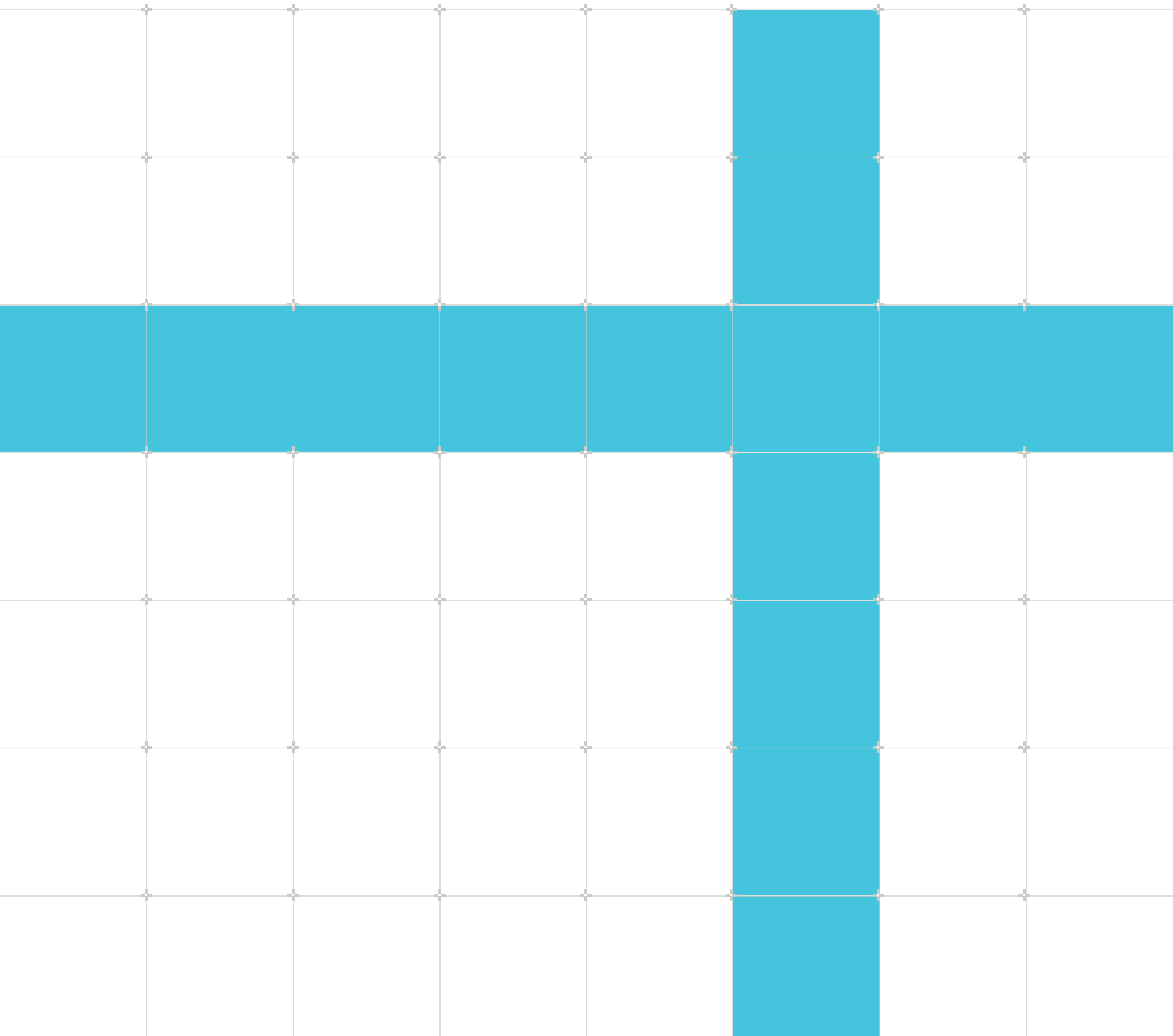
Linux Application Debugging Using DS-5

Version 1.0

Issue 02
102624_0100_02_en



Linux Application Debugging Using DS-5

Release information

Document history

Issue	Date	Confidentiality	Change
0100-02	1 January 2020	Non-Confidential	First version

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly

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

This tutorial takes you through the process of creating a simple Create a simple Hello World Linux application using C Linux application and then loading the application on a Cortex-A9 Fixed Virtual Platform (FVP) model running Arm embedded Linux. The Cortex-A9 Fixed Virtual Platform (FVP) model is provided with DS-5.

Prerequisites

This tutorial assumes that you have installed Arm DS-5 and acquired the license to use it. If not, use the Getting Started with Arm DS-5 tutorial to install DS-5 and acquire a license.

2. Create a simple Hello World Linux application using C

To create a Linux application using C in DS-5:

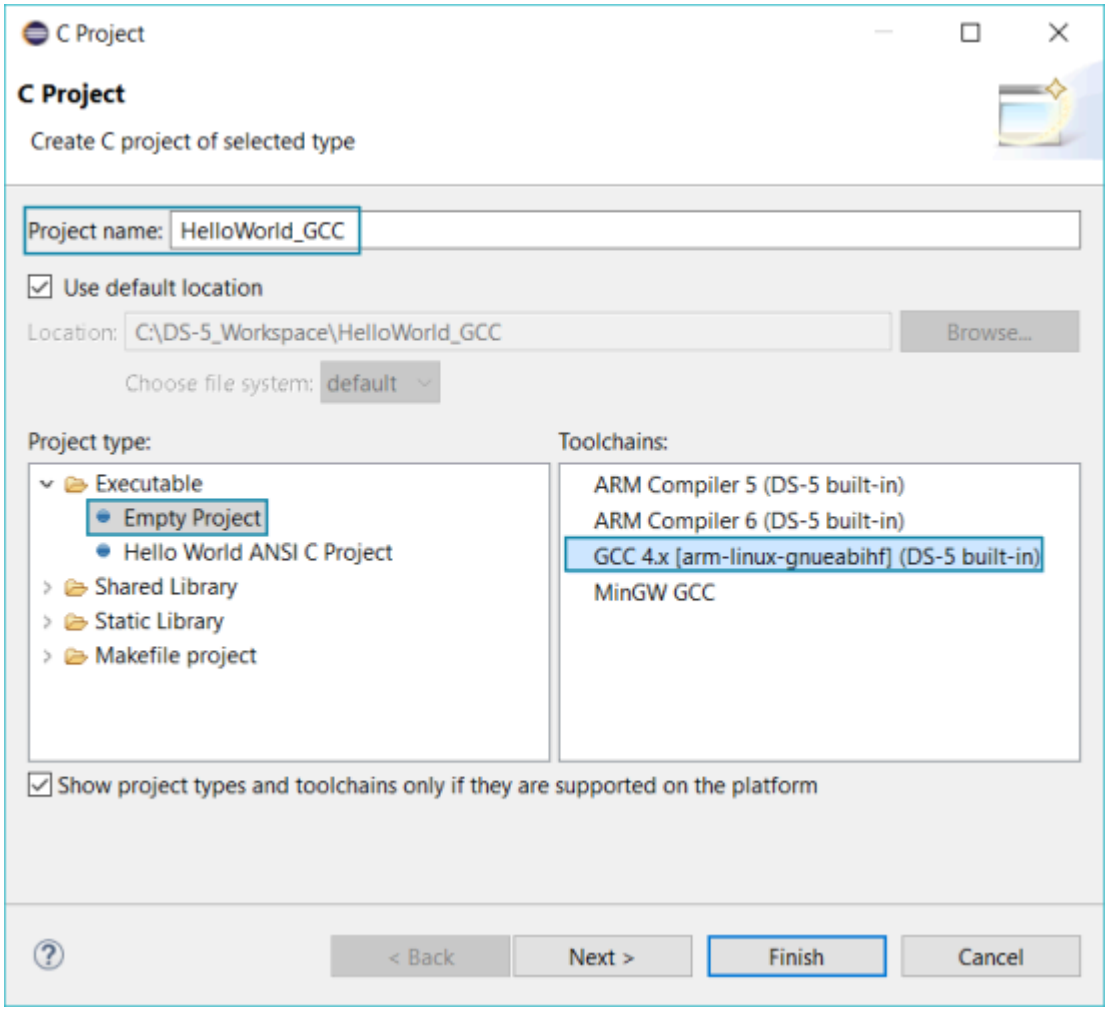
- Create a new C project and use the DS-5 GCC toolchain.
- Set up the DS-5 GCC toolchain compiler and linker options to build with the appropriate settings for Arm Embedded Linux running on a Fixed Virtual Platform (FVP) model.
- Create the source file and build it to create an application.

3. Create a new C project

The following steps describe how to create a new C project.

1. Start DS-5 and from the **DS-5 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 `HelloWorld_GCC` as the name of your project.
   b. Under **Project type**, select **Executable** > **Empty Project**.
   c. Under **Toolchains**, select the **GCC 4.x [arm-linux-gnueabihf] (DS-5 built in)** option.

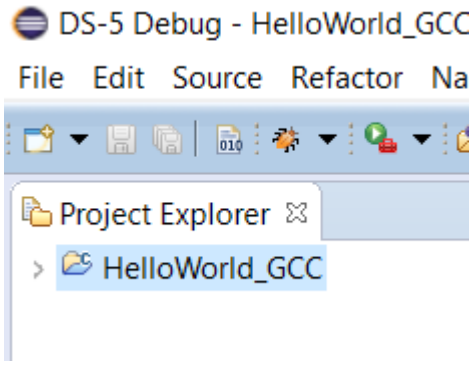
   Figure 3-1: Project type toolchain selection.



   d. Click **Finish** to create a C project called `HelloWorld_GCC`.

   You can view the project in the Project Explorer view.

Figure 3-2: Project Explorer view of newly-created project.



4. Configure the settings for new project

The following steps describe how to configure the settings for a new project.

1. In the **Project Explorer** view, right-click the **HelloWorld_GCC£ project** and select **Properties**.

You can also access the project properties from the main DS-5 menu. From the main menu, select **Project** > **Properties**.

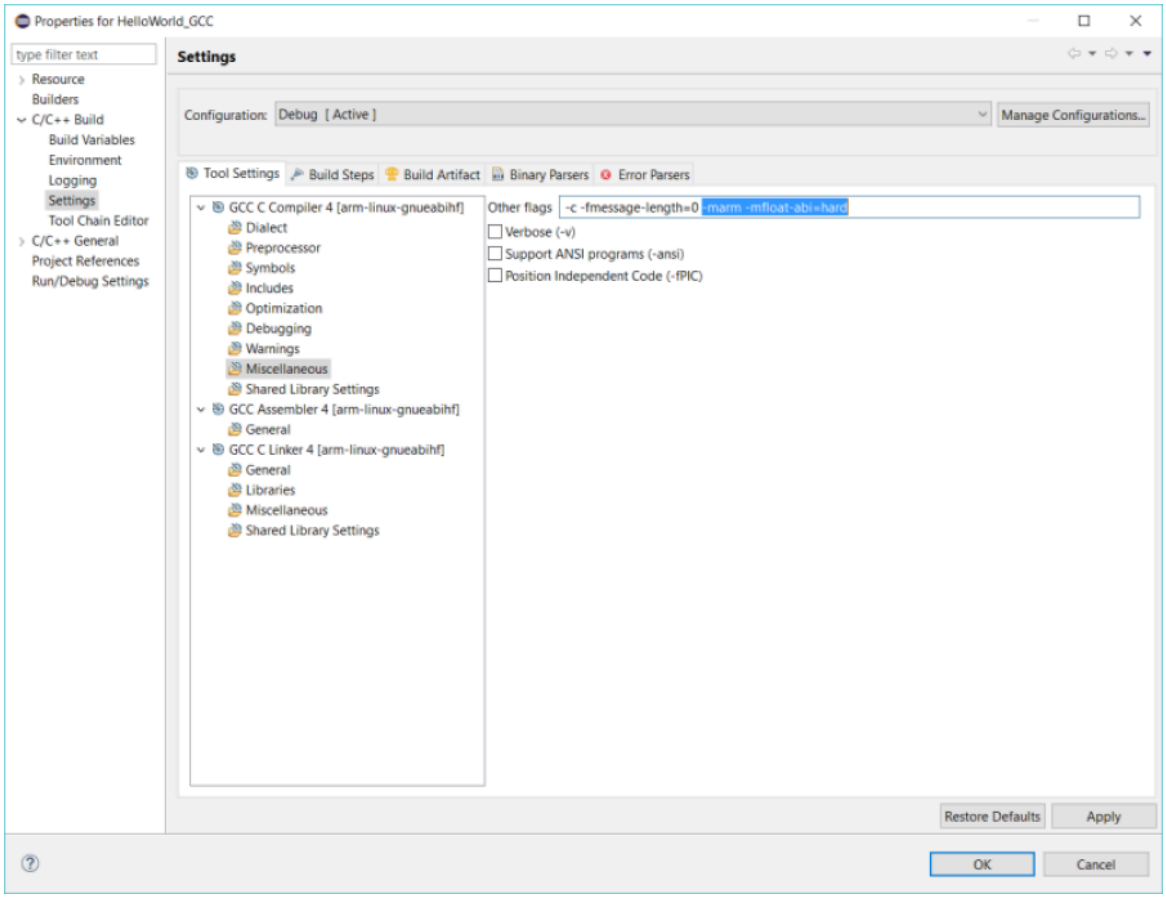
2. Select **C/C++ Build** > **Settings** > **Tool Settings** tab.
   a. You need to specify the relevant flags under **GCC C Compiler 4 [arm-linux-gnueabihf]** > **Miscellaneous** > **Other flags**:
      - DS-5 v5.21.1 and earlier support a soft-float file system, so enter:

```
-marm -march=armv4t -mfloat-abi=soft
```

      - DS-5 v5.22 and later support a hard-float file system, so enter:

```
-marm -mfloat-abi=hard
```

Figure 4-1: Screenshot of the Tool Settings tab.



These flags instruct the GCC compiler to compile a binary that is compatible with a particular architecture and file system. For more information about GCC compiler options for Arm, see: http://gcc.gnu.org/onlinedocs/gcc/ARM-Options.html

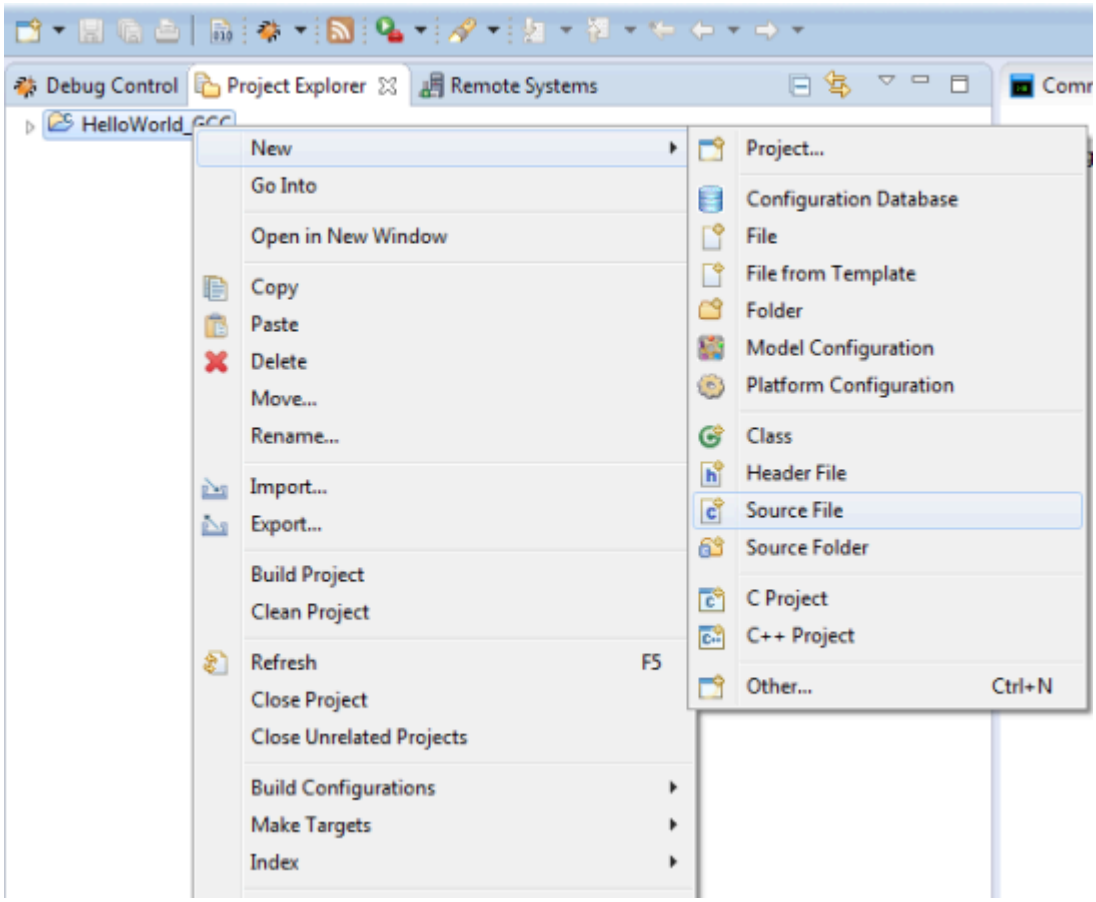
b. On the **Properties** for HelloWorld_GCC project dialog, click OK to apply the settings and close the dialog.

5. Create the source code and building the project

The following steps describe how to create the source code and build the project:

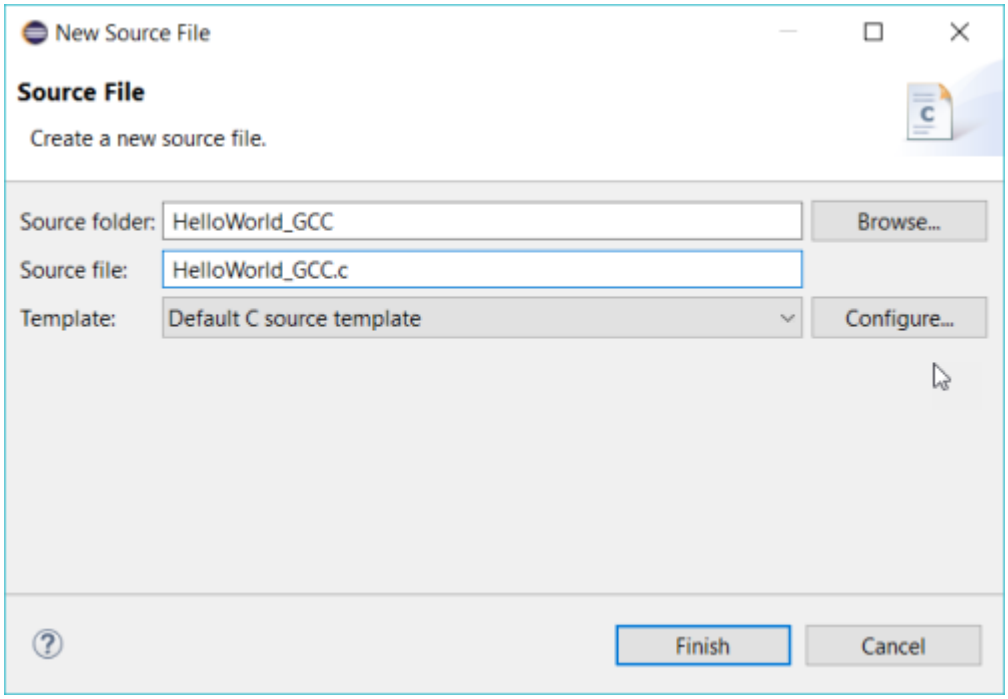
1. In the **Project Explorer** view, right-click the **HelloWorld_GCC** project and select **New** > **Source File**.

   Figure 5-1: New Source file.



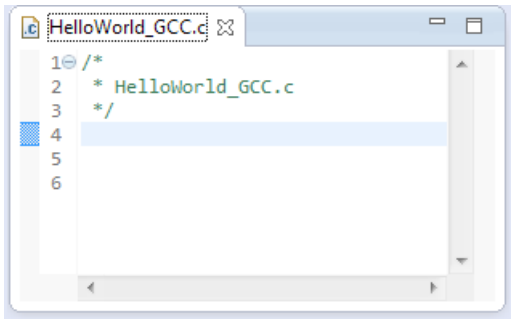
2. In the **New Source File** dialog, enter the file name `HelloWorld_GCC.c`.

Figure 5-2: Source file dialog box.



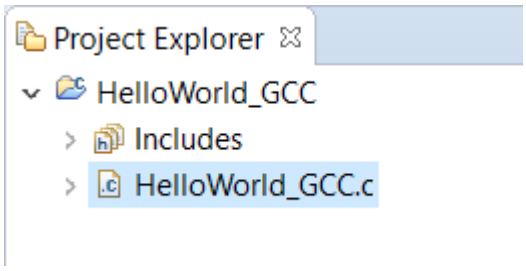
3. Click **Finish** to create the source file and open it in the code editing view.

Figure 5-3: Code Editing view.



The source file is also visible in the **Project Explorer** view, under the **HelloWorld_GCC project**.

Figure 5-4: Source file in Project Explorer view.



4. Add the following code to the new source file, and press **CTRL+S** to save it.

```
#include <stdio.h>

int main(int argc, char** argv)
{
      printf("Hello world\n");
      return 0;
}
```

What is `argc` and `argv`?

`argc` and `argv` are how command line arguments are passed to `main()` in C and C++.

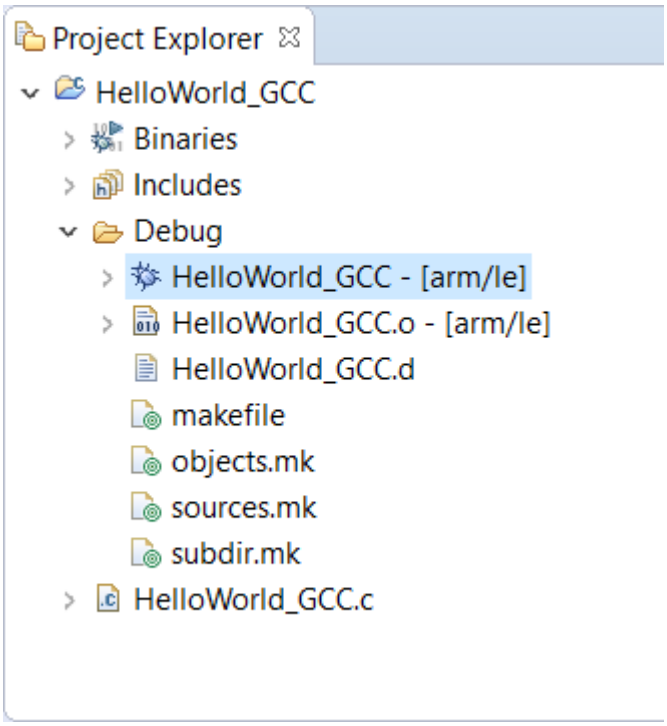
`argc` will be the number of strings pointed to by `argv`.

The variables are named `argc` (argument count) and `argv` (argument vector) by common convention.

5. In the **Project Explorer** view, right-click the **HelloWorld_GCC** project and select **Build Project**.

This creates the Linux executable and required support files.

Figure 5-5: Creating Linux executable and support files.



The items in the **Debug folder** are additional files required for debugging.

6. Debug the Linux application on a Fixed Virtual Platform (FVP) model

Once you have created the project and built the code, launch the debugger to run the application on one of the Fixed Virtual Platform (FVP) models provided with DS-5.

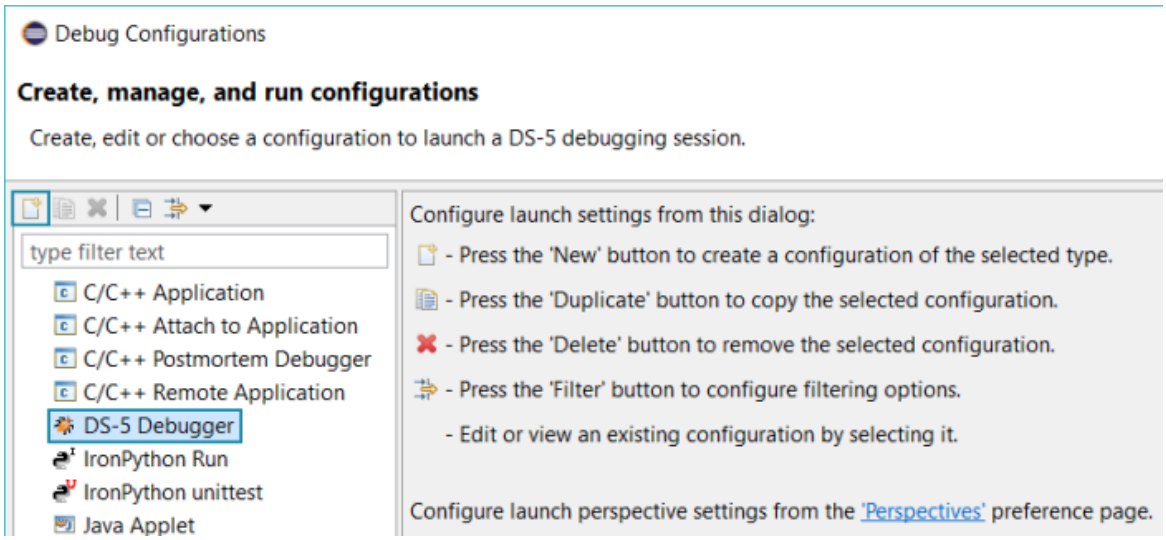
For this tutorial, we use the FVP_VE_Cortex-A9x4 model provided with DS-5.

7. Create a DS-5 debug configuration and connecting to an FVP model

These steps describe how to debug a DS-5 configuration and connect it to an FVP model.

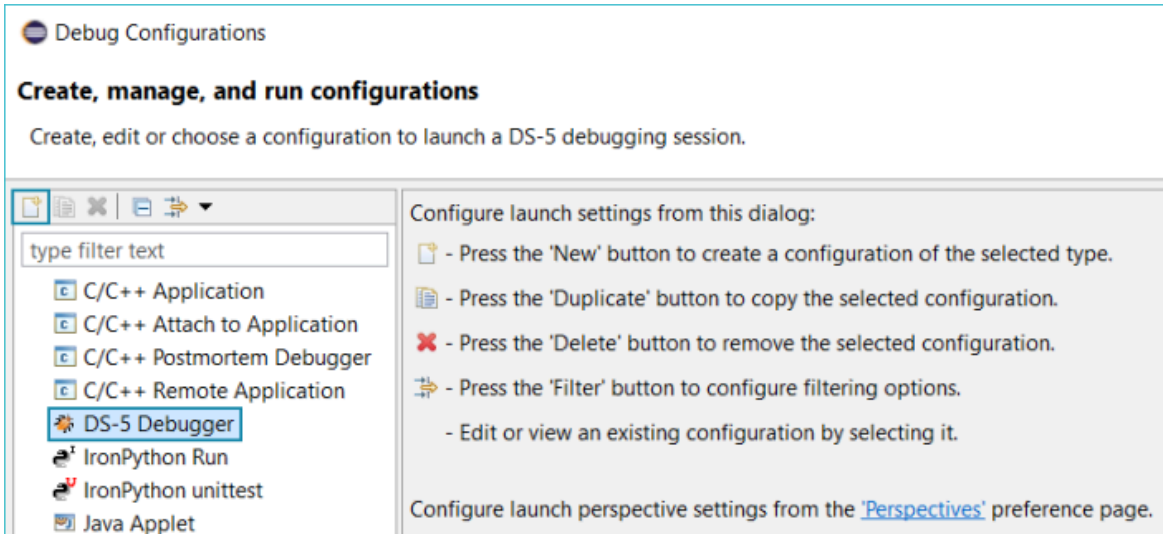
1. From the DS-5 main menu, select **Run** > **Debug Configurations**.
2. In the **Debug Configurations** dialog:
   a. Select **DS-5 Debugger**.
   b. Click the **New launch configurations** button.

Figure 7-1: Debug Configurations screen.



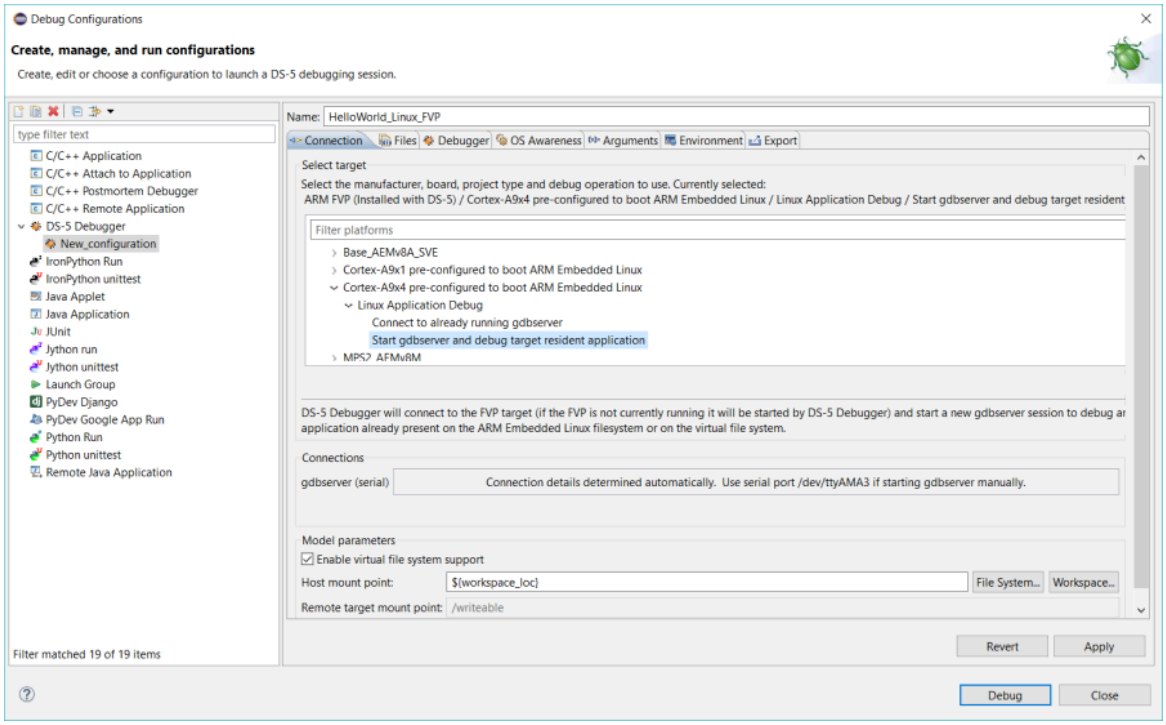
This creates a new DS-5 debug configuration and displays the various tabs required to specify settings for loading your application on the target.

Figure 7-2: New Debug Configuration all tabs.



3. On the Debug Configurations dialog:
   a. Give a name to the debug configuration. For example, `HelloWorld_Linux_FVP`.
   b. In the Connection tab, select **Arm FVP (Installed with DS-5)** > **Cortex-A9x4 pre-configured to boot Arm Embedded Linux** > **Linux Application Debug** > **Start gdbserver and debug target resident application**.

Figure 7-3: Debug configurations screen Connection tab.



By default, a relative path to your workspace location is specified in the **Host mount point field**. This location is used by the /writeable directory specified in the **Remote target mount point field**.

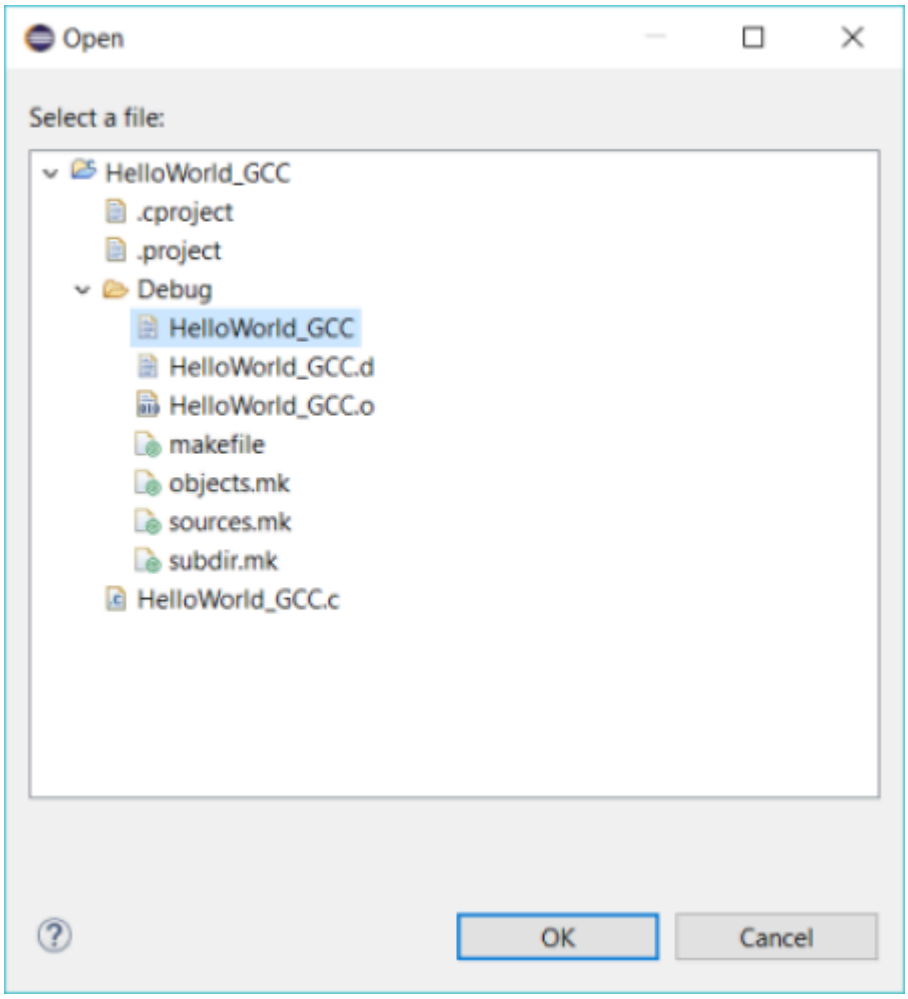
c. In the **Files tab**, and under **Target Configuration** > **Application on target field**, enter `/writeable/HelloWorld_GCC/Debug/HelloWorld_GCC`.

   This specifies that the **HelloWorld_GCC** application is available under the **/writeable/HelloWorld_GCC/Debug/** location on the target.

d. Under **Files**, select **Load symbols from file**, and click **Workspace**.

e. In the **Open** dialog, select the **HelloWorld_GCC** application in the **Debug** folder.

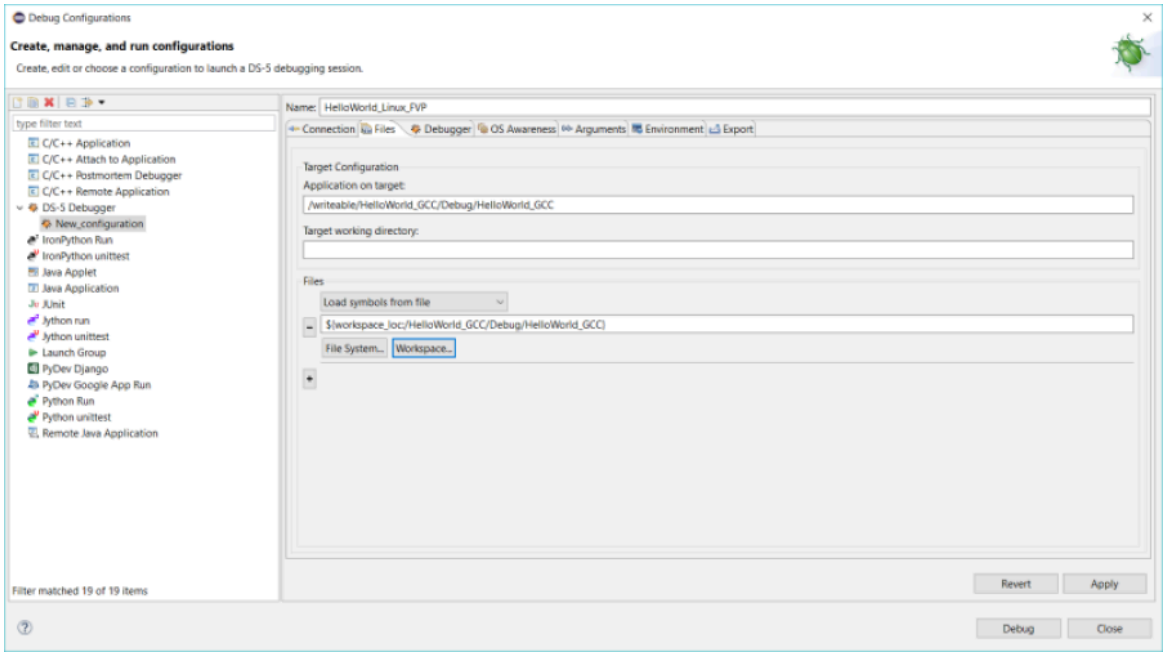
Figure 7-4: Open dialog box.



f. Click **OK**

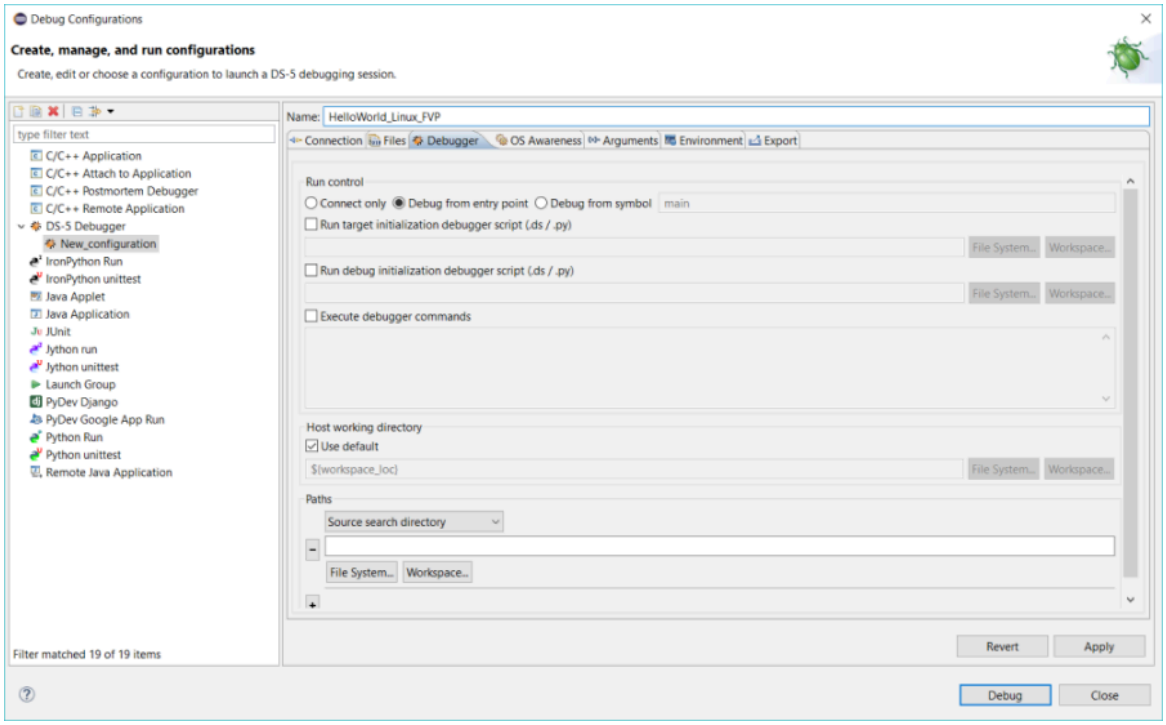
   This sets the path to the file that contains the required symbols information.

Figure 7-5: Debug configurations screen Files tab.



g. Select the **Debugger** tab, and select **Debug from entry point**.

Figure 7-6: Debug configurations screen Debugger tab.

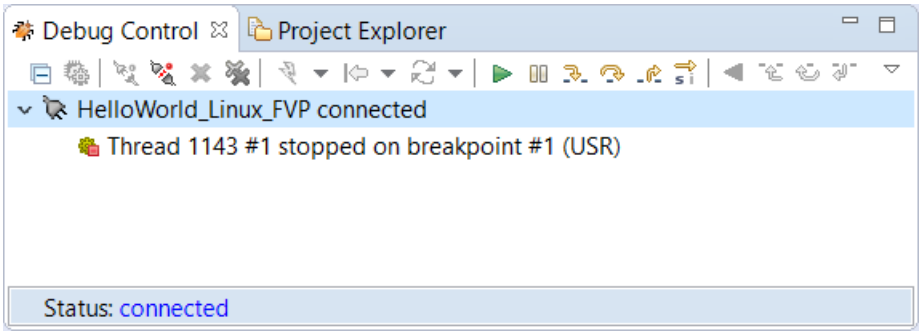


h. Click **Debug** to load the application on the target, and load the debug information into the debugger.

i. In the **Confirm Perspective Switch** dialog that appears, click **Yes**.

DS-5 connects to the FVP model, loads Linux on the FVP model, and displays the connection status in the **Debug Control** view.

Figure 7-7: Debug control view.



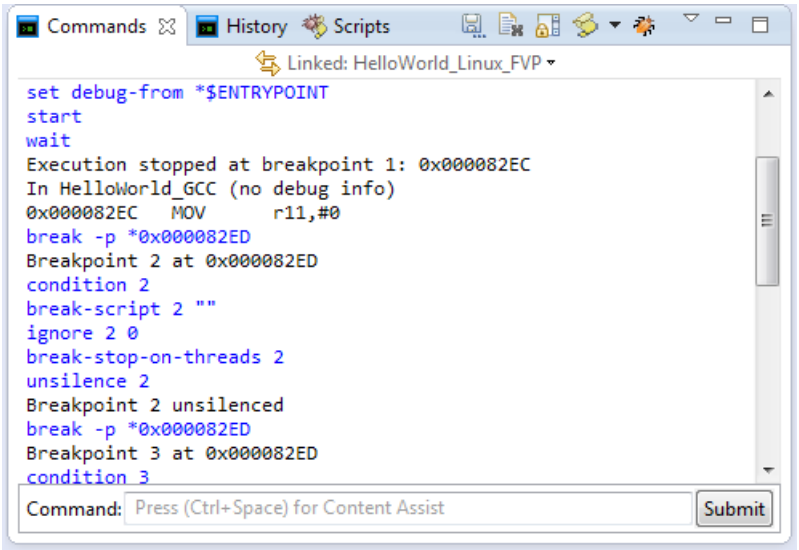
The application is loaded on the target, and has stopped at the entry point, ready to run.

Other views display information relevant to the debug connection.

For example:

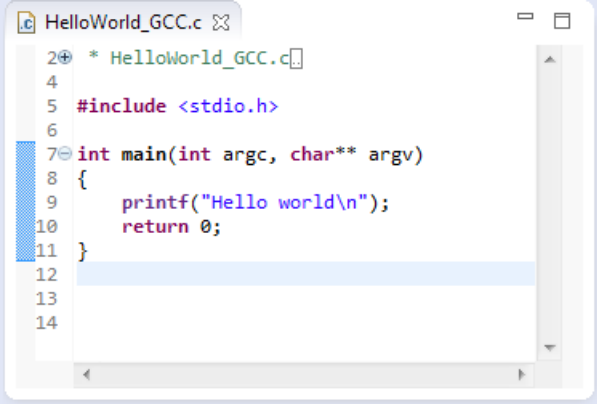
- The **Commands** view displays messages output by the debugger. Also use this view to enter DS-5 commands.

Figure 7-8: Commands view.



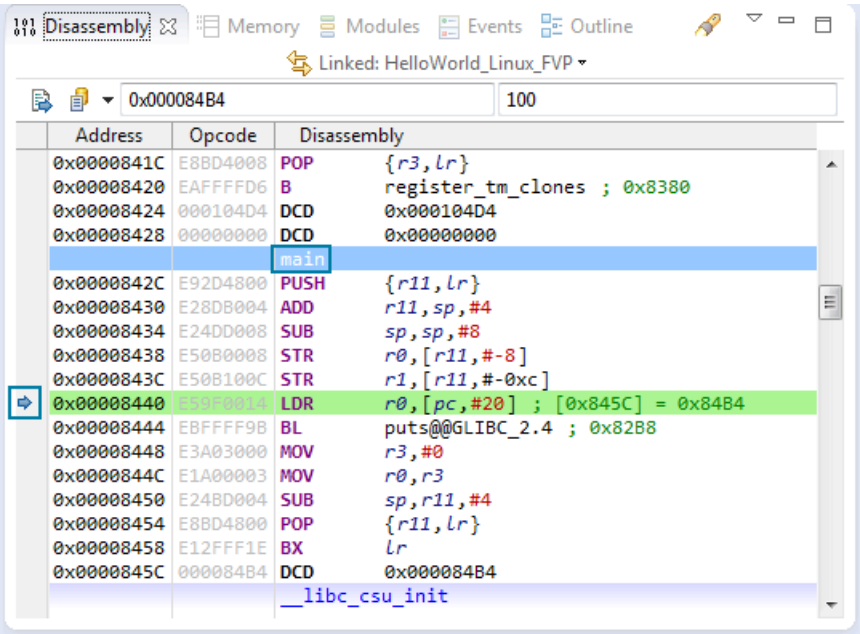
- The **C/C++ Editor** view shows the structure of the active C, C++, or makefile. The view is updated as you edit these files.

Figure 7-9: Editor view.



- The **Disassembly** view shows the loaded program in memory as addresses and assembler instructions.

Figure 7-10: Disassembly view.

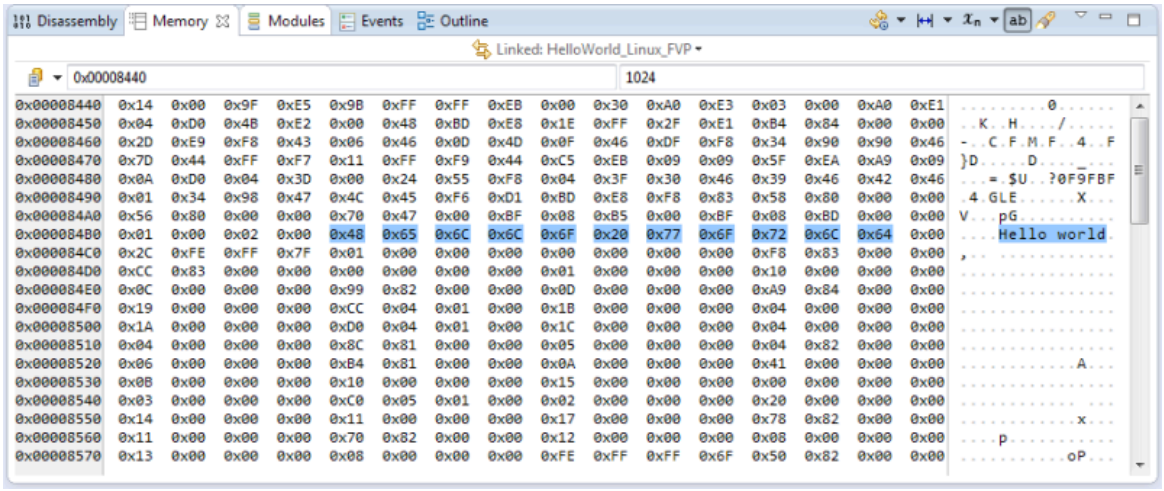


Indicates the location in the code where your program is stopped. In this case, it is at the `main()` function. The view shows other information that enables you to drill down into the details of the code.

- The **Memory** view shows how code is represented in the target memory. For example, to view how the string `Hello World` from the application is represented in memory:
  1. Open the **Memory** view.
  2. In the **Address** field enter, `0x00008440` and press **Enter** on your keyboard. The view displays contents of the target's memory.

3. Select and highlight the words `Hello World` to see their ASCII equivalents.

Figure 7-11: Memory view.

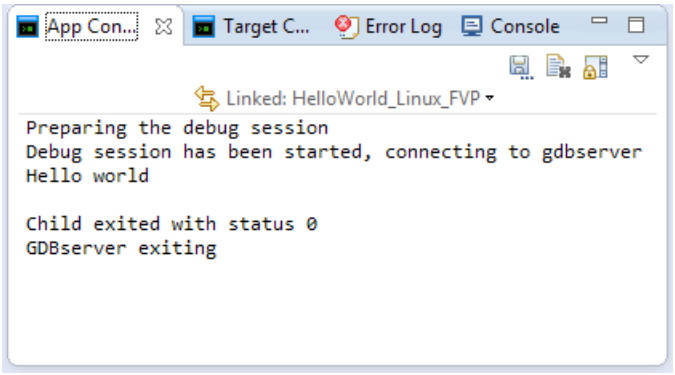


The **Memory** view displays the hexadecimal values for the code, and also the ASCII character equivalent of the memory values.

-  to run the application.

You can view the application output in the **App Console** view.

Figure 7-12: App console view.



8. Step Through the Application

Use the controls provided in the Debug Control view to step through the application.

Figure 8-1: Step through controls in Debug Control.

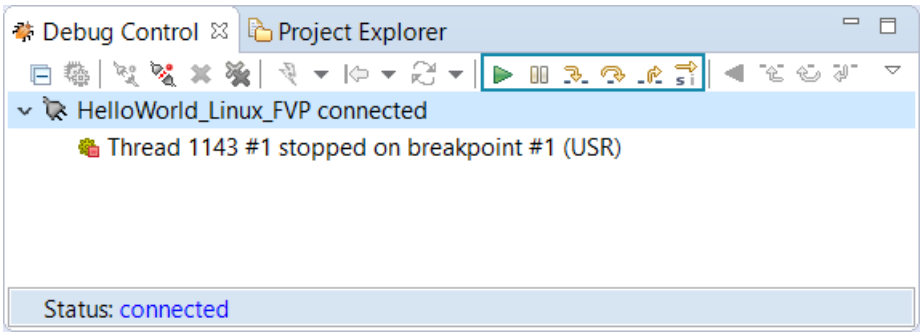


Figure 8-2: Continue Button

Click to continue processing code.

Figure 8-3: Pause Button

Click to interrupt or pause processing code.

Figure 8-4: Step Through button

Click to step through the code.

Figure 8-5: Step Over button

Click to step over source line.

Figure 8-6: Step Out button

Click to step out.

Figure 8-7: Step instruction

This is a toggle. Select this if you want the above controls to step through instructions.

9. Disconnect from the debug connection

To disconnect from a debug connection, you can do one of the following:

- Right-click the connection and select **Disconnect from Target**
- Select the connection and in the **Debug Control** view toolbar click 
- Double-click on the selected connection.