Les interruptions

GENERALITES

Mécanisme d'interruption

L'interruption est un mécanisme fourni par un microprocesseur ou un système informatique pour gérer les conditions d'erreur, les événements d'urgence ou coordonner l'utilisation des ressources partagées, etc. Les interruptions sont très utiles dans des nombreuses situations de programmation ; sans ce mécanisme, plusieurs opérations deviendront soit impossibles, soit très difficiles à mettre en œuvre. Citons à titre d'exemple quelques opérations qui nécessitent l'utilisation des interruptions :

- Réponse à des événements critiques (bouton d'urgence, capteur de fumé…)
- Réception d'une trame de données sur le port série.
- Exécution des tâches périodiques (Acquisition de données, affichage multiplexé…)
- Ordonnancement des tâches dans un système d'exploitation.

Une interruption est un événement qui nécessite que la CPU arrête l'exécution normale du programme et exécute un sous-programme lié à l'événement. Autrement dit, les interruptions sont utilisées pour favoriser l'exécution des tâches considérées prioritaires par rapport aux autres. La tâche d'interruption est appelée communément routine d'interruption (ou ISR pour *Interrupt Service Routine*). On distingue deux types d'interruptions :

- Interruption matérielle : c'est un événement asynchrone (signal électrique) provenant d'un périphérique externe ou interne qui traduit une demande de service auprès de la CPU. Une fois l'événement détecté, la CPU arrête le traitement en cours et procède à l'exécution de la routine d'interruption.
- Interruption logicielle appelée aussi « exception » : celle-ci est causée, soit par l'appel d'une instruction spéciale (instruction `RESET` pour PIC18 par exemple), soit suite à une erreur logicielle (division par zéro par exemple) ou bien l'appel des fonctions systèmes (INT21H du DOS par exemple). Contrairement à une interruption matérielle, une interruption logicielle est appelée explicitement par le programme.

Dans un langage haut niveau, tel que MPLAB CX8, la partie déclarative d'une routine d'interruption est une fonction C de type **`void`**, dont on ajoute dans le prototype le mot clé **`__interrupt().`** Le nom qui suit le mot clé n'est plus utilisé, puisqu'elle n'est pas appelée explicitement. Un exemple de code d'interruption de haute priorité est donné ci-dessous :

```
void __interrupt(high_priority) TimerISR(void)
{
  if (INTCONbits.TMR0IE && INTCONbits.TMR0IF) {// Interruption timer 0 ?
        INTCONbits.TMR0IF = 0;
        ……….;        // code de la routine d'interruption
  }
}
```

INTERRUPTIONS DU MICROCONTROLEUR PIC18F4520

Le microcontrôleur PIC18F4520 dispose de 20 sources d'interruptions. Ces interruptions sont fondamentalement divisées en deux types, à savoir les interruptions externes et internes. Quatre interruptions externes sont déclenchées par des périphériques externes via les broches d'interruption **RB0/INT0**, **RB1/INT1**, **RB2/INT2** et **RB4-RB7**. Les seize interruptions restantes sont activées en interne par des périphériques intégrés (ADC, Timers, USART …).

Priorité des interruptions

L'interruption consiste à interrompre l'exécution du programme principal (appelé aussi *tâche de fond* ou *tâche d'arrière-plan*) pour exécuter une routine d'interruption considérée comme prioritaire. La priorité de la routine d'interruption par rapport à la tâche de fond est évidente. La priorité que nous considérons, est la priorité entre les sources interruptions. Lorsque plusieurs événements se produisent en même temps, quelle interruption sera servie la première ? Lorsqu'une interruption est en cours d'exécution, peut-elle être interrompue par une interruption plus prioritaire ? De même cette seconde interruption peut-elle aussi être interrompue par une autre plus prioritaire… A quel niveau de priorité (d'imbrication) doit-on s'arrêter ?

Généralement, à chaque interruption, un numéro de priorité lui est attribué; ce numéro fixe le niveau de priorité de l'interruption. Le nombre de niveaux de priorité est généralement fixé par le constructeur du composant. Dans la plupart des

processeurs 8 bits, tel que le PIC18, le nombre des niveaux de priorité des interruptions masquables est de 1 ou 2.

Un seul niveau (cas des PIC16) : toutes les interruptions ont le même niveau (le même numéro de priorité). La notion de *priorité n'existe pas* ; les interruptions ne peuvent pas s'interrompre.

Deux niveaux de priorité (cas des PIC18) : il y a deux numéros de priorité ; une interruption peut avoir le numéro "**0**" ou "**1**". Une interruption ayant, le numéro "0" se classe dans le ***niveau de basse priorité*** et celle de numéro "1" se classe dans le ***niveau de haute priorité***. Les interruptions de même niveau ne s'interrompent jamais. Une interruption de haute priorité peut interrompre une interruption de basse priorité et non pas le contraire. Pour les microcontrôleurs PIC18, un bit dénommé **IP** (**I**nterrupt **P**riority) est attribué à chaque interruption pour sélectionner le niveau de priorité.

Lorsque plusieurs demandes d'interruptions se présentent simultanément, certainement, l'interruption la plus prioritaire l'emporte ; mais au cas où les interruptions partagent le même niveau de priorité ; dans ce cas, deux possibilités peuvent être envisagées :

- Si les interruptions sont vectorisées, chaque interruption a sa propre adresse, la sélection de l'interruption à exécuter dépend d'un classement fixé par le constructeur.
- Si les interruptions partagent la même adresse, cas de notre microcontrôleur, la sélection dépend de l'ordre de test des indicateurs (Flags) dans la routine d'interruption. Cette flexibilité se paye par l'augmentation du temps de réponse.

Procédure d'interruption

Le microcontrôleur PIC18F4520 ne dispose que de deux vecteurs d'interruption pour gérer toutes les demandes d'interruptions. Ces interruptions sont divisées en deux niveaux de priorité, haute et basse. Le vecteur d'interruption de haute priorité est situé à l'adresse **0x0008** et le vecteur de faible priorité à l'adresse **0x0018**. Au sein de chaque niveau de priorité, la routine d'interruption doit vérifier les bits indicateurs (Flags) afin d'identifier la source d'interruption en attente.

Les microcontrôleurs PIC18, utilisent 10 registres pour le contrôle des interruptions :

RCON
INTCON
INTCON2
INTCON3
PIR1, PIR2
PIE1, PIE2
IPR1, IPR2

Chaque source d'interruption est contrôlée par trois bits :

- Un bit ***Flag*** **(IF : I**nterrupt **F**lag**)**, indique qu'un événement d'interruption s'est produit.
- Un bit ***Enable*** **(IE : I**nterrupt **E**nable**)**, autorise la prise en compte de la demande d'interruption.
- Un bit **Priority (IP : I**nterrupt **P**riority**)**, Sélectionne le niveau de priorité (haute ou bien basse).

Bit IPEN du registre RCON

Le PIC18 dispose de deux niveaux de priorité. Cependant, le fait de placer toutes les interruptions dans un même groupe, revient à n'utiliser qu'un seul niveau. Le bit **IPEN** (RCON<7> : bit 7 du registre RCON), permet d'activer ou désactiver la logique de priorité. Lorsque `IPEN = 1`, la priorité des interruptions est activée. `IPEN = 0` (état par défaut) la priorité des interruptions n'est pas prise en compte (compatibilité avec la famille PIC16) et par conséquent, toutes les interruptions admettent l'unique adresse **0x0008** comme vecteur d'interruption. La logique de priorité peut être programmée en exécutant l'instruction « `RCONbits.IPEN = 1 ;` » en langage XC8.

Les traces d'héritage

La logique de contrôle des interruptions porte les traces de compatibilité avec les versions précédentes. Lorsque Microchip a introduit la famille PIC16, les premières versions ne disposaient que quatre sources d'interruptions :

- **INT** : interruption externe sur la broche RB0
- **RB** : interruption externe sur les broches RB4 à RB7
- **T0** : interruption du périphérique interne TIMER0
- Une interruption spécifique au composant, (validation de fin d'écriture sur l'EEPROM pour le PIC16F84)

Les bits indicateurs (**INTF**, **RBIF**, **T0IF**), les bits de validations individuelles (**INTE**, **RBIE**, **T0IE**) et le bit de validation globale **GIE**, forment les sept bits du registre **INTCON** (INTerrupts CONtrol) ; le bit restant est spécifique au composant.

Des versions plus récentes ont ajouté des périphériques supplémentaires, qui peuvent générer plusieurs interruptions. Microchip a conservé la structure de trois interruptions initiales. Les périphériques additionnels forment un groupe logique avec leur propre bit de validation globale **PEIE** (*PEripheral Interrupt Enable bit)* ; tous les bits indicateurs (Flags) sont situés dans des registres **PIRx** (*Peripheral Interrupt Request*) et les bits de validations individuelles dans les registres **PIEx** (*Peripheral Interrupt Enable*). Le nombre de ces registres dépend du nombre des périphériques intégrés ; le PIC16F877 dispose de deux registres PIR1, PIR2 et deux registres PIE1, PIE2. Toutes les interruptions des périphériques additionnelles admettent un bit de validation globale **PEIE** (*PEripheral Interrupt Enable bit*).

Le bit **GIE** (*Global Interrupt Enable*) commande toutes les interruptions (initiales et additionnelles). Le bit **PEIE** ne commande que les interruptions des périphériques additionnelles. Le schéma bloc de la logique d'interruption est illustré à la Figure 18, la logique d'interruption est divisée en deux groupes : les interruptions externes avec le TIMER0, qui sont considérées comme interruptions de base ; et les interruptions dites des périphériques.

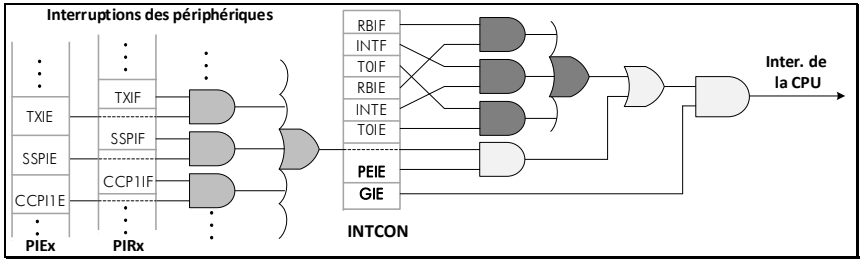

Figure 18 : schéma bloc de la logique d'interruption de PIC16

La logique d'interruptions des PIC18 est assez similaire. En mode compatible (IPEN = 0), une légère modification est apportée au bloc d'interruptions de base.

En mode priorité (IPEN = 1), c'est pratiquement une duplication du mode compatibilité, l'un pour le niveau haut et l'autre pour le niveau bas.

Validation globale des interruptions

En mode compatibilité (IPEN = 0)

La priorité des interruptions est désactivée, un seul bit de validation globale **GIE** (INCON<7>) pour toutes les interruptions, et un bit de validation **PEIE** (INCON<6>) pour les interruptions des périphériques.

En mode priorité (IPEN = 1)

La priorité des interruptions est activée, il y a deux niveaux de priorité. Les mêmes bits de validation globale GIE et PEIE sont utilisés avec des appellations différentes : **GIEH** (**G**lobal **I**nterrupt **E**nable **H**igh priority) et **GIEL** (**G**lobal **I**nterrupt **E**nable **L**ow priority).

- **GIEH** (INCON<7>) : active ou désactive toutes les interruptions de haute et de basse priorité.
- **GIEL** (INCON<6>) : active ou désactive toutes les interruptions de basse priorité.

Les bits 7 et 6 du registre INTCON sont dénommés **GIE/GIEH** et **PEIE/GIEL**.

Interruptions INT0, INT1, INT2, RB et TMR0

Le bloc des interruptions de base est enrichi de deux interruptions externes INT1 et INT2. Les bits indicateurs et de validations individuelles (INT0IF, INT0IE, RBIF, RBIE, TMR0IF, TMR0IE) ainsi que les bits de validations globales GIE/GIEH et PEIE/GIEL ont conservé leur emplacement dans le registre INTCON.

Les interruptions externes sur les broches RB0/INT0, RB1/TNT1 et RB2/INT2 sont sensibles au front montant ou bien descendant. Les bits INTEDG0, INTEDG1 et INTED2 correspondants dans le registre INTCON2 permettent le choix du front. A la mise sous tension, ces interruptions sont actives au front montant (INTEDGx = 1). Lorsqu'un front valide est détecté sur une broche RBx/INTx, le bit indicateur correspond INTxIF est positionné et une interruption pourra être générée, si le bit INTxIE correspondant est défini. Le bit indicateur INTxIF doit être remis à zéro dans la routine d'interruption.

La priorité d'interruption pour INT1 et INT2 est déterminée par les bits de priorité d'interruption INT1IP et INT2IP du registre INCON3. L'interruption INT0 est considérée toujours de haute priorité, par conséquent aucun bit de prioritaire n'est associé.

L'interruption RB est utilisée dans l'interfaçage du clavier. Le PIC18F4520 fournit quatre broches d'interruption sur changement d'état KBI0 à KBI3 multiplexées avec RB4 à RB7 du PORTB. Une transition positive ou négative sur l'une ou plus de ces broches mettra l'unique bit indicateur RBIF (INTCON<0>) à 1. Notez bien qu'un seul bit indicateur est affecté aux quatre sources interruption. L'interruption peut être activée ou désactivée par le bit RBIE (INTCON<3>). La priorité d'interruption est déterminée par le bit de priorité RBIP (INTCON2<0>).

Le système de gestion d'interruption du TMR0 dispose d'un bit indicateur TMR0IF, d'un bit de validation TMR0IE et d'un bit priorité TMR0IP. Le TIMER0 sera étudié en détail dans le chapitre consacré aux Timers.

Tous les bits de contrôle des interruptions de base sont implémentés dans les registres INTCON, INTCON2 et INTCON3. Il est important de savoir que lorsque la logique de priorité est activée, l'utilisateur doit mettre à 1 les bits GIEH et GIEL pour activer les interruptions de faible priorité. Positionner le bit GIEL sans positionner le bit GIEH n'activera aucune interruption de faible priorité.

REGISTRE INTCON

b7 R/W-0	b6 R/W -0	b5 R/W-0	b4 R/W-0	b3-R/W-0	b2 R/W -0	b1 R/W -0	b0 R/W -x
GIE/GIEH	**PEIE/GIEL**	**TMR0IE**	**INT0IE**	**RBIE**	**TMR0IF**	**INT0IF**	**RBIF**

REGISTRE INTCON2

b7 R/W-1	b6 R/W -1	b5 R/W -1	b4 R/W -1	b3 U -0	b2 R/W -1	b1 U -0	b0 R/W -1
$\overline{RBPU}$	**INTEDG0**	**INTEDG1**	**INTEDG2**	**-**	**TMR0IP**	**-**	**RBIP**

REGISTRE INTCON3

b7 R/W-1	b6 R/W -1	b5 U -0	b4 R/W -0	b3 R/W -0	b2 U -0	b1 R/W -0	b0 R/W -0
INT2IP	**INT1IP**	**-**	**INT2IE**	**INT1IE**	**-**	**INT2IF**	**INT1IF**

GIE/GIEH : Global Interrupt Enable bit

Priorité activée (IPEN = 1)	Priorité désactivée (IPEN = 0)
1 : valide les interruptions de haute priorité 0 : désactive toutes les interruptions	1 : valides toutes les interruptions non masquées 0 : désactive toutes les interruptions

PEIE/GIEL : PEripheral Interrupt Enable bit

Priorité activée (IPEN = 1)	Priorité désactivée (IPEN = 0)
1 : valide les interruptions de basse priorité 0 : désactive les interruptions de basse priorité	1 : valides les interruptions périphériques 0 : désactive toutes les interruptions des périphériques

Interruption	Bit Indicateur	Bit de Validation	Bit de Priorité	Front d'activation
RB0/INT0	INT0IF	INT0IE	-	INTEDG0
RB1/INT1	INT1IF	INT1IE	INT1IP	INTEDG1
RB2/INT2	INT2IF	INT2IE	INT2IP	INTEDG2
RB4..RB7	RBIF	RBIE	RBIP	-
TIMER 0	TMR0IF	TMR0IE	TMR0IP	-

$\overline{RBPU}$: PORTB Pull-up Enable bit

1 : les résistances pull-up internes du PORTB sont désactivées

0 : les résistances pull-up internes du PORTB sont activées

Exemple 1

Etant donné le schéma de la Figure 19. Nous voulons changer l'état de la LED à chaque action sur le bouton poussoir BP. Ecrire un programme en langage XC8 permettant de gérer l'entrée RB1/INT1 par interruption.

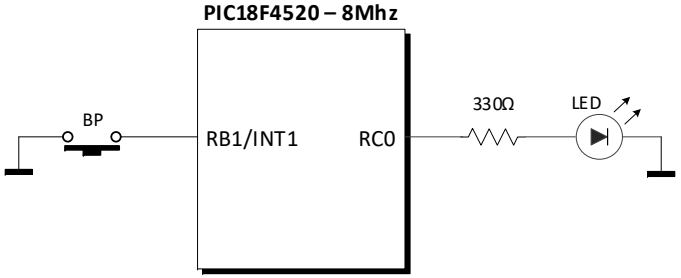

Figure 19

Solution

- Par défaut la broche RB1 est configurée en analogique, il faut la configurer en numérique.
- Configurez le broche RB1 en entrée et la broche RC0 en sortie.
- Activez les résistances Pull Up interne du PORTB.
- Une seule interruption, l'activation de la logique de priorité n'a pas de sens ; Par défaut IPEN = 0.
- L'action sur le bouton poussoir, entraine une transition négative ; nous devons activez l'interruption INT1 sur le front descendant.
- Mettre les bits de validation individuelle et globale de INT1 à 1.

```
#include <xc.h>
#define _XTAL_FREQ 8000000
#pragma config OSC = HS, WDT = OFF, LVP = OFF
#define LED PORTCbits.RC0
void __interrupt() isr_int1(void)
{   INTCON3bits.INT1F = 0;      // remettre le flag à 0
    LED = !LED;                 // changer l'état de la LED
}
```

```
void main(void) {
    ADCON1 = 0x0F;              // configurer les PORTB en numérique
    TRISBbits.RB1 = 1;          // RB1 en entrée
    TRISCbits.RC0 = 0;          // RC0 en sortie
    INTCON2bits.nRBPU = 0;      // Activez les résistances Pull Up
    INTCON2bits.INTEDG1 = 0;    // Activez INT1 sur front descendant
    INTCON3bits.INT1IE = 1;     // Validation individuelle de INT1
    INTCONbits.GIE = 1;         // Validation globale
    while(1) continue;          // boucle infinie
}
```

Exemple 2

La Figure 20 montre un système de stockage temporaire équipé de deux tapis roulants. Le tapis roulant 1 transporte les paquets à l'espace de stockage, tandis que le second transporte les paquets qui en sortent à une rampe de chargement. Deux barrières photoélectriques à l'extrémité de chaque tapis roulant permettent de détecter le nombre de paquets qui entrent dans l'espace de stockage et ceux qui en sortent.

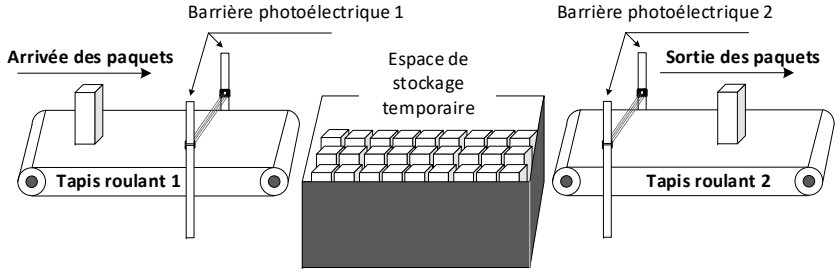

Figure 20

Supposons que les capteurs photoélectrique 1 et 2 sont connectés respectivement aux entrées RB0 et RB1 d'un microcontrôleur PIC18F4520 cadencé à 8 MHz. Nous voulons écrire un programme qui permet de déterminer le nombre des paquets contenus dans l'espace de stockage. La variable *CntPaquets* sera incrémentée lors de la détection de l'entrée d'un paquet dans l'espace de stockage, et décrémentée lorsqu'il en sorte.

Solution

Compte tenu de la fréquence rapide de scrutation des entrées, la variable compteur se trouve incrémentée ou décrémentée plusieurs fois lorsqu'un seul paquet passe devant un capteur photoélectrique. La solution correcte, consiste à détecter le changement de niveau (détection du front montant ou descendant). Un exemple de

détection de front montant a été présenté dans le chapitre précédent. Cependant, la méthode interruption que nous allons utiliser s'avère plus intéressante.

```
#include <xc.h>
#define _XTAL_FREQ 8000000
#pragma config OSC = HS, WDT = OFF, LVP = OFF
char CntPaquets = 0;
void __interrupt() isr_int01(void) {
   if(INTCONbits.INT0F == 1) {
        INTCONbits.INT0F = 0;   // remettre le flag INT0IF à 0
        CntPaquets++;           // Incrémenter la variable compteur
   }
   if(INTCON3bits.INT1F == 1) {
        INTCON3bits.INT1F = 0;  // remettre le flag INT1IF à 0
        CntPaquets--;           // Décrémenter la variable compteur
   }
}
void main(void) {
    ADCON1 = 0x0F;              // configurer les PORTB en numérique
    TRISB = 0xFF;               // PORTB en entrée
    INTCONbits.INT0IE = 1;      // Validation individuelle de INT0
    INTCON3bits.INT1IE = 1;     // Validation individuelle de INT1
    INTCONbits.GIE = 1;         // Validation globale des ITs
    INTCONbits.PEIE = 1;        // Validation globale des ITs périphé.
    while(1) continue;
    return;
}
```

Les interruptions périphériques

Comme nous avons mentionné précédemment, chaque interruption est contrôlée par trois bits : bit ***Flag***, bit ***Enable*** et bit ***Priority***. Pour les interruptions des périphériques, ces bits sont situés respectivement dans les registres **PIRx, PIEx** et **IPRx**.

Registres indicateurs (FLAGS)

Registre PIR1

b7	b6	b5	b4	b3	b2	b1	b0
R/W-0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
PSPIF	**ADIF**	**RCIF**	**TXIF**	**SSPIF**	**CCP1IF**	**TMR2IF**	**TMR1IF**

Registre PIR2

b7	b6	b5	b4	b3	b2	b1	b0
R/W-0	R/W -0	U -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
OSCFIF	**CMIF**	**-**	**EEIF**	**BCLIF**	**HLVDIF**	**TMR3IF**	**CCP2IF**

Registres de validation (ENABLE)

Registre PIE1

b7	b6	b5	b4	b3	b2	b1	b0
R/W-0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
PSPIE	**ADIE**	**RCIE**	**TXIE**	**SSPIE**	**CCP1IE**	**TMR2IE**	**TMR1IE**

Registre PIE2

b7	b6	b5	b4	b3	b2	b1	b0
R/W-0	R/W -0	U -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
OSCFIE	**CMIE**	**-**	**EEIE**	**BCLIE**	**HLVDIE**	**TMR3IE**	**CCP2IE**

Registres de priorité (PRIORITY)

Registre IPR1

b7	b6	b5	b4	b3	b2	b1	b0
R/W-1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -0	R/W -1
PSPIP	**ADIP**	**RCIP**	**TXIP**	**SSPIP**	**CCP1IP**	**TMR2IP**	**TMR1IP**

Registre IPR2

b7	b6	b5	b4	b3	b2	b1	b0
R/W-1	R/W -1	U -0	R/W -1	R/W -1	R/W -1	R/W -0	R/W -1
OSCFIP	**CMIP**	**-**	**EEP**	**BCLIP**	**HLVDIP**	**TMR3IP**	**CCP2IP**