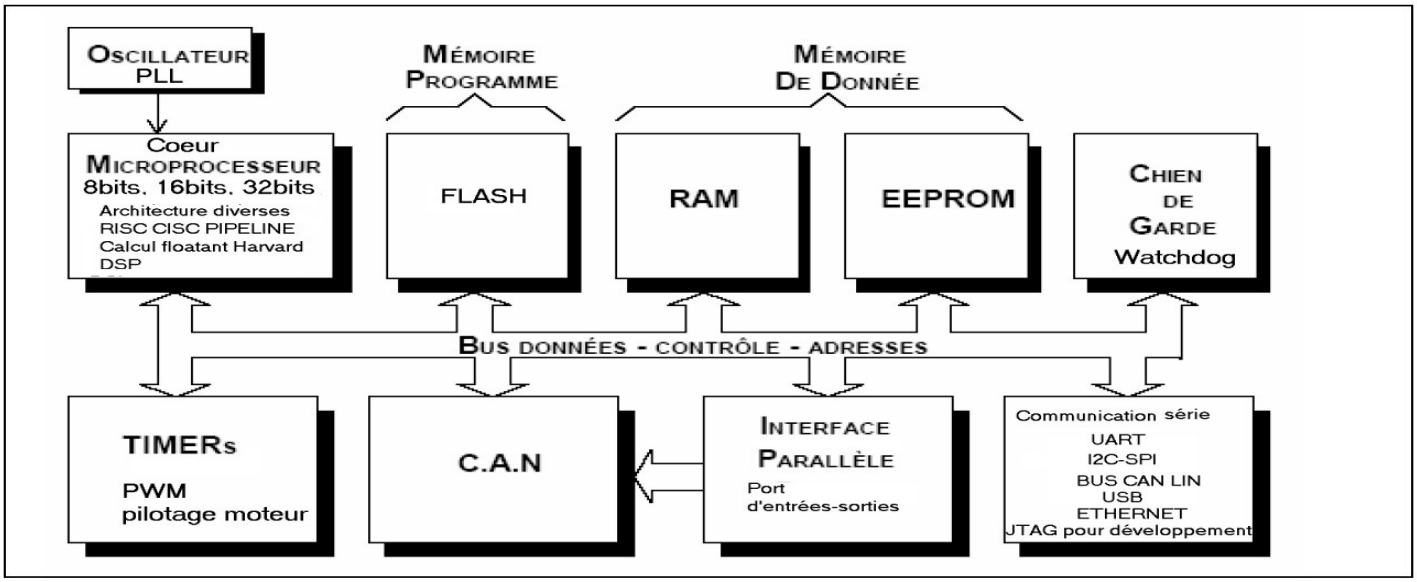
Les microcontrôleurs

(Description simplifiée)

1) Généralités sur les microcontrôleurs

Le marché des microcontrôleurs, qui représente 5% du marché des semi-conducteurs est extrêmement concurrentiel. Plus d'une douzaine d'acteurs se disputent ce marché
Un microcontrôleur est un composant qui intègre sur la même puce un cœur de microprocesseur des périphériques et de la mémoire vive ou morte.



Son utilisation se situe dans le domaine de l'embarqué (l'électronique automobile, les cartes à puce, le médical, la domotique, la gestion de puissance et les compteurs intelligents, l'éclairage à DEL, le photovoltaïque et l'éolien, le téléphone mobile, les consoles de jeu ….). Il existe une très large variété de microcontrôleurs, des plus petits modèles aux fonctions extrêmement basiques, généralement basés sur des architectures 8-bits voir 16bits , au plus imposants, exploitant des architectures 32-bits, en passant par des modèles spécialisés, que ce soit dans le traitement du signal comme les DSP (« Digital Signal Processor », processeur pour signaux analogiques) ou dans la sécurité comme les microcontrôleurs sécurisés. Les principaux acteurs du marché sont :

Analog devices	Fairchild	Temic	philips
Atmel	Infineon	Zilog	Motorola(freescale)
cypress	Maxim	Texas instrument	National semi
Dallas	Microchip	On semiconductor	St microelectronics

Les 6 leaders mondiaux toute gamme confondu dans ce domaine sont :
Renesas (27%), Freescale (9%), Infineon (8%) , Atmel (7%) et STmicroelectronics (6%)

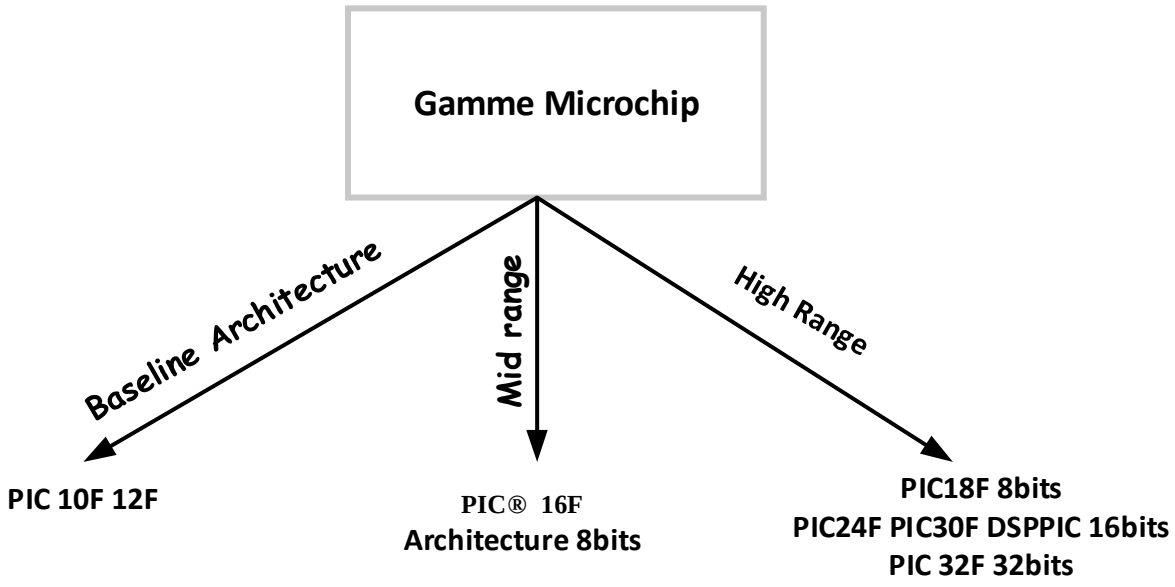
Pour le marché du microcontrôleur haut de gamme 32bits, la plupart des constructeurs font appel à la société britannique **ARM** qui vend des cœurs de microprocesseur implantables sur microcontrôleur. A partir de cette base de cœur, chaque constructeur façonne son circuit avec ces propres périphériques.

Parmi tous les constructeurs de microcontrôleurs 8-bits se dégagent essentiellement Atmel, Microchip Technology et Motorola qui est maintenant devenu Freescale.

Ce dernier est un des premiers constructeurs et a fondé sa réputation notamment grâce au MC 68HC11 qui est l'un des premiers microcontrôleurs 8-bits du marché.

Atmel est plus connu pour ses séries « **AVR 8-bits RISC** » parmi lesquels figurent trois niveaux de gamme : les « **Attiny** » représentant la gamme bon marché, les « **At90** » pour le milieu de gamme et les « **Atmega** » regroupant le haut de gamme.

Dans le même style, trois catégories de performance sont également retrouvées chez **Microchip**. Il s'agit de trois groupes : « Baseline Architecture » (Architecture de base), « Mid Range Architecture » (Architecture de gamme moyenne) et « High Range Architecture » (Architecture haute gamme).

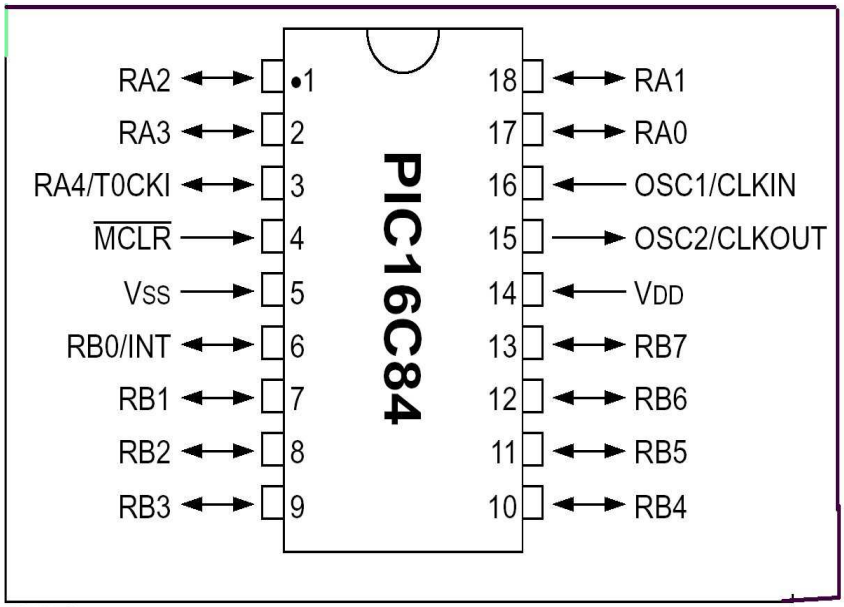


Le microcontrôleur PIC 16F84
(description simplifiée)

I - PRESENTATION :

Le micro-contrôleur PIC 16F84 est un micro-contrôleur de la société MICROCHIP. Les PICs sont des composants dits RISC à architecture Harvard (Reduce Instructions Construction Set), ou encore composant à jeu d'instructions réduit. Pourquoi ? Et bien, sachez que plus on réduit le nombre d'instructions, plus facile et plus rapide en est le décodage, et plus vite le composant fonctionne.
Vous aurez deviné qu'on trouve sur le marché 2 familles opposées, les RISC et les CISC (Complex Instructions Construction Set). Chez les CISC, on diminue la vitesse de traitement, mais les instructions sont plus complexes, plus puissantes, et donc plus nombreuses. Il s'agit donc d'un choix de stratégie.
Son architecture simplifiée fait qu'il est très utilisé en domotique ou pour les petits automatismes industriels.

II – Brochage du pic 16F84



Vss et **Vdd** : broches d'alimentation (3 à 5,5 V)

OSC1, **OSC2** : signaux d'horloge. Peuvent recevoir un circuit RC ou un résonnateur. **CLKIN** peut être connectée à une horloge externe. (0 à 4 , 10 ou 20 MHz)

$\overline{\text{MCLR}}$: Reset (Master Clear).

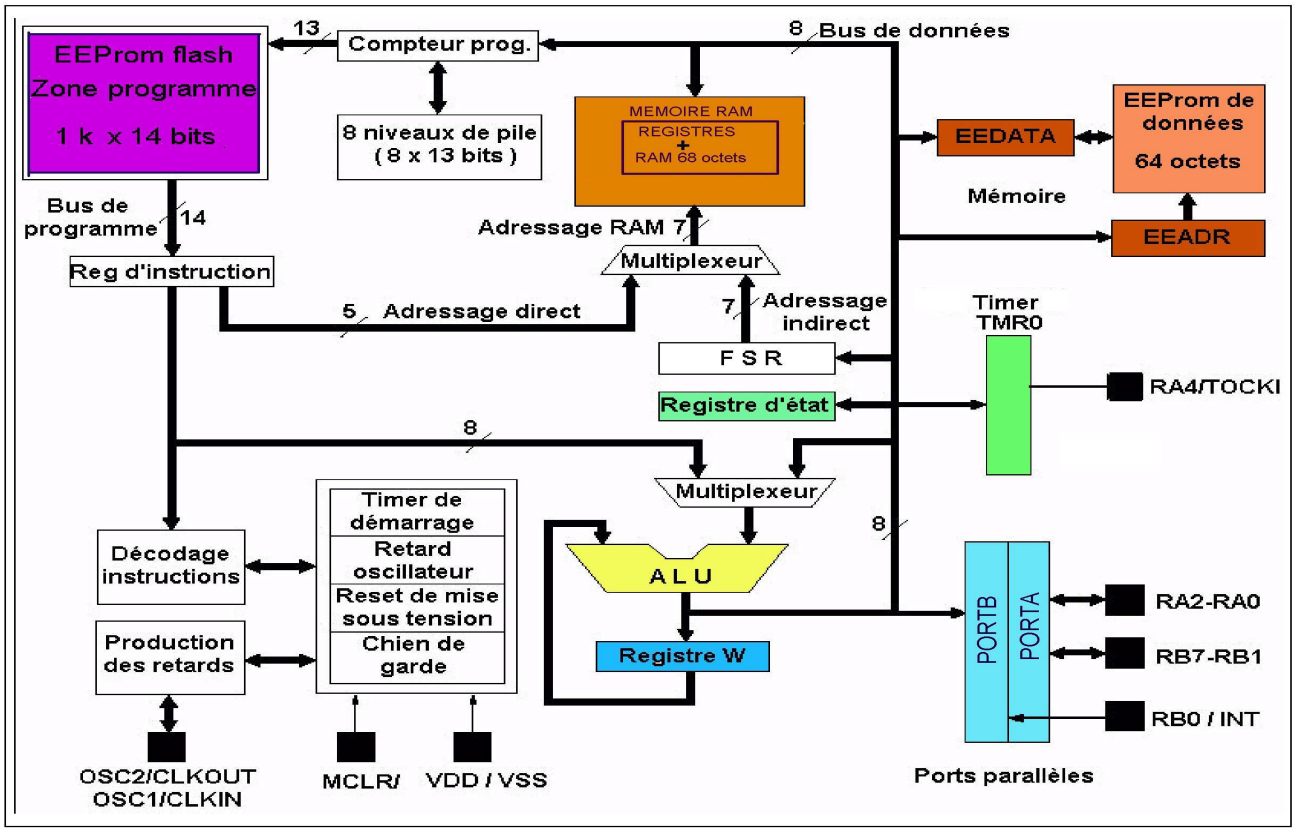
RA0..RA4 : 5 E/S du port A
RB0..RB7 : 8 E/S du port B

T0CKI : Entrée d'horloge externe du timer TMR0.

INT : Entrée d'interruption externe

II - ARCHITECTURE INTERNE :

Schéma fonctionnel :



3.1) Organisation de la mémoire :

a.- Mémoire programme (Type EEPROM) :

La taille de la mémoire programme est de 1 K x 14 bits, ce qui correspond à 1024 instructions. Néanmoins, les adresses $ØØØØ et $ØØØ4 sont interdites car réservées aux vecteurs de Reset et d'interruption.

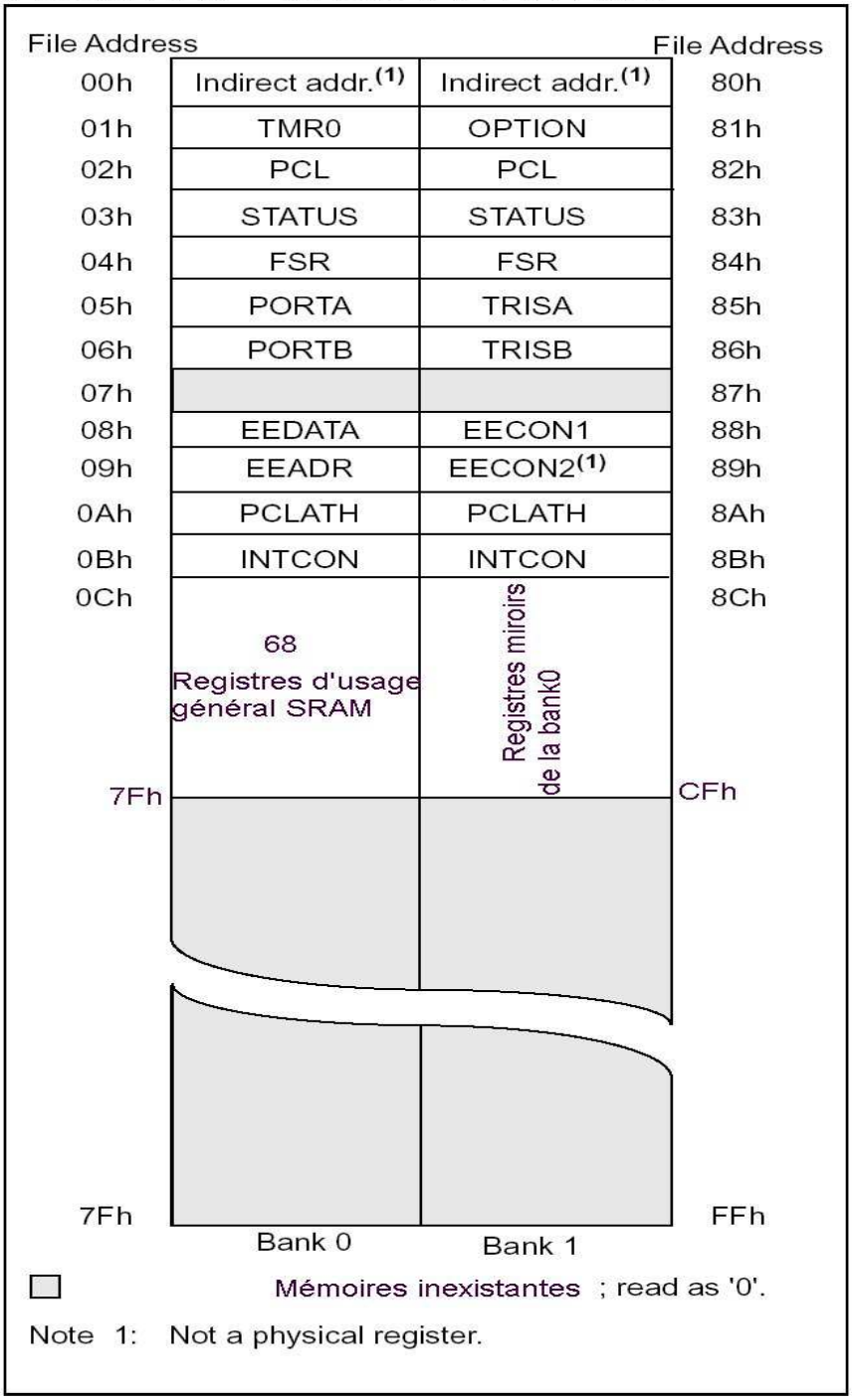
b.- Mémoire EEPROM de données :

Il est parfois utile de stocker des données non volatiles dans une application microprogrammée (étalonnage de capteurs, par exemple), pour cela le PIC 16F84 dispose d'une mémoire EEPROM de 64 octets.

c.- *Mémoire de données RAM :*

La mémoire RAM (128 octets) permet au programme de stocker et de lire des données. Ces données sont volatiles.

Elle contient aussi tous les registres du circuit :



Le programme ne peut stocker que 68 mots différents. En effet, la zone située entre $8C et $CF contient les mêmes valeurs que celle située de $∅C à $7F.

3.2 - Les registres du PIC 16F84 :

*a.- Régistre **W** :*

Fonction : accumulateur ou registre de travail, il permet le transit des données avant et après leur traitement.

*b.- **STATUS**(registre d'état) :*

Fonction : indique l'état du micro-contrôleur à la suite d'une opération et permet de gérer l'accès à la bank ∅ et la bank 1.

7	6	5	4	3	2	1	∅
0	RP1	RP0	T0	PD	Z	DC	C

C : retenue arithmétique (opérations sur 8 bits)
DC : retenue BCD (opérations sur 4 bits)
Z : passe à 1 si le résultat d'une opération est nulle
PD : indique par un ∅, la mise en veille du micro-contrôleur
T0 : utilisé pour la gestion du chien de garde.

RP0 et
RP1 : définissent l'accès aux banques de registre.

RP0 = 0 ; RP1 = 0 ∏ Accès bank 0
RP0 = 1 ; RP1 = 0 ∏ Accès bank 1
Par défaut, RP0 = RP1 = 0.

Le bit 7 est inutilisé et positionné à 0.

*c.- **PORT A - PORT B** (entrées/sorties) :*

Fonction : mémorise les données présentes sur les ports en entrée ou en sortie.

PORT A	X	X	X	X	PA3	PA2	PA1	PA0
PORT B	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0

d.- ***TRIS A - TRIS B*** *(registres de direction) :*

Fonction : configure les ports A et B.

- en entrée : le bit correspondant est fixé à 1 (I = in)
- en sortie : le bit correspondant est fixé à 0 (0 = out).

Exemple : soit à positionner le port A en entrée et le port B en sortie.

TRIS A	X	X	X	X	1	1	1	1
TRIS B	0	0	0	0	0	0	0	0

e.- ***PCLATCH - PCL*** *:*

Fonction : l'association de ces deux registres constitue le compteur programme (13 bits).

f.- ***FSR*** *(registre d'index) :*

Fonction : pointeur d'adresse dans le cas d'un adressage indexé(ou indirect).

g.- ***INTCON*** *:*

Fonction : registre de contrôle d'interruption (fera l'objet d'une étude détaillée).

h.- ***TMRO - OPTION*** *:*

Fonction : gèrent le fonctionnement du Timer (feront l'objet d'une étude détaillée).

i.- ***EExx***

Fonction : gèrent l'accès à la mémoire EEPROM (feront l'objet d'une étude détaillée).

4) Les instructions et les modes d'adressages

L'étude n'est pas nécessaire si le programme du PIC est écrit exclusivement en langage C.

On peut utiliser des instructions en assembleur dans un programme écrit en langage C
Exemple :

```
# include <pic.h>
unsigned char var ;
void main (void)
{
.var=1 ;
#asm          // les instructions en assembleur débutent par #asm et finissent par #endasm //
rlf _var,f
#endasm
}
```

5) Exemple de programme écrit en langage C 16F84 (compilateur Hi-Tech)

```
#include <PIC1684.h>
void main(void)
{
TRISB=0x00;
PORTB=0xF0;    // PORTB = 11110000 //
}
```

6) Les interruptions du PIC 16f84

6.1 - PRESENTATION :

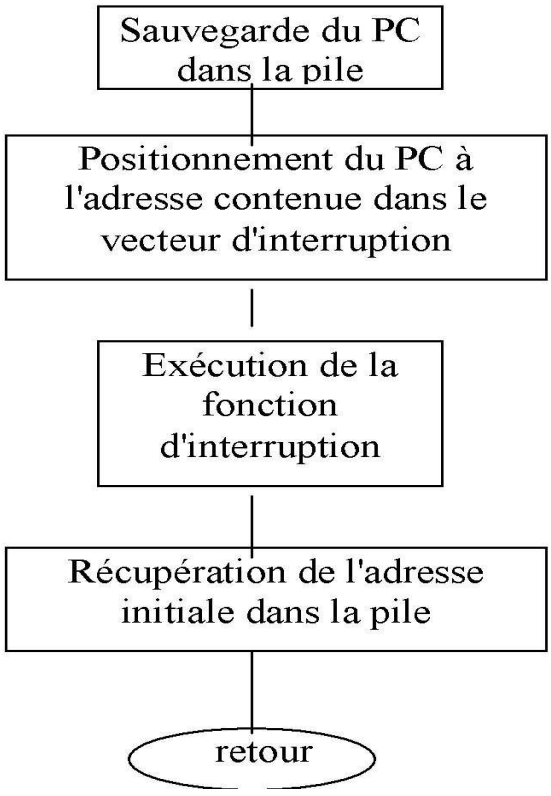
Une interruption est un événement qui impose au micro-contrôleur une suspension du programme principal pour effectuer une fonction annexe. Lorsque l'exécution de la fonction interruption est terminée, le micro-contrôleur reprend le programme principal là où il s'était interrompu.

6.2 - TRAITEMENT DE L'INTERRUPTION :

Le PIC 16F84 ne contient qu'un vecteur d'interruption mais peut gérer plusieurs sources d'interruption.

a - Le vecteur d'interruption ($øøø4) :

Le vecteur d'interruption contient l'adresse à laquelle se situe la fonction interruption.
Lorsqu'il reçoit un signal d'interruption, le micro-contrôleur réagit de la manière suivante
Toutes les sources ayant le même vecteur d'interruption, si le programmeur utilise plusieurs sources d'interruptions, il lui faudra déterminer d'ou vient la source d'interruption.



b - Les sources d'interruption :

Interruption EE : indique le remplissage de l'EEPROM.
Interruption T0 : indique la fin de comptage du Timer une fois que le contenu de tmr0 passe de 0xff à 0x00, une interruption peut être générée).
Interruption INT : indique un front sur l'entrée RBO. Le front est programmé par le bit du registre OPTION.
Interruption RBI : indique un changement d'état des bits RB7 à RB4.

c - Registre INTCON :

Pour pouvoir générer une interruption, la source concernée doit être sélectionnée. C'est le rôle des bits de sélection.

Lorsque le processeur reçoit une interruption, il doit en localiser la provenance. C'est le rôle des indicateurs (flag).

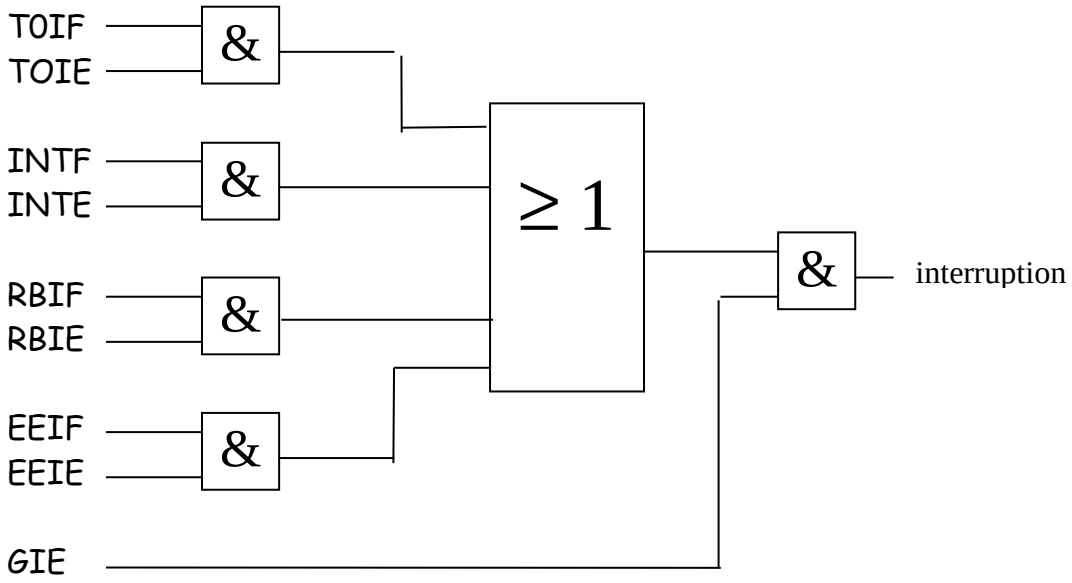
INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

GIE : sélection du fonctionnement en interruption
EEIE : sélection de l'interruption EE
T0IE : sélection de l'interruption T0
INTE : sélection de l'interruption INT
RBIE : sélection de l'interruption RB
T0IF : indication de l'interruption T0
INTF : indicateur de l'interruption INT
RBIF : indicateur de l'interruption RBI

Le flag de l'interruption EE est situé dans le registre de contrôle EECON1.

Important : A la fin du traitement d'une interruption, le flag de la source correspondante doit être remis à zéro.

Remarque : La logique des interruptions peut se schématiser ainsi :



Gestion d'une interruption en langage C

Une fonction void interrupt doit être déclarée et ne peut avoir de paramètres

```
long tick_count;
void interrupt tc_int(void)
{
++tick_count;
}
```

Seulement une fonction d'interruption peut être déclarée. Le vecteur d'interruption en $0004 pointera automatiquement sur cette fonction.

7) Le timer du PIC 16f84

7.1 - PRESENTATION :

Le timer permet d'alléger les tâches effectuées par le microcontrôleur en réalisant les opérations de comptages. Il permet aussi d'obtenir des bases de temps précises sans se soucier des cycles des instructions.

7.2- UTILISATION :

Le timer du PIC 16F84 utilise deux registres :

TMR0 : registre de comptage et de prépositionnement (8 bits). Dans tous les cas, le timer compte à partir de la valeur chargée dans ce registre jusqu'à $FF.

OPTION : registre de configuration.

OPTION	RBPU	INTEG	T0CS	T0SE	PSA	PS2	PS1	PS0

RBPU : inutilisé pour le timer.

INTED : sélection du front de l'interruption (inutilisé pour le timer).

T0CS : sélection de la source de l'horloge (1 : externe RA4 ; 0 : interne).

T0SE : sélection du front actif de l'horloge externe (1 : montant ; 0 : descendant).

PSA : sélection de la division d'horloge (1 : au chien de garde ; 0 : au timer).

PS2 à PS0 : sélection de la valeur de la division d''horloge.

1 Fonctionnement avec horloge externe :

Le bit T0CS doit être positionné à 1, le signal d'horloge est alors envoyé sur la broche RA4. Le front actif de l'horloge est défini par le bit T0SE.

2 Fonctionnement avec horloge interne :

Le bit T0CS est positionné à 0. La fréquence de l'horloge est égale à la fréquence d'un cycle machine (ex : quartz à 4Mhz F=1Mhz) si le bit PSA est positionné à 1. Si le bit PSA est positionné à 0, la fréquence est réglée en fonction des bits PSx.

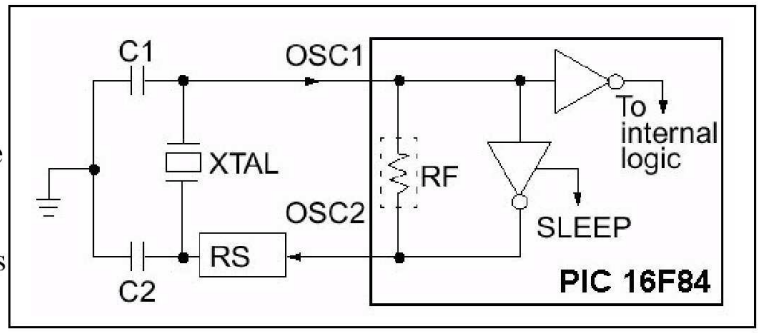
PS2	PS1	PS0	Ratio
0	0	0	1/2
0	0	1	1/4
0	1	0	1/8
0	1	1	1/16
1	0	0	1/32
1	0	1	1/64
1	1	0	1/128
1	1	1	1/256

3 Fonctionnement en mode interruption :

Le timer génère une interruption lorsque sa valeur atteint $FF. Elle peut être traitée en programmant le registre INTCON.

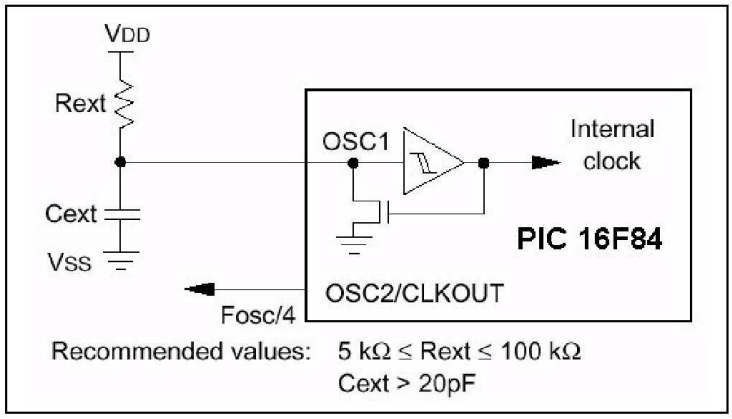
1) L'horloge du pic 16F84

Horloge interne



0 à 4 , 10 ou 20 MHz selon le type de µC.

Utilisation d'un résonateur quartz ou céramique
(oscillateur Colpitts).
RF assure la polarisation de l'inverseur CMOS.
Le filtre passe bas RS,C2 limite les harmoniques dus à l'écrêtage (le résonateur vibre en sinus) et
Réduit l'amplitude de l'oscillation.

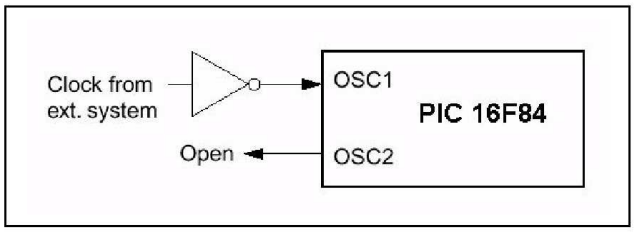


Utilisation d'un circuit RC
(Relaxateur)

Lorsque la tension aux bornes de Cext atteint le seuil haut de la bascule de Schmitt, le transistor MOS devient fortement conducteur et décharge Cext.
Quand la tension au point OSC1 atteint le seuil bas, le transistor MOS se bloque et Cext se (re)charge.

Horloge externe

Dans certains cas, une horloge externe au microcontrôleur peut être utilisée pour synchroniser le PIC sur un processus particulier.
La broche OSC2 est alors inutilisée.



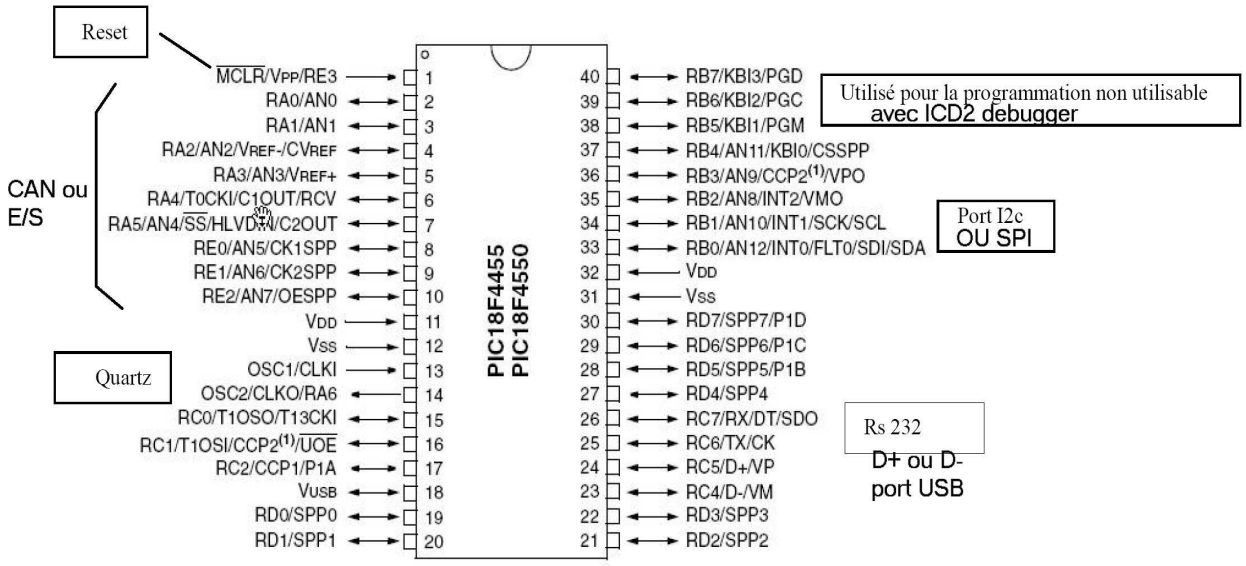
Le microcontrôleur PIC 18F4550

(Description simplifiée)

1) Présentation

Un PIC® est un microcontrôleur fabriqué par Microchip®, et les modèles identifiés comme 18F font partie de la gamme high-end de ce constructeur.
Les PIC® high-end 18F sont, comme les PIC® mid-range 16F, des microcontrôleurs 8 bits. Cependant, le bus d'instruction passe de 14 à 16 bits.
D'une architecture HARVARD, c'est un composant à jeu d'instruction réduit. Le PIC18 possède une multiplication 8x8 matérielle, extrêmement rapide (100nS à 1uS) ce qui lui confère quelques possibilités particulièrement utiles pour le traitement numérique du signal.

2) Brochage du PIC18F4550

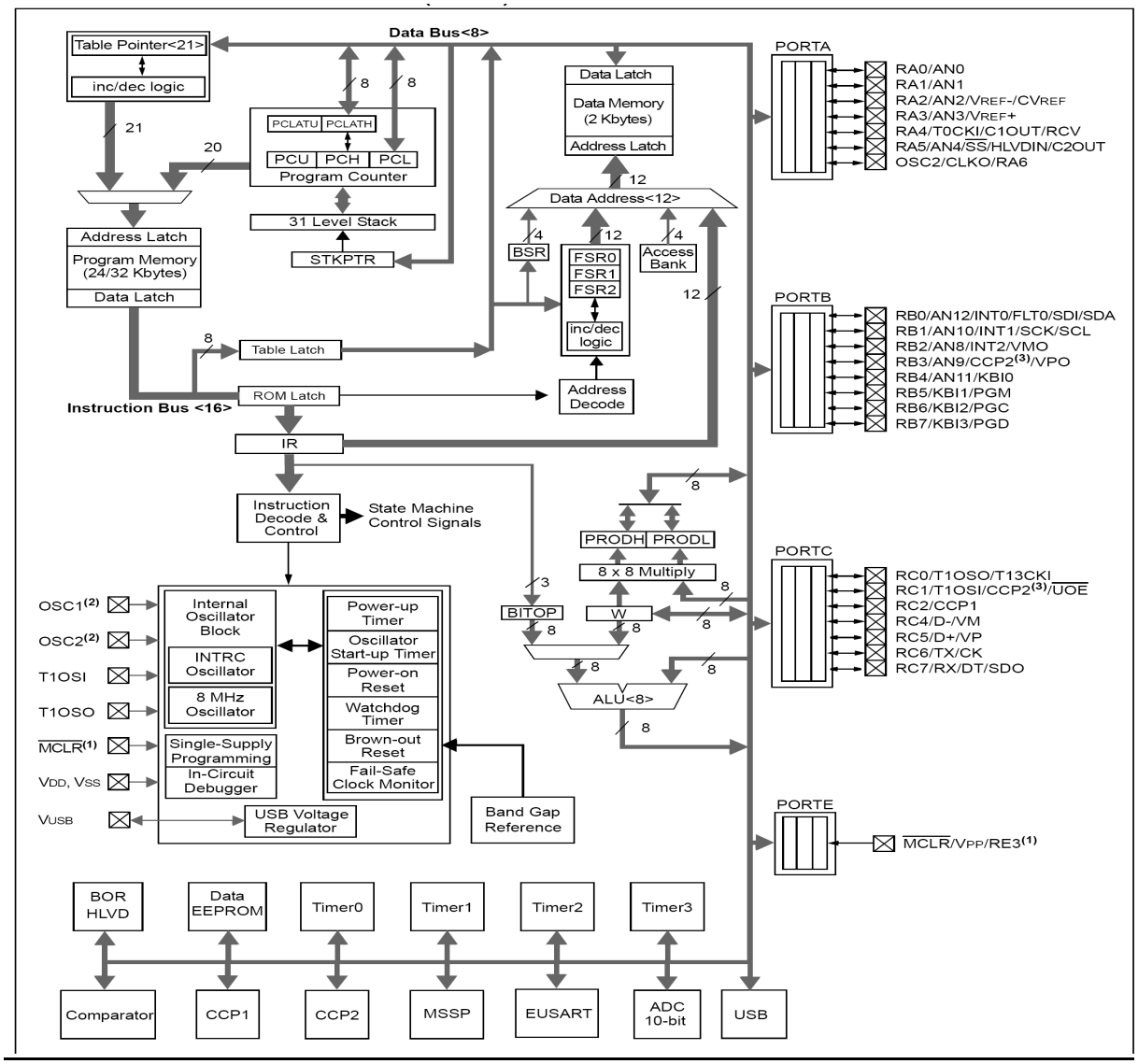


3) Architecture interne

Ce microcontrôleur intègre les fonctions spécifiques notamment :

- Horloge interne rapide (max 48MHz).
- Suffisamment de port pour piloter un afficheur et une interface homme machine (BP).
- Gestion matérielle du bus I2C et SPI.
- Gestion matérielle de l'USB et liaison série RS232 (EUSART).
- Bonne capacité de mémoire programme (32Kinstructions)
- Intègre 10 entrées pour la conversion analogique numérique.
- Programmation et debug sur cible (In-Circuit Serial Programming, In-Circuit debug, perte de RB6, RB7)
- Horloge max de 48 MHz par multiplication interne avec PLL.

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3



4) Les ports d'entrées sorties

a) Les ports de données PORTA, PORTB, PORTC, PORTD, PORTE

Fonction : mémorise les données présentes sur les ports en entrée ou en sortie

b) Les registres de direction TRISA, TRISB, TRISC, TRISD, TRISE

Fonction configure les ports :

- En entrée : le bit correspondant est fixé à 1 (I=in)
- En sortie : le bit correspondant est fixé à 0 (0=out)

Exemple : soit à programmer le portA en entrée et le portB en sortie

5) Exemple de programme en langage C pour un PIC18F4550

```
#include <PIC18F4550.h>
void main(void)
{
TRISB=0x00;
PORTB=0xF0
}
```

6) Les interruptions en mode simple

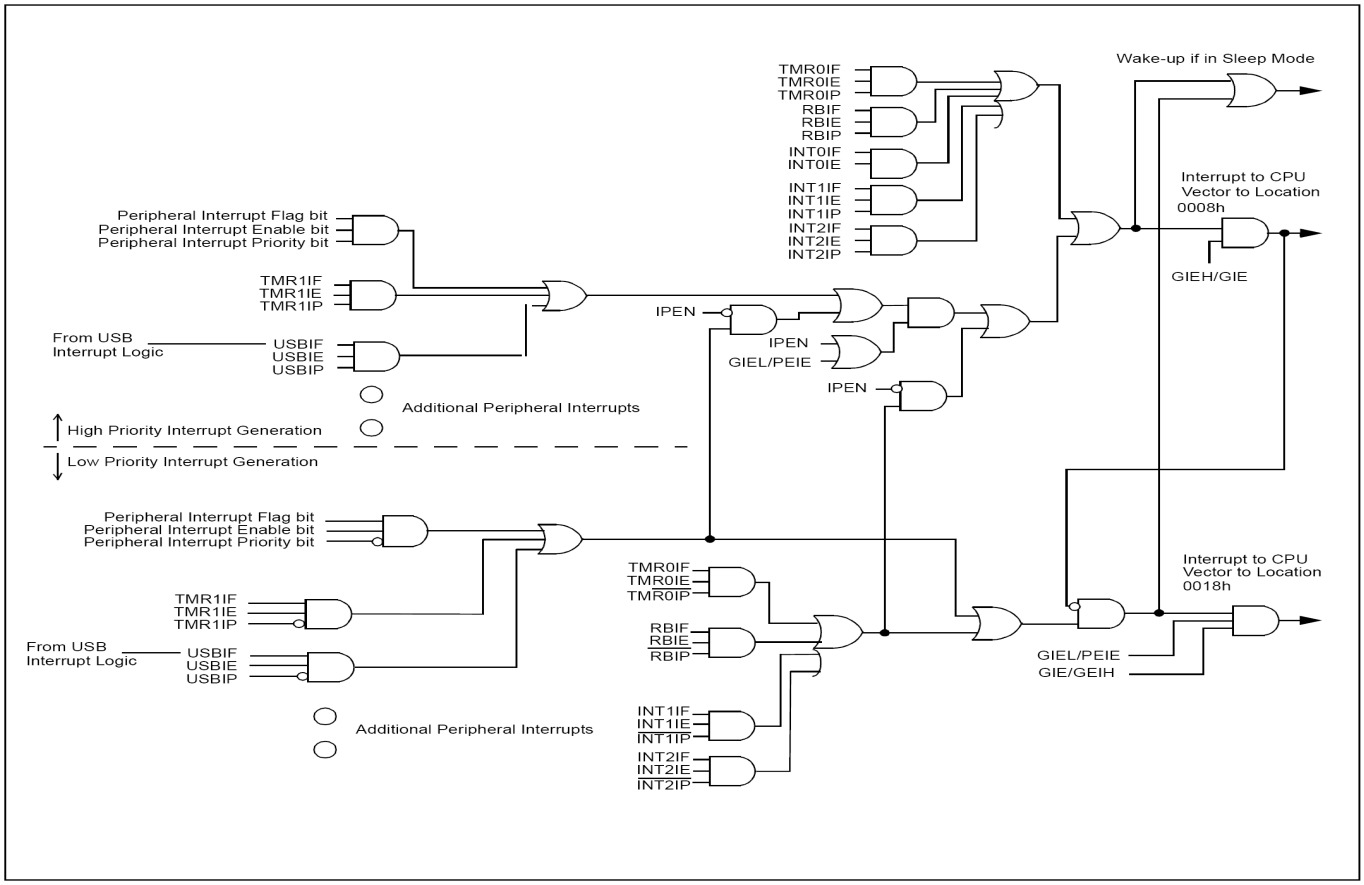
a) Présentation

Comme sur les PIC16F, la gamme 18F dispose de la possibilité d'interrompre votre programme sur base d'événements asynchrones à son déroulement, les interruptions.
Comme sur les 16F, le déclenchement d'une interruption conduit le PIC® à sauter à une adresse fixe de la mémoire programme. Cette adresse est figée par hardware. Pour les spécialistes d'autres processeurs, on peut dire que le vecteur d'interruption est unique.
En fait, nous verrons qu'il y en a deux.
Souvenez-vous que pour qu'un événement déclenche une interruption, il faut que les éléments suivants soient réunis :
1) Les interruptions générales doivent être validées
2) Le bit de validation de cet événement doit être validé
3) Si nécessaire pour ce type d'interruption ou pour le mode choisi, les interruptions périphériques doivent être validées
4) Le flag d'interruption doit avoir été forcé par l'événement lui-même.

N'oubliez pas non plus que pour sortir d'une interruption, il faut d'abord effacer le flag qui a déclenché l'interruption, sauf dans le cas où ce flag est en lecture seule car dans ce cas vous avez une action à effectuer qui l'effacera automatiquement (lecture d'un registre par exemple). Enfin, souvenez-vous qu'un flag sera positionné lorsque survient l'événement prévu, même si le bit de validation n'est pas positionné. Dans ce cas, évidemment, le flag ne déclenchera pas l'interruption correspondante.

b) La logique

La logique des interruptions se schématise ainsi :



Chaque source d'interruption possède

-un bit d'autorisation E (enable) ce bit doit être à 1 pour valider l'interruption
-un bit d'état F (flag) qui indique s'il y a eu ou non un événement

On peut remarquer sur le schéma ci-dessous du principe de la circuiterie de prise en compte des interruptions, un bit particulier, appelé GIE/GIEH qui ouvre ou ferme des deux dernières portes ET, autorisant ou interdisant ainsi la prise en compte de toutes les autres interruptions.
La hiérarchisation des interruptions se fait avec la mise à un du bit IPEN du registre RCON. Dès cet instant, les niveaux de priorité sont activés. La validation de cette hiérarchisation rend actif un deuxième bit de validation générale qui a pour non GIEL. On dispose alors de GIE/GIEH(H comme High) pour autoriser ou non les interruptions de priorité haute (adresse PC

0008h) et de GIEL (L comme Low) pour autoriser les interruptions de priorité basse (adresse PC 0018h).

c) Les registres concernés

Registre RCON :

L : lecture
E : Ecriture

	L/E	L/E	L	L/E	L	L	L/E	L/E
RCON	**IPEN**	**SBOREN**	**-**	**/RI**	**/TO**	**/PD**	**/POR**	**/BOR**
Etat à le mise sous tension	0	1	0	1	1	1	0	0

• Bit 0 ou Bit */BOR* pour Brown Out Reset :

Ce bit indique, lorsqu'il est à 0, qu'un reset dû à une détection de chute de la tension d'alimentation vient de se produire. Ce bit est un peu particulier en ce sens que son état est inconnu suite à une mise sous tension du circuit. Il doit donc être positionné à 1 par programme. De même, après sont passage à 0 suite à la détection d'une baisse de tension d'alimentation, il faut le remettre à 1 par programme.

• Bit 1 ou Bit */POR* pour Power On Reset :

Ce bit indique, lorsqu'il est à 0, qu'un reset suite à une mise sous tension du circuit vient de se produire. Comme le bit 0, il faut le remettre à 1 par programme après son passage à 0 suite à la détection d'un reset de mise sous tension.

• Bit 2 ou Bit */PD* pour Power Down :

Ce bit est mis à 0 lors de l'exécution d'une instruction SLEEP. Il est mis à 1 lors d'une mise sous tension ou lors de l'exécution d'une instruction CLRWDT.

• Bit 3 ou Bit */TO* pour Timer Out :

Ce bit est mis à 0 lorsque le timer chien de garde déborde. Il est mis à 1 lors d'une mise sous tension ou lors de l'exécution d'une instruction CLRWDT ou SLEEP.

• Bit 4 ou Bit */RI*pour Reset Instruction :

Ce bit est mis à 0 lors de l'exécution d'une instruction RESET ayant eu comme conséquence de provoquer une reset du circuit. Comme le bit 0, il faut le remettre à 1 par programme après son passage à 0.

• Bit 6 ou Bit *SBOREN* pour Software BOR ENable :

Ce bit permet de contrôler par logiciel la validation (SBOREN=1) ou non (SBOREN=0) du reset en cas de baisse de l'alimentation ou reset BOR. Il faut au préalable positionner les bits BOREN0 et BOREN1 du registre configuration (CONFIG2L) du circuit respectivement à 1 ou 0.

• Bit 7 ou Bit *IPEN* pour Interrupt Priority ENable :

Lorsqu'il est mis à 1, la gestion de priorité des interruptions est assurée alors qu'elle ne l'est pas dans le cas contraire.

Registre INTCON :

L : lecture E : Ecriture	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
INTCON	**GIE/GIEH**	**PEIE/GIEL**	**TMR0IE**	**INTOIE**	**RBIE**	**TMR0IF**	**INT0IF**	**RBIF**
Etat à le mise sous tension	0	0	0	0	0	0	0	X

• Bit 0 ou Bit *RBIF* pour RB Interrupt Flag :

S'il est mis à 1, ce bit indique un changement d'état de l'une des lignes RB4 à RB7, alors qu'il reste à 0 dans le cas contraire. Ce bit doit être remis à 0 par logiciel.

• Bit 1 ou Bit *INT0IF* pour INT0 Interrupt Flag :

S'il est mis à 1, ce bit indique une interruption provoquée par la ligne d'interruption externe INT0 alors qu'il reste à 0 dans le cas contraire. Ce bit doit être remis à 0 par logiciel.

• Bit 2 ou Bit *TMR0IF* pour TiMeR 0 Interrupt Flag :

S'il est mis à 1, ce bit indique un débordement du timer 0 alors qu'il reste à 0 dans le cas contraire. Ce bit doit être remis à 0 par logiciel.

• Bit 3 ou Bit *RBIE* pour RB Interrupt Enable :

S'il est mis à 1, ce bit autorise les interruptions provoquées par un changement d'état des lignes du port parallèle B

• Bit 4 ou Bit *INT0IE* pour INT0 Interrupt Enable :

S'il est mis à 1, ce bit autorise les interruptions provoquées par la ligne d'interruption externe INT0..

• Bit 5 ou Bit *TMR0IE* pour TiMeR 0 Interrupt Enable :

S'il est mis à 1, ce bit autorise les interruptions dues au débordement du timer 0.

• Bit 6 ou Bit *PEIE/GIEL* pour PEripheral Interrupt Enable ou Global Interrupt Enable Low :

Lorsque la hiérarchisation des interruptions n'est pas autorisée (Bit IPEN=0), ce bit se nomme PEIE et fonctionne de la façon suivante :

- S'il est mis à 1, il autorise toutes les interruptions d'origine périphérique non masquées par leurs bits individuels de validation à être prises en compte.
- S'il est mis à 0, toutes les interruptions périphériques sont masquées, même si leurs bits de validation individuels sont correctement positionnés.

Lorsque la hiérarchisation des interruptions est autorisée (Bit IPEN=1), ce bit se nomme GIEL et fonctionne de la façon suivante :

- S'il est mis à 1, il autorise toutes les interruptions de priorité basse non masquées par leurs bits individuels de validation à être prises en compte.
- S'il est mis à 0, toutes les interruptions de priorité basse sont masquées, même si leurs bits de validation individuels sont correctement positionnés.

• Bit 7 ou Bit *GIE/GIEH* pour Global Interrupt Enable ou Global Interrupt Enable High :

Lorsque la hiérarchisation des interruptions n'est pas autorisée (Bit IPEN=0), ce bit se nomme GIE et fonctionne de la façon suivante :

- S'il est mis à 1, il autorise toutes les interruptions non masquées par leurs bits individuels de validation à être prises en compte.
- S'il est mis à 0, toutes les interruptions sont masquées, même si leurs bits de validation individuels sont correctement positionnés.

Lorsque la hiérarchisation des interruptions est autorisée (Bit IPEN=1), ce bit se nomme GIEH et fonctionne de la façon suivante :

- S'il est mis à 1, il autorise toutes les interruptions de priorité haute non masquées par leurs bits individuels de validation à être prises en compte.
- S'il est mis à 0, toutes les interruptions de priorité haute sont masquées, même si leurs bits de validation individuels sont correctement positionnés.

Registre INTCON2 :

L : lecture
E : Ecriture

	L/E	L/E	L/E	L/E	L	L/E	L	L/E
INTCON2	**/RBPU**	**INTEDG0**	**INTEDG1**	**INTEDG2**	**-**	**TMR0IP**	**-**	**RBIP**
Etat à le mise sous tension	1	1	1	1	0	1	0	1

• Bit 0 ou Bit *RBIP* pour RB port on change Interrupt Priority :

S'il est mis à 1, ce bit place en mode haute priorité les interruptions de type changement d'état de l'une des lignes RB4 à RB7. Dans le cas contraire, elles sont en mode basse priorité.

• Bit 2 ou Bit *TMR0IP* pour TiMeR 0 Interrupt Priority :

S'il est mis à 1, ce bit place en mode haute priorité l'interruption de débordement du timer 0. Dans le cas contraire, elles sont en mode basse priorité.

• Bit 4 ou Bit *INTEDG2* pour external INTerrupt 2EDGe select :

S'il est mis à 1, ce bit provoque une interruption externe via l'entrée INT2 sur un front montant alors que c'est sur un front descendant dans le cas contraire..

• Bit 5 ou Bit *INTEDG1* pour external INTerrupt 1EDGe select :

S'il est mis à 1, ce bit provoque une interruption externe via l'entrée INT1 sur un front montant alors que c'est sur un front descendant dans le cas contraire..

• Bit 6 ou Bit *INTEDG0* pour external INTerrupt 0EDGe select :

S'il est mis à 1, ce bit provoque une interruption externe via l'entrée INT0 sur un front montant alors que c'est sur un front descendant dans le cas contraire.

• Bit 7 ou Bit */RBPU* pourRB Pull Up enable :

Lorsque ce bit est mis à 1, toutes les résistances de tirage au niveau haut du port B sont désactivées alors qu'elles sont validées une à une sous le contrôle du registre « latch » du port B dans le cas contraire.

Registre INTCON3 :

L : lecture
E : Ecriture

	L/E	L/E	L	L/E	L/E	L	L/E	L/E
INTCON3	**INT2IP**	**INT1IP**	**-**	**INT2IE**	**INT1IE**	**-**	**INT2IF**	**INT1IF**
Etat à le mise sous tension	1	1	0	0	0	0	0	0

• Bit 0 ou Bit *INT1IF* pour INT1 Interrupt Flag :

Ce bit passe à 1 lorsqu'une interruption valide est détectée sur l'entrée d'interruption externe INT1. Il doit être remis à 0 par logiciel.

• Bit 1 ou Bit *INT2IF* pour INT2 Interrupt Flag :

Ce bit passe à 1 lorsqu'une interruption valide est détectée sur l'entrée d'interruption externe INT2. Il doit être remis à 0 par logiciel.

• Bit 3 ou Bit *INT1IE* pour INT1 Interrupt Enable :

Lorsqu'il est mis à 1, ce bit autorise les interruptions depuis l'entrée externe INT1 alors qu'il les interdit dans le cas contraire.

• Bit 4 ou Bit *INT2IE* pour INT2 Interrupt Enable :

Lorsqu'il est mis à 1, ce bit autorise les interruptions depuis l'entrée externe INT2 alors qu'il les interdit dans le cas contraire.

• Bit 6 ou Bit *INT1IP* pour INT1 Interrupt Priority :

S'il est mis à 1, ce bit place en mode haute priorité l'interruption externe INT1. Dans le cas contraire, elle est en mode basse priorité.

• Bit 7 ou Bit *INT2IP* pour INT2 Interrupt Priority :

S'il est mis à 1, ce bit place en mode haute priorité l'interruption externe INT2. Dans le cas contraire, elle est en mode basse priorité.

PIR2 (0xFA1): Ecriture EEPROM / Bus collision / Faible VDD / TIMER3/ CCP2

OSCFIF	CMIF	USBIF	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF

OSCFIF: Oscillator Fail Interrupt Flag bit
1 = System oscillator failed, clock input has changed to INTOSC (must be cleared in software)
0 = System clock operating

CMIF: Comparator Interrupt Flag bit
1 = Comparator input has changed (must be cleared in software)
0 = Comparator input has not changed

USBIF: USB Interrupt Flag bit
1 = USB has requested an interrupt (must be cleared in software)
0 = No USB interrupt request

EEIF: Data EEPROM/FLASH Write Operation Interrupt Flag bit
BCLIF: Bus Collision Interrupt Flag bit
1 indique qu'une collision s'est produite
LVDIF: Low Voltage Detect Interrupt Flag bit
1 indique une détection de faible tension
TMR3IF: TMR3 Overflow Interrupt Flag bit
1 indique un débordement sur TMR3
CCP2IF: CCPx Interrupt Flag bit
Idem PIR1 ci dessus

PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

SPPIE: Streaming Parallel Port Read/Write Interrupt Enable bit(1)
1 = Enables the SPP read/write interrupt
0 = Disables the SPP read/write interrupt

Note 1: This bit is reserved on 28-pin devices; always maintain this bit clear.

ADIE: A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

RCIE: EUSART Receive Interrupt Enable bit
1 = Enables the EUSART receive interrupt
0 = Disables the EUSART receive interrupt

TXIE: EUSART Transmit Interrupt Enable bit
1 = Enables the EUSART transmit interrupt
0 = Disables the EUSART transmit interrupt

SSPIE: Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

CCP1IE: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

TMR2IE: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

TMR1IE: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

PIE2 (0xFA0): EEPROM / BUS collision / Faible VDD / TIMER 2 / CPP2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIE	CMIE	USBIE	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE
bit 7							bit 0

OSCFIE: Oscillator Fail Interrupt Enable bit

1 = Enabled
0 = Disabled

CMIE: Comparator Interrupt Enable bit

1 = Enabled
0 = Disabled

USBIE: USB Interrupt Enable bit

1 = Enabled
0 = Disabled

EEIE: Data EEPROM/FLASH Write Operation Interrupt Enable bit
Autorise l'interruption de fin d'écriture
BCLIE: Bus Collision Interrupt Enable bit
Autorise l'interruption lors d'un collision
LVDIE: Low Voltage Detect Interrupt Enable bit
Autorise l'interruption lors de la détection d'un tension faible
TMR3IE: TMR3 Overflow Interrupt Enable bit
Autorise l'interruption lors du débordement de TMR3
CCP2IE: CCP2 Interrupt Enable bit
Autorise l'interruption CCP2 (pour capture ou compare)

IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
SPPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

PSPIP: Parallel Slave Port Read/Write Interrupt Priority bit
ADIP: A/D Converter Interrupt Priority bit
RCIP: USART Receive Interrupt Priority bit
TXIP: USART Transmit Interrupt Priority bit
SSPIP: Master Synchronous Serial Port Interrupt Priority bit
CCP1IP: CCP1 Interrupt Priority bit
TMR2IP: TMR2 to PR2 Match Interrupt Priority bit
TMR1IP: TMR1 Overflow Interrupt Priority bit

1 haute priorité
0 basse priorité

IPR2 (0xFA2) : Priorités

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	CMIP	USBIP	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP
bit 7							bit 0

OSCFIP: Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

CMIP: Comparator Interrupt Priority bit
1 = High priority
0 = Low priority

USBIP: USB Interrupt Priority bit
1 = High priority
0 = Low priority

EEIP: Data EEPROM/FLASH Write Operation Interrupt Priority
BCLIP: Bus Collision Interrupt Priority bit
LVDIP: Low Voltage Detect Interrupt Priority bit
TMR3IP: TMR3 Overflow Interrupt Priority bit
CCP2IP: CCP2 Interrupt Priority bit

1 haute priorité
0 basse priorité

RCON: RESET CONTROL REGISTER

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

IPEN: Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

SBOREN: BOR Software Enable bit
For details of bit operation, see Register 4-1.

Unimplemented: Read as '0'

$\overline{RI}$: RESET Instruction Flag bit
For details of bit operation, see Register 4-1.

$\overline{TO}$: Watchdog Time-out Flag bit
For details of bit operation, see Register 4-1.

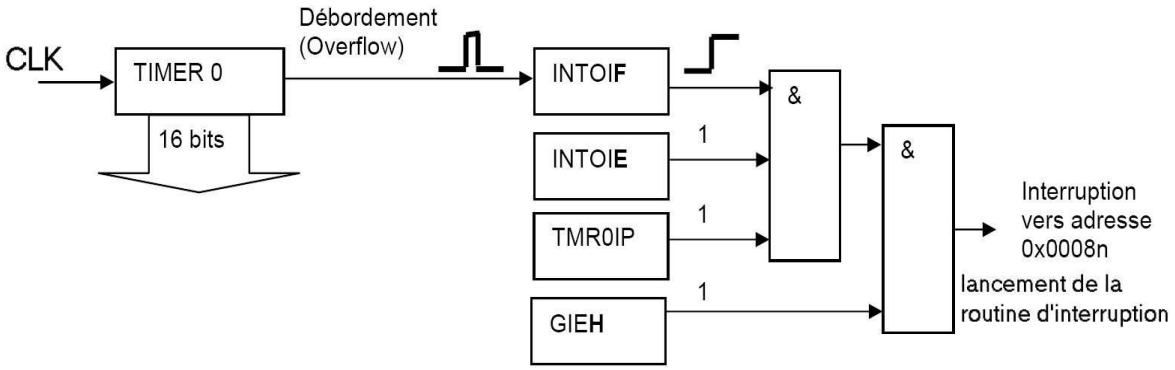
$\overline{PD}$: Power-Down Detection Flag bit
For details of bit operation, see Register 4-1.

$\overline{POR}$: Power-on Reset Status bit
For details of bit operation, see Register 4-1.

$\overline{BOR}$: Brown-out Reset Status bit
For details of bit operation, see Register 4-1.

Note: Actual Reset values are determined by device configuration and the nature of the device Reset. See Register 4-1 for additional information.

Exemple : Pour utiliser l'interruption générée lors du débordement du TIMER0 il faut
-mettre INT0IE à 1 -mettre GEIH ou GEIL à 1 TMR0IP à 1
L'interruption est déclanchée lors du débordement, il faut OBLIGATOIREMENT remettre INT0IF à 0 avant de ressortir de l'interruption, sinon elle reste active.



d) Gestion d'une interruption en langage C sous le compilateur MCC18

Sans hiérarchisation des interruptions

Le compilateur Microchip C18 ne crée pas automatique un saut vers le programme d'interruption. Une instruction de saut goto doit être crée afin que le vecteur redirige l'interruption vers le bon programme d'interruption. Une section de code a été créée à l'adresse 0x08 pour que le vecteur d'interruption haute priorité redirige l'interruption vers l'adresse du programme d'interruption

```
#pragma code high_vector=0x08   /*Place l'adresse du prog d'interruption à l'adresse 0x08*/

void interruption(void)
{
      _asm
            goto itprioritaire  // saut vers le programme d'interruption
      _endasm
}

#pragma code
#pragma interrupt itprioritaire // Directive qui définit notre programme d'interruption

void itprioritaire (void)
{
// Code de la fonction interruption
}
```

Cette routine d'interruption itprioritaire sera lancée pour chaque interruption. Le vecteur d'interruption en $0008 pointera sur cette fonction.
itprioritaire est le nom de la fonction, on peut lui donner n'importe quel nom.
Une interruption ne peut toujours pas être interrompue par une autre.

Avec hiérarchisation des interruptions

```
/*****************High priority interrupt vector ***************************/
#pragma code high_vector=0x08
void interrupt_at_high_vector(void)
{
  _asm GOTO high_isr _endasm
}


#pragma code
```

```
/*****************High priority ISR ************************/

#pragma interrupt high_isr // directive interrupt pour définir notre prog interruption high
void high_isr (void)
{
// Votre code ici
   }
}

void low_isr(void);
#pragma code low_vector=0x18
void interrupt_at_low_vector(void)
{
_asm  GOTO low_isr  _endasm
}
#pragma code /* return to the default code section */
#pragma interruptlow low_isr // directive interruptlow pour définir notre prog interruption low
void low_isr (void)
{
/* ISR code goes here */
}
```

7) Les interruptions hiérarchisées

En fait, il s'agit d'un grand mot, sur les 18F la hiérarchisation se limite à deux niveaux de priorité des interruptions. Mais ce mécanisme de priorité est déjà un grand pas en avant pour des applications aux timings précis.
Si vous mettez ce mécanisme en service, vous disposez alors de deux niveaux d'interruptions :

- Les interruptions basses priorité
- Les interruptions hautes priorité

L'intérêt est que les interruptions basse priorité peuvent être elles-mêmes interrompues par les interruptions de type haute priorité.
Une interruption haute priorité est une interruption qui peut interrompre.
Une interruption basse priorité (et évidemment également le programme en lui-même) peuvent être interrompue par une interruption haute priorité.
A l'inverse, une interruption haute priorité ne peut être interrompue par rien d'autre, ni par une interruption basse priorité, ni par une autre interruption haute priorité.
Ceci vous permet de toujours réagir aux événements les plus critiques, même si une interruption « classique » est déjà en cours d'exécution.
Pour mettre en service les interruptions hiérarchisées, ou encore les interruptions prioritaires, il vous suffit de placer à 1 le bit IPEN du registre RCON. Dès ce moment, les interruptions répondront à la logique des priorités.
Vous devez donc, et ce pour chaque interruption, précisé si vous désirez que celle-ci soit considérée comme interruption haute ou basse priorité. Ceci est fait en positionnant ou non le bit « IP » correspondant à l'interruption concernée. A la mise sous tension, tous les bits « IP » sont forcés à « 1 », ce qui correspond à un fonctionnement en mode haute-priorité.

Une fois les interruptions prioritaires en service, toute interruption est considérée par défaut comme interruption haute priorité.

Le rôle de certains bits change en mode haute priorité, ce qui explique qu'ils possèdent un double nom.
Ainsi, le bit GIE, qui coupait toutes les interruptions, devient le bit GIEH (General Interrupt Enable High priority), qui permet de mettre ou couper d'une seule fois toutes les interruptions haute priorité.

Remarque : Si vous coupez GIEH, vous coupez TOUTES les priorités, haute et basse priorité. Par contre, si vous positionnez GIEH, vous n'autorisez que les interruptions hautes priorités, car la mise en service des interruptions basses priorités nécessite un bit supplémentaire.
De même, le bit PEIE qui validait toutes les interruptions périphériques (liées aux registres PIEx) devient le bit GIEL (General Interrupt Enable Low priority), qui permet de couper d'une fois toutes les interruptions basse priorité.

8) Le convertisseur Analogique/Numérique :

Le PIC18F4550 est doté d'un convertisseur analogique numérique 10 bits. Le nombre d'entrées analogiques est de 13 (AN0 à AN12) :
5 registres sont utilisés pour configurer et gérer le CAN :

- Les registres de résultat : ***ADRESH et ADRESL (ou ADRES(16 bits)= ADRESH et ADRESL)***
- Les registres de contrôle : ***ADCON0, ADCON1 et ADCON2***

Le registre ***ADCON0*** est lié au ***pilotage du CAN***, ***ADCON1*** permet de ***configurer les lignes*** comme étant des entrées numériques, analogiques ou de référence. Enfin, ***ADCON2*** sert à choisir l'horloge de conversion, un temps d'acquisition programmé et la justification du résultat.

8.11. Registre ADCON0 :

L : lecture
E : Ecriture

	L	L	L/E	L/E	L/E	L/E	L/E	L/E
ADCON0	**-**	**-**	**CHS3**	**CHS2**	**CHS1**	**CHS0**	**GO/DONE**	**ADON**
Etat à le mise sous tension	0	0	0	0	0	0	0	0

- **Bit 0 ou Bit *ADON* :**

Ce bit doit être mis à 1 pour valider le convertisseur. Dans le cas contraite le convertisseur est arrêté et ne consomme aucune énergie.

- **Bit 1 ou Bit *GO/DONE* :**

Le fait de mettre ce bit à 1 démarre une conversion. Ensuite, **tant que ce bit reste** à 1 la conversion est en cours. Le passage de ce bit à 0, provoqué par la circuiterie du convertisseur, indique une fin de conversion et la possibilité de lire son résultat dans ADRESH et ADRESL ou ADRES.

- **Bits 2 à 5 ou Bits *CHS0* à *CHS3* pour Channel Select :**

Ces bits permettent de choisir l'entrée à convertir conformément aux indications du tableau ci-dessous.

CHS3	CHS2	CHS1	CHS0	Entrée sélectionnée
0	0	0	0	AN0
0	0	0	1	AN1
0	0	1	0	AN2
0	0	1	1	AN3
0	1	0	0	AN4
0	1	0	1	AN5
0	1	1	0	AN6
0	1	1	1	AN7
1	0	0	0	AN8
1	0	0	1	AN9
1	0	1	0	AN10
1	0	1	1	AN11
1	1	0	0	AN12

• **Bits 6 et 7 :**

Bits non utilisés et lus comme étant des 0.

8.2. Registre ADCON1 :

L : lecture
E : Ecriture

	L	L	L/E	L/E	L	L	L	L
ADCON1	-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
Etat à le mise sous tension	0	0	0	0	0	0	0	0

• **Bits 0 à 3 ou Bits *PCFG0* à *PCFG3* pour Port ConFiGuration :**

Ces bits permettent de choisir la répartition entre les ports analogiques et digitaux sur le ou les ports parallèles avec le convertisseur conformément aux indications di tableau ci-dessous.

PCGF (3…0)	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

D : Digital A : Analogique

• **Bits 4 et 5 ou Bits *VCFG0* et *VCFG1* pour Voltage référence ConFiGuration :**

Ces bits permettent de sélectionner les sources de tensions de référence conformément aux tableaux ci-dessous :

VCFG1	VCFG0	V_{ref+} du convertisseur	V_{ref-} du convertisseur
0	0	V_{DD}	V_{SS}
0	1	Entrée V_{ref+}	V_{SS}
1	0	V_{DD}	Entrée V_{ref-}
1	1	Entrée V_{ref+}	Entrée V_{ref-}

• **Bits 6 et 7 :**

Bits non utilisés et lus comme étant des 0.

8.3. Registre ADCON2 :

L : lecture
E : Ecriture

	L/E	L	L/E	L/E	L/E	L/E	L/E	L/E
ADCON2	**ADFM**	**-**	**ACQT2**	**ACQT1**	**ACQT0**	**ADCS2**	**ADCS1**	**ADCS0**
Etat à le mise sous tension	0	0	0	0	0	0	0	0

• Bits 0 à 2 ou Bits *ADCS0* à *ADCS2* pour AD Clock Slect :

Ces trois bits permettent de définir la fréquence de l'horloge de conversion conformément aux indications du tableau ci-dessous :

ADCS2	ADCS1	ADCS0	Fréquence
0	0	0	$F_{osc}/2$
0	0	1	$F_{osc}/8$
0	1	0	$F_{osc}/32$
0	1	1	F_{RC} (oscillateur RC interne)
1	0	0	$F_{osc}/4$
1	0	1	$F_{osc}/16$
1	1	0	$F_{osc}/64$
1	1	1	F_{RC} (oscillateur RC interne)

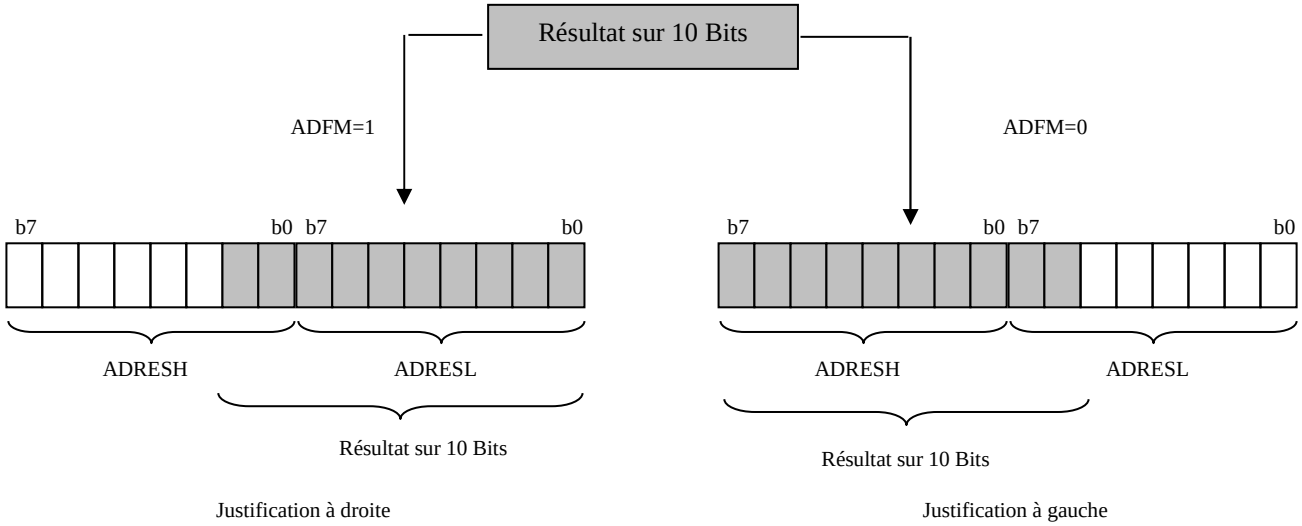
• Bits 3 à 5 ou Bits *ACQT0* à *ACQT2* pour ACQuisition Time select:

Ces trois bits permettent de définir le temps d'acquisition conformément aux indications du tableau ci-dessous :

ACQT2	ACQT1	ACQT0	Temps d'acquisition
0	0	0	0 T_{AD}
0	0	1	2 T_{AD}
0	1	0	4 T_{AD}
0	1	1	6 T_{AD}
1	0	0	8 T_{AD}
1	0	1	12 T_{AD}
1	1	0	16 T_{AD}
1	1	1	20 T_{AD}

• Bits 7 ou Bit *ADFM* pour AD result ForMat select:

S'il est à 1, ce bit positionne le résultat contenu dans ADRESH et ADRESL avec une justification à droite. S'il est à 0, ce bit positionne le résultat avec une justification à gauche.



8.4. Fonctionnement du convertisseur analogique numérique CAN :

Les références de tension à utiliser par le CAN sont sélectionnables par logiciel. Celles-ci peuvent être :

- L'une des deux tensions d'alimentation du µC (Vdd et Vss)
- La tension sur la broche RA3/AN3/VREF+ comme référence « haute »
- La tension sur la broche RA2/AN2/VREF-/CVREF comme référence « basse »

Le convertisseur est utilisable tandis que le processeur est en mode « sleep ». Pour cela, l'horloge du CAN doit être issue de l'oscillateur RC interne.
Chaque ligne associée au convertisseur peut être configurée comme une E/S numérique ou une entrée analogique. Les registres ADRESH et ADRESL reçoivent les résultats des conversions.
Lorsqu'une conversion est achevée, le résultat est chargé dans ces deux registres, le bit ***GO/DONE est positionné à 0*** et le drapeau d'interruption ***ADIF(Registre PIR1)*** est positionné.

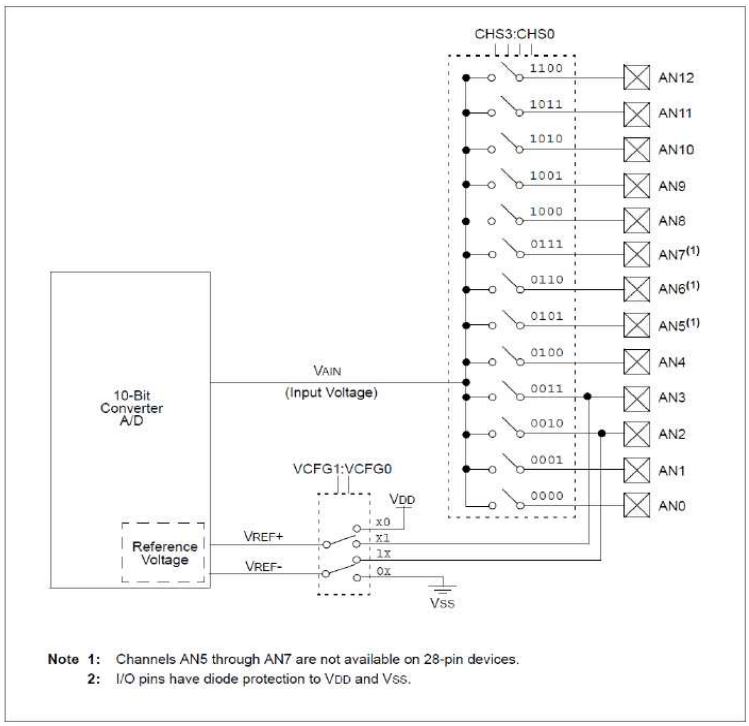


Schéma du convertisseur

8.4.1. Démarche de programmation :

Après que le module convertisseur ait été configuré, la voie sélectionnée doit faire l'objet d'une acquisition avant que la conversion ne démarre. ***Les lignes d'entrées analogiques doivent voir le bit TRIS associé positionné à 1 (entrée)***.
Le temps d'acquisition est programmable. Ce temps correspond à un délai intervenant entre le positionnement du bit GO/DONE à 1 (lancement de la conversion) et le début de la conversion à proprement parler (au niveau du CAN).

La procédure suivante est à suivre pour mettre en oeuvre correctement le convertisseur :

1. **Configurer le module convertisseur :**
   - Configurer les lignes, les références.. (ADCON1)
   - Choisir la voie à convertir (ADCON0)
   - Choisir le temps d'acquisition (ADCON2)
   - Choisir l'horloge de conversion (ADCON2)
   - Activer le module CAN (ADON à 1 dans ADCON0)

2. **Configurer les interruptions (si nécessaire)**
   - ADIF = 0 (Registre PIR1)
   - ADIE = 1 (Registre PIE1)
   - GIE = 1

3. Attendre une durée identique au temps d'acquisition

4. Lancer la conversion : GO/DONE = 1

5. Attendre la fin de la conversion :

- Soit en surveillant le passage de GO/DONE à '0'
- Interruption ADIF

6. Lire le résultat dans ADRESH:ADRESL, ou ADRES repositionner ADIF à 0 si nécessaire

7. Retour à l'étape 1 ou 2 pour une nouvelle acquisition.

Un délai minimal correspondant à 2 Tad (Temps de conversion par bit) est nécessaire avant de relancer une acquisition.

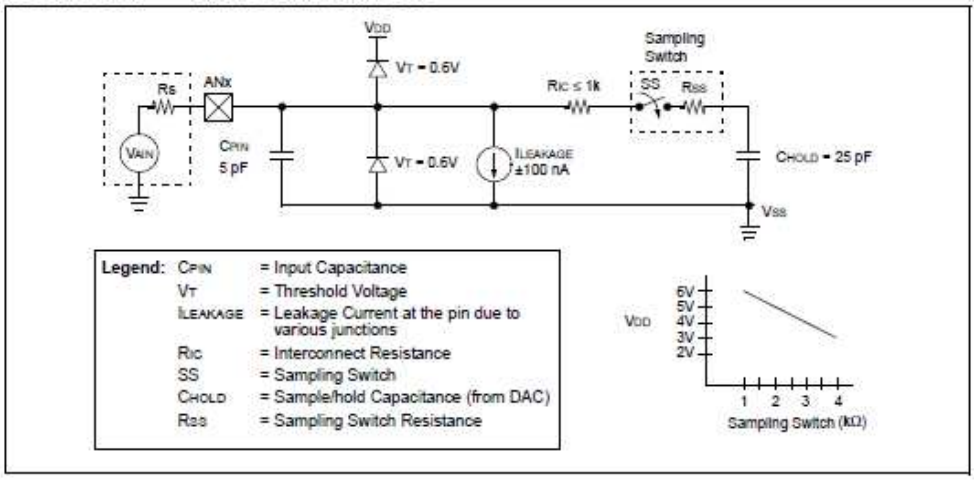
☞ **Remarques :**

- Une mise à 0 de GO/DONE pendant une conversion aura pour effet d'abandonner celle-ci. Les registres ADRES ne seront pas mis à jour.
- Après qu'une conversion soit terminée ou abandonnée, il est nécessaire d'attendre 2 Tad avant d'en relancer une.
- **Le bit GO/DONE ne doit pas être positionné à 1 dans la même instruction que l'activation du CAN (ADON = 1).**

8.4.2. Problème liées à l'acquisition d'une tension :

Pour permettre au convertisseur de fonctionner au maximum de ses possibilités, il est nécessaire que le condensateur d'échantillonnage (CHOLD) ait le temps de se charger complètement à la valeur de la tension appliquée sur l'entrée analogique.

Schéma d'une entrée analogique

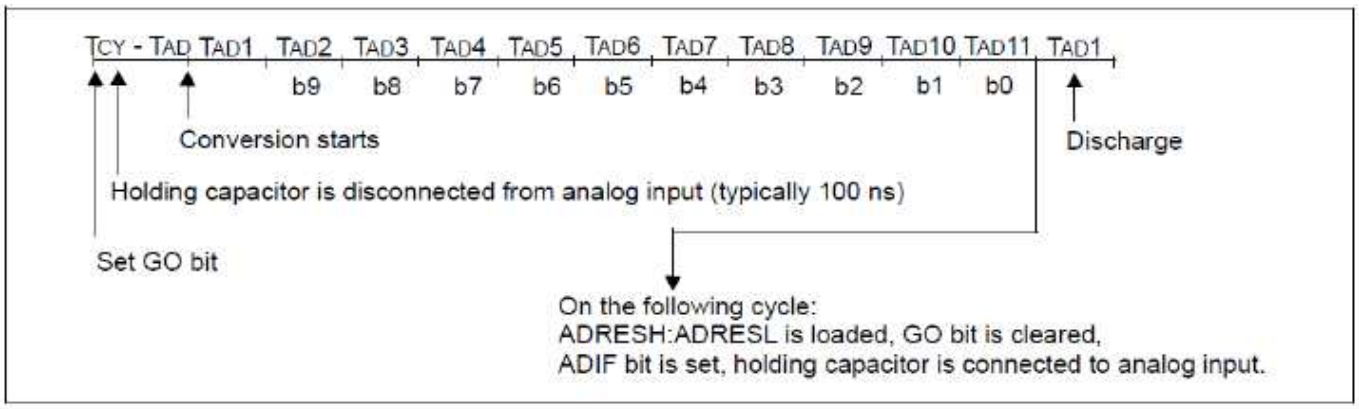


L'impédance de sortie de la source de tension (Rs) ainsi que la résistance de l'interrupteur analogique (Rss) ont une influence directe sur le temps de charge de CHOLD. Dans tous les cas, il est recommandé que **l'impédance de sortie de la source de tension ne dépasse pas 2,5kΩ**.

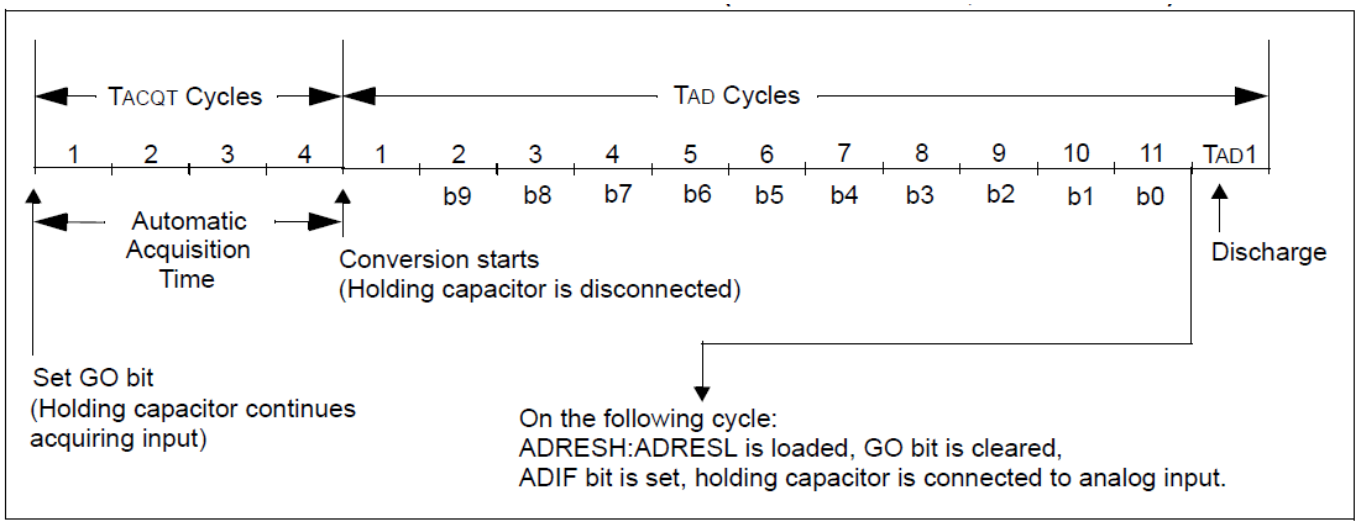
8.4.3. Configurer le temps d'acquisition :

Le registre ***ADCON2*** permet à l'utilisateur de configurer un « temps d'acquisition ». Ce temps intervient entre le moment où le bit GO/DONE est mis à 1 et le moment ou la conversion démarre réellement.
Les bits ***ACQT2 à ACQT0 (ADCON2<5:3>)*** permettent de configurer un temps d'acquisition compris entre 2 et 20 Tad (Tad = période de l'horloge de conversion, 1 Tad = 1 bit). Si un temps d'acquisition est programmé, il n'est pas nécessaire d'attendre entre le moment ou une voie est sélectionnée et celui où l'on place le bit GO/DONE à 1.
Il est possible de ne pas utiliser ce temps d'acquisition (ACQT2:ACQT0 = 000). Dans ce cas, la conversion démarre immédiatement après que le bit GO/DONE ait été positionné à 1. Il est de la responsabilité de l'utilisateur de s'assurer que le délai nécessaire entre la sélection d'une voie et le lancement de la conversion ait eu lieu. Dans tous les cas, lorsque la conversion est terminée, **le bit GO/DONE est remis à 0 et le bit ADIF positionné à 1**.

Cycles T_{AD} du convertisseur avec ACQT<2 :0>=000, T_{ACQ}=0



Cycles T_{AD} du convertisseur avec ACQT<2 :0>=010, T_{ACQ}=4T_{AD}



8.4.4. Choix de l'horloge de conversion :

Le temps de conversion par bit est appelé **Tad**. Une conversion 10 bits **nécessite 11 Tad**. La source permettant de générer l'horloge de conversion est sélectionnable logiciellement parmis 7 possibilités (Tosc = Période de l'oscillateur du µC) :

- 2 Tosc
- 4 Tosc
- 8 Tosc
- 16 Tosc
- 32 Tosc
- 64 Tosc
- Oscillateur RC interne

Pour qu'une conversion se passe bien, il faut que l'horloge de conversion propose une période Tad la plus petite possible tout en restant supérieure au Tad minimum (0,7µs typiquement)

Le tableau suivant aide à définir la configuration de l'horloge :

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18FXXXX	PIC18LFXXXX(4)
2 TOSC	000	2.86 MHz	1.43 kHz
4 TOSC	100	5.71 MHz	2.86 MHz
8 TOSC	001	11.43 MHz	5.72 MHz
16 TOSC	101	22.86 MHz	11.43 MHz
32 TOSC	010	40.0 MHz	22.86 MHz
64 TOSC	110	40.0 MHz	22.86 MHz
RC(3)	x11	1.00 MHz(1)	1.00 MHz(2)

Note 1: The RC source has a typical TAD time of 4 ms.
2: The RC source has a typical TAD time of 6 ms.
3: For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.
4: Low-power devices only.

ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

<ADCS2>	<ADCS1:ADCS0>	Horloge de conversion
0	00	FOSC/2
0	01	FOSC/8
0	10	FOSC/32
0	11	FRC (Provient de l'oscillateur RC interne)
1	00	FOSC/4
1	01	FOSC/16
1	10	FOSC/64
1	11	FRC (Provient de l'oscillateur RC interne)

ADFM: Format du résultat (sur 16 bits)
1 = justification à droite, les 6 bits de poids fort d'ADRESH sont à 0.
0 = justification à gauche, les 6 bits de poids faible d'ADRESL sont à 0.

ACQT2:ACQT0: A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

9) Le timer0

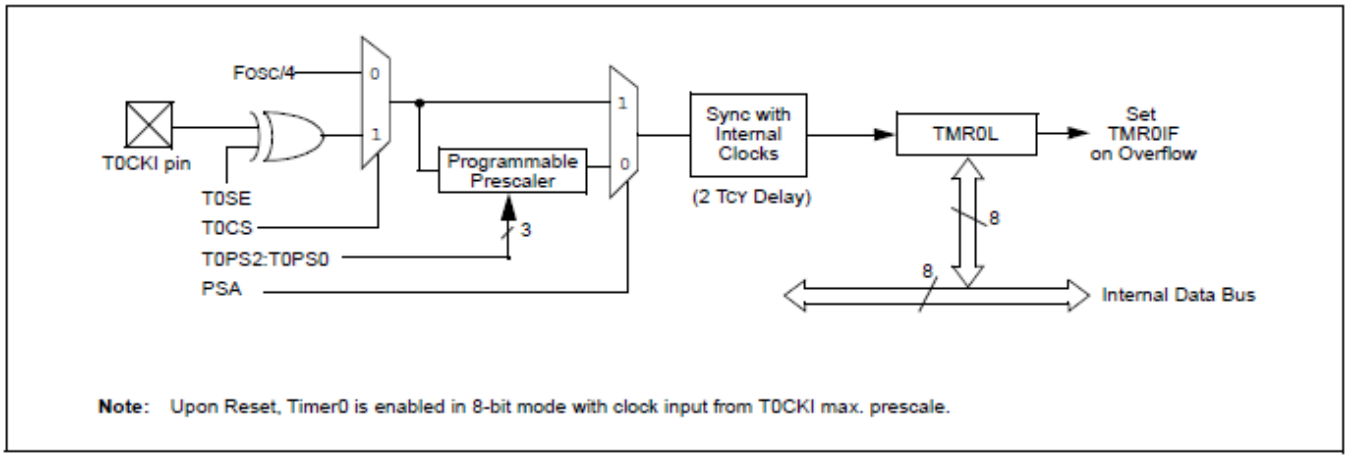
Le timer 0 :

9.1. Fonctionnement du timer 0 :

Le timer 0 présente les caractéristiques suivantes :

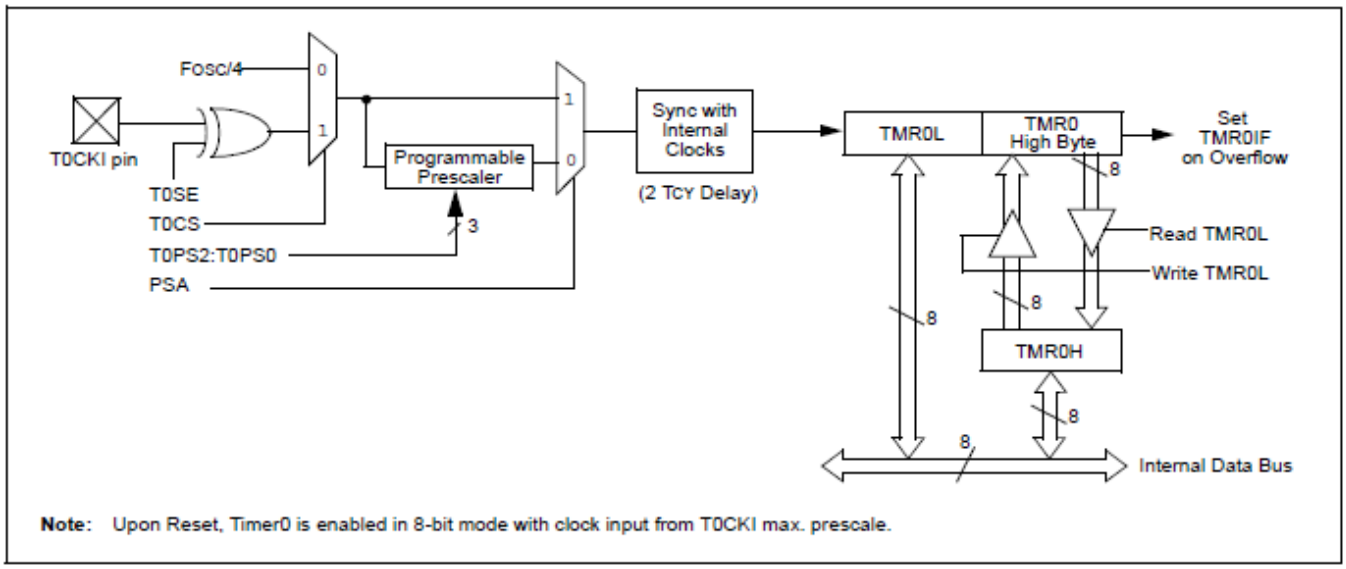
- Timer 8 ou 16 bits. Le mode de fonctionnement (8 ou 16 bits) est sélectionnable à l'aide du bit **T08BIT** du registre **T0CON** (**Mode 8 bits : T08BIT = 1, Mode 16 bits : T08BIT = 0**).
- Registres de comptage **TMR0L** et **TMR0H** accessibles en lecture/écriture.
- Pré diviseur 8 bits programmable à l'aide des bits **T0PS[2..0]** et **PSA** du registre **T0CON**.
- L'horloge peut être interne (timer) ou externe (compteur). Dans ce dernier cas, le front de comptage est configurable. Ces configurations sont à faire dans le registre **T0CON** :
  - T0CS = 0 : Timer (horloge interne)
  - T0CS = 1 : Compteur (horloge externe sur RA4/T0CKI)
    - T0SE = 1 : Incrémentation sur front descendant
    - T0SE = 0 : Incrémentation sur front montant
- Une interruption de type **Overflow** (**T0IF**) peut être déclenchée. Elle est déclenchée par le passage de **0xFF à 0x00 en mode 8 bits** ou par le passage de **0xFFFF à 0x0000 en mode 16 bits**.

9.2. Schéma bloc en mode 8 bits :



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

9.3. Schéma bloc en mode 16 bits :



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

9.4. Registre T0CON :

L : lecture
E : Ecriture

	L/E	L/E	L/E	L/E	L/E	L/E	L/E	L/E
T0CON	**TMR0ON**	**T08BIT**	**T0CS**	**T0SE**	**PSA**	**T0PS2**	**T0PS1**	**T0PS0**
Etat à le mise sous tension	1	1	1	1	1	1	1	1

• Bits 0 à 2 ou Bits *T0PS0* à *T0PS2* pour Timer 0 PreSclaler :

Ces trois bits servent à définir le taux de prédivision appliqué à l »entrée du timer 0 conformément aux tableau ci-dessous :

T0PS2	T0PS1	T0PS0	Taux de division
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

• Bit 3 ou Bit *PSA* pour PreScaler Assignement :

Ce bit permet d'affecter le prédiviseur au timer 0 lorque'il est mis à 0 et de l'ignorer lorsqu'il est mis à 1. Dans ce cas l'état des bits T0PS0 à T0PS2 est sans importance.

• Bit 4 ou Bit *T0SE* pour Timer 0 Source Edge :

Lorsqu'il est à 0, ce bit indique que le timer 0 est incrémenté sur un front descendant du signal appliqué sur l'entrée T0CK1. Il est incrémenté sur front montant dans le cas contraire.

• Bit 5 ou Bit *T0CS* pour Timer 0 Clock Select :

Lorsqu'il est à 0, ce bit sélectionne l'horloge instruction ($F_{osc}/4$) comme horloge du timer 0. Il sélectionne l'entrée T0CK1 lorsqu'il est à 1.

• Bit 6 ou Bit *T08BIT* pour Timer 0 8 BITs :

Lorsqu'il est à 0, ce bit sélectionne le mode de fonctionnement 16 bits. Il valide le mode 8 bits dans le cas contraire.

• Bit 7 ou Bit *TMR0ON* pour TiMeR 0 ON :

Lorsqu'il est à 0, ce bit arrête le timer 0 alors qu'il lui permet de fonctionner dans le cas contraire.

REGISTRE T0CON

7	6	5	4	3	2	1	0
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

TMR0ON: activation
1 = Active Timer0
0 = Stoppe Timer0
T08BIT: 8/16 bits
1 = Timer0 est un compteur 8-bits
0 = Timer0 est un compteur 16-bits
T0CS: sélection de l'horloge
1 = compte les fronts sur T0CKI
0 = compte les fronts sur l'horloge interne (CLKOUT)
T0SE: front détecté
1 = compte sur front descendant sur T0CKI
0 = compte sur front montant sur T0CKI
PSA: Pré diviser
1 = pas de pré diviseur.
0 = Le pré diviseur est activé
T0PS2:T0PS0: Pré Division de l'horloge
111 = 1:256
110 = 1:128
101 = 1:64
100 = 1:32
011 = 1:16
010 = 1:8
001 = 1:4
000 = 1:2

Pour utiliser l'interruption :
TMR0F dans INTCON <5> (drapeau d'interruption)
TMR0E dans INTCON <2> (validation de l'interruption)

```
// Demo pour TMER0
#include <p18f452.h>  //pour LCD

// sous programme d'interruption
#pragma interrupt itcomp
#pragma code interruption=0x8
void itcomp(void)
{
        PORTB^=0x01; // bascule PB0
        INTCONbits.TMR0IF=0;
}
#pragma code

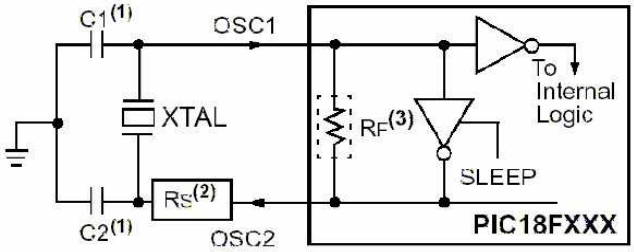
void main(void)
{
TRISB=0xFE;            // PB0 en sortie
// active Timer 16bits sur CLKOUT
// prediviseur 1/8
// avec Q=4MHz, CLK=1uS
// IT toutes les 1*8*65536= 524mS
T0CON=0b10000010;
INTCONbits.TMR0IE=1;// autorise IT débordement
RCONbits.IPEN=1;// Interruption prioritaires
INTCONbits.GIE=1;
While(1);      // ne rien faire
}
```

10) L'horloge du PIC18F4550

Choix de l'horloge

1. LP Quartz faible puissance
2. XT Quartz ou oscillateur externe
3. HS Quartz ou oscillateur externe rapide
4. HS + PLL (F x 4)

Types	Fréquences	C1	C2
LP	200kHz	15 pF	15 pF
XT	200 kHz	47-68pF	47-68 pF
	1.0MHz	15 pF	15 pF
	4.0MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0MHz	15-33 pF	15-33 pF
	20.0MHz	15-33pF	15-33 pF



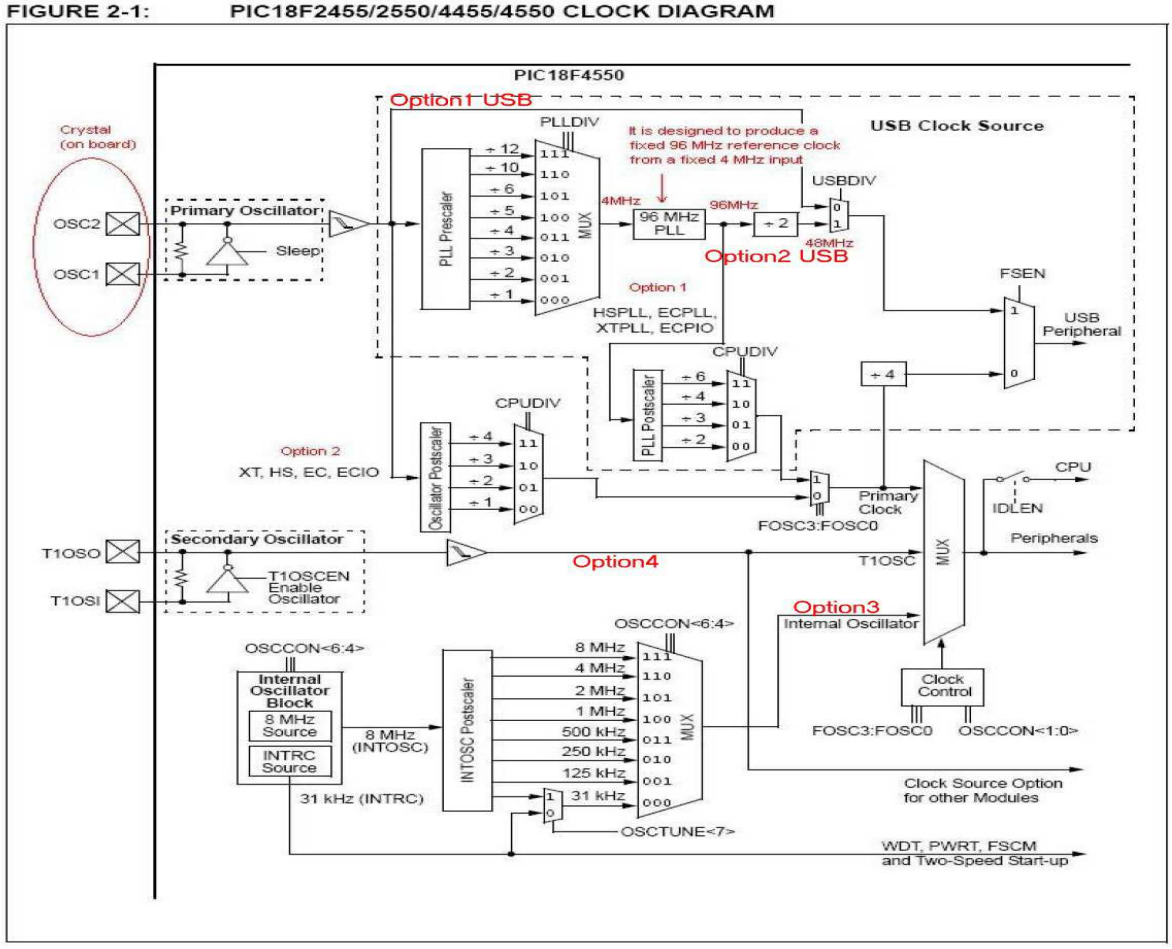
Le diagramme du circuit d'horloge est donné ci-dessous :

Pour l'horloge tu dois fournir :

- 48MHz au module USB
- L'horloge au CPU

Le diagramme du circuit d'horloge est donné ci-dessous :

FIGURE 2-1: PIC18F2455/2550/4455/4550 CLOCK DIAGRAM



Pour l'horloge on doit fournir :

- 48MHz au module USB
- L'horloge au CPU

La configuration de l'horloge sous MPLAB s'effectue de la manière suivante :

Le module USB a été conçu pour produire une horloge de reference à 96MHz à partir d'une entrée de 4MHz .

Exemple pour un quartz de 8MHz

1) Prépositionne PLLDIV Pour le module USB.

Le prédiviseur de la PLL (PLLDIV) Divisera 8MHZ par 2 et produira une fréquence de 4MHz absolument Nécéssaire pour l'entrée de la PLL.

```
000        00     PLLDIV    PLL Prescaler Selection bits

No prescale (4 MHz oscillator input drives PLL directly)
Divide by 12 (48 MHz oscillator input)
Divide by 10 (40 MHz oscillator input)
Divide by 6 (24 MHz oscillator input)
Divide by 5 (20 MHz oscillator input)
Divide by 4 (16 MHz oscillator input)
Divide by 3 (12 MHz oscillator input)
Divide by 2 (8 MHz oscillator input)
No prescale (4 MHz oscillator input drives PLL directly)
```

Choix possible

2) Choix de l'horloge CPU

- **Option 1 PLL active**

Si tu choisis HS OSCILLATOR avec PLL Validé

```
   300001        05     FOSC      Oscillator Selection bits

EC oscillator, CLKO function on RA6 (EC)
HS oscillator, PLL enabled (HSPLL)
HS oscillator (HS)
Internal oscillator, HS oscillator used by USB (INTHS)
Internal oscillator, XT used by USB (INTXT)
Internal oscillator, CLKO function on RA6, EC used by USB (INTCKO)
Internal oscillator, port function on RA6, EC used by USB (INTIO)
EC oscillator, PLL enabled, CLKO function on RA6 (ECPLL)
EC oscillator, PLL enabled, port function on RA6 (ECPIO)
EC oscillator, CLKO function on RA6 (EC)
EC oscillator, port function on RA6 (ECIO)
XT oscillator, PLL enabled (XTPLL)
XT oscillator (XT)
```

Choix possible

Dans ce cas la CPU utilisera 96MHz HSPLL divisé par le prédiviseur (PLL Postcaler) , Prépositionne les flags CPUDIV pour choisir la fréquence du CPU.

```
                    CPUDIV    System Clock Postscaler Selection bits
[Primary Oscillator Src: /1][96 MHz PLL Src: /2]
[Primary Oscillator Src: /4][96 MHz PLL Src: /6]
[Primary Oscillator Src: /3][96 MHz PLL Src: /4]
[Primary Oscillator Src: /2][96 MHz PLL Src: /3]
[Primary Oscillator Src: /1][96 MHz PLL Src: /2]
```

Choix possible

Si CPUDIV est programmé à la valeur 2 Le CPU travaillera à 48MHz .

…

Si CPUDIV est programmé à 6 On divisera 96MHz par 6 et le CPU travaillera à 16MHz

- **Option 2** PLL inactive

Si tu choisis *HS* oscillator

```
   300001       05     FOSC      Oscillator Selection bits
EC oscillator, CLKO function on RA6 (EC)
HS oscillator, PLL enabled (HSPLL)
HS oscillator (HS)
Internal oscillator, HS oscillator used by USB (INTHS)
Internal oscillator, XT used by USB (INTXT)
Internal oscillator, CLKO function on RA6, EC used by USB (INTCKO)
Internal oscillator, port function on RA6, EC used by USB (INTIO)
EC oscillator, PLL enabled, CLKO function on RA6 (ECPLL)
EC oscillator, PLL enabled, port function on RA6 (ECPIO)
EC oscillator, CLKO function on RA6 (EC)
EC oscillator, port function on RA6 (ECIO)
XT oscillator, PLL enabled (XTPLL)
XT oscillator (XT)
```

Choix sans PLL

Choix possible

Dans ce cas le CPU Utilisera 8MHz divisé par le prédiviseur (Oscillator Postscaller), prépositionne CPUDIV Pour choisir la valeur de division.

```
000        00     PLLDIV    PLL Prescaler Selection bits

No prescale (4 MHz oscillator input drives PLL directly)
Divide by 12 (48 MHz oscillator input)
Divide by 10 (40 MHz oscillator input)
Divide by 6 (24 MHz oscillator input)
Divide by 5 (20 MHz oscillator input)
Divide by 4 (16 MHz oscillator input)
Divide by 3 (12 MHz oscillator input)
Divide by 2 (8 MHz oscillator input)
No prescale (4 MHz oscillator input drives PLL directly)
```

Choix possible

Si le prédiviseur est programmé à 1 l' horloge du CPU sera donc de 8MHz
Il existe aussi deux autres modes de fonctionnement le mode oscillateur interne (option 3) et le mode oscillateur secondaire (option 4).

```
   300001        05     FOSC      Oscillator Selection bits

EC oscillator, CLKO function on RA6 (EC)
HS oscillator, PLL enabled (HSPLL)
HS oscillator (HS)
Internal oscillator, HS oscillator used by USB (INTHS)
Internal oscillator, XT used by USB (INTXT)
Internal oscillator, CLKO function on RA6, EC used by USB (INTCKO)
Internal oscillator, port function on RA6, EC used by USB (INTIO)
EC oscillator, PLL enabled, CLKO function on RA6 (ECPLL)
EC oscillator, PLL enabled, port function on RA6 (ECPIO)
EC oscillator, CLKO function on RA6 (EC)
EC oscillator, port function on RA6 (ECIO)
XT oscillator, PLL enabled (XTPLL)
XT oscillator (XT)
```

Choix sans PLL

Choix possible

L'horloge du module USB est choisi en utilisant 2 modes de fonctionnement:
Option1 USB et Option2 USB

```
USBDIV    USB Clock Selection bit (used in Full-Speed USB mode only; UCFG:FSEN = 1)

USB clock source comes directly from the primary oscillator block with no postscale
USB clock source comes from the 96 MHz PLL divided by 2
USB clock source comes directly from the primary oscillator block with no postscale
```

Option2

Option1