MIV_ESS Design Guide

Introduction

The MIV_ESS is a Mi-V ecosystem IP core available for the Microchip FPGA and System-on-Chip (SoC) FPGA device families. This design guide describes four different Libero® designs highlighting various features of the MIV_ESS IP core. The following table lists the Libero designs and the corresponding features.

Table 1. MIV Design Options and Configurations

Configuration Argument	Feature	GitHub Repository
DGC1	SPI Flash Bootstrap	PolarFire® Evaluation Kit
DGC2	I²C Flash Bootstrap[1]	PolarFire® Avalanche Kit
DGC3	PolarFire® µPROM Bootstrap	PolarFire® Evaluation Kit
DGC4	MIV_RV32 basic design using MIV_ESS[2]	PolarFire® Evaluation Kit

Notes:

1. Mikroe Bus Dual EE Click board is available from https://www.mikroe.com/dual-ee-click
2. This is a basic design, which allows you to debug the software and use external peripherals such as UART, GPIO, and CoreTimers.

This guide provides quick-start steps to set up the MIV_ESS IP core in a Libero SoC SmartDesign environment, generate a bitstream, and program the FPGA data to a device. It also describes how to debug the software and use a serial terminal to capture the UART output, and how to copy a program from LSRAM to off-chip memory devices.

Note: All the designs discussed in this document are available as FlashPro Express Project job files in the "FlashPro_Express_Projects" folder, under the `BoardName_MIV_RV32_DGCX_BaseDesign.job`. BoardName is the target FPGA board name and DGCX is the configuration parameter listed in the preceding table.

Table of Contents

1. Designs

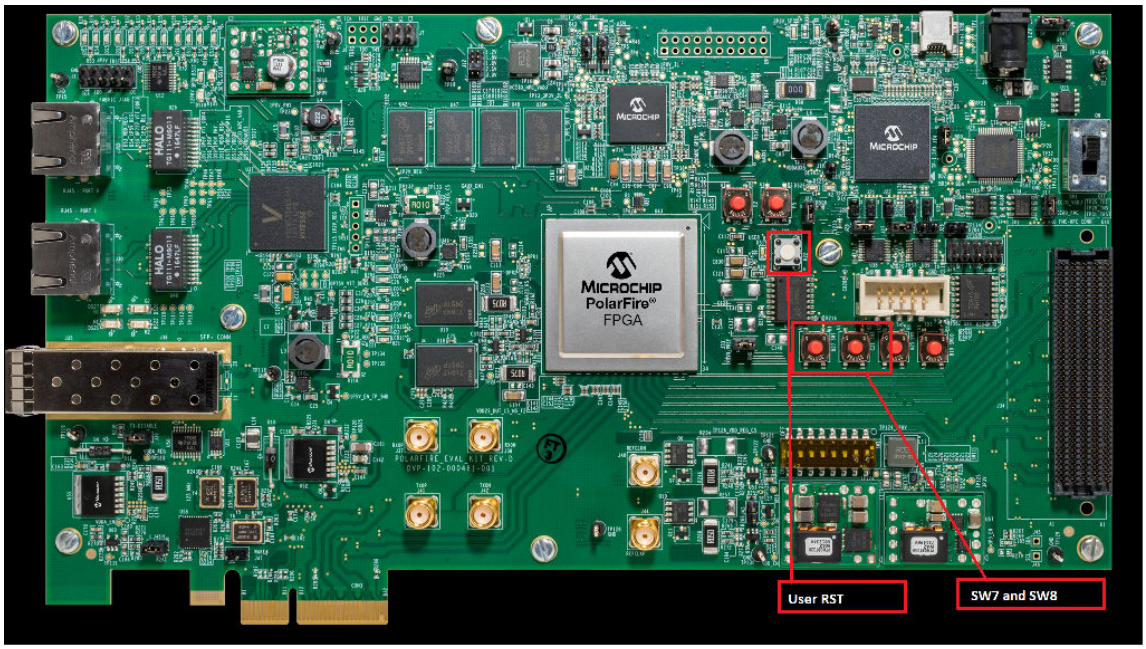
This section describes the four Libero design configurations DGC1 to DGC4 listed in Table 1. In addition, it also describes the example firmware and the software configuration changes that are required to make the firmware compatible with the design configurations.

1.1 SPI Flash Bootstrap

This design demonstrates the SPI bootstrap feature of the MIV_ESS IP core on the PolarFire Evaluation Kit board. The following features are enabled on the MIV_ESS core in this design.

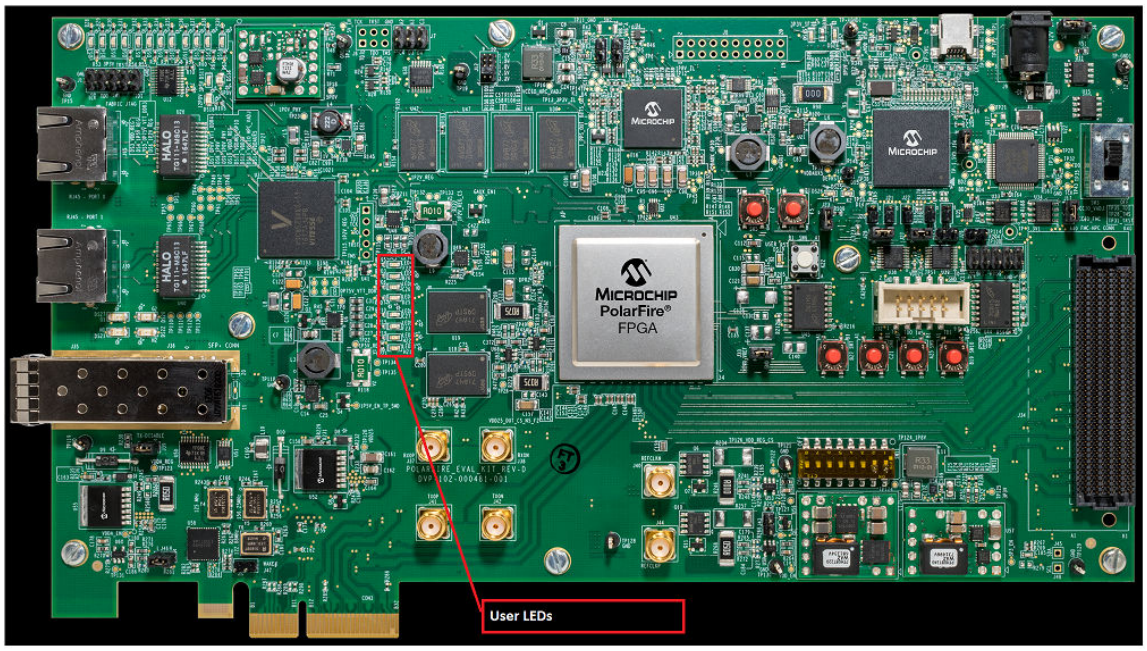
- SPI Flash
- Bootstrap
- UART
- GPIO
  - GPIO inputs connected to Push Buttons (SW7 and SW8 on the PolarFire Evaluation Kit), as shown in the following figure.

Figure 1-1. PolarFire Evaluation Kit



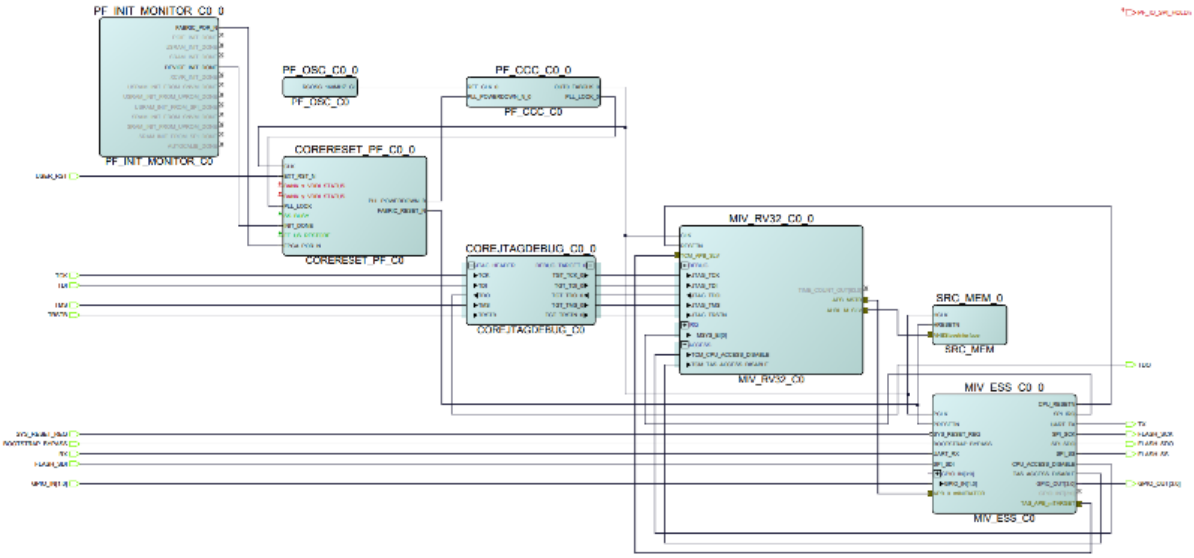
  - GPIO outputs connected to user LEDs, as shown in the following figure.

Figure 1-2. LEDs on PolarFire Evaluation Kit



The following figure illustrates the SPI Flash bootstrap design configuration.

Figure 1-3. SPI Flash Boot



The Libero SoC project is available at: github.com/Mi-V-Soft-RISC-V/PolarFire-Eval-Kit. For more information on how to run this configuration, see the `README.md` file in the GitHub project. The DGC1 design is used for this configuration.

1.1.1 Operation

On power ON, the MIV_ESS copies a program from a Flash device to the MIV_RV32 Tightly-Coupled Memory (TCM). When the MIV_ESS releases the MIV_RV32 reset, the MIV_RV32 will boot the application. The following steps describe how this is accomplished.

1. Download the Libero SoC programming files from the GitHub link provided.

   Note: This design is set up to disable the BOOTSTRAP_BYPASS, by default.

2. Program the bitstream to the development kit.
3. Open **SoftConsole**.
   a. To run the provided Elf project, see 5. Appendix: Running an Elf file.
   b. To run the provided SoftConsole project, see 3. Appendix: Running a Project from SoftConsole.
4. Open a serial terminal session, see the 3.5. Communication Through UART section.
5. Launch the debug session in SoftConsole. The program is built to copy a program from volatile to nonvolatile memory. The MIV_RV32 TCM is the target nonvolatile memory in this application design.
6. The serial terminal window is shown in the following figure. Select **1** to initiate copying the example firmware to SPI Flash from the LSRAM.

   Figure 1-4. Bootload Menu

```
================================================================================
  MIV_ESS Bootstrap support utility to load binary executable from LSRRAM to Non-Volatile memory
================================================================================


This program supports writing HEX data from Source LSRAM (@ Address 0x800000000) into a Non-Volatile memory


Choose the destination Non-Volatile memory:
 Type 0 to show this menu
 Type 1 copy .hex from LSRAM to SPI Flash
 Type 2 copy .hex from LSRAM to MikroBus EEPROM
```

7. Once the program is copied, end the debug session.
8. Push and hold SW8 and press and release SW6 or SW7.
9. Release SW8 when the LEDs on the board are actively blinking.

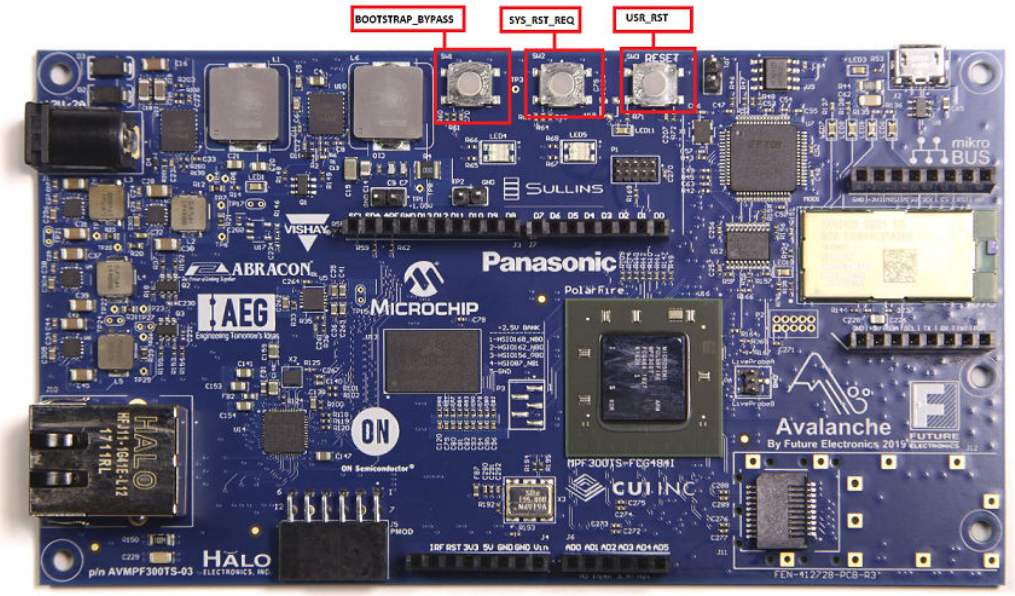
For more information on how to update a hex file in the LSRAM, see 2. Appendix: Configuring LSRAM.

1.2 I²C Bootstrap

This design demonstrates the I²C bootstrap feature of the MIV_ESS core on the Avalanche Development Kit board. This design requires a Dual EE Click board from mikroBUS inserted into the mikroBUS header on the Avalanche Development Kit board. The Dual EE Click is available at Mikroe.com. The following features are enabled on the MIV_ESS core in this design.

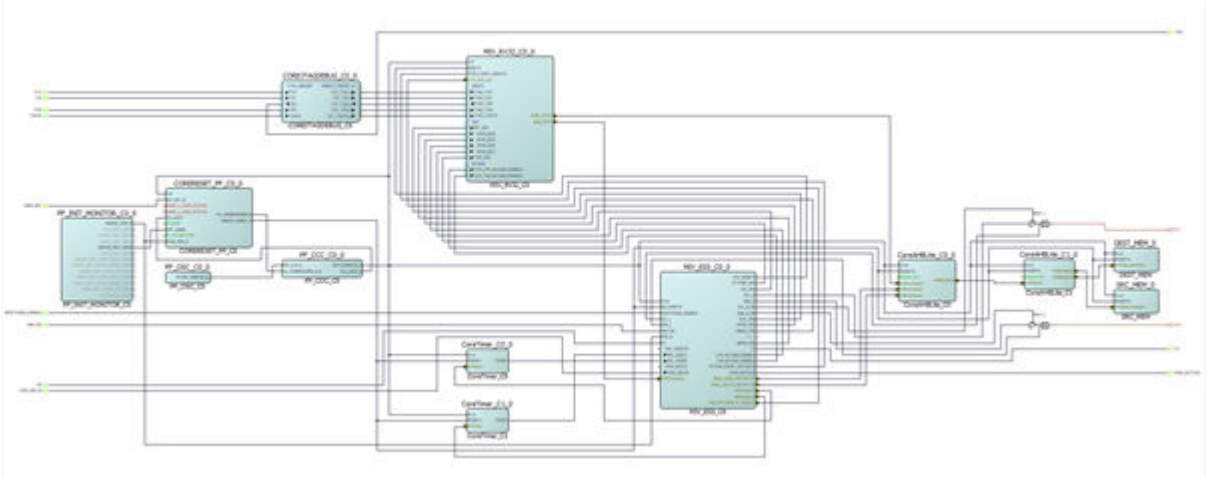
- I²C
- Bootstrap
- UART
- GPIOs
  – GPIO inputs connected to Push Buttons (SW1 and SW2 on the PolarFire Avalanche Kit), as shown in the following figure.

Figure 1-5. PolarFire Avalanche Kit



The following figure illustrates the I²C bootstrap design configuration.

Figure 1-6. I²C Bootstrap



The Libero SoC project for this configuration is available at: github.com/Mi-V-Soft-RISC-V/Future-Avalanche-Board. For more information on how to run a configuration, see the `README.md` file in the GitHub project. The DGC2 design is used for this configuration.

1.2.1 Operation

On power ON, the MIV_ESS copies a program from the I²C Flash device to the MIV_RV32 Tightly-Coupled Memory (TCM). When the MIV_ESS releases the MIV_RV32 reset, the MIV_RV32 will boot the application. The following steps describe how this is accomplished.

1. Download the Libero SoC programing files from the GitHub link provided.
2. Connect the Dual EE Click board to the Avalanche Board mikroBUS header.
3. Program the bitstream to the development kit.
   Note: This design is set up to disable the BOOTSTRAP_BYPASS, by default.
4. Open **SoftConsole**.

a. To run the provided Elf project, see 5. Appendix: Running an Elf file.
b. To run the provided SoftConsole project, see 3. Appendix: Running a Project from SoftConsole.

5. Open a serial terminal session, see 3.5. Communication Through UART section.
6. Launch the debug session. The application is built to copy a program from volatile to nonvolatile memory. The MIV_RV32 TCM is the target memory in this software project.
7. A menu is displayed in the terminal window, as shown in the following figure. Select **2** to initiate a copy operation to I^2C Flash from LSRAM.

Figure 1-7. Bootload Menu

```
================================================================================
  MIV_ESS Bootstrap support utility to load binary executable from LSRRAM to Non-Volatile memory
================================================================================


This program supports writing HEX data from Source LSRAM (@ Address 0x800000000) into a Non-Volatile memory


Choose the destination Non-Volatile memory:
 Type 0 to show this menu
 Type 1 copy .hex from LSRAM to SPI Flash
 Type 2 copy .hex from LSRAM to MikroBus EEPROM
```

8. Once the program is copied, end the debug session.
9. Push and hold SW1 to disable BOOTSTRAP_BYPASS.
10. Press and release SW2 to perform a system reset request.
11. Release SW1 when the LEDs on the board are actively blinking. The program that is stored in I^2C Flash is copied by the bootstrap into the MIV_R32 TCM, and the MIV_RV32 starts executing the example firmware.

For more information on how to update a hex file in the LSRAM, see 2. Appendix: Configuring LSRAM.

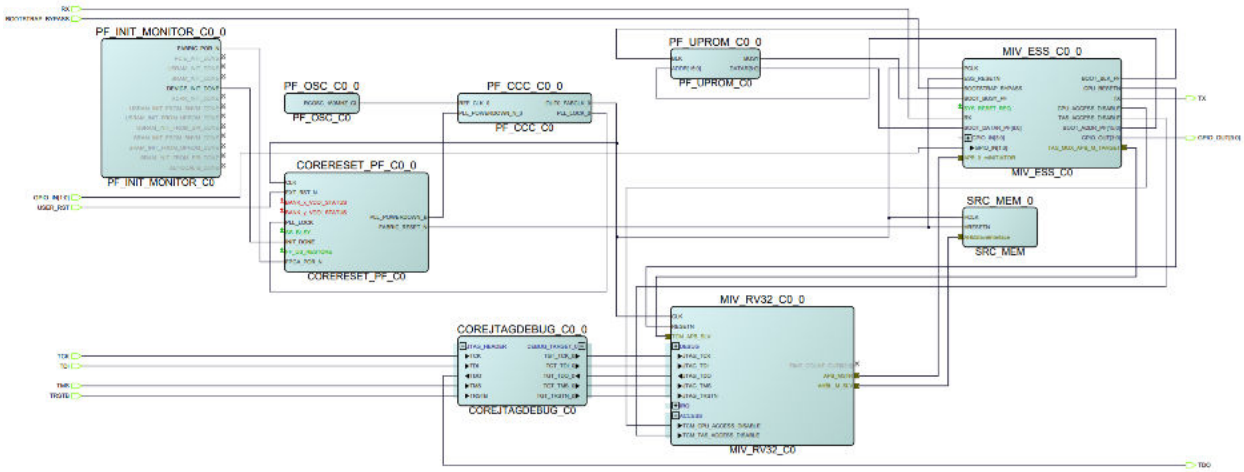
1.3 µPROM Bootstrap

This design demonstrates the µPROM bootstrap feature of the MIV_ESS core on the PolarFire Evaluation Kit board. The following features are enabled on the MIV_ESS core in this design.

- PF_uPROM
- Bootstrap
- UART
- GPIOs

The following figure illustrates the µPROM bootstrap design configuration.

Figure 1-8. µPROM Boot Load



The Libero SoC project is available at: github.com/Mi-V-Soft-RISC-V/PolarFire-Eval-Kit. The DGC3 design is used for this configuration.

1.3.1 Operation

On power ON, the MIV_ESS copies a program from the µPROM device to the MIV_RV32 Tightly-Coupled Memory (TCM). When the MIV_ESS releases the MIV_RV32 reset, the MIV_RV32 will boot the application. The following steps describe how this is accomplished.

1. Download the Libero SoC programing files from the GitHub link provided.

   Note: This design is set up to disable the BOOTSTRAP_BYPASS, by default.
2. Program the bitstream to the development kit.
3. Open a serial terminal session, see 3.5. Communication Through UART section.
4. Press the push button reset SW6 on the PolarFire Evaluation Kit.
5. The bootstrap copies a program from µPROM to MIV_RV32 TCM. When the program is copied, MIV_RV32 can execute the program in TCM.

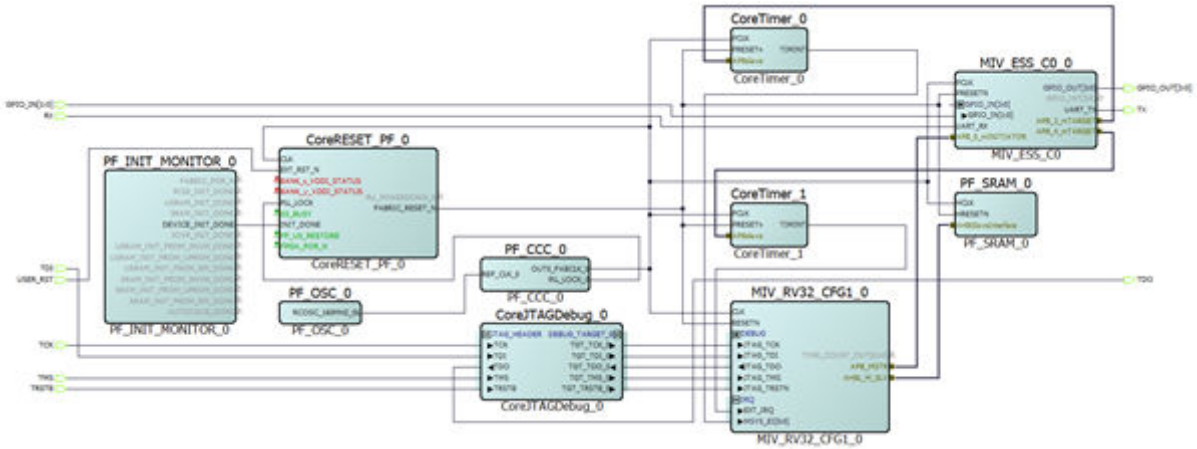
1.4 Basic Example

This section demonstrates how to set up a basic design with MIV_ESS and MIV_RV32 cores for a PolarFire Evaluation Kit. It is based on the MIV_RV32 Design Guide with the DirectCore peripherals, CoreGPIO and CoreUART, replaced with the MIV_ESS UART and GPIO. The following features are enabled on the MIV_ESS core in this design.

- UART
- GPIOs
- APB Target interfaces (used by CoreTimers)

The following figure illustrates a sample design with MIV_ESS and MIV_RV32 cores in the SmartDesign.

Figure 1-9. Basic Example



The Libero SoC project is available at: github.com/Mi-V-Soft-RISC-V/PolarFire-Eval-Kit. The DGC4 design is used for this configuration.

1.4.1 Operation

The following steps describe how to set up a design.

1. Download the Libero SoC programing files from the GitHub link provided.
2. Open a terminal session, see 3.5. Communication Through UART
3. Program the bitstream to the development kit.
4. On a reset SW6, the LEDs start blinking and a welcome message appears on the terminal.

You can run a debug example, see 3. Appendix: Running a Project from SoftConsole. For more information on how to update a hex file in the LSRAM, see 2. Appendix: Configuring LSRAM.

2. Appendix: Configuring LSRAM

To boot a program on power-on-reset, you can configure the PolarFire LSRAM using one of the following two methods.

- Configuring the LSRAM memory using the **LSRAM** configurator.
- Configuring the LSRAM memory using the **Configure Design Initialization Data and Memories** configurator.

2.1 Example Firmware

The example firmware included in the PF_SRAM in the Libero SoC design is built from the MIV_RV32 HAL example in the Firmware Catalog bundled with Libero. The `fpga_design_config.h` file from the Firmware Catalog is modified to use the MIV_ESS components and the default linker script is modified to use MIV_RV32 TCM.

The following changes are required in the `fpga_design_config.h` file.

Figure 2-1. Changes in `fpga_design_config.h`

Original **fpga_design_config.h**

```
/*******************************************************************************//**
 * Non-memory Peripheral base addresses
 * Format of define is:
 * <corename>_<instance>_BASE_ADDR
 * The <instance> field is optional if there is only one instance of the core
 * in the design
 */
#define COREUARTAPB0_BASE_ADDR          0x70001000UL
#define COREGPIO_IN_BASE_ADDR           0x70002000UL
#define CORETIMER0_BASE_ADDR            0x70003000UL
#define CORETIMER1_BASE_ADDR            0x70004000UL
#define COREGPIO_OUT_BASE_ADDR          0x70005000UL
#define FLASH_CORE_SPI_BASE             0x70006000UL
#define CORE16550_BASE_ADDR             0x70007000UL
```

Updated **fpga_design_config.h**

```
/*******************************************************************************//**
 * Non-memory Peripheral base addresses
 * Format of define is:
 * <corename>_<instance>_BASE_ADDR
 * The <instance> field is optional if there is only one instance of the core
 * in the design
 */
#define MIV_PLIC_BASE_ADDR          0x70000000UL
#define COREUARTAP0_BASE_ADDR       0x71000000UL
#define MIV_MTIMER_BASE_ADDR        0x72000000UL
#define CORETIMER0_BASE_ADDR        0x73000000UL
#define CORETIMER1_BASE_ADDR        0x74000000UL
#define COREGPIO_INOUT_BASE_ADDR    0x75000000UL
#define FLASH_CORE_SPI_BASE_ADDR    0x76000000UL
#define PF_UPROM_BASE_ADDR          0x77000000UL
#define uDMA_BASE_ADDR              0x78000000UL
#define MIV_WDT_BASE_ADDR           0x79000000UL
#define MIV_I2C_BASE_ADDR           0x7A000000UL
```

The following changes are required to the linker script (`miv-rv32-ram-imc.ld`). The RAM ORIGIN needs to be updated from 0x80000000 to 0x40000000, and the RAM_START_ADDRESS needs to be updated from 0x80000000 to 0x40000000, as shown in the following figure.

Figure 2-2. Changes to Linker Script

```
MEMORY
{
    ram (rwx) : ORIGIN = 0x80000000, LENGTH = 16k
}

RAM_START_ADDRESS    = 0x80000000;       /* Must be the same value MEMORY region ram ORIGIN above. */
RAM_SIZE             = 16k;              /* Must be the same value MEMORY region ram LENGTH above. */
STACK_SIZE           = 1k;               /* needs to be calculated for your application */
HEAP_SIZE            = 0k;               /* needs to be calculated for your application */
```

2.2 Configuring LSRAM Using the LSRAM Configurator

The following steps describe how to configure LSRAM using the LSRAM Configurator.

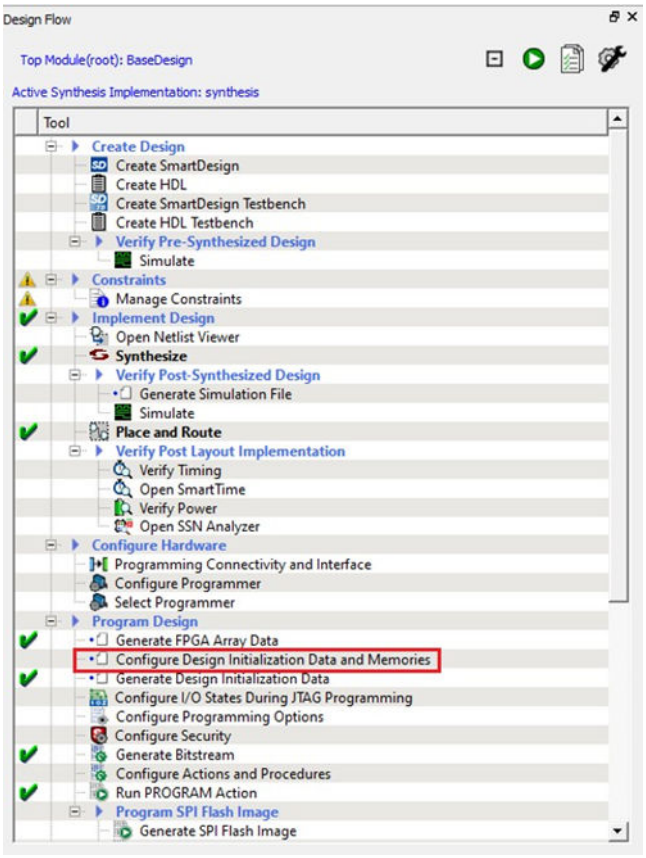
1. Right click on the design hierarchy and select **Configure**.
2. Click on the **Memory Initialization Settings** tab.
3. Select the **Initialize RAM at Power up** check box.
4. Select the hex file created by SoftConsole while building a project, see Creating a Deployable Hex File.

Note: The maximum size of the hex file is 32 KB.

2.3 Configuring LSRAM Using the Configure Design Initialization Data and Memories Configurator

If you have run the bitstream generation and want to add or change the client in an LSRAM, you can configure it using the **Configure Design Initialization Data and Memories** configurator as shown in the following figure.

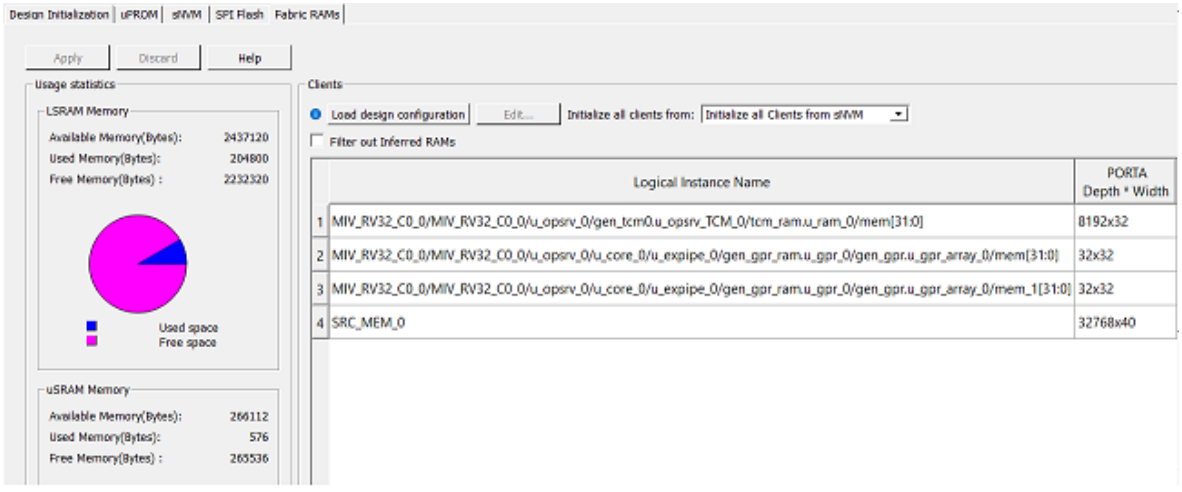
Figure 2-3. Design Flow: Configure Design Initialization Data and Memories



The following steps describe how to configure LSRAM using the Configure Design Initialization Data and Memories Configurator.

1. Ensure to run the Generate FPGA Array Data step of the design flow.
2. Run the design flow to complete the “Generate FPGA Array Data” step in the design flow.
3. Click on the **Fabric RAMs** tab.
4. Double click on the RAM to be initialized to open its configurator.

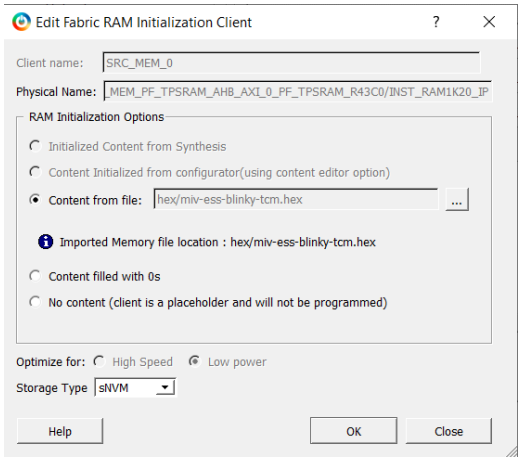
Figure 2-4. RAM Initialization GUI



5. Under **Ram Initialization Options**, select **Content from file** option and select the hex file that SoftConsole generates while building a project. See 3.4. Creating a Deployable Hex File

6. Choose the appropriate storage location, as shown in the following figure. For example, **sNVM**.

Figure 2-5. RAM Initialization Client



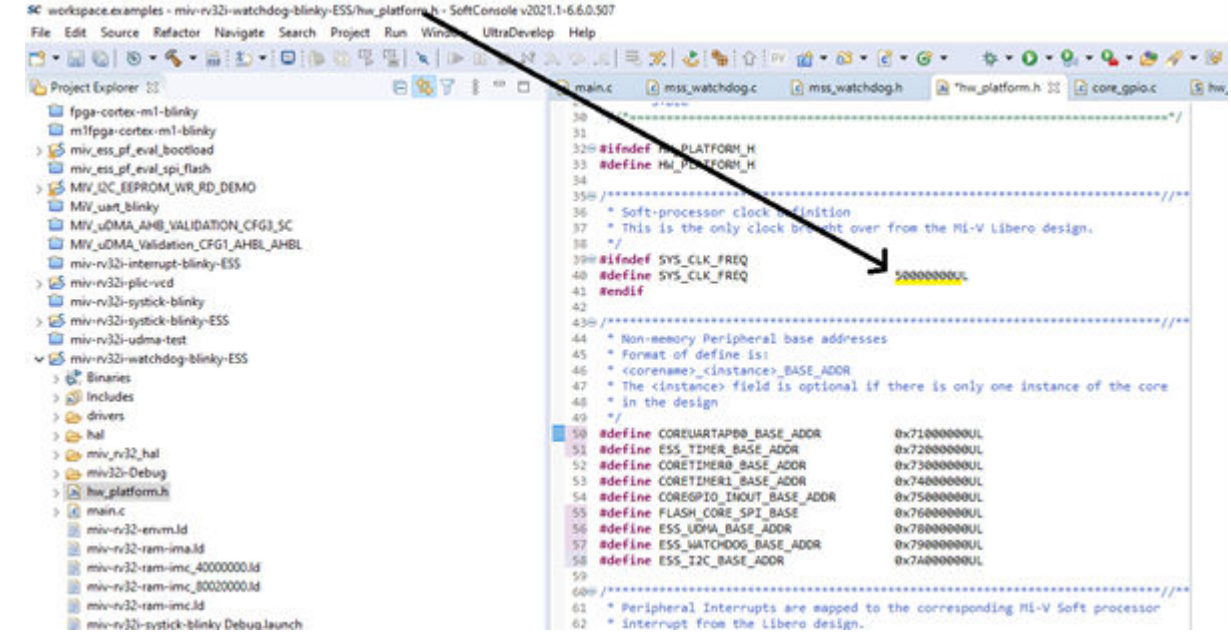
7. Click **OK**.
8. Click **Apply** in the **Fabric RAMs** tab.

3. Appendix: Running a Project from SoftConsole

This chapter describes how to build and run a project in SoftConsole for the MIV_RV32 core. Open SoftConsole and an example project.

The system clock frequency and the peripheral base addresses of the MIV_ESS components are set in the `fpga_design_config.h` file, as shown in the following figure.

Figure 3-1. Update to `fpga_design_config.h` System Clock and Peripheral Addresses

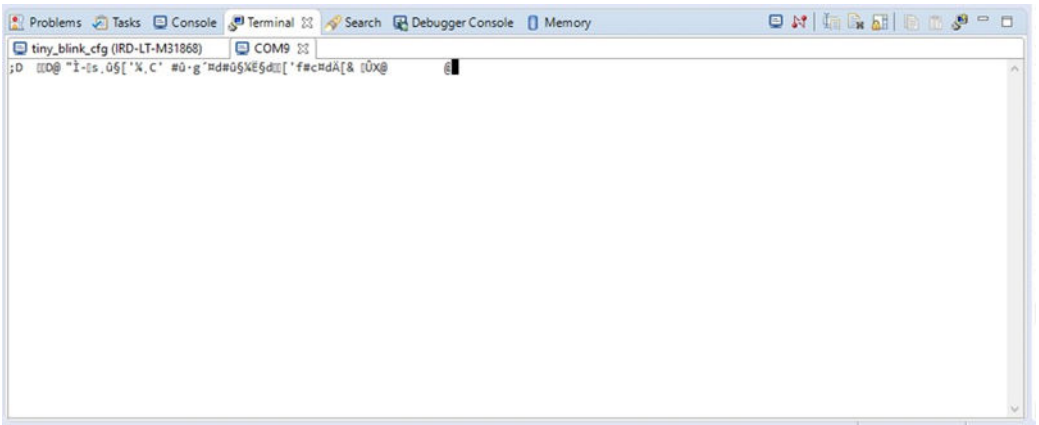


3.1 Setting the System Clock Frequency

If UART is used in the design, the system clock frequency must be updated in the `fpga_design_config.h` file by changing the **#define SYS_CLK_FREQ** to the clock frequency. An example of unexpected behavior is shown the following figure.

Note: The system clock frequency value must be in hertz.

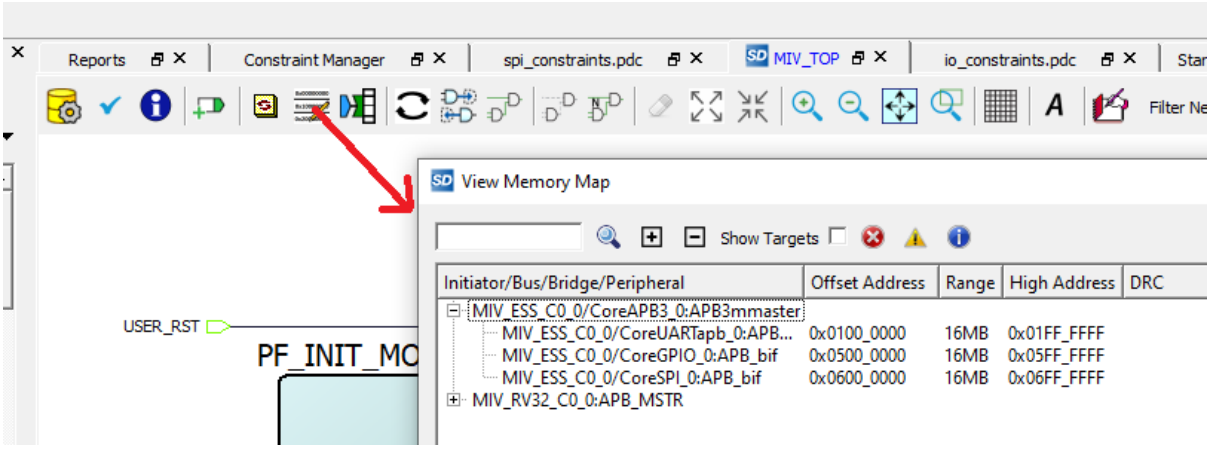
Figure 3-2. Unexpected Behavior on the Terminal



3.2 Setting the Peripheral Base Addresses

The `fpga_design_config.h` file is used to set the base address for peripherals. You can find the base address of a peripheral in the Libero generated project memory map view, as shown in the following figure.

Figure 3-3. Generated Memory Map in Libero SoC



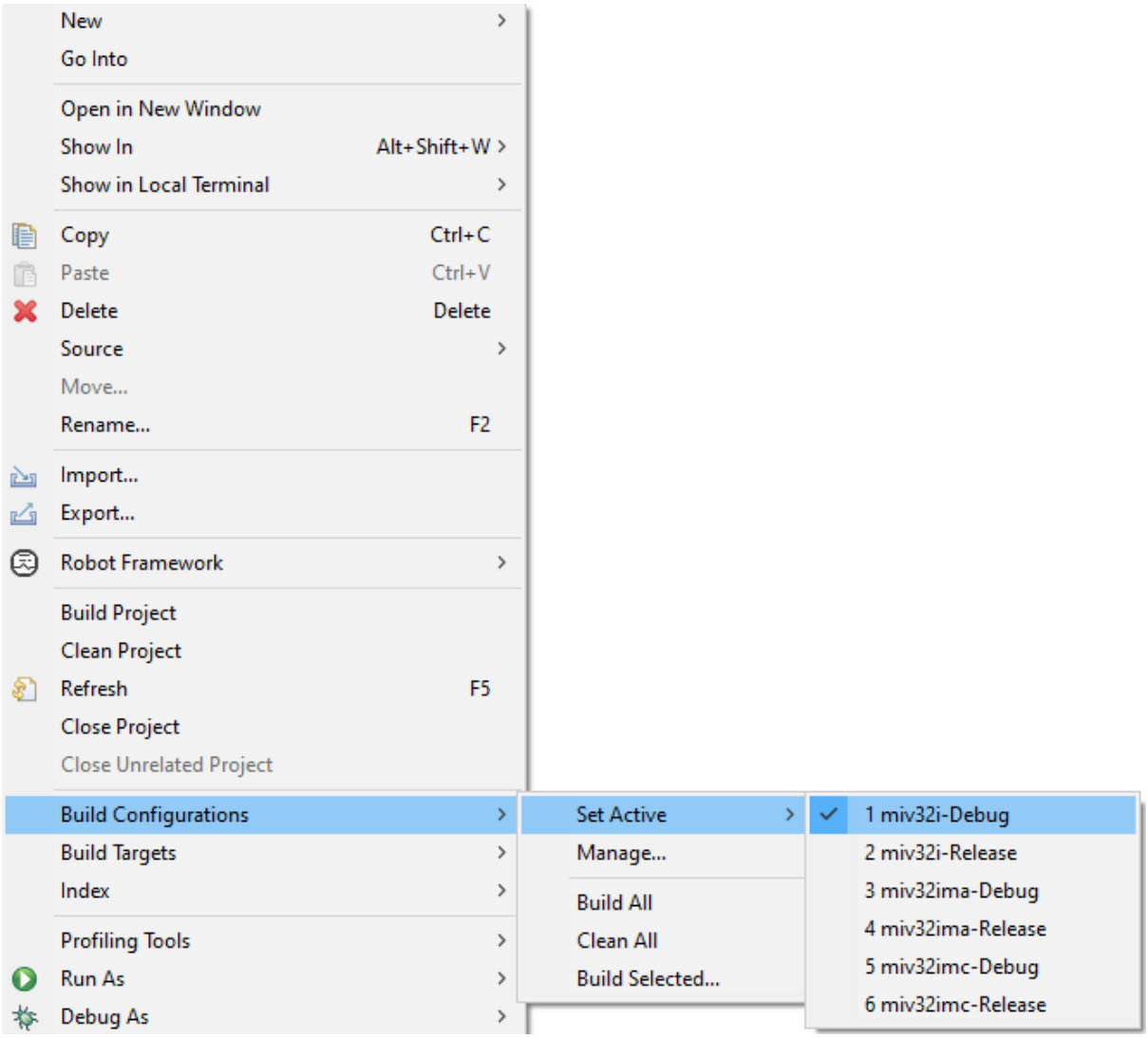
The peripheral module address [27:0] in the `fpga_design_config.h` file must match the address in Libero for the peripheral to function correctly. This upper nibble [31:28] is determined by the APB Initiator connected to the MIV_ESS core. In MIV_RV32, the default APB Initiator address is 0x7000_0000, which corresponds to the address shown in the following figure.

Figure 3-4. MIV_RV32 Peripheral Port Base Addresses

3.3 Building a SoftConsole Project

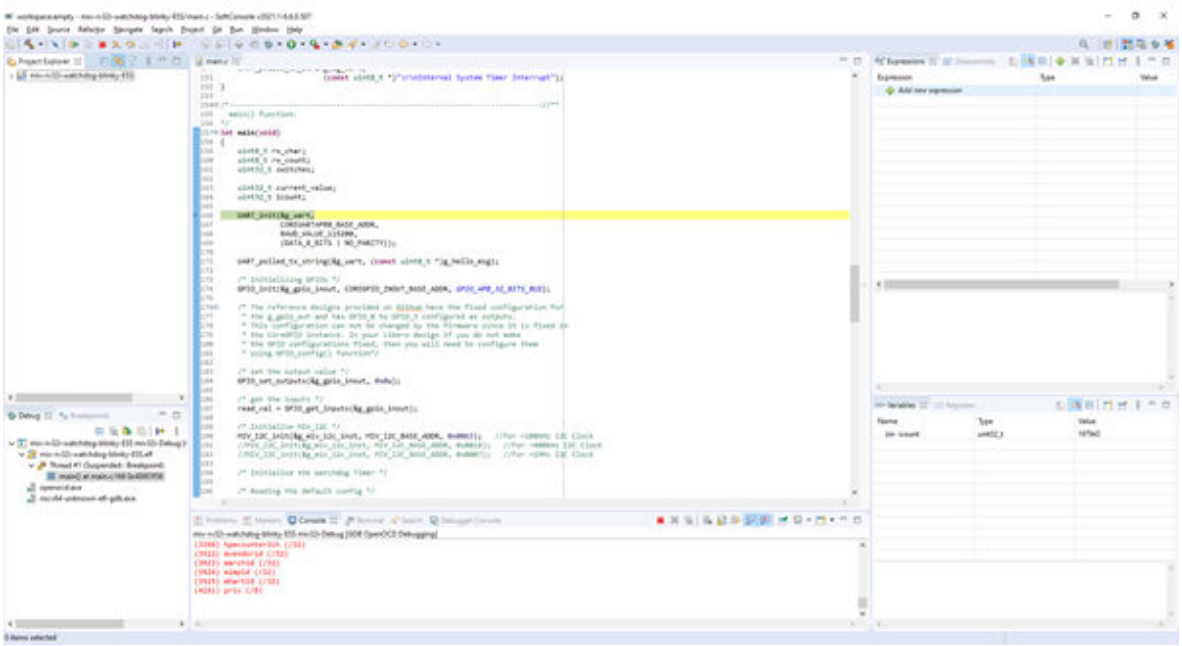
1. Right click on the project and select (for example) **Build Configurations > Set Active > miv32i-Debug**, as shown in the following figure.

Figure 3-5. Setting the Build Configurations



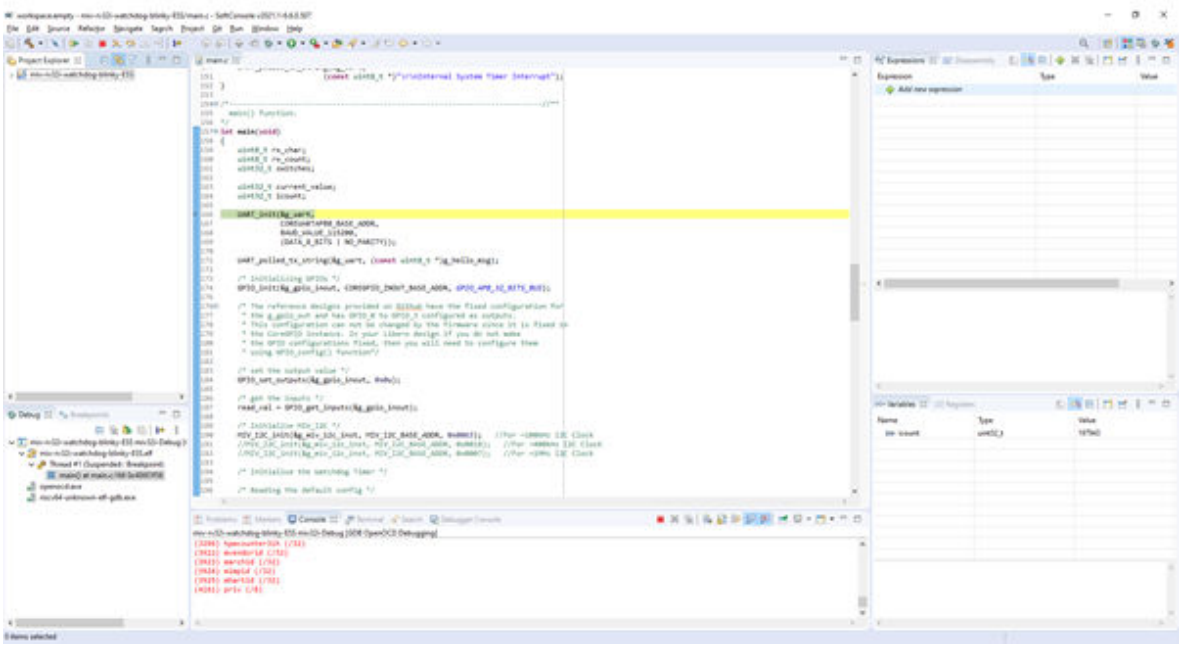
2. Click the **Build** icon. In this case, the project is built for miv32i-Debug.
3. Click the **Debug** icon.
4. If you are debugging this project for the first time, you must select a debug configuration as shown in the following figure.

Figure 3-6. Selecting Debug Configuration



5. Once you select a debug configuration, click **Debug**. This downloads the program to your device and take you to the debug perspective. The debug session gets launched and connected to the target device.

Figure 3-7. Downloading Program



3.4 Creating a Deployable Hex File

The following steps describe how to create a hex file in SoftConsole.

1. Right click and open the **Project Properties** and select **Project Properties**.
2. Select the **C/C++ Build > Settings > GNU RISC-V Cross Create Flash Image > General**.

3. In the **Other Flags** window, add where the MEMORY_ADDRESS is the base address in the linker. For example,
   `0x40000000`

   `--change-section-lma *-MEMORY_ADDRESS`
4. Build the project.
5. Click **Apply** and **Close**.

A build configuration is created, which allows you to create deployable builds without repeating this step. For more information, see the *SoftConsole Release Notes*.

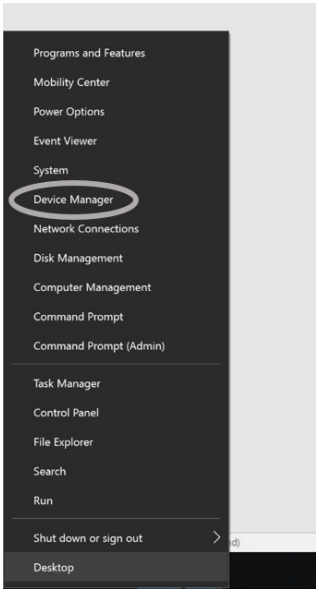
3.5 Communication Through UART

To communicate through UART, you need the following.

- The number of the COM port you are using and its baud rate.
- A serial communication client (the built-in terminal in SoftConsole is used in this example).

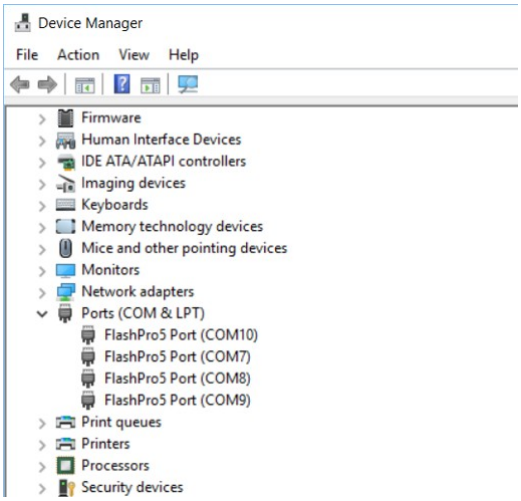
For example, you can use **Device Manager** on Windows to find your device's COM port. To open it, right click the **Start** button and select **Device Manager**.

Figure 3-8. Opening Device Manager



Note:
Your device's COM port is one of the ports listed under **Ports (COM & LPT)**.

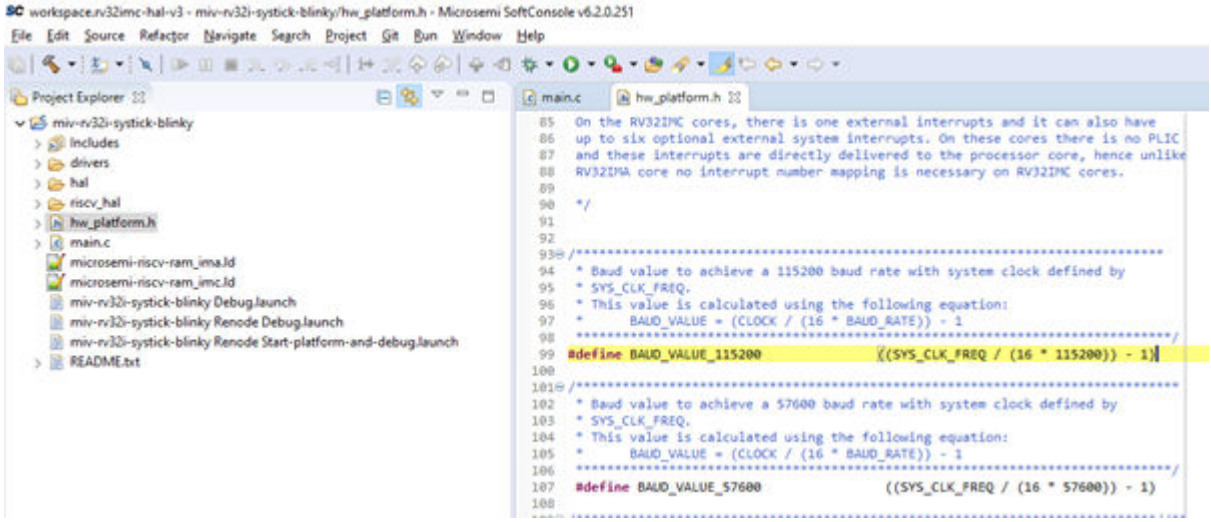
Figure 3-9. COM Ports



Note:
You might have to try them all to find your device's port.

You can find the baud rate for the UART connection in the `fpga_design_config.h` file within the SoftConsole project, see Setting the System Clock Frequency.

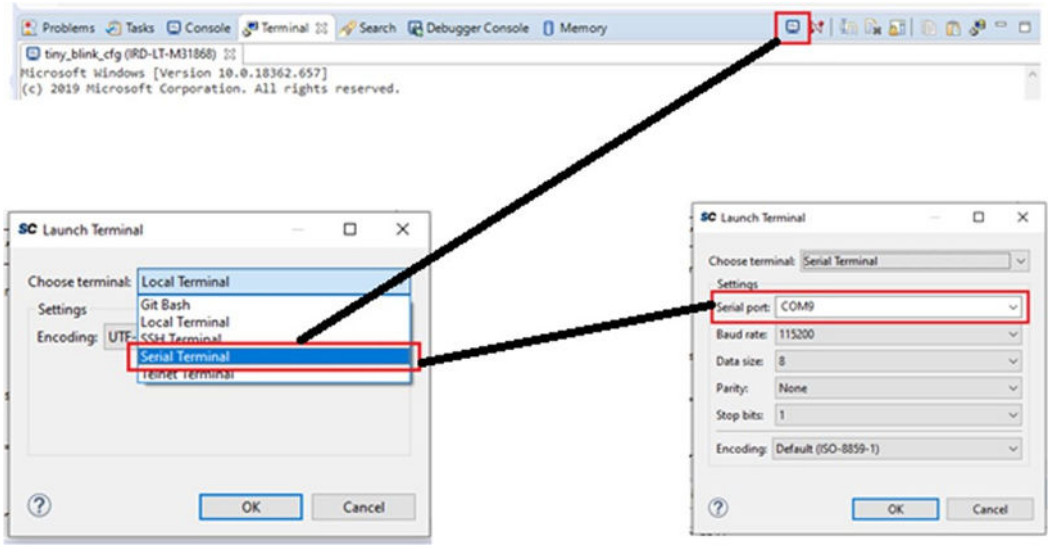
Figure 3-10. Baud Rate Location



The following steps describe how to communicate through UART.

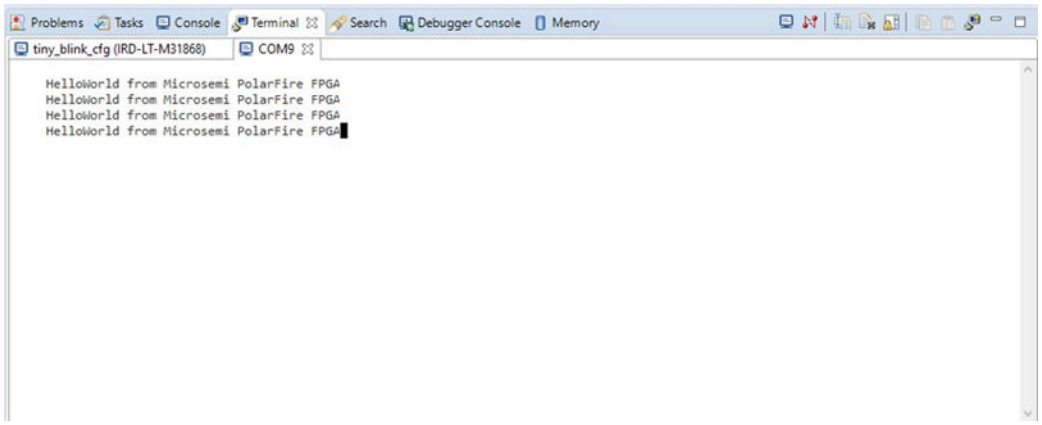
1. Open the serial communication client within SoftConsole by navigating to **Window > Show View > Terminal**.
2. Input the values for your COM port and baud rate, as shown in the following figure.

Figure 3-11. Opening a Terminal



3. Run your program and the UART output will be displayed in the communication client's terminal, as shown in the following figure.

Figure 3-12. FPGA UART Output



4. Appendix: Design Constraints

The following are constraints for the PolarFire Evaluation Kit design. The pin constraints can be modified for other development kits. See the example Libero SoC designs on GitHub for more example design constraints.

```
# -- User PushButtons I/O -- #
set_io -port_name {GPIO_IN[1]} \
 -pin_name B27 \
 -fixed true \
 -DIRECTION INPUT
set_io -port_name {GPIO_IN[0]} \
 -pin_name C21 \
 -fixed true \
 -DIRECTION INPUT
set_io -port_name DEVRST_N \
 -pin_name K22 \
 -fixed true \
 -DIRECTION INPUT
# -- LEDs I/O --#
set_io -port_name {GPIO_OUT[0]} \
 -pin_name F22 \
 -fixed true \
 -DIRECTION OUTPUT
set_io -port_name {GPIO_OUT[1]} \
 -pin_name B26 \
 -fixed true \
 -DIRECTION OUTPUT
set_io -port_name {GPIO_OUT[2]} \
 -pin_name C26 \
 -fixed true \
 -DIRECTION OUTPUT
set_io -port_name {GPIO_OUT[3]} \
 -pin_name D25 \
 -fixed true \
 -DIRECTION OUTPUT
# -- UART RX/TX -- #
set_io -port_name RX \
 -pin_name H18 \
 -fixed true \
 -DIRECTION INPUT
set_io -port_name TX \
 -pin_name G17 \
 -fixed true \
 -DIRECTION OUTPUT
```

The following steps describe how to add constraints.

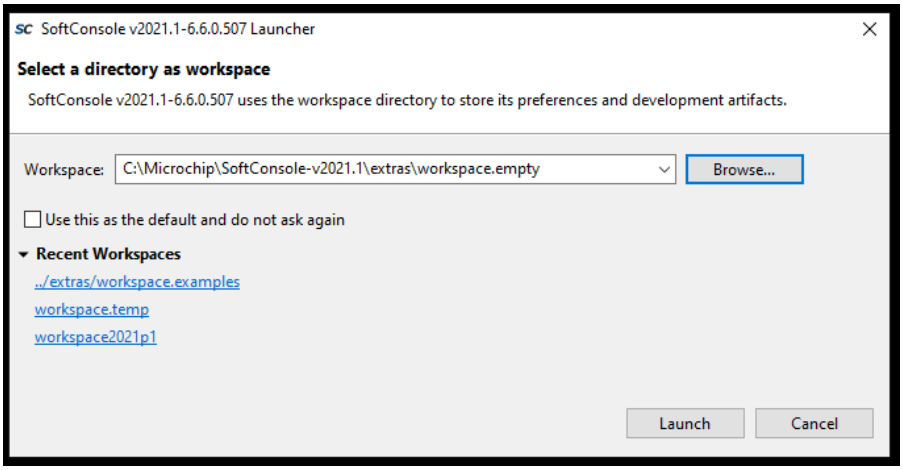
1. Add **Timing Constraints** to the **Design Flow** tab and select **Manage Constraints**.
2. Under the **Timing** tab, select **New** and name the file as `io_jtag_constraints`.
3. Copy the following constraints into this file.
   ```
   #Constraining the JTAG clock to 6 MHz

   create_clock -name {TCK} -period 166.67 -waveform {0 83.33} [ get_ports { TCK } ]
   set_clock_groups -name {async1} -asynchronous -group [ get_clocks { PF_CCC_C0_0/
   PF_CCC_C0_0/pll_inst_0/OUT0 } ] -group [ get_clocks { TCK } ]
   ```
4. Save the constraints file, and associate it with Synthesis, Place-and-Route, and Timing Verification by selecting each of the check boxes.
5. To add the derived constraints, select **Derive Constraints** in the **Timing** tab.
6. Save the **Timing Constraints**.

5. Appendix: Running an Elf file

Use the following steps to run the provided `Elf` file.

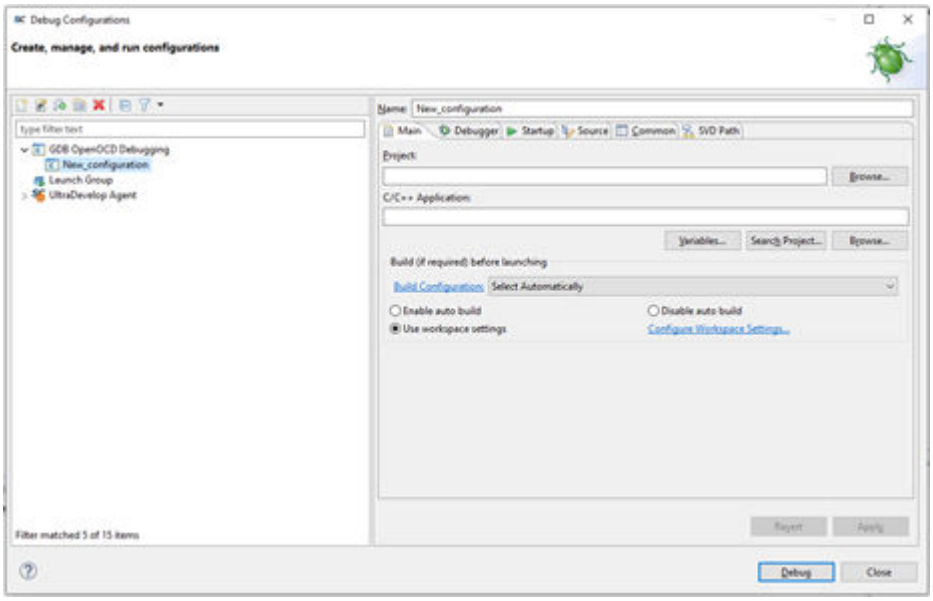
1. Create an empty workspace by extracting the `workspace.empty.zip` file from `<SoftConsole-install-directory>\extras`.
2. Open SoftConsole and locate workspace.empty, which is extracted in the previous step.

   Figure 5-1. Creating a New Workspace



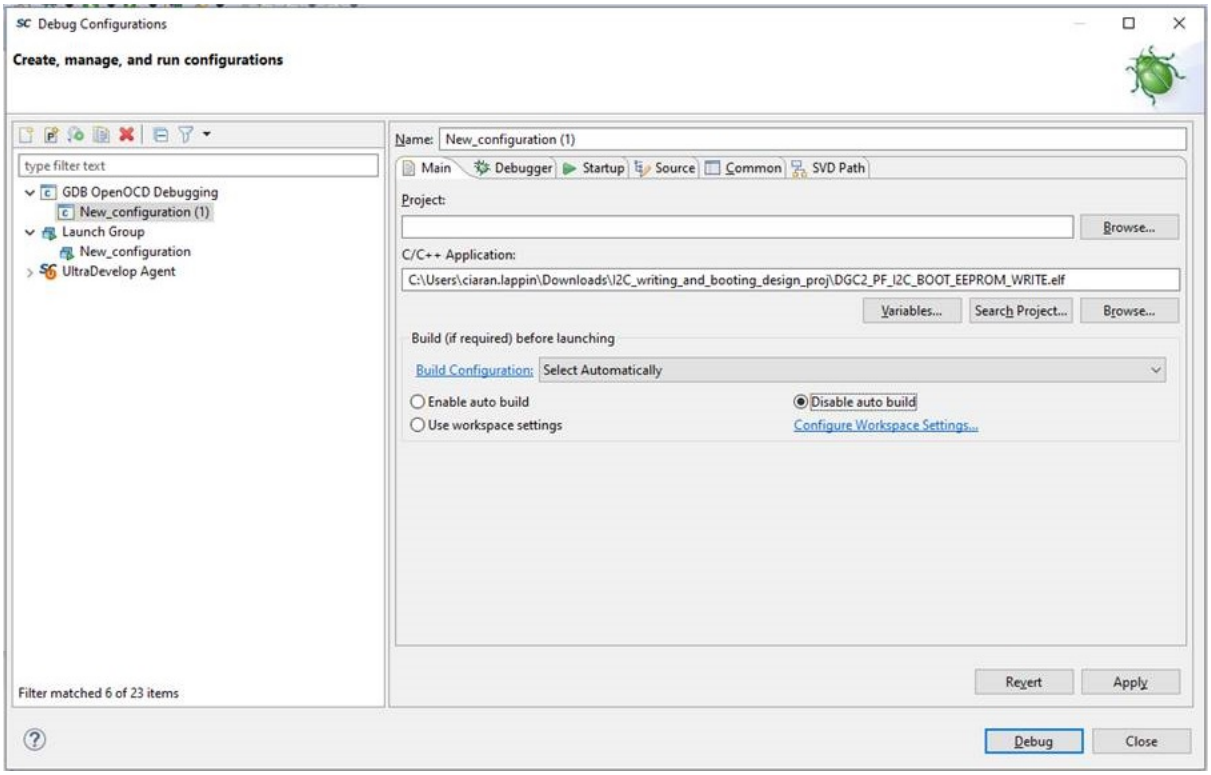
3. Select **Launch**.
4. From the empty workspace, create a new Debug Configuration.
   a. From **Run > Debug Configurations**, double click **GDB OpenOCD Debugging**. This creates a new debug configuration called New_configuration.

   Figure 5-2. Creating a New Debug Configuration



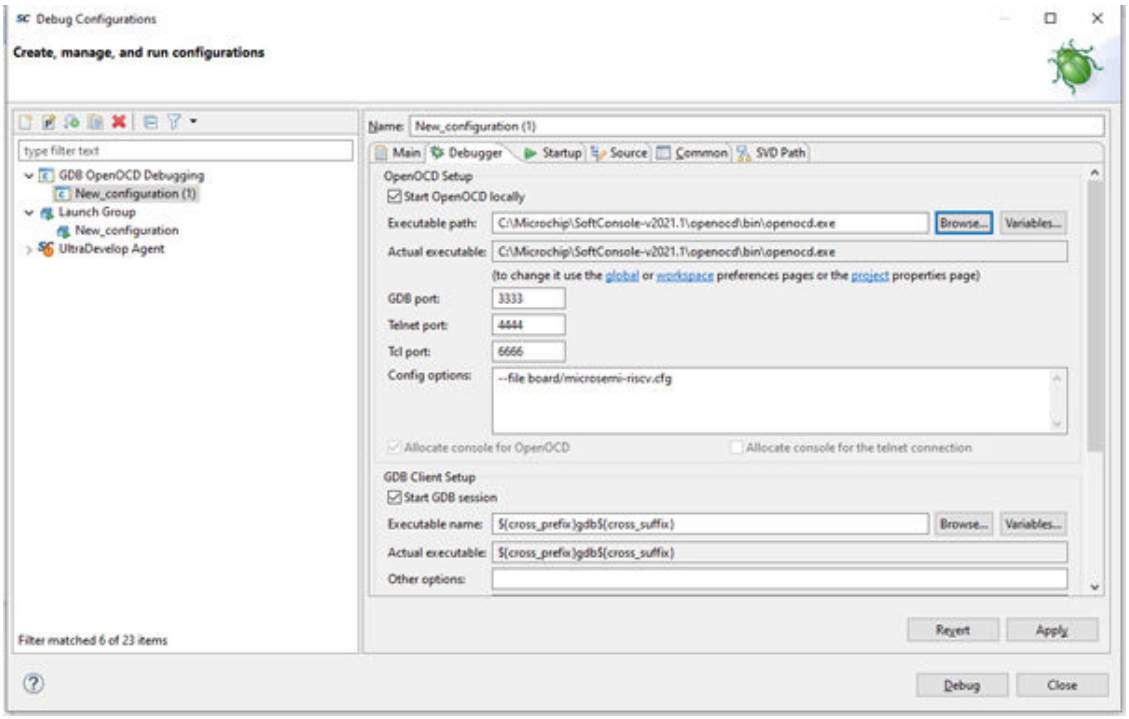
5. In the **Main** window, select **C/C++ Applications** using the **Browse** button, then select the provided **`.elf`** file as shown in the following figure.

Figure 5-3. Selecting the elf File



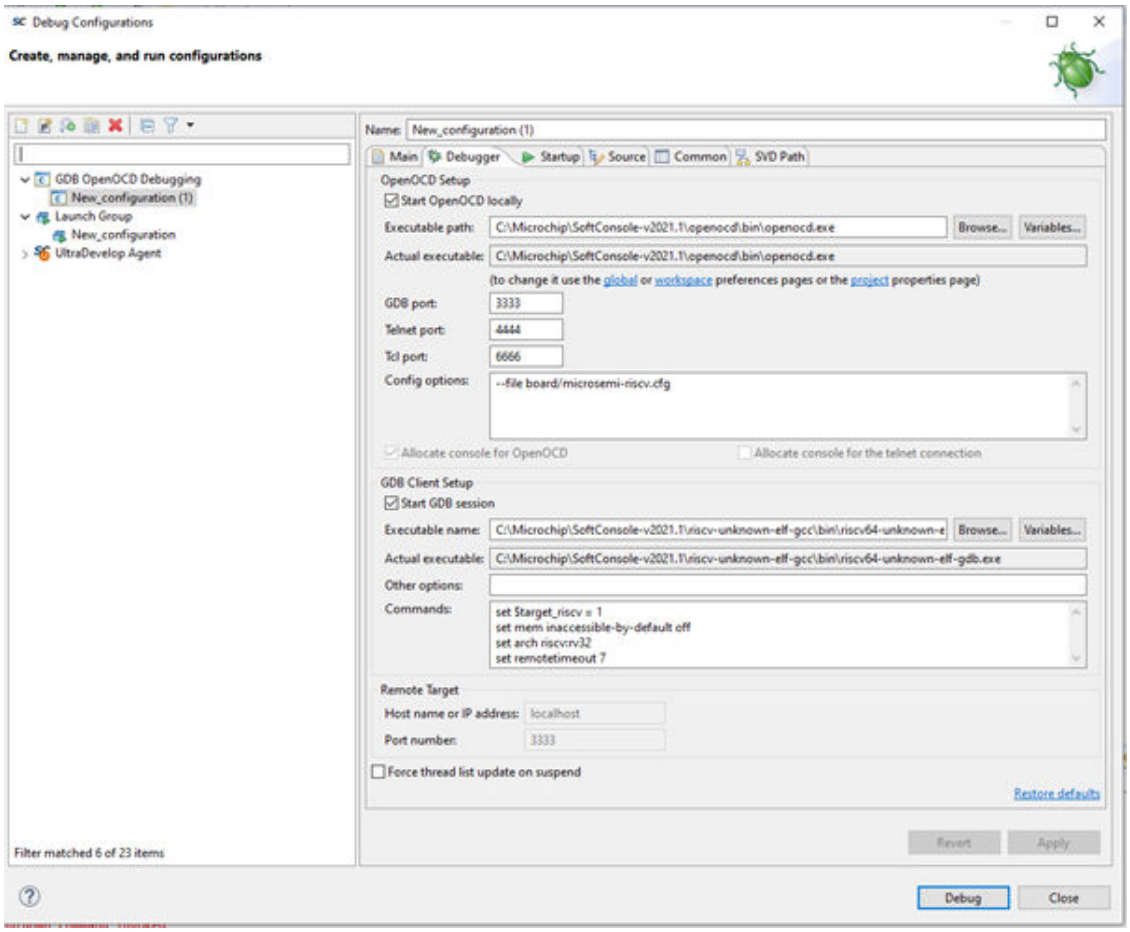
6. Select the **Debugger** tab to set up OpenOCD and GDB.
   a. To set up OpenOCD, check the **Start OpenOCD locally** check box and browse to the OpenOCD path `<SoftConsole-install-directory>\openocd\bin\openocd.exe` as shown in the following figure.

Figure 5-4. Selecting OpenOCD



b. To set up GDB, browse to the GDB path `<SoftConsole-install-directory>\riscv-unknown-elf-gcc\bin\riscv64-unknown-elf-gdb.exe`.

Figure 5-5. Selecting the GDB Location



7. Click **Apply**.
8. Click **Debug** to launch the debug session.

6. Revision History

Revision	Date	Description
A	01/2022	Initial Revision

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable". Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

Trademarks

ISBN: 978-1-5224-9661-8

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820