksi. Da bi primjere mogli provjeriti i razumjeti dobro bi bilo imati slijedeću 'opremu':

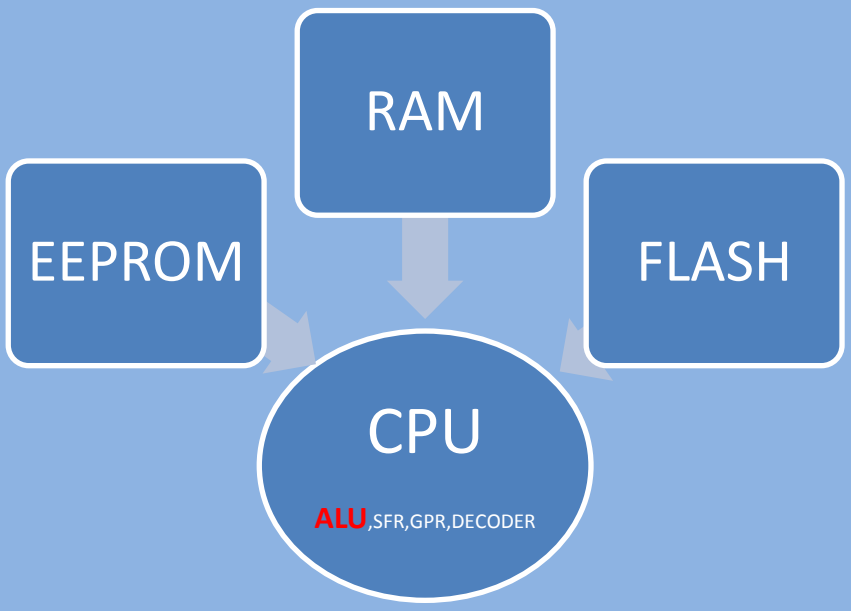
- BASCOM 8051 demo
- Programator PG302 ili sličan za 8051 skupinu
- Testnu ploču za 89C2051 , 89C4051 ili 89C1051
- *☺ ideju što napraviti s tim znanjem kasnije

Za one koji rade laboratorijske eksperimente prema ovoj knjizi ciljevi bi trebali biti slijedeći:

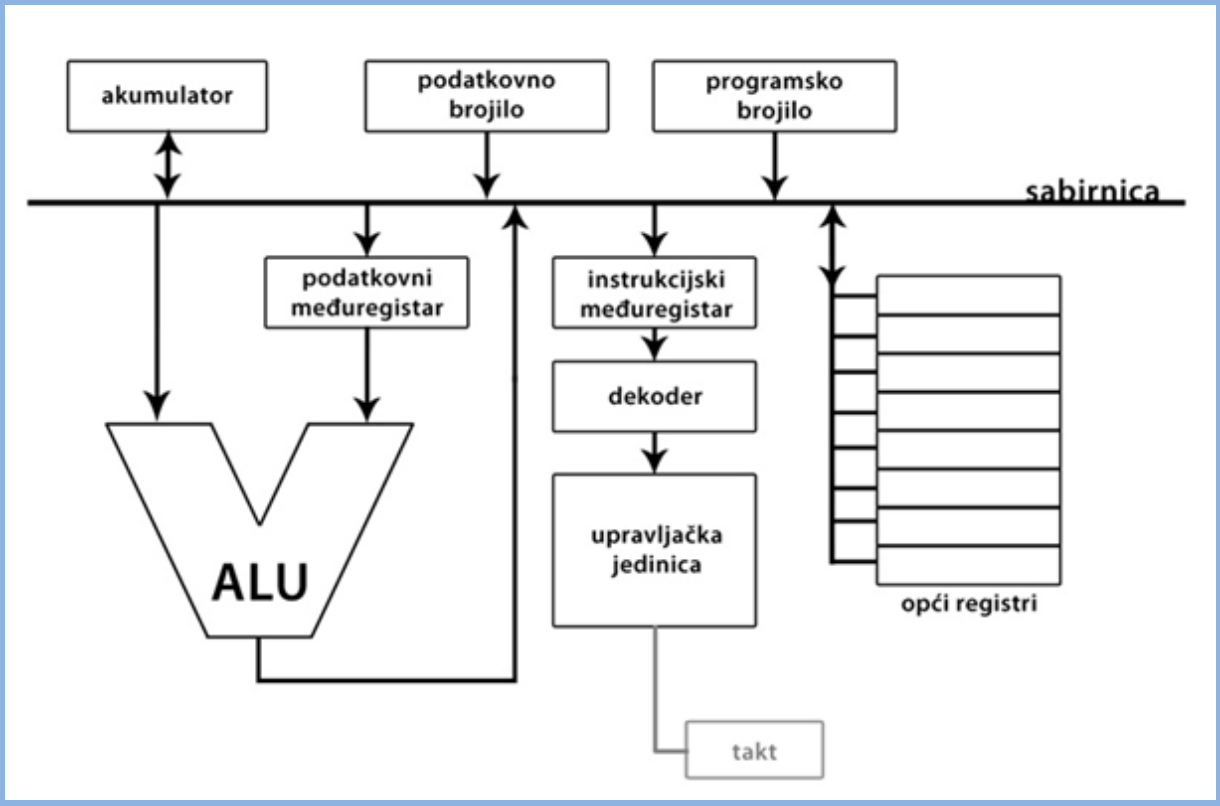
Naučiti:

- Upotrebljavati softver za programiranje mikroračunala
- Pisati Bascom-ove naredbe
- Pisati asemblerske naredbe
- Spojiti programator na PC
- Umetnuti mikrokontroler pravilno u PG302
- Izvršiti transfer HEX datoteke u mikroračunalo
- Spojiti mikroračunalo na eksperimenalnu pločicu
- Pokrenuti dobiveni sklop na eksperimentalnoj pločici, provjeriti da li radi kako je predviđeno
- Napraviti elektronički sklop baziran na mikroračunalu

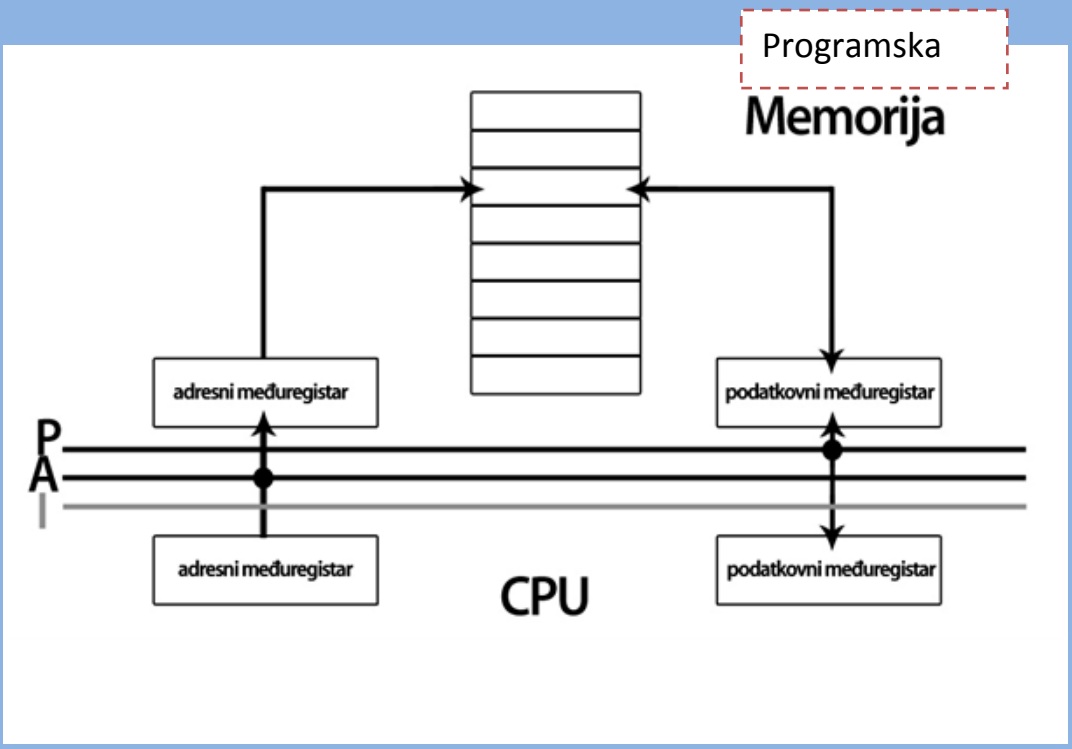
# Arhitektura mikroračunala




CPU izvršava naredbe,odrađuje aritmetičke i logičke operacije. Naredbe dolaze iz FLASH memorije gdje su trajno pohranjene u binarnom obliku. Sabirnica povezuje FLASH i CPU. RAM može biti vanjski i unutrašnji. Obično manji mikrokontroleri imaju unutrašnji RAM i to im je dovoljno za izvršavanje zadanih programa. Neki mikrokontroleri mogu adresirati i vanjski RAM. EEPROM služi za pohranu podataka koji nastanu prilikom rada mikroračunalala na ciljnom mjestu. Svojstva EEPROMA omogućuju da se podaci spremaju dok uređaj radi ali i da se pamte kad je uređaj isključen. Primjetit će te da se ponekad koristi blok shema 8051, a ponekad 89C2051. To je zato što se ta dva modela razlikuju samo u nekim detaljima i te pojedinosti bit će više puta razmatrane i spomenute s ciljem da se uoče razlike među modelima mikroračunala iz iste skupine.




Na gornjoj slici prikazana je unutrašnjost CPU jedinice. Vidimo da je jezgra procesora ALU. Naredbe iz FLASH memorije stižu preko sabirnice. Često uz naredbu trebaju stići i podaci koje će naredba obraditi (operandi). Naredbe koje su u binarnom obliku zapisane u FLASH-u moraju se dekodirati da bi ih razumjela ALU. Akumulator je obično 8-bitni registar koji služi za kratkotrajnu pohranu rezultata prije nego se oni premjeste u RAM. Aritmetičko logička jedinica vrši operacije koje proizlaze iz naredbi koje stižu iz programske memorije (FLASH). Naredbe su napisane u asembleru u skladu s arhitekturom odabranog procesora. Takve naredbe dekodira dekoder i tek onda ih izvršava ALU. Dekoder može biti hardverski sklop ali može sadržavati i program koji se ugrađuje tvornički (microcode).




Rečeno je da programska memorija sadrži program. Pretpostavimo da je program jednostavan i da je smješten na početku memorijskog polja od početne adrese do npr. adrese 3E8H. Naredbe se izvode u ALU a memorija je dislocirana (naravno izvan ALU). Preko sabirnice naredbe se očitavaju jedna po jedna počevši od prve do zadnje redosljedom koji je određen u programskom brojilu. Iz gornje sheme vidljivo je da preko iste sabirnice kolaju različiti podaci u različitim smjerovima. Zato sabirnica ima i upravljački dio koji regulira promet. Inače preko sabirnice mogu ići podaci, naredbe ali i adrese.

Zaključimo dakle slijedeće:

- Flash je programska memorija i sadrži program (firmware)
- Taj program je napisan u asembleru
- Da bi jezgra izvršila program trebaju postojati sabirnice za vezu između ALU i memorija
- Rezultati izvršenih naredbi pohranjuju je najprije u ACC a kasnije u RAM-u
- Po potrebi rezultati se pohrane u EEPROM

# Arhitektura 8051

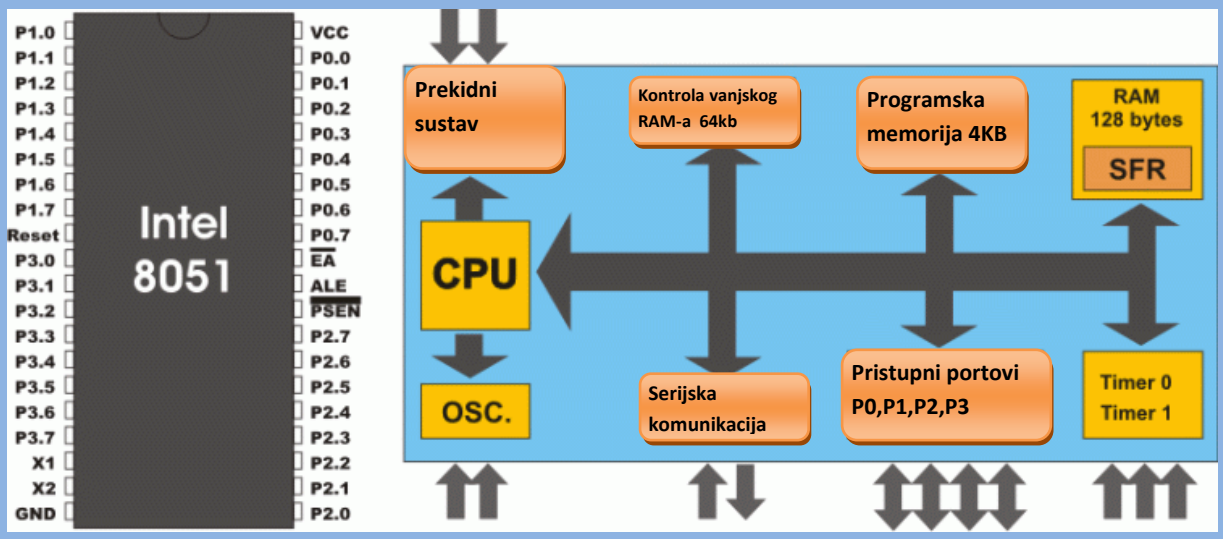
Arhitektura mikroprocesora podrazumjeva konfiguraciju registara i ALU unutar CPU , te instrukcijski set asemblerskih naredbi. Asembler je programski jezik niske razine i može se pisati binarno, heksadekadski ali i u obliku mnemonika. Ovaj zadnji način je uobičajen za većinu programera. Svaka arhitektura ima specifičan instrukcijski set.

Klasična arhitektura podudara se s Von Neumanovim modelom računala gdje postoji :

- Ulazni dio
- Izlazni dio
- Memorija
- ALU
- Upravljački sklop

Kod ovog modela memorija nije podjeljena na programsku i podatkovnu.Danas se na tržištu mikroračunala plasiraju dvije arhitekture: RISC i CISC. Prva predstavlja reducirani set instrukcija ali zato veći broj registara. Druga arhitektura ima kompleksan set instrukcija ali zato smanjen broj registara. Treba spomenuti i podatak da za određeni model procesora imamo fiksan broj instrukcija. Za skupinu 8051 npr. navedeno je 110 asemblerskih naredbi.

# Skupina 8051




Ovdje vidimo strukturu 8051. Mikroračunalo ima četiri pristupna porta, 4 kB flash memorije, najmanje 2 tajmera, unutrašnji RAM od 128B ili 256B, sustav za serijsku komunikaciju, prekidni sustav i naravno CPU. Upravo ovdje je vidljiva razlika između procesora(CPU) i mikroračunala.

Došli smo do točke kad trebamo odabrati konkretni mikrokontroler da bi smo ga proučili do detalja. Zašto? –Da bi smo mogli:

1. Napisati program
2. Izvršiti kompajliranje
3. Napraviti simulaciju
4. Transferirati program u mikroračunalo
5. Pokrenuti mikroračunalo
6. Provjeriti da li uređaj izvodi ono što smo htjeli

Na širokom tržištu chipova mikroračunala izdvajamo nekoliko modela:

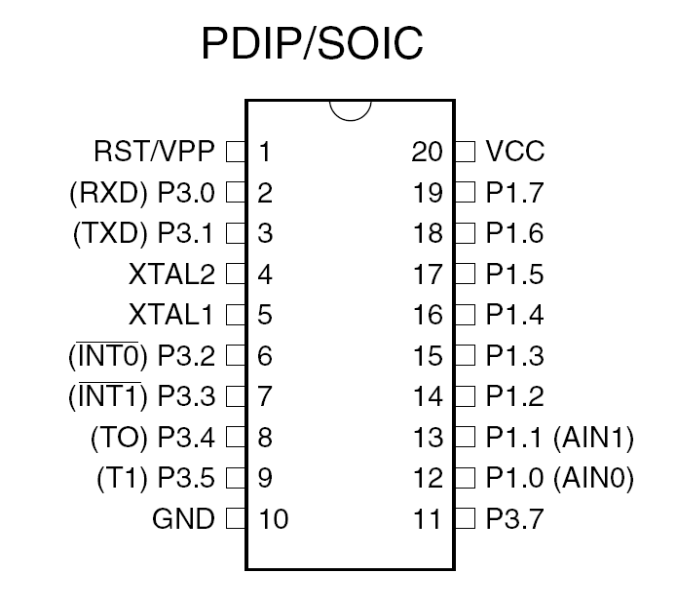
- AT89C2051 koji je jako sličan 8051, a ima samo 20 pinova

- PIC16F84 jako popularan na tržištu RISC arhitektura 18 pinova
- ATTiny2313 –AVR jako sličan prvom navedenom
- Atmega8 , moćan mikrokontroler 28 pinova, osnova za razumjevanje AVR arhitekture

Za rad u laboratoriju odabran je prvi (AT89C2051). Razlog za to je laka dostupnost na domaćem tržištu, niska cijena, jednostavnost i sličnost 8051 arhitekturi koja je jako zastupljena na realnom tržištu. Još jedan važan razlog je to što postoji besplatni kompajler – demo verzija BASCOM-8051 koji je više nego dovoljan za školske uvjete. Ovdje su dakle uzete u obzir mnoge realne okolnosti. Jedna od njih je i jednostavnost nabavke programatora i razvojnog okruženja.

Model AT89C2051

Chip pripada 8051 arhitekturi, izrađen je u CMOS tehnologiji, 20 pinova a od toga je 15 UI portova koji se mogu adresirati kao bitovi ali i kao 2 zasebna byte-a: P1 i P3. Ima 2 kB FLASH programske memorije u koju se program može upisati oko tisuću puta.



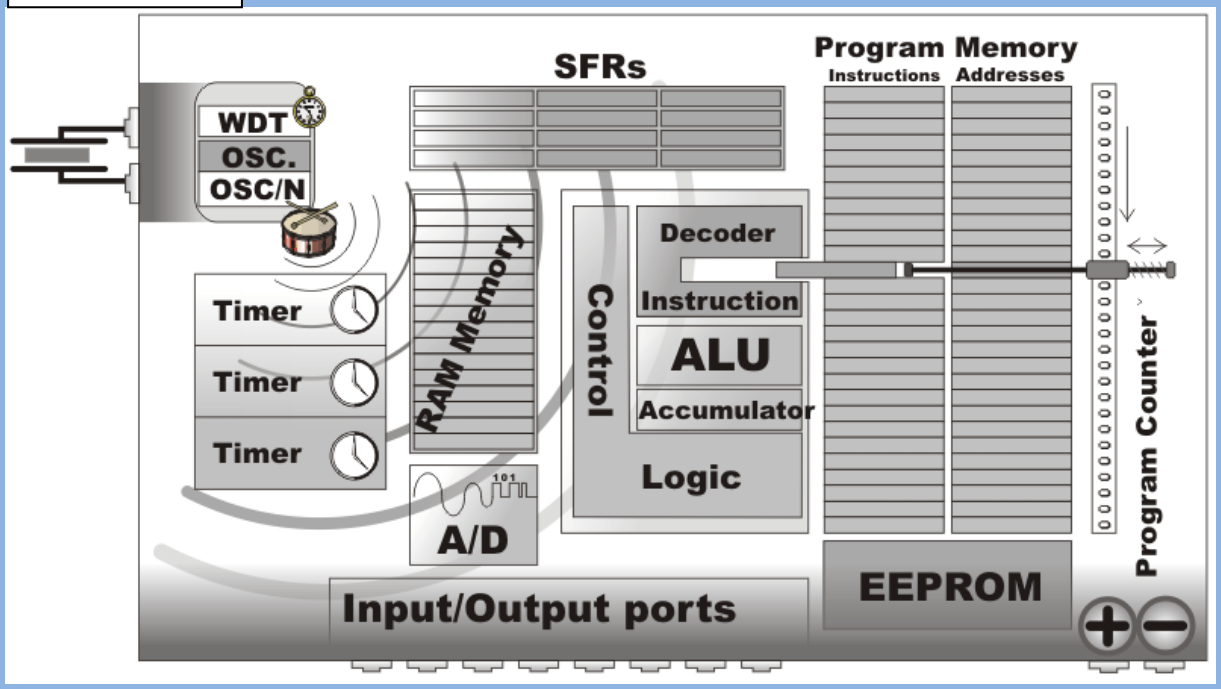
Konfiguracija:

- 128 B RAM
- 2 kB FLASH
- 15 IO portova p1 i p3
- dva 16-bitna tajmera
- Pet izvora prekida
- AD komparator
- UART
- Power DOWN i IDLE nacin rada

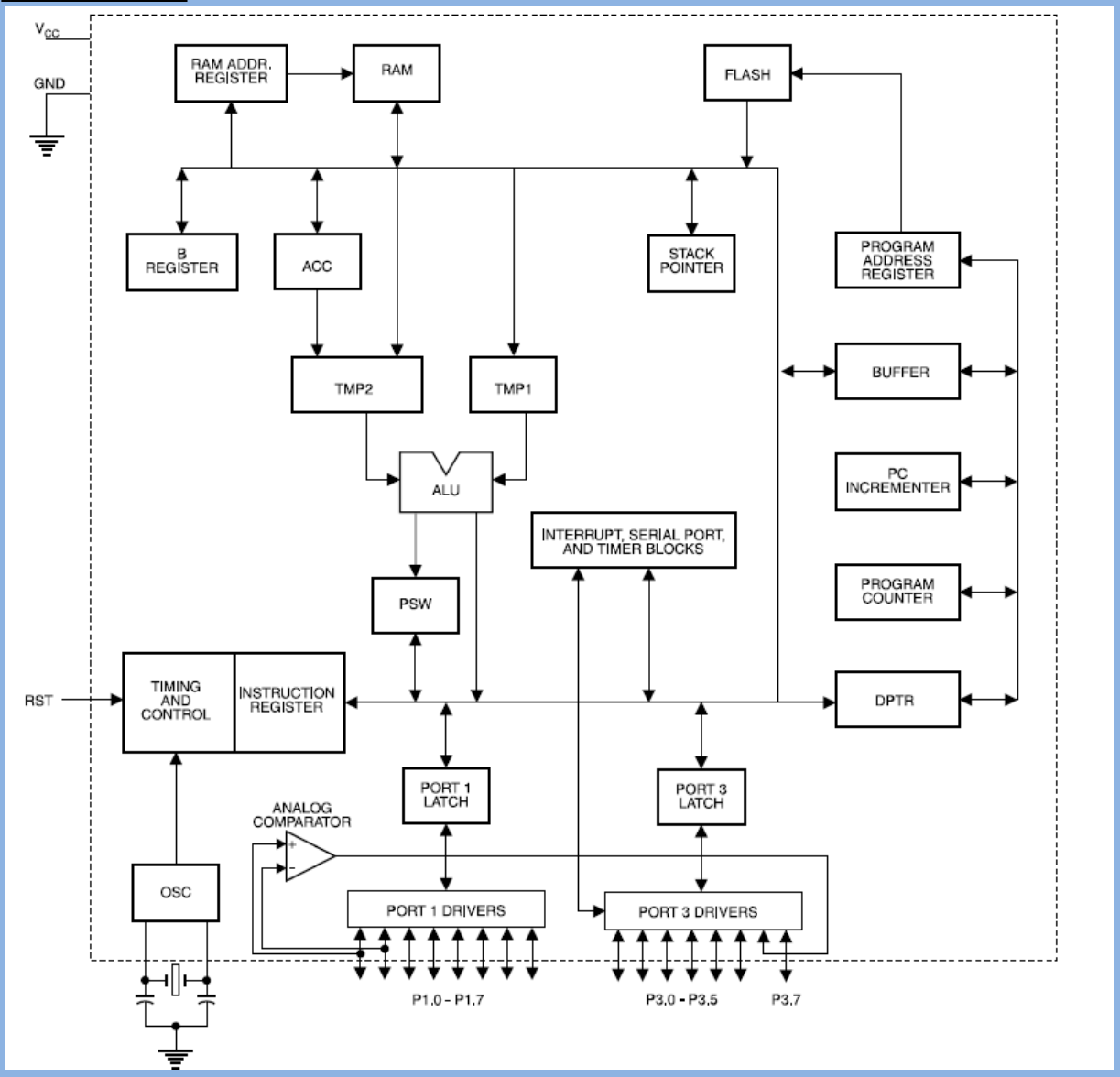
Blok shema:

8051

**8051**



**89C2051**




Objašnjenje funkcije pinova:

Da bi programer mogao programirati chip trebao bi poznavati hardware. To podrazumjeva poznavanje pinova i njihovih funkcija unutar chipa. Treba poznavati strukturu memorije i registara unutar IC-a. Na gornje dvije slike nudi nam se vizualno objašnjenje koje bi trebalo znati isčitati na slijedeći način:

- Prvi pin služi za resetiranje čipa kad radi ali i za programiranje čipa kod režima programiranja
- Drugi pin je pristup P3.0 , a druga mu je funkcija vezana za UART ,RX
- Treći pin je P3.1, a druga mu je funkcija vezana za UART, TX
- Četvrti i peti pin služe za priključivanje rezonatora(obično kristal 24MHz)

- Šesti i sedmi pin vezani su za prekidni sustav, prva im je funkcija p3.3 i p3.3
- Osmi i deveti pin vezani su za tajmere i brojače , primarna im je funkcija p3.4 i p3.5
- Deseti pin je GND –masa
- 11. Pin je p3.7
- 12. I 13. Pin vezani su za interni analogni komparator p1.0 i p1.1 ali i interno za p3.6
- Pinovi od 14 do 19 dio su P1 pristupa i nemaju sekundarnu funkciju
- Pin 20 je napajanje 5V DC

Pristupi P1 i P3 vezani su za unutarnje 8 bitne registre što je vidljvo iz blok sheme na gornjoj slici. Logičko stanje moguće im je mijenjati programski i hardverski (npr. tipkom izvana). Prioritet ima hardverska promjena stanja.

| P1.7 | P1.6 | P1.5 | P1.4 | P1.3 | P1.2 | P1.1 | P1.0 |
|---|---|---|---|---|---|---|---|

P1 pristup (port)

| P3.7 | P3.6 | P3.5 | P3.4 | P3.3 | P3.2 | P3.1 | P3.0 |
|---|---|---|---|---|---|---|---|

P3 pristup

Primjetimo da P3.6 postoji ali nije spojen prema vani na pin nego služi interno kao izlaz analognog komparatora čiji su ulazi izvana P1.0 i P1.1.

Ako obratimo pažnju na blok shemu vidjet ćemo na naš mikrokontroler ima sve komponente koje treba imati jedan takav uređaj. Ima interni RAM koji iznosi 128 B, ima flash memoriju od 2 kB ,ima 16 bitne tajmere i brojače te sustav za serijsku komunikaciju s drugim čipovima ili čak s PC računalom. Na shemi je vidljiv i prekidni sustav čiji su registri vezani djelomično s registrima za tajmere i brojače.

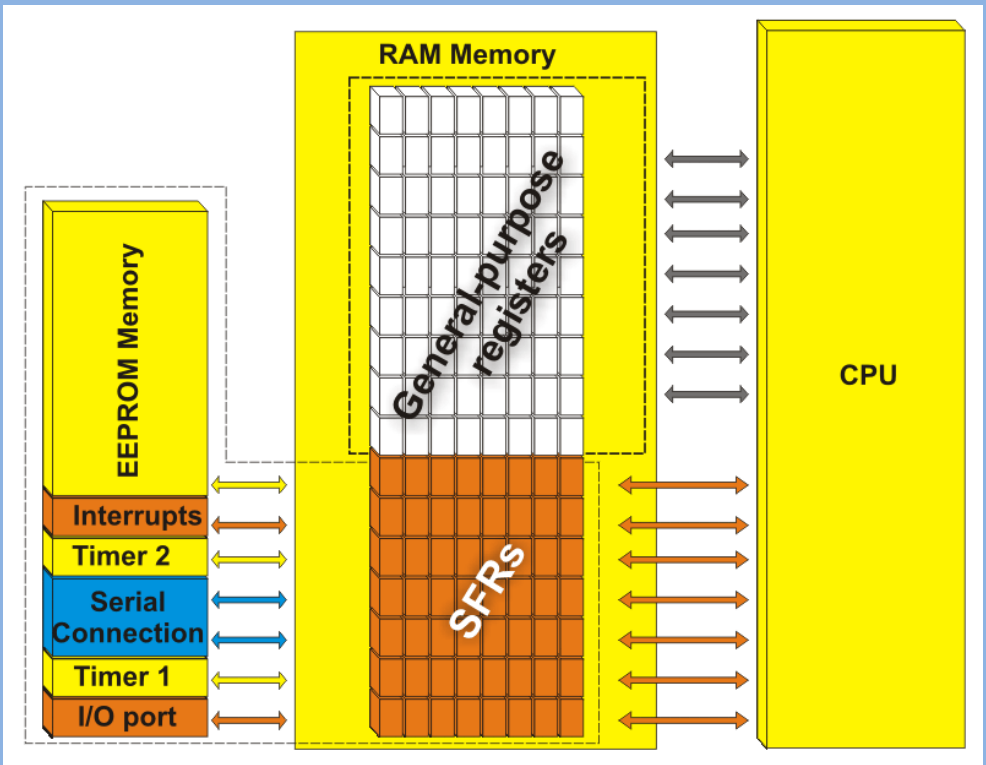
Sada prelazimo na fazu opisivanja dijelova blok sheme:

- ALU aritmetičko logička jedinica
- ACC akumulator
- B pomoćni registar
- TMP1 i TMP2 pomoćni registri
- PSW program status registar

- Programsko brojilo
- DPTR, podatkovna kazaljka
- Stack pointer ili registar kazaljke stoga
- Adresni registar

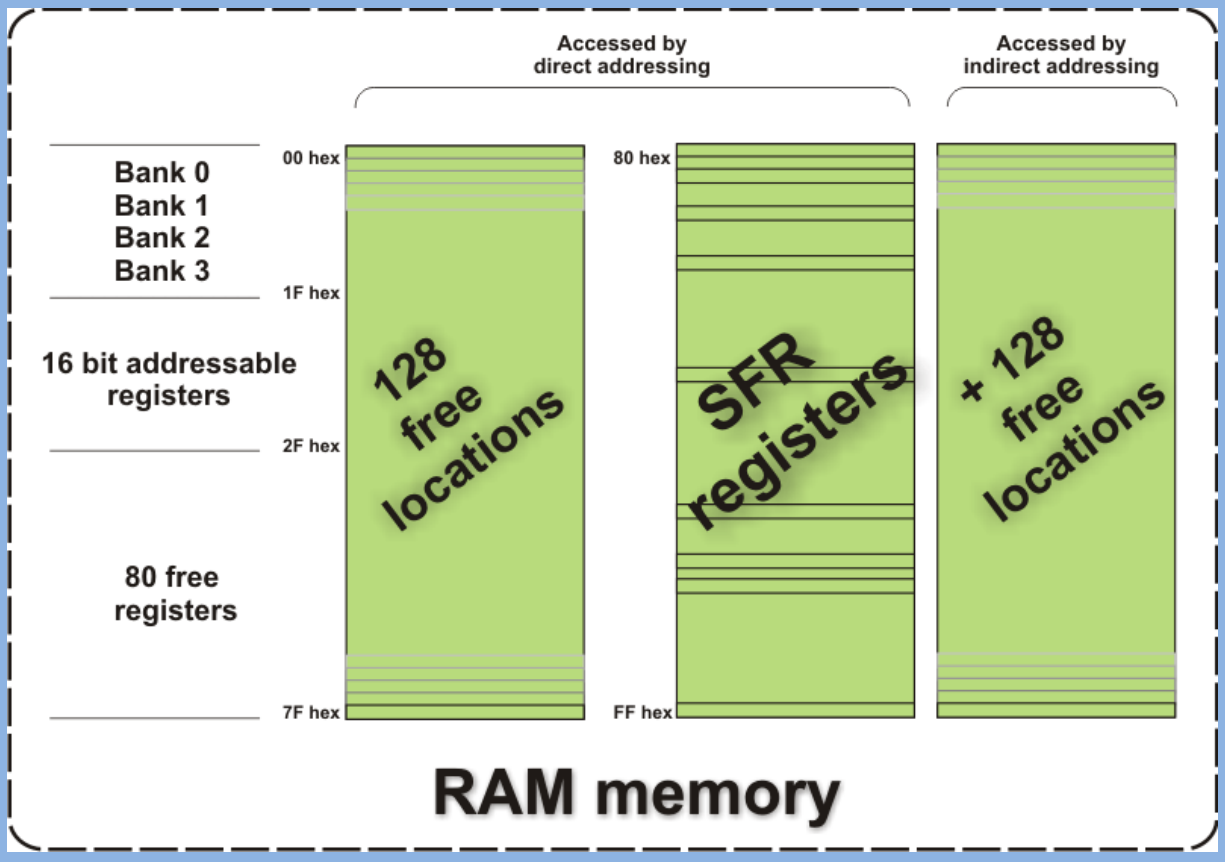
Organizacija slobodnog RAM-a i SFR (specijalnih registara):

Prva adresa je 00H, zadnja adresa je FFH. Od 00H do 1F nalaze se registri opće namjene (R0 do R7) * 4. U prostoru od 20Hdo 2FH nalazi se bitovni RAM.Od 30H do 7FH nalazi se slobodni RAM. Iznad toga su SFR registri kao što su ACC,PC,PSW,DPTR,P1,P3,SCON,TMOD,SBUF….(plavo područje). Ne treba zbunjivati činjenica da u plavom području ima nepopunjenih mjesta. To je zbog kompatibilnosti arhitekture. Što je model mikrokotrolera manji i slabiji ima više praznina jer jači uređaji imaju više registara. Dakle prazna mjesta su rezervirana za one registre koji su sadržani u složenijim mikroračunalima iste arhitekture. Ako pažljivo čitate primjetit će te da se opći registri ponavljaju četiri puta pa nije jednostavno zaključiti koji će se Rn koristiti ako ih četiri s istom oznakom. To se određuje programski u SP registru.



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| F8 | | | | | | | | FF |
| F0 | B | | | | | | | F7 |
| E8 | | | | | | | | EF |
| E0 | ACC | | | | | | | E7 |
| D8 | | | | | | | | DF |
| D0 | PSW | | | | | SPCR | | D7 |
| C8 | T2CON | T2MOD | RCAP2L | RCAP2H | TL2 | TH2 | | CF |
| C0 | | | | | | | | C7 |
| B8 | IP | SADEN | | | | | | BF |
| B0 | P3 | | | | | | IPH | B7 |
| A8 | IE | SADDR | SPSR | | | | | AF |
| A0 | P2 | | | | | | WDTRST | WDTCON | A7 |
| 98 | SCON | SBUF | | | | | | | 9F |
| 90 | P1 | | | | | | EECON | | 97 |
| 88 | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | AUXR | CLKREG | 8F |
| 80 | P0 | SP | DP0L | DP0H | DP1L | DP1H | SPDR | PCON | 87 |

Bit adresibilni registri



Ako smo pažljivo proučili registre konkretnog mikroračunala možemo pristupiti programiranju. Ovdje je bitno odlučiti da li će mo se baviti asemblerom ili će mo programirati u višem programskom jeziku. Autor ove skripte prisiljen je zbog okolnosti kombinirati i jednu i drugu varijantu što je također moguće.

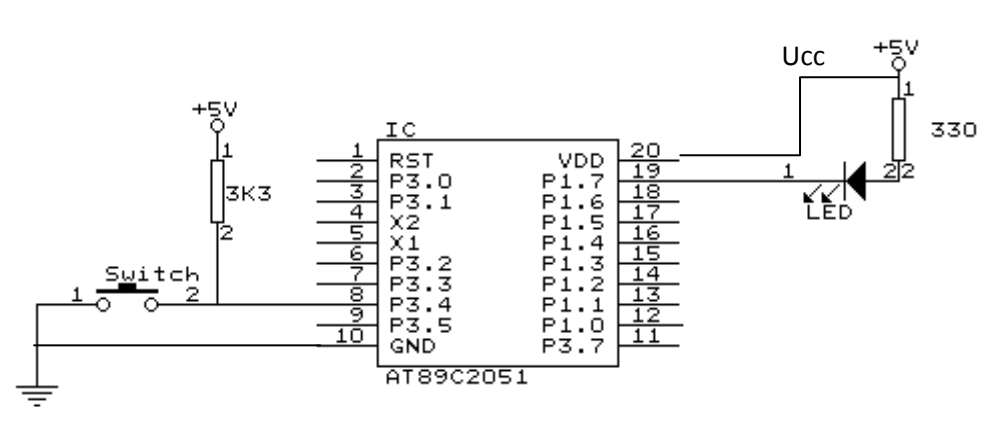
AT89C2051 moguće je programirati u asembleru , C-u ili pak BASCOM-u za 8051 verziji. Ovdje je odabran BASCOM 8051 demo. Ova verzija dozvoljava ograničen broj naredbi jer je demo, ali to je dovoljno za većinu školskih primjera. Još jedna pozitivna karakteristika je i ta što je transparentan za asembler. Drugim riječima u BASCOM 8051 razvojnom okruženju možemo naizmjenično pisati bascom-ove i asemblerske naredbe. To može biti zbunjujuće za početnike pa se na početku izbjegava. Nakon što je program napisan potrebno ga je kompajlirati. Prilikom prvog kompajliranja potrebno je stvoriti novi folder u koji trebea smjestiti novonastale datoteke kojih ima pet. Bitno je mapi dati ime jer su u njoj sve datoteke projekta. Jedna od datoteka sadrži naredbe u heksadekadskom kodu i ona je bitna kod transfera podataka u mikroračunalo.

Budući da se ovdje kao programator koristi PG302 to je potrebno postaviti u opcijama u BASCOM-u. Potrebno je i naglasiti o kojem se chipu radi.

Ostatak skripte bavit će se primjerima programiranja. Kreće se od vrlo jednostavnih primjera što podrazumjeva pokretanje LED dioda spojenih na pinove mikrokontrolera, očitavanja stanja tipki, pokretanje DC motora ili pak zvučnika pa do multipleksiranja 7-segmentnih displeja. Kad ove tehnike savladamo prelazimo na komunikaciju s vanjskim čipovima memorijama, RTC ili senzorima u obliku čipova koji u sebi imaju ugrađene sofisticirane načine komunikacije kao što su 1W,I2C,SPI, UART….

Prvi primjer

Kroz ovaj primjer naučit će mo osnovni spoj mikrokontrolera, tj onaj spoj koji je minimalan da bi uređaj radio. Dakle treba spojiti napajanje i masu, a zatim na pravo mjesto dodati rezonator – kristal 24MHz ili 12MHz. Vidjet će mo i ulogu pina za reset. Zatim će mo naučiti kako na mikrokontroler spajamo LED diode i tipkala.




Na pinove X1 i X2 potrebno je još sopjiti rezonator – kristal.

Kako bi trebao izgledati program koji će raditi slijedeće:

1. Kad je tipka na p3.4 pritisnuta neka se dioda upali , a kad je tipka otpustena neka se dioda ugasi.
   **If p3.4=0 then**
   **Reset p1.7**
   **Else**
   **Set p1.7**
   **End if**

2. Kad je tipka pritisnuta dioda se pali , a kad tipku otpustimo dioda ostaje upaljena. Pin za resetiranje gasi tipku.
   **do**
   **If p3.4=0 then**
   **P1.7=0**
   **End**
   **End if**
   **loop**

3. Dioda se pali na prvi pritisak tipke a gasi na drugi pritisak tipke i tako redom.

```
Kreni:
If P3.4 = 1 Then
 Goto Kreni  'skeniraj tipku na p3.1, ako nije pritisnuta vraćaj se na kreni i skeniraj
Else             'ako je tipka pritisnuta
P1.7 = 0         'upali diodu
End If
Sacekaj:
If P3.4 = 0 Then   ' sačekaj dok tipka ne bude otuštena
Goto Sacekaj
End If
Stani:
If P3.4 = 1 Then      'sada ponovo skeniraj tipku
Goto Stani
Else
P1.7 = 1
Sacekaj1:
If P3.4 = 0 Then   ' sačekaj dok tipka ne bude otuštena
Goto Sacekaj1
End If
Goto Kreni         ' ponovi
End If
```

4. Kad pritisnemo tipku pet puta dioda se pali. Pin za resetiranje gasi tipku.

```
Dim Brojac As Byte 'deklariraj varijablu kao bajt
Pocetak:
If P3.4 = 1 Then       'skeniraj tipku ili pin
Goto Pocetak
Else
Incr Brojac              ' brojac=brojac+1
If Brojac = 5 Then      'ako je brojac dostigao stanje 5 upali diodu
'Print Brojac
P1.7 = 0
End If
End If
Pricekaj:
```

**If P3.4 = 0 Then 'sačekaj dok tipka ne bude otpuštena**
**Goto Pricekaj**
**Else**
**Goto Pocetak**
**End If**

5. Kad pritisnemo tipku dioda se naizmjenično pali i gasi. Dioda prestaje titrati na reset.

**Pocetak:**
**If P3.4 = 1 Then**
**Goto Pocetak**
**Else**
**Do**
**P1.7 = Not P1.7**
**Waitms 250 'čekaj 250 ms**
**Loop**
**End If**

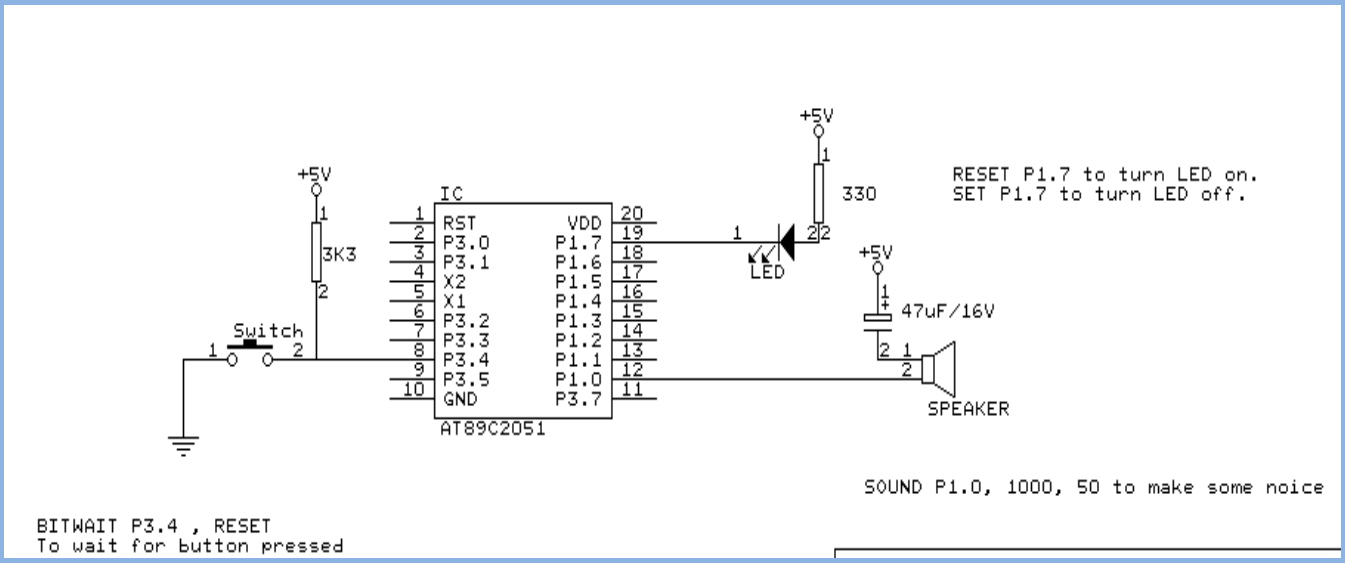
Iz navedenih primjera vidimo da je sintaksa BASOM-a slična sintaksi BASIC-a. Ovdje su korištene naredbe grananja i odlučivanja IF, THEN, ELSE, ELSE IF, END IF zatim beskonačna petlja DO LOOP, korištenja je naredba skoka GOTO , naredba čekanja WAITMS i naredba postavljanja stanja SET , RESET. Dobro je primjetiti da je tipka spojena i djeluje prema negativnoj logici, tj pritisnuta tipka na ulazu generira logičko stanje 0 a otpuštena tipka stanje 1. Slično je i na izlazu, naime LED svijetli onda kad je na izlazu stanje nula jer tada struja teče iz izvora kroz diodu u mikrokontroler. To je uobičajeni način kod ove vrste chipova iako smo prije navikli na pozitivnu logiku TTL IC-a.

Zaključimo ovaj dio primjera izjavom koja je bitna:

**Programer mora razumjevati hardver i registre mikrokontrolera i dobro poznavati djelovanje naredbi. Ako već prije nije naučio binarne brojeve i logičku algebru imat će poteškoća u radu.**

## Drugi primjer

U drugom primjeru zadržat će mo konfiguraciju iz prvog zadatka ali na p1.0 spojit će mo zvučink.




Ako smo dobro proučili prvi primjer drugi će mo razumjeti lako. Ovdje je potrebno 'zamoliti' zvučnik da odsvira neku melodiju. To možemo učiniti naredbom SOUND. Ta naredba proizvodi niz pravokutnih impulsa zadanog trajanja i zadane periode. Ako na pin P1.0 pošaljemo naredbu SOUND p1.0, 200,150 zvučnik će generirati zvuk frekvencije koja je ovisna o trećem parametru (150) a trajat će 200 impulsa. Možemo zaključiti da će trajanje zvuka za isti broj impulsa biti ovisno i o frekvenciji, drugim riječima ako je frekvencija manja zvuk će trajati duže iako je broj impulsa isti. Te okolnosti treba uzeti u obzir ovisno o vrsti zadatka koji rješavamo.

1. Neka zvučnik proizvodi zvuk na prvi pritisak tipke , a gasi se na drugi pritisak iste tipke. Paralelno sa zvučnikom neka se pali i gasi LED.

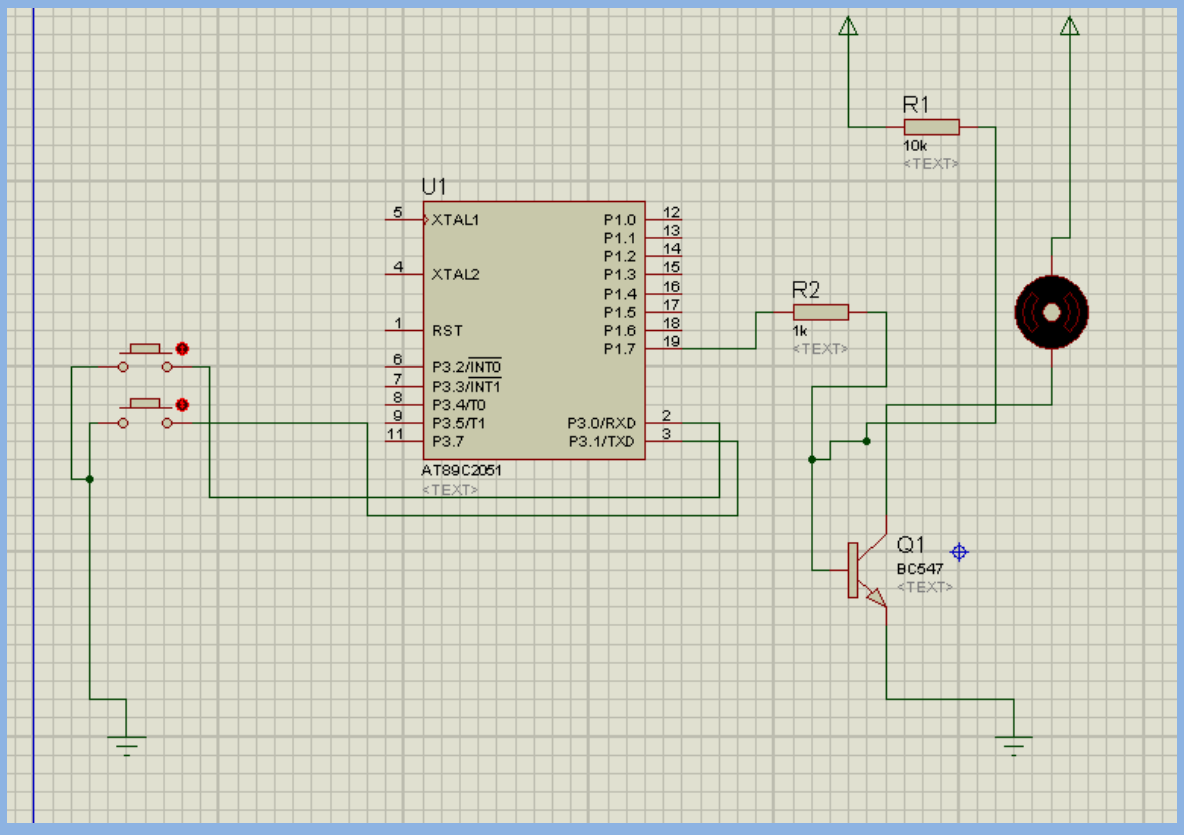
```
Kreni:
If P3.4 = 1 Then
 Goto Kreni
Else
P1.7 = 0                    'upali diodu
Sound p1.0,300,250   'generiraj zvuk
End If
Sacekaj:                     'čekaj dok se tipka ne otpusti
If P3.4 = 0 Then
Goto Sacekaj
End If
Stani:
If P3.4 = 1 Then
Goto Stani
Else
P1.7 = 1          'ugasi diodu
Set p1.0          'zaustavi zvuk
Sacekaj1:
If P3.4 = 0 Then
Goto Sacekaj1
End If
Goto Kreni
End If
```

2. Pogledajmo sad kako bi riješili 3. Zadatak iz prvog primjera ali korištenjem asemblera. Asembler možemo pisati i ugrađivati u BASCOM ali to treba naglasiti pseudonaredbom $ASM i kraj asemblerskog dijela označiti pseudonaredbom $END ASM. Možemo dakle naizmjenično koristiti asembler i BASCOM u istom razvojnom okruženju. Početnicima to djeluje samo zbunjujuće ali kad riješavamo ozbiljne zadatke ta značajka je jako bitna. Kažemo da je BASCOM transparentan za asembler.

```
$asm
setb p1.7
Tipka:
Jb P3.4, *+0  'skeniraj dok  bit ne postane low (tipka pritisnuta)
clr p1.7      'upali LED
Jnb p3.4 , *+0  'čekaj dok tipka ne bude otpuštena
Nula:
Jb P3.4, nula  'skeniraj dok  bit ne postane low (tipka pritisnuta)
setb p1.7     'ugasi LED
Jnb p3.4 , *+0  'čekaj dok tipka ne bude otpuštena
sjmp tipka     'skoči na oznaku tipka i ponovi sve
$end Asm
```

## Treći primjer

**Ovd**je je zadatak slijedeći: spojiti DC motor male snage na mikrokontroler. Isplanirati upravljački dio s LED diodama i tipkama.Treba odabrati i odgovoarajući tranzistor za prilagodbu između motora i mikrokontrolera.
Shema bi mogla biti ovakva:



1.Potrebo je napraviti program koji će omogućiti da prvom tipkom pokrenemo motor a drugom tipkom zaustavimo motor.

Pripadajući programski kod:

```
P1.7 = 0
Kreni:
If P3.0 = 1 Then
Goto Kreni
Else
Set P1.7
End If
Cekaj:
If P3.0 = 0 Then
Goto Cekaj
End If
Zaustavi:
If P3.1 = 1 Then
Goto Zaustavi
Else
Reset P1.7
End If
Cekaj1:
If P3.1 = 0 Then
Goto Cekaj1
End If
Goto Kreni
```

Zadatak za vježbu: Priključiti LED diodu na pin p1.6. Prilagoditi program tako da LED svijetli kad motor radi. Kad taj dio proradi dodajte zvučnik na p1.5.Prilagoditi program tako da zvučnik dade znak prije startanja motora.

Četvrti primjer
SEMAFOR
Ako vam se učinilo da su primjeri postali komplicirani možete sad pogledati kako će mo riješiti jedan jednostavni sekvencijalni zadatak.

1. Prisjetimo se što se događa sa svijetlima semafora kad smo na raskrižju. Kojim se redoslijedom pale svijetla? Razmislite, proučite programski kod te odgovorite: Na koje pinove su spojene crvena,žuta i zelena LED za automobile, a na koje pinove LED-ice za pješake?

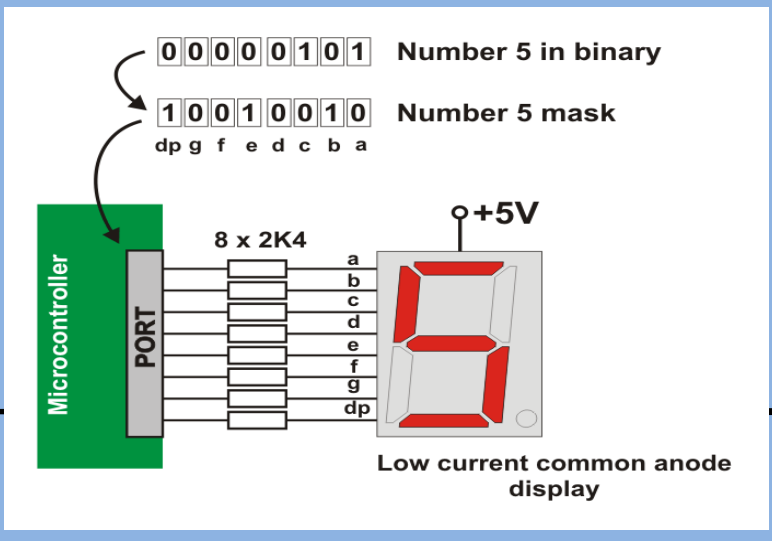
```
    Dim I As Long
P1 = 255
Do
P1.7 = 0
P1.0 = 0
P1.1 = 1
P1.2 = 1                          'crvena
Wait 4
P1.6 = 0
P1.0 = 1
P1.1 = 0
                          'crvena i žuta
Wait 4
P1.7 = 1                          'zelena
P1.6 = 1
P1.5 = 0
 P1.1 = 1
 P1.2 = 0
Wait 4
P1.6 = 0                          'žuta
P1.5 = 1
p1.2=1
P1.1 = 0
Wait 4
Sound P1.4 , 1000 , 255
Wait 1
Sound P1.4 , 5000 , 100
Wait 1
```

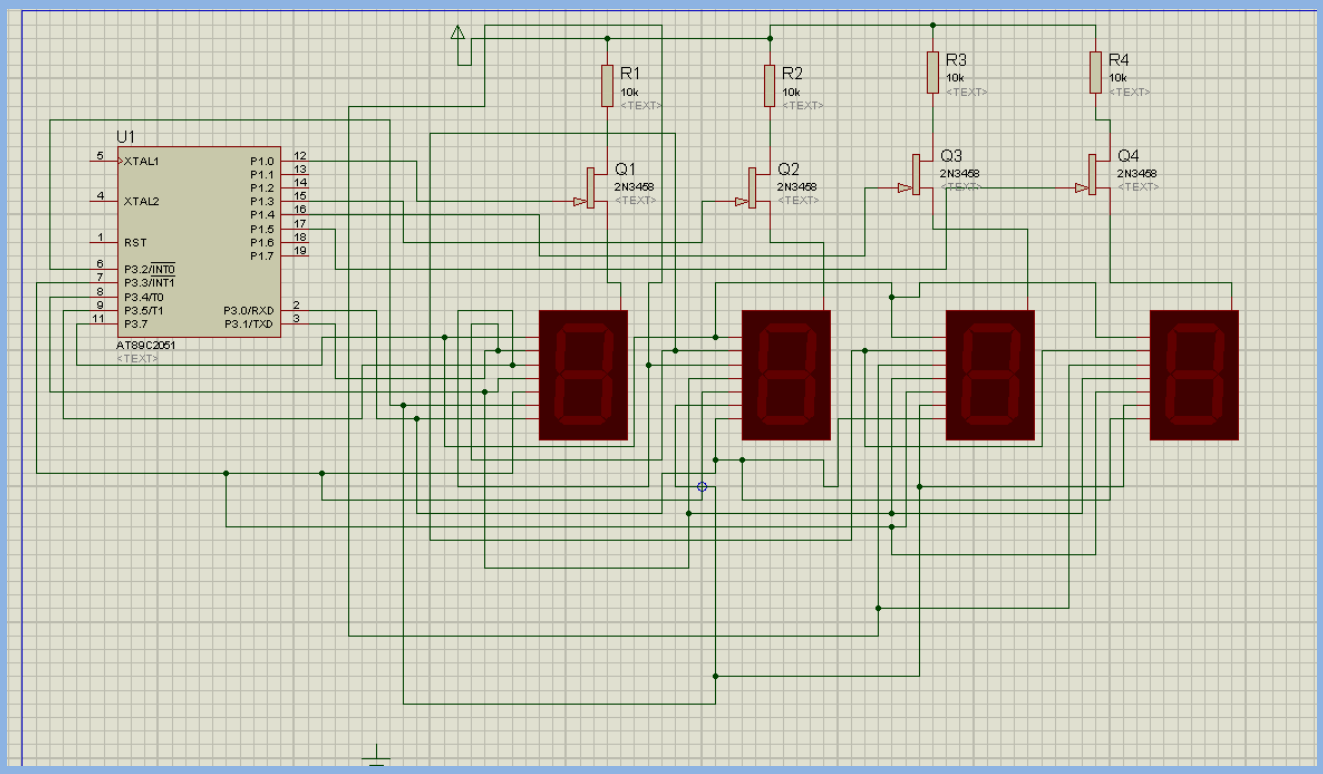
**Loop**

Što je vaš zadatak ovdje? Jednostavno proučiti program te nacrtati shemu spoja mikrokontrolera i LED dioda. Na koji je pin spojen zvučnik?

**Peti primjer**
**Upravljanje 7-segmentnim displejima –multipleksiranje displeja.**

Česta uloga mikroračunala je i upravljanje displejima. To se može napraviti i pomoću specijaliziranih tvornički programiranih čipova – dekodera. Lakši je međutim način pomoću mikroračunala zato jer program koji napišete naprosto zamjenjuje vanjski čip –dekoder. Druga je prednost mikroračunala vrlo moćna – omogućuje multipleksiranje displeja. Što je sad to? Poznato je da 7-segm displej ima deset pinova. Konkretan mikrokontroler ima 15 IO pinova. Nije baš preporučljivo potrošiti 70% pinova samo za jedan displej a on nam može poslužiti samo za jednu znamenku. Zato se primjenjuje multipleksiranje. Naime naizmjenično se uključuju : prvi displej, drugi displej, treći displej, četvrti displej….itd(ali brzo – u milisekundama tako da imamo privid da su svi displeji upaljeni). Da bi smo situaciju shvatili do kraja prisjetimo se kako rade 7-segm. displeji. Displeji o kojima govorimo obično se prave u spju sa zajedničkom katodom i u spoju sa zajedničkom anodom. Osim zajedničke elektrode displej ima barem sedam pinova za pojedine segmente, zatim jedan pin za točku. To je ukupno devet pinova. Dva pina su kratko spojena i vode na zajedničku elektrodu. Ako se radi o displeju sa zajedničkom anodom onda je situacija ovakva: anoda se spaja na izvor VCC ili 5V a katode ili segmenti spajaju se na pinove mikrokotrolera , pale se logičkom nulom, gase logičkom jedinicom. Kod displeja sa zajedničkom katodom , katodu spajamo na GND segmenti se pale logičkom jedinicom koje im šaljemo sa pinova kontrolera.






Ovdje je prikazana električna shema multipleksiranja 4 displeja sa zajedničkom anodom. Pristup p3 korišten je za slanje podataka na displeje a četiri pina s p1 pristupa naizmjenično uključuju displeje. Ovako se za četiri znamenke troši deset pinova. Ostalih pet pinova može se koristiti za povezivanje s ulaznim uređajima, senzorima ili tipkama….

Princip rada je jednostavno shvatiti: tranzistori Q1 do Q4 rade kao sklopke i naizmjenično otvaraju displeje jedan za drugim u kratkim vremenskim razmacima. Radom tranzistora upravljaju ovdje pinovi p1.0, p1.3,p1.4 i p1.5 a za dobivanje podatka na displej odgovorni su pinovi p3 pristupa. Sveukupno ovdje je angažirano 4+7 pinova(11).

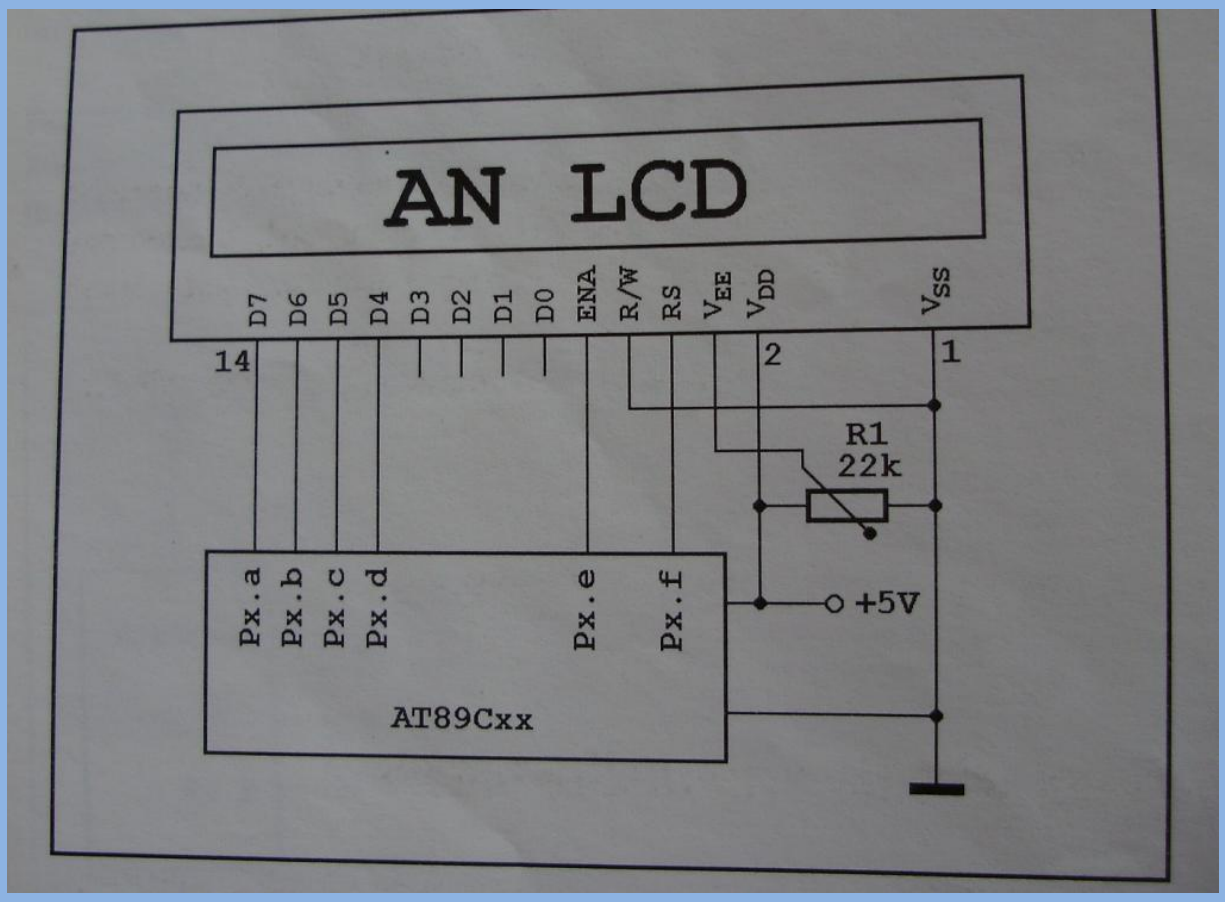
Pripadajući programski kod za prikaz zadanih znakova na displejima:

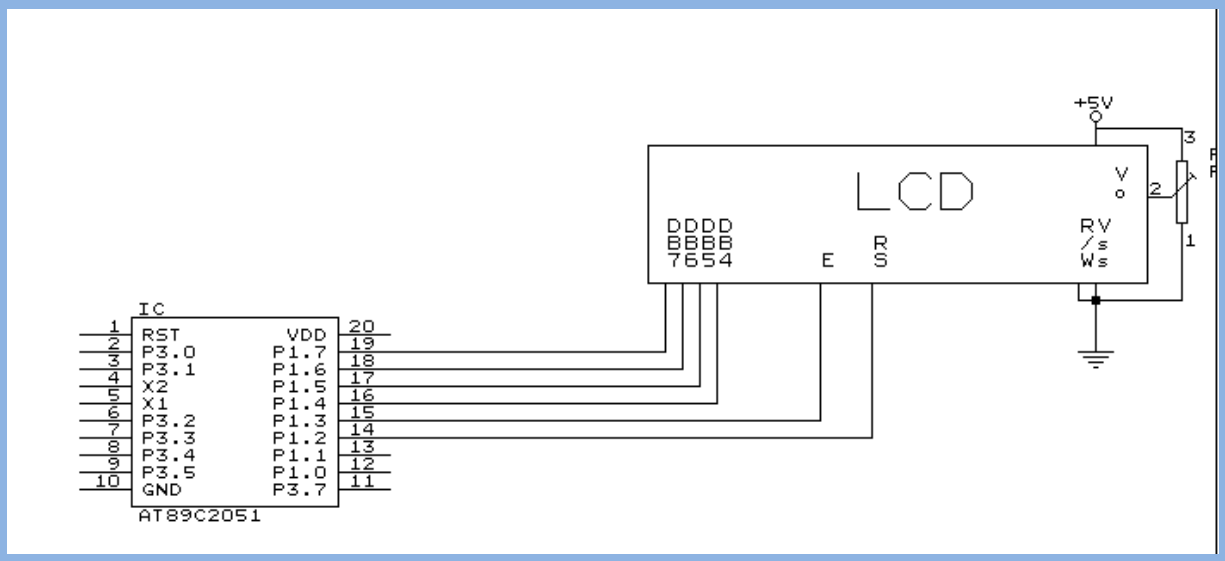
```
Dim P As Const &B01110000                    ' P
Dim U As Const &B11000001                    ' U
Dim L As Const &B11100011                    ' L
Dim A As Const &B01010000                    ' A
Dim Cetiri As Const &B11011000
Dim Bb As Const &B11000010
Dim Gasi As Const &B11111111
Dim Dva As Const &B01100100
Dim Jedan As Const &B11011101
```

```
Dim Nula As Const &B01000001
P1 = 0
P3 = 255
 Dim I As Word
 Pocetak:

Do

  P1 = &B00100000                    ' 1. displej
  P3 = A
  Waitms 5
  P1 = &B00010000                    ' 2. displej
  P3 = L
  Waitms 5
  P1 = &B00001000                    ' 3. displej
  P3 = U
  Waitms 5
  P1 = &B00000001                    ' 4. displej
  P3 = P
  Waitms 5
   Incr I
   If I > 100 Then
    Exit Do
   End If
Loop
   P3 = 255
  I = 0
  Waitms 255
 Do
 P1 = &B00100000                     ' 1. displej
  P3 = Jedan
  Waitms 1
  P1 = &B00010000                    ' 2. displej
  P3 = Jedan
  Waitms 1
  P1 = &B00001000                    ' 3. displej
  P3 = Nula
  Waitms 1
  P1 = &B00000001                    ' 4. displej
  P3 = Dva
```

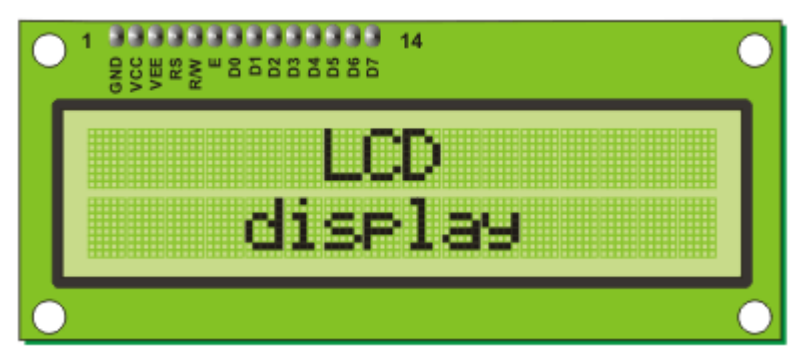
```
  Waitms 1


   Incr I
   If I > 500 Then
    Exit Do
   End If
Loop
   P3 = 255
  I = 0
  Waitms 255

     Do

 P1 = &B00100000                    ' 1. displej
  P3 = Gasi
  Waitms 1
  P1 = &B00010000                   ' 2. displej
  P3 = Gasi
  Waitms 1
  P1 = &B00001000                   ' 3. displej
  P3 = Bb
  Waitms 1
  P1 = &B00000001                   ' 4. displej
  P3 = Cetiri
  Waitms 1

   Incr I
   If I > 600 Then
    Exit Do
   End If
Loop
   P3 = 255
  I = 0
  Waitms 255
 Goto Pocetak
```

Šesti primjer:
Upravljanje alfanumeričkim displejima.



Ova vrsta displeja složenija je po izvedbi i ima puno veće mogućnosti: može prikazati sve znakove sa tipkovnice računala, dakle mala i velika slova, znakove interpunkcije i brojeve. Prema tome zaključujemo da je i princip rada složeniji. Glavni dio displeja je matrični displej s tekućim kristalom. Svaki znak moguće je formirati pomoći 5*8=40 segmenata. Najčešće koristimo takve displeje koji imaju 2 reda po 16 znakova. Displejem se upravlja iz mikrokontrolera pomoću RS i E pinova, podaci se prenose preko DB pinova RW koristi se za read write načine rada. Displej se napaja s 5V a može imati i napon za podešavanje kontrasta , a neki displeji zahtijevaju i osvjetljenje ekrana pomoću dviju LED dioda koje također trebaju napajanje od 5V DC.



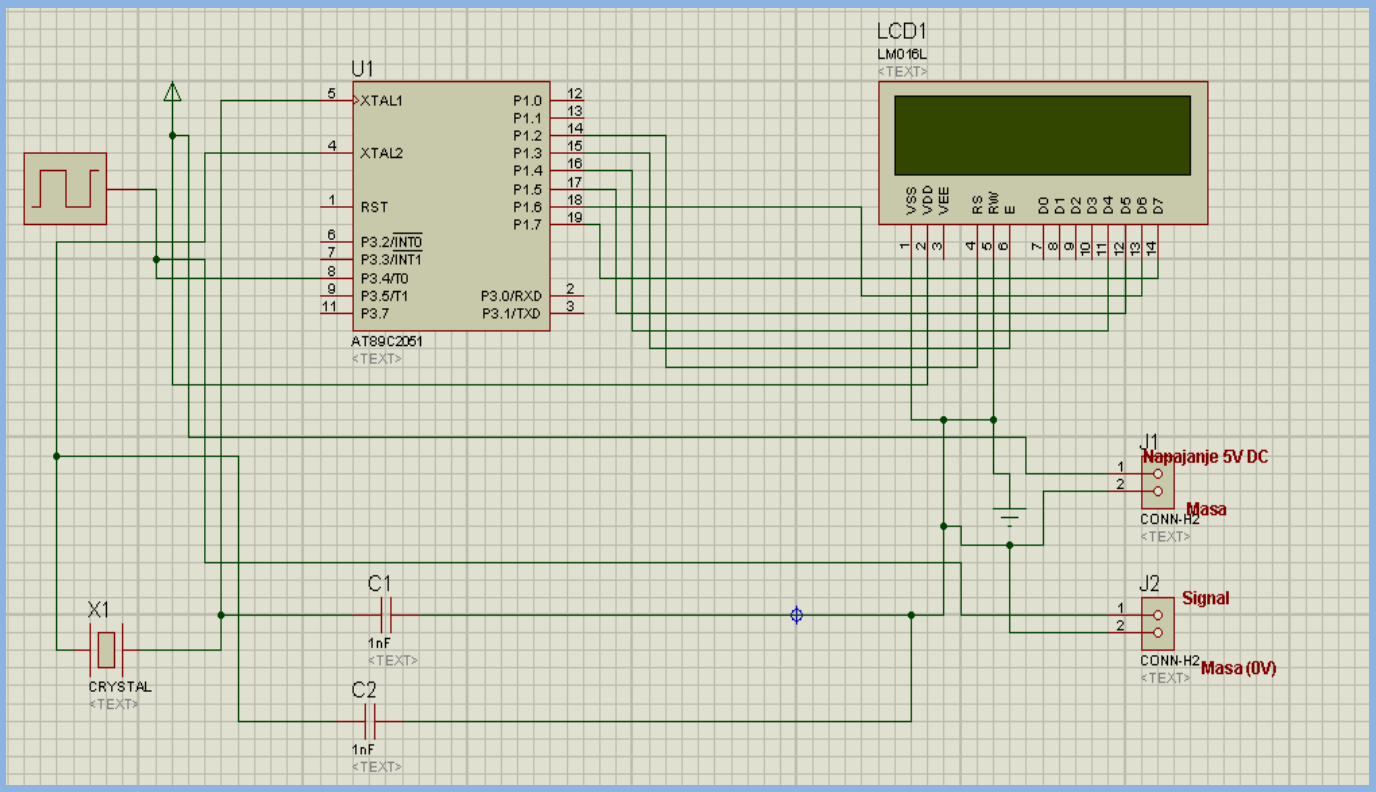

Ovdje je dan detaljniji opis funkcija pinova displeja.

Primjer mjerenja frekvencije i prikaz na LCD-u:

**Osnovni kod**

```
Dim Frekv As Word
Dim Brojac As Word

Config Timer0 = Counter , Gate = Internal , Mode = 1
Counter0 = 0
Start Counter0
Waitms 1
Stop Counter0
Brojac = Counter0

Cls
Lcd Brojac
```

Shema:




Ovaj primjer obuhvaća dvije teme:

- Upravljanje displejem
- Mjerenje frekvenicje

Upravljanje displejem podrazumjeva poznavanje naredbi namjenjenih alfanumeričkom displeju: CLS, LCD, HOME,LOWERLINE, UPPERLINE,SHIFTLCD…

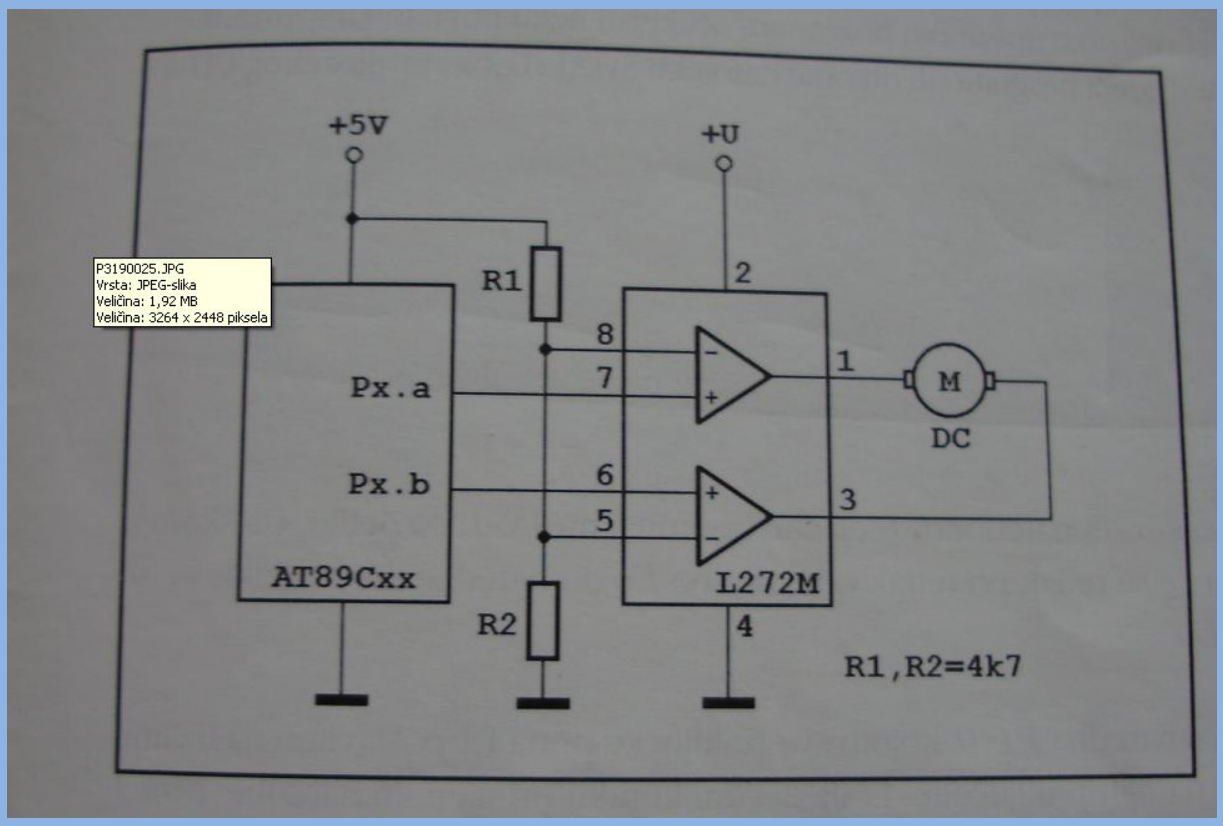
Mjerenje frekvencije je opisano kao sekundarna tema: Generator impulsa šalje niz pravokutnih impulsa na P3.4 pin mikrokontrolera. Na taj pin inače je priključen unutrašnji COUNTER koji je već programski pokrenut. Counter broji koliko je impulsa došlo u jednoj sekundi. Taj podatak prema fizikalnoj definiciji predstavlja frekvenciju. Podatak o izmjernoj frekvenciji se šalje na LCD pomoću naredbe LCD. Naredba CLS u programu služi za brisanje podataka s displeja i pozicioniranje pokazivača na početno mjesto: prvi red, prvi supac.

Vidimo da je ovdje pola D pinova na strani displeja neiskorišteno. To je zato što je sabirnica između displeja i mikrokontrolera konfigurirana kao 4-bitna.(CONFIG LCDBUS=4).

Sedmi primjer

Upravljanje DC motorom

- Pokretanje
- Promjena smjera vrtnje
- Promjena brzine vrtnje



Ponavljanje gradiva I plugodište:

Recimo nesto općenito o mikroračunalima: Od čega su napravljena? Gdje se koriste? Kako se dijele? Kako se programiraju? Koje periferne komponente idu uz nijh?
Mikroprocesor je sačinjen od ALU, RAM-a i registara opće i specijalne namjene, te od dekodera instrukcija. Ako procesoru dodamo programsku i podatkovnu memoriju u jedan jedinstveni IC dobijemo mikroračunalo. Mikrouupravljači nove generacije u sebi imaju i tzv dodatne sklopove koji ih čine efikasnim u riješavanju zadanih zadataka. Ti dodatni sklopovi su redovno tajmeri, AD pretvornici, INTERUPT-i,analogni komparatori,i neizbježni UART ili USART.Sve nabrojeno mora se nekako povezati. To se radi pomoću unutrašnjih sabirnica.
Današnja mikroračunala ili popularnije mikrokontroleri nalaze se u mnogim komercijalnim uređajima široke potrošnje ,ali i u zahtjevnim aplikacijama u industriji, ugrađeni su u u mnoge telekomunikacijske uređaje ali i u automobile, avione , dio su svakog PLC-a , PC računala. Dijele se prema unutrašnjoj arhitekturi na RISC i na CISC. Dijele se i na porodice (ENGL familyes)
Programski jezik mikroupravljača je ASSEMBLER. Svaka porodica ima svoj specifični ASSEMBLER koji je ovisan o arhitekturi mikroprocesora. ASSEMBLER je vezan za hardver danog mikroračunala.
Brzi razvoj zahtjeva i brže pisanje i razvijanje programa pa postoje i viši programski jezici za mikroračunala bazirani na C jeziku i na BASIC-u. Tako imamo slijedeće programske jezike i razvojna okruženja: BASCOM 8051, BASCOM AVR, PICBASIC, MIKROBASIC, AVR STUDIO,mikroKEIL.....
Periferne komponente su nužne jer bez njih ne možemo vidjeti rad mikroračunala niti ga upotrijebiti. Tu spadaju:
TTL i CMOS IC krugovi i senzori bazirani na I2C,1W,SPI,UART,CAN,Ethernet,RS232 standardima . AD i DA konvertori, EEPROM memorije, IC Real time kalendari i satovi,RAM memorije, FET i bipolarni tranzistori, driveri DC step, i servo motora, IC prijemnici i detektori satelitskih frekvencija , IC prijemnici radio frekvencija.
Od jednostavnijih perifernih komponenti imamo tipke,sklopke, adresibilne sklopke, identifikacijske tipke,matrične tipkovnice, 7 segmentne displeje, alfanumeričke displeje, digitalne potenciometre i sl...

Nastavne teme:

1. Podjela mikroračunala
2. Arhitektura mikroprocesora
3. Memorije i operacije s binarnim brojevima
4. Unutrašnje sabirnice i dohvaćanje instrukcija
5. Assembler
6. Periferni dijelovi mikroračunala
7. Programski jezik Bascom
8. Timeri i brojači
9. Prekidni sustavi
10. UART i druge komunikacije
11. Komparatori i AD pretvorba, multipleksori
12. Usporedba 89c2051, 8051, ATtiny2313, Atmega8, PIC16F84,PIC16F628

Pitanja , priprema za teoretski dio ispita:

## **1.** Što je **mikroprocesor**?

- mikroprocesor je složeni programski upravljivi elektronički sklop koji pribavlja, dekodira i izvršava

instrukcije

## **2.** Objasni razliku **mikroprocesora** i **mikroračunala**.

- mikroprocesor je dio mikroračunala

- mikroprocesor je čip koji pribavlja, dekodira i izvršava instrukcije a mikroračunalo se sastoji od

mikroprocesora, memorije, sabirnice, ulaza i izlaza

## **3.** Nabroji osnovne dijelove **sabirnice**.

- sabirnica se sastoji od 3 dijela - adresni dio (prenose se adrese), podatkovni dio (prenose se podaci) i

instrukcijski dio (prenose se instrukcije)

## **4.** Objasni prirodni binarni kod - **NBC kod**.

- kod NBC koda kodira se cijeli broj (za razliku od BCD koda gdje se kodira svaka znamenka za sebe), iz tog

razloga BCD kod ima prednost jer unaprijed zna koliko bitova treba rezervirati za određen broj, dok kod

NBC koda taj broj varira

- kodira se jednostavnim dijeljenjem sa 2

- u jedan bajt se mogu zapisati brojevi s najviše 8 bitova tj. brojevi od 0-255 ( $2_8$ )

- veći brojevi se zapisuju u više uzastopnih bajtova

## **5.** Objasni kodiranje dekadskih znamenaka - **BCD kod**.

$(_{10})$ 753 0111 0101 0011$_{BCD}$ =

0011

0101

0111

- **BCD** je težinski kod *(engl. weighted code)* jer bitovi u kombinacijama imaju težine mjesta 8, 4, 2 i 1

- zbroj težina brojevnih mjesta na kojima je binarna znamenka koda 1 vrijednost daju vrijednost

kodirane dekadne znamenke (svaka znamenka se kodira zasebno)

- nedostatak toga je što sadrži i kombinaciju 0000 pa prekid pri prijenosu podataka može biti

prihvaćen kao podatak 0

- BCD kod je neekonomičan zato što koristi samo 10 znamenaka od mogućih 16 ( $2_4$ ), stoga se trećina koda

odbacuje (ne koristi se $2_3$ kombinacija tj. 3 bita zato što daje ukupno 9 znamenaka od potrebnih 10)

## **6.** Objasni kod za zapisivanje znamenki - **ASCII kod**.

- ASCII je skraćenica od American Standard Code for Information Interchange

- služi za zapisivanje raznih znakova - slova, brojeva, raznih instrukcija, posebnih znakova i sl.

- za kodiranje se koristi 7 bitova (8. bit je kontrolni - i on je 0)

- pomoću ASCII koda može se kodirati $2_7 = 128$ različitih znakova

- u svijetu postoje razne „podvrste“ ASCII koda, ovisno o potrebama samog jezika neke države (npr. u

hrvatskoj to su slova šđžćč)

- tablica ASCII koda sadrži stupce koji su težinski viši bitovi (3 bita) te od redaka koji su težinski niži (4 bita)

## **7.** Opiši i objasni funkciju **programskog brojila**.

- programsko brojilo je registar u mikroprocesoru u kojemu se nalazi adresa slijedeće instrukcije koja će se

obaviti

- upravljačka jedinica automatski povećava programsko brojilo za 1 (inkrementira) u toku jednog

instrukcijskog ciklusa zato što je program niz/slijed instrukcija i iz tog razloga se instrukcije moraju izvoditi

jedna iza druge (ukoliko programer nije naveo drugčije), a ne svaka druga, treća i sl.

## **8.** Opiši i objasni funkciju **brojila podataka**.

- podatkovno brojilo je registar koji sadrži adresu jednog od operanada koji će se dovesti na ulaz

arimetičko-logičke jedinice

- kada upravljačka jedinica dekodira instrukciju koja traži da se dovede operand iz memorije, tada adresa iz

podatkovnog brojila „iscuri“ na sabirnicu, dolazi do adresnog međuregistra u memoriji gdje se pronalazi

željeni operand na navedenoj adresnoj lokaciji, prebacuje se u podatkovni međuregistar u memoriji te

preko sabirnice dolazi u mikroprocesor u akumulator, podatkovni međuregistar ili registre opće namjene

## **9.** Opiši i objasni funkciju **akumulatora**.

- akumulator je registar u kojem se nalazi jedan od operanada ili se u njega može pohraniti rezultat obrade

- prilikom pronalaženja operanda u memoriji on se može spremiti u akumulator radi bržeg pristupa,

također nakon izvršene instrukcije aritmetičko-logička jedinica može proslijediti (ako je tako programer

napisao) rezultat (najčešće onaj parcijalni) u akumulator

- dobio je naziv zato što akumulira podatke (operand ili rezultat)

## **10.** Nabroji **faze** izvođenja **instrukcije**.

- prva faza je **dobavljanje** instrukcije (instrukcija se pomoću adrese iz programskog brojila dobavlja iz

memorije), zatim **dekodiranje** instrukcije (dekoder dekodira instrukciju tako da ALU zna koju instrukciju

treba obaviti te koje operande treba dovesti na ulaz ALU) te **izvršavanje** instrukcije (ALU izvršava željenu

operaciju s operandima i prosljeđuje rezultat na prethodno definiranu lokaciju)

## **11.** Detaljno opiši **fazu dohvata instrukcije**.

- sadržaj programskog brojila (adresa instrukcije) se prebacuje u adresni međuregistar na strani memorije

- *Memory Managment System* pronalazi instrukciju na navedenoj adresi te samu instrukciju (sadržaj)

prebacuje u podatkovni međuregistar na strani memorije

- iz podatkovnog međuregistra putem podatkovnog dijela sabirnice dolazi u instrukcijski međuregistar u

mikroprocesoru

- prva faza (faza dohvata instrukcije) završava sa instrukcijom smještenoj u instrukcijskom međuregistru

## **12.** Detaljno **opiši fazu dekodiranja instrukcije**.

- nakon dohvata instrukcije ona iz instrukcijskog međuregistra dolazi u dekoder

- dekoder utvrđuje na temelju dijela bitova instrukcije (operacijski kod) operaciju koju treba ALU provesti

- povećava programsko brojilo za 1 (tako da pokazuje na adresu slijedeće instrukcije koja će se obaviti)

- šalje upravljačke signale ALU da zna koju operaciju će obavljati slijedeći put kada će se na njen ulaz

dovesti operandi

- na temelju dijela bitova instrukcije (adresni dio) ustanovljuje adrese operanada tj. iz kojih registara dolaze

operandi i gdje se rezultat treba spremiti

- ukoliko adresa pokazuje da se operand nalazi u memoriji tada se adresa operanda prebacuje u adresni

međuregistar na strani memorije koja pronalazi željene operand, prebacuje ga u podatkovni međuregistar

na strani memorije te iz njega putem sabirnice dolazi u podatkovni međuregistar u procesoru

- druga faza završava da ALU zna koju operaciju treba obaviti, oba operanda su spremna na njenom ulazu

te zna kuda treba spremiti rezultat

## **13.** Detaljno opiši **fazu obavljanja instrukcije**.

- ALU zna koju operacjiu treba obaviti, operandi su spremni na ulazima te zna kuda treba spremiti rezultat

- upravljačka jedinica daje signal za „start“ te ALU obavlja operaciju i sprema rezultat

- vrijednosti zastavica koje ovise o dobivenom rezultatu pohranjuju se u registar stanja (zastavice ne

omogućuju provjeru samog rezultata već procesor ispituje je li samo izvođenje instrukcije prošlo kako

treba, ukoliko je sve u redu tada i se pretpostavlja da je i rezultat točan)

## **14.** Objasni princip **podjele memorije** na **stranice** i navedi njene **prednosti**.

- straničenje memorije je podjela memorije na stranice radi bolje organizacije i bržeg pristupa podacima

- ukoliko stranice nebi postojale (sve bi bila samo jedna velika stranica) tada bi bilo potrebno puno više

vremena za pronalazak same lokacije jer procesor traži adresu na način da ide od prve prema zadnjoj i

ispituje je li adresa ispravna tj. jednaka onoj traženoj

- kada je memorija raspoređena na stranice tada ukoliko procesor zna na kojoj stranici se nalazi određen

podatak on može puno prije naći željeni podatak (trebat će mu mnogo manje vremena za pronalazak

adrese između npr. 255 adresa i 64000 adresa) - na neki način lokacija je preciznije određena i tada

procesoru treba manje vremena za pronalazak željene adrese

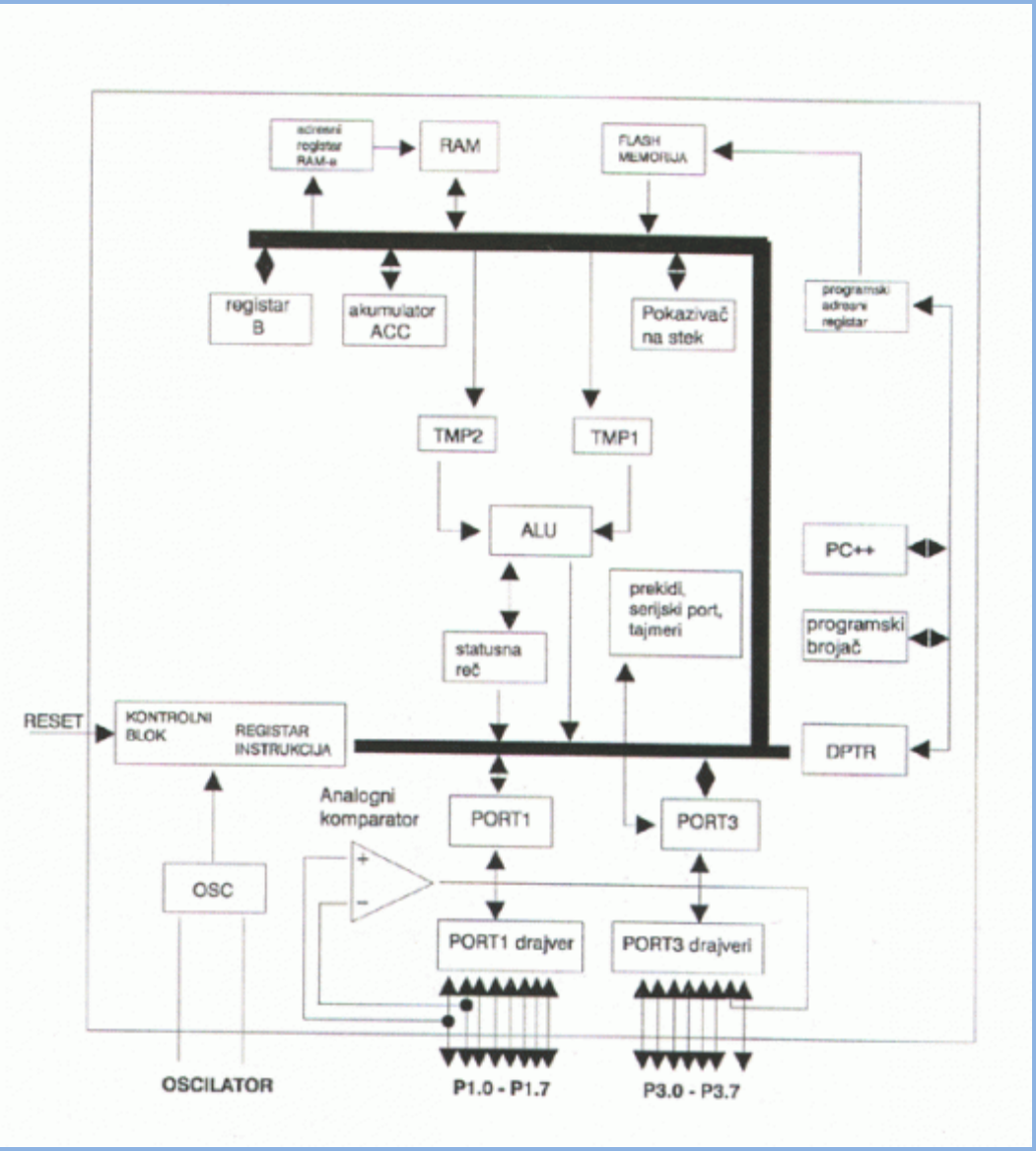
Sažetak 89C2051 i BSCOM8051:

**Konfiguracija 89c2051:**

**•Kompatibilan sa MCS-51™ skupinom**
**• 2K Bytes reprogramibilne Flash Memorije**
**– Endurance: 1,000 Write/Erase Cycles-trajanje**
**• 2.7V to 6V DC napajanje**
**•Frekvencija takta: 0 Hz to 24 MHz**
**• Two-level Program Memory Lock**
**• 128 x 8-bit slobodni RAM**
**• 15 Programmable I/O Lines - pristupi P3 i P1**
**• Dva - 16-bit Timer/Counters**
**• Šest izvora prekida (Interrupt Sources)**
**• Programibilni Serijski UART kanal**
**• Direktni LED Drive izlazi**
**• On-chip Analog Comparator - (P1.0 i P1.1 analgni ulazi + unutrašnji P3.6 kao izlaz komparatora)**
**• Low-power Idle and Power-down Modes -štedni načini rada**

Atmelov AT89C2051je 8-mo bitni mikrokontroler koji u sebi sadrži 2 Kbajta Flash memorije koju je moguće više puta programirati i brisati (engl. Flash programmable and erasable read only memory). Potpuno je kompatibilan sa setom naredbi i funkcijama pinova industrijskog standarda MCS-51ä. Zbog ove kombinacije 8-bitne centralne procesorske

jedinice i Flash memorije u jednom čipu, Atmel-ov AT89C2051 predstavlja zanimljiv mikrokontroler sa kojim je moguće u mnogim slučajevima napraviti vrlo fleksibilne i jeftine aplikacije.

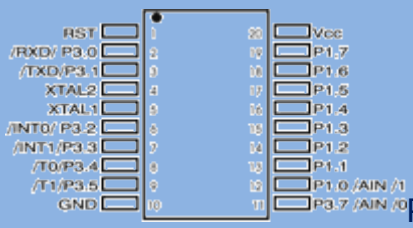



Ovaj procesor odlikuju sljedeće standardne osobine: 2 Kbajta Flash-a, 128 bajtova internog RAM-a, 15 ulazno-izlaznih linija, dva 16-to bitna tajmera-brojača, 5 izvora prekida u dva nivoa, potpuni dupleks serijski port, precizni analogni komparator, integriran oscilator i kolo za takt. Uz to, AT89C2051 je projektiran za potpuni statički rad sve do frekvencije 0 Hz, i podržava dva načina rada koje je moguće softverski birati, za smanjenje vlastite potrošnje. "Idle" mod zaustavlja centralnu procesorsku jedinicu, pri čemu RAM, tajmeri-brojači, serijski port i prekidni sistem, nastavljaju da funkcionirati. Power Down mod čuva sadržaj RAM-a ali zamrzava rad oscilatora, onemogućavajući ostale funkcije čipa, sve do hardverskog reseta.

## Opis pinova

Port 1 - Port 1 je 8-bitni dvosmjerni ulazno-izlazni port. Pinovi porta P1.2 do P1.7 imaju interne otpornike prema pozitivnom naponu napajanja

(eng. pullup). Pinovi P1.0 i P1.1 takođe služe i kao neinvertujući (AIN0) i neinvertujući (AIN1) ulaz-respektivno, preciznog analognog komparatora, koji je ugrađen u samom procesoru. Izlazni stepeni Porta 1 mogu primiti 20mA, i mogu direktno pogoniti led indikatore. Kada se jedinice upišu na pinove Porta 1,oni se mogu koristiti kao ulazi. Kada se pinovi P1.2 do P1.7 koriste kao ulazi i izvana dovedu na nizak nivo (logičku 0), oni će odavati struju (IIL), zbog internih pullup otpornika. Prilikom programiranja i verifikacije programa, Port 1 prima podatke.



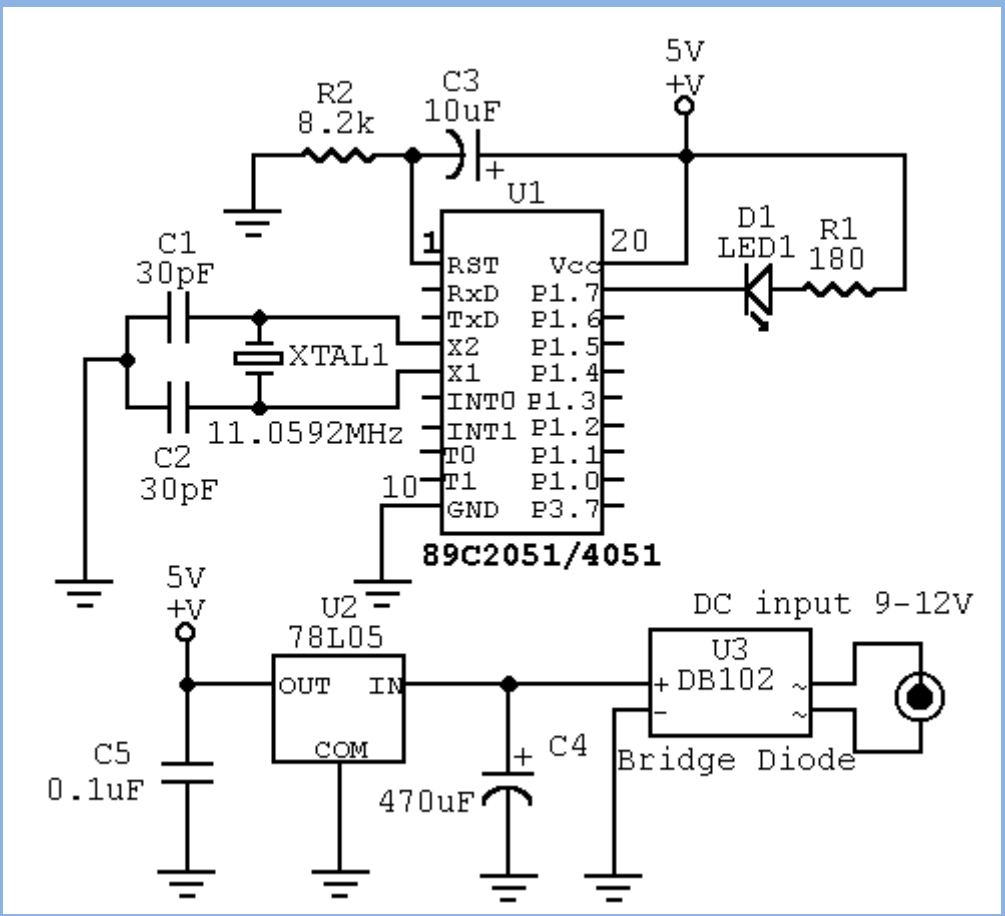

Port 3 - Pinovi Porta 3 P3.0 do P3.5 i P3.7 su dvosmjerni ulazno-izlazni pinovi, sa internim otpornicima prema pozitivnom naponu napajanja. P3.6 je interno spojen na analogni komparator koji se nalazi u komponenti, i nije dostupan kao ulazno-izlazni pin opšte namene. Kao i kod Porta1, izlazni stepeni Porta 3 mogu primiti struju intenziteta 20mA, a kada se koriste kao ulazi (kada se na njih upiše jedinica), odaju struju zbog internih pullup otpornika. Port 3 takođe prima neke upravljačke signale prilikom programiranja i verifikacije EEPROM-a.
RST - Reset ulaz. Držanje ovog pina na "1" u trajanju od 2 mašinska ciklusa dovodi do resetovanja komponente. Dok je reset ulaz na visokom potencijalu, svi ulazno-izlazni pinovi se postavljaju na "1".
XTAL1 - Ulaz invertujućeg pojačavača oscilatora
XTAL2 - Izlaz invertujućeg pojačavača oscilatora.

## Ograničenja kod određenih instrukcija

Procesor 89C2051 je potpuno kompatibilan sa MCS-51 arhitekturom, i može biti programiran koristeći MCS-51 set instrukcija. Međutim, prilikom programiranja ove komponente mora se imati u vidu sledeće: Sve naredbe skoka i naredbe grananja, moraju biti ograničene tako da odredišna adresa padne unutar fizičkog prostora programske memorije procesora, koja iznosi 2K za ovu komponentu. Na primer, LJMP 7E0H je korektna naredba , dok LJMP 900H nije. Naredbe grananja: ACALL, LJMP, ACALL, AJMP, SJMP, JMP @A+DPTR Ove naredbe bezuslovnog grananja će se izvršavati korektno sve dok programer vodi računa da odredišna adresa unutar fizičkih okvira programske memorije (adrese 00H do 7FFH za 89C2051). Prekoračenje ove fizičke granice može dovesti do nepoznatog ponašanja programa. Za sljedeće naredbe također važe ista pravila CJNE [...], DJNZ [...], JB, JNB, JC, JNC, JBC, JZ, JNZ AT89C2051 sadrži 128 bajtova interne memorije za podatke, što je upravo i maksimalna veličina steka. Pristup eksternoj memoriji za podatke, nije podržan kod ovog procesora, kao ni izvršenje programa iz eksterne programske memorije. Zbog toga naredbe tipa MOVX[...] nebi trebale da

se koriste u programima.Uobičajeni 80C51 asembler će prevesti naredbe, iako one krše prethodno nabrojana ograničenja. Isključiva je odgovornost programera da izbjegne greške ovog tipa.
U cilju smanjenja potrošnje struje, AT89C2051 podržava dva načina rada: IDLE MOD
U Idle modu procesor se prebacuje u stanje spavanja, dok periferije na čipu ostaju aktivne. Ovaj mod se aktivira softverski. Sadržaj RAM-a na čipu i registri posebne namjene ostaju neizmjenjeni dok se procesor nalazi u ovom modu. Idle mod se prekida bilo kojim dozvoljenim prekidom, ili hardverskim resetom.
POWER DOWN MODE
U power down modu oscilator se zaustavlja, a naredba koja je izazvala ovaj mod je poslednja izvršena naredba. Interni RAM i registri posebne namene zadržavaju svoje vrednosti do prekida power down moda. Jedini izlaz iz ovog moda je hardverski reset. Reset redefinira registre posebne namene, ali interni RAM ostaje nepromenjen. Reset ne treba da se aktivira dok Vcc ne uspostavi svoju normalnu radnu vrednost i mora ostati aktivan dovoljno dugo da dopusti da se oscilator resetira i stabilizira.

Osnovni spoj:



Na gornjem dijelu sheme je mikrokontroler kojem se obavezno mora dati napajanje oko 5V i kristal 24MHz.
R3 i C2 služe za pravilan početak rada kontrolera, ako ih nema isto će raditi ali ponekad pri uključivanju treba resetirati.
Na donjem dijelu slike je riješeno napajanje kontrolera ali taj dio možese riješiti i baterijom.

*Cilj vježbi: Savladavanje osnovnih programskih zadataka pomoću mikroupravljača*

*AT89C2051 u Bascom programskom jeziku*

**Tipovi varijabli**

**BIT (1/8) bajta**

**BYTE 8 bitova ili numerička vrijednost od 0 do 255**

**INTEGER (dva bajta) ili numerička vrijednost od -32 768 do 32 787**

**WORD (dva bajta / 16 bita) ili numerička vrijednost od 0 do 65 535**

**LONG (četiri bajta) ili numerička vrijednost od -2147483648 do 2147483647**

**SINGLE ( četiri bajta / 32 bitni broj) od 0 do 4 294 967 296**

**STRING (do 254 bajta)**
**String definiran kao 10 bajtova zauzima prostor kao 11 bajtova.**

**Ograničenja u definiranju varijabli:**

| Tip varijable | Maksimalni broj varijabli |
|---|---|
| Bit | 120 |
| Byte | 20 |
| Word | 10 |
| Integer | 10 |

| DEKLARACIJE | |
|---|---|
| ALIAS | Davanje alternativnog imena |
| DECLARE | Definicija potprograma |
| DEFINT | Sve nedefinirane varijable postaju tipa INT |
| DEFBIT | Sve nedefinirane varijable postaju tipa BIT |
| DEFBYTE | Sve nedefinirane varijable postaju tipa BYTE |
| DEFWORD | Sve nedefinirane varijable postaju tipa WORD |
| DIM | Definicija varijabli |

| LOGIČKE I ARITMETIČKE OPERACIJE | |
|---|---|
| ERASE | Brisanje varijable |
| DECR | Smanjenje vrijednosti varijable za 1 |
| HIGH | Gornji *nibble* varijable |
| HIHGW | Dva gornja bajta varijable tipa LONG |
| LEN | Računanje dužine stringa |
| INCR | Povečanje vrijednosti varijable za 1 |
| LOW | Donji *nibble* varijable |
| LOWW | Dva donja bajta varijable tipa LONG |
| MAX | Traženja najveće vrijednosti polja |
| MIN | Traženje najmanje vrijednosti polja |
| MOD | Ostatak dijeljenja |
| RESET | Postavljanje adresnog bita na „0“ |
| ROTATE | Rotiranja sadržaja varijable lijevo/desno |
| SET | Postavljanje adresnog bita na „1“ |
| SHIFT | Pomicanje sadržaja varijable lijevo/desno |
| SWAP | Zamjena vrijednosti dviju varijabli istog tipa |
| PRETVORBE | |
| ABS | Računanje apsolutne vrijednosti varijable |
| ASC | Pretvaranje STRINA u odgovarajući ASCII kod |
| AVG | Računanje prosječne vrijednosti polja |
| BCD | Pretvaranje vrijednosti varijable u odgovarajući BSC kod |
| CHR | Pretvaranje numeričke vrijednosti u odgovarajući znak |
| CONST | Definiranje konstante |
| FUSING | Oblikovanje decimalnog broja za ispis |
| HEX | Izračun HEX-a vrijednosti i ispis u obliku STRING –a |
| HEXVAL | Pretvaranja STRINGA u HEX oblik |

| | |
|---|---|
| LCASE | Pretvaranje STRING-a u velika slova |
| LEFT | Izdvaja prvih n znakova STRING-a s lijeve strane |
| MAKEDEC | Pretvaranje BSC vrijednosti u decimalnu |
| MAKEBCD | Pretvaranje BCD vrijednosti u decimalnu |
| MAKEINT | Spajanje dva bajta u jedan WORD ili INTEGER |
| MID | Izdvaja n znakova iz STRING-a |
| REPLACE | Zamjena slogova u STRING-u |
| RIGHT | Izdvaja prvih n znakova STRING-a s desne strane |
| SHIFTIN | Prihvaća BIT-ove na zadanom pinu i slaže ih u varijablu |
| SHIFTOUT | Sadržaj varijable šalje BIT po BIT na zadani pin |
| STR | Pretvara broj u njegov tekstualni ekvivalent |
| STRING | Formira STRING od nekoliko jednakih znakova |
| UCASE | Pretvaranje STRING-a u velika slova |
| VAL | Pretvaranje STRING-a u njegov brojčani ekvivalent |

| NAREDBE ZA TVORBU PROGRAMSKE STRUKTURE | |
|---|---|
| #IF | Uvjetno grananje programa, THEN naredba nije potrebna |
| #ELSE | Uvjetno grananje programa, ide uz #IF naredbu |
| #ENDIF | Kraj uvjetnog grananja programa |
| $ASM-$END ASM | Početak-završetak asemblerskih rutina unutar Bascom-a |
| CALL | Pozivanje i izvođenje potprograma |
| DO | Početak beskonačne petlje |
| ELSE | Izvrši naredbu ako postavljeni IF uvjet nije ispunjen |
| END | Kraj programa |
| END IF | Kraj IF petlje |
| EXIT | Izlaz iz petlje |
| FOR | Početak petlje sa zadanim brojem izvršenja |
| GOSUB | Poziv potprograma |
| GOTO | Bezuvjetni skok |
| IF | Testiranje uvjeta |
| LOOKUP | Čitanje vrijednosti iz tablice |
| LOOKUPSTR | Čitanje STRING-a iz tablice |
| LOOP | Kraj DO petlje |
| NEXT | Kraj FOR petlje |
| ON Value | Grananje s obzirom na vrijednost varijable |
| READ | Sekvencialno čitanje podataka iz tablice |
| RESTORE | Omogući naredbi READ čitanje iz tablice |
| SELECT | Izvrši rutinu po ispunjenju uvjeta |
| THEN | Dio IF petlje |
| TO | Uvjet do čijeg se ispunjenja izvodi FOR petlja |
| WHILE-VEND | While petlja se izvodi sve dok je uvjet ispunjen |
| SUB | Početak potprograma |

| OČITAVANJE I ISPIS PODATAKA | |
|---|---|
| CONFIG DEBOUNCE | Definiranje vremenskog intervala DEBOUNCE naredbe |
| CONFIG GETRC | Konfiguriranje GETRC naredbe |
| DEBOUNCE | Definiranje DEBOUNCE ulaznog pina |
| GETAD | Očitavanje digitalne vrijednosti iz A/D pretvarača |
| GETAD2051 | Očitavanje digitalne vrijednosti iz AT892051/4051 mikro kontrolera |
| GETRC | Računanje vrijednosti otpornika i kondenzatora |
| P0, P1, P2, P3 | Port P0, P1, P2, P3 |
| READMAGCARD | Očitavanje podataka iz čitača magnetskih kartica |
| SOUND | Generiranje tona na izlaznom U/I pinu |

| PREKIDI | |
|---|---|
| DISABLE | Onemogući prekid |
| ENABLE | Omogući prekid |
| LOAD | Definicija vrijednosti timera |
| ON Interrupt | Grananje s obzirom na pojavu prekida |
| PRIORITY | Definicija prioriteta prekida |
| RETURN | Povratak iz potprograma |
| CONFIG TIMER0 | Odabir načina rada za TIMER0 |
| CONFIG TIMER1 | Odabir načina rada za TIMER1 |
| CONFIG TIMER2 | Odabir načina rada za TIMER2 (samo za 8052) |
| START | Pokretanje timera/countera |
| STOP TIMER | Zaustavljanje timera/countera |
| COUNTER | Postavljanje/čitanja vrijednosti registra Counter |

| OPĆE NAREDBE | |
|---|---|
| $NOBREAK | Onemogući izvođenja BREAK naredbi |
| $NOINIT | Ascom ne radi inicijalizaciju mikro kontrolera |
| $OBJ | Uključivanje Intelovog OBJ koda |
| $REGFILE | Određivanje upotrebe DAT datoteke određenog mikro kontrolera |
| $SIM | Simulacija programa bez vremenskih kašnjenja |
| BREAK | Generiranje prekidnih točaka za simulator |
| CONFIG WATCHDOG | Odabir vremenskog intervala WD timera |
| CONFIG SERVOS | Odabir U/I pinova kontrolera na koje su spojeni servomotori |
| CONFIG ADUC812 | Konfiguriranje ADuC812 mikeo kontrolera |

| | |
|---|---|
| DATA | Početak tabele s podacima koje čitamo naredbom READ |
| IDLE | Prebacivanje mikro kontrolera u IDLE mod rada |
| INSTR | Vrati adresu pozicije znaka u STRING |
| POWERDOWN | Prebacivanje mikro kontrolera u POWER DOWN mod rada |
| REM | Komentar |
| RND | Generiranje slučajnog broja |
| SPACE | Generiranje praznih mjesta |
| STOP | Zaustavljanje programa |

| I2C FUNKCIJE | |
|---|---|
| CONFIG SDA | Odabir U/I pina za SDA vod |
| CONFIG SCL | Odabir U/I pina za SCL vod |
| CONFIG I2CDELAY | Odabir kašnjenja kod I2C komunikacije |
| I2CRECEIVE | Čitanje podatka sa I2C sabirnice |
| I2CSEND | Slanje podataka na I2C sabirnicu |
| I2CSTART | Generiranje start uvjeta na I2C sabirnici |
| I2CSTOP | Generiranje stop uvjeta na I2C sabirnici |
| I2CRBYTE | Prijem 1 bajta iz neke I2C jedinice |
| I2CWBYTE | Slanje 1 bajta na I2C jedinicu |

| NAREDBE ZA RAD S ALFANUMERIČKIM I GRAFIČKIM LCD MODULOM | |
|---|---|
| $BGF | Prikaz Bascom grafičkih datoteka na grafičkom displeju |
| $LCD | Generiranje 8-bitne adrese za aktiviranje LCD-a |
| CLS | Brisanje displeja i postavljanje kursora u početni položaj |
| CONFIG LCD | Odabir tipa displeja |
| CONFIG LCDBUS | Odabir 4 odnosno 8 bitne komunikacije |
| CONFIG LCDPIN | Odabir U/I pinova mikro kontrolera za komunikaciju sa LCD |
| CONFIG GRAPHLCD | Konfiguriranje grafičkog LCD-a |
| CURSOR BLING | Treperenje kursora |
| CURSOR NO BLINK | Kursor ne treperi |
| CURSOR ON/OFF | Kursor uključen / isključen |
| DEFLCDHAR | Definiranje vanjskih znakova na LCD –u |
| DISPLAY ON/OFF | Uključivanje / isključivanje displeja |
| FOURTHLINE | Postavljanje kursora na početak četvrtog reda LCD-a |
| HOME | Postavljanje kursora na početak LCD-a |
| LCD | Ispis konstante /varijable na LCD |
| LCDHEX | Ispis konstante /varijable u HEX kodu na LCD |
| LOCATE | Postavlja kursor na adresu x,y x-horizontalno, y-vertikalno |
| LOWERLINE | Postavljanje kursora na početak drugog reda displeja |
| PSET | Postavlja ili briše jedan piksel na grafičkom displeju |

| | |
|---|---|
| SHIFTCURSOR | Pomicanje kursora lijevo/desno |
| SHIFTLCD | Pomicanje sadržaja ispisanog na LCD lijevo / desno |
| SHOWPIC | Prikaz BGF slike na grafičkom displeju |
| THIRDLINE | Postavlja kursor na početak trećeg reda displeja |
| UPPERLINE | Postavlja kursor na početak prvog reda displeja |

| SPI KOMUNIKACIJA | |
|---|---|
| CONFIG SPI | Odabir U/I pinova za SPI sabirnicu |
| SPIIN | Čitanje SPI sabirnice |
| PIOUT | Ispis preko SPI sabirnice |

| 1 WIRE KOMUNIKACIJA | |
|---|---|
| 1WRESET | Reset 1WIRE sabirnice |
| 1WREAD | Prihvat preko 1WIRE sabirnice |
| 1WWRITE | Slanje preko 1WIRE sabirnice |
| CONFIG 1WIRE | Odabir U/I pina za 1WIRE komunikaciju |

| SERIJSKA KOMUNIKACIJA PREKO UART-a | |
|---|---|
| $BAUD | Definiranje brzine komunikacije preko UART-a |
| $CRYSTAL | Definiranje frekvencije sistemskog takta |
| $SERIALINPUT | Određivanje imena rutine u kojoj se vrši serijski prijem podataka |
| $SERIALINPUT2LCD | Kompajler preusmjeri sve podatke primljene UART-om na LCD |
| $SERIALOUTPUT | Kompajler preusmjeri sve podatke na UART |
| CLOSE | Zatvaranje serijskog kanala |
| CONFIG BAUD | Odabir brzine UATR-a |
| CONFIG MICROWIRE | Odabir U/I pinova za microwire komunikaciju |
| CONFIG PRINT | Konfiguriranje print naredbe |
| GET | Čitanje bajta iz UART-a |
| GETRC5 | Čitanje IR koda po RC5 protokolu |
| INKEY | Računanje ASCII vrijednosti prvog bajta primljenog iz UART-a |
| INPUT | Unos podataka preko PC tipkovnice |
| INPUTBIN | Čitanje binarne vrijednosti iz UART-a |
| INPUTHEX | Unos podataka preko PC tipkovnice |
| OPEN | Otvaranje programskog UART kanala |
| OUT | Slanje bajta na U/I port ili vanjsku memoriju |
| PRINT | Ispis varijable ili konstante preko UART-a |
| PRINTBIN | Ispis binarne vrijednosti varijable preko UART-a |
| PRINTHEX | Ispis HEX vrijednosti varijable preko UART-a |
| PUT | Slanje bajta na programski UART |
| SPC | Generiranje praznih mjesta i slanje na UART ili LCD |
| WAITKEY | Čekanje do primitka znaka preko UART-a |

| RAD S MEMORIJOM | |
|---|---|
| $DEFAULT XRAM | Spremanje svih varijabli u XRAM |
| $IRAMSTART | Određivanje adresa interne memorije u koju se spremaju varijable |
| $LARGE | Omogućuje adresiranje do 64K adresnog prostora |
| $NOSP | Onemogući pokazivač stoga |
| $RAMSIZE | Određivanje veličine vanjskog RAM-a |
| $RAMSTART | Određivanje lokacije vanjskog RAM-a |
| $ROMSTART | Određivanje početne lokacije ROM-a |
| CPEEK | Ispis vrijednosti određene memorijske lokacije |
| INP | Čitanje vrijednosti bajta s određene lokacije |
| PEEK | Čitanje vrijednosti bajta spremljenog u određenoj memorijskoj lokaciji |
| POKE | Spremanje vrijednosti bajta u određenu memorijsku lokaciju |
| READEEPROM | Čitanja iz internog EEPROMA |
| VARPTR | Čitanja adrese memorijske lokacije određene varijable |
| WRITEEPROM | Upis u interni EEPROM |

| KAŠNJENJA | |
|---|---|
| BITWAIT | Čekanje dok neki bit nije postavljen / izbrisan |
| DELAY | Čekanje od 100µ za 12MHz ulazni klok |
| WAIT | Kašnjenje u sekundama |
| WAITMS | Kašnjenje u mili sekundama |

| OSTALO | |
|---|---|
| $INCLUDE | Učitavanje ASCII datoteka koje sadrže Bascom program |
| $NONAN | Postavljanje neobičnih varijabli u 0.0 |
| $NONULL | Promjena načina rada DANA naredbe |
| $TIMEOUT | Definicija čekanja kod serijske komunikacije |