1. Définition

Le Timer est un compteur de cycles d'instructions. Il suffit de lire régulièrement ce compteur pour évaluer le temps qui s'écoule. Il est également possible d'initialiser ce compteur avec le nombre de notre choix. Il faut aussi noter que ce compteur doit être configuré pour fonctionner.

2. Constitution du timer

Le timer est constitué de :

- ✓ le TMR0. C'est un registre de 8 bits, qui compte les fronts montants du signal qu'on lui fournit. Arrivé à 255, il repasse à 0. Il est accessible en lecture ou écriture de la même façon que le PORTA, le PORTB ou les variables.

- ✓ le pré diviseur. Il divise la fréquence du signal qu'on lui fournit par une constante qu'il est nécessaire de programmer.

Ex : un signal de période 1us divisé par 32 donnera un signal de période 32 us.

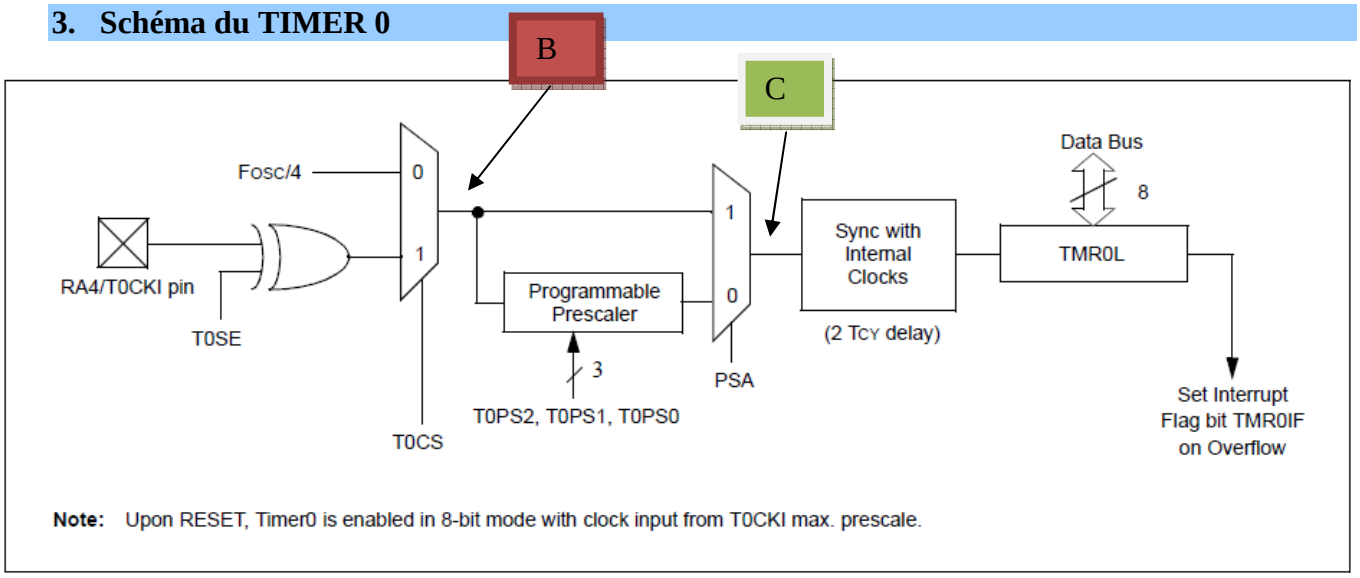
- ✓ les aiguillages. Ils permettent de définir le chemin qu'emprunteront les informations (sortie des aiguillages en B et C).

- ✓ les bits de configuration (RTE, RTS …). Ils permettent de définir les différentes fonctions.

- ✓ la synchro. Retarde le début du fonctionnement de 2 cycles d'instruction lorsqu'un bit relatif au timer est modifié. Cette fonction est due à l'architecture du timer et ne peut être contournée.

Voici le schéma du timer. Pas d'affolement, nous allons l'expliquer.

3. Schéma du TIMER 0



A gauche, sont notées les entrées du timer. Elles sont au nombre de deux :

- le signal du quartz divisé par 4 (soit un signal de période 1us) et la patte RA4.

- L'entrée RA4 attaque une porte "ou exclusif" qui fournit un signal dépendant du bit TOSE selon le tableau ci-contre :

TOSE =0	**Entrée active sur front descendant**
TOSE =1	Entrée active sur front montant

- Le signal en B dépend de l'aiguillage commandé par TOCS.

TOCS =0	**B = fosc/4**
TOCS =1	B = RA4

- Le signal en C dépend de l'aiguillage commandé par PSA.

PSA =0	**C = Prédiviseur**
PSA = 1	C = B

- Les bits TOPS0 à TOPS2 configure la prédivision.
- La synchro retarde de 2 cycles d'instruction le début du comptage.
- TMR0 compte les fronts montants

4. Configuration du Timer

La configuration du timer se fait par la configuration des bits vus précédemment. Ces bits sont accessibles par le registre appelé TOCON

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

5. Utilisation du timer

5.1 Fonctionnement en mode polling (scrutation du bit T0IF ou lecture de TMR0)

L'utilisation du timer est assez simple et se fait en 3 étapes.

- mise à jour du registre OPTION
- lecture du TMR0
- mise à jour du TMR0

Ce mode de fonctionnement consiste à ne pas utiliser le fonctionnement en interruption et à lire régulièrement le bit T0IF, afin de détecter le débordement, ou à lire la valeur de TMR0 afin de déterminer le temps.

Ce mode est utilisé dans les cas simples ou le seul traitement à effectuer est une attente, et ou la boucle d'attente du positionnement T0IF est suffisant (et même plus rapide que l'interruption).

Utilisation d'une temporisation précise en langage C en utilisant l'indicateur T0IF.

```
//********************** Tempo ****************************
// la variable duree donne la valeur de la temporisation effectuée par le Timer
void Tempo(unsigned char duree)
{ T0CS=0; // Clock select sur Horloge interne Fq/4
PSA=0; // prédiviseur sur TMR0
PS2=0; PS1=1; PS0 =1; // H TMR0 = Hq/4 / 16 = 16µs si Quartz 4Mhz
T0IF=0; // RàZ Flag débordement TMR0
TMR0=duree;
while (T0IF==0);
}
```

La fonction (Sous programme) Tempo est appelée en lui transmettant le paramètre durée. Ex: Tempo(200)
Le Timer0 est initialisé avec TMR0=128, T0CS=0 soit horloge interne (Fh = Fq/4) soit si Fq=4MHz alors Fh =1MHz, soit Th = 1µs.
PSA=0 et PS2,PS1,PS0 = 0,1,1 soit un prédiviseur avec une division par 16.
Le compteur Timer0 est donc incrémenté toutes les 16µs. (Horloge compteur Thc=16µs).
Si durée=200, le compteur doit compter de 200 jusqu'à 255 puis passer à 0 pour provoquer le passage de T0IF à 1. Il faut donc attendre 256-200=56 période de l'horloge du compteur. Soit
Ttotal=56*Thc=56*16µs = 896µs

Remarques :

- A chaque mise à jour du registre TMR0 ou d'un bit de configuration, 2 cycles d'instructions sont nécessaires avant le commencement du comptage.
- La lecture d'un bit par contre ne perturbe pas le comptage

L'intérêt du timer est d'être précis, de permettre d'effectuer des opérations pendant la temporisation. Son inconvénient est d'être difficile à programmer, nécessite la réalisation de comparaison régulièrement pendant le programme. Il rend donc difficile l'utilisation de boucle longue. Pour palier au dernier inconvénient, il sera nécessaire d'avoir recours aux interruptions. Certains pic ont plusieurs timers. Il est donc nécessaire en fonction de la complexité de l'application à réaliser de choisir le pic le plus approprié. Il sera alors indispensable de bien lire la documentation du pic afin de pouvoir le programmer.

5.2 Utilisation du mode de fonctionnement interruption

Une interruptions 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***.

Autorisation des interruptions:

Bit **GIE**(Global Interrupt Enable) du registre **INTCON** (Interrupt Control), et**T0IE** (Timer0 Interrupt Enable). Ces bits doivent être mis à 1 pour autoriser les interruptions. (Voir détail des bits du registre INTCON).

FIGURE 4-1: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

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	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **GIE:** Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

Note: For the operation of the interrupt structure, please refer to Section 8.5.

bit 6: **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 5: **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4: **INTE**: RB0/INT Interrupt Enable bit
1 = Enables the RB0/INT interrupt
0 = Disables the RB0/INT interrupt

bit 3: **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2: **T0IF**: TMR0 overflow interrupt flag bit
1 = TMR0 has overflowed (must be cleared in software)
0 = TMR0 did not overflow

bit 1: **INTF**: RB0/INT Interrupt Flag bit
1 = The RB0/INT interrupt occurred
0 = The RB0/INT interrupt did not occur

bit 0: **RBIF**: RB Port Change Interrupt Flag bit
1 = When at least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Mécanisme de prise en compte et du traitement d'une interruption:

- Lors de la demande d'interruption (T0IF = 1), GIE est automatiquement mis à 0 pour interdire toute nouvelle interruption.
- La valeur du PC est sauvegardée dans la pile (adresse de retour).
- PC est chargé par la valeur de l'adresse de l'interruption.
- On exécute le programme interruption.
- Libération de l'interruption:
  L'indicateur "flag" mis à 1 indiquant le périphérique source de la demande d'interruption devra être remis à 0 pour qu'une nouvelle demande d'interruption future puisse être prise en compte.
- Sortie du sous programme d'interruption.