# TP 4 : PRISE EN MAIN DE L'ENVORONNEMENT CODE COMPOSER STUDIO "CCS"
# CONFIGURATION DE L'ENVIRONNEMENT CCS POUR L'EXPLOITATION DES DSP

**Objectif** : L'objectif essentiel de ce premier TP est de se familiariser avec les outils Code Composer Studio (CCS 3.3) de Texas Instruments. Nous allons apprendre à utiliser l'environnement CCS pour le développement d'un système DSP et l'analyse d'un programme et des résultats produits par simulation.

## I- Rappel : DSP et outils de développement

Les DSP (Digital Signal Processor) sont des processeurs spécialisés de traitements numériques des signaux (un type particulier des microprocesseurs). En effet, les domaines d'applications du traitement numérique du signal sont nombreux et variés : traitement du son, de l'image, synthèse et reconnaissance vocale, analyse, compression de données, télécommunications (modems, GPS, etc.),…etc. Chacun de ces domaines nécessite un système de traitement numérique, dont le cœur est un DSP ayant une puissance de traitement adaptée (pour un coût économique).

La famille des DSP la plus répondue actuellement est celle de Texas Instruments (environ 70% du marché, 30% partagés entre Motorola et les autres). En bref, les constructeurs commercialisent généralement des DSP réservés à des secteurs d'activités ciblés. A titre d'exemple, la série TMS320C54x de Texas Instruments est développée spécialement pour l'exécution rapide des algorithmes exploités dans les téléphonies sans fils. Dans une optique pédagogique, nous allons nous intéresser dans les TP au TMS320C6713 de Texas Instruments.

Les outils de développement proposés par Texas Instruments sont libres d'utilisation du moment que nous n'utilisons que le mode simulation. Chaque exercice de TP peut être pré-compilé et testé à la maison avant l'arrivé en séance.

En effet, l'installation des outils CCS puis l'utilisation en mode simulation, pour l'analyse de traduction de programme C vers ASM (TMS320C6713), est sans aucun doute l'une des solutions les plus efficaces pour bien comprendre.

## II- Installation du Code Composer Studio

Code Composer Studio (CCS) fournit plusieurs outils pour faciliter la construction et la mise au point des programmes de DSP. Il comprend un éditeur de code source, un compilateur de langage C/C++, un assembleur de code relocalisable, un éditeur de liens, et un environnement d'exécution qui permet de télécharger un programme exécutable sur une carte cible, de l'exécuter et de le déboguer au besoin. CCS comprend aussi des outils qui permettent l'analyse en temps réel d'un programme en cours d'exécution et des résultats produits. Enfin, CCS fournit un environnement de gestion de fichiers qui facilite la construction et la mise au point des programmes.

Pour commencer à utiliser CCS en simulation, il faut l'installer correctement sur votre PC (allez télécharger directement de Texas Instruments ou bien autrement : http://www.jeromedemers.com/blog/tag/code-composer-studio/ ).


Après l'installation de CCS version 3.3 (CCS v3.3), nous allons avoir les deux icones suivantes sur le bureau :


icône de l'IDE


icône Setup CCS

**Remarques** : Si nécessaire, avant d'installer le CCS 3.3, il faut installer tout d'abord le DotNet 1.1 de Microsoft qui se trouve dans le même répertoire téléchargé de Texas Instruments. Sous Windows 7 et plus, pour ouvrir les applications de CCStudio v3.3 directement, il faut faire un click droite de la souris sur l'icône d'application puis aller vers le menu "Exécuter en tant que administrateur". Sinon, faire la configuration définitive par le menu "Propriétés" >> "Compatibilité" >> "Exécuter en tant que administrateur" >> "Appliquer" >> "Ok".

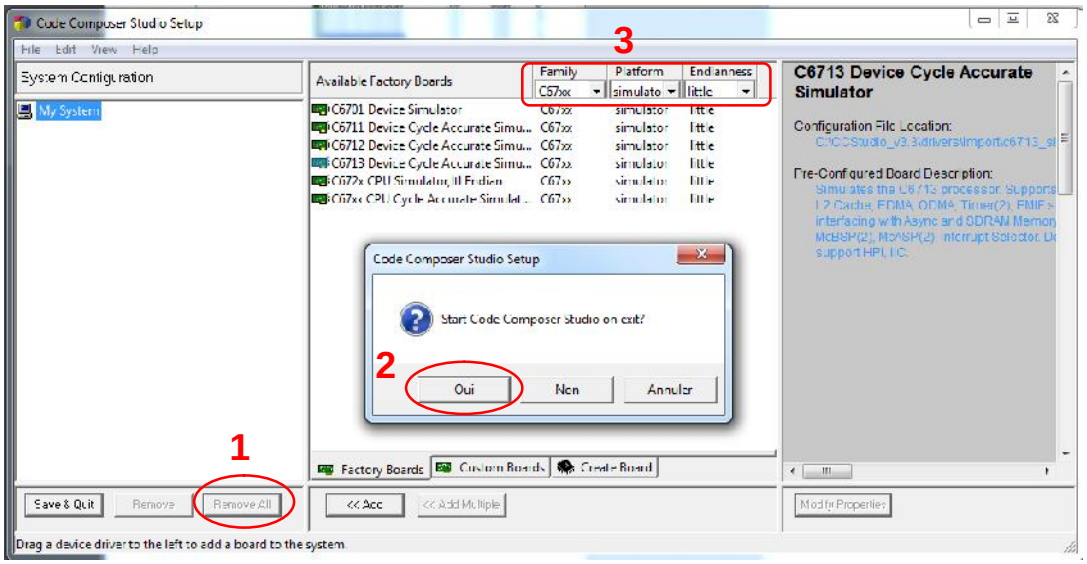
## III- Exemple d'utilisation : *Génération d'une onde sinusoïdale*

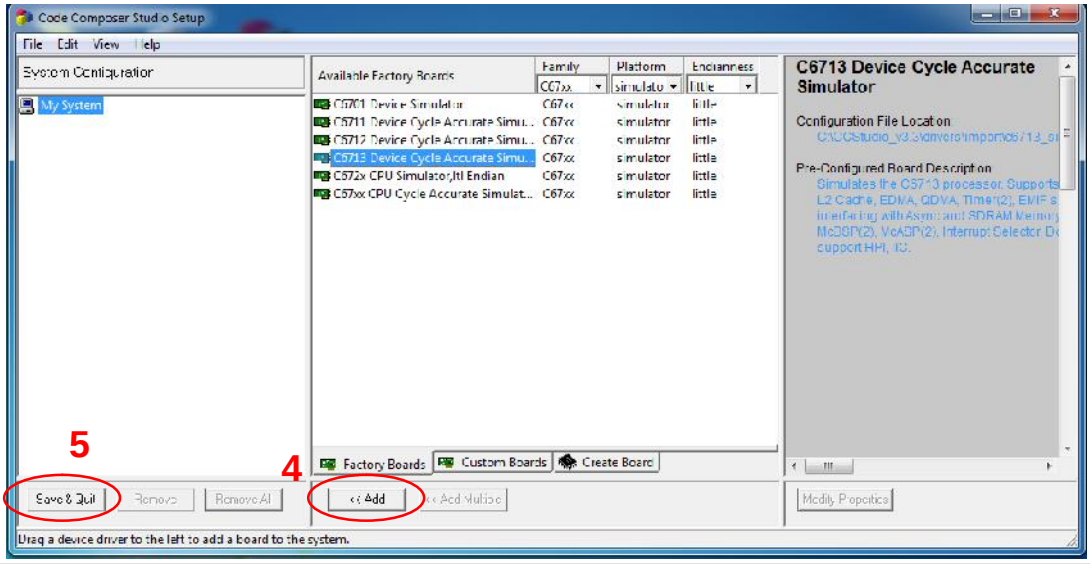
Le premier exemple qui suit montre comment un algorithme multifilaire simple peut être compilé, assemblé et lié en utilisant CCS. En premier lieu, plusieurs valeurs de données sont écrites en mémoire. Ensuite, un pointeur est assigné au début des données de sorte qu'elles puissent être traitées comme elles sont présentées.

### 1. Créer un répertoire de travail et un projet

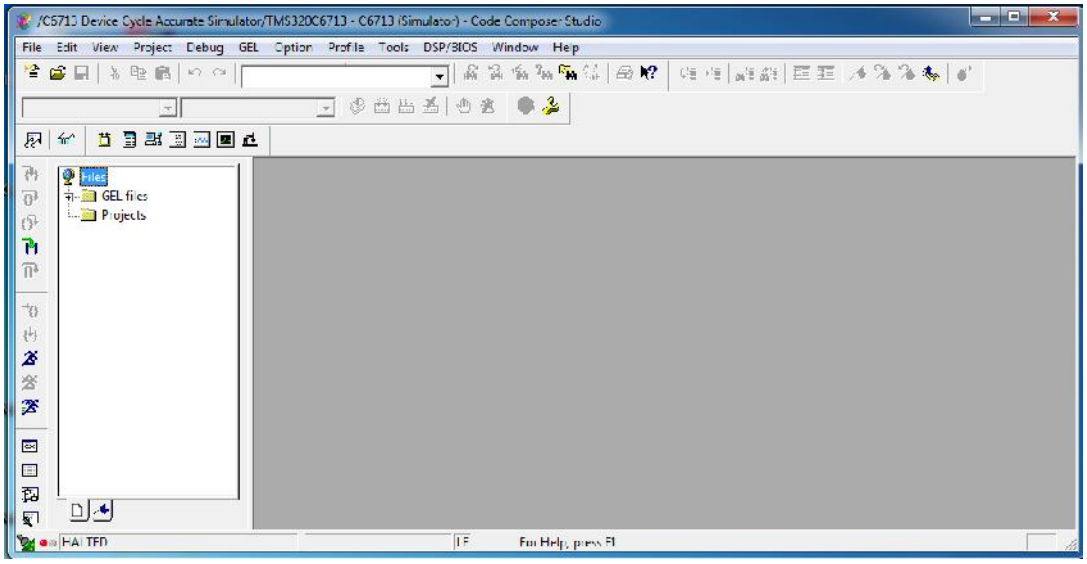
Le répertoire de travail est l'endroit où les projets seront sauvegardés. Lors de la première utilisation de CCS v3.3, il est nécessaire de créer un répertoire de travail (exemple un répertoire dans MyProjets) et configurer CCS pour un Trajet.

- Ouvrir **Setup CCStudio v3.3** (icône sur le bureau) pour débuter la configuration d'un TRAGET (DSP Device).
- Cliquer sur **Remove All** avant de commencer puis configurer : C6713 Device, Cycle Accurate, Simulator et Little.

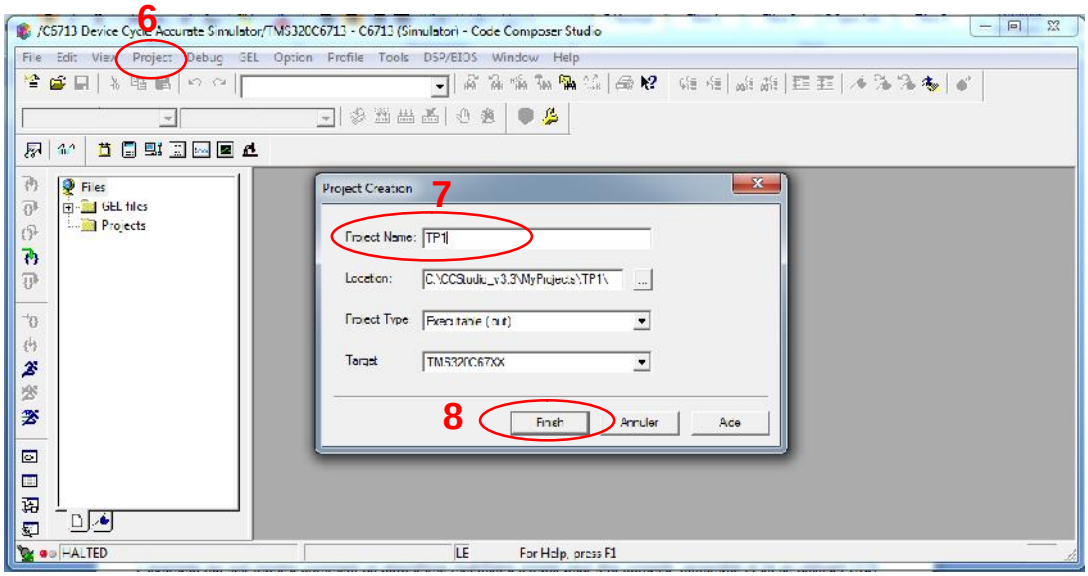


- Cliquer maintenant le bouton **<<Add** ensuite cliquer **Save & Quit** puis sur **OUI**. On obtient la page suivante :



- Cliquer sur **Project → New…** et donner le nom du répertoire de travail (à titre d'exemple **TP1**) puis cliquer **Finish**.



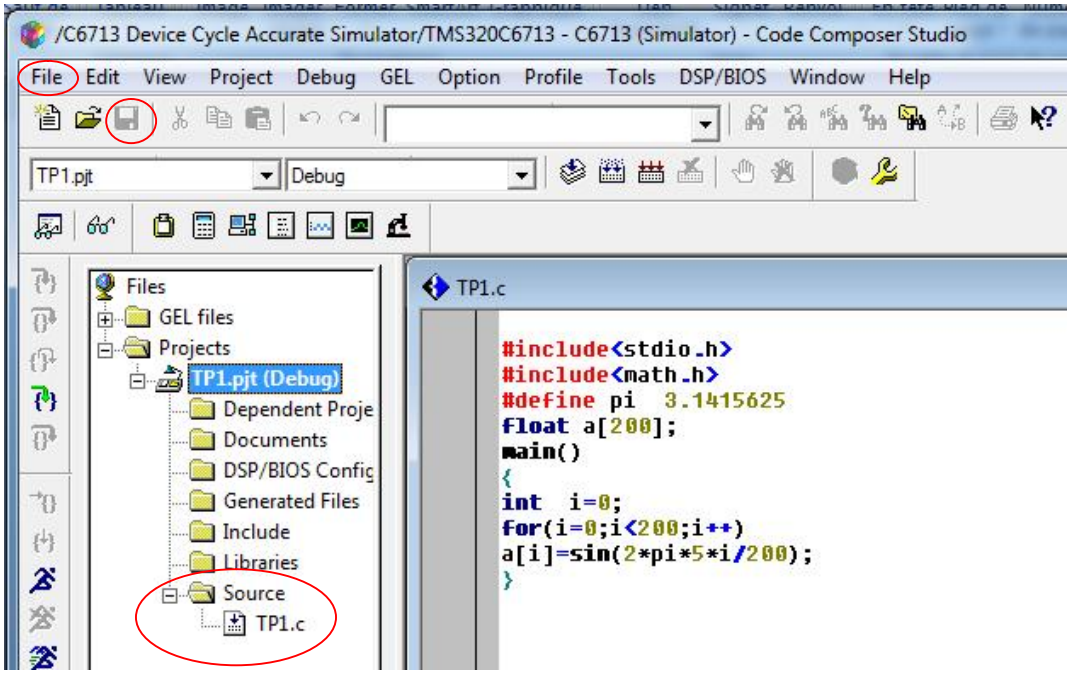

En résumé, le processus de développement de code de CCS commence toujours par la création d'un projet qui facilite l'intégration des fichiers requis pour produire un fichier exécutable. Le répertoire projet contient des références aux fichiers source (***.c, *.asm**), aux fichiers d'en-tête (***.h**), au fichier de commande (***.cmd**) pour l'éditeur de liens, et aux fichiers de bibliothèques (***.lib**). Le projet permet encore de spécifier les paramètres du compilateur, de l'assembleur, et de l'éditeur de liens ; afin de produire le fichier exécutable.

**2. Créer un fichier source**

CCS fournit un éditeur intégré qui permet la création des fichiers sources. Une fenêtre d'éditeur apparait en l'élément de menu **File → New Source File**. Le fichier (nom du fichier) doit être sauvegardé avec une extension ***.c** ou bien ***.asm** (à titre d'exemple, pour notre 1$^{er}$ TP : **TP1.c**), ensuite faire la saisie du code (programme) dans cette fenêtre d'éditeur. En Sauvegarder le fichier source créé dans le répertoire de travail (**File → Save → CCStudio_V3.3 → MyProjects → TP1**).

Dans cette première approche de l'environnement CCS v3.3, nous nous intéressons à la génération (par simulation) d'une onde sinusoïdale **A(t) = Sin(*wt*)** en utilisant TMS320C6713 ; en exploitant la méthode de la table de consultation. Pour un DSP, le signal sinusoïdal doit être digital et mathématiquement il s'écrit : $A(n) = Sin(wn) = Sin(2\pi 5n/200)$.

- Faire la saisie et la sauvegarde du code en langage C.
- Ajouter le fichier au projet : **Project → Add Files to Project… → TP1.c → Ouvrir**.

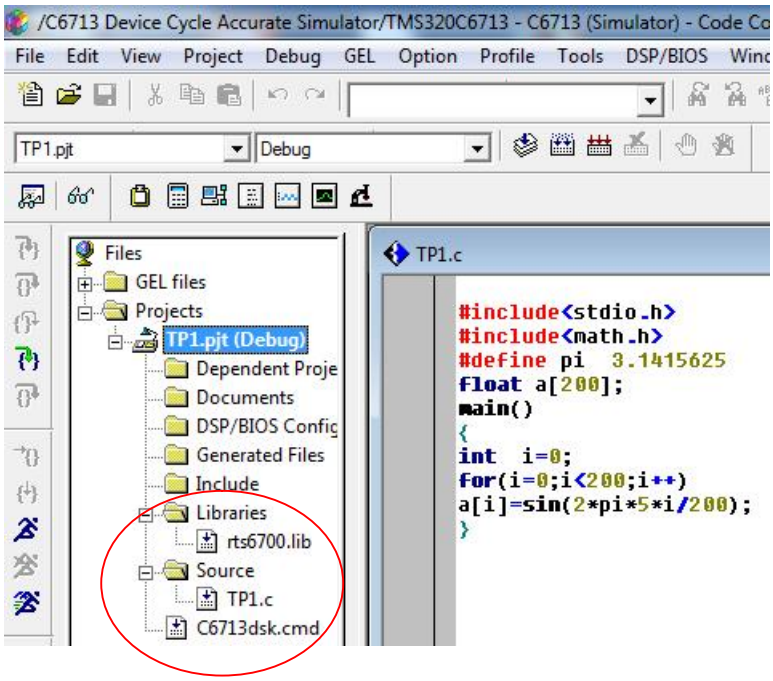



### 3. Ajouter les fichiers de support au projet

En plus du fichier source (possibilité d'avoir des fichiers sources), un fichier de commande d'éditeur de liens (Linker) et des fichiers supports pour le TMS320C6713 et la cible DSK6713 devront être inclus au projet sous forme de librairies et répertoires (ces étapes seront définies plus tard). Par conséquent, ces fichiers seront implantés au projet dans les prochains Travaux Pratiques.

Dans cette application, nous allons ajouter un seul fichier de librairie (**rts6700.lib**). Pour ajouter ce fichier, il faut :

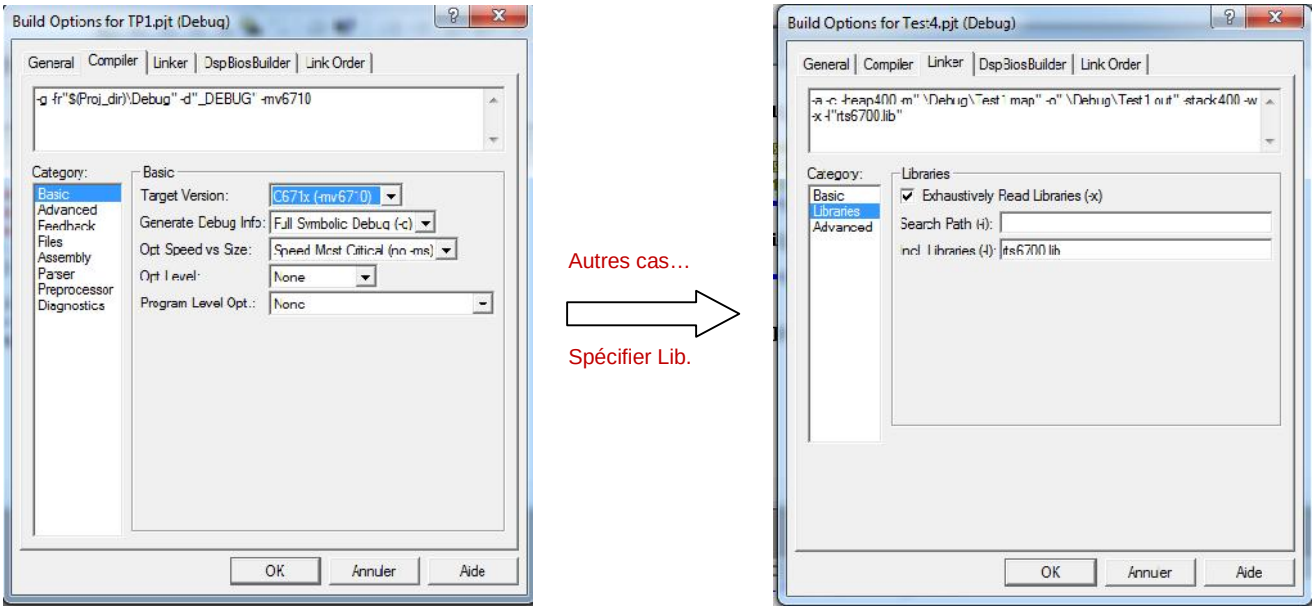
- Copier le fichier rts6700.lib dans le répertoire de travail ( **C:\CCStudio_v3.3\C6000\cgtools\lib\rts6700.lib**)
- Ajouter le fichier rst6700.lib au projet : **Project → Add Files to Project… → rts6700.lib**
- Copier puis ajouter le fichier C6713dsk.cmd au projet : **Project → Add Files to Project… → C6713dsk.cmd**



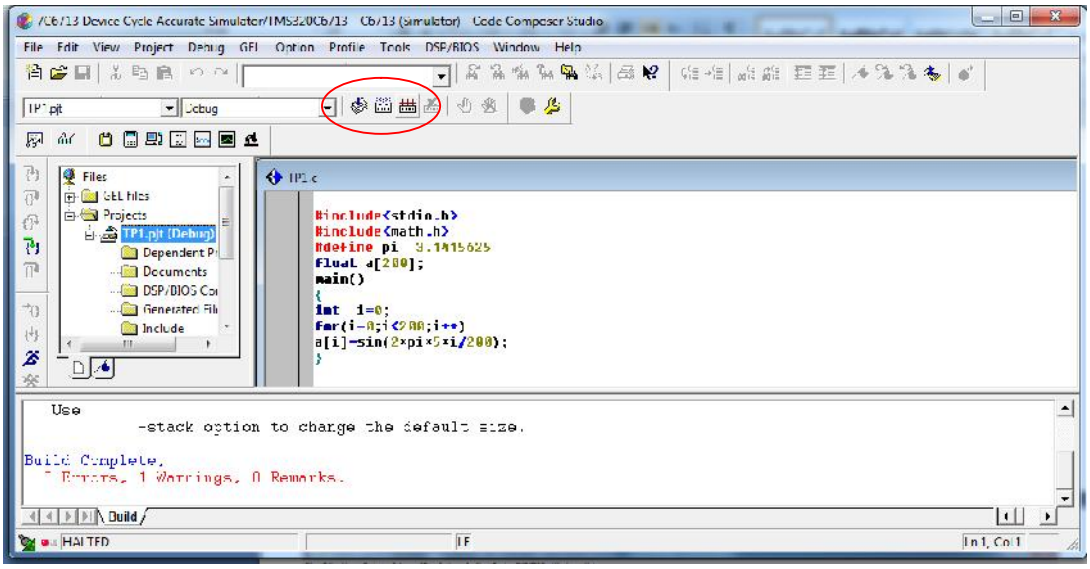

Rappel : Les fichiers ajoutés au projet sont nécessaires pour produire le fichier exécutable. DSK : DSP Starter Kit.

### 4. Mise au point du projet

Après avoir ajouté le fichier source, fichier de commande et fichier de bibliothèque au projet, vous pouvez construire le projet et créer un fichier exécutable pour le DSP-cible. Pour cela, choisissez l'élément de menu **Projet → Build Options** et faire l'adaptation du compilateur via : **Project → Build Options… → Basic → Target Version → C671x → OK**




- Compiler, assembler, et corriger les erreurs si nécessaire : Compile File >> Inceremental Build >> Rebuild all




- Si le processus de construction est complété sans erreurs, le fichier exécutable (***.out**) est produit (TP1.out).

**5. Suivi de l'exécution du projet (mode Debug)**

Une fois que le processus de construction est complété sans aucune erreur, le programme peut être chargé et exécuté sur Simulateur ou bien sur une carte DSP-cible (Simulateur dans notre cas).

**a)** Pour charger le programme dans le simulateur et simuler (débogage d'un code), faire les étapes suivantes :

- Choisissez l'élément : **File → Load Program…** pour transférer le programme compilé (TP1.out) vers l'éditeur.
- Spécifier le chemin du fichier exécutable : **Ouvrir → Debug → TP1.out**

**b)** Pour exécuter le programme, faire les étapes suivantes :

- Dans certains cas, avant l'exécution, vous devriez **faire apparaître un point d'arrêt** (Debug : Toggle Breakpoint) via l'icône disponible sur la ligne de menu rapide en haut (la main en haut de l'écran).
- Choisissez maintenant l'élément : **Debug → Run** pour un mode continu. Cette commande est aussi disponible sous forme d'icône sur la colonne de menu rapide.

**c)** Pour l'affichage des différentes fenêtres de débogage (paramètres et résultats), allez vers le menu « **View** » :

- **View → Memory**
- **View → Graph → Time/Frequency…** ; ensuite configurer le graphe Windows via **Graph Property Dialog**.
- **View → Registers → Core Registers**,…etc.

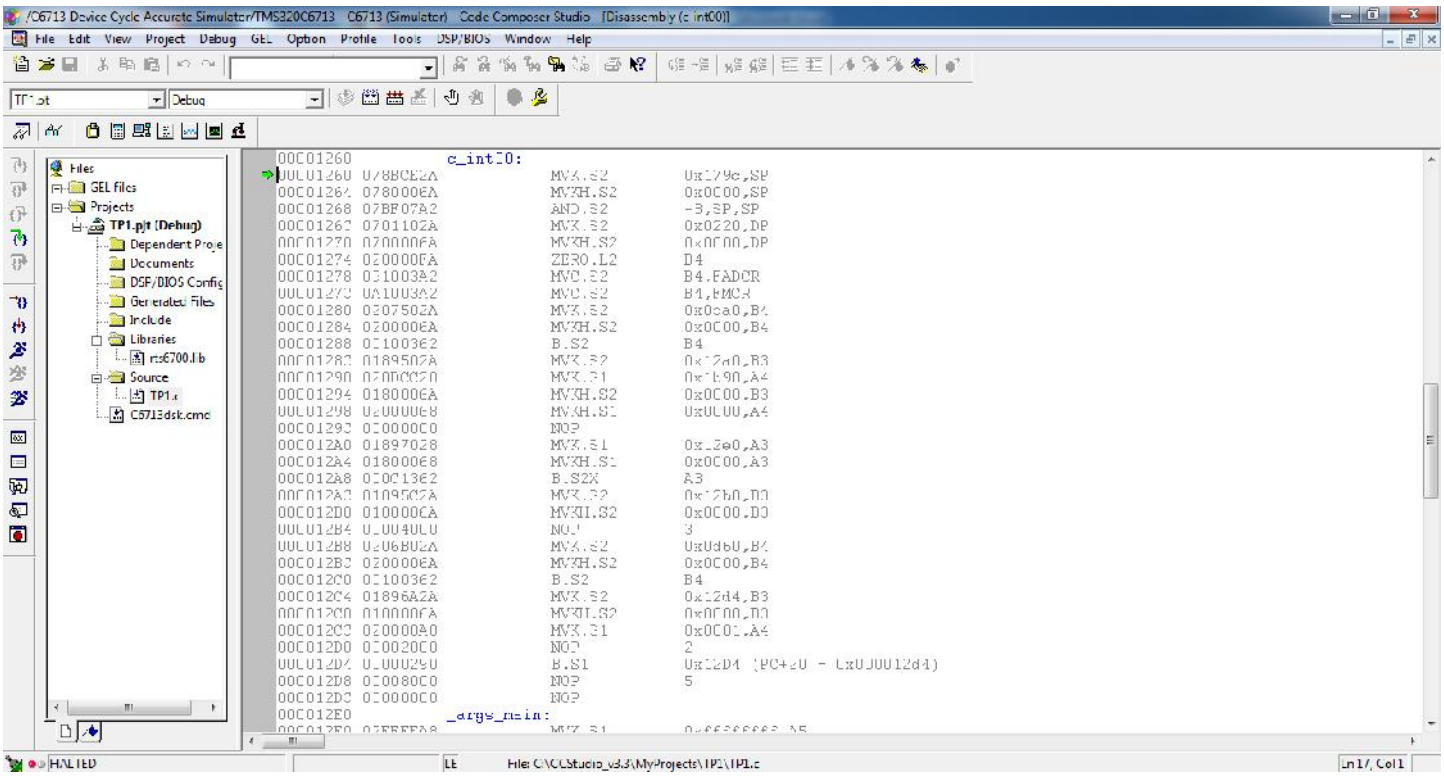


Figure (a). Conséquence de chargement du programme dans le simulateur (Load Program…).

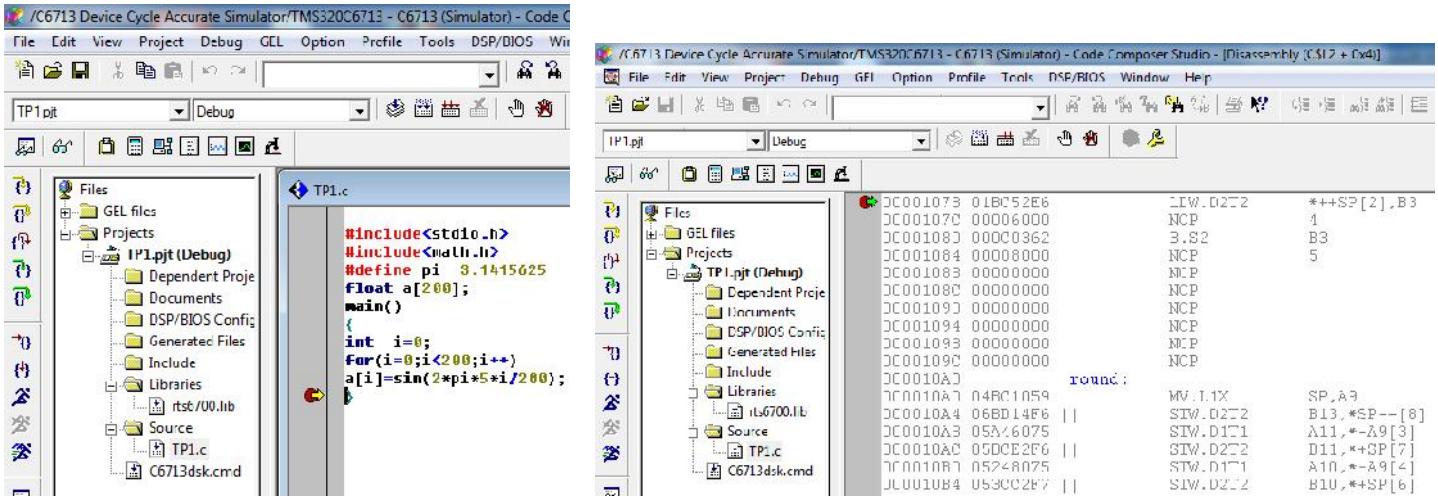


Figure (b). Conséquences de l'exécution du programme chargé (Run) ; aprsè détermination du point d'arrêt.

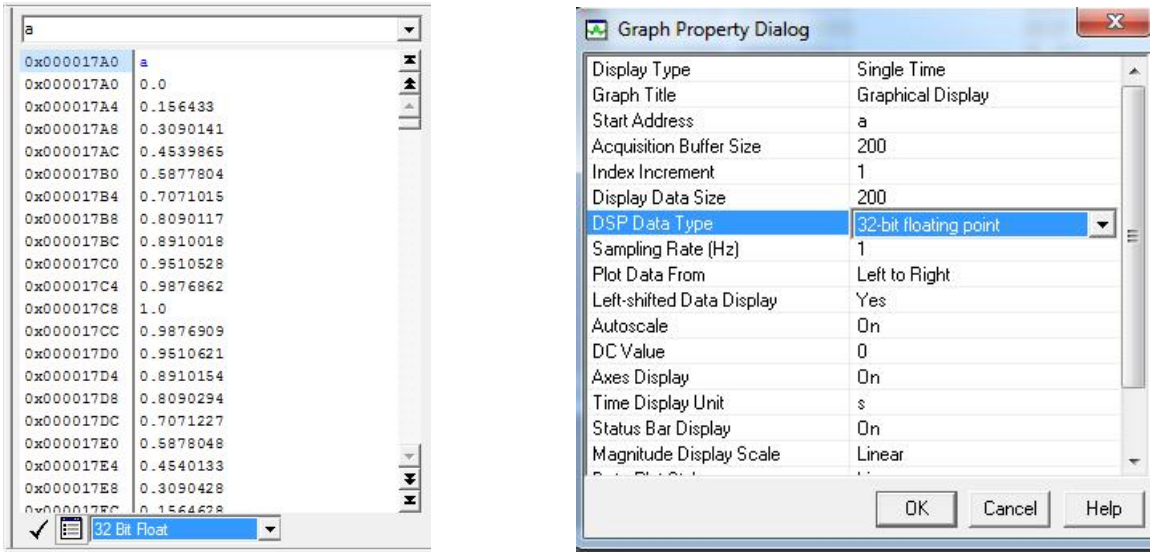
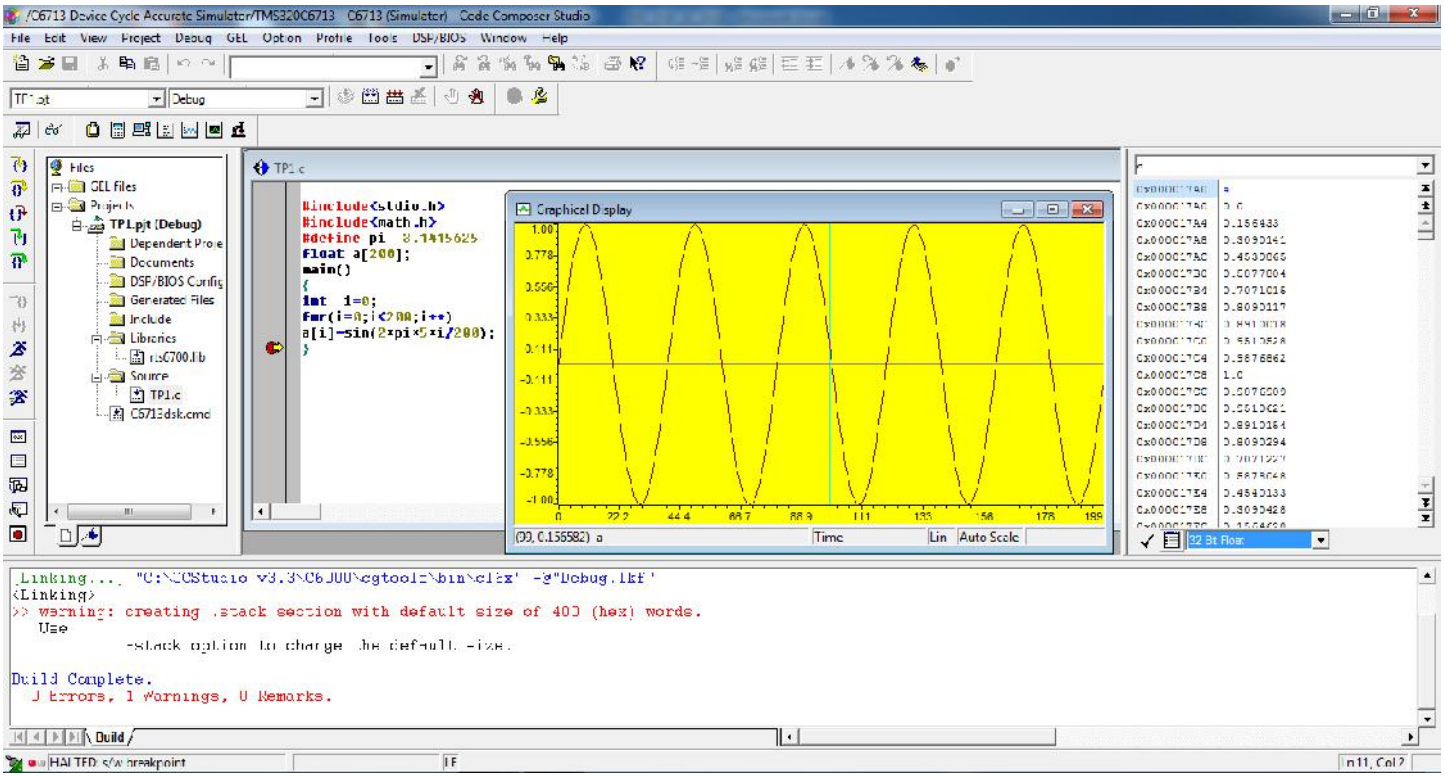
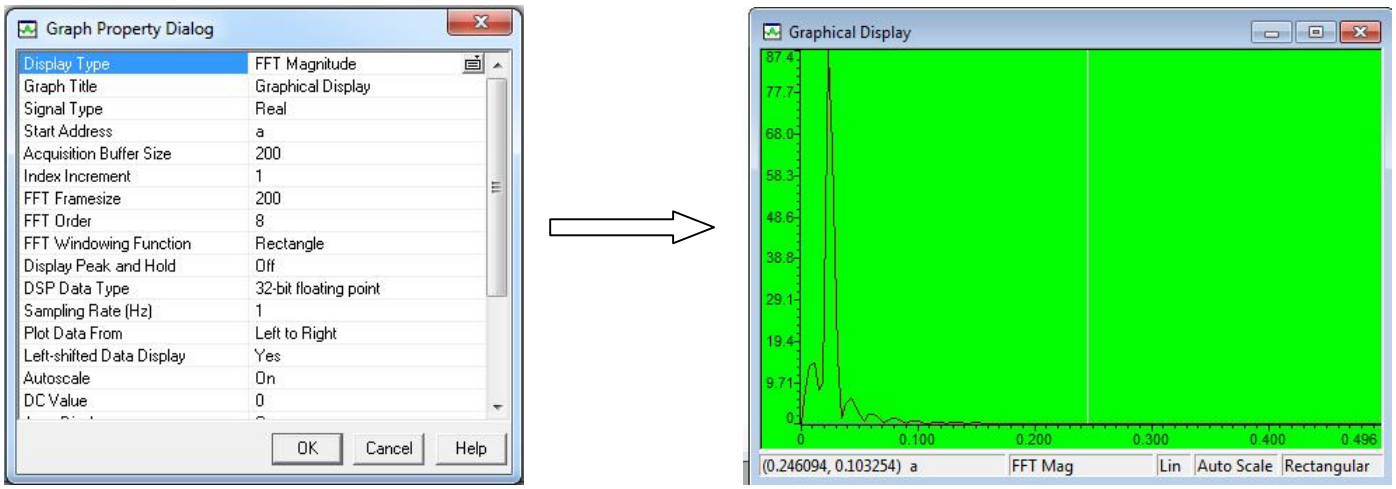
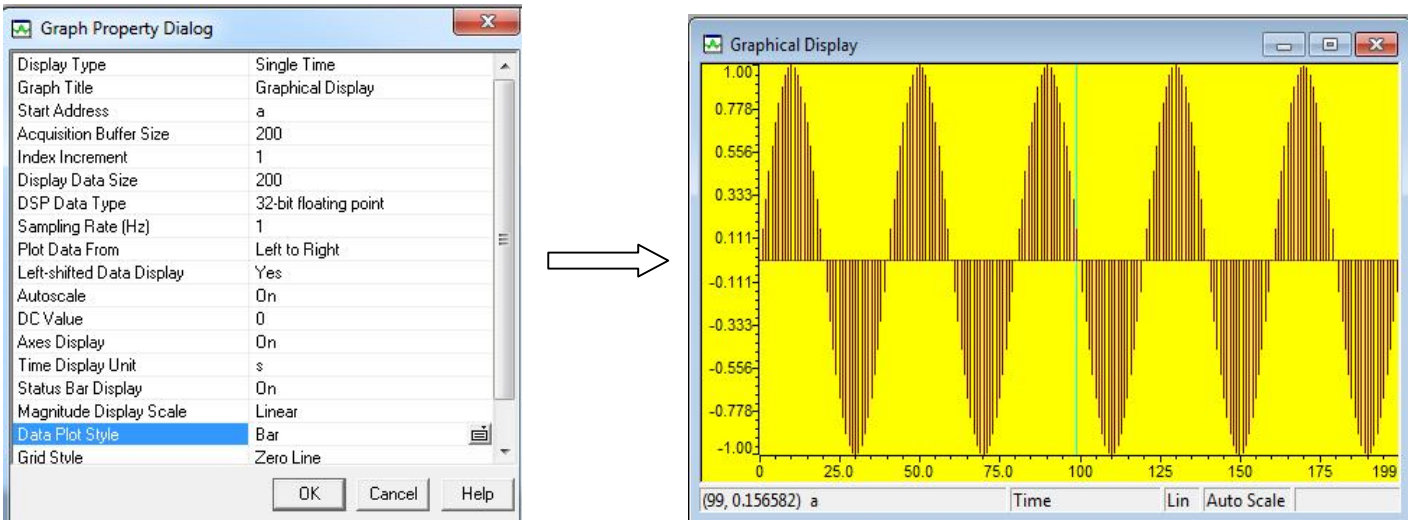



Figure (c). Différentes fenêtres de débogage pour l'ajustement et l'affichage des résultats (View).

Les figures suivantes montrent quelques graphs des résultats de simulation visualisés.




**Réalisé par : Dr. S. ABADLI.**

**Annexe : CCS, librairies et linker**

**Code Composer Studio (CCS)**

Code Composer Studio (CCS) fournit un environnement de développement intégré (IDE) pour les applications de traitement de signaux numériques, en temps réel, basées sur le langage de programmation C. Il insère un compilateur C, un assembleur et un éditeur de liens. Il possède des fonctionnalités graphiques et prend en charge le débogage en temps réel. Le compilateur C compile un programme source C avec l'extension « **.c** » pour produire un fichier source d'assemblage portant l'extension « **.asm** ». L'assembleur assemble un fichier source « .asm » pour générer un fichier objet en langage machine avec l'extension « **.obj** ». L'éditeur de liens associe en entrée des fichiers d'objet et des bibliothèques d'objets pour générer un fichier exécutable portant l'extension « **.out** ». Ce fichier exécutable représente un format de fichier d'objet commun lié (COFF), répandu dans les systèmes Unix et adopté par plusieurs fabricants de processeurs de signal numérique. Ce fichier exécutable peut être chargé et exécuté directement sur le processeur de signal numérique.

Un projet CCS comprend tous les fichiers (ou liens vers tous les fichiers) requis pour générer un fichier exécutable. Diverses options permettant d'ajouter ou de supprimer des fichiers de types différents dans un projet sont fournies. En outre, un projet CCS contient des informations sur la manière dont les fichiers doivent être utilisés pour générer un fichier exécutable. Les options du compilateur / éditeur de liens peuvent être spécifiées. Un certain nombre de fonctionnalités de débogage sont disponibles, notamment la définition de points d'arrêt et l'observation de variables, l'affichage de la mémoire, les registres, les codes de code d'assemblage et de code C combinés, les résultats graphiques et le temps d'exécution de la surveillance.

On peut faire défiler un programme de différentes manières (entrée, sortie ou sortie). L'analyse en temps réel peut être effectuée à l'aide de la fonction d'échange de données en temps réel (RTDX) de CCS. Cela permet l'échange de données entre le PC hôte et le DSK cible, ainsi que l'analyse en temps réel sans arrêter la cible.

Vous pouvez créer un projet dans CCS pour gérer facilement une application impliquant plusieurs fichiers source, bibliothèques, cartes mémoire et fichiers de commandes spéciaux. Le fichier contenant toutes les informations du projet porte l'extension « **pjt** ». En cliquant sur les boutons "Rebuild All" ou "Incremental Build" ou par les sélections de menu.

**Librairies et éditeur de liens (Linker) pour implémentation sur carte C6713DSK (C6713 DSP Starter Kit)**

**Librairie de support de carte (BSL) : dsk6713bsl.lib**

Une librairie (bibliothèque) spéciale de support de carte ou BSL (Board Support Library) est fournie avec le TMS320C6713 DSK. Le BSL fournit des fonctions en langage C pour la configuration et le contrôle de tous les dispositifs intégrés. La bibliothèque comprend des modules pour l'initialisation générale de la carte, l'accès au codec AIC23, la lecture des commutateurs DIP, le contrôle des voyants, ainsi que la programmation et l'effacement de la mémoire Flash. Le code source de cette bibliothèque est également inclus. La version de Code Composer fournie avec DSK est configurée pour utiliser automatiquement le BSL.

**Librairie de support de puce (CSL) : csl6713.lib**

La librairie de support de puce ou CSL (Chip Support Library) contient des fonctions C et des macros pour la configuration et l'interfaçage avec tous les périphériques C6713 intégrés au processeur et le contrôleur d'interruption du CPU. Cette bibliothèque est chargée sur le PC lorsque le logiciel DSK est installé. Les fichiers d'en-tête CSL fournissent une description symbolique complète de tous les registres de périphériques et de champs de registres.

**Librairie de support au moment d'exécution (RTS) : rts6700.lib**

Un fichier de librairie de support au moment de l'exécution ou RTS (Run Time Support-library) est aussi nécessaire.

**L'éditeur de lien (Linker) : C6713dsk.cmd**

La dernière étape de la création d'un programme consiste à relier tous les modules déplaçables. L'éditeur de liens, lnk6x.exe, combine les modules d'objet déplaçables pour former un programme de sortie exécutable. L'extension par

défaut pour les programmes exécutables est « **out** ». En outre, l'éditeur de liens peut générer un fichier de carte indiquant les adresses de mémoire absolues de toutes les variables globales. Une entrée très importante pour l'éditeur de liens est un fichier de commandes d'éditeur de liens qui porte l'extension « **cmd** ». Le fichier de commandes peut contenir des noms de modules d'objet supplémentaires à lier, des chemins d'accès à des bibliothèques, des noms pour les fichiers mappés et sortants, une mappe mémoire pour le système matériel cible et des commandes décrivant où mettre des sections spécifiques du programme en mémoire.

En bref, le développement en C ou en ASM sous l'environnement CCS se base essentiellement sur les instructions et les directives d'assemblage, pour guider le Compilateur/Linker à créer un fichier binaire correct. Les directives les plus utilisées en C pour le développement d'un code basique sont : **.EXT_RAM, .vectors, .text, .bss, .cinit, .stack, .sysmem, .const, .switch, .far, .cio, .csldata.**

Le fichier de commande « **C6713dsk.cmd** », nécessaire pour l'implémentation sur carte C6713DSK, est un fichier d'organisation des sections (segments) mémoires. Le contenu de ce fichier est montré dans la figure suivante :

```
/*C6713dsk.cmd  Linker command file*/

MEMORY
{
  IVECS:    org=0h,             len=0x220
  IRAM:         org=0x00000220, len=0x0002FDE0 /*internal memory*/
  SDRAM:        org=0x80000000, len=0x01000000 /*external memory*/
  FLASH:        org=0x90000000, len=0x00020000 /*flash memory*/
}

SECTIONS
{
  .EXT_RAM :> SDRAM
  .vectors :> IVECS      /*in vector file*/
  .text    :> IRAM       /*Created by C Compiler*/
  .bss     :> IRAM
  .cinit   :> IRAM
  .stack   :> IRAM
  .sysmem  :> IRAM
  .const   :> IRAM
  .switch  :> IRAM
  .far     :> IRAM
  .cio     :> IRAM
  .csldata :> IRAM
}
```

La figure suivante montre une vue externe d'une carte d'implémentation TMS320C6713DSK de Texas Instruments.