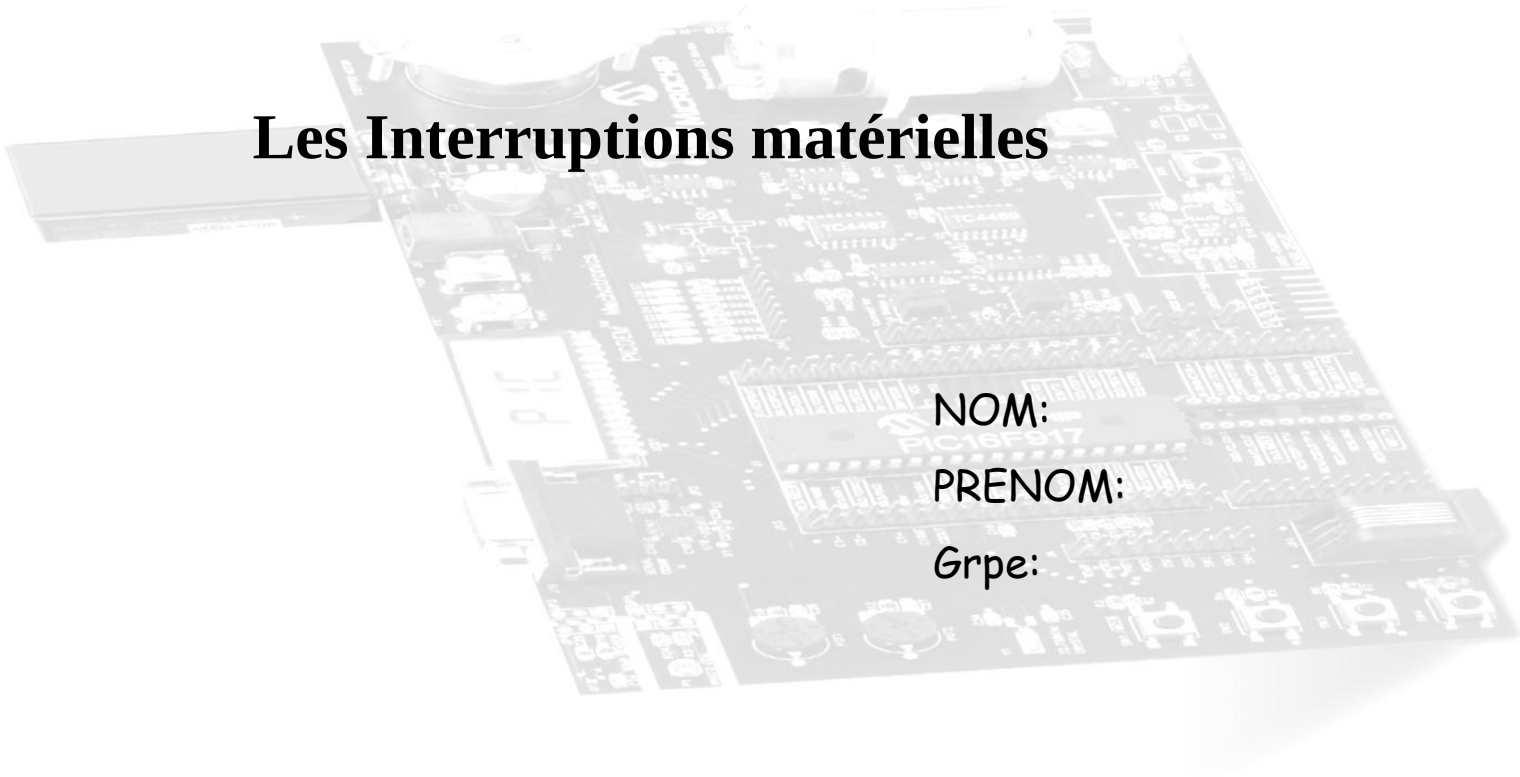
Cours de PIC

Les Interruptions matérielles



NOM:

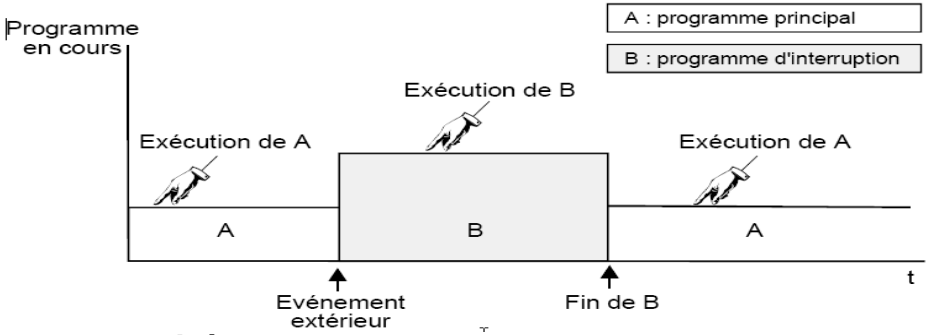
PRENOM:

Grpe:

Principes

Définition

- Une interruption est un événement qui interrompt un programme en cours pour faire exécuter au microprocesseur un autre programme appelé **programme d'interruption** (Interrupt Service Routine, **ISR**). Cet autre programme gère un phénomène plus important ou plus urgent que celui géré par le programme interrompu
- Ce programme d'IT se termine par une instruction spéciale (RETFIE en assembleur) qui permet au programme interrompu de reprendre le cours normal de son exécution (en C automatiquement rajouté avec la dernière accolade' }')



✓ L'éxécution de B est déclenchée automatiquement par le processeur lors de l'événement

✓ Il n'y a d'appel explicite de l'ISR

✓ A peut être interrompu à n'importe quel instant par B (asynchronisme)

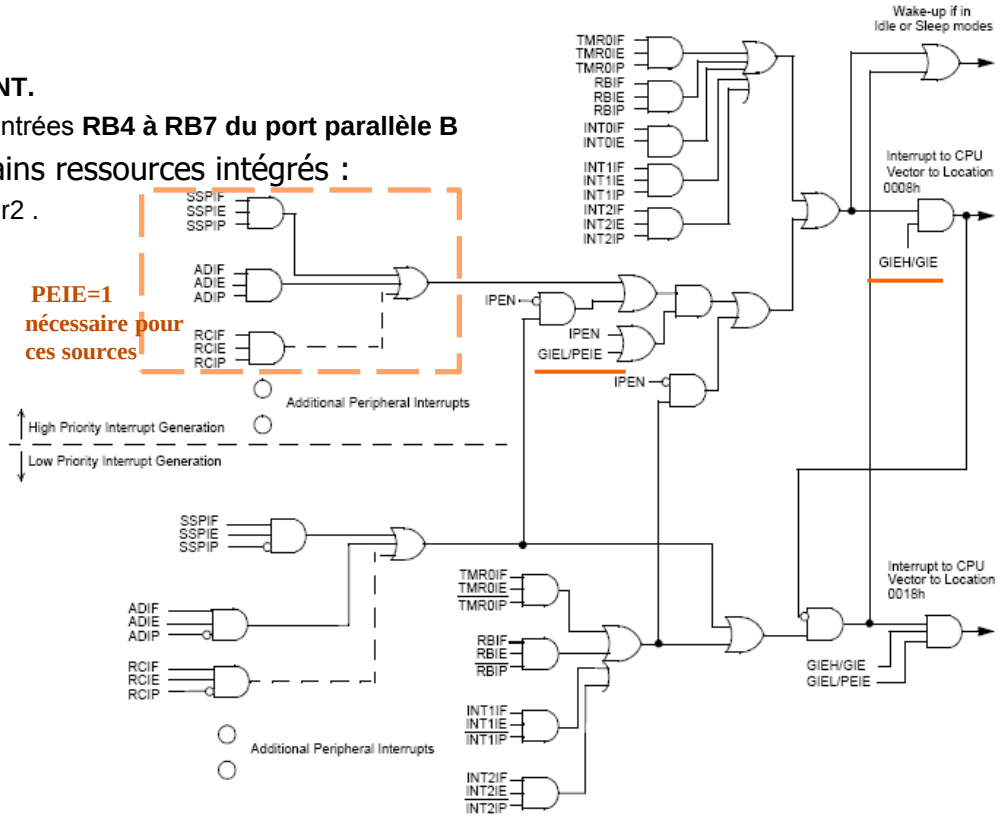
Intérêt des interruptions

- Gain de ressources CPU par rapport de la technique du polling
  - ✓ Les tests (instructions IF) d'événements mobilisent le processeur pour des évènements qui restent parfois rares
    - Ex: Traitement d'un arrêt d'urgence
- Traitement immédiat possible de l'événement
  - ✓ Le fréquence du polling est liée aux instructions dans la boucle for(;;)
    - Le temps de réaction (pire des cas) est le temps de la boucle donc!
  - ✓ L'interruption garanti un temps minimum de traitement:, notion NECESSAIRE des systèmes temps-réels
    - Ex: un avion, une centrale nucléaire
- Associées à un timer ,permet des actions périodiques
  - ✓ Période constante *-à la différence du for(;;) -*
  - ✓ Notion de système échantillonnée
- Permet de hiérarchiser les différents événements
  - ✓ Notion de priorité
  - ✓ Pour le PIC 2 niveaux de priorité: hautes et/ou basses
    - RCONbits.IPEN=1 => 2 niveaux activés: bas et haut

Principes

Sources d'IT du PIC

- **Matérielles externes :**
  - ✓ Déclenchée par un front actif sur l'entrée **RB0/INT.**
  - ✓ Déclenchée par le changement d'état d'une des entrées **RB4 à RB7 du port parallèle B**
- **Matérielles internes :** Déclenchées par certains ressources intégrés :
  - ✓ Débordements du Timer 0, du Timer1 ou du Timer2 .
  - ✓ Le Comparateur Analogique.
  - ✓ L'USART en Emission ou Réception de données.
  - ✓ Le Convertisseur Analogique Digital.
  - ✓ Le circuit de Capture et de Comparaison CCP.
  - ✓ Le port série synchrone ou SSP.
  - ✓ Ecriture dans la mémoire EEPROM



Modèle de programmation

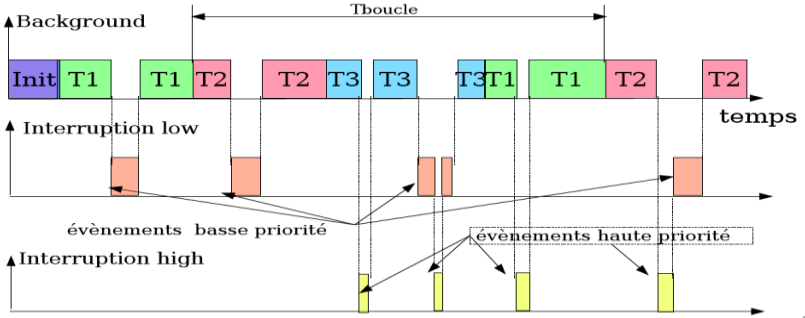
```
void main(void)
{
    séquence d'initialisation ;
    ....;
    autorisation éventuelle des interruptions ;
    for( ;;)
    {
        Tâche n°1 ;
        Tâche n°2 ;
        ....;
        Tâche n° N ;
    }
}
```

Tâche de fond

```
#pragma interrupt InterruptHandlerHigh
void InterruptHandlerHigh ()
{
....;
}

#pragma interrupt InterruptHandlerLow
void InterruptHandlerLow ()
{
....;
}
```

Gestionaires d'interruptions

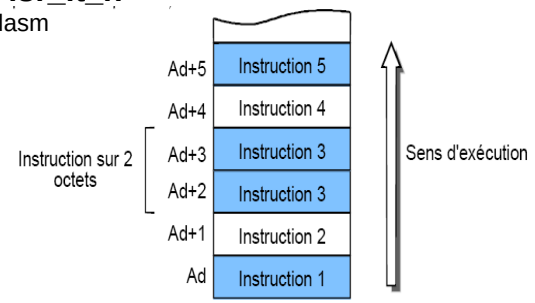


Programmation

Mise en place du code de l'ISR

- 2 adresses importantes
  - ✓ Ces adresses contiennent la 1ère instruction de la routine d'interruption
    - 0x08 pour les IT hautes
    - 0x18 pour les IT basses
- Etape 1: placer un saut vers votre ISR

```
#pragma code vect_it_h=0x08
void_vect_it_h (void)
{
_asm
//permettre de se brancher sur la fonction d'interruption
goto isr_it_h
_endasm
}
```



- Etape 2: code de votre ISR

```
#pragma code
#pragma interrupt isr_it_h
void isr_it_h (void)
{
// votre routine d'IT ICI
}
```

Miminum à faire dans l'ISR

- Etape 1: tester qui a déclenché l'IT
  - ✓ Un bit événement Interrupt Flag : xxxIF
    - **CAN ->PIR1bits.ADIF**
    - **Bouton poussoir RB0 ->INTCONbits.INT0IF**
- Etape 2: acquitter manuellement l'interruption -cas le + général -
  - ✓ Remettre le flag précédent à 0

```
#pragma code
#pragma interrupt isr_it_h
void isr_it_h (void)
{
if (INTCONbits.INT0IF=1){ // Si BP sur RB0 a declenché IT
  cpt=cpt+1; // var globale
  INTCONbits.INT0IF=0;} // acquitte IT
}
```

Début du programme d'IT

```
Si (bit drapeau xxIF* = 1)
    Prendre décision pour demande venant de x
Sinon
    Si ( bit drapeau yyIF* =1)
        Prendre décision pour demande venant de y
    Sinon
        Si ( bit drapeau zzIF* =1)
            Prendre décision pour demande venant de z
        Sinon
            Prendre décision pour demande venant du dernier périphérique
        Fin de si
    Fin de si
Fin de si
Fin du programme d'IT
```

Programmation

- **Validation des IT**
  - À mettre dans le programme principal
  - Placement avant le for(;;)
  - Les étapes

1 seul niveau d'IT

1- Un seul niveau d'IT
RCONbits.IPEN=0;
2- validation IT globales:
INTCONbits.GIE=1
3- validation IT périphériques
INTCONbits.PEIE= 1 ou 0
4- validation individuelle des IT
Validation du périphérique xxx**: xxxxIE=1**
5- acquittement des IT
Si IT en suspens l'acquitter : **xxxIF=0**

```
// exemple bouton poussoir sur RBO en IT
RCONbits.IPEN=0; // 1 seul nv
INTCONbits.GIE=1; // IT autorisé
INTCONbits.PEIE=0; // pas de périph en IT
INTCONbits.INT0IE=1; // IT bouton poussoir autorisé
INTCON2bits.INTEDG0=0; // choix du front
INTCONbits.INT0IF=0; // acquittement init
```

2 niveaux d'IT

1- Deux seul niveau d'IT
RCONbits.IPEN=1;
2- validation IT hautes:
INTCONbits.GIEH=1 ***(ou INTCONbits.GIE=1)***
3- validation IT périphériques
INTCONbits.GIEL=1 ***(ou INTCONbits.PEIE= 1)***
4 – Pour chaque source d'IT choisir si priorité haute ou basse
xxxxIP=…….; //1= haute priorité 0=basse

5- validation individuelle des IOT
Validation du périphérique xxx**:** **xxxxIE=1**
6- acquittement des IT
Si IT en suspens l'acquitter : **xxxIF=0**

Type d'action dans votre ISR

- Traitement immédiat de l'événement
  - ✓ Le traitement associé à l'interruption est effectué directement dans la routine d'interruption -Voir page précédente
  - ✓ *Utilisé pour des événements urgents dont le traitement est court*
- Traitement différé de l'événement
  - ✓ La routine d'interruption positionne simplement une variable(globale) qui sera utilisée par une tâche pour effectuer le traitement associé.
  - ✓ Le temps de réponse maximum est alors le temps de réponse du for(;;)

Volatile : indique au compilateur une interdiction d'optimisation sur cette variable
A utiliser lorsque la valeur d'une variable change indépendamment du programme
- zone mémoire mappée sur un périphérique: exemple PORT,LAT etc
- Variable globale modifiée par une interruption

volatile unsigned char it;

Variable utilisateur (flag) pour détecter un passage dans l'ISR

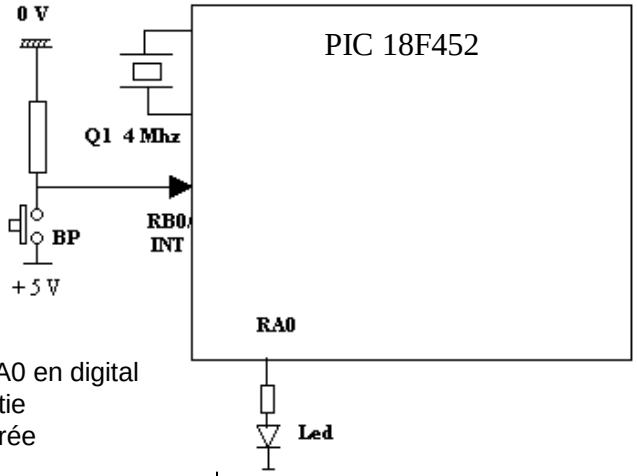
```
unsigned char it;
void main(void) {
    ......;
    for( ;;) {
        ..... ;
        if ( it )
        {
           traitement ;
           it=0;
        }
        ..... ;
           }
}
```

```
#pragma code
#pragma interrupt isr
void isr(void)
{
  if(bit F){
    it = 1 ;
    RAZ bit F ; }
}
```

Programmation

- **Un exemple**

```
#include <p18cxxx.h>
// configuration
#pragma config OSC = HS
#pragma config WDT = OFF  .
#pragma config LVP = OFF .
#pragma config PBADEN = OFF
#include <stdio.h>
#include "xlcd200V.h"
#define LED  LATAbits.LATA0
unsigned char etat_led;
Volatile unsigned char flagIT_RB0;
void isr_it_h(void);
void main()
{
        ADCON1=ADCON1| 0x0F; // RA0 en digital
        TRISA.TRISA0=0; //RA0 en sortie
        TRISB.TRISB0=1; //RB0 en entrée
        RCONbits.IPEN=0; // config IT
        INTCONbits.GIE=1;
        INTCONbits.PEIE=0;
        INTCONbits.INT0IE=1;
        INTCON2bits.INTEDG0=0;
        INTCONbits.INT0IF=0;
        OpenXLCD(OPEN_PICDEM_LCD);
        stdout = _H_USER ;
        clearXLCD();
        LED=0;
        for(;;)
                {
                if (flagIT_RB0 ==1)
                  {
                  LED= !LED ;
                  gotoXLCD(LCD_LINE_ONE);
                  printf("la led est: %d",LED);
                  flagIT_RB0 = 0 ;
                  }
                }
}
```




```
#pragma code vect_it_h=0x08
void_vect_it_h (void)
{
        _asm
        goto isr_it_h
        _endasm
}
#pragma code
#pragma interrupt isr_it_h
void isr_it_h (void)
{
        if (INTCONbits.INT0IF=1){
         flagIT_RB0=1;;
        INTCONbits.INT0IF=0;
        }
}
```