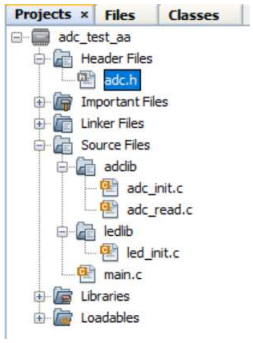
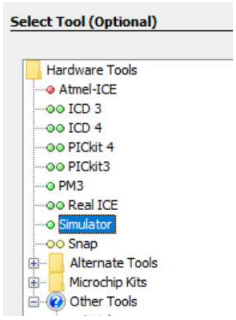
Simulation du convertisseur ADC

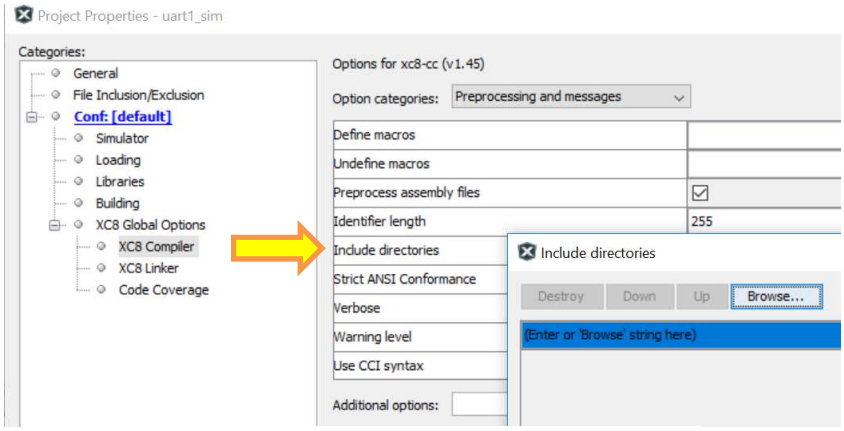
- Créer un nouveau projet dans le dossier **bsp/adc** et associer les fichiers disponibles pour arriver à l'arborescence suivante :



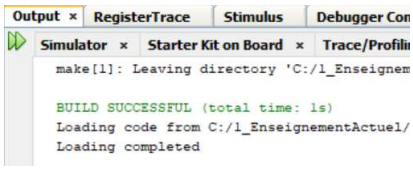
- Sélectionner le simulateur comme outil de **debug.**



- 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 d'entête **adc.h**

```
/**
  * @brief enable adc module
  */
#define adc_enable()    // Set ADON bit

/**
  * @brief start conversion
  */
#define adc_start()     // Set GO bit

/**
  * @brief wait the end of conversion
  */
#define adc_busy()      // Test GO bit
```

- Coder la fonction d'initialisation du convertisseur en suivant l'algorithme suivant :

```
void adc_init(void)
{
    /* TODO */

//->    load ADCON0 register
//->    load ADCON1
//->    load ADCON2
//->    load ADCON3
//->    load ADREF
//->    load ADPCH
//->    load ADPRE
//->    load ADACQ
//->    load ADCAP

    /* pin selection */
//->    set TRISA0 bit
//->    set ANSELA0 bit

//->    enable ADC
}
```

- C'est au tour de la fonction de lecture de l'échantillon converti **adc_read.**

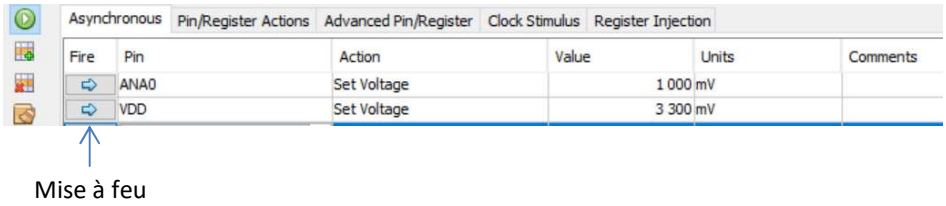
```
uInt8_t adc_read(void)
{
//->    start ADC

//->    DO WHILE (ADC is busy)
//->        continue;
//->    ENDDO

//->    RETURN result of conversion
}
```

- Maintenant, nous pouvons utiliser les fonctions précédentes dans la fonction **main.**

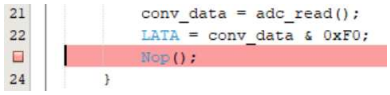
```
void main(void)
{
    /* TODO */
    uInt8_t conv_data;

    adc_init();
    led2_init();
    led3_init();
    led4_init();
    led5_init();

    while (1) {
        /* TODO */
        conv_data = adc_read();
        LATA = conv_data & 0xF0;
        Nop();
    }
}
```

- Compiler à nouveau pour vérifier qu'il n'y a pas de soucis

- Menu **Window > Simulator > Stimulus** pour ouvrir la fenêtre de génération de stimuli. Créer deux stimuli, un pour fixer la tension d'alimentation du PIC et l'autre pour régler la tension à convertir.



- Mettre un point d'arrêt sur l'instruction **Nop()**



- Demander à voir la variable **conv_data** ainsi que les registres **ADRESH** et **ADRESL**

- **Debug Project**
- Mettre à feu les deux stimuli et faire **run (F5)**

Le résultat vaut 77 en décimal. Sachant que pour la pleine échelle, on doit obtenir 255 pour 3,3v : $\frac{1000*255}{3300} = 77{,}27$

Nous ne sommes pas loin de la vérité.

- Essayer d'autres valeurs : 0, 100, 1650, 3300, …