

Compte rendu :

Tp1: Initiation au ccs v4

Tp2: Optimisation

Tp3: Image processing

Tp4: Image processing + Optimisation

LOGICIEL: CODE COMPOSER STUDIO V4

Khelifi Marwa

4ème Informatique G2.2

Objectifs des TP

L'objectif de nos tps c'est l'apprentissage de l'utilisation du logiciel Code Composer Studio (CCS4) : se familiariser avec le développement d'un système DSP de Texas Instruments en utilisant le langage C et langage assembleur, analyser et étudier des programmes avec des techniques d'optimisation sur DSP.

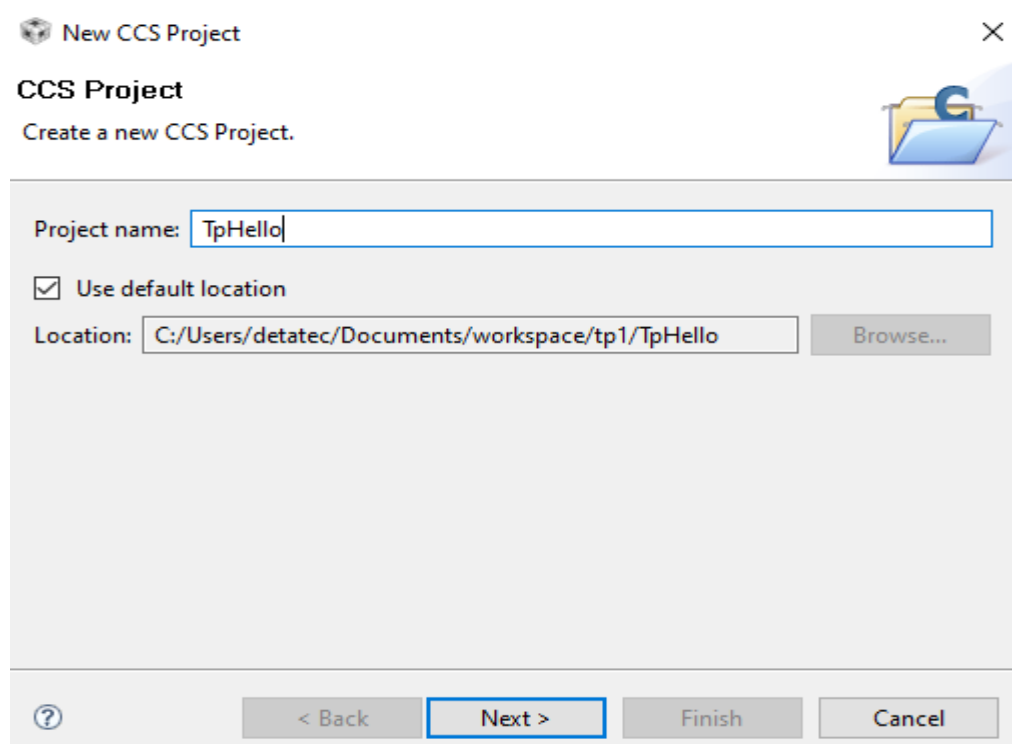
TP1

Définition de Code Composer Studio :

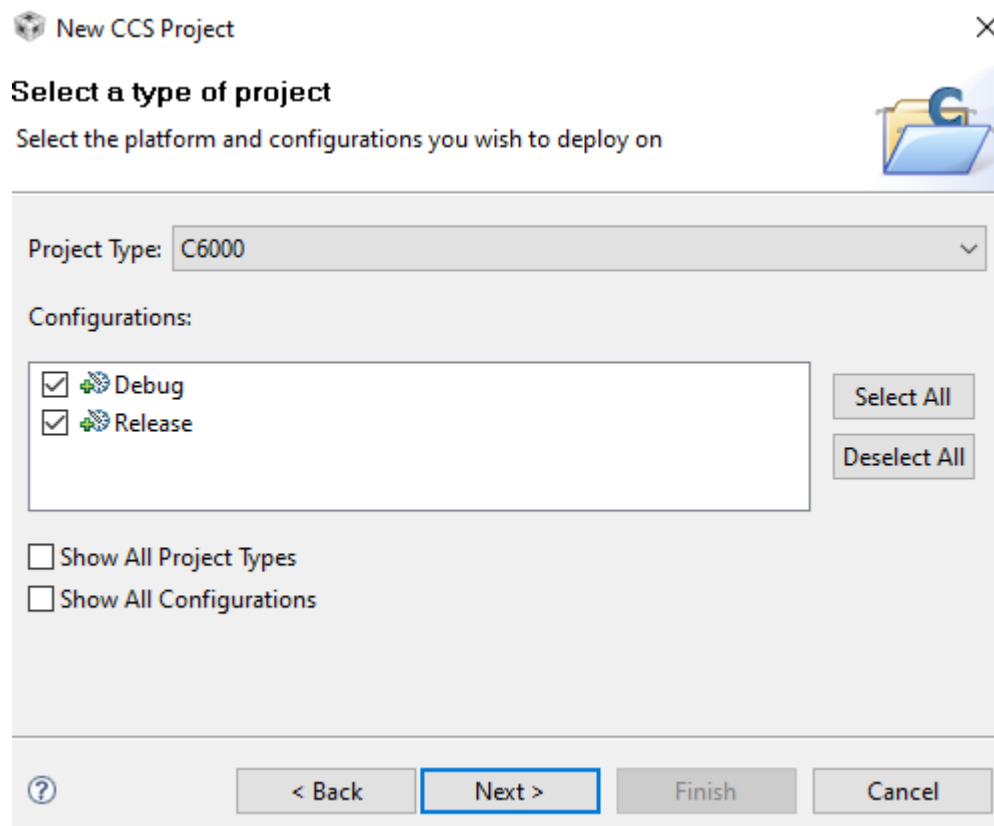
Code Composer Studio (CCS) fournit plusieurs outils pour faciliter la construction et la mise au point des programmes de DSP. Il comprend un éditeur de code source, un compilateur de langage C/C++, un assembleur de code relocalisable, un éditeur de liens, et un environnement d'exécution qui permet de 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. Finalement, il fournit un environnement de gestion de fichiers qui facilite la construction et la mise au point des programmes.

1. Les étapes de création d'un projet :

Pour créer un projet, on va choisir l'élément de menu « **File → New → CCS Project** » de la barre menu. Dans celle-ci, on va introduire un nom de projet dans le champ « **Project name** » (TpHello.c).



On va poursuivre avec le type du projet en sélectionnant : « **Project Type** : C6000 ».



Ici, on va sélectionner dans «**Device Variant** : Generic C64xx Device» et dans le «**Runtime Support Library** : RTS6400.lib ».

New CCS Project

Project Settings
Select the project settings.

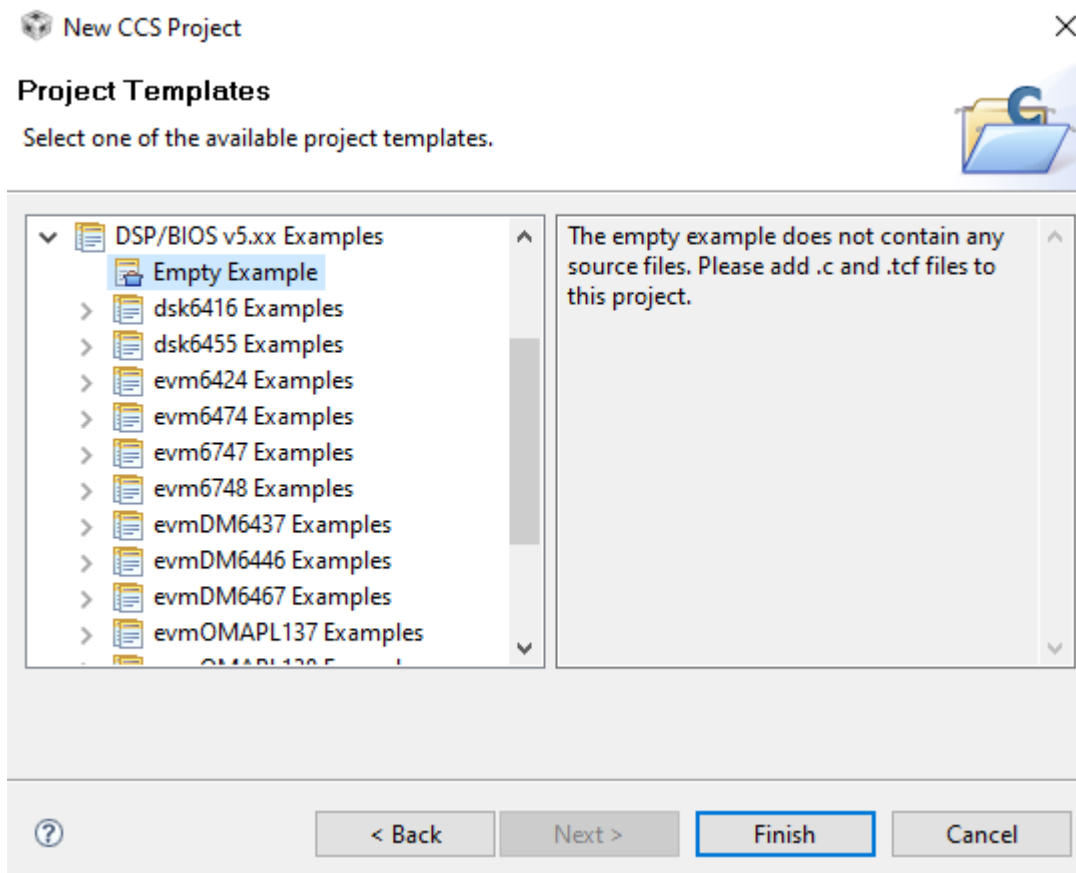
Output type: Executable

Project settings

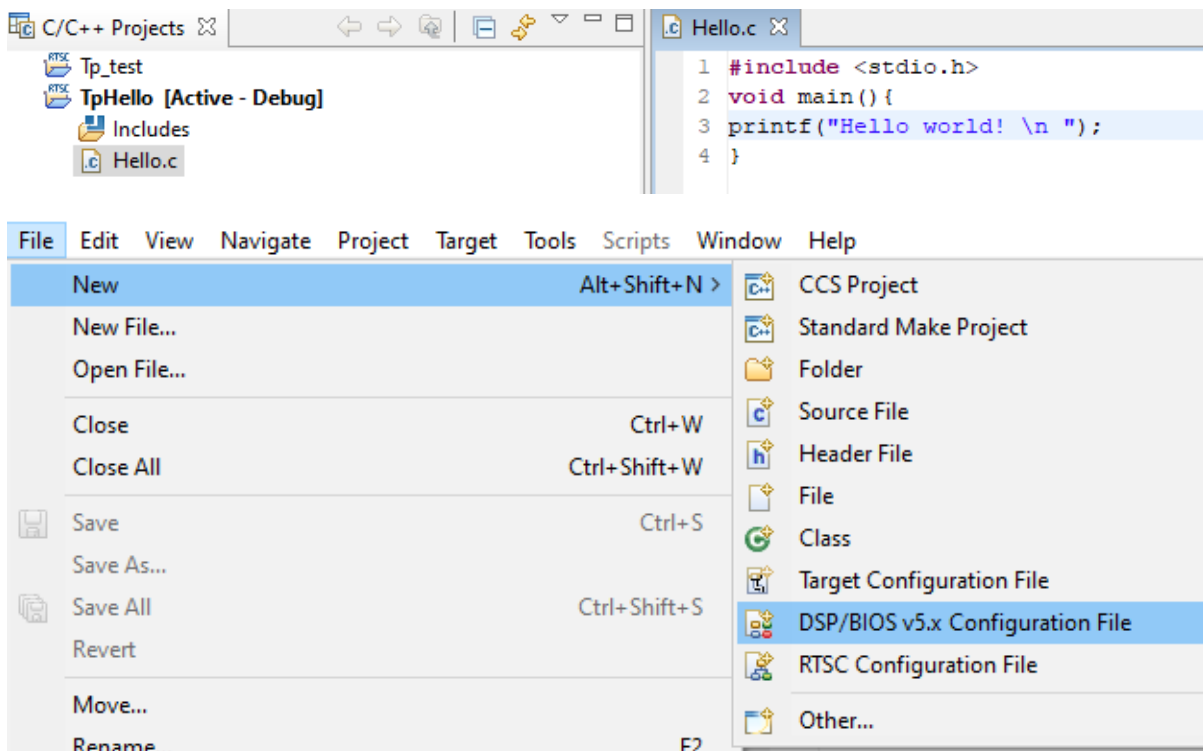
Device Variant:	Generic C64xx Device	More...
Device Endianness:	little	
Code Generation tools:	TI v7.2.0	More...
Output Format:	legacy COFF	
Linker Command File:		Browse...
Runtime Support Library:	rts6400.lib	Browse...

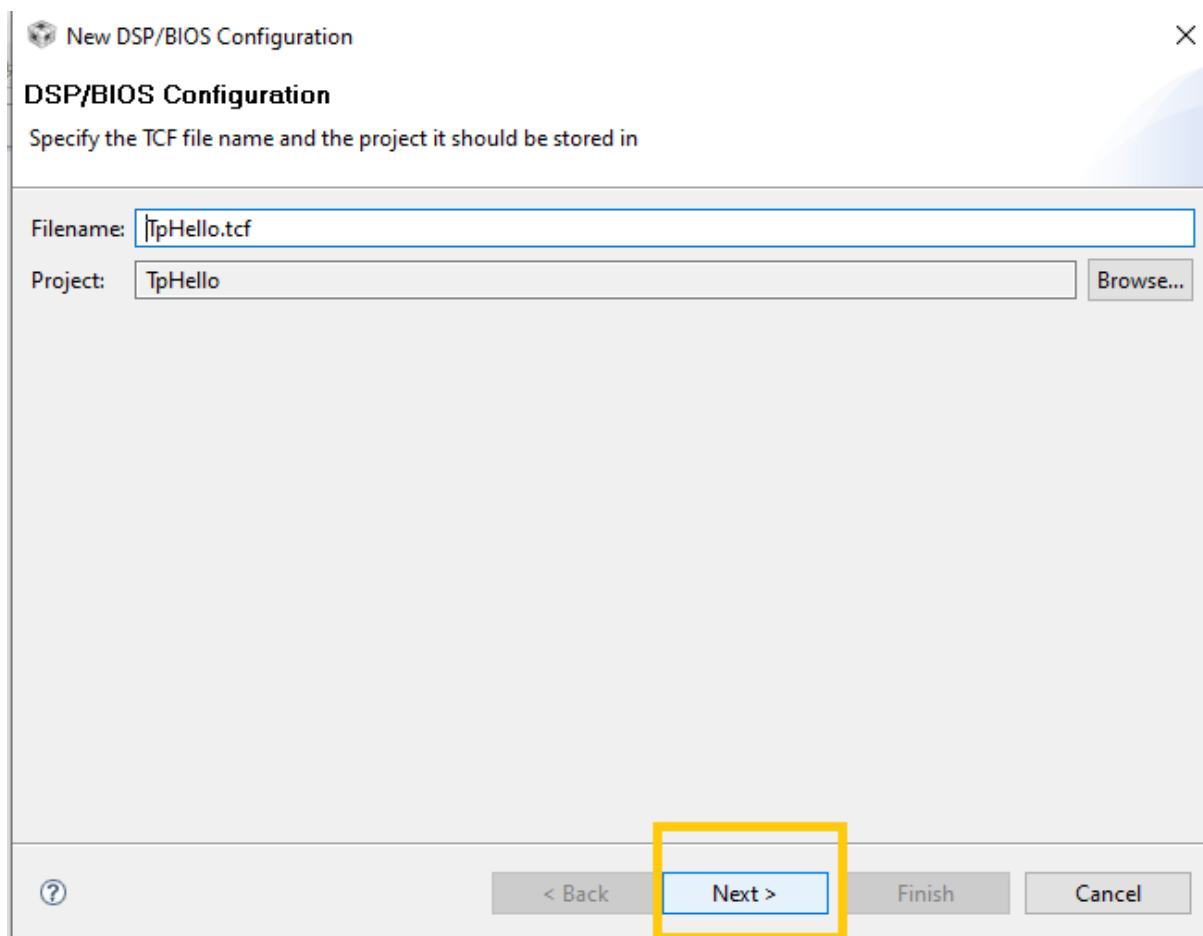
? < Back Next > Finish Cancel

Pour finir, on va sélectionner dans la fenêtre «**Project Templates**» l'item «**DSP/BIOS v5.xx Examples / Empty Exemple** ».



Pour la mise au point, on va ajouter un fichier source en C que j'ai l'appelé « main.c » puis on va ajouter le fichier de configuration de DSP/BIOS et on va sélectionner **"File New DSP/BIOS Configuration File"** dans le but de spécifier la plateforme DSP à utiliser.





The image shows a 'New DSP/BIOS Configuration' dialog box. At the top, it says 'Specify the TCF file name and the project it should be stored in'. There are two input fields: 'Filename:' with the text 'TpHello.tcf' and 'Project:' with the text 'TpHello'. To the right of the 'Project:' field is a 'Browse...' button. At the bottom, there are four buttons: a help button with a question mark, '< Back', 'Next >' (which is highlighted with a yellow rectangle), 'Finish', and 'Cancel'.

New DSP/BIOS Configuration

DSP/BIOS Configuration

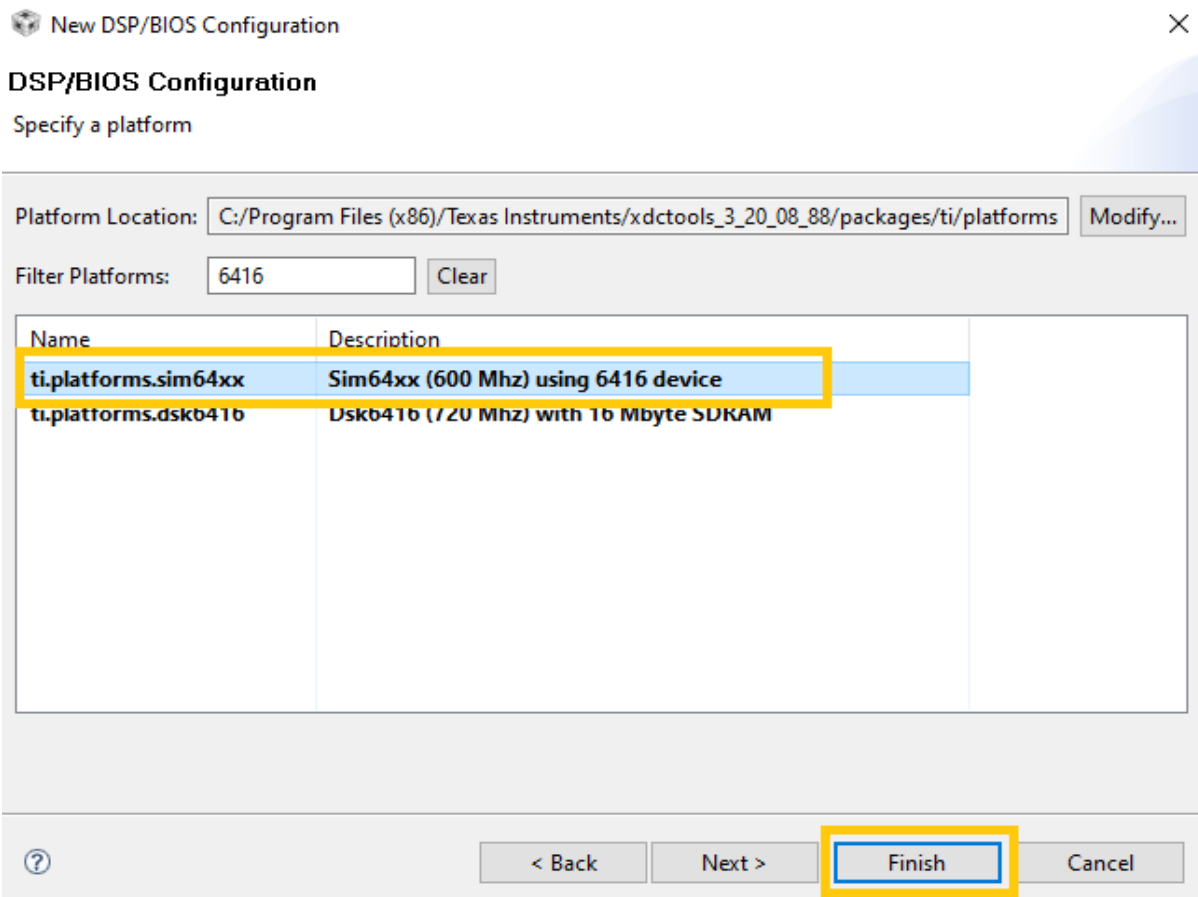
Specify the TCF file name and the project it should be stored in

Filename: TpHello.tcf

Project: TpHello Browse...

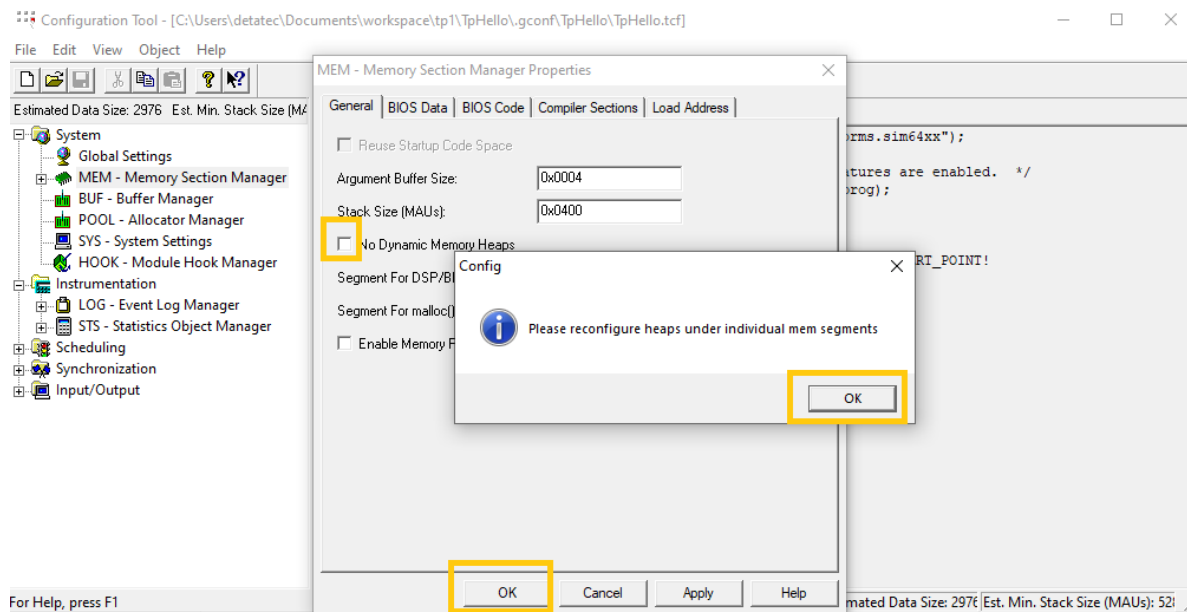
? < Back Next > Finish Cancel

Ici, on va choisir la plateforme simulateur 6416 (*ti.platforms.sim64xx*).



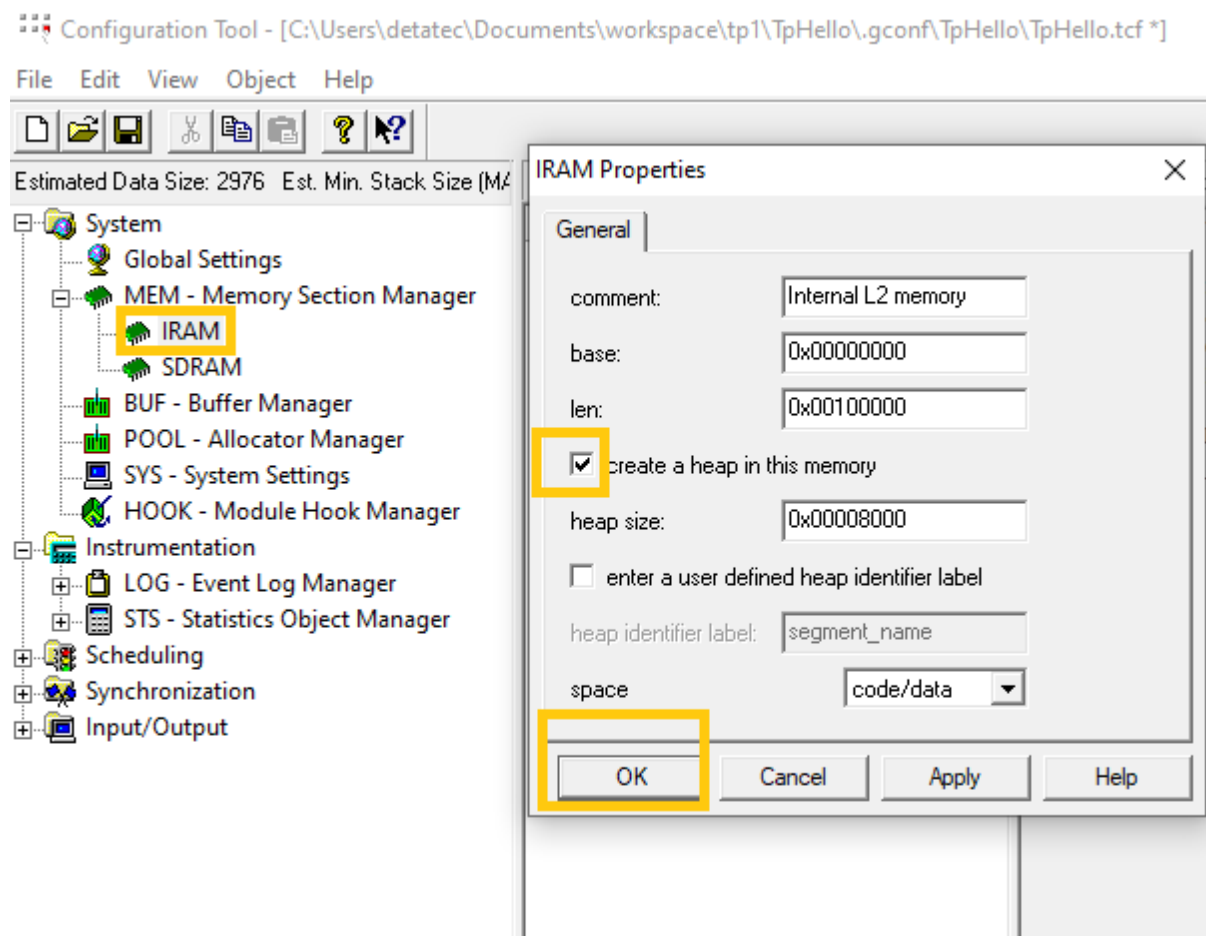
Après cette étape une fenêtre sera affichée, on va faire quelques modifications dans les propriétés de "MEM", "IRAM" et "SDRAM" et on va configurer les différentes sections mémoires du fichier de configuration DSP/BIOS "Hello.tcf". Ce fichier permet de faire une gestion des mémoires et de modifier les paramètres de RTDX.

On va cliquer avec le bouton droit sur "MEM-Memory Section Manager" puis on va choisir **Properties** et décocher la case "No Dynamic Memory Heaps" .

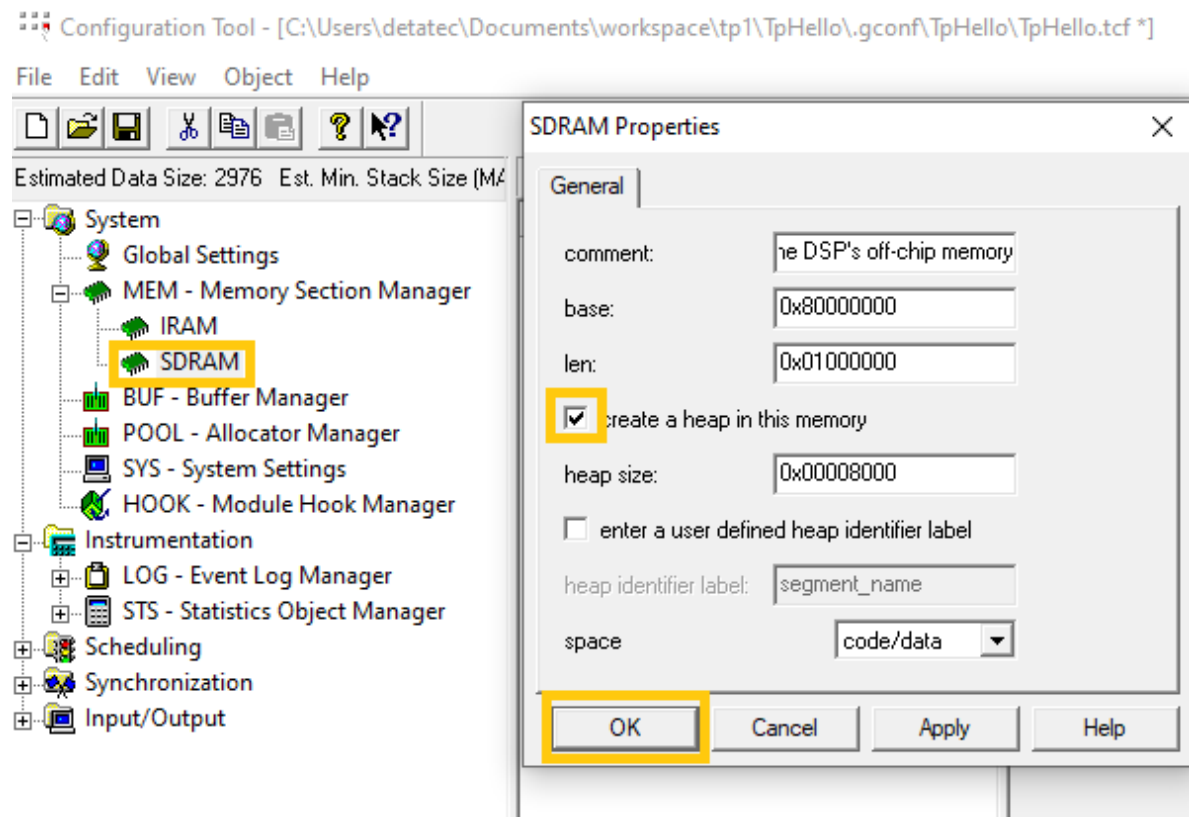


Pour activer la création de zone mémoire “heap” de mémoire SDRAM qui est la mémoire externe, on va changer les propriétés de IRAM et SDRAM.

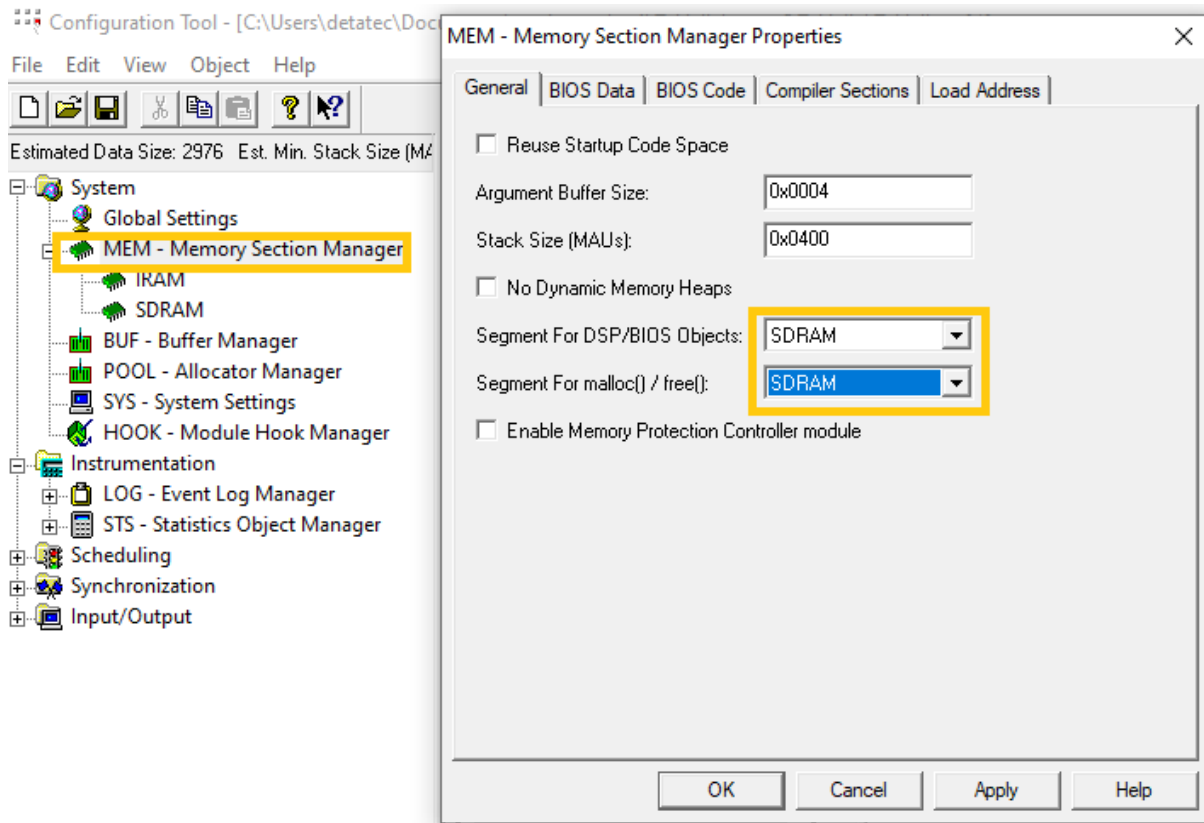
Pour la mémoire IRAM:



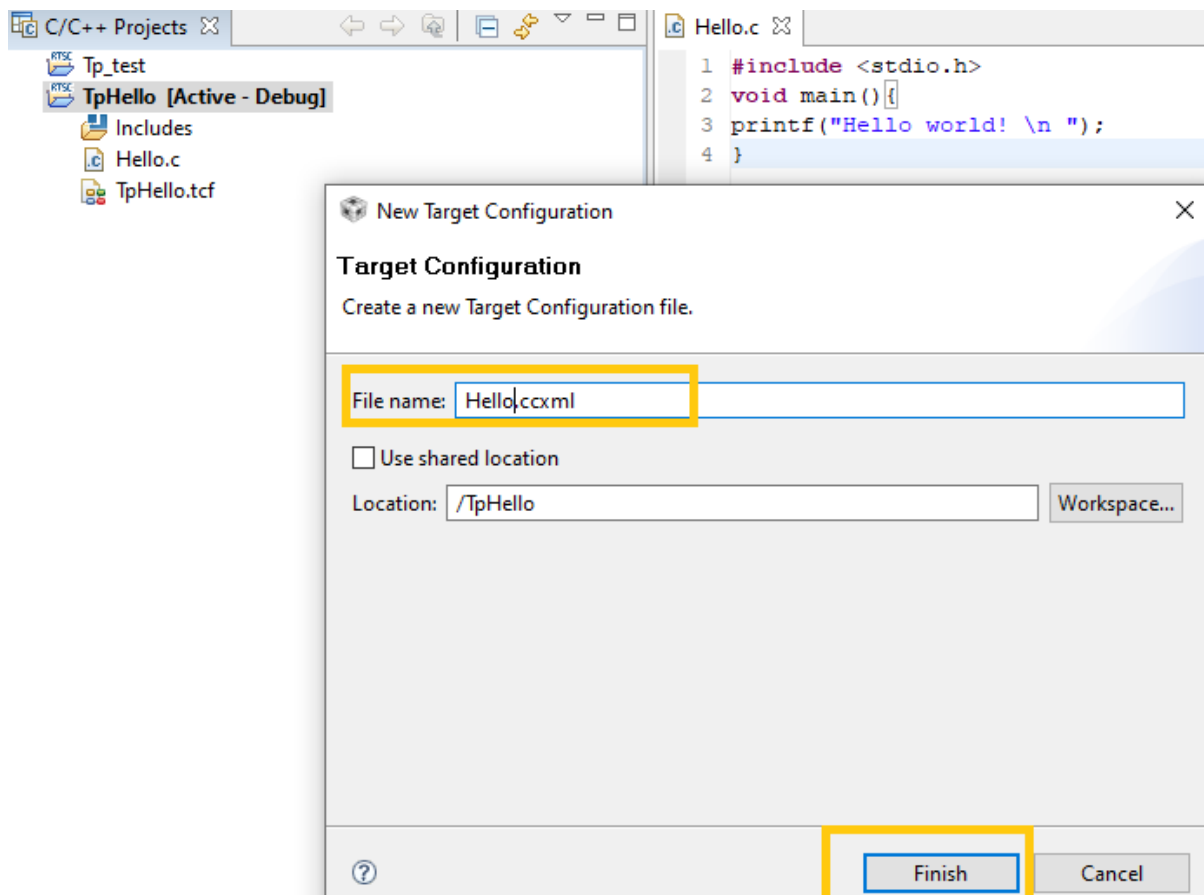
Pour la mémoire SDRAM:



On va retourner à la propriété du MEM et on va changer ses segments pour la bonne gestion des mémoires et le bon fonctionnement du DSP.



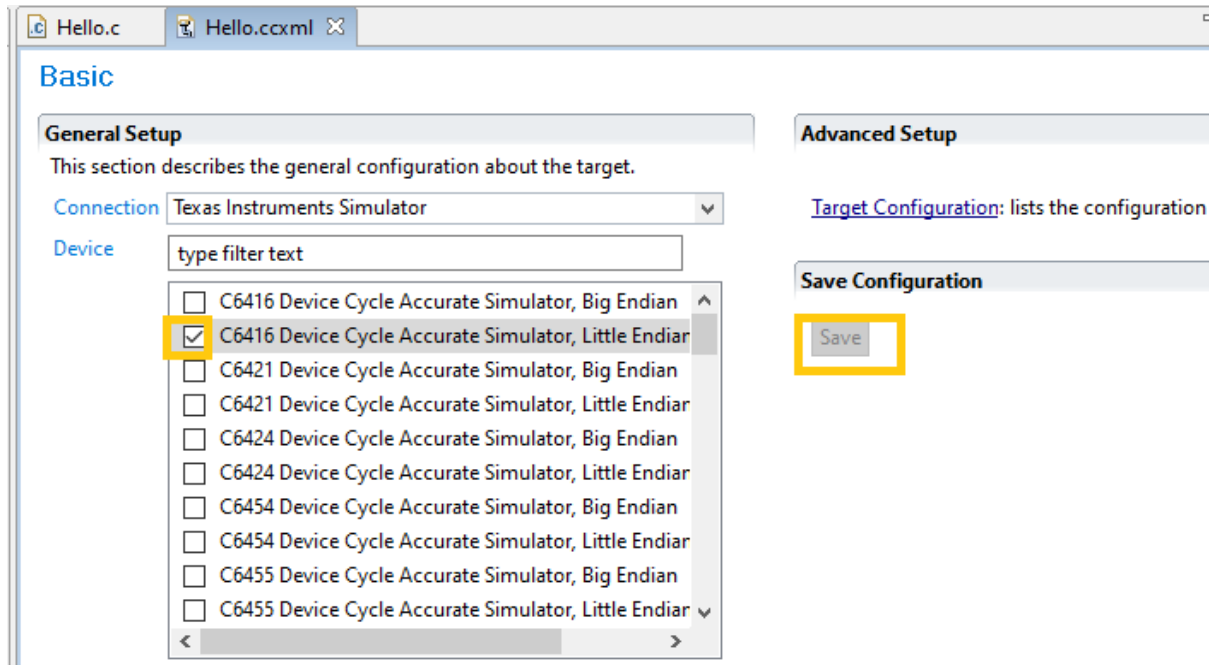
Maintenant, on va ajouter un **"Target Configuration File"** en cliquant avec le bouton droit sur le nom du Projet **"New Target Configuration File"** et on va le nommer par **"Hello.ccxml"**



Pour définir la cible de l'implantation, On va sélectionner notre simulateur. Dans ce dernier, on va choisir la configuration suivante :

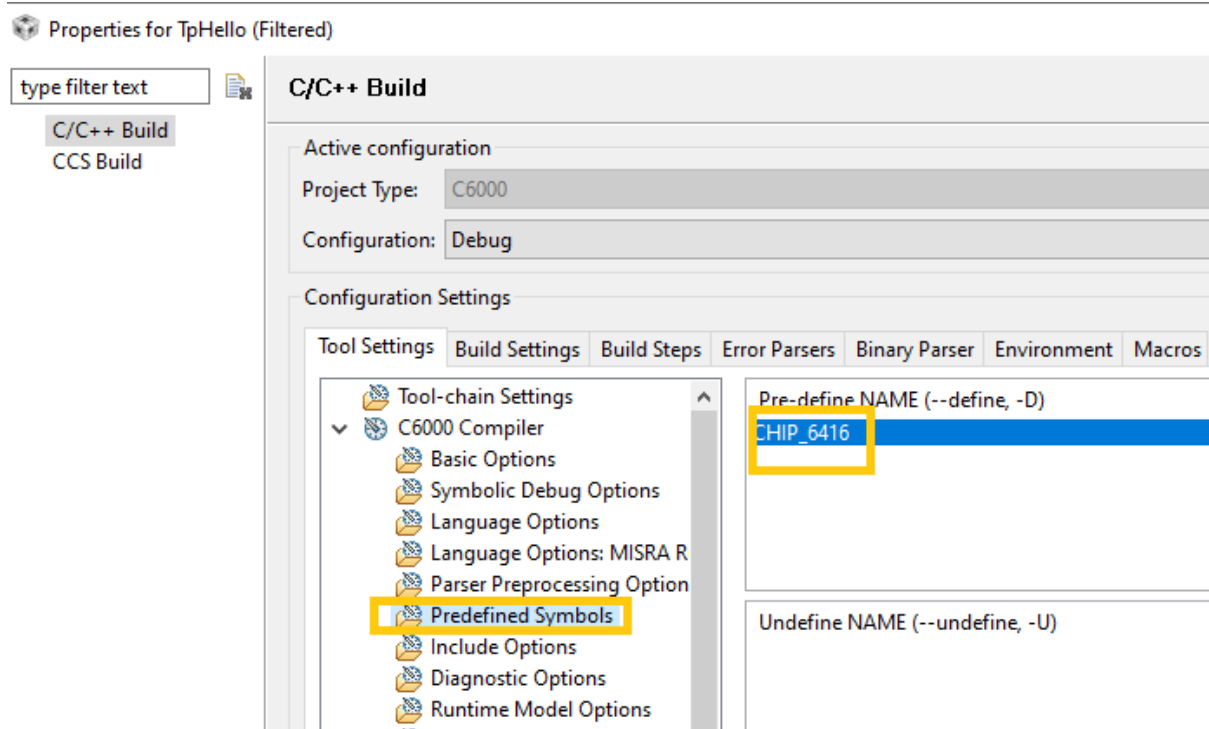
Connection : Texas Instruments Simulator .

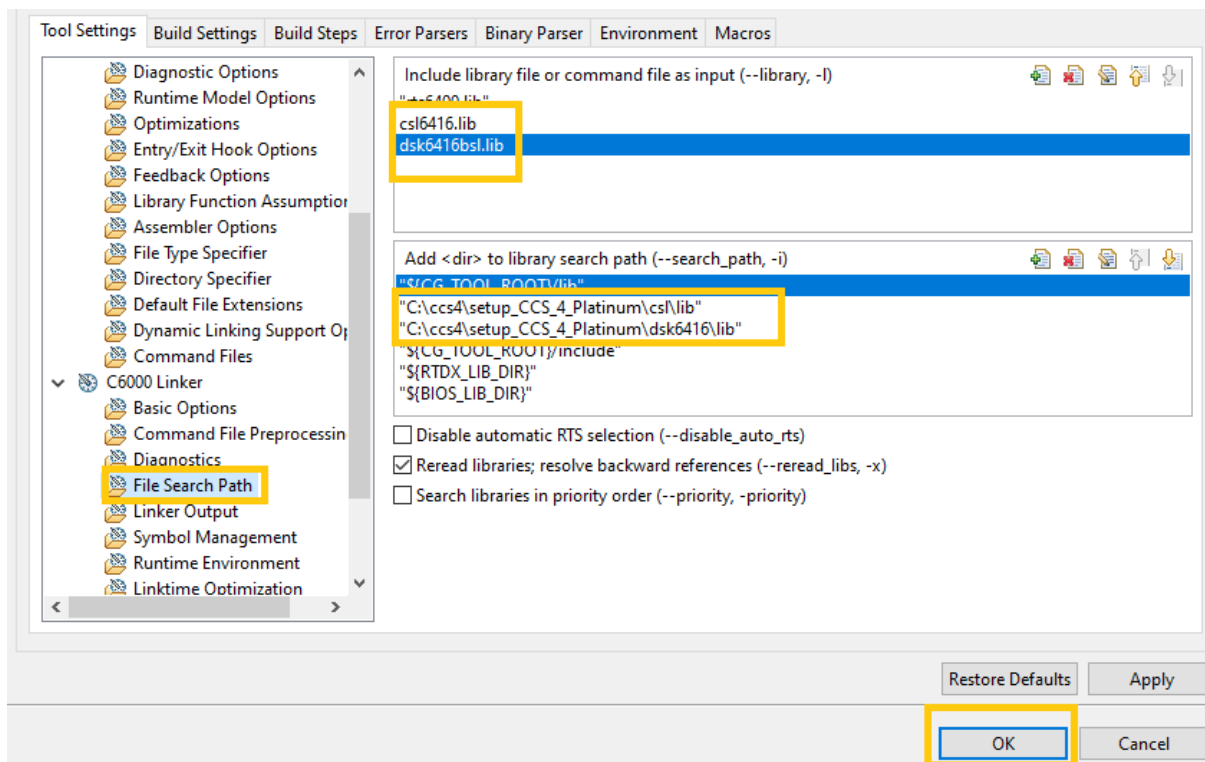
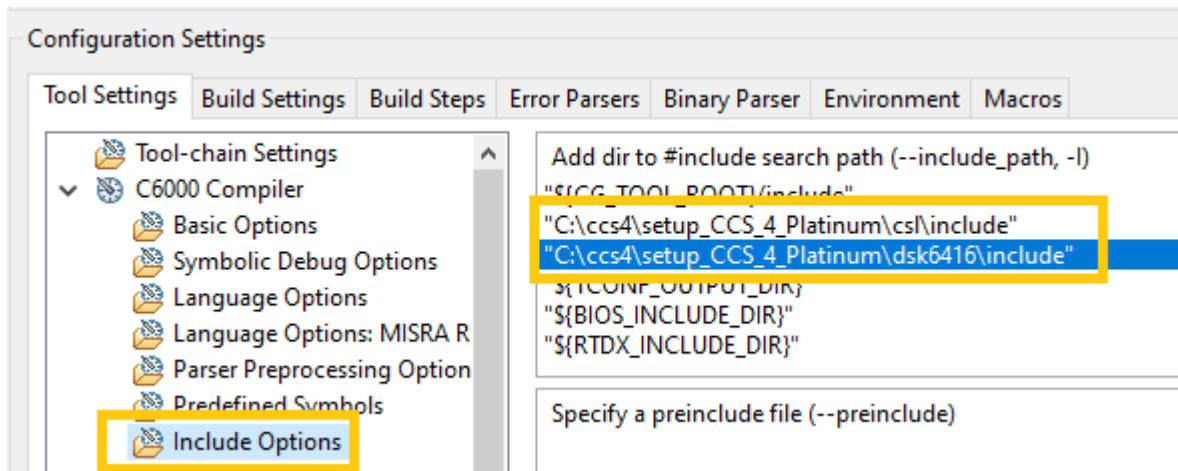
Device : C6416 Device Cycle Accurate Simulator, Little Endian.



2. Compilation du projet:

Pour compléter notre configuration, on va cliquer sur le projet et on va choisir **“Build properties”**. Il faut faire quelques modifications dans **“Predefined Symbols, Include Options et file search path”**



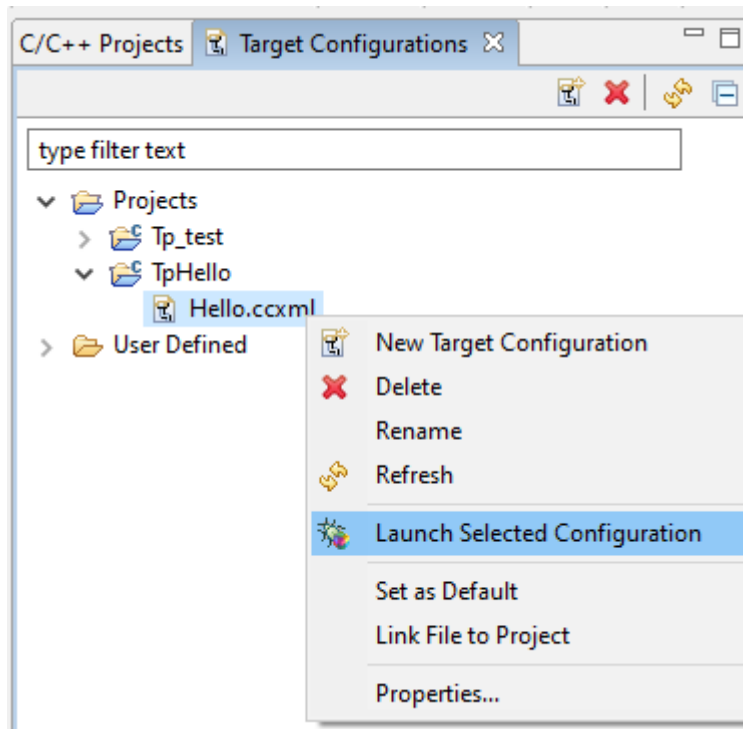


Pour compiler le projet, on va cliquer sur le bouton droit sur le nom du projet et sélectionner “ **Build Project** ” et on va choisir “ **Build Active Project** ” pour générer notre *fichier.out* si la compilation se termine avec succès (pas d’erreurs) .

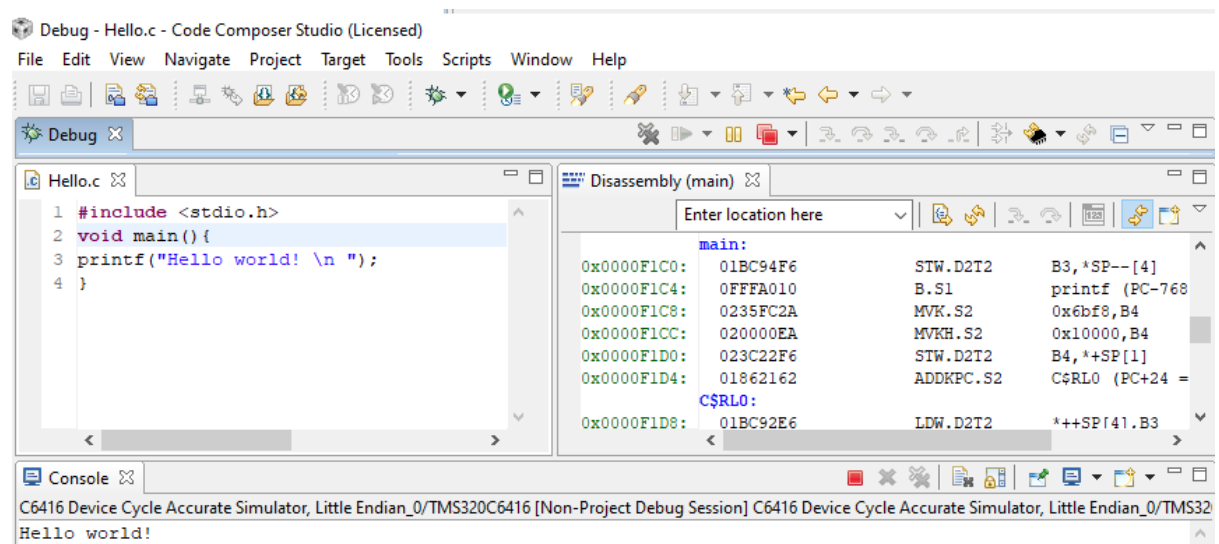


3. Execution du projet

On va ouvrir le fichier de configuration avec le bouton droit de la souris sur *Hello.ccxml* et on va sélectionner “**Launch Selected Configuration**”.



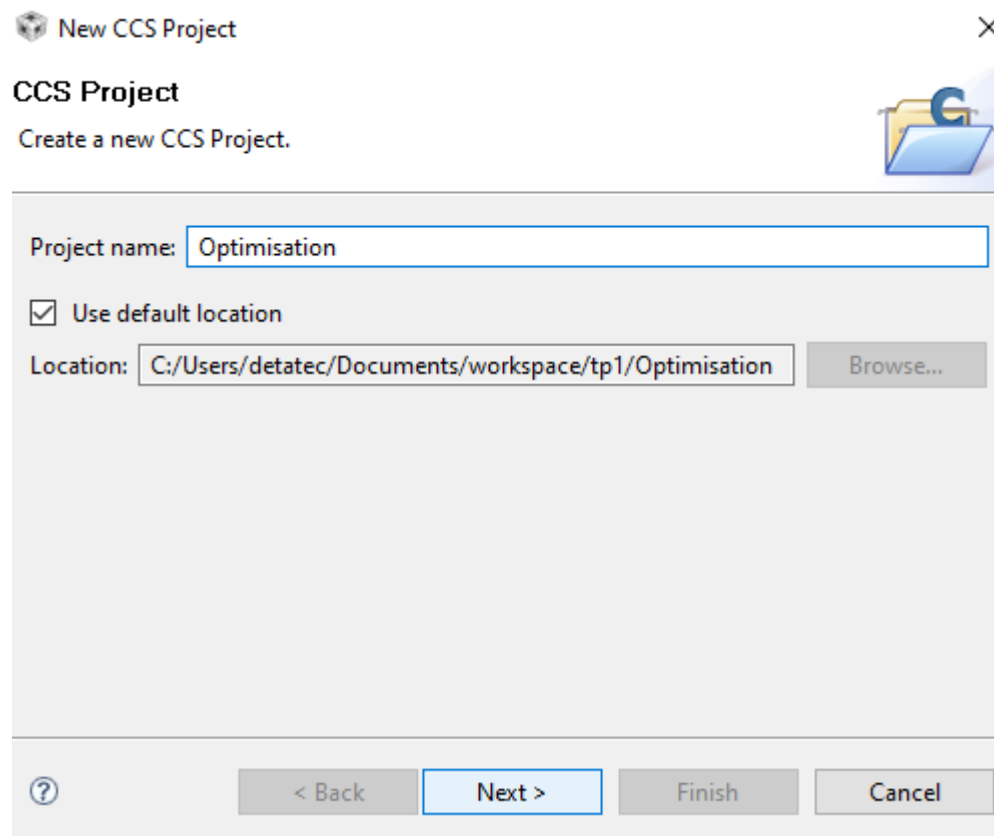
Finalement, on va cliquer sur **RUN** pour obtenir le résultat d'exécution de notre programme et on va obtenir un affichage de notre message du fichier “*main.c*” sur la console comme le montre ci-dessous:



TP2

1. Création d'un nouveau projet

On va créer un nouveau projet "Optimisation" donc on doit suivre les étapes suivante :



New CCS Project

CCS Project

Create a new CCS Project.

Project name: Optimisation

☒ Use default location

Location: C:/Users/detatec/Documents/workspace/tp1/Optimisation Browse...

? < Back Next > Finish Cancel

Select a type of project

Select the platform and configurations you wish to deploy on



Project Type: C6000

Configurations:

- ☒ Debug
- ☒ Release

Select All

Deselect All

- ☐ Show All Project Types
- ☐ Show All Configurations



< Back

Next >

Finish

Cancel

Project Settings

Select the project settings.



Output type: Executable

Project settings

Device Variant: Generic C64xx Device

More...

Device Endianness: little

Code Generation tools: TI v7.2.0

More...

Output Format: legacy COFF

Linker Command File:

Browse...

Runtime Support Library: rts6400.lib

Browse...

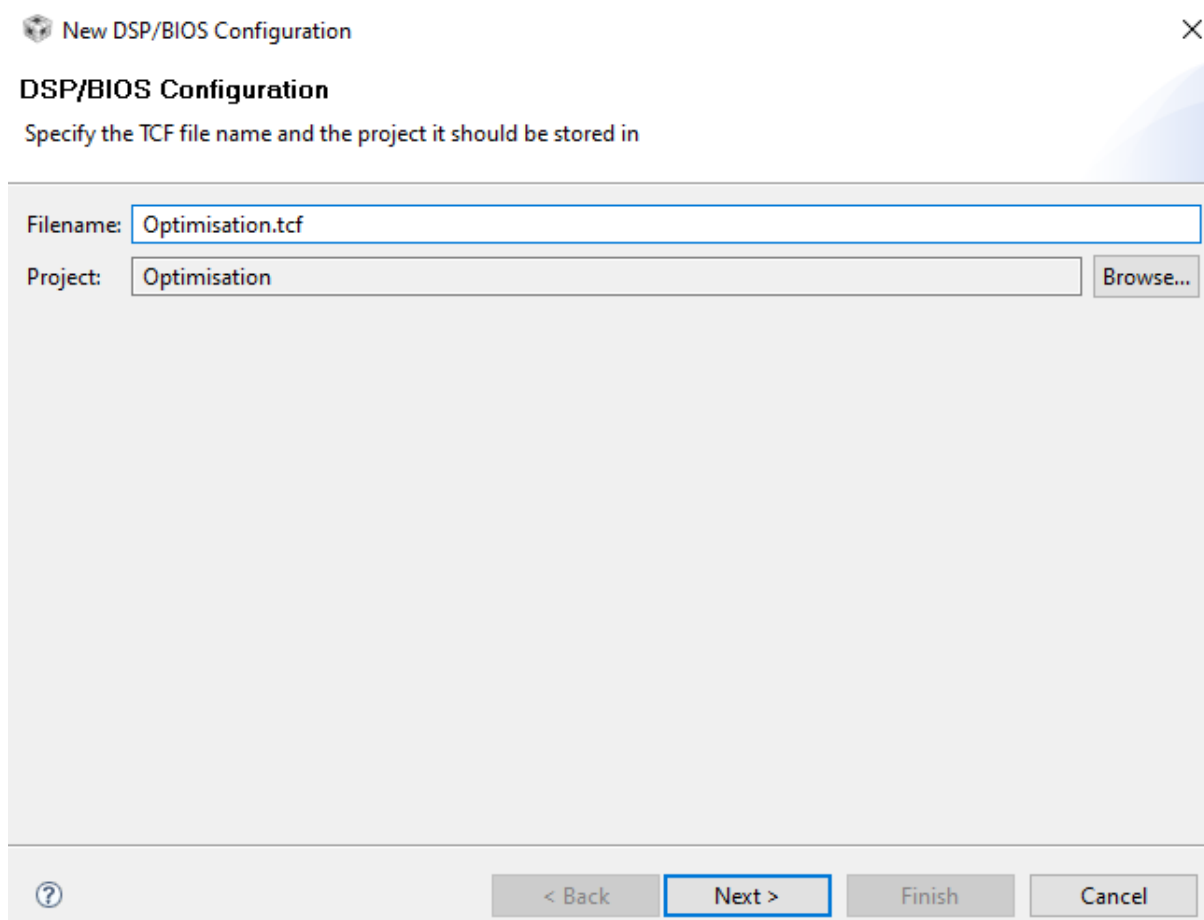
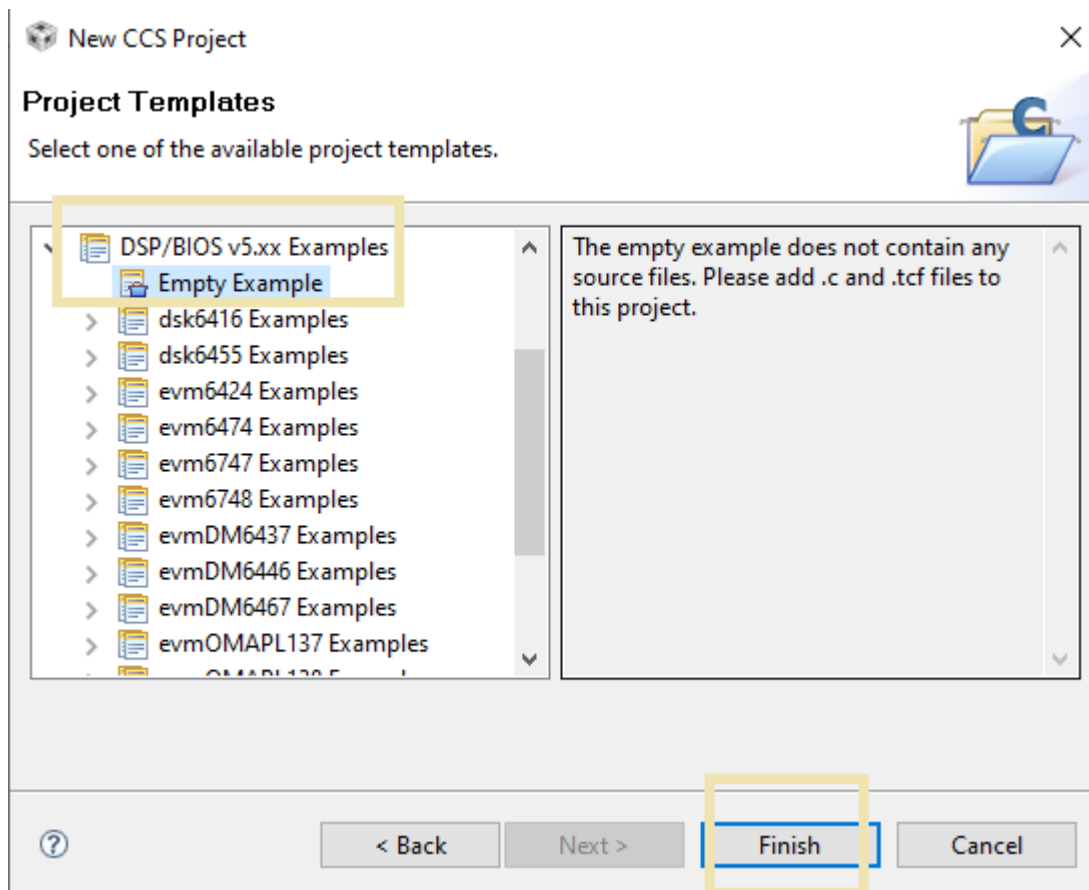


< Back

Next >

Finish

Cancel



DSP/BIOS Configuration

Specify a platform

Platform Location:

Filter Platforms:

Name	Description
ti.platforms.sim64xx	Sim64xx (600 Mhz) using 6416 device
ti.platforms.dsk6416	Dsk6416 (/20 Mhz) with 16 Mbyte SDRAM

Configuration Tool - [C:\Users\detatec\Documents\workspace\tp1\Optimisation\gconf\Optimisation\Optimisation.tcf]

File Edit View Object Help

Estimated Data Size: 2976 Est. Min. Stack Size (MAUs)

- System
 - Global Settings
 - MEM - Memory Section Manager
 - IRAM
 - SDRAM
 - BUF - Buffer Manager
 - POOL - Allocator Manager
 - SYS - System Settings
 - HOOK - Module Hook Manager
- Instrumentation
- Scheduling
- Synchronization
- Input/Output

MEM - Memory Section Manager Properties

General BIOS Data BIOS Code Compiler Sections Load Address

☐ Reuse Startup Code SpaceArgument Buffer Size: Stack Size (MAUs): ☐ No Dynamic Memory HeapsSegment For DSP/BIOS Objects: Segment For malloc() / free(): ☐ Enable Memory Protection Controller module

OK

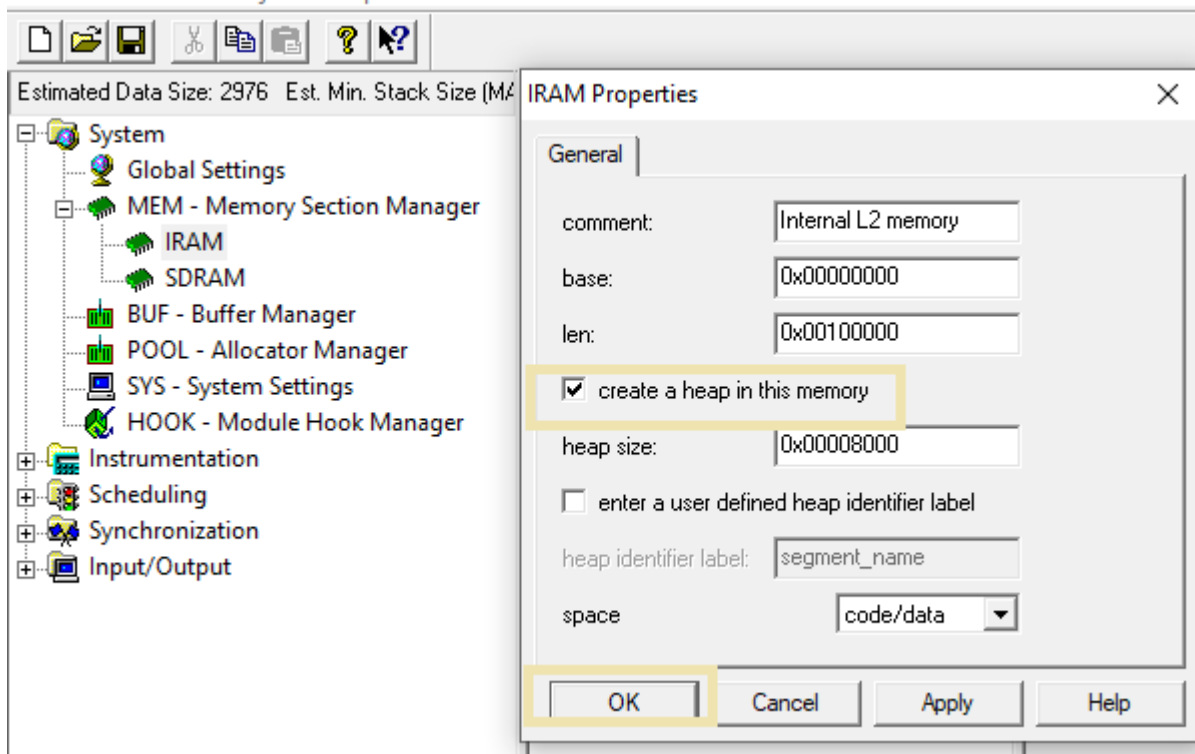
Cancel

Apply

Help

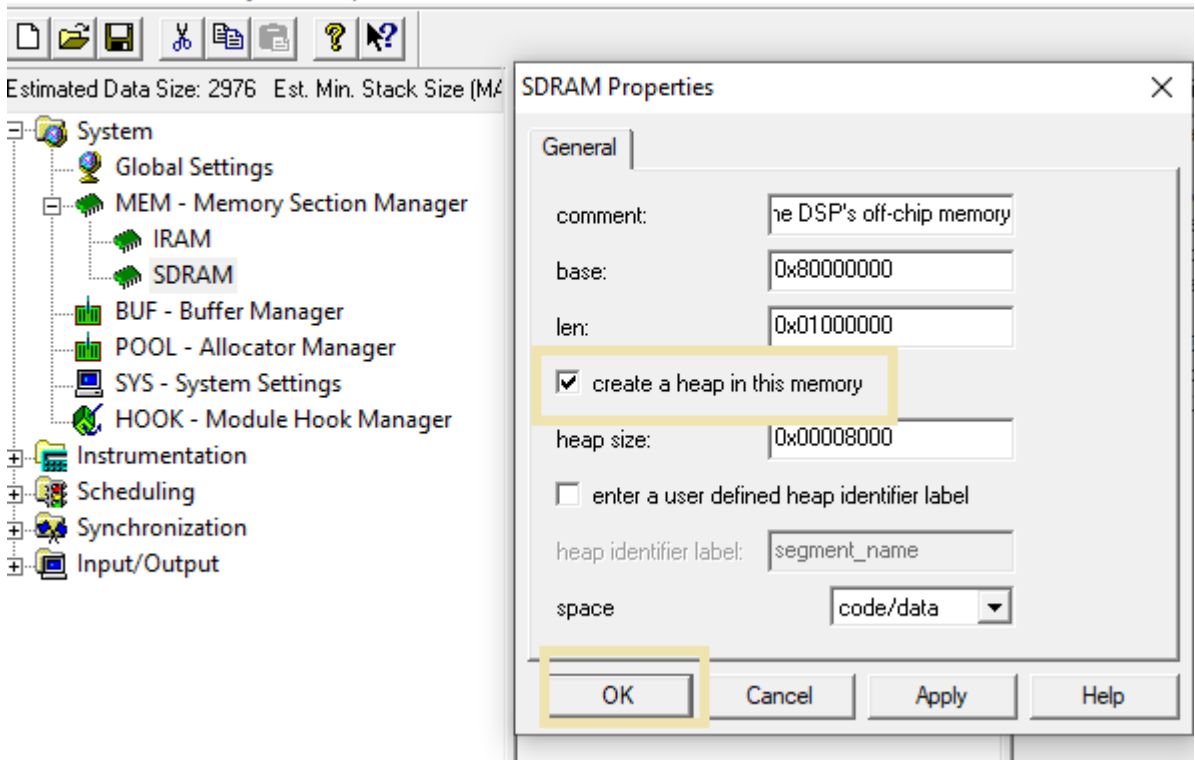
or Help, press F1

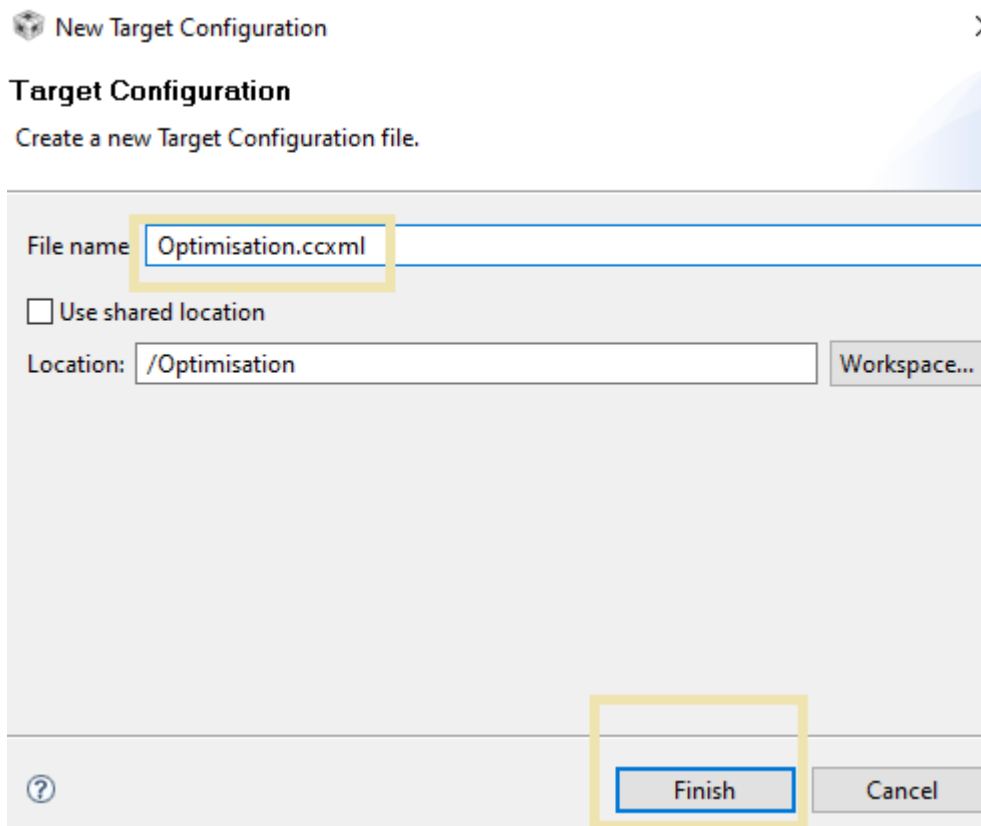
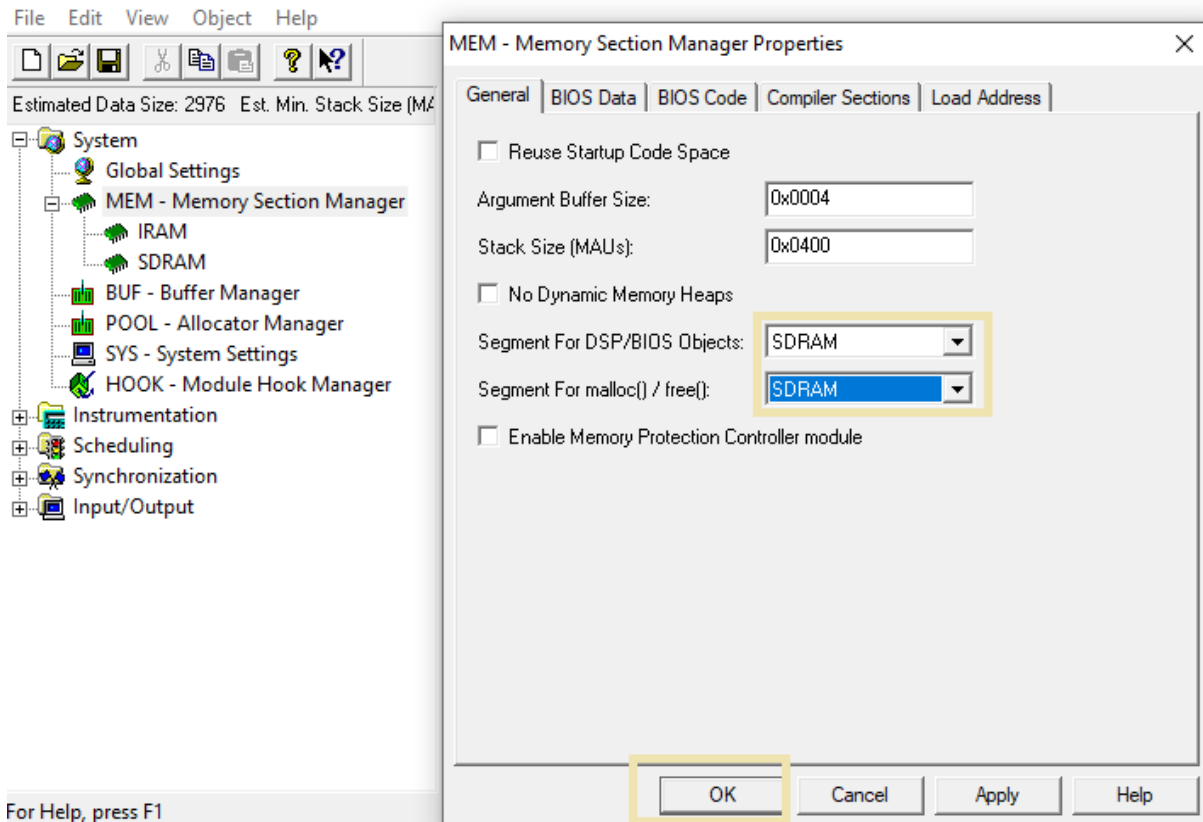
[Etc.]



Configuration Tool - [C:\Users\detatec\Documents\workspace\tp1\Optimisation\gconf\Optimisation\Optir

File Edit View Object Help





Optimisation.ccxml

Basic

General Setup

This section describes the general configuration about the target.

Connection
 Texas Instruments Simulator

Device
 type filter text

☐ C6416 Device Cycle Accurate Simulator, Big Endian
 ☒ C6416 Device Cycle Accurate Simulator, Little Endian
 ☐ C6421 Device Cycle Accurate Simulator, Big Endian

Advanced Setup

[Target Configuration:](#)

Save Configuration

Save

Configuration Settings

Tool Settings
Build Settings
Build Steps
Error Parsers
Binary Parser
Environment
Macros

Tool-chain Settings

C6000 Compiler

Basic Options
 Symbolic Debug Options
 Language Options
 Language Options: MISRA R
 Parser Preprocessing Option
 Predefined Symbols
 Include Options

Pre-define NAME (--define, -D)

CHIP_6416

Undefine NAME (--undefine, -U)

Configuration Settings

Tool Settings
Build Settings
Build Steps
Error Parsers
Binary Parser
Environment
Macros

Tool-chain Settings

C6000 Compiler

Basic Options
 Symbolic Debug Options
 Language Options
 Language Options: MISRA R
 Parser Preprocessing Option
 Predefined Symbols
 Include Options

Add dir to #include search path (--include_path, -I)

"C:\ccs4\setup_CCS_4_Platinum\cs\include"

"C:\ccs4\setup_CCS_4_Platinum\dsk6416\include"

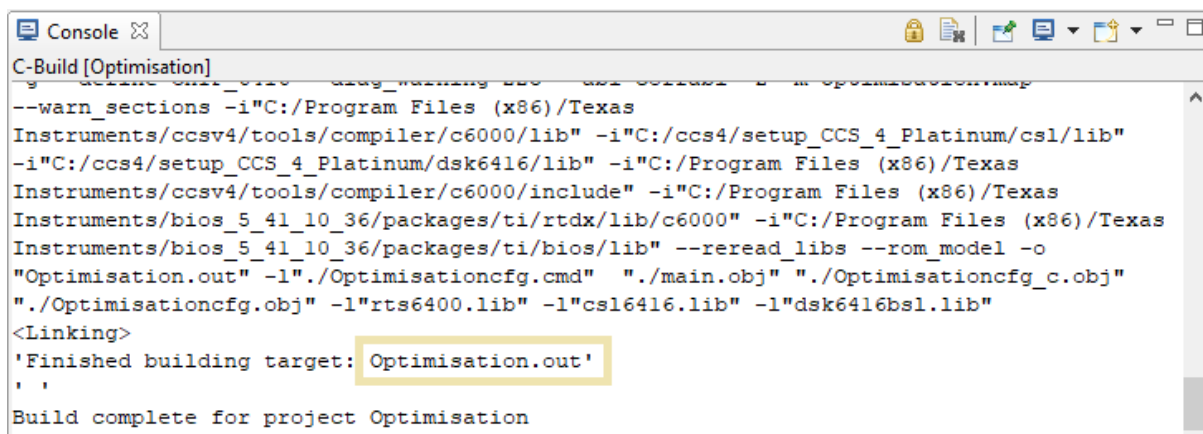
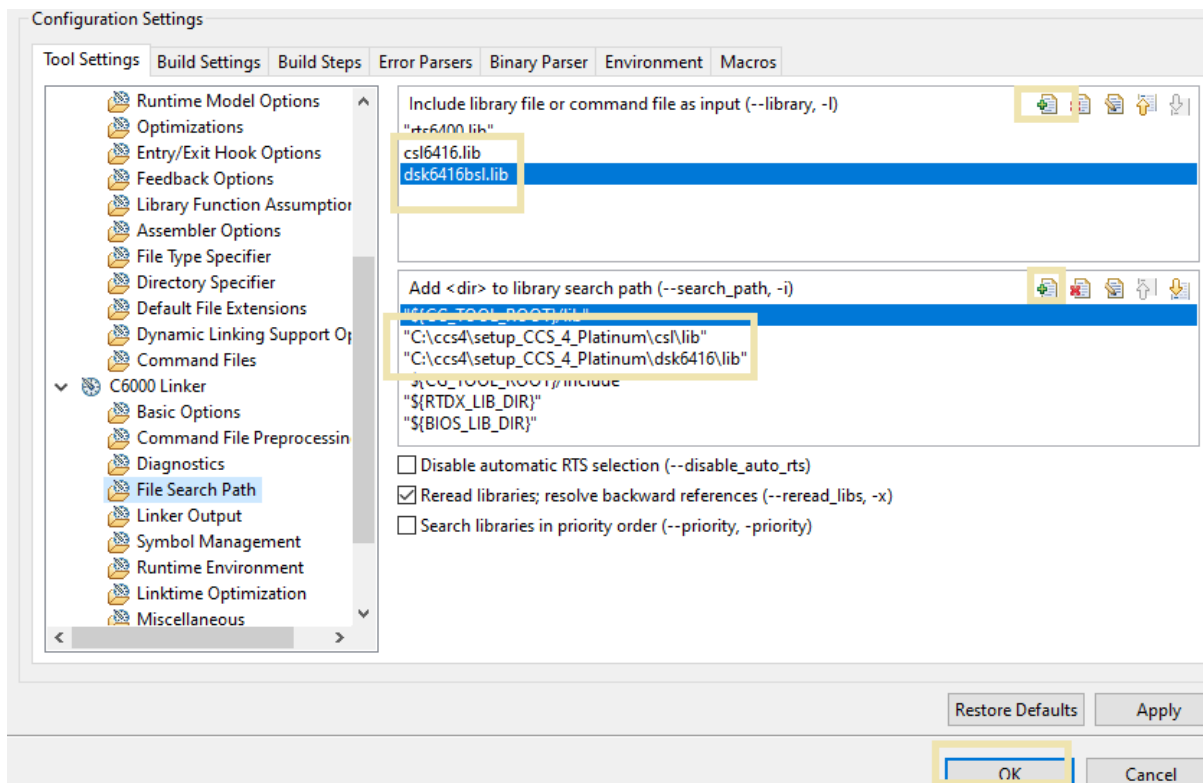
"\${COMPILED_OUTPUT_DIR}"

"\${BIOS_INCLUDE_DIR}"

"\${RTDX_INCLUDE_DIR}"

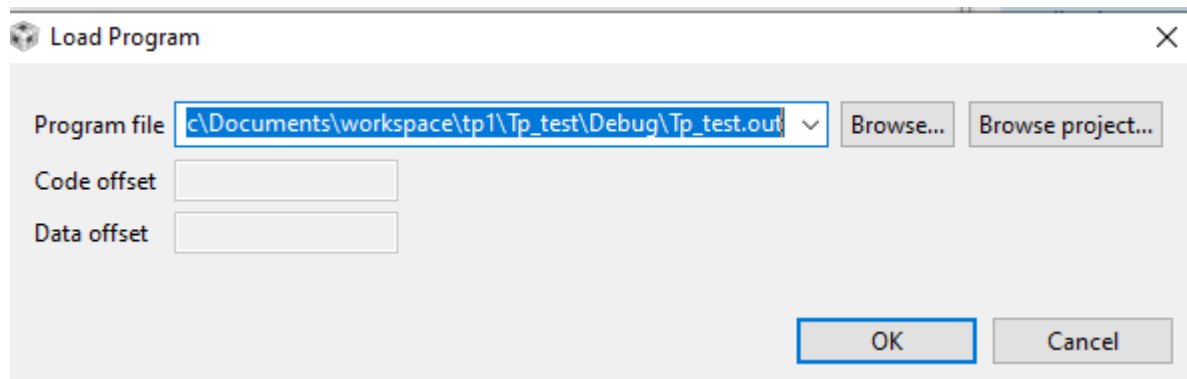
Specify a preinclude file (--preinclude)

20

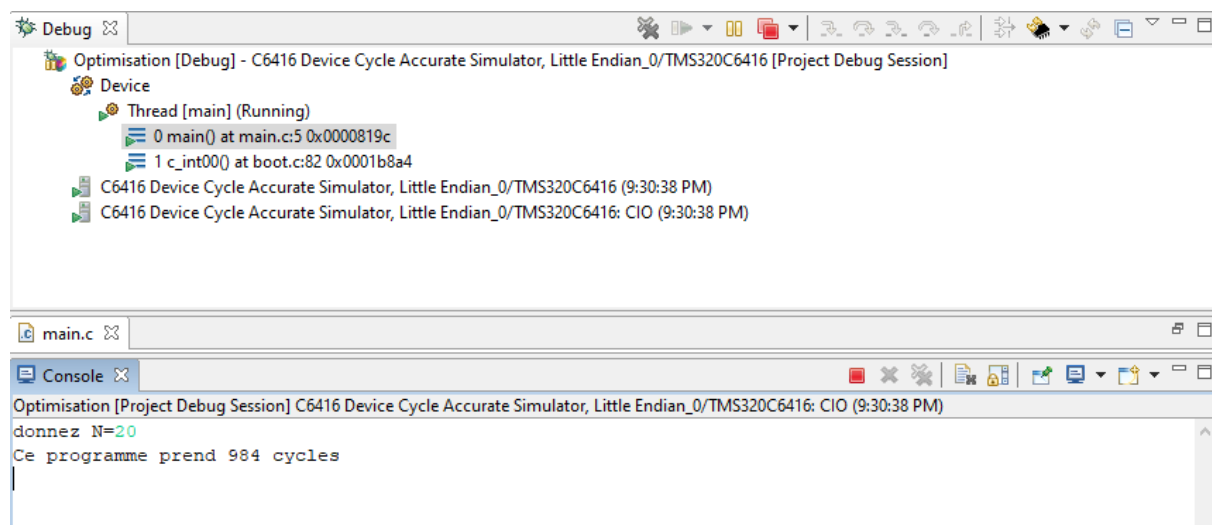


Pour exécuter le programme, j'ai fait les étapes suivantes :

- ✓ Choisir « **Target→Launch TI Debugger** ». Code composer ouvrira une page avec la perspective du mode «Debug».
- ✓ Choisir « **Target→Load Program...** » pour transférer le programme compilé vers le simulateur.



Le résultat :



TP3

1. Créer un projet

Lors de l'ouverture de CCS un écran «Welcome to Code Composer Studio v4» s'affiche qui introduit à un tutoriel de départ, des exemples et des informations sur les composants.

Le processus de développement de code de CCS commence 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) aux fichiers d'en-tête (*.h), au fichier de commande pour l'éditeur de liens (*.cmd), et aux fichiers de bibliothèque (*.lib). Le projet permet aussi de spécifier les paramètres du compilateur, de l'assembleur et de l'éditeur de liens afin de produire le fichier exécutable.

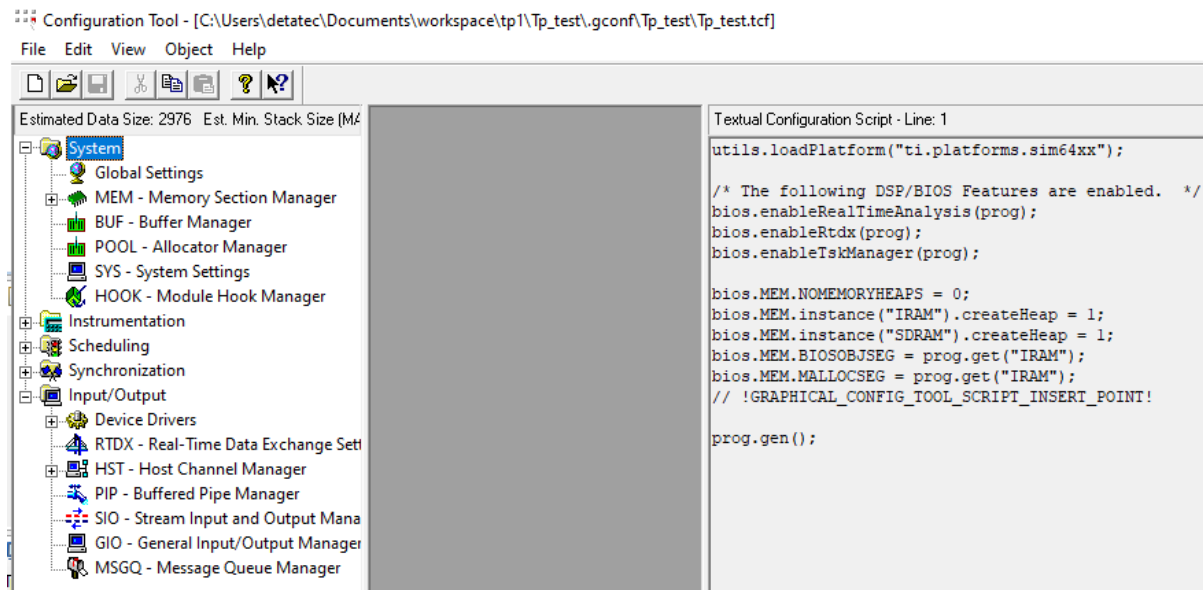
- Pour créer un projet, j'ai choisi l'élément de menu « File →New→CCS Project » de la barre menu. Dans celle-ci, j'ai introduit un nom de projet dans le champ « **Project name** » (Tp_test).
- J'ai poursuivi avec le type du projet en sélectionnant : « **Project Type** : C6000».
- J'ai sélectionné dans «**Device Variant**: Generic C64xx Device» et «**Runtime Support Library** : RTS6400.lib».
- J'ai sélectionné aussi dans la fenêtre «**Project Templates**» l'item « DSP/BIOS v5.xx Examples / Empty Example ».
- Pour la mise au point, j'ai ajouté un fichier source en C que j'ai appelé « main.c ».

Bien que CCS fournisse des options par défaut, ces options peuvent être changées en choisissant « **Project→Properties** ».

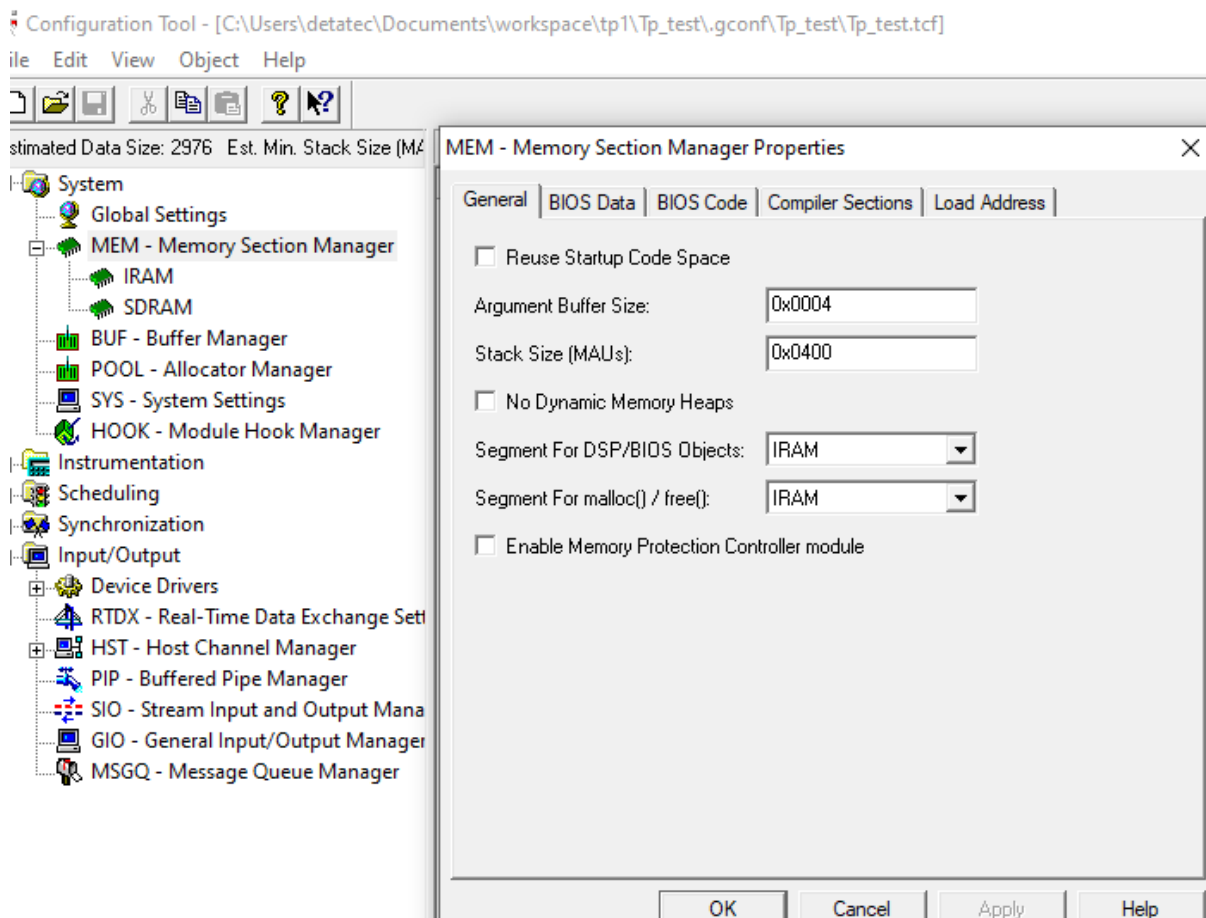
2. Ajout des fichiers au projet

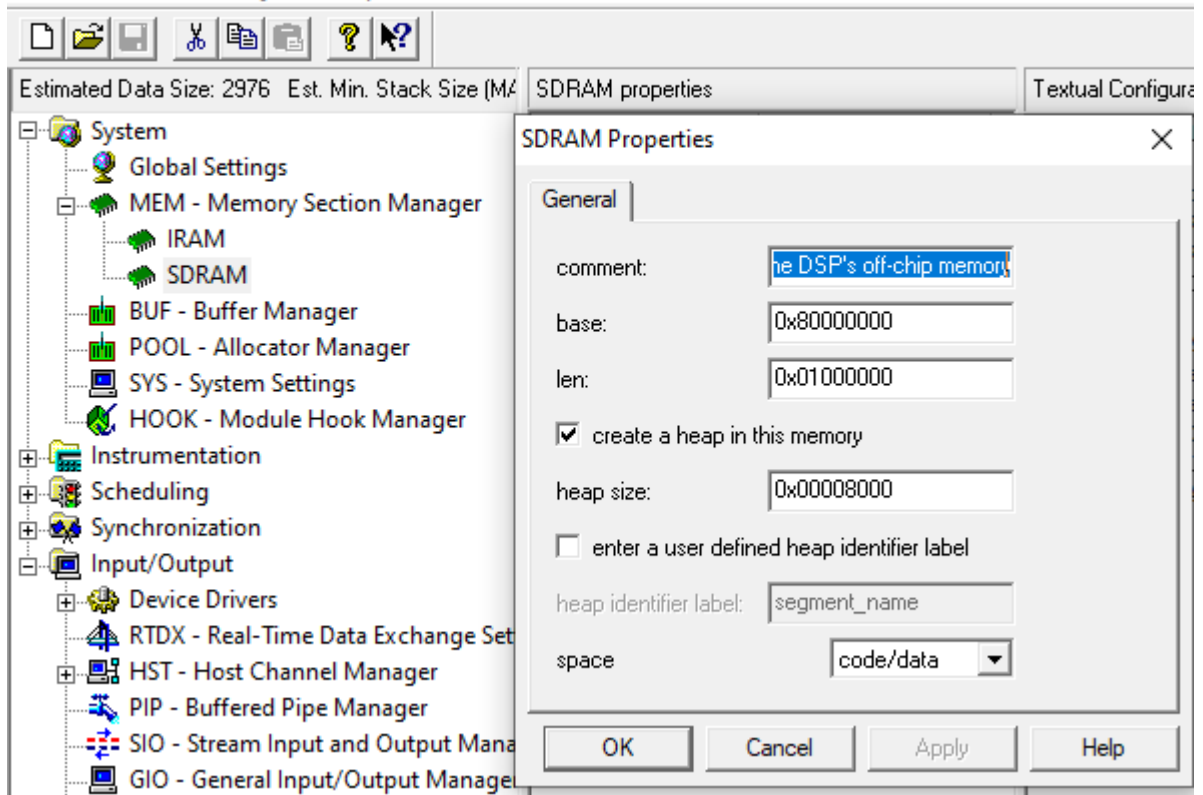
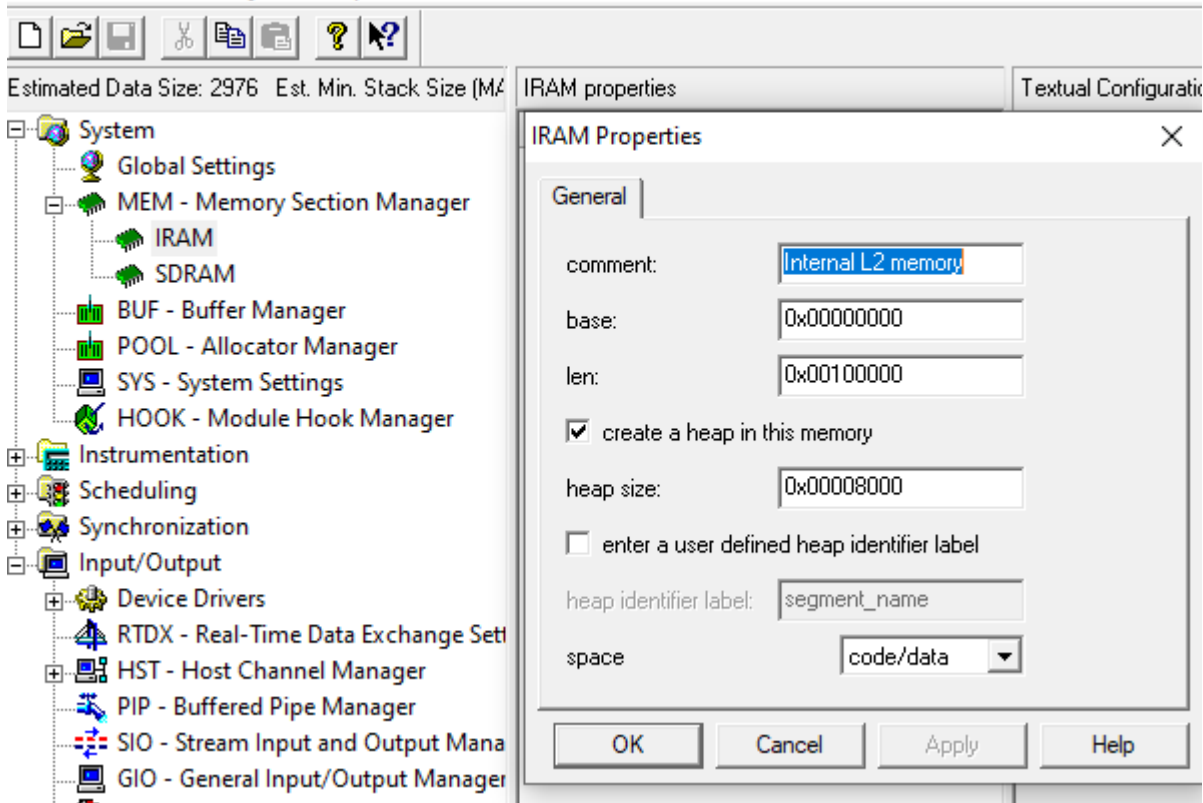
Pour ajouter le fichier de configuration de DSP/BIOS, j'ai sélectionné "File New DSP/BIOS Configuration File" et pour spécifier la plateforme DSP à utiliser. J'ai choisi la plateforme simulateur 6416 (ti.platforms.sim64xx).

Après ces étapes le fichier de configuration s'ouvre. Ce fichier permet de faire une gestion des mémoires et de modifier les paramètres de RTDX.



Dans cette étape, j'ai fait quelques modifications pour configurer les différentes sections mémoires (sur MEM-Memory Section Manager).





3. Le suivi de l'exécution (mode Debug)

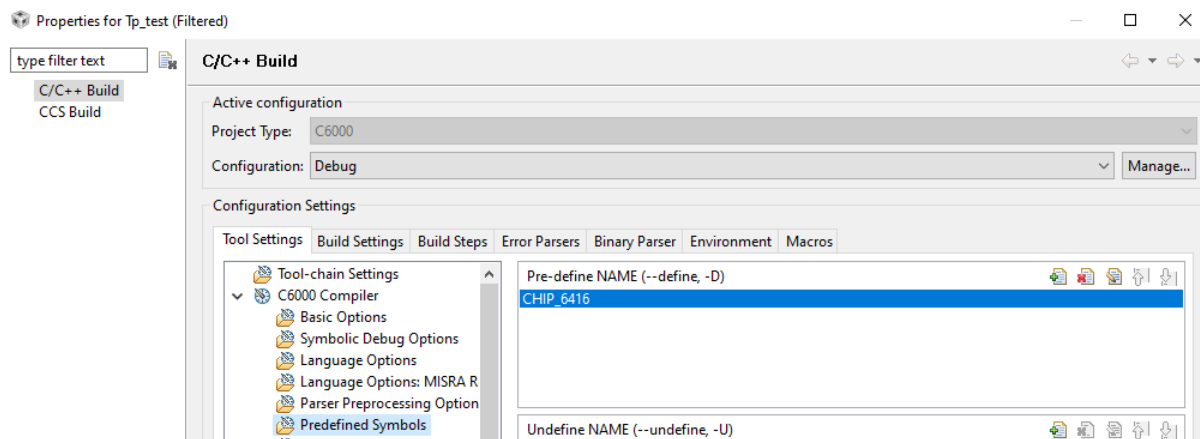
Une fois que le processus de construction est complété sans aucune erreur, le programme peut être chargé et exécuté sur le simulateur.

Dans le cas d'un simulateur, il suffit de choisir la configuration suivante :

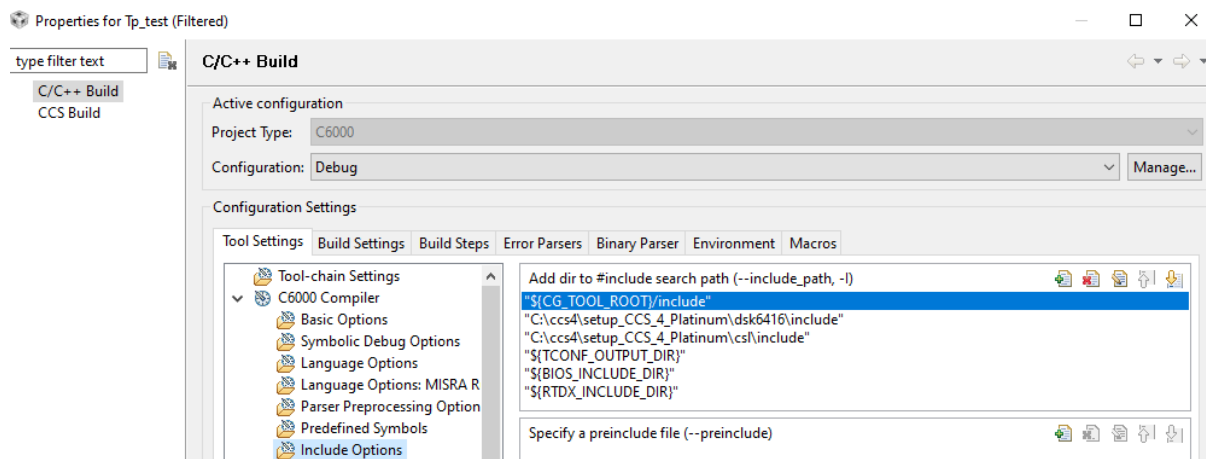
Connection : Texas Instruments Simulator

Device : C6416 Device Cycle Accurate Simulator, Little Endian dans la fenêtre «Device» et terminer en sauvegardant la configuration avec «Save».

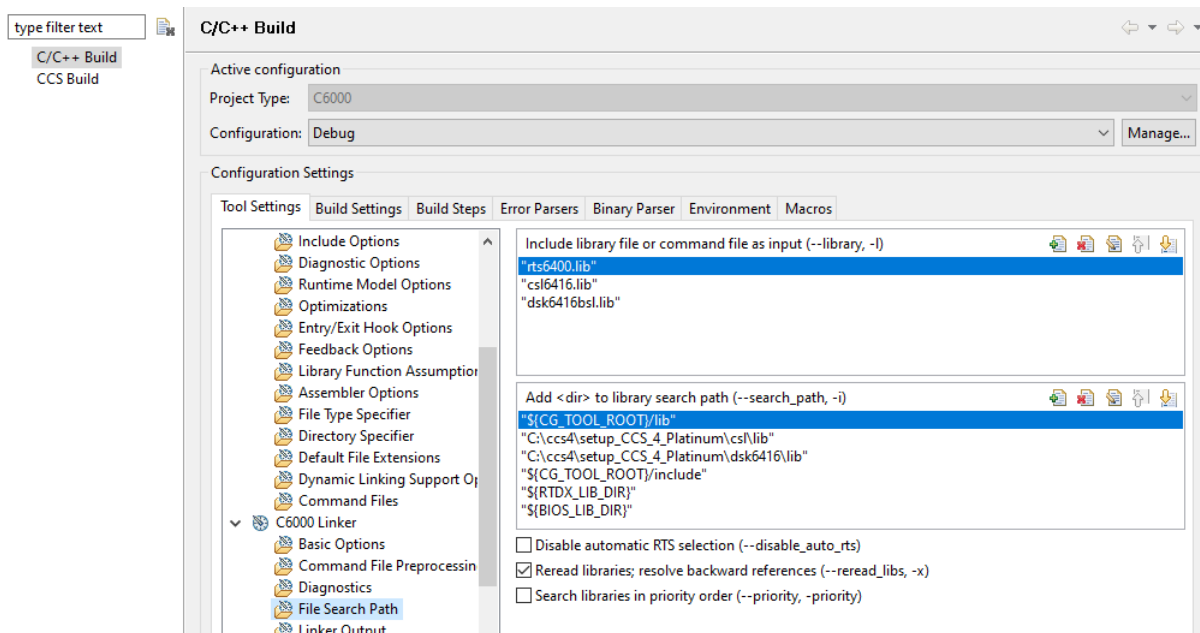
Pour compléter notre configuration, j'ai cliqué sur le projet et j'ai choisi "**Build properties**". Il faut faire quelques modifications dans "Predefined Symbols, Include Options et file search path".



Avant de commencer la compilation du projet, Il faut ajouter les chemins des bibliothèques nécessaires s'ils existent.



Même chose pour les bibliothèques ".lib" utilisées. J'ai sélectionné *FileSearch Path* dans la fenêtre de "**Build Properties**" et j'ai ajouté le nom de la bibliothèque en donnant le chemin du répertoire contenant le fichier .lib utilisé.



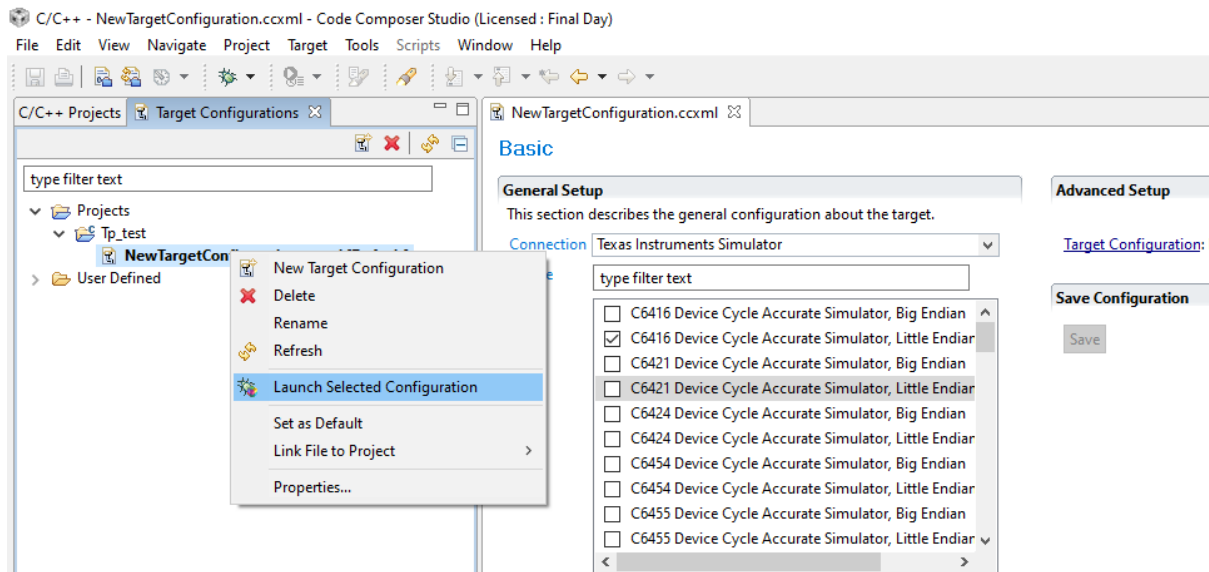
4. Compilation du projet

Après avoir ajouté tous les fichiers du projet (le main et l'image "output.jpg" dans le dossier Debug). Je peux maintenant construire notre projet et de créer un fichier exécutable pour le DSP-cible. J'ai choisit l'élément de menu « **Project→Build Active Project** ».

Cette fonction permet au CCS de compiler, assembler, et joindre tous les fichiers dans le projet. Si le processus de construction est complété sans erreur, le fichier exécutable «Tp_test.out» est produit.

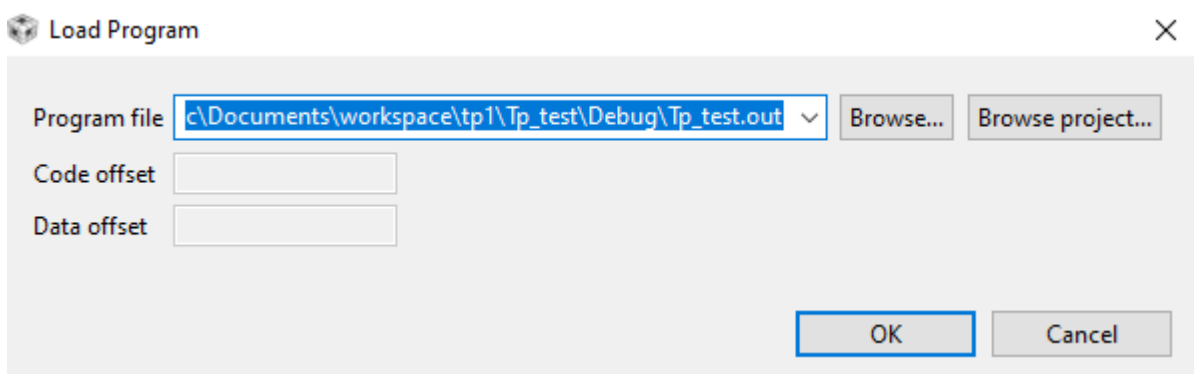
Il est également possible de compléter cette opération par incréments; c'est-à-dire en recompilant ou en assemblant seulement des fichiers changés depuis la dernière construction, en choisissant l'élément de menu « **Project→Rebuild Active Project** ».

Dans cette étape on doit établir une connexion entre le Code Composer et la plateforme cible "Target. Dans notre cas c'est le simulateur donc j'ai Sélectionné le menu **View Target configurations**. Pour Ouvrir le fichier de configuration, j' ai cliqué avec le bouton droit de la souris sur et target et j'ai sélectionné "**Launch Selected Configuration**".

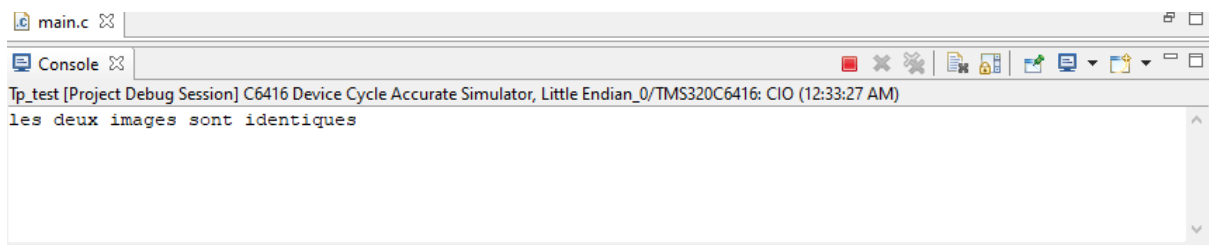


Pour exécuter le programme, j'ai fait les étapes suivantes :

- ✓ Choisir « **Target→Launch TI Debugger** ». Code composer ouvrira une page avec la perspective du mode «Debug».
- ✓ Choisir « **Target→Load Program...** » pour transférer le programme compilé vers le simulateur.

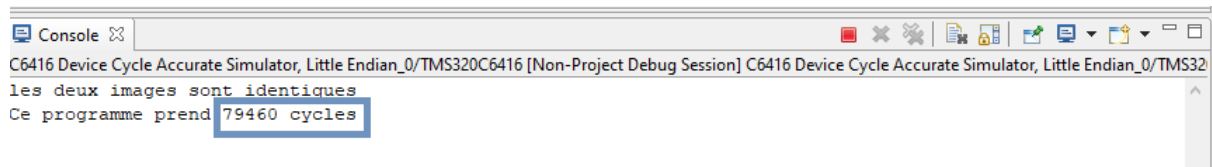


Pour terminer l'exécution du programme, j'ai cliqué sur l'élément « **Target→Run** ». Et voilà le resultat:



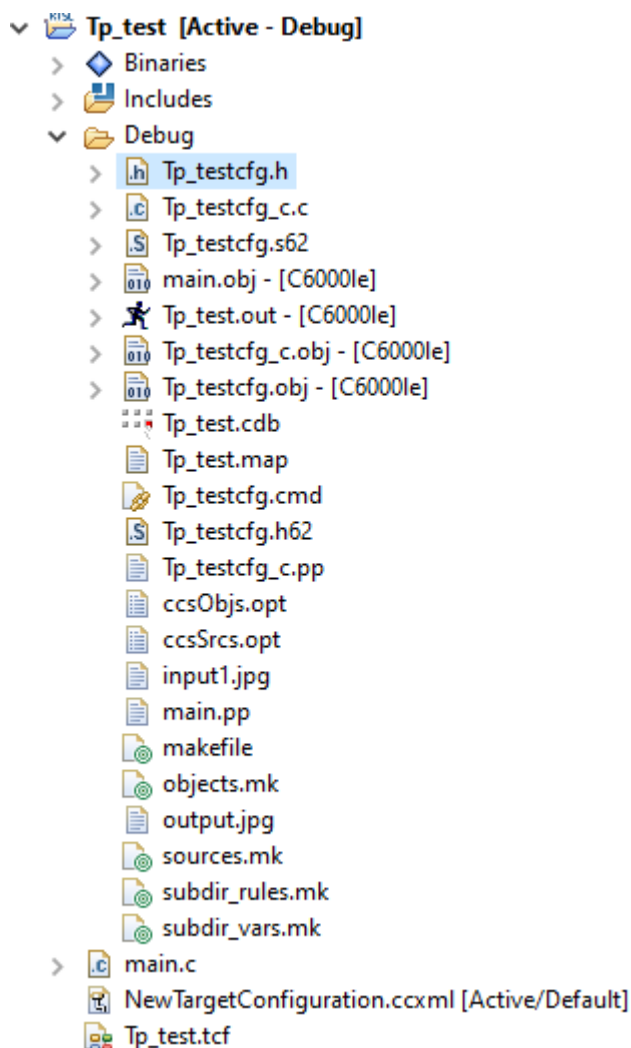
Lorsque on a modifié le *main.c* pour calculer le nombre de cycle, on a obtenu le résultat ci-dessous.

Comme l'indique au console notre programme a été exécuté avec succès et a indiqué que l'image Identique à celle de l'entrée « input1 ». Le programme faire 79460 cycles pour obtenir ce résultat. On a généré un programme exécutable mais avec un grand nombre d'opération.

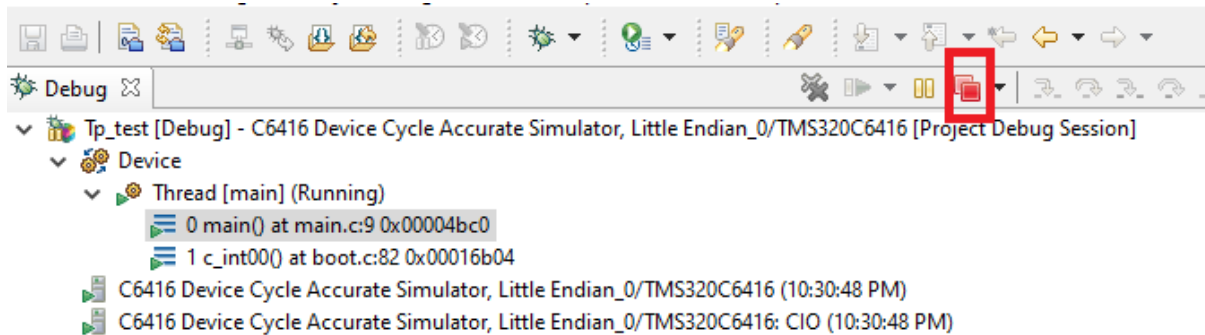


Le DSP / BIOS est configuré à l'aide de l'outil de configuration DSP / BIOS. Les paramètres de cet exemple sont stockés dans une configuration fichier appelé *Tp_test.cdb*. Au moment de la compilation, Code Composer générera automatiquement les fichiers liés au DSP / BIOS en fonction de ces paramètres.

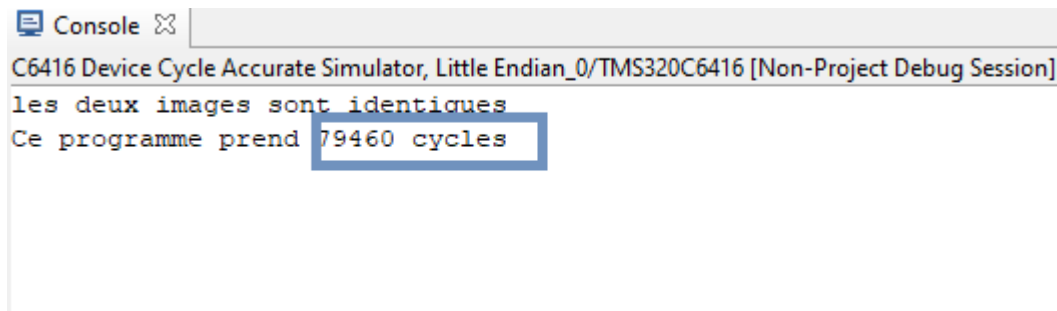
Un fichier d'en-tête appelé *Tp_testcfg.h* contient les résultats de l'auto génération et doit être inclus pour un fonctionnement correct. Le nom du fichier provient de *Tp_test.cdb* et l'ajout de *cfg.h*.



Pour revenir en perspective d'édition C/C++, on peut terminer la session «Debug» en choisissant «**Target→Terminate All**» ou encore en cliquant sur le carré rouge du menu.

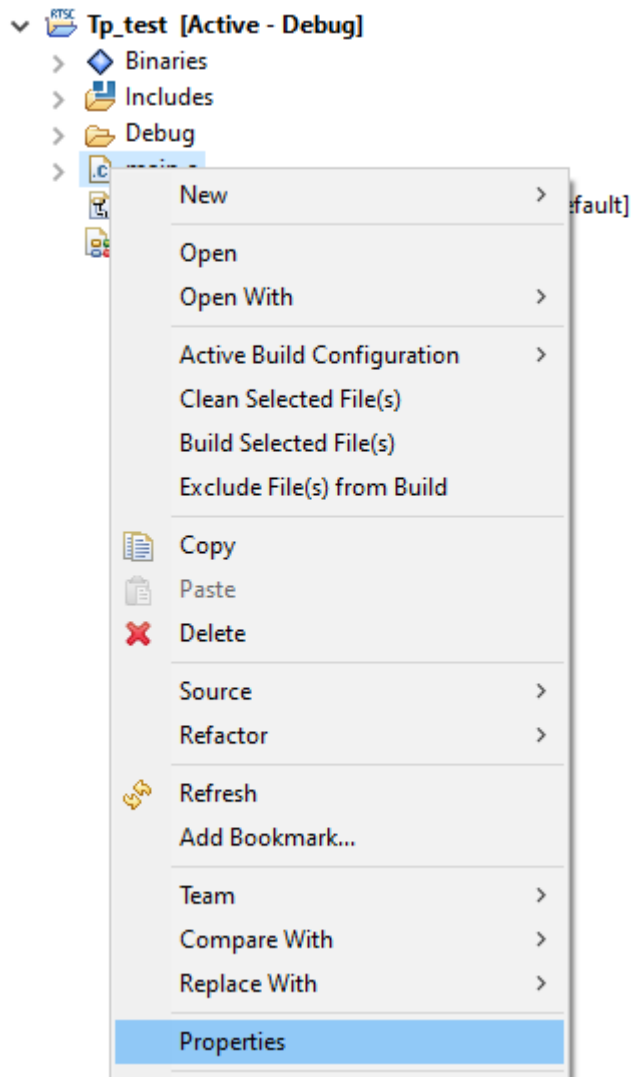


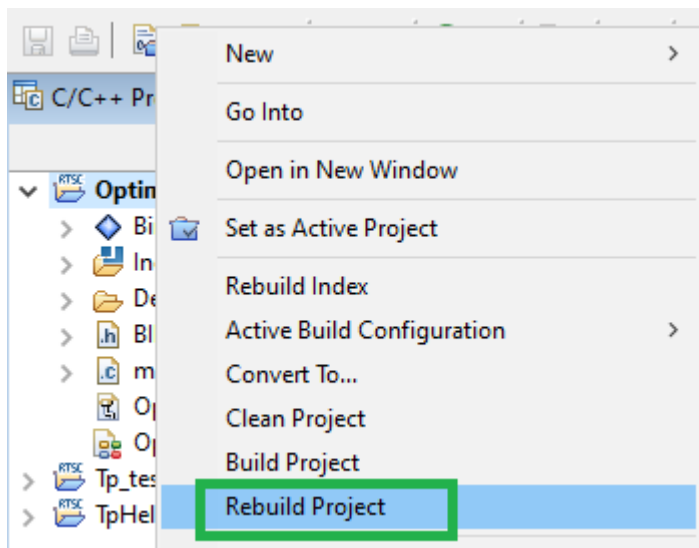
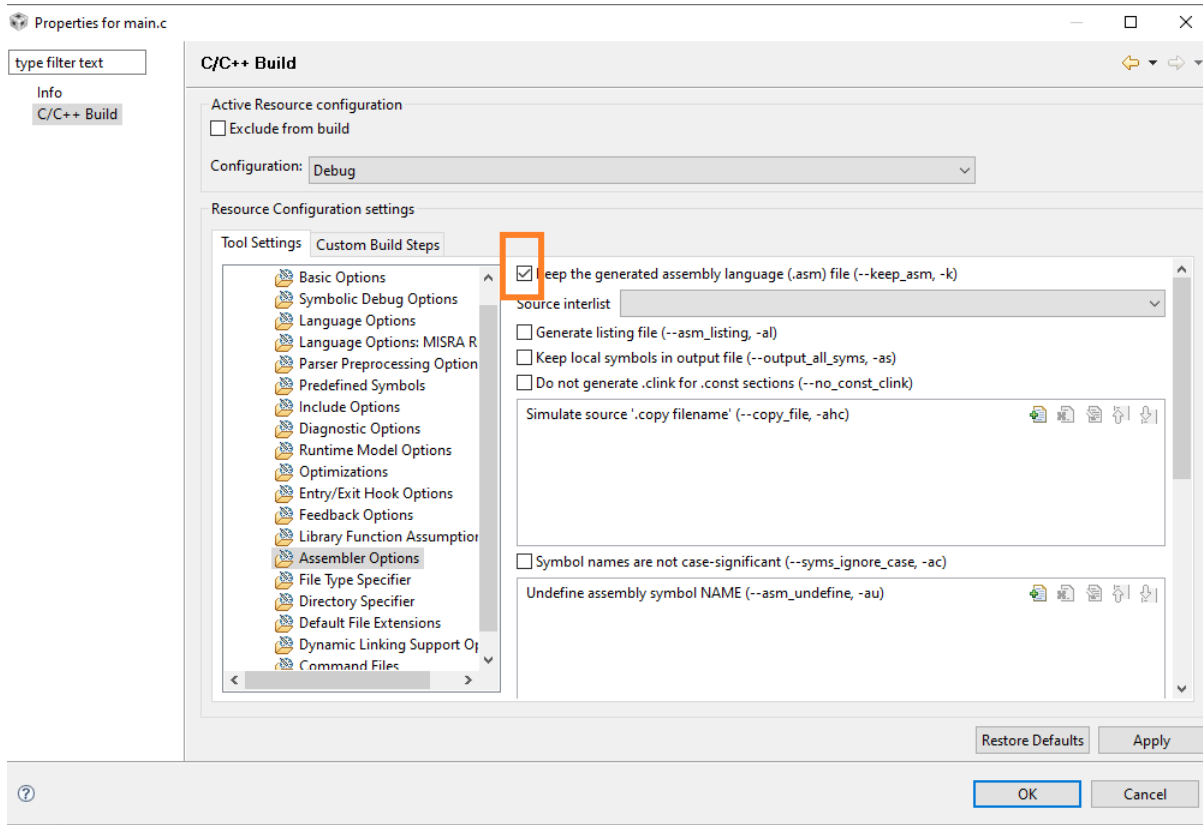
Pour calculer le nombre de cycle de l'exécution de notre programme on peut ajouter le fichier BIB.h à notre projet et on modifie le code de main.c. Le resultat est ci-dessous.



TP4

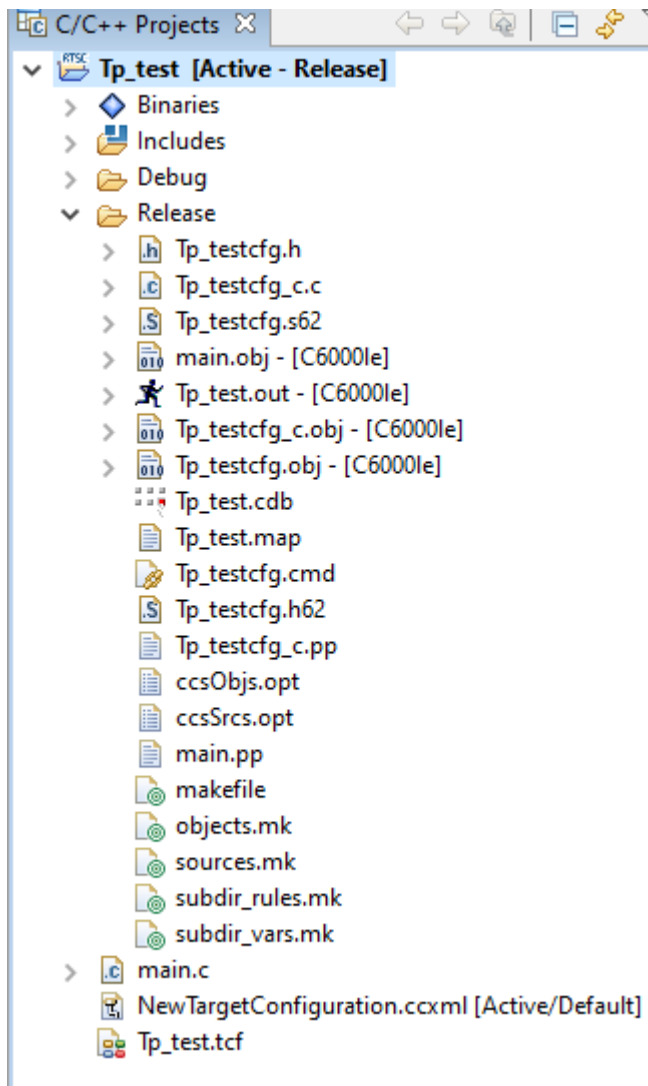
Dans le 4^{ème} TP, on va utiliser les 2 anciens TP pour avoir un résultat plus optimisé et pour cela on va changer la configuration vers « **release** ». Donc on va suivre toutes les étapes ci-dessous:



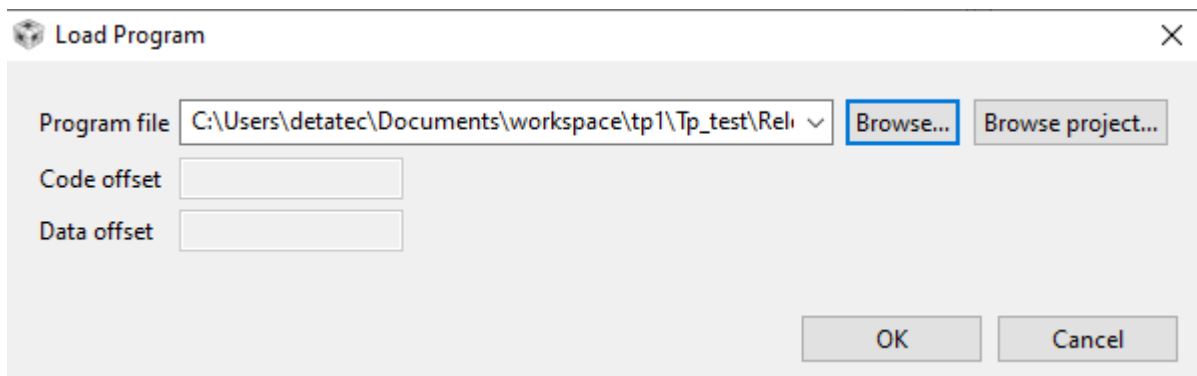



```
Console [Tp_test]
C-Build [Tp_test]
--preproc_dependency="Tp_testcfg_c.pp" "Tp_testcfg_c.c"
'Finished building: Tp_testcfg_c.c'
'
'
'Building file: ../main.c'
'Invoking: Compiler'
"C:/Program Files (x86)/Texas Instruments/ccsv4/tools/compiler/c6000/bin/cl6x" -mv6400 -g --define=CHIP_6416
--include_path="C:/Program Files (x86)/Texas Instruments/ccsv4/tools/compiler/c6000/include"
--include_path="C:/ccs4/setup_CCS_4_Platinum/dsk6416/include" --include_path="C:/ccs4/setup_CCS_4_Platinum/csl/include"
--include_path="C:/Users/detatec/Documents/workspace/tpl/Tp_test/Debug" --include_path="C:/Program Files (x86)/Texas
Instruments/bios_5_41_10_36/packages/ti/bios/include" --include_path="C:/Program Files (x86)/Texas
Instruments/bios_5_41_10_36/packages/ti/rtdx/include/c6000" --diag_warning=225 --abi=coffabi -k --preproc_with_compile
--preproc_dependency="main.pp" "../main.c"
'Finished building: ../main.c'
'
'
'Building target: Tp_test.out'
'Invoking: Linker'
"C:/Program Files (x86)/Texas Instruments/ccsv4/tools/compiler/c6000/bin/cl6x" -mv6400 -g --define=CHIP_6416
--diag_warning=225 --abi=coffabi -z -m"Tp_test.map" --warn_sections -i"C:/Program Files (x86)/Texas
Instruments/ccsv4/tools/compiler/c6000/lib" -i"C:/ccs4/setup_CCS_4_Platinum/csl/lib"
-i"C:/ccs4/setup_CCS_4_Platinum/dsk6416/lib" -i"C:/Program Files (x86)/Texas
Instruments/ccsv4/tools/compiler/c6000/include" -i"C:/Program Files (x86)/Texas
Instruments/bios_5_41_10_36/packages/ti/rtdx/lib/c6000" -i"C:/Program Files (x86)/Texas
Instruments/bios_5_41_10_36/packages/ti/bios/lib" --reread_libs --rom_model -o "Tp_test.out" -l"./Tp_testcfg.cmd"
"./main.obj" "../Tp_testcfg_c.obj" "../Tp_testcfg.obj" -l"rts6400.lib" -l"csl6416.lib" -l"dsk6416bsl.lib"
<Linking>
'Finished building target: Tp_test.out'
'
'
Build complete for project Tp_test
```

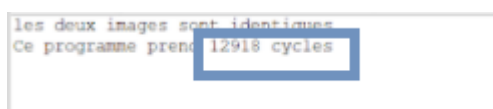
On remarque qu'un nouveau dossier a été générer.



Dans le “**Load program**” on doit choisir notre *Release.out* pour faire l’exécution.



Le résultat:



Dans le mode Release les exécutions ont été fait en mode parallèle. C'est pour cela le nombre de cycle a été diminuer. On peut dire qu'on a minimiser le programme "image processing" avec le changement de mode de configuration vers « Release ».