Programmation des microcontrôleurs en BASIC

I) Pourquoi utiliser le BASIC?

Pour programmer les microcontrôleurs on peut utiliser les langages suivants: assembleur, C, pascal, BASIC et les logiciels qui dérivent d'organigrammes comme Flowcode.

L'assembleur: Permet d'obtenir le code le plus concis, et le mieux adapté. Mais il demande un apprentissage de directives et d' instructions. Il est le seuls utilisable dans certains cas très rares ou la vitesse est primordiale.

Le C: C'est le langage à la mode. Le code produit est proche de la machine, mais comme pour l'assembleur, il demande des définitions et inclusion de fichiers, qui complique la compréhension microcontrôleurs. C'est le langage des « pros »

Le PASCAL: Bien qu'il existe le mikropascal de chez mikroelectronika ce n'est pas un langage très utilisé par les électroniciens

Le Basic: C'est lui que nous allons utilisé dans le début de notre cours. Son grand avantage est sa simplicité. Cela permet d'étudier les caractéristiques d'un microcontrôleur sans que les difficultés du langage viennent perturber l'apprentissage. Le basic que nous allons utilisé est celui de PIC SIMULATOR IDE, de oshonsoft disponible iciAu prix de 39 euros environ./

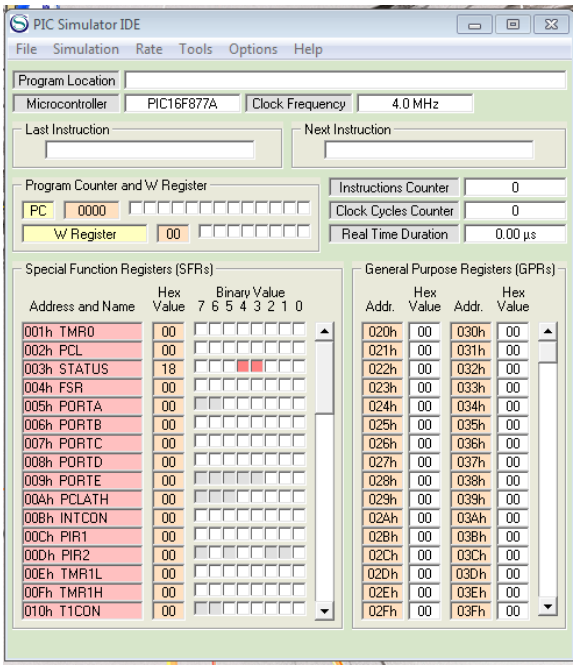
http://www.oshonsoft.com/pic.html

II) Installation de PIC simulator IDE.

Lancez le fichier picsimulatouidesetup.exe en tant qu'administrateur. (Si vous avec le zip, il faudra d'abord le décompresser).

Ensuite il faut installer la licence pour avoir une version complète. (Licence que vous aura envoyé oshonsoft par email.)

Une fois le tous installé lancez le programme. Vos devez obtenir ceci:



1) Il faut tout d'abord sélectionner le microcontrôleur avec lequel vous devez travailler: (Dans certains micro-contrôleurs ont aura plus de timers, de mémoires que d'autres etc…) Il faut donc dire au programme quel est notre microcontrôleur.

Pour cela dans la barre de menu de PIC simulator ide, cliquez sur Options puis sur select Microcontroller. Choisissez dans la liste celui qui vous convient (PIC 16F84 pour notre premier essai) puis cliquez sur le bouton nommé select.

2) Il faudra ensuite indiqué à quelle fréquence on voudra faire travailler le 16F84. Par exemple 4 MHz. Pour cela cliquez sur Options puis sur Change clock Frequency. (Si vous voulez que des LED restent allumées 1s, cette durée va dépendre du Quartz de la base de temps (horloge). Il faut donc que pic simulator sache quel quartz vous aller mettre dans votre montage.

3) Choix des bits de configuration.

Supposons que vous ayez écrit un super programme, et réalisé un super montage (la plus belle merveille du monde !), vous ne voudriez surement pas que quelqu'un le copie (comme les chinois savent si bien le faire). Il y a dans les pics une zones mémoire $2007 et $2008 où certains bits permettent d'interdire à quiconque de relire votre programme. Ce sont les bits de configurations. Ils permettent bien sur de configurer un grand nombre de choses.

Dans notre cas. Nous choisirons:

Code protection: off —> on ne va pas protéger la lecture de notre pic

Power up timer: enable (en français valide),. Lors du démarrage d'un pic (quand on applique la tension), les tensions ne sont pas tout de suite stables. Avec cette option le PIC attendra 72mS pour que les tensions soient stables, et lancera v otre programme.

Watch dog timer: (chien de garde) Votre microcontrôleur peut planter (faire n'importe quoi, à la suite d'un éclair par exemple), le watchdog permettra à des intervalles de temps réguliers, de voir si le programme suit bien son cours normalement. Pour l' »instant nous n'utiliseront pas le chien de garde.

Oscillator sélection: L'horloge de votre microcontroleur (qui définit la durée d'exécution d'une instruction) peut être construite de diverses façons. Réseau RC, quartz, interne etc… Il faut l'indiquer dans les bits de confiquration. Ici nous mettrons XT (pour un quartz).

On peut voir afficher en bas à gauche la valeur $3FF1 qui sera inscrite en 2007 et N/A en $2008 (N/A abréviation de not available = non valide car le 16f84 n'a pas besoin de ces bits là)

Il ne reste plus qu'a cliquez sur apply puis sur close.

III) Notre premier programme en basic

Pour écrire notre programme: Dans le menu de PIC simulator IDE cliquez sur:

Tools puis sur Basic compiler

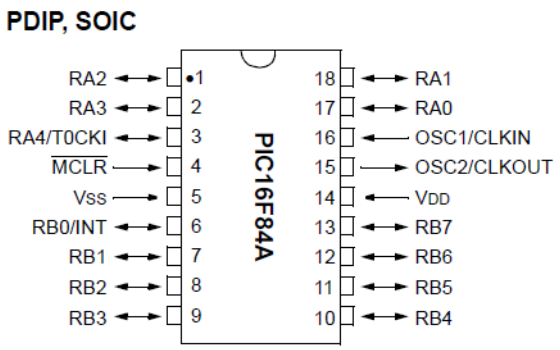
Dans le cadre qui s'affiche tapez le texte suivant:

' une LED qui Clignote.

Comme dans la majorité des langages basic, l'instruction représentée par une apostrophe signifie commentaire (ou remarque). Tous ce qui sur la ligne suit l'apostrophe sera ignoré par le compilateur. Ceci a pour seul but de permettre au programmeur de se souvenir de ce que la partie de programme qui suit va faire. Dans notre cas clignoter une LED. En C les commentaires doivent se trouver entre /* et */. Pic simulator met les commentaires en vert.

Trisa=0 ' port A en sortie

Le port a du 16f84 est formé des pattes 1,2,3,18,17 appellées RA0,RA1,RA2,RA3RA4.



Pour Allumer des LED placées dans le bon sens sur ces pattes, il faut dire au microcontrôleur qu'on les veut en sortie, On dit configurer la ou les pattes ou le port en sortie.

Si on met une patte à 0 en écrivant par exemple trisa.0=0 la patte numéro 17 sera en sortie, mais s'il on écrit trisa.0=1 la patte 17 sera en entrée.

En faisant trisa=0 (cet à dire %0000), c'est toutes les pattes du port a qui seront en sortie. Le commentaire indique ce que l'on a voulu faire.

jesuisduref11: un texte suivi de deux point s'appelle une étiquette. Cela sert à marquer un endroit dans le programme (emplacement mémoire). Ainsi si l'on veut que le programme retourne à cette endroit, on pourra le faire.

Porta=0 'met 0V sur toutes les broches du port A qui sont configurées en sortie.

Une fois qu'une patte à été configurée en sortie par l'intermédiaire de trisa, On peut y mettre 0V ou +Vcc.

Par exemple pour mettre la patte 17 (RA0) à 0V on écrit RA0=0 ou à +Vcc en écrivant RA0=1. Pour tous les bits du porta A configurés en sortie: PORTA=31 (%11111)

Avec porta=0, toutes les LED seront éteintes.

Waitms 1000 (l'instruction waitms ordonne au microcontroleur d'attendre 1 seconde sans rien faire) En fait il y a une boucle d'attente, mais on verra cela lorsque nous ferons de l'assembleur. C'est là un des avantages des langages évolués tels que basic et c, de pouvoir utiliser des fonctions toutes prêtes pour des taches délicates à programmer. C'est aussi pourquoi j'ai débuté ce cours par le basic.

Porta=31 Cela met les bits du port a à +vcc et les Led vont s'allumer.

Waitms 1000 attendre encore 1s

Goto jesuisduref11 Lorsqu'un programme s'exécute, exécute une instruction, puis la suivante jusqu'à la fin du programme. Dans le programme que l'on a fait, on a initialiser le port a en sortie, puis éteint les LED pendant 1s, puis allumé les LED pendant 1s. Si on ne met pas le goto, le programme est terminé et le microcontroleur va exéciter les instruction qui se trouve donc après la fin du programme, des instructions que nous n'avons pas écrite et qui dans le meilleur des cas correspond à 0, c'est-à-dire l'instruction nop ne rien faire. Dans ce cas nos LED restent allumées et ne clignotent pas. Avec le goto jesuisduref11, le programme va continuer à l' »instruction qui suit l'étiquette jesuisduref11:. Les LED vont donc pouvoir s'éteindre et s'allumer indéfiniment avec une période de 2s=0,5Hz. (avec la précision d'un quartz.)

Voici donc notre programme:

```
'Une LED qui clignotte
TRISA = 0  'port a en sortie
jesuisduref11:
PORTA = 0  'met 0V sur toutes les broches du port A qui sont configurées en sortie
WaitMs 1000  'attendre 1s
PORTA = 31  '31=%11111 toutes les pattes du port A sont à +Vcc
WaitMs 1000  'attendre 1s
Goto jesuisduref11
```

Il comprend en tout 8 lignes

Pour le sauver, il faut dans le menu du basic compiler choisir file puis save as. Et le sauver dans le répertoire de votre choix et le nom de votre choix (par exemple LED).

Une fois sauvé, vous pouvez quitter pic simulator iDE, ouvrir wordpad et ouvrir le fichier led.bas que vous venez de créer.

On peut voir que ce n'est qu'un fichier texte.

Quittez worpad, relancez picsimulator ide puis le basic, le dernier programme que l'on a écrit est automatiquement chargé.

Dans le menu du compilateur basic cliquez sur tools, puis sur compile. Cliquez sur close pour fermer la fenêtre des statitique. Et regarder dans le répertoire où vous aviez sauvé le fichier led.bas, il existe maintenant un fichier appelé les.asm. C'est la version assembleur qu'il est possible d'observer avec wordpad. On peut voir alors que nos 8 lignes de basic on été converties en une centaines de lignes assembleur plus ou moins énigmatiques. C'est une des raison qui m'ont poussées à débuter ce cours en basic.

Retournez (ou relancez) pic simulator ide, puis le basic et choisissez tool et compile et assemble. Dans le même répertoire que led.bas et led.asm se trouve aussi un fichier led.hex. C'est ce fichier que l'on mettra dans le pic. C'est l'aboutisement de tout notre travaille de programmation

Nous verrons la prochaine fois la simulation pui senfin le même programme en assembleur MPLAB