Laborator 4

Tratarea întreruperilor la dsPIC33

Obiectiv

Familiarizarea cu modul în care sunt tratate întreruperile de către controlerul de întreruperi al familiei dsPIC33. Drept exemplu, se va lucra cu întreruperea externă INT0.

Intreruperile sunt evenimente care pot aparea in cadrul executiei unui program prin care se doreste semnalarea faptului ca un anumit eveniment a avut loc. Astfel de evenimente care pot exista in cadrul unui sistem cu microcontroller pot fi: terminarea unei conversii A/D sau D/A, resetarea unui timer, intrerupere externe primita pe un pin INTx.

Controlerul de intreruperi al familie dsPIC33FJ are rolul de a reduce numeroasele cereri de intreruperi ale perifericelor intr-o singura cerere care va fi trimisa catre CPU. Acest modul de tratare a intreruperilor are urmatoarele caracteristici:

- Pana la 8 exceptii ale CPU si trap-uri software
- 7 nivele de prioritate selectabile de catre utilizator
- O tabela cu vectorii de intreruperi (126 de vectori) – IVT (Interrupt Vector Table)
- Alternate Interrupt Vector Table(AIVT) – folosit pentru debugging

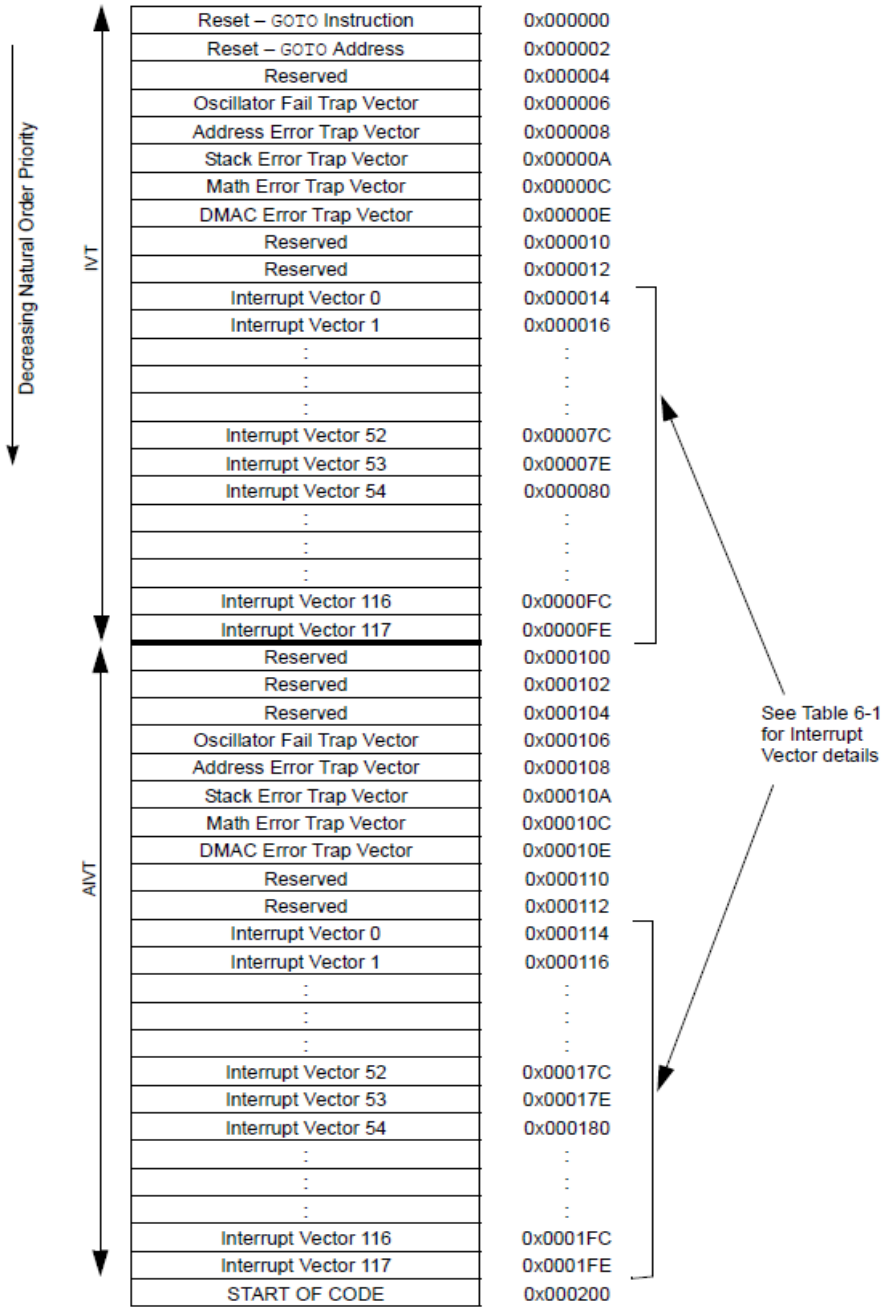
1. IVT – Interrupt Vector Table

Tabela vectorilor de intreruperi se afla in memoria program la adresa 0x000004 si contine 126 de vectori dintre care: 8 vectori pentru trapurile nemascabile si 118 pentru alte surse de intrerupere. In general, fiecarei surse de intrerupere i se va asocia un vector in aceasta tabela, acesta continand adresa de 24 de biti a rutinei corespunzatoare acelei surse de intrerupere (ISR-Interrupt Service Routine).

Secventa de RESET care se observa in aceasta tabela nu poate fi considerata ca fiind o exceptie, intrerupere deoarece modulul de tratare a intreruperilor nu este implicat direct in aceasta faza. Microcontrolerul isi reseteaza toti registrii (deci si registrul PC) si executia programului va incepe de la adresa 0x000000 unde se va afla o instructiune GOTO care va redirecta executia programului catre rutina principala a programului.

2. AIVT – Alternate Interrupt Vector Table

AIVT se afla in memoria program imediat dupa IVT si utilizarea acestuia este posibila prin setarea bitului Enable Alternate Interrupt Vector Table din registrul 2 de control al intreruperilor (INTCON2<15>). Daca acest bit este setat toate intreruperile si exceptiile folosesc aceasta tabela alternativa pentru obtinerea adreselor rutinelor de tratare a intreruperilor. Aceasta tabela este organizata in aceeasi maniera cu cea principala si este folosita in principal pentru debugging.



Exercitiu: Sa se identifice utilizand foaia de catalog vectorul de intrerupere pentru intreruperea INT0.

3. Prioritatea CPU

CPU-ul din cadrul unui microcontroler poate functiuna cu un grad de prioritate de la 0 la 15. Astfel, o intrerupere va trebui sa aiba un nivel de prioritate mai mare decat cel al CPU-ului pentru ca aceasta sa fie validata. Trebuie mentionat ca fiecarei surse de intrerupere i se poate asocia un nivel de prioritate de la 0 la 7 si ca in cazul CPU-ului nivelele 8 – 15 sunt rezervate trap-urilor.

Nivelul de prioritate al CPU-ului este indicat de catre urmatorii biti de stare:

- CPU Interrupt Priority Level (IPL<2:0>) din registrul SR<7:5>: acestia pot fi cititi si scrisi, astfel ca aplicatia poate dezactiva intreruperile care pot aparea de la o sursa cu nivelul de prioritate mai mic decat o anumita valoare. De exemplu, daca IPL = 3, CPU nu va fi intrerupt de nici o sursa cu un nivel de prioritate de 0, 1, 2 sau 3.
- CPU Interrupt Priority Level 3 (IPL3) – bit din cadrul registrului CORCON<3> care va fi setat atat timp cat are loc o intrerupere de tip TRAP.

4. Prioritate surselor de intrerupere

Fiecarei surse de intrerupere i se poate asocia un nivel de prioritate de la 0 la 7. Acest nivel poate fi modificat cu ajutorul celor mai putin semnificativi 3 biti din fiecare grupare de 4 biti (nibble) rezervata fiecarei surse din registrul IPCx. O sursa de intrerupere ce va avea setati cei 3 mai putin semnificativi biti va avea nivelul 7, deci un grad de prioritate maxim. Daca acesti 3 biti sunt 0 putem considera ca acea sursa de intrerupere este dezactivata.

Exista cazuri cand vom avea 2 sau mai multe surse de intrerupere care vor avea acelasi nivel de prioritate setat de catre utilizator. Acest exemplu de configurare a prioritatilor poate duce la aparitia unor conflicte ce vor fi rezolvat tinandu-se seama si de prioritate naturala a surselor de intrerupere si anume de pozita in IVT. Acele surse de intrerupere care vor avea indice mai mic in tabela vectorilor de intrerupere vor fi mai prioritare decat cele cu indice mai mare. Un nivel de prioritate global este dat in primul rand de catre nivelul de prioritate asignat de catre utilizator si in caz de conflict se va tine cont de catre prioritatea naturala a surselor de intrerupere rezultata din tabela vectorilor de intrerupere.

5. Intreruperile de tip TRAP

Trap-urile sunt intreruperi nemascabile provocate de catre probleme la nivel hardware sau software ale microcontroller-ului avand ca rol executarea unor rutine prin care se doreste remedierea acestor probleme. Daca utilizatorul nu doreste rezolvarea unor astfel de probleme ce pot aparea pe parcursul executiei unui program este de dorit ca vectorul de intrerupere asociat acelei surse sa pointeze catre o rutina generica de resetare a microcontrollerului.

Dupa cum se observa si din tabela IVT exista 5 surse care pot genera trap-uri:

- Oscillator Failure Trap

- Stack Error Trap
- Address Error Trap
- Math Error Trap
- DMAC Error Trap

Fiecare trap din cele enumerate mai sus are un nivel de prioritate fix dat de pozitia din tabela IVT.

6. Registri utilizati la lucrul cu intreruperi

Familia de microcontrolere dsPIC33FJ utilizeaza un total de 33 de registri pentru controlul, gestionarea si lucrul cu intreruperi. In continuare vom trata doar tipurile de registri si acei registri particulari implicati in lucrul cu intreruperea INT0.

Registrul **SR** (CPU Status Registrer) contine bitii IPL <2:0> care preprezinta cei mai putin semnificativi 3 biti cu ajutorul carora se seteaza nivelul de prioritate al CPU

Registrul **CORCON** (Control Register) contine bitul IPL3 care reprezinta cel mai semnificativ bit pentru setarea nivelului de prioritate al CPU

Registrul **INTCON1** (Interrupt Control Register 1) contine bitul NSTDIS (Interrupt Nesting Disable) cat si flagurile pentru sursele de intrerupere ale procesorului

Registrul **INTCON2** (Interrupt Control Register 2) permite selectarea lucrului cu tabela IVT sau AIVT prin intermediul bitului ALTIVT cat si a polaritatii de validare a intreruperilor externe prin intermediul bitilor INTxEP.

Registrii **IFS0-4** (Interrupt Flag Status) contin flagurile tuturor intreruperilor. Un astfel de flag reprezinta un bit ce este setat odata ce intreruperea respective si-a facut aparitia si este resetat prin software dupa achitarea ei. Flagul intreruperii INT0, respectiv INT0IF se gaseste in registrul IFS0 sub forma bitului 0

REGISTER 7-5: IFS0: INTERRUPT FLAG STATUS REGISTER 0

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	DMA1IF	AD1IF	U1TXIF	U1RXIF	SPI1IF	SPI1EIF	T3IF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T2IF	OC2IF	IC2IF	DMA01IF	T1IF	OC1IF	IC1IF	INT0IF
bit 7							bit 0

Registrii **IEC0-4** (Interrupt Enable Control) contin bitii de validare ai tuturor intreruperilor. In cazul in care bitul de validare al unei intreruperi este setat acea intrerupere este functionala iar la aparitia ei este setat flagul corespunzator, in cazul in care bitul sau de validare este resetat acea intrerupere este nefunctionala, la aparitia ei nu se intampla nimic. Bitul de validare al intreruperii INT0 se gaseste in registrul IEC0 (INT0IE)

REGISTER 7-10: IEC0: INTERRUPT ENABLE CONTROL REGISTER 0

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	DMA1IE	AD1IE	U1TXIE	U1RXIE	SPI1IE	SPI1EIE	T3IE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T2IE	OC2IE	IC2IE	DMA0IE	T1IE	OC1IE	IC1IE	INT0IE
bit 7							bit 0

Registrii **IPC0-17** (Interrupt Priority Control)sunt utilizati pentru setarea nivelului de prioritate al fiecarei intreruperi in parte, astfel fiecarei intrerupei i se poate aloca un nivel de prioritate intre 1 si 8. Prioritatea intreruperii INT0 se seteaza cu ajutorul bitilor INT0IP<2:0> din registrul IPC0

REGISTER 7-15: IPC0: INTERRUPT PRIORITY CONTROL REGISTER 0

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T1IP<2:0>			—	OC1IP<2:0>		
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC1IP<2:0>			—	INT0IP<2:0>		
bit 7							bit 0

Registrul **INTTREG** (Interrupt Control and Status Register) contine numarul vectorului de intrerupere asociat retinut de bitii VECNUM<6:0>(Vector Number) si noul nivel de prioritate al intreruperii CPU retinut de bitii ILR<3:0> (Interrupt Level)

7. Procedura de configurare

Pentru configurarea unei intreruperi este de preferat sa se urmareasca urmatoarele etape:

- Se va seta bitul NSTDIS din registrul INTCON1<15> daca nu se doreste utilizarea intreruperilor de tip nested. Intreruperile nested sunt considerate acele intreruperi care apar in momentul in care microcontrollerul se afla deja intr-o rutina de tratare a intreruperilor. In cazul in care intreruperile nested nu sunt dezactivate si noua cere de intrerupere care apare are prioritate mai mare decat cea a carei rutina se executa, aceasta va fi intrerupta de catre rutina asociata intreruperii nou aparute.
- Se va seta nivelul de prioritate al sursei de intrerupere pe care dorim sa o configuram. Acest lucru se realizeaza prin modificarea bitilor din registrul IPCx corespunzator. Daca nu se doreste utilizarea nivelelor multiple de prioritate, registrul IPCx poate fi lasat

nemodificat deoarece atunci cand dsPIC-ul se va reseta acesta isi face seta implicit toate sursele de intrerupere cu nivelul de prioritate la 4.

- Se va reseta bitul flagului de intrerupere din registrul IFSx corespunzator.
  Se va activa intreruperea prin setarea bitului de enable din registrul IECx corespunzator.

Exercitiu: Sa se configureze registrii corespunzatori pentru lucrul cu intreruperea INT0. Se va considera un nivel de prioritate egal cu 5.

8. Exemplu de program folosind intreruperea externa INT0

In continuare se prezinta un program demonstrativ care comandă schimbarea stării LED-ului conectat la pinul RB15 la apasarea butonului de intrerupere.

```
#if defined(__dsPIC33F__)
#include "p33Fxxxx.h"
#elif defined(__PIC24H__)
#include "p24Hxxxx.h"
#endif

// Select Internal FRC at POR
_FOSCSEL(FNOSC_FRC);
// Enable Clock Switching and Configure
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF);         // FRC + PLL
//_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF & POSCMD_XT);      // XT + PLL
_FWDT(FWDTEN_OFF); // Watchdog Timer Enabled/disabled by user software

void __attribute__ ((interrupt, no_auto_psv)) _INT0Interrupt(void)
{
      _RB15 = ~_RB15;
      _INT0IF = 0;// Resetam flagul corespunzator intreruperii
                  // INT0 pentru a nu se reapela rutina de intrerupere
}

void initPLL(void)
{
// Configure PLL prescaler, PLL postscaler, PLL divisor
      PLLFBD = 41;          // M = 43 FRC
      //PLLFBD = 30;        // M = 32 XT
      CLKDIVbits.PLLPOST=0;       // N1 = 2
      CLKDIVbits.PLLPRE=0;  // N2 = 2

// Initiate Clock Switch to Internal FRC with PLL (NOSC = 0b001)
```

```
      __builtin_write_OSCCONH(0x01);   // FRC
      //__builtin_write_OSCCONH(0x03); // XT
      __builtin_write_OSCCONL(0x01);

// Wait for Clock switch to occur
      while (OSCCONbits.COSC != 0b001);     // FRC
      //while (OSCCONbits.COSC != 0b011);   // XT

// Wait for PLL to lock
      while(OSCCONbits.LOCK!=1) {};
}

int main(void)
{
      initPLL();

      TRISB = 0x0000;
      _TRISB7 = 1;     // RB7 este setat ca intrare
      PORTB = 0xF000;

      _INT0IF = 0;     // Resetem flagul coresp. intreruperii INT0
      _INT0IE = 1;     // Se permite lucrul cu întreruperea INT0

      _INT0EP = 1;     // Se stabileşte pe ce front se generează INT0

      while(1)
      {
      }
}
```