LE TIMER 1

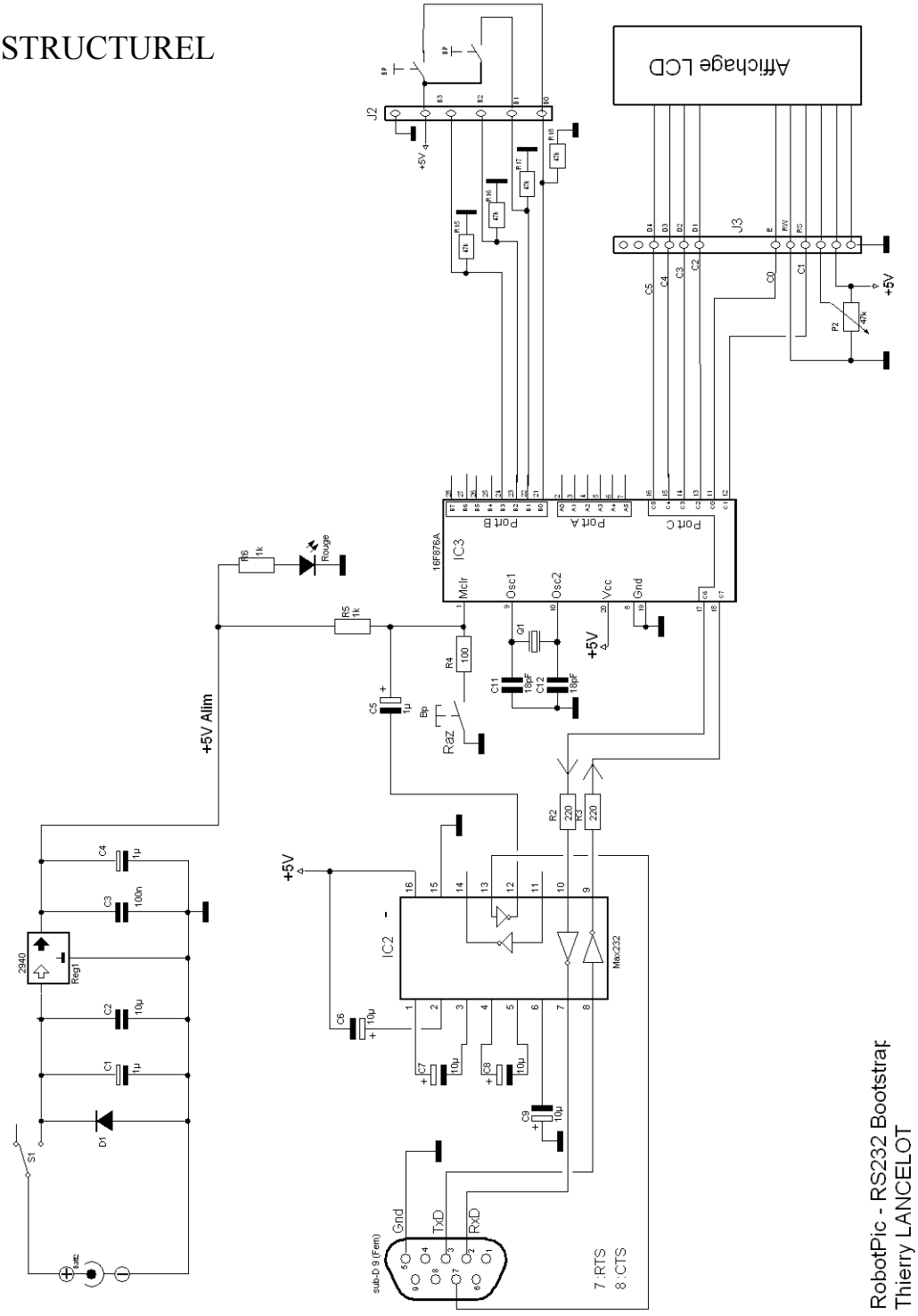
Objectif : Il s'agit de comprendre et d'utiliser le timer des microcontroleurs type 16F. Ce coach complète le coach sur les timers

1 - La cible

Les programmes sont réalisés sur la platine BootRS232. Cette cible est constituée par un microcontroleur 16F876A avec un oscillateur interne à 20MHz. Pour fonctionner en mode bootstrap, le pic devra être programmé une première fois avec un bootloader (16F876A_20M.hex).

Avec un quartz à 20 MHz, il faut utiliser Logipic V213, en effet, la prise en charge de l'affichage LCD avec un quartz à 20 MHz ne fonctionnera pas avec les versions antérieures à V209.

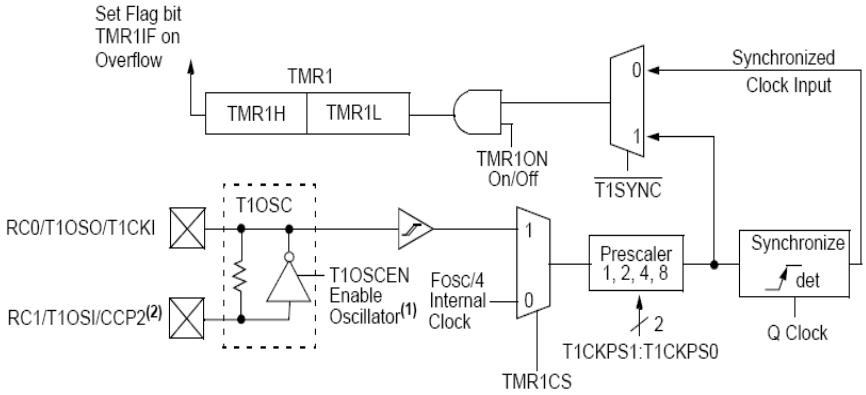
SCHEMA STRUCTUREL



2 - Le timer1

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**



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

T1CKPS1 :
T1CKPS0 : } choix du prédiviseur

T1OSCEN : permet d'activer un oscillateur interne (double quartz). 0 : normal. 1 : double quartz.

T1SYNC : dépend de TMR1CS. Permet la synchronisation. 0 : synchro. 1 : pas de synchro.

Si **TMR1CS** = 0 (horloge interne) **T1SYNC** est ignoré.

TMR1CS : permet de choisir entre l'horloge externe ou l'horloge interne.

0 : horloge interne. 1 : horloge externe, double quartz.

TMR1ON : mise en route du timer1. 0 : arrêt. 1 : marche.

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

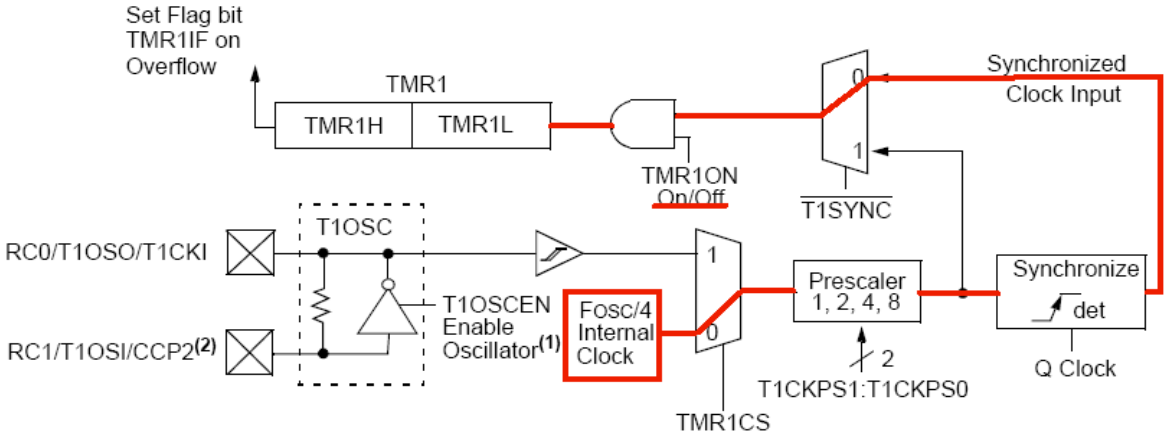
2 - Fonctionnement en mode timer

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.
Pour travailler sur 16 bits, il est nécessaire d'utiliser 2 registres : TMR1L et TMRH.
Attention, le contenu de TMR1L et TMRH n'est pas remis à 0 lors d'un reset. Il faudra donc toujours charger ces registres lors de l'écriture du programme.
Pour choisir ce mode, nous devons configurer T1CON de la façon suivante :

bit7	bit6	**T1CKPS1**	**T1CKPS0**	**T1OSCEN**	**T1SYNC**	**TMR1CS**	bit0
0	**0**	**Choix du prédiviseur**		**0**	**0**	**0**	**TMR1ON**

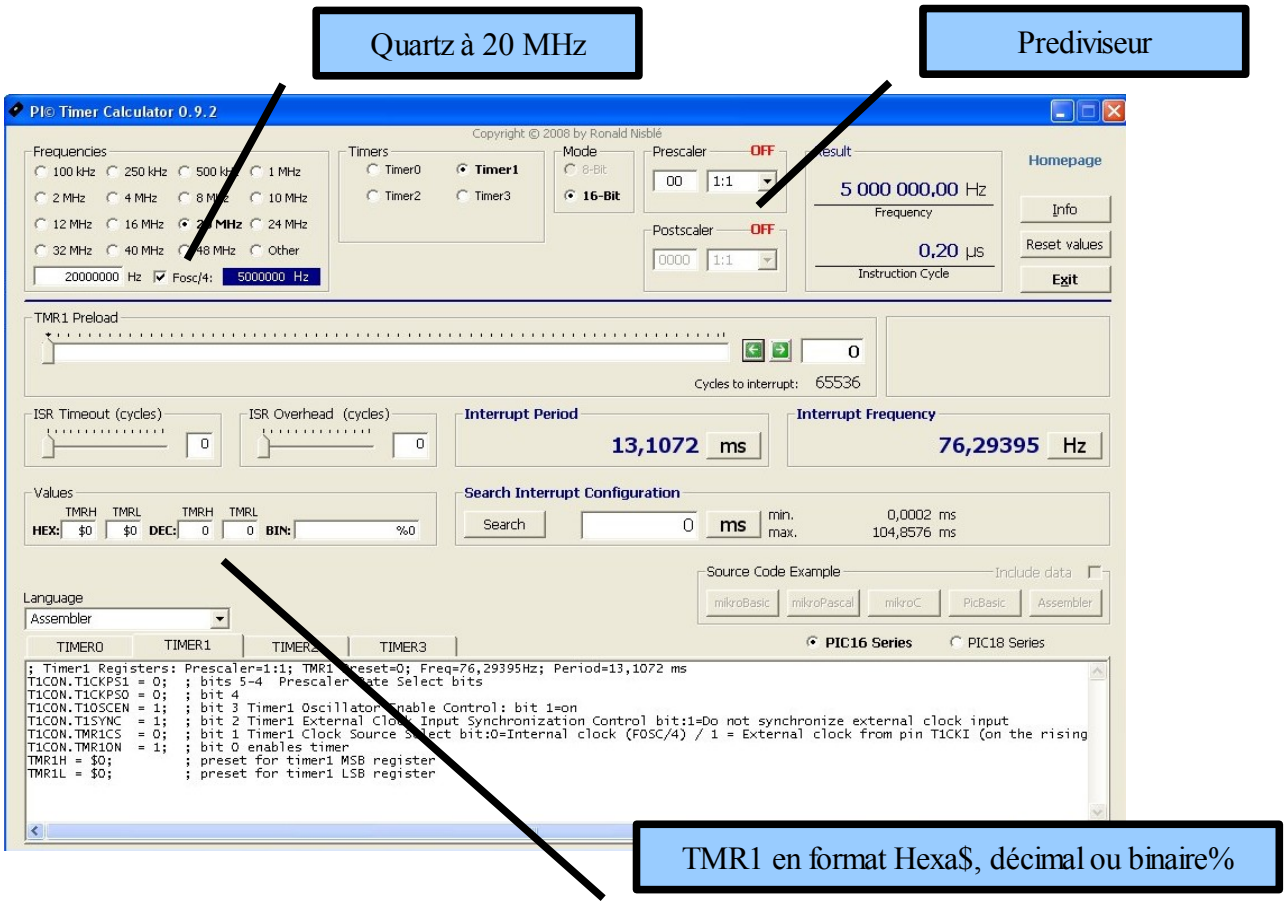
T1CON = B '00xx000x x pourra prendre la valeur 0 ou 1

Dans ce cas là, le timer1 se résume à :



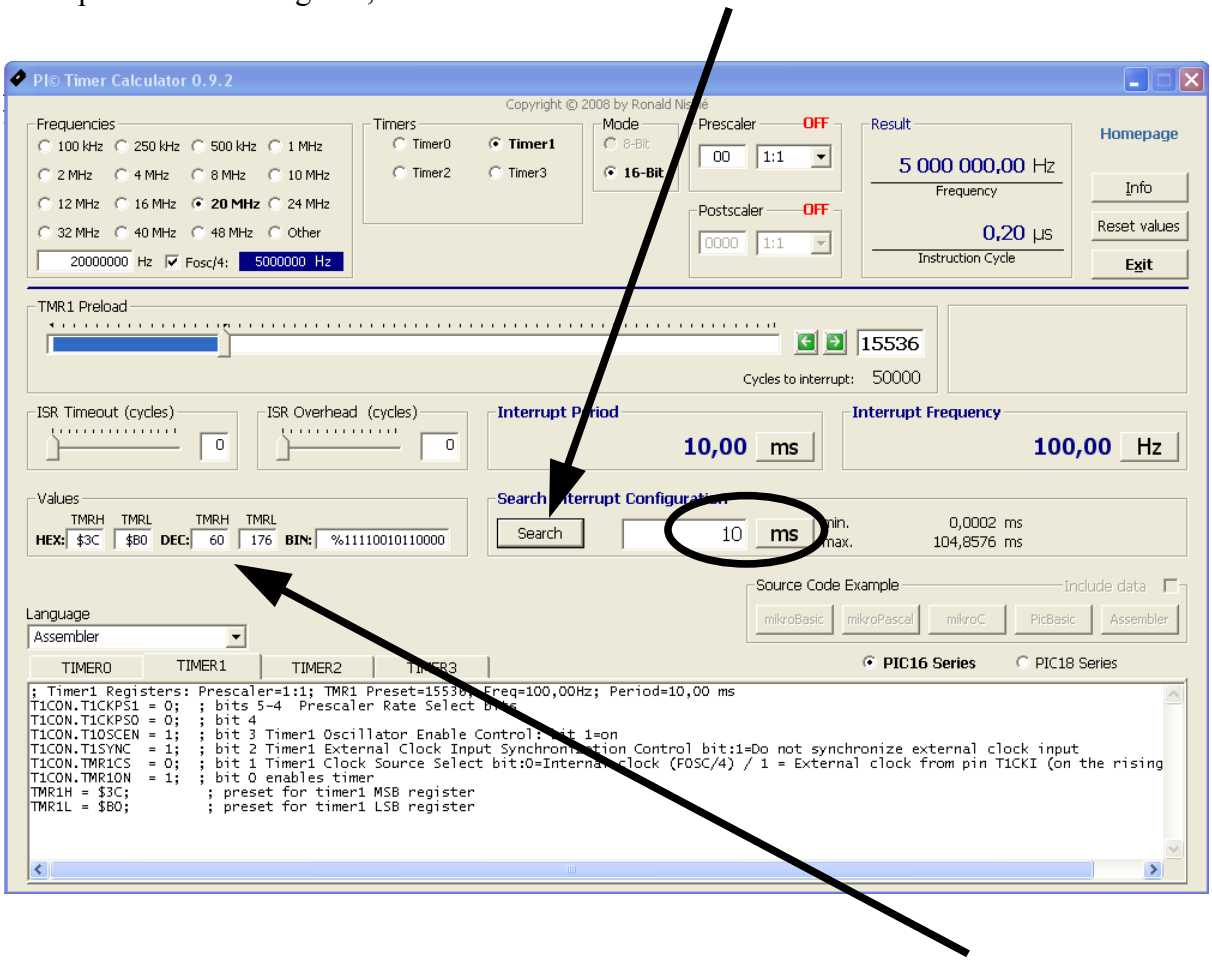
A partir du moment ou TMR1ON est à 1, l'appel de l'interruption dépend de la fréquence du Quartz, du prédiviseur et du comptage de TMR1.
Exemple1 : si le quartz est de 20MHz, le prédiviseur à 1 et le TMR1 à 0, il va falloir passer 65536 fois pour provoquer une interruption, soit une fréquence de Q/4/65536 = 76,2939 Hz ou 13,1072ms
Exemple2 : si le quartz est de 20MHz, le prédiviseur à 1 et le TMR1 à FFFB, il va falloir passer 5 fois pour provoquer une interruption, soit une fréquence de Q/4/5 = 1 MHz ou 1 µs

Pour connaître les différentes valeurs de T1CKPS1, T1CKPS0 TMR1H et TMR1L en fonction du temps désiré, j'utilise le logiciel Pic Timer Calculator.



Pic timer Calculator indique ainsi qu'avec un quartz à 20 MHz, le prédiviseur positionné à /1 et TMR1 à H'0000', l'interruption est effectué tous les 13,1072ms.

Pour réaliser un compteur capable de travailler à la seconde, il va falloir trouver une valeur entière, utilisons les possibilités du logiciel, demander 10 ms et demander Search :



Le calcul indique un prédiviseur à /1 (T1CKPS1 et T1CKPS0 à 0), TMR1H à 60 et TMR1L à 176.
Avec une interruption toutes les 10ms, il sera très facile d'effectuer un comptage indiquant la seconde.

Il est temps de passer sur Logipic.

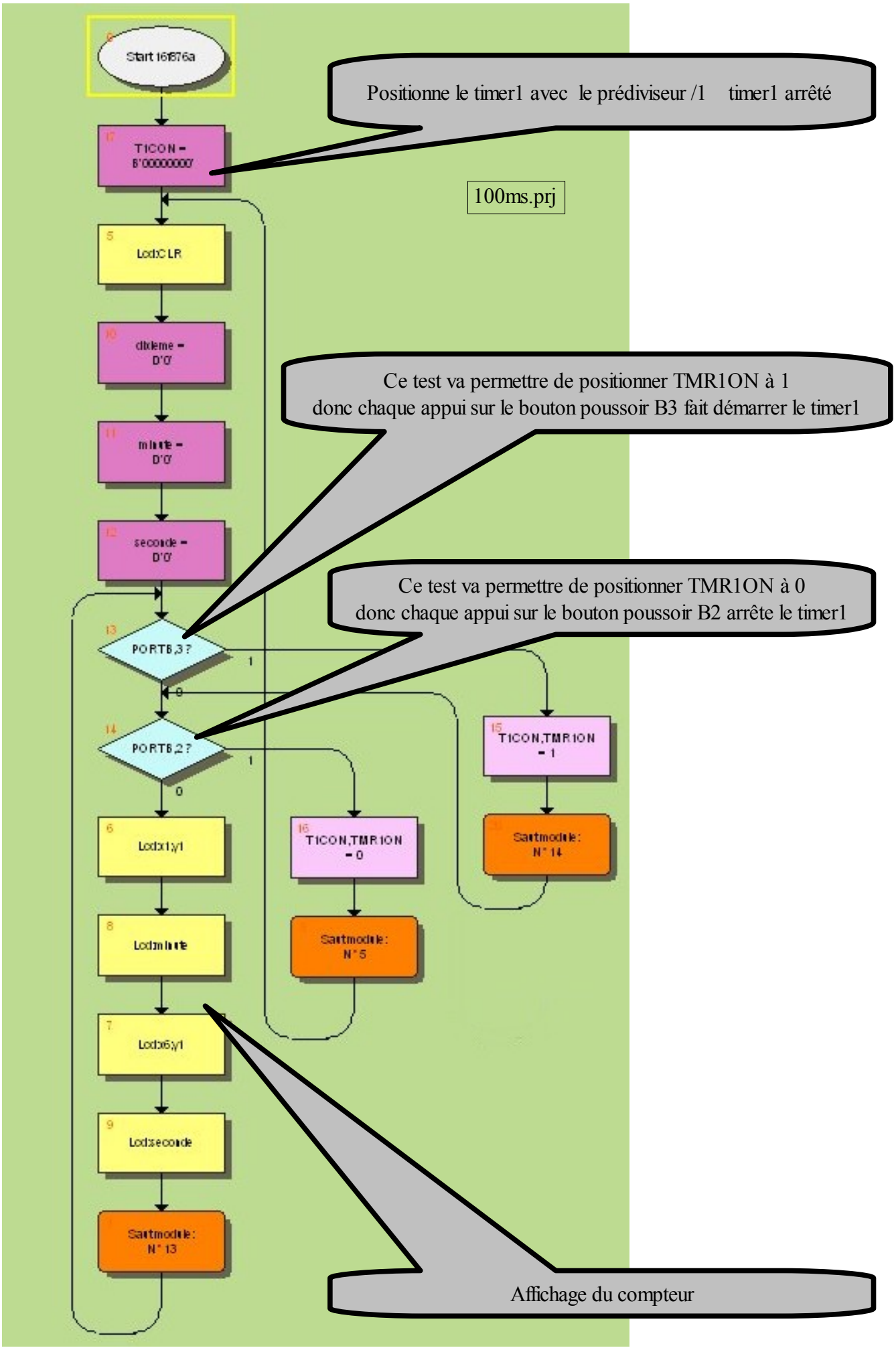
Le programme principal permet de gérer la mise en route du timer1.
Dès la mise en route du timer1 l'appel de l'interruption s'effectue toutes les 100ms, un comptage jusqu'à 10 permettra donc de faire avancer un compteur de seconde.
Le sous-programme d'interruption ne prend pas en compte le dépassement au bout de 60 minute, l'intérêt étant essentiellement de comprendre la gestion du timer1.

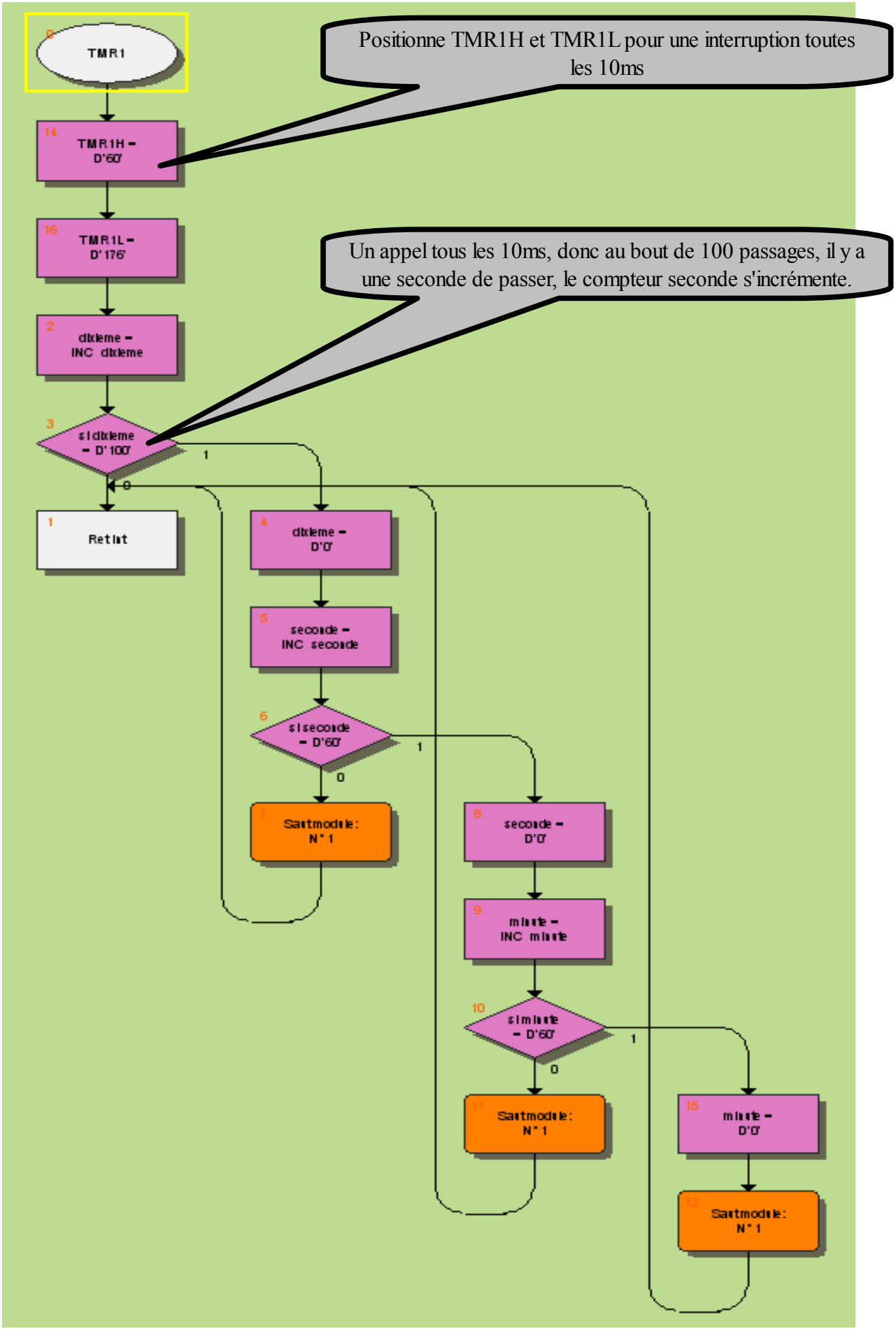
Avant de compiler le projet, n'oubliez pas de configurer correctement les options nécessaire à la cible et tout particulièrement le choix de l'oscillateur interne (HS pour un Quartz à 20 MHz). Il est nécessaire aussi de configurer l'option LCD pour être compatible avec la cible.
Après compilation, le fichier peut-être transférer sur la cible avec 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 (115200 pour un Quartz à 20 MHz).

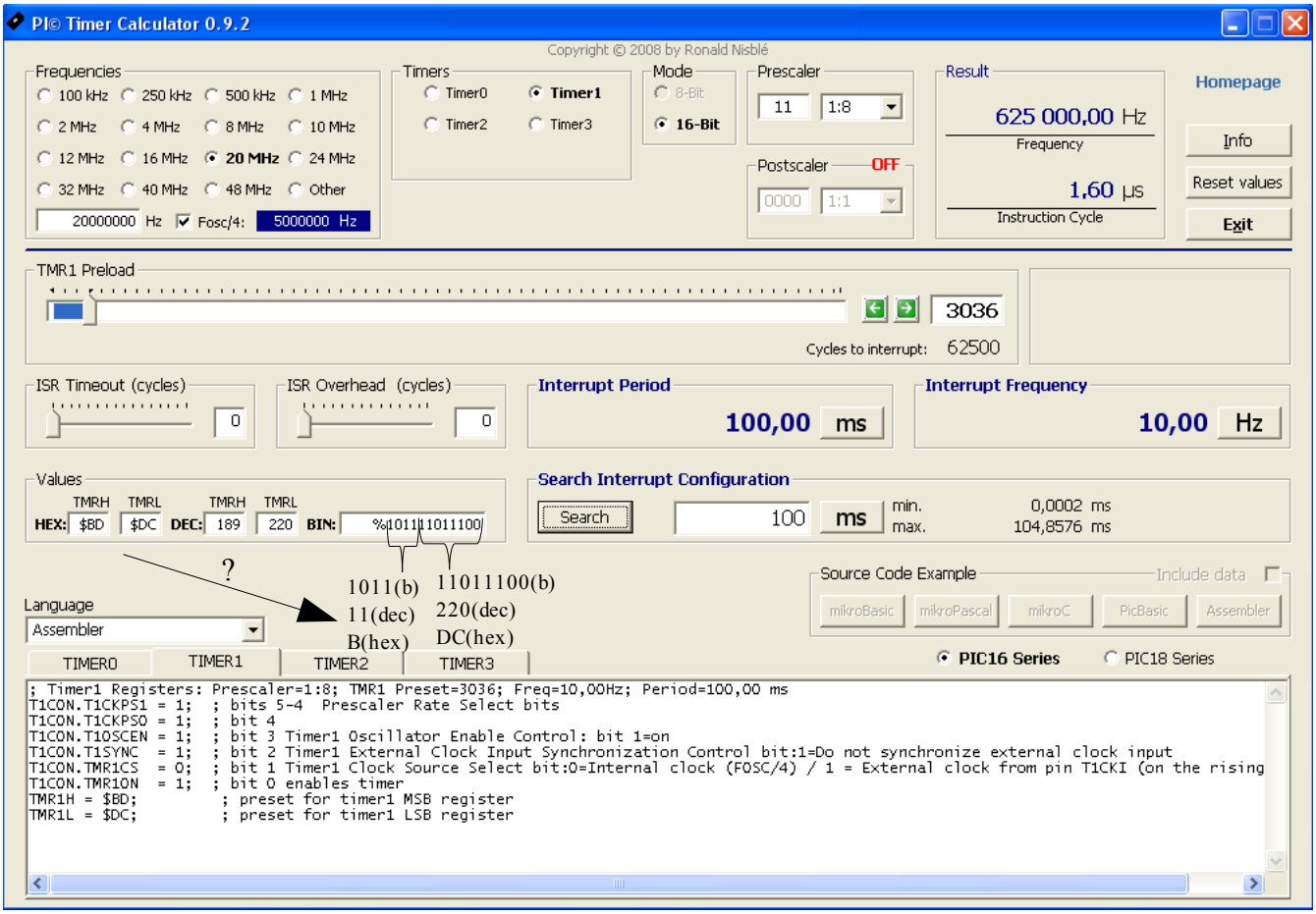
Attention de ne pas utiliser des temporisations lors des appels des timers, j'ai remarqué de nombreux problèmes . Les temporisations du programme provoquent des fonctionnement bizarres du timer. En effet, un appel timer tous les 10ms alors qu'il y a une tempo de 20ms n'est pas compatible.

Les calculs effectués avec pic calculator doivent être vérifiés par mesure sur le pic en fonctionnement, vérifier au moins si la valeur bianire est égale à la valeur héxa ou décimale (ce coach explique tout ça).


Start 16f876a
TICON = B'00000000'
Positionne le timer1 avec le prédiviseur /1 timer1 arrêté
100ms.prj
LcdCLR
dixieme = D'0'
minute = D'0'
seconde = D'0'
PORTB,3 ?
1
0
Ce test va permettre de positionner TMR1ON à 1
donc chaque appui sur le bouton poussoir B3 fait démarrer le timer1
PORTB,2 ?
1
0
Ce test va permettre de positionner TMR1ON à 0
donc chaque appui sur le bouton poussoir B2 arrête le timer1
TICON,TMR1ON = 1
Saut module: N° 14
Lcdx1y1
TICON,TMR1ON = 0
Saut module: N° 5
Lcdminute
Lcdx6y1
Lcdseconde
Saut module: N° 13
Affichage du compteur


0
TMR1
Positionne TMR1H et TMR1L pour une interruption toutes les 10ms
14
TMR1H = D'60'
16
TMR1L = D'176'
Un appel tous les 10ms, donc au bout de 100 passages, il y a une seconde de passer, le compteur seconde s'incrémente.
2
dixieme = INC dixieme
3
si dixieme = D'100'
1
0
1
Retint
4
dixieme = D'0'
5
seconde = INC seconde
6
si seconde = D'60'
1
0
7
Sattmodule: N°1
8
seconde = D'0'
9
minute = INC minute
10
si minute = D'60'
1
0
11
Sattmodule: N°1
15
minute = D'0'
13
Sattmodule: N°1

Pic timer calculator est d'une aide précieuse, cependant dans certains cas, il faut vérifier les informations entre les valeurs binaires et hexa/décimale.
Exemple :
Concernant une recherche sur le timer 100ms :



Les bonnes valeurs sont TMRH = 11(dec) ou B(hex): je ne sais pas pourquoi il y a ce problème ?

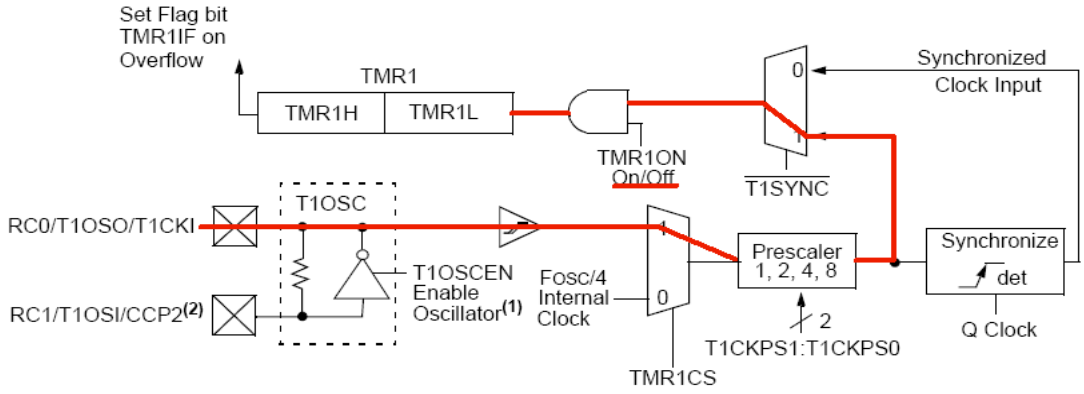
3 - Fonctionnement en mode compteur asynchrone

Le signal sur la borne T1CKPI est utilisé pour incrémenter le compteur TMR1 au lieu de l'horloge interne
Pour choisir ce mode, nous devons configurer T1CON de la façon suivante :

bit7	bit6	**T1CKPS1**	**T1CKPS0**	**T1OSCEN**	**T1SYNC**	**TMR1CS**	bit0
0	**0**	**Choix du prédiviseur**		**0**	**1**	**1**	**TMR1ON**

T1CON = B '00xx011x x pourra prendre la valeur 0 ou 1

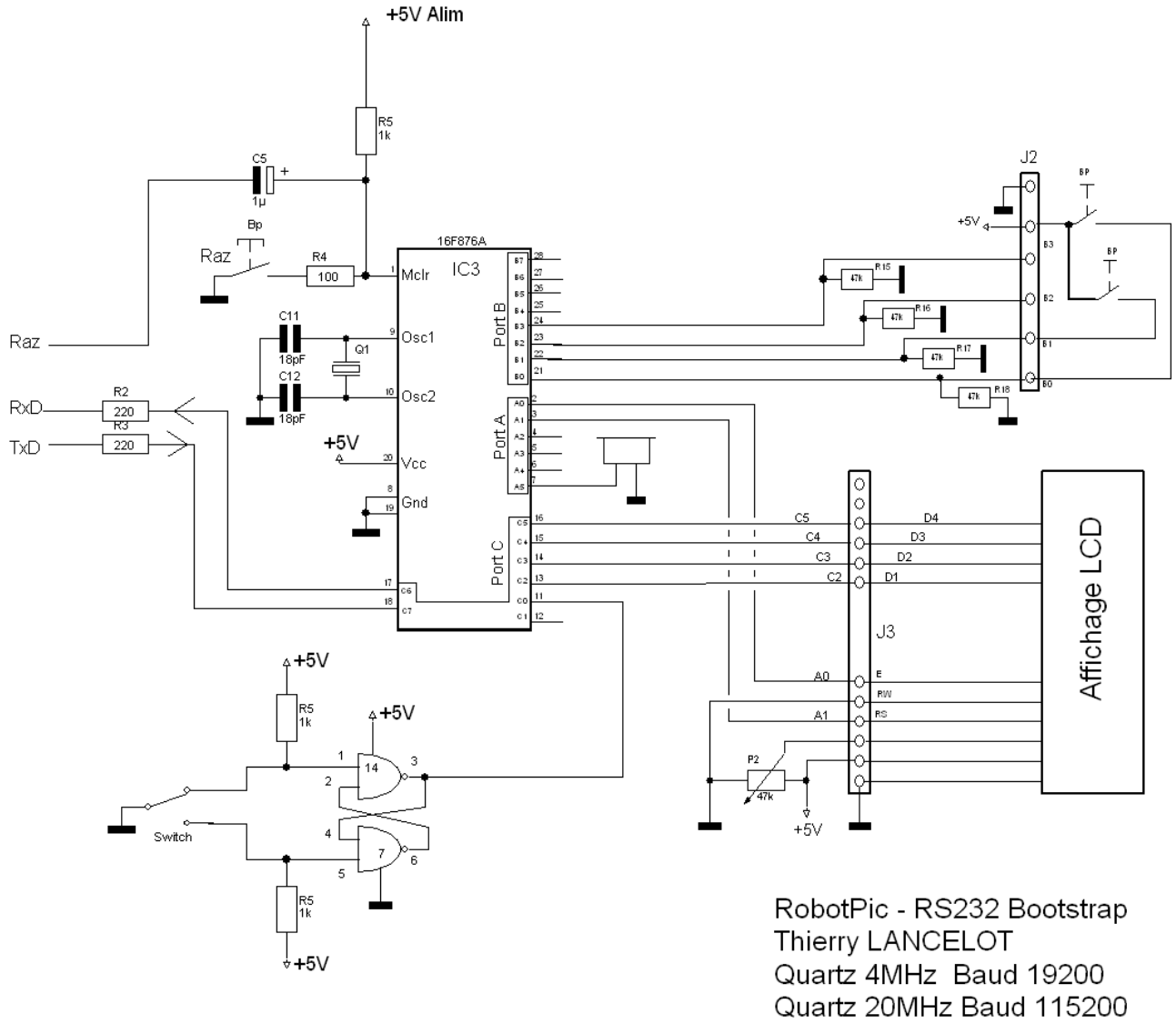
Dans ce cas là, le timer1 se résume à :



Ce mode permet de compter les évènements qui se présentent sur la pin RC0/T1CKI, (pin11 d'un 16F876).
Pour pouvoir compter les évènements sur la borne T1CKPI, il faut que cette pin soit configurée en entrée via le bit 0 du TRISC (donc ce bit doit être à 1).

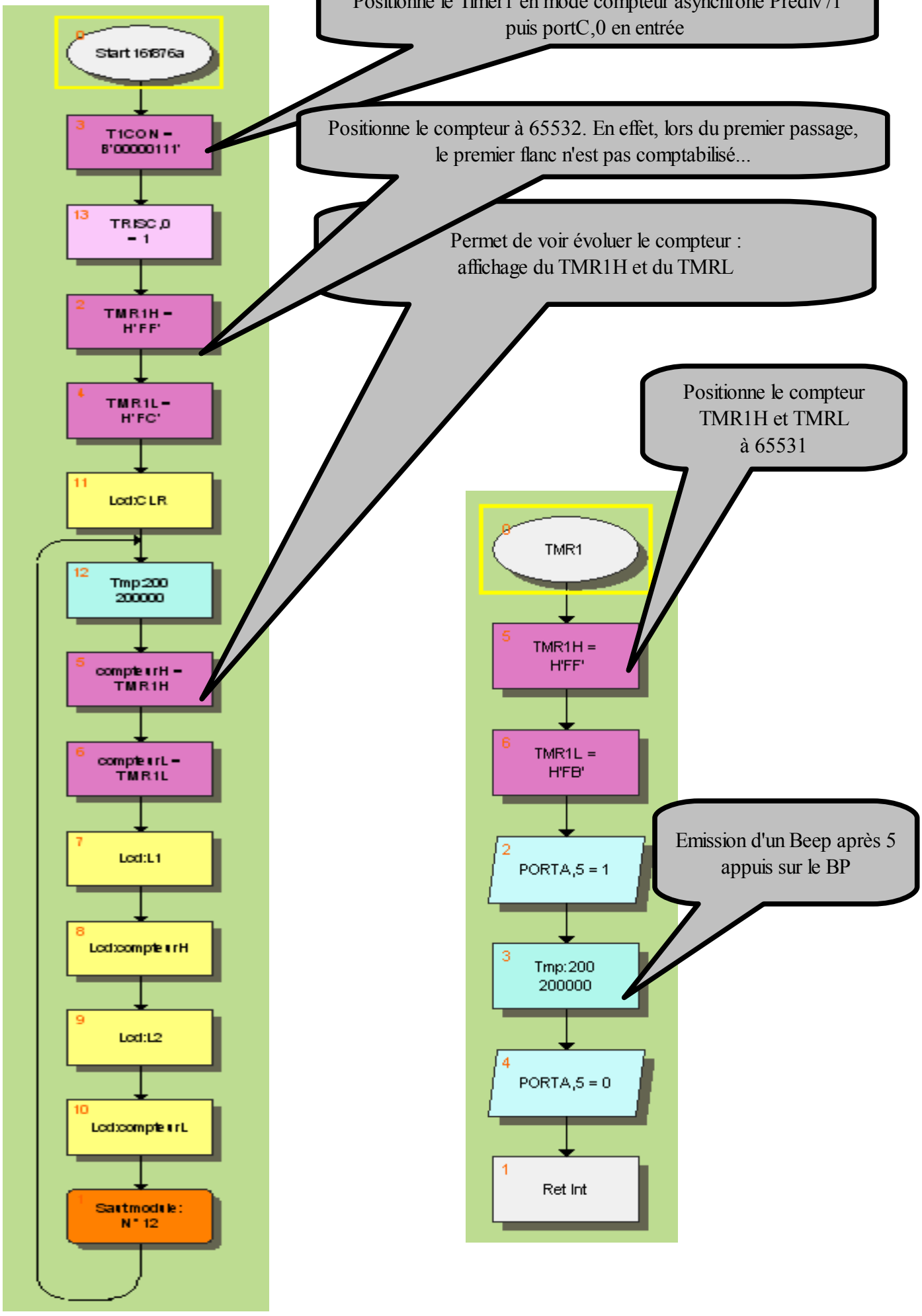
Le comptage s'effectue uniquement sur le front montant. Le flanc montant de T1CKI n'est pris en compte que s'il est précédé d'un flanc descendant donc si le signal est au niveau 0 à la mise en route du pic, le premier flanc montant ne sera pas comptabilisé.
Dans cette configuration, la pin RC1 n'est pas utilisée.

Je vous propose d'utiliser ce mode pour réaliser un programme permettant d'actionner le beeper lorsque l'on appui 5 fois sur un bouton-poussoir. En théorie un bouton-poussoir permet de créer des impulsions de comptage, malheureusement, les effets de rebond ne permettent pas un fonctionnement réel. Pour éliminer ses rebonds, j'utilise un montage simple anti-rebond avec un circuit TTL NAND 7437.
De plus la platine indiquée en page 1 devra être modifiée afin de disposer de RC0 et RC1. Il seront remplacé par A0 et A1 pour l'afficheur LCD.
Le schéma sera donc le suivant :



Le compteur étant sur 16 bits, pour déborder à 65535, il faudra donc le positionner à 65536-5 = 65531
65531(10) = 11111111 11111011 (2)
TMR1H TMR1L
Soit TMRH1 = FF(hexa) TMR1L = FB(hexa) ou 251 (dec)

Programme principal :



Remarquer le fonctionnement du programme :
A la mise en route, le compteur est positionné à 65532, cela permet de prendre en compte la mise en route du compteur lors du premier appui ensuite il ne restera plus qu'un comptage sur 5 : cela fait donc 5 appuis...d'ailleurs l'affichage permet de visualiser ce qui se passe, le premier appui après la mise en route ne décrémente pas le compteur. Si l'on modifie la valeur du prédiviseur par exemple par /2, il faudra appuyer 10 fois pour provoquer le Beep.

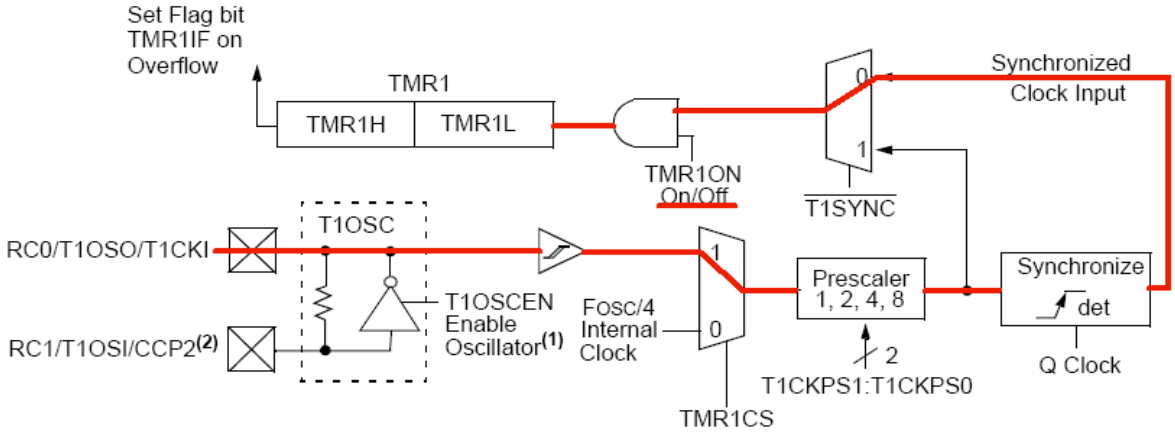
4 - Fonctionnement en mode compteur synchrone

Le fonctionnement est quasiment identique, cependant, la mise à 0 du bit **T1SYNC** force le compteur à se synchroniser sur l'horloge interne.
Pour choisir ce mode, nous devons configurer T1CON de la façon suivante :

bit7	bit6	**T1CKPS1**	**T1CKPS0**	**T1OSCEN**	**T1SYNC**	**TMR1CS**	bit0
0	**0**	**Choix du prédiviseur**		**0**	**0**	**1**	**TMR1ON**

T1CON = B '00xx001x x pourra prendre la valeur 0 ou 1

Dans ce cas là, le timer1 se résume à :

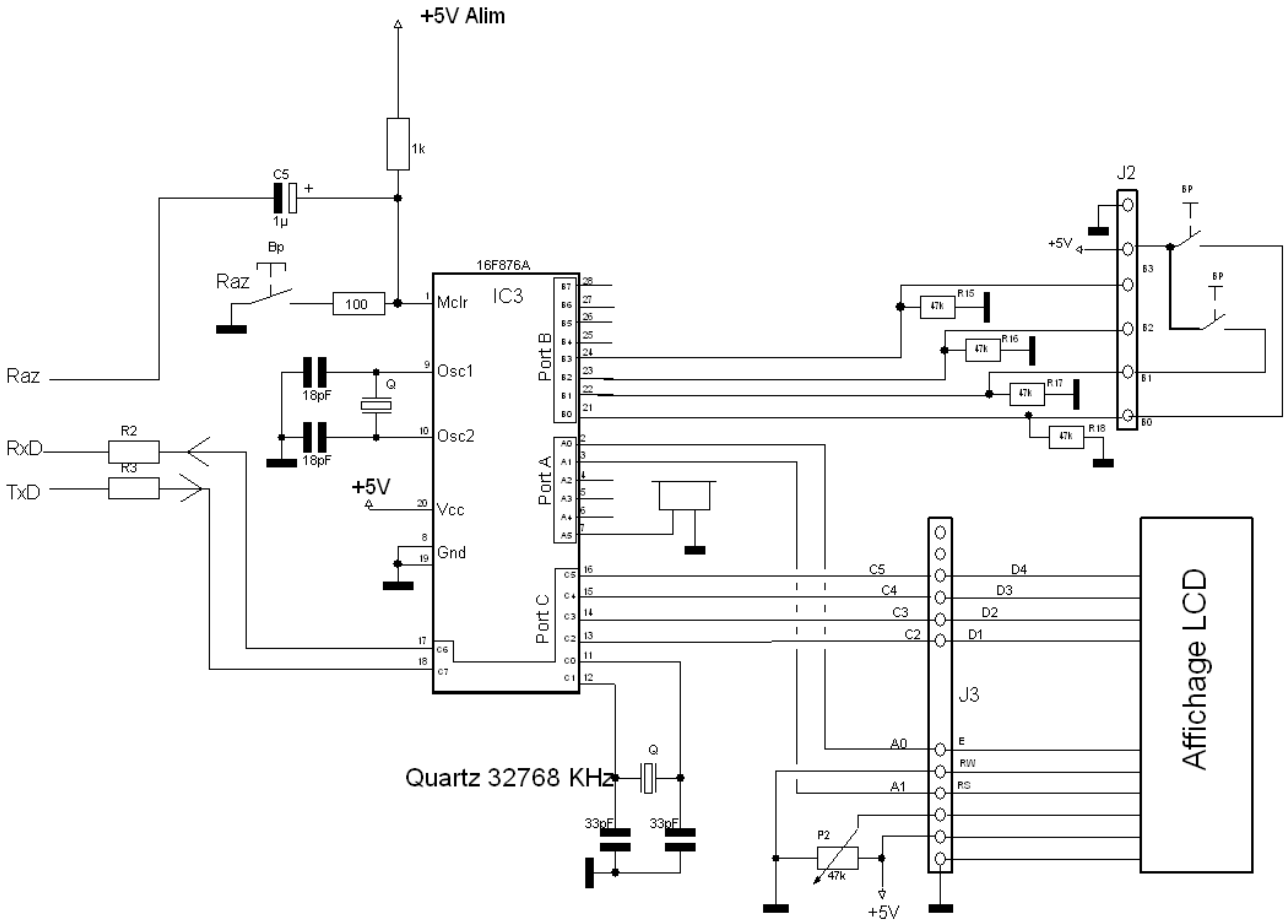


La présence du synchronisateur retarde la prise en compte de l'évènement, le flanc montant sera transmis uniquement au flanc montant de l'horloge interne
La synchronisation permet de compter uniquement quand le pic est en fonctionnement, c'est à dire que le pic ne doit pas être en mode « sleep ».
concernant le montage d'essai et vu la vitesse de comptage cela ne change pas grand chose...

Ici s'arrête mes compétences et pour plus d'informations, je vous invite à lire le datasheet du pic ou l'excellent ouvrage de Bigonoff.

5 - Fonctionnement en mode double quartz

Il est utile d'utiliser une base de temps différente de celle du quartz du pic. En effet, ceci permet d'utiliser une base de temps multiple de la seconde (quartz horloge à 32768Khz) tout en conservant une vitesse maximale pour le fonctionnement du pic soit 20 MHz.
Dans ce mode, l'oscillateur fonctionne avec une fréquence plus faible que le quartz du pic.
L'oscillateur secondaire est prévue pour fonctionner sur une Fréquence de l'ordre de 32KHz, cependant cela doit fonctionner jusqu'à 200 KHz (vérifier le datasheet pour la valeur des condensateurs entourant le quartz).

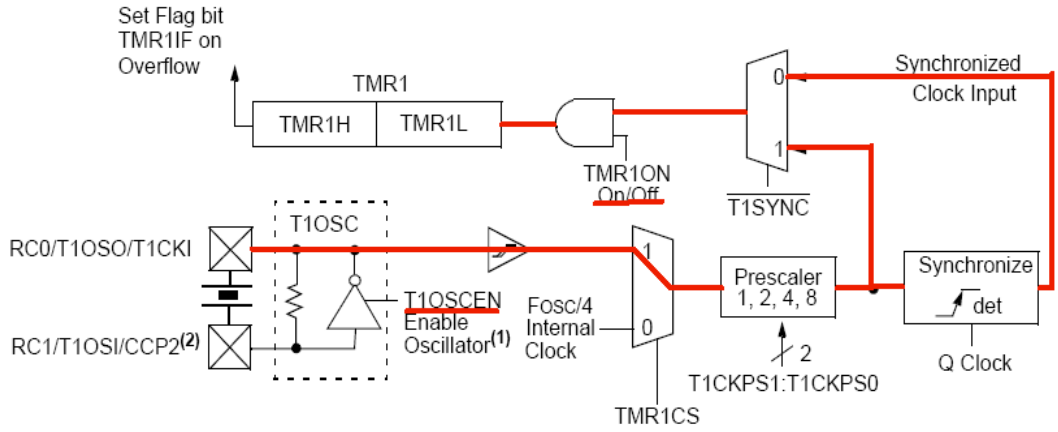


RobotPic - RS232 Bootstrap
Thierry LANCELOT
Quartz 4MHz Baud 19200
Quartz 20MHz Baud 115200

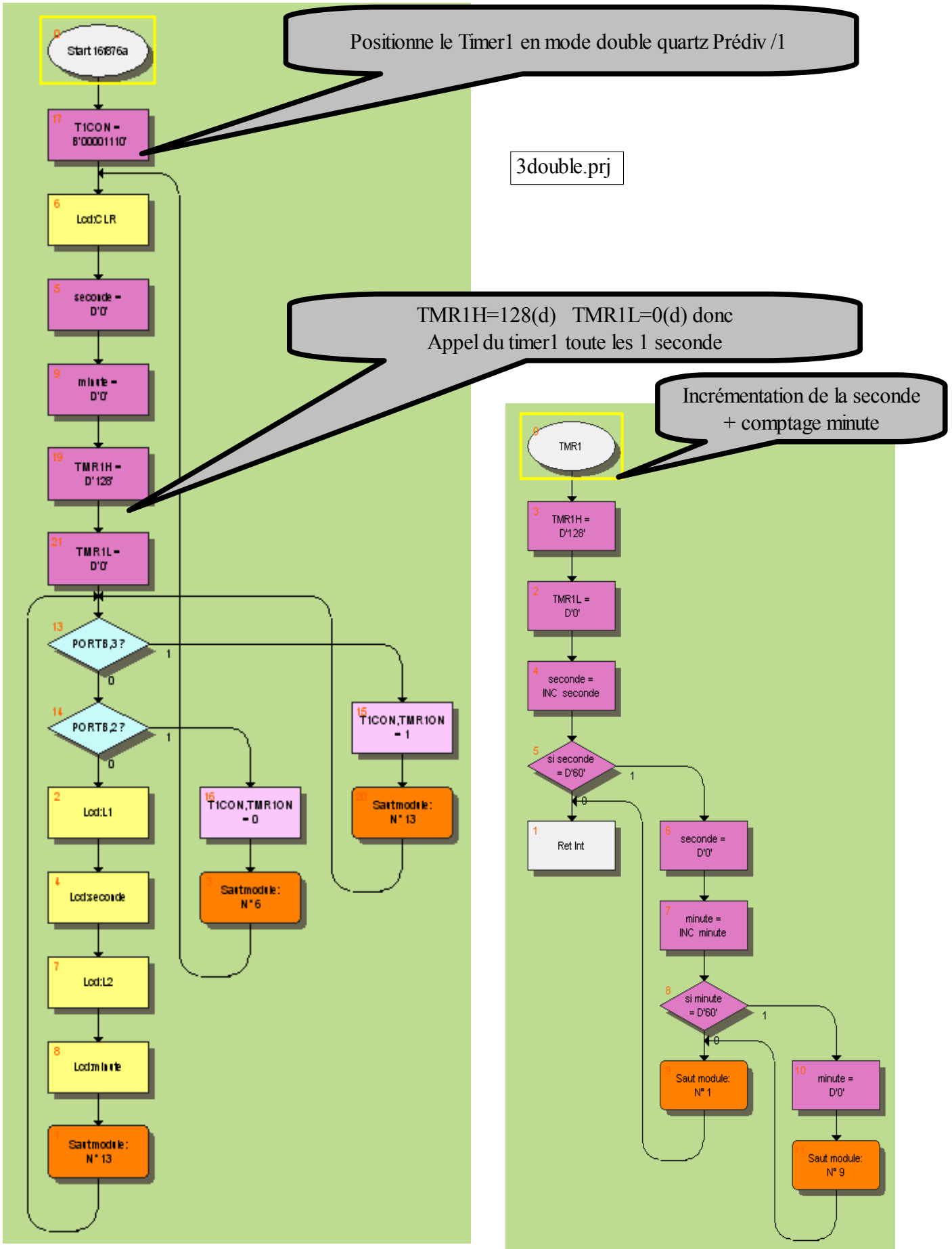
Dans ce mode particulier, les pins RC0 et RC1 sont automatiquement configurées en entrée, inutile de configurer TRISC.
Pour choisir ce mode, nous devons configurer T1CON de la façon suivante :

bit7	bit6	**T1CKPS1**	**T1CKPS0**	**T1OSCEN**	bit0	**TMR1CS**	bit0
0	**0**	**Choix du prédiviseur**		**1**	**T1SYNC**	**1**	**TMR1ON**

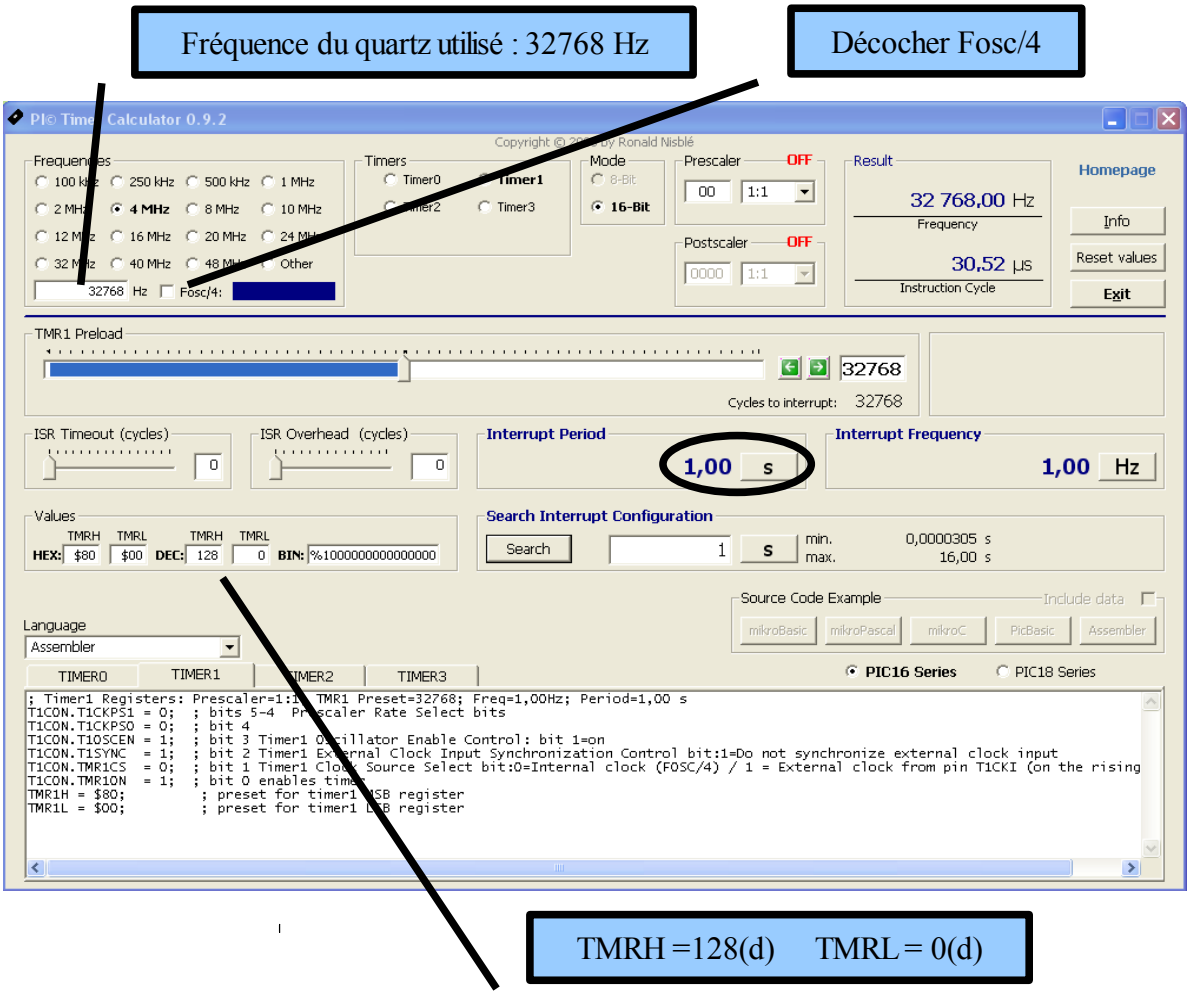
T1CON = B '00xx1x1x x pourra prendre la valeur 0 ou 1



Dans ce mode, la synchronisation permet principalement de prendre en compte ou non le mode « sleep » du pic. Dans l'exemple qui suit, je positionne T1SYNC à 1.



Pour le calcul du timer n'oubliez pas de configurer correctement Pic Timer :



6 - Exemple d'utilisation : un fréquencemètre

Pour réaliser un fréquencemètre, il suffit de compter le nombre de fois où l'entrée RC0 passe de l'état haut à l'état bas pendant un temps donnée. Il faudra faire fonctionner le timer 1 en mode compteur synchrone ou asynchrone.
Si le signal à mesurer est de 1KHz et que l'on effectue un comptage de ce signal pendant 1 seconde, le compteur indiquera 1000. Il va falloir faire attention à ne pas dépasser la capacité de comptage du compteur (65536) ni les possibilités du pic (fréquence maximum). Cela implique d'adapter la base de temps avec le comptage. Par exemple, pour une fréquence de 100KHz, il est nécessaire de compter pendant 500ms au lieu de 1 seconde.
En tout état de cause, la mise au point du logiciel d'un fréquencemètre est un exercice particulièrement riche pour les amateurs de logipic.