DG0834
Demo Guide
Running Webserver and IAP Using TFTP on PolarFire Device

Microsemi
a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

50200834. 3.0 11/19

Contents

1 Revision History . . . 1
1.1 Revision 3.0 . . . 1
1.2 Revision 2.0 . . . 1
1.3 Revision 1.0 . . . 1

2 Running Webserver and IAP Using TFTP on PolarFire Device . . . 2
2.1 Webserver and TFTP Server Demo Design Layers . . . 3
2.1.1 Application Layer . . . 3
2.1.2 Transport Layer (lwIP TCP/IP Stack) . . . 3
2.1.3 RTOS and Firmware Layer . . . 3
2.2 Design Requirements . . . 4
2.3 Prerequisites . . . 4
2.4 Demo Design . . . 5
2.4.1 Design Implementation . . . 6
2.5 Clocking Structure . . . 19
2.6 SoftConsole Firmware Project . . . 20

3 Libero Design Flow . . . 21
3.1 Synthesize . . . 22
3.2 Place and Route . . . 22
3.2.1 Resource Utilization . . . 22
3.3 Verify Timing . . . 22
3.4 Generate FPGA Array Data . . . 23
3.5 Configure Design Initialization Data and Memories . . . 23
3.6 Generate Bitstream . . . 24
3.7 Export FlashPro Express Job . . . 24
3.8 Run PROGRAM Action . . . 25
3.9 Program SPI Flash Image . . . 26

4 Running the Demo . . . 27
4.1 Tera Term Setup . . . 27
4.2 Running Webserver Demo . . . 28
4.3 Running TFTP Demo . . . 29
4.3.1 Running IAP Authentication . . . 32
4.3.2 Running IAP Program . . . 33

5 Appendix 1: Enable TFTP Client . . . 35

6 Appendix 2: Running the SoftConsole Project in Debug Mode from LSRAM or DDR Memory . . . 38
6.1 Running the Design in Static IP Mode . . . 40

7 Appendix 3: Programming the Device and External SPI Flash Using FlashPro Express . . . 42

Figures

Figure 1 Webserver and TFTP Server Applications on a PolarFire Device . . . 3
Figure 2 Demo Design High-level Block Diagram . . . 5
Figure 3 Top-Level Libero Implementation . . . 6
Figure 4 Mi-V Configurator . . . 8
Figure 5 PF_SRAM_0 . . . 8
Figure 6 PF_LSRAM_1_0 . . . 9
Figure 7 CoreAXI4Interconnect Configurator . . . 9
Figure 8 CoreAXI4Interconnect Configurator—Master Configuration . . . 9
Figure 9 CoreAXI4Interconnect Configurator—Slave Configuration . . . 10
Figure 10 CoreAXI4Interconnect Configurator—Crossbar Configuration . . . 10
Figure 11 DDR3 Configuration . . . 11
Figure 12 PF_CCC_0 Input Clock Configuration . . . 12
Figure 13 PF_CCC_0 Output Clock Configuration . . . 12
Figure 14 NWC_PLL_0 Input Clock Configuration . . . 13
Figure 15 NWC_PLL_0 Output Clock Configuration . . . 14
Figure 16 NWC_PLL_0 DLL Configuration . . . 15
Figure 17 CoreSPI_0 Configuration . . . 15
Figure 18 Mi-V Processor Bus Interface Memory Map . . . 16
Figure 19 CoreAHBLite_2 Configuration . . . 17
Figure 20 CoreAPB3 Configuration . . . 18
Figure 21 Clocking Structure . . . 19
Figure 22 Directory Structure of Webserver SoftConsole Project . . . 20
Figure 23 Directory Structure of TFTP IAP SoftConsole Project . . . 20
Figure 24 Libero Design Flow Options . . . 21
Figure 25 Configure Design Initialization Data and Memories Option . . . 23
Figure 26 Fabric RAMs Tab . . . 23
Figure 27 Start Address for SPI Flash Clients . . . 24
Figure 28 Export FlashPro Express Job . . . 25
Figure 29 Run Program Action . . . 26
Figure 30 SPI Flash Programming . . . 26
Figure 31 Select Serial as the Connection Type . . . 27
Figure 32 Tera Term Configuration . . . 28
Figure 33 Tera Term General Setup . . . 28
Figure 34 Tera Term with IP Address . . . 28
Figure 35 Webserver Demo Page . . . 29
Figure 36 Tera Term Window . . . 29
Figure 37 Erasing the SPI Flash Memory Location [0xA00000 - 0x13FFFFF] . . . 30
Figure 38 Acquiring IP Address . . . 30
Figure 39 Transfer Programming Image1 . . . 30
Figure 40 Bytes Received for Image1 . . . 31
Figure 41 Erasing the SPI Flash Memory Location [0x1400000 - 0x1DFFFFF] . . . 31
Figure 42 Transfer IAP Image2 . . . 31
Figure 43 Bytes Received for Image2 . . . 32
Figure 44 SPI Directory . . . 32
Figure 45 Successful IAP Image1 Authentication . . . 32
Figure 46 Successful IAP Image2 Authentication . . . 33
Figure 47 Successful IAP with Image1 . . . 33
Figure 48 Successful IAP by Image2 . . . 34
Figure 49 Control Panel—Programs and Features . . . 35
Figure 50 Selecting TFTP Client from Windows Features . . . 35
Figure 51 System and Security Window . . . 36
Figure 52 Allow Programs Window . . . 36
Figure 53 Add an app Window . . . 37
Figure 54 Selecting Trivial File Transfer Protocol App in Allowed apps Window . . . 37
Figure 55 SoftConsole Debug Configuration . . . 38

Figure 56 Project Explorer Window 39
Figure 57 Project Properties 39
Figure 58 Properties for gbe_webserver 40
Figure 59 Host PC TCP/IP Settings 41
Figure 60 FlashPro Express Job Project 42
Figure 61 New Job Project from FlashPro Express Job 43
Figure 62 Programming the Device 43
Figure 63 FlashPro Express—RUN PASSED 44

Tables

Table 1 Design Requirements 4
Table 2 I/O Signals 7
Table 3 Resource Utilization 22
Table 4 Jumper Settings 25

1 Revision History

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

1.1 Revision 3.0

The following is a summary of the changes made in this revision.

- Updated the document for Libero SoC v12.2.
- Removed the references to Libero version numbers.

1.2 Revision 2.0

The document was updated for Libero SoC v12.0 release.

1.3 Revision 1.0

The first publication of this document.

2 Running Webserver and IAP Using TFTP on PolarFire Device

Microsemi PolarFire® FPGAs support 1G Ethernet solutions for various networking applications. In PolarFire devices, 10/100/1000 Mbps (1G) Ethernet is implemented using the CoreTSE_AHB Media Access Control (MAC) soft IP core. The CoreTSE_AHB IP implements a Serial Gigabit Media-Independent Interface (SGMII or GMII) with an Ethernet PHY. This Ethernet interface can be implemented in the FPGA by using either a transceiver (PF_XCVR IP) or a GPIO with clock and data recovery (PF_IOD_CDR IP) capability. In this demo, the 1G Ethernet solution is implemented in the FPGA design by using GPIOs with CDR capability and CoreTSE_AHB IP.

The CoreTSE_AHB IP core enables system designers to implement a broad range of Ethernet designs, from low cost 10/100 Ethernet to higher performance 1 gigabit ports. The CoreTSE_AHB IP core is suitable for use in networking equipment such as switches, routers, and data acquisition systems.

The CoreTSE_AHB IP has the following major interfaces:

- 10/100/1000 Mbps Ethernet MAC with a Gigabit Media Independent Interface (GMII) and Ten Bit Interface (TBI) to support Serial Gigabit Media Independent Interface (SGMII), 1000BASE-T, and 1000BASE-X.
- GMII or TBI physical layer interface connects to Ethernet PHY
- MAC data path interface

The CoreTSE_AHB IP core is available in two different versions:

- CoreTSE_AHB: Uses AHB interface for both the transmit and receive paths.
- CoreTSE_AHB (Non-AMBA): Uses direct access to the MAC with a streaming packet interface.

For more information about CoreTSE_AHB IP, see the *CoreTSE_AHB Handbook*.

CoreTSE_AHB IP core requires license for using in Libero® SoC design. For license request, contact *soc_marketing@microsemi.com*.

This demo design implements a Webserver application and a Trivial File Transfer Protocol (TFTP) server using the PolarFire Evaluation Kit board. For more information about this board, see *UG0747: PolarFire FPGA Evaluation Kit User Guide*.

This demo design demonstrates the following:

- Use of Ethernet MAC connected to a serial gigabit media independent interface (SGMII) PHY.
- Integration of CoreTSE_AHB MAC driver with lwIP TCP/IP stack and FreeRTOS operating system.
- Implementation of Webserver on the PolarFire Evaluation board.
- Implementation of TFTP server on the PolarFire Evaluation board.
- Procedure to run Webserver and TFTP server designs on the PolarFire Evaluation board.

This demo design can be programmed using either of the following options:

- Using the pre-generated Job file: To program the device using the job file provided along with the demo design files, see Appendix 3: Programming the Device and External SPI Flash Using FlashPro Express, page 42.
- Using Libero SoC: To program the device using Libero SoC, see Libero Design Flow, page 21.

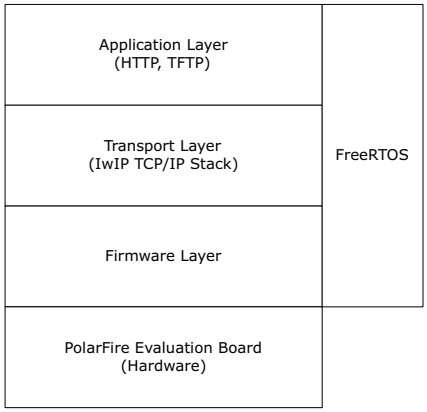
2.1 Webserver and TFTP Server Demo Design Layers

The Webserver and TFTP server demo design have the following layers.

- Application layer
- Transport layer (lwIP TCP/IP stack)
- RTOS and firmware layer

The following figure is a block diagram of the three layers in the Webserver and TFTP server applications on a PolarFire device.

Figure 1 • **Webserver and TFTP Server Applications on a PolarFire Device**



2.1.1 Application Layer

The Webserver handles the HTTP request from the client (host PC) browser and transfers the static pages to the client in response to its request. When the IP address (for example, http://10.60.3.25) is typed in the address bar of the browser, an HTTP request is sent to the port associated with the Webserver. The Webserver then interprets the request and responds to the client with the requested page or resource.

The TFTP client (the host PC) transfer files to the PolarFire device (the TFTP server) using the `TFTP PUT` command. Transferred files are stored in the PolarFire Evaluation board external flash memory, which is connected to the System Controller SPI interface.

2.1.2 Transport Layer (lwIP TCP/IP Stack)

The lwIP TCP/IP stack, developed by Adam Dunkels at the Swedish Institute of Computer Science (SICS), is suitable for embedded systems because of its low system resource usage. The lwIP stack can be used with or without an operating system. It consists of actual implementations of IP, ICMP, UDP, and TCP protocols, as well as the support functions such as buffer and memory management.

lwIP is available (under a BSD license) in C source-code format for download at *http://download.savannah.gnu.org/releases/lwip/*.

2.1.3 RTOS and Firmware Layer

FreeRTOS is an open-source, real-time operating system kernel. In this demo, FreeRTOS is used to prioritize and schedule tasks. For more information about FreeRTOS and the latest source code, see *http://www.freertos.org*.

The firmware provides software drivers to configure and control the following components.

- Ethernet MAC
- Core UART APB
- SPI

2.2 Design Requirements

The following table lists the resources required to run the demo.

Table 1 • **Design Requirements**

Requirement	Version
Operating System	Windows 7, 8.1, or 10
Hardware	
PolarFire Evaluation Kit (MPF300-EVAL-KIT)	Rev D or later
Ethernet cable	RJ45
Software	
FlashPro Express	**Note:** Refer to the readme.txt file provided in the design files for the software versions used with this reference design.
Libero SoC	
SoftConsole	
A serial terminal emulation program	HyperTerminal, TeraTerm, or PuTTY
Browser	Mozilla Firefox, Internet Explorer

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.3 Prerequisites

Before you begin:

1. For demo design files download link:
   http://soc.microsemi.com/download/rsc/?f=mpf_dg0834_df
2. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location:
   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc#downloads

   The latest versions of ModelSim and Synplify Pro are included in the Libero SoC installation package.

2.4 Demo Design

The following is the data flow of the demo design:

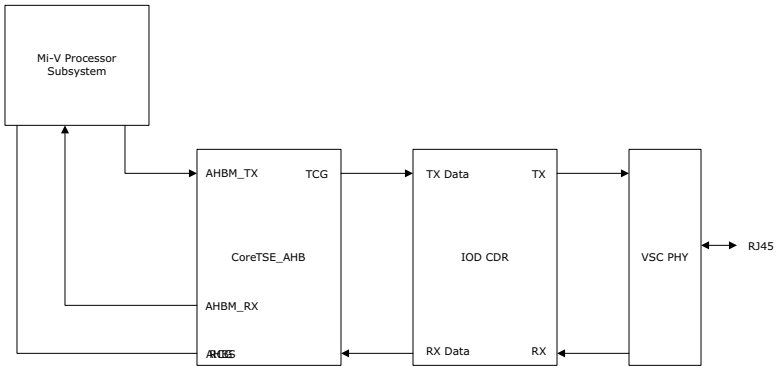
1. PF_CCC_0 provides the clock to the Mi-V processor and other APB peripherals.
2. NWC_PLL_0 drives the IOD CDR clocks SGMII_CDR_0: TX_CLK_G and HS_IO_CLK.
3. Mi-V performs the following functions:
   - Executes the application from LSRAM (PF_SRAM IP)
   - Configures the ZL30364 clock generation hardware through the CoreSPI IP to generate reference clocks for the VSC PHY and the IOD CDR fabric module.
   - Configures the CoreTSE_AHB IP MAC in TBI mode and initializes the MAC in 1000 Base-T.
   - Sends a request to the CoreTSE_AHB IP to negotiate with the on-board VSC8575 PHY.
4. CoreTSE_AHB IP implements the 1G Ethernet MAC and is configured to interface with the PF_IOD_CDR block in the SGMII mode. The CoreTSE IP has an inbuilt MDIO interface to exchange control and status information with the VSC PHY.
5. PF_IOD_CDR IP does the following:
   - Interfaces with the on-board VSC8575 PHY.
   - Recovers the data and clock from the incoming RX_P and RX_N ports. Deserializes the recovered data and sends 10-bit parallel data to the CoreTSE.
   - Receives Ethernet data from VSC PHY through the RX_P and RX_N input pads, gears down the receive data rate, and deserializes the data.
6. The deserialized data is sent from SGMII_CDR_0:RX_DATA[9:0] to CoreTSE_AHB IP: RCG[9:0]. The CoreTSE_AHB IP MAC receives the Ethernet packet from the on-board Ethernet PHY through high-speed PF_IOD_CDR IP using the built-in DMA controller and the Mi-V processes the Ethernet packets.
7. The Ethernet packets from the Mi-V processor are sent to CoreTSE_AHB IP, and CoreTSE_AHB IP:TCG[9:0] is sent to SGMII_CDR_0:TX_DATA[9:0].
8. SGMII_CDR_0 serializes the data, gears up the transmit data rate, and transmits the data to the on-board VSC PHY through the TX_P and TX_N output pads.

Following are the demo design features:

- Webserver
- IAP using TFTP server

The following figure shows the high-level demo design block diagram. In this demo design, CoreTSE_AHB IP is instantiated in the FPGA fabric and connected to the on-board VSC PHY using the IOD CDR IP.

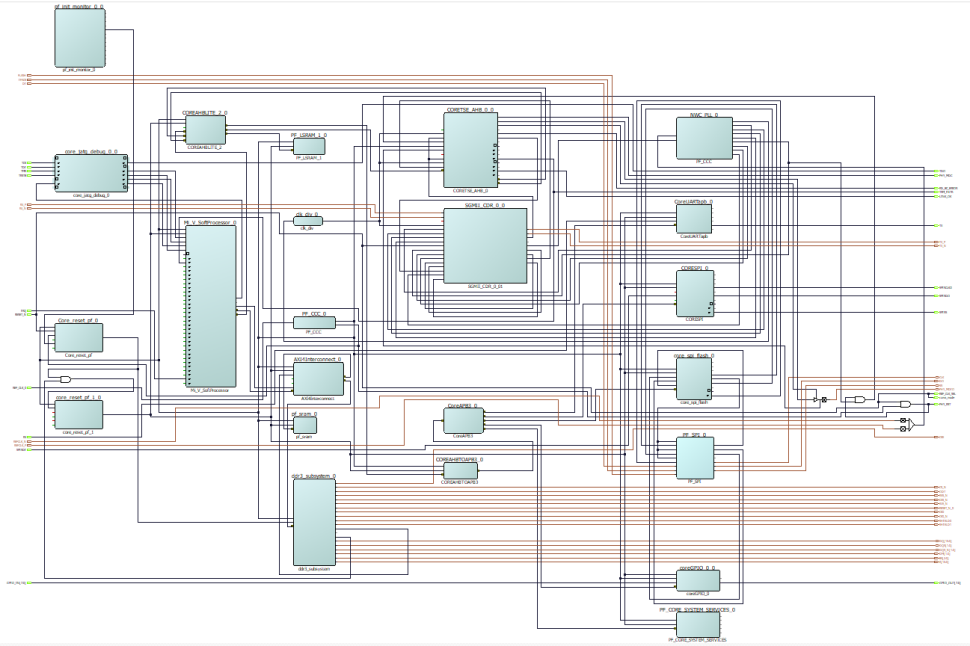
Figure 2 • **Demo Design High-level Block Diagram**



2.4.1 Design Implementation

The following figure shows the top-level Libero implementation of the demo design. The libero project implementation is the same for both Webserver and IAP using TFTP but the application firmware is different.

Figure 3 • **Top-Level Libero Implementation**



The following table lists the important I/O signals of the design.

Table 2 • **I/O Signals**

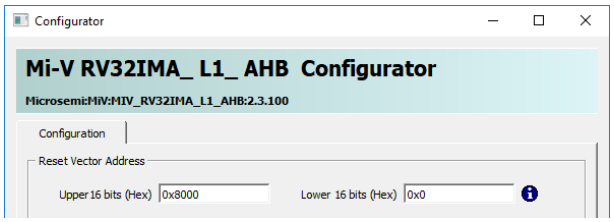
Signal	Direction	Description
RX_P, RX_N	Input	IOD CDR receive signals connected to the VSC PHY transmit data signals
REFCLK_N, REFCLK_P	Input	125 MHz input clock received from the on-board ZL30364 and fed to NWC_PLL_0.
RESET_N	Input	Mi-V reset. Asserted by pressing the on-board K22 push-button
REF_CLK_0	Input	50 MHz input clock received from the on-board 50 MHz oscillator and fed to PF_CCC_0.
TCK, TDI, TMS, and TRSTB	Input	JTAG signals interfaced to the soft processor for debugging
TDO	Output	
TX_P, TX_N	Output	IOD CDR transmit signals connected to the VSC PHY receive data signals.
LINK_OK	Output	Link status indicator. Provides the link up or down status with the on-board PHY. This signal is mapped to on-board LED7. The LED ON condition indicates that the link is up.
PHY_RST	Output	Reset signal to the on-board VSC8575 PHY
PHY_MDC	Output	MDIO clock fed to the on-board VSC8575 PHY
PHY_MDIO	Output	Management Data IO interface for accessing the on-board VSC8575 PHY registers
coma_mode	Output	Signal held low (connected to ground) to keep the VSC PHY fully active when it is out of reset.
REF_CLK_SEL	Output	Reference clock speed pin of the VSC PHY. Held high for selecting the 125 MHz reference clock speed
RD_BC_ERROR	Output	CoreTSE receive error signal. Indicates the receive code group error. This signal is synchronous to RX_CLK_R and mapped to on-board LED4. The LED ON condition indicates an error in the received code group.
SPISCLKO, SPISS, and SPISDO	Output	SPI controller signals to interface with the ZL30364 clock generation hardware.
SPISDI	Input	

2.4.1.1 Mi-V Soft Processor

The Mi-V soft processor supports RISC-V processor-based designs. The Mi-V soft processor executes the application from the LSRAM mapped at 0x80000000. It configures the ZL30364 clock generation hardware through the CoreSPI IP and the VSC PHY through the CoreTSE_AHB MDIO interface. It also configures the CoreTSE_AHB registers using the AHB interface.

The following figure shows the Mi-V soft processor configuration, where the **Reset Vector Address** is set to 0x8000_000. This is because in the Mi-V processor memory map, the memory range used for the AHB memory interface is 0x8000_0000 to 0x8FFF_FFFC, and the memory range used for the AHB I/O interface is 0x6000_0000 to 0x7FFF_FFFF.

Figure 4 • **Mi-V Configurator**



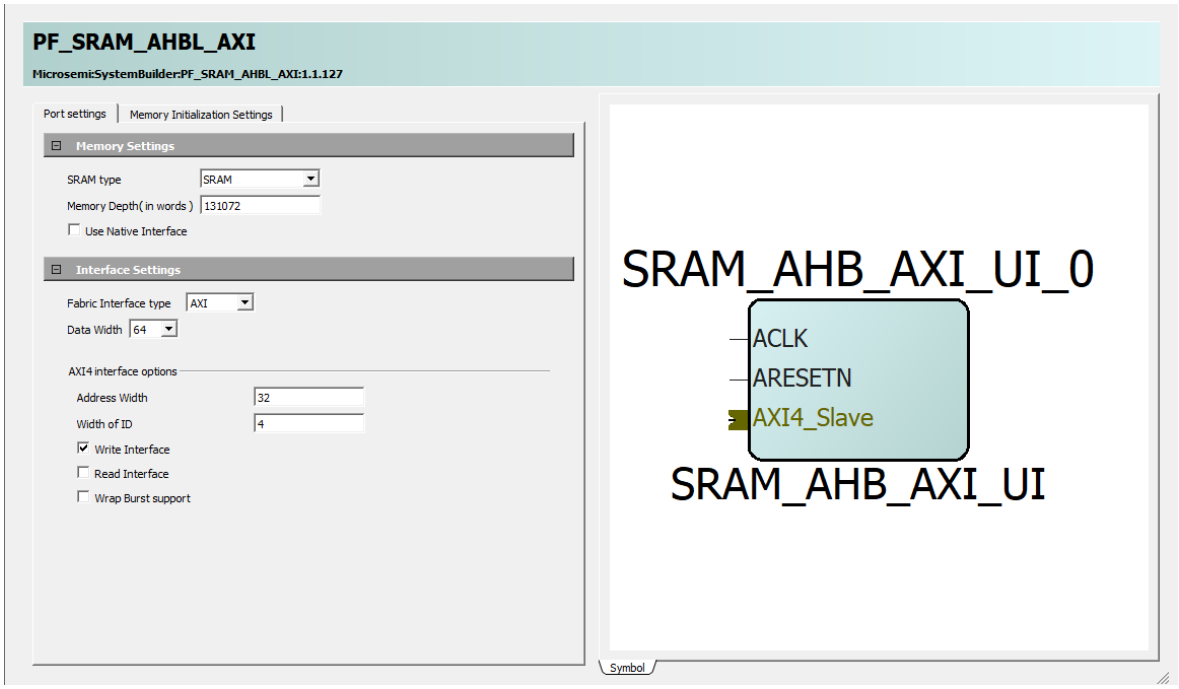
2.4.1.2 PF_SRAM_AHBL_AXI

This design uses two instances of PF_SRAM_AHBL_AXI core—pf_sram_0 and PF_LSRAM_1_0.

The pf_sram_0 IP is connected to Mi-V as an AHB slave using Core AXI4Interconnect. The LSRAM blocks are initialized with the user application code from the external SPI flash.

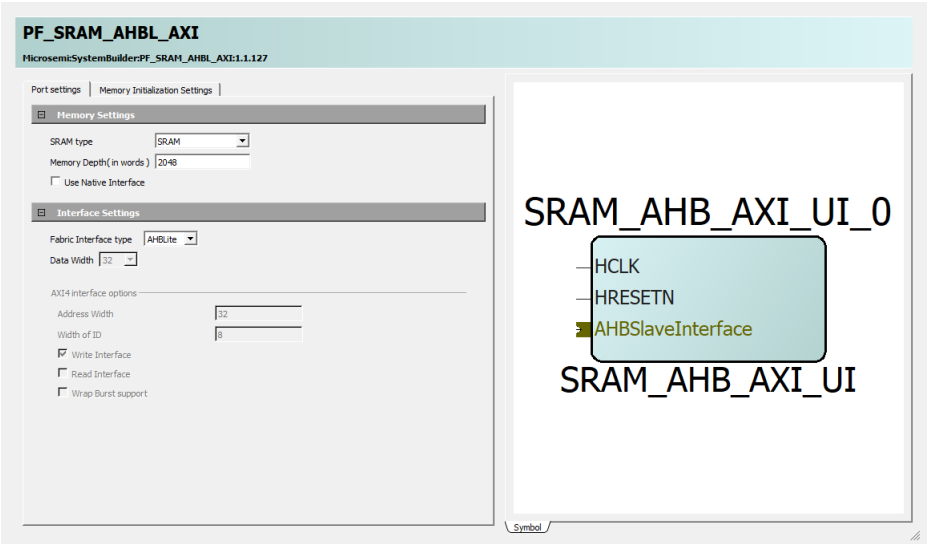
The processor uses the SRAM memory to execute the application. The following figure shows the LSRAM depth and the interface settings. The **Fabric Interface type** is selected AXI because the fabric interfaces with the Mi-V processor using Core AXI4Interconnect. The memory depth can be selected based on the application size. This design uses 512 KB RAM (131072 words).

Figure 5 • **PF_SRAM_0**



The PF_LSRAM_1_0 is connected to the Mi-V MMIO interface using CoreAHBLite. This memory is used for Ethernet MAC transmit and receive buffers.

Figure 6 • **PF_LSRAM_1_0**



2.4.1.3 CoreAXI4Interconnect

The AXI interconnect bus must be configured to connect the Mi-V core with memory. The following figure shows the bus configuration and other configuration of CoreAXI4Interconnect.

Figure 7 • **CoreAXI4Interconnect Configurator**

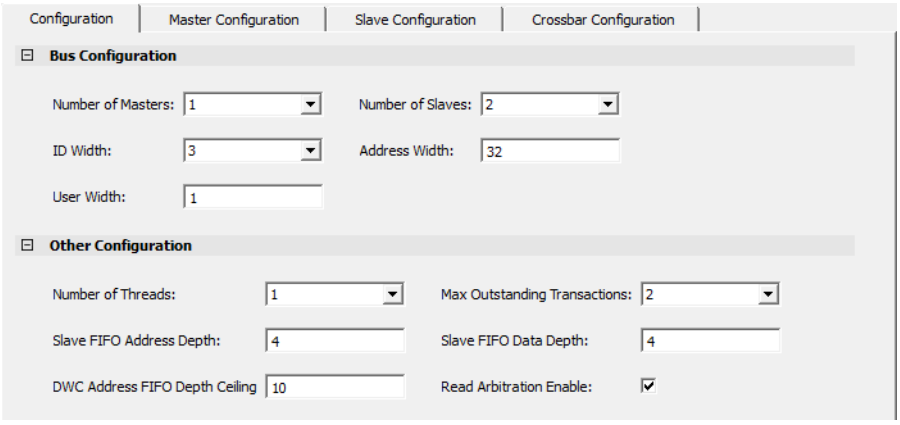


Figure 8 • **CoreAXI4Interconnect Configurator—Master Configuration**

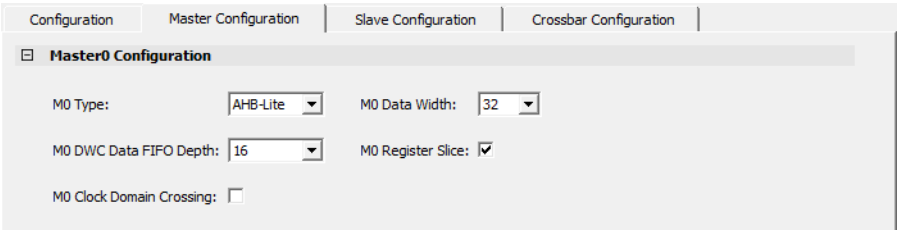


Figure 9 • **CoreAXI4Interconnect Configurator—Slave Configuration**

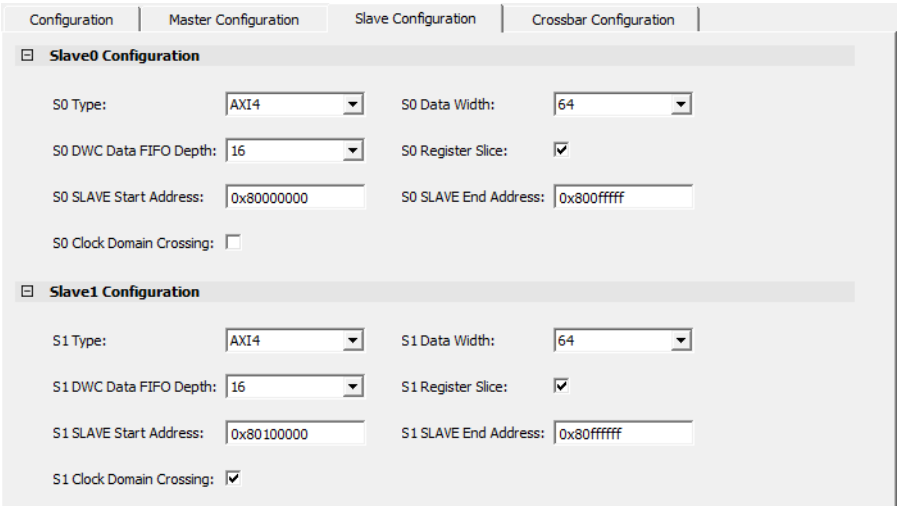
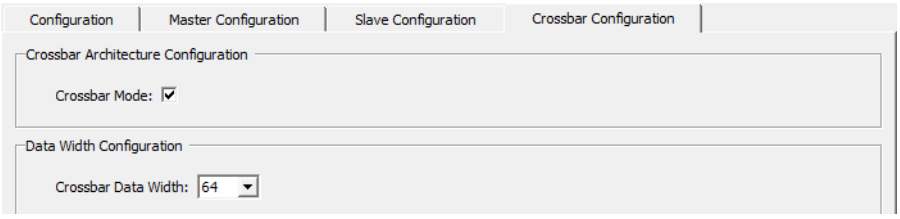


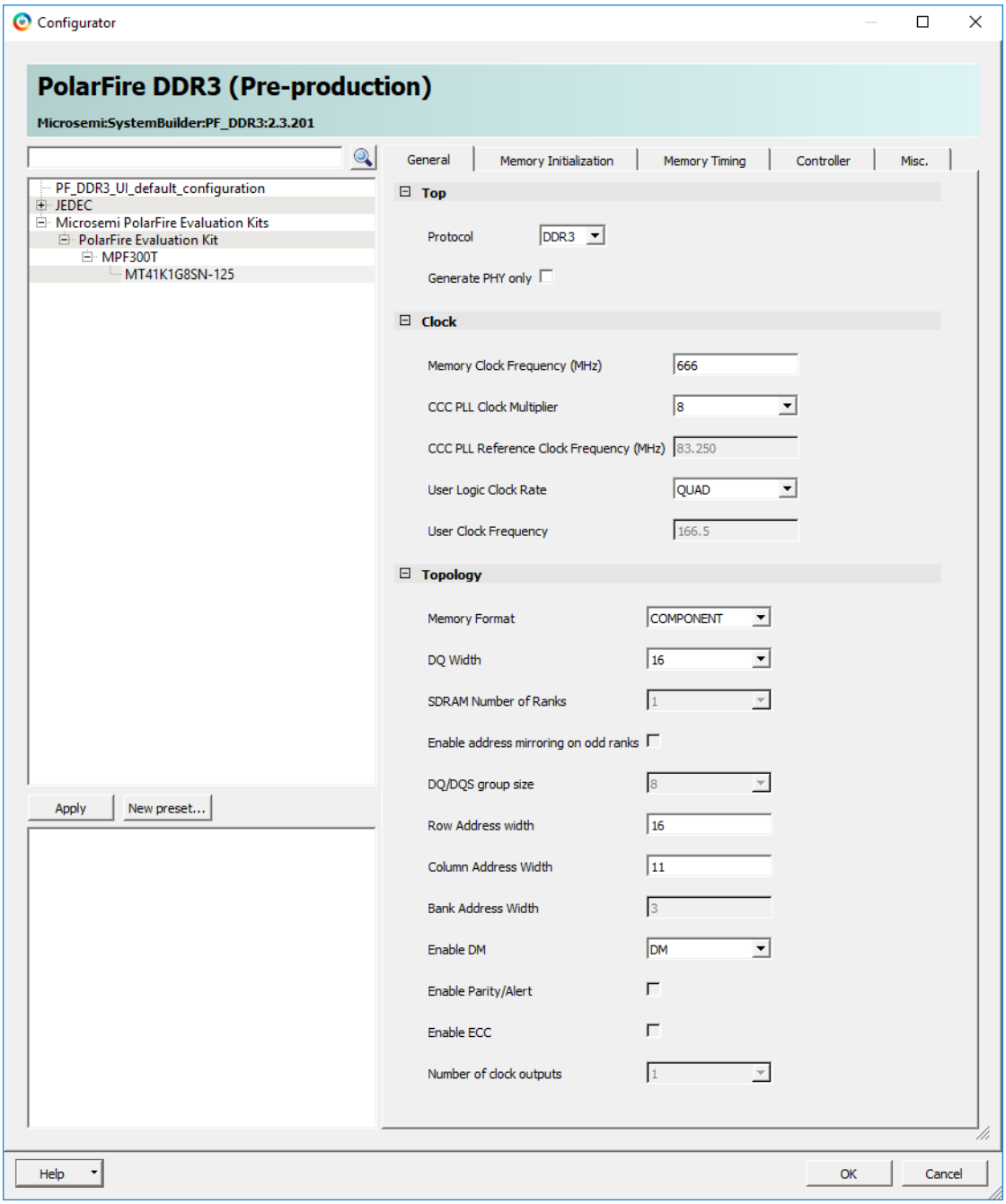
Figure 10 • **CoreAXI4Interconnect Configurator—Crossbar Configuration**



2.4.1.4 DDR3

The DDR3 subsystem is configured to access the 16-bit DDR3 memory through an AXI4 interface. The PolarFire evaluation kit DDR3 memory preset is applied to configure all of the memory initialization and timing parameters in the DDR configurator. The following figure shows general configuration settings for the DDR3 memory.

Figure 11 • **DDR3 Configuration**

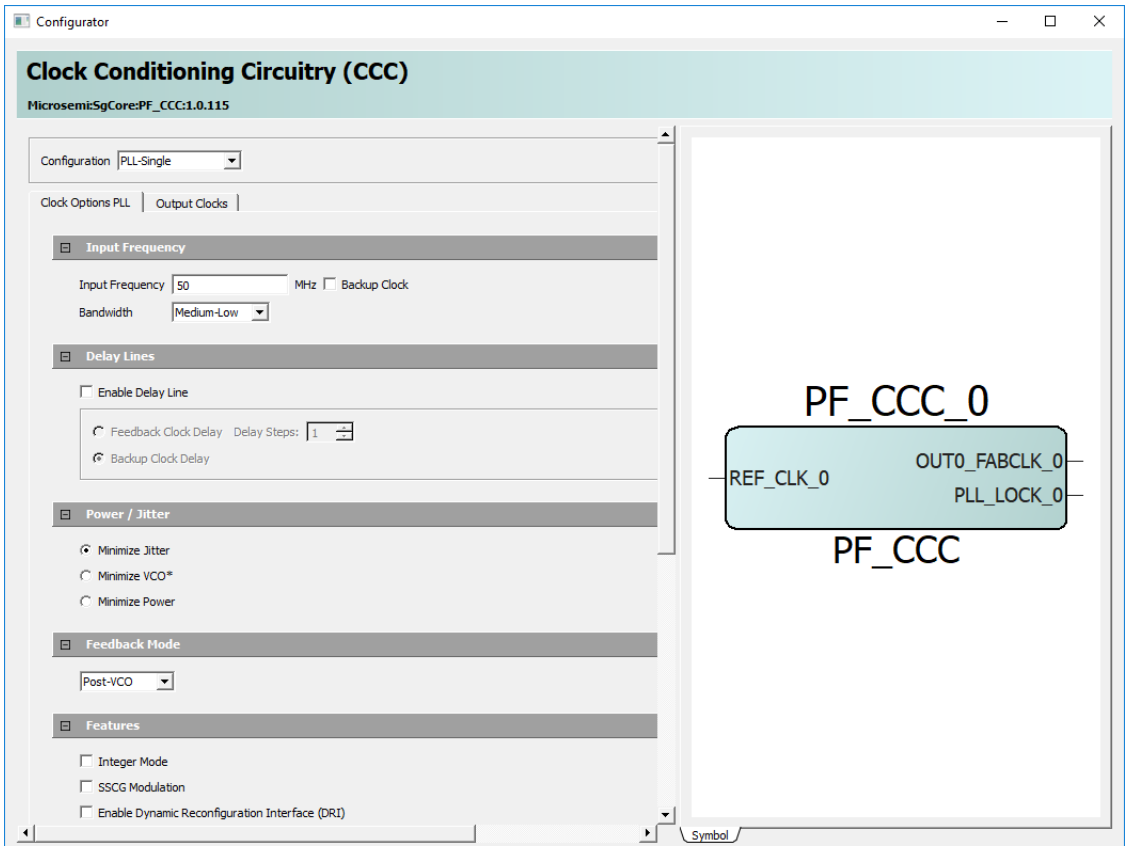


2.4.1.5 PF_CCC_0

The PF_CCC_0 (PolarFire Clock Conditioning Circuitry) generates the fabric reference clock that drives the soft processor and the APB peripherals. The PF_CCC_0 IP is configured to generate one output fabric clock from a 50 MHz input.

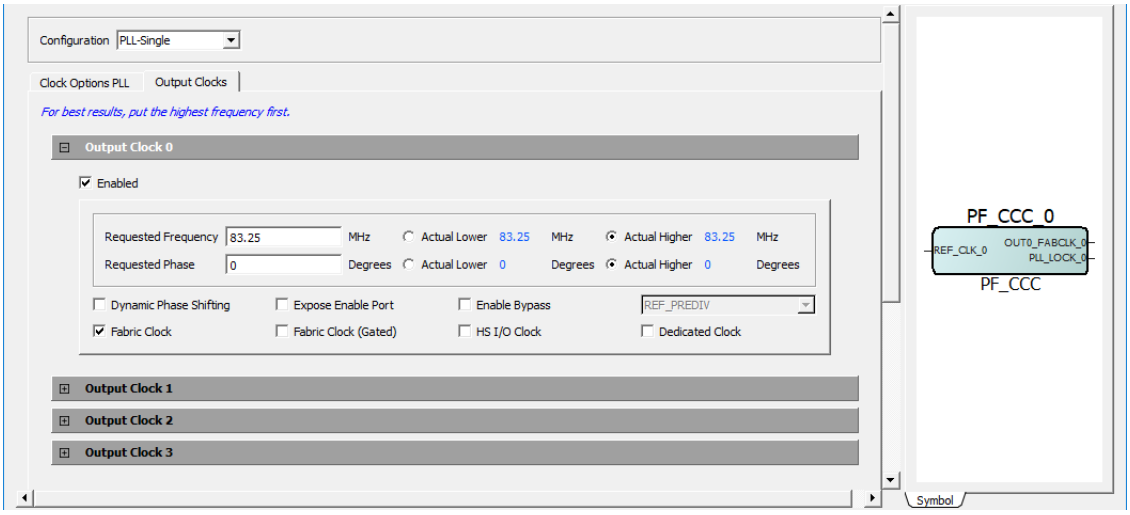
The following figure shows the PF_CCC_0 input clock configuration.

Figure 12 • **PF_CCC_0 Input Clock Configuration**



The following figure shows the PF_CCC_0 output clock configuration. The Mi-V processor supports up to 120 MHz. This design uses an 83.25 MHz system clock for configuring the APB peripherals.

Figure 13 • **PF_CCC_0 Output Clock Configuration**

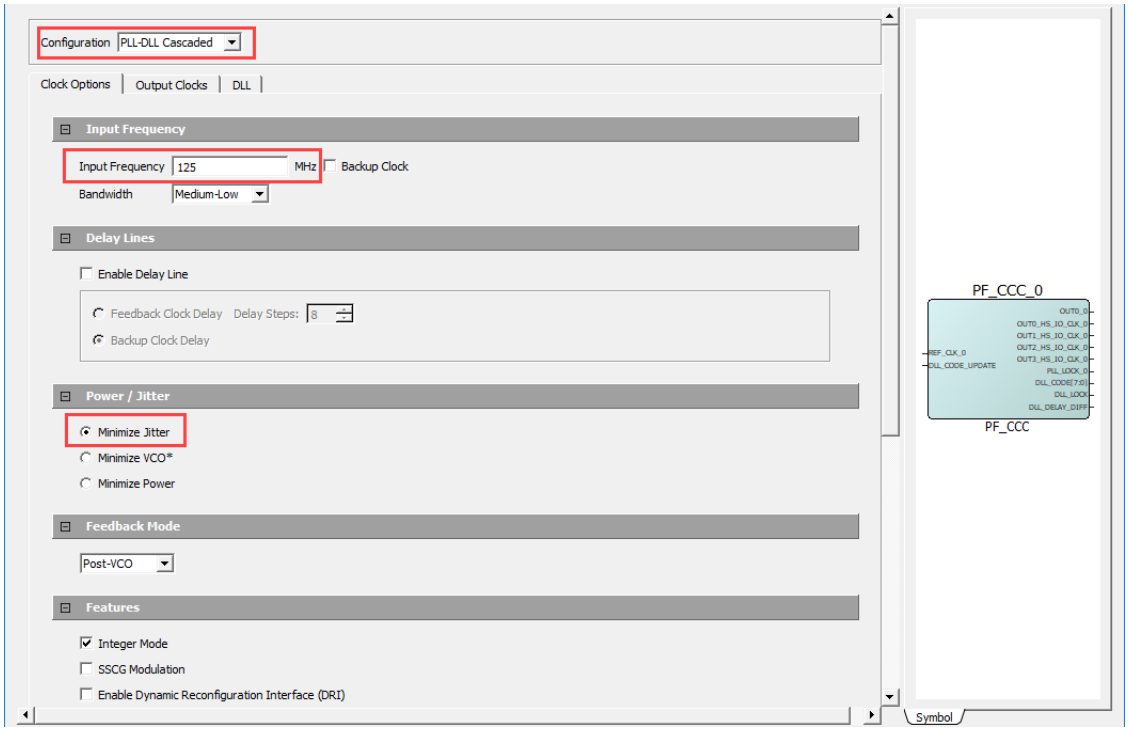


2.4.1.6 NWC_PLL_0

The NWC_PLL_0 block is used for IOD CDR. NWC_PLL_0 is configured in PLL-DLL cascaded mode to generate five output fabric clocks from a 125 MHz input. Four clocks are required for clock recovery and one clock for the fabric Tx interface of the CoreTSE block. The DLL is needed to control the clock position (delay) with DLL codes when the data is active on the Rx interface (RX_P/RX_N). A glitch-less DLL can adjust the clock delay setting when the data is active. Therefore, the HSIO clock frequency is selected as 625 MHz with four phases.

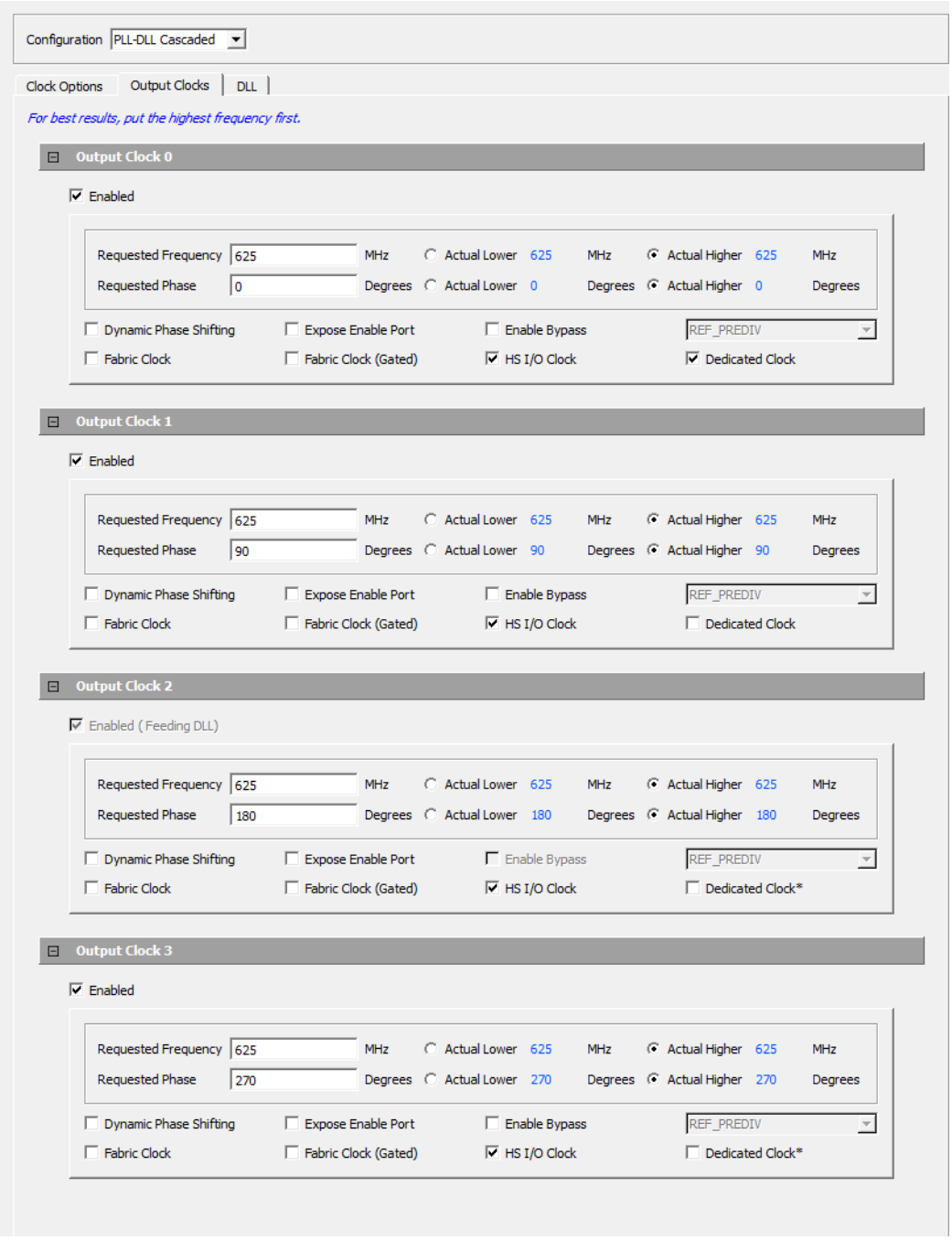
The following figure shows the NWC_PLL_0 input clock configuration.

Figure 14 • **NWC_PLL_0 Input Clock Configuration**



The following figure shows the NWC_PLL_0 output clock configuration. A bank clock is generated for 0, 90, 180, and 270 degrees, as shown in Figure 15, page 14. Output clock 2 is given as an input to the DLL. The output clock 0 (dedicated clock of 625 MHz) is sent to the clock divider to generate the 125 MHz frequency required for IOD CDR TX block and CoreTSE AHB block.

Figure 15 • **NWC_PLL_0 Output Clock Configuration**



The following figure shows the DLL configuration of NWC_PLL_0. The settings selected for DLL configuration are:

- **Clock Modes:** Phase Reference Mode.
- **Reference and Phase Shifting:** Output2. This indicates that the Output2 of the PLL is given as input to the DLL because this CCC was configured in the PLL-DLL cascaded mode.

Figure 16 • **NWC_PLL_0 DLL Configuration**

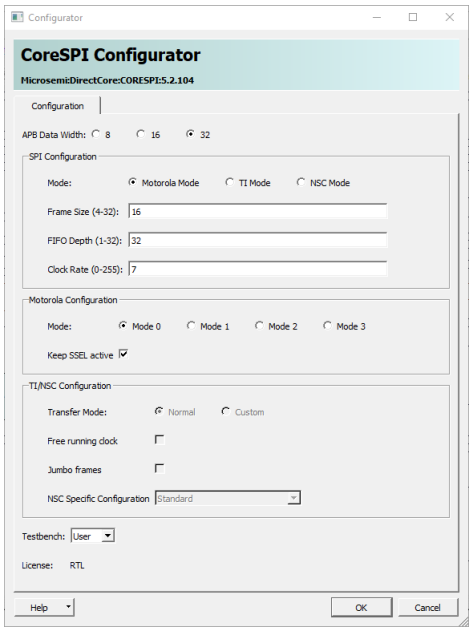
2.4.1.7 CORESPI_0

The CORESPI0 (CoreSPI) block is a controller IP, which implements serial communication. Mi-V configures the ZL30364 clock generation hardware using the CORESPI_0 block. The following points describe the CoreSPI configuration, as shown in the following figure.

- APB Data Width is selected as 32 because the design uses an APB data width of 32 bit.
- The default serial protocol mode, Motorola mode is retained to interface with ZL30364.
- Frame size is set to 16 to match the read/write cycles supported by ZL30364.
- FIFO depth is set to 32 to store maximum frames (TX and RX) in FIFO.
- The clock rate for the SPI master clock is selected as 7. This is used to generate the SPICLK, which is generated as PCLK/(2*(clock rate+1) = 83.25/(2*(7+1)).
- The **Keep SSEL active** checkbox is enabled to keep the slave peripheral active between back-to-back data transfers.

The following figure shows the CoreSPI configuration.

Figure 17 • **CoreSPI_0 Configuration**



2.4.1.8 Core_SPI_FLASH_0

This core is configured for accessing the external SPI flash for IAP. The configuration options are same as Figure 17, page 15.

2.4.1.9 CoreGPIO_0_0

This core is configured to control the on-board LEDs and switches.

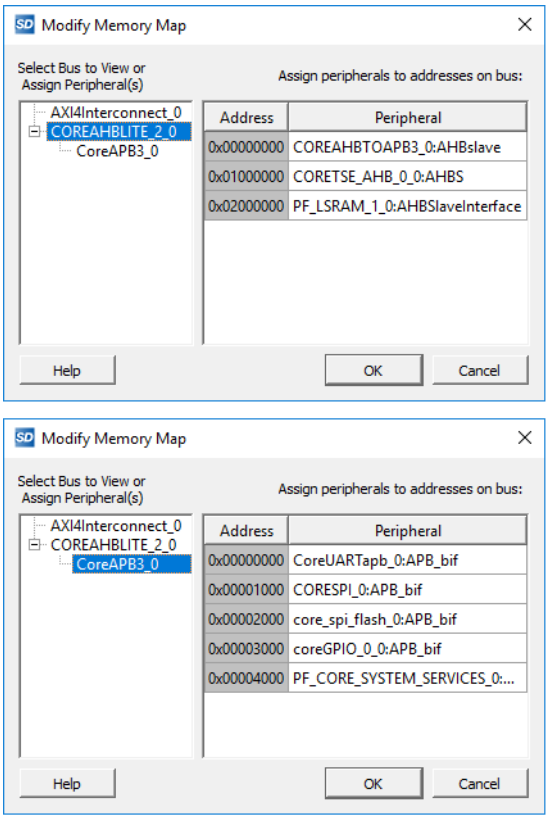
2.4.1.10 PF_SPI_0

This macro is an interface between system controller SPI and CoreSPI controller.

2.4.1.11 Design Memory Map

The following figure shows the Mi-V processor bus interface memory map.

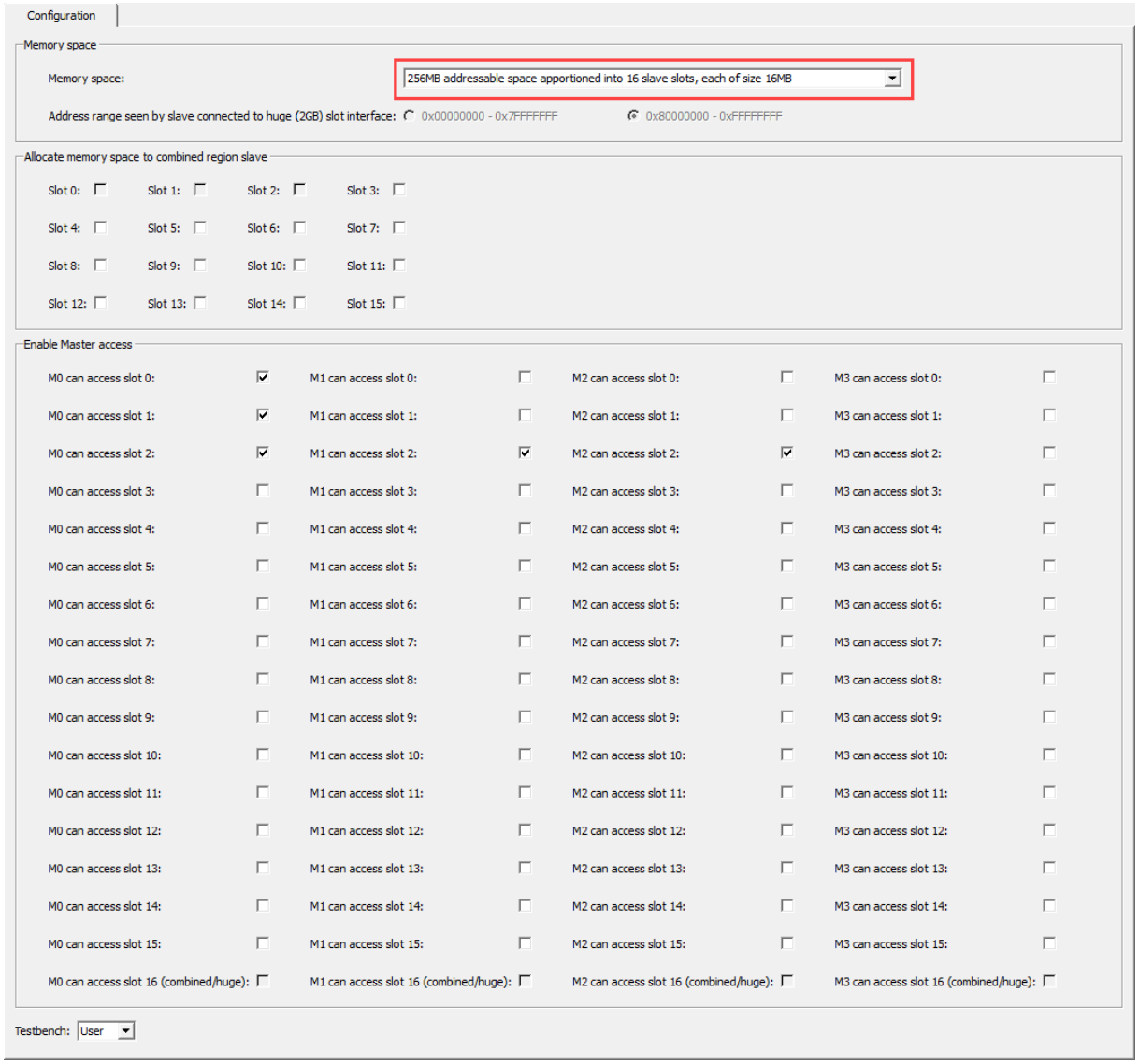
Figure 18 • **Mi-V Processor Bus Interface Memory Map**



2.4.1.12 CoreAHBLite_2

CoreAHBLite_2 is configured as shown in the following figure to interface the APB peripherals to the Mi-V processor at 0x6000_0000.

Figure 19 • **CoreAHBLite_2 Configuration**

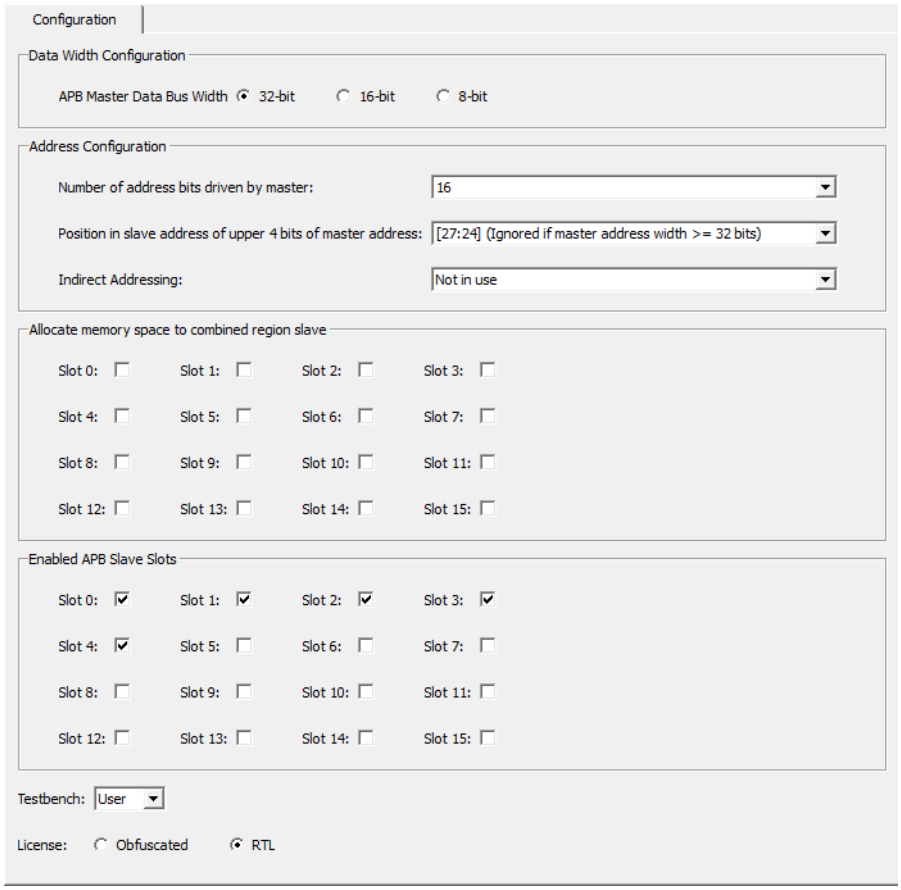


2.4.1.13 CoreAPB3_0

CoreAPB3_0 is configured as shown in the following figure to connect the peripherals CoreSPI, Core_SPI_Flash, CoreGPIO, PF_SYSTEM_SERVICES, and CoreUARTapb as slaves.

- APB Master Data bus width: 32 bit
- A number of address bits are driven by the master: 16. The Mi-V processor addresses slaves using 16-bit addressing, so the final address for these slaves translates to 0x6000_0000, 0x6000_1000, and 0x6000_2000
- Enabled APB Slave Slots: S0, S1, S2, S3, and S4 (for CoreSPI, Core_SPI_Flash, CoreGPIO, PF_SYSTEM_SERVICES, and CoreUARTapb, respectively.

Figure 20 • **CoreAPB3 Configuration**



2.4.1.14 COREAHBTOAPB3_0

The COREAHBTOAPB3 IP connects to CoreAPB3. This IP retains the default configuration.

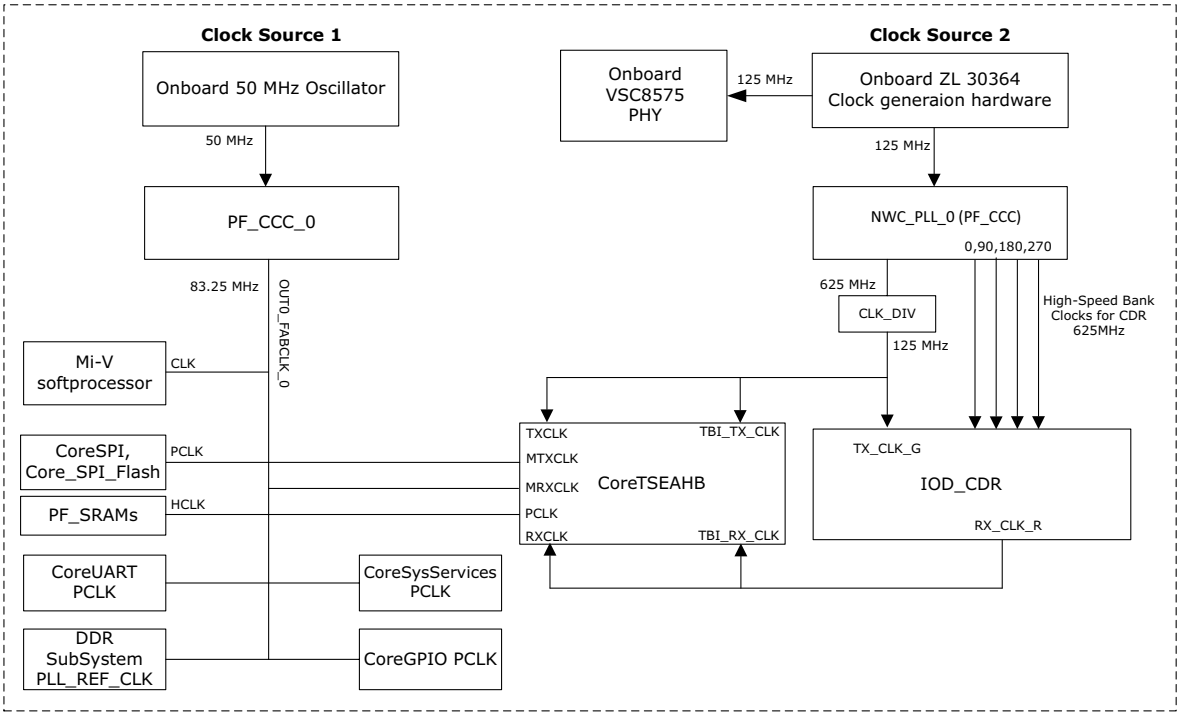
2.5 Clocking Structure

In the demo design, there are two clock domains—the on-board 50 MHz oscillator and the on-board ZL30364 clock generation hardware.

- **On-board 50 MHz oscillator:** This oscillator drives the PLL that generates an 83.25 MHz clock for the Mi-V soft processor and peripherals. The Mi-V soft processor can operate up to 120 MHz. In this design, the Mi-V processor runs at 83.25 MHz.
- **On-board ZL 30364 clock generation hardware:** This hardware generates the reference clocks for the VSC PHY and the IOD CDR fabric module.

The following figure shows the clocking structure of the demo design.

Figure 21 • **Clocking Structure**



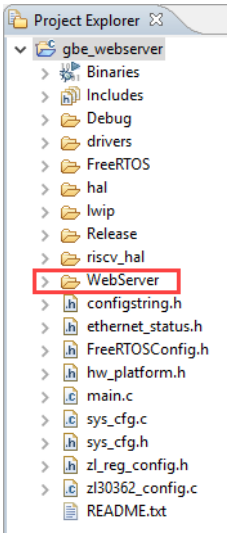
2.6 SoftConsole Firmware Project

The following stacks available in the SoftConsole Project Explorer are used in this demo design.

- lwIP TCP/IP stack v1.4.1
- FreeRTOS

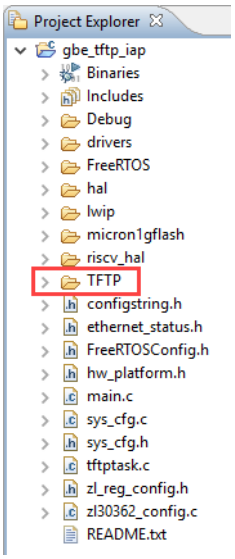
The following figure shows the directory structure of the Webserver SoftConsole project (located at *<$design_file_directory>\mpf_dg0834_df\webserver\libero\SoftConsole*). It contains the Webserver application (which uses LWIP and FreeRTOS) and all the firmware and hardware abstraction layers that correspond to the hardware design.

Figure 22 • **Directory Structure of Webserver SoftConsole Project**



The following figure shows the directory structure of the TFTP_IAP SoftConsole project (located at *<$design_file_directory>\mpf_dg0834_df\tftp_iap\libero\SoftConsole*). It contains the IAP application, TFTP server (which uses LWIP and FreeRTOS) and all the firmware and hardware abstraction layers that correspond to the hardware design.

Figure 23 • **Directory Structure of TFTP IAP SoftConsole Project**



3 Libero Design Flow

This chapter describes the Libero design flow for running this demo design, which includes:

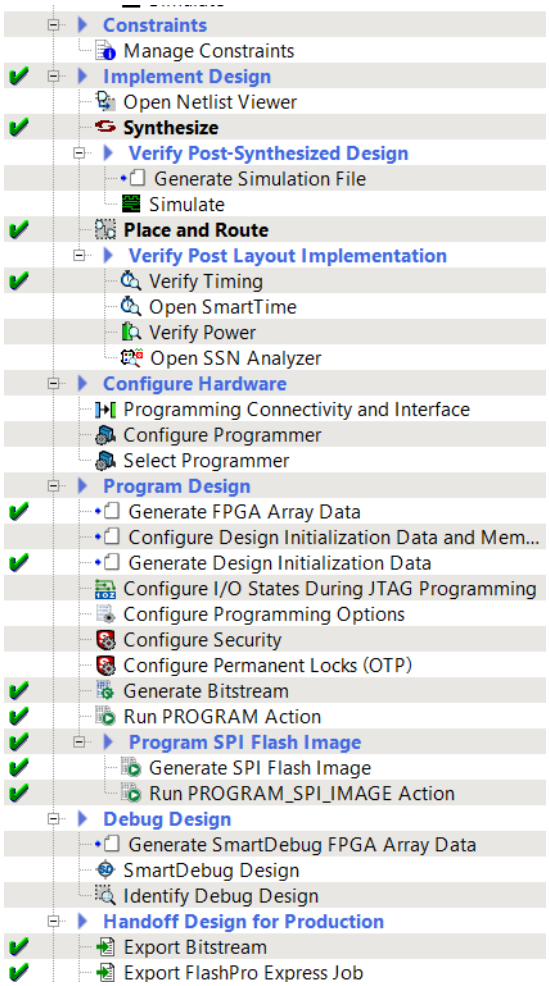
- Synthesize, page 22
- Place and Route, page 22
- Verify Timing, page 22
- Generate FPGA Array Data, page 23
- Configure Design Initialization Data and Memories, page 23
- Generate Bitstream, page 24
- Export FlashPro Express Job, page 24
- Run PROGRAM Action, page 25
- Program SPI Flash Image, page 26

The Libero project is available at the following design files folder location:

- **Webserver**: mpf_dg0834_df\webserver\libero
- **TFTP_IAP**: mpf_dg0834_df\tftp_iap\libero

The following figure shows these options in the **Design Flow t**ab.

Figure 24 • **Libero Design Flow Options**



3.1 Synthesize

To synthesize the design, perform the following steps:

1. On the **Design Flow** tab, double-click **Synthesize**.
   When the synthesis is successful, a green tick mark appears next to **Synthesize**, as shown in the preceding figure.
2. Right-click **Synthesize** and select **View Report** to view the synthesis report and log files in the **Reports** tab.

3.2 Place and Route

The demo project includes the IO PDC file and the floor planner PDC constraint files. The Place and Route process uses these PDC files to place the I/Os.

To place and route the design, perform the following steps:

1. On the **Design Flow** tab, double-click **Place and Route**.
   When place and route is successful, a green tick mark appears next to **Place and Route**, as shown in Figure 24, page 21.
2. Right-click **Place and Route** and select **View Report** to view the place and route report and the log files in the **Reports** tab.

3.2.1 Resource Utilization

The following table lists the resource utilization of the design after place and route. These values may vary slightly for different Libero runs, settings, and seed values.

Table 3 • **Resource Utilization**

Type	Used	Total	Percentage
4LUT	58856	299544	19.65
DFF	44408	299544	14.83
I/O register	0	1536	0.00
User I/O	87	512	16.99
– Single-ended I/O	75	512	14.65
– Differential I/O pairs	6	256	2.34

3.3 Verify Timing

To verify timing, perform the following steps:

1. On the **Design Flow** tab**,** double-click **Verify Timing**.
   When the design successfully meets the timing requirements, a green tick mark appears next to **Verify Timing**, as shown in Figure 24, page 21.

3.4 Generate FPGA Array Data

To generate FPGA array data, perform the following step:

1. On the **Design Flow** tab, double-click **Generate FPGA Array Data**.

When the FPGA array data is successfully generated, a green tick mark appears next to **Generate FPGA Array Data**, as shown in Figure 24, page 21.

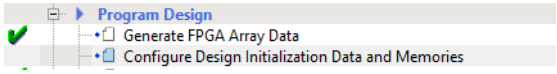
3.5 Configure Design Initialization Data and Memories

The **Configure Design Initialization Data and Memories** option creates the LSRAM initialization client. When the PolarFire device powers up, the LSRAM memory is initialized with the sNVM contents.

To create the LSRAM initialization client, perform the following steps:

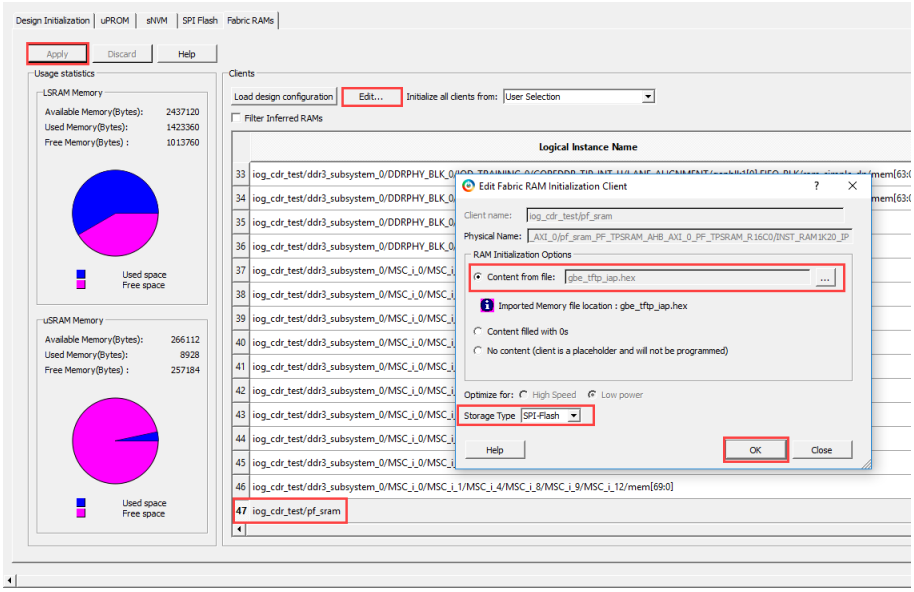
1. On the **Design Flow** tab, double-click **Configure Design Initialization Data and Memories**, as shown in the following figure.

Figure 25 • **Configure Design Initialization Data and Memories Option**



2. In the **Configure Design Initialization Data and Memories** window, select the **Fabric RAMs** tab, and then select the **pf_sram** file to import the memory information, as shown in the following figure.

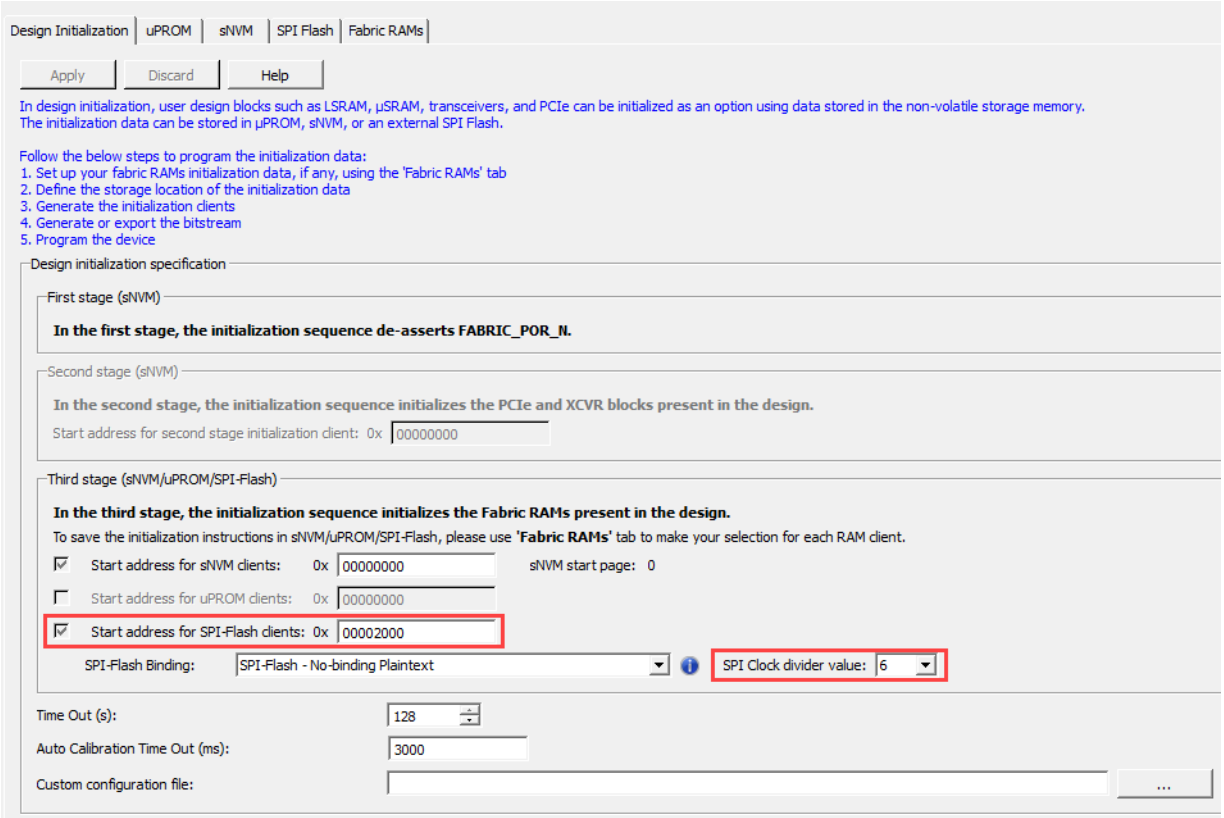
Figure 26 • **Fabric RAMs Tab**



3. Import the hex file (`gbe_webserver.hex` for Webserver design and `gbe_tftp_iap.hex` for TFTP_IAP design) provided with the design files from `mpf_dg0834_df\tftp_iap\libero\gbe_webserver.hex` **or** `mpf_dg0834_df\tftp_iap\libero\gbe_tftp_iap.hex`.
   The `gbe_webserver.hex` or `gbe_tftp_iap.hex` file is a application file generated using SoftConsole that configures the ZL clock generation hardware, the CoreTSE_AHB registers, and the VSC PHY. The application code is initially stored in an external SPI flash. On device power-up, the system controller copies the code to LSRAM from external SPI flash, and the Mi-V processor executes the code from LSRAM. To ensure that the fabric LSRAM contents are stored in external SPI flash, select **Storage Type SPI flash**, as shown in Figure 26, page 23.
4. Click **Apply**.

5. Select **Start address for SPI-FLash clients** and **SPI Clock divider value**, in the **Design Initialization** tab as shown in the following figure.

Figure 27 • **Start Address for SPI Flash Clients**



Note: The default start address for SPI-Flash clients 0x400 is used for Webserver design. The start address for TFTP design is modified to 0x2000. This is required to support flash erase of 4 KB while writing the SPI directory into initial SPI flash 1 KB memory using design firmware.

6. On the **Design Flow** tab, double-click **Generate Design Initialization Data**.
   When the LSRAM initialization client is successfully generated in sNVM, a green tick mark appears next to **Generate Design Initialization Data**, as shown in Figure 24, page 21.

   When the device is programmed, the LSRAM block is initialized from the sNVM.

3.6 Generate Bitstream

To generate the bitstream, perform the following steps:

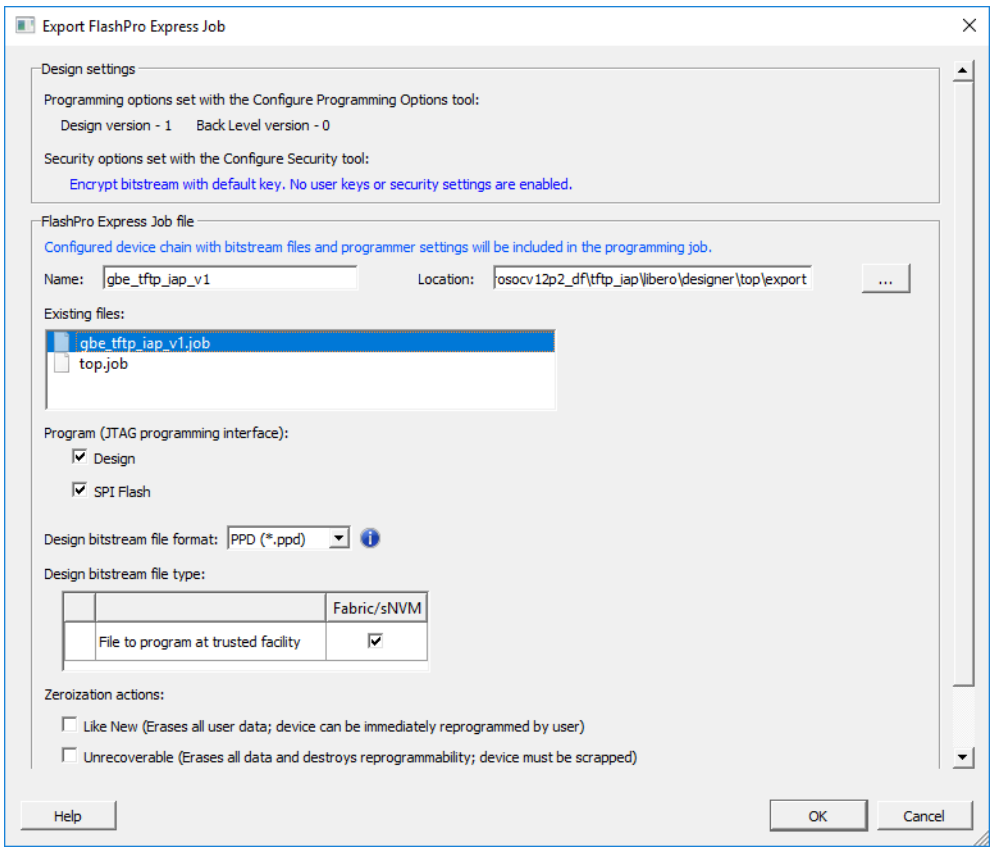
1. On the **Design Flow** tab, double-click **Generate Bitstream**.
   When the bitstream is successfully generated, a green tick mark appears next to **Generate Bitstream**, as shown in Figure 24, page 21.
2. Right-click **Generate Bitstream** and select **View Report** to view the corresponding log file in the **Reports** tab.

3.7 Export FlashPro Express Job

To generate .job file, perform the following steps:

On the **Design Flow** tab, double-click Export FlashPro Express Job and select Design and SPI Flash as shown in figure. The exported job file contains the data contents to be programmed into PolarFire FPGA and external SPI flash. This Job file is utilized in FlashPro Express software to program both Device and external SPI flash as shown in Appendix 3: Programming the Device and External SPI Flash Using FlashPro Express, page 42.

Figure 28 • **Export FlashPro Express Job**



3.8 Run PROGRAM Action

After generating the bitstream, the PolarFire device must be programmed. Follow these steps to program the PolarFire device using the Libero design flow:

Note: If you want to program the PolarFire FPGA using the .job file instead, see Appendix 3: Programming the Device and External SPI Flash Using FlashPro Express, page 42.

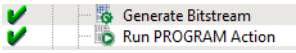
1. Ensure that the jumper settings on the board are as listed in the following table.

Table 4 • **Jumper Settings**

Jumper	Setting
J18, J19, J20, J21, and J22	Close pins 2 and 3 for programming through FTDI
J28	Close pins 1 and 2 for programming through the on-board FlashPro5
J4	Close pins 1 and 2 for switching the power manually using SW3
J12	Close pins 3 and 4 for 2.5 V

2. Connect the power supply cable to the **J9** connector on the board.
3. Connect the USB cable from the host PC to **J5** (FTDI port) on the board.
4. Connect any one of the open network 1G Ethernet capable ports to the **J15** connector (RJ45-PORT 0) on the board.
5. Power up the board using the **SW3** slide switch.
6. On the Libero **Design Flow** tab, double-click **Run PROGRAM Action**.
   When the device is successfully programmed, the LEDs 6 and 7 on the board glow, and a green tick mark appears, as shown in the following figure.

Figure 29 • **Run Program Action**



7. Right-click **Run Program Action** and select **View Report** to view the corresponding log file in the **Reports** tab.

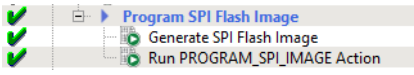
3.9 Program SPI Flash Image

To program SPI Flash Image, perform the following steps:

1. Double-click **Generate SPI Flash Image** and double-click **Run PROGRAM_SPI_IMAGE Action** to get the SPI flash programmed with the application as shown in the following figure.

Note: If you want to program the external SPI flash using the .job file instead, see Appendix: Programming the Device Using FlashPro Express, page 41.

Figure 30 • **SPI Flash Programming**



2. Power-cycle the board once you program the PolarFire device and external SPI flash.
   The demo is ready to be run. For information about how to run the demo, see Running the Demo, page 27.

4 Running the Demo

To run the demo design, perform the following steps:

1. For demo design files download link: http://soc.microsemi.com/download/rsc/?f=mpf_dg0834_df
2. Power up the board using the **SW3** slide switch.
3. Start a serial terminal emulation program such as HyperTerminal, PuTTY, or TeraTerm.

Note: For this demo, TeraTerm is used.

For more information about configuring serial terminal emulation programs, see *Configuring Serial Terminal Emulation Programs Tutorial*.

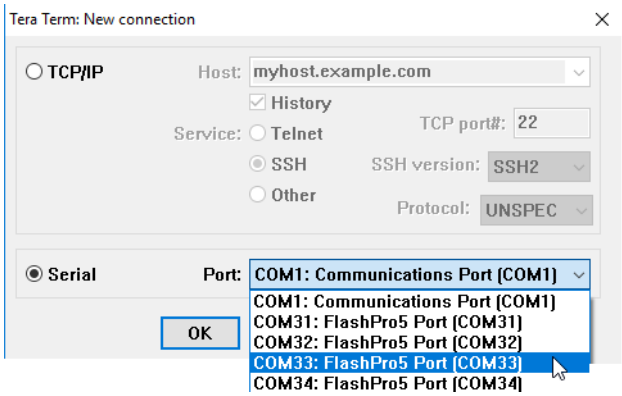
4.1 Tera Term Setup

The user application provides a user interface on the Tera Term terminal through the UART interface.

To set up the Tera Term program, perform the following steps:

1. Ensure that the USB cable connects the host PC to the **J5** (USB) port on the PolarFire Evaluation board.
2. Start Tera Term.
3. Select **Serial** as the Connection type.
4. Set the Serial **Port** to the second highest COM port number from the drop-down list as shown in the following figure. For example, **COM33: FlashPro5 Port [COM33]** in this instance.

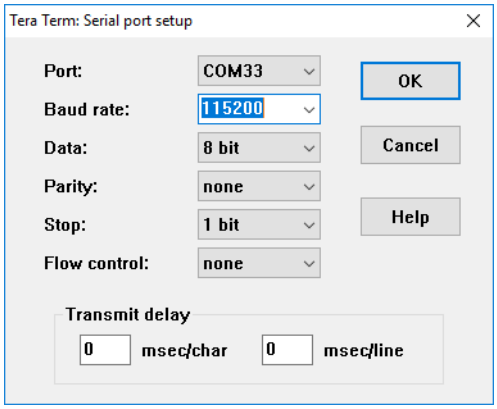
Figure 31 • **Select Serial as the Connection Type**



5. In the **Tera Term** window, go to **Setup > Serial port...**, set **Baud rate** to 115200

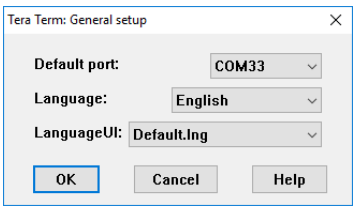
Rest of the serial port settings must be at default state as shown in the following figure and click **OK**.

Figure 32 • **Tera Term Configuration**



6. In the **Tera Term** window, go to **Setup > General...**, set the **Language** to **English** and click **OK**, as shown in the following figure. This setup is required for running the Tera Term macro script.

Figure 33 • **Tera Term General Setup**



This completes the Tera Term program setup.

4.2 Running Webserver Demo

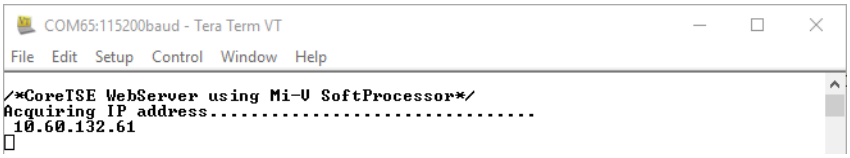
This section describes how to run the Webserver. The following procedure assumes that the serial terminal is setup, for more information about setting up the serial terminal, see Tera Term Setup, page 27.

Before you begin:

1. Connect the power supply cable to the **J9** connector on the board.
2. Connect the USB cable from the host PC to **J5** (FTDI port) on the board.
3. Open pin 1 and 2 of the **J23** jumper.
4. Connect any one of the open network 1G Ethernet capable ports to the **J15** connector (RJ45-PORT 0) on the board.
5. Power-up the board using the **SW3** slide switch.
6. Ensure that the device is programmed with the `gbe_webserver.job` file and external SPI flash is programmed with the application. See Program SPI Flash Image, page 26 to program the external SPI flash.

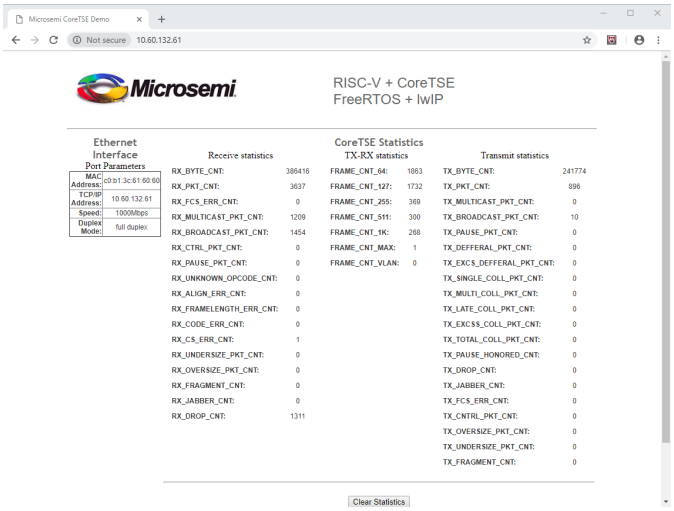
After the device is programmed, power cycle the board. The application prints a welcome message with an IP address on the Tera Term program through the UART interface, as shown in following figure.

Figure 34 • **Tera Term with IP Address**



Open a web browser, and enter the IP address displayed on the address bar of the browser. The PolarFire Webserver demo page appears, as shown in the following figure. To use the design in static IP mode, see Running the Design in Static IP Mode, page 40.

Figure 35 • **Webserver Demo Page**



4.3 Running TFTP Demo

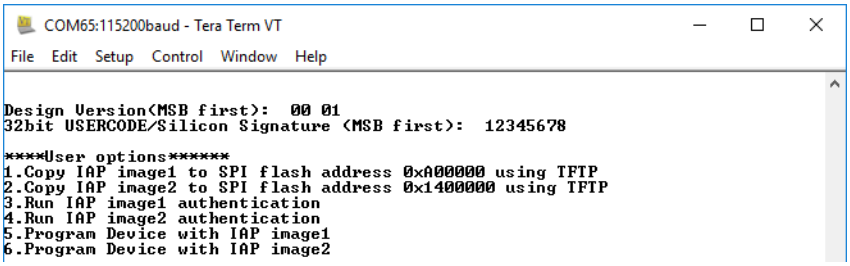
This section describes how to run the IAP using TFTP. The following procedure assumes that the serial terminal is setup, for more information about setting up the serial terminal, see Tera Term Setup, page 27.

Before you begin:

1. Connect the power supply cable to the **J9** connector on the board.
2. Connect the USB cable from the host PC to **J5** (FTDI port) on the board.
3. Open pin 1 and 2 of the **J23** jumper.
4. Power-up the board using the **SW3** slide switch.
5. Ensure that the device is programmed with the `gbe_tftp_iap_v1.job` file and external SPI flash is programmed with the application. See Program SPI Flash Image, page 26 to program the external SPI flash.
6. Enable TFTP client in Host PC. To enable the TFTP client in Host PC, see Appendix 1: Enable TFTP Client, page 35.

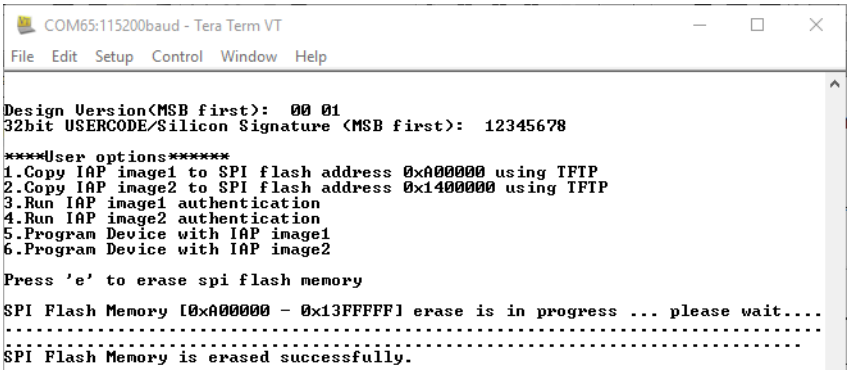
After power-up, Tera Term displays the options as shown in the following figure. Observe the design version **01** in the device.

Figure 36 • **Tera Term Window**



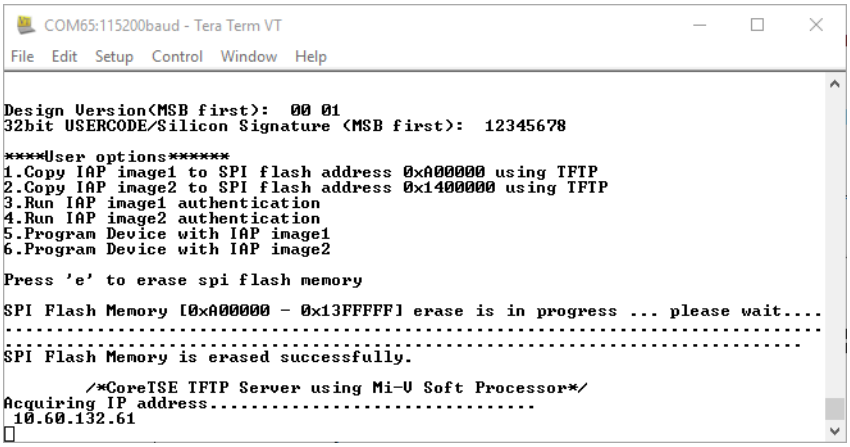
1. Press **1** to load IAP Image1 to SPI flash address 0xA00000 using TFTP.
2. Press **e** to erase the SPI flash memory location (0xA00000 - 0x13FFFFF).

Figure 37 • **Erasing the SPI Flash Memory Location [0xA00000 - 0x13FFFFF]**



3. After completion of the SPI flash erase operation, the Ethernet link is up, and the IP address is displayed on the Tera Term terminal. In this example, the IP address is 10.60.132.61. The TFTP command uses this IP address to transfer the file to the external SPI flash. The LED **G1** on the PolarFire Evaluation Kit board starts blinking. To use the design in static IP mode, see Running the Design in Static IP Mode, page 40.

Figure 38 • **Acquiring IP Address**

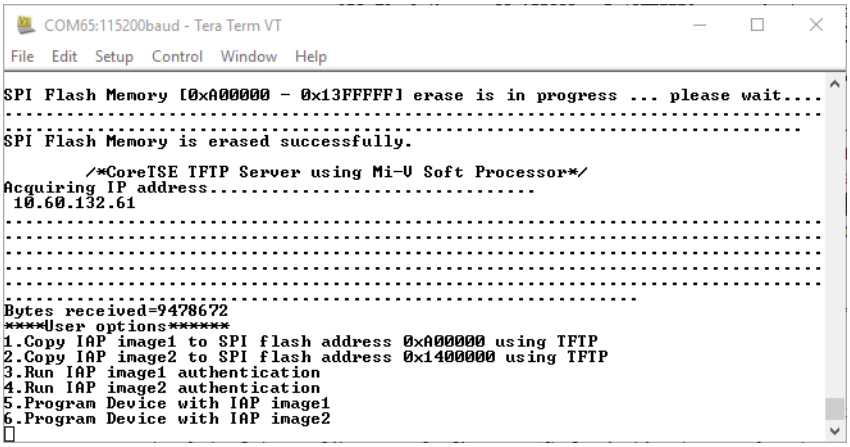


4. On the Host PC command prompt, browse to the folder *<$design file directory>\mpf_dg0834_df\tftp_iap\iap_images*
5. Type the `tftp -i 10.60.132.61 PUT iog_cdr_tftp_iap_v2.spi` command to transfer the `iog_cdr_tftp_iap_v2.spi` programming file to the SPI flash as shown in the following figure.

Figure 39 • **Transfer Programming Image1**

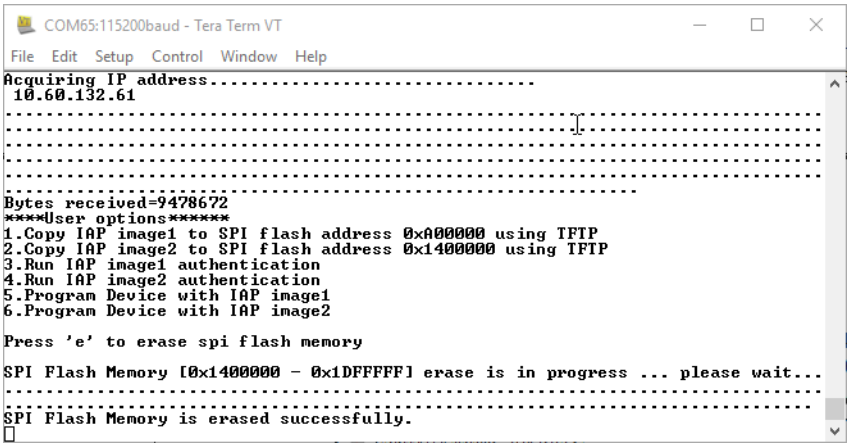
6. Wait until total bytes received message is displayed on the Tera Term, to ensure that the programming image1 is transferred to the SPI flash. On completion of the Image1 transfer, the user options are displayed.

Figure 40 • **Bytes Received for Image1**



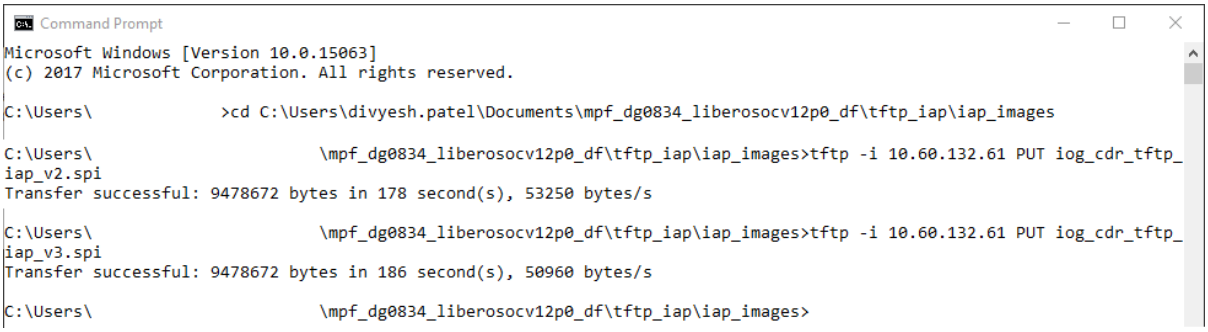
7. Press **2** to load IAP Image2 to SPI flash address 0x1400000 using TFTP.
8. Press **e** to erase the SPI flash memory location (0x1400000 - 0x1DFFFFF).

Figure 41 • **Erasing the SPI Flash Memory Location [0x1400000 - 0x1DFFFFF]**



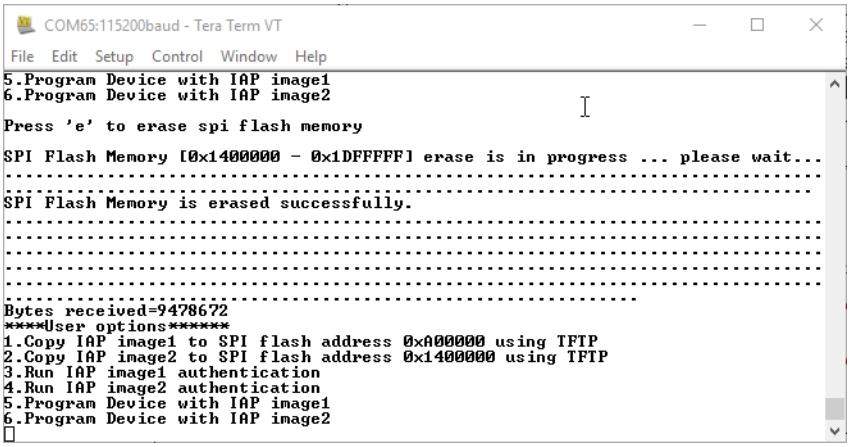
9. On the Host PC command prompt, make sure to browse the folder *<$design file directory>\mpf_dg0834_df\tftp_iap\iap_images*
10. Type the `tftp -i 10.60.132.61 PUT iog_cdr_tftp_iap_v3.spi` command to transfer the programming image2 as shown in the following figure.

Figure 42 • **Transfer IAP Image2**



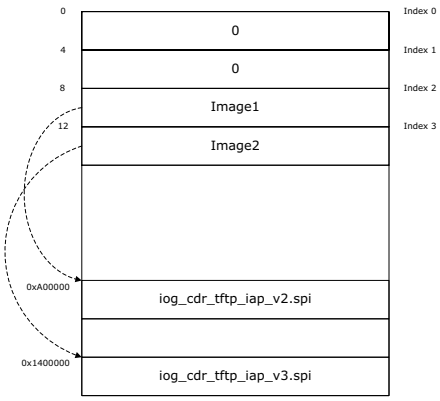
11. Wait until total bytes received message is displayed on the Tera Term, to ensure that the programming image2 is transfered to the SPI flash. On completion of the Image2 transfer, the user options are displayed.

Figure 43 • **Bytes Received for Image2**



The images are transferred successfully to external SPI flash using TFTP. The firmware application takes care of external SPI flash programming with SPI directory as shown in the following figure.

Figure 44 • **SPI Directory**

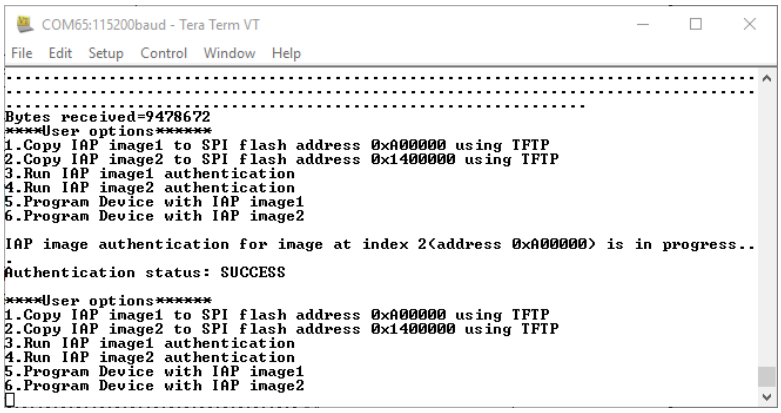


4.3.1 Running IAP Authentication

To run the IAP authentication, perform the following steps:

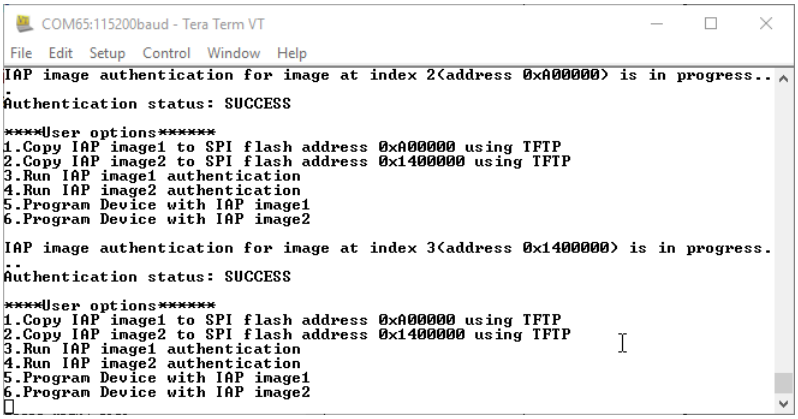
1. Press 3 to initiate IAP image1 authentication. The IAP authentication with image at index 2 is executed successfully. Tera Term displays the status code as shown in the following figure.

Figure 45 • **Successful IAP Image1 Authentication**



2. Press 4 to initiate the IAP image2 authentication. The IAP authentication with image at index 3 is executed successfully. Tera Term displays the status code, as shown in the following figure.

Figure 46 • **Successful IAP Image2 Authentication**



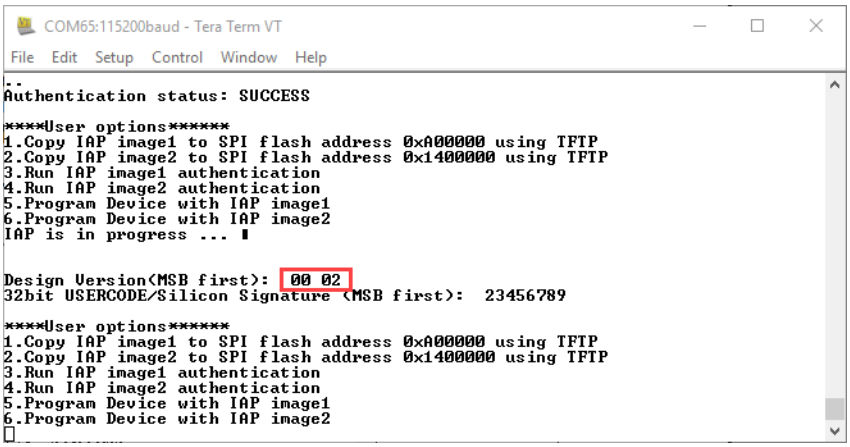
This concludes the IAP image authentication.

4.3.2 Running IAP Program

To run the IAP with programming images, perform the following steps:

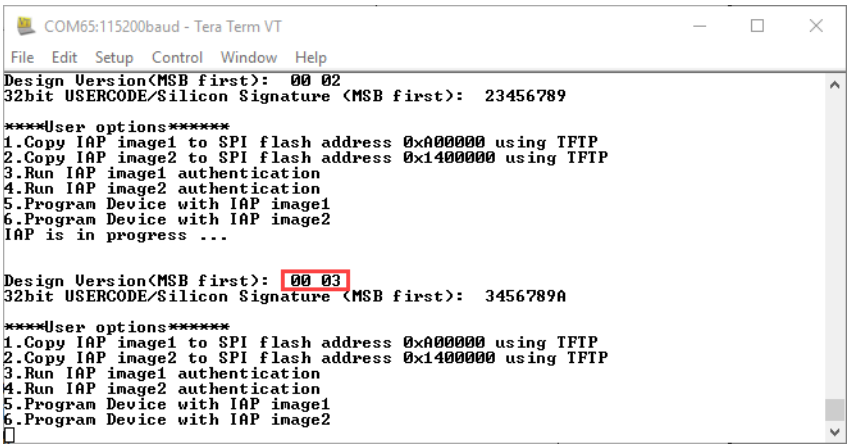
1. Press 5, **Program Device with IAP image1**. The IAP program with image1 is executed successfully and the design version **02** with different silicon signature is displayed as shown in the following figure. This operation takes few seconds.

Figure 47 • **Successful IAP with Image1**



2. Press 6, **Program Device with IAP image2**. The IAP program with image2 is executed successfully and the design version **03** with different silicon signature is displayed as shown in the following figure. This operation takes few seconds.

Figure 48 • **Successful IAP by Image2**



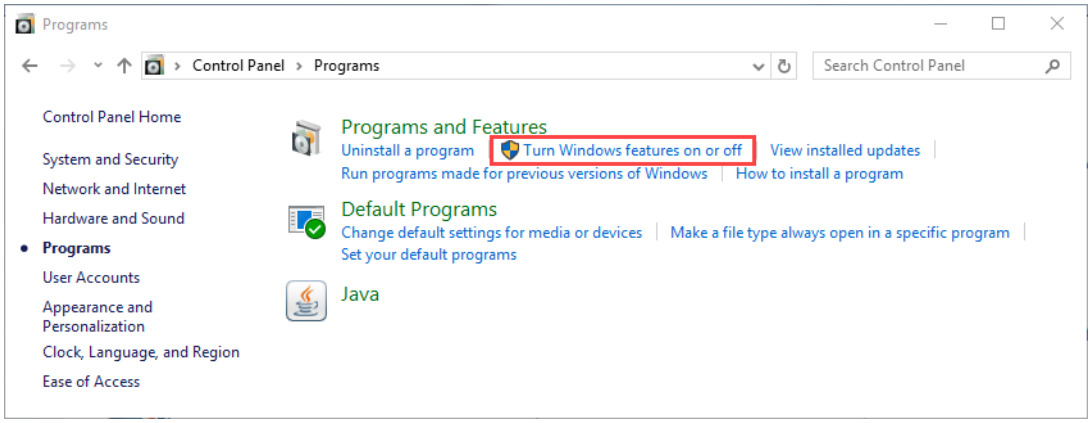
This concludes running the IAP Program with images.

5 Appendix 1: Enable TFTP Client

The following steps describe how to enable TFTP client:

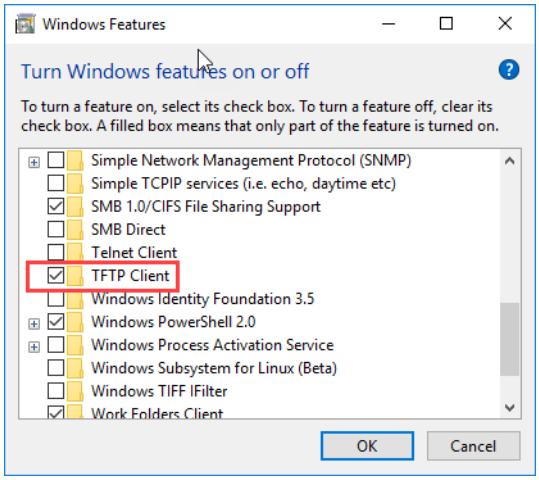
1. Navigate to **Control Panel** > **Programs**. Click **Turn Windows features on or off** as shown in the following figure.

Figure 49 • **Control Panel—Programs and Features**



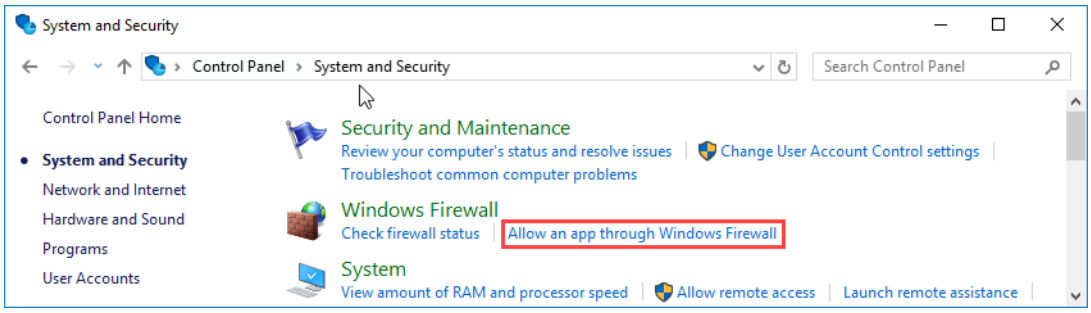
2. Select the **TFTP Client** check box from Windows Features as shown in the following figure.

Figure 50 • **Selecting TFTP Client from Windows Features**



3. Browse through **Control Panel** > **System and Security** and click **Allow an app through Windows Firewall**.

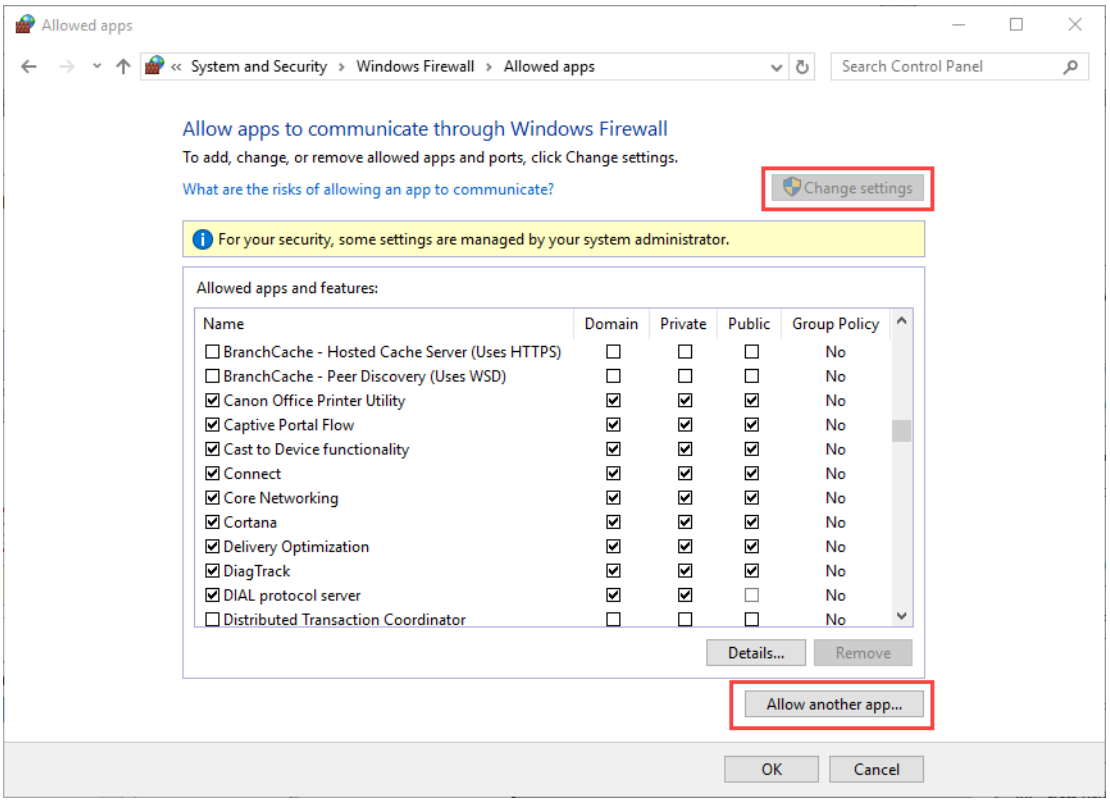
Figure 51 • **System and Security Window**



Note: If the System and Security option is not available, then enter the firewall in the search window to perform step 3.

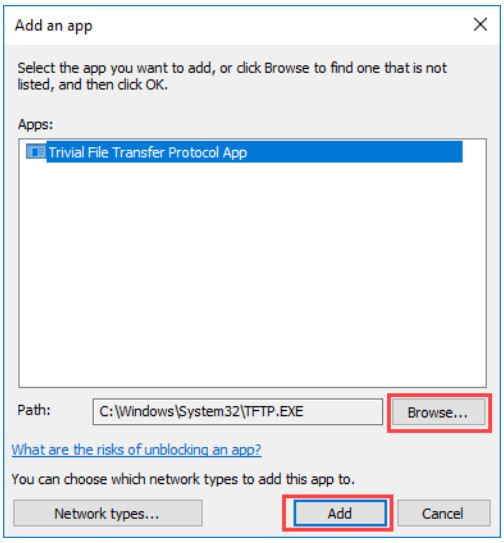
4. Click **Change settings** and choose **Allow another program…**

Figure 52 • **Allow Programs Window**



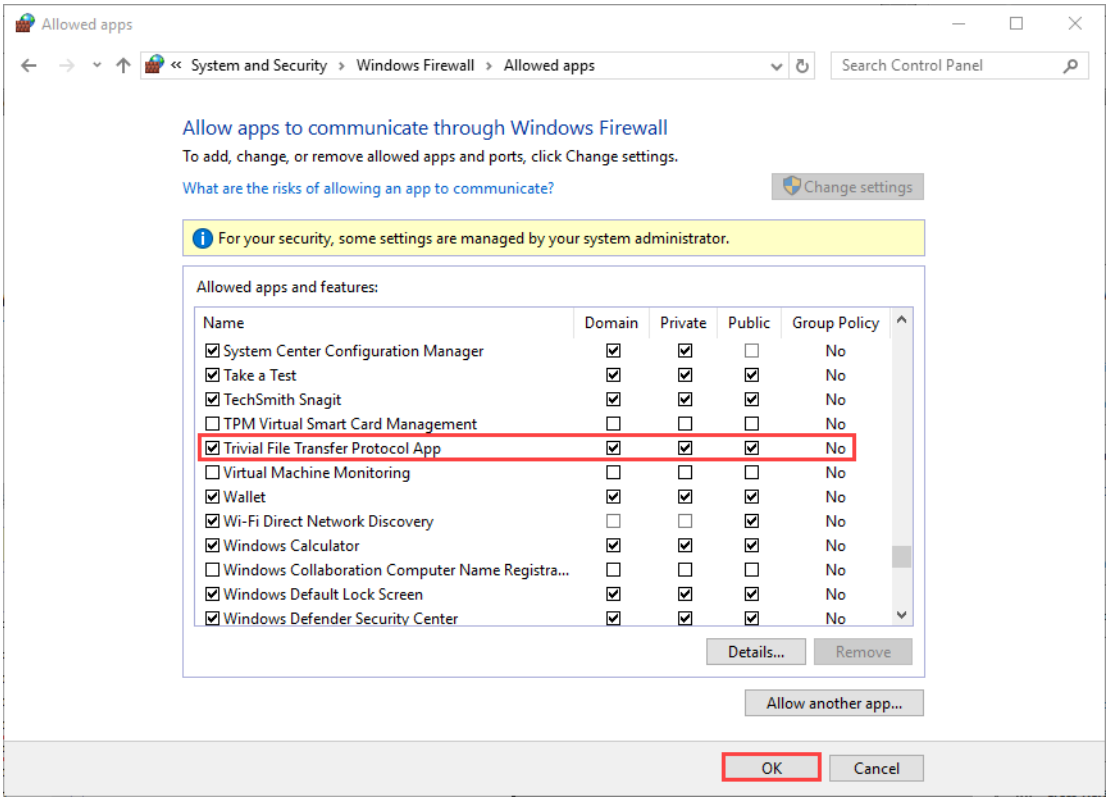
5. The **Add an app** window is displayed and click **Browse…**
6. Browse through `C:\ -> Windows->System32` and choose **TFTP.exe** and click **Open**.
7. Ensure that the TFTP.EXE path (C:\Windows\System32\TFTP.EXE) is selected correctly and click **Add.**

Figure 53 • **Add an app Window**



8. Ensure that the **Trivial File Transfer protocol App** is added and also select all the check boxes (Domain, Home/Work, and Public) as shown in the following figure.

Figure 54 • **Selecting Trivial File Transfer Protocol App in Allowed apps Window**



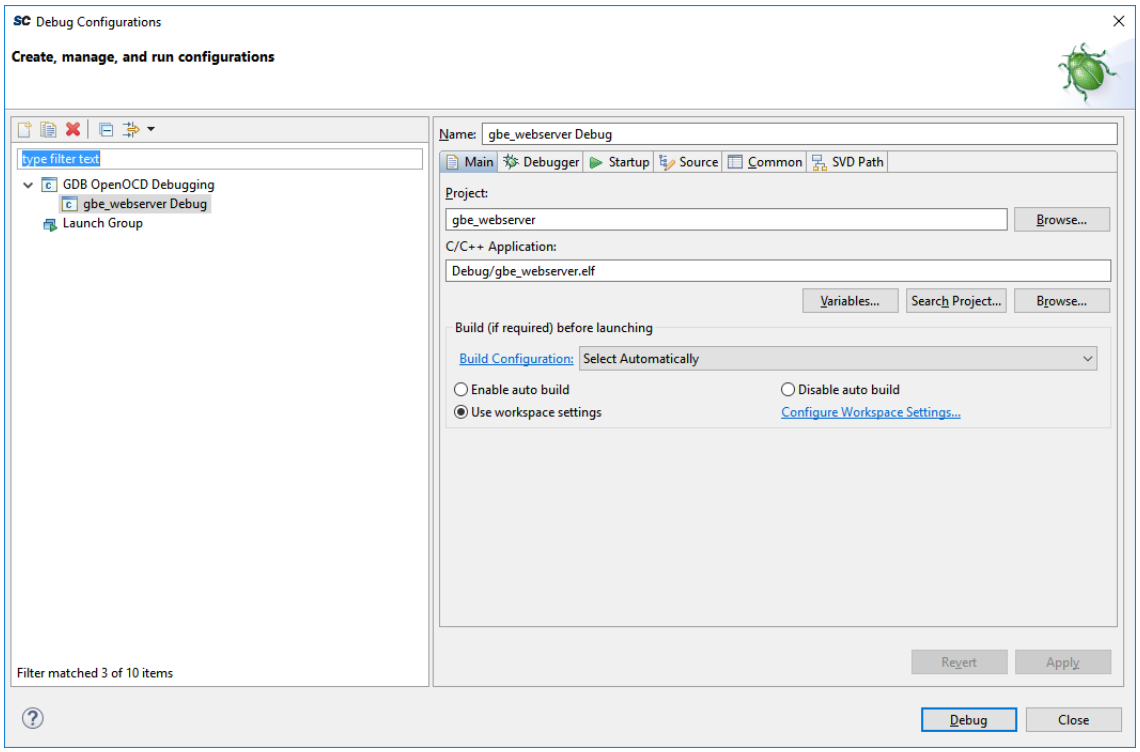
9. Click **OK**.

6 Appendix 2: Running the SoftConsole Project in Debug Mode from LSRAM or DDR Memory

The following steps describe how to run the SoftConsole project in debug mode.

1. Open the Webserver or TFTP_ IAP application project using SoftConsole.
   - Webserver SoftConsole project location: mpf_dg0834_df\webserver\libero\SoftConsole
   - TFTP_ IAP SoftConsole project location: mpf_dg0834_df\tftp_iap\libero\SoftConsole
2. In SoftConsole, select **Run > Debug Configurations**. The Debug Configurations dialog box displayed. To debug the Webserver project, select **gbe_webserver Debug**, as shown in the following figure.

Figure 55 • **SoftConsole Debug Configuration**

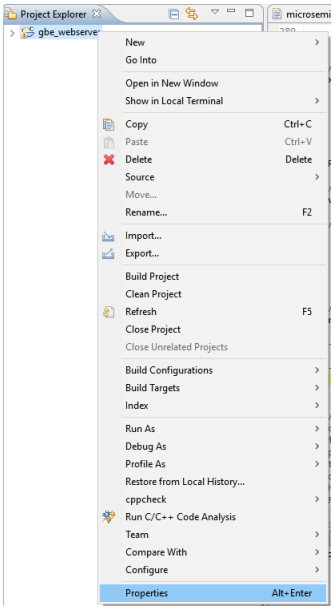


3. Click **Debug**. The tool copies the code to LSRAM memory and launches the debug session. This SoftConsole project is configured to debug from LSRAM.

Debugging the application from DDR memory:

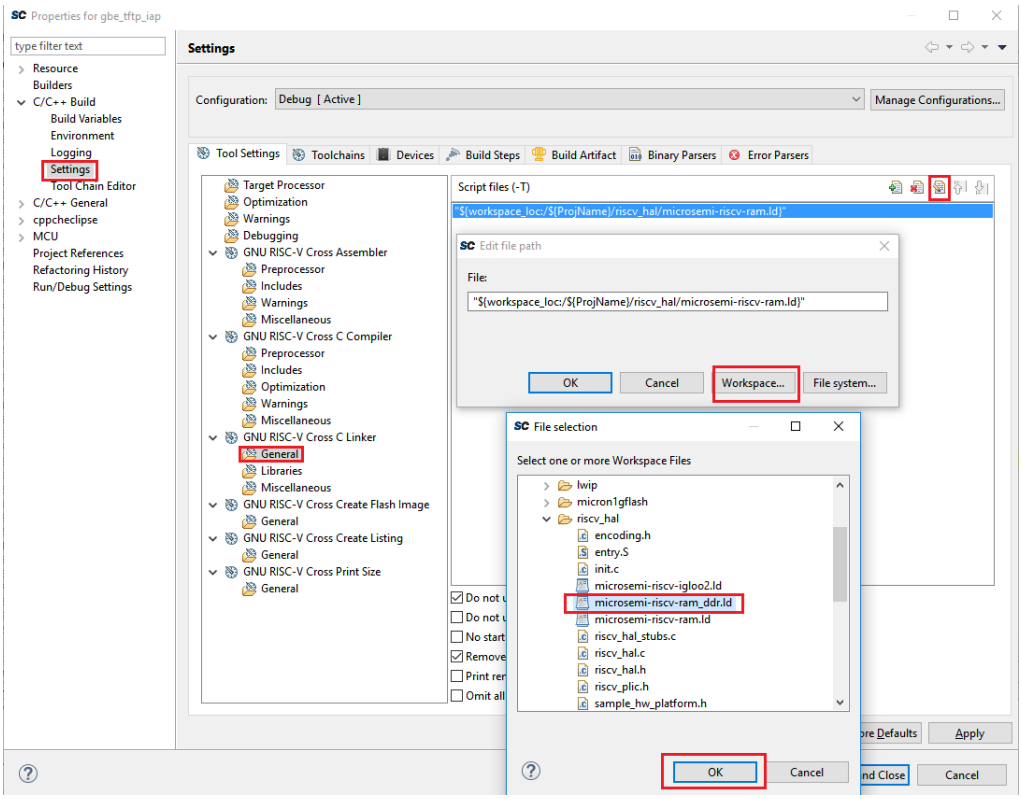
1. In the SoftConsole Project Explorer window, right-click **gbe_webserver** project, and select **Properties**, as shown in the following figure.

Figure 56 • **Project Explorer Window**



2. Change the linker script file setting to `microsemi-riscv-ram_ddr.ld` and re-build the project.

Figure 57 • **Project Properties**



3. In SoftConsole, select **Run > Debug** Configurations to debug the application from DDR memory.

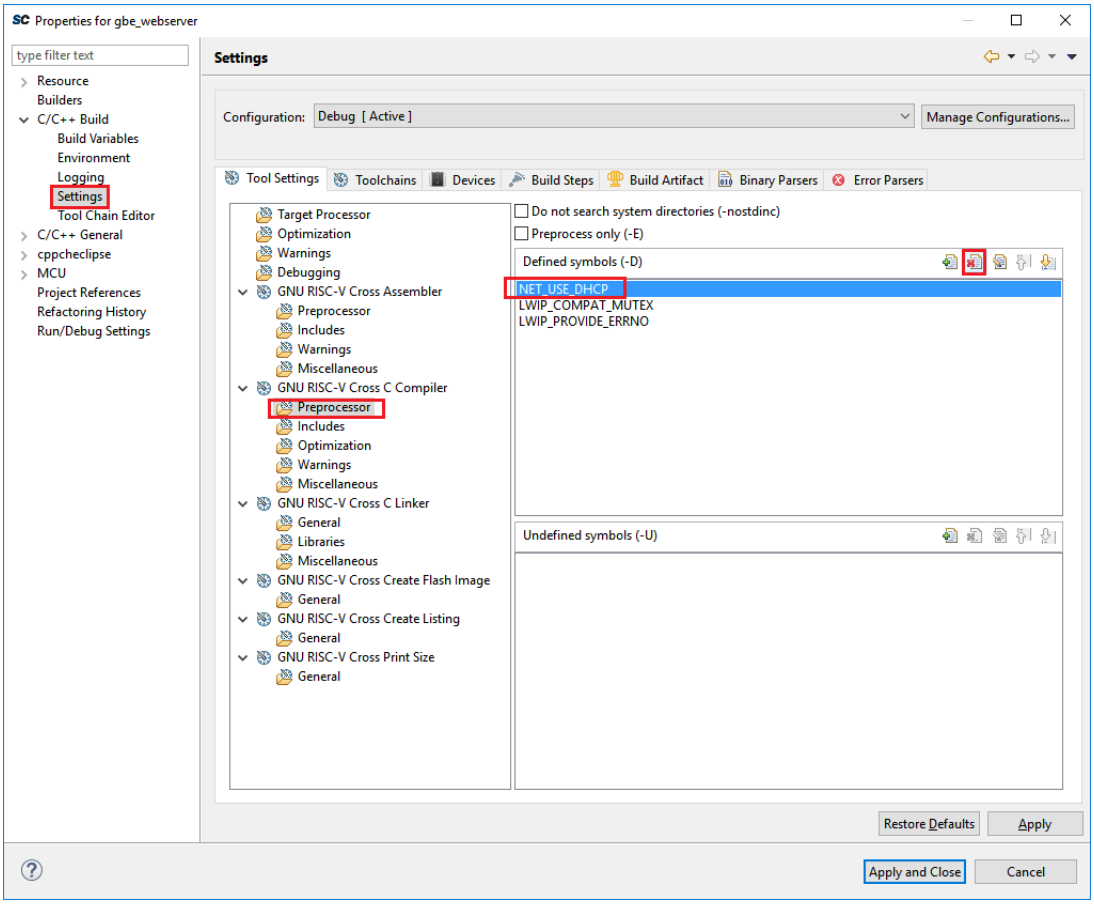
6.1 Running the Design in Static IP Mode

The following steps describe how to run the design in static IP mode.

Note: This procedure provide steps to run the Webserver design. To run the IAP using TFTP design, perform the same steps by opening the IAP_TFTP project in SoftConsole.

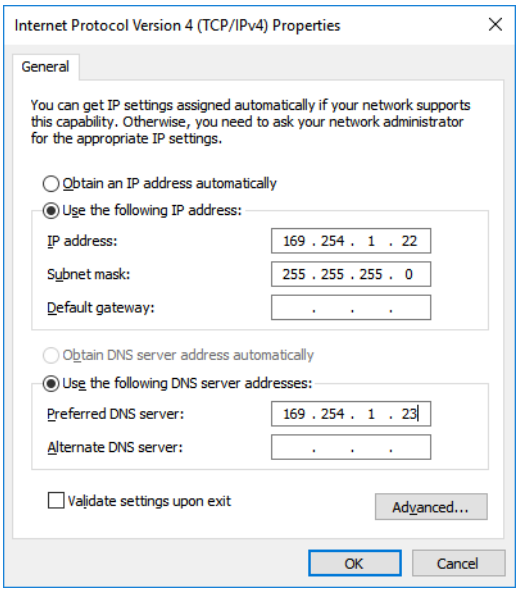
1. In SoftConsole Project Explorer window, right-click the Webserver (**gbe_webserver**) project, and select **Properties**, as shown in Figure 56, page 39.
2. In **Properties for gbe_webserver** window, remove the **NET_USE_DCHP** symbol listed under **Defined symbols (-D)**, and click **Apply**, as shown in the following figure.

Figure 58 • **Properties for gbe_webserver**



3. Change the host TCP/IP settings to connect with the board which has static IP address, **169.254.1.23**. The following figure shows the host PC TCP/IP settings.

Figure 59 • **Host PC TCP/IP Settings**



4. Connect the PolarFire Evaluation board port **J15** to Host PC using RJ45 Ethernet cable.
5. After configuring the settings, compile the design, load it into memory, and run it using SoftConsole. The Serial terminal shows board static IP:

```
/*CoreTSE WebServer using Mi-V SoftProcessor*/
Acquiring IP address.
 169.254.1.23
```

6. Use the IP address in web browser to display the Microsemi web page.

7 Appendix 3: Programming the Device and External SPI Flash Using FlashPro Express

This section describes how to program the PolarFire device and external SPI flash with the .job programming file using FlashPro Express. The .job file is available at the following design files folder location:

- **Webserver**: mpf_dg0834_df\webserver\programming_job
- **TFTP_IAP**: mpf_dg0834_df\tftp_iap\programming_job

To program the device, perform the following steps:

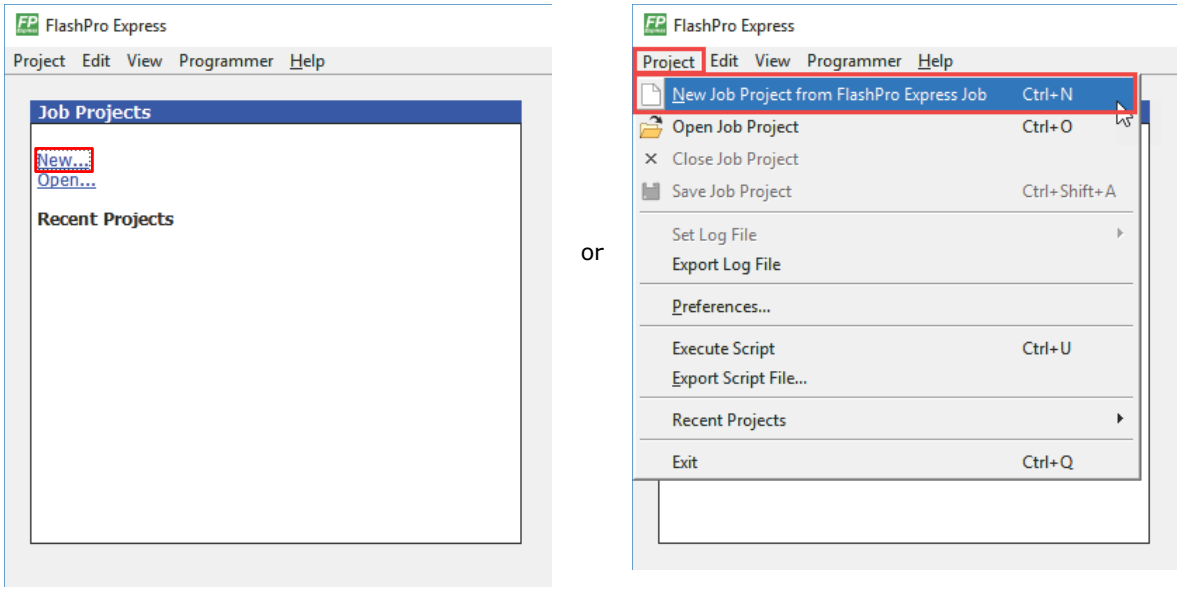
1. Ensure that the jumper settings on the board are the same as listed in Table 4, page 25.

Note: The power supply switch must be switched off while making the jumper connections.

2. Connect the power supply cable to the **J9** connector on the board.
3. Connect the USB cable from the Host PC to the **J5** (FTDI port) on the board.
4. Power on the board using the **SW3** slide switch.
5. On the host PC, launch the **FlashPro Express** software.
6. To create a new job project, click **New** or

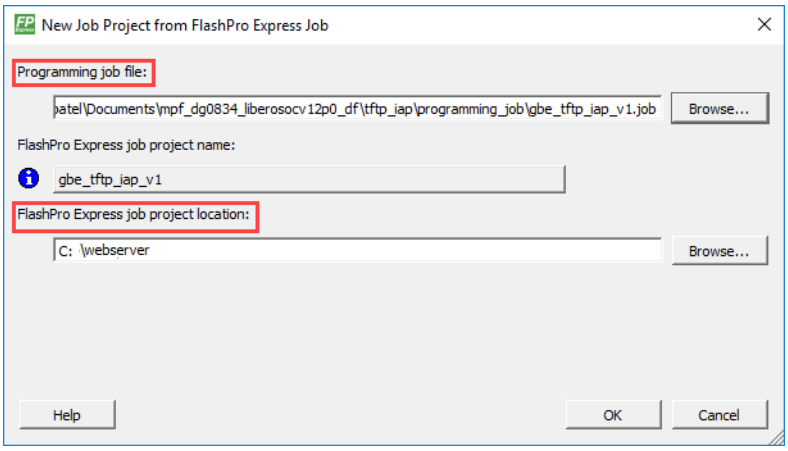
   In the Project menu, select **New Job Project from FlashPro Express Job,** as shown in the following figure.

Figure 60 • **FlashPro Express Job Project**



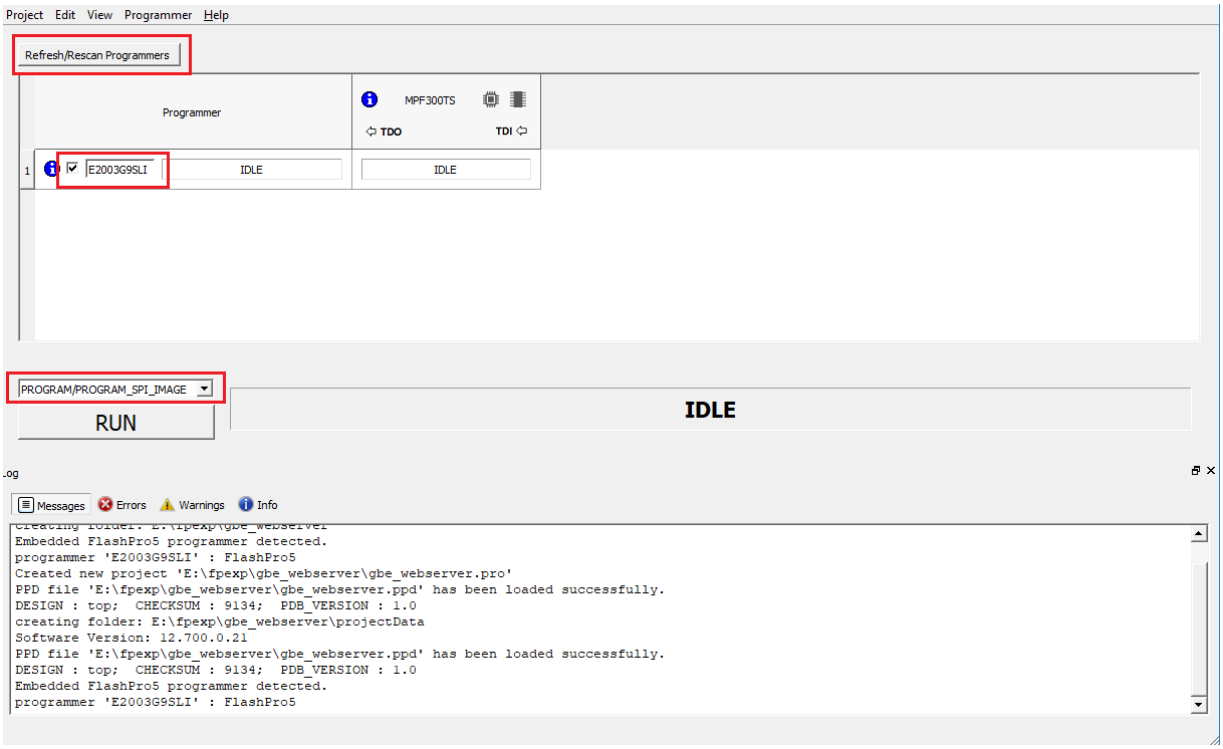
7. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:
   - **Programming job file**: Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is: *<download_folder>\mpf_dg0834_df\webserver\programming_job or <download_folder>\mpf_dg0834_df\tftp_iap\programming_job*
   - **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.

Figure 61 • **New Job Project from FlashPro Express Job**



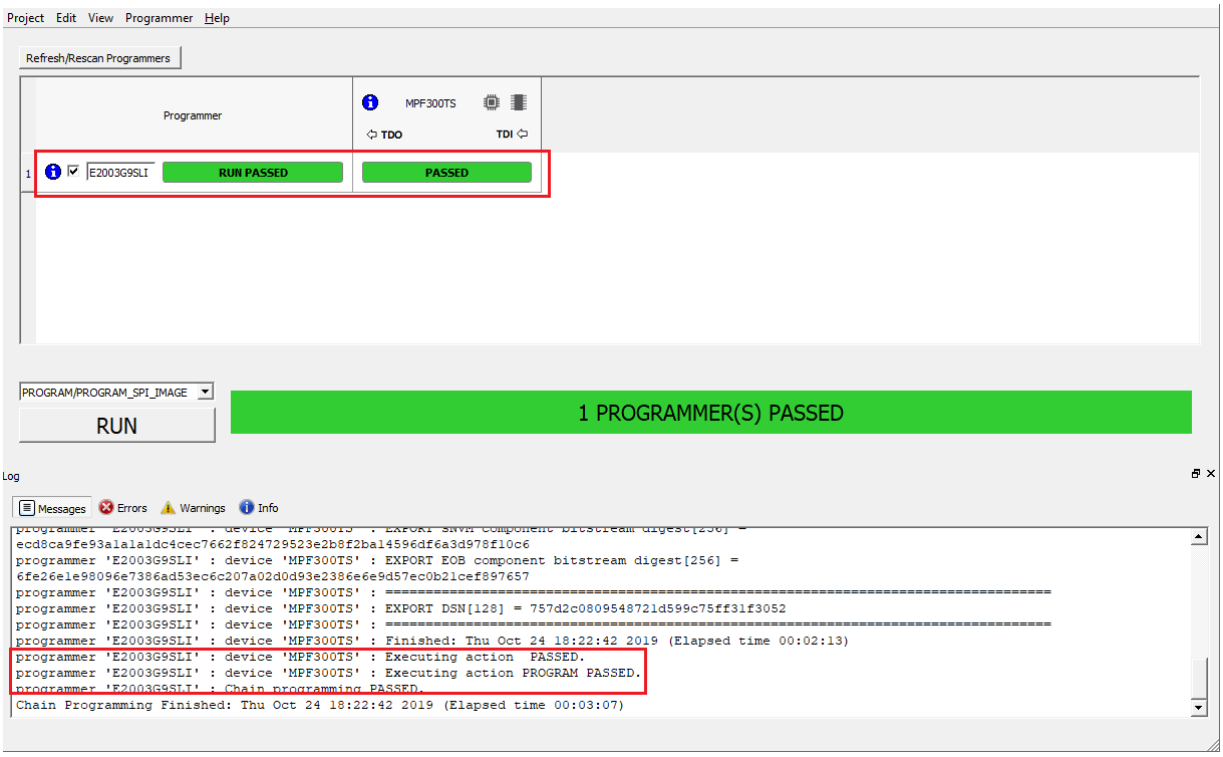
8. Click **OK**. The required programming file is selected and ready to be programmed in the device.
9. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan Programmers**.

Figure 62 • **Programming the Device**



10. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure. See Running the Demo, page 27 to run the webserver and TFTP_IAP demo.

Figure 63 • **FlashPro Express—RUN PASSED**



11. Close **FlashPro Express** or in the **Project** tab, click **Exit**.