TP1: découverte de PIC C, écriture & lecture sur port

I. Créer un projet.

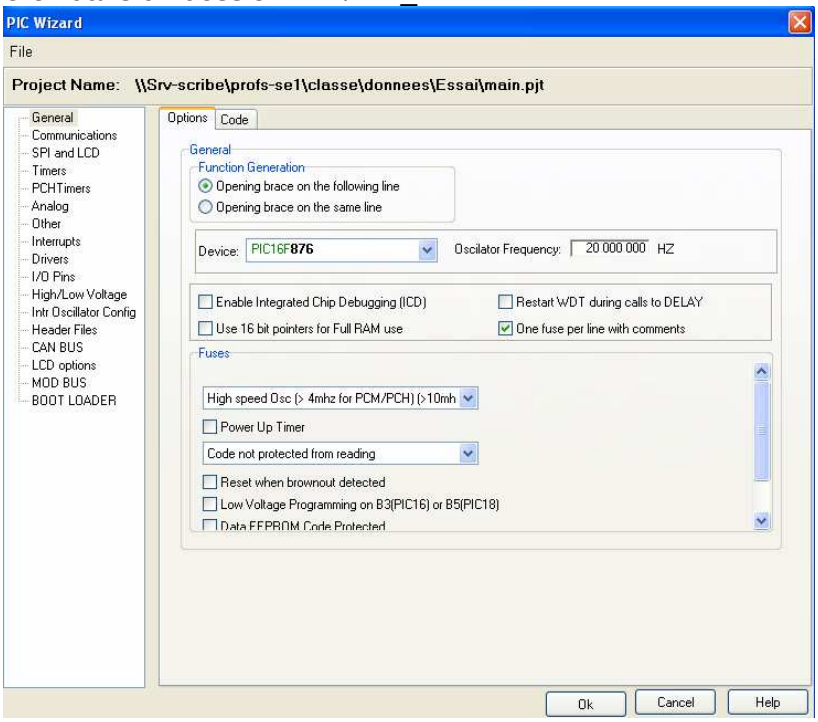
Reportez-vous au document "TP PIC 16F876 notice_CCSC STS.pdf" et à la démonstration par les intervenants.

Le but de ce programme consistera simplement à allumer la led rouge (PC0)

Il est inutile d'écrire un algorithme pour cela: il faut configurer PC0 en sortie puis appliquer un 1 logique sur PC0.

I.1. Travail à effectuer:

Utilisez l'assistant pour créer un projet, on l'appellera tp1_1 (ne pas écrire d'extension) que vous placerez dans un dossier TP1\TP1_1.



Configuration :

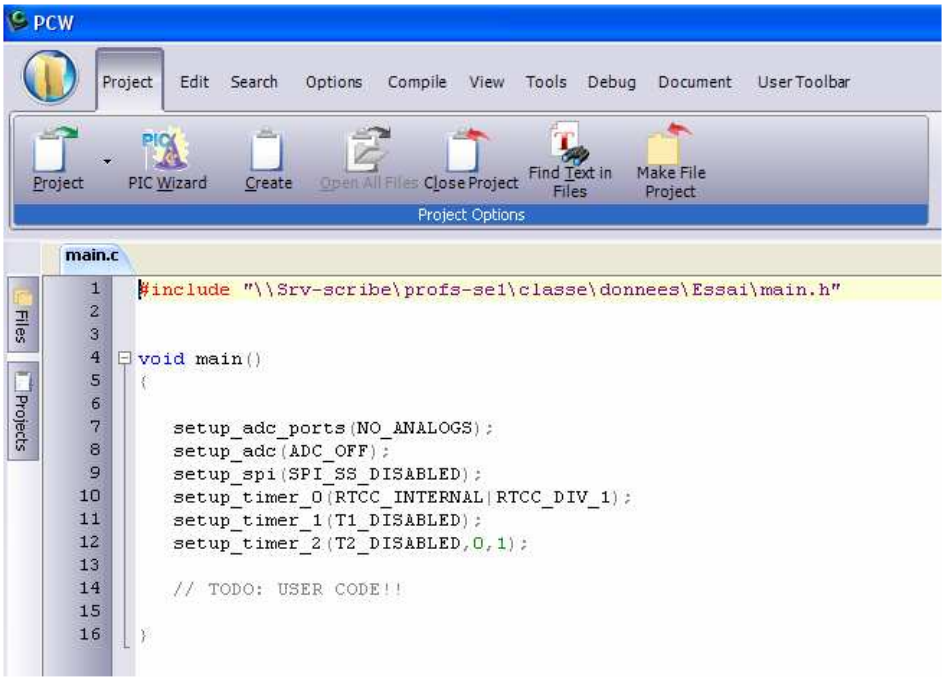
PIC16F876
20MHz
High speed Osc (>4MHz)
Code non protected from reading

décochez toutes les cases sauf "*One fuse per line with comment*" et « Opening brace on the following line ».

Vous pouvez voir les valeurs des fusibles en sélectionnant l'onglet CODE

```
Inserted into .h file:
#device adc=8
#FUSES NOWDT            //No Watch Dog Timer
#FUSES HS               //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPUT            //No Power Up Timer
#FUSES NOPROTECT        //Code not protected from reading
#FUSES NOBROWNOUT        //No brownout reset
#FUSES NOLVP            //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD            //No EE protection
#FUSES NOWRT            //Program memory not write protected
#FUSES NODEBUG          //No Debug mode for ICD
#use delay(clock=20000000)
```

Validez par "OK". Vous obtiendrez le texte suivant:



L'assistant a également créé dans le même dossier un fichier *tp1_1.h*.
Avant l'accolade fermante, vous ajouterez les 2 lignes suivantes:

```
set_tris_C(0b11111110);        /* PC0 en sortie */
output_C(0b00000001);          // état haut sur PC0
```

Appuyer sur la touche F9 ou le bouton (compile) Cette fenêtre apparaît quelques instants.

I.2. Chargement du programme

Effectuer le branchement du kit PICKIT2. Chargez le programme puis le lancer : la led rouge s'allume ! ! !

Laissez le bouton RESET est appuyé: elle s'éteint, relâchez-le, elle s'allume.

Votre premier programme est opérationnel.

I.3. Analyse:

On a utilisé deux fonctions intégrées de PIC C: ***set_tris_X*** et ***output_X*** .

Vous pouvez avoir de l'aide sur ces fonctions en cliquant une fois sur le mot puis en appuyant sur la touche F1.

Pour allumer la LED, on peut aussi utiliser la fonction **output_high** qui s'applique à un bit isolé.

Cherchez dans l'aide comment l'utiliser, modifier le programme et testez.

II. Lecture d'un bit de port

Créer un nouveau projet (tp1_2), mêmes caractéristiques que le précédent.
Algorithme du programme à effectuer:

```
Début
    Faire
        Si lecture(joystick_centre) = 0
            Alors allumer(led rouge)
            Sinon allumer(led verte)
    Tant que(vrai)
Fin
```

Le bouton joystick_centre est câblé sur PB4, les leds Rouge, Jaune, Verte sur PC0, PC1, PC2.

La lecture d'un bit de port se fait avec la fonction ***input***. Voir l'aide sur cette fonction.

Écrire le programme correspondant et le tester.

III. Utilisation d'une variable

III.1. Affichage binaire sur 3 segments des appuis sur BP1.

```
Début
    compte ← 0
    Faire
        attendre(joystick_centre)
        compte ← compte + 1
        Suivant  compte
            si 1:  allumer(led verte)
                   éteindre(led jaune)
                   éteindre(led rouge)
            si 2:  éteindre (led verte)
                   allumer (led jaune)
                   éteindre(led rouge)
            si 3:  allumer (led verte)
                   allumer (led jaune)
                   éteindre(led rouge)
            si 4:  éteindre (led verte)
                   éteindre (led jaune)
                   allumer (led rouge)
            si 5:  allumer (led verte)
                   éteindre (led jaune)
                   allumer (led rouge)
            si 6:  éteindre (led verte)
                   allumer (led jaune)
                   allumer (led rouge)
            si 7:  allumer (led verte)
                   allumer (led jaune)
                   allumer (led rouge)
            si 8:  éteindre (led verte)
                   éteindre (led jaune)
                   éteindre(led rouge)
                   compte ← 0

        tempo(200ms)
    Tant que(vrai)
Fin
```

On veut compter les appuis sur le centre du joystick et les afficher en binaire sur les leds rouge, jaune, verte (0b000 à 0b111), le LSB sera représenté par la led verte, le MSB par la led rouge)

Il faut donc créer une variable de type entier. On l'appellera ***compte***.

Le koystick présente des rebonds, pour éviter de les prendre en compte, les essais ont montré qu'il faut attendre 200ms entre 2 détections, nous sommes donc appelés à utiliser une fonction de tempo: ***delay_ms***. (voir l'aide).

ALGORITHME:

Il n'existe pas de fonction ***attendre***!!! il faut donc utiliser une boucle tant que:

tant que (lecture(joystick_centre)==1)
faire: rien

Pour faciliter l'écriture du programme, on mettra des équivalences:

```
#define   led rouge        PIN_C0
#define   led jaune        PIN_C1
#define   led verte        PIN_C2
#define   joystick_centre  PIN_B4
```

Travail à effectuer

a) Créer un projet tp1_3.
b) Traduire l'algorithme en C, utilisez le type ***int8*** pour ***compte***. Tester le programme
c) Que se passe-t-il si on laisse le bouton poussoir appuyé?
d) Donner une solution pour remédier à cela.

III.2. Utilisation du port RS232

Il est donc intéressant d'utiliser la RS232 pour afficher des données sur un terminal RS232 (PC, Minitel, afficheur série …).

Ici on va bien sûr se servir du PC comme terminal.

Il faut l'interface série du micro ordinateur sur l'interface série de la carte « Pic_V2».

L'environnement PIC C intègre un terminal (Menu Tools → Serial Port Monitor), on peut aussi utiliser l'hyperterminal de Windows ou tout type de logiciel d'émulation de terminal.

PIC C intègre quelques fonctions d'envoi et de réception de données de type caractère:

getc, **putc**, **gets**, **puts** et **printf**.

La fonction la plus utilisée pour l'envoi de caractères est la fonction ***printf***. Son usage n'est pas très simple (voir diapo et aide).

```
Début
    afficher("bonjour, compteur d'appui touche!!!!")
    compte ← 0
    afficher(valeur de compte en décimal)
    afficher(valeur de compte en hexadécimal)
    afficher( nouvelle valeur de compte en signé )

    Faire
        attendre(appui joystick_centre)
        compte ← compte + 1
        afficher( nouvelle valeur de compte en décimal)
        afficher( nouvelle valeur de compte en hexadécimal)
        afficher( nouvelle valeur de compte en signé )
        attendre(relaché du bouton)
        tempo(200ms)
```

Travail à effectuer

a) Créer un projet TP1_3RS232. Pour ce projet, il faut configurer la RS232 avec l'assistant en utilisant le 2ème onglet: 9600 b/s, sans parité, Transmit sur PC6, Receive sur PC7, Bits sur 8.

b) traduire l'algorithme en C, le tester.

c) On a défini ***compte*** comme un entier sur 8 bits. Au lieu de commencer à 0, commencez à 254. Que constatez-vous? Remplacez le type ***int8*** par ***signed int8*** et dans le corps du programme, n'oubliez pas de remplacer les ***%u*** par des ***%d***.
Essayez en commençant à 126.

d) Remplacez par ***int16***, ***%lX*** et ***%ld***. Commencez à 32766.