## Mise en oeuvre de l'UART

⊠-

Universal Asynchronous Receiver/Transmitter



#### 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 :

Formation

Formation PSoC

- 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.

Configure 'UART' **?**× Configure 'UART' ? X Name: UART 1 Name UART\_1 Configure Advanced Built-in 4 ⊳ Configure Advanced Built-in 4 ⊳ Mod Clock Selection Full UART (TX + RX) 🔘 RX Only 🔘 External Clock Internal Clock 🔘 Half Duplex 🔘 TX Only Interrupts TX - On TX Complete 9600 Bits per second: ¥ 🔽 TX - On FIFO Empty RX - On Parity Error Data bits: 8 ¥ 🔲 TX - On FIFO Full RX - On Stop Error Parity Type: None ¥ 📃 TX - On FIFO Not Full RX - On Break API control enabled RX - On Overrun Error Stop bits: 1 RX -On Address Match ¥ RX - On Address Detect Flow Control: None **RX Address Configuration** Buffer Sizes: RX Buffer Size (bytes): 16 \$ Address Mode: None Internal RX Interrupt ISR is enabled Address #1: TX Buffer Size (bytes): 10 \$ Address #2: Internal TX Interrupt ISR is enabled Advanced Features RS-485 Configuration Options 📃 Hardware TX-Enable Break signal bits: 13 ¥ Enable 2 out of 3 voting per bit Oversampling rate Enable CRC outputs 📀 8x 🔘 16x Datasheet ΟK Apply Cancel Datasheet ΟK Cancel

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]



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 {
            char lo;
42
43
            char hi;
44
        } octet;
45 L };
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];
             Temp = Temp >> 5;
101
102
             Temp = Temp & OxOEFF;
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 OxOD
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	<pre>val = 0x40 ^ LM75 ^ Temp2.octet.hi ^ Temp2.octet.lo;</pre>
132	UART_1_WriteTxData(val);
133	UART_1_WriteTxData(OxOD);

PSOC\_Fiche\_UART.docx

# Formation PSoC

La trame envoyée est structurée de la manière suivante :



- 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

PSOC\_Fiche\_UART.docx



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 */
              if(ch > 0)
143
144 📥
              {
                  tampon[count]=ch;
145
146
                  count++;
147
              }
148
149
              if ( ch == OxOD )
150
              {
151
                  CharLCD_Position(2,0);
                  for (i=0;i<count-1;i++)</pre>
152
153
                  {
                       CharLCD PutChar(tampon[i]);
154
155
                  }
                  CharLCD_PutChar(CharLCD_CUSTOM_0);
156
                  CharLCD_PutChar('C');
157
                  for (i=count+3;i<16;i++)</pre>
158
159
                  {
160
                       CharLCD_PutChar(' ');
161
                  }
162
163
                  count=0;
164
              }
```



### Synoptique général



**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

🅦 TESTCOM - Version : 2.2.0.45	
<u>C</u> OM Fe <u>n</u> être IP	

Ouvrir une fenêtre sur un port série RS232





#### Il faut configurer la liaison

COM Eenêtre IP	sion : 2.2.0.45			_ 🗆 X	
COM Fenetre IP	r voie 2 BeritText Sélection des paramètre Nom du port Windows à ouv	es d'une voie		<u> </u>	
	Vitesse         Longue           C 110         C 7 b           C 300         © 8 b           C 1200         Mode a           C 2400         © Sur           C 4800         © Sur           C 1200         C 38400           C 38400         ⊂ Sur           C 56000         ✓ Ne           C 115200         ✓ Ava           C 256000         ✓ Ava	eur Parité Arrêt its Sans Impaire Paire 2 stops de capture r longueur : 10 ÷ r caractère de stop : Caractère : Caractère : CR CLF CAutre : 0 ÷ commencer une réception que si buffer non vic ec affichage chaine ec affichage hexadécimal	Time outs Global 1000 Caractère 0 Ecriture Envoyer caractère Envoyer CR LF en Envoyer CRC16 er Envoyer \$FF - Som Envoyer LRC8 ASI V OK	de début 2 主 de fin 3 😴 fin fin me() en fin Cll en fin Cll en fin	
	Sélection des pa	aramètres d'une voie			×
	Nom du port Wind Vitesse C 110 C 300 C 600 C 1200 C 2400 C 4800 C 9600 C 14400 C 19200 C 38400 C 56000 C 57600 C 115200 C 128000 C 256000	Image: Complexity of the second state of the second sta	Arrêt I stop 2 stops Stère : R F utre : 0 🚖 que si buffer non vide	Time outs Global 1000 Caractère 0 Ecriture Envoyer caractère de début 2 Envoyer caractère de fin 3 Envoyer CR LF en fin Envoyer CRC16 en fin Envoyer \$FF - Somme() en fin Envoyer LRC8 ASCII en fin X Annu	↓ ↓

Gen VO

-8 -8

-8

-8

Gen I/O



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

🗯 Suivi de la voie 2 port : \\.\COM1	
eEcritText	
i       6:       40       4F       00       C7       B8       0D <ok>         6:       40       4F       00       C7       B8       0D<ok>         6:       40       4F       00       C7       B8       0D<ok></ok></ok></ok></ok></ok></ok></ok></ok></ok></ok></ok>	

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 OxOD
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	<pre>val = 0x40 ^ LM75 ^ Temp2.octet.hi ^ Temp2.octet.lo;</pre>
132	UART_1_WriteTxData(val);
133	UART_1_WriteTxData(OxOD);
	Pour les curieux on obtient la
	temperature par la relation :
	Temp2 . 0,125 d'où θ = ?



## 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 :

😑 Essa	ai_serie.m					
1	€ Le	cture des données t	températures via R	S232	•	
2	% Le	s données ont le fo	ormat TP LM75			
3	8					
4	clea	r all				
5	clos	e all			trame <6x1 do	uble>
6	s =	<pre>serial ('COM1');</pre>	_			
	set (	<pre>s, 'Baudkate', 9600);</pre>	;		1	2
0	set (	8, FlowControl, Inc		1	- 64	
10	set	<pre>B, Terminator (15);</pre>		1	тU —-	
11	set (	s, railty, none,,	,	2	/9	
12	fore	5, Stopbits , 1) ,		3	0	
13	tran	= freed(s, 40):		4	152	
14	fclo	(a) ·			102	
	1010	ac(a),		5	151	
				6	13	
La tabl	le trame	contient les caractères	s recus :			
			,			
				Ö		
40	erifier q 0 4F 00	ue nous avons la bonne 98 97 0D	e trame a savoir :			
ll reste retrouv	e à explo ver la tei	viter ces résultats pour mpérature.	// Envoi sur l' // .Octet ident // .Adresse cap // .Valeur hi d // .Valeur lo	UART ifics teur u uir	ation de l'écha nt16 Températur	ange 0x40 re
			<pre>// .Calcul et e // .Envoi du ca UART_1_WriteTxD UART_1_WriteTxD</pre>	nvoi ractè ata(C ata(I	d'un checksum ère de fin OxOI )x40); .M75);	Trame émise Par le PSoC
			UART_1_WriteTxD UART_1_WriteTxD val = 0x40 ^ LM UART_1_WriteTxD UART_1_WriteTxD	ata(1 ata(1 75 ^ ata(v ata(0	<pre>Femp2.octet.hi) Femp2.octet.lo) Temp2.octet.hi ral); DxOD);</pre>	; ; . ^ Temp2.octet.lo;
Lvcée	Vauca	nson PSOC_Fic	che UART.docx			Page 10



Exploitation des résultats

🛃 Mesure P 🔳 🗖 🔀
19°C
ок

Voilà le code complet :

Trame reçue

 $\boxtimes$ 

⊠

 $\boxtimes$ 

-12

% 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 chaine de caractère strtemp = num2str(temp); % Préparation du texte final texte = strcat ( strtemp, '°C'); % Affichage dans une boite de dialogue msgbox(texte,'Mesure PSoC')

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





4 Source du programme

*	
* LYCEE VAUCANSON	
* P.G juillet 2012	
<ul> <li>* Exemple de lecture d'un caj</li> <li>* utilisation du bus I2C</li> <li>* Traitement et affichage sur</li> <li>* Envoi sur l'UART pour repri</li> <li>* sur un port com</li> </ul>	pteur I2C LM75 le LCD ise par une IHM
<ul> <li>* Pour l'utilisation d'un capteu</li> <li>* Le code ci-dessous ne traite</li> </ul>	ur LM75 voir sa documentation e pas les températures négatives
* Utilisation du générateur de * Pour créer le caractère ° *	eractère intégré à l'API PSoC Creator
* Projet : LM75 et Rx OK	
* =====================================	
<pre>#include <device.h> #include <stdio.h></stdio.h></device.h></pre>	
<ul> <li>#define PCF8574 0x20</li> <li>#define PCF8591 0x49</li> <li>#define LM75 0x4F</li> <li>#define CONF_PCF8591 0b0</li> <li>#define APPUYE 0</li> <li>#define REPOS 1</li> <li>#define LCD_NUM_COLUMN</li> </ul>	1000000 √S 16
uint8 status,wbuffer[1],rbuffer uint16 Temp;	[1];
union composite { uint16 mot; struct { char lo; char hi; } octet;	
, ,	
; union composite Temp2;	





// 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 débuter 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 ) // .Supression 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



```
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;
```

PSOC\_Fiche\_UART.docx



CyDelay(200); } }

/\* [] END OF FILE \*/



Indique un document ressource



Retour au sommaire



Retour à la page courante