

11.1 Design Fabrication flow

11.1.1 Periferice și Întreruperi in Microcontrolerul PIC16F68, Introducere

Pe lângă unitatea logica de calcul al microcontrolerului mai sunt o serie de dispozitive digitale sau analogice cu diferite funcții bine determinate. Acestea sunt perifericele. Ele sunt capabile să execute anumite sarcini specifice fiecăruia fără a avea nevoie sa folosească unitatea logica de calcul, astfel economisind putere de calcul.

Printre cele mai importante periferice amintim: contoarele, porturile digitate, unitatea de comunicație seriala, sincronă sau asincrona, convertorul analog digital sau generatorul de PWM.

Când un periferic își termina sarcina, sau e dispus sa primească o noua sarcina acesta va anunța unitatea centrală a microcontrolerului prin intermediul unor biți numiți „Flag” (steguleț). In program se pot pune condiții de așteptare până când un steguleț își schimbă starea. Acest mod de așteptare se numește “polling” este un mod ce consuma resurse ale unității centrale a microcontrolerului inutil. Totuși acest mod poate fi acceptat daca perioadele de așteptare sunt foarte scurte si rar se apelează la ele.

Un mod de lucru diferit de “polling” este modul de lucru cu întreruperi. Microcontrolerul are un mecanism hardware prin care când un steguleț își schimbă starea contorul de program (PC) sa sară într-o zonă a memoriei program bine cunoscuta. Aici trebuie tratate “întreruperile”, mai bine zis trebuie inspectat care periferic a generat întreruperea și pentru fiecare periferic se executa o funcție ce se numește “rutina de deservire a întreruperii”.

Dacă în prima lecție am realizat aprinderea si stingerea LED-urilor prin „polling” acum vom încerca un exemplu prin metoda întreruperilor folosindu-ne de un periferic numărător.

11.1.2 Perifericul numărător (Timer1)

Contorul 1 din structura microcontrolerului PIC16F684 este un numărător pe 16biti. Este format din doua numărătoare pe 8 biți TMR1H si TMR1L. Ceasul contorului poate fi extern sau intern si poate fi divizat printr-un divizor programabil prin cu 3 biți. Acest contor poate funcționa sincron cu ceasul intern al microcontrolerului sau asincron. Poate genera o întrerupere după un ciclu complet de numărare (overflow). De asemeni acest numărător poate funcționa împreuna cu modulul de PWM.

Structura contorului 1 este prezenta în Figura 11. 1.



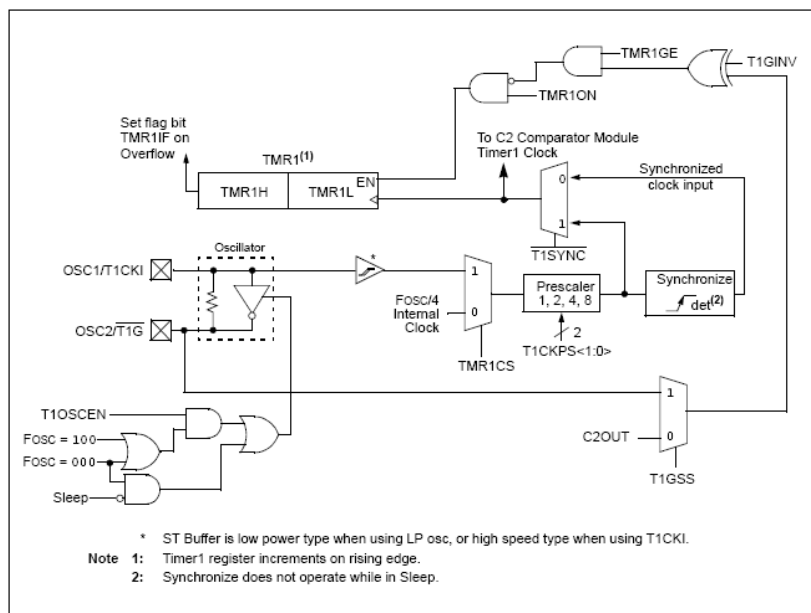


Figura 11.1. Structura Numărătorului 1

11.1.3 Registrul de control al numărătorului 1

Registrul ce controlează funcționarea numărătorului 1 se numește T1CON si este un registru pe 8 biți.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T1GINV ⁽¹⁾	TMR1GE ⁽²⁾	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **T1GINV**: Timer1 Gate Invert bit(1)

1 = Poarta contorului 1 este activa pentru valori "1" (numărătorul 1 funcționează când poarta are potențial "1")

0 = Poarta contorului 1 este active pentru valori "0" (numărătorul 1 funcționează când poarta are potențial "0")

bit 6 **TMR1GE**: Timer1 Gate Enable bit(2)

daca TMR1ON = 0:

acest bit este ignorat

daca TMR1ON = 1:

1 = Numărătorul 1 este pornit daca poarta nu este activa

0 = Numărătorul 1 este pornit

bit 5-4 **T1CKPS<1:0>**: Timer1 Input Clock Prescale Select bits

11 = 1:8 Rata de divizare

10 = 1:4 Rata de divizare

01 = 1:2 Rata de divizare

00 = 1:1 Rata de divizare

Structuri hardware si algoritmi specifici microsystemelor EMBEDDED

bit 3 **T1OSCEN**: LP (Low Power) Oscillator Enable Control bit

daca oscilatorul intern al microcontrolerului funcționează fără să furnizeze semnal extern:

1 = Oscilatorul pentru funcționarea în mod curent redus este pornit

0 = Oscilatorul pentru funcționarea în mod curent redus este oprit

altfel:

Acest bit este ignorat

bit 2 **T1SYNC**: Timer1 External Clock Input Synchronization Control bit

pentru TMR1CS = 1:

1 = Nu se realizează sincronizarea ceasului extern cu cel intern

0 = Are loc sincronizarea ceasului extern

pentru TMR1CS = 0:

Acest bit este ignorat deoarece se folosește ceas intern.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit

1 = Numărătorul va folosi ceas extern de pe pin-ul T1CKI (active pe frontal crescator)

0 = Numărătorul va folosi ceasul intern (FOSC/4)

bit 0 **TMR1ON**: Timer1 On bit

1 = Pornește Numărătorul 1

0 = Oprește Numărătorul 1

Pentru a putea folosi întreruperile generate de numărătorul 1 mai sunt o serie de biți ce trebuie configurați:

Bitul GIE din registrul INTCON - General Interrupt Enable: prin scrierea valorii „1” in acest bit întreruperile devin posibile (de la orice sursa). Prin scrierea valorii „0” toate întreruperile vor fi ignorate.

Bitul PEIE din registrul INTCON – periferic interrupt enable, prin scrierea valorii „0” In acest bit se blochează întreruperile venite de la periferice, dar nu și întreruperile venite de la alte surse (întreruperi externe, etc)

Bitul TMR1IE din registrul PIE1 – prin scrierea valorii „1” in acest bit se activează posibilitatea de generare a întreruperilor de către numărătorul 1.

Bitul TMR1IF din registrul PIR1 – este stegulețul ce indica o numărare completa a numărătorului 1. El trebuie șters din program (sa se scrie valoarea „0”)

11.1.4 Program exemplu



UNIUNEA EUROPEANĂ



MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRU



FONDUL SOCIAL EUROPEAN
POSDRU
2007-2013



INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsystemelor EMBEDDED

```
#include <system.h>
//Target PIC16F684 configuration word
#pragma DATA _CONFIG1( _LVP_ON & _MCLRE_OFF & _WDT_OFF & _INTRC_OSC_NOCLKOUT
)

//Set clock frequency
//#pragma CLOCK_FREQ 8000000
unsigned int timer1_value, adci; //declararea variabilelor globale

void no0(void)    //rutina de aprindere a ledului D0
{
    trisa.4=0;
    trisa.5=0;
    trisa.2=1;
    trisa.1=1;

    porta.4=1;
    porta.5=0;
    porta.2=0;
    porta.1=0;
}
void no1(void)    //rutina de aprindere a ledului D1
{
    trisa.4=0;
    trisa.5=0;
    trisa.2=1;
    trisa.1=1;

    porta.4=0;
    porta.5=1;
    porta.2=0;
    porta.1=0;
}
void no2(void)    //rutina de aprindere a ledului D2
{
    trisa.4=0;
    trisa.5=1;
    trisa.2=0;
    trisa.1=1;

    porta.4=1;
    porta.5=0;
    porta.2=0;
    porta.1=0;
}

void no3(void)    //rutina de aprindere a ledului D3
{
    trisa.4=0;
    trisa.5=1;
    trisa.2=0;
    trisa.1=1;

    porta.4=0;
```



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsystemelor EMBEDDED

```
        porta.5=0;
        porta.2=1;
        porta.1=0;
    }
    void no4(void)    //rutina de aprindere a ledului D4
    {
        trisa.4=1;
        trisa.5=0;
        trisa.2=0;
        trisa.1=1;

        porta.4=0;
        porta.5=1;
        porta.2=0;
        porta.1=0;
    }
    void no5(void)    //rutina de aprindere a ledului D5
    {
        trisa.4=1;
        trisa.5=0;
        trisa.2=0;
        trisa.1=1;

        porta.4=0;
        porta.5=0;
        porta.2=1;
        porta.1=0;
    }
    void no6(void)    //rutina de aprindere a ledului D6
    {
        trisa.4=1;
        trisa.5=1;
        trisa.2=0;
        trisa.1=0;

        porta.4=0;
        porta.5=0;
        porta.2=1;
        porta.1=0;
    }
}
```

```
void no7(void)    //rutina de aprindere a ledului D7
{
    trisa.4=1;
    trisa.5=1;
    trisa.2=0;
    trisa.1=0;

    porta.4=0;
    porta.5=0;
    porta.2=0;
    porta.1=1;
}
```



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

```
void interrupt( void ) //rutina de deservire a intreruperilor
{
    if( intcon & (1<<T0IF) ) //daca rutina este cauzata de un
    {                          //sursa intreruperii - T0IF din
        clear_bit( intcon, T0IF ); //se sterge "flagul" care indica
        //sursa intreruperii - T0IF din
        //registrul intcon
        tmr0=0; //timerul 0 se face 0;
        tlcon.0=0; //timerul 1 se opreste
        timer1_value=256*tmr1h+tmr1l; //in variabila globala
        //timer1value care este un numar
        //intreg pozitiv pe 16 biti se
        //stocheaza valoarea indicata de
        //timerul 1 - care este un timer pe
        //16 biti
        if(timer1_value<5000) //in functie de valoarea timerului
        //1 se aprind diferite led-uri.
        //Astfel ca daca timer-ul 1 va avea
        //ca sursa de ceas un
        { //semnal cu frecventa mica
            //se va aprinde ledul 0 sau ledul
            //1, pe masura ce frecventa
            //semanlului de ceas creste se vor
            //aprinde Led-urile cu numar mai
            //mare
            no0(); //astfel rezulta un frecventmetru
            //"bar graph"
        }
        if(timer1_value>=5000 && timer1_value <15000)
        {
            no1();
        }
        if(timer1_value>=15000 && timer1_value<24567)
        {
            no2();
        }
        if(timer1_value>=24567 && timer1_value<32768)
        {
            no3();
        }

        if(timer1_value>=32768 && timer1_value<40960)
        {
            no4();
        }
        if(timer1_value>=40960 && timer1_value<49152)
        {
            no5();
        }
        if(timer1_value>=49152 && timer1_value<57344)
        {
            no6();
        }
    }
}
```



Structuri hardware si algoritmi specifici microsystemelor EMBEDDED

```

        if(timer1_value>57344)
        {
            no7();
        }
tmr1h=0;                                //dupa tot procesul asta se scrie
                                        //in timerul 1 valoarea 0
tmr1l=0;
tlcon.0=1;                             //se da enable la timerul 1 iar si
                                        //se reincede o noua masurare a
                                        //frecventei semnalului folosit ca
                                        //ceas pentru timer 1
    }

//Handle timer1 interrupt
if( pirl & (1<<TMR1IF) )
{
    clear_bit( pirl, TMR1IF ); //clear timer 1 interrupt bit
}

//Handle timer2 interrupt
if( pirl & (1<<TMR2IF) )
{
    clear_bit( pirl, TMR2IF ); //clear timer 2 interrupt bit
}
if(pirl.6 == 1)
{
    clear_bit(pirl, ADIF);
}

}

void main( void )
{
    unsigned long i;

    osccon=0b01110000;                //ceasul intern merge la 8MHz - viteza de
                                        //calcul este de 4 ori mai mica-> 2MIPS -
                                        //-> o instructiune in 0.5us

    trisc = 0xFF;                      //pinii portului c sunt toti de intrare
                                        //si digitali
    ansel=0b00000001;                 //toti pinii portului cu exceptia lui RA0
                                        //sunt pini digitali, RA0 este pin de
                                        //intrare analogic
    //Initialize port A
    porta = 0b00000000;
    //Initialize port C
    portc = 0x00;                      //pinii portului C sunt hiZ

    option_reg=0b00000111; //din bitii 0, 1, si 2 se seteaza
                            //frecventa ceasului lui Timer0 - in
                            //cazul acesta avem o divizare a ceasului
                            //intern de 256 ori

```



Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

```

intcon=0b11100000;           //se fac intreruperile generale enable,
                             //intreruperile perifericelor enable si
                             //intreruperea Timerului 0 este enable

t1con=0b00000000;           //din bitii 5 si 4 se seteaza rata de
                             //divizare a ceasului pentru timer-ul 1
                             //acum este nedivizat.
                             //timer-ul este oprit si se foloseste ca
                             //semanl de ceas ceasul intern de 8MHz.
                             //Practic ar fi trebuit setat sa se
                             //foloseasca ceasul extern prin scrierea
//valorii 1 in bitul 1 (al doilea de la
//dreapta la stanga)
//Endless loop
while( 1 )
{

}
}

```

La începutul programului sunt definite funcțiile nox(), unde x este numărul LED-ului de pe placa PICkit1. La apelarea unei astfel de funcții LED-ul corespunzător se aprinde stingând orice alt LED.

Interrupt() este funcția de tratează întreruperile. Ea se executa ori de cate ori un steguleț trece în starea 1. În interiorul acestei funcții regiștrii ce conțin stegulețele diferitelor periferice sunt inspectate pentru a găsi perifericul ce a provocat întreruperea. Odată stegulețul identificat acesta este trecut în starea 0 printr-o operație de scriere. Dacă nu se face acest lucru întreruperea se va repeta la nesfârșit blocând funcționarea microcontrolerului.

Dacă bitul T0IF din registrul INTCON este cel ce a provocat întreruperea înseamnă ca a fost provocat de Numărătorul 0. Odată ajuns aici numărătorul este șters (se scrie 0), se oprește numărătorul 1.

Pentru citirea valorii numărătorului 1 se vor citii regiștrii TMR1H și TMR1L. Valoarea pe 16biți obținută va fi scrisă într-o variabilă globală de tip unsigned int numita timer1_value. În funcție de valoarea scrisă în acest registru se vor aprinde diferite LED-uri. După aceasta operație se resetează regiștrii numărătorului 1 și timerul1 se pornește și se așteaptă o nouă întrerupere.

Bibliografie:

1. Istvan Sztojanov, Sever Pașca, Elisabeta Buzoianu, Aplicații hardware și software cu microcontrolerul PIC12F675, Editura Cavallioti, ISBN 978-973-7622-54-9, Bucuresti 2008
2. Istvan Sztojanov, Alexandru Vasile, Elisabeta Buzoianu, Sever Pașca, *Programarea microcontrolerelor din familia Intel, Aplicații practice hardware cu 80C552*, Editura Man-Dely, ISBN 973-85681-5-3, București 2004.
3. <http://vega.unitbv.ro/~romanca/EmbSys/>
4. <http://facultate.regielive.ro/cursuri/electronica/>
5. www.microcip.com



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013