COMPILATEUR C HITECH POUR PIC UTILISÉ AVEC MPLAB

Version 7.85 (quelques indications sur les versions 8 et 9)

1) GÉNÉRALITÉS 3

2) EXTENSIONS AU C ANSI / PARTICULARITÉS DU COMPILATEUR 4

2.1) Divergence par rapport au C ANSI 4

2.2) Données 4

Donnée de type bit 4

Codage des données sur plusieurs octets 5

Bases pour les entiers 5

Champs de bits 5

2.3) Déclaration d'une donnée placée en registre avec adresse 5

Déclaration pour une donnée d'un octet 5

déclaration pour une donnée de type bit 5

2.4) Qualificateurs (ou modificateurs) de types de données 6

Qualificateur « persistent » 6

Qualificateur « bank » 6

2.5) Pointeurs 6

Pointeurs sur données en RAM 6

Pointeurs sur données en ROM ou en RAM 7

2.6) interruption en C 7

2.7) Insertion de lignes en langage d'assemblage dans un programme source en C 7

2.8) Gestion de la mémoire EEPROM 8

3) FICHIERS EN-TÊTE FOURNIS 8

3.1) Utilisation des fichiers « en-tête » 8

3.2) Nom des registres et des bits 8

4) INTERFAÇAGE AVEC MPLAB 9

4.1) Interfaçage avec MPLAB 5.x 9

4.2) Interfaçage avec MPLAB 6 et 7 10

5) ÉDITION DE LIENS ET PROGRAMME DE DÉMARRAGE 10

5.1) Programme de démarrage 10

5.2) Options et configuration de l'édition de liens avec MPLAB 5.x 10

5.3) Liaison de programmes objet dans les sources sont en langage d'assemblage en langage C 11

6) LES OPTIONS DE COMPILATION ET D'ASSEMBLAGE AVEC MPLAB 5.X 12

6.1) Compilateur + assembleur 12

6.2) Assembleur seul 13

VUE D'ENSEMBLE DE LA COMPILATION 15

EDITION DE LIENS 15

Programme de démarrage 15

Options de l'édition de liens 18

1) GÉNÉRALITÉS

Ce qui est présenté par la suite comme le « compilateur HiTech » est un ensemble de logiciels DOS (pour les versions 7 et antérieures) ou win32 pour la version 9 (dernière en date à la rédaction de ce document (février 2006). Il est notamment constitué de :

- un compilateur proprement dit (pré processeur, analyseur syntaxique, générateur de code)
- un assembleur
- un éditeur de liens
- quelques utilitaires (conversion de format du fichier final Objtohex, etc.)
- un gestionnaire de bibliothèque

Le compilateur HiTech peut être utilisé avec tous les PIC d'entrée de gamme et de milieu de gamme et avec certains PIC haut de gamme. Un compilateur spécifique est disponible pour les PIC haut de gamme de la série 18C.
Le présent document décrit surtout l'utilisation du compilateur avec les PIC milieu de gamme.

Le compilateur HiTech peut être lancé :

- directement par la ligne de commande DOS (nom du prog exécutable suivi d'options et des noms de fichiers à utiliser)
- depuis l'environnement de développement intégré (EDI) HiTech *Le gestionnaire de bibliothèque ne peut être lancé depuis l'EDI de HiTech*.
- depuis l'EDI MPLAB de MicroChip, après configuration de ce dernier (*voir manuel de l'utilisateur page 66*). *Le gestionnaire de bibliothèque ne peut être lancé depuis MPLAB*. MPLAB génère automatiquement la ligne ce commande lorsqu'une compilation est demandée.

Puisque MPLAB doit être utilisé pour la simulation, la programmation et le déboguage (avec émulateur, module ICD, etc.), le plus simple est d'utiliser aussi MPLAB pour l'édition des fichiers source et pour le lancement de la compilation.
Cette façon de procéder présente l'inconvénient suivant :
lors de la construction d'un projet, tous les fichiers source sont compilés, même si certains avaient déjà été compilés auparavant, sans aucun modification apportée au fichier source depuis la dernière compilation.

Cet inconvénient est mineur pour par rapport à l'utilisation de deux EDI.

Une interactivité entre la fenêtre des messages et la ou les fenêtres d'édition (un double clic sur un message d'erreur renvoie à la ligne de l'erreur dans la fenêtre d'édition) existe. Avec la version 7 ceci est possible à condition d'insérer les lignes suivantes dans l'autoexec.bat (Windows 95 ou 98) :

```
SET HTC_ERR_FORMAT=Error[ ] file %%f %%l : %%s
SET HTC_WARN_FORMAT=Warning[ ] file %%f %%l : %%s
```

Bien que le compilateur HiTech puisse être utilisé pour produire un fichier exécutable à partir uniquement de fichiers source en langage d'assemblage, cette possibilité n'est pas envisagée ici. On ne s'intéresse qu'aux cas où les fichiers source sont en langage C (tous ou au moins un).

L'enchaînement des différents modules du compilateur est automatique. L'utilisateur doit configurer depuis MPLAB le compilateur pour chaque fichier source et l'éditeur de liens. Lors de la compilation, MPLAB envoie les commandes et les options adéquates sur la ligne de commande.
Une figure en annexe résume les différents modules logiciels utilisés et les fichiers utilisés ou créés.

Pour l'utilisation du compilateur HiTech avec MPLAB, voir le document « MPLAB de Microchip ».
Pour plus d'information, voir les documents de Microchip « MPLAB User's Guide » et « MPLAB IDE Project Tutorials for Third Party Tools » (réf DS51234).

2) EXTENSIONS AU C ANSI / PARTICULARITÉS DU COMPILATEUR

2.1) DIVERGENCE PAR RAPPORT AU C ANSI

Les fonctions ne peuvent être utilisées de façon récursive (limitation due aux faibles ressources du PIC, notamment une pile de quelques niveaux seulement).
Avec les PIC milieu de gamme et haut de gamme, une fonction correspond à un sous programme. Lors de l'exécution, seule l'adresse de retour est placée dans la pile ; les paramètres et variables locales ne sont pas placées dans la pile[1].
Une fonction peut être appelée par le programme principal ou par le gestionnaire d'interruption.

2.2) DONNÉES

En plus des types de données du C ANSI, le compilateur dispose du type **bit**.
La taille des données de types float ou double est inférieure à celle de C ANSI (*voir PIC C Compiler User's Guide page 120*)

DONNÉE DE TYPE BIT

Ce type de donnée est uniquement utilisable pour une donnée de classe de mémorisation statique : donnée globale ou globale cachée ou locale rémanente (déclaration précédée de static).
Une fonction ne peut avoir en paramètre une donnée de type bit mais peut retourner une valeur de ce type.
Il n'est pas possible de déclarer un pointeur sur une donnée de type bit.

Ce type de donnée permet d'économiser de la RAM, ce qui est très important avec les PIC d'entrée de gamme.

Déclaration	Utilisation
Comme n'importe quelle donnée. Ex : bit Fonctionnement ; /*Fonctionnement ne peut valoir que 2 valeurs 0 ou 1*/	Comme n'importe quelle donnée. Ex : if (Fonctionnement == NORMAL) {...} /* NORMAL vaut 1 par exemple */

Pour la déclaration d'un bit placé dans un registre avec adresse, voir ci-dessous.

[1] Pour l'implémentation des fonctions avec les PIC d'entrée de gamme, voir le manuel de l'utilisateur Hi-Tech §5.28, page 140.

Utilisation avec MPLAB 5.x

MPLAB 5.x ne peut réaliser un débogage correct avec des données de type bit à cause du codage particulier de l'« adresse » pour une donnée de ce type.

CODAGE DES DONNÉES SUR PLUSIEURS OCTETS

L'octet le moins significatif (poids faible) est placé à l'adresse la plus basse. Méthode petit boutiste (little endian).

BASES POUR LES ENTIERS

En plus des bases du C ANSI, le compilateur HiTech accepte la base 2.
Notation : 0bnombre ou 0Bnombre. Ex : 0b01110011

CHAMPS DE BITS

Le LSB est le 1er dans la définition.

2.3) DÉCLARATION D'UNE DONNÉE PLACÉE EN REGISTRE AVEC ADRESSE

L'utilisateur a peu à manipuler de telles déclarations. Toutes les déclarations pour les registres SFR sont dans le fichier en-tête fourni pic.h.

DÉCLARATION POUR UNE DONNÉE D'UN OCTET

syntaxe : static volatile unsigned char [bankx] <NOM_REGISTRE> @ <adresse_registre> ;
bankx est inutile si le registre est en banque 0. Dans les autres cas, bankx vaut bank1, bank2 ou bank3

exemple : static volatile unsigned char TMR0 @ 0x01 ;

DÉCLARATION POUR UNE DONNÉE DE TYPE BIT

syntaxe : static volatile bit <NOM_BIT> @(unsigned) &<NOM_REGISTRE>*8+<N°_BIT>

l'« adresse » est mise au format d'un entier sur 16 bits. Elle est égale à l'adresse de NOM_REGISTRE + une valeur correspondant au n° du bit.

exemple : static volatile bit RC7 @(unsigned) &PORTC*8+7 /* PORTC doit avoir été déclaré */

utilisation de l'adresse par le compilateur

Le compilateur calcule l'adresse du registre où est rangée la variable en divisant par 8 l' « adresse », puis détermine la position du bit en faisant un masquage.

ex :

```
if (PhaseInst == 0)
        est traduit en partie par :
btfsc   (_PhaseInst/8),(_PhaseInst)&7
```

2.4) QUALIFICATEURS (OU MODIFICATEURS) DE TYPES DE DONNÉES

En plus des qualificateurs du C ANSI (extern, const, volatile, static, register), le compilateur Hi-Tech dispose des qualificateurs **persistent** et **bank1**, **bank2** et **bank3**. Ces qualificateurs ne sont utilisables que pour les variables de classe de mémorisation statique.
Si ces qualificateurs sont utilisés pour les variables locales d'une fonction, il faut utiliser le mot clé static.

QUALIFICATEUR « PERSISTENT »

Ce qualificateur est utilisé pour les variables qui doivent être conservées à travers des RàZ ou des cycles de marche / arrêt / marche.
Ces variables sont mémorisées dans une zone particulière de la mémoire (RAM sauvegardée). Ceci n'est possible que lorsque la configuration matérielle le permet.

QUALIFICATEUR « BANK »

L'éditeur de liens ne permet pas de placer automatiquement les données définies par l'utilisateur dans une banque ou une autre en fonction de l'espace occupé. L'utilisateur doit donc lui-même placer ces données dans les différentes banques avec les qualificateurs bank1, bank2 et bank3. Les données définies sans qualificateur bankx sont placées dans la banque 0.

Ex :

```
bank1 union
{unsigned char Col[5][12] ;
unsigned char Tab[60] ;
} MemAff ;
```

2.5) POINTEURS

Le format des pointeurs dépend de la gamme du PIC. Seuls sont mentionnés ici les pointeurs pour les PIC milieu de gamme. Pour les autres PIC, voir la doc HiTech compilateur 7.85 page 126 et suivantes.

POINTEURS SUR DONNÉES EN RAM

Ce sont des pointeurs de taille d'un octet. Ils ne peuvent accéder qu'à la banque 0 ou 1. Pour accéder à la banque 2 ou 3, il faut déclarer le pointeur avec le qualificateur bank2 ou bank3
ex : bank2 unsigned char* ptVar1 ;

Un même pointeur ne peut accéder qu'à 2 banques : 0-1 ou 2-3.

POINTEURS SUR DONNÉES EN ROM OU EN RAM

Ce sont des pointeurs de taille 2 octets. Ils peuvent accéder à n'importe quel emplacement mémoire. Ils doivent être déclarés avec le qualificateur const.

Ex : const unsigned char* PtDebutMsg ;

Les pointeurs « const » peuvent être utilisés en paramètre d'une fonction.

Il est impossible de modifier la donnée pointée.

```
Ex : const char* cptr ;
...
Var = *cptr ; // accepté
*cptr = Var ; // refusé
```

2.6) INTERRUPTION EN C

Le nom de la fonction doit être précédée de **interrupt**. La fonction ne doit retourner aucune valeur et ne doit pas avoir de paramètre.

En tête : void interrupt TraitementInter(void)

Le PIC ne sauvegarde automatiquement dans la pile que l'adresse de retour. Le compilateur se charge de sauver les différents registres et variables nécessaires puis de les restituer en fin de programme de traitement d'interruption.

La validation/inhibition des interruptions peut s'effectuer à partir des bits prédéfinis ou de macros prédéfinies dans le fichier pic.h

Validation globale des interruptions	GIE = 1 ; /* GIE : bit prédéfini */ ei() ; /* macro prédéfinie (**e**nable **i**nterrupt)*/
inhibition globale des interruptions	GIE = 0 ; di() ; /* **d**isable **i**nterrupt */

2.7) INSERTION DE LIGNES EN LANGAGE D'ASSEMBLAGE DANS UN PROGRAMME SOURCE EN C

Des lignes en langage d'assemblage peuvent être insérées dans un programme source en C de 2 manières :

- entre les directives #asm et #endasm
- avec la fonction asm(" ")

La fonction asm(" ") est utilisée pour intégrer une simple instruction dans le code source en C.
Ex : asm("movlw 33");

Les directives #asm et #endasm permettent d'intégrer plusieurs instructions.

Attention : les instructions en langage d'assemblage entrées avec les directives précédentes ne font pas partie de manière syntaxique du C et elles ne se conforment pas aux règles de contrôle de flux. Par exemple, on ne peut utiliser un bloc #asm dans une structure if.
Pour intégrer correctement des lignes en langage d'assemblage dans des structures de contrôle de flux, il faut utiliser la fonction asm().

L'insertion de lignes en langage d'assemblage dans un programme source en C doit rester rare et utilisée avec beaucoup de précautions (pas d'interférence avec les registres utilisés par le compilateur, etc.)

2.8) GESTION DE LA MÉMOIRE EEPROM

Le compilateur et l'assembleur HiTech ne disposent pas de directive pour placer des constantes en mémoire EEPROM lors de la programmation.

La documentation de la version 7.85 mentionne, page 262, les fonctions eeprom_read() et eeprom_write() pour la lecture et l'écriture de la RAM lors de l'exécution du programme.. Les en-têtes de ces fonctions sont bien déclarés dans pic1687x.h mais elles ne sont pas implantées en bibliothèques.
L'examen des bibliothèques montre que deux fonctions eeread() et eewrite() sont précompilées et placées en bibliothèque mais leurs en-têtes ne sont pas dans un fichier .h.

3) FICHIERS EN-TÊTE FOURNIS

3.1) UTILISATION DES FICHIERS « EN-TÊTE »

De nombreux fichiers « en-tête » contenant les définition de tous les registres et de beaucoup de bits de contrôle / état sont fournis avec le compilateur.

Avec MPLAB, le fichier en-tête peut être inclus avec la directive #include <pic.h>. Ce fichier contient des directives pour inclure le fichier en-tête spécifique utilisé d'après un symbole défini par MPLAB selon le PIC choisi dans la boîte de paramétrage du projet.

Il est aussi possible de nommer directement dans le fichier source le fichier en-tête spécifique.
Ex : #include <pic1687x.h> pour un PIC16F877.

Avec MPLAB, il est inutile de spécifier le chemin pour les fichiers « include » dans la boîte de paramétrage du projet.

3.2) NOM DES REGISTRES ET DES BITS

Le compilateur reprend les noms des registres définis dans la documentation Microchip, sauf pour quelques registres :

Appellation Microchip	Appellation HiTech
OPTION_REG	OPTION
AD/$\overline{GONE}$	ADGO

De nombreux bits de contrôle / état sont définis (avec le type bit –voir ci-dessus-).
Les bits du registre SSPSTAT (registre de commande / état) de la liaison série synchrone sont préfixés avec STAT.

Appellation Microchip	Appellation HiTech
SMP	STAT_SMP
...	...
BF	STAT_BF

Pour voir le détail des bits définis, éditer le fichier en-tête spécifique au PIC utilisé.

4) INTERFAÇAGE AVEC MPLAB

4.1) INTERFAÇAGE AVEC MPLAB 5.X

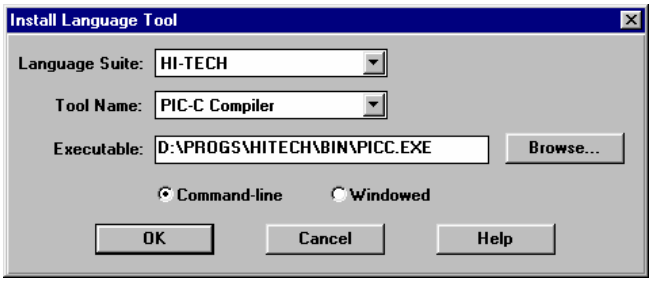
Seules les versions 7 et 8 permettent un interfaçage avec MPLAB 5.x (déboguage « in circuit » uniquement avec ICD1). Lorsqu'on lance une compilation depuis MPLAB, une ligne de commande DOS est générée. Cette ligne de commande est du style : C:\HT-PIC\BIN\PICC.EXE -FAKELOCAL -G -E -ICD -16F877 -oTEST.HEX TEST.OBJ. On voit en début le nom du fichier exécutable qui réalise la compilation suivi des paramètres.

La configuration n'est à effectuer qu'**une fois** pour chaque suite logicielle. Il s'agit de donner le nom et le chemin d'accès des logiciels suivants : assembleur, compilateur et éditeur de liens.

Depuis MPLAB, la configuration s'effectue avec la commande Project / Install Language Tool.
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.

Pour les versions 7 et 8 du compilateur.

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



La version 9 ne dispose pas des mêmes paramètres sur la ligne de commande que les versions antérieures. Certains anciens paramètres nécessaires au fonctionnement avec MPLAB pour le déboguage ne sont pas reconnus par la version 9.

4.2) INTERFAÇAGE AVEC MPLAB 6 ET 7

Les versions 8 et 9 du compilateur sont prévues pour s'interfacer avec MPLAB 6 et 7 (déboguage « in circuit » uniquement avec ICD2).
A compléter

5) ÉDITION DE LIENS ET PROGRAMME DE DÉMARRAGE

L'édition de liens est à configurer avant la compilation avec MPLAB, ce qui explique qu'elle est présentée ici avant la compilation.

L'édition de liens permet de réunir tous les fichiers objet du projet. Elle insère de plus en début du programme exécutable un programme de démarrage.

5.1) PROGRAMME DE DÉMARRAGE

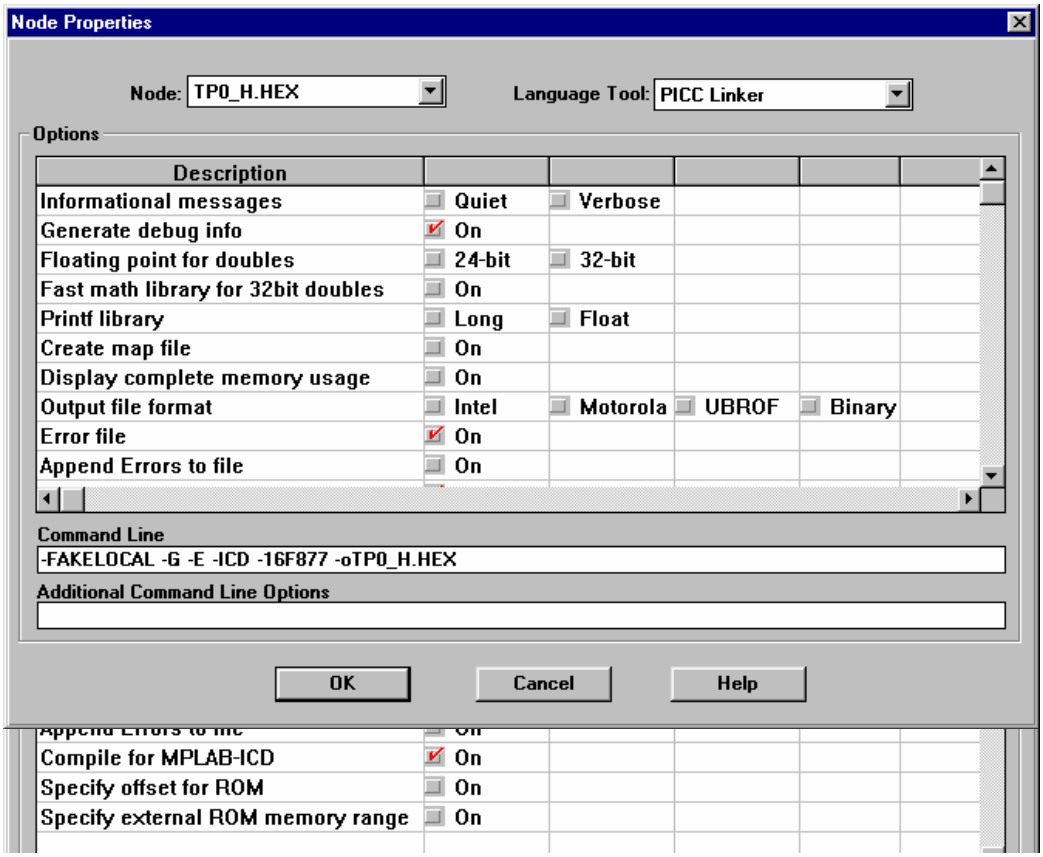
Le programme de démarrage permet si nécessaire d'initialiser à 0 les variables globales non initialisé (conformément au C ANSI), de recopier de la ROM vers la RAM les valeurs des données globales initialisées et de sauter au programme utilisateur main.
L'insertion du programme de démarrage est transparente à l'utilisateur. Une version parmi plusieurs est choisie en fonction du PIC utilisé, de la présence ou non de données de classe de mémorisation statique.

Pour plus de détail sur le programme de démarrage, voir l'annexe et le guide de l'utilisateur du compilateur C HiTech 7.85 pages 144 et suivantes.

5.2) OPTIONS ET CONFIGURATION DE L'ÉDITION DE LIENS AVEC MPLAB 5.X

L'éditeur de liens dispose de plusieurs options.
La configuration de l'édition de liens s'effectue avec MPLAB avec la boîte de dialogue Edit Project, en sélectionnant le fichier cible .hex et en cliquant sur Node Properties...
Certaines options peuvent être fixées en cochant des cases dans la boîte de dialogue; les autres doivent être écrites dans une case spécifique de cette boîte de dialogue.

Les options indispensables sont montrées avec la boîte de dialogue (MPLAB version 5.5).



Generate Debug Info peut être supprimée pour la compilation finale avant programmation définitive, après tous les essais. Le nom dans la colonne Data correspond au nom du fichier créé.

L'option **Compile for MPLAB ICD** doit être cochée lorsque cela est nécessaire.

Voir en annexe la relation entre les options de cette boîte de dialogue et les options décrites dans le manuel de l'utilisateur HiTech.

Pour plus d'informations sur les options, voir le manuel de l'utilisateur du compilateur HiTech pages 92 et suivantes.

5.3) LIAISON DE PROGRAMMES OBJET DANS LES SOURCES SONT EN LANGAGE D'ASSEMBLAGE EN LANGAGE C

La liaison de programmes avec des sources de différents langages[2] nécessite une connaissance assez fine du fonctionnement du compilateur.
Il faut notamment :

- respecter les modifications des identificateurs lors de la compilation (_ ajouté en début. Ex Calcul en C → _Calcul en langage d'assemblage)
- placer les différentes parties du programme en langage d'assemblage dans les sections prédéfinies du compilateur
- respecter les passages de paramètres et les retours de valeurs pour les fonctions

[2] les programmes en langage d'assemblage doivent être assemblés avec l'assembleur intégré du compilateur HiTech

Tous ces points ne sont pas abordés dans cette présentation rapide.
Voir le manuel de l'utilisateur HiTech version 7.8 :

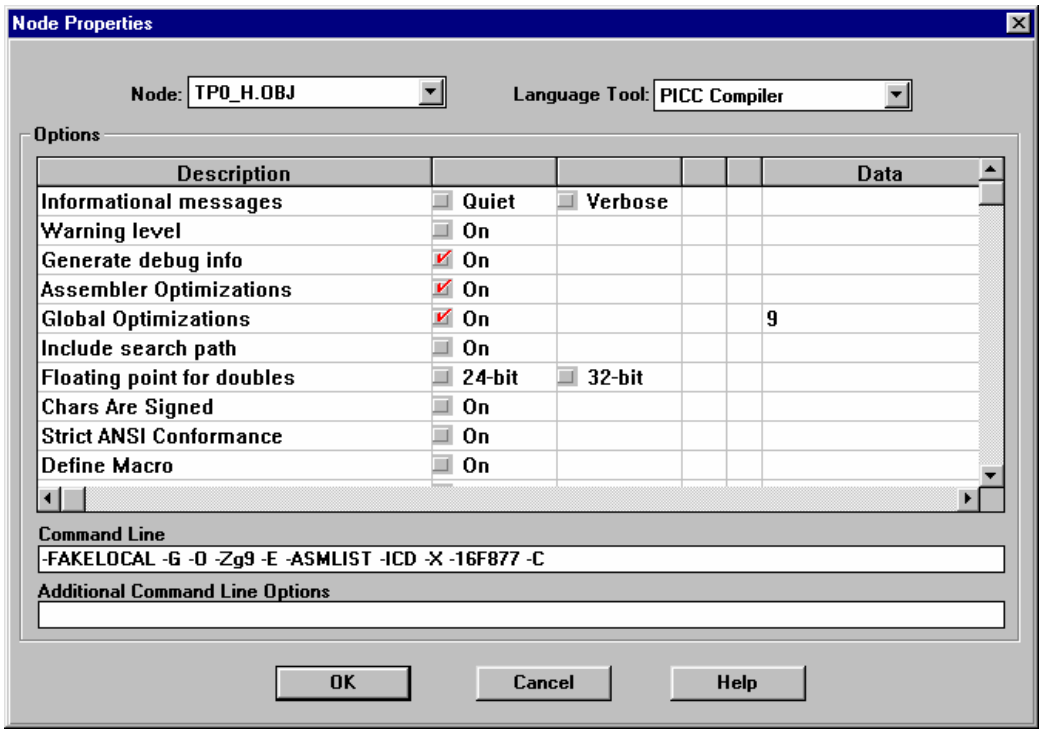
- pour les généralités : §5.20.1 et 5.20.2, page 133
- pour les sections prédéfinies du C : §5.30, page 141
- pour la déclaration des sections en langage d'assemblage : §6.3.11, page 161

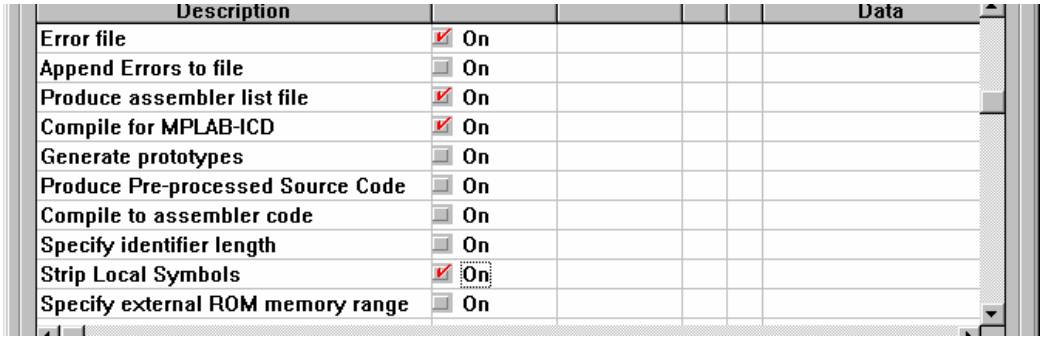
6) LES OPTIONS DE COMPILATION ET D'ASSEMBLAGE AVEC MPLAB 5.X

6.1) COMPILATEUR + ASSEMBLEUR

Le compilateur + l'assembleur sont utilisés pour un programme source en langage C.
Le compilateur (+ l'assembleur) dispose de plusieurs options.
La configuration de la compilation s'effectue avec MPLAB avec la boîte de dialogue Edit Project, en sélectionnant un des fichier source du projet .c et en cliquant sur Node Properties.
Certaines options peuvent être fixées en cochant des cases dans la boîte de dialogue; les autres doivent être écrites dans une case spécifique de cette boîte de dialogue.

Les options standard sont montrées avec la boîte de dialogue (MPLAB version 5.5).





Description		Data
Error file	On	
Append Errors to file	On	
Produce assembler list file	On	
Compile for MPLAB-ICD	On	
Generate prototypes	On	
Produce Pre-processed Source Code	On	
Compile to assembler code	On	
Specify identifier length	On	
Strip Local Symbols	On	
Specify external ROM memory range	On	

Les options à cocher pour une utilisation avec MPLAB sont :

- **Generate Debug Info**. Le nom dans la colonne Data correspond au nom du fichier créé. Ce nom peut être omis. Cette option peut être non cochée pour la compilation finale, avant programmation définitive, après tous les essais.
- **Compile for MPLAB-ICD** (si nécessaire)

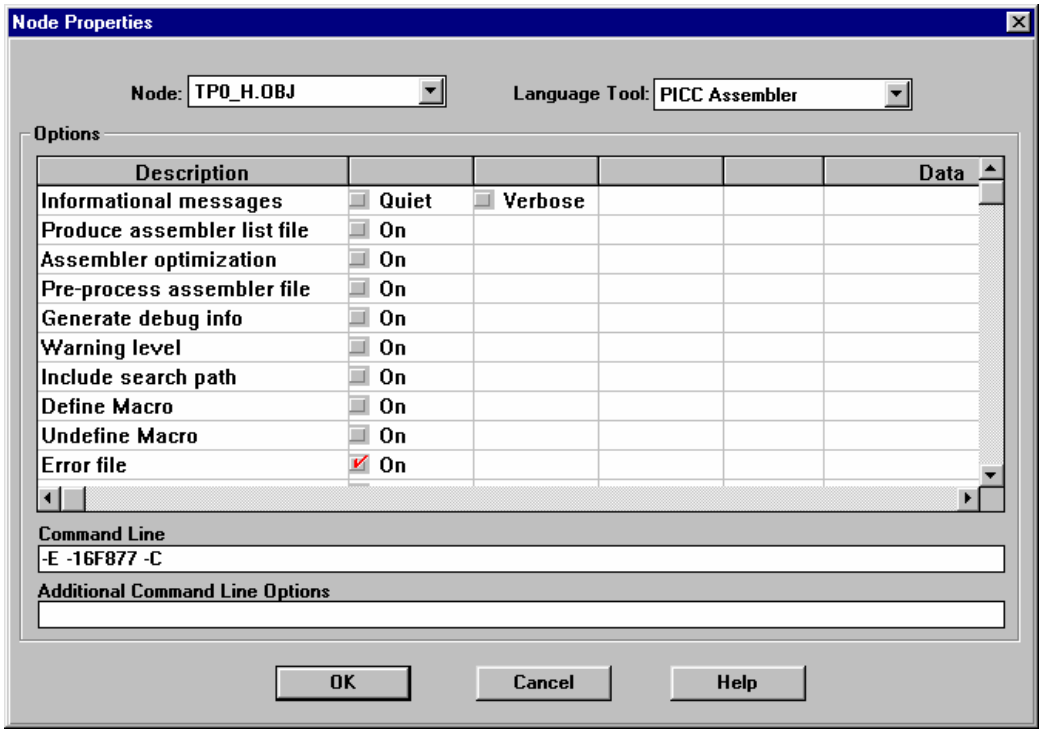
Les autres options utiles sont :

- Produce Assembler List File : pour examen du code généré en langage d'assemblage. Si rien n'est indiqué dans la colonne Data, le fichier créé a même nom que le fichier source avec l'extension lst.
- Global Optimization : pour réduire la taille et le temps d'exécution du code créé, lors de la compilation. La colonne Data contient une valeur de 1 à 9 (optimisation max)
- Assembler Optimization : pour réduire la taille du code créé après la compilation
- Error File

6.2) ASSEMBLEUR SEUL

L 'assembleur seul est utilisé pour les programmes source en langage d'assemblage (extension as).

Le paramétrage s'effectue avec la boîte de dialogue suivante (MPLAB 5.5) :



Pour plus d'information, voir les manuels de Microchip et HiTech.

ANNEXE

VUE D'ENSEMBLE DE LA COMPILATION

cpp : pré processeur fichiers .h fournis : définition des registres à fonction spéciale

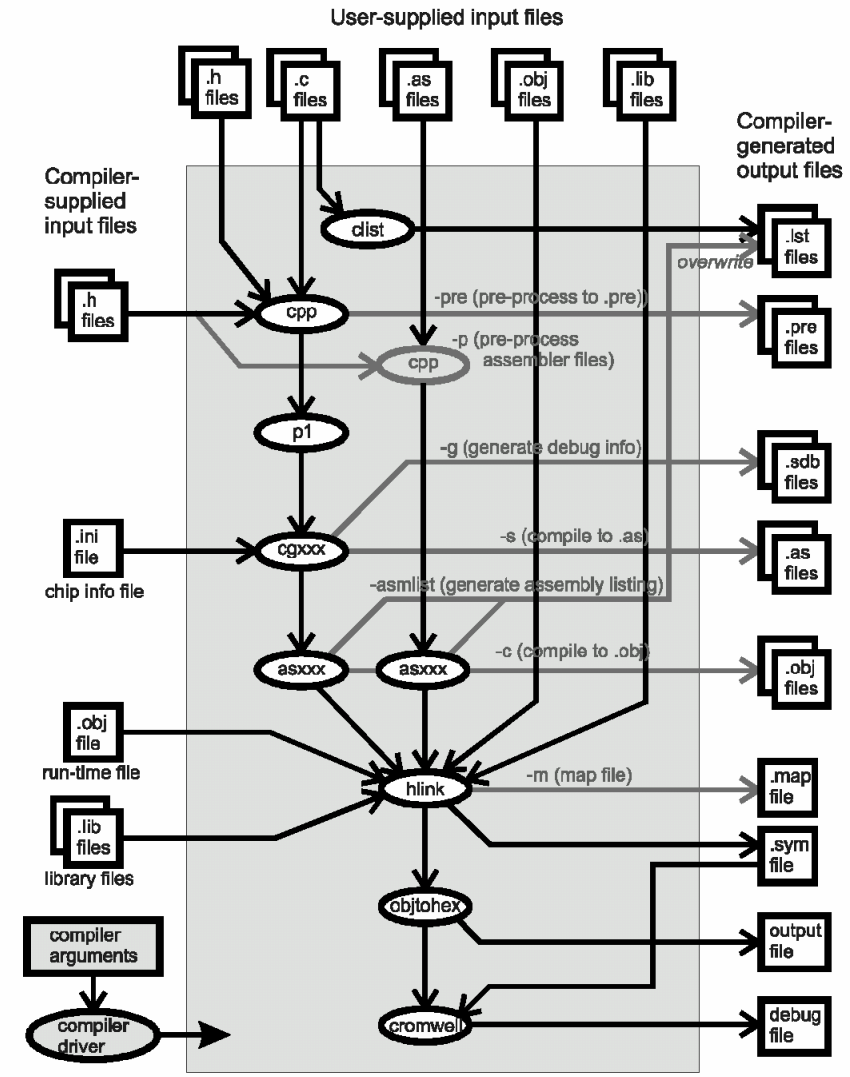
p1 : analyseur syntaxique

cgpic : générateur de code

aspic : assembleur

hlink : éditeur de liens

fichier .obj : programme de démarrage (startup ou run-time)



EDITION DE LIENS

PROGRAMME DE DÉMARRAGE

Pour pouvoir s'adapter à tous les PIC, et donc aux particularités de certains d'entre eux, et à la présence ou non de données globales (initialisées et/ou non initialisées) dans le programme utilisateur, le programme de démarrage se décompose en 3 :

- un programme **powerup** très court qui tient en quelques lignes et qui renvoie par un saut au programme **start.** Ce programme commence à l'adresse 0 et se termine à l'adresse 3, avant l'adresse 4 utilisée par les interruptions
- un programme **start** qui renvoie par un saut au programme main. Les programmes décrits ci-dessous peuvent être automatiquement insérés dans start par l'éditeur de liens si nécessaire.
- un programme qui permet d'initialiser à 0 toutes les variables globales non initialisées et de recopier de la ROM vers la RAM les valeurs des variables globales initialisées.

Le programme de démarrage déjà compilé est placé dans un module objet qui est lié lors de l'édition de liens. Il existe plusieurs modules pour s'adapter aux différents PIC. L'éditeur de liens choisit le module selon le PIC passé en argument sur la ligne de commande (voir ci-dessous Options et configuration de l'éditeur de liens).

Tous les programmes figurant ci-dessous sont dans le dossier Dossier_d'installation\Sources.

Programme source **powerup.as**

#include "sfr.h" global powerup,start psect powerup,class=CODE,delta=2 powerup #if defined(_PIC14) clrf STATUS movlw start>>8 movwf PCLATH goto start & 7FFh #endif #if defined(_PIC16) movlw start>>8 movwf PCLATH movlw start & 0xFF movwf PCL #endif end powerup	PCLATH ← 8 bits de poids fort de start PCL ← 8 bits de poids faible de start. Provoque PCH ← PCLATH → saut à l'adresse de start

Extrait du programme source **picrt66x.as**

psect init start #if (defined(_12C508) || defined(_12C509) || defined(_16C505)) movwf 5 ;calibrate oscillator #endif _exit ; Other modules get linked in here to clear bss and copy ; data if necessary psect end_init #if defined(_PIC16) setf STATUS	L'éditeur de liens insère ici automatiquement les autres programmes si nécessaire

```
#if     !defined _17C42
        movlr   0
#endif
        movlw _main >> 8
        movwf PCLATH
        movlw _main & 0xFF;
        movwf PCL
#else
#ifndef _PIC12
        clrf    STATUS
#endif
        ljmp    _main           ;go do the main stuff
#endif
```

Programme source **clr.as** (effectue une RàZ des variables de classe de mémorisation statique)

```
#ifdef  _PIC12
        psect   clrtext,class=ENTRY,delta=2
#else
        psect   clrtext,class=CODE,delta=2
#endif

#include        "sfr.h"

;       Clear the rbss_x psects

        global  clear_ram

;       Called with FSR containing the base address, and
;       W with the last address+1

#ifdef  _PIC16
clear_ram
        bsf     STATUS,4        ;set up auto-increment
        bcf     STATUS,5
        goto    loopent
clrloop
        clrf    INDF,f          ;clear memory, auto-increment
loopent
        cpfseq  FSR
        goto    clrloop
        return
#else

clrloop
        xorwf   FSR,w           ;restore correct value in W
        clrf    INDF            ;clear a byte
        incf    FSR,f           ;increment pointer
clear_ram
        xorwf   FSR,w
#ifdef  _PIC12
        andlw   1Fh             ;Test low bits only
```

#endif btfss STATUS,2 ;test zero goto clrloop retlw 0 #endif /* _PIC16 */	

Le programme source pour la copie des valeurs des données initialisées de la ROM vers la RAM est copy.as

OPTIONS DE L'ÉDITION DE LIENS

Options décrites dans la boîte de dialogue	traduction sur la ligne de commande	
Informational messages	-q ou -V	Verbose : affiche le détail des commandes utilisées pour l'édition de liens
Strip Local Symbols	-X	
Generate Debug Info	-G	tous les fichiers source doivent aussi être compilés avec cette option **Indispensable pour débogage avec MPLAB**
Hex Format	-INTEL ou -MOTOROLA	**soit aucun option cochée, soit INTEL**
Generate binary output	-BIN	inutile
Error file	-E	crée un fichier d'erreur pour l'édition de liens. Inutile
Append Errors to file	-E+	ajoute les erreurs de l'édition de liens dans un fichier d'erreurs unique pour toutes les opérations de la compilation.
Map file	-M	crée un fichier de la cartographie mémoire utilisée. Le nom du fichier doit être précisé dans la colonne data (*la cartographie est cependant toujours affichée à l'écran après l'édition de liens*)
Display Complete Memory Usage	-PSECTMAP	affiche tous les segments de la mémoires d'après les sections du programme.
Compile for MPLAB ICD	-ICD	