

TRAVAUX PRATIQUES DE C.A.O

T.P. 4 : Simulation d'un microcontrôleur PIC

Avant de faire ce T.P. vous devez :

- Etre initié à l'utilisation d'ISIS (avoir fait le T.P. 1)
- Connaître les outils de simulation animée (avoir fait le T.P. 2)

Objectifs de ce T.P. :

- Savoir placer des bus
- Simuler un circuit à base de microcontrôleur.
- Installer un nouvel outil de compilation.

Matériel nécessaire :


- 1 Ordinateur équipé du logiciel Proteus
- Mini guides ISIS & ARES
- Fichiers TL.ASM et TL2.C
- Fichiers du compilateur CC5X

1. Simulation d'un PIC en assembleur

Le circuit que nous utiliserons pour la simulation est issu de la documentation ISIS VSM. Il est composé de deux feux de contrôle du trafic connectés à un microcontrôleur PIC16F84A.

① Démarrez ISIS et saisissez le schéma ci-contre (Les feux tricolores sont dans la bibliothèque, ils se nomment « *TRAFFIC LIGHTS* »).

Pour placer un bus:

1. Appuyez sur le bouton « Bus » .

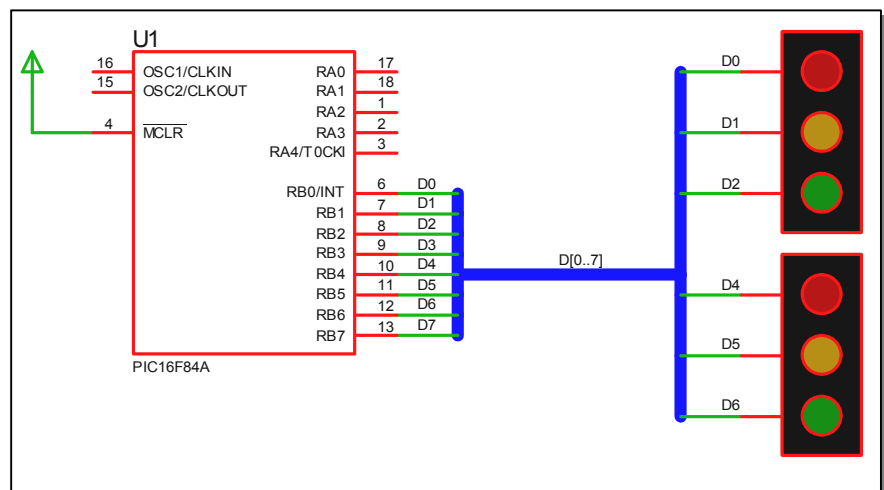
2. Pointez sur l'endroit de départ du bus. Il peut s'agir d'une patte de bus, d'un bus existant, ou d'un espace libre sur le schéma.

3. Clic gauche pour commencer le bus, puis cliquez à chaque angle souhaité pour définir le chemin du bus.

4. Pour finir le bus sur un point de connexion de bus (une patte de bus ou un bus existant), mettez le pointeur dessus et clic gauche. Pour finir un bus sur un espace vide, clic droit.

Labels des bus :

On peut donner un label à un bus, tout comme à un fil. Cependant ISIS définit une syntaxe spéciale pour les bus. Ce sera quelque chose comme $D[0..7]$ ou $A[8..15]$.



② Associer le programme au processeur :
Le programme est présent dans le fichier TL.ASM.

- Copiez le fichier TL.ASM vers votre répertoire.

```

                                TL.ASM

LIST      p=16F84                ; PIC16F844 est le processeur cible
#include  "P16F84.INC"          ; inclure fichier entete

CBLOCK 0x10                      ; zone tampon
    state
    11,12
ENDC

org      0                        ; Vecteur de debut
goto    setports                ; Aller au code de lancement

org      4                        ; Vecteur d'interruption.
halt    goto    halt            ; Stopper en cas de boucle sans fin et ne rien faire

setports  clrw                    ; Zero dans W.
          movwf  PORTA           ; S'assurer que PORTA vaut zero avant validation.
          movwf  PORTB           ; S'assurer que PORTB vaut zero avant validation.
          bsf    STATUS,RP0      ; Selectionner la Bank 1
          clrw                    ; Masquer tous les bits comme sorties
          movwf  TRISB           ; Valider registre TRISB
          bcf    STATUS,RP0      ; Reselectionner Bank 0.

initialise  clrw                    ; Etat initial.
            movwf  state         ; Valider.

loop       call   getmask         ; Convertir etat.
            movwf  PORTB        ; Ecrire vers port.
            incf  state,W        ; Incrementer etat dans W.
            andlw 0x04           ;
            movwf  state        ; Mettre en memoire
            call  wait           ; Attendre :-
            goto  loop          ; et boucler :-

; Fonction qui renvoie un masque de bits pour l'etat courant du port de
; Le nibble superieur contient les bits pour un groupe de feux et celui
; inferieur pour l'autre groupe. Bit 1 pour rouge, 2 pour orange, 3 pour
vert.
; Le bit 4 n'est pas utilise.

getmask    movf  state,W          ; Obtenir etat dans W.
            addwf PCL,F          ; Ajoute offset dans W de PCL pour calc. goto.
            retlw 0x41           ; state==0 est Vert et Rouge
            retlw 0x23           ; state==1 est Orange et Rouge
            retlw 0x14           ; state==3 est Rouge et Vert
            retlw 0x32           ; state==4 est Rouge/Orange et Orange

; Fonction qui utilise deux boucles pour definir un retard.
wait       movlw 5
            movwf 11

w1         call  wait2
            decfsz 11
            goto  w1

            return

wait2     clrf 12
w2        decfsz 12
            goto  w2
            return
END

```

- Dans le menu « Source » Sélectionnez la commande « Ajout/suppression fichiers source ». Cliquez sur le bouton « Nouveau », sélectionnez le fichier TL.ASM.
- Choisissez « l'outil de génération de code ». Pour les microcontrôleurs PIC, l'outil est MPASM (Pour un microcontrôleur 68HC11, il faudrait utiliser le ASM11). C'est ce compilateur qui sera lancé par Proteus lors du lancement de l'animation. Le résultat de la compilation produira entre autres le fichier « TL.HEX ».

③ Simulation du circuit :

- Editez le PIC16F84A et ajoutez « TL.HEX » dans le champ « Program File ».
- Sauvegardez votre fichier dans votre répertoire (le même que celui où se trouve TL.ASM).
- Appuyez sur le bouton Jouer du « magnétoscope ». La barre de message doit indiquer le temps écoulé depuis le lancement de l'animation.



Vous noterez qu'un des feux est vert alors que l'autre est rouge mais que les feux ne changent pas d'état. Ceci provient d'une erreur volontairement introduite dans le programme. A ce stade, il convient de mettre au point le programme pour résoudre le problème.

④ Mise au point :

- Lancez le mode « Debug » en utilisant le menu « debug / start/restart debugging » (ou la combinaison des touches CTRL+F12).
- Vous pouvez choisir d'afficher les fenêtres de mise au point par l'intermédiaire du menu « debug ». Activez les fenêtres « Registers », « Source Code » et « Data Memory ».

The screenshot displays three windows from the Proteus IDE:


- PIC CPU Registers - U1:** Shows register values such as PC: \$0000, INSTR.: GOTO 0x0005, and various status and control bits.
- PIC CPU Source Code - U1:** Shows assembly code for the PIC16F84A. A callout bubble points to the PC register value in the code, stating "Position courante du compteur de programme (PC)".
- PIC CPU Data Memory - U1:** Shows a memory dump with addresses and hex values.

A red circle highlights the debug toolbar in the Source Code window, with a callout bubble stating "Boutons de mise au point".

Boutons de mise au point de la fenêtre « Source Code »

| | | | |
|--|---|--|---|
| | Relance l'animation | | Pas à pas avec sortie forcée de la sous fonction courante |
| | Pas à pas qui n'entre pas dans les sous fonctions | | Exécution jusqu'à la ligne sélectionnée |
| | Pas à pas | | Ajouter / enlever des points d'arrêt |


⑤ Analyse du déroulement du programme :

- Pour lancer le programme en mode pas à pas, appuyez sur le bouton  de la fenêtre « Source Code » (ou sur F11) autant de fois que nécessaire pour que le programme se trouve dans la boucle w2. L'instruction « `decfsz 12` » décrémente la variable 12 qui est en mémoire de données.




REMARQUE : Pour connaître l'adresse d'une donnée, placez le « Compteur de Programme » sur l'instruction qui modifie cette donnée. L'instruction et l'adresse sont affichées en vert dans la fenêtre « Registers ».


Quelle est l'adresse de 12 ?

- La boucle w2 est une boucle d'attente. Pour éviter d'avoir à appuyer 256 fois sur F11, il est possible de forcer la sortie de cette boucle en cliquant sur le bouton .

A quelle adresse se trouve le « Compteur de Programme » à la sortie de la boucle w2 ?

- Il est possible d'exécuter la boucle w1 sans exécuter l'appel vers le sous-programme wait 2. Pour cela, cliquez sur le bouton .

Combien de fois s'exécute la boucle w1 ?

- Il est également possible d'exécuter le programme jusqu'à un point donné : Sélectionnez la ligne jusqu'à laquelle vous souhaitez que le programme s'exécute, puis cliquez sur . Attention ! le programme tourne jusqu'à la ligne sélectionnée mais celle-ci n'est pas exécutée.



Quelle est la valeur du registre W lorsque le programme a exécuté la ligne d'adresse 001B ?

Une investigation plus poussée révèle que le problème est causé par un ET logique avec 4 au lieu de 3. La solution est de modifier l'instruction « `andlw` » par 3 au lieu de 4.

- A partir du menu « Source », affichez TL.ASM. Corrigez le programme, enregistrez-le et simulez-le de nouveau.

Le trafic semble un peu rapide !

- Editez le processeur et modifiez sa vitesse d'horloge (10kHz).

- Il peut être intéressant de placer un point d'arrêt. Pour ce faire, dans la fenêtre « *Source Code* », sélectionnez la ligne souhaitée (par exemple sur l'instruction `andlw` à l'adresse 0011), puis cliquez sur le bouton  (ou appuyez sur F9). A chaque fois que vous relancerez l'animation (bouton  ou F12), le programme se stoppera sur l'instruction précédent le point d'arrêt.

Quelle est la valeur de la variable `state` (adresse 0010 de la mémoire de données) lorsque les signaux D1, D4 et D5 sont à l'état haut ?

- Il est possible de visualiser l'état des variables alors que le programme est en cours d'animation :
 - Enlevez le point d'arrêt
 - Relancez l'animation.
 - Dans le menu « *Mise au point* », cochez l'affichage de la fenêtre « *Watch window* ».
 - Cliquez à droite sur la fenêtre « *Watch window* » et sélectionnez « *Ajout items (par noms)* ». Ajoutez « `PORTB` » (double clic). Notez que la valeur de « `PORTB` » évolue en même temps que la couleur des feux.
 - Cliquez à droite sur la fenêtre « *Watch window* » et sélectionnez « *Ajout items (par adresses)* ». Ajoutez l'adresse `0x0011` que vous nommerez L1.

Quelle est la valeur de `PORTB` lorsque les signaux D1 et D6 sont à l'état haut ?



Selon le niveau de licence de Proteus, il n'est possible de simuler que certains processeurs. Par exemple, la licence 2+ permet de simuler les PIC16, HC11 et 8051. Il est impossible de simuler un PIC18 ou un PIC12 !

2. Installation d'un compilateur C

Il est possible de rajouter des compilateurs externes en plus de ceux existants. Nous allons dans l'exemple suivant intégrer un compilateur C pour les PIC 16. Le compilateur choisi est le CC5X dont une version limitée à 1024 instructions est disponible gratuitement sur le site <http://www.bknd.com/>

① Installation du compilateur :

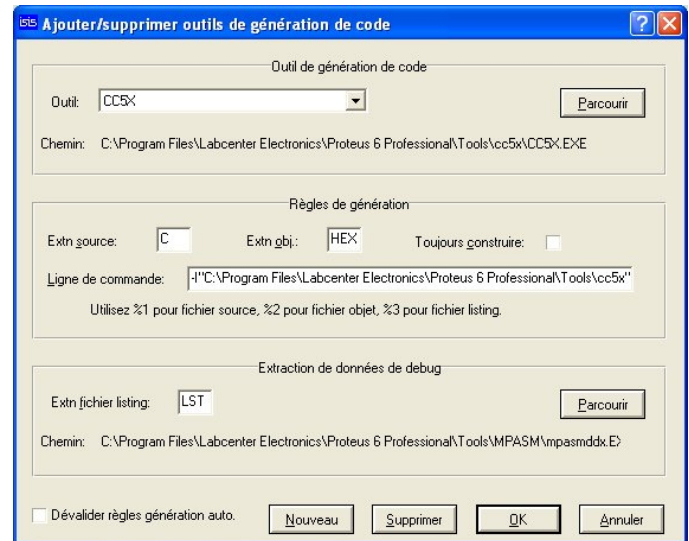
- Procurez vous CC5X.
- Copiez le dossier CC5X contenant le programme `CC5X.EXE` ainsi que les fichiers `.H` dans le sous-dossier TOOLS de l'installation de Proteus.
- Dans Isis, allez dans « *Source / Définir outil de génération de code* »

- Cliquez sur le bouton « *Nouveau* »
- Sélectionnez le programme **CC5X.EXE** (*il y a des chances qu'il soit dans le dossier C:\Program Files\Labcenter Electronics\Proteus 6 Professional\Tools\cc5x*)
- Complétez la fenêtre comme ci-dessous :
 - Le fichier source a comme extension **.C**
 - Le fichier compilé a comme extension **.HEX**
 - La ligne de commande :

%1 -L -CC -A1+6+10 -AuJm -"C:\Program Files\Labcenter Electronics\Proteus 6 Professional\Tools\cc5x"

- **%1** correspond au nom du fichier source
- **-L -CC -A1+6+10 -AuJm** correspond à des options de CC5X
- **-"C:\Program Files\Labcenter Electronics\Proteus 6 Professional\Tools\cc5x"** est le chemin où sont placés les fichiers **.H**
- Pour plus d'informations, voir la documentation de CC5X

- Le fichier listing a comme extension **.LST**
- Cliquez sur le bouton « *Parcourir* » associé aux données de debug et donnez le chemin du programme de débogage de MPASM : **MPASMDDX.EXE** En général, le chemin est :



"C:\Program Files\Labcenter Electronics\Proteus 6 Professional\Tools\MPASM\mpasmddx.EXE"

- Cliquez sur OK

② Simulation d'un programme en C :

- Réutilisez le schéma précédent avec les feux de croisement. Modifiez les informations relatives au programme :
 - Le fichier source est **TL2.C**
 - L'outil de génération de code est **CC5X**
 - Le fichier compilé est **TL2.HEX**
- Simulez ce programme