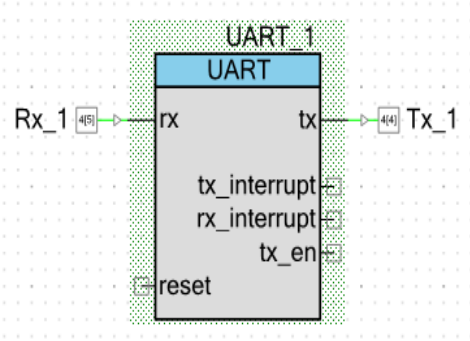
Documents fournis	Apports de cours
- Schéma de la maquette - schéma carte mère - Docs intégrées au logiciel	- variables de type chaîne de caractères - Les codes " \n \r \t " - les pointeurs par ex ptc++ et *ptc++

Important : tout ce que vous faites doit être consigné sur le compte-rendu de TP. C'est le compte-rendu de TP qui sera noté. Coller le texte de TP sur le compte-rendu.

1 Transmission série (envoi d'un message).



1.1 A l'aide du logiciel PSOC CREATOR créer un nouveau projet qui s'appelle par ex **Tpserie**.
Ouvrir la feuille de schéma du composant (workspace => TopDesign.cysch). Depuis la fenêtre **Component Catalog**, faire glisser sur le schéma le composant **UART. Configurer** le bloc : données 8 bits, 1 stop, pas de bit de parité, vitesse : 9600 bauds (bits /seconde). Ouvrir **----.cydrw** depuis WorkSpace et à l'aide du schéma de la carte mère, affecter les 2 bits de uart sur les bonnes connections. Cliquer sur **generate application** (icône direct ou menu build) pour câbler le projet et générer les fichiers ***.h** (header).

1.2 Traduire **: UART (**Universal Asynchronous Receiver Transmitter)

1.3 Pourquoi cette liaison série est elle asynchrone ?

1.4 Récupérer dans le ---.h la fonction d'initialisation et celle qui permet d'envoyer une chaîne de caractères. Expliquer la différence entre les fonctions uart...PutString() et uart...PutChar()

1.5 Réaliser un programme (et le commenter) qui envoie « lycee Deodat de Severac » sur le port série. La réception coté PC se fera par un logiciel de type terminal (ex Comm.exe).

1.6 Mesurer la durée d'un bit à l'oscilloscope

1.7 On rappelle que sur série asynchrone, le bit poids faible est envoyé en 1° juste après le bit de start.

1.8 Décoder sur la ligne série le premier caractère.

1.9 Si vous êtes en avance, tester le BusBee : appareil qui décode presque toutes les liaisons séries.

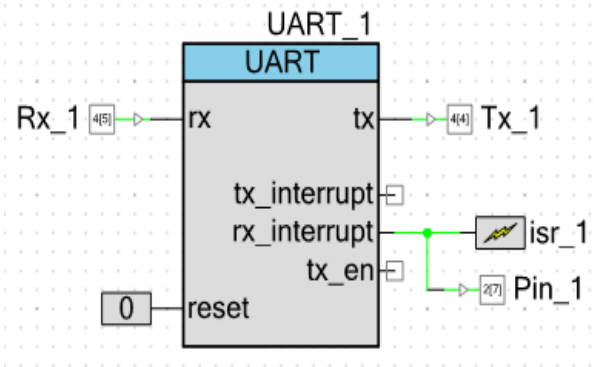
2 Envoi message et réception d'un seul caractère.

2.1 Votre programme doit permettre ce fonctionnement :

Envoi sur la liaison série de " appuyer sur une touche " . Ensuite appuyer sur une touche coté terminal (PC). Le programme affiche ensuite "la touche pressée est : …" en affichant le caract. envoyé par le terminal

3 Réception en interruption d'une chaîne terminée par \r

On souhaite maintenant recevoir un message de plusieurs caractères, terminé par la touche entrée ('\r' en C) du PC. Seul le message complet nous intéresse, le PSoC doit pouvoir faire autre chose pendant la réception du message, et ne réagir que lorsque le message reçu est complet.
Une solution pour faire cela est de garnir une mémoire tampon (buffer) en interruption, puis de positionner un flag (indicateur) quand un message complet a été reçu,



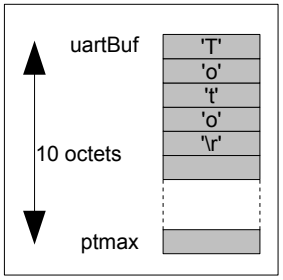
On ajoute donc un composant interruption
(ISR : interrupt service request).
On peut sortir le signal qui déclenche l'interruption sur une broche pour l'observer à l'oscilloscope.
Dans le bloc uart, vérifier que interruption sur octet reçu est bien cochée dans l'onglet avancé de config de uart_1.

Refaire generate application.
Il apparaît un fichier isr_1_.C et un isr_1_.h

Le isr_1_.C contient 2 zones où vous pouvez écrire
dans la première, entre les commentaires contenant #START et #END vous déclarez les variables globales de l'inter

```
/*****************************************************
*  Place your includes, defines and code here
*****************************************************/
/* `#START isr_1_intc` */

/* `#END` */
```



vos déclarations devraient ressembler à ceci :

```
#include <device.h>
#define MaxBuf 10      // taille maxi chaine recue MODIFIABLE
uint8 uartBuf[MaxBuf+1];   // pour messages recus en inter + terminateur 0
int8 uartRflag=0;          // flag      -1  0  1
uint8 *ptc = uartBuf;       // pointeur @ où ecrire le caract reçu
uint8 *ptmax = uartBuf+MaxBuf;      // dernière @ autorisée
```

On a donc réservé 11 octets (10 caractères + terminateur de chaîne) en mémoire dans le tableau **uartBuf[]**.
@ du 1° : uartBuf, @ du caractère en cours : **ptc**, @ de fin du buffer : **ptmax**.
Le flag **uartRflag** va permettre d'informer de ce qui se passe :

- -1 une réception (non terminée) est en cours.
- 0 repos rien reçu.
- 1 message complet a été reçu le programme principal peut lire. Ensuite le prog principal remet le flag à 0

Dans la 2° zone, vous allez écrire le code de cette fonction d'inter qui se déclenche automatiquement quand un octet est reçu sur la liaison série.

```
 /*  Place your Interrupt code here. */
 /* `#START isr_1_Interrupt` */

 /* `#END` */
```

3.1 **Écrire le code en interruption** qui commence par :

```
if(UART_1_RX_STS_FIFO_NOTEMPTY == UART_1_ReadRxStatus( ) )    //RAZ flag
      {       a = UART_1_ReadRxData() ;
            ….............
```

Expliquer ces 2 lignes de code.
ensuite il y a 3 cas à traiter :

- le flag est déjà à 1, on ne fait rien (ignore les nouveaux caractères reçus tant que main() n'a pas utilisé le message)
- le car reçu est '\r' on ajoute le terminateur de chaîne, on met le flag à 1, ptc au début du buffer pour la prochaine fois.
- Dans les autres cas, c'est une réception normale : s'il reste de la place,
  écrire le car ds buffer, flag à -1 (réception en cours), incrémenter pointeur pour la prochaine fois.

Le programme principal devrait ressembler à ceci :

```
#include <device.h>
extern uint8 uartBuf[];       // pour messages recus en inter + terminateur 0
extern int8 uartRflag;
void faire(void)  // consignes utilisateur
{     UART_1_PutString("\n\n\r Entrer un message\r\n") ;
      UART_1_PutString("termine par entree\r\n") ;
}
void main()
{   UART_1_Start() ;
    CyGlobalIntEnable;  // inters valides
    isr_1_Start();
    faire();
    for(;;)
    {    if (uartRflag==1)
         {  UART_1_PutString("\r\nmessage recu : ") ;  //actions liées au message recu
            UART_1_PutString(uartBuf) ;
            uartRflag=0;                          // message recu utilisé => raz flag
            faire();                              // consigne à nouveau
         }
    }
}
```

3.2 **Tester le projet expliquer, commenter,** et faire les copies d'écran qui montrent le fonctionnement
3.3 S'il reste du temps, refaire l'inter en syntaxe tableau.