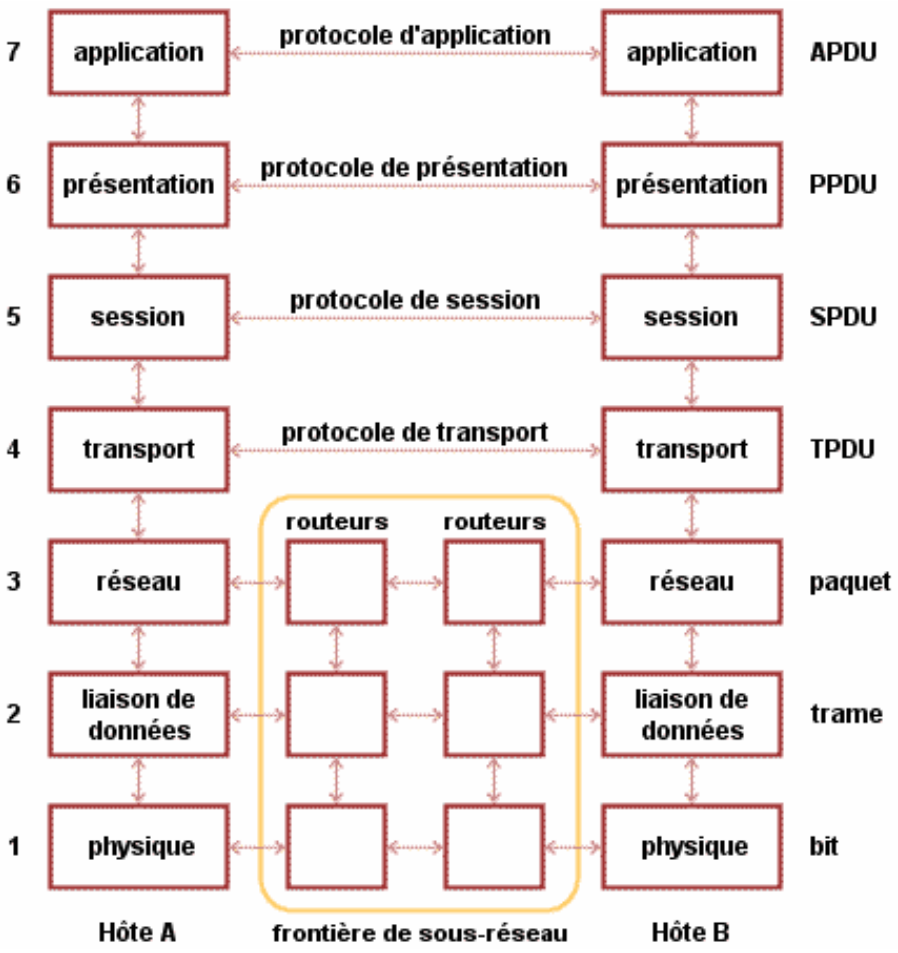
Liaison série RS232

Introduction

Pour transmettre des informations numériques d'un système à un autre, un certain nombre de règles ont été codifiées afin que tout système puisse communiquer avec un autre. Il s'agit de transmettre ces informations avec le minimum d'erreurs.
L'OSI (International Standards Organisation) a développé un modèle de référence appelé modèle OSI (Open Systems Interconnection).
Chacune des sept couches a un rôle particulier, mais toutes ne sont pas indispensables.
Dans les liaisons que nous allons étudier, seules les deux dernières couches sont utilisées : couche physique et constitution de la trame.



Nous proposons l'étude d'un type de transmission série asynchrone : RS232. Cette liaison dans son mode simplifié utilise trois lignes : une ligne de masse, une ligne de transmission de données Tx, une ligne de réception de données Rx.
Remarquons que le signal d'horloge n'est pas transmis, d'où son nom d'asynchrone.

La liaison RS232, très utilisée jusqu'à présent sur les ordinateurs, a tendance à disparaître aux profits de l'USB. Cependant, avec le développement de l'électronique embarquée, ce type de liaison reste très utilisé car la plupart des microcontrôleurs savent très facilement les gérer.

En préparation de la manipulation, vous décrirez le protocole de la liaison série. Vous expliciterez le terme point à point par opposition à multipoint.

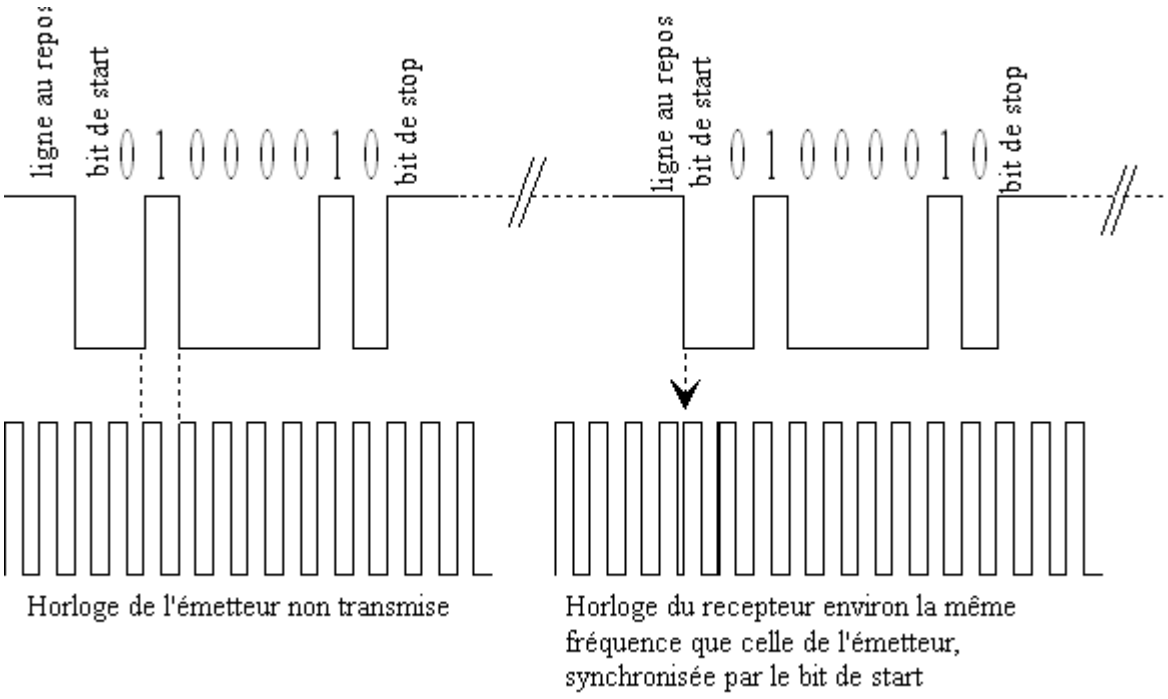
Mise en œuvre d'une liaison RS232.

I- Mise en œuvre d'une liaison série à partir d'un PC.

A partir du logiciel X-CTU et d'un PC connecté via RS232 à la maquette PsoC, lors de la frappe d'un caractère, on affichera sur le LCD le code ASCII en hexadécimal ainsi que le caractère de la touche frappée au clavier.

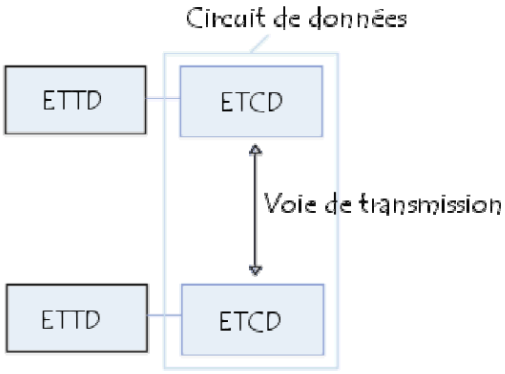
Pour que le PC et le PsoC puissent se comprendre, il faut qu'ils utilisent le même protocole.
Ce protocole définit la vitesse de transmission et le contenu de la trame.
On choisira comme vitesse 9600 bauds
La trame sera constituée de :

- 1 bit de start,
- 8 bits de données,
- pas de bit de parité,
- 1 bit de stop,
- pas de contrôle de flux (s'il y a contrôle de flux, il peut être matériel ou logiciel).

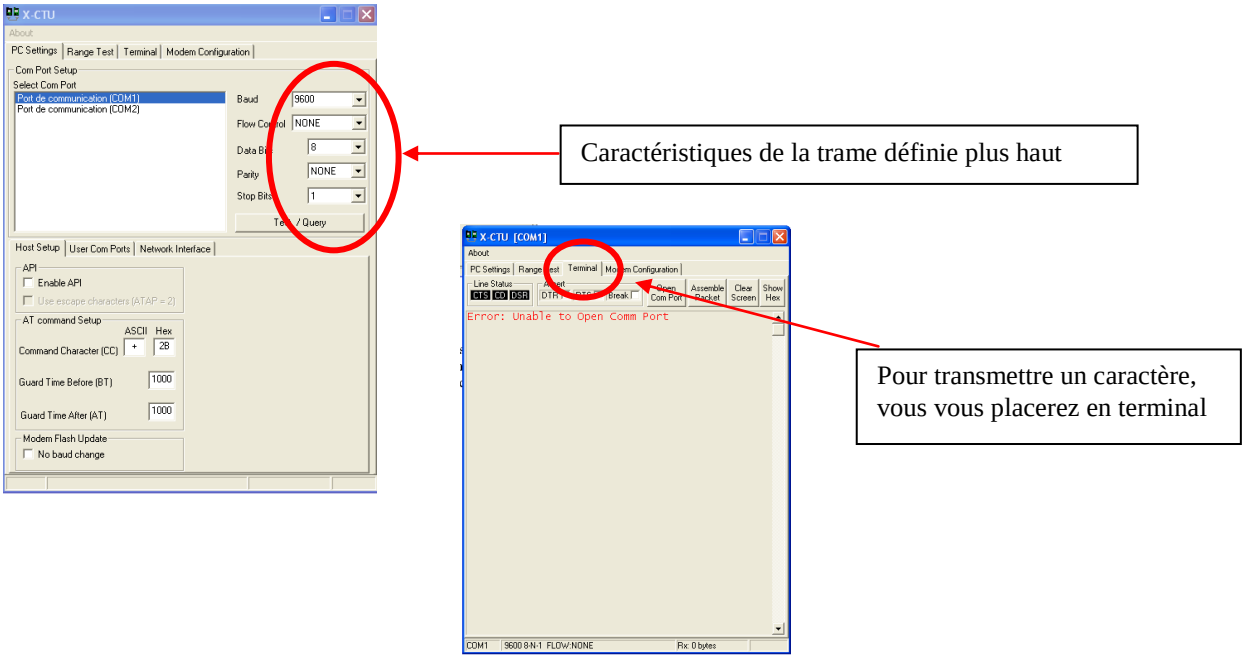


La liaison est composée de

- deux ETTD (Equipement terminal de traitement de données) : l'un est le PC, l'autre est le PsoC
- deux ETCD (Equipement terminal de circuit de données) : ce sont les circuits qui adaptent les signaux émis et reçus aux caractéristiques de la ligne.



Pour transmettre les caractères, on utilisera X-CTU (Vous pouvez aussi utiliser hyper terminal que l'on trouve dans : *tous les programmes ; accessoires ; communications.)*
Ouvrir une liaison sur le port COM1 et la paramétrer comme ci dessous.



Notez que ce logiciel X-CTU vous servira pour programmer les modules Xbee qui fonctionnent avec le protocole RS232.
Ces petits modules fonctionnent à une fréquence de 2,4Ghz et vous serviront à réaliser notamment des réseaux de capteurs.

Programmation du PsoC

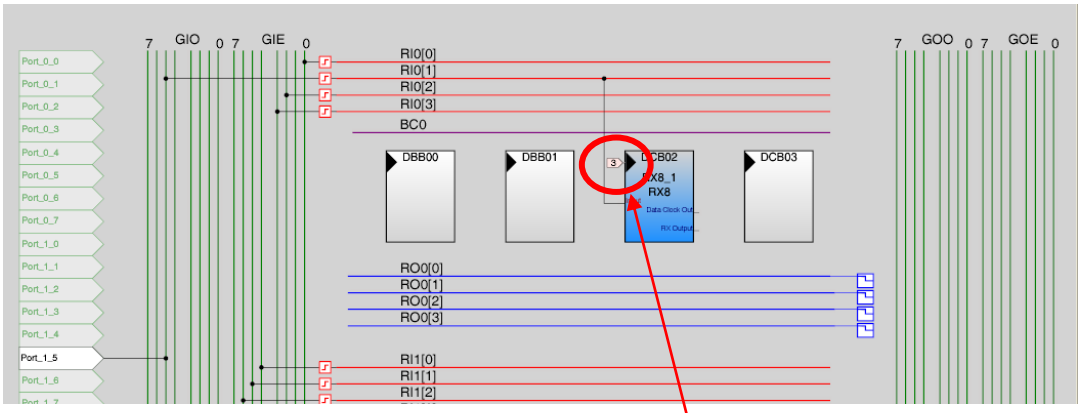
Deux modules sont indispensables pour la réception et l'affichage du caractère :

- Un module Rx que vous trouvez dans la bibliothèque Digital Comm
- Un module LCD que vous trouvez dans la bibliothèque Misc Digital

Sur le portail, télécharger le fichier "Liaison_RS232_avec_PC". Vérifiez qu'il est conforme aux indications ci-dessous puis implantez le programme dans le PSoC.

Pour chaque module, le constructeur Cypress met a disposition une datasheet dans laquelle vous trouverez toutes les informations nécessaires à son bon fonctionnement. Toutes les lignes de commandes y sont aussi mentionnées.
Pour y accéder, clic droit sur le module puis datasheet.

Le LCD est branché physiquement (sur la carte que vous utilisez) sur le port 2 du PsoC.
L'entrée de Rx sera connectée sur le port P1(5) du PSoC.



Le paramétrage du module Rx doit être identique à celui du PC pour que la communication puisse s'établir.

Properties - RX8_1

Name	RX8_1
User Module	RX8
Version	3.3
Clock	VC3
Input	Row_0_Input_1
ClockSync	Sync to SysClk
RxCmdBuffer	Enable
RxBufferSize	24 Bytes
CommandTermina	13
Param_Delimiter	32
IgnoreCharsBelow	32
RX Output	
Data Clock Out	
InvertInput	Normal

Vous pouvez choisir l'horloge qui cadence la réception des bits dans Clock. Ici on a choisi VC3. Le chiffre 3 apparaît sur le dessin.

Cette horloge est définie dans les paramètres globaux.

Global Resources - liaison_rs232_avec_pc

Power Setting [Vc	5.0V / 24MHz
CPU_Clock	SysClk/8
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	2
VC2= VC1/N	16
VC3 Source	VC1
VC3 Divider	156
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

VC3 est donc égal à VC1/156
VC1 étant égal à SysClk/2
VC3 = 76 923 Hz.

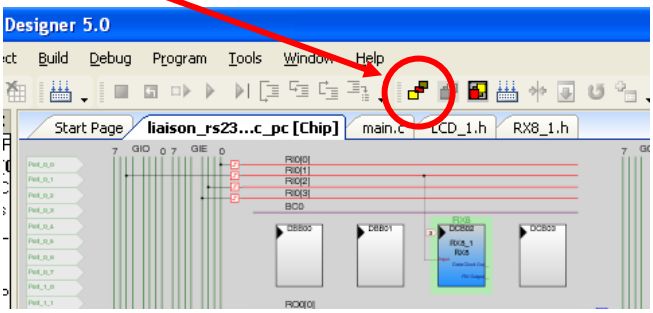
La datasheet indique que cette fréquence doit être huit fois supérieure à la vitesse de transmission des bits.

Ce qui fait :
76 923 / 8 = 9615 bauds.
Les bauds sont des bits par seconde lorsque la valence vaut 2 (seulement deux valeurs, 0 ou 1). Pour une modulation, on peut transmettre

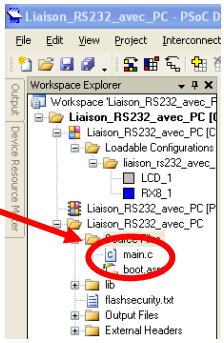
plusieurs bits en même temps ; si on transmet 2 bits simultanément, le débit sera par exemple de 1200 bauds mais 2400 bits / s.

On voit que l'on n'a pas exactement 9600 mais comme chaque trame est resynchronisée, on fait le pari que tout se passera bien sur 8, 9 ou 10 bits suivant la configuration de la trame. L'erreur entre vitesse de transmission et fréquence choisie ne doit pas dépasser 10%.

En cliquant dans l'onglet, PsoC Designer va construire toute l'architecture soft nécessaire à la programmation.



Dans Workspace Explorer, Source Files, vous pouvez ouvrir le fichier main.c dans lequel vous allez écrire le programme principal en langage C.




```
1  //--------------------------------------------------------------------------
2  // C main line
3  //--------------------------------------------------------------------------
4
5  #include <m8c.h>        // part specific constants and macros
6  #include "PSoCAPI.h"    // PSoC API definitions for all User Modules
7
8
9  void main()
10 {
11     BYTE bRxStatus=0;
12     BYTE bRxData;
13     char str[2];
14
15     LCD_1_Start();
16
17     M8C_EnableGInt;
18     RX8_1_Start(RX8_1_PARITY_NONE);
19     LCD_1_Init();
20
21     LCD_1_Position(0,0);
22     LCD_1_PrCString("RX (ASCII)=");
23     LCD_1_Position(1,0);
24     LCD_1_PrCString("Caract=");
25
26     while(1)
27     {
28         while (!(bRxStatus=RX8_1_bReadRxStatus() & RX8_1_RX_COMPLETE))
29         {
30         }
31         if ((bRxStatus & RX8_1_RX_NO_ERROR) ==0 )
32         {
33             bRxData = RX8_1_bReadRxData();
34             LCD_1_Position(0,11);
35             LCD_1_PrHexByte(bRxData);
36             LCD_1_Position(1,8);
37             str[0]=bRxData;
38             str[1]=0x00;
39             LCD_1_PrString(str);
40         }
41     }
42 }
43
```

Déclaration des variables locales
Déclaration d'une chaîne da caractère de deux octets

Démarrage du LCD

Autorisation des interruptions
Démarrage du module Rx sans test de parité
Initialisation du LCD

Positionnement du curseur du LCD ligne 0 caractère 0
Ecriture de la chaîne de caractères RX (ASCII) =
Positionnement du curseur du LCD ligne 1 caractère 0
Ecriture de la chaîne de caractères Caract

Ouverture d'une boucle while (tant que la condition est vérifiée, la boucle s'exécute

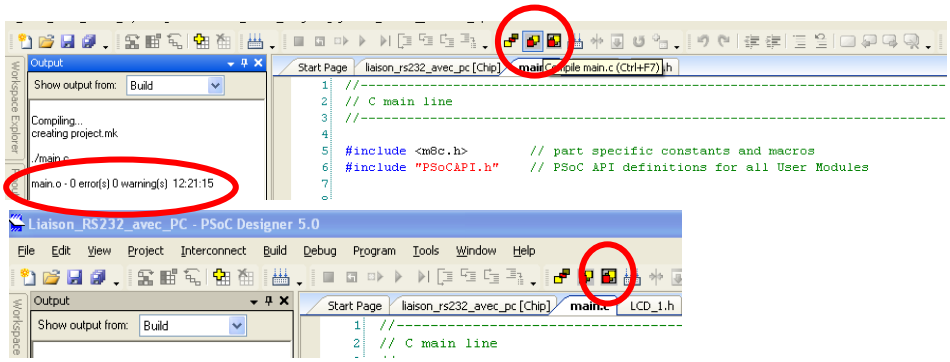
Attente de réception complète de l'octet

On vérifie d'éventuelles erreurs

Pas d'erreur détectée, bRxData prend la valeur de l'octet reçu
Positionnement du curseur du LCD
Ecriture de la valeur hexadécimale de l'octet
Positionnement du curseur du LCD
Lecture du premier octet de la chaîne de caractères
Le deuxième octet étant 0, arrêt de la lecture.
Ecriture sur le LCD de la chaîne de caractère.

.

Lorsque le main est écrit, vous devez le compiler et vérifier qu'il n'y a pas d'erreur.

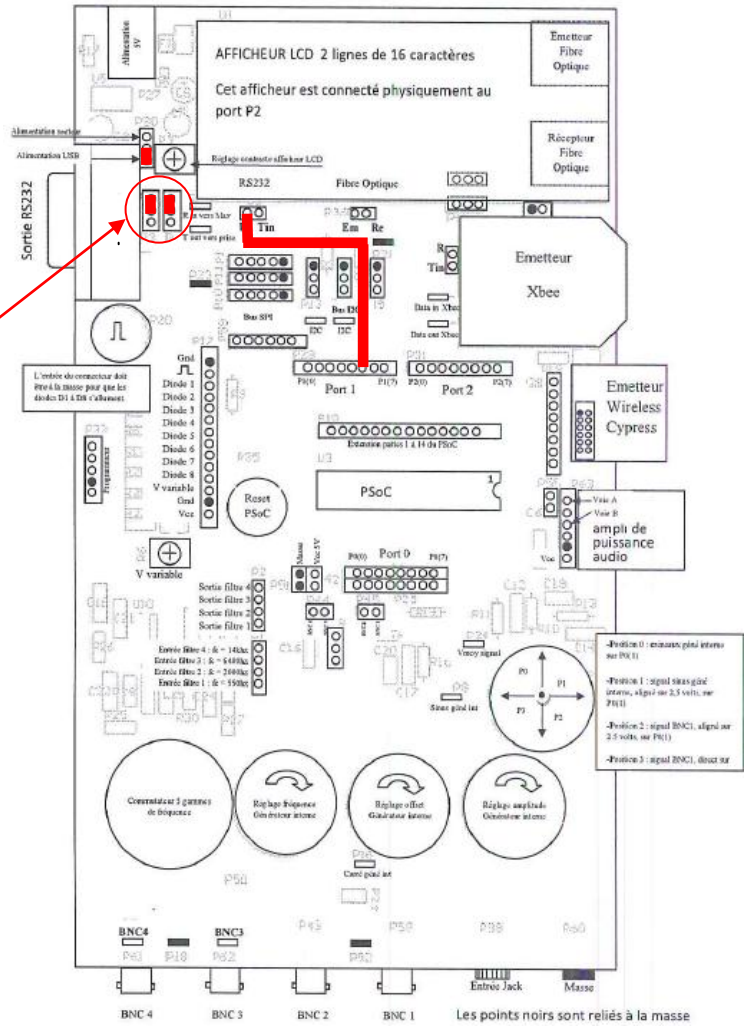


Il ne reste plus qu'à construire le projet

Pour la phase de programmation, reliez le programmeur à la carte PsoC. Vérifiez que le cavalier d'alimentation est bien sur la position USB.

Câble RS232

Si votre cordon RS est croisé (chatterton jaune/vert), câblez la carte comme indiqué, sinon inversez les deux cavaliers prêts de la prise RS.



- ***Appuyez sur le caractère a du clavier et vérifier qu'il est bien reçu par le PsoC.***
- ***Vérifier à l'aide de la table des caractères ASCII, que la valeur hexadécimal correspond bien au caractère affiché***

- ***En utilisant le mode monocoup de l'oscilloscope, faîtes le relevé temporel du signal provenant du PC et celui du signal entrant sur le PsoC.***
  Entre l'entrée RS232 et l'entrée du PsoC, le signal transite par un MAX232. Quel est son rôle.

- ***Passez la vitesse de transmission de l'ordinateur à 19200 bauds, celle du PsoC étant inchangée. Si l'on tape le caractère a, quel caractère s'affiche sur le LCD de la carte PsoC ? Est-ce normal ?***

II- Liaison RS232 entre deux PsoC : l'un émetteur, l'autre récepteur.

On se propose de transmettre la valeur d'une tension continue à un récepteur par liaison RS.

Importez le dossier "Liaison_TX_PSoC_vers_PSoC" à partir du portail.

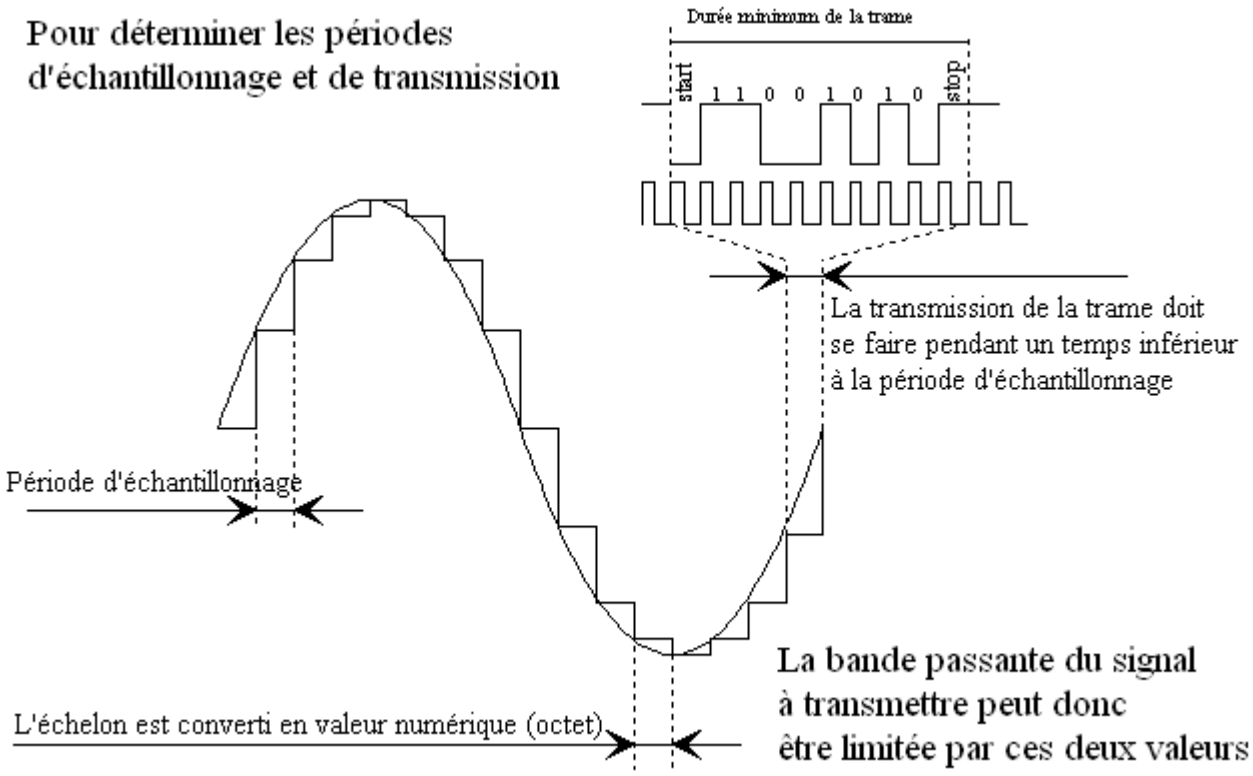
Global Resources - liaison_tx...

Power Settin	5.0V / 24MHz
CPU_Clock	SysClk/8
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk	16
VC2= VC1/N	2
VC3 Source	VC1
VC3 Divider	1
SysClk Sourc	Internal
SysClk*2 Dis	No
Analog Powe	SC On/Ref High
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Pow	Low
SwitchModel	OFF
Trip Voltage	4.81V (5.00V)
LVDThrottleE	Disable
Watchdog E	Disable

Ouvrez le projet dans Psoc Designer.
A l'aide d'un PGA, d'un Delsig et d'un LCD, on affiche en hexadécimal la valeur d'une tension.
Un module TX transmet cette information suivant le protocole RS232.

L'entrée de la tension continue, issue du potentiomètre, se fait sur le port P0(3).
La sortie « PWM Output » du delsig se fait sur le port P0(5). Cette sortie fournie une impulsion chaque fois qu'une acquisition a eu lieu.
La sortie de TX, module de transmission se fait sur le port P2(7).
L'horloge VC2 pilote le Delsig, alors que VC3 pilote le module de transmission.

- ***Déterminez la fréquence d'échantillonnage et la vitesse de transmission des bits .***
- ***D'après les commentaires suivants, la fréquence de transmission est-elle bien choisie par rapport à la fréquence d'échantillonnage ?***



Visualisez à l'oscilloscope les signaux en P0(5) et en P2(7).
Expliquez

Mettez maintenant l'affichage en commentaire.

```
//LCD_1_Position(1,8);
//LCD_1_PrHexByte(valeur);
```

Refaites la même mesure. Expliquez.

Visualisez le main :

```
//--------------------------------------------------------------------------
// C main line
//--------------------------------------------------------------------------

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules


void main()
{
   BYTE valeur =0;
   PGA_1_Start(PGA_1_HIGHPOWER);
   LCD_1_Start();
   DelSig_Start(DelSig_HIGHPOWER);
   DelSig_StartAD();
   M8C_EnableGInt;

   TX8_1_Start(0x00);

        LCD_1_Position(0,1);
       LCD_1_PrCString("Tension");
        LCD_1_Position(1,1);
               LCD_1_PrCString("V = Ox");

       while(1)
    {
        while(DelSig_fIsDataAvailable() == 0);

        valeur=DelSig_cGetData();
               valeur=valeur+0x80;

               LCD_1_Position(1,8);
               LCD_1_PrHexByte(valeur);

               TX8_1_SendData(valeur);

    }
}
```

Démarrer le module TX :
`TX8_1_Start(0x00);`

Envoie de la valeur à TX :
`TX8_1_SendData(valeur)`

Changez la ligne « `valeur=DelSig_cGetData();` » par
« `valeur=DelSig_cGetDataClearFlag();` »

Visualisez à l'oscilloscope les signaux en P0(5) et en P2(7). Expliquez.
Déterminez le nombre d'octets transmis à la seconde et le nombre d'échantillons que l'on doit transmettre à la seconde.

Mettez maintenant l'affichage en commentaire.

```
//LCD_1_Position(1,8);
//LCD_1_PrHexByte(valeur);
```

Refaites les mêmes mesures, concluez.

On voit que dans cette configuration, on transmet tous les échantillons.
Inconvénient, on ne gère plus l'affichage.
Pour remédier à cela, on se propose de travailler sous interruption.
Lors de la phase de génération des fichiers, PSoC Designer crée des fichiers d'interruption pour chaque module pouvant fonctionner de cette manière et notamment le Delsig.
Ce fichier s'appelle « DelSigINT.asm » et se trouve sous lib → Library Source Files.
Se fichier est écrit en assembleur.

```
;;===========================================================================
;;  FUNCTION NAME: _DelSig_ADConversion_ISR:
;;===========================================================================
;;
_DelSig_ADConversion_ISR:
   push  A
   mov   A, reg[DEC_DH]              ; Is value == 2^RES?
   jz    .NoOverflow                 ;    No, data is in normal range
   mov   A, reg[DEC_DL]
   mov   A, UMAX                     ;   Yes, limit to 2^RES - 1
   jmp   .ConversionReady

.NoOverflow:
   mov   A, reg[DEC_DL]

.ConversionReady:
IF DelSig_2S_COMPLEMENT
   ; Internal hardware format is unsigned; convert so zero is half scale
   sub   A, (UMAX+1)>>1
ENDIF
   mov [DelSig_cResult],  A
   mov [DelSig_bfStatus], DelSig_DATA_READY_BIT               ; Set valid data flag


   ;@PSoC_UserCode_BODY@ (Do not change this line.)
   ;---------------------------------------------------
   ; Insert your custom code below this banner
   ;---------------------------------------------------

   call TX8_1_PutChar
   ;---------------------------------------------------
   ; Insert your custom code above this banner
   ;---------------------------------------------------
   ;@PSoC_UserCode_END@ (Do not change this line.)
   pop   A
   reti

; end of file DelSigINT.asm
```

En rajoutant cette :ligne, on appelle directement la fonction TX et on charge son registre dès que l'échantillon est pris.

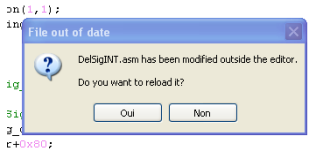
Il nous faut modifier le main puisque certaines lignes ne sont plus indispensables :
Remettre l'affichage en fonctionnement.

```
//-------------------------------------------------------------------------
---
// C main line
//-------------------------------------------------------------------------
---

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules


void main()
{
   BYTE valeur =0;
   PGA_1_Start(PGA_1_HIGHPOWER);
   LCD_1_Start();
   DelSig_Start(DelSig_HIGHPOWER);
   DelSig_StartAD();
   M8C_EnableGInt;
```

```
    TX8_1_Start(0x00);

         LCD_1_Position(0,1);
        LCD_1_PrCString("Tension");
         LCD_1_Position(1,1);
                LCD_1_PrCString("V = Ox");

        while(1)
    {
         //while(DelSig_fIsDataAvailable() == 0);

         valeur=DelSig_cGetDataClearFlag();
                valeur=valeur+0x80;

                LCD_1_Position(1,8);
                LCD_1_PrHexByte(valeur);

                //TX8_1_SendData(valeur);

    }
}
```

A mettre en commentaire

Lors de la compilation, le message suivant apparaît :
Cliquez sur non sinon le fichier précédent est rechargé et les modifications ne sont pas prises en compte.



Mesurez le temps mis pour le transfert de l'octet dans le module TX.
Visualisez le signal transmis à l'afficheur (P2(0) ou P2(1) ou P2(2) ou P2(3)) par rapport aux impulsions délivrées par le Delsig. Q'en concluez-vous ?

Attention :
Afin de diminuer le nombre de fils utilisés pour commander l'afficheur, comme, par exemple lorsqu'on dispose de très peu de broches d'entrées sorties disponibles sur un microcontrôleur, on peut utiliser le mode quatre bits de l'afficheur LCD. Dans ce mode, seuls les 4 bits de poids fort (**D4** à **D7**) de l'afficheur sont utilisées pour transmettre les données et les lire. Les 4 bits de poids faible (**D0** à **D3**) sont alors connectés à la masse. Les données sont alors écrites ou lues en envoyant séquentiellement les quatre bits de poids fort suivi des quatre bits de poids faible. Une impulsion positive d'au moins 450 ns doit être envoyée sur la ligne E pour valider chaque demi-octet ou nibble.

Passez maintenant le CPU_Clock à SysClk/1

Visualisez les signaux P0(5) et P2(7).
Y a-t-il une différence avec le précédent relevé ? Expliquez.

Importez le dossier "Liaison_TX_PSoC_vers_PSoC_avec_RX" à partir du portail.

En plus de l'émetteur, le programme comprend un récepteur RX et un DAC. (Normalement, le RX et le DAC se trouve dans un autre PSoC distant).
Reliez la sortie TX à l'entrée RX.
Vérifiez que l'on retrouve bien la tension continue d'entrée à la sortie du DAC.

Remplacez le signal continu d'entrée par un signal sinusoïdal sur la BNC2 en plaçant l'inverseur vers le bas pour le superposer à la tension de 2,5V.
Jusqu'à quelle fréquence d'entrée peut-on aller pour avoir un signal correct à la sortie du DAC ?

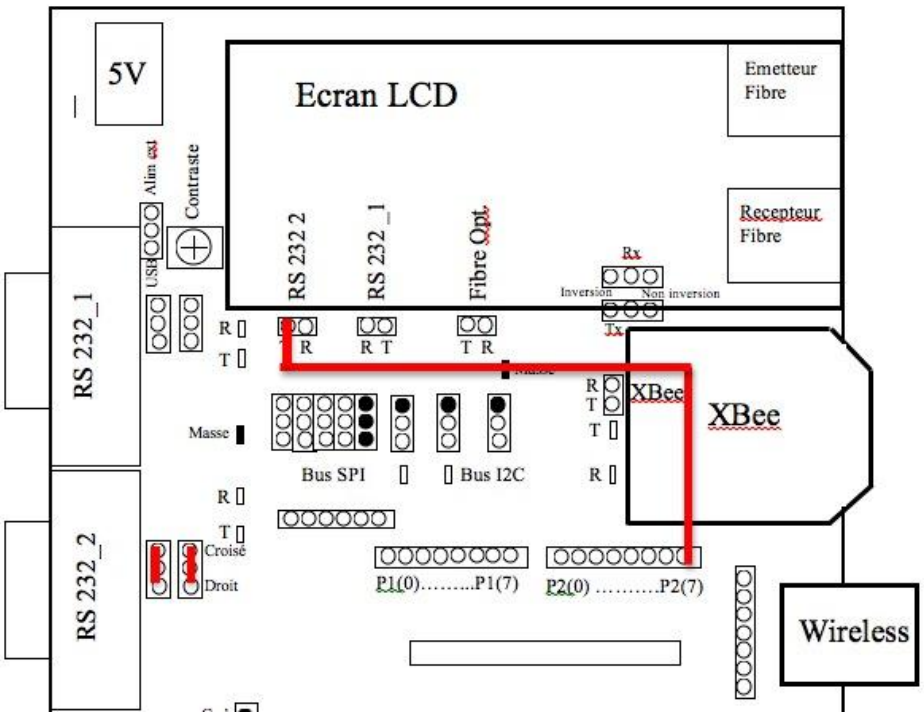
III Transmission PSoC (émetteur TX) vers PC (récepteur)

L'objectif est maintenant d'effectuer une liaison du PSoC vers le PC avec le PSoC émetteur et le PC récepteur. C'est typiquement le genre d'application permettant de centraliser des données issues de systèmes embarqués déportés sur des centrales d'acquisition.

A partir de du projet précédent modifiez le code afin d'avoir celui ci-dessous :

```
void main()
{
   int i;
   BYTE valeur =0;
   PGA_1_Start(PGA_1_HIGHPOWER);
   LCD_1_Start();
   CAN_Start(CAN_HIGHPOWER);
   CAN_StartAD();
   M8C_EnableGInt;

   LCD_1_Position(0,1);
   LCD_1_PrCString("Tension");
   LCD_1_Position(1,1);
   LCD_1_PrCString("V = Ox");

   while(1)
   {
      //while(CAN_fIsDataAvailable() == 0);

      //valeur=CAN_cGetDataClearFlag();
            valeur=valeur+0x80;
      valeur = 0x45 //lettre E
      //temporisation
      while(i<10000)
      {
         i++;
      }
      i=0; //réinitialisation du compteur de boucle
            LCD_1_Position(1,11);
            LCD_1_PrHexByte(valeur);
          TX8_1_SendData(valeur);

   }
}
```

Effectuer les branchements comme sur le schéma suivant et relier le connecteur au port série par un câble droit.



Sur le PC, à l'aide du logiciel du picoscope, observez le signal sur la broche P2_7. Quelle est la tension correspondant à un bit « 1 » et celle correspondant à un bit « 0 » ?

Le signal en sortie du PSoC transite à travers un composant MAX 232. Observez à l'aide du picoscope le signal envoyé sur le PC. Quels sont alors les niveaux de tension correspondant aux bits « 0 » et aux bits « 1 » ?

En allant dans le menu Outils -> Serial Decoding. Dans l'onglet protocols choisissez RS232
En réglant le trigger sur Seul et sur la voie visualisez effectuez une acquisition des données transférées du PSoC au PC. Vérifier que vous retrouvez bien la valeur indiquée dans le code.

Décommenter les lignes

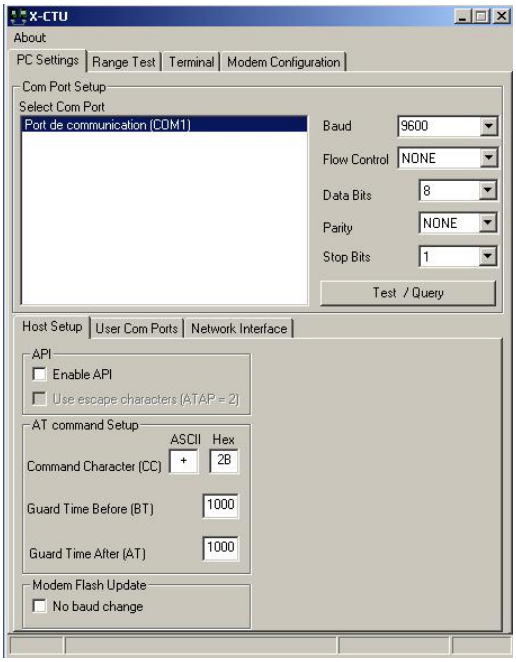
```
        //while(CAN_fIsDataAvailable() == 0);
    //valeur=CAN_cGetDataClearFlag();
```

et commentez celle-ci

```
    valeur = 0x45 //lettre E
```

Refaîtes une acquisition avec le picoscope de la donnée. Retrouvez la valeur transférée du PSoC au PCen indiquant le bit de start, la valeur de chaque bit, et le bit de stop.

Lancer X-CTU (l'icône est sur le bureau) et paramétrer de la manière suivante :



Aller dans l'onglet Terminal, cliquez sur le bouton . Show hex

Qu'observez-vous ?

Revenez dans l'onglet PC Settings et changer la vitesse de transmission. Retourner dans l'onglet Terminal, **Expliquez vos observations.**