LES OUTILS DE DEVELOPPEMENT

Pour s'initier à l'utilisation d'un microcontroleur il faut s'astreindre à écrire de nombreux programmes en s'inspirant par exemple des textes donnés plus loin. Mais ce n'est profitable que si l'on dispose des outils nécessaires à l'écriture et au test de ces programmes. Avant de passer aux exemples concrets nous consacrerons donc un chapitre à la description de ces outils.

Plusieurs sociétés ont commercialisé des assembleurs pour les PIC ce fut la cas de PARALAX mais MICROCHIP diffuse un tel outil gratuitement , jusqu'en 1999 sur le DC de la société , depuis par téléchargement sur son site WEB. www.microchip.com

Le logiciel de MICROCHIP est MPLAB IDE Il faut d'abord apprendre à le mettre en œuvre puis connaître les commandes et règles de l'assembleur .

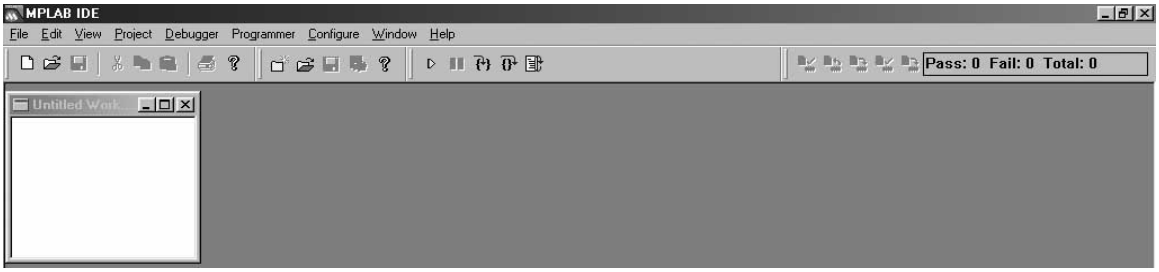
Attention la mise en œuvre et le format requis varie sensiblement d'une version à l'autre , la version décrite ici est la 6 .

L'ASSEMBLEUR MPLAB.IDE

MISE EN ŒUVRE :

Le téléchargement étant effectué (on peut aussi se procurer le logiciel sur CD) son installation sur l'ordinateur n'offre pas de difficultés particulières. Sous Windows une icône apparaît sur le bureau . En cliquant sur elle on obtient (si aucun projet n'a été préalablement entré) l'écran ci dessous .

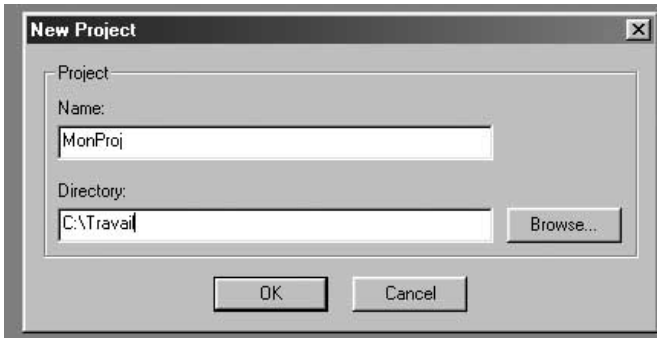
Ecran initial de MPLAB



Pour faire fonctionner votre système vous serez amenés à développer plusieurs fichiers (Source , objet etc..) dont l'ensemble est appelé un PROJET MPLAB vous demande d'attribuer à ce PROJET un nom , il ouvrira alors un répertoire spécial à ce nom pour stocker tous les fichiers nécessaires .

Dans la barre d'outils on commence donc par cliquer sur **PROJECT** puis **NEW** : Une fenêtre s'ouvre dans laquelle vous entrez le nom du projet ;par exemple MONPROJ ainsi que le répertoire dans lequel il sera créé sur le disque ; par exemple C:\TRAVAIL

Entrée du nom du projet et localisation



Dans le cadre en haut à gauche apparaît le noms du projet qui va être créé MONPROJ .MCP ainsi que les outils qui seront utilisés. .

Il faut maintenant configurer l'outil pour cela on utile encore la barre d'outils

CONFIGURE > Setting Project

Plusieurs cadres s'ouvrent .Dans un premier temps ne rien changer aux options par défaut proposées .

Puis **CONFIGURE > Select Device**

Il s'agit de définir le type de microcontroleur que vous allez utiliser .

Dans la fenêtre qui s'ouvre vous pouvez choisir le boîtier .Les boutons rouges ou verts vous indiquent quels sont les outils de développement commerciaux de MICROCHIP sont compatibles avec ce boîtier.

Choix du boîtier

Valider **MPASM Assembler**

Puis on sélectionne le langage

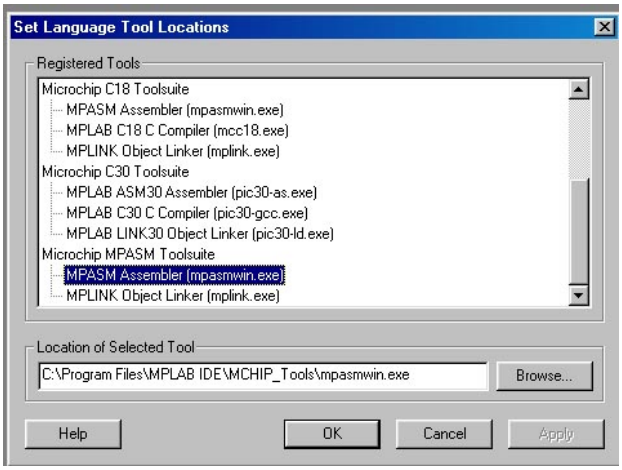
Project > Set Language ToolSuite

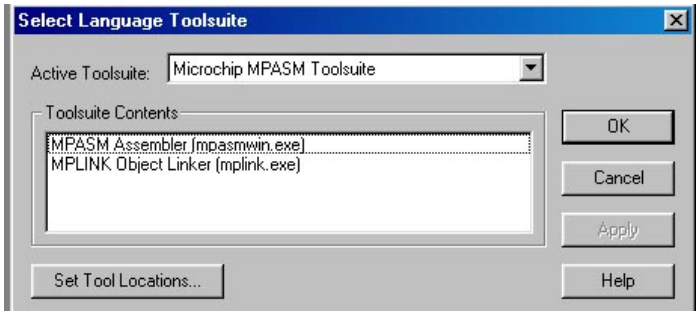
Il faut maintenant introduire le texte assembleur proprement dit /

MPLAB contient tous les outils nécessaires à l'introduction du fichier assembleur et son test logiciel .

Il faut d'abord préciser quel est l'outil utiliser . Pour cela **Project > Set Language Tool Location**

Apparaît la fenêtre ci dessous :





Dans la fenêtre supérieure du cadre activer (si ce n'est pas déjà le cas)
Microchip MPASM TooolSuite
Puis OK

On peut maintenant créer le code source

Sélectionner **Files > New**

Une grande fenêtre s'ouvre à l'écran dans laquelle on peut introduire le texte du projet . Nous verrons plus loin le détail de la syntaxe nécessaire . A titre d'exemple préliminaire entrons le texte suivant qui est un programme minimal affichant seulement 99H sur le port B .

Attention de bien respecter la disposition en colonnes comme cela sera précisé plus loin .

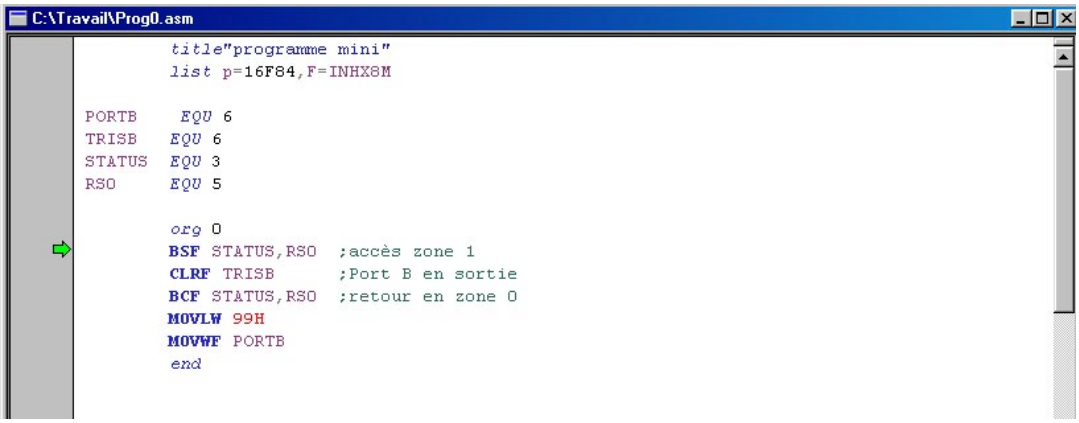
```
                Title 'programme mini '
                List p=16F84 , f=inhx8M

PORTB           equ 6
TRISB           equ 6
STATUS          equ 3
RS0             equ 5

                Org 0
                BSF STATUS,RS0          ;accès  page 1
                CLRF TRISB              ; port B en sortie
```

```
        BCF STATUS,RS0          ;retour page 0
        MOVLW 99h
        MOVWF PORTB
        End
```

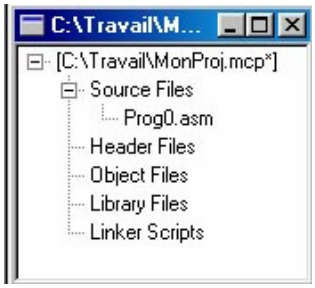
Lorsque le texte est entièrement tapé le sauver avec un nom par exemple PROG0.ASM
Fichier > **Save as** PROG0.ASM . dans le répertoire de travail C:\travail\
La fenêtre à l'écran prend des couleurs



En violet les noms et étiquettes
En bleu gras les mnémoniques des instructions
En bleu fin le texte d'assemblage
En rouge les nombres (litteral)
En vert les commentaires
et le titre choisi apparaît.

Vous pouvez maintenant insérer ce fichier dans votre projet
Project > Add Files to Project
Dans la nouvelle fenêtre qui s'ouvre entrer le nom du fichier précédent .Il est alors inséré dans la petite fenêtre initiale

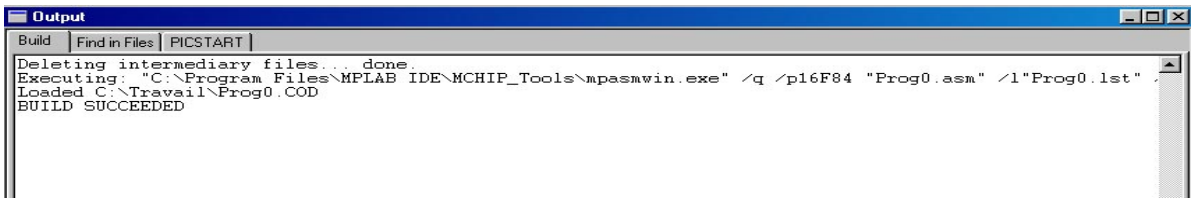
Il reste à sauver le projet par **Project > Save Project**



Maintenant que l'ensemble du projet est bien défini il est temps de le construire c'est à dire d'abord de lancer l'assemblage pour créer le fichier objet .

Project > Build All

L'assembleur travaille et un fichier de sortie s'affiche. Il indique si l'assemblage s'est bien déroulé, sinon signale les erreurs .



Sont alors créés plusieurs fichiers

Prog0.cod fichier non texte ,non lisible
Prog0..lst Le détail des opérations d'assemblage contient le texte de départ, les codes machine et position mémoire et éventuellement les erreurs trouvées
Prog0.hex La liste des mots de 14 bits constituant le programme suivant le format Intel .

Prog0.err le détail des erreurs s'il y en a.
Dans le cas présent :

Le fichier .lst

```
MPASM 03.20.07 Released                              PROG0.ASM   PAGE
1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00001                 title"programme mini"
                      00002                 list p=16F84,F=INHX8M
                      00003
  00000006            00004 PORTB   EQU 6
  00000006            00005 TRISB   EQU 6
  00000003            00006 STATUS  EQU 3
  00000005            00007 RS0     EQU 5
                      00008
0000                  00009                 org 0
0000   1683           00010                 BSF STATUS,RS0  ;accÞs
page 1
0001   0186           00011                 CLRF TRISB
;Port B en sortie
0002   1283           00012                 BCF STATUS,RS0  ;retour
en page 0
0003   3099           00013                 MOVLW 99H
0004   0086           00014                 MOVWF PORTB
                      15                    end


MPASM 03.20.07 Released                              PROG0.ASM
PAGE  2
programme mini

SYMBOL TABLE
  LABEL                             VALUE

PORTB                             00000006
RS0                               00000005
STATUS                            00000003
TRISB                             00000006
__16F84                           00000001


MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXX----------- ---------------- ---------------- ----------
------

All other memory blocks unused.

Program Memory Words Used:     5
Program Memory Words Free:  1019


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Le fichier .HEX

```
:0A0000008316860183129930860 0F2
:00000001FF
```

c'est ce fichier qui sera exploité par le programmateur .

Rappel des opération à effectuer au lancement de MPLAB	
PROJECT > NEW	Nom Localisation
CONFIGURE >Setting Project	
CONFIGURE >Select Device	CPU
PROJECT> Set Language Tool Location	MPASM Assembler
PROJECT> Set Language Tool Suite	Microchip MPASM Toolsuite
FILES > New	Taper le code
SAVE As	MonProjet.asm
PROJECT >Add Files To Project	
PROJECT> Save Project	
PROJECT> Build All	

LE FICHIER SOURCE

Le fichier source peut être tapé dans MPALB comme ci dessus ou à l'aide d'un éditeur de texte quelconque par exemple l'outil EDIT du DOS .et bien sûr dans MPLAB lui-même.

Ce fichier est sauvé avec le suffixe .ASM

Format du fichier source

Ce fichier doit être conforme au format suivant :

Chaque ligne peut contenir 4 types d'information

Des labels (étiquettes)
Des mnémoniques d'instruction
Des opérandes
Des commentaires

La position de ces éléments est importante.

Les labels doivent se trouver impérativement dans la première colonne
Un mnémonique doit commencer dans la seconde colonne (ou plus loin) , il en est de même des directives d'assemblage
L'opérande suit le mnémonique
Les commentaires suivent les opérandes et sont précédés d'un point virgule ;
Chaque colonne a une largeur maximale de 255 caractères .

Label Commandes ou mnémonique Opérandes (séparés par ,) ; commentaires

En tête de fichier il est possible de placer un titre et un texte quelconque sous forme de commentaire , c'est à dire derrière un ; Le titre peut également être introduit après la commande TITLE dans la seconde colonne .format :

Format : **TITLE « titre_text »**

C'est ce qui a été fait dans l'exemple précédent :

TITLE ″programme mini ″

Puis ensuite , toujours en seconde colonne , la commande LIST suivie d'un certain nombre d'options : **LIST [« option1 », »option2 »,……]**

La plus importante , le type de microprocesseur

P =

Puis après une virgule F= le format définissant l'organisation des mots de programme :

INHX16 qui organise le résultat en mots de 16 bits
INHX8S
INHX8M qui organisent le résultat en mots de 8 bits ,nous utiliserons de format .
et PICICE adapté à l'émulateur de MICROCHIP.

```
List  p=16F84 ,F=INHX8M
```

Dans certains cas il est nécessaire d'inclure des fichiers ou des programmes , le mot clé est #include Par exemple :

```
#include «P16f84.inc »
```

Les informations contenues dans le fichier .INC sont nécessaires pour que l'assembleur reconnaisse les noms des registres internes du microprocesseur .Par exemple la ligne :
MOVWF STATUS qui charge le contenu de W dans STATUS est admise , en absence de fichier .INC il aurait fallu écrire :
MOVWF 03
ou préalablement définir l'adresse du registre STATUS par
STATUS EQU 03
(voir note en fin de chapitre)

Les directives du langage

Ce sont des commandes qui apparaissent dans le texte source mais ne sont pas directement transcrites en mots de code .

Directives concernant les données

DATA
[<label>) DATA <expression>,[DATA <expression],…
initialise un ou plusieurs mots de la mémoire programme avec des données qui peuvent être des nombres ,des constantes préalablement définies, des caractères (entre ' expl : data 'A')
Les nombres peuvent être entrés :

En décimal	45 ou .45
En hexadécimal	0xF3 ou H'F3'
En octal	Q'173' ou O'777'
En binaire	B'00011101'

Des opérations sont admises

+ - * /	les opérations élémentaires
>> et <<	décalages à droite et gauche
==	égalité logique
!=	Non égalité logique
<= >=	Plus petit ou plus grand que
!	non exemple !(x==y)
~	complément
|	ou inclusif exemple x|y
&	AND inclusif
|| &&	OU et ET logiques
^	OU exclusif
$	Valeur en cours du compteur programme

exemple GOTO $

Par exemple 0x47 & 0x23 est équivalent à DATA 3

DB déclare une donnée d'un seul octet , nombre ou caractère mais les opérations ne sont pas admises
[<label<] db <expression>
attention les octets sont placès à partir de la gauche or les mots mémoire n'ont que 14 bits DB 0x1F est accepté car il place en mémoire comme premier mot 00011111 00000000 alors

que DB 0x81 est refusé car il donnerait un mot 10000001 00000000 qui a un 1 en position 15 .

DE initialise des mots de programme 1 octet dans l'EEPROM
Par exemple DE 12 place comme premier mot mémoire 0012 alors que DB 12 place 1200

Label EQU valeur
Donne la valeur indiquée au label
Ainsi PortB EQU 05 donne la valeur 5 au mot portB qui peut être ensuite utilisé à la place de 5, par exemple MOVWF PORTB est équivalent à MOWF 5 charge la valeur de W dans la case 5

Label SET valeur
Est équivalente à la précédente mais la valeur peut être changée dans le programme .Par exemple
DEBUT SET 6 DEBUT prend la valeur 6
DEBUT SET DEBUT+4 DEBUT vaut maintenant 10

VARIABLE <label> [=<expression>][,<label>[=<expression>]
Crée des symboles qui seront utilisés par l'assembleur Cette directive est analogue à SET mais l'initialisation n'est pas nécessaire .

Directives de Listing

Elles contrôlent le format du fichier .LST fourni par l'assembleur

LIST <option>,[<option>]…
Introduit une suite d'options dont 2 ont déjà été citées .
Les plus importantes :

C= largeur des colonnes du .LST , par défaut 132
F= format la façon dont son représentés les mots de 14 bits , voir plus haut , par défaut INHX8M
N= nombre de lignes par page par défaut 60
P= le processeur
R= HEX/DEC/OCT la base de numération utilisée , par défaut HEX

TITLE « nom de programme « voir plus haut

SUBTITLE « nom de sous programme «

PAGE
Insère un saut de page dans le listing

SPACE
Insère une ligne de blanc dans le listing

Positionnement des Fusibles

__CONFIG *Attention aux deux caractères _ devant le C*

Les fusibles sont des bits qui déterminent le type d'oscillateur choisi, la mise en œuvre ou non du chien de garde etc.. .Ils sont écrits dans la mémoire à l'adresse 2000 qui n'est pas accessible comme les autres cases de la mémoire programme. L'écriture dans cette case particulière ne peut être effectuée que lors de la programmation du circuit.. Certains programmateurs permettent de choisir l'état des fusibles au moment de la programmation , mais il est possible d'inclure en début de programme une directive spéciale _CONFIG (Attention au tiret devant le mot CONFIG) .
Dans le cas du PIC 16F84 le mot à écrire est le suivant (c'est bien sûr un mot de 14 bits)

13	12	11	10	9	8	7	6	5	4	3	2	1	à
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	/PWRTE	WDTE	FOSC1	FOSC0

CP est un bit de protection Si CP=0 la relecture du programme mémoire est impossible
/PWRTE Power-up Timer Enable Bit s'il est au 0 un temps de retard est introduit au RESET de façon à ce que tous les registres internes du CPU aient eu le temps de se positionner .
WDTE Watchdog Timer Enable Bit Au 1 le chien de garde est activé .Attention le programme doit en tenir compte .
FOSC1-FOSC0 Définit le type d'oscillateur choisi:

11 Oscillateur RC
10 HS Oscillateur
01 XT Oscillateur
00 LP Oscillateur

Par exemple si un oscillateur à quartz est choisi (XT) , le chien de garde désactivé et le retard au démarrage présent (conseillé) , le mot sera :
11111111110001 ou 3FF1 et on placera en début de programme la directive :

__CONFIG H'3FF1'

Les instructions de contrôle

ORG n détermine le début de l'implantation en mémoire Au début par défaut ORG 0
Il peut y avoir plusieurs ORG dans un listing source .

END fin du programme il n'y en a qu'un seul

Les MACRO

Une macro instruction permet d'appeler par un seul nom une longue suite d'instructions
LABEL (nom de la MACRO) MACRO [<arguments>,<arguments>,…]
ENDM
La définition est divisée en 3 parties , l'entête, le corps et la terminaison

L'entête est formée du nom (LABEL) du mot MACRO et des paramètres passés
Le corps est une suite d'instructions
La MACRO se termine par ENDM

Exemple : la MACRO suivante nommée CFL_JGE compare le contenu du registre FX à la constante CON et saute en SAUT si FX=> CON

```
CFL_JGE   MACRO FX,CON,SAUT   l'entête
MOVLW CON & 0xFF                          Ce qui suit MOVWF doit être un
                                  literal d'ou l'opération qui fabrique le literal égal au contenu de CON
                                  , cette instruction met à 0 le carry
SUBWF FX,1                                Soustraction W-FX
BTFSC STATUS,CARRY                  Saut si le carry est nul
GOTO SAUT                                 exécutée si carry=1
ENDM
```

Si **une MACRO doit comporter des sauts internes**, les labels doivent être définis comme locaux sur la ligne qui suit immédiatement celle ou figure le mot MACRO. Le ENDM terminant la macro ne doit pas être précédé d'un label.
Exemple : La MACRO WC transfert le contenu du CARRY sur le fil 2 du port A :

```
WC  MACRO
LOCAL  ZEROW,FINW          ;définition locale des labels
BTFSS   STATUS,CARRY       ;préalablement définis saut si 1
GOTO ZEROW          ;exécutée si 0
BSF PORTA,2
```

```
        GOTO FINW
        ZEROW   BCF  PORTA,2
        FINW
        ENDM                        ;seul sur la ligne
```

* **Note** : Contenu du fichier P16f84

```
        LIST
; P16F84.INC  Standard Header File, Version 2.00    Microchip Technology, Inc.
        NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F84 microcontroller.  These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included.  The processor may be selected the following ways:

;       1. Command line switch:
;               C:\ MPASM MYFILE.ASM /PIC16F84
;       2. LIST directive in the source file
;               LIST   P=PIC16F84
;       3. Processor Type entry in the MPASM full-screen interface

;==========================================================================
;
;       Revision History
;
;==========================================================================

;Rev:   Date:    Reason:

;2.00   07/24/96 Renamed to reflect the name change to PIC16F84.
;1.01   05/17/96 Corrected BADRAM map
;1.00   10/31/95 Initial Release

;==========================================================================
;
;       Verify Processor
;
;==========================================================================

        IFNDEF __16F84
           MESSG "Processor-header file mismatch.  Verify selected processor."
        ENDIF

;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                           EQU     H'0000'
F                           EQU     H'0001'

;----- Register Files------------------------------------------------------

INDF                        EQU     H'0000'
TMR0                        EQU     H'0001'
PCL                         EQU     H'0002'
STATUS                      EQU     H'0003'
FSR                         EQU     H'0004'
PORTA                       EQU     H'0005'
PORTB                       EQU     H'0006'
EEDATA                      EQU     H'0008'
EEADR                       EQU     H'0009'
PCLATH                      EQU     H'000A'
INTCON                      EQU     H'000B'

OPTION_REG                  EQU     H'0081'
TRISA                       EQU     H'0085'
TRISB                       EQU     H'0086'
EECON1                      EQU     H'0088'
EECON2                      EQU     H'0089'

;----- STATUS Bits --------------------------------------------------------
```

```
IRP                             EQU     H'0007'
RP1                             EQU     H'0006'
RP0                             EQU     H'0005'
NOT_TO                          EQU     H'0004'
NOT_PD                          EQU     H'0003'
Z                               EQU     H'0002'
DC                              EQU     H'0001'
C                               EQU     H'0000'

;----- INTCON Bits --------------------------------------------------------

GIE                             EQU     H'0007'
EEIE                            EQU     H'0006'
T0IE                            EQU     H'0005'
INTE                            EQU     H'0004'
RBIE                            EQU     H'0003'
T0IF                            EQU     H'0002'
INTF                            EQU     H'0001'
RBIF                            EQU     H'0000'

;----- OPTION Bits --------------------------------------------------------

NOT_RBPU                        EQU     H'0007'
INTEDG                          EQU     H'0006'
T0CS                            EQU     H'0005'
T0SE                            EQU     H'0004'
PSA                             EQU     H'0003'
PS2                             EQU     H'0002'
PS1                             EQU     H'0001'
PS0                             EQU     H'0000'

;----- EECON1 Bits --------------------------------------------------------

EEIF                            EQU     H'0004'
WRERR                           EQU     H'0003'
WREN                            EQU     H'0002'
WR                              EQU     H'0001'
RD                              EQU     H'0000'

;=========================================================================
;
;       RAM Definition
;
;=========================================================================

        __MAXRAM H'CF'
        __BADRAM H'07', H'50'-H'7F', H'87'

;=========================================================================
;
;       Configuration Bits
;
;=========================================================================

_CP_ON                          EQU     H'000F'
_CP_OFF                         EQU     H'3FFF'
_PWRTE_ON                       EQU     H'3FF7'
_PWRTE_OFF                      EQU     H'3FFF'
_WDT_ON                         EQU     H'3FFF'
_WDT_OFF                        EQU     H'3FFB'
_LP_OSC                         EQU     H'3FFC'
_XT_OSC                         EQU     H'3FFD'
_HS_OSC                         EQU     H'3FFE'
_RC_OSC                         EQU     H'3FFF'

        LIST
```

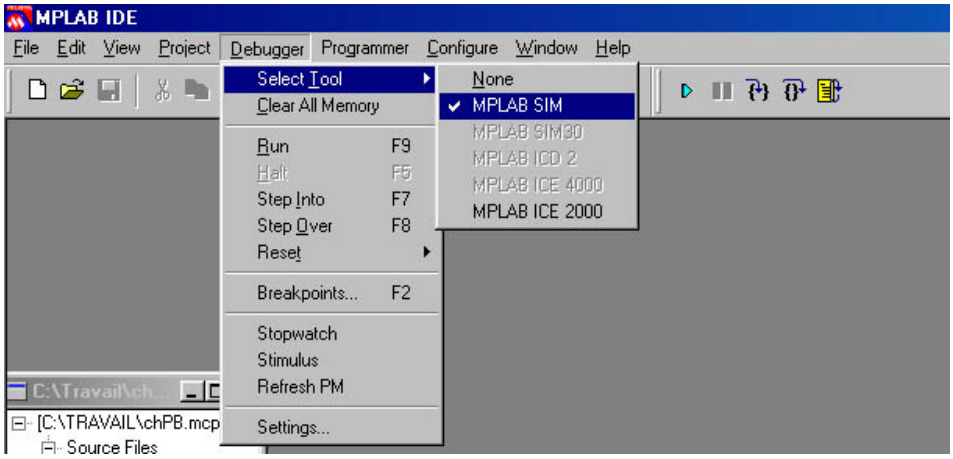
Le SIMULATEUR LOGICIEL ou DEBUGGER

Le programme ayant été écrit et compilé, il faut vérifier qu'il rempli bien les fonctions attendues. Il est bien sûr possible de programmer le composant et de lancer l'exécution pour voir si tout fonctionne comme prévu ou non , mais en cas d'échec il n'est pas facile de trouver où est l'erreur.

Il existe des émulateurs matériels qui permettent en faisant tourner le programme à sa vitesse normale de suivre l'évolution du contenu des mémoires internes et d'observer à quel moment le fonctionnement cesse d'être nominal. Ces émulateurs sont fragiles et coûteux et avant de faire appel à eux il est commode d'effectuer une exécution pas à pas du programme grâce à un simulateur logiciel Un tel outil qui exploite le fichier .OBJ crée par l'assembleur est fourni par MICROCHIP dans la suite MPLAB .Il s'agit de MPLAB SIM .

Le debugger se lance directement lorsque le listing du programme est présent à l'écran

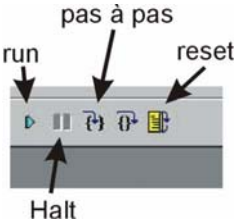
Dans la barre d'outils sélectionner **Debugger > Select Tool > MPLAB SIM**



5 nouvelles icônes sont activées dans la barre d'outils. Elles permettent de démarrer , arrêter , exécuter pas à pas le programme présent .

Il faut d'abord faire un reset :

Debugger > reset

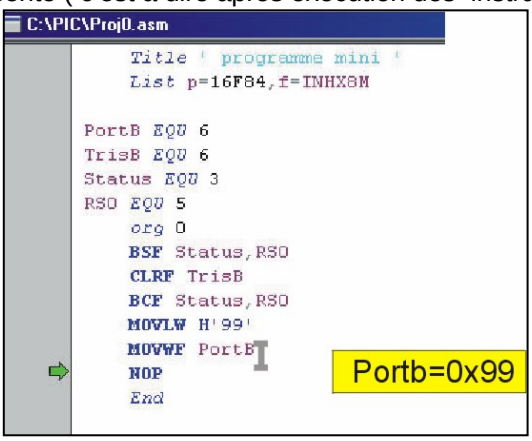


Une flèche verte apparaît face à la première instruction .

Pour lancer le programme pas à pas il suffit de cliquer sur le bouton correspondant (figure ci contre)

A chaque appui la flèche verte avance d'une ligne

En amenant le curseur sur le nom d'une variable sa valeur présente (c'est à dire après exécution des instructions précédentes) apparaît.



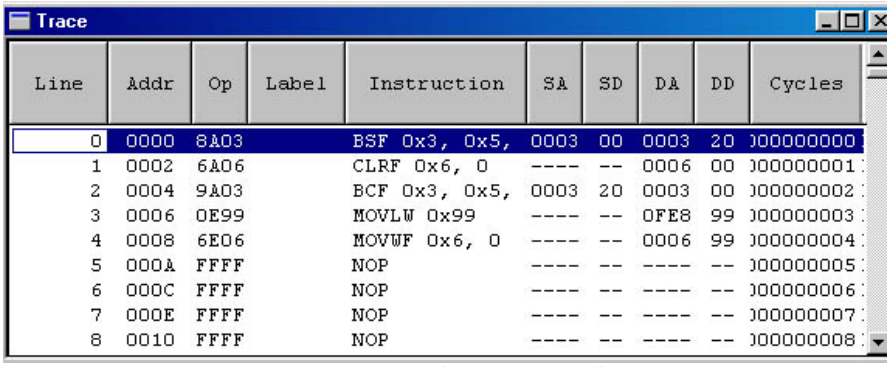
A titre d'exemple pour notre premier programme après 5 pas la flèche est arrêtée sur le NOP final et si le curseur est placé sur PORTB une fenêtre s'ouvre indiquant la valeur 99 .

Si le curseur est placé sur ce même portB immédiatement après le RESET ou après seulement 2 ou 3 instructions la valeur affichée serait 0 (sauf si le programme vient d'être exécuté , alors la valeur 99 n'a pas été effacée. Pour repartir vraiment à zéro il faut faire

Debugger > Clear all memory

Il est possible également de visualiser le

listing complet de l'exécution



Line	Addr	Op	Label	Instruction	SA	SD	DA	DD	Cycles
0	0000	8A03		BSF 0x3, 0x5,	0003	00	0003	20	00000000
1	0002	6A06		CLRF 0x6, 0	----	--	0006	00	00000001
2	0004	9A03		BCF 0x3, 0x5,	0003	20	0003	00	00000002
3	0006	0E99		MOVLW 0x99	----	--	0FE8	99	00000003
4	0008	6E06		MOVWF 0x6, 0	----	--	0006	99	00000004
5	000A	FFFF		NOP	----	--	----	--	00000005
6	000C	FFFF		NOP	----	--	----	--	00000006
7	000E	FFFF		NOP	----	--	----	--	00000007
8	0010	FFFF		NOP	----	--	----	--	00000008

Cette fenêtre peut pour plus de commodité être imprimée en cliquant sur le bouton droit de la souris , puis print .

La fenêtre comprend jusqu'à 18 colonnes. Les 8 dernières à droite ne sont utiles que si l'on dispose d'un émulateur matériel de MICROCHIP . Pour les supprimer il suffit de cliquer avec le bouton droit sur la première ligne de ces colonnes et de décocher les lignes probe de 0 à 7 dans la fenêtre vertical qui est présentée.

La seconde colonne affiche l'adresse de l'instruction

La 3eme son code en héxa

La 4eme Le label s'il y en a

La 5eme le mnémonique de l'instruction avec les adresses ou données en héxa

Les suivantes

Pour des instructions utilisant deux opérandes par exemple BSF Adresse ,data SA est l'adresse et SD la donnée

DA et DD sont les adresse et donnée résultante

Pour l'exemple présent sur la 1ere ligne l'instruction BSF STATUS,RP0 place un 1 en position 5 (soit data=0x20) dans le registre STATUS(adresse 03 soit SA=03) Au départ ce registre est à 0 soit SD=0 La destination est ce même registre soit DA=03 et la donnée à écrire 0x20 soit DB=0x20

En seconde ligne , l'nstruction CLRF TRISB n'a qu'un seul opérande , ne seront présentes que l'adresse et la donnée DA=03 et DD = 0

La 3eme ligne est de nouveau une instruction à 2 opérande BCF STATUS,RS0 , apparaîtront dans SA et SD l'adresse et le contenu de STATUS 03 et 20 (on vient d'y entrer 20) et le résultat DA=03 et DD=0 (clear)

4eme ligne MOLWF 0x99 seul opérande 0x99 donc DA=0FE8 l'adresse interne de l'accumulateur et DD=99 la donnée écrite .

Enfin pour la dernière MOVWF PORTB l'adresse du portB 06 et la donnée 99

Enfin la colonne cycle indique le temps écoulé exprimé en microsecondes .Il faut préalablement choisir la fréquence du quartz dans **Debugger >Setting > Cloc**k

Pour 4Mhz une période d'horloge = 1µS

Fenêtre de suivi (Watch Windows)

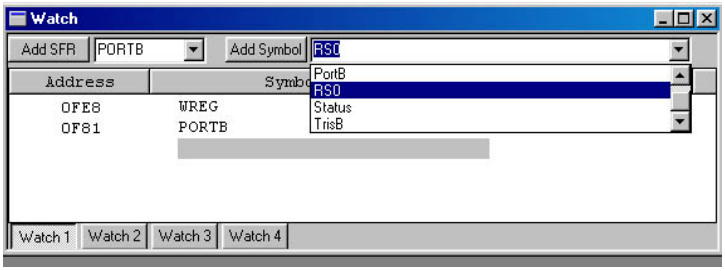
On vient de voir comment en positionnant le curseur sur le nom d'une variable on pouvait suivre l'évolution de sa valeur instructions après instructions. Ce suivi peut être obtenu de façon plus commode en utilisant la fonction Watch .

Pour activer cette fonction **View > Watch**

Une fenêtre s'ouvre .Il est possible d'afficher le comportement des registres internes du microprocesseur (SFR) ou de ceux auxquels on à donné un nom . Pour afficher les premiers utiliser l'ascenseur associé au bouton AddSFR , choisir le registre et presser le bouton .Le même mode opératoire est utilisé pour afficher un registre nommé , en utilisant le bouton AddSymbol

Les deux fenêtres Watch et Listing doivent figurer simultanément à l'écran (il faut utiliser un grand écran et une police de petite taille .)

Recompiler par **Project > Build all** (Cela s'avère parfois nécessaire pour que la fonction Watch fonctionne) puis **Debugger > Reset** et **Debugger > Step into** .(ou le bouton correspondant sur la barre d'outils)



A chaque nouvelle exécution les valeurs affichées dans la fenêtre Watch évoluent, lorsqu'une instruction modifie l'une des valeurs elle apparaît temporairement en rouge .

Points d'arrêt (BREAKPOINTS)

Pour un long programme l'exécution pas à pas est fastidieuse , il vaut mieux lance RUN et placer dans le fichier source des points d'arrêt. (BREAKPOINT)

Dans la fenêtre affichant le programme amener le curseur sur l'instruction à laquelle vous voulez vous arrêter , presser le bouton droit de la souris et choisir **Set Breakpoint**

La suppression d'un point d'arrêt se fait de la même façon (**Remove Breakpoint**)

LE PROGRAMMATEUR

Pour tester un programme il faut au préalable l'introduire dans la mémoire du composant , contrairement au 8031 il n'est pas possible de mettre en œuvre une mémoire extérieure (Emulateur de ROM) Heureusement le PIC16F84 possède une mémoire de type Flash que l'on peut remplir et effacer de nombreuses fois. IL faut pour cela utiliser un PROGRAMMATEUR .

La programmation s'effectue en série par les deux bornes PB6 et PB7 . Il est possible de trouver la procédure exacte sur le WEB .

Le programmateur lui-même est très simple peu coûteux, on peu l'acquérir dans le commerce (Par exemple SELECTRONIC) pour quelques dizaines d'euros , ou mieux le fabriquer .On trouvera un schéma et tous les logiciels nécessaires sur le site de Christian Tavernier .
www.tavernier-c.com

Il existe sur le WEB quantité de sites consacrés à ce circuit , sous Google l'appel avec l'intitulé PIC16F84 fournit plus de 16000 réponses. Bon courage ! .