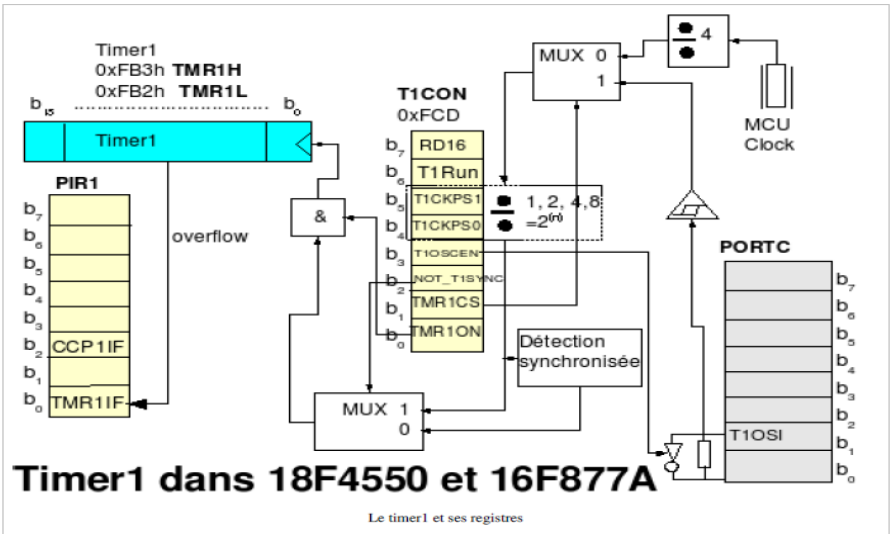
Travaux dirigés N°5
Le mode comparaison du module CCP (Capture Compare PWM)

Un petit rappel de la documentation du module comparaison avant de commencer



Exercice 1

Le PIC® possède un oscillateur de fréquence 4 MHz.
Quelle est la plus petite valeur de la période de déclenchement des interruptions que l'on peut générer avec cette horloge interne ?

Exercice 2 Réalisation d'un signal de 1Hz et son amélioration

1°) Notre horloge quartz a une fréquence de 20 MHz. Utiliser le résultat de l'exercice 1 pour répondre : par combien doit-on diviser la fréquence de CCP1IF minimale pour obtenir un temps de 500000 ms (0,5 s)?
2°) On choisit une division par 5. Le principe sera donc d'attendre CCP1IF cinq fois sans oublier de le remettre à 0 et d'inverser par logiciel la sortie RC2. Il nous faut donc choisir un mode de fonctionnement logiciel (1010 qui n'agit pas sur RC2). Écrire le programme complet
3°) Avec la technique de la question précédente, il nous est absolument impossible de régler exactement la fréquence de sortie. On va essayer de palier à cet inconvénient en utilisant un autre mode qui n'agit pas sur RC2 mais a l'avantage de déclencher une remise à 0 du timer1 quand il y a comparaison : c'est le mode 1011. Calculer la valeur à mettre dans CCPR1.
Écrire le programme.
4°) Compléter le programme précédent pour qu'il envoie sur une liaison série (avec un printf) le temps en heure minute seconde.

Solutions

Exercice 1

division par 4 puis 8 donne 125 kHz et le timer divise encore par 65536 ce qui donne 1,91 Hz

Exercice 2

1°) Quartz 5 fois plus rapide donc nos 1,81 Hz se transforment en 9,53674 Hz alors que l'on veut 2 Hz donc division par 4,77 soit environ 5.

2°) Dans la correction ci-dessous CCPR1 = 0xFFFF mais toute autre valeur ferait l'affaire.//******** Hitech C ********

```
#include <htc.h>

// Quartz a {{Unité|20|{{abréviation|MHz|méga-hertz}}}} on genere 2Hz sur b2 du PORTC

void initTimer1(void);


void main(void) {

unsigned char nb;

//Initialisation des variables

    nb=0;

// PORTC bit b2 en sortie

    TRISC2=0;

// CCP1IF seul donc 1010

    CCP1M3 = CCP1M1 = 1;

    CCP1M2 = CCP1M0 = 0;

// initialise le registre comparaison

    CCPR1H=CCPR1L=0xFF;
```

```
        initTimer1();
// clear le flag CCP1IF
        CCP1IF=0;
        while(1){
// on attend le flag
                while(CCP1IF==0);
                nb++;
// vu la frequence faible on a le temps
                if (nb==5){
                        nb=0;
                // ou exclusif pour inverser le bit
                        RC2 = RC2 ^ 1;
                }
// clear le flag CCP1IF
        CCP1IF=0;
        }
}
void initTimer1(void){
  // division par 8
  T1CKPS0 = 1;
  T1CKPS1 = 1;
 // choix du quartz
   TMR1CS = 0;
// en synchronisé car comparaison
```

```
   T1SYNC = 0;
//initialisation du timer TMR1H en premier
  TMR1H = 0x00;
  TMR1L = 0x00;
// mise en route du timer1
  TMR1ON = 1;
}
```

3°) 2Hz => 0,5s On garde la division par 5 alors pour réaliser 2Hz => à réaliser 0,1s

TQuartz x 4 x 8 x (x+1) = 0,1s Trouver x ?

Remarque

Le +1 du (x+1) est un détail de fonctionnement du PIC, la synchronisation se fait toujours à l'instruction suivante. On peut peut-être passer ce point de détail aux étudiants.

x+1=62500. => x=62499.

62500 = 0xF424//******** Hitech C ********

```
#include <htc.h>
// Quartz a {{Unité|20|{{abréviation|MHz|méga-hertz}}}} on genere 2Hz sur b2 du PORTC
...
// CCP1IF seul et RAZ timer1 donc 1011
      CCP1M3 = CCP1M1 = CCP1M0 = 1;
      CCP1M2 = 0;
```

```
// initialise le registre comparaison
        CCPR1H=0xF4;
        CCPR1L=0x23; // ou 0x24
        initTimer1();
...
```