# Quick Start Design Guide

# PolarFire MIV_RV32 v3.0

October 2020

Microsemi

a Microchip company

# Contents

# Tables

# Figures

# 1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

## 1.1 Revision 2.0

Revision 2.0 was published in October 2020. The following is a summary of changes.

- Changed the core name to MIV_RV32 from MIV_RV32IMC.
- Updated figures to reflect v3.0 updates.

## 1.2 Revision 1.0

Revision 1.0 was published in March 2020. It is the first publication of this document.

# 2 Introduction

This guide provides quick-start steps to set up the MIV_RV32 IP core in Libero, generate a bitstream, and download the FPGA data to a device. Debug software and use a serial terminal to capture UART output in SoftConsole. This guide is targeted towards the PolarFire Evaluation Kit. Note that the MIV_RV32 IP core is written in System Verilog.

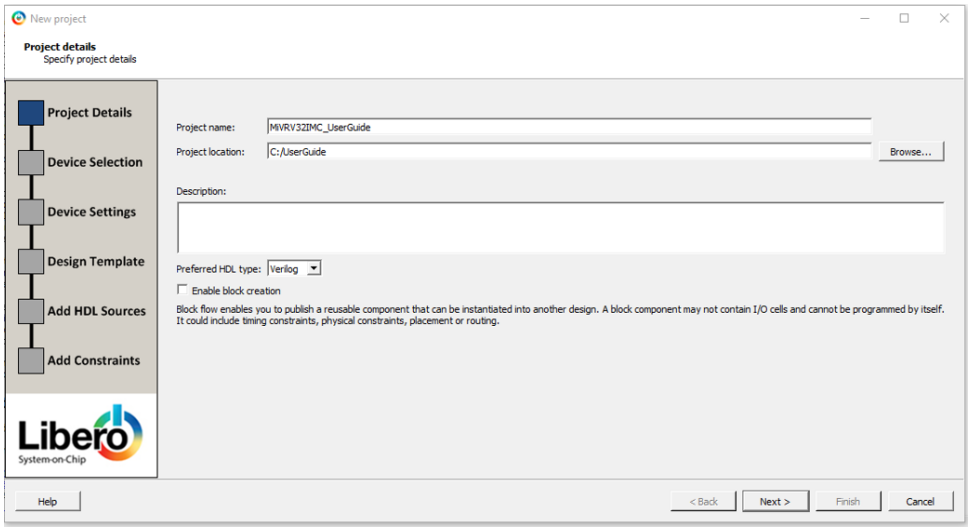
# 3 Create a Libero Project

Open Libero and select **Project** > **New Project** from the top menu.

**Figure 1 • Create a Libero Project**


From Project Details, enter **Project name** and **Project location** then click **Next**.

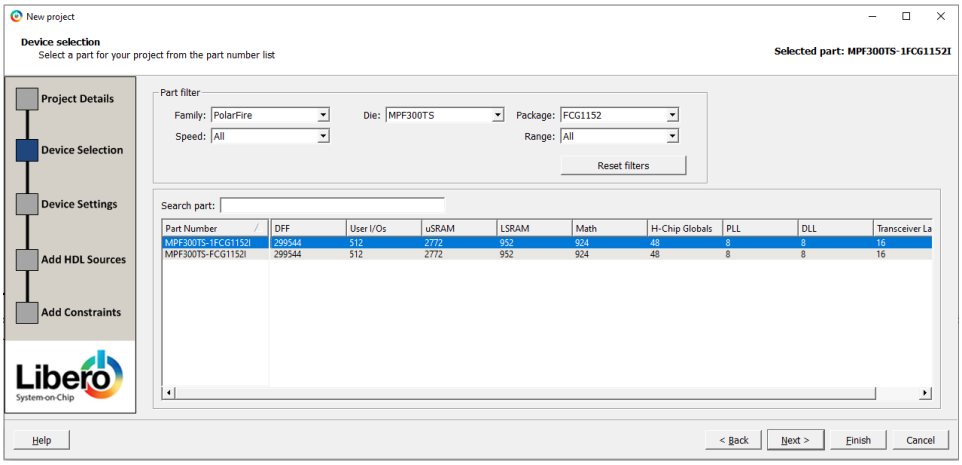
**Figure 2 • Project Name and Location**




From Device Selection, for example select **MPF300TS_1FCG1152I used on the PolarFire Evaluation Board**. The following Part Filter search terms can be used to find the correct device.

- Family: PolarFire
- Die: MPF300TS
- Package: FCG1152I
- Range: IND

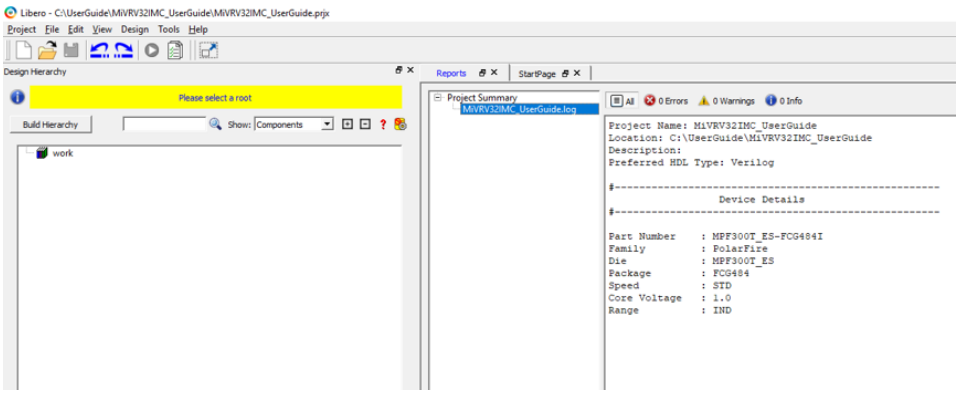
**Figure 3 • Select Device and Part**




Click **Finish**.

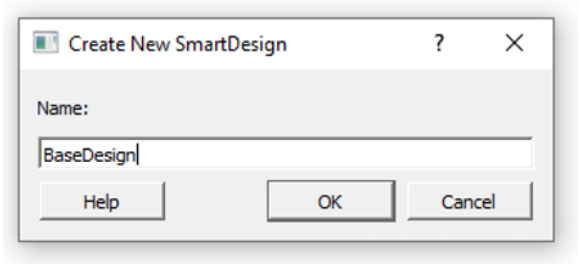
The following New Project Home Screen will appear.

**Figure 4 • New Project Home Screen**



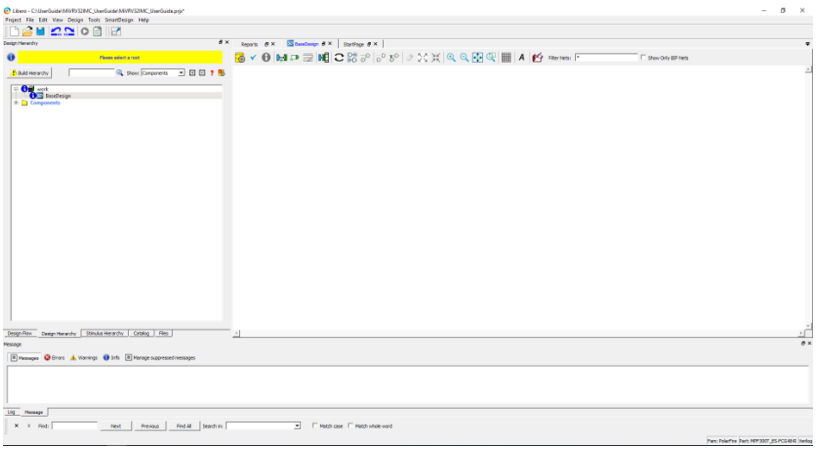

Create a new SmartDesign by selecting **File** > **New** > **SmartDesign** and name it BaseDesign.

**Figure 5 • Create a New SmartDesign**




Once the SmartDesign has been created, a blank canvas will appear where the example design will be created.

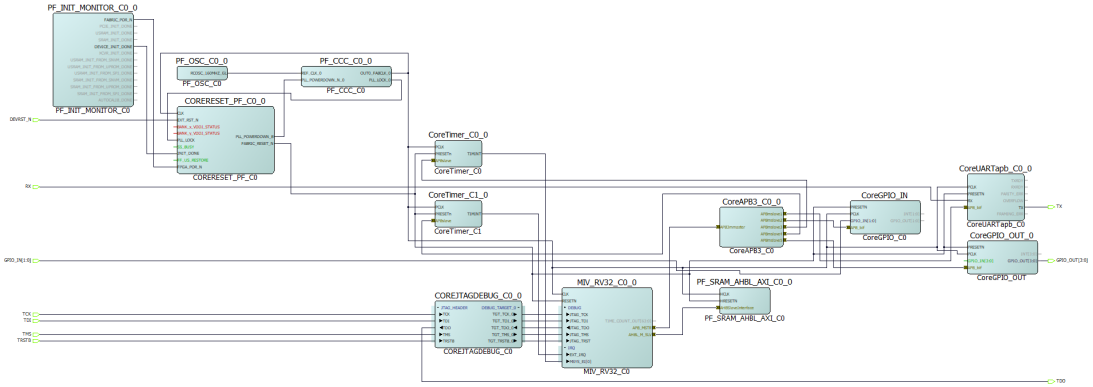
**Figure 6 • New SmartDesign Canvas**



## 3.1 Create the Core SmartDesign

The following design can be produced using this document.

**Figure 7 • Design Segment**




**Components needed:**

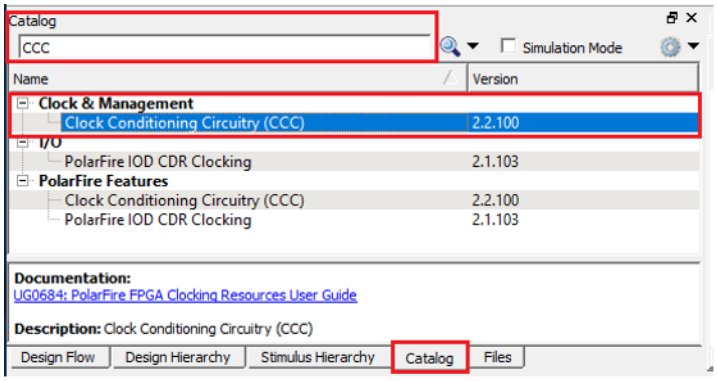
- 1 × PolarFire RC Oscillators (OSC) (v1.0.102)
- 1 × Clock Conditioning Circuit (CCC) (v2.2.100)
- 1 × CORERESET_PF (v2.2.107)
- 1 × PF_INIT_MONITOR (v2.0.105)
- 1 × CoreJTAGDebug (v3.1.100)
- 1 × MIV_RV32 (v3.0.100)
- 1 × PF_SRAM_AHBL_AXI (v1.2.105)
- 1 × CoreAPB3 (v4.1.100)
- 1 × CoreUARTapb (v5.6.102)
- 2 × CoreGPIO (GPIO_IN and GPIO_OUT) (v3.2.102)
- 2 × CoreTimer (v2.0.103)

**Note:** These are the latest components at the time of publication, please always use the latest available components from the catalog.

## 3.2 Clock Conditioning Circuit Configuration

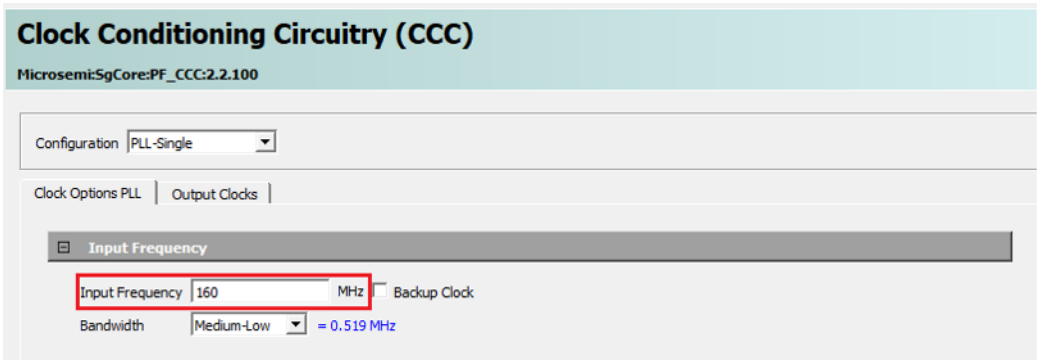
To add the Clock Conditioning Circuitry to the SmartDesign, select the **Catalog** tab, search for **CCC**, and double click on it. A dialog box will appear with the default name PF_CCC_0. Select **OK**.

**Figure 8 • Create a Component for SmartDesign**



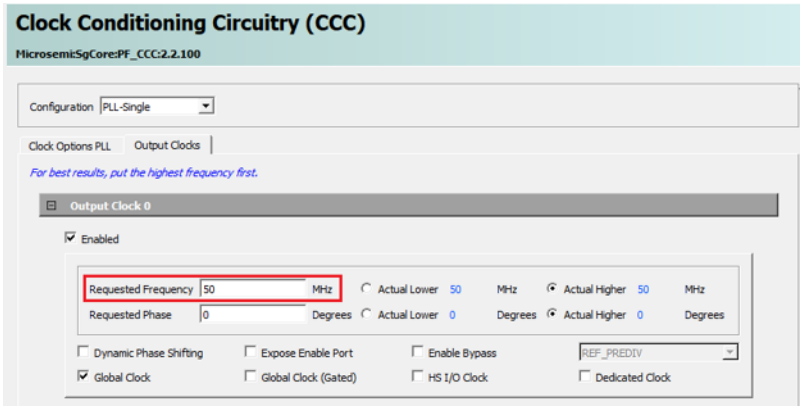

The following Clock Conditioning Circuitry (CCC) configuration menu will open. Change the clock **Input Frequency** to **160 MHz**.

**Figure 9 • Clock Conditioning Circuitry (CCC) Configuration Menu**




Under Output Clock 0, change **Requested Frequency** to **50 MHz**.

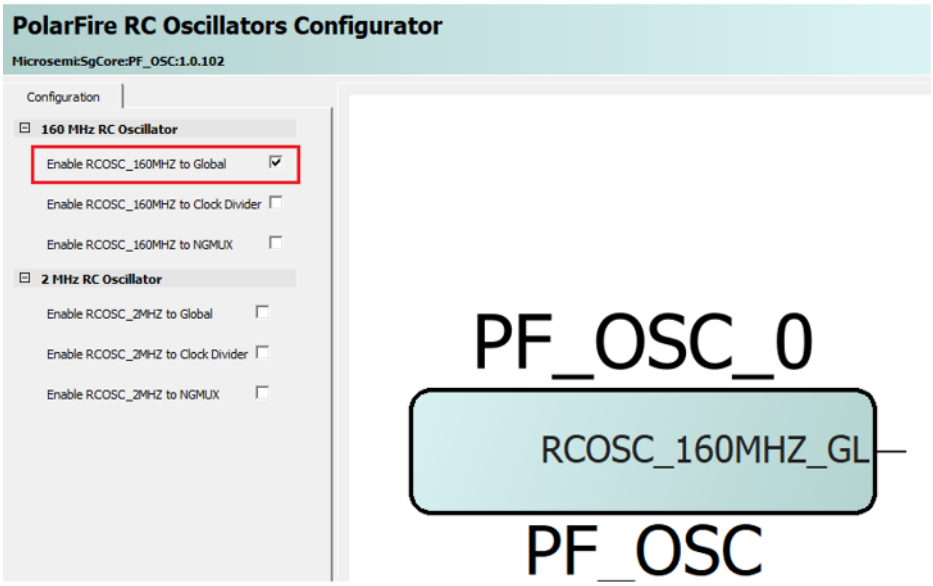
**Figure 10 • Change Requested Frequency to 50 MHz**



## 3.3 PolarFire RC Oscillator Configuration

To add the PolarFire RC Oscillator to the SmartDesign, select the **Catalog** and search for **OSC**. Double click on the **PolarFire RC Oscillator** to instantiate it in the SmartDesign. The default configuration should be used.

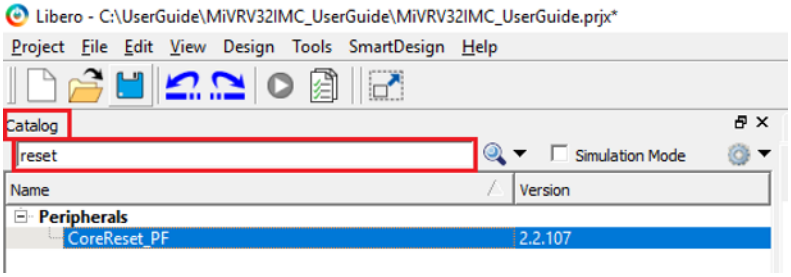
**Figure 11 • Enable 160 MHz Oscillator**




## 3.4 CoreRESET_PF Configuration

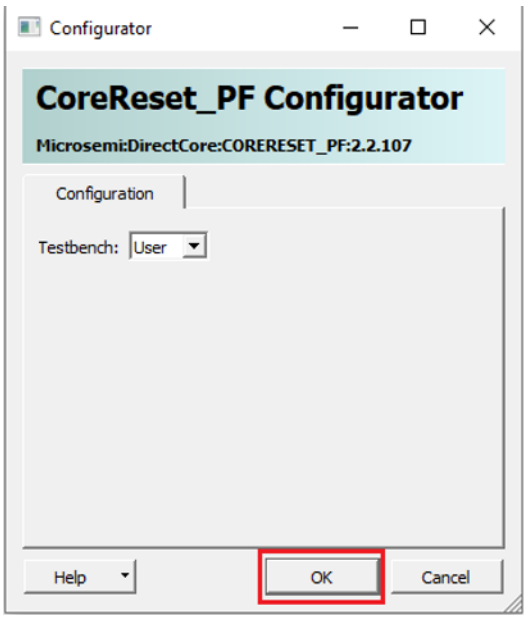
To add the CORERESET_PF to the SmartDesign, select the **Catalog** and search for **reset**. Double click on **CoreReset_PF** to instantiate it into the SmartDesign.

**Figure 12 • Instantiating CoreRESET_PF**




The default configuration for CoreRESET_PF can be used.

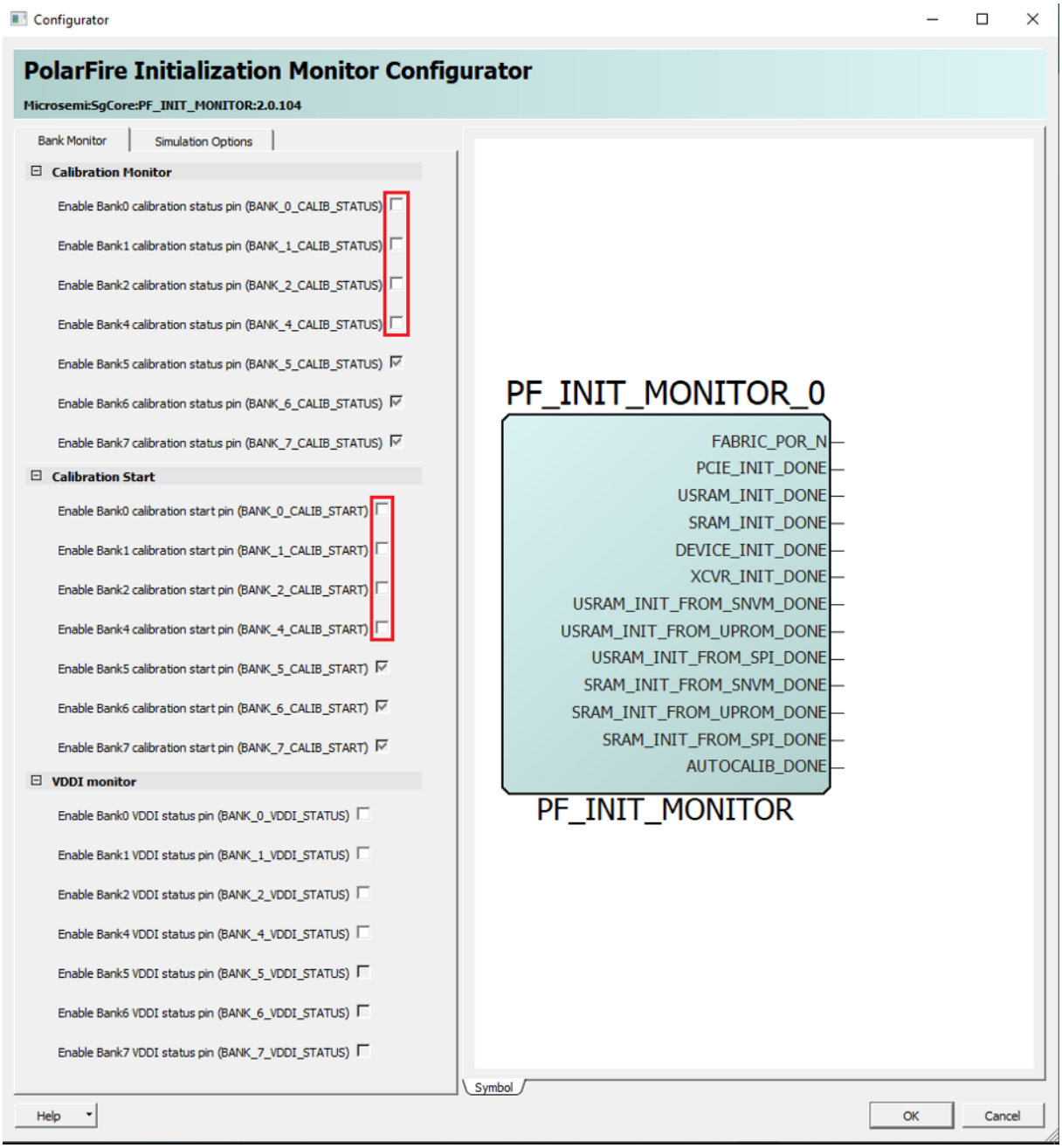
**Figure 13 • CoreRESET_PF Configurator Default Settings**




## 3.5 PolarFire Initialization Monitor Configuration

Search for monitor in the catalog and double click **PolarFire Initialisation Monitor** to instantiate in the SmartDesign. Uncheck the **Calibration Monitor** and **Calibration Start** boxes as shown in the following figure.

**Figure 14 • PolarFire Initialization Monitor Configurator Required Settings**




## 3.6 MIV_RV32 Configuration

To add the MIV_RV32 to the SmartDesign, search for **MIV RV32** in the catalog and double click to instantiate. The default configuration is shown in the following figure.

**Figure 15 • Instantiate MIV_RV32 Core**

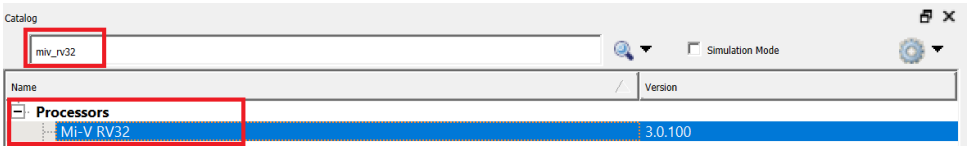
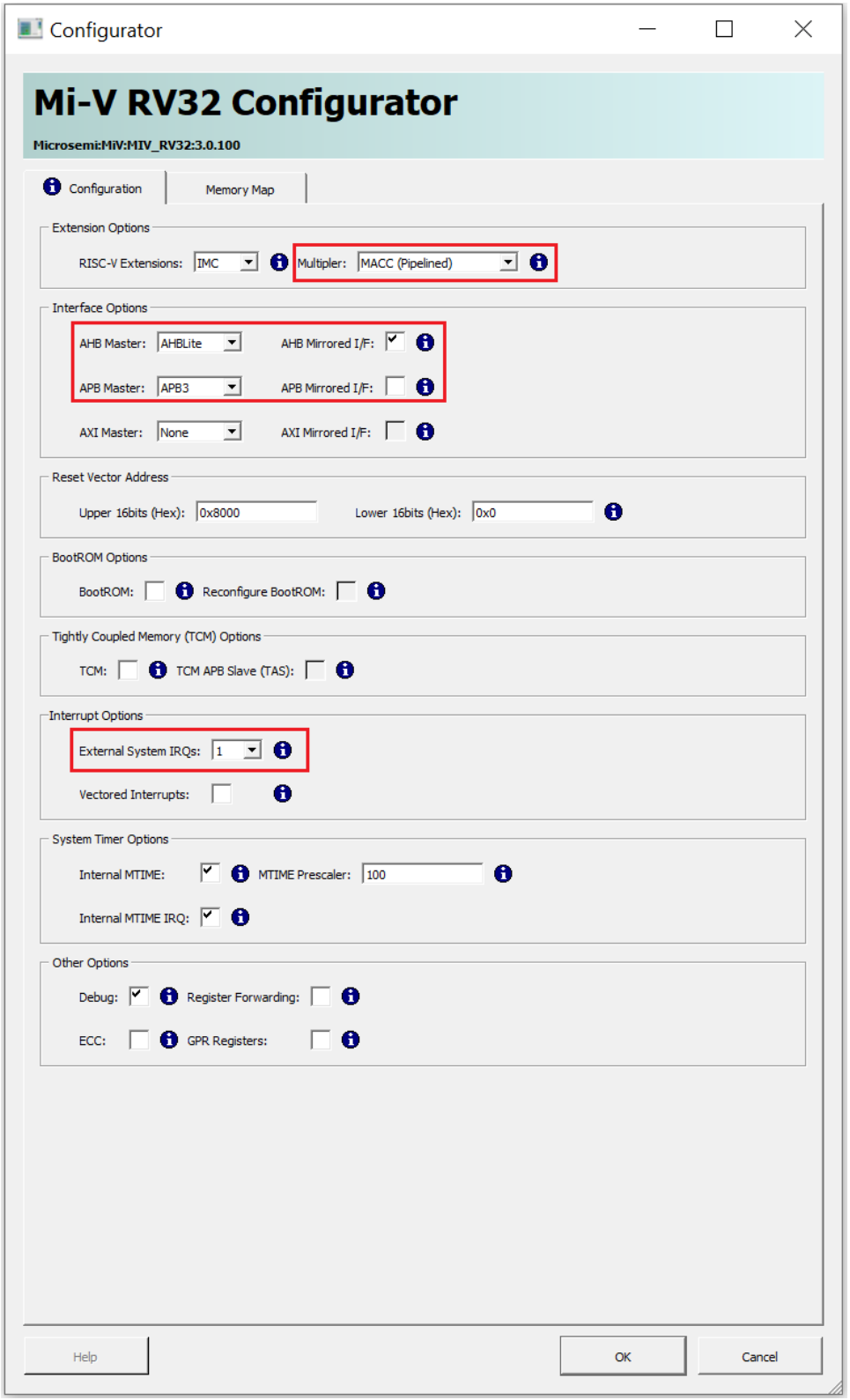
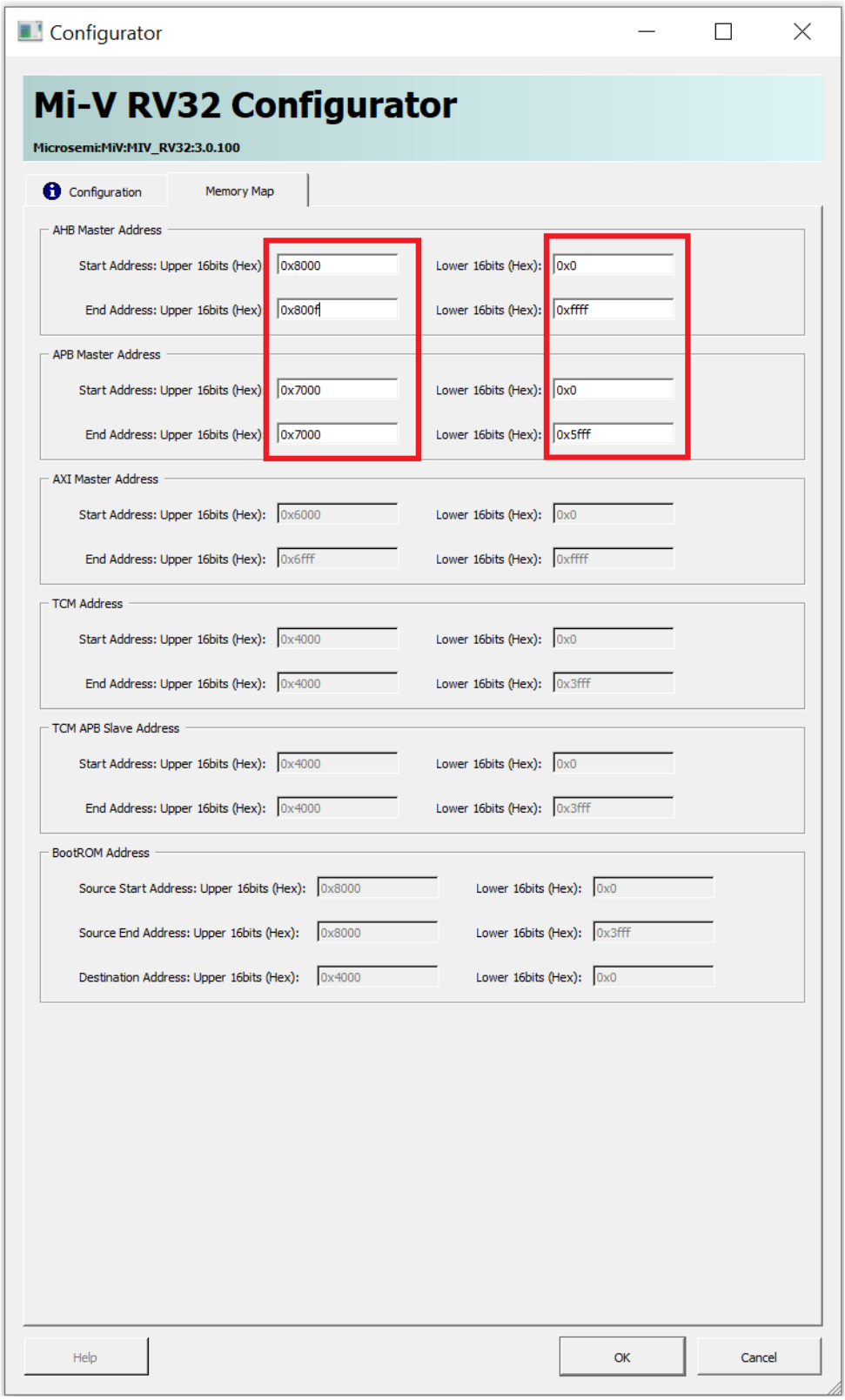


**Figure 16 • MIV_RV32 Configurator Setup**




The memory map is set to the following address range.

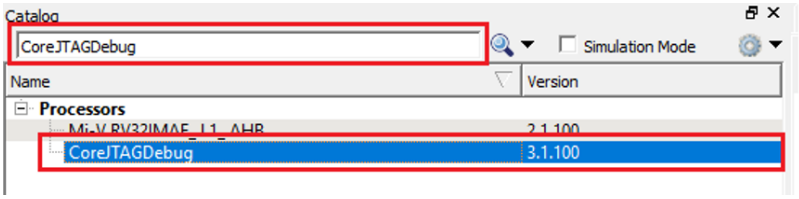
**Figure 17 • Update the MIV_RV32 Memory Map**



## 3.7 CoreJTAGDebug Configuration

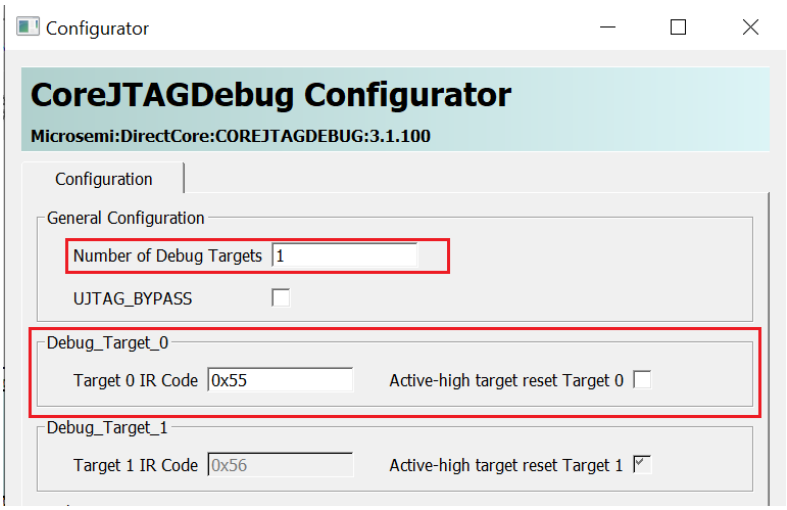
To add the CoreJTAGDebug block to the SmartDesign, search CoreJTAGDebug in the catalog and double click to instantiate.

**Figure 18 • Instantiate CoreJTAGDebug**




The default configuration for CoreJTAGDebug can be used.

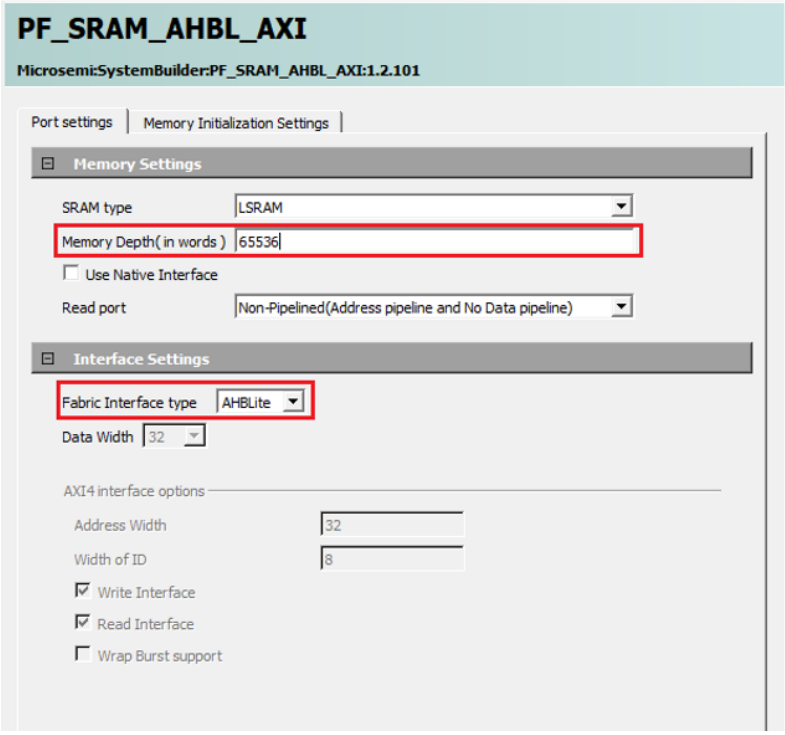
**Figure 19 • CoreJTAGDebug Configurator Default Settings**




## 3.8 Memory and Peripheral Configurations—PolarFire SRAM

To add the PolarFire SRAM to the SmartDesign, search for sram and select PolarFire SRAM (AHBLite and AXI). Under Memory Settings, configure **Memory Depth (in words)** to **65536**. Under Interface Settings, configure **Fabric Interface type** to **AHBLite**. The MIV_RV32 core is connected directly to the PF_SRAM_AHBL_AXI block.

**Figure 20 • PF_SRAM_AHBL_AXI Configuration**



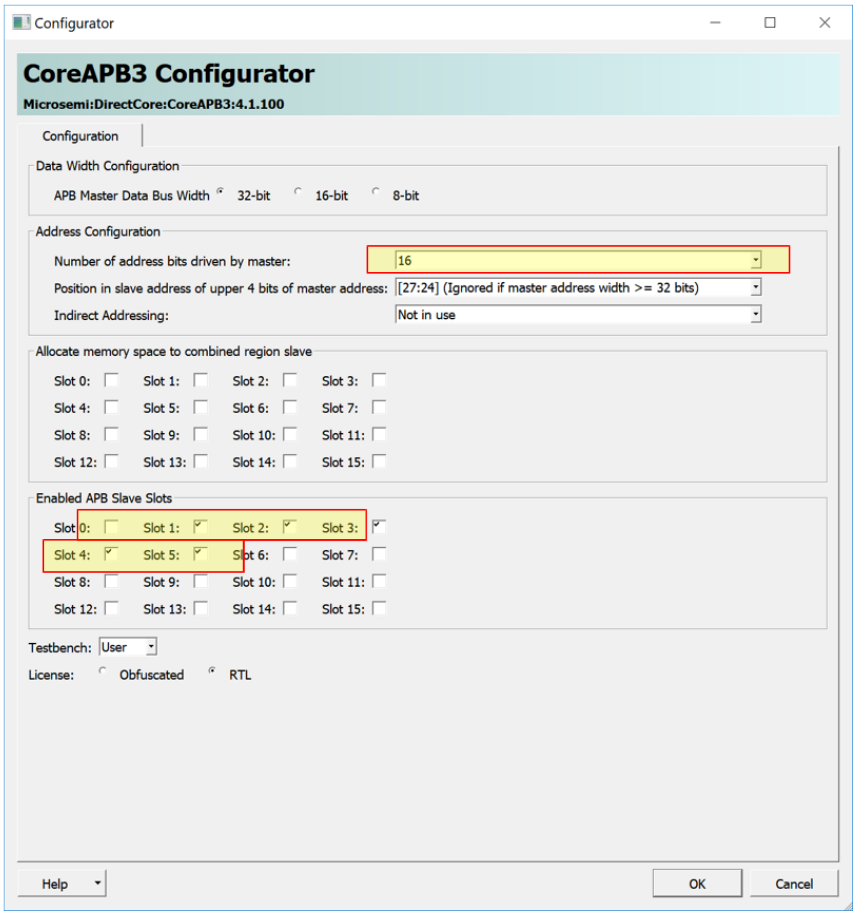
# 4 Peripheral Connections

The following section describes the peripheral configurations and connections for the design.

## 4.1 CoreAPB3 Configuration

To add the CoreAPB3 to the SmartDesign, search for coreapb3 and double-click to instantiate it in the SmartDesign. Configure the core to set the **Number of address bits driven by master** to **16** and under **Enable APB Slots** check **Slots 1, 2, 3, 4, and 5**. CoreAPB3 can be connected directly to the core.

**Figure 21 • CoreAPB3 Required Configuration**




## 4.2 CoreUARTapb Configuration

To add CoreUARTapb to the SmartDesign, search for CoreUARTapb in the catalog and double click to instantiate. The default name and configuration for CoreUARTapb can be used.

## 4.3 CoreTimer 0 Configuration

To add CoreTimer_0 to the SmartDesign search for CoreTimer in the catalog and double click to instantiate. The default name and configuration for CoreTimer can be used.

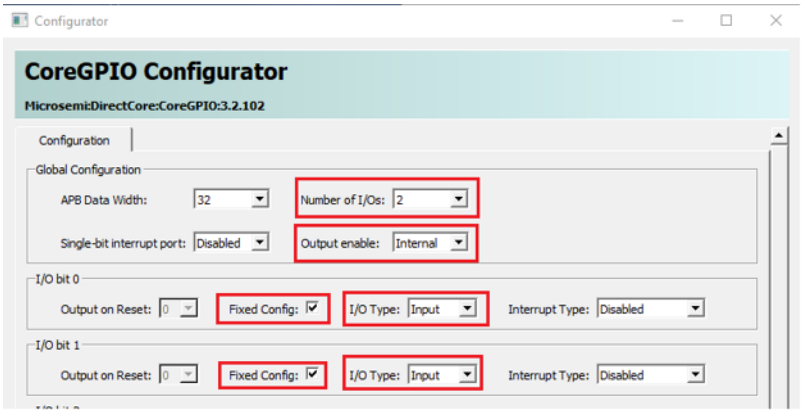
## 4.4 CoreTimer 1 Configuration

To add CoreTimer_1 to the SmartDesign search for CoreTimer in the catalog and double click to instantiate. The default name and configuration for CoreTimer can be used.

## 4.5 CoreGPIO_IN Configuration

To add CoreGPIO to the SmartDesign search for CoreGPIO in the catalog and double click to instantiate. Enter the component name as CoreGPIO_IN. This CoreGPIO is configured to have two fixed configured inputs as shown in the following figure.

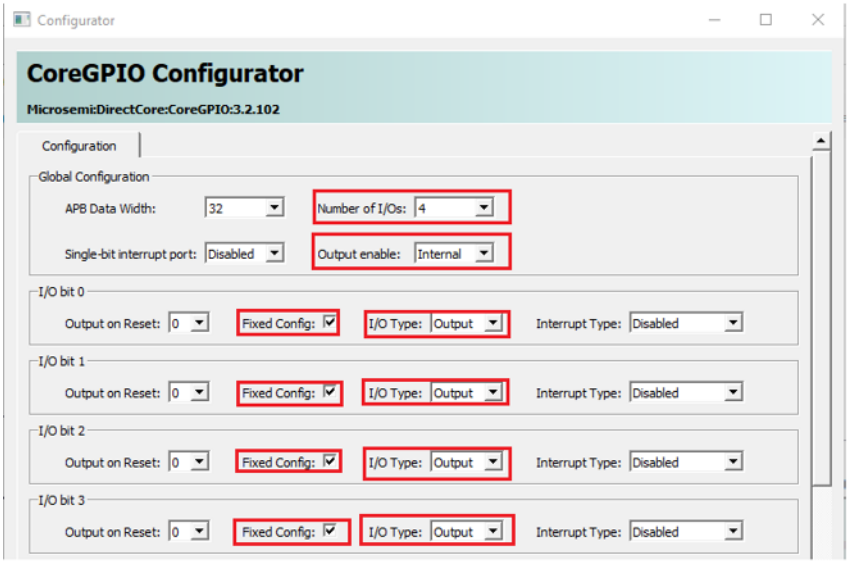
**Figure 22 • CoreGPIO_IN Configuration**




## 4.6 CoreGPIO_OUT Configuration

To add CoreGPIO to the SmartDesign search for CoreGPIO in the catalog and double click to instantiate. Enter the component name as CoreGPIO_OUT. This CoreGPIO is configured to have four fixed configured outputs as shown in the following figure.

**Figure 23 • CoreGPIO_OUT Configuration**



## 4.7 Component Connections List

The following tables list each component and each component it is connected to.

**Table 1 • PF_OSC_C0_0**

| Connect From | Connect To |
|---|---|
| PF_OSC_C0_0:RCOSC_160MHZ_GL | PF_CCC_C0_0:REF_CLK_0 |

**Table 2 • PF_CCC_C0_0**

| Connect From | Connect To |
|---|---|
| PF_OSC_C0_0:OUT0_FABCLK_0 | CORERESET_PF_C0_0:CLK |
| | MIV_RV32_C0_0:clk |
| | PF_SRAM_AHBL_AXI_C0_0:HCLK |
| | COREUARTAPB_C0_0:PCLK |
| | CORETIMER_C0_0:PCLK |
| | CORETIMER_C1_0:PCLK |
| | COREGPIO_IN_0:PCLK |
| | COREGPIO_OUT_0:PCLK |
| PF_CCC_C0_0:PLL_LOCK_0 | CORERESET_PF_C0_0:PLL_LOCK |

**Table 3 • PolarFire Initialization Monitor**

| Connect From | Connect To |
|---|---|
| PF_INIT_MONITOR_C0_0:FABRIC_POR_N | CORERESET_PF_C0_0:FPGA_POR_N |
| PF_INIT_MONITOR_C0_0:DEVICE_INIT_DONE | CORERESET_PF_C0_0:INIT_DONE |
| PF_INIT_MONITOR_C0_0:PCIE_INIT_DONE | Mark Unused |
| PF_INIT_MONITOR_C0_0:SRAM_INIT_DONE | Mark Unused |
| PF_INIT_MONITOR_C0_0:XCVR_INIT_DONE | Mark Unused |
| PF_INIT_MONITOR_C0_0:USRAM_INIT_FROM_SNVM_DONE | Mark Unused |
| PF_INIT_MONITOR_C0_0:USRAM_INIT_FROM_UPROM_DONE | Mark Unused |
| PF_INIT_MONITOR_C0_0:USRAM_INIT_FROM_SPI_DONE | Mark Unused |
| PF_INIT_MONITOR_C0_0:SRAM_INIT_FROM_SNVM_DONE | Mark Unused |
| PF_INIT_MONITOR_C0_0:SRAM_INIT_FROM_UPROM_DONE | Mark Unused |
| PF_INIT_MONITOR_C0_0:SRAM_INIT_FROM_SPI_DONE | Mark Unused |

| Connect From | Connect To |
|---|---|
| PF_INIT_MONITOR_C0_0:AUTOCALIB_DONE | Mark Unused |

**Table 4 • MIV_RV32**

| Connect From | Connect To |
|---|---|
| MIV_RV32_C0_0:JTAG_TCK | COREJTAGDEBUG_C0_0:TGT_TCK_0 |
| MIV_RV32_C0_0:JTAG_TDI | COREJTAGDEBUG_C0_0:TGT_TDI_0 |
| MIV_RV32_C0_0:JTAG_TDO | COREJTAGDEBUG_C0_0:TGT_TDO_0 |
| MIV_RV32_C0_0:JTAG_TMS | COREJTAGDEBUG_C0_0:TGT_TMS_0 |
| MIV_RV32_C0_0:JTAG_TRSTN | COREJTAGDEBUG_C0_0:TGT_TRSTN_0 |
| MIV_RV32_C0_0:EXT_RESETN | Mark unused |
| MIV_RV32_C0_0:JTAG_TDO_DR | Mark unused |
| MIV_RV32_C0_0:EXT_IRQ | CORETIMER_C1_0:TIMINT |
| MIV_RV32_C0_0:MSYS_EI[5:1] | Tie low |
| MIV_RV32_C0_0:MSYS_EI[0] | CORETIMER_C0_0:TIMINT |
| MIV_RV32_C0_0:TIME_COUNT_OUT[63:0] | Mark unused |
| MIV_RV32_C0_0:AHBL_M_SLV | PF_SRAM_AHBL_AXI_C0_0:AHBSLAVEINTERFACE |
| MIV_RV32_C0_0:APB_MSTR | COREAPB3_C0_0:APB3MASTER |

**Table 5 • CoreRESET_PF**

| Connect From | Connect To |
|---|---|
| CORERESET_PF_C0_0:EXT_RSTN | Promote to top level and rename to DEVRST_N. To rename the input port EXT_RSTN, right-click the port and select Modify/R-ename, a text box will open. Enter DEVRST_N and select OK. |
| CORERESET_PF_C0_0:BANK_x_VDDI_STATUS | Tie high |
| CORERESET_PF_C0_0:BANK_y_VDDI_STATUS | Tie high |
| CORERESET_PF_C0_0:SS_BUSY | Tie low |
| CORERESET_PF_C0_0:FF_US_RESTORE | Tie low |
| CORERESET_PF_C0_0:PLL_POWERDOWN_B | Mark unused |
| CORERESET_PF_C0_0:FABRIC_RESET_N | MIV_RV32_C0_0:RESETN |
| | PF_SRAM_AHBL_AXI_C0_0:HRESETN |
| | COREUARTAPB_C0_0:PRESETN |

| Connect From | Connect To |
|---|---|
| | CORETIMER_C0_0:PRESETN |
| | CORETIMER_C1_0:PRESETN |
| | COREGPIO_IN_0:PRESETN |
| | COREGPIO_OUT_0:PRESETN |

**Table 6 • CoreAPB3_C0_0**

| Connect From | Connect To |
|---|---|
| COREAPB3_C0_0:APBMSLAVE1 | COREUARTAPB_C0_0:APB_BIF |
| COREAPB3_C0_0:APBMSLAVE2 | GOREGPIO_IN_0:APB_BIF |
| COREAPB3_C0_0:APBMSLAVE3 | CORETIMER_C0_0:APBSLAVE |
| COREAPB3_C0_0:APBMSLAVE4 | CORETIMER_C1_0:APBSLAVE |
| COREAPB3_C0_0:APBMSLAVE5 | COREGPIO_OUT_0:APB_BIF |

**Table 7 • CoreUARTapb_C0_0**

| Connect From | Connect To |
|---|---|
| COREUARTAPB_C0_0:RX | Promote to top level |
| COREUARTAPB_C0_0:TX | Promote to top level |
| COREUARTAPB_C0_0:TXRDY | Mark unused |
| COREUARTAPB_C0_0:RXRDY | Mark unused |
| COREUARTAPB_C0_0:PARITY_ERR | Mark unused |
| COREUARTAPB_C0_0:OVERFLOW | Mark unused |
| COREUARTAPB_C0_0:FRAMING_ERR | Mark unused |

**Table 8 • CoreGPIO_IN_0**

| Connect From | Connect To |
|---|---|
| COREGPIO_IN_0:GPIO_IN[1:0] | Promote to top level |
| COREGPIO_IN_0:INT[1:0] | Mark unused |
| COREGPIO_IN_0:GPIO_OUT[1:0] | Mark unused |

**Table 9 • CoreGPIO_OUT_0**

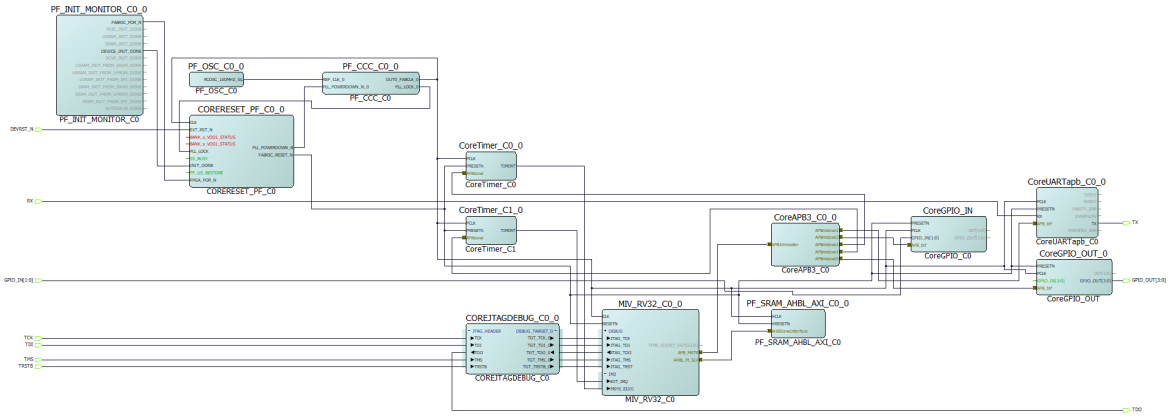
| Connect From | Connect To |
|---|---|
| COREGPIO_OUT_0:GPIO_IN[3:0] | Tie low |

| Connect From | Connect To |
|---|---|
| COREGPIO_OUT_0:INT[3:0] | Mark unused |
| COREGPIO_OUT_0:GPIO_OUT[3:0] | Promote to top level |

### Table 10 • CoreJTAGDebug_C0_0

| Connect From | Connect To |
|---|---|
| COREJTAGDEBUG_C0_0:TCK | Promote to top level |
| COREJTAGDEBUG_C0_0:TDI | Promote to top level |
| COREJTAGDEBUG_C0_0:TDO | Promote to top level |
| COREJTAGDEBUG_C0_0:TMS | Promote to top level |
| COREJTAGDEBUG_C0_0:TRSTB | Promote to top level |

### Figure 24 • Final Design



# 5 Design Constraints

Add the I/O constraints for the PolarFire Evaluation Kit to the design flow tab and select Manage Constraints. Under the I/O Attributes tabs, add the following constraints by selecting New and naming the file io_constraints. Then, copy the constraints below into this file. Save the constraints file and select the check box under Place and Route. Save the I/O 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
```

Add the Timing constraints to the design flow tab and select Manage Constraints. Under the Timing tab, add the following constraints by selecting New and naming the file io_jtag_constraints. Then, copy the constraints below 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 } ]
```

Save the constraints file, and associate it with Synthesis, Place and Route and Timing Verification by selecting each of the check boxes. It is also important to add in the derived constraints. This is done by selecting Derive Constraints in the Timing tab and excepting the default settings. Save the Timing Constraints.

# 6 Configure Design Initialization Data and Memories

The PolarFire LSRAM can be configured to boot a program from power-on reset. There are two ways to configure this:

- Configure the LSRAM memory via the LSRAM configurator.
- Configure the LSRAM memory via the Configure Design Initialisation Data and Memories configurator.

## 6.1 Configure LSRAM Memory via LSRAM Configurator

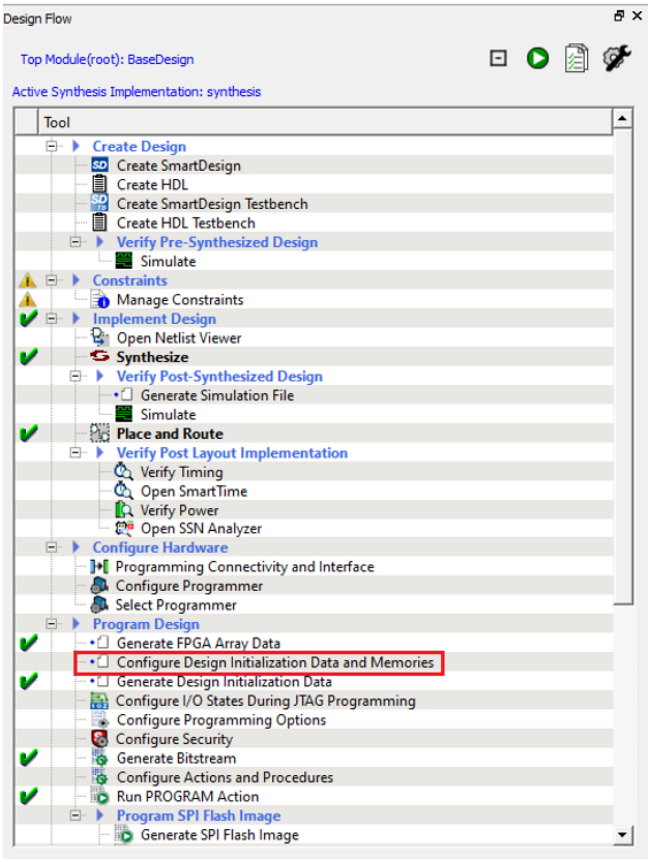
Open the SmartDesign, BaseDesign if closed. Locate the PF_SRAM_AHBL_AXI_C0_0.

1. Right-click on the core and select **Configure**.
2. Select the Memory Initialisation Settings tab.
3. Select the **Initialise RAM at Power up** check box.
4. Select the hex file created by SoftConsole when building a project (see section *Create a Deployable Hex File*).
5. In the Design Flow tab, select **Generate bitstream**. Once this has completed, the programming file is ready to download to the FPGA board.
6. Connect the power cable to your FPGA board.
7. Connect the USB cable to the Embedded FlashPro5 micro USB port and to the USB port of your computer.
8. In the Design Flow tab, select **Run Program Action** to download the programming file to the FPGA.

## 6.2 Configure LSRAM Memory via Configure Design Initialization Data and Memories Configurator

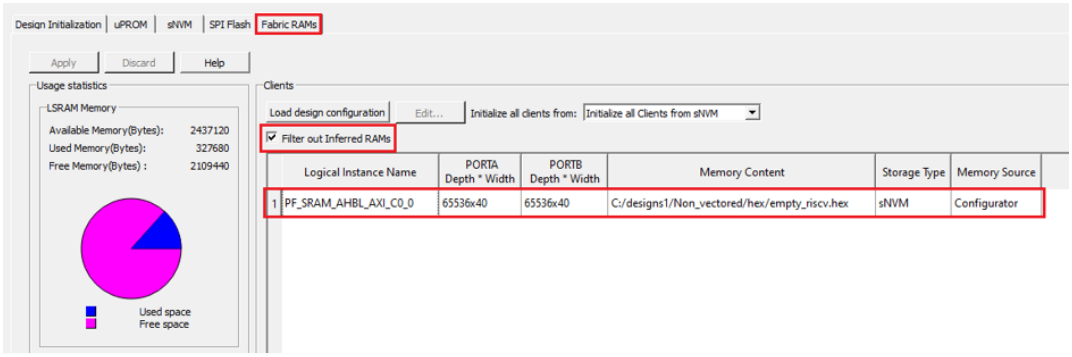
If you have run through the bitstream generation and you'd like to add or change the firmware in the LSRAM, this can be done via the Configure Design Initialization Data and Memories configurator.

**Figure 25 • Design Flow Progression**



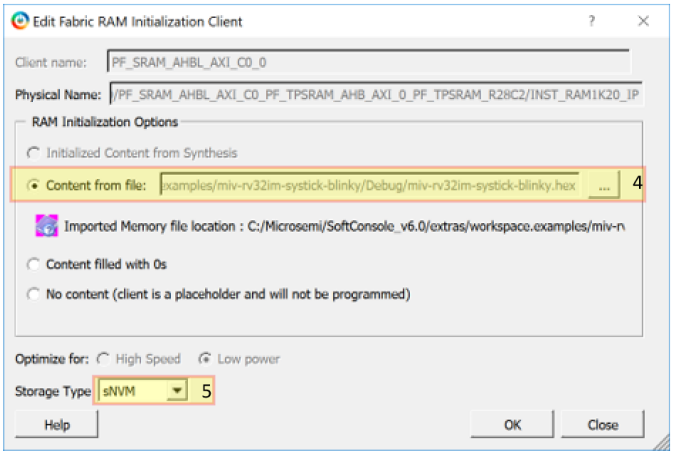

1. Run the flow so that the "Generate FPGA Array Data" step in the design flow is completed.
2. Select the Fabric RAMs tab.
3. Select Filter out Inferred RAM (Optional).
4. Double-click on the RAM to be initialized to open its configurator.

**Figure 26 • Design Initialization Data and Memories Configurator**




5. Under **Ram Initialization Options**, select **Content from file** and select the hex file created by SoftConsole when building a project (see section *Create a Deployable Hex File*).
6. Choose the appropriate storage location, for example **sNVM**.

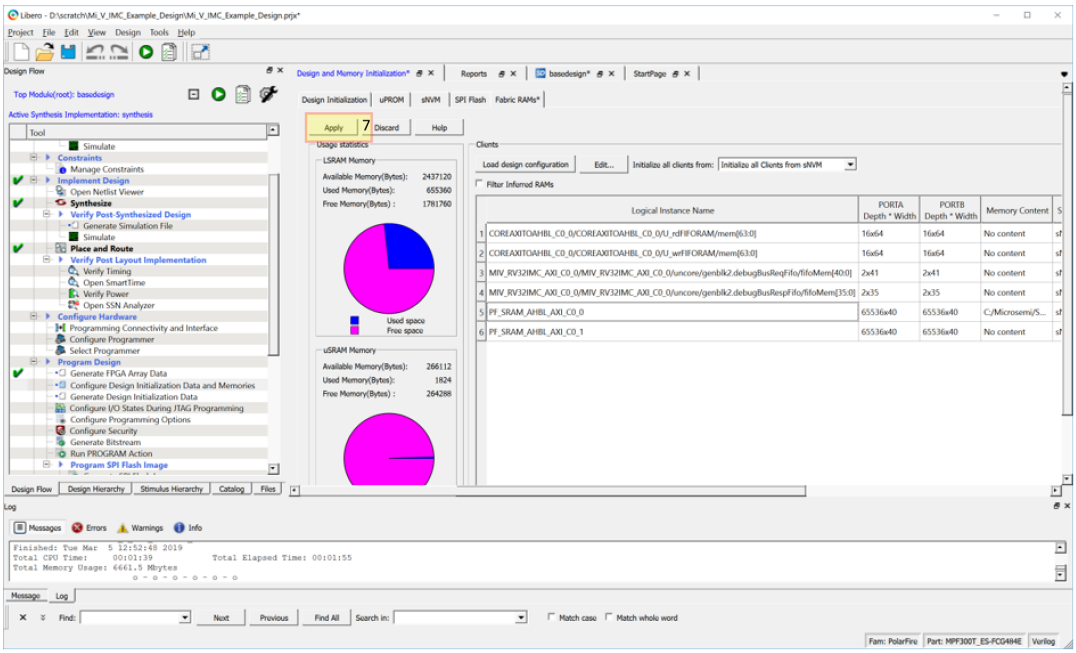
**Figure 27 • Fabric RAM Initialization Configurator**




7. Click **OK** and return to Libero.

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

**Figure 28 • Design Initialization Data and Memories Configurator**




9. In the Design Flow tab, select **Generate bitstream**. Once this has completed, the programming file is ready to download to the FPGA board.

10. Connect the power cable to your FPGA board.

11. Connect the USB cable to the Embedded FlashPro5 USB port and to the USB port of your computer.

12. In the Design Flow tab, select **Run Program Action** to download the programming file to the FPGA.

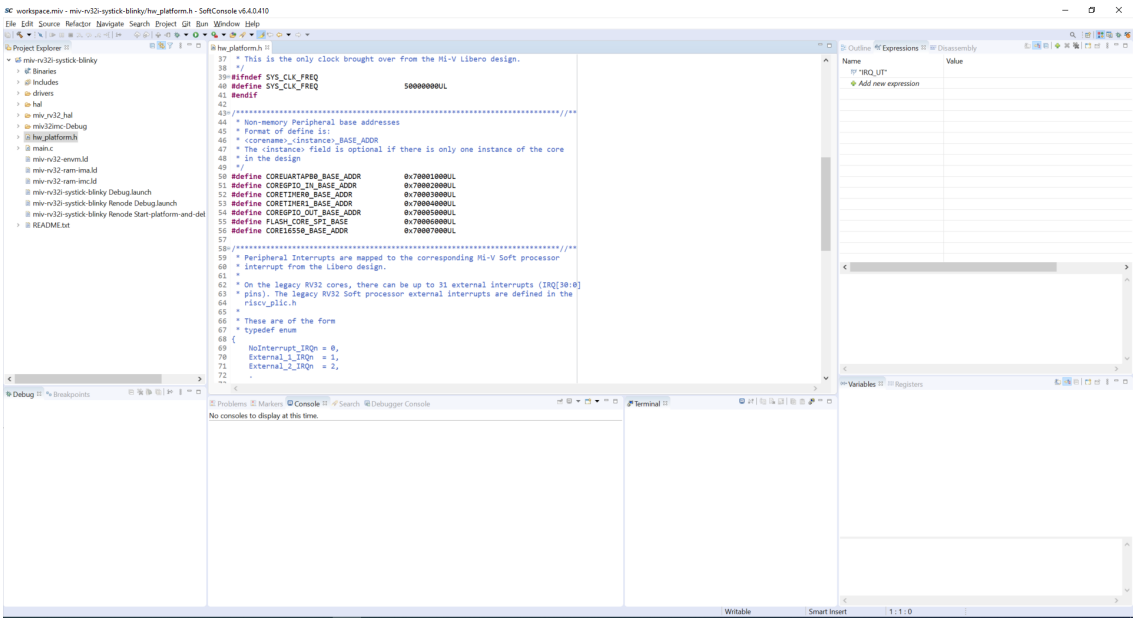
# 7 Run Project from SoftConsole

The following steps explain how to build and run a project in SoftConsole for the MIV_RV32 core. Open SoftConsole and the example project X.

## 7.1 Set System Clock Frequency and Peripheral Base Addresses

If UART is being used, the system clock frequency needs to be provided to the software and is done in the “hw_platform.h” file by changing the #define SYS_CLK_FREQ to the clock frequency (note this value must be in hertz).

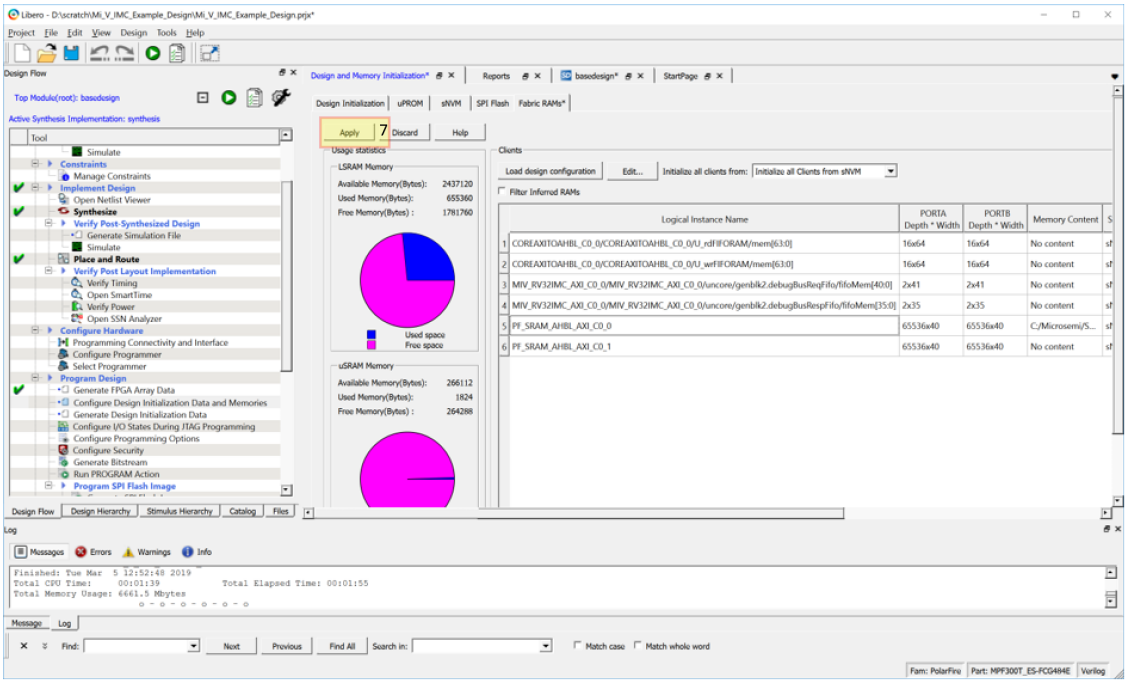
**Figure 29 • Set System Clock Frequency and Peripheral Base Addresses**




The “hw_platform.h” file is also used to set the base address for peripherals, the base address of a peripheral can be found in the project memory map generated by Libero.

The peripheral address in the hw_platform.h file must match the address in Libero for the peripheral to function correctly. These peripherals are addressed for 0x0 because the core redirects these addresses accordingly. Configuration settings for peripherals are displayed in the following figure.

**Figure 30 • MIV_RV32 Configurator Memory Map**



## 7.2 Build and Debug a Project

When you're ready to build your project, right-click on the project and select **Build Configurations** > **Set Active** > **5 miv32imc-Debug** as shown in the following figure. Then, click the **Build** icon. In this case, the project is being built for debug.

**Figure 31 • Build Configurations**

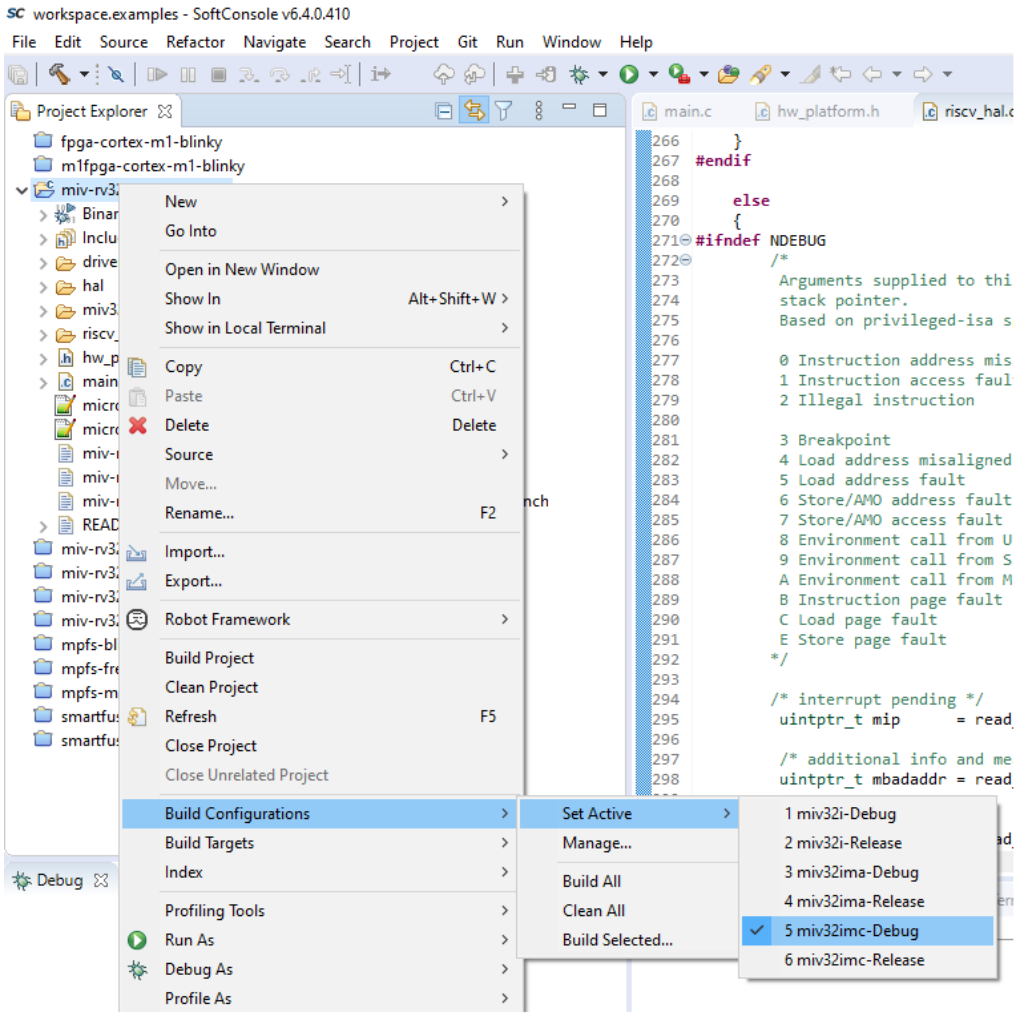
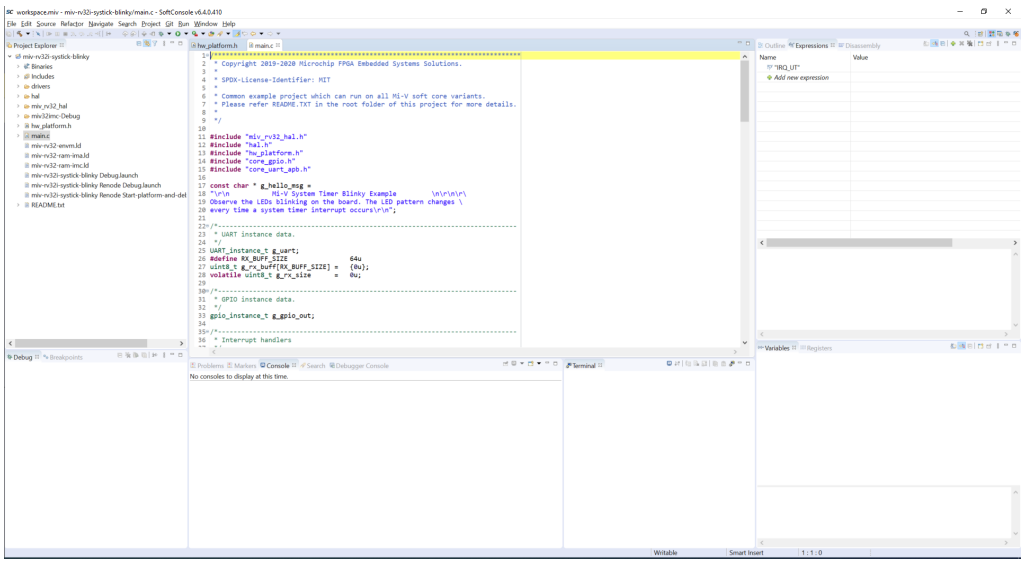
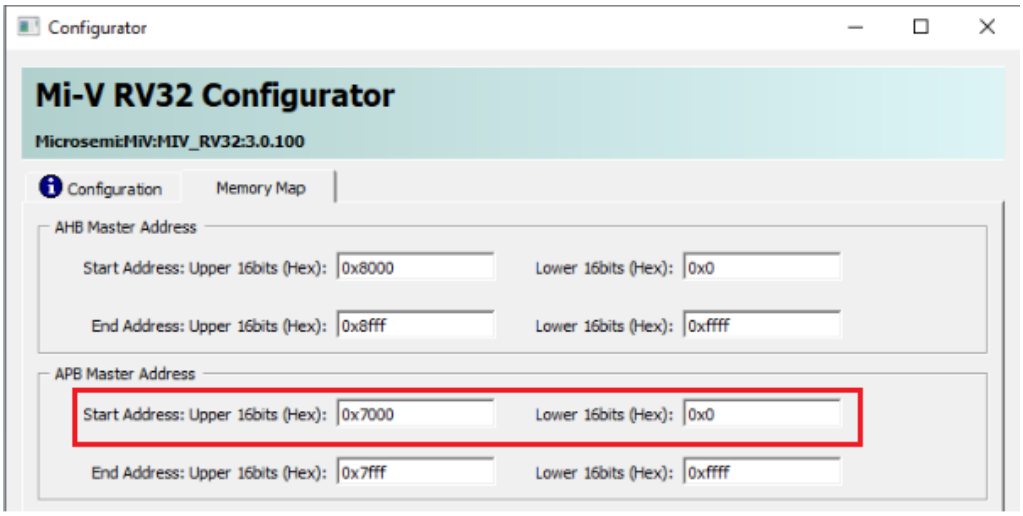



**Figure 32 • Build and Debug a Project**



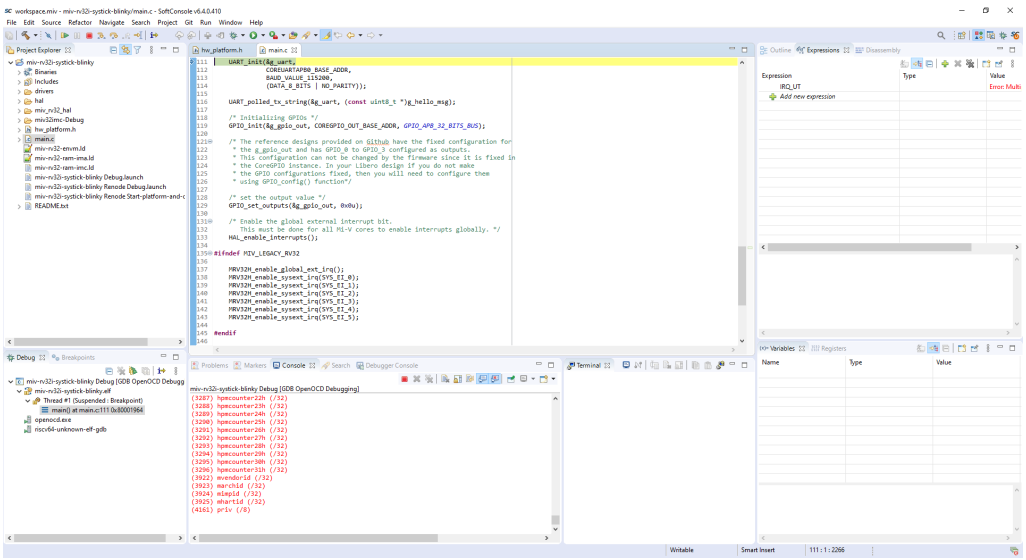
Once you have an error-free build, you can test it on your device. Make sure your device is connected and powered on, then click the **Debug** icon. If this is your first time debugging this project, you will have to select a debug configuration.

**Figure 33 • Select a Debug Configuration**




Once you have selected a debug configuration, click **Debug**. This will download the program to your device and take you to the debug perspective. The program will launch and halt on main. It can be run by clicking **Resume** or pressing **F8**.

**Figure 34 • Run a Program**



To add breakpoints and step through the code, ensure you are in the main.c tab.

## 7.3 Create a Deployable Hex File

The following steps create a hex file in SoftConsole:

1. Open the Project Properties, **right-click** > **Project Properties**.
2. Select **the correct linker script** > **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, 0x60000000).

4. Select **Apply** and **Close**.

A build configuration can be created, which will allow you to create deployable builds without needing to repeat this step. For more information, see the SoftConsole Release Notes section “Projects”.

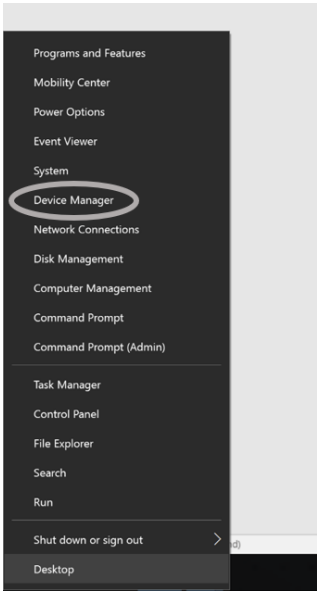
## 7.4 Communicate via UART

The following are needed to communicate via UART:

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

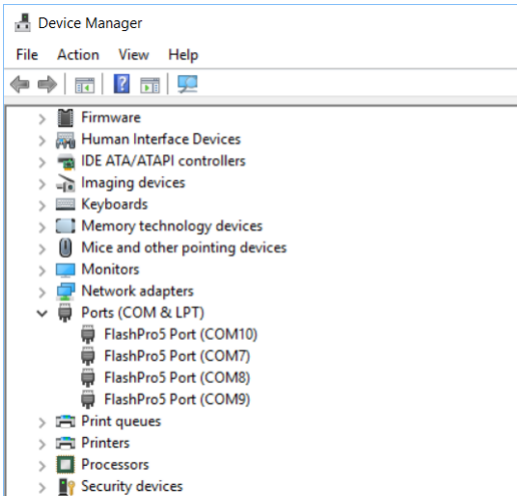
Device Manager can be used to find the COM port of your device. To open it, right-click on **Start** and select **Device Manager**.

**Figure 35 • Open Device Manager**



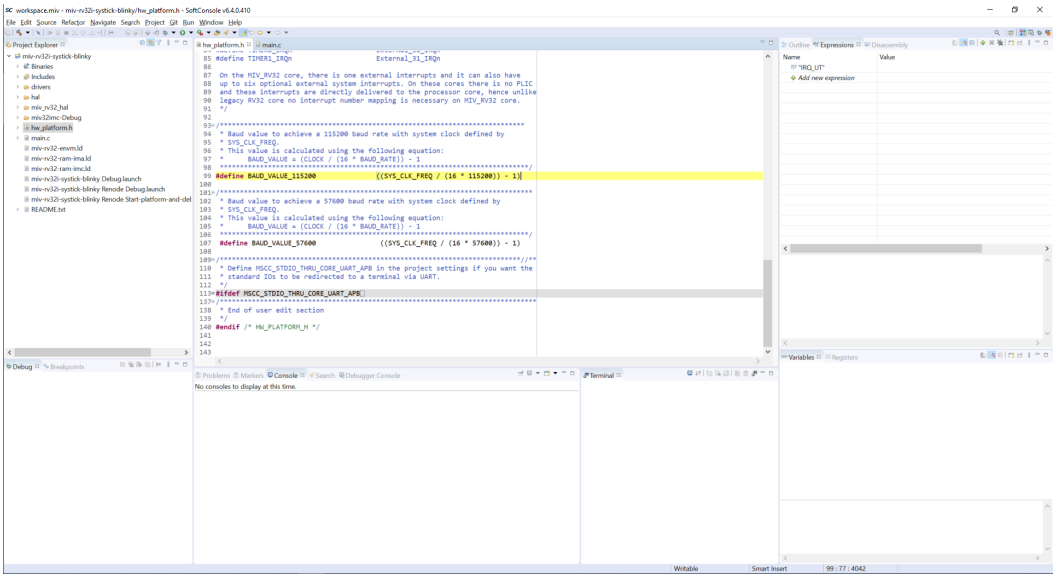

The COM port for your device will be one of those listed under **Ports (COM & LPT)**. It may not be clear which one is your device and may require trial and error to find the correct port.

**Figure 36 • Find COM Ports in Device Manager**



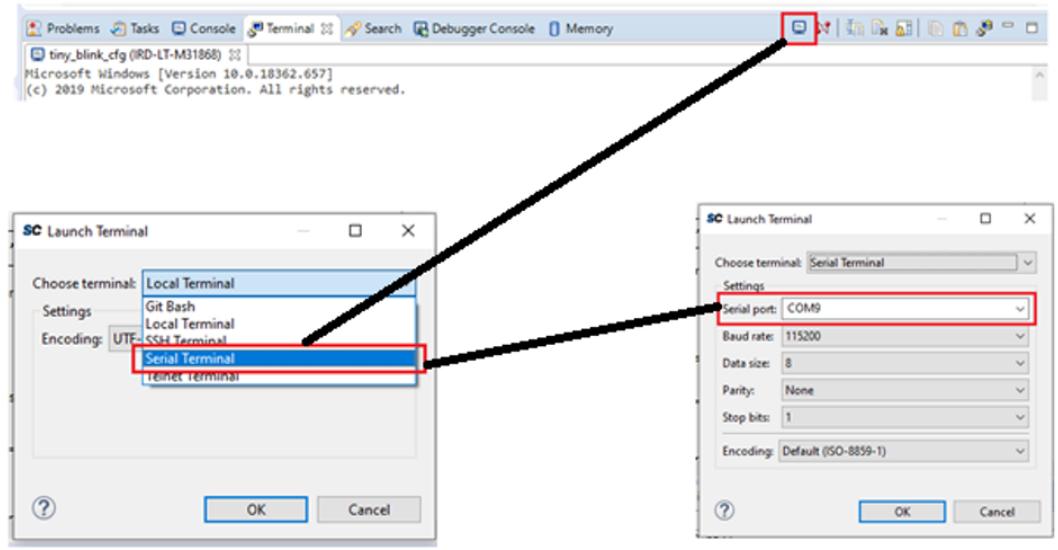
The baud rate for the UART connection can be found in the hw_platform.h file within the SoftConsole project.

**Figure 37 • Location of Baud Rate Value**



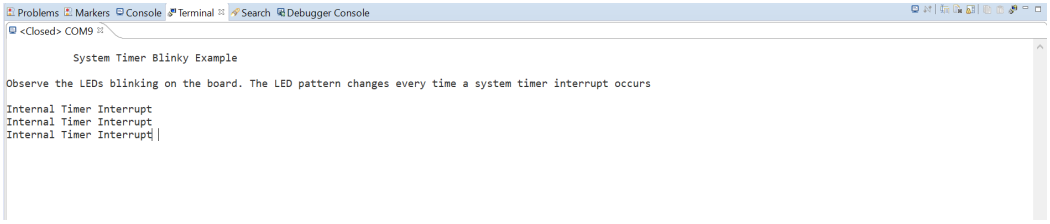
Open the serial communication client within SoftConsole by selecting **Window** > **Show View** > **Terminal**. Then, input the values for your COM port and baud rate.

**Figure 38 • Serial Output from COM9 Terminal**



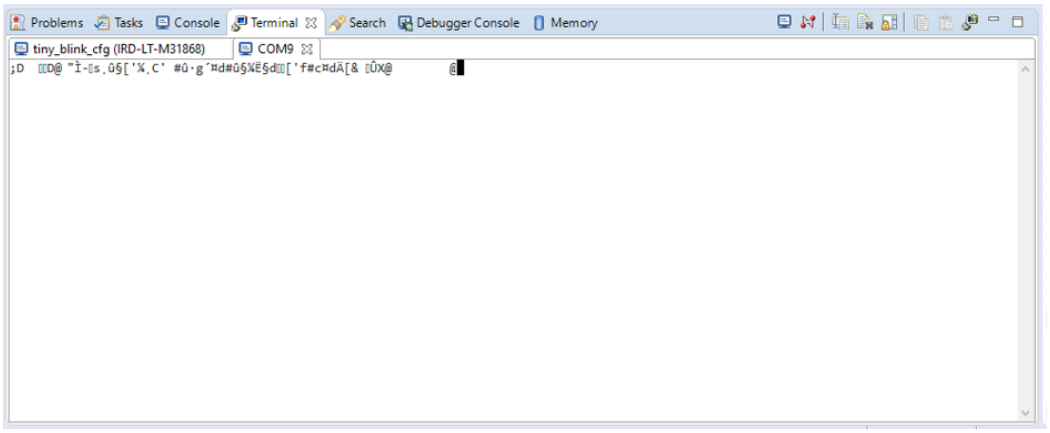

Run your program and any UART output will be displayed in the communication client's terminal.

**Figure 39 • Output from SoftConsole Terminal**




If the UART output isn't as expected (that is, the random/unknown characters are as shown in the following figure) it is more than likely an issue with the clock frequency in the hw_platform.h file and can be fixed by following the steps in this section.

**Figure 40 • UART Clock Frequency Issues**



# 8 Simulation of the Design

This section describes how to simulate the design. The PF_LSRAM must be initialized with a program (.hex) for the MIV_RV32 core to execute.

**Note:** If the PF_LSRAM has not been initialized with a program, complete steps 1–4 from the previous section *Configure LSRAM Memory via LSRAM Configurator*.

The following steps create a testbench with the BaseDesign instantiated in it.

1. Right-click on the BaseDesign SmartDesign, then select **Create Testbench** > **HDL**.
2. Name the testbench **BaseDesign_TB** and select the language **Verilog**.

The following steps run the simulation:

1. Select the **Simulation** tab and right-click on the **BaseDesign_TB**.
2. Select **Simulate Pre-Synthesis Design** > **Open Interactively** to open ModelSim.
3. Once ModelSim has opened, expand **BaseDesign_TB** and add **MIV_RV32_C0_0** to the waveform.

   **Note:** To add signals to the waveform, right-click and select **Add wave**. This will add all of the top level signals of MIV_RV32_C0_0 to the waveform viewer.
4. To run the simulation, go to the transcript window and type **run 100us**. This will run the simulation for 100 µs.

**Microsemi**
2355 W. Chandler Blvd.
Chandler, AZ 85224 USA

Within the USA: +1 (480) 792-7200
Fax: +1 (480) 792-7277





Microsemi Corporation, a subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), and its corporate affiliates are leading providers of smart, connected and secure embedded control solutions. Their easy-to-use development tools and comprehensive product portfolio enable customers to create optimal designs which reduce risk while lowering total system cost and time to market. These solutions serve more than 120,000 customers across the industrial, automotive, consumer, aerospace and defense, communications and computing markets. Headquartered in Chandler, Arizona, the company offers outstanding technical support along with dependable delivery and quality. Learn more at **www.microsemi.com**.

50200909