TD 1 - Prise en main environnement de développement

En préalable :

- Dans un explorateur Windows, valider l'affichage des extensions : Menu Outils → Options des dossiers, onglet Affichage, et décocher la case "Masquer les extensions..."
- Se créer un dossier de travail sur le DD

- Outils : carte explorer16 + ICD2 + MPLAB
- Langage utilisé : Assembleur PIC
- Documents : voir Moodle

1 Forme générale d'un programme en assembleur

- Un programme assembleur est constitué :
- de commentaires (caractère " ;"),
- de **mnémoniques**, correspondant à une instruction exécutable par le processeur,
- de **directives assembleur** (identifiés par un "." comme premier caractère), qui ne génèrent pas de code exécutable, mais permettent :
  - de réserver de la mémoire pour des variables,
  - de spécifier diverses informations à l'assembleur ou au linker.

Le source est organisé en 3 colonnes, dont la 1ère est vide le plus souvent (champ d'étiquettes) :

```
        .section .nbss, bss, near
var1:   .space 2                ;Example of allocating 1 word of space for
                                ;variable "var1".
Flags:  .space 2

;.....................................................................
;Code Section in Program Memory
;.....................................................................
.text                           ;Start of Code section
__reset:
        MOV #__SP_init, W15     ;Initalize the Stack Pointer
        MOV #__SPLIM_init, W0   ;Initialize the Stack Pointer Limit Register
        MOV W0, SPLIM
        NOP                     ;Add NOP to follow SPLIM initialization

        RCALL _wreg_init         ;Call _wreg_init subroutine
                                ;Optionally use RCALL instead of CALL
        rcall   InitTMR1          ; Initialize TMR1
        rcall   InitPortA         ;Initialize PORTA
        rcall   InitUART2         ;Initialize UART2
        rcall   InitPSVOperation  ;Initialize PSV operation
```

2 Premier programme

Etape 1 : construction de l'exécutable

1. Démarrer MPLAB.
2. Menu Configure → Select Device : choisir le PIC24FJ128GA010.
3. Créer un nouveau projet : Menu Project → Project Wizard, en sélectionnant le **PIC24FJ128GA010**, puis :
4. Choisir la *ToolSuite* Microchip C 30 (laisser le champ *location* tel quel !)
5. Donner un chemin et un nom au projet (TD1_A).
6. Récupérer depuis Moodle le fichier `source_td1.s`, le copier dans son dossier de travail, et l'ajouter au projet.
7. Vérifier qu'on est en mode *Debug*, puis cliquer sur *Build All*.

⇒ Vous devez obtenir *BUILD SUCCEEDED* dans le fenêtre *Output*.

Question : combien d'octets occupe le programme dans la mémoire du microcontroleur ? (analyser les informations affichées dans la fenêtre "Output", qu'on retrouve également dans le fichier .map) : _____

Analyse du code Etudier le source, et essayer de comprendre ce que doit faire le programme (voir éléments donnés en fin de ce document).

Etape 2 : téléchargement dans la cible et exécution

Une fois la construction de l'exécutable achevée :

1. menu Debugger → Select Tool. Sélectionner "MPLAB ICD 2"
2. menu Debugger → Connect

Vérifier qu'il n'y a pas d'erreur : (message *MPLAB ICD 2 Ready*)

Le transfert du fichier se fait avec la commande menu Debugger → Program. La programmation de la mémoire Flash se fait alors (voir barre de progression en bas). A chaque build du projet, il faudra penser à re-transférer le programme !

On peut ensuite exécuter le programme en cliquant sur le bouton de la barre d'outils *Run* (petit triangle). L'arrêt se fait avec le bouton *Halt*.

On peut poser des points d'arrêts (*Breakpoint*) par un simple double clic sur la ligne du source. En mode *Run*, l'exécution s'arrêtera dès le passage sur cette instruction. On peut exécuter le programme en pas-à-pas, avec le bouton de la barre d'outils *Step Into* (touche F7).

Prendre le temps d'essayer ces différents modes d'exécution afin de se familiariser avec l'IDE.

3 Travail à effectuer

1. Repérer sur la doc de la carte (accessible sur Moodle, pp.39-45) sur quels ports et quelle broches sont connectés les dels et les BP (Boutons Poussoirs).
2. Copier/coller le source dans un nouveau fichier (TD_12.s), en faire un nouveau projet, et modifier le programme pour simuler une connexion directe entre le BP S3 (RD6, bit 6 du port D) et la del D3 (RA0, bit 0 du port A) : DEL allumée en cas d'appui, et éteinte au relâchement du BP.
3. Copier/coller le source dans un nouveau fichier (TD_13.s), et modifier le programme pour avoir le fonctionnement suivant : Un appui sur S3 allume la del D3, un appui sur S6 éteint la del.
4. Copier/coller le source dans un nouveau fichier (TD_14.s), et modifier le programme pour avoir le fonctionnement suivant : Chaque appui sur S3 inverse l'état de la del D3.

Annexe : quelques instructions assembleur PIC24F

Voir datasheet du PI24F128 sur Moodle, p.192 et suivantes.

Ecriture dans les registres ou la RAM

- `bset XXXX,#n` : Positionne à 1 le bit n (n=0 à 15, ou 0 à 7) du registre (ou de l'adresse mémoire) XXXX
- `bclr XXXX,#n` : Positionne à 0 le bit n du registre (ou de l'adresse mémoire) XXXX
- `move #n,XXXX` : Copie la valeur n dans le registre ou l'adresse XXXX
- `clr XXXX` : Met à 0 le registre ou l'adresse XXXX
  (équivalent à `move #0,XXXX`)

Tests de bits

En assembleur, les tests s'effectuent presque toujours en deux instructions :

1. D'abord une comparaison, qui va positionner les bits d'états N, Z, OV, C (registre d'état) :
   - N : Negative : recopie du bit de poids fort de la valeur ou du résultat
   - Z : Zero : activé en cas de valeur nulle, ou d'égalité lors d'une comparaison
   - C : Carry (débordement)
   - OV : OVerflow (débordement arithmétique)
2. Ensuite un branchement conditionnel, qui va réaliser - ou non - le déroutement du programme, selon la valeur du bit en question.

Exemple :

```
1       btst    PORTA,#3    ; bit test: positionne le bit Z
2                           ; en fonction de l'état du bit 3
3       bra     Z,LABAS     ; branchement conditionnel
4                           ; (si Z à 1 => si le bit est à 0)
```

⇒ "Se brancher à l'étiquette LABAS si le bit 3 du registre PORTA est à 1"

On pourra obtenir le branchement pour la condition inverse (si le bit est à 1 ici) en préfixant la condition avec un 'N' :

```
1       bra     NZ,LABAS    ; branchement si le bit est à 1
```

Il existe également des instructions de branchement inconditionnel, qui déroutent l'exécution à une autre adresse du programme.

`bra debut` ⇒ Déroute l'exécution, qui continue à la ligne où se trouve l'étiquette "debut"