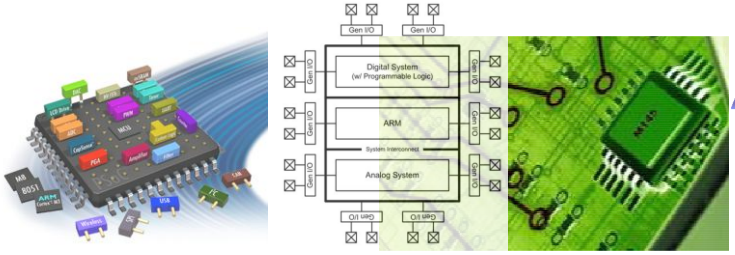
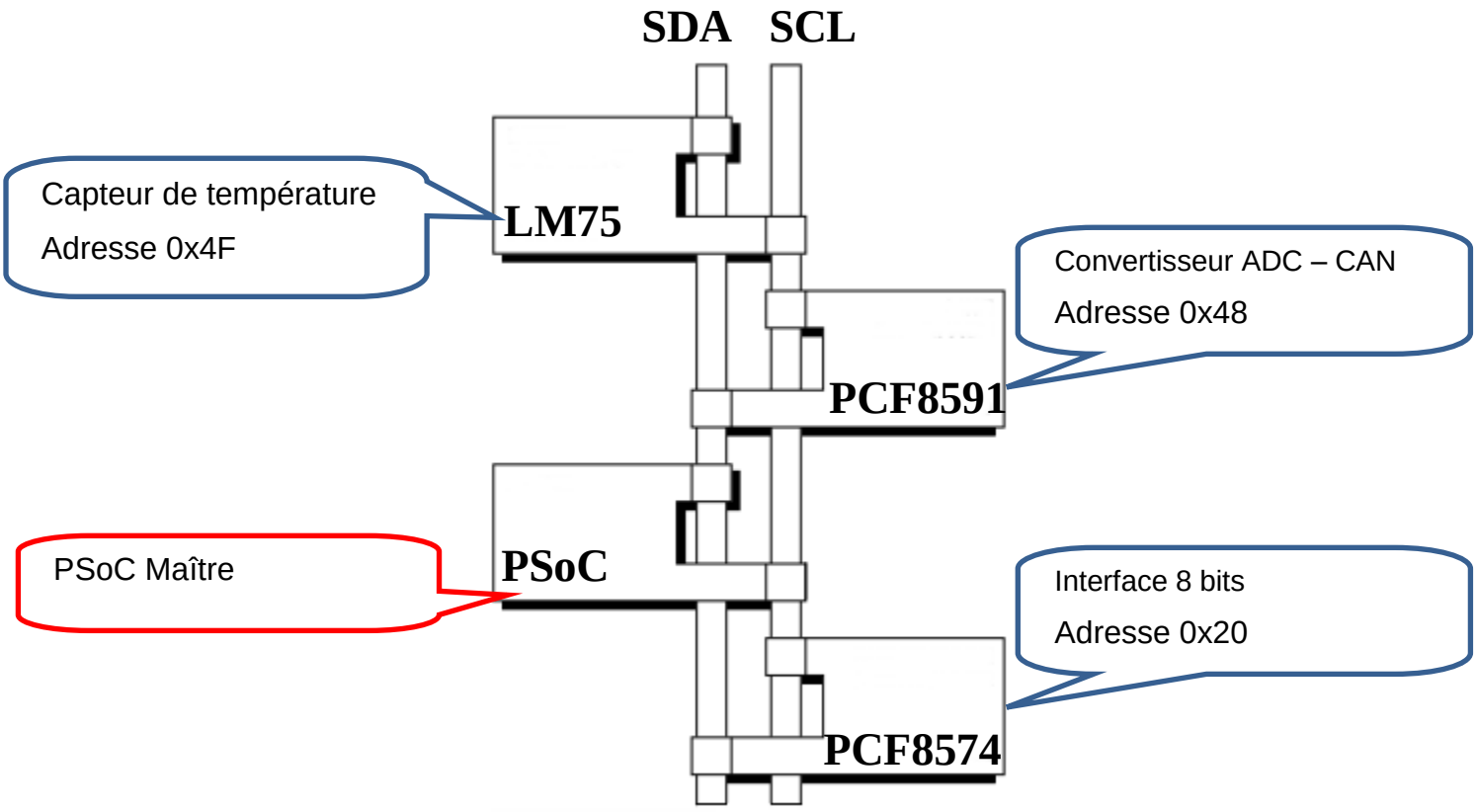


Mise en oeuvre du bus I2C

Le bus I2C, brevet de la société Philips, permet d'interconnecter des composants électronique entre eux.

I2C signifie : **I**nter **I**ntegrated **C**ircuit

1 Description de la configuration utilisée

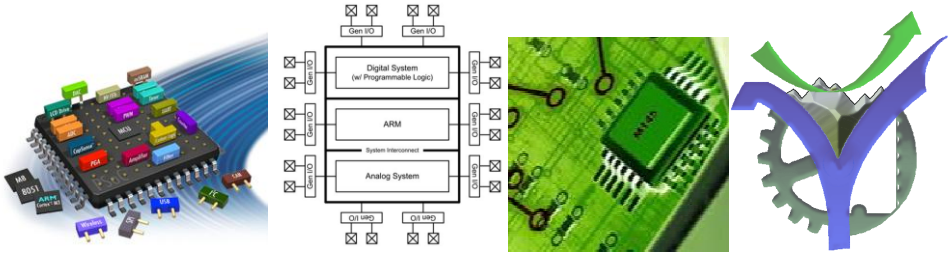


Les adresses sont définies conformément aux spécifications de la normalisation sur 7 bits, par exemple l'adresse du capteur de température 0x4F donne en binaire 1001111 ⇨ A2=A1=A0=1 dans notre cas.

1	0	0	1	A2	A1	A0
MSB						LSB

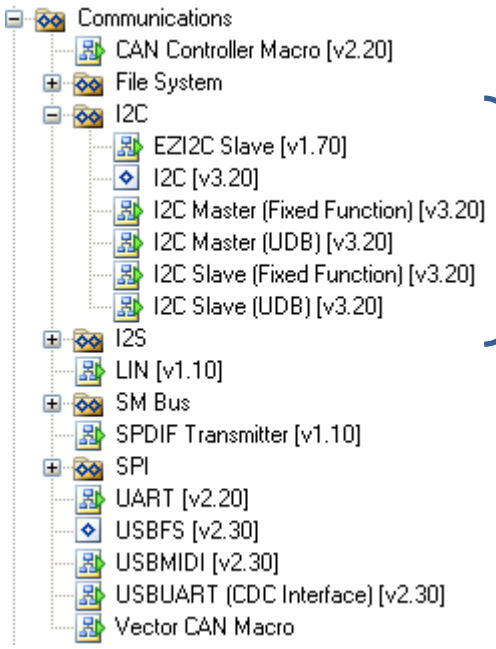
Adressage du LM75

Formation PSoC



2 Le composant I2C dans PSoC Creator

Les différentes variantes du composant I2C sont données ci-dessous :



Toutes les options disponibles

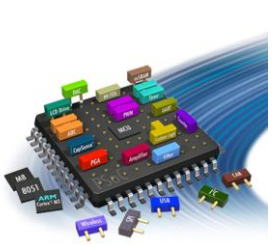
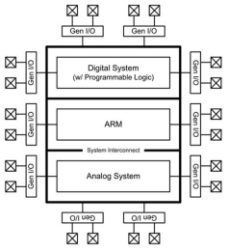
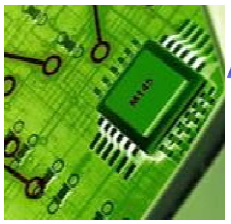
Universal **D**igital **B**locks

Le composant I2C peut fonctionner en esclave seul, maitre seul, multi-maître, multi-maître ou esclave, de quoi faire pâlir de jalousie les plateformes de développement à base de microcontrôleur. Il y a ensuite deux catégories d'implémentation du bus soit l'utilisation du block Fixe Function soit l'implémentation dans les blocs logiques programmables UDB. Pour plus de précision se référer à la documentation du PSoC et à la documentation du composant I2C disponible dans PSoC Creator.

L'adressage du bus est prévue sur 7 bits, l'extension à 10 bits doit être géré par le logiciel. La fréquence d'utilisation maximum peut aller jusqu'à 1 MHz.

Le datasheet du composant I2C donne toutes les précisions nécessaires, (52 pages pour la version 3.20 !)

Formation PSoC

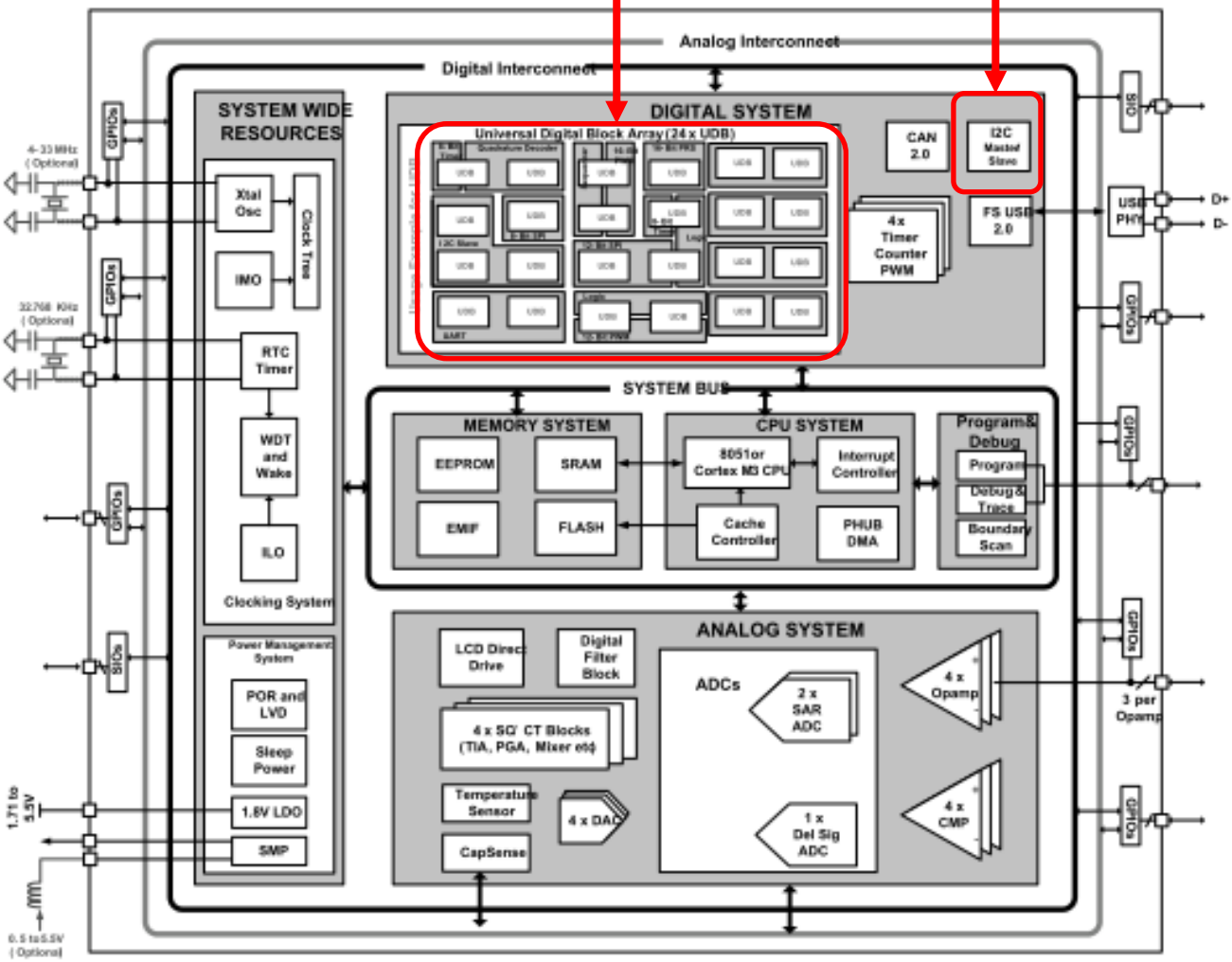




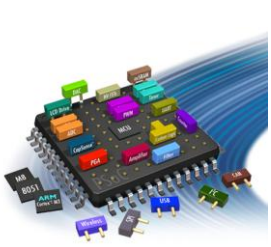
Implémentation possible du bus I2C

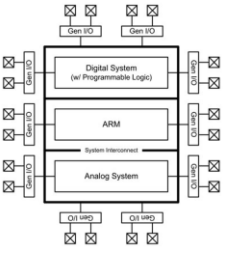
Implémentation dans les blocs UDB

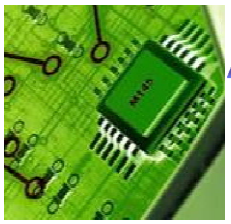
Utilisation de la fonction I2C intégrée

Figure 1-1. Simplified Block Diagram



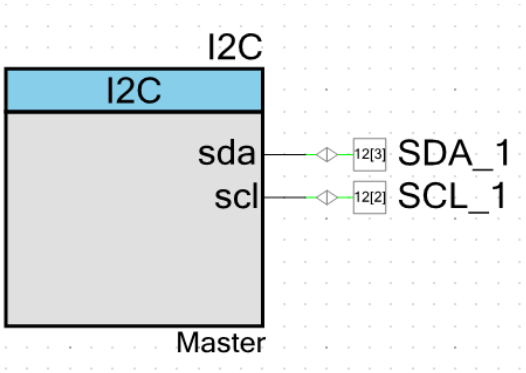






3 Implémentation du composant I2C dans notre projet

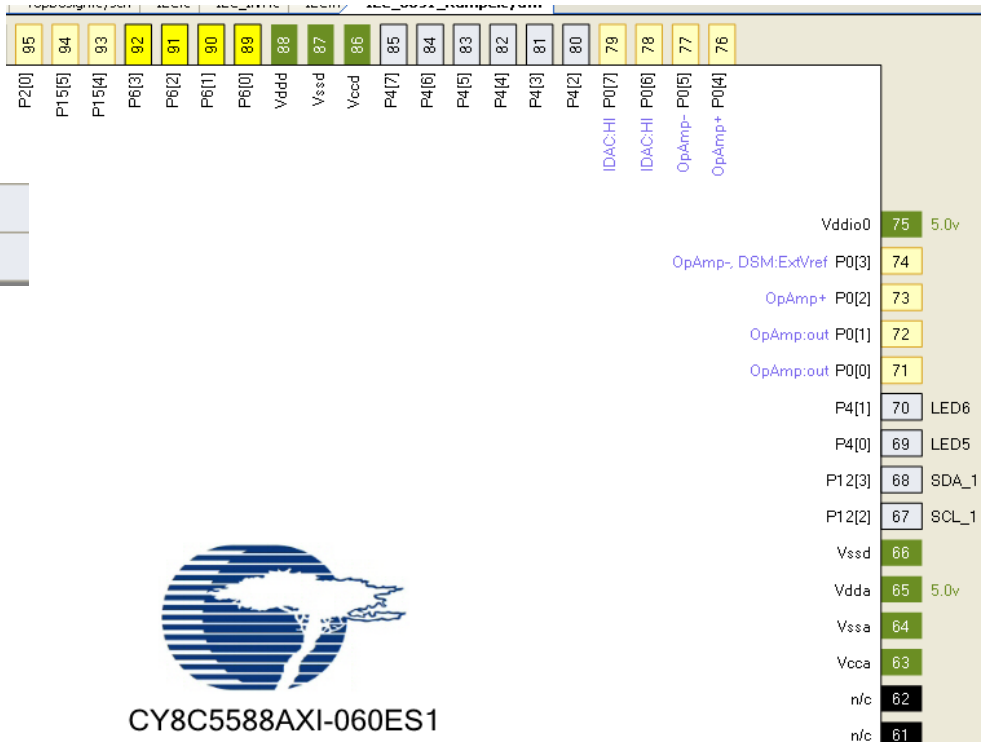
Le composant I2C Master Fixed Function est implanté dans le projet par une simple action de glisser déposer. A noter que le nom du composant ici I2C servira de racine à toutes les fonctions de codes utilisant celui-ci. Cette façon de procéder est obligatoire car il est possible d'implémenter plusieurs composant I2C dans un projet.



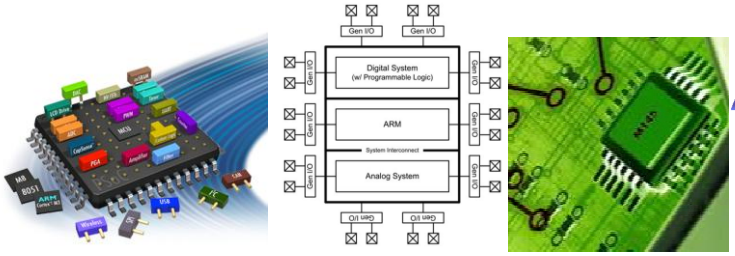
Le placement des broches sda et scl dépend de la cible utilisée, pour le ProtoVox nous utilisons les broches P12[3] et P12[2]. Voir ci-dessous les informations issues de l'onglet de configuration des broches <NomDuProjet.cydwr>.

Il n'y a pas de modification à effectuer sur la configuration des pins de sorties car elles sont configurées avec l'implantation du composant dans le projet.

SCL_1	P12[2]
SDA_1	P12[3]



CY8C5588AXI-060ES1



4 Accès du bus en écriture

Avant d'utiliser les fonctions logicielles du bus il faut initialiser la gestion des interruptions puis le bus par les commandes :

```
65      // Initialisation des interruptions
66      CyGlobalIntEnable;

77      // Initialisation du composant I2C
78      I2C_Start();
```

Les déclarations particulières aux adresses du bus :

```
18  #define PCF8574 0x20
19  #define LM75 0x4F
20
21  #define PCF8591 0x48
22  #define CONF_PCF8591 0b01000000
```

La procédure d'écriture sur le bus :

```
45  void WriteI2C(uint8 Adresse, uint8 Donnee)
46  {
47      uint8 wbuffer[2];
48      wbuffer[0]=Donnee;
49      I2C_MasterClearStatus();
50      status=I2C_MasterWriteBuf(Adresse,wbuffer,1,I2C_MODE_COMPLETE_XFER);
51      while(Ou == (I2C_MasterStatus() & I2C_MSTAT_WR_CMPLT)){ }
52  }
```

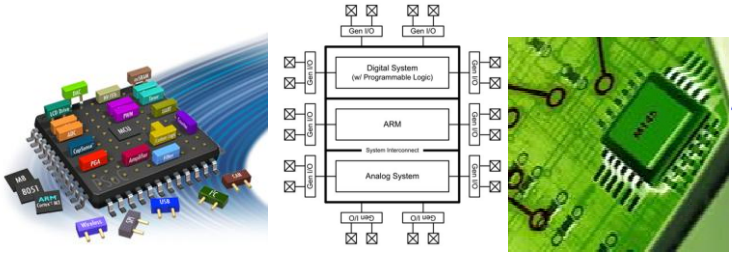
Commentaire :

Ligne 49 : lecture et mise à zéro du master status

uint8 I2C_MasterClearStatus(void)

Description:	This function clears all status flags and returns the master status.
Parameters:	None
Return Value:	uint8: Current status of the master. See the I2C_MasterStatus() function for constants.
Side Effects:	None

Le status du master est résumé par 8 flags récapitulés dans le tableau ci-dessous :



Master status constants	Description	Valeur masque
I2C_MSTAT_RD_CMPLT	Read transfer complete. The error condition bits must be checked to ensure that the read transfer was successful.	0x01u
I2C_MSTAT_WR_CMPLT	Write transfer complete. The error condition bits must be checked to ensure that the write transfer was successful.	0x02u
I2C_MSTAT_XFER_INP	Transfer in progress	0x04u
I2C_MSTAT_XFER_HALT	Transfer has been halted. The I^2C bus is waiting for the master to generate a Restart or Stop condition.	0x08u
I2C_MSTAT_ERR_SHORT_XFER	Error condition: Write transfer completed before all bytes were transferred.	0x10u
I2C_MSTAT_ERR_ADDR_NAK	Error condition: The slave did not acknowledge the address.	0x20u
I2C_MSTAT_ERR_ARB_LOST	Error condition: The master lost arbitration during communication with the slave.	0x40u
I2C_MSTAT_ERR_XFER	Error condition: This is the ORed value of error conditions provided in this table. If all error condition bits are cleared, but this bit is set, the transfer was aborted because of slave operation.	0x80u

Le huitième flag est intéressant car il globalise tous les bits d'erreurs.

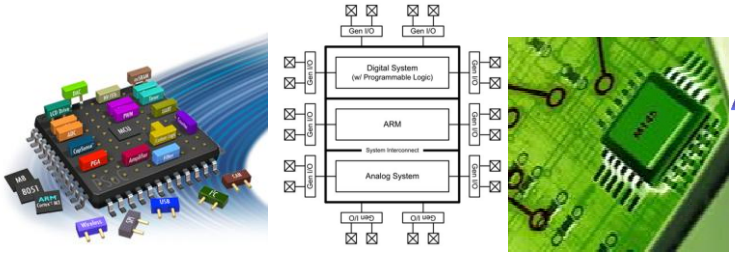
Ligne 50 51: écriture de la valeur sur le bus

```
status=I2C_MasterWriteBuf(Adresse,wbuffer,1,I2C_MODE_COMPLETE_XFER);
while(0u == (I2C_MasterStatus() & I2C_MSTAT_WR_CMPLT)){ }
```

L'écriture du tampon sur le bus avec comme mode :

Mode Constants	Description
I2C_MODE_COMPLETE_XFER	Perform complete transfer from Start to Stop.
I2C_MODE_REPEAT_START	Send Repeat Start instead of Start.
I2C_MODE_NO_STOP	Execute transfer without a Stop

Le while de la ligne 51 attend que l'écriture soit terminée pour poursuivre, quand l'écriture est terminée le flag correspondant passe à 1, fin de l'attente.



Pour écrire plusieurs valeurs à la suite il suffit de travailler avec un buffer plus grand par exemple dans l'envoi de la valeur du DAC du PCF8591 il faut envoyer l'octet de configuration suivi de la valeur à convertir soit :

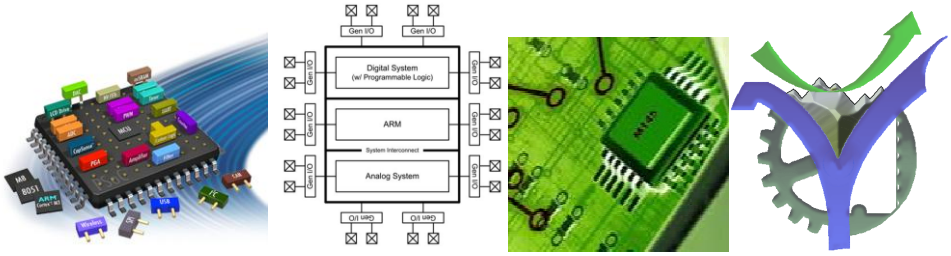
```
28  uint8 status, wbuffer[2],rbuffer[2];

 99      // Initialisation du PCF8591
100      wbuffer[0]=CONF_PCF8591;
101      wbuffer[1]=val;
102      I2C_MasterClearStatus();
103      status=I2C_MasterWriteBuf(PCF8591,wbuffer,2,I2C_MODE_COMPLETE_XFER);
104      while(Ou == (I2C_MasterStatus() & I2C_MSTAT_WR_CMPLT)){ }
105
```

5 Accès du bus en lecture

Les fonctions de lecture du bus sont similaires aux fonctions d'écriture que nous avons vues au paragraphe précédent. Voici la lecture du LM75 le résultat retourne sur 2 octets :

```
205          // Lecture de la température sur le capteur LM75 I2C Read mode complet
206          I2C_MasterReadBuf(LM75, rbuffer,2, I2C_MODE_COMPLETE_XFER );
207          // wait until Transfer is complete
208          while((I2C_MasterStatus() & I2C_MSTAT_RD_CMPLT )==0);
```

Le traitement de la donnée est réalisé avec des instructions c 'classiques'. Voir à ce sujet le listing complet en annexe.



Les temporisations avec PSoC Creator

Les temporisations sont utilisées pour générer de courts délais dans la mise en œuvre de protocole de communication, accès à des périphériques. Ces temporisations sont calculées par la construction du code exécutable, le build, du projet en tenant compte de la fréquence déclarée à PSoC Creator.

La modification dynamique de cette fréquence modifie donc les valeurs réelles des temporisations, dans ce cas il faut reconfigurer dynamiquement les temporisations avec void CyDelayFreq(uint32 freq). Les délais sont correctement établis quand le cache instruction est validé pour le PSoC 5, sinon les valeurs des temporisations sont dibisées par 2.

Les délais générés sont des délais software, ils sont donc sensibles à la présence d'interruption survenant pendant le calcul de ceux-ci, de même des durées incompressibles d'appel et de retour de sous-programme interviennent particulièrement pour les valeurs faibles de délai demandés.

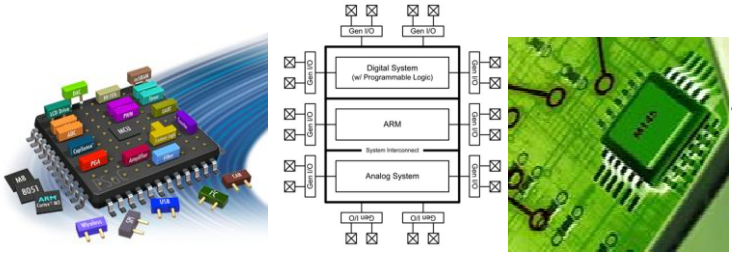
Pour plus de détail consulter : PSoC_Creator_system_reference_guide

En guise de bilan il y a trois temporisations disponibles dans PSoC Creator à savoir :

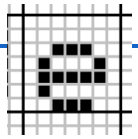
- void CyDelay(uint32 milliseconds)
- void CyDelayUs(uint16 microseconds)
- void CyDelayCycles(uint32 cycles)

Exemple d'utilisation de la temporisation en ms pour faire clignoter les leds reliées aux PCF8574 :

```
152         // Clignotement du PCF8574 si présent
153         if ( (BP4_Read() == APPUYE) && PCF8574_Present )
154         {
155             for (i=0;i<20;i++)
156             {
157                 WriteI2C(PCF8574,~0xAA);
158                 CyDelay(50);
159                 WriteI2C(PCF8574,~0x55);
160                 CyDelay(50);
161                 }
162             WriteI2C(PCF8574,~0x00);
163         }
```

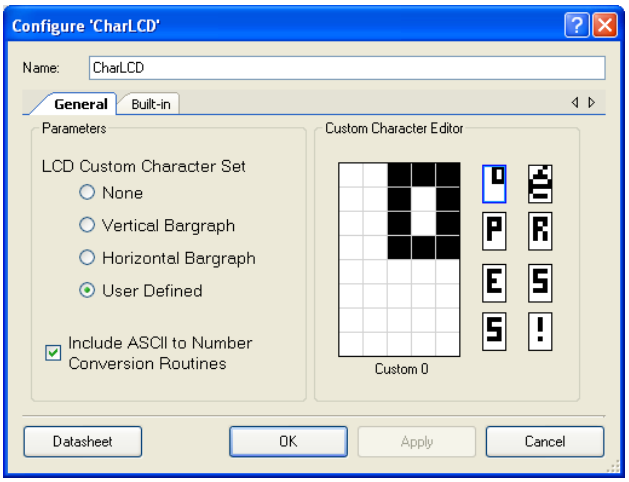


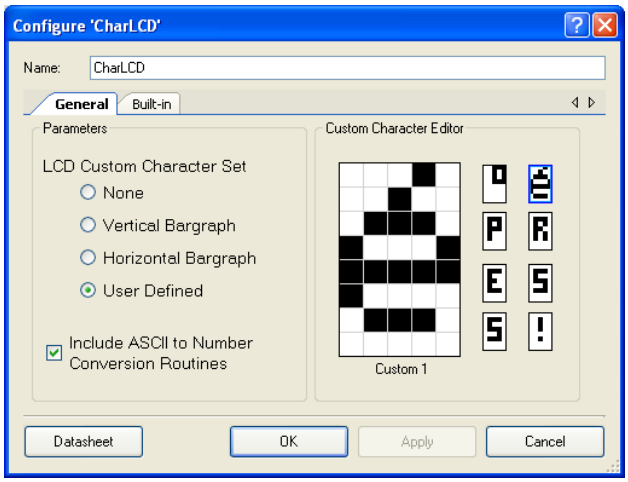
PSoC Creator caractères LCD customisés



La customisation des caractères spéciaux pour l'affichage sur le LCD est très simple.

Il suffit de cliquer sur le composant LCD du schéma et de réaliser son nouveau caractère directement graphiquement pixel par pixel. Dans notre projet nous avons deux caractères spéciaux à savoir la température ° et la lettre minuscule é voilà ce que donne les deux descriptions graphiques :



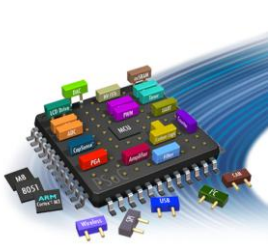
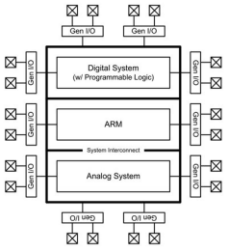


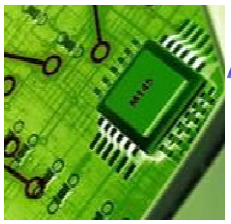
L'appel de ce caractère est fait avec l'instruction putchar par exemple pour afficher le mot 'présent' :

```
125     // Affichage du message d'accueil
126     CharLCD_Position(0,0);
127     if (PCF8574_Present) CharLCD_PrintString("8574 => Appui BP4"); else CharLCD_PrintString("8574 => NACK");
128     CharLCD_Position(1,0);
129     if (PCF8591_Present) CharLCD_PrintString("8591 => Stop => BP2"); else CharLCD_PrintString("8591 => NACK");
130     CharLCD_Position(2,0);
131     if (LM75_Present) {
132         CharLCD_PrintString("LM75 => Pr");
133         CharLCD_PutChar(CharLCD_CUSTOM_1);
134         CharLCD_PrintString("sent");
135         }
136         else CharLCD_PrintString("LM75 => NACK");
137     CharLCD_Position(3,0);
138     CharLCD_PrintString("Start => Appui BP1");
```

Affichage du caractère customisé numéro 1 :

```
CharLCD_PutChar(CharLCD_CUSTOM_1);
```

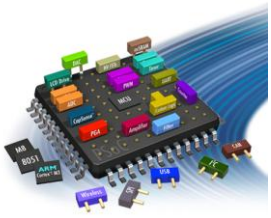



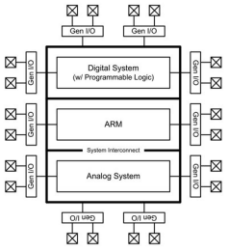


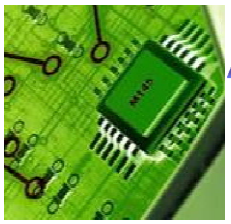
6 Source du programme

Voir aussi le projet complet I2C_8591_Rampe_v2.zip

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * Mise en oeuvre du BUS I2C version 2
 * Recherche de la robustesse du fonctionnement et de la détection des esclaves présents
 * Enumération au début par détection de tous les composants I2C connectés
 *
 * ========================================
*/
#include <device.h>

#define PCF8574 0x20
#define LM75 0x4F

#define PCF8591 0x48
#define CONF_PCF8591 0b01000000


#define APPUYE  0
#define REPOS   1

uint8 status, wbuffer[2],rbuffer[2];

typedef enum  { False=0, True=1} Booleen;

Booleen PCF8574_Present=False;
Booleen PCF8591_Present=False;
Booleen LM75_Present=False;

uint16 Temp=0;

 union composite {
        uint16 mot;
        struct {
                char lo;
                char hi;
        } octet;
};
union composite Temp2;


void WriteI2C(uint8 Adresse, uint8 Donnee)
{
        uint8 wbuffer[2];
        wbuffer[0]=Donnee;
```

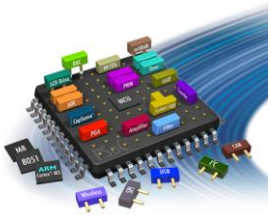
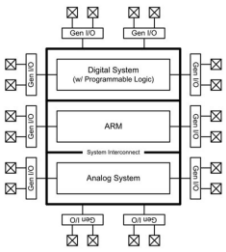


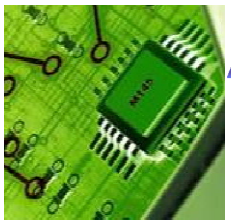




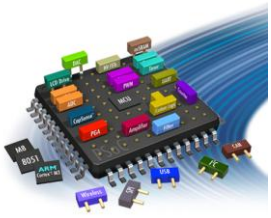
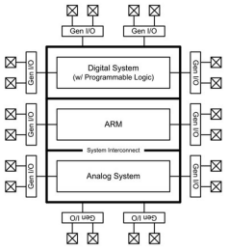
```
        I2C_MasterClearStatus();
        status=I2C_MasterWriteBuf(Adresse,wbuffer,1,I2C_MODE_COMPLETE_XFER);
        while(0u == (I2C_MasterStatus() & I2C_MSTAT_WR_CMPLT)){ }
}



void main()
{
        // Déclaration de variables locales
        unsigned char i,flag;
        uint8 val,I2C_Status;
        char tstr[16];
        rbuffer[0]=0;
        rbuffer[1]=0;

        // Initialisation des interruptions
        CyGlobalIntEnable;

        // Initialisation du LCD
        CharLCD_Start();

        // Affichage du message de bienvenu
        CharLCD_Position(0,0);
        CharLCD_PrintString("Hello  World");
        CharLCD_Position(3,0);
        CharLCD_PrintString("Start => Appui BP1");
        while (BP1_Read() == REPOS ) {};
        CharLCD_ClearDisplay();
        while (BP1_Read() == APPUYE ) {};

        // Initialisation du composant I2C
        I2C_Start();


        // Envoi de l'octet FF sur le PCF8574 => extinction des Leds
        WriteI2C(PCF8574,~0x33);


        // lecture du statut de la liaison I2C
        // vérification de la présence du PCF8574
        I2C_Status=I2C_MasterStatus();
        // On test le bit global erreur I2C
        I2C_Status=I2C_Status & I2C_MSTAT_ERR_XFER;
        if (I2C_Status==0) PCF8574_Present=True;

        // Initialisation du PCF8591
        wbuffer[0]=CONF_PCF8591;
        wbuffer[1]=val;
        I2C_MasterClearStatus();
        status=I2C_MasterWriteBuf(PCF8591,wbuffer,2,I2C_MODE_COMPLETE_XFER);
        while(0u == (I2C_MasterStatus() & I2C_MSTAT_WR_CMPLT)){ }

        // lecture du statut de la liaison I2C
```

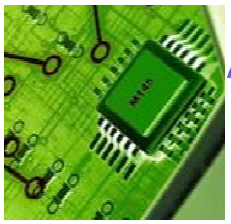
Formation PSoC

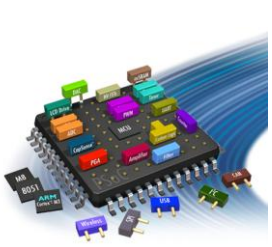
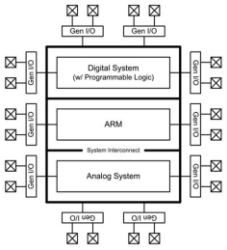


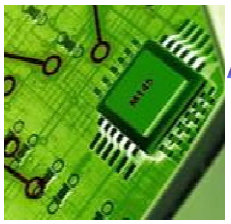
```
        // vérification de la présence du PCF8591
        I2C_Status=I2C_MasterStatus();
        // On test le bit global erreur I2C
        I2C_Status=I2C_Status & I2C_MSTAT_ERR_XFER;
        if (I2C_Status==0) PCF8591_Present=True;


        // 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);
        // lecture du statut de la liaison I2C
        I2C_Status=I2C_MasterStatus();
        // On test le bit global erreur I2C
        I2C_Status=I2C_Status & I2C_MSTAT_ERR_XFER;
        if (I2C_Status==0) LM75_Present=True;


        // Affichage du message d'accueil
        CharLCD_Position(0,0);
        if (PCF8574_Present) CharLCD_PrintString("8574 => Appui BP4");
        else CharLCD_PrintString("8574 => NACK");
        CharLCD_Position(1,0);
        if (PCF8591_Present) CharLCD_PrintString("8591 => Stop => BP2");
        else CharLCD_PrintString("8591 => NACK");
        CharLCD_Position(2,0);
        if (LM75_Present) {
                CharLCD_PrintString("LM75 => Pr");
                CharLCD_PutChar(CharLCD_CUSTOM_1);
                CharLCD_PrintString("sent");
                }
                else CharLCD_PrintString("LM75 => NACK");
        CharLCD_Position(3,0);
        CharLCD_PrintString("Start => Appui BP1");



        while (BP1_Read() == REPOS ) {};
        CharLCD_ClearDisplay();
        CharLCD_Position(0,0);
        CharLCD_PrintString("Bus I2C v2.0");

        val = 0;

        // Boucle permanente
for(;;)
{
                // Clignotement du PCF8574 si présent
                if ( (BP4_Read() == APPUYE) && PCF8574_Present )
                {
                        for (i=0;i<20;i++)
                        {
                                WriteI2C(PCF8574,~0xAA);
                                CyDelay(50);
                                WriteI2C(PCF8574,~0x55);
```

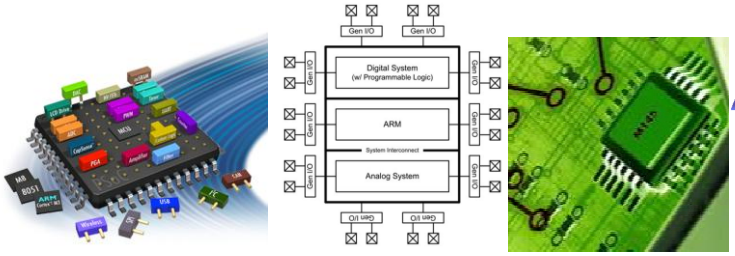


```
                    CyDelay(50);
                    }
            WriteI2C(PCF8574,~0x00);
}

// Gestion du CNA avec relecture sur le CAN et affichage
// Pour bloquer le défilement appuyer sur BP2
if ( PCF8591_Present )
{
            wbuffer[0]=CONF_PCF8591;
            wbuffer[1]=val;
            I2C_MasterClearStatus();
            status=I2C_MasterWriteBuf(PCF8591,wbuffer,2,I2C_MODE_COMPLETE_XFER);
            while(0u == (I2C_MasterStatus() & I2C_MSTAT_WR_CMPLT)){ }

            CharLCD_Position(2,0);
            CharLCD_PrintString("CAN => ");
            sprintf(tstr, "%+5.3f", 5.0/256*wbuffer[1] );
            CharLCD_PrintString(tstr);
            CharLCD_PrintString(" ");
            CharLCD_PrintInt8(val);

            CyDelayUs(20);
            val = val + 10;



            I2C_MasterReadBuf(PCF8591, rbuffer,2, I2C_MODE_COMPLETE_XFER );
            //wait until Transfer is complete

            while((I2C_MasterStatus() & I2C_MSTAT_RD_CMPLT )==0);
            CharLCD_Position(3,0);
            CharLCD_PrintString("CNA => ");
            sprintf(tstr, "%+5.3f", 5.0/256*rbuffer[1] );
            CharLCD_PrintString(tstr);
}

// Arrêt du défilement en appuyant sur BP2
while ( BP2_Read() == APPUYE ) {};



if ( LM75_Present )
{
            // 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
```



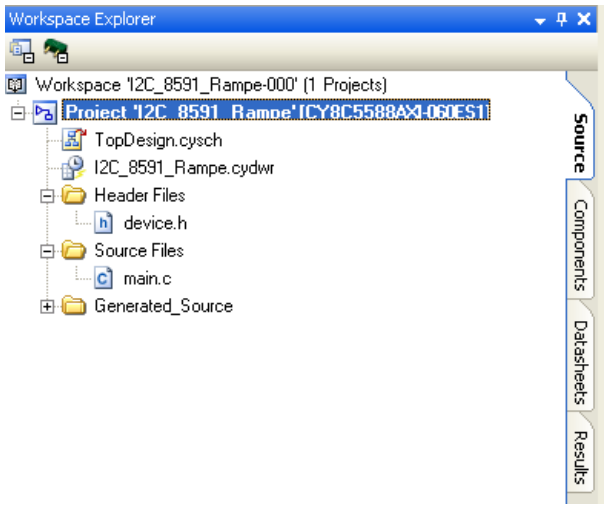
```
                    Temp = rbuffer[0]*256|rbuffer[1];
                    Temp = Temp >> 5;
                    Temp = Temp & 0x0EFF;


                    // 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
                    // .Préparation de la chaine de caractère float => Char
                    // .Affichage sur le LCD
                    // .Affichage du caractère utilisateur '°' défini en Custom_0
                    CharLCD_Position(1,0);
                    CharLCD_PrintInt8(rbuffer[0]);CharLCD_PutChar(' ');
                    CharLCD_PrintInt8(rbuffer[1]);

                    CharLCD_Position(1,8);
                    sprintf(tstr, "%+5.2f", 0.125*Temp );
                    CharLCD_PrintString(tstr);
                    CharLCD_PutChar(CharLCD_CUSTOM_0);
                    CharLCD_PutChar('C');

                }


    }
}

/* [] END OF FILE */
```



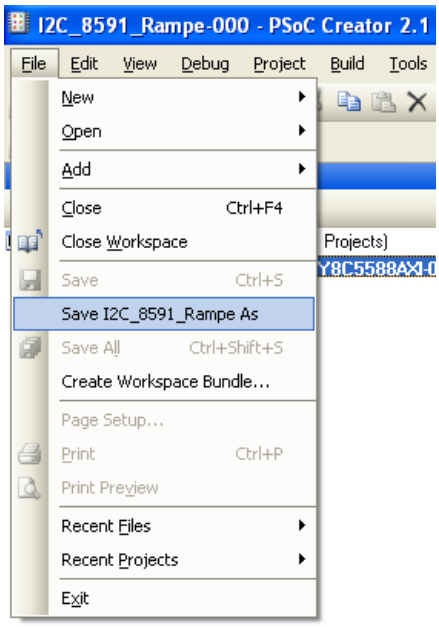
Enregistrer un projet pour archive

Il est fréquent de vouloir enregistrer un projet pour l'archiver. De cette manière on peut faire évoluer un projet par étape en enregistrant les étapes intermédiaires. Ou bien se servir d'un projet comme point de départ pour un autre. Pour enregistrer un projet il faut :

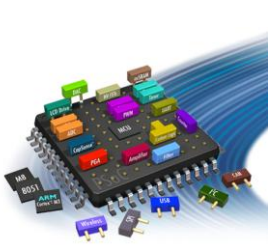
- sélectionner le projet dans le workspace Explorer.

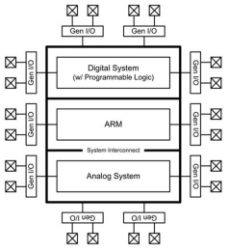


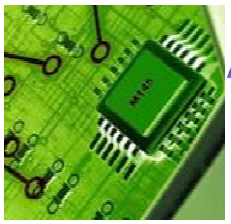
- faire la commande : File \ Save <nom_du_projet> As



- Enregistrer le projet sous le nom choisi. Pour ouvrir le projet avec Psoc Creator il suffit d'aller dans le répertoire de sauvegarde et de cliquer sur le fichier <nom_projet.cyprj>







Indique un document ressource

Retour au sommaire

Retour à la page courante