AC490
Application Note
RTG4 FPGA: Building a Mi-V Processor Subsystem

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.

51900490. 3.0 9/21

Contents

Figures

Tables

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 3.0

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

- Updated the document for Libero SoC v2021.2.
- Updated Figure 1, page 3 through Figure 3, page 5.
- Replaced Figure 4, page 5, Figure 5, page 7, and Figure 18, page 17.
- Updated Table 2, page 6 and Table 3, page 7.
- Added Appendix 1: Programming the Device Using FlashPro Express, page 14.
- Added Appendix 3: Running the TCL Script, page 20.
- Removed the references to Libero version numbers.

1.2 Revision 2.0

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

- Added information about the COM port selection in Setting Up the Hardware, page 9.
- Updated how to select the appropriate COM port in Running the Demo, page 11.

1.3 Revision 1.0

The first publication of the document.

2 Building a Mi-V Processor Subsystem

Microchip offers the Mi-V processor IP, a 32-bit RISC-V processor and software toolchain to develop RISC-V processor based designs. RISC-V, a standard open Instruction Set Architecture (ISA) under the governance of the RISC-V Foundation, offers numerous benefits, which include enabling the open source community to test and improve cores at a faster pace than closed ISAs.
RTG4® FPGAs support Mi-V soft processor to run user applications. This application note describes how to build a Mi-V processor subsystem to execute a user application from the designated fabric RAMs or DDR memory.

2.1 Design Requirements

The following table lists the hardware and software requirements for running the demo.

Table 1 • **Design Requirements**

Requirement	Version
Hardware	
RTG4 Development Kit • RTG4 Development Board with RT4G150 - 1CG 1657PROTO FPGA • 12V, 5A AC power adapter • USB A to mini-B cable	Rev B
Host PC or Laptop	64-bit Windows 7 and 10
Software	
Libero® System-on-Chip (SoC)	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	
SoftConsole	

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.

2.2 Prerequisites

Before you start:

1. 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*
2. For demo design files download link:

   http://soc.microsemi.com/download/rsc/?f=rtg4_ac490_df

2.3 Design Description

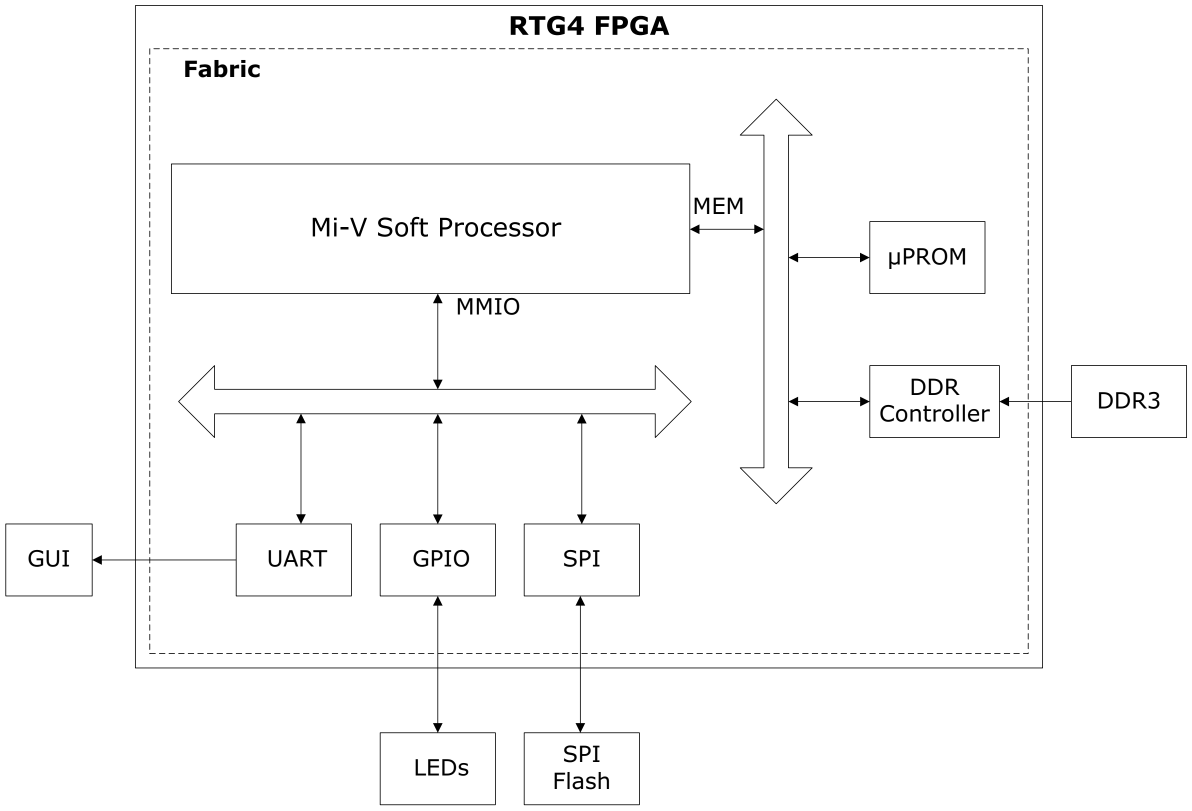
The size of RTG4 µPROM is 57 KB. User applications that do not exceed the µPROM size can be stored in µPROM and executed from internal Large SRAM memories (LSRAM). User applications that exceed the µPROM size must be stored in an external non-volatile memory. In this case, a bootloader executing from µPROM is required to initialize internal or external SRAM memories with the target application from the non-volatile memory.

The reference design demonstrates the bootloader capability to copy the target application (of size 7 KB) from SPI flash to DDR memory, and execute from the DDR memory. The bootloader is executed from internal memories. The code section is located in µPROM, and the data section is located in the internal Large SRAM (LSRAM).

Note: For more information about how to build the Mi-V bootloader Libero project and how to the build SoftConsole project, refer to *TU0775: PolarFire FPGA: Building a Mi-V Processor Subsystem Tutorial*

Figure 1 shows the top-level block diagram of the design.

Figure 1 • **Top Level Block Diagram**



As shown in Figure 1, the following points describe the data flow of the design:

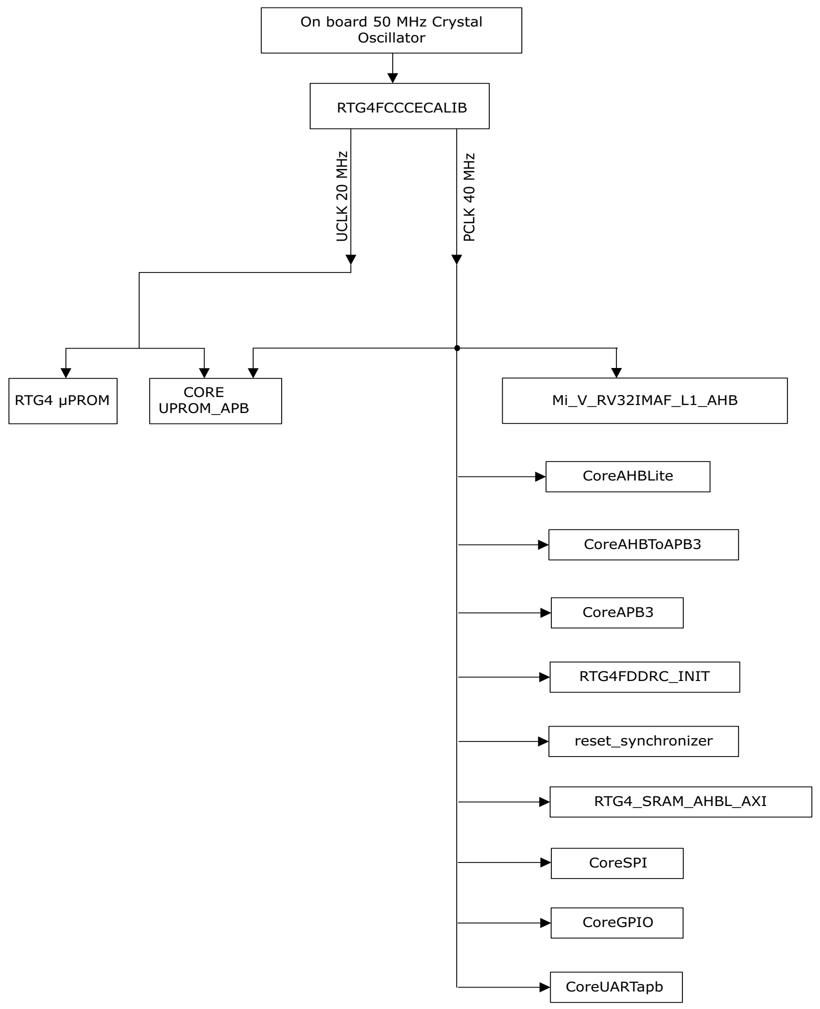
- The Mi-V processor executes the bootloader from the µPROM and designated LSRAMs. The bootloader interfaces with the GUI through the CoreUARTapb block and waits for the commands.
- When the SPI flash program command is received from the GUI, the bootloader programs the SPI flash with the target application received from the GUI.
- When the boot command is received from the GUI, the bootloader copies the application code from the SPI flash to DDR and then executes it from DDR.

2.4 Clocking Structure

There are two clock domains (40 MHz and 20 MHz) in the design. The on-board 50 MHz crystal oscillator is connected to the PF_CCC block which generates 40 MHz and 20 MHz clocks. The 40 MHz system clock drives the complete Mi-V processor subsystem except μPROM. The 20 MHz clock drives the RTG4 μPROM and the RTG4 μPROM APB interface. RTG4 μPROM supports a clock frequency of up to 30 MHz. DDR_FIC is configured for the AHB bus interface, which operates at 40 MHz. The DDR memory operates at 320 MHz.

Figure 2 shows the clocking structure.

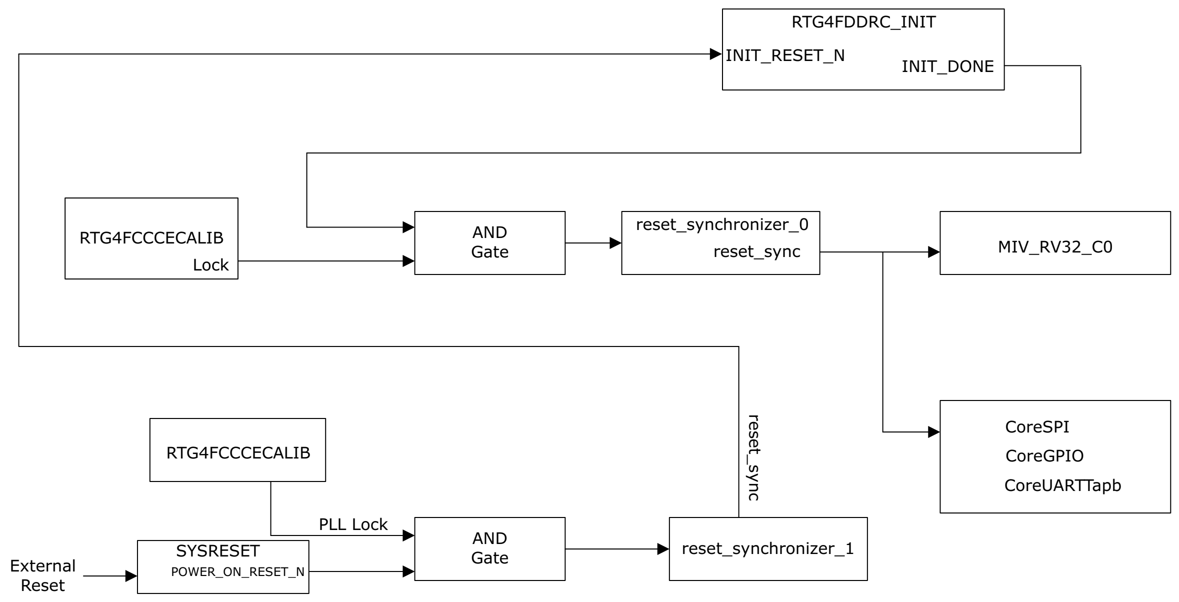
Figure 2 • **Clocking Structure**



2.5 Reset Structure

The POWER_ON_RESET_N and the LOCK signals are ANDed, and the output signal (INIT_RESET_N) is used to reset the RTG4FDDRC_INIT block. After releasing the FDDR reset, the FDDR controller gets initialized, and then the INIT_DONE signal is asserted. The INIT_DONE signal is used to reset the Mi-V processor, peripherals, and other blocks in the design.

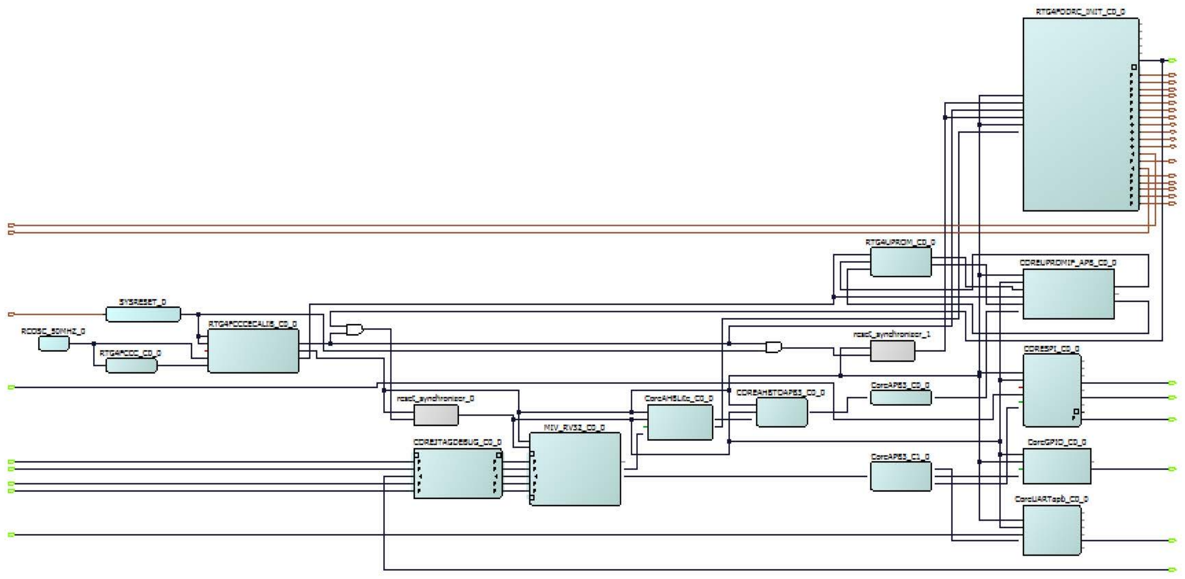
Figure 3 • **Reset Structure**



2.6 Hardware Implementation

Figure 4 shows the Libero design of the Mi-V reference design.

Figure 4 • **SmartDesign Module**



Note: Libero SmartDesign screenshot shown in this application note is for illustration purpose only. Open the Libero project to see the latest updates and IP versions.

2.6.1 IP Blocks

Figure 2 list the IP blocks used in the Mi-V processor subsystem reference design and their function.

Table 2 • **IP Blocks**[1]

IP Name	Function
MIV_RV32_C0	Mi-V soft processor.
CoreJTAGDEBUG	Facilitates the connection of Joint Test Action Group (JTAG) compatible soft core processors to the JTAG header for debugging. It provides fabric access to the JTAG interface using UJTAG macro.
CoreAHBLite	Multi-master AHB-Lite bus.
CoreAHBtoAPB3	Bridge between AHB master and APB slave.
CoreUARTapb, CoreSPI, and CoreGPIO	UART, SPI, and GPIO controllers with APB interface.
RTG4FCCCECALIB	Macro to access RTG4 CCC block. It is used to synthesize 55.5 MHz and 27.7 MHz clock frequencies from the CCC with an on-board 50 MHz reference clock.
RTG4UPROM	Used for storing the bootloader program.
RTG4UPROMIF_APB	CoreUPROMIF_APB is an APB wrapper core that provides read-only access to the µPROM memory block within the RTG4 fabric via the APB interface. This facilitates easy access to the µPROM for APB masters.
RTG4FDDRC_INIT	The Fabric External Memory DDR (FDDR) Configurator is used to configure the external DDR memory parameters.
CoreAPB3	CoreAPB3 is an advanced microcontroller bus architecture (AMBA) 3 advanced peripheral bus (APB) fabric for interconnecting between an APB master and up to 16 APB slaves.

1. All the IP user guides and handbooks are available from Libero SoC -> Catalog.

RTG4 µPROM stores up to 10,400 36-bit words (374,400 bits of data). It supports only read operations during normal device operation after the device is programmed. The MIV_RV32_C0 processor core comprises an instruction fetch unit, an execution pipeline, and a data memory system. The MIV_RV32_C0 processor memory system includes instruction cache and data cache. The MIV_RV32_C0 core includes two external AHB interfaces-the AHB memory (MEM) bus master interface and the AHB Memory Mapped I/O (MMIO) bus master interface. The cache controller uses the AHB MEM interface to refill the instructions and the data caches. The AHB MMIO interface is used for an un-cached access to I/O peripherals.

The memory maps of the AHB MMIO interface and the MEM interface are 0x60000000 to 0X6FFFFFFF and 0x80000000 to 0x8FFFFFFF, respectively. The processor's reset vector address is configurable. The MIV_RV32_C0's reset is an active-low signal, which must be de-asserted in sync with the system clock through a reset synchronizer.

The MIV_RV32_C0 processor accesses the application execution memory using the AHB MEM interface. The CoreAHBLite_C0_0 bus instance is configured to provide 16 slave slots, each of size 1 MB. The RTG µPROM memory, and RTG4FDDRC blocks are connected to this bus. The µPROM is used for storing the bootloader application.

The MIV_RV32_C0 processor directs the data transactions between addresses 0x60000000 and 0x6FFFFFFF to the MMIO interface. The MMIO interface is connected to the CoreAHBLite_C1_0 bus to communicate with peripherals connected to its slave slots. The CoreAHBLite_C1_0 bus instance is configured to provide 16 slave slots, each of size 256 MB. The UART, CoreSPI, and CoreGPIO peripherals are connected to the CoreAHBLite_C1_0 bus via the CoreAHBTOAPB3 bridge and the CoreAPB3 bus.

2.6.2 Memory Map

Table 3 lists the memory map of the memories and peripherals.

Table 3 • **Memory Map**

Peripherals	Start Address
TCM	0x70000000
µPROM	0x80100000
DDR3	0x80200000
UART	0x60000000
GPIO	0x60001000
SPI	0x60002000

2.7 Software Implementation

The reference design files include the SoftConsole workspace that contains the following software projects:

- Bootloader
- Target Application

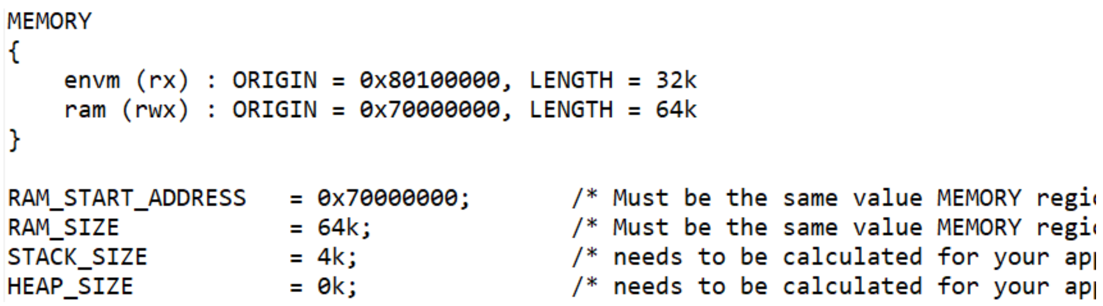
2.7.1 Bootloader

The bootloader application is programmed on the µPROM during device programming. The bootloader implements the following functions:

- Programming the SPI Flash with the target application.
- Copying the target application from SPI Flash to DDR3 memory.
- Switching the program execution to the target application available in DDR3 memory.

The bootloader application must be executed from µPROM with LSRAM as stack. Hence, the addresses of ROM and RAM in the linker script are set to the starting address of µPROM and designated LSRAMs, respectively. The code section is executed from ROM and data section is executed from RAM as shown in Figure 5.

Figure 5 • **Bootloader Linker Script**



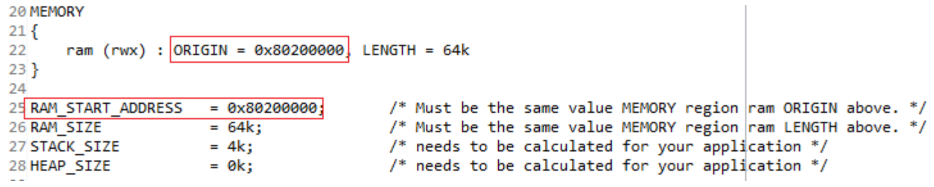
```
MEMORY
{
    envm (rx) : ORIGIN = 0x80100000, LENGTH = 32k
    ram (rwx) : ORIGIN = 0x70000000, LENGTH = 64k
}

RAM_START_ADDRESS   = 0x70000000;       /* Must be the same value MEMORY regi
RAM_SIZE            = 64k;              /* Must be the same value MEMORY regi
STACK_SIZE          = 4k;               /* needs to be calculated for your ap
HEAP_SIZE           = 0k;               /* needs to be calculated for your ap
```

The linker script (microsemi-riscv-ram_rom.ld) is available at the `SoftConsole_Project\mivrv32im-bootloader` folder of the design files.

2.7.2 Target Application

The target application blinks the onboard LEDs 1, 2, 3, and 4 and prints UART messages. The target application must be executed from DDR3 memory. Hence, the code and stack sections in the linker script are set to the starting address of DDR3 memory as shown in Figure 6.

Figure 6 • **Target Application Linker Script**



```
MEMORY
{
    ram (rwx) : ORIGIN = 0x80200000, LENGTH = 64k
}

RAM_START_ADDRESS    = 0x80200000;        /* Must be the same value MEMORY region ram ORIGIN above. */
RAM_SIZE             = 64k;               /* Must be the same value MEMORY region ram LENGTH above. */
STACK_SIZE           = 4k;                /* needs to be calculated for your application */
HEAP_SIZE            = 0k;                /* needs to be calculated for your application */
```

The linker script (microsemi-riscv-ram.ld) is available at the `SoftConsole_Project\miv-rv32im-ddr-application` folder of the design files.

3 Setting Up the Hardware

The following steps describe how to set up the hardware:

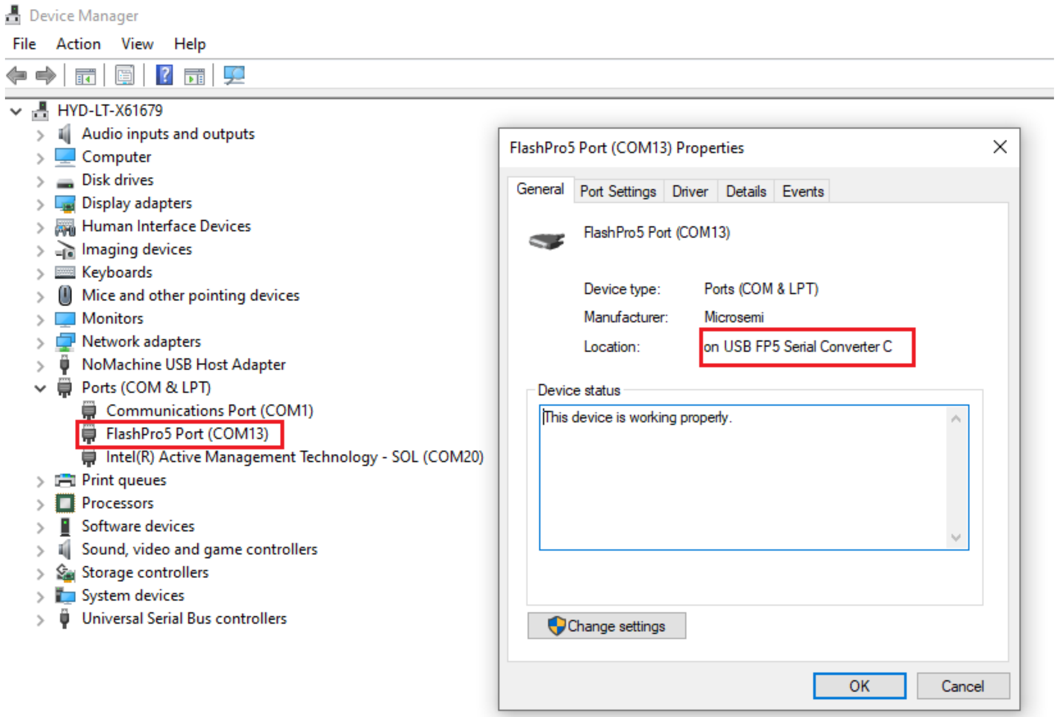
1. Ensure that the board is powered OFF using the **SW6** switch.
2. Connect the jumpers on the RTG4 development kit, as shown in the following table:

Table 4 • **Jumpers**

Jumper	Pin From	Pin To	Comments
J11, J17, J19, J23, J26, J21, J32, and J27	1	2	Default
J16	2	3	Default
J33	1 3	2 4	Default

3. Connect the host PC to the **J47** connector using the USB cable.
4. Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the device manager of the host PC.
5. As shown in Figure 7, the port properties of COM13 show that it is connected to USB Serial Converter C. Hence, COM13 is selected in this example. The COM port number is system specific.

Figure 7 • **Device Manager**

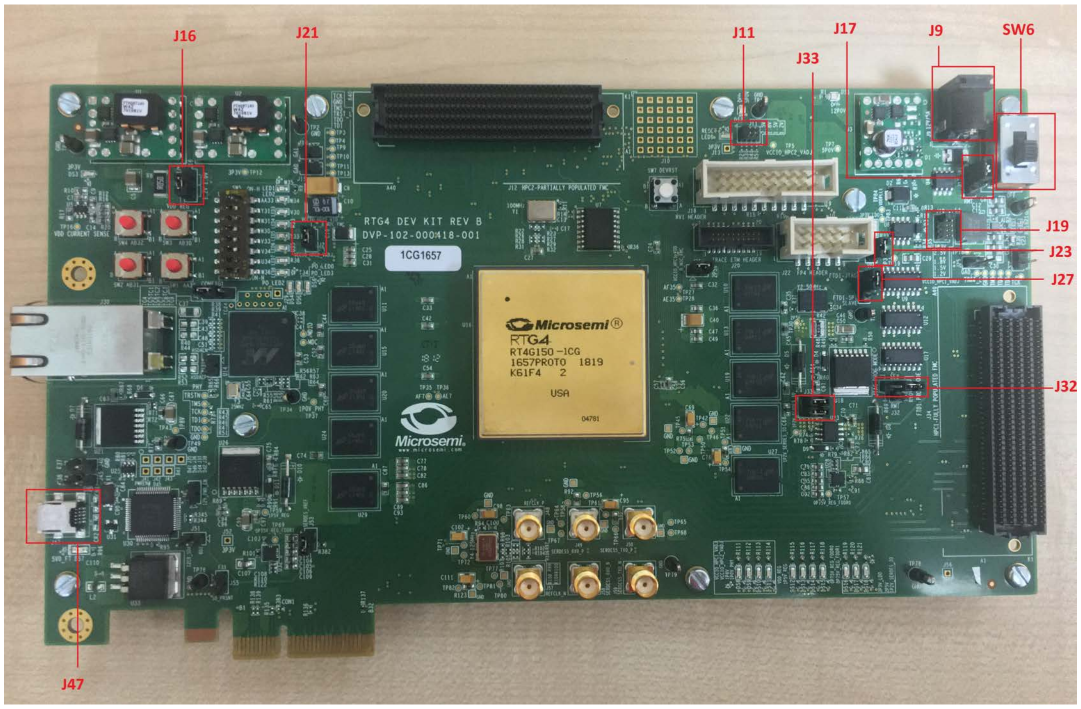


Device type: Ports (COM & LPT)
Manufacturer: Microsemi
Location: on USB FP5 Serial Converter C
Device status: This device is working properly.

Note: If the USB to UART bridge drivers are not installed, download and install the drivers from www.microsemi.com//documents/CDM_2.08.24_WHQL_Certified.zip.

6. Connect the power supply to **J9** connector and switch ON the power supply switch, SW6.

Figure 8 • **RTG4 Development Kit**



4 Running the Demo

This chapter describes steps to program the RTG4 device with the reference design, programming the SPI Flash with the target application, and booting the target application from DDR memory using the Mi-V Bootloader GUI.

Running the demo involves the following steps:

1. Programming the RTG4 Device, page 11
2. Running the Mi-V Bootloader, page 11

4.1 Programming the RTG4 Device

The RTG4 device can be programmed either using FlashPro Express or Libero SOC.

- To program the RTG4 Development Kit with the job file provided as part of the design files using FlashPro Express software, refer to Appendix 1: Programming the Device Using FlashPro Express, page 14.
- To program the device using Libero SoC, refer to Appendix 2: Programming the Device Using Libero SoC, page 17.

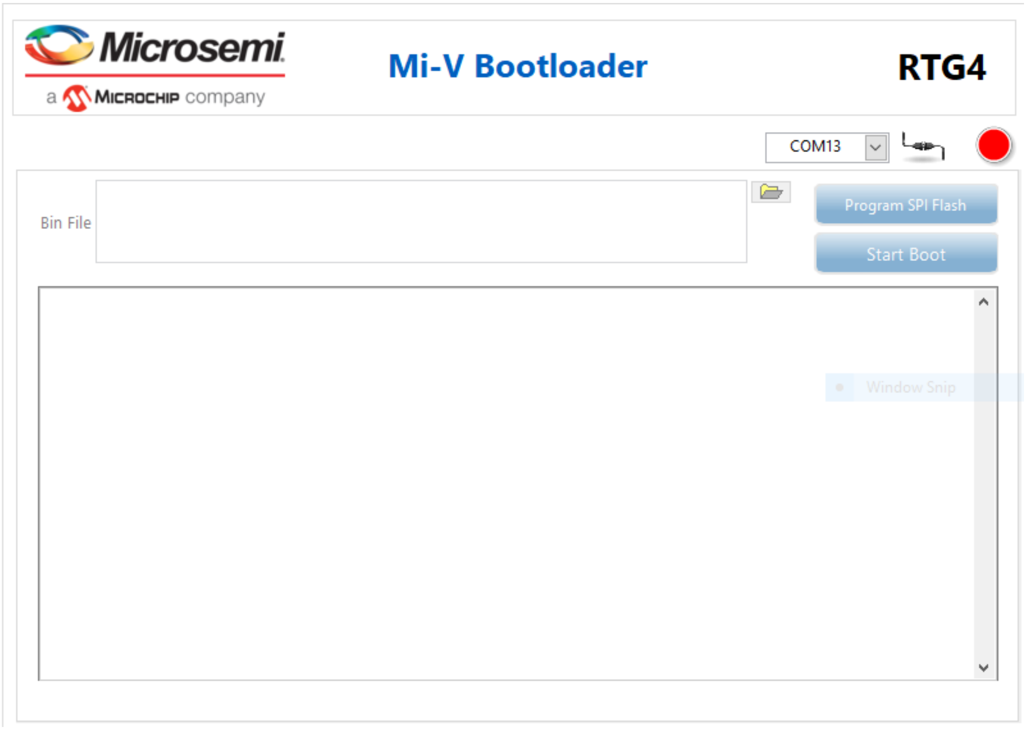
4.2 Running the Mi-V Bootloader

On successful completion of programming, follow these steps:

1. Run the `setup.exe` file available at the following design files location.
   `<$Download_Directory>\rtg4_ac490_df\GUI_Installer\Mi-V Bootloader_Installer_V1.4`
2. Follow the installation wizard to install the Bootloader GUI application.

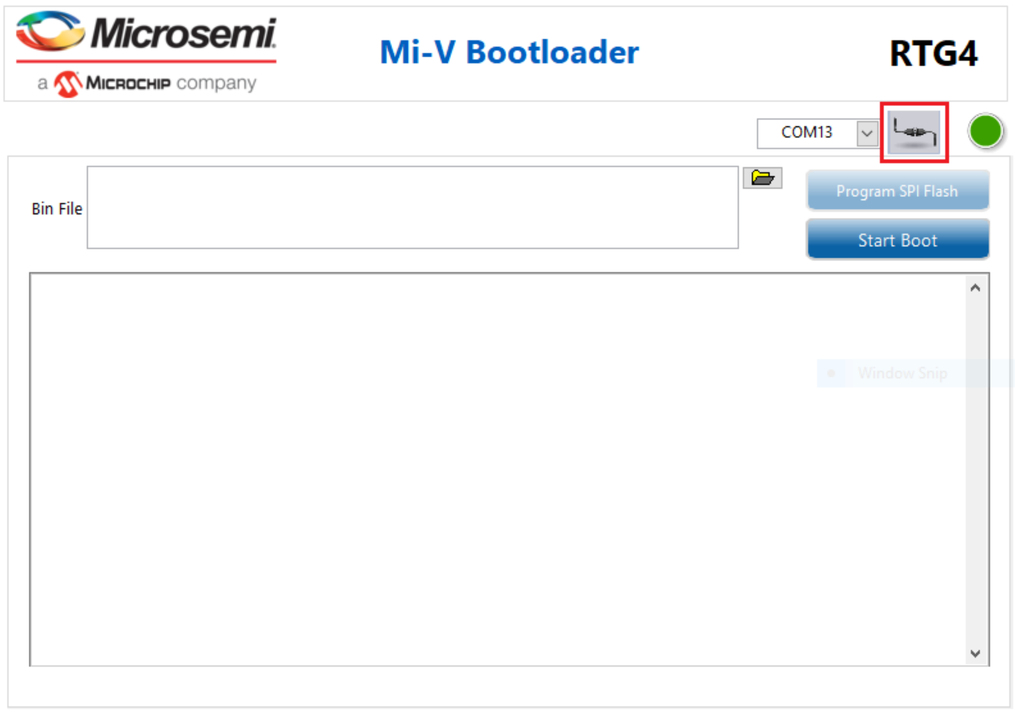
Figure 9 shows the RTG4 Mi-V Bootloader GUI.

Figure 9 • **Mi-V Bootloader GUI**



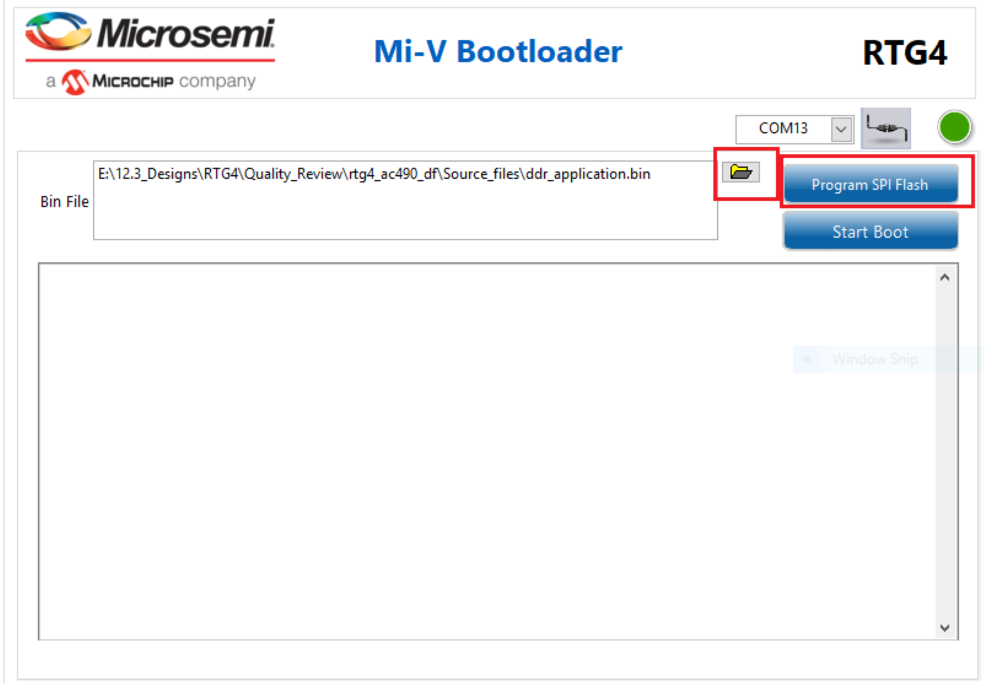
3. Select the COM port connected to USB Serial Converter C as shown in Figure 7.
4. Click the connect button. After successful connection the Red indicator turns Green as shown in Figure 10.

Figure 10 • **Connect COM Port**



5. Click the Import button and select the target application file (.bin). After importing, the path of the file is displayed on the GUI as shown in Figure 11.
`<$Download_Directory>\rtg4_ac490_df\Source_files`

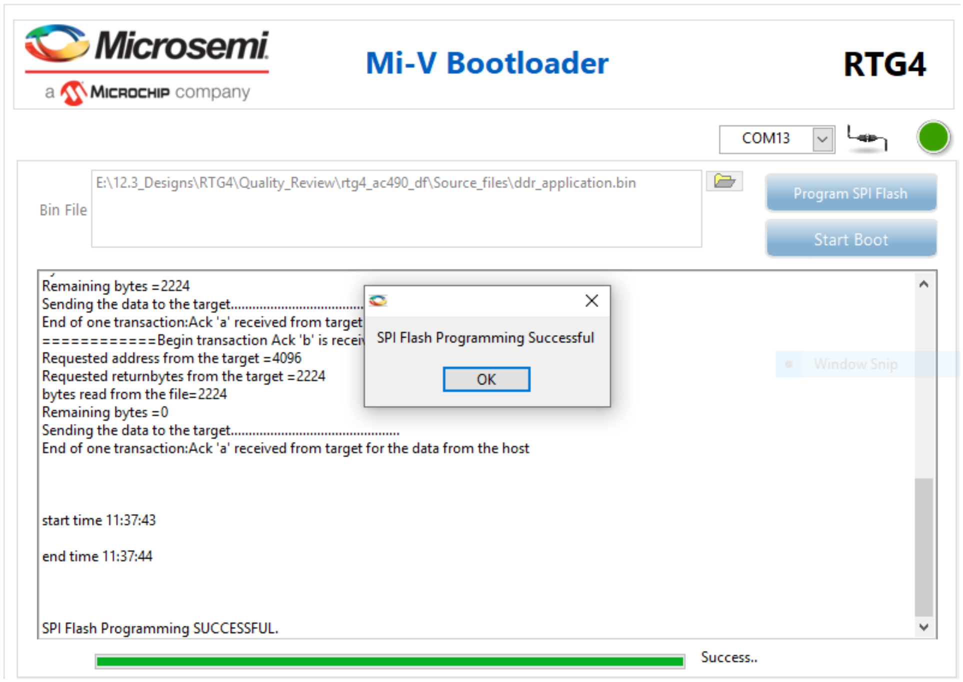
Figure 11 • **Import the Target Application File**



E:\12.3_Designs\RTG4\Quality_Review\rtg4_ac490_df\Source_files\ddr_application.bin

6. As shown in Figure 11, click **Program SPI Flash** option to program the target application on the SPI Flash. A pop-up is displayed after the SPI Flash is programmed as shown in Figure 12. Click **OK**.

Figure 12 • **SPI Flash Programmed**



E:\12.3_Designs\RTG4\Quality_Review\rtg4_ac490_df\Source_files\ddr_application.bin

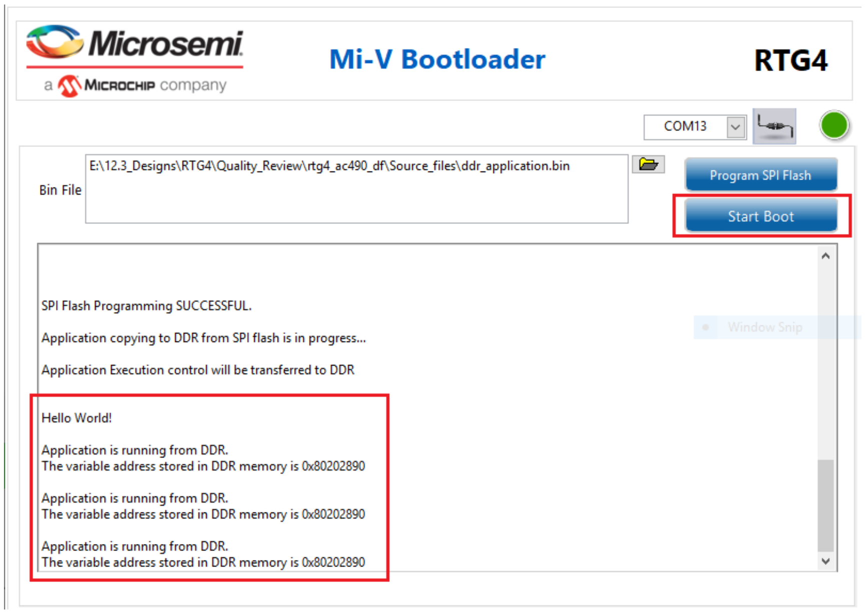
```
Remaining bytes =2224
Sending the data to the target.....................................
End of one transaction:Ack 'a' received from target
============Begin transaction Ack 'b' is recei
Requested address from the target =4096
Requested returnbytes from the target =2224
bytes read from the file=2224
Remaining bytes =0
Sending the data to the target..............................................
End of one transaction:Ack 'a' received from target for the data from the host


start time 11:37:43

end time 11:37:44


SPI Flash Programming SUCCESSFUL.
```

SPI Flash Programming Successful

Success..

7. Select the **Start Boot** option to copy the application from SPI Flash to DDR3 memory and start executing the application from DDR3 memory. After successful booting of the target application from DDR3 memory, the application prints UART messages and blinks on-board user LED1, 2, 3, and 4 as shown in Figure 13.

Figure 13 • **Execute Application From DDR**



E:\12.3_Designs\RTG4\Quality_Review\rtg4_ac490_df\Source_files\ddr_application.bin

```
SPI Flash Programming SUCCESSFUL.

Application copying to DDR from SPI flash is in progress...

Application Execution control will be transferred to DDR

Hello World!

Application is running from DDR.
The variable address stored in DDR memory is 0x80202890

Application is running from DDR.
The variable address stored in DDR memory is 0x80202890

Application is running from DDR.
The variable address stored in DDR memory is 0x80202890
```

8. The application is running from the DDR3 memory and this concludes the demo. Close the Mi-V Bootloader GUI.

5 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the RTG4 device with the programming job file using FlashPro Express.

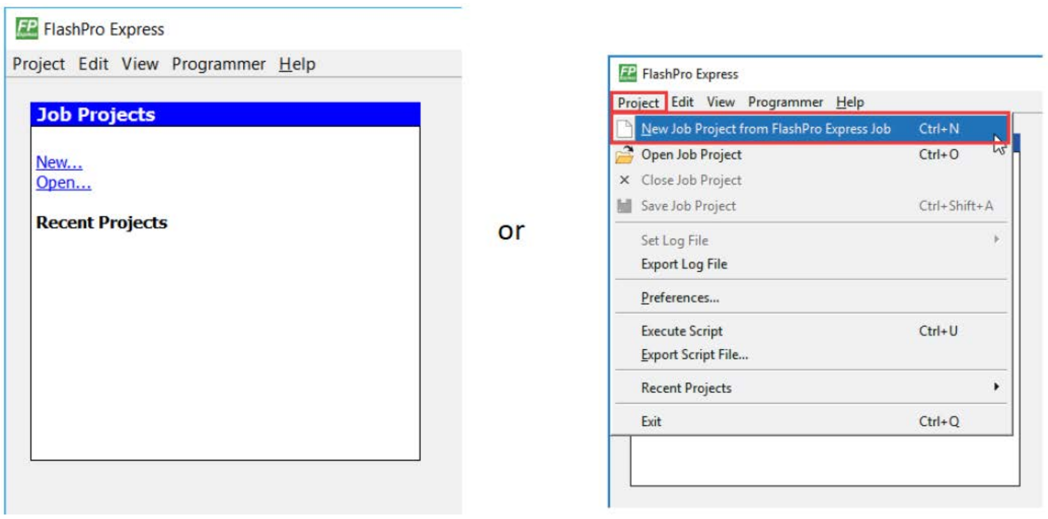
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in *Table 3 of UG0617: RTG4 Development Kit User Guide.*
2. Optionally, jumper **J32** can be set to connect pins 2-3 when using an external FlashPro4, FlashPro5, or FlashPro6 programmer instead of the default jumper setting to use the embedded FlashPro5.

Note: The power supply switch, **SW6** must be switched **OFF** while making the jumper connections.

3. Connect the power supply cable to the **J9** connector on the board.
4. Power **ON** the power supply switch **SW6**.
5. If using the embedded FlashPro5, connect the USB cable to connector **J47** and the host PC. Alternatively, if using an external programmer, connect the ribbon cable to the JTAG header **J22** and connect the programmer to the host PC.
6. On the host PC, launch the **FlashPro Express** software.
7. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

Figure 14 • **FlashPro Express Job Project**



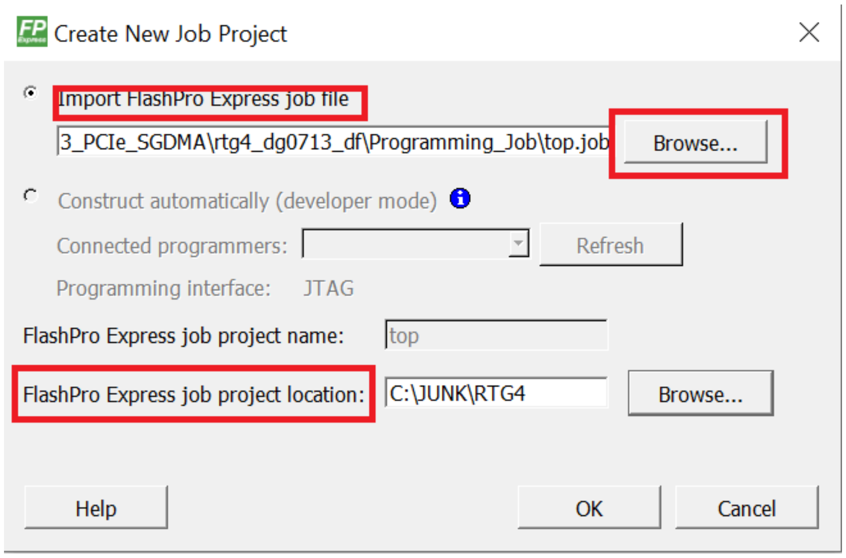
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

- **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>\rtg4_ac490_df\Programming_Job
  ```

- **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 15 • **New Job Project from FlashPro Express Job**


Create New Job Project

Import FlashPro Express job file

3_PCIe_SGDMA\rtg4_dg0713_df\Programming_Job\top.job

Construct automatically (developer mode)

Connected programmers:

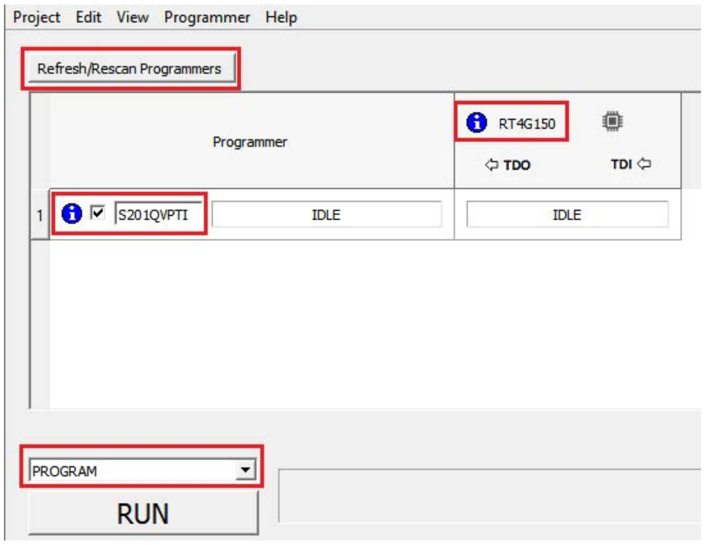
Programming interface: JTAG

FlashPro Express job project name: top

FlashPro Express job project location: C:\JUNK\RTG4

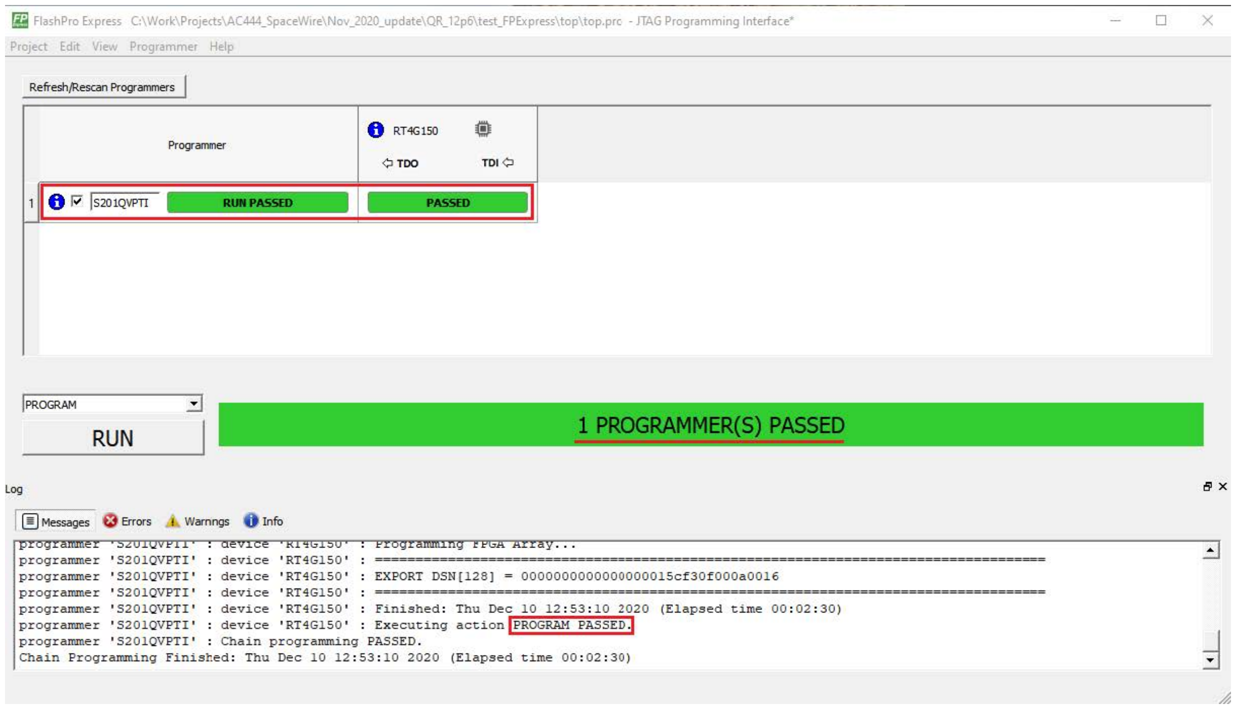
9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan**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

Figure 16 • **Programming the Device**



11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure.

Figure 17 • **FlashPro Express—RUN PASSED**


FlashPro Express C:\Work\Projects\AC444_SpaceWire\Nov_2020_update\QR_12p6\test_FPExpress\top\top.pro - JTAG Programming Interface*
Project Edit View Programmer Help
Refresh/Rescan Programmers
Programmer
RT4G150
TDO
TDI
1
S201QVPTI
RUN PASSED
PASSED
PROGRAM
RUN
1 PROGRAMMER(S) PASSED
Log
Messages
Errors
Warnngs
Info

```
programmer 'S201QVPTI' : device 'RT4G150' : Programming FPGA Array...
programmer 'S201QVPTI' : device 'RT4G150' : ==============================================================================
programmer 'S201QVPTI' : device 'RT4G150' : EXPORT DSN[128] = 00000000000000000015cf30f000a0016
programmer 'S201QVPTI' : device 'RT4G150' : ==============================================================================
programmer 'S201QVPTI' : device 'RT4G150' : Finished: Thu Dec 10 12:53:10 2020 (Elapsed time 00:02:30)
programmer 'S201QVPTI' : device 'RT4G150' : Executing action PROGRAM PASSED.
programmer 'S201QVPTI' : Chain programming PASSED.
Chain Programming Finished: Thu Dec 10 12:53:10 2020 (Elapsed time 00:02:30)
```

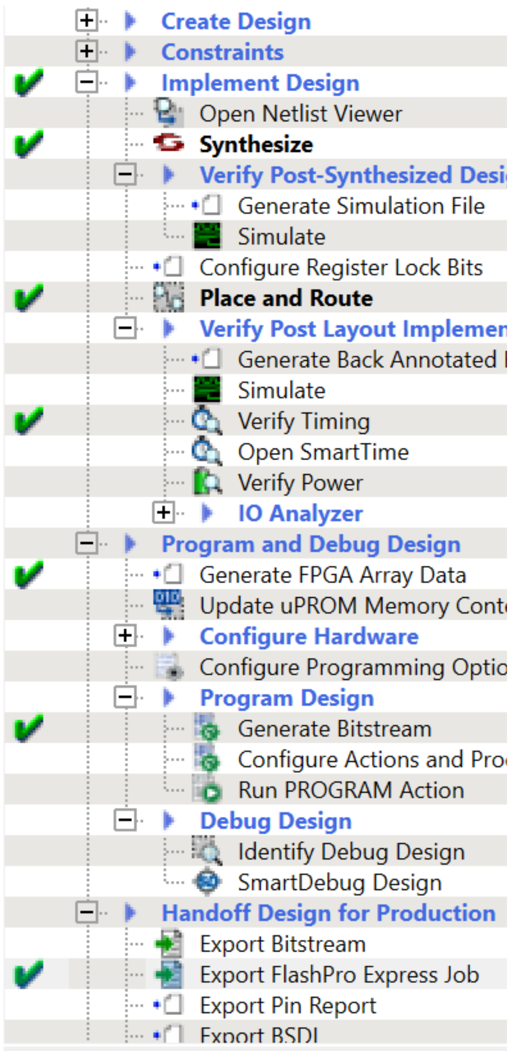
12. Close **FlashPro Express** or click **Exit** in the Project tab.

6 Appendix 2: Programming the Device Using Libero SoC

The reference design files include the Mi-V processor subsystem project created using Libero SoC. The RTG4 device can be programmed using Libero SoC. The Libero SoC project is completely built and run from Synthesis, Place and Route, Timing Verification, FPGA Array Data Generation, Update µPROM Memory Content, Bitstream Generation, FPGA Programming.

The Libero design flow is shown in the following figure.

Figure 18 • **Libero Design Flow**



- Create Design
- Constraints
- ✓ Implement Design
  - Open Netlist Viewer
  - ✓ **Synthesize**
  - Verify Post-Synthesized Desi
    - Generate Simulation File
    - Simulate
  - Configure Register Lock Bits
  - ✓ **Place and Route**
  - Verify Post Layout Implemen
    - Generate Back Annotated I
    - Simulate
    - ✓ Verify Timing
    - Open SmartTime
    - Verify Power
    - IO Analyzer
- Program and Debug Design
  - ✓ Generate FPGA Array Data
  - Update uPROM Memory Cont
  - Configure Hardware
  - Configure Programming Optio
  - Program Design
    - ✓ Generate Bitstream
    - Configure Actions and Pro
    - Run PROGRAM Action
  - Debug Design
    - Identify Debug Design
    - SmartDebug Design
- Handoff Design for Production
  - Export Bitstream
  - ✓ Export FlashPro Express Job
  - Export Pin Report
  - Export BSDL

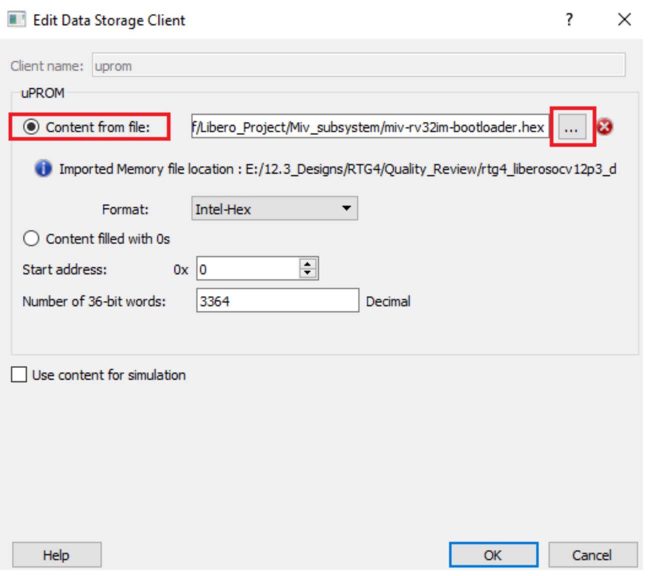
To program the RTG4 device, the Mi-V processor subsystem project must be opened in Libero SoC and the following steps must be re-run:

1. **Update uPROM Memory Content**: In this step, µPROM is programmed with the bootloader application.
2. **Bitstream Generation**: In this step, the Job file is generated for the RTG4 device.
3. **FPGA Programming**: In this step, the RTG4 device is programmed using the Job file.

Follow these steps:

1. From Libero Design Flow, select **Update uPROM Memory Content**.
2. Create a client using the Add option.
3. Select the client and then choose the Edit option.
4. Select Content from file and then select the Browse option as shown in Figure 19.

Figure 19 • **Edit Data Storage Client**



Edit Data Storage Client

Client name: uprom

uPROM

(●) Content from file: f/Libero_Project/Miv_subsystem/miv-rv32im-bootloader.hex

Imported Memory file location : E:/12.3_Designs/RTG4/Quality_Review/rtg4_liberosocv12p3_d

Format: Intel-Hex

( ) Content filled with 0s

Start address: 0x 0

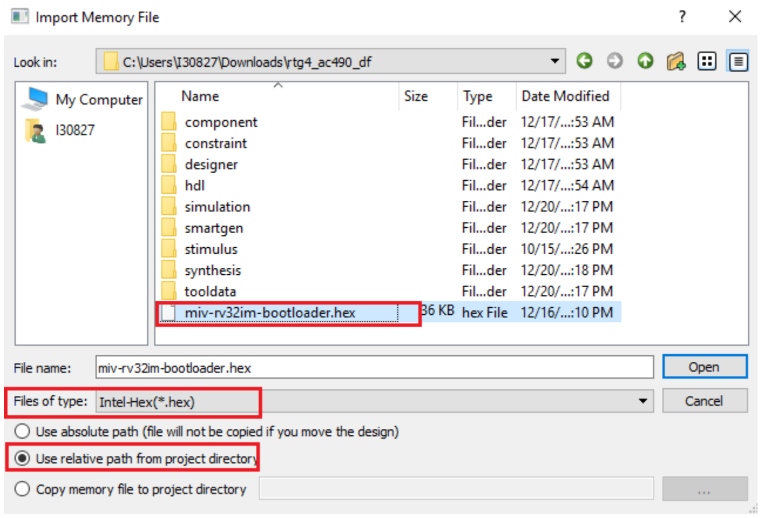
Number of 36-bit words: 3364 Decimal

[ ] Use content for simulation

5. Navigate to the following design files location and select the `miv-rv32im-bootloader.hex` file as shown in Figure 20.
   `<$Download_Directory>\rtg4_ac490_df`

- Set the File Type as **Intel-Hex (*.hex)**.
- Select **Use relative path from project directory**.
- Click **OK**.

Figure 20 • **Import Memory File**



Import Memory File

Look in: C:\Users\I30827\Downloads\rtg4_ac490_df

Name	Size	Type	Date Modified
component		Fil...der	12/17/...:53 AM
constraint		Fil...der	12/17/...:53 AM
designer		Fil...der	12/17/...:53 AM
hdl		Fil...der	12/17/...:54 AM
simulation		Fil...der	12/20/...:17 PM
smartgen		Fil...der	12/20/...:17 PM
stimulus		Fil...der	10/15/...:26 PM
synthesis		Fil...der	12/20/...:18 PM
tooldata		Fil...der	12/20/...:17 PM
miv-rv32im-bootloader.hex	36 KB	hex File	12/16/...:10 PM

File name: miv-rv32im-bootloader.hex

Files of type: Intel-Hex(*.hex)

( ) Use absolute path (file will not be copied if you move the design)

(●) Use relative path from project directory

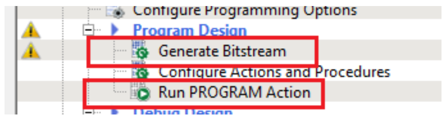
( ) Copy memory file to project directory

6. Click **OK**.

The µPROM content is updated.

7. Double-click **Generate Bitstream** as shown in Figure 21.

Figure 21 • **Generate Bitstream**



8. Double-click **Run PROGRAM Action** to program the device as shown in Figure 21.

The RTG4 device is programmed. See Running the Demo, page 11.

7 Appendix 3: Running the TCL Script

TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To run the TCL, follow the steps below:

1. Launch the Libero software.
2. Select **Project > Execute Script....**
3. Click Browse and select `script.tcl` from the downloaded TCL_Scripts directory.
4. Click **Run**.

After successful execution of TCL script, Libero project is created within TCL_Scripts directory.

For more information about TCL scripts, refer to **rtg4_ac490_df/TCL_Scripts/readme.txt.**

Refer to *Libero® SoC TCL Command Reference Guide* for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.