الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne démocratique et populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
المركز الجامعي لعين تموشنت
Centre Universitaire Belhadj Bouchaib d'Ain-Temouchent
Institut des Sciences et de la Technologie
Département de Génie Electrique

Projet de fin d'études
Pour l'obtention du diplôme de Master en :
Domaine : SCIENCE ET TECHNOLOGIE
Filière Electronique
Spécialité : Génie des Télécommunications
Thème

Segmentation et Détection de Contours d'Image sur TMS320C6713 DSK

Présenté Par :
1) BENFODDA Hicham Elhocin
2) BENALLAL Lahouari

Devant les jurys composés de :

Mme.FEROUANI Souhila	Docteur	C.U.B.B (Ain Temouchent)	Président
Mlle.BOUTKHIL Malika	Docteur	C.U.B.B (Ain Temouchent)	Encadrant
M.BEDDAD Boucif	Doctorant	U.D.L (Sidi Bel Abbès)	Co-Encadrant
M.DEBBAL.Mohammed	Docteur	C.U.B.B (Ain Temouchent)	Examinateur

Année universitaire 2015/2016

REMERCIMENTS

Nous remercions, en premier lieu dieu le plus puissant qui a bien voulu nous donner la force et le courage pour effectuer le présent travail.

Nos plus vifs remerciements vont à l'égard de notre encadreur Mme.BOUTKHIL Malika pour ces orientations, ses conseils, ses remarques judicieuses et sa disponibilité, ainsi que Mr.BEDDAD Boucif pour la qualité de ses conseils, le sérieux de son travail, et le haut sens de l'intérêt pour les étudiants motivés.

Merci également aux membres du jury pour avoir consenti à lire ce modeste travail, ainsi qu'à l'ensemble des enseignants du Département de Génie Électrique.

Bien évidemment ,nous remercions tout le personnel d'Algérie télécom d'Ain Temouchent.

DÉDICACE

Nous dédions ce modeste travail et nos profondes gratitudes à :

Notre cher papa et notre chère maman pour l'éducation qu'ils nous ont prodigués; avec tous les moyens et au prix de tous les sacrifices qu'ils ont consentis à nos égards, pour le sens du devoir qu'ils nous ont enseigné depuis l'enfance.

A mes frères : Mohamed et Zoheir.

A mes sœurs : Imane et Marwa.

A mes grands mère : Farida et Yamina.

A toute la famille BENFODDA et BRIK.

BENFODDA Hicham Elhocin

DÉDICACE

Louange à dieu, Seigneur des mondes ;

C'est toi que nous adorons et de toi que nous implorons secours ;

Que la paix et la bénédiction soit sur son dernier envoyé ;

A mes chers parents qui ont été présents, à chaque instant, leur

Irremplaçable et inconditionnel soutien m'a permis d'écarter les doutes,

de soigner les blessures et partager mes joies ;

A mon père ***Ahmed****,*

Mes frères et sœurs ;

A ma petite famille,

A tout mes tantes et mes oncles ;

A mes cousins ;

A tout la famille ……;

A tout mes amis sans exception ;

BENALLAL Lahouri

SOMMAIRE

Résumé :

Le traitement du signal numérique est considéré comme la plus importante technique qui contrôle le développement actuel. Là où il est impliqué dans des nombreuses applications surtout dans le traitement d'image numérique, l'imagerie médicale, radar, et bien d'autres applications. Chacune des applications précédentes ont leurs propres techniques de base mathématique et algorithmes.

Au cours de notre projet de fin d'étude, nous avons consacré une partie pour étudier et développer certain algorithmes de traitement numérique d'image : filtrage 2D, Segmentation d'image, détection de contour, et reconnaissance ,l'autre partie pour implémenter ces algorithmes sur une carte TMS320C6713 DSK à base de DSP.

Mots clés : Traitement numérique d'image, TMS320C6713 DSK, Code Composer Studio CCS, MATLAB, Modèle Simulink, DSP.

ملخص:

تعتبر معالجة الإشارة الرقمية من أهم التقنيات على الإطلاق التي تحكم التطور في القرن الحالي، حيث إنها تدخل في تطبيقات كثيرة و متنوعة منها معالجة الصورة الرقمية، التصوير الطبي، الرادار وغيرها من التطبيقات. كل من التطبيقات السابقة لها تقنياتها الخاصة بها من أساس رياضي وخوارزميات.

وخلال الدراسة النهائية لمشروع البحث، استخدمنا جزء لدراسة وتطوير بعض خوارزميات معالجة الصور الرقمية: الترشيح 2D وتجزئة الصورة، الكشف عن الحافة،النعرف, والجزء الآخر لتنفيذ هذه الخوارزميات على لوحة TMS320C6713 DSK على أساس DSP.

كلمات المفتاح: معالجة الصور الرقمية، TMS320C6713 DSK, Code Composer Studio CCS, DSP MATLAB, Simulink معالج الإشارات الرقمية.

Abstract :

The digital signal processing is considered the most important technique that controls the current development. where it is involved in many applications especially in the digital image processing, medical imaging, radar, and many other applications. Each of the above applications have their own mathematical basic techniques and algorithms.

During our final project study, we used a part to study and develop some digital image processing algorithms : 2D filtering, image segmentation, edge detection, and the other part to implement these algorithms on a TMS320C6713 DSK board based on DSP.

Keywords : Digital image processing, TMS320C6713 DSK, CCS Code Composer Studio, MATLAB, Simulink Model, digital signal processor DSP.

Liste des Figures

Liste des Tableaux

Introduction Générale

Avec la parole, l'image constitue l'un des moyens les plus importants qu'utilise l'homme pour communiquer avec autrui. C'est un moyen de communication universel dont la richesse du contenu permet aux êtres humains de tout âge et de toute culture de se comprendre. C'est aussi le moyen le plus efficace pour communiquer, chacun peut analyser l'image à sa manière, pour en dégager une impression et d'en extraire des informations précises. De ce fait, le traitement numérique d'images est l'ensemble des méthodes et techniques opérant sur celles-ci, dans le but de rendre cette opération possible, plus simple, plus efficace et plus agréable, pour extraire des informations jugées pertinentes.

Dans notre projet de fin d'étude nous allons montrer les différentes étapes de notre processus proposé pour l'implémentation d'algorithme de traitement d'image sur DSP. L'objectif de notre travail est d'étudier une carte TMS 320C6713 DSK à base du DSP (Digital Signal Processor) afin d'implémenter l'algorithme de la segmentation et détection de contours. Notre projet de fin d'étude est organisé comme suit :

Chapitre 1 : Traitement Numérique d'image

Dans ce chapitre nous décrivons tout d'abord la description de l'image numérique, et nous avons définir les caractéristiques principales des images numériques afin d'étudier les opérations de traitement puis Mettre une analyse fine et approfondie porte sur les différents approches de segmentation et détection de contours dans une image.

Chapitre 2 : Généralités sur le TMS 320C6713

Nous décrirons les différentes familles de TMS 320Cx puis faire une description en détaille de la carte TMS 320C6713 DSK avec leurs caractéristiques et utilisation.

Chapitre 3 : Code Composer Studio CCS

Dans ce chapitre nous décrivons l'outil de développement: Code Composer Studio CCS, en basant sur le langage C pour la programmation (écriture de l'algorithme de traitement), et aussi les séquences de construction et de chargement et d'exécution.

Chapitre 4 : Algorithme de traitement d'image embarqué sur TMS320C6713

Ce chapitre présente le cœur de Notre mémoire de fin d'étude: il est dédié à l'implémentation d'algorithme de segmentation et détections des contours sur la carte TMS 320C6713 DSK.

Enfin, nous concluons notre travail en résumant les principales contributions que nous avons apportées dans le domaine de traitement numérique d'image et on justifie exactement le bon choix de la carte DSK TMS 320C6713 à base de DSP.

1

CHAPITRE :

Traitement Numérique d'Image

Introduction

La technologie numérique moderne est devenue omniprésente. Grâce à elle, il est devenu possible de traiter des signaux multidimensionnels avec des systèmes très divers, depuis les téléphones portables jusqu'aux ordinateurs massivement parallèles. Dans le domaine des images, on fait habituellement la distinction entre deux catégories: Le traitement Numérique des images et L'analyse Numérique d'images.

I-1 Image Numérique

Une image numérique est composée d'unités élémentaires (appelé pixel) qui représentent chacun une portion de l'image. Une image est définie par :

- Le nombre de pixels qui la compose en largeur et en hauteur (qui peut varier presque à l'infini),
- L'étendu des teintes de gris ou des couleurs que peut prendre chaque pixel (on parle de dynamique de l'image).

Toutes les données correspondant aux informations contenues dans l'image sont structurées d'une certaine façon afin de permettre leur stockage. Il existe un grand nombre de formats d'images, tous ces formats ne correspondent ni plus ni moins qu'à une structuration particulière des données concernant l'image. Une image numérique en elle-même est en fait un concept tout à fait abstrait (des données numériques) qui ne trouve une signification à nos yeux qu'à la visualisation lorsque l'on utilise un logiciel adéquat. La numérisation d'une image peut être obtenue par une simple Conversion analogique Numérique…CAN.de la sortie du capteur. La synchronisation du balayage de l'image avec le CAN, permet d'échantillonner la fonction image en un nombre de points finis, dans le plan x-y et de mémoriser ainsi les valeurs obtenues. Les formats images usuelles sont : 32×32 ; 100×100 ; 256×256 ; 512×512. Une image peut être ainsi numérisée sous la forme d'un tableau à L lignes et C colonnes. **[2]**

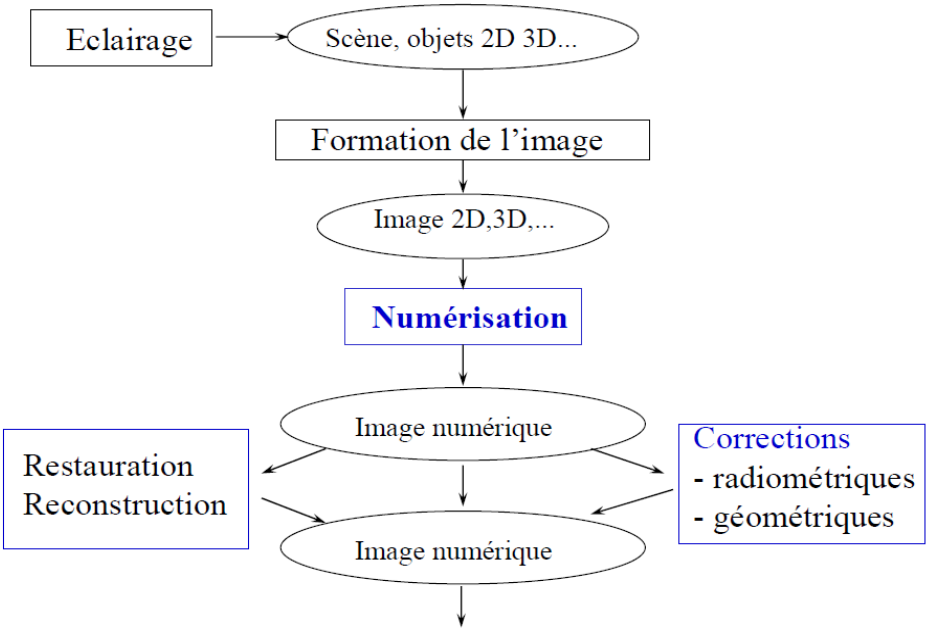


Figure 1.1 : Construction d'une image numérique

1-1 Pixel « Picture_element »

C'est l'élément d'image ou «Picture-element », le pixel est le plus petit point de l'image numérique (figure 1.2), c'est une entité calculable qui peut être quantifiée. Un pixel est aussi le plus petit élément que peuvent manipuler les matériels et logiciels d'affichage ou d'impression. L'ensemble des pixels est contenu dans un tableau à deux dimensions constituant l'image.

* Dans le cas d'une image monochrome chaque pixel est codé sur un octet.

* Dans une image en couleur ou à base de RVB (rouge, vert, bleu), un pixel est représenté sur trois octets (un octet pour chaque couleur). **[1]**

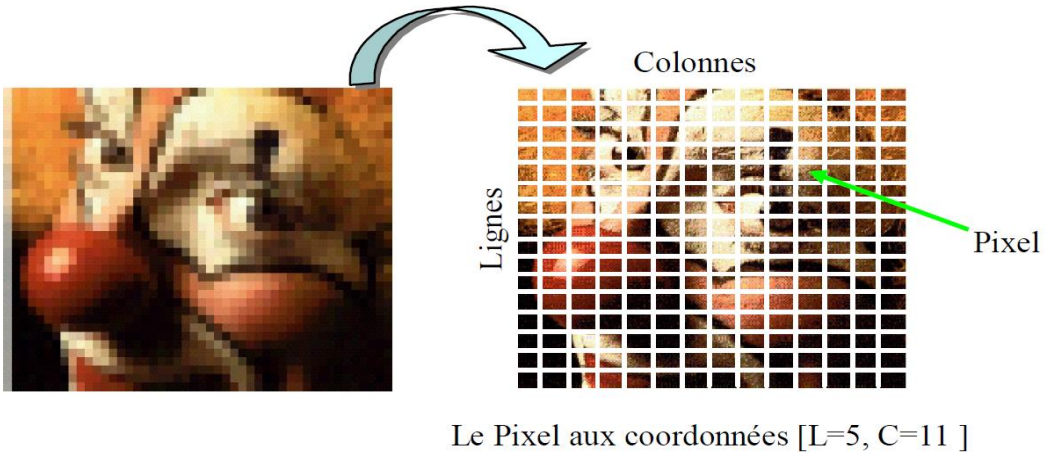


Figure 1.2 : Numérisation d'une image continue

I-2 Acquisition : Echantillonnage/Quantification

L'acquisition d'images est une mesure spatiale d'une interaction entre une onde et de la matière. L'onde est émise par une source et reçue par un capteur. Par exemple dans le cas de l'échographie, l'ultrason, une onde acoustique, est émis et reçue par la sonde. L'interaction est la réflexion de l'ultrason sur la structure du corps.

Dans le cas d'onde électromagnétique, la photographie utilise le spectre visible c'est-à-dire qui est visible pour l'œil humain. Il y a des applications sur l'ensemble du spectre électromagnétique, des rayons gamma jusqu'aux ondes radio. Ainsi, les images acquises par rayons X ou par rayons gamma sont surtout utilisées en imagerie médicale et en astronomie.

L'opération d'échantillonnage consiste à prélever sur un signal analogique dont l'évolution est continue dans le temps, des échantillons représentant l'amplitude aux instants de prélèvement. Pour des raisons de simplification, les prélèvements sont réalisés régulièrement avec une périodicité constante Te appelée période d'échantillonnage. L'échantillonnage est qualifié d'idéal dès lors que l'on peut supposer ou approcher une prise instantanée des échantillons. **[2]**

Echantillonnage d'une fonction f(x,y) :

$$f_e(x,y) = f(x,y).\Sigma_i \Sigma_i \delta(x - i \Delta x , y - j \Delta y) \quad (1.1)$$

Δx c'est le pas d'échantillonnage dans la direction x

Δy c'est le pas d'échantillonnage dans la direction y

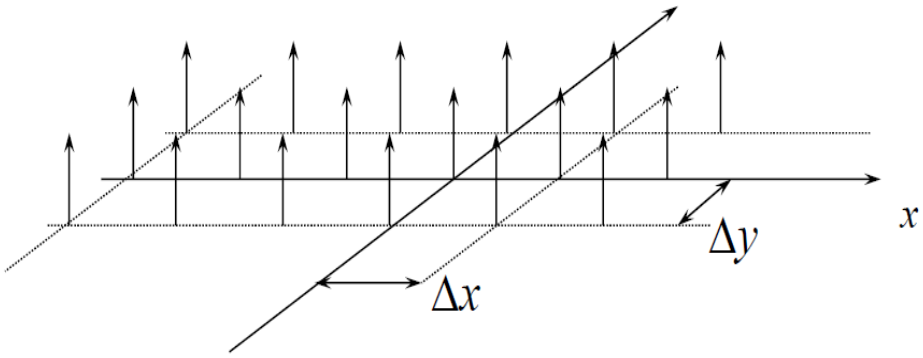


Figure 1.3 : Représentation échantillonnée

Tel que : $\Sigma i \Sigma j \, \delta(x - i\Delta x, y - j\Delta y)$ Peigne de Dirac 2D

Le poids $f(i\Delta x, j\Delta y)$ de chaque Dirac est :

- ✓ . Soit la valeur de f(x,y) en x = i Δx et y = j Δy
- ✓ . Soit la valeur «moyenne» de f(x,y) dans une région entourant (i Δx , j Δy) (f(x,y) est pondérée et intégrée dans la région R).

Dans le cas général on aura (cas variant) :

$$\tilde{f}(i\Delta x, j\Delta y) = \int_R f(x,y)h(x,y,i\Delta x, j\Delta y)dxdy \tag{1.2}$$

Si h(.,.) est identique en tout point (x,y), on aura (cas invariant) :

$$\tilde{f}(i\Delta x, i\Delta y) = \int_R f(x,y)h(i\Delta x - x, j\Delta y - y)dxdy \tag{1.3}$$

h représentera la réponse impulsionnelle du système de prise de vue. C'est une opération de convolution, donc de filtrage.

L'image échantillonnée est donc :

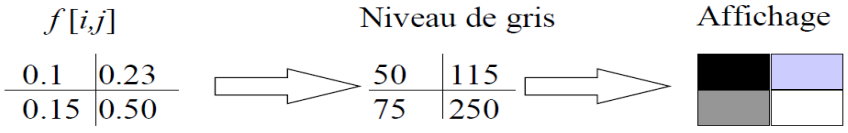
$$f_e(x,y) = \sum_i \sum_j \tilde{f}(i\Delta x, j\Delta y)\delta(x - i\Delta x, y - j\Delta y) \tag{1.4}$$

Dans un ordinateur, l'image (numérique) sera représentée par une matrice (tableau 2D) :

$$f[i,j] = \tilde{f}(i\Delta x, j\Delta y) \tag{1.5}$$

f [i,j] est appelé : «valeur du PIXEL (i,j) » (Pixel: PICture ELement)

Pour visualiser une image, on remplit une région rectangulaire (Pixel) avec un niveau de gris (ou de couleur) correspondant à la valeur du pixel. En général les niveaux de gris (ou de couleur) utilisés pour la visualisation sont compris entre 0 et 255 (code de longueur fixe sur 8 bits).



. CAN sur les systèmes d'acquisition d'images.

. Codage de la valeur de chaque pixel sur N bits (En général 8 bits).

La quantification est la seconde étape nécessaire à la numérisation des signaux. Succédant à l'échantillonnage-blocage, elle permet le traitement numérique ou la mémorisation du signal. Son rôle est d'affecter une valeur de résolution finie à un échantillon dont l'amplitude est en théorie infiniment précise si l'on fait abstraction du bruit de fond propre au signal.

Quantifier un échantillon, c'est arrondir sa valeur à celle de l'échelon le plus proche sur une grille de niveaux. Lorsque les échelons sont à pas constant, la quantification est uniforme.
La quantification est une opération non conservatrice car la précision originale ne peut être retrouvée après arrondi. Elle peut être vue comme la superposition d'une composante aléatoire sur l'amplitude de chaque échantillon original. **[2]**

Figure 1.4 : Exemple de quantification sur 8 bits, 4 bits, 2 bits

I-3 Codage des couleurs

En plus de sa définition, une image numérique utilise plus ou moins de mémoire selon le codage des informations de couleur qu'elle possède. C'est ce que l'on nomme le codage de couleurs ou profondeur des couleurs, exprimé en bit par pixel (bpp): 1, 4, 8, 16 bits...
En connaissant le nombre de pixels d'une image et la mémoire nécessaire à l'affichage d'un pixel, il est possible de définir exactement le poids que va utiliser le fichier image sur le disque dur (ou l'espace mémoire requis en RAM pour réaliser un calcul sur cette image). **[2]**
Comment calculer le poids d'une image en octet :
Poids (octet) = Nombre de pixel total X codage couleurs (octet)
Petit rappel du code binaire, utilisé par l'ordinateur pour enregistrer des informations. On sait que:
1bit = 2 états; (0 ou 1) = 21

2bits = 4 états, = 22

4bits = 16 états, = 24

8bits = 256 états, = 28 etc...

Un ensemble de 8bit = 1 Octet.

1024 Octets forment un kilo-octet (Ko).

1024 Kilo-Octets forment un Mega-Octet (Mo)...Giga-Octet...Terra-Octet...

1-3-1 Mode bitmap (Noir et blanc)

Avec ce mode, il est possible d'afficher uniquement des images en deux couleurs: noir et blanc. Il utilise une seule couche. Codage en 1 bit par pixel (bpp) : => 21 = 2 possibilités: [0,1] Chaque pixel peut donc avoir 2 couleurs possibles : soit noir ou soit blanc.

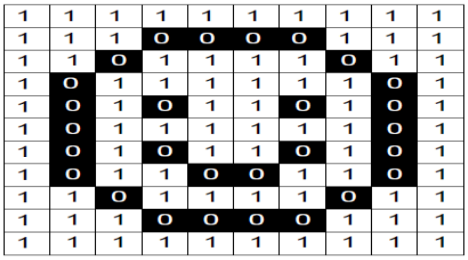


Figure 1.5 : Exemple d'image Bitmap

1-3-2 Mode Niveau de gris

Il permet d'obtenir différentes valeurs de gris, afin d'afficher des images nuancées. Il utilise qu'une seule couche.

- Codage en 8 bits par pixel (bpp) => 28= 256 possibilités, Chaque pixel peut avoir 256 nuances de gris possibles.
- Codage en 16 bits par pixel (bpp) => 216= 65536 possibilités, Chaque pixel peut avoir 65536 nuances de gris possibles.

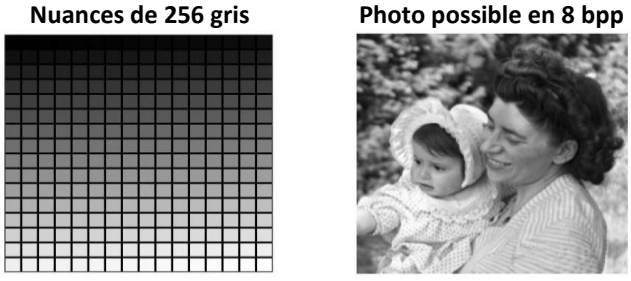


Figure 1.6 : Exemple d'image Niveau de gris

1-3-3 Mode couleurs indexées

Permet d'obtenir jusque 256 couleurs fixes, définies à l'avance dans une palette. Il n'utilise qu'une seule couche.

- Codage en 8 bits par pixel (bpp) => 28= 256 possibilités, Chaque pixel peut avoir jusqu'à 256 couleurs fixes possibles.

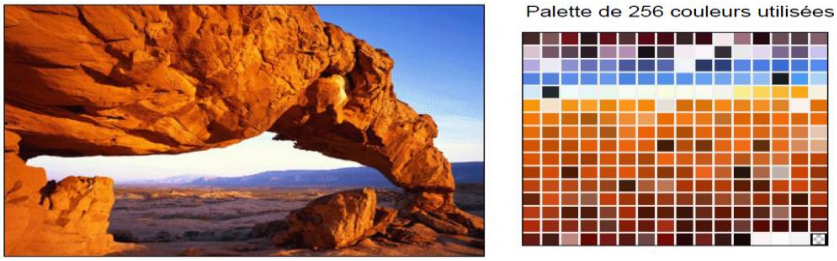
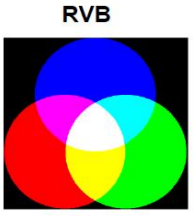


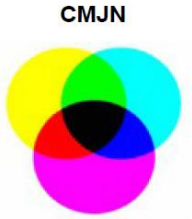
Figure 1.7 : Exemple d'image indexée

1-3-4 Mode colorimétriques RVB/CMJN

Afin de créer des images encore plus riches en couleurs (est donc disposer de plus qu'une palette limitée à 256 couleurs), l'idée de mélanger des couleurs primaires en « couches » est arrivée. Il faut savoir qu'il existe deux systèmes de représentation des couleurs par mélange, selon qu'on les reproduit sur un écran d'ordinateur ou sur support papier via une imprimante :

- **La synthèse additive** : c'est le phénomène qui se passe lorsqu'un un écran affiche une image par la lumière. On part du noir (lumière éteinte) et on va vers le blanc. L'addition du rouge, du vert et du bleu donne le blanc.
- **La synthèse soustractive** : c'est le phénomène qui se passe lorsqu'on mélange des pigments colorés en peinture. On part du blanc (support papier) pour aller vers le noir. L'addition du Cyan, du Magenta et du Jaune donne le Noir.





Mode RVB

Grâce au mélange des 3 couches de couleur, il est possible de reproduire un plus grand nombre de nuances qu'avec une palette en mode couleurs indexées.

- Avec un codage en RVB 8 bits PAR COUCHE:

Chaque couche utilise 8bit (1 octet), soit 256 nuances possibles: 8Bits pour le Rouge, 8bit pour le Vert et 8bits pour le Bleu. Donc utilisation de 3 x 8bits = 24 bits utilisées au total.

=> 256 x 256 x 256 = 224= 16,7milions, Chaque pixel peut prendre 16,7Milions de couleurs possibles.

- Avec un codage en RVB 16 bits PAR COUCHE:

Chaque couche utilise le double, soit 16bits! (65535 nuances). 3 x 16 = 48bits utilisées au total.

=> 65535 x 65535 x 65535 = 248= 4 milliards (4 milliards de nuances de couleurs sont possibles!)

Mode CMJN

Comme les écrans d'ordinateur ne peuvent afficher que du RGB, Photoshop sépare les images CMJN en 4 couches (Cyan, Magenta, Jaune et Noir ou chaque couleur est exprimée en pourcentage) et converti le tout en RGB pour être lu sur l'écran. Cependant le fichier possède bien 4 couches distinctes.

- Avec un codage en CMJN 8 bits PAR COUCHE:

Chaque couche utilise 8bit (soit 256 nuances possibles): 8Bits pour le Cyan, 8bit pour le Magenta, 8bits pour le Jaune et 8bits pour le Noir. Donc utilisation de 4 x 8bits = 32 bits utilisées au total.

=> 256 x 256 x 256 x 256 = 232= 4 milliards 4 milliards de nuances de couleurs sont possibles.

- Avec un codage en CMJN 16 bits PAR COUCHE:

Chaque couche utilise le double, soit 16bits! (65535 nuances). 3 x 16 = 64bits utilisées au total.

=> 65535 x 65535 x 65535 x 65535 = 248= 18446744073709551616 nuances de couleurs sont possibles.

I-4 Caractéristiques de l'image numérique

1-4-1 Dimension

C'est la taille de l'image, qui n'est rien qu'une matrice de pixels. La taille est obtenue en multipliant le nombre de colonne par le nombre ligne. Une image possédant 640 pixels en largeur et 480 en hauteur aura une définition (dimension) de 640 pixels par 480.

1-4-2 Taille d'une image

Pour connaître la taille (en octets) d'une image, il est nécessaire de compter le nombre de pixels que contient l'image, cela revient à calculer le nombre de cases du tableau, soit la hauteur de celui-ci que multiplie sa largeur. La taille (ou poids) de l'image est alors le nombre de pixels que multiplie la taille (en octets) de chacun de ces éléments, Voici le calcul pour une image 640 × 480 en True color

_ Nombre de pixels : 640 × 480 = 307200 24bits /8 = 3octets

_ Le poids de l'image est ainsi égal à : 307200 × 3 = 921600 octets 921600/1024 = 900Ko

Voici quelques exemples (en considérant que l'image n'est pas compressée) :

finition de l'image	Noir et blanc($1bit$)	256 couleurs ($8bits$)	65000 couleurs ($16bits$)	True color ($24bits$)
$320x200$	$7.8Ko$	$62.5Ko$	$125Ko$	$187.5Ko$
$640x480$	$37.5Ko$	$300Ko$	$600Ko$	$900Ko$
$800x600$	$58.6Ko$	$468.7Ko$	$937.5Ko$	$1.4Mo$
$1024x768$	$96Ko$	$768Ko$	$1.5Mo$	$2.3Mo$

Tableau 1.1- Tableau comparatif entre tailles des images

Cela explique la mémoire vidéo que nécessite votre carte graphique en fonction de la résolution de l'écran (nombre de points affichés) et du nombre de couleurs. On voit sur l'exemple qu.il faut une carte ayant 2 .3Mo de mémoire vidéo pour pouvoir afficher une résolution de 1024 × 768 en true color...

1-4-3 Résolution

C'est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'une image. La résolution est exprimée en unité de mesure (pouce ou centimètre) par nombre de pixels.

1-4-4 Histogramme

C'est une fonction qui donne la fréquence d'apparition de chaque niveau de gris ou couleur Il peut être utile dans les cas suivant :

_ Pour diminuer l'erreur de quantification ;

_ Pour comparer deux images obtenues sous des éclairages déférents ;

_ Pour améliorer certaine propriété d'une image (rehaussement) pour en tirer des informations.

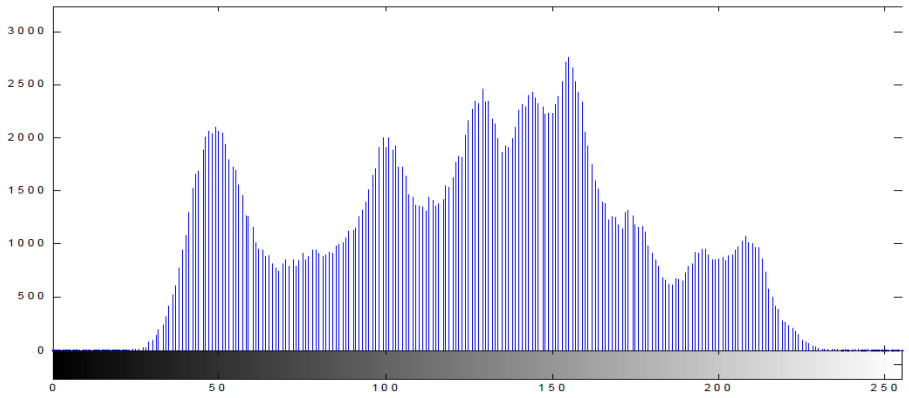


Figure 1.8 : Histogramme des niveaux de gris

1-4-5 Luminance

C'est la variation d'intensité lumineuse ou le degré de luminosité des points de l'image, une bonne luminance se caractérise par :

- Des images brillantes
- Un bon contraste
- L'absence de parasites

1-4-6 Niveau de gris

La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux intermédiaires , on peut donc dire que chaque pixel de l'image correspond à une quantité de lumière renvoyer comprise entre 0 et 255.

1-4-7 Bruit d'image

Un bruit dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins. Il provient de l'éclairage des dispositifs optiques et électroniques du capteur.

1-4-8 La transparence de l'image

La transparence est une caractéristique permettant de définir le niveau d'opacité des éléments d'une image, c'est-à-dire la possibilité de voir à travers l'image des éléments graphiques située derrière celle-ci. Il existe deux modes de transparence :

- La transparence simple s'applique pour une image indexée et consiste à définir parmi la palette de couleurs une des couleurs comme transparente.
- La transparence par couche alpha (ou canal alpha, en anglais alpha chanel) consiste à rajouter pour chaque pixel de l'image un octet définissant le niveau de transparence (0 à 255)
- Le processus consistant à ajouter une couche transparente à une image est généralement appelé alpha blending. **[1]**

I-2 Traitement Numérique d'image

2-1 Définition

C'est un ensemble des opérations relatives à la collecte, à l'enregistrement, à l'élaboration, à la modification, à l'édition, . . . des données.

Mettons de cotés les termes enregistrement et édition. Le principe général du traitement d'image est donc à quelques détails près toujours le même un système reçoit des images, et y applique un traitement, et produit une information de nature liée à l'application visée.

Une source de rayonnement envoie des ondes sur un objet, qui sont ensuite réfléchies et collectées par un capteur. Le capteur transforme ces ondes en un ensemble de points. Ces points sont traités et une information est produite en sortie du système. On peut résumer le traitement d'image en quatre étapes principales :

Acquisition des images

Mise en œuvre des processus physiques de formation des images suivis d'une mise en forme pour que ces images puissent être traitées par des systèmes informatiques.

Traitement des Images

Son but : améliorer ces images lorsqu'elles possèdent du bruit ou des défauts.

Segmentation des images

Son but : construire une image symbolique en générant des régions homogènes selon un critère défini à priori.

Analyse des images

Consiste à extraire des paramètres ou des fonctions représentatives de l'image ou des régions. **[2]**

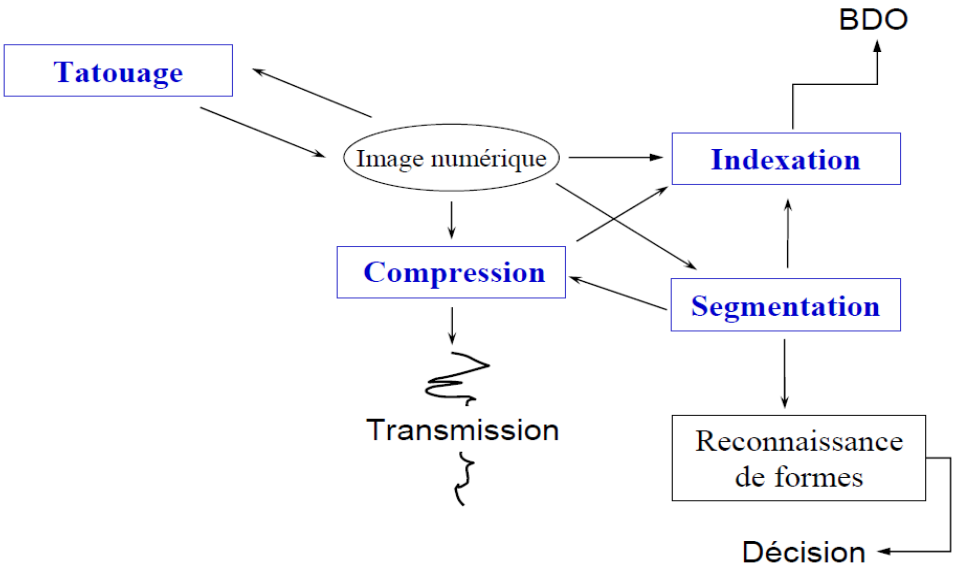


Figure 1.9 : Différents opérations de traitement Numérique

2-2 Les Avantages de traitement numérique

(Comparativement au traitement analogique) qu'on peut les citer sont les suivants :

- ✓ **flexibilité :**

Il est possible de reconfigurer ou de reprogrammer un système de traitement numérique, simplement en changeant des paramètres ou des coefficients dans un programme. Au contraire, modifier un système de traitement analogique du signal requiert de changer des composants physiques, ce qui inclut des procédures de design et de validation.

- ✓ **Précision :**

Une bonne précision peut être atteinte par le traitement numérique. Cette précision dépend de la résolution numérique du système numérique utilisé (16 bits, 32, bits), et de la résolution des convertisseurs A/N et N/A qui sont utilisés (8 bits, 12bits, 16 bits ou 18 bits). Par exemple, un filtre numérique ayant des caractéristiques fréquentielles **quasi**-idéales est facile à réaliser en temps discret. Au contraire, pour les systèmes de traitement analogique du signal, l'imprécision sur les composantes (par exemple sur des résistances et des condensateurs : 1% ou 5%) fait qu'il n'est pas possible d'avoir des filtres analogiques quasi-idéaux.

- ✓ **Capacité de réaliser des systèmes plus complexes :**

Réaliser un système numérique ayant une réponse désirée complexe est relativement facile, puisqu'il s'agit de trouver les bons coefficients et de les utiliser ensuite dans un programme ou dans un circuit numérique. Au contraire, réaliser une certaine réponse désirée complexe avec un système analogique en temps continu peut être beaucoup plus difficile. Certains algorithmes sophistiqués peuvent difficilement se mettre en œuvre en temps continu, et peuvent facilement se mettre en œuvre en temps discret. Le traitement numérique du signal a donc permis de réaliser des traitements qui n'étaient pas envisageables jusqu'à présent.

- ✓ **Mémorisation et entreposage de données :**

Les médias numériques (tels disques durs, CD-ROM, etc.) qui sont couramment utilisés par les ordinateurs modernes sont faits pour enregistrer l'information de signaux numériques et en temps discret, et non de signaux analogiques et en temps continu.

- ✓ **Transmission de données :**

Les réseaux de communication modernes (Internet, LAN, etc.) sont également conçus pour transmettre des données numériques et en temps discret.

- ✓ **Coût :**

Le coût des systèmes de traitement numérique diminue de plus en plus, à cause de la baisse des prix des circuits électroniques numériques. De plus, à cause du gain de flexibilité des systèmes, le coût de ces systèmes est parfois inférieur au coût d'un système de traitement analogique.

Ces avantages font en sorte que le traitement numérique est utilisé depuis 30 ans dans plusieurs domaines, dont certains sont :

- Traitement de la parole, de la musique et des images.
- Égalisation numérique de lignes téléphoniques et de canaux de télécommunications.
- Détection de séismes, de couches de pétrole, d'explosions nucléaires.
- Contrôle numérique adaptatif.

2-3 Différents opérations de traitement numérique

2-3-1 Filtrage des images numériques (Convolution bidimensionnelle)

Le filtrage est une opération qui a pour but d'extraire une information ou d'améliorer l'aspect de l'image, par exemple en éliminant un bruit (lignage, speckle des images radar, etc.) ou en améliorant les contours d'une image floue. Les filtres utilisent souvent la notion intuitive que des variations locales des teintes de gris résultent de bruits. On appelle filtrage adaptatif une opération de filtrage qui effectue une étape préalable de sélection des pixels. Ainsi, il peut être choisi de n'effectuer un filtrage que sur une partie d'une image et non sur la totalité de l'image.

Considérons une image monochrome dans laquelle la fonction f(i, j) représente l'intensité lumineuse du pixel de coordonnées (i, j). La convolution numérique de cette fonction avec une réponse impulsionnelle bidimensionnelle h(m ,n) conduit à une nouvelle image de fonction g(i, j). Cette convolution s'écrit :

$$g(i,j) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} h(m,n) f(i-m, j-n) \tag{1.6}$$

g(i, j) est la somme, pondérée par les coefficients h(m,n), des intensités des pixels appartenant à un voisinage du pixel de coordonnées (i, j). Le traitement est dit localisé. La réponse impulsionnelle h(m,n) est appelée masque de convolution.

Illustre le passage d'un masque de convolution sur une image numérique monochrome.

Le masque est déplacé dans toute l'image initiale pour obtenir une image traitée complète.

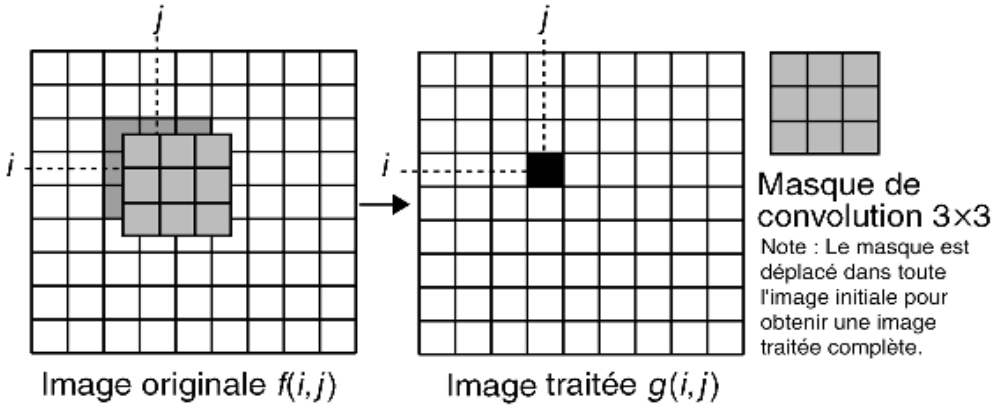


Figure 1.10 : Application d'un masque de convolution sur une image

2-3-1-1 Filtres linéaires

Les filtres linéaires effectuent des opérations linéaires sur des comptes numériques, qui en général sont ceux des pixels du voisinage du pixel traité. Ce sont donc des convolutions. Deux exemples sont présentés ici : "moyenne / lissage" d'images et "détection de frontières". L'extension spatiale des filtres est nécessairement limitée : c'est la dimension de l'opérateur de convolution. La récursivité permet de l'étendre. Un filtre est dit récursif si sa sortie yn dépend des entrées (xi où in) et sorties (yi où i<n) précédentes: yn = F(yi; {i<n}) + G(xj;{jn}). C'est le cas du filtre "yn = xn + a.yn-1" où |a|<1. aj.xn-j (i.e., image initiale convolée par un filtre d'étalement n tel que h(n)=an).

- **Moyenne / Lissage**

Le filtre "moyenne" calcule la moyenne locale en tout point de l'image (i.e., moyenne des comptes numériques des pixels du pixel traité. Son inconvénient majeur est d'élargir les frontières entre les différents objets de l'image, ce qui donne un aspect flou aux images. La taille et la forme du voisinage utilisé sont choisies selon la taille et forme des hétérogénéités (frontières,…) que l'on veut éliminer, mettre en évidence, etc. Avec une fenêtre 3 x 3, on a :

$$\frac{1}{9}\cdot\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}$$

Une caractéristique de ces filtres est qu'ils donnent des images dont certains comptes numériques peuvent ne pas être présents dans l'image initiale qui est traitée. Ainsi, l'application du filtre "moyenne" sur une image composée de deux zones juxtaposées avec des comptes numériques à 0 et 100 fait apparaître une zone frontière avec des comptes numériques compris entre 0 et 100.

- **Détection de Frontière**

Le choix d'un tel opérateur dépend de la forme, dimension des frontières :

Figure 1.11 : différents types de frontières

A-Opérateur "laplacien"

Cet opérateur de différentiation donne des comptes numériques d'autant plus élevés que leurs pixels correspondent à des hétérogénéités de l'image (e.g., zones frontières). Bien utilisé, il peut contribuer à améliorer la netteté d'une image floue. Son principe est qu'une hétérogénéité correspond à un passage à zéro du Laplacien (i.e., maximum de la dérivée).

L'approximation discrète la plus simple du Laplacien de fenêtre 3 x 3 est : ¼[0 -1 0 ;-1 4 -1 ;0 -1 0].

Ceci correspond à la différence entre le pixel et la moyenne de certains pixels de son voisinage. Par suite, le Laplacien est très similaire à l'opérateur "Ecart à la moyenne". Ainsi, dans le cas d'une fenêtre 3 x 3 :

$$\text{Laplacien} \approx \frac{1}{9}\cdot\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9}\cdot\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

A- Opérateur "gradient"

L'opérateur "gradient" est souvent utilisé pour créer l'image des frontières au sein d'une image. Son principe repose sur le calcul de gradients : l'image obtenue a des comptes numériques proportionnels au gradient local dans l'image de départ. Cet opérateur offre l'intérêt de donner le module et la direction de la dérivée, ce qui n'est pas le cas du Laplacien. La binarisation de l'image gradient obtenue fournit une image des frontières. Les exemples donnés ci-dessous correspondent à des gradients selon la verticale et l'horizontale.

On considère le plan Oxy et une fonction f (x,y). Le gradient de la fonction f (x,y) est le vecteur Grad f défini par :

$$\overrightarrow{\text{Grad}}\, f = \begin{bmatrix} \frac{\mathrm{d}f(x,y)}{\mathrm{d}x} \\ \frac{\mathrm{d}f(x,y)}{\mathrm{d}y} \end{bmatrix} \text{ soit } \|\overrightarrow{\text{Grad}}\, f\| = \sqrt{\left(\frac{\mathrm{d}f}{\mathrm{d}x}\right)^2 + \left(\frac{\mathrm{d}f}{\mathrm{d}y}\right)^2} \text{ et } \theta = \text{arctg}\left(\frac{\frac{\mathrm{d}f}{\mathrm{d}y}}{\frac{\mathrm{d}f}{\mathrm{d}x}}\right) \qquad (1.7)$$

On note Gradx (f) = d f/dx et Grady(f) = d f/dy , les gradients partiels de la fonction f.

L'obtention du gradient d'une image nécessite l'utilisation de masques de convolution qui permettent l'estimation des gradients partiels. Des opérations non linéaires fournissent ensuite le module et l'orientation du gradient au pixel considéré. Le gradient permet la détection des transitions d'une image et l'orientation de celles-ci ; en effet la direction du gradient est normale au contour en un point considéré.

Les différents types de masques de convolution utilisables pour l'obtention du gradient d'une image sont décrits dans l'exemple (Tableau 1.2), nous utilisons les masques de Sobel pour obtenir l'amplitude du gradient. La dynamique de l'amplitude est recadrée dans la gamme [0,255] afin de permettre une visualisation en niveau de gris.

Types de masque	Gradients partiels	Extraction de l'amplitude	Extraction de la direction
Masques de Roberts $\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}$ $\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}$	$G_1; G_2$ Substitution du pixel supérieur gauche	$A = \sqrt{G_1^2 + G_2^2}$	$\theta = \frac{\pi}{4} + \arctan\left(\frac{G_2}{G_1}\right)$
Masques de Sobel $\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}$ $\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$	$G_x; G_y$	$A = \sqrt{G_x^2 + G_y^2}$	$\theta = \arctan\left(\frac{G_y}{G_x}\right)$
Masques de Prewitt $\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix}$ $\begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix}$	$G_x; G_y$	$A = \sqrt{G_x^2 + G_y^2}$	$\theta = \arctan\left(\frac{G_y}{G_x}\right)$
Masques de Kirsh $\begin{bmatrix}5 & 5 & 5\\ -3 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}$ + les 7 autres masques obtenus par permutation circulaire des coefficients	G_i pour i de 1 à 8	Maximum des G_i	Direction correspondant au G_i sélectionné
Masques de Robinson $\begin{bmatrix}1 & 1 & 1\\ 1 & -2 & 1\\ -1 & -1 & -1\end{bmatrix}$ + les 7 autres masques obtenus par permutation circulaire des coefficients	G_i pour i de 1 à 8	Maximum des G_i	Direction correspondant au G_i sélectionné
Laplacien discret $\begin{bmatrix}1 & 1 & 1\\ 1 & -8 & 1\\ 1 & 1 & 1\end{bmatrix}$	L	$\lvert L\rvert$	

Tableau 1.2 : Différents type de masques de convolution

La détection de contours est une opération très fréquente en analyse et traitement des images. On trouve la détection de contours dans des applications comme :

– le comptage et l'étiquetage des objets d'une image ;

– la détection de caractéristiques géométriques intra image ;

– la recherche de relations structurelles inter pixel; Réalisée généralement en début d'une chaîne de traitement et suivie d'une binarisation, cette étape réduit considérablement le nombre de pixels à traiter par la suite. **[2]**

Figure 1.12 : Détection de contours d'une image par convolution de type Sobel

2-3-1-2 Filtres non linéaires

Ces filtres donnent un compte numérique qui dépend de l'ordre des valeurs des pixels du voisinage du pixel traité P(x, y). Il existe plusieurs types de filtres médians. Deux exemples sont donnés ici.

- Filtre médian "moyenne" : le compte numérique de P(x, y) est transformé en une valeur telle que dans le voisinage de P(x, y) il y ait autant de pixels supérieurs et inférieurs à cette valeur.

- Filtre médian "classement" : soit p le nombre de comptes numériques différents présents dans le voisinage de P(x, y). Ce filtre transforme le compte numérique de P(x, y) en une valeur égale au même compte numérique parmi les p comptes numériques du voisinage.

Comme les filtres morphologiques, et contrairement aux filtres linéaires, les filtres médians ne donnent pas de nouveaux comptes numériques. Cette propriété permet de préserver les frontières. Ils sont robustes, car la présence d'un faible nombre de pixels associés à du bruit affecte peu ou pas l'image "résultat". Un de leurs inconvénients est de nécessiter des temps de calcul supérieurs à ceux des filtres linéaires, en raison des nombreux tests informatiques obligatoires.

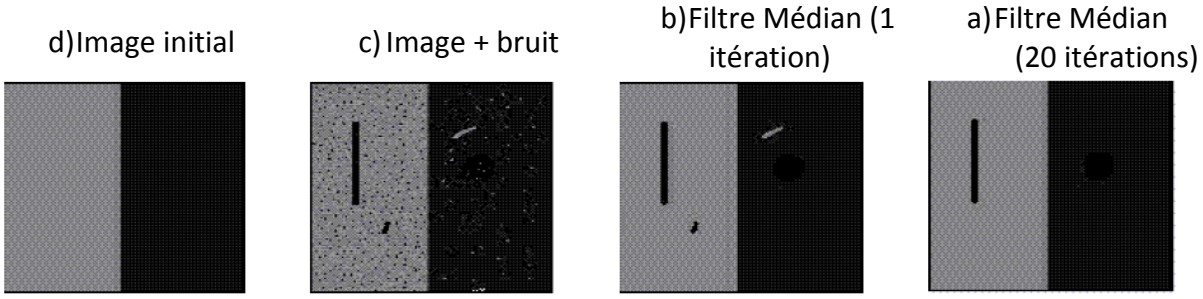


Figure 1.13 : Elimination du bruit avec un Filtre Médian

2-3-1-3 Filtres Adaptatifs

Ils combinent deux opérations : détermination pour chaque pixel d'un paramètre (*e.g.*, variance) qui indique si ce pixel appartient à une frontière, puis application d'un opérateur de lissage, en tenant compte de l'appartenance ou non du pixel traité à une frontière. Cette approche réduit le risque de rendre floues les frontières de l'image. Elle peut rendre plus homogène des zones plutôt homogènes, tout en préservant les frontières entre ces zones.

Ces filtres permettent d'améliorer le contraste en préservant les contours des objets, de rendre les statistiques locales beaucoup plus homogènes, etc. Ainsi, on peut transformer l'image de manière à ce que toutes les zones homogènes aient la même moyenne et la même variance.
Exemple : Filtre Nagao. Il transforme le compte numérique d'un pixel par la moyenne des comptes numériques des pixels d'un "sous voisinage" de son "voisinage" (e.g., fenêtre 3x3 dans une fenêtre 5x5. Ce "sous voisinage" est celui qui a la plus petite variance. **[1]**

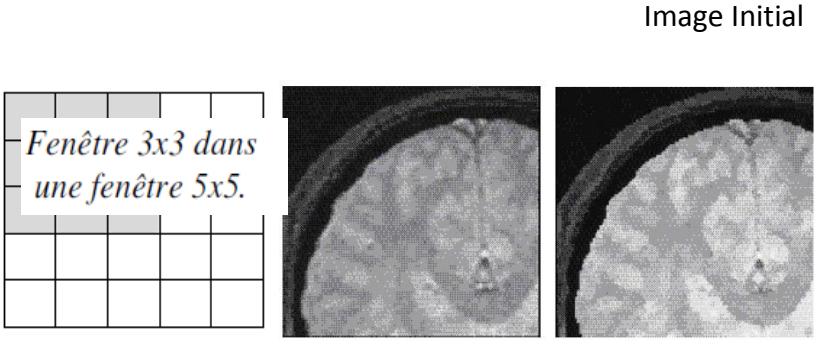


Figure 1.14: Filtre Nagao

2-3-2 Segmentation des images

2-3-2-1 Définition de la segmentation

La segmentation d'image est un traitement de bas- niveau et une des étapes critiques de l'analyse d'images qui a pour but de rassembler des pixels entre eux suivant des critères prédéfinis. Chaque groupe de pixels forme alors une région. Une région est donc un ensemble connexe de pixels ayant des propriétés communes (intensité, texture,…) qui les différencient des pixels des régions voisines. Si le nombre de régions est égal à deux, la segmentation est appelée aussi binarisation. De façon plus précise, on peut définir la segmentation comme étant une partition d'une image I en n ensembles Ri appelés régions tels que :

$$
\begin{aligned}
&1)\ \cup^{n}_{i=1} C_i = I \\
&2)\ \forall i,j \in \{1,\dots n\}^2 \text{ et } i \neq j \Rightarrow C_i \cap C_j = \emptyset \\
&3)\ \forall i \in \{1,\dots n\}^2\ C_i \text{ est connexe} \\
&4)\ \forall i,j \in \{1,\dots n\}^2\ P_{(C_i)} = \text{Vrai}, \\
&5)\ \forall i,j \in \{1,\dots n\}^2\ C_i \text{ est adjacent à } C_j \text{ et } i \neq j \Rightarrow P(C_i \cup C_j) = \text{faux}
\end{aligned}
\qquad (1.8)
$$

Où le prédicat P est utilisé pour tester l'homogénéité.

La première condition signifie que l'image I est partitionnée en n classes. Et la deuxième explique que tous les classes sont disjointes deux à deux.

La troisième étape, la quatrième et la cinquième imposent à chaque pixel d'une classe de satisfaire à la même propriété au sens du prédicat P. Le prédicat P n'est plus vrai pour la réunion de deux classes adjacentes. Donc, la segmentation permettant une analyse de données en regroupant les pixels formants l'image à classifier de telle sorte les pixels appartenant à une classe soient plus similaires entre eux que ceux des classes différents, on affecter à chaque entité de la scène traité une étiquette indiquant son appartenance à une classe particulière. L'entité utilisée caractérise généralement un pixel ou un ensemble de pixel, tandis que l'étiquette constitue un thème choisi par l'utilisateur.

Il n'y a pas de méthode unique de segmentation d'une image, le choix d'une technique est lié :

- **A la nature de l'image :**

- éclairage non homogène, reflets,
- présence de bruit, de zones texturées,
- contours flous, en partie occultés,

- **Aux opérations situées en aval de la segmentation :**

- localisation, mesure, calcul 3D,
- reconnaissance des formes, interprétation,
- diagnostic, contrôle qualité,

- **Aux primitives à extraire :**

- contours, segments de droite, angles,…
- régions, formes,
- textures,

- **Aux contraintes d'exploitation :**

- complexité algorithmique, fonctionnement en temps réel,
- taille de la mémoire disponible en machine.

2-3-2-2 Les différentes approches de segmentation

Essentiellement, l'analyse d'images a pour but l'extraction de l'information caractéristique contenue dans une image. Cette information peut prendre la forme, la couleur, le contour, la texture,... Donc, il est nécessaire de procéder tout d'abord à la segmentation de la lésion. Cette segmentation est généralement abordée selon deux modèles complémentaires : l'approche région et l'approche frontière.

L'approche frontière regroupe les techniques associées à une variation d'intensité : détection de contours. Cependant, les contours obtenus sont rarement connexes, donc elle doit faire face au problème de fermeture de contours.

L'approche région considère des groupements de pixels ayant des propriétés communes, chaque pixel est affecté à une région unique après une partition de l'image.

Toutefois, il n'existe pas d'algorithme spécifique pour déterminer les régions et les frontières dans une image. D'ailleurs les chercheurs ont compromis que la segmentation idéale n'existe pas, la bonne technique de segmentation sera donc celle qui permettra d'arriver à une bonne interprétation.

Une panoplie de techniques est proposée dans la littérature ou chacune présente des avantages et des inconvénients. Parmi les nombreuses approches de segmentation d'images qui existent, nous présentons quel que techniques les plus citées dans la littérature appliquées à la mammographie, à savoir les méthodes basées sur pixels, l'approche région ainsi que celles basées sur les contours.

2-3-2-2-1 Les Segmentation basé sur les contours

La détection de contour est souvent le premier problème qu'on rencontre en traitant une image. La difficulté augmente avec l'importance de bruit présent.

- Définition (Un Contour) :

Un contour peut approximativement être défini comme une frontière entre deux régions ou l'intensité des pixels change brusquement.

Généralement l'utilisation d'un tel operateur de contour se combine avec un seuillage et comme étant ce dernier est généralement imparfait, on obtient, d'une part, des contours qui ne limitent pas les régions fermées. Donc on doit faire recours à des algorithmes de fermetures des contours. D'un autre part, les zones de fortes variations ne correspondent pas forcement a un contour d'objet. Alors un post-traitement est nécessaire pour analyser les différents contours obtenus.

Méthodes des contours actifs:

L'idée de cette méthode est de déplacer les points pour les rapprocher des zones de fort radient tout en conservant des caractéristiques comme la courbure du contour ou la répartition des points sur le contour ou d'autres contraintes liées à la disposition des points.

Au démarrage de l'algorithme, le contour est dispose uniformément autour de l'objet a détourer puis il va se rétracter pour en épouser au mieux la forme. A chaque itération, l'algorithme va tenter de trouver un meilleur positionnement pour le contour pour minimiser les dérives par rapport aux contraintes utilisées.

L'algorithme s'arrêtera lorsqu'il ne sera plus possible d'améliorer le positionnement ou simplement quand le nombre maximum d'itérations aura été atteint. On utilise les notions d'énergies interne et externe pour caractériser respectivement la forme du contour et son positionnement sur l'image en tenant compte des lignes de gradient.

2-3-2-2-2 Segmentation par région

L'approche par régions consiste à regrouper des points selon des propriétés communes.

- Définition (segmentation par région):

Globalement, elle peut être définie comme une partition d'une image I en une ou plusieurs régions R1, …, Rn telles que :

$$I = \bigcup_{i=1}^{n} R_i \quad et \quad R_i \cap R_j = \varnothing \quad pour \quad i \neq j \tag{1.9}$$

- Méthode par croissance de régions:

Le principe de ces méthodes est de réunir de façon itérative un ensemble de points connectes en une région de plus en plus large, en fonction de critères d'homogénéité. Pour définir une région, on définit un *germe* dans la région d'intérêt qui sert comme un point de départ pour l'agrégation.

Cependant un inconvénient à marquer a cette méthode c'est qu'elle est récursive: risque de débordements (pile).

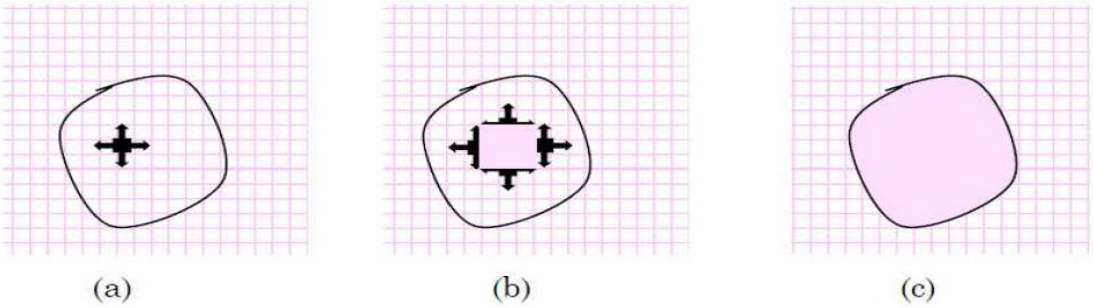


Figure 1.15: Le processus d'un algorithme de croissance de région

(a) début de processus, le point noir présente le germe et les flèches la direction de croissance. (b) la croissance de pixels après quelques itérations.

(c) le résultat de la segmentation.

- K-means:

Cette méthode consiste à rassembler les pixels en K groupes (clusters). K étant un paramètre préfixé qui détermine le nombre de régions. Et après avoir déterminé les paramètres (couleur par exemple) de toute région, chaque point est affecté au centre le plus proche, pour recalculer ensuite le paramètre de chaque région, jusqu'à ce qu'elles soient stables.

- **Algorithme K-means:**

L'algorithme des K-Means est l'une des techniques de clustering non supervisée les plus utilisées. La méthode consiste à placer aléatoirement dans l'espace K "centroides" afin de déterminer K clusters. On affecte à chacun de ces "centroides" les objets les plus proches, puis on calcule la position moyenne des objets associes aux "centroides" que l'on déplace en ce point. Les opérations d'affectation d'objet et de déplacement du centroides sur la moyenne repérées jusqu'a ce que chaque centroide ait atteint une position stable.

Malgré sa simplicité, cet algorithme se révèle efficace. Toutefois il nécessaire de prédéterminer le nombre de catégories et la position de départ des prototypes qui a un impact sur le découpage en classes. les principales étapes de cette algorithme sont:

- Choix aléatoire de la position initiale des K clusters.
- (Ré-)Affecter les objets à un cluster suivant un critère de minimisation des distances (généralement selon une mesure de distance euclidienne).
- Une fois tous les objets placés, recalculer les K centroïdes.
- Réitérer les étapes 2 et 3 jusqu'à ce que plus aucune ré-affectation ne soit faite.

Le principal avantage de cette méthode est que le nombre de régions est connu au préalable (il n'y aura pas de problème de sur-segmentation ou sous-segmentation), mais l'inconvénient réside en la difficulté de déterminer le nombre de clusters et l'incohérence des régions (deux objets éloignés peuvent appartenir à la même région).

2-3-2-2-3 Les Segmentation basé sur les pixels

La segmentation basée sur le pixel travaille sur des histogrammes de l'image, on citera essentiellement la méthode du seuillage. La segmentation est inhérente puisque les régions sont déduites du seuillage. Le seuillage est une méthode faisant partie de l'approche basée sur

le pixel, même si d'autres références classent cette méthode dans l'approche région de la segmentation, du fait qu'elle met en évidence les régions d'une image.

Un seuil est une valeur numérique correspondant à un paramètre de l'image (exemple : le niveau de gris), et l'histogramme des niveaux de gris est utilisé afin d'en déduire le seuil.

Le seuillage peut être :

-**Global** : un seuil pour toute l'image

-**Local** : un seuil pour une portion de l'image

-**Adaptatif** : un seuil qui s'ajuste selon les images, ou parties de l'image

Le seuillage le plus simple est le seuillage global, il a pour principe : Soit p un pixel et *V* une fonction définissant le niveau de gris et S un seuil :

$$V(p)=\begin{cases}0 \text{ si } V(p)<S\\ 1 \text{ si } V(p)\geq S\end{cases}$$

Le seuillage global est certes rapide, mais il ne donne pas de pixels contigus, et le bruit peut donner de faux éléments.Le seuillage local détermine pour chaque pixel un seuil en fonction de la luminosité de son voisinage.

Le seuillage adaptatif se résume en la séparation de l'image en sous images et de traiter chacune avec son propre seuil (Le choix de la dimension des sous-images est critique).

Dans tous les cas, le problème majeur reste la détermination du seuil, il existe quelques méthodes de détermination telles que :

- La recherche des vallées de l'histogramme, en considérant que les vallées significatives correspondent aux pixels de la frontière des régions.- La segmentation de l'histogramme (développée par Otsu) qui ne s'applique que dans le cas de segmentation de l'image en deux régions. **[2]**

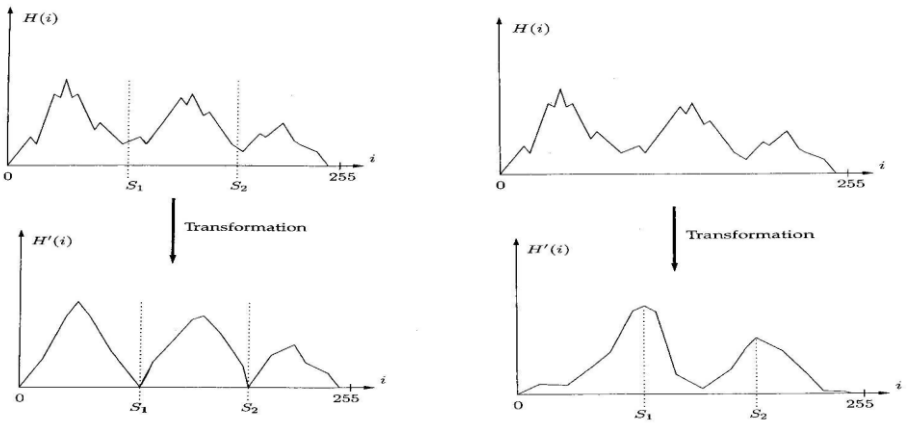


Figure 1.16 : Exemple de détermination du seuil

2-3-4 Reconnaissance

La reconnaissance optique de caractères (en anglais optical character recognition : OCR), Cette méthode désigne les procédés informatiques pour la traduction d'images de textes imprimés ou dactylographiés en fichiers de texte..

La technique d'OCR (*optical character recognition*) permet de situer et de reconnaître les chaînes de caractères dans une image et donc d'opérer la conversion des mots qui peuvent ensuite être utilisés pour différents usages..

Cette conversion est assurée automatiquement par un un algorithme approprié et fait l'économie de la retranscription manuelle des contenus. Même si les techniques d'OCR sont en progrès constant, la qualité de reconnaissance dépend malgré tout d'un grand nombre de facteurs liés tant au document original qu'à la numérisation elle-même. De de fait, un certain nombre de défis fait face à l'OCR tels que la dégradation du papier ou de l'encrage, polices de caractères ou orthographes anciennes, etc. De plus, les anciens modes de numérisation (en noir et blanc, d'après microfilm) ont un impact négatif sur les performances.

Conclusion

Cette étude n'a fait qu'effleurer l'immense liste des traitements que l'on peut faire subir à une image. Le traitement Numérique des images est un domaine très actif, où les avancées théoriques se concrétisent sous la forme d'algorithmes rapides et puissant de calcul qui ont des applications importantes pour la manipulation des contenus numériques.

2

CHAPITRE :

Généralités sur le TMS 320C6713

Introduction

L'objet de cette partie est d'étudier le DSP TMS320C6713. Néanmoins les informations fournies ne sont pas exhaustives et, pour une utilisation précise de telle ou telle partie, on pourra avantageusement utiliser les documentations fournies par le constructeur pour mieux détaillera ce chapitre.

II-1 La Famille TMS 320C

Cette famille de DSP englobe des processeurs 16-32 bits, à points fixes ou flottants. Aujourd'hui, la famille TMS320 est divisée en trois Plates-formes qui sont les TMS320c2000, TMS320c5000, TMS320c6000. cette dernière englobe trois type de processeur, les TMS320c62x, TMS320c64x, TMs320c67x. On peut voir sur la figure 2.1 les différents types de processeur de cette famille (TMS320). **[3]**

Application		*Type de DSP*	*Caractéristiques*
TMS320C6000 DSP hautes performances			
C62x	Applications exigeantes en vitesse : stations de base des réseaux de communications mobiles, équipement de radiodiffusion, réseaux informatiques	16 bits virgule fixe architecture VLIW	1200-2400 MIPS
C67x	Applications exigeantes en précision, dynamique et vitesse :Antennes adaptatives des stations de base, imagerie médicale, reconnaissance de parole, graphisme …	32 bits virgule flottante architecture VLIW	600MFLOPS-1GFLOPS
TMS320C5000 DSP optimisés en consommation			
C54x	Applications de télécommunications exigeantes en coût, consommation, vitesse : terminaux mobile, voix sur IP, alphapages …	16 bits virgule fixe	0,54 mW/MIPS 30-200 MIPS
TMS320C2000 DSP optimisés pour les applications de contrôle			
C20x	Applications de grands volume en téléphonie, électronique grand public, appareils photos numériques ou contrôleurs de disques durs ….	16 bits virgule fixe	PLL, UART, timers, mémoire flash intégrée 20-40MIPS
C24x	Applications de contrôle moteur, automatisation, robotique, contrôle d'appareils électroménagers… Bon compromis prix/performance	16 bits virgule fixe	Port série SCI, SPI et CAN, Convertisseur Analogique /numérique 20MIPS

Tableau 2.1 : Principales caractéristiques des 3 plates-formes de DSP

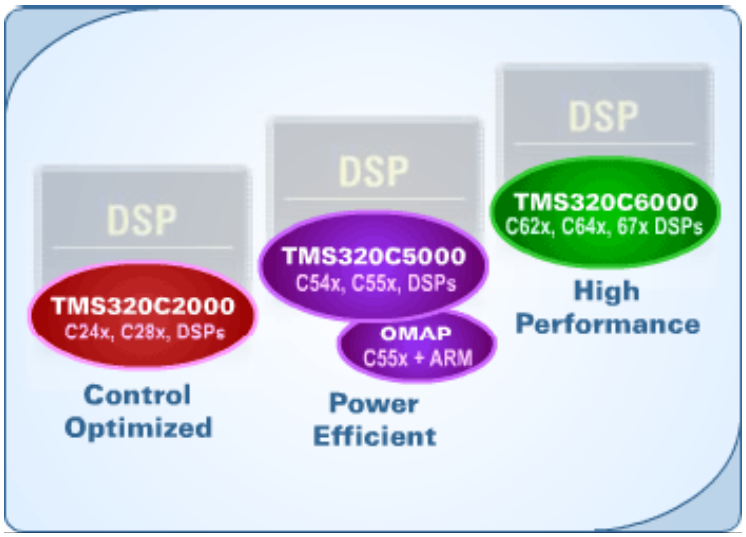


Figure 2.1 : La famille TMS320C

II-2 Les différentes familles de C6x

2-1 TMS320C62X :

Ce sont les premiers C6x sortis par Texas Instruments en 1998. Avec des fréquences allant de 150 à 300 MHz, ils ont été les premiers DSP à architecture VLIW (very-long-instruction-word).

2-2 TMS320C64X :

Ce sont les derniers DSP de gamme proposée par Texas Instruments. Comme les C62x, ce sont des DSP 16 bits virgule fixe. L'avantage de cette famille est qu'elle bénéficie des progrès technologiques et peut atteindre des fréquences allant de 300 à 600 MHz. Des efforts ont également été réalisés pour les transferts de données dans le processeur avec des bus 64 bits. Le jeu d'instructions a été étoffé par rapport à celui du C62x avec des instructions SIMD (single-instruction multiple-data) pour pouvoir traiter plusieurs données avec une unique instruction et ainsi diminuer la taille du programme. Ces instructions ont été spécialement ajoutées pour les applications de traitement d'images et pour l'utilisation du bus interne 64 bits.

2-3 TMS320C67X :

Ce sont les DSP à virgule flottante de la gamme. Les données sont de 64 bits, et les bus internes également. Les fréquences proposées vont de 100 à 225 MHz. Le jeu d'instructions est identique à celui du C62x, auquel ont été ajoutées des instructions pour les calculs spécifiques en virgules flottantes. Actuellement, aucune nouvelle version n'est annoncée. **[4]**

II-3 Présentation de la plateforme TMS 320C6000

Avec une performance jusqu'à 4800 millions instructions par second (mips) et un compilateur C effectif, les TMS320C6000 DSP donnent à l'architecte du système, une possibilité illimitée de différencier son produit d'un autre. La Haute performance, et les prix accessibles, font de la plateforme TMS320C6000 l'idéale solution pour multicanal, demande multifonction tel que:

- Stations de base de la ligne locale sans fil.
- Serveurs à accès à distance (RAS).
- Ligne d'abonnement Numérique (DSL) système.
- Modems câblés.
- Systèmes de la téléphonie multivoies.

La plateforme TMS320C6000 est aussi une solution idéale pour exciter des nouvelles applications, par exemple:

- Protection de maison personnalisée avec type de caractère et reconnaissance de main ''empreinte digitale''.
- Contrôle de la croisière avancée avec navigation GPS et évasion de l'accident.
- Diagnostics médicaux éloignés.
- Faisceau-formant stations de base.
- Reconnaissance vocale.
- Audio.
- Radar.
- Modelage atmosphérique. **[3]**

II-4 Caractéristiques principales du TMS 320C6713

Le C6713 DSK est une plateforme du développement autonome bas-prix qui permet aux utilisateurs, d'évaluer et développer des applications pour le TI C67xx DSP famille. Le DSK sert aussi comme hardware référence du matériel pour le TMS320C6713 DSP. **[6]**

Dans notre cas, le DSP est monté sur une carte Starter Kit de Texas Instrument munie des synchrones :

- 512 Kbytes de mémoire non éléments suivants.
- Le TMS320C6713 DSP fonctionnant à 225 MHZ.

- Un convertisseur audio bidirectionnel de type « CODEC » AIC23 stéréo.
- 8 Mbytes de mémoire DRAM volatile Flash.
- Le software embarqué d'auto-configuration de la carte sur un asic programmable de type CPLD.
- Quelques interfaces adressables telles que :
- 4 LEDs et des switches DIP.
- des ports standard pour cartes filles.
- un émulateur JTAG embarqué.
- une interface USB avec l'ordinateur hôte.
- − L'alimentation est externe mais très simple (+5V). **[4]**

II-5 Architecture générale du starter KIT TMS 320C6713

5-1 Le Processeur :

Les **TMS320C6713 DSP** composent la génération à point mobile de DSP dans la plateforme de TMS320C6000™ DSP. Le dispositif C6713, basé sur l'architecture à rendement élevé et avancée du VLIW ‘’très-long-instruction-mot ‘’ ‘’very-long-instruction-word‘’, et développé par Texas Instruments fait de ce DSP un excellent choix pour les applications multicanaux et multifonctionnelle. Fonctionnant à 225 mégahertz, le C6713 fournit jusqu'à 1350 millions d'opérations à point mobile par seconde (MFLOPS), 1800 millions d'instructions par seconde (MIPS), et avec fixed-/floating-point fournit aux multiplicateurs jusqu'à 450 millions ‘’multiplier-accumuler ‘’opérations par seconde (MMACS). **[6]**

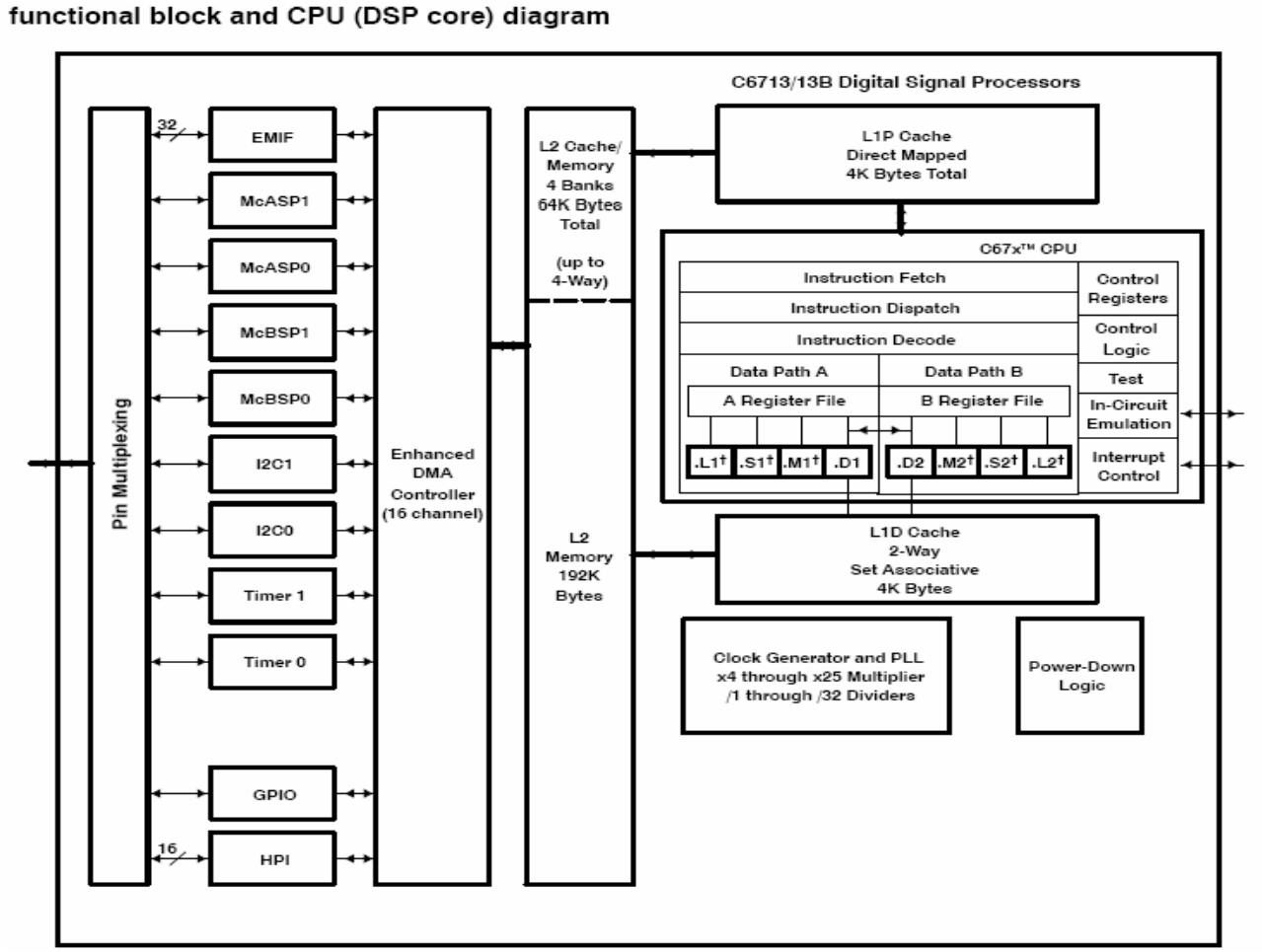


Figure 2.2 : Architecture de TMS320C6713 DSP

Le C6713 emploie une architecture cachette-basée à deux niveaux, et a un ensemble puissant de divers périphériques. La cachette de programme du niveau 1 (L1P) est une cachette ‘’dirigé-tracé‘’par 4K-Byte, et la cachette de données du niveau 1(L1D) est une cachette 4K-Byte placer-associative bidirectionnelle. La mémoire du niveau 2 ou la cachette (L2) se compose d'un espace mémoire 256K-Byte, partagé entre le programme et l'espace de données. Les 64K bytes des 256K bytes dans la mémoire L2 peuvent être configurés en tant que mémoire, cachette, ou combinaison tracée des deux. Les bytes 192K restant dans L2 sert de SRAM tracé. Le C6713 possède un périphérique riche qui inclut :

✓ Deux (McASP) ‘portes série audio multicanales ‘ ‘multichannel audio serial port’.

✓ Deux portes série protégées multicanales (McBSP) ‘’multichannel buffered serial port‘’.

✓ deux ‘’Inter-Intégrés - circuit‘’les bus (I2C),

- ✓ un module à usage universel consacré d'entrée-sortie (GPIO) ‘’ General-Purpose-Input/Output ‘’.
- ✓ Deux temporisateurs à usage universel, une interface de centre-port (HPI) ‘’ Host Port Interface ‘’.
- ✓ Une interface externe glueless mémoire (EMIF) ‘’External Memory Interface ‘’ incluant SDRAM, SBSRAM, et périphériques asynchrones. La porte série soutient le multiplexage temporel sur chaque goupille de 2 à 32 fentes de temps.

Le C6713 a une largeur de bande suffisante pour soutenir chacune des 16 goupilles périodiques de données transmettant un signal stéréo de 192 kilohertz. Des données périodiques dans chaque zone peuvent être transmises et reçues sur les goupilles périodiques multiples de données simultanément et être composées dans une multitude de variations sur le format du bruit d'Inter-IC de Philips (I2S). En outre, l'émetteur de McASP peut être programmé pour produire S/PDIF multiple, IEC60958, AES-3, et les canaux de données codés par CP-430 simultanément, avec une RAM simple contenant la pleine exécution des champs de statut de données et de canal d'utilisateur.

Le McASP fournit également les dispositifs étendus de contrôle d'erreurs et de rétablissement, tels que le mauvais circuit de détection d'horloge pour chaque horloge principale à haute fréquence et qui vérifie que l'horloge principale est dans une marge de fréquence programmée.

Les deux ports d'I2C sur le TMS320C6713 permettent au DSP de commander facilement les périphériques et de communiquer avec un processeur du centre serveur. En outre, la porte série protégée multicanal standard (McBSP) peut être employée pour communiquer avec les périphériques de mode périphérique périodique de l'interface (SPI). Le dispositif TMS320C6713 a deux boot modes, du HPI ou de la ROM asynchrone externe.

La génération de TMS320C67x DSP est soutenue par l'ensemble d'expresses™ de TI, outil de développement de repère d'industrie, y compris un compilateur fort de linéarisation de C/C++. L’environnement de développement intégré par ‘’Studio™ de compositeur de code’’ (IDE), ‘’Studio Code Composer’’ (CCS), émulateur JTAG-basée et correction en temps réel, et le grain de DSP/BIOS™.**[5]**

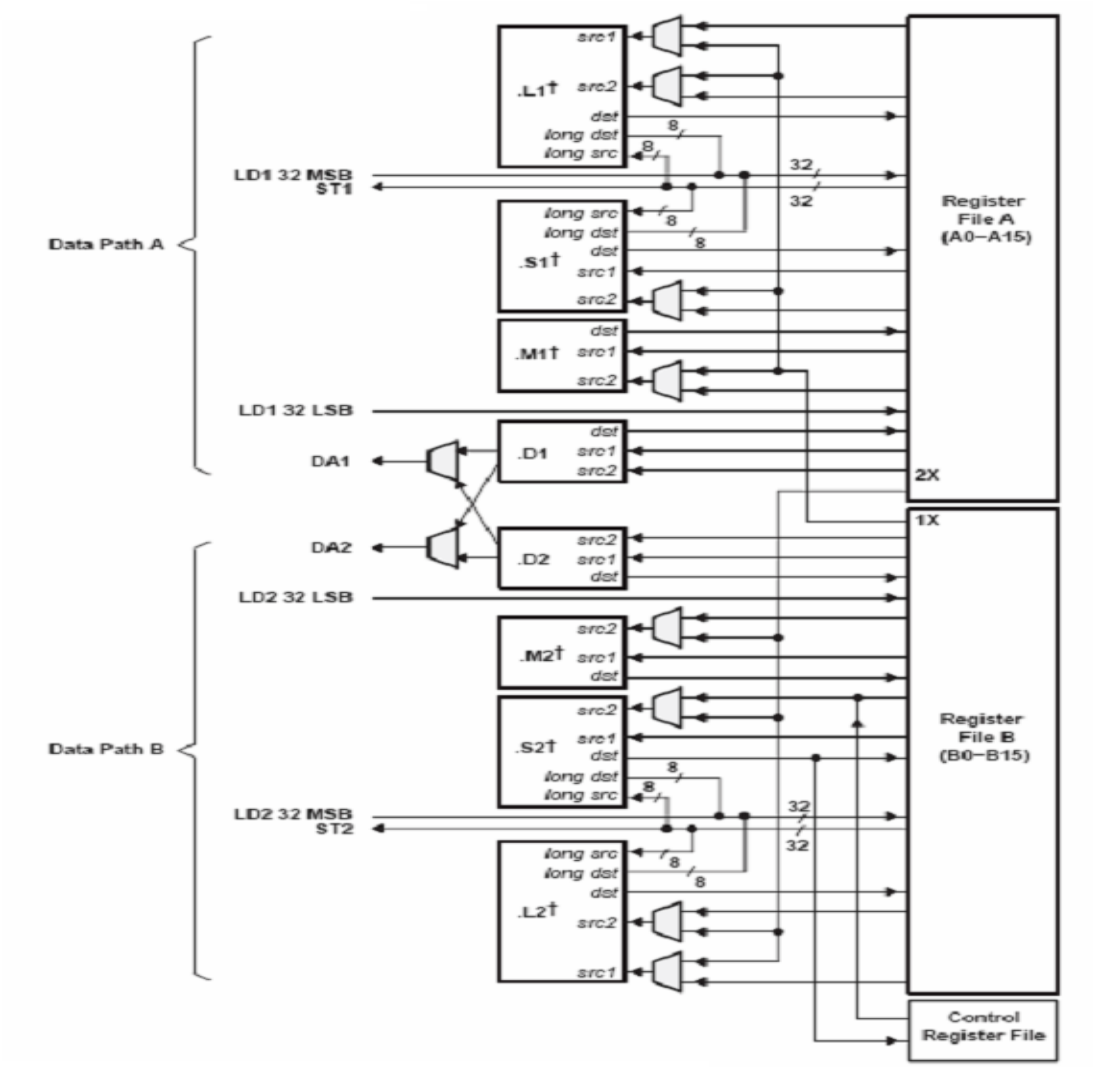


Figure 2.3: TMS320C6713 CPU Data path

La figure montre les deux « data paths » (vois accès de donnée) du TMS320c67 et apparaissent :

- Les deux files registrent (A et B).
- Les huit unités (L1, L2, M1, M2, S1, S2, D1, et D2).
- Les deux voies de chargement depuis la mémoire LD1 et LD2 (LD=load).
- Les deux voies de stockage vers la mémoire ST1 et ST2 (ST=store).
- Deux croisements (registre file cross paths) entre les files de registre 1X et 2X.
- Deux chemins d'adresse de données (data adresse paths) DA1 etDA2

5-1-1 Cartographie de mémoire :

La famille TMS 320c67xx a un grand espace d'adressage adressable. Le code de programme et de données peut être placé n'importe où dans l'espace d'adressage unifié. L'EMIF a 4 régions d'adresse séparées (CE0-CE3) .

La SDRAM occupe CE0, le flash CE1, le CPLD CE2 et CE3 sont généralement réservés pour le daughtercards '' la carte fille ''.

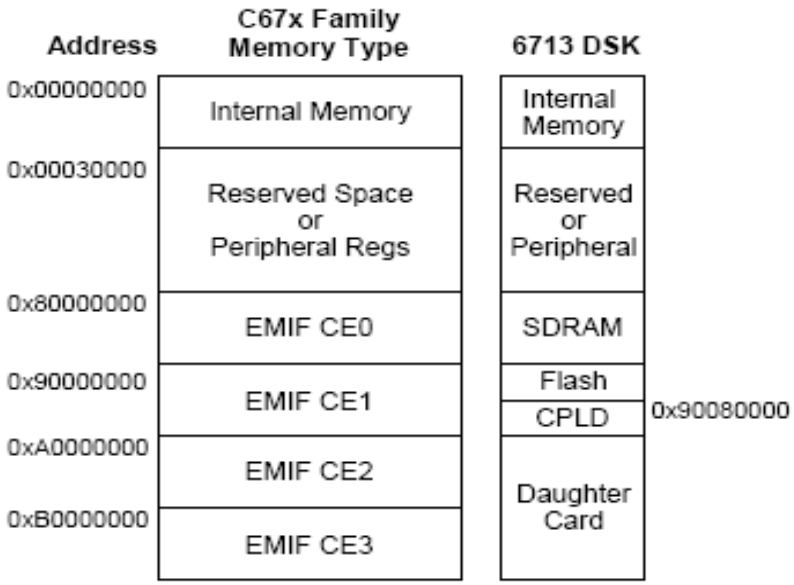


Figure 2.4 : Schéma de cartographie de mémoire

Les banques de mémoire indépendantes sur la C6x permettent deux accès de mémoire dans un cycle d'instruction. Deux banques de mémoire indépendantes sont accessibles via deux bus indépendants. Puisque la mémoire interne est organisée en banques de mémoire, deux chargements ou deux stockages d'instructions peuvent être exécutées en parallèle. Aucun conflit ne se produira si les données accédées sont dans différentes banques de mémoire.

Les bus séparés pour le programme, les données et l'accès direct à la mémoire (DMA) permettent au processeur C6x d'effectuer en parallèle, les opérations « program fetch », lecture et écriture de données, et DMA. Avec les données et les instructions se trouvant dans des espaces mémoire séparés, l'accès aux mémoires simultanément est possible.

L'espace mémoire du C6x est adressable par octet. La mémoire interne est organisée en deux espaces distincts : un pour le programme et l'autre pour les données, avec deux ports internes de 32 bits pour accéder à la mémoire interne. **[5]**

Memory Block Description	Block Size (Bytes)	Hex Address Range
Internal RAM (L2)	192K	0000 0000–0002 FFFF
Internal RAM/cache	64K	0003 0000–0003 FFFF
Reserved	24M–256K	0004 0000–017F FFFF
External memory interface (EMIF)	256K	0180 0000–0183 FFFF
registers	128K	0184 0000–0185 FFFF
L2 registers	128K	0186 0000–0187 FFFF
Reserved	256K	0188 0000–018B FFFF
HPI registers	256K	018C 0000–018F FFFF
McBSP 0 registers	256K	0190 0000–0193 FFFF
McBSP 1 registers	256K	0194 0000–0197 FFFF
Timer 0 registers	256K	0198 0000–019B FFFF
Timer 1 registers	512	019C 0000–019C 01FF
Interrupt selector registers	4	019C 0200–019C 0203
Device configuration registers	256K–516	091C 0204–019F FFFF
Reserved	256K	01A0 0000–01A3 FFFF
EDMA RAM and EDMA registers	768K	01A4 0000–01AF FFFF
Reserved	16K	01B0 0000–01B0 3FFF
GPIO registers	240K	01B0 4000–01B3 FFFF
Reserved	16K	01B4 0000–01B4 3FFF
I2C0 registers	16K	01B4 4000–01B4 7FFF
I2C1 registers	16K	01B4 8000–01B4 BFFF
Reserved	16K	01B4 C000–01B4 FFFF
McASP0 registers	16K	01B5 0000–01B5 3FFF
McASP1 registers	160K	01B5 4000–01B7 BFFF
Reserved	8K	01B7 C000–01B7 DFFF
PLL registers	264K	01B7 E000–01BB FFFF
Reserved	256K	01BC 0000–01BF FFFF
Emulation registers	4M	01C0 0000–01FF FFFF
Reserved	52	0200 0000–0200 0033
QDMA registers	16M–52	0200 0034–02FF FFFF
Reserved	720M	0300 0000–2FFF FFFF
Reserved	64M	3000 0000–33FF FFFF
McBSP0 data port	64M	3400 0000–37FF FFFF
McBSP1 data port	64M	3800 0000–3BFF FFFF
Reserved	1M	3C00 0000–3C0F FFFF
McASP0 data port	1M	3C10 0000–3C1F FFFF
McASP1 data port	62M	3C20 0000–7FFF FFFF
Reserved 1G +	256M	8000 0000–8FFF FFFF
EMIF CE0*	256M	9000 0000–9FFF FFFF
EMIF CE1*	256M	A000 0000–AFFF FFFF
EMIF CE2*	256M	B000 0000–BFFF FFFF
EMIF CE3*	1G	C000 0000–FFFF FFFF

Tableau 2.2 : Organisation de la mémoire

5-1-2 Les Registres de contrôle :

Le tableau 2.3 suivant représente les différents registres de contrôle de la famille TMS320c6000

Abbreviation	Register Name	Description
AMR	Addressing mode register	Specifies whether to use linear or circular addressing for each of eight registers; also contains sizes for circular addressing
CSR	Control status register	Contains the global interrupt enable bit, cache control bits, and other miscellaneous control and status bits
IFR	Interrupt flag register	Displays status of interrupts
ISR	Interrupt set register	Allows manually setting pending interrupts
ICR	Interrupt clear register	Allows manually clearing pending interrupts
IER	Interrupt enable register	Allows enabling/disabling of individual interrupts
ISTP	Interrupt service table pointer	Points to the beginning of the interrupt service table
IRP	Interrupt return pointer	Contains the address to be used to return from a maskable interrupt
NRP	Nonmaskable interrupt return pointer	Contains the address to be used to return from a nonmaskable interrupt
PCE1	Program counter, E1 phase	Contains the address of the fetch packet that is in the E1 pipeline stage

Tableau 2.3 : Registres de contrôles

Les registres additionnels de la sous familleTMS320c67xx sont représenté sur le tableau 2.4 suivant :

Register		Description
Abbreviation	Name	
FADCR	Floating-point adder configuration register	Specifies underflow mode, rounding mode, NaNs, and other exceptions for the .L unit.
FAUCR	Floating-point auxiliary configuration register	Specifies underflow mode, rounding mode, NaNs, and other exceptions for the .S unit.
FMCR	Floating-point multiplier configuration register	Specifies underflow mode, rounding mode, NaNs, and other exceptions for the .M unit.

Tableau 2.4 : Registres Additionnels de la sous famille TMS320c67xx

5-1-3 Les Périphériques :

❖ Les Timers :

Les timers permettent de mesurer la durée des événements, de compter des événements, de générer des impulsions, de générer des interruptions CPU et de synchroniser les échanges lors des accès directs à la mémoire.

Chaque timer peut être piloté par une horloge interne ou une horloge externe. Il possède chacun une TINPx et une TOUTx.

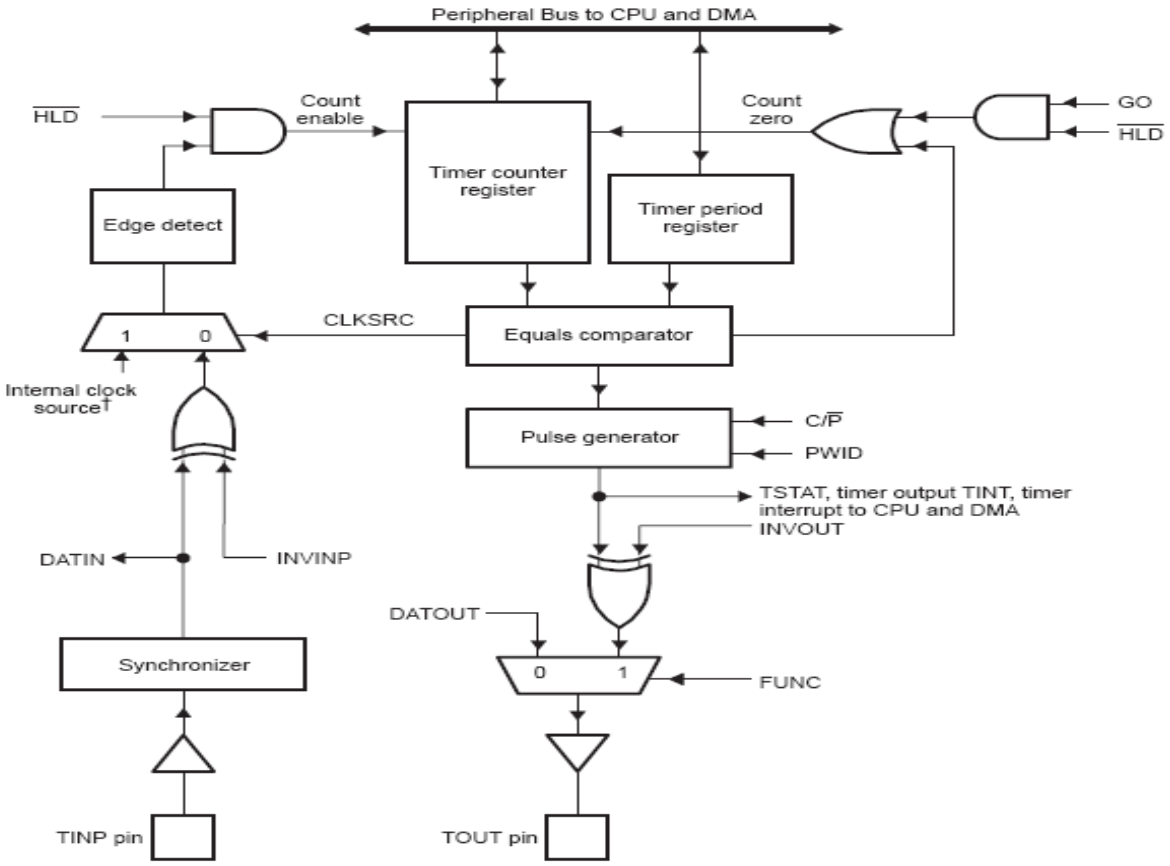


Figure 2.5 : Schéma bloc du timer

Les registres du timer :

Chaque timer possède 3 registres permettant de le configurer, de fixer la période et un registre contenant du comptage.

N.B : TIMERx_CTL, TIMERx_PRD et TIMERx_CNL sont des registres 32bits.

Adresse Timer0 (hexa)	Adresse Timer1 (hexa)	Adresse Timer2 (hexa)	Nom	Description
01940000	**01980000**	**01AC0000**	**TIMERx_CTL**	Détermine le mode opératoire du timer, et contrôle le fonctionnement de TOUT pin
01940004	**01980004**	**01AC0004**	**TIMERx_PRD**	Contient le nombre d'entrée d'horloge à compter .ce nombre control la fréquence de TSTAT
01940008	**01980008**	**01AC0008**	**TIMERx_CNT**	Valeur courante du compteur incrémentant

Tableau 2.5 : Les Registres du Timer

- **Registre de TIMERx_CTL :**

La figure 2.6 suivante représente la structure du registre TIMERX_CTL

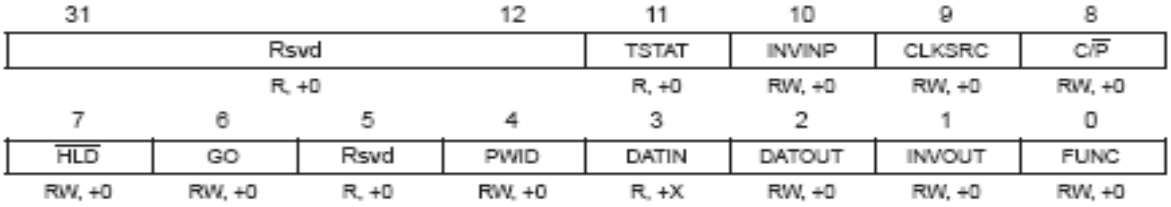


Figure 2.6 : Registre TIMERX_CTL

Description de registre TIMERX_CTL:

Champ	Description
FUNC	Fonction de la broche Tout FUNC=0 – TOUT est une sortie quelconque FUNC=1 – TOUT est la broche de sortie du timer
DATAOUT	Data output FUNC=0 – DATAOUT = TOUT FUNC=1 – TSTAT est dirigée sur TOUT après inversion par INVOUT
DATAIN	Valeur présente sur TINP
GO	Reset et démarre le compteur du timer GO=0 – pas d'effet sur le timer GO=1 – si HLD=1 le registre TIMER_CTR est mis à 0 et commence à compter au coup d'horloge suivant.
HLD	Le compteur doit être lu en regard à la valeur de HLD HLD=0 le compteur est désactivé HLD=1 – le compteur est autorisé à compter.
C/P	Clock/ pulse mode C/P=0 Pulse mode. TSTAT est activé sur le front d'horloge CPU après que le timer est atteint la période du timer. .PWID détermine quand il devient inactif C/P=1 Clock mode. TSTAT est un signal périodique de rapport cyclique 0,5 où un niveau dure un cycle complet de comptage.
PWID	Pulse width. Seulement utilisé lorsque C/P = 0 PWID = 0 - TSTAT redevient inactif après un seul front d'horloge actif supplémentaire PWID = 1 – TSTAT devient inactif après deux fronts d'horloge actif supplémentaire
CLKSRC	Timer input clock source CLKSRC = 0 – Une horloge externe pilote TINP CLKSRC = 1 - CPU clock /4
INVINP	TINP inverter control. (efficace seulement si CLKSRC =0) INVINP = 0 - TINP pilote le timer INVINP = 1 - l'inverse de TINP pilote le timer.
TSTAT	Timer Statut. Valeur de sortie du timer.
INVOUT	TOUT inverter control. Utilisé seulement si FUNC =1. INVOUT = 0 – TSTAT pilote TOUT INVOUT = 1 – l'inverse de TSTAT pilote TOUT

Tableau 2.6 : Description de registre TIMERX_CTL

- **Registre de TIMERx_PRD :**

Le registre de TIMERx PRD contient le nombre de cycle d'horloge de ''timer input '' à compter. Ce nombre contrôle la fréquence de TSTAT.

Figure 2.7 : Registre TIMERX_PRD

❖ **Registre de TIMERx_CNT :**

Le registre de TIMERx_CNT incrémente quand il lui est permis de compter. Il réinitialise à 0 sur la prochaine CPU chronométrée après que la valeur dans le registre de TIMERx est atteinte.

Figure 2.8 : Registre TIMERX_CNT

❖ **Le Port d'interface « HOTE » : (HPI)**

Le port HPI ''Host Port Interface '' est un port 16 bits pouvant être adressé directement à la mémoire. L'hôte et le CPU peuvent échanger des informations via une mémoire interne ou externe. Le HPI permet notamment à un processeur «hôte » externe d'effectuer des accès mémoire directs, que celle-ci soit interne ou dans l'espace mémoire du 6713 propriétaire.

La figure représente une interface entre un hôte et un 6713. On pourra remarquer que le format des données soit sur 32 bits, le port, lui n'en contient que 16 bits. Le transfert se fera donc en 2 étapes, l'ordre du demi-mot transféré étant repéré par le HWOB du registre HPIC.

Les broches HDS1, HDS2, HR/W et HAS permettent un interfaçage avec un nombre de composant sans ajouts de logique d'interface. **[6]**

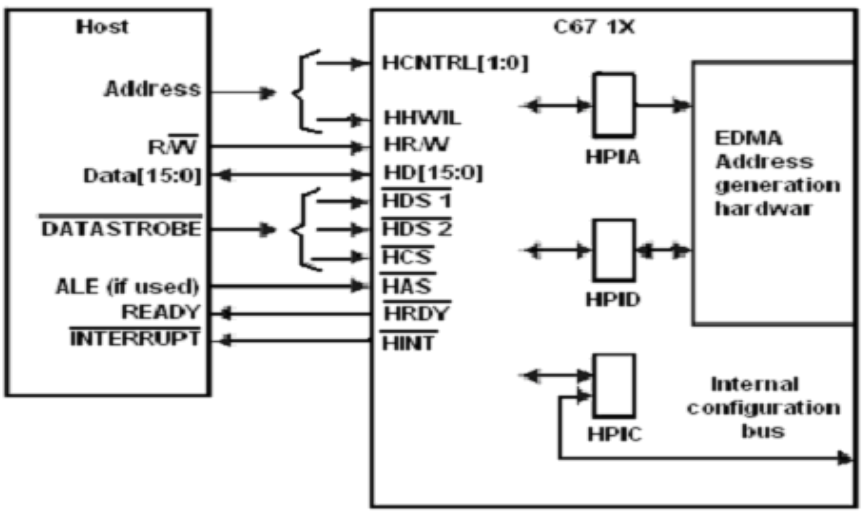


Figure 2.9 : Description du module HPI

Les registres de HPI sont utilisés pour la communication entre l'appareil hôte et le CPU. Ceci est montre le tableau 2.7 ci-dessous :

Abréviation	Nom de registre	accès autorisé à l'hôte	accès CPU	adresse (HEXA)
HPID	**HPI data**	**R/W**	-	-
HPIA	**HPI adresse**	**R/W**	-	-
HPIC	**HPI contrôle**	**R/W**	**RW**	**01880000**

Tableau 2.7 : Les registres de HPI

- HPID contient la donnée à échange sur 32 bits.
- HPIA contient l'adresse sur 30 bits donc les deux LSBs (les deux bits de poids faible) ne sont pas utilisés par les HPIA writes et sont toujours lus comme 0.
- HPIC est normalement le premier registre à accéder à la configuration des bits et à initialiser l'inter-face.

Le HPIC est organisé en deux mots de 16 bits dont l'organisation du point de vue des champs est identique. Lorsque l'hôte écrit, les deux mots doivent être identiques. La CPU seule écrite dans le mot de poids faible.

Figure 2.10 : le Registre HPIC

Description des champs :

Champ	Description
HWOB	Halfword ordering bit Si HWOB=1, le premier mot est le mot de poids faible. si HWOB=0, le premier mot est le bit de poids fort. cette propriété est affectée aux données et aux adresses. Seul l'hôte peut modifier ce bit.
DSPINT	Interruption du processeur hôte vers la CPU
HINT	interruption du DSP vers l'hôte. L'inverse de ce bit détermine la sortie CPU HINT
HRDY	Ready signal to Host Si HRDY=0, le bus interne attend une requête d'accès d'une donnée pour finir.
FETCH	Host fetch request la valeur lue par l'hôte ou le CPU est toujours 0. l'hôte écrite un 1 pour effectuer une requête un accès dans HPID à l'adresse pointée par HPIA

Tableau 2.8 : Description des champs de HPI

5-1-4 La liaison série « MCBSP »: (Multichannel buffered serial port)

Le 6713 contient deux modules de liaison série (McBSP0 et McBSP1). Ces liaisons séries permettent des communications en full-duplex, un flot continu des données, une indépendance dans le timing, le format des données à l'émission et à la réception, une interface directe avec les Codec, CNA/CAN, et une synchronisation par horloge interne ou externe.

Ces liaisons transmettent des données et des informations de contrôle. La communication avec des composants externes se fait en émission et en réception sur des broches différentes.

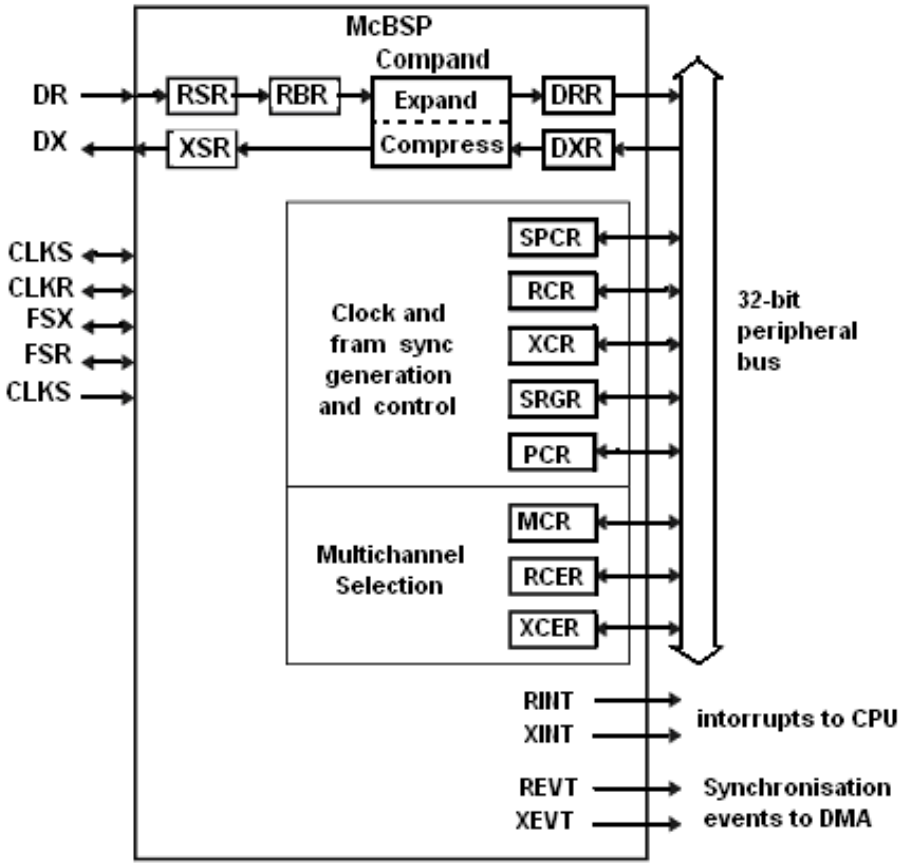


Figure 2.11 : Schéma bloc du module « liaison série » du TMS320C6713

Registre de configuration :

McBSP0	McBSP1	Abréviation	Rôle
-	-	**RBR**	Buffer de réception
-	-	**RSR**	Buffer de l'horloge de réception
-	-	**XSR**	Buffer de l'horloge d'émission
018C0000	01900000	**DRR**	Registre de réception
018C0004	01900004	**DXR**	Registre d'émission
018C0008	01900008	**SPCR**	Registre de contrôle du port série
018C000C	0190000C	**RCR**	Registre de contrôle en réception
018C0010	01900010	**XCR**	Registre de contrôle en émission
018C0014	01900014	**SRGR**	Registre de génération des échantillons
018C0018	01900018	**MCR**	Registre de contrôle des canaux
018C001C	0190001C	**RCER**	Registre d'autorisation de réception des canaux
018C0020	01900020	**XCER**	Registre d'autorisation des canaux à l'émission
018C0024	01900024	**PCR**	Registre de contrôle des broches

Tableau 2.9 : Registre de configuration de McBSP

Interruption :

- RINT est l'interruption générée à la réception.
- XINT est l'interruption générée à l'émission.

Signaux d'interfaces :

Broche	I/O/Z	Description
CLKR	**I/O/Z**	Horloge de réception
CLKX	**I/O/Z**	Horloge de transmission
CLKS	**I**	Horloge externe
DR	**I**	Donnée série en réception
DX	**O/Z**	Donnée série en émission
FSR	**I/O/Z**	synchronisation de trame en réception
FSX	**I/O/Z**	synchronisation de trame en transmission

Tableau 2.10 : Signaux d'interface de MCBSP

5-1-5 L'accès Mémoire en mode direct « EDMA » :

Le TMS320c6713 élabore des transferts de donnée depuis un élément interne ou externe en utilisant soit la CPU soit l'EDMA. Typiquement, les transferts depuis les périphériques internes au TMS320C6713 sont faits par l'EDMA libérant ainsi la CPU pour des tâches de calcul.

L'EDMA contient 16 canaux de transmission dont les priorités sont programmables et permet des transferts d'informations depuis la mémoire cache L2, vers les périphériques du TMS320c6713 et vers la mémoire externe.
L'EDMA contient des registres d'évènements et d'interruptions, des encodeurs d'évènement, de la RAM ainsi qu'un module de génération d'adresses.

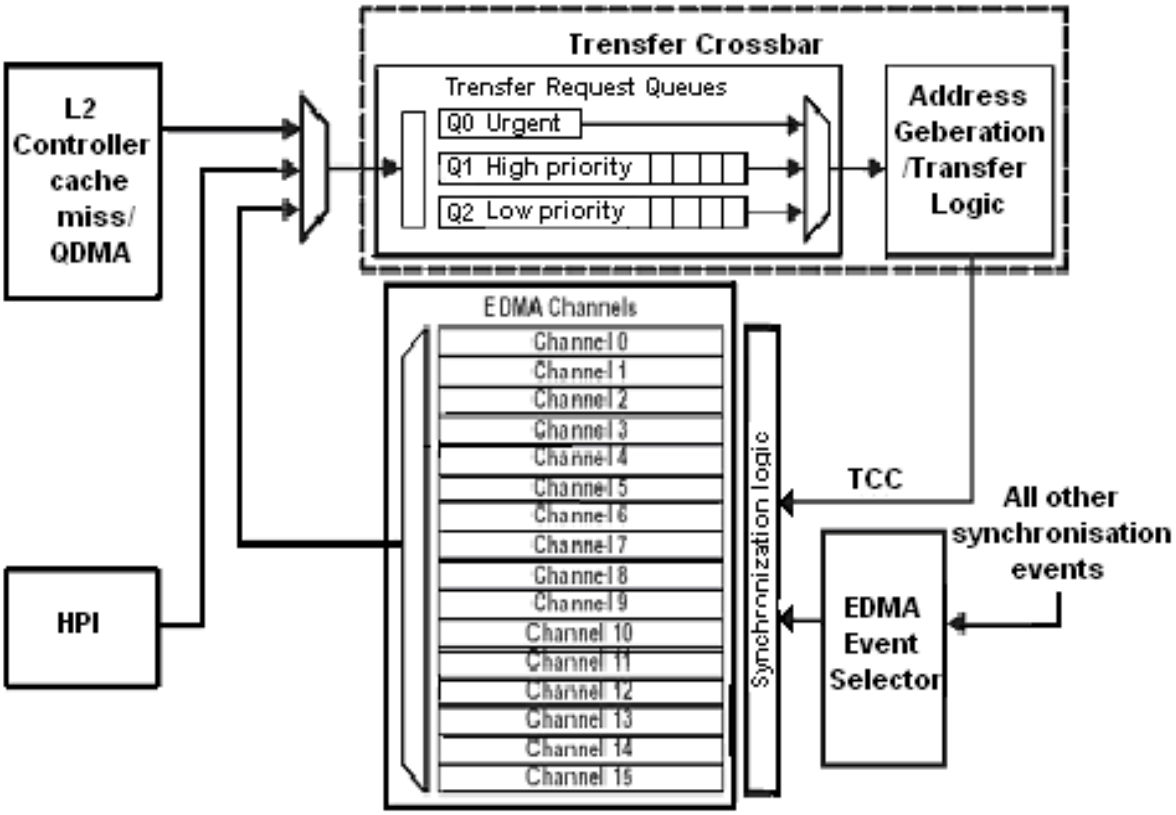


Figure 2.12 : Architecture d'EDMA

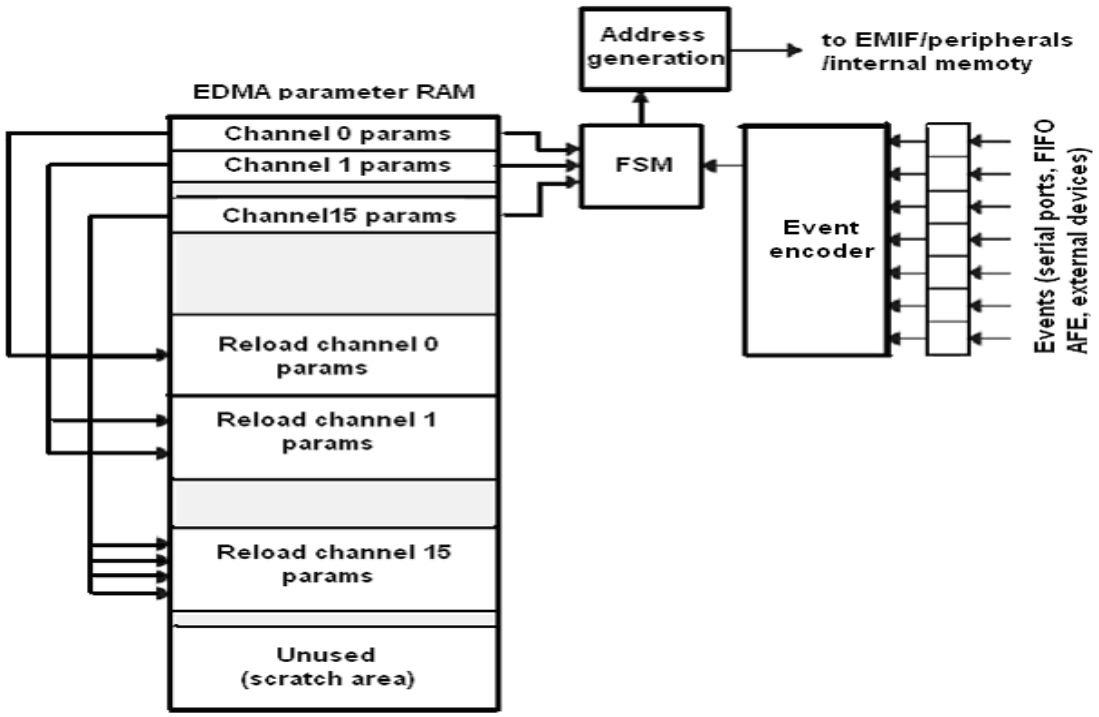


Figure 2.13 : Schéma bloc du module EDM

Les évènements EDMA sont capturés dans le registre d'évènements. Un évènement est un signal de synchronisation qui déclenche un canal EDMA pour démarrer le transfert.

Si plusieurs évènements arrivent en même temps, ils sont traités par priorités par l'encodeur d'évènements.

Les paramètres correspondants à un évènement sont enregistrés dans la « RAM paramètre » et sont transmis à la génération d'adresse qui transmet à l'EMIF et/ou aux périphériques, les instructions nécessaires à la lecture et à l'écriture.

Les registres d'EDMA :

Chacun des 16 canaux a des évènements qui lui sont associés. Ces évènements déclenchent le transfert de données en association avec le canal. La liste des registres de contrôle est variable suivant l'évènement.

Tous les évènements sont signalés par un front montant sur l'une des 16 entrées. Un évènement qui apparaît, engendre le positionnement du bit, lui correspondant dans le registre d'évènement (ER). Les registres utilisés sont donnés dans le tableau suivant :

adresse (hexa)	Abréviation	Nom du registre
01A0FFE0	**PQSR**	Priority Queue Status Registre
01A0FFE4	**CIPR**	Channel Interrupt Pending
01A0FFE8	**CIER**	Channel Interrupt Enable Register
01A0FFEC	**CCER**	Channel Chain Enable Register
01A0FFE0	**ER**	Event Register
01A0FFE4	**EER**	Event Enable Register
01A0FFE8	**ECR**	Event clear Register
01A0FFEC	**ESR**	Event Set Register

Tableau 2.11 : Les registres EDMA

5-1-6 L'interface de mémoire externe « EMIF » :

L'interface de mémoire externe du TMS320C6713 permet l'interfaçage avec différents types de composants : SBSRAM, SRAM, RAM, FIFO, EDMA …. L'EMIF du TMS320C6713 nécessite une horloge externe (ECLKIN) fournie par le système.

L'horloge ECLKOUT est élaborée par l'EMIF et toutes les mémoires externes doivent travailler avec cette horloge.

Les signaux de l'EMIF :

Le schéma suivant représente les signaux nécessaires au fonctionnement de l'EMIF.

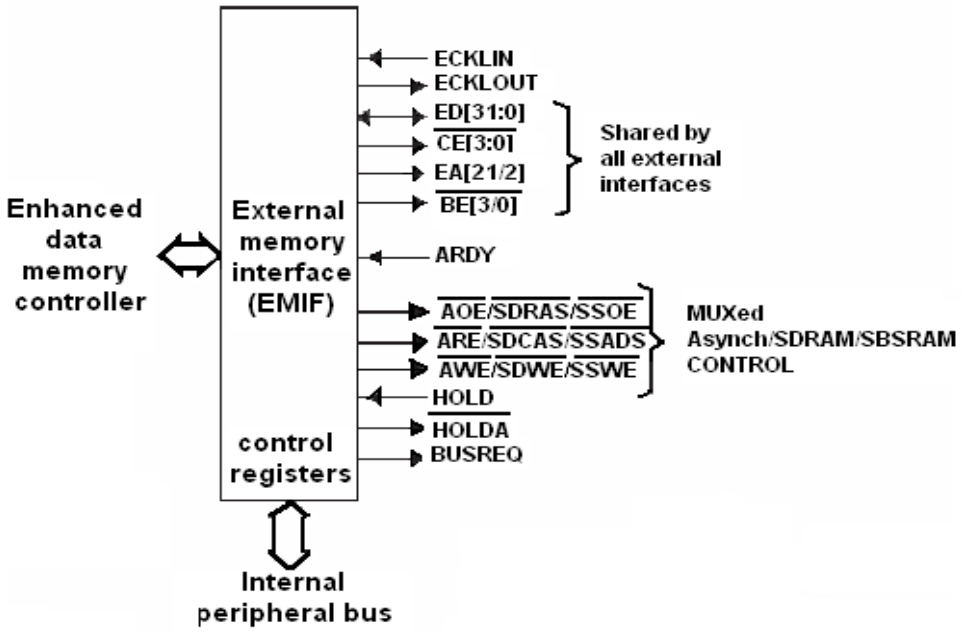


Figure 2.14 : Les signaux de l'EMIF

Description des signaux :

ECLKIN	I	EMIF clock input. Must be provided by the system on C621x/C671x. Optionally provided by the system on C64x.
ECLKOUT	O	EMIF clock output. All EMIF I/O are clocked relative to ECLKOUT.
ED[31:0]§	I/O/Z	EMIF 32-bit data bus I/O§
CLKOUT1	O	Clock output. Runs at the CPU clock rate.
CLKOUT2	O	Clock output. Runs at 1/2 the CPU clock rate. Used for synchronous memory interface on Group 2 devices.
EA[21:2]	O/Z	External address output. Drives bits 21–2 of the byte address. (Effectively a word address.)
$\overline{CE0}$	O/Z	Active-low chip select for memory space CE0
$\overline{CE1}$	O/Z	Active-low chip select for memory space CE1
$\overline{CE2}$	O/Z	Active-low chip select for memory space CE2
$\overline{CE3}$	O/Z	Active-low chip select for memory space CE3
$\overline{BE[3:0]}$	O/Z	Active-low byte enables. Individual bytes and halfwords can be selected for write cycles. For read cycles, all four byte-enables are active.
ARDY	I	Ready. Active-high asynchronous ready input used to insert wait states for slow memories and peripherals.
$\overline{AOE}$	O/Z	Active-low output enable for asynchronous memory interface
$\overline{AWE}$	O/Z	Active-low write strobe for asynchronous memory interface
$\overline{ARE}$	O/Z	Active-low read strobe for asynchronous memory interface
$\overline{SSADS}$	O/Z	Active-low address strobe/enable for SBSRAM interface
$\overline{SSOE}$	O/Z	Active low output buffer enable for SBSRAM interface
$\overline{SSWE}$	O/Z	Active-low write enable for SBSRAM interface
$\overline{SDRAS}$	O/Z	Active-low row address strobe for SDRAM memory interface
$\overline{SDCAS}$	O/Z	Active-low column address strobe for SDRAM memory interface
$\overline{SDWE}$	O/Z	Active-low write enable for SDRAM memory interface
$\overline{HOLD}$	I	Active-low external bus hold (3-state) request
$\overline{HOLDA}$	O	Active-low external bus hold acknowledge

Tableau 2.12 : Description des signaux de l'EMIF

Les registres de l'EMIF :

adresse (hexa)	Abréviation	Nom de registre
01800000	**GBLCTL**	**EMIF global control**
01800004	**CE1CTL**	**EMIF CE1 space control**
01800008	**CE0CTL**	**EMIF CE0 space control**
0180000C	**Reserved**	
01800010	**CE2CT**	**EMIF CE2 space control**
01800014	**CE3CTL**	**EMIF CE3 space control**
01800018	**SDCTL**	**EMIF SDRAM control**
0180001C	**SDTIM**	**EMIF SDRAM refresh control**

Tableau 2.13 : Les registres de l'EMIF

5-2 Starter KIT TMS320C6713 :

Le C6713 DSK est une plateforme du développement autonome bas-prix qui permet aux utilisateurs, d'évaluer et développer des applications pour le TI C67xx DSP famille. Le DSK sert aussi comme hardware référence du matériel pour le TMS320C6713 DSP.

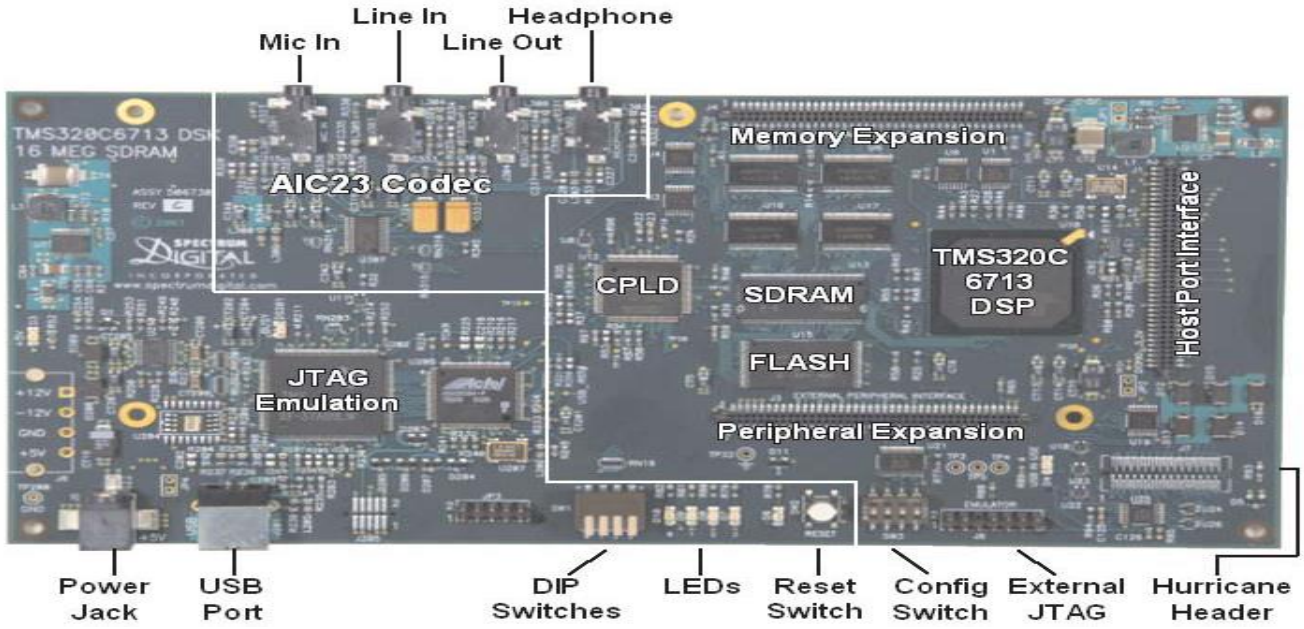


Figure 2.15 : La Carte Starter Kit (C6713 DSK)

La figure 2.16 représente le corps du microcontrôleur intégrant la CPU, les périphériques et les interfaces externes:

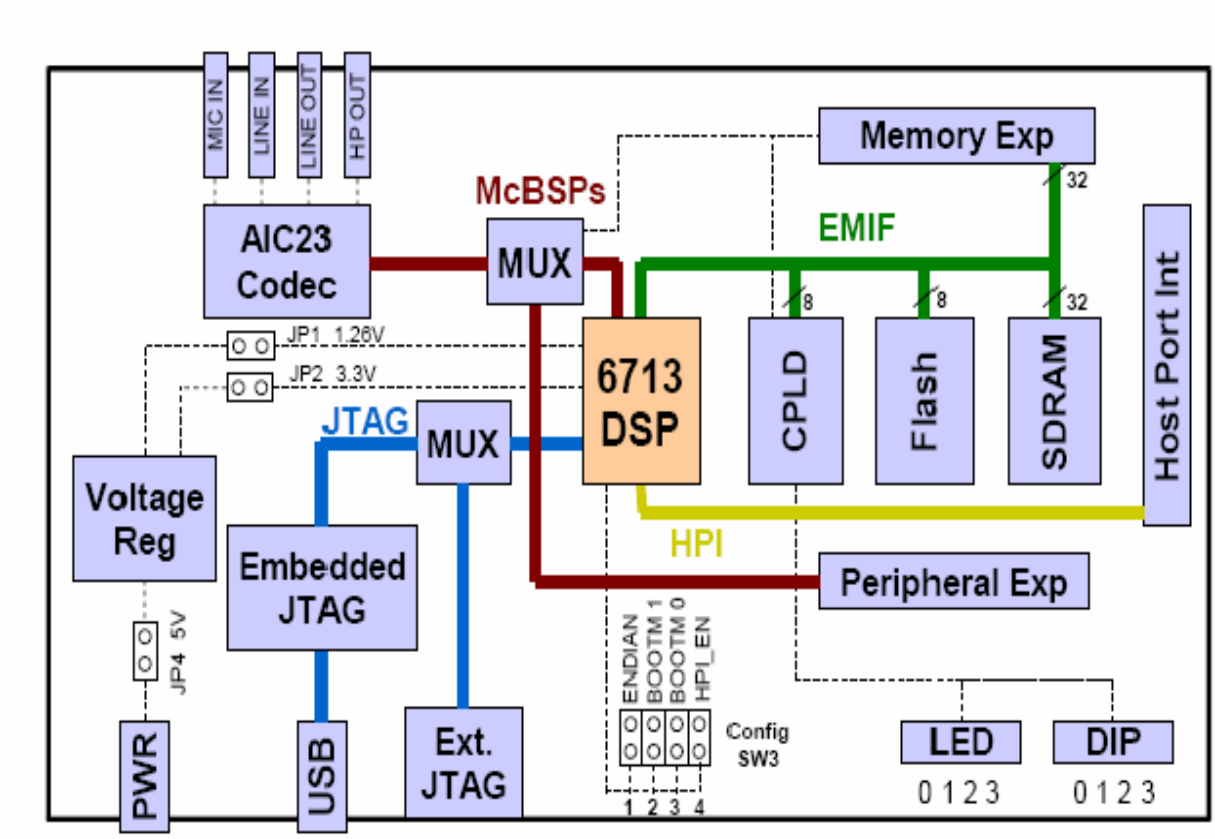


Figure 2.16 : Schéma globale de la carte C6713 DSK

5-2-1 Alimentation d'énergie :

Le DSK fonctionne à partir d'une alimentation d'énergie externe simple de +5V reliée à la puissance principale fournie (J5). Intérieurement, l'entrée de +5V est convertie en +1.26V et +3.3V en utilisant les régulateurs de tension séparés.

L'approvisionnement de +1.26V est employé pour le noyau de DSP tandis que l'approvisionnement de +3.3V est employé pour les amortisseurs de l'I/O du DSP et tous autres morceaux sur le panneau. Le connecteur de puissance est un baril-type-prise de 2.5mm. Il y a trois points test de mesure de puissance sur le DSK (JP1, JP2 et JP4).

Tout le courant d'I/O traverse JP2 tandis que courant principal passe par JP1. Tout le courant du système passe par JP4. Normalement, ces jumpers sont fermés. Pour mesurer le courant de passage il faut enlever les jumpers et relier les goupilles à un appareil de mesure du courant tel qu'un multimètre ou une sonde de courant. Il est possible d'approvisionner la carte fille avec +12V et -12V quand le connecteur de puissance externe (J6) est utilisé. **[5]**

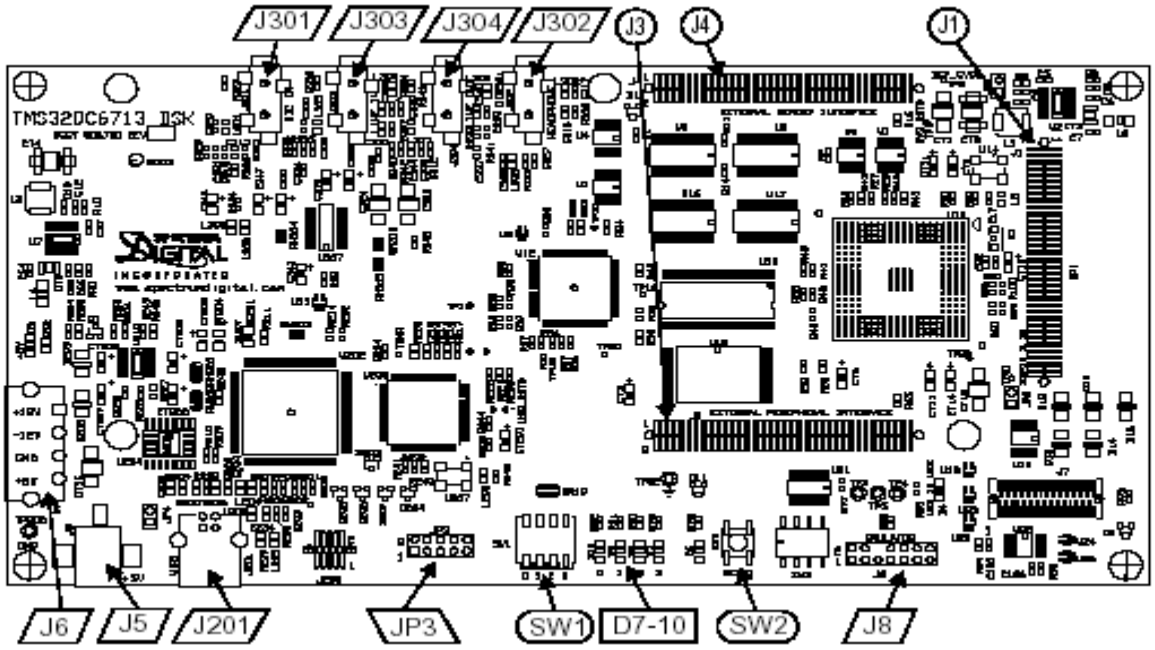


Figure 2.17 : Schéma d'Alimentation d'énergie

5-2-2 Fonctionnement :

Le DSP sur les 6713 interfaces de DSK aux périphériques à bord par un EMIF de 32 bits. Le SDRAM, le flash et le CPLD sont reliés au bus.

Les signaux d'EMIF sont des connecteurs également branchés d'expansion de daughter card ''carte fille '' qui sont utilisés pour les panneaux addition de tiers. Le DSP connecte aux signaux audio analogues par un codec AIC23 à bord et quatre crics audio de 3.5 millimètres (entrée de microphone, ligne entrée, ligne de sortie, et la sortie d'écouteur). Le codec peut choisir le microphone ou la ligne entrée comme entrée active. La sortie analogique est conduite à la ligne sortie (gain fixe) et aux connecteurs d'écouteur (gain réglable).

Un programmable logique dispositif appelé un CPLD est utilisée pour mettre en application la ''glue logic'' qui attache les composants entre eux. Le DSK inclut 4 LED et 4 positions DIP interrupteur comme manière simple de fournir à l'utilisateur la rétroaction interactive. Tous les deux sont consultés par la lecture et l'inscription aux registres de CPLD.

Une alimentation de l'énergie 5V externe incluse est employée pour actionner le panneau. Les régulateurs de tension à bord de commutation fournissent la tension de noyau de +1.26V DSP et des approvisionnements de +3.3V I/O. Le panneau est tenu dans la remise jusqu'à ce que ces approvisionnements soient selon des caractéristiques de fonctionnement.

Le code composer communique avec le DSK par un émulateur JTAG incorporé avec une hôte interface d'USB. Le DSK peut également être utilisé avec un émulateur externe par le connecteur externe de JTAG. **[6]**

5-2-3 Configuration des Paramètres des Switchs :

Le DSK a 4 switchs qui permettent aux utilisateurs de contrôler l'état opérationnel du DSP lorsqu'il y a une réinitialisation.

Le switch 1 Contrôle l'endianness du DSP et les switch 2 et 3 configure le mode de la mise en route qui sera utilisé lorsque le DSP commence à exécuter. Le switch 4 contrôle le multiplexage des signaux HPI et McASPs fait sortir au HPI expansion connecteur.

Switch 1	Switch 2	Switch 3	Switch 4	Configuration Description
Off				Little endian (default)
On				Big endian
	Off	Off		EMIF boot from 8-bit Flash (default)
	Off	On		HPI/Emulation boot
	On	Off		32-bit EMIF boot
	On	On		16-bit EMIF boot
			Off	HPI enabled on HPI pins (default)
			On	McASP1 enabled on HPI pins

Tableau 2.14 : Configuration de Paramètres des switchs

5-2-4 Les Composants de la carte :

5-2-4-1 Le Codec AIC23 :

Les entrées et sorties audio sont traitées par un circuit CODEC (AIC23). Les entrées audio sont découplées par une impédance capacitive (couplage AC). Le CODEC échantillonne les signaux analogiques de l'entrée ligne ou de l'entrée microphone et les convertit en données numériques. Il est également chargé de la reconstitution des signaux de sortie (Sortie ligne ou haut-parleur). Il s'agit donc d'un double convertisseur ADC et DAC.

Le CODEC communique avec via deux canaux de communication séries, l'un pour le control de ces registres interne, l'autre pour envoyer et recevoir les échantillons.

Le CODEC est cadencé par une horloge à 12MHz mais on peut diviser ou multiplier cette fréquence en interne et il est possible donc fonctionner à des fréquences d'échantillonnage telles que 48KHz, 44.1KHz ou 8KHz. **[5]**

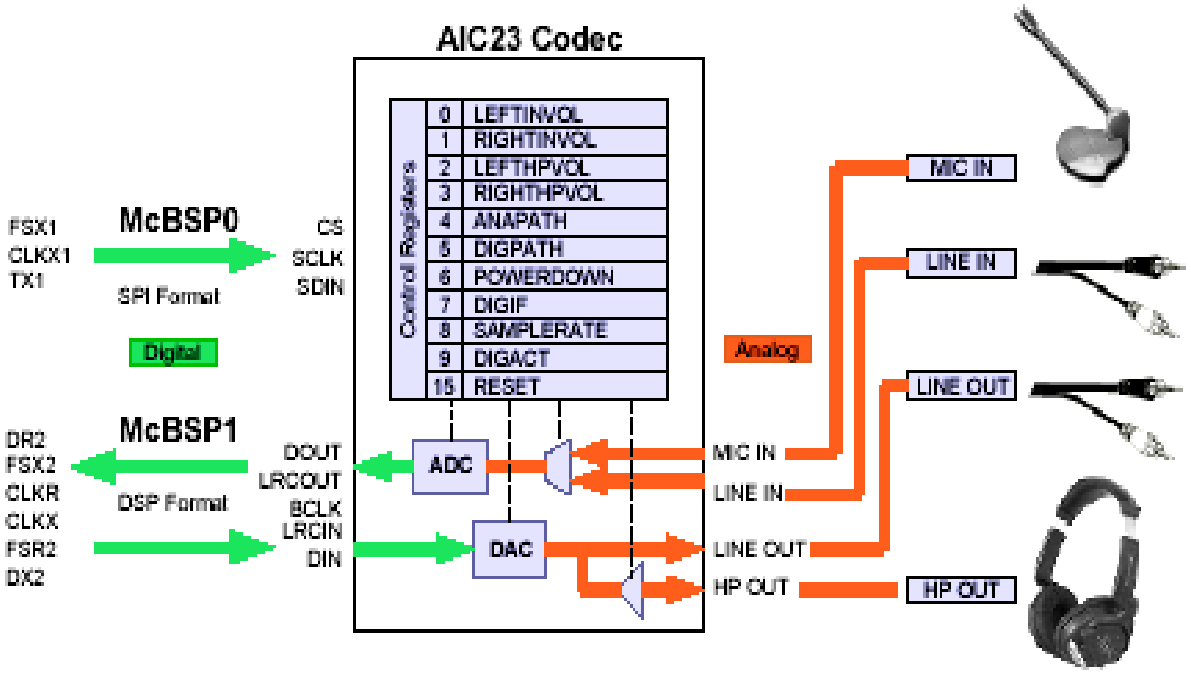


Figure 2.18 : Schéma du Codec AIC23

5-2-4-2 JTAG :

Comme tous les processeurs récents, le DSP est équipé d'une **interface JTAG** qui permet, depuis Code Composer Studio exécuté sur un PC, de contrôler très précisément le DSP : mis en place de points d'arrêts, lecture-écriture des registres internes, accès en lecture-écriture à tous l'espace mémoire. Cette dernière fonctionnalité sera d'ailleurs utilisée pour charger la mémoire du DSP avec un programme et des données.

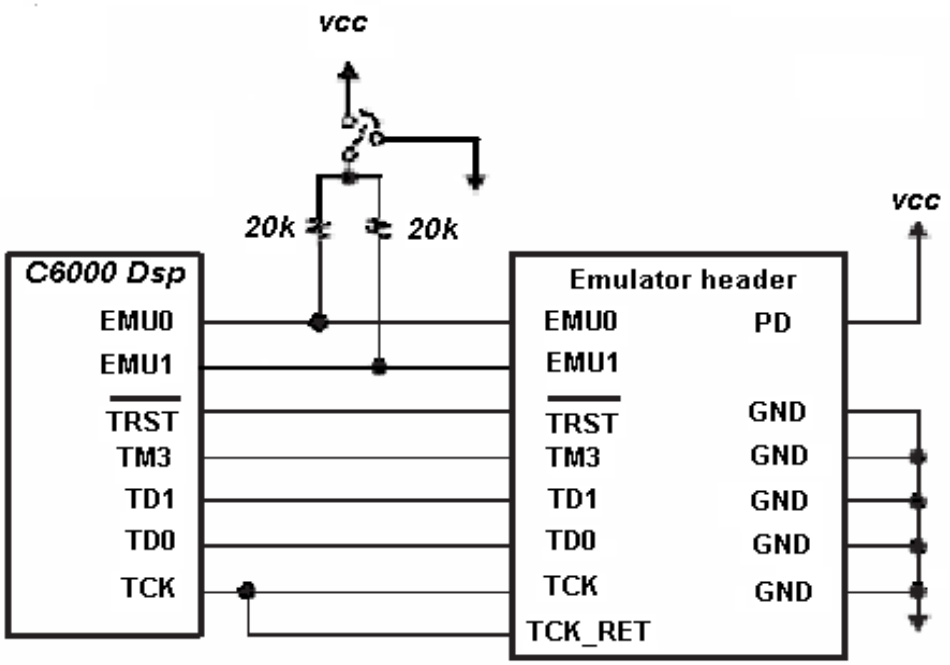


Figure 2.19 : Schéma de JTAG

5-2-4-3 Le CPLD :

Le DSK C6713 utilise un Altera EPM3128TC100-10 (CPLD) dispositif pour mettre en application :

- 4 Memory-mapped de control/status registres qui permettent la commande de logiciel de divers dispositifs de la carte.
- Commande de l'interface et des signaux de daughter card ‘’carte fille ‘’.
- Attache les composants de la carte ensemble.

❖ **Synchronous DRAM :**

Le DSK utilise 128 Mbits de synchrone DRAM (SDRAM) sur 32 bits de l'EMIF. La SDRAM est placé au début de CE0 (adressé à 0x80000000). La mémoire totale disponible est de 8 méga-octets. Le contrôleur intégré du SDRAM fait partie de l'EMIF et peut être configuré d’une manière soft dans des opérations appropriées. L'horloge de l'EMIF provient des arrangements de PLL et peut être configurée d’une manière soft à 90MHz.

Ce nombre est basé sur une horloge interne du PLL de 450MHz exigé pour réaliser des opérations de 225 mégahertz avec un diviseur de 2 et une horloge de l’EMIF de 90MHz avec un diviseur de 5.

En utilisant SDRAM, le contrôleur peut être initialiser pour régénérer une rangée de la mémoire toute les15.6 microsecondes pour maintenir l'intégrité des données. Avec une horloge de 90MHz de l'EMIF, cette période est l'équivalent de 1400 cycles.

- **Flash Memory :**

La mémoire non volatile est un type de mémoire qui ne perd pas son contenu quand le courant est coupé. Une fois lue, elle ressemble à une mémoire morte asynchrone simple (ROM).

Le flash peut être effacé dans de grands blocs généralement désignés sous le nom des secteurs ou des pages. Une fois un bloc a été effacé chaque mot peut être programmé une fois par un ordre spécial de commande. Après cela le bloc entier peut être effacé entièrement pour changer le contenu. Le DSK emploie un flash 512Kbyte externe comme option d'initialisation. Ceci est visible au début de CE1 (adresse 0x90000000). Le flash est câblé comme un 256K par 16 bits pour supporter les options d'initialisation de 16 bits du DSK.

Cependant, le logiciel qui se vend avec le DSK traite le flash comme un dispositif de 8 bits (ignorant les 8 bits du début) pour se conformer avec le mode de 8 bits par défaut nécessaire pour le 6713. Dans cette configuration, seulement 256Kbytes sont utilisés sans changements de logiciel. [6]

- **LEDS et DIP Switchs :**

Le DSK inclut 4 LED (D7 - D10) accessibles et DIP switch (SW1) qui fournissent à l'utilisateur une simple forme d'entrée-sortie. Tous les deux sont consultés par le registre de CPLD USER_REG.

Conclusion

Dans ce chapitre, on a décrit le DSP TMS320C6713, et on a présenté son architecture interne, ces caractéristiques pour mieux comprendre son fonctionnement global puis être prêt pour l'outil de développement CCS qu'il sera étudié dans le chapitre suivant.

3

CHAPITRE :

Code Composer Studio

Introduction

La carte DSK est supportée par le logiciel Code Composer Studio qui contient des outils de développement, y compris un compilateur de haute optimisation de C/C++, émulation basée sur le JTAG, débogueur à temps réel (real-time debugging).

III-1 Présentation du Code Composer Studio CCS

Le Studio Code composé est un compilateur conçu exclusivement pour les C6x et C5x, développé par Texas Instrument.

Il s'agit d'un environnement complet avec un éditeur, une aide à la création d'un projet, un compilateur croisé (''cross-compiler'', un compilateur produisant un exécutable pour une autre architecture que celle sur laquelle la compilation se fait), un émulateur (qui exécute le programme cible en simulant le fonctionnement du DSP), des outils de debugging et de traçage, et un loader (qui charge l'application en mémoire RAM ou FLASH du DSP) etc.…

L'application **CCS** fournit tous les outils Software nécessaires pour le développement du DSP.

Au cœur de **CCS**, il existe l'original compositeur **IDE**. L'IDE fournit une seule fenêtre d'application dans laquelle est exécuté le code de développement ; d'entrée et d'édition du code de programme, la compilation et construction d'un fichier exécutable ''building '', et le déboguage du code de programme**. [9]**

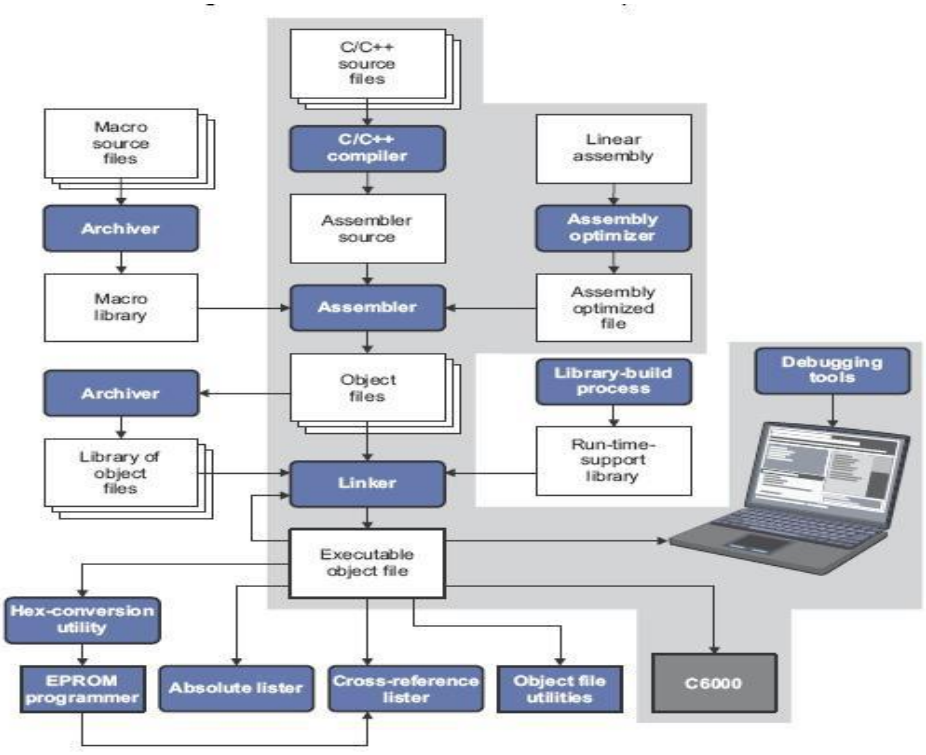


Figure 3.1 : Organigramme d'un système de développement de logiciel pour DSP

III-2 Configuration du développement et environnement

Les exécutables sur DSP sont élaborés en utilisant des environnements de développement intégrés (IDE) fournies par les fabricants des DSP, ils intègrent de nombreuses fonctions, telles que l'éditeur, le débogage, la gestion des fichiers de projet, et le profilage. L'environnement de développement pour Texas Instrument (TI) est appelé « Code Composer Studio ». Le constructeur TI met à disposition gratuitement le compilateur, l'assembleur et l'éditeur de liens optimiseur pour une utilisation non commerciale.

La figure 3.2 montre un exemple d'un écran typique du Code Composer. Sur le côté gauche il y a la liste de tous les fichiers inclus dans le projet. Au centre de l'écran deux fenêtres affichent le code, comme un fichier C (process.c) et le code assembleur (fenêtre Disassemby). L'exécution s'est arrêtée sur un point d'arrêt (breakpoint) activé au préalable. A dessous des fenêtres de codes, deux fenêtres de mémoire sont également visibles, détaillant les données présentes à partir des adresses 0x80000000, et à partir des adresses 0x40000030. La donnée à l'adresse 0x80000002 est d'une couleur différente parce que sa valeur a changé récemment. Au bas de l'écran de l'IDE, on trouve :

a) la fenêtre de compilation/Link, qui donne les résultats de la dernière compilation du code ;
b) la fenêtre Watch, qui affiche les valeurs prise par deux variables
c) et la fenêtre Register, qui affiche le contenu de tous les registres du DSP.

Sur le côté droit il y a trois graphiques: les jaunes montrent les régions de mémoire, tandis que le vert représente la transformée de Fourier rapide de données stockées en mémoire, tel que calculé par l'IDE. **[11]**

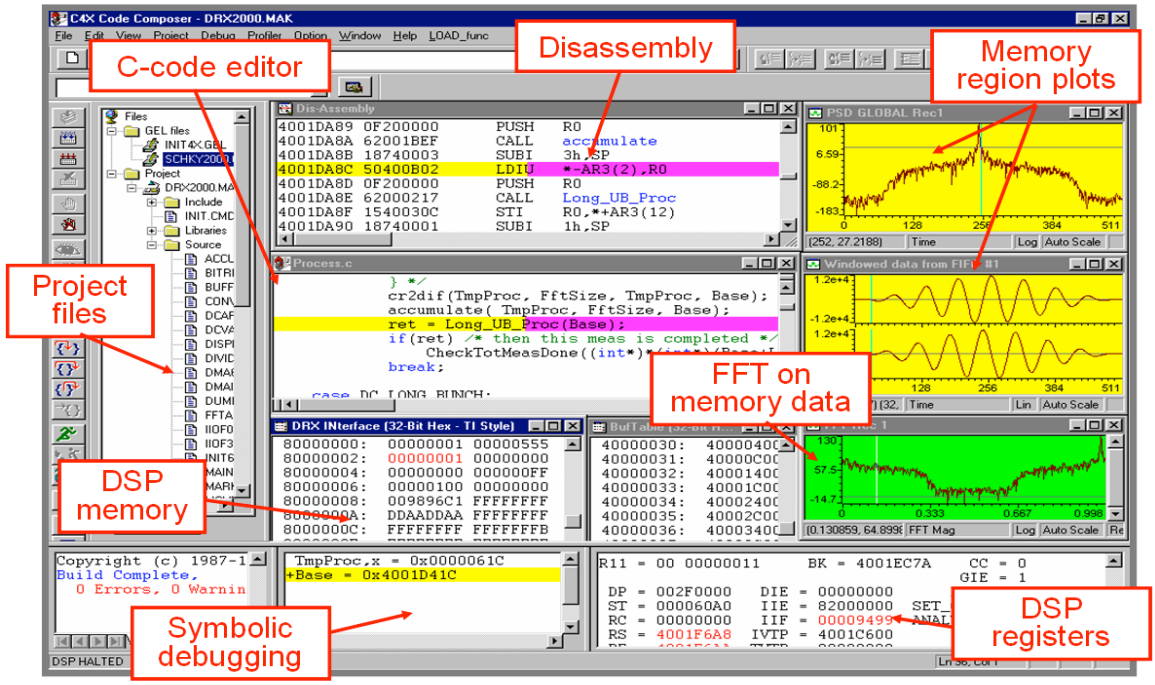


Figure 3.2 : Capture d'écran du Code Composer Studio

III-3 Installation du Code Composer Studio CCS

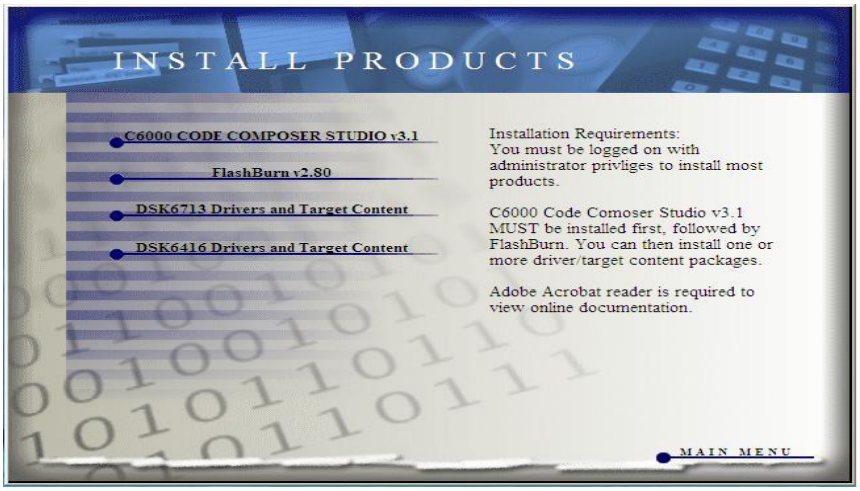
La TI a fourni le logiciel de développement CCS IDE, pour tous les DSP de TI, le compilateur C pour TMS320C6713 est construit dans le logiciel installé sur vos machines.

- Le CD d'installation de Code Composer Studio est inclus dans le kit.
- Les CD auto-Runs pour ouvrir une boîte de dialogue du menu principal.

1. MENU PRINCIPAL BOÎTE DE DIALOGUE

- Cliquez sur ' Install Products'. Il va ouvrir une autre fenêtre avec 4 options. Cliquez sur C600 CODE COMPOSER STUDIO v3.1

2. ECRAN D'INSTALLATION

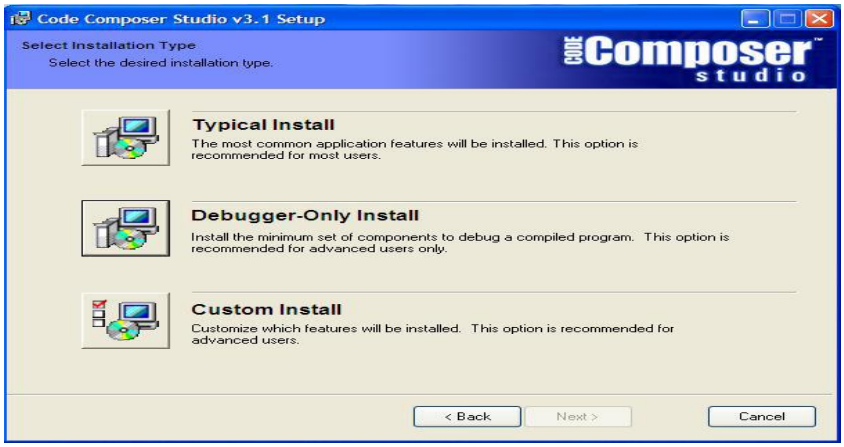


- Cliquez sur C600 CODE Composer Studio v3.1
- Cliquez sur Next.

3. Écran d'accueil

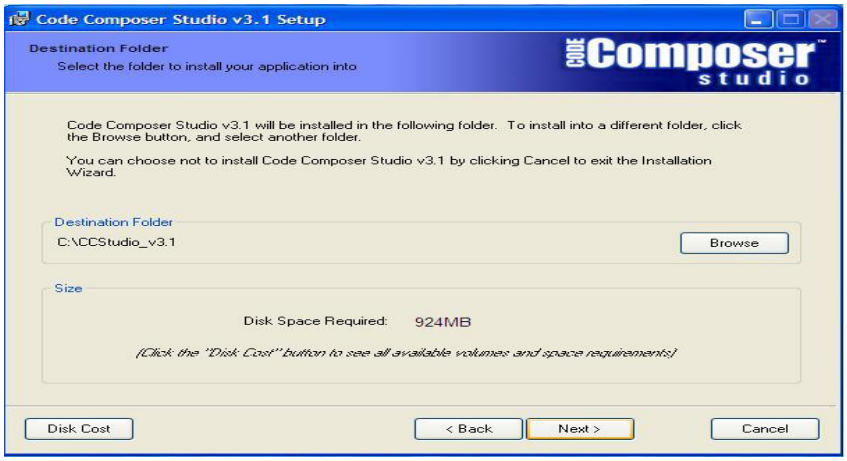
- L'installation vous demandera le type d'installation. Un 'typical' installé est recommandé.

4. Personnaliser l'installation



- Le reste des étapes sont interactives et dépendent de choix de l'utilisateur.

5. INSTALLATION LOCATION

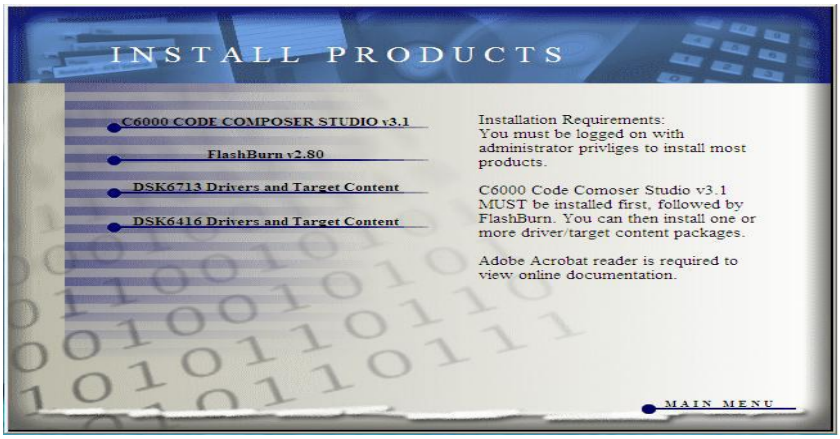


6. INSTALLATION EN COURS

- Après l'installation de CCS est terminée, vous êtes redirigé vers le menu principal.

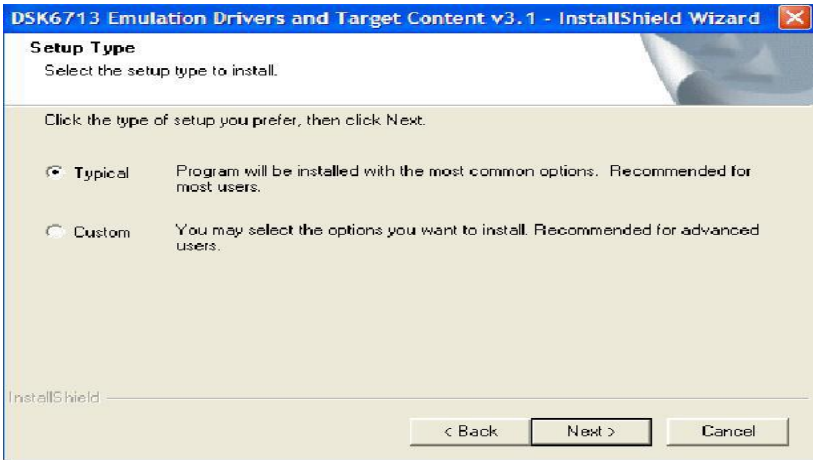
- Une fois que vous y êtes, cliquez sur le ‘DSK 6713 Drivers and Target Co

7. DSK 6713 PILOTES ET CONTENU DE CIBLE



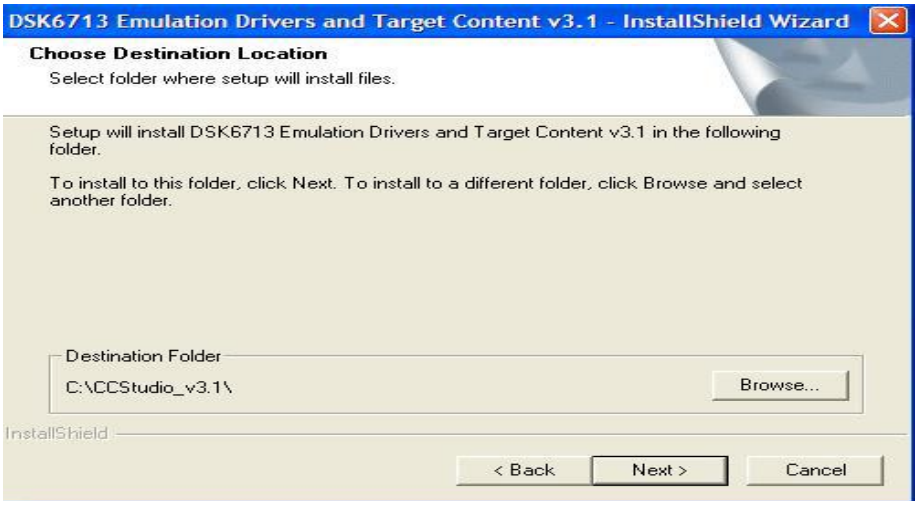
- Encore une fois, vous êtes invité à entrer le type d'installation. Un «typical» est installé recommandé.

8. Assistant d'installation



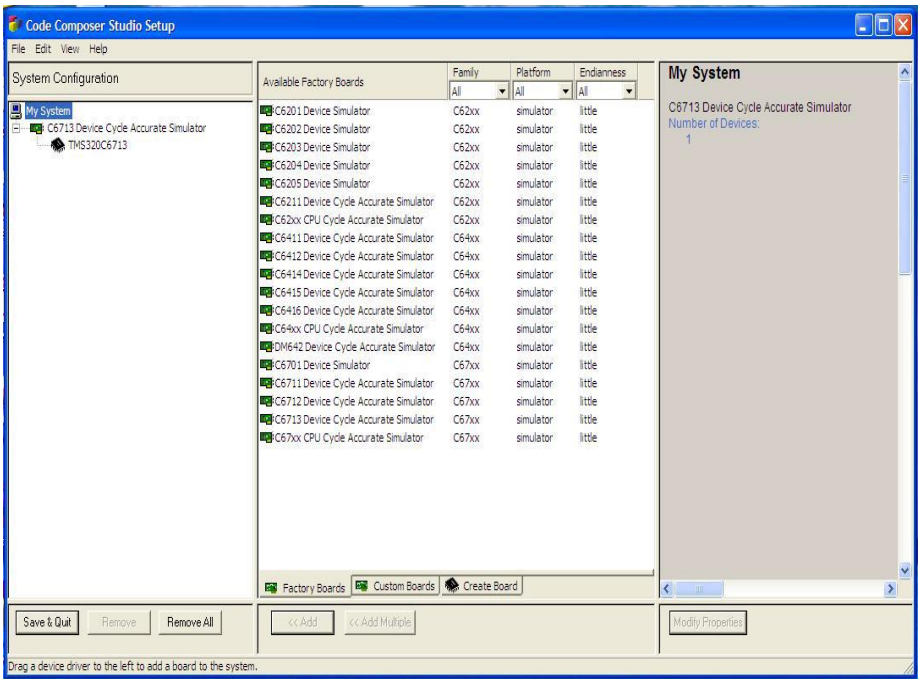
- Sélectionner le répertoire et terminer l'installation.

9. Dossier de destination



- Afin d'interfacer le dispositif cible (DSK) avec l'ordinateur en utilisant le CCS, nous avoir à ouvrir la configuration Code Composer Studio et sélectionnez DSK 6713. Ensuite, nous pouvons aller plus loin.

10. DISPOSITIF CIBLE CONNEXION

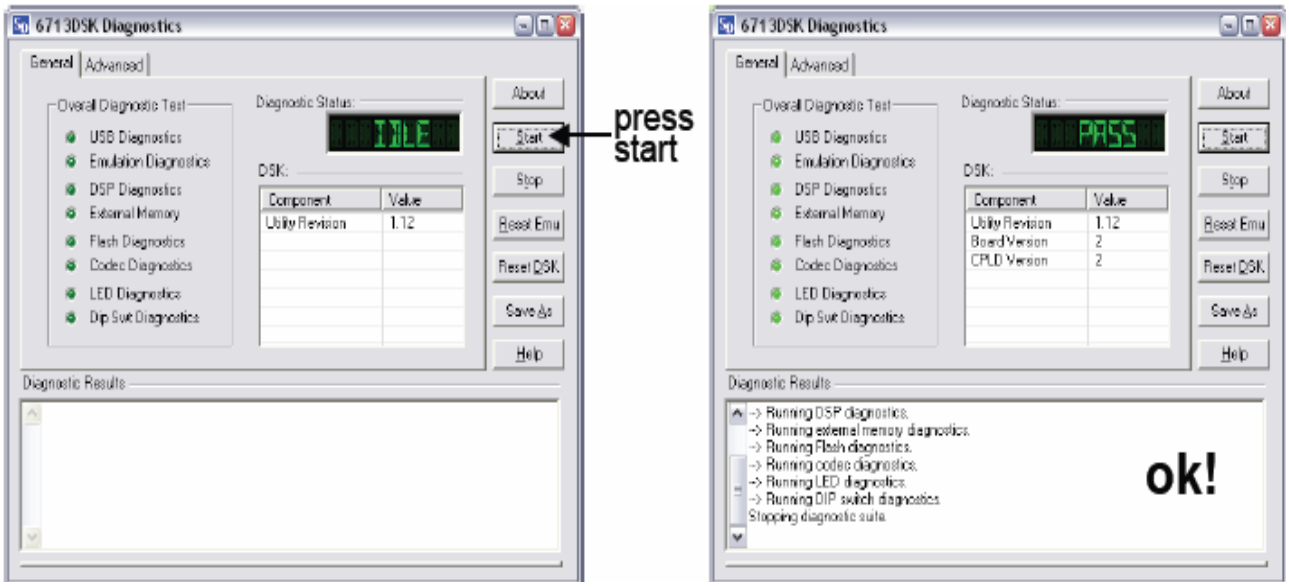


Notez que avant d'installer le logiciel DSK, assurez-vous que le PC dispose d'un port USB et un système d'exploitation (Windows 98SE/2000/XP) compatible USB. pour Windows 2000 et XP, vous devez installer Code Composer Studio à l'aide des privilèges d'administrateur. à exécuter CCS sur ces systèmes nécessite l'autorisation d'écriture sur le registre. **[11]**

11. Test de votre connexion

- Si vous voulez tester votre connexion DSK et USB, vous pouvez lancer le C6713 DSK Diagnostic Utility de l'icône sur votre bureau.
- De diagnostic utility, appuyez sur le bouton de démarrage pour exécuter les diagnostics. dans environ 20 secondes, tous les indicateurs de test sur l'écran devient verte.

12. TEST CONNEXION DU DSK



III 4 Exécution du Code Composer Studio CCS

Pour ouvrir le code composer studio dans un système d'exploitation Windows XP, double clique sur le CCS icône C6713 DSK sur votre bureau.

Ou bien cliquer sur le menu ''démarrer'' puis choisir tous les programmes, puis « Texas instrument » Code Composer Studio [3.1] et cliquer sur l'icône : ……

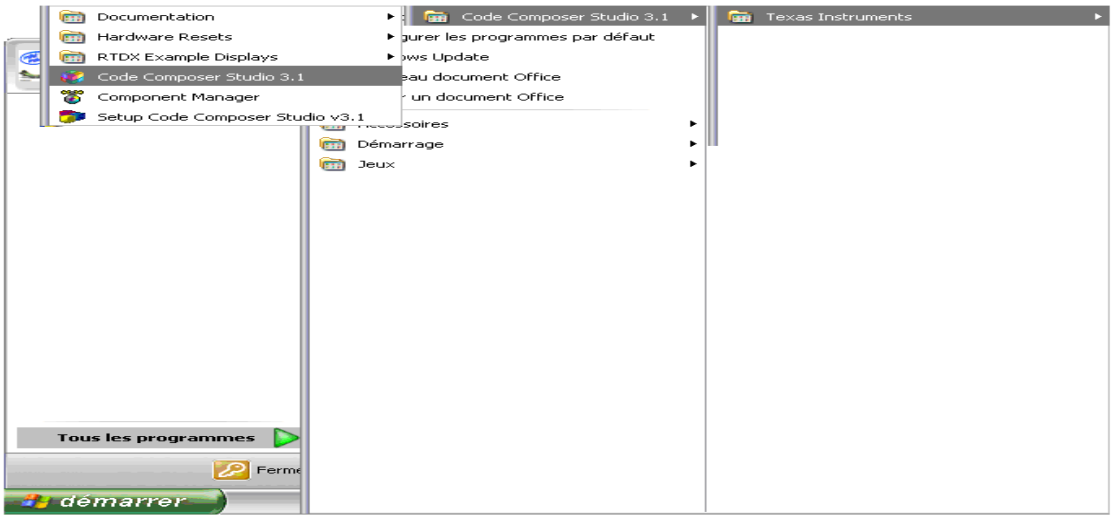
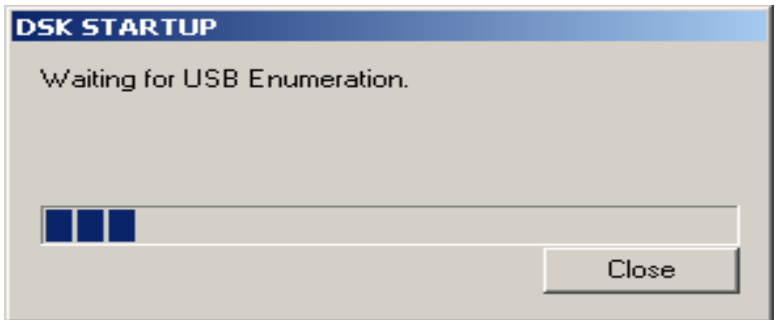


Figure 3.3 : Etapes d'exécution de CCS

La fenêtre suivante apparaît lors du lancement de la CCS ou de l'utilitaire de diagnostic indiquant l'état de recensement.



- Après l'exécution du code composer studio on remarque une image apparaît sur l'écran :

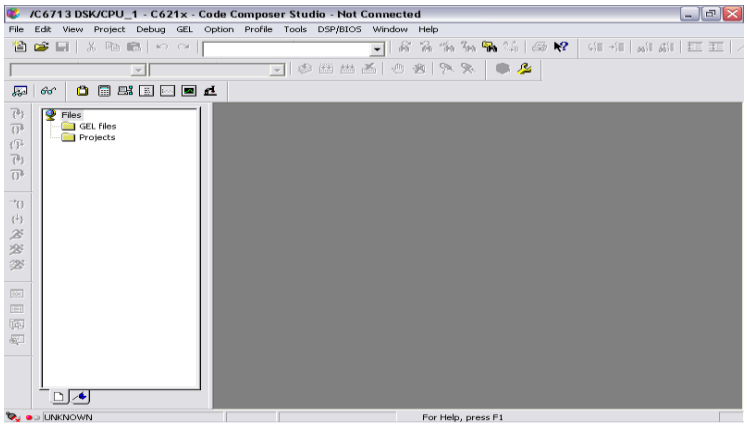


Figure 3.4 : Exécution du Code Composer Studio

4-1 Mode de simulation

Pour utiliser le CCS V3.3 en mode simulateur C6713, sélectionnez ‘C6713 Device Cycle Accurate Simulator’ de la liste des cartes. Il doit être ajouté et montré sur le côté gauche de l'écran de configuration comme illustré dans la Figure 3.5. Pour modifier les propriétés de la mémoire, on clique sur l'icône "TMS320C6713" comme le montre la figure 3.5 avec la flèche rouge et sélectionnez " Properties ". La propriété "Detect Reserved Memory Access" doit être sélectionnée sur "No". Appuyez sur "OK" pour revenir à l'écran de configuration. Appuyez sur “Save & Quit”. CCS est prêt à fonctionner en mode simulateur C6713.

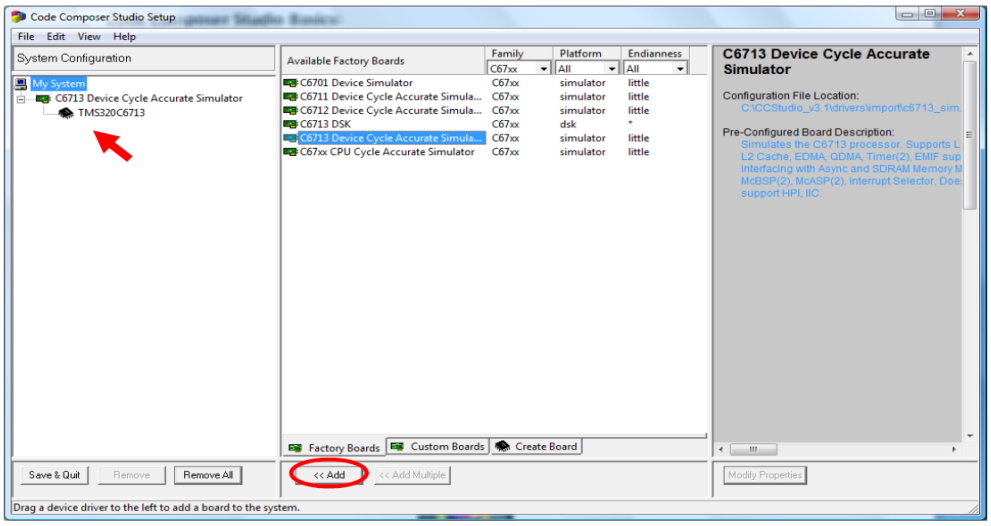


Figure 3.5 : Ecran setup du CCS

4-2 La carte C6713 DSK

Si on veut utiliser le logiciel CCS pour programmer le kit DSK C6713, alors on doit choisir le "C6713 DSK" de la liste des cartes de l'écran de configuration. On appuie ensuite sur le bouton "Add". On doit voir "C6713 DSK" sur le côté gauche de l'écran de configuration comme indiqué dans la Figure 3.6. On appuie sur “Save & Quit”. CCS est maintenant prêt à programmer et exécuter les programmes sur le kit DSK C6713. **[9]**

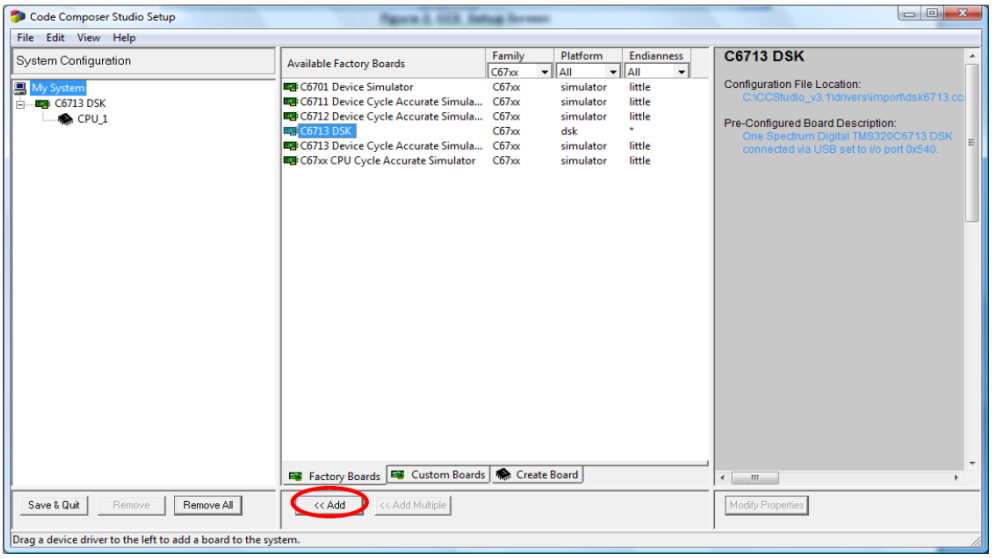


Figure 3.6 : Sélection du DSK C6713

III-5 Programmation en C/C++

5-1 Création d'un projet LAB4.pjt

Dans **CCS**, un projet est créé pour chaque programme exécutable. Le projet emmagasine toutes les informations exigées.

Pour créer un fichier il faut appuyer sur : Project ⇨ New

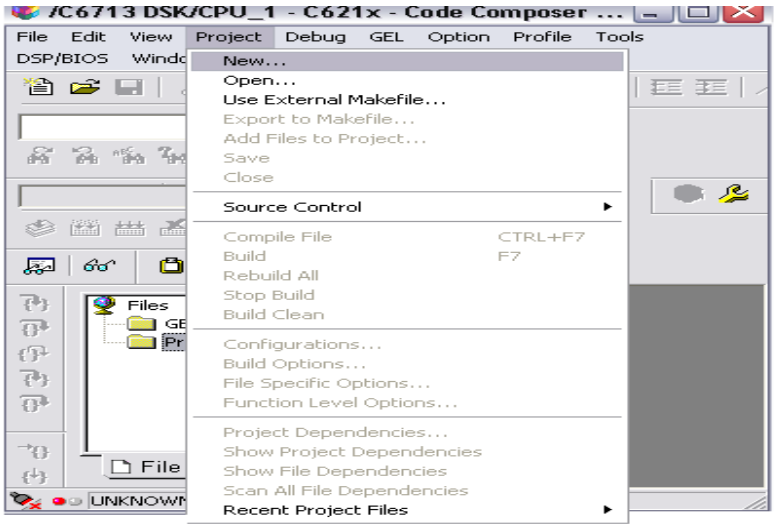


Figure 3.7 : Ouverture d'un nouveau projet

- Le message de sollicitation suivant apparaît :

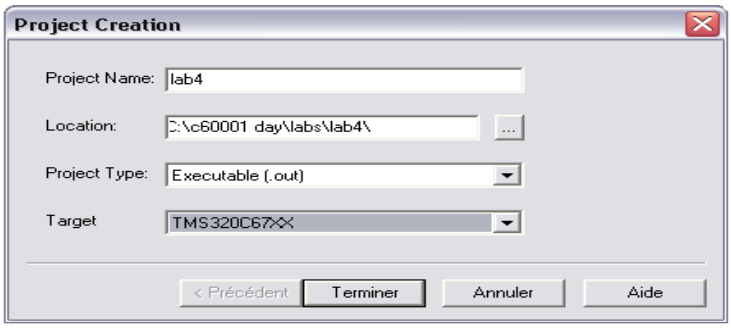
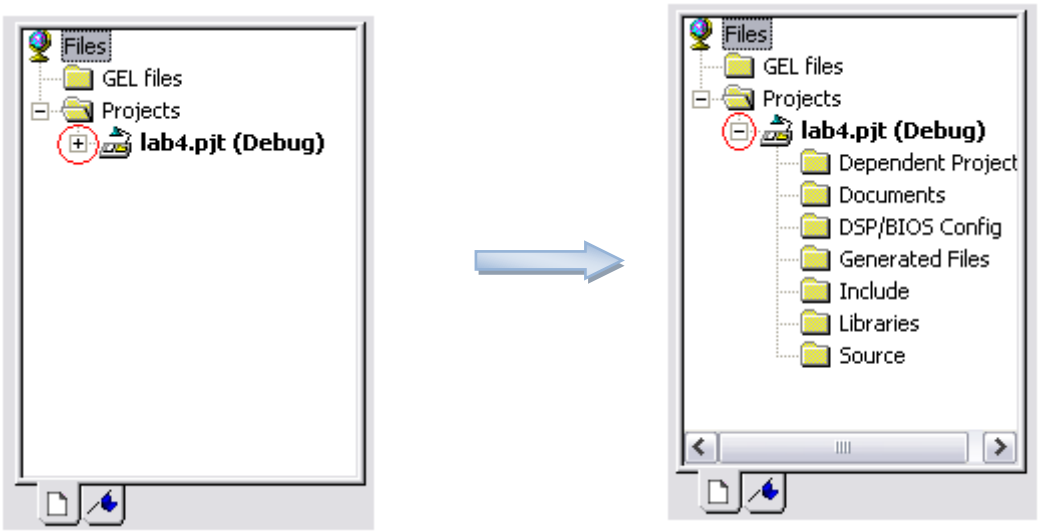


Figure 3.8 : Exemple de création de projet sous CCS

- Dans le champ « Project Name » Compléter le nom de projet dans la case ‘’Project name ‘’.

  Par exemple « Lab4 »
- Dans le champ **« Location »**, rechercher le dossier actif qui a été crée.
- Dans le champ « Project type » sélectionner ‘’Exécutable [.out] ‘’.
- Dans le champ « Target » sélectionner la cible (Dans ce cas ‘’TMS320c67xx’’).
- Cliquer sur ‘’terminer ‘’. **[5]**
- Vérifier que le projet récemment créé est ouvert dans ‘’CCS ‘’en cliquant sur le signe ‘’+ ‘’ à côté du dossier des projets dans la fenêtre d'affichage du projet.



Avant d'augmenter le projet **Après l'extension du projet**

5-2 Création du fichier CDB

Pour la création d'un nouveau fichier **CDB**, il faut :

Appuyer sur (File ⇨ New ⇨ DSP/Bios Configuration…)

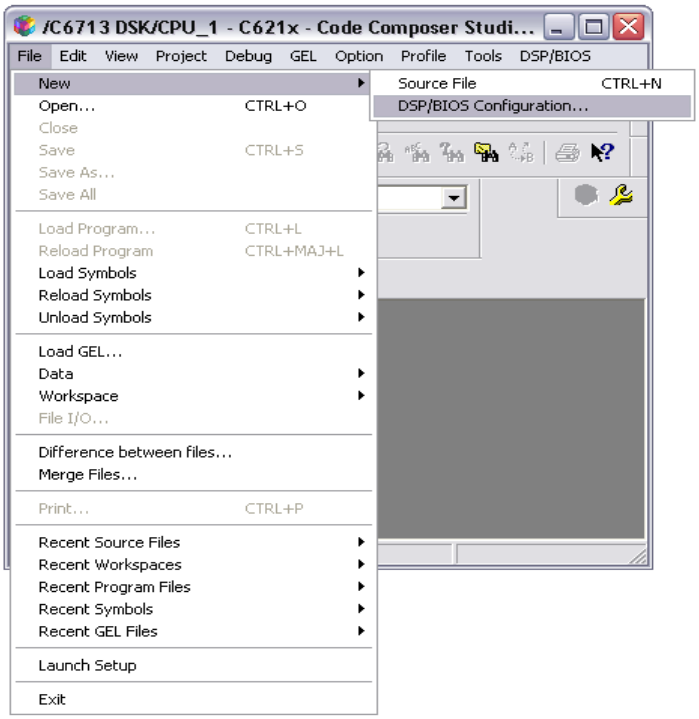


Figure 3.9 : Création d'un fichier CDB

- Sélectionner le modèle CDB approprié au système.
- Dans le cas proposé sélectionner ‘’DSK6713. Cdb ‘’.

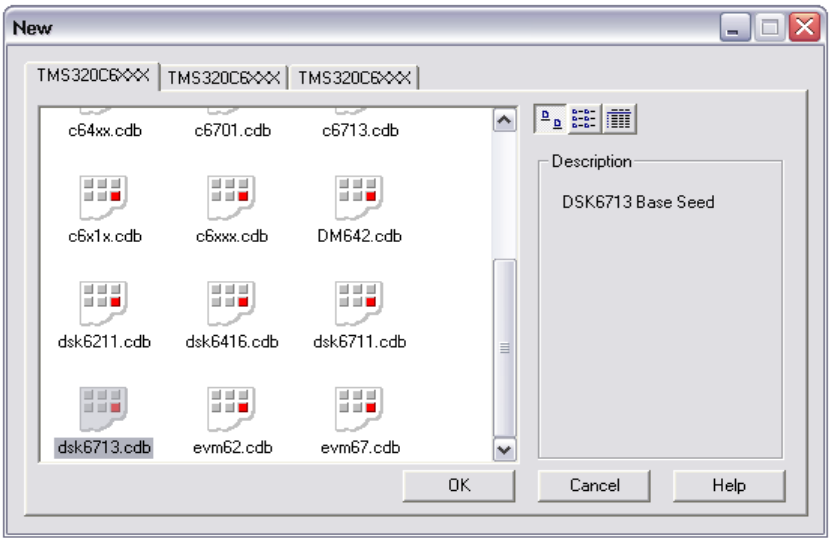


Figure 3.10 : Sélection du CDB

- Puis le message de sollicitation suivant apparaît :

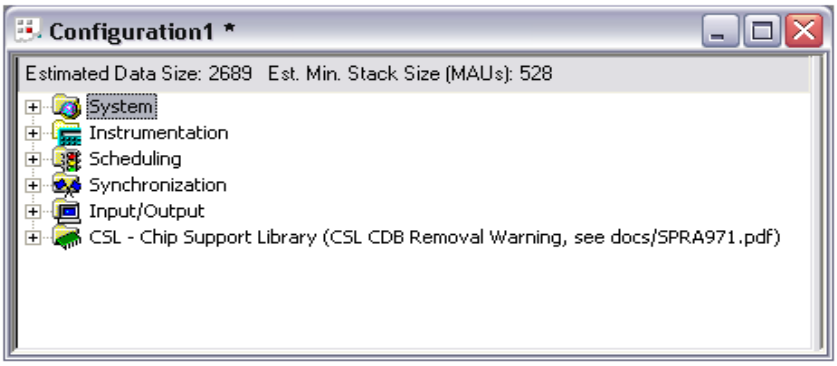
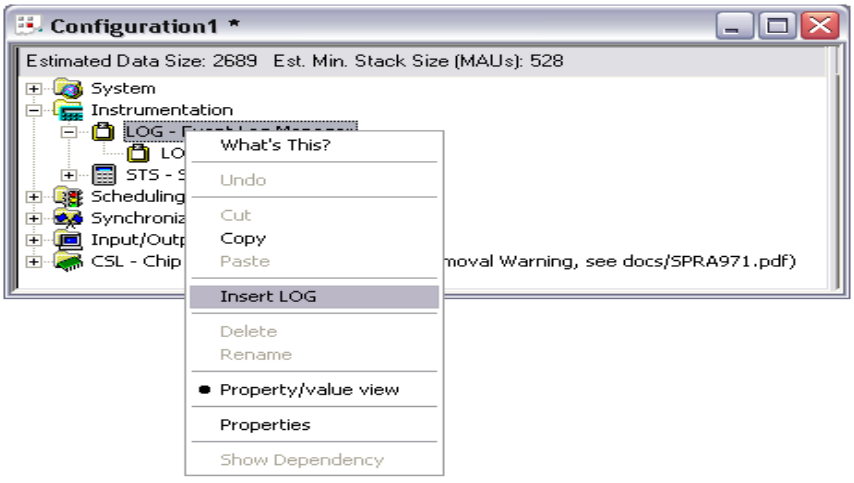


Figure 3.11 : Fenêtre apparente après la sélection du CDB

L'information est rapportée à ce **Lab** par un click sur le bouton droit de la souris sur ''event log manager '' et sélectionner ''insert LOG ''.



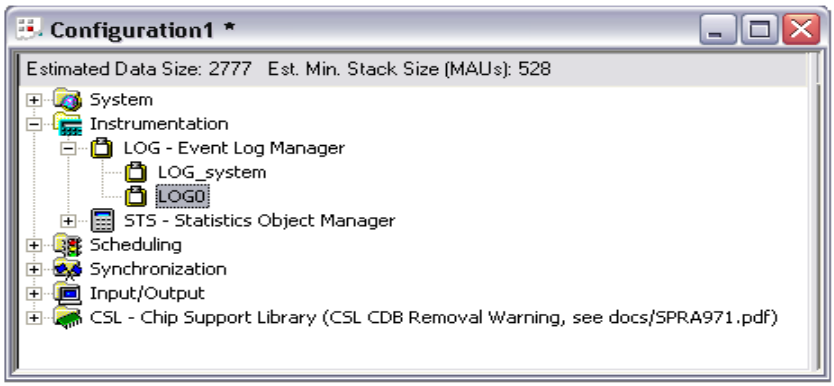


Figure 3.12 : Insert LOG

- Pour sauvegarder le fichier **CDB** il faut appuyer sur: File ⇨ Save as

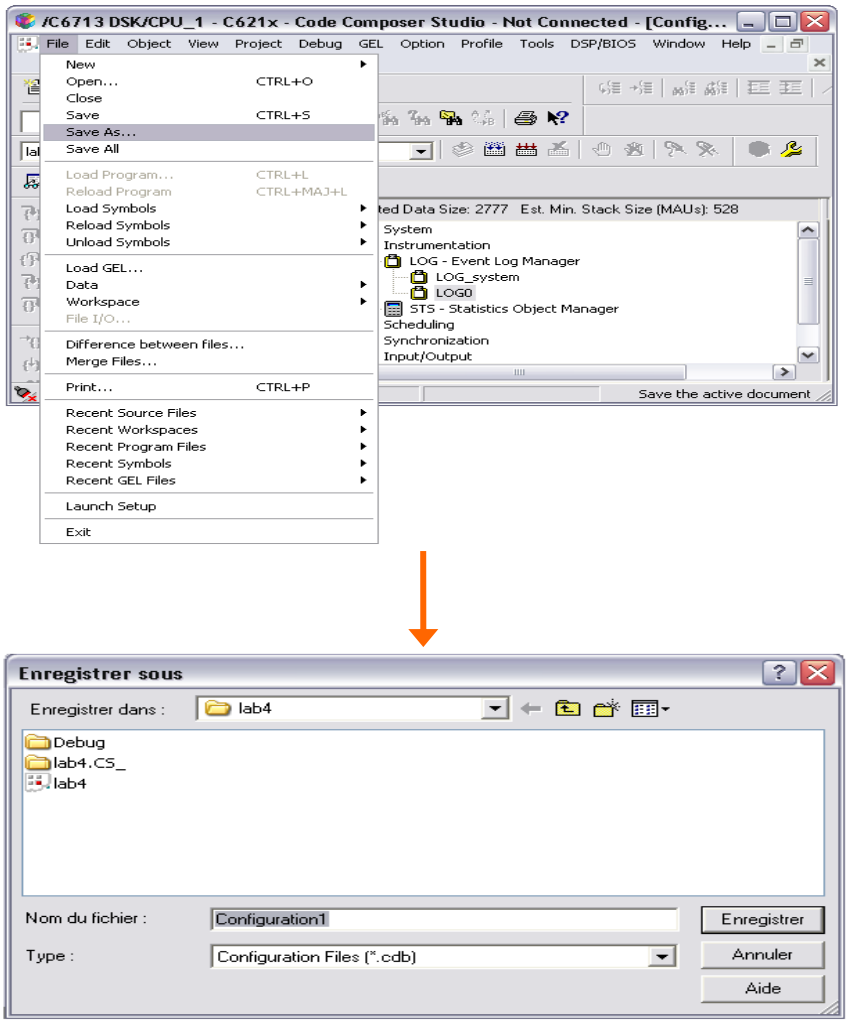


Figure 3.13 : Sauvegarde du fichier CDB

Ajouter les fichiers suivants au projet :

- lab4. C
- lab4. Cdb

Cliquer sur le bouton droit de la sourie sur « Lab4.pjt ». **[5]**

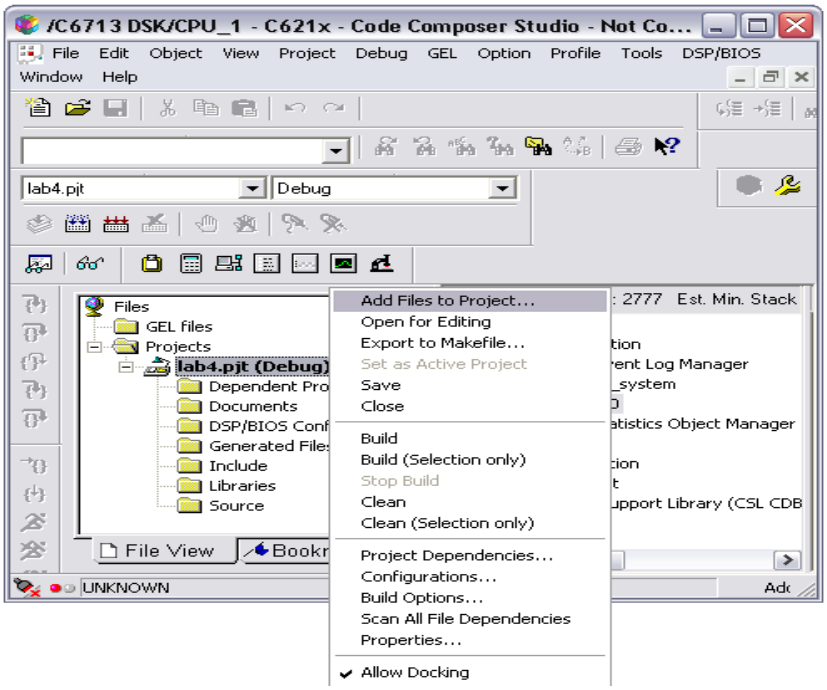


Figure 3.14 : L'ajout de programme source

Lorsque ces fichiers ont été ajoutés le projet devrait être comme :

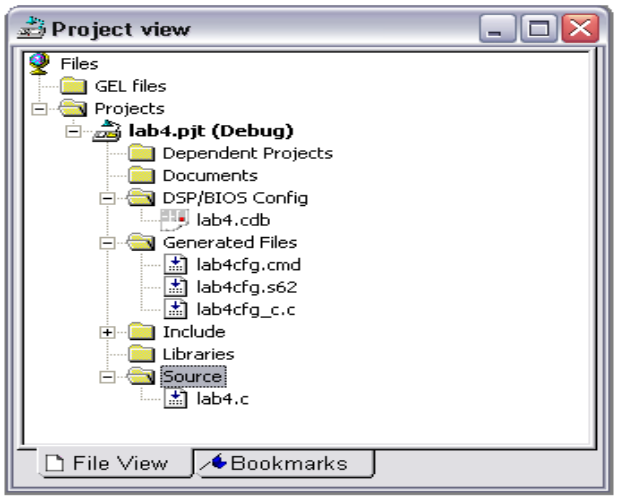


Figure 3.15 : Vérification du programme source

5-3 Modification de la « Debug Configuration »

Pour un débogage facile, il faut utiliser la ‘’Debug ‘’ configuration, ce ci devrait être par défaut.

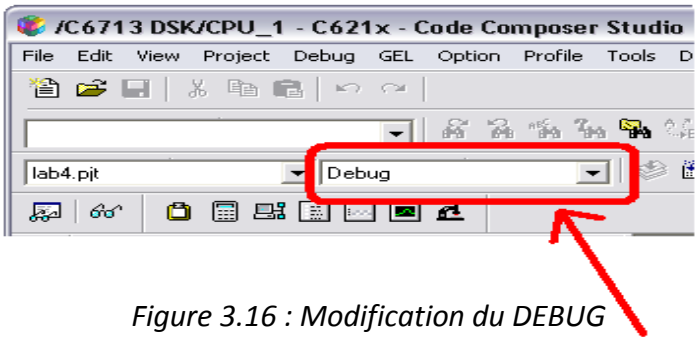


Figure 3.16 : Modification du DEBUG

- Puis Sélectionner project ➪ Build option.

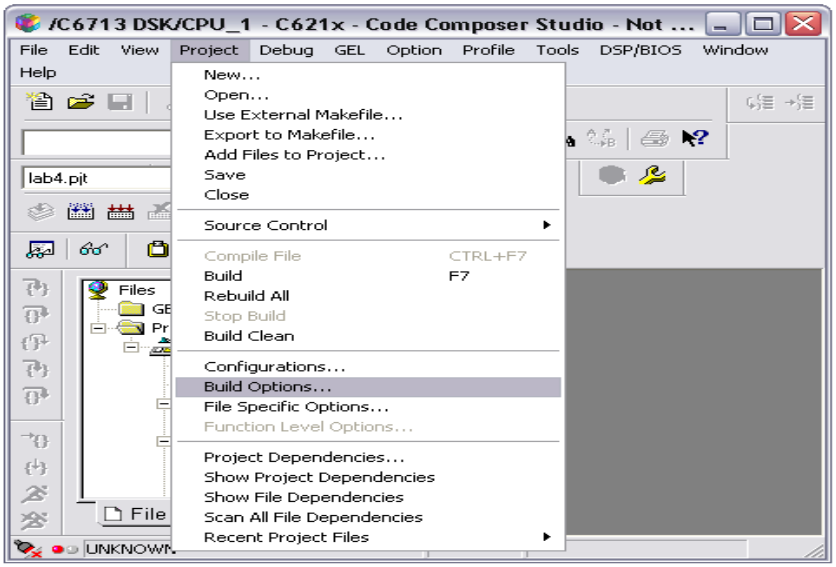
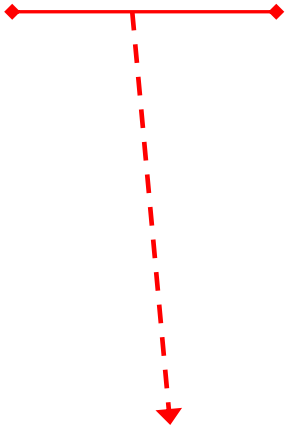


Figure 3.17 : Build Option

- Après sélectionner ‘’ Preprocessor ‘’.



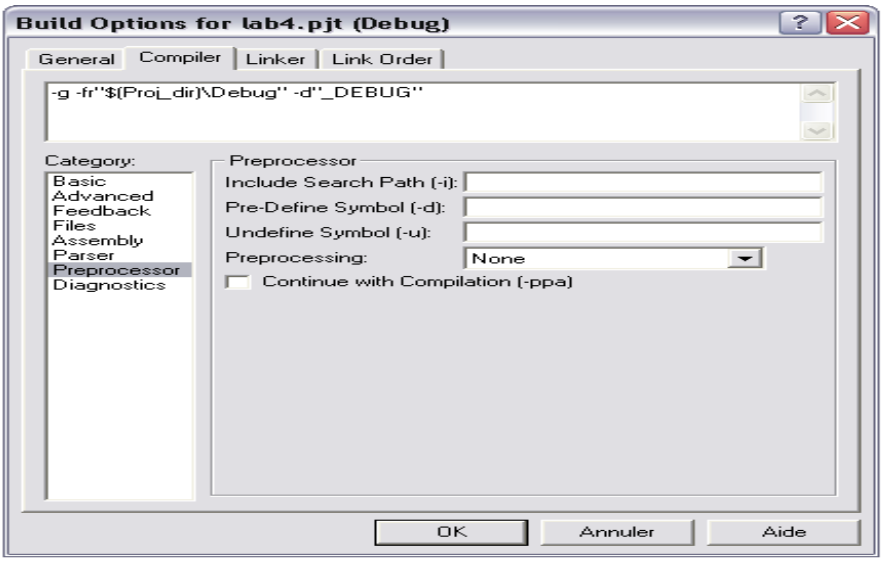


Figure 3.18 : Sélection du Preprocessor

- Taper ‘’ CHIP_6713 ‘’ dans la case « Pre-Define Symbols {-d} ».
- Puis cliquez sur Ok

Le message demandé apparaît :

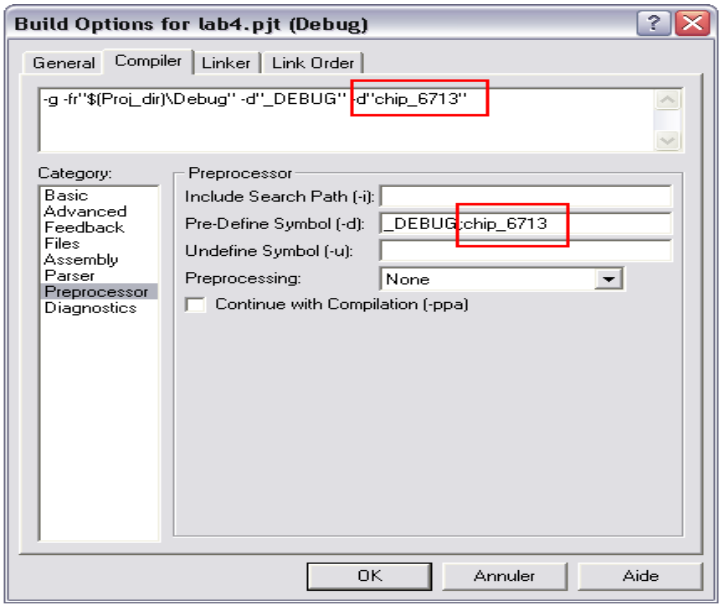


Figure 3.19 : Sélection du CHIP_6713

5-4 Building du Programme

Pour élaborer un programme, il faut appuyer sur : Project ⇨ Build

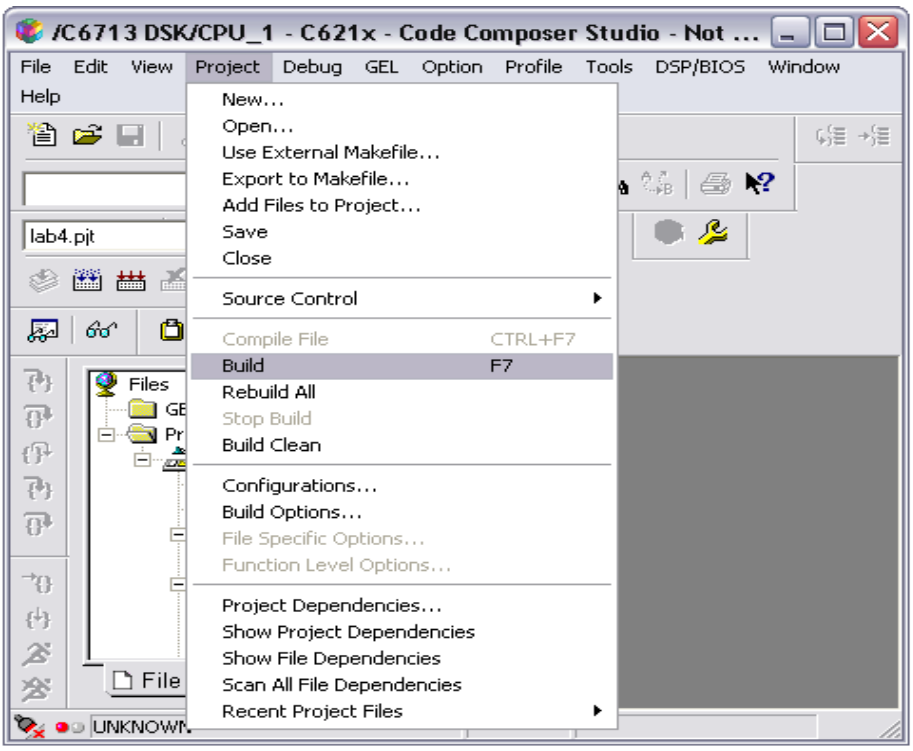


Figure 3.20 : L'option Build

La fenêtre suivante apparaît au bas du contexte de ''studio compositeur de code ''

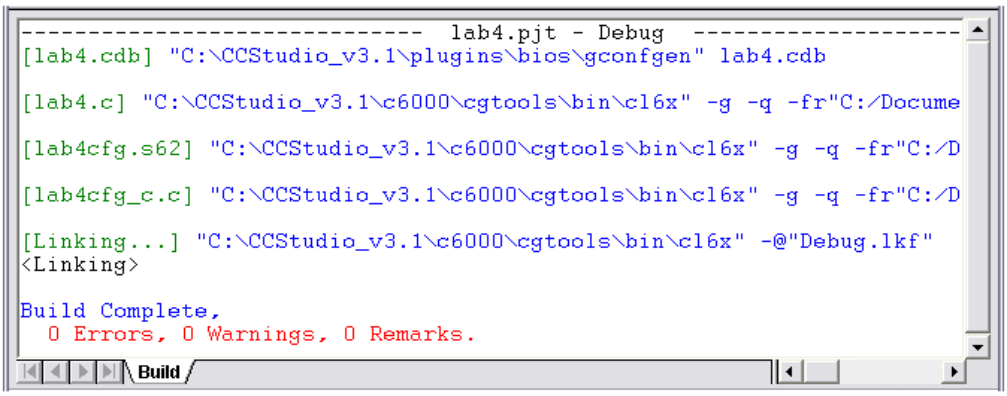


Figure 3.21 : Résultat du build

Dans le cas où des erreurs existent dans le code, elles seront énumérées.

5-5 Etape à suivre pour ouvrir un projet nommé led.pjt existant

• Aller au Projet> Ouvrir> Exemples> dsk6713> bsl> led> led.pjt.

• Sélectionnez-le et cliquez sur Ouvrir

• Cliquez sur '+' comme vu précédemment led.pjt.

• Cliquez sur '+' comme vu précédemment dossier source.

• Double-clique sur led.c, côté droit vous verrez C Code.

• Pour la construction du projet, cliquez sur l'icône Régénérer tout comme indiqué dans la figure ci-dessous.

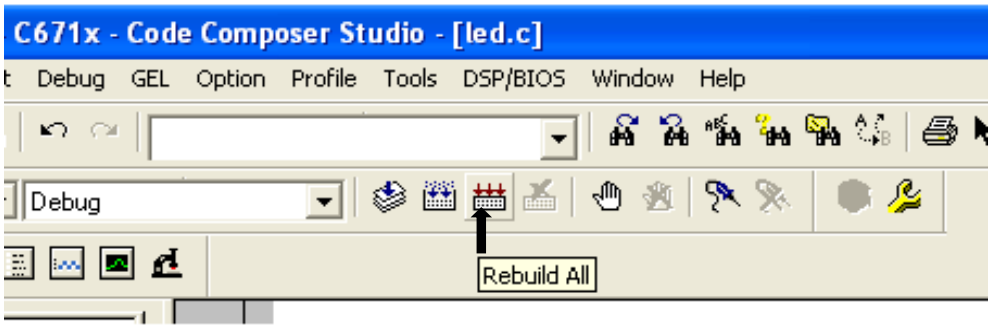


Figure 3.22 : L'option rebuild

• Vérifier pour 0 erreurs sur la fenêtre de sortie.

• Allez dans Fichier> Charger> Debug Dossier> led.out Fichier

• Exécutez le programme en sélectionnant debug> Exécuter ou utiliser la touche F5.

• Vous verrez la LED clignotante sur la carte DSP. **[5]**

III-6 Exécution du programme

Ce premier code est prêt à être compilé et exécuté. Pour compiler le code, suivre les étapes "Project ➔ Build". S'il n'y a pas d'erreurs dans le code, CCS génère un message semblable à celui de la figure 3.23.

```
Build Complete,
  0 Errors, 2 Warnings, 0 Remarks.
```

Figure 3.23 : Compilation réussie (Successful code building)

Pour exécuter un programme, on doit suivre les étapes suivantes :

File ➔ Load Program ...: Sélectionner "experiment2_demo.out" sous le répertoire debug.

Debug ➔ Reset CPU

Debug ➔ Restart

Debug ➔ Go Main

Debug ➔ Run

6-1 Insertion du point d'arrêt (Break point)

Pour insérer un point d'arrêt dans le programme, vous devez déplacer le curseur à l'endroit désiré. Cliquez sur l'icône en haut de l'écran comme illustré dans la Figure 3.24.

Figure 3.24 : Insertion de point d'arrêt (break point)

6-2 Ajout d'une fenêtre de surveillance (Watch Window)

Pour surveiller ou modifier les valeurs des variables, on peut utiliser la fenêtre de surveillance. Pour ajouter une variable à la fenêtre de surveillance, on fait un clic droit sur la variable et on sélectionne "Add to watch window" comme indiqué dans la Figure 3.25.

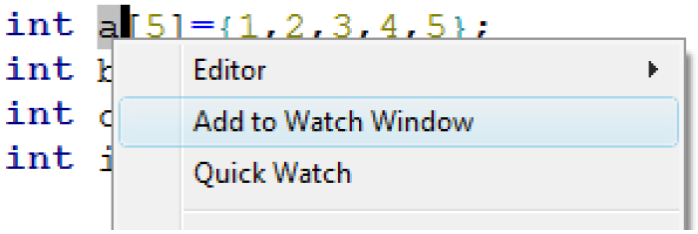


Figure 3.25 : Ajout de fenêtre de surveillance

On peut visualiser les valeurs et les changements dans la fenêtre de surveillance en bas à droite de l'écran du logiciel CCS, voir la Figure 3.26.

Name	Value	Type	Radix
a	0x0000F034	int[5]	hex
[0]	1	int	dec
[1]	2	int	dec
[2]	3	int	dec
[3]	4	int	dec
[4]	5	int	dec

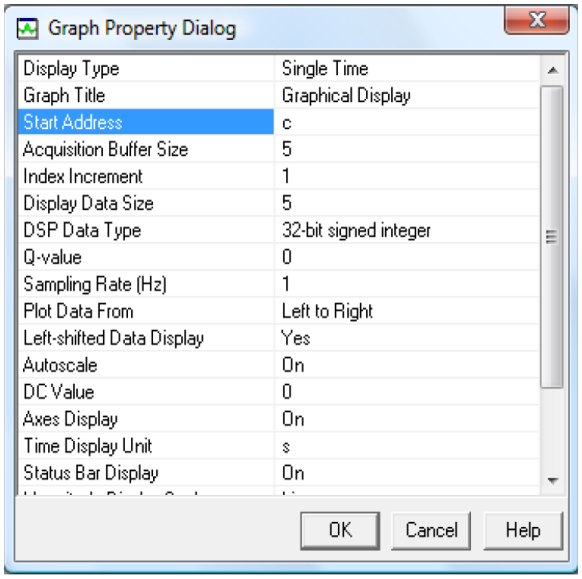
Figure 3.26 : Visualisation et changement des valeurs dans une fenêtre de surveillance

6-3 Plots

Afin de visualiser un vecteur sous forme de graphe, on utilise : "View➔ Graph➔ Time/Frequency...". Sur la fenêtre pop-up, remplir:

- Start Address: nom du vecteur à tracer.
- Acquisition Buffer Size: taille du vecteur.
- Display Data Size: taille des données à afficher sur le graphe.
- DSP Data Type: type du vecteur (integer ou float).
- Sampling Rate: fréquence d'échantillonnage du signal dans le vecteur.

Par exemple, pour visualiser le vecteur 'c' dans le programme précédant, on règle les paramètres comme montré dans la figure 3.27. Le graphe à obtenir est semblable au graphe de la figure 3.28.



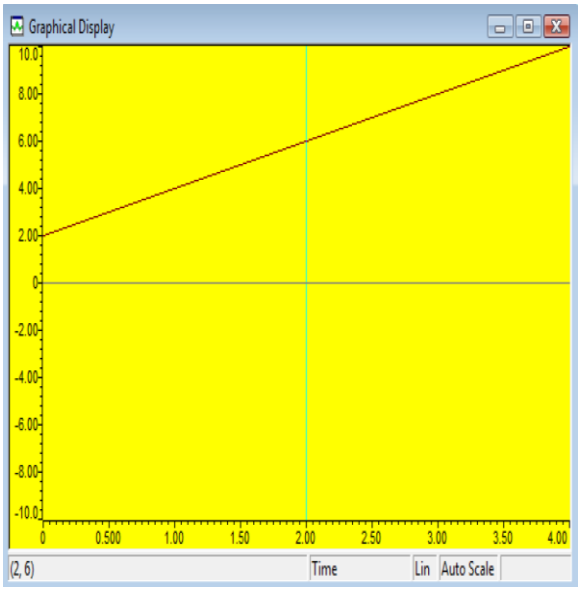


Figure 3.27 : Réglage des propriétés du graphe

Figure 3.28 : Graphe

6-4 Images

Pour visualiser une image sous CCS on doit aller à "View➔ Graph➔ image". On remplit les paramètres de la fenêtre pop-up comme indiqué dans la figure 3.29.

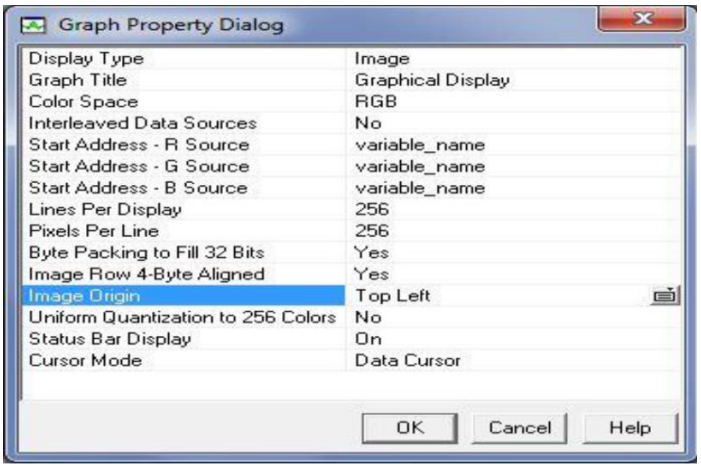


Figure 3.29 : Paramètres de visualisation d'images

6-5 Enregistrement de données

Pour sauvegarder les matrices dans un fichier (data file), on doit utiliser "File➔ Data➔ Save". Dans la figure 3.30, on choisit le format d'enregistrements des données.

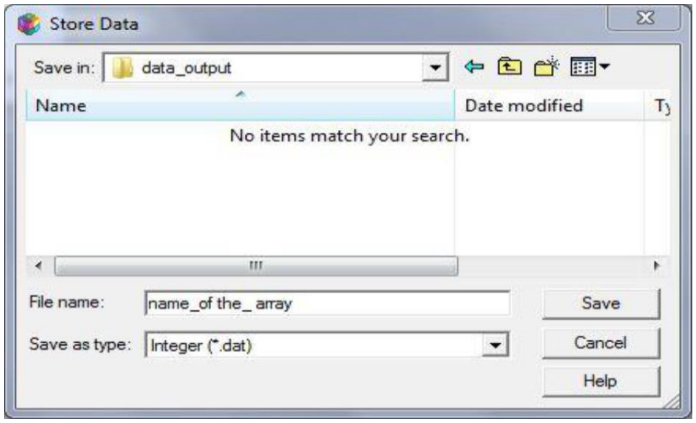


Figure 3.30 : Enregistrement de data-file

Une fois le bouton "save" est actionner, une nouvelle fenêtre apprêtera (figure 3.31). Il faut mettre le nom de la matrice dans le champ "Address" et sa taille dans le champ "length". CCS enregistrera les données dans un fichier ".dat".

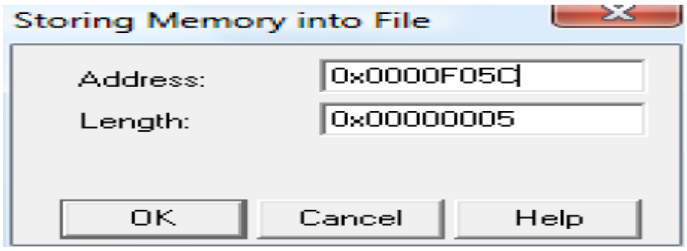


Figure 3.31 : Enregistrement de variables

III-7 Programmation à l'aide du Matlab/Simulink

Simulink est un logiciel de simulation de systèmes dynamiques, il s'agit d'une extension du logiciel de calcul Matlab qui propose de nombreuses bibliothèques de model intéressant. Le tableau suivant indique les versions de Matlab et de code composer studio qui sont compatibles entre eux :

Version Link for CCS	Version Matlab	Version CCS
3.0	2007A	• CCS 3.2 pour C64xx • CCS 3.1 pour C2000 ; C5000 ; C6000
2.1	R2006B	• CCS 3.2 pour C64xx • CCS 3.1 pour C2000 ; C5000 ; C6000
2.0	R2006A+	• CCS 3.1 pour C2000 ; C5000 ; C6000
1.5	R2006A	• CCS 3.1 pour C2000 ; C5000 ; C6000
1.4.2	R14SP3	• CCS 3.0 pour C6000 • CCS 2.2 pour C2000 ; C5000 ; C6000

Tableau 3.1 : Compatibilité entre Matlab et CCS

En utilisant Real-Time Workshop, il sera permis de générer un code efficace pour les processeurs C6000 directement à partir des modèles Simulink. L'embedded Target for TI C6000 DSP permet le prototypage rapide des logiciels temps réel pour DSP Texas instruments (TI) C67xx a virgule flottante et C62x et C64x a virgule fixe. **[5]**

7-1 Real-Time Workshop

Real-Time Workshop est un outil de Matlab qui permet de générer automatiquement un code 'C' à partir d'un modèle simulink. De plus, il permet de produire un code optimise, portable et personnalisable d'après les modèles générés par simulink. Il supporte tout type de systèmes, c'est-à-dire continus, discrets ou hybrides. Il permet la simulation du système étudie en mode externe, c'est-à-dire en s'interfaçant avec un pc cible. **[10]**

7-2 Embedded IDE Link Software

Embedded IDE Link fournit un raccordement entre Matlab et un processeur dans CCS. Nous pouvons employer des objets pour commander et manœuvrer une application de traitement

des signaux utilisant la puissance informatique du logiciel matlab, et il nous permet d'employer des fonctions de matlab pour communiquer avec le code composer studio, ainsi que les données stockées dans la mémoire. Avec ces fonction on peut transférer et récupérer les données à partir de code composer studio, cette approche peut nous aider à corriger et développer notre application.

Une autre utilisation possible pour automation, est de créer les scripts de Matlab qui vérifient et examinent les algorithmes fonctionnant dans leur exécution finale sur le processeur utilisé. **[10]**

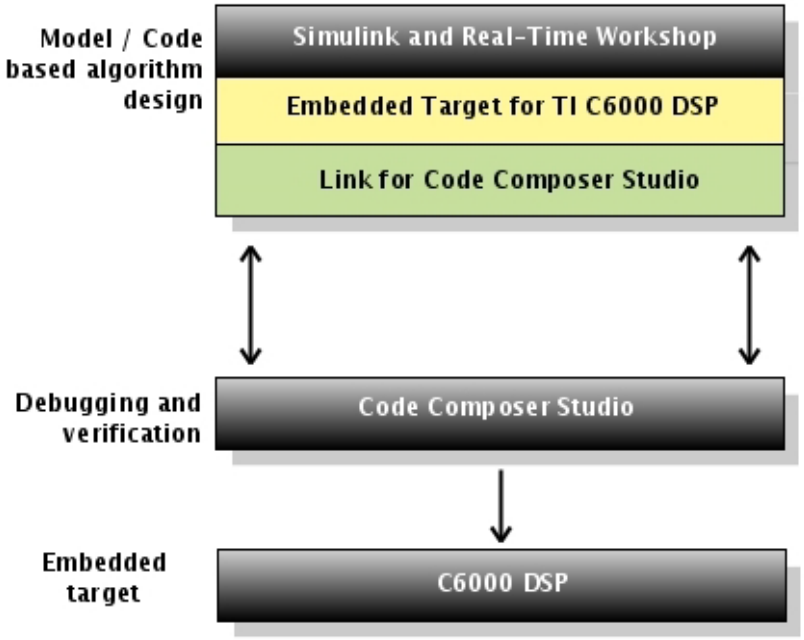


Figure 3.32 : Lien entre Matlab et CCS

7-3 L'environnement de développement intégré (IDE)

Avec IDE (L'environnement de développement intégré), nous pouvons utiliser Matab et Simulink pour analyser, profiler et corriger le comportement d'exécution du code dans code composer studio, de cette façon IDE automatise le déploiement de l'application embarquée complète, et le rend facile pour évaluer des différences possibles entre le modèle de simulation et les résultats d'exécution de code de processeur.

Pour présenter les techniques et les outils disponibles dans le 'Link for CCS' pour l'usage du RTDX, les procédures suivantes, emploient plusieurs méthodes dans le logiciel de lien pour configurer le processeur, ouvrir et activer une voie, envoyer les données a la cible, et nettoyer après avoir fini l'essai. **[11]**

Fonction	Description
CCSDSP	• Créer un lien a CCS IDE and RTDX.
Cd	• Changer le répertoire de travail CCS IDE de Matlab.
Open	• Changer les fichiers programme dans CCS IDE.
Run	• Exécutez le programme charge sur le processeur.

Tableau 3.2 : Fonctions du Liens pour CCS IDE

III-8 Les extensions des fichiers CCS

- **xx.Mak** Fichier Projet
- **xx.C** Ce fichier contient code source qui fournit les principales fonctionnalités de ce projet.
- **xx.h** Ce fichier déclare la mémoire C-structure aussi bien que définit les constantes exigées
- **xx.asm** Ce fichier contient des instructions d'assemblage. ‘’Fichier écrit en assembleur
- **xx.cmd** Ce fichier construire les sections pour la mémoire. ‘’Fichier de linkage’’
- **xx.obj** Fichier désassemblé
- **xx.out** Fichier implantation dans la cible
- **xx.lib** Fichier de librairie standard dont on peut se passer si on utilise un fichier .cdb
- **xx.cdb** Fichier de configuration lié à une cible permettant de configurer les interruptions,….
- **xx.pjt** Ce fichier contient toutes votre construction du projet et options de la configuration.
- **xxcfg.s62** fichier assembleur de configuration généré automatiquement lors du « build » lorsqu’on utilise un fichier .cdb
- **xxcfg.h62** Fichier assembleur d’en tête inclus dans XXcfg.s62 et généré automatiquement lors du « build » lorsqu’on utilise un fichier .cdb **[5]**

III-9 Les Fichiers de support

Les fichiers de support suivants situés dans le support de dossier (sauf les fichiers de bibliothèque) sont utilisés pour la plupart des exemples et des projets:

- ✓ **C6713dskinit.c:** contient des fonctions pour initialiser le DSK, le codec, les ports série, et pour I / O. Il n'est pas inclus avec le CSC.
- ✓ **C6713dskinit.h:** fichier d’en-tête avec les prototypes de fonctions. Des fonctionnalités telles que ceux utilisés pour sélectionner l'entrée micro au lieu d'entrée de ligne (par

défaut), le gain d'entrée, et ainsi de suite sont obtenus à partir de ce fichier d'en-tête (modifié à partir d'un fichier similaire inclus avec CCS).

- ✓ **C6713dsk.cmd:** échantillon linker fichier de commandes. Ce fichier peut être générique changé lors de l'utilisation mémoire externe à la place de la mémoire interne.
- ✓ **Vectors_intr.asm:** une version modifiée d'un fichier vectoriel inclus avec le CSC pour traiter les interruptions. Douze interruptions, INT4 a INT15 sont disponibles, et INT11 est sélectionné dans ce vecteur file. Ils sont utilisés pour pilotées par interruption programmes.
- ✓ **Vectors_poll.asm:** fichier vectoriel pour les programmes utilisant scrutin.
- ✓ **rts6700.lib, dsk6713bsl.lib, csl6713.lib:** moment de l'exécution, du conseil et fichiers de bibliothèque, de support, de puce. **[5]**

Conclusion

La programmation des DSP nécessite de vastes connaissances techniques (électronique, informatique, etc.), et il en résulte de nombreuses applications dans des domaines les plus Variés, l'importance du langage C dans les systèmes DSP a considérablement augmenté au cours des derniers ans. En outre, TI a travaillé dur pour fournir facile à utiliser, Bien qu'étant le plus répandu, le langage C n'est pas le seul utilisable pour programmer un DSP, il existe quelques rares compilateurs ADA, FORTRAN et PASCAL.

4

CHAPITRE :

Algorithme de traitement d'image Embarqué sur TMS 320C6713

Introduction

Dans l'objectif du traitement rapide nous allons présenter et montrer les différents résultats de traitement numérique d'image : filtrage numérique 2D, Détection de contours et Segmentation des images numérique sous l'environnement de simulation MATLAB, ensuite on implémente ces Algorithmes sur une carte à base de DSP (TMS320C6713 DSK de Texas Instrument).

Concernant le passage de Matlab au code composer studio CCS, MATLAB a développé un outil ''Link for code composer studio'' qui lui permit de communiquer avec le code composer studio et on utilisant cet outil ; le model SIMULINK est traduit en projet écrit en C qui pourrai être exécuté sur CCS, pour des applications en temps réel.

IV-1 Model Simulink Proposé pour l'implémentation

1-1 Présentation du Model Simulink

Le model présente une architecture efficace pour la segmentation d'une image de voiture et extraire leurs matricule. L'architecture proposée offre une alternative à travers un outil graphique MATLAB. L'objectif principal de l'utilisation d'un outil avec une interface graphique de haut niveau sous les Blocks en fonction Matlab Simulink qui le rend très facile à manipuler par rapport au d'autres, Il existe des nombreuses différentes Techniques pour la segmentation d'image, mais tout ce que les techniques exigent un langage de haut niveau. Donc à l'aide des Opérations morphologiques, on peut construire notre model proposé pour la segmentation d'images. Ce modèle Utilise des différents Blocks de « Video and Image Processing Blockset», on présente ces différents blocks comme suit :

Block	Library	Quantity
Image From File	Video and Image Processing Blockset> Sources	01
Color space conversion	Video and Image Processing Blockset> Conversions	01
Median Filter	Video and Image Processing Blockset> Analysis & Enhancement	01
Video Viewer	Video and Image Processing Blockset> Sinks	04
Embedded MATLAB Function	Simulink> User-Defined Functions	01
Image data type conversion	Video and Image Processing Blockset> Conversions	02
Edge Detection	Video and Image Processing Blockset> Analysis & Enhancement	01

Tableau 4.1 : Différents blocs Simulink utilisés

1-2 Architecture du Model Simulink

L'architecture de notre Model Simulink proposé est présentée comme suit :

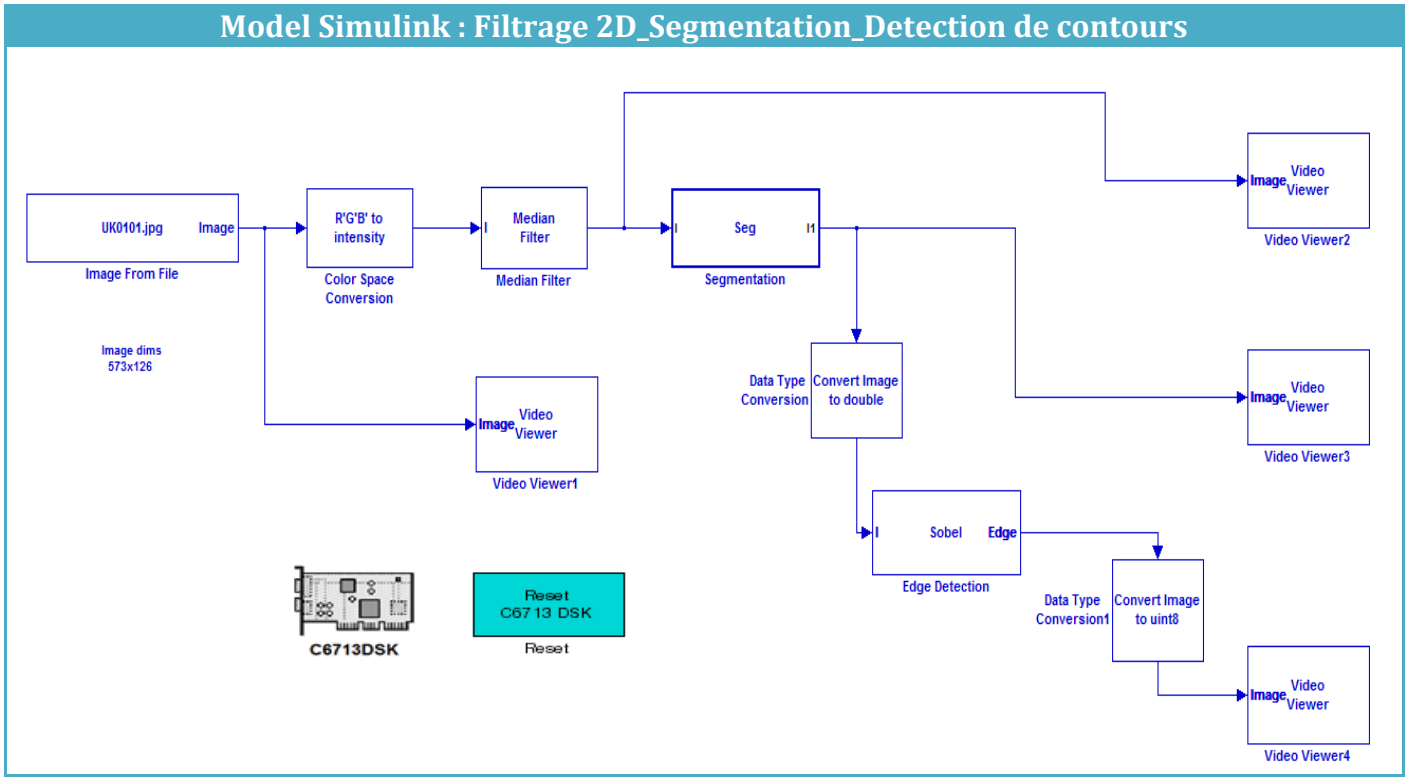


Figure 4.1 : Architecture du modèle Simulink

1-2-1 Les Paramètres de chaque block

Concernant les paramètres de chaque block, ils sont bien détaillé (Voir l'ANNEXE) sauf le bloc « **Embedded MATLAB Function** » nommé : « **Segmentation** » qui contient le code Matlab.

1-3 Simulation et résultats obtenus à l'aide de Matlab Simulink

Dans le model Simulink proposé, le premier block « Image from file » permet de définir le lien de l'image choisie et faire une lecture de fichier puis l'image sélectionnée sera converti de RGB à l'intensité (niveau de gris) à l'aide du block « Color space conversion », le filtre Médian de 3X3 permet d'éliminé le bruit lié à l'image pour avoir une image sans bruit (filtré) puis elle sera traité à l'aide du block « Segmentation », qu'il a un rôle de faire segmenté

l'image et extraire du matricule de voiture, cette image est converti en double à travers un block « Image Data Type Conversion ».

L'opération de détection de contours (filtre de Sobel) est faite avec le block « Edge detection » à la fin l'image traitée sera convertie en uint8 à l'aide du block « convert image to uint8 ».

Pour exécuter notre model Simulink proposé en mode Matlab Simulink il faut choisi le menu « simulation » puis on appuyé sur « Start » ou bien « Ctrl+T ». on obtient les résultats suivantes :

Matricule 01

(a)

(c)

(b) **(d)**

Matricule 02

(a)

(c)

(b) **(d)**

(a) : Image original (b) : Image filtré à l'aide d'un filtre Médian 3x3

(c) : Image segmenté (extraire du matricule) (d) : détection de contours d'un matricule

IV-2 Implémentation des Algorithmes sur TMS320C6713 DSK

Comme décrit précédemment, MATLAB a développé un outil ''Link for code composer studio'' qui lui permit de communiquer avec le code composer studio et on utilisant cet outil ; le model SIMULINK est traduit en projet écrit en C qui pourrai être exécuté sur CCS.

Sur Simulink, il y a toute une librairie contenant des blocks spécifiques pour chaque type de carte DSP, dans notre model on ajoute le block C6713, et on choisit le type de la carte et le type de processeur et la fréquence d'horloge.

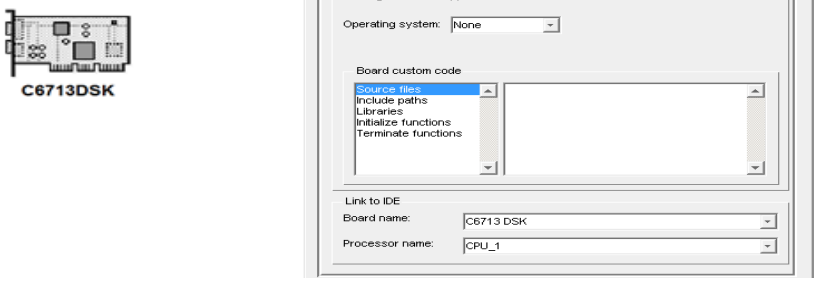


Figure 4.2 : Le Block du C6713

2-1 Instruction pour la connexion de la carte

Cette procédure est à réaliser avec attention chaque fois que pour une raison ou une autre la connexion USB ne fonctionne plus.

- Assurer que le code composer studio CSS est fermé.
- Brancher l'alimentation de la carte DSK : d'abord l'alimentation au secteur, puis l'alimentation continue à la carte.
- Attendre que les LEDs aient clignoté puis restent allumées.
- Brancher le câble USB.
- Lancer CCS (Code Composer Studio)

- Se connecter à la cible : Debug > Connect

Une icône verte doit apparaître en bas à gauche de la fenêtre de CCS pour préciser que la carte est bien connectée au PC.

Puis il faut faire une vérification du bon fonctionnement de la carte, si elle est correctement connectée au PC, Pour cela les étapes suivantes sont à suivre :

Vérification du bon fonctionnement de tous les périphériques de la carte DSK :
Double clique sur l'icône « 6713DSK Diagnostics Utility » puis sur le bouton « Start ».

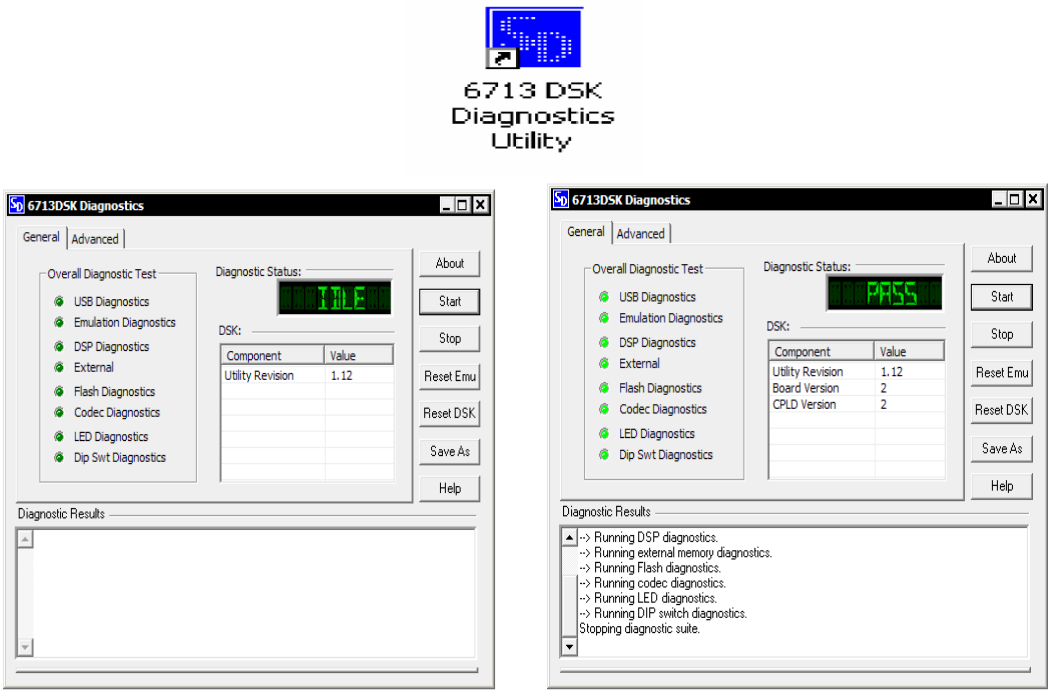


Figure 4.3 : Vérification de la bonne marche de la carte DSK

2-2 Les Paramètres de configuration

Pour régler les paramètres de la simulation, on clique sur « simulation » puis « Configuration Paramaters » ou bien sur Ctrl+E. La fenêtre suivante apparait :

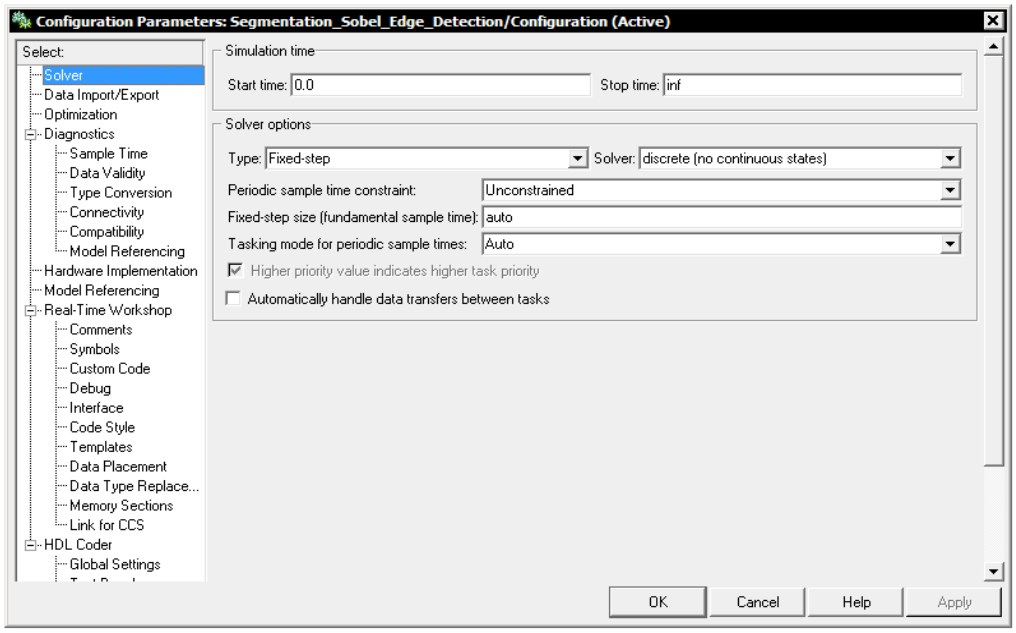


Figure 4.4 : fenêtre de la configuration des Paramètres

Sélectionner le champ « Real-Time Workshop » pour modifier les paramètres de l'application, dans le champ « Build action » sélectionner l'option « Creat_projet ».

N.B : il existe d'autres options sur ce champ qui permettent la création et le chargement du programme sur la carte DSK.

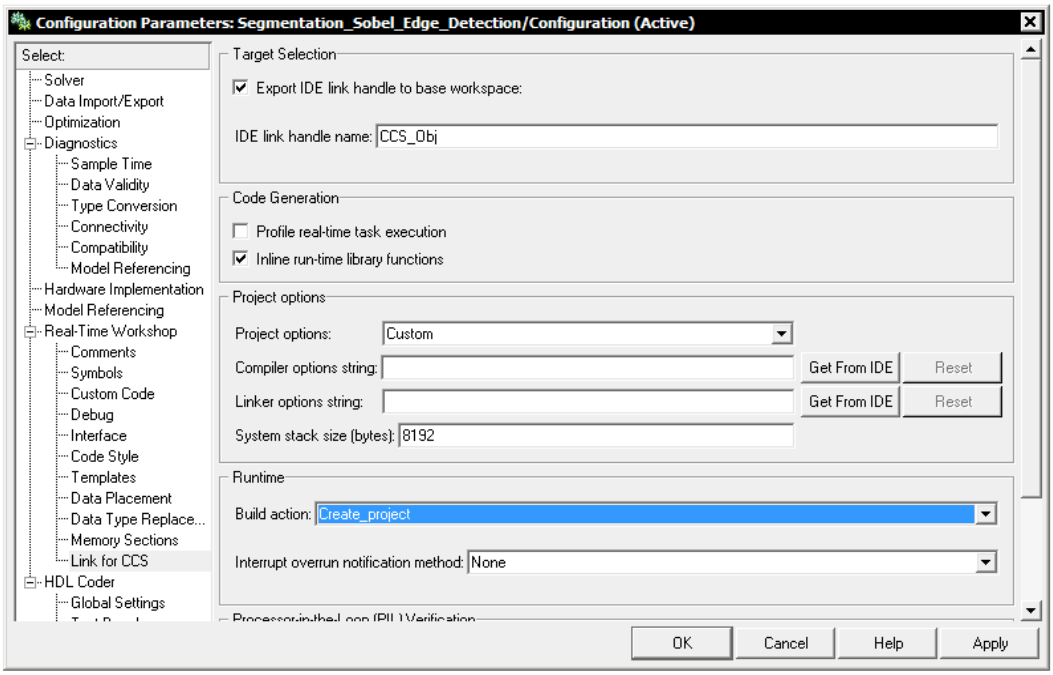


Figure 4.5 : Paramètre du link for CCS

2-3 Création d'un projet en « C/C++ » à partir d'un model Simulink

Pour démarrer la création du projet en cliquant sur « Incremental build » ou bien sur Ctrl+B, Matlab crée un lien avec le code composer studio et génère le projet écrit en « C ».

```
### Loading TLC function libraries

.....
### Initial pass through model to cache user defined code
.
### Caching model source code
.........................
### Writing header file Segmentation_Sobel_Edge_Detection_types.h
### Writing header file Segmentation_Sobel_Edge_Detection.h
.
### Writing header file Segmentation_Sobel_Edge_Detection_private.h
### Writing source file Segmentation_Sobel_Edge_Detection.c
### Writing source file Segmentation_Sobel_Edge_Detection_data.c
.
### Writing source file Segmentation_Sobel_Edge_Detection_main.c
### TLC code generation complete.
### Creating project in Code Composer Studio(tm)...
```

Figure 4.6 : Le passage du Simulink au « C/C++ »

2-3-1 Le programme en « C/C++ »

Concernant le programme en « C/C++ » crée par le Code composer studio veuillez trouver dans l'ANNEXE sous le nom « Segmentation_Sobel_Edge_Detection.c ».

2-3-2 Compilation du programme en « C/C++ »

Dans la liste du projet à gauche de la fenêtre du CCStudio, cliquez sur le programme (C/C++) ajouté. Ensuite dans le menu principal cliquez sur « Project » et choisissez « Compile File ».

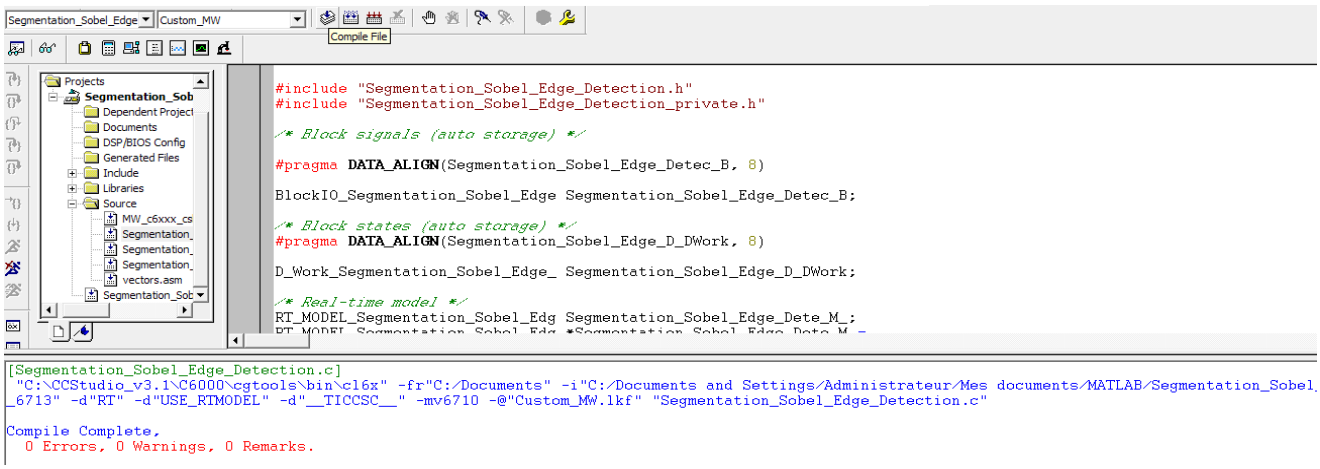


Figure 5.7 : Capture d'écran de l'opération de compilation

2-3-3 Build du programme en « C/C++ »

Après la compilation du programme, construisez le projet en cliquant dans le menu de « **Project »** puis sur « **Rebuild all** ».

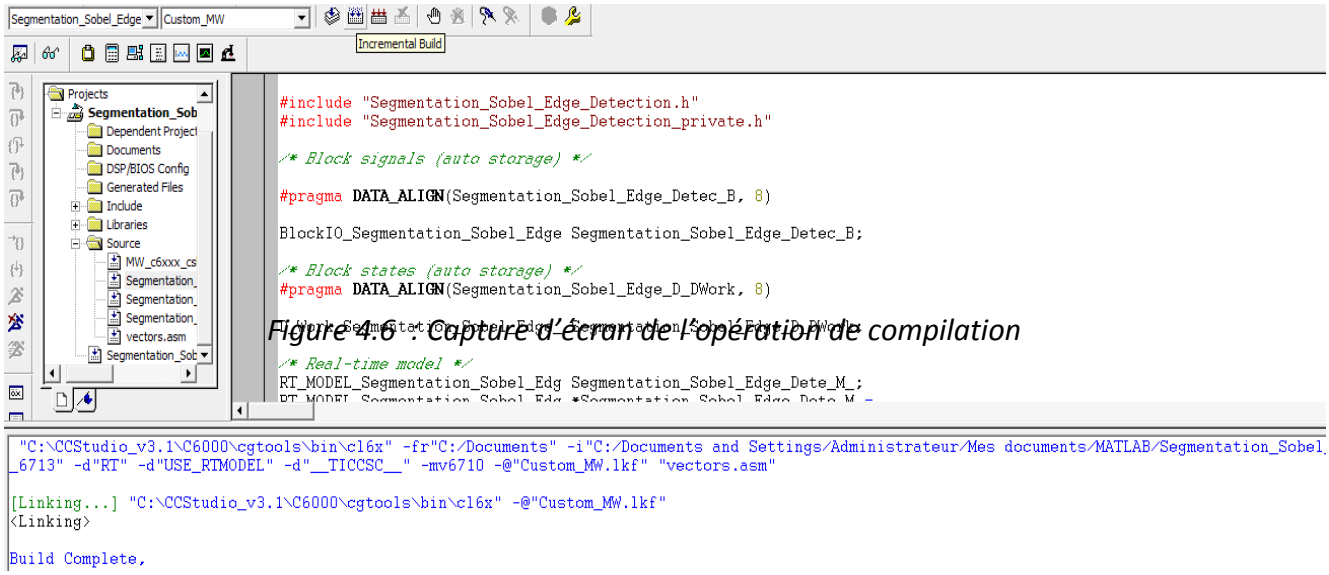


Figure 4.6 : Capture d'écran de l'opération de compilation

Figure 4.8 : Capture d'écran de l'opération Build

2-3-4 Chargement et exécution du programme en « C/C++ »

Après la construction du projet, le programme doit être chargé sur la carte. Dans le menu principal choisissez « **File »** et ensuite « **Load Program »**. Une nouvelle fenêtre s'ouvrira dans laquelle vous choisissez le fichier « Segmentation_Sobel_Edge_Detection.out » et cliquez sur « **Open »**.

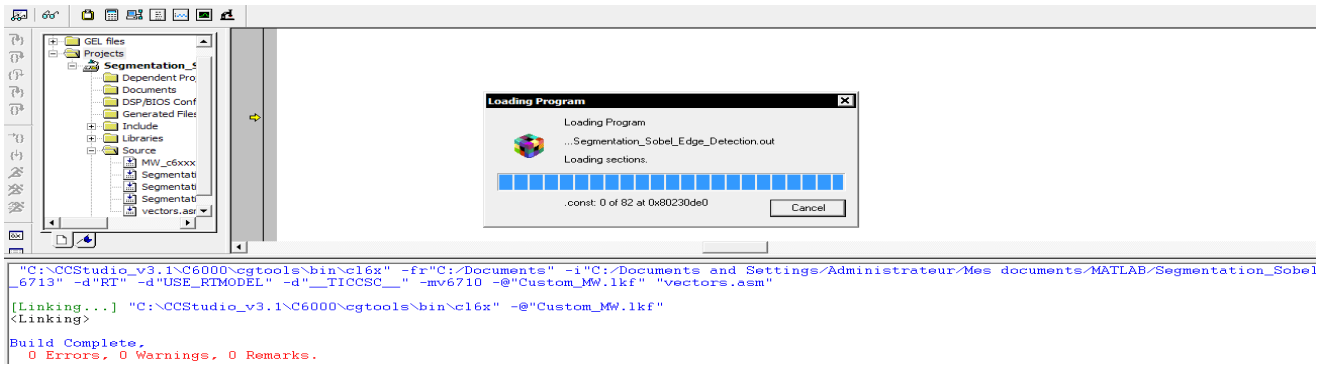


Figure 49 : Capture d'écran de l'opération de chargement du programme sur la carte

Pour exécuter ce programme, cliquez dans le menu « **Debug »** sur « **Go Main »** et ensuite dans le même menu « **Debug »** cliquez sur **« Run »**.

IV-3 Résultats obtenu à l'aide du Code Composer Studio CCS

Pour visualiser les résultats obtenus sous Code Composer Studio, on doit aller à "View➔ Graph➔ image" puis on remplit les paramètres de la fenêtre pop-up comme indiqué dans la figure 5.10.

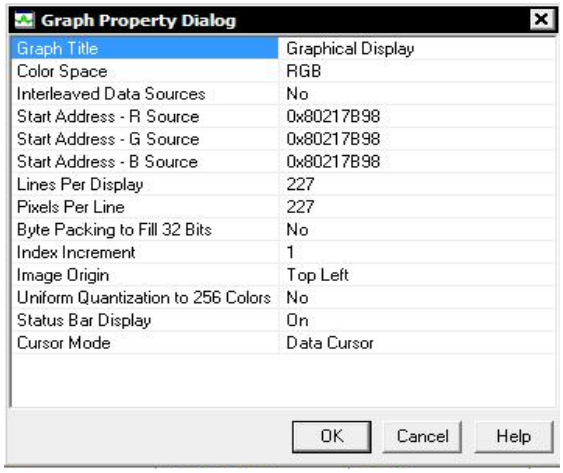
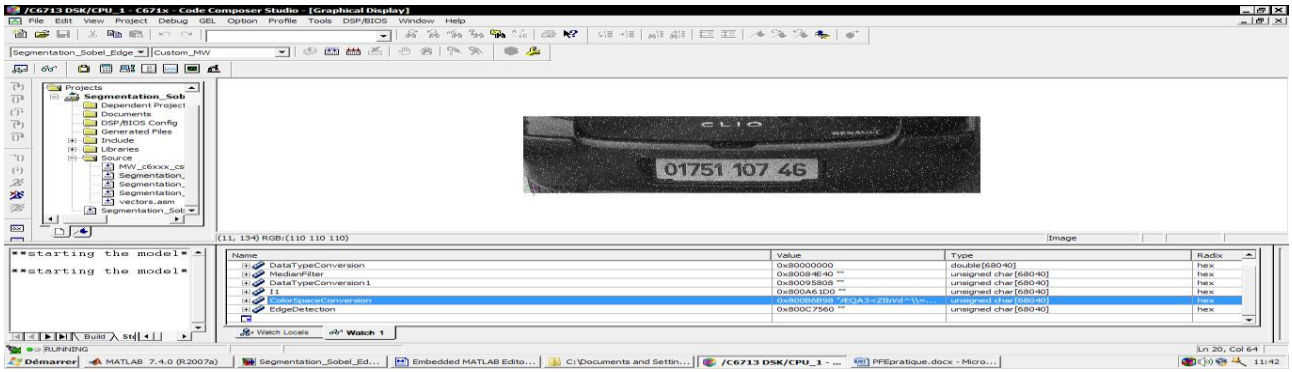
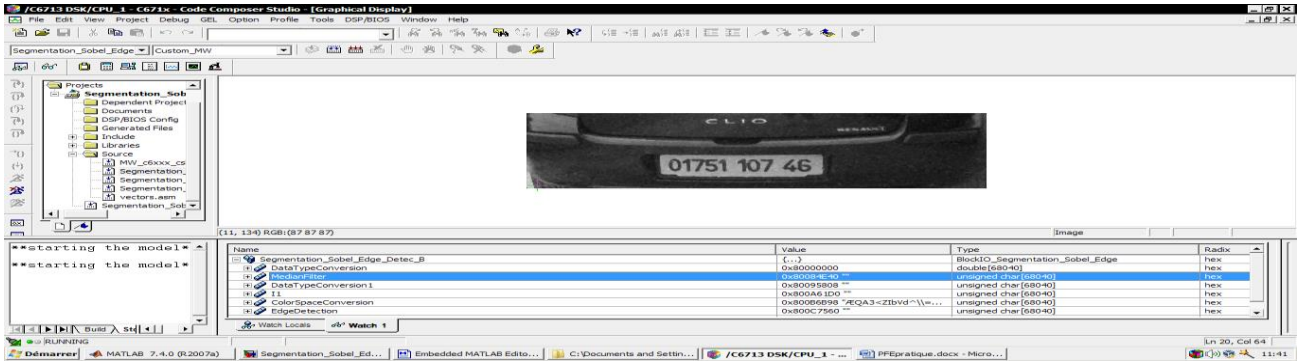


Figure 4.10 : paramètres de visualisation d'image

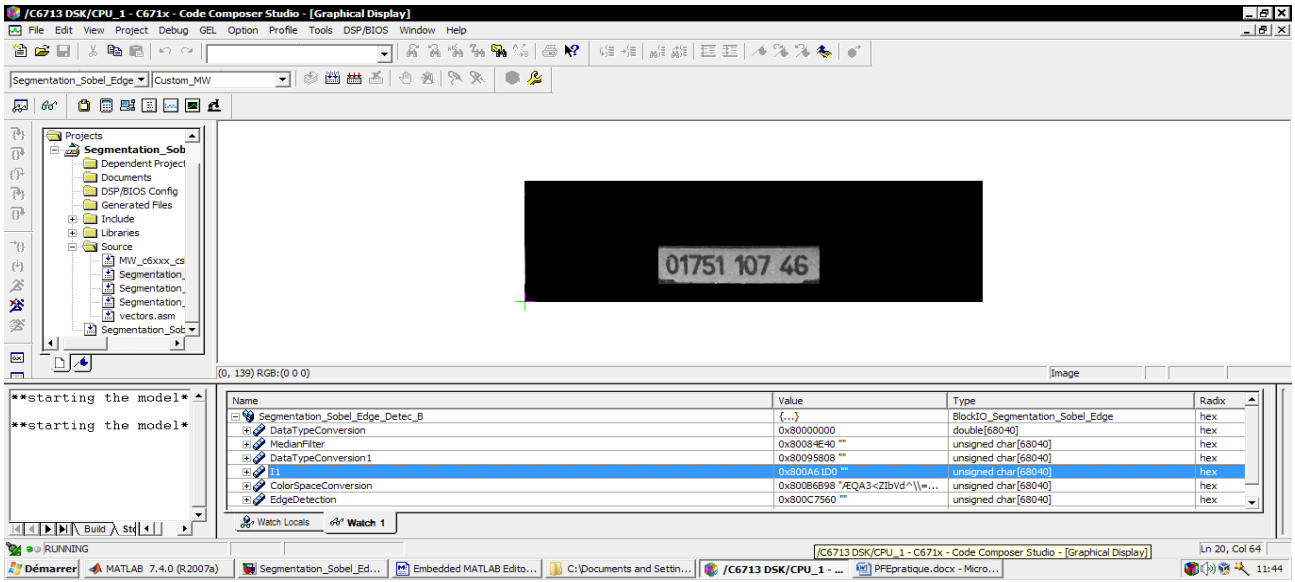
3-.1 Image bruitée et converti au niveau de gris $



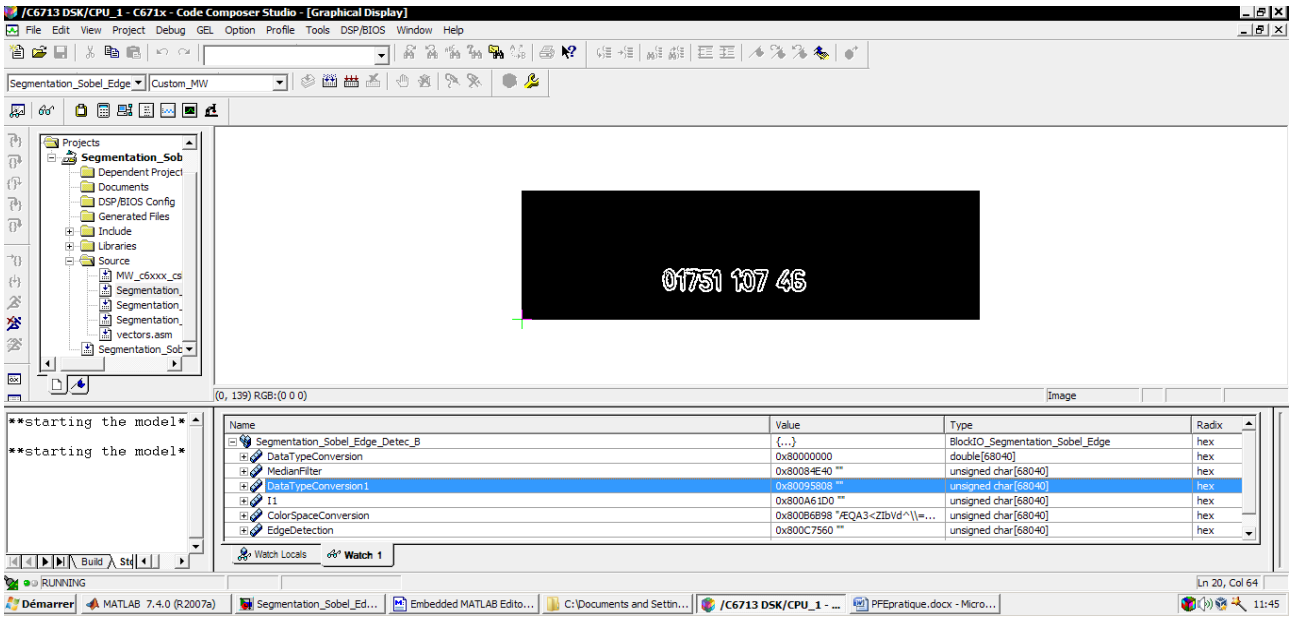
3-2 Image filtrée à l'aide de filtre Médian 3x3(image sans bruit)



3-3 Segmentation de l'image et extraction du matricule



3-4 Détection de contours au niveau de l'image segmentée



IV-4 Instructions pour la déconnexion

- Fermer le code composer studio, et tout programme qui se connecte à la carte DSK.
- Débrancher le câble USB.
- Débrancher l'alimentation de la carte DSK d'abord l'alimentation continue de la carte, puis l'alimentation au secteur.

Conclusion

Dans ce chapitre nous avons mis au point un algorithme de segmentation très performant basé sur l'analyse de l'histogramme horizontal et vertical pour détecter et faire une extraction du matricule puis on a implémenté ce algorithme sur une Carte à base de DSP (TMS320C6713 DSK), nous avons appris la grande différence entre un microprocesseur (Intel core™ i3) et un DSP. Nous avons compris qu'un microprocesseur est dédié pour un traitement général alors qu'un DSP est dédié spécifiquement au traitement numérique des signaux 2D. Nous avons conclu que le TMS320c6713 est un meilleur choix pour le traitement Numérique d'image. Ceci est traduit par le nombre de cycles d'horloge nécessaire pour effectuer ce traitement.

Conclusion Générale

La manipulation des images pose cependant des problèmes beaucoup plus complexes que celle du texte. En effet, l'image est un objet à deux dimensions, censé représenter un espace à trois dimensions, ce qui a deux conséquences majeures puisque le volume des données à traiter est beaucoup plus important et la structure de ces données est nettement plus complexe, Il en résulte que la manipulation, le stockage et la représentation de ces données se heurtent à certaines limitations.
Pour cela il existe des nombreuses applications pour lesquelles le DSP devient un choix idéal car ils fournissent la meilleure combinaison possible de performances, la puissance et le coût. Les questions de pouvoir prennent de l'importance que les processeurs DSP sont incorporés dans les appareils portatifs, mobiles et portables. Cela conduit au développement d'une classe importante de DSP, Avec les nouvelles technologies de fabrication de circuits intégrés disponibles nous pouvons nous attendre à voir plus sur puce périphériques et la mémoire.

ous avons pu constater que les DSP ne représentent qu'une partie du traitement numérique du signal, mais il reste essentiel par son rôle important dans la chaîne de traitement. Nous avons choisi ce DSP car il est adapté à des applications de traitement d'images. Puisque notre application concerne la Segmentation et détection de contours dans une image, nous avons trouvés que le choix du DSP TMS320C6713 est très performant, au niveau de calcule et mémoire, est un choix judicieux.

Nous souhaitons que ce travail serve de base pour les prochains projets et qu'ils auront la chance d'étudier le TMS 320C6713, et nous invitons les prochains étudiants à essayer de poursuivre dans ce vaste sujet.

Abréviations et acronymes

2D	*Bidimensionnel*
A/N	*Analogique/Numerique*
ALU	*Arithmetic and Logic Unit*
bpp	*bit par pixel*
C6416	*DSP TMS320C6416*
C64x	*Famille de DSP TMS320C64x, inclua*
c-à-d	*c'est-à-dire*
CAN	*Convertisseur Analogique-Numérique*
CCS	*Code composer studio*
CMJN	*Cyan, Magenta, Jaune et Noir*
CNA	*Convertisseur Numérique-Analogique*
CPU	*Central Processor Unit*
DCT	*Discrete Cosine Transform*
DSL	*Digital Subscriber Line*
DSP	*Digital Signal Processor*
E/S	*Entrée sortie*
EMIF	*External Memory Interface*
FFT	*Fast Fourier Transform*
GPIO	*General-Purpose Input/Output*
GPS	*Global Positioning System*
GSM	*Global System for Mobile*
HPI	*Host Port Interface*
I/O	*Input/Output*
IDE	*Integral developpement environement*
IRM	*Imagerie par résonance magnétique*
LAN	*Local Area Network*
MAC	*Multiplier and Accumulator*
McASP	*Multichannel Audio Serial Port*
McBSP	*Multichannel Buffered Serial Port*
MFLOPS	*Millions Floating Point Operations P*
MIPS	*Millions d'Instructions Par Seconde*
MOPS	*Millions Operations Per Second*
N/A	*Numerique/Analogique*
NG	*Niveau de gris*
Pixel	*Picture-element*
RAM	*Random Access Memory*
RGB	*Red Green Bleu*
ROM	*Read Only Memory*
SIMD	*single-instruction multiple-data*
TI	*Texas Instruments*
UT	*Unité de Traitement*
VLIW	*very-long-instruction-word*

Bibliographie

I) Ouvrages :

[1] Alexandre Renaux ‘’Traitement Numérique du Signaux ‘’
IFIPS / Université Paris Sud Orsay .9, avenue de la Division Leclerc PB 140 94234 CACHAN Cedex

[2] S. Bres, J. Jolion, F. Lebourgeois. Traitement et analyse des images numérique. Paris, Lavoisier, (Oct 2003, 411 pages).

[3] Texas Instruments Inc .. (2000). TMS320C6000 CPU and Instruction Set Reference Guide,
http://focus. tLcom/lit/ug/spru 189f/spru 189f.pdf, (16 décembre 2003).

[4] Z.W. Mekonnen, *“ Digital Signal Processing Applications using DSK C6713 ”*,
Communications Laboratory, University of Kassel, 2009.

[5] Shehrzad Qureshi, ‘’ Embedded Image Processing on the TMS320C6000 DSP’’, Exemple in
Code Composer Studio (20 Juilly 2006, 452 pages).

[6] Texas Instruments, *TMS320C6713, Floating-Point Digital Signal Processors, Data Sheet,*Dallas, TX, June 2006.

[7] Digital Signal Processing and Applications with C6713 and C6416 DSK, Rulph Shassaing

[8] Steven W. Smith, The Scientist and Engineer’s Guide to Digital Signal Processing, Second
Edition, California Technical Publishing, 1999.

[9] TMS320C6000 Programmer’s Guide, SPRU198G, Texas Instruments, Dallas, TX, 2002.

[10] Embedded IDE Link™ for use with Texas instruments, code composer studio™ user’s Guide

[11] Implémentation d’un corrélateur d’image sur TMS320C6713 DSK, PFE2007,UDL, Algérie

II) Documents PDF :

[12] Master 2 : K. Beloullata, ‘’Traitement d’image’’, Note de cours, UDL-SBA, 2013-2014, Algérie.

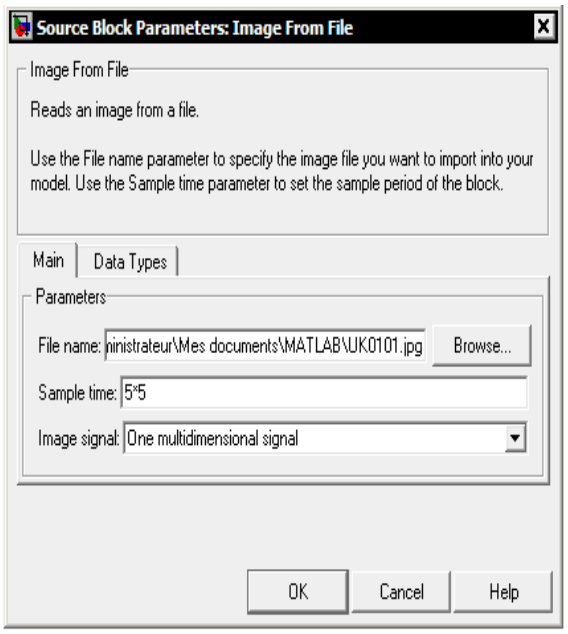
[13] Nicolas Nolhier,‘’ Initiation au DSP ’’, Note de cours, Université Paul Sabatier, Toulouse, 2001

[14] Caroline Petitjean, ‘’ Les Processeurs de traitement du signal ‘’, Université de Rouen, 2007.

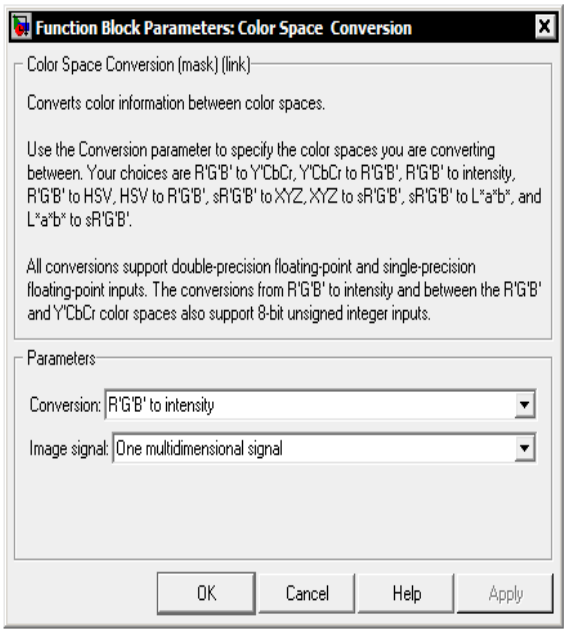
III) Webographie :

- http://www.ti.com/
- http://www.mathworks.com/
- http://www.spectrumdigital.com/
- http://uuu.enseirb-matmeca.fr/~megret/Enseignement/DSP/TP/

ANNEXE

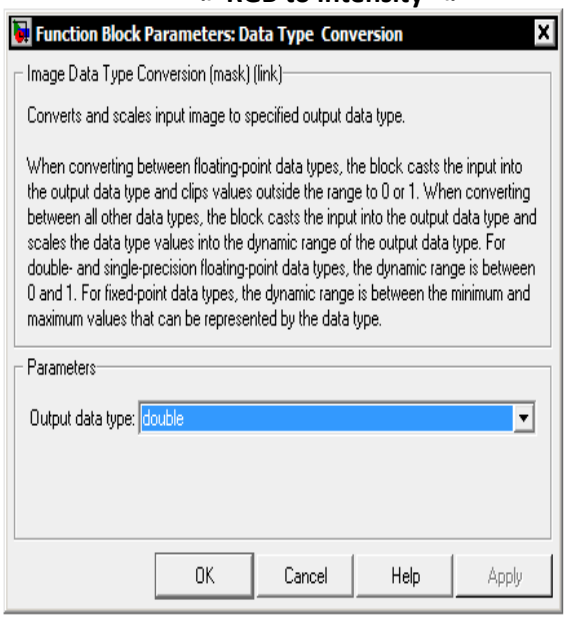


« Paramètres de configuration du block »
Image From File
« UK0101.jpg »

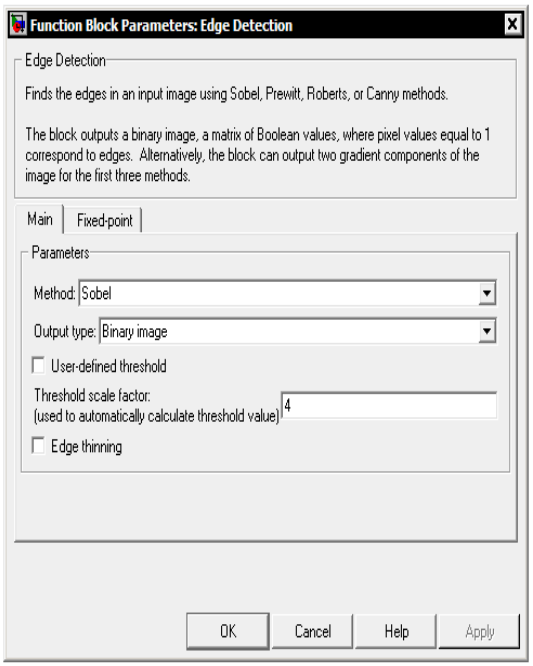


« Paramètres de configuration du block »
Color Space Conversion
« RGB to intensity »

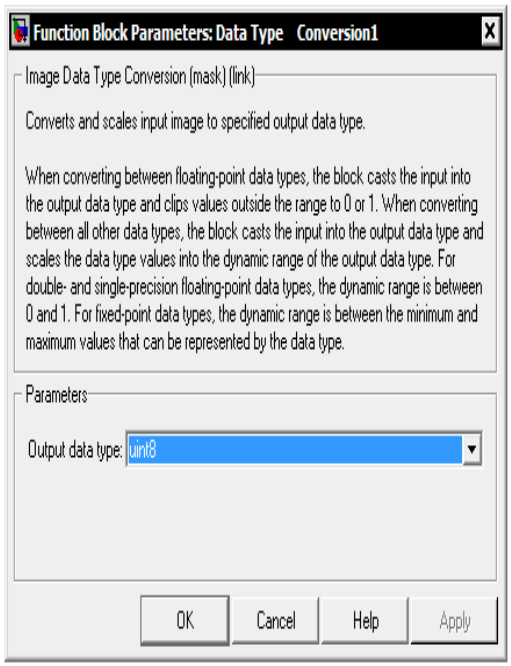
« Paramètres de configuration du block »
Median Filter
« [3 3] »



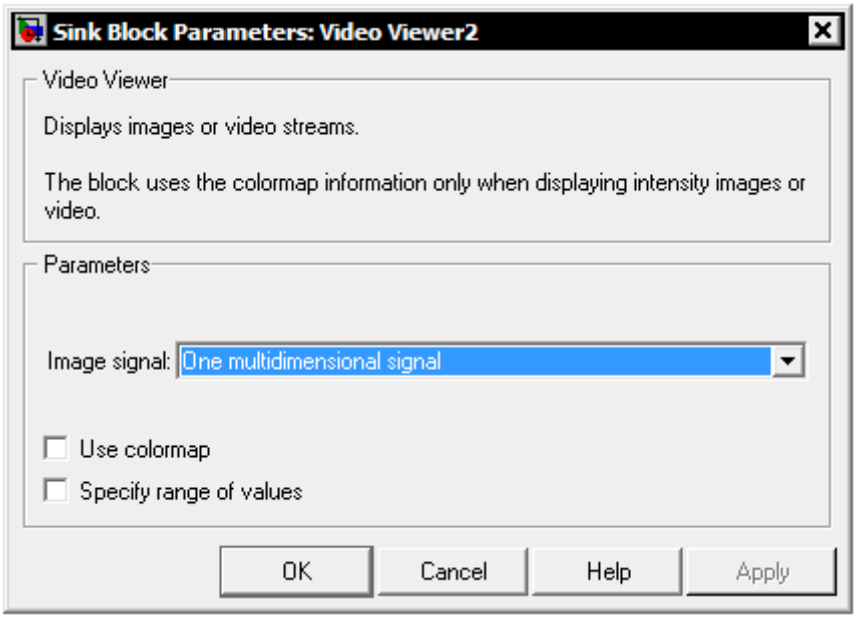
« Paramètres de configuration du block »
Image Data Type Conversion
« Double »



« Paramètres de configuration du block »
Edge Detection
« Sobel »



« Paramètres de configuration du block »
Image Data Type Conversion1
« Uint8 »



« Paramètres de configuration du block Video Viewer »

« One multidimensional signal »