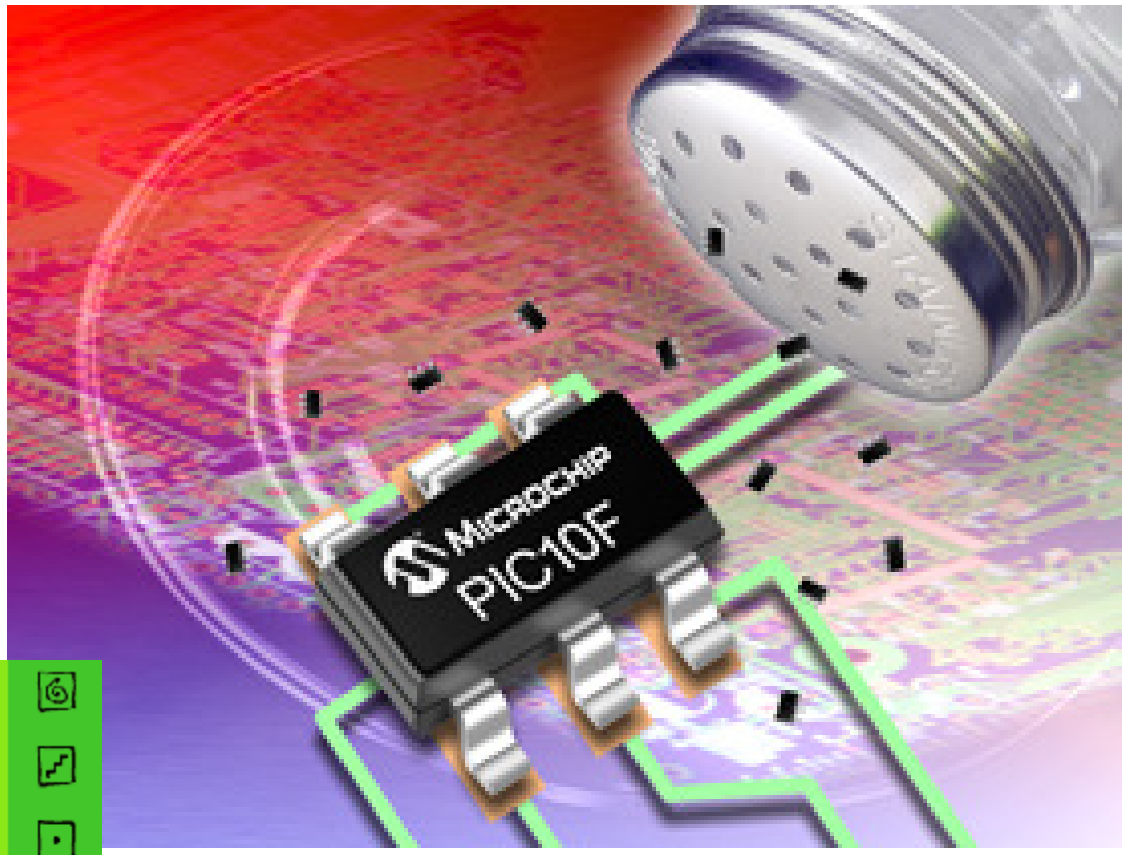
IUT BELFORT MONTBÉLIARD

Mesures Physiques



UNIVERSITÉ DE FRANCHE-COMTÉ

MICROCONTROLEURS PIC
PROGRAMMATION EN C

Chapitre 1 – LE COMPILATEUR

1 – INTRODUCTION

1.1 – MICROCONTROLEUR PIC

Un microcontrôleur est un microprocesseur RISC (Reduced Instruction Set Computeur) comportant un nombre d'instructions réduit et disposant de ports d'entrée/sortie pour communiquer avec l'environnement extérieur, de registres internes, de mémoire et d'une horloge interne ou externe.

Les microcontrôleurs PIC sont des microcontrôleurs fabriqués par la société Microchip qui fournit par ailleurs gratuitement la plate-forme logiciel de développement MPLAB IDE.

L'intérêt est, pour un faible coût, de disposer d'un composant programmable de nombreuses fois, pouvant être utilisé de façon autonome : plus besoin d'ordinateur une fois le composant programmé.

L'utilisation d'un microcontrôleur dans une application simplifie notablement les montages électroniques entraînant par la même occasion un gain de temps et de coût.

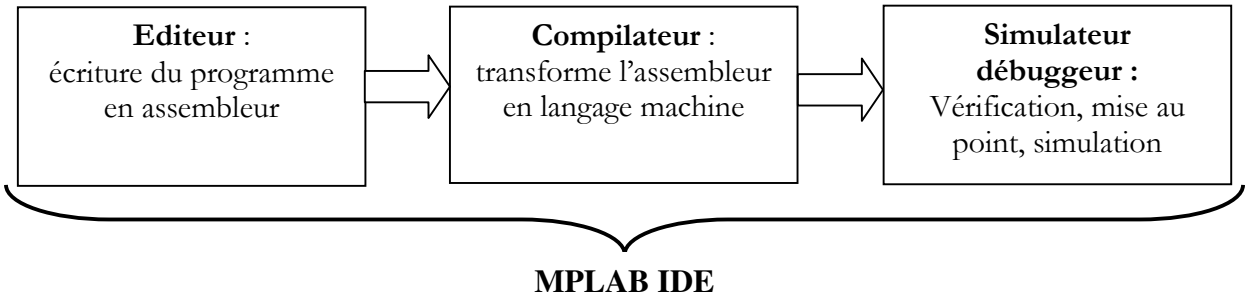
Les domaines d'utilisation principaux sont la robotique, la domotique, l'industrie.

1.2 – LES OUTILS POUR REALISER UNE APPLICATION

Pour développer une application fonctionnant à l'aide d'un microcontrôleur, il faut disposer d'un compilateur et d'un programmateur.

Compilateur : Fabrique le programme	**Programmateur** : Transfert le programme du PC au PIC (hard+soft)

Le compilateur est un logiciel traduisant un programme écrit dans un langage donné (C, basic, assembleur) en langage machine. Ce logiciel peut aussi comporter un « debugger » permettant la mise au point du programme, et un simulateur permettant de vérifier son fonctionnement.



Le fabricant Microchip fournit gratuitement le logiciel MPLAB IDE téléchargeable sur le site www. microchip.com

Le programmateur permet de transférer le programme compilé (langage machine) dans la

mémoire du microcontrôleur. Il est constitué d'un circuit branché sur le port COM du PC, sur lequel on implante le PIC, et d'un logiciel permettant d'assurer le transfert. Il existe différents logiciels, nous utiliserons Icprog.

1.3 – LANGAGE DE PROGRAMMATION UTILISE

Dans l'environnement MPLAB, Le programme doit être écrit en assembleur, langage peu évolué, peu convivial, et donc peu accessible aux étudiants bac+2.

On préfère donc un langage de programmation évolué : basic ou c. Notre choix se porte sur le langage c étudié par ailleurs en cours d'informatique d'instrumentation.

Le code source écrit en langage c doit donc être compilé en assembleur à l'aide d'un compilateur c.

On utilisera le compilateur CC5X dans sa version gratuite téléchargeable sur www.bknd.com. Cette version gratuite permet d'écrire environ 1ko de programme.

On peut alors intégrer CC5X dans l'environnement MPLAB. Ainsi CC5X devient un outil de MPLAB dans lequel l'écriture, la simulation et le debugging du programme en c devient alors possible.

2 – COMPILATEUR CC5X

2.1 – INSTALLATION

Cette installation a déjà été réalisée. Les indications suivantes vous sont fournies pour l'installation sur votre ordinateur personnel.

Créer un répertoire CC5X où vous le souhaitez sur le disque dur de votre PC.
Télécharger CC5X free sur le site www.bknd.com
Décompresser ce fichier.

Le répertoire CC5X contiendra le fichier exécutable cc5x.exe et les fichiers de définition (header .h) des microcontrôleurs utilisables avec CC5X.

2.2 – CARACTERISTIQUES

La version gratuite est limitée à 1 ko de programme.

Les divers types de variables sont codés de la façon suivante :

Type char : forcément non signés sur 8 bits
Type signed char : 8 bits signés.
Type int : 8 bits signés
Type unsigned int : 8 bits non signés
Type long : 16 bits signés
Type unsigned long : 16 bits non signés
Type bit : 1 bit
Type float : nombre à virgule flottante codé sur 24 bits.

La version commerciale utilise des types entiers sur 24 et 32 bits et des nombres à virgule fixe.

3 – MPLAB IDE v7.31

3.1 – INSTALLATION

Cette installation a déjà été réalisée. Les indications suivantes vous sont fournies pour l'installation sur votre ordinateur personnel.

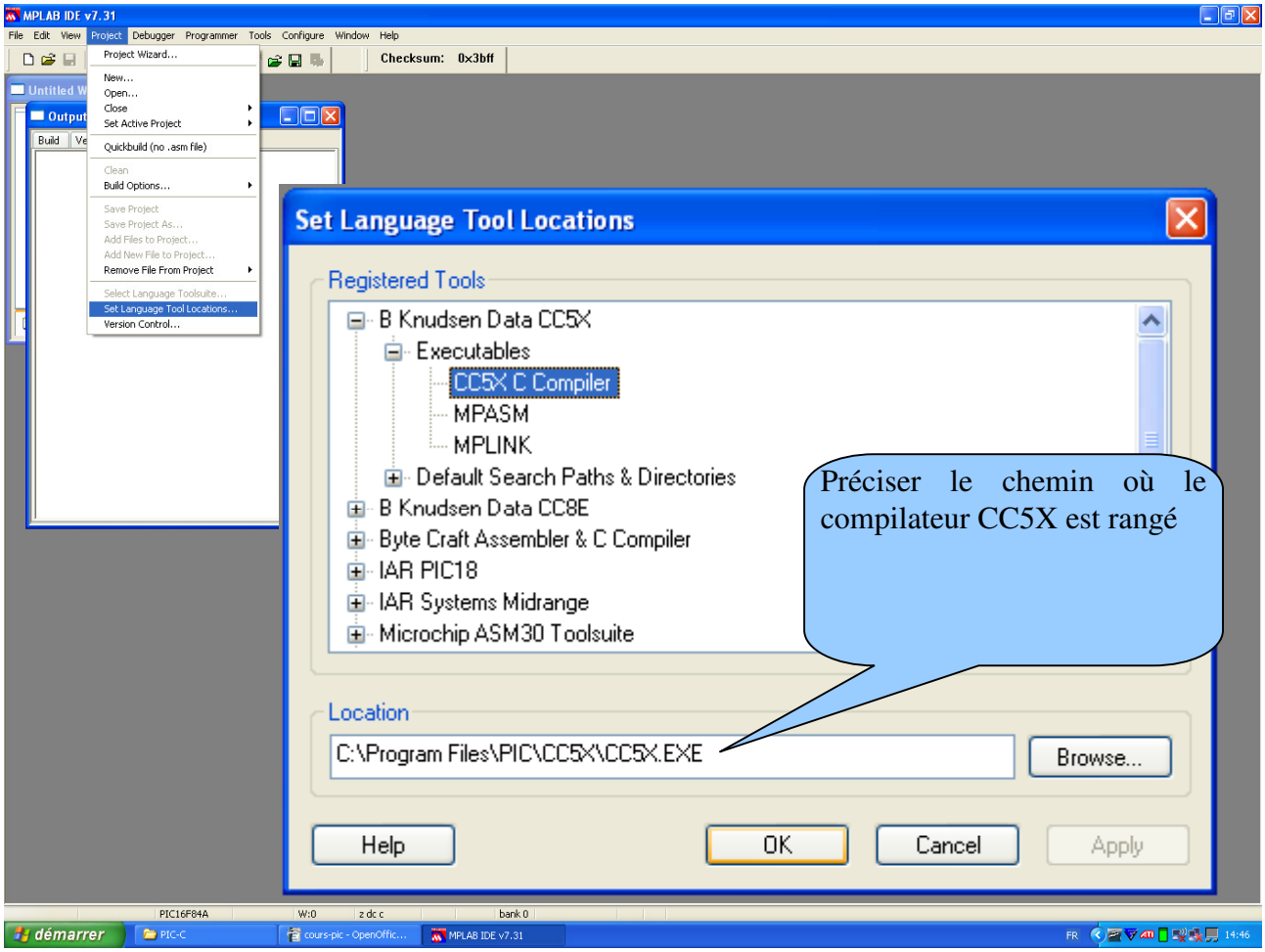
Créer un répertoire MPLAB sur le disque dur de votre ordinateur.
Télécharger MPLAB sur le site www.microchip.com
Décompresser le fichier.

Suivre les indications lors de l'installation.

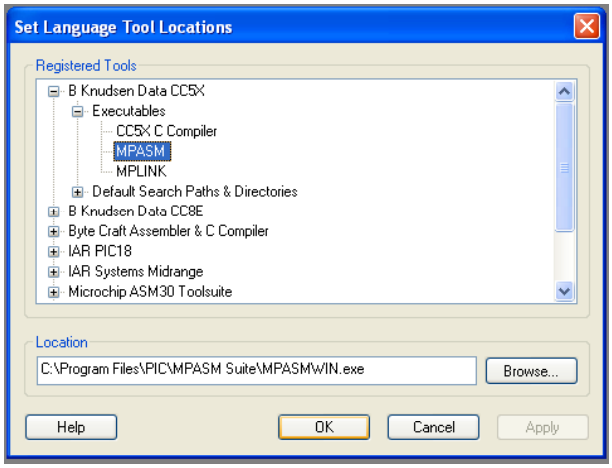
Pour pouvoir utiliser le debugger, il faut ensuite corriger le fichier TLCC5X.INI situé dans le répertoire MPLAB IDE\Core\MTCSuites : Il faut remplacer « Target=HEX » par « Target=COD » et sauvegarder la modification.

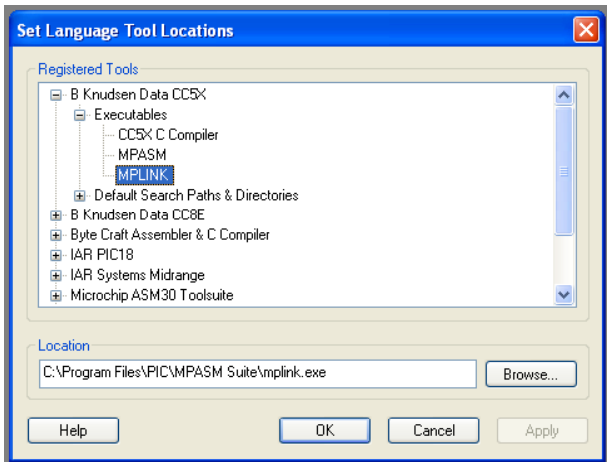
3.2 - CONFIGURATION

Déclaration du compilateur CC5X : Menu Project/Set Langage Tool Locations.
Cette configuration a déjà été réalisée. Les indications suivantes vous sont fournies pour votre ordinateur personnel.



Déclarer également le chemin de MPASM et MPLINK :

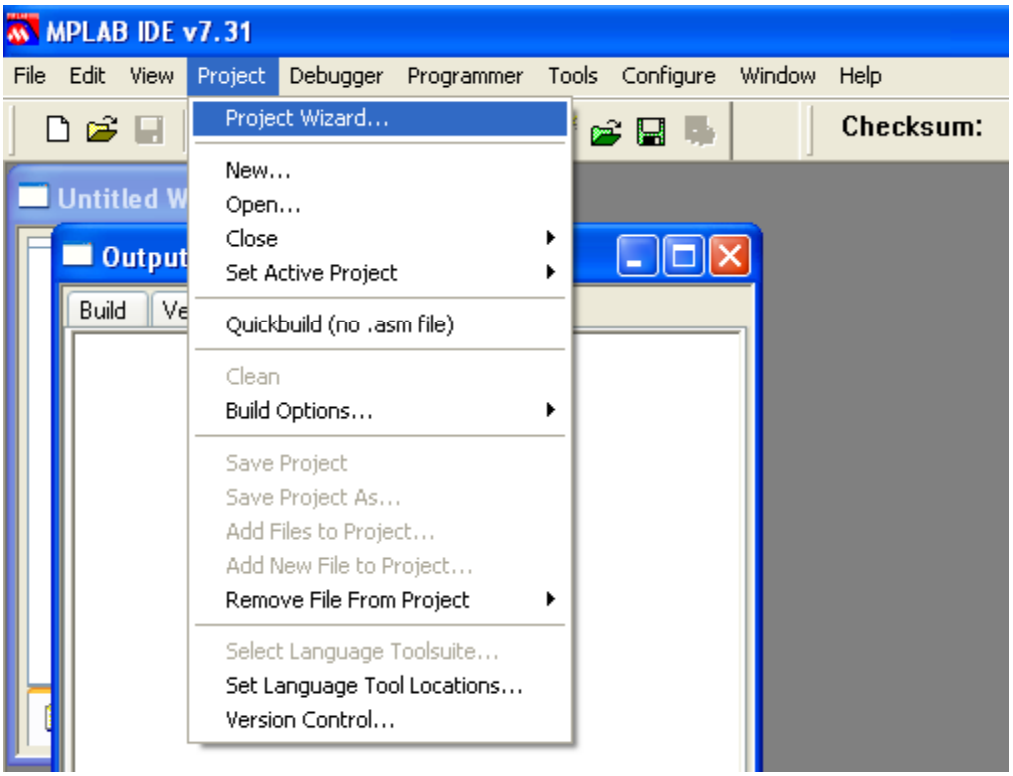




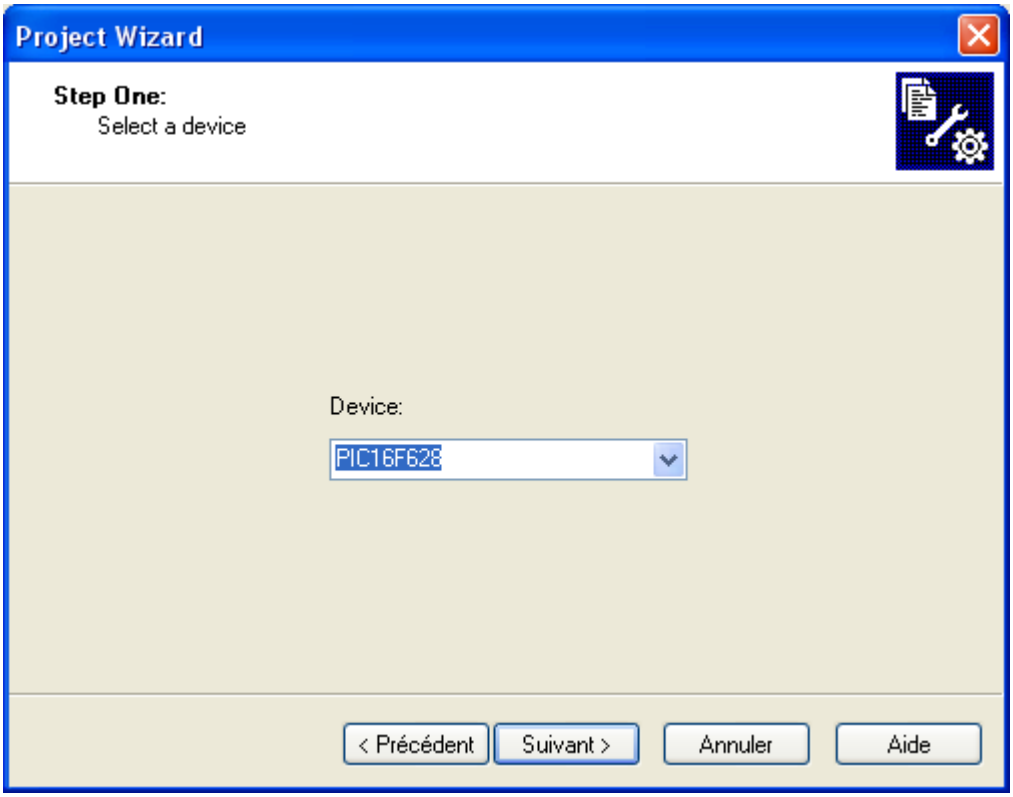
4 – CREATION D'UN NOUVEAU PROJET

4.1 – DEFINITION DU PROJET AVEC L'ASSISTANT

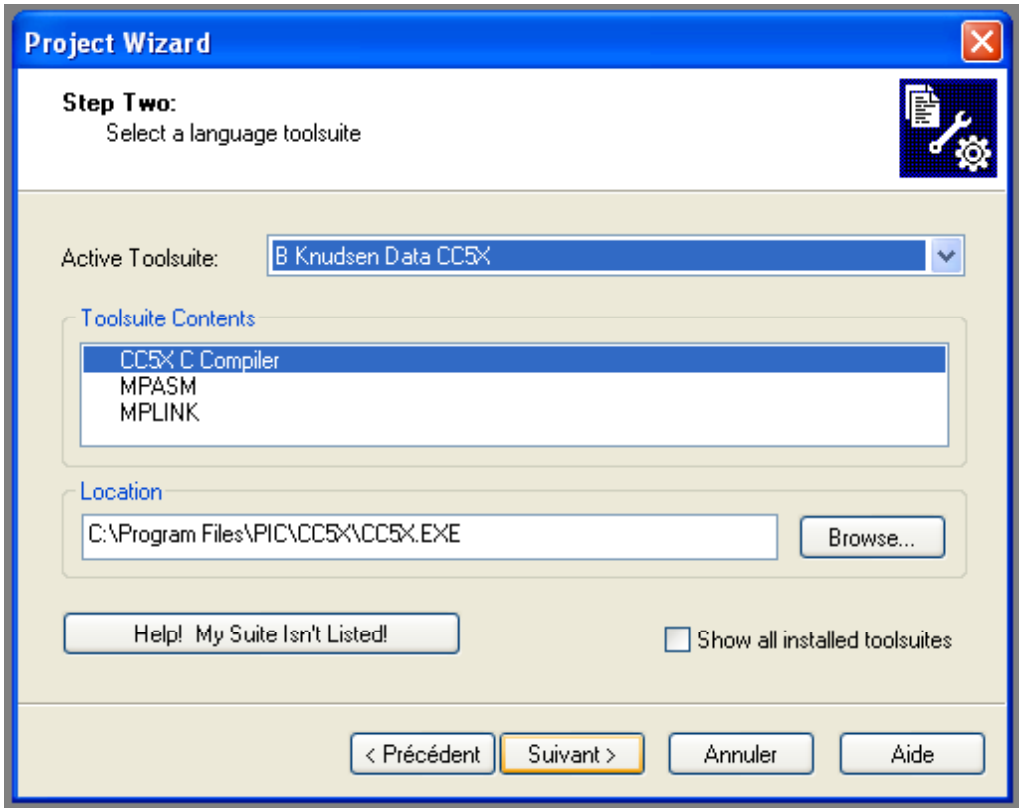
Dans le menu Project, sélectionner Project Wizard. Cela lance un assistant permettant de définir certaines options du projet.



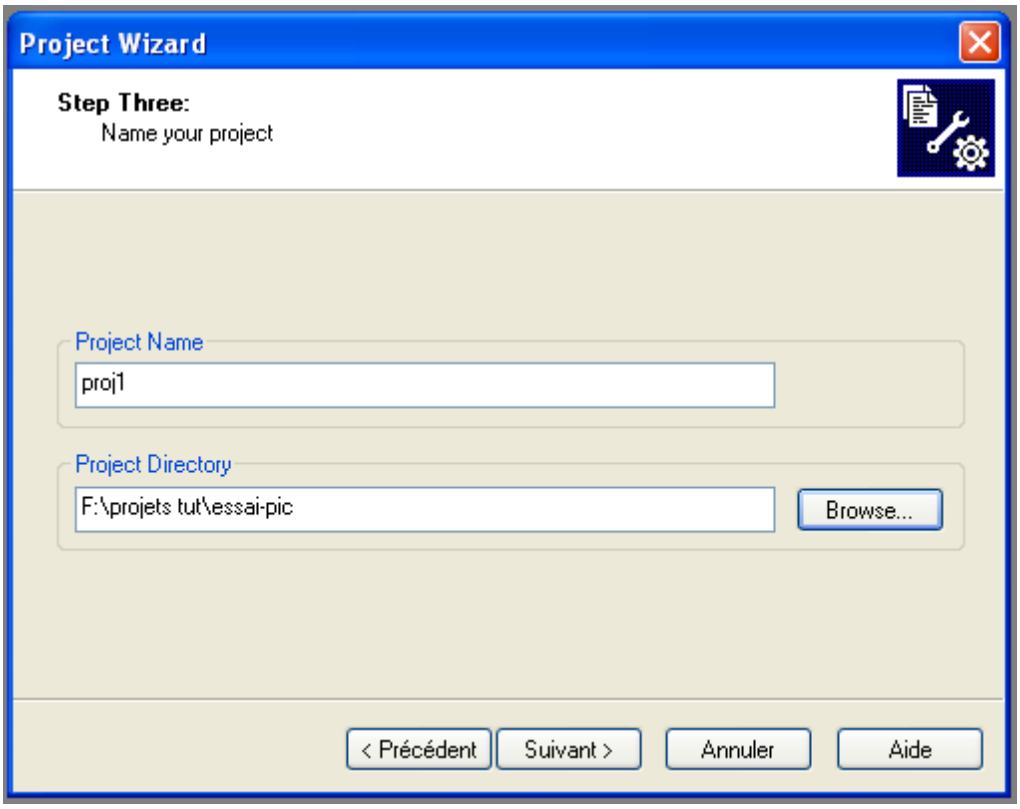
Sélectionner d'abord un microcontrôleur :



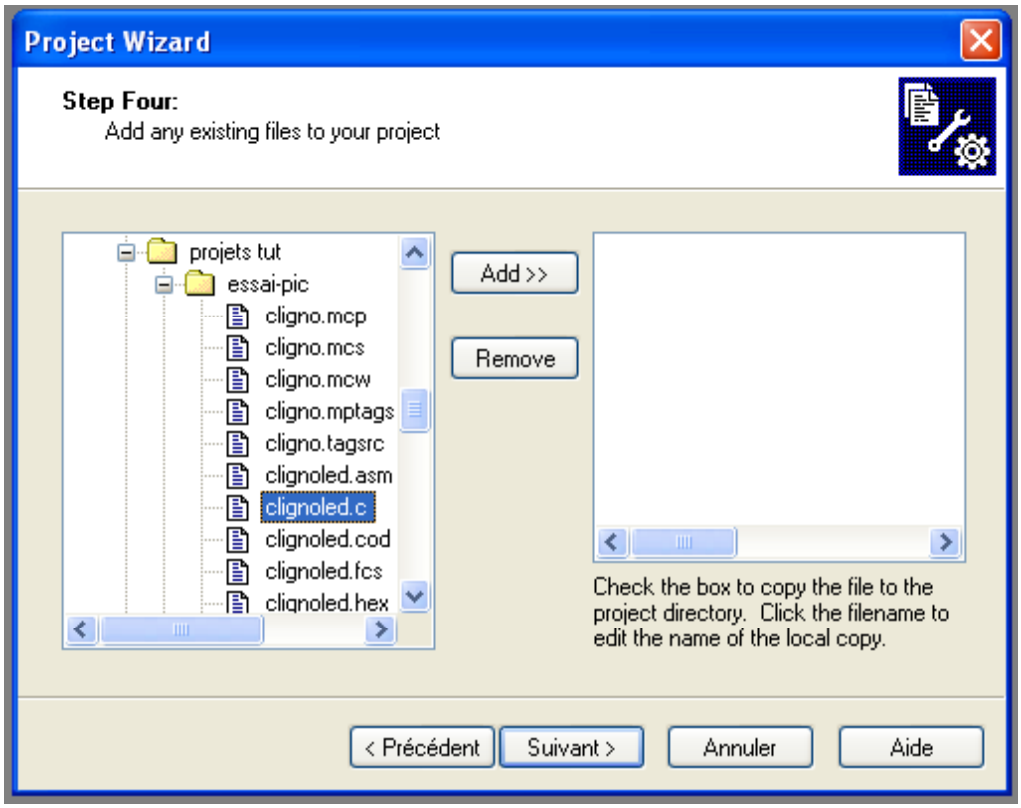
Si la configuration décrite au §3.2 n'a pas été réalisée, il convient de le faire à présent :



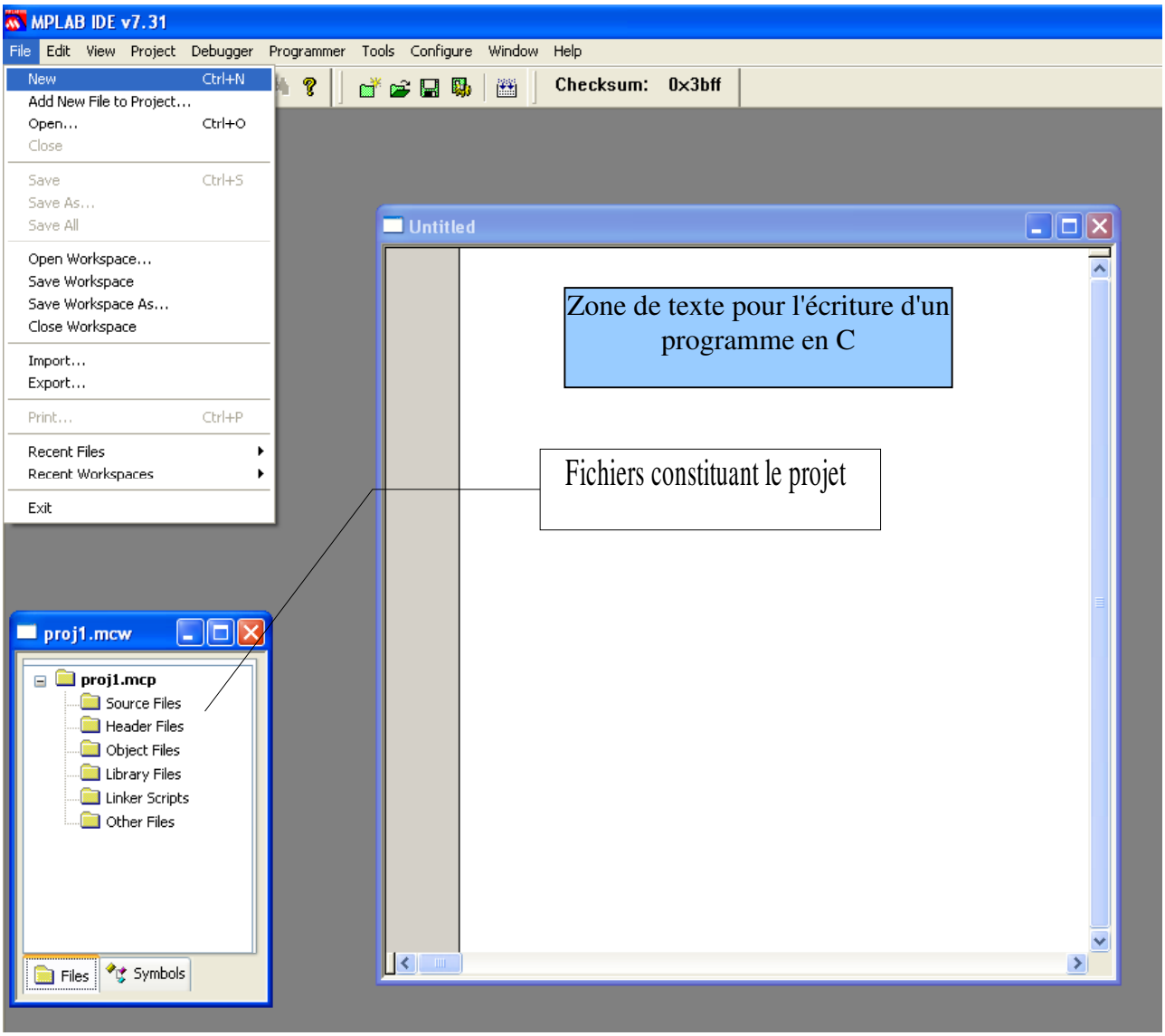
Définir ensuite un nom de projet et un chemin pour la sauvegarde du projet :



La 4ème étape permet d'ajouter éventuellement un fichier déjà créé, par exemple un programme source en c. Si on désire écrire le programme ultérieurement, il faut cliquer sur annuler.

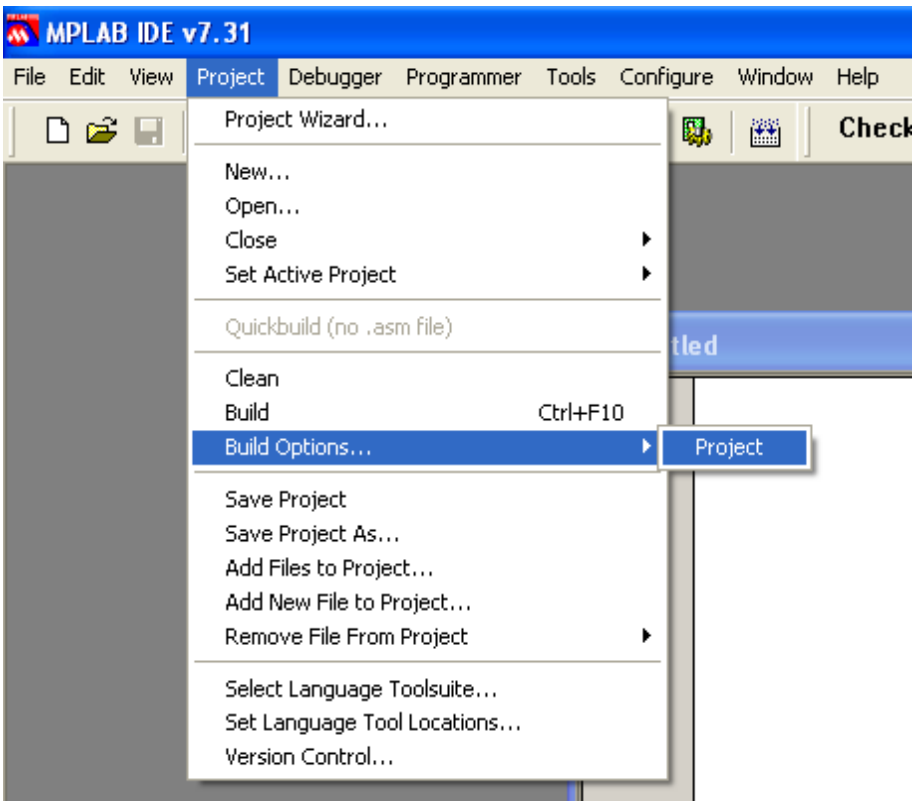


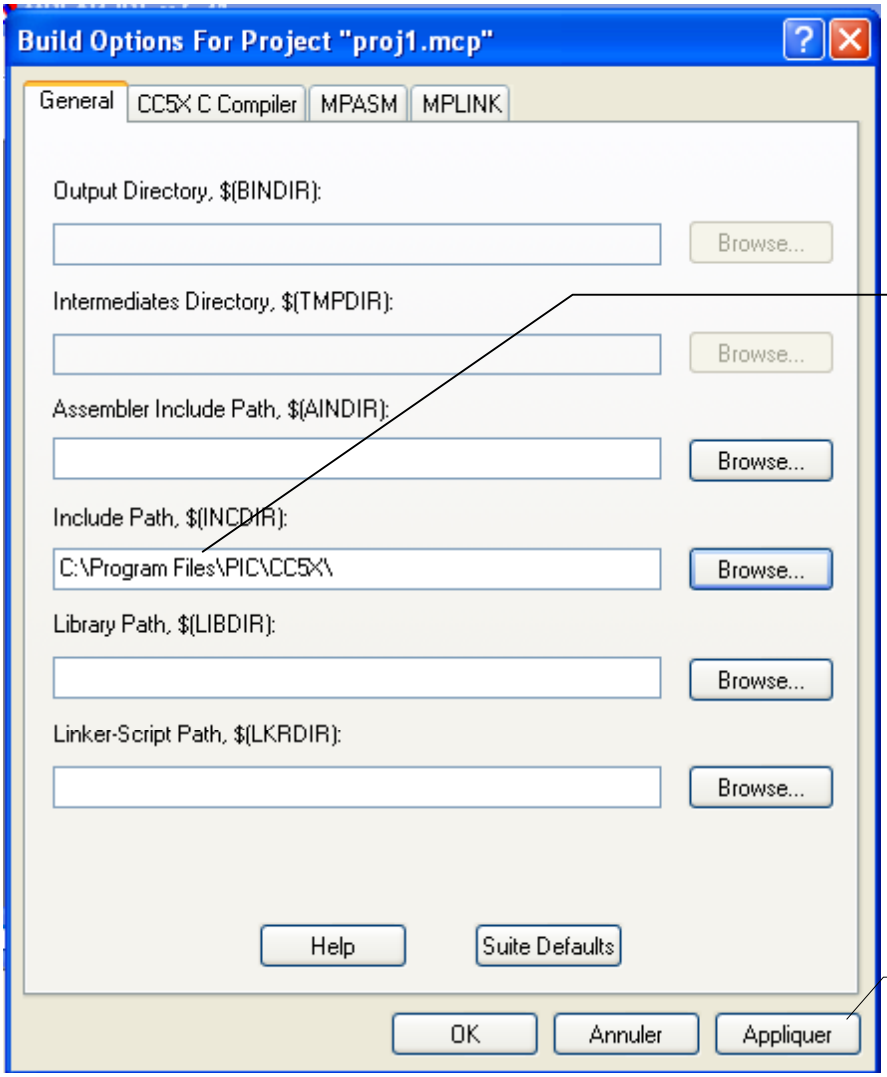
Ayant annulé cette dernière étape, on désire maintenant ouvrir une fenêtre pour l'écriture du programme en langage c. Pour cela, dans le menu fichier, sélectionner new :



4.2 – LES OPTIONS

Pour fonctionner correctement, CC5X a besoin d'accéder aux données spécifiques du PIC sélectionné. Ces données sont définies dans des fichiers de définition (header .h) situés dans le répertoire où CC5X a été installé. Il convient de définir ce chemin dans une fenêtre ouverte par le menu Project/Build Options.



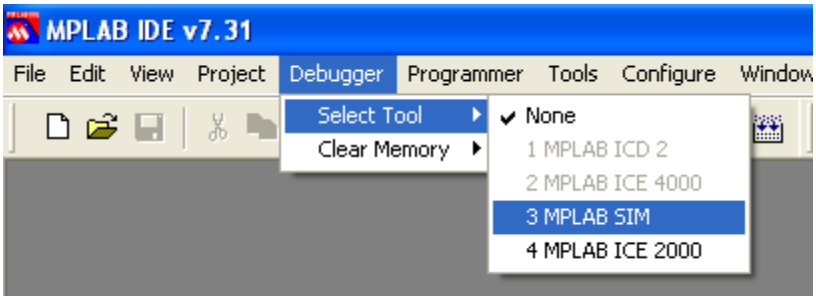


Si nécessaire, remplacer « program Files » par « Progra~1 » car les noms de fichiers trop longs ne sont pas acceptés.

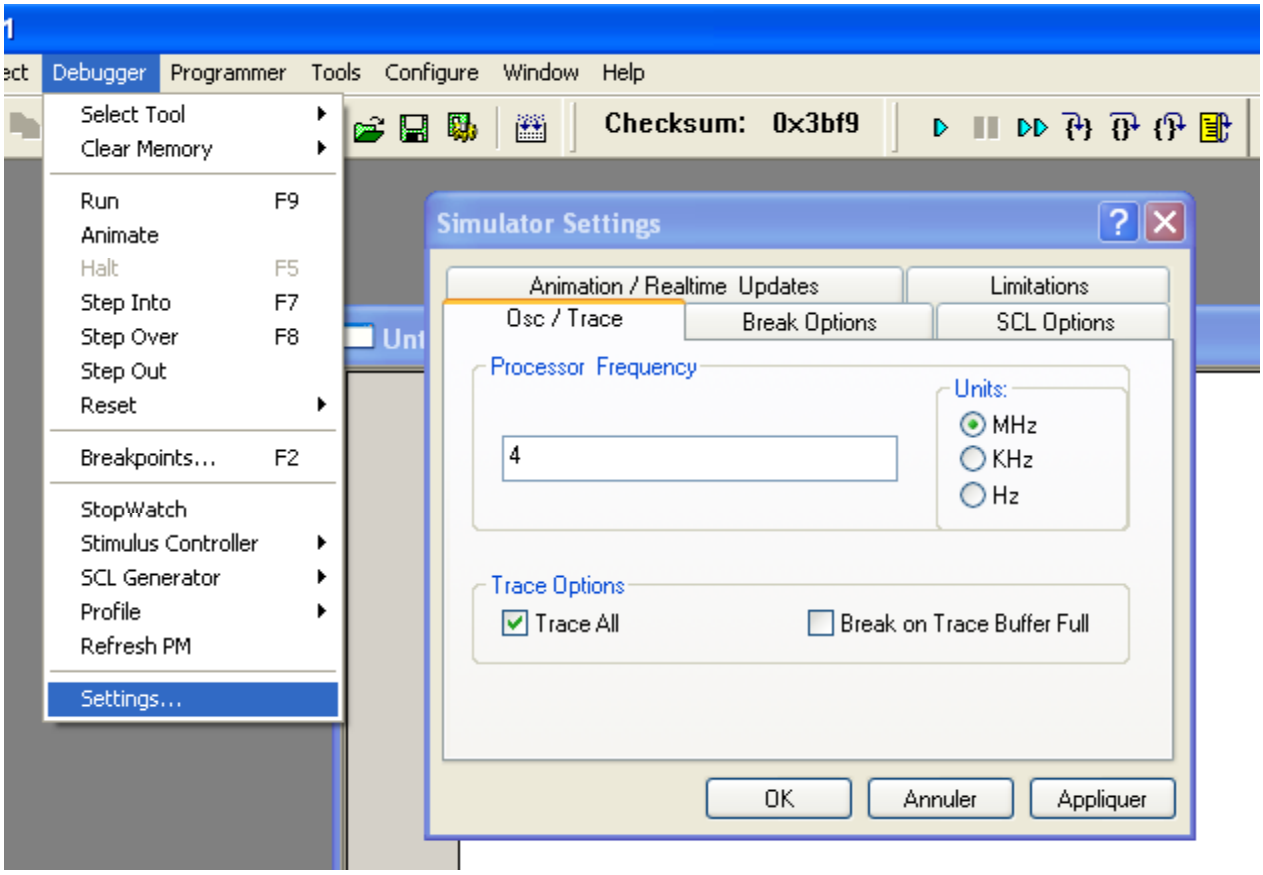
Appliquer la modification

5 – DEBUGGER

Pour pouvoir utiliser le Debugger, il faut sélectionner MPLAB SIM dans le menu Debugger :

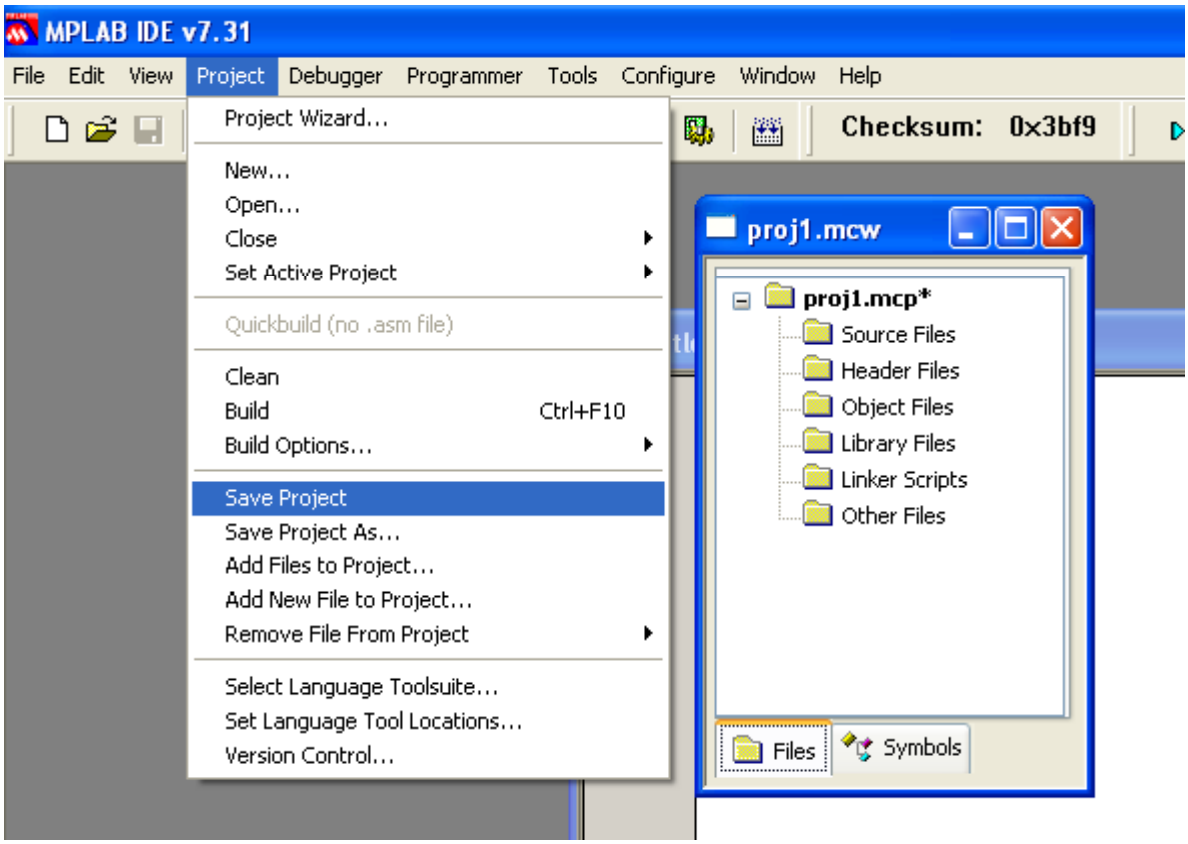


Dans le menu Debugger, de nouvelles sélections apparaissent. Choisir settings pour définir quelques options pour la simulation, en particulier la fréquence de l'horloge dépendant du PIC choisi (4 Mhz pour un 16F84A).



6 – SAUVEGARDE

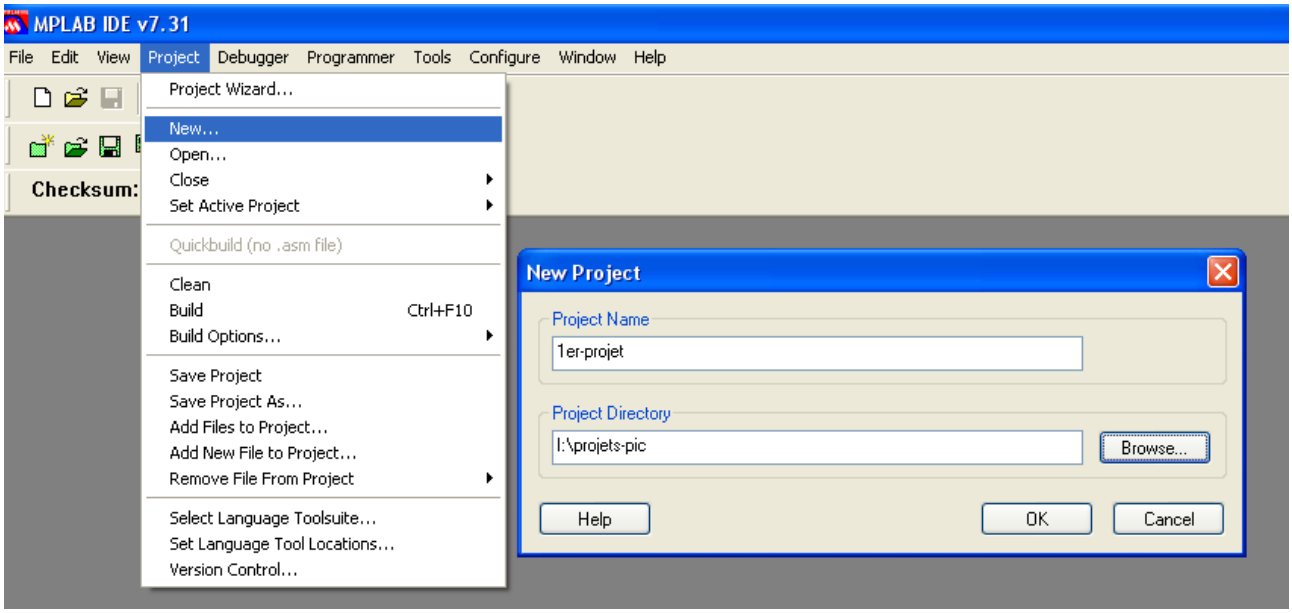
Le projet a été modifié, il convient de le sauvegarder.


MPLAB IDE v7.31
File Edit View Project Debugger Programmer Tools Configure Window Help
Checksum: 0x3bf9
Project Wizard...
New...
Open...
Close
Set Active Project
Quickbuild (no .asm file)
Clean
Build Ctrl+F10
Build Options...
Save Project
Save Project As...
Add Files to Project...
Add New File to Project...
Remove File From Project
Select Language Toolsuite...
Set Language Tool Locations...
Version Control...
proj1.mcw
proj1.mcp*
Source Files
Header Files
Object Files
Library Files
Linker Scripts
Other Files
Files
Symbols

Chapitre 2 – PREMIER PROJET

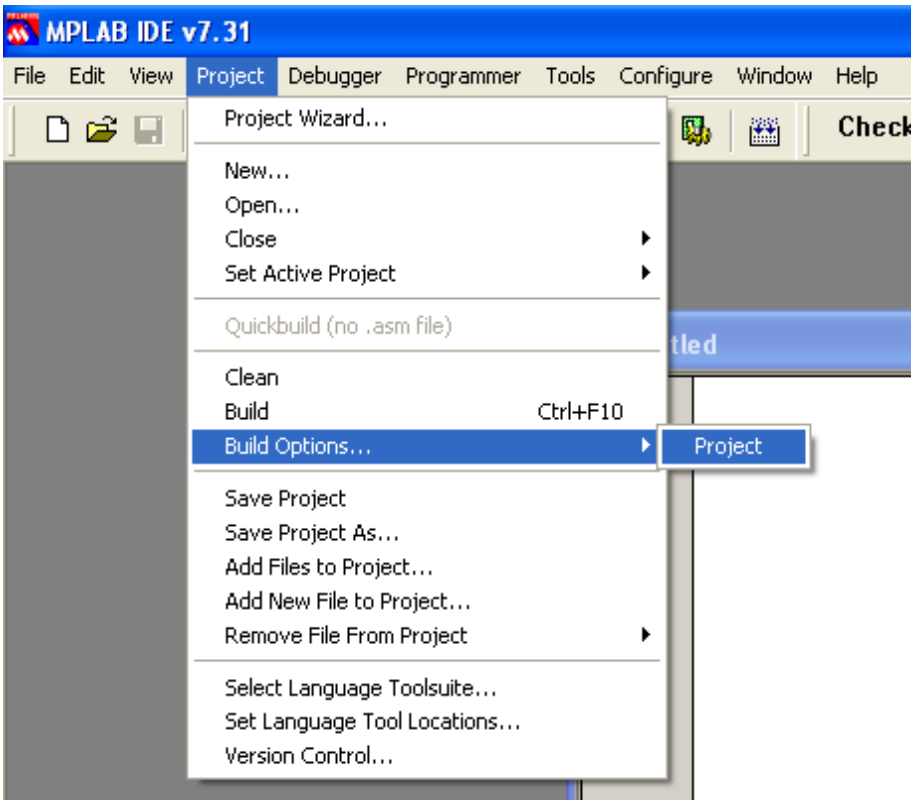
1 – CREATION DU NOUVEAU PROJET

Lancer MPLab. Dans le menu Projet, sélectionner new.

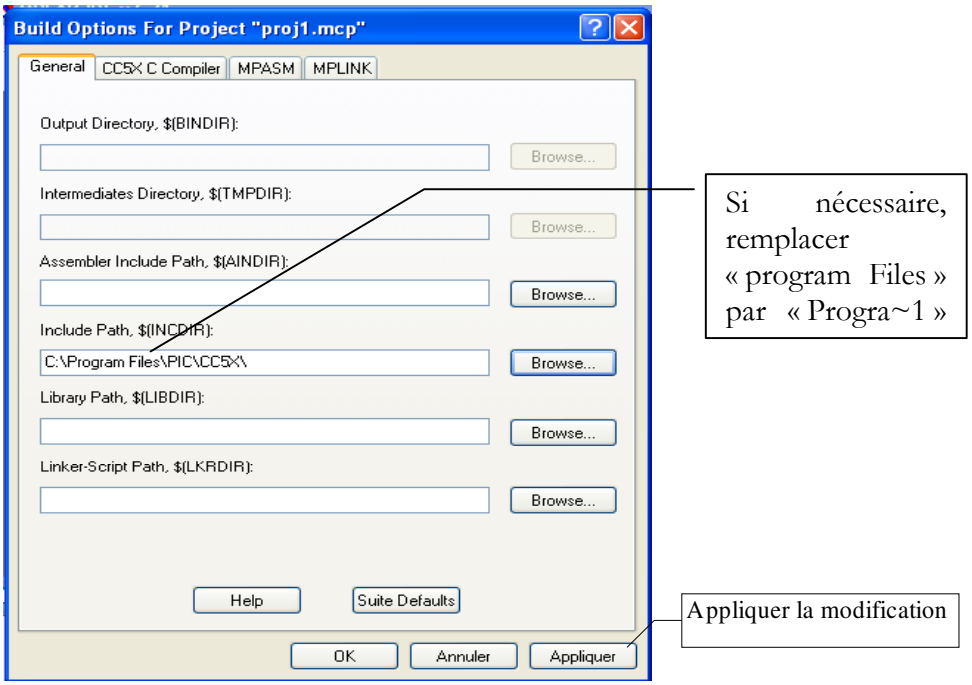


Définir le nom de votre projet et le répertoire pour la sauvegarde.

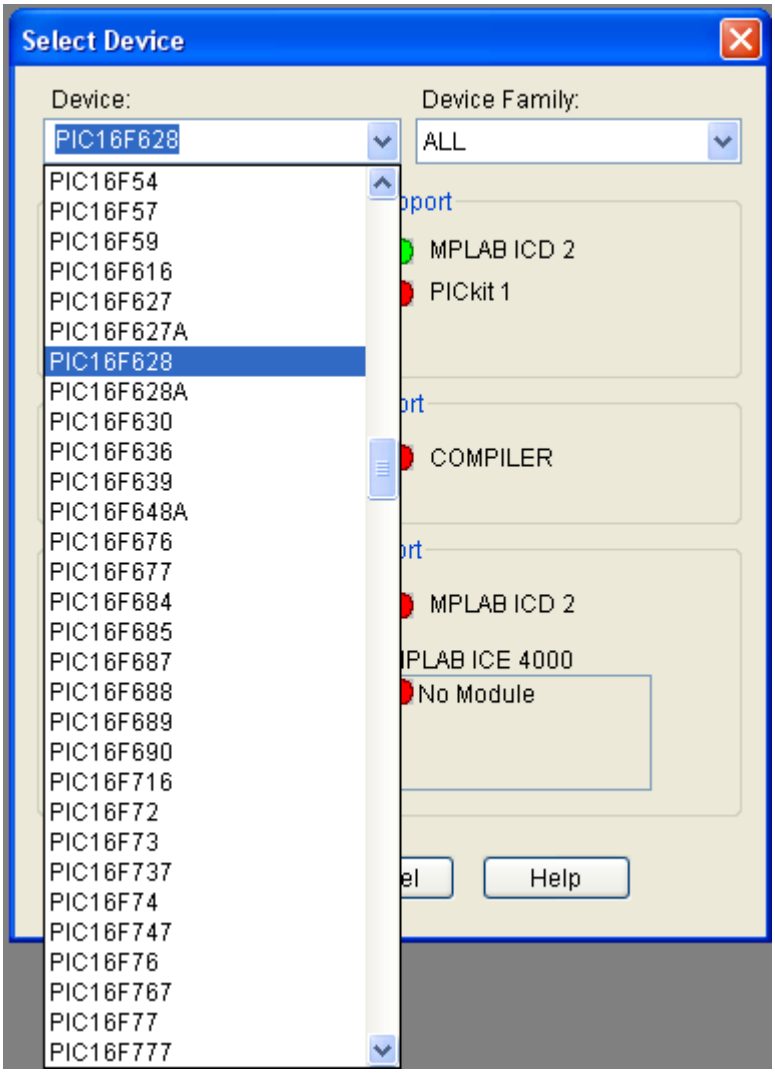
Définir les options :



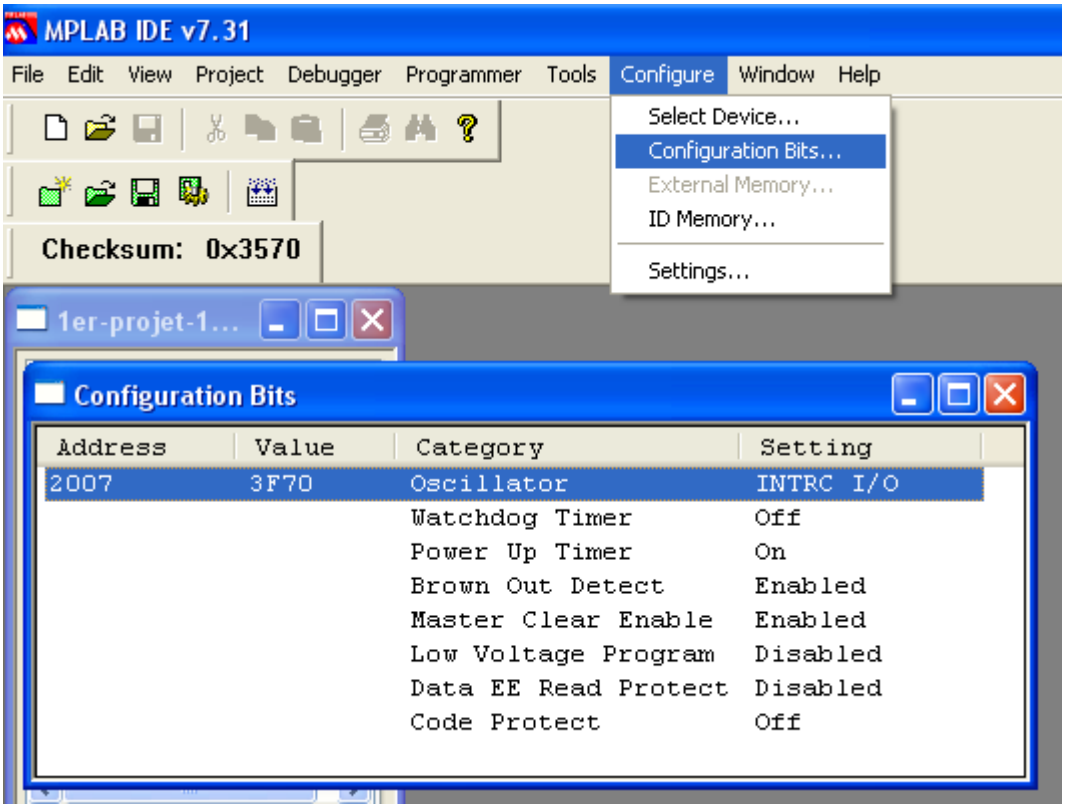
Cela permet de définir le chemin du répertoire d'installation de CC5X.



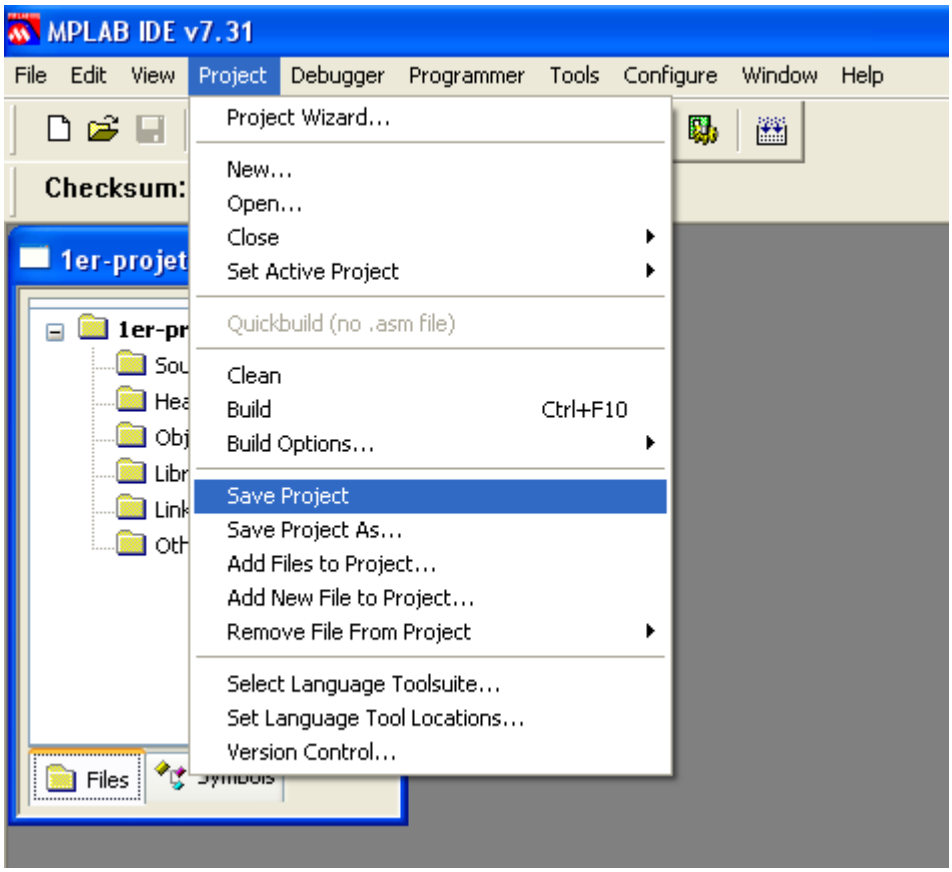
Il convient ensuite de définir le microcontrôleur utilisé : Menu Configure/select device.



Puis il faut définir les options propres au microcontrôleur choisi : Menu configure/configuration bits

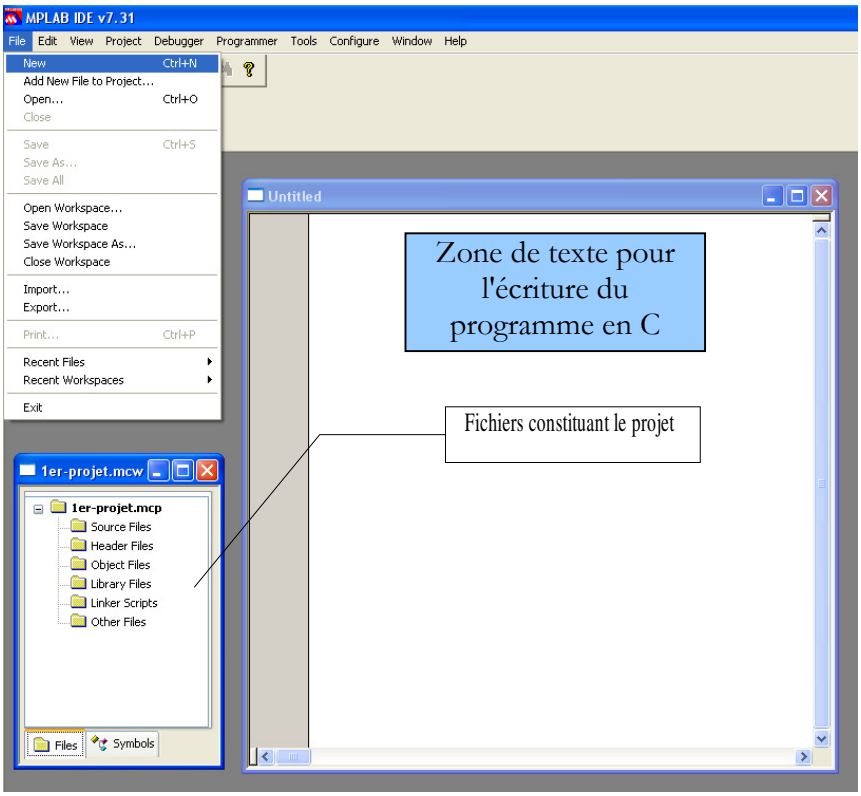


Ne pas oublier de sauvegarder régulièrement les modifications apportées au projet :



2 – ECRITURE DU PROGRAMME EN C

Dans le menu File, sélectionner New. Cela fait apparaître la zone de texte pour l'écriture du programme.

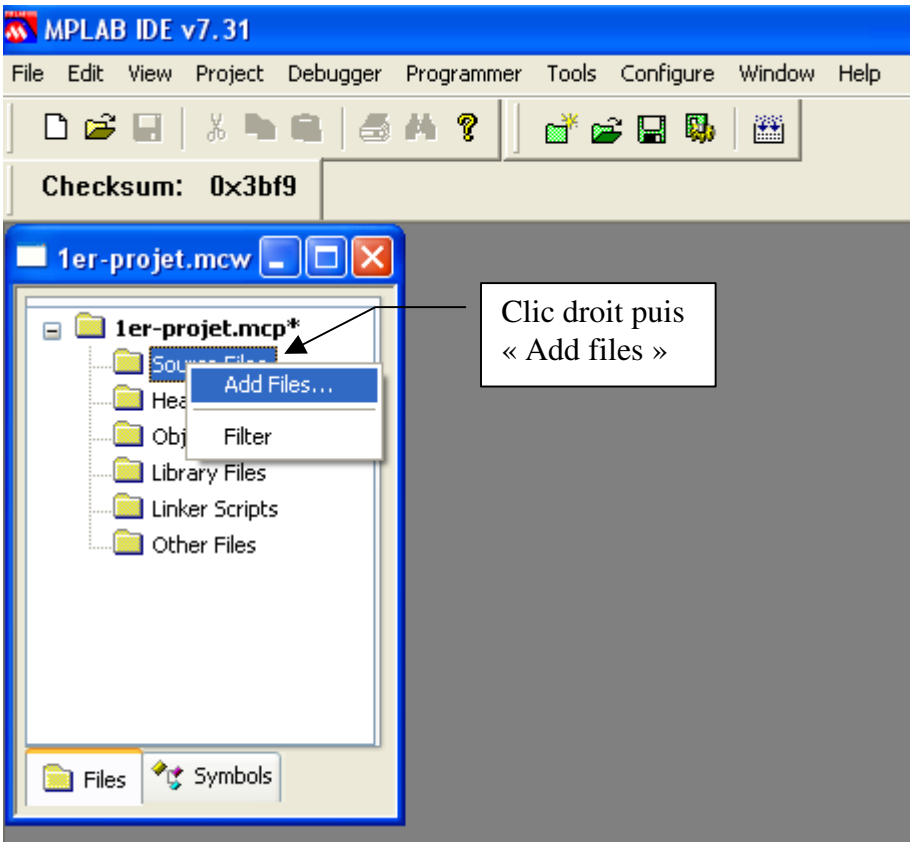


Taper dans la zone de texte, sans pour l'instant chercher à comprendre, le programme suivant :

```
void main(void)
{
        CMCON=7 ;
        TRISA=0 ;
        RA0=0 ;
        RA1=1 ;
        RA2=1 ;
        RA3=0 ;
}
```

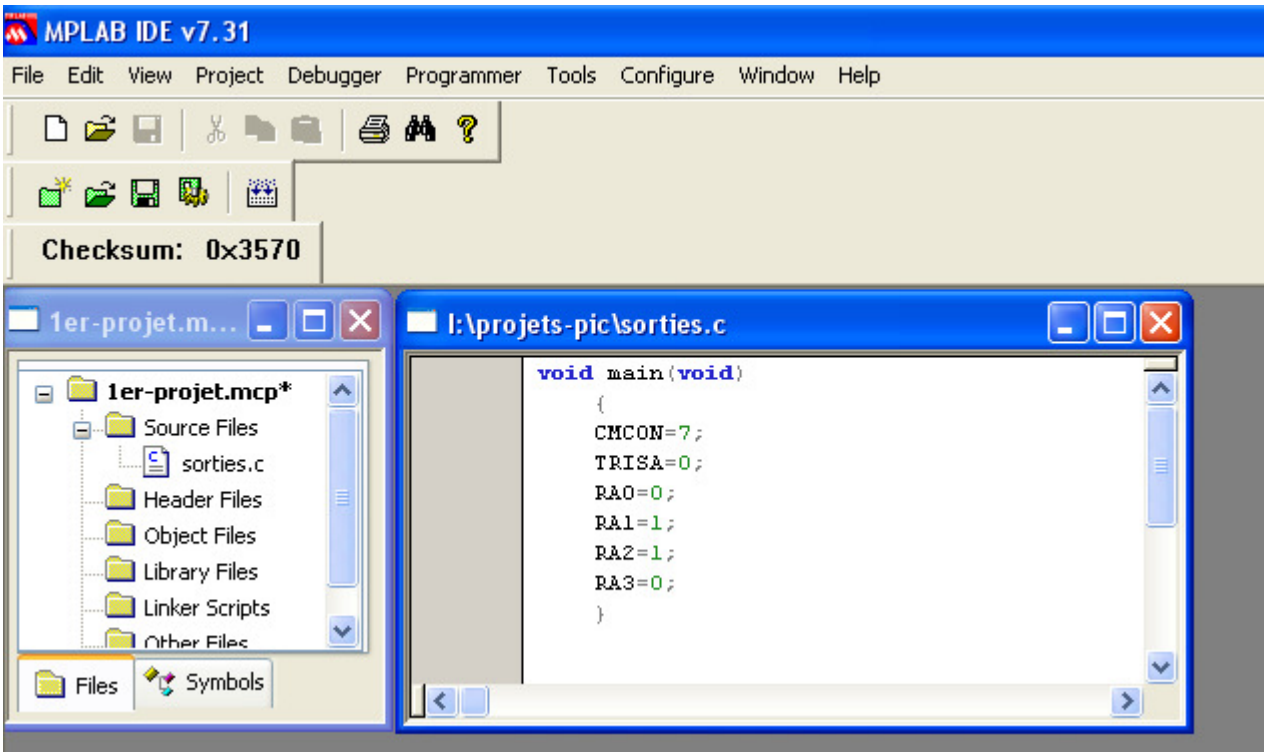
Sauvegarder ensuite le fichier que l'on nommera par exemple sorties.c **dans le même répertoire** que le projet : Menu File/Save as

Le fichier ainsi créé doit alors être ajouté comme fichier source dans le projet :



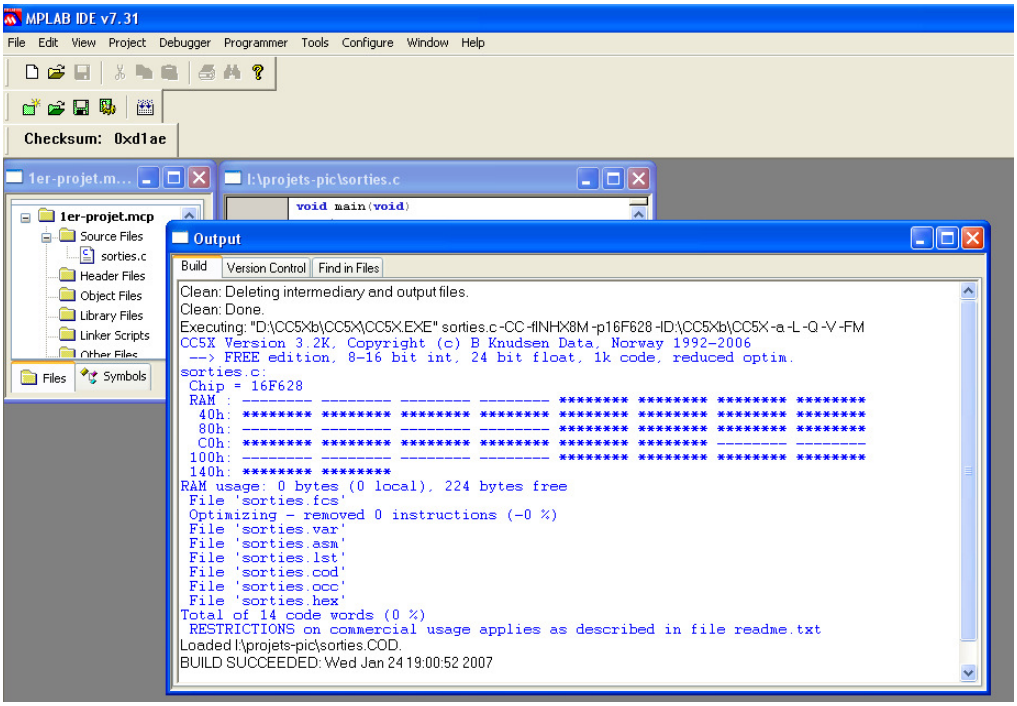
Ouvrir alors le fichier sorties.c que vous venez de créer.
On peut bien sûr ouvrir un autre fichier .c à condition qu'il soit dans le même répertoire.

Quelle que soit la méthode, nous obtenons :

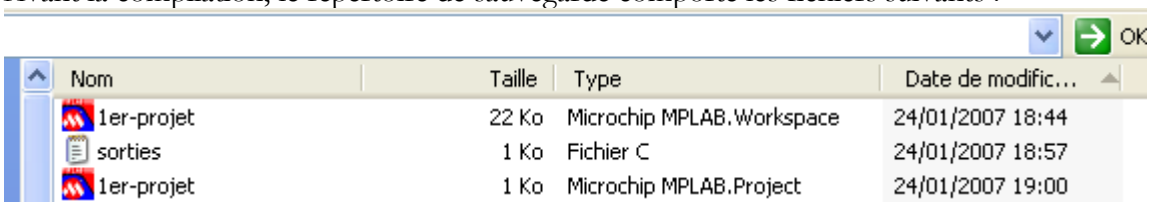


3 – COMPILATION

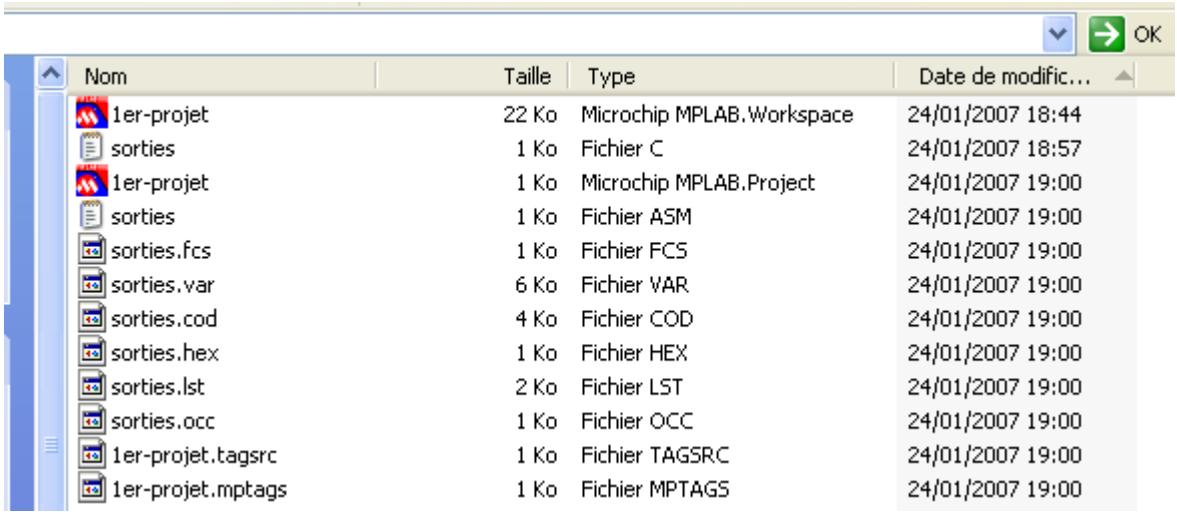
Le projet créé peut maintenant être compilé : Menu Projet/Build



Avant la compilation, le répertoire de sauvegarde comporte les fichiers suivants :



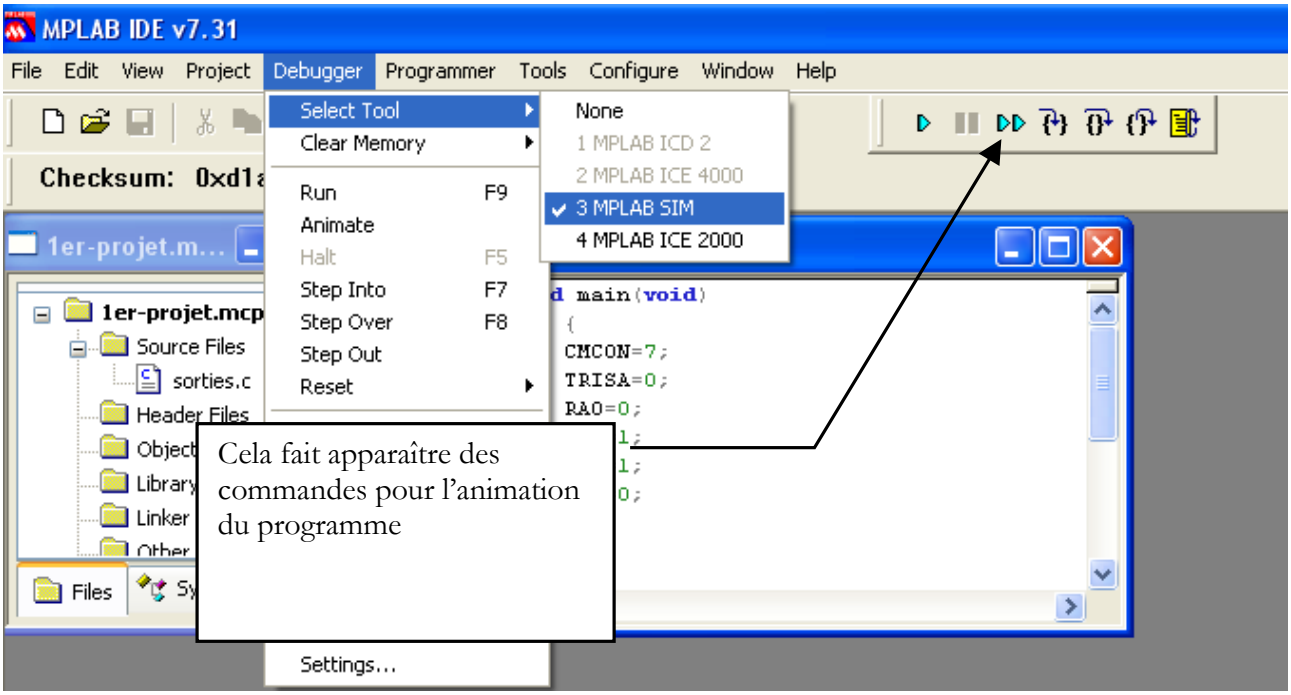
La compilation réalisée à 19h00 ajoute 9 fichiers dans le répertoire de travail :



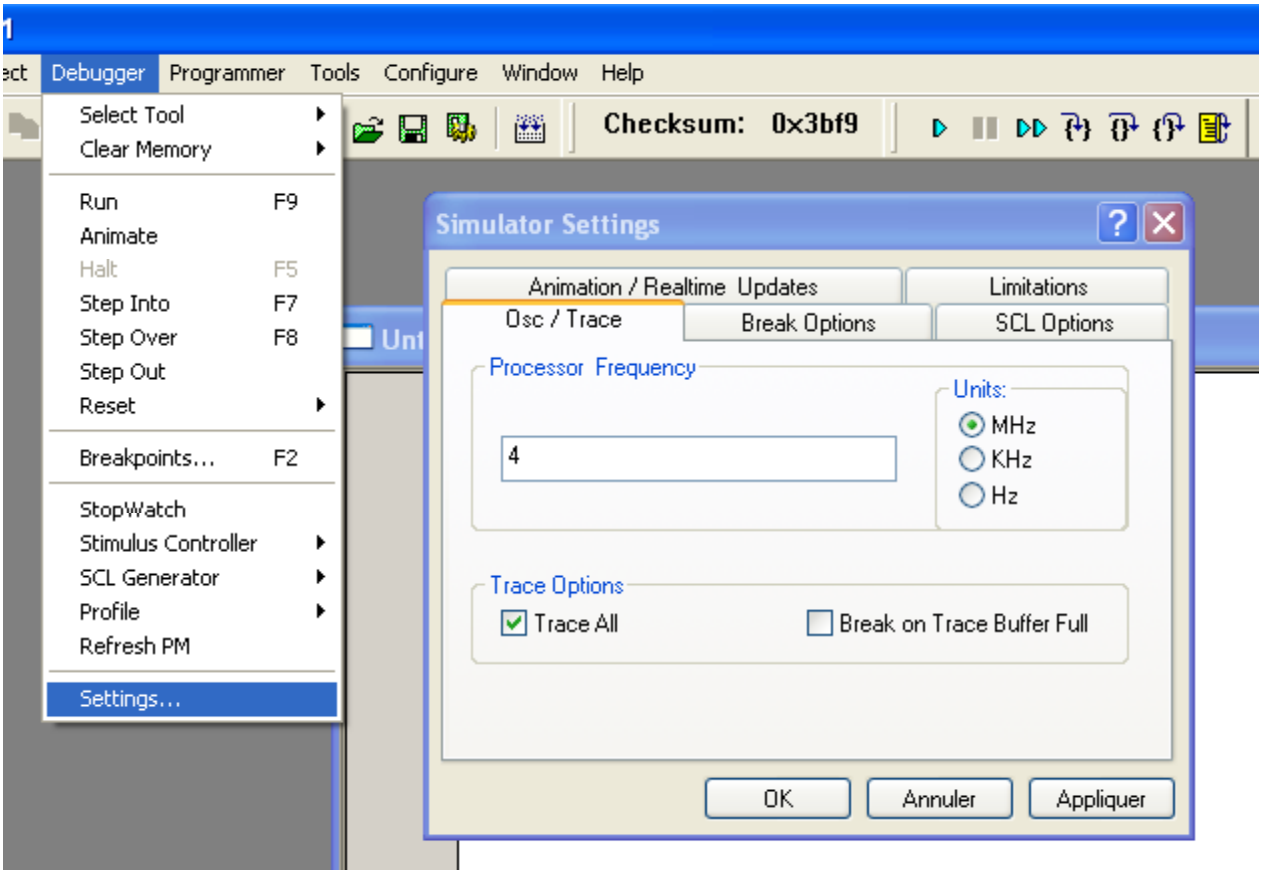
Nous verrons dans le chapitre suivant, le fichier devant être transféré dans le PIC.

4 – SIMULATION

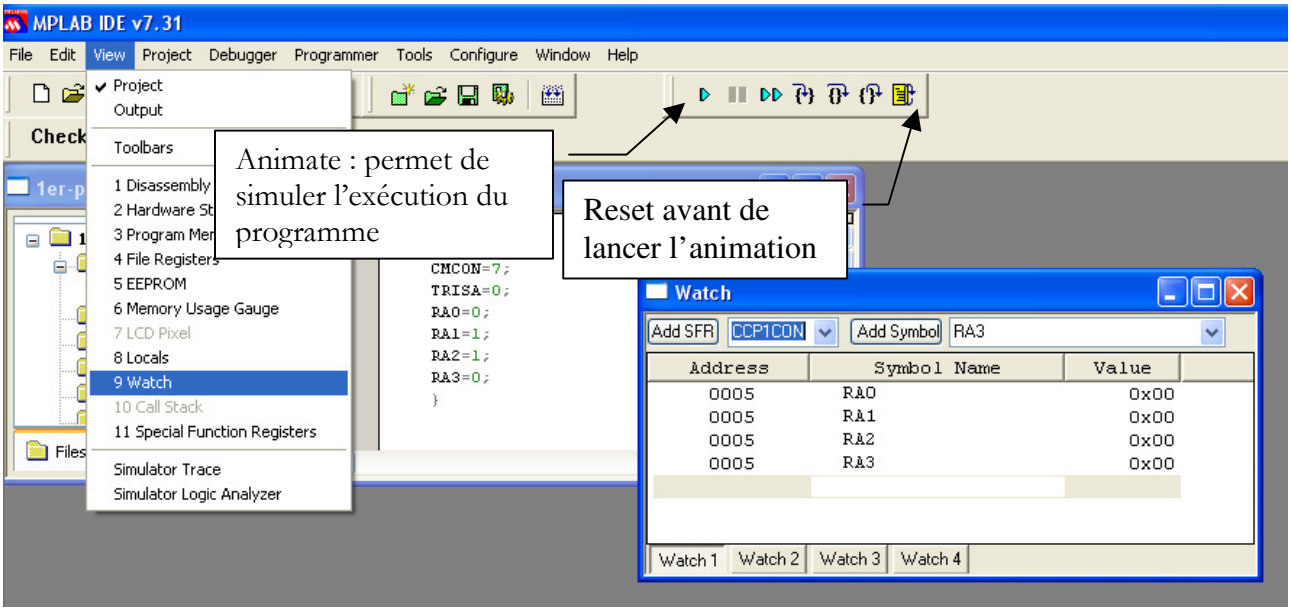
Comme indiqué au chapitre 1, il faut préciser au logiciel que l'outil de mise au point est MPLAB SIM grâce au menu Debbuger, Select Tool :



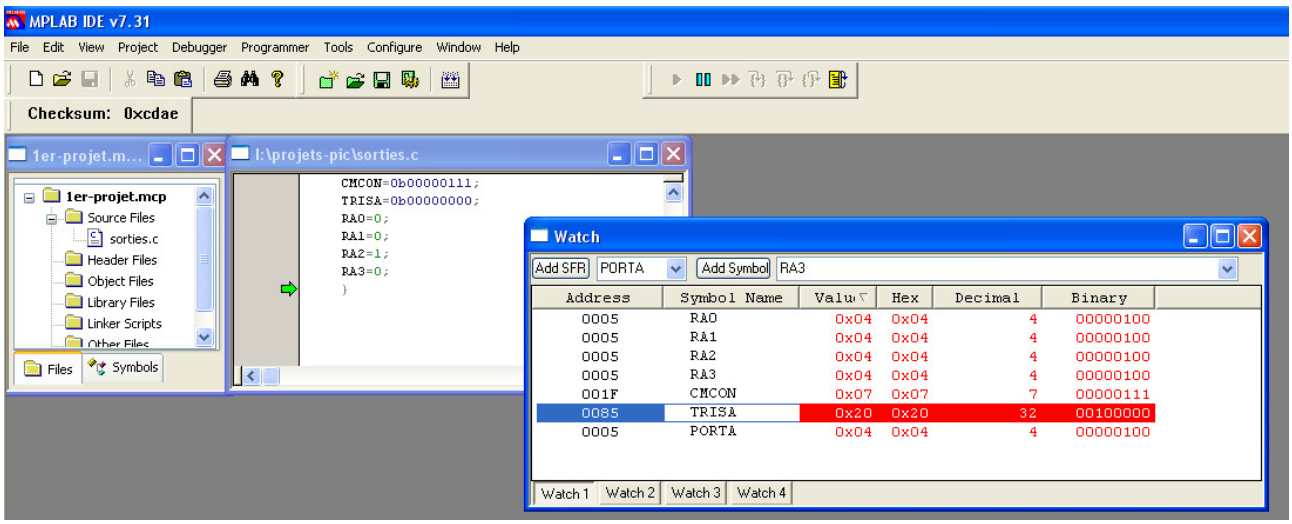
Dans le menu Debugger, de nouvelles sélections apparaissent. Choisir settings pour définir quelques options pour la simulation, en particulier la fréquence de l'horloge dépendant du PIC choisi (4 Mhz pour un 16F628 utilisé avec l'horloge interne (Cf Configuration bits INTRC I/O sur on).



Avant de simuler le fonctionnement du programme, il faut définir ce qu'il convient d'observer. Pour cela sélectionner Watch dans le menu View :



La compilation ayant été réalisée auparavant, on peut sélectionner Add symbol, RA0 pour visualiser l'état de RA0 lors de la simulation du programme.
Puis sélectionner dans la liste Add SFR : CMCON et TRISA pour visualiser l'état de ces registres.
Sélectionner aussi PORTA pour voir le mot binaire disponible sur le port A du pic.



Lancer l'exécution de la simulation. On observe alors la modification des valeurs des registres et du port de sortie.

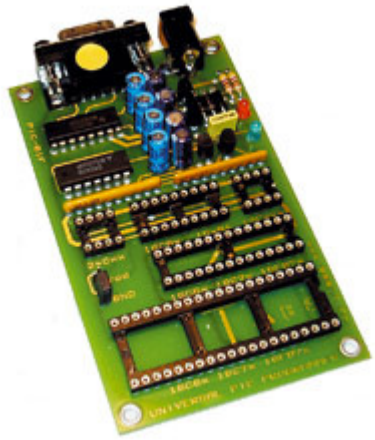
Noter que PORTA et les bits RA0, RA1 etc … affichent en réalité la même information qui est le mot binaire disponible sur le port de sortie, donc de chaque bit RA0 à RA7.

Chapitre 3 – LE PROGRAMMATEUR

1 – CONSTITUTION DU PROGRAMMATEUR

Le programmateur de PIC est constitué d'un circuit imprimé relié par câble au port COM de l'ordinateur.

Ce programmateur PIC-01 sera relié à une alimentation stabilisée 16V.
Les alimentations stabilisées traditionnellement réglées à 12 V pour les TP d'électronique devront donc être ajustées à 16 V.



Le PIC-01 permet la programmation des microcontrôleurs PIC de chez MICROCHIP (familles PIC12Cxxx, PIC12Cxxx, PIC16Cxxx et PIC16Fxxx), ainsi que les EEPROM séries (famille 24 Cxx). Connectable sur le port série de tout compatible PC, il fonctionne avec un logiciel sous Windows 95/98/NT/2000 et maintenant XP. Il supporte les boîtiers DIP 8, 18, 28 et 40 broches permettant la programmation de plus de 60 composants différents.

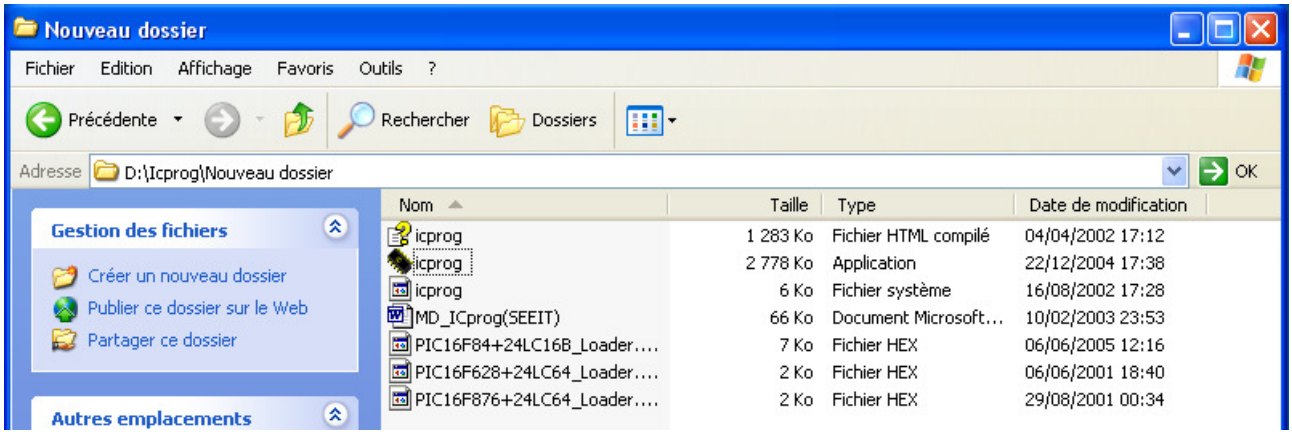
Le PIC utilisé sera placé sur un premier support tulipe, duquel il ne devra pas être ôté, afin d'éviter de tordre puis casser les pattes du microcontrôleur lors des manipulations.

On veillera à ne pas se tromper sur le sens de branchement du PIC sur le programmateur :

2 – INSTALLATION DU LOGICIEL

Le logiciel IC-prog fonctionne avec le programmateur PIC-01.
Les mises à jour du logiciel sont téléchargeables sur www.seeit.fr

Décompresser les fichiers téléchargés dans un répertoire. Bien vérifier que le fichier système icprog est bien présent dans ce répertoire.

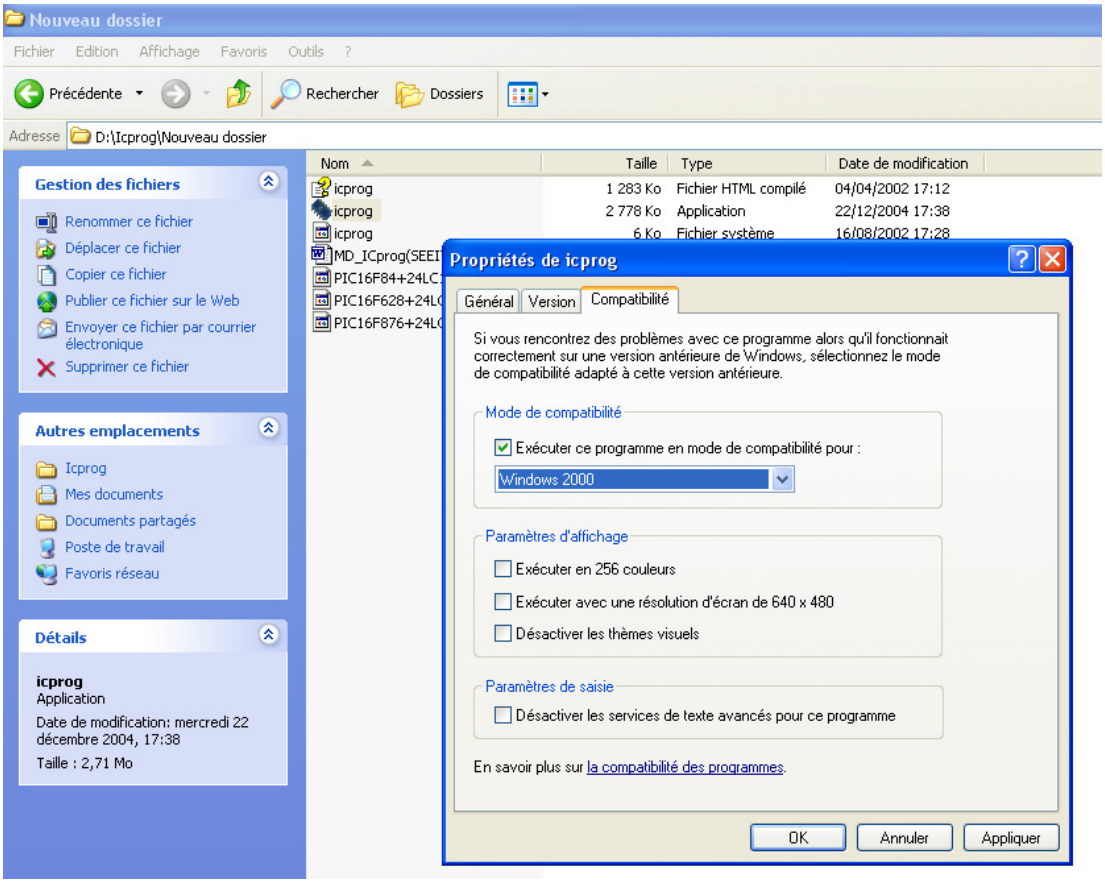


Lancer le logiciel en double cliquant sur l'application icprog.

3 – CONFIGURATION

3.1 – Configuration sous Windows XP

Sous WindowsXP, avec l'explorateur Windows, il faut sélectionner le fichier ICprog.exe. Faire un clic droit sur le fichier ICprog.exe. Dans le menu « Propriétés », sélectionner l'onglet « Compatibilité », cocher la case située dans le cadre « Mode de compatibilité », puis sélectionner « Windows 2000 » dans le menu déroulant.



3.2 - Configuration\Hardware F3

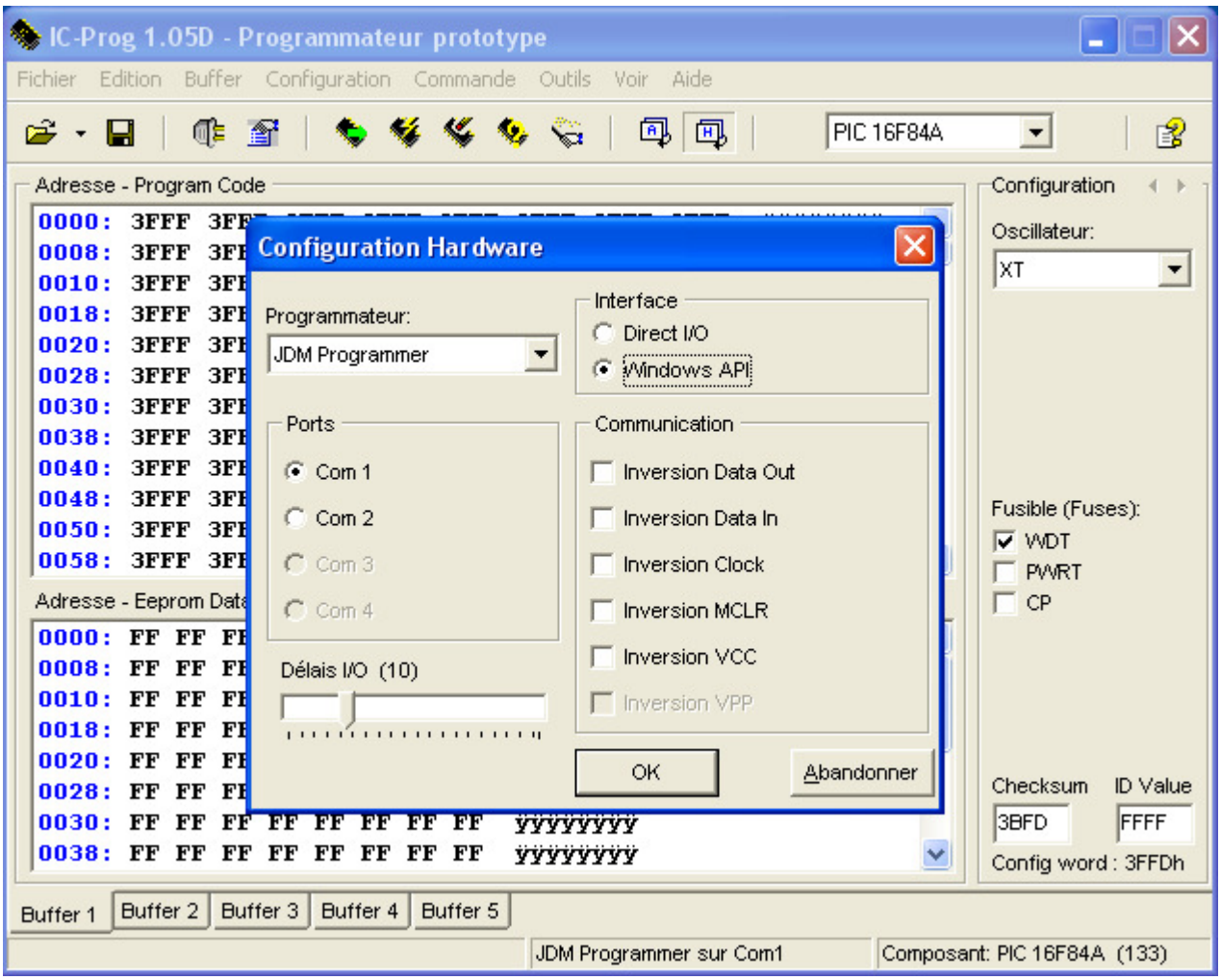
Permet de configurer l'interface de programmation entre le logiciel et la carte de programmation.

Programmateur :
JDM programmer pour le programmateur PIC-01

Ports :
COM1 ou COM2. Dans tous les cas la LED verte de votre programmateur doit s'allumer lorsque vous effectuez une opération de lecture ou d'écriture. Si ce n'est pas le cas changez de port sélectionné.

Délais I/O :
Ce réglage dépend du PC utilisé, essayez sur 1 ou sur 20 en cas de problème de programmation.

Interface :
Sélectionner toujours Windows API.



Communication :
Permet d'inverser les signaux envoyés ou reçus sur le port série. En général aucune case n'est cochée.
Pour la configuration exacte en fonction du programmateur utilisé, se référer au fichier « MiseEnOeuvreXXX-XX.doc » se trouvant sur la disquette livrée avec le PIC01.

3.3 - Configuration\Options\Misc

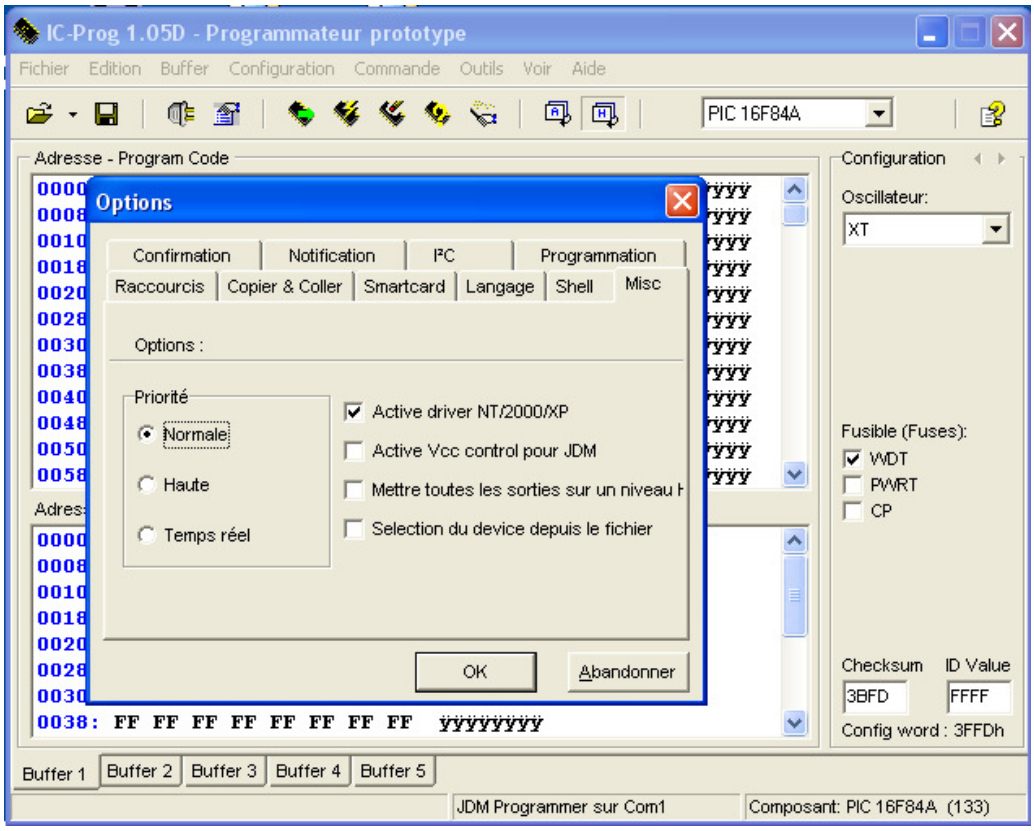
Priorité:
Permet de définir la priorité du logiciel par rapport aux autres logiciels fonctionnant en multitâches sous Windows. En général utiliser le mode « normal ». Utiliser le mode « haute » pour que ICprog soit prioritaire par rapport aux autres logiciels.

Active Driver NT/2000/XP :
Sous Windows 95/98/ME cette option n'est pas accessible. Sous Windows NT/2000/XP cocher cette case. Vérifier dans ce cas que le fichier « ICprog.sys » se trouve bien dans le même répertoire que ICprog.exe.

Active Vcc Control pour JDM :
Ne pas cocher cette case.

Mettre toutes les sorties au niveau haut :
Cette fonction permet de mettre toutes les sorties du port parallèle au niveau haut lorsque le port série est utilisé et de mettre toutes les sorties du port série au niveau haut lorsque le port

parallèle est utilisé. Cette fonction sert uniquement lorsque l'on utilise un programmateur spécial ayant à la fois le port série et le port parallèle de connecté sur le PC.



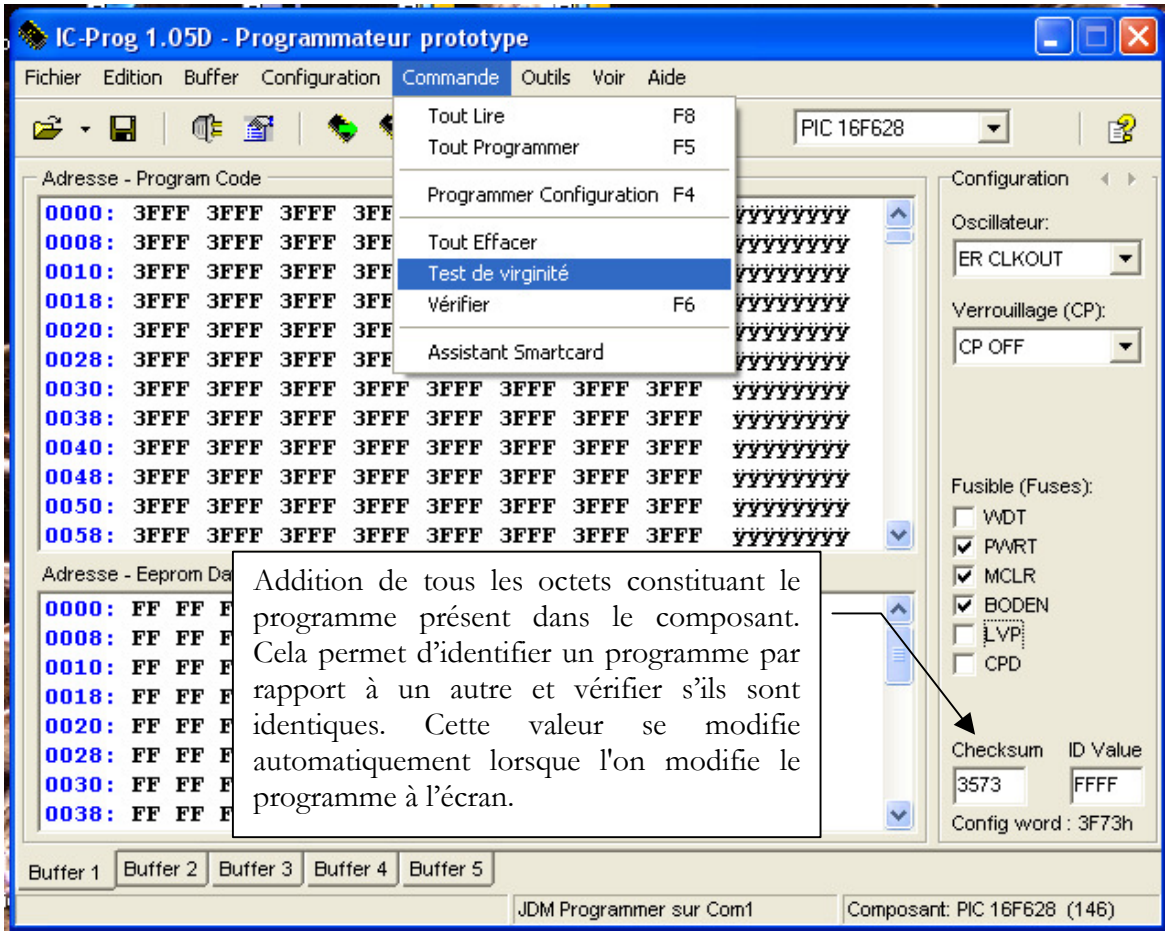
3 – PREMIERE PROGRAMMATION 16F628

4.1 - PRINCIPE

Le logiciel du programmateur utilise un buffer, c'est à dire une mémoire intermédiaire entre les fichiers sur disques et les mémoires programmables des composants, tableau hexadécimal visualisé à l'écran.
Pour programmer un composant à partir d'un fichier il faut d'abord charger le contenu d'un fichier dans le buffer à l'aide de la commande « Fichier\Ouvrir fichier », puis transférer le contenu du buffer vers le composant avec le menu « Commande\Tout programmer ».

4.3 – TEST DE VIRGINITE

Relier le programmateur PIC-01 au port COM du PC par l'intermédiaire du câble.
Placer un PIC dans le bon sens sur le support adéquat.
Alimenter le programmateur à l'aide de l'alimentation stabilisée réglée à 16 V (vérifier au voltmètre).
Lancer le logiciel ICprog. Menu Commande/Test de virginité, permet de vérifier si le composant est vide.



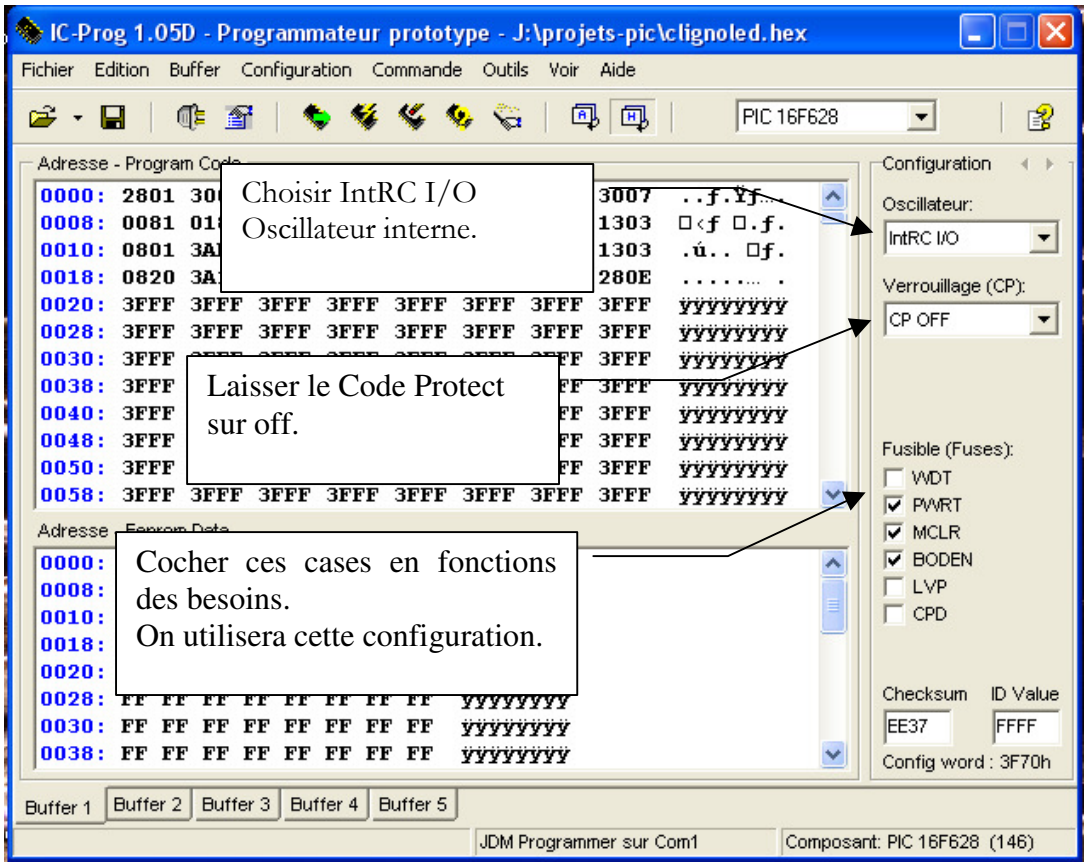
Si le composant est vierge ou effacé tous les bits de la mémoire seront au niveau logique 1 (FF). Cette fonction est à utiliser avant toute programmation car il n'est pas possible de programmer un composant correctement si celui-ci n'est pas vierge ou n'a pas été effacé préalablement.
Si ce n'est pas le cas, il faut effacer le composant : menu "Commande\Tout Effacer".

4.3 – CHOIX DU COMPOSANT, CONFIGURATION

Permet de sélectionner un microcontrôleur PIC du type 12Cxxx, 12Fxxx, 16Cxxx, 16Fxxx, 18Fxxx pour une utilisation avec le programmateur PIC-01. Pour les composants de la série 16C54/55/56/57/58, le mode de programmation est différent et il faut utiliser le programmateur PIC-02.

Différentes options apparaîtront également dans le cadre "Configuration" permettant de modifier les registres de configurations. Pour connaître l'utilisation de ces registres veuillez consulter le datasheet du fabricant concerné. Cependant quelques informations vous sont données ci-dessous pour les microcontrôleurs PIC.

Un choix entre plusieurs oscillateurs peut être réalisé.
Cette sélection dépend du type d'oscillateur connecté sur les entrées OSC1/CLKIN et OSC2/CLKOUT lors de l'utilisation du microcontrôleur sur son circuit final après la programmation. Pour les modes XT, un oscillateur à quartz ou un oscillateur TTL/C-MOS est connecté sur les entrées OSC1/CLKIN et OSC2/CLKOUT. Pour le mode RC, un pont RC est connecté sur l'entrée OSC1/CLKIN, (fréquence moins précise).



Validation ou non du WDT :
En validant cette case par une croix, le "Watchdog timer" sera activé. C'est à dire qu'un oscillateur interne indépendant de l'oscillateur externe sera fonctionnel même si le microcontrôleur est en position sommeil.

Validation ou non du PWRT :
En validant cette case par une croix, le "Power-up Timer" sera activé. Le microcontrôleur effectuera à sa mise sous tension un Reset général d'une durée de 72ms, le temps que la tension d'alimentation se stabilise.

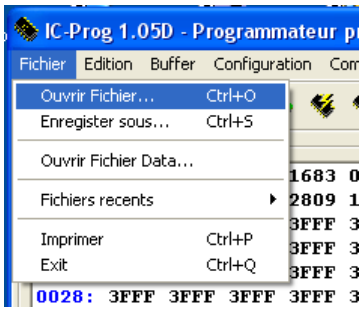
Validation ou non du MCLR :
En validant cette case par une croix, le "Memory Clear" sera activé. Il sera possible de faire une remise à zéro externe par la broche " GP3\MCLR\Vpp " du microcontrôleur. Cette borne sera reliée au +5V du pic à travers une résistance (2,2 kΩ par exemple).

Validation ou non du CP :
En validant cette case par une croix, le "Code Protect" sera activé. Le programme intégré dans la mémoire du composant ne sera pas lisible si l'on fait une re-lecture de celui-ci. Cependant le composant reste effaçable pour être reprogrammé si celui -ci contient une mémoire Flash.
Attention si vous cochez cette case, le composant ne pourra pas être vérifié après programmation et un message d'erreur interviendra systématiquement lors de la vérification du composant après programmation. **On évitera donc de cocher cette case.**

D'autres explications sur le rôle des fusibles seront données plus loin.

4.4 – EXEMPLE DE PROGRAMMATION

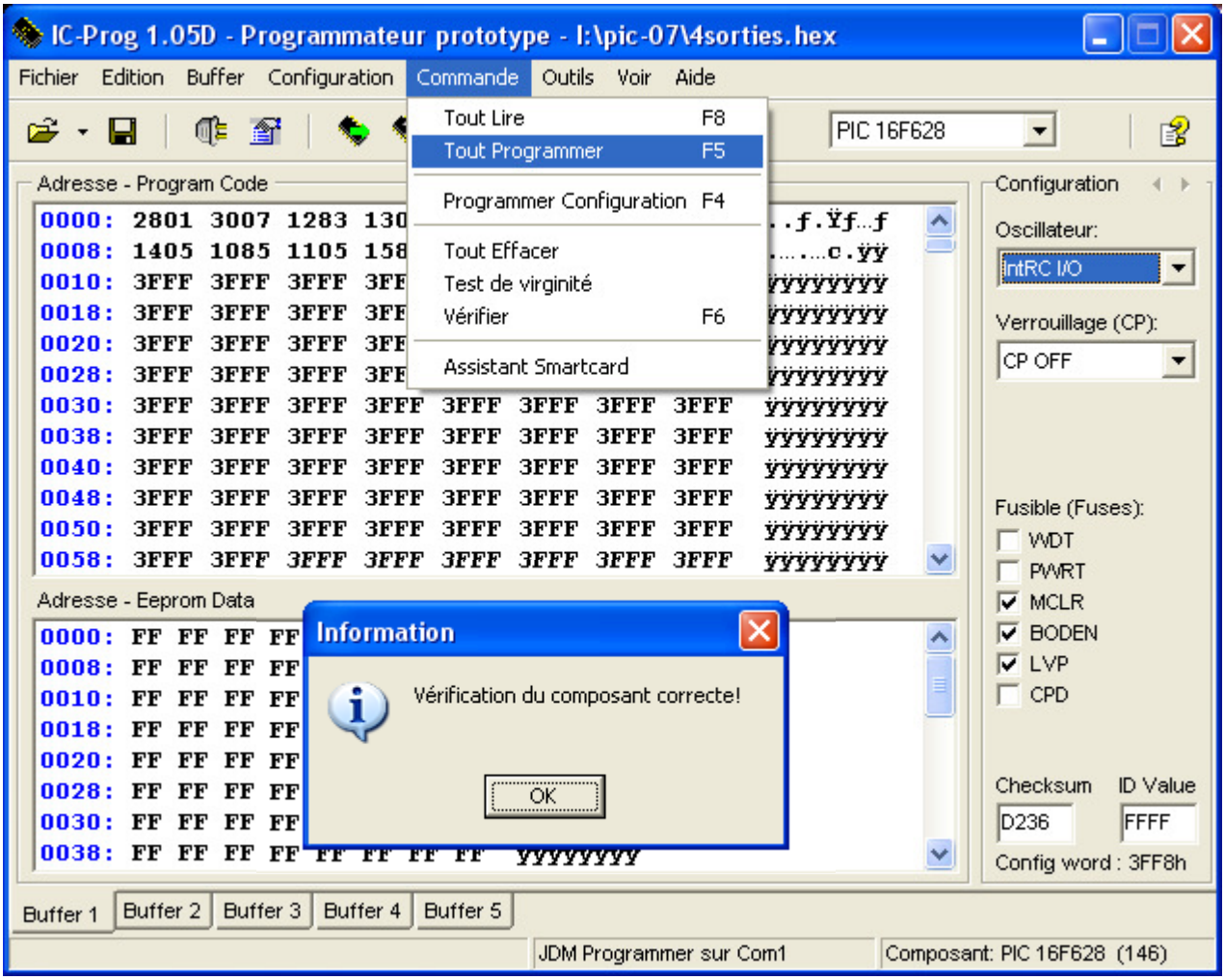
Dans ICprog, ouvrir le fichier sorties.hex créé au chapitre 2 § 3.



Le programme peut être affiché en hexadécimal ou en assembleur dans la fenêtre Adresse-Program Code.

On constate que le Checksum a changé de valeur.

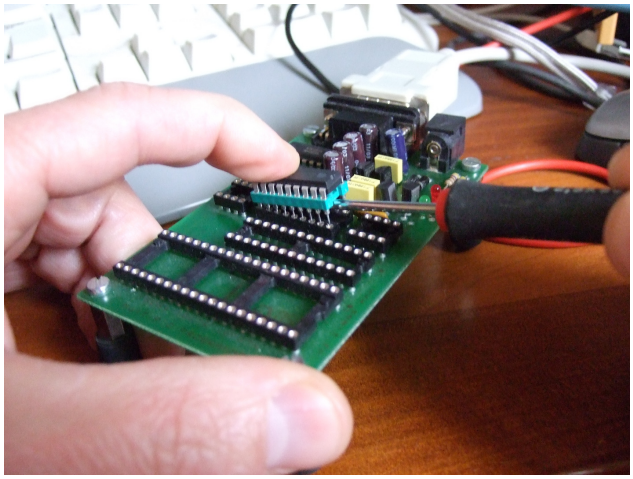
Vérifier que la configuration des fusibles correspond à celle de la compilation du programme dans MP Lab, puis choisir Commande/ Tout programmer.



Lorsque le transfert du programme dans le pic est réalisé, le logiciel procède à une vérification. Si un message d'erreur apparaît, il peut s'agir d'un mauvaise connexion du programmateur (erreur de port série) ou d'une mauvaise alimentation du programmateur.

4 – UTILISATION DU PIC DANS UN MONTAGE

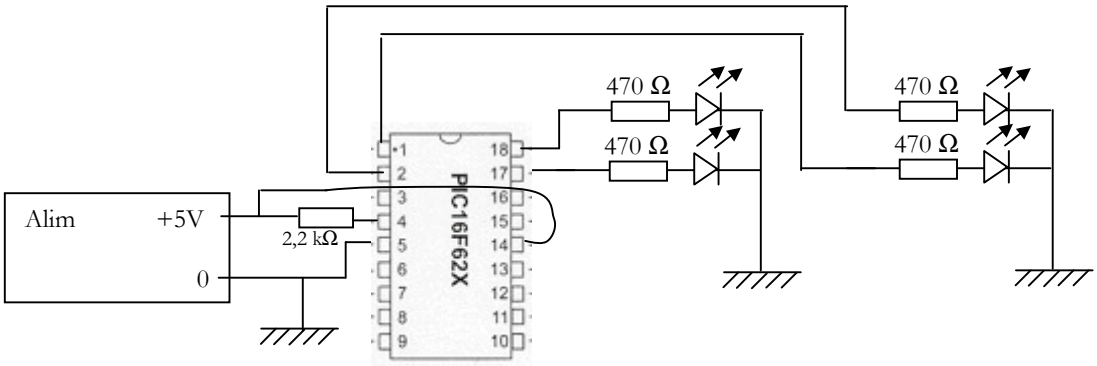
Le microcontrôleur ayant été programmé, il faut maintenant tester le fonctionnement du circuit dans le montage auquel il est destiné.

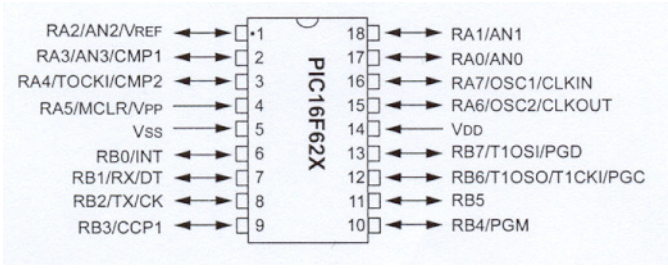


Mettre le programmateur hors tension en coupant l'alimentation stabilisée.
Sortir délicatement le PIC et son premier support de la carte programmateur. Utiliser une pince ou un tournevis glissé entre les deux supports.

Implanter le composant et son support sur une platine d'essais type Labdec.

Réaliser le câblage du montage correspondant au programme sur la platine Labdec :





Mettre sous tension et tester le fonctionnement.
Si tout s'est déroulé normalement, les leds branchées sur les sorties mises à 1 dans le programme sont allumées, les autres sont éteintes .

Chapitre 4 - MICROCONTROLEUR PIC 16F628

1 – PRESENTATION DU COMPOSANT

1.1 - INTRODUCTION

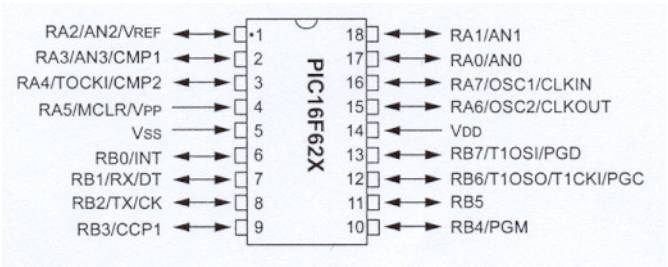
Le circuit 16F628 se présente sous la forme d'un circuit intégré disponible en boîtier DIL de 18 pattes.

Réalisé en technologie HCMOS FLASH, il constitue un microcontrôleur, c'est à dire un microprocesseur RISC c'est à dire à jeu d'instructions réduit (35 instructions) et de périphériques. Il est cadencé par une horloge interne ou externe pouvant avoir une fréquence de 0 à 20 MHz.

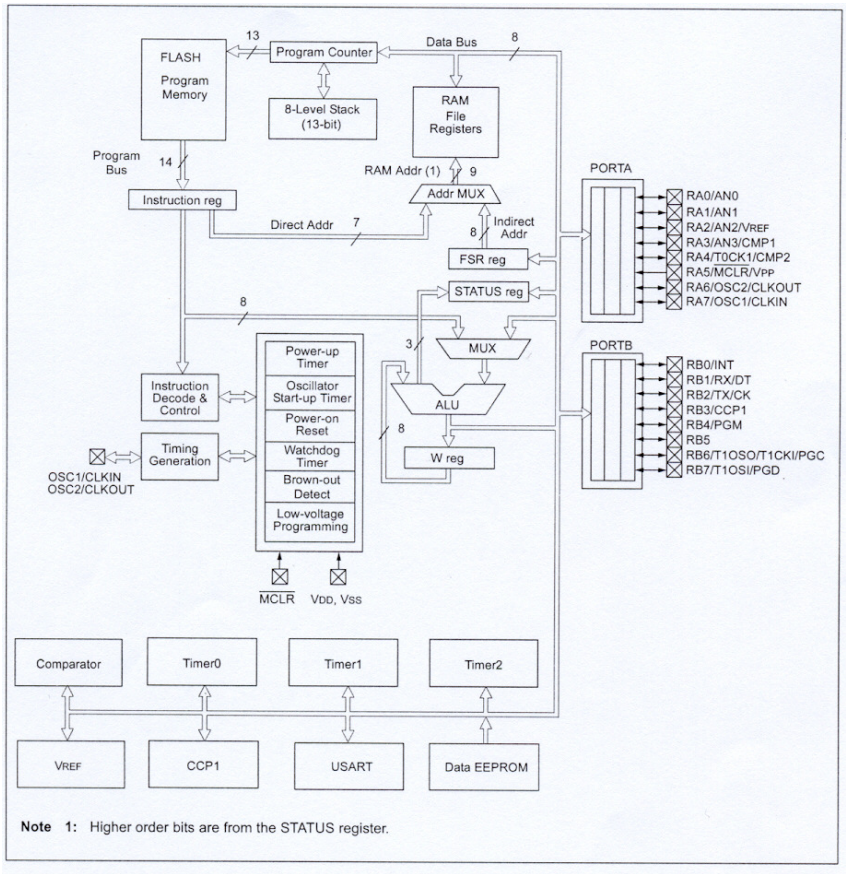
Ce composant dispose de en particulier de :

- deux ports d'entrée sortie. (RA et RB)
- d'un module comparateur analogique (AN et CMP)
- d'un module de capture et de comparaison de signaux PWM (CCP)

1.2 - BROCHAGE



1.3 – SCHEMA BLOCK



2 – PORTS D'ENTREES/SORTIES : Utilisation en Entrée.

2.1 – DESCRIPTION

Ce microcontrôleur dispose de deux ports bidirectionnels d'E/S : PORTA et PORTB de 8 bits

Ces ports peuvent servir d'E/S standard ou d'E/S de périphériques. En effet, certaines pattes de ces ports sont multiplexées avec d'autres fonctions de périphériques internes (comparateur et référence de tension par exemple). Chaque borne du port a donc plusieurs rôles qui doivent être définis par des registres de configuration associés. Quand le périphérique est activé, la borne ne peut plus être utilisée en E/S.

2.2 – PORT A

2.2.1 – FONCTION MULTIPLEXES

Le tableau ci-dessous décrit les différentes fonctions multiplexées sur le port A.

Name	Function	Input Type	Output Type	Description
RA0/AN0	RA0	ST	CMOS	Bi-directional I/O port
	AN0	AN	—	Analog comparator input
RA1/AN1	RA1	ST	CMOS	Bi-directional I/O port
	AN1	AN	—	Analog comparator input
RA2/AN2/VREF	RA2	ST	CMOS	Bi-directional I/O port
	AN2	AN	—	Analog comparator input
	VREF	—	AN	VREF output
RA3/AN3/CMP1	RA3	ST	CMOS	Bi-directional I/O port
	AN3	AN	—	Analog comparator input
	CMP1	—	CMOS	Comparator 1 output
RA4/T0CKI/CMP2	RA4	ST	OD	Bi-directional I/O port
	T0CKI	ST	—	External clock input for TMR0 or comparator output. Output is open drain type
	CMP2	—	OD	Comparator 2 output
RA5/$\overline{\text{MCLR}}$/VPP	RA5	ST	—	Input port
	$\overline{\text{MCLR}}$	ST	—	Master clear
	VPP	HV	—	Programming voltage input. When configured as $\overline{\text{MCLR}}$, this pin is an active low RESET to the device. Voltage on $\overline{\text{MCLR}}$/VPP must not exceed VDD during normal device operation
RA6/OSC2/CLKOUT	RA6	ST	CMOS	Bi-directional I/O port.
	OSC2	XTAL	—	Oscillator crystal output. Connects to crystal resonator in Crystal Oscillator mode.
	CLKOUT	—	CMOS	In ER/INTRC mode, OSC2 pin can output CLKOUT, which has 1/4 the frequency of OSC1
RA7/OSC1/CLKIN	RA7	ST	CMOS	Bi-directional I/O port
	OSC1	XTAL	—	Oscillator crystal input
	CLKIN	ST	—	External clock source input. ER biasing pin.

Legend: ST = Schmitt Trigger input HV = High Voltage OD = Open Drain AN = Analog

Certaines de ces fonctions seront examinées dans les paragraphes suivants.

2.2.2 - UTILISATION EN ENTREES SORTIES NUMERIQUES

a) **Registre CMCON**

Les pattes du port A étant multiplexées avec les entrées du comparateur, il convient de définir leur rôle grâce au registre CMCON (Comparator Control Register) registre de contrôle du comparateur.

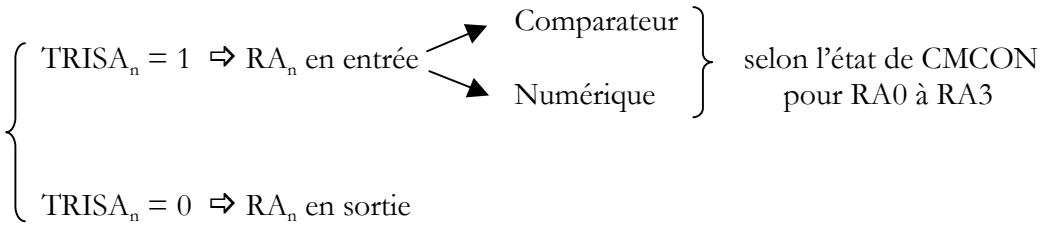
On doit avoir : CMCON = 0b00000111 = 0x07 = 7 pour forcer toutes les entrées en entrées numériques.

b) **Registre TRISA**

Ce registre permet de définir si la patte considérée fonctionne en entrée ou en sortie.

Un « 1 » dans un bit du registre TRISA met la sortie correspondante en haute impédance, elle peut ainsi servir d'entrée.

Un « 0 » dans une bit de ce registre transfert le contenu de la sortie de la bascule D sur la sortie correspondante.



Remarque : $TRISA_6$ et $TRISA_7$ sont forcés par la configuration de l'oscillateur. Dans ce cas la donnée lue est « 0 » et ces deux bits sont alors ignorés.

Remarque : Toute opération d'écriture sur une des sorties est précédée d'une lecture de la patte correspondante.

2.3 – PORT B

2.3.1 – FONCTION MULTIPLEXES

Le tableau ci-dessous décrit les différentes fonctions multiplexées sur le port B.

Name	Function	Input Type	Output Type	Description
RB0/INT	RB0	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	INT	ST	—	External interrupt.
RB1/RX/DT	RB1	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	RX	ST	—	USART Receive Pin
	DT	ST	CMOS	Synchronous data I/O
RB2/TX/CK	RB2	TTL	CMOS	Bi-directional I/O port
	TX	—	CMOS	USART Transmit Pin
	CK	ST	CMOS	Synchronous Clock I/O. Can be software programmed for internal weak pull-up.
RB3/CCP1	RB3	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	CCP1	ST	CMOS	Capture/Compare/PWM/I/O
RB4/PGM	RB4	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	PGM	ST	—	Low voltage programming input pin. Interrupt-on-pin change. When low voltage programming is enabled, the interrupt-on-pin change and weak pull-up resistor are disabled.
RB5	RB5	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
RB6/T1OSO/T1CKI/PGC	RB6	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSO	—	XTAL	Timer1 Oscillator Output
	T1CKI	ST	—	Timer1 Clock Input
	PGC	ST	—	ICSP Programming Clock
RB7/T1OSI/PGD	RB7	TTL	CMOS	Bi-directional I/O port. Interrupt-on-pin change. Can be software programmed for internal weak pull-up.
	T1OSI	XTAL	—	Timer1 Oscillator Input
	PGD	ST	CMOS	ICSP Data I/O

Legend: O = Output, — = Not used, TTL = TTL Input, CMOS = CMOS Output, I = Input, OD = Open Drain Output, P = Power, ST = Schmitt Trigger Input, AN = Analog

Certaines de ces fonctions seront examinées dans les paragraphes suivants.

2.3.2 - UTILISATION EN ENTREES SORTIES NUMERIQUES

a) **Multiplexage sur le port B**

Le port B est multiplexé avec :

- interruption externe
- USART
- CCP module
- TMR1 clock in/out

b) **Registre TRISB**

Ce registre permet de définir si la patte considérée fonctionne en entrée ou en sortie.

Un « 1 » dans un bit du registre TRISB met la sortie correspondante en haute impédance, elle peut ainsi servir d'entrée.

Un « 0 » dans une bit de ce registre transfert le contenu de la sortie de la bascule D sur la sortie correspondante.

- $TRISB_n = 1$ ⇨ RB_n en entrée
- $TRISB_n = 0$ ⇨ RB_n en sortie

Remarque : Toute opération d'écriture sur une des sorties est précédée d'une lecture de la patte correspondante.

3 – UTILISATION DU TIMER 0

Le composant dispose de 3 timers : timer0 (TMR0), timer1 (TMR1) et timer2 (TMR2)

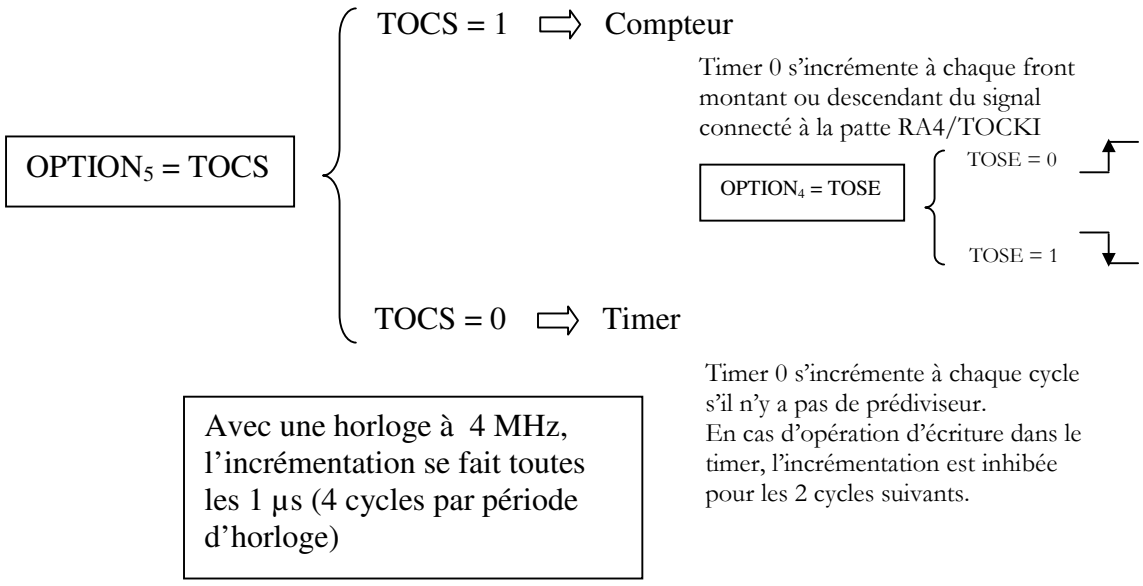
3.1 – CARACTERISTIQUES DU TIMER 0

Le module Timer0 a les caractéristiques suivantes :

- timer ou compteur 8 bits
- Utilisable en lecture ou écriture
- Pré diviseur 8 bits programmable
- Sélection de l'horloge interne ou externe
- Interruption sur dépassement
- Sélection du front montant ou descendant pour l'horloge externe

3.2 – SELECTION DU MODE TIMER OU COMPTEUR

Cette sélection s'opère grâce au 5ème bit TOCS du registre OPTION



3.3 – REGISTRE OPTION

OPTION REGISTER (ADDRESS: 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Choix du taux de division

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

3.4 - PREDIVISEUR

Le pré-diviseur est partagé entre Timer0 et le chien de Garde (Watchdog Timer) ; Ce choix s'opère par l'état du bit 3 PSA du registre OPTION.

$OPTION_3 = PSA$

PSA = 1 ⇨ Pré diviseur sur le Watchdog timer

PSA = 0 ⇨ Pré diviseur sur le Timer0

Le taux de division est alors réglable par les bits PS0, PS1 et PS2 du registre OPTION (Voir ci dessus § 3.3)

3.5 – EXEMPLE : Temporisation

Pour utiliser le TIMER0 avec pré-division par 256, il faudra une ligne de code dans le programme :

```
        OPTION=0b00000111 ;   |  TMR0 = 0 ;
Ou      OPTION=0x07 ;         |  do {
Ou      OPTION=7 ;            |  }
                              |  while (TMR0 < 240);
```

3.6 – INTERRUPTION DU TIMER0

Une interruption est générée par le timer0 si le timer ou le compteur passe de xFF à x00 (en hexadécimal).

Ce dépassement met à 1 le bit TOIF, bit 2 du registre INTCON.

On peut activer ou pas cette interruption par le bit TOIE, bit 5 du registre INTCON.

$INTCON_5 = TOIE$

- TOIE = 1 ⇨ Interruption timer0 active
- TOIE = 0 ⇨ Interruption timer0 inactive

Le taux de division est alors réglable par les bits PS0, PS1 et PS2 du registre OPTION (Voir ci dessus § 3.3)

INTCON REGISTER (ADDRESS: 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

⇨ bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE:** RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

⇨ bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = When at least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

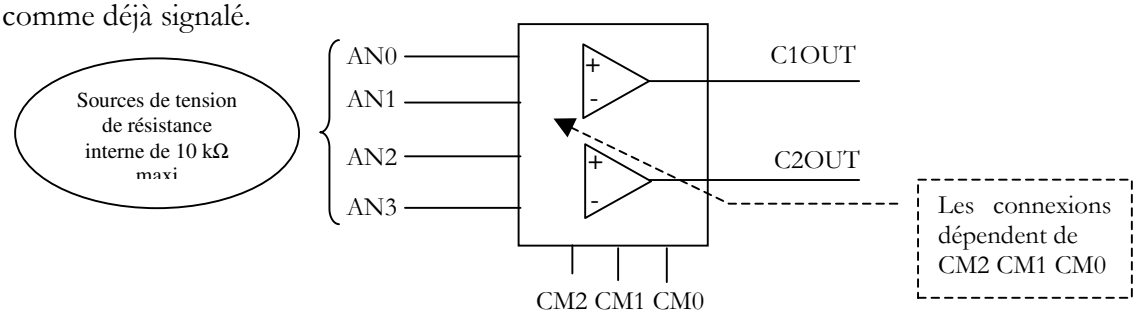
4 – UTILISATION DU COMPARATEUR

4.1 – RAPPEL

4.2 – LES COMPARATEURS

Le microcontrôleur comporte 2 comparateurs donc 4 entrées et 2 sorties.

Ces entrées sorties de comparateur sont multiplexées avec les entrées sorties numériques du port A comme déjà signalé.



Les fonctions de comparaisons réalisées sont définies par les bits CM2 CM1 et CM0 du registre CMCON (Cf § 4.3).

Le registre TRISA contrôle la direction des données (E/S) pour chaque entrée/sortie du port A même en mode comparateur. Il convient donc d'initialiser correctement TRISA.

4.3 – DIFFERENTS MODES DE COMPARAISON

Le tableau ci-dessous récapitule les différentes configurations possibles.

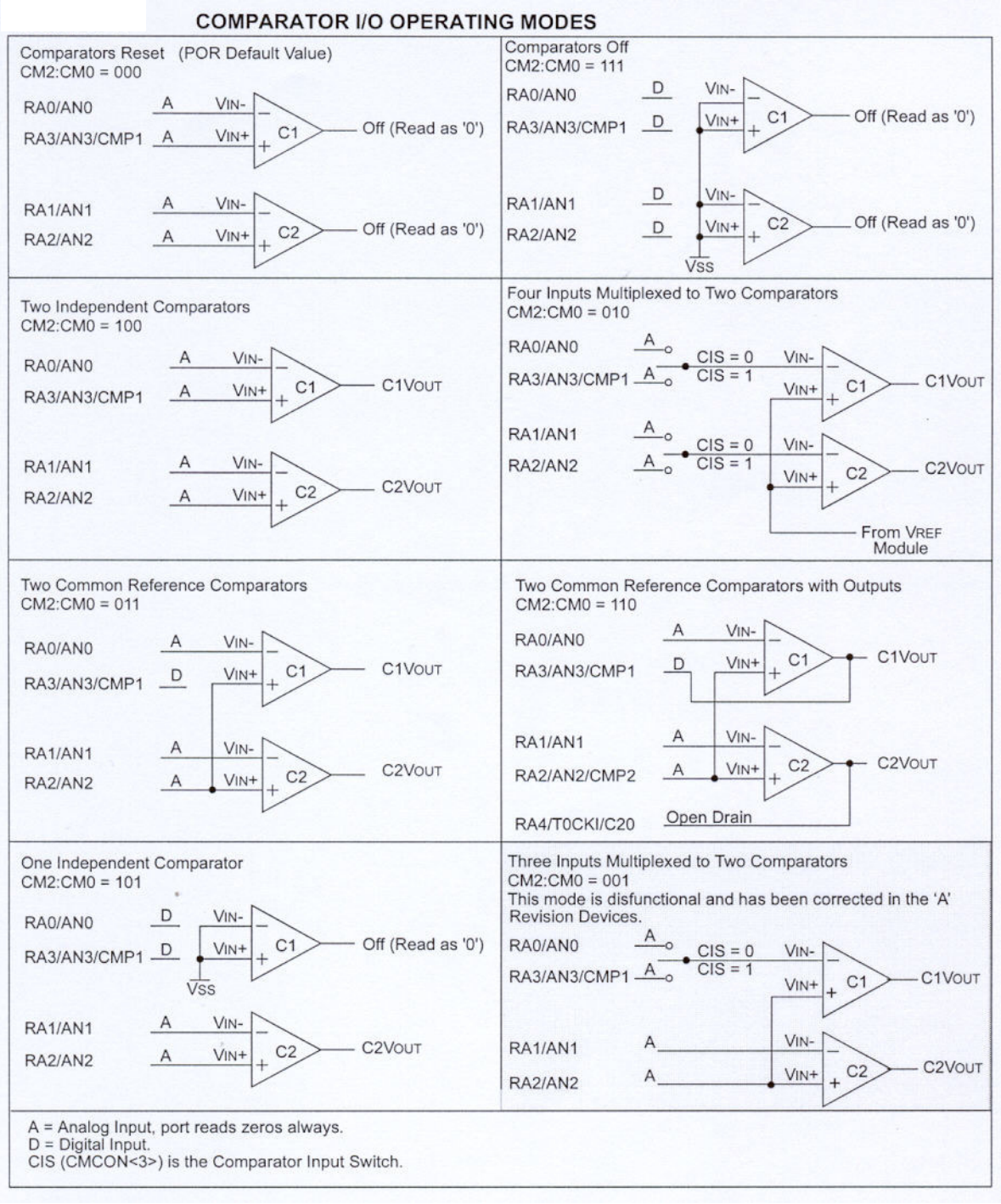
Exemple :

Si l'on souhaite utiliser un seul comparateur, les bits CM2 CM1 et CM0 du registre CMCON sont respectivement 101.

Le comparateur est alors constitué des deux entrées RA1/AN1 (patte 18) et RA2/AN2 (patte 1). Le résultat de la comparaison est disponibles sur les bits 6 et 7 du registre CMCON.

Ces deux bornes appartiennent au port A qui doit être configuré correctement à l'aide de TRISA : RA1/AN1 et RA2/AN2 en entrées

On a alors TRISA = 0bxxxxx**11x**.



4.4 – SORTIES DU COMPARATEUR

Les résultats des deux comparaisons sont disponibles sur les bits 6 et 7 du registre CMCON. (Ces deux bits sont en lecture seule).

Ils peuvent aussi être transmis sur les sorties RA3/AN3/CMP1 (patte 2) et RA4/TOCKI/CMP2 (patte3). Pour cela il faut être dans la configuration où CM2 :CM0=110.

Ces deux bornes appartenant au port A, il faut les configurer en sortie à l'aide de TRISA : TRISA = 0bxxx**00**xxx

4.5 – COMPLEMENTS SUR LE REGISTRE CMCON

CMCON REGISTER (ADDRESS: 01Fh)

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
bit 7							bit 0

bit 7 **C2OUT**: Comparator 2 Output
When C2INV = 0:
1 = C2 VIN+ > C2 VIN-
0 = C2 VIN+ < C2 VIN-

When C2INV = 1:
1 = C2 VIN+ < C2 VIN-
0 = C2 VIN+ > C2 VIN-

bit 6 **C1OUT**: Comparator 1 Output
When C1INV = 0:
1 = C1 VIN+ > C1 VIN-
0 = C1 VIN+ < C1 VIN-

When C1INV = 1:
1 = C1 VIN+ < C1 VIN-
0 = C1 VIN+ > C1 VIN-

bit 5 **C2INV**: Comparator 2 Output Inversion
1 = C2 Output inverted
0 = C2 Output not inverted

bit 4 **C1INV**: Comparator 1 Output Inversion
1 = C1 Output inverted
0 = C1 Output not inverted

Inversion des sorties
Du comparateur

bit 3 **CIS**: Comparator Input Switch
When CM2:CM0: = 001
Then:
1 = C1 VIN- connects to RA3
0 = C1 VIN- connects to RA0

When CM2:CM0 = 010
Then:
1 = C1 VIN- connects to RA3
C2 VIN- connects to RA2
0 = C1 VIN- connects to RA0
C2 VIN- connects to RA1

Bit de contrôle du multiplexage
des entrées des comparateurs

bit 2-0 **CM2:CM0**: Comparator Mode
Figure 9-1 shows the Comparator modes and CM2:CM0 bit settings

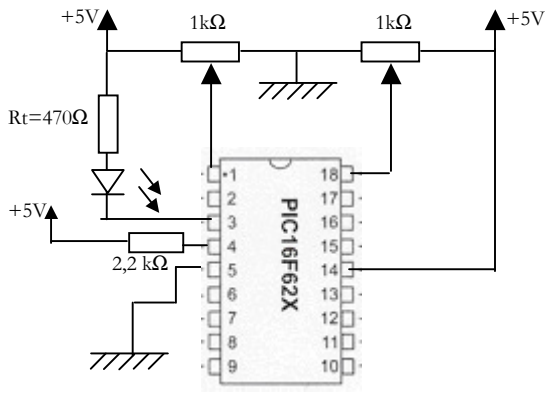
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.6 – EXEMPLE

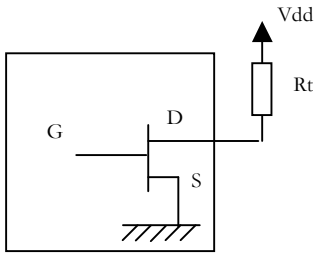
Le programme suivant permet de faire fonctionner le pic en comparateur, comme un simple ampli op en boucle ouverte.

```
void main(void)
        {
        CMCON=0b00000110 ; /*choix du mode de comparaison Cf tableau §4.3 */
        TRISA=0b00000110 ; /*RA1 et RA2 entrées, RA4 en sortie */
        }
```

Après voir compilé le programme source et programmé le composant, on peut câbler le montage suivant pour tester le fonctionnement :



La résistance Rt est une résistance dite de tirage (pull up).
Elle est nécessaire car la sortie RA4 en mode comparateur est à drain ouvert :

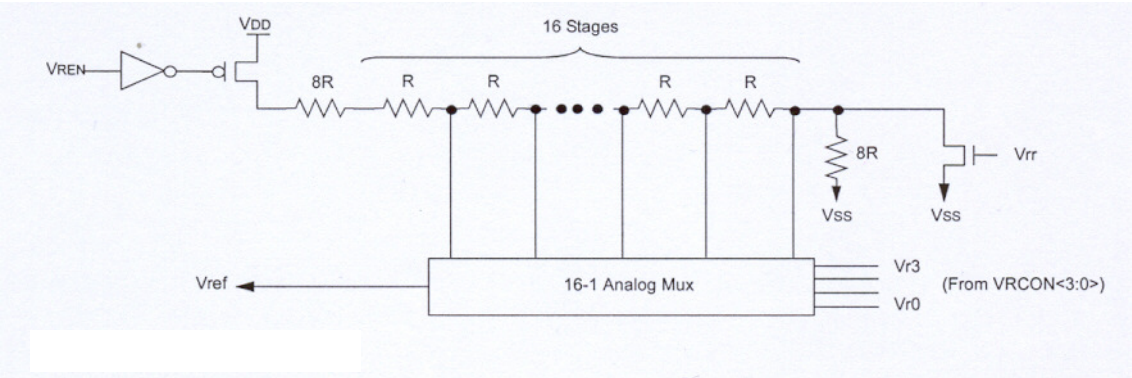


Si G = 0 => V_{DS} = 0 et V_{Rt} = Vdd
Si G = 1 => V_{DS} = Vdd et V_{Rt} = 0

5 – UTILISATION DE TENSIONS DE REFERENCE

5.1 – REALISATION DE LA TENSION DE REFERENCE

Le module est constitué d'un réseau de résistances en échelle permettant de fabriquer une tension de référence V_{Ref}.



5.2 – VALEUR DE LA TENSION DE REFERENCE

On dispose de deux gammes dont le choix s'opère par V_{RR} le bit 5 du registre VRCON.
La valeur exacte dans la gamme est commandée par V_R les bits 3 à 0 du registre VRCON.

$VRON_5 = V_{RR}$

$V_{RR} = 1$ $\quad Vref = V_R<3:0> * V_{DD}/24$

$V_{RR} = 0$ $\quad Vref = V_R<3:0> * V_{DD}/32 + V_{DD}/4$

$V_R<3:0>$ est la valeur décimale du mot binaire contitué par les bits V_{R3} V_{R2} V_{R1} V_{R0} du registre VRCON.

5.3 – TRANSMISSION DE LA TENSION DE REFERENCE SUR LA SORTIE

La tension de référence doit être activée par le bit 7 mis à 1 dans le registre VRCON.

La tension de référence élaborée est envoyé sur la sortie RA2/AN2/V_{REF} du port A si V_{ROE} le bit 6 du registre VRCON est à 1. Sinon, la tension de référence est déconnectée.

Ainsi on doit avoir VRCON = 0b**11**10 0110 => Vref = 6*V_{DD}/24 =1,25 V envoyé sur la sortie RA2.

En même temps, **la sortie RA2 du port A doit être configurée en entrée (?!)** par l'intermédiaire du registre TRISA : TRISA = 0bxxxxx**1**xx

```
void main(void)
        {
        VRCON=0b11100110 ;
        TRISA=0b00000100 ;
        }
```

5.4 – REGISTRE VRCON

VRCON REGISTER (ADDRESS: 9Fh)

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
VREN	VROE	VRR	—	VR3	VR2	VR1	VR0
bit 7							bit 0

bit 7 **VREN:** VREF Enable
1 = VREF circuit powered on
0 = VREF circuit powered down, no IDD drain

bit 6 **VROE:** VREF Output Enable
1 = VREF is output on RA2 pin
0 = VREF is disconnected from RA2 pin

bit 5 **VRR:** VREF Range selection
1 = Low Range
0 = High Range

bit 4 **Unimplemented:** Read as '0'

bit 3-0 **VR<3:0>**: VREF value selection 0 ≤ VR [3:0] ≤ 15
When VRR = 1: VREF = (VR<3:0>/ 24) * VDD
When VRR = 0: VREF = 1/4 * VDD + (VR<3:0>/ 32) * VDD

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

5.5 – UTILISATION EN CNA

La sortie Vref programmée par l'intermédiaire d'un mot binaire V_R<3 :0> constitue un Convertisseur Numérique Analogique. Cependant, cette sortie ne peut être chargée sans l'utilisation d'un étage suiveur.

6 – UTILISATION DU TIMER 1

6.1 – DESCRIPTION DU TIMER 1

Le module Timer 1 est un timer/compteur 16 bits constitué de deux registres 8 bits TMR1H et TMR1L pouvant être en lecture ou écriture.

Le Timer 1 s'incrémente donc de 0x0000 à 0xFFFF puis repasse à 0x0000. Le dépassement est signalé par le bit TMR1IF du registre PIR1.

6.2 – REGISTRE T1CON

Le bit 0 de ce registre permet d'activer ou désactiver le timer1.

$T1CON_0 = TMR1ON$

- TMR1ON = 1 : Timer 1 actif
- TMR1ON = 0 : Timer 1 désactivé

Les bits 4 et 5 permettent de choisir le taux de division de la fréquence d'horloge interne ou externe.

T1CON: TIMER1 CONTROL REGISTER (ADDRESS: 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut off(1)

bit 2 **$\overline{T1SYNC}$**: Timer1 External Clock Input Synchronization Control bit
TMR1CS = 1
1 = Do not synchronize external clock input
0 = Synchronize external clock input
TMR1CS = 0
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit
1 = External clock from pin RB6/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON**: Timer1 On bit
1 = Enable Timer1
0 = Stops Timer1

Note 1: The oscillator inverter and feedback resistor are turned off to eliminate power drain.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

6.3 – SELECTION DU MODE TIMER OU COMPTEUR

Cette sélection s'effectue grâce au bit TMRCS du registre T1CON.

$T1CON_1$ = TMR1CS

- TMR1CS = 0 ⇨ Timer
  - Timer 1 s'incrémente à chaque cycle, donc à Fosc/4.
- TMR1CS = 1 ⇨ Compteur
  - Timer 0 s'incrémente à chaque front montant de l'horloge externe

$T1CON_3$ = T1OSCEN

- T1OSCEN = 0 Horloge externe appliquée à RB6
- T1OSCEN = 1 Horloge externe appliquée à RB7

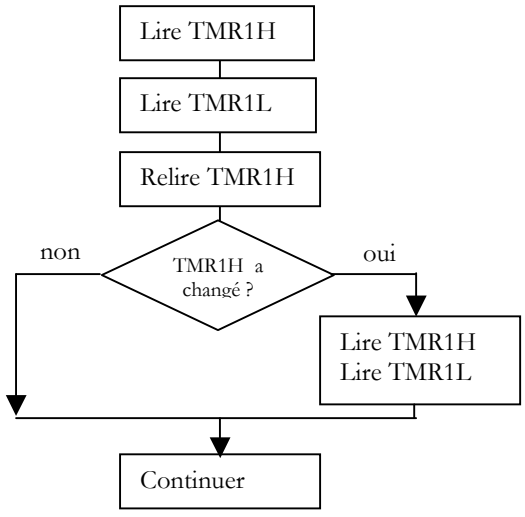
$T1CON_2$ = T1SYNC

- T1SYNC = 0 ⇨ Compteur synchrone
  - Horloge externe synchronisée avec l'horloge interne.
  - En mode sleep (sommeil), le compteur ne s'incrémente plus.
- T1SYNC = 1 ⇨ Compteur asynchrone
  - Horloge externe non synchronisée.
  - En mode sleep, le compteur continue de s'incrémenter. Il peut générer une interruption sur le dépassement pouvant être utilisée pour réveiller le processeur.

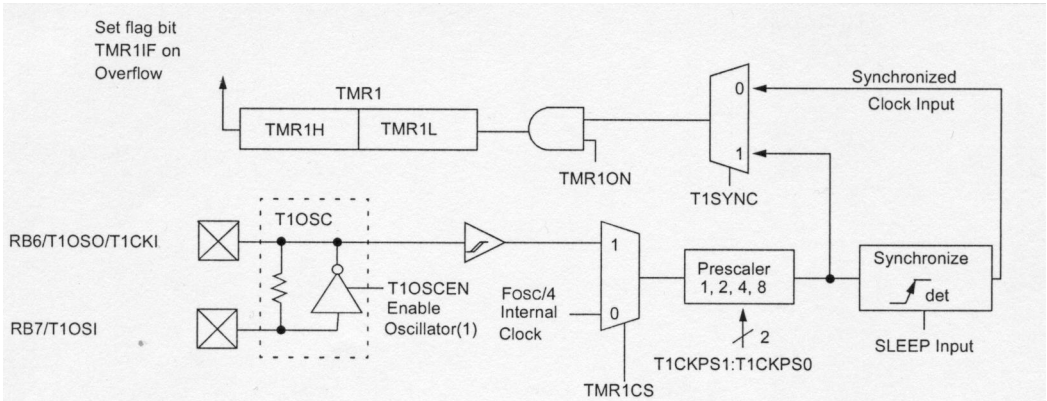
Dans ce mode, le timer 1 ne peut pas être utilisé comme base de temps pour le module CCP (Cf § 7)

La lecture et l'écriture du timer se fait en 2 opérations du fait de la séparation du timer 1 en deux registres TMR1H et TMR1L. Un dépassement peut intervenir entre temps.

- ⇨ Pour l'écriture, il est préférable d'arrêter le timer auparavant.
- ⇨ Pour la lecture, il faut réaliser la procédure suivante :



6.4 – BLOC DIAGRAMME DU TIMER 1



Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

7 – MODULE CCP

Ce module contient un registre 16 bit constitué en réalité de deux registres 8 bits : CCPR1H et CCPR1L.

Les opérations de ce module sont contrôlées par le registre CCP1CON.

Ce module peut fonctionner de trois façons :

- mode capture, en relation avec le timer 1
- mode compare, en relation avec le timer 1
- mode PWM, en relation avec le timer 2.

7.1 – REGISTRE CCP1CON

CCP1CON REGISTER (ADDRESS: 17h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCP1X:CCP1Y**: PWM Least Significant bits
Capture Mode: Unused
Compare Mode: Unused
PWM Mode: These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCP1M3:CCP1M0**: CCPx Mode Select bits
0000 = Capture/Compare/PWM off (resets CCP1 module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, set output on match (CCP1IF bit is set)
1001 = Compare mode, clear output on match (CCP1IF bit is set)
1010 = Compare mode, generate software interrupt on match (CCP1IF bit is set, CCP1 pin is unaffected)
1011 = Compare mode, trigger special event (CCP1IF bit is set; CCP1 resets TMR1
11xx = PWM mode

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.2 – MODE CAPTURE

Ce mode permet de transférer la valeur (16 bits) du timer 1 dans les deux registres 8 bits CCPR1H et CCPR1L, lorsqu'un évènement se produit sur le port RB3 d'entrée.

Le timer 1 doit alors être en mode timer ou compteur synchrone.

RB3 doit bien sûr être configuré en entrée $TRISB_3 = 1$.

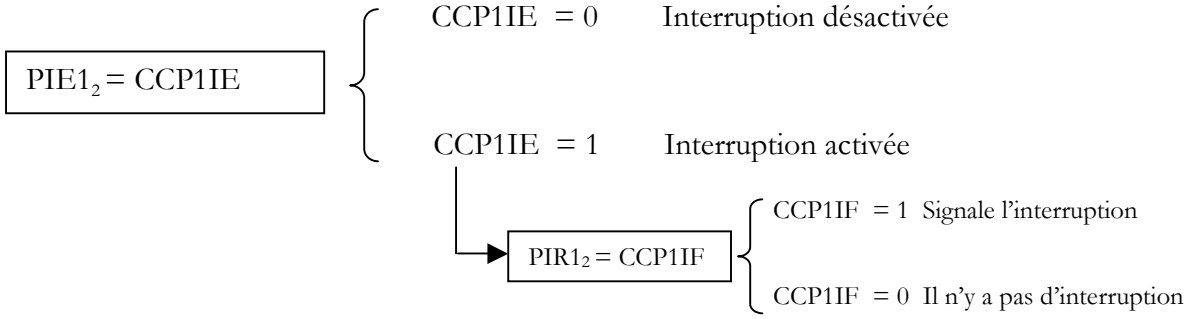
L'évènement déclenchant le transfert est sélectionné par les bits 0 à 3 du registre CCP1CON. (Cf §7.1) :

- Sur chaque front descendant du signal connecté sur RB3
- Sur chaque front montant du signal connecté sur RB3
- Tous les 4 fronts montants du signal connecté sur RB3
- Tous les 16 fronts montants du signal connecté sur RB3

Les bits 0 à 3 du registre CCP1CON permettent donc de régler un pré-diviseur agissant sur le signal appliqué à RB3.

La réalisation de la capture est signalée par la mise à 1 du drapeau correspondant : bit 2 CCP1IF du registre PIR1. Ce bit doit alors être remis à 0 dans le programme.

Le contrôle de l'interruption est réalisé par le bit 2 CCP1IE du registre PIE1.



Attention : le changement du mode de capture entraîne une fausse interruption. Avant toute modification du mode, il faut désactiver l'interruption et remettre à zéro le drapeau CCP1IF.

7.3 – REGISTRES PIR1 ET PIE1

Le registre PIR1 est un registre regroupant les drapeaux d'interruption c'est à dire des bits signalant qu'un évènement déclenchant une interruption s'est produit.

Le registre PIE1 est le registre qui active ou pas les interruptions.

PIR1 REGISTER (ADDRESS: 0Ch)

R/W-0	R/W-0	R-0	R-0	U-0	R/W-0	R/W-0	R/W-0
EEIF	CMIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **EEIF:** EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation has not completed or has not been started

bit 6 **CMIF**: Comparator Interrupt Flag bit
1 = Comparator output has changed
0 = Comparator output has not changed

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **Unimplemented**: Read as '0'

bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture Mode
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare Mode
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM Mode
Unused in this mode

bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIE1 REGISTER (ADDRESS: 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
EEIE	CMIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **EEIE:** EE Write Complete Interrupt Enable Bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 6 **CMIE**: Comparator Interrupt Enable bit
1 = Enables the comparator interrupt
0 = Disables the comparator interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **Unimplemented**: Read as '0'

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.4 – MODE COMPARE

Dans ce mode, le registre 16 bits CCPR1 (CCPR1H et CCPR1L) est constamment comparé au timer 1.

Quand ils sont égaux, cela est signalé sur la sortie RB3 :

- Soit par un état haut
- Soit par un état bas
- Soit par un maintient de la valeur présente.

L'action sur la patte RB3 est choisie par les bits 0 à 3 du registre CCP1CON. (Cf § 7.1).

RB3 doit bien sûr être configuré en sortie $TRISB_3 = 0$.

Le timer 1 doit alors être en mode timer ou compteur synchrone.

Comme dans le mode capture, la réalisation de l'égalité est signalée par la mise à 1 du drapeau correspondant : bit 2 CCP1IF du registre PIR1. Ce bit doit alors être remis à 0 dans le programme.

Le contrôle de cette interruption est réalisé par le bit 2 CCP1IE du registre PIE1.

Si CCP1CON <3 :0> = 1010 alors la réalisation de l'égalité entraîne une interruption logiciel sans affecter RB3.

Si CCP1CON <3 :0> = 1011 alors la réalisation de l'égalité remet à zéro le timer 1. On a ainsi un timer 1 dont la période est programmable par le registre CCP1.

8 – UTILISATION DU TIMER 2

Le timer 2 est un timer 8 bits avec pré et post-diviseurs programmables par l'intermédiaire du registre T2CON.

Il peut être utilisé en lecture et écriture et est remis à zéro par le reset du microcontrôleur.
Il est activé ou désactivé par le bit 2 du registre T2CON.

Il sert de base de temps pour le mode PWM du module CCP : timer 2 s'incrémente de 0x00 jusqu'à ce qu'il atteigne la valeur du registre PR2 puis repasse à 0x00 lors du cycle suivant. (fonctionnant en lecture et écriture).

Le bit 1 TMR2IF du registre PIR1 signale l'égalité timer 2 = PR2 en passant à 1. Ce drapeau doit alors être remis à zéro dans le programme.

T2CON: TIMER CONTROL REGISTER (ADDRESS: 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale Value
0001 = 1:2 Postscale Value
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = 1:1 Prescaler Value
01 = 1:4 Prescaler Value
1x = 1:16 Prescaler Value

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

9 – MODE PWM DU MODULE CCP

Rappel : PWM signifie Pulse Width Modulation. En Français MLI, modulation de largeur d'impulsion. Il s'agit donc de réaliser un signal dont la largeur de l'impulsion et donc le rapport cyclique est programmable.

Ce mode permet de délivrer sur la patte RB3 utilisée en sortie un signal PWM grâce à l'utilisation du timer 2.

RB3 doit donc être configuré en sortie $TRISB_3 = 0$.
Le timer 2 doit être activé (registre T2CON).

Dans ce mode, le registre 8 bits CCPR1L auquel s'ajoutent les bits 4 et 5 du registre CCP1CON constitue un mot de 10 bits correspondant à la largeur de l'impulsion :

Largeur de l'impulsion = CCPR1L :CCP1CON<5 :4> . Tosc . (valeur du pré-diviseur du timer 2)

La période du signal PWM dépend d'une valeur devant être écrite dans le registre PR2.
Elle est donnée par la relation :

Période PWM = (PR2 + 1) . 4 T_{osc} . (valeur du pré-diviseur du Timer 2)

La période PWM doit être supérieure à la durée de l'impulsion.

Quand TMR2 = PR2 cela entraîne au cycle suivant :

- La remise à zéro du timer 2
- La mise à 1 de la sortie RB3
- Le rapport cyclique est transmis de CCPR1L à CCPR1H.

Le bit 1 TMR2IF du registre PIR1 signale l'égalité timer 2 = PR2 en passant à 1. Ce drapeau doit alors être remis à zéro dans le programme.