LES TIMERS

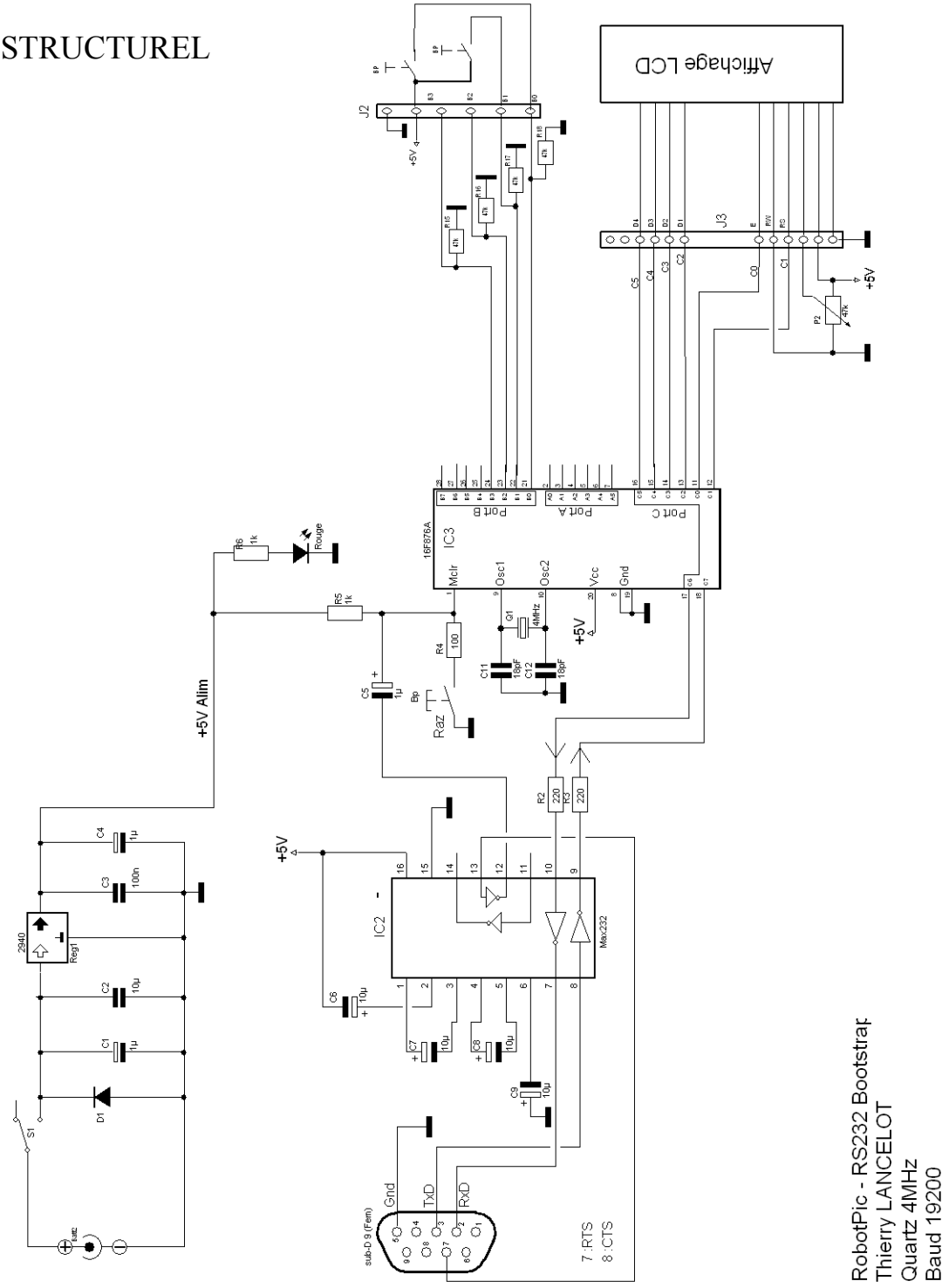
Objectif : Il s'agit de comprendre et d'utiliser les différents timer des microcontroleurs type 16F.

1 - LA CIBLE

Les programmes sont réalisés sur la platine BootRS232. Cette cible est constituée par un microcontroleur 16F876A avec un oscillateur interne à 4MHz. Pour fonctionner en mode bootstrap, le pic devra être programmé une première fois avec un bootloader (16F876A_4M.hex), le test des programmes s'effectuera avec Tinybootloader dont l'appel est possible dans la version 2.07 de Logipic.

L'ensemble des explications ne s'attardera pas sur la prise en main de Logipic considérant que vous connaissez suffisamment ce formidable logiciel.

SCHEMA STRUCTUREL



2 - LES TIMERS

Les timers sont en fait des compteurs permettant :

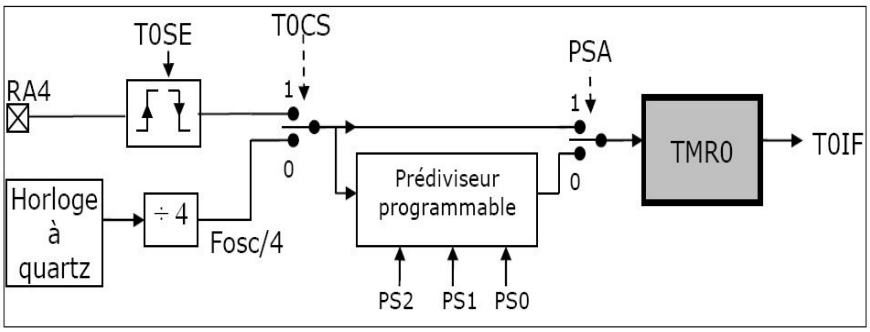
- ➔ de compter des impulsions sur une pin spécifique du pic (voir les datasheets).
- ➔ de compter des cycles d'horloge de la pic elle-même, comme l'horloge du pic est fixe, nous pourrons ainsi disposer d'une base de temps, c'est le mode timer.

Le pic 16F876 (mais aussi le pic 16f628) dispose de 3 timers.
Dans Logipic, l'appel d'un timer s'effectue en effectuant un « insert interruption » : c'est normal puisqu'il s'agit d'une interruption. C'est pourquoi, le programme principal se déroute vers le timer de façon temporelle et non pas à partir d'un événement comme l'appui sur un bouton-poussoir.
Pour ne pas restreindre le pic à une seule fréquence (ou temps), des diviseurs (que l'on appelle des prédiviseurs) permettent de choisir le temps compatible avec l'application utilisée.

2 - LE TIMER0

La mise en œuvre du timer0 s'effectue grâce au registre **OPTION** :

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
NOTRBPU	**INTEDG**	**TOCS**	**TOSE**	**PSA**	**PS2**	**PS1**	**PS0**
1	**1**	**0**	**1**	**0**	**Calcul du prédiviseur: 1 1 1**		



Après un reset le registre OPTION se positionne à 11111111

NOTRBPU : sans rapport avec le timer0, ce bit permet de renseigner le mode de fonctionnement concernant la présence de résistances de rappel sur le portB. 0 : positionne des résistances en interne.

INTEDG : sans rapport avec le timer0, ce bit indique le sens de déclenchement sur l'interruption RB0. Nous n'utilisons pas cette option, nous pouvons laisser ce bit à 1.

TOCS : permet de choisir le mode de fonctionnement du timer0, 0 permet de sélectionner l'horloge interne du pic, 1 permet de sélectionner les impulsions sur la pin RA4. Il faudra donc positionner le bit5 à 0.

TOSE : fonctionne en relation avec TOCS, si TOCS est à 1, cela permet de choisir le sens de déclenchement sur la pin RA4. On peut laisser le bit4 à 1.

PSA : permet de choisir entre le prédiviseur et le watchdog. 0: timer, 1 : watchdog. Il faudra donc positionner le bit3 à 0.

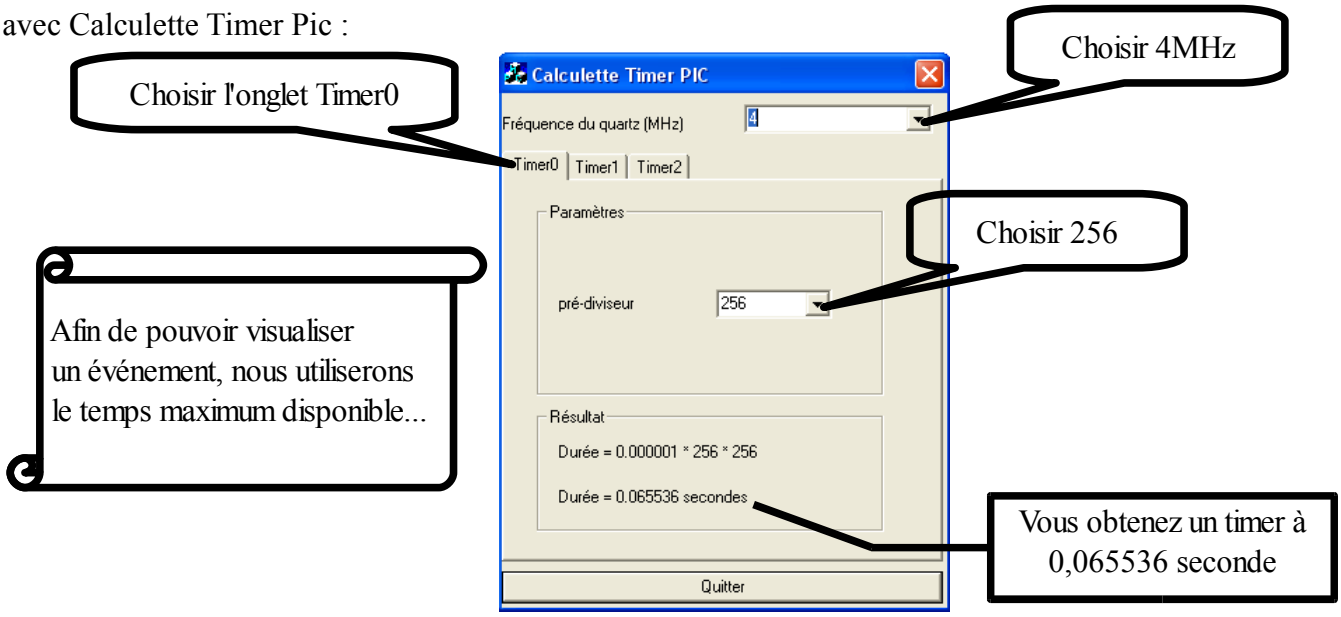
PS2 :
PS1 : } choix du prédiviseur dans notre exemple tous ces bits sont à 1. / de 256
PS0 :

PS2	**PS1**	**PS0**	Prédiviseur
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

Le calcul de durée s'effectue à partir de la fréquence de base du pic (oscillateur interne de 4 Mhz). Pour bien comprendre le calcul de temps reportez vous au datasheet du pic, pour cette exemple, nous utiliserons le logiciel Calculette Timer Pic.

Vérifier le timer0

avec Calculette Timer Pic :



Il est temps de passer à la réalisation d'un programme exploitant le timer0 :

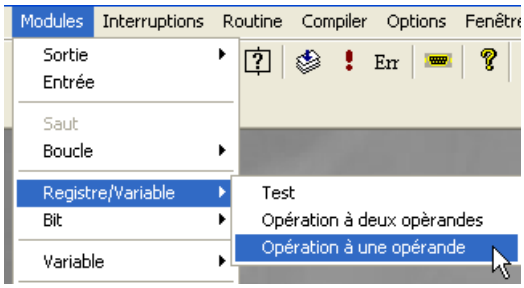
Lancer Logipic

Cliquer sur Nouveau Sélectionner 16f876A

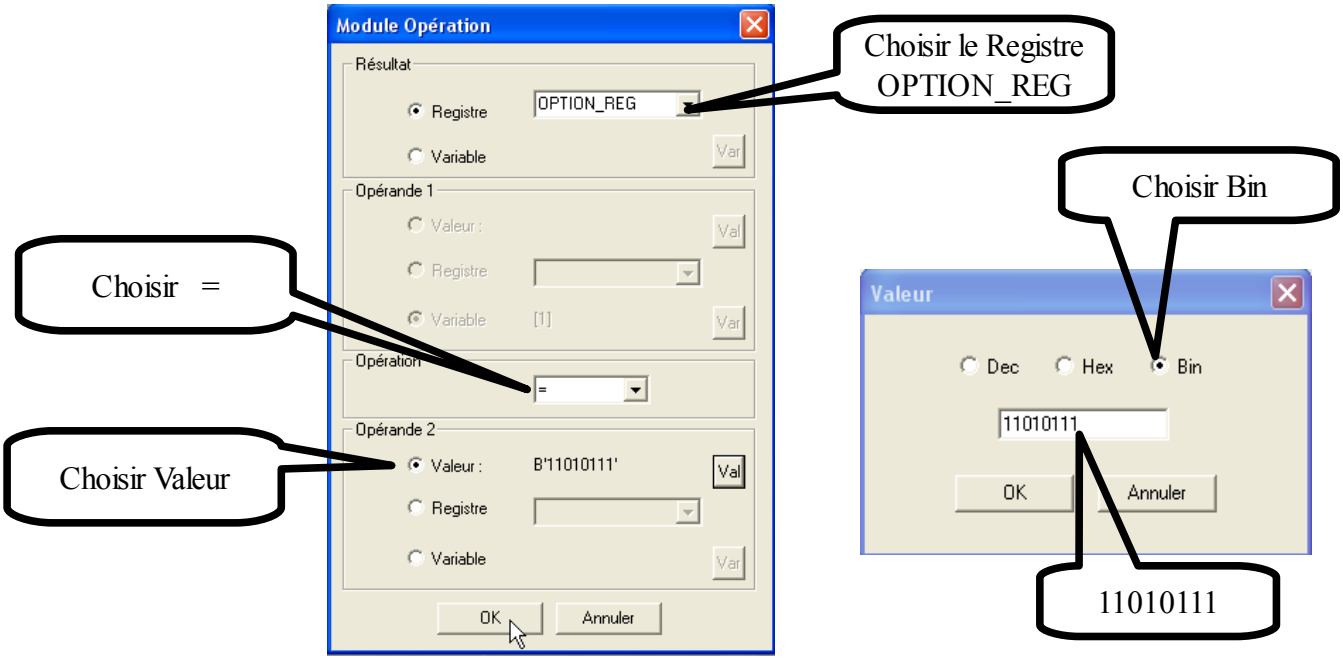
Enregistrer votre projet avec un nom facile à retenir par exemple timer0

Dans un premier temps, il faut configurer le registre option :

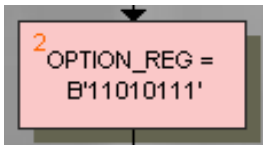
Choisir Modules
Registre/Variable
Opération à une opérande :



ou directement sur =



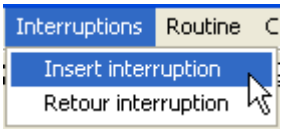
Le registre OPTION est maintenant positionné correctement :

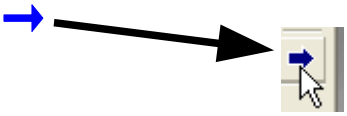


Le timer0 est en fonctionnement

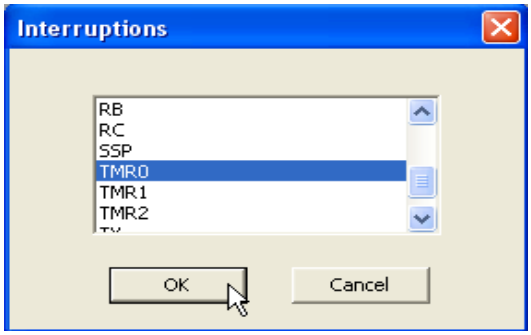
L'appel au timer0 s'effectue en demandant une interruption :

Choisir Interruptions
Insert interruption



ou directement sur 

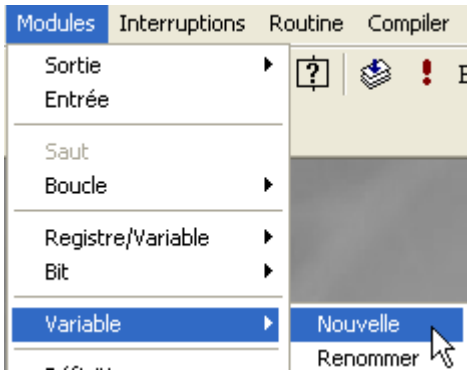
Choisir le Timer0



Il faut bien comprendre que l'appel au sous-programme timer0 s'effectue dorénavant tous les 0,065536 econde.

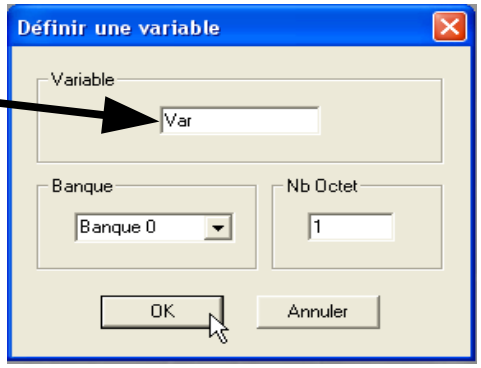
Pour faire fonctionner notre programme, nous allons demander l'affichage d'une variable lors de chaque interruption timer0.

Nous avons besoin d'une variable que nous appellerons Var :
Choisir Modules
Variable
Nouvelle



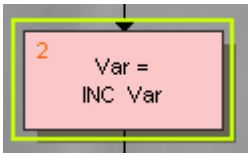
ou directement sur VAR 

définir la variable Var

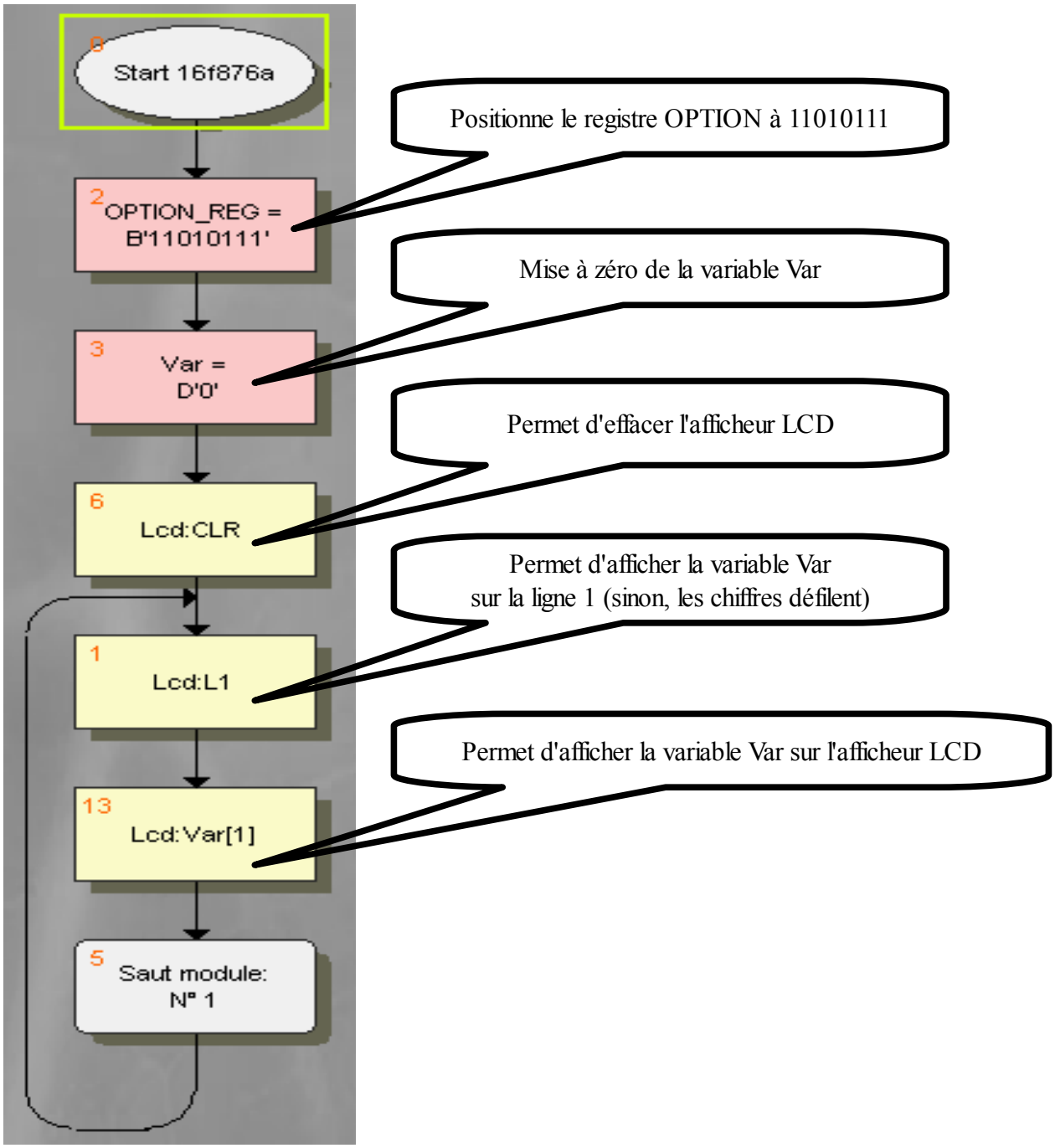


Dans le sous-programme timer0, demander une incrémentation de la variable Var :
Demander 

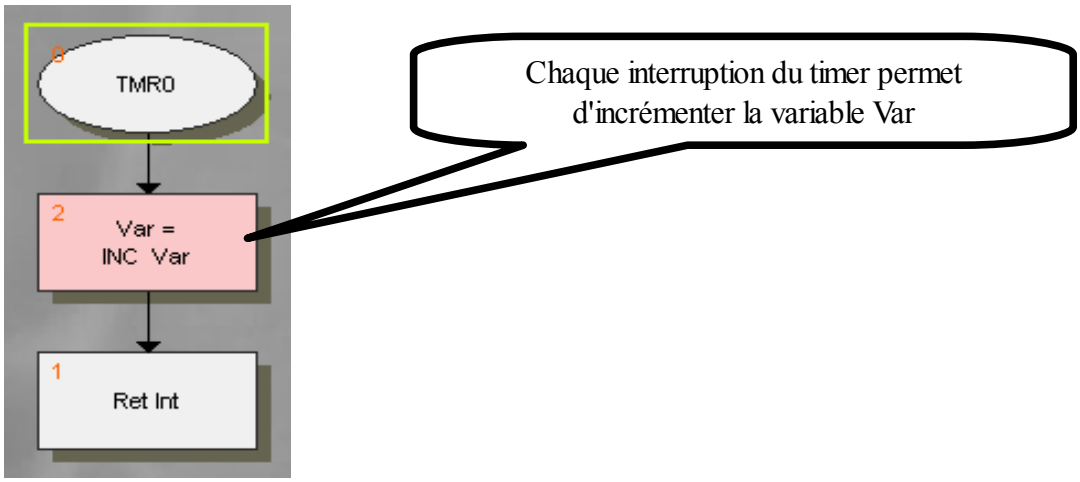
écrire le module suivant :



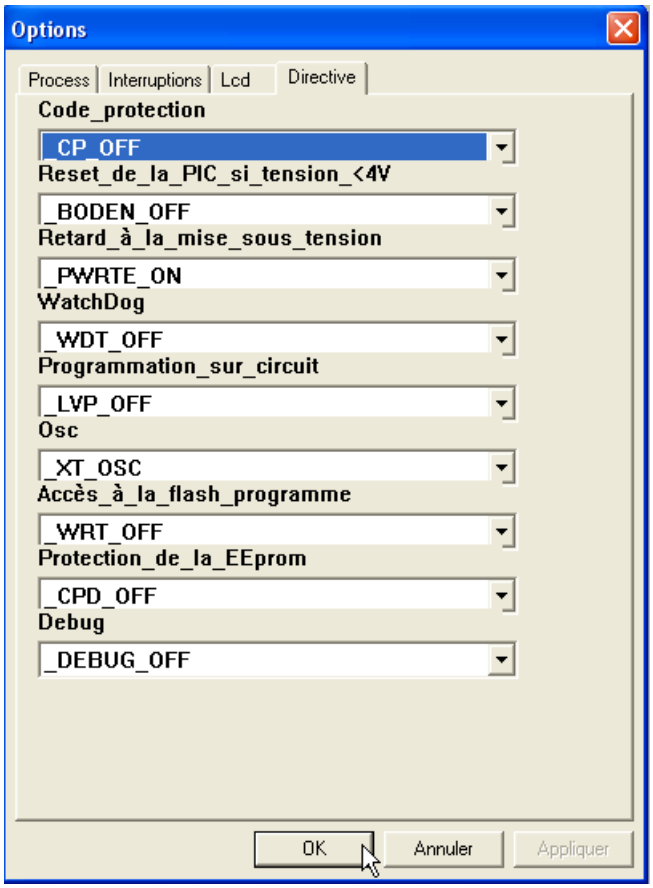
Sans être impératif, il est prudent de positionner la variable Var à 0 et d'effacer l'écran LCD
Le programme principal se résume donc à :

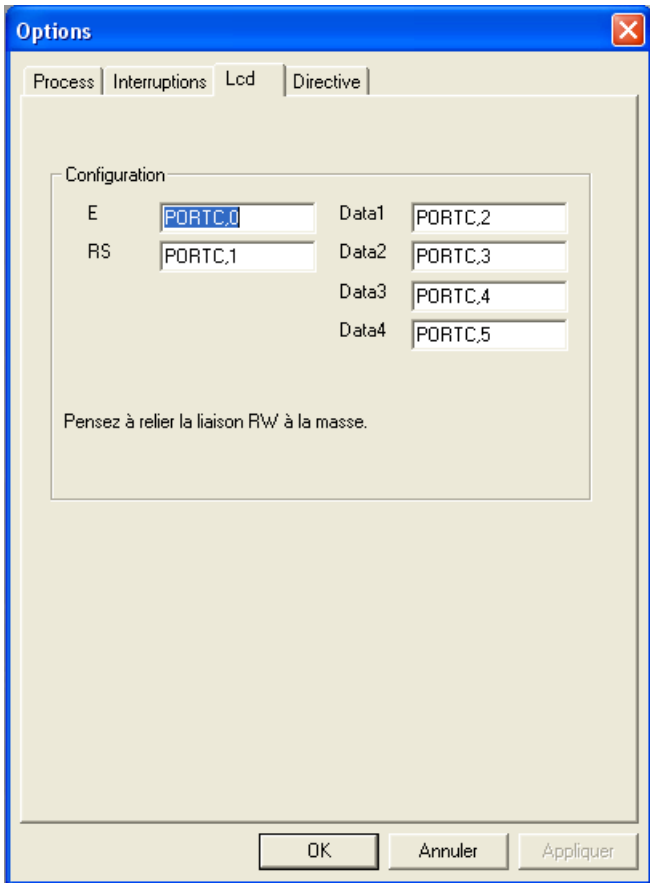


L'interruption timer0 se résume à



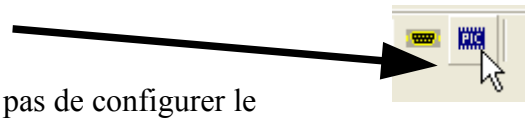
Avant de compiler le projet, n'oubliez pas de configurer correctement les options nécessaire à la cible et tout particulièrement le choix de l'oscillateur interne. Il est nécessaire aussi de configurer le portC sur lequel se trouve l'affichage LCD.



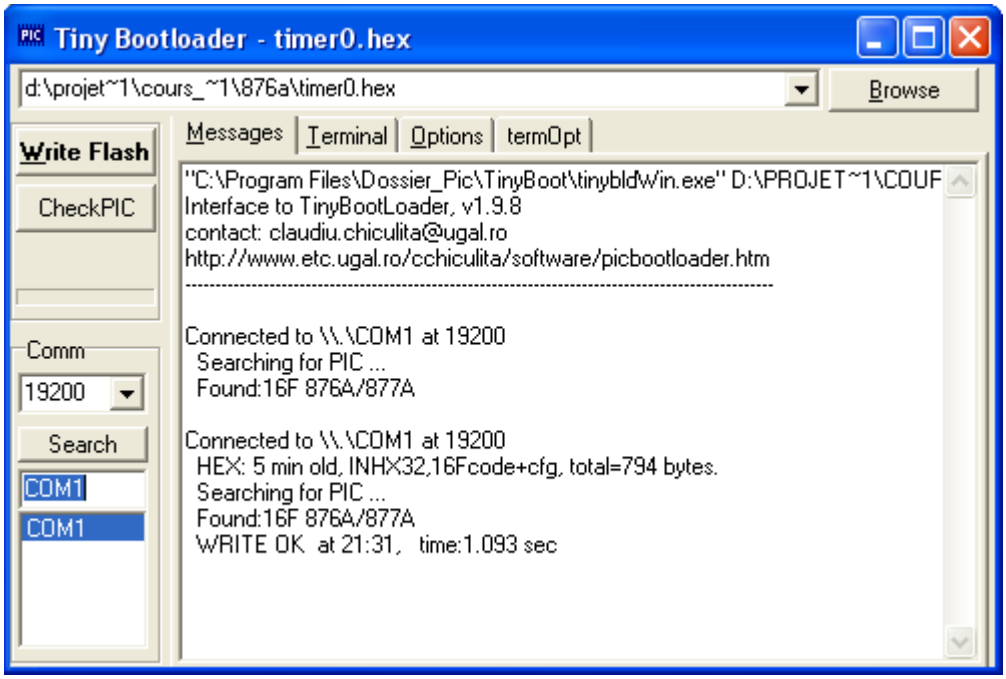


Après compilation, le fichier peut-être transférer sur la cible avec TinyBootloader (ou bien classiquement avec IcProg). Vous pouvez transférer le programme sur la cible, si tout se passe bien, vous obtenez le résultat suivant :

Logipic V2.07 dispose d'un appel direct à Tinybootloader



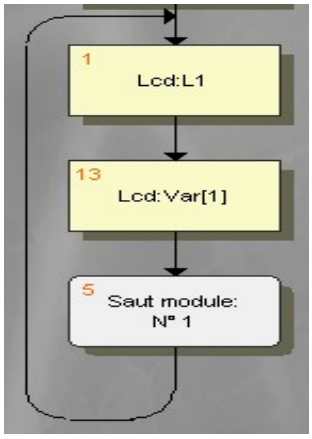
Pour que l'ensemble fonctionne correctement, n'oubliez pas de configurer le chemin d'accès de Tinybootloader ainsi que les options de celui çi. : choisir le port Com de votre système et la vitesse de transfert (19200).



Dès le transfert terminé, la cible se met en route, l'afficheur LCD affiche la variable Var qui évolue.

FONCTIONNEMENT DU PROGRAMME :

Dès la mise en route du programme, OPTION_REG est configuré pour faire fonctionner le timer0, la variable Var est positionnée à 0. Ensuite, le programme boucle indéfiniment dans cette configuration :



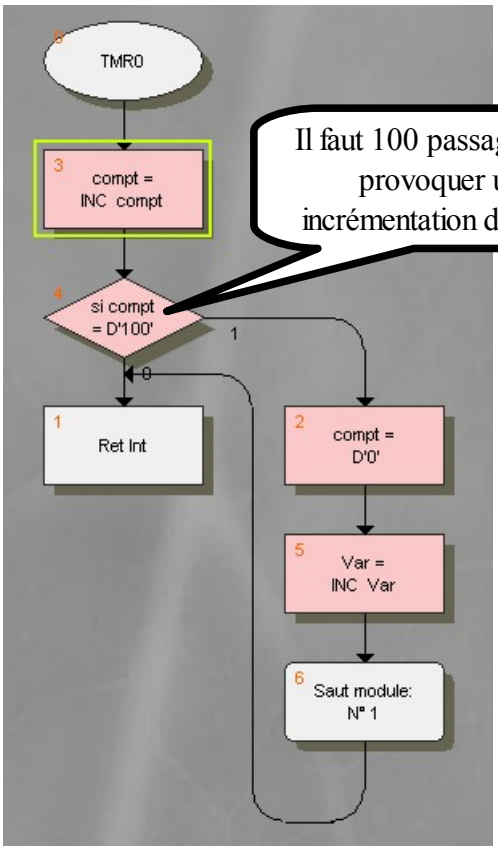
Seul le timer0 permet de sortir de cette boucle.

Le Ret Int du timer permet de retourner au programme principal.

Le comptage peut aller nettement plus rapidement en modifiant la valeur du prédiviseur. En utilisant le même programme, modifier le registre OPTION_REG avec la valeur suivante : 10000110
Avec cette valeur, le compteur est 2 fois plus rapide...

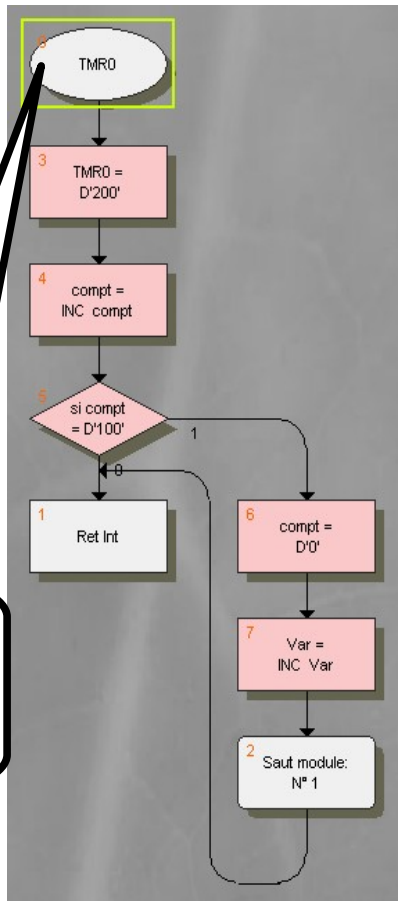
Si vous voulez ralentir le comptage sur l'afficheur, il n'est pas possible de diminuer le timer0, le seul moyen c'est d'effectuer un comptage dans le timer lui même.
Essayer le programme suivant :
N'oubliez pas de déclarer une nouvelle variable compt.

L'incrémentation de la variable Var ne s'effectuera qu'après 100 passages dans le timer, donc
100 x 0,65536 soit 6,5536 secondes.



Il faut 100 passages pour provoquer une incrémentation de la Var

TMR0 est positionné à 200 le comptage s'effectue entre 200 et 255 : c'est plus rapide.



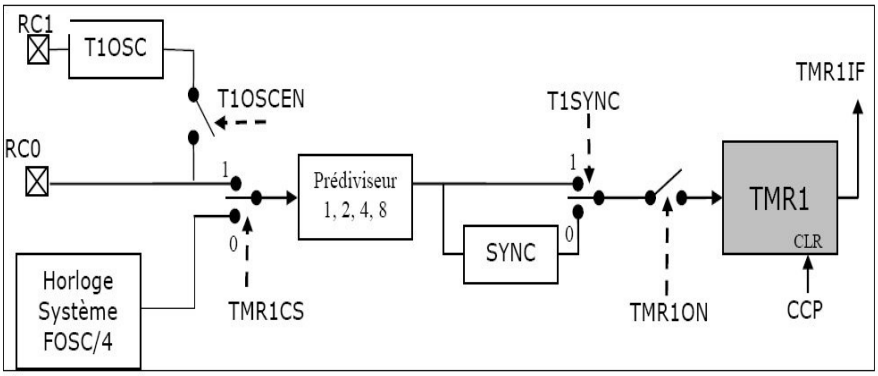
Il est possible aussi de modifier le temps en mettant une valeur dans la variable TMR0. En effet, le timer0 compte de 0 à 255 et provoque l'interruption au passage de 255 à 0. Si le timer0 (TMR0) débute à 200, le comptage ira beaucoup plus vite.

Malheureusement, le timer0 est toujours en marche, il serait très pratique de mettre en route le timer à la demande, nous allons voir que c'est possible avec le timer1.

3 - LE TIMER1

Le timer1 fonctionne globalement comme le timer0, cependant le timer1 est capable de compter sur 16bits ce qui lui permet d'être plus souple quand au choix du temps. Il permet aussi de fonctionner avec un autre quartz que celui utilisé par l'oscillateur du pic, cependant, ce cours n'abordera que le mode timer simple.
TMR1ON permet de mettre en route et d'arrêter très simplement ce timer.
La mise en œuvre du timer1 s'effectue grâce au registre **T1CON** :

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Non utilisé	**Non utilisé**	**T1CKPS1**	**T1CKPS0**	**T1OSCEN**	**T1SYNC**	**TMR1CS**	**TMR1ON**
0	**0**	**Calcul du prédiviseur: 1 1**		**0**	**0**	**0**	**0**



Après un reset le registre **T1CON** se positionne à 00000000

T1CKPS1 :
T1CKPS0 : } choix du prédiviseur dans notre exemple tous ces bits sont à 1.

T1OSCEN : permet d'activer un oscillateur interne (double quartz). 0 : normal. 1 : double quartz. On positionnera le bit3 à 0.

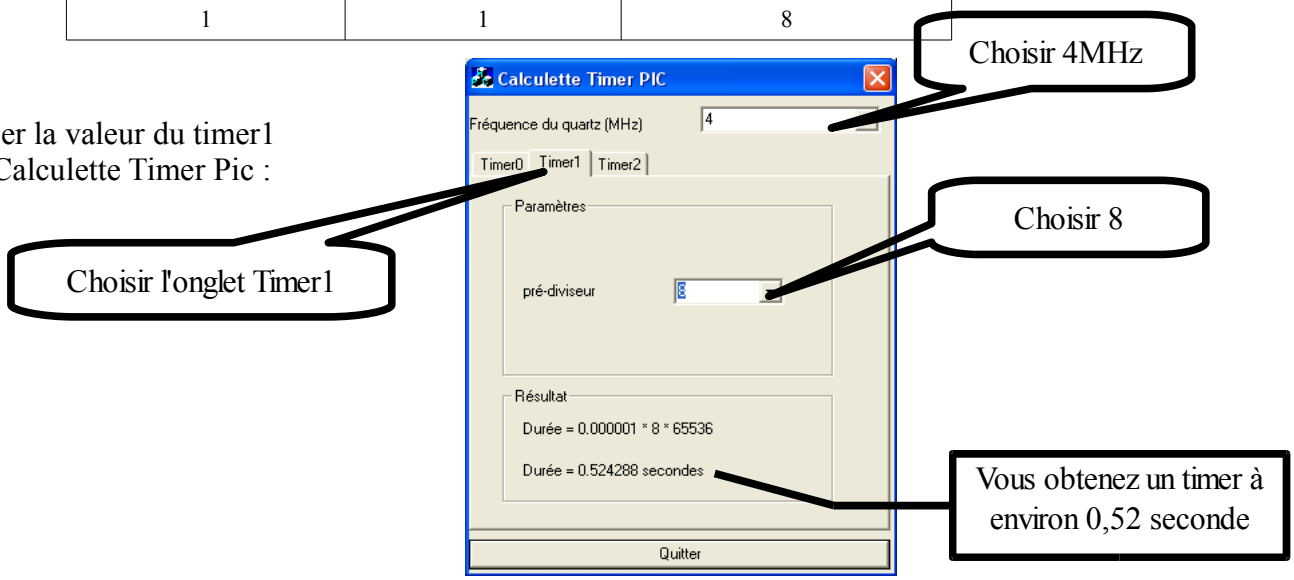
T1SYNC : dépend de TMR1CS. Quand TMR1CS est à 0, non utilisé. On laissera le bit2 à 0.

TMR1CS : permet de choisir entre l'horloge externe ou l'horloge interne.
0 : horloge interne. 1 : horloge externe. Il faudra donc positionner le bit1 à 0.

TMR1ON : mise en route du timer1. 0 : arrêt. 1 : marche. Dans un premier temps, on positionnera le bit0 à 0.

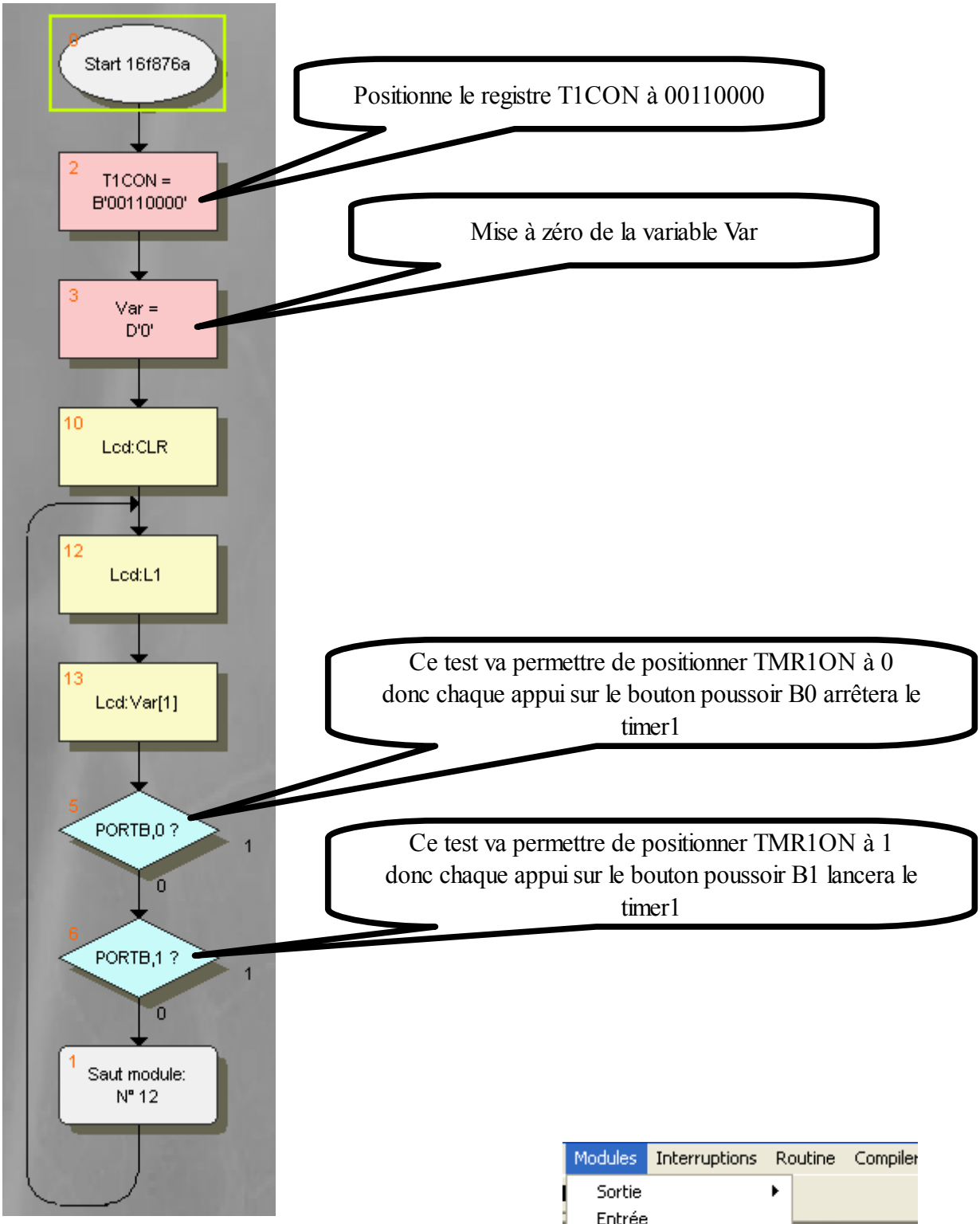
bit5 :T1CKPS1	**Bit4 : T1CKPS0**	/ Prédiviseur
0	0	1
0	1	2
1	0	4
1	1	8

Vérifier la valeur du timer1 avec Calculette Timer Pic :

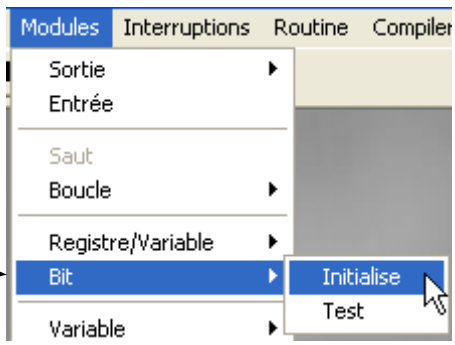


Il est temps de passer à la réalisation d'un programme exploitant le timer1 :

Lancer Logipic
Cliquer sur Nouveau puis sélectionner votre pic 16f876A
Enregistrer le projet sous timer1
Vous pouvez écrire le programme suivant :

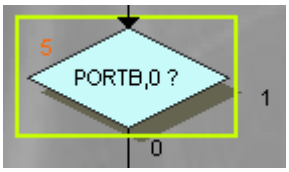


Pour modifier le bit TMR1ON, au lieu de réécrire complètement le registre T1CON, noua allons utiliser les possibilités de logipic avec la fonction :
Modules > Bit > Initialise

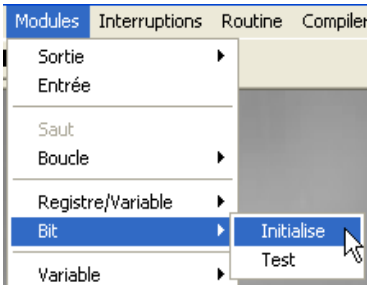


Ce module permet de modifier un seul bit sur un registre...

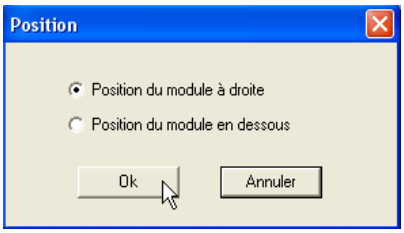
Placer vous sur le test PortB,0 ?



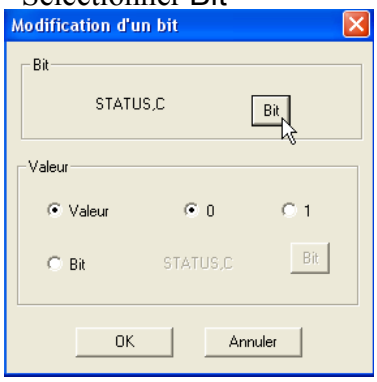
Puis



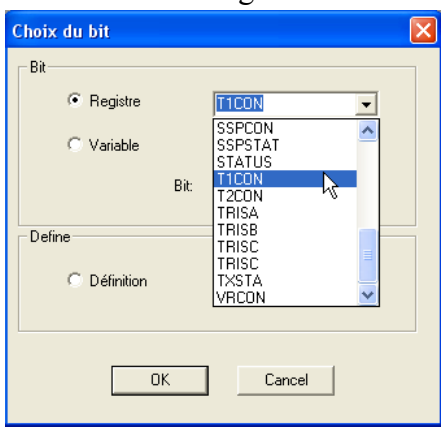
Demander un module à droite



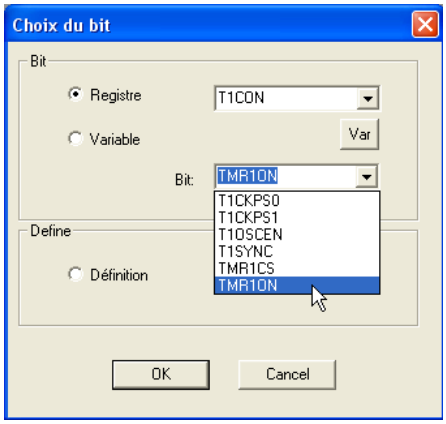
Sélectionner Bit



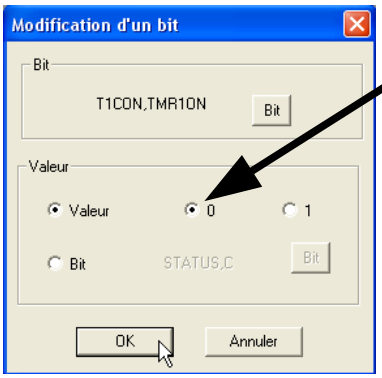
Sélectionner le registre T1CON



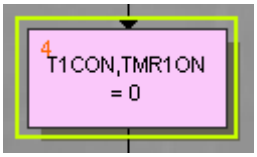
Sélectionner le bit TMR1ON



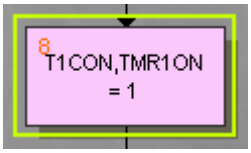
Et enfin terminer par le positionnement du bit à 0

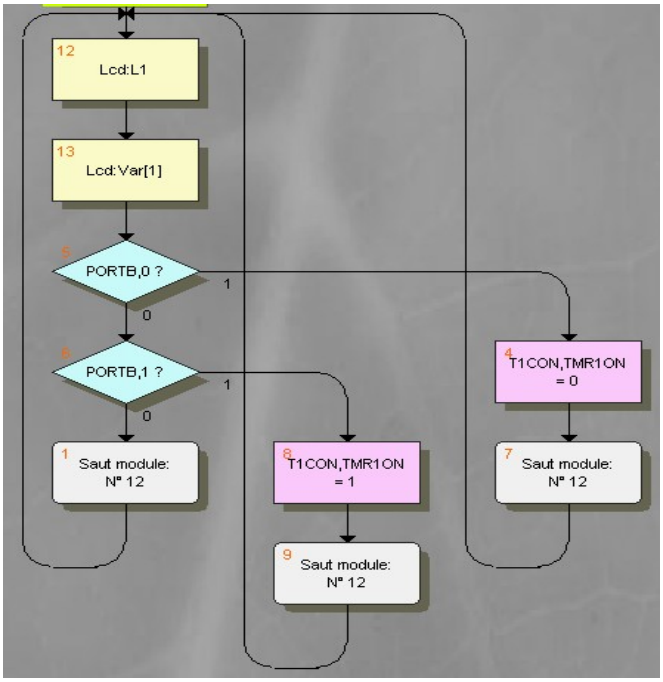


Vous obtenez le module suivant :



Faire de même avec le test PortB,1 ? Mais en positionnant le bit TMR1ON à 1 :



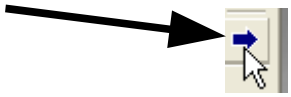


Reboucler le tout avec des sauts.

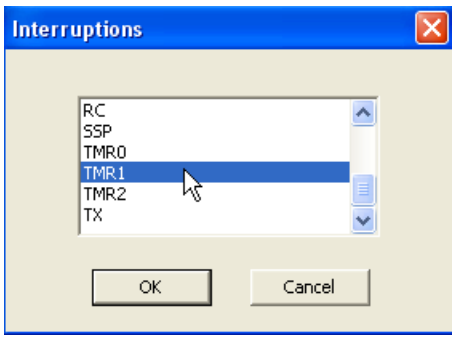
Récapitulons :
- chaque appui sur le bouton- poussoir B0 désactive le timer1.

- chaque appui sur le bouton- poussoir B1 active le timer1.

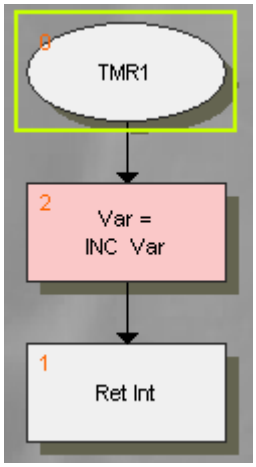
L'appel au timer1 s'effectue en demandant une interruption :



Choisir le Timer1



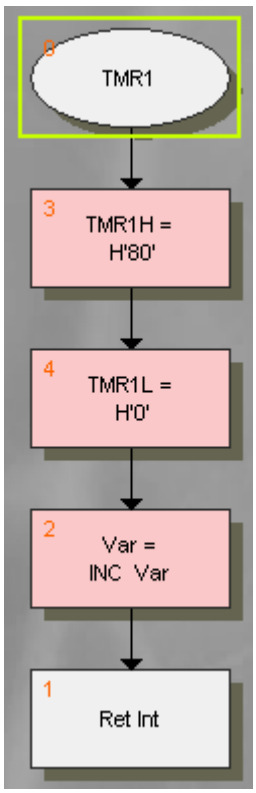
Ecrire le TMR1 (identique à l'exercice précédent)



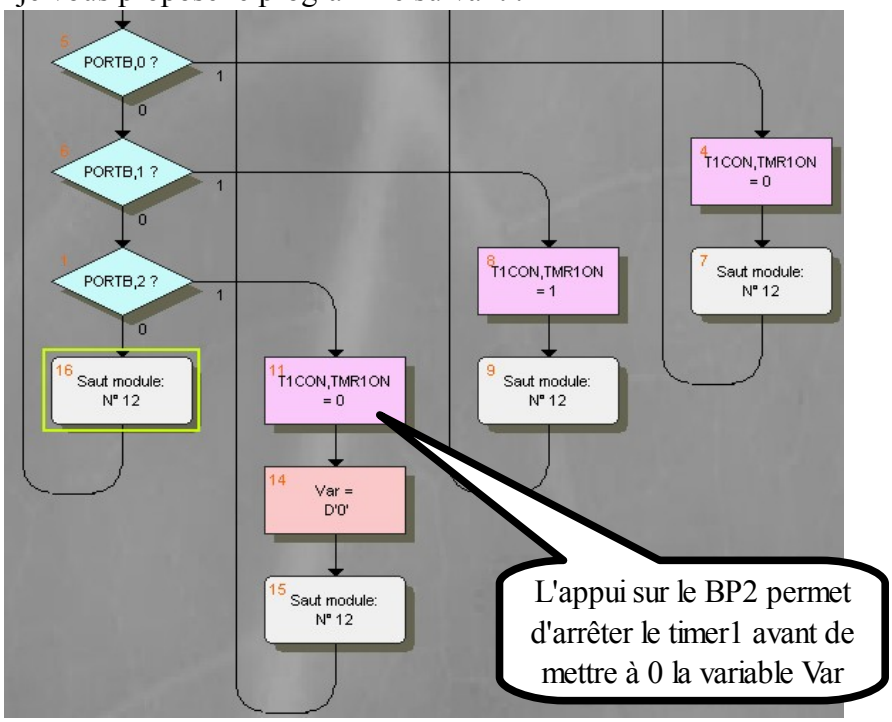
Grâce à Calculette Timer Pic, nous savons que l'appel s'effectue tous les 0,524288 seconde.

Vérifier les directives et les configurations de l'afficheur LCD.
Compiler et transférer le programme.
La mise en route du comptage s'effectue par appui sur B1, l'arrêt par B0.

Le timer0 effectue un comptage sur 8 bits (donc de 0 à 255), le timer1 travaille lui sur 16bits (donc de 0 à 65535), c'est pourquoi le timer0 et le timer1 à diviseur équivalent ont des temps différents...
Pour travailler sur 16 bits, timer1 dispose des variables TMR1H et TMR1L, cela permettra d'effectuer des comptages de façon très précise. L'exemple suivant permettra d'exploiter cette possibilité.
A partir du projet Timer1 exploitant un appel tous les 0,524288s, nous allons réaliser un appel tous les 0,262144s. Plutôt que de compter de 0 à 65535, il faudra compter en démarrant au milieu du compteur soit 65536/2=32768 or 32768 en décimal donne 10000000 00000000 en binaire ou 80 et 00 en héxa
On positionnera TMR1H à **80 en Héxa** et le TMR1L à 0.



Il serait intéressant de mettre à zéro l'affichage par l'intermédiaire d'un bouton-poussoir en B2 :
je vous propose le programme suivant :



Cependant; il difficile de travailler sur la valeur précise de 1 seconde si utile pour un chronomètre. Heureusement le timer2 nous permet cette possibilité.

4 - LE TIMER2

Timer2 est un compteur sur 8 bits. Cependant le timer2 dispose d'un prédiviseur, d'un postdiviseur et d'un ajustement permettant ainsi de disposer d'un très large éventail de diviseurs effectifs. La mise en œuvre du timer2 est très facile mais les calculs de temps doivent être rigoureux.
Nous pourrons ainsi réaliser un chronomètre précis au dixième de seconde...
La mise en œuvre du timer2 s'effectue grâce au registre **T2CON et au registre PR2**

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Non utilisé	**TOUTPS3**	**TOUTPS2**	**TOUTPS1**	**TOUTPS0**	**TMR2ON**	**T2CKPS1**	**T2CKPS0**
0	**/5 Calcul du postdiviseur: 0 1 0 0**				**0**	**/16 Calcul du prédiviseur: 1 0**	

TOUTPS3 :
TOUTPS2 :
TOUTPS1 :
TOUTPS0 : } choix du postdiviseur

Après un reset le registre **T2CON** se positionne à 00000000

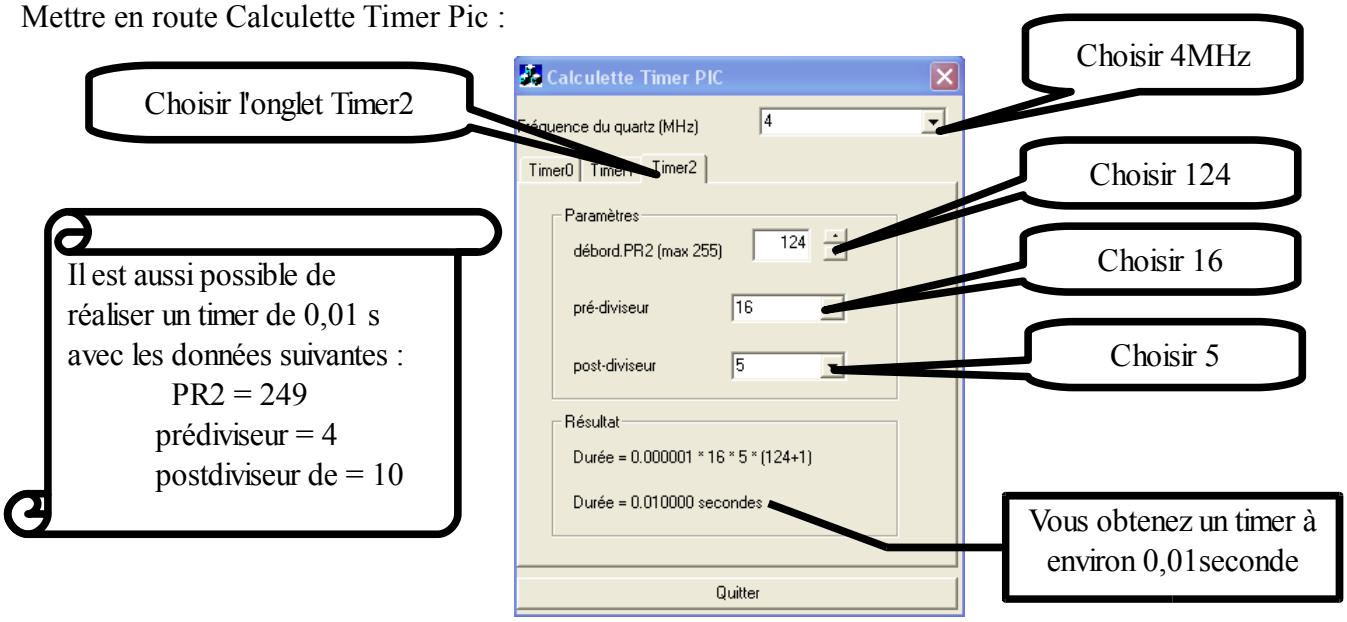
TMR2ON : mise en route du timer2. 0 : arrêt. 1 : marche. On positionnera le bit0 à 0

T2CKPS1 :
T2CKPS0 : } choix du prédiviseur

bit6 : TOUTPS3	bit5 : TOUTPS2	bit4 : TOUTPS1	bit3 : TOUTPS0	/ postdiviseur
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14
1	1	1	0	15
1	1	1	1	16

bit1 : T2CKPS1	bit0 : T2CKPS0	/ Prédiviseur
0	0	1
0	1	4
1	0	16

Mettre en route Calculette Timer Pic :



Nous avons toutes les données pour réaliser un chronomètre sur le le timer2.
Pour effectuer un comptage à la seconde, il sera indispensable de réaliser 100 boucles dans le timer2 à 0,01 seconde...
Les explications d'un chronomètre sont réalisés dans le coach timer. Je vous renvoie donc à ce dossier.
Ceci explique le début de programme pour réaliser un chronomètre :