Les interruptions du PIC16F877

cours n°4
LI326

Plan

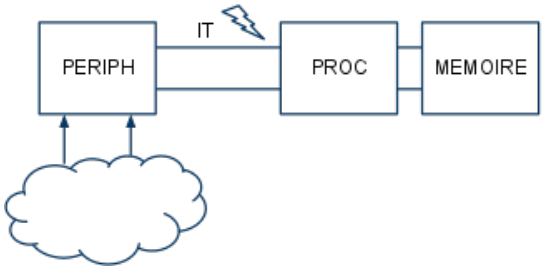
- Qu'est-ce qu'une interruption
- Cas particulier du p16f877
- Echange avec le programme principal
- Tâches ?

Qu'est-ce qu'une interruption

Une interruption est une demande d'un périphérique matériel de traitement d'un événement par le processeur.

- Le programme normal est interrompu
- L'état du processeur est sauvegardé
- Le traitement de l'interruption est effectué
- L'état du processeur est restauré
- Le programme normal est répris

Le périphérique a "volé" des cycles au programme normal



Acquittement d'une interruption

- Le programme normal n'est pas toujours interruptible. Il peut être en train de modifier l'état de ses données et demander au processeur de différer le traitement des interruptions.

- Un périphérique doit maintenir sa ligne d'interruption jusqu'à ce que le traitement demandé soit réalisé.

- Le processeur peut "masquer" ses lignes d'interruptions.

- Lors du traitement d'une interruption, il faut acquitter l'interruption en écrivant dans le périphérique pour lui demander de "baisser" sa ligne.

Routine d'interruption : ISR

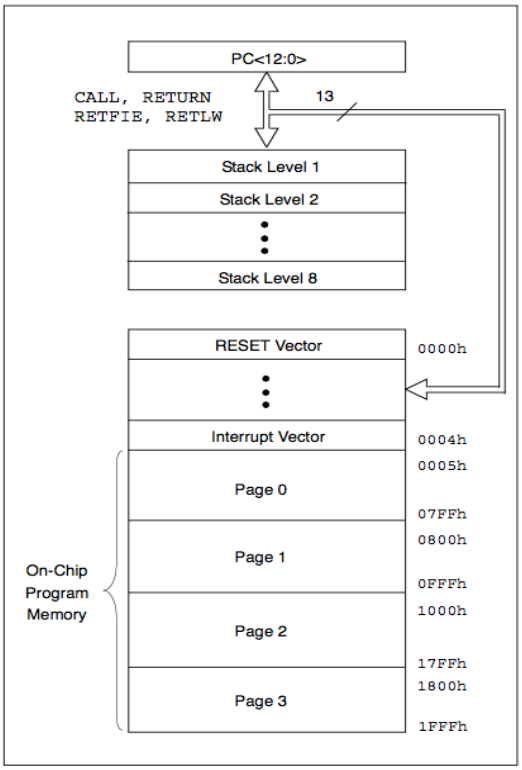
- Le traitement d'une interruption est effectué par une ISR
  Interrupt Service Routine.
- En général, on a autant d'ISR que de type d'interruption.
- Un périphérique peut produire plusieurs types d'interruption.
  par exemple: le terminal peut lever une interruption si une touche est tapé au clavier et une interruption si l'écran attend un nouveau caractère à afficher.
- En général, l'objectif d'une ISR est de lire ou d'écrire des donnée dans le périphérique concerné, puis d'acquitter l'interruption pour que le périphérique baisse la ligne.

Gestionnaire d'interruption

- Le morceau de programme qui gère le déroutement du programme normal vers les routines ISR se nomme: gestionnaire d'interruption (interrupt handler).
- Le gestionnaire fait 4 choses.
  1. Il sauve le contexte pour ne pas perturber le programme interrompu (il ne doit se rendre compte de rien)
  2. Il analyse la cause d'interruption pour choisir la bonne ISR
  3. Il appelle l'ISR
  4. Il restaure le contexte
- L'exécution du gestionnaire est toujours faîte avec les interruptions masquées.
- L'adresse du gestionnaire est nommé vecteur d'interruption

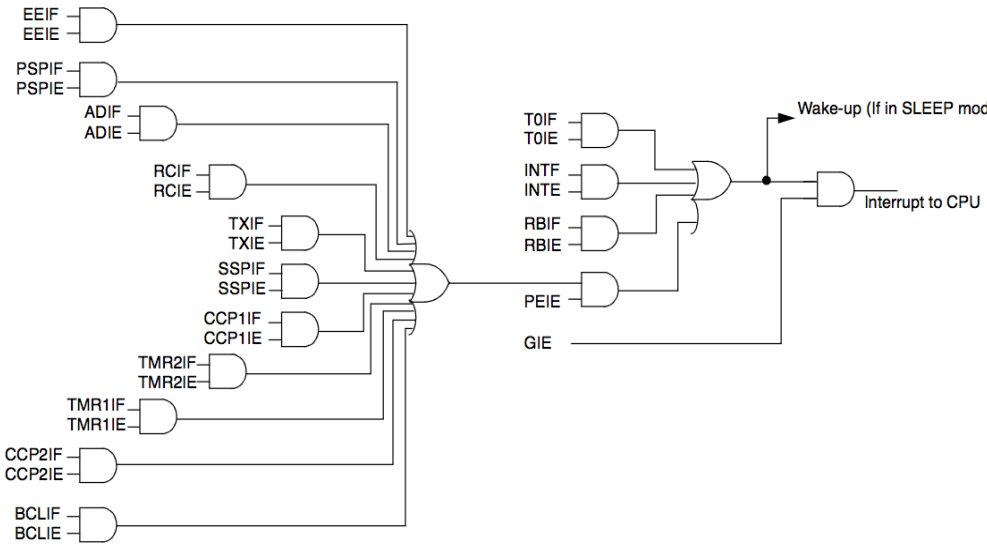
Cas du p16f877 : Vecteurs

- Le vecteur d'interruption est : 4 pour les interruptions normales
- Le vecteur du reset est 0
  Le reset peut être considéré comme une interruption non masquable.
- L'adresse de retour est copié dans la pile des adresses.



Cas du p16f877 : Sources

Le p16f877 dispose de 14 sources d'interruptions.

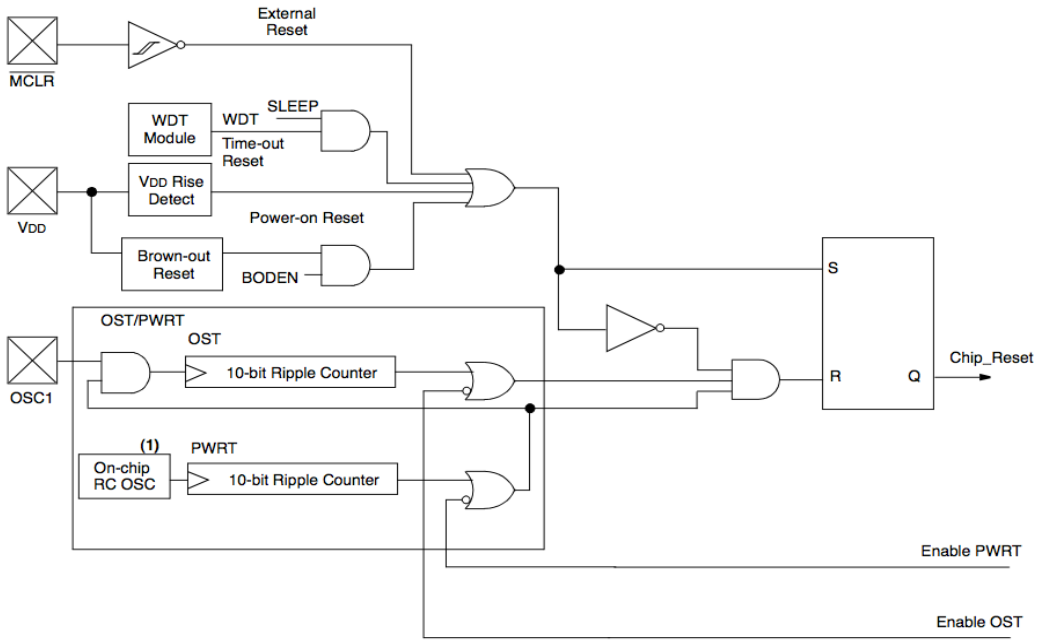


Pour chaque source, il y a 2 bits de contrôle : IF et IE
Il y a 2 masques globaux : GIE et PIEE

T0	Timer 0
INT	Broche d'interruption
RB	Changement d'état des broches
EE	EEprom
PSP	Port parallèle
AD	Convertisseurs analogique
RC	Entrée série
TX	Sortie série
SSP	I2C / SPI
CCP	Capture 1
TMR2	Timer 2
TMR1	Timer 1
CCP2	Capture 2
BCL	Collision de bus

Cas du p16f877 : Source du reset

4 sources de reset qui peuvent se gérer comme les interruptions mais on ne va pas écrire de gestionnaire de reset en TME



Cas du p16f877 : Registres de contrôle

	File Address		File Address		File Address		File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h	General Purpose Register 16 Bytes	118h	General Purpose Register 16 Bytes	198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes	
	7Fh	accesses 70h-7Fh	EFh F0h FFh	accesses 70h-7Fh	16Fh 170h 17Fh	accesses 70h - 7Fh	1EFh 1F0h 1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

PIR2 REGISTER (ADDRESS 0Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	Reserved	—	EEIF	BCLIF	—	—	CCP2IF
bit 7							bit 0

PIE1 REGISTER (ADDRESS 8Ch)

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(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

PIE2 REGISTER (ADDRESS 8Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	Reserved	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

Sauvegarde et restauration

Les registres à sauver au minimum sont :

- W, STATUS et PCLATH : obligatoire
- FSR : optionnel

```
MOVWF     W_TEMP            ;Copy W to TEMP register
SWAPF     STATUS,W          ;Swap status to be saved into W
CLRF      STATUS            ;bank 0, regardless of current bank, Clears IRP,RP1,RP0
MOVWF     STATUS_TEMP       ;Save status to bank zero STATUS_TEMP register
MOVF      PCLATH, W         ;Only required if using pages 1, 2 and/or 3
MOVWF     PCLATH_TEMP       ;Save PCLATH into W
CLRF      PCLATH            ;Page zero, regardless of current page
:
:(ISR)                      ;(Insert user code here)
:
MOVF      PCLATH_TEMP, W    ;Restore PCLATH
MOVWF     PCLATH            ;Move W into PCLATH
SWAPF     STATUS_TEMP,W     ;Swap STATUS_TEMP register into W
                            ;(sets bank to original state)
MOVWF     STATUS            ;Move W into STATUS register
SWAPF     W_TEMP,F          ;Swap W_TEMP
SWAPF     W_TEMP,W          ;Swap W_TEMP into W
```

- La sortie du gestionnaire est : retfie

Analyse de la cause

- L'analyse de la cause consiste à lire les drapeaux non masqués des périphériques susceptibles de lever une interruption et d'appeler les ISR correspondantes.
- Le plus simple est que les ISR soient interruptibles donc courtes.
- L'ISR doit toujours acquitter l'interruption en :
  baissant le drapeau ou en utilisant le périphérique
- On peut ne tester qu'une seule cause en faisant l'hypothèse que les interruptions sont courtes et ne sont pas simultanées généralement.
- Les ISR doivent utiliser des registres propres pour ne pas écraser des données du programme interrompu.

Routine d'interruption : ISR

- La durée d'une ISR doit être bornée puisque non interruptible.
- L'acquittement de l'interruption dépend du périphérique:
  - pour le timer 0, il faut mettre le bit T0IF à 0.
  - pour l'entrée du port série, il faut lire la donnée reçue.
- L'echange avec le programme principal se fait par des canaux de communications simples constitués de :
  - un buffer : ça peut être un registre ou plusieurs
  - un drapeau : un bit dans un registre réservé pour ça
- Les canaux de communications sont nommés des événements

Echanges par événement

- Les événements sont un couple drapeau + buffer
  - Le drapeau permet de connaitre l'état du buffer: p.ex. 0/1 (vide/plein)
  - Le buffer permet de transporter de l'information
- Un événement permet à deux *"tâches"* de communiquer: ici les tâches sont: les ISR et le programme principale main.
- L'état du drapeau permet de savoir qui a le droit d'utiliser le buffer entre l'ISR et le main.

Exemple d'échange : sortie écran

- On suppose que l'on a un périphérique de sortie fonctionnant de la manière suivante:
  - on l'initialise pour définir, par exemple, le débit.
  - on dispose d'un registre de sortie tel qu'une écriture dans ce registre provoque l'émission de la donnée écrite à faible débit.
  - Une interruption est levée lorsque le registre d'échange est vide.

- On veut envoyer un message complet:
  - on réserve un buffer de 16 caractères (p.ex.)
  - on reserve une variable index qui indique le n° du caractère à envoyer
  - on reserve un bit drapeau initialisé à 0

    - BUFFER[16]
    - INDEX
    - FLAG

Exemple d'échange : sortie écran

Le périphérique écran est controlé grâce à 3 registres

- IE_ecran
- IF_ecran
- data_ecran

```
main
    tant que (FLAG !=0);
    copy (BUFFER, "hello");
    INDEX = 0;
    DEMASQUER IE_ecran ;
    FLAG = 1;
```

```
isr
    c = BUFFER[INDEX] ;
    si c == 0 alors
        MASQUER IE_ecran ;
        FLAG = 0;
    sinon
        INDEX++ ;
        data_ecran = c ;
    fsi
```

On suppose que l'écrire dans data_ecran acquite IF_ecran

Forme général du programme

Nous allons vous proposer une forme général pour les programmes qui vous guidera pour l'ensemble de TME

```
    org 0
    goto init

    org 4
    SAVE_CONTEXT
    ANALYSE_CAUSE
        call isr1
    REST_CONTEXT
    retfie

init
    PROGRAM_PERIPH
    DEMASQUE_PERIPH
    bsf    INTCON,PEIE
    bsf    INTCON,GIE
    goto main
```

```
isr1
    TRAITEMENT
    ACQUITEMENT
    return

isr2
    TRAITEMENT
    ACQUITEMENT
    return

main
    BOUCLE_SANS_FIN
```

Echange avec ou sans perte

- L'échange décrit dans l'exemple est un échange sans perte.Toutes les données écrites par le main (l'écrivain du buffer) sont envoyées par l'ISR (le lecteur du buffer).
- On peut aussi décrire un échange à perte, il suffit que l'écrivain ne tiennent pas compte de l'état du drapeau pour reremplir le buffer.
- C'est utilise lorsque les données à écrire ont une durée de validité limitée

un parfum de multi-tâches

Le programme principal peut être décrit comme une super-boucle qui appelle à tour de rôle des tâches. Chaque tâche traite un événement.

```
main
    call tache1
    call tache2
    ....
    call tache3
    goto main
```

```
tache1
    si drapeau event1 != 1
        return
    usage du buffer
    drapeau event1 = 0
    return
```

Cas particulier du timer

- Les tâches d'un micro-contrôleur sont souvent contrôlées par le temps. Par exemple il faut lire les boutons toutes les 20ms ou prendre la température toutes les minutes...

- On peut utiliser un timer qui va être programmer pour définir une base de temps périodique (p. ex. 10ms) et programmer des timers multiples de cette période dans l'ISR du timer.

- Pour chaque période dérivée, il y aura un drapeau.
  Ce drapeau est levé par l'ISR du timer et baissé par la tâche en attente de la période.