INFO2 : INFORMATIQUE EMBARQUEE
TD5 : Le timer 0 du PIC18F4520

1 Principe de fonctionnement des timers

1.1 Introduction

Tous les microcontrôleurs ont un (ou plusieurs) module périphérique appelé ***timer*** pour gérer le temps. Une première utilisation, objet de ce TD, est de déclencher un évènement lorsqu'un temps prédéfini s'est écoulé ; c'est la génération de durée. Une autre utilisation possible, non traitée ici, est de mesurer un temps écoulé entre une origine et la survenance d'un évènement, c'est la mesure de durée (le timer est dit en *capture mode* dans ce cas).

Le PIC 18F4520 possède 4 timers : le timer 0, le timer 1, le timer 2 et le timer 3. On s'attache ici au descriptif du fonctionnement du timer 0.

Dans l'utilisation qui nous intéresse ici (la génération de durée), le timer est lancé par le programme et compte des impulsions survenant à intervalle régulier. Quand un nombre prédéfini d'impulsions est atteint, une durée connue s'est écoulée.

Ces impulsions peuvent provenir :

- Du quartz cadençant le microcontrôleur. Sa fréquence est notée F_{quartz}.
- D'une entrée binaire du PIC (par exemple : RA4 pour le timer 0, RC0 pour le timer 1), par exemple si l'on veut utiliser une horloge extérieure ou compter des événements. Cette option ne sera pas abordée en détail ici.

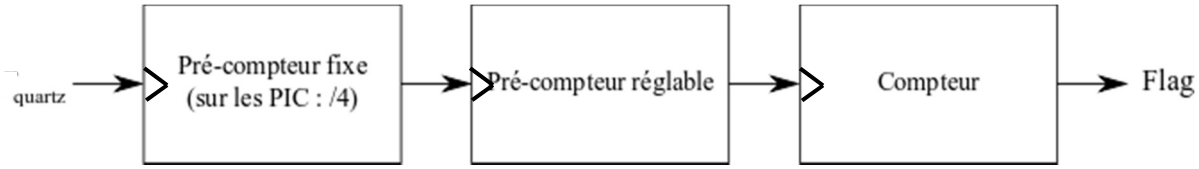
Quand le nombre prédéfini d'impulsions est atteint, un **drapeau** (*flag*) se **lève**. C'est un simple indicateur binaire qui passe de 0 à 1. Une interruption peut être également générée (cf. chapitre sur les interruptions).

Les timers ont presque tous le même fonctionnement :

Ils possèdent un **compteur** (*counter*), un **pré-compteur** (*prescaler*), et parfois un **post-compteur** (non représenté sur la figure ci-dessous).

L'horloge du microcontrôleur est pré-divisée pour obtenir une fréquence plus lente. La sortie du pré-compteur attaque ensuite un compteur.

Dans le cas des PIC, l'horloge du quartz n'est pas directement connectée au pré-compteur mais passe par un diviseur par 4 (« Pré-compteur fixe » dans la figure ci-dessous).



On règle le compteur à une **valeur initiale** et on attend qu'il atteigne une **valeur finale**. La différence entre les deux constitue le « Nombre d'impulsions à compter » de la formule ci-dessous.

Lorsque le compteur atteint cette valeur finale, le *flag* devient actif.

Le temps *T* mis par le compteur pour faire lever le *flag* est égal à :

$$T = \frac{1}{F_{quartz}} \times \text{Valeur du précompteur fixe} \times \text{Valeur du précompteur réglable} \times \text{Nombre d'impulsions à compter}$$

Certains timers utilisent pour le compteur le principe des **compteurs**, d'autres celui des **décompteurs.**

1.2 Les compteurs

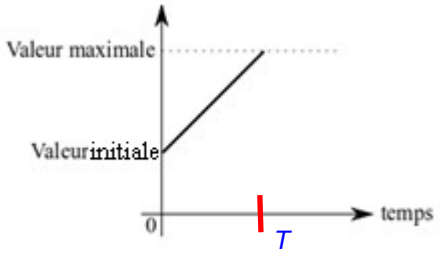
Il en existe deux types :

- ceux qui partent d'une valeur prédéfinie pour arriver à la valeur maximum ;
- ceux qui partent de 0 pour arriver à une valeur prédéfinie.

1.2.1 Compteurs qui partent d'une valeur prédéfinie pour arriver à la valeur max

C'est le cas du timer 0 du PIC18F4520.

Le timer 0 peut être utilisé en mode 8 bits (la valeur maximum est alors 0xFF), mais également en mode 16 bits pour obtenir des temps plus longs (la valeur maximum est alors 0xFFFF).



Dans le cas du **timer 0 en mode 8 bits**, la formule du temps devient :

$$T = \frac{1}{F_{quartz}} \times 4 \times \text{Valeur du précompteur réglable} \times (256 - \text{Valeur initiale})$$

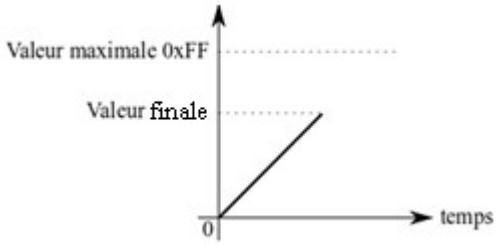
Dans le cas du **timer 0 en mode 16 bits**, la formule du temps devient :

$$T = \frac{1}{F_{quartz}} \times 4 \times \text{Valeur du précompteur réglable} \times (65536 - \text{Valeur initiale})$$

1.2.2 Compteurs qui partent de 0 pour arriver à une valeur prédéfinie

C'est le cas d'un autre timer du PIC18F4520 (timer 2).

La valeur max est 0xFF car c'est un timer 8 bits.



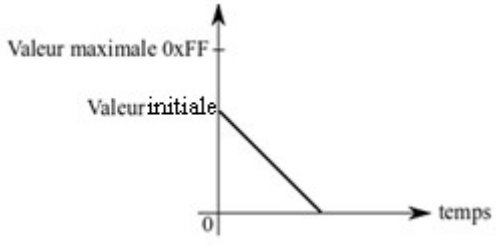
Dans ce cas, la formule du temps devient :

$$T = \frac{1}{F_{quartz}} \times 4 \times \text{Valeur du précompteur réglable} \times \text{Valeur finale}$$

1.3 Les décompteurs

On fixe une valeur de départ et on décompte jusqu'à zéro.

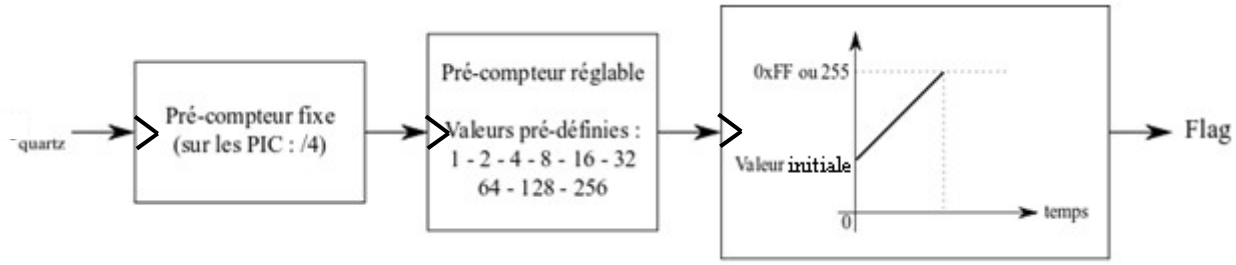
Ce type de timer est utilisé par exemple dans les microcontrôleurs ST6 de ST-Microelectronic.



2 Timer 0 du PIC 18F4520 en mode 8 bits pour la génération de durée

2.1 Principe de fonctionnement du timer en compteur de temps

Nous avons vu que le timer 0 est basé sur le principe :



Remarque : Comme on le voit ci-dessus, le pré-compteur réglable ne peut prendre qu'une valeur parmi 9 valeurs discrètes (1, 2, 4, 8, 16, 32, 64, 128, 256).

2.1.1 Calcul du temps maximum

Calculons le temps maximum T_{MAX} qui peut se passer avant que le *flag* ne se lève :

On doit prendre la valeur de pré-compteur maximale et faire partir le compteur de 0 : $T_{MAX} = \frac{1}{F_{quartz}} \times 4 \times 256 \times (256 - 0)$

Avec un quartz de 8 MHz par exemple : $T_{MAX} = \frac{1}{8000000} \times 4 \times 256 \times (256 - 0) = 32{,}768\ ms$

2.1.2 Calcul d'un temps particulier

Calculons les réglages à effectuer pour que le flag se lève au bout de T = 10 ms (bien-sûr $T < T_{MAX}$). Pour trouver les valeurs à mettre dans le pré-compteur et dans le compteur, il nous faut résoudre l'équation suivante :

$$10\ ms = \frac{1}{F_{quartz}} \times 4 \times Valeur\ du\ précompteur\ réglable \times (256 - \text{Valeur initiale})$$

Soit : Valeur du pré-compteur réglable × (256 − Valeur initiale) $= \frac{10\text{ ms} \times F_{quartz}}{4}$

Avec un quartz de 8 MHz, on obtient la formule suivante :

Valeur du pré-compteur réglable × (256 − Valeur initiale) = 20000

Nous avons deux inconnues et une formule, il faut imposer une des deux inconnues et trouver l'autre.

Il est préférable de fixer le pré-compteur puisqu'il ne peut prendre que 9 valeurs discrètes. On dresse donc un tableau comme celui-ci-dessous. La ligne avec la valeur de pré-compteur à 1 traduit l'absence d'utilisation du pré-compteur (cf. **partie 2.2.4** pour ce réglage).

Pré-compteur	256 – Valeur initiale	Valeur initiale	TMR0L = Valeur initiale arrondie
1	20000	-19744	-19744
2	10000	-9744	-9744
4	5000	-4744	-4744
8	2500	-2244	-2244
16	1250	-994	-994
32	625	-369	-369
64	312.5	-56.5	-57
128	156.25	99.75	100
256	78.125	177.875	178

Dans ce tableau, la dernière colonne TMR0L représente la valeur initiale qu'il faudra mettre dans le registre 8 bits TMR0L. Cette valeur ne peut être qu'entière et comprise entre 0 et 255. Les lignes grisées montrent donc des couples impossibles du pré-compteur et de TMR0L.

On remarque qu'il est possible de choisir deux couples de valeurs. Lequel prendre ?

Pour le savoir, il faut faire le calcul de la temporisation réelle et prendre le couple de valeur qui s'approche le plus de 10 ms :

- $T = (1 / 8\,000\,000) * (4 * 128 * (256 - 100)) = 9.984$ ms
- $T = (1 / 8\,000\,000) * (4 * 256 * (256 - 178)) = 9.984$ ms

Dans cet exemple, aucun des deux couples ne produit une durée plus correcte que l'autre. On peut donc a priori prendre n'importe quel couple de valeurs.

Cela dit, en règle générale, on choisit le couple ayant le plus faible pré-compteur parmi tous les couples possibles. Ainsi la fréquence en entrée du compteur est la plus grande possible et la résolution du comptage est meilleure. C'est donc l'avant dernière ligne qu'il faut retenir.

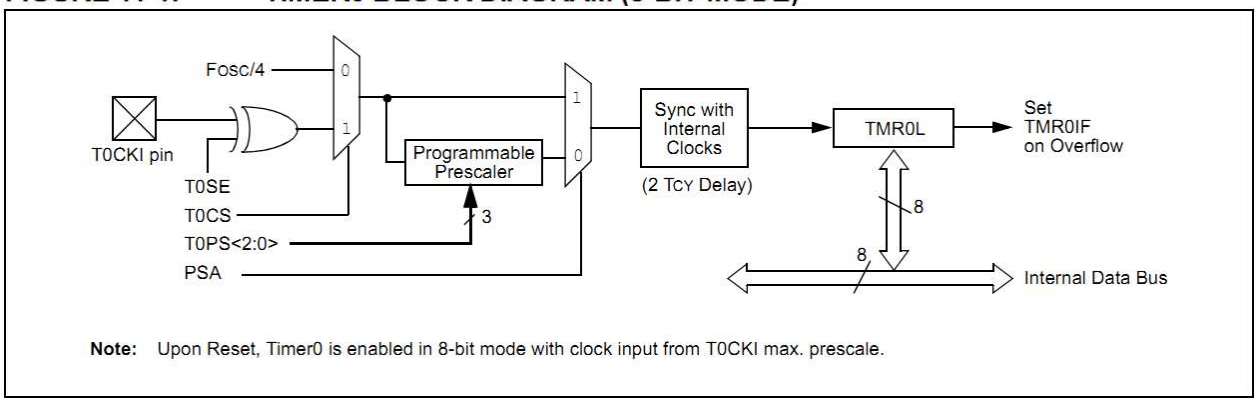
2.2 Le timer 0 de façon plus complète

2.2.1 Schéma fonctionnel

Le timer 0 est légèrement plus complexe que ce que nous avons décrit dans les paragraphes précédents.

La figure suivante, extraite de la *datasheet* du PIC (**`39631E.pdf`**), récapitule son fonctionnement.

FIGURE 11-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)



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

2.2.2 Sélection de l'horloge

Le timer 0 possède deux sources d'horloges :

- Soit l'horloge du PIC (en général un quartz) divisée par 4, repérer par Fosc/4 sur la figure.
- Soit une horloge externe connectée sur la broche RA4 / T0CKI (*Timer 0 ClocK In*) ; c'est lorsque l'on utilise le timer en compteur d'événements extérieurs (détection du passage d'une porte ou signal provenant d'un autre circuit intégré par exemple).

Le choix entre ces deux horloges se fait à l'aide du bit T0CS (*Timer0 Clock Source*) du registre T0CON (*Timer 0 CONtrol*).

Sur la figure précédente, on voit que nous devons mettre T0CS à 0 pour valider l'horloge du PIC (en général un quartz).

Bit T0CS = 0

2.2.3 Sélection du type de front lors de l'utilisation de l'horloge externe

Si on utilise la broche RA4 / T0CKI comme entrée d'horloge, on peut, grâce au bit T0SE (*Timer 0 Source Edge*) du registre T0CON, préciser si l'on veut que le compteur s'incrémente sur un front montant de l'horloge externe (T0SE = 0) ou sur un front descendant (TOSE = 1).

Remarque : Compte-tenu du fait que nous n'utilisons pas l'entrée RA4, on peut mettre n'importe quelle valeur dans le bit T0SE.

Bit T0SE : quelconque (choisir 0 par simplicité)

2.2.4 Pré-compteur (prescaler)

Sur la figure précédente, on voit que le bit PSA (*PreScaler Assignment*) du registre T0CON permet d'utiliser le pré-compteur (PSA = 0) ou pas (PSA = 1). Dans ce dernier cas, il n'y a donc pas de pré-division.

2.2.5 Synchronisation

Sur la figure précédente, on voit que le signal sortant du multiplexeur est ensuite synchronisé avec l'horloge du PIC. Cette synchronisation est utile uniquement si on utilise une horloge externe.

2.2.6 Registres de configuration du timer 0

Tous les registres ou les bits utilisés pour le timer 0 sont indiqués page 125 de la *datasheet* du PIC (**`39631E.pdf`**) :

TABLE 11-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TMR0L	Timer0 Register Low Byte								50
TMR0H	Timer0 Register High Byte								50
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	50
TRISA	RA7(1)	RA6(1)	RA5	RA4	RA3	RA2	RA1	RA0	52

Il y a cinq registres qui nous intéressent :

- TMR0L : Unique registre de comptage utilisé en mode 8 bits. C'est dans ce registre que nous allons rentrer la valeur de départ de notre compteur.
- TMR0H : Deuxième registre de comptage utilisé en mode 16 bits. Dans ce mode, l'octet de poids faible de la valeur de départ du compteur est rentré dans TMR0L et l'octet de poids fort dans TMR0H.
- INTCON : Seuls les bits 7, 6, 5 et 2 sont utiles pour le timer 0 (ce sont les seuls non grisés). Dans ce chapitre, nous ne nous intéresserons qu'au bit 2 appelé TMR0IF et qui correspond au flag permettant de tester la fin du comptage. Les bits 7, 6 et 5 sont en lien avec les interruptions.
- TRISA est à configurer correctement au cas où l'entrée d'impulsion est la patte RA4 et non le quartz. Dans ce cas RA4 est une entrée.
- Registre T0CON :

La page 123 de la *datasheet* du PIC (**`39631E.pdf`**) nous informe sur l'utilisation des différents bits :

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

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 **T0PS<2:0>**: 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

2.3 Exemple de programme avec le timer 0

Cahier des charges : Toutes les 10 ms, inverser l'état de la LED connectée à RD0. Les aspects temporels seront gérés par le timer 0 en mode 8 bits. Le quartz est à 8 MHz.

Quelques pages en arrière, nous avions trouvé pour un quartz à 8 MHz des valeurs initiales du compteur de 100 et du pré-compteur de 128 afin d'obtenir une durée de 10 ms. Voici un exemple possible de programme :

```
#include <p18f4520.h>
#include "../configuration_bits.h"

void main(void)
{
    TRISDbits.TRISD0 = 0;   // RD0 en sortie
    T0CON = 0xC6;           // pré-compteur = 128, mode 8 bits, timer on, horloge interne
    INTCONbits.TMR0IF = 0;  // On met à 0 le drapeau.
    TMR0L = 100;            // On charge la valeur initiale du timer.

    while(1)
    {
        if(INTCONbits.TMR0IF == 1)                // Si la durée s'est écoulée :
        {
            INTCONbits.TMR0IF = 0;                //      // On remet à 0 le drapeau.
            TMR0L = 100;                          //      // On recharge la valeur initiale du timer.
            LATDbits.LATD0 = !LATDbits.LATD0;     //      // On change l'état de la LED.
        }
    }
}
```

1. Quelle doit être la durée maximum d'exécution du bloc de code entouré en pointillé si on ne veut pas rater de lever de drapeau ?

3 Timer 0 du PIC 18F4520 en mode 16 bits pour la génération de durée

On l'a vu avec l'exemple précédent, les durées générées avec le timer 0 en mode 8 bits sont de l'ordre de quelques millisecondes pour des quartz de quelques MHz. Si ces durées sont insuffisantes, il faut passer en mode 16 bits (bit T08BIT du registre T0CON à ajuster en conséquence) pour compter pendant plus longtemps avant que le *flag* ne se lève.

La formule de calcul à utiliser est la deuxième de la **partie 1.2.1**. La valeur initiale est alors précisée sur 16 bits dans TMR0H et TMR0L.

4 Exercices

4.1 Timer 0 en mode 8 bits

2. Ecrire un programme qui fait changer l'état de la LED connectée à RD0 toutes les 2.5 ms, le quartz étant à 40 MHz. On vérifiera préalablement qu'une telle durée est réalisable en mode 8 bits.
3. Quelle est la durée exacte générée ?

4.2 Timer 0 en mode 16 bits

4. Ecrire un programme qui fait changer l'état de la LED connectée à RD0 toutes les 500 ms, le quartz étant à 40 MHz. Une telle durée n'étant pas réalisable en mode 8 bits, on vérifiera préalablement qu'elle l'est en mode 16 bits.
5. Quelle est la durée exacte générée ?