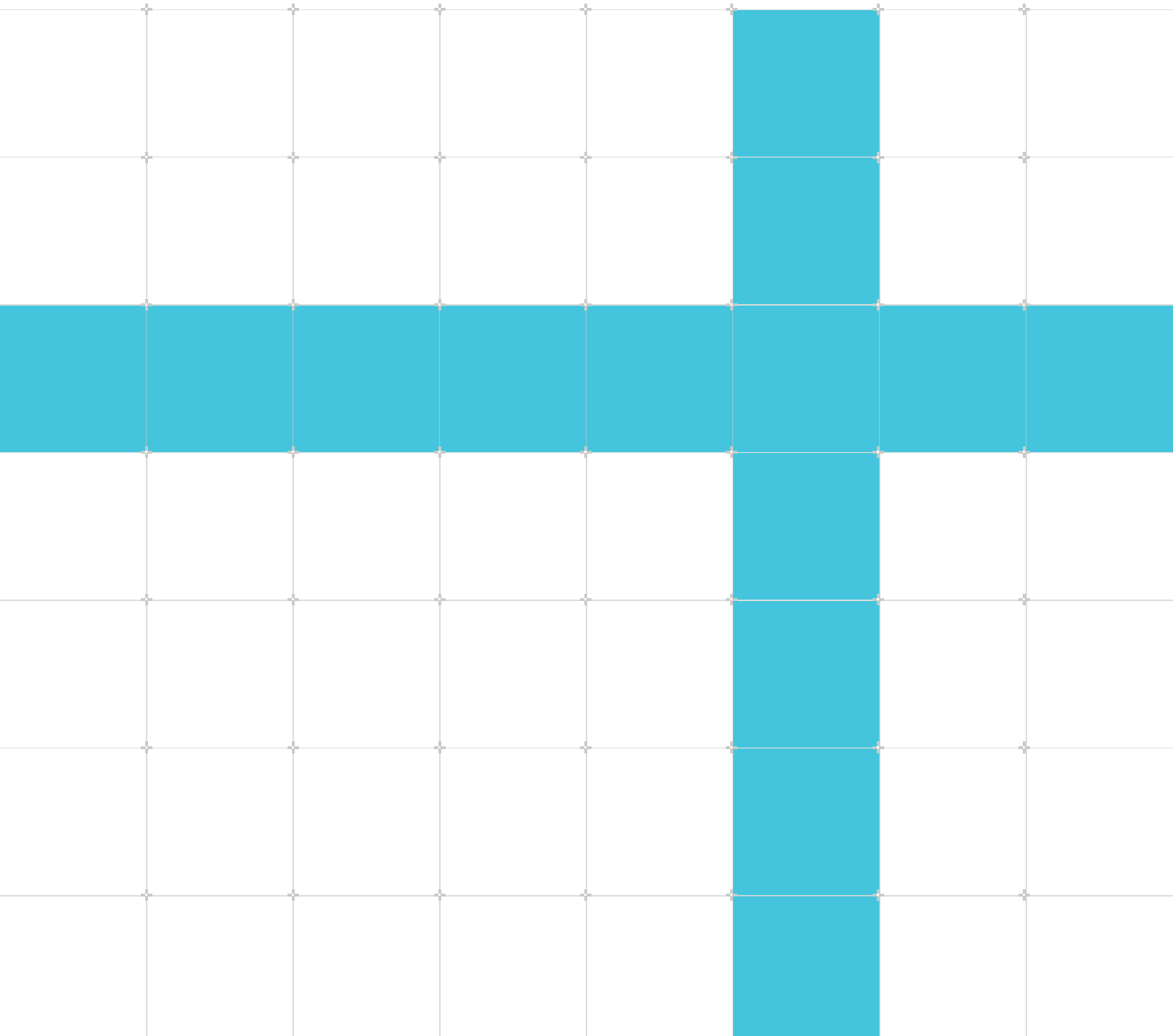
Arm Compiler 6 - Bare-metal Hello World C using the Armv8 model

Version 1.0

Issue 02
102675_0100_02_en



Arm Compiler 6 - Bare-metal Hello World C using the Armv8 model

Release information

Document history

Issue	Date	Confidentiality	Change
0100-02	11 April 2019	Non-Confidential	Initial release

Proprietary Notice

or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Overview

In this tutorial, you will learn how to build Hello World with Arm Compiler 6 and debug it on the Armv8 Fixed Virtual Platform (FVP)

Building Hello World with Arm Compiler 6 and debugging it on the Armv8 Fixed Virtual Platform

This Arm® DS-5 Development Studio tutorial covers a basic Hello World C program. It will be useful if you want to carry out bare-metal software development on Armv8 platforms, and shows you how to:

- Write “Hello World” in C
- Build it using the Arm Compiler 6 (this is Arm’s next generation LLVM-based toolchain)
- Set up a debug session in Arm DS-5 Development Studio
- Run it on a Armv8 Fixed Virtual Platform (FVP)

To complete this tutorial, you’ll need:

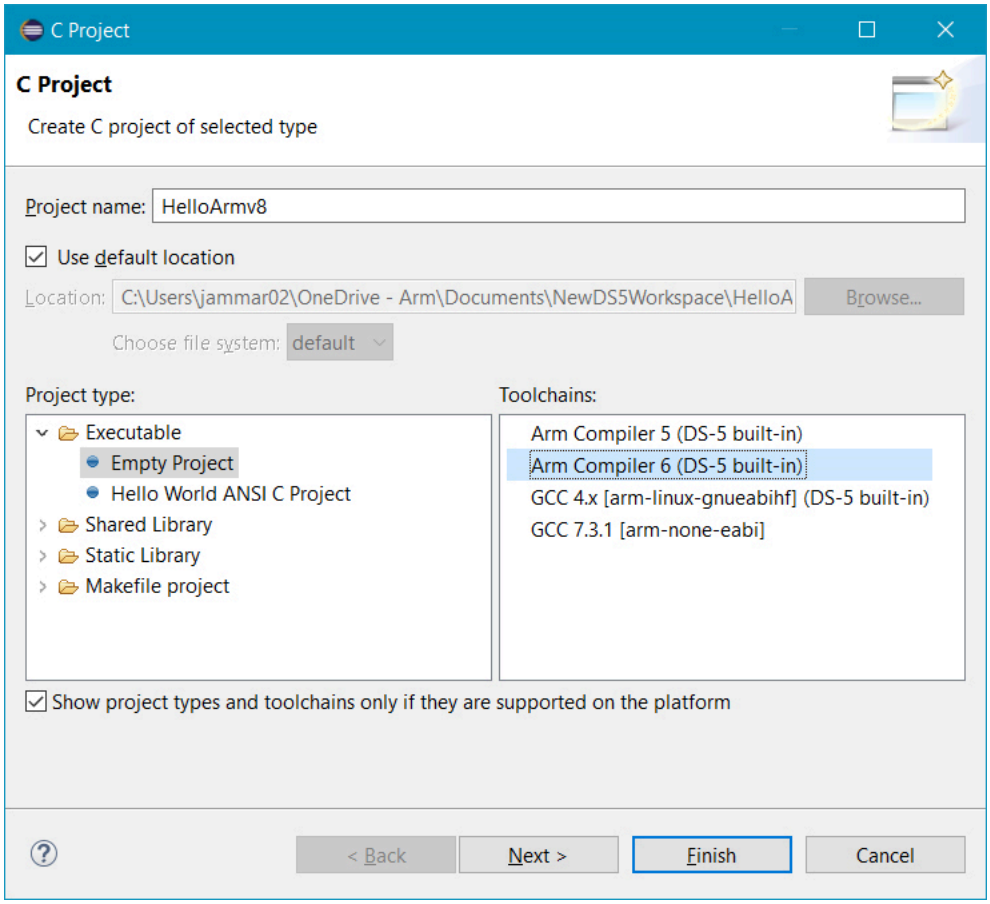
- DS-5 Ultimate Edition: Download the 30-day trial
  - Arm Compiler 6 toolchain (included)
  - Armv8 Fixed Virtual Platform (FVP) (included)
  - DS-5 Debugger (included)

2. Creating a new project

To create a new project follow the steps:

1. From the DS-5 C/C++ perspective main menu, select File > New > C Project to display the C Project dialog.
2. In the C Project dialog:
   a. In the Project name field, enter HelloArmv8 as the name of your project.
   b. Under Project type, select Executable > Empty Project.

Figure 2-1: C Project dialog options for ARM Compiler 6



   c. Under Toolchains, select Arm Compiler 6.
   d. Click Finish to create your C project.

Your project will appear in the Project Explorer view.

3. Configuring the project toolchain settings

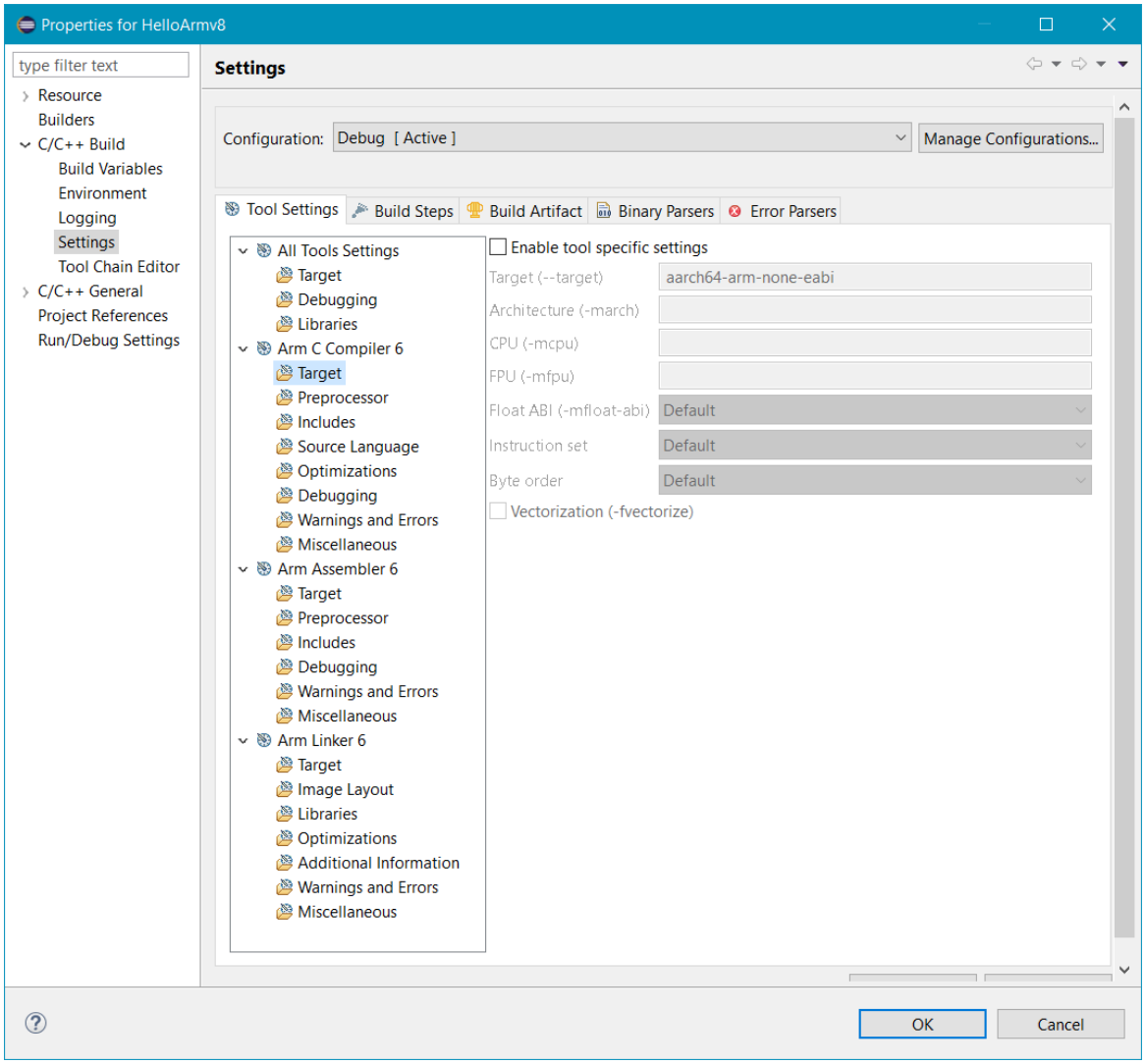
Now that you've created a new project, you need to configure the settings for the compiler toolchain. These settings allow you to specify a target architecture, CPU and starting address for where your executable will be loaded in the memory of the target.

Setting up the commands and flags for the Arm Compiler 6 toolchain

This configuration exists on a per-project basis. You must separately reconfigure all projects that you want to use with the new toolchain.

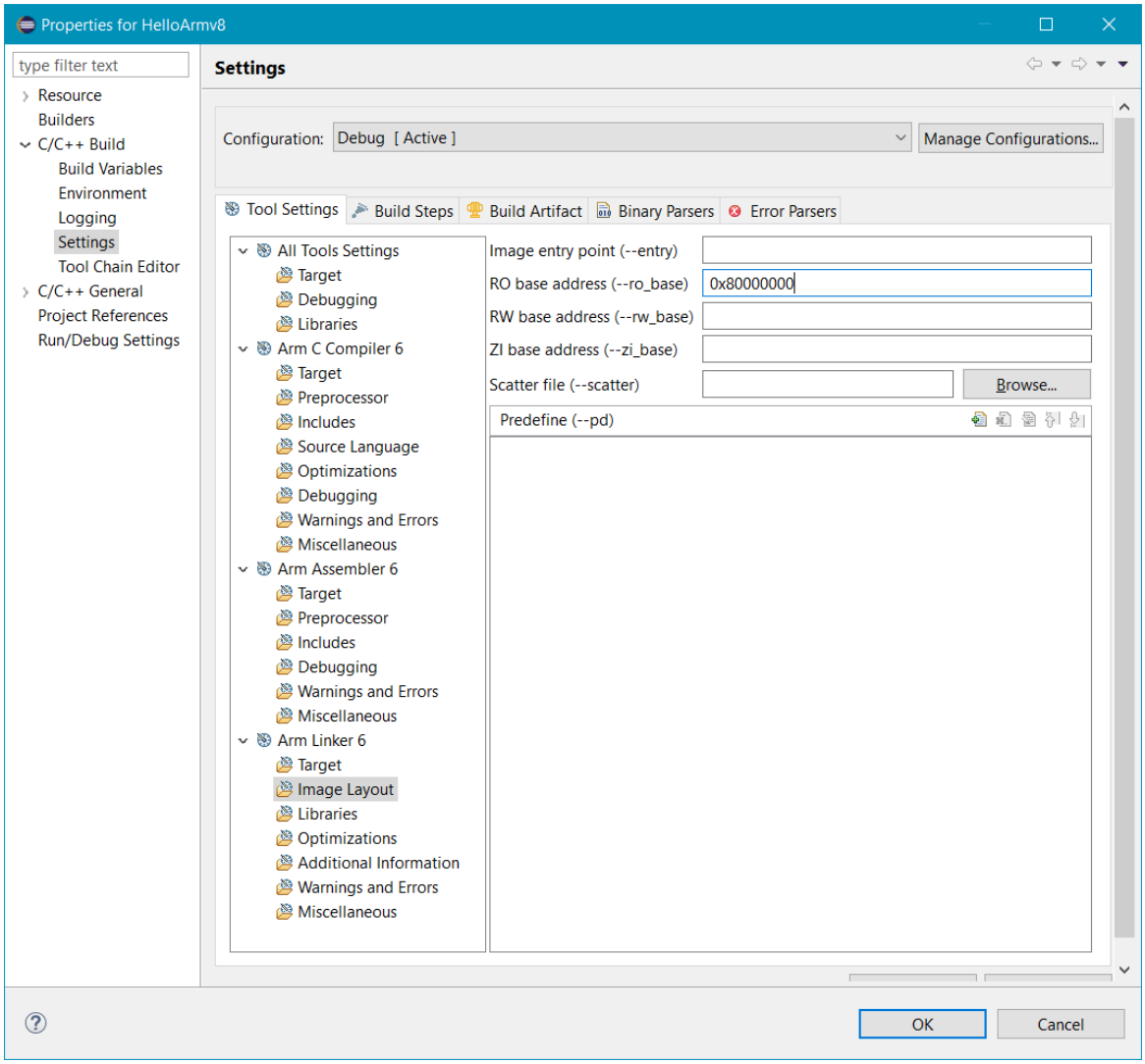
1. Locate your project in the Project Explorer view, right-click it, and select Properties.
2. In the Properties dialog:
   a. Navigate to C/C++ Build > Settings.
   b. Select the Tool Settings tab.
   c. Select Arm C Compiler 6 > Target and check that `aarch64-arm-none-eabi` has been specified, so that you are building for the Armv8 architecture, AArch64 execution state, targeting a generic device.

Figure 3-1: Arm Compiler 6 Hello World target architecture



d. Now, in Arm Linker 6 > Image Layout enter a read only base address of `0x80000000`. This needs to be done in order to insert the Arm executable at the correct point in the model's memory.

Figure 3-2: Arm Compiler 6 Hello World base address



The commands and flags for the Arm Compiler 6 are now set up, so it's time to write our Hello World C code.

4. Writing the source code and building the project

Now that the project is set up, we can start creating code for it and build the application. After a successful build, we can then create a debug configuration, and run the application on the target.

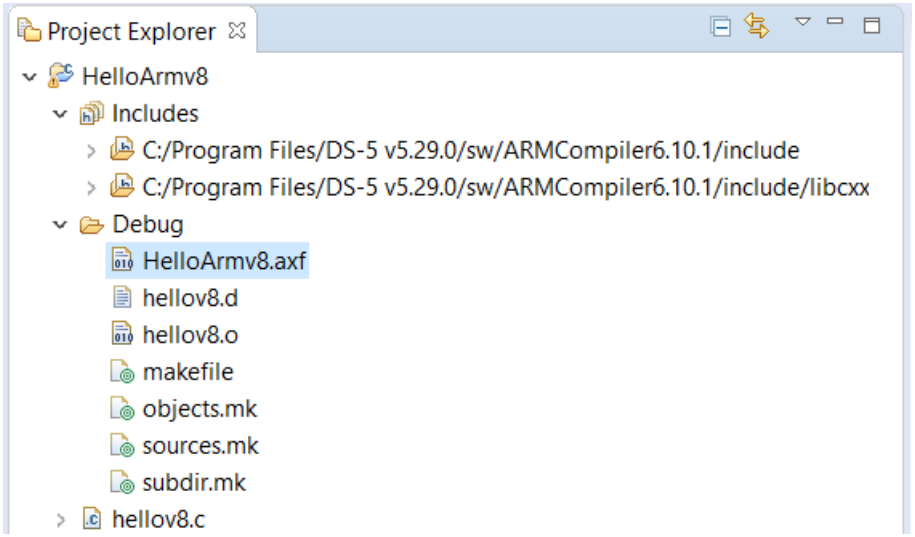
1. In the Project Explorer view, right-click the HelloArmv8 project and select New > Source File.
2. In the New Source File dialog, enter the file name hellov8.c.
3. Click Finish to create the source file and open it in the code editing view. The source file is visible in the Project Explorer view, under the HelloArmv8 project.
4. Add the following code to the new source file, and press Ctrl+S to save it.

```
#include <stdio.h>
int main()
{
  printf("Hello world\n");
  return 0;
}
```

5. In the Project Explorer view, right-click on the HelloArmv8 project and select Build Project. You can view the output image HelloARMv8.axf in the Debug folder under the HelloArmv8 project.

The `.axf` file contains both the object code and debug symbols that enable the debugger to perform source-level debugging.

Figure 4-1: rm Compiler 6 Hello World Project

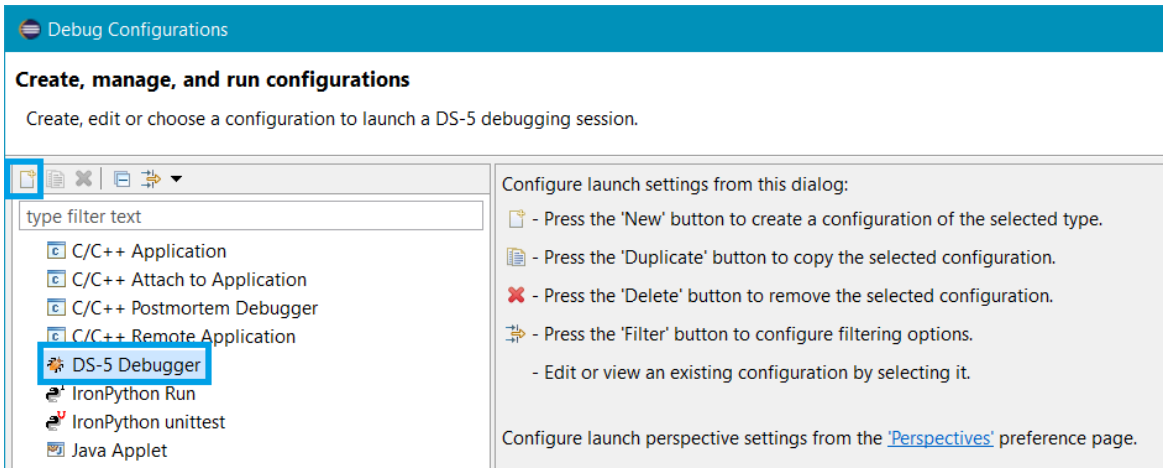


5. Creating a DS-5 debug configuration and connecting to the FVP

Here we describe the steps on how to create and connect a DS-5 debug configurationto the FVP.

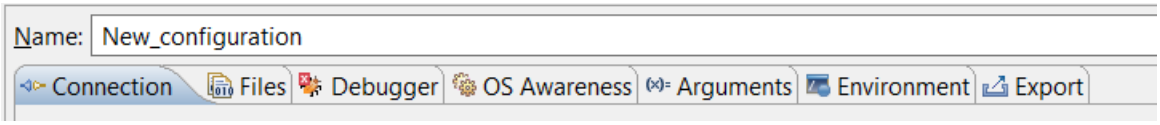
1. From the DS-5 main menu, select Run > Debug Configurations.
2. In the Debug Configurations dialog:
   a. Select DS-5 Debugger.
   b. Click the New launch configurations button.

Figure 5-1: Debug configurations



This creates a new DS-5 debug configuration and displays the various tabs required to specify settings for loading your application on the target.

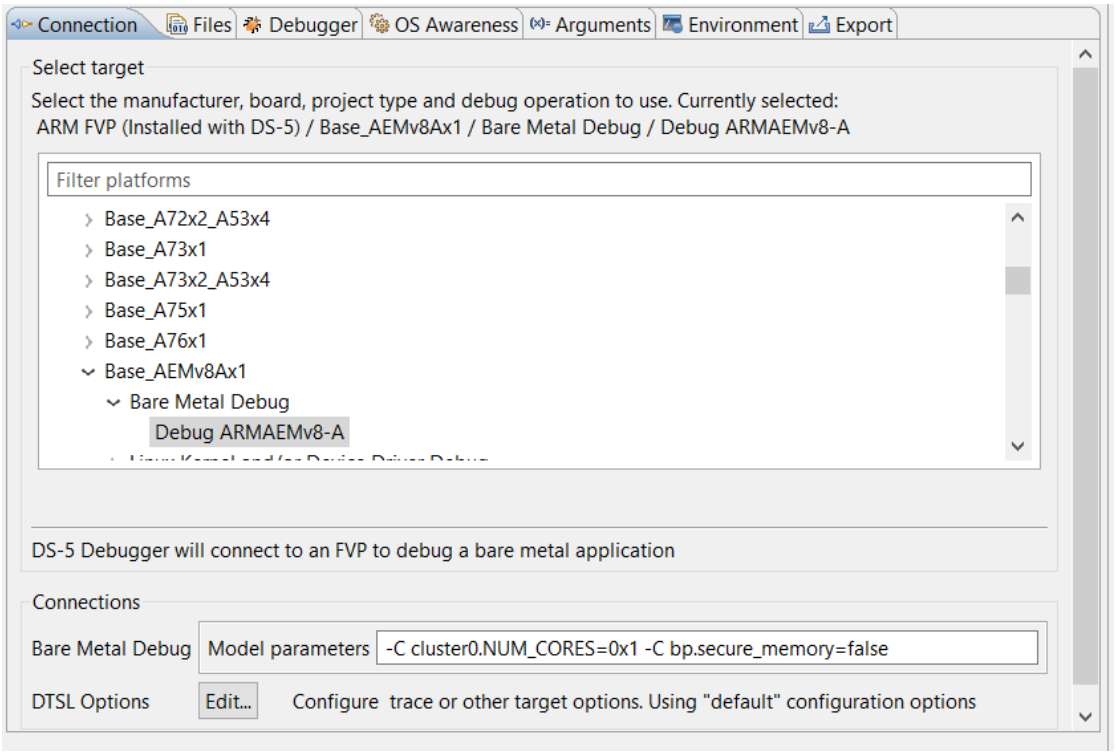
Figure 5-2: Debug configurations - Tabs



   c. In the Debug Configurations dialog:
      1. Give a name to the debug configuration. For example, HelloArmv8.
      2. In the Connection tab, select Arm FVP (Installed with DS-5) > Base_AEMv8Ax1 > Bare-Metal Debug > Debug ArmAEMv8-A.
      3. Under Bare Metal Debug, in the Model parameters field, append the following parameter: `-C bp.secure_memory=false`

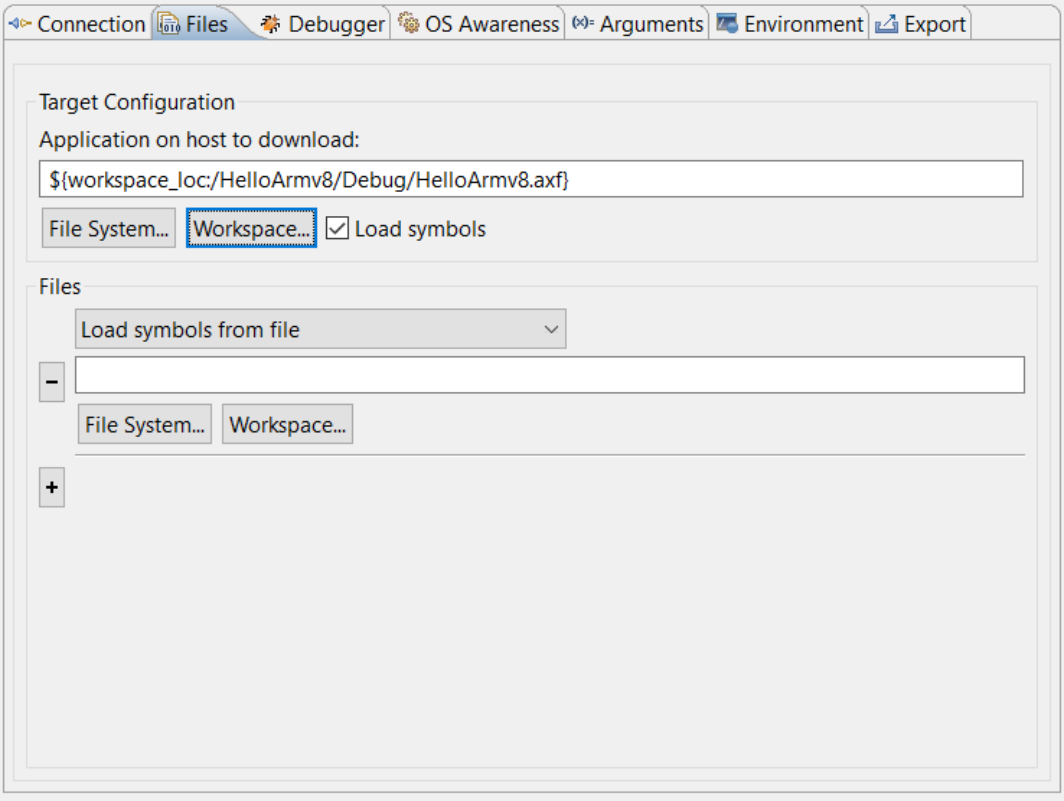
This parameter disables the TZC-400 TrustZone memory controller included in the `Base_AEMv8x1` FVP. By default, the memory controller disallows all accesses to DRAM memory.

Figure 5-3: Debug configurations platform selection



4. Select the Files tab, and:
   a. Under Target Configuration in the Application on host to download field, click Workspace.

Figure 5-4: Debug configurations Files Tab



The Workspace contains the HelloARMv8.axf application file you created when you built the Hello World project.

Note

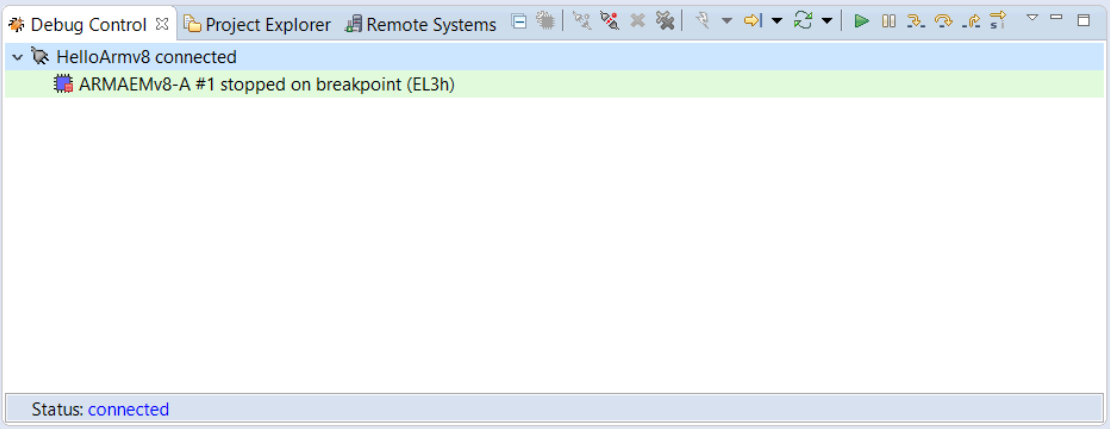
Ensure that the Load symbols option is selected.

Select `HelloARMv8.axf`.

b. Click OK to load the file.

5. Select the Debugger tab, and ensure the Debug from symbol option is selected and set to `main`.
6. Click Debug to load the application on the FVP, and load the debug information into the debugger.
7. In the Confirm Perspective Switch dialog that appears, click Yes. DS-5 connects to the FVP and displays the connection status in the Debug Control view.

Figure 5-5: DS-5 Debug Control view



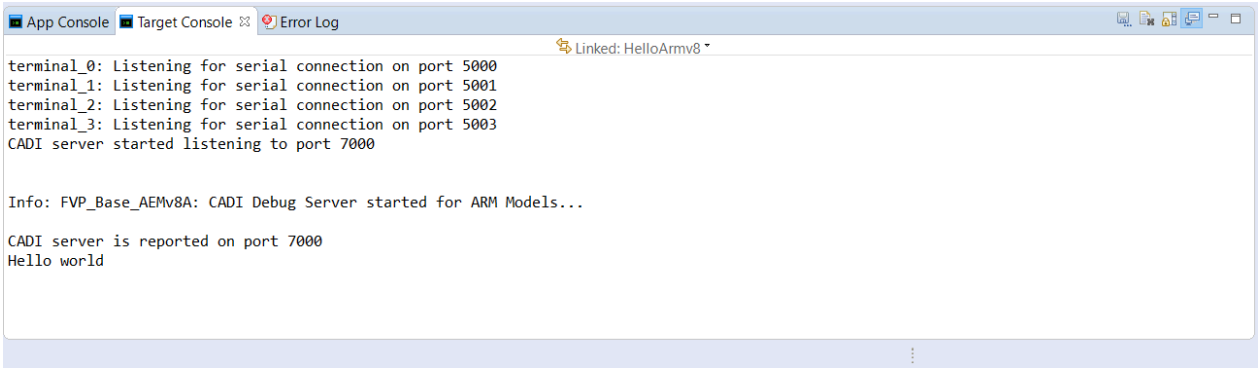
The application is loaded on the target and stopped at the `main()` function, ready to run.

6. Running Hello World

Click the Continue Button to continue running the application.

You can view the application output in the Target Console view.

Figure 6-1: Target Console output



ERROR(TAB180)

You may have noticed that on connecting, the Commands view displays the following error message:

```
ERROR(TAB180): The semihosting breakpoint address has not been specified
```

In a Armv8 FVP, running in either AArch64 or AArch32, you have to set a "semihosting trap" at a vector address in order for DS-5 Debugger to take control of the semihosting operation.

However, the Armv8 FVP is also carrying out semihosting, which is how you are able to see the Hello World"output in the Target Console view.

If you'd like to remove the TAB180 error message, then you need to disable DS-5 Debugger from attempting any semihosting. This can be done by creating a `.ds` debugger script containing:

```
set semihosting enabled off
```

This script needs to be included in the DS-5 debug configuration. Do this by checking the Run target initialization debugger script (.ds/.py) box in the Debugger tab and locating the script from your filesystem or workspace.

Figure 6-2: Arm Compiler 6 Hello World target initialization script