

Mise en oeuvre de l'UART

Universal **A**synchronous **R**eceiver/ **T**ransmitter



1 Description de la configuration utilisée

Nous allons ajouter la fonctionnalité de la transmission via une liaison série d'informations entre un projet PSoC et une interface homme machine réalisée en Delphi, matlab, ou autre ...

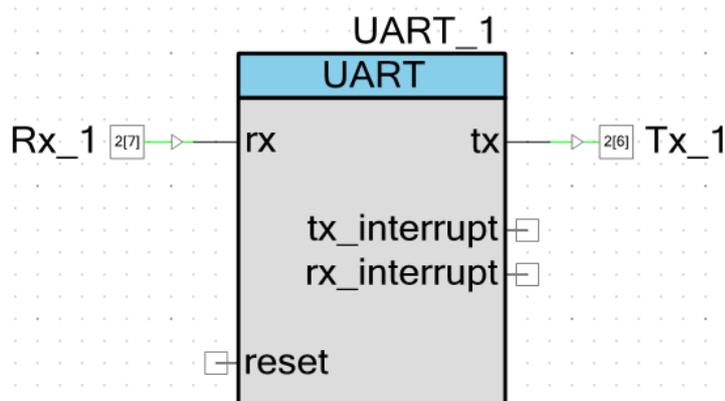
La liaison série est toujours utilisée car :

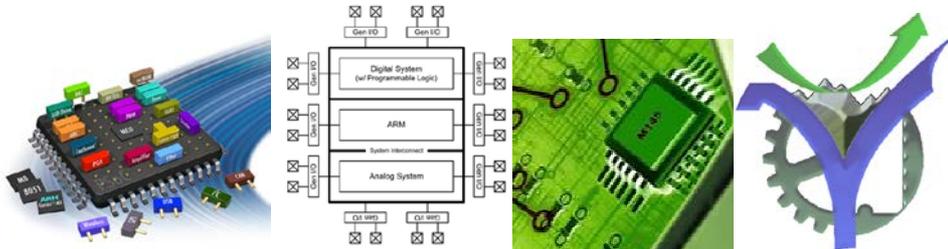
- de nombreux périphériques utilisent cette interface pour communiquer avec un microprocesseur maître : shield arduino lecteur mp3, serveur web, écran OLED ...
- d'autre part les logiciels comme matlab, proteus peuvent recevoir des informations via ce mode de transmission.
- les pilotes de périphériques USB sont reconnus comme des ports RS232.

Nous allons donc ajouter un périphérique UART à un projet existant, le projet retenu pour cet exemple sera le projet avec une mesure de température avec un capteur LM75. Nous allons ajouter la transmission de la température vers un ordinateur dans une interface homme machine réalisée en Delphi, un essai de réception dans matlab sera également réalisé

2 Ajout de l'UART

Dans le catalogue des composants glisser déposer une UART sur le schéma :

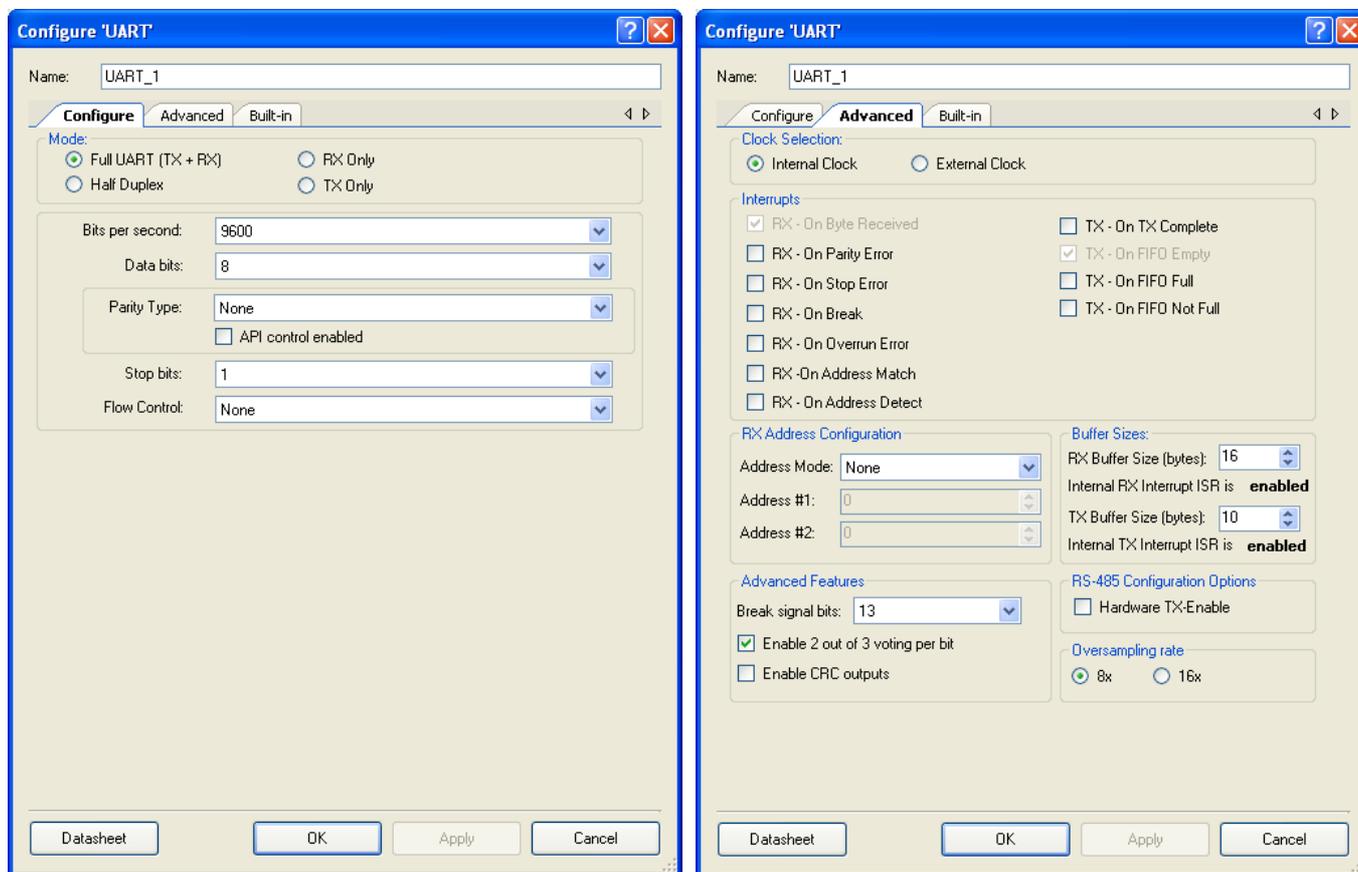




Il faut maintenant la configurer conformément à son usage dans notre projet, pour la liaison série les caractéristiques retenues sont :

- 9600 bauds
- 8 bits de données
- pas de parité
- 1 bit de stop.

La configuration est donnée ci-dessous :

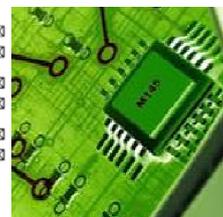
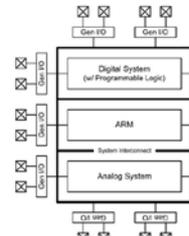


Les entrées sorties Tx et Rx doivent être assignées conformément à la configuration matérielle utilisée, pour la carte PSOCVOX prendre :

Tx Port 2[6]

Rx Port 2[7]

Formation PSoC Formation PSoC



Le composant doit être initialisé :

```
70 | |     UART_1_Start();
```

Dans notre projet la température est lue sur deux octets, la définition suivante permet de travailler à la fois sur un mot de 16 bits ou bien sur les octets mot.hi et mot.low :

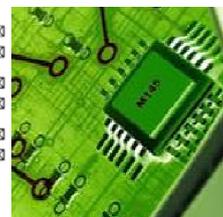
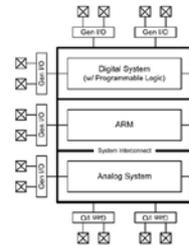
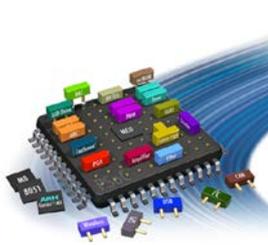
```
36 | |     uint8 status, wbuffer[1], rbuffer[1];
37 | |     uint16 Temp;
38 | |
39 | |     union composite {
40 | |         uint16 mot;
41 | |         struct {
42 | |             char lo;
43 | |             char hi;
44 | |         } octet;
45 | |     };
46 | |     union composite Temp2;
```

Lors de l'acquisition le mot complet est recopié dans Temp2 :

```
96 | |         // Traitement de la donnée
97 | |         // .Regroupement des deux octets dans un mot
98 | |         // .Recadrage de 5 positions vers la droite ( voir documentation du LM75 )
99 | |         // .Supression des bits non significatifs induits par le décalage
100 | |         Temp = rbuffer[0] * 256 | rbuffer[1];
101 | |         Temp = Temp >> 5;
102 | |         Temp = Temp & 0x0EFF;
103 | |         // Recopie dans la donnée utilisée pour l'UART
104 | |         Temp2.mot = Temp;
```

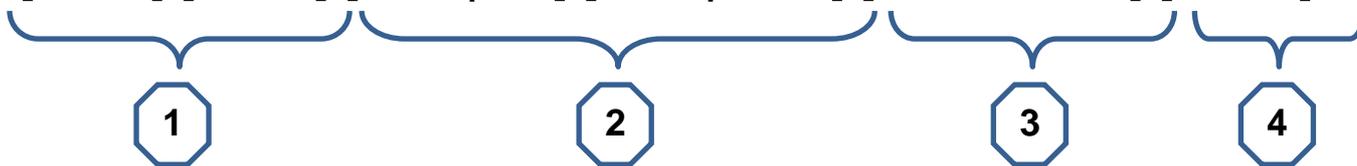
Il suffit ensuite d'envoyer les résultats :

```
120 | |         // Envoi sur l'UART
121 | |         // .Octet identification de l'échange 0x40
122 | |         // .Adresse capteur
123 | |         // .Valeur hi du uint16 Température
124 | |         // .Valeur lo
125 | |         // .Calcul et envoi d'un checksum
126 | |         // .Envoi du caractère de fin 0x0D
127 | |         UART_1_WriteTxData(0x40);
128 | |         UART_1_WriteTxData(LM75);
129 | |         UART_1_WriteTxData(Temp2.octet.hi);
130 | |         UART_1_WriteTxData(Temp2.octet.lo);
131 | |         val = 0x40 ^ LM75 ^ Temp2.octet.hi ^ Temp2.octet.lo;
132 | |         UART_1_WriteTxData(val);
133 | |         UART_1_WriteTxData(0x0D);
```



La trame envoyée est structurée de la manière suivante :

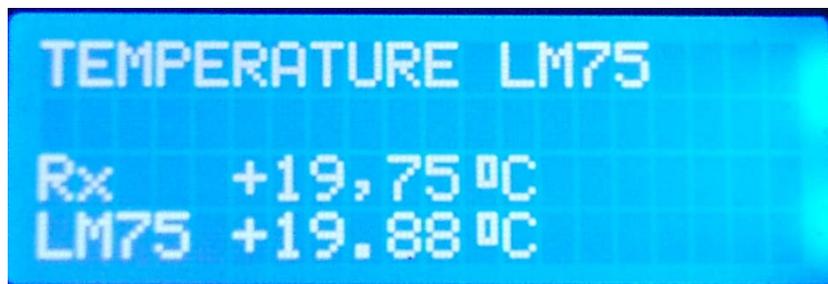
[0x40] [0x4F] [Temp.hi] [Temp.low] [Checksum] [0x0D]



- 1 : En tête
- 2 : Valeur de la température
- 3 : Checksum ou exclusif entre les quatre premiers octets du message
- 4 : Terminaison

3 Réception de caractères

Pour essayer la réception de caractère la valeur de température reçue par une IHM sous Delphi est ensuite, à la demande par appui sur un bouton de cette interface, envoyée sur la platine PSoC. Le résultat est ensuite affiché :

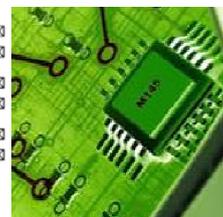
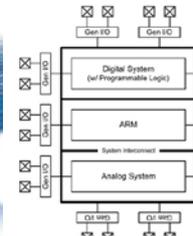


Les caractères reçus sont stockés dans le buffer de réception de l'UART, la fonction `UART_GetChar` renvoi le dernier caractère si disponible :

`uint8 UART_GetChar(void)`

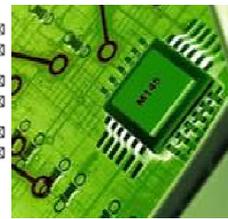
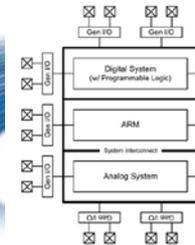
- Description:** Returns the last received byte of data. `UART_GetChar()` is designed for ASCII characters and returns a `uint8` where 1 to 255 are values for valid characters and 0 indicates an error occurred or no data is present.
- Parameters:** void
- Return Value:** `uint8`: Character read from UART RX buffer. ASCII character values from 1 to 255 are valid. A returned zero signifies an error condition or no data available.
- Side Effects:** None

Formation PSoC Formation PSoC

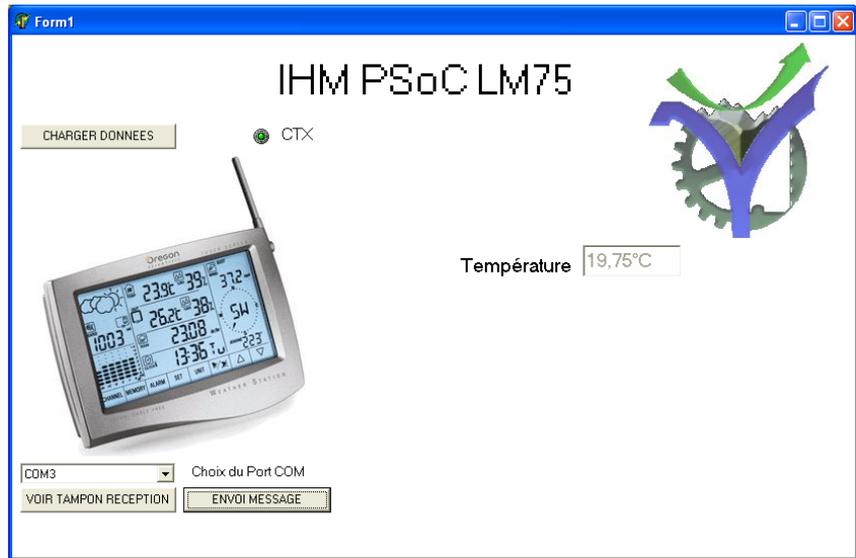


Les caractères sont stockés dans un tableau et affichés uniquement lors de la réception du dernier caractère de terminaison 0x0D :

```
139 | /* Check the UART status */
140 | ch = UART_1_GetChar();
141 |
142 | /* If byte received */
143 | if( ch > 0)
144 | {
145 |     tampon[count]=ch;
146 |     count++;
147 | }
148 |
149 | if ( ch == 0x0D )
150 | {
151 |     CharLCD_Position(2,0);
152 |     for (i=0;i<count-1;i++)
153 |     {
154 |         CharLCD_PutChar(tampon[i]);
155 |     }
156 |     CharLCD_PutChar(CharLCD_CUSTOM_0);
157 |     CharLCD_PutChar('C');
158 |     for (i=count+3;i<16;i++)
159 |     {
160 |         CharLCD_PutChar(' ');
161 |     }
162 |
163 |     count=0;
164 | }
```



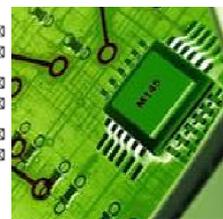
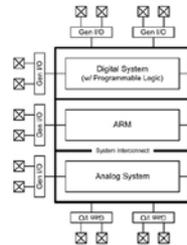
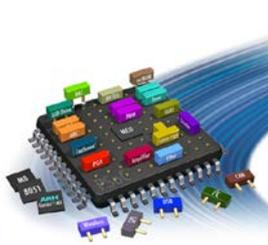
Synoptique général



↑
↓
RS232



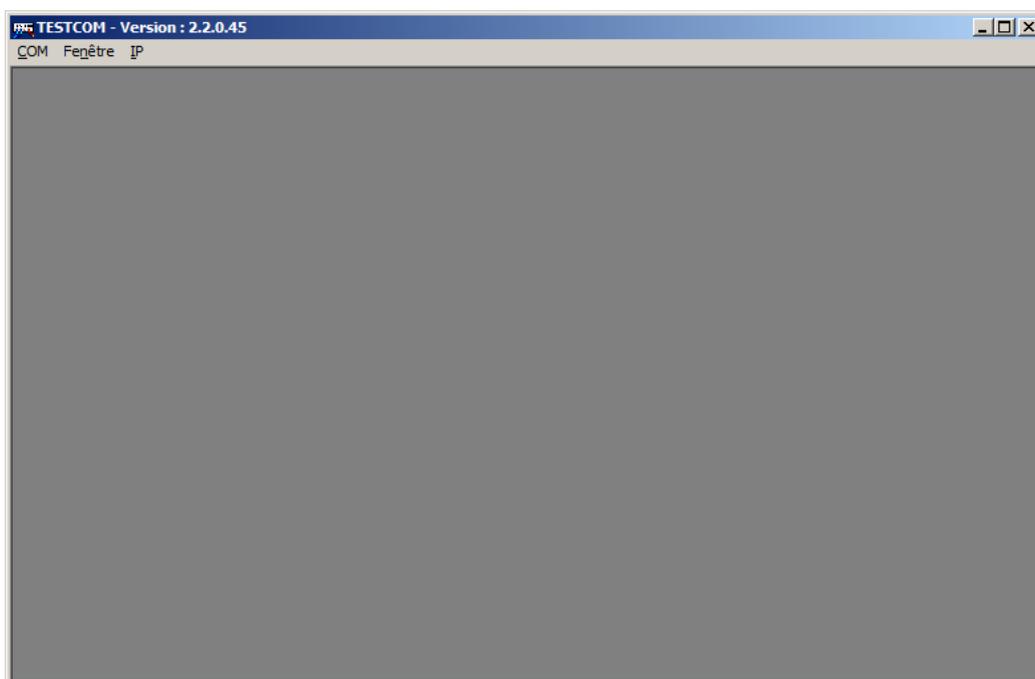
Platine PSoC



SNIFFER de liaison série

Pour observer la liaison série il est utile d'utiliser des logiciels nommés 'sniffer', ici nous utilisons le logiciel TestComVox, ce logiciel permet d'observer le flux sur la liaison série en réception et en émission, dans notre cas uniquement en réception.

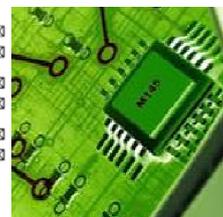
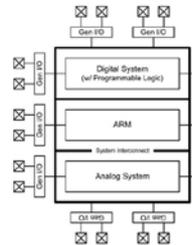
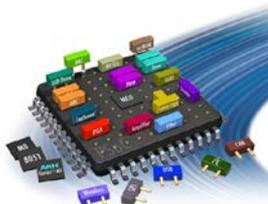
Lancement du logiciel



Ouvrir une fenêtre sur un port série RS232



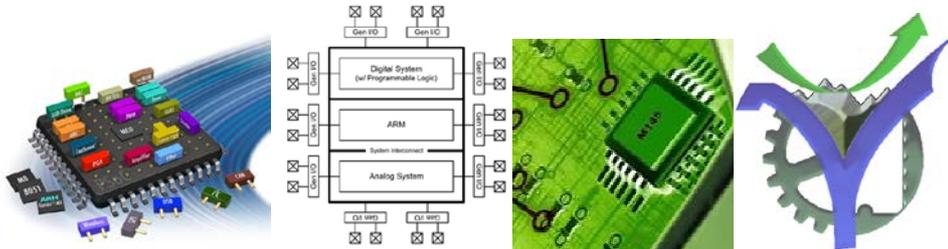
Formation PSoC Formation PSoC



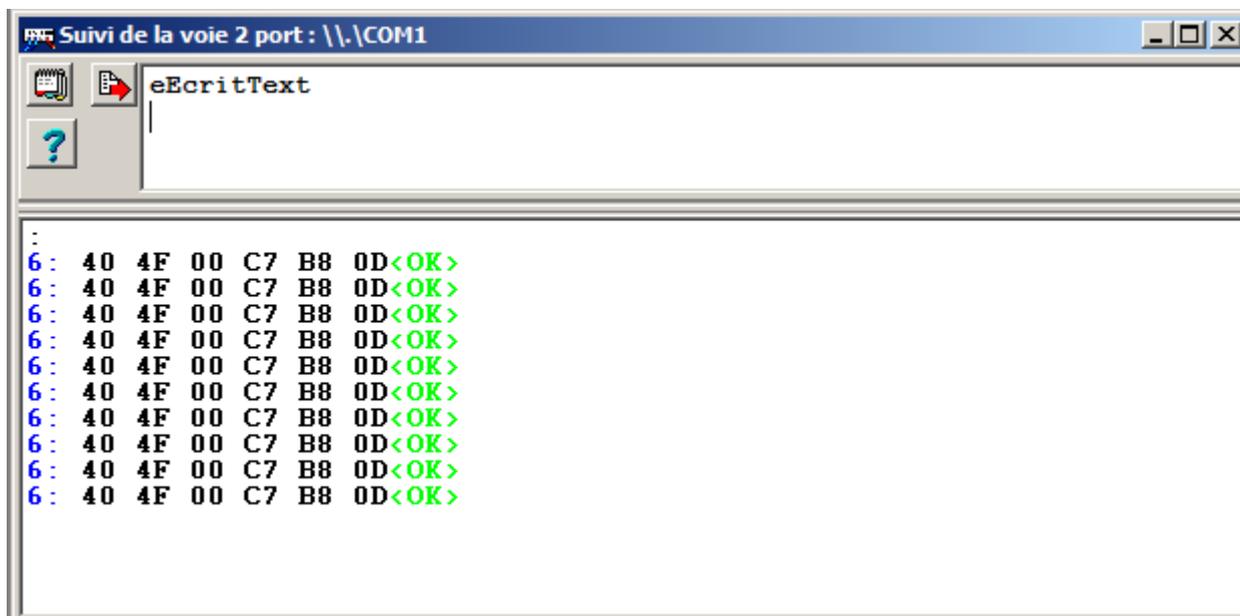
Il faut configurer la liaison

The image shows two overlapping dialog boxes from the TESTCOM software. The top dialog, titled 'Sélection des paramètres d'une voie', is for a generic port. It has a dropdown menu for 'Nom du port Windows à ouvrir :'. The settings are: Vitesse (9600), Longueur (8 bits), Parité (Sans), Arrêt (1 stop), Time outs (Global: 1000, Caractère: 0). Mode de capture is set to 'Sur longueur' with a value of 10. The 'Ecriture' section has several checkboxes, all of which are unchecked. The bottom dialog is also titled 'Sélection des paramètres d'une voie' but is specifically for 'COM1'. Its settings are: Vitesse (9600), Longueur (8 bits), Parité (Sans), Arrêt (1 stop), Time outs (Global: 1000, Caractère: 0). Mode de capture is set to 'Sur caractère de stop' with 'Caractère' set to CR. The 'Ecriture' section has several checkboxes, all of which are unchecked. Both dialogs have 'OK' and 'Annuler' buttons at the bottom.

Formation PSoC Formation PSoC



Nous pouvons alors observer les trames reçues, la mise en page est faite sur le caractère de fin de trame envoyé CR Carriage Return code 0x13

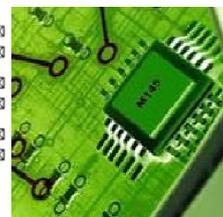
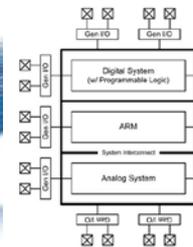
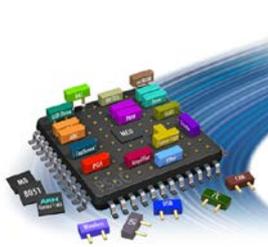


Le code source de la trame est indiqué ci-dessous :

```
120 // Envoi sur l'UART
121 // .Octet identification de l'échange 0x40
122 // .Adresse capteur
123 // .Valeur hi du uint16 Température
124 // .Valeur lo
125 // .Calcul et envoi d'un checksum
126 // .Envoi du caractère de fin 0x0D
127 UART_1_WriteTxData(0x40);
128 UART_1_WriteTxData(LM75);
129 UART_1_WriteTxData(Temp2.octet.hi);
130 UART_1_WriteTxData(Temp2.octet.lo);
131 val = 0x40 ^ LM75 ^ Temp2.octet.hi ^ Temp2.octet.lo;
132 UART_1_WriteTxData(val);
133 UART_1_WriteTxData(0x0D);
```

Pour les curieux on obtient la température par la relation :

$Temp2 \cdot 0,125$ d'où $\theta = ?$



Communication avec matlab

Matlab peut accéder à la liaison série, il faut créer un objet 'COM', le configurer puis le lire comme un périphérique voir l'exemple ci-dessous :

```
Essai_serie.m
1  % Lecture des données températures via RS232
2  % Les données ont le format TP LM75
3  %
4  clear all
5  close all
6  s = serial ('COM1');
7  set(s,'BaudRate',9600);
8  set(s,'FlowControl','none');
9  set(s,'Terminator',13);
10 set(s,'Parity','none');
11 set(s,'StopBits',1);
12 fopen(s);
13 trame=fread(s,40);
14 fclose(s);
```

trame <6x1 double>		
	1	2
1	64	
2	79	
3	0	
4	152	
5	151	
6	13	
8		

La table trame contient les caractères reçus :

Vérifier que nous avons la bonne trame à savoir :

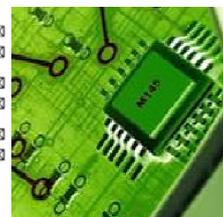
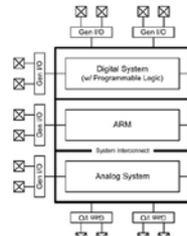
40 4F 00 98 97 0D

Il reste à exploiter ces résultats pour retrouver la température.

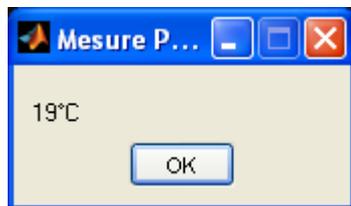
```
// Envoi sur l'UART
// .Octet identification de l'échange 0x40
// .Adresse capteur
// .Valeur hi du uint16 Température
// .Valeur lo
// .Calcul et envoi d'un checksum
// .Envoi du caractère de fin 0x0D
UART_1_WriteTxData(0x40);
UART_1_WriteTxData(LM75);
UART_1_WriteTxData(Temp2.octet.hi);
UART_1_WriteTxData(Temp2.octet.lo);
val = 0x40 ^ LM75 ^ Temp2.octet.hi ^ Temp2.octet.lo;
UART_1_WriteTxData(val);
UART_1_WriteTxData(0x0D);
```

Trame émise
Par le PSoC

Formation PSoC Formation PSoC



Exploitation des résultats



Trame reçue

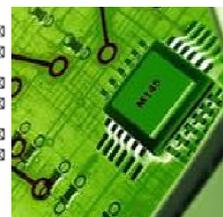
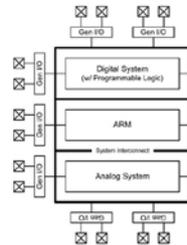
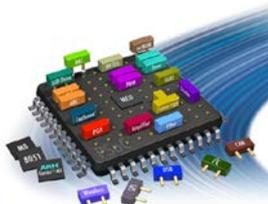
Voilà le code complet :

```

% Lecture des données températures via RS232
% Les données ont le format TP LM75
% 0x40 LM75 Temp.hi Temp.lo checksum Terminaison
%
clear all
close all
s = serial ('COM3');
set(s, 'BaudRate', 9600);
set(s, 'FlowControl', 'none');
set(s, 'Terminator', 13);
set(s, 'Parity', 'none');
set(s, 'StopBits', 1);
fopen(s);
trame=fread(s, 6);
fclose(s);

% Affichage du résultat dans une fenêtre de dialogue
%
% calcul de la valeur de la température
temp = ( (trame(3)*16 + trame(4))*0.125 );
% conversion en chaîne de caractère
strtemp = num2str(temp);
% Préparation du texte final
texte = strcat ( strtemp, '°C');
% Affichage dans une boîte de dialogue
msgbox(texte, 'Mesure PSoC')
    
```

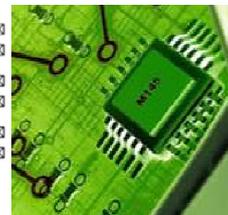
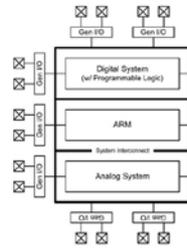
trame <6x1 double>		
	1	2
1	64	
2	79	
3	0	
4	152	
5	151	
6	13	
7		
8		



4 Source du programme

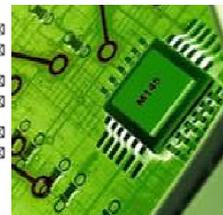
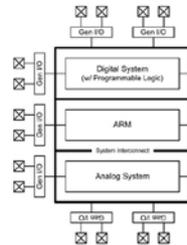
```
/* =====  
*  
* LYCEE VAUCANSON  
*  
* P.G juillet 2012  
*  
* Exemple de lecture d'un capteur I2C LM75  
* utilisation du bus I2C  
* Traitement et affichage sur le LCD  
* Envoi sur l'UART pour reprise par une IHM  
* sur un port com  
*  
* Pour l'utilisation d'un capteur LM75 voir sa documentation  
* Le code ci-dessous ne traite pas les températures négatives  
*  
* Utilisation du générateur de caractère intégré à l'API PSoC Creator  
* Pour créer le caractère °  
*  
* Projet : LM75 et Rx OK  
*  
* =====  
*/  
  
#include <device.h>  
#include <stdio.h>  
  
#define PCF8574 0x20  
#define PCF8591 0x49  
#define LM75 0x4F  
#define CONF_PCF8591 0b01000000  
#define APPUYE 0  
#define REPOS 1  
#define LCD_NUM_COLUMNS 16  
  
uint8 status,wbuffer[1],rbuffer[1];  
uint16 Temp;  
  
union composite {  
    uint16 mot;  
    struct {  
        char lo;  
        char hi;  
    } octet;  
};  
union composite Temp2;  
  
// Programme principal  
void main()
```

Formation PSoC Formation PSoC



```
{  
    // Déclaration de variables locales  
    uint8 val,i;  
    char tstr[16];  
    rbuffer[0]=0;  
    rbuffer[1]=0;  
    char8 ch;  
    uint8 count=0;  
    uint8 pos = 0;  
    char8 tampon[16];  
  
    // Adding this line to enable global interrupt  
    CyGlobalIntEnable;  
  
    // Initialisation du composant I2C  
    I2C_Start();  
    UART_1_Start();  
  
    // Affichage du message d'accueil  
    CharLCD_Start();  
    CharLCD_ClearDisplay();  
    CharLCD_PrintString("Hello World !");  
  
    // Attente d'un appui sur BP1 pour débiter  
    CharLCD_Position(1,0);  
    CharLCD_PrintString("Appui sur BP1");  
    while (BP1_Read()== REPOS ) {};  
    CharLCD_ClearDisplay();  
    CharLCD_PrintString("TEMPERATURE LM75");  
  
    val=0x00;  
  
    /* Programme principal en 'boucle' */  
    for(;;)  
    {  
        // Lecture de la température sur le capteur LM75 I2C Read mode complet  
        I2C_MasterReadBuf(LM75, rbuffer,2, I2C_MODE_COMPLETE_XFER );  
        // wait until Transfer is complete  
        while((I2C_MasterStatus() & I2C_MSTAT_RD_CMPLT )==0);  
  
        // Traitement de la donnée  
        // .Regroupement des deux octets dans un mot  
        // .Recadrage de 5 positions vers la droite ( voir documentation du LM75 )  
        // .Suppression des bits non significatifs induits par le décalage  
        Temp = rbuffer[0]*256|rbuffer[1];  
        Temp = Temp >> 5;  
        Temp = Temp & 0x0EFF;  
        // Recopie dans la donnée utilisée pour l'UART  
        Temp2.mot=Temp;  
  
        // Affichage de la valeur de la température  
        // .Positionnement du LCD en ligne 3 colonne 0
```

Formation PSoC Formation PSoC



```
// .Préparation de la chaine de caractère float => Char
// .Affichage sur le LCD
// .Affichage du caractère utilisateur '0' défini en Custom_0
```

```
CharLCD_Position(3,0);
CharLCD_PrintString("LM75 ");
sprintf(tstr, "%+4.2f", 0.125*Temp );
CharLCD_PrintString(tstr);
CharLCD_PutChar(CharLCD_CUSTOM_0);
CharLCD_PutChar('C');

// Envoi sur l'UART
// .Octet identification de l'échange 0x40
// .Adresse capteur
// .Valeur hi du uint16 Température
// .Valeur lo
// .Calcul et envoi d'un checksum
// .Envoi du caractère de fin 0x0D
UART_1_WriteTxData(0x40);
UART_1_WriteTxData(LM75);
UART_1_WriteTxData(Temp2.octet.hi);
UART_1_WriteTxData(Temp2.octet.lo);
val = 0x40 ^ LM75 ^ Temp2.octet.hi ^ Temp2.octet.lo;
UART_1_WriteTxData(val);
UART_1_WriteTxData(0x0D);
```

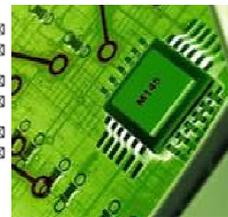
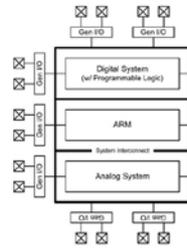
```
// Test de la réception de caractère
```

```
/* Check the UART status */
ch = UART_1_GetChar();

/* If byte received */
if( ch > 0)
{
    tampon[count]=ch;
    count++;
}

if ( ch == 0x0D )
{
    CharLCD_Position(2,0);
    for (i=0;i<count-1;i++)
    {
        CharLCD_PutChar(tampon[i]);
    }
    CharLCD_PutChar(CharLCD_CUSTOM_0);
    CharLCD_PutChar('C');
    for (i=count+3;i<16;i++)
    {
        CharLCD_PutChar(' ');
    }
    count=0;
}
```

Formation PSoC Formation PSoC



```
}  
  
        CyDelay(200);  
    }  
}
```

/* [] END OF FILE */



Indique un document ressource



Retour au sommaire



Retour à la page courante