

Prise en main de MPLAB 6.40



Équipe de formation sur les micro contrôleurs PIC
Louis ANDRE
Lycée Philippe de Girard
84000 Avignon
Académie d'Aix-Marseille
louis.andre@laposte.net





Sommaire

1 - Introduction.....	4
1 Présentation.....	4
2 Installation.....	4
3 Lancement de MPLAB IDE.....	4
1.Créer un projet.....	5
1 Lancement du « Wizard ».....	5
2 Choix du processeur.....	5
3 Sélectionnez les outils.....	6
4 Nommez le projet.....	6
5 Insérez le fichier source.....	7
6 Validez le projet.....	7
3 - Écrire un programme et construire un projet.....	8
1 Créez votre fichier source.....	8
2 Ajouter le fichier au projet.....	8
3 Construire le projet.....	9
4 - Simuler le comportement d'un programme.....	10
1 Configurez le simulateur.....	10
2 Exécutez le programme.....	10
3 Visualisez la fenêtre d'observation.....	11
4 Exécution et visualisation.....	12
5 - Utiliser les points d'arrêt et le mode Trace.....	12
1 Utilisation d'un point d'arrêt.....	12
2 Utilisation du mode Trace.....	12
6 - Programmer un composant.....	14
1 Sélectionnez le programmeur.....	14
2 Spécifiez les bits de configurations.....	14
3 Programmez le composant.....	15
7 - Utiliser les fonctions avancées du simulateur.....	15
1 Configurez le simulateur.....	15
2 Création de stimulus.....	16
8 - Utiliser ICD2.....	17
1 Configurez l'ICD2.....	17



1 - Introduction

1 Présentation

MPLAB 6 est un Environnement de Développement Intégré (IDE) qui permet le développement logiciel des micro contrôleurs 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 "linqueur" et un gestionnaire de bibliothèques 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) 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).
- ◆ De programmer les composants grâce à PICSTART Plus (unité) ou PROMATE II (série)

La version 6.40 sortie fin 2003 supporte la totalité des processeurs Flash..

Installation

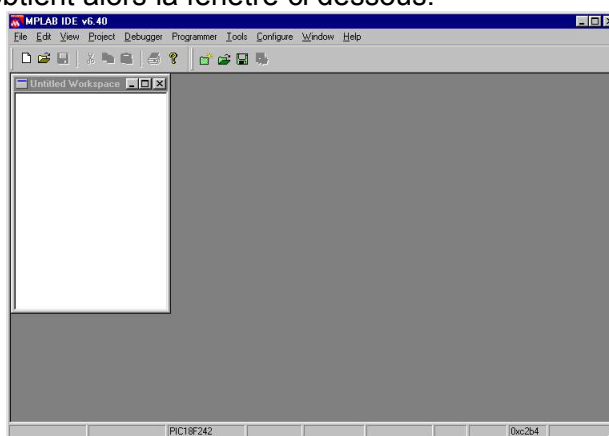
MPLAB 6.40 nécessite l'utilisation de Microsoft Windows 98 SE ou suivant, d'avoir 45 Mo de libre sur le disque dur et 64 (ou 128 recommandé) Mo de RAM.

MPLAB 6.40 peut être télé chargé depuis le site www.microchip.com ou installé à partir d'un CD Rom fourni lors de l'achat d'un outil de développement.

Si vous disposez de l'ICD2 il vous faut installer le driver; pour cela connectez l'ICD au port USB de votre PC, Windows le détecte et lance le programme d'installation; il faut lui indiquer le chemin ou trouver le driver: <chemin de MPLAB>\driversxx\ICD2 USB.

2 Lancement de MPLAB IDE

Pour démarrer Mplab IDE faire *Démarrer > Programme > Microchip MPLAB IDE > MPLAB IDE*. On obtient alors la fenêtre ci dessous.



1. Créer un projet

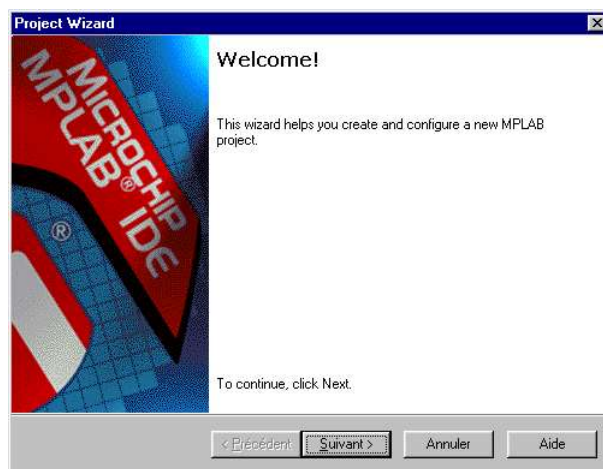
La meilleure manière de développer une application est d'utiliser un projet; celui ci contient tous les fichiers nécessaires pour construire une application. L'espace de travail d'un projet mémorise l'emplacement des différentes fenêtres, les outils utilisés etc.

La sortie du projet est un fichier .HEX au format INHX32 qui sera programmé dans la mémoire du processeur cible.

Depuis la version 6.30 MPLAB fournit un « MPLAB Project Wizard » qui vous guide pour créer un nouveau projet.

1 Lancement du « Wizard »

Démarrez le « Project Wizard » en faisant *Project>Project Wizard*, vous obtenez alors la fenêtre ci dessous.



Pour continuer cliquez sur Suivant.

2 Choix du processeur

Sélectionner le processeur utilisé à l'aide de la liste déroulante. Dans l'exemple utilisé ici sélectionner le PIC16F877.



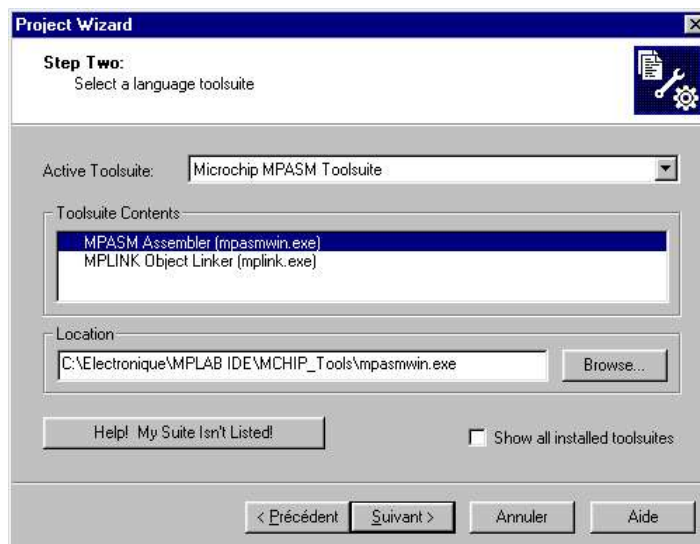
Pour continuer cliquez sur Suivant.



3 Sélectionnez les outils

La fenêtre suivante vous permet de choisir la suite d'outils logiciels que vous souhaitez utiliser.

Dans le menu *Active Toolsuite* sélectionnez *Microchip MPASM Toolsuite*. Vous devez alors obtenir la fenêtre ci dessous.



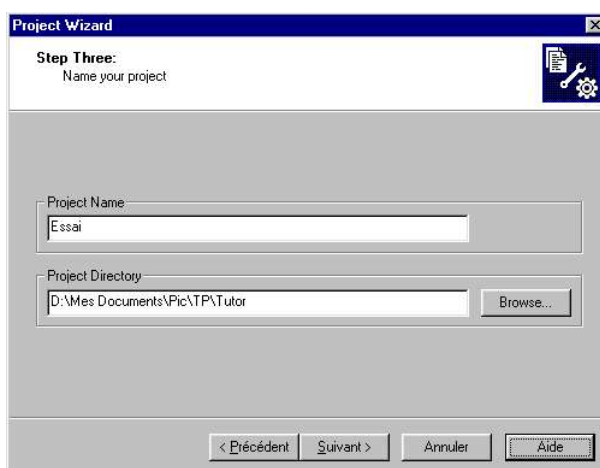
Si une croix rouge apparaît devant MPASM et MPLINK vous devez indiquer le chemin où se trouvent les deux logiciels à l'aide du bouton *Browse*.

Pour continuer cliquez sur *Suivant*.

4 Nommez le projet

La fenêtre suivante vous permet de donner un nom à votre projet et d'indiquer dans quel répertoire vous voulez le ranger.

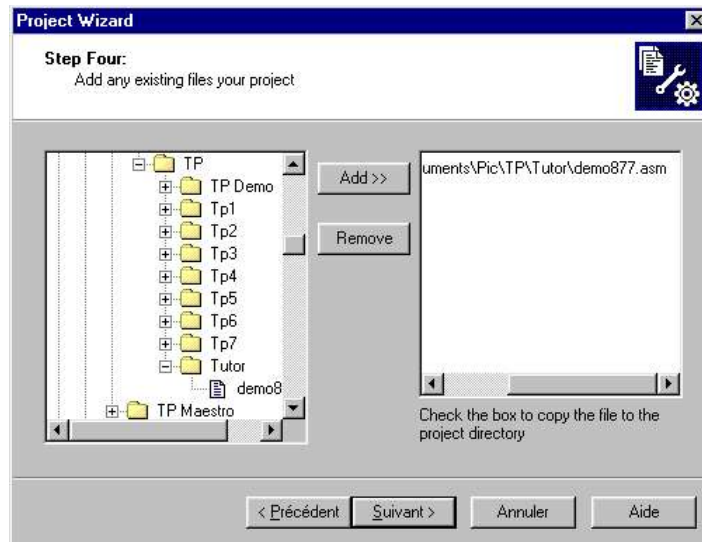
Si le nom du répertoire n'existe pas, MPLAB vous l'indique et vous demande si vous voulez le créer.



Pour continuer cliquez sur *Suivant*.

5 Insérez le fichier source

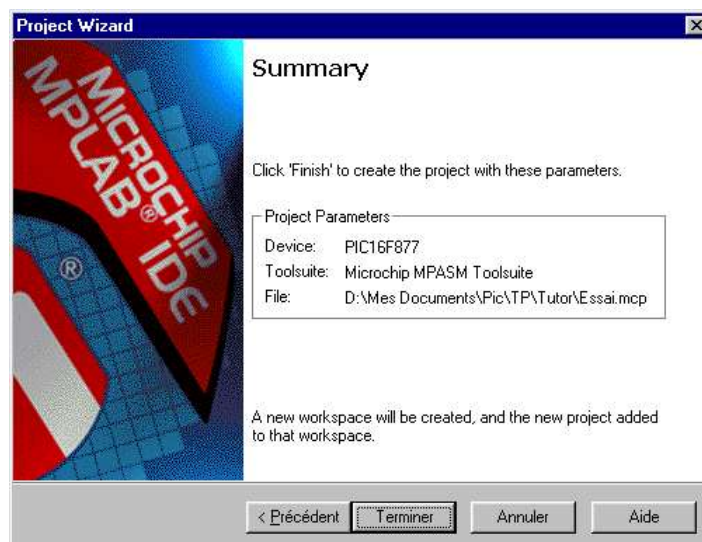
La fenêtre qui suit vous permet de sélectionner le fichier source que vous voulez utiliser. Il est toujours préférable que fichier source et projet soient dans le même répertoire.



Attention Si le fichier demo877.asm n'a pas été créé au paravent il n'existe pas et vous ne pouvez pas l'insérer. Nous ferons plus loin une autre manière de rajouter un fichier source à un projet. Pour continuer cliquez sur Suivant.

6 Validez le projet

La fenêtre qui apparaît résume le projet que vous venez de créer et vous permet de vous assurer que vous n'avez pas fait d'erreurs.



Cliquez sur terminer pour valider le projet.



3 - Écrire un programme et construire un projet

1 Créez votre fichier source

Une fois le projet entièrement défini, vous pouvez commencer à développer le code de votre application.

Sélectionnez *File<New*. Vous obtenez une fenêtre vierge de l'éditeur. Saisir le fichier ci-dessous en exemple (joint sous le nom demo877.asm).

```

*****
;
; Ce programme permet de faire clignoter la LED connectée à la ligne RB0      *
; il utilise la carte PicDem2Plus équipée d'un PIC16F877                      *
*****
LIST P=16F877 ;directive qui définit le processeur utilisé
#include <P16F877.INC>; Fichier de définition des constantes

VAR0 EQU      0x20 ; Variable de temporisation
VAR1 EQU      0x21 ; Variable de temporisation

        org      0x0 ; Adresse de départ après reset
        nop      ; Indispensable pour déboguer le programme avec ICD2
        goto     debut

        org      0x10 ; adresse de début du programme principal
debut call     init_port
boucle bsf      PORTB,0 ;on allume la LED
        call     tempo
        bcf      PORTB,0 ;on éteint la LED
        call     tempo
        goto     boucle

init_port clrf      PORTB ; Init PORTB : RAZ des bascules D
          BANKSEL TRISB ; choix de la "BANK" ou est TRISB
          movlw   b'00000000'
          movwf   TRISB ; toutes les lignes de PORTB en sortie
          BANKSEL PORTB
          return

tempo movlw     h'FF' ;durée de la tempo :
      movwf    VAR1 ;3*256*256 cycles machines
bou   movlw     h'FF' ; 1 cycle
      movwf    VAR0 ; 1 cycle
tempo1 decfsz   VAR0,1 ; 1 cycle decremente VAR0
      goto     tempo1 ; 2 cycles-->3x256*1E-6=768E-6s
      decfsz   VAR1,1 ; 1 cycle decremente VAR1
      goto     bou ; 2 cycles
      return
      END

```

Après avoir saisi le fichier sauvez le (*File>Save*) avec l'extension *.asm*.

Remarquez le changement d'apparence du texte ; Mplab permet de configurer l'environnement du texte (couleurs, polices, tabulation ...). Pour en savoir plus faire *Help > MPLAB Editor Help*.

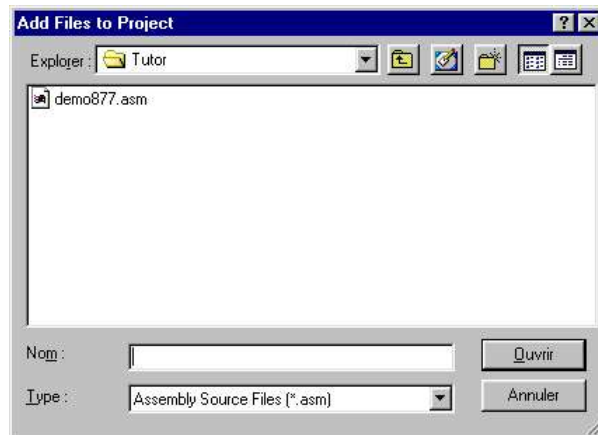
2 Ajouter le fichier au projet

Après avoir créé le fichier source il faut l'inclure dans le projet. Il faut **remarquer**



que le projet et les fichiers sources ne sont pas nécessairement dans le même répertoire.

1. Dans la fenêtre *nom du projet.mcw* sélectionnez *Source Files* puis avec le bouton droit faire *Add File*,



2. Sélectionnez le fichier que vous venez de créer et de sauver.

3. Cliquez sur *Ouvrir*.

Le nom du fichier doit apparaître dans la fenêtre projet au dessous de "Source Files". S'il apparaît sous "Unclassifiable" vérifiez que vous avez bien sélectionné "Microchip Toolsuite" comme suite logicielle (*Project>Set Language Toolsuite*).

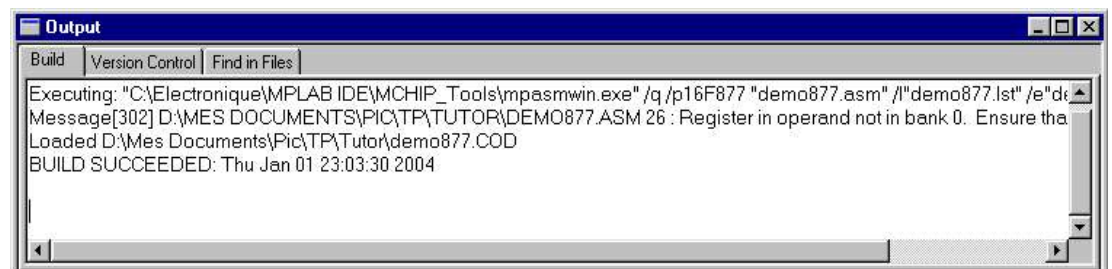
Remarque:

Si vous faites un clic droit sur le nom du projet vous obtenez le menu complet relatif au projet.

3 Construire le projet

Après avoir configuré le projet et saisi le code il est temps de construire le projet. Cela provoquera la compilation du fichier source en utilisant la suite logicielle choisie.

Sélectionnez *Project>Build All* (ou Ctrl + F10) pour construire le projet. Votre fichier doit être assemblé correctement et vous obtenez la fenêtre ci dessous.



S'il y a eu un problème de compilation celui ci est indiqué dans la fenêtre pour permettre de retrouver les erreurs. Un double clic sur l'erreur pointe directement la ligne du programme sur laquelle se situe l'erreur. Le type des informations affichées peut être défini: faire *Project > Build Options > Project* puis choisir l'onglet *MPASM Assembler* et enfin la catégorie *Output*.

Une fois la compilation terminée le fichier debug (*.cod ou *.cof) généré par Mplab



sera chargé. Ce fichier permet de déboguer le code source et de visualiser les variables symboliques dans les fenêtres d'observation.

4 - Simuler le comportement d'un programme

1 Configurer le simulateur

Après avoir construit notre projet nous voulons vérifier qu'il fonctionne comme nous l'avions prévu. Pour cela il faut choisir un outil de débogage. Nous utiliserons ici le simulateur mais le principe de fonctionnement est le même pour tous les outils.

Remarque: Seul le simulateur est installé automatiquement avec MPLAB. Les autres outils nécessitent d'être installés séparément.

Pour activer le simulateur sélectionnez *Debugger>Select Tools>MPLAB SIM*. On voit alors apparaître les changements suivants :

- ◆ La barre d'état au bas de la fenêtre devient celle de MPLAB SIM et indique notamment le contenu du PC, de W et l'état des 3 bits d'état Z, DC et C,
- ◆ De nouvelles fonctions apparaissent dans le menu debug,
- ◆ Sept nouveaux éléments apparaissent dans la barre d'outils.
 - 1 Exécution automatique
 - 2 Arrêt
 - 3 Exécution automatique animée
 - 4 Pas à pas
 - 5 Pas à pas avec exécution automatique des sous programmes
 - 6 Exécution automatique jusqu'à la fin du sous programme
 - 7 Reset



2 Exécutez le programme

Faites un Reset du programme; sélectionnez *Debugger>Reset* ou touche de fonction F6 ou bouton 7. Une flèche verte apparaît à gauche de la première ligne du programme.

```

D:\MES DOCUMENTS\PIC\TP\TUTOR\DEMO877.ASM
; Ce programme permet de faire clignoter la LED connectée à la ligne
; il utilise la carte PicDem2Plus équipée d'un PIC16F877
;*****
LIST P=16F877 ;directive qui définit le processeur utilisé
#include <P16F877.INC> ; Fichier de définition des con

VAR0 EQU 0x20 ; Variable de temporisation
VAR1 EQU 0x21 ; Variable de temporisation

org 0x0 ; Adresse de départ après reset
nop ; Indispensable pour déboguer le programme av
goto debut

org 0x10 ; adresse de début du programme principal
debut call init port
  
```

Attention: Il se peut que la flèche verte n'apparaisse pas et que lors de l'exécution

une fenêtre "Program Memory" s'ouvre; cela arrive lorsque le chemin absolu du fichier .asm est trop long (maxi 62 caractères).

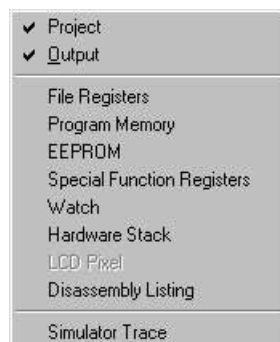
1. Lancez l'exécution de votre programme; faites *Debugger>Run* ou F9 ou bouton 1. L'information "Running" apparaît à la gauche de la barre d'état.
4. Pour arrêter l'exécution faites *Debugger>Halt* ou F5 ou bouton 2.

1. Vous pouvez aussi exécuter le programme pas à pas: faites *Debugger>Step Info* ou touche F7 ou bouton 3. A ce moment là la flèche verte pointe l'instruction qui va être exécutée.

A tout instant vous pouvez voir le contenu d'une variable en plaçant le pointeur de la souris sur le nom de la variable.

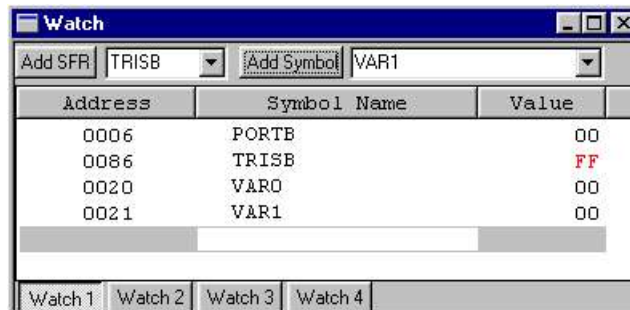
3 Visualisez la fenêtre d'observation

Lorsque l'on exécute le programme il est intéressant de voir évoluer le contenu des registres utilisés dans le programme. Toutes les fenêtres de debugage sont accessibles par le menu *View*.



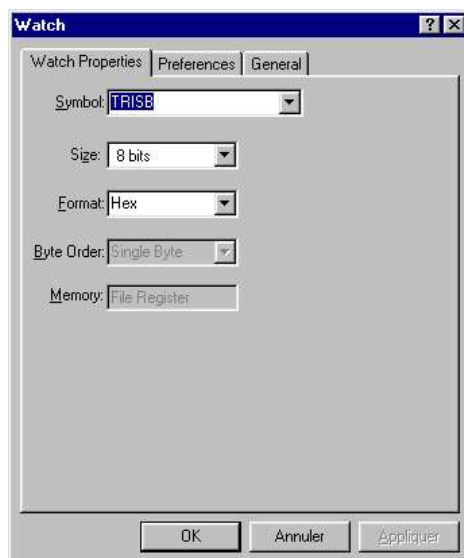
<i>File Register</i>	contenu de la RAM et des FSR
<i>Programm Memory</i>	contenu de la mémoire programme
<i>EEPROM</i>	contenu de la mémoire de sauvegarde en EEPROM
<i>Sp. Fct. Registers</i>	tous les registres SFR
<i>Watch</i>	Voir ci dessous
<i>Hardware Stack</i>	Contenu de la pile
<i>Disassembly Listing</i>	Code Hex desassemblé
<i>Simulator Trace</i>	Voir plus loin

Faites *View>Watch* pour ouvrir une fenêtre d'observation (Watch Windows),



1. Deux menus déroulants vous permettent de sélectionner soit le contenu des registres (Add SFR) soit le contenu des variables utilisés (Add Symbol).
2. Sélectionnez par exemple PORTB puis cliquez sur Add SFR - pour aller plus vite vous pouvez directement taper PORTB dans la fenêtre du menu déroulant -.
3. Faites la même chose pour sélectionner TRISB,VAR0 et VAR1.





Vous obtenez ainsi une fenêtre qui vous indique le nom, l'adresse et la valeur des quatre symboles sélectionnés. L'affichage de la valeur se fait par défaut en hexadécimal, ce mode peut être modifié en faisant un clic droit dans la fenêtre. Vous pouvez afficher les informations sur 8, 16, 24 ou 32 bits dans les format hexadécimal, binaire, décimal ASCII ou en nombre flottant. Sélectionnez *Properties* pour obtenir la fenêtre ci dessous et effectuer les modifications éventuelles.

4 Exécution et visualisation

1. Effectuez un Reset du programme,
2. Faites exécuter le programme pas à pas jusqu'à la ligne
boucle `bsf PORTB,0` ;on allume la LED
3. Faites avancer le programme d'un pas de plus et observez le changement d'état du bit 0 du PORTB.
4. Avancez de deux pas pour voir le contenu VAR1 à FF,
5. La temporisation est réalisée par décrémentation de deux variables; pour raccourcir la durée vous pouvez modifier directement les valeurs de VAR0 et VAR1 en cliquant deux fois sur la valeur et en saisissant la nouvelle valeur (par ex. 01),
6. Exécutez le programme de manière à revenir à la ligne
`bcf PORTB,0` ;on éteint la LED
7. Pour continuer utilisez le bouton *4 Step Over* ceci permet d'exécuter tout le sous-programme en mode *Run* et de pointer l'instruction suivante.

5 - Utiliser les points d'arrêt et le mode Trace

Ces deux modes permettent soit d'arrêter l'exécution du programme à des endroits précis (point d'arrêt) soit d'enregistrer les actions du programme (mode Trace).

1 Utilisation d'un point d'arrêt

Il est souvent intéressant d'exécuter un programme et de l'arrêter à un endroit bien déterminé ; ceci est possible en utilisant les points d'arrêt. Ci-dessous est décrite l'utilisation d'un point d'arrêt.

1. Effectuez un Reset de l'application,
2. Effectuez un clic droit sur la ligne suivante
`bcf PORTB,0` ;on éteint la LED
3. Dans le menu contextuel qui apparaît choisissez *Set Break Point*. Un signal stop doit apparaître en face de la ligne indiquée,

```

D:\MES DOCUMENTS\PIC\TP\TUTOR\DEMO877.ASM*
VAR0 EQU 0x20 ; Variable de temporisation
VAR1 EQU 0x21 ; Variable de temporisation

org 0x0 ; Adresse de départ après reset
nop ; Indispensable pour déboguer le programme av
goto debut

debut org 0x10 ; adresse de début du programme principal
call init_port
boucle bsf PORTB,0 ; on allume la LED
call tempo
bcf PORTB,0 ; on éteint la LED
call tempo
goto boucle

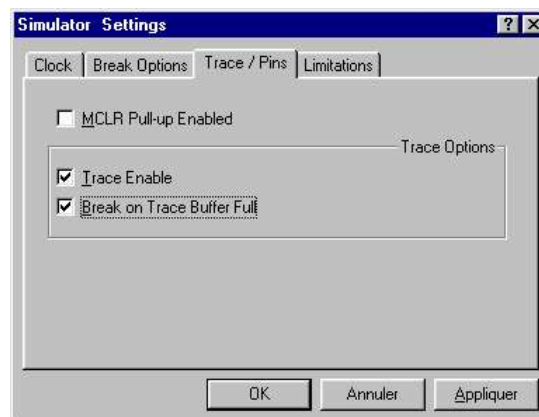
```

4. Lancez la simulation en mode *Run* ; l'exécution s'arrête à l'emplacement du point d'arrêt pré défini. Pour utiliser les points d'arrêt multiples voir l'aide en ligne.

2 Utilisation du mode Trace

Ce mode permet d'enregistrer les actions du programme en cours d'exécution et de suivre ensuite son comportement.

Validez le mode Trace en faisant *Debugger>Settings*; choisissez ensuite l'onglet *Trace / Pins* pour obtenir la fenêtre ci dessous.



Deux cases à cocher contrôlent la manière dont s'effectue la collecte des données. Si seule la première case est cochée (Trace Enable) le simulateur enregistre les données en mode Run jusqu'à ce qu'il rencontre un point d'arrêt ou qu'il soit arrêté manuellement. Il permettra ensuite de visualiser les 8192 derniers cycles.

Si la deuxième case est aussi cochée (Break on Trace Buffer Full) le simulateur enregistre les 8192 cycles d'exécution et arrête la mémorisation et l'exécution du code.

Faites *View>Simulator Trace*. La trace du simulateur montre pour chaque cycle machine l'instruction exécutée plus les données qui ont été lues ou écrites dans les registres.

Pour chaque instruction la trace donne l'adresse du Compteur de Programme, le code opération (Op) en hexadécimal, l'étiquette s'il y a lieu et l'instruction désassemblée. Les quatre colonnes suivantes montrent les données lues ou écrites:

Line	Addr	Op	Label	Instruction	SA	SD	DA	DD	Cycles
0	0000	EFOE		GOTO Ox1c	----	--	----	--	000000000000
1	0002	F000		NOP	----	--	----	--	000000000001
2	001C	6AE8	Start	CLRF Oxfe8, 0	----	--	OFE8	00	000000000002
3	001E	6E82		MOVWF Oxf82, 0	----	--	OF82	00	000000000003
4	0020	6E94		MOVWF Oxf94, 0	----	--	OF94	00	000000000004
5	0022	6A00	Init	CLRF 0, 0	----	--	0000	00	000000000005
6	0024	2A00	IncCount	INCF 0, 0x1, 0	0000	00	0000	01	000000000006
7	0026	5000		MOVF 0, 0, 0	0000	01	OFE8	01	000000000007
8	0028	6E82		MOVWF Oxf82, 0	----	--	OF82	00	000000000008
9	002A	EC19		CALL Ox32, 0	----	--	----	--	000000000009
10	002C	F000		NOP	----	--	----	--	00000000000A
11	0032	0EFF	Delay	MOVLW Oxff	----	--	OFE8	FF	00000000000B
12	0034	6E02		MOVWF Ox2, 0	----	--	0002	FF	00000000000C
13	0036	6E04		MOVWF Ox4, 0	----	--	0004	FF	00000000000D

- ◆ SA Adresse Source adresse du registre où a été lue la donnée
- ◆ SD Source Data donnée lue
- ◆ DA Destination Adresse adresse du registre où a été écrite la donnée
- ◆ DD Data Destination donnée écrite

La dernière colonne donne le nombre de cycles. Cela peut permettre de mesurer la durée d'exécution d'une partie du programme.

Si vous cliquez avec le bouton droit de la souris dans la ligne grise du haut vous obtenez un menu qui vous permet de choisir les colonnes à afficher. Remarquez les "Probe" qui peuvent être utilisées avec l'émulateur ICE2000.

6 - Programmer un composant

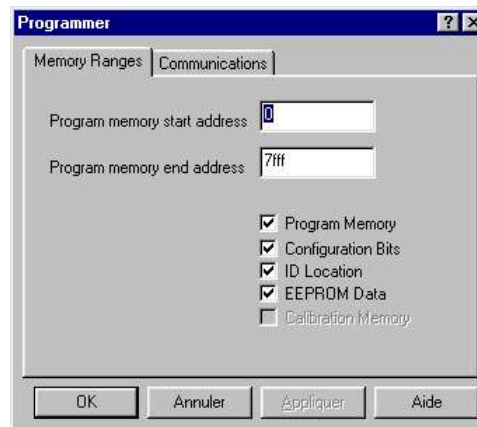
Microchip propose trois outils pour programmer ses micro contrôleurs.

- ◆ PICSTART Plus programmeur de base idéal pour les prototypes,
- ◆ PRO MATE II permet la programmation en série,
- ◆ MPLAB ICD 2 permet à la fois le débogage et la programmation "in situ".

1 Sélectionnez le programmeur

Pour sélectionner un programmeur suivre la démarche ci dessous.

1. Faites *Programmer>Select Programmer* et choisissez l'outil que vous souhaitez utiliser. Le menu déroulant *Programmer* change en fonction du choix effectué et de nouvelles icônes sont ajoutées.
2. Établissez la communication avec le programmeur. Pour PRO MATE II et PICSTART Plus faire *Programmer>Enable Programmer*. Pour MPLAB ICD2 faire *Programmer>Connect*. Un « setup Wizard » permet de paramétrer l'ICD2 et de faire en sorte qu'il s'autoconnecte dès qu'on le sélectionne.
3. Utilisez *Programmer > Settings* pour sélectionner la communication appropriée avec votre programmeur et les parties des mémoires à programmer (pour commencer utilisez les valeurs par défaut). La fenêtre ci-dessous correspond à PICSTART Plus. Voir plus loin pour ICD2.



2 Spécifiez les bits de configurations

Ces bits peuvent être spécifiés directement dans votre code source (méthode recommandée) soit manuellement en utilisant la fenêtre de configurations de bits en choisissant *Configure > Configuration Bits*. La fenêtre ci dessous correspond aux bits de configuration d'un 16F876.

Address	Value	Category	Setting
2007	3FFF	Oscillator	RC
		Watchdog Timer	On
		Power Up Timer	Off
		Brown Out Detect	On
		Low Voltage Program	Enabled
		Flash Program Write	Enabled
		Data EE Read Protect	Off
		Code Protect	Off

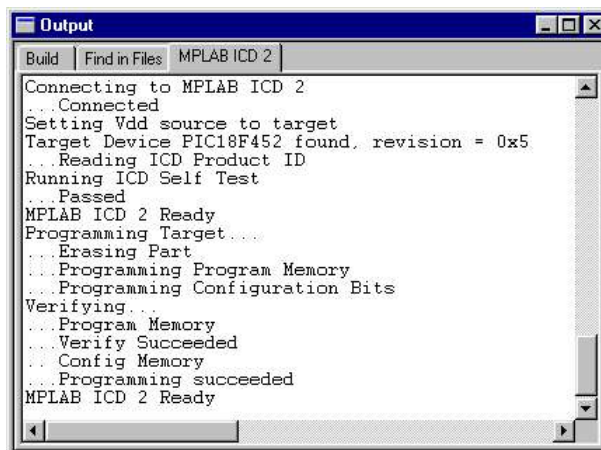
Ceux ci permettent d'indiquer au processeur quel type d'oscillateur est utilisé ainsi que les fonctions internes de surveillance que l'on désire valider. Leur nombre dépend du type de processeur utilisé.

Remarque : le WDT (Watch Dog Timer) est validé par défaut, pensez à le dévalider ou à insérer l'instruction "Clrwdt" dans la boucle principale de votre programme. Lorsque le composant est programmé il faut débrancher l'ICD2 pour que l'application puisse fonctionner.

3 Programmez le composant

Pour programmer le composant cible il suffit de faire *Programmer > Program*, le code chargé dans MPLAB IDE est alors transféré dans la mémoire du micro. La progression des opérations est indiquée dans la fenêtre "Output"; celle qui est présentée ci dessous correspond à la programmation d'un 18F452 par le module ICD2





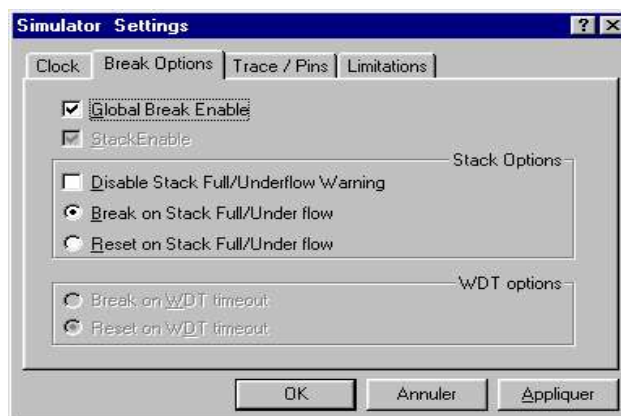
7 - Utiliser les fonctions avancées du simulateur

1 Configurer le simulateur

Après avoir sélectionné MPLAB SIM comme outil de débogage, sélectionnez *Debugger>Settings*; vous obtenez la fenêtre ci dessous.

L'onglet Clock permet de fixer la fréquence de l'horloge pour permettre à la trace du simulateur de définir les durées d'exécution des cycles machines et donc de mesurer des durées d'exécution de parties de programme.

L'onglet "Break Options" donne des information sur les points d'arrêt. Si la case "Global Break Enable" n'est pas cochée alors les points d'arrêt seront inopérants. Ceci est utile quand on a de nombreux points d'arrêt et que l'on veut les invalider sans les supprimer. Remarquez la possibilité d'observer les dépassements de la pile.



L'onglet "Trace / Pins a été vu au paragraphe utilisation du mode Trace.

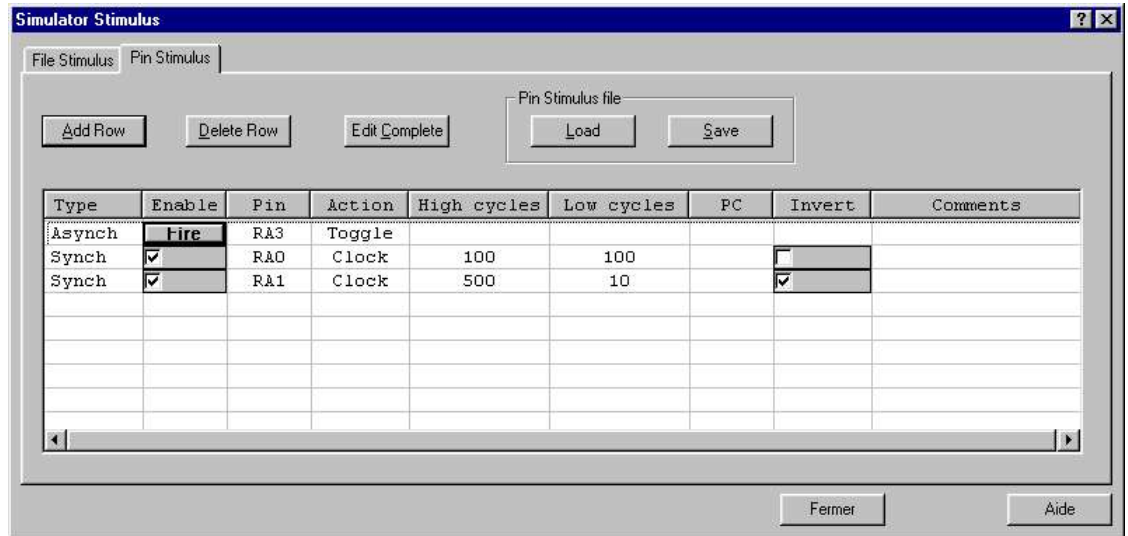
L'onglet "Limitations" indique les limitations générales du simulateur par rapport au composant qui doit être simulé ; le bouton "Détails" ouvre le menu d'aide et permet de détailler les limitations liées à chaque composant.

2 Création de stimulus

Les stimuli sont des pseudo signaux appliqués aux entrées du processeur. Ils permettent de simuler l'influence de l'environnement sur le déroulement du

programme.

Choisissez *Debugger* > *Stimulus* pour ouvrir la fenêtre ci-dessous

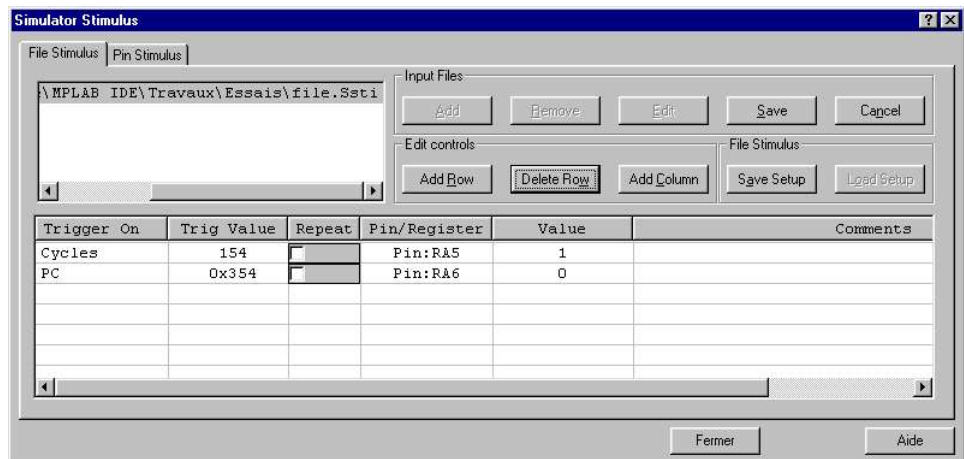


L'onglet "Pin Stimulus" permet de définir les pseudo signaux que l'on veut créer; ils peuvent être de deux types :

- ◆ soit synchrones et ils sont définis comme des horloges dont on donne la durée à l'état haut et bas,
- ◆ soit comme des signaux asynchrone actionnés par clic sur le bouton "Fire", ce clic produisant l'effet indiqué dans la colonne "Action".

Fermer la fenêtre de stimulus invalide les stimuli.

L'onglet "File Stimulus" permet de créer des fichiers de stimulus qui peuvent être mémorisés et rejoués.



Pour créer un fichier faites *Add* puis donnez un nom; faites ensuite *Edit* et ensuite *Add Row* pour créer de nouvelles lignes de stimulus. L'aide de MPLAB SIM donne la démarche à suivre.

8 - Utiliser ICD2

Le module ICD2 (In Circuit Debugger) est un module qui permet à la fois de déboguer et de programmer les processeurs supportés. Il se connecte au PC sur



un port série ou sur un port USB. Il se raccorde à l'application à l'aide d'un connecteur du type RJ12 ; la communication est une liaison série synchrone.

En mode *Debugger* le composant est programmé mais l'exécution reste sous le contrôle du PC, celle ci peut s'effectuer en mode run ou en mode pas à pas. Dans le premier cas un seul point d'arrêt peut être placé. Toutes les fenêtres de visualisation peuvent être utilisées et il est possible de modifier la valeur contenue dans un registre.

Le mode *debugger* nécessite certaines ressources du micro:

- 1 - Quelques centaines d'octets de mémoire programme
- 2 - Quelques registres
- 3 - Un ou deux niveaux de la pile

Le bit "Background Debug" n'est plus accessible avec la version 6.40; il est automatiquement configuré lors du choix du mode *Debug* ou du mode *Program*. Il faut noter que cette version ne permet pas l'utilisation simultanée de l'ICD2 en programmeur et en débogueur, quand vous choisissez le mode debug, le mode program est automatiquement dévalidé (et vice versa).

1 Configurez l'ICD2

La version 6.40 dispose d'un « *Setup Wizard* » qui sert de guide lors de la configuration. Pour l'obtenir faire *Debugger* (ou *Programmer*) puis *MPLAB ICD2 Setup Wizard* vous obtenez la fenêtre ci dessous.

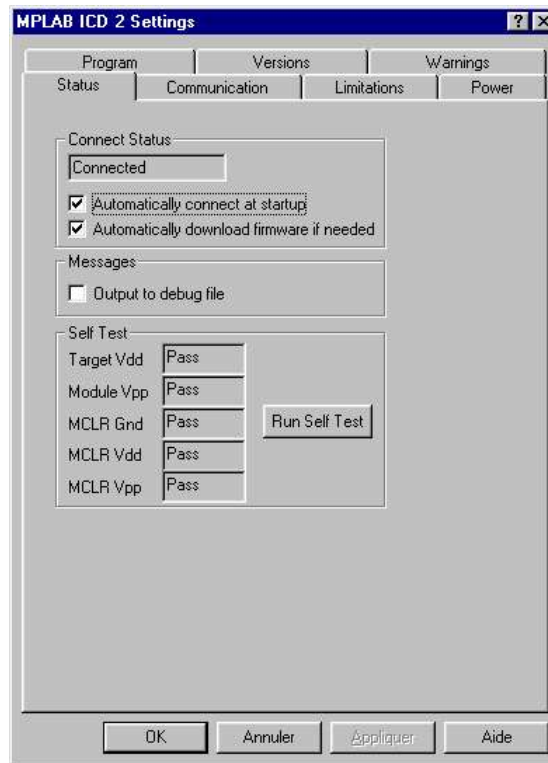


En cliquant sur le bouton suivant vous pouvez configurer les paramètres suivants:

1. Le mode de communication (USB ou RS232)
2. La source d'alimentation de la carte cible; soit elle possède sa propre alimentation soit elle est alimentée par l'ICD2 (Imax 200mA) qui doit lui être alimenté par un bloc 9V.
3. Autoriser ou non l'auto connexion à l'ICD2 lors de l'ouverture du projet.
4. Autoriser ou non le chargement automatique de l'« operating systems » de l'ICD2 si une version plus récente existe.

La dernière fenêtre résume la configuration effectuée.

L'ensemble des paramètres peuvent être modifiés en faisant *Debugger>Settings* ou *Programmer>Settings* ;vous obtenez alors la fenêtre ci dessous.



1. Configurez la communication

Choisissez l'onglet Communication pour modifier ou reconfigurer le port de communication entre le PC et l'ICD2. Deux vitesses sont possibles pour la RS232 : 19200 ou 57600 bauds.

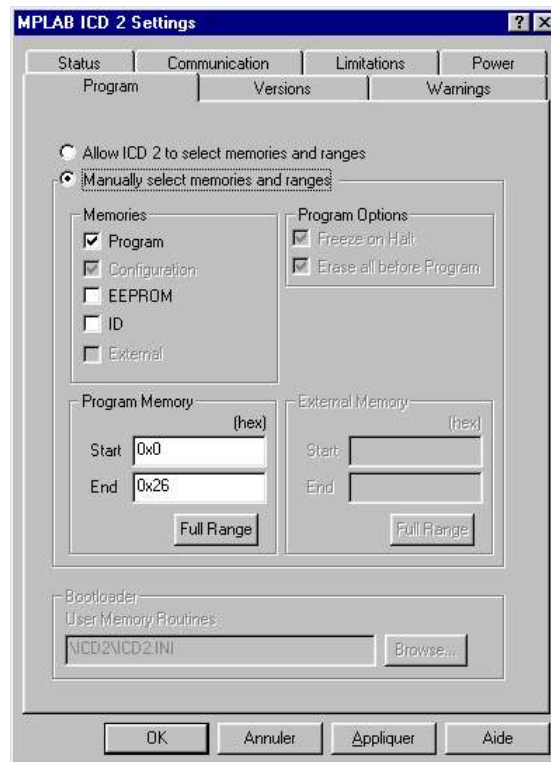
Attention: si vous l'utilisez, le port série COMx utilisé doit être configuré de la manière suivante : le contrôle de flux doit être matériel et les tampons FIFO doivent être désactivée. Pour cela faire, depuis le bureau de Windows, un clic droit sur l'icône "Poste de travail", puis "Propriétés". Dans l'onglet "Gestionnaire de périphériques" sélectionnez "Port de communication COMx" puis "Paramètres du port"; dans "contrôle de flux" sélectionnez "matériel" puis cliquez sur le bouton "Avancées" et décochez la case "Tampon FIFO".

2 Choix de la mémoire à programmer

Choisissez l'onglet Program pour obtenir la fenêtre ci-dessous. Si le bouton « *Allow ICD2 to select memories and ranges* » est coché MPLAB choisit lui même les zones mémoire à programmer. C'est une fonction très pratique car elle permet d'obtenir le temps de programmation le plus court possible. **Attention** les adresses de fin de programme sont parfois mal détectées et vous risquez de ne pas programmer tous le code nécessaire; il faut alors cocher le bouton « *manually select memories and ranges* »

Vous pouvez alors définir la ou les partie de mémoire que vous voulez programmer : la mémoire programme, les bits de configuration, l'EEPROM ou le code de protection et aussi de définir l'adresse de fin de la mémoire programme. Faites attention que l'adresse que vous saisissez soit supérieure à l'adresse de fin de votre programme.





3 Contrôlez l'alimentation

Le module ICD2 peut être alimenté directement par la liaison USB; si vous utilisez la liaison RS232 il est impératif d'alimenter le module sous 9V, 750mA.

Attention : on peut alimenter la carte cible à partir de l'ICD2 mais pas le contraire cela peut endommager l'ICD2.

Choisissez l'onglet Power pour obtenir la fenêtre ci-dessous.

Cet onglet visualise la valeur des tensions Vdd et Vpp. Une case à cocher permet d'alimenter la carte cible à partir du module ICD2 (Imax 200mA).

Si vous constatez des erreurs ou des imprécisions dans le document, je vous remercie de me les communiquer par email j'en tiendrai compte pour les prochaines version. La dernière version sera a tout moment disponible sur le site Genelaix au format pdf.
[Http://artemmis.univ-mrs.fr/iufm-genelec/pic.htm](http://artemmis.univ-mrs.fr/iufm-genelec/pic.htm)

André L.