

### Version des logiciels utilisés dans le tutoriel :

- Quartus Prime 18.1 Standard Edition
- Development Studio 2020.1-1

## I. Quartus Prime

### 1) Ouvrir un projet

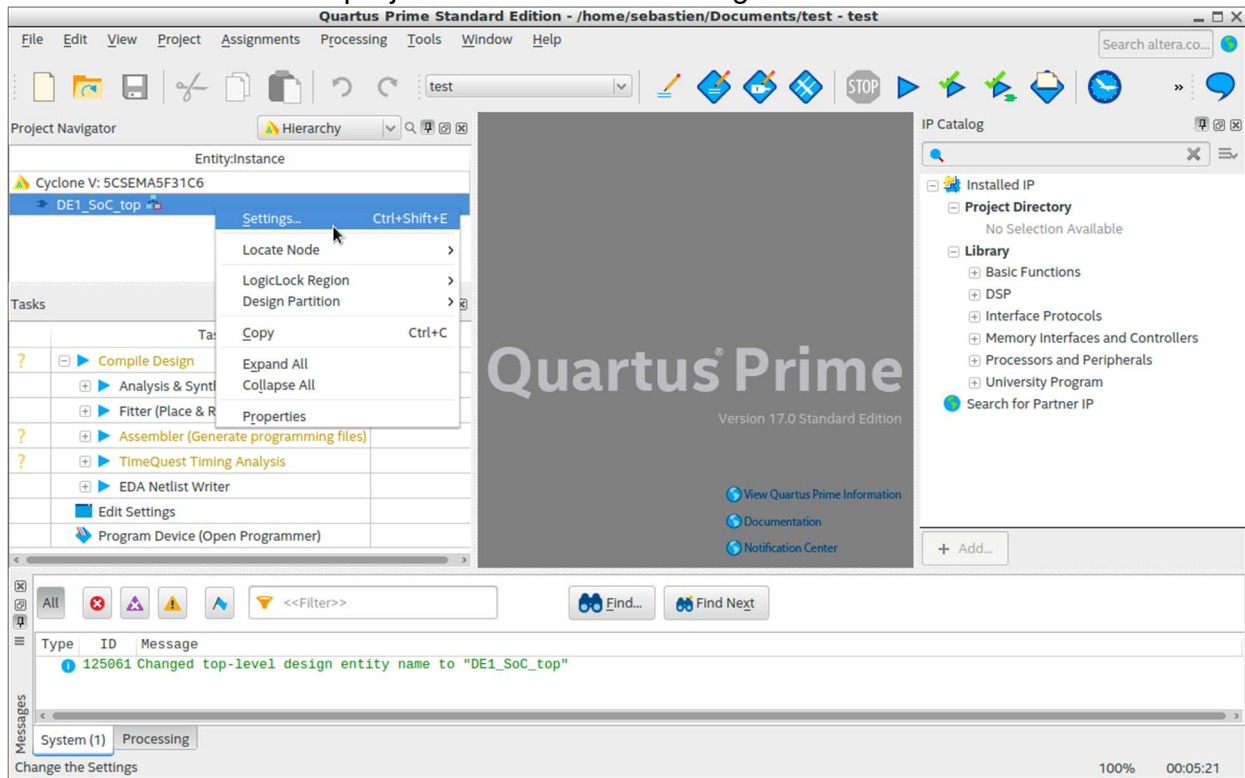
File → Open Project ...

Sélectionner le projet Quartus : .../hps\_gpio/hard/eda/DE1\_SoC.qpf → Ouvrir

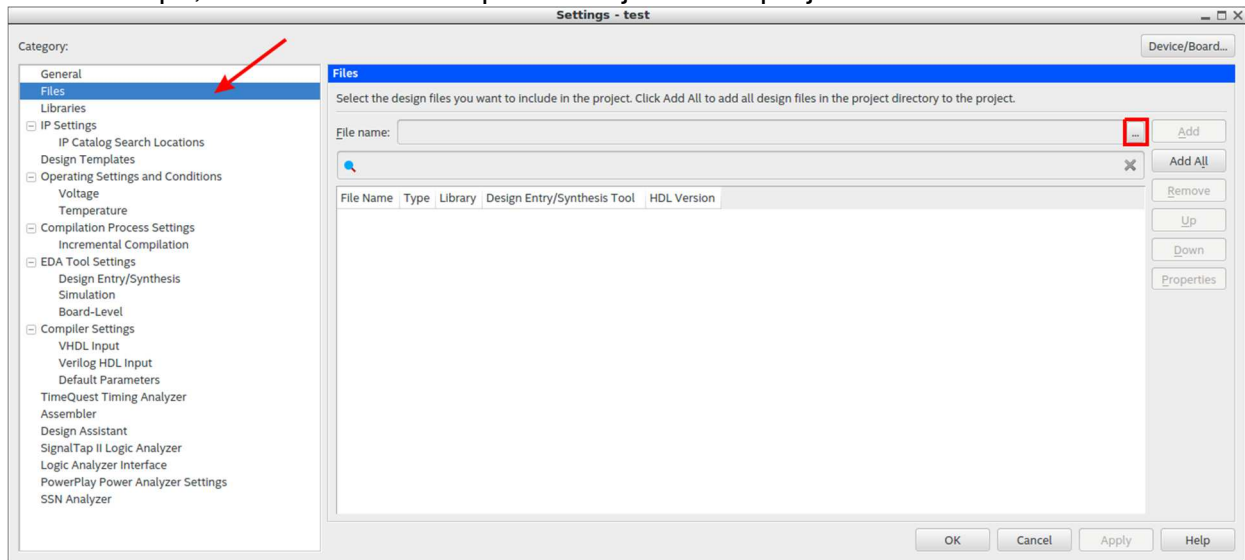
### 2) Ajouter des sources dans le projet

Si vous avez des nouveaux fichiers sources à ajouter dans le projet, voici la procédure :

Clic droit sur le nom du projet et sélectionner Settings...



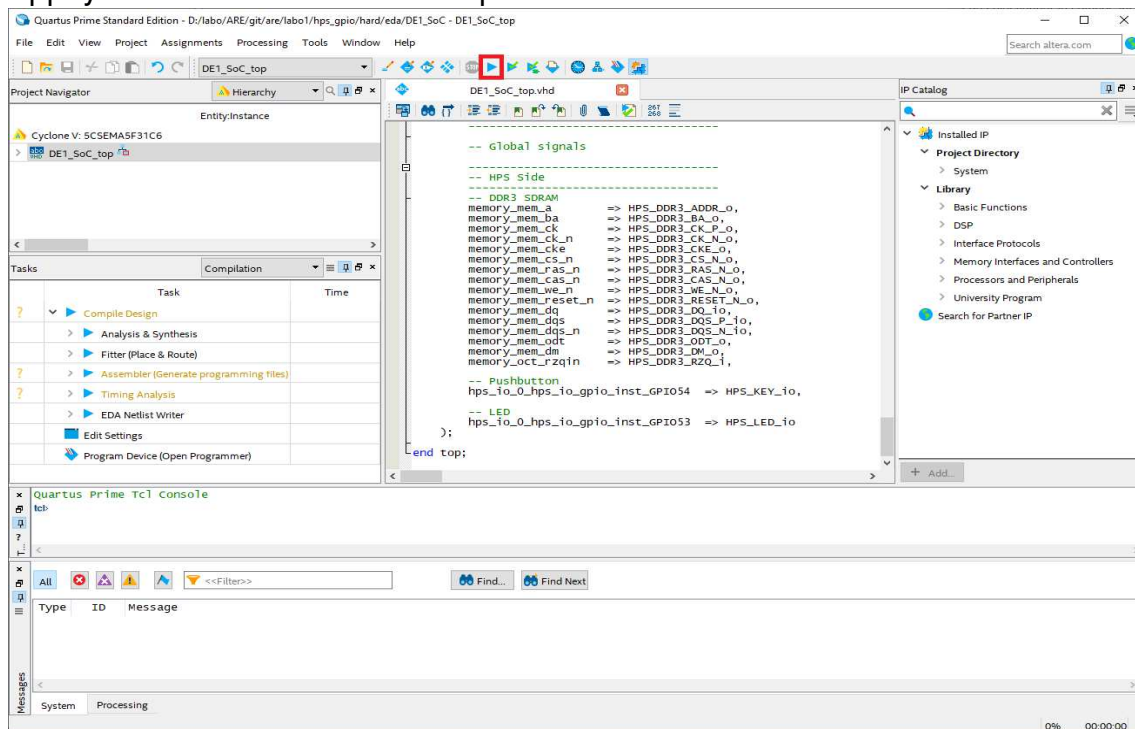
Sélectionner l'onglet Files dans la colonne de gauche. Cliquer sur le bouton "..."  
dans la partie de droite pour ajouter des fichiers sources. Aller dans le répertoire où  
se situe les fichiers. Si un seul fichier est sélectionné, il faut ensuite appuyer sur le  
bouton Add pour qu'il soit ajouté, alors que si plusieurs fichiers sont sélectionnés en  
même temps, ils seront automatiquement ajoutés au projet.



Cliquer sur OK une fois que les nouveaux fichiers sources sont présents dans le projet.

### 3) Générer le bitstream

Appuyez sur le bouton Start Compilation.



Lorsque la compilation est terminée, le bitstream se trouve :  
.../hps\_gpio/hard/eda/output\_files/DE1\_SoC\_top.sof

## II. ARM Développement Studio

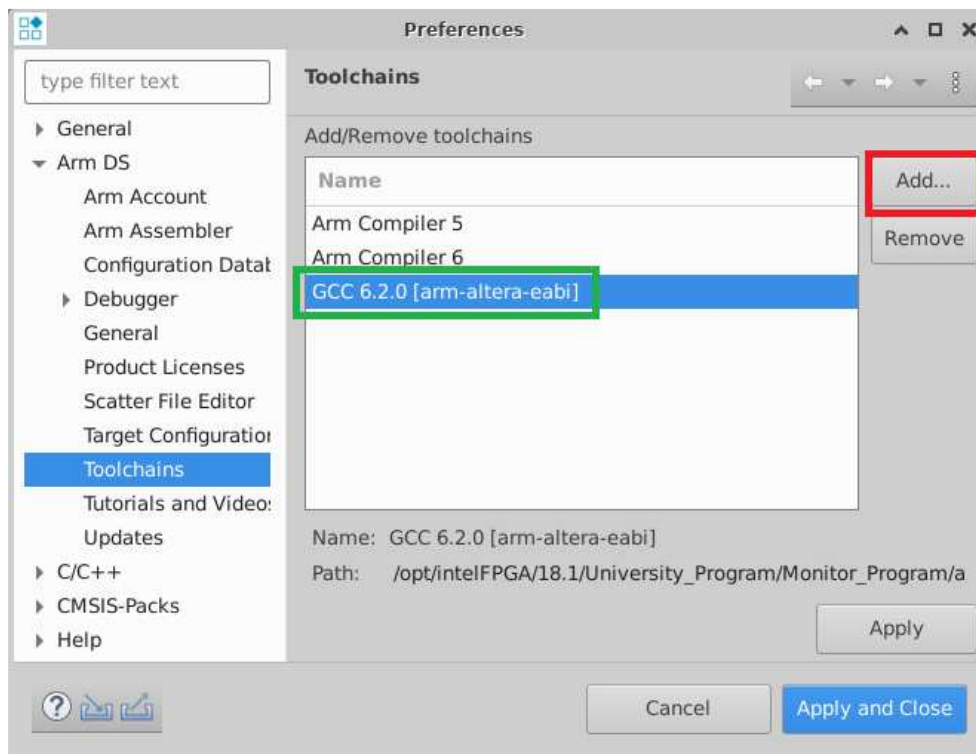
### 1) Ajout de la toolchain

Lors du premier démarrage du logiciel ARM-DS dans la VM, il faut ajouter la toolchain dans l'outil. Ensuite, il n'y aura plus besoin de faire cette étape.

Dans l'onglet **Window** → **Preferences**.

Choisissez dans le menu de gauche **ArmDS** → **Toolchains**.

Ensuite si la toolchain "**GCC 6.2.0 [arm-altera-eabi]**" n'est pas listée, il faut l'ajouter en cliquant sur **Add...**



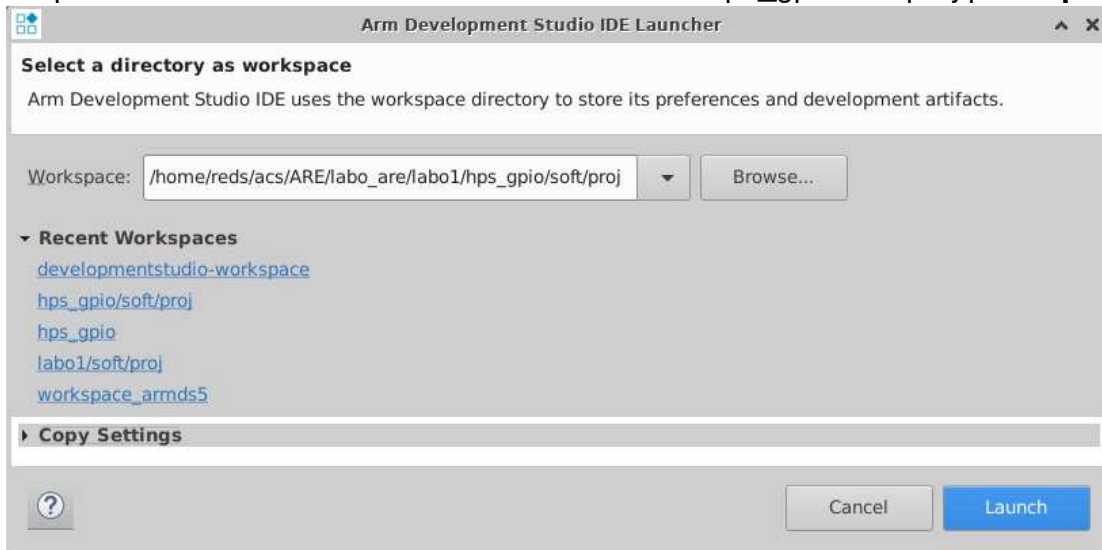
Puis pour sélectionner le Path cliquez sur **Browse...** et aller chercher le dossier contenant tous les exécutables arm-altera-eabi-\* et se trouvant :  
/opt/intelFPGA/18.1/University\_Program/Monitor\_Program/arm\_tools/baremetal/bin  
Puis **Open**, et ensuite **Next**.

La toolchain devrait être reconnue, laissez tout pour défaut et cliquez sur **Finish**.

## 2) Changer le workspace

Dans l'onglet File → Switch Workspace → Other...

Cliquer sur Browse et sélectionner le dossier : .../hps\_gpio/soft/proj puis **Open**



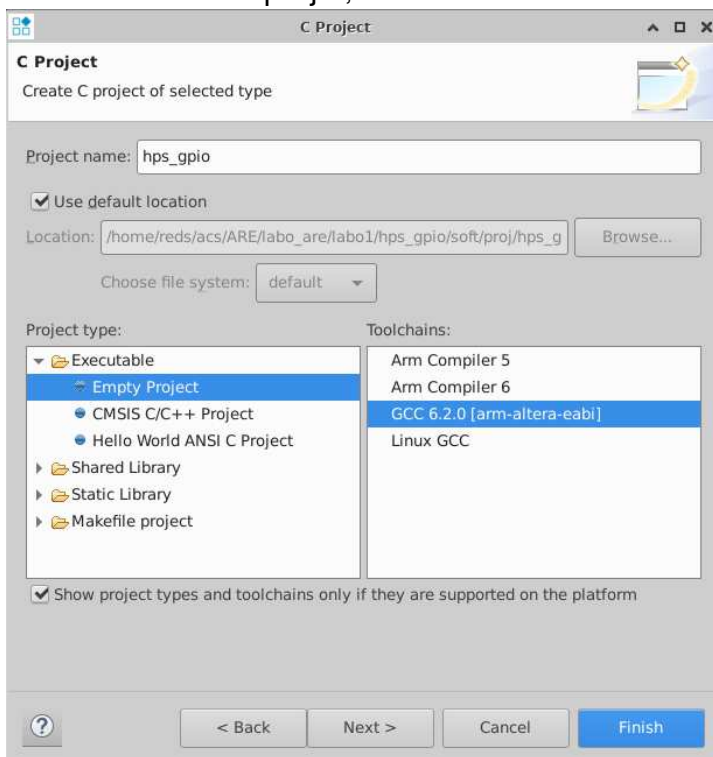
Cliquer ensuite sur **Launch**.

Le logiciel va redémarrer automatiquement avec le nouveau workspace.

## 3) Création d'un projet

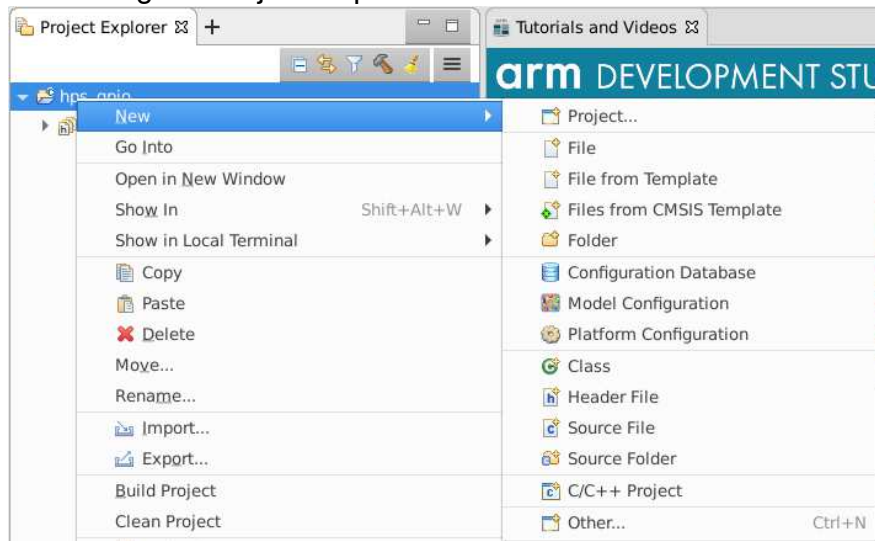
Dans l'onglet File → New → Project... choisissez C/C++ → C Project puis **Next >**.

Choisir le nom du projet, la location et la bonne toolchain de la manière suivante

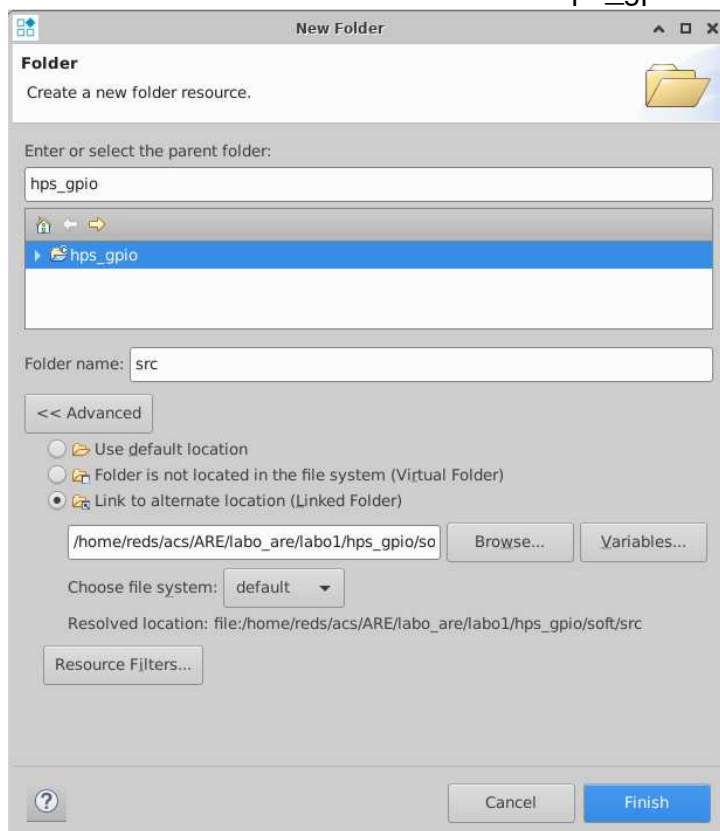


Vous pouvez ensuite tout laisser par défaut et cliquer sur **Next >** puis **Next >** et enfin **Finish**.

Pour ajouter le dossier source au projet, faite un clic droit sur le projet (hps\_gpio) dans l'onglet "Project Explorer" → **New** → **Folder**.

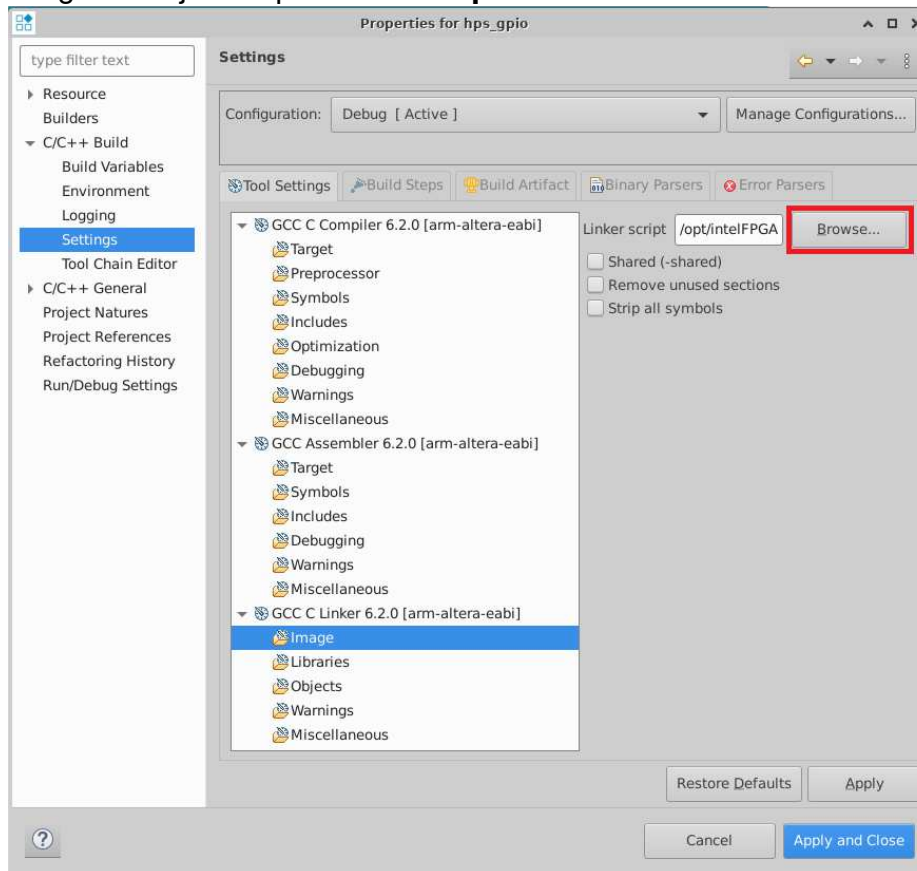


Dans la fenêtre qui s'ouvre, appuyer sur le bouton **Advanced>>**. Ensuite sélectionner « **Link to alternate location (Linked Folder)** » et Cliquer sur **Browse** et sélectionner le dossier : `.../hps_gpio/soft/src` puis **Open**. Et **Finish**.



Les fichiers contenus dans le dossier src seront automatiquement ajouté au projet.

Pour compiler le projet, il est nécessaire d'ajouter le script de linkage lié au HPS cyclone V. Aller dans les propriétés du projet, clic droit sur le projet (hps\_gpio) dans l'onglet "Project Explorer" → **Properties**.



Dans **C/C++ Build** → **Settings**, puis dans l'onglet "Tool Settings" → **GCC C Linker** → **Image**. Cliquer sur **Browse** et aller chercher le fichier se trouvant dans le dossier d'installation d'Altera Monitor Program :  
`/opt/intelFPGA/18.1/University_Program/Monitor_Program/arm_tools/baremetal/arm-altera-eabi/lib/cycloneV-dk-ram-hosted.ld`  
 Puis **Open**, et ensuite **Apply**.

Pour ajouter la génération du code assembleur, dans la même fenêtre, sélectionner l'onglet "Tool Settings" → **GCC C Compiler** → **Miscellaneous**. Compléter le champ Other flags avec : `-Wa,-adhlns="$@.lst"`  
 Puis **Apply and Close**.

Vous pouvez maintenant compiler votre projet en cliquant sur le petit marteau en haut à gauche dans la partie "Project Explorer", OU clic droit sur le projet (hps\_gpio) dans l'onglet "Project Explorer" → **Build Project**.

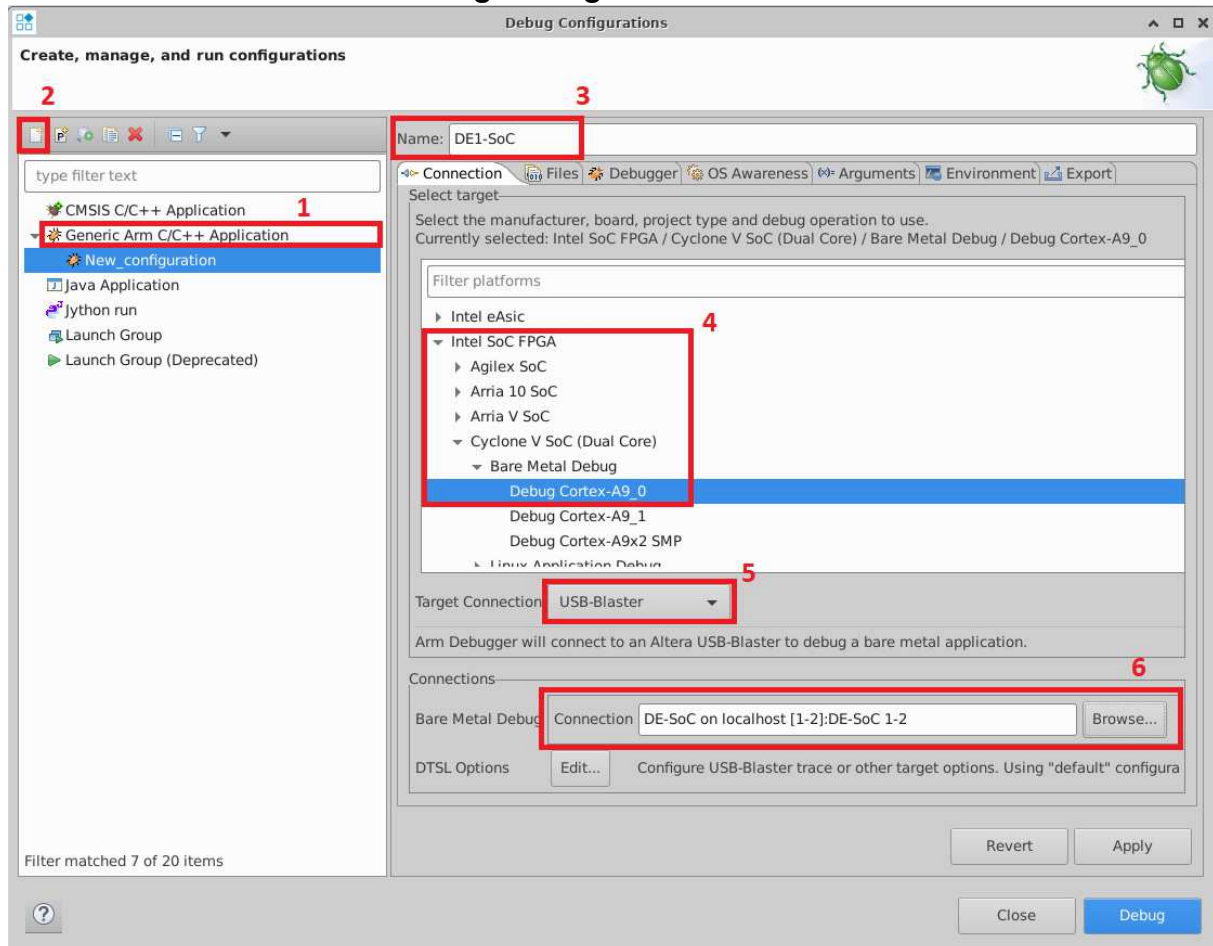
Après la compilation de votre projet, le ou les fichiers \*.lst sont générés dans le répertoire Debug/src de votre projet. Ils correspondent à votre code assembleur de vos sources C.



#### 4) Création d'une connexion de debug

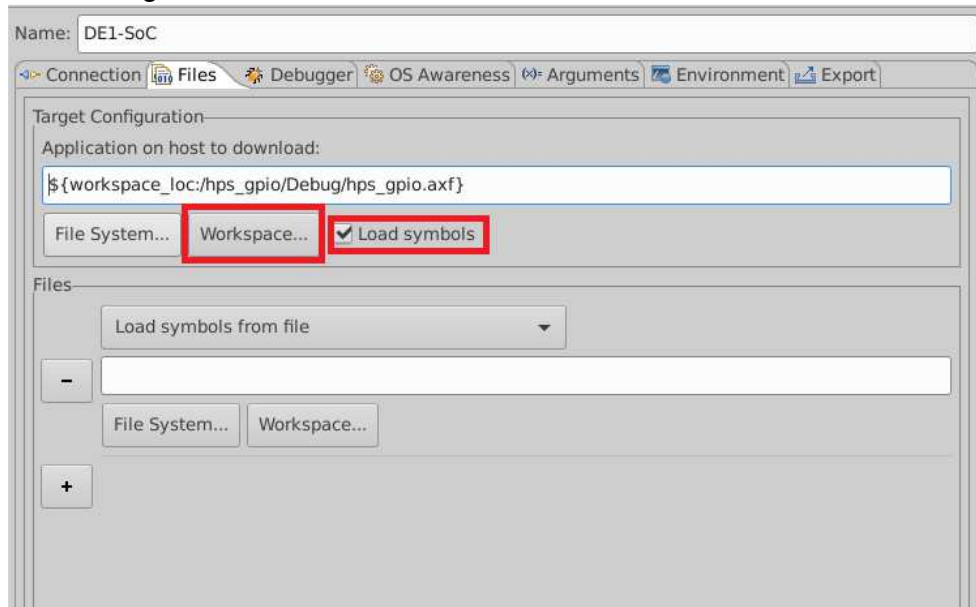
Afin de pouvoir exécuter et debugger votre programme, il est nécessaire de créer une configuration propre à votre cible (processeur).

Aller dans le menu **Run → Debug Configuration...**



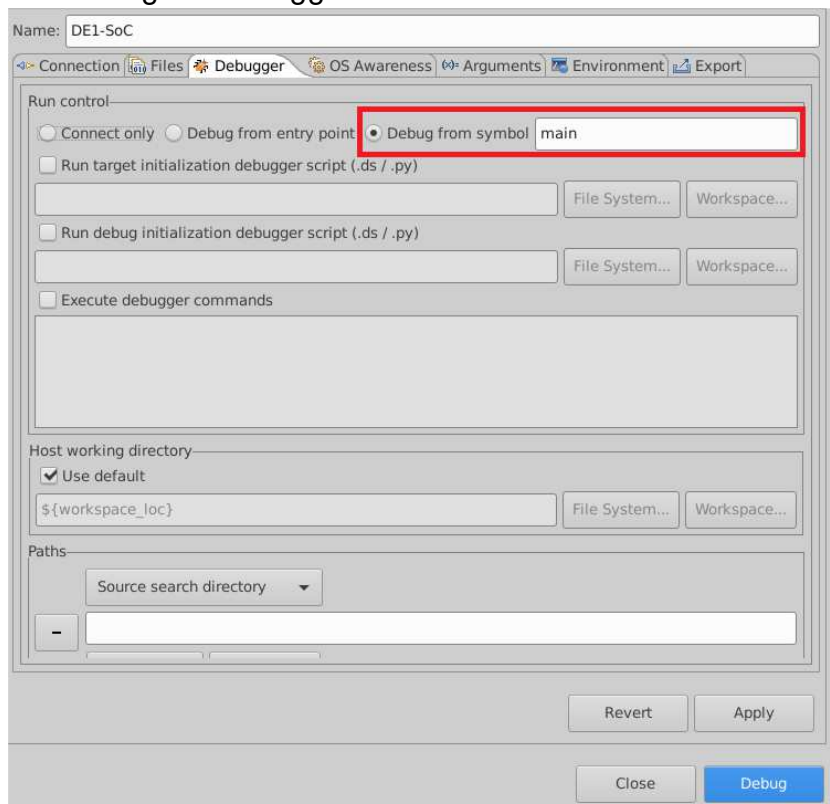
1. Sélectionner une configuration de type **Generic Arm C/C++ Application**
2. Appuyer sur **New launch configuration**
3. Donner un nom à votre configuration : **DE1-SoC**
4. Dans l'onglet "Connection" : Choisir la plateforme : **Intel SoC FPGA → Cyclone V SoC (Dual Core) → Bare Metal Debug → Debug Cortex-A9\_0**
5. Sélectionner **USB-Blaster** dans Target Connection.
6. Dans la partie Connections, cliquer sur **Browse...** et sélectionner la connexion vers la carte DE-SoC, puis **Select**. (Il faut attendre quelques secondes pour que la connexion vers la carte apparaisse, la carte doit être branchée au PC et allumée).

## Dans l'onglet "Files" :



1. Dans la partie Target Configuration, sélectionner le fichier « xxx.axf » générée lors de la compilation du projet en cliquant sur **Workspace...** Puis hps\_gpio/Debug/hps\_gpio.axf
2. Cocher la case **Load Symbols**.

## Dans l'onglet "Debugger" :



1. Cochez **Debug from symbol** et spécifier « main ». Cela mettra le debugger en pause à l'entrée de la fonction main lors de son lancement.
2. Puis cliquer sur **Apply** et **Close**.



## 5) Exécuter / debugger le programme C

### 4.1) Programmation DE1-SoC et chargement preloader du HPS

**Important :** Avant de pouvoir exécuter et debugger un programme C, il est nécessaire de programmer la fpga de la DE1-SoC et de charger le preloader du HPS. Pour cela, il faut exécuter les scripts python suivant.

Ouvrir un terminal en dehors de ARM-DS. Puis aller dans le répertoire

.../hps\_gpio/hard/script

Pour programmer la fpga, exécuter le script python3 avec en paramètre le bitstream à l'aide de la commande suivante :

```
$ python3 pgm_fpga.py -s=../eda/output_files/DE1_SoC_top.sof
```

Pour charger le preloader du HPS, exécuter le script python3 suivant :

```
$ python3 upld_hps.py
```

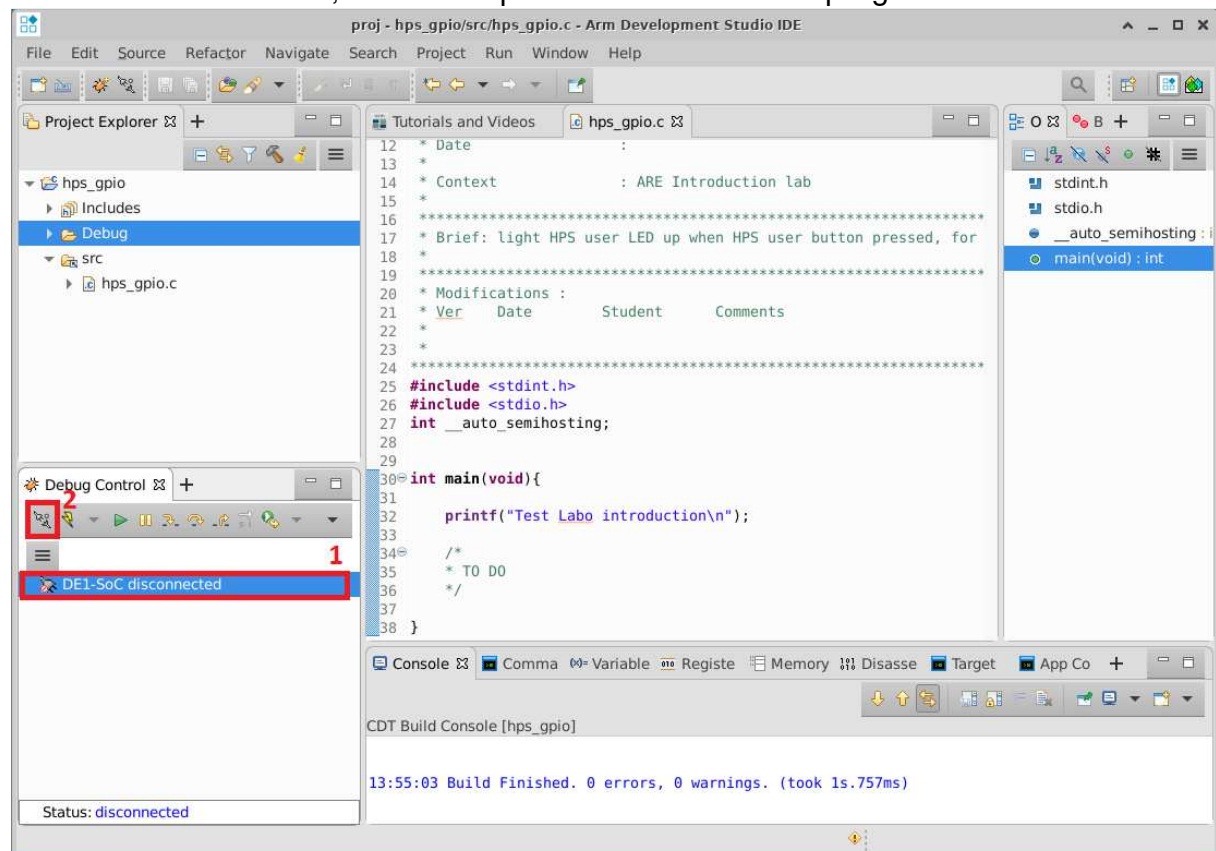
A la fin du log vous devez obtenir le message suivant :

>>Preloader successfully run.

**Note :** Lors de la génération d'un nouveau bitstream (.sof), il faudra relancer cette programmation avec les scripts python3. Attention, il est nécessaire de déconnecter le debugger de la carte (partie 4.2) pour exécuter les scripts.

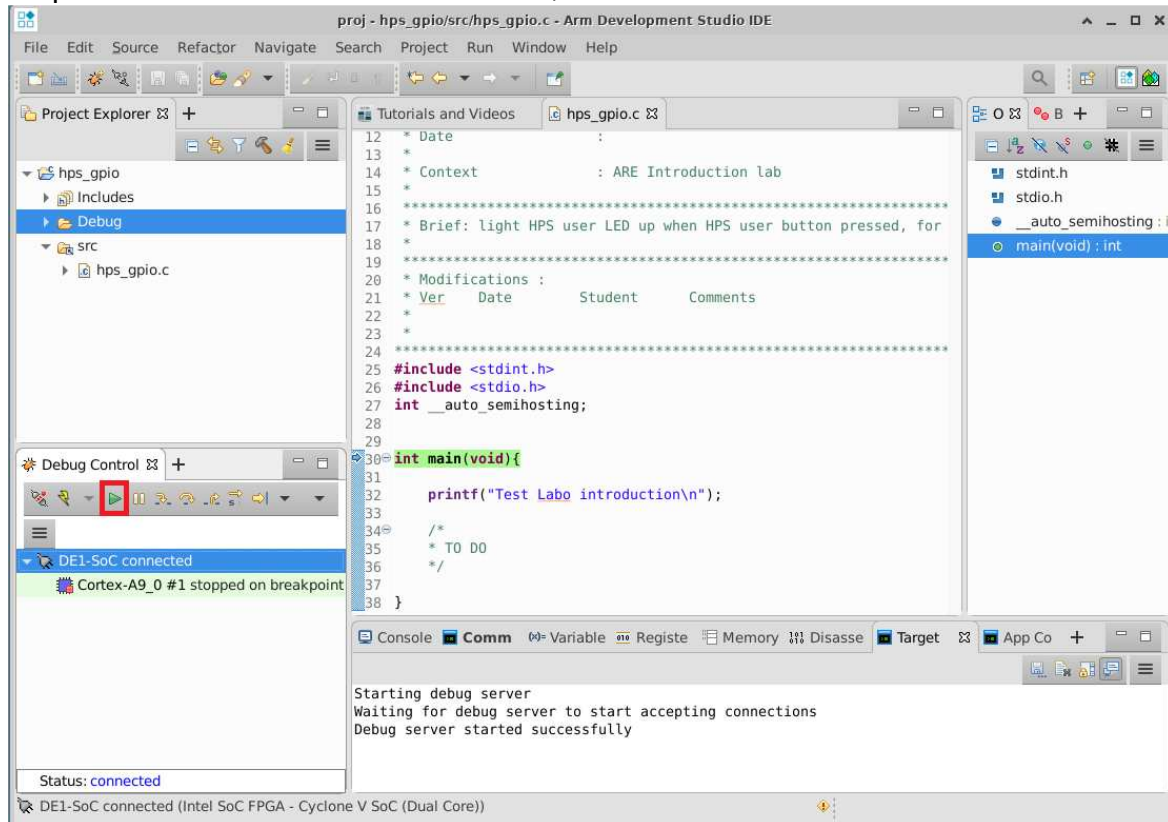
### 4.2) Exécution du programme C

Ensuite dans ARM-DS, vous allez pouvoir exécuter votre programme C :







1. Dans l'onglet "Debug Control", sélectionner la configuration de debug créée précédemment.
2. Appuyer sur le bouton "Connect to target".

La connexion avec la carte doit s'établir et l'exécution du programme doit se mettre en pause au début de la fonction main, comme sur la fenêtre suivante.



Cliquez sur le bouton "Continue" pour poursuivre l'exécution du programme.

Depuis l'onglet "Debug Control", vous pouvez effectuer différents types d'actions, en voici certaine :

-  Continuer l'exécution du programme jusqu'à la fin ou un breakpoint.
-  Mettre le programme en pause.
-  Exécuter la ligne suivante.
-  Déconnecter le debugger de la carte.

Si vous souhaitez quitter proprement l'exécution du programme pour ensuite le réexécuter plus tard, il faut mettre le programme en pause et déconnecter le debugger de la carte.

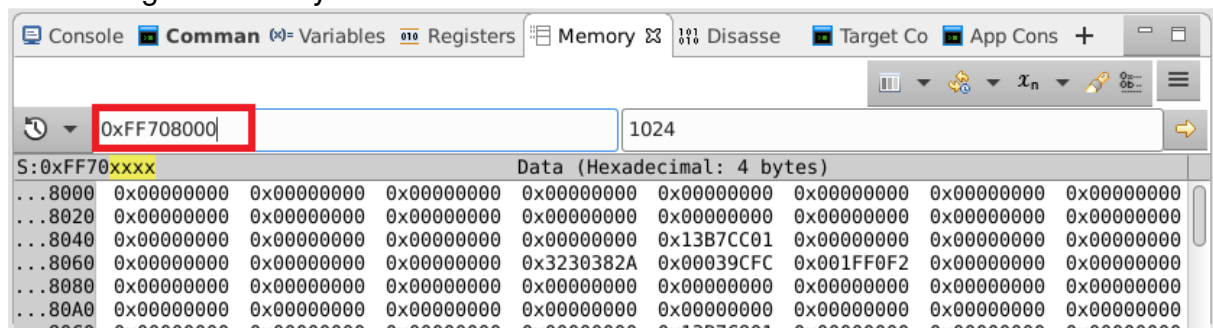
Afin de pouvoir utiliser la commande printf dans votre programme C, il est nécessaire d'ajouter les lignes suivantes avant la fonction main :

```
#include <stdio.h>
int __auto_semihosting;
```

## 6) Ecriture et lecture à une adresse mémoire

Pour exécuter des écritures et lectures à des adresse mémoire, il faut avoir connecté le debugger à la carte et mettre l'exécution du programme en pause.

Dans l'onglet "Memory" :



Entrer l'adresse mémoire complète sur 32bits pour l'écriture ou la lecture, puis appuyer sur la touche « Enter » de votre clavier.

Vous obtiendrez une représentation de la mémoire avec toutes les valeurs lus.

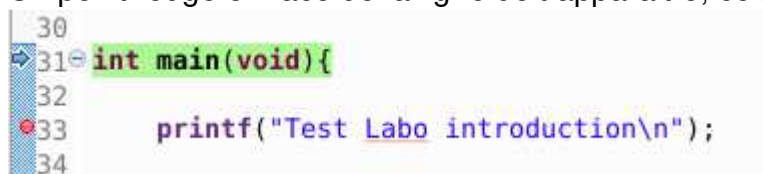
Pour faire une écriture, double-clic sur la valeur dans la représentation de la mémoire. Puis, écrire la nouvelle valeur et appuyer sur la touche « Enter » de votre clavier. L'écriture est exécutée à l'adresse mémoire.

## 7) Ajouter un breakpoint

Pour ajouter un breakpoint dans un programme C, ouvrir le code C dans l'éditeur de ARM-DS.

Faire un double-clic sur le numéro de ligne à gauche de la ligne désiré pour mettre le breakpoint.

Un point rouge en face de la ligne doit apparaître, cela signale le breakpoint :



Lors de l'exécution du programme, il doit s'arrêter à chaque passage sur un breakpoint.

Pour supprimer un breakpoint, faire également un double-clic sur le numéro de ligne et le point rouge doit s'enlever.

## 8) Visualisation du code assembleur du programme

Lorsque l'exécution du programme est en pause, dans l'onglet "Disassembly" vous pouvez voir le programme assembleur.

### III. Structure du répertoire projet

