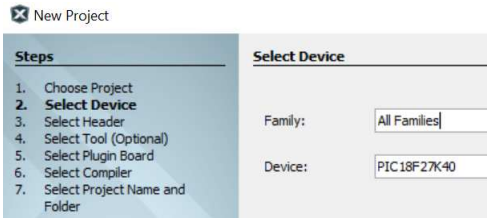
Simulation du TP UART1

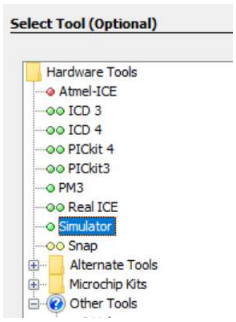
Quand on ne dispose pas de maquette avec soi, on peut utiliser le simulateur de MPLAB pour valider certains programmes. C'est ce que nous allons faire à travers ce tutoriel.

Nous allons partir de zéro, ceux qui ont déjà un projet de créé peuvent prendre le train à l'étape où ils en sont.

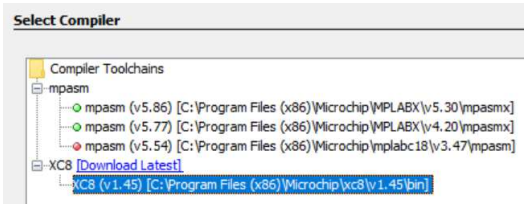
- Créer un nouveau projet en choisissant le bon composant.



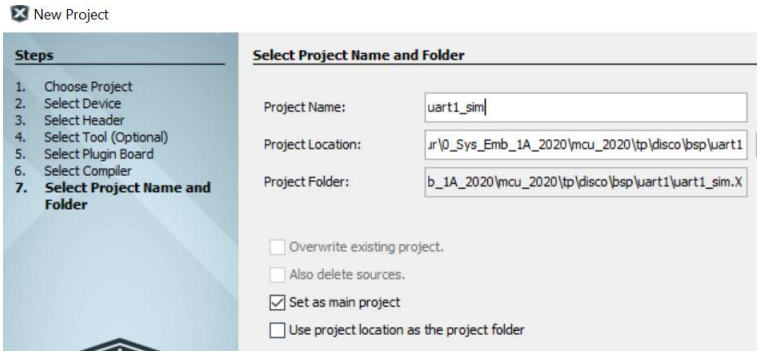
- Sélectionner le simulateur comme outil de **debug.**



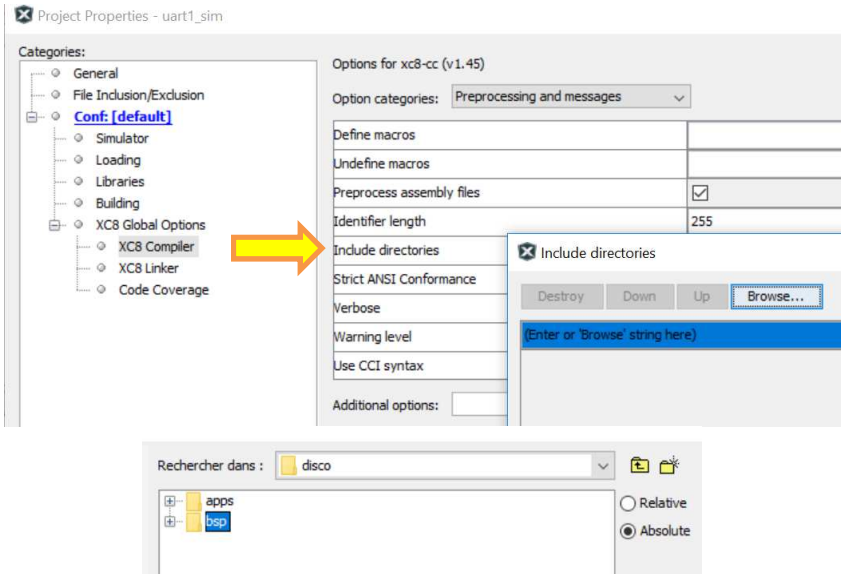
- Choisir la chaine de compilation XC8.



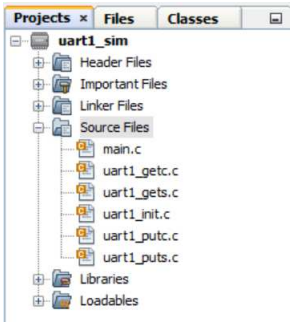
- Nommer le projet.



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



- Inclure au projet les fichiers concernant l'uart1.



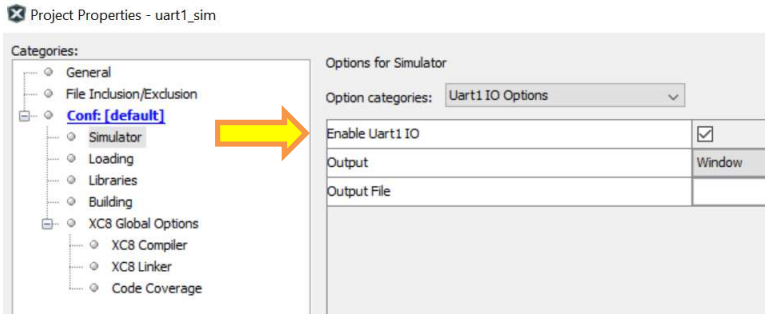
- Voici le programme **uart1_init** que nous avons écrit à la dernière séance.

```
void uart1_init(uInt32_t bdrate)
{
    /* TODO */
    uInt8_t flush;  // clear FIFO
    uInt16_t    bdrate_calcul; // = SP1BRGH:SP1BRGL (16 bits))

        /* connect RX1 and TX1 to I/O pins */
    RC6PPS = 0x09;  // TX1 output connected to RC6 pin (p311 datasheet))
    RX1PPS = 0x17;  // RX1 input connected to RC7 pin
    ANSELCbits.ANSELC7 = 0; // ST and TTL RC7 input buffer enable

        /* baudrate calculate */
    BAUDCON1bits.BRG16 =1;   // baud rate 16 bits mode
    TXSTA1bits.BRGH  = 1;   // hight speed baud rate
    bdrate_calcul = (uInt16_t) ((CPU_FREQ_HZ / (4 * bdrate) - 1));

    SP1BRG = bdrate_calcul;     // = 0x0681
    // SP1BRG is a 16BIT register = SP1BRGH:SP1BRGL

    TX1STA = 0x24 ;        //TXEN=1, BRGH=1
    RC1STA = 0x90;     //SPEN=1, CREN=1

        /* rx interrupt */
    flush = RC1REG;     // flush reception fifo buffer
    flush = RC1REG;
    PIE3bits.RC1IE = 1;
    IPR3bits.RC1IP = 0;
```

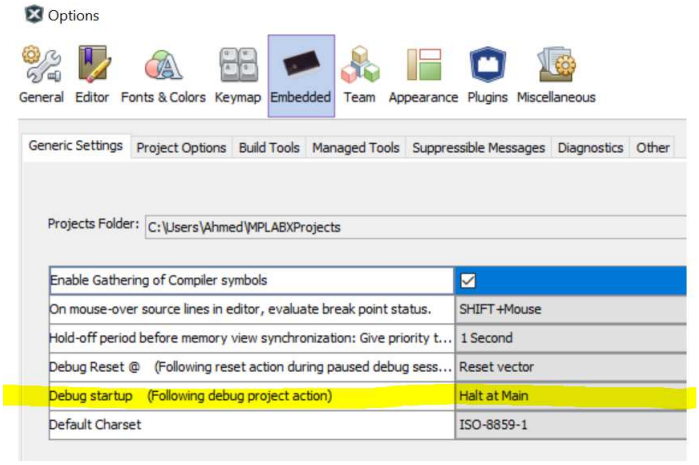
- La fonction **main** qui utilise **uart1_putc** pour écrire sans arrêt la lettre **A.**

```
void __interrupt(low_priority) isr_low (void)
{
    /* TODO */
    uart1_isr_process ();
}

void main(void)
{
    /* TODO */
    uart1_init(9600);    //uart1_init(UART1_BAUD_RATE);
    sys_interrupt_enable();

    while (1)
    {
        /* TODO */
        uart1_putc('A');
    }
}
```

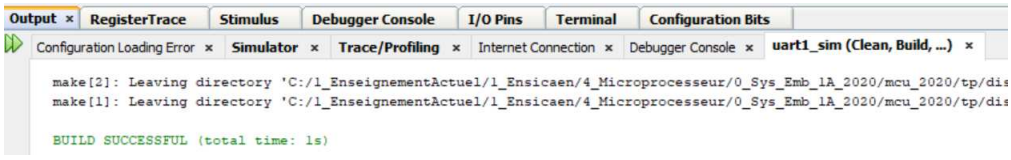
- Configurer le simulateur pour qu'il puisse afficher ce qui transite par l'UART1.



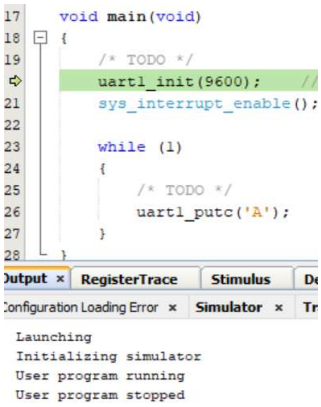
- Menu déroulant (Tools >> Options) pour arrêter le **debuger** au début du main au démarrage.



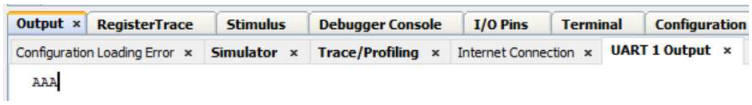
- Après compilation (F11), vérifier qu'il n'y a pas d'erreur.



- Maintenant, faire : **Debug Main Project**, le debuger doit s'arrêter au début du main.



- **Step Over** (F8) pour une exécution pas à pas. Une nouvelle fenêtre **UART1 Output** apparaît et affiche la lettre envoyée 'A'. Refaire **F8** plusieurs fois.



- **Continue** (F5) pour une exécution en continue, les **A** défilent.

- Essayer les points d'arrêt.
- Écrire la fonction **uart1_puts** et la simuler.

2ème partie

Maintenant, nous allons voir comment simuler une réception de données par le port série RS232. Vu du PIC, les données vont tomber dans le registre RC1REG de l'UART1.

Voici comment se déroule une réception de donnée :

- Nous avons tout fait dans la fonction uart1_init pour qu'une interruption de priorité basse soit déclenchée dès qu'un caractère arrive dans la salle d'attente RC1REG.
- Après les sauvegardes d'usage, le CPU va exécuter le sous-programme d'interruption (ISR) **isr_low** qui se trouve dans le fichier **main.c**
- Comme il faut déporter tous les traitements lourds, la fonction **low_isr** se contente d'appeler la fonction qui s'occupera du traitement. Cette fonction se nomme **uart1_isr_process**, elle se trouve dans le fichier **uart1_getc**
- La fonction **uart1_isr_process** teste si le flag **RC1IF** est actif, auquel cas elle lit la donnée arrivée dans **RC1RG** et la stocke dans un buffer. Elle signale par un flag à la fonction **main** qu'une donnée présente dans le buffer nécessite d'être traitée.

Votre travail consiste à compléter les fonctions suivantes : **isr_low**, **uart1_usr_process**, **uart1_getc** et **main**

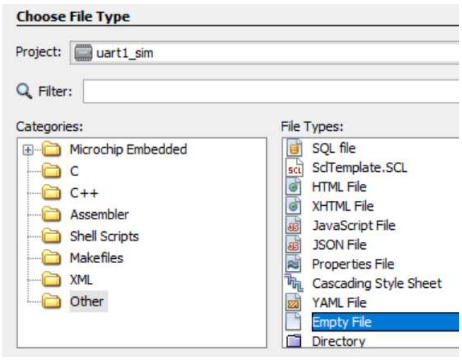
La fonction main se contente de retourner les caractères reçus, la voici :

```
void main(void)
{
    /* TODO */
    uInt8_t data_in;
    uart1_init(9600);    //uart1_init
    sys_interrupt_enable();

    while (1)
    {
        /* TODO */
     //   uart1_putc('A');
        if (uart1_getc(&data_in, 0))
        {
            uart1_putc(data_in);
        }

    }
}
```

Une fois les fonctions précédentes complétées, voici comment simuler une réception de données :

Nous allons simuler les données qui arrivent dans l'UART1 par le contenu d'un fichier texte qui va servir de **stimulus**

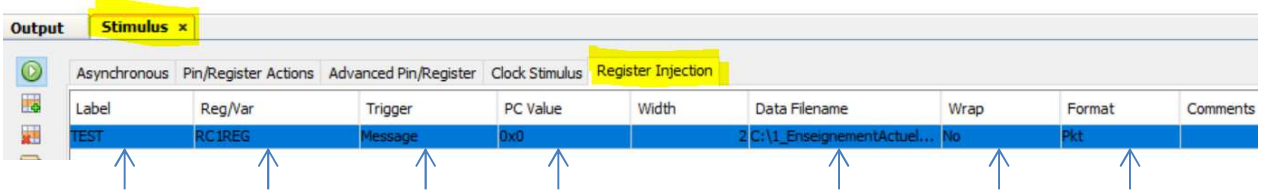
- Créer un nouveau fichier :



- Compléter le fichier avec les données suivantes et sauvegarder :

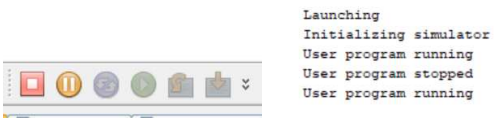
```
// start

wait 100 ms
"A"

wait 100 ms
"B"

wait 100 ms
// lettre G
0x47

wait 100 ms
//Line Feed
0x0A

wait 100 ms
"FIN"
```

- Maintenant, il faut configurer le stimulus : menu **window>>simulator>>stimulus**

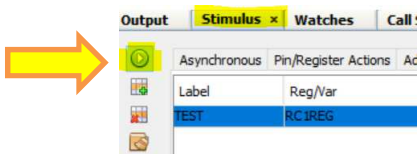


Nom — Registre de réception — Type de Trigger — 0 si Trigger n'est pas PC — Fichier de data : data_in.txt — 1 envoi — Packet pour UART

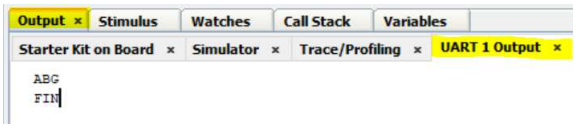
Une fois le programme est en cours d'exécution :



- Lancer le stimulus :



- Voici le résultat sur la fenêtre uart1 output :



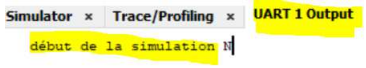
Pour rediriger la fonction « printf » vers l'UART1

La fonction **printf()** fait appel à la fonction **putch()** qui envoie chaque caractère vers la sortie standard **stdout**.

Pour diriger les caractères vers la console de **l'UART1**, il faut s'assurer que **putch()** écrit dans **l'UART1**.

Coder la fonction **putch()** comme ceci, devant le main, pour diriger les caractères vers **l'UART1**.

```c
#include <bsplib.h>
#define _XTAL_FREQ 64000000
#include <stdio.h>
#include <stdlib.h>

void putch(unsigned char data) {
    while(uart1_tx_busy()) // OU while (!TX1STAbits.TRMT);
        continue;
    TX1REG = data;
}

void main(void)
{
    /* TODO */
    uInt8_t data_in;
    uart1_init(9600);    //uart1_init(UART1_BAUD_RATE);
    sys_interrupt_enable();
    printf("début de la simulation");
```



---

uart1_gets

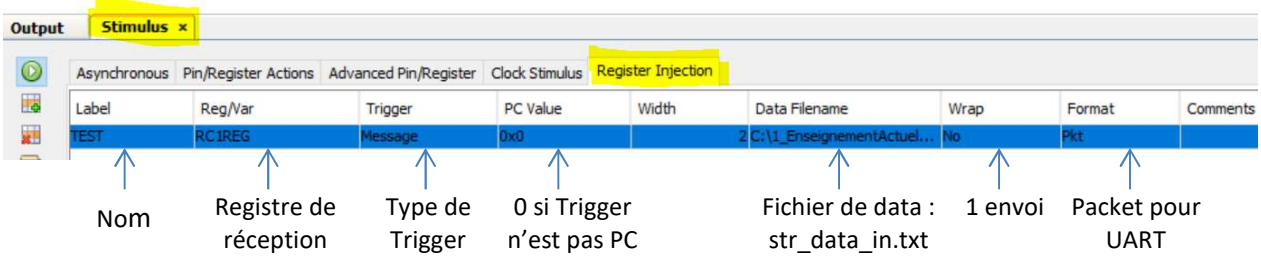
- Coder la fonction uart1_gets

```c
static uInt8_t i = 0; //conservée pour l'appel suivant
uInt8_t tmp;
// DO
    // IF (uart1_getc(&tmp, echo) == 0)
    //         return 0;
    // ENDIF

    // str_buf[i] = tmp;
    // i++
// WHILE (tmp != '\r')

//  i--                         //position précédente
//  str_buf[i] = '\0';          //terminer par '\0'
// i = 0;                       //réinitialiser le buffer
// return 1;
```

- Créer un nouveau fichier texte : **str_data_in.txt**

```
//str_data_in.txt
// start

wait 100 ms
"ENSICAEN"

wait 100 ms
// retour chariot
0x0D

wait 500 ms
```

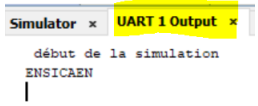
- Configurer un nouveau stimulus



- Modifier la fonction **main** pour tester **uart1_gets**

```
void main(void)
{
    /* TODO */
    uInt8_t str_in[120];    //tableau pour uart1_gets
    uart1_init(9600);
    sys_interrupt_enable();
    printf(" début de la simulation ");

    while (1)
    {
        if (uart1_gets(str_in, 120, UART1_ECHO_OFF)) {
            uart1_puts(str_in);
            uart1_puts("\r\n");
        }
    }
}
```

- Observer le résultat dans la fenêtre **Output**



- Modifier le stimulus pour qu'il envoie en boucle le message et observer le résultat