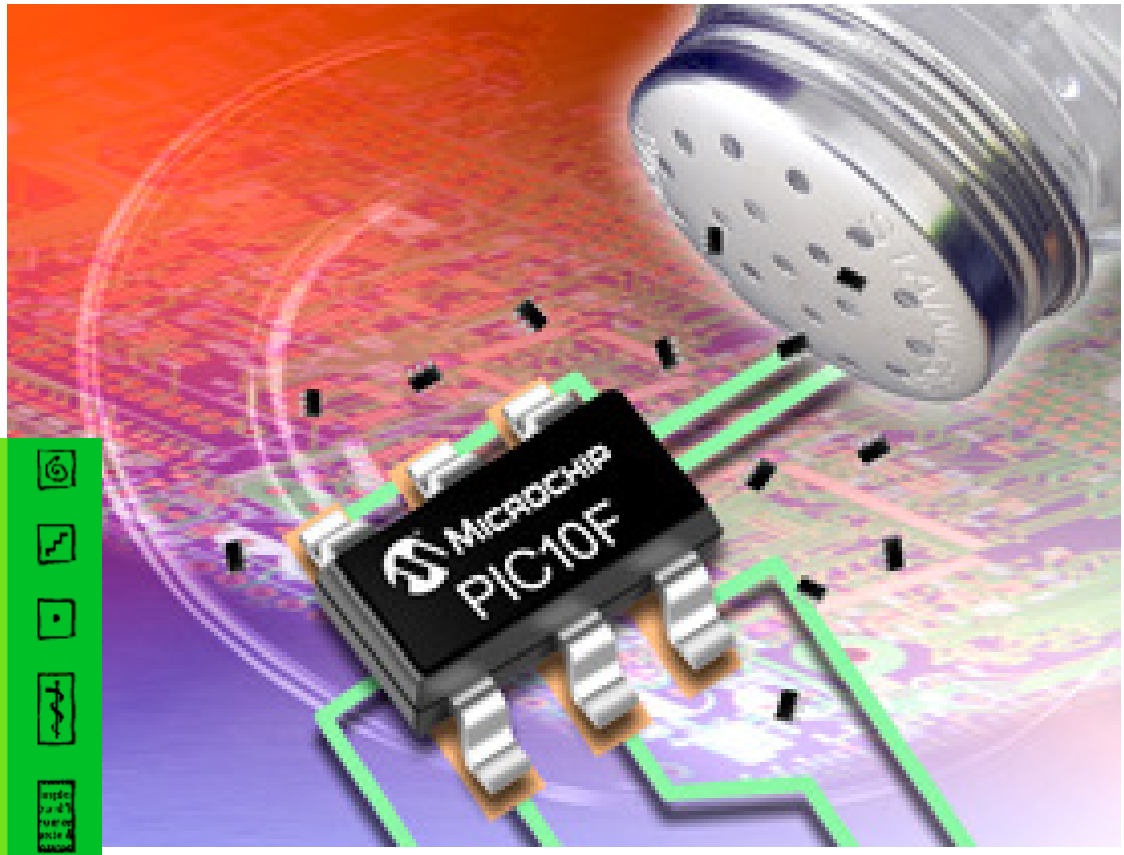
IUT Belfort Montbéliard

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.

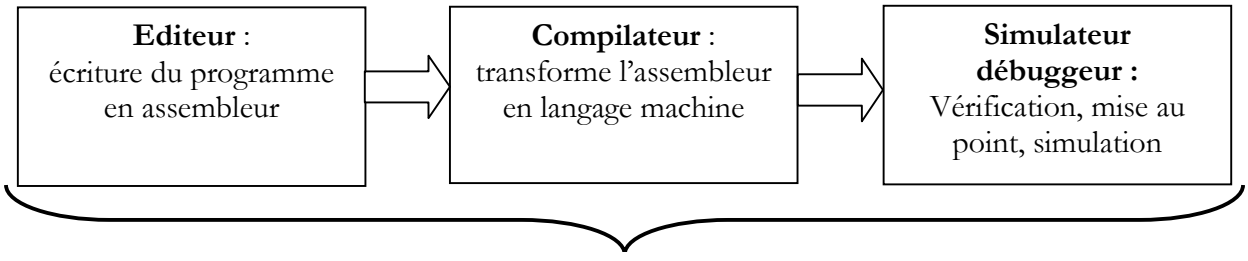
Les domaines d'utilisation principaux sont la robotique, la domotique, l'industrie.

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.

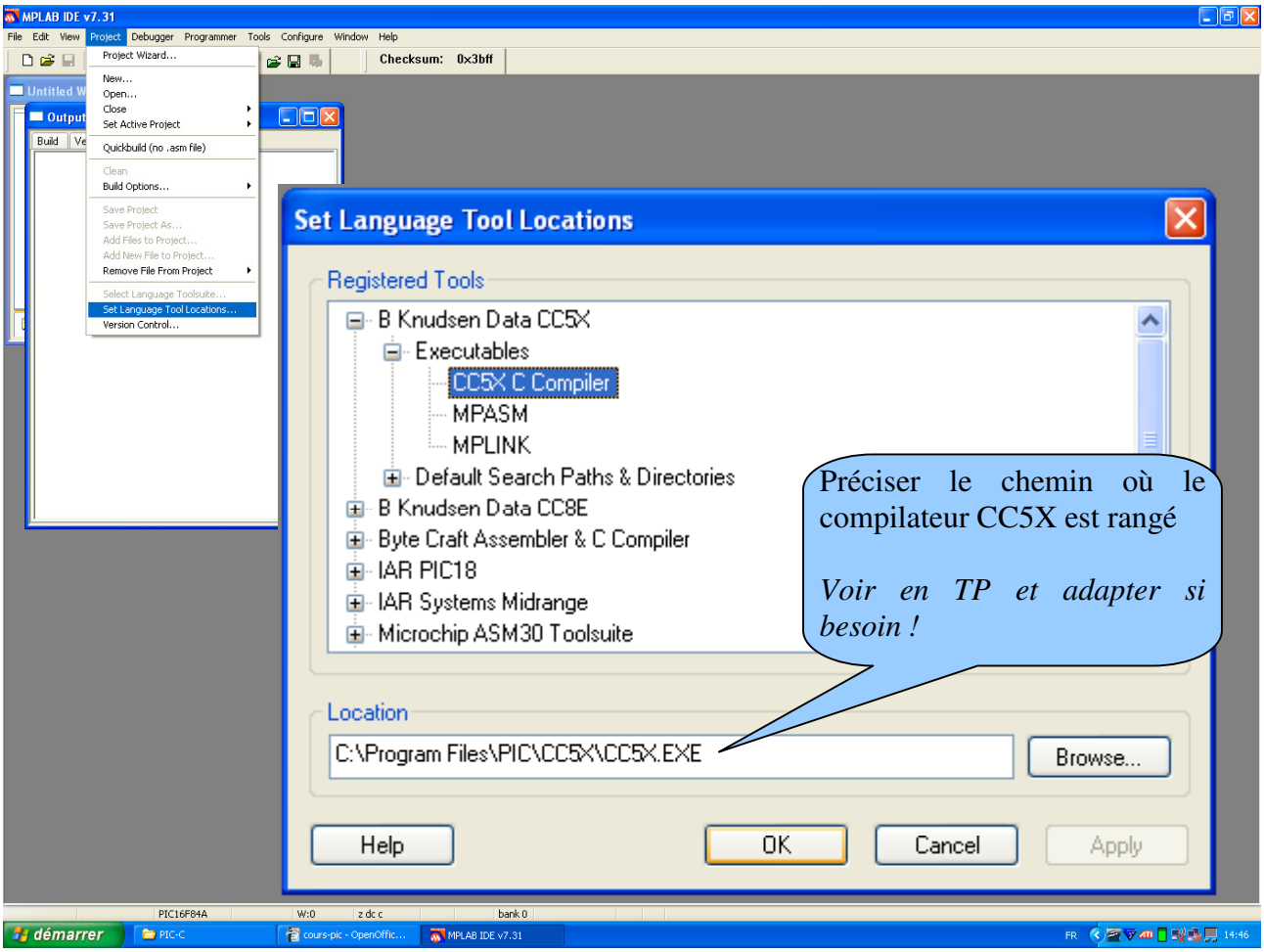
Créer un répertoire MPLAB sur le disque dur de votre ordinateur.
Télécharger MPLAB sur le site www.microchip.com
Décompresser le fichier.

Suivre les indications lors de l'installation.

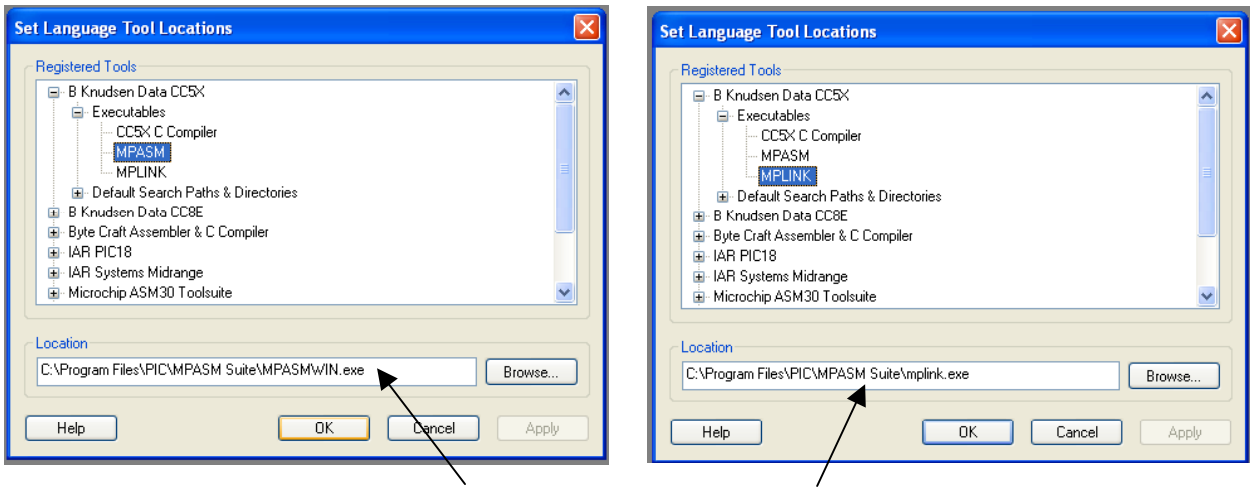
Pour pouvoir utiliser le debugger, il faut ensuite corriger le fichier TLCC5X.INI situé dans le répertoire MPLAB IDE\Core\MTCSuites : Il faut remplacer « Target=HEX » par « Target=COD » et sauvegarder la modification.

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 :



Adapter si besoin les chemins en fonction de l'installation réalisée sur les PC utilisés.

Chemin utilisé :

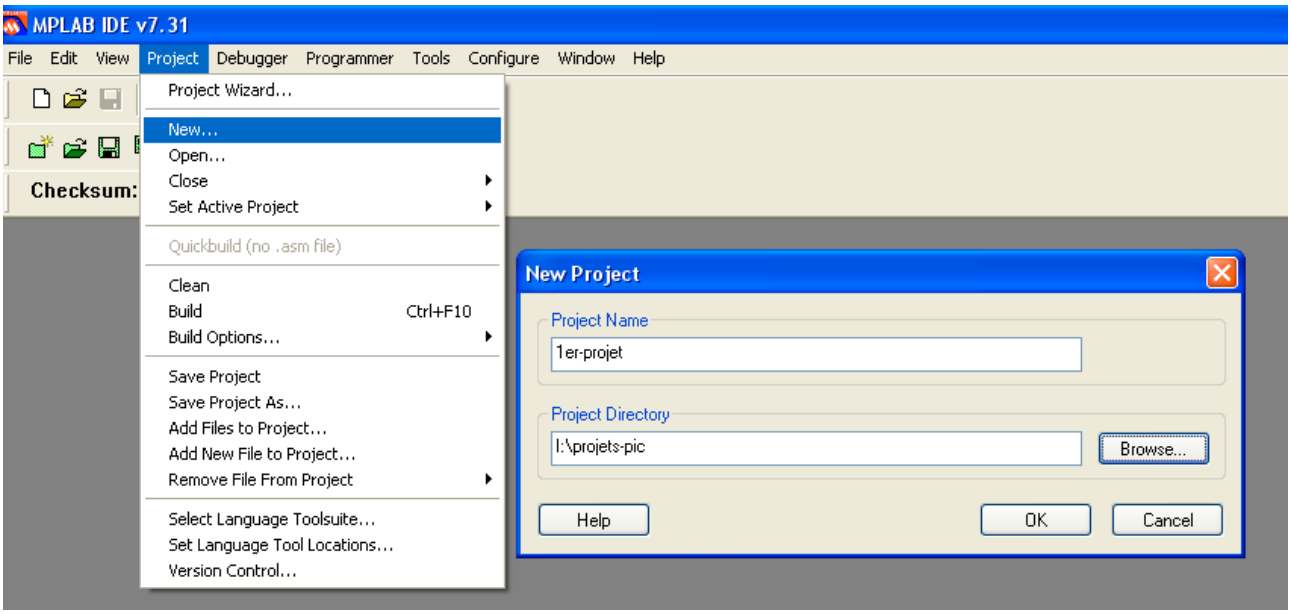
C:\Program Files\Microchip\MPLABIDE\ etc ….

Retrouver ensuite **MPASMWIN.exe** et **mplink.exe** dans l'arborescence.

Chapitre 2 – PREMIER PROJET

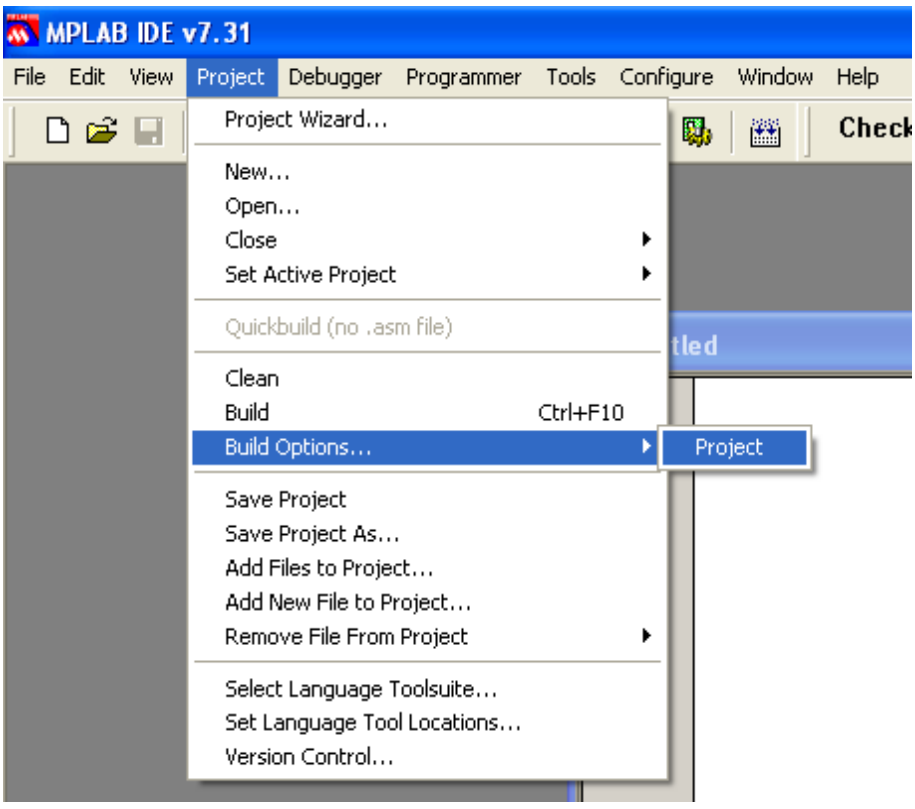
1 – CREATION DU NOUVEAU PROJET

Lancer MPLab. Dans le menu Projet, sélectionner new.

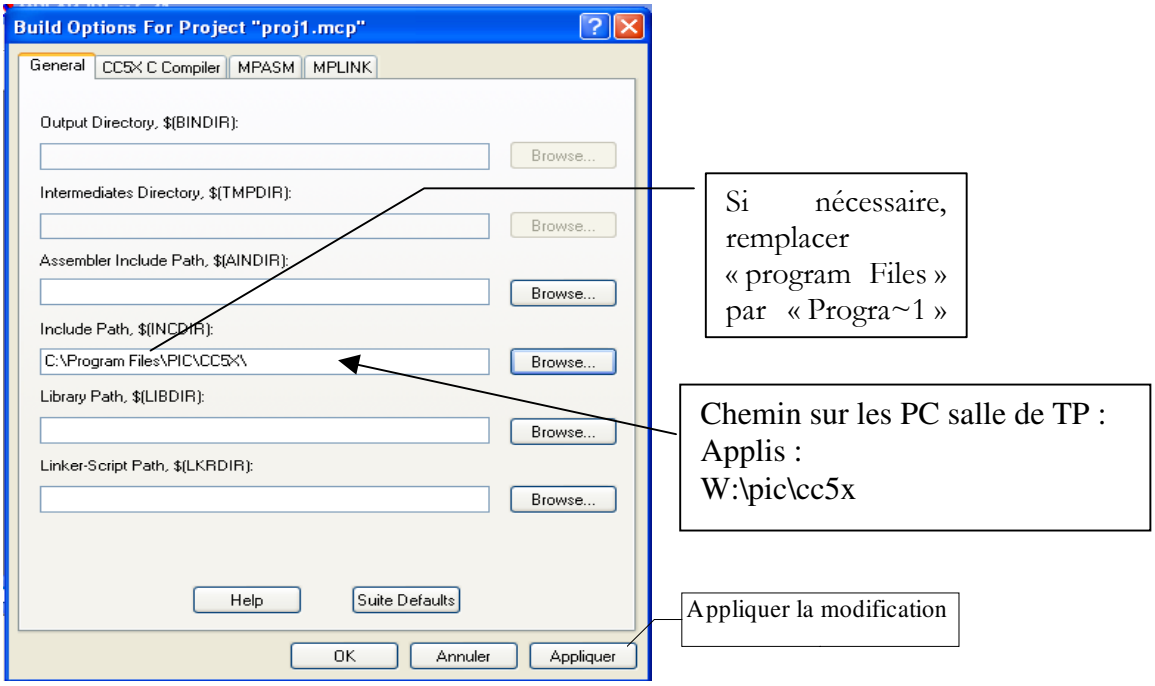


Définir le nom de votre projet et le répertoire pour la sauvegarde.

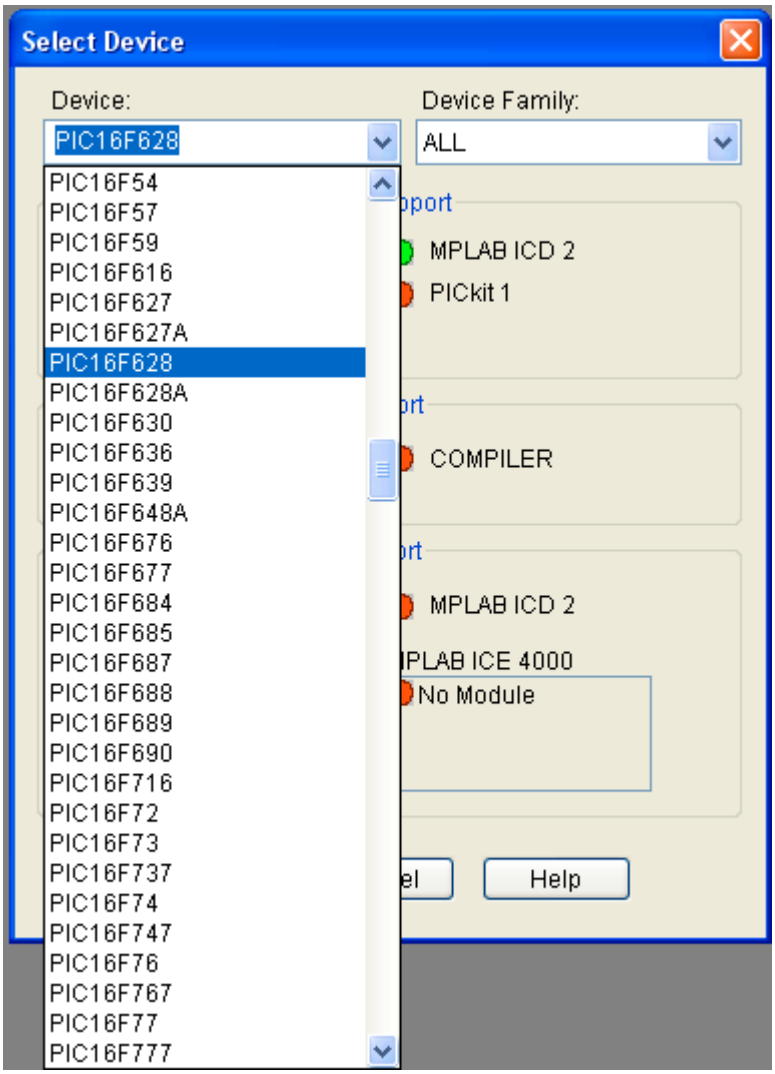
Définir les options :



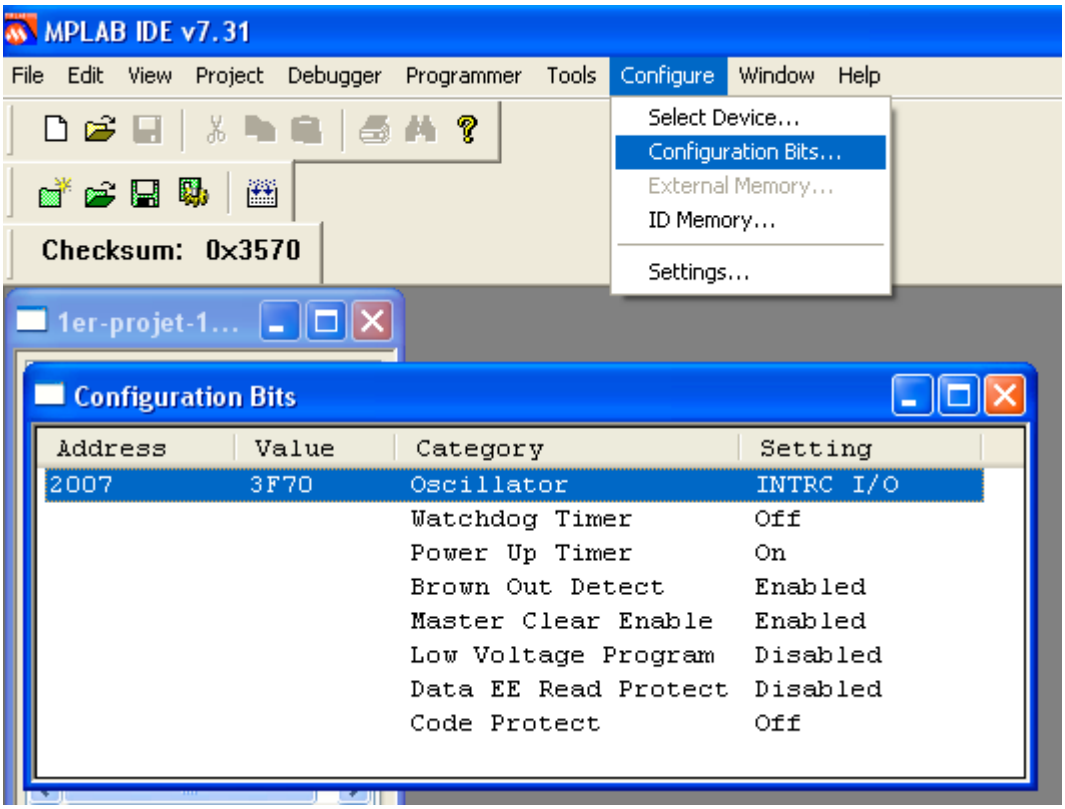
Cela permet de définir le chemin du répertoire d'installation de CC5X.



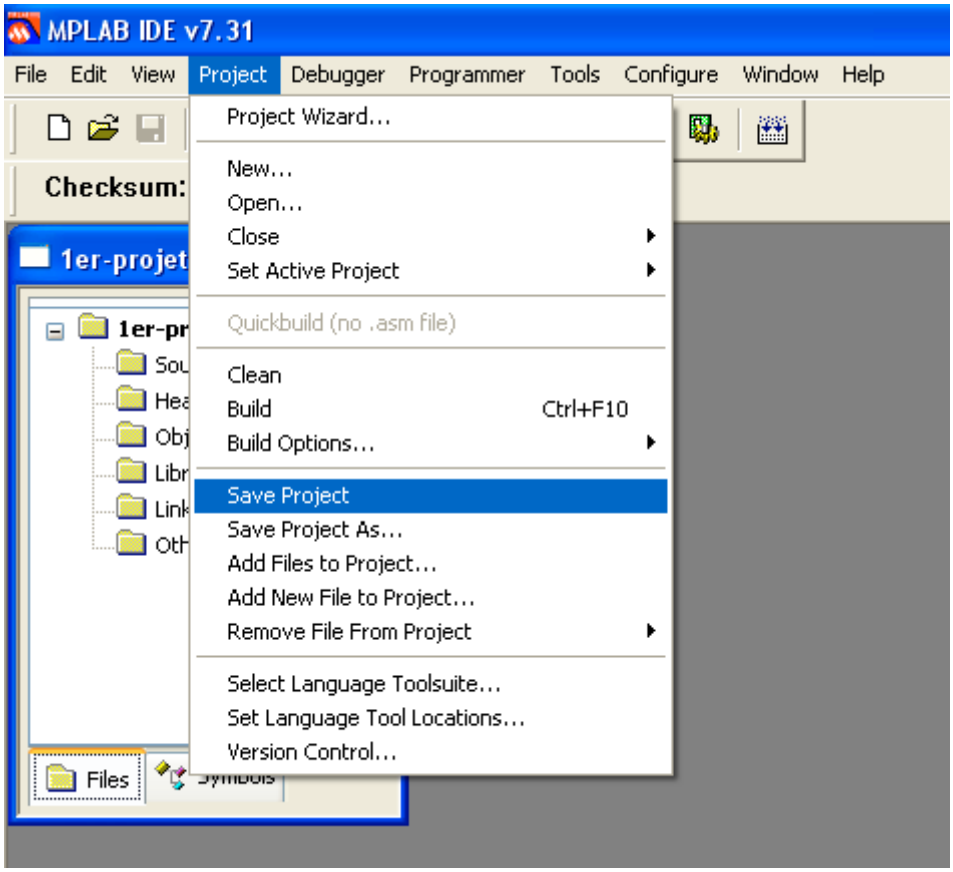
Il convient ensuite de définir le microcontrôleur utilisé : Menu Configure/select device.



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

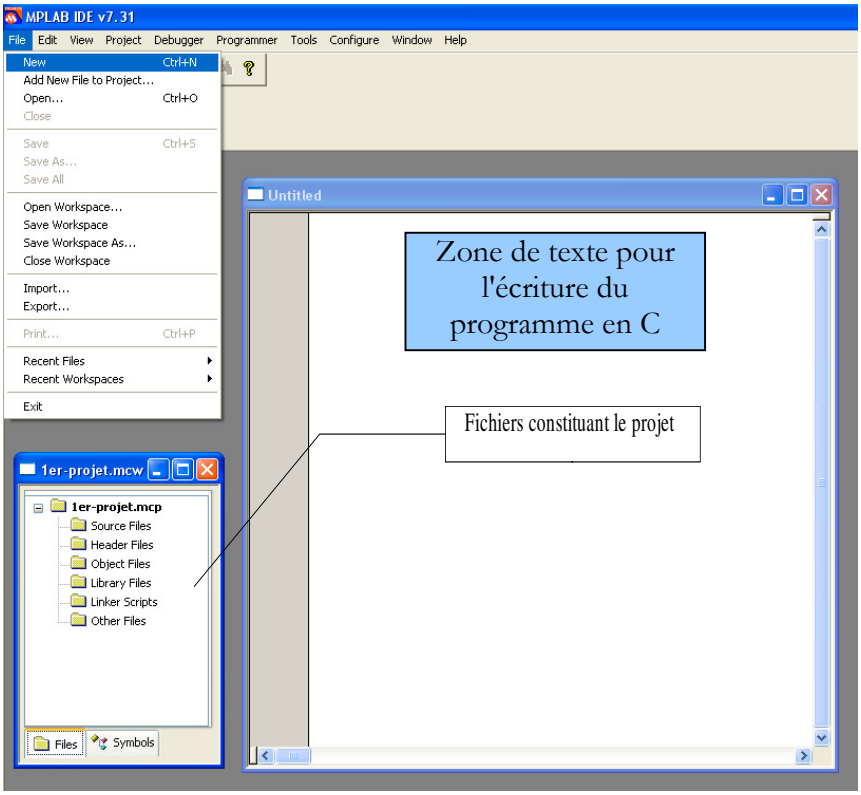


Ne pas oublier de sauvegarder régulièrement les modifications apportées au projet :



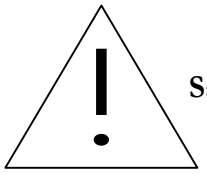
2 – ECRITURE DU PROGRAMME EN C

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



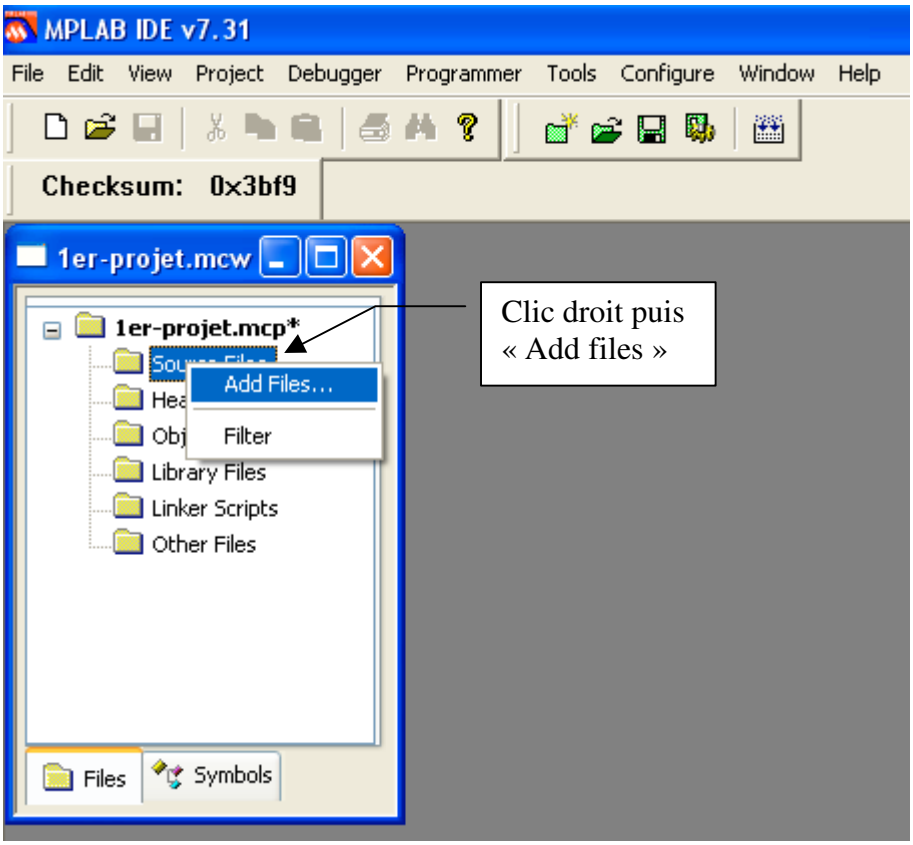
Taper dans la zone de texte, sans pour l'instant chercher à comprendre, le programme suivant :

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



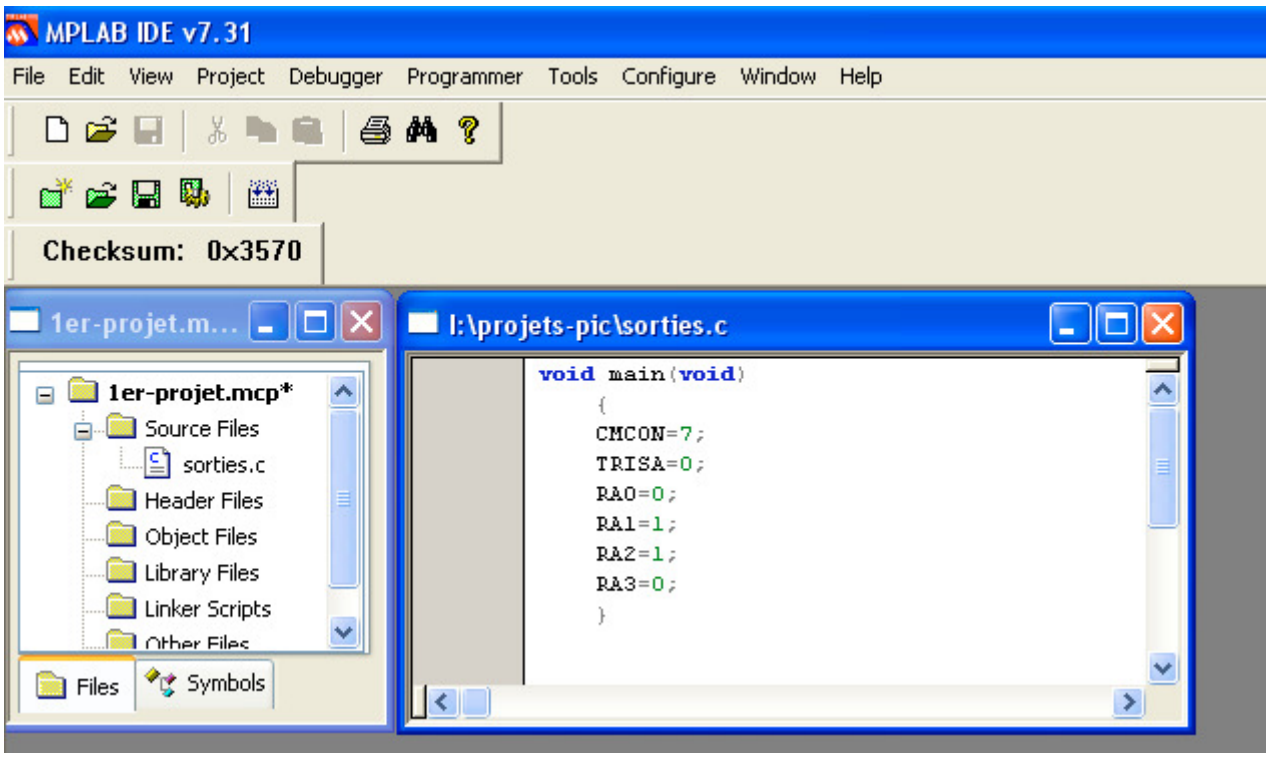
Sauvegarder ensuite le fichier que l'on nommera par exemple sorties.c dans le même répertoire que le projet : Menu File/Save as

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



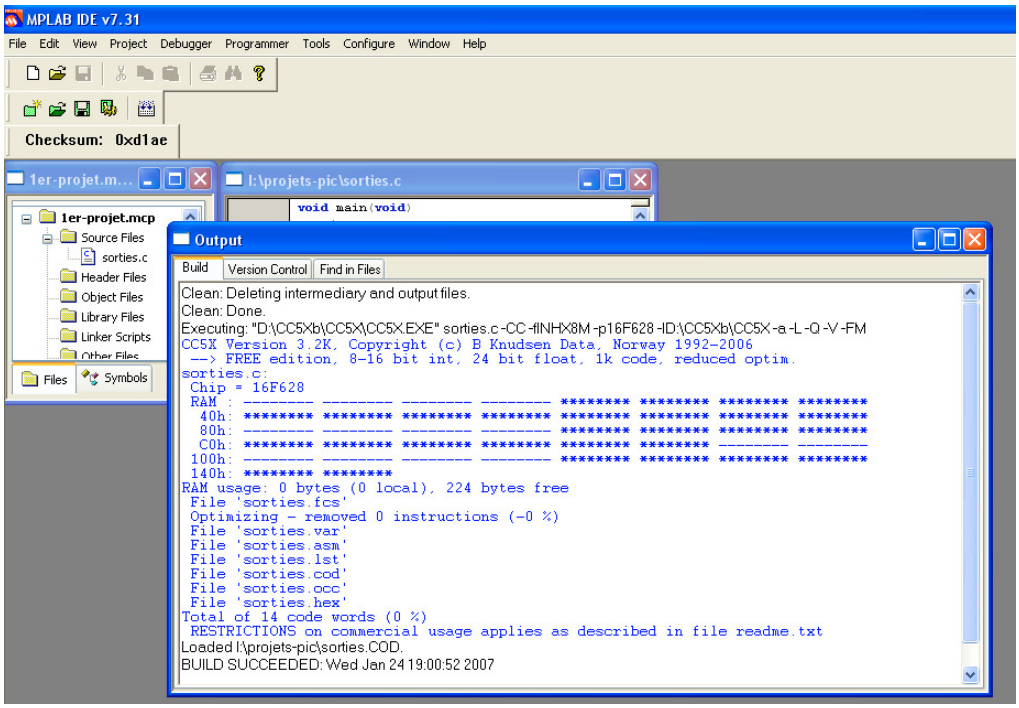
Ouvrir alors le fichier sorties.c que vous venez de créer.
On peut aussi ouvrir un autre fichier.c à condition qu'il soit dans le même répertoire.

Quelle que soit la méthode, nous obtenons :

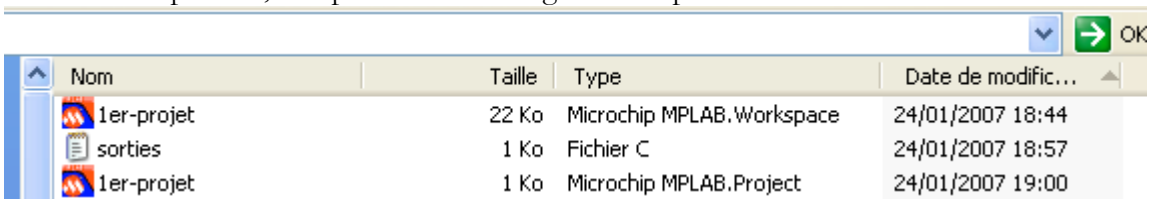


3 – COMPILATION

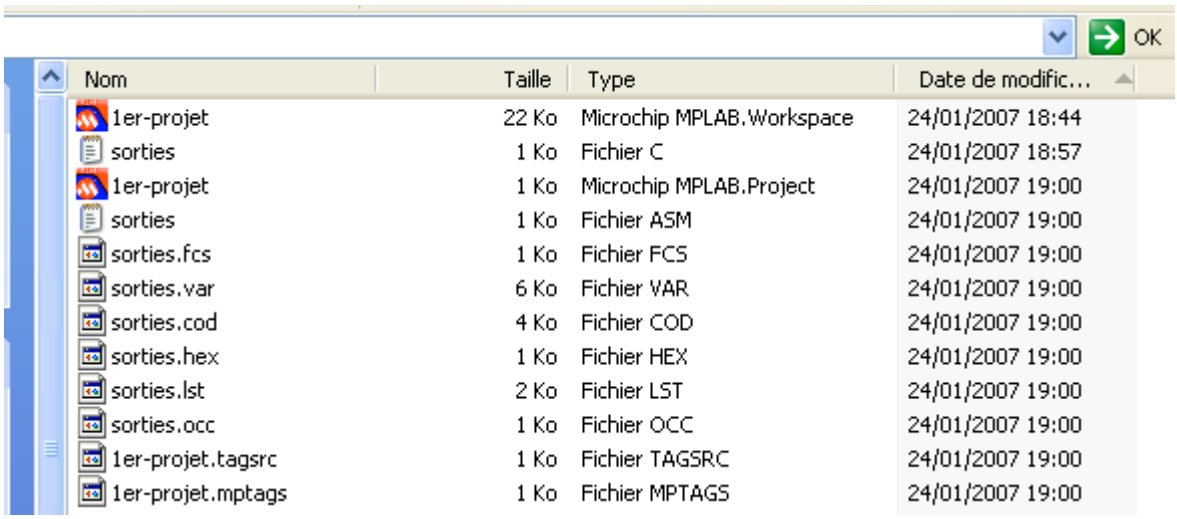
Le projet créé peut maintenant être compilé : Menu Projet/Build



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



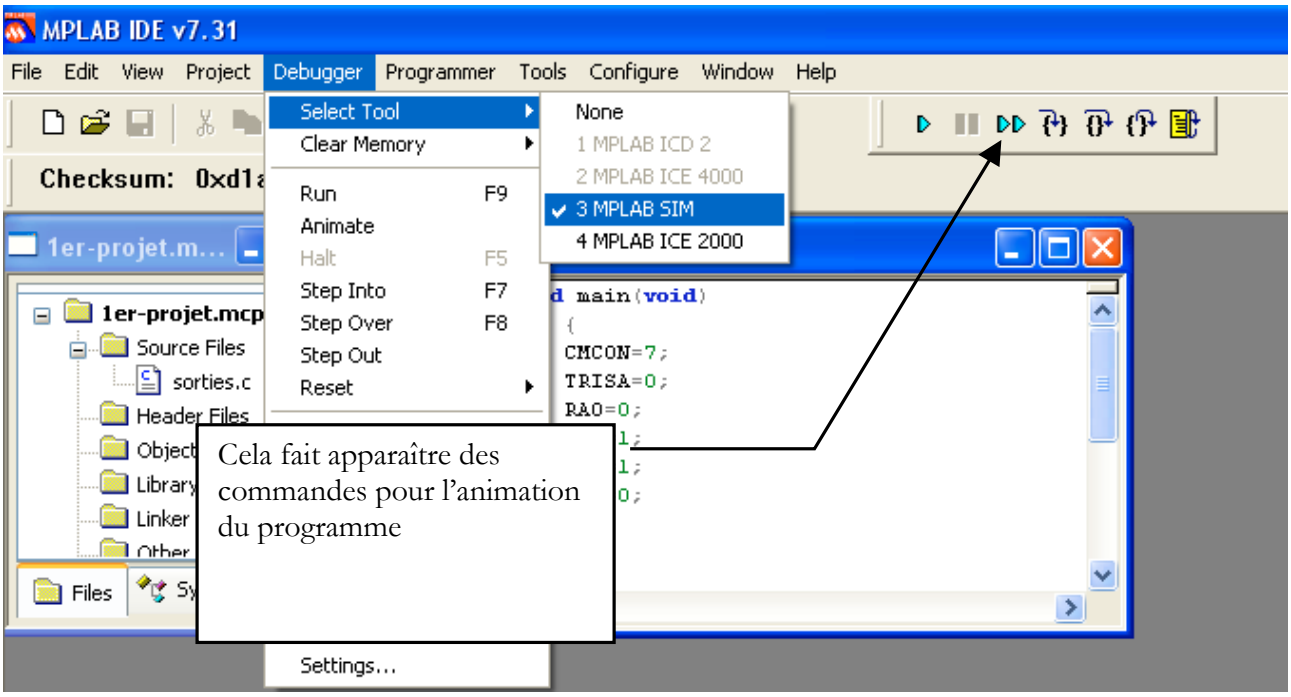
La compilation réalisée à 19h00 ajoute 9 fichiers dans le répertoire de travail :



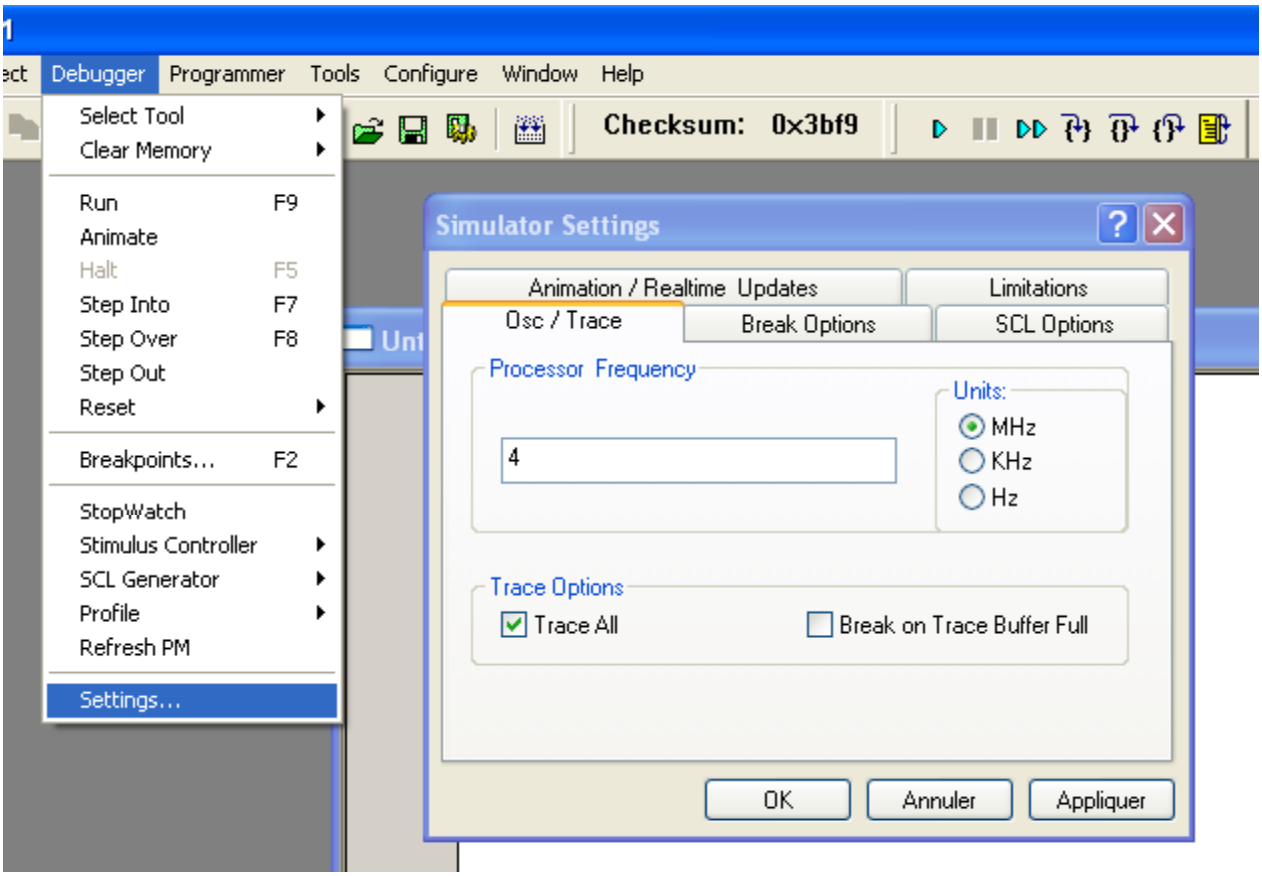
Nous verrons dans le chapitre suivant, le fichier devant être transféré dans le PIC.

4 – SIMULATION

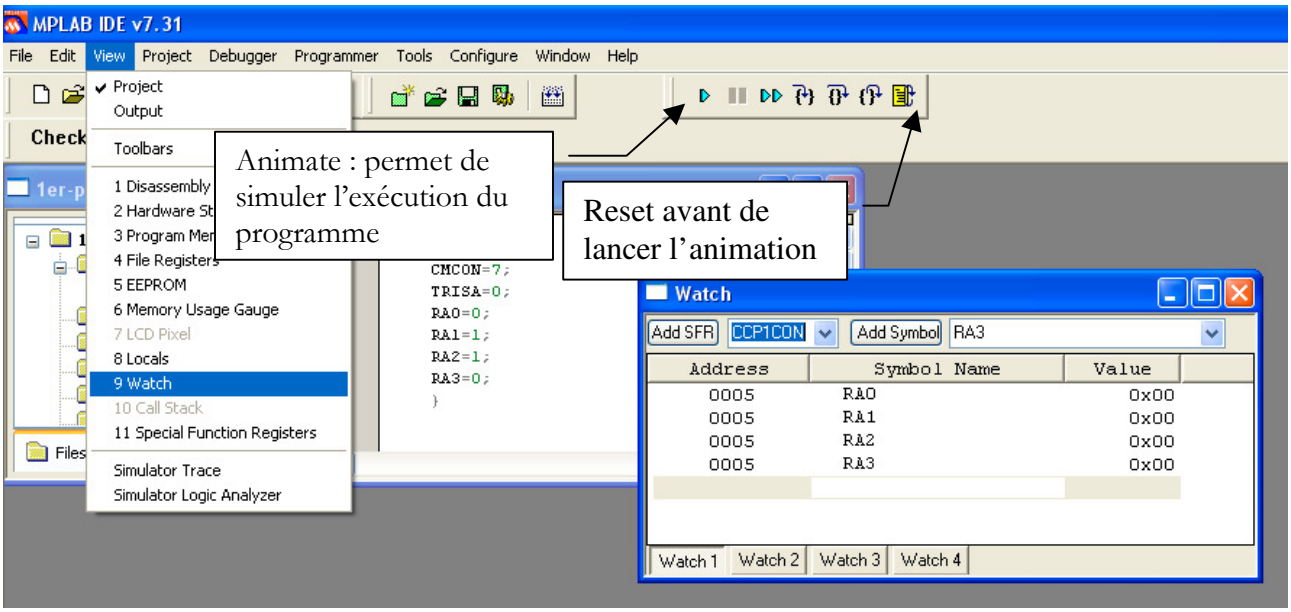
Comme indiqué au chapitre 1, il faut préciser au logiciel que l'outil de mise au point est MPLAB SIM grâce au menu Debbuger, Select Tool :



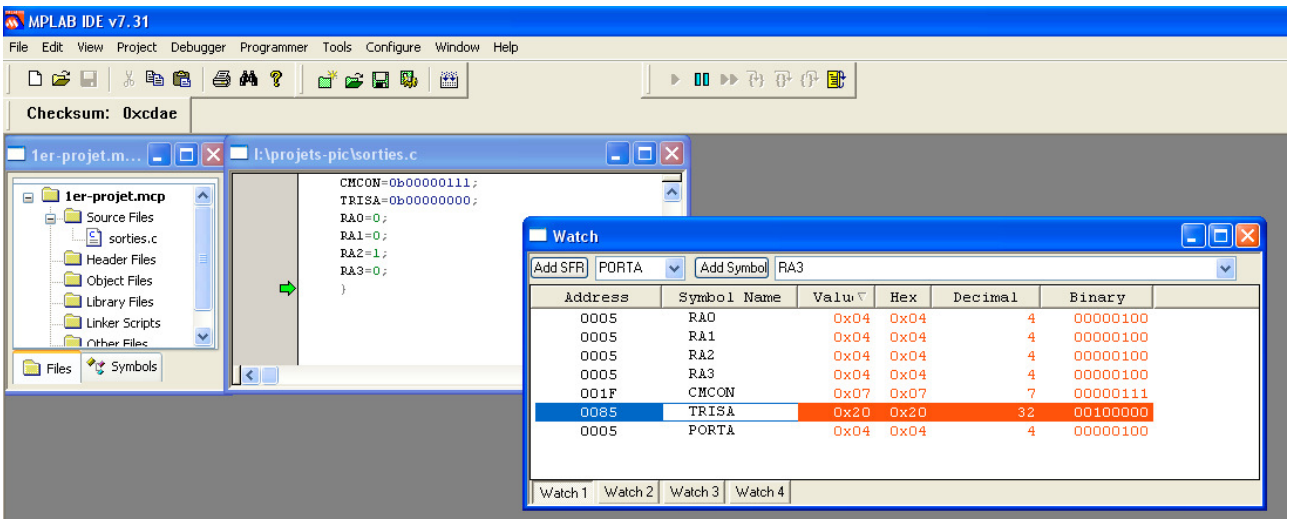
Dans le menu Debugger, de nouvelles sélections apparaissent. Choisir settings pour définir quelques options pour la simulation, en particulier la fréquence de l'horloge dépendant du PIC choisi (4 Mhz pour un 16F628 utilisé avec l'horloge interne (Cf Configuration bits INTRC I/O sur on).



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.

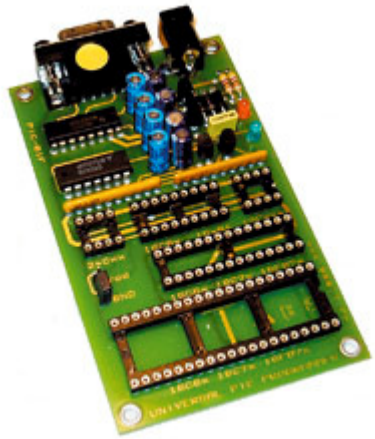
Noter que PORTA et les bits RA0, RA1 etc … affichent en réalité la même information qui est le mot binaire disponible sur le port de sortie, donc de chaque bit RA0 à RA7.

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.

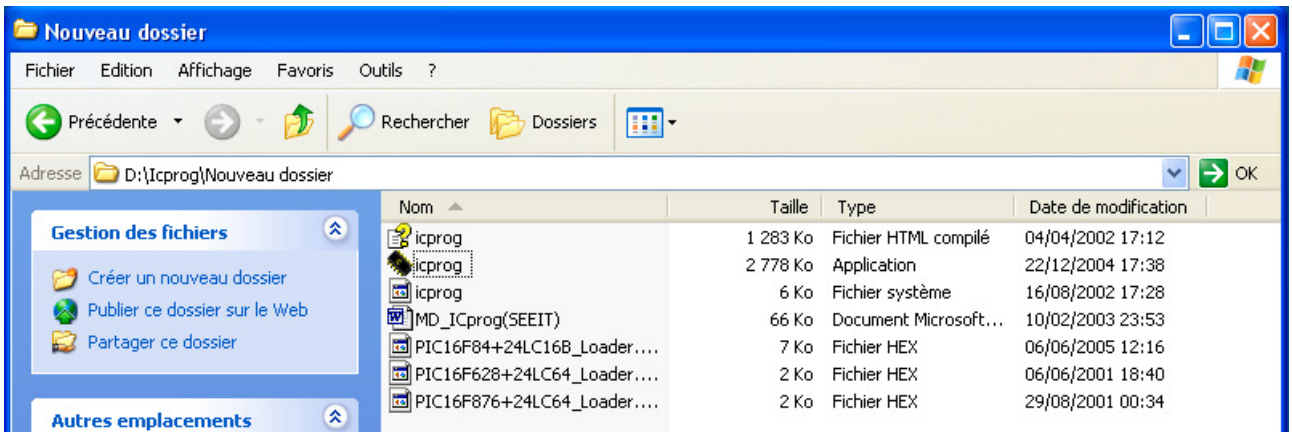
Le PIC utilisé sera placé sur un premier support tulipe, duquel il ne devra pas être ôté, afin d'éviter de tordre puis casser les pattes du microcontrôleur lors des manipulations.

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.

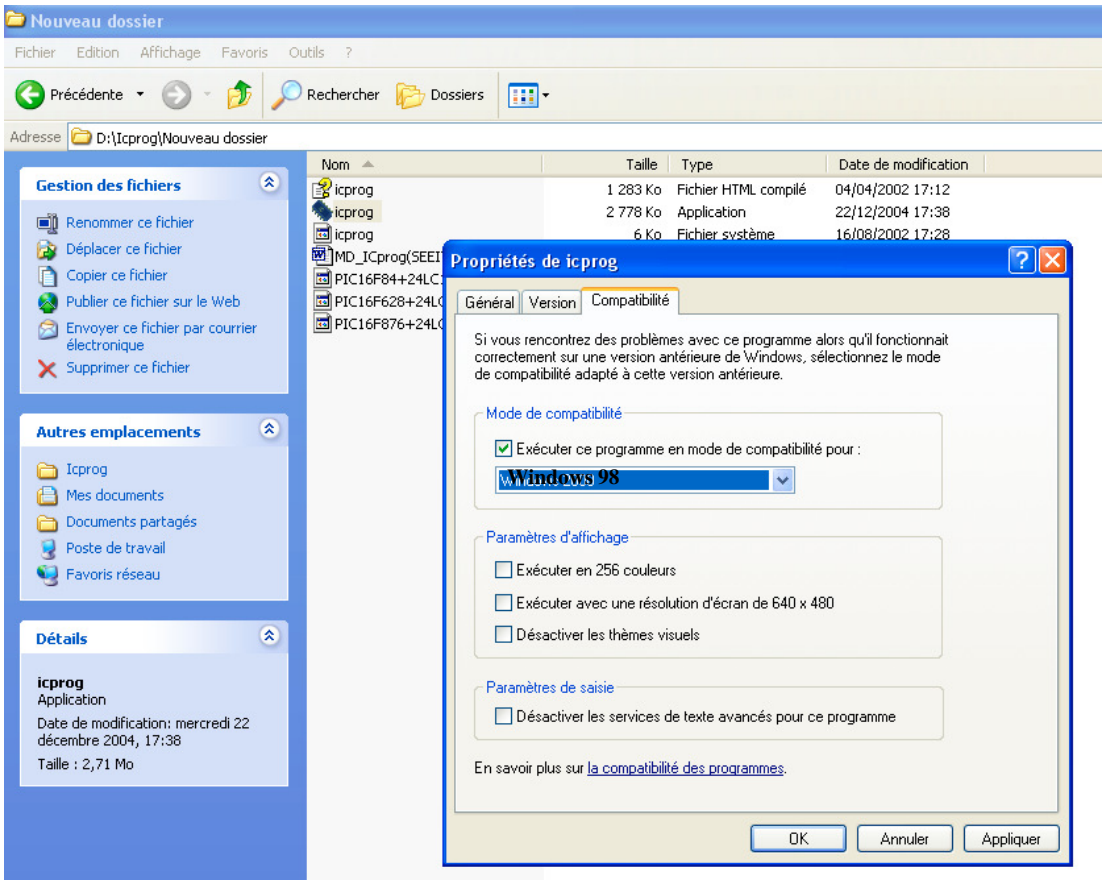


Lancer le logiciel en double cliquant sur l'application icprog.

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 98 » dans le menu déroulant.



3.2 - Configuration\Hardware F3

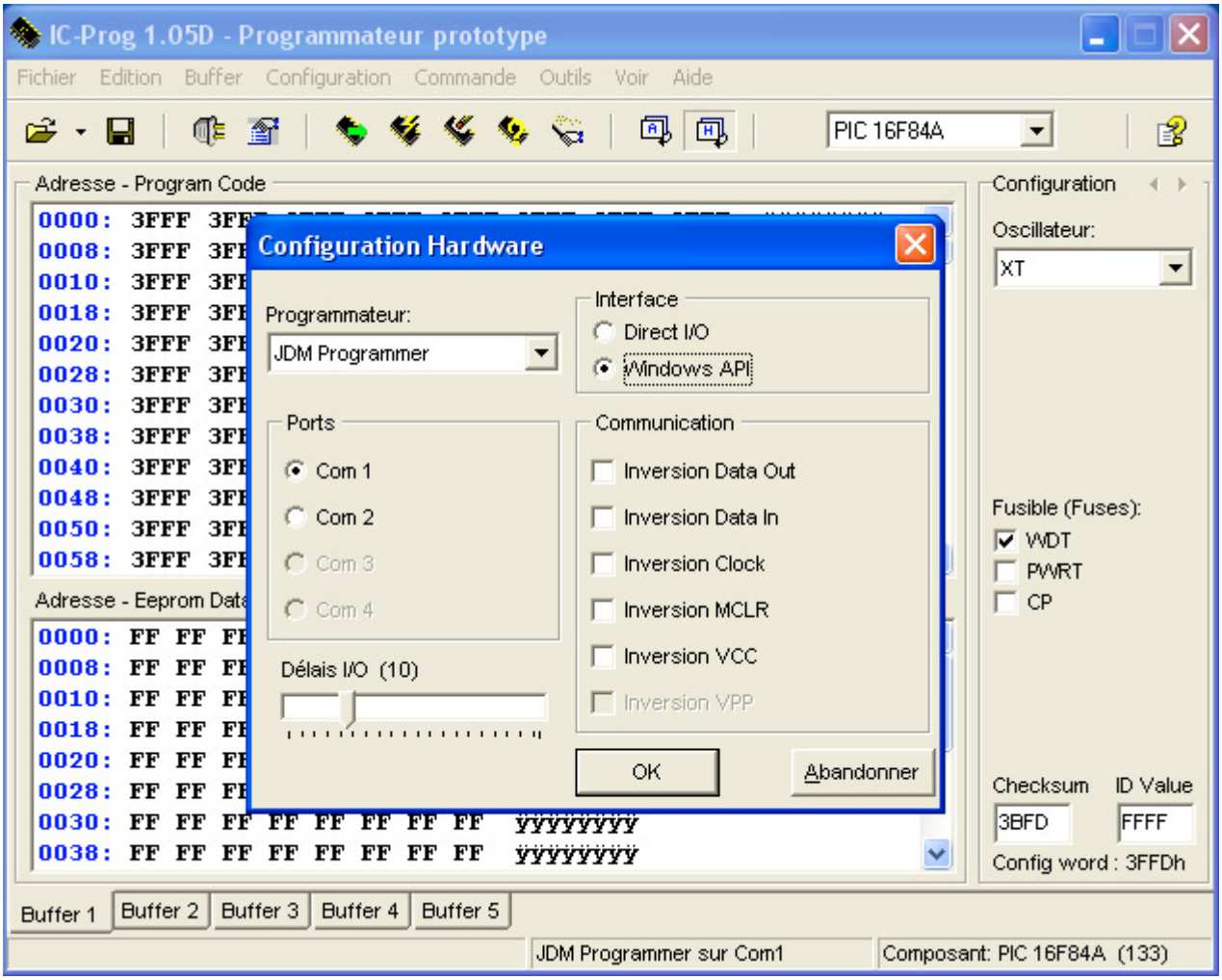
Permet de configurer l'interface de programmation entre le logiciel et la carte de programmation.

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.

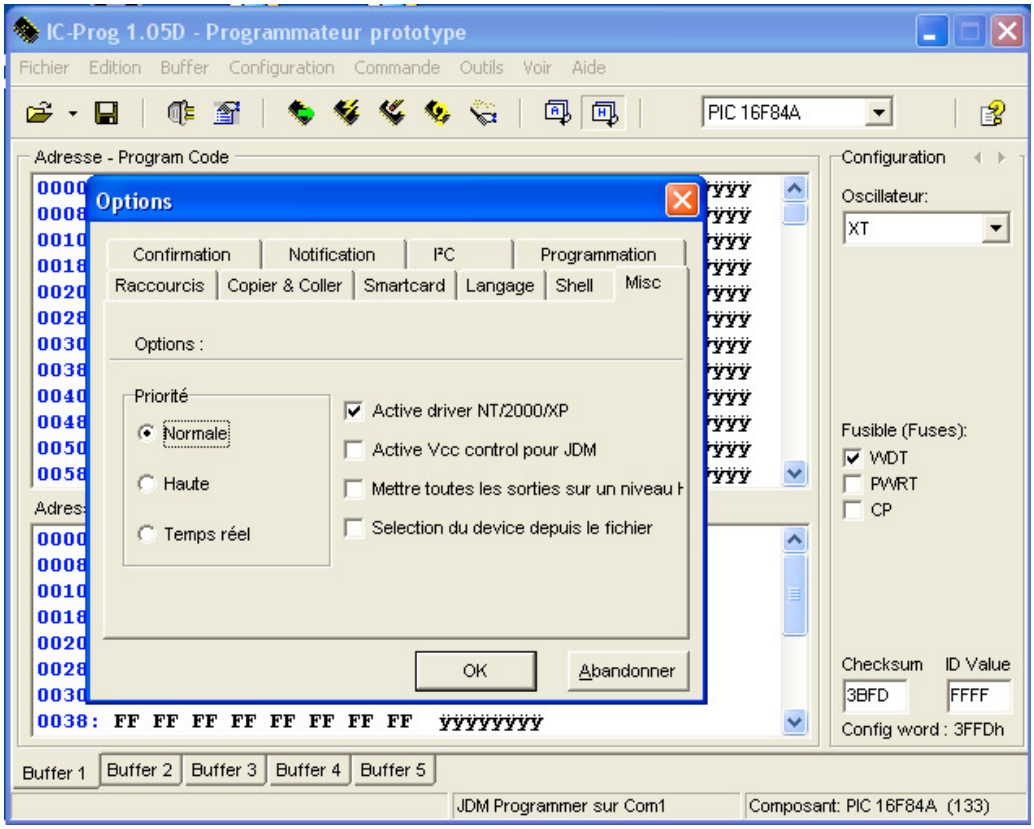
3.3 - Configuration\Options\Misc

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.



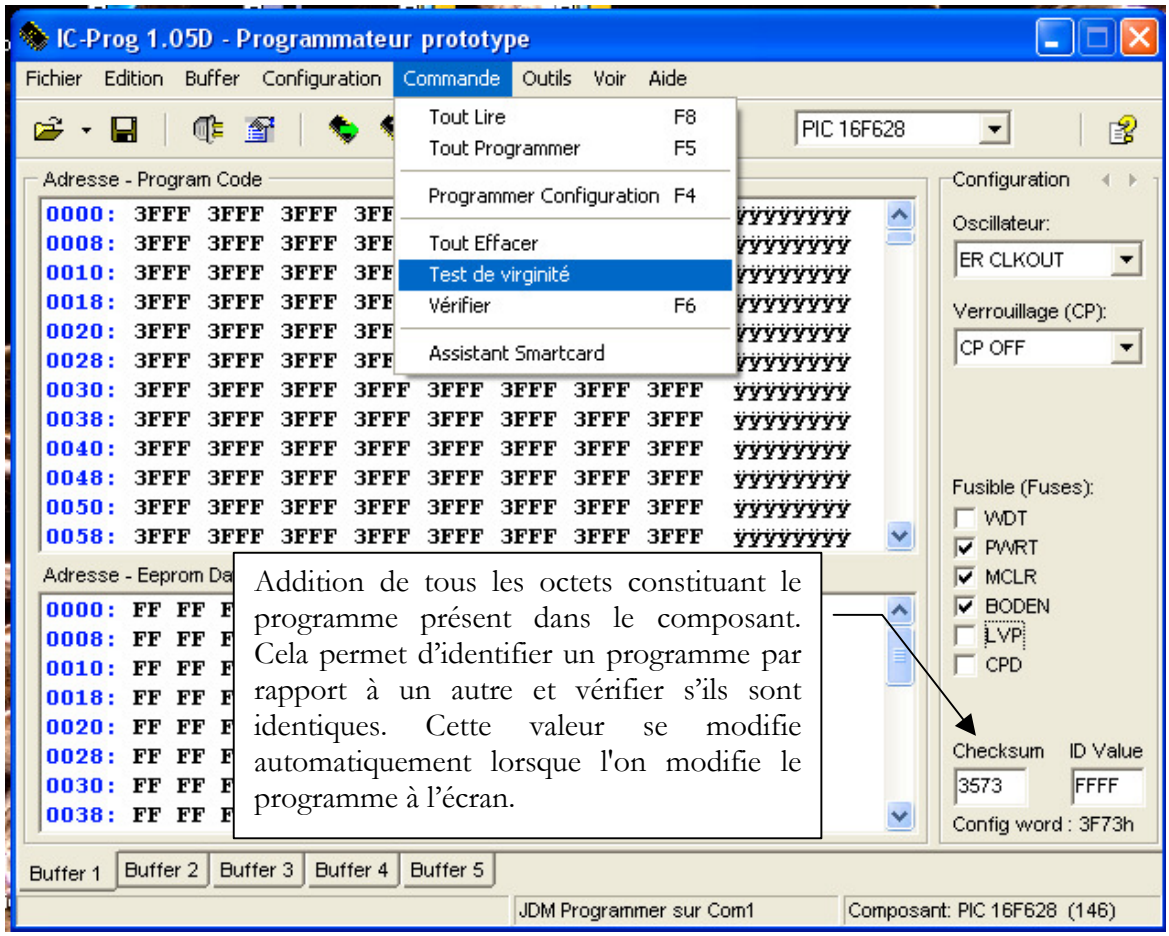
4 – PREMIERE PROGRAMMATION 16F628

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.



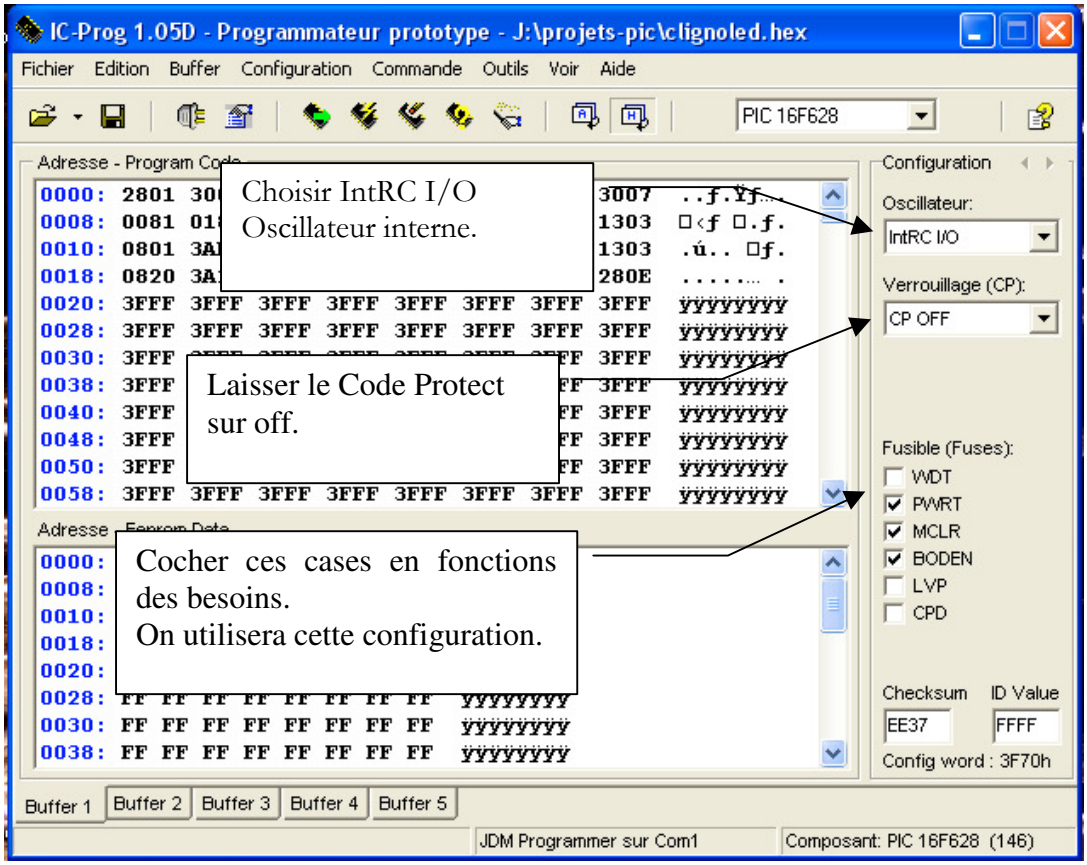
Si le composant est vierge ou effacé tous les bits de la mémoire seront au niveau logique 1 (FF). Cette fonction est à utiliser avant toute programmation car il n'est pas possible de programmer un composant correctement si celui-ci n'est pas vierge ou n'a pas été effacé préalablement.
Si ce n'est pas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

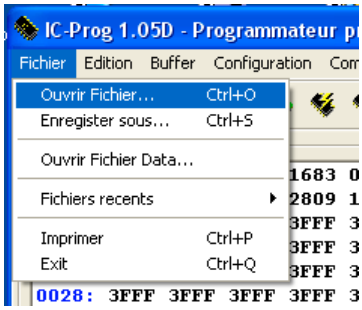
Validation ou non du CP :

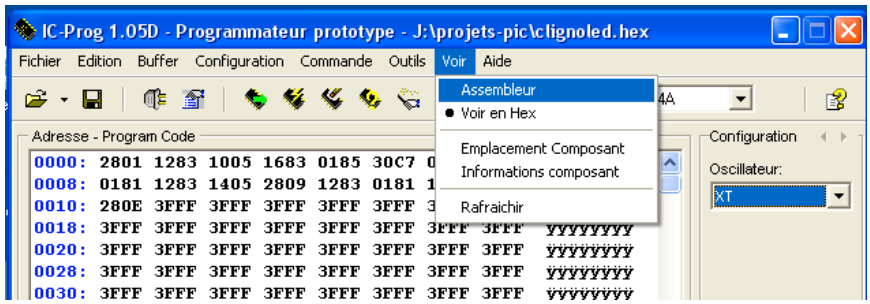
En validant cette case par une croix, le "Code Protect" sera activé. Le programme intégré dans la mémoire du composant ne sera pas lisible si l'on fait une re-lecture de celui-ci. Cependant le composant reste effaçable pour être reprogrammé si celui -ci contient une mémoire Flash.
Attention si vous cochez 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.

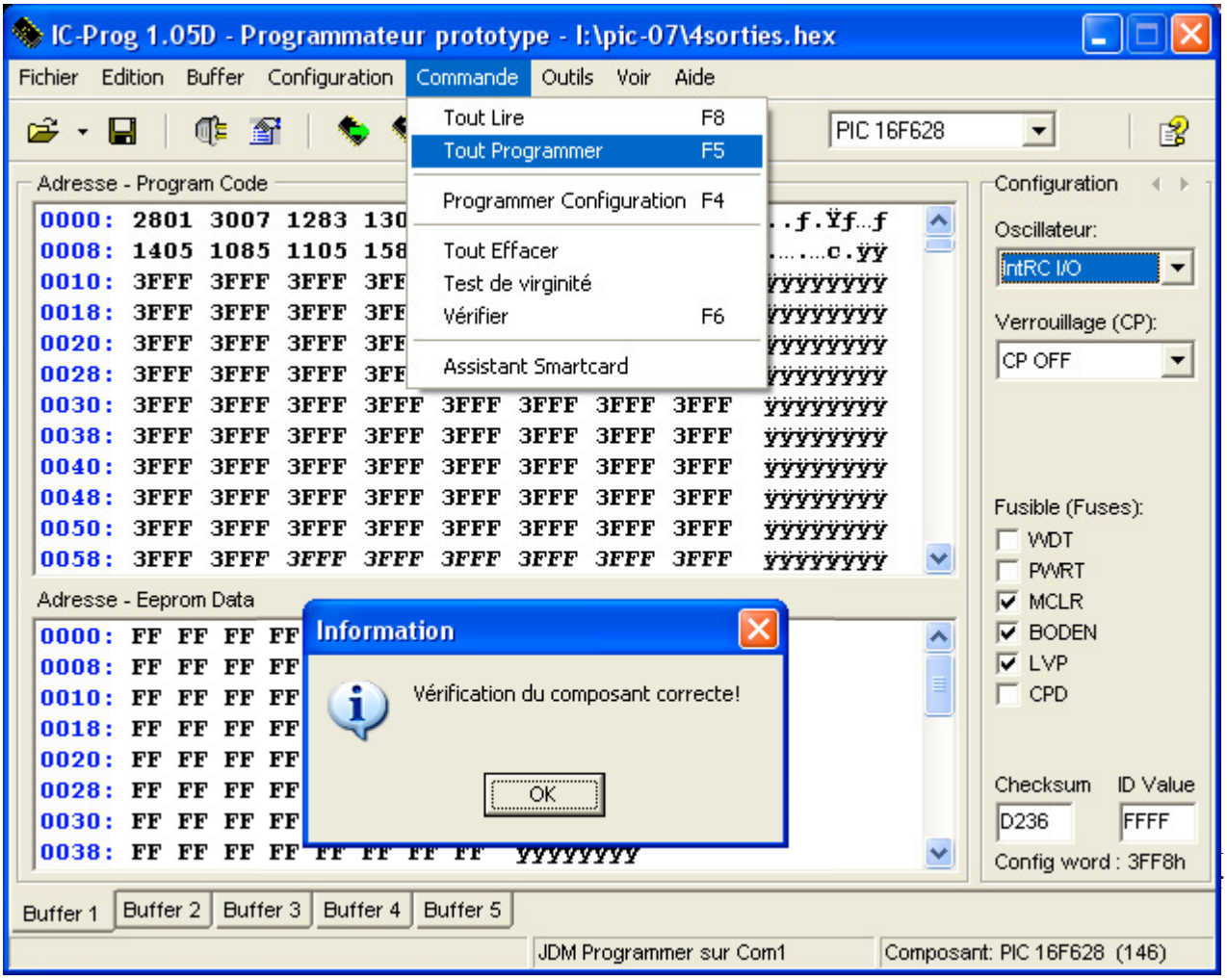




Le programme peut être affiché en hexadécimal ou en assembleur dans la fenêtre Adresse-Program Code.

On constate que le Checksum a changé de valeur.

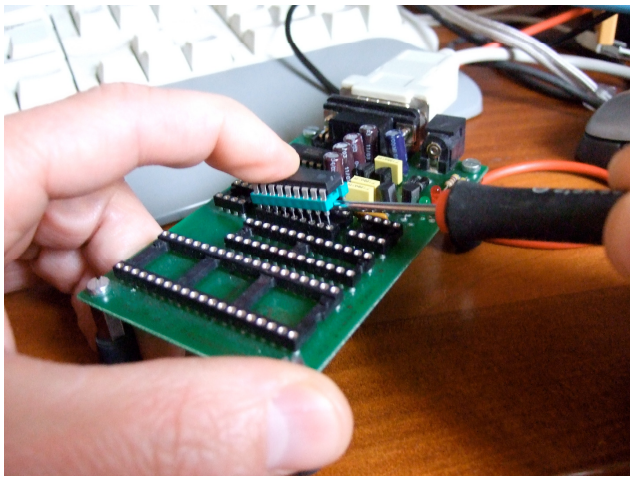
Vérifier que la configuration des fusibles correspond à celle de la compilation du programme dans MP Lab, puis choisir Commande/ Tout programmer.



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'une mauvaise connexion du programmateur (erreur de port série) ou d'une mauvaise alimentation du programmateur.

5 – UTILISATION DU PIC DANS UN MONTAGE

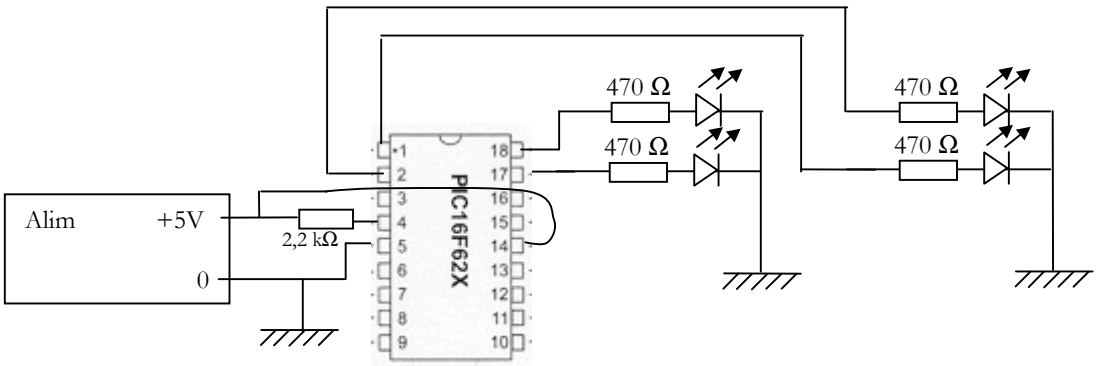
Le microcontrôleur ayant été programmé, il faut maintenant tester le fonctionnement du circuit dans le montage auquel il est destiné.

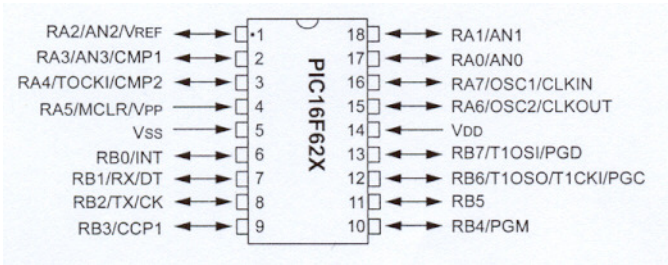


Mettre le programmateur hors tension en coupant l'alimentation stabilisée.
Sortir délicatement le PIC et son premier support de la carte programmateur. Utiliser une pince ou un tournevis glissé entre les deux supports.

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

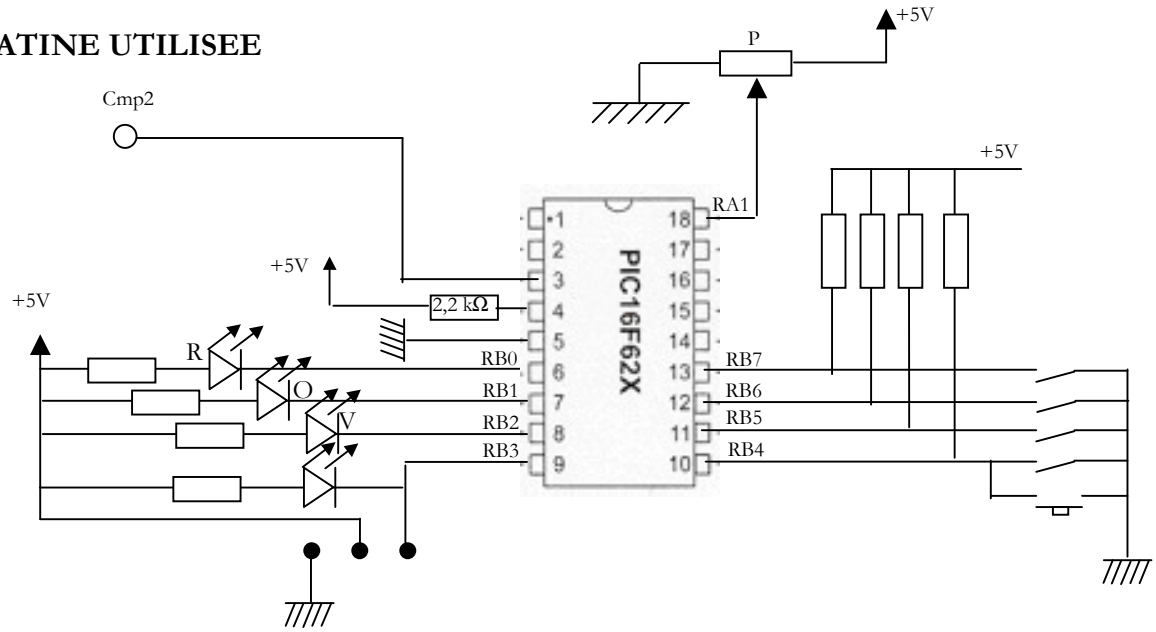
Réaliser le câblage du montage correspondant au programme sur la platine Labdec :





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.

PLATINE UTILISEE



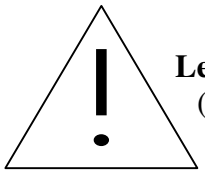
CONSIGNES SAUVEGARDES

N. B : Ne pas enregistrer les fichiers projets ou les codes sources en c dans les ordinateurs en local (sur votre bureau ou dans mes documents) cela a pour effet de ralentir la mise en route des ordinateurs.

Tous les fichiers doivent donc être enregistrés sur le serveur. Pour cela il est conseillé de créer un répertoire pic dans votre espace personnel.

Enregistrer les fichiers sources sous une dénomination type exo1.c. La compilation dans MPLab créera ainsi un fichier exo1.hex qui sera transféré sur le pic à l'aide d'Icprog.

N. B : Toujours utiliser le même projet : cela évite d'écrire les chemins à chaque fois.
Seuls les fichiers source sont modifiés quand on change d'exercice.



Les fichiers .c doivent être enregistrés dans le même répertoire que le projet
(voir page 9 et 10)

CONSIGNES EXERCICES

Pour chaque exercice, bien préciser et justifier la configuration des périphériques. Dessiner les organigrammes. Relever les codes sources.

Exercice 1

Lire le chapitre 1 du livret polycopié puis répondre aux questions suivantes :

1°/ De quels outils logiciels faut-il disposer pour réaliser une application utilisant un PIC.

2°/ Quel est le rôle du compilateur ? Quel est le rôle du programmateur ?

Contrôle :

Exercice 2

Suivre le chapitre 2 pour réaliser un premier projet et utiliser le simulateur et le débogueur de MPLab.
N. B : On ne réalisera pas cette manipulation physiquement (la platine de TP n'est pas conçue pour l'exemple du chapitre)

Contrôle :

Exercice 3

Lire le chapitre 3 afin d'être capable de réaliser la programmation du pic.

Au vu du schéma de la platine de test quel doit être l'état de RB0 pour allumer la led rouge ?

Ecrire un programme en langage c, permettant d'allumer simultanément les leds rouge et verte, la led orange étant éteinte.

Compiler ce programme. Programmer le µcontroleur à l'aide d'Icprog en transférant le fichier hexadécimal dans le pic.
Implanter le pic sur la platine de test et vérifier le bon fonctionnement.

Contrôle :

Exercice 4 : Entrées-sorties sur port A et B

Ecrire un programme qui lit l'état des switchs connectés aux bits RB5, RB6, RB7 du port B utilisés en entrée, et allume la led située en vis à vis sur la platine si le switch est à l'état haut.

Compiler, programmer le pic et tester le fonctionnement.

Contrôle :

Exercice 5 : Timer 0

Ecrire un programme qui fait clignoter à la fréquence de 4 Hz, la led rouge connectée sur la sortie RB0 du port B.

Compiler, programmer le pic et tester le fonctionnement.

Contrôle :

Exercice 6 : Chenillard à 3 leds

Ecrire le programme qui permet d'allumer successivement chaque led rouge, orange puis verte de la platine de test (l'allumage de la led suivante éteint la précédente, chaque led sera allumée pendant 0,5 s). Le programme tournera indéfiniment.

Compiler, programmer le pic et tester le fonctionnement.

Contrôle :

Exercice 7 : Comparateur

Ecrire un programme délivrant sur la sortie RB0, le résultat de la comparaison de la tension Vin appliquée sur RA1 et de la tension de référence interne réglée à 1,25 V.
Compiler, programmer le pic et tester le fonctionnement par action sur le potentiomètre P de la platine de test.

Contrôle :

Exercice 8 : Testeur de batterie

Le potentiomètre P, en réglant la tension u appliquée sur l'entrée RA1 du pic, permet de simuler l'état de charge d'une batterie.

Si $3,4 < u < 5$ allumer la led verte
Si $2,5 < u < 3,4$ allumer la led orange,
Si $0 < u < 2,5$ allumer la led rouge.

Ecrire le programme permettant au pic de réaliser la fonction décrite ci-dessus.
Compiler, programmer le pic et tester le fonctionnement.

Contrôle :

Exercice 9 : CAN 4 bits

Ecrire un programme permettant d'utiliser un pic pour réaliser un CAN 4 bits ayant les caractéristiques suivantes : Ve comprise entre 0 et 3,125 V quantum de 0,2 V.

Le mot binaire résultat de la conversion sera affiché sur 4 leds connectées sur les sorties RB3 RB2 RB1 RB0 du port B.

Compiler, programmer le pic et tester le fonctionnement.

Contrôle :

Exercice 10 : Attente d'un événement

Ecrire un programme qui **attend** l'appui sur le bouton poussoir pour allumer la led verte.

Modifier ce programme pour qu'un deuxième appui sur le bouton poussoir allume la led orange, puis qu'un troisième appui allume la led rouge.

Contrôle :

Exercice 11 : Boucle d'attente et Timer

Ecrire un programme qui attend l'appui sur le bouton poussoir puis attend son relâchement pour allumer la led verte pendant 2 s.

Contrôle :

Exercice 12 : PWM

Générer sur la sortie RB3 un signal ayant les caractéristiques suivantes :

- *Période 20 ms.*
- *Durée état haut : 1,5 ms.*

Contrôle :

Exercice 13 : PWM

Générer sur la sortie RB3 un signal ayant les caractéristiques suivantes :

- *Période 20 ms.*
- *Durée état haut : 1,5 ms*

Après appui sur le bouton poussoir, la durée de l'état haut passera à 0,9 s

Après un 2ème appui sur le bouton poussoir, la durée de l'état haut passera à 2 ms.

Contrôle :

Exercice 14 : PWM et timer

Générer sur la sortie RB3 un signal ayant au départ les caractéristiques suivantes :

- *Période 20 ms.*
- *Durée état haut : 1 ms*

La durée de l'état haut augmente chaque ½ seconde de 0,1 ms jusqu'à atteindre 2ms, puis recommencera ainsi indéfiniment.

Contrôle :