Laborator 5

Lucrul cu canale de timp

Obiectiv

Deprinderea modului de programare a canalelor de timp.

1. Introducere

Canalele de timp sunt periferice de baza ale oricarui microcontroller. In familia dsPIC33F/PIC24H sunt incluse mai multe canale de timp ce opereaza pe 16 biti. Microcontrollerul dsPIC33FJ32MC302 contine 5 astfel de module.

Aceste canale sunt clasificate in 3 categorii :

- Canalul de timp de tip A (Timer 1);
- Canalul de timp de tip B (Timer 2, Timer 4);
- Canalul de timp de tip C (Timer 3, Timer 5).

Un canal de timp de tip B poate fi concatenat cu unul de tip C pentru a forma un timer pe 32 biti.

Fiecare canal de timp este un temporizator/numarator pe 16 biti ce contine urmatoarele registre :

- Registrul de numarare (TMRx);
- Registrul de perioada (PRx);
- Registrul de control pentru configurarea timer-ului (TxCON).

Fiecarui timer ii sunt asociati si urmatorii biti de control ai intreruperilor :

- Bitul de validare a intreruperii (TxIE din registrul IEC0);
- Indicatorul de intrerupere (TxIF din registrul IFS0);
- Bitii de control a prioritatii intreruperii (TxIP<2:0> din registrul IPCx).

Canalele de timp functioneaza ca temporizatoare avand frecventa derivata de la ceasul intern (FCY), iar ca numaratoare avand frecventa derivata de la o sursa de ceas extern primita la pinul TxCK.

2. Canalul de timp de tip A

Canalul de timp de tip A sau Timer 1 este un timer cu scop general ce permite generarea de temporizari si de intreruperi periodice, masurarea unor intervale de timp sau contorizarea unor evenimente externe.

Fata de celelalte timere, Timer-ul 1 poate fi pilotat de catre oscilatorul de joasa frecventa de 32kHz sau poate functiona ca numarator asincron folosind o sursa de ceas externa.

O proprietate importanta a acestui timer este ca poate fi folosit in aplicatii ca ceas de timp real.

Diagrama bloc a Timer-ului 1 este prezentata in Fig. 1.

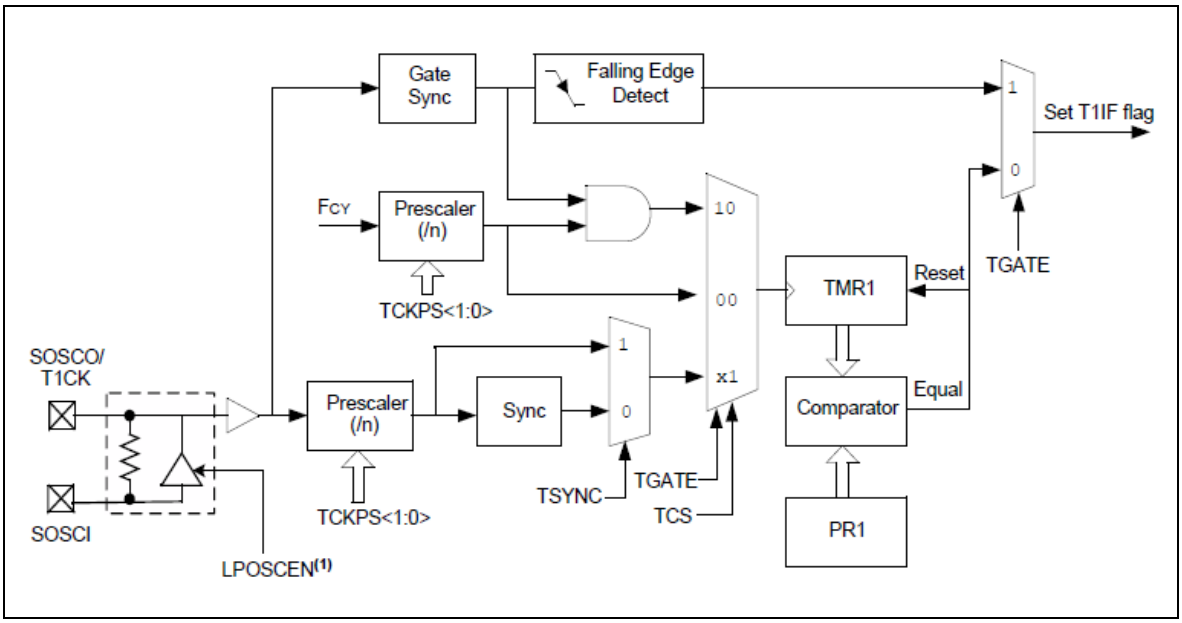

Fig. 1 : Structura circuitelor de configurare pentru ceasul „Timer 1”

Timer-ul 1 poate fi configurat sa functioneze ca temporizator sau ca numarator sincron sau asincron, ii poate fi asociat si un prescaler pentru divizarea tactului initial, poate functiona in starea de SLEEP sau IDLE a microcontrollerului si genereaza o cerere de intrerupere cand valorile din registrii TMR1 si PR1 sunt egale.

Modurile de functionare ale Timer-ului 1 pot fi selectate prin intermediul a 3 biti din registrul de control T1CON :

- Bitul de control al sursei de ceas (TCS);
- Bitul de control al sincronizarii (TSYNC);
- Bitul de control al modului „gated” (TGATE).

Moduri de configurare a ceasului Timer 1 :

Mod	TCS	TGATE	TSYNC
Temporizator	0	0	x
Gated Timer	0	1	x
Numarator sincron	1	x	1
Numarator asincron	1	x	0

Exemplu de configurare a Timer-ului 1 ca temporizator:

```
T1CON = 0;                  // Resetare Timer 1
IFS0bits.T1IF = 0;          // Resetare flag de intrerupere
IPC0bits.T1IP = 1;          // Setare nivel de prioritate
IEC0bits.T1IE = 1;          // Validare intrerupere
T1CONbits.TCS = 0;          // Selectare ceas intern
```

```
T1CONbits.TGATE = 0;            // Nu se valideaza modul „gated”
T1CONbits.TCKPS = 0b00;         // Selectare prescaler 1:1
TMR1=  0x0000;                  // Resetare numarator
PR1 = 10000;                    // Setare valoare perioada de numarat
T1CONbits.TON = 1;              // Pornire ceas
```

```
/* Rutina de tratare a intreruperii pentru Timer 1 */
void __attribute__ ((interrupt, no_auto_psv)) _T1Interrupt( void )
{
    /* Codul rutinei de tratare a intreruperii */
    IFS0bits.T1IF = 0;                  // Resetare flag de intrerupere
}
```

3. Canalele de timp de tip B si de tip C

Timer 2 si Timer 4 sunt canale de timp de tip B si pot fi concatenate cu un canal de tip C pentru a forma un timer pe 32 biti.

Pentru aceste timere semnalul de ceas extern de la bitul TxCK este intotdeauna sincronizat cu semnalul de ceas intern, sincronizare ce se realizeaza dupa divizarea cu prescaler-ul.

Diagrama bloc pentru canalele de tip B este prezentata in Fig. 2.

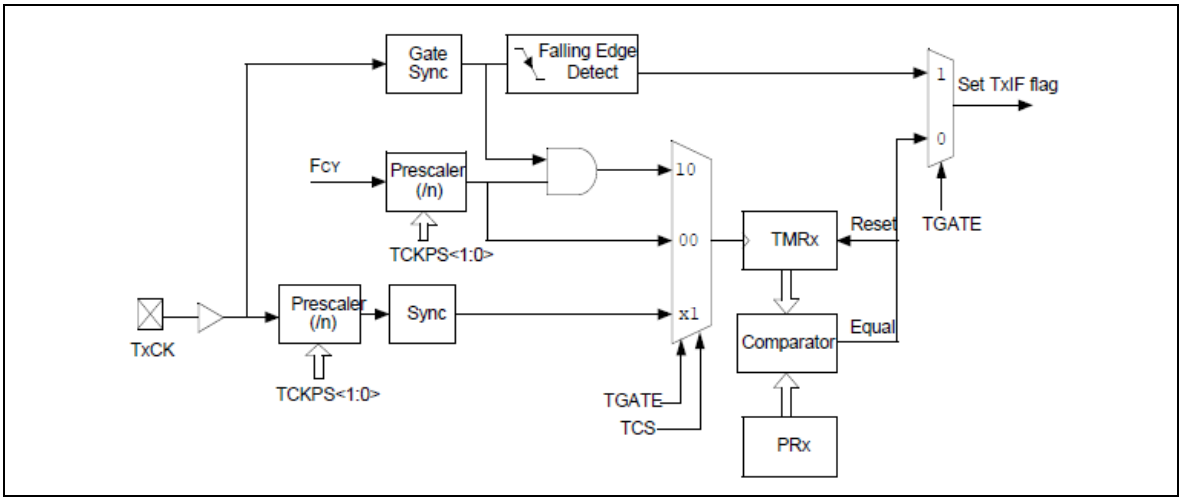

Fig. 2 : Structura circuitelor de configurare pentru ceasul de tip B (x = 2 sau 4)

Timer 3 si Timer 5 sunt canale de timp de tip C si pot fi concatenate cu un canal de tip B pentru a forma un timer pe 32 biti.

Pentru timerele de tip C semnalul de ceas extern de la bitul TxCK este intotdeauna sincronizat cu semnalul de ceas intern, sincronizare ce se realizeaza inainte de divizarea cu prescaler-ul. Totodata cel putin un canal de tip C poate realiza conversia A/D.

Diagrama bloc pentru canalele de tip C este prezentata in Fig. 3.

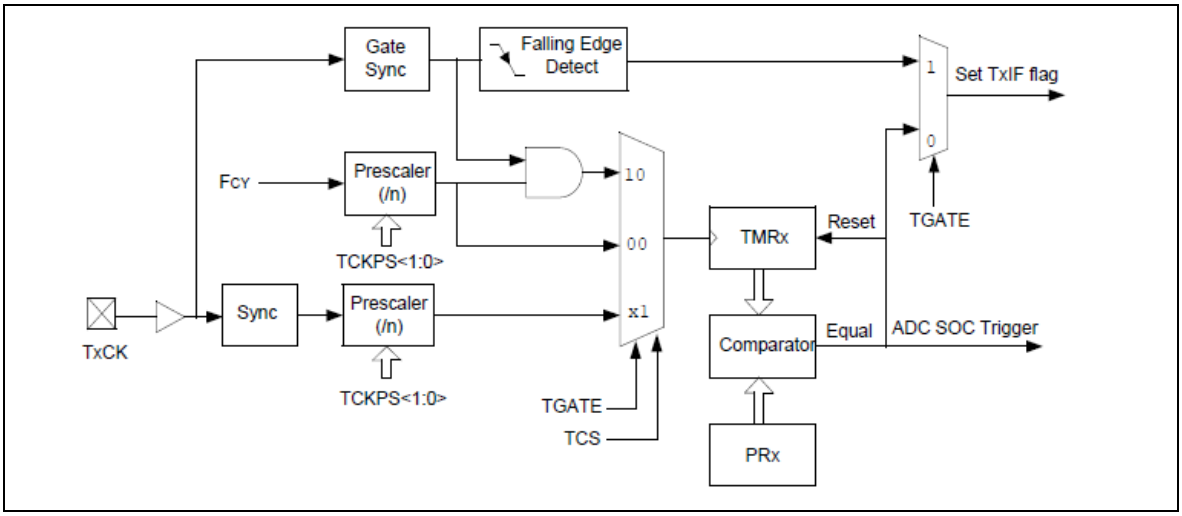

Fig. 3 : Structura circuitelor de configurare pentru ceasul de tip C (x = 3 sau 5)

Atat timerele de tip B cat si cele de tip C pot fi configurate sa functioneze ca temporizator sau ca numarator sincron dupa cum este prezentat in tabelul de mai jos.

Moduri de configurare a ceasurilor Timer 2/3 si Timer 4/5 :

Mod	TCS	TGATE
Temporizator	0	0
Gated timer	0	1
Numarator sincron	1	x

4. Canalul de timp pe 32 biti

Canalul de timp pe 32 biti este rezultatul concatenarii unui timer de tip B cu unul de tip C. Modurile in care se poate realiza aceasta concatenare sunt alaturarea Timer-elor 2 si 3, respectiv a Timer-elor 4 si 5. De obicei se utilizeaza prima varianta pentru obtinerea unui ceas pe 32 biti.

Acest canal poate functiona ca temporizator sau ca numarator sincron si poate realiza o conversie A/D, sau utiliza un prescaler pentru divizarea tactului initial, poate functiona in starea de IDLE a microcontrollerului, sau poate genera o cerere de intrerupere atunci cand registrul concatenat TMR3/TMR2 are aceeasi valoare cu cea a registrului setat initial PR3/PR2.

Pentru operatiile pe 32 biti bitul de control T32 (TxCON<3>) al timer-ului de tip B trebuie setat. Canalul de tip B contine cei mai putin semnificativi biti pentru operatiile pe 32 biti, in timp ce canalul de tip C ii contine pe cei mai semnificativi.

Pentru configurarea acestui canal sunt necesari doar bitii din registrul de control al timer-ului de tip B, cei ai timer-ului de tip C sunt ignorati cu exceptia bitului pentru selectarea starii de IDLE, TSIDL (TyCON<13>).

Pentru controlul intreruperilor sunt utilizati doar bitii de control ai timer-ului de tip C. Bitii de control ai intreruperilor si cei de stare ai ceasului de tip B sunt ignorati in timpul operatiilor pe 32 biti.

Diagrama bloc a canalului pe 32 biti este prezentata in Fig. 4.

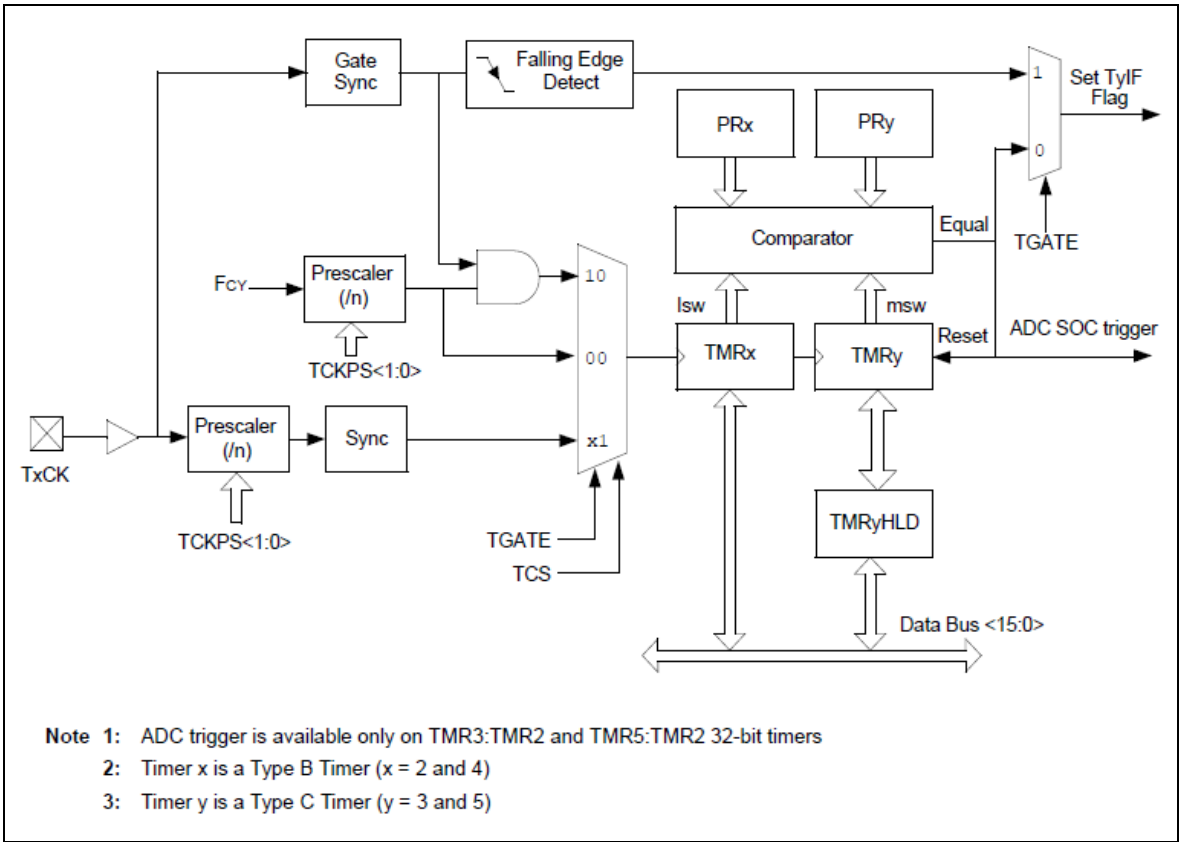

Fig. 4 : Structura circuitelor de configurare pentru ceasul pe 32 biti

Exemplu de configurare a timer-ului pe 32 biti ca temporizator :

```
T3CONbits.TON = 0;          // Oprire operatii pe 16 biti a Timer-ului 3
T2CONbits.TON = 0;          // Oprire operatii pe 16 / 32 biti a Timer-ului 2
T2CONbits.T32 = 1;          // Validare mod pe 32 biti
T2CONbits.TCS = 0;          // Selectare ceas inter
T2CONbits.TGATE = 0;        // Nu se valideaza modul „gated”
T2CONbits.TCKPS = 0b00;     // Selectare prescaler 1:1
TMR3 = 0x0000;              // Resetare numarator (msw)
TMR2 = 0x0000;              // Resetare numarator (lsw)
PR3 = 0x0001;               // Setare valoare perioada de numarat (msw)
PR2 = 0x0000;               // Setare valoare perioada de numarat (lsw)
IPC2bits.T3IP = 0b001;      // Setare nivel de prioritate
IFS0bits.T3IF = 0;          // Resetare flag de intrerupere
IEC0bits.T3IE = 1;          // Validare intrerupere
```

```
    T2CONbits.TON = 1;          // Pornire ceas


/* Rutina de tratare a intreruperii pentru Timer 3 */
void __attribute__ ((interrupt, no_auto_psv)) _T3Interrupt( void )
{
    /* Codul rutinei de tratare a intreruperii */
    IFS0bits.T3IF = 0;          // Resetare flag de intrerupere
}
```

Exemplul 1

Aplicatie ce aprinde pe rand unul dintre cele 4 leduri la un interval de aproximativ 1 secunda utilizand un temporizator pe 16 biti.

Mod de calcul :

FCY = FOSC/2

*T = PR1 * 1 / (FCY / Prescaler)*

FOSC = 7.37 MHz
T = 1 s
Prescaler = 256
} => *PR1 ≈ 14740*

Cod program :

```
#if defined(__dsPIC33F__)
#include "p33fxxxx.h"
#elif defined(__PIC24H__)
#include "p24hxxxx.h"
#endif

_FWDT(FWDTEN_OFF);                  // Watchdog Timer controlat prin soft

int led = 0;

void Comutare_led()
{
        switch(led)
        {
                case 1 : PORTB = 0xE000;
                         break;
                case 2 : PORTB = 0xD000;
                         break;
```

```
                case 3 : PORTB = 0xB000;
                        break;
                case 4 : PORTB = 0x7000;
                        led = 0;
                        break;
        }
}

void Init_Timer1( void )
{
        T1CON = 0;
        IFS0bits.T1IF = 0;
        IPC0bits.T1IP = 1;
        IEC0bits.T1IE = 1;
        T1CONbits.TCS = 0;
        T1CONbits.TGATE = 0;
        T1CONbits.TCKPS = 0b11;          // Selectare prescaler 1:256
        TMR1=  0x0000;
        PR1 = 14740;
        T1CONbits.TON = 1;
}

void __attribute__ ((interrupt, no_auto_psv)) _T1Interrupt( void )
{
        led++;
        Comutare_led();
        IFS0bits.T1IF = 0;               // Resetare flag de intrerupere
}

int main( void )
{
        RCONbits.SWDTEN = 0;             // Dezactivare Watch Dog Timer
        TRISB = 0x0000;                  // Setare PORTB ca iesire
        LATB = 0x0000;                   // Setare valoare latch-uri
        Init_Timer1();                   // Configurare Timer 1
        while(1)
        {
        }
        return 0;
}
```

Exemplul 2 :

Aplicatie ce aprinde si inchide toate ledurile la un interval de aproximativ 0.25 secunde utilizand un temporizator pe 32 biti.

Mod de calcul :

FCY = FOSC/2

*T = (PR3*65536 + PR2) * 1 / (FCY / Prescaler)*

FOSC = 7.37 MHz
T = 0.25 s
Prescaler = 8
} => *PR3*65536 + PR2=115156*
PR3 = 0x0001
PR2 ≈ 49620

Cod program :

```
#if defined(__dsPIC33F__)
#include "p33fxxxx.h"
#elif defined(__PIC24H__)
#include "p24hxxxx.h"
#endif

_FWDT(FWDTEN_OFF);                  // Watchdog Timer controlat prin soft

int valport;

void Init_Timer32( void )
{
    T3CONbits.TON = 0;
    T2CONbits.TON = 0;
    T2CONbits.T32 = 1;
    T2CONbits.TCS = 0;
    T2CONbits.TGATE = 0;
    T2CONbits.TCKPS = 0b01;         // Selectare prescaler 1:8
    TMR3 = 0x0000;
    TMR2 = 0x0000;
    PR3 = 0x0001;
    PR2 = 49620;
    IPC2bits.T3IP = 0b001;
    IFS0bits.T3IF = 0;
    IEC0bits.T3IE = 1;
    T2CONbits.TON = 1;
}

void __attribute__ ((interrupt, no_auto_psv)) _T3Interrupt( void )
{
```

```
    valport = PORTB;
    valport = valport ^ 0xF000;          // Complementare valoare PORTB
    PORTB = valport;
    IFS0bits.T3IF = 0;                   // Resetare flag de intrerupere
}

int main(void)
{
    RCONbits.SWDTEN = 0;                 // Dezactivare Watch Dog Timer
    TRISB = 0x0000;                      // Setare PORTB ca iesire
    LATB = 0x0000;                       // Setare valoare latch-uri
    valport = PORTB;
    valport = valport & 0x0FFF;          // Utilizare biti cei mai semnificativi
    PORTB = valport;
    Init_Timer32();                      // Configurare timer pe 32 biti
    while(1)
    {
    }
    return 0;
}
```

Probleme propuse :

1. Sa se modifice Exemplul 1 astfel incat sa se obtina si alte intervale semnificative de timp.
2. Sa se realizeze un program care sa efectueze o aproximare cat mai corecta a intervalului de 1 secunda de comutare a ledurilor marind frecventa oscilatorului cu ajutorul buclei PLL. Se va folosi un temporizator pe 32 biti.