Code Composer Studio®

Tutoriel rapide

Version de Code Composer Studio® 7.0.0

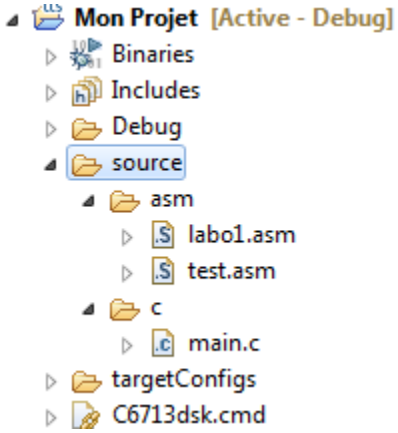
Les licences de Code Composer Studio® ainsi qu'une bonne partie des cartes DSP utilisés pour la session S5 Électrique sont un don de Texas Instruments

Auteur: David Beaudette

Mise à jour : Yves Bérubé-Lauzière, Rémi Pelletier, Bruno Gagnon, Julien Pichette, Daniel Gaucher (2017)

1.1 Structure d'un répertoire de projet

Sous CCS, vous pouvez utiliser l'emplacement de votre choix pour créer et gérer des projets, que ce soit votre lecteur réseau **U:**, votre clé USB personnelle ou autre. Toutefois, pour plus de sureté, il est conseillé de copier vos projets sur le disque local de l'ordinateur et de travailler à partir de cette version. Il est important de structurer vos répertoires de projet CCS de façon à obtenir un regroupement logique et efficace de votre code. La structure standard adoptée pour la session S5 est la suivante :



Les répertoires **Debug** et **Release** et **Mon_Projet** sont créés automatiquement.

Vous devez créer manuellement les répertoires **include**, **source**, **asm** et **c**.
Voici ce que l'on retrouve dans chacun des répertoires :

1. **Mon Projet** (racine) : Ce répertoire contient, outre les autres répertoires :
   a) les fichiers de projet de CCS (.ccsproject, .cproject, .project)
   b) la sortie écran lors de la dernière compilation (.log)
   c) un dossier **targetConfigs** listant les options utilisées par le linker

2. **Debug** contient :
   a) l'exécutable (.out)
   b) le linker map listing (.map)
   c) les fichiers compilés (.obj)

3. **Includes** contiendra tous les fichiers d'en-tête (.h) propres au projet
4. **source** contiendra uniquement les sous-répertoires **c** et **asm**
5. **c** contiendra tous les fichiers de code source en language C (.c) propres au projet
6. **asm** contiendra tous les fichiers de code source en assembleur (.asm) propres au projet

1.2 Tutoriel

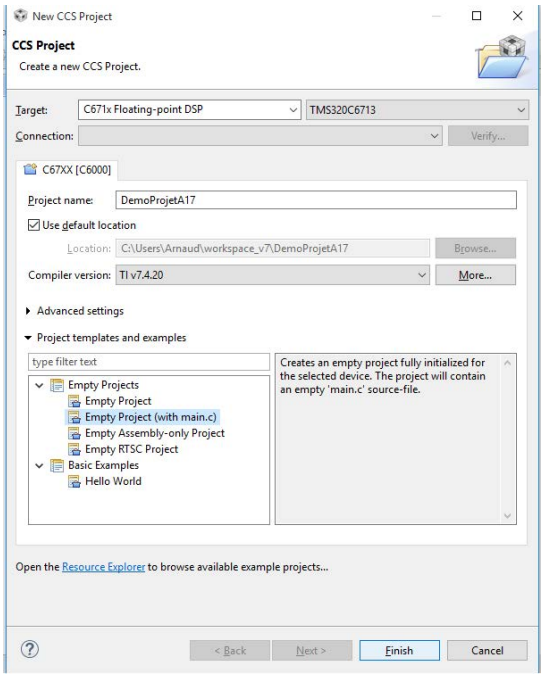
Note : Pour la session, il faut utiliser la procédure décrite ici et non pas celle donnée en exemple aux pp.11-15 dans le livre *Digital Signal Processing* de Chassaign. Ne récupérer que les fichiers source des exemples du livre. Si vous voulez faire les exemples du livre de Chassaign, utiliser toutefois la procédure ici. Ceci ne vous empêche pas de faire les exemples du livre tels que décrits pour votre curiosité, mais pour la session, la procédure décrite ici prévaut.

Voici les opérations nécessaires pour créer le projet en partant de rien :

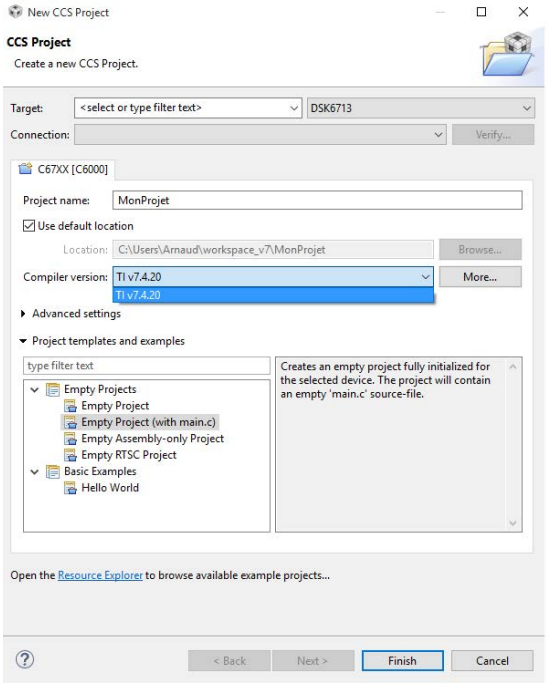
1. **Brancher le DSK+US**
   Brancher le module DSK+US au port USB du PC ainsi qu'à son bloc d'alimentation. Attendre que les 4 LEDs D7 à D10 restent allumées après qu'elles aient clignoté.
   Ouvrir Code Composer Studio 5.5.0 (ci-après nommé CCS).
   Démarrer > Électronique > Texas Instruments > Code Compose Studio

2. **Répertoire de travail (*workspace*)**
   Travailler sur un répertoire local (CCS ne prend pas les liens réseaux).

3. **Créer un projet**
   Dans CCS, faire *File > Project > New > CCS Project*.

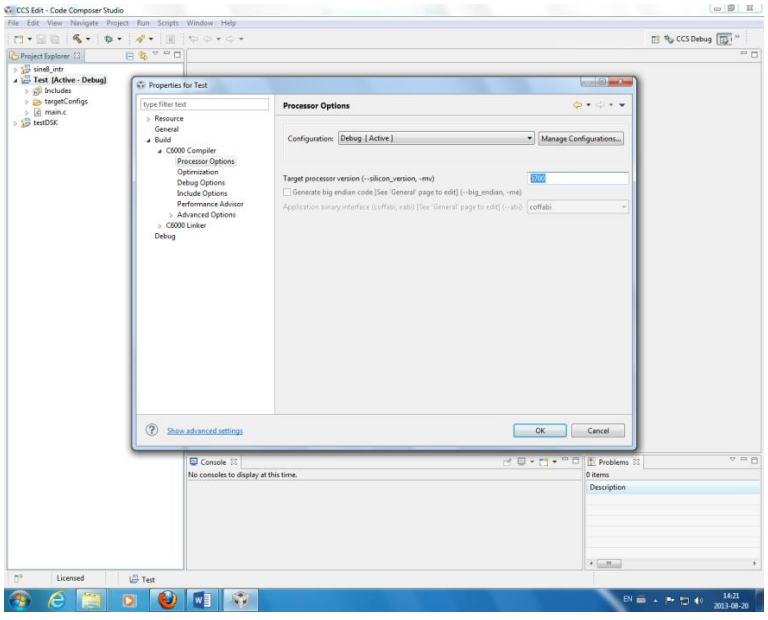


4. **On choisit le nom du projet, *Family* à « C6000 », *Variant* à « DSK6713 », *Connection* à « Spectrum Digital DSK-EVM-eZdsp onboard USB Emulator, et *template* à « Empty Project (with main.c)" »**
   OU
   Si vous travaillez uniquement avec le simulateur, prenez la famille "C670x Floating-point DSP" et la variante "TMS320C6701"

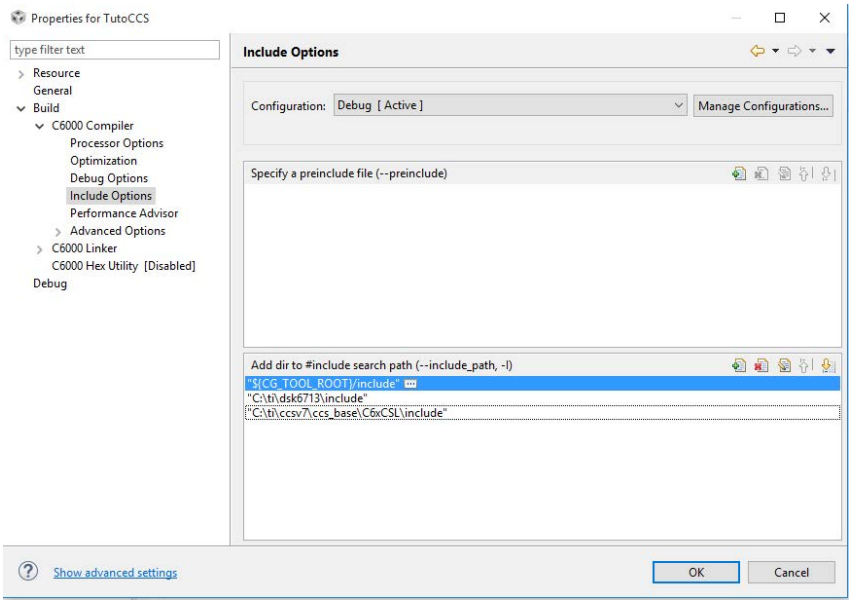


5. **Modifier les propriétés du projet (**allez dans *Project > Properties*)

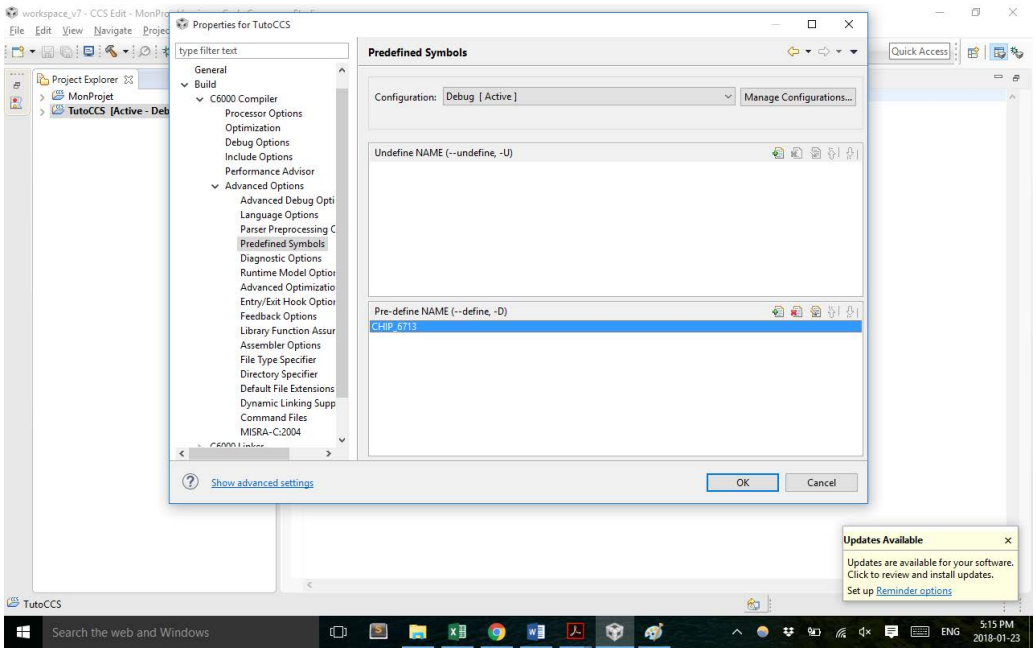
   Dans les *Build > C6000 Compiler > Processor Options*, on écrit “6700” dans le champ *Target Processor Version* :



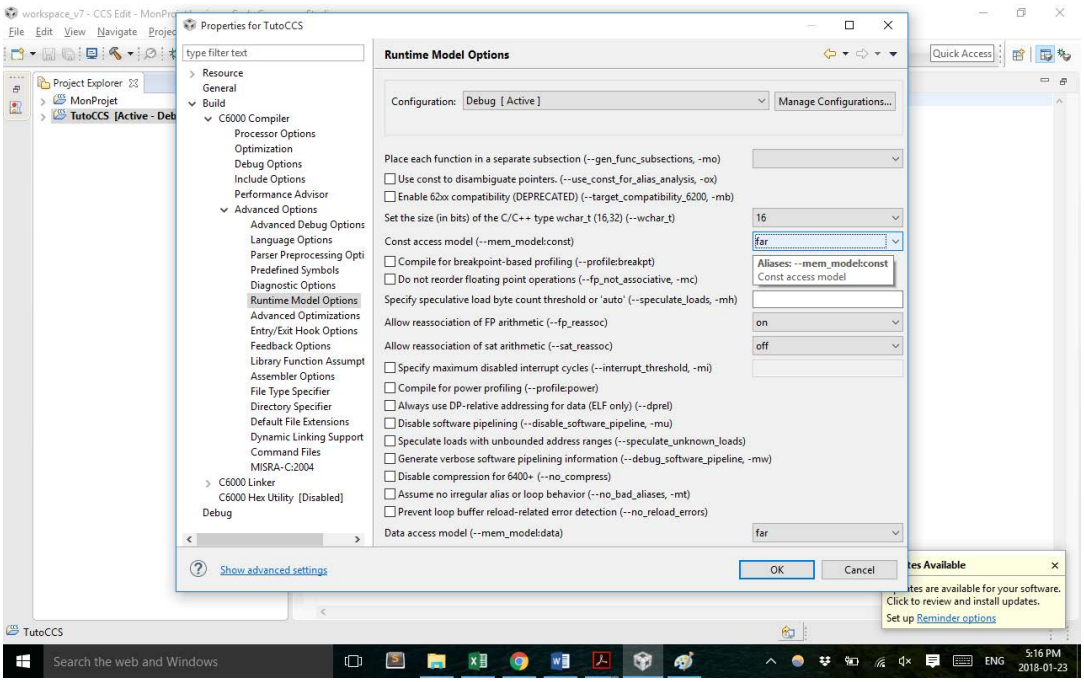
Dans les *Build > C6000 Compiler > Include Options*, on rajoute les répertoires “C:\ti\ccsv7\ccs_base\C6xCSL\include” et “C:\ti\dsk6713\include”



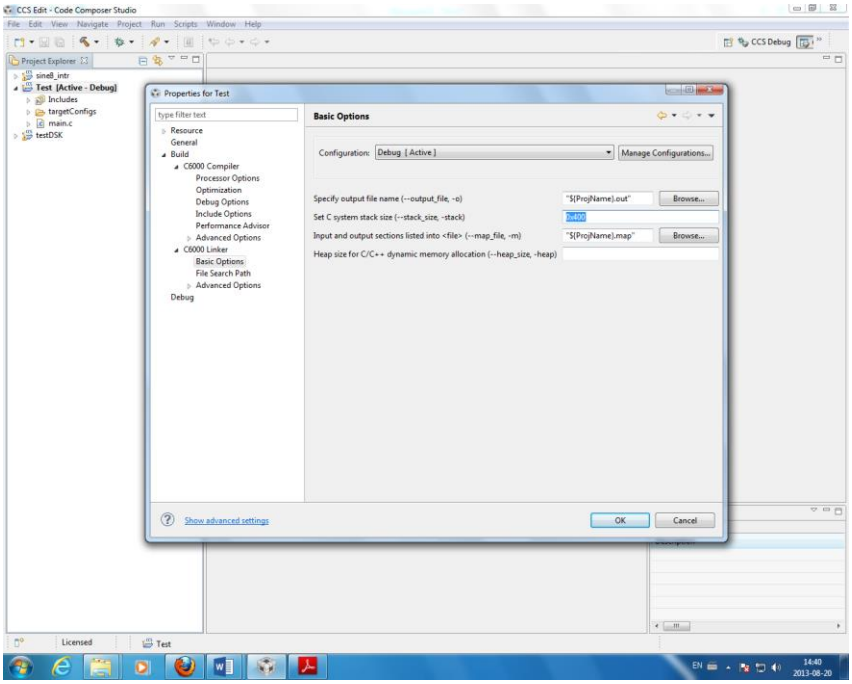
Dans *Build > Advanced Options > Predefined Symbols*, on rajoute “CHIP_6713”



Dans *Build > Advanced Options > Runtime Model Options*, on met : “Const access model (--mem_model:const)” à “far” et “Data access model (mem_model:data)” à “far”

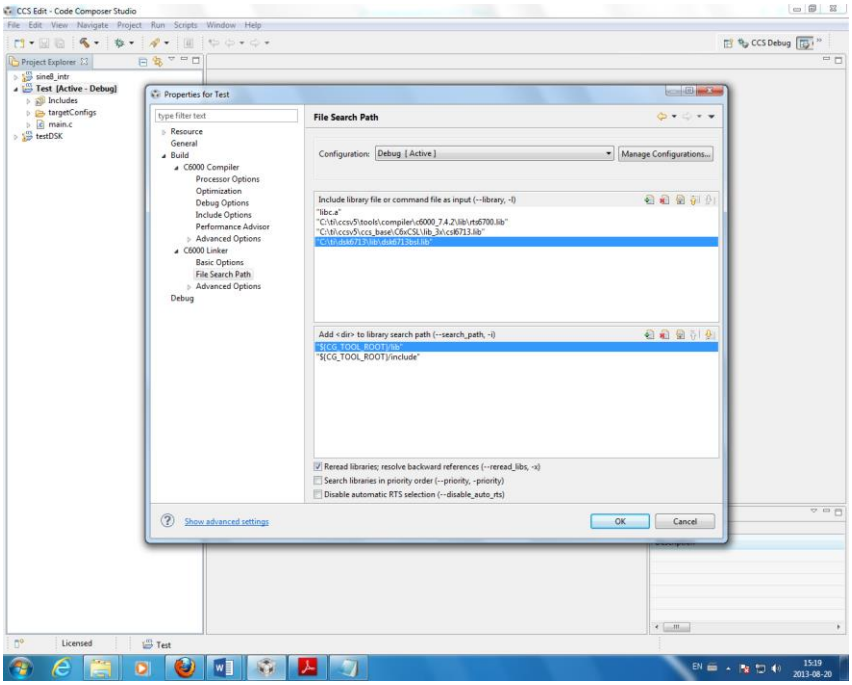


Dans *Build > C6000 Linker > Basic Options*, on met le stack size **et** le heap size à 0x1000



On rajoute les fichiers de support dans l'onglet *Build > C6000 Linker > File Search* Path :

- Le real time support (rts) : "C:\ti\ccsv7\tools\compiler\c6000_7.4.20\lib \rts6700.lib" Si ce fichier n'existe pas, CCS va le construire pour vous
- Le chip support library (csl) : "C:\ti\ccsv7\ccs_base\C6xCSL\lib_3x\csl6713.lib"
- Le board support library (bsl) : "C:\ti\dsk6713\lib\dsk6713bsl.lib"



6. **Ajouter le fichier linker (gestion de la mémoire du DSK)**

   Si vous voulez utiliser le simulateur, supprimez tous les fichiers .cmd présents dans votre projet, sautez cette étape et allez voir la section 1.3.

   Copier le fichier C:\ti\dsk6713\cmd\C6713dsk.cmd dans le répertoire du projet. Il s'ajoutera automatiquement dans l'arborescence du projet.

7. **Main**

   On peut maintenant éditer le fichier main.c qui contiendra la fonction du même nom. Ici est un exemple de *Hello World* :

```
/* Hello World program */

#include <stdio.h>

main()
{
    printf("Hello World");

}
```

8. **Exécuter votre projet**

   Il suffit d'appuyer sur le bouton *Debug* et l'interface basculera dans ce mode. Le code sera automatiquement compilé et chargé dans le DSP. Vous devriez arriver à créer ainsi un projet qui compile sans erreur.

9. **Exécuter son code dans le module DSK+US**

   Debug > *Run* : permet d'exécuter votre code.
   Debug > *Step Into* : permet d'exécuter votre code pas à pas
   Debug > *Restart* : permet de réinitialiser votre code.

   Notez qu'à la fin de l'exécution du code dans le Debugger, vous aurez une erreur :

   Can't find a source file at ".../ src/exit.c"
   Locate the file or edit the source lookup path to include its location.

   Cette erreur est due au fait qu'à la fin de l'exécution, le Debugger essaie d'ouvrir le fichier source correspondant à la fonction exécutée au *breakpoint* de fin du programme qui se trouve dans la *C runtime support library* dont le source n'est pas disponible. Ignorez simplement cette erreur, elle ne fait pas partie de l'exécution de votre code.

10. **Visualiser vos résultats en temps réel dans le module DSK+US**

    Il suffit d'ajouter des variables à la *Watch List*. Les onglets *Variables*, *Expressions* et *Registers* permettent de surveiller ce qui se passe à l'intérieur du DSP et de valider l'exécution du code.

Note : Le *Build Configuration* de CCS est par défaut à *Debug*. Si on veut passer en mode *Release*, il faut refaire la configuration au complet en répétant les étapes 5 et 6 après avoir changé *Project* > *Build Configurations* > *Active* de *Debug* à *Release*. Toutefois, dans le cadre de la session, c'est le mode *Debug* qui sera principalement utilisé.

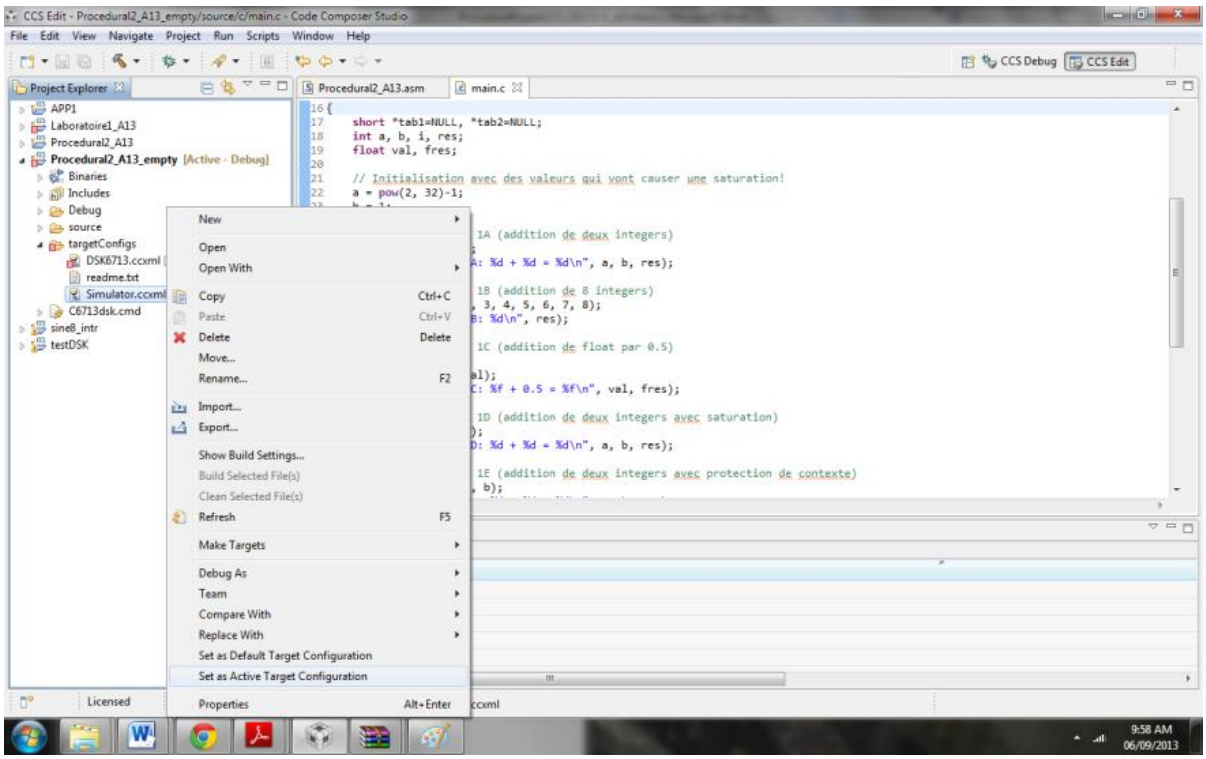
Note : Pour vous familiariser avec CCS, il vaut la peine de prendre le temps d'écouter le « *Getting Started* » proposé à l'ouverture du programme. Vous pouvez aussi vous référer à ce lien expliquant certain concepts derrière la structure des projets sur la plate-forme *Eclipse* : http://processors.wiki.ti.com/index.php/Eclipse_Concepts

1.3 Utilisation du simulateur

Pour l'utilisation du simulateur, après avoir complétée la procédure ci-haut (section 2.1), on copiera le fichier **Simulator.ccxml** du répertoire C:\ti\ dans le sous-répertoire \targetConfigs\ de votre projet. Ce fichier ce trouve déjà dans le répertoire c:\ti .

Pour éviter des problèmes potentiels, supprimez tous les autres fichiers .cmd et .ccxml qui se trouvent dans votre projet pour éviter des conflits (à moins que vous sachiez ce que vous faites).

Il suffit ensuite d'aller dans l'arborescence du projet et dans le répertoire targetConfigs et avec un right-click sur le fichier **Simulator.ccxml** de sélectionner l'option « Set as Active Target Configuration » :



1.4 Fichiers de support

Les compagnies Spectrum Digital Inc., Texas Instruments Inc. et John Wiley & Sons Inc. ont fourni (à l'achat du DSK, de CCS, et du livre de Chassaing, respectivement) différents fichiers dont la plupart des projets de la session vont se servir. Il s'agit de librairies, de code source et de fichiers d'entêtes.

L'équipe professorale et le personnel de soutien de la session a jugé bon de regrouper tous ces fichiers dans un seul endroit, soit dans le répertoire **C:\ti\dsk6713**. Cette pratique évite le dédoublement de fichiers (vous n'avez pas à recopier les librairies pour chaque nouveau projet) et permet de standardiser les options de compilation et d'assemblage.

Voici la description des fichiers que l'on y trouve.

SupportFiles\	cmd\		C6713dsk.cmd	Linker command file
	include\		C6713dskinit.h	Déclaration de fonctions haut niveau du livre de Chassaing pour utiliser le codec
			dsk6713.h	Déclaration de fonctions et définitions de constantes du BSL pour le CPLD
			dsk6713_aic23.h	Déclaration de fonctions et définitions de constantes du BSL pour le codec
			dsk6713_dip.h	Déclaration de fonctions et définitions de constantes du BSL pour les interrupteurs de SW1
			dsk6713_flash.h	Déclaration de fonctions et définitions de constantes du BSL pour la mémoire FLASH
			dsk6713_led.h	Déclaration de fonctions et définitions de constantes du BSL pour les LEDs D7 à D10
	lib\		csl6713.lib	Librairie de fonctions pour les éléments internes du processeur C6713
			dsk6713bsl.lib	Librairie de fonctions pour les éléments du DSK (codec, LEDs, etc.)
			rts6700.lib	Définition des fonctions du langage C sur le C67x
	source\	asm\	c6713dskinit.asm	Définition de fonctions haut niveau du livre de Chassaing pour utiliser le codec (version assembleur)
			Vectors_intr.asm	Vecteurs d'interruption pour utiliser le codec par interruption dans les exemples de Chassaing
			Vectors_poll.asm	Vecteurs d'interruption pour utiliser le codec par interrogation dans les exemples de Chassaing
		c\	c6713dskinit.c	Définition de fonctions haut niveau du livre de Chassaing pour utiliser le codec (version C)