

LAB4

Le but du LAB est comment programmer une temporisation à base du SOFT et comment faire aussi avec du HARD TIMER0.

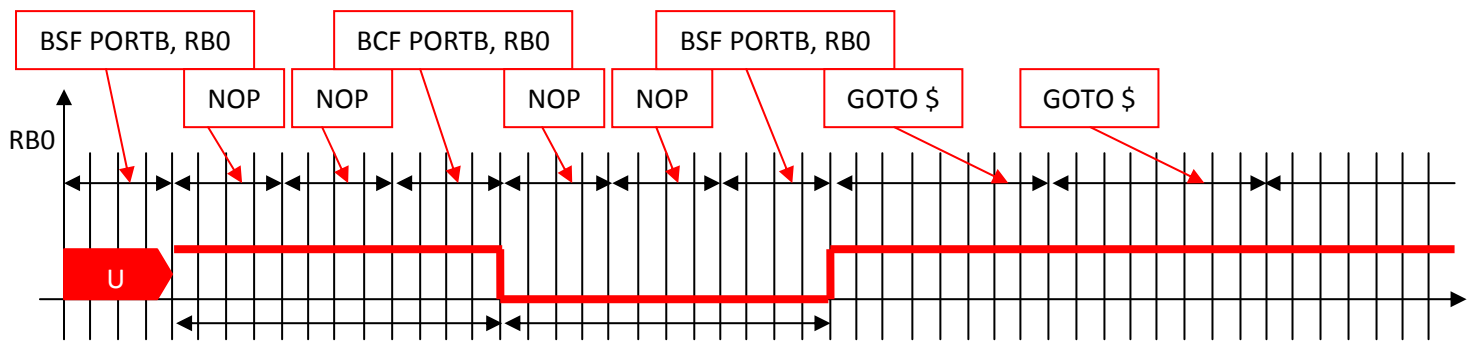
Avant de commencer la solution du LAB, il faut faire un petit tour sur le PORT d'un μ C.

Le PORT d'un μ C est un ensemble de bascules (registre) garde son valeur dès qu'il n'y a pas d'une autre mise à jour.

Exemple :

```
BSF PORTB, RB0 ; RB0 ← 1
NOP
NOP
BCF PORTB, RB0 ; RB0 ← 0
NOP
NOP
BSF PORTB, RB0 ; RB0 ← 1
GOTO $
```

Dessiner le signal qu'on va voir sur la patte RB0.



Que remarquez-vous ?

On remarque que la patte RB0 change son état dès qu'il y a une mise à jour par BSF/BCF au moment de la phase Q4 de ses dernières.

Si vous avez compris, essayer de faire un code en assembleur qui génère un signal carré de période 10 cycles sur la patte RB1.

La solution :

Un signal carré périodique se répète chaque 10 cycle s'implique que son niveau haut sera sur 5 cycles et son niveau bas aussi.

```
AGAIN:    BSF  PORTB, RB1
          NOP
          NOP
          NOP
          NOP
          BCF  PORTB, RB1
          NOP
          NOP
          GOTO AGAIN
```

Que remarquez-vous ?

On remarque que le nombre de NOP dans le niveau haut différent que le nombre de NOP de niveau bas.

- Nombre de NOP de niveau haut = 4 (nombre de cycles de niveau haut -1).
- Nombre de NOP de niveau bas = 2 (nombre de cycles de niveau bas -3 [à cause de GOTO])

Et on remarque aussi que le code se compose par trois actions principales :

- La mise à jour de la patte par BCF/BSF,
- Le délai par les NOP,
- Le saut par GOTO pour répéter.

Questions : si la période est grande et nécessite un nombre de NOP très grand, est ce que c'est pratique d'utiliser une suite de NOP comme l'exemple ? Est ce qu'il y a une autre solution que la suite de NOP ?

Absolument, oui !

Exemple :

Essayer de faire un code en assembleur qui génère un signal carré de période 100 cycles sur la patte RA0.

Donc, 50 cycles pour le niveau haut et 50 cycles pour le niveau bas.

Nombre de NOP de niveau haut : $50-1 = 49$ cycles.

Nombre de NOP de niveau bas : $50-3 = 47$ cycles.

L'issu qu'on va prendre, c'est d'écrire un sous-programme épuise le nombre de cycles nécessaire pour créer notre temporisation.

Le code :

```
AGAIN:    BSF  PORTA, RA0
          CALL DELAY_49
          BCF  PORTA, RA0
          CALL DELAY_47
          GOTO AGAIN
```

Le code de sous-programme DELAY_49 épuise 49 cycles et le sous-programme DELAY_47 épuise 47 cycles. Le sous-programme lui-même au moment d'appel par CALL épuise 2 cycles et au moment de retour par RETURN épuise 2 cycles, donc le code principal du sous-programme = le nombre de cycles nécessaire – 4.

Avant d'aller vers le code des sous-programmes, soit le code suivant :

```
MOVLW 0x03
MOVWF CNT
DECFSZ CNT, F
GOTO $-1
NOP
```

Compter le nombre de cycles passé par ce code.

Si on note l'instruction par 1 jusqu'à l'instruction GOTO par 4 et NOP par 5, on va voir l'exécution suivante :

CNT	XX	3	2	2	1	1	0	0
INST	1	2	3	4	3	4	3	5
# cyc	1 cyc	1cyc	1 cyc	2 cyc	1 cyc	2 cyc	2 cyc	1 cyc
Total								11 cyc

On remarque que les instructions 3-4 se répètent 2 fois avec la combinaison 3-5 fait le même nombre de cycles, donc, on peut trouver une relation entre la valeur initiale dans CNT avec le nombre de cycles passé par code au-dessus

Nombre de cycles = 2 + CNT_INITIALE*3

Avec 3 présente le temps passé par les instructions [3-4, 3-5] et 2 présente le temps passé avant la séquence 3-4.

Voici un autre code :

```
MOVLW 0x03
MOVWF CNT
NOP
DECFSZ CNT, F
GOTO $-2
NOP
```

Distinguer le nombre de cycles passé par ce code.

Nombre de cycles = 2 + CNT_INITIALE*4

A vous mes chers étudiants, faire le code de DELAY_49 et DELAY47.

Le code de DELAY_49 :

D'après l'étude qu'on a fait avant, le nombre de cycles qu'il faut passer par le code principal de sous-programme est $49 - 4 = 45$ cycles. Il reste maintenant de déterminer la valeur de CNT_INITIALE qui assure un temps de 45 cycles.

Si on utilise la première formule, on trouve :
 $2 + V \cdot 3 = 45 \rightarrow V = 43/3$ (Q=14 et R=1)

Si on utilise la deuxième formule, on trouve :
 $2 + V \cdot 4 = 45 \rightarrow V = 43/4$ (Q=10 et R=3)

Dans ce cas, on préfère d'utiliser la première formule seulement à cause du reste (R=1).

Le code par la formule 1
<pre>DEALY_49:MOVLW D'14' MOVWF CNT DECFSZ CNT, F GOTO \$-1 NOP NOP ; 1cyc RETURN</pre>
Optimisation
<pre>DEALY_49:MOVLW D'14' MOVWF CNT DECFSZ CNT, F GOTO \$-1 GOTO \$+1 ; 2cyc RETURN</pre>

Le même travail avec DELAY_47.

Nombre de cycles du code principal = $47 - 4 = 43$.

Si on utilise la première formule, on trouve :

$$2 + V \cdot 3 = 43 \rightarrow V = 41/3 \text{ (Q=13 et R=2)}$$

Si on utilise la deuxième formule, on trouve :

$$2 + V \cdot 4 = 43 \rightarrow V = 41/4 \text{ (Q=10 et R=1)}$$

Dans ce cas, on préfère d'utiliser la deuxième formule seulement à cause du reste (R=1).

Le code par la formule 2
<pre>DEALY_47:MOVLW D'10' MOVWF CNT NOP DECFSZ CNT, F GOTO \$-2 NOP NOP ; 1cyc RETURN</pre>
Optimisation
<pre>DEALY_47:MOVLW D'10' MOVWF CNT NOP DECFSZ CNT, F GOTO \$-2 GOTO \$+1 ; 2cyc RETURN</pre>

Question: quel est le nombre de cycles maximal qu'on peut réaliser par les formules 1 et 2 ?

Formule1 : $2 + 256 \times 3 = 770$ cycle

Formule2 : $2 + 256 \times 4 = 1026$ cycle

La 256 est une valeur virtuelle par ce qu'elle dépasse les 8bit de la variable CNT ; réellement, la 256 présente la valeur 0.

Maintenant, on revient à notre LAB.

Il demande de clignoter une LED reliée à RB0 par une fréquence de 2.5Hz si le Q = 4MHz.

Clignoter une LED, c'est de créer un signal carré ; dans notre cas de période = $1/f = 1/2.5 = 0.4$ secondes = 400 ms (niveau haut durant 200ms et niveau bas durant 200ms)

L'étude qu'on a réalisée précédemment, elle utilise la notion de nombre de **cycles** et ici en termes du **temps**; donc, **comment faire ?**

Tout simplement, essayer de convertir le temps en cycles par :

Le cycle de notre $\mu C = 4/Q = 4/4\text{MHz} = 1\mu s$

Donc : Nombre de cycles pour réaliser 200ms : $200\text{ms}/1\mu s = 200000$ cycles.

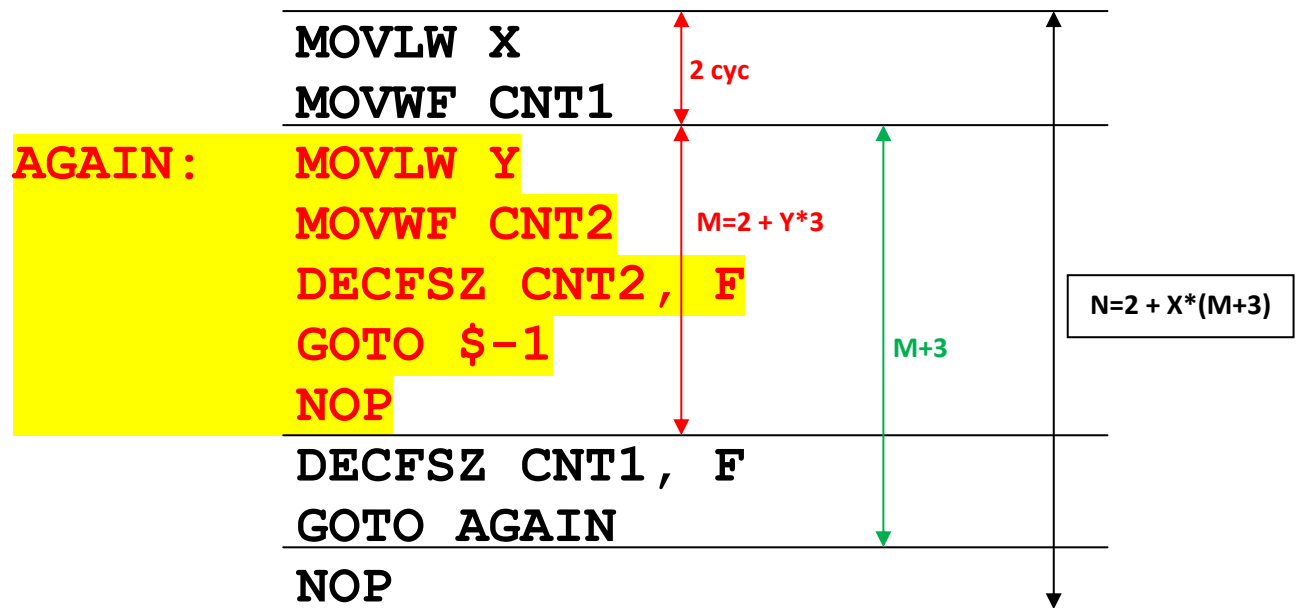
On utilise le même code de l'étude :

```
AGAIN:    BSF  PORTB, RB0
          CALL DELAY_199999
          BCF  PORTB, RB0
          CALL DELAY_199997
          GOTO AGAIN
```

Le nombre de cycles passé par les formules 1 et 2 par rapport au nombre de cycles qu'il faut faire, reste plus loin, comment faire ?

Essayer mes chers étudiants de trouver le nombre de cycles passé par le code suivant en fonction de X et Y :

```
          MOVLW X
          MOVWF CNT1
AGAIN:    MOVLW Y
          MOVWF CNT2
          DECFSZ CNT2, F
          GOTO $-1
          NOP
          DECFSZ CNT1, F
          GOTO AGAIN
          NOP
```



$$N(X, Y) = 2 + X * (2 + Y * 3 + 3) = 2 + X * (5 + Y * 3) \text{ cycle}$$

$$N_{\text{MAX}}(X=256, Y=256) = 197890 \text{ cycle}$$

La même question pour le code suivant:

```
        MOVLW X
        MOVWF CNT1
AGAIN1: MOVLW Y
        MOVWF CNT2
AGAIN2: MOVLW Z
        MOVWF CNT3
        DECFSZ CNT3, F
        GOTO $-1
        NOP
        DECFSZ CNT2, F
        GOTO AGAIN2
        NOP
        DECFSZ CNT1, F
        GOTO AGAIN1
        NOP
```

$$N(X, Y, Z) = 2 + X*(M(Y, Z) + 3)$$

$$M(Y, Z) = 2 + Y*(5 + Z*3)$$

S'implique:

$$N(X, Y, Z) = 2 + X*(5 + Y*(5 + Z*3))$$

$$N_{\text{MAX}}(X=256, Y=256, Z=256) = 50.660.610 \text{ cycle}$$

Donc, il reste de chercher les bonnes valeurs de X, Y et Z qui assurent notre temporisation !!!

La temporisation DELAY_199999 :

Nombre de cycles du code principal = $199999 - 4 = 199995$ cycles.

$$199995 \approx 2 + X*(5 + Y*(5 + Z*3))$$

$$199993 \approx X*(5 + Y*(5 + Z*3))$$

On suppose que $X = 3$, s'implique que :

$$66664 \approx 5 + Y*(5 + Z*3)$$

$$66659 \approx Y*(5 + Z*3)$$

On suppose que $Y = 100$, s'implique que :

$$666 = 5 + Z*3$$

$$661 = Z*3, \text{ s'implique que } Z = 661/3 \approx 220$$

Donc : $N(X=3, Y=100, Z=220) = 199517$ cycle

Il reste = $199995 - 199517 = 478$ cycle

Pour le reste (478), on peut le faire par les formules 1 ou 2.

Formule 1 : $2 + V*3 = 478, V=476/3(Q=158, R=2)$.

Formule 2 : $2 + V*4 = 478, V=476/4(Q=119, R=0)$.

Donc, on préfère la deuxième formule.

Le code de DELAY_199999 :

```
DELAY_199999 : MOVLW D' 3'
                MOVWF CNT1
AGAIN1:         MOVLW D' 100'
                MOVWF CNT2
AGAIN2:         MOVLW D' 220'
                MOVWF CNT3
                DECFSZ CNT3, F
                GOTO $-1
                NOP
                DECFSZ CNT2, F
                GOTO AGAIN2
                NOP
                DECFSZ CNT1, F
                GOTO AGAIN1
                NOP
                ;=====
                MOVLW D' 119'
                MOVWF CNT1
                NOP
                DECFSZ CNT1, F
                GOTO $-2
                NOP
                ;=====
                RETURN
```

A vous mes chers étudiants, faire le DELAY_199997

Il existe une autre solution optimale qui nécessite une seule temporisation :

Algorithme :

```
; Initialisation
RB0 ← 0;

; Clignotement de la LED
While (True)
    RB0 ← not(RB0);
    Call DELAY;
End While
```

Assembleur :

```
; Initialisation
                BCF PORTB, RB0
; Clignotement
                MOVLW B'00000001'
AGAIN:         XORWF PORTB, F
                CALL DELAY_199997
                GOTO AGAIN
```