

## TP2

### Utilisation des interruptions

---

1) Etudier et tester le programme en langage d'assemblage "tp2.s" ci-dessous qui utilise :

- un timer pour déclencher **automatiquement** la conversion analogique numérique
- et **l'interruption de fin de conversion ADC** pour recopier l'entrée AN0 sur le PORT C

2) Modifications du programme :

2-1 Calculer la valeur à mettre dans PR3 pour obtenir 48000 conversions par seconde. Vérifier à l'aide de l'oscilloscope.

2-2 Modifier la routine d'interruption pour sauvegarder W0 dans la pile au début de la routine d'interruption et le restaurer à la fin pour ne pas perturber le programme principal qui va utiliser W0 dans la suite du TP. (voir les instructions assembleur « push » et « pull »)  
Vérifier que le programme fonctionne toujours (simulation ou test sur la cible)

2-3 Fabriquer un créneau sur le bit 2 du port B dans la boucle principale.

- Initialiser RB2 en sortie numérique
- Utiliser : mov #0xFFB,W0 puis xor LATB pour changer l'état logique de RB2.
- Vérifier le bon fonctionnement en simulation.
- Charger le programme dans le dspic et observer les signaux sur RB2 et en sortie du réseau R/2R sur le port C.
- Essayer de mettre en évidence l'effet des interruptions sur le signal RB2.

3) Compléter le programme pour faire clignoter 1 fois par seconde une LED branchée sur le bit 3 du port B (RB3) en utilisant l'interruption « Timer2 overflow » et PR2.  
Le programme principal doit continuer de fonctionner ainsi la copie de l'entrée AN0 sur le port C

4) Faire les modifications sur la version en langage C : tp2.C et vérifier le bon fonctionnement

# INSSET DE ST QUENTIN

;Programme assembleur tp2.s à étudier et à modifier

```
;-----  
;La conversion de l'entrée analogique AN0 est déclenchée automatiquement par le TIMER3  
;Lorsque la conversion est terminée le convertisseur déclenche une demande d'interruption  
;la routine d'interruption envoie les 8bits de poids forts du résultat de la conversion sur le port C  
;Le programme principal après l'appel des sous programmes d'initialisation exécute une boucle infinie  
;-----
```

;\*\*\*\* Désactiver le watchdog\*\*\*\*\*

```
,  
    .section __FWDT.sec,code  
    .global __FWDT  
__FWDT:    .pword 0xFF7F  
;*****
```

```
,  
    .global __reset          ;adresse 000000 de la mémoire programme ( voir fichier 30F6014.gld )  
    .global __ADC1Interrupt  ;adresse du vecteur d'interruption ADC ( voir fichier 30F6014.gld )  
    .section .text           ;indique au linker d'utiliser la mémoire programme pour la suite
```

;\*\*\*\*\*

; Le Programme principal

;\*\*\*\*\*

```
__reset:  
    mov    #0x800,W15        ;initialisation du pointeur de pile  
    rcall  __init_timer3     ;sous-programme d'initialisation  
    rcall  __init_portC  
    rcall  __init_adc  
    rcall  __init_int  
  
boucle:  
    nop                      ; Le programme principal fonctionne en boucle infinie.  
    bra    boucle            ; ne fait rien mais prend 1 cycle (no operation)
```

;fin du\_programme principal.....

;\*\*\*\*\*

; Routine de traitement de l'interruption ADC

;\*\*\*\*\*

```
__ADC1Interrupt:  
    mov    ADC1BUF0,W0        ;lecture AN0  
    lsr    W0,#8,W0           ;décalage à droite de huit bits  
    mov    W0,PORTC           ;envoi en sortie  
    bclr   IFS0,#13           ;réinitialise l'indicateur d'interruption ADC  
    RETFIE
```

;\*\*\*\*\*

; Les sous-programmes

;\*\*\*\*\*

\_\_init\_portC:

```
    clr    TRISC  
    return
```

\_\_init\_timer3:

```
    mov    #0x8000,W0  
    mov    W0,T3CON  
    mov    #0x100,W0  
    mov    W0,PR3  
    return
```

\_\_init\_adc:

```
    mov    #0xfffe,W0  
    and    AD1PCFGL          ; mise à 0 du bit 0 par un ET avec W0 (fffe)  
    mov    #0x010A,W0  
    mov    W0,AD1CON3  
    mov    #0x0000,W0  
    mov    W0,AD1CON2  
    mov    W0,AD1CHS0  
    mov    #0x8244,W0  
    mov    W0,AD1CON1  
    return
```

\_\_init\_int:

```
    disi   #4  
    bclr   IFS0,#13  
    bset   IEC0,#13  
    return
```

.end ; fin du fichier

```

/*-----
Le programme écrit en langage C (utile pour la question 4)
Programme tp2.c
-----*/

/*-----
La conversion de l'entrée analogique AN0 est déclenchée automatiquement par le TIMER3
Lorsque la conversion est terminée le convertisseur déclenche une demande d'interruption
la routine d'interruption envoie les 8bits de poids forts du résultat de la conversion sur le port C
Le programme principal après l'appel des sous programmes d'initialisation exécute une boucle infinie
-----*/

#include "p33FJ16MC304.h"
#pragma config FWDTEN = OFF          // Watchdog timer enabled / disabled by user software (désactivé)

/*****
/ Liste des fonctions (on peut utiliser un fichier .h et #include... )
void init_portC(void);
void init_timer3(void);
void init_adc(void);
void init_int(void);
/*****/

/*****
/ Le programme principal
int main(void)
{
    init_timer3(); // appel de sous-programmes
    init_portC();
    init_adc();
    init_int();
    while(1)
    {
        // Boucle infinie qui ne fait rien...
        // On y place le programme principal
    }
}

/ fin du programme principal

/*****/
/ Routine de traitement de l'interruption ADC
void __attribute__((__interrupt__, __shadow__, __no_psv__)) _ADC1Interrupt(void)
{
    unsigned int a;
    a = ADC1BUF0;          // lecture AN0
    a = a >> 8;            // décalage à droite de huit bits
    PORTC = a;            // envoi en sortie
    asm("bclr IFS0,#13"); // ou IFS0bits.AD1IF = 0; // efface le bit indicateur d'interruption AD
}

/ fin de la routine

/*****/
/ Définition des fonctions (on peut les placer dans un autre fichier .c
void init_portC(void)
{
    TRISC = 0x0000;
}

void init_timer3(void)
{
    T3CON = 0x8000;
    PR3 = 0x0100;
}

void init_adc(void)
{
    AD1PCFGL = 0xFFFE;
    AD1CON2 = 0x0000;
    AD1CON3 = 0x010A;
    AD1CHS0 = 0x0000;
    AD1CON1 = 0x8244;
}

void init_int(void)
{
    asm("DISI #4");
    asm("bclr IFS0,#13"); // IFS0bits.AD1IF = 0; // autorisation int ADC
    asm("bset IEC0,#13"); // IEC0bits.AD1IE = 1; // autorisation int ADC
}

/*****/

```