AN 022: Using the Ruby SoC with Frame Buffer Example Design

AN022-v1.0
August 2020
www.efinixinc.com

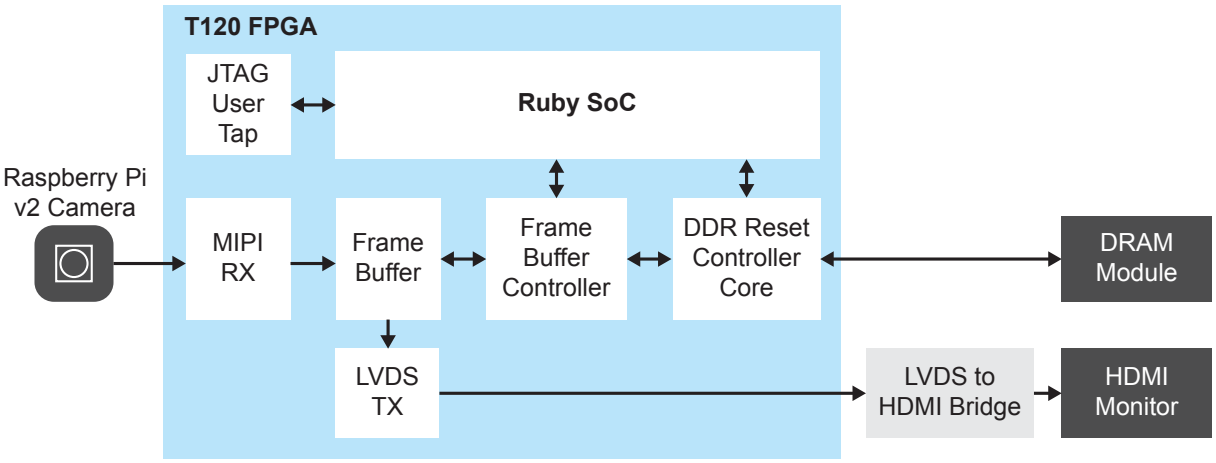
Contents

Introduction

The Ruby SoC with Frame Buffer example design illustrates how to use a RISC-V processor to control video, generated by a Raspberry Pi camera, displayed on an HDMI monitor. The RTL design targets the Trion® T120 BGA324 Development Board.

The SoC with Frame Buffer example design consists of 2 parts: an RTL design that implements the hardware and targets the Trion® T120 BGA324 Development Board and a software application that lets you interactively control the video displayed on the HDMI monitor.

Figure 1: Ruby SoC with Frame Buffer RTL Design Block Diagram



FPGA Support

The Ruby RISC-V SoC and frame buffer example design supports Trion® T120 FPGAs only.

Resource Utilization and Performance

FPGA	Logic Utilization (LUTs)	Memory Blocks	f_{MAX} (MHz)	Language	Efinity Version
T120 F324 I4	20,189	225	53.8	Verilog HDL	2020.1

Required Hardware

The example design uses the following hardware from the Trion® T120 BGA324 Development Kit:

- Trion® T120 BGA324 Development Board
- Raspberry Pi Camera Connector Daughter Card
- MIPI and LVDS Expansion Daughter Card
- Raspberry Pi v2 camera module
- 15-pin flat cable
- Micro-USB cable
- 12 V power adapter

You also need the following hardware, which you provide yourself:

- USB-to-UART module
- 3 jumper wires
- 1080p monitor with HDMI connector
- HDMI cable
- Computer with Efinity® software installed

Required Software

The example design uses the following software:

- RISC-V SDK for Windows or Ubuntu
- Efinity® software version 2020.1

Note: Refer to the Ruby RISC-V SoC Hardware and Software User Guide for instructions on installing the RISC-V SDK.

Download and Install the Example Design

The Ruby SoC with Frame Buffer example design includes a hardware example and a software example.

- *Hardware example*—The **hw** directory includes an Efinity® project and a **.hex** file for programming the board.
- *Software example*—The **sw** directory includes a makefile project and an **.elf** file for OpenOCD debugging.

1. Download the example design file, **t120f324-riscv-frame-buffer-v<version>.zip**, from the Support Center.
2. Unzip the file into your working directory.
3. The software project requires files that are included with the RISC-V SoC package. Copy the **RiscV_wPiCam** directory (which is in **RiscV_PiCam/sw**) into the **soc_Ruby/soc_Ruby_sw/software/standalone** directory.

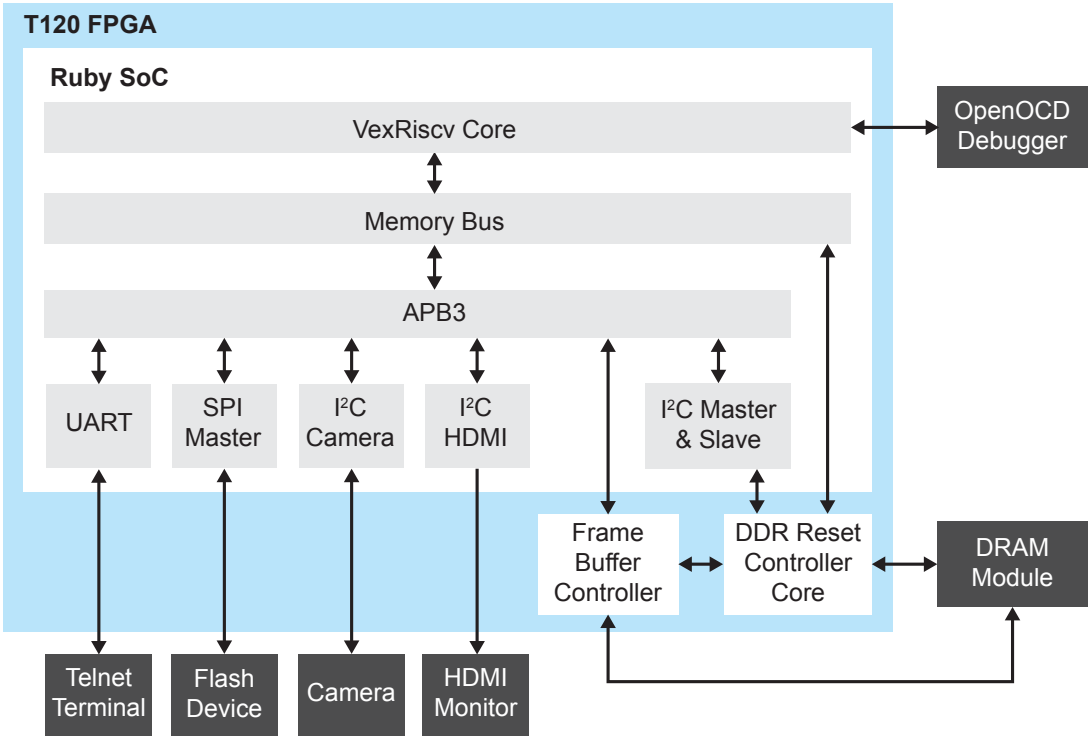
Functional Description

The Ruby SoC with Frame Buffer example design integrates the Ruby SoC with a custom APB3 peripheral (the frame buffer controller), a frame buffer, and the DDR Reset Controller core. The frame buffer controller block controls the frame buffer, which stores data and sends it to the monitor.

With the UART peripheral, you can use a terminal to send commands to the RISC-V processor. These commands control the video display on the monitor, such as cropping the height or width, setting the display location, adding a filter, etc. An I^2C peripheral sends/receives camera control commands to/from the Raspberry Pi camera.

The RISC-V processor controls the frame buffer through the APB3 bus. The frame buffer controls the LPDDR3 traffic to store the image date that streams in RGB from the camera. The frame buffer also off-loads image data to the HDMI monitor.

Figure 2: Ruby SoC Peripherals

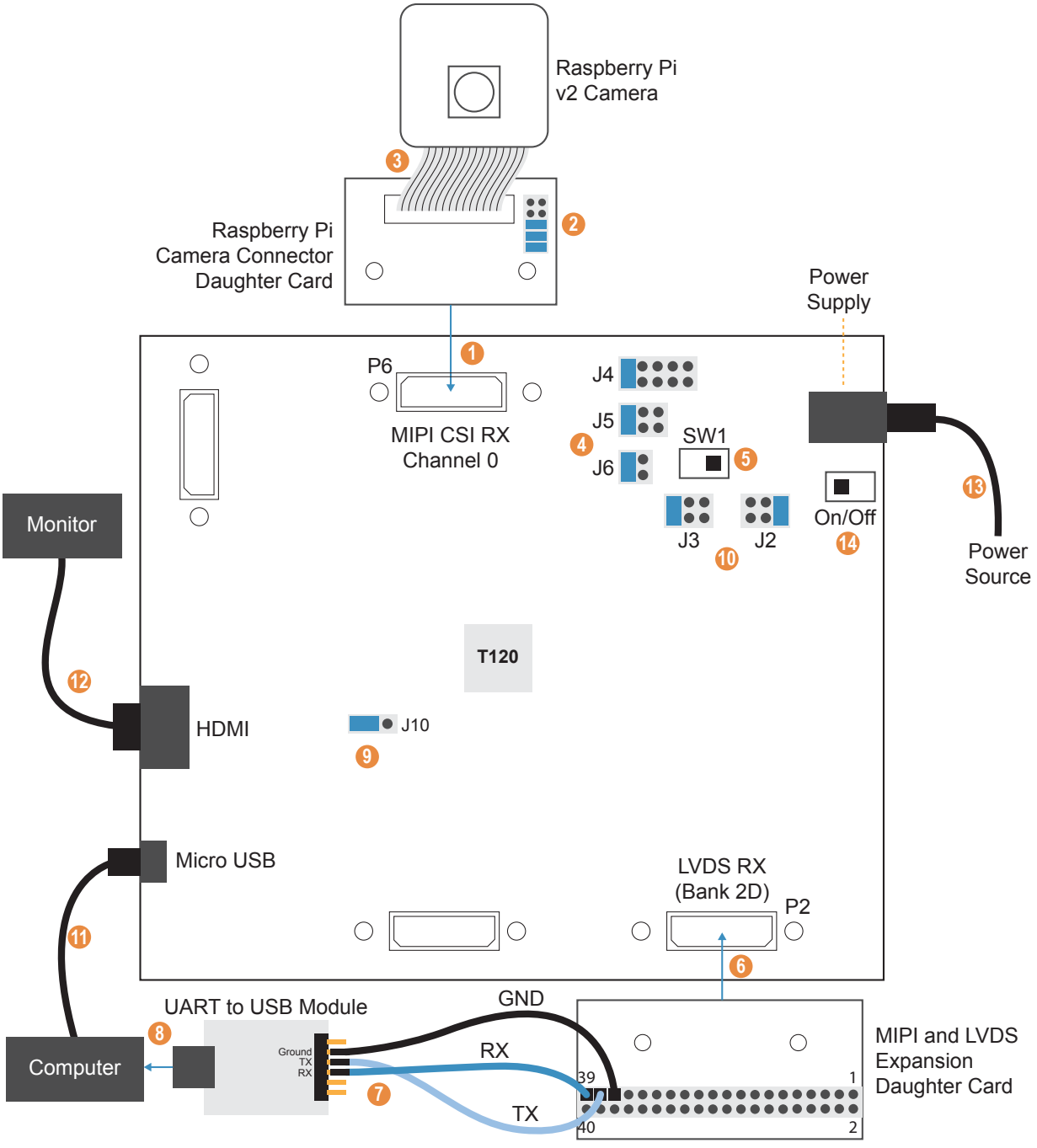


Note: The instructions in this example design assume that you are familiar with the Ruby SoC and the RISC-V SDK.

Set Up the Hardware

The following figure shows the hardware setup steps. If you have not already done so, attach standoffs to the board.

Figure 3: Hardware Setup



Important: Make sure that the Trion® T120 BGA324 Development Board is turned off before connecting any cards or cables.

Set Up the Camera

1. Connect the Raspberry Pi Camera Connector Daughter Card to the board at P6.
2. On the daughter card, connect the following pins with jumpers: 1 - 2, 3 - 4, and 5 - 6.
3. Connect the Raspberry Pi v2 camera module to the daughter card using the 15-pin flat cable.
4. Connect jumpers to set the power sequence:
   - *VSUP1 (J4)*—Connect pins 9 - 10
   - *VSUP2 (J5)*—Connect pins 5 - 6
   - *VSUP3 (J6)*—Connect pins 3 - 4
5. Slide SW1 to position 3, which enables the power up sequence circuit for the MIPI CSI-2 cameras.

Set Up the UART Module

6. Connect the MIPI and LVDS Expansion Daughter Card to the board at P2.
7. Connect jumper wires to the daughter card and UART module:
   - RX to pin 39
   - TX to pin 37
   - GND to pin 35
8. Connect the UART module to your computer.

Connect Cables and Jumpers

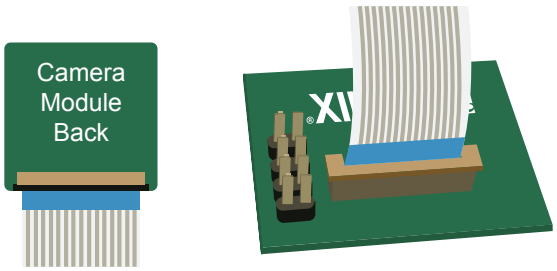
9. On header J10, connect pins 2 - 3 with a jumper (default) to use the on-board oscillator.
10. Leave the jumper on J2 at the defaults (connecting pins 1 - 2).
11. On J3, connect pins 5 - 6.
12. Connect a USB cable to the board and to your computer.
13. Connect an HDMI cable to the board and to your monitor.
14. Connect the 12 V power supply to the board connector and to a power source.
15. Turn on the board using the power switch.

Connecting the Raspberry Pi Cable

The 15-pin flat cable for the Raspberry Pi camera has a blue stripe on one side.

- When connecting to the camera, the stripe faces away from the camera.
- When connecting to the Raspberry Pi Camera Connector Daughter Card, the stripe faces away from the Efinix® logo.

Figure 4: Connecting Raspberry Pi Cable



Program the Trion® T120 BGA324 Development Board

The Trion® T120 BGA324 Development Board ships pre-loaded with an example design that sends color bars to an HDMI monitor. To use the Ruby SoC with Frame Buffer example design, you must program the design into the board.

1. Open the project (**RiscV_wFrameBuffer_top.xml**) in the Efinity® software and review it.
2. Use the Efinity® Programmer and SPI active mode to download the bitstream file to your board. The example includes a bitstream file, **RiscV_wFrameBuffer_top.hex**. You use SPI active mode because you need to reset the FPGA.
3. Press SW2 (CRESET) on the Trion® T120 BGA324 Development Board to reset the FPGA. This reset ensures that the DDR memory initialization happens before the user application runs.

Learn more: Instructions on how to use the Efinity® software is available in the Support Center.

Using the Software Example

The example comes with software in the **RiscV_wPiCam** directory. You can build the software yourself or you can use the pre-compiled **RiscV_wPiCam.elf** file.

Using this Example with the RISC-V SDK

Before working with the software included with this example design, you should already be familiar with using the Ruby SoC and RISC-V SDK. For example, you should know how to:

- Launch Eclipse using the **run_eclipse.bat** file (Windows) or **run_eclipse.sh** file (Linux) scripts.
- Set up global environment variables.
- Create and build a software project.
- Debug using the OpenOCD debugger.
- Open a UART terminal.

Learn more: Refer to the Ruby RISC-V SoC Hardware and Software User Guide for detailed instructions on how to perform these tasks.

Create a New Project

In this step you create a new project from the **RiscV_wPiCam** code example.

1. Launch Eclipse.
2. Select the Ruby workspace if it is not open by default.
3. Make sure you are in the C/C++ perspective.

Import the **RiscV_wPiCam** example:

4. Choose **File > New > Makefile Project with Existing Code**.
5. Click **Browse** next to **Existing Code Location**.
6. Browse to the **software\standalone\RiscV_wPiCam** directory and click **Select Folder**.
7. Select **<none>** in the **Toolchain for Indexer Settings** box.
8. Click **Finish**.

Import Project Settings (Optional)

Efinix provides a C/C++ project settings file that defines the include paths and symbols for the C code. Importing these settings into your project lets you explore and jump through the code easily.

Note: You are not required to import the project settings to build. These settings simply make it easier for you to write and debug code.

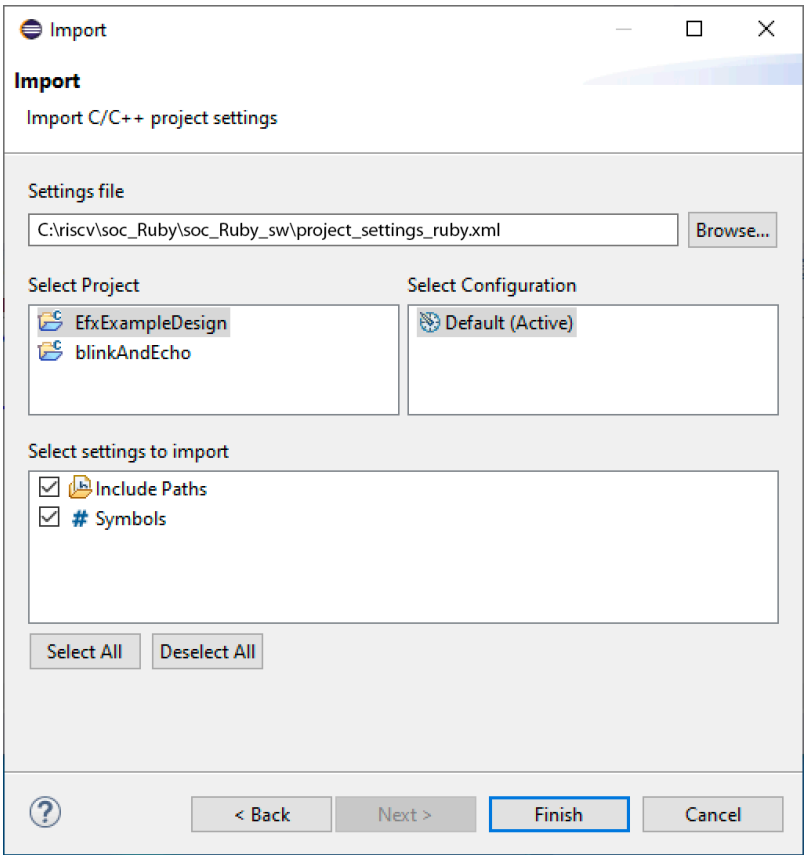
To import the settings:

1. Choose **File > Import** to open the **Import** wizard.
2. Expand **C/C++**.
3. Choose **C/C++ > C/C++ Project Settings**.
4. Click **Next**.
5. Click **Browse** next to the **Settings file** box.
6. Browse to one of the following files and click **Open**:

Option	Description
Windows	soc_Ruby\soc_Ruby_sw\config\project_settings_ruby.xml
Linux	soc_Ruby/soc_Ruby_sw/config_linux/project_settings_ruby_linux.xml

7. In the **Select Project** box, select the project name(s) for which you want to import the settings.

8. Click **Finish**.



Eclipse creates a new folder in your project named **Includes**, which contains all of the files the project uses.

After you import the settings, clean your project (**Project > Clean**) and then build (**Project > Build Project**). The build process indexes all of the files so they are linked in your project.

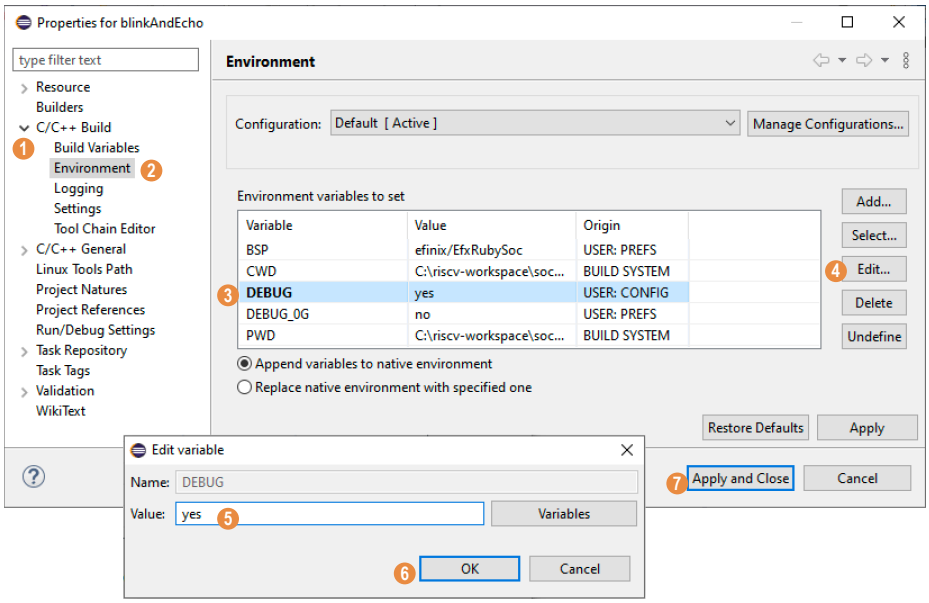
Enable Debugging

When you set up your workspace, you defined an environment variable for debugging with a default value of no.

- To run the program for normal operation, keep **DEBUG** set to `no`.
- To debug with the OpenOCD debugger, set **DEBUG** to `yes`.

In debug mode, the program suspends operation after loading so that you can set breakpoints or perform debug tasks.

To change the debug settings for your project, right-click the project name **RiscV_wPiCam** in the Project Explorer and choose **Properties** from the pop-up menu.



1. Expand **C/C++ Build**.
2. Click **C/C++ Build** > **Environment**.
3. Click the **Debug** variable.
4. Click **Edit**.
5. Change the **Value** to `yes`.
6. Click **OK**.
7. Click **Apply and Close**.

Important: When you change the debug value for a project you previously built, you must clean the project (**Project** > **Clean**) before building again. Otherwise, Eclipse gives a message in the Console that there is `Nothing to be done for 'all'`

Build

Choose **Project** > **Build Project** or click the Build Project toolbar button.

The **makefile** builds the project and generates these files in the **build** directory:

- **RiscV_wPiCam.asm**—Assembly language file.
- **RiscV_wPiCam.bin**—Download this file to the flash device on your board using OpenOCD. When you turn the board on, the SoC loads the application into the RISC-V processor and executes it.
- **RiscV_wPiCam.elf**—Use this file when debugging with the OpenOCD debugger.
- **RiscV_wPiCam.hex**—Hex file.
- **RiscV_wPiCam.map**—Contains the SoC address map.

Import the Debug Configuration

To simplify the debugging steps, the Ruby SoC v1.1 and higher includes a debug configuration that you import.

Note: If you are using v1.0, you need to set up the debug configuration manually as described in Create a Debug Configuration and Set Up Debug Environment.

1. Connect your board to your computer using a JTAG cable. If you are using an Efinix development board, you can use a USB cable instead.
2. Launch Eclipse by running the **run_eclipse.bat** file (Windows) or **run_eclipse.sh** (Linux).
3. Select a workspace (if you have not set one as a default).
4. Open the **RiscV_wPiCam** project or select it under **C/C++ Projects**.
5. Right-click the **RiscV_wPiCam** project name and choose **Import**.
6. In the Import dialog box, choose **Run/Debug > Launch Configurations**.
7. Click **Next**. The Import Launch Configurations dialog box opens.
8. Browse to the following directory and click **OK**:

Option	Description
Windows	soc_Ruby\soc_Ruby_sw\config
Linux	soc_Ruby/soc_Ruby_sw/config_linux

9. Check the box next to **config** (Windows) or **config_linux** (Linux).
10. Click **Finish**.
11. Right-click the **RiscV_wPiCam** project name and choose **Debug As > Debug Configurations**.
12. Choose **GDB OpenOCD Debugging > default**.
13. Enter `RiscV_wPiCam` in the **Project** box.
14. Enter `build\RiscV_wPiCam.elf` in the **C/C++ Application** box.
15. *Windows only:* you need to change the path to the **cpu0.yaml** file:
    a. Click the **Debugger** tab.
    b. In the **Config options** box, change `${workspace_loc}` to the full path to the **soc_Ruby_sw** directory.

    Note: For the **cpu0.yaml** path, make sure to use **** as the directory separator because the first slash escapes the second one. For example, use:
    c:\\riscv\\soc_Ruby\\soc_Ruby_sw\\cpu0.yaml

16. Click **Debug**.

Note: If Eclipse prompts you to switch to the Debug Perspective, click **Switch**.

Control the Display

You can change the display by issuing commands in a terminal.

1. Open a terminal with a speed of 115200.
2. Press the Enter or spacebar. The terminal displays a list of commands.
3. Set the window height and width to 256, and set the x and y window start position to 256. The image on the monitor shrinks and is positioned in the upper left corner.
4. Press `r` to reset the display.
5. Press `i`. The monitor displays the color bar test on the left half of the screen and video on the right half.

Revision History

Table 1: Revision History

Date	Version	Description
August 2020	1.0	Initial release.