DUT Génie Électrique – Département GEII2

Génie Electrique – Semestre 1

II2 - Travaux pratiques

Informatique Industrielle – Utilisation de ports E/S

Auteur(s) : F. Wilmot, Y. Guinand, H. Discours
Date – Version : Novembre 2023

LISTE DES OBJECTIFS
Mettre en œuvre une carte microcontrôleur à base de PIC18F4550.
Utiliser les outils de développement associés, avec programmation en langage C.
Utiliser les ports d'E/S du microcontrôleur.
Réaliser le câblage de broches d'entrée/sortie TOR du microcontrôleur.
Réaliser l'acquisition d'une grandeur analogique à l'aide du CAN intégré au microcontrôleur.
Générer un signal à rapport cyclique variable à l'aide d'une sortie PWM du microcontrôleur.
Programmer une machine à états pour réaliser un système séquentiel simple.
Utiliser un Timer pour réaliser des temporisations.
LOGICIEL ET MATERIEL UTILISES
Carte microcontrôleur « Gamel Trophy » IUT de Cachan à base de microcontrôleur PIC18F4550
Logiciels MPLABX IDE de Microchip et compilateur XC8 de Microchip
Programmateur PicKit3
Carte d'extension IUT Cachan avec broches E/S et LEDs
CONSIGNES
La partie PREPARATION est à traiter avant la séance de TP. Elle sera évaluée par le professeur avant de commencer le TP.
Un compte rendu MANUSCRIT par personne est à rédiger pendant la séance. Ce compte rendu sera à remettre au professeur en fin de séance.

1. Préparation

1.1 Utilisation d'E/S TOR du microcontrôleur – Rappels du TP II1

On souhaite réaliser le câblage suivant :

- Un bouton poussoir BP, qui envoie sur l'entrée RB0 un « 1 » quand il est appuyé.
- Une led, qui s'allume quand on écrit un « 1 » sur la sortie RB1.

Q1. Le bouton doit-il être câblé en pull-up ou pull-down ? Rappeler le schéma correspondant.
La led doit-elle être câblée avec tirage masse ou +Vcc ? Rappeler le schéma correspondant.
Sur le document réponse DR1, représenter le câblage complet à réaliser (choix libre pour la led).

Q2. Ecrire un programme qui allume la led, quand on appuie sur BP (et qui l'éteint sinon).

1.2 Convertisseur analogique-numérique

Un potentiomètre est connecté sur l'entrée analogique 0 du microcontrôleur (AN0). Pour utiliser les fonctions de gestion du convertisseur analogique-numérique, il faut inclure le fichier `iut_adc.h` dans le programme (voir texte de TD pour le détail des fonctions).

Q3. A quoi servent les fonctions `adc_init` et `adc_read` ? Préciser le rôle des arguments de ces fonctions.

Q4. Commenter le programme suivant et déterminer son utilité.

```
#include <xc.h>
#include "iut_lcd.h"
#include "iut_adc.h"

void main(void)
     {
     int ValeurADC;
     lcd_init();
     adc_init(0);
     while (1)
          {
          ValeurADC = adc_read(0);
          lcd_position(0, 0);
          lcd_printf("ADC= %010b", ValeurADC);
          lcd_position(1, 0);
          lcd_printf("ADC= %4d", ValeurADC);
          }
     }
```

Q5. Le format "%4d" permet d'afficher un int sur 4 caractères. Justifier ce choix de format par rapport à la résolution de 10 bits du CAN.

1.3 Génération d'un signal PWM

Pour utiliser les fonctions de gestion des sorties PWM du microcontrôleur, il faut inclure le fichier `iut_pwm.h` dans le programme (voir texte de TD pour le détail des fonctions).

Q6. Quelles sont les valeurs des paramètres à passer à la fonction `pwm_init()` pour utiliser les deux sorties PWM avec une fréquence de 20 kHz ?

Q7. Quelle est la valeur du paramètre à passer à la fonction `pwm_setdc1()` pour générer un signal de rapport cyclique égal à 0,75 sur la broche RC2 ? (avec une fréquence de 20 kHz).
Reprendre le calcul pour les valeurs extrêmes du rapport cyclique (0 et 1).

1.4 Les Timers

Q8. De façon générale, c'est quoi un « timer » ? Que permet-il de mesurer ?
Sur un microcontrôleur, c'est quoi une horloge ?

Le PIC18F4550 comporte 4 timers (de timer0 à timer3), de 8 ou 16 bits (seul le timer2 est sur 8 bits). Tous les timers, qui ne sont en définitive que des compteurs, s'incrémentent au rythme d'une horloge. Par défaut, sur le PIC utilisé, l'intervalle de temps, entre chaque top d'horloge est de 83,333 ns. Pour compter à un rythme plus lent, il est possible de diviser la fréquence de l'horloge par un coefficient, appelé PRESCALER (prédiviseur en français).

Q9. Avec une valeur de PRESCALER égale à 256, quelle sera la cadence de comptage ?

Pour utiliser les timers, on dispose d'une bibliothèque qui regroupe les fonctions nécessaires. Voir page suivante.

void OpenTimer0(unsigned char config);

Cette fonction permet de configurer et démarrer le fonctionnement du TIMER0. Le paramètre de configuration est une association de différents masques validant chacun une caractéristique.

Exemple :
```
OpenTimer0(TIMER_INT_OFF &
           T0_16BIT &
           T0_SOURCE_INT &
           T0_PS_1_256 );
```

Détails des masques : **Enable Timer0 Interrupt :**

`TIMER_INT_OFF` pour ne pas utiliser les interruptions
`TIMER_INT_ON` pour que le timer génère des interruptions

Timer Width :

`T0_16BIT` pour utiliser le mode 16 bits du TIMER
`T0_8BIT` pour utiliser le mode 8 bits du TIMER

Clock Source :

Permet de sélectionner l 'horloge du TIMER :

- horloge interne : `T0_SOURCE_INT` Internal clock source (TOSC)
- horloge externe : `T0_SOURCE_EXT` External clock source (I/O pin)

External Clock Trigger :

Si l 'horloge externe est utilisée :

`T0_EDGE_RISE` pour compter les fronts montants (rising edge)
`T0_EDGE_FALL` pour compter les fronts descendants (falling edge)

Prescale Value :

Permet de choisir la valeur du PRESCALER (le pré-diviseur)

`T0_PS_1_1` Pas de pré-division
`T0_PS_1_2` Pré-division par 2 de la clock

`T0_PS_1_4`	…	4	…
`T0_PS_1_8`	…	8	…
`T0_PS_1_16`	…	16	…
`T0_PS_1_32`	…	32	…
`T0_PS_1_64`	…	64	…
`T0_PS_1_128`	…	128	…
`T0_PS_1_256`	…	256	…

void WriteTimer0(unsigned int Start);

Cette fonction permet d'écrire la valeur Start dans le TIMER. Ce dernier commencera alors à compter, à partir de cette valeur.

unsigned int ReadTimer0(void);

Cette fonction permet de lire la valeur actuelle du TIMER. Donc le nombre de nombre de tops d'horloge depuis le start. Cette valeur correspond donc à un temps.

Q10. Quel est le paramétrage du timer, sur l'exemple donné en haut de cette page ?

Q11. En considérant une période d'horloge de 21,333 µs, que réalisent les instructions suivantes :

```
writeTimer0(0);
if(ReadTimer0()>23437);
```

Fin de la partie préparation, mais il n'est pas interdit de lire la suite avant la séance de TP…

2. Expérimentation

2.1 Utilisation d'E/S TOR du microcontrôleur

Tous les registres du microcontrôleur sont accessibles bit à bit. Par exemple, pour accéder à l'octet du port B, on utilise `PORTB` et pour accéder au port B bit à bit, on utilise `PORTBbits`.

Exemples : `PORTBbits.RB0` c'est le bit 0 du Port B
`PORTBbits.RB2` c'est le bit 2 du Port B

Q12. Tester le programme donné ci-dessous.
Expliquer ce qu'il réalise.

```
#include <xc.h>
#include "iut_lcd.h"

void main(void)
     {
     char bp0, bp1, led ;

     TRISB = TRISB | 0x18; // Pin RB4, RB3 du port B, définies en entrée
     TRISA = TRISA & 0xBF; // Pin RA6 définie en sortie

     lcd_init();

     while (1)
        {
        bp0 = PORTBbits.RB3 ; // lecture du bit 3 du port B
        bp1 = PORTBbits.RB4 ; // lecture du bit 4 du port B

        lcd_position(0, 0);
        lcd_printf("BP0: %d BP1: %d", (int) bp0, (int) bp1);

        if (bp0 == 0) //bp0 appuyé
              {
              led = 1 ; //mise à 1 de la variable LED
              }
        else {
              led = 0 ; // mise à 0 de la variable LED
             }
        PORTAbits.RA6 = led ; // affectation de LED au bit 6 du port A
        }
     }
```

Q13. Ajouter au programme précédent, le nécessaire pour réaliser le fonctionnement suivant :

Si BP2 (sur RB5) est appuyé, alors allumer une led de la carte d'extension, qui sera branchée sur RB1 (ne pas oublier de modifier la ligne de configuration E/S du port B).

Imprimer le programme **et faire valider le fonctionnement par le professeur.**

2.2 Machine à états

Q14. Dessiner le graphe de la machine à états programmée ci-dessous, en montrant les différents états et les conditions de passage entre eux.

```
#include <xc.h>
#include "iut_lcd.h"

void main(void)
      {
      char etat = 0;
      char bp0, bp1, bp2;

      TRISB = TRISB | 0x38;

      lcd_init();

      while (1)
           {
           bp0 = PORTBbits.RB3;
           bp1 = PORTBbits.RB4;
           bp2 = PORTBbits.RB5;

           switch (etat)
             {
             case 0 :
                   if (bp1 == 0)  etat = 1;
                   break;
             case 1 :
                   if (bp2 == 0)  etat = 2;
                   break;
             case 2 :
                   if (bp0 == 0)  etat = 0;
                   break;
             }
          lcd_position(0, 0);
          lcd_printf("Etat = %d", (int) etat );
          }
}
```

Q15. Saisir et tester le programme donné ci-dessus.
Expliquer ce qu'il réalise.

Q16. Dessiner un graphe à 3 états, correspondant au fonctionnement suivant :
- L'état initial est l'état 0.
- A partir de l'état 0, un appui sur le bouton vert permet d'atteindre l'état 1.
- A partir de l'état 0, un appui sur le bouton bleu permet d'atteindre l'état 2.
- A partir de l'état 1, un appui sur le bouton jaune renvoie à l'état 0.
- A partir de l'état 2, un appui sur le bouton jaune renvoie à l'état 0.
- A partir de l'état 1, un appui sur le bouton bleu permet d'atteindre l'état 2.
- A partir de l'état 2, un appui sur le bouton vert permet d'atteindre l'état 1.

Q17. Programmer la machine à états de la question précédente.
Sur le compte-rendu, noter les modifications apportées par rapport au premier programme.
Faire valider le fonctionnement par le professeur.
Imprimer le programme avec des **commentaires** sur toutes les lignes importantes.

2.3 Les timers

Pour utiliser les timers, on dispose d'une bibliothèque qui regroupe les fonctions nécessaires. Pour utiliser cette bibliothèque, il faut :

- Ajouter les fichiers `iut_timers.h` et `iut_timer.c` dans le projet
- Inclure le fichier d'en tête `iut_timer.h` avec l'instruction `#include "iut_timer.h"`

Saisir et tester le programme suivant :

```
#include <xc.h>
#include "iut_timers.h"

void main(void)
      {
      TRISA = TRISA & 0xBF; // Pin RA6 en sortie

      OpenTimer0( TIMER_INT_OFF &
                  T0_16BIT &
                  T0_SOURCE_INT &
                  T0_PS_1_256 );//config TIMER
        // Horloge interne à 12 MHz, donc un top toutes les 83,333 ns
        // Horloge divisée par 256, donc incrémentation toutes 21,333 us

      WriteTimer0(0); // mise à 0 du timer0 et START

      while(1)
          {
          if(ReadTimer0()>23437); // Test si timer > 0,5 sec
             {
             PORTAbits.RA6 = !PORTAbits.RA6; //inversion bit 6 du port A
             WriteTimer0(0); // Relance du timer
             }
          }
      }
```

Q18. Expliquer ce que réalise le programme ci-dessus.

Q19. Quelle est la valeur maximale de temps, que peut compter un timer 16bits ?

2.4 Timer et machine à états

Q20. Dessiner un graphe d'états, correspondant au fonctionnement suivant :

- L'état initial est l'état 0.
- A partir de l'état 0, un appui sur le bouton vert permet d'atteindre l'état 1 et lancer timer
- A partir de l'état 1, si timer a atteint 1 seconde, aller à l'état 2 et relancer le timer
- A partir de l'état 2, si timer a atteint 1 seconde, aller à l'état 3 et relancer le timer.
- A partir de l'état 3 : - un appui sur le bouton bleu permet d'atteindre l'état 0.
  - le non-appui sur le bouton bleu permet d'atteindre l'état 1 et relance le timer.
- A l'état 1 la LED verte (RA6) est allumée (et éteinte dans tous les autres états).

Q21. Programmer et tester la machine à états. La valeur de la variable d'état sera affichée sur l'écran.
Faire valider le fonctionnement par le professeur.
Imprimer le programme avec des **commentaires** sur toutes les lignes importantes.

2.5 Convertisseur analogique-numérique

Q22. Mettre au point le programme de la question Q4 de la préparation. Noter les valeurs pour les positions extrêmes du potentiomètre. A partir de la valeur max, vérifier que la résolution du CAN est bien de 10 bits.

2.6 Génération d'un signal PWM

Q23. Réaliser un programme qui génère deux signaux PWM de fréquence 20 kHz. Le premier, sur la broche RC2, avec un rapport cyclique égal à 0,75 et le second avec un rapport cyclique de 0,25 sur la broche RC1. Vérifier le fonctionnement à l'oscilloscope et fournir le programme dans le compte-rendu (recopié ou imprimé).

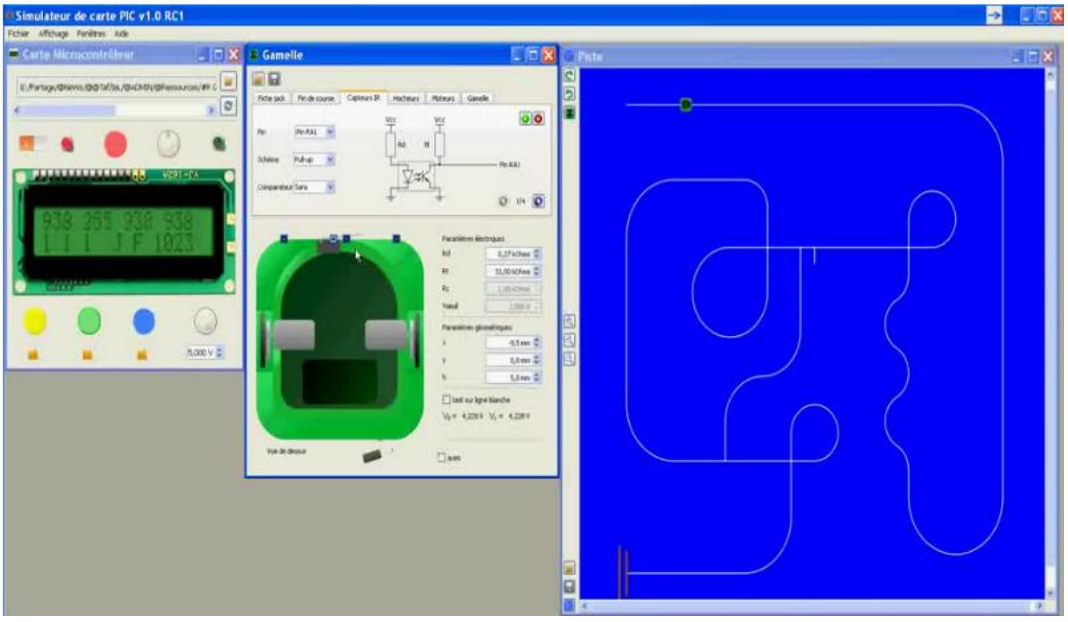
Q24. Modifier le programme pour que le potentiomètre contrôle la valeur du rapport cyclique du premier signal. Faire afficher la valeur du rapport cyclique sur l'écran. Il est demandé d'utiliser toute la course du potentiomètre pour obtenir un réglage entre 0 et 1.
Vérifier le fonctionnement à l'oscilloscope.

Q25. Compléter le programme pour que le potentiomètre contrôle aussi la valeur du rapport cyclique du second signal, mais en « sens inverse ». C'est-à-dire qu'il augmente si le premier diminue, et inversement. Ainsi lorsque le premier est à 0, le second sera à 1, et inversement le premier sera à 1 si le second est à 0. Faire afficher la valeur des rapports cycliques sur l'écran.
Vérifier le fonctionnement à l'oscilloscope et **faire valider par le professeur.** Imprimer le programme avec des **commentaires** sur toutes les lignes importantes.

2.7 La Gamel

Q26. Programmer la Gamel sur PicSimu pour qu'elle avance et se déplace en dessinant un carré (changement de direction toutes les 2 secondes).

Faire valider le fonctionnement par le professeur.
Imprimer le programme avec des **commentaires** sur toutes les lignes importantes.



Document Réponse DR1 **Noms :**.....

Utilisation d'E/S TOR du microcontrôleur