DG0703
Demo Guide
Error Detection and Correction on RTG4 LSRAM Memory

Microsemi
a Microchip company

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.

50200703. 4.0 8/21

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 4.0

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

- Updated the document for Libero SoC v2021.2.
- Added Appendix 1: Programming the Device Using FlashPro Express, page 14.
- Added Appendix 2: Running the TCL Script, page 16.
- Removed the references to Libero version numbers.

1.2 Revision 3.0

Updated the document for Libero v11.9 SP1 software release.

1.3 Revision 2.0

Updated the document for Libero v11.8 SP2 software release.

1.4 Revision 1.0

The first publication of this document.

2 Error Detection and Correction on RTG4 LSRAM Memory

This reference design describes the error detection and correction (EDAC) capabilities of the RTG4™ FPGA LSRAMs. In a single event upset (SEU) susceptible environment, RAM is prone to transient errors caused by heavy ions. These errors can be detected and corrected by employing error correction codes (ECCs). The RTG4 FPGA RAM blocks have built-in EDAC controllers to generate the error correction codes for correcting a 1-bit error or detecting a 2-bit error.

If a 1-bit error is detected, the EDAC controller corrects the error bit and sets the error correction flag (SB_CORRECT) to active high. If a 2-bit error is detected, the EDAC controller sets the error detection flag (DB_DETECT) to active high.

For more information about RTG4 LSRAM EDAC functionality, refer to *UG0574: RTG4 FPGA Fabric User Guide*.

In this reference design, the 1-bit error or 2-bit error is introduced through SmartDebug GUI. EDAC is observed using a graphical user interface (GUI), utilizing the UART interface to access the LSRAM for data reads/writes, Libero® System-on-Chip (SoC) SmartDebug (JTAG) is used to inject the errors into LSRAM memory.

2.1 Design Requirements

Table 1 lists the reference design requirements for running the RTG4 LSRAM EDAC demo.

Table 1 • **Design Requirements**

Requirement	Version
Hardware	
RTG4 Development Kit: • USB 2.0 cable • 12 V, 5A AC power adapter and cords	Rev B kit with RTG4150-CB1657PROTO FPGA
Host PC or laptop	64-bit Windows 7 and 10
Software	
Libero SoC	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	
SmartDebug	
Host PC drivers	USB to UART drivers

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:

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*

2.3 Demo Design

Download the demo design files from the Microsemi website at:

http://soc.microsemi.com/download/rsc/?f=rtg4_dg0703_df

The demo design files include:

- Libero SoC project
- GUI Installer
- Programming files
- Readme.txt file
- TCL_Scripts

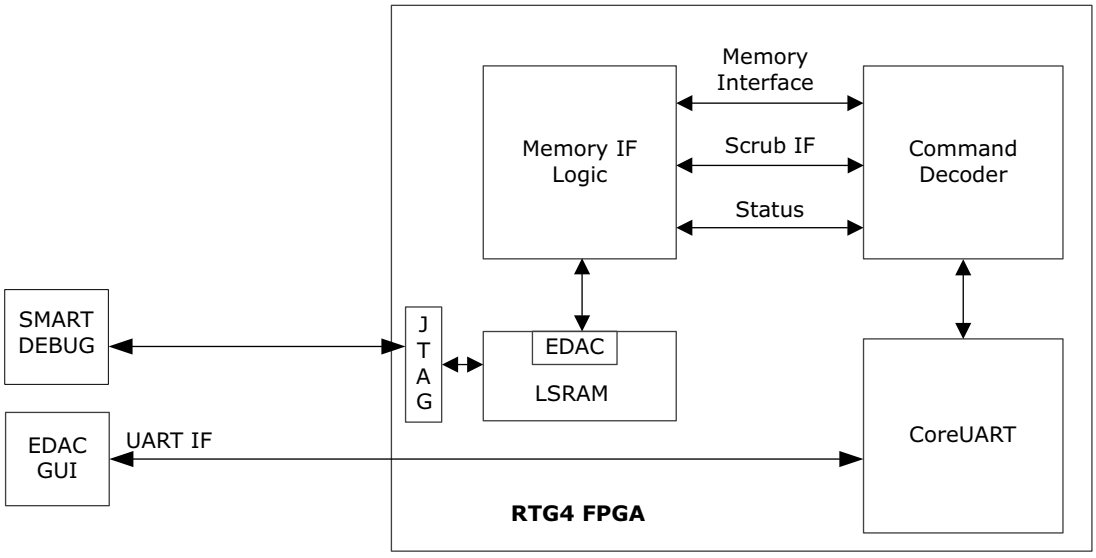
The GUI application on the host PC issues commands to the RTG4 device through the USB-UART interface. This UART interface is designed with CoreUART, which is a logic IP from the Libero SoC IP catalog. The CoreUART IP in the RTG4 fabric receives commands and transmits them to the command decoder logic. The command decoder logic decodes the read or write command, which is executed using the memory interface logic.

The memory interface block is used to read/write and monitor the LSRAM error flags. The built-in EDAC corrects the 1-bit error while reading from LSRAM and provides corrected data to the user interface but does not write corrected data back to LSRAM. The built-in LSRAM EDAC does not implement a scrubbing feature. The demo design implements scrub logic, which monitors the 1-bit correction flag and updates the LSRAM with the corrected data if a single bit error occurs.

SmartDebug GUI is used to inject 1-bit or 2-bit error into the LSRAM data.

Figure 1 shows the top-level block diagram of RTG4 LSRAM EDAC demo design.

Figure 1 • **Top-Level Block Diagram**



The following are the demo design configurations:

1. The LSRAM is configured for ×18 mode and EDAC is enabled by connecting LSRAMs ECC_EN signal to high.

Note: The LSRAM EDAC is supported for only ×18 and ×36 modes.

2. The CoreUART IP is configured to communicate with the host PC application at a 115200 baud rate.
3. The RTG4FCCCECALIB_C0 is configured to clock the CoreUART and other fabric logic at 80 MHz.

2.3.1 Features

The following are the demo design features:

- Read and write to LSRAM
- Inject 1-bit and 2-bit error using SmartDebug
- Display 1-bit and 2-bit error count values
- Provision to clear the error count values
- Enable or disable the memory scrubbing logic

2.3.2 Description

This demo design involves the implementation of the following tasks:

- **Initializing and accessing LSRAM**
  The memory interface logic implemented in the fabric logic receives the initialization command from GUI and initializes the first 256 memory locations of LSRAM with the incremental data. It also performs the read and write operations to the 256 memory locations of LSRAM by receiving the address and data from the GUI. For a read operation, the design fetches the data from LSRAM and provides it to GUI for display. The expectation is that the design will not induce errors before using SmartDebug.

Note: Uninitialized memory locations may have random values, and SmartDebug may show single-bit or double-bit errors in those locations.

- **Injecting 1-bit or 2-bit errors**
  SmartDebug GUI is used to inject the 1 bit or 2-bit errors into the specified memory location of LSRAM. The following operations are performed using SmartDebug to inject 1-bit and 2-bit errors to LSRAM:

  a) Open SmartDebug GUI, click **Debug FPGA Array**.

  b) Go to the **Memory Blocks** tab, select the memory instance, and right-click **Add**.

  c) To read the memory block, click **Read Block**.

  d) Inject single-bit or double-bit error into any location of the LSRAM of a certain depth.

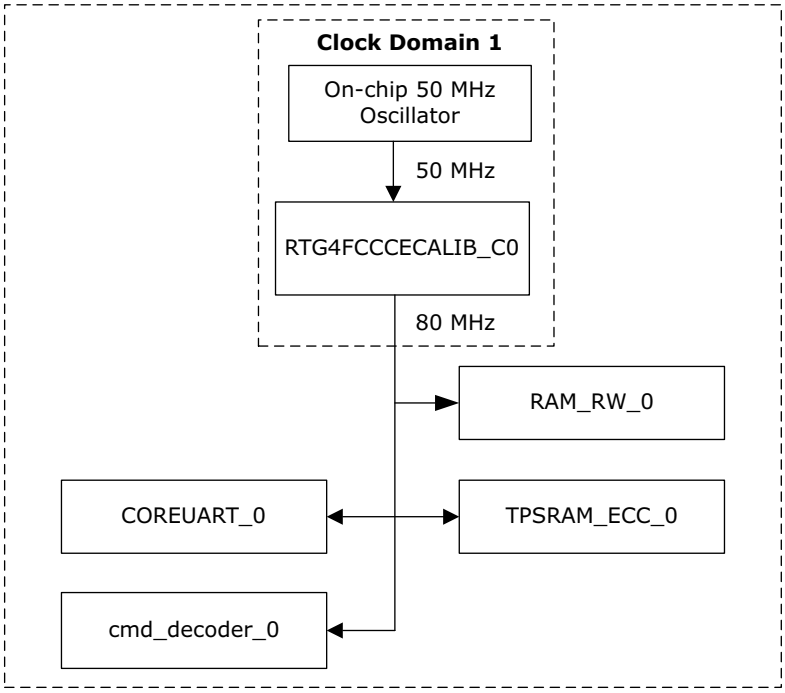
  e) To write to the modified location, click **Write Block**.

  During the LSRAM read and write operation through the SmartDebug (JTAG) interface, the EDAC controller is bypassed and does not compute the ECC bits for the write operation in step e.
- **Error Counting**
  8-bit counters are used to provide an error count and is design into the fabric logic to count 1-bit or 2-bit errors. The command decoder logic provides the count values to the GUI when receiving commands from the GUI.

2.4 Clocking Structure

In this demo design, there is one clock domain. The internal 50 MHz oscillator drives the RTG4FCCC, which further drives RTG4FCCCECALIB_C0. The RTG4FCCCECALIB_C0 generates an 80 MHz clock that provides a clock source to the COREUART, cmd_decoder, TPSRAM_ECC, and RAM_RW modules. The following figure shows the clocking structure of the demo design.

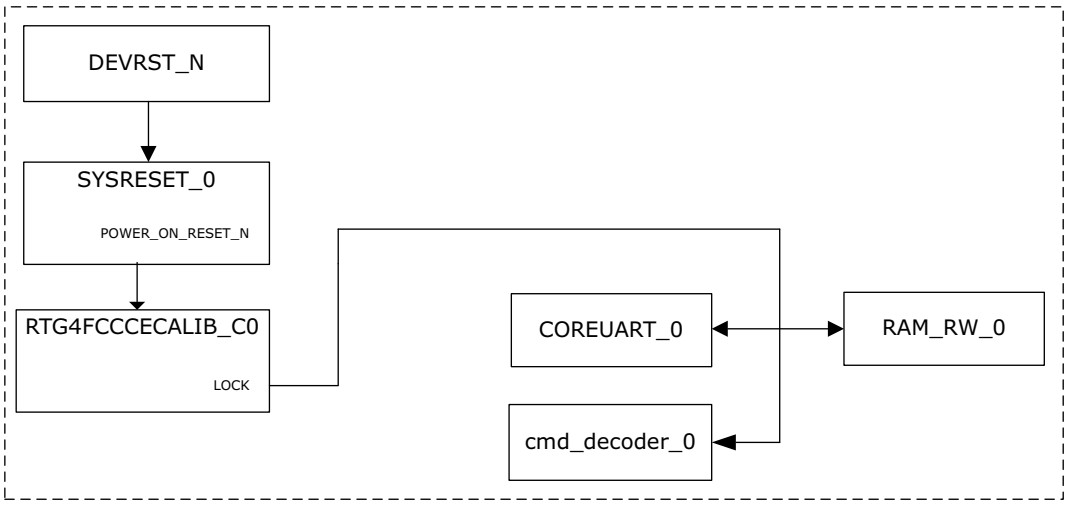
Figure 2 • **Clocking Structure**



2.5 Reset Structure

In this demo design, the reset signal to the COREUART, cmd_decoder, and RAM_RW modules are provided through the LOCK port of RTG4FCCCECALIB_C0. The following figure shows the reset structure of the demo design.

Figure 3 • **Reset Structure**



2.6 Setting Up the Demo Design

The following sections describe how to set up the RTG4 Development Kit and GUI to run the demo design.

2.6.1 Jumper Settings

1. Connect the jumpers on the RTG4 Development Kit, as shown in Table 2.

Table 2 • **Jumper Settings**

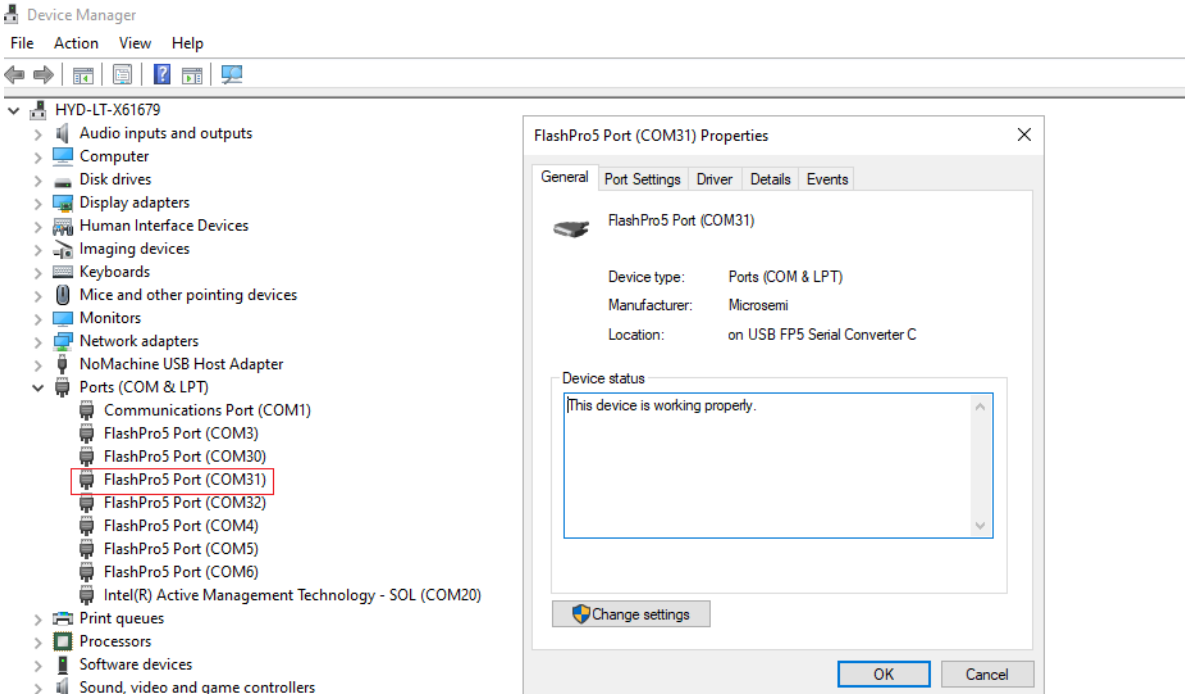
Jumper	Pin (From)	Pin (To)	Comments
J11, J17, J19, J21, J23, J26, J27, J28	1	2	Default
J16	2	3	Default
J32	1	2	Default
J33	1 2	3 4	Default

Note: Switch off the power supply switch, SW6, while connecting the jumpers.

2. Connect the USB cable (mini USB to Type-A USB cable) to J47 of the RTG4 Development Kit and other end of the cable to the USB port of the host PC.
3. Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the device manager of the host PC.

Figure 4 shows the USB 2.0 serial port properties and the connected COM31 and USB serial converter C.

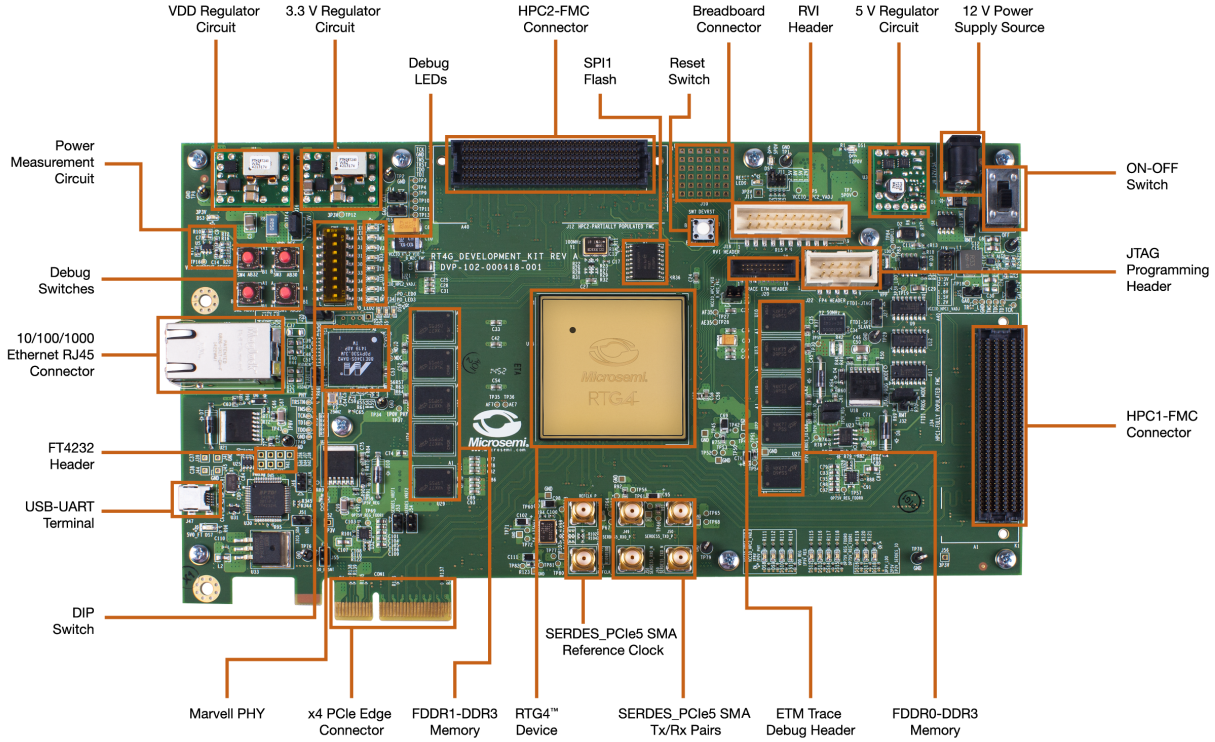
Figure 4 • **USB to UART Bridge Drivers**



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

Figure 5 shows the board setup for running the EDAC demo on the RTG4 Development Kit.

Figure 5 • **Board Set Up**



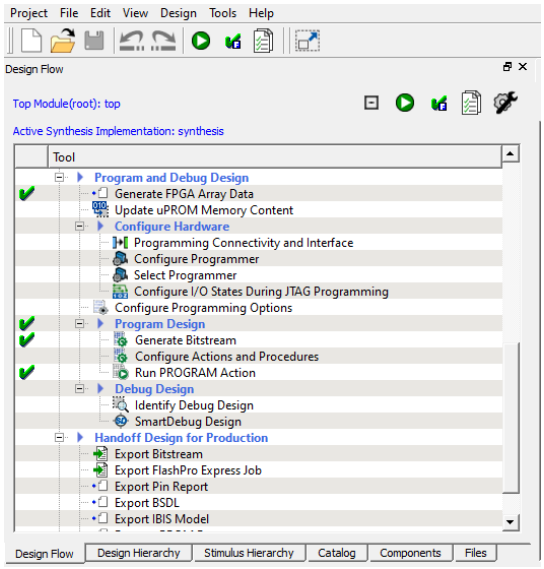
2.6.2 Programming the Demo Design

1. Launch Libero SOC software.
2. 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.

Note: Once the programming is done with the job file through FlashPro Express software, proceed to EDAC Demo GUI, page 9. Otherwise, proceed to the next step.

3. In the Libero design flow, click **Run Program action**.
4. Once Programming is complete, green tick appears in front of 'Run Program action' indicating successful programming of the demo design.

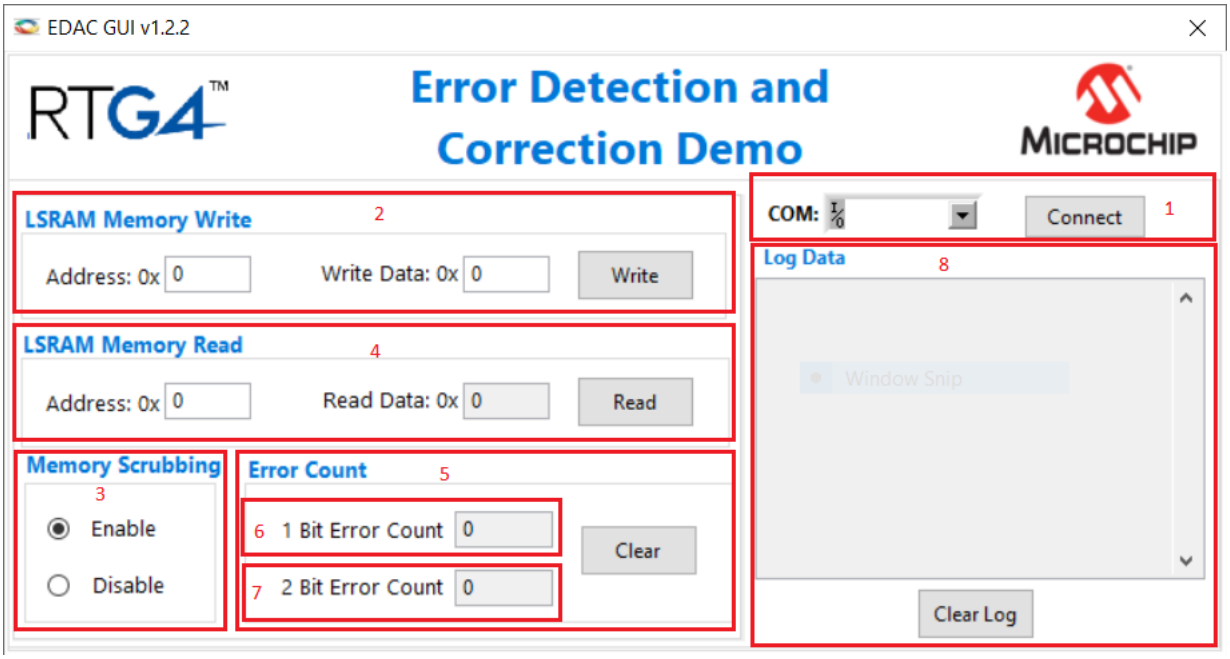
Figure 6 • **Run Program Action**



2.6.3 EDAC Demo GUI

The EDAC demo is provided with a user-friendly GUI, as shown in Figure 7, that runs on the host PC, which communicates with the RTG4 Development Kit. The UART is used as the underlying communication protocol between the host PC and RTG4 Development Kit.

Figure 7 • **EDAC Demo GUI**



The GUI contains the following sections:

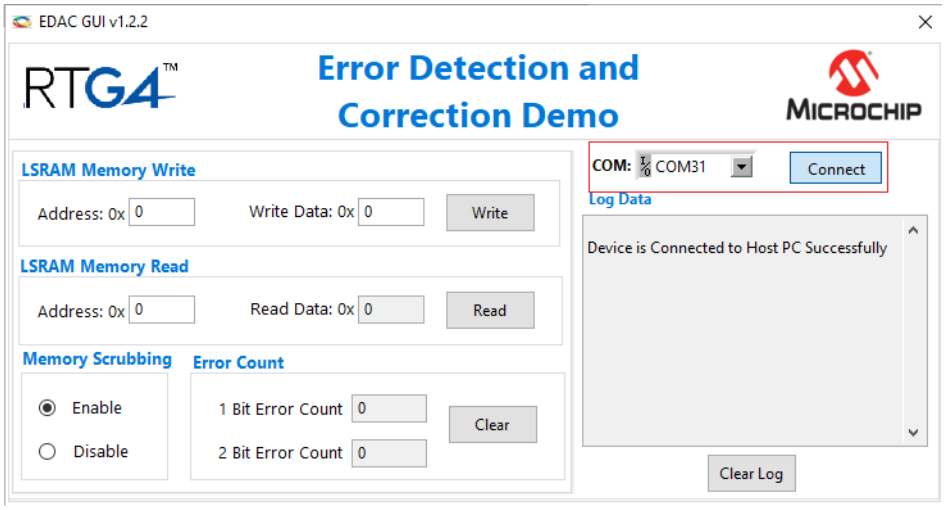
1. COM port selection to establish the UART connection to RTG4 FPGA with the 115200 baud rate.
2. **LSRAM Memory Write**: To write the 8-bit data to the specified LSRAM memory address.
3. **Memory Scrubbing**: To enable or disable the scrubbing logic.
4. **LSRAM Memory Read**: To read the 8-bit data from the specified LSRAM memory address.
5. **Error Count**: Displays the error count and provides an option to clear the counter value to zero.
6. **1-bit Error Count:** Displays 1- bit error count and provides an option to clear the counter value to zero.
7. **2-bit Error Count**: Displays 2-bit error count and provides an option to clear the counter value to zero.
8. **Log Data**: Provides the status information for every operation performed using the GUI.

2.7 Running the Demo

The following steps describe how to run the demo:

1. Go to *<EDAC_GUI_folder>\v1.2.2\v1.2.2\Exe* and double-click EDAC_GUI.exe as shown in Figure 8.
2. Select the **COM31** port from the list and click **Connect**.

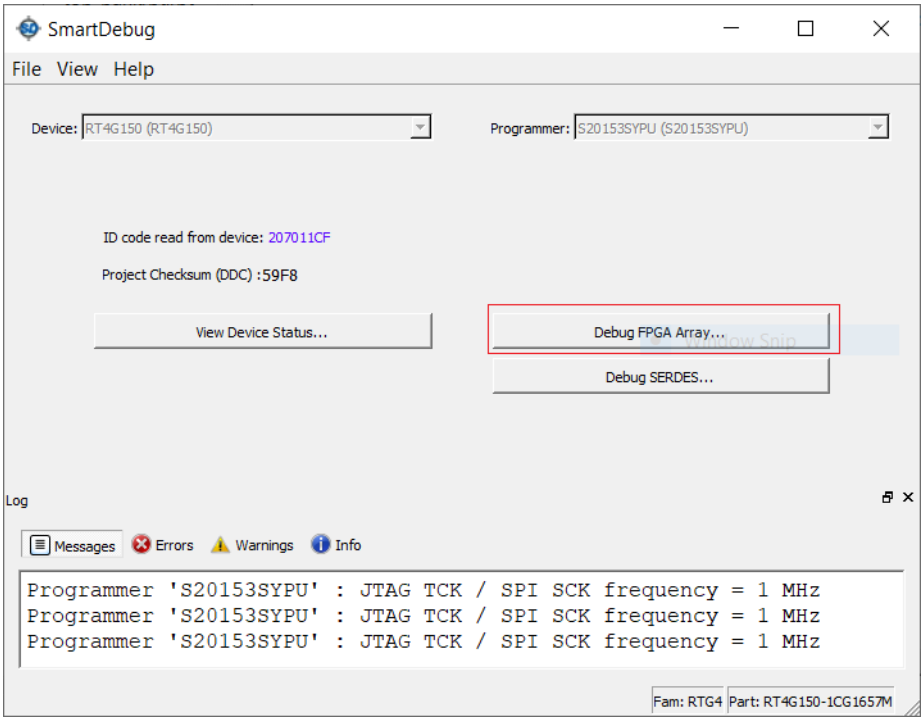
Figure 8 • **EDAC GUI**



2.7.1 Single bit error injection and correction

1. In the provided Libero design, double-click on the SmartDebug Design in the design flow.
2. In the SmartDebug GUI, click **Debug FPGA Array**.

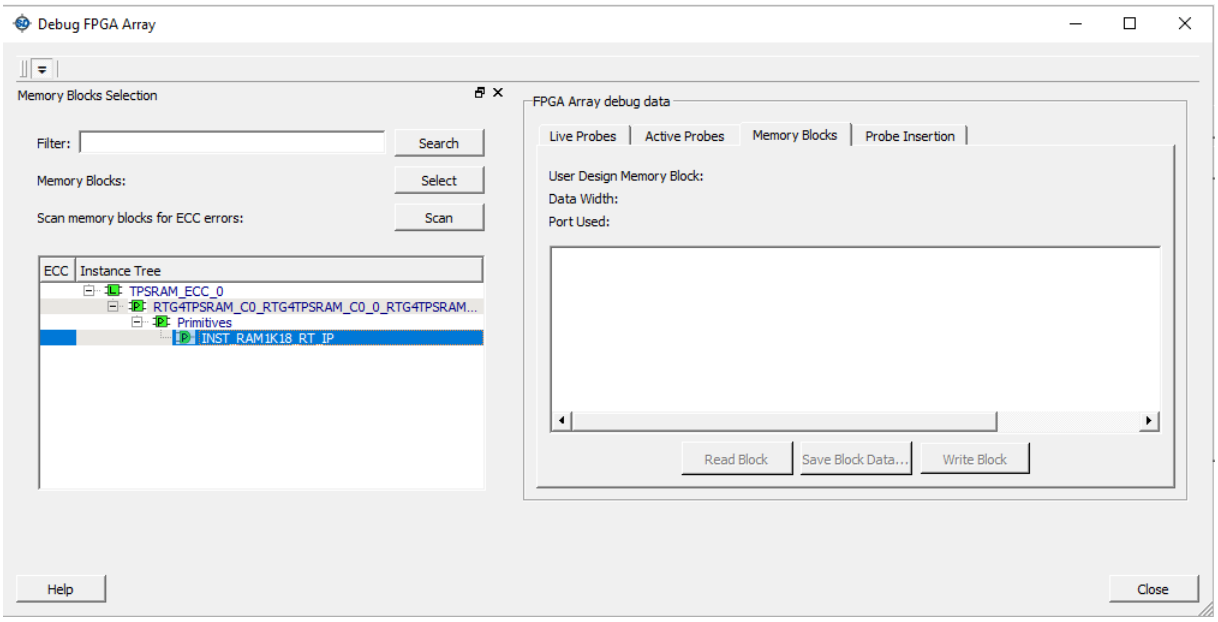
Figure 9 • **Open SmartDebug**



3. In the Debug FPGA Array window, go to the **Memory Blocks** tab. It will show the LSRAM block in the design with a logical and physical view. Logical blocks are shown with an L icon, and physical blocks are shown with a P icon.

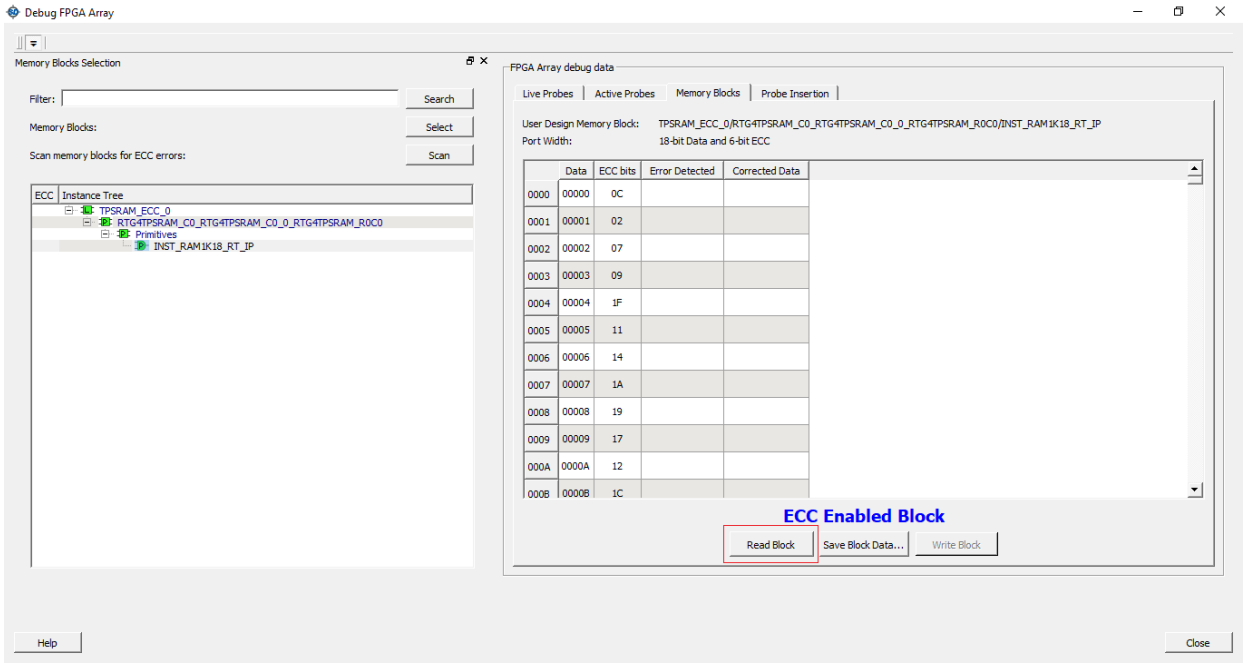
4. Select the physical block instance and right-click **Add**.

Figure 10 • **Select Memory Block**



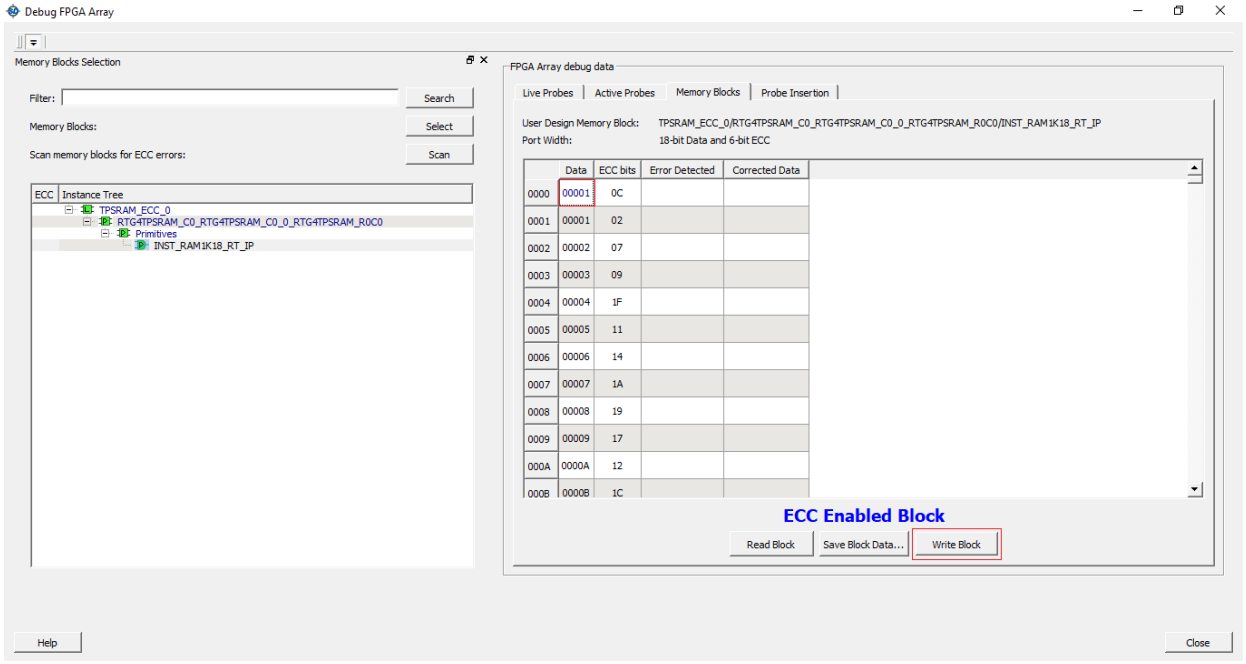
5. To read the memory block, click **Read Block**.

Figure 11 • **Read Memory Block**



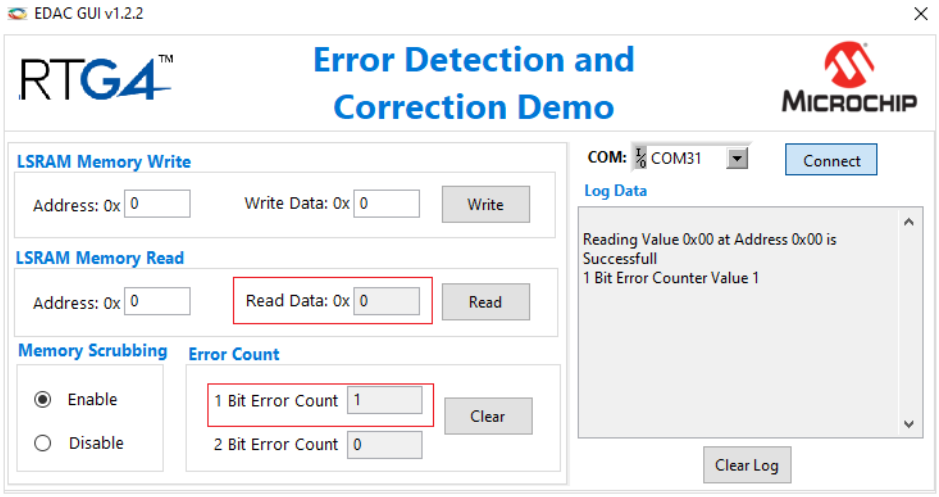
6. Inject 1 bit error in the 8 bit data at any location of LSRAM up to depth 256, as shown in the following figure where 1 bit error is injected at 0^{th} location of the LSRAM.
7. Click **Write Block** in order to write the modified data to the intended location.

Figure 12 • **Single Bit Error Write SmartDebug**



8. Go to the EDAC GUI and enter the **Address** field in the **LSRAM Memory Read** section and click **Read,** as shown in the following figure.
9. Observe **1 Bit Error Count** and **Read Data** fields in the GUI. The error count value increases by 1. The Read Data field displays the correct data as the EDAC corrects the error bit.

Figure 13 • **Single Bit Corrected EDAC with Error Count**

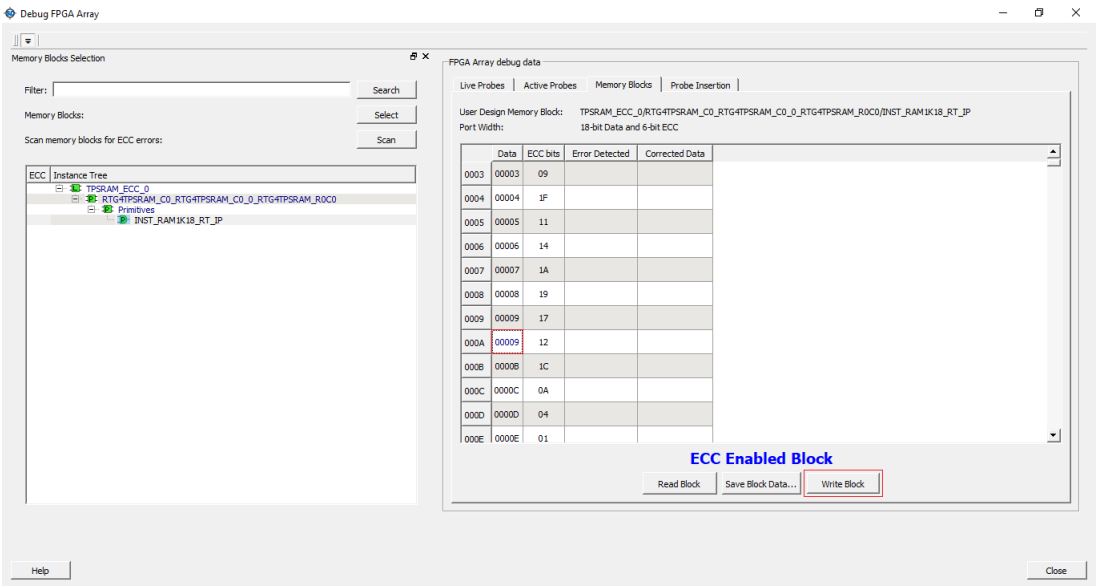


Note: If memory scrubbing is not enabled, then the error count is incremented for every read from the same LSRAM address as it causes the 1-bit error.

2.7.2 Double bit error injection and Detection

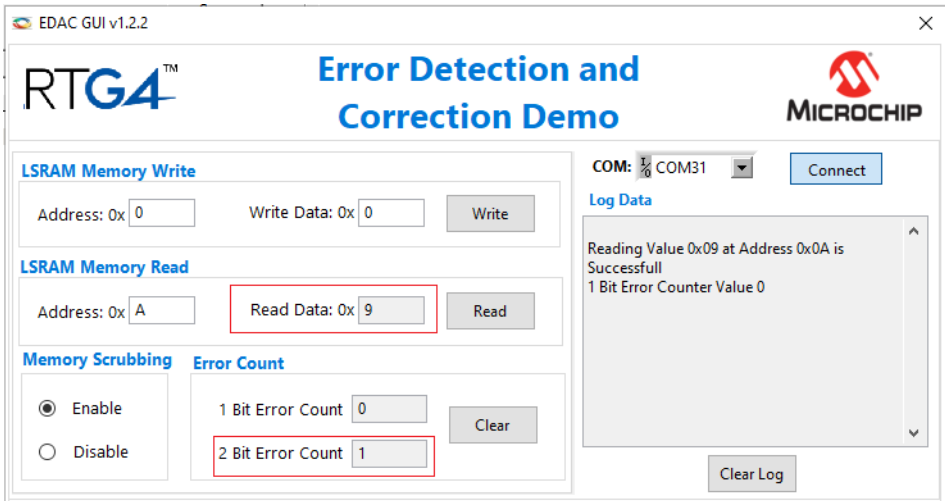
1. Perform step 1 to step 5 as given in Single bit error injection and correction, page 10.
2. Inject 2-bit error in the 8-bit data at any location of LSRAM up to depth 256, as shown in the following figure where the 2-bit error is injected at location 'A' of the LSRAM.
3. Click **Write Block** to write the modified data to the intended location.

Figure 14 • **2-Bit Write SmartDebug**



4. Go to the EDAC GUI and enter the **Address** field in the **LSRAM Memory Read** section and click **Read,** as shown in the following figure.
5. Observe **2-bit Error Count** and **Read Data** fields in the GUI. The error count value increases by 1. The **Read Data** field displays the corrupted data.

Figure 15 • **2-Bit EDAC Detection Error Count**



All the actions performed in RTG4 are logged in the Serial Console section of GUI.

2.8 Conclusion

This demo highlights the EDAC capabilities of the RTG4 LSRAM memories. The 1-bit error or 2-bit error are introduced through SmartDebug GUI. 1-bit error correction and 2-bit error detection is observed using an EDAC GUI.

3 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.

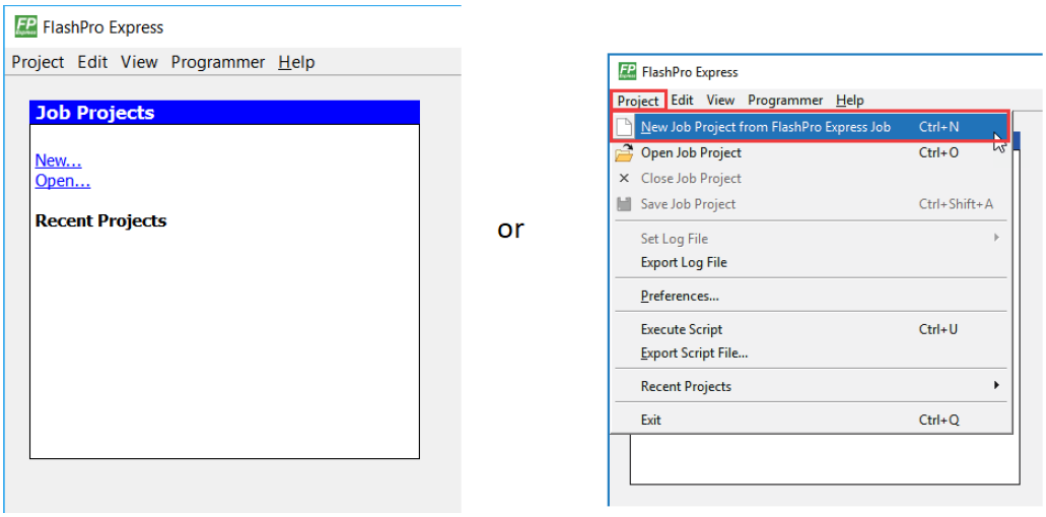
To program the device, perform the following steps:

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 16 • **FlashPro Express Job Project**



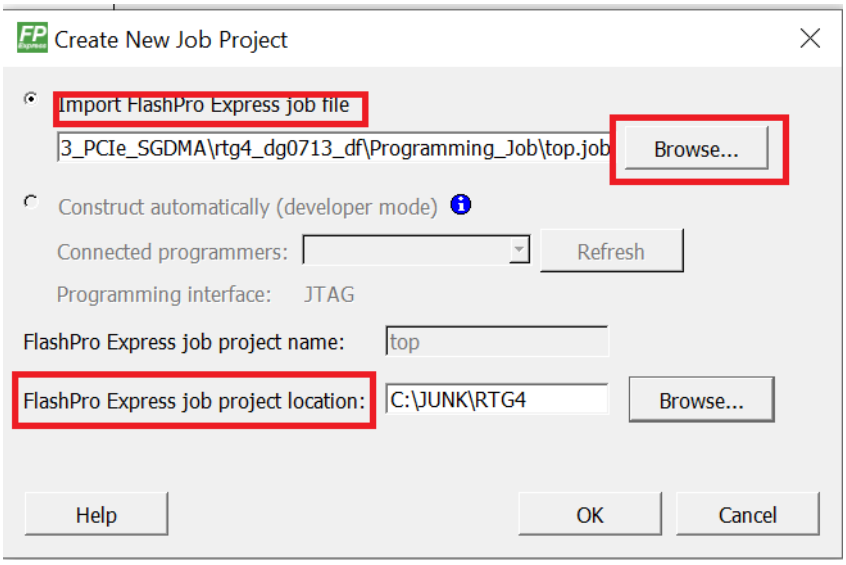
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

- **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_dg0703_df\Programming_Job`

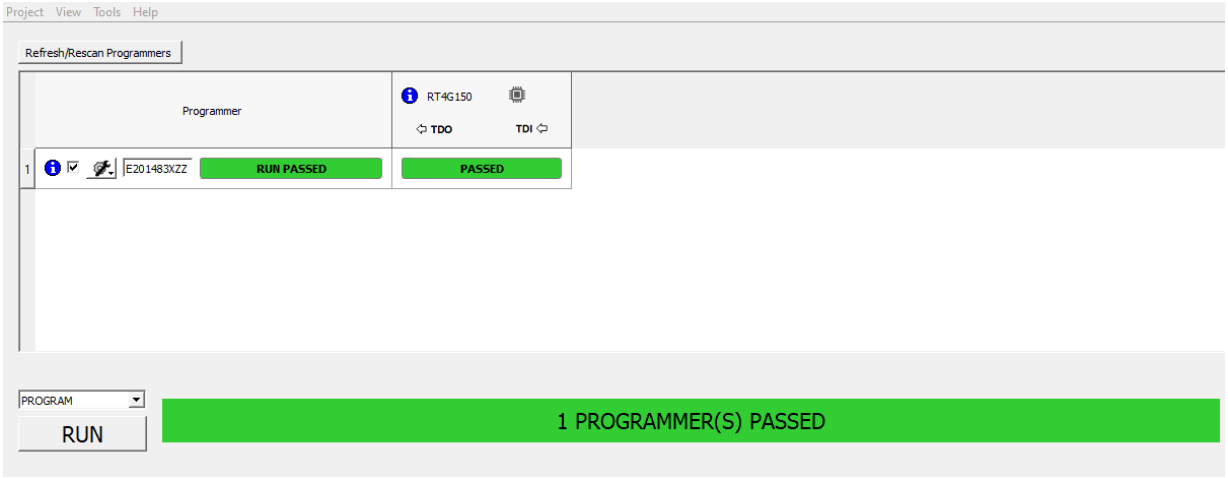
- **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 17 • **New Job Project from FlashPro Express Job**



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

Figure 18 • **FlashPro Express—RUN PASSED**



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

4 Appendix 2: 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_dg0703_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.