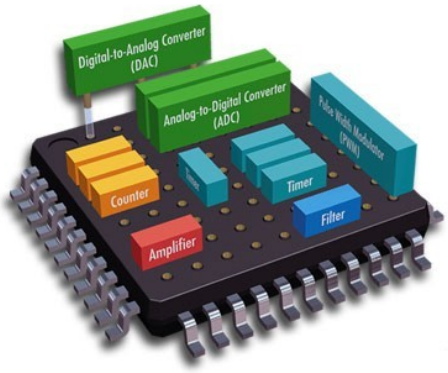
Débuter avec PsoC
ou
PsoC niveau 0

Objectifs : se familiariser avec les circuits PsoC et leur environnement, prendre en main le logiciel et le kit d'évaluation, développer une application simple avec le kit d'évaluation.

Késaco PSoC ?

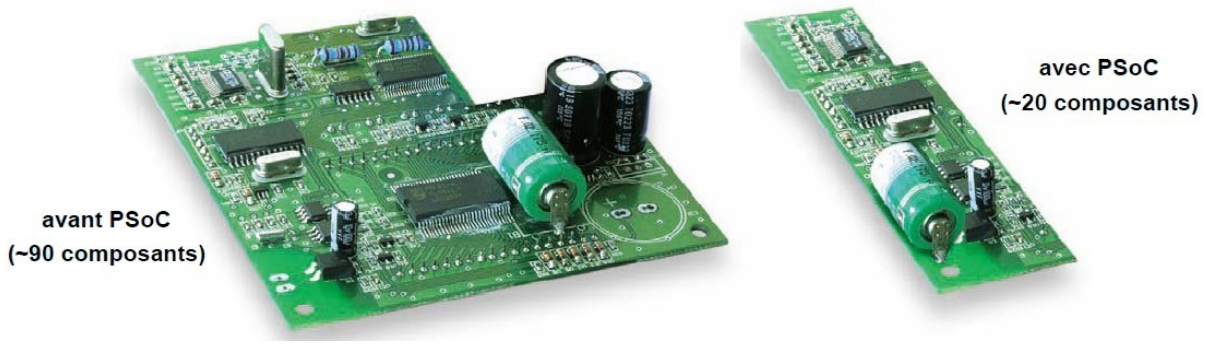
En quelques lignes...

PsoC : **P**rogrammable **S**ystem **o**n **C**hip, est une famille de circuits intégrés introduits au début des années 2000 par Cypress.

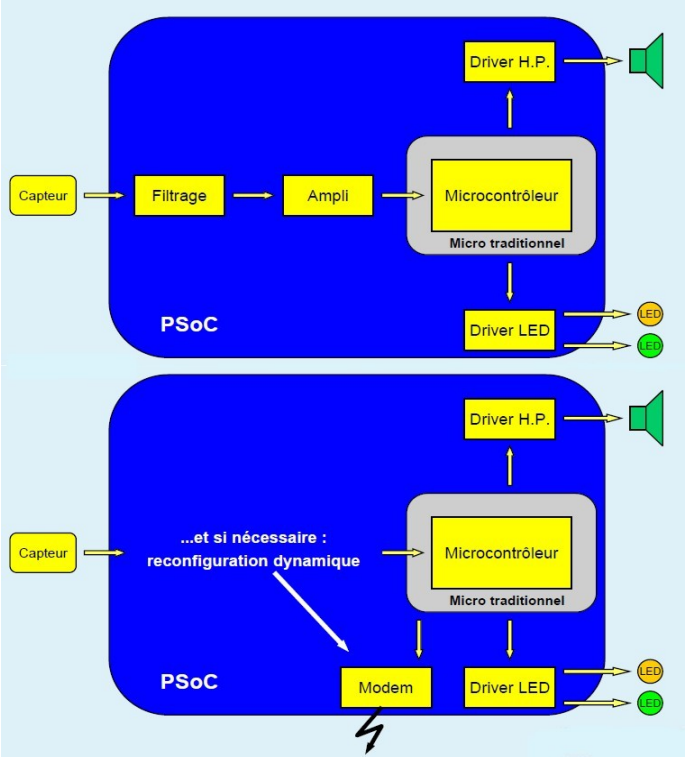


C'est un circuit intégré qui comprend un microcontrôleur et des fonctions logiques et **analogiques** configurables et inter-connectables entre eux.

L'idée est de remplacer le microcontrôleur et les circuits d'interfaces analogiques (convertisseurs AN et NA, filtres, amplificateurs opérationnels, etc..) ou numériques (compteurs, timers, uart, interfaces pour bus divers, etc..) associés par un circuit unique. On intègre ainsi un système électronique embarqué complet dans un circuit intégré unique, ou tout au moins, on réduit très considérablement le nombre de composants !



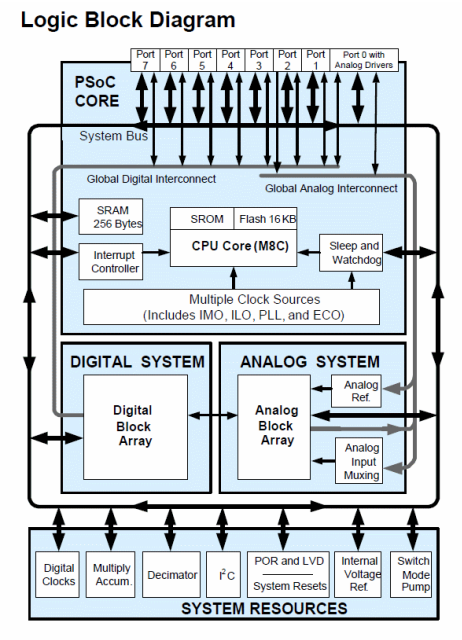
Le PsoC est un circuit reconfigurable en fonctionnement : on peut par exemple imaginer un système embarqué qui s'arrête de fonctionner 1 à 2 minutes par jour pour se reconfigurer en modem et envoyer toutes les données qu'il a acquises dans la journée. Les fonctions utilisées pour la mesure sont reconfigurées en modem.



Quelles applications ?

Elles sont très variées ainsi que le montre ce document Cypress.

Architecture d'un PSoC



Un coeur Microcontrôleur8-bits, avec

- Mémoire Flash
- Oscillateurs internes(pas de quartz externe)
- Multiplieur/accumulateur
- Contrôleur d'interruptions, Sleep Timer, Watchdog Timer
- Contrôleur I2C

Des blocs logiques et analogiques banalisés configurables

- Blocs logiques configurables en compteurs, timers, UARTs, PWMs, SPI, générateurs CRC, séquences pseudo aléatoires, etc.
- Blocs analogiques configurables en amplis-op simples, comparateurs, filtres, CNAs, CANs, modem, etc.

Quelles sont les différences entre PIC et PSOC ?

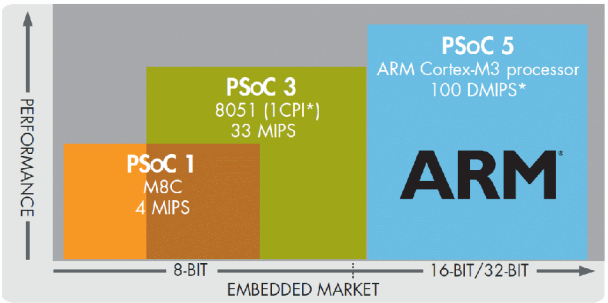
Le PIC ne possède pas de circuits logiques ou analogiques configurables, en conséquence lorsque l'on développe une application, en fonction de ses besoins, on doit choisir son circuit dans toute une série de PICs avec des variantes spécifiques. Avec un PsoC, un seul modèle peut répondre à une multitude d'application.

En bref, le PIC c'est chic, mais le PSOC, c'est choc !

Quels types de PsoC ?

A ce jour (2011), il existe 3 grandes familles de PsoC, nommées PsoC1, PsoC 3 et PsoC 5.
PsoC 1 : C'est la famille d'origine qui date du début des années 2000.
PsoC 3 et PsoC 5 : Deux nouvelles familles de PsoC introduites par Cypress début 2010.

Voici comment Cypress définit ses 3 grandes familles :



The PSoC platform offers unparalleled design flexibility and a wide performance range for your 8-, 16-, or 32-bit designs.

FEATURE	PSoC 1	PSoC 3	PSoC 5
INTERFACE	SPI, UART, GPIO FS-USB, I²C	PSoC 1, plus: CAN, I²S	Same as PSoC 3
INPUTS	Sensors, CapSense, touchscreen, analog	PSoC 1, plus: precision analog	PSoC 3, plus: high-speed analog
OUTPUTS	LED control, motor control, analog buffers	PSoC 1, plus: LCD segment drive, LED drive, advanced motor control	PSoC 3, plus: QVGA LCD control
PROCESSING	M8 24 MHz	8051 67 MHz	ARM® Cortex™-M3 processor 80 MHz

*DMIPS = Dhrystone MIPs.
*CPI = Cycles per instruction.

A ce jour (février 2011), nous n'utilisons à Cabanis que des PsoC 1.

Voici les différents types de PsoC 1ainsi que leurs principales caractéristiques :

PSoC 1 : Récapitulatif des différentes caractéristiques

	CY8C29xxx	CY8C27xxx	CY8C24xxx	CY8C21xxx
Operating Frequency	94KHz-24MHz	94KHz-24MHz	94KHz-24MHz	94KHz-24MHz
Operating voltage	3.0-5.25V	3.0-5.25V	2.4-5.25V	2.4-5.25V
Program Memory (Kbytes)	32	16	4/16	4/8
Data Memory (Bytes)	2K	256	256/1024	256/512
Digital Blocs	24	8	4	4
Analog Blocs	12	12	6	2
I/O pins	24-64	6-44	6-24-48	6-28
Packages	DIP,SSOP,MLF	DIP,SSOP,MLF	DIP,SSOP,MLF	SOIC,SSOP,MLF

Ordre de grandeur de prix pour 1 composant entre 1 et 10 euros.

L'environnement PSoC

Quels outils de développement ?

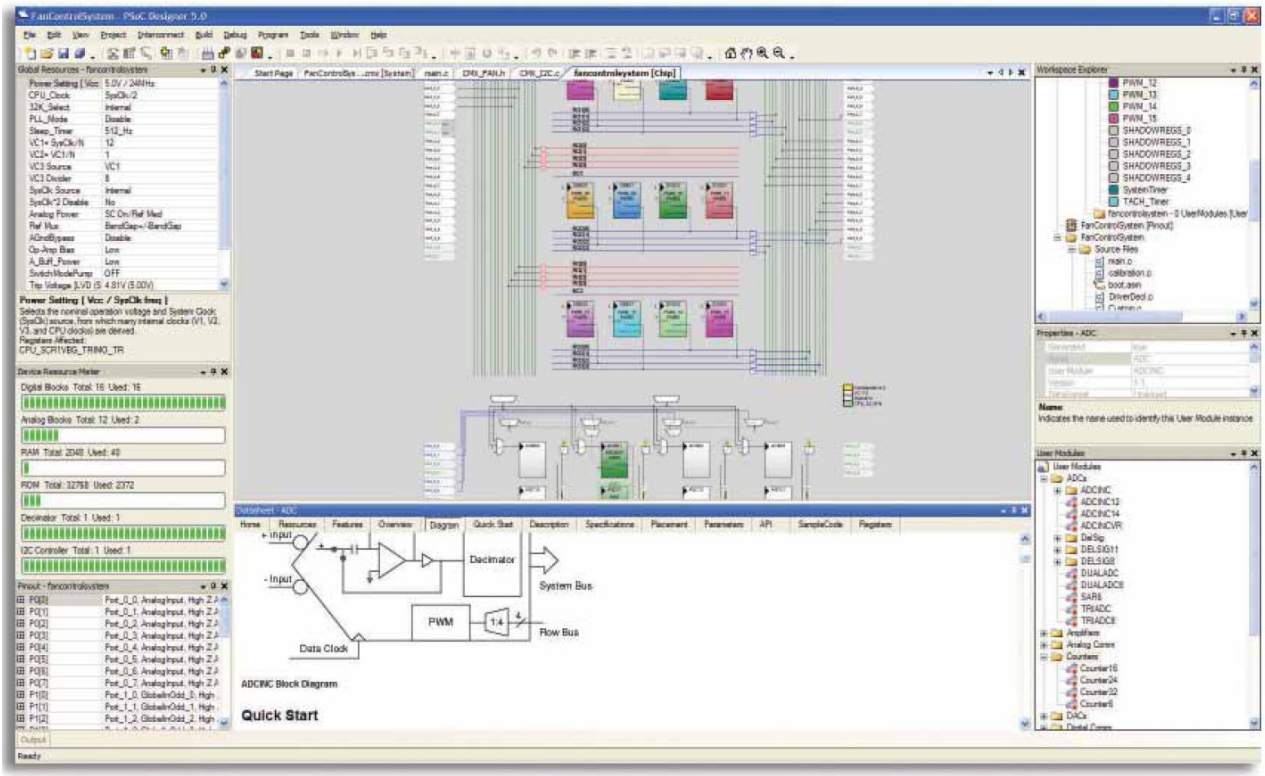
Cypress fournit des outils de développements gratuits téléchargeables gratuitement sur son site (il faut toutefois s'inscrire).

Le développement des PsoC1 s'effectue à l'aide du logiciel **PsoC Designer** (Version 5.1 à ce jour, février 2011, incluant le compilateur C gratuit, ce qui n'était pas toujours le cas précedemment).
Le développement des PsoC 3 et 5 se fait à l'aide du logiciel **PsoC Creator**.

La programmation du circuit peut se faire in-situ à l'aide de PsoC programmer, accessible aussi à partir de PSoC Designer et PsoC Creator. Il est nécessaire de disposer d'un petit accessoire de programmation (périphérique USB) type PsoC MiniProg.

Programmateur MiniProg et câble USB

« Philosophie » de PsoC designer

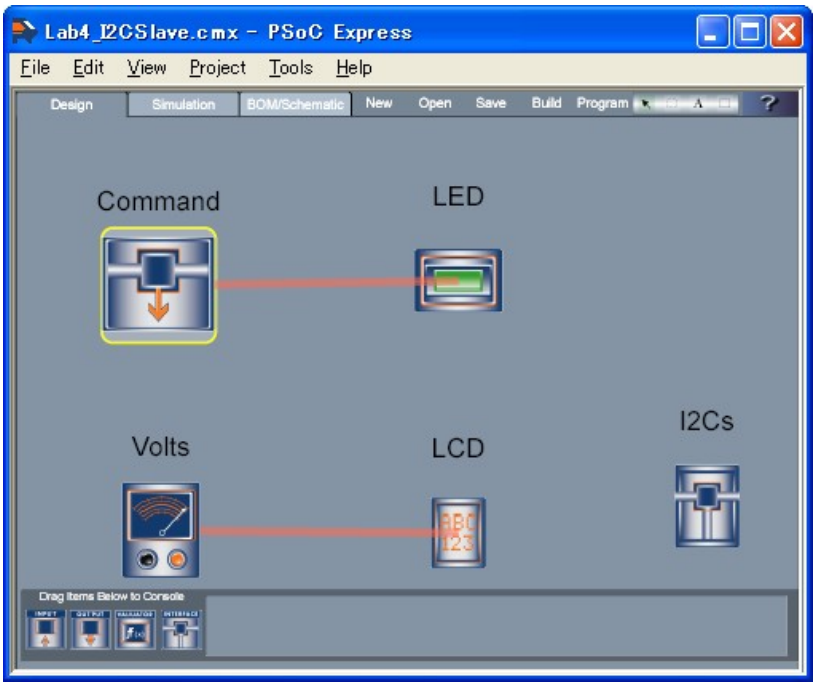


Conception en 4 étapes	1. Sélection et placement des modules	2. Configuration des modules et des drivers d'entrées et sorties	3. Connexion des modules	4. Coder, Compiler, vérifier, débugger
	Choisir les modules analogiques et numériques	Configurer les paramètres globaux (horloge, etc..) ; les parametres des modules et les entrées sorties (type, vitesse, etc..)	Routage des connections avec l'outil graphique.	Programmation en C ou en assembleur, compilation et programmation.

Peut-on programmer graphiquement en PsoC ?

A l'origine, Cypress avait prévu un outil de conception uniquement graphique, qui permettait de programmer un PSoC en quelques minutes sans écrire aucune ligne de C ou d'assembleur : PsoC Express. Cet outil intéressant pour des applications simples a été hélas récemment abandonné par Cypress à partir de la version 5.1 de PsoC designer. Cypress recommande de ne plus utiliser cet outil de développement pour des systèmes en production !

> ***Avertissement de Cypress sur PsoC express :*** *Beginning with PSoC Designer 5.0, service pack 6, we are de-emphasizing System-Level Design (PSoC Express) . With the next major release of PSoC Designer (PSoC Designer 5.1), System Level Design will be completely removed. While no functionality has been removed from this release, we recommend using Chip-Level Design (PSoC Designer style). We will continue to make service pack 6 available for System Level Design users, and it will co-exist with future PSoC Designer releases. However, we are not recommending System Level Design for production designs.*



Logiciel PsoC Express

Carte d'évaluation

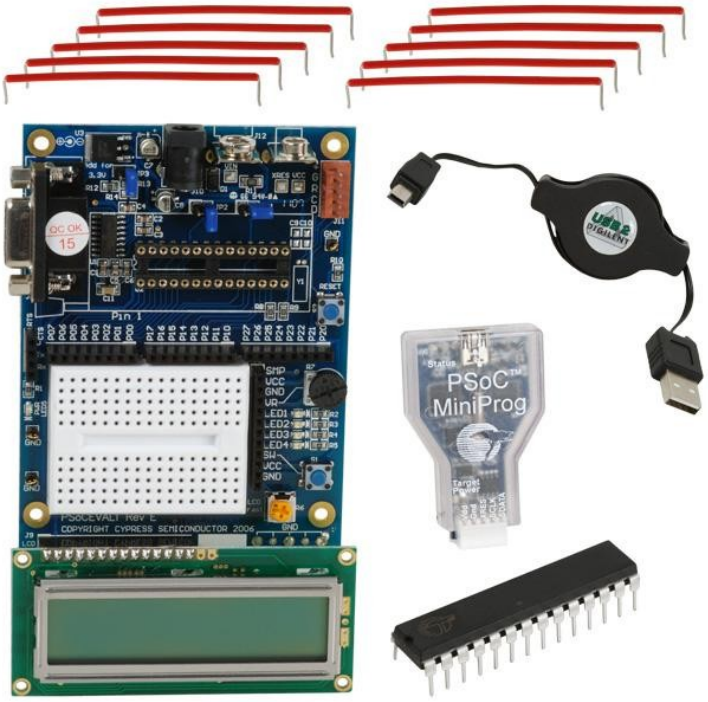
Pour le PSOC 1 il existe un kit d'évaluation : PsoC 1 Evaluation kit : CY3210-PSoCEVAL1.

Ce kit est disponible chez Radiospares ou Farnell pour environ 100 euros.

Le kit comprend :

- une carte d'évaluation incluant un support pour un PsoC DIL 28 broches, 1 afficheur LCD, leds, potentiomètre sur Vcc, circuit d'alimentation, interface RS232, connecteur pour programmation, espace « Labdec » pour prototypage.
- 2 circuit PsoC 1 en DIL 24 (CY8C29466-24PXI et CY8C27445-24PXI)
- Programmateur MiniProg + cable USB
- jeu de fils pour faire des connexions
- CD et notice

Sur la carte d'évaluation, les ports du PsoC sont accessibles sur des connecteurs.



Kit d'évaluation CY3210-PSoCEVAL1

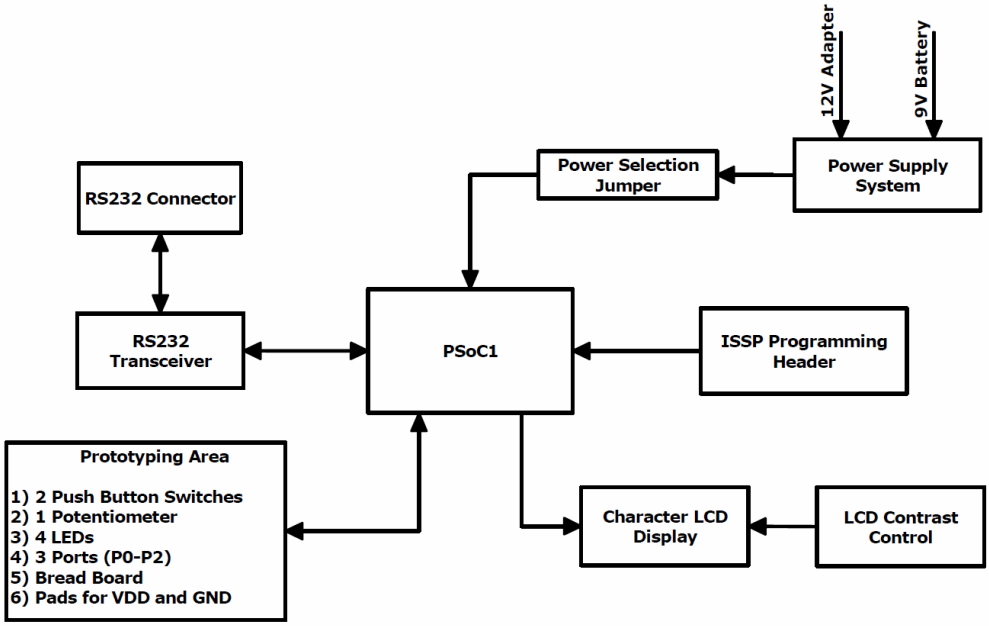


Schéma fonctionnel du kit CY3210-PSoCEVAL1

Conseil : Ne pas utiliser le CD fourni comprenant PsoC Designer et la documentation (fichiers exemples, etc..) : le logiciel et la documentation ne sont pas à jour. Télécharger sur le site de Cypress la dernière version de PsoC designer ainsi que la documentation du kit, incluant les fichiers d'exemples et la notice du kit. Voici un lien pour le kit (actif en février 2011) :

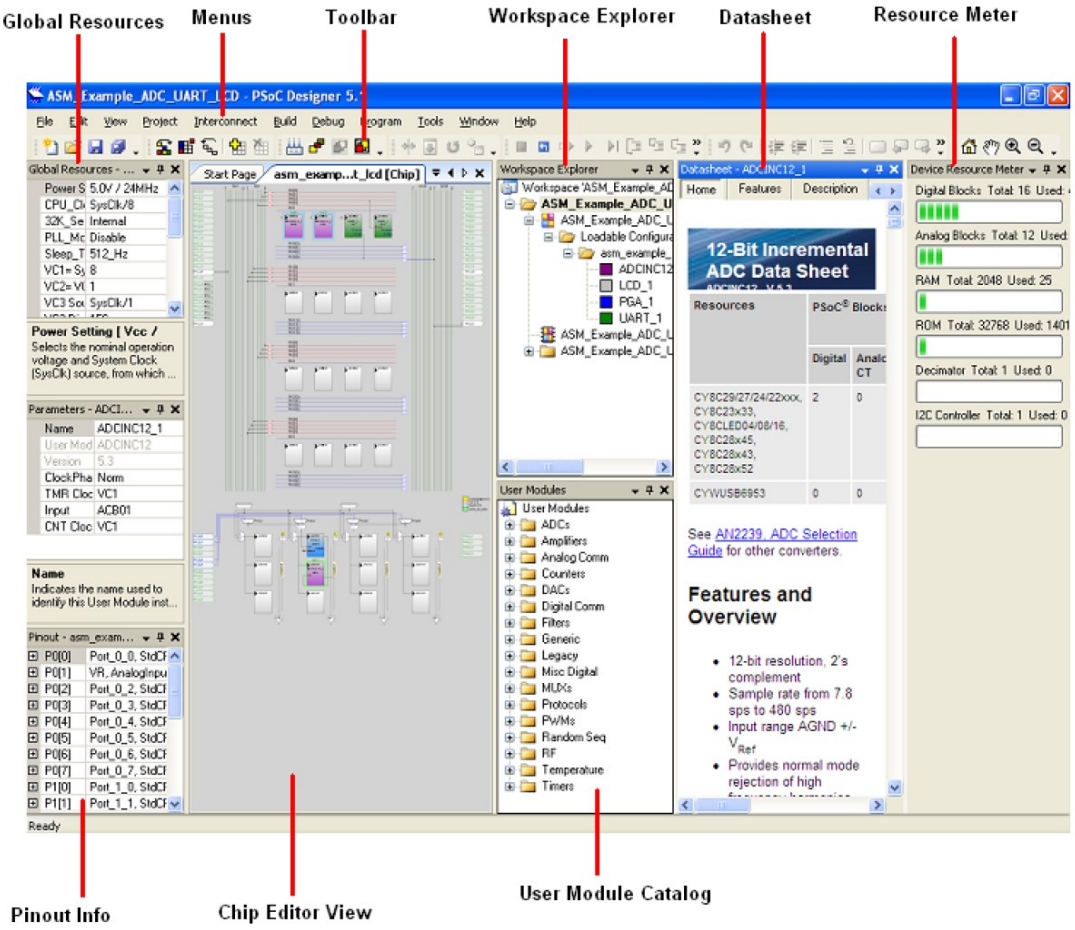
http://www.cypress.com/?rID=2541

Prise en main de PSoC Designer 5.1

Lancement du programme

Après avoir installé PsoC Designer, lancer le programme. PsoC Designer présente une page d'accueil. Pour commencer un projet, cliquer sur **File** (en haut à gauche) puis **New Project**. Choisir un nom et un répertoire de travail, puis (écran suivant) choisir un circuit (**device**), pour le kit d'évaluation, le circuit à utiliser est le **CY8C29466-24PXI**. Choisir un langage de programmation pour le programme main (C ou assembleur).

On accède ensuite à l'écran principal de PsoC Designer qui est divisé en plusieurs fenêtres : il vaut mieux travailler sur un grand écran !

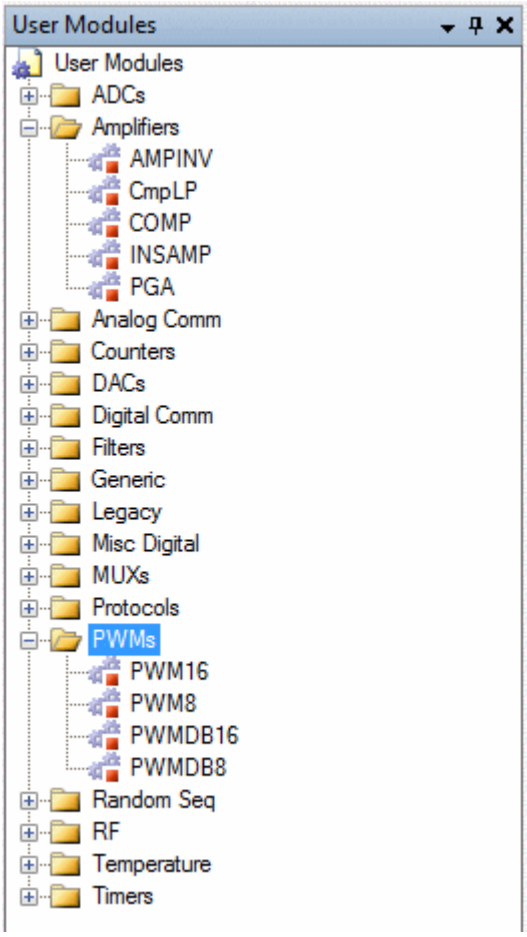


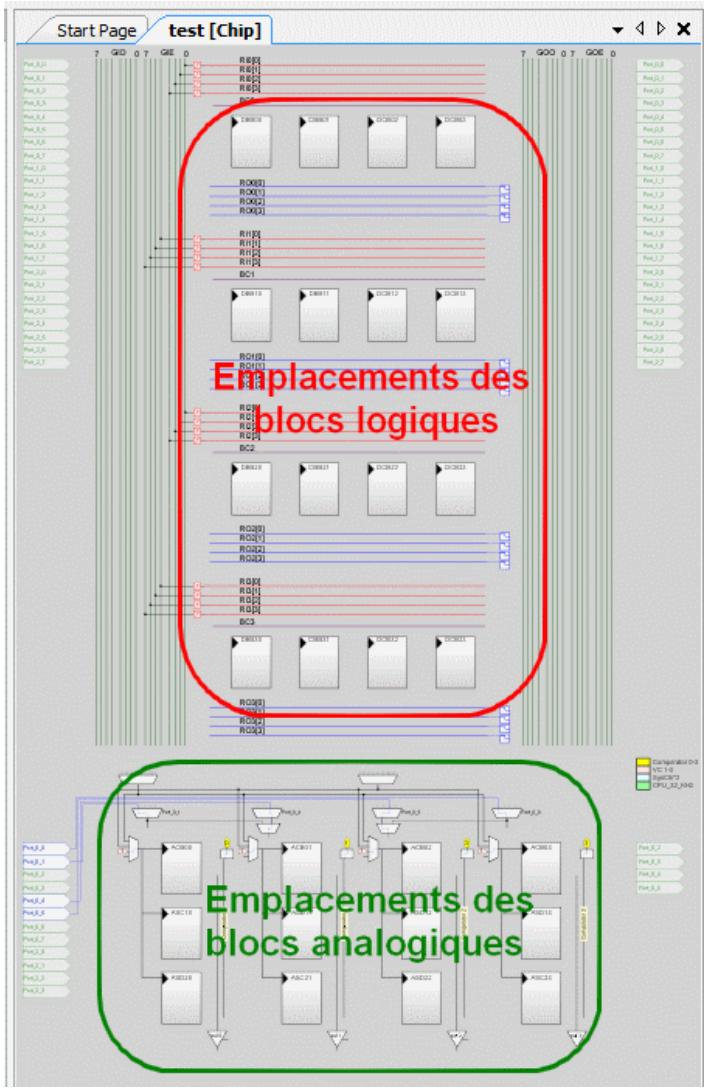
On rappelle que la conception d'après Cypress se fait en 4 étapes.

Conception en 4 étapes	1. Sélection et placement des modules	2. Configuration des modules et des drivers d'entrées et sorties	3. Connexion des modules	4. Coder, Compiler, vérifier, débugger
	Choisir les modules analogiques et numériques	Configurer les paramètres globaux (horloge, etc..) ; les parametres des modules et les entrées sorties (type, vitesse, etc..)	Routage des connections avec l'outil graphique.	Programmation en C ou en assembleur, compilation et programmation.

Remarque : Les 4 étapes décrites ici ne sont ni formelles, ni irréversibles, rien n'empêche par exemple de rajouter un module après avoir écrit des lignes de code !

Etape 1 : Sélection et placement des modules





Les modules (constitués de blocs logiques et analogiques) se trouvent dans la section User Modules. On installe un module en double-cliquant sur celui-ci. Les blocs viennent alors s'installer dans la fenetre [chip]».

Il est possible d'accéder à la documentation de chaque module par un clic droit sur le module puis Datasheet.

Attention : un seul module peut occuper plusieurs blocs, logiques ou analogiques (en fonction de la complexité). Exemple : 1 module PWM 8 bits occupe un bloc logique, un module PWM 16 bits occupe 2 blocs logiques, un module amplificateur inverseur occupe un bloc analogique, un module ADCIN (CAN incremental) occupe un bloc logique et un bloc analogique.

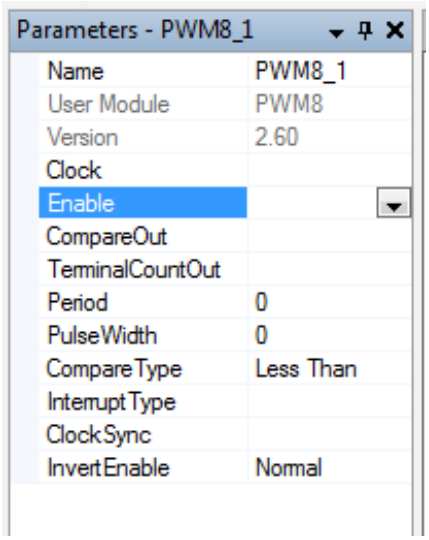
Les blocs analogiques ne peuvent pas être placés n'importe ou : la première ligne de modules analogiques reçoit les blocs correspondant à des modules amplificateur inverseur ou non inverseur tandis que les deux lignes suivantes de modules reçoivent des blocs à capacités commutés (filtres, etc..).

Les modules installés dans la fenêtre [chip] peuvent être déplacés à la souris par glissé-déposé.

Etape 2 : Configuration des modules et des drivers d'entrées et sorties.

Au cours de cette étape, il faut configurer et paramétrer les différents modules, les paramètres globaux et les entrées et sorties du circuit.

Configuration des modules



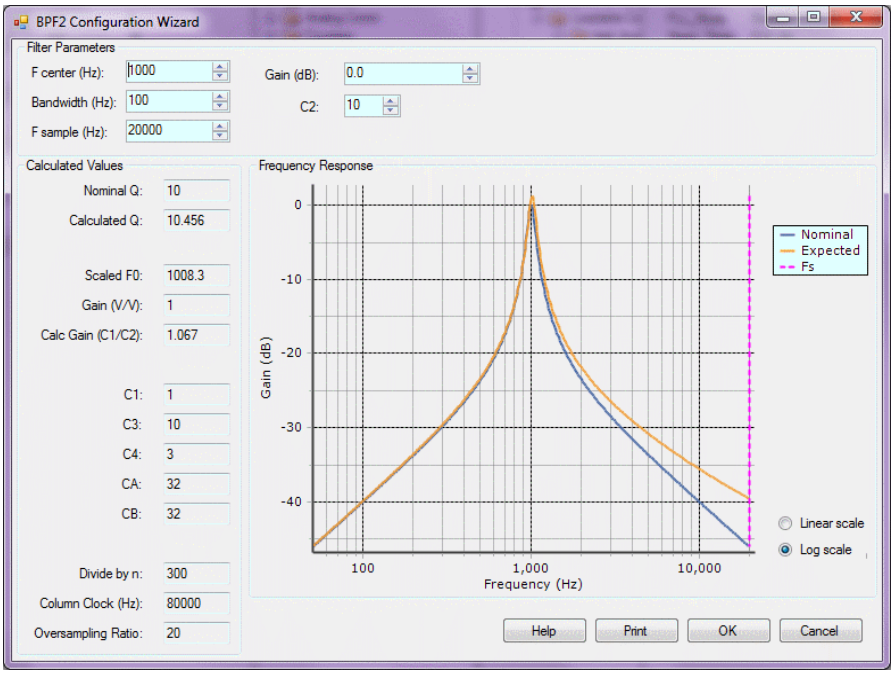
La fenêtre Paramètres (Parameters) permet de configurer chaque module utilisé.

Ici, pour un PWM 8 bits, on peut configurer les entrées horloge et Enable, les sorties, la période, la largeur etc..

Toutes ces données peuvent aussi être modifiées par le programme C. Les instructions détaillées se trouvent dans la « datasheet » du composant accessible par le logiciel.

Attention : pour les modules analogiques la mise à la « masse » d'une entrée peut se faire de 2 facons AGND et VSS. AGND est une masse analogique à un potentiel de VCC/2.

A noter : certains modules disposent de « Wizard » qui permettent de configurer les paramètres des circuits complexes : filtres, comparateurs.



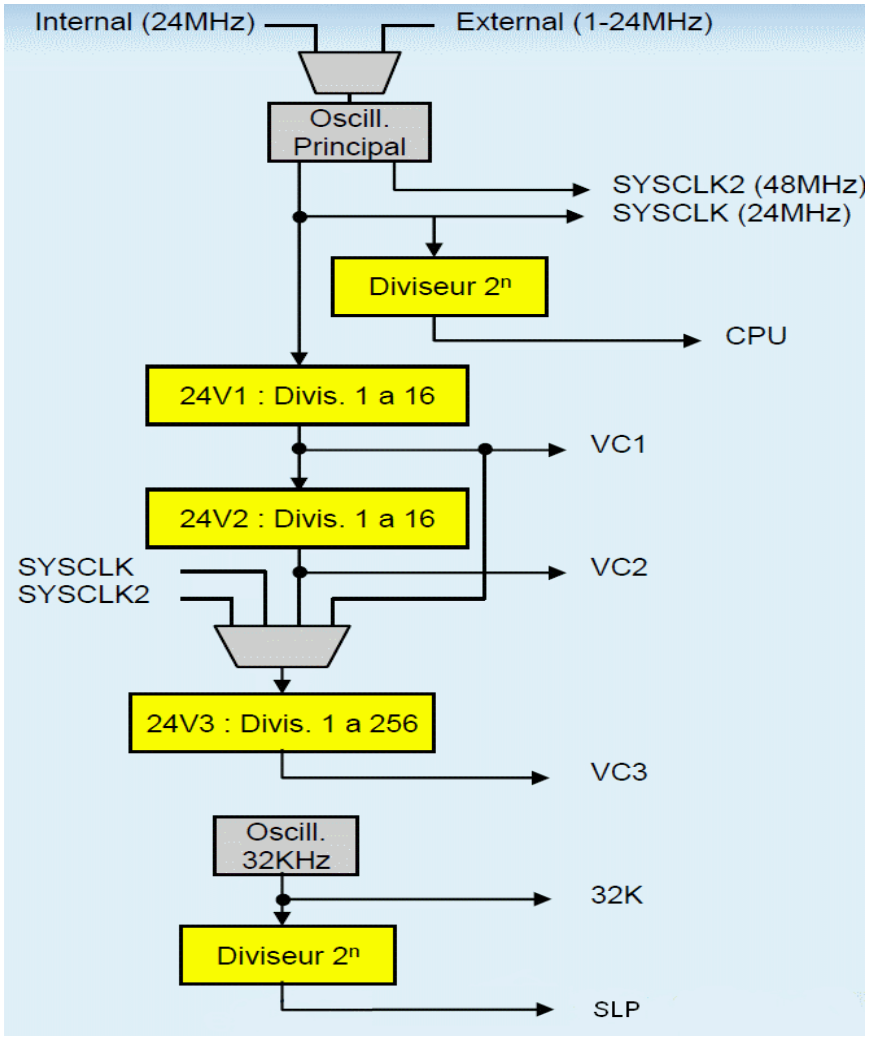
Configuration d'un filtre passe-bande

Configuration des paramètres globaux

Global Resources - test

Power Setting [Vcc / SysClk freq]	5.0V / 24MHz
CPU_Clock	SysClk/8
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	16
VC2= VC1/N	16
VC3 Source	VC2
VC3 Divider	256
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 (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

La fenêtre Ressources globales (Global Ressources) permet de configurer les paramètres du PsoC communs aux différents modules placés, en particulier les circuits d'horloge VC1, VC2, VC3 (VC : Variable Clock), diviseurs programmables qui peuvent attaquer les circuits compteurs ou PWM.



Circuits d'horloges du PSOC

Configurations des entrées et sorties

Pinout - test

P0[0]	Port_0_0, StdCPU, High Z Analog, DisableInt, 0
P0[1]	Port_0_1, StdCPU, High Z Analog, DisableInt, 0
P0[2]	Port_0_2, StdCPU, High Z Analog, DisableInt, 0
P0[3]	Port_0_3, StdCPU, High Z Analog, DisableInt, 0
P0[4]	Port_0_4, StdCPU, High Z Analog, DisableInt, 0
P0[5]	Port_0_5, StdCPU, High Z Analog, DisableInt, 0
P0[6]	Port_0_6, StdCPU, High Z Analog, DisableInt, 0
P0[7]	Port_0_7, StdCPU, High Z Analog, DisableInt, 0
P1[0]	Port_1_0, StdCPU, High Z Analog, DisableInt, 0
P1[1]	Port_1_1, StdCPU, High Z Analog, DisableInt, 0
P1[2]	Port_1_2, StdCPU, High Z Analog, DisableInt, 0
P1[3]	Port_1_3, StdCPU, High Z Analog, DisableInt, 0
P1[4]	Port_1_4, StdCPU, High Z Analog, DisableInt, 0
P1[5]	Port_1_5, StdCPU, High Z Analog, DisableInt, 0
P1[6]	Port_1_6, StdCPU, High Z Analog, DisableInt, 0
P1[7]	Port_1_7, StdCPU, High Z Analog, DisableInt, 0
P2[0]	Port_2_0, StdCPU, High Z Analog, DisableInt, 0
P2[1]	Port_2_1, StdCPU, High Z Analog, DisableInt, 0
P2[2]	Port_2_2, StdCPU, High Z Analog, DisableInt, 0
P2[3]	Port_2_3, StdCPU, High Z Analog, DisableInt, 0
P2[4]	Port_2_4, StdCPU, High Z Analog, DisableInt, 0
P2[5]	Port_2_5, StdCPU, High Z Analog, DisableInt, 0
P2[6]	Port_2_6, StdCPU, High Z Analog, DisableInt, 0
P2[7]	Port_2_7, StdCPU, High Z Analog, DisableInt, 0

La fonction de chacune des broches d'entrée et sortie du PsoC est paramétrable :

- Entrée et/ou Sortie standard (accessible en lecture et/ou en écriture dans l'espace registre)
- Entrée et/ou Sortie globale (connexion aux PSoC Blocks numériques)
- Entrée et/ou Sortie analogique.

L'interface d'entrée ou de sortie est paramétrable :

- Strong (CMOS), Pull-up, Pull-down, High-Z, Open Drain

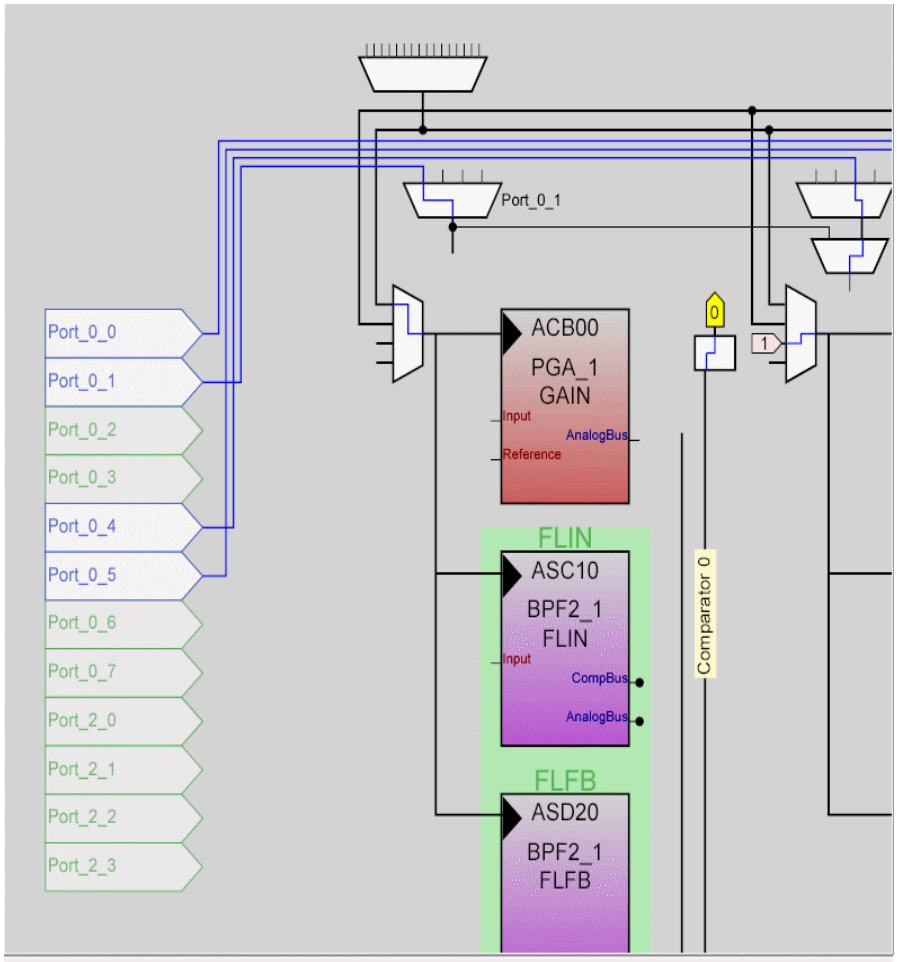
Le Mode Interruption est configurable pour chaque broche :

- Front montant, descendant, ou changement d'état

La configuration des broches d'entrées/sorties se fait soit dans la fenêtre Pinout, soit dans la fenêtre [chip] (dessin du circuit) en cliquant sur chacune des broches.

Etape 3 : Connexion des modules.

L'interconnexion des modules se fait graphiquement dans la fenêtre [chip]. Les connexions se font à l'aide de multiplexeurs que l'on configure en cliquant dessus. Tout se fait graphiquement. Attention n'importe quel module placé ne peut pas être relié à n'importe quelle entrée : d'où l'importance du placement !



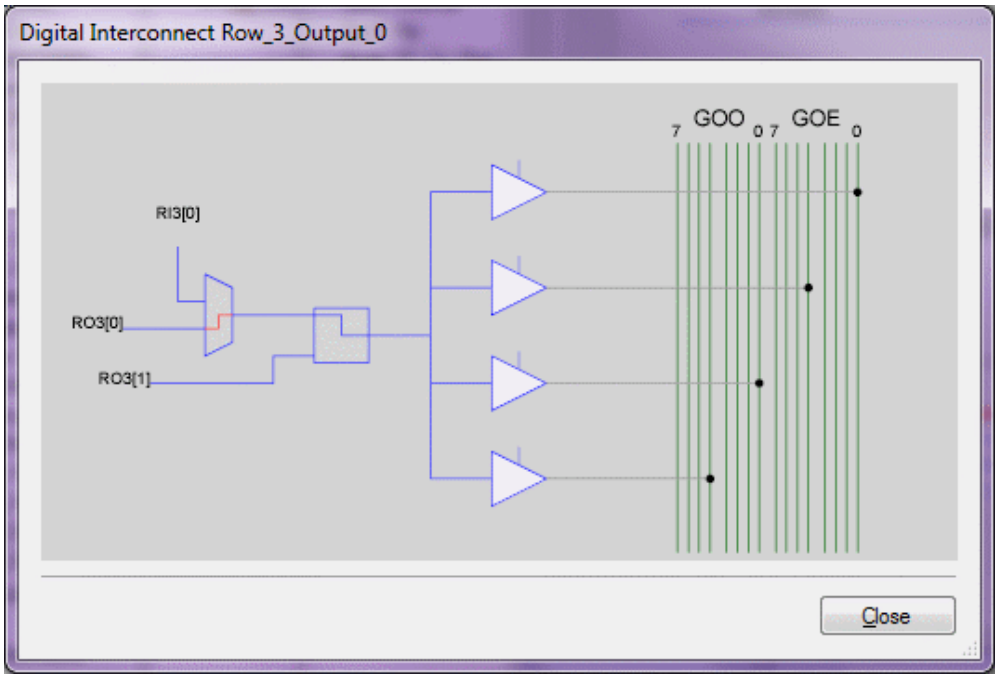
Interconnection des modules analogiques

Pour les modules numériques (voir figure page suivante) on relie les sorties lignes connectées à des colonnes sur lesquelles pourront se connecter les sorties.

Astuces :
Pour les lignes :
Les lignes permettent de déterminer sur quelle **ligne de port** le signal sera connecté.
Les lignes Rxy[0] permettent de se connecter sur une sortie ou une entrée d'indice 0 ou 4.
Les lignes Rxy[1] permettent de se connecter sur une sortie ou une entrée d'indice 1 ou 5.
Les lignes Rxy[2] permettent de se connecter sur une sortie ou une entrée d'indice 2 ou 6.
Les lignes Rxy[3] permettent de se connecter sur une sortie ou une entrée d'indice 3 ou 7.

Pour les colonnes :
Les colonnes permettent de déterminer sur quel **port** le signal sera connecté.
GOO : Global Output Odd : ODD = IMPAIR : C'est ici que l'on pourra connecter une broche appartenant à un port impair : Port 1 ou port 3.
GOE : Global Output Even : EVEN : PAIR : C'est ici que l'on pourra connecter une broche appartenant à un port pair : Port 0 ou port 2. (Remarque : le port 0 est considéré comme pair)

Mon avis :
L'utilisation au début peut surprendre un peu, mais on s'y fait assez vite ! C'est assez intuitif !



Interconnection des modules analogiques

Etape 4 : Coder, compiler, vérifier, débugger.

Il reste maintenant à coder en C. Pour cela, il faut ouvrir le programme principal avec la fonction File puis Open File. Le programme principal est crée par défaut par PsoC designer, il est situé à la racine du projet et s'appelle main.c .
Quand on débute un projet, celui-ci est vide, à l'exception de l'en-tête suivant, et est prêt à être complété !

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

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


void main(void)
{
      // M8C_EnableGInt ; // Uncomment this line to enable Global Interrupts
      // Insert your main routine code here.
}
```

Attention : avant d'utiliser un module placé, il est nécessaire de le mettre en marche par logiciel.
L'instruction à utiliser est donnée dans la datasheet.
Exemple :

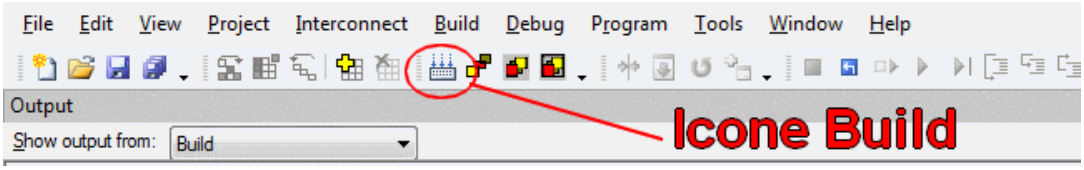
```
      PGA_1_Start(PGA_1_MEDPOWER);
      PGA_2_Start(PGA_1_MEDPOWER);
```

```
LPF2_1_Start(LPF2_1_HIGHPOWER);
BPF4_1_Start(BPF4_1_HIGHPOWER);
COMP_1_Start(COMP_1_MEDPOWER);
```

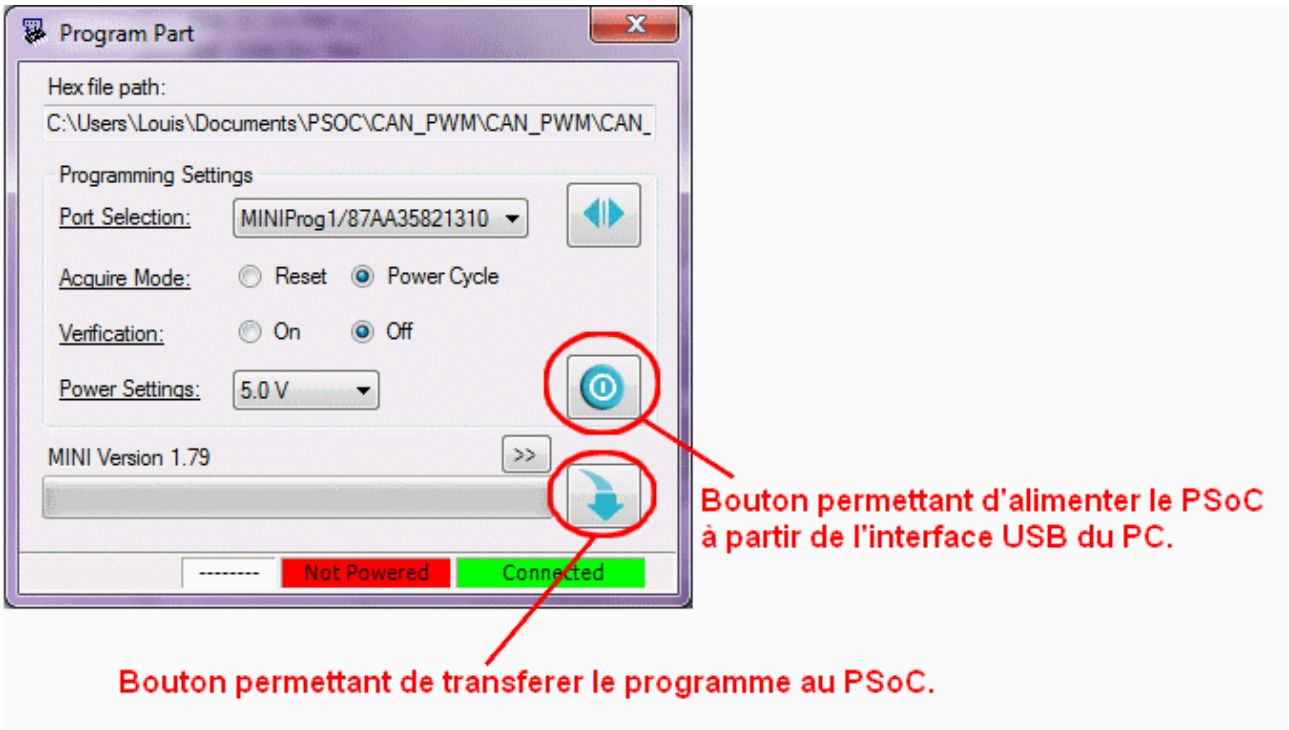
Pour bien utiliser les instructions C relatives à l'utilisation d'un module, se reporter au paragraphe Sample code (échantillon de programme) de la datasheet.

Le reste est de la programmation en C « standard ». Voir quelques astuces dans le paragraphe suivant.

Pour compiler, vérifier et générer les fichiers de configuration du PsoC, il suffit de cliquer sur l'icone Build :



Enfin, il ne reste plus qu'à programmer le PsoC en cliquant sur le menu Program qui ouvre la fenêtre de programmation.



On transfère dans un premier temps le programme dans le PsoC, ensuite, il est possible de faire fonctionner la carte d'évaluation en l'alimentant via l'interface USB du PC.

Guide de survie du programmeur PSOC

Quelques astuces et conseils en vrac pour l'utilisateur de PSOC designer et programmeur. Je mets ici toutes les astuces qui m'ont fait gagner du temps.

Astuces pour la fenêtre [chip]

Déplacement dans la fenêtre [chip] :

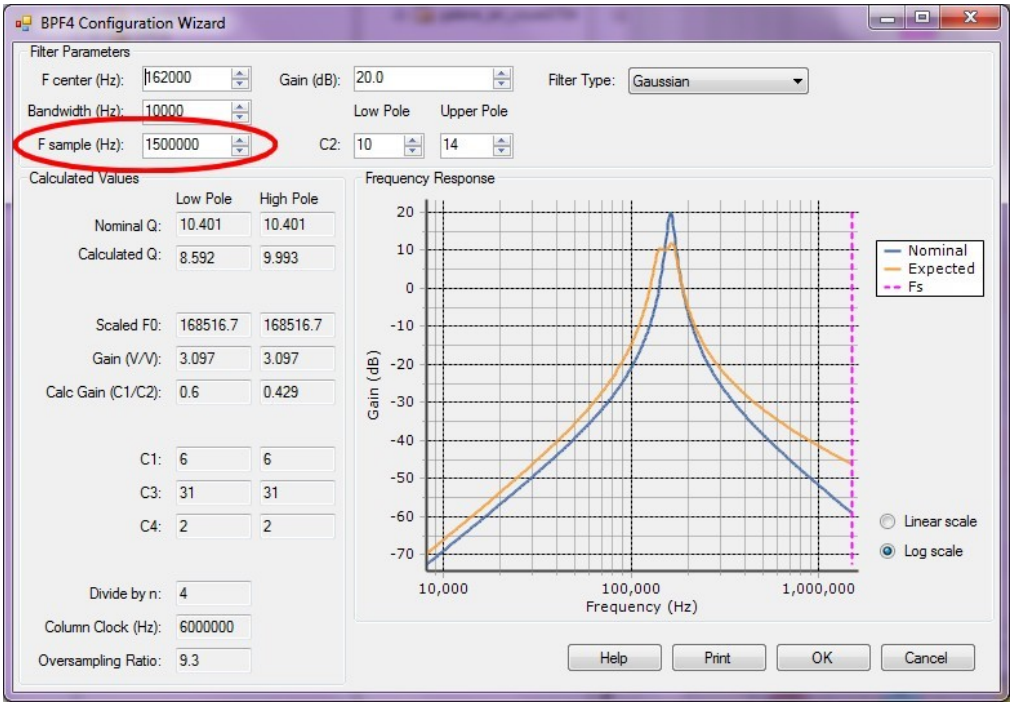
Touche CNTRL + Clic gauche : Zoom grossissant
Touches CNTRL + Shift + Clic gauche : Zoom diminuant
Touche ALT + Clic gauche : permet d'avoir acces à l'outil Main, qui permet de déplacer la feuille.

Masse analogique

Attention, lorsque l'on réalise un amplificateur par exemple, le logiciel propose deux choix pour la « masse » AGND et VSS. VSS correspond au 0V, AGND correspond par défaut à la moitié de la tension d'alimentation. Attention à bien choisir ce paramètre.

Utilisation du Wizard Filtre

Lorsque l'on utilise le Wizard Filtre, le champs fréquence d'échantillonnage est à renseigner.



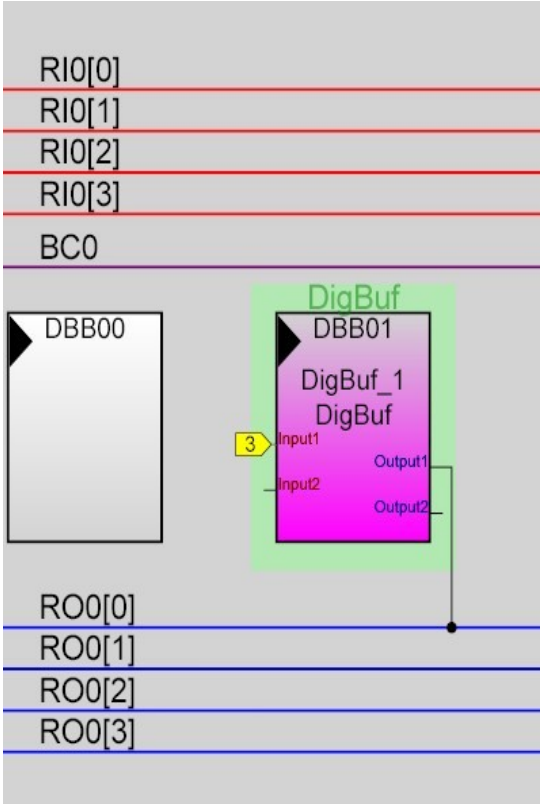
Ce champs ne doit pas être choisi n'importe comment mais est fixé à $f_{CLK}/4$. Où f_{CLK} est l'horloge du filtre utilisé. Celle ci peut être VC1, VC2 ou une horloge externe.

Visualiser la sortie d'un comparateur.

Pas toujours facile de faire sortir sur une broche la sortie d'un comparateur. Confronté à ce

problème, j'ai trouvé une astuce. Je ne sais pas si on peut faire mieux, mais ca fonctionne !

J'utilise un bloc numérique Digital buffer. L'entrée du buffer = la sortie du comparateur, Sortie du buffer = Port du PSoC.



Lire et écrire un port

Pour accéder au contenu d'un port il suffit simplement de lire le registre de 8 bits correspondant au port. Par exemple, pour le lire le contenu du port 2, il suffit simplement de lire le contenu du registre PRT2DR le contenu sera bien sur en hexadécimal. Pour l'écriture sur un port « complet » c'est la même chose !

Lire et écrire un bit d'un port

Pour lire ou écrire à une unique broche d'un des ports, il est nécessaire d'utiliser des masques, comme au bon vieux temps de l'assembleur !

Quelques exemples :

Mise à « 1 » d'un bit d'un port :

Instruction C permettant de mettre à 1 la ligne 4 du port 3,

```
PRT3DR |= 0x10;
```

Cette instruction se traduit par PRT3DR reçoit PRT3DR OU (0001 0000) en binaire
Exemple si PRT3DR = XXXX XXXX, PRT3DR = PRT3DR OU 0001 0000 = XXX1 XXXX
On ne modifie ainsi que la ligne 4 du port 3 sans modifier les autres.

Mise à « 0 » d'un bit d'un port :

Instruction C permettant de mettre à 0 la ligne 4 du port 3,

```
PRT3DR &= ~0x10;
```

Cette instruction se traduit par PRT3DR reçoit PRT3DR et le complément de (0001 0000)en binaire soit PRT3DR ET (1110 1111) en binaire.
Exemple si PRT3DR = XXXX XXXX, PRT3DR = PRT3DR ET 1110 1111 = XXX0 XXXX
On ne modifie ainsi que la ligne 4 du port 3 sans modifier les autres.

Lecture d'un bit d'un port :

Pareillement la lecture se fait à l'aide d'un masque en utilisant un test IF.
Exemple : Suite d'instructions permettant de recopier l'état du bit 0 du port 2 sur le bit 1 du port 0.

```
if (PRT2DR & 0x01)
            {
            PRT0DR |= 0x02;
            }
      else
            {
            PRT0DR &=~ 0x02;
            }
```

Les masques

Voici un aide-mémoire rappelant les masques à utiliser pour sélectionner un seul bit d'un port.

Bit du port	Masque en binaire	Masque en hexa
0	« 00000001 »	« 01 »
1	« 00000010 »	« 02 »
2	« 00000100 »	« 04 »
3	« 00001000 »	« 08 »
4	« 00010000 »	« 10 »
5	« 00100000 »	« 20 »
6	« 01000000 »	« 40 »
7	« 10000000 »	« 80 »

Utilisation de macro

Trouvé sur le forum de psoczone, une macro permettant de simplifier le codage de l'écriture d'un bit. Je cite :

Par contre si vous souhaitez mettre simplement une pin à l'état haut ou à l'état bas je vous conseillerai d'utiliser cette macro (que j'utilise moi-même) et qui simplifie votre code:

Code :

```
#define setBit( MSKN_REG, MASK )    (MSKN_REG |= MASK)
#define clearBit( MSKN_REG, MASK )   (MSKN_REG &= ~MASK)
```

De sorte que pour mettre la pin 5 du port0 à l'état haut vous n'auriez plus qu'a faire un :

```
setBit(PRT0DR,0x20);
clearBit(PRT0DR,0x20);
```

Utilisation d'un module LED

Cette autre astuce m'a été donnée par M. Guy Pirat, enseignant au Lycée J.Ferry de Cannes.

Cette solution consiste à placer un module LED dans l'espace chip. Ce module peut être renommé en fonction du rôle de la broche. Dans les propriétés du module on choisit le port et la broche associés ainsi que le niveau sur lequel le signal est actif (état bas ou état haut). Ainsi, on disposera de fonctions en C (voir la Datasheet associée et le fichier .h) permettant de mettre ce signal à 1, à 0, ou de l'inverser (Led_On , Led_Off, Led_Invert). Cette méthode a l'avantage d'être simple et lisible.

Quelques exemples de projets

Voici quelques exemples de petits projets simples à réaliser avec le kit d'évaluation. Les projets sont classés par ordre de difficultés.

- Un amplificateur non inverseur de gain 2 : V_{IN} = 0 à 2,5V et V_S = 0 à 5V
- Faire clignoter une led à une fréquence donnée (environ 1Hz) en utilisant un PWM.
- Allumer une led lorsque l'on appuie sur un bouton poussoir à l'aide d' un programme en C.
- Sortir les 4 bits de poids fort d'un convertisseur A-N sur 4 leds.
- Sortir le résultat d'une conversion A-N sur un bargraph à 4 leds.
- Générer un signal PWM de fréquence donnée (proche de 40 kHz) dont le rapport cyclique est donnée par une tension analogique (V_{IN} = 0V,η=0%, V_{IN} = 5V, η=100%).
-

Bibliographie

Ce document a été rédigé essentiellement à partir de documents Cypress, en particulier :

Présentation des circuits PSoC par Philippe LARCHER, ingénieur d'application chez Cypress : document disponibles sur le site PsoCzone (inscription préalable nécessaire).

Beaucoup de documents très intéressants : cours, exemples de projets, etc..., mais aussi des posts sur le forum, sur le site psoczone.com, site francophone !

http://psoczone.com/

Retrouvez d'autres cours et documents sur :

http://www.louisreynier.com