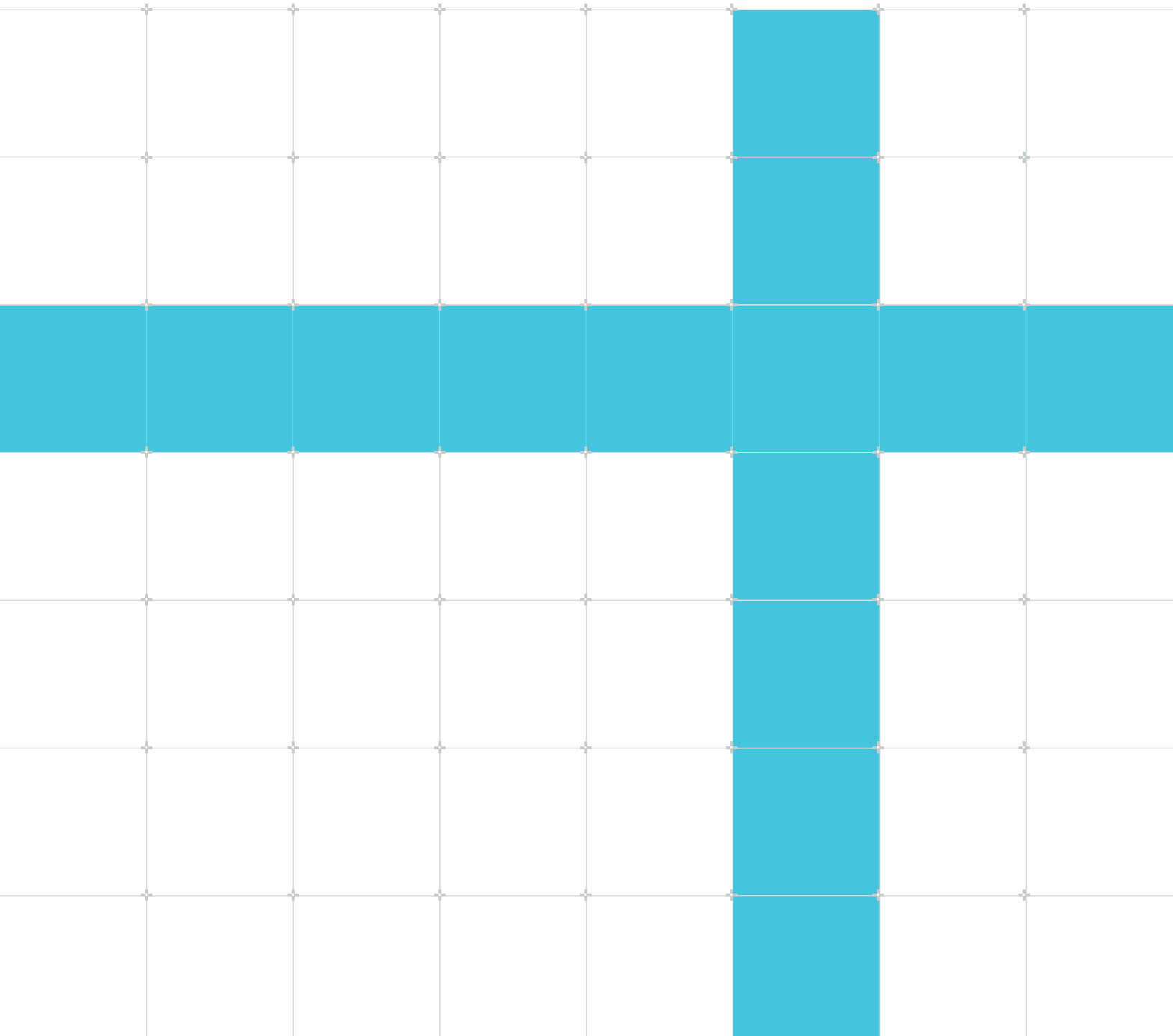
Linux Application Debugging using Arm DS

Version 1.0

Issue 01
102583_0100_01_en



Linux Application Debugging using Arm DS

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	22 July 2021	Non-Confidential	First release

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 Hello World 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 an Arm Development Studio (Arm DS) all editions.

This tutorial is only applicable to versions up to Arm DS 2019.0 as future releases will not include pre-configured models to boot Arm Embedded Linux.

2. Creating a Simple Hello World Linux Application using C

This tutorial assumes that you have installed an Arm DS and acquired the license to use it. If not, go to Arm Development Studio to install DS and acquire a license.

This example is intended to be built with Linaro arm-linux-gnueabihf GCC. If you wish to modify and rebuild the example, you must have Linaro arm-linux-gnueabihf GCC installed. Linaro arm-linux-gnueabihf GCC is not supplied with Development Studio, but can be downloaded from GNU Toolchain.

To create a Linux application using C in Arm DS:

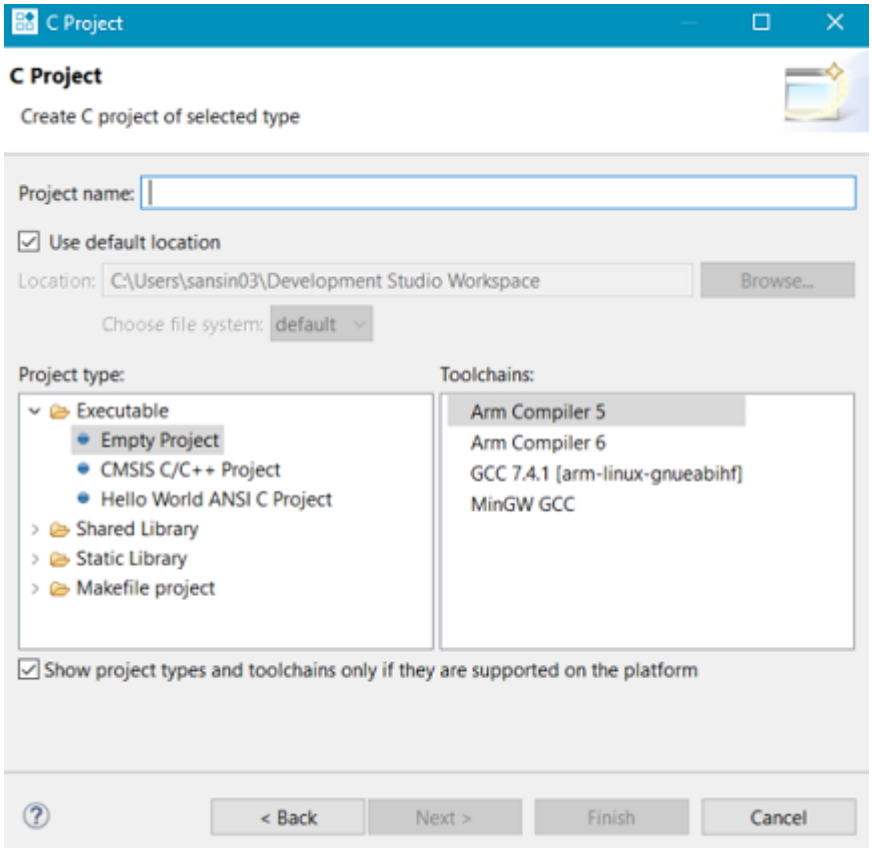
1. Create a new C project and use the GCC toolchain.
2. Set up the GCC toolchain compiler and linker options to build with the appropriate settings for Arm Embedded Linux running on a Fixed Virtual Platform (FVP) model.
3. Create a source file and build it to create an application.

3. Creating a New C Project

To create a new C project:

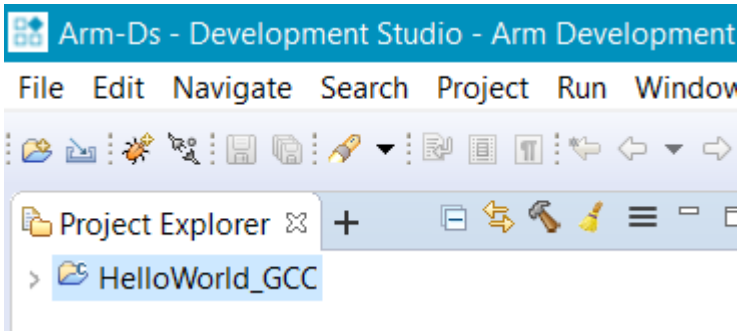
1. In the Project name field, enter `HelloWorld_GCC` as the name of your project.
2. Under Project type, select Executable > Empty Project.

Figure 3-1: C Project window



3. Under Toolchains, select the GCC x.x [arm-linux-gnueabihf] option. You must install this toolchain to proceed further.
4. Click Finish to create a C project called `HelloWorld_GCC`. You can view the project in the Project Explorer view.

Figure 3-2: The created project in the Project Explorer view

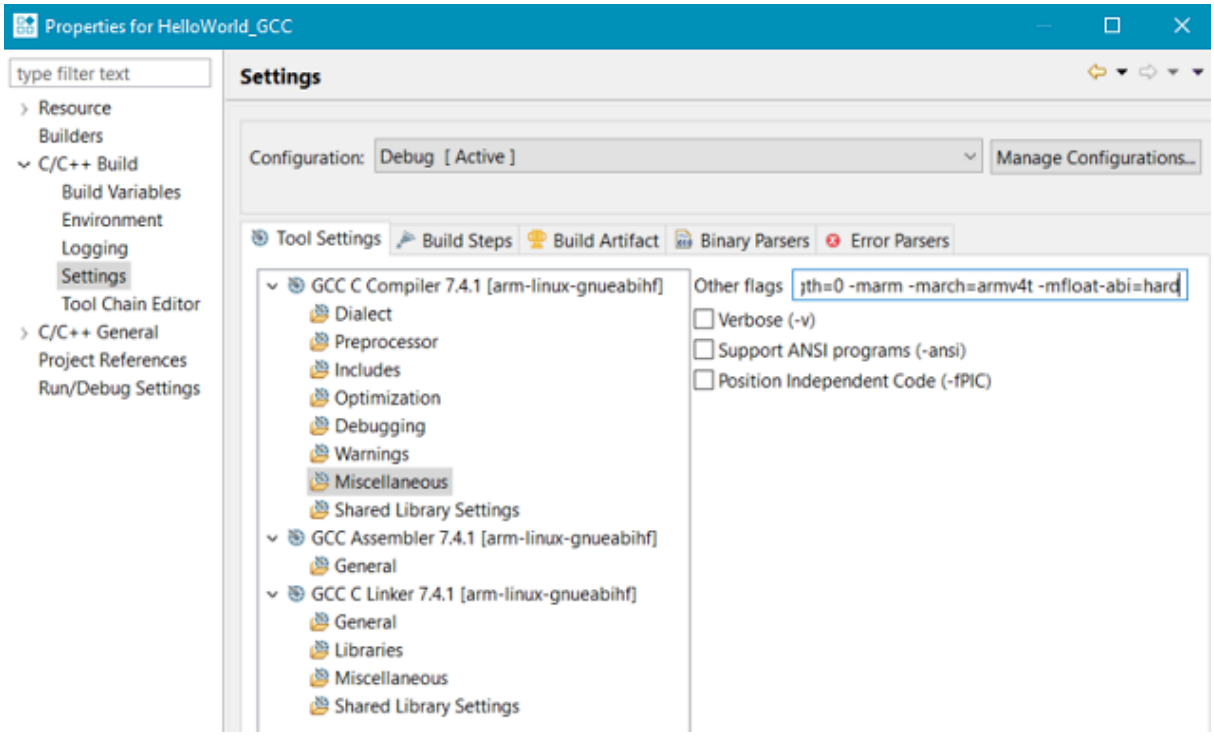


4. Configuring the settings for a new project

To configure setttings 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 Arm DS menu.
2. From the main menu, select Project > Properties.
3. Select the C/C++ Build > Settings > Tool Settings tab.
4. Specify the relevant flags under GCC C Compiler 4 [arm-linux-gnueabihf] > Miscellaneous > Other flags
5. Arm DS and later supports a hard-float file system, so enter `-marm -mfloat-abi=hard` as shown in the following screenshot:

Figure 4-1: Project settings window



These flags instruct the GCC compiler to compile a binary that is compatible with a particular architecture and file system.

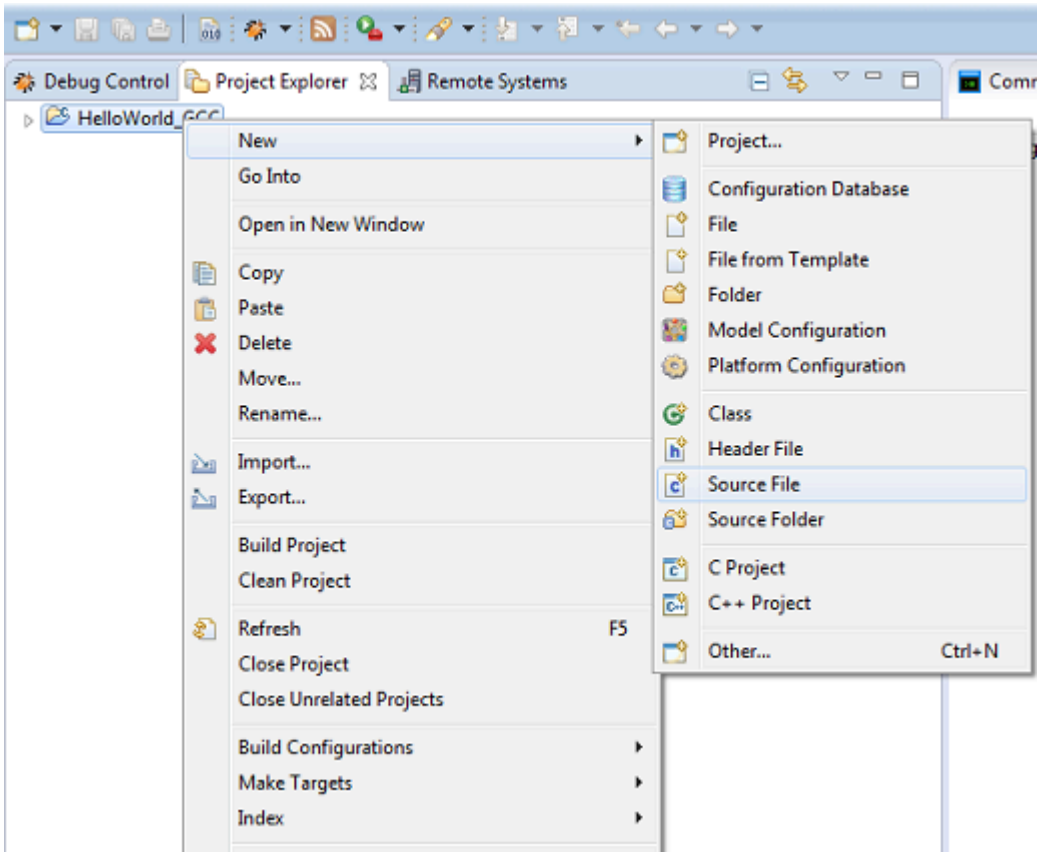
6. On the Properties for HelloWorld_GCC project dialog, click OK to apply the settings and close the dialog.

5. Creating the source code and building the project

To create the source code and build the project:

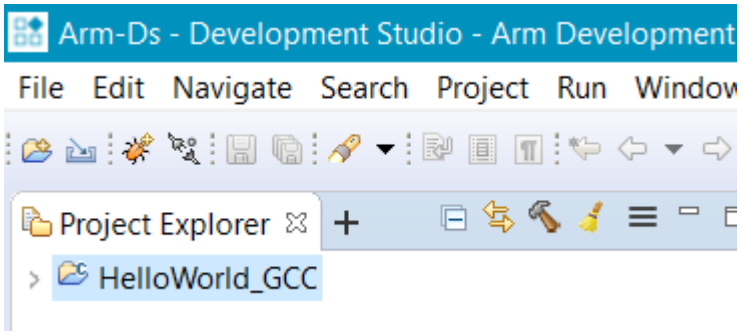
1. In the Project Explorer view, right-click the HelloWorld_GCC project and select New > Source File as shown in the following screenshot:

Figure 5-1: add a new source file



2. In the New Source File dialog, enter the file name `HelloWorld_GCC.c`.
3. Click Finish to create the source file and open it in the code editing view.

Figure 5-2: The location of the Hello World project in Arm DS



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

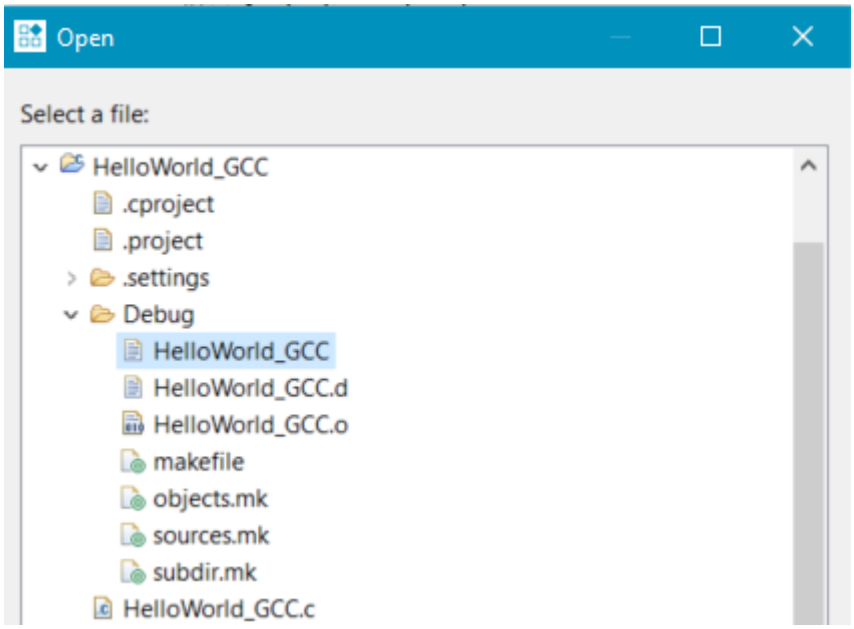
```
#include <stdio.h>

int main(int argc, char** argv)
{
      printf("Hello world\n");
      return 0;
}
</stdio.h>
```

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.

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

Figure 5-3: Hello World project in the Debug folder



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 Arm DS.

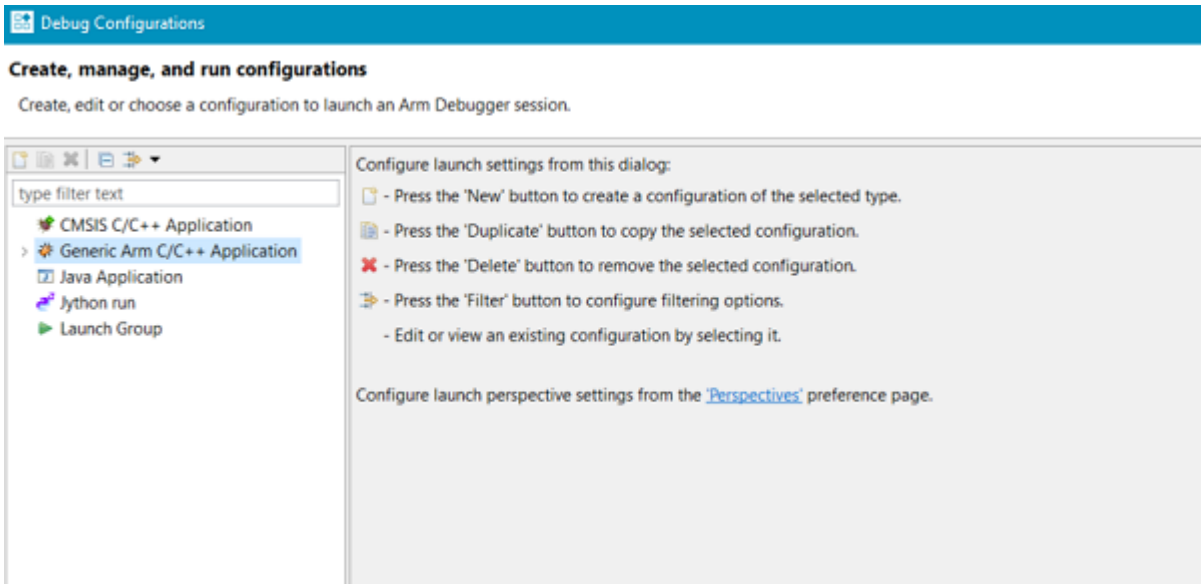
For this tutorial, we use the FVP_VE_Cortex-A9x4 model provided with Arm DS.

7. Creating an Arm DS debug configuration and connecting to a FVP model

To create a debug configuration and connect to FVP:

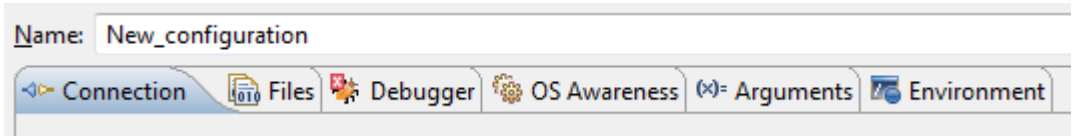
1. From the Arm DS main menu, select Run > Debug Configurations.
2. In the Debug Configurations dialog, select Generic Arm C/C++ Application from the options in the Debug Configuration window as shown:

Figure 7-1: Debug Configurations window



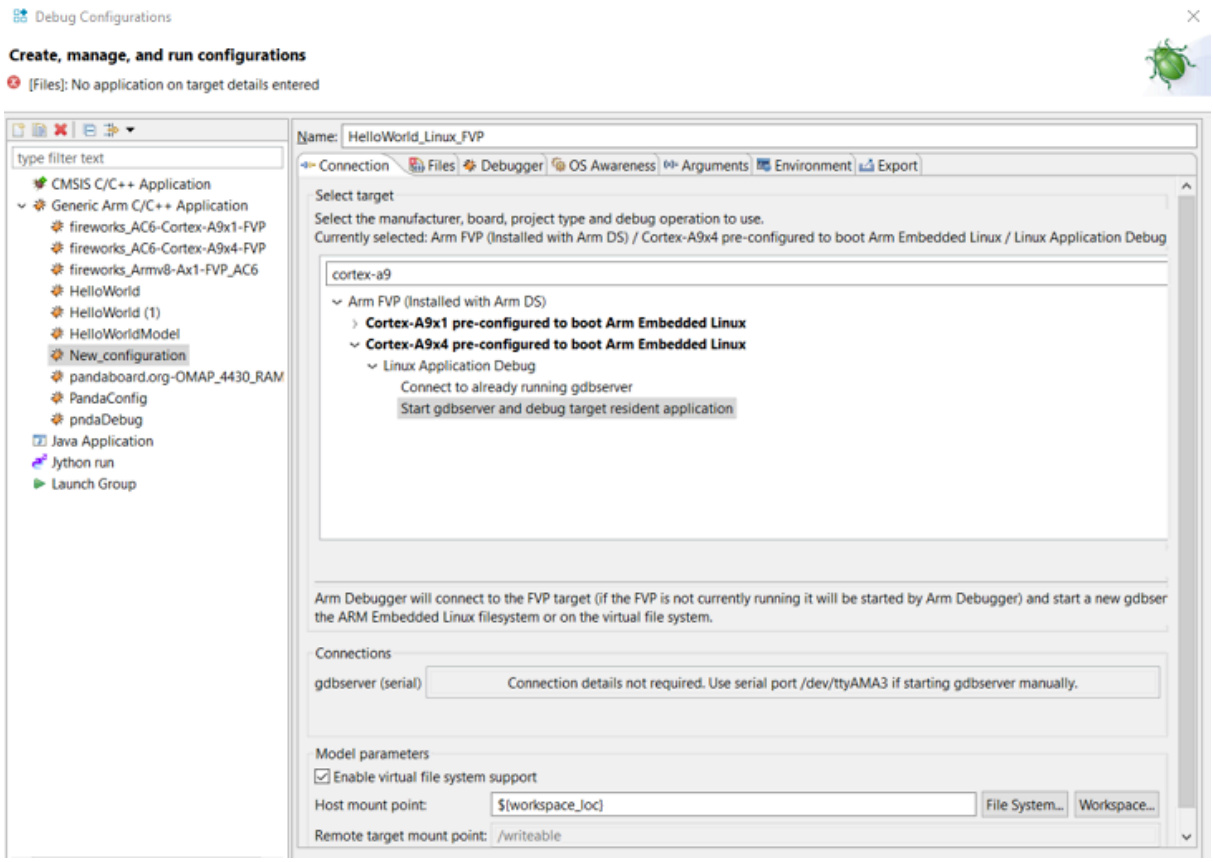
3. Select Press the New button to create a configuration of the selected type. This creates a new Arm DS debug configuration and displays the various tabs required to specify settings for loading your application on the target, as shown in the following screenshot:

Figure 7-2: Name field



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

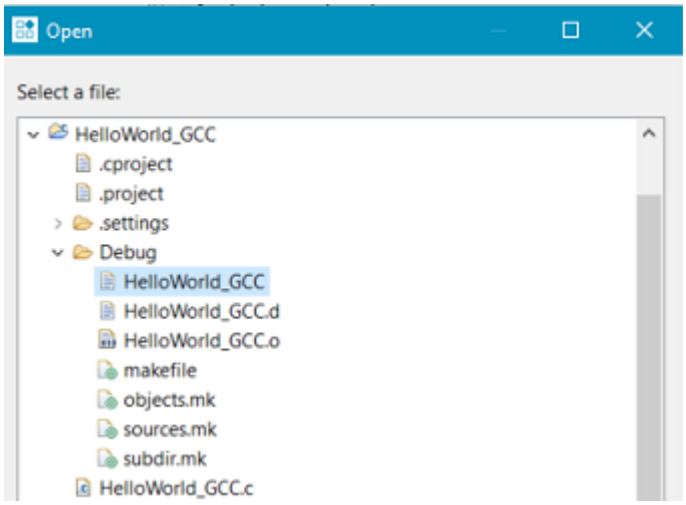
Figure 7-3: Connection tab



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.

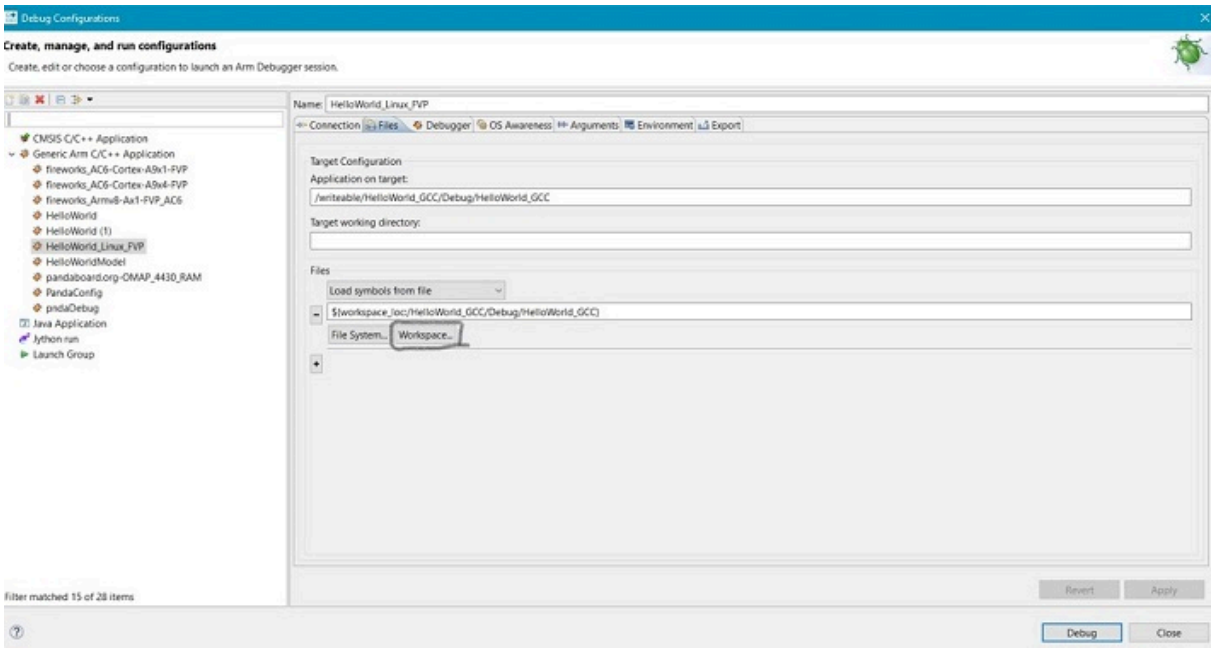
6. In the Files tab, and under Target Configuration > Application on target field, enter `/writeable/HelloWorld_GCC/Debug/HelloWorld_GCC`. This specifies that the HelloWorld_GCC application is available under the/writeable/HelloWorld_GCC/Debug/ location on the target.
7. Under Files, select Load symbols from a file, and click Workspace.
8. In the Open dialog, select the HelloWorld_GCC application in the Debug folder:

Figure 7-4: Open dialog



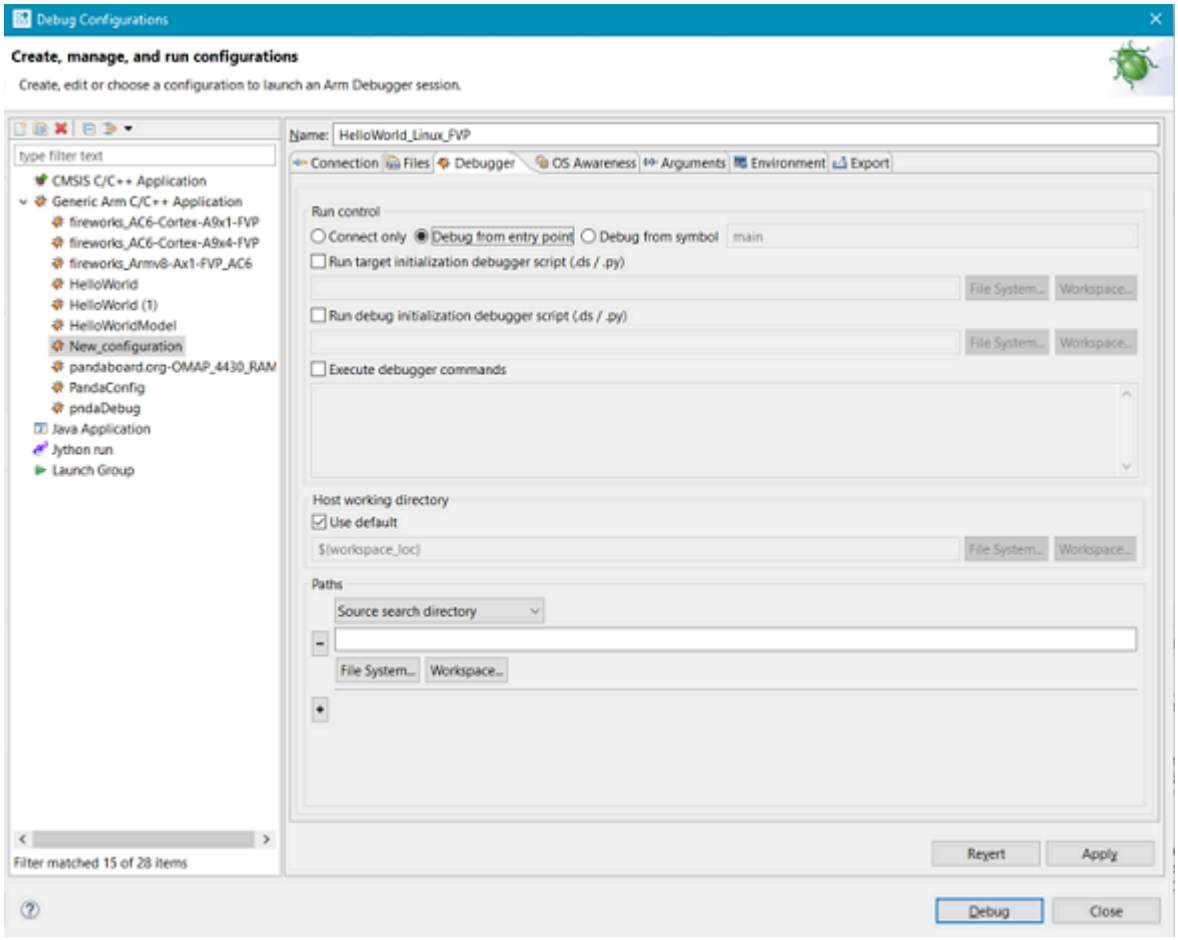
9. Click OK. This sets the path to the file that contains the required symbols information:

Figure 7-5: Debug Configuration window



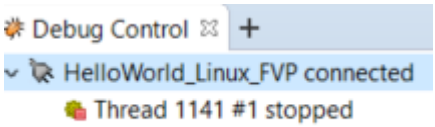
10. Select the Debugger tab, and select Debug from the entry point.

Figure 7-6: Debugger tab



11. Click Debug to load the application on the target, and load the debug information into the debugger.
12. In the Confirm Perspective Switch dialog that appears, click Yes. Arm DS-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



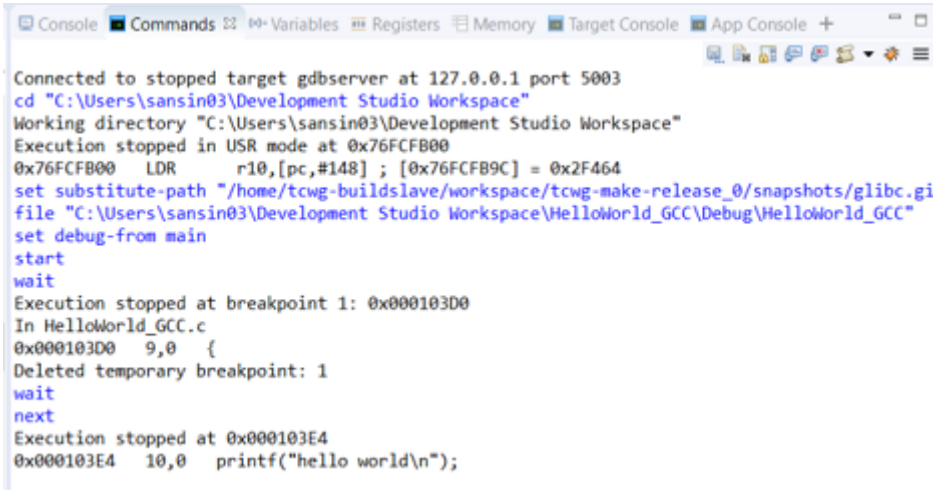
The application is loaded on the target and has stopped at the entry point, ready to run.

8. Debug views

Once the debug connection is established some of the display information relevant to the debug connection are:

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

Figure 8-1: Commands tab

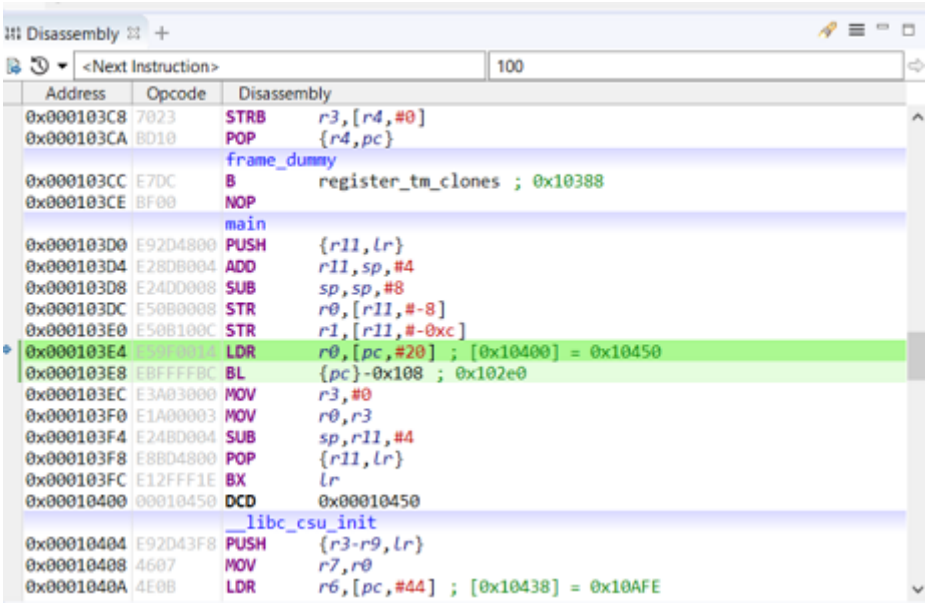


The C/C++ Editor view shows the structure of the active C, C++, or makefile. The view is updated as you edit these files.

Figure 8-2: C Editor view

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

Figure 8-3: Disassembly tab



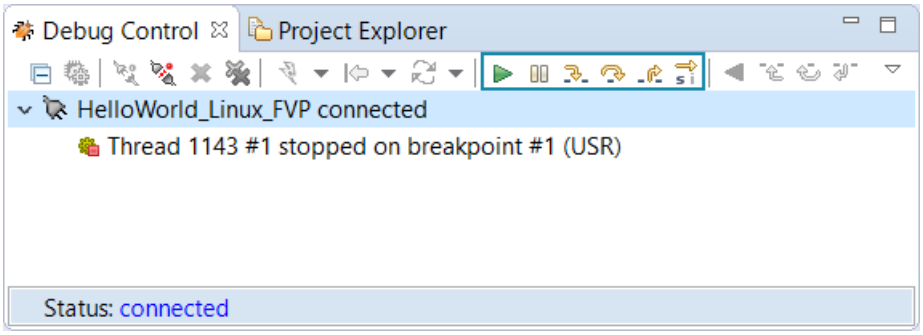
Green line indicates the location in the code where your program is stopped. In this case, it is at the `main()` function. Light Green line indicates the next instruction to be executed.

Click Continue to run the application. You can view the application output in the App Console view.

9. Stepping through the application

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

Figure 9-1: Debug Control options



- Click Continue to continue processing code.
- Click Pause to interrupt or pause processing code.
- Click Step Through to step through the code.
- Click Step Over to step over source line.
- Click Step Out to step out.
- Use the toggle if you want the above controls to step through instructions.

10. Disconnecting from the debug connection

To disconnect from the debug connection:

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