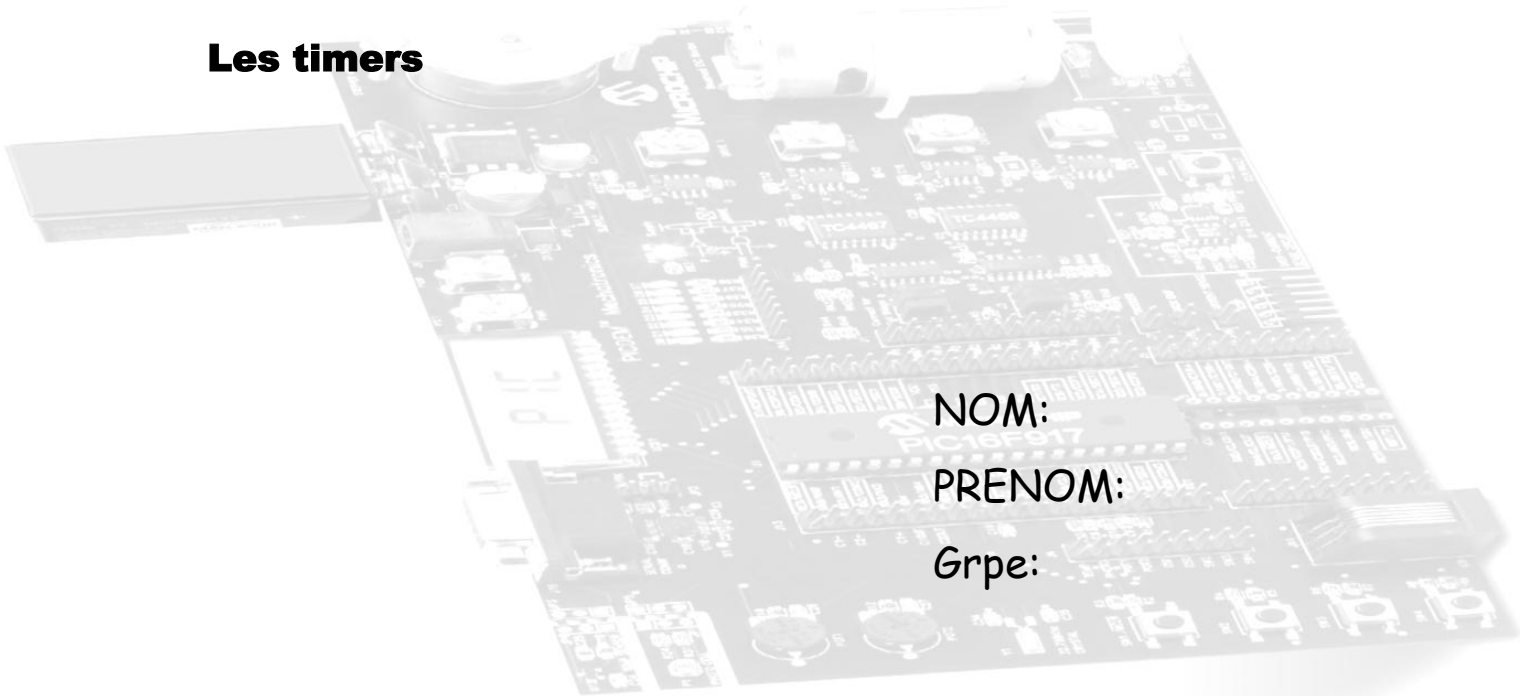
Cours de PIC

Les timers



NOM:

PRENOM:

Grpe:

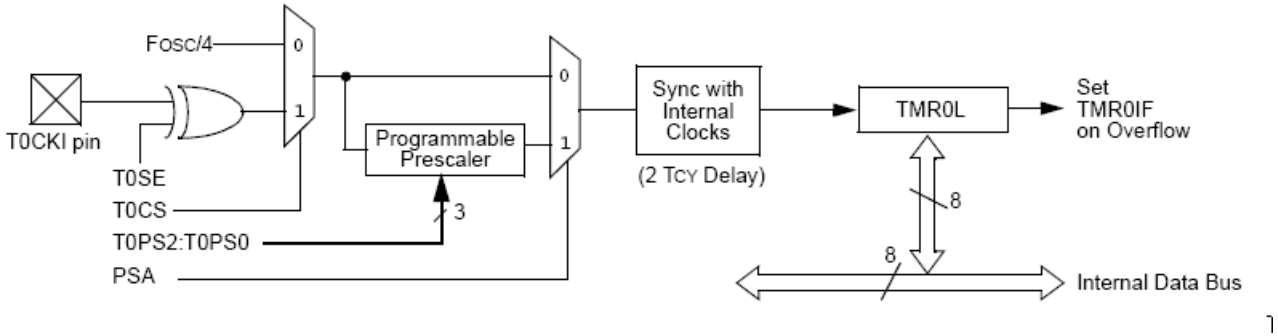
Timers: principes

Objectifs

- **Les Timers ou compteurs de temps écoulé permettent :**
  - ✓ **De réaliser des temporisations précises puisqu'ils utilisent des oscillateurs de référence à quartz.**
  - ✓ **De générer des signaux électriques 0V / 5V.**
  - ✓ **De mesurer des durées.**
  - ✓ **De compter des évènements extérieurs sur des fronts montants ou descendants**

Principe de fonctionnement

- Un timer est un compteur 8 ou 16 bits préchargeable dont l'horloge Th peut être
  - ✓ Soit dérivée de l'horloge principale du PIC (Tosc ou Kp.Tosc si on utilise un prédiviseur)
  - ✓ Soit fournie par un signal externe (broche RA4/T0CKI pour le timer0)
  - ✓ Soit fournie par un oscillateur interne supplémentaire



- Le passage de la valeur maximum (0xFF ou 0xFFFF) à 0 est appelé **overflow**.
  - ✓ L'overflow provoque le passage à 1 du bit TMRxIF.
  - ✓ Une interruption est disponible sur l'overflow.
- La durée entre deux overflow est :
  - ✓ Toverflow = (2^n – valeur initiale). Th avec (n = 8 ou 16)
- On peut lire/écrire une valeur dans le compteur.
  - ✓ (Précautions nécessaires pour les compteurs 16bits)

Timers: principes

Les paramètres d'ajustements

- Pour obtenir un temps désiré on peut jouer sur :
  - ✓ L'horloge externe ou interne.
  - ✓ La pré-division.
  - ✓ Le nombre d'incrémentations pour un cycle, sachant que si on recharge le compteur à chaque fin de cycle avec une valeur N, le nombre d'incrémentations par cycle est de $(2^n - N)$.

Carte PICDEM + :calculer la pré-division nécessaire

- Durée de comptage maximale en utilisant l'oscillateur de la carte PICDEM +
  - ✓ Avec Fosc = 4 Mhz divisée par 4, soit 1Mhz, la période d'horloge compteur est de Thorloge = 1 ms.
  - ✓ Si on utilise la valeur maximum de pré-division soit 1/256, le compteur s'incrémente de 1 chaque Tincrémentation = 1µs x pré-division soit toutes les 256 µs.
  - ✓ **Avec le comptage maximum :**
    - **En mode 8 bits: 256 incrémentations pour aller de 0X00 à 0XFF, on obtient : 256 incré. x 256 us soit 65,536 ms.**
    - **En mode 16 bits: 16,77 s**
- Les étapes avec un exemple: temporisation de 10ms, quartz 4Mhz, timers en mode ???
  - ✓ On calcule les valeurs max de tempo sur 8 bits et 16 bits**: ici 10ms<65,536 ms => mode 8 bits**
  - ✓ **La formule donne: (256 - N) . 4Tosc . Pré-Div de TMR0L = 10ms**
  - ✓ **Etape1**: poser N=0 dans un premier temps
    - Pré-division = (10ms / 1µs) / 256 = 39
    - On choisit la valeur existante du pré-diviseur immédiatement supérieure soit : pré-division = 64
  - ✓ Etape2: on ajuste N pour obtenir la bonne tempo
    - Le nombre d'incrémentations par cycle est égal à : (256 – N) = (10ms / 1µs) / 64 = 156 soit : N = 256-156 =100

Etapes de programmation des timers

- Des IT?
- Initialisation du timer
- Lancement du timer
- Acquittement du flag overflow
- Rechargement du timer

Les étapes de programmation

Initialisation du timer 0

- Les réglages fonctions des calculs

```
void initTimer(void)
{
// init T0 50ms à 4Mhz sans it avec prechargement
T0CONbits.TMR0ON = 0; // Timer O off
T0CONbits.T08BIT = 1; // mode 8 bits
T0CONbits.T0CS = 0; // source HI interne
T0CONbits.PSA = 0; // autorisation des prédivisions
T0CONbits.T0PS0 = 1; //
T0CONbits.T0PS1 = 1; // prédivision par 256
T0CONbits.T0PS2 = 1; //
TMR0L = 0x3D;
}
```

- Ne pas oublier de préciser **si** interruptions
  - ✓ Choix des niveaux de priorité
    - Exemple si pas de priorité: RCONbits.IPEN=0
  - ✓ Validation IT globale
    - INTCONbits.GIE=1 (=1 si IT autorisée, =0 sinon)
    - INTCONbits.PEIE=0 (**ou 1 si timer ½ ou 3**)
  - ✓ Validation IT périphérique
    - INTCONbits.TMR0IE=1

Lancer le timer 0(ou l'arrêter)

- T0CONbits.TMR0ON = 1;

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

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

Les étapes de programmation

- **Tester la fin du comptage en scrutation**
  - Bit d'overflow à tester: INTCONbits.TMR0IF
  - INTERDIT de faire un polling direct : Tant que (INTCONbits.TMR0IF = 0) ;
  - Faire un test non bloquant comme suit par exemple

```
init();
for(;;)
{
        if(INTCONbits.TMR0IF==1)// drapeau timer  fini
        {
                LED=!LED; // je commute la LED
                INTCONbits.TMR0IF=0; // je baisse le drapeau
                TMR0L = 0x3D; // prechargement ? 50ms
        }
}
```

- **Acquitter le bit overflow**
  - INTCONbits.TMR0IF=0
- **Relancer le timer**
  - TMR0L = ……;

- **Utilisations des fonctions de la librairie C18**

```
// INIT
OpenTimer0( TIMER_INT_OFF &
T0_8BIT &
T0_SOURCE_INT &
T0_PS_1_32 );
// Chargement à 0 ici
WriteTimer0( 0 );
// lecture valeur
result = ReadTimer0();
```