Buot Adrien – Guicharnaud Léo – Sabatey Clara

Rapport Microprocesseurs :

Rapport TP I :

1) On s'attend à trouver en résultat le PORTC qui passe d'une valeur 0 à 1 en continu. Pourtant en utilisant le debugger on se rend compte que la valeur ne change pas et reste à 0. Cela est dû au fait que nous n'avons pas initialisé le port C.

2) On attend en résultat une LED qui s'allume et qui s'éteint en continu. Cependant, rien ne s'allume. L'explication est que nous n'utilisons pas de Delay ainsi la LED s'allume et s'éteint beaucoup trop vite pour que le changement soit perçu par l'œil humain.

3) Pour pouvoir distinguer les 2 états de la LED, il faut ralentir le programme. Pour ce faire, on va devoir créer une fonction Delay.

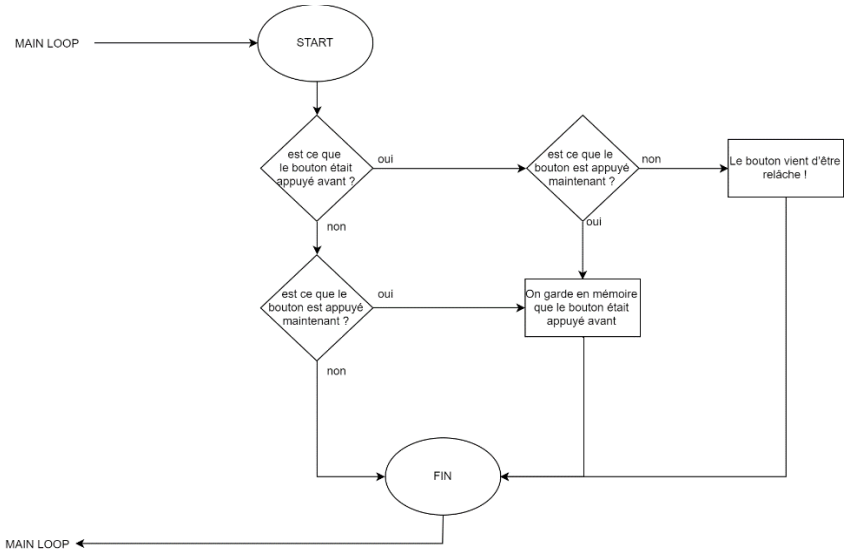
4) Le programme my_delay attend d'être sur un état 1 de count 1 et count 2 pour continuer le programme. Tant que count 1 et/ou count 2 ne sont pas à 1, la fonction my_delay continue de s'appeler. On compte 4 cycles.

5) LED 3 =1/2 fréquence LED 2= 1/4 fréquence de LED 1 = 1/8 fréquence de LED 0

6) On peut lire l'état de la pin RD0 en initialisant le port D comme une entrée digitale. On aura ensuite accès à l'état de PORTD il faut appliquer le masque 0x01 à PORTD, et comme ça, il est possible de savoir si le bouton est relâché ou non :
   - si le bouton est appuyé, alors PORTD & 0x01 = 1
   - si le bouton est relâché, alors PORTD & 0x01 = 0

7) Dans le code fourni, le BTFSS agit comme une condition IF ; ainsi lorsque le bouton RD0 est à l'état 1, le programme ira dans la fonction *pressed* et à l'état 0 dans la fonction *not_pressed*. Inversement, BTFSC agit de la même manière mais à l'inverse : l'état 1 induit la première instruction.

8) Il est en effet possible de faire cette même opération sans bloquer le programme dans une boucle, il faut pour cela utiliser une mémoire : algorithme du code :

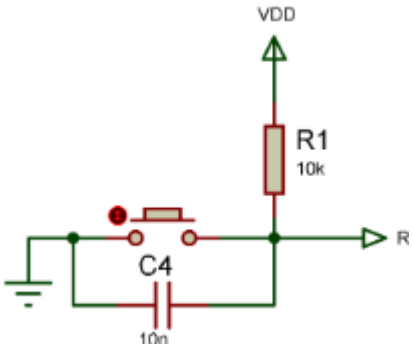


9) Le microcontrôleur est alimenté uniquement à l'aide du PC par câble.

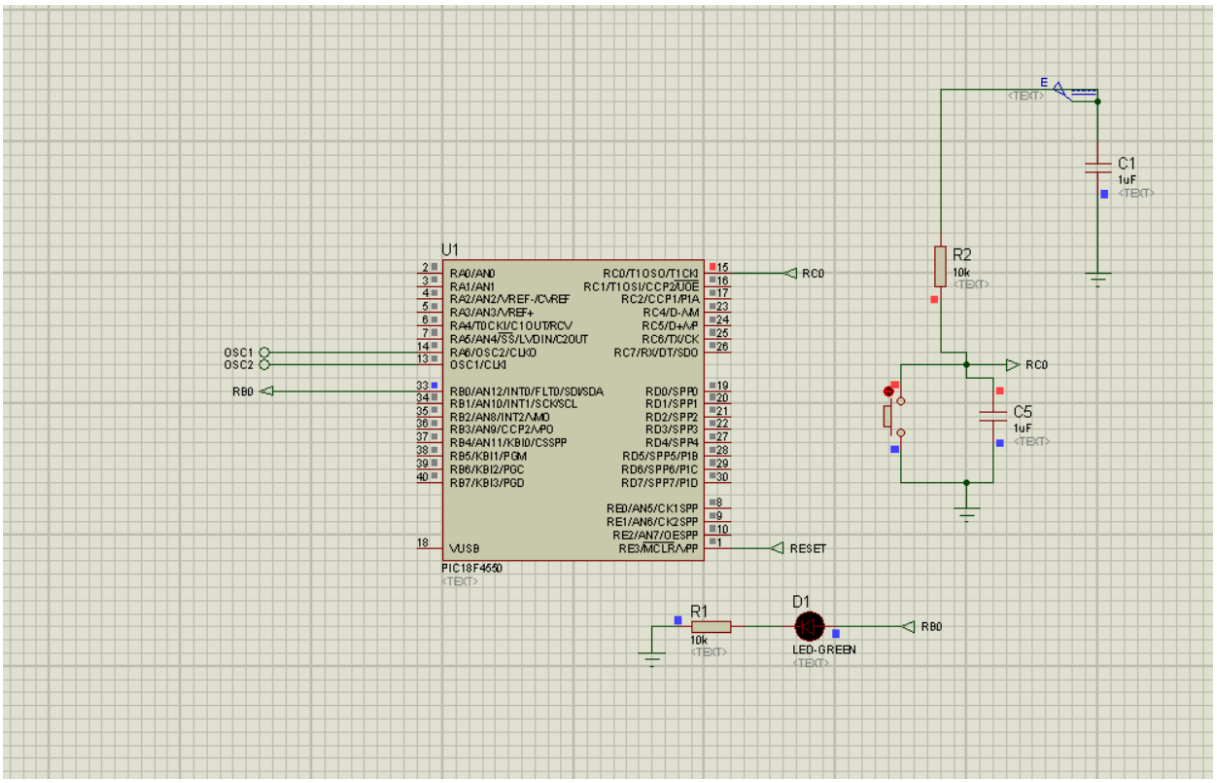
10) Les condensateurs C1 et C2 ont pour fréquence 48 MHz.

11) Le condensateur C3 sert de condensateur de découplage. Il est branché juste avant le terminal Ground et permet d'évacuer les harmoniques de hautes fréquences.

12) La résistance R1 et le condensateur C4 sont reliés au bouton et forment un système antirebond pour le bouton. Ainsi dès que l'on appuiera sur le bouton, le signal sera instantané sans restes de signaux décalés dans le temps. (peut être parler des effet de Gibbs)



13) Schéma sur ISIS en version debug

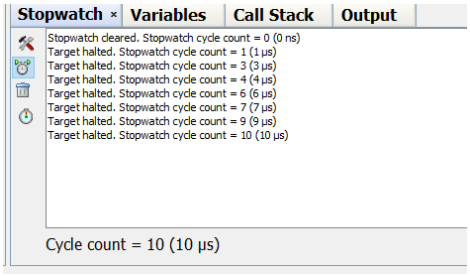


14)

Rapport TP II :

15) Pour échapper à la boucle infinie, on utilise des interruptions. Une interruption va lancer une partie du code avec une priorité plus élevé que la boucle infinie de al boucle principale, ce qui fait que on s'en échappe.

16) Le temps des interruptions n'est pas en secondes mais en microseconde. D'après le test des stopwatchs sur le code, le code s'interrompt toutes les microsecondes environ.



Task 9

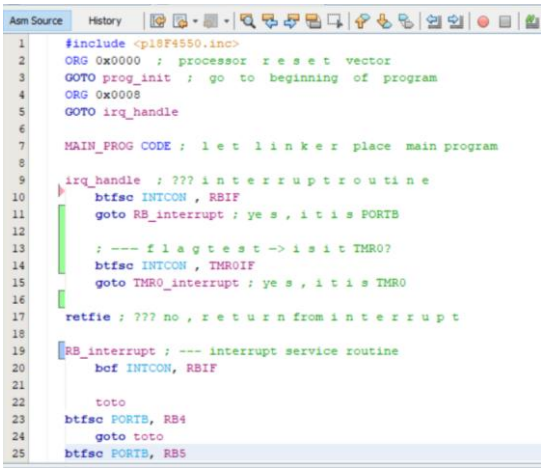
```
#include <p18F4550.inc>
ORG 0x0000  ;  processor  r e s e t  vector
GOTO prog_init  ;  go  to  beginning  of  program
ORG 0x0008
GOTO irq_handle

MAIN_PROG CODE ;  l e t  l i n k e r  place  main program

irq_handle  ; ??? i n t e r r u p t r o u t i n e
    btfsc INTCON , RBIF
    goto RB_interrupt ; ye s , i t i s PORTB
retfie ; ??? no , r e t u r n from i n t e r r u p t

RB_interrupt ; --- TMR0 interrupt service routine
    bcf INTCON, RBIF

    toto
btfsc PORTB, RB4
    goto toto
btfsc PORTB, RB5
    goto toto
btfsc PORTB, RB6
    goto toto
btfsc PORTB, RB7
    goto toto
```

```
20  btfsc PORTB, RB5
21      goto toto
22  btfsc PORTB, RB6
23      goto toto
24  btfsc PORTB, RB7
25      goto toto
26
27      incf PORTD
28  retfie ; return from interrupt
29
30  prog_init  ; i n i t i a l i z e your code
31      clrf TRISD ; PORTD is an output
32
33      bsf INTCON , GIE ; a c t i v a t e i n t e r r u p t s
34      bsf INTCON , RBIE ; a c t i v a t e RB i n t e r r u p t
35
36  goto MAIN_LOOP
37
38
39  MAIN_LOOP
40      ; --- wait for an interrupt
41  goto MAIN_LOOP
42
43  end
```

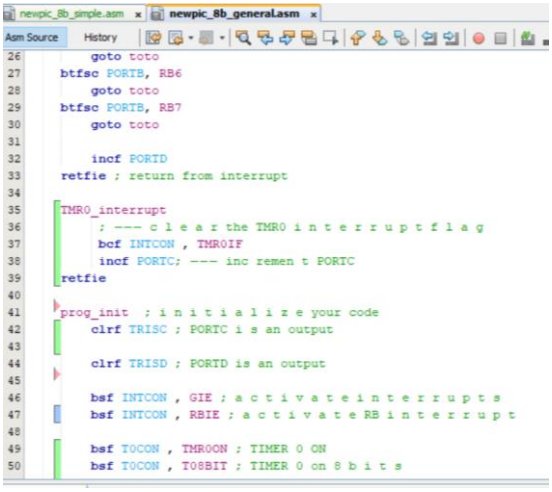
Task 10

Ici on utilise 2 sources d'interuptions : un timer et un bouton

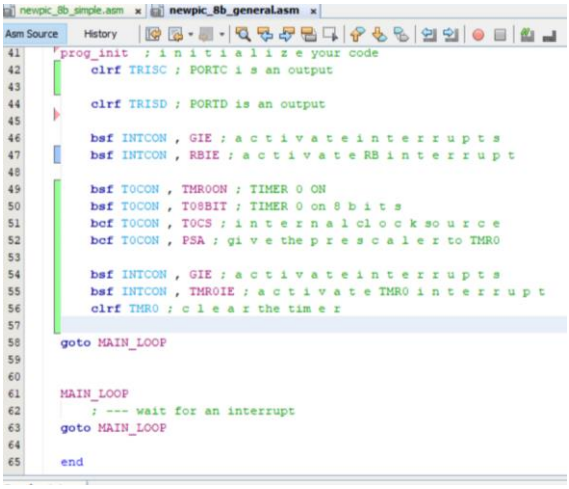
On compte les interuptions du timer sur les leds du port C et celles du bouton sur les leds du port D



```
#include <p18F4550.inc>
ORG 0x0000  ;  processor  r e s e t  vector
GOTO prog_init  ;  go  to  beginning  of  program
ORG 0x0008
GOTO irq_handle

MAIN_PROG CODE ;  l e t  l i n k e r  place  main program

irq_handle  ; ??? i n t e r r u p t r o u t i n e
    btfsc INTCON , RBIF
    goto RB_interrupt ; ye s , i t i s PORTB

    ; --- f l a g t e s t -> i s i t TMR0?
    btfsc INTCON , TMR0IF
    goto TMR0_interrupt ; ye s , i t i s TMR0

retfie ; ??? no , r e t u r n from i n t e r r u p t

RB_interrupt ; --- interrupt service routine
    bcf INTCON, RBIF

    toto
btfsc PORTB, RB4
    goto toto
btfsc PORTB, RB5
```



```
    goto toto
btfsc PORTB, RB6
    goto toto
btfsc PORTB, RB7
    goto toto

    incf PORTD
retfie ; return from interrupt

TMR0_interrupt
    ; --- c l e a r the TMR0 i n t e r r u p t f l a g
    bcf INTCON , TMR0IF
    incf PORTC; --- inc remen t PORTC
retfie

prog_init  ; i n i t i a l i z e your code
    clrf TRISC ; PORTC i s an output

    clrf TRISD ; PORTD is an output

    bsf INTCON , GIE ; a c t i v a t e i n t e r r u p t s
    bsf INTCON , RBIE ; a c t i v a t e RB i n t e r r u p t

    bsf T0CON , TMR0ON ; TIMER 0 ON
    bsf T0CON , T08BIT ; TIMER 0 on 8 b i t s
```



```
prog_init  ; i n i t i a l i z e your code
    clrf TRISC ; PORTC i s an output

    clrf TRISD ; PORTD is an output

    bsf INTCON , GIE ; a c t i v a t e i n t e r r u p t s
    bsf INTCON , RBIE ; a c t i v a t e RB i n t e r r u p t

    bsf T0CON , TMR0ON ; TIMER 0 ON
    bsf T0CON , T08BIT ; TIMER 0 on 8 b i t s
    bcf T0CON , T0CS ; i n t e r n a l c l o c k s o u r c e
    bcf T0CON , PSA ; gi v e the p r e s c a l e r to TMR0

    bsf INTCON , GIE ; a c t i v a t e i n t e r r u p t s
    bsf INTCON , TMR0IE ; a c t i v a t e TMR0 i n t e r r u p t
    clrf TMR0 ; c l e a r the tim e r

goto MAIN_LOOP


MAIN_LOOP
    ; --- wait for an interrupt
goto MAIN_LOOP

end
```