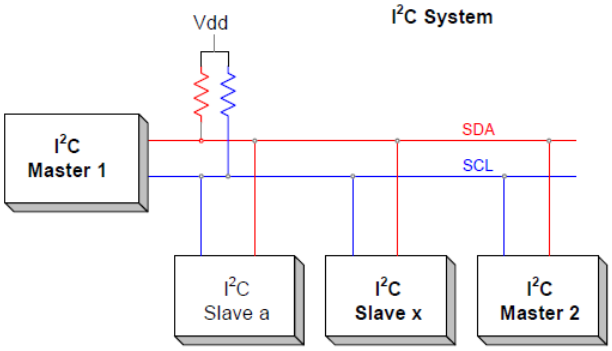
Liaison série I2C

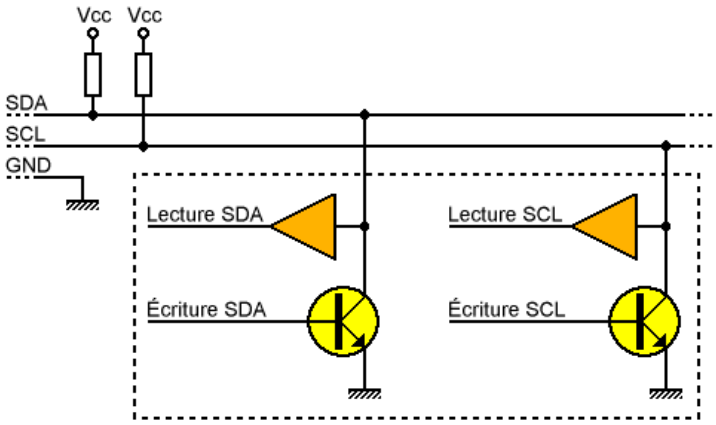
Mise en œuvre d'une liaison I2C

Rappel sur le bus I2C

Contrairement au bus RS232 qui nous permet de faire de la liaison point à point, le bus I2C autorise le dialogue entre plusieurs modules.

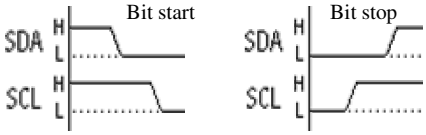


Tous ces modules sont raccordés sur les mêmes lignes SDA et SCL. Il est donc nécessaire, pour les différencier de leur affecter une adresse propre.



Pour la lecture, aucun problème.
Pour l'écriture, tous les transistors de sortie des différents modules sont raccordés à une seule résistance (pull up). Ainsi, seul le module qui émet impose le niveau sur la ligne. Tous les autres modules ayant leur transistor de sortie bloqué sont en état haute impédance et n'ont donc aucune action sur le bus.

Prise de contrôle du bus

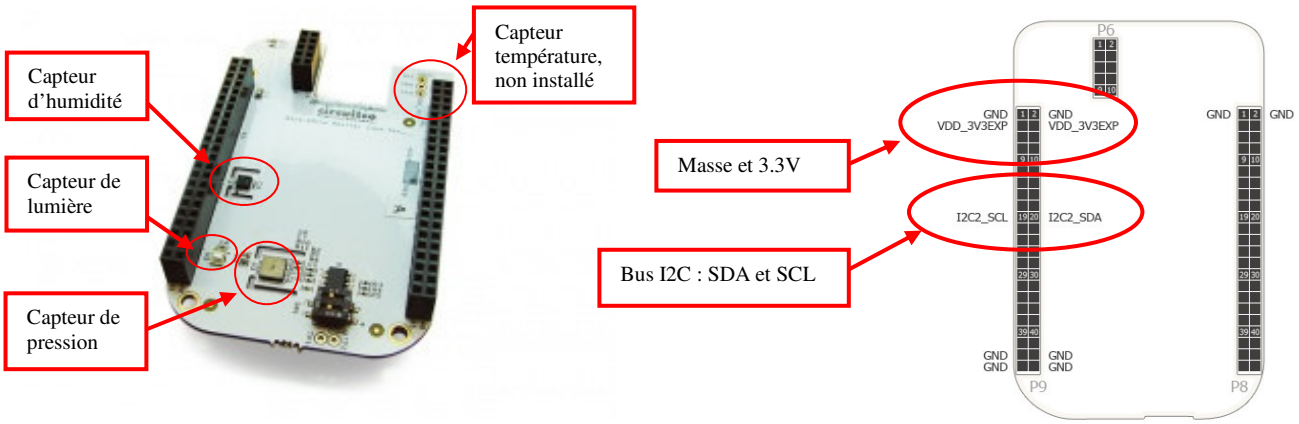


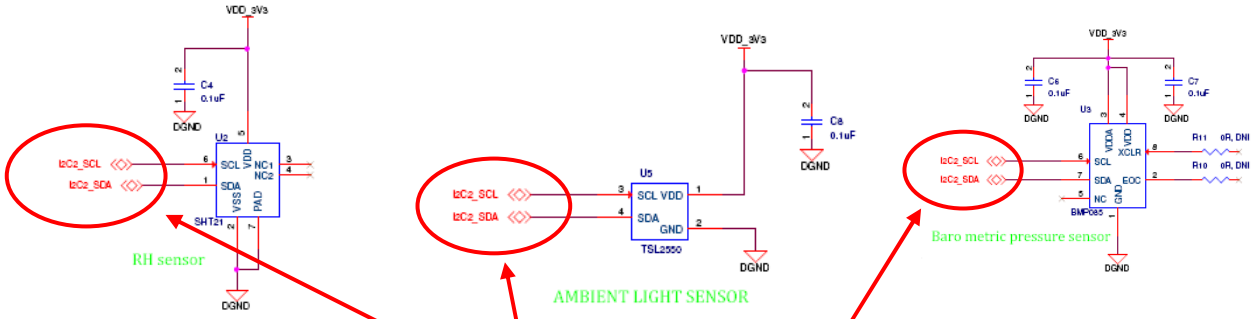
- Le bus doit être au repos avant la prise de contrôle : SDA et SCL à 1.
- Pour transmettre des données, il faut surveiller :
  - La condition de départ : SDA passe à 0, SCL reste à 1
  - La condition d'arrêt : SDA passe à 1, SCL reste à 1
- Après avoir vérifier que le bus est libre, puis pris le contrôle de celui-ci, le circuit en devient le maître : c'est lui qui gère le signal d'horloge.

I- Le maître lit une valeur chez un esclave.

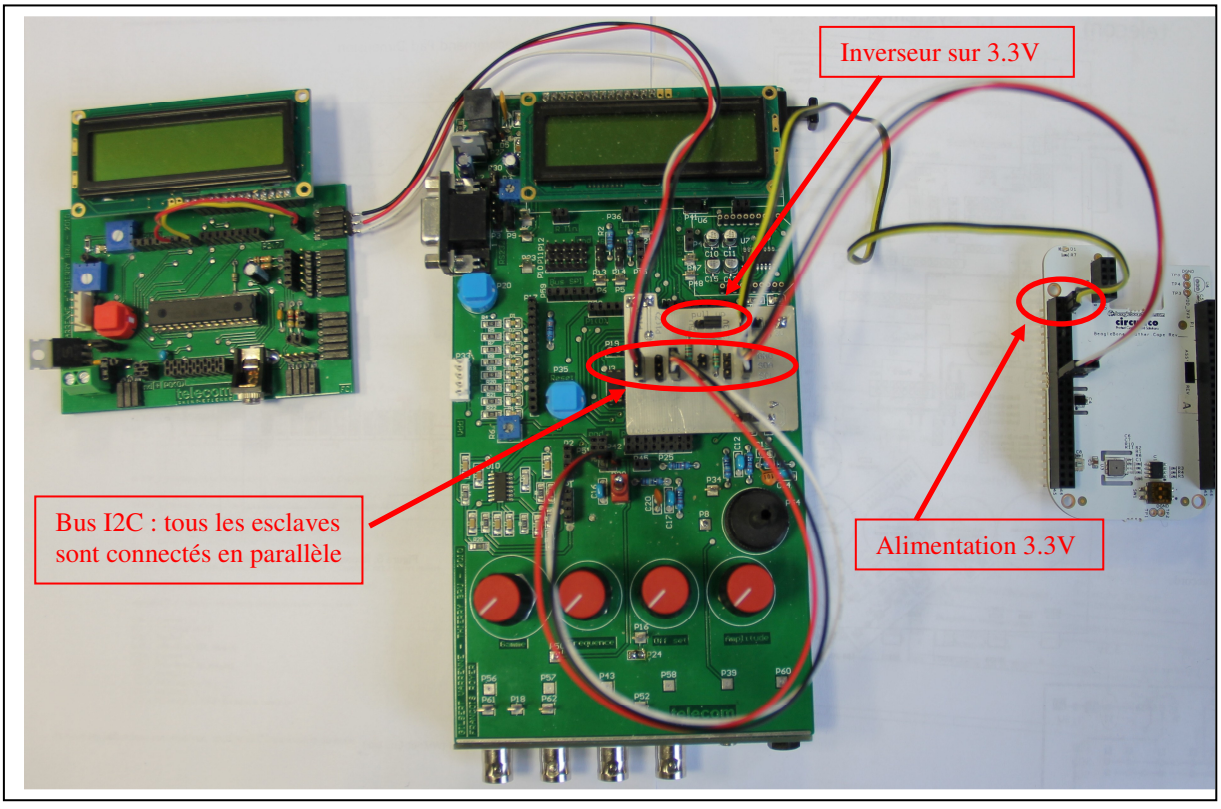
L'esclave est constitué d'un module « circuitco » spécial météo, adaptable sur une carte BeagleBone et comportant plusieurs capteurs. (température, pression, humidité et lumière ambiante). Chaque capteur dispose d'une adresse propre. Ces adresses sont intégrées au capteur et leur valeur est donnée dans la datasheet constructeur. Tous ces capteurs sont reliés à un seul bus I2C.

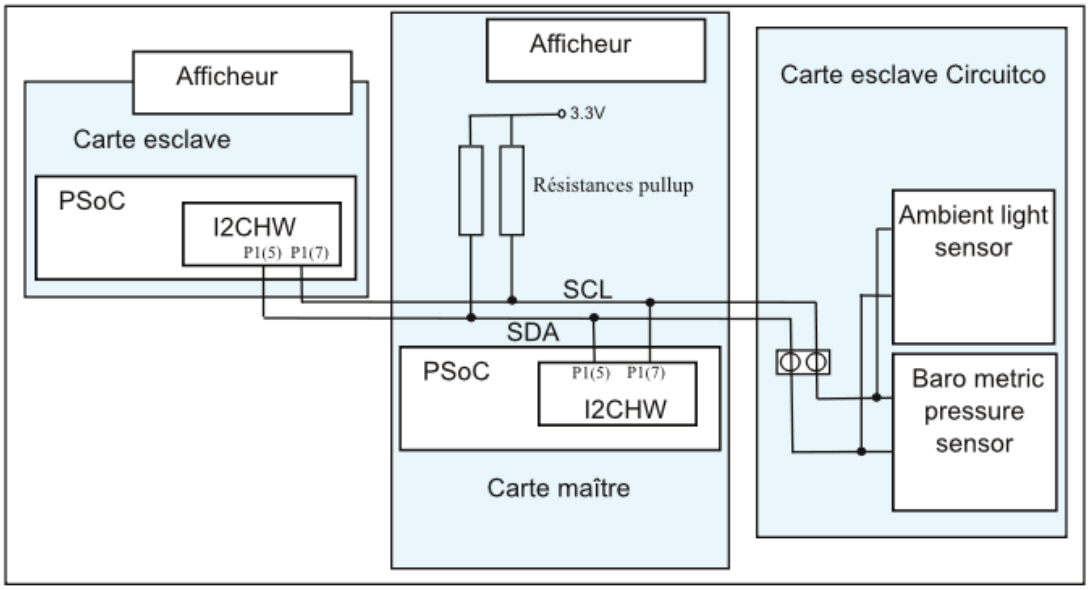
T.P Système Electronique





Tous les capteurs sont reliés au même bus I2C

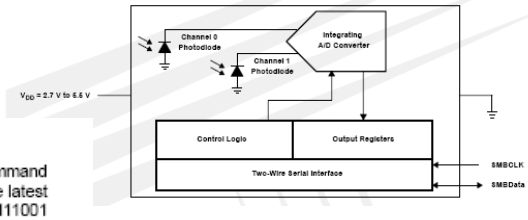




Nous allons relever les informations données par le capteur de lumière TSL2550

Principales informations données dans la datasheet :

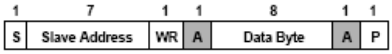
Functional Block Diagram



Digital Interface

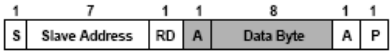
The TSL2550 contains an 8-bit command register that can be written and read via the SMBus. The command register controls the overall operation of the device. There are two read-only registers that contain the latest converted value of each of the two ADC channels. The SMBus slave address is hardwired internally as 0111001 (MSB to LSB, A6 to A0).

Both the *send byte protocol* and the *receive byte protocol* are implemented in the TSL2550. The send byte protocol allows single bytes of data to be written to the device (see Figure 6). The written byte is called the COMMAND byte. The receive byte protocol allows single bytes of data to be read from the device (see Figure 7). The receive data can be either the previously written COMMAND byte or the data from one of the ADC channels. In Figure 6 and Figure 7, the clear area represents data sent by the host and the shaded area represents data returned by the ambient light sensor or slave device.



S = Start Condition P = Stop Condition Shaded = Slave Transmission

Figure 6. Send Byte Protocol



S = Start Condition P = Stop Condition Shaded = Slave Transmission

Figure 7. Receive Byte Protocol

Command Register

The command register is used primarily to:

- Select which ADC register will be read during a read cycle
- Switch the dynamic range of the device between standard and extended range modes
- Power the device up for operation or power it down for minimum power consumption

Table 1 shows the six primary commands used to control the TSL2550.

Table 1. Command Summary

COMMAND	FUNCTION
0x00h	Power-down state
0x03h	Power-up state/Read command register
0x1Dh	Write command to assert extended range mode
0x18h	Write command to reset or return to standard range mode
0x43h	Read ADC channel 0
0x83h	Read ADC channel 1

Vous trouverez la notice complète sur MOOTSE

On utilise :

- Un LCD pour l'affichage. Ne pas oublier de le mettre sur le port 2.
- Un I2CHW pour l'émission et la réception en I2C. Vous le trouverez le module la bibliothèque « Digital Comm ». Il utilise le bloc hardware I2C du PsoC. Vous ne pouvez utiliser qu'un bloc hardware par PsoC donc qu'un utilisateur de ces modules au maximum. Ce module peut être un maître ou un esclave. Lorsque vous placez ce module il n'apparaît pas dans les blocs mais il faut tout de même le configurer. Vous constatez, dans sa fenêtre de configuration, que la vitesse d'émission est de 50khz, 100khz ou 400kHz maximum. (Pour 400khz, le CPU doit être au moins à 12Mhz).
  Sur la ligne de transmission, vous pouvez connecter plusieurs maîtres ou plusieurs esclaves. Vous devez donc affecter une adresse à chacun d'eux pour savoir qui dialogue avec qui.

Placez ces deux modules dans PSoC Designer. Le module I2CHW doit être un maître. Régler sa vitesse de transmission à 100kHz standard. Choisir les sorties P1(5) et P1(7), la carte fille étant pré câblée ainsi.

Générez les fichiers puis écrire le « main » Vérifiez que vous utilisez les mêmes noms.

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

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

#define adresse_esclave 0x39

        BYTE comlumiere ;
        BYTE lumiere;

void main(void)
{

        LCD_1_Start();
        LCD_1_Position(0,0);
        LCD_1_PrCString("lumiere=");
        M8C_EnableGInt;
        I2CHW_1_Start();
        I2CHW_1_EnableMstr();
        I2CHW_1_EnableInt();


        while(1)

                {

        comlumiere=0x83;
                I2CHW_1_bWriteBytes(adresse_esclave,&comlumiere ,1, I2CHW_1_CompleteXfer); // commande lecture lumière
                I2CHW_1_fReadBytes(adresse_esclave,&lumiere,1, I2CHW_1_CompleteXfer); // lecture lumière

                LCD_1_Position(0,9);
                LCD_1_PrHexByte(lumiere);

                }

}
```

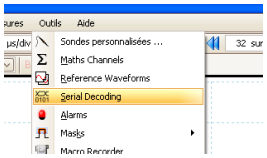
Adresse esclave indispensable pour que le module réponde.

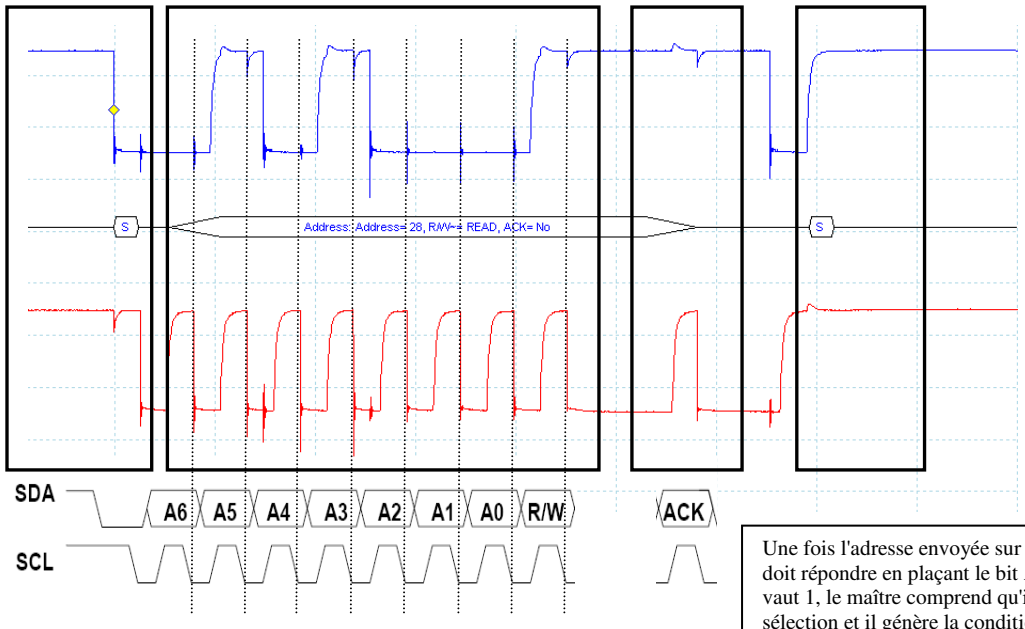
Démarrage de l'I2C et du LCD et activation des interruptions

On demande la lecture en boucle, de la donnée délivrée par le module ayant cette adresse puis on l'affiche sur le LCD.

I-1 Dans les données du constructeur, retrouvez la valeur de l'adresse esclave.

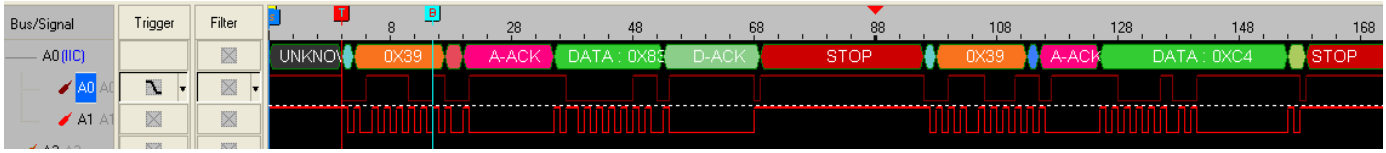
I-2 Sans brancher le module esclave, relevez à l'aide du picoscope les oscillogrammes de SDA et CLK. Faire faire l'analyse de la trame par le picoscope (amplitude 5v, balayage 20µs). Mettre le balayage sur Stop et retrouvez l'organisation binaire de cette trame.



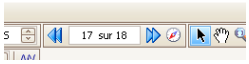


Une fois l'adresse envoyée sur le bus, l'esclave concerné doit répondre en plaçant le bit ACK à 0. Si le bit ACK vaut 1, le maître comprend qu'il y a une erreur de sélection et il génère la condition arrêt. En revanche, si le bit ACK vaut 0, le maître peut continuer les opérations.

Branchez le module suiveur puis relevez de nouveau la trame.



L'analyse de la trame du bus I2C nous donne le résultat ci-dessus. (Ne pas tenir compte des valeurs hexadécimales qui peuvent changer).
Mettre la base de temps sur 50µs , la synchronisation sur front descendant en début de balayage avec un niveau correct. Mettre le balayage sur stop et recherchez l'image qui vous permet de voir toute la trame.



I-3 Retrouvez ces informations en binaire sur l'oscilloscope USB et relevez précisément les acquittements.
.

Conserver ce branchement pour la suite du TP

En fin de TP nous lirons l'information de deux capteurs de la même carte.

II- Le maître écrit une valeur dans un esclave.

Nous conservons la lecture du module esclave sur le bus I2C mais, dans le même temps, nous allons transférer une valeur numérique sur le même bus afin de l'afficher sur un second module esclave.

- Dans le premier cas on lit une valeur provenant de l'esclave
- Dans le second cas on écrit une valeur dans l'esclave.

Configuration du maître :

Outre les modules I2CHW et LCD déjà existants, vous devez rajouter un PGA et un DelSig (8 bits) pour générer la valeur numérique que vous afficherez sur le module esclave.
Entrez la valeur analogique en P0(7). Vous utiliserez la valeur continue de la carte ajustable grâce au potentiomètre ou une valeur continue fournie par le générateur de votre oscilloscope. Dans ce cas, entrez cette valeur sur le BNC2 sans offset. Gain du PGA à 1.
Sortir l'impulsion d'interruption du PWM sur P0(5).

lobal Resources - suiveur_de_ligne	
Power Setting [Vcc /	5.0V / 24MHz
CPU_Clock	SysClk/4
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	6
VC2= VC1/N	16
VC3 Source	VC1
VC3 Divider	156
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref High
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SM	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

CPU à 6Mhz,
Fréquence d'échantillonnage à 1953 Hz

rameters - DelSig	
Name	DelSig
User Module	DelSig
Version	1.30
DataFormat	Unsigned
Data Clock	VC2
ClockPhase	Normal
PosInput	ACB00
NegInput	
NegInputGain	Disconnected
PWM Output	Row_0_Output_1
PulseWidth	127

Après avoir générer les fichiers, écrire le main
Les lignes à rajouter apparaissent en gras:

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

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#define adresse_esclave 0x39
#define slave_address 0x0F
```

Adresse attribuée au module esclave

```
        BYTE comlumiere;
        BYTE lumiere;
        BYTE value;

void main(void)
{

        LCD_1_Start();
```

```
        LCD_1_Position(0,0);
        LCD_1_PrCString("lumiere=");
        M8C_EnableGInt;
        I2CHW_1_Start();
        I2CHW_1_EnableMstr();
        I2CHW_1_EnableInt();
        PGA_1_Start(PGA_1_HIGHPOWER);
        DelSig_Start(DelSig_HIGHPOWER);
        DelSig_StartAD();
        LCD_1_Position(1,0);
        LCD_1_PrCString("Volt en Hex =");

        while(1)

                {

                value=DelSig_cGetDataClearFlag();
                LCD_1_Position(1,14);
                LCD_1_PrHexByte(value);

                I2CHW_1_bWriteBytes(slave_address, &value, 1, I2CHW_1_CompleteXfer);

        comlumiere=0x83;
                I2CHW_1_bWriteBytes(adresse_esclave,&comlumiere ,1, I2CHW_1_CompleteXfer); // commande lecture lumière
                I2CHW_1_fReadBytes(adresse_esclave,&lumiere,1, I2CHW_1_CompleteXfer); // lecture lumière

                LCD_1_Position(0,9);
                LCD_1_PrHexByte(lumiere);

                }

}
```

Démarrage des modules pour étalonner et numériser la valeur analogique

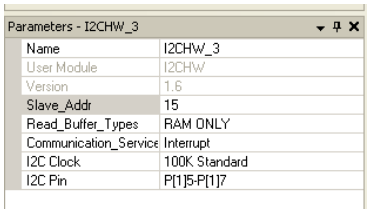
Transfert de la valeur numérique sortie delsig dans le registre value

Ecriture de cette valeur dans le module esclave

Réaliser le récepteur.

Il s'agit d'une mini carte PSoC que vous devez programmer.

Un module I2C esclave et un LCD suffisent pour visualiser la donnée reçue.



Dans les paramètres du module I2C vous devez indiquer l'adresse ainsi que la vitesse de transfert (la même que le module maître).

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

void main()
{
   BYTE reception_donnee;

        I2CHW_3_Start();
        I2CHW_3_EnableSlave();
        M8C_EnableGInt;
        I2CHW_3_EnableInt();
        LCD_1_Start();
        LCD_1_Init();

        LCD_1_Position(0,0);
        LCD_1_PrCString("Data maitre=");

        while(1)
        {

        I2CHW_3_InitWrite(&reception_donnee,1);
        LCD_1_Position(0,14);
        LCD_1_PrHexByte(reception_donnee);

        }
}
```

Réception et affichage de la donnée en boucle

Faire attention à la bonne configuration des ports d'entrée / sortie utilisés.
Toujours démarrer l'esclave avant le maître.
II-1 Sans débrancher le premier module, connectez ce second module sur le bus I2C ; visualisez sur l'oscilloscope la data et la clock et faîtes faire le décodage I2C par l'appareil. La valeur est donnée en hexadécimal.
Reconstituez cette valeur en binaire et placez les bits sur la trame data.
En déduire l'organisation de la trame I2C.
II-2 Dans le module esclave, changez la vitesse de transmission et passez de 100k Standard à 400k fast sans changer celle du maître. Reprogrammez le PSoC de l'esclave. La liaison fonctionne-t-elle ? Qu'en déduisez-vous ?
II-3 Si vous donnez l'adresse 0x03 au module dans lequel on écrit, (n'oubliez pas de changer cette valeur dans le maître et l'esclave) vous constaterez que le décodeur indique : « **Address : Address = 03 (Reserved for future purposes)** ». Il y a en effet des adresses réservées à des modes de fonctionnement particulier qui sont 00000xxx et 111111xx.

III- Influence de la vitesse de transmission :

Visualisez simultanément la ligne SDA et le port P0(5) sur lequel on sort l'impulsion d'interruption du PWM nous indiquant à quel instant l'échantillon est disponible (même impulsion que celle utilisée au TP précédent).
Que constatez-vous et qu'en déduisez-vous ?
Quelles solutions proposer pour un meilleur fonctionnement ?

The I2C clock is based off a SysClk of 24 MHz. If SysClk is less than 24 MHz the I2C clocks will scale down. For example, if SysClk is 6 MHz the possible clock speeds are 12.5K, 25K, and 100K. SysClk is separate from the CPU clock.

Attention : Notice constructeur du TSL2550

Recommended Operating Conditions

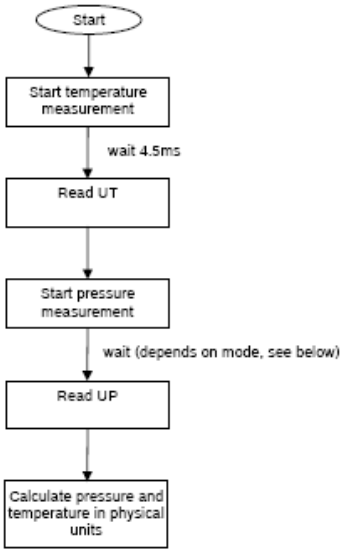
	MIN	MAX	UNIT
Supply voltage, V_{DD}	2.7	5.5	V
Operating free-air temperature, T_A	0	70	°C
SMBus input low voltage @ V_{DD} = 3.3 V ± 5%, V_{IL}		0.8	V
SMBus input high voltage @ V_{DD} = 3.3 V ± 5%, V_{IH}	2.1		V
SMBus operating frequency, $f_{(SMBCLK)}$	10	100	kHz

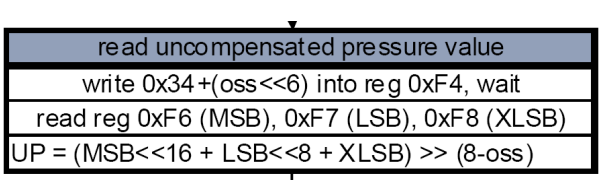
Enregistrez votre projet, vous en aurez besoin éventuellement pour la question subsidiaire (question V).

IV- Lecture de plusieurs capteurs sur le bus I2C

Téléchargez sur MOOTSE le fichier **« Lecture_capteur_carte_circuitco »**

Vous pouvez débrancher la carte secondaire PSoC. Seule la carte circuitco est raccordée au bus I2C.
Le projet ne contient qu'un module I2C maître et un afficheur LCD.





4.2 Device and register address

The BMP085 module address is shown below. The LSB of the device address distinguishes between read (1) and write (0) operation, corresponding to address 0xEF (read) and 0xEE (write).

A7	A6	A5	A4	A3	A2	A1	W/R
1	1	1	0	1	1	1	0/1

There is an easy way to connect two BMP085 to the same I^2C bus: You can use the XCLR input of BMP085 to set one BMP085 part silent while you communicate with the other BMP085 part via I^2C and vice versa. The signals can be provided by two digital outputs of the micro-controller, or one digital output and one inverter.

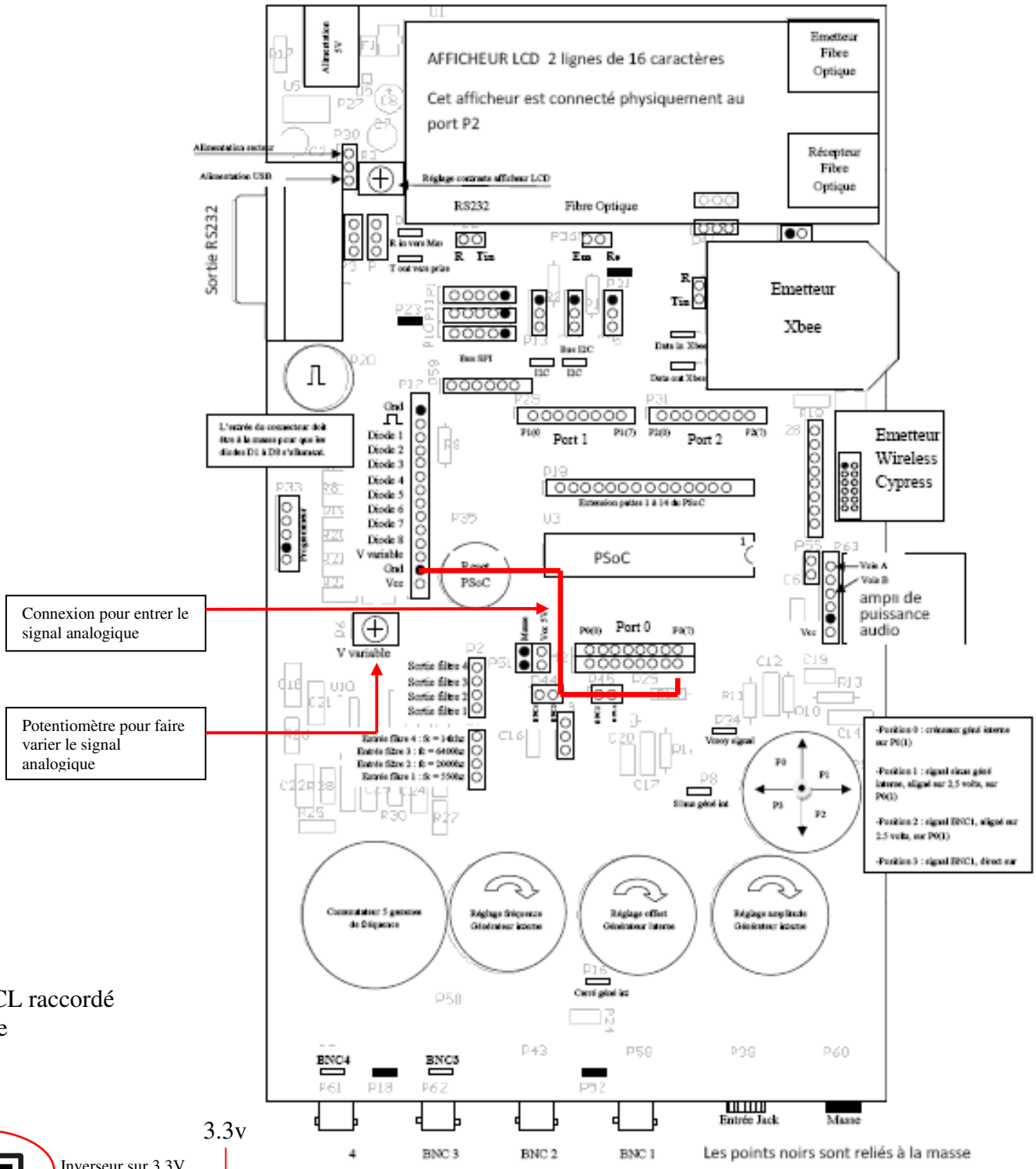
Nous nous contenterons de relever les valeurs de température sans compensation.

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

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#include <stdio.h>
#define adresse_esclave 0x39
#define slave_address 0x77

            BYTE comlumiere;
            BYTE lumiere;
            BYTE comgeneral[2]={0xF4,0x2E};
            BYTE comlecturetemp;

            BYTE MSB_mesure;
            BYTE LSB_mesure;

            int i=0;
            int result;
            char str[6];

void main(void)
{
            LCD_1_Start();
            LCD_1_Position(0,0);
            LCD_1_PrCString("lumiere=");
            M8C_EnableGInt;
           I2CHW_1_Start();
            I2CHW_1_EnableMstr();
            I2CHW_1_EnableInt();

            while(1)
                        {

            comlumiere=0x83;
                        I2CHW_1_bWriteBytes(adresse_esclave,&comlumiere ,1, I2CHW_1_CompleteXfer); // commande lecture lumière
                        I2CHW_1_fReadBytes(adresse_esclave,&lumiere,1, I2CHW_1_CompleteXfer); // lecture lumière

                        LCD_1_Position(0,9);
                        LCD_1_PrHexByte(lumiere);

                        I2CHW_1_bWriteBytes(slave_address, comgeneral ,2, I2CHW_1_CompleteXfer); // commande lecture lumière

            comlecturetemp=0xF6;
                        I2CHW_1_bWriteBytes(slave_address, &comlecturetemp ,1, I2CHW_1_CompleteXfer); // commande lecture lumière

                        for (i=0; i<300
                        ; i++) // boucle d'attente de 65ms au moins pour une portée maximum.
                        {
                        }

                        I2CHW_1_fReadBytes(slave_address, &MSB_mesure,1, I2CHW_1_CompleteXfer); // lecture lumière

            comlecturetemp=0xF7;
                        I2CHW_1_bWriteBytes(slave_address, &comlecturetemp ,1, I2CHW_1_CompleteXfer); // commande lecture lumière

                        for (i=0; i<300
                        ; i++) // boucle d'attente de 65ms au moins pour une portée maximum.
                        {
                        }

                        I2CHW_1_fReadBytes(slave_address, &LSB_mesure,1, I2CHW_1_CompleteXfer); // lecture lumière

                        result= (MSB_mesure << 8) | LSB_mesure; // concatenage poids fort, poids faible

                        csprintf(str,"%d  ",result);// transforme le short int en chaîne de caractères

                        LCD_1_Position(1,0);
                        LCD_1_PrCString("Hex");
                        LCD_1_Position(1,3);
                        LCD_1_PrHexByte(MSB_mesure);
                        LCD_1_Position(1,5);
                        LCD_1_PrHexByte(LSB_mesure);
                        LCD_1_Position(1,8);
                        LCD_1_PrCString("Dec");
                        LCD_1_Position(1,11);
                        LCD_1_PrString(str);// affiche la chaîne de caractères
                        }

}
```

Bibliothèque standard du C déclarant les macros, utile pour la commande csprintf.

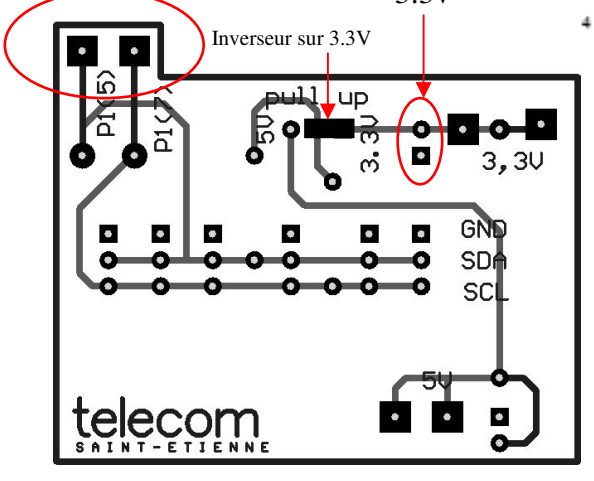
Vue au I° paragraphe

Suivant les indications du constructeur, expliquez l'organisation du main.



SDA, SCL raccordé au maitre

3.3v



Carte fille pour brancher les modules esclaves