Les Timers

INTRODUCTION

Les Timers sont des périphériques de gestion de temps. Ils sont probablement, les périphériques les plus utilisés ; même les autres périphériques l'utilisent de façon directe ou indirecte. Ces périphériques permettent de réaliser les fonctions suivantes :

- Diviseur de fréquence
- Comptage des évènements
- Synchronisation des signaux
- Fixer le débit d'une liaison série synchrone et asynchrone
- Génération des événements périodiques (échantillonnage des signaux analogiques, rafraichissement des afficheurs multiplexés, régulation numérique …)
- Génération des signaux périodiques (carré, MLI …)
- Mesure de temps…

FONCTIONNEMENT DES TIMERS

Les TIMERS sont des compteurs formés généralement d'un **pré-diviseur** suivi d'un **registre compteur** de 8 ou 16 bits. L'entrée d'horloge peut être interne (*mode timer*) ou externe (*mode compteur d'événements*). Lorsque le registre compteur atteint sa valeur maximale et repasse à 0, un bit indicateur de débordement (*Flag*) sera positionné et une interruption pourra être générée, informant ainsi la CPU du débordement du TIMER. Il faut bien noter que vous devrez remettre à zéro cet indicateur après chaque débordement.

Le microcontrôleur PIC18F4520 dispose de quatre Timers appelés TIMER0, TIMER1, TIMER2 et TIMER3.

TIMER 0

Fonctionnement interne du TIMER 0

Le TIMER 0 est implémenté dans toutes les versions des microcontrôleurs de Microchip, son ancienne appellation était **RTC** (**R**eal **T**ime **C**lock). Il est formé d'un pré-diviseur programmable 8 bits (Programmable Prescaler) suivi d'un registre compteur 8 bits (**TMR0**), étendue pour les PIC18 à 16bits. Ce TIMER travaille en deux modes : le mode 8 bits (mode par défaut, pour assurer la compatibilité avec les anciennes versions) et le mode 16 bits. Le bit **T08BIT** de registre T0CON permet de basculer d'un mode à l'autre.

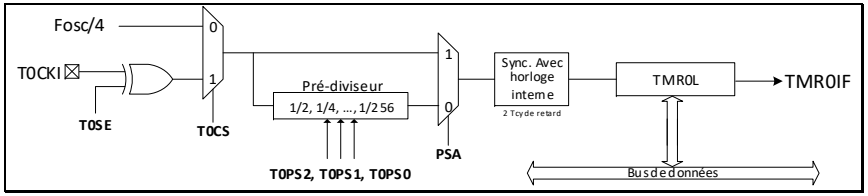

Figure 21 : TIMER 0 en mode 8 bits

Le bit **T0CS** permet de choisir l'horloge, interne (Fosc/4) ou externe RA4/T0CKI. Dans le cas d'une horloge externe l'incrémentation du TIMER0 peut se faire soit sur front montant ou descendant suivant la valeur du bit **T0SE**. Le bit **PSA** permet l'insertion du prédivisieur dans la chaîne de comptage ou non (pas de prédivision). Les bits **T0PS2**, **T0PS1** et **T0PS0** sont utilisés pour fixer la valeur de prédivision à 2,4, 8, 16, 32, 64, 128 ou 256.

Quand le contenu du registre **TMR0** passe de FF à 00 (ou de FFFF à 0000 en mode 16 bits) le bit indicateur **TMR0IF** du registre I**NTCON** passe à 1 pour signaler un débordement, une interruption pourra être générée, si le bit **TMR0IE** est défini.

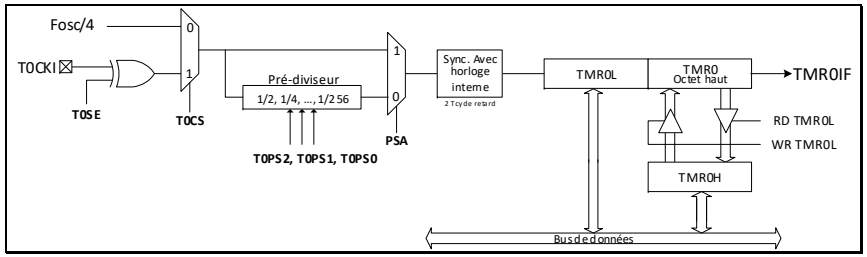

Figure 22 : TIMER 0 en mode 16 bits

En mode 16 bits, le registre **TMR0H** ne présente pas réellement l'octet haut du **TMR0**, il s'agit d'un buffer dans lequel sera transféré automatiquement l'octet haut du TMR0 au moment de la lecture du **TMR0L**. De même, l'écriture dans **TMR0L** charge automatiquement le contenu du registre **TMR0H** dans l'octet haut du registre **TMR0**. Etant donné que le PIC18 est un microcontrôleur 8 bits, il ne peut lire que huit

bits à la fois. En raison de cette fonctionnalité, la lecture ou l'écriture dans l'octet haut du **TMR0** est synchronisée avec **TMR0L**.

Initialisation du registre TMR0 par le contenu de la variable `Init_TMR0`

```
TMR0H = Init_TMR0 >> 8 ;        // Ecriture sur TMR0H en premier lieu
TMR0L = Init_TMR0 ;             // puis écriture sur TMR0L (le
                        // compilateur fait la troncature)
```

Lecture du registre TMR0

```
Count0 = TMR0L;                 // lire le contenu du TMR0L
Count0 | = TMR0H << 8 ;         // ensuite celui du TMR0H
```

Le compilateur MPLAB XC8 a défini une variable **`TMR0`** de 16 bits qui facilite la manipulation de ces registres :

```
TMR0 = Init_TMR0 ;
Count0 = TMR0 ;
```

Tous les bits de configuration du TIMER0 sont implémentés dans le registre **T0CON**.

Registre T0CON

b7	b6	b5	b4	b3	b2	b1	b0
R/W-1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1	R/W -1
TMR0ON	**T08BIT**	**T0CS**	**T0SE**	**PSA**	**PS2**	**PS1**	**PS0**

Symbole	Fonction
TMR0ON	**Timer0 On/Off Control bit**. TMR0ON = 1 : validation du Timer 0. TMR0ON = 0 : Timer 0 stoppé
T08BIT	**Timer0 8-Bit/16-Bit Control bit** T08BIT = 1 : mode 8 bits (par défaut). T08BIT = 0 : mode 16 bits.
T0CS	**TMR0 Clock Source Select bit** T0CS = 1 : fonctionnement en mode compteur (horloge externe) T0CS = 0 : fonctionnement en mode Timer (horloge interne)
T0SE	**TMR0 Source Edge Select bit.** T0SE = 1 : incrémentation sur transition positive de l'horloge externe T0SE = 0 : incrémentation sur transition négative de l'horloge externe
PSA	**Prescaler Assignment bit.** PSA = 1 : Prédiviseur non assigné PSA = 0 : Prédiviseur assigné au TIMER0
PS2 **PS1** **PS0**	**Prescaler Rate Select bits**.

PS2	PS1	PS0	Prédiv Timer0
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

Période de débordement

La période de débordement $\boldsymbol{T_d}$ est l'intervalle de temps entre deux débordements successifs (Figure 23). $\boldsymbol{T_d}$ est le produit de la période d'horloge $\boldsymbol{T_h}$ par le nombre d'impulsions comptés par le TIMER.

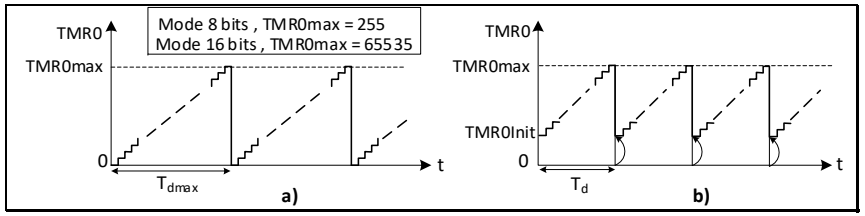

Figure 23

Le TIMER 0 déborde toujours lorsque le registre TMR0 atteint sa valeur maximale et repasse à zéro. La valeur maximale $TMR0max = 255$ en mode 8 bits et $TMR0max = 65535$ en mode 16 bits. Il est possible que le registre TMR0 commence le comptage par une valeur initiale **TMR0Init** non nulle (Figure 23 b)). L'expression de la période de débordement est donnée par l'expression suivante :

$\boldsymbol{T_d = T_h \cdot Prediv \cdot (TMR0max + 1 - TMR0\,Init)}$ avec $TMR0max = 2^N - 1$ où $N = 8$ ou 16.

Exemple : Fonction delay avec le TIMER 0

La fonction *delay* est l'une des applications la plus simple et la plus utilisée avec les TIMERS. Les fonctions implémentées avec les TIMERS sont plus précises que les fonctions fournis par le compilateur (__delay_us() et __delay_ms(), où la CPU compte le nombre de cycle d'instructions). Deux techniques sont disponibles pour gérer le TIMER :

- Interroger l'indicateur de débordement.
- Utiliser le TIMER en mode interruption.

Nous vous proposons de programmer une fonction *delay* de 500ms avec un microcontrôleur PIC18F4520 cadencé à 8 MHz. La fonction *delay_500ms* est utilisée pour fixer le temps de clignotement d'une LED connectée à la broche RC1.

Méthode interrogation (scrutation)

Nous utilisons le mode TIMER (horloge interne) :

$F_{osc} = 8MHz \Rightarrow T_h = 4 \cdot T_{osc} = 0{,}5\mu s$. Cherchons la valeur initiale $TMR0Init$ pour une période de débordement de $500ms$.

$T_d = T_h \cdot Prediv \cdot (TMR0max + 1 - TMR0\,Init)$

Soit $K = \frac{T_d}{T_h} = Prediv \cdot (TMR0max + 1 - TMR0\,Init)$

$$K = \frac{T_d}{T_h} = \frac{500000}{0.5} = 1000000$$

$K = Prediv \cdot (TMR0max + 1 - TMR0\,Init)$, nous avons une seule équation, contenant deux inconnus. Il suffit dans ce cas, de fixer l'un des paramètres et de chercher l'autre. $Prediv$ n'a que huit valeurs ; nous devons à chaque fois fixer une valeur de $Prediv$ et chercher $TMR0Init$.

$$\boldsymbol{TMR0Init = TMR0max + 1 - \frac{K}{Prediv}}$$

avec $0 \leq TMR0Init \leq TMR0max$

Utilisation du mode 16 bits

$$TMR0max + 1 = 65536 \;\; d'où \;\; TMR0Init = 65536 - \frac{1000000}{Prediv}$$

Vous remarquez que pour $Prediv < 16 \;\Rightarrow\; TMR0Init < 0$ (Cette valeur est inacceptable)

Pour $Prediv = 16 \;\Rightarrow\; TMR0Init = 3036$ (valeur acceptable)

Pour $Prediv = 32 \;\Rightarrow\; TMR0Init = 34286$ (valeur acceptable)

Pour $Prediv = 64 \;\Rightarrow\; TMR0Init = 49911$ (valeur acceptable)

Pour $Prediv = 128 \;\Rightarrow\; TMR0Init = 57723{,}5$ (valeur acceptable, mais imprécise)

Pour $Prediv = 256 \;\Rightarrow\; TMR0Init = 61629{,}75$ (valeur acceptable, mais imprécise)

A partir $Prediv = 128$, la résolution se dégrade, puisque $TMR0Init$ doit être une valeur entière.

Nous allons prendre $Prediv = 16 \;et\; TMR0Init = 3036$.

Cherchons le mot de commande à charger dans le registre T0CON :

Mode 16 bits, horloge interne, utilisation du prédiviseur avec la valeur 16

TMR0ON	T08BIT	T0CS	T0SE	PSA	PS2	PS1	PS0
1	0	0	0	0	0	1	1

T0CON = 0b10000011 = 0x83

```
/****************************************************************/
/* Programme delay_500ms     - TIMR0 en mode 16 bits -          */
/****************************************************************/
#include <xc.h>
#define _XTAL_FREQ 8000000
#pragma config OSC = HS, WDT = OFF, LVP = OFF
#define LED PORTCbits.RC1
#define TMR0Init 3036;
void delay__500ms(void)
{       INTCONbits.TMR0IF = 0;
        TMR0 = TMR0Init;
        while(INTCONbits.TMR0IF == 0) continue;
}
void main(void) {
    TRISCbits.TRISC1 = 0;               // RC1 en sortie
    T0CON = 0x83;                       // Mot de commande
    while(1)
    {
        delay__500ms();                 // Attente 500ms
        LED = !LED;                     // changement d'état de la LED
    }
}
```

Utilisation du mode 8 bits

$$TMR0max + 1 = 256 \;\; d'où \;\; TMR0Init = 256 - \frac{1000000}{Prediv}$$

Même pour $Prediv = 256 \;\Rightarrow\; TMR0Init < 0$, cela veut dire que le TIMER0 en mode huit bits n'a pas la capacité de compter 1000000 impulsions.

Cherchons alors la période de débordement maximale en mode 8 bits :

$$T_{dmax} = T_h \cdot Prediv_{max} \cdot (TMR0max + 1)$$
$$= 0{,}5 \times 256 \times 256 = 32{,}768ms$$

La période de débordement ne dépasse pas $32{,}768ms$. Il est possible dans ce cas, de programmer une fonction delay de 500ms en comptant le nombre de débordement.

$T = cnt \cdot T_d$, avec $T = 500ms$

Il faut choisir T_d de sorte que la valeur de $cnt = \frac{T}{T_d}$ soit entière ; prenons $T_d = 25ms \Rightarrow cnt = 20$.

Cherchons maintenant les valeurs de $Prediv$ et $TMR0Init$ pour $T_d = 25ms$

$$K = \frac{T_d}{T_h} = \frac{25000}{0{,}5} = 50000$$

La seule valeur acceptable pour $Prediv = 256 \Rightarrow TMR0Init = 256 - \frac{50000}{256} = 60{,}6875$. Nous devons prendre $TMR0Init = 60$

Cherchons le mot de commande à charger dans le registre T0CON

TMR0ON	T08BIT	T0CS	T0SE	PSA	PS2	PS1	PS0
1	1	0	0	0	1	1	1

T0CON = 0b11000111 = 0xC7

```
/*****************************************************************/
/* Programme delay_500ms     - TIMR0 en mode Compatible -        */
/*****************************************************************/
#include <xc.h>
#define _XTAL_FREQ 8000000
#pragma config OSC = HS, WDT = OFF, LVP = OFF
#define LED PORTCbits.RC1
#define TMR0Init  60
void delay__500ms(void)
{ int cnt;
    for(cnt = 0; cnt < 20; cnt++){
        INTCONbits.TMR0IF = 0;
        TMR0L = TMR0Init;
        while(INTCONbits.TMR0IF == 0) continue;
    }
}
void main(void) {
    TRISCbits.TRISC1 = 0;       // RC1 en sortie
    T0CON = 0xC7;               // mot de commande
    while(1) {
        delay__500ms();         // Attente 500ms
        LED = !LED;             // changement d'état de la LED
    }
}
```


Méthode interruption

La méthode interruption est plus simple de mettre en œuvre.

```
/*****************************************************************/
/* Programme delay_500ms     - Méthode Interruption -            */
/*****************************************************************/
#include <xc.h>
#define _XTAL_FREQ 8000000
#pragma config OSC = HS, WDT = OFF, LVP = OFF
#define TMR0Init 60
#define LED PORTCbits.RC1
int cnt = 20;
void __interrupt(high_priority) isr_TIMER0(void)
{   TMR0L = TMR0Init;           // Réinitialiser TMR0
    INTCONbits.TMR0IF = 0;      // remettre le flag à 0
    cnt--;                      // décrémentation du Compteur
    if(cnt == 0){               // compte épuisé?
       cnt = 20;
       LED = !LED;              // changer l'état de la LED
    }
}
void main(void) {
    TRISCbits.TRISC1 = 0;       // RC1 en sortie
    T0CON = 0xC7;               // mot de commande
    TMR0 = TMR0Init;            // Initialiser TMR0
    RCONbits.IPEN = 1;          // activer la logique de priorité
    INTCONbits.TMR0IF = 0;      // remettre le flag à 0
    INTCONbits.TMR0IE = 1;      // Validation de l'interruption TIMER0
    INTCON2bits.TMR0IP = 1;     // Priorité haute
    INTCONbits.GIEH = 1;        // Validation globale
    while(1) continue;
}
```

TIMER 1

Fonctionnement du TIMER1

Le TIMER1 est un Timer formé d'un prédiviseur et d'un registre compteur 16 bits appelé TMR1 (TMR1H : TMR1L). Le passage du TMR1 de 0xFFFF à 0x0000, positionne le bit indicateur de débordement **TMR1IF** (**PIR1<0>**). Dans ce cas une interruption pourra être générée, si le bit de validation **TMR1IE** (**PIE1<0>**) est défini. L'entrée d'horloge peut être interne (mode Timer) ou externe (mode compteur). Le choix d'horloge est sélectionné par le bit **TMR1CS**.

En mode compteur l'horloge peut être appliquée à l'entrée **RC0/T13CKI** ou à partir d'un oscillateur à quartz. Si l'oscillateur est activé (**T1OSCEN = 1**), les broches RC0/T1OSCO et RC1/T1OSCI deviennent des entrées, et les bits correspondants dans le registre TRISC n'auront aucun effet. L'oscillateur est un circuit à faible puissance conçu pour des Quartz de 32KHz. Normalement, un Quartz de 32,768 KHz est utilisé pour implémenter une horloge temps réel (RTC), qui peut fonctionner même en mode veille (sleep). De plus, cet oscillateur peut être spécifié comme horloge du processeur. Cette fonctionnalité est un moyen de mettre en œuvre un mode de fonctionnement à faible puissance, sans désactiver l'ensemble des périphériques. Le fonctionnement en mode asynchrone est une autre particularité du mode compteur.

En mode asynchrone TIMER1 continu à fonctionner même en mode veille ; toutefois, la lecture du registre TRM1 pose un problème de dépassement entre les lectures de TMR1H et TMR1L. Pour les opérations d'écriture, le TIMER peut être arrêté momentanément par le bit TMR1ON afin d'écrire les valeurs souhaitées. Cette conception héritée des PIC16, a été heureusement rectifié dans les PIC18. Lorsque le bit RD16 = 1, la lecture de deux registres est synchronisée de la même manière que celle du TIMER0. Par défaut RD16 = 0 ; vous devez le mettre à un "1", pour garantir une lecture valide.

Outre les fonctionnalités précitées, le TIMER1 est utilisé par le module CCP (Capture, Compare PWM).

Tous les bits de configuration, associés au TIMER 1 sont situés dans le registre **T1CON.**

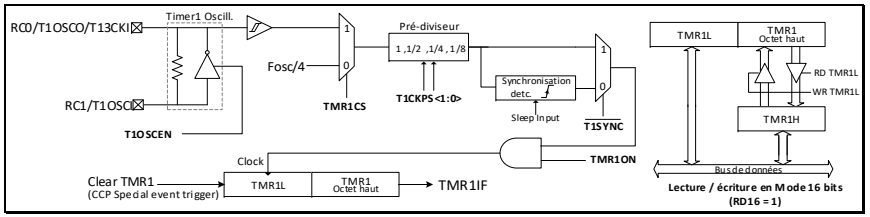

Figure 24

La formule du calcul de la période de débordement est similaire à celle du TIMER0 :

$$\boldsymbol{T_d = T_h \cdot Prediv \cdot (TMR1max + 1 - TMR1\,Init)} \text{ avec } TMR1max = 2^{16} - 1$$

Registre T1CON

b7 R/W-0	b6 R/W -0	b5 R/W -0	b4 R/W -0	b3 R/W -0	b2 R/W -0	b1 R/W -0	b0 R/W -0
RD16	**T1RUN**	**T1CKPS1**	**T1CKPS0**	**T1OSCEN**	$\overline{T1SYNC}$	**TMR1CS**	**TMR1ON**

Symbole	Fonction
RD16	**16-Bit Read/Write Mode Enable bit.** RD16 = 1 : valide la lecture ou l'écriture du timer en mode 16 bits RD16 = 0 : valide la lecture ou l'écriture du timer en double octets
T1RUN	**Timer1 System Clock Status bit.** T1RUN = 1 : l'horloge du microcontrôleur est dérivée de l'oscillateur du TIMER1 T1RUN = 0 : l'horloge du microcontrôleur est fournie par une autre source.
T1CKPS1 **T1CKPS0**	**Timer1 Input Clock Prescale Select bits** T1CKPS1 | T1CKPS0 | Prédivision 0 | 0 | 1 0 | 1 | 2 1 | 0 | 4 1 | 1 | 8
T1OSCEN	**Timer1 Oscillator Enable Control bit** T1OSCEN = 1 : oscillateur autorisé ; T1OSCEN = 0 : oscillateur désactivé
$\overline{T1SYNC}$	**Timer1 External Clock Input Synchronization Control bit** $\overline{TISYNC} = 1$: Pas de synchronisation de l'horloge externe $\overline{TISYNC} = 0$: Synchronisation de l'horloge externe
TMR1CS	**Timer1 Clock Source Select bit** TMR1CS = 1 : horloge externe ; TMR1CS = 0 : horloge interne
TMR1ON	**Timer1 On bit** TMR1ON = 1 : comptage en progression TMR1ON = 0 : comptage bloqué

TIMER 2

Fonctionnement du TIMER2

Le TIMER 2 comporte un registre compteur 8 bits (**TMR2**) et un registre période de 8 bits (**PR2**). Il dispose également d'un pré-diviseur et d'un post-diviseur. Ce

TIMER admet uniquement une horloge interne ($F_{osc}/4$). Le pré-diviseur peut être paramétré par l'une de trois valeurs : 1, 4 ou 16, tandis que le post-diviseur permet des divisions de 1 à 16 : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 ou 16.

A la mise sous tension (Power On Reset), le registre TMR2 est remis à zéro, tandis que le registre PR2 est initialisé par 0xFF. Le pré-diviseur est le post-diviseur sont remis à zéro suite à l'un des évènements suivants :

- Ecriture dans le registre TRM2
- Ecriture dans le registre T2CON
- Lors de réinitialisation du microcontrôleur (Power-on Reset, MCLR Reset, Watchdog Timer Reset ou Brown-out Reset)

Le principe de fonctionnement du Timer 2 est différent à ces précédents. La valeur de **TMR2** est comparée à celle du registre **PR2** à chaque cycle d'horloge. En cas d'égalité, le comparateur génère un signal d'égalité (TMR2 output), remet à zéro le registre TMR2 et incrémente le post-diviseur le prochain cycle d'horloge.

La période totale de débordement est donnée par la formule suivante :

$$\boldsymbol{T_d = 4 \times T_{osc} \times Pré\,div \times (PR2 + 1) \times Postdiv}$$

Chaque débordement entraine le positionnement de l'indicateur **TMR2IF (PIR1<1>)** et pourra générer une interruption si le bit **TMR2IE (PIE1<1>)** est défini.

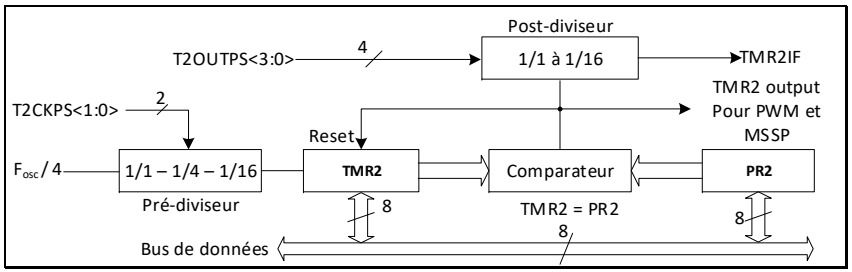

Figure 25

Le TIMER 2 est contrôlé par le registre **T2CON**. La mise à zéro du bit de contrôle TMR2ON (T2CON <2>) permet d'arrêter le TIMER pour réduire la consommation.

Registre T2CON

b7	b6	b5	b4	b3	b2	b1	b0
U-0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0	R/W -0
-	TOUTPS3	TOUTPS 2	TOUTPS 1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

<table>
<tr><th>Symbole</th><th>Fonction</th></tr>
<tr><td>TOUTPS3
TOUTPS2
TOUTPS1
TOUTPS0</td><td>Timer2 Output Postscale Select bits.
<table>
<tr><th>TOUTPS3</th><th>TOUTPS2</th><th>TOUTPS1</th><th>TOUTPS0</th><th>Post-division</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>3</td></tr>
<tr><td colspan="4">:
:</td><td>:
:</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>16</td></tr>
</table></td></tr>
<tr><td>TMR2ON</td><td>Timer2 On bit.
TMR2ON = 1 : Timer 2 ON (mise en service)
TMR2ON = 0 : Timer 2 OFF (mise hors service)</td></tr>
<tr><td>T2CKPS1
T2CKPS0</td><td>Timer2 Clock Prescale Select bits.
<table>
<tr><th>T2CKPS1</th><th>T2CKPS0</th><th>Prédivision</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>4</td></tr>
<tr><td>1</td><td>x</td><td>16</td></tr>
</table></td></tr>
</table>