Laborator 3

Biţi de configurare şi pini de uz general

Obiective

- Înţelegerea rolului biţilor de configurare şi a modului în care aceştia pot fi setaţi;
- Înţelegerea modului de lucru cu pini de uz general, setaţi ca intrări sau ieşiri; familiarizarea cu regiştrii de control TRISB, PORTB şi LATB.

1. Bitii de configurare

Inainte de a începe programarea unui microcontroller, este necesar să se stabilească condiţiile de functionare ale microcontrolerului. Printre aceste conditii se pot enumera: definirea ceasului, viteza de execuţie a instrucţiunilor, sursa de ceas, protecţia împotriva executarii unor parti gresite de program etc.

Configurarea microcontroller-ului DSPIC33FJ32MC302
Configurarea bitilor se poate face in 2 moduri:

- direct din softul MPLAB IDE
- in codul sursa

Avantajul configurarii in codul sursa este acela ca informatia se pastreaza, programul putand fi rulat de pe orice calculator.
Printre registrii importanti in configurarea microcontrolarului putem enumara :

- **FBS (Boot Code Segment Configuration Register)**
- **FGS (General Code Segment Configuration Register)**
- **FOSCSEL (Oscillator Source Selection Register)**
- **FOSC (Oscillator Selection Configuration Bits Register)**
- **FWDT (Watchdog Timer Configuration Register)**

Adresa	Nume	Bit 7	Bit 6	Bit 5	Bit 4	Bit3	Bit 2	Bit 1	Bit 0
0xF80000	FBS	RBS<1:0>		—	—	BSS<2:0>			BWRP
0xF80004	FGS	—	—	—	—	—	GSS<1:0>		GWRP
0xF80006	FOSCSEL	IESO	—	—	—	—	FNOSC<2:0>		

0xF80008	**FOSC**	FCKSM <1:0>	IOL1WAY	____	___	OSCIOFNC	POSCMD<1:0>
0xF8000A	**FWDT**	FWDTEN	WINDIS	____	WDTPRE	WDTPOST<3:0>	

a. Registrul FBS

Cu ajutorul acestui registru putem configura marimea memoriei program, marimea memoriei RAM, dar si securitatea memoriei program.

BWRB - Memoria program este:

1 = Neprotejata(poate si scrisa).
0 = Protejata (asemenea unui boot loader).

BSS<2:0> - Marimea memoriei program:

X11 = Fara memorie program.

- Marimea este de 1K.
110 = Securitate standard; memoria program se termina la adresa 0x0007FE.
010 = Securitate ridicata; memoria program se termina la adresa 0x0007FE.

- Marimea este de 4K.
101 = Securitate standard; memoria program se termina la adresa 0x001FFE.
001 = Securitate ridicata; memoria program se termina la adresa 0x001FFE.

- Marimea este de 8K.
100 = Securitate standard; memoria program se termina la adresa 0x003FFE.
000 = Securtate ridicata; memoria program se termina la adresa 0x003FFE.

RBS<1:0> - Marimea memoriei RAM.
11 = Nu este definita.
10 = Memorie RAM de 128 bytes.
01 = Memorie RAM de 256 bytes.
00 = Memorie RAM de 1024 bytes.

b. Registrul FGS

Acest registru poate si configurat pentru a oferi protectie la scrierea programului.

GSS<1:0> - Protectia segmentului general de cod
11 = Fara protectiea programului.
10 = Securitate standard.
0x = Securitate ridicata.

GWRP - Protectia programului la scriere
1 = Programul este neprotejat la scriere.
0 = Programul este proteja la scriere.

c. Registrul FOSCSEL

Acest registru poate fi configurat pentru a alege tipul oscilatorului (intern, extern, secundar).

IESO - Bitul de configurare si selectie al oscilatorului sursa

1 = Pornire cu FRC, apoi trecere automata pe oscilatorul selectat.
0 = Pornire directa cu oscilatorul selectat.

FNOSC<2:0> - Selectia oscilatorului implicit

111 = Oscilator intern Fast RC (FRC) cu postscaler.
110 = Oscilator intern Fasc RC (FRC) cu frecventa cristalului de quartz /16.
101 = Oscilator LPRC (low power).
100 = Oscilator secundar LP.
011 = Oscilator primar cu PLL(XT , HS , EC).
010 = Oscilator primar (XT , HS , EC).
001 = Oscilator intern Fast RC (FRC) cu PLL.
000 = Oscilator FRC.

Oscilatorul intern FRC functioneaza normal cu frecventa cristalului de quartz la 7.37MHz. Aceasta poate fi divizata pe o scala de la 1:2 la 1:256 folosind bitul **FRCDIV.**

Oscilatorul primar poate folosi urmatoarele sursa de ceas:

- Cristal de quartz (XT) cu frecventa intre 3-7 MHz conectat la pinii OSC1 si OSC2. Oscilatorul XT ste conceput pentru a oferi un compromis intre o performanta la frecventa ridicata si un comsun modest de energie.

- Cristal de quartz (HS) cu frecventa intre 10-40MHz conectat la pinii OSC1 si OSC2.Oscilatorul HS este conceput pentru a oferi beneficiu maxim. Este aproximativ de 5 ori mai profitabil decat un oscilator XT din punct de vedere al performantelor.

- Sursa de ceas externa legata direct la pinul OSC1.

Oscilatorul secundar LP (Low Power) este conceput pentru puteri mici si foloseste un cristal de quartz de 32.768 kHz.Este conectat la pinii SOSCI si SOSCO.

d. Registrul FOSC

FCKSM<1:0> - Modul comutarii ceasului

1X = Comutarea ceasului dezactivata, Fail-Safe Clock, Monitor dezactivat.
01 = Comutarea cesului activata Fail-Safe Clock, Monitor dezactivat.
00 = Comutarea cesului activata Fail-Safe Clock, Monitor activat.

IOL1WAY - Bit configurare perifarice

1 = Permite o singura reconfigurare.
0 = Permite mai multe reconfigurari.

OSCIOFNC - Functia pinului OSC2 (exceptie in modurile XT si HS)

1 = OSC2 clock-output.
0 = cu scop genera digital I/O.

POSCMD<1:0> - Modul oscilatorului primar

11 = Oscilator primar dezactivat.
10 = Mod oscillator HS.

01 = Mod oscillator XT.
00 = Mod EC (external clock).

e. Registrul FWDT

FWDTEN - Bit de activare watchdog timer
1 = watchdog timer mereu activat.
0 = watchdog timer activat sau dezactivat prin soft.

WINDIS - Bit de activare a watchdog timer window
1 = Mod de functionare in non-window.
0 = Mod de functionare in window.

WDTPRE - Watchdog timer prescaler
1 = 1:128
0 = 1:32

WDTPOST<3:0> - Watchdog timer postscaler
1111 = 1:32,768
1110 = 1:16,384
0001 = 1:2
0000 = 1:1

2. Pinii de uz general

Capsula dsPIC-ului 33FJ32MC302 are 28 de pini insa acest dsPIC contine numeroase periferice (printre care se numara convertoare A/D, comparatoare, module de generare semnale PWM , module de comunicatie seriala etc.) acesta fiind motivul pentru care functiile pe pinii dsPIC-ului sunt multiplexate adica pe langa functia de intrare/iesire paralelă, pinii de port mai pot avea si alte functii aditionale. De la aceasta regula fac exceptie pinii Vdd, Vss, MCLR, CLKI si OSC1. Toate porturile dsPIC-ului prezinta intrari de tip "Trigger Schmitt" ce sporesc imunitatea la zgomote.

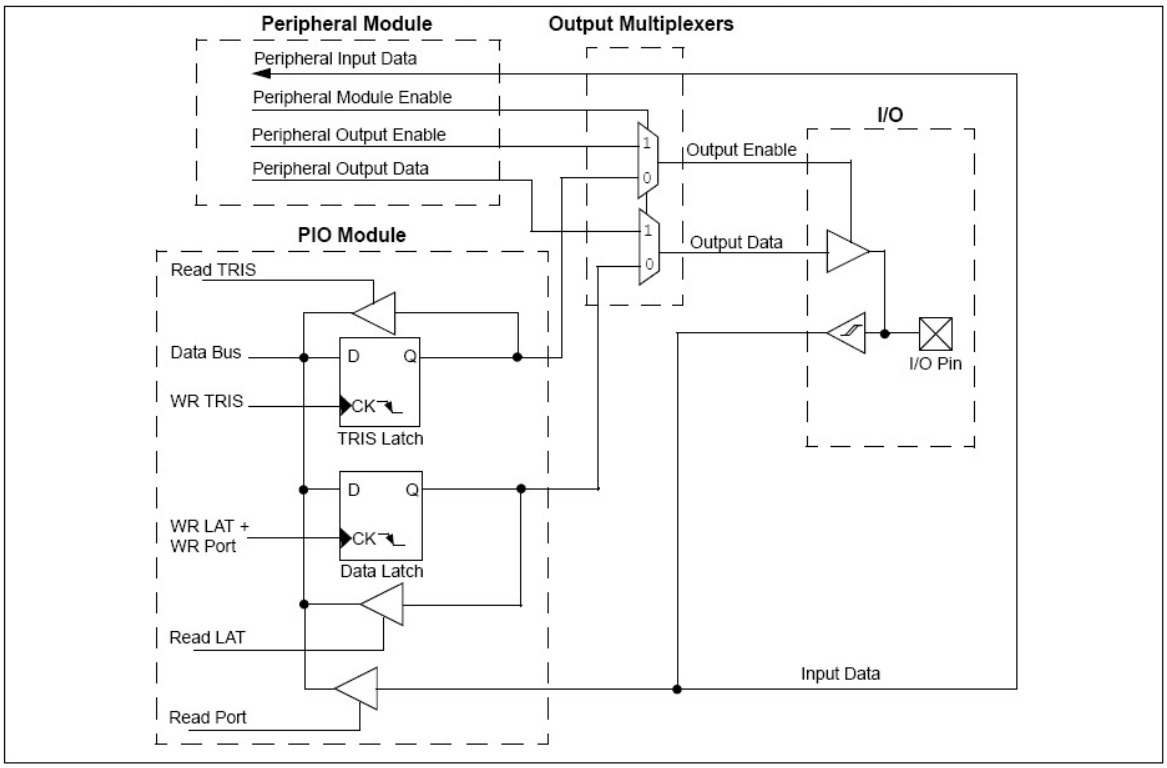
Aceasta versiune de dsPIC prezinta 2 porturi I/O , **PORTA** avand 5 pini (RA0-RA4) si **PORTB** ce are 16 pini (RB0-RB15). In general un port ai carui pini sunt impartiti si cu alte periferice este subordonat respectivului periferic adica activarea functiilor periferice este prioritară si deconectează pinul respectiv de la structura paralelă de intrare/iesire, comutând iesirea. Semnalele de date si control ale perifericelor sunt duse la niste multiplexoare prin intermediul acestora selectandu-se care dintre periferic sau portul I/O are dreptul de a folosi pinul respectiv.

Logica hardware intalnita pt fiecare pin de intrare/iesire reflecta multiplexarea functiilor pe pinii dsPIC-ului si nu este transparenta pentru programatorul microcrontrolerului si de aceea explicatiile legate de figura de mai jos au caracter doar informativ.

In figura se poate observa ca multiplexoarele „aleg" semnalul de control si semnalul de data al pinului I/O. Se poate observa ca semnalul de control al pinului este fie valoarea memorata in latchul TRIS fie semnalul numit Peripheral Output Enable in functie de valoarea semnalului Peripheral Module Enable. Acelasi lucru se intampla si in cazul semnalului de data al pinului cand este selectat unul dintre semnalele Peripheral Output Data si valoarea din Data latch.

In cazul in care se doreste folosirea perifericului valoarea semnalului Peripheral Module Enable fiind semnal de selectie pt multiplexoare face ca semnalele Output Enable si Output Data sa fie de fapt semnalele Peripheral Output Enable si Peripheral Output Data. In caz contrar, adica atunci cand perifericul nu este activate, semnalele Output Enable si Output Data devin valorile memorate in TRIS latch si DATA latch.

FIGURE 11-1: BLOCK DIAGRAM OF A TYPICAL SHARED PORT STRUCTURE



In cazul in care un periferic activ foloseste un pin pentru iesire acesta poate fi doar citit ca si pin al unui port I/O caz in care valoarea returnata este valoarea logica scoasa pe pinul respectiv de catre periferic. In cazul in care perifericul activ nu foloseste respectivul pin ca iesire acest pin poate fi citit dar si scris de catre un port.

Functionarea fiecarui pin, independenta de restul pinilor din porturile I/O este configurata folosindu-se trei registre. Aceste registre sunt **TRISA** , **PORTA** si **LATA** pentru primul port ce are 5 pini si respectiv **TRISB**, **PORTB** si **LATB** pentru cel de-al doilea port ce are 16 pini. Valoarea din registrul TRIS**X** determina sensul de transmitere a datelor pe PORTX. Pinii ce au ca valoare corespunzatoare in registrul TRIS**X** 1 logic sunt configurati ca pini de intrare iar cei care au ca valoare corespunzatoare 0 sunt configurati ca pini de iesire. Dupa un reset toti pinii sunt setati ca intrari, in registrul TRIS**X** introducandu-se automat valoarea 0XFFFF.

Exemplu: Daca in registrul TRISA este memorata valoarea 01011(valoare binara) atunci pinii RA0 si RA2 sunt configurati ca iesiri iar pinii RA1, RA3 si RA4 sunt configurati ca intrari.

Registrele LATA si LATB sunt formate efectiv din latch-urile de date (DATA latch) a fiecarui port corespunzator. Citirea acestor registre are ca rezultat ultima valoarea scoasa pe port fata de citirea lui PORTX cand valoarea rezultata este data de nivelul tensiunii scoase pe pinii portului ce poate fi si rezultatul folosirii acestora de catre periferice. Scrierea in registrele LATX are acelasi efect ca si scrierea in PORTX adica inscrierea in DATA lacth-uri a valorilor corespunzatoare.

In plus unii pini pot fi configurati individual ca iesiri cu drena deschisa (open-drain). Acest lucru este controlat folosindu-se registrul **ODCX** ce este valabil pt fiecare port. Bitii setati din acest registru permit pinilor corespunzatori sa poata fi folositi ca iesiri cu drena deschisa. Iesirile cu drena deschisa permit furnizarea de tensiuni mai mari decat Vdd folosindu-se in exterior rezistente de fixare.

TABLE 4-32: PORTA REGISTER MAP FOR dsPIC33FJ128MC202/802, dsPIC33FJ64MC202/802 AND dsPIC33FJ32MC302

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISA	02C0	—	—	—	—	—	—	—	—	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	079F
PORTA	02C2	—	—	—	—	—	—	—	—	—	—	—	RA4	RA3	RA2	RA1	RA0	xxxx
LATA	02C4	—	—	—	—	—	—	—	—	—	—	—	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx
ODCA	02C6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-34: PORTB REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISB	02C8	TRISB15	TRISB14	TRISB13	TRISB12	TRISB11	TRISB10	TRISB9	TRISB8	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	FFFF
PORTB	02CA	RB15	RB14	RB13	RB12	RB11	RB10	RB9	RB8	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx
LATB	02CC	LATB15	LATB14	LATB13	LATB12	LATB11	LATB10	LATB9	LATB8	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx
ODCB	02CE	—	—	—	—	ODCB11	ODCB10	ODCB9	ODCB8	ODCB7	ODCB6	ODCB5	—	—	—	—	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

O parte dintre pinii dsPIC-ului au si functii de intrari analogice (ANX). Configurarea pinilor ce suporta aceasta facilitate este realizata cu ajutorul registrelor **TRISX** si **AD1PCFGL.** Acesti pini coincid cu pinii RA0, RA1, RB0, RB1, RB2 si RB3.

Pentru a folosi un pin ca si intrare analogica trebuie ca acesta sa fie setat prin intermediul lui TRISX ca intrare iar bitul corespunzator din AD1PCFGL sa fie 0. Daca pinul respectiv este configurat ca iesirea va scoate nivelul de tensiune pentru 0 sau 1 logic. Valoarea predefinita din registrul **AD1PCFGL** este 0x0000, **deci pinii ce suporta aceasta functie in mod implicit sunt configurati ca intrari analogice** si deci trebuie configurati explicit prin program ca intrari/iesiri digitate daca se doreste functionarea in acest regim. Daca se citeste valoarea unui port pinii configurati ca intrari analogice sunt cititi ca 0 logic.

In cazul programelor in care se citesc si scriu valori pe porturi trebuie tinut cont de faptul ca schimbarea sensului datelor pe un port sau scrierea datelor de pe un port urmata de citirea aceluiasi port se pot face doar cu un delay de 1 ciclu intre operatii. Acest delay poate fi obtinut folosindu-se o instructiune NOP.

Exemplu:

```
MOV     0xFF00, W0 ;configurarea bitilor PORTB<15:8> ca intrari
MOV     W0, TRISBB ; si a bitilor PORTB<7:0> ca iesiri
NOP                ; Delay de un ciclu
btss    PORTB, #13 ; urmatoarea instructiune
```

3. Exemple de cod ce folosesc bitii de configurare si pini de uz general

Exemplul 1:
(aplicatie ce aprinde pe rand unul dintre cele 4 leduri de pe placa)

```
#include "p33Fxxxx.h"

_FWDT(FWDTEN_OFF);

void delay(unsigned int t)
{
        unsigned int x;
        unsigned int i;
        for(i=0;i<650;i++)
                for(x=0;x<t;x++);
}
```

```
int main(void)
{
    TRISB = 0x0000;
    PORTB = 0x0000;

    while(1)
        {
        PORTB=0x7000;
        delay(615);
        PORTB=0xB000;
        delay(615);
        PORTB=0xD000;
        delay(615);
        PORTB=0xE000;
        delay(615);
        }
    return 0;
}
```

Problema propusa:

Sa se realizeze un program in asa fel incat secventa de la Ex1 sa fie afisata pe led-uri la apasarea butonului in felul urmator: daca se apasa si se elibereaza butonul led-ul aprins sa se mute cu o pozitie iar daca este tinut apasat secventa sa ruleze ca la ex1 pana la eliberarea butonului.

Se pot incerca variatii pe acelasi program de genul schimbarii secventei afisate in urma apasarii butonului.