REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou

Faculté De Génie Electrique et Informatique

Département d'Electronique

Mémoire de fin d'études

En vue d'obtention du diplôme Master en Electronique

Option: Réseau et Télécommunication

THEME

Utilisation du Piccolo et ses applications

Promoteurs:

M^r LAGHROUCHE

M^r TAHANOUT

Réalisé par :

M^elle ZERROUKI Malika

M^r SELLOUM Mokrane

Remerciements

Merci à ALLAH, et nulle remerciement ne Lui sera suffisant, l'unique Dieu, l'Omniscient, l'Omnipotent, le Pur, pour nous avoir éclairé le droit chemin, et pour tous ses bienfaits apparents et cachés.
Merci au messager d'ALLAH qui a porté la charge du message.

Nous remercions tout d'abord tous les membres de jury d'avoir accepté d'examiner notre travail.

On tient à remercier aussi notre Promoteur Mr Laghrouche et notre Co-Promoteur Mr Tahanout pour tous les efforts qu'ils ont bien voulu nous consacrer durant toute la phase de préparation de ce modeste travail.

Dédicaces

A mes très chers parents Ouardia et Abdarehmane

Que pourrai-je vous dire !!
Les mots me manquent pour exprimer toute la reconnaissance, la fierté et le profond amour que je vous porte pour les sacrifices que vous avez consentis pour ma réussite.
Que vous trouvez ici le témoignage de mon attachement, ma reconnaissance, ma gratitude et mon respect.
Que Dieu vous préservent bonne santé et longue vie..

A mes chers frères Yacine, Kamel et Abdallah

J'espère avoir atteint le seuil de vos espérances. Que ce travail soit l'expression de ma profonde affection. Je vous remercie pour le soutient moral et l'encouragement que vous m'avez accordés. Je vous souhaite un brillant avenir.

A mes soeurs Razika et son mari, Anissa et Samira

Je vous adresse mon attachement et mes affections les plus sincères pour tout l'aide que vous m'avez toujours apporté. Sachez, chères soeurs que je vous dois beaucoup si je suis aujourd'hui tel que je suis.
Enfin, je vous souhaite tout le bonheur du monde.

A mon cher fiancé Mohammed ainsi qu'à ma belle famille

A mes oncles, mes tantes en particulier tante Sihem

Malika

Dédicaces

A mes très chers parents Nadjia et Mehenna

Que pourrai-je vous dire !!
Les mots me manquent pour exprimer toute la reconnaissance, la fierté et le profond amour que je vous porte pour les sacrifices que vous avez consentis pour ma réussite.
Que vous trouvez ici le témoignage de mon attachement, ma reconnaissance, ma gratitude et mon respect.
Que Dieu vous préservent bonne santé et longue vie..

A mes chers frères Akli et Sid-Ali

J'espère avoir atteint le seuil de vos espérances. Que ce travail soit l'expression de ma profonde affection. Je vous remercie pour le soutient moral et l'encouragement que vous m'avez accordés. Je vous souhaite un brillant avenir.

A mes soeurs Sonia, Naima et son mari Hamid

Je vous adresse mon attachement et mes affections les plus sincères pour tout l'aide que vous m'avez toujours apporté. Sachez, chères soeurs que je vous dois beaucoup si je suis aujourd'hui tel que je suis.
Enfin, je vous souhaite tout le bonheur du monde.

A mes tantes Ouiza, Drifa et Razika

Que ce travail soit l'expression de ma profonde affection et remerciement pour vous.

A mes chères cousins Sofiane et Hamid et cousines Fedoua, Djodjo
Ainsi qu'à Ourida et son fils Nadir

A mon cher Ami Amar et sa femme Safia

A ma très chère grand-mère Ferroudja

Mokrane

Chapitre III : Tests sur le Piccolo C2000.

Chapitre IV : Réalisation d'un accéléromètre en utilisant le Piccolo C2000

Introduction Générale

De nos jours, le domaine de l'électronique se développe et s'améliore d'une façon exponentielle. Les dispositifs électroniques sont de plus en plus compacts, simple à utiliser et leurs coûts est très réduit. En effet ces dispositifs voient leurs tailles se réduire de jour en jour et leurs domaines d'applications s'élargi de plus en plus.

Les conséquences pratiques ont été notamment l'intégration de fonctions électroniques de plus en plus complexes et différentes dans la majeure partie des domaines techniques (industriels ou domotiques) et des objets de la voie courante. Généralement, ces fonctions sont engendrées par l'un des dispositifs de base de l'électronique qui est le microcontrôleur.

Dans notre travail, nous nous intéressons à un dispositif récemment développé par Texas Instrument qui est le Piccolo C2000 Control Stick qui est à base du microcontrôleur TMS320. Ce dernier offre un vaste éventail d'option en matière de performance, de flash, d'intégration analogique et de périphérique basé sur la commande afin de satisfaire les demandes diverses des applications de commande en temps réel à bas coût.

Ce travail a été réalisé au sein du département d'électronique de l'université de Mouloud Mammeri. Il est organisé en quatre chapitres.

Dans le premier chapitre, nous avons vu la description des différents blocs qui composent le Piccolo C2000. Le deuxième chapitre est consacré a la partie logicielle dont on explique le fonctionnement du logiciel « Code Composer Studio » (CCS). Dans le

troisième chapitre, nous présentons une application pour acquérir différents formes de signaux avec le Piccolo C2000. Enfin, dans le quatrième chapitre nous donnons les résultats pratiques d'un accéléromètre que nous avons réalisé à base du capteur ADXL311 et du piccolo C2000.

I .1 Introduction

Le piccolo C2000 control stick est un dispositif électronique avec interface USB développé par les ingénieurs de Texas Instrument. Grâce au microcontrôleur TMSF2802X/3X, le piccolo C2000 control stick offre de multiples performances dans plusieurs domaines et en temps réel.

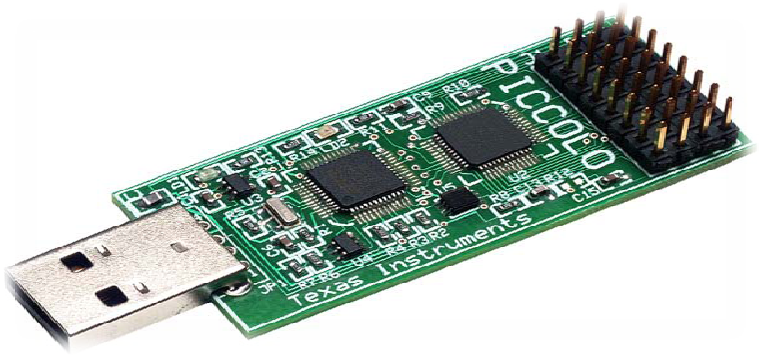


Fig.1 Piccolo C2000 controlStick

I.2 Les caractéristiques du Piccolo c2000 controlStick

Largement plus grand qu'une clé USB, le piccolo c2000 control stick a les caractéristiques suivantes :

- Microcontrôleur Piccolo TMS 320F2802X/3X
- CPU C28X 32bits
- Fréquence de travail jusqu'à 60MHZ
- CAN de 12 bits
- Mémoire intégrée de 32 à 128 ko
- Modulateur haute résolution
- Boitier TSSOP à partir de 38 broches

I.3 Les avantages du piccolo c2000 controlStick

Avec le piccolo c2000 control stick, on peut citer comme avantages :

- Sa performance 32 bits pour le contrôle en temps réel
- Le cout des systèmes réduit
- Simplicité d'utilisation
- Format compact

I.4 Les inconvénients du Piccolo C2000 :

- La mémoire reste insuffisante pour d'autres applications plus complexes, d'où l'apparition de la série Delfino par Texas instrument avec des mémoires qui peuvent atteindre 512 KB**.**
- La fréquence de travail est de 60Mhz, elle reste insuffisante par rapport aux fréquences de la série Delfino qui peuvent atteindre jusqu'à 300Mhz.

I.5 Les domaines d'application du piccolo C2000 controlStick

Parmi les domaines d'applications du piccolo C2000 control stick, on peut citer :

- Convertisseur solaire
- Eclairage à LED
- Véhicule électrique
- Appareil électroménager
- Alimentation numérique
- Commande de moteur
- Energies renouvelables
- CPL : communication sur courant porteur

I.6 Schéma de développement des contrôleurs C2000

Dans la famille piccolo C2000 controlStick on trouve deux séries. Comme le montre la figure (Fig 2). On constate deux grandes familles du piccolo C2000, tout dépend de microcontrôleur qui les compose, on trouve donc la famille F2802x ainsi que la famille F2803x. Chaque famille a ses propres caractéristiques.

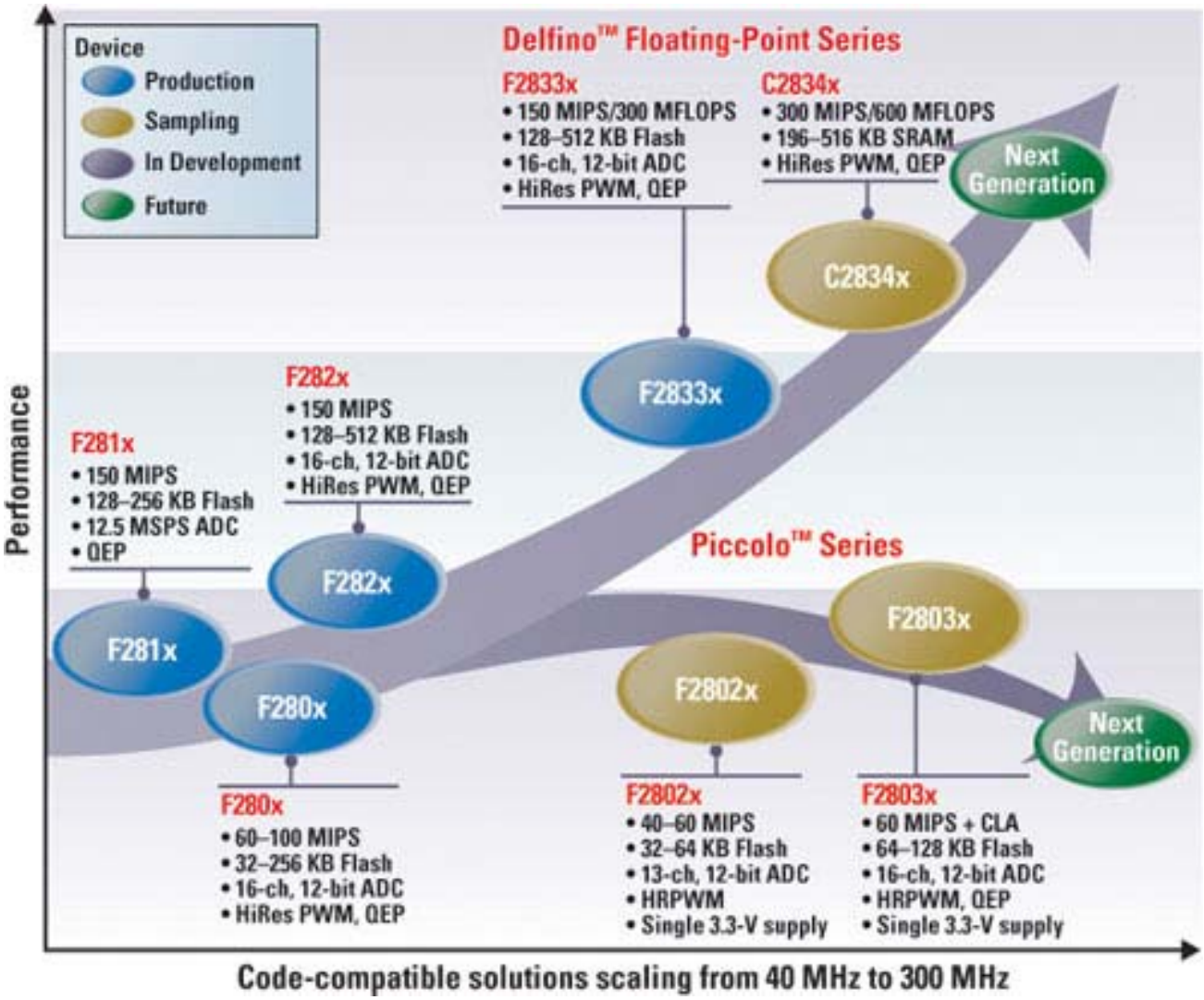


Fig.2 Schéma de développement des contrôleurs C2000

Toute notre étude se porte sur un model de la famille F2802x qui est le TMS320F28027.

I.7 Schéma synoptique du piccolo C2000 control stick

Voici le schéma synoptique du Piccolo C2000 control stick :

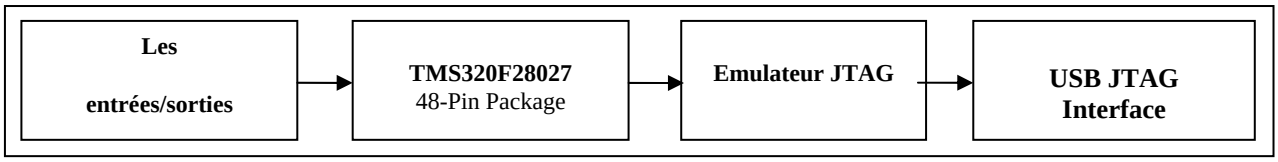


Fig.3 Schéma synoptique du Piccolo C2000 control stick

On donne la description de chaque composant :

I.7.1 Les entrées/sorties

Ce composant contient les différentes pins qui servent à brancher les différents composants électroniques. Quelques une de ces pins peuvent servir en tant qu'entrées comme elles peuvent fonctionner en tant que sorties. Le tableau ci-contre illustre les différentes pins sur le Piccolo :

Tab.1 : Les différentes entrées/sorties

1 ADC-A7	2 ADC-A2 COMP1 (+VE)	3 ADC-A0 Vref-HI	4 3V3
5 ADC-A4 COMP2 (+VE)	6 ADC-B1	7 EPWM-4B GPIO-07	8 TZ1 GPIO-12
9 SCL GPIO-33	10 ADC-B6	11 EPWM-4A GPIO-06	12 ADC-A1
13 SDA GPIO-32	14 ADC-B7	15 EPWM-3B GPIO-05	16 5V0
17 EPWM-1A GPIO-00	18 ADC-B4 COMP2 (-VE)	19 EPWM-3A GPIO-04	20 SPISOMI GPIO-17
21 EPWM-1B GPIO-01	22 ADC-B3	23 EPWM-2B GPIO-03	24 SPISIMO GPIO-16
25 SPISTE GPIO-19	26 ADC-B2 COMP1 (-VE)	27 EPWM-2A GPIO-02	28 GND
29 SPICLK GPIO-18	30 GPIO-34 (LED)	31 PWM1A-DAC (Filtered)	32 GND

A chaque programme, des pins lui sont réservé.

Si on prend le programme « ADC continuous » qu'on étudiera au chapitre III, 11 pins lui sont réservés. Ces pins sont représentés en bleu comme le montre le tableau ci-contre (Tab.2) :

Tab.2 Pins réservées pour l'application « ADC continuous »

1 ADC-A7	2 ADC-A2 COMP1 (+VE)	3 ADC-A0 Vref-HI	4 3V3
5 ADC-A4 COMP2 (+VE)	6 ADC-B1	7 EPWM-4B GPIO-07	8 TZ1 GPIO-12
9 SCL GPIO-33	10 ADC-B6	11 EPWM-4A GPIO-06	12 ADC-A1
13 SDA GPIO-32	14 ADC-B7	15 EPWM-3B GPIO-05	16 5V0
17 EPWM-1A GPIO-00	18 ADC-B4 COMP2 (-VE)	19 EPWM-3A GPIO-04	20 SPISOMI GPIO-17
21 EPWM-1B GPIO-01	22 ADC-B3	23 EPWM-2B GPIO-03	24 SPISIMO GPIO-16
25 SPISTE GPIO-19	26 ADC-B2 COMP1 (-VE)	27 EPWM-2A GPIO-02	28 GND
29 SPICLK GPIO-18	30 GPIO-34 (LED)	31 PWM1A-DAC (Filtered)	32 GND

Remarque :

Les pins N°4 (3v3), N°16(5V0), N°28(GND) et N°32 (GND) peuvent servir pour une alimentation externe.

I.7.2 TMS320F28027

I.7.2.A Préambule

Le TMS320F28027 fait partie de la grande famille des microcontrôleurs TMS320 à traitement de signal numérique DSP (Digital Signal Processor) fabriqués par Texas Instrument.

Comme toutes les DSP, le TMS320F28027 a une architecture optimisée pour effectuer des calculs complexes mais aussi pour accéder très facilement à un grand nombre d'entrée/sortie numérique ou analogique.

Le TMS320F28027 se présente sous forme d'un boitier TSSOP (Thin Shrink Small Outline Package) de 38 à 48 pins. Pour notre cas, il comportera 48 pins. Voici la figure ci-dessous montrant le TMS320F28027.

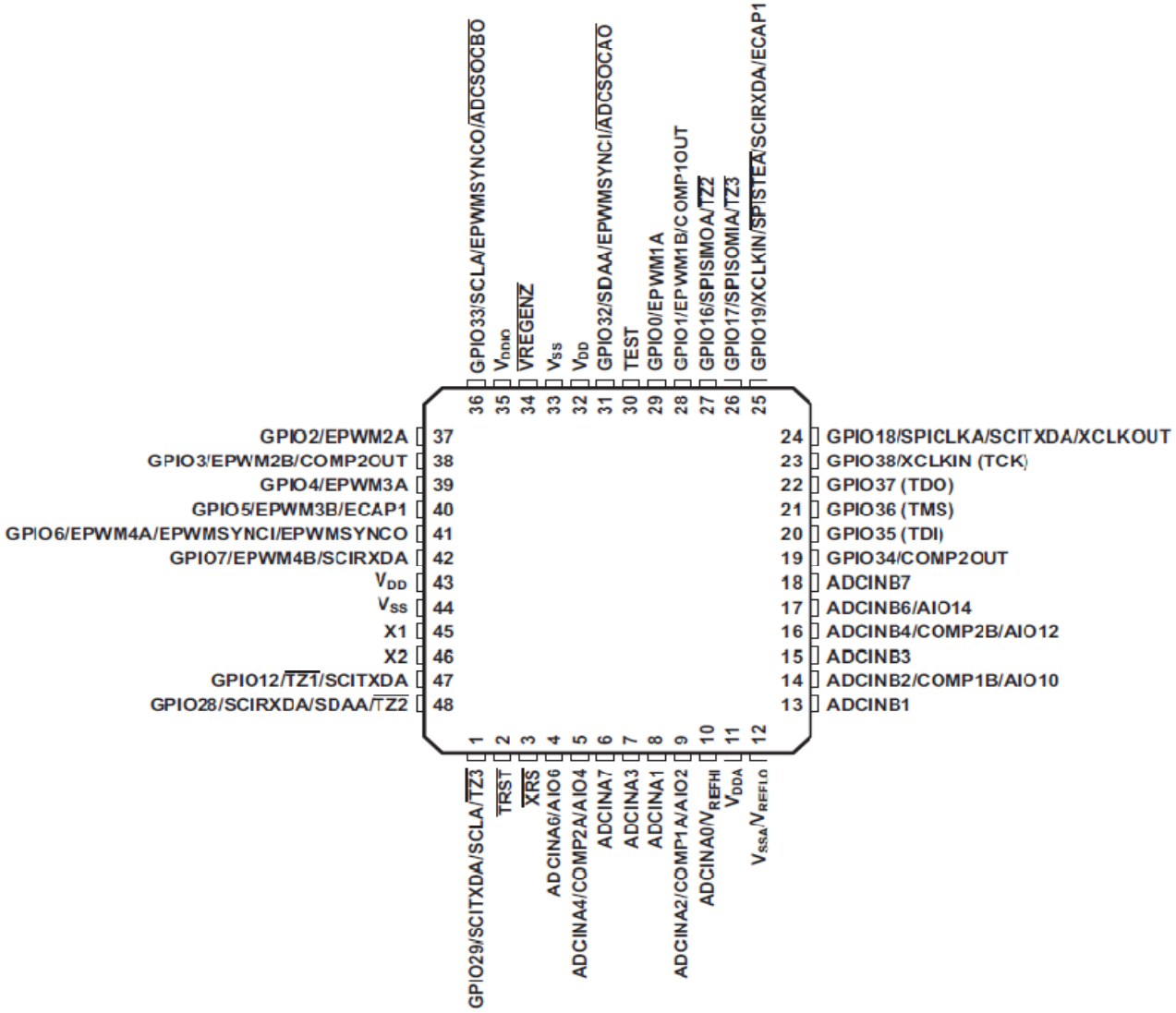


Fig.4 Schéma externe du TMSF28027

La fonction de chaque pin est donnée dans l'annexe [N°1].

I.7.2.B Configuration interne du TMS320F28027

Voici l'architecture interne du TMS320F28027 avec ses différents composants :

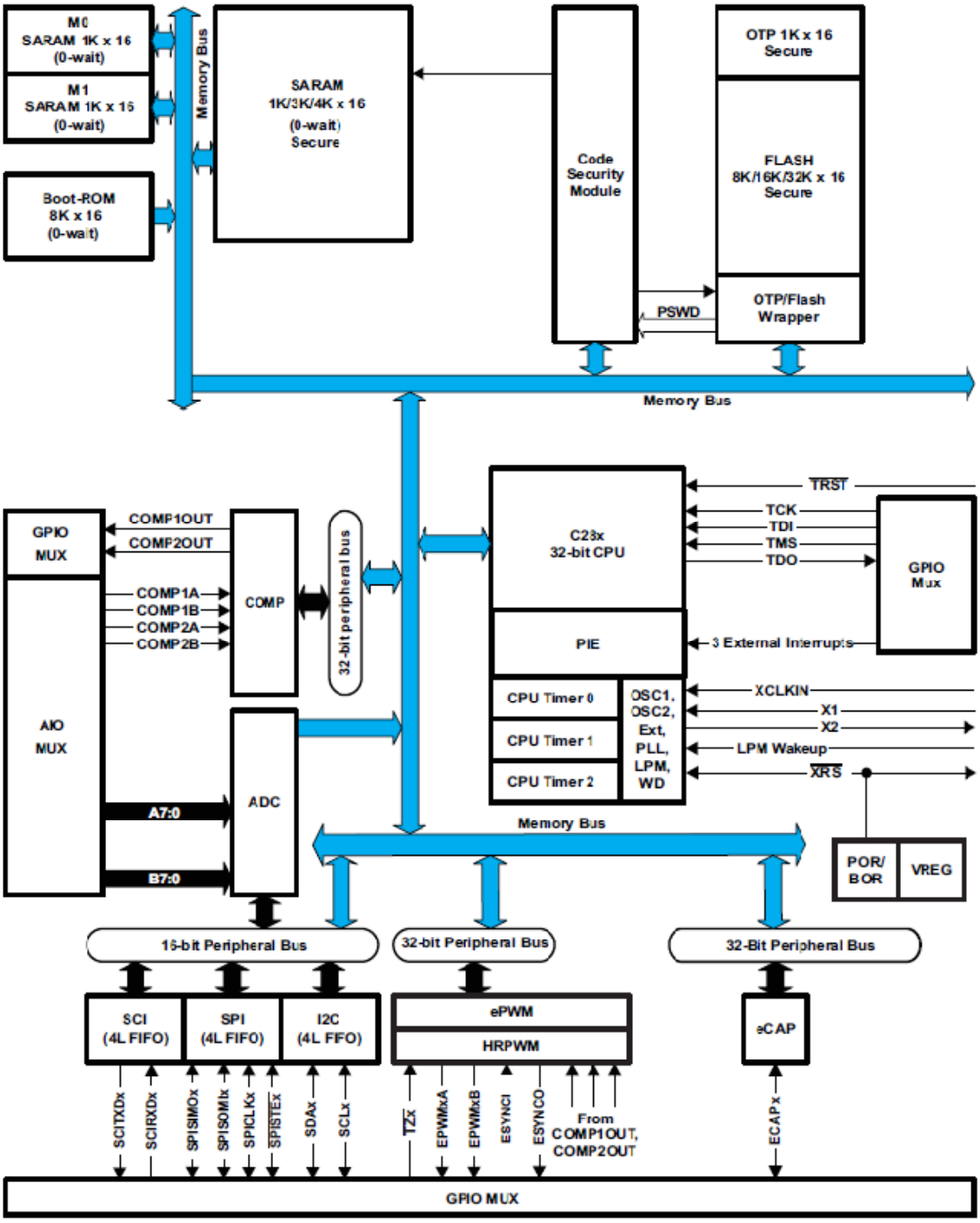


Fig.5 Configuration interne du TMS320F28027

Comme tout microcontrôleur le TMS320F28027 contient des registres, des RAMs, un boot ROM, des bus de données, des multiplexeurs.

On s'intéressera essentiellement aux blocs les plus importants comme la mémoire, la CPU ainsi qu'aux bus qui sont représentés par le bus mémoire, les bus périphériques et le bus I2C.

I.7.2.B.1 Le bus memoire (Memory bus)

Le bus mémoire sert à véhiculer les données entre la mémoire, les périphériques et le processeur. L'architecture du bus mémoire contient un bus de lecture de programme, un bus d'écriture de données et un bus de lecture de données.

Le bus de lecture de programme contient 22 lignes d'adresse et 32 lignes de données. Les bus d'écriture et de lecture des données contiennent chacun 32 lignes d'adresses et 32 lignes de données.

Grâce a l'architecture du bus mémoire (bus multiples), la CPU C28x cherche l'instruction, la lit et l'écrit au en un seul cycle.

I.7.2.B.2 Les bus périphériques (Peripheral Bus)

Se présentant en deux versions dont une supportant que 16 bits d'accès et l'autre 16 à 32 bits d'accès, les bus périphériques permettent d'une façon générale de véhiculer des données entre les périphériques et le bus mémoire.

I.7.2.B.3 Le bus I2C :

- ✓ **Définition :**

Le bus I²C permet d'établir une liaison série synchrone entre 2 ou plusieurs composants. Il a été créé dans le but d'établir des échanges d'informations entre circuits intégrés se trouvant sur une même carte. Son nom, d'ailleurs, traduit son

origine : Inter Integrate Circuit, ou I.I.C., ou plus communément I²C (I carré C). Ce bus est le descendant du CBUS, qui est de moins en moins utilisé.

La liaison entre les composants se fait grâce à 3 fils qui sont:

- Signal de données: SDA
- Signal d'horloge: SCL
- Signal de référence électrique: Masse

✓ **Caractéristiques du Bus I2C :**

- Deux lignes uniquement (SDA et SCL) + masse
- Une adresse unique pour chaque périphérique
- Bus multi-maitre, détection des collisions et arbitrage
- Bus série, 8 bits, bidirectionnel de 10 jusqu'à 400 Kbps (fast mode)
- Filtrage intégré : réjection des pics parasites

I.7.2.B.4 La mémoire :

La taille totale du TMS320F28027 est repartie en 4 blocs qui sont :

✓ **La Flash :**

Sa taille est de 16K x16 (16K mots à 16 bits) réparties en 4 secteurs de 8k x16 chacun, elle est programmable et effaçable. Sa plage mémoire est repartie comme le montre le tableau Tab.3 :

Tab.3 Organisation de la mémoire Flash

ADDRESS RANGE	PROGRAM AND DATA SPACE
0x3F 0000 – 0x3F 1FFF	Sector D (8K x 16)
0x3F 2000 – 0x3F 3FFF	Sector C (8K x 16)
0x3F 4000 – 0x3F 5FFF	Sector B (8K x 16)
0x3F 6000 – 0x3F 7F7F	Sector A (8K x 16)
0x3F 7F80 – 0x3F 7FF5	Program to 0x0000 when using the Code Security Module
0x3F 7FF6 – 0x3F 7FF7	Boot-to-Flash Entry Point (program branch instruction here)
0x3F 7FF8 – 0x3F 7FFF	Security Password (128-Bit) (Do not program to all zeros)

✓ **La Boot ROM :**

Sa taille est de 8K x16, elle est programmée en mode usine avec un logiciel de démarrage (boot loader). L'utilisateur peut choisir de démarrer normalement avec le programme de la ROM ou avec un programme spécifique se trouvant dans la FLASH ROM.

✓ **SARAM (Single Access Memory)**

- **1. M0, M1 SARAM :**
  Ces RAM, on chacune d'entre elle la taille de 1K x16.
- **2. L0 SARAM :**
  Sa taille est de 4K x16.

REMARQUE :

Dans le TMS320F28027, la mémoire est mappée pour les deux espaces, données et programmes (DATA SPACE et PROGRAM SPACE).

I.7.2.B.5 La CPU (Central Processing Unit)

➢ **Définition**

C'est un circuit processeur appartenant à la famille des VLSI (Very Large Scale Integration) capable d'effectuer séquentiellement et automatiquement des suites d'opérations arithmétiques élémentaires.

➢ **Caractéristiques**

La CPU contient les éléments suivants :

- **L'UAL (Unité Arithmétique et Logique)**
  Prend en charge les calculs arithmétiques élémentaires et les tests.
- **Les registres**
  Se sont des mémoires de petite taille, suffisamment rapides pour que l'UAL puisse manipuler leur contenu à chaque cycle de l'horloge.

- ✓ **Compteur ordinal**

Ce registre contient l'adresse mémoire de l'instruction en cours d'exécution.

- ✓ **Accumulateur**

Ce registre est utilisé pour stocker les données en cours de traitement par l'UAL.

- ✓ **Registre d'adresses**

Il contient toujours l'adresse de la prochaine information à lire par l'UAL, soit la suite de l'instruction en cours, soit la prochaine instruction.

- ✓ **Registre d'instruction**

Il contient l'instruction en cours de traitement.

- ✓ **Registre d'état**

Il sert à stocker le contexte du processeur, ce qui veut dire que les différents bits de ce registre sont les drapeaux (flags) servant à stocker des informations concernant le résultat de la dernière instruction exécutée.

- ✓ **Pointeur de pile**

Ce type de registre, dont le nombre varie en fonction du type de processeur, contient l'adresse du sommet de la pile

- ✓ **Registres généraux**

Ces registres sont disponibles pour les calculs.

- **Architecture pipeline**

Permet de découper temporellement les traitements à effectuer.

- **Unité de calcul en virgule flottante (Floating Point Unit)**

Permet d'accélérer les calculs sur des nombres réels codés en virgule flottante.

- **La mémoire cache**

Permet d'accélérer les traitements, en diminuant les temps d'accès à la mémoire. Ces mémoire tampons sont en effet beaucoup plus rapides que la RAM et ralentissent moins le CPU. La cache instructions reçoit

les prochaines instructions à exécuter, la cache donnée manipule les données.

- **Les bus**

  Le CPU possède trois types de bus :

  - ✓ Un bus de donnée
  - ✓ Un bus d'adresse
  - ✓ Un bus de contrôle

I.7.2.B.6 Les Timers (Timer0,Timer1 et Timer2)

Ces Timer sont des compteurs 32 bits, accessibles en lecture et écriture. L'horloge peut être interne ou externe par l'intermédiaire d'un pré-diviseur de 16 bits.

Le Timer 0 est fait pour une utilisation générale, il est surtout connecté au block PIE (Peripheral Interrupt Expansion).

Le Timer1 est fait aussi pour une utilisation générale, il peut être relié à l'INT13 de la CPU.

Le Timer2 est réservé pour la DSP/BIOS et connecté à l'INT14 de la CPU.

La figure suivante (Fig.6) représente les différents Timers ainsi leurs connexions.

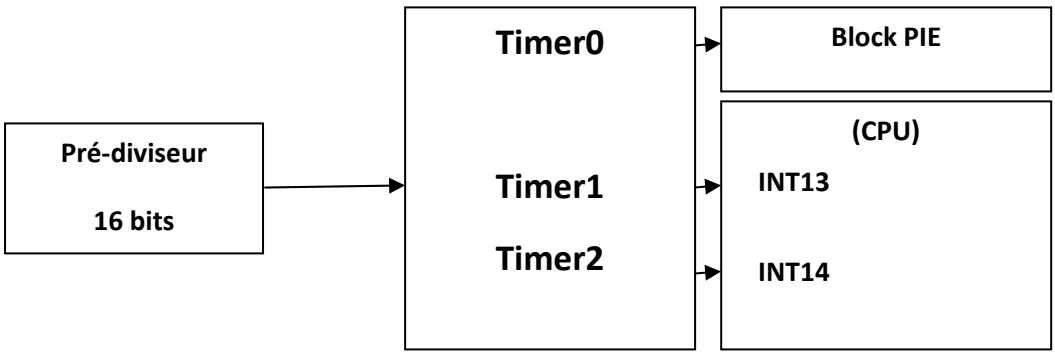


Fig.6 Schéma synoptique des TIMERs

I.7.2.B.7 L'horloge

Le circuit d'horloge permet de cadencer tous les échanges internes ou externes au microcontrôleur.

Dans le TMS320F28027, on trouve deux oscillateurs internes et un oscillateur externe. Au démarrage, l'oscillateur interne 1 est sélectionné en tant qu'horloge par défaut.
La fréquence nominale pour l'oscillateur interne est de 10MHZ.
Les deux registres INTOSC1TRIM et INTOSC2TRIM sont prévus pour permettre de découper d'une façon organisée les oscillateurs externes, ces derniers fournissent l'horloge pour le noyau, le watchdog et le Timer 2 du CPU.
Le registre CLKCTRL est ajouté pour configurer les options des oscillateurs internes et l'oscillateur externe.

I.7.2.B.8 Le Watchdog

Le Watchdog, est un composant matériel qui permet de palier au plantage du microcontrôleur. Ce plantage peut être :

- **Logiciel**
  Oublie de retour de routines, mis en boucle du programme sur une situation non prévue ou tout simplement erreur de structuration du logiciel.
- **Matériel**
  Parasites, chute de tensions.

Dans les deux cas, le blocage du programme peut avoir des conséquences catastrophiques, erreurs et/ou perte de données, états incertains des entrées/sorties, défaillance des interfaces de périphériques ...
D'où tout l'intérêt du Watchdog qui est un dispositif permettant de détecter automatiquement les anomalies de fonctionnement du logiciel, et si nécessaire interrompre le fonctionnement du processeur pour forcer un Reset ou une interruption prioritaire.

Caractéristiques

- Le timeout est de 131,072 cycles, soit 13ms.
- Le système doit fonctionner à une horloge de 10Mhz.

I.7.2.B.9 La PLL

I.7.2.B.9.1 Principe de la PLL

La boucle à verrouillage de phase ou Phase Locked Loop permet d'asservir la fréquence d'un VCO à la fréquence d'un signal de référence dans une certaine plage autour de la fréquence centrale f0.

Voici ci-contre le schéma synoptique de la structure de base d'une boucle à verrouillage de phase.

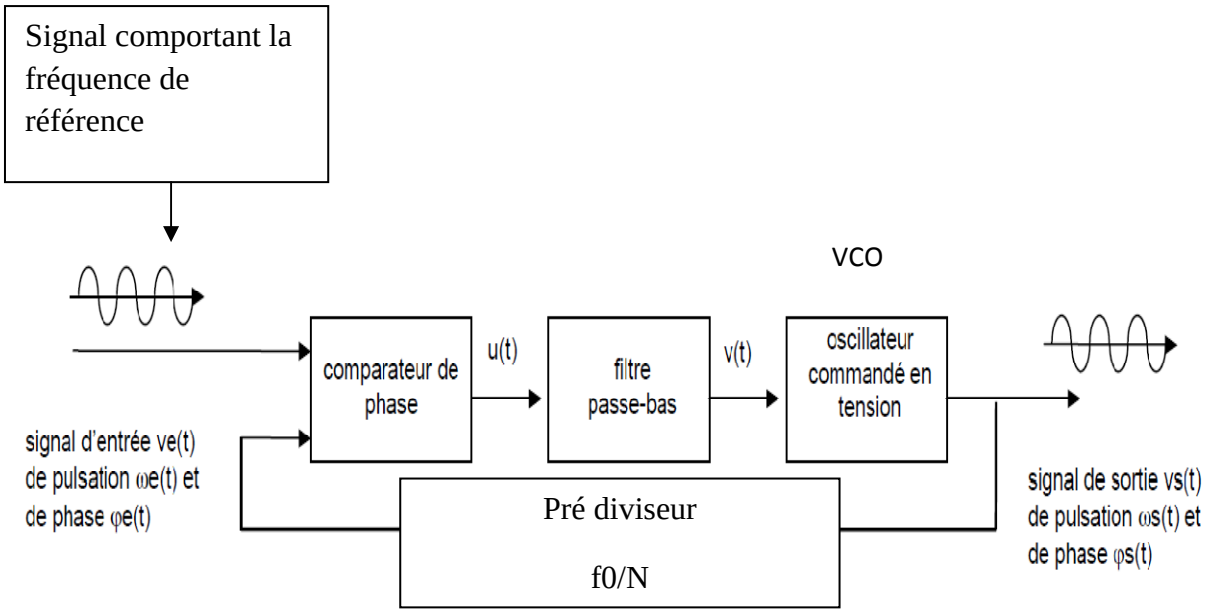


Fig.7 Structure de base de la boucle à verrouillage de phase

Les éléments de la PLL sont les suivants :

- L'oscillateur VCO donne une fréquence qui varie en fonction de la tension de commande V appliquée sur son entrée. Il est linéarisé autour de fo et caractérisé par sa pente Ko

$$Ko = \frac{\text{variation de la pulsation du signal de sortie}}{\text{variation de la tension de commande}} \quad \text{en radian/sV}$$

- La fréquence du VCO est comparée avec une fréquence de référence (consigne) grâce à un comparateur de phase (ou exclusif, comparateur phase-fréquence…)
- Le comparateur de phase fournit à sa sortie une tension u alternative dont la valeur moyenne v donnée par un passe-bas est proportionnelle au déphasage entre Ve et Vs. Il est caractérisé par un coefficient souvent noté Kd défini par :

$$Kd = \frac{\text{valeur moyenne de la tension en sortie}}{\text{déphasage entre les signaux d'entrée}} \quad \text{en volts/radian}$$

I.7.2.B.9.2 Fonctionnement

- en l'absence de signal injecté à l'entrée de la boucle, ou si la fréquence du signal injecté est en dehor de la plage de fonctionnement du VCO, la boucle est dite non verrouillée et fs=fo.
- Si on injecte dans la boucle un signal de fréquence fe voisin de fo, la PLL se verrouille et on aboutit au bout d'un temps bref à un état stable caractérisé par fs=fe.
- Une fois la boucle verrouillé ou accrochée, la fréquence d'entrée peut varier dans la plage de verrouillage sans que cette boucle ne décroche et on a toujour fs=fe.
- Si la fréquence d'entrée sort de la plage de verrouillage, la boucle décroche et on revient à la situation d'une boucle non verrouillée.

Voici le schéma ci-contre Fig.7 regroupant l'horloge, le watchdog et la PLL dans le TMS320F28027 :

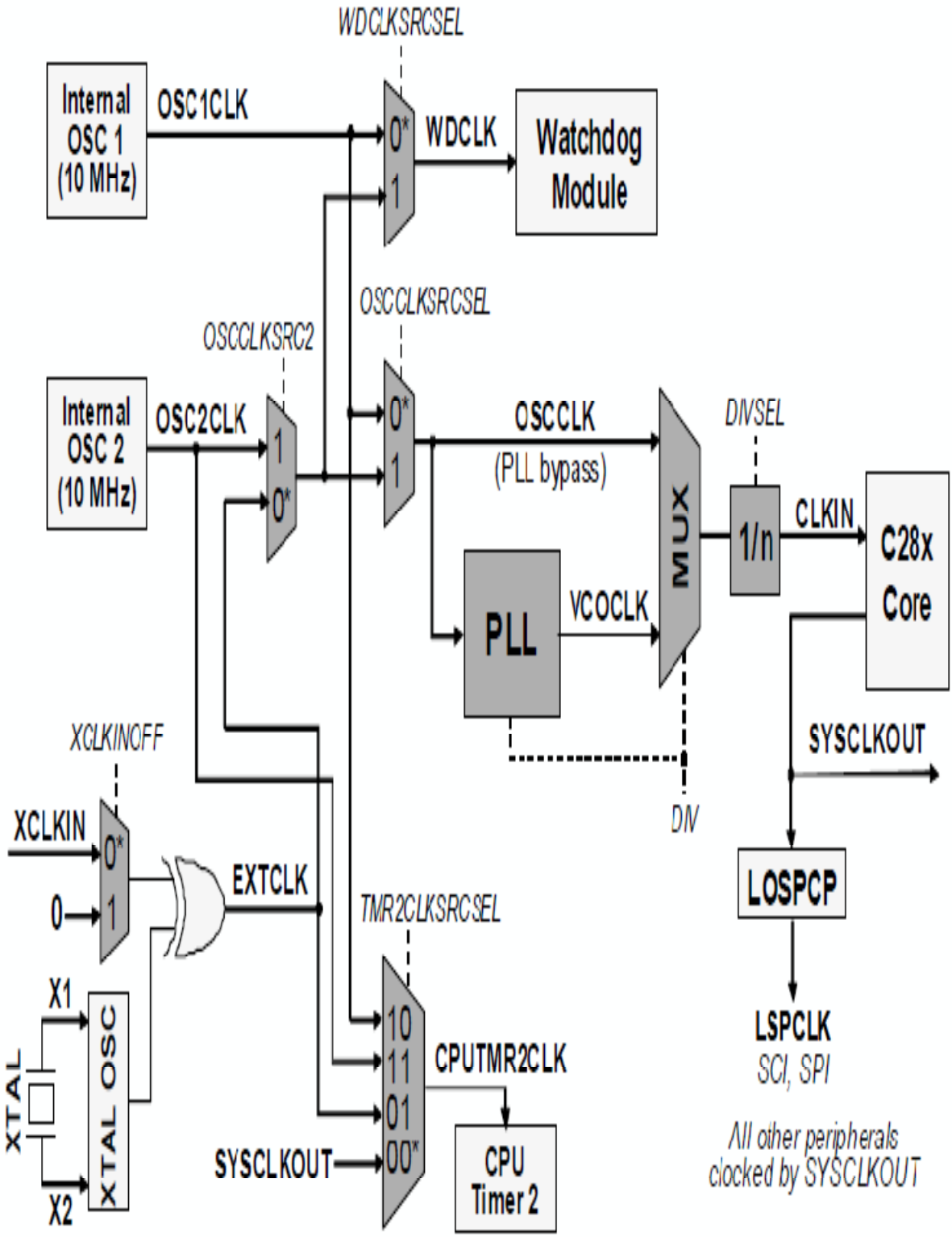


Fig.8 Schéma regroupant l'horloge, le Watchdog et la PLL dans le TMS320F28027

I.7.3 Emulateur Jtag

I.7.3.1 Préambule

Cet émulateur basé sur USB fournit une solution de débogage facile, portable, non intrusive et ciblée pour les processeurs et les DSP JTAG Analog Devices. Il effectue une large gamme de fonctions d'émulation, y compris l'exécution en une seule étape et à pleine vitesse avec des points d'interruption prédéfinis, et la visualisation et/ou la modification du contenu de registres et de mémoires. Grâce à sa capacité à détecter et à prendre en charge automatiquement de multiples tensions d'E/S, il permet la communication avec l'ensemble des processeurs et des DSP JTAG Analog Devices à l'aide d'un port USB 2.0 sur le PC hôte. Les applications et les données peuvent être facilement et rapidement testées et transférées entre l'émulateur et l'environnement de développement et de débogage CCS. L'architecture plug&play de l'USB permet à l'émulateur d'être automatiquement détecté et configuré par le système d'exploitation hôte. Il peut également être connecté et déconnecté de l'hôte sans ouvrir le PC ni mettre le PC hors tension.

Cette émulation est assurée par le dispositif FT2232D dont voici le schéma ci-contre Fig.8 :

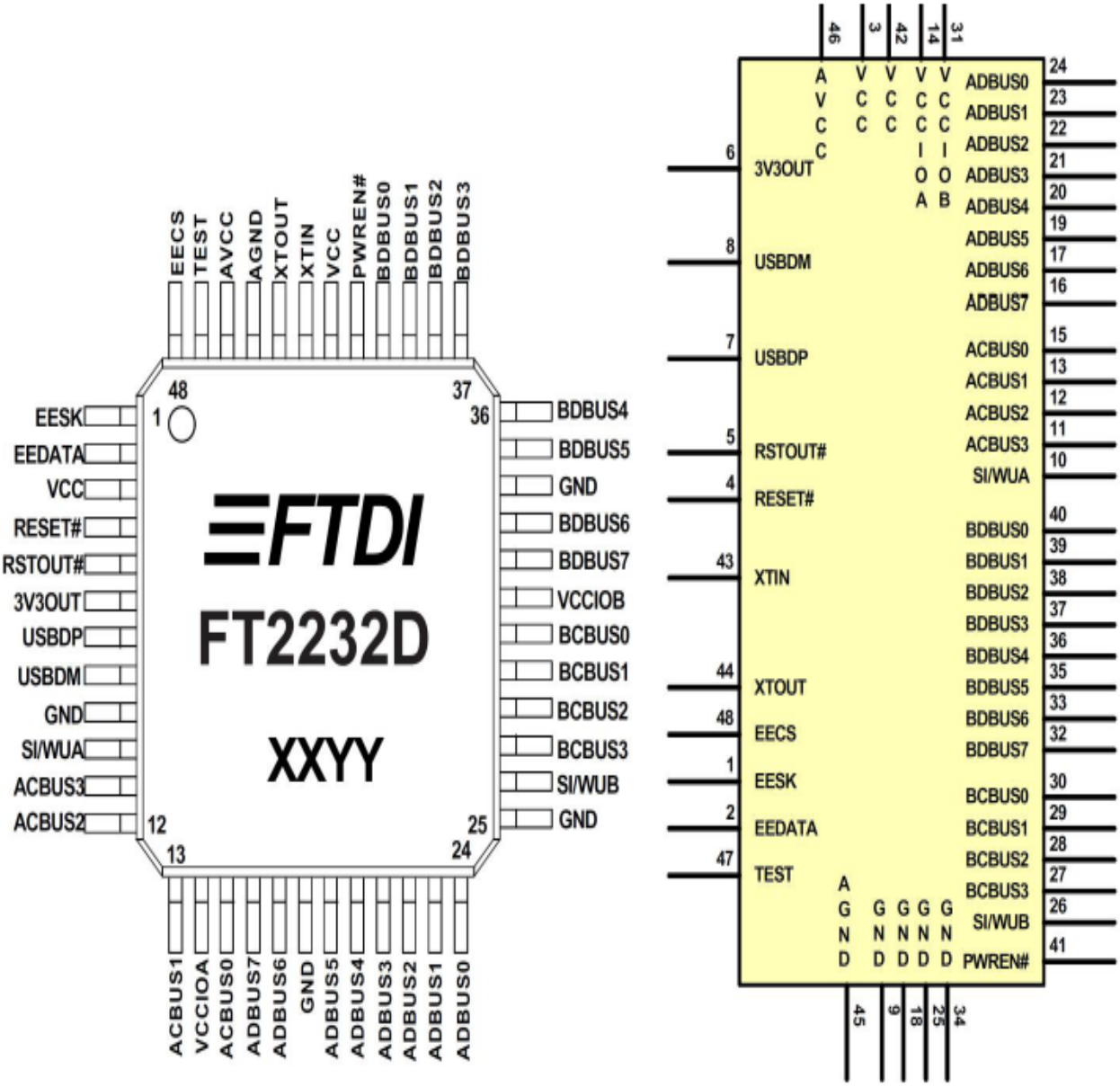


Fig.9 Schéma de l'émulateur FTDI FT2232D

La fonction de chaque pin de l'émulateur FTDI FT2232D est donnée dans l'annexe [N°2].

La figure suivante Fig.10 représente la structure interne de l'émulateur FTDI FT2232D avec le branchement de toutes ses pins.

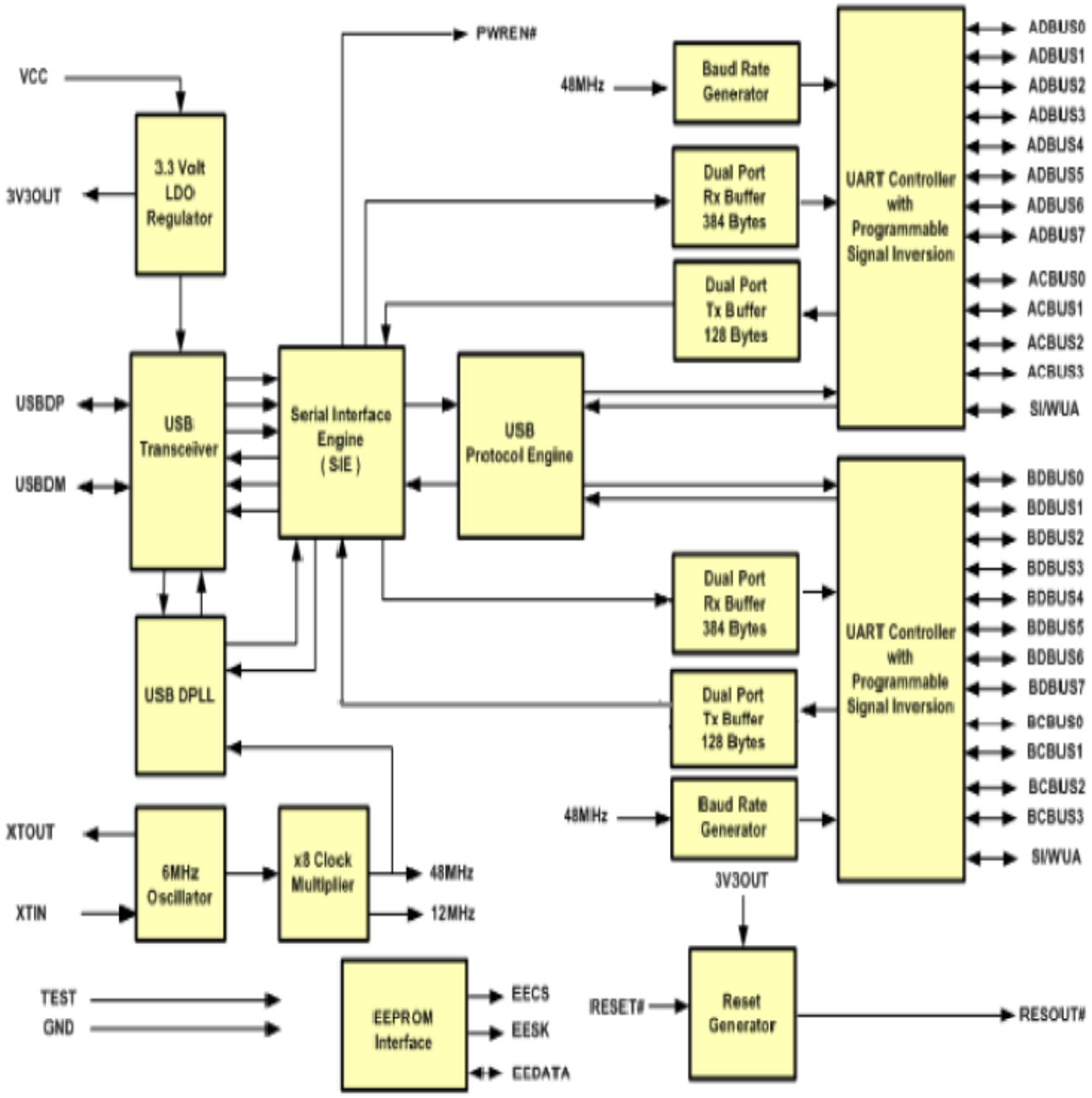


Fig.10 Structure interne de l'émulateur FTDI FT2232D

I.7.3.2 Caractéristiques de l'émulateur FTDI FT2232D

Voici ci-dessous les caractéristiques de l'émulateur FTDI FT2232D :

- USB-UART/FIFO.
- Gamme de tension d'alimentation : 3V à 5,25V, 4,35V à 5.25V.
- Température de fonctionnement : -40°C à +85°C.
- Nombre de broches : 48.
- Numéro générique : 2232.
- Type de boitier : LQFP (Low Quad Flat package).
- Tension d'alimentation max : 5,25V.
- Tension d'alimentation min : 3V.
- Tout le protocole USB est programmé sur une puce et pas de modification requise sur le firmware pour une programmation.
- Le taux de transfert est de 300 à 3 Mbaud.
- La vitesse de transfert de type USB au type parallèle FIFO est de 1Mbps.
- Compatibilité avec la norme USB 2.0 pour une vitesse max (12Mbps).

II.1 Introduction

Code Composer Studio (CCS) est un environnement intégré de développement de code pour les DSP de Texas Instrument. Il est fourni en standard avec l'équipement Piccolo C2000 ControlStick.

Le CCS fournit plusieurs outils pour faciliter la construction et la mise au point des programmes de DSP. Il comprend un éditeur de code source, un compilateur de langage C/C++, un assembleur, un éditeur de liens, et un environnement d'exécution qui permet de télécharger un programme exécutable sur un composant cible (le piccolo c2000 pour notre cas), de l'exécuter et déboguer au besoin. CCS comprend aussi des outils qui permettent l'analyse en temps réel d'un programme en cours d'exécution et des résultats produits. Finalement, il fournit un environnement de gestion de fichiers qui facilite la construction et la mise au point des programmes. Par ailleurs, d'autres bibliothèques peuvent être utilisées pour mieux exploiter le CCS c'est le cas par exemple de ControlSuite.

Nous allons, ci-après, détailler les principales fonctions de cet environnement de développement montrant ses capacités.

II.2 Définition de l'espace de travail (workspace) du CCS

A chaque exécution du CCS, la fenêtre suivante(Fig.11) apparait :

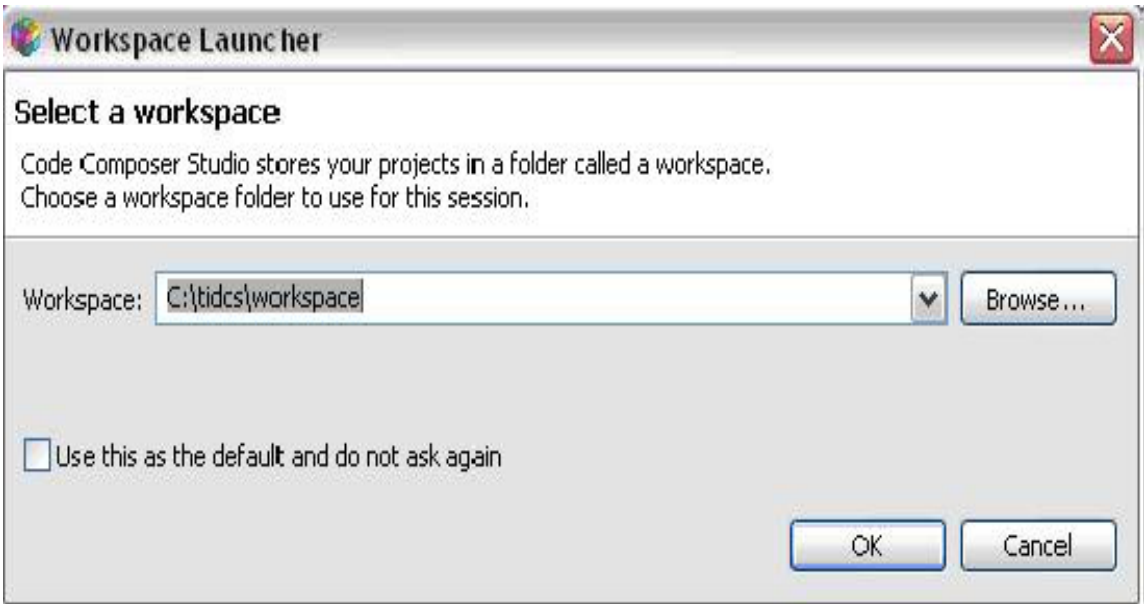


Fig.11 Répertoire du Workspace

Cette fenêtre porte le chemin de sauvegarde des différents programmes qui seront exécutés avec le CCS ainsi que tous les projets.

Le workspace sauvegarde d'une façon automatique tous ces projets lors de la fermeture du CCS.

On peut également changer le chemin du workspace et pour ça il faut choisir l'option « Switch workspace » dans l'onglet « file » du CCS. (Voir Fig.12)

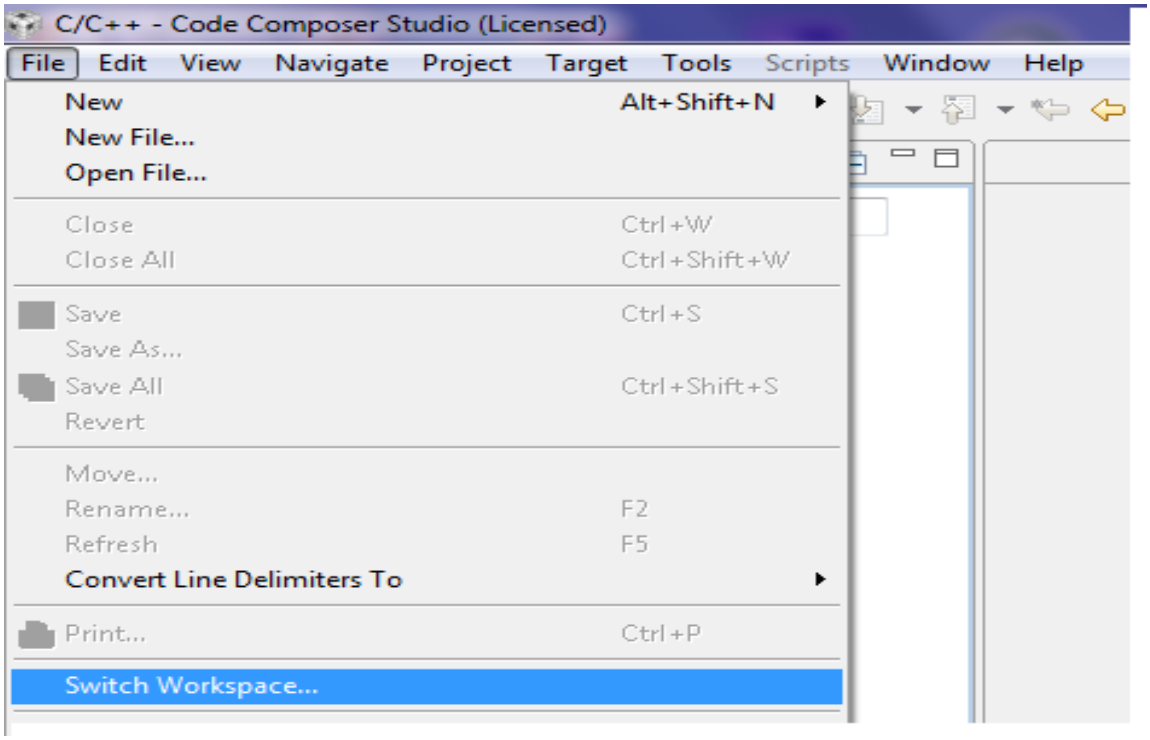


Fig.12 Changement de l'emplacement du Workspace

II.3 Accès aux projets

Pour accéder aux différents projets avec le CCS, on suit les étapes suivantes :

- Dans l'onglet « Project », on sélectionne l'option « Import existing CCS/CCE Eclipse Project ». (voir Fig.13).

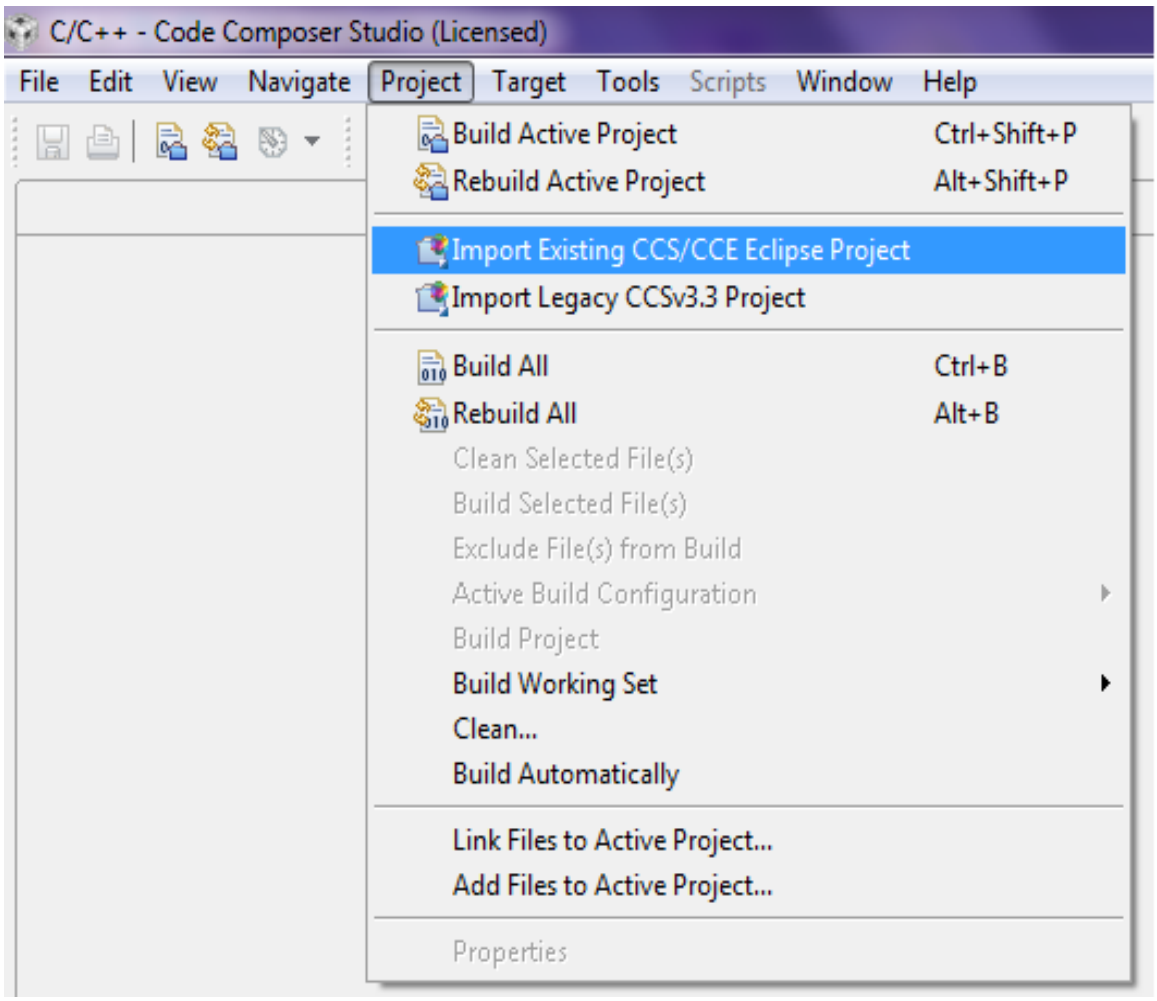


Fig.13 Importation des différents projets existants

- La fenêtre « Import » apparait, on clique sur « browse » pour accéder au dossier contenant les exemples de projets (voir Fig.14).

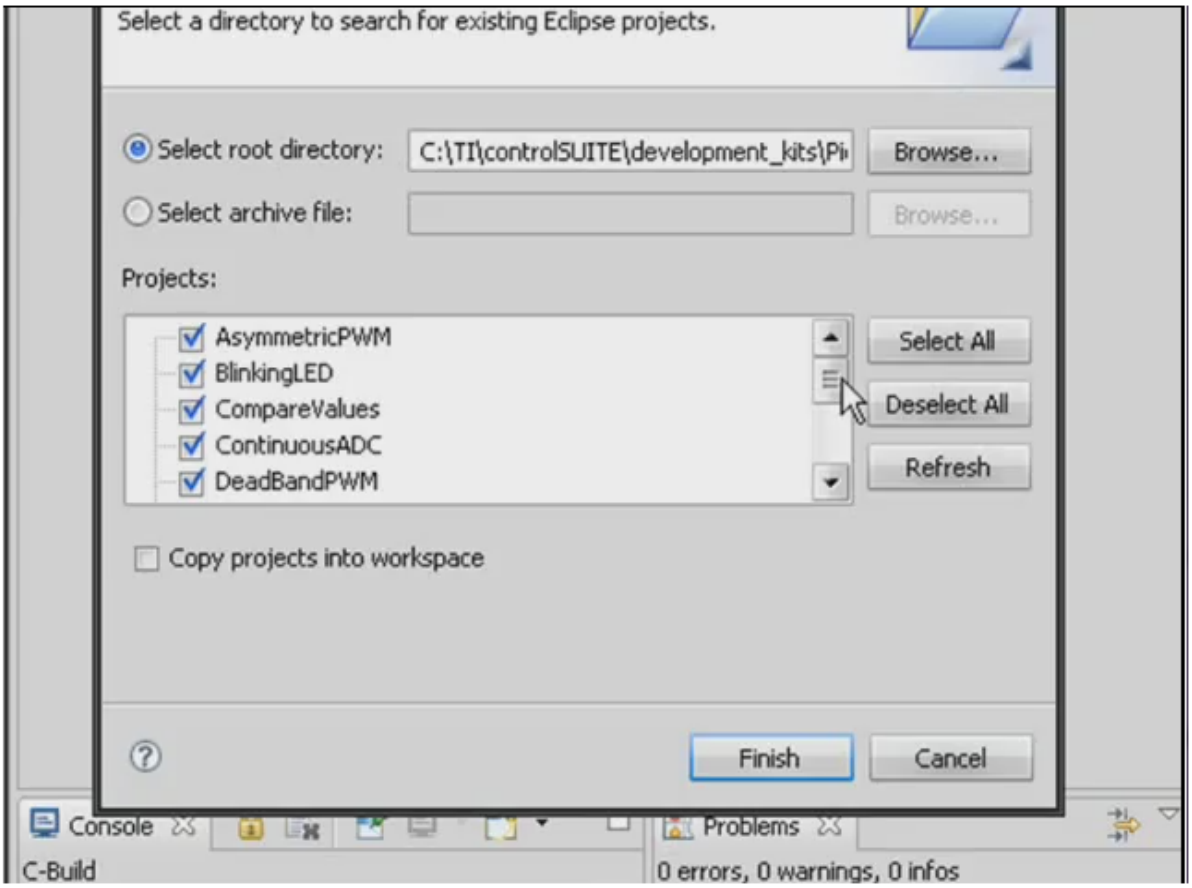


Fig.14 Accés aux différents projets existants

Voici le chemin exact du dossier contenant les exemples de projets :

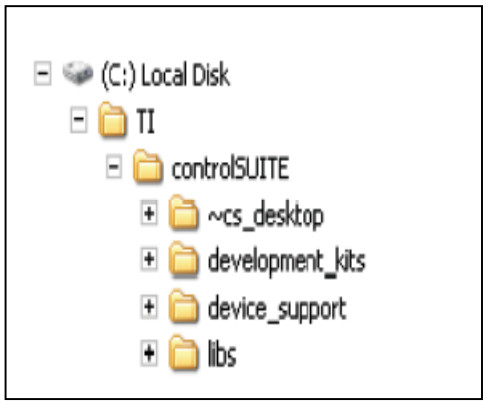
C:\TI\controlSUITE\developement_kits\Piccolo controlSTICK\

II.4 ControlSuite

✓ Définition

Le Controlsuite est le logiciel indispensable pour travailler avec le Piccolo C2000, c'est un ensemble de fichiers bourrés de bibliothèques et des exemples de projets qui viennent compléter le travaille du CCS.

Le Controlsuite comme tout logiciel s'installe dans un répertoire nommé « TI » contenant des sous répertoires suivants :



- **Developement-kits**

  Ce sous répertoire contient tous les exemples de projets

- **Libs**

  Toutes les librairies sont situées dans ce sous répertoire. Cela inclus les bibliothèques de mathématiques et les bibliothèques d'applications et d'utilitaires.

En plus de ces deux sous répertoires, on peut citer :

Le « ~cs_desktop » et le « device_support ».

II.5 Configuration logicielle du Piccolo C2000

Pour effectuer cette configuration, on doit d'abord :

- Dans l'onglet « Target », on sélectionne l'option « New Target configuration… » (voir Fig.15)

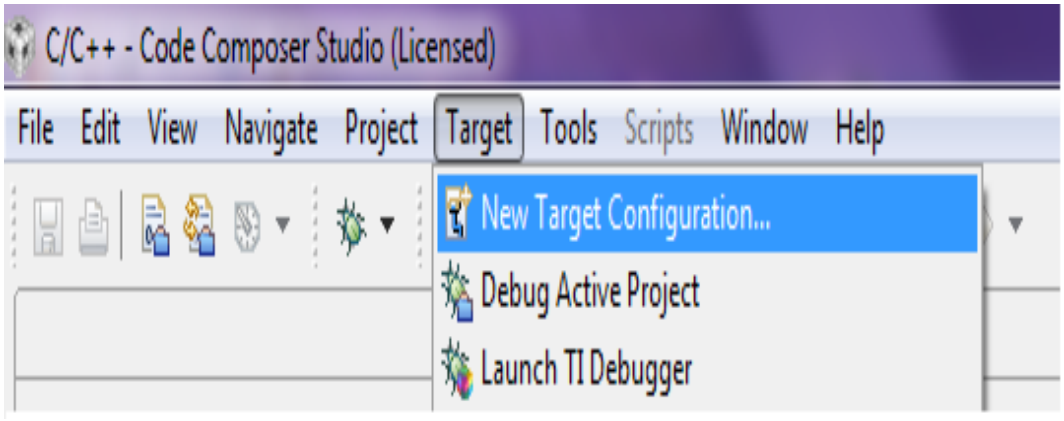


Fig.15 Nouvelle configuration

- La fenêtre suivante(Fig.16) apparait :

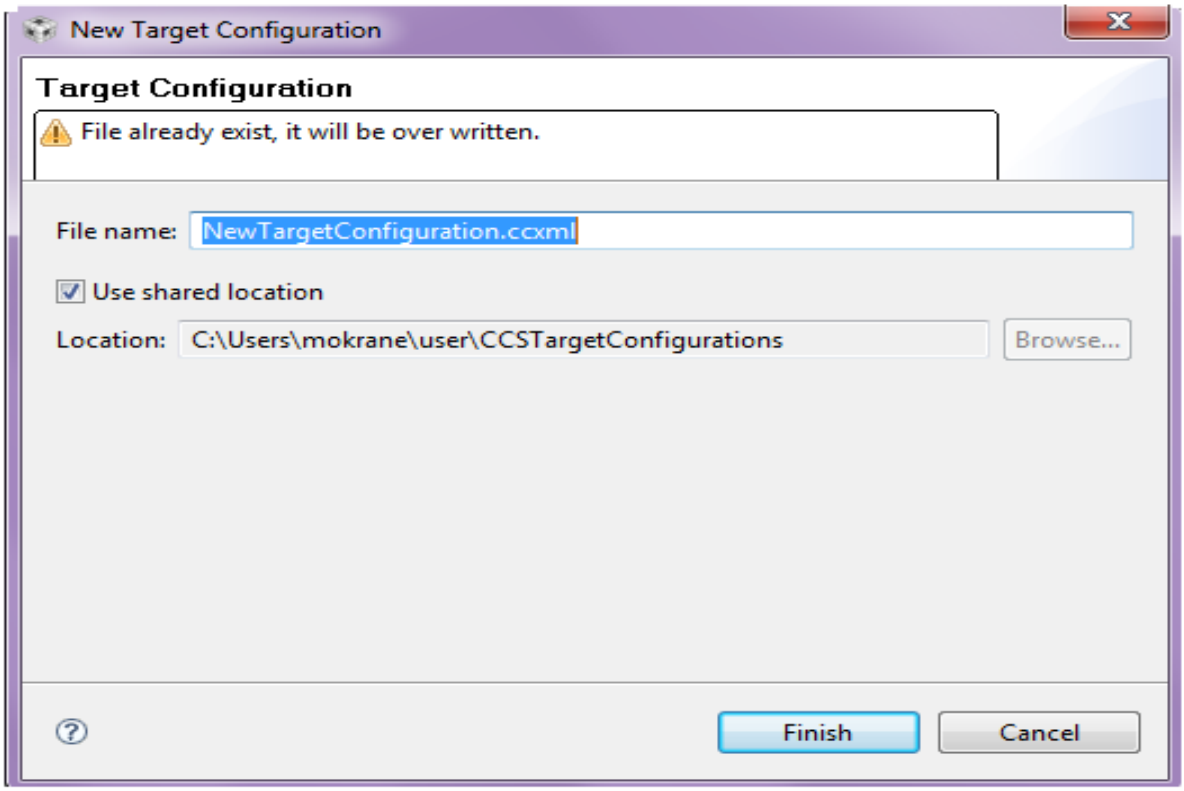


Fig.16 fichier de configuration

Dans la case « File name », on donne un nom pour cette configuration suivi de l'extension ccxml.

Une le nom est donné, on clique sur «Finish ».

Une notre fenêtre doit être apparue (Fig.17), on doit ensuite rentrer les données suivantes :

Connexion : Texas Instruments XDS100v1 USB

Device : TMS320F28027

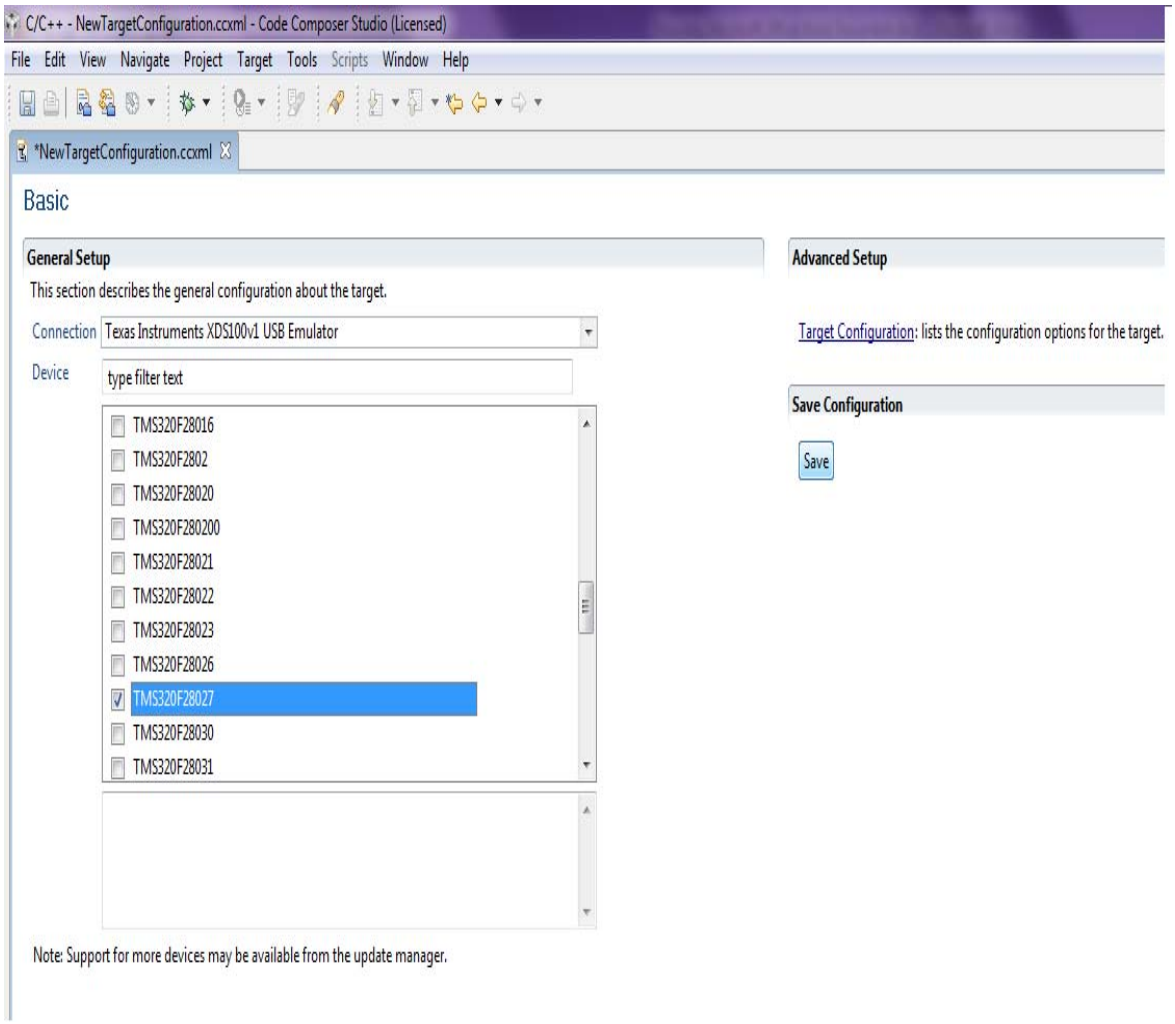


Fig.17 Choix du composant

Ensuite, on clique sur « Save » et on ferme la fenêtre.

II.6 Débogage et exécution d'un programme

II.6.1 Débogage

Avant d'exécuter un programme avec le CCS, il faut d'abord le déboguer. Cette fonction de débogage permet de détecter tous les bugs présentant dans un programme.

Avant de déboguer, il faut choisir le programme à exécuter. Et pour cela, il faut faire apparaître la liste des programmes avec la console « c/c++Project » dans l'onglet « View ». (Voir Fig.18)

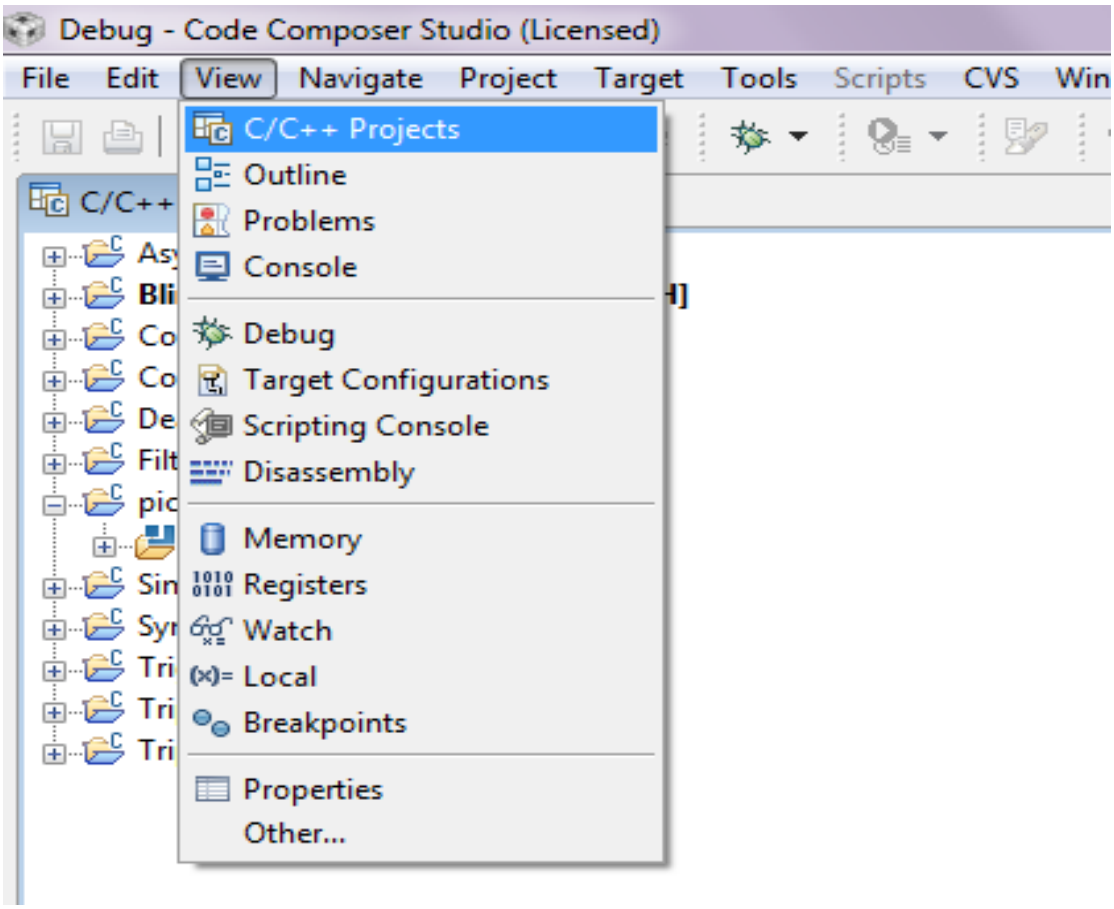


Fig.18 Affichage des différents projets

Une fois la liste des programmes est apparue, il faut ensuite sélectionner le programme à exécuter et avec le bouton droit, on sélectionne l'option « Set as active Project ».(voir Fig.19)

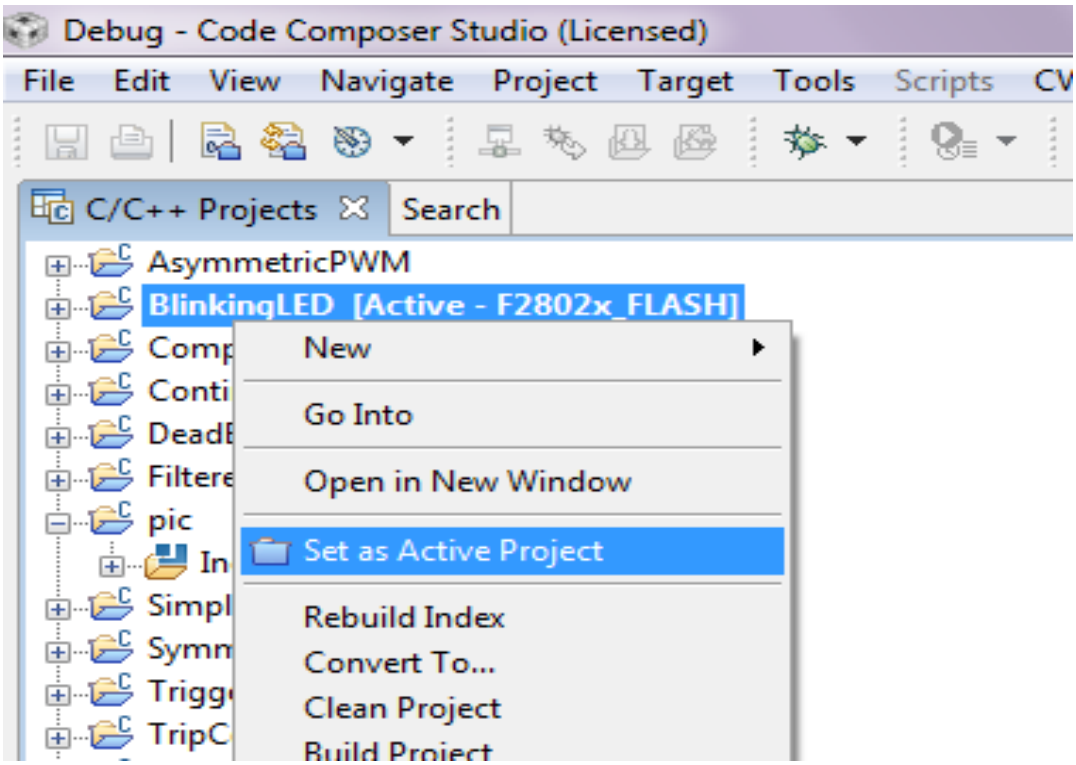


Fig.19 Définir le projet actif

Une fois le projet est activé, il faut maintenant le débuguer, et pour cela, on clique sur « Debug » dans l'onglet « Target ». (voir Fig.20)

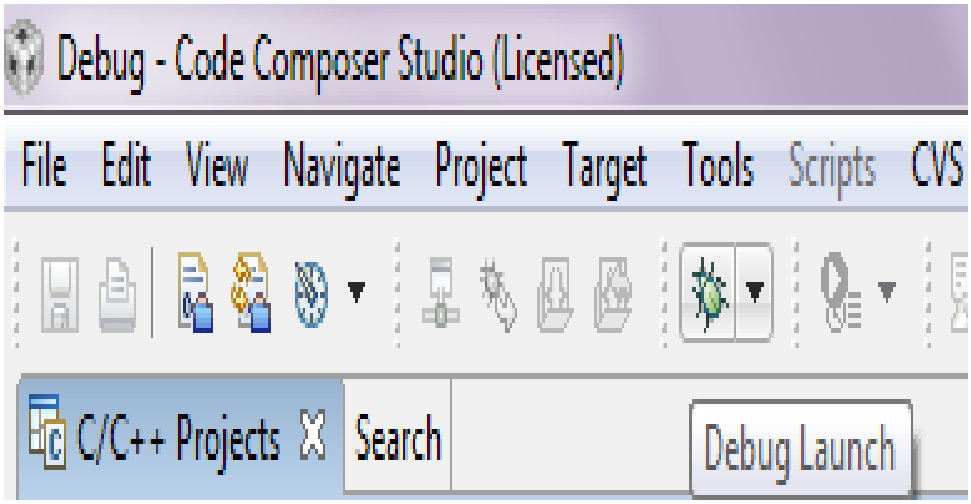


Fig.20 Débogage

II.6.2 Exécution

Une fois le programme est débugué, une fenêtre d'exécution s'ouvrira, contenant le programme en C++ ainsi qu'une barre comportant les options « «Run » et « Stop », donc pour faire marcher notre programme, on sélectionne « Run ». (Voir Fig.21).

Run : mise en marche du programme

Halt : mise en pause du programme.

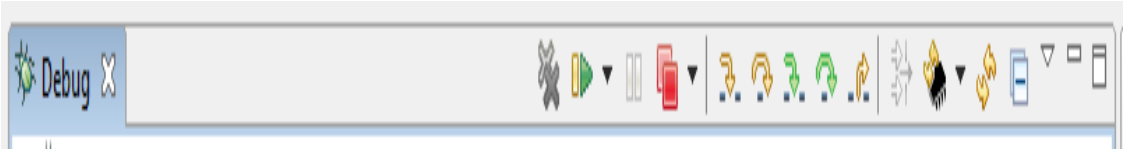


Fig.21 Barre de commande

II.6.3 Exemple d'exécution d'un projet

- **Projet « Blinking led »**

Le projet « blinking led » est le premier projet à exécuter avec le CCS, vu sa simplicité, il nous permet d'assurer la liaison avec le piccolo C2000.

Ce projet permet de faire clignoter une led dans un intervalle de temps constant.

Voici les étapes d'exécution du projet « blinking led »:

- On ouvre le CCS et on branche le Piccolo C2000.
- Dans l'onglet « Project », on clique sur l'option « Import existing CCS/CCE project », ensuite on parcourt le dossier contenant les « Project exemple ». (Voir Fig.22)

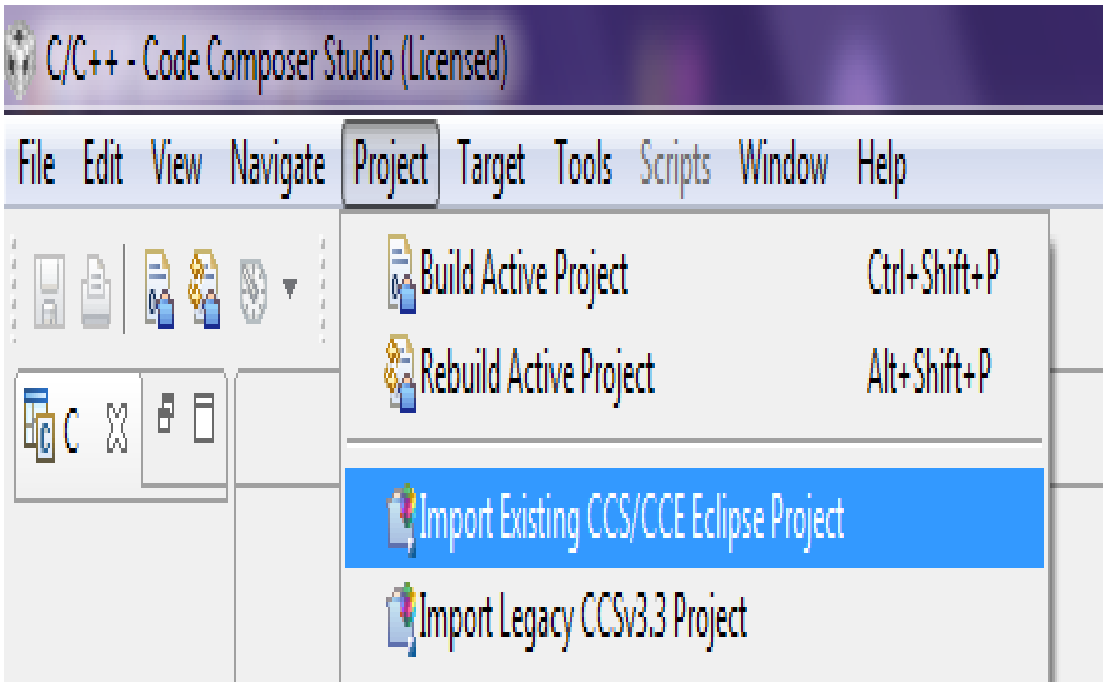


Fig.22 Importer un projet

- La liste des projets sera affichée, contenant le « projet blinking led ».

- On choisira le projet « blinking led » et avec le bouton droit, on clique sur « Set as active project ». (voir Fig.23)

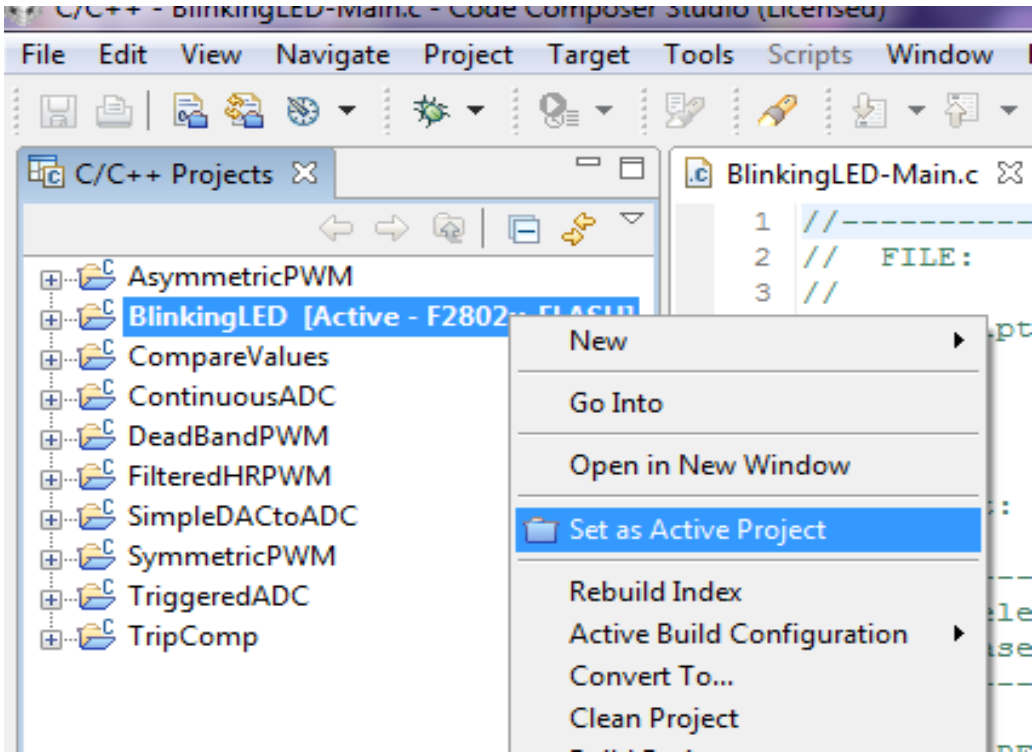


Fig.23 Choix du projet importé

- On débugue le programme avec l'option « Debug » dans l'onglet « Project ».

  Erasing Flash Sectors : permet d'effacer tous les secteurs mémoires réservés pour le programme « Blinking Led ». (Voir Fig.24)

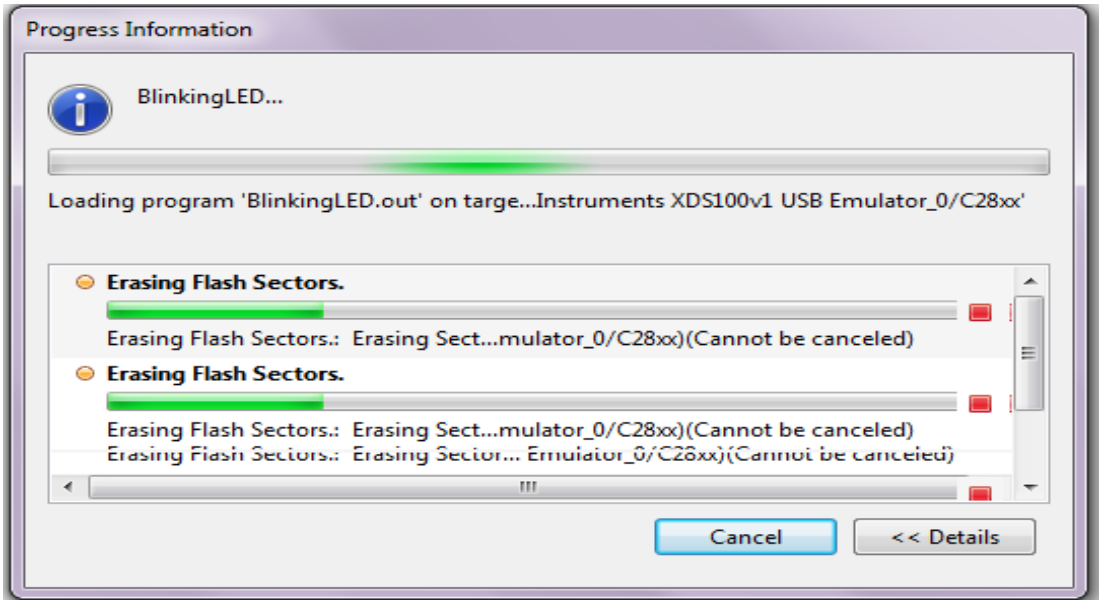


Fig.24 Effacement de la Flash mémoire

Loading : permet de transférer le programme sur la mémoire du Piccolo C2000. (Voir Fig.25)

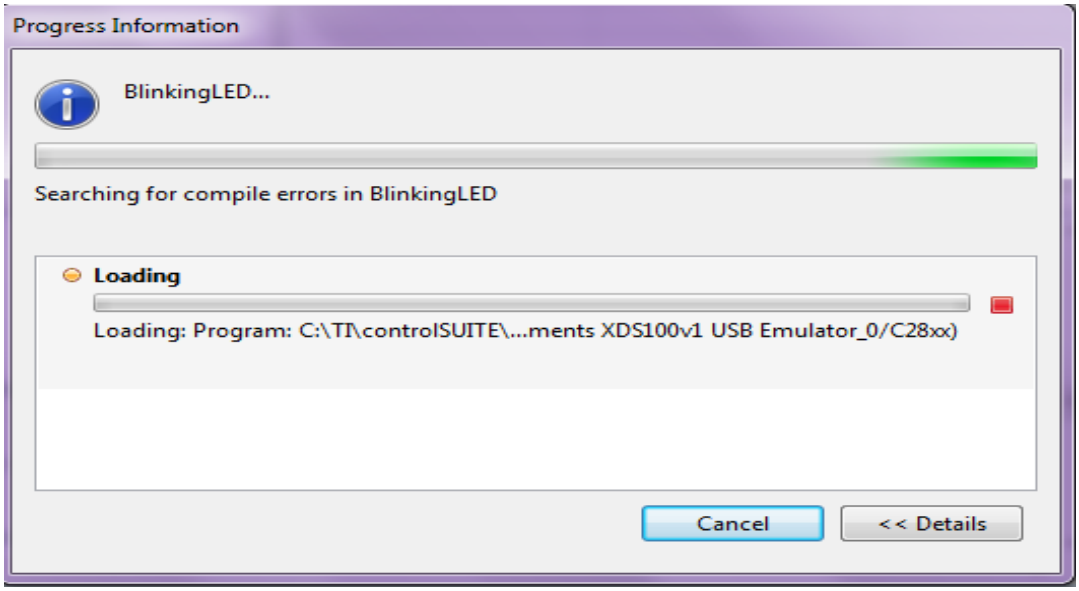


Fig.25 Transfert du code vers le Piccolo C2000

- S'il n'y a pas de problèmes pendant le débogage, notre projet sera prêt à exécuter.

- On exécute notre projet avec le bouton « Run ».

Pendant l'exécution du programme « Blinking Led », la led « LD2 » se met a clignoter pendant un temps constant.

Le programme en C++ correspondant au projet « Blinking Led » est donné dans l'annexe N°[3].

III.1 Introduction

Dans ce chapitre, on s'intéresse à l'acquisition de différents signaux analogiques avec le Piccolo C2000 en utilisant le module ADC (Analog Digital Converter) dont-il défini ci-dessous.

III.2 Le module ADC (CAN)

Le module ADC contient un noyau contenant un convertisseur de 12 bits alimenté par deux circuits échantillonneur-bloqueur (S/H). Ces derniers peuvent être échantillonnés simultanément ou successivement et sont alimentés avec 16 canaux d'entrée analogique.

Il est facile pour l'utilisateur de créer une série de conversions à partir d'un seul déclencheur, le principe de base du fonctionnement est centré autour des configurations des conversions individuelles, appelé SOC (Start Of Conversion).

Le nombre de SOC est égale au nombre de canaux d'entrée analogique qui est de 16, donc on a le SOC0 jusqu'au SOC15.

Chaque SOC peut être configuré indépendamment.

III.2.1 Principe de fonctionnement

Le schéma suivant (Fig.25), montre le principe de fonctionnement de l'ADC avec les SOCs.

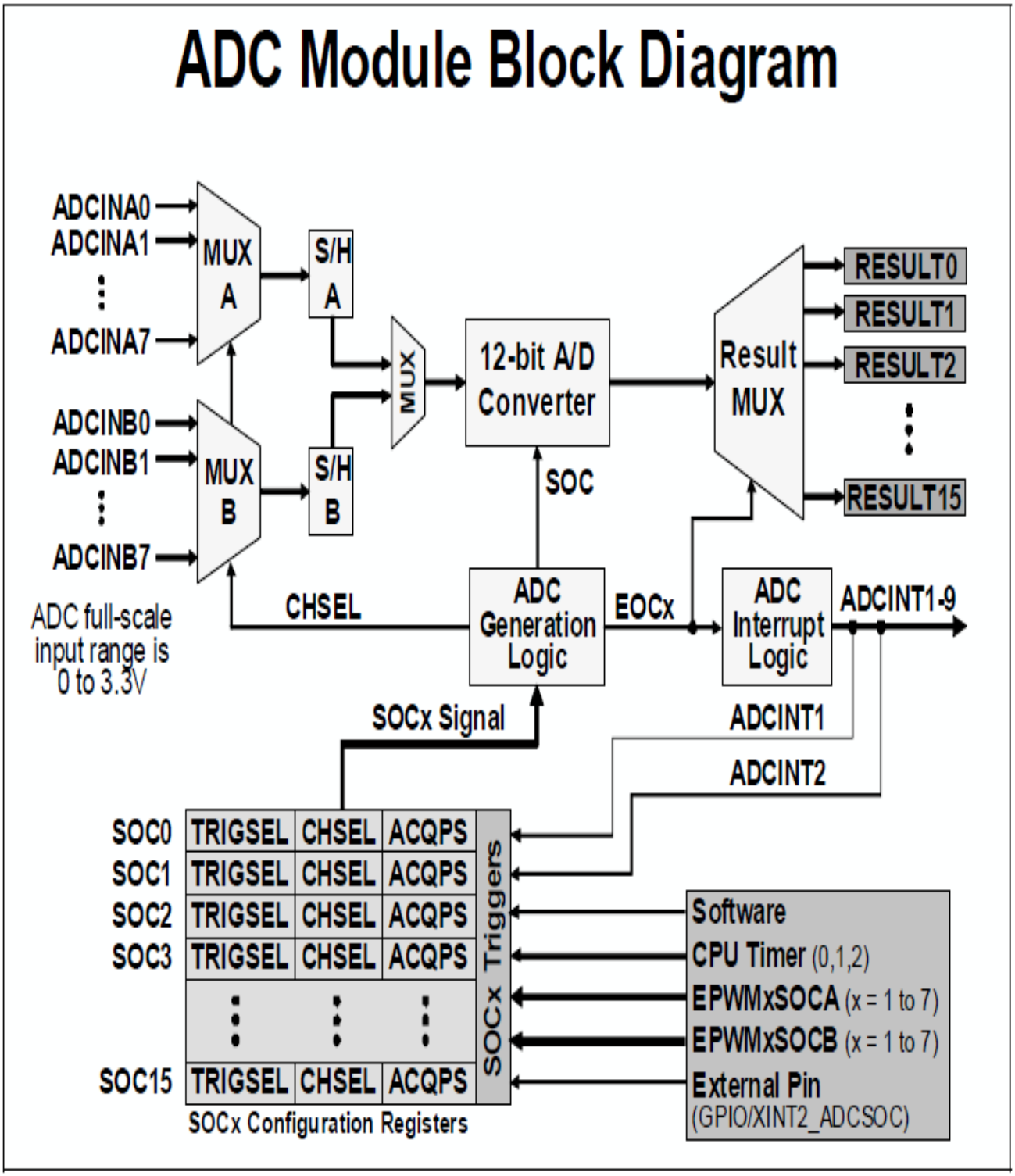


Fig.25 Le module ADC du TMS320F28027

Dans ce type de conversion, on trouve trois configurations. On a donc : la source de déclenchement qui commence la conversion, la chaîne à convertir, et l'acquisition.

On a donc les 16 entrées analogiques configurées comme suit :

Les entrées A0, A1, A2, A3, A4, A5, A6 et A7.

Les entrées B0, B1, B2, B3, B4, B5, B6 et B7.

Les entrées A sont reliées au multiplexeur A et les entrées B sont reliées au B.

Ces deux multiplexeurs sont reliés successivement aux échantillonneurs bloqueurs A et B qui sont eux même multiplexés.

La sortie est reliée au « 12-bit A/D Converter».

Les registres CHSEL et ADCSOCxCTL définissent les canaux a convertir.

Si on choisi de convertir le canal 1 c'est-à-dire le SOC0, le CHSEL va sélectionner le canal en envoyant un signal a l' « ADC Generator Logic », ce dernier va interrompre toutes les autres conversions en générant le signal EOC (End Of Conversion).

Le module « ADC Converter » va donc convertir le canal et le résultat sera engendrer par le « Result Mux ».

Des signaux d'interruptions générés par l'ADC Interrupt Logic vont apparaitre et stopper toute sélection d'autres canaux.

III.2.2Cycle SOC

Voici le schéma ci-contre montrant un exemple de cycle de conversion de différents canaux :

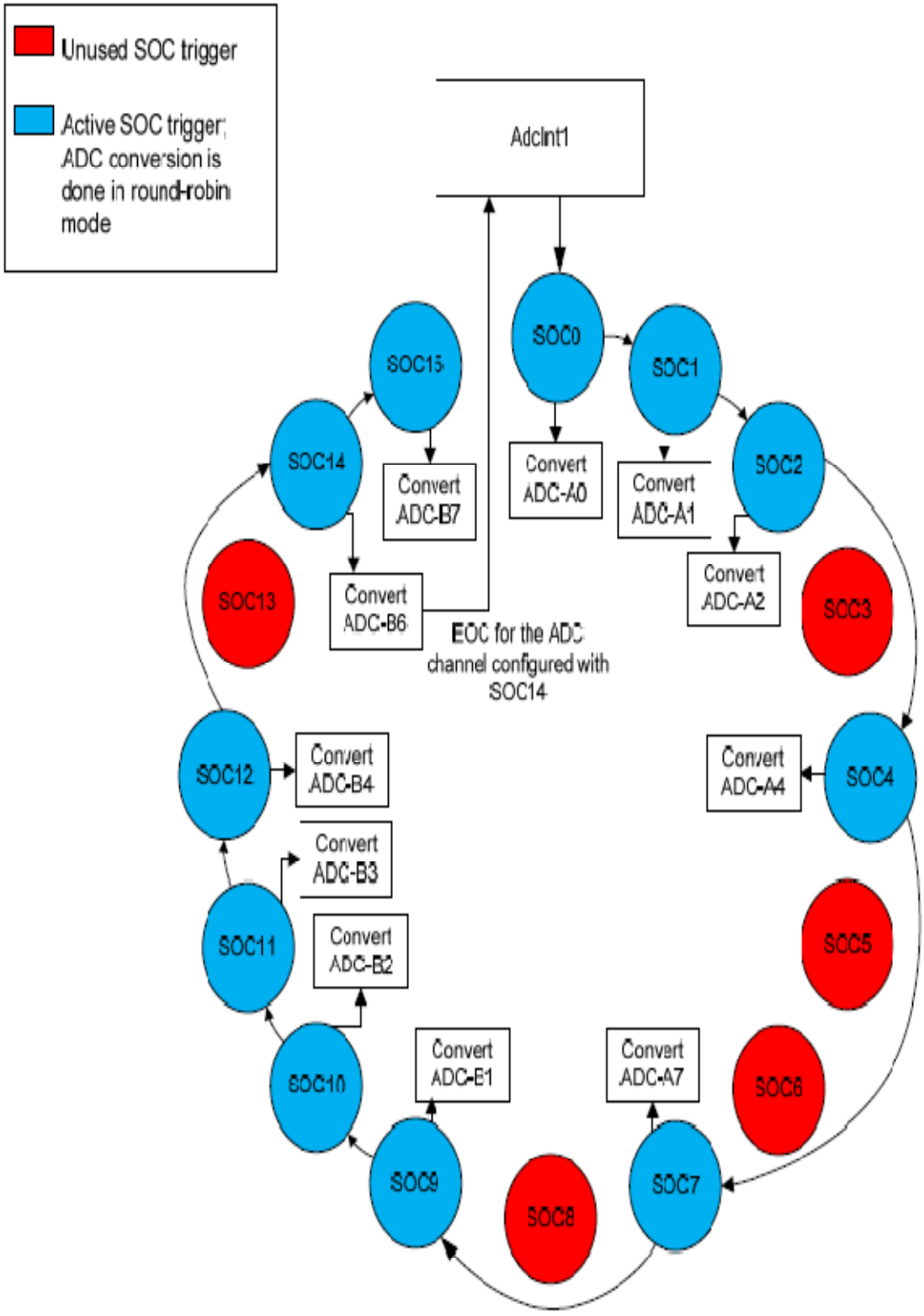


Fig.26 Exemple de cycle de conversion

Dans cette conversion, le canal ADC14 (ADC-B6) provoquera une EOC avec la génération d'un signal d'interruption « ADC Interrupt1 », celui-ci est aussi configuré pour générer une série de SOC en utilisant le registre ADCINTSOCSELx.

Donc chaque canal peut être converti avec les commande CHSEL et ADCSOCxCTL, le champ CHSEL attend une valeur de 0 à 15, donc ADC A0 jusqu'à ADC A7 et ADC B1 jusqu'à ADC B7.

Si le champ ADCINTSOCSELx est fixé à 0, donc il n'aura pas d'interruption qui provoquera un SOC.

Voici le code en c++ montrant les différents registres de cette conversion :

```
AdcRegs.ADCINTSOCSEL1.bit.SOC0 = 1; // ADCInterrupt 1 provoque SOC0
.
.
.
AdcRegs.ADCINTSOCSEL2.bit.SOC15 = 1;
AdcRegs.ADCSOC0CTL.bit.CHSEL = 0; //convertir ADCA0(CH0) quand SOC0 est
reçu
AdcRegs.ADCSOC1CTL.bit.CHSEL = 1; // convertir ADCA1(CH1) quand SOC1
est reçu
AdcRegs.ADCSOC2CTL.bit.CHSEL = 2;
AdcRegs.ADCSOC4CTL.bit.CHSEL = 4;
AdcRegs.ADCSOC7CTL.bit.CHSEL = 7;
AdcRegs.ADCSOC9CTL.bit.CHSEL = 9; //convertir ADCB1(CH9) quand SOC9 est
reçu

AdcRegs.ADCSOC10CTL.bit.CHSEL = 10;
AdcRegs.ADCSOC11CTL.bit.CHSEL = 11;
AdcRegs.ADCSOC12CTL.bit.CHSEL = 12;
AdcRegs.ADCSOC14CTL.bit.CHSEL = 14;
AdcRegs.ADCSOC15CTL.bit.CHSEL = 15;
```

III.3 Test d'acquisition pour le Piccolo C2000

Cette application a pour objectif de tester l'acquisition analogique du Piccolo C2000, pour se faire, nous avons procédé selon le schéma de la figure Fig.27. Pour générer les différents signaux, nous avons utilisé un GBF(Générateur Basses Fréquences). Afin d'étalonner les signaux à l'entrée, nous avons visualisé les différents signaux par un oscilloscope en ajustant le niveau et la fréquence sur le GBF.

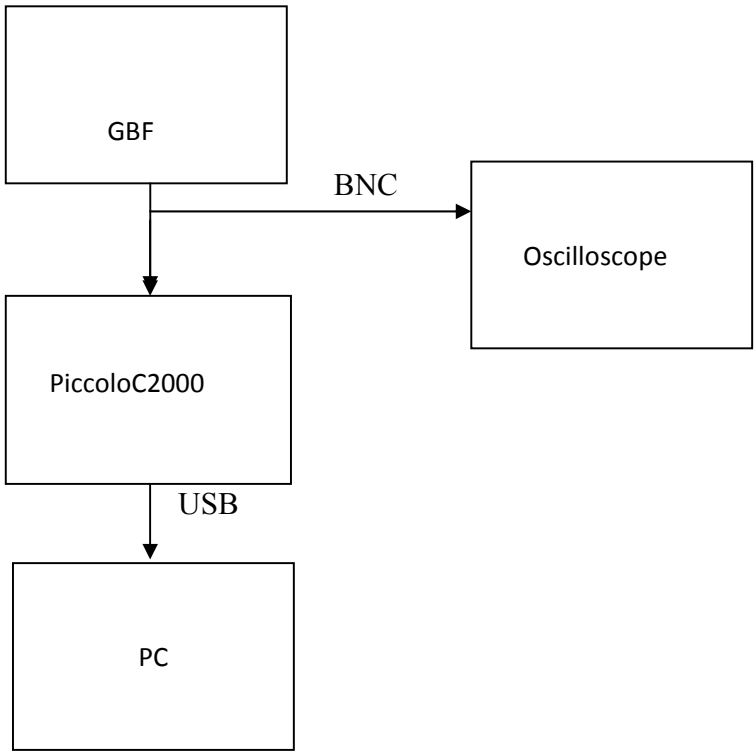


Fig.27 Schéma du montage de tests d'acquisition.

III.3.1 Programme d'acquisition avec le CCS

Le programme est écrit en C++ avec le CCS voir Annexe N°3. Voici ci-contre l'organigramme du programme principal :

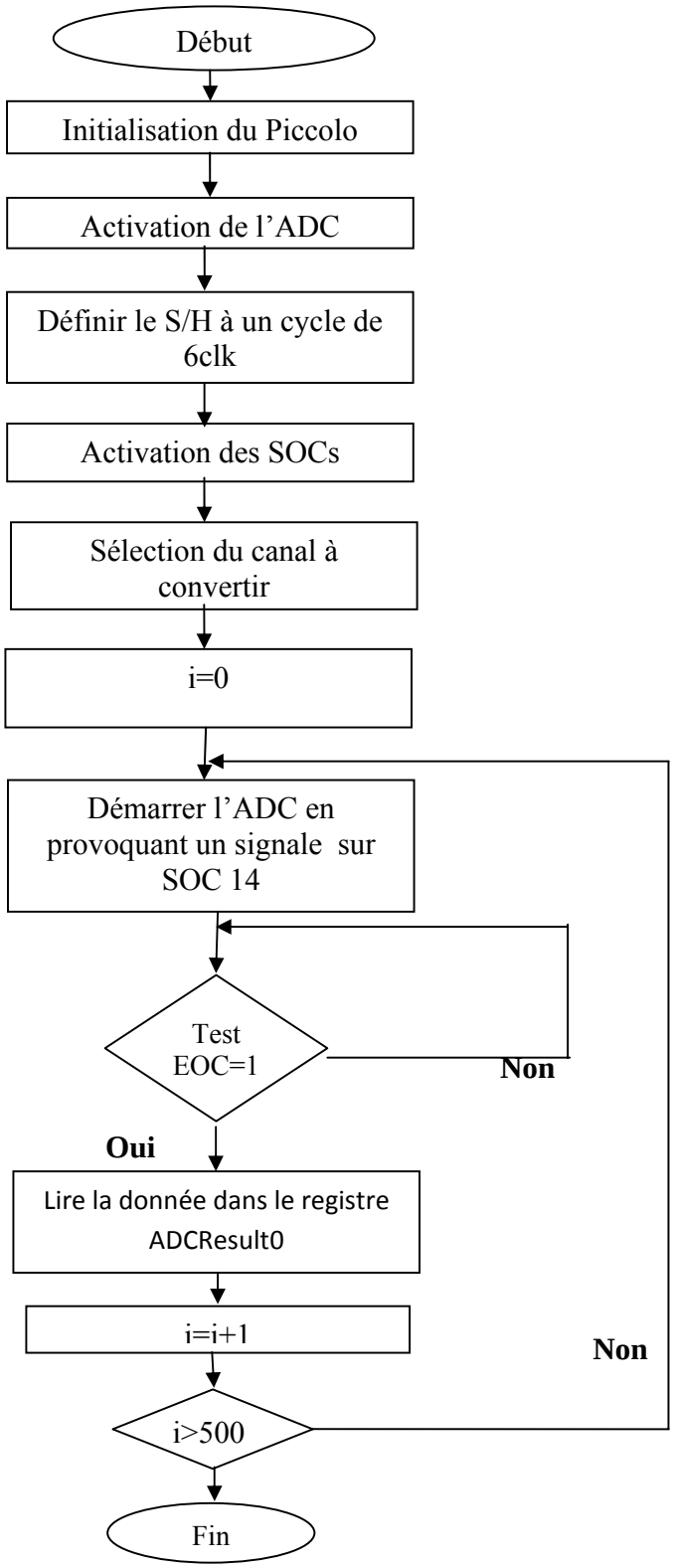


Fig.28 Organigramme du programme principal

III.3.2 Acquisition d'un signal carré

Le signal carré est signal tout ou rien d'amplitude A et de période T avec un rapport cyclique de 50%.

A = 1.5V.

T = 1ms.

La fréquence est : F = 1/T = 100Hz.

Dans ce cas la fréquence d'échantillonnage est prise Fe = 1KHz, pour satisfaire le théorème de Shannon $Fe \geq 2Fmax$.

La figure Fig.29 montre le résultat d'acquisition du signale carré par le Piccolo C2000 sur 260 échantillons.

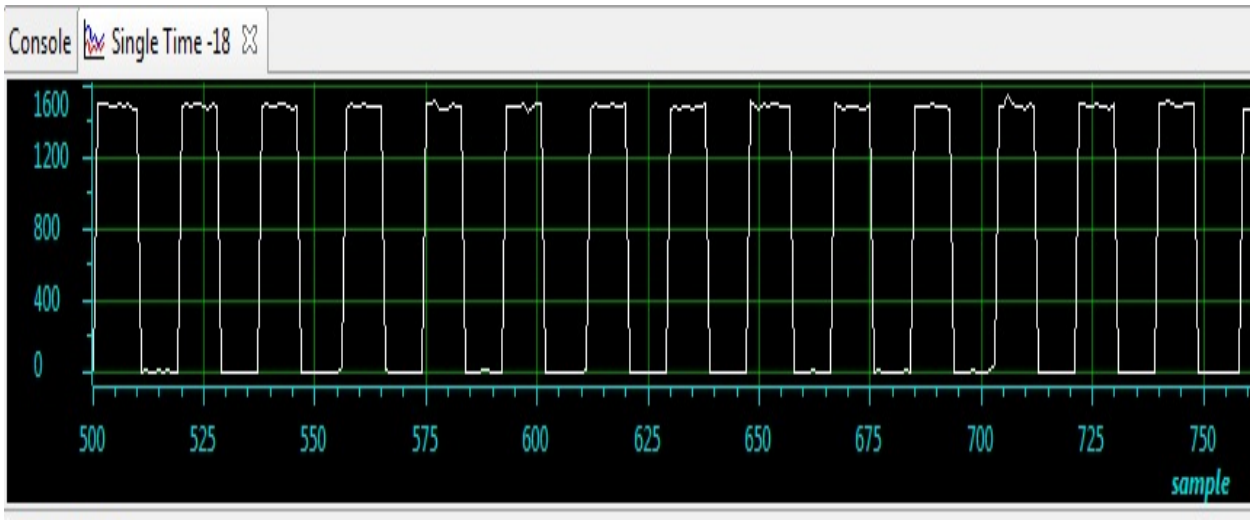


Fig.29 acquisition d'un signale carré le Piccolo C2000

III.3.3 Acquisition signal sinusoïdal

Le signale à acquérir est un signale sinusoïdal sous forme s(t) =C+A*sin (2πft), d'amplitude A et de période T

C=A=1.5/2 V.

T = 1ms.

La fréquence est: F = 1/T = 100Hz.

La fréquence d'échantillonnage est : Fe=1KHz.

La figure Fig.30 montre le résultat d'acquisition du signale sinusoïdal par le Piccolo C2000 sur 200 échantillons.

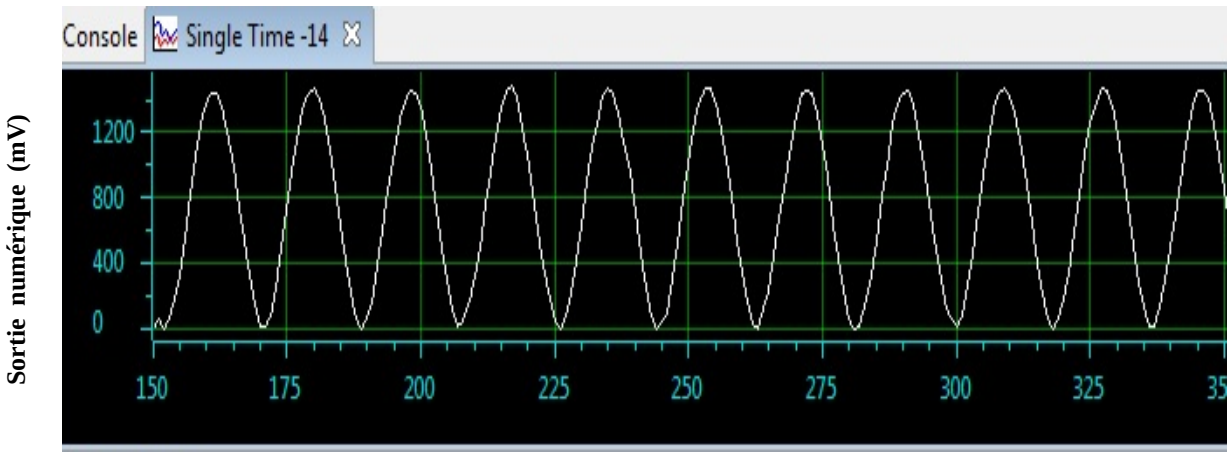


Fig.30 Signale sinusoïdal

III.3.4 Acquisition d'un signal triangulaire

Le signale à acquérir est un signale triangulaire d'amplitude A et de période T.

A =1.5V.

T = 1ms

F=1/T=100 Hz.

La fréquence d'échantillonnage est : Fe=1KHz.

La figure Fig.31 montre le résultat d'acquisition du signale triangulaire par le Piccolo C2000 sur 260 échantillons.

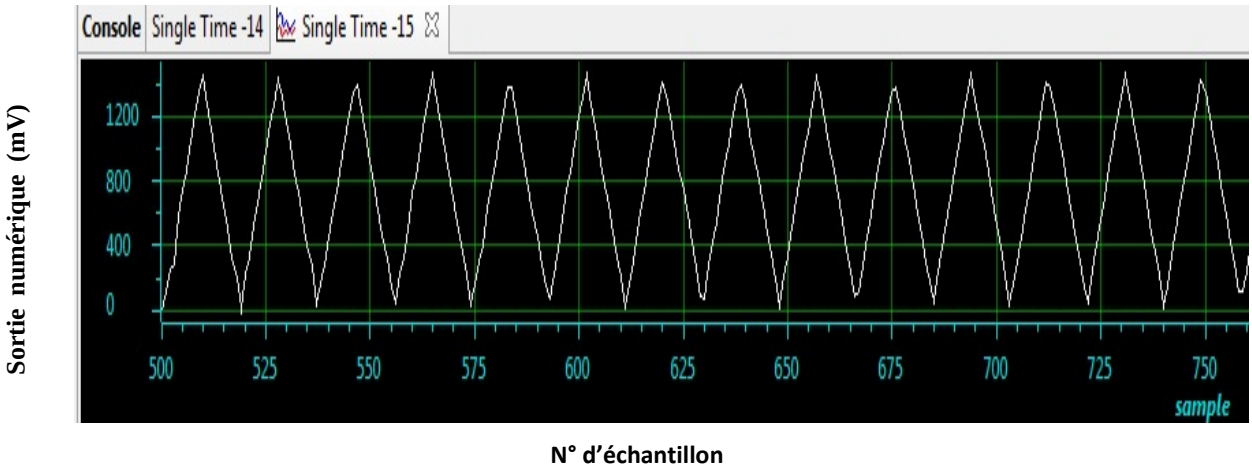


Fig.31 Signale triangulaire

III.3.5 Acquisition d'un Signal modulé en fréquence(FSK)

Le signal à acquérir est un signale modulé en fréquence sous forme
$s(t)= C+A* \sin (2\pi*f(t)*t)$ tel que :

C = A = 1.5/2V

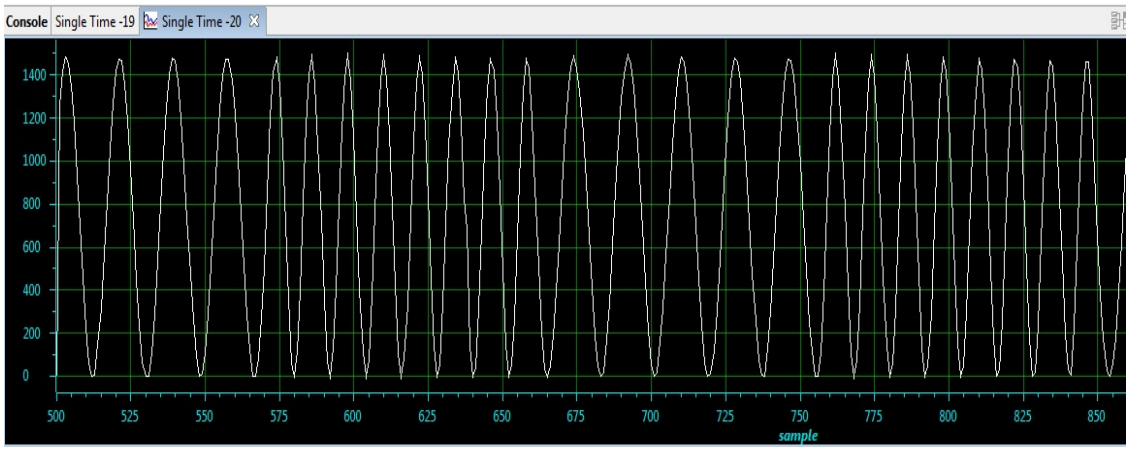
F1= 200 Hz

F2=5/8*F1=125 Hz

La fréquence d'échantillonnage est : Fe = 1KHz.

La figure Fig.32 montre le résultat d'acquisition du signale triangulaire par le Piccolo C2000 sur 360 échantillons.

F1(Hz) **F2(Hz)**



Sortie numérique (mV)

N° d'échantillon

Fig.32 Signale modulé en fréquence

IV.1 Introduction

Un accéléromètre est un composant de plus en plus courant des objets quotidiens : stabilisateurs d'image de caméscope numérique grand public, manette de console de jeux vidéo (Nintendo, Wii,…etc).

On peut aussi les utiliser dans les robots pour détecter des chocs et mesurer un déplacement.

Pour la réalisation de cet accéléromètre, nous avons utilisé le capteur d'accélération ADXL311.

Voici ci-dessous le schéma synoptique de l'accéléromètre.

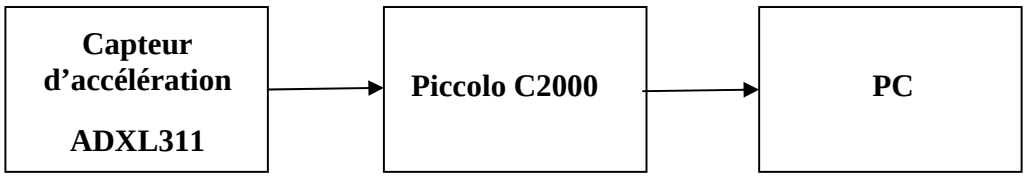


Fig.33 Schéma synoptique de l'accéléromètre

IV.2 Etude du capteur d'accélération ADXL311 :

Le capteur ADXL311 prend la forme d'un petit bout de Silicium de quelques millimètres de large. On utilise une puce électronique fixée à un circuit de test.

Cette puce utilise une technologie qui permet de faire des mesures grâce à un microsystème électromécanique ou MEMS (Voir l'annexe N°[4]) qui reproduit à l'échelle électronique des réactions mécaniques. Grâce à ces MEMS beaucoup de

machines ont vu leur tailles réduire, comme pour le cas du gyroscope, un dispositif qui faisait une dizaine de centimètres, remplacé par un MEMS d'à peine 5mm de large et à un prix dérisoire.

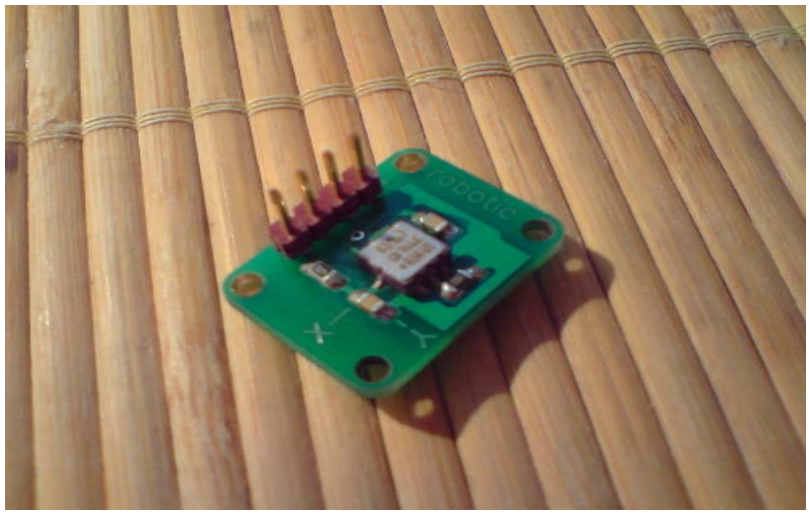


Fig.34 Vue de dessous de l'ADXL311

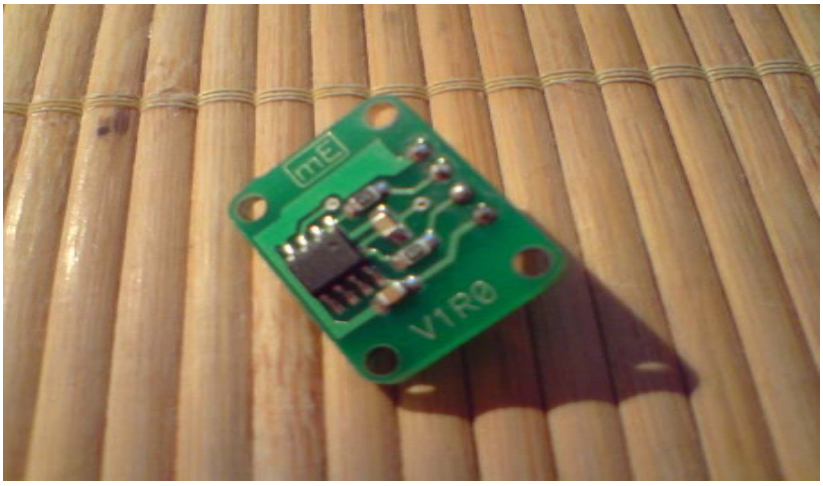


Fig.35 Vue de dessous de l'ADXL311

IV.3 Description générale de l'ADXL311

L'ADXL311 est un capteur d'accélération à faible coût et faible puissance. Il est composé de deux axes avec sortie de tension conditionnée.

Les deux axes X et Y comportent successivement des accélérations horizontales et des accélérations verticales.

Sa fiabilité est démontrée par 1 échec sur 1 milliard d'heures d'exploitation.

L'ADXL311 mesure une accélération avec une plage de +/- 2g. Il peut mesurer l'accélération dynamique (les vibrations) et l'accélération statique (la gravité).

L'utilisateur sélectionne la bande passante du capteur en utilisant les capacités Cx et Cy sur les broches XFILT et YFILT. Des bandes passantes varient de 1Hz à 5Khz selon la capacité utilisée. Voir le tableau ci-dessous Tab.4.

Tab.4 Les différentes bandes passantes de l'ADXL311

Bande passante	Capacité (µF)
10 Hz	0.47
50 Hz	0.10
100 Hz	0.05
200 Hz	0.027
500 Hz	0.01
5 kHz	0.001

Remarque :

La fréquence d'échantillonnage est fixée à Fe=1Khz, la bande passante est de F=500Hz, donc on a utilisé des capacités de 0.01 µF.

IV.4 Caractéristique de performance de l'ADXL311

La caractéristique suivante (Fig.35) représente la variation de la sensibilité de l'ADXL311 (en %) en fonction de la température (en C°).

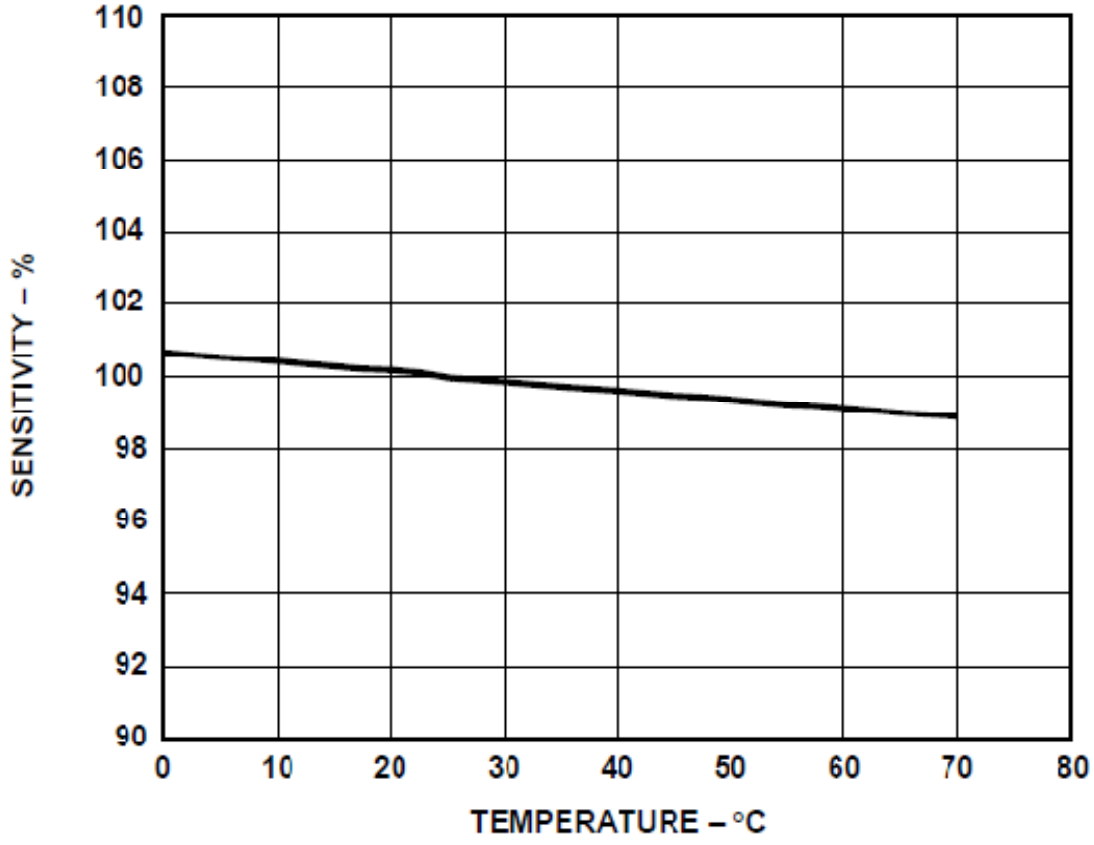


Fig.36 Caractéristique de la sensibilité en fonction de la température

On constate bien que la sensibilité décroit à chaque augmentation de la température.

IV.5 Schéma de conditionnement de l'ADXL311

Pour l'acquisition du signal de l'accélération, on a besoin d'un lab d'essai, une source de tension, des condensateurs et des résistances comme le montre le schéma de conditionnement suivant :

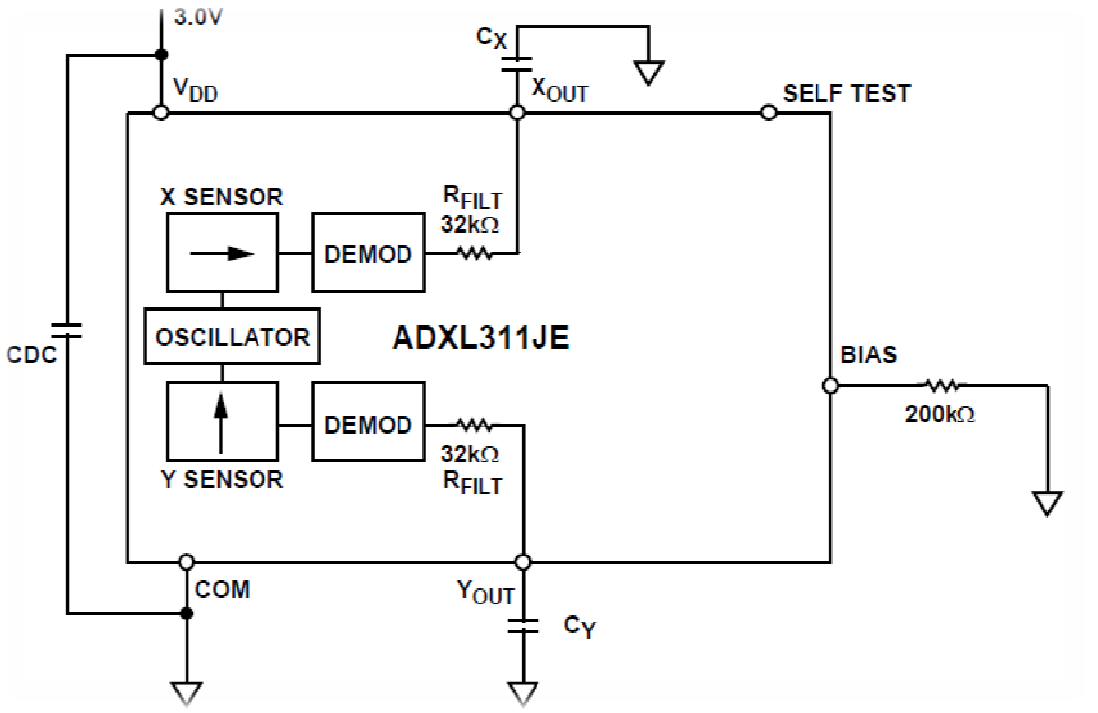


Fig.37 Schéma de conditionnement de l'ADXL311

On utilise 3 résistances dont deux avec 32kΩ et une autre avec 200kΩ.

Des capacités de 0.01 µF.

La tension injectée est <=3V.

Une fois le montage est terminé, on prépare la partie logicielle qui est le programme d'acquisition.

IV.6 Programme d'acquisition

Pour l'acquisition du signal d'accélération, on utilise l'entrée 14 avec le programme « ContinousADC » précédent (voir Annexe N°3) dont voici l'organigramme :

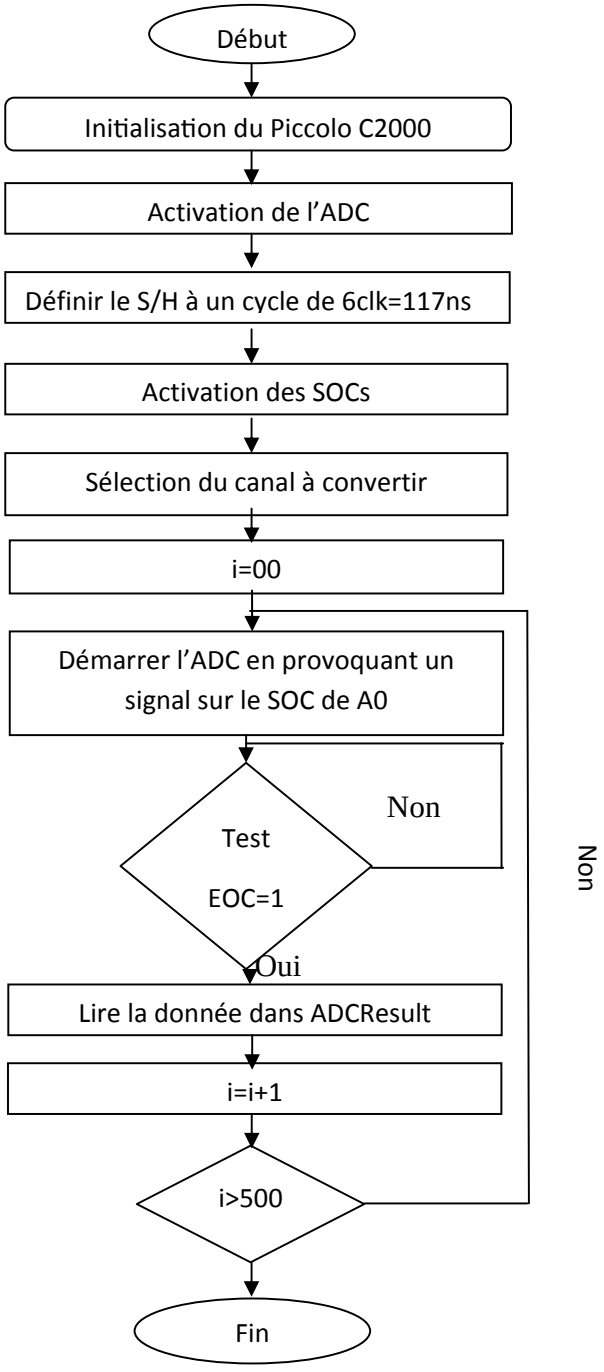
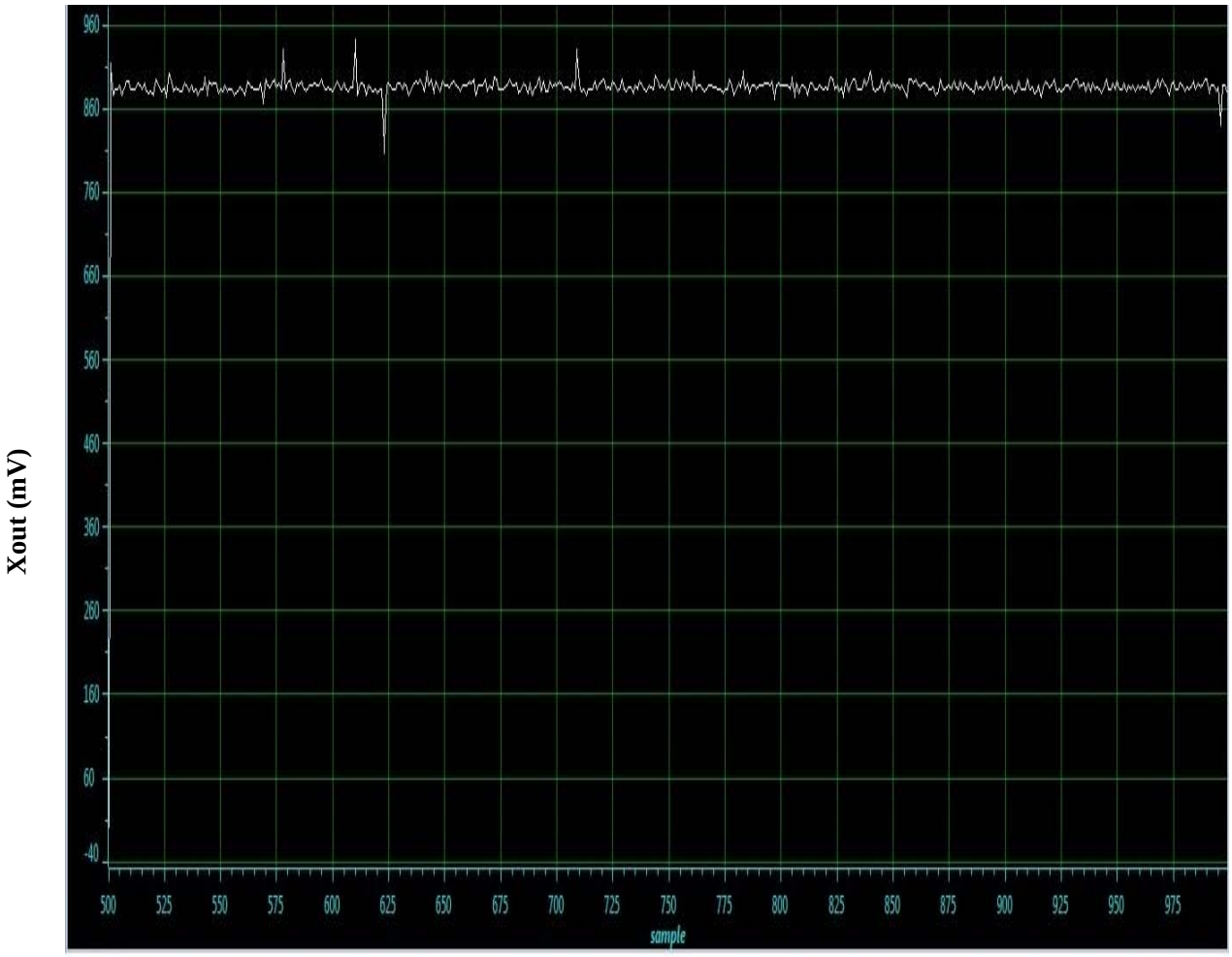


Fig.38 Organigramme du programme d'acquisition

IV.7 Visualisation des signaux d'accélérations

Après avoir débugué et exécuté le programme avec le code composer studio, on secoue légèrement le lab d'essai horizontalement puis verticalement pour avoir les différentes accélérations sur l'axe X et sur l'axe Y.

Sur l'axe X, on aura le graphe suivant :



N° d'échantillon

Fig.39 Accélérations sur l'axe X

Verticalement sur l'axe Y, on aura le graphe suivant :

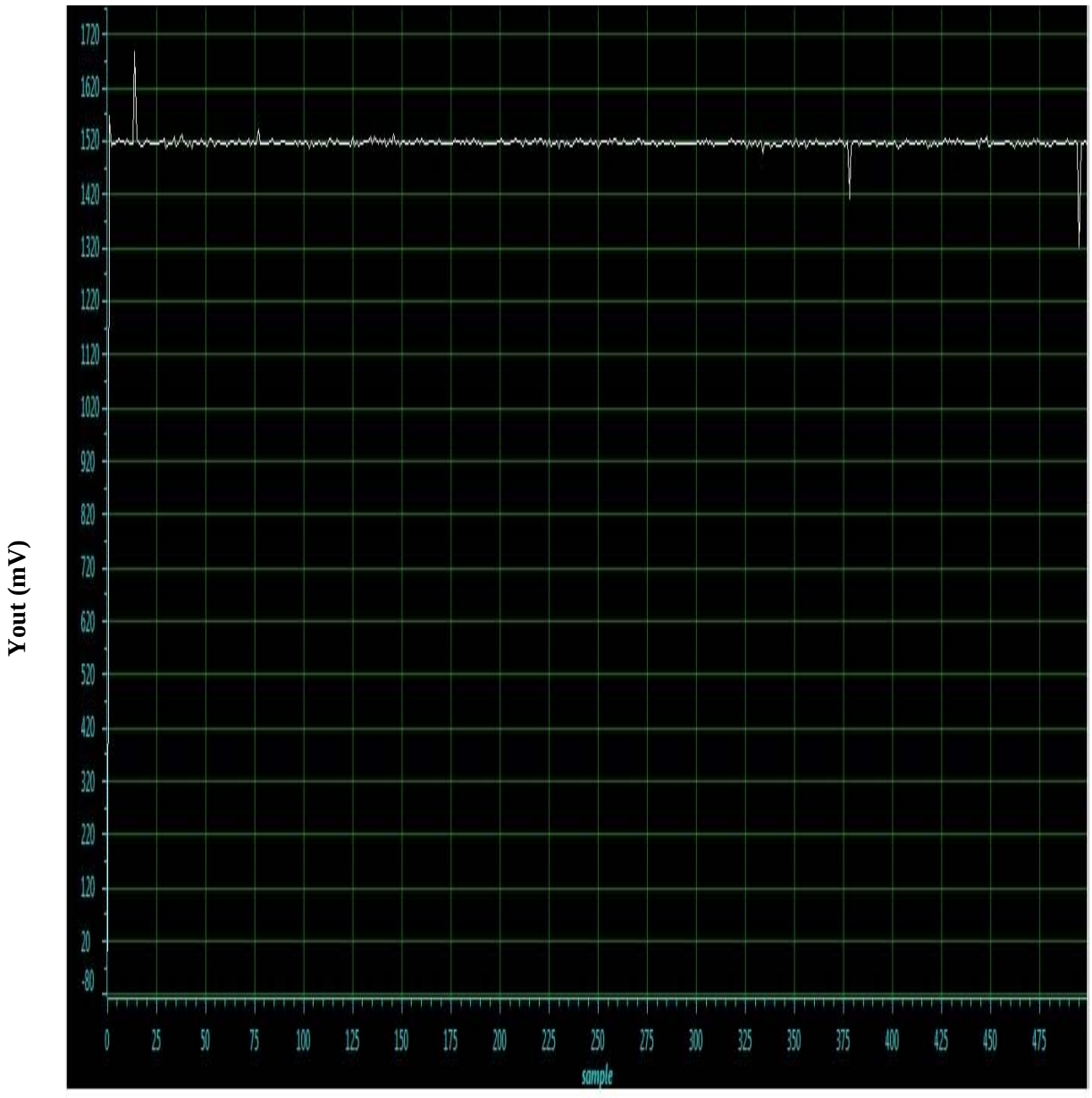


Fig.40 Accélérations sur l'axe Y

IV.8 Interprétation des résultats des deux graphes

- **Interprétation des résultats du graphe X :**

On constate l'apparition de plusieurs pics sur l'axe X correspondants aux différentes accélérations apparues pendant le secouement de l'accéléromètre.

D'après les caractéristiques de l'ADXL311, à chaque 167 mV correspond une accélération de 1g (1g=9.8m/ s^2)

Si on prend le pic d'accélération qui est de 70mv/g, on a la valeur de l'accélération Ax1 suivante :

$$Ax1 = \frac{70*1}{167} = 0.42 \text{ m/}s^2.$$

Sur le même axe, on constate une décélération Ax2 dont la valeur est :

$$Ax2 = -\frac{85*1}{167} = -0.51 \text{ m/}s^2.$$

- **Interprétation des résultats du graphe Y :**

Les pics d'accélérations sur cet axe sont plus importants par rapport à l'axe X, la différence est due à la façon de secouer l'accéléromètre.

Le premier pic d'accélération est de 180 mV/g ce qui donne une valeur Ay1 suivante :

$$Ay1 = \frac{180*1}{167} = 1.08 \text{ m/}s^2.$$

Sur le même axe, on a une décélération Ay2 dont la valeur est :

$$Ay2 = -\frac{200*1}{167} = \text{- } 1.2 \text{ m/}s^2.$$

Conclusion Générale

Le dispositif Piccolo C2000 fait partie des systèmes intelligents de Texas Instruments (TI), peut être exploité dans les systèmes de commande, d'acquisition et traitement de signal (DSP) en temps réel. Sa taille miniature et son interface USB lui permettent une intégration facile dans un système embarqué. Par ailleurs, son horloge rapide qui atteint 60MHz lui permet d'avoir une performance de vitesse d'exécution très importante. Cette performance est renforcée par son architecture conçue autour de l'architecture Harvard. A cause de cette architecture la mémoire flash du Piccolo n'est pas très optimale, toutefois on dispose de 128ko pour la mémoire programme et données.

Pour faciliter le développement des applications à base du Piccolo C2000, Texas Instruments met à la disposition des utilisateurs l'interface PC « Code Composer Studio 4.0» (CCS). Cette interface permet le développement de programmes en Assembleur, C et C++. Pour des raisons de simplicité et du temps, nous avons préféré travailler en C++. Pour une meilleure maitrise du composant Piccolo C2000 nous avons élaboré un ensemble de tests. Le premier est simplement un test de clignotement de la LED du Piccolo C2000 afin de tester à la fois le CCS et la connexion PC-Piccolo pour vérifier du bon fonctionnement. Une fois cette étape est réalisée nous avons entamé d'autres tests sur le module d'acquisition des signaux analogiques. A l'aide d'un GBF, nous avons fait l'acquisition de plusieurs signaux de basse fréquence, un signal carré, rectangulaire, sinusoïdal et un signal modulé en fréquence (FSK).

Comme application pour notre projet nous avons choisi de réaliser un accéléromètre Numérique à base du Piccolo C2000 et d'un capteur d'accélération ADXL311. Dans cette application nous avons réalisé le schéma de conditionnement du capteur ainsi que le programme d'acquisition du Piccolo. Le dispositif fonctionne sur deux axes X et Y. des

signaux d’accélération sur les deux axes son acquis et enregistrés. La plage de mesure atteint ± 2g.

Néanmoins, comme tout projet, le notre présente quelques insuffisances qui peuvent être sans doute comblées. En effet, dans l’application en temps réel, on n’a pas pu intégrer cette option dans notre programme d’acquisition en vu de manque de logiciels adéquats.

Enfin, nous souhaitons que ce travaille suscrite un intérêt auprès des étudiants et servira de référence pour ceux qui s’intéresseraient à ce domaine et les inciteraient à apporter d’autres améliorations.

Bibliographie

[1] **Christian Tavernier**, 1991 : Les microcontrôleurs PIC description et mise en œuvre. Dunod

[2] **Christian Tavernier**, 2011 : Application des microcontrôleurs PIC. Des PIC10 aux PIC18, 4^{em} édition. Dunod.

[3] **Pascal Mayeux**, 2005 : Apprendre la programmation des pics par l'expérimentation et la simulation, 3ème édition. ETSF EDITION TECNHIQUEET SCIENTIFIQUE FRANÇAISE.

[4] **M^{elle} Challali Fatiha, M^{elle} Karaoui Fazia**. Etude et réalisation d'une carte d'acquisition et de transmission des données à base du pic 16F877. Thèse d'ingénieur département d'électronique UMMTO promotion 2004.

[5] **Houssemddine Ghadacha**, Conception et vérification d'un watchdog STM75. Thèse d'ingénieur de l'école supérieure des communications de Tunis promotion 2006/2007.

[6] **Texas Instruments**. C2000^{TM} Piccolo^{TM} 1-day workshop guide and lab manual.

[7] **Texas Instruments**, SPRS523F-November 2008 – revised December 2010.

[8] **Texas Instruments**, Application Report SPRAAU8-March 2008.

[9] **Texas Instruments**, TMS320C28x CPU and Instruction set Reference Guide.

[10] Sites internet:

- www .TI.com
- http:/ forum.futura-sciences.com/électronique/55677-accelerometre-pic.html.
- http:/ processors-wiki-TI.com/index.php/category:code-composer-studio-V4.

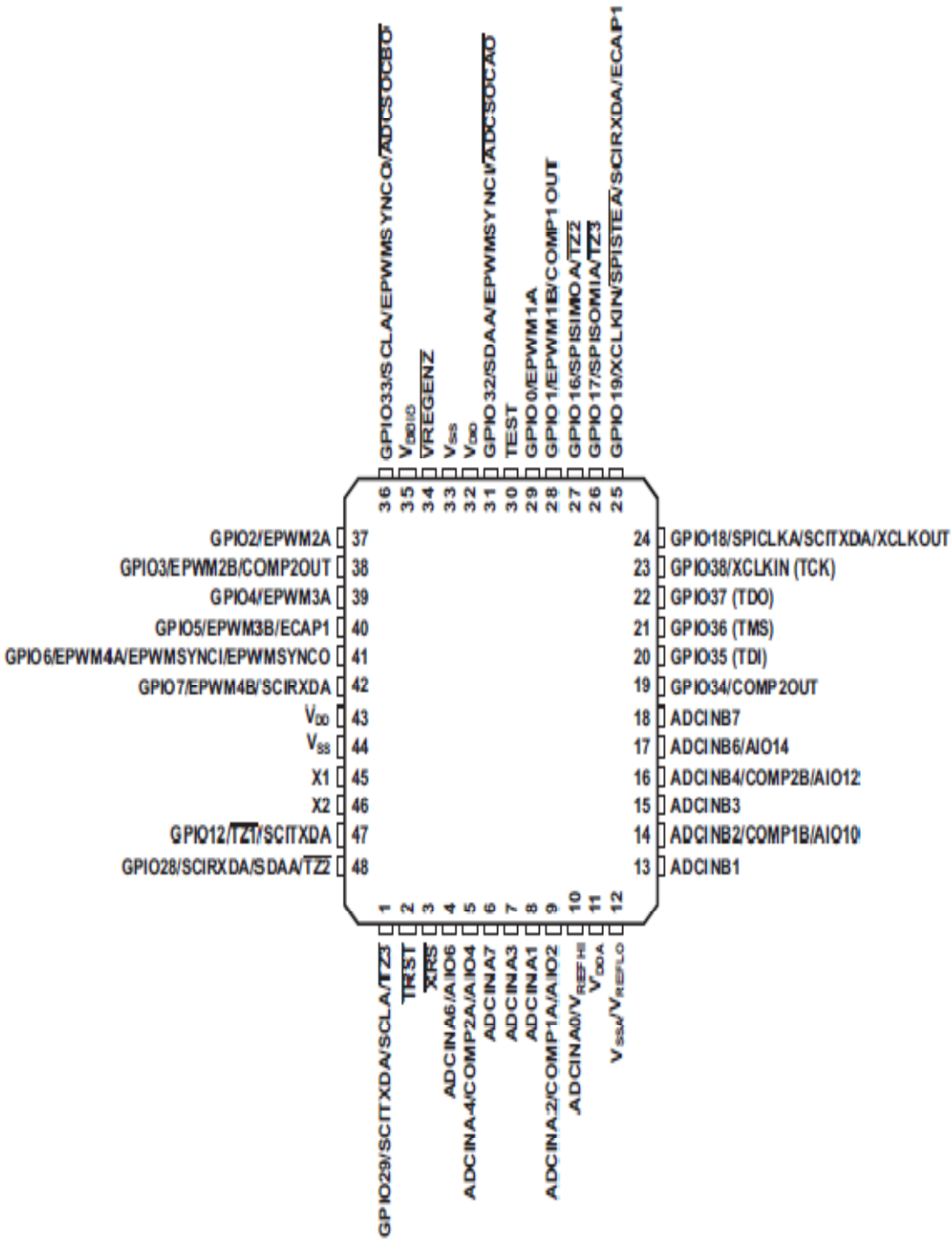


Schéma externe du TMSF28027

Fonction des pins du TMS320F28027

TERMINAL			I/O/Z	DESCRIPTION
NAME	PT NO.	DA NO.		
RESET				
$\overline{XRS}$	3	17	I/O	Device Reset (in) and Watchdog Reset (out). Piccolo devices have a built-in power-on-reset (POR) and brown-out-reset (BOR) circuitry. As such, no external circuitry is needed to generate a reset pulse. During a power-on or brown-out condition, this pin is driven low by the device. See the electrical section for thresholds of the POR/BOR block. This pin is also driven low by the MCU when a watchdog reset occurs. During watchdog reset, the $\overline{XRS}$ pin is driven low for the watchdog reset duration of 512 OSCCLK cycles. If need be, an external circuitry may also drive this pin to assert a device reset. In this case, it is recommended that this pin be driven by an open-drain device An R-C circuit must be connected to this pin for noise immunity reasons. Regardless of the source, a device reset causes the device to terminate execution. The program counter points to the address contained at the location 0x3FFFC0. When reset is deactivated, execution begins at the location designated by the program counter. The output buffer of this pin is an open-drain with an internal pullup. (I/OD)
ADC, COMPARATOR, ANALOG I/O				
ADCINA7	6		I	ADC Group A, Channel 7 input
ADCINA6 AIO6	4	18	I I/O	ADC Group A, Channel 6 input Digital AIO 6
ADCINA4 COMP2A AIO4	5	19	I I I/O	ADC Group A, Channel 4 input Comparator Input 2A Digital AIO 4
ADCINA3	7	–	I	ADC Group A, Channel 3 input
ADCINA2 COMP1A AIO2	9	20	I I I/O	ADC Group A, Channel 2 input Comparator Input 1A Digital AIO 2
ADCINA1	8	–	I	ADC Group A, Channel 1 input
ADCINA0 VREFHI	10	21	I I	ADC Group A, Channel 0 input ADC External Reference – only used when in ADC external reference mode. See ADC Section.
ADCINB7	18	–	I	ADC Group B, Channel 7 input
ADCINB6 AIO14	17	26	I I/O	ADC Group B, Channel 6 input Digital AIO 14
ADCINB4 COMP2B AIO12	16	25	I I I/O	ADC Group B, Channel 4 input Comparator Input 2B Digital AIO12
ADCINB3	15	–	I	ADC Group B, Channel 3 input
ADCINB2 COMP1B AIO10	14	24	I I I/O	ADC Group B, Channel 2 input Comparator Input 1B Digital AIO 10
ADCINB1	13	–	I	ADC Group B, Channel 1 input
CPU AND I/O POWER				
V_{DDA}	11	22		Analog Power Pin
V_{SSA} V_{REFLO}	12	23	I	Analog Ground Pin ADC Low Reference (always tied to ground)
V_{DD}	32	1		CPU and Logic Digital Power Pins – no supply source needed when using internal VREG. Tie with 1.2 µF ceramic capacitor to ground when using internal VREG.
V_{DD}	43	11		
V_{DDIO}	35	4		Digital I/O and Flash Power Pin – Single Supply source when VREG is enabled
V_{SS}	33	2		Digital Ground Pins
V_{SS}	44	12		
VOLTAGE REGULATOR CONTROL SIGNAL				
$\overline{VREGENZ}$	34	3	I	Internal VREG Enable/Disable – pull low to enable VREG, pull high to disable VREG
GPIO AND PERIPHERAL SIGNALS				
GPIO0	29	37	I/O/Z	General purpose input/output 0

TERMINAL			I/O/Z	DESCRIPTION
NAME	PT NO.	DA NO.		
EPWM1A			O	Enhanced PWM1 Output A and HRPWM channel
–			–	–
–			–	–
GPIO1	28	36	I/O/Z	General purpose input/output 1
EPWM1B			O	Enhanced PWM1 Output B
–			–	
COMP1OUT			O	Direct output of Comparator 1
GPIO2	37	5	I/O/Z	General purpose input/output 2
EPWM2A			O	Enhanced PWM2 Output A and HRPWM channel
–				–
–				–
GPIO3	38	6	I/O/Z	General purpose input/output 3
EPWM2B			O	Enhanced PWM2 Output B
–				–
COMP2OUT			O	Direct output of Comparator 2
GPIO4	39	7	I/O/Z	General purpose input/output 4
EPWM3A			O	Enhanced PWM3 output A and HRPWM channel
–				–
–				–
GPIO5	40	8	I/O/Z	General purpose input/output 5
EPWM3B			O	Enhanced PWM3 output B
–				–
ECAP1			I/O	Enhanced Capture input/output 1
GPIO6	41	9	I/O/Z	General purpose input/output 6
EPWM4A			O	Enhanced PWM4 output A and HRPWM channel
EPWMSYNCI			I	External ePWM sync pulse input
EPWMSYNCO			O	External ePWM sync pulse output
GPIO7	42	10	I/O/Z	General purpose input/output 7
EPWM4B			O	Enhanced PWM4 output B
SCIRXDA			I	SCI-A receive data
–				–
GPIO12	47	13	I/O/Z	General purpose input/output 12
$\overline{TZ1}$			I	Trip Zone input 1
SCITXDA			O	SCI-A transmit data
–				–
GPIO16	27	35	I/O/Z	General purpose input/output 16
SPISIMOA			I/O	SPI slave in, master out
–				–
$\overline{TZ2}$			I	Trip Zone input 2
GPIO17	26	34	I/O/Z	General purpose input/output 17
SPISOMIA			I/O	SPI-A slave out, master in
–				–
$\overline{TZ3}$			I	Trip zone input 3
GPIO18	24	32	I/O/Z	General purpose input/output 18
SPICLKA			I/O	SPI-A clock input/output
SCITXDA			O	SCI-A transmit

TERMINAL			I/O/Z	DESCRIPTION
NAME	PT NO.	DA NO.		
XCLKOUT			O/Z	Output clock derived from SYSCLKOUT. XCLKOUT is either the same frequency, one-half the frequency, or one-fourth the frequency of SYSCLKOUT. This is controlled by bits 1:0 (XCLKOUTDIV) in the XCLK register. At reset, XCLKOUT = SYSCLKOUT/4. The XCLKOUT signal can be turned off by setting XCLKOUTDIV to 3. The mux control for GPIO18 must also be set to XCLKOUT for this signal to propogate to the pin.
GPIO19	25	33	I/O/Z	General purpose input/output 19
XCLKIN				External Oscillator Input. The path from this pin to the clock block is not gated by the mux function of this pin. Care must be taken not to enable this path for clocking if it is being used for the other peripherhal functions
$\overline{\text{SPISTEA}}$			I/O	SPI-A slave transmit enable input/output
SCIRXDA			I	SCI-A receive
ECAP1			I/O	Enhanced Capture input/output 1
GPIO28	48	14	I/O/Z	General purpose input/output 28
SCIRXDA			I	SCI receive data
SDAA			I/OC	I2C data open-drain bidirectional port
$\overline{\text{TZ2}}$			I	Trip zone input 2
GPIO29	1	15	I/O/Z	General purpose input/output 29.
SCITXDA			O	SCI transmit data
SCLA			I/OC	I2C clock open-drain bidirectional port
$\overline{\text{TZ3}}$			I	Trip zone input 3
GPIO32	31	-	I/O/Z	General purpose input/output 32
SDAA			I/OC	I2C data open-drain bidirectional port
EPWMSYNCI			I	Enhanced PWM external sync pulse input
$\overline{\text{ADCSOCAO}}$			O	ADC start-of-conversion A
GPIO33	36	–	I/O/Z	General-Purpose Input/Output 33
SCLA			I/OC	I2C clock open-drain bidirectional port
EPWMSYNCO			O	Enhanced PWM external synch pulse output
$\overline{\text{ADCSOCBO}}$			O	ADC start-of-conversion B
GPIO34	19	27	I/O/Z	General-Purpose Input/Output 34
COMP2OUT			O	Direct output of Comparator 2
–				–
–				–
GPIO35	20	28	I/O/Z	General-Purpose Input/Output 35
TDI			I	JTAG test data input (TDI) with internal pullup. TDI is clocked into the selected register (instruction or data) on a rising edge of TCK
GPIO36	21	29	I/O/Z	General-Purpose Input/Output 36
TMS			I	JTAG test-mode select (TMS) with internal pullup. This serial control input is clocked into the TAP controller on the rising edge of TCK.
GPIO37	22	30	I/O/Z	General-Purpose Input/Output 37
TDO			O/Z	JTAG scan out, test data output (TDO). The contents of the selected register (instruction or data) are shifted out of TDO on the falling edge of TCK (8 mA drive)
GPIO38	23	31	I/O/Z	General-Purpose Input/Output 38
TCK			I	JTAG test clock with internal pullup
XCLKIN			I	External Oscillator Input. The path from this pin to the clock block is not gated by the mux function of this pin. Care must be taken to not enable this path for clocking if it is being used for the other peripherhal functions.

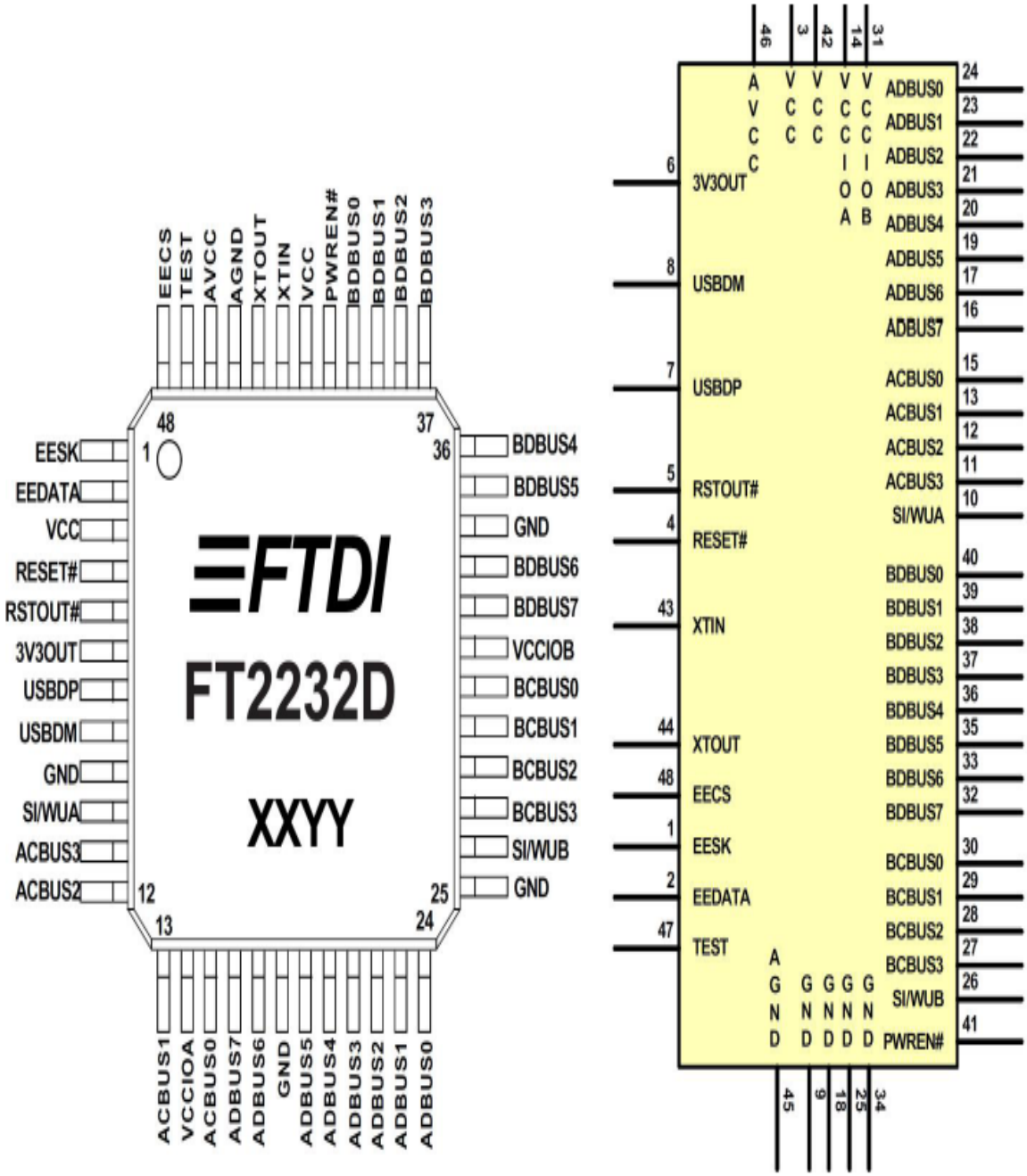


Schéma de l'émulateur FTDI FT2232D

Fonction de chaque pin de l'émulateur FTDI FT2232D

3.1 Pin Out Description

This section describes the operation of the FT2232D pins. Common pins are defined in the first section, and then the I/O pins are defined by chip mode.

Note: The convention used throughout this document for active low signals is the signal name followed by#

3.2 Common Pins

The operation of the following FT2232D pins do not change regardless of the configured mode:-

Pin No.	Name	Type	Description
7	USBDP	I/O	USB Data Signal Plus (Requires 1.5K pull-up to 3V3OUT or RSTOUT#)
8	USBDM	I/O	USB Data Signal Minus

Table 3.1.1 USB Interface Group

Pin No.	Name	Type	Description
48	EECS	I/O	EEPROM – Chip Select. Tri-State during device reset. ****Note 1**
1	EESK	OUTPUT	Clock signal to EEPROM. Tri-State during device reset, else drives out. ****Note 1**
2	EEDATA	I/O	EEPROM – Data I/O Connect directly to Data-In of the EEPROM and to Data-Out of the EEPROM via a 2.2K resistor. Also, pull Data-Out of the EEPROM to VCC via a 10K resistor for correct operation. Tri-State during device reset. ****Note 1**

Table 3.2.2 EEPROM Interface Group

Pin No.	Name	Type	Description
4	RESET#	INPUT	Can be used by an external device to reset the FT2232D. If not required, tie to VCC. ****Note 1**
5	RSTOUT#	OUTPUT	Output of the internal Reset Generator. Drives low for 5.6 ms after VCC > 3.5V and the internal clock starts up, then clamps it's output to the 3.3V output of the internal regulator. Taking RESET# low will also force RSTOUT# to drive low. RSTOUT# is NOT affected by a USB Bus Reset.
47	TEST	INPUT	Puts device into I.C. test mode – must be tied to GND for normal operation.
41	PWREN#	OUTPUT	Goes Low after the device is configured via USB, then high during USB suspend. Can be used to control power to external logic using a P-Channel Logic Level MOSFET switch. Enable the Interface Pull-Down Option in EEPROM when using the PWREN# pin in this way.
43	XTIN	INPUT	Input to 6MHz Crystal Oscillator Cell. This pin can also be driven by an external 6MHz clock if required. Note: Switching threshold of this pin is VCC/2, so if driving from an external source, the source must be driving at 5V CMOS level or a.c. coupled to centre around VCC/2.
44	XTOUT	OUTPUT	Output from 6MHz Crystal Oscillator Cell. XTOUT stops oscillating during USB suspend, so take care if using this signal to clock external logic.

Table 3.3.3 Miscellaneous Signal Group

Pin No.	Name	Type	Description
6	3V3OUT	OUTPUT	3.3 volt Output from the integrated L.D.O. regulator This pin should be decoupled to GND using a 33nF ceramic capacitor in close proximity to the device pin. It's prime purpose is to provide the internal 3.3V supply to the USB transceiver cell and the RSTOUT# pin. A small amount of current (<= 5mA) can be drawn from this pin to power external 3.3V logic if required.
3, 42	VCC	PWR	+4.35 volt to +5.25 volt VCC to the device core, LDO and non-UART / FIFO controller interface pins.
14	VCCIOA	PWR	+3.0 volt to +5.25 volt VCC to the UART / FIFO A Channel interface pins 10..13, 15..17 and 19..24. When interfacing with 3.3V external logic in a bus powered design connect VCCIO to a 3.3V supply generated from the USB bus. When interfacing with 3.3V external logic in a self powered design connect VCCIO to the 3.3V supply of the external logic. Otherwise connect to VCC to drive out at 5V CMOS level.
31	VCCIOB	PWR	+3.0 volt to +5.25 volt VCC to the UART / FIFO B Channel interface pins 26..30, 32..33 and 35..40. When interfacing with 3.3V external logic in a bus powered design connect VCCIO to a 3.3V supply generated from the USB bus. When interfacing with 3.3V external logic in a self powered design connect VCCIO to the 3.3V supply of the external logic. Otherwise connect to VCC to drive out at 5V CMOS level.
9,18, 25, 34	GND	PWR	Device - Ground Supply Pins
46	AVCC	PWR	Device - Analog Power Supply for the internal x8 clock multiplier. A low pass filter consisting of a 470 Ohm series resistor and a 100 nF to GND should be used on the supply to this pin.
45	AGND	PWR	Device - Analog Ground Supply for the internal x8 clock multiplier

Table 3.4.4 Power and Ground Group

3.3 IO Pin Definitions by Chip Mode

The FT2232D will default to dual serial mode (232 UART mode on both channel A and B, if no external EEPROM is used, or the external EEPROM is blank. The definition of the following pins varies according to the chip's mode:-

Channel A

Pin#	*Generic Pin name*	*Pin Definitions by Chip Mode **Note 2*						
		232 UART Mode	*245 FIFO*	*Enhanced Asynchronous and Synchronous Serial*	*MPSSE **Note 4*	*MCU Host Bus Emulation Mode **Note 5*	*Fast Opto-Isolated Serial Mode*	*CPU FIFO Interface Mode*
24	ADBUS0	TXD	D0	D0	TCK/SK AD0	**Note 3	D0	D0
23	ADBUS1	RXD	D1	D1	TDI/D0	AD1	D1	D1
22	ADBUS2	RTS#	D2	D2	TDO/DI	AD2	D2	D2
21	ADBUS3	CTS#	D3	D3	TMS/CS AD3	D3		D3
20	ADBUS4	DTR#	D4	D4	GPIOL0	AD4	D4	D4
19	ADBUS5	DSR#	D5	D5	GPIOL1	AD5	D5	D5
17	ADBUS6	DCD#	D6	D6	GPIOL2	AD6	D6	D6
16	ADBUS7	RI#	D7	D7	GPIOL3	AD7	D7	D7
15	ACBUS0	TXDEN	RXF#	WR# **Note 6	GPIOH0	I/O0	CS#	CS#
13	ACBUS1	SLEEP#	TXE#	RD# **Note 6	GPIOH1	I/O1	A0	A0
12	ACBUS2	RXLED#	RD#	WR# **Note 7	GPIOH2	IORDY	RD#	RD#
24	ADBUS0	TXD	D0	D0	TCK/SK AD0	**Note 3	D0	
11	ACBUS3	TXLED#	WR	RD# **Note 7	GPIOH3	OSC	WR#	WR#
10	SI/WUA	SI/WUA	SI/WUA	SI/WUA	**Note 8	**Note 8	**Note 8	**Note 8

Table 3.5.5 Pin Definition by Chip Mode (Channel A)

****Note 2:** 232 UART, 245 FIFO, CPU FIFO Interface, and Fast Opto-Isolated modes are enabled in the external EEPROM. Enhanced Asynchronous and Synchronous Bit-Bang modes, MPSSE, and MCU Host Bus Emulation modes are enabled using the driver command set bit mode. See Section 3.3 for details.

****Note 3:** Channel A can be configured in another IO mode if channel B is in Fast Opto-Isolated Serial Mode. If both Channel A and Channel B are in Fast Opto-Isolated Serial Mode all of the IO will be on Channel B.

****Note 4:** MPSSE is Channel A only.

****Note 5:** MCU Host Bus Emulation requires both Channels.

****Note 6:** The Bit-Bang Mode (synchronous and asynchronous) WR# and RD# strobes are on these pins when the main Channel mode is 245 FIFO, CPU FIFO interface, or Fast Opto-Isolated Serial Modes.

****Note 7:** The Bit-Bang Mode (synchronous and asynchronous) WR# and RD# strobes are on these pins when the main Channel mode is 232 UART Mode.

**Note 8: SI/WU is not available in these modes.

Channel B

Pin#	Generic Pin name	Pin Definitions by Chip Mode **Note 2						
		232 UART Mode	245 FIFO	Enhanced Asynchronous and Synchronous Serial	MPSSE **Note 4	MCU Host Bus Emulation Mode **Note 5	Fast Opto-Isolated Serial Mode	CPU FIFO Interface Mode
40	BDBUS0	TXD	D0	D0	A8	FSDI	D0	D0
39	BDBUS1	RXD	D1	D1	A9	FSCLK	D1	D1
38	BDBUS2	RTS#	D2	D2	A10	FSDO	D2	D2
37	BDBUS3	CTS#	D3	D3	A11	FSCTS	D3	D3
36	BDBUS4	DTR#	D4	D4	A12	**Note 3	D4	D4
35	BDBUS5	DSR#	D5	D5	A13	D5		D5
33	BDBUS6	DCD#	D6	D6	A14	D6		D6
32	BDBUS7	RI#	D7	D7	A15	D7		D7
30	BCBUS0	TXDEN	RXF#	WR# **Note 9	CS#	CS#		CS#
29	BCBUS1	SLEEP#	TXE#	RD# **Note 9	ALE	A0		A0
28	BCBUS2	RXLED#	RD#	WR# **Note 7	RD#	RD#		RD#
27	BCBUS3	TXLED#	WR	RD# **Note 7	WR#	WR#		WR#
26	SI/WUB	SI/WUB	SI/WUB	SI/WUB	**Note 8	**Note 8	SI/WUB	**Note 8
40	BDBUS0	TXD	D0	D0	A8	FSDI	D0	

Table 3.6.6 Pin Definition by Chip Mode (Channel B)

**Note 2: 232 UART, 245 FIFO, CPU FIFO Interface, and Fast Opto-Isolated modes are enabled in the external EEPROM. Enhanced Asynchronous and Synchronous Bit-Bang modes, MPSSE, and MCU Host Bus Emulation modes are enabled using the driver command set bit mode. See Section 3.3 for details.

**Note 3: Channel A can be configured in another IO mode if channel B is in Fast Opto-Isolated Serial Mode. If both Channel A and Channel B are in Fast Opto-Isolated Serial Mode all of the IO will be on Channel B.

**Note 4: MPSSE is Channel A only.

**Note 5: MCU Host Bus Emulation requires both Channels.

**Note 6: The Bit-Bang Mode (synchronous and asynchronous) WR# and RD# strobes are on these pins when the main Channel mode is 245 FIFO, CPU FIFO interface, or Fast Opto-Isolated Serial Modes.

**Note 7: The Bit-Bang Mode (synchronous and asynchronous) WR# and RD# strobes are on these pins when the main Channel mode is 232 UART Mode.

**Note 8: SI/WU is not available in these modes.

**Note 9: The Bit-Bang Mode (synchronous and asynchronous) WR# and RD# strobes are on these pins when the main Channel mode is 245 FIFO, CPU FIFO interface. Bit-Bang mode is not available on Channel B when Fast Opto-Isolated Serial Mode is enabled.

3.4 IO Mode Command Hex Values

Enhanced Asynchronous and Synchronous Bit-Bang modes, MPSSE, and MCU Host Bus Emulation modes are enabled using the D2XX driver command FT_SetBitMode. The hex values used with this command to enable these modes are as follows-

Mode	Value (Hex)
Reset the IO bit Mode	0
Asynchronous Bit Bang Mode	1
MPSSE	2
Synchronous Bit bang Mode	4
MCU Host bus Emulation	8
Fast Opto Isolated Serial Mode	10

Table 3.7.7 IO Mode Command Hex Values

See application note **AN2232-02, "Bit Mode Functions for the FT2232D"** for more details and examples.

Note that all other device modes can be enabled in the external EEPROM, and do not require these values to be configured.

In the case of Fast Opto-Isolated Serial mode sending a value of 10 will hold this device mode in reset, and sending a value of 0 will release this mode from reset.

Programme principal d'acquisition: ContinuousADC

```
//        FILE:                   ContinuousADC-Main.C
//
//        Description:    This program converts the 11 available ADC inputs on the
//                                        Piccolo controlSTICK in continous mode.
//
//        Version:                1.0
//
//  Target:               TMS320F2802x or TMS320F2803x families (Piccolo)
//
//---------------------------------------------------------------------------
//  $TI Release:$         V1.1
//  $Release Date:$ 26 Oct 2009 - BL
//---------------------------------------------------------------------------
//
// PLEASE READ - Useful notes about this Project

// Although this project is made up of several files, the most important ones are:
//          "ContinuousADC-Main.C"    - this file
//                  - Application Initialization, Peripheral config
//                  - Application management
//                  - Slower background code loops and Task scheduling
//          "ContinuousADC-DevInit_F28xxx.C
//                  - Device Initialization, e.g. Clock, PLL, WD, GPIO mapping
//                  - Peripheral clock enables
// The other files are generally used for support and defining the registers as C
// structs. In general these files will not need to be changed.
//   "F28027_RAM_ContinuousADC.CMD" or
"F28027_FLASH_ContinuousADC.CMD"
//                  - Allocates the program and data spaces into the device's memory map.
//   "DSP2802x_Headers_nonBIOS.cmd" and "DSP2802x_GlobalVariableDefs.c"
//                  - Allocate the register structs into data memory.  These register structs are
//                    defined in the peripheral header includes (DSP2802x_Adc.h, ...)
//
//---------------------------------------------------------------------------

#include "PeripheralHeaderIncludes.h"



// FUNCTION PROTOTYPES
void DeviceInit(void);
void InitFlash(void);
void MemCopy(Uint16 *SourceAddr, Uint16* SourceEndAddr, Uint16* DestAddr);
```

```
// VARIABLE DECLARATIONS - GENERAL

// Used for running BackGround in flash, and ISR in RAM
extern Uint16 RamfuncsLoadStart, RamfuncsLoadEnd, RamfuncsRunStart;

Uint16 AdcResults[16];
Uint16 x[500];

// MAIN CODE - starts here

void main(void)
{
        int16 i = 0;
   Uint16 j = 0;

//=================================
//      INITIALISATION - General
//=================================

        DeviceInit();   // Device Life support & GPIO mux settings

// Only used if running from FLASH
// Note that the variable FLASH is defined by the compiler with -d FLASH
// (see TwoChannelBuck.pjt file)
#ifdef FLASH
// Copy time critical code and Flash setup code to RAM
// The  RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
// symbols are created by the linker. Refer to the linker files.
        MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);

// Call Flash Initialization to setup flash waitstates
// This function must reside in RAM

        InitFlash();      // Call the flash wrapper init function

#endif //(FLASH)

//=================================
//      INITIALISATION - Peripherals
//=================================

// ADC INITIALISATION
   EALLOW;
        AdcRegs.ADCCTL1.bit.ADCREFSEL      = 0;     // Use internal bandgap
        AdcRegs.ADCCTL1.bit.ADCBGPWD       = 1;     // Power up band gap
        AdcRegs.ADCCTL1.bit.ADCREFPWD      = 1;     // Power up reference
```

```
        AdcRegs.ADCCTL1.bit.ADCPWDN     = 1;    // Power up rest of ADC
        AdcRegs.ADCCTL1.bit.ADCENABLE   = 1;    // Enable ADC

                                        // wait 60000 cycles = 1ms (each iteration is 12
cycles)

        AdcRegs.ADCCTL1.bit.INTPULSEPOS     = 1;    // create int pulses 1 cycle
prior to output latch

        // set S/H window to 6 clk cycles (117ns)
//      AdcRegs.ADCSOC0CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC1CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC2CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC4CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC7CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC9CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC10CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC11CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC12CTL.bit.ACQPS = 6;
        AdcRegs.ADCSOC14CTL.bit.ACQPS = 6;
//      AdcRegs.ADCSOC15CTL.bit.ACQPS = 6;

        AdcRegs.INTSEL1N2.bit.INT1SEL =14;          // ADCCH14 (ADC-B6)
EOC causes ADCInterrupt 1
        AdcRegs.INTSEL1N2.bit.INT1CONT = 1;         // set ADCInterrupt 1 to auto
clr
        AdcRegs.INTSEL1N2.bit.INT1E = 1;            // enable ADC interrupt 1

// Note that SOC3, 5, 6, 8, & 13 are valid, but these SOCs are not configured
// since these ADC outputs do not exist on the controlSTICK. The configuration
// is configured as it is for readability.

        //EOC = end of conversion event; SOC = start of conversion event
//      AdcRegs.ADCINTSOCSEL1.bit.SOC0 = 0;         // ADCInterrupt 1 causes
SOC0
//      AdcRegs.ADCINTSOCSEL1.bit.SOC1 = 0;
//      AdcRegs.ADCINTSOCSEL1.bit.SOC2 = 0;
//      AdcRegs.ADCINTSOCSEL1.bit.SOC4 = 0;
//      AdcRegs.ADCINTSOCSEL1.bit.SOC7 = 0;
//      AdcRegs.ADCINTSOCSEL2.bit.SOC9 = 0;
//      AdcRegs.ADCINTSOCSEL2.bit.SOC10 = 0;
//      AdcRegs.ADCINTSOCSEL2.bit.SOC11 = 0;
 //         AdcRegs.ADCINTSOCSEL2.bit.SOC12 = 1;
           AdcRegs.ADCINTSOCSEL2.bit.SOC14 = 1;
//      AdcRegs.ADCINTSOCSEL2.bit.SOC15 = 0;
```

```
// Select the channel to be converted when SOCx is received
//        AdcRegs.ADCSOC0CTL.bit.CHSEL= 0;     // convert ADC-A0 (CH0) when
SOC0 is received
//        AdcRegs.ADCSOC1CTL.bit.CHSEL= 1;     // convert ADC-A1 (CH1) when
SOC1 is received
//        AdcRegs.ADCSOC2CTL.bit.CHSEL= 2;
//        AdcRegs.ADCSOC4CTL.bit.CHSEL= 4;
//        AdcRegs.ADCSOC7CTL.bit.CHSEL= 7;
//        AdcRegs.ADCSOC9CTL.bit.CHSEL= 9;     // convert ADC-B1 (CH9) when
SOC9 is received
//        AdcRegs.ADCSOC10CTL.bit.CHSEL= 10;
//        AdcRegs.ADCSOC11CTL.bit.CHSEL= 11;
//        AdcRegs.ADCSOC12CTL.bit.CHSEL= 12;
          AdcRegs.ADCSOC14CTL.bit.CHSEL= 14;
//        AdcRegs.ADCSOC15CTL.bit.CHSEL= 15;

          EDIS;

          AdcRegs.ADCSOCFRC1.all = 0x4000;        // kick start ADC by causing a
SOC14 event



//===================================
//        Forever LOOP
//===================================
   i=0;
          for(i=0; i<500; i++)  //infinite loop
          {

                        for(j=0; j<5000; j++){}; //1ms

                        x[i] = AdcResult.ADCRESULT14;



                                    }
} //END MAIN CODE
```

Les MEMS

Un microsystème électromécanique est un microsystème comprenant un ou plusieurs éléments mécaniques, utilisant l'électricité comme source d'énergie, en vue de réaliser une fonction de capteur et/ou d'actionneur avec au moins une structure présentant des dimensions micrométriques ; et la fonction du système est en partie assurée par la forme de cette structure. Le terme systèmes microélectromécaniques est la version française de l'acronyme anglais MEMS (*Microelectromechanical systems*). En Europe, le terme MST pour *MicroSystem Technology* est également d'usage, bien que nettement moins répandu.

Issus de la technologie de la micro-électronique, les MEMS font appel pour leur fabrication aux microtechnologies, qui permettent une production à grande échelle. Ils sont utilisés dans des domaines aussi variés que l'automobile, l'aéronautique, la médecine, la biologie, les télécommunications, ainsi que dans certaines applications « de tous les jours » telles que certains vidéoprojecteurs, téléviseurs haute-définition ou coussins gonflables de sécurité pour automobiles (« Airbags »).

Les MEMS ont été développés au début des années 1970 en tant que dérivés de la micro-électronique et leur première commercialisation remonte aux années 1980 avec des capteurs de pression sur silicium qui remplacèrent rapidement les technologies plus anciennes et constituent encore une part importante du marché des MEMS. Depuis lors les MEMS ont connu un important développement et restent encore en plein essor.

C'est un domaine de recherche relativement récent qui combine l'utilisation des techniques électroniques, informatiques, chimiques, mécaniques, optiques. Les MEMS sont le plus souvent à base de silicium, mais on utilise également d'autres matériaux suivant l'adéquation de leurs propriétés physiques à certaines applications, comme les métaux, les matériaux piézoélectriques, divers polymères, etc.

Face au développement de ce domaine, on a vu apparaitre des termes dérivés pour désigner des MEMS spécialisés. Par exemple, dans le domaine optique on utilise le terme MOEMS (*Micro Opto Electro Mechanical Systems*) ou Optical MEMS, alors que dans le domaine biologique on utilise bioMEMS. On notera aussi un nouveau terme, NEMS (*Nano Electro Mechanical Systems*), Nanosystèmes en français, désignant des structures semblables aux MEMS mais de taille sub-micrométrique

Les technologies de fabrication des microsystèmes dérivent assez largement de celles de la microélectroniques. Des *Wafers* de silicium sont généralement utilisés comme substrat, et les microsystèmes sont produits par une succession d'étapes d'épitaxie, de résinage, de photolithographie et d'attaque sèche ou humide.

Les principales spécificités des technologies de microsystèmes, comparées à la microélectronique, sont liées à la réalisation de parties mobiles, donc relativement détachées du substrat, ce qui s'obtient généralement par recours à une couche sacrificielle.

Si les laboratoires ont imaginé et produit un nombre immense de MEMS, avec des applications allant de l'électronique à la biologie, les plus importants (industriellement) sont :

- les injecteurs pour imprimantes à jet d'encre ;
- les micro-miroirs qui définissent les pixels de certains modèles de vidéoprojecteurs ;
- la première projection cinéma numérique publique d'Europe (2000) réalisée par Philippe Binant reposait sur l'utilisation d'un *Optical* MEMS développé par TI[1] ;
- les accéléromètres destinés à des domaines divers tels que l'automobile ou plus récemment le jeu vidéo, comme la manette à détection de mouvement de la console de jeu Wii de Nintendo[2], ou le téléphone iPhone d'Apple ;
- les vannes de contrôle microfluidiques ;
- les micro-relais, le plus souvent à actionnement capacitif ;
- les émetteurs/récepteurs acoustiques, comme les cMUTs (capacitifs) ou les pMUTs (piézoélectriques) ;

- les capteurs de pression ;
- les filtres électromécaniques, qui isolent une fréquence du signal d'entrée en utilisant la résonance d'un système masse-ressort.