CHAPITRE 4

LES INTERRUPTIONS

1. Principe

L'interruption est un mécanisme fondamental de tout processeur. Il permet de prendre en compte des événements extérieurs au processeur et de leur associer un traitement spécifique.. Il faut noter que l'exécution d'une instruction n'est jamais interrompue ; c'est à la fin de l'instruction en cours lors de l'arrivée de l'événement que le sous-programme d'interruption est exécuté. A la fin de cette procédure, le microcontrôleur reprend le programme principal à l'endroit ou il l'a laissé.

1. Le Mécanisme général d'une interruption est :
2. Le programme se déroule normalement
3. L'évènement survient
4. Le programme achève l'instruction en cours de traitement
5. Le programme saute à l'adresse de traitement de l'interruption
6. Le programme traite l'interruption
7. Le programme saute à l'instruction qui suit la dernière exécutée dans le programme principal

Les interruptions peuvent être causées par des sources externes ou par des sources internes

- Sources externes:
  - broches parallèles (Exemples: clavier, alarme)
  - ports séries
- Sources internes
  - Timer
  - Convertisseur A-N
  - Reset

2. Source d'interruption dans le PIC 16F877

Le microcontrôleur PIC16F877 dispose 14 sources d'interruptions. Chaque interruption a un bit d'autorisation (Enable) et un bit indicateur (Flag). Les différentes sources d'interruptions sont :

- Timer 0
- Pin RB0
- Ch. RB4/RB7
- Convert. A/D
- Rx USART
- Tx USART
- Port série SSP
- Module CCP1
- Module CCP2
- Timer 1
- Timer 2
- EEPROM
- SSP mode I2C
- Port parallèle

2. 1 Programmer une interruption avec CCS

- **Source de l'interruption**

La source de l'interruption est indiquée au début du sous-programme d'interruption par la directive **#int_XXXX** (XXXX nom de l'interruption) .

```
#INIT_XXXX                //Nom de l'interruption
Void   nom de fonction (Void)
{
    Instruction 1 ;
    Instruction n ;
}
```

Exemple:

#int_ext : Cette directive identifie RB0 comme source de l'interruption : interruption externe

#int_rb : Changement d'état de RB4 à RB7.

#int_TIMER0 : débordement du timer0.

#int_EEPROM : fin d'écriture dans l'EEPROM.

Seules les broches définies en entrée peuvent déclencher une interruption. Ces directives ne discriminent pas quelle entrée est la source, mais on peut utiliser bit_test()

- **Validation de l'interruption**

L'interruption ne peut être active que si elle est validée :

enable_interrupts(INT_EXT); // Valide l'interruption sur RB0

enable_interrupts(GLOBAL); // validation globale

L'interruption doit être validée individuellement (INT_EXT) et globalement.

- **Interdire une interruption**

On peut interdire une interruption par l'instruction **disable_interrupt()** Les paramètres sont les mêmes :

disable_interrupts(INT_EXT);

disable_interrupts(GLOBAL);

2. 1 Interruption avec une source externe: RB0

Cette interruption est provoquée par un changement d'état sur l'entrée RB0 du port B quand elle est programmée en entrée.

Exemple 1:

Écrire un programme qui fait clignoter une Led branchée sur RC0 chaque fois que la broche RB0 passe de 1 à 0.

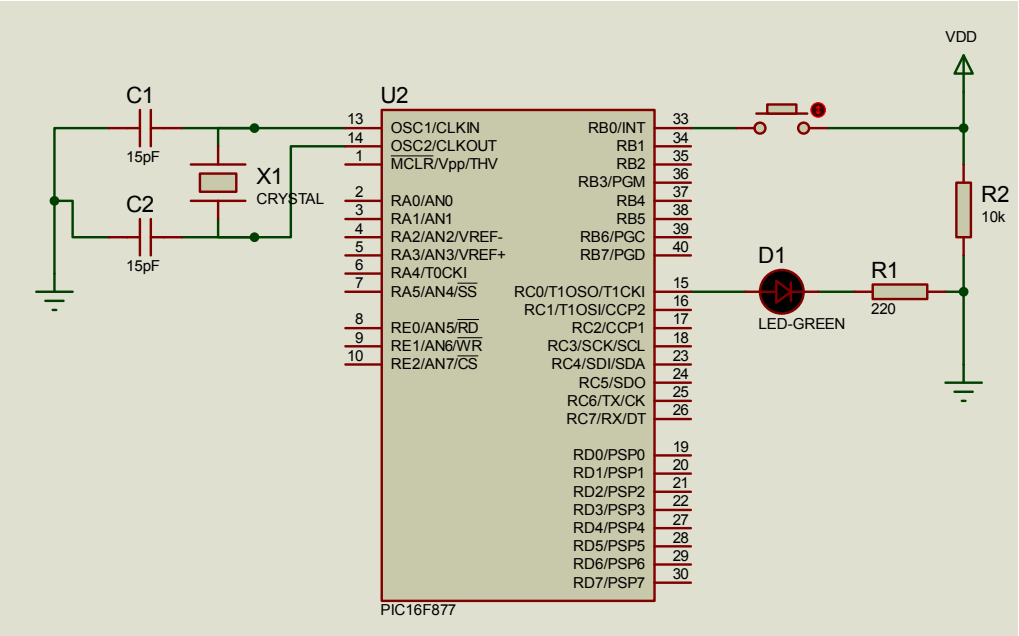

Fig. 4.1 : Montage illustrant l'interruption RB0

```
#include  "16F877.H"
#use delay(clock=20000000)
#define LED  PIN_C0          // Led temoin
```

```
void cligne(int x);              // prototype de la fonction cligne()
//-----------------------------------------------------//
// Sous programme de traitement de l'interruption externe
//-----------------------------------------------------//
#int_ext        // Cette directive indique que la fonction suivante est la tache de l'interruption
rb_ext( )
{
  cligne(3);
}
//-------------------- Programme principal ----------------------
void main(void)
{
 ext_int_edge(H_TO_L);              // Front descendant
 enable_interrupts(INT_EXT);        // Valide l'interruption sur RB0
 enable_interrupts(GLOBAL);         // Valide les interruptions
 set-tris_c(0x00);                  // port c en entrée

 while(1);                          // ce programme ne fait rien
}
//-----------------------------------------------------//
//  void cligne(int x)          //  Clignotement de la led verte x fois à intervalle de 1 s.
//-------------------------------------------------- //
void cligne(int x)
{
int i;
   for (i=0; i<x; ++i)
     {
      output_low(led);
      delay_ms(1000);
      output_high(led);
      delay_ms(1000);
     }
}
```

2. 2 Interruption BI (RB4 A RB7 du port B)

Cette interruption est provoquée par un changement d'état sur l'une des entrées RB4 à RB7 du port B, Le front n'a pas d'importance.

Exemple 2:

On reprend l'exemple 2 sauf que la Led ne s'allume que s'il y'a un changement sur B4_B7

```
#include  "16F877.H"
#use delay(clock=20000000)
#byte port_b = 6                  // adresse du port B
#define LED  PIN_C0          // Led temoin
void cligne(int x);               // prototype de la fonction cligne()
//-----------------------------------------------------//
// Sous programme de traitement de l'interruption externe
//-----------------------------------------------------//
#int_rb
rb_ext( )
{
  int lecture;
  disable_interrupts(GLOBAL);  // Évite de s'interrompre soi-même
  lecture = port_b & 0xF0;     // isole les 4 bits d'en haut
  if(bit_test(lecture, 4))         // on peut tester les bits pour décider des actions
   cligne(4);
   if(bit_test(lecture, 5))
   cligne(5);
   if(bit_test(lecture, 6))
   cligne(6);
  if(bit_test(lecture, 7))
     cligne(7);
  enable_interrupts(GLOBAL);
}
//-------------------- Programme principal ----------------------
void main(void)
{
 set_tris_b(0xFF);               // port_b en entrée
```

```
 set_tris_c(0x00);              // port_b en entrée

 enable_interrupts(INT_RB);     // Valide l'interruption sur B4-B7
 enable_interrupts(GLOBAL);     // Valide les interruptions
  while(1);                     // ce programme ne fait rien
}
//-----------------------------------------------------//
//  void cligne(int x)           //  Clignotement de la led verte x fois à intervalle de 1s.
//----------------------------------------------------- //
void cligne(int x)
{
int i;
   for (i=0; i<x; ++i)
     {
      output_low(led);

      delay_ms(200);
      output_high(led);
      delay_ms(200);
    }
}
```

2.3 Les autres interruptions

Les autres interruptions seront abordées au moment de l'étude des modules qui les déclenchent.