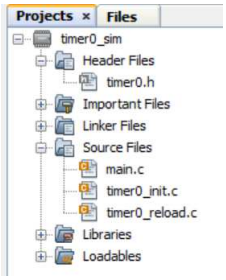
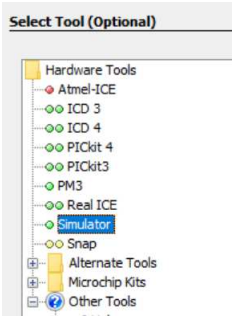
Programme timer0_test

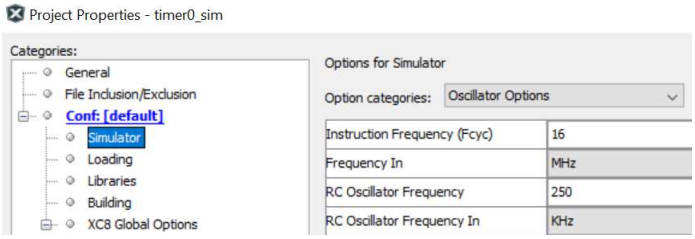
- Créer un nouveau projet nommé **timer0_test** dans le dossier **timer0/test/pjct**
- Associer les fichiers sources selon l'arborescence suivante :



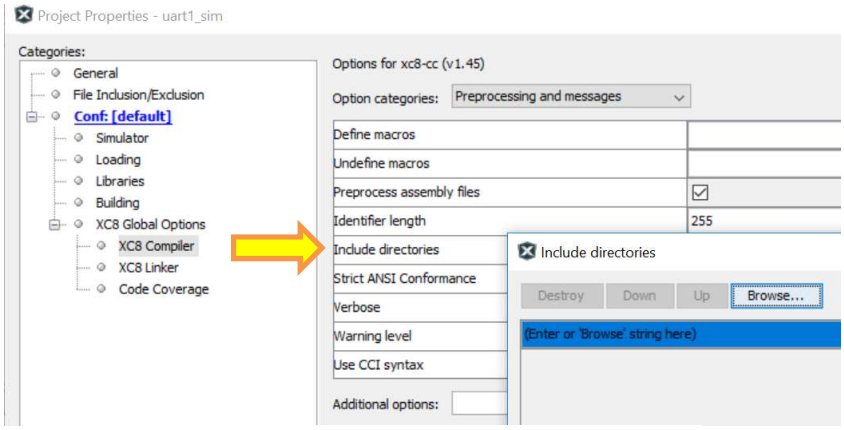
- Sélectionner le simulateur comme outil de **debug.**

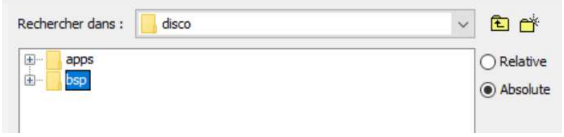


- Fixer la fréquence des cycles machine à 16 MHz (64 MHz / 4)

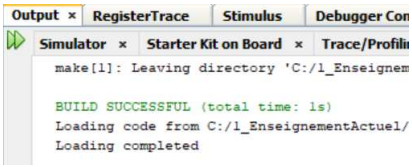


- Configurer le compilateur pour qu'il puisse accéder au dossier **bsp.**





- Compiler pour être s'assurer que tout va bien pour le moment



- Commencer par compléter le fichier **timer0.h**

```
/**
  * @brief enable timer 0 module
  */
#define timer0_start()      // Set T0EN bit

/**
  * @brief disable timer 0 module
  */
#define timer0_stop()       // Test T0EN bit
```

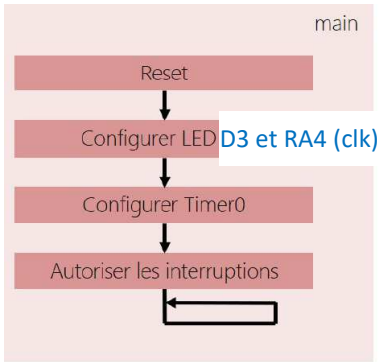
- Continuer avec **timer0_init.c** (attention ! l'horloge interne n'est pas reconnue par le simulateur, nous allons choisir une horloge externe dans la configuration)

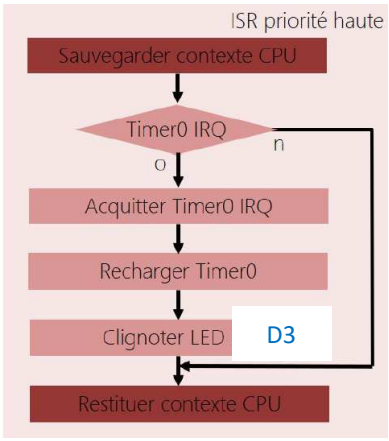
```
void timer0_init(uInt16_t count)
{
    uInt16_t    preload;
    /* TODO count[µs] */

//->    set T0CON0 reg (stop, 16BIT, Postscaler=1:1)
//->    set T0CON1 reg (clk source=T0CKIPPS, Asynchrone, Prescaler=1:64)
//->    set T0CKIPPS reg (xxx=000, PORTA=00, pin4=100) clk in RA4
//->    caculate preload value
//->    write TMR0H value (you can use WRITETIMER0 function)
//->    write TMR0L value
//->    set IDLEN bit to allow sleep instruction
//->    clear TMR0IF flag
//->    enable timer0 interrupt
//->    select high priority for timer interrupt
//->    start timer0
}
```

- C'est au tour de la fonction **timer0_reload** qui n'est qu'une partie de la fonction précédente

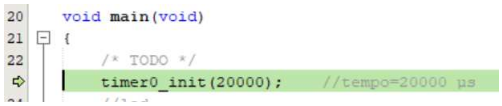
```
void timer0_reload(uInt16_t count)
{
    uInt16_t    preload;
    /* TODO count[µs] */

//->    caculate preload value
//->    write TMR0H value (you can use WRITETIMER0 function)
//->    write TMR0L value
}
```

- Programmer l'algorithme du polycopié de TP en remplaçant LED2 par LED3 (RA4 est prise par clk)



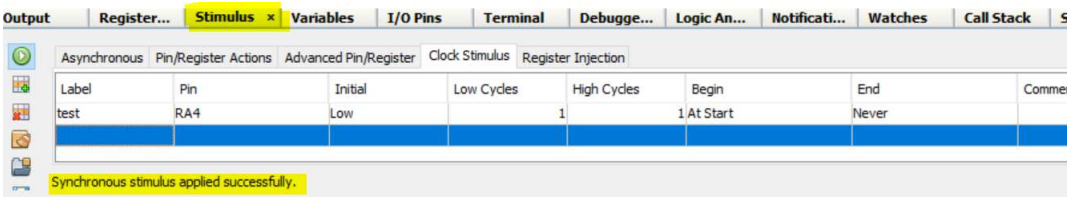


- **Debug Project** (il s'arrête au début du main comme prévu)

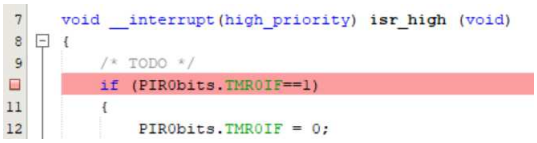


- Avant de continuer, il faut ouvrir la fenêtre **Stopwatch** qui affiche à chaque arrêt quel temps et combien de cycles machine se sont écoulés depuis le dernier arrêt (**Window > Debugging > Stopwatch)**

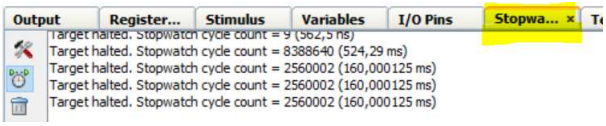
- Créer un stimulus pour l'horloge qui incrémente le timer0 (pin RA4) et l'appliquer



- Mettre un point d'arrêt dans le programme d'interruption



- Faire **Continue (F5)** pour passer plusieurs fois dans le point d'arrêt tout en observant la fenêtre **Stopwatch**



C'est le moment d'analyser ces résultats (une fois le simulateur converge, il donne toujours le même résultat).

$$Nb\ de\ cyc\ =\ 2560002$$

Sachant que $Tcyc\ =\ 1/16Meg\ =\ 62{,}5ns$

$2560002 * 62{,}5ns\ =\ 160ms$ (jusque-là c'est cohérent ! et pourtant on a demandé 20ms !)

La réponse vient du fait que le stimulus fournit une période $Tclk = 2cyc\ =\ 125ns$ -> $fclk\ =\ 8MHz$

Avec une horloge de 64MHz, on aurait eu une temporisation $t = \frac{8*160}{64} = 20ms$