

ARM DS-5 Tools and Avnet ZED Series

#3

Creating a Debug Connection using ARM DStream and Avnet ZedBoard or MicroZed



Sept 2013
Version 01

Table of Contents

ARM DS-5 Tools and ZedBoard Series	3
Required Installations	4
Technical Support	5
Connecting a DS-5 Debug Session to ZedBoard.....	6
Create a C Project in DS-5	6
Debug a C Program in DS-5	11
Revision History	18
Resources	18

ARM DS-5 Tools and ZedBoard Series

This tutorial is one in a series of step by step instruction manuals. Together they document the procedures necessary to utilize the ARM Development Studio 5 (DS-5™) Software Suite and the DSTREAM Debugging tools with the Avnet Zynq Evaluation and Development (ZED) boards. These tutorials can be used on their own, or in combination with Avnet online videos and OnRamp Technical Session™.

The ARM software and hardware tools provide a powerful debugging suite for processor-based systems built around the dual Cortex-A9 cores present in the Xilinx Zynq SoC, at the heart of the Avnet ZED boards. A Linux software developer can simultaneously debug applications and kernel module code, with separate control over each thread. You can step through Linux boot code, first stage bare metal boot code, and bare metal applications. When used in concert with the Xilinx Vivado tools for FPGA fabric development, the ARM debugger and Internal Logic Analyzer (ILA) IP can be cross-triggered to stop on software and hardware breakpoints, or when a hardware event occurs. For difficult-to-isolate intermittent faults, DS-5 provides access to the Cortex-A9 on-chip Trace facility. Once your embedded system is running correctly, DS-5 uses Streamline, a graphical system profiler, to identify performance bottlenecks in your design to ensure top-shelf operation.

This tutorial series begins with the most basic tool configuration and board connection. It takes you all the way through to the most complex aspects of hardware/software co-debugging to root out design errors that are otherwise apparent only in very complex use cases, or worse, after a product is released. Together the ARM DS-5 tools, Xilinx Vivado and Avnet ZED boards provide an unparalleled combination to compress design timelines, cut project costs and optimize your product for the marketplace.

Required Installations

Software

The recommended software for this tutorial series is:

- ARM Development Studio 5 (Exact version used is 5.14, build 1702)
- Xilinx ISE WebPACK 14.5 (Free license and download from Xilinx website)
- Cypress CY7C64225 USB-to-UART Bridge Driver (for ZedBoard serial output)
- Silicon Labs CP2104 USB-to-UART Bridge Driver (for MicroZed serial output)
- Tera Term (Exact version used is V4.75)
- Xilinx Software Development Kit, version 14.5
- For hardware/software co-debugging, Xilinx Vivado 2013.2

Hardware

The targeted hardware consists of the following:

- PC workstation with at least 5 GB RAM, 30GB free hard disk space, Windows 7 64-bit operating system, and a wired GB Ethernet connection
- Available SD card slot on PC or external USB-based SD card reader
- One of:
 - Avnet ZedBoard Kit (**AES-Z7EV-7Z020-G**)
 - USB cable (Type A to Micro-USB Type B)
 - 4GB SD card
 - 12v Power supply
 - Avnet MicroZed Kit (**AES-Z7MB-7Z010-G**)
 - USB cable (Type A to Micro-USB Type B)
 - 4GB SD card
- Avnet ZedBoard Debug Adapter Kit (**AES-ZBDB-ADPT-G**)
 - 14-pin Xilinx PC4 ribbon cable
- ARM DSTREAM unit and Keil pod with wide cable connector
 - 20-pin JTAG ribbon cable
 - USB cable (Type A to Printer)
 - 5v Power supply
- CAT-5 Ethernet cable

Technical Support

For technical support with any of the instructions, please contact your local Avnet/Silica FAE or visit the support forums:

<http://www.zedboard.org/forum>

<http://www.microzed.org/forum>

Additional technical support resources are listed below.

ZedBoard Kit/MicroZed Kit support page with Documentation and Reference Designs

<http://www.zedboard.org/content/support>

<http://www.microzed.org/content/support>

For Xilinx technical support, you may contact your local Avnet/Silica FAE or Xilinx Online Technical Support at www.support.xilinx.com . On this site you will also find the following resources for assistance:

- Software, IP, and Documentation Updates
- Access to Technical Support Web Tools
- Searchable Answer Database with Over 4,000 Solutions
- User Forums
- Training - Select instructor-led classes and recorded e-learning options

Contact your Avnet/Silica FAE or Avnet Support for any additional questions regarding the reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design.

<http://www.em.avnet.com/techsupport>

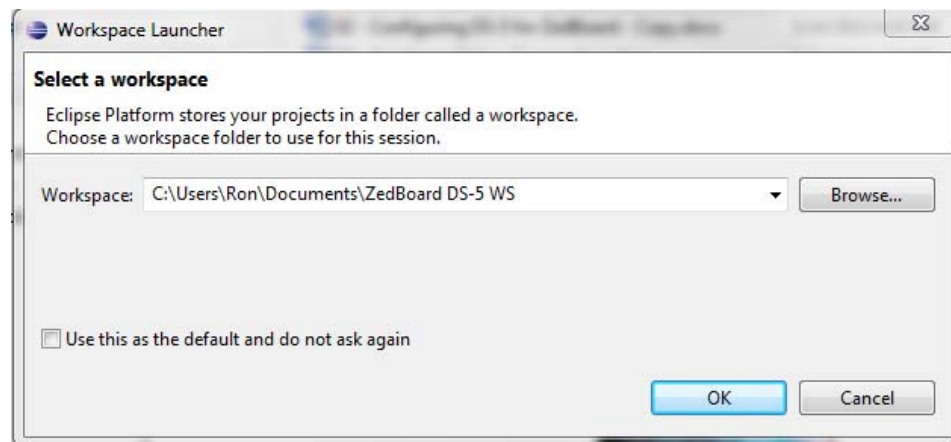
For ARM technical support, you may contact your local Avnet/Silica FAE or ARM Online Technical Support at www.arm.com/support .

Connecting a DS-5 Debug Session to Avnet ZED Boards

In this tutorial we will create a standalone bare metal project in the DS-5 IDE, and use the USB-JTAG connection via the DSTREAM to download the application to one of the ARM Cortex-A9 cores on the target.

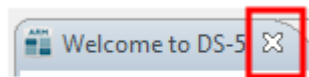
Create a C Project in DS-5

1. Open the DS-5 IDE on your host PC. Select a workspace in any folder where your userID has full access permissions. By default, DS-5 will select a folder within the Documents folder under your userID. You can use this folder, or select a different name as you wish. For the purposes of this tutorial, we will create an empty workspace called **ZedBoard DS-5 WS**.



Select a DS-5 Workspace

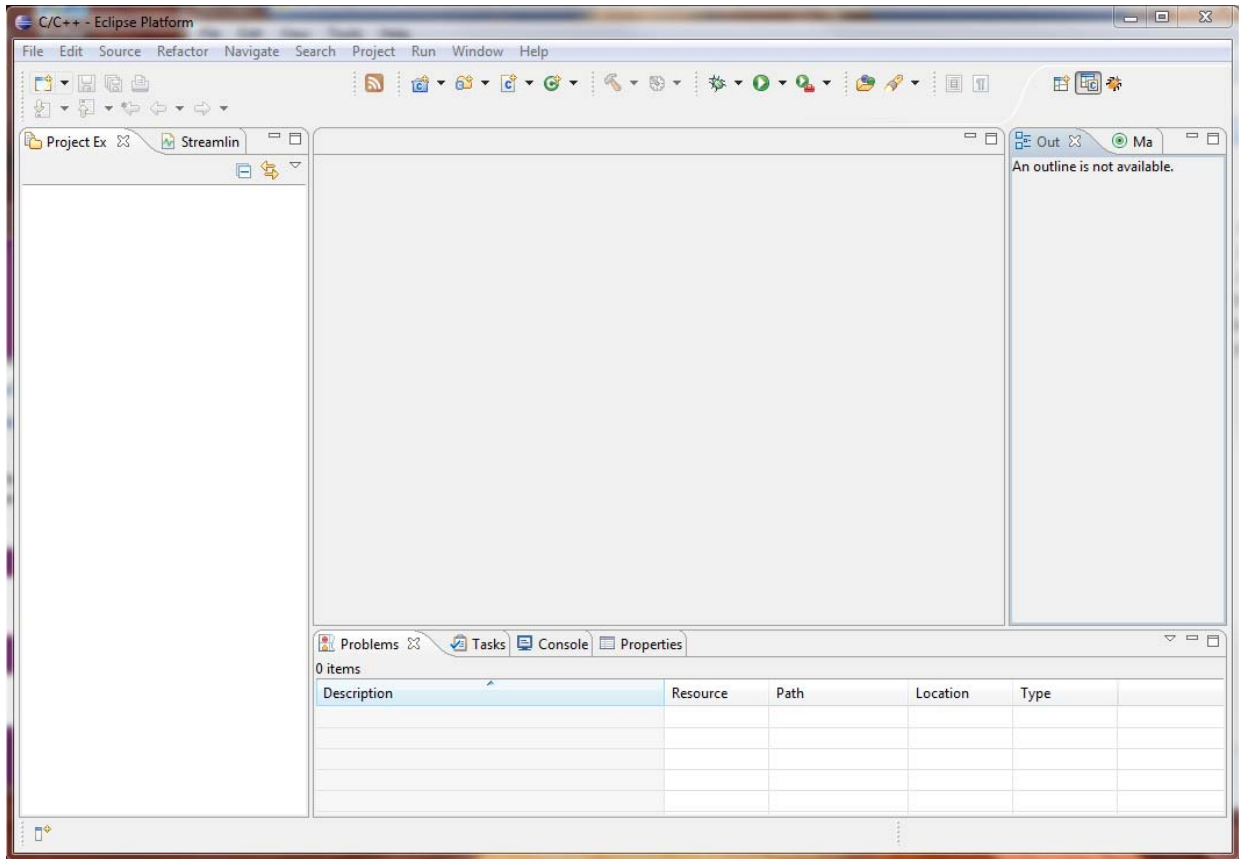
2. If the Welcome to DS-5 tab activates, you can delete it by clicking the close icon as shown below:



Tab Close Icon

You can activate the Welcome screen at any time from **Help | Welcome** in the main menu.

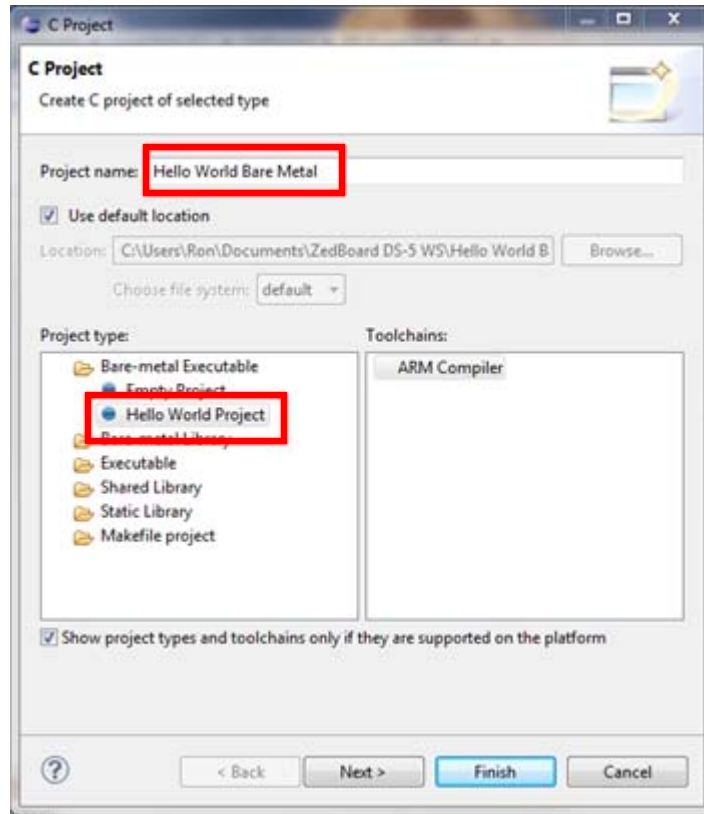
3. If you have started with an empty workspace, DS-5 will appear with a blank slate, as shown below. In a new workspace, by default the Eclipse viewer begins in the C++ perspective. If you are using a previous workspace, it will open wherever you left off last time.



Empty DS-5 C/C++ Perspective

If you are new to Eclipse, a perspective describes a particular set of panels that contain information relevant to the current activity. The C++ perspective is used for creating software projects, while the Debug perspective is used for actively debugging software on a target or software model.

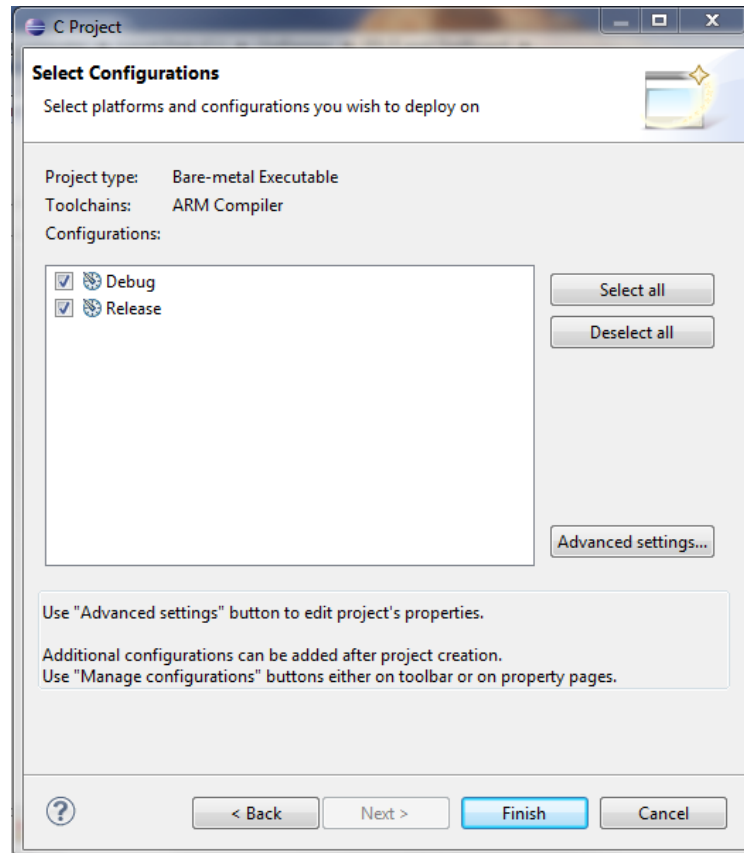
4. We will begin by creating a new Hello World software project in the C++ perspective. In the main menu, select **File | New | C Project**.



New C Project Window

Name the project **Hello World Bare Metal** to indicate that we are not relying on any underlying operating system, and select the **Hello World Project** under Bare-metal Executable in the Project Type panel. By default, the ARM C Compiler will be used to generate the object files. Click the **Next** button.

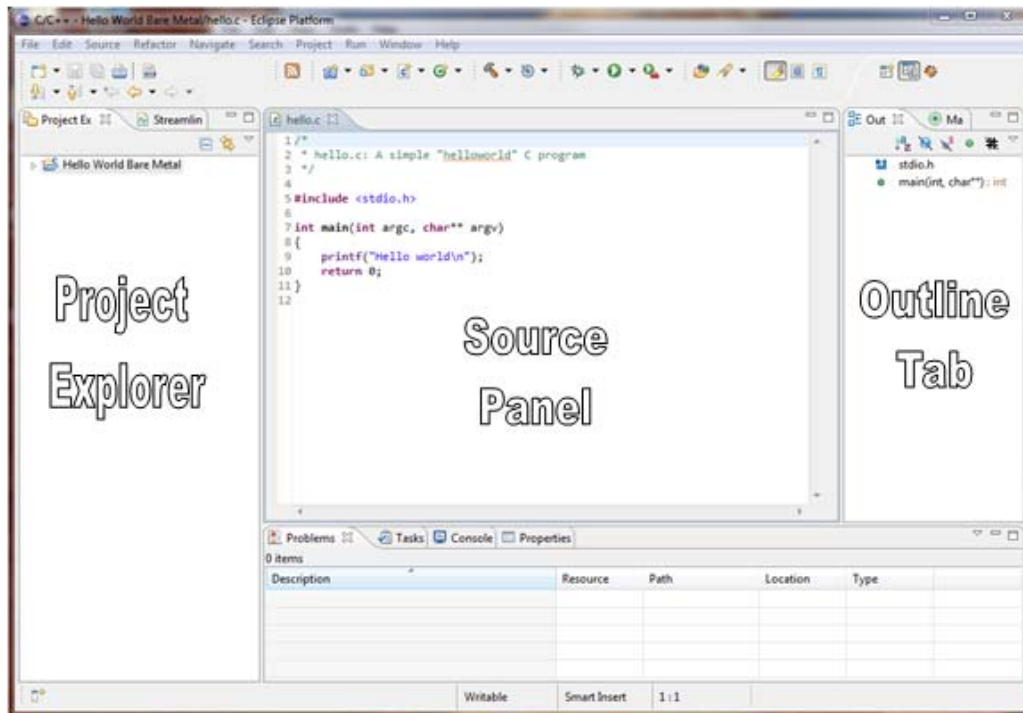
5. Within Eclipse you can set up multiple configurations for building each software project. By default, the Debug and Release configurations will be created. The Debug configuration will be compiled with no optimization and with all symbols included, so the debugger can use them to link to source level information presented in the Debug Perspective. The Release configuration uses the maximum optimization level and creates the smallest executable without sacrificing performance. As only one configuration can be active at any time, the Debug configuration is activated first.



C/C++ Build Configuration Window

If you want to change or explore the default attributes for any configuration, click on the **Advanced settings** button. For our purposes, we will accept the default attributes so click **Finish** to generate the new project.

6. You will now see a new project shown in the Project Explorer tab, the generated source code in the **hello.c** tab of the Source display panel, and an outline view in the Outline tab on the right. The tabs across the bottom will show the results of our project builds.



DS-5 C/C++ Perspective

7. In the Project Explorer tab, click on the **Hello World Bare Metal** entry to ensure it is selected. Then from the main menu, select **Project | Build Project** to compile the source code. If there were any warnings or errors, they would appear in the **Problems** tab, where you may double-click on the entry to take you directly to the line in your source. This is useful for large programs with many source modules. You can also click on the **Console** tab to see the step by step execution results of the compilation process. Note that the executable file created has an axf extension, which stands for ARM eXecutable Format. It is identical to the standard ELF format.

You can resize any of the panels in an Eclipse view by moving the cursor over the double line boundary, until it changes to a double-headed arrow.



Left mouse-click and drag the border to resize the panel as you desire.

Debug a C Program in DS-5

For ZedBoard:

To begin the procedure, set the ZedBoard Boot Mode to **JTAG only** using jumpers JP11 to JP7 set to the following:

	JP11	JP10	JP9	JP8	JP7
Position	SIG-GND	SIG-GND	SIG-GND	SIG-GND	SIG-GND

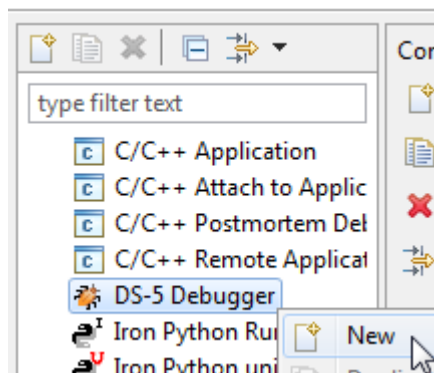
For MicroZed:

To begin the procedure, set the MicroZed Boot Mode to **JTAG only** using jumpers JP3 to JP1 set to the following:

	JP3	JP2	JP1
Position	1-2	1-2	1-2

You should have your DSTREAM, ZED target, ZedBoard Adapter and host PC connected together and powered as described in the Tutorial #1 of this Series.

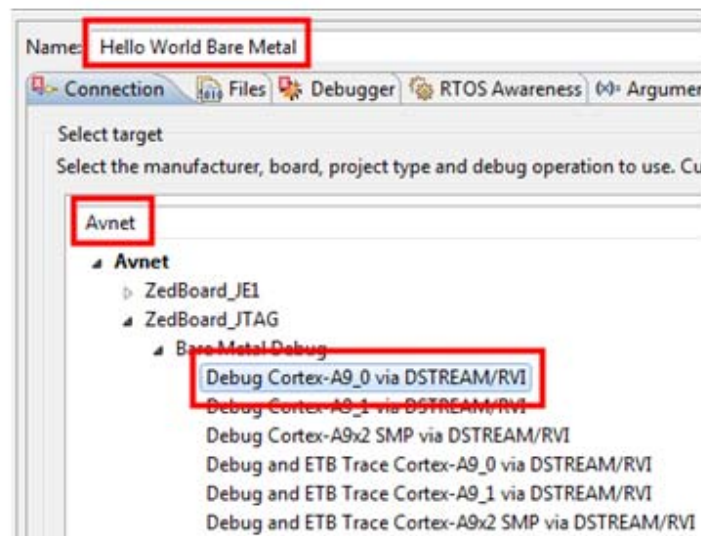
1. In the Project Explorer tab, right-click on your **Hello World Bare Metal** project and select **Debug As | Debug Configurations** from the drop-down menu.
2. Right-click on the **DS-5 Debugger** entry and select **New** from the drop-down menu.



New DS-5 Debugger Configuration

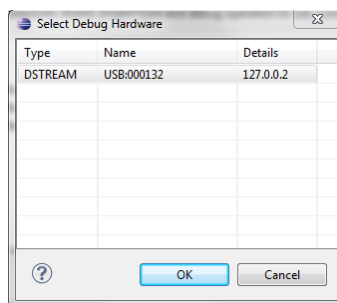
3. In the **Connection** tab:

- a. enter **Hello World Bare Metal** for the configuration Name
- b. Type **Avnet** into the filter box in the **Select target** panel
- c. Expand the **Avnet**, **ZedBoard_JTAG** and **Bare Metal Debug** entries as shown by clicking on the triangle to the right of the entries
- d. Select **Debug Cortex-A9_0 via DStream/RVI**. This selects processor 0 to run our application, and indicates we will be using the DStream unit for debugging.



Board Selection

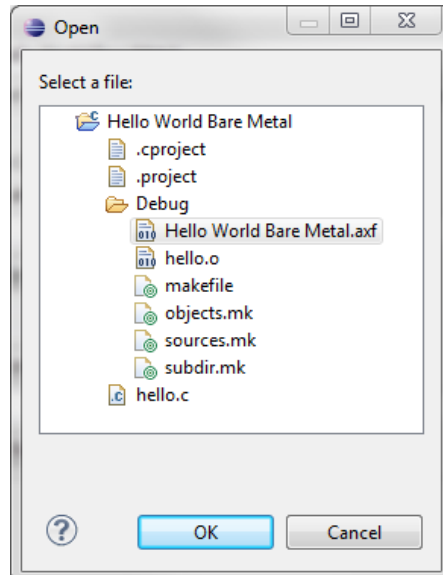
4. Still in the **Connection** tab, click on the **Browse** button in the **Connections** panel. In the pop-up window that appears, after a few moments you should see a USB connection to the DSTREAM unit. Select the DSTREAM entry and click **OK** to populate the Connection box.



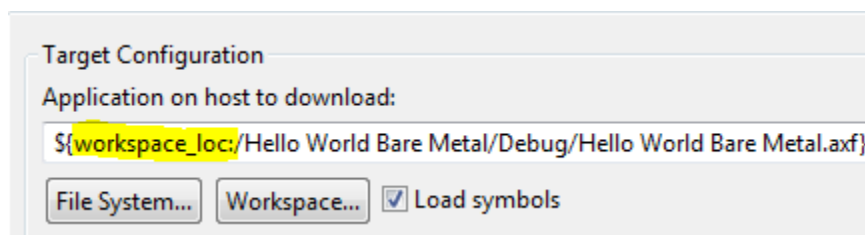
DStream Connection

If you do not see an entry in the Select Debug Hardware window, check that your DSTREAM unit is powered on and is plugged into an active USB port on your host computer. Make sure all connections are made as specified in Tutorial #1 of this Series.

5. Click on the **Files** tab. Under **Target Configuration**, click the **Workspace** button. Expand the **Hello World Bare Metal** and **Debug** entries by clicking on the triangle to the right of each entry. Select **Hello World Bare Metal.axf** and click the **OK** button to populate the **Application on host to download** text box.

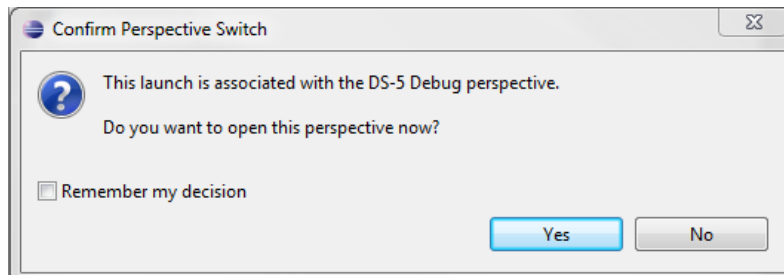


Note that DS-5 uses a variable to indicate the local workspace. You can use these variables to refer to specific folders wherever you need to type in a path.



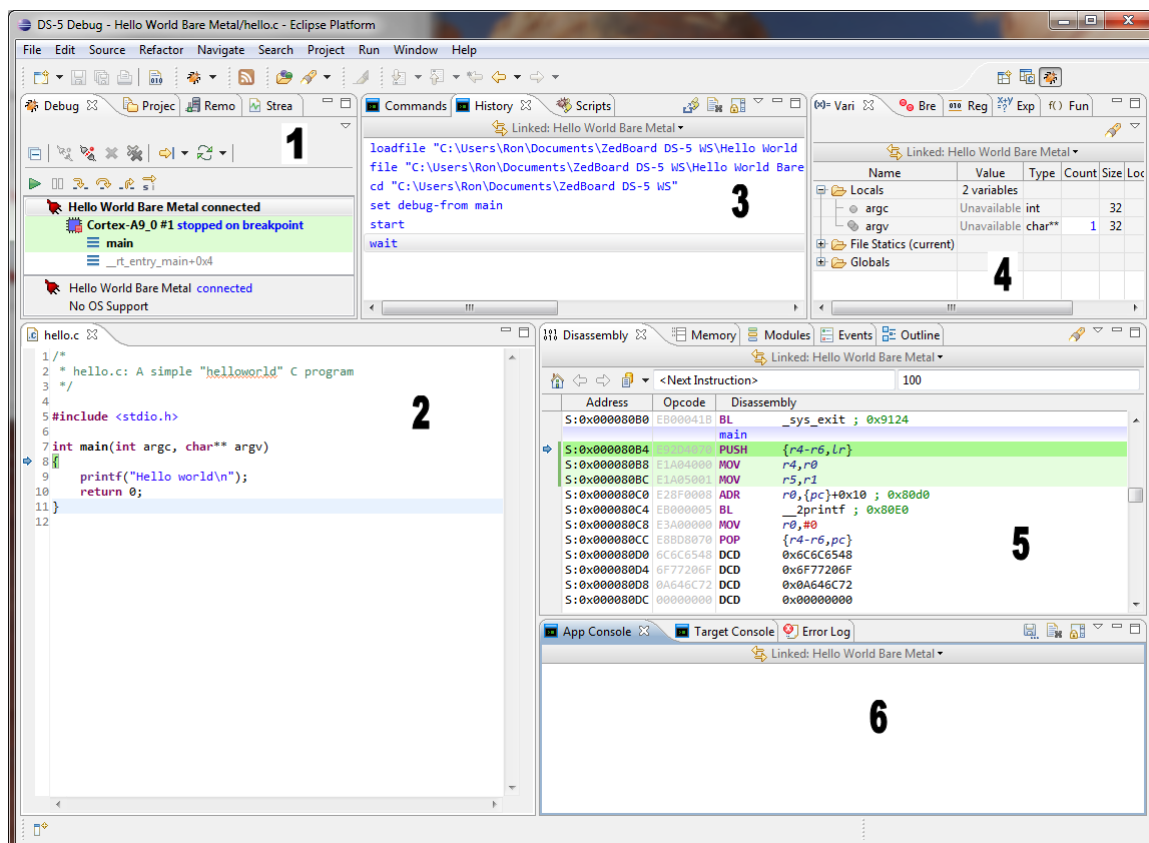
6. Still in the Files tab, in the **Files** panel click the **Workspace** button. Use the same procedure as in the previous step to select Hello World Bare Metal.axf as the file containing the symbols for debugging.
7. Select the Debugger tab and click the **Apply** button. We want our application to stop as soon as the main entry point is reached, so we leave the **Debug from symbol main** selected.

- Click the **Debug** button. We are still in the C/C++ perspective, so Eclipse asks if you want to switch to the Debug perspective. This will present a new set of windows specific to the debugging procedures, and since this is what we want click the **Yes** button.



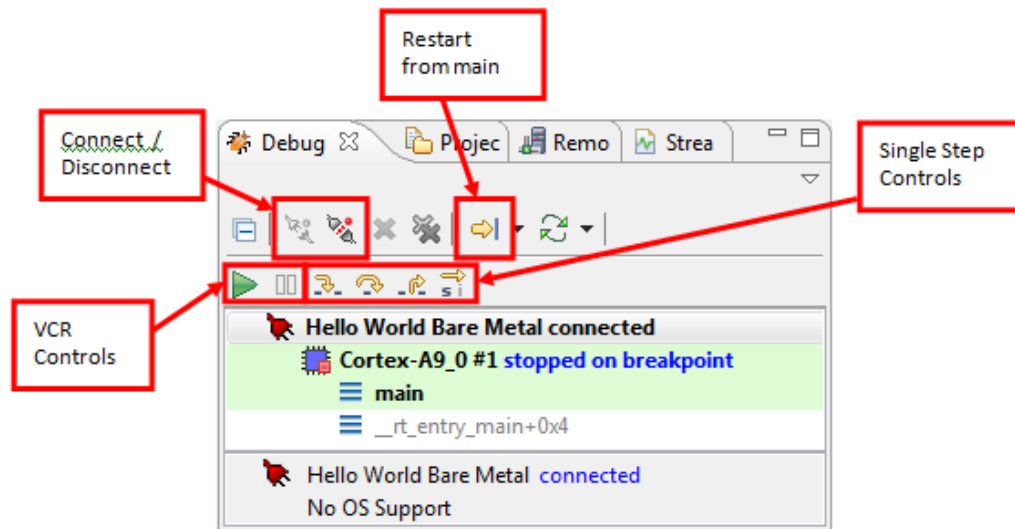
Confirm Perspective Window

- After a few seconds to connect and download the application, the Debug Perspective will populate as shown below. There are a number of panels which you should take note of.



DS-5 Debug Perspective

Panel 1: This is where the primary debug session controls are placed. Standard symbolic debugger single step operations for step into, step over and step out, as well as a toggle switch to step by source line or by machine instruction are found here. The most commonly used controls are shown below.



Debug Control Panel

The current state of the processor(s) is shown here as well. In this case the Cortex-A9 processor 0 is stopped at the initial breakpoint in main, and the call stack is appears directly below.

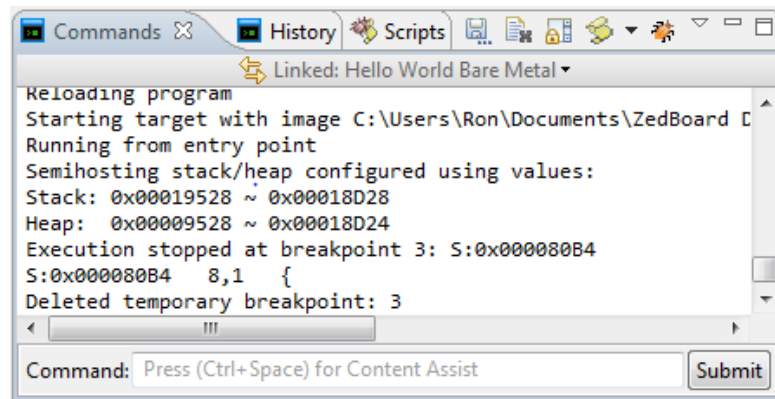
Panel 2: This is the source debug panel, where the current point of execution is shown. As you manipulate the debug session with the controls above, the source line indicator will reflect the changes.

```
hello.c
1 /*
2  * hello.c: A simple "helloworld" C program
3  */
4
5 #include <stdio.h>
6
7 int main(int argc, char** argv)
8 {
9     printf("Hello world\n");
10    return 0;
11 }
12
```

Source Panel

You can set a breakpoint here by double-clicking in the margin to the left of the source line.

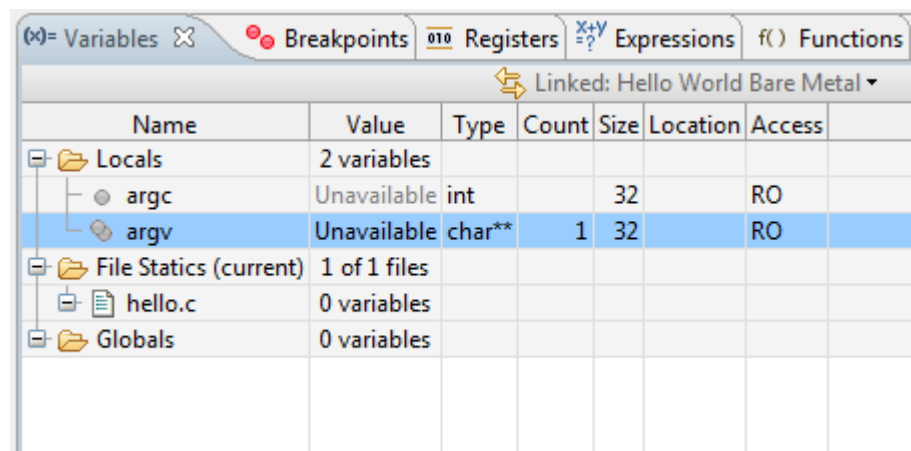
Panel 3: The Status panel maintains a list of all the commands and responses, in command line format, as executed by the IDE. Debug commands can be entered manually in the box provided on the **Commands** tab. The History tab shows a list of only the commands, without any of the responses. This is very useful for creating a repeatable debugging session, as you can copy the history entries and paste them into the Scripts tab to capture all or any portion of a debugging session.



Status Panel

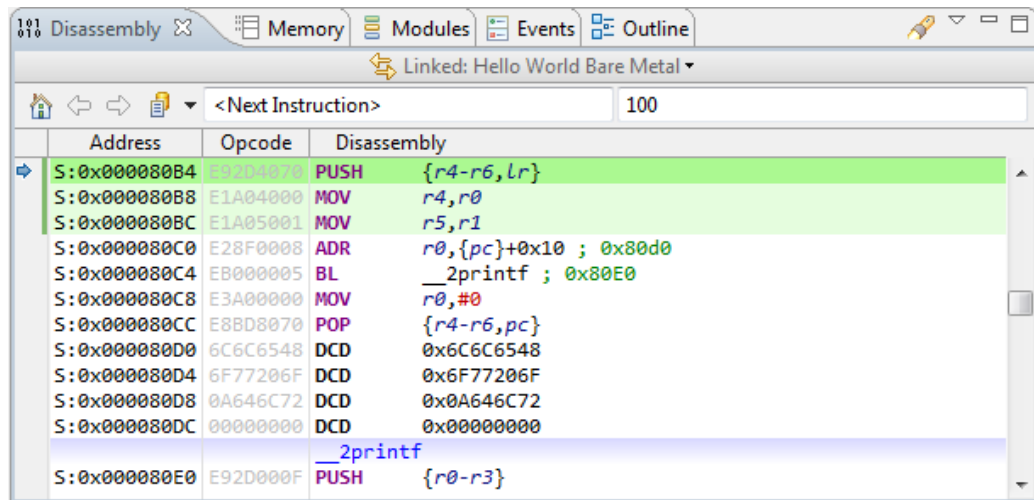
Panel 4: The State panel reports the current state of the symbolic debugger, including:

- Which variables are in scope and their current values
- Breakpoints set through the program
- Register values
- Expression values (you can enter expressions for evaluation here)
- Function list



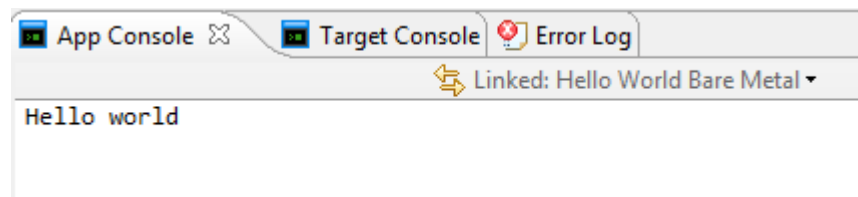
State Panel

Panel 5: In this panel low-level information is presented. The Disassembly tab tracks the source panel as execution through the debugger proceeds. In the Memory tab, you can display the contents of any memory address.



Low-level Panel

Panel 6: In the DS-5 console, you can watch for output from the executing application, as well as any Errors that may occur. If you single step through the printf statement in the source window, you will see the hello world text display in the App Console tab.



Console Panel

- This has been a brief introduction to the DS-5 Debug perspective and how you can use it in conjunction with the DSTREAM hardware to download a program to real hardware for a software debugging session. Hello world provides a simple mechanism to do this, but the program is not complex enough to require any advanced features of the debug system. More advanced topics will be covered in subsequent tutorials in this series.

You can disconnect the current debug session at this point, close DS-5 and power down the hardware.

The next time you wish to connect with this debug configuration, simply open the **Debug Control** tab, select **Hello World Bare Metal** and click the **Connect to Target** button (See Step 9, Panel 1).

Revision History

Date	Version	Revision
14 May 13	00	Initial Draft
20 Sept 13	01	Release

Resources

<http://www.zedboard.org>

<http://www.xilinx.com/zyng>

<http://www.arm.com/products/tools/software-tools/ds-5/index.php>