

Vežba 6.

Rad sa tajmerima i prekidima u C-u.

Promene u odnosu na assembler

U odnosu na assemblyski opis rada tajmera i sistema prekida, opis u C programskom jeziku je znatno prostiji, razumljiviji i pregledniji. Naravno, sa stanovišta optimizacije, assembler znatno bolje koristi ograničene resurse mikrokontrolera. Međutim, današnji kompajleri poseduju sasvim zadovoljavajuće metode za optimizaciju, kako po brzini izvršavanja programa tako i po veličini korisničkog koda.

Način korišćenja mikrokontrolera u procesima koji zahtevaju tačno merenje vremena nameće potrebu da se detaljnije upoznamo sa radom tajmera i prekida u programskom jeziku C. Znanje stečeno kod assemblera u radu sa brojačima i prekidima se aktivno koristi, a razlika postoji samo u izgledu koda.

Opis prekida u C-u

Pošto su ranije obrađeni izvori prekida, ostaje nam samo da se upoznamo sa sintaksom pri njihovom definisanju. Evo kako se definiše eksterni prekid:

```
void extint0() interrupt 0
{
    EA=0;    //zabrana svih prekida
    {
        /* telo rutine za
           obradu prekida */
    }
    EA=1;    //dozvola prekida
}
```

Interrupt 0 je pokazuje da je u pitanju spoljašnji prekid koji smo u ovom slučaju proizvoljno nazvali *extint0()*. Prilikom aktiviranja spoljašnjeg prekida, ulazi se u prekidni program. Ovde je dat primer prekidnog programa čija funkcionalnost nije od interesa i pokazuje sintaksnu opis tela prekidnog programa. Prvo se vrši zabrana svih prekida (EA=0), zatim se izvršava prekidni program, i na kraju dozvoljavamo sve prekide (EA=1). U tabeli je dat pregled svih izvora prekida kod mikrokontrolera 8051 po brojevima, zajedno sa adresama vektora 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

Za merenje vremena kod upravljanja mikrokontrolerima koristimo tajmere. Vrednost tajmera se uvećava nakon svakog mašinskog ciklusa, a mogu, u zavisnosti od konfiguracije, da budu 8-bitni ili 16-bitni. Ako brojač broji u punom opsegu onda možemo da generišemo jednu vremensku bazu. Naime, kada tajmer izvrši preticanje generiše se prekid i na taj način se stvara jedna vremenska baza. Punjenjem brojača sa odgovarajućim početnim vrednostima mogu se generisati različite neophodne vremenske baze. Evo jednog načina reagovanja na prekid usled prekida brojača:

```
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;
        temp=D4;
        D4=D3;
        D3=D2;
        D2=D1;
        D1=temp;
    }
}
```

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.

Razlika u odnosu na spoljašnji prekid je da u ovom slučaju prekidni program treba da obezbedi punjenje brojača na početnu vrednost za generisanje željene vremenske baze. U trenutku punjenja brojač treba zaustaviti da bi se izbeglo nedefinisano stanje brojača. Treba primetiti da ovde nema maskiranja prekida (ne koristi se EA), što znači da ukoliko se dogodi više prekida u istom trenutku ili se dogodi prekid u toku obrade prekidnog programa, u tom slučaju se razmatraju prioriteti izvora prekida. Ukoliko su svi prekidi istog prioriteta, onda je korisno maskirati sve prekide u toku izvršavanja prekidnog programa.

Pre početka izvršavanja programa neophodno je izvršiti inicijalizaciju promenljivih i postaviti parametre tajmera tako da se obezbedi željeni način rada. Ukoliko se ne uradi inicijalizacija, mikrokontroler može ući u nedefinisano stanje koje se ogleda u tome da program ne radi onako kako treba. Uobičajeno je da se pravi funkcija za inicijalizaciju koja se u glavnom programu prva pozove (pre ulaska u beskonačnu petlju). Evo primera funkcije za inicijalizaciju:

```

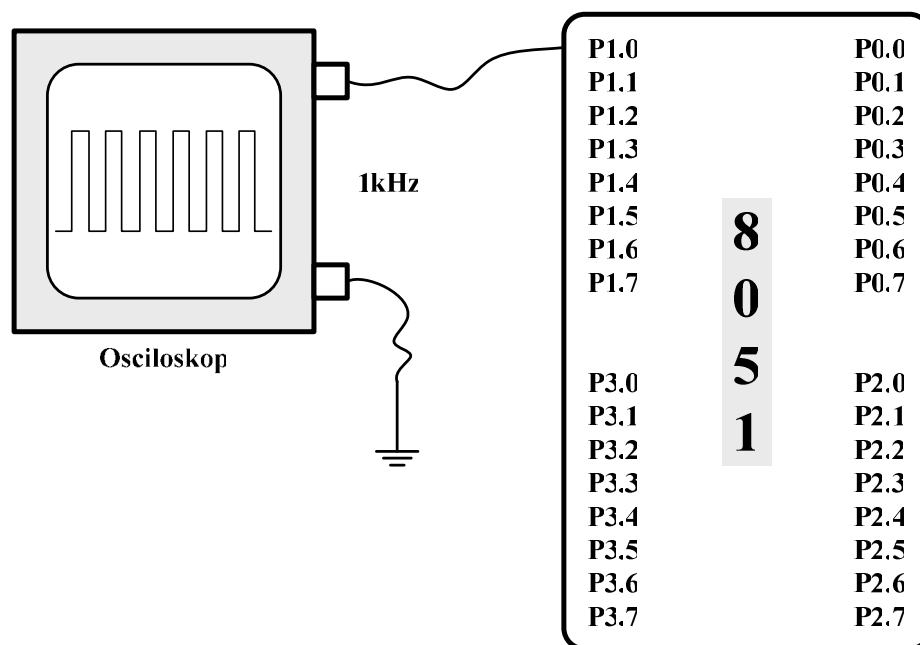
void Inicijalizacija (void)
{
    EA = 1;                // dozvoljeno je bit adresibilno postavljanje
interrupta
    ET0 = 1;               // dozvola interrupta prekoracenja Timera0
    TMOD = 0x01;           // T0 je 16-bitni Timer
    TH0 = 0x3C;            // gornji bajt Timera0
    TL0 = 0xB0;            // donji bajt Timera0
    brojac = 1;
    frekv = 1;
    D1=0x7F;
    D2=0x77;
    D3=0x50;
    D4=0x78;
    TR0 = 1;              // Timer0-on
}

```

U funkciji se definiše ponašanje tajmera 0, definiše se način rada kao i početno punjenje tajmera. U inicijalizaciji se mora takođe pokrenuti tajmer pošto se tajmer 0 ne pokreće automatski po pokretanju programa.

U nastavku, dato je nekoliko zadataka za vežbu. Zadaci se rešavaju programiranjem u programskom jeziku C, korišćenjem programskog paketa *Keil μ Vision 2*.

ZADATAK 1. Generisanje impulsa. Sa mikrokontrolerom 8051, čija je frekvencija spoljašnjeg oscilatora 12MHz realizovati povorku pravougaonih impulsa jednakog trajanja (četvrtki) frekvencije $f=1\text{kHz}$ na portu P1, bit 0 (P1.0) (sl. 1).



Slika 1. Generisanje impulsa mikrokontrolerom

Da ponovimo kako se podešava tajmer. 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:

$$T_{timer} = 12 \cdot T_{osc} = 1 \mu s$$

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 (interrupt):

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

Pošto ovoliki broj ne može da izbroji osmobarbitni 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

```
#include <reg51.h>

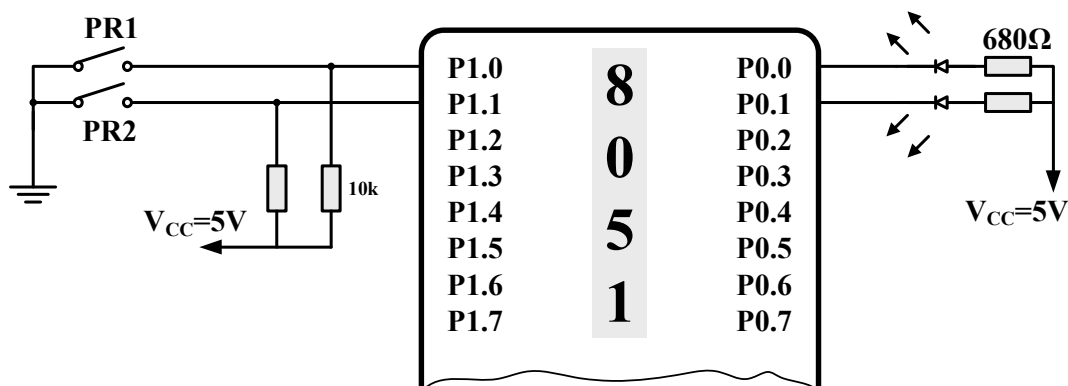
sbit Port=P1^0; // definisanje globalne promenljive

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
    }
}
```

ZADATAK 2. Kontrola rada LED. 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) (sl. 2).



Slika 2. Kontrola dve LED sa prekidačima

Rešenje:

Broj tajmerskih ciklusa koji treba da se odbroji kada je PR2 otvoren je $N1=50ms/1\mu s=50000$, a kada je zatvoren $N2=1s/1\mu s=1000000$. 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 izvrši promena logičkog nivoa signala na nožici P0.0 ili P0.1, za frekvenciju od 0.5Hz.

U Timer0 je potrebno uneti vrednost: $0-50000=0000h-C350=3CB0$.

```
#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;
}
```

```

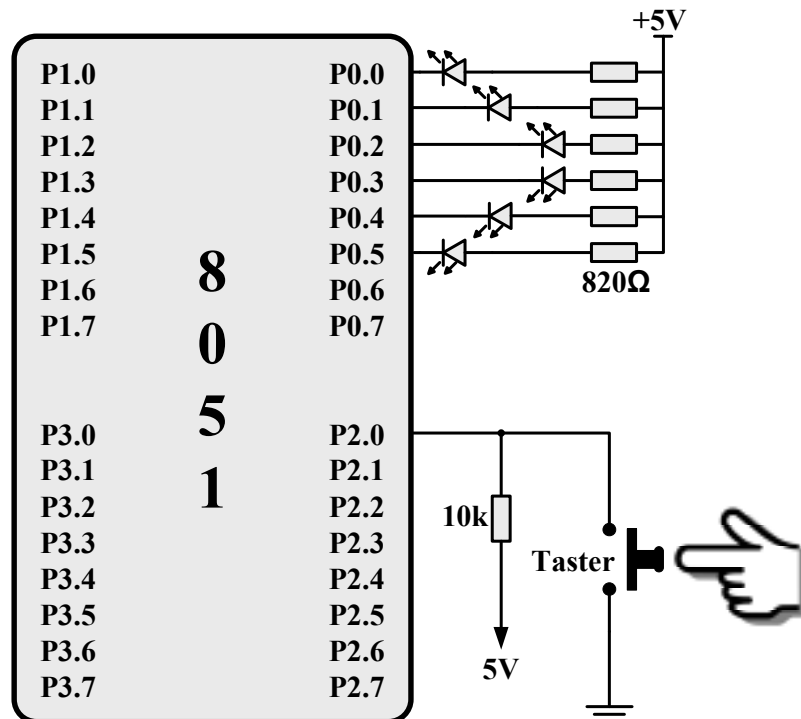
        TR0 = 1;                // Timer0-on
    }

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
    TLO = 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;
    }
}

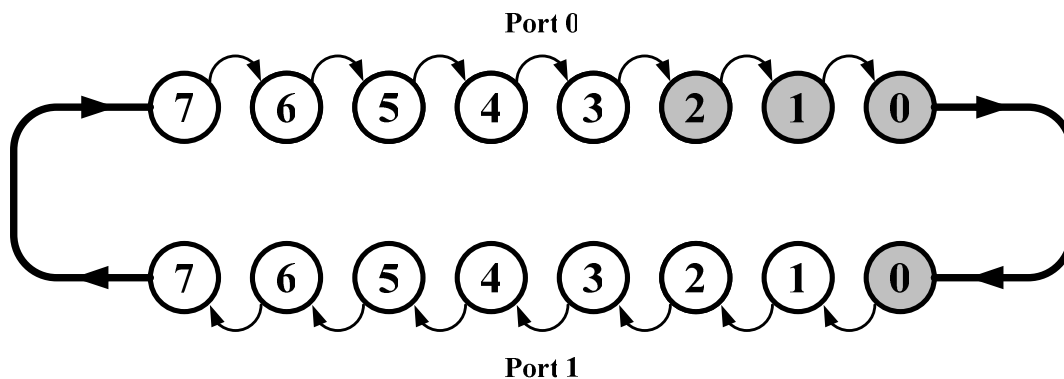
```

ZADATAK 3. Elektronska kockica. Koristeći mikrokontroler Atmel 89C51 radne frekvencije 12MHz kod koga je na port P0 vezano šest LED dioda i na port **P2.0** jedan taster, napisati program koji realizuje elektronsku kockicu (brojanje od jedan do šest). Koristiti timer1, a frekvencija promene brojeva treba da bude 200Hz (sl. 3). Podrazumeva se da je rezultat bacanja kockice jedan ako se uključi LED dioda vezana na nožicu P0.0; rezultat je dva ako se uključi ledovka na nožici P0.1 itd.







Slika 3. Mikrokontroler kao elektronska kockica

ZADATAK 4. Rotacije. Koristeći mikrokontroler Atmel 89C51 radne frekvencije 12MHz na čije portove 0 i 1 je povezano po 8 led dioda. Na ulaz eksternog prekida **EX0** povezan je taster. Napisati program koji vrši kružnu rotaciju svetlosne grupe od 4 upaljene led diode u desno ili u levo. Promena smera rotacije vrši se pritiskom na taster. Ilustracija rotacije grupe od 4 svetleće diode u desno ilustrovana je na slici (sl. 4).



Slika 4. Način rotacije grupe od 4 LED od ukupno 16 LED povezanih na portove 0 i 1

ZADATAK 5. Blink. Koristeći mikrokontroler Atmel 89C51 radne frekvencije 12MHz na čiji je svaki port povezan po jedan 7 segmentni displej (od nultog do šestog bita), a na **P0.7** i **P1.7** su povezana dva prekidača. Napisati program koji vrši treptanje stringa AbCd sa jednom od 4 učestanosti date u tabeli u zavisnosti od položaja prekidača.

Prekidač 1	Prekidač 2	Učestanost [Hz]	
0	0	40Hz	
0	1	20Hz	
1	0	10Hz	
1	1	5Hz	

Slika 5. Frekvencija oscilovanja stringa na displejima u zavisnosti od položaja prekidača