6. Časovnik Timer0 in periodično generiranje prekinitev

Spoznali boste osnovni 8-bitni časovnik. Naučili se ga boste konfigurirati in generirati natančne časovne intervale s pomočjo prekinitev.

Modul **Timer0** je 8-bitni časovnik/števec [2][6][8] z naslednjimi lastnostmi:

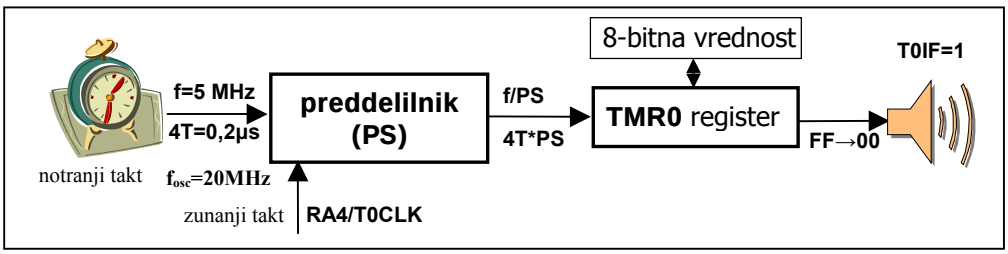
- vsebuje 8-bitni programsko izbirljiv preddelilnik (prescaler PS);
- ima možnost izbire med notranjim (urin takt, npr.: 20 MHz) in zunanjim taktom (na voljo je tudi možnost izbire aktivnega prehoda);
- sproži zastavico (**T0IF**) in (če so prekinitve omogočene) zahtevo za prekinitev (**Interrupt**) ob napolnitvi števca (prehod iz FFh na 00);
- omogoča štetje dogodkov (prehodov signala) na zunanjem priključku **RA4/T0CLK**
- je standardna enota večine Microchip PIC mikrokrmilnikov (npr.: 16F84) srednje kategorije (Mid Range).

Z modulom **Timer0** so povezani registri:

- **TMR0** (01h) – 8-bitni števni register (vpišemo lahko 8-bitno število: 256 vrednosti, register vedno **prišteva** prehode takta do napolnitve, zatem se register inkrementira ali povečuje od začetne vrednosti 00 oz. vpisane vrednosti);
- **OPTION_REG** (81h) – register z biti za nastavitev preddelilnika in kontrolnimi biti timerja;
- **INTCON** (0Bh) – register s kontrolnimi in statusnimi biti prekinitvenih zahtev.

6.1. Splošno o časovniku Timer0

Izbiramo lahko (Slika 6-1) med **notranjim taktom** (četrtina frekvence urinega takta) in **zunanjim taktom**, ki ga pripeljemo na vhod **RA4/T0CLK**.

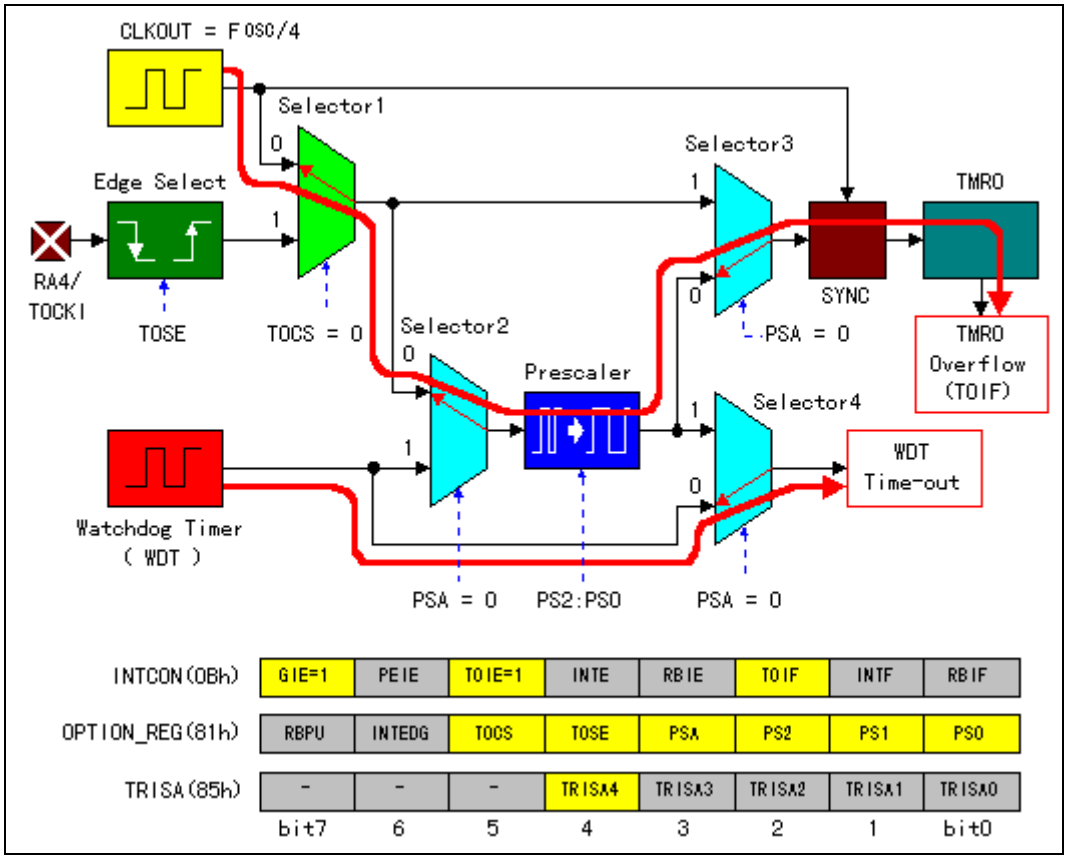


Slika 6-1: Simbolična zgradba časovnika Timer0

Največji možni doseg (interval) časovnika **Timer0** je: (256*256)*0,2 µs=**13,1 ms** (pri izbiri preddelilnika 1:256, začetni vrednosti **TMR0**=0 in taktu 20 MHz). Večjo (ugodnejšo) ločljivost nastavitve časovnega intervala dosežemo pri čim manjši vrednosti preddelilnika.

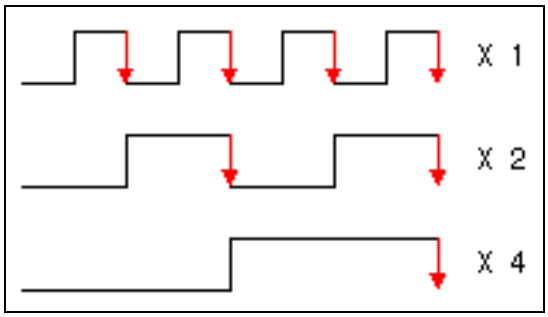
Ob vsaki napolnitvi (prelivu) števca (Slika 6-2) se samodejno aktivira zastavica (**T0IF**=1) in hkrati tudi sproži prekinitvena zahteva (če je omogočena). V prekinitvenem strežnem programu je potrebno programsko zbrisati zastavico (**T0IF**=0), preden ponovno omogočimo prekinitve.

$Timer0_{Interval}$ = (256-**TMR0**) * PS * 4 * T_{osc}, pri čemer je T_{osc} = 50 ns, če je frekvenca urinega takta 20 MHz.

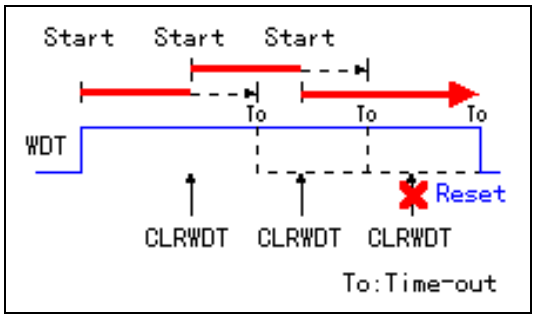


Slika 6-2: Podrobnejši grafični prikaz delovanja časovnika Timer0

Preddelilnik (Prescaler) ima pomembno funkcijo (Slika 6-3), kajti osnovni takt (frekvenco) deli z izbranim faktorjem, kar posredno pomeni povečanje števila bitov števca (dejansko se perioda signala v števec pomnoži z vrednostjo preddelilnika).



Slika 6-3: Delovanje preddelilnika (Prescaler)



Slika 6-4: »Časovni stražnik« (Watch Dog Timer)

Časovni stražnik (Watch Dog Timer) si ekskluzivno deli (en sam) preddelilnik s časovnikom Timer0 (Slika 6-4).

Časovni stražnike sicer uporabljamo v **vgrajenih sistemih**, kjer skrbijo, da se aplikativni programi ne bi zazankali [1][6]. V PIC mikrokrmilnikih ima **WDT** (Slika 6-2) svoj takt (RC oscilator, ki je neodvisen od sistemskega kvarčnega oscilatorja). **Največji interval** do izteka **WDT** je približno **18 ms** (pri preddelilniku 1) in 2,3 s (pri preddelilniku 1:128). Uporabniški program mora pred iztekom **WDT** izvesti namenski ukaz **CLRWDT**, ki zbriše števec. **Če se to ne zgodi, povzroči časovni stražnik RESET mikrokrmilnika.**

6.1.1. Register OPTION_REG

Nahaja se na naslovu 81h oz 01h v segmentu Bank1.

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	**INTEDG**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**
bit7	6	5	4	3	2	1	bit0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bit 7: **$\overline{\text{RBPU}}$**: **PORTB** "Pull-up" upori - izbira (1 – upori niso vključeni, 0 – upori so vključeni)

bit 6: **INTEDG**: izbira prehoda signala za prekinitev preko RB0/INT vhoda (1 – naraščajoči, 0 – padajoči)

bit 5: **T0CS**: izbira takta za **TMR0** (1 – zunanji takt na RA4/T0CKL, **0** – notranji takt iz CLKOUT)

bit 4: **T0SE**: izbira prehoda signala za **TMR0** (1 – padajoči, **0** – naraščajoči na RA4/T0CKL zunanjem taktu)

bit 3: **PSA**: dodelitev preddelilnika (1 – **WDT** "Watch Dog Timer" časovni stražnik1, **0** – **Timer0**)

biti 2-0: **PS2:PS0**: biti za izbiro preddelilnika

PS2,PS1,PS0	TMR0 (PS)	WDT vrednost
0 0 0	1:2	1:1
0 0 1	1:4	1:2
0 1 0	1:8	1:4
0 1 1	1:16	1:8
1 0 0	1:32	1:16
1 0 1	1:64	1:32
1 1 0	1:128	1:64
1 1 1	1:256	1:128

Table 6-1: Vrednosti preddelilnika za TMRO in WDT

6.1.2. Register INTCON

Nahaja se na naslovu 0Bh v vseh segmentih.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	**PEIE**	**T0IE**	**INTE**	**RBIE**	**T0IF**	**INTF**	**RBIF**
bit7	6	5	4	3	2	1	bit0

Legenda: R = omogočeno branje bita, W = omogočeno pisanje v bit, U = neuporabljen bit, beri kot '0', -n = vrednost ob POR resetu

bit 7: **GIE**: Globalni prekinitveni bit (**1** – omogoči vse nemaskirane prekinitve, **0** – onemogoči vse prekinitve)

bit 6: **PEIE**: bit za omogočanje prekinitev periferije (1 – omogoči vse nemaskirane prekinitve periferije, **0** – onemogoči)

bit 5: **T0IE**: omogočanje prekinitev **TMR0** (**1** – omogoči TMR0 prekinitev, 0 – onemogoči ali maskiraj)

bit 4: **INTE**: omogoči zunanjo prekinitev RB0/INT (1 – omogoči, **0** – onemogoči ali maskiraj)

bit 3: **RBIE**: omogoči prekinitve ob spremembi na RB vratih (1 – omogoči, **0** – maskiraj)

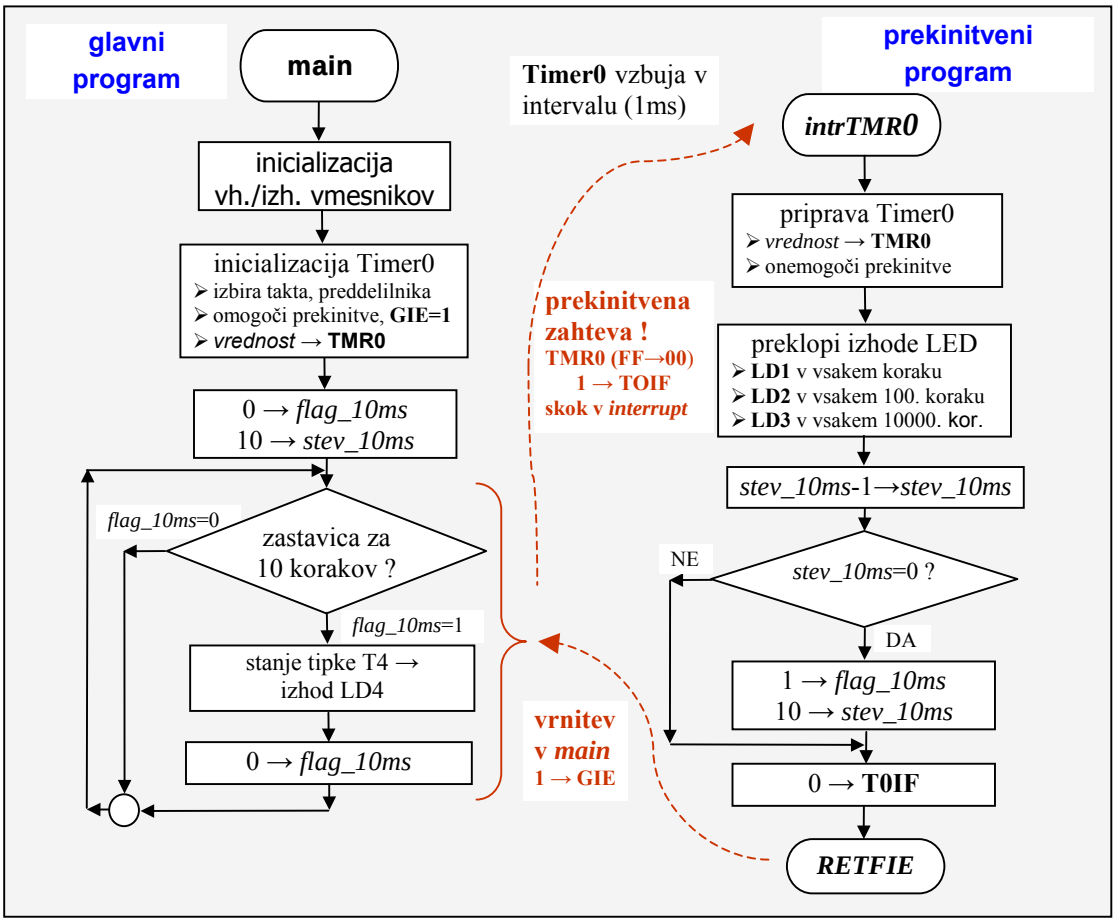
bit 2: **T0IF**: zastavica prekoračitve števca **TMR0** (**1** – prekoračitev ali prehod iz FFh na 0 dosežena, 0 – ni prekoračitve)

bit 1: **INTF**: zastavica prekinitve zaradi RB0/INT vhoda (1 – se je zgodila, 0 – ni)

bit 0: **RBIF**: zastavica prekinitve na RB4 - RB7 vhodih (1 – se je zgodila, 0 – ni)

6.2. Timer0 in prekinitve

Poglejmo si diagram poteka **tipičnega glavnega (*main*)** in prekinitvenega **strežnega programa (*interrupt*)** za periodično generiranje različnih časovnih intervalov (Slika 6-5):



Slika 6-5: Diagram poteka glavnega in prekinitvenega strežnega programa

Ob začetku prekinitvenega strežnega programa (***interrupt***) je priporočljivo (v kolikor ne uporabljamo drugih prekinitvenih virov) **zbrisati bit za omogočanje prekinitev** (**GIE=0**), zato da so med izvajanjem prekinitvenega programa vse prekinitve blokirane in s tem ni nevarnosti ponovnega aktiviranja prekinitvene zahteve, še preden se strežni program konča. Prekinitveni strežni program mora biti **vedno zaključen z ukazom RETFIE**, ki povzroči vrnitev v prej prekinjeni program in ponovno omogočanje prekinitev (ob tem se samodejno postavi bit **GIE=1**).

Programske koda algoritma v zbirnem jeziku (Slika 6-5) je nekoliko daljša, zato je v nadaljevanju prikazan pripadajoč program le v izvorni kodi C-jezika (Hi-TECH PICC Lite).

Izvorni program v C-jeziku, ki ustreza kodiranju algoritma (Slika 6-5):

```
// Demo program - utripanje LED diod: LD1 na 1ms, LD2 na 100ms, LD3 na 10s
// stanje tipke T4 (RC4 -> RA4) se prenese na LD4 (Ra5) vsakih ~10ms
//    avtor: Janez Pogorelc, 3.5.04

#include <pic1687x.h>

// definicija konstant (macro)
#define Timer0  100  //zacetna vrednost za TMR0
#define STO_ms  100  //konstanta 100 za stevec - 100ms
#define DESET_s 100  //konstanta 100 za stevec - 10s

// deklaracija globalnih spremenljivk
unsigned char flag_10ms, stev_10ms, sto_ms, deset_s;

// --------- prekinitveni strezni program se izvaja v intervalu 10ms
void interrupt intTMR0(void)
{
TMR0=Timer0;  // vrednost stevca: 100=256-156, int: 156*32*0,2us=998,4us ~1ms
GIE=0;        // blokiraj vse prekinitve med izvajanjem strezbe

// LD1 preklopi v vsakem prekin. intervalu (1ms)
RC0=!RC0;     // izmenicni vklop LED diode LD1 na RC0

// LD2 preklopi na vsakih 100ms
  sto_ms--;   // dekrement stevca ms
  if (sto_ms==0)
  {
  RC1=!RC1;   // izmenicni vklop LED diode LD2 na RC1
  sto_ms=STO_ms;     //zacetna vrednost stevca za 100ms
  // ---- LD3 preklopi na vsakih 10s
       deset_s--;    // dekrement stevca s
       if (deset_s==0)
       {
       RC3=!RC3;     // izmenicni vklop LED diode LD3 na RC3
       deset_s=DESET_s; //zacetna vrednost stevca za 10s
       }
  }

  // postavitev zastavice za 10ms v vsakem 10. koraku
  stev_10ms--;// dekrement stevca
  if(stev_10ms==0) //ali je 10 korakov ?
  {
  flag_10ms=1;// postavitev zastavice
  stev_10ms=10;      // zacetna vrednost stevca ms
  }

T0IF=0;       // brisi zastavico Timer0 prekinitev
} //------- konec prekinitv. streznega programa


// ---------------- glavni program
void main(void)
{

// deklaracija lokalnih spremenljivk
char tmpPORTC;

// --------- Inicializacija registrov vhodno/izhodnih vmesnikov
ADCON1=0x06;  // 0b00000110, ne uporabljamo analognih vhodov
TRISA=0x10;   // 0b00010000, vsi pini porta A so izhodni, razen RA4
```

```
TRISB=0;      // vsi pini porta B so izhodni
TRISC=0;      // vsi pini porta c so izhodni
PORTC=0;      // izhode porta C postavi na 0
PORTA=0;      // izhode porta A postavi na 0

// inicializacija TMR0
OPTION=0x04;  // 0b00000100, notranji takt, PSA=32,
INTCON=0xA0;  // 0b10100000, GIE=1, T0IE=1,
TMR0=Timer0;  // vrednost stevca: 100=256-156, int: 156*32*0,2us=998,4us ~1ms

sto_ms=STO_ms;// zacetna vrednost stevca za 100ms
deset_s=DESET_s; // zacetna vrednost stevca za 10s
flag_10ms=0;  // brisanje zastavice
stev_10ms=10; // zacetna vrednost stevca ms

// -------- jedro programa

  while(1)    // neskoncna zanka
  {

       if (flag_10ms==1)     // ali je zastavica za 10. korak
       {
  //------ vklop LD4 ob pritisku na T4, izvede se vsakih 10ms
       tmpPORTC=PORTC; // shrani trenutno vrednost PORTC
       // testiraj T4
       PORTC=(PORTC&0x0f)|0X10; // postavi 1 na RC4, ohrani RC0 do RC3
       RA5=RA4;              // stanje vhoda RA4 je enako stanju T4
       PORTC=(PORTC&0x0F)|(tmpPORTC&0xF0); //ohrani PORTC in stanja LD

       flag_10ms=0;  // brisanje zastavice
       }
  }
}
// dolzina programa (brez optimizacij!): 126 Words, pomnilnik: 8 Bytes
// interval prekinitev: 5.002c (1000,4us)
// trajanje prekinitv. str. programa (min.): 23c (4,6us), (maks.): 36c (7,2us)
// trajanje zanke v main(): 132c (26,4us)
// izvajanje programa po Resetu do main(): 53c (10,6us)
// stevilo vrstic (brez komentarjev): 44 vrstic
```

Program 6-1: Popoln zgled programa v C-jeziku za prekinitve Timer0

Prekinitveni strežni program naj bo v splošnem čim krajši in učinkovitejši (brez zank). V kolikor so omogočeni (razen Timer0) tudi drugi prekinitveni viri (npr.: vhod **RB0/INT**, vhodi na **PORTB** vratih **RB4** do **RB7**, Timer1, Timer2, A/D pretvornik, programiranje EEPROM lokacij, ...), je potrebno v začetku prekinitvenega strežnega programa (programsko) preveriti (angl.: polling), katera enota je sprožila prekinitveno zahtevo. Namreč, vse enote (možnih je 13 različnih virov) sporočajo status o dogodku – »prekinitvena zahteva« v registrih: **OPTION_REG**, **PIR1**, **PIE2** in **PIR2**.

Izjemno **pomembno je, da programsko poskrbimo za »kontekst«** - vsebina registrov **W** in **STATUS** v glavnem programu. **Poskrbeti moramo, da registra ohranita vrednost tudi ob vrnitvi v glavni program**.

Strukturo »glavnega« in »prekinitvenega« programa (izpis ni primeren za izvajanje, ker ni popoln!) prikazuje zgled izvornega programa v zbirnem jeziku (Program 6-2).

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

```
list   p=16f876      ; izbira tipa čipa
       include <p16f876.inc>; vključitev datoteke z definicijami simbolov registrov

BCKW   equ    0x20   ; definicija naslova spremenljivke - rezerva (backup za shranj. W)
BCKST  equ    0x21   ; definicija naslova spremenljivke - (backup za shranj. STATUS)

;-------------------------------------------------------------------------------------
; Tukaj nastavim Reset vektor
;-------------------------------------------------------------------------------------
       org    0x0    ; RESET vektor
       goto   Main   ; skok na glavni program, Main
       org    0x4    ; prekinitveni vektor, Interrupt
       goto   Int    ; skok v prekinitveno rutino - Int
       org    0x5    ; začetek programske kode

;-------------------------------------------------------------------------------------
; Prekinitveni program
;-------------------------------------------------------------------------------------
RefDisp
       ;---------------------------------------------------------------------------
              ; Vpiši program, ki se izvede ob prekinitvi - časovno zahtevne operacije
       ;---------------------------------------------------------------------------
       return

;-------------------------------------------------------------------------------------
; Prekinitvena rutina
;-------------------------------------------------------------------------------------
Int                          ; Shranjevanje stanja delovnega in statusnega registra
       movwf  BCKW           ; backup za delovni register W
       swapf  STATUS, W      ;
       clrf   STATUS         ; brisanje statusnega registra
       movwf  BCKST          ; backup za STATUS
       ;---------------------------------------------------------------------------
       ; Prekinitveni program
       ;---------------------------------------------------------------------------
       call   RefDisp        ; skok v uporabnikovo rutino prekinitvenega programa
       bcf    INTCON, T0IF   ; prekinitveni program je bil izveden
       ;---------------------------------------------------------------------------
                             ; Vzpostavitev starega stanja
       swapf  BCKST, W       ;
       movwf  STATUS         ; vzpostavi stari STATUS
       swapf  BCKW, F        ;
       swapf  BCKW, W        ; vzpostavi staro stanje delovnega registra W
       retfie

;-------------------------------------------------------------------------------------
; Glavni program
;-------------------------------------------------------------------------------------
Main
       ;---------------------------------------------------------------------------
                     ; Vpiši ukaze za inicializacijo (vhodno/izhodna enota, ...)
       ;---------------------------------------------------------------------------
       ;--------------------; Konfiguracija OPTION_REG (za prekinitve)
       bcf    STATUS,RP1     ;
       bsf    STATUS,RP0     ; BANK 1
       movlw  0x3            ; preddelilnik 1/16, pull-up vklopljeni, notranji pulzi
       movwf  OPTION_REG     ; inicializacija Timer0
       movlw  B'10100000'    ; nastavitev prekinitev ( GIE  = 1 , T0IE = 1 )
       movwf  INTCON         ; omogočitev prekinitev Timer0
       bcf    STATUS,RP0     ; BANK 0
       ;---------------------------------------------------------------------------
       ;---------------------------------------------------------------------------
                             ; Vpiši glavni program - časovno nezahtevne operacije
```

```
        ;-------------------------------------------------------------------------------
        end                     ; konec programa (psevdoukaz)
;-------------------------------------------------------------------------------
```

Program 6-2: Zgled ogrodja programa v zbirnem jeziku pri uporabi prekinitev

Za razumevanje tematike je potrebno tudi predznanje iz osnov programiranja in številskih sistemov. Dodatna znanja iz programiranja v zbirnem jeziku so v: [2][4][6][7]. Za razumevanje programov, ki so kodirani v C-jeziku, je potrebno poznavanje osnov ANSI C strukturnega programiranja: [15][11][13][5]. Veliko rešitev praktičnih nalog z uporabo PIC mikrokrmilnikov je na spletnem portalu http://www.interq.or.jp/japan/se-inoue/e_pic6.htm. Podrobnejši opis delovanja časovnika Timer0 s primeri uporabe v PIC mikrokrmilnikih je na voljo v: [8][16][17][18].

Vprašanja za utrjevanje:

Napišite podprogram iz imenom InTMR0 za inicializacijo časovnika Timer0, ki bo periodično prožil prekinitve na 0,2 ms.

Rešitev:

```
InTMR0 bsf     STATUS,RP0      ; 1 -> RP0 , izbira Bank1 za dostop do OPTION_REG
       movlw   b'00000001'     ;priprava vrednosti 0x01 za OPTION_REG
       movwf   OPTION_REG      ;preddelilnik casovnika Timer0 nastavimo na 4:1
       bcf     STATUS,RP0      ; 0 -> RP0, nazaj v Bank0
       movlw   b'10100000'     ;priprava vrednosti 0xA0 za INTCON, GIE=1, T0IE=1
       movwf   INTCON          ;Omogocimo prekinitve, ki jih sproza casovnik Timer0
       movlw   d'6'            ;priprava vrednosti (desetisko) 6 za TMR0
       movwf   TMR0            ;vrednost intervala: 0,2*(256-6)*4=200us=0,2ms
       return                  ;konec podprograma
```

1. *Napišite podprogram z imenom InTMR0_2 za inicializacijo časovnika Timer0, ki bo periodično prožil prekinitve na 2,0 ms.*
2. *Koliko bitni je števec TMR0 in kakšne vrednosti preddelilnika lahko izbiramo ?*
3. *Kakšna je funkcija časovnega stražnika (Watch Dog Timer) ?*
4. *Ali lahko Timer0 uporabljamo, ne da bi prožil prekinitve ?*
5. *Kakšen interval bi dosegli v zgornjem rešenem zgledu, če ne bi vpisali nobene vrednosti v register TMR0 ?*