Université Abou-Bakr Belkaïd Tlemcen **M1/TR842**
Faculté de Technologie
Département de Télécommunications

TP 1 – L'environnement de développement intégré Code Composer Studio (CCS)

Objectif

L'objectif de ce TP est de se familiariser avec l'environnement de développement intégré « Code Composer Studio » tout en écrivant, compilant et exécutant des programmes C et assembleur.

Introduction

CODE Composer StudioTM (CCS) est l'environnement de développement intégré (IDE) de Texas Instruments (TI) pour le développement de programmes d'application sur une grande variété de leurs DSP. Dans CCS, les outils d'édition, de génération de code et de débogage sont tous intégrés dans un environnement unifié. Vous pouvez sélectionner le DSP cible, ajuster les paramètres d'optimisation et définir les préférences de l'utilisateur comme vous le souhaitez.

Un programme d'application est développé sur la base du concept de projet, où les informations contenues dans le fichier de projet déterminent quels fichiers de code source sont utilisés et comment ils seront traités. Apprendre à utiliser Code Composer Studio est une étape nécessaire pour combler le fossé entre la théorie du DSP et le DSP en temps réel si vous envisagez d'utiliser les processeurs TI. CCS possède des capacités graphiques et prend en charge le débogage en temps réel. Il fournit un outil logiciel facile à utiliser pour construire et déboguer les programmes.

Le compilateur C compile un programme source C avec l'extension .c pour produire un fichier source assembleur avec comme extension .asm. L'assembleur assemble une source .asm pour produire un fichier objet en langage machine avec extension .obj. L'éditeur de liens combine des fichiers d'objets et des bibliothèques d'objets en entrée pour produire un fichier exécutable avec extension .out. Ce fichier exécutable représente un format de fichier d'objet commun lié (COFF). Ce fichier exécutable peut être chargé et exécuté directement sur le processeur comme le C6711. Pour créer un projet d'application, on peut «ajouter» les fichiers appropriés au projet. Les options du compilateur / éditeur de liens peuvent être facilement spécifiées. Un certain nombre de débogage les fonctionnalités sont disponibles, y compris la définition des points d'arrêt et la surveillance des variables, la visualisation mémoire, registres et code C et assemblage mixtes, représentation graphique des résultats et surveillance du temps d'exécution.

Lancement de CCS

Pendant le démarrage du programme, vous devriez voir un écran d'accueil CCS semblable à celui de la figure 1. Il peut y avoir une barre de progression au bas de l'écran d'accueil pour indiquer l'état d'avancement de l'initialisation du programme. Cette initialisation peut prendre plusieurs secondes selon votre système.

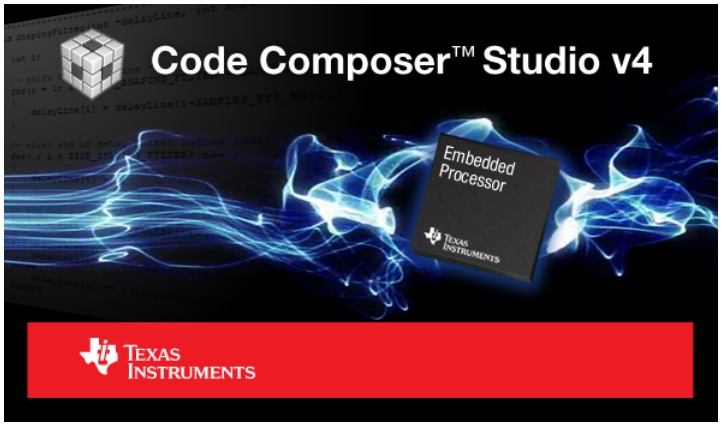

Figure 1 : L'écran d'ouverture de Code Composer Studio version 4.

CCS vous invite à sélectionner un espace de travail comme le montre la Figure 2.

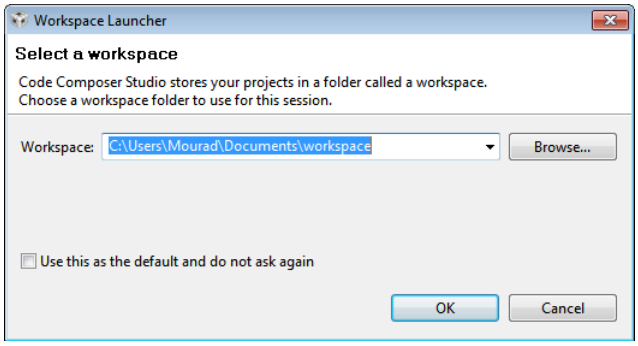

Figure 2 : Sélection de l'espace de travail

Lorsque l'initialisation est terminée, l'écran d'accueil devrait disparaître et la fenêtre principale du projet devrait apparaître à sa place. Cette fenêtre principale de projet devrait ressembler à la Figure 3.

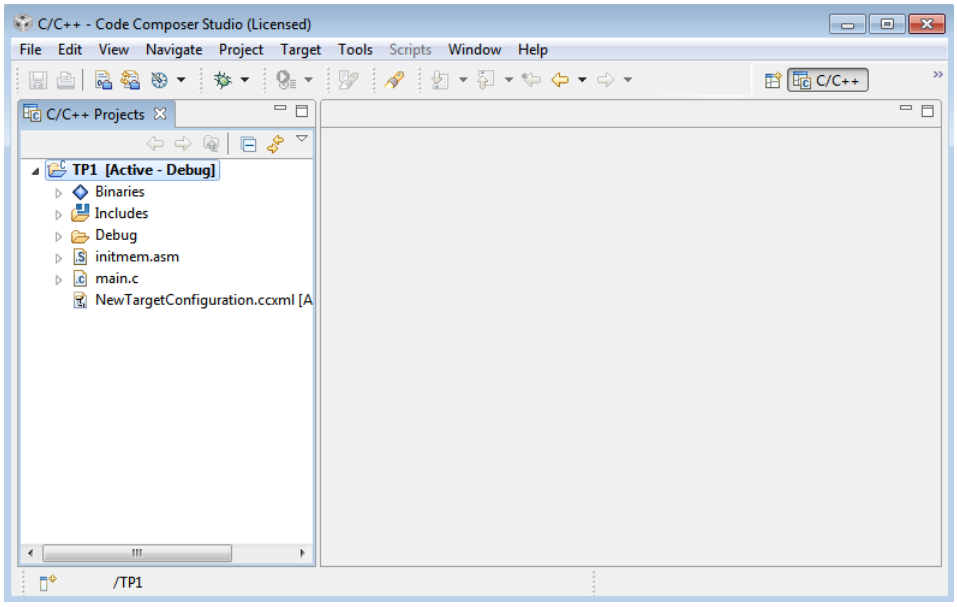

Figure 3 : Fenêtre d'accueil de CCS.

Il peut y avoir une barre de progression au bas de l'écran d'accueil pour indiquer l'état d'avancement de l'initialisation du programme. Cette initialisation peut prendre plusieurs secondes selon votre système.

Lorsque l'initialisation est terminée, l'écran d'accueil devrait disparaître et la fenêtre principale du projet devrait apparaître à sa place. Cette fenêtre de projet principale devrait ressembler à la Figure 3, qui montre un projet typique déjà chargé dans l'environnement CCS.

Création d'un projet

Voyons maintenant comment créer un projet CCS. Pour cela, sélectionnez "File > New > CCS Project " (Figure 4).

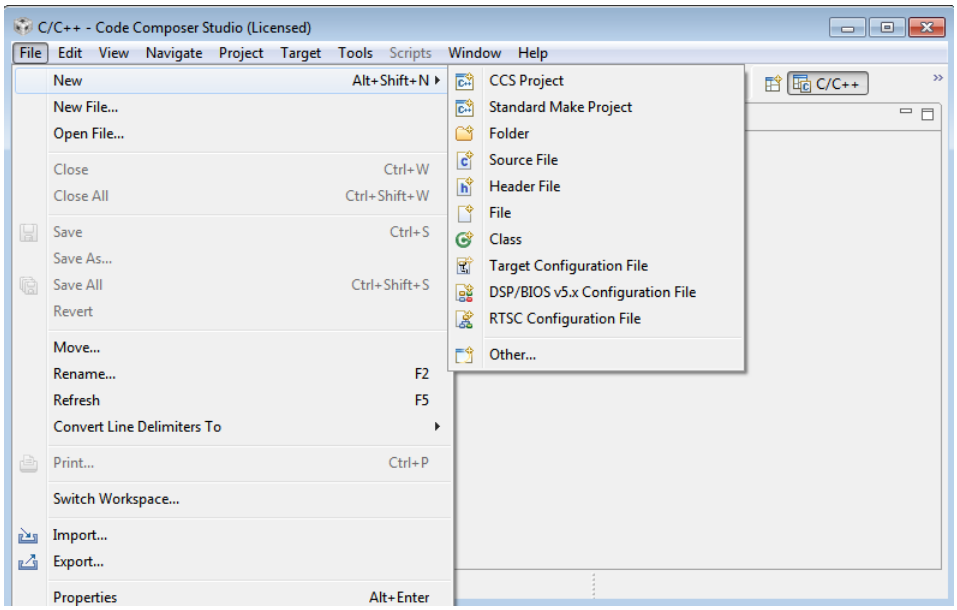

Figure 4 : Création d'un nouveau projet.

Dans la fenêtre montrée par la figure 5, nommez le projet "TP1_CCS", puis appuyez sur le bouton "Next".

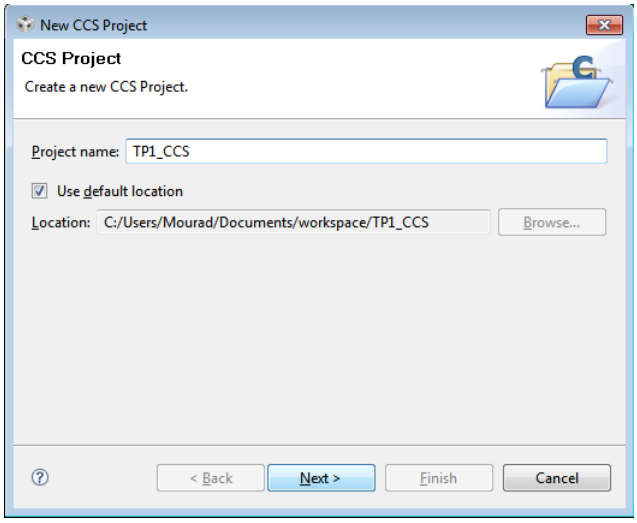

Figure 5 : Introduction du nom de projet.

Vous serez invité à entrer le type de projet (voir Figure 6).

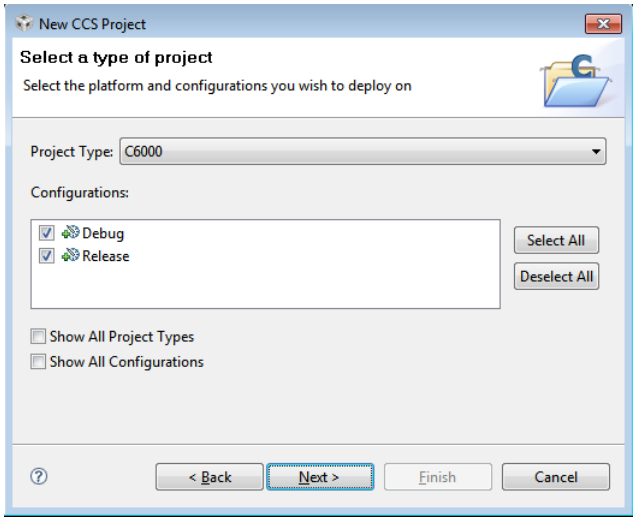

Figure 6 : Introduction du type de projet.

Le type de projet doit être "C6000".

Appuyez sur "Next"

Appuyez sur "Next"

Appuyez sur "Finish"

Le projet que vous venez de le créer apparaît dans le volet gauche de la fenêtre principale (voir Figure 7).

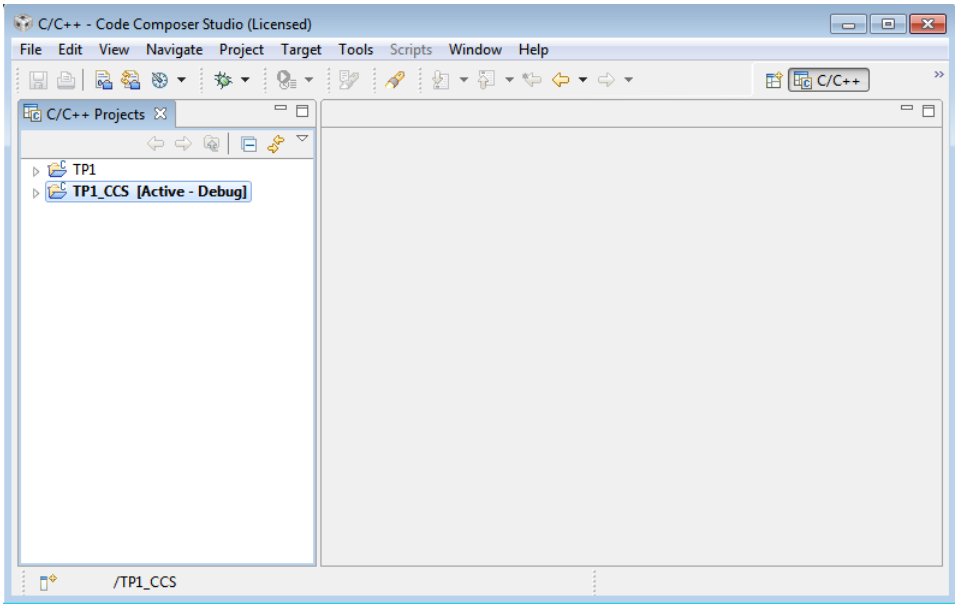

Figure 7 : Liste des projets C/C++.

Maintenant, on va créer un fichier source c dans notre projet TP1_CCS.

Sélectionnez le projet, cliquez sur le menu "File > New > Source File". Une fenêtre s'affiche en vous invitant à introduire le nom du fichier source. Tapez le nom du fichier comme le montre la Figure 8. Après validation, le nom du fichier s'ajoute au projet et apparaît dans le volet gauche.

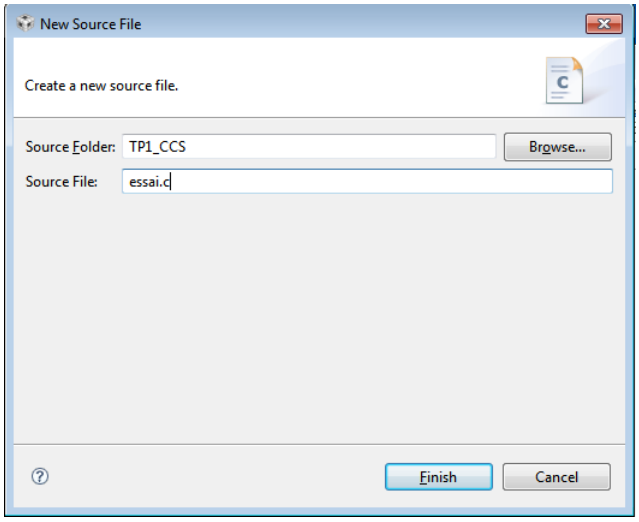

Figure 8 : Création d'un nouveau fichier source.

Cliquez sur le fichier et tapez le code figurant dans la fenêtre montrée par la Figure 9.

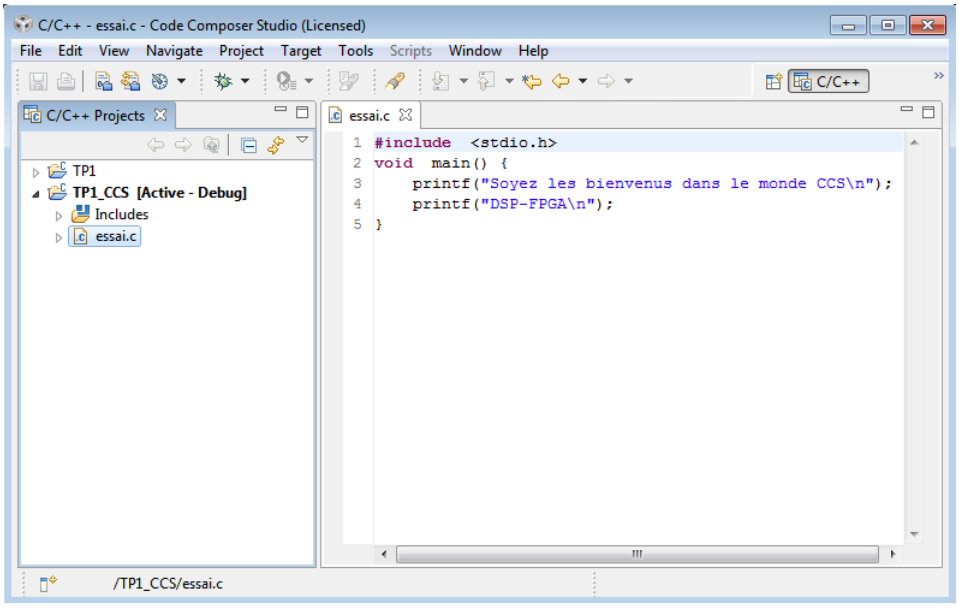

Figure 9 : Ouverture du fichier source dans l'éditeur de texte.

Construction du projet (Build Project)

Afin de construire le projet, cliquez sur "Project > Build Active Project" ou bien appuyez tout simplement sur la combinaison de touches "Ctrl+Shift+P" (voir Figure 10).

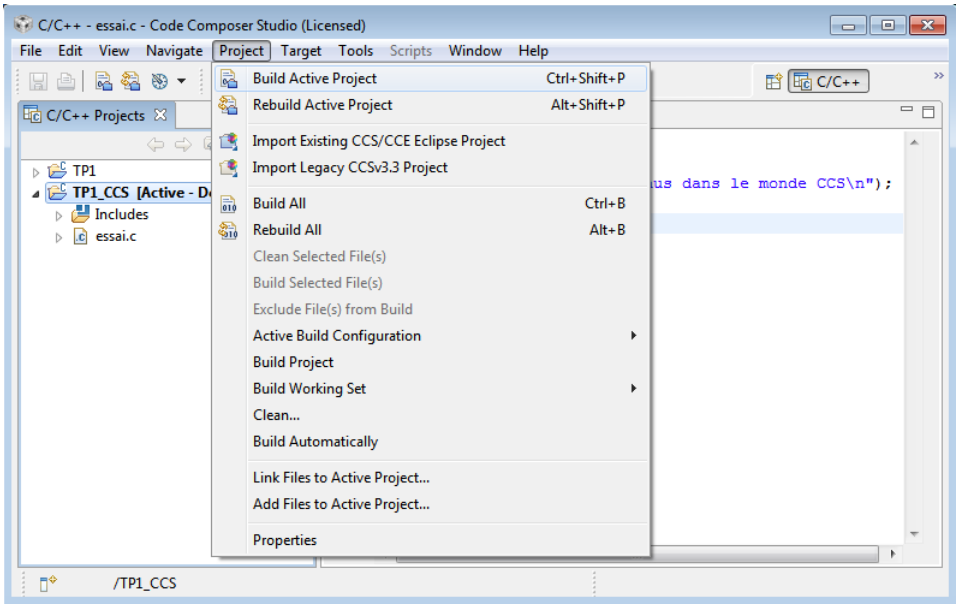

Figure 10 : Construction du projet actif.

Si le fichier source ne contient pas d'erreurs alors le processus de construction se déroule avec succès. La Figure 11 montre le résultat dans des zones en dessous de la fenêtre d'édition.

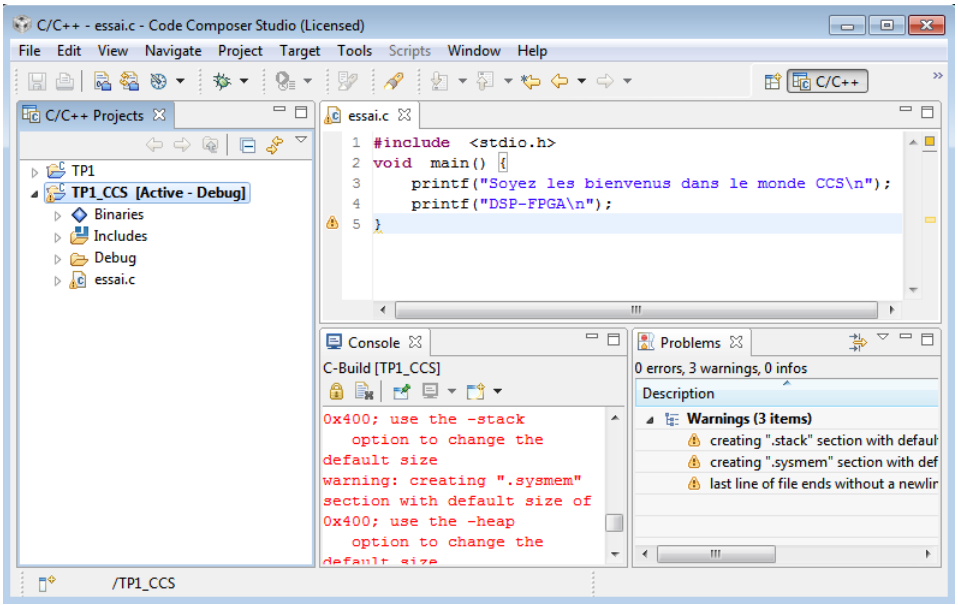

Figure 11 : Sortie de la construction du projet.

Création d'une cible de configuration

Un clic sur le menu "Target > New Target Configuration…" affiche la fenêtre illustrée par la Figure 12. Un onglet intitulé "" apparaît dans la barre des titre de la zone d'édition. Dans "General Setup", cochez par exemple "C6416 Device Cycle Accurate Simulator, Little Endian" puis appuyez sur le bouton "Save" situé juste à droite (voir Figure 13).

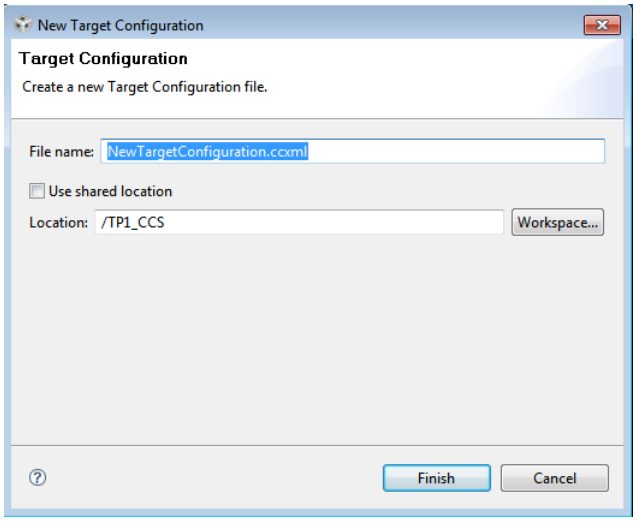

Figure 12 : Configuration de la cible.

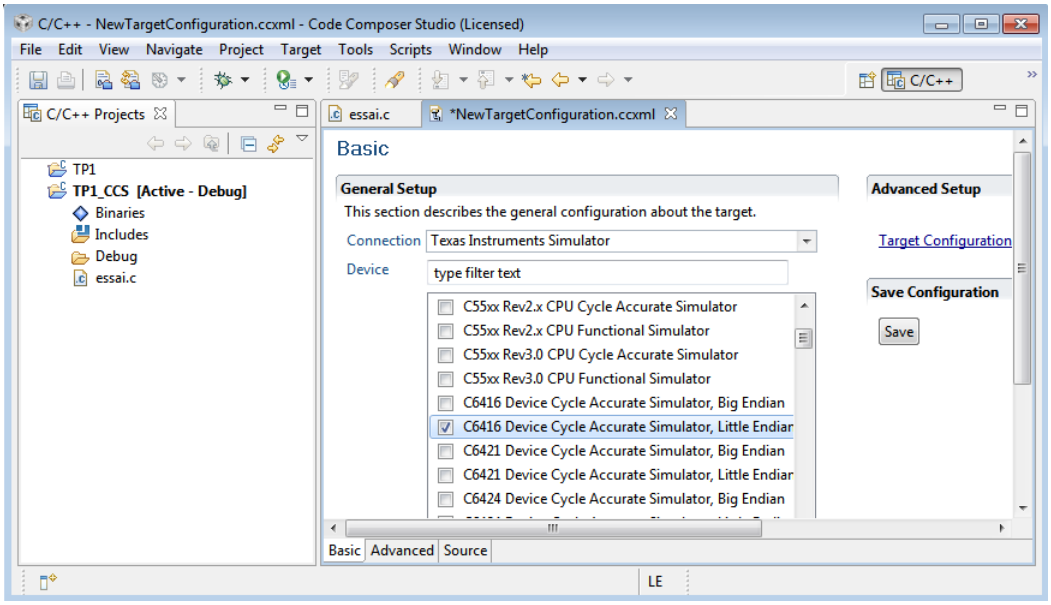

Figure 13 : Sauvegarde de la Configuration de la cible.

Exécution du projet

Cliquez sur le menu "Target > Debug Active Project", puis passez en mode en cliquant sur "Debug" dans le coin supérieur droit.

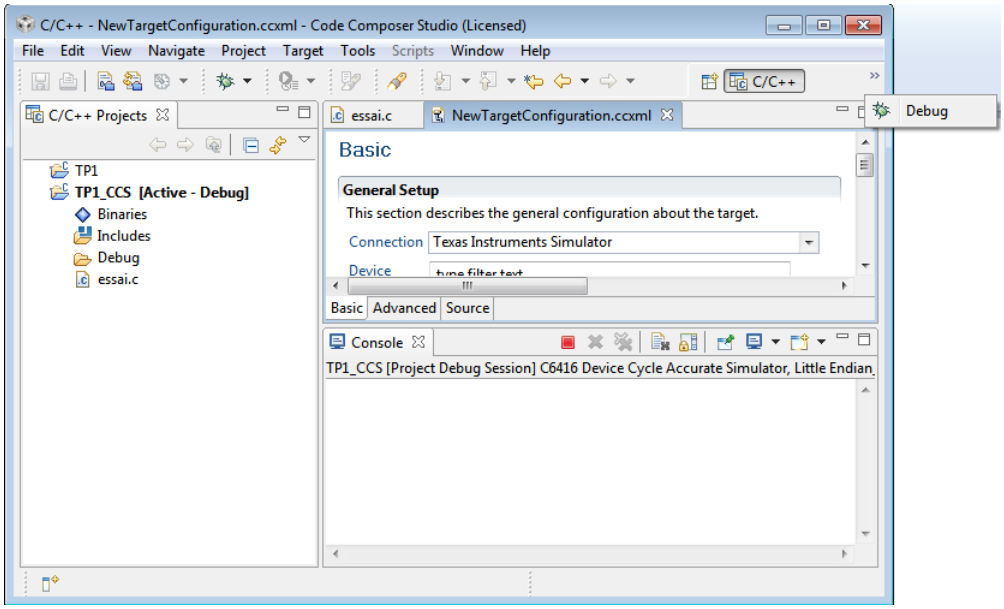

Figure 14 : Apparition de fichier de configuration dans la zone de texte.

Pour exécuter le projet, choisissez le menu "Target > Run" (voir Figure 15), le résultat s'affiche dans la zone console en bas de la fenêtre (voir Figure 16).

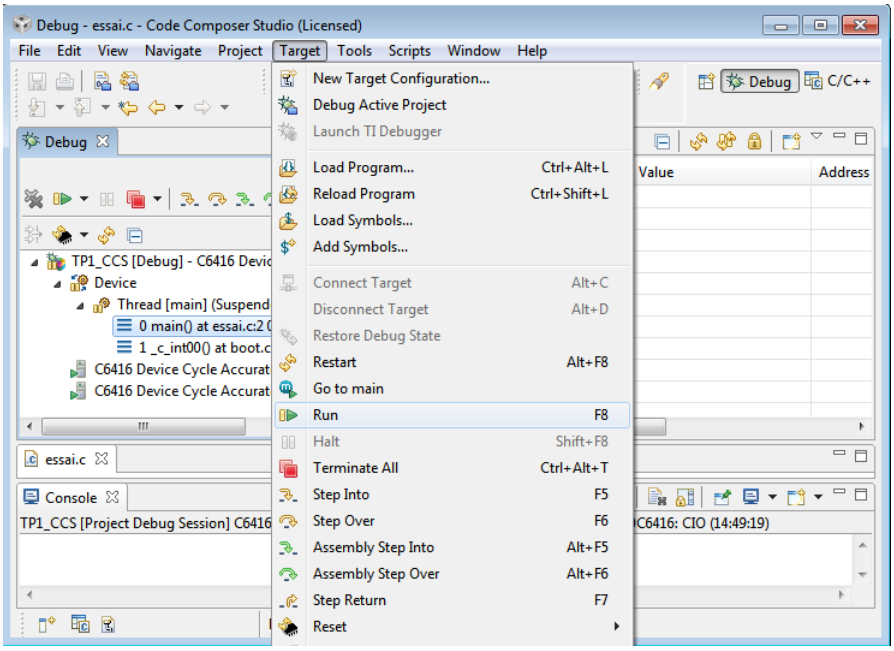

Figure 15 : Exécution du projet.

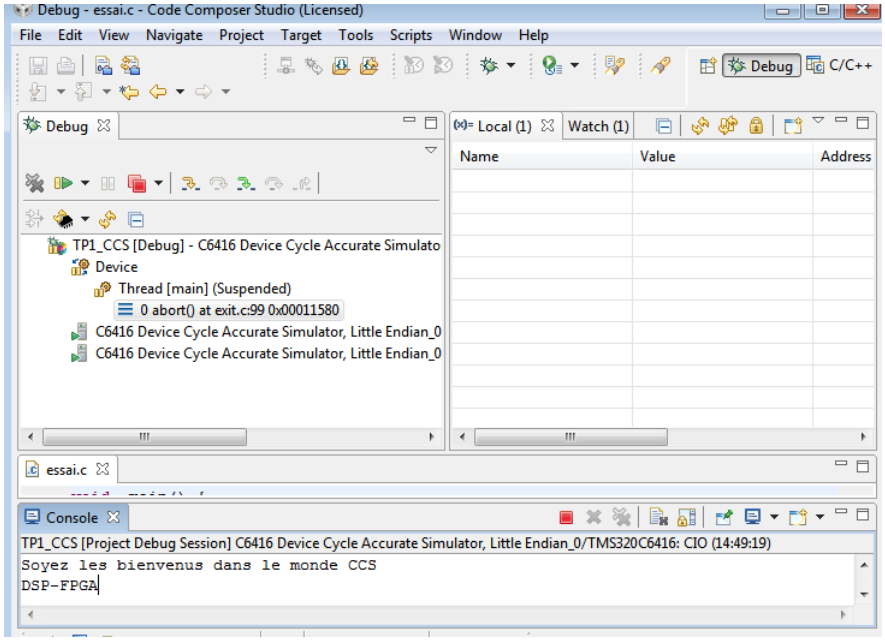

Figure 16 : Affichage du résultat d'exécution.

Travail demandé

Exercice 1 : Dans le projet TP1_CCS créé, modifier le fichier source C de telle sorte qu'il soit comme suit :

Listing 1 : fichier source C (essai.c)

```
#include  <stdio.h>
void  main( ) {
        //printf("BEGIN\n");
        //printf("END\n");
        short n=6;                    //set value
        short result;                 //result from asm function
        result = sumfunc(n);          //call ASM function sumfunc
        printf("sum = %d", result);  //print result from asm function
}
```

Ajoutez au projet TP1_CCS le fichier assembleur donné par le listing suivant :

Listing 2 : fichier source asm (sumfunc.asm)

```
;sumfunc.asm Assembly function to find n + (n-1) + ... + 1
                .def sumfunc                ;function called from C
sumfunc:        MV   .L1    A4,A1           ;setup n as loop counter
                SUB  .S1    A1,1,A1         ;decrement n
LOOP:           ADD .L1  A4,A1,A4           ;accumulate in A4
                SUB  .S1    A1,1,A1         ;decrement loop counter
        [A1]     B     .S2   LOOP           ;branch to LOOP if A1#0
                NOP 5                       ;five NOPs for delay slots
                B     .S2   B3              ;return to calling routine
                NOP 5                       ;five NOPs for delay slots
                .end
```

Reconstruire le projet.
Exécuter le projet.

Explication

Cet exercice illustre un programme C appelant une fonction assembleur. Le programme source C « essai.c » présenté par le « listing 1 » appelle la fonction codée en assembleur « sumfunc.asm » présentée par le « listing 2 ». Il implémente la somme de n + (n - 1) + (n - 2) + - - - + 1. La valeur de n est fixée dans le programme C. Elle est transmise par le registre A4 (par convention). Par exemple, l'adresse de plus d'une valeur peut être transmise à la fonction d'assemblage par A4, B4, A6, B6, et ainsi de suite. La somme résultante de la fonction d'assemblage (asm) est renvoyée comme résultat dans le programme C, qui imprime ensuite cette somme résultante.

La valeur n dans le registre A4 de la fonction asm est déplacée vers le registre A1 pour définir A1 comme compteur de boucle puisque seuls A1, A2, B0, B1 et B2 peuvent être utilisés comme registres conditionnels. A1 est ensuite décrémenté. Une section de code en boucle commence par l'étiquette ou l'adresse LOOP et se termine par la première instruction de branchement B. La première addition ajoute n + (n - 1) avec le résultat en A4. A1 est à nouveau décrémenté à (n - 2). L'instruction de branchement est conditionnelle en fonction du registre A1, et comme A1 n'est pas nul, le branchement a lieu et l'exécution revient à l'instruction à l'adresse LOOP, où A4 = n + (n - 1) est ajouté à A1 = (n - 2). Ce processus se poursuit jusqu'à ce que le registre A1 = 0.

La deuxième instruction de branchement est à l'adresse de retour B3 (par convention) du programme d'appel C. La somme résultante est contenue ou accumulée dans A4, qui est passée au résultat dans le programme C. Les cinq NOP (no operation) sont utilisés pour tenir compte des cinq slots de retard associés à une instruction de branchement.

Les unités fonctionnelles .S et .L sélectionnées sont affichées mais ne sont pas requises dans le programme. Ils peuvent être utiles pour déboguer et analyser quelles unités fonctionnelles sont utilisées afin d'améliorer l'efficacité du programme. De même, les deux deux-points après l'étiquette LOOP et le nom de la fonction ne sont pas nécessaires.

Générez et exécutez ce projet en tant que somme. Avec une valeur de n définie sur 6 dans le programme C, vérifiez que la somme et sa valeur de 21 sont imprimées.

Exercice 2 : Créer un nouveau projet CCS nommé TP11_CSS. Ajoutez au projet les deux fichiers sources donnés par les listings 3 et 4.

Listing 3 : Programme C qui appelle une fonction ASM pour trouver la factorielle d'un nombre (factorial.c)

```
/factorial.c finds factorial of n. Calls function factfunc
#include  <stdio.h>
void  main( ) {
        short n=7 ;                         //set value
        short result ;                      //result from asm function
        result = factfunc(n) ;              //call ASM function factfunc
        printf("factorial = %d", result) ;  //print result from asm function
}
```

Listing 4 : Fonction ASM appelée depuis le C qui trouve la factorielle d'un nombre (factfunc.asm)

```
;factfunc.asm Assembly function called from C to find factorial
              .def factfunc          ;ASM function called from C
factfunc:     MV    A4,A1            ;setup loop count in A1
              SUB   A1,1,A1          ;decrement loop count
LOOP:         MPY   A4,A1,A4         ;accumulate in A4
              NOP                    ;for 1 delay slot with MPY
              SUB   A1,1,A1          ;decrement for next multiply
        [A1]  B     LOOP             ;branch to LOOP if A1 # 0
              NOP   5                ;five NOPs for delay slots
              B     B3               ;return to calling routine
              NOP   5                ;five NOPs for delay slots
              .end
```

Explication

Cet exercice permet de trouver la factorielle d'un nombre n ≤ 7 avec n ! = n (n - 1)(n - 2) - - - (1). Il illustre en outre la syntaxe du code assembleur. Il est très similaire à l'exercice 1. La valeur de n est définie dans le programme source C factorial.c, présenté dans le listing 3, qui appelle la fonction d'assemblage factfunc.asm, présentée dans le listing 4. Il est instructif de lire les commentaires.

Le registre A1 est à nouveau défini comme un compteur de boucle. Dans la section de code de la boucle commençant par l'adresse LOOP, le premier multiplicateur est n (n - 1) et s'accumule dans le registre A4.

La valeur initiale de n est transmise à la fonction asm via A4. L'instruction MPY a un slot de retard, qui tient compte du NOP qui la suit. Le traitement se poursuit dans la section de code de la boucle jusqu'à ce que A1 = 0. Notez que les unités fonctionnelles ne sont pas spécifiées dans ce programme. La factorielle résultante est renvoyée au programme C appelant via A4.

Construisez et exécutez ce projet en tant que factorielle. Vérifiez que la factorielle et sa valeur de 5040 (7!) sont imprimées. Notez que la valeur maximale de n est 7, puisque le résultat est exprimé comme un short et que 8! est supérieur à 2^{15} .