ABSA-NT
Développement
Electronique Informatique

PIC EVAL Dev Board

PIC18F97J60

TP1 :

Prise en main de l'environnement de programmation pour la carte

PIC EVAL-ANFA

Pour répondre aux questions et justifier vos réponses, vous pouvez faire des copies d'écran ou des schémas.

I. Objectifs du TP

Le but de ces manipulations est de découvrir l'environnement de programmation MPLAB et la carte de développement Pic Eval.

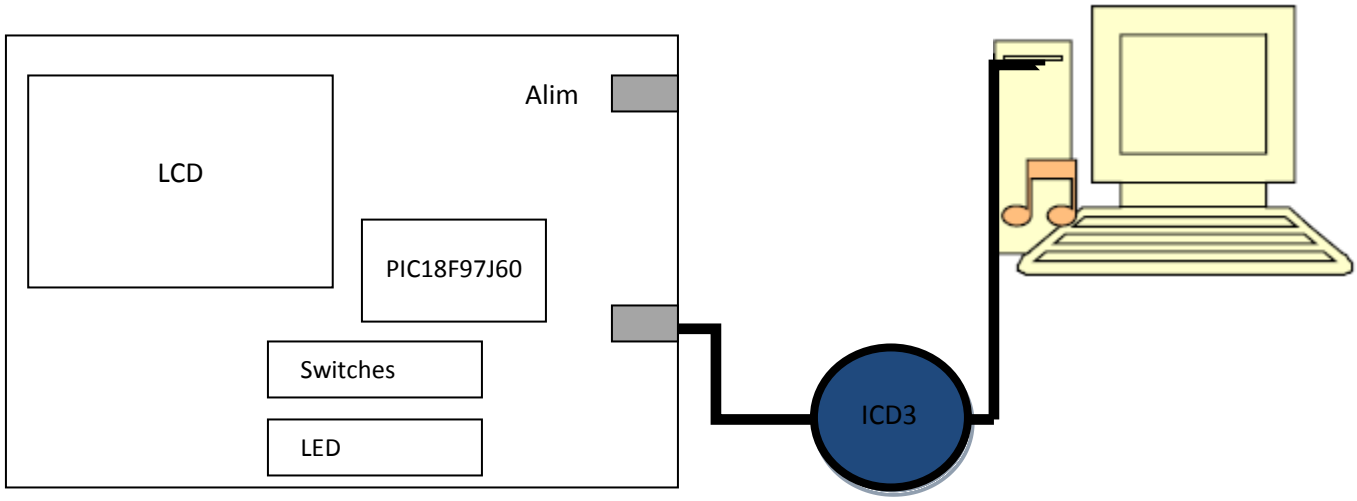
1. Matériel à votre disposition

Certains des éléments suivants sont présents sur votre plan de travail.

- Une carte Pic Eval
- Un ICD3
- Un câble USB
- Un câble d'alimentation pour la carte

2. Montage du TP

Il faut impérativement suivre les instructions du fichier 'MPLAB_Manuel.pdf' pour éviter tout dysfonctionnement.



2. Présentation de la carte PIC EVAL

La carte de développement «PIC EVAL» est un outil de développement adapté pour la programmation et l'expérimentation avec le Micro-contrôleur PIC18F97J60 de chez Microchip.

PIC EVAL intègre :

- 1 connecteur de programmation mémoire Pic.
- 1 connecteur USB pour la communication PC.
- 1 afficheur 7 segments et 8 leds.
- 1 Capteur de proximité et lumière.
- 1 module Bluetooth / Android.
- 1 microcontrôleur PIC 18F67J60.
- 1 mémoire flash 16 MB.
- 1 écran tactile touch screen.
- 1 capteur de température.
- 1 horloge temps réel RTC.
- 2 Relais.
- 1 serveur web.
- 1 liaison Ethernet Maitre.
- 1 liaison Ethernet esclave.
- 1 liaison I2C.

3. Présentation de MPLAB

Ouvrir le projet TP_Led (File > Open Project > TP_Led)

1. Présentation générale

MPLAB est un Environnement de Développement Intégré (IDE) qui permet le développement logiciel des microcontrôleurs PIC et les contrôleurs de signal numériques dsPIC de la société Microchip.

MPLAB IDE permet :
- De créer le code source à l'aide de l'éditeur intégré.
- D'assembler, compiler et lier les fichiers sources qui peuvent provenir de langages différents. Un assembleur, un "linkeur" et un gestionnaire de librairies sont fournis avec MPLAB. Un compilateur C est vendu à part par Microchip; des outils de tierces parties peuvent aussi être utilisés.

- De déboguer le code exécutable en observant le déroulement du programme à l'aide du simulateur fourni, de l'émulateur temps réel ICE 2000 ou de l'ICD2 (in circuit debugger) ou encore l'ICD3 développés par Microchip. Des outils de tierces parties peuvent aussi être utilisés.
- D'effectuer des mesures temporelles avec le simulateur ou l'émulateur.
- De voir les variables grâce à des fenêtres d'observation (watch windows).

3. Premier programme

Le programme de la Pic Eval utilise un bouton poussoir et une led. Ce programme allume la led L1 tant que le bouton poussoir BTN1 reste appuyé.

Votre fonction **main ( )** pour ce projet sera constitué de la fonction suivante :

```
void main(void)
{
   // ---------------- Initialisation des fonctions ----------------- //
   InitializeBoard();
   Timer0_Init();

   while(1) {
        if(!BUTTON0_IO==1) {
                INTCONbits.TMR0IE = 1;
                LED1_IO=0; }
         else LED1_IO=1;
   }
}
```

Compiler le projet en entier (Run> Clean and Build Project). Lancer l'application dans la Pic Eval (Run > Run Project) et vérifier le fonctionnement. Pour cela appuyez sur le bouton poussoir BTN1.

Q1. En vous appuyant sur le programme donné, réaliser un programme qui allume toutes les Leds.

Q2. Réalisez un programme que, lorsqu'on appuie une fois sur le bouton poussoir BTN1, allume la led L1, ensuite on appuie à nouveau et la Led L2 s'allume, on appuie à nouveau, la led L3 s allume et ainsi de suite.

La carte Pic Eval possède un timer 0, on peut l'utiliser comme interruption pour réaliser un chenillard avec un certain temps (modifiable via l'interruption Timer0) entre chaque nouvelle Led allumée.

Q3. Écrire un chenillard simple : une led se déplaçant de la gauche vers la droite.

Code du timer :

```
void TIMER0_ISR(void) // Interruption Timer0
{
   if(INTCONbits.TMR0IF)
   {

      iCptTimerRotationLed++;
      if(iCptTimerRotationLed==10)
      {
         iCptTimerRotationLed=0;

         //Mettre fonction qui réalise le chenillard

      }
         ////////////////////////////////////////////////////////
      TMR0H = Clock_Tick >> 8;
      TMR0L = Clock_Tick & 0xFF;
      // Reset interrupt flag
      INTCONbits.TMR0IF = 0;
   }
}
```

Q4. Écrire un chenillard simple : une led se déplaçant vers la gauche.

Q5. Écrire un chenillard qui va d'abord de la gauche vers la droite et arrivé au bout, reviens vers la gauche.

Q6. Écrire un chenillard qui allume dans un premier temps les 4 premières leds, et dans un second temps, les 4 leds de droite, successivement.

Maintenant on va se servir des boutons poussoirs et des Leds pour lancer ces programmes.

Q7. En utilisant la condition switch case, réaliser un programme qui, lorsqu'on appuie sur le BTN1 lance le chenillard vers la gauche, le BTN2 lance le chenillard vers la droite, BTN3 lance le chenillard de la question Q5 et le BTN4 lance le programme de la question Q6.

Codes donnés :

Fonction TIMER0_ISR(void)

```
void TIMER0_ISR(void) // Interruption Timer0
{
   if(INTCONbits.TMR0IF){
      iCptTimerRotationLed++;
      if(iCptTimerRotationLed==10)
      {
         iCptTimerRotationLed=0;
         switch(iSensRotationLed)
         {
           case 1 :
                        // sens rotation gauche
            break;
            case 2 :
                        // sens rotation droite
            break;

            case 3 :
                        // sens rotation droite max ensuite gauche
            break;
            case 4 :
                        // sens rotation droite 4 par 4
            break;

         }
      }
      TMR0H = Clock_Tick >> 8;
      TMR0L = Clock_Tick & 0xFF;
      INTCONbits.TMR0IF = 0;
   }
}
```

Fonction main :

```
void main(void)
{
   // ---------------- Initialisation des fonctions ----------------- //
   InitializeBoard();
   Timer0_Init();

   while(1)
   {
      if(!BUTTON0_IO==1)
      {
         iBouton=0;
      }

         //Compléter pour les autres boutons poussoirs

      switch(iBouton)
      {
         case 0 :
            INTCONbits.TMR0IE = 1;
            iSensRotationLed=1;                    // sens rotation gauche
            iBouton=attenteAutreCommande;
            break;

         // Compléter ici
         case attenteAutreCommande :
            break;
      }
   }
}
```