LOGIPIC

Logiciel de programmation graphique des microcontrôleurs PIC

Utilisation des Timers : Création d'un chronomètre

Cette séquence va vous permettre de juger la puissance de LogiPic, en effet en quelques clicks de souris, vous allez créer un chronomètre précis au centième de secondes avec contrôles start, stop et reset.

Les fonctions utilisées :
- Affichage lcd
- Interruption Timer2
- Routines
- Temporisations
- Gestion des Entrées
- Gestions des variables

Matériel nécessaire :
- Pic 16f628A
- Afficheur lcd 2*16
- Quartz 4 MHz
- 2 condensateurs non polarisés 15 pF
- 2 poussoirs

Pourquoi utiliser une interruption Timer ?

Une des prérogative d'un chronomètre est d'être le plus précis possible. L'intérêt d'une interruption timer est que le microcontrôleur arrête l'exécution du programme périodiquement pour faire appel à un sous-programme, on est donc sûr d'avoir des laps de temps réguliers.

Fonctionnement des registres du Timer2 (un peu de maths)

Les registres utilisés sont : PR2 et T2CON
Nous allons déterminer les paramètres des ces 2 registres pour l'utilisation du chronomètre.

Nous devons créer un chrono avec une précision au centième de seconde, il nous faut pour cela une interruption timer toutes les 0,01 s (10ms).

Calcul avec un **Quartz de 4MHz** :

On sait (datasheet) que un cycle du microcontrôleur correspond à ¼ de la fréquence du quartz donc 1MHz, soit 1 000 000 de cycles par seconde (période = 0,000 001 s). La période de notre interruption devant être de 0,01s il faudra que celle-ci soit appelée toutes les 0,01 / 0,000001 = 10000 cycles.

Le Timer2 dispose d'un pré-diviseur et d'un post-diviseur, le produit des 2 donne la valeur du diviseur total.

Les combinaisons possibles sont les suivantes : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 , 11, 12, 13, 14, 15, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240, 256.

Formule : Durée totale = Durée d'un cycle * pré-diviseur * post-diviseur * (PR2 +1)
Donc : 0,01 = 0,000 001 * pré-diviseur * post-diviseur * (PR2 +1)
10000 = pré-diviseur * post-diviseur * (PR2 +1)

Il faut déterminer le produit post*pré-diviseur pour que PR2 soit un entier inférieur ou égal à 255.

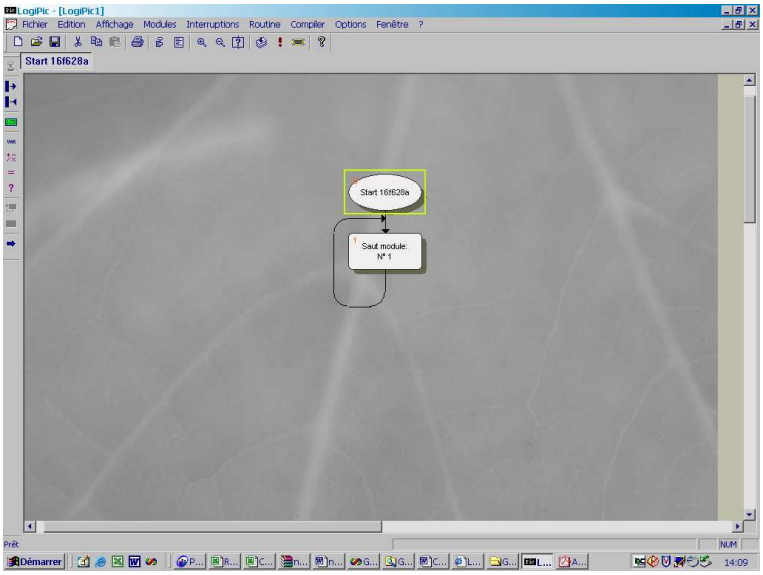
Par exemple : 10000 / 80 –1 = 124 5*16 = 80 pré-diviseur = 5 (0100 datasheet) post-diviseur = 16 (10 datasheet)

PR2 = 124
T2CON = 00100010

Ce calcul n'est valable que pour un fonctionnement avec un quartz de 4MHz.

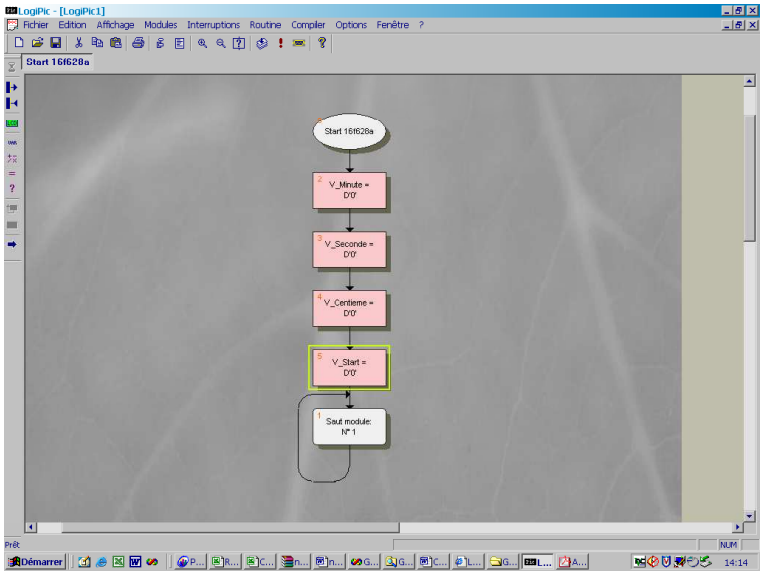
Maintenant que nous avons calculé la valeur de nos registres, nous allons pouvoir commencer la programmation graphique.

- Lancer LogiPic et créer un nouveau programme en sélectionnant un 16f628A



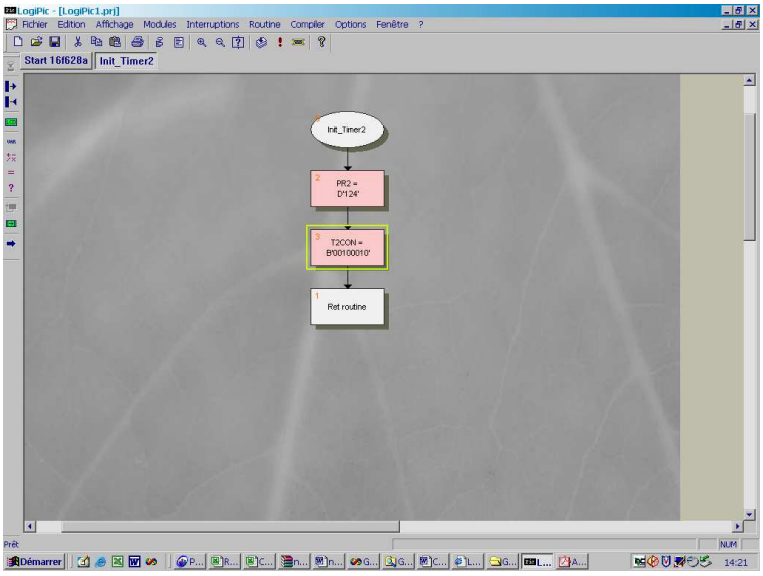
Nous allons définir les variables de fonctionnement :

- Créer les variables suivantes V_Minute, V_Secondes, V_Centieme, et V_Start
- Initialiser ces variables à zéro dans la boucles principale



Il faut maintenant initialiser les registre PR2 et T2CON du Timer2 en fonction des valeurs calculées précédemment

- Créer une routine « Init_Timer2 »
- Insérer deux modules d'initialisation pour le registre PR2 = 124 (décimale) et T2CON = 00100010 (binaire)

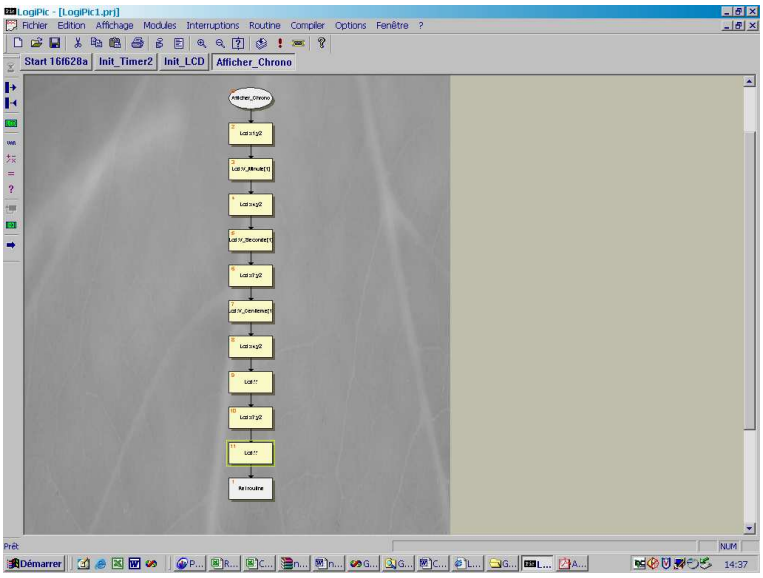


Nous allons créer une routine pour afficher le temps écoulé, elle sera appelé toutes les 1/100éme de seconde.

- Créer une routine « Afficher_Chrono »
- Insérer :
  - un module de positionnement du curseur à x=1 et y=2
  - un module d'affichage de la variable V_Minute
  - un module de positionnement du curseur à x=4 et y=2
  - un module d'affichage de la variable V_Secondes
  - un module de positionnement du curseur à x=7 et y=2
  - un module d'affichage de la variable V_Centieme

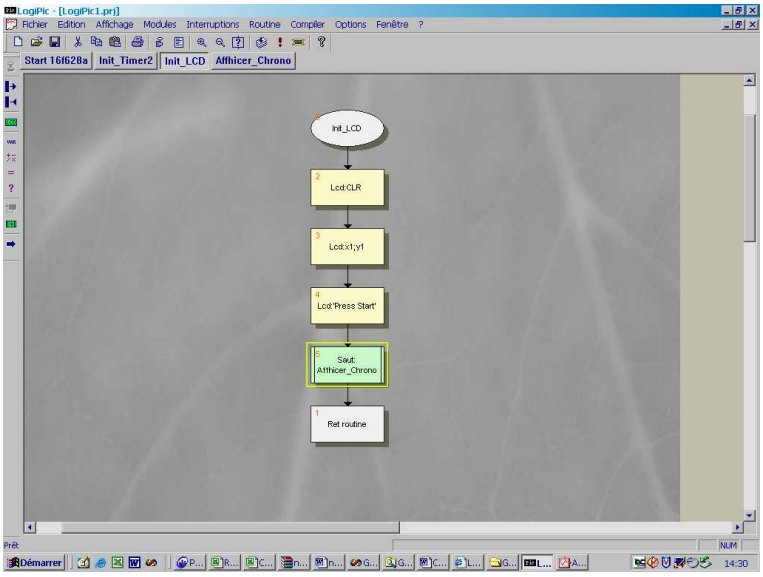
Ensuite nous allons afficher les « : » entre les variables

- Insérer :
  - un module de positionnement du curseur à x=4 et y=2
  - un module d'affichage du texte « : »
  - un module de positionnement du curseur à x=7 et y=2
  - un module d'affichage du texte « : »



Maintenant il faut créer une routine pour initialiser l'affichage de l'écran

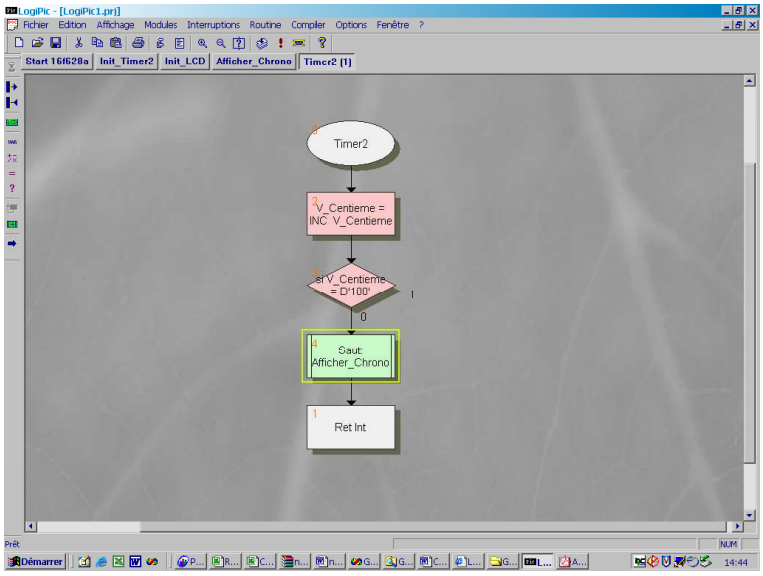
- Créer une routine « Init_LCD »
- Insérer un module d'effacement de l'écran (Clear)
- Insérer un module de positionnement du curseur à x=1 et y=1
- Insérer le texte « Press Start »
- Enfin placer un module de saut vers la routine « Afficher_Chrono » ce qui aura pour effet d'afficher au démarrage 00:00:00 étant donné que les variables ont été initialisées à zéro.



Nous avons paramétré les registres du timer2, il est maintenant nécessaire de gérer l'appel de l'interruption.

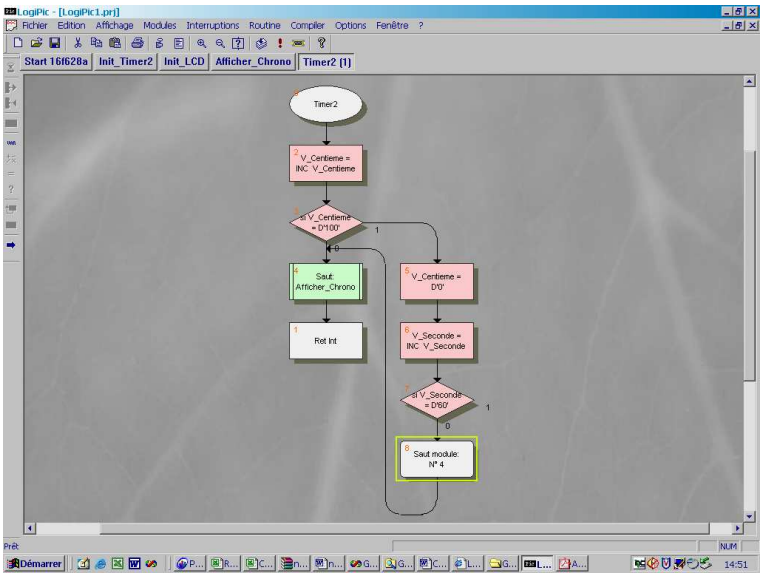
- Dans le menu Interruption, cliquer sur nouvelle, sélectionnez Timer2
- Placer un module (opération à une opérande) pour incrémenter la variable V_Centième
- Placer un module pour tester si V_Centième = 100 avec sortie à droite si condition vrai
- Sous le module test, placez un saut vers la routine « Affiche_Chrono »

A chaque passage dans l'interruption, la variable V_Centième sera incrémentée et affichée sur l'écran.



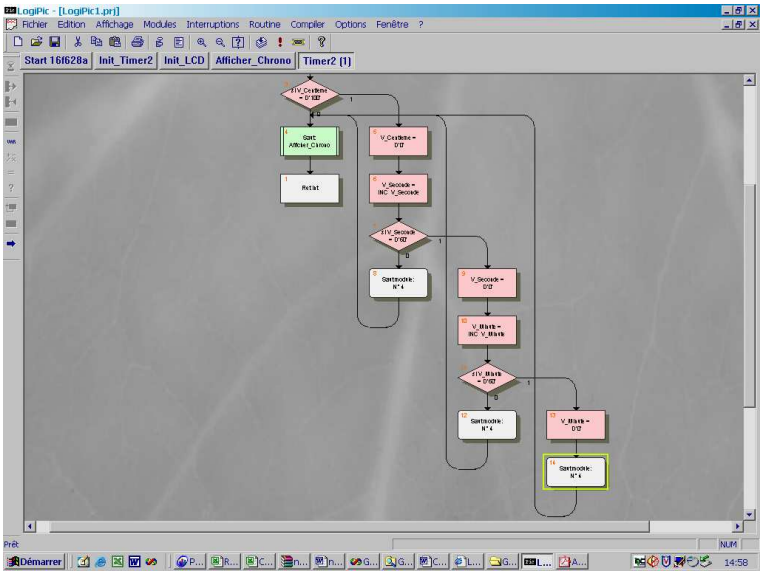
Il faut traiter le cas où l'on dépasse la valeur 99 :
Si V_Centième = 100 alors on la remet à zéro et on incrémente V_Seconde….

- Insérer un module d'initialisation V_Centième = 0 à droite du test
- Incrémenter V_Seconde
- Placer un test pour V_Seconde=60 avec sortie à droite si vrai
- Placer sous le test un retour vers le module 4 « Affiche_Chrono »



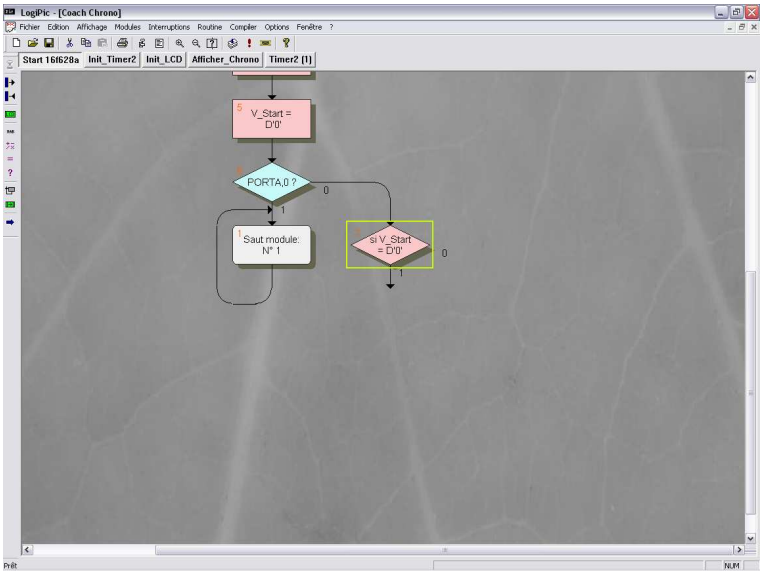
Si V_Seconde = 60 alors on la remet à zéro et on incrémente V_Minute….

- Insérer un module d'initialisation V_Seconde = 0 à droite du test
- Incrémenter V_Minute
- Placer un test pour V_Minute =60 avec sortie à droite si vrai
- Placer sous le test un retour vers le module 4 « Affiche_Chrono »
- Insérer un module d'initialisation V_Minute = 0 à droite du test
- Placer sous le test un retour vers le module 4 « Affiche_Chrono »



Nous allons maintenant gérer l'appui sur les bouton poussoirs, un appui sur l'entrée PORTA,0 fera démarrer le chrono, un autre appui l'arrêtera.

- Placer dans la boucle principale sous le dernier saut de routine un module entrée PORTA,0 avec sortie en dessous si entrée à 1.
- Placer à droite du module entrée un test sur la variable V_Start = 0 avec sortie dessous si vrai

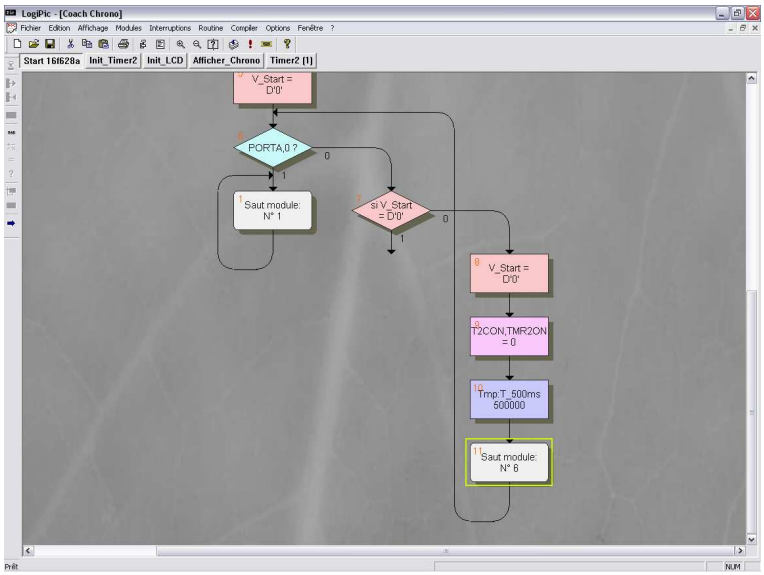


Il faut que la variable V_Start change d'état à chaque appui :

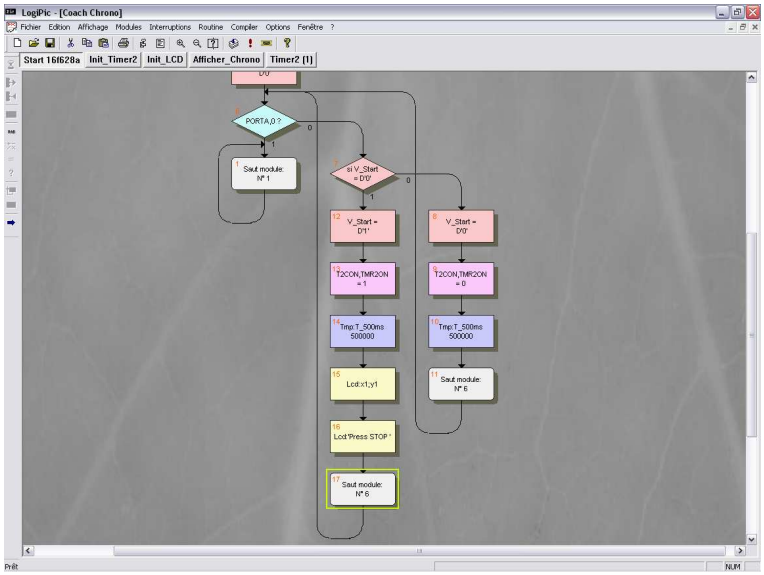
- A droite du test, initialiser la variable V_Start=0
- Placer ensuite le bit TMR2ON du registre T2CON à 0 pour arrêter le timer2

Lorsque l'on appui sur un bouton, une multitude de petits contacts se créent pendant une période transitoire de quelques ms,

- Créer et placer une temporisation de 500ms pour masquer les rebonds du bouton poussoir
- Placer un retour vers le module entrée PORTA,0

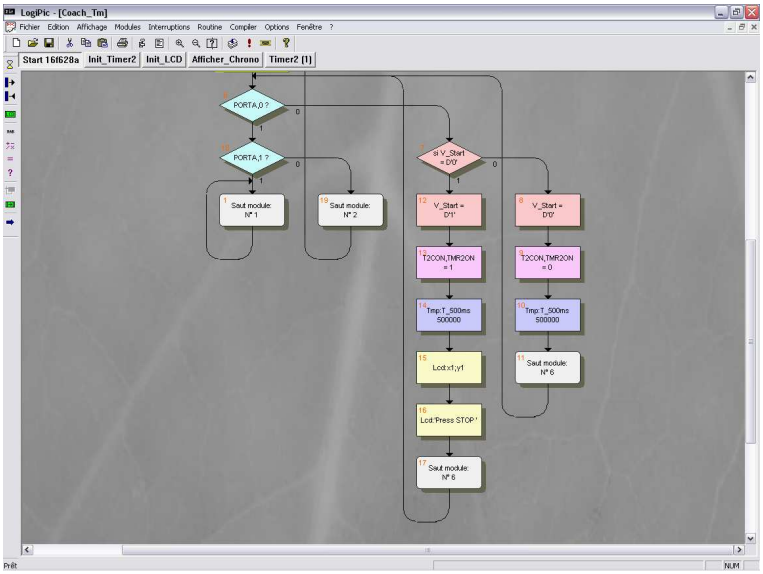


- Sous le test, initialiser la variable V_Start = 1.
- Placer ensuite le bit TMR2ON du registre T2CON à 1 pour la mise en route du timer2
- Placer la temporisation précédemment créée de 500ms pour masquer les rebonds du poussoir
- Placer un positionnement LCD à x=1, y=1
- Placer un affichage texte « Press STOP »
- Placer un retour vers le module entrée PORTA,0



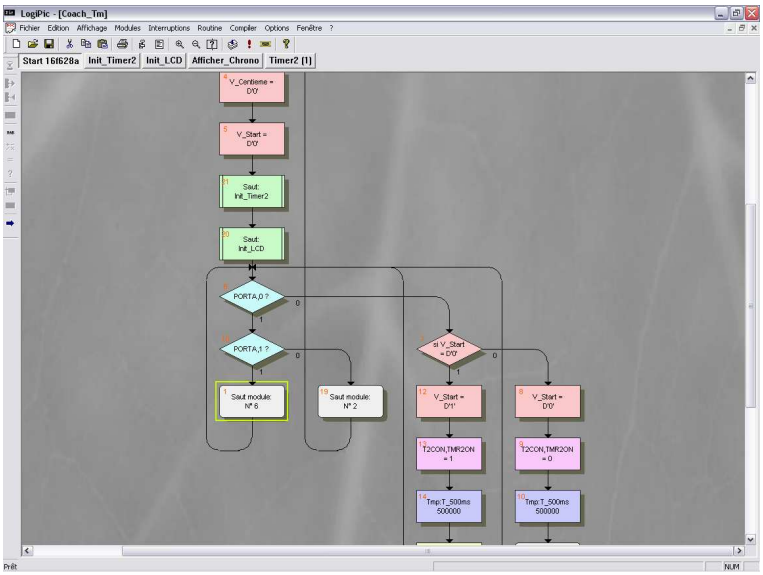
Il nous reste à gérer l'appui sur le deuxième bouton pour remettre à zéro le chrono.

- Placer un deuxième module entrée PORTA,1 avec sortie en dessous si entrée à 1.
- Placer à droite du test un saut vers le début du programme pour tout remettre à zéro.

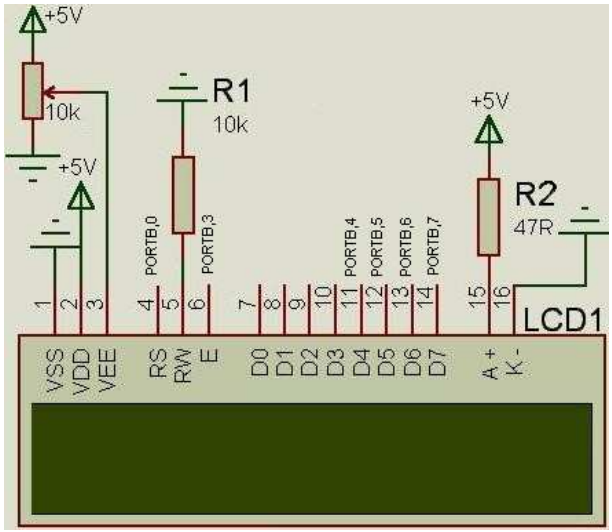


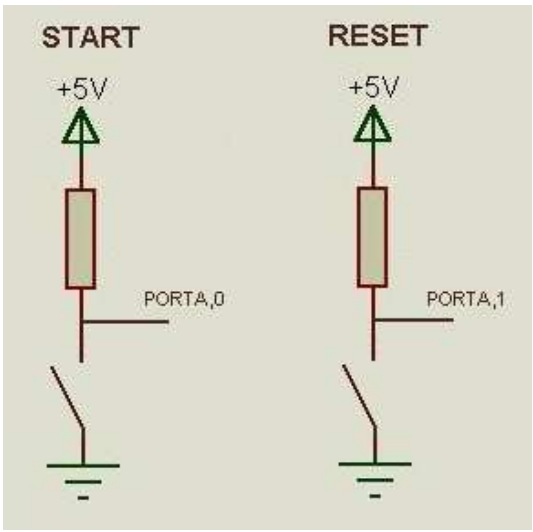
Nous avons créer des routines d'initialisation du LCD et du Timer2, nous devons les appeler dans la boucle principale
Sous l'initialisation des variables,

- Placer un appel routine vers « Init_timer2 »
- Placer un appel routine vers « Init_LCD »
- Boucler le programme vers le test de l'entrée PORTA,0, pour scruter en permanence l'appui sur un des boutons par l'utilisateur.



Le programme est maintenant terminé, il ne vous reste plus qu'a réaliser le montage sur une platine d'essai.


+5V
10k
+5V
R1
10k
+5V
R2
47R
PORTB,0
PORTB,3
PORTB,4
PORTB,5
PORTB,6
PORTB,7
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
LCD1
VSS VDD VEE RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A + K -


START
+5V
PORTA,0
RESET
+5V
PORTA,1