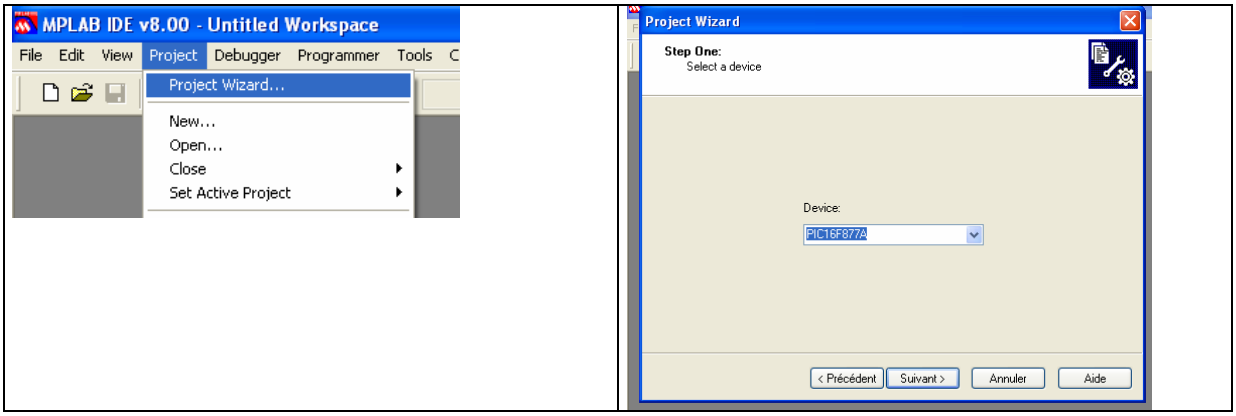
Premiers pas avec MPLAB 8.0

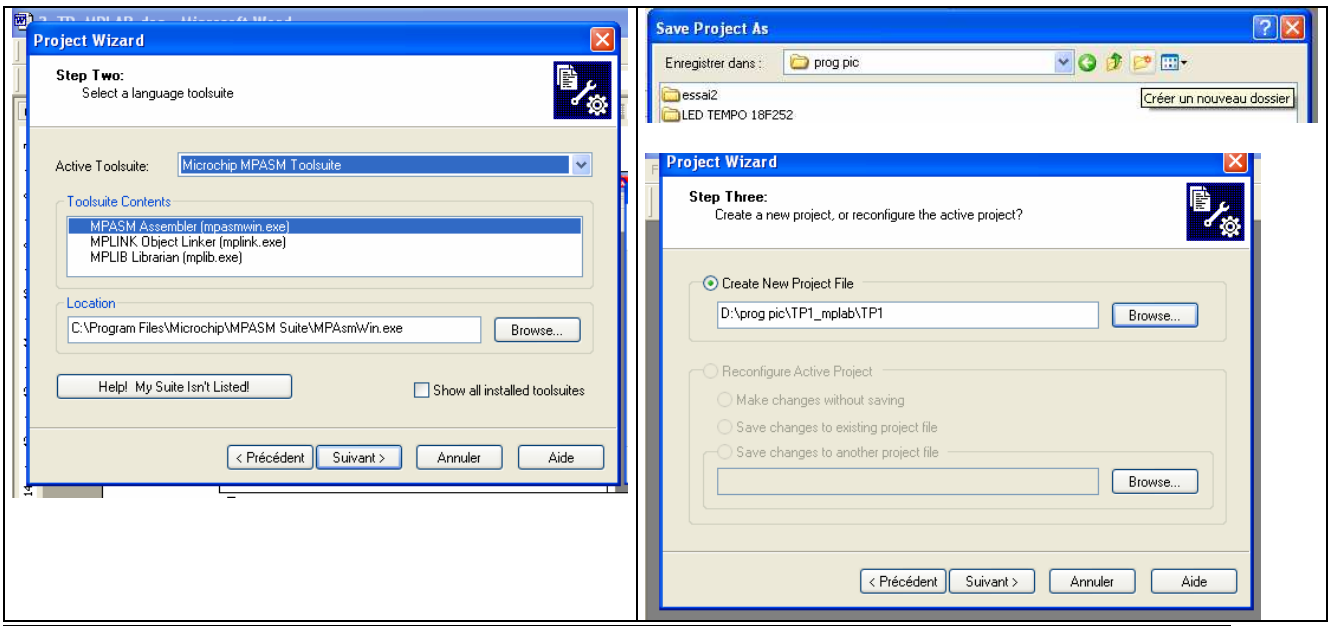
Microchip propose pour le développement de ses microcontrôleurs, un « Environnement de Développement Intégré » ou IDE, sous la forme d'un logiciel du nom de MPLAB, téléchargeable gratuitement sur le site de Microchip.
MPLAB permet, entre autres, d'éditer le code source de votre programme en assembleur, de le compiler, le simuler, de déboguer à l'aide du module ICD2 par exemple, et de programmer le circuit cible à l'aide ce même module.

1 Créer un projet

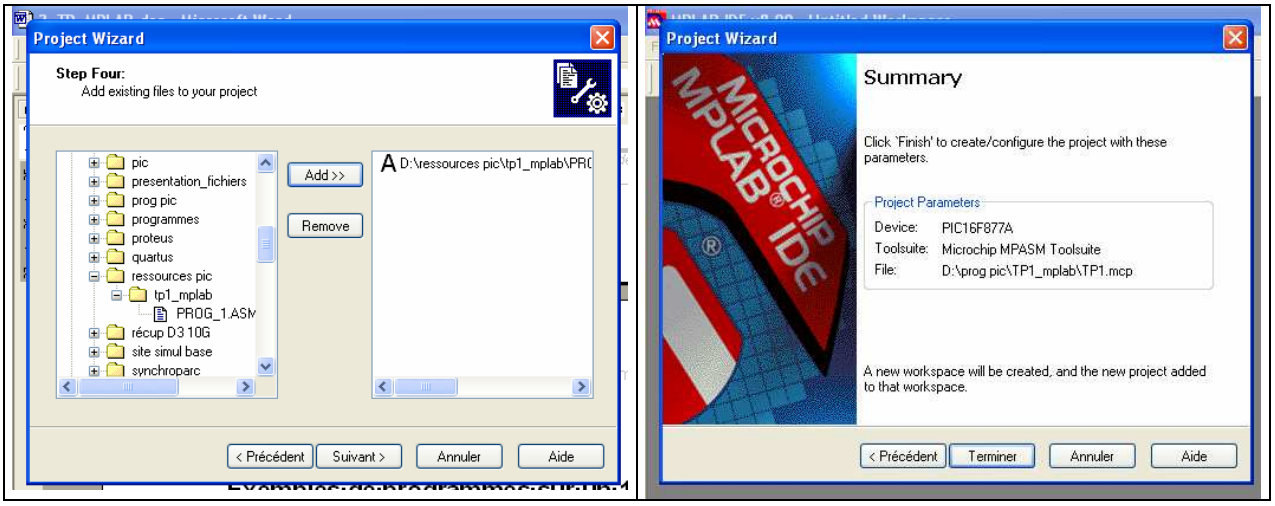
Après avoir lancé le programme par « **Démarrer -> Tous les Programmes -> Microchip MPLAB IDE** »,
créer un nouveau projet à l'aide de l'assistant :



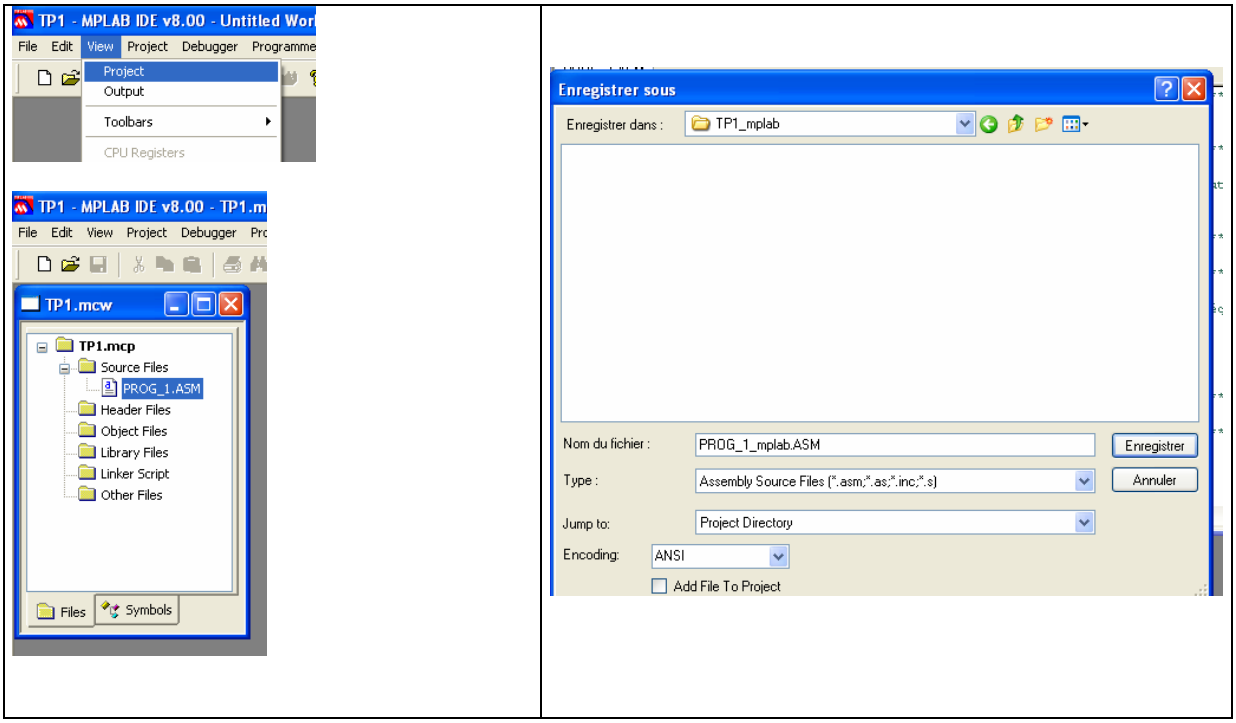
Choisir le circuit cible (PIC16F877 ou 16F877A) lors de la première phase, puis les outils lors de la seconde (laisser les options par défaut) et le répertoire de travail lors de la troisième phase. On pourra créer par exemple un répertoire TP1.



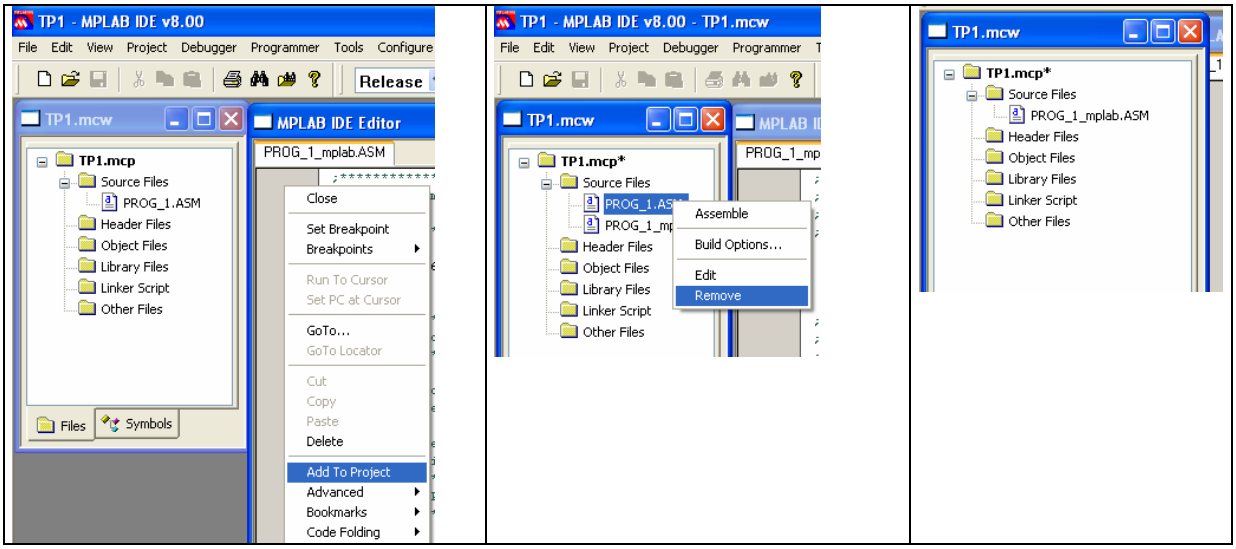
Ajouter ensuite le fichier « PROG_1.asm » qui se trouve dans le répertoire « Ressources / T¨P1 ».
Ce programme est celui étudié au chapitre précédent et est reproduit en annexe 1.
La dernière fenêtre récapitule les actions précédentes.



Ouvrir ensuite le répertoire que vous avez chargé (si la fenêtre de « navigation dans le projet » ne s'ouvre pas, faire « **View -> Project** ») et le sauvegarder, dans votre répertoire, sous un nom différent (Prog1_mplab.asm par exemple).



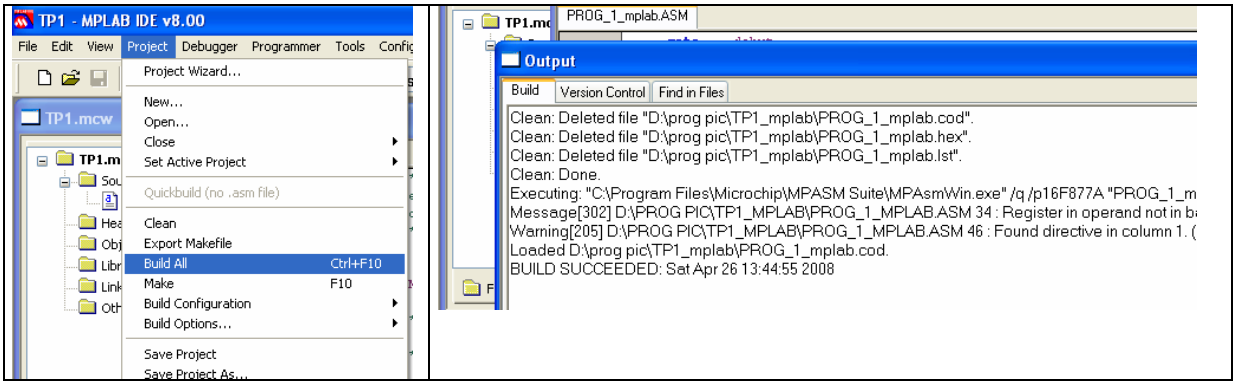
Par un clic droit dans le navigateur de projet sur le répertoire « **Sources Files** », accéder au menu contextuel, pour ajouter au projet le nouveau fichier que vous venez de sauvegarder.
Supprimer l'ancien du projet (clic droit sur le nom).



Par un double clic sur le nom du fichier restant, ouvrir celui-ci et analyser le code source.

2 Compiler

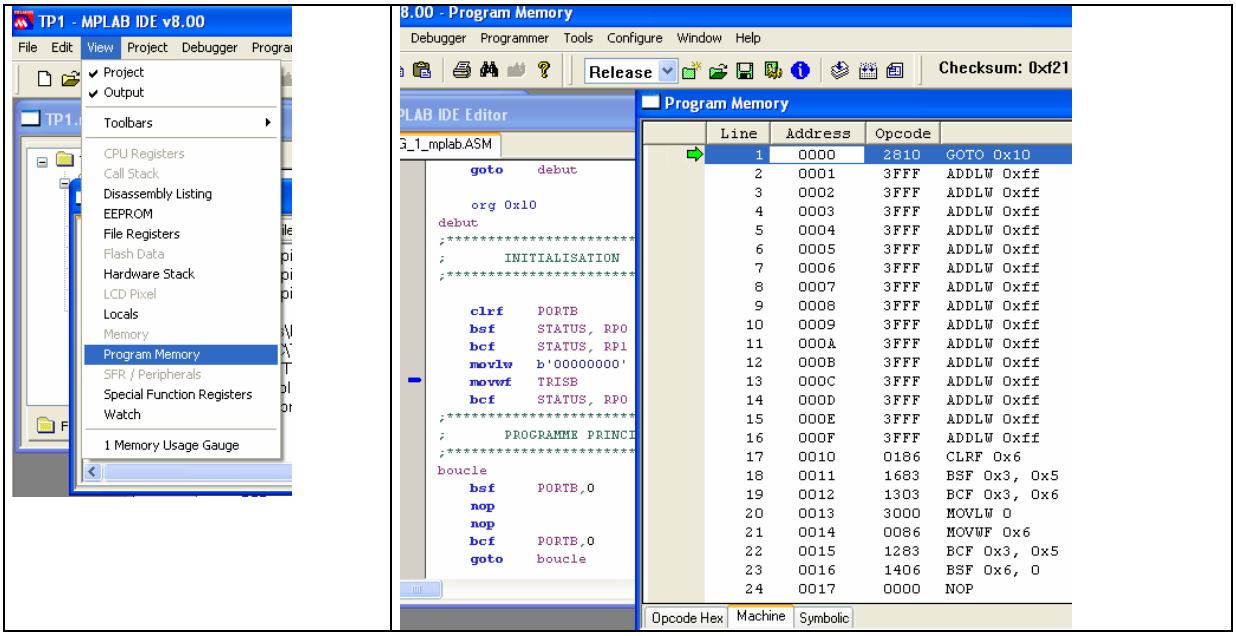
Compiler le fichier, la compilation devant se dérouler sans problèmes, comme l'indique la fenêtre « **Output** » de la figure suivante.



Si une erreur se produit lors de la compilation, un double clic sur la description de cette erreur dans la fenêtre « **Output** », ouvre le code source, le curseur se trouvant à l'endroit de l'erreur.

3 Vérifier la mémoire programme

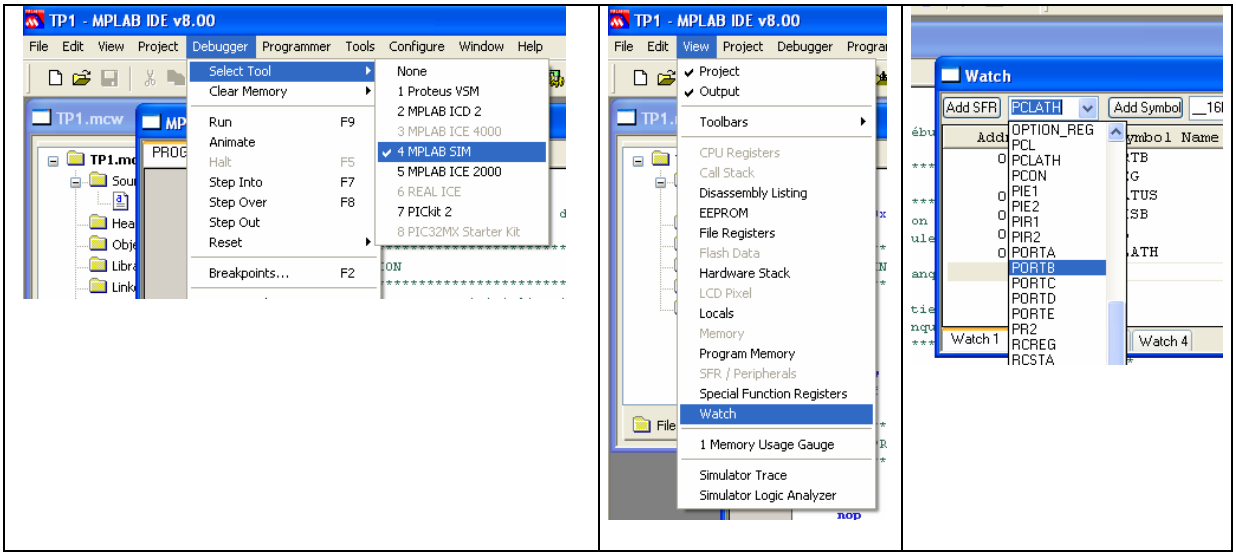
MPLAB permet de visualiser se qui va être écrit dans la mémoire programme.



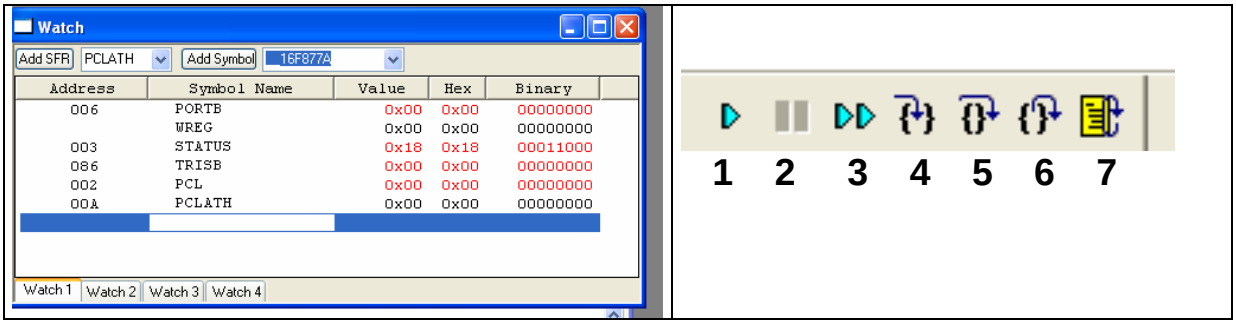
Vérifier que les instructions du code sources sont bien aux adresses prévues.
Remarque : sur un emplacement mémoire non utilisé, tous les bits sont à 1 ce qui correspond à l'instruction de mnémonique « ADDLW 0xff ».

4 Simuler

Choisir ensuite le simulateur de MPLAB comme outil de simulation, et ouvrir une fenêtre de visualisation.



Placer dans cette fenêtre tous les éléments intervenant dans notre programme comme indiqué ci-après :



Ajouter, par le menu contextuel de la barre du nom des colonnes (clic droit) une colonne « Hex » afin d'obtenir les valeurs en hexadécimal du contenu des registres.

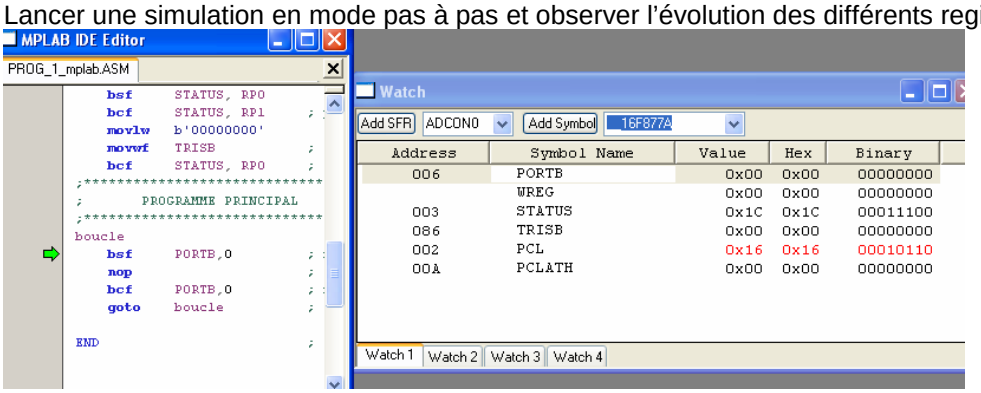
Une barre d'outil permet de commander le simulateur :

- icône 1 : mode « **Run** », la simulation est exécuté à grande vitesse, jusqu'à un point d'arrêt ou l'appui sur l'icône « **Halt** » ;
- icône 2 : « **Halt** », arrête le mode « Run » et « Animate » ;
- icône 3 : mode « **Animate** », exécute le programme lentement jusqu'à l'appui sur « Halt » ;
- icône 4 : mode « **Step Into** », exécute le programme pas à pas en passant dans tous les sous programmes ;
- icône 5 : mode « **Step Over** », exécute le programme pas à pas sans passer dans les sous programmes ;
- icône 6 : mode « **Step Out** », exécute le programme pas à pas en sortant des sous programmes ;
- icône 7 : mode « **Reset** », place le compteur programme à l'adresse 0x0000.

Remarques :

- si on souhaite initialiser la valeur d'un registre, il suffit de cliquer deux fois sur celle-ci, et entrer la nouvelle valeur ;
- il est possible, pour des programmes plus complexes, de définir jusqu'à 4 fenêtres de visualisation.

Lancer une simulation en mode pas à pas et observer l'évolution des différents registres.



On notera que la ligne pointée par la flèche verte dans le programme n'est pas la ligne en cours d'exécution, mais celle qui va être exécuté au cycle suivant, de même que PCL pointe l'adresse qui va être exécutée et non celle en cours d'exécution.

4.1 Une erreur classique…

On a pu constater que le programme précédent permettait de faire basculer la sortie PB0. De nombreuses erreurs de programmation avec les microcontrôleurs PIC sont dues à une écriture dans la mauvaise banque : nous aurions pu oublier par exemple de revenir en banque 0 dans notre programme après la phase d'initialisation du port B.
Observons ce qui ce passe alors en plaçant un « ; » devant la ligne suivante du programme source :

```
    bcf         STATUS, RP0         ; retour en banque 0
```

compiler, simuler et observer les différences.

5 Le mode débogage

Le mode débogage permet de simuler le pic en situation, tout en gardant le contrôle du microcontrôleur (mode pas à pas, observation de registres etc…).
Cette solution est moins onéreuse que l'émulateur, mais utilise certaines ressources du circuit cible :

- des ressources matérielles - bornes PGC et PGD (qui sont aussi RB6 et RB7 sur le PIC 16F877A) ainsi que Vpp/MCLR ;
- des ressources logicielles : implantation d'un programme en mémoire haute, utilisation de registre, de la pile.

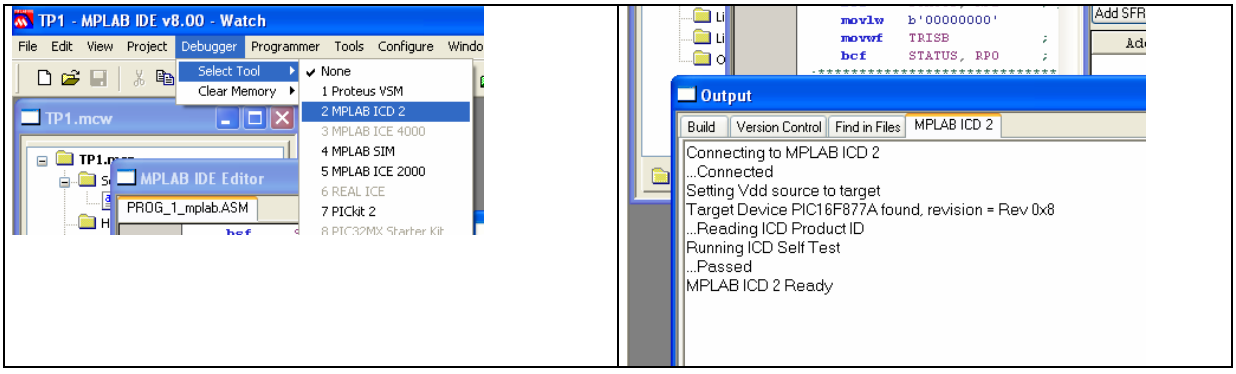
Le chien de garde, la protection contre la relecture, la protection de relecture de l'EEPROM doivent être de plus désactivés, ainsi que la programmation basse tension ; l'oscillateur doit avoir la configuration correspondant à la cible.

Si ce n'est pas encore fait :

- connecter l'ICD 2 au port USB du PC ;
- connecter la carte ¨PICDEM 2 PLUS à l'ICD2 ;
- alimenter la carte par le bloc secteur.

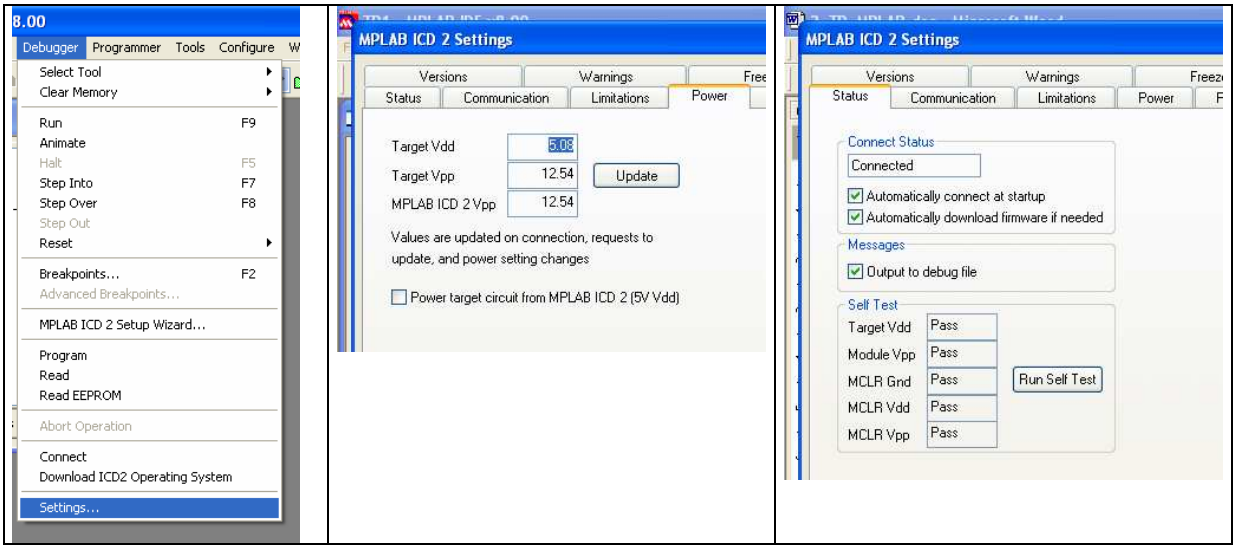
Il est possible que le système d'exploitation souhaite installer les pilotes de l'ICD2.
Le fichier demandé « **icd2w2kl.inf** » se trouve dans le répertoire **C:\Program Files \ `Microchip \ MPLAB IDE \ ICD2 \ Drivers** ».

Sélectionner dans un premier temps l'ICD2 comme débogueur, la fenêtre « Output » devant signaler qu'elle a bien reconnu ce module.

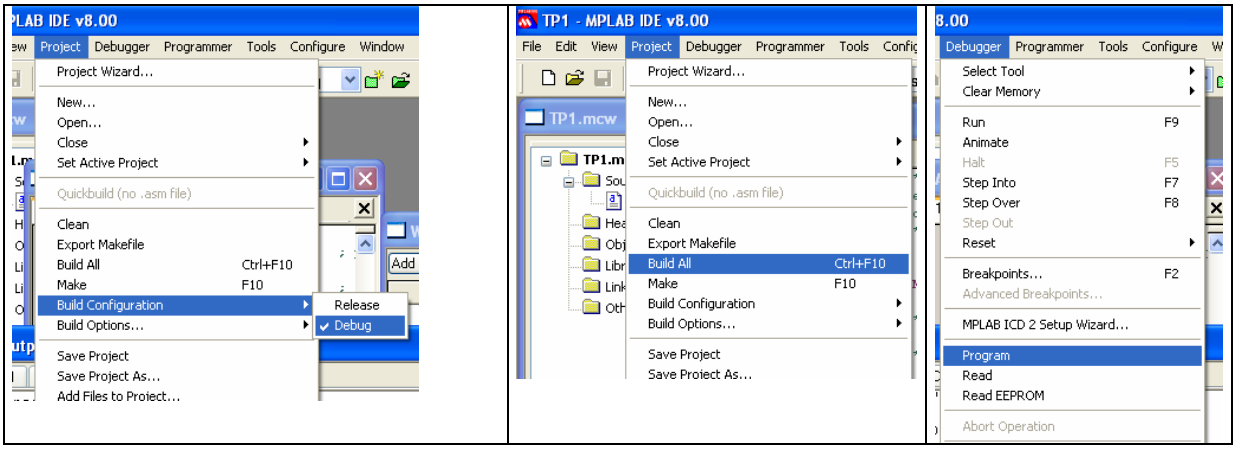


En cas de problème, on pourra vérifier que :

- le circuit installé sur la carte est bien celui déclaré au logiciel ;
- la connectique est correcte, la carte correctement alimentée ;
- les paramètres sont corrects, en particuliers en ce qui concerne l'alimentation de la carte (qui n'est pas alimentée depuis l'ICID2 dans notre cas).



Il faut ensuite préciser que la compilation à effectuer est destinée au mode « débogage » (un morceau de programme va être ajouté en partie haute de la mémoire programme, afin de contrôler le circuit cible).
Compiler ensuite, puis programmer (attention depuis le menu « Debugger » et non « Programmer ») le circuit cible.



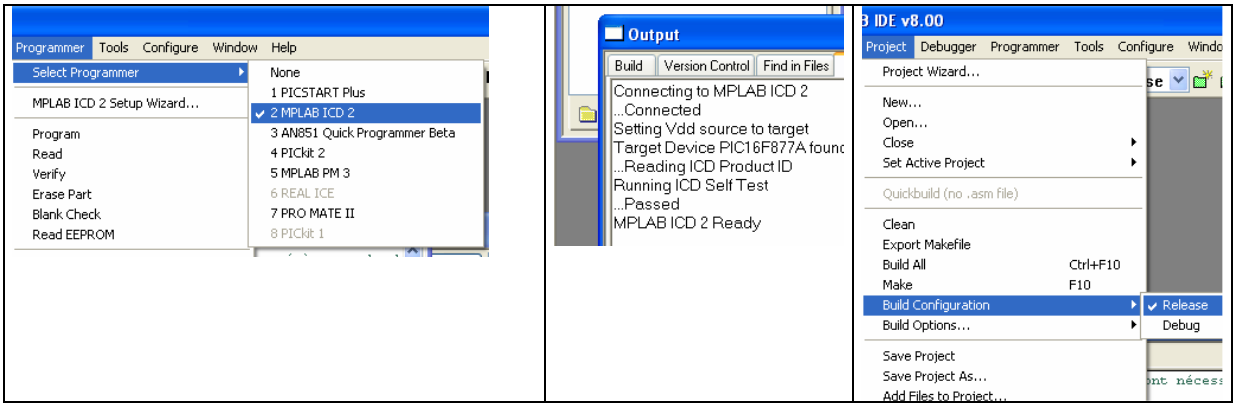
La démarche est ensuite identique à la simulation, mais c'est le circuit cible qui fonctionne réellement.

Vérifier en mode pas à pas que l'on a bien le fonctionnement attendu et que la DEL s'allume et s'éteint.

6 Programmer

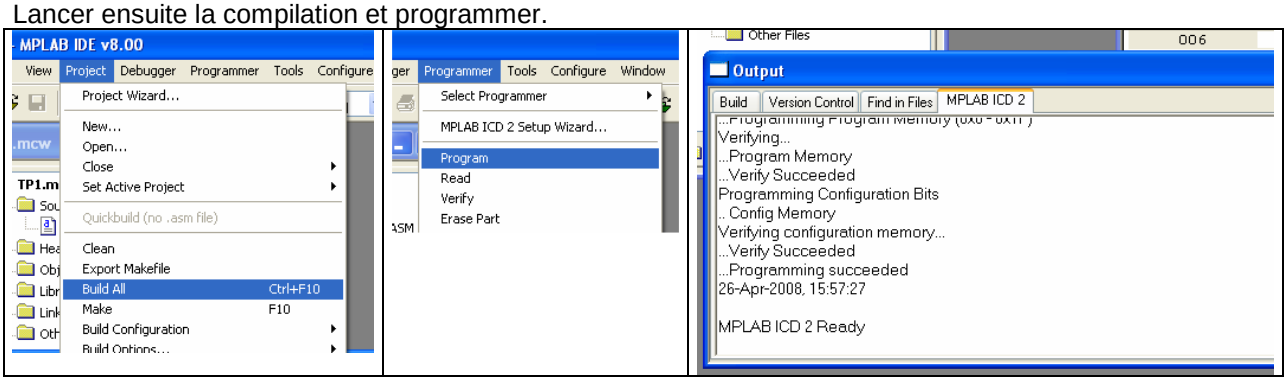
Pour programmer le circuit cible, il faut d'abord préciser au logiciel quel programmateur on va utiliser (ici l'ICD2), vérifier que le programmateur est bien détecté (fenêtre « **Output** »).
En cas de problème, on pourra vérifier que :

- le circuit installé sur la carte est bien celui déclaré au logiciel ;
- la connectique est correcte, la carte correctement alimentée ;
- les paramètres sont correctes dans « **Programmer -> Settings** », en particuliers en ce qui concerne l'alimentation de la carte (qui n'est pas alimentée depuis l'ICID2 dans notre cas) ;
- le pilote « **icd2w2kl.inf** » de l'ICD2 est installé (voir chapitre précédent).



On précisera également que la compilation doit se faire maintenant en mode « **Release** » et non « Debug »

Lancer ensuite la compilation et programmer.



Déconnecter le cordon RJ45 de la carte PICDEM 2 PLUS ; la DEL clignotant à fréquence élevée, elle semble allumée en permanence et il n'est pas possible de constater à l'œil le clignotement.
Si nous souhaitons voir clignoter la DEL, il va falloir diminuer la fréquence de commande.

7 Modification du programme

Afin de ralentir la fréquence de la sortie RB0 nous allons inclure dans notre programme une boucle de temporisation.
La version modifiée est disponible dans le dossier « **Ressources / TP1 mplab / Prog2.asm** ». On trouvera ce programme reproduit en annexe 2.

Ajouter ce programme au projet et supprimer l'ancien du projet (il reste malgré tout disponible dans notre répertoire).

Etudier ce nouveau programme.
On notera en particuliers :

- l'utilisation de la directive « **#DEFINE** » pour remplacer « PORTB,0 » par « DEL »
- la déclaration de variables en RAM par la directive « **EQU** » qui remplace T_INT et T_EXT dans le programme, respectivement par 0x20 et 0x21 comme nous l'avons vu pour le fichier « include » ; attention 0x20 et 0x21 sont les adresses de T_INT et T_EXT et non leur contenu.
- l'appel de sous programme pour l'initialisation du port et la temporisation, ce qui améliore la lisibilité de l'ensemble ;

Compiler le programme en mode « réalisation » par « **Project -> Build Configuration -> Release** » puis « **Ctrl F10** ».

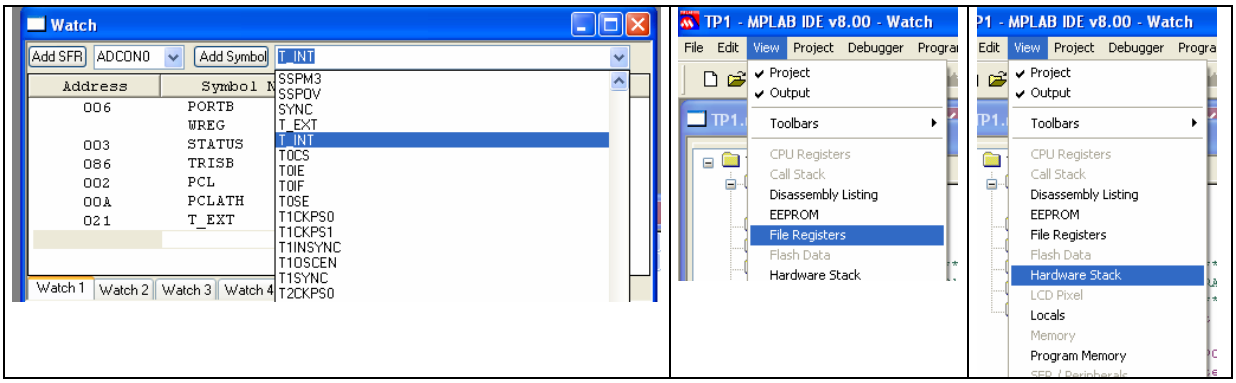
Programmer le composant et vérifier que le fonctionnement du microcontrôleur est bien celui attendu.

Revenons maintenant en mode simulateur, afin d'étudier quelques fonctionnalités avancées de celui-ci.

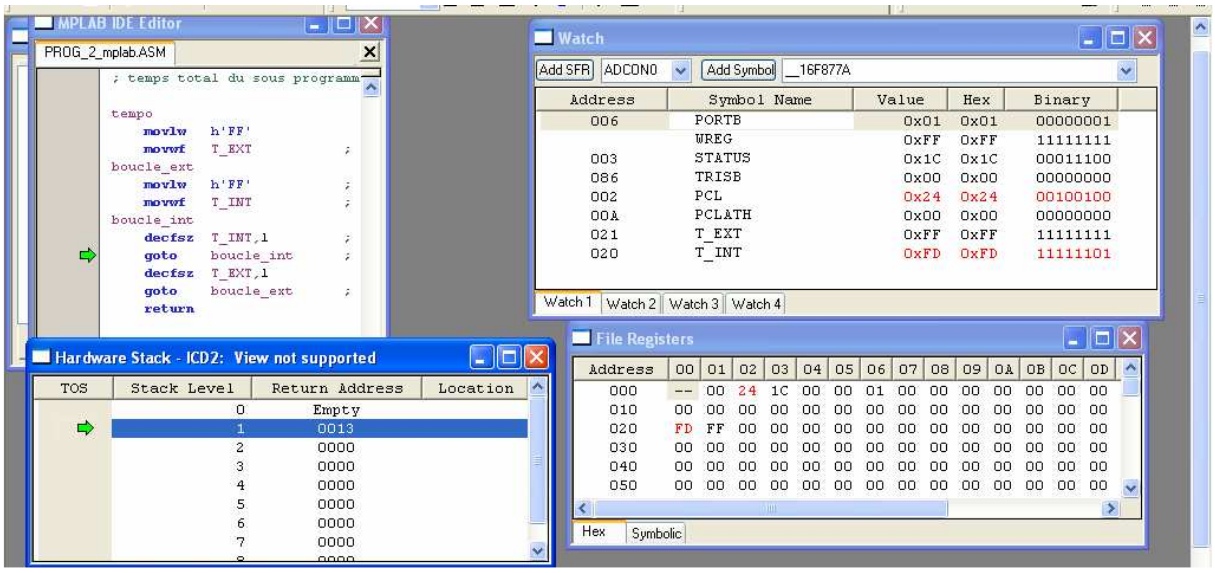
7.1 Les SFR et la pile

Préparer la simulation en ajoutant T_INT et T_EXT dans la fenêtre « **Watch** » comme nous l'avons vu précédemment.

Ouvrir ensuite des fenêtres représentant le plan mémoire des fichiers et données, et enfin le contenu de la pile.



Réinitialiser le programme par un « **Reset** » puis jouer sur les options « **Step Into** », « **Step Over** » et « **Step Out** ».

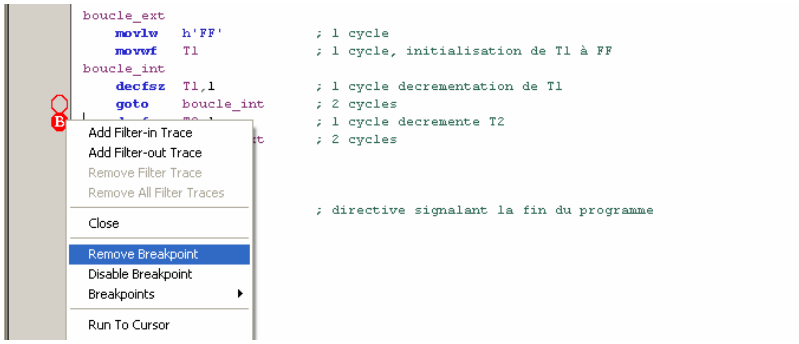


Observer dans un premier temps l'évolution de T_INT et T_EXT, à la fois dans la fenêtre « **Watch** » et dans le plan mémoire des registres « **File Registers** », ainsi que le contenu du compteur de programme PC (adresse 02 du plan mémoire) en vérifiant qu'il est identique à PCL pour ce petit programme.

Observer ensuite l'évolution du contenu de la pile et de PC lors des appels de sous programmes.

7.2 Les points d'arrêts

Lorsque le simulateur fonctionne en mode « **Run** », il est possible de placer des points d'arrêts dans le programme en double cliquant sur n'importe quelle ligne exécutable. La simulation s'arrêtera alors sur ce point lorsqu'elle y passera (si elle y passe !). Cet outil est très pratique pour vérifier qu'un programme passe bien par une sous-routine.



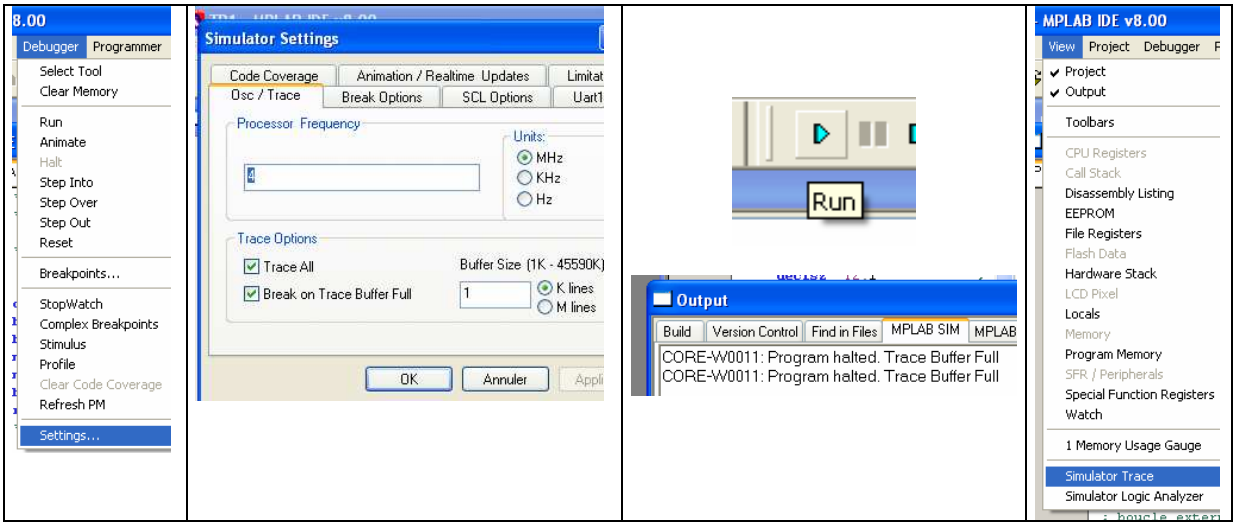
Par le menu contextuel (clic droit), il est ensuite possible d'enlever complètement le point d'arrêt, qui disparaît alors (un double clic a le même effet), ou de le désactiver, il reste alors un cercle rouge.

Tester ce mode de fonctionnement en plaçant un point d'arrêt dans la temporisation et lancer le mode « **Run** ».

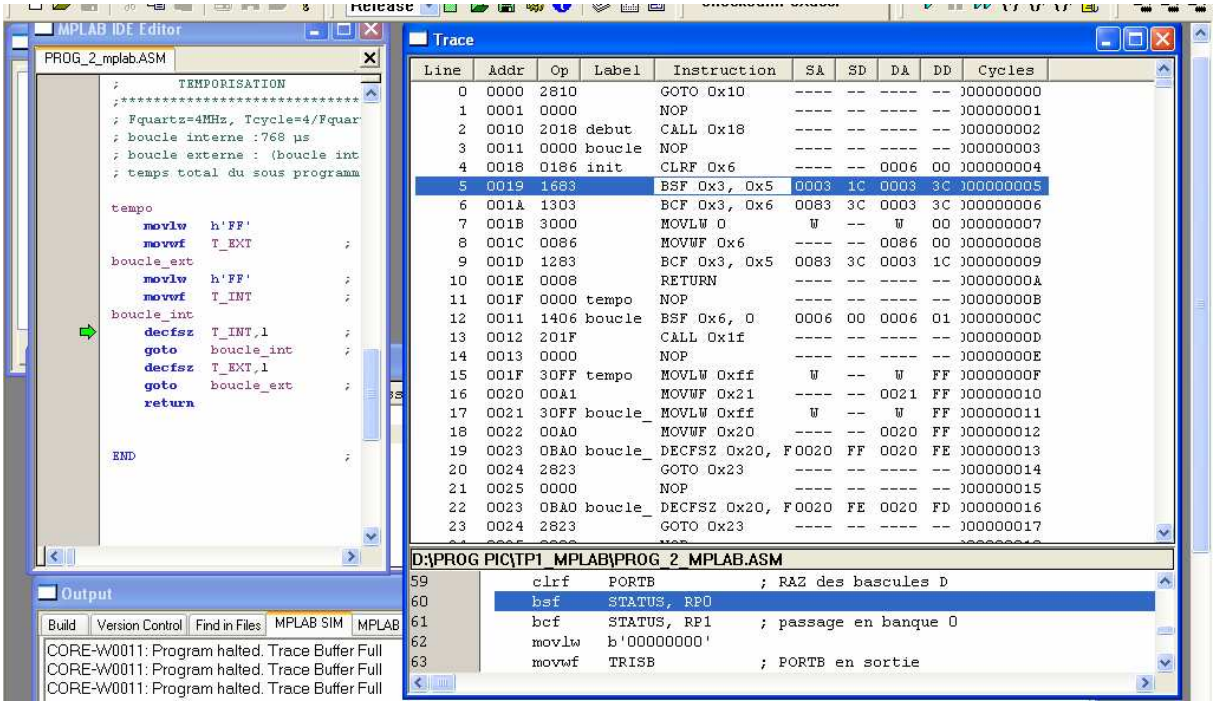
Supprimer les points d'arrêts avant de passer au paragraphe suivant.

7.3 Le mode Trace

Ce mode permet d'enregistrer les actions du programme et le contenu des registres.



Dans le fenêtre de configuration de l'oscillateur, cocher l'option « **Trace All** » et choisir l'enregistrement jusqu'à ce que le tampon soit plein, à 1 K lignes (ce qui est suffisant pour notre programme).
Lancer une simulation, qui va être arrêté au bout de 1024 lignes d'exécution.
Ouvrir la fenêtre « **Simulator Trace** ».

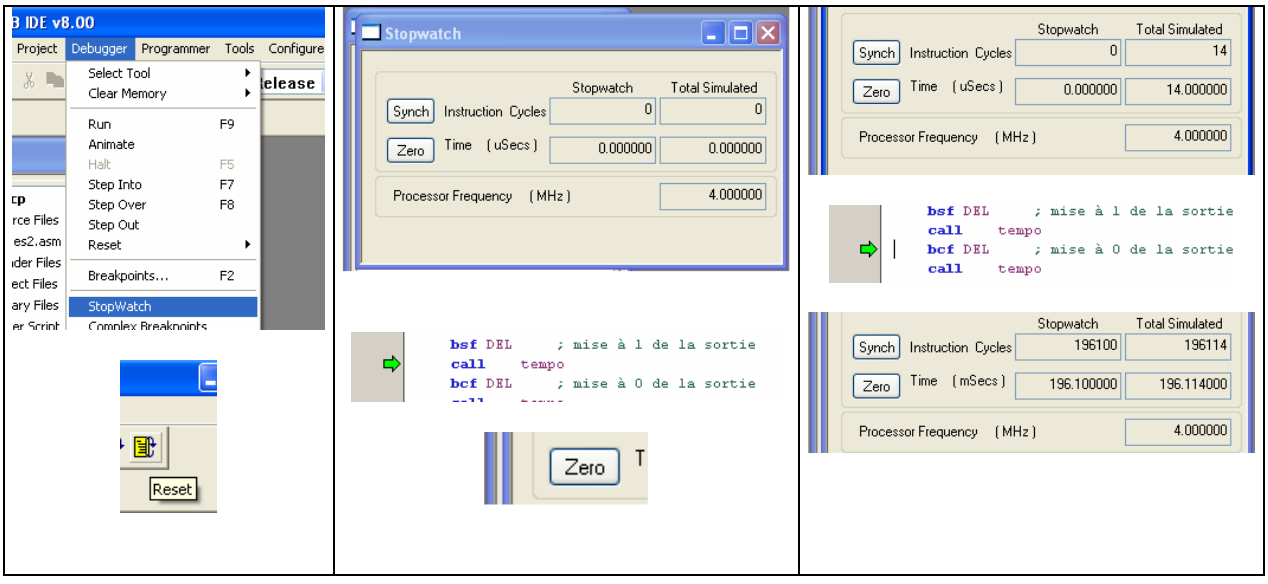


Un double clic sur une ligne de la fenêtre « Trace » ouvre une sous fenêtre avec la ligne de programme correspondant.
Dans la fenêtre « Trace » on peut lire :

- le numéro de la ligne de programme concernée ;
- l'adresse « **Addr** » de la mémoire programme à laquelle nous sommes ;
- le code « **Op** » de l'opération ;
- l'étiquette s'il y en a une ;
- l'instruction
- l'adresse du registre sources « **SA** » ;
- la donnée « **SD** » contenue dans ce registre avant l'opération ;
- l'adresse « **DA** » du registre destination ;
- la donnée « **DD** » contenue dans le registre destination après l'opération.

7.4 Mesure de temps

Il est également de mesurer le temps entre deux instructions avec la fenêtre « **Stopwatch** ».



- vérifier dans un premier temps que le simulateur est bien paramètré pour un quartz de 4 MHz (« **Debugger -> Settings -> Osc / Trace** ») ;
- **ouvrir la fenêtre « Stopwatch » par le menu «** Debugger » ;
- réinitialiser la simulation, ce qui réinitialise les valeurs de la fenêtre ;
- placer le curseur dans le programme, à l'endroit où l'on souhaite démarrer la mesure de temps, puis par le menu contextuel (clic droit), faire « **Run to the cursor** » ; la fenêtre « Stopwatch » affiche alors le nombre de cycle d'instructions écoulées depuis le reset et le temps passé (dans notre cas 1 µs par cycle) ;
- initialiser la fenêtre « Stopwatch » par « **Zero** » ;
- placer le curseur dans le programme à l'endroit où l'on souhaite terminer la mesure, puis amener la simulation jusqu'à cette ligne ;
- la fenêtre « Stopwatch » affiche alors dans la première colonne la durée depuis le « Zero » (c'est à dire la durée de notre temporisation), ainsi que la durée depuis le début du programme dans la colonne de droite ;
- en cliquant sur « **Synch** » il est possible de resynchroniser la première colonne sur la seconde.

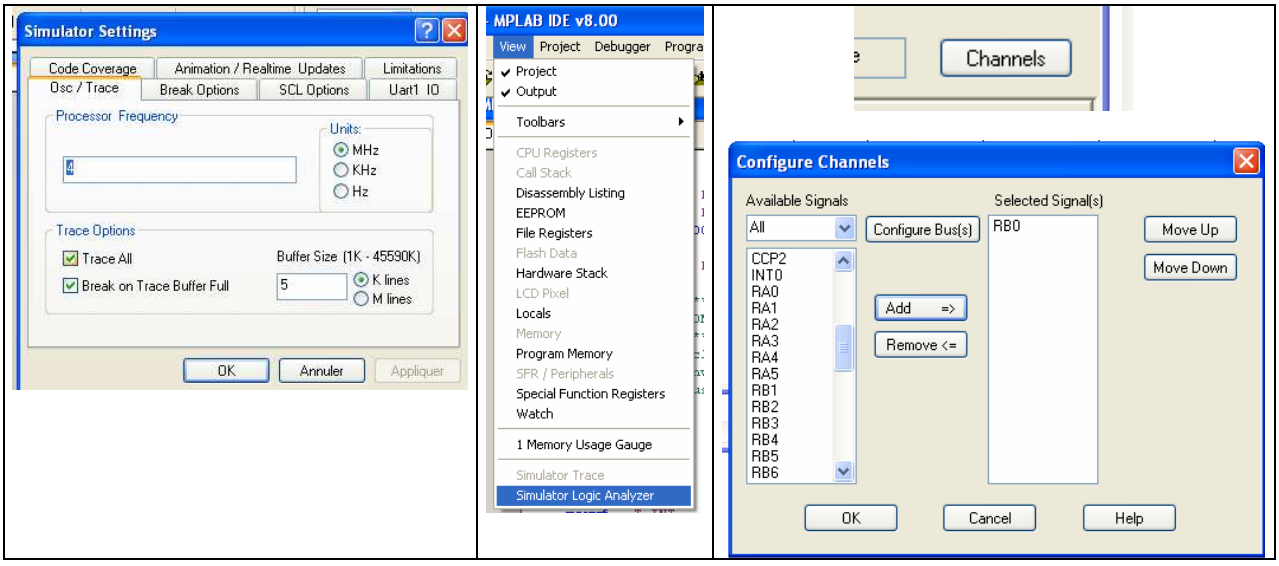
7.5 Analyseur logique

Il est possible de visualiser les signaux sous forme de chronogrammes.

Pour éviter une simulation trop longue, modifier le programme pour que la fréquence de RB0 soit environ 1,5 ms puis compiler de nouveau.

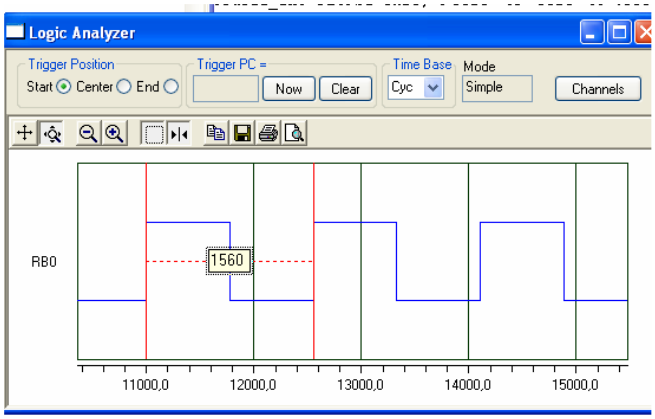
Vérifier que la configuration du simulateur correspond bien à une fréquence de quartz de 4 MHz et imposer une taille de tampon de 5 K lignes.

Ouvrir l'analyseur logique et ajouter le signal RB0 dans la sélection.



Lancer la simulation en mode « **Run** » ; elle s'arrête dès que le tampon est plein.

A l'aide des curseurs, vérifier que la période est bien celle attendue, l'affichage se faisant en « temps de cycle », qui vaut un quart de la période du quartz d'horloge, soit 1 µs.



7.6 Simulation d'évènements externes

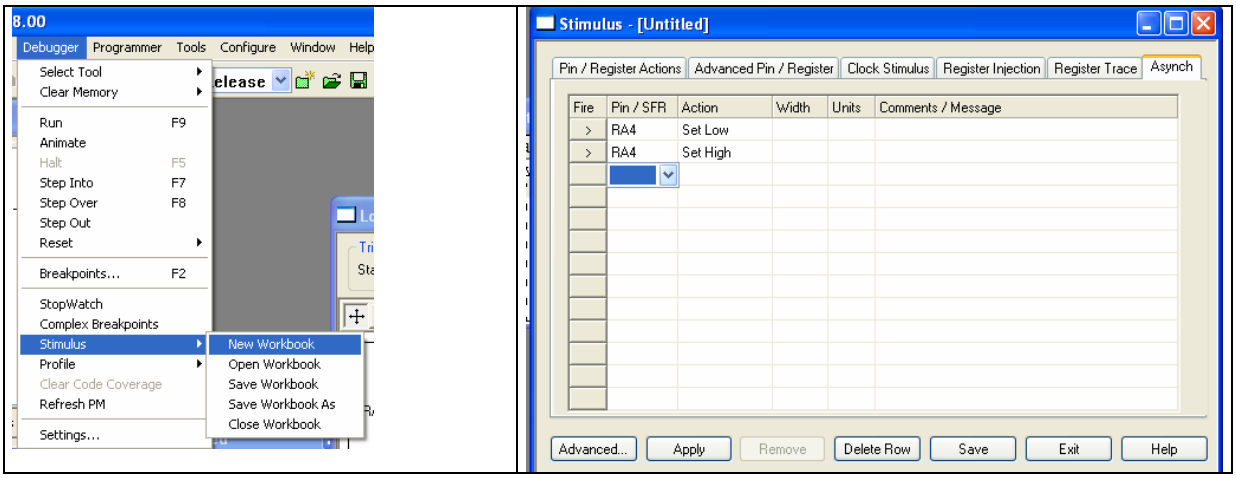
On souhaite maintenant que la DEL ne clignote que si on appui sur le bouton poussoir S2 de la carte, ce qui aura pour effet de mettre à 0 le bit 4 du port A.

Modifier le programme (on trouvera un exemple de solution en annexe 4) :

- en déclarant une variable BP liée au bit 4 du port A par « **#DEFINE** » ;
- en configurant en entrée ce bit (consulter la documentation) ;
- en réalisant le test de ce bit par une instruction du type « btfsc » ou « btfss ».

Introduire un « **stimuli** » et sur l'onglet « **Asynch** » de la fenêtre qui s'ouvre (notre événement sera à priori asynchrone) :

- par un clic gauche dans la colonne « **Pin/SFR** » on accède par un menu déroulant aux différentes entrées ;
- par un clic gauche dans la colonne « **Action** », on choisit l'action liée à cette entrée ;
- il suffit ensuite, au moment de la simulation, de cliquer sur la colonne « **Fire** » pour déclencher l'action correspondante.

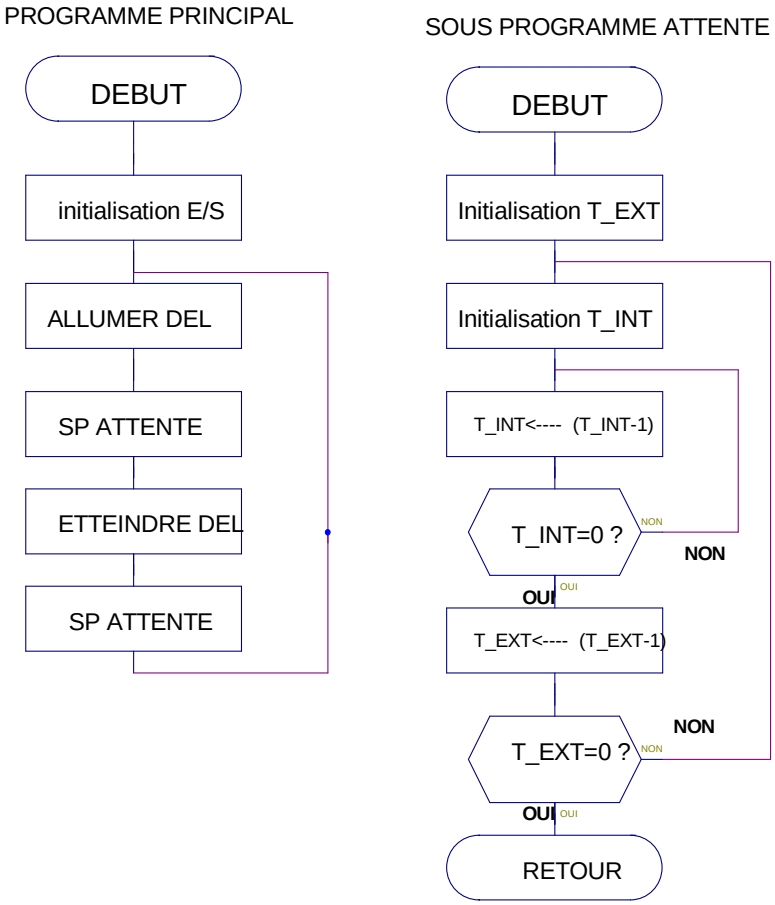


Simuler votre et programme et le tester sur le microcontrôleur une fois qu'il est au point.

Annexe 1 : Premier programme

```
;*******************************************************************************
; Ce programme génère une horloge en RB0
; de fréquence 24 fois plus faible que celle du quartz.          *
; il utilise la carte PicDem2Plus équipée d'un PIC16F877A        *
;*******************************************************************************

        LIST P=16F877A              ; directive qui définit le processeur utilisé
        #include <P16F877A.INC>     ; fichier de définition des constantes
;*******************************************************************************
;       BITS DE CONFIGURATION
;*******************************************************************************
        __CONFIG  _HS_OSC & _WDT_OFF & _CP_OFF & _CPD_OFF & _LVP_OFF

; _XS_OSC l'oscillateur est configuré en oscillateur à quartz haute fréquence
; _WDT_OFF le watchdog est désactivé
; _CP_OFF le code de protection de la mémoire programme est désactivé
; _CPD_OFF le code de protection de la mémoire EEPROM est désactivé
; _LVP_OFF la programmation basse tension est désactivée
; ces opérations sont nécessaires pour fonctionner en mode "debug"
;*******************************************************************************
;       DEMARRAGE SUR RESET                                                    *
;*******************************************************************************
        org     0x0             ; Adresse de départ après reset
        goto    debut

        org     0x10            ; adresse de début du programme
debut
;*******************************************************************************
;       INITIALISATION                                  *
;*******************************************************************************

        ; initialisation du PORTB en sortie (voir datasheet)


        bcf     STATUS, RP0
        bcf     STATUS, RP1         ; passage en banque 0
        clrf    PORTB               ; RAZ des bascules D
        bsf     STATUS, RP0         ; passage en banque 1
        movlw   b'00000000'
        movwf   TRISB               ; PORTB en sortie
        bcf     STATUS, RP0         ; retour en banque 0

;*******************************************************************************
;       PROGRAMME PRINCIPAL                             *
;*******************************************************************************
boucle
        bsf     PORTB,0             ; mise à 1 de la sortie
        nop                         ; 2 temps morts pour compenser le saut
        nop
        bcf     PORTB,0             ; mise à 0 de la sortie
        goto    boucle              ; rebouclage

END                                 ; directive signalant la fin du programme
;*******************************************************************************
```

Annexe 2 : Second programme




```
;*******************************************************************************
; Ce programme fait clignoter la DEL de la sortie RB0 sur la carte PICDEM2PLUS
; à une fréquence de 2,5 Hz                *                                    *
;*******************************************************************************
;
        LIST P=16F877A                      ; directive qui définit le processeur utilisé
        #include <P16F877A.INC>             ; fichier de définition des constantes
;*******************************************************************************
;                   BITS DE CONFIGURATION
;*******************************************************************************
;
        __CONFIG  _HS_OSC & _WDT_OFF & _CP_OFF & _CPD_OFF & _LVP_OFF

; _XS_OSC l'oscillateur est configuré en oscillateur à quartz haute fréquence
; _WDT_OFF le watchdog est désactivé
; _CP_OFF le code de protection de la mémoire programme est désactivé
; _CPD_OFF le code de protection de la mémoire EEPROM est désactivé
; LVP_OFF la programmation basse tension est désactivée
; ces opérations sont nécessaires pour fonctionner en mode "debug"
;*******************************************************************************
;                   DEFINITIIONS                                                *
;*******************************************************************************
;
#DEFINE DEL PORTB,0                 ;la DEL sur le port B, bit 0
;*******************************************************************************
;                   DECLARATION de VARIABLES                                    *
;*******************************************************************************
;
T_INT           EQU     0x20    ; Variable de temporisation à l'adresse 0x20
T_EXT           EQU     0x21    ; Variable de temporisation à l'adresse 0x21
```

```
;*******************************************************************************
;                   DEMARRAGE SUR RESET                                       *
;*******************************************************************************
        org     0x0             ; Adresse de départ après reset
        goto    debut

        org     0x10            ; adresse de début du programme

debut
;*******************************************************************************
;                   PROGRAMME PRINCIPAL                                       *
;*******************************************************************************
        call init               ; l'initialisation du port se fait par sous programme
boucle
        bsf     DEL             ; mise à 1 de la sortie
        call    tempo
        bcf     DEL             ; mise à 0 de la sortie
        call    tempo
        goto    boucle          ; rebouclage
;*******************************************************************************
;                   SOUS PROGRAMMES                                           *
;*******************************************************************************
;*******************************************************************************
;                   INITIALISATION                                            *
;*******************************************************************************
; initialisation du PORTB en sortie (voir datasheet)
init
        bcf     STATUS, RP0
        bcf     STATUS, RP1         ; passage en banque 0
        clrf    PORTB               ; RAZ des bascules D
        bsf     STATUS, RP0         ; passage en banque 1
        movlw   b'00000000'
        movwf   TRISB               ; PORTB en sortie
        bcf     STATUS, RP0         ; retour en banque 0
        return
;*******************************************************************************
;                   TEMPORISATION                                             *
;*******************************************************************************
; Fquartz=4MHz, Tcycle=4/Fquartz=1µs
; boucle interne :environ 3 µs .255= 765 µs
; temps total du sous programme : environ 255.765 µs, environ 200 ms

tempo
        movlw   h'FF'
        movwf   T_EXT               ; initialisation de T_EXT à FF
boucle_ext
        movlw   h'FF'               ; 1 cycle
        movwf   T_INT               ; 1 cycle, initialisation de T_INT à FF
boucle_int
        decfsz  T_INT,1             ; 1 ou 2 cycles, décrémentation de T_INT
        goto    boucle_int          ; 2 cycles
        decfsz  T_EXT,1             ; 1 ou 2 cycles, décrémentation T_EXT
        goto    boucle_ext          ; 2 cycles
        return
END                     ; directive signalant la fin du programme
```

Annexe 3 : les ports A et B

Les ports d'entrées sorties sont configurables bit à bit en entrée ou en sortie par le registre TRIS du même nom que le port considéré : l'écriture d'un NL0 sur le bit x du registre TRISY configure le bit x du PORTY en sortie (« 0 » comme « Output », c'est la valeur par défaut après une réinitialisation) tandis qu'un NL1 le configure en entrée (« 1 » comme « Input »).
Particularité du port A :

- il s'agit d'un port 5 bits ;
- les bornes 0 à 3 sont multiplexées avec de fonctionnalités du CAN.
- la sortie 4 est à drain ouvert
- la borne 4 est multiplexée avec la fonction « entrée d'horloge du timer 0 » et sortie du « comparateur » ;
- la borne 5 est multiplexée avec une fonction entrée du CAN, entré pour la liaison série synchrone, et sortie du comparateur.

Particularité du port B :

- il s'agit d'un port 8 bits ;
- sur chaque borne il est possible de configurer par logiciel une résistance de tirage vers le haut ;
- le bornes 3, 6 et 7 sont multiplexée pour la programmation ;
- les entrées 4 à 7 peuvent servir d'entrée d'interruption ;

Pour configurer une borne en entrés sortie numérique, le constructeur préconise :

- de réinitialiser le port ;
- de configurer les bornes en entrées numérique s'il y a un multiplexage avec une fonction analogique ;
- d'écrire dans le registre TRIS ;

Ci-après un exemple pour le port A ;

```
BCF     STATUS, RP0   ;
BCF     STATUS, RP1   ; Bank0
CLRF    PORTA         ; Initialize PORTA by
                      ; clearing output
                      ; data latches
BSF     STATUS, RP0   ; Select Bank 1
MOVLW   0x06          ; Configure all pins
MOVWF   ADCON1        ; as digital inputs
MOVLW   0xCF          ; Value used to
                      ; initialize data
                      ; direction
MOVWF   TRISA         ; Set RA<3:0> as inputs
                      ; RA<5:4> as outputs
                      ; TRISA<7:6>are always
                      ; read as '0'.
```

Annexe 4 :exemple de solution

Exemple de solution pour la partie « simulation d'événements externes » ;
Seules les parties à changer par rapport au programme précédent ont été reproduites.

```
;*******************************************************************************
;                   DEFINITIIONS                                              *
;*******************************************************************************
#DEFINE DEL PORTB,0                 ;la DEL sur le port B, bit 0
#DEFINE BP PORTA,4                  ;la DEL sur le port B, bit 0
;*******************************************************************************
;                   PROGRAMME PRINCIPAL                                       *
;*******************************************************************************
        call init
boucle
        btfsc   BP                  ; test du bouton poussoir
        goto    boucle              ; bouclage sur le test si BP=1
        bsf     DEL                 ; mise à 1 de la sortie
        call    tempo
        nop                         ; 2 temps morts pour compenser le goto
        nop
        bcf     DEL                 ; mise à 0 de la sortie
        call    tempo
        goto    boucle              ; rebouclage

;*******************************************************************************
;                   SOUS PROGRAMMES                                           *
;*******************************************************************************
;*******************************************************************************
;                   INITIALISATION                                            *
;*******************************************************************************
                                    ; initialisation du PORTB en sortie (voir datasheet)
                                    ; initialisation du bit 4 du PORTA en entrée (voir datasheet)
init
        bcf      STATUS, RP0
        bcf     STATUS, RP1         ; passage en banque 0
        clrf    PORTB               ; RAZ des bascules D du port B
        clrf    PORTA               ; RAZ des bascules D du port A
        bsf     STATUS, RP0         ; passage en banque 1
        movlw   b'00000000'
        movwf   TRISB               ; PORTB en sortie
        movlw   0x06
        movwf   ADCON1              ; PORTA configuré en numérique
        movlw   b'00010000'
        movwf   TRISA               ; PORTA bit 4 en entrée
        bcf     STATUS, RP0         ; retour en banque 0

        return
```