5. Prise en main des PICs sur des exercices à l'aide de MPLAB (MICROCHIP).

Les différents exercices ont pour but, de mettre en oeuvre l'environnement MPLAB de MICROCHIP en utilisant un PIC 16F84 dans un premier temps en assembleur, puis en langage C avec le compilateur HI-TECH PIC C.

5.1.) Extraits principaux de la documentation.

Brochage et structure interne.

Pin Diagrams

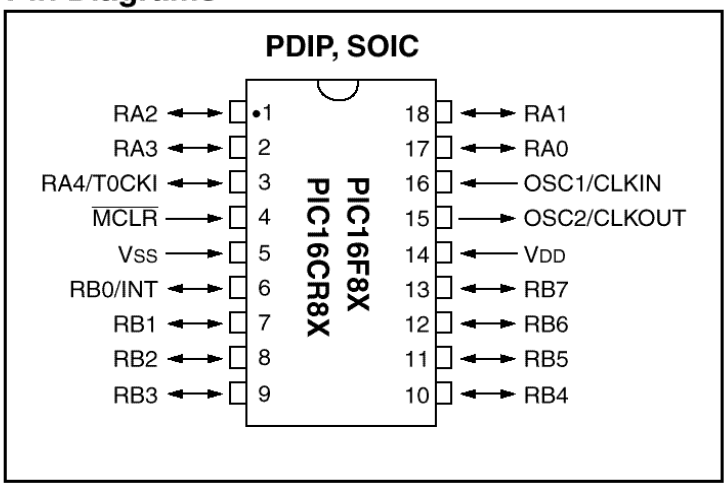
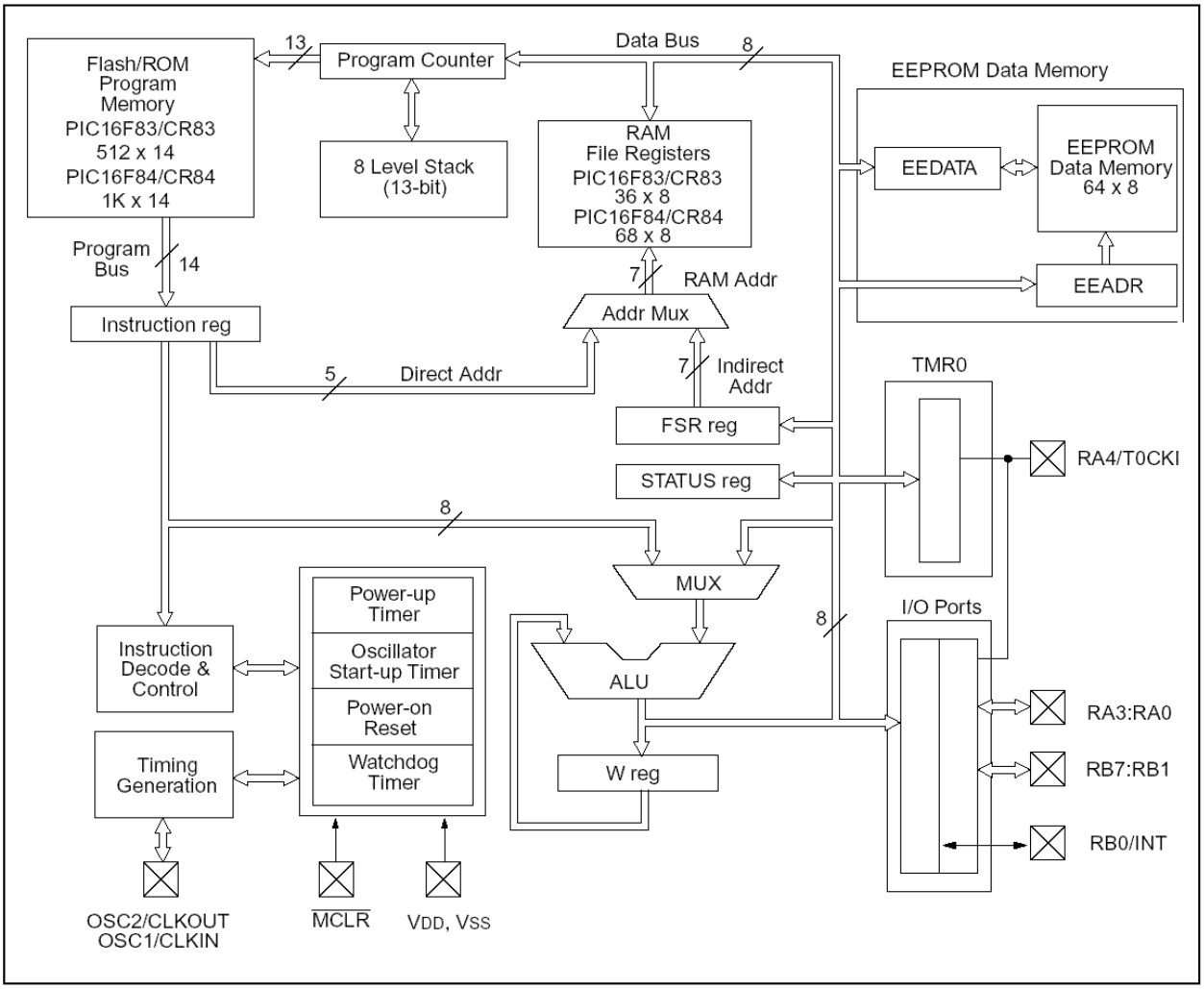


FIGURE 3-1: PIC16F8X BLOCK DIAGRAM



La mémoire programme et la pile sont organisées en mots de 14 bits (voir figure 4-2 de gauche). La Mémoire RAM comporte également les registres permettant d'accéder aux périphériques. Pour le PIC 16F84 deux pages mémoires appelées Bank 0 et Bank1 sont accessibles à travers le bit RP0 du registre d'état (STATUS).

Certaines version de PIC comportent plus de 2 Bank de registres. D'autres bits de contrôle sont alors nécessaires (Ex: RP1).

FIGURE 4-2: PROGRAM MEMORY MAP AND STACK - PIC16F84/CR84

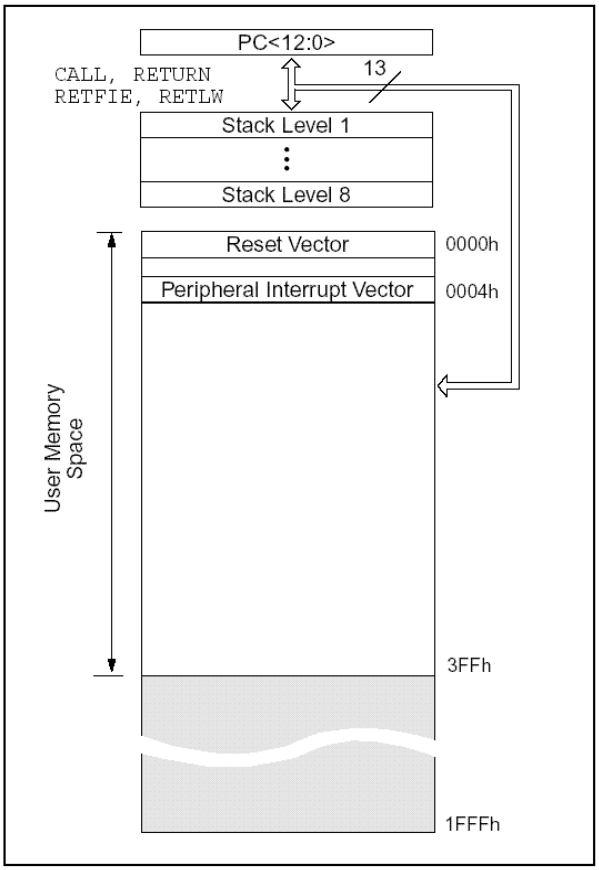
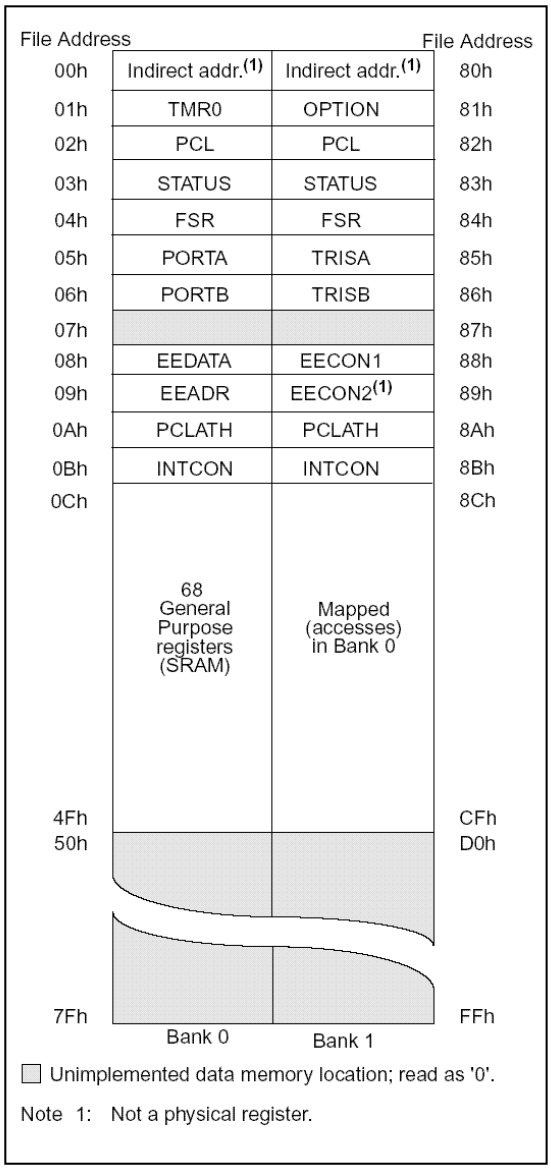


FIGURE 4-2: REGISTER FILE MAP - PIC16F84/CR84



Le programme principal commencera toujours à l'adresse 0000h. Les instructions sont directement codées sur 14 bits et n'occupent donc qu'une adresse. La pile ne comporte que 8 niveau.

La RAM utilisateur ne commence qu'en 0Ch, à la suite des registres permettant la gestion des périphériques.

Attention l'utilisation de périphérique nécessite parfois la commutation des banques à l'aide du registre d'état. Ex: Pour mettre le PORTB en sortie, puis sortir une valeur. Il faut: 1) Mettre RP0 à 1 pour permettre l'accès à TRISB. 2) Mettre TRISB à 0 pour initialiser l'ensemble du PORTB en sortie. 3) Remettre RP0 à 0 dans le registre d'état pour permettre l'accès au PORTB.

Description des registres de périphérique. Voir documentation constructeur pour détail.

TABLE 4-1 REGISTER FILE SUMMARY

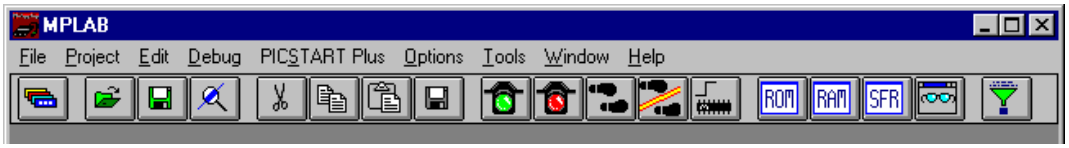
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---- ----	---- ----
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
81h	OPTION_ REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	—	—	—	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---- ----	---- ----
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---- ----	---- ----
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition.
Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.
2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ reset.
3: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

5.2.) Utilisation de l'environnement MPLAB.

Utilisez l'icone ou le menu démarrer pour lancer MPLAB.

L'environnement présente alors une barre d'icône. La signification de chaque icône apparaît en bas dans la barre d'état lorsque la souris est placée au dessus de l'icône.



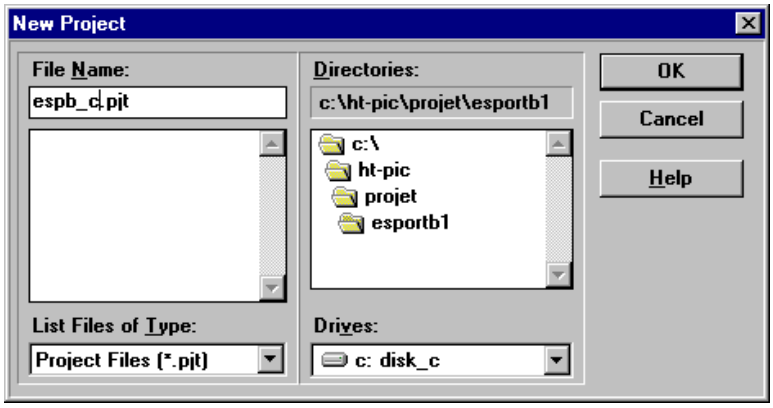
5.2.1) Création d'un projet

Attention : lors d'une création d'un projet, il n'est pas possible de créer un nouveau dossier depuis MPLAB pour contenir les fichiers créés. Il faut donc créer le dossier indépendamment, avec l'explorateur de Windows par exemple.

Création avec : Project / New Project

Ouverture d'un projet déjà existant avec Project / Open Project

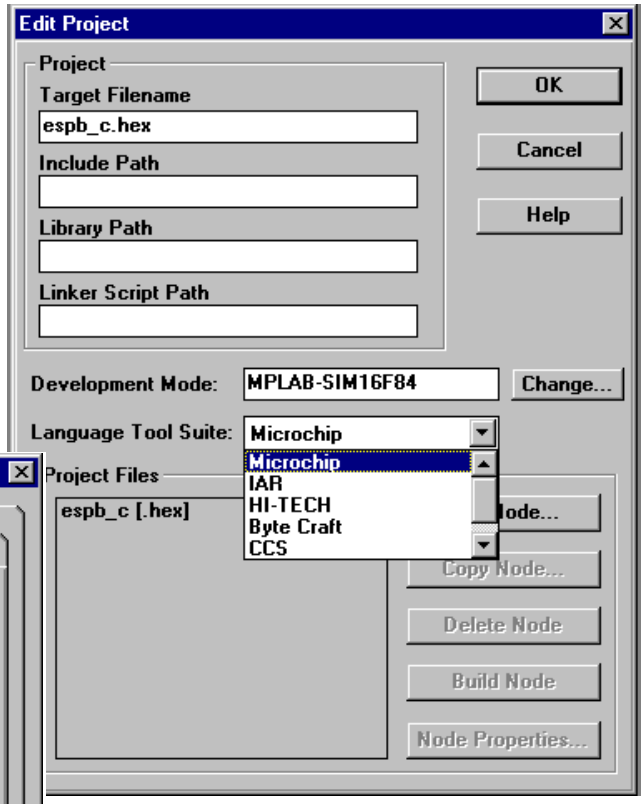
Modification avec : Project / Edit Project

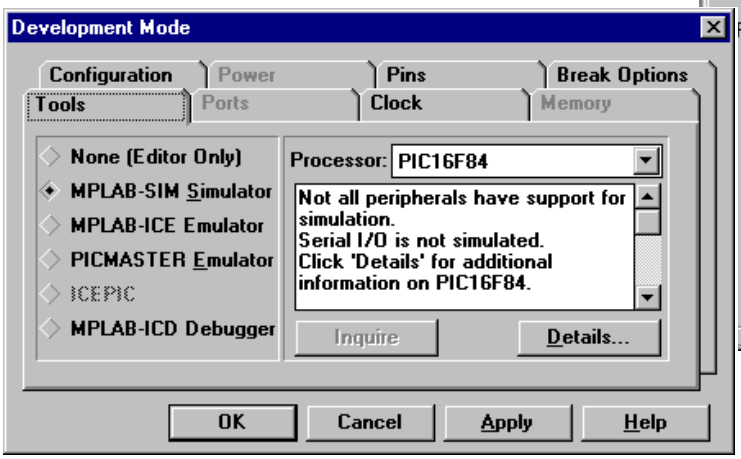


Dans tous les cas, la même boîte de dialogue est utilisée.

Une zone de sélection permet de choisir le langage à utiliser ***"Langage Tool Suite"*** *(Microchip = Assembleur PIC, HI-TECH = compilateur C...).*

Il est de même possible de changer le mode de développement et le microcontroleur utilisé à l'aide du bouton ***"Change"****.*



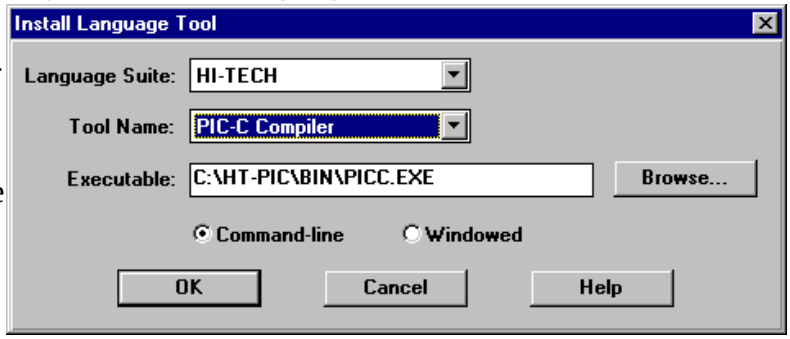


Attention: Le compilateur HI-TECH PIC C est un compilateur externe. Il doit être installé séparément et configuré avec l'option **"Project / Install Language Tool"**.

Il faut configurer le compilateur, l'assembleur et l'éditeur de liens (Linker).

C'est le même exécutable qui lance l'assemblage ou la compilation ou l'édition de liens.

La configuration est donc la même pour les 3 outils.

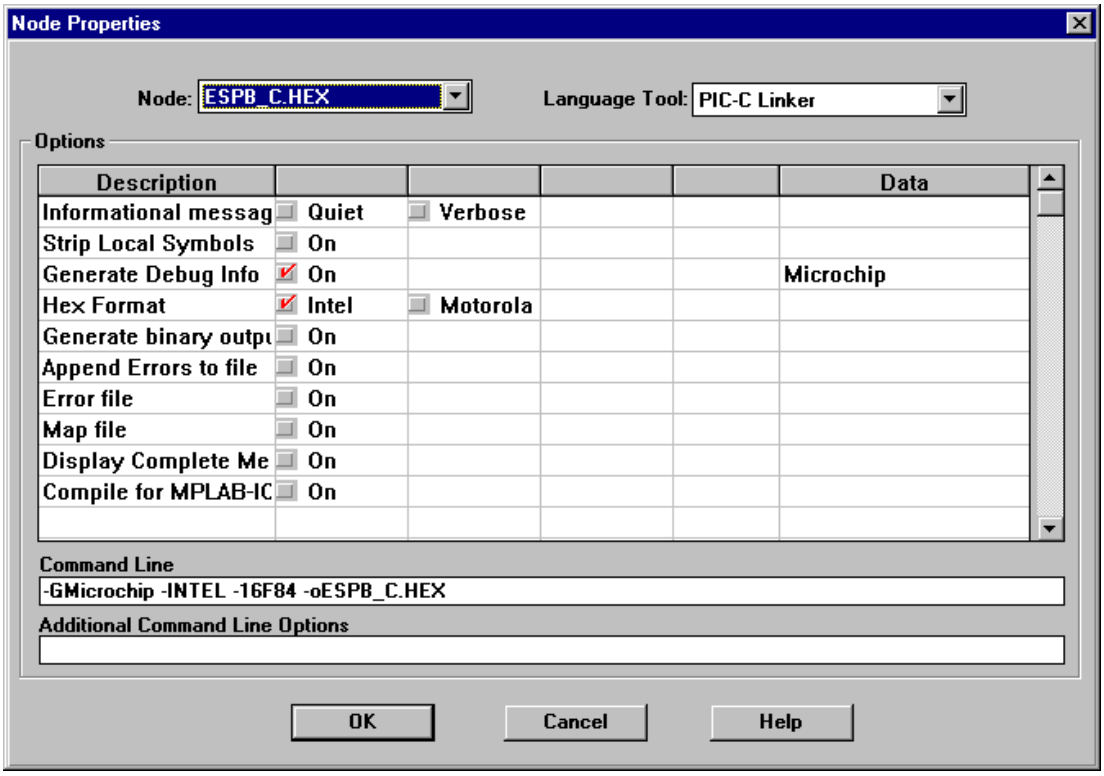


Language Suite	Tool Name	Executable	
HI-TECH	PIC-C Compiler	Dossier_d'installation\BIN\PICC.EXE	Command Line
	PIC-C Assembler	Dossier_d'installation\BIN\PICC.EXE	Command Line
	PIC-C Linker	Dossier_d'installation\BIN\PICC.EXE	Command Line

a) Validez la suite HI-TECH

b) Sélectionnez le fichier destination **(.HEX)**. **Pour pouvoir ajouter des fichiers source, il faut commencer par définir les propriétés du nœud du projet (fichier cible ou exécutable), avec le bouton « Node Properties… », et sélectionnez l'éditeur de liens (PIC-C Linker).**

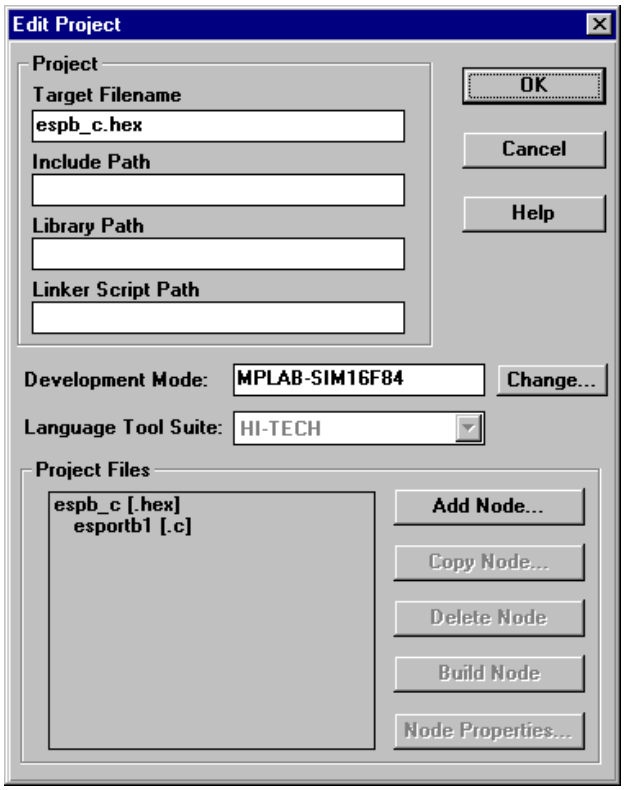
Les propriétés du fichier cible dépendent de la suite logicielle. *Voir ci-dessous avec la suite HI-TECH.*



c) Cochez **"Generate Debug Info"** et ajoutez **"Microchip"** dans le champ Data.

Rem: La ligne de commande en bas se complète dès que le curseur est déplacé.

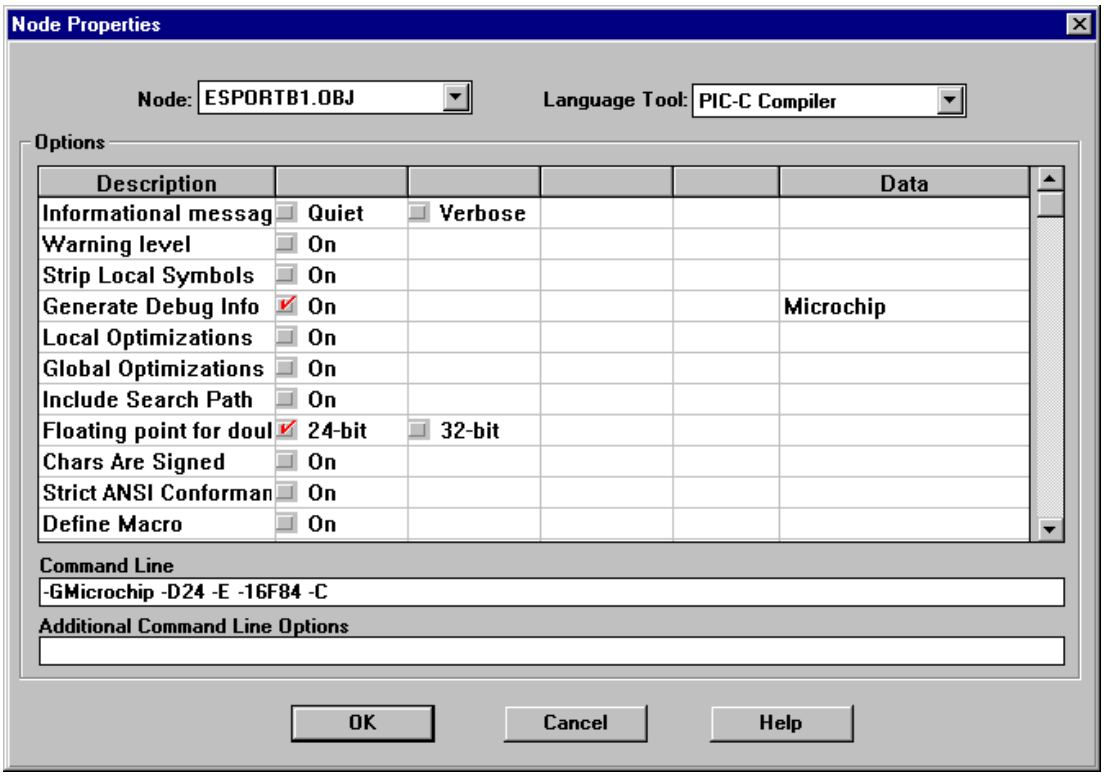
d) Vous pouvez ajouter le fichier source en C avec l'option **"Add Node"** de la fenêtre **"Edit Project"**.



Il faut alors sélectionner le fichier C et éditer ces propriétés avec "**Node Properties**".

De même il faudra cochez **"Generate Debug Info"** et ajoutez **"Microchip"** dans le champ Data de la fenêtre de propriété.

Attention: dans ce cas le fichier sera le **(.OBJ)** et l'outil le "**PIC C Compiler**". (Voir page suivante).



5.2.2) Création du fichier source.

Utilisez l'option "**File / New**" ou ouvrez un fichier existant pour reprendre l'entête et sauvegardez le immédiatement sous le nom (.C) défini précédemment.

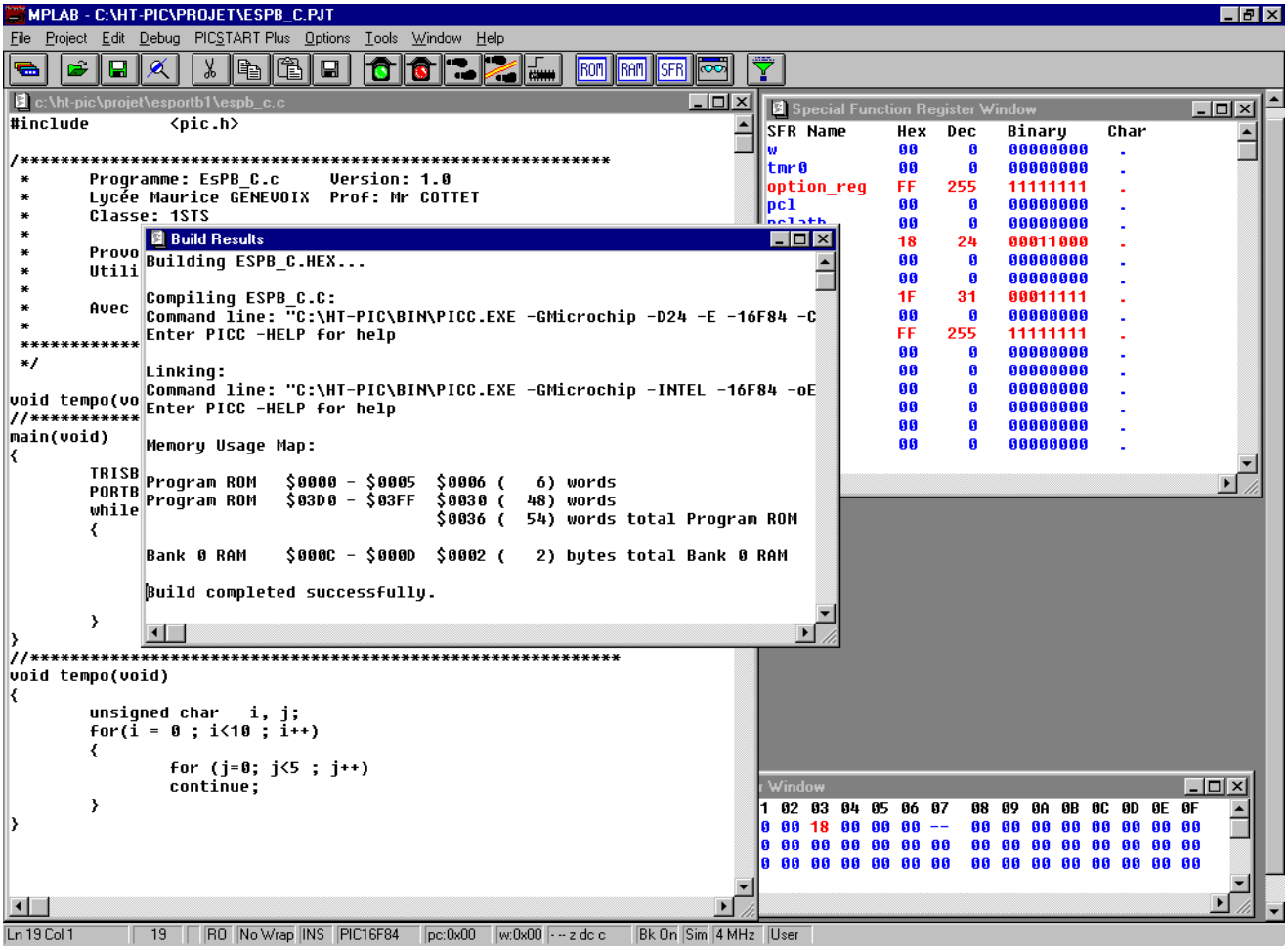
Exercices avec clignotement lumineux sur 8 LEDs reliées au PortB.

```
 #include   <pic.h>

/*********************************************************
 *  Programme: EsPB_C.c        Version: 1.0
 *  Lycée Maurice GENEVOIX     Prof: Mr COTTET
 *  Classe: 1STS
 *
 *  Provoque le clignotement des LEDs du Port B,
 *      Utilisable sur PIC16F84.
 *
 *  Avec MPLAB et compilateur C  HI-TECH.
 *
 *********************************************************
 */

void tempo(void);
//****************** Programme Principal ******************
main(void)
{
    TRISB = 0;          /* PortB en sortie */
    PORTB = 0;
    while(1)
    {
        PORTB = 0x55;           /* Allumage des LEDs paires */
        tempo();
        PORTB = 0xaa;           /*Allumage des LEDs impaires */
        tempo();
    }
}
//*********************************************************
void tempo(void)
```

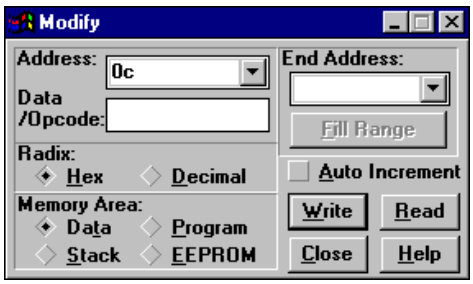
```
{
    unsigned char      i, j;
    for(i = 0 ; i<10 ; i++)
    {
        for (j=0; j<5 ; j++)
        continue;
    }
}
```

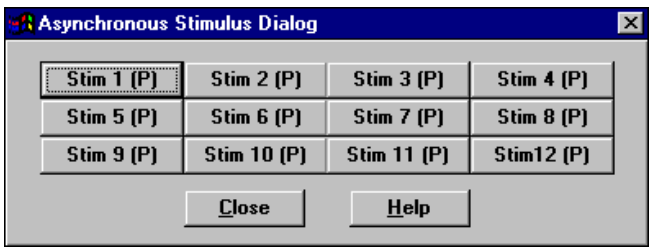
Vérifiez la compilation avec "**Project / Buil All**".
Une fenêtre "**Build Result**" annonce la réussite ou l'échec de la compilation.

S'il n'y a pas d'erreurs, ouvrez une fenêtre avec les registres (SFR) et une avec la zone RAM pour le débug et utilisez les icônes (Reset, pas à pas, Run, Halt, ...) pour vérifier ou débugger le programme.



Attention: pour modifier des valeur en mémoire il faut cliquer sur l'octet à modifier, puis utiliser le click droit (menu contextuel "**Fill Register**").
De même pour modifier des entrées il faut appeler "Debug / Simulator Stimulus / Asynchronous Stimulus).





5.3.) Exercice complet sur le Filancemètre (Organisation matérielle et logicielle, utilisation du PIA).