Katedra za elektroniku
021/59-449
kel@uns.ns.ac.yu

Mikroprocesorska elektronika

MIKROKONTROLER 8051/8052

Skripta za vežbe

Miloš Slankamenac, Kalman Babković, Milan Nikolić i Ivan Mezei

Novi Sad, 2004.

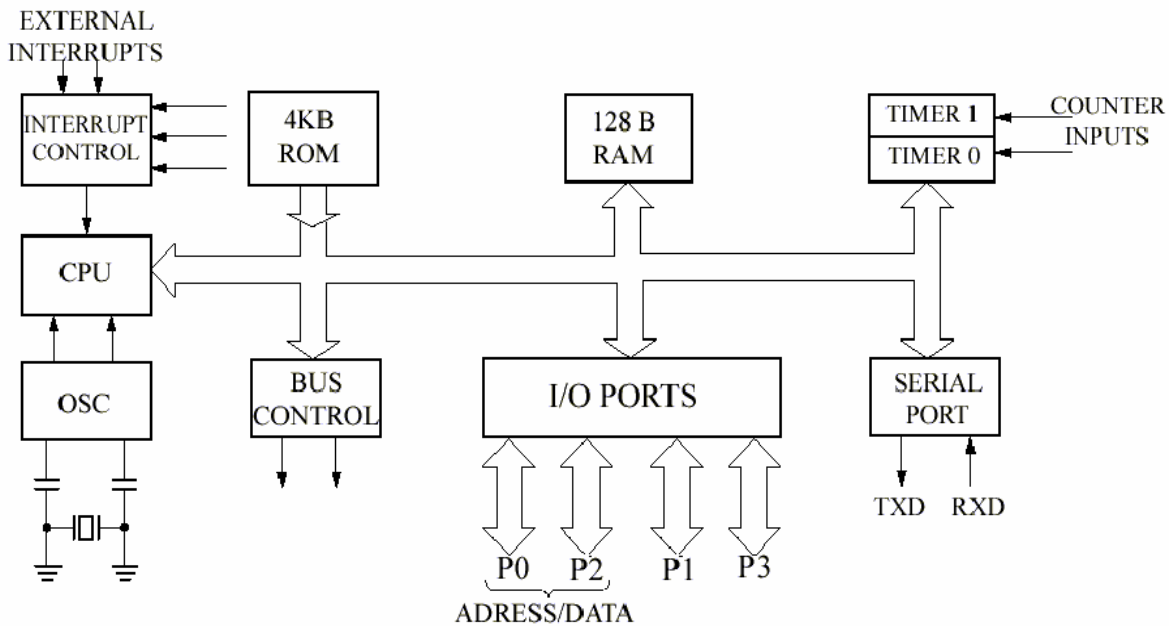
SADRŽAJ

1. ARHITEKTURA MIKROKONTROLERA INTEL 8051

U familiji INTEL-ovih mikrokontrolera izvedenih u vidu jednog čipa mikrokontroler, INTEL 8051 je sintetizovan namenski da služi kao procesor u sistemima digitalnog upravljanja. Osnovna varijanta čipa INTEL 8051 ima strukturu prikazanu na sl.1.1. Unutar čipa se nalaze centralna procesorska jedinica (CPU), ROM memorija, RAM memorija, tajmeri, brojači, paralelni i serijski ulazi/izlazi. Sem osnovne verzije, na tržištu se može naći veliki broj varijanti ovog mikrokontrolera, koji se od osnovne verzije razlikuju u broju i vrsti periferijskih komponenti, veličini i tipu ugrađene memorije (RAM i ROM), kao i po mehaničkoj izvedbi (tip kućišta i broj nožica). Ovo uputstvo obuhvata osnovnu varijantu mikrokontrolera 8051, uz opis dodatnih elemenata za mikrokontroler 8052.

Osnovne karakteristike mikrokontrolera INTEL 8051 su:

- 8-bitni CPU;
- 4 KB (8 KB za 8052) interne ROM ili EPROM memorije za čuvanje programa, sa mogućnošću proširenja do 64 KB spoljašnje memorije;
- 128 (256 za 8052) bajtova RAM memorije namenjene za upisivanje i čitanje podataka. U ovoj memoriji se nalaze 4 registarske banke od po 8 registara, kao i stek memorija;
- 64 KB adresnog prostora memorije podataka;
- 64 KB adresnog prostora programske memorije;
- 8-bitni pokazivač stek memorije, koji pokriva internu memoriju
- Dva (tri za 8052) programabilna 16-bitna tajmera/brojača;
- Programabilni serijski ulaz/izlaz (puni dupleks).
- Četiri 8-bitna ulazno/izlazna priključka (portovi P0, P1, P2 i P3);
- Tajmerski i ulazno/izlazni prekidi sa dva nivoa prioriteta;
- 111 naredbi;
- Aritmetičko-logička jedinica (ALU) koja može da izvršava aritmetičke operacije sabiranja, oduzimanja, množenja i deljenja, kao i logičke operacije I, ILI, EXILI, komplement i negacija;
- 256 adresabilnih bita za rad sa Bulovom algebrom, od toga 128 bita u internoj memoriji i 128 bita pridruženih postojećim internim registrima.

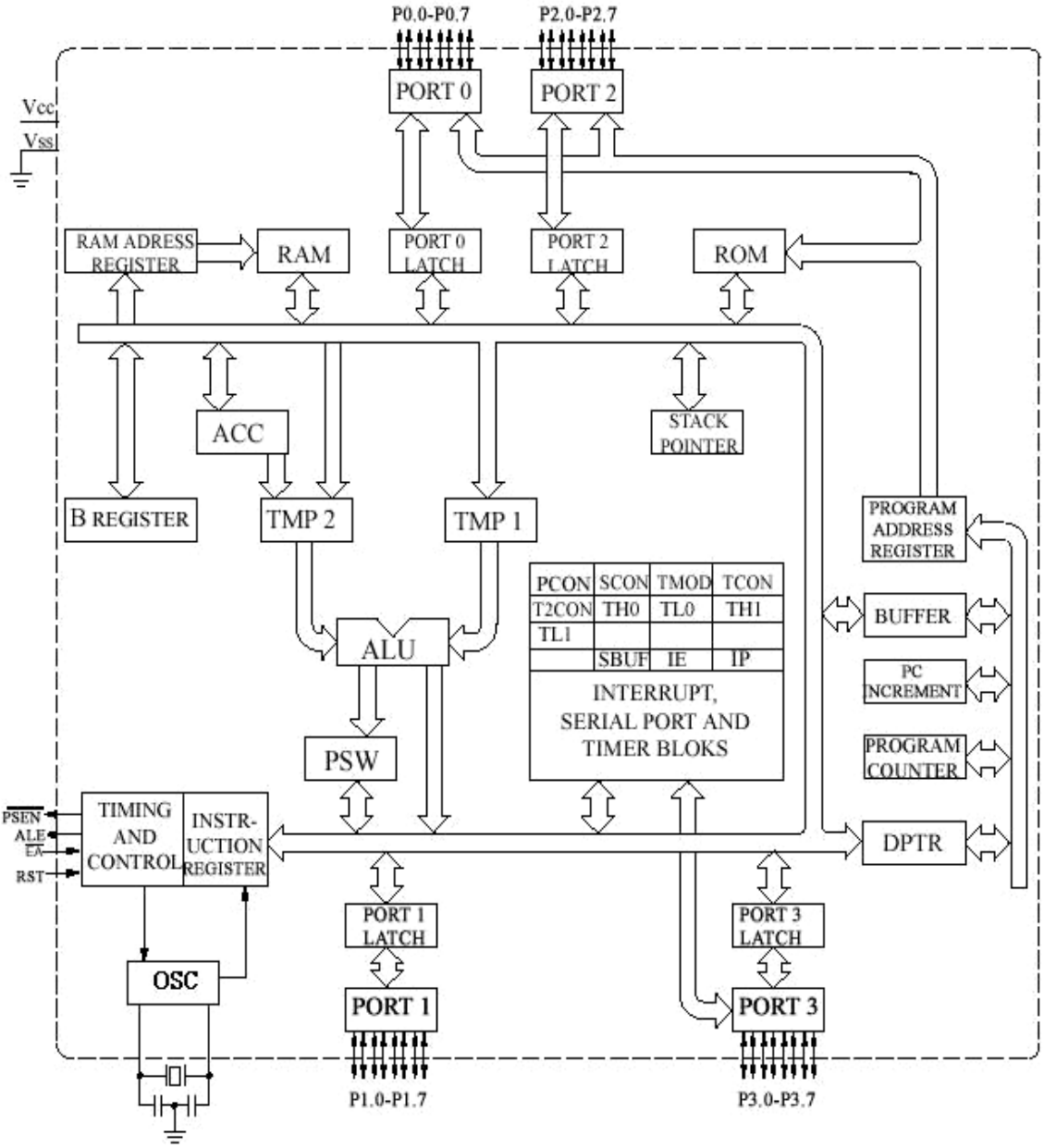


Slika 1.1 Struktura čipa mikrokontrolera INTEL 8051

Ovako sintetizovan mikrokontroler u mnogome olakšava posao projektanta hardvera, smanjuje ukupno hardversko okruženje i broj spoljašnjih veza i na taj način povećava pouzdanost sistema.

Na sl. 1.2 prikazana je detaljnija arhitektura mikrokontrolera INTEL 8051. Ovde se vide ranije pomenute komponente: aritmetičko-logička jedinica (ALU) sa parom registara (TMP1, TMP2) za privremeno upisivanje podataka, akumulator (ACC) sa pomoćnim registrom B, statusni registar (PSW),

registar naredbi sa dekoderom, programski brojač (PC) i registar za inkrementiranje (povećanje za 1) programskog brojača (PC-incrementer).



Slika 1.2 Detaljnija struktura mikrokontrolera INTEL 8051

Pomoćni registar **B**, koji se naziva multiplikativnim registrom, služi za smeštanje drugog operanda za aritmetičke operacije množenja i deljenja. Posle izvršene operacije množenja ili deljenja u njemu se nalazi viši bajt rezultata množenja ili ostatak deljenja, respektivno. Zajedno sa **ACC** pomoćni registar B čini registarski par.
Pokazivač steka (**SP**-Stack Pointer) služi za adresiranje vrha (najviše lokacije) stek memorije. Ovaj registar se inkrementira prilikom upisivanja podataka u stek, a dekrementira prilikom čitanja podataka iz ove memorije.

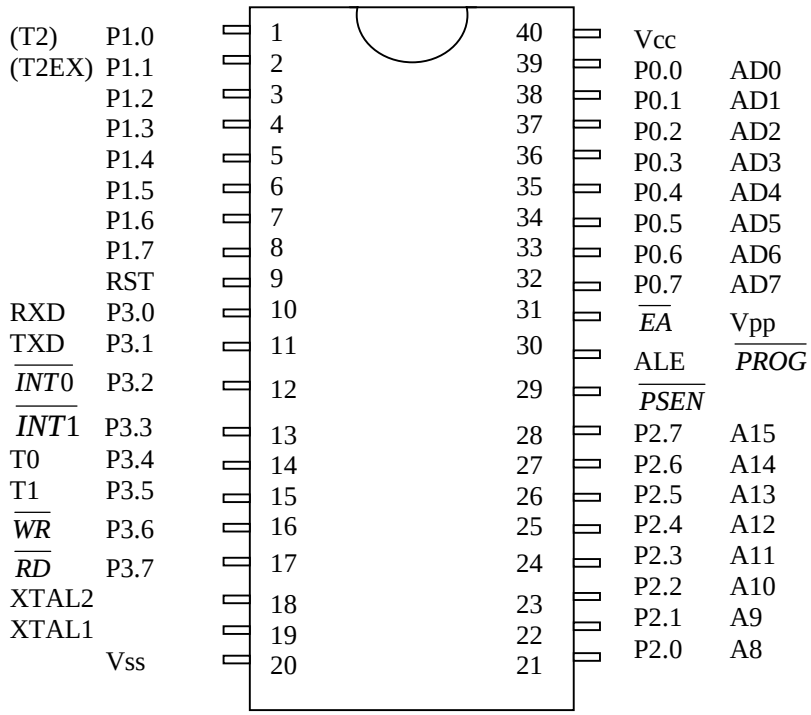
Mikrokontroler INTEL 8051 poseduje 4 prihvatna registra (**LATCH**-a) za čuvanje stanja izlaza na portovima P0, Pl, P2 i P3.
Registar serijskog prenosa (**SBUF**) služi za upis podatka koji se šalje i čitanje podatka koji se prima preko serijske veze.
Registarski parovi (**TH0, TL0**) i (**TH1, TL1**) čine dva 16-bitna tajmera ili brojača.
Za kontrolu i upisivanje statusa prilikom prekida (**INTERRUPT**-a) za tajmere, brojače i za serijski prenos podataka koriste se registri specijalne namene označeni na sl 1.2 sa IP, IE, TMOD, TCON, SCON i PCON.

IP služi za određivanje nivoa prioriteta prekida, **IE** za maskiranje (dozvolu ili zabranu) prekida, **TMOD** i **TCON** za određivanje načina rada tajmera i brojača, **SCON** za kontrolu serijskog prenosa i **PCON** za dodatnu kontrolu serijskog prenosa i režim rada mikrokontrolera.
Memorijski adresni prostor mikrokontrolera INTEL 8051 je podeljen u dva osnovna dela: adresni prostor rezervisan za programe (Code Address Space) i adresni prostor rezervisan za podatke (Data Address Space).
Mikrokontroler može da adresira 64KB programske memorije; interno u samom čipu ima 4KB, a ostatak od 60KB je predviđen kao spoljašnja memorija. **Programska memorija** je tipa ROM, a sem izvođenja programa moguće je i čitanje nekih konstantih podataka.

Memorija podataka je tipa RAM, a sastoji se iz interne (128/256 bajtova) i eksterne memorije (do 64KB). Pri tome, ako nema previše podataka, eksterna memorija ne mora da se koristi. Internu memoriju je moguće adresirati direktno registarski ili registarski indirektno. Od toga u prostoru od 16 bajtova moguće je adresirati svaki bit (Bit Addressable Segment). U okviru interne memorije podataka nalazi se i **stek**, koji je organizovan tako da se poslednji upisani podatak prvi čita (Last Input First Output ili **LIFO**-princip), a koristi se za privremeno čuvanje sadržaja brojača naredbi prilikom poziva na potprograme i za pamćenje adrese izvršenja programa u slučaju prekida.
Eksternoj memoriji se uvek pristupa indirektno, preko odgovarajućih registara.

2. RASPORED I FUNKCIJE POJEDINIH NOŽICA

Na slici 2.1 prikazan je izgled mikrokontrolera 8051. Ako su signali nadvučeni to znači da je aktivan nizak nivo signala a pasivan visok.



Slika 2.1 Raspored nožica mikrokontrolera

2.1 Opis funkcija pojedinih nožica

Od 1 do 8 - Port 1:
Svaka od ovih nožica može da se koristi kao ulazni ili izlazni priključak, prema potrebi. Za 8052, P1.0 (T2) je spoljašnji brojački ulaz za tajmer 2 a P1.1 (T2EX) je spoljašnja kontrola (triger) za tajmer 2.

9 - Reset:
Visok logički nivo na ovom ulazu resetuje sve interne registre (registre dovodi u stanje 00000000), sa sledećim izuzecima:

- P0, P1, P2 i P3 (izlazni registri svih spoljnih portova) se dovode u stanje 11111111
- SBUF se ne menja
- SP se dovodi u stanje 00000111 (07h)
- Neki biti u registrima IP, IE i PCON fizički ne postoje, pa tako ne mogu ni da se resetuju
- Sadržina celog internog RAM-a se ne menja

Najvažnija posledica aktiviranja RESET ulaza je da se PC (Program Counter) resetuje, tako da će započeti izvršavanje programa od adrese 0000h.

od 10 do 17 - Port 3:
Ako se koristi kao univerzalni ulaz ili izlaz, po svemu je sličan portu 1, ali na svakoj nožici ima još po neku specijalnu funkciju:

- 10 (P3.0) RXD - Serijski ulaz za asinhronu komunikaciju (mod 1, 2 i 3) ili serijski ulaz za sinhronu komunikaciju (mod 0)
- 11 (P3.1) TXD - Serijski izlaz za asinhronu komunikaciju (mod 1, 2 i 3) ili taktni (clock) izlaz sa sinhronu komunikaciju (mod 0)
- 12 (P3.2) INT0 - Ulaz za prekid (interapt) 0
- 13 (P3.3) INT1 - Ulaz za prekid (interapt) 1
- 14 (P3.4) T0 - Ulaz spoljnjeg takta za brojač 0
- 15 (P3.5) Tl - Ulaz spoljnjeg takta za brojač 1
- 16 (P3.6) WR - Signal za upis u spoljnu memoriju
- 17 (P3.7) RD - Signal za čitanje iz spoljne memorije

18 i 19 - X2 i X1:
Izlaz i ulaz internog oscilatora. Ako se koristi kvarc-kristal za stabilizaciju učestanosti oscilatora (što je najčešći slučaj), on se vezuje za ove dve nožice, s tim što na svaku nožicu (prema masi) treba dodati još po jedan kondenzator od 20-40pF. Ovoje potrebno da bi se sprečilo oscilovanje na nekom višem harmoniku. Opseg učestanosti je od 1 do 12 MHz, a izrađuju se i mikrokontroleri koji rade i na znatno višim frekvencijama.

20 - Masa

Od 21 do 28 - Port 2 ili adrese A8 do A15:
Ako se koristi mikrokontroler sa internim ROM-om i nema spoljnjeg ROM-a ili RAM-a, mogu se koristiti sve linije ovog porta kao univerzalni ulazi ili izlazi. Ako se koristi spoljna memorija, onda su ovo visoki adresni izlazi, od A8 do A15. U tom slučaju, čak i ako se koriste samo neke adrese, preostale nožice ovog porta ne mogu da se koriste kao ulazi ili izlazi.

29 - PSEN: Program Select Enable (aktiviranje spoljašnjeg ROM-a):
Normalno se ovaj izlaz spaja sa CS ili OE ulazom na spoljnom EPROM-u, jer ga mikrokontroler aktivira (dovodi na nizak nivo) svaki put kad čita bajt iz programske memorije (za kontrolu spoljašnjeg RAM-a se koriste druge nožice).

30 - ALE: Address Latch Enable (Upis u adresni registar):
Da bi sve željene funkcije spakovao u standardno kućište od samo 40 nožica, Intel je morao da pribegne multipleksiranju nekih signala. Tako je port P0 dobio dve funkcije: izlazne adrese A0-A7 i ulaz/izlaz podataka D0-D7. Pre svakog očitavanja programa iz spoljne memorije ili prozivanja RAM-a mikrokontroler na P0 prosleđuje niži bajt adresnog registra i aktivira izlaz ALE. Spoljni registar (najčešće LATCH registar tipa 373 ili 573 iz TTL 74xx familije) na visok nivo ALE memoriše stanje P0, a izlazi ovog registra se koriste kao A0-A7. U drugom delu mašinskog ciklusa mikrokontrolera P0 se koristi kao magistrala podataka (Data Bus).

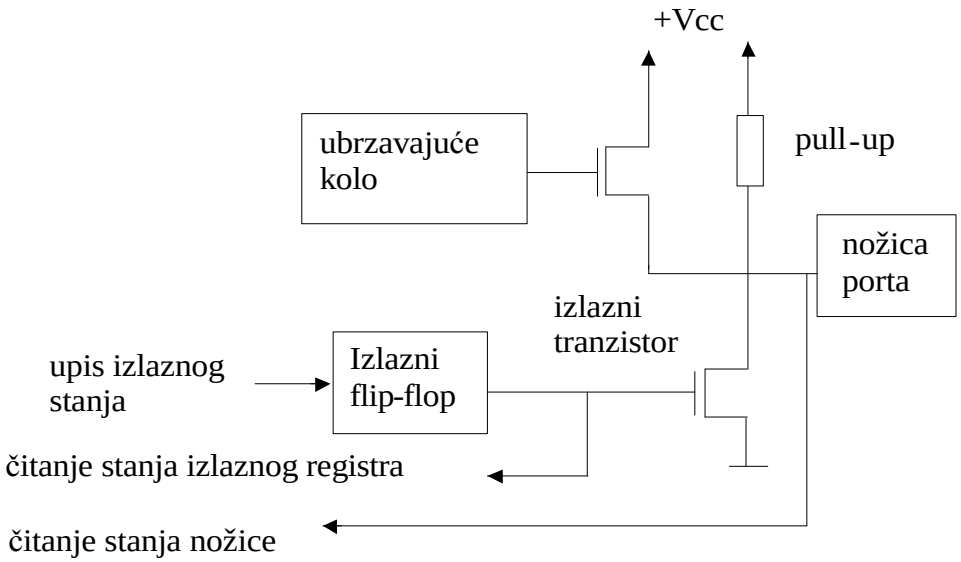
31 - EA: External Access (Pristup spoljašnjem ROM-u):
Ako je ovaj ulaz nizak, mikrokontroler će sve instrukcije čitati iz spoljnog ROM-a, bez obzira da li ima interni ROM, a ako je visok, prvih 4 KB (8051, 8751) ili 8 KB (8052, 8752) će čitati iz internog, a sve ostalo do kraja adresnog prostora iz eksternog ROM-a. Ako se koristi 8031 ili 8032, na ovaj ulaz treba uvek dovesti nizak nivo (najbolje je spojiti ga sa masom).

od 32 do 39 - Port 0, Adrese A0-A7 ili magistrala podataka:
Slično portu P2, i port P0 može da se koristi kao univerzalni ulaz i izlaz samo ako se ne koristi spoljna memorija. Ako se koristi, tada je P0 adresni izlaz za A0-A7 kad je ALE visok, a magistrala podataka (**Data Bus**) kada je ALE nizak.

40 - Napajanje +5V

2.2 Hardverska struktura portova

Kao što je prikazano, postoje četiri 8-bitna porta, pri čemu kod konfiguracija sa spoljnom memorijom portovi P0 i P2 služe za adresiranje i pristup podacima iz memorije, a P1 i P3 su univerzalni i koriste se onako kako to konkretan projekat zahteva. Svi ulazi i izlazi su kompatibilni sa TTL standardom, a to praktično znači da možemo direktno da ih spajamo sa TTL ili kompatibilnim kolima. Ipak, neke specifičnosti portova treba imati u vidu kada se projektuje okruženje ovih mikrokontrolera.
Svi portovi su realizovani sa otvorenim drejnom (**Open Drain**), uz dodatni MOSFET tranzistor u spoju aktivnog otpornika prema napajanju (pull-up, oko 50KΩ), kao na slici 2.2. Ovakvo rešenje obezbeđuje dobru logičku nulu (kada je izlazni tranzistor uključen). Kada je ovaj tranzistor isključen, tada samo aktivni otpornik određuje logičku jedinicu, pri čemu se odgovarajuća nožica ponaša i kao ulazni priključak, jer spoljašnje kolo može da obori napon na logičku nulu (rešenje je poznato i pod nazivom "ožičeno I").



Slika 2.2 Pojednostavljeni prikaz nožica porta

U slučaju kada se na izlaz porta postavlja logička jedinica, uključuje se kratkotrajno dodatni tranzistor preko ubrzavajućeg kola, čime je obezbeđeno savlađivanje izlaznih kapacitivnosti.
Zavisno od tipa instrukcije, moguće je čitanje stanja izlazne nožice, ali i stanja izlaznog flip-flopa. Instrukcije koje samo čitaju port uzimaju informaciju o stanju direktno sa nožice (označeno kao "čitanje stanja nožice"), dok instrukcije koje modifikuju port, tj. pročitaju stanje, promene ga i zatim ponovo upišu na port, pri čitanju uzimaju stanje izlaznog flip-flopa, a ne izlazne nožice (označeno kao "čitanje stanja izlaznog registra"). Ovakve instrukcije su poznate pod nazivom "Read-Modify-Write" (čitanje-izmena-upis), a primeri su:

```
ORL   P1,#72h ; Port se pročita, uradi se logička operacija OR, a zatim se vrednost upiše u port
SETB  P1.2    ; Port se pročita, postavi se jedan bit, a zatim se vrednost upiše u port
```

Primeri instrukcija koje čitaju izlazno stanje su:

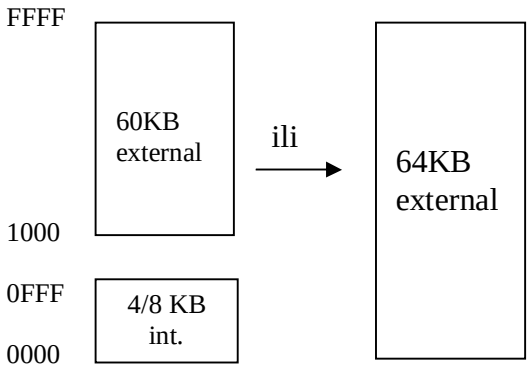
```
MOV   A,P2
ANL   C,P1.1
```

3. ORGANIZACIJA MEMORIJE

Memorijski adresni prostor mikrokontrolera INTEL 8051 je podeljen u dva osnovna dela: adresni prostor rezervisan za programe (Code Address Space) i adresni prostor rezervisan za podatke (Data Address Space).

3.1 Programska memorija

Mikrokontroler može da adresira 64KB programske ROM memorije, slika 3.1.

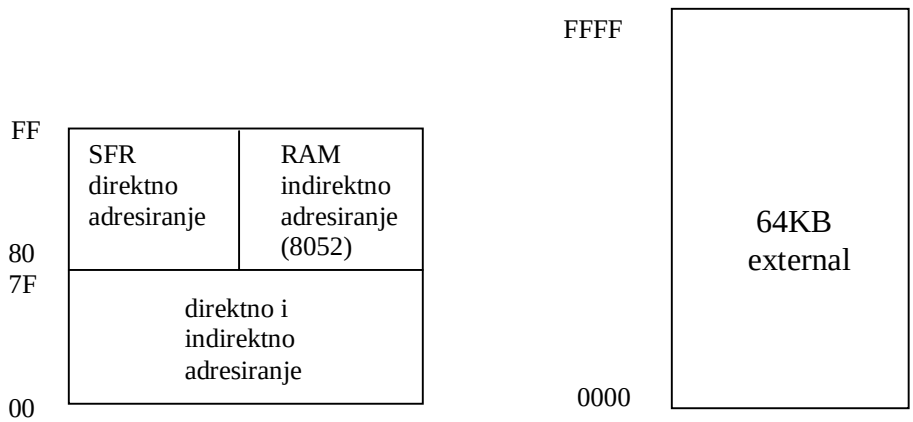


Slika 3.1 Programska memorija

Donjih 4/8 KB programske memorije može biti interna memorija (nalazi se unutar čipa) ako je $\overline{EA}$=Vcc, ili spoljašnja memorija ako je $\overline{EA}$=0. Signal $\overline{PSEN}$ se koristi za pristup spoljašnjoj programskoj memoriji, tako što se vodi na $\overline{OE}$ ulaz EPROM-a. Kada se koristi spoljašnja memorija, port P2 služi za adresiranje višeg bajta a P0 za adresiranje nižeg bajta te memorije.

3.2 Memorija podataka

Memorija podataka je tipa RAM, slika 3.2, a sastavljena je od interne (128 bajtova) i eksterne (do 64KB) memorije. Sem toga, u zoni interne memorije (u opsegu adresa 128 - 255) se nalazi i blok specijalnih registara (**S**pecial **F**unction **R**egisters - **SFR**) za kontrolu periferijskih jedinica i rada mikrokontrolera.



Slika 3.2 Memorija podataka

F8h									*FFh*
F0h	*B*								*F7h*
E8h									*EFh*
E0h	*ACC*								*E7h*
D8h	*PSW*								*DFh*
D0h									*D7h*
C8h	*(T2CON)*	*(T2MOD)**	*(RCAP2L)*	*(RCAP2H)*	*(TL2)*	*(TH2)*			*CFh*
C0h									*C7h*
B8h	*IP*								*BFh*
B0h	*P3*								*B7h*
A8h	*IE*								*AFh*
A0h	*P2*								*A7h*
98h	*SCON*	*SBUF*							*9Fh*
90h	*P1*								*97h*
88h	*TCON*	*TMOD*	*TL0*	*TL1*	*TH0*	*TH1*			*8Fh*
80h	*P0*	*SP*	*DPL*	*DPH*				*PCON*	*87h*

Memorijska mapa SFR zone

* - T2MOD registar ne pripada standardnom mikrokontroleru 8052, ali postoji kod Atmel AT89C52.

30h-7Fh	Slobodna memorija	
20h-2Fh	Bit-adresibilna memorija	
18h-1Fh	*R0..R7*	*GRUPA 3*
10h-17h	*R0..R7*	*GRUPA 2*
08h-0Fh	*R0..R7*	*GRUPA 1*
00h-07h	*R0..R7*	*GRUPA 0*

Memorijska mapa internog RAM-a

Slika 3.3 Memorijska mapa internog RAM-a i SFR zone

Na slici 3.3, registri dati u zagradi postoje samo u mikrokontroleru 8052, a registri označeni sa dvostrukim okvirom su bit-adresabilni.

Na slici 3.4. uočavamo registar **PSW** (Program Status Word), koji je veoma važan za rad mikrokontrolera. Kod njega nije zanimljiva njegova globalna vrednost, nego isključivo sadržina pojedinačnih bita.

7	6	5	4	3	2	1	0
C	AC	F0	RS1	RS0	OV	-	P

Slika 3.4 PSW registar

C (PSW.7) Carry Flag (Bit prenosa). Ovo je ekstenzija (deveti bit) za sve aritmetičke operacije i instrukcije pomeranja (šiftovanja), a takođe je i glavni registar za 1-bitne operacije.

AC (PSW.6) Auxiliary Carry Flag (pomoćni bit prenosa), samo za BCD operacije, a odnosi se na prenos između nižeg i višeg nibla (donja i gornja grupa od po 4 bita). Koristi ga uglavnom mikrokontroler preko naredbi za BCD konverziju.

F0 (PSW.5) Fleg 0 stoji na raspolaganju programeru kao bit za univerzalnu upotrebu.

RS1 i **RS0** (PSW.4 i PSW.3) Regster Select, služe za izbor dela internog RAM-a u kome je smeštena tekuća grupa registara (registarska banka) R0-R7, u skladu sa sledećom tabelom:

RS1	RS0	Mesto u RAM-u	
0	0	Grupa 0	00h-07h
0	1	Grupa 1	08h-0Fh
1	0	Grupa 2	10h-17h
1	1	Grupa 3	18h-1Fh

OV (PSW.2) Overflow (prekoračenje). Setuje se (postavlja na 1) ako je rezultat aritmetičke operacije sa predznakom takav da ne može da se prikaže u jednom bajtu, tj. dođe do prekoračenja opsega (na primer,

sabiranjem dva pozitivna broja dobije se negativan broj ili obrnuto). Ako nema prekoračenja, ovaj bit će biti 0.

(PSW.1) Rezervisano od stane proizvođača za budući razvoj čipa.

P (PSW.0) Parnost. Ako je broj setovanih bita u akumulatoru paran, ovaj bit će biti postavljen na 1, a ako je neparan, biće resetovan na 0. Uglavnom se koristi za generisanje bita parnosti kod slanja bajta na serijski port ili za testiranje parnosti posle serijskog prijema.

Data Pointer (DPTR) je 16-bitni registar, koji se sastoji od dva 8-bitna registra: DPH (Data Pointer High – viši bajt) i DPL (Data Pointer Low – niži bajt).

Ostali registri posebne namene će biti objašnjeni u poglavljima o tajmerima/brojačima i prekidima.

4. INSTRUKCIJE

Set instrukcija za 8051 obuhvata ukupno 111 instrukcija, od kojih je 49 jednobajtnih, 45 dvobajtnih i 17 trobajtnih. Ako se uzmu u obzir i varijante istih instrukcija (na primer ista instrukcija primenjena na različite registre iz registarske banke), ukupno postoji 255 instrukcija. Generalni format koji važi u svakom asembleru za ove instrukcije je sledeći (mada postoje i varijacije u ovom formatu):

mnemonik odredište, izvor

Mnemonik nosi kod instrukcije, a operandi ***odredište*** i ***izvor*** samo nose podatke ili adresu odakle se uzimaju podaci kojima se vrši operacija. Ako postoje dva operanda, onda se rezultat operacije uvek smešta u prvi koji je naveden. Recimo, ako instrukcija glasi ADD A,B to je isto kao da smo u nekom višem programskom jeziku napisali A=A+B.

4.1 Prenos podataka

MNEMONIC	OPERATION	ADRESSING MODES				EXECUTION TIME (μs)
		DIR	IND	REG	IMM	
MOV A, <src>	A= <src>	X	X	X	X	1
MOV <dest>, A	<dest>= A	X	X	X		1
MOV <dest>,<src>	<dest>=<src>	X	X	X	X	2
MOV DPTR,#data16	DPTR=16-bit constant				X	2
PUSH <src>	INC SP:MOV"@SP".<src>	X				2
POP <dest>	MOV <dest>,"@SP":DEC SP	X				2
XCH A, <byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A, @Ri	ACC and @Ri exchange low nibbles		X			1

Slika 4.1 Prenos podataka sa instrukcijama koje pristupaju internoj memoriji

ADRESS WIDTH	MNEMONIC	OPERATION	**EXECUTION TIME (ms)**
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @DPTR	2

Slika 4.2 Prenos podataka sa instrukcijama koje pristupaju spoljašnjoj memoriji

MNEMONIC	OPERATION	EXECUTION TIME (μs)
MOVC A,@A+DPTR	Read program memory at (A+DPTR)	2
MOVC A,@A+PC	Read program memory at (A+PC)	2

Slika 4.3 Instrukcije za čitanje lookup tabele

Kod instrukcija iz grupe prenosa podataka (MOV, MOVC i MOVX) zarez između operanda možemo da shvatimo kao strelicu nalevo, jer se uvek sadržina drugog operanda upisuje u prvi, pri čemu se drugi ne menja. Ovo ne važi za instrukcije XCH i XCHD, gde operandi međusobno izmenjuju vrednosti (kod XCHD samo četiri najniža bita).

Sve instrukcije iz ove grupe obavljaju prenos 8-bitnog podatka, sa dva izuzetka: XCHD vrši razmenu 4-bitnog podatka, a MOV DPTR, #data upisuje 16-bitnu vrednost u registre DPH i DPL.

Instrukcije čija svrha je upis bajta iz programske memorije ili komunikacija sa spoljnim RAM-om imaju 16-bitno adresno polje. Prvi slučaj (upis bajta iz programske memorije u akumulator) se uglavnom koristi za lookup tabele, gde se pozicija u tabeli izračunava iz bazne adrese početka tabele (obično se smešta u DPTR) i ofseta (koji je u akumulatoru), pa je zbog toga ova instrukcija unapred proširena i glasi MOVC A,@A+DPTR. Ona izvršava tačno ono što možemo i da pretpostavimo: najpre sabere 16-bitni DPTR sa 8-

bitnim akumulatorom, pa rezultat iskoristi za adresiranje programske memorije, pročita bajt i upiše ga u akumulator. Ako nam treba samo 16-bitno adrsiranje bez ofseta, pre ove instrukcije treba upisati 0 u akumulator (CLR A).
Instrukcija MOVC A, @A+PC radi isto to, ali umesto DPTR uzima 16-bitnu adresu sledeće instrukcije u programu. Sledeći potprogram uzima vrednost člana pod rednim brojem koji je upisan u akumulator (u ovom slučaju od 0 do 4), iz tabele koja sledi odmah iza potprograma:

```
REL_PC:   INC    A
          MOVC A, @A+PC
          RET
          DB   27h
          DB   41h
          DB   00h
          DB   5Eh
          DB   7Fh
```

Instrukcija INC A je potrebna da bi se u kalkulaciji preskočila instrukcija RET, koja u programu zauzima jedan bajt. Ako ovde ima još koda, onda broj njegovih bajtova treba sabrati sa akumulatorom.
Instrukcije koje komuniciraju sa spoljnim RAM-om (MOVX) mogu svoje 16-bitno adresno polje da uzmu iz DPTR, ili kombinovanjem P2 (visoki bajt) i R0 ili Rl (niski bajt).
Nijedna od instrukcija iz ove grupe ne utiče na flegove, izuzev POP i MOV instrukcija kojima je prvi operand PSW, jer njihova funkcija i jeste izmena sadržaja registra koji sadrži flegove.

4.2 Aritmetičke instrukcije

MNEMONIC	OPERATION	ADRESSING MODES				EXECUTION TIME (μs)
		DIR	IND	REG	IMM	
ADD A,<byte>	A=A+<byte>	X	X	X	X	1
ADDC A,<byte>	A=A+<byte>+C	X	X	X	X	1
SUBB A,<byte>	A=A-<byte>-C	X	X	X	X	1
INC A	A=A+1					1
INC <byte>	<byte>=<byte>+1	X	X	X		1
INC DPTR	DPTR =DPTR+1					2
DEC A	A=A-1					1
DEC <byte>	<byte>=<byte>-1	X	X	X		1
MUL AB	B:A = B*A					4
DIV AB	A=Int[A/B] B=Mod[A/B]					4
DA A	Decimal Adjust					1

Mikrokontroler 8051 podržava sve četiri osnovne aritmetičke operacije. Samo osmobitna aritmetika je podržana (sa izuzetkom INC DPTR, koji uvećava stanje 16-bitnog registra), i to sa pozitivnim brojevima, mada overflow fleg omogućava sabiranje i oduzimanje sa predznakom. Sabiranje može da se obavlja i sa binarno kodiranim decimalnim (BCD) brojevima, kod kojih osmobitni registar nosi dve odvojene decimalne cifre, svaku u po 4 bita.

Sabiranje

INC (increment) uvećava operand za 1.
ADD sabira akumulator sa izvornim operandom i smešta rezultat u A. Bit C je deveti bit rezultata, zapravo bit prenosa za sabiranje višebajtnih brojeva.
ADDC (ADD with Carry) sabira akumulator sa izvornim operandom i sa bitom C, a rezultat smešta u akumulator.

Primer sabiranja višebajtnih binarnih brojeva:
Ako treba DPTR sabrati sa 16-bitnim binarnim brojem čiji je viši bajt u R4, a niži u R5 i rezultat smestiti u DPTR, program može da izgleda ovako:

```
MOV   A,DPL      ; uzmi niži bajt
ADD   A,R5       ; saberi
MOV   DPL,A      ; smesti rezultat nižeg bajta
```

```
MOV   A,DPH        ; uzmi viši bajt
ADDC  A,R4         ; saberi i dodaj prenos
MOV   DPH,A        ; smesti rezultat višeg bajta
```

DA A (Decimal Add Adjust for Addition) koriguje rezultat posle binarnog sabiranja, tako da odgovara za BCD brojeve. Na primer, ako akumulator i registar B sadrže BCD brojeve (00-99) onda će sledeći program sabrati ova dva BCD broja i ispravan BCD rezultat smestiti u akumulator:

```
ADD  A,B           ; saberi binarno
DA   A             ; koriguj rezultat
```

Ako je rezultat posle instrukcije DA A veći od 99, tada je bit C (prenos) setovan.

Oduzimanje

DEC (decrement) umanjuje operand za 1.
SUBB (subtract with borrow) oduzima drugi operand od prvog (koji je uvek akumulator). Pošto instrukcija SUB (oduzimanje na koje bit C ne utiče) ne postoji u setu za 8051, možemo da ga sintetišemo tako što pre SUBB izvedemo jedno CLR C (C=0).
Oduzimanje višebajtnih binarnih brojeva se vrši slično kao kod sabiranja. Od DPTR treba oduzeti 16-bitni binarni broj čiji je viši bajt u R4, a niži u R5 i rezultat smestiti u DPTR:

```
MOV   A,DPL        ; uzmi niži bajt
CLR   C            ; da bi SUBB radio kao SUB
SUBB  A,R5         ; oduzmi R5
MOV   DPL,A        ; smesti rezultat nižeg bajta
MOV   A,DPH        ; uzmi viši bajt
SUBB  A,R4         ; oduzmi R4 i prenos
MOV   DPH,A        ; smesti rezultat višeg bajta
```

Množenje

MUL AB množi dva 8-bitna broja bez predznaka, od kojih je jedan smešten u akumulator, a drugi u B registar, i 16-bitni rezultat smešta u akumulator (niži bajt) i registar B (viši bajt). Pošto je ovde rezultat 16-bitni, prekoračenje nije moguće pa je bit C uvek resetovan, a bit OV je setovan ako je rezultat veći od 255 (odnosno ako je registar B veći od 0), a resetovan ako je B=0.

Deljenje

DIV AB deli akumulator sa registrom B i celobrojni rezultat smešta u A, a ostatak deljenja u B. Deljenje sa nulom (ako je B=0) rezultira sa nepredvidivim sadržajem registara, i u tom slučaju bit OV će biti setovan. Inače, u normalnom slučaju (kad je B>0) bitovi OV i C su resetovani.

Logičke instrukcije

MNEMONIC	OPERATION	ADRESSING MODES				EXECUTION TIME (μs)
		DIR	IND	REG	IMM	
ANL A,<byte>	A=A AND <byte>	X	X	X	X	1
ANL <byte>,A	<byte>=<byte> AND A	X				1
ANL <byte>,#data	<byte>=<byte> AND #data	X				2
ORL A, <byte>	A=A OR <byte>	X	X	X	X	1
ORL <byte>,A	<byte>=<byte> OR A	X				1
ORL <byte>,#data	<byte>=<byte> OR #data	X				2
XRL A,<byte>	A=A XOR <byte>	X	X	X	X	1
XRL <byte>,A	<byte>=<byte> XOR A	X				1
XRL <byte>,#data	<byte>=<byte> XOR #data	X				2
CLR A	A=00H					1
CPL A	A= not A					1
RL A	Rotate A left 1 bit					1
RLC A	Rotate left through Carry					1
RR A	Rotate A right through Carry					1
RRC A	Rotate right through Carry					1
SWAP A	Swap nibbles in A					1

MNEMONIC	OPERATION	EXECUTION TIME (μs)
ANL C,bit	C = C AND bit	2
ANL C,/bit	C = C AND NOT bit	2
ORL C,bit	C = C or bit	2
ORL C,/bit	C = C or NOT bit	2
MOV C,bit	C = bit	1
MOV bit,C	bit = C	2
CLR C	C = 0	1
CLR bit	Bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = NOT C	1
CPL bit	Bit = NOT bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit,rel	Jump if bit = 1	2
JNB bit,rel	Jump if bit = 0	2
JBC bit,rel	Jump if bit = 1;CLR bit	2

CLR resetuje navedeni bit, a CLR A resetuje sve bitove u akumulatoru.
CPL komplementira navedeni bit (menja mu stanje), a CPL A komplementira sve bitove u akumulatoru.
SETB setuje navedeni bit.
RL (RR) rotira nalevo (nadesno) akumulator. Bit 7 (bit 0) dolazi na mesto bita 0 (bita 7). Bit koji izlazi sa krajnje pozicije i ulazi sa druge strane.
RLC (RRC) rotira nalevo (nadesno) akumulator kroz bit C (9-bitna rotacija). Tako će bit C da se preseli u bit 0 (bit 7), a bit 7 (bit 0) u bit C.
SWAP rotira akumulator 4 puta nalevo (odnosno na desno), tj. vrši zamenu nižeg i višeg nibla akumulatora.
ANL izvodi logičku operaciju AND između prvog i drugog operanda, a rezultat smešta u prvi operand. Ova operacija je moguća sa operandima koji zauzimaju jedan bajt ili jedan bit.
ORL izvodi logičku operaciju OR između prvog i drugog operanda, a rezultat smešta u prvi operand. Ova operacija je moguća sa operandima koji zauzimaju jedan bajt ili jedan bit.
XRL izvodi logičku operaciju XOR između prvog i drugog operanda (oba su 8-bitna), a rezultat smešta u prvi operand.

4.3 Potprogrami

ACALL (Absolute Call) i LCALL (Long Call) pozivaju potprograme na sledeći način: najpre 16-bitnu adresu sledeće instrukcije u programu stave na stek i uvećaju SP za dva, a onda u PC upišu pridruženu adresu. Ta adresa u slučaju poziva ACALL može da se nađe samo u istom bloku od 2K u kome je i prva instrukcija posle poziva (ona koja se stavlja na stek), a kod LCALL nema ograničenja, jer je adresa 16-bitna. Razlog za ograničenje kod ACALL je taj što ova instrukcija zauzima samo 2 bajta (prema 3 bajta koje zauzima LCALL), i 5 bitova zauzima kod instrukcije, pa tako za adresno polje ostaje samo 11 bitova.
RET izaziva akciju koja je inverzna instrukciji ACALL odnosno LCALL: sa steka se uzimaju dva bajta i smeštaju u programski brojač. SP se umanjuje za dva. To je povratak iz potprograma, jer će posle ovoga program nastaviti da se izvršava od instrukcije koja je sledila iza poziva.
RETI radi isto što i RET, samo što se uz to omogućava i prekid tekućeg nivoa prioriteta. Ovo je standardna instrukcija za povratak iz prekidnog potprograma.

4.4 Bezuslovni skokovi

MNEMONIC	OPERATION	EXECUTION TIME (μs)
JMP addr (*)	Jump to addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2

RETI	Return from interrupt	2
NOP	No operation	1

(*) Pod naredbom JMP se podrazumeva jedna od tri naredbe skoka, AJMP, LJMP i SJMP.

AJMP (Absolute jump) i LJMP (Long Jump) prenose pridruženu adresu u programski brojač, tako da će program nastaviti da se izvršava nadalje od te adrese. AJMP i LJMP su po broju bitova adrese analogni pozivima potprograma ACALL i LCALL.

SJMP (Short jump) je dvobajtna instrukcija koja omogućava skokove u opsegu -128 do +127 u odnosu na sledeću instrukciju u programu. Sem dužine skoka, razlika između SJMP i AJMP je u tome što SJMP može da preskoči blok od 2KB, dok AJMP uvek skače unutar ovakvog bloka.
U tabeli, kod instrukcije "JMP addr", reč JMP se odnosi na sve tri instrukcije skoka, AJMP, LJMP i SJMP.

JMP @A+DPTR skače na adresu koja se izračunava tako što se sabere 16-bitni registar DPTR sa ofsetom koji se nalazi u akumulatoru. Ovo je od koristi kod složenih grananja u potprogramu, kad od stanja akumulatora zavisi na koju adresu treba izvesti skok (tabelarni skokovi). Sledeći program će izvesti skok na jednu od lokacija iz tabele JP_TAB, zavisno od stanja akumulatora na ulazu:

```
                MOV  B,3            ;  priprema za  množenje
                MUL  AB             ; A = A*3, jer je dužina LJMP instrukcije 3 bajta
                MOV DPTR,#JP_TAB
                JMP  @A+DPTR        ; skok na željeno mesto
JP_TAB:         LJMP SERVICE_A
                LJMP SERVICE_B
                LJMP SERVICE_C
                LJMP SERVICE_D
```

Svaka instrukcija u tabeli mora da zauzme po 3 bajta, a posebnu pažnju treba obratiti da akumulator na ulazu nema vrednost veću od broja članova tabele umanjenog za 1 (u ovom primeru A mora da bude u opsegu od 0 do 3, jer ima 4 instrukcije skoka).

4.5 Uslovni skokovi

MNEMONIC	OPERATION	ADRESSING MODES				EXECUTION TIME (μs)
		DIR	IND	REG	IMM	
JZ rel	Jump if A = 0					2
JNZ rel	Jump if A ≠ 0					2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A, <byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

Svi uslovni skokovi su mogući samo u opsegu -128 do +127 u odnosu na sledeću instrukciju u programu, kao što je slučaj kod SJMP. Ako je potrebno skočiti na neku dalju tačku, treba blizu uslovnog skoka postaviti repetitor, koji se sastoji od instrukcije AJMP ili LJMP, i onda izvesti kratki uslovni skok na repetitor. Druga mogućnost je da se komplementira uslov i preskoči repetitor, kao u ovom primeru, koji zamenjuje skok JZ FAR_LAB:

```
instrukcija
        JZ      FAR_LAB
se zamenjuje sekvencom
        JNZ     LAB1
        LJMP  FAR_LAB
  LAB1:
```

Bitovi koji mogu da se iskoriste za uslovne skokove:

JZ (Jump on Zero) izvršava skok ako je akumulator jednak nuli
JNZ (Jump on Not Zero) izvršava skok ako je akumulator različit od nule
JC (Jump on Carry) izvršava skok ako je flag C setovan
JNC (Jump on Not Carry) izvršava skok ako je fleg C resetovan
JB (Jump on Bit) izvršava skok ako je navedeni bit setovan
JNB (jump on Not Bit) izvršava skok ako je navedeni bit resetovan
JBC (Jump on Bit and Clear bit) izvršava skok ako je navedeni bit setovan i istovremeno resetuje navedeni bit
CJNE (Compare and jump if Not Equal) poredi prvi operand sa drugim i skače ako su različiti. Bit C će biti setovan ako je prvi operand manji od drugog, u suprotnom biće resetovan.
DJNZ (Decrement and jump if Not Zero) umanjuje operand za 1, a ako je posle umanjenja operand različit od nule, skače na navedenu adresu, dok u suprotnom ignoriše skok i nastavlja izvršenje programa od sledeće instrukcije (slično instrukciji LOOP kod mikroprocesora 8086).

4.6 Uticaj instrukcija na flegove

U toku izrade programa od velike je važnosti znati na koji način instrukcije utiču na flegove u PSW registru. Neke instrukcije uopšte ne utiču na flegove C (carry), OV (overflow) i AC (auxiliary carry), pa te instrukcije nisu pomenute u tabeli.

	C	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	-
DIV	0	X	-
DA	X	-	-
RRC	X	-	-
RLC	X	-	-
SETB C	1	-	-
CLR C	0	-	-
CPL C	X	-	-
ANL C,bit	X	-	-
ANL C,/bit	X	-	-
ORL C,bit	X	-	-
ORL C,/bit	X	-	-
MOV C,bit	X	-	-
CJNE	X	-	-

Legenda:

- instrukcija ne utiče na fleg
0 instrukcija resetuje fleg
1 instrukcija setuje fleg
X fleg se menja zavisno od rezultata operacije
Napomena: instrukcije koje direktno menjaju sadržaj registra PSW će uticati na flegove, bez obzira što ovde nisu navedene.
Preostala dva flega: P (parnost) i Z (zero) nisu fizički implementirani u PSW registru kao flip-flopovi, nego su zapravo grupe logičkih kola koje testiraju stanje akumulatora. Tako će izlaz iz logike za fleg P biti visok ako je broj setovanih bita u akumulatoru paran, a za fleg Z ako su svi biti u akumulatoru jednaki nuli. Prema tome, nema nikakvog smisla navoditi kako instrukcije utiču na ove flegove, jer to zavisi samo od trenutnog stanja akumulatora.

5. TAJMERI/BROJAČI

5.1 Tajmeri 0 i 1

Mikrokontroler 8051 sadrži dva šesnaestobitna tajmersko-brojačka registra. Označavaju se kao tajmer 0 i tajmer 1. Mogu služiti kao brojački registri, ili tajmeri koji mere zadati vremenski interval. Oba tajmera mogu raditi u četiri radna režima.

Kada radi kao brojač, sadržaj tajmerskog registra se uvećava za jedan na svaku silaznu ivicu odgovarajućeg ulaza mikrokontrolera (T0 za tajmer 0 i T1 za tajmer 1). Maksimalna frekvencija ulaznog signala koju brojač može da prati je 24 puta manja od radnog takta mikrokontrolera.

Kad radi kao tajmer, sadržaj tajmerskog registra se uvećava za jedan u svakom mašinskom ciklusu. Jedan mašinski ciklus traje 12 perioda radnog takta mikrokontrolera. (Ako je radni takt 12MHz, registar se uvećava svake mikrosekunde.)

(MSB) **(LSB)**

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
◄ TAJMER 1 ►				◄ TAJMER 0 ►			

GATE kontrola gejta. Kada je ovaj bit na jedinici, odgovarajući tajmer može da broji kada je pripadajući TR bit jedinica i pripadajući INT ulaz mikrokontrolera vosok. Ako je vrednost GATE bita 0, dovoljno je da samo odgovarajući bit TR bude visok

C/$\overline{T}$ ovaj bit određuje da li će odgovarajući tajmer da radi kao brojač opadajućih ivica na odgovarajućem T ulazu (C/T=1) ili kao tajmer (C/T=0)

M1	M0	
0	0	Mod 0. tajmer radi kao 13-bitni brojač
0	1	Mod 1. tajmer radi kao 16-bitni brojač
1	0	Mod 2. tajmer radi kao osmobitni brojač. Brojački registar je TL. nakon preticanja, TL se puni sadržajem TH registra (auto reload)
1	1	Mod 3, Tajmer 1 je u ovom modu zaustavljen, a tajmer 0 radi kao dva odvojena osmobitna tajmera. Za tajmer 1 ovaj mod se ne koristi (zabranjeno je).

Slika 5.1 Registar TMOD

Svakom tajmeru/brojaču pripadaju po dva osmobitna registra, TH0 i TL0 tajmeru 0, a TH1 i TL1 tajmeru 1.

5.1.1 Kontrola rada tajmera 0 i 1

Radom tajmera upravlja se pomoću registara TMOD (slika 5.1) i TCON (slika 5.2). Gornja četiri bita registra TMOD kontrolišu rad tajmera 1, a donja četiri rad tajmera 0. Bitovi TR1 i TR0 registra TCON određuju da li će odgovarajući tajmer/brojač biti aktivan ili ne. Npr. ako je brojač 0 neaktivan on se neće uvećavati kada na ulazu T0 nastupi prelaz sa visokog na nizak nivo. Ovo je ilustrovano na slikama 5.3, 5.4 i 5.5. Način rada tajmera bira se kombinacijom bitova M1 i M0 registra TMOD.

(MSB)							(LSB)
TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

TF1 zastavica preteka tajmera 1. Automatski se briše kada se skoči na potprogram za opsluživanje prekida

TR1 postavljanjem ovog bita tajmer 1 počinje da broji. U suprotnom ne broji

TF0 zastavica preteka tajmera 0. Automatski se briše kada se skoči na potprogram za opsluživanje prekida

TR0 postavljanjem ovog bita tajmer 0 počinje da broji. U suprotnom ne broji

IE1 zastavica detektovanog spoljašnjeg zahteva za prekidom 1. Kada je zahtev detektovan, postavlja se na jedinicu, a briše se automatski kada se skoči na potprogram za opsluživanje prekida

IT1 određuje da li je ulaz za spoljašnji prekid 1 osetljiv na opadajuću ivicu ili na nizak nivo

IT0 zastavica detektovanog spoljašnjeg zahteva za prekidom 0. Kada je zahtev detektovan, postavlja se na jedinicu, a briše se automatski kada se skoči na potprogram za opsluživanje prekida

IE0 određuje da li je ulaz za spoljašnji prekid 0 osetljiv na opadajuću ivicu ili na nizak nivo

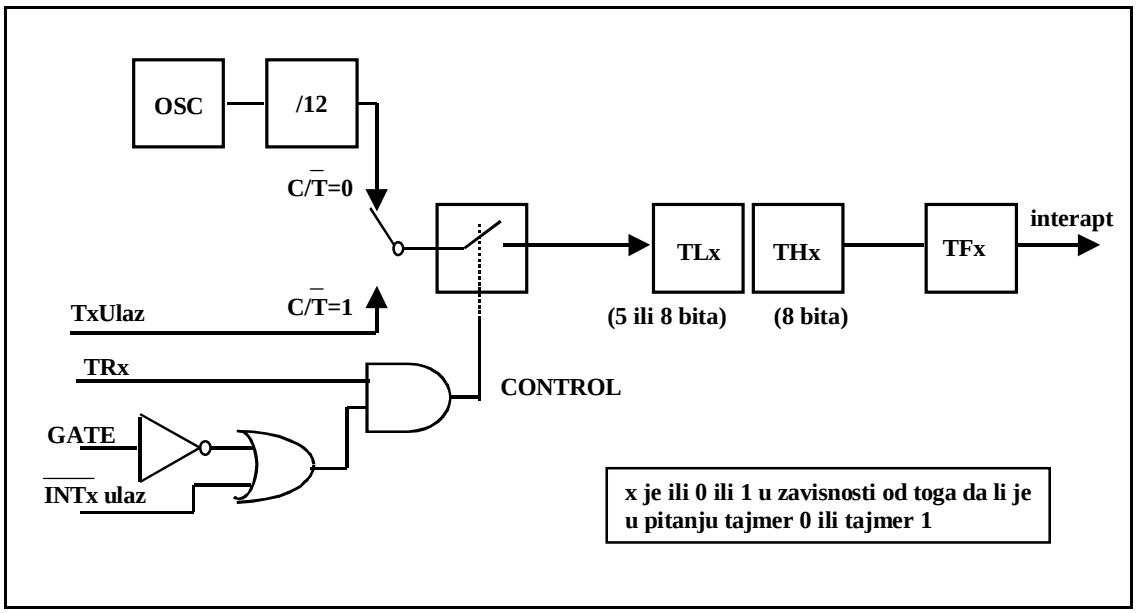
Slika 5.2 Registar TCON

Biti C/T registra TMOD (u viša ili niža četiri bita u zavisnosti od tajmera) definišu da li odgovarajući tajmer radi kao tajmer (C/T=0) ili kao brojač (C/T=1). Biti GATE svojim stanjem 0 omogućuju odgovarajućem tajmeru da radi uvek kada je odgovarajući TR bit jednak jedinici. Ako je GATE=1, rad tajmera uslovljen je i stanjem na ulazu INT1 odnosno INT0 u zavisnosti od tajmera. Odgovarajući INT ulaz tada mora biti jedinica da bi tajmer radio.
Objašnjenja koja slede važe za oba tajmera osim tamo gde je drugačije napomenuto.

5.1.2 Modovi rada tajmera 0 i 1

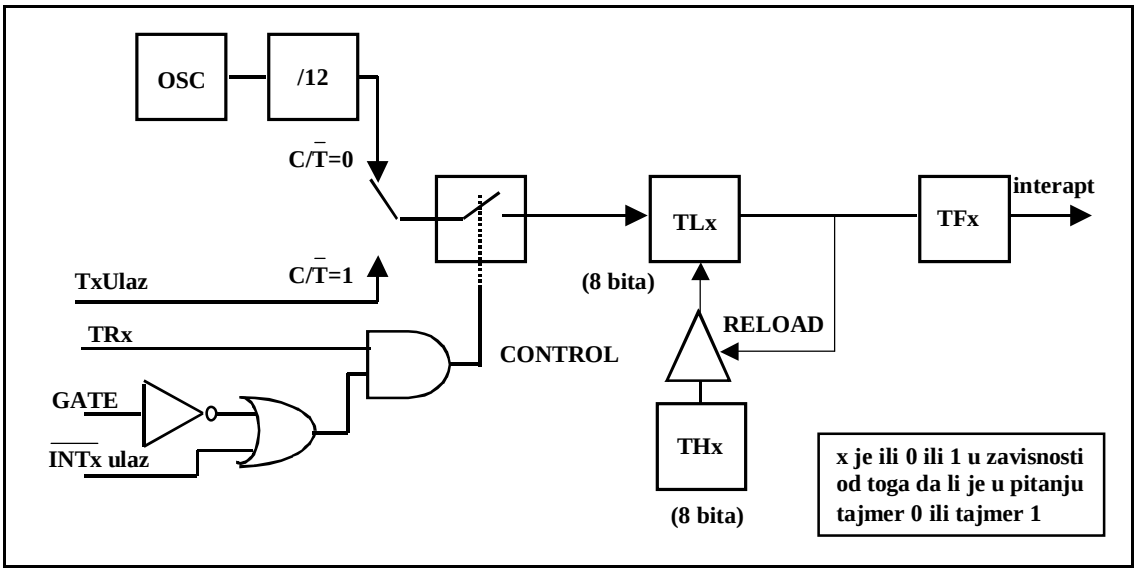
Mod 0

Ovaj mod je ilustrovan na slici 5.3. U ovom modu tajmer predstavlja 13-bitni tajmer/brojač. Kada svi biti registra postanu jedinice, sledećim uvećanjem sadržaja registra svi biti postaju nule. To se zove preticanje tajmera/brojača. Preticanje izaziva postavljanje odgovarajuće interapt zastavice (bit TF1 registra TCON za tajmer 1 i bit TF0 za tajmer 0). Postavljanje interapt zastavice izaziva prelazak na izvršavanje servisne rutine ako je odgovarajući interapt dozvoljen. (Biti TF1 i TF0 se automatski vraćaju na vrednost 0 kada se desi skok na interapt servisnu rutinu).



Slika 5.3 Mod 0 i 1 Tajmera 0 ili 1

Pomenuti 13-bitni registar sastoji se od svih osam bita TH i nižih pet bita TL odgovarajućeg tajmer registra. Viša tri bita TL registra u ovom modu su nedefinisana. Izbor moda 0 vrši se postavljanjem bita M1 i M0 u registru TMOD na 0.



Slika 5.4 Mod 2 Tajmera 0 ili 1

Mod 1

Ovaj mod je po svemu isti kao i mod 0 s tim što je brojački registar u ovom slučaju 16-bitni. Koristi se svih osam bitova i TL i TH registra. Bit TF1 ili TF0 se opet postavlja pri prelasku iz stanja svih šesnaest bita jednako 1 u stanje svih šesnaest bita jednako 0.

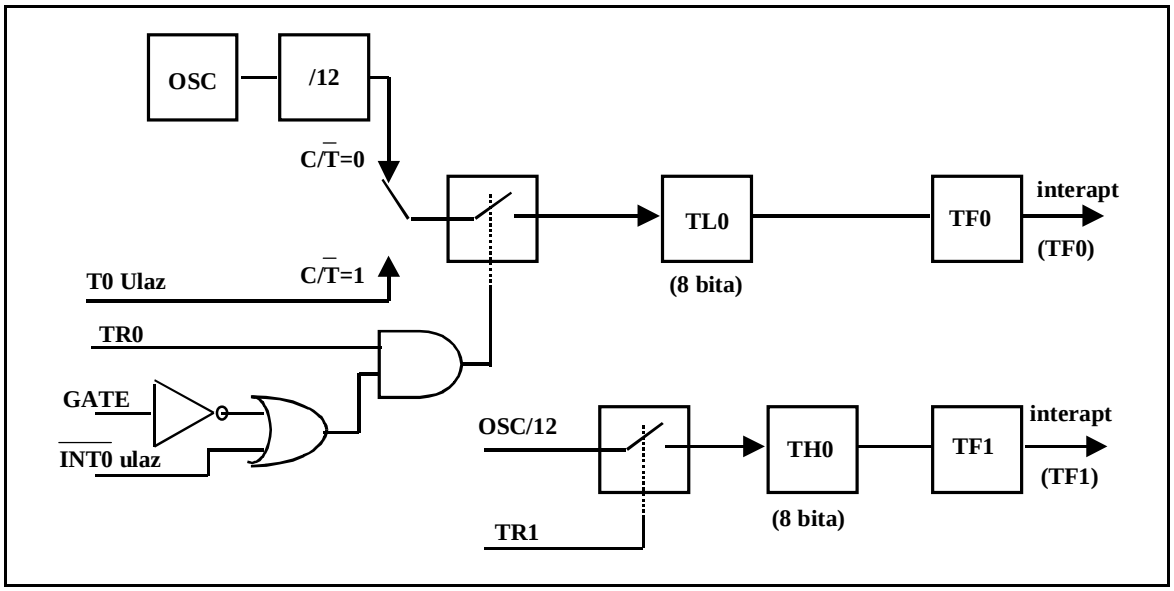
Mod 2

U ovom modu tajmer radi kao osmobitni tajmer sa automatskim punjenjem inicijalne vrednosti nakon preticanja. Rad u modu 2 ilustrovan je na slici 5.4. Kao tajmer/brojač uvećava se odgovarajući TL registar, a kada dođe do preticanja postavlja se bit TF1 ili TF0 u zavisnosti od tajmera, dok se vrednost iz odgovarajućeg TH registra upisuje u TL. Vrednost TH registra ostaje neizmenjena.

Mod 3

Ovaj mod različito prihvataju tajmer 0 i tajmer 1. Kada se tajmer 1 prebaci u mod 3 on jednostavno prestane da broji zadržavajući svoje prethodno stanje, kao da je bit TR1=0. Ako je tajmer 0 u modu 3, a tajmer 1 u bilo kom modu osim moda 3, tajmer 1 nastavlja svoj regularan rad u datom modu samo što ne može da generiše interapt jer je bit TF1 upravljan tajmerom 0

Tajmer 0 u modu 3 radi kao dva odvojena osmobitna tajmera. Bitovi GATE, TR0 i $C/\overline{T}$ (i ulazi T0 i INT0) kontrolišu brojanje registra TL0 čijim preticanjem se postavlja bit zastavica TF0. Registar TH0 broji mašinske cikluse, a njegovim preticanjem postavlja se bit zastavica TF1. Bit TR1 kontroliše brojanje TH0. Dakle, tajmerom 0 je kontrolisano izazivanje interapta i tajmera 0 i tajmera 1.



Slika 5.5 Mod 3 Tajmera 0

5.2 Tajmer 2

Tajmer 2 je 16 – bitni tajmer/brojač koji može da radi u tri moda: capture, auto-reload i kao baud rate generator. Tajmer 2 sadrži dva 8 – bitna registra TH2 i TL2, koji uvek rade kao jedinstveni 16-bitni brojač. Sem toga postoje još dva dodatna registra, RCAP2H i RCAP2L koji se takođe ponašaju kao 16-bitni registar (RCAP2). Kada je u funkciji tajmera, TH2:TL2 registar se uvećava svakog mašinskog ciklusa. Pošto se mašinski ciklus sastoji od 12 perioda oscilatora, brzina brojanja je 1/12 frekvencije oscilatora.

5.2.1 Kontrola rada tajmera 2

Upravljački registar T2CON prikazan je na slici 5.6

7	6	5	4	3	2	1	0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

Slika 5.6 Registar T2CON

TF2: Setuje se kod prekoračenja brojača (promena vrednosti sa 0FFFFh na 0000h) i **mora** da se resetuje softverski. Ovaj bit neće biti setovan ako se tajmer 2 koristi u modu *boud rate* generatora, tj. ako je RCLK=1 ili TCLK=1.

EXF2: Setuje se kad negativna ivica na T2EX nožici izazove prepis stanja u RCAP2 u tajmer 2, ili iz tajmera 2 u RCAP2 (zavisno od moda u kom tajmer 2 radi, konkretno od bita CP/$\overline{RL2}$). Kad je prekid tajmera 2 omogućen (setovani biti ET2 i EA u registru IE), EXF2 će (ako je setovan) uzrokovati prekid. Ovaj bit se, kao i TF2, ne resetuje automatski već se **mora** resetovati softverski.

RCLK: Setovanje ovog bita prebacuje tajmer 2 u režim *baud rate* generatora za prijemnu sekciju serijskog porta.

TCLK: Kao i kod RCLK bita, ali za predajnu sekciju serijskog porta. To omogućava da se generišu različite učestanosti za prijem (RXD) i predaju (TXD), jer se za jedan od od ova dva smera može generisati učestanost pomoću tajmera 1, a za drugi (kome je setovan RXD ili TXD), pomoću tajmera 2. Ako su učestanosti za prijem i predaju jednake mogu da se setuju i RCLK i TCLK. U režimu *baud rate* generatora ulazni takt tajmera 2 je 1/2 a ne 1/12 frekvencije oscilatora.

EXEN2: Ovaj bit, ako je setovan i ako se tajmer 2 ne koristi u *boud rate* generatorskom modu (RCLK=0 i TCLK=0), tada se na opadajuću ivicu signala na nožici T2EX izaziva prepis sadržaja iz tajmera 2 u RCAP2 ako je $CP/\overline{RL2}$ setovan (*capture* mod), a prepis iz RCAP2 u tajmer 2 ako je taj bit resetovan (*auto-reload* režim).

TR2: Ovo je start/stop bit za tajmer 2. Ako je TR2=1, tajmer je startovan.

$C/\overline{T2}$**:** Kada je setovan, tajmer 2 radi kao brojač opadajućih ivica na nožici T2EX (takt se dovodi spolja), a ako je resetovan, tajmer 2 se taktuje internim taktom, čija je frekvencija 1/12 učestanosti oscilatora.

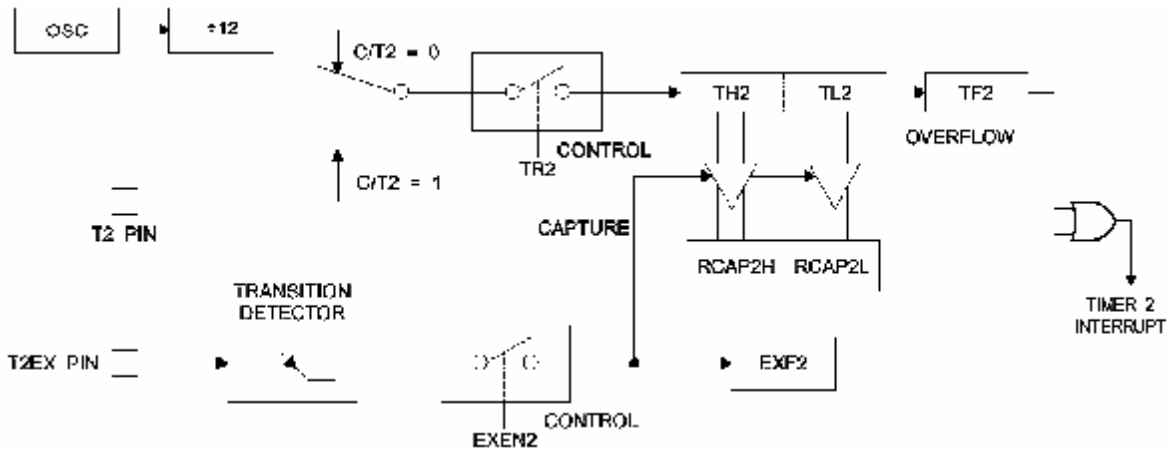
$CP/\overline{RL2}$**:** Ovo je *capture/auto-reload* selekcija. Kada je ovaj bit setovan, tajmer 2 je u *capture* modu (na silaznu ivicu signala na nožici T2EX stanje tajmera 2 se prebacuje u RCAP2 registar). Kada je ovaj bit resetovan, onda je u *auto-reload* modu (tajmer 2 se puni iz registra RCAP2 na prelazu iz stanja 0FFFFh u 0000h). Ako je setovan RCLK ili TCLK, važeći mod je *baud rate* generator i stanje ovog bita se ignoriše.

5.2.2 Modovi rada tajmera 2

U sledećoj tabeli prikazani su mogući modovi tajmera 2.

RCLK+TCLK	$CP/\overline{RL2}$	TR2	MOD
0	0	1	16-bit Auto-Reload
0	1	1	16-bit Capture
1	X	1	Baud Rate generator
X	X	0	Off

Slika 5.7 prikazuje tajmer 2 u ***capture modu***. Ako je EXEN2=0, tajmer 2 je 16-bitni tajmer ili brojač, čije prekoračenje setuje bit TF2 u registru T2CON. Ovaj bit se može koristiti za generisanje prekida. Ako je EXEN2=1, tajmer 2 i dalje radi na isti način, a na svaku opadajuću ivicu signala na ulazu T2EX, u registar RCAP2 se upisuje trenutno stanje tajmera 2, pri čemu se istovremeno setuje bit EXF2 u registru T2CON. Bit EXF2, kao i TF2, može generisati prekid.



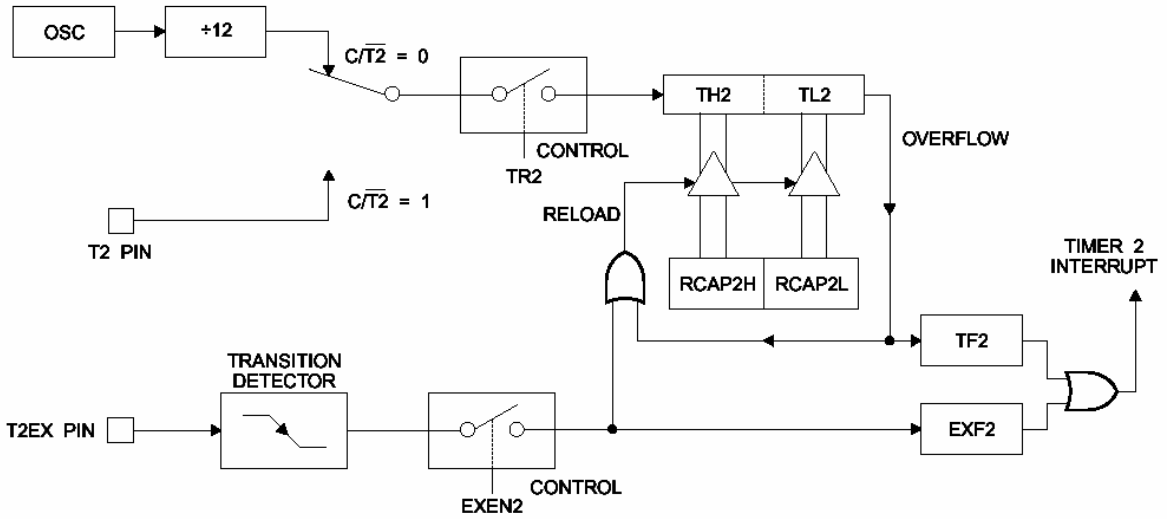
Slika 5.7 Tajmer 2 u Capture modu

U ***auto-reload modu*** tajmer 2 može biti programiran da broji na gore/dole kada je konfigurisan u 16-bitnom auto-reload modu. Ta opcija se bira pomoću DCEN (Down Counter Enable) bita, koji se nalazi u registru **T2MOD** (ovaj registar ne postoji u standardnom 8052) , koji je prikazan u sledećoj tabeli.

7	6	5	4	3	2	1	0
-	-	-	-	-	-	T2OE	DCEN

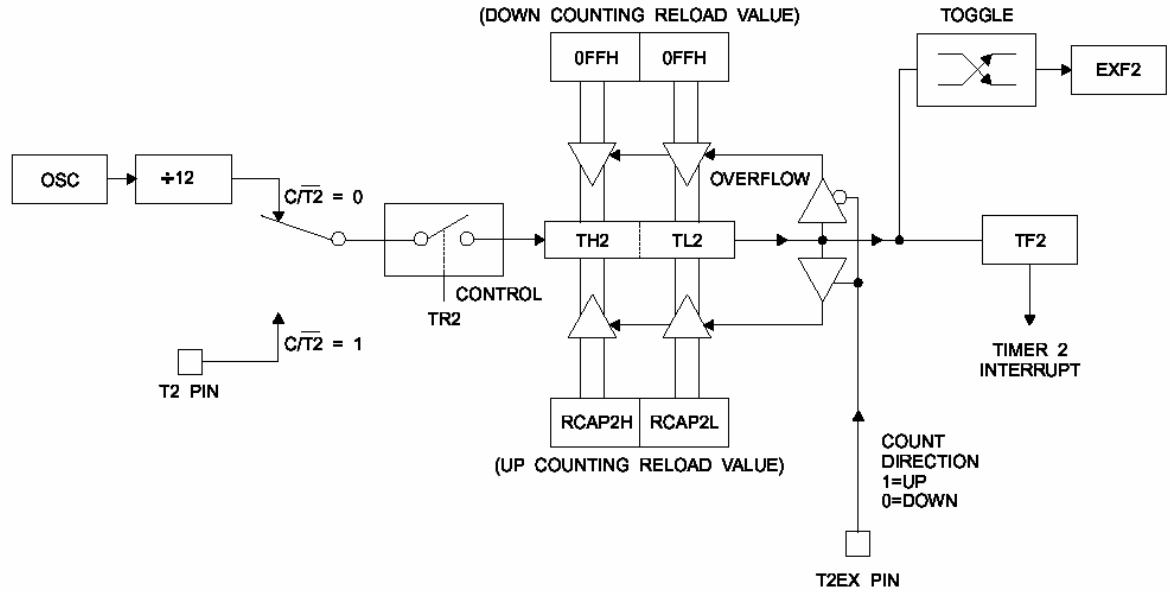
Simbol	Funkcija
-	Bit nije implementiran, rezervisan je za buduću nadogradnju
T2OE	Tajmer 2 Output Enable bit
DCEN	Kada je DCEN=1 tajmer 2 je konfigurisan kao brojač na gore/dole

Nakon reseta, DCEN je postavljen na nulu pa tajmer 2 inicijalno broji na gore. Kada se bit DCEN setuje, tajmer 2 može da broji na gore ili na dole, u zavisnosti od vrednosti signala na nožici T2EX.
Slika 5.8 prikazuje tajmer 2 kada je DCEN=0 (ovo je standardni način za 8052). U tom modu postoje dve opcije u zavisnosti od bita EXEN2 u registru T2CON. Ako je EXEN2=0, tajmer 2 broji na gore do 0FFFFh, a na sledeći takt setuje bit prekoračenja TF2. Prekoračenje takođe prouzrokuje da registri tajmera 2 (TH2 i TL2) dobiju vrednost registra RCAP2 (16-bitni reload). Vrednost registra RCAP2 (RCAP2H i RCAP2L) se postavljaju softverski. Ako je EXEN2=1, 16-bitni reload će se desiti ili prilikom prekoračenja, ili na silaznu ivicu signala na nožici T2EX. Ta silazna ivica signala takođe setuje EXF2 bit.



Slika 5.8 Tajmer 2 u Auto-reload modu (DCEN=0)

Na slici 5.9 prikazan je slučaj kada je DCEN=1 (ova konfiguracija postoji samo kod AT89C52, a ne i u standardnom 8052) tako da tajmer 2 može da broji i na gore i na dole, zavisno od stanja signala na nožici T2EX (1-broji na gore, 0-broji na gore). U slučaju brojanja na gore, tajmer 2 se ponaša kao u standardnom *auto-reload* režimu (tajmer se puni iz RCAP2 na prelazu iz 0FFFFh u 0). Ako tajmer broji na dole, tada se, u momentu izjednačavanja sadržaja tajmera 2 i registra RCAP2, tajmer 2 puni sa 0FFFFh. U oba slučaja punjenja tajmera 2 setuje se i bit TF2. Pri tome, bit EXF2 ne izaziva prekid, a setuje se nakon svakog prekoračenja (može se koristiti kao 17. bit).

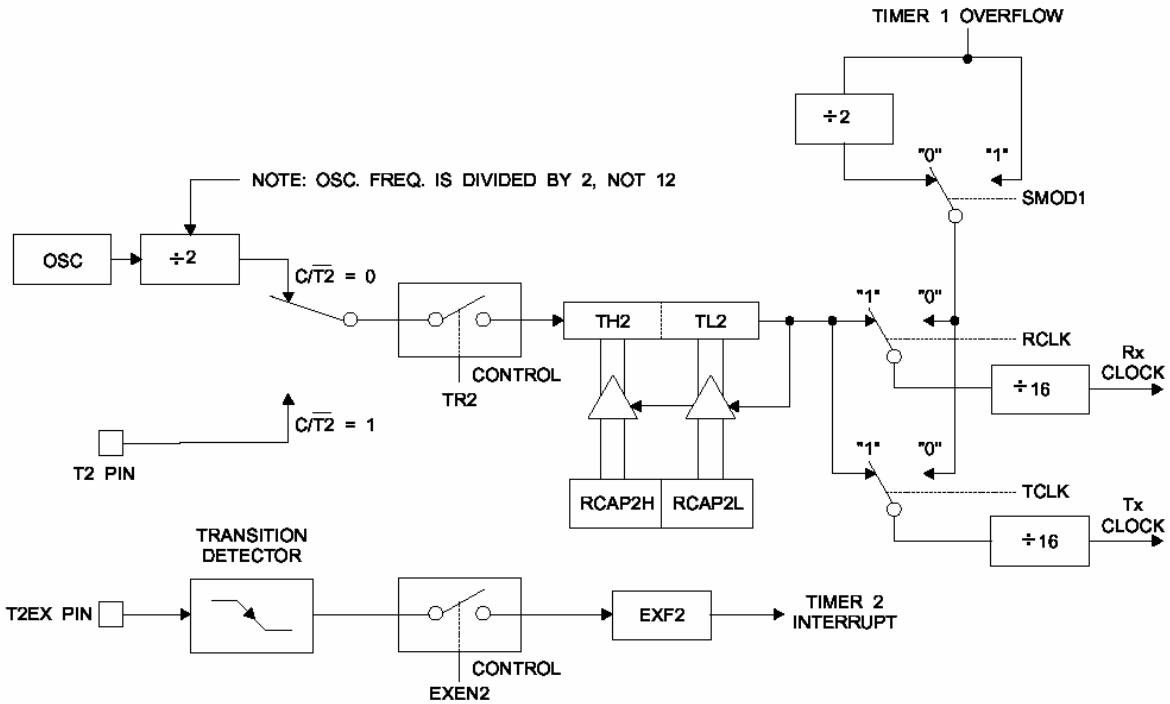


Slika 5.9 Tajmer 2 u Auto-reload modu (DCEN=1)

Slika 5.10 prikazuje tajmer 2 u ***baud rate generator modu***. Da bi tajmer 2 radio u tom modu potrebno je setovati bit TCLK i/ili RCLK u registru T2CON. Ovaj mod je sličan *auto-reload modu*, po tome što prekoračenje tajmera 2 uzrokuje njegovo punjenje sadržajem registra RCAP2. U ovom režimu tajmer 2 se taktuje sa 1/2 frekvencije oscilatora, pa se brzina komunikacije u modovima 1 i 3 određuje sledećom jednačinom:

$$Baud\ Rate = \frac{Timer2\_Overflow\_Rate}{16} = \frac{Oscilator\_Frekvency}{32*[65536-(RCAP2H, RCAP2L)]}$$

U ovom načinu rada tajmera 2 prekoračenje tajmera 2 ne setuje TF2, tako da ne prouzrokuje prekid. Ako je EXEN2=1, negativna ivica signala na nožici T2EX može da setuje EXF2, ali to ne utiče na rad tajmera2. Takođe, u ovom režimu se ne preporučuje upis u TH2, TL2, RCAP2H i RCAP2L dok tajmer 2 radi (TR2=1). Pre eventualnog upisa tajmer 2 treba zaustaviti (TR2=0), a tek nakon upisa se može ponovo startovati tajmer.

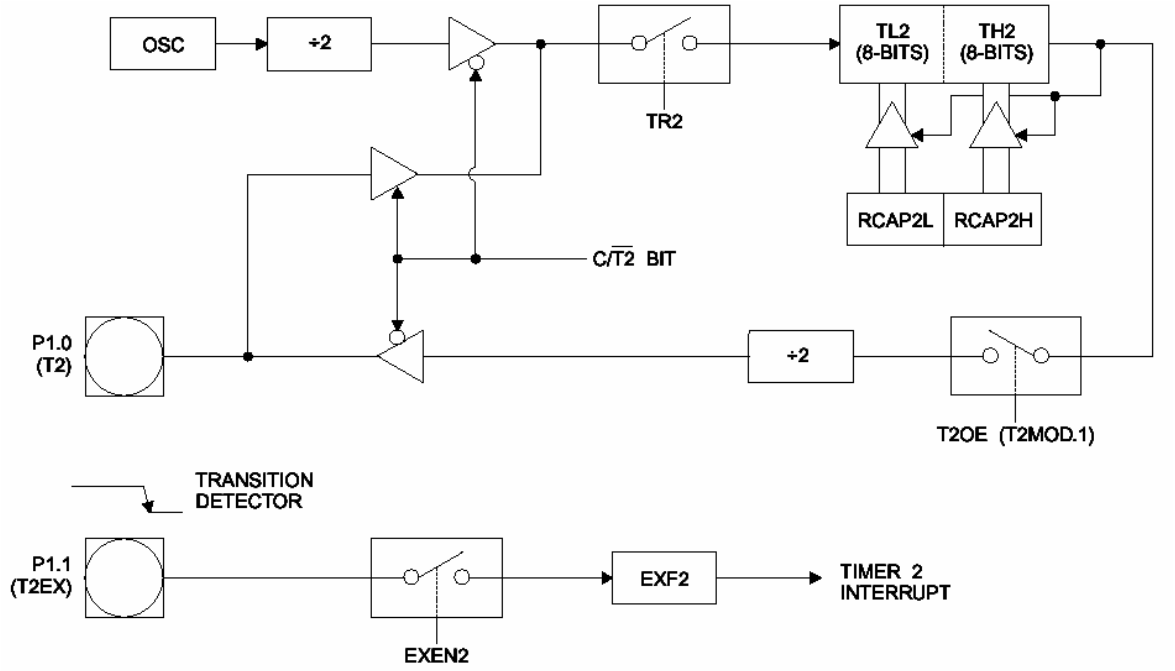


Slika 5.10 Tajmer 2 u Baud Rate Generator modu

Na slici 5.11 prikazano je dobijanje takta faktora ispune od 50% promenljive učestanosti na nožici P1.0 (važi samo za AT89C52). Sem već opisane funkcije, ova nožica se može koristiti kao izlaz takta, faktora ispune 50% i frekvencije u opsegu 1/(4*65536) do 1/4 frekvencije oscilatora. Da bi se konfigurisao tajmer/brojač 2 kao **generator takta** (clock generator), potrebno je da $C/\overline{T2}$=0 i T2OE=1. Bit TR2 startuje i zaustavlja tajmer. Frekvencija takta zavisi od frekvencije oscilatora i vrednosti sadržaja registra RCAP2, koji se upisuje u registre tajmera 2:

$$Clock-Out\_Frequency = \frac{Oscillator\_Frequency}{4*[65536-(RCAP2H, RCAP2L)]}$$

Kada se tajmer 2 koristi na ovaj način, on ne generiše prekid. Moguće je koristiti tajmer 2 kao *baud rate* generator i kao generator takta istovremeno. Međutim, brzina prenosa i frekvencija takta nisu međusobno nezavisne veličine pošto obe koriste RCAP2H i RCAP2L.



Slika 5.11 Tajmer 2 u Clock-out modu

6. SERIJSKI INTERFEJS

Serijski port je dupleksni, što znači da predaja i prijem mogu da teku istovremeno. Prijemni deo serijskog porta je dopunjen jednim paralelnim registrom, u koji se iz prijemnog pomeračkog registra automatski upisuje bajt koji je preko RxD nožice upravo pristigao na port. To omogućava da se odmah nastavi prijem sledećeg bajta. Pošto postoji samo jedan *hold* registar, ako se bajt ne pročita do kraja prijema sledećeg, tada se primljeni bajt nepovratno gubi. Zbog toga treba voditi računa o organizaciji softvera, tako da se u najgorem slučaju ne može dogoditi ova greška (*overrun*). Jedan od načina za izbegavanje ove greške je rad prijemnika pod prekidom.

SBUF (Serial Data Buffer) registar, čine dva fizički odvojena registra. U jednom se nalazi bajt koji je upravo stigao na serijski port, a drugi služi za upis bajta koji će biti poslat. Upis u predajni i čitanje iz prijemnog bafera se interno sinhronizuje pomoću dva *ready-busy handshaking bita*: **TI** bit se automatski setuje kada se predajni bafer isprazni, a **RI** bit kad se prijemni bafer napuni. Ova dva bita se softverski moraju resetovati.

6.1 Kontrola rada serijskog porta

Kontrola rada serijskog porta se obavlja pomoću registra **SCON,** čiji je sadržaj dat na sledećoj slici.

	7	6	5	4	3	2	1	0
SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 i **SM1** određuju mod u kom radi serijski port:

SM0	SM1	MOD	OPIS	BAUD RATE
0	0	0	shift registar	Fosc/12
0	1	1	8-bitni serijski port	promenljivo
1	0	2	9-bitni serijski port	Fosc/64 ili /32
1	1	3	9-bitni serijski port	promenljivo

SM2 omogućava multiprocesorsku komunikaciju u modovima 2 i 3. U ova dva moda, za SM2=1, RI se aktivira samo ako je primljeni deveti bit RB8=1. U modu 1, ako je SM2=1 tada se RI neće aktivirati ako nije primljen ispravan stop bit. U modu 0, SM2 treba uvek da bude 0.

REN omogućava (REN=1) ili zabranjuje (REN=0) serijski prijem. Postavlja se i briše softverski.

TB8 u modu 2 i 3 je deveti bit koji će biti poslat preko serijskog porta narednim upisom u SBUF. Postavlja se i briše softverski, a može se koristiti i kao bit parnosti.

RB8 u modu 2 i 3 je deveti bit koji je primljen preko serijskog porta (može se koristiti kao bit parnosti). U modu 1, ako je SM2=0 onda RB8 sadrži stop bit koji je primljen. U modu 0, RB8 se ne koristi.

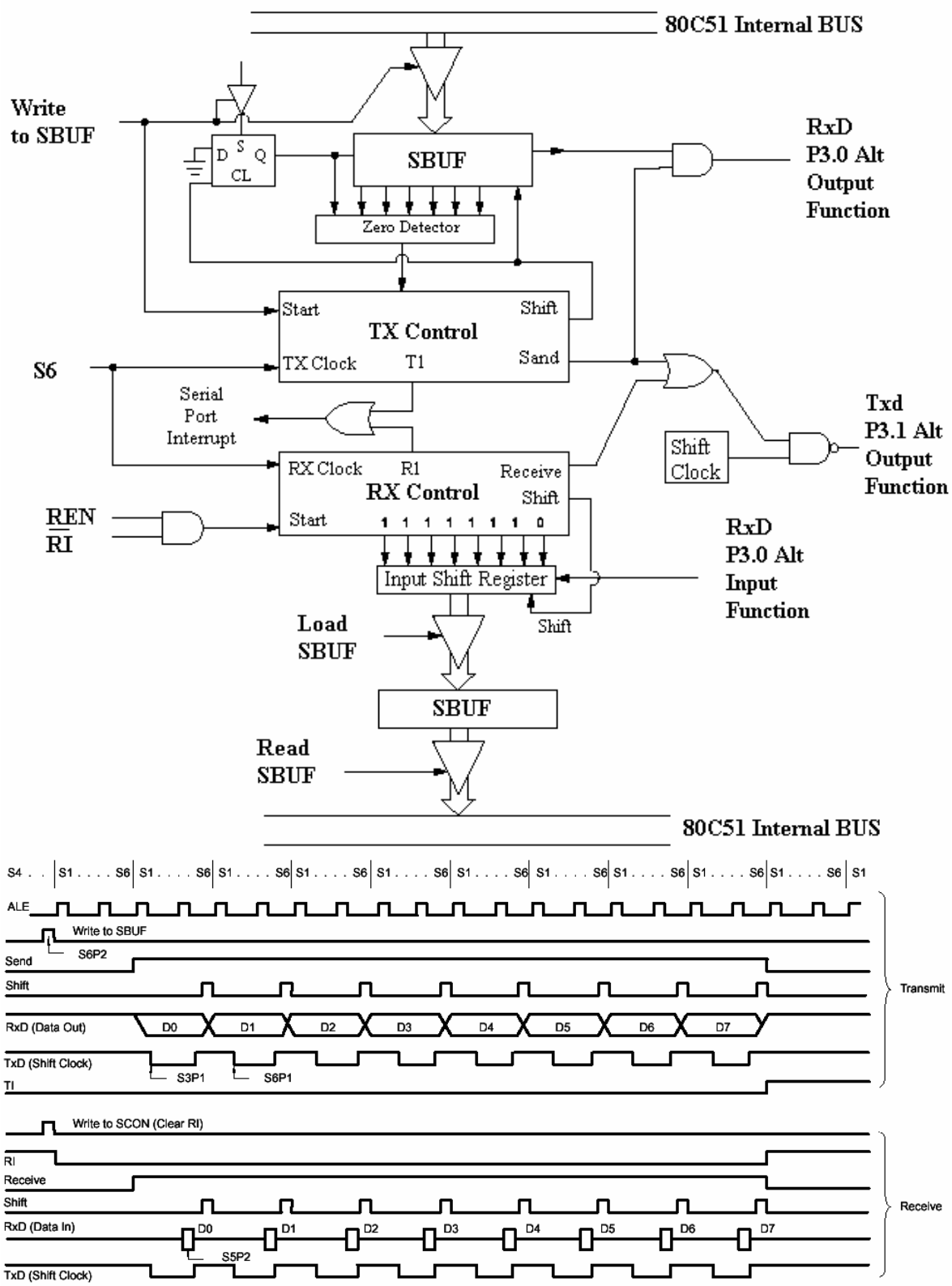
TI (Transmit Interrupt Flag) je bit koji hardver automatski setuje na kraju slanja osmog bita u modu 0 ili na kraju slanja stop bita u ostalim modovima. Mora se resetovati softverski.

RI (Receive interrupt Flag) je bit koji hardver automatski setuje na kraju prijema osmog bita u modu 0 ili na sredini stop bita u ostalim modovima, osim tokom multiprocesrske komunikacije kada nastupa izuzetak opisan kod bita SM2. Mora se resetovati softverski.

6.2 Modovi rada serijskog porta

Mod 0 : *brza sinhrona komunikacija sa pomeračkim registrima*

Ovaj mod nije namenjen za standardnu asinhronu komunikaciju, već za proširivanje ulaznih i izlaznih portova pomoću pomeračkih (shift) registara. Ulaz ili izlaz serijskih podataka od 8 bita je preko nožice RxD (P3.0) a izlaz takta je preko TxD (P3.1). Učestanost ovog takta je fiksna i iznosi 1/12 učestanosti oscilatora. Na slici 6.1 je prikazan pojednostavljeni funkcionalni dijagram serijskog porta u modu 0.



Slika 6.1 Serijski port u modu 0

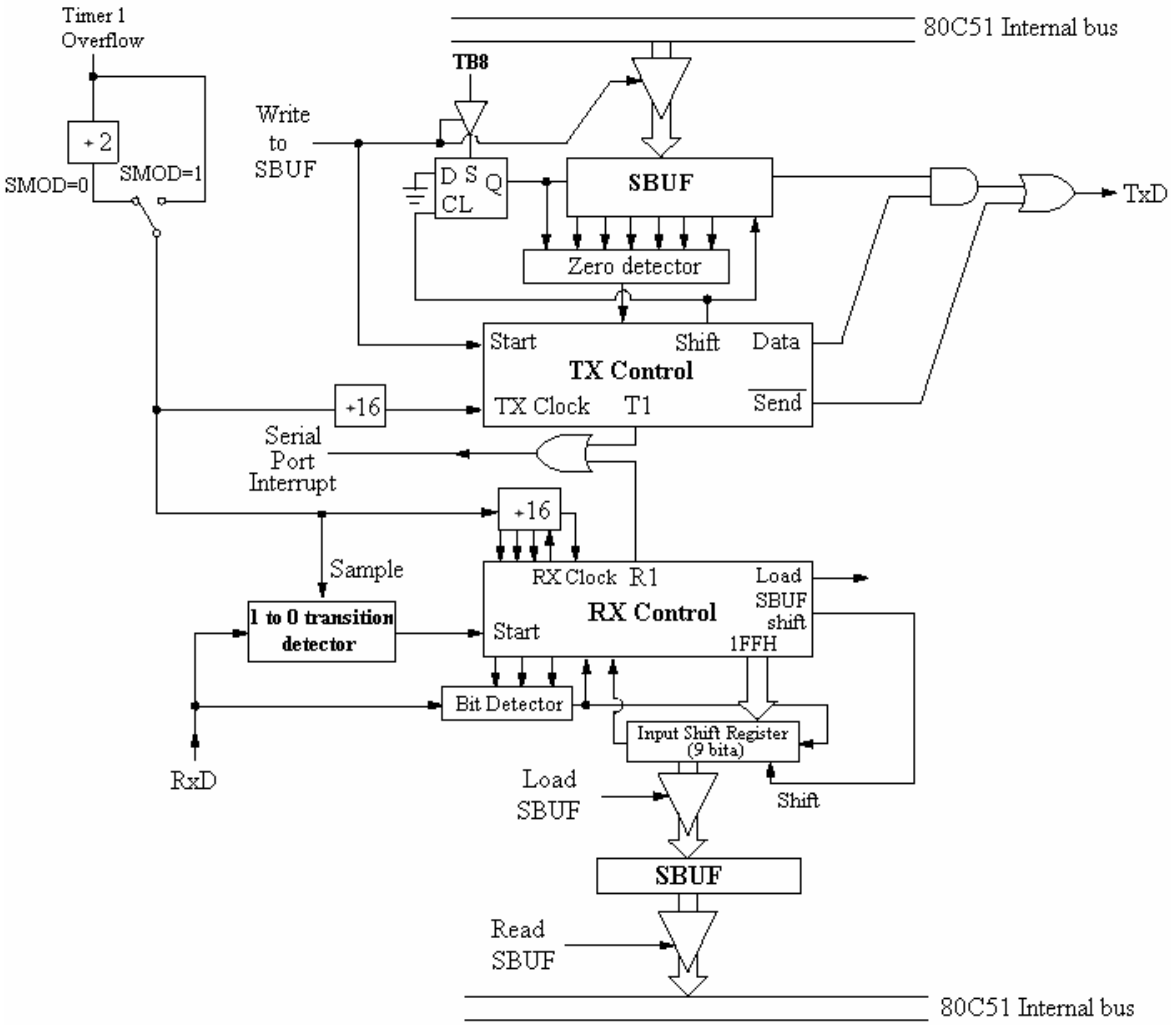
Slanje podataka započinje bilo koja instrukcija koja koristi SBUF kao odredišni registar. Pri upisu u SBUF, u trenutku S6P2, dovodi se 1 na devetu poziciju predajnog pomeračkog registra i time inicira TX Control blok da započne predaju. Nakon jednog punog mašinskog ciklusa se aktivira signal SEND, koji omogućava postavljanje izlaza pomeračkog registra na nožicu P3.0 kao i postavljanje signala SHIFT CLOCK na nožicu P3.1. Kada se biti podataka pomeraju u desno i šalju van pomeračkog registra, nule ulaze sa leve strane.

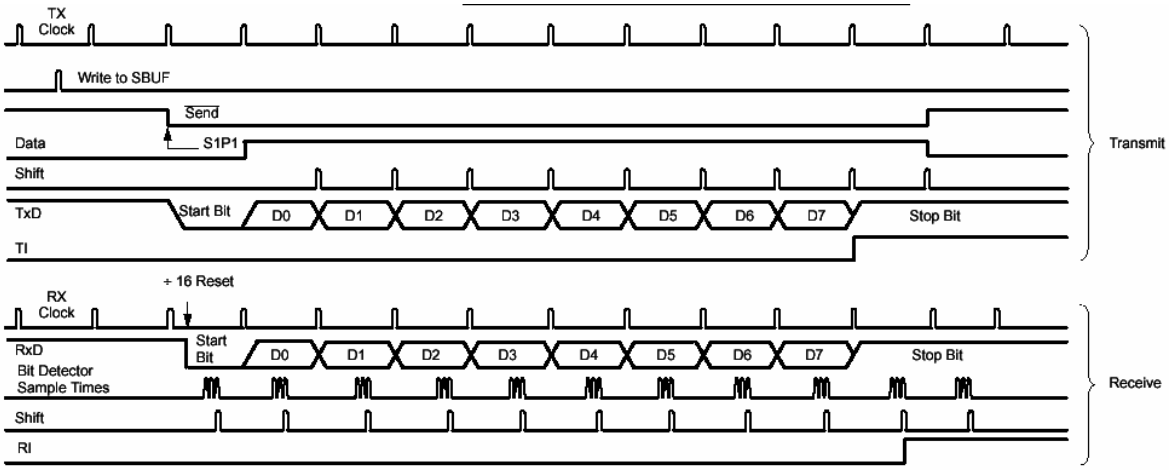
Nakon slanja svih bita podataka, TX Control jedinica deaktivira signal SEND i postavi TI=1. Prijemnik priključen na ovu predajnu sekciju treba da vrši odmeravanj na rastuću ivicu SHIFT CLOCK signala.

Prijem podataka započinje kada je REN=1 i R1=1. Nakon jednog mašinskog ciklusa od pojave internog signala S6P2, RX Control jedinica upisuje 11111110 u prijemni pomerački registar i u sledećem taktu aktivira interni signal RECEIVE, koji omogućava postavljanje signala SHIFT CLOCK na nožicu P3.1. Kada je signal RECEIVE aktivan, sadržaj prijemnog pomeračkog registra se pomera u levo za po jednu poziciju (odabira) na svaku prednju ivicu SHIFT CLOCK signala. Nakon osam taktova, RX Control jedinica završava poslednje pomeranje u levo i učitava paket podataka u SBUF. Na kraju se resetuje signal RECEIVE.

Mod 1 : *asinhroni 8 – bitni prenos*

Ovaj mod se koristi za standardnu 8-bitnu RS232 komunikaciju. Ukupno deset bita se šalje preko TxD ili prima preko RxD: jedan start bit (0), 8 bita podataka (prvo LSB) i stop bit (1). Prilikom prijema stop bit se automatski upisuje u RB8 u registru SCON. Brzina komunikacije (*baud rate*) se određuje konfigurisanjem tajmera 1 kod 8051, a dodatno i tajmerom 2 kod 8052, kao što je opisano u poglavlju o tajmerima. Na slici 6.2 prikazan je funkcionalni dijagram serijskog porta u modu 1.





Slika 6.2 Serijski port u modu 1

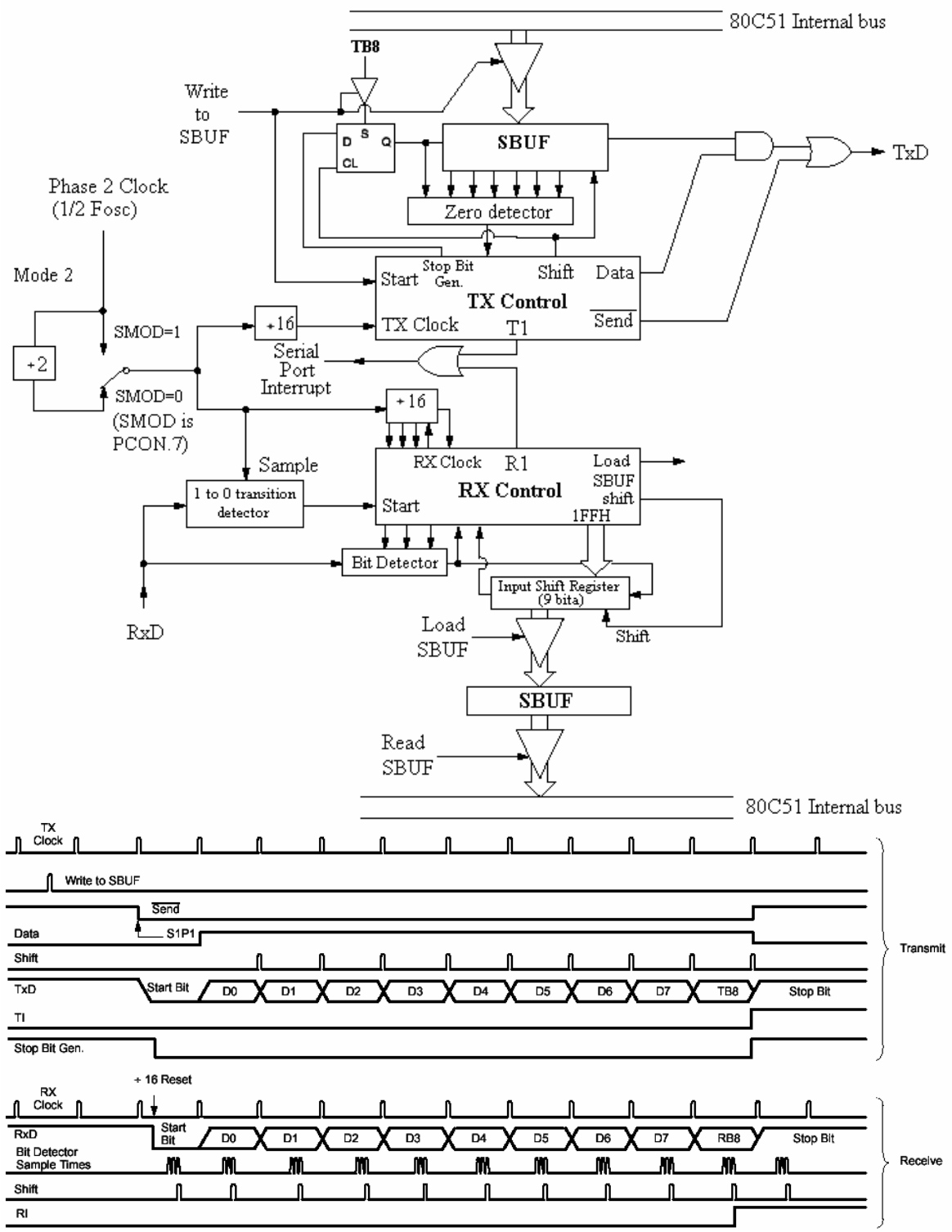
Slanje podataka započinje bilo koja instrukcija koja koristi SBUF kao odredišni registar. Pri upisu u SBUF, upisuje se 1 na devetu poziciju predajnog pomeračkog registra, čime se startuje predaja. Nakon slanja svih 10 bita setuje se bit TI.

Prijem podataka započinje kada je detektovana silazna ivica na nožici RxD. Prilikom prijema, svaki bit se očitava po 3 puta u sredini bit-intervala, a za određivanje važećeg nivoa se koristi majoritetna logika (bar 2 iste od 3 vrednosti). Nakon prijema zadnjeg (stop bita), primljeni podatak se prebacuje u SBUF i postavlja se bit RI. Prijem će biti izvršen samo ako je stop bit pravilno primljen i ako je prethodno stanje RI bita bilo 0.

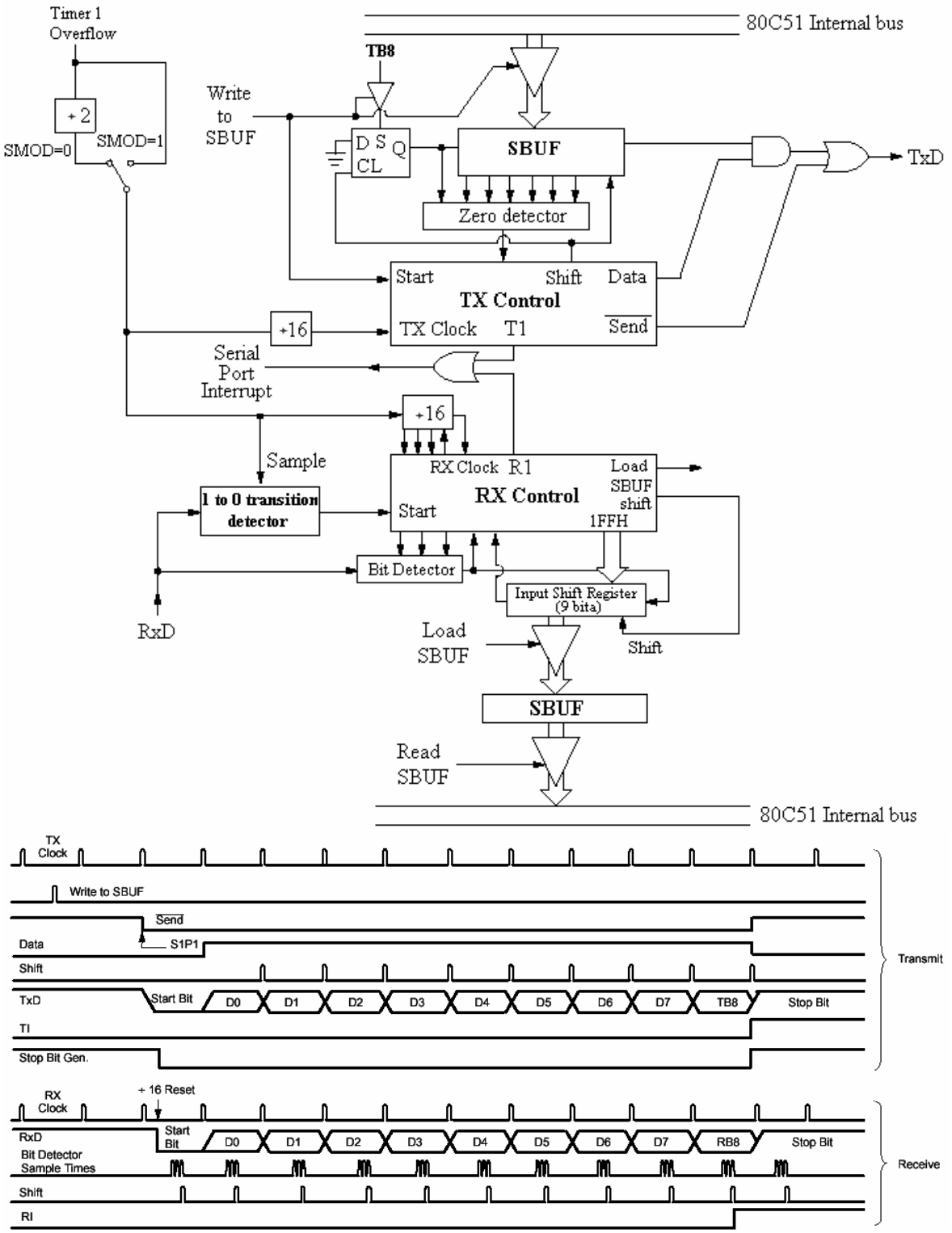
Modovi 2 i 3 : *asinhroni 9 – bitni prenos*

Ova dva načina rada odgovaraju modu 1, sa tom razlikom što se, umesto ukupno 10 bita, šalje 11 bita (jedan start bit, osam bita podataka, deveti bit – TB8/RB8 - i jedan stop bit). *U* modu 2 brzina prenosa je fiksirana na 1/32 (SMOD=1) ili 1/64 (SMOD=0) učestanosti oscilatora, a u modu 3 je promenljiva kao i u modu 1. Mod 2 je pogodan za brzu komunikaciju na malim udaljenostima, a mod 3 za standardni RS-232 prenos sa bitom parnosti.
U modovima 2 i 3 omogućena je **multiprocesorska komunikacija.** Kada je SM2=1 i ako je deveti primljeni bit RB8=0, tada RI neće biti setovan. U ovom načinu komunikacije, preko serijskog porta se šalju i primaju višebajtni paketi. Prvi bajt svakog paketa uvek ima setovan deveti bit, a sadržaj bajta je adresa odredišta. Svi ostali bajtovi u paketu imaju resetovan deveti bit i predstavljaju sadržaj podataka. Ako prijemnik ima setovan SM2, on će primiti samo prvi (adresni) bajt paketa. Ukoliko prijemnik ima odgovarajuću adresu, on resetuje bit SM2 da bi mogao da primi sve ostale bajtove iz paketa. Nakon prijema kompletnog paketa podataka, ispravno adresiran prijemnik ponovo setuje bit SM2 i prelazi u režim čekanja adresnog bajta. Svi ostali prijemnici (sa neodgovarajućom adresom) ostavljaju setovan bit SM2, čime ignorišu bajtove podataka i čekaju naredni adresni bajt.

Na slikama 6.3 i 6.4 su prikazani funkcionalni dijagrami serijskog porta u modu 2 i 3. Prijemni deo je potpuno isti kao u modu 1. Predajni deo se razlikuje od onog u modu 1 samo zbog devetog bita koji se prenosi pomeračkim registrom.



Slika 6.3 Serijski port u modu 2



Slika 6.4 Serijski port u modu 3

6. SISTEM PREKIDA MIKROKONTROLERA 8051

U mikroprocesorskim sistemima često je potrebno istovremeno pratiti rad više perifernih jedinica. To se može postići neprestanim prozivanjem jedne po jedne jedinice, proveravajući njihova stanja. Ako se utvrdi da je došlo do neke promene na nekoj od tih periferija na koju treba reagovati, preduzimaju se odgovarajuće akcije kao odgovor na promene. Na primer, nakon startovanja konverzije A/D konvertora neprekidno se vrši čitanje stanja BUSY nožice konvertora, koje označava da li je konverzija u toku ili je završena. Ako neko

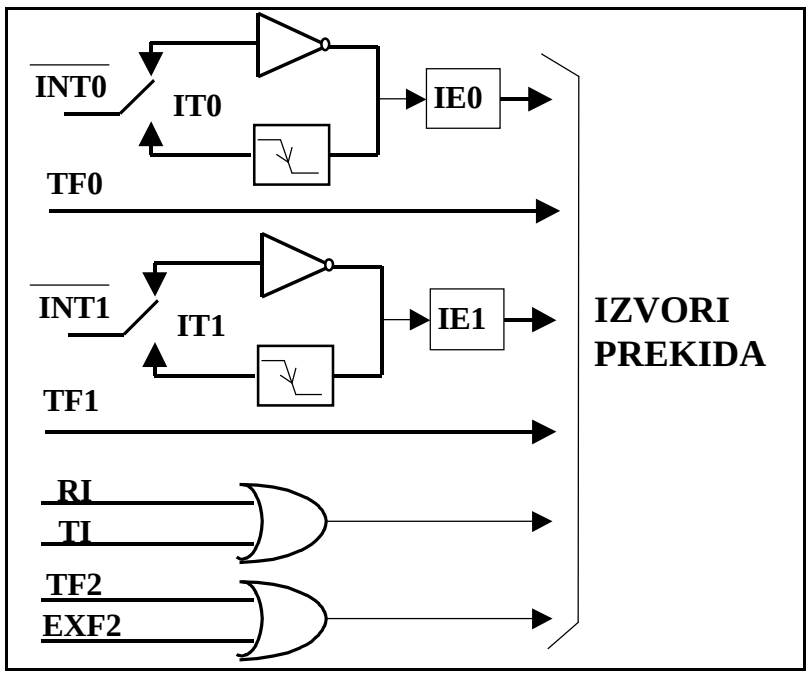
očitavanje pokaže da je konverzija završena, tada se učita odgovarajući podatak sa konvertora. U ovom slučaju mikrokontroler je neprestano zauzet proverom stanja na liniji BUSY A/D konvertora.
Drugi način praćenja rada više perifernih jedinica je da same jedinice jave kada je potrebno opsluživanje. Ta metoda se naziva metodom prekida, dok se prethodno opisana naziva metodom prozivanja (pooling). Dakle, kada se koristi metoda prekida, periferna jedinica posebnim signalom javlja kada je neophodna reakcija upravljačkog uređaja tj. mikrokontrolera. Taj signal se zove zahtev za prekidom odnosno interaptom (interrupt request). Slučaj iz prethodnog primera bi se mogao rešiti povezivanjem BUSY nožice konvertora na odgovarajuću INT (spoljašnji prekid) nožicu mikrokontrolera. Kada se BUSY deaktivira izaziva se prekid. Nakon što mikrokontroler uvaži zahtev, prelazi na podprogram za opsluživanje prekida (u ovom slučaju čitanje vrednosti konverzije), a nakon toga nastavlja sa izvršavanjem programa gde je prekinut u trenutku stizanja zahteva za prekidom. U ovom slučaju mikrokontroler se ne opterećuje proverom stanja na periferiji nego samo izvodi odgovarajuću akciju kada je to potrebno. Iz ovoga je jasno da u ovom slučaju mikrokontroler potroši mnogo manje vremena za opslužlivanje periferija, zbog čega je i program može biti mnogo efikasniji.

6.1 Izvori prekida kod 8051

Kod ovog mikrokontrolera postoji pet (šest za 8052) izvora prekida:

Spoljašnji prekidi 0 i 1 (external interrupt)

Mikrokontroler poseduje dva ulaza – INT0 i INT1 (deo su porta 3). Na njih se može dovesti signal zahteva za prekidom sa bilo kog uređaja iz okruženja mikrokontrolera. Dva bita, IT0 i IT1 registra TCON (pominjan kod tajmera), određuju hoće li ulazi INT0 i INT1 biti osetljivi na nizak nivo ili opadajuću ivicu signala zahteva za prekidom. Na primer, ako je IT0=0 pojava niskog nivoa na INT0 će se smatrati zahtevom za prekidom, dok u slučaju IT0=1 samo opadajuća ivica signala na INT0 izaziva prekid, a sledeći zahtev može se desiti se tek kada se INT0 vrati na visok nivo i ponovo padne na nizak (slika 6.1). U oba slučaja važeći zahtev za prekidom postavlja odgovarajući bit IE registra TCON na jedinicu (zahtev na INT0 bit IE0, a na INT1 bit IE1). Tek postavljanje ovih bita izaziva prelazak na odgovarajući potprogram za opsluživanje prekida (ako je prekid dozvoljen tj. nije maskiran u registru IE). Bitovi IE0 i IE1 se automatski vraćaju u stanje 0 kada mikrokontroler pređe na potprogram za opsluživanje odgovarajućeg prekida. Adresa od koje počinje potprogram za opsluživanje spoljašnjeg prekida 0 je 0003H. Za spoljašnji prekid 1 ta adresa je 0013H.



Slika 6.1 Izvori prekida kod 8051/8052

Prekidi tajmera 0 i 1 (timer interrupt)

Preticanje tajmera 0 ili 1, u bilo kom modu, može predstavljati zahtev za prekidom. Kao što je pomenuto ranije, preticanje tajmera 0 ili 1 postavlja bite TF0 ili TF1 registra TCON, označavajući da je došlo do

preticanja. Ako je odgovarajući prekid dozvoljen, tj. nije maskiran, prelazi se na potprogram za opsluživanje prekida, a time se automatski bit TF0 ili TF1, u zavisnosti od opsluženog prekida, vraća na vrednost 0. Servisni potprogram za prekid tajmera 0 počinje na adresi 000BH, a za prekid tajmera 1 na adresi 001BH.

Prekid serijskog interfejsa (serial port interrupt)

Mikrokontroler poseduje i dvosmerni serijski interfejs. Ukoliko je dozvoljen prekid serijskog interfejsa, zahtev za prekidom će se desiti svaki put kada je završeno slanje reči preko serijske veze, kao i svaki put kada je primljena cela reč. Servisni potprogram za prekid serijskog interfejsa počinje na adresi 0023H.

Prekid tajmera 2 (za 8052)

Prekid tajmera 2 se generiše logičkom ILI kombinacijom dva izvora prekida, bita TF2 (premašaja tajmera) i EXF2 (eksterni prekid). Kako prekid može biti izazvan iz bilo koje kombinacije ova dva uslova, ulaskom u prekidni potprogram interni hardver nema informaciju o tome koji se od dva moguća uzroka obrađuje. Zbog toga se u ovom slučaju uzrok prekida ne poništava automatski, nego je to zadatak prekidnog potprograma, koji treba da proveri stanje bita TF2 i EXF2, izvrši potrebnu akciju i na kraju da poništi odgovarajući bit. Na primer, ako se koristi samo tajmer u osnovnom (tajmerskom) načinu, ukoliko se bit TF2 ne obriše u prekidnom potprogramu, prekid će stalno biti aktivan i prekidni potprogram će stalno biti pozivan.

6.2 Maskiranje prekida

Svaki od pomenutih prekida se može dozvoliti ili zabraniti tj. maskirati. Kada je prekid maskiran, odgovarajući biti (TF0, TF1, IE0 i IE1) se postavljaju na 1, ali to ne izaziva prelazak na odgovarajući servisni potprogram. Kod 8051 svaki od izvora prekida se može maskirati, što se kontroliše preko registra IE (interrupt enable, slika 6.2).

Kao što se vidi na slici 6.2, postavljanjem određenog bita na 1 odgovarajući prekid biva dozvoljen. U suprotnom je maskiran. Najviši bit EA (enable all) kontroliše maskiranje svih prekida. Ukoliko je on 0, svi prekidi su maskirani bez obzira na stanje bitova istog registra koji kontrolišu maskiranost pojedinačnih izvora prekida.

(MSB) **(LSB)**

$\overline{EA}$	X	(ET2)	ES	ET1	EX1	ET0	EX0

$\overline{EA}$	ako je ovaj bit 0, svi prekidi su maskirani. Ako je 1, svaki prekid se pojedinačno maskira pomoću ostalih bita ovog registra	ET1	ako je ovaj bit jednak jedinici, prekid zbog preticanja tajmera 1 je dozvoljen, u suprotnom je maskiran
ET2	ako je ovaj bit jednak jedinici, prekid zbog preticanja tajmera 2 je dozvoljen, u suprotnom je maskiran. Bit postoji samo u 8052.	EX1	ako je ovaj bit jedinica, spoljašnji prekid 1 je dozvoljen, u suprotnom je maskiran
		ET0	ako je ovaj bit jednak jedinici, prekid zbog preticanja tajmera 0 je dozvoljen, u suprotnom je maskiran
ES	ako je ovaj bit jednak jedinici, prekid serijskog porta je dozvoljen, inače je maskiran	EX0	ako je ovaj bit jedinica, spoljašnji prekid 0 je dozvoljen, u suprotnom je maskiran

Slika 6.2 Registar za maskiranje prekida (IE)

6.3 Struktura prioriteta prekida

Kod 8051 svaki prekid može da ima dva nivoa prioriteta. Ako se tokom opsluživanja nekog prekida desi novi prekid, opsluživanje prethodnog će se prekinuti samo ako je novi prekid višeg prioriteta. Prioritet prekida određuje stanje registra IP (slika 6.3). Ako je neki bit 0, njemu pripadajući prekid je nižeg prioriteta, a u suprotnom višeg. Na taj način se svi prekidi dele u dve grupe po prioritetu. U slučaju da dva (ili više) prekida nastupe istovremeno, prvo se opslužuje prekid sa višim, a zatim sa nižim prioritetom. Ako su oba prekida istog prioriteta, tada je redosled prioriteta fiksan i sledeći: spoljašnji prekid 0, tajmer 0, spoljašnji prekid 1, tajmer 1 i na kraju prekid serijskog interfejsa. Naravno, ovaj redosled važi samo ako zahtevi za prekidom stignu istovremeno.

(MSB)							(LSB)
X	**X**	**(PT2)**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

PT2 postavljanjem ovog bita na jedinicu, prekid tajmera 2 se programira kao prekid višeg prioriteta (samo u 8052)

PS postavljanjem ovog bita na jedinicu, prekid serijskog porta se programira kao prekid višeg prioriteta

PT1 postavljanjem ovog bita na jedinicu, prekid tajmera 1 se programira kao prekid višeg prioriteta

PX1 postavljanjem ovog bita na jedinicu, spoljašnji prekid 1 se programira kao prekid višeg prioriteta

PT0 postavljanjem ovog bita na jedinicu, prekid tajmera 0 se programira kao prekid višeg prioriteta

PX0 postavljanjem ovog bita na jedinicu, spoljašnji prekid 0 se programira kao prekid višeg prioriteta

Slika 6.3 Registar za kontrolu prioriteta prekida (IP)

6.4 Procedura pozivanja potprograma za opsluživanje prekida

Potrebno je još razmotriti kako se tačno odvija priznavanje zahteva za prekidom i prelazak na potprogram za opsluživanje prekida. Nakon izvršenja svake pojedine instrukcije proverava se stanje svih bita koji označavaju validan zahtev za prekidom (IE0, TF0, IE1, TF1, TI i RI). Svaki od ovih bita može biti postavljen automatski, validnim zahtevom za prekidom (uobičajena situacija), ili softverski, upisivanjem u odgovarajući registar (softversko izazivanje prekida) – efekat je isti. Ako se utvrdi da postoji zahtev za prekidom, a pri tome:

- Taj prekid nije maskiran.
- Nije u toku opsluživanje prekida istog ili višeg prioriteta.
- Instrukcija koja je sledeća po redu nije RETI ili instrukcija upisa u registre IE ili IP.

tada se na stek stavlja sadržaj programskog brojača (PC – 16 bita), a sam programski brojač se puni adresom početka servisnog potprograma tog konkretnog prekida (registar PSW – registar stanja - se NE čuva na steku). Početne adrese servisnih potprograma su fiksne i unapred zadate za svaku vrstu prekida:

vrsta prekida	početna adresa servisne rutine
spoljašnji prekid 0	0003H
prekid tajmera 0	000BH
spoljašnji prekid 1	0013H
prekid tajmera 1	001BH
prekid serijskog porta	0023H
prekid tajmera 2	002BH

Ovo je, u principu, ekvivalentno izvršenju instrukcije LCALL sa početnom adresom servisnog potprograma kao argumentom.

Servisni potprogram se obavezno mora završiti instrukcijom RETI koja sa steka skida poslednji gurnuti 16-bitni podatak i ubacuje ga u programski brojač, što je adresa instrukcije koja je trebalo da se izvrši kada je prekid nastupio. Dakle, nakon instrukcije RETI nastavlja se sa redovnim izvršavanjem programa. Važno je primetiti da isto to radi i instrukcija RET samo što ona ne obaveštava hardver mikrokontrolera da je prekid opslužen te je instrukcija RETI obavezna. Dodatno, za razliku od naredbe RET, naredba RETI briše interne indikatore tekućeg prekida, čime su novi prekidi istog prioriteta dozvoljeni.

7. LISTA INSTRUKCIJA (SUMARNO)

Na sledećim stranama dat je pregled svih instrukcija za mikrokontroler 8051 sa mnemonikom, kratkim opisom operacije, brojem bajtova koje instrukcija zauzima u programskoj memoriji i vremenom koje je potrebno za izvršenje instrukcije, datom u broju taktova oscilatora.

Skraćenice korišćene u setu instrukcija:

- Rn - Registar R0-R7
- direct - 8-bitna adresa lokacije na kojoj se nalazi podatak. Ovo može da bude u internom RAM-u (0-127) ili neki od specijalnih registara (na primer neki port, upravljački registar, statusni registar, itd)
- @Ri - lokacija u internom RAM-u (0-127) adresirana 8-bitnim registrom R0 ili Rl
- #data - 8-bitna konstanta neposredno pridružena u instrukciji
- #data16 - 16-bitna konstanta neposredno pridružena u instrukciji
- addrl6 - 16-bitna adresa, koja sledi iza instrukcija LJMP ili LCALL

- § addr11 - 11-bitna adresa, koja sledi iza instrukcija AJMP ili ACALL. Skok koji izvode ove instrukcije je ograničen samo na zaokruženi blok od 2KB u kome se nalazi prvi bajt sledeće instrukcije posle AJMP ili ACALL.
- § rel - 8-bitni ofset bajt za adresiranje lokacije koja se nalazi u opsegu -128 do +127 u odnosu na adresu prvog bajta sledeće instrukcije.
- § bit - direktno adresirani bit u internom RAM-u (u opsegu 0 do 127, tj na adresama interne memorije u opsegu 20h-2Fh) ili u specijalnom registru (koji dozvoljava adresiranje pojedinačnih bita).

7.1 Aritmetičke operacije

Mnemonik		Opis operacije	Br. bajtova	Br. taktova
ADD	A,Rn	Saberi akumulator sa registrom	1	12
ADD	A,direct	Saberi akumulator sa direktnim RAM-om	2	12
ADD	A,@Ri	Saberi akumulator sa indirektnim RAM-om	1	12
ADD	A,#data	Saberi akumulator sa priloženom vrednošću	2	12
ADDC	A,Rn	Saberi akumulator sa registrom i C flegom	1	12
ADDC	A,direct	Saberi akumulator sa direktnim RAM-om i C flegom	2	12
ADDC	A,@Ri	Saberi akumulator sa ind. RAM-om i C flegom	1	12
ADDC	A,#data	Saberi akumulator sa datom vrednošću i C flegom	2	12
SUBB	A,Rn	Oduzmi registar i C fleg od akumulatora	1	12
SUBB	A,direct	Oduzmi direktan RAM i C fleg od akumulatora	2	12
SUBB	A,@Ri	Oduzmi indirektan RAM i C fleg od akumulatora	1	12
SUBB	A,#data	Oduzmi priloženu vrednost i C fleg od akumulatora	2	12
INC	A	Uvećaj akumulator za 1	1	12
INC	Rn	Uvećaj registar za 1	1	12
INC	direct	Uvećaj direktan RAM za 1	2	12
INC	@Ri	Uvećaj indirektan RAM za 1	1	12
DEC	A	Umanji akumulator za 1	1	12
DEC	Rn	Umanji registar za 1	1	12
DEC	direct	Umanji direktan RAM za 1	2	12
DEC	@Ri	Umanji indirektan RAM za 1	1	12
INC	DPTR	Uvećaj Data Pointer za 1	1	24
MUL	AB	Pomnoži A i B	1	48
DIV	AB	Podeli A sa B	1	48
DA	A	Podesi decimalno akumulator	1	12

7.2 Logičke operacije

Mnemonik		Opis operacije	Br. bajtova	Br. taktova
ANL	A,Rn	AND akumulator sa registrom	1	12
ANL	A,direct	AND akumulator sa direktnim RAM-om	2	12
ANL	A,@Ri	AND akumulator sa indirektnim RAM-om	1	12
ANL	A, #data	AND akumulator sa priloženom vrednšću	2	12
ANL	direct,A	AND akumulator sa direktnim RAM-om	2	12
ANL	direct,#data	AND direktni RAM sa priloženom vrednšću	3	24
ORL	A,Rn	OR akumulator sa registrom	1	12
ORL	A,direct	OR akumulator sa direktnim RAM-om	2	12
ORL	A,@Ri	OR akumulator sa indirektnim RAM-om	1	12
ORL	A,#data	OR akumulator sa priloženom vrednošću	2	12
ORL	direct,A	OR direktni RAM sa akumulatorom	2	12
ORL	direct,#data	OR direktni RAM sa priloženom vrednošću	3	24
XRL	A,Rn	EX-OR akumulator sa registrom	1	12
XRL	A,direct	EX-OR akumulator sa direktnim RAM-om akumulatorom	2	12

XRL	A,@Ri	EX-OR akumulator sa indirektnim RAM-om	1	12
XRL	A,#data	EX-OR akumulator sa priloženom vrednošću	2	12
XRL	direct,A	EX-OR direktni RAM sa akumulatorom	2	12
XRL	direct,#data	EX-OR direktni RAM sa priloženom vrednošću	3	24
CLR	A	Poništi stanje akumulatora	1	12
CPL	A	Komplementiraj sve bitove u akumulatoru	1	12
RL	A	Rotiraj nalevo akumulator	1	12
RLC	A	Rotiraj nalevo akumulator kroz C fleg	1	12
RR	A	Rotiraj nadesno akumulator	1	12
RRC	A	Rotiraj nadesno akumulator kroz C fleg	1	12
SWAP	A	Zameni gornja i donja 4 bita akumulatora	1	12

7.3 Prenos podataka

Mnemonik		Opis operacije	Br. bajtova	Br. taktova
MOV	A,Rn	Upiši u akumulator registar	1	12
MOV	A,direct	Upiši u akumulator direktan RAM	2	12
MOV	A,@Ri	Upiši u akumulator indirektan RAM	1	12
MOV	A,#data	Upiši u akumulator priloženu vrednost	2	12
MOV	Rn,A	Upiši u registar akumulator	1	12
MOV	Rn,direct	Upiši u registar direktan RAM	2	24
MOV	Rn,#data	Upiši u registar priloženu vrednost	2	12
MOV	direct,A	Upiši u direktan RAM akumulator	2	12
MOV	direct,Rn	Upiši u direktan RAM registar	2	24
MOV	direct,direct	Upiši direktan RAM u direktan RAM	3	24
MOV	direct,@Ri	Upiši u direktan RAM indirektan RAM	2	24
MOV	direct,#data	Upiši u direktan RAM priloženu vrednost	3	24
MOV	@Ri,A	Upiši u indirektan RAM akumulator	1	12
MOV	@Ri,direct	Upiši u indirekni RAM direktni RAM	2	24
MOV	@Ri,#data	Upiši u indirektni RAM priloženu vrednost	2	12
MOV	DPTR,#data16	Upiši u Data Pointer priloženu 16-bitnu vrednost	3	24
MOVC	A,@A+DPTR	Upiši u akumulator Indirektni Code Byte @A+DPTR	1	24
MOVC	A,@A+ PC	Upiši u akumulator indirektni Code Byte @A+PC	1	24
MOVX	A,@Ri	Upiši u akumulator indirektni Ext RAM @Ri	1	24
MOVX	A,@DPTR	Upiši u akumulator indirektni Ext RAM @DPTR	1	24
MOVX	@Ri, A	Upiši u indirektni Ext RAM @Ri akumulator	1	24
MOVX	@DPTR,A	Upiši u indirektni Ext RAM @DPTR akumulator	1	24
PUSH	direct	Stavi direktni RAM na stek	2	24
POP	direct	Uzmi direktni RAM sa steka	2	24
XCH	A,Rn	Izmeni vrednosti registra i akumulatora	1	12
XCH	A,direct	Izmeni vrednosti direktnog RAM-a i akumulatora	2	12
XCH	A,@Ri	Izmeni vrednosti indirektnog RAM-a i akumulatora	1	12
XCHD	A,@Ri	Izmeni vrednosti donje cifre ind. RAM-a i	1	12

7.4 Bulove operacije

Mnemonik		Opis operacije	Br. bajtova	Br. taktova
CLR	C	Resetuj C fleg	1	12
CLR	bit	Resetuj direktan bit	2	12

SETB	C	Setuj C fleg	1	12
SETB	bit	Setuj direktan bit	2	12
CPL	C	Komplementiraj C fleg	1	12
CPL	bit	Komplementiraj direktan bit	2	12
ANL	C,bit	AND C fleg sa direktnim bitom	2	24
ANL	C,/bit	AND C fleg sa komplementiranim direktnim bitom	2	24
OR	C,bit	OR C fleg sa direktnim bitom	2	24
ORL	C,/bit	OR C fleg sa komplementaranim direktnim bitom	2	24
MOV	C,bit	Upiši u C fleg direktan bit	2	12
MOV	bit,C	Upiši u direktan bit C fleg	2	24
JC	rel	Skoči ako je C fleg setovan	2	24
JNC	rel	Skoči ako C fleg nije setovan	2	24
JB	bit,rel	Skoči ako je direktan bit setovan	3	24
JNB	bit,rel	Skoči ako direktan bit nije setovan	3	24
JBC	bit,rel	Skoči ako je direktan bit setovan i resetuj bit	3	24

7.5 Potprogrami i skokovi

Mnemonik		Opis operacije	Br. bajtova	Br. taktova
ACALL	addr 11	Poziv potprograma na apsolutnoj (11-bitnoj) adresi unutar tekućeg bloka od 2KB	2	24
LCALL	addr 16	Poziv potprograma na apsolutnoj 16-bitnoj adresi	3	24
RET		Povratak iz potprograma	1	24
RETI		Povratak iz prekidne rutine	1	24
AJMP	addr 11	Skok na apsolutnu (11-bitnu) adresu unutar tekućeg bloka od 2KB	2	24
LJMP	addr 16	Skok na apsolutnu 16-bitnu adresu	3	24
SJMP	rel	Kratki skok (8-bitna relativna adresa)	2	24
JMP	@A+DPTR	Indirektni skok na adresu iz Code Mem @A+DPTR	1	24
JZ	rel	Skoči ako je akumulator jednak nuli	2	24
JNZ	rel	Skoči ako je akumulator različit od nule	2	24
CJNE	A,direct,rel	Uporedi A sa direktnim RAM-om i skoči ako su različiti	3	24
CJNE	A,#data,rel	Uporedi A sa pridr. podatkom i skoči ako su različiti	3	24
CJNE	Rn,#data,rel	Uporedi Rn sa pridr. podatkom i skoči ako su različiti	3	24
CJNE	@Ri,#data,rel	Uporedi ind.RAM sa podatkom i skoči ako su različiti	3	24
DJNZ	Rn,rel	Umanji registar i skoči ako nije jednak nuli	3	24
DJNZ	direct,rel	Umanji direktni RAM i skoči ako nije jednak nuli	3	24
NOP		No operation - bez operacije	1	12

8. ASEMBLER A51

Direktive koje će ovde biti date odnose se na Franklin asembler A51, za druge asemblere oznake mogu da se razlikuju, pa o tome treba voditi računa.

8.1 Direktive

EQU dodeljuje numeričku vrednost navedenom simbolu.
Primer:

```
temperatura  EQU  20
```

Numerička vrednost koja je ovom direktivom dodeljena simbolu temperatura ne može se redefinisati u programu. Ovo se koristi u programu da bi kod bio čitljiviji, jer kada se u programu naiđe na simbole *temperatura* ili *brzina*, zna se da taj simbol označava vrednost temperature ili brzine. Ako bi bile napisane samo brojčane vrednosti, program bi bio znatno nerazumljiviji.

SET radi isto kao i EQU, ali se može redefinisati neograničen broj puta tokom izvršavanja programa.
Primer:

```
Brzina  SET  25
…
Brzina  SET  100
…
Brzina  SET  Brzina+10
```

BIT dodeljuje adresu bita navedenom simbolu.
Primer:

```
RELE        BIT  P1.0
LED         BIT  P3.4
RAMWR       BIT  WR
IZLAZ       BIT  P2.0
```

ORG definiše tekuću vrednost brojača lokacija.
Primer:

```
ORG  2000
```

Sada će adresa sledeće instrukcije ili komande DB (ili DW) imati vrednost 2000.

DS rezerviše prostor u memoriji izražen u bajtovima.
Primer:

```
        ORG  100        ; počni od adrese 100
        DS  7           ; rezerviši 7 bajtova
NovaLok:                ; nastavi program
```

Ovo znači da će labela NovaLok imati vrednost 107, i na nju se može skočiti ako želimo da nastavimo rad na adresi 107.

DB upisuje navedenu vrednost bajta u programsku memoriju. Ako se navodi više vrednosti, one se odvajaju zarezima. Ako se navodi ASCII niz, stavlja se pod jednostruke navodnike.
Primer:

```
DB  22,33,'Alarm',0
```

DW je isto kao DB, ali se upisuje 16-bitna (dvo-bajtna) vrednost. Za razliku od nekih drugih procesora (na primer 8086), prvo se upisuje visoki, pa niski bajt.

CSEG označava da se naredni segment odnosi na programsku memoriju.

XSEG označava da se naredni segment odnosi na spoljni RAM.

DSEG označava da se naredni segment odnosi na interni RAM.

ISEG označava da se naredni segment odnosi na gornji deo internog RAM-a, koji se može adresirati isključivo indirektno pomoću registara R0 i R1 (lokacije 80h-FFh, interna memorija koja nije implementirana u mikrokontroleru 8051).

8.2 Rad u asembleru A51

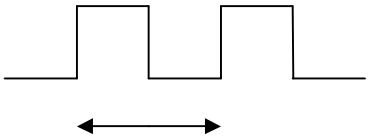
Izvorni program može se napisati u bilo kom editoru, ali pošto A51 radi pod operativnim sistemom DOS, ovde će to biti opisano.

Kada se *startuje* **MSDOS** potrebno je ukucati **Edit.** Dobiće se editorski prozor u kom treba odabrati opciju **File-New,** i uneti izvorni kod i snimiti pod nazivom **ImeProg.asm** (ImeProg može imati najviše 8 karaktera). Posle toga treba odabrati opciju **File-Exit** i vratiti se u MSDOS.
Asembliranje se vrši unosom komande **A51 ImeProg.asm.** Posle toga se na ekranu pojavljuje informacija da li ima grešaka ili je program ispravan. Ako ima grešaka one se mogu videti unosom komande **Edit ImeProg.lst.**
Kada se greške isprave dobijanje **hex fajla** se vrši unosom komande **OHS51 ImeProg.obj**.

9. PRIMERI

Korišćenje **prekidne rutine** kod tajmera biće prikazano na sledećim zadacima:

Zadatak 1. Sa mikrokontrolerom 8051, čija je frekvencija spoljašnjeg oscilatora 12MHz realizovati povorku pravougaonih impulsa jednakog trajanja (četvrtki) frekvencije f=1kHz na portu P1, bit 0 (P1.0).
Mikrokontroler isprogramirati u asembleru A51.



T=1ms

Rešenje:
Kako je potrebna kontrola vremena trajanja stanja 0 i 1 impulsa, koristiće se Timer0. Sad je pitanje u kom režimu treba tajmer da radi. Da bi se to znalo, treba prvo videti koliko impulsa tajmer treba da izbroji. Pošto je potrebno 12 perioda ciklusa takta oscilatora za jedan taktni impuls tajmera, perioda ulazne frekvencije tajmera je:

$T_{timer\ .} = 12 \cdot T_{osc} = 1ms$

Kako se tajmer uvećava za 1 posle T_{timer}, a potrebno je da se stanje impulsa menja svakih T/2, onda je broj koji tajmer treba da odbroji pre nego što napravi prekid (interapt):

$$n = \frac{T/2}{\mathrm{T_{timer.}}} = 500$$

Pošto ovoliki broj ne može da izbroji osmobitni tajmer, uzima se 16-bitni. Sad treba odrediti od kog broja tajmer treba da počne da broji, imajući u vidu da tajmer broji na više i da se prekid događa kada tajmer prelazi iz stanja FFFFh u stanje 0:
0000 - 01F4 = FE0C jer je 500dec=01F4hex.

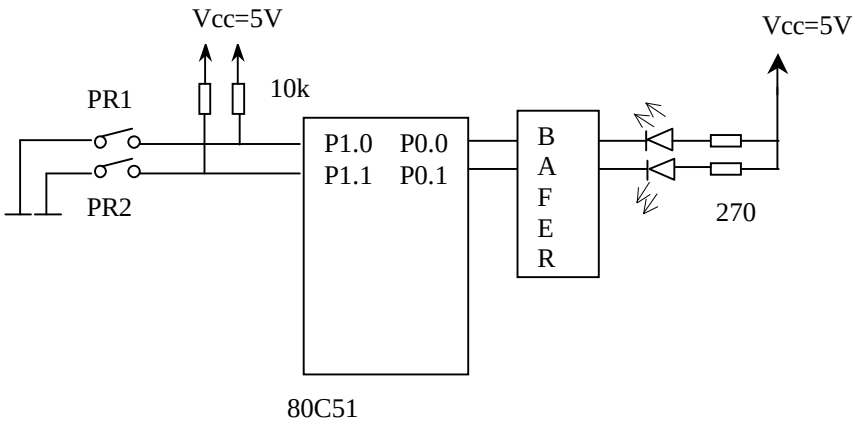
Program:

```
DSEG                        ; naredni segment se odnosi na interni RAM
        ORG     20h
Var1:   DS      1           ; rezervisanje jednog bajta na lokaciji 20h
Stanje  BIT     Var1.0      ; dodeljivanje adrese bita
Izlaz   BIT     P1.0        ; simbolima Stanje i Izlaz

CSEG                        ; naredni segment odnosi na programsku memoriju
        ORG     0000H       ; kad se startuje, program kreće od adrese 0000h
        AJMP    INICIJALIZACIJA

        ORG     000BH       ; adresa vektora prekida Timera0
        AJMP    PREKID      ; skok na prekidnu rutinu
INICIJALIZACIJA:
        MOV     IE,#82H     ; dozvola interapta prekoračenja Timera0 (EA=1,ET0=1)
        MOV     TMOD,#01H   ; Timer0 je 16-bitni
        MOV     TH0,#0FEH   ; početna vrednost
        MOV     TL0,#0CH    ; tajmera je FE0C
        SETB    Stanje      ; jer je početno stanje porta  1
        SETB    TR0         ; start Timer0 (TR0=1)
```

```
PETLJA:
        NOP                 ; program se vrti u ovoj beskonačnoj petlji i čeka
        SJMP    PETLJA      ; prekide Timera0

PREKID:
        CLR     TR0         ; tajmer je potrebno zaustaviti pre punjenja,
        MOV     TH0,#0FEH   ; a zatim ponovo inicijalizovati Timer0, jer on
        MOV     TL0,#0CH    ; kad jednom odbroji, nastavlja da broji od 0
        SETB    TR0         ; start tajmera
        CPL     Stanje      ; komplementira tekuće stanje
        MOV     C,Stanje    ; prebacuje tekuće stanje na izlaz
        MOV     Izlaz,C
        RETI                ; povratak iz prekidne rutine
END
```

Zadatak 2. Sa mikrokontrolerom 8051, čija je frekvencija spoljašnjeg oscilatora 12MHz izvršiti kontrolu rada dve LED diode. Izbor LED diode se vrši prekidačem PR1 (kada je PR1 otvoren trepće crvena, a kada je zatvoren trepće zelena LED dioda), a brzina treperenja LED diode se određuje prekidačem PR2 (kada je PR2 otvoren dioda treba da trepće sa učestanošću od 10Hz, a kada je zatvoren, dioda treba da trepće sa učestanošću od 0.5Hz). Obe LED diode treba da se uključe na nizak logički nivo nožice P0.0 (crvena) i P0.1 (zelena).
Mikrokontroler isprogramirati u asembleru A51.

Uprošćena slika:



Rešenje:

Broj tajmerskih ciklusa koji treba da se odbroji kada je PR2 otvoren je N1=50ms/1μs=50000, a kada je zatvoren N2=1s/1μs=1000000

Kako 16-bitni tajmer može maksimalno da odbroji 65536 puta potrebno je uvesti pomoćni brojač u prekidnoj rutini za detekciju broja prekida.
Pošto je N2=20*50000 vidi se da je potrebno da se desi 20 prekida pa da se onda izvši promena logičkog nivoa signala na nožici P0.0 ili P0.1, za frekvenciju od 0.5Hz.
U Timer0 je potrebno uneti vrednost: 0-50000=0000h-C350=3CB0

Program:

```
DSEG                    ; naredni segment se odnosi na interni RAM
        ORG   20h       ; skok na adresu 20h
F10     EQU   1         ; dodela vrednosti 1 simbolu F10 (označava da je na frekv. 10Hz
                        ; potreban 1 prekid pre promene stanja LED diode)
F05     EQU   20        ; dodela vrednosti 20 simbolu F05 (označava da je na frekv.
                        ; 0.5Hz potrebno 20 prekida pre promene stanja LED diode)
Mask:   DS    1         ; rezervisanje jednog bajta na lokaciji 20h
Count:  DS    1         ; rezervisanje jednog bajta na lokaciji 21h
CLED    BIT   P0.0      ; izlazno stanje crvene LED diode
ZLED    BIT   P0.1      ; izlazno stanje zelene LED diode
PR1     BIT   P1.0      ; ulazno stanje prekidača PR1
PR2     BIT   P1.1      ; ulazno stanje prekidača PR2

CSEG
        ORG   0000H     ; kad se startuje, program kreće od adrese 0000h
        JMP   INICIJALIZACIJA

        ORG   000BH     ; adresa vektora prekida Timera0
        JMP   PREKID    ; skok na prekidnu rutinu

INICIJALIZACIJA:
        MOV   IE,#82H   ; dozvola interapta prekoračenja Timera0
        MOV   TMOD,#01H ; Timer0 je 16-bitni
        MOV   TH0,#3CH  ; početna vrednost
        MOV   TL0,#0B0H ; Timera0 je 3CB0
        MOV   R4,#1     ; R4 <- 1       R4 je pomoćni brojač za brojanje N2
        SETB  TR0       ; start Timer0

MAIN:                   ; program se vrti u ovoj petlji (proverava prekidače) i čeka prekide
```

```
        MOV     A,#1            ; izbor aktivne LED diode
        JB      PR1,Crvena      ; skoči na labelu Crvena ako je PR1=1
        MOV     A,#2
        SETB    CLED            ; P0.0=1 (zabrana paljenja crvene LED diode)
        SJMP    L1
Crvena:
        SETB    ZLED            ; P0.1=1 (zabrana paljenja zelene LED diode)
L1:
        MOV     MASK,A
        MOV     A,#F10          ; Izbor frekvencije
        JB      PR2,PostaviBrojac
        MOV     A,#F05
PostaviBrojac:
        MOV     Count,A
        SJMP    MAIN            ; nazad u petlju

PREKID:
        CLR     TR0             ; Tajmer0 je potrebno zaustaviti
        MOV TH0,#3CH            ; početna vrednost
        MOV TL0,#0B0H           ; Timera0 je 3CB0
        SETB    TR0
        DJNZ    R4,Int_Kraj
        MOV     R4,Count
        MOV     A,Mask
        XRL     P0,A
Int_Kraj:
        RETI
END
```

10. KOMPAJLER PROGRAMSKOG JEZIKA C ZA MIKROKONTROLERE IZ FAMILIJE 8051

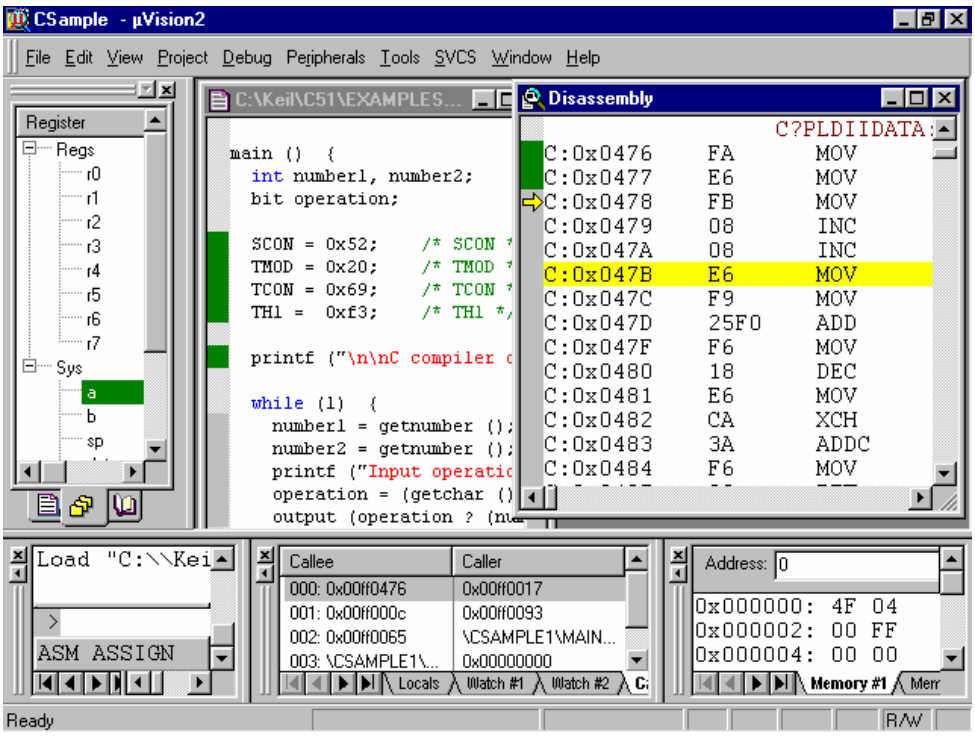
Kompajler koji će ovde biti razmatran je Keil-ov C51 sa integrisanim okruženjem µ**Vision2** koje sadrži: editor izvornog koda, sistem za podešavanje parametara i poziv kompajlera, kao i sistem za testiranje i simulaciju prevedenog programa. µ**Vision2** se koristi za kreiranje izvornog fajla (koda) i njegovo povezivanje unutar projekta, koji definiše željenu aplikaciju. µ**Vision2** automatski kompajlira, asemblira i povezuje sve programe u projektu. On podržava: C51 compiler, A51 macro assembler, BL51 linker/locator, LIB51 library manager, OH51 object-hex converter i RTX-51 real-time operating system.

Postupak kreiranja softverske aplikacije:

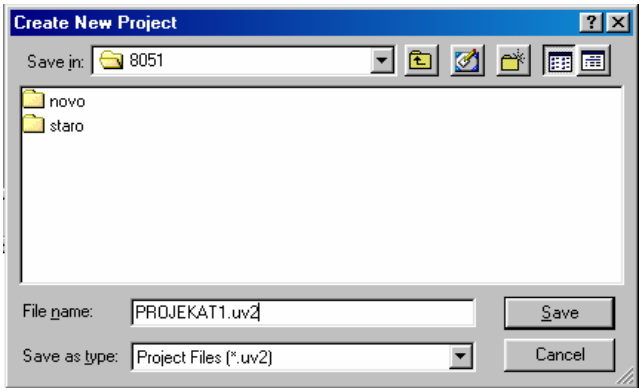
- Startuje se µ**Vision2,** definiše se projekat i odabere se mikrokontroler sa ponuđene liste.
- Napiše se **izvorni (source) fajl** u editoru i doda se u projekat.
- Izvrši se podešavanje (unos) parametara hardverske strukture izabranog mikrokontrolera.
- **Izvorni fajl** prolazi **C51 compiler** ili **A51 macro assembler.** Procesi kompajliranja ili asembliranja stvaraju objektne fajlove.
- **Objektni fajlovi** se korišćenjem LIB51 library managera formiraju u objektne biblioteke.
- **Objektni fajlovi i objektne biblioteke** se povezuju sa BL51 Linker/Locator-om u apsolutni objektni modul. Ceo kod u apsolutnom objektnom fajlu zauzima fiksne memorijske lokacije.
- Zatim se izvrši komanda **build project,** pri čemu se formira HEX fajl ako nema grešaka u programu.
- Ako postoje **greške**, one se se prikazuju u donjem delu ekrana, uz kratak komentar o kakvoj je greški reč.
- Kada su ispravljene sve formalne greške, može se koristiti µ**Vision2 debugger** za simulaciju rada 8051 sistema. On prikazuje: memorijsku mapu promenljivih, lokalne promenljive i periferije.
- Na kraju se može programirati dati čip, direktnim korišćenjem **HEX** fajla, a po potrebi se **HEX** fajl pre programiranja prvo konvertuje u binarni (**BIN)** fajl.

11. STARTOVANJE µVISION2 PROGRAMA I KREIRANJE PROJEKTA

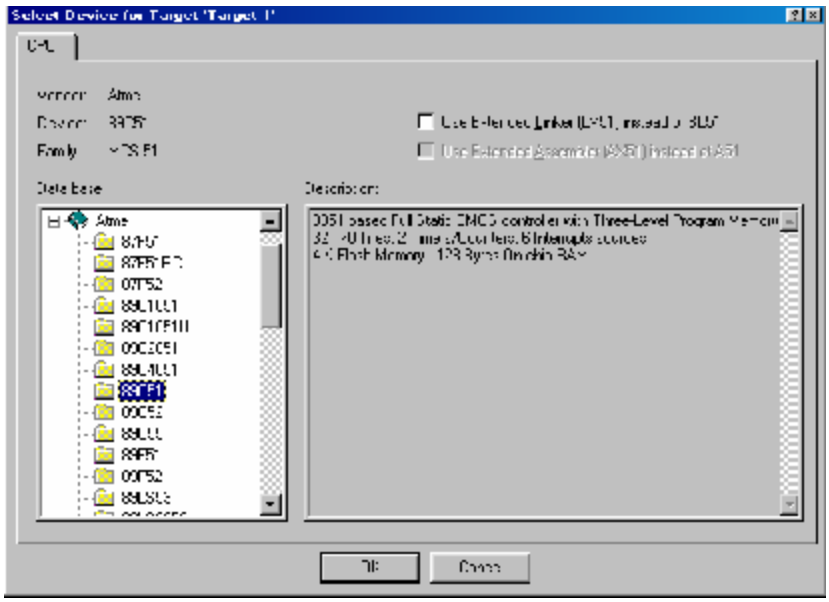
Program se startuje pozivom µVision2. Izgled menija i pojedinih ‘prozora' prikazan je na sledećoj slici:



Kreiranje novog projekta se vrši izborom opcije **Project-New Project,** posle čega se odabere mesto na hard disku gde će biti smešten projekat sa ekstenzijom .uv2 (preporučuje se prethodno kreiranje odgovarajućeg foldera na hard disku). Na primer, ako se projekat zove PROJEKAT1.UV2:

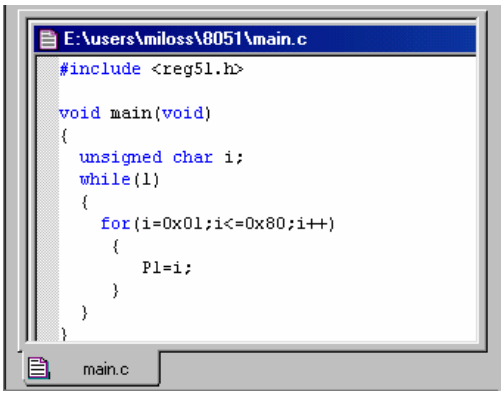


Odmah posle toga program zahteva izbor mikrokontrolera (CPU), kao što je prikazano na sledećoj slici gde je odabran Atmelov 89C51:

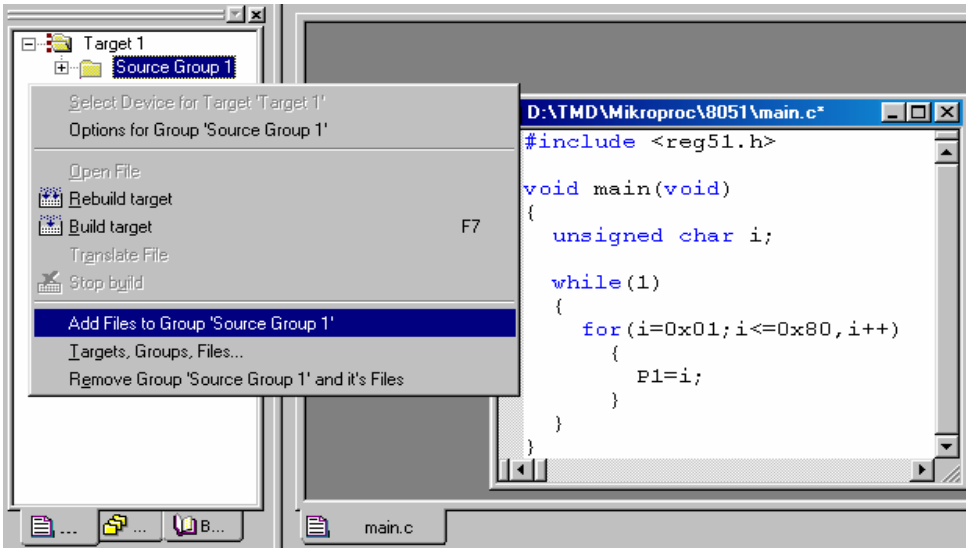


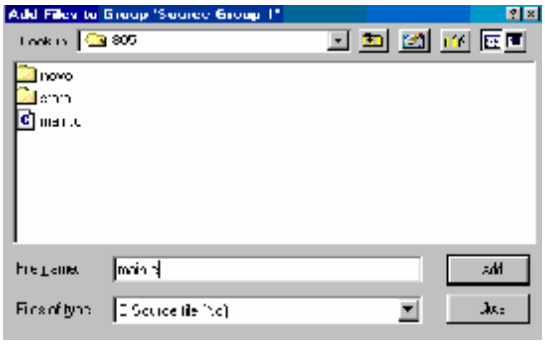
12. KREIRANJE IZVORNOG (SOURCE) FAJLA

Kreiranje novog izvornog fajla se vrši izborom opcije **File-New,** pri čemu se otvara prazan editorski prozor u koji se unosi izvorni kod u **C**-u. Zatim se izabere opcija **File-Save As** i zada se ime fajla sa ekstenzijom **.c**, npr. Main.c:



Sada treba taj fajl dodati u projekat. To se može uraditi tako što se prvo otvori **Project Window** (**View - Project Window**), ako već nije otvoren. Zatim se klikne desnim tasterom miša na **Source Group1,** a potom izabere **Add Files to Group 'Source Group1'**:





Prvi red programa sadrži direktivu **#include** za čitanje (ubacivanje) izvornog fajla sa deklaracijama za odgovarajući mikrokontroler. U toku kompajliranja, fajl naveden u direktivi #include se ubacuje u osnovni fajl (u ovom slučaju u main.c). Pored #include značajna direktiva je i **#define** kojom se definiše makro ili konstanta.

Primeri:

#include<math.h>	uključenje matematičkih funkcija
#include<reg51.h>	uključenje adresa portova, registara i specijalnih bita u memoriji
#define PI 3.14159	/* definisanje konstante PI, sada se u programu svuda piše PI a kompajler zna da je to broj 3.14159 */
#define Timer0H 0xE4	
#define Timer0L 0xF0	

Tipovi podataka koje podržava C51:

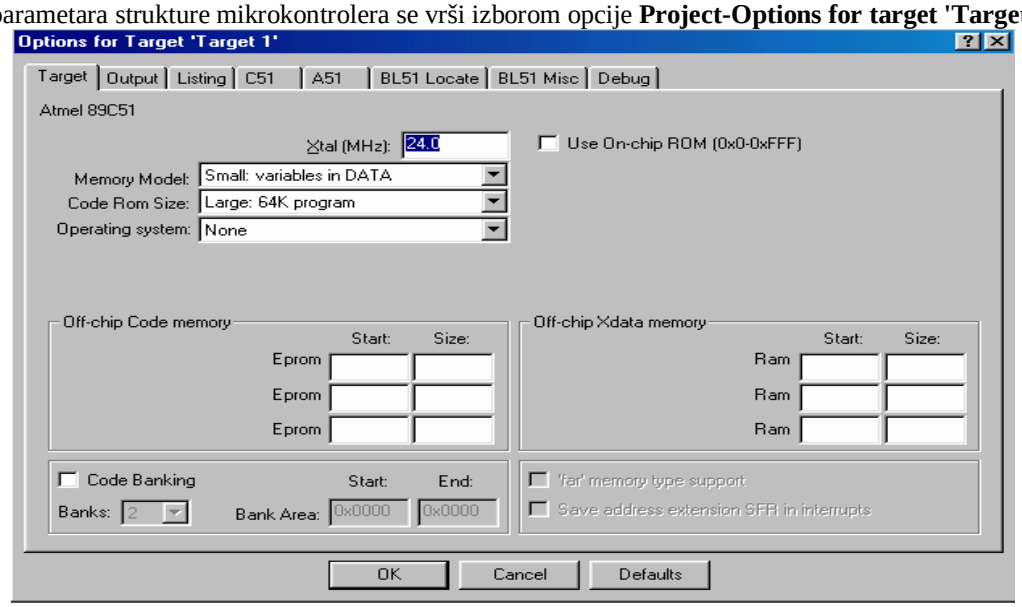
DATA TYPE	BITS	BYTES	VALUE RANGE
Bit	1		0/1
Signed char	8	1	-128 to +127
Unsigned char	8	1	0 to 255
Enum	8/16	1/2	-128 to +127 / –32768 to +32768
Signed short	16	2	–32768 to +32768
Unsigned short	16	2	0 to 65535
Signed int	16	2	–32768 to +32768
Unsigned int	16	2	0 to 65535
Signed long	32	4	-2147483648 to 2147483647
Unsigned long	32	4	0 to 4294967295

Float	32	4	±1.175494E-38 to ±3.402823E+38
Sbit	1		0/1
Sfr	8	1	0 to 255
Sfr 16	16	2	0 to 65535

Treba zapaziti da je u programu **main.c** promenljiva **i** definisana kao **unsigned char,** što znači da je osmobitna. Kako port P1 ima osam nožica, vrednost koja mu se pridružuje mora biti osmobitna tj. u intervalu od 0 do 255. Pojedinim nožicama porta P1 pristupa se pisanjem P1^x gde x označava broj nožice (npr. P1^2=0), gde je x u opsegu od 0 do7.

13. UNOS PARAMETARA HARDVERSKE STRUKTURE MIKROKONTROLERA

Unos parametara strukture mikrokontrolera se vrši izborom opcije **Project-Options for target 'Target1':**



Xtal(MHz) označava frekvenciju rada mikrokontrolera (bitno samo za simulator).
Memorijski model određuje koliko će memorije biti korišćeno za promenljive i podatke. Memorijski modeli koje podržava C51 su:

- **Small Model** – sve promenljive se nalaze u internoj memoriji podataka, i ovaj pristup promenljivama je jako brz (deklariše se sa **data**).
- **Compact Model** – sve promenljive se nalaze u grupisanim blokovima od po 256 bajta spoljašnje memorije podataka (deklariše se sa **pdata**).
- **Large Model** – sve promenljive se nalaze u spoljašnjoj memoriji podataka (deklariše se sa **xdata**).

Memory Type	Description
code	Program memory (64 KBytes); accessed by opcode by MOVC @A+DPTR
data	Directly addressable internal data memory; fastes access to variables (128 bytes)
idata	Indirectly addressable internal data memory; accessed across the full internal adress space (256 bytes)
bdata	Bit-adressable internal data memory; allows mixed bit and byte access (16 bytes)
xdata	External data memory (64 KBytes); accessed by opcode MOVX @DPTR
pdata	Paged (256 bytes) external data memory; accessed by opcode MOVX @Rn

Programska (CODE) **memorija** može se nalaziti unutar 8051 CPU (internal data memory) a može biti i spoljašnja (external data memory). Programski kod, uključujući sve funkcije i biblioteke, kao i program koji se izvršava, čuva se u programskoj memoriji. Takođe, konstante mogu da se čuvaju u programskoj memoriji. Ovoj memoriji može se pristupati korišćenjem **CODE** direktive.

Internal data memory se nalazi unutar 8051 CPU i u nju se mogu pisati i čitati podaci. Internu memoriju čini 256 bajta, od kojih prvih 128 se mogu adresirati direktno i indirektno, a drugih 128 se adresiraju samo indirektno. Pristup internoj memoriji je jako brz zato što se koriste 8 bitne adrese.

Interna memorija se može podeliti na tri tipa:

- **data memory** označava pristup prvom bloku od 128 bajtova interne memorije. Promenljivama koje su tamo uskladištene pristupa se direktnim i indirektnim adresiranjem.
- **idata memory** označava pristup kompletnoj internoj memoriji (svih 256 bajtova). Međutim, adresiranje je isključivo indirektno, što je sporije od direktnog pristupa.
- **bdata memory** označava pristup bit-adresibilnom delu u internoj memorijskoj oblasti 20h-2Fh (16 bajtova).

External data memory je memorija smeštena izvan 8051 CPU i njoj se može pristupiti pri pisanju i čitanju podataka. Pristup spoljašnjoj memoriji je dosta sporiji nego internoj, zato što joj se pristupa indirektno preko **data pointer** registra (DPTR), koji mora biti napunjen 16-bitnom adresom pre pristupa spoljašnjoj memoriji, ili preko internih registara **R0** i **R1**.

Spoljašnja memorija ima do 64 KBajta, čiji adresni prostor ne mora biti u potpunosti korišćen kao memorijski. Periferijski uređaji mogu biti kontrolisani pristupanjem spoljašnjoj memoriji (memorijsko mapiranje periferijskih uređaja).

Spoljašnja memorija se može podeliti na dva tipa:

- **xdata memory** označava pristup svim lokacijama u 64KBajta adresnog prostora spoljašnje memorije.
- **pdata memory** označava pristup blokovima od po 256 bajtova spoljašnje memorije.

Special function register (SFR) memory je memorija veličine 128 bajta, koja se koristi za kontrolu: tajmera, brojača, serijskog i paralelnog I/O porta i periferija. U osnovi, ovo nije memorija, nego zona memorijski mapiranih periferijskih uređaja, koji ne popunjavaju kompletan adresni prostor.

Primeri deklaracije promenljivih:

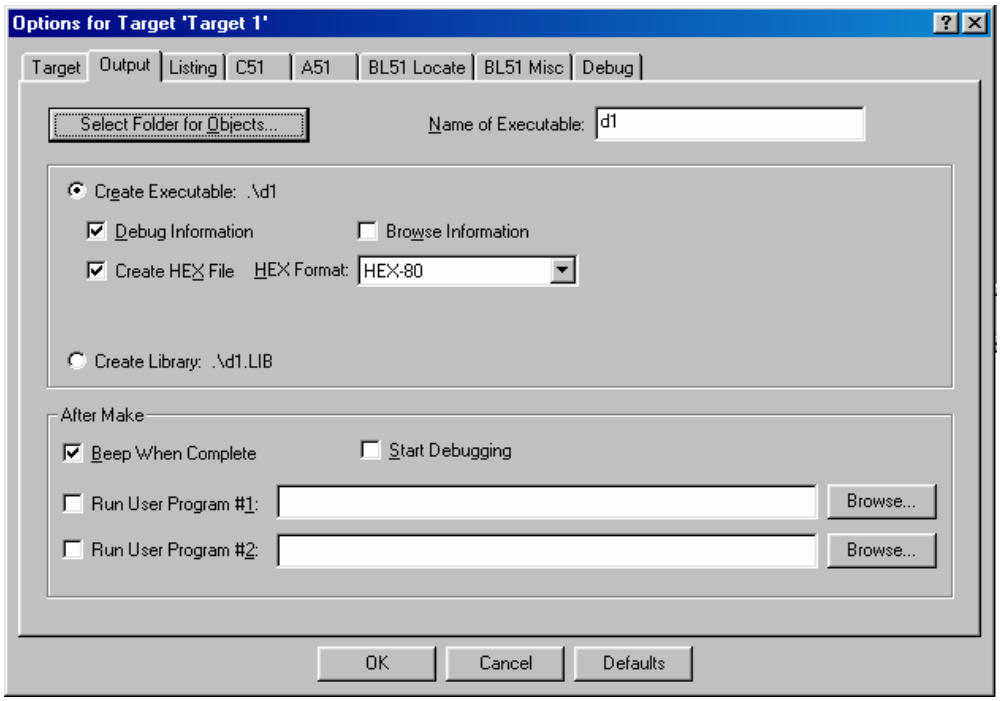
```
char code var1 = 15;            // konstanta var1=15 se nalazi u ROM memoriji
char data var1;
unsigned long xdata array[100];
float idata x,z,y;
unsigned int pdata dimension;
int bdata ibase;
char bdata flags;
sbit flag;
extern uchar BrojacSenzora;     // promenljiva BrojacSenzora se nalazi u nekom drugom C fajlu
                                // koji je takođe uključen u projekat
```

Primeri dodele vrednosti promenljivama:

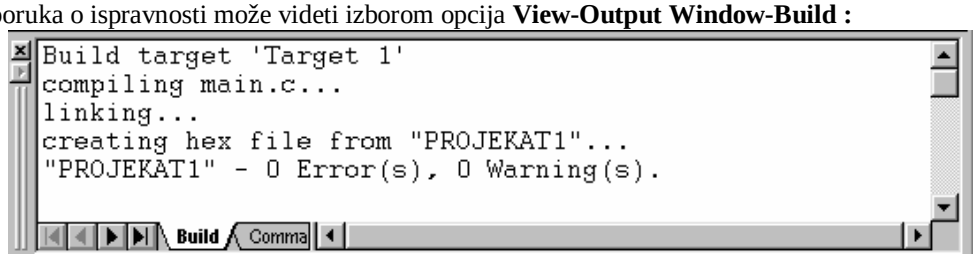
```
flag = 1;
flag = ibase^3;                 // od 16-bitne promenljive ibase prom. flag dobija vrednost
                                // trećeg bita (0 ili 1)
P2=0x03;                        // P2=00000011
EA=1;                           // dozvoljeno je bit adresibilno postavljanje prekida
ET0=1;                          // dozvola prekida prekoračenja Timera0
TMOD=0x01;                      // T0 je 16-bitni Timer
TH0=Timer0H;
TL0=Timer0L;
TR0=1;                          // uključenje Timera 0
```

14. TESTIRANJE PROJEKTA I KREIRANJE HEX FAJLA

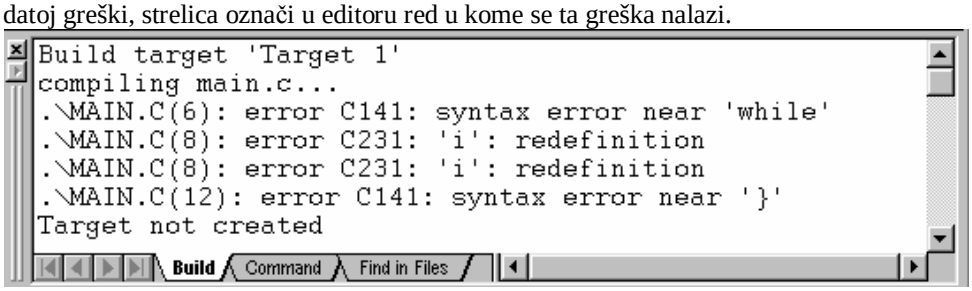
Pošto je projekat napravljen, može se pristupiti testiranju njegove ispravnosti i kreiranju odgovarajućeg HEX fajla, pomoću koga se vrši programiranje mikrokontrolera preko programatora. Prvo se izabere opcija **Project-Options for target 'Target1'-Output** gde se uključi opcija **Create HEX File:**



Zatim se vrši provera ispravnosti unešenog programa u projektu izborom opcije **Project–Build Target,** posle čega se poruka o ispravnosti može videti izborom opcija **View-Output Window-Build :**

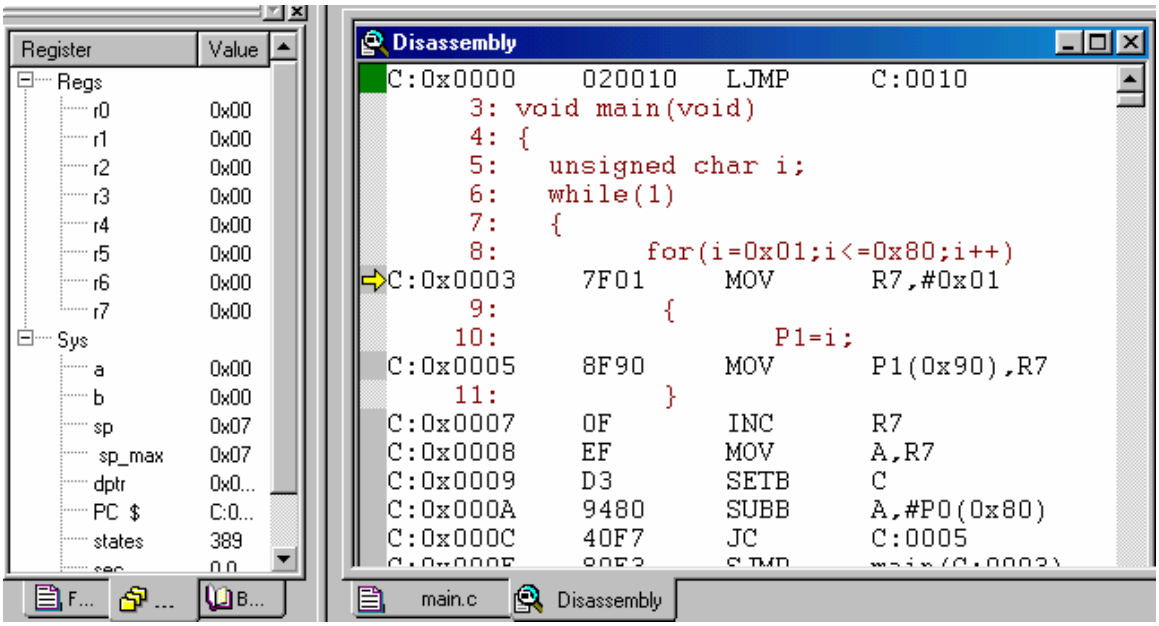


Na gornjoj slici je prikazan program bez grešaka, dok na donjoj postoje 4 greške. Kada se dva puta klikne na poruku o datoj greški, strelica označi u editoru red u kome se ta greška nalazi.

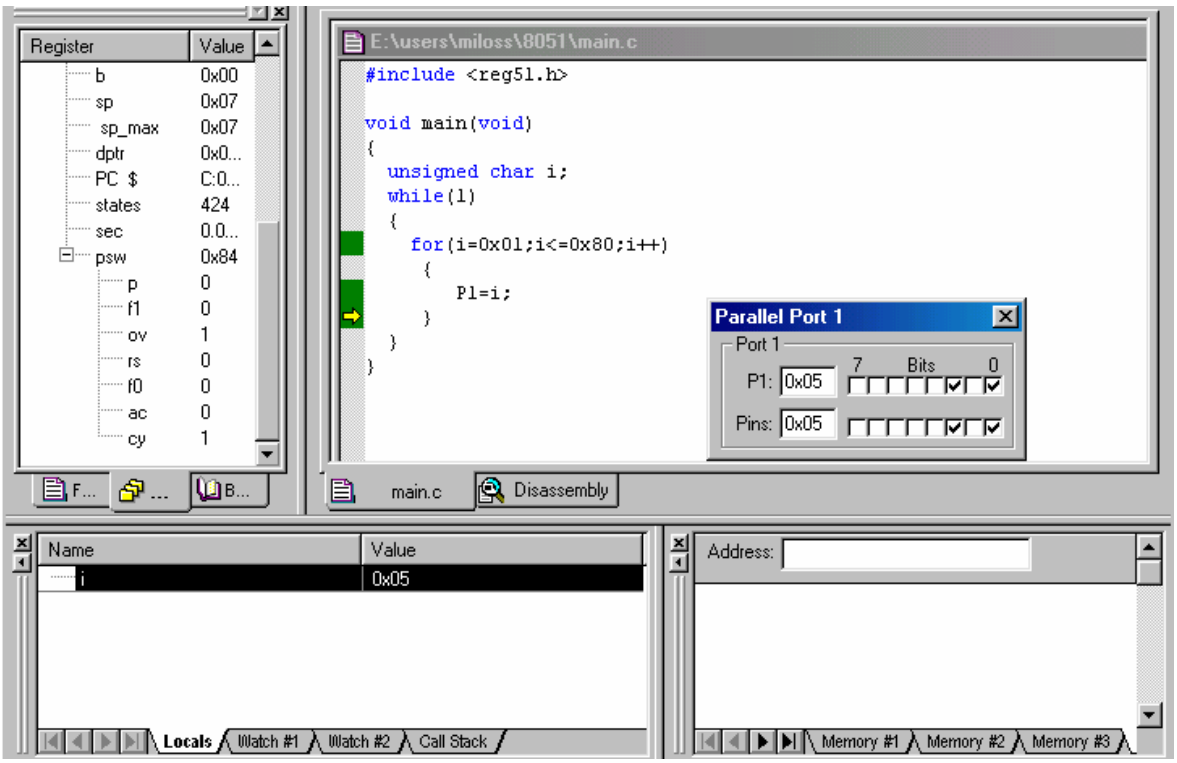


15. SIMULACIJA RADA MIKROKONTROLERA 8051

Tek kada su ispravljene sve greške u programu može se pristupiti simulaciji rada mikrokontrolera. Postoje mnogi programi za simulaciju, a ovde će u kratkim crtama biti opisane mogućnosti m**Vision2 debugger-**a**.** On simulira memorijsku mapu promenljivih, lokalne promenljive i periferije (serijski port, spoljašnje I/O nožice i tajmere). Startovanje se vrši izborom opcije **Debug-Start/Stop Debug Session**. Ako se uključi opcija **View-Disassembly Window** dobija se izvorni kod sa svojim asemblerskim prevodom:



Sa leve strane se vide registri u koje su smeštene promenljive, a takođe se može videti i sadržaj **PSW** registra. Pritiskom na taster **F11** izvršavaju se instrukcije jedna po jedna, pri čemu žuta strelica prati redosled izvršavanja programa. Istovremeno, sadržaj registara se menja zavisno od izvršene instrukcije. Mogu se posmatrati i lokalne promenljive (ovde je to **i**) uključenjem opcije **View-Watch&Call Stack Window**, kao i stanje na portovima izborom opcije **Peripherals-I/0-Ports** (ovde se posmatra samo port P1):



Prekidanje simulacije se vrši izborom opcije **Debug-Stop Running (Debug-Start/Stop Debug Session).** Ovaj vid simulacije je pogodan za proveru redosleda izvršavanja programa, matematičkih izračunavanja i osnovnog prikaza periferija.

16. PRIMERI

Zadatak 3. Uraditi zadatak 1 programiranjem mikrokontrolera 8051 u programskom jeziku C korišćenjem kompajlera C51.

Rešenje:

```
#include <reg51.h>

sbit Port=P1^0;                 // definisanje globalne promenljive

void Inicijalizacija (void)
{
        EA=1;                   // dozvoljeno je bit adresibilno postavljanje prekida
        ET0=1;                  // dozvola prekida prekoračenja Timera0
        TMOD=0x01;              // T0 je 16-bitni Timer
        TH0=0xFE;               // gornji bajt Timera0
        TL0=0x0C;               // donji bajt Timera0
        TR0=1;                  // Timer0-on
        P1=0x00;                // inicijalno stanje porta P1
}

void timer0 (void) interrupt 1 using 2          // prekidni potprogram
{
        TR0 = 0;                // tajmer je potrebno zaustaviti pre punjenja
        TH0 = 0xFE;             // potrebno je ponovo inicijalizovati Timer0 jer on
        TL0 = 0x0C;             // kad jednom odbroji nastavlja da broji od 0
        TR0 = 1;                // tajmer se ponovo startuje
        Port ^= 1;              // invertuje se Port (Port=Port EX-ILI 1)
}

void main(void)                 // glavni program
{
        Inicijalizacija();      // poziv potprograma za inicijalizaciju tajmera0 i porta P0
        while(1)                // glavni program se vrti u ovoj beskonačnoj petlji
        {                       // i čeka na prekide tajmera 0
        }
}
```

Kada nastupi prekid izvršava se sledeći prekidni potprogram:
void timer0 (void) interrupt 1 using 2
Kod ove deklaracije **interrupt 1** označava vrstu prekida (Timer/Counter0), dok **using 2** označava da se svi podaci posle nastupanja prekida čuvaju u registarskoj banci 2. Ako se izostavi **using 2** podaci (registri koji se koriste u funkciji) će se čuvati na steku.
Oznake i adrese prekida:

Interrupt Number	Interrupt Description	Address
0	EXTERNAL INT 0	0003h
1	TIMER/COUNTER 0	000Bh
2	EXTERNAL INT 1	0013h
3	TIMER/COUNTER 1	001Bh
4	SERIAL PORT	0023h

Zadatak 4. Uraditi zadatak 2 programiranjem mikrokontrolera 8051 u programskom jeziku C.
Rešenje:

```
#include <reg51.h>
//def. globalnih promenljivih
unsigned char brojac;                   // brojač prekida
unsigned char frekv;
unsigned char maska;
sbit PR1 = P1^0;
sbit PR2 = P1^1;
sbit CRVENA = P0^0;
sbit ZELENA = P0^1;

void Inicijalizacija (void)
{
        EA = 1;                         // dozvoljeno je bit adresibilno postavljanje interrupta
        ET0 = 1;                        // dozvola interrupta prekoračenja Timera0
        TMOD = 0x01;                    // T0 je 16-bitni Timer
        TH0 = 0x3C;                     // gornji bajt Timera0
        TL0 = 0xB0;                     // donji bajt Timera0
        brojac = 1;
        frekv = 1;
        TR0 = 1;                                // Timer0-on
}

void timer0 (void) interrupt 1 using 2         // prekidni potprogram
{
        TR0 = 0;                        // tajmer je potrebno zaustaviti pre punjenja
        TH0 = 0x3C;                     // potrebno je ponovo inicijalizovati Timer0 jer on
        TL0 = 0xB0;                     // kad jednom odbroji nastavlja da broji od 0
        TR0 = 1;                        // tajmer se ponovo startuje
        if (!(--brojac))
        {
                brojac = frekv;
                P0 ^= maska;
        }
}

void main(void)                                 //glavni program
{
    Inicijalizacija();                  // poziv potprograma za inicijalizaciju tajmera0 i porta P0
    while(1)                            // glavni program se vrti u ovoj petlji i proverava stanje
    {                                   // spoljašnjih prekidača i čeka na prekide tajmera 0
        if (PR1)                        // crvena
        {
                maska = 1;
                ZELENA = 1;
        }
        else                            // zelena
        {
                maska = 2;
                CRVENA = 1;
        }
        if (PR2)                        // 10Hz
                frekv = 1;
        else                            // 0.5Hz
                frekv = 20;
    }
}
```