DG0625
Demo Guide
Interfacing RTG4 with External DDR3 Memory

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.

50200625. 8.0 11/21

Contents

Figures

Tables

1 Revision History

The following table shows important changes made in this document for each revision.

1.1 Revision 8.0

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

- Updated Table 1, page 2.
- Replaced Figure 4 on page 6.
- Replaced Figure 5 on page 7.
- Replaced Figure 7 on page 8.
- Replaced Figure 8 on page 10.
- Updated Reset Structure, page 8.
- Updated Running the Simulation, page 13.

1.2 Revision 7.0

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

- Replaced Figure 4 on page 6, and Figure 8 on page 10 through Figure 10 on page 11.
- Added Appendix 1: Programming the Device Using FlashPro Express, page 23.
- Added Appendix 2: Running the TCL Script, page 26.
- Removed the references to Libero version numbers.

1.3 Revision 6.0

Updated the document for Libero SoC v11.9.

1.4 Revision 5.0

Updated the document for Libero v11.8 SP2.

1.5 Revision 4.0

Updated the document for Libero v11.7 SP1.

1.6 Revision 3.0

Updated the document for Libero v11.7.

1.7 Revision 2.0

Updated the document for Libero v11.6.

1.8 Revision 1.0

The first publication of this document.

2 Interfacing RTG4 FPGA with External DDR3 Memory Through FDDR

This demo shows how the FPGA fabric logic can access external double-data-rate three (DDR3) memories using built-in fabric double-data-rate (FDDR) in RTG4 devices.

The demo has two parts:

- Simulation
- Running the demo on the RTG4 Development Kit

In the demo design, the advanced extensible interface (AXI) master in the FPGA fabric accesses the DDR memory present in the RTG4 Development Kit using the FDDR. A host utility, `RTG4_DDR_Demo_Utility` is provided along with the demo design files. Using the utility, the AXI master logic is driven. The AXI master converts the commands from the utility to AXI transactions for the FDDR to perform the read or write operations on the external DDR3 memory.

2.1 Design Requirements

Table 1 shows the design requirements for running this demo design.

Table 1 • **Design Requirements**

Requirement	Version
Hardware	
RTG4 FPGA Development Kit: • USB 2.0 cable • 12 V, 5A AC power adapter and cords	RT4G150-CB1657PROTO FPGA
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	
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

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

2.3.1 Introduction

The demo design files are available for download from the following path in the Microsemi website: *http://soc.microsemi.com/download/rsc/?f=rtg4_dg0625_df*

The demo design files include:

- Demo_Utility
- Libero_project
- Programming_File
- TCL_scripts
- Source_File
- Readme.txt

Figure 1 shows the top-level structure of the design files. For further details, refer to the `Readme.txt` file.

Figure 1 • **Demo Design Files Top-Level Structure**

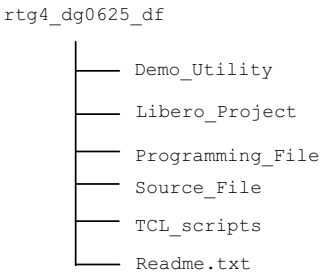
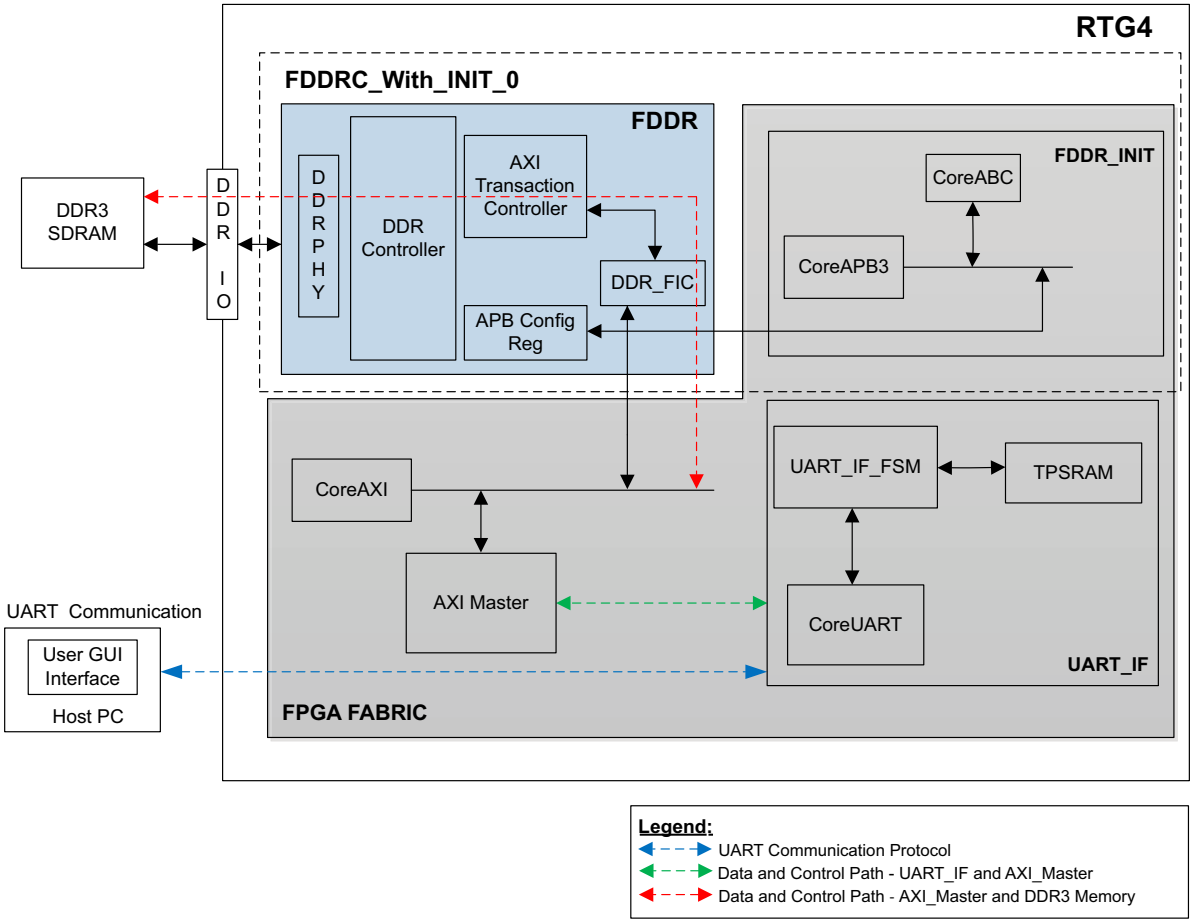


Figure 1 shows the top-level view of the demo design. In the demo design, the AXI master implemented in the FPGA fabric accesses the DDR3 memory present on the RTG4 Development Kit using the FDDR. The AXI master logic communicates to the FDDR through the CoreAXI interface and the DDR_FIC interface. The read or write operations initiated by the `RTG4_DDR_Demo_Utility` are sent to the UART_IF block using the UART protocol. The AXI master receives the address and data from the UART_IF block.

During a write operation, the UART_IF block sends the address and data to the AXI master logic.

During a read operation, the UART_IF block sends the address to the AXI master. The AXI master reads the data from DDR3 memory and stores it in TPSRAM. When the read operation is complete, the data read is sent to the host PC through UART.

Figure 2 • **RTG4 DDR Demo Block Diagram**



In this demo design, different blocks are configured, which are as follows:

- FDDR is configured for DDR3 memory available on the RTG4 Development Kit. The DDR3 memory is a Micron DRAM (Part Number: MT41K256M8DA-125 IT:K).
- DDR_FIC is configured for AXI bus interface.
- AXI clock is configured for 80 MHz and DDR3 clock is configured for 320 MHz.
- CoreUART IP has the following configuration:
  - Baud Rate: 115200
  - Data Bits: 8
  - Parity: None
- RTG4TPSRAM has the following configuration:
  - Write port depth: 256
  - Write port width: 64
  - Read port depth: 1024
  - Read port width: 16

For more information about the RTG4TPSRAM, refer to *RTG4 FPGA Two-Port Large SRAM Configuration User's Guide*.

For more information about how to configure the FDDR, refer to Appendix 3: RTG4 DDR Memory Controller Configuration and Initialization, page 27.

2.4 Demo Design Features

The RTG4 DDR demo design has the following features:

- Single AXI read or write transactions
- 16-beat burst AXI read or write transactions
- DDR3 memory model simulation using testbench
- Design validation using the RTG4 Development Kit that has the DDR3 memory
- Initiation of the read or write transactions using `RTG4_DDR_Demo_Utility`

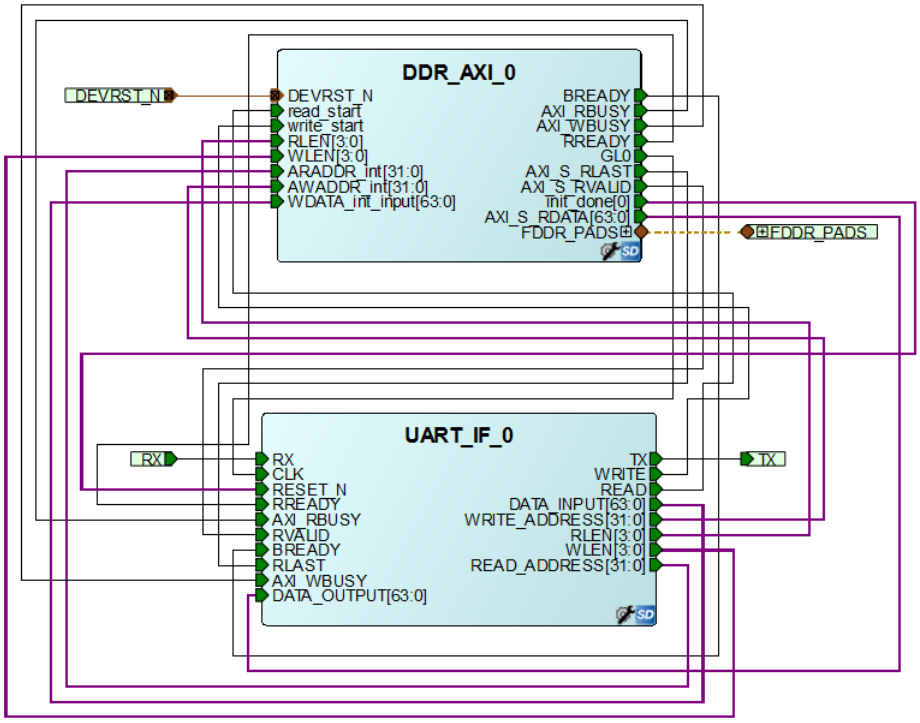
2.5 Demo Design Description

The demo design consists of the following SmartDesign components:

- **DDR_AXI_0**: Handles the data transactions between the FDDR and the DDR3 SDRAM.
- **UART_IF_0**: Handles the communication between the host PC and the RTG4 Development Kit.

Figure 3 shows the DDR_AXI_0 and UART_IF_0 connection.

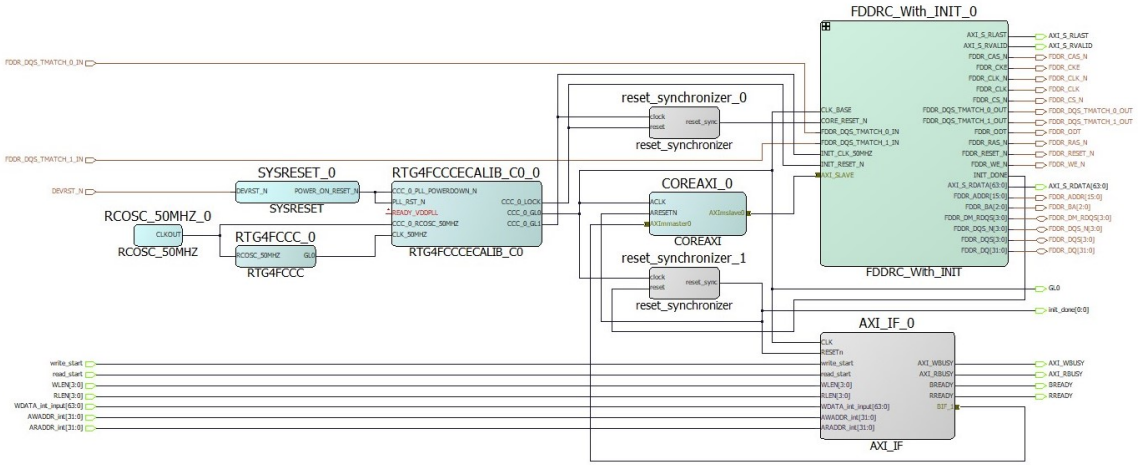
Figure 3 • **RTG4_DDR_Demo SmartDesign**



2.5.1 DDR_AXI_0

DDR_AXI_0 consists of the FDDR subsystem and the AXI_IF_0 master logic. The AXI_IF_0 master logic is an RTL code that implements the AXI read and write transactions. It receives the read or write operations, burst length (RLEN and WLEN), address, and data as inputs. Based on inputs received, it communicates with the DDR3 memory through the FDDR. Figure 4 shows the DDR_AXI_0 SmartDesign component.

Figure 4 • **DDR_AXI_0 SmartDesign Component**



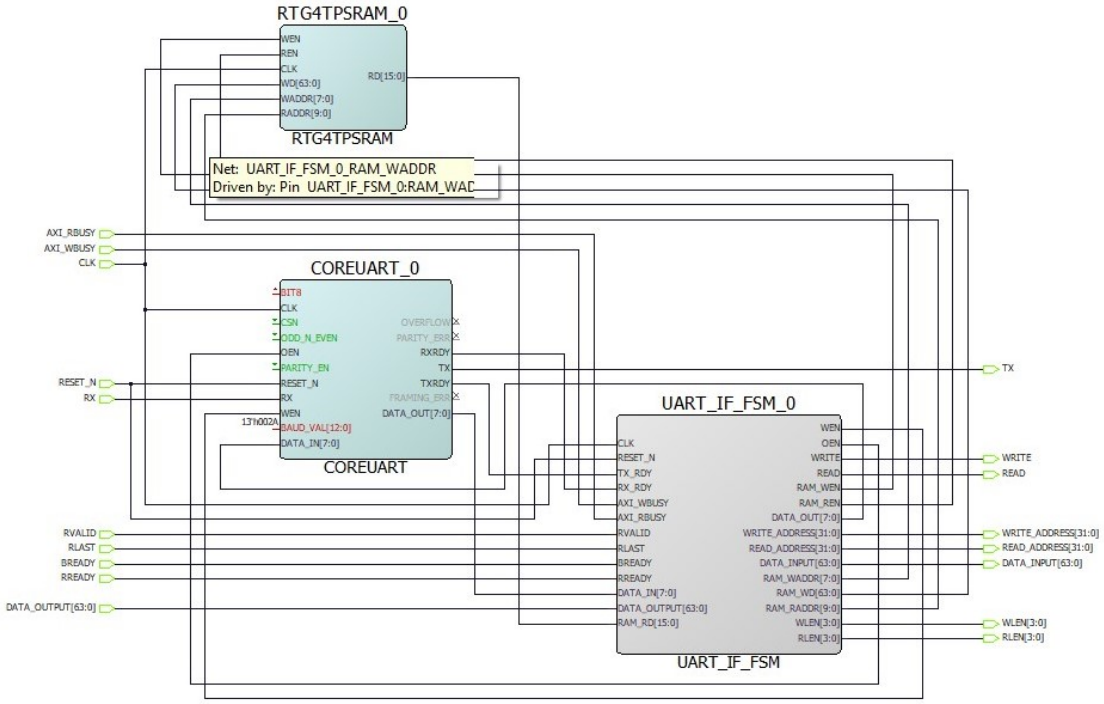
2.5.2 UART_IF_0

The UART_IF_0 SmartDesign component handles the UART communication between the host PC demo utility and the AXI master logic. The COREUART_0 IP receives the UART signals from the host PC user interface. The UART_IF_FSM_0 is a wrapper for the COREUART_0, collects the data from the COREUART_0 IP and converts the data to the relevant AXI_IF_0 master signals. For more information about CoreUART, refer to *CoreUART Handbook*.

For a single write operation, the UART_IF_FSM_0 wrapper receives the address and data from the demo utility. For a burst write operation, the address and data are received from the demo utility and the subsequent incremental data is provided by the UART_IF_FSM_0 wrapper.

For a burst read operation, UART_IF_FSM_0 collects the address from the demo utility and sends that to the AXI_IF_0 master logic. It receives the read data from the AXI_IF_0 master logic and stores it in the RTG4TPSRAM_0. After completion of the read burst transactions, the UART_IF_FSM_0 wrapper fetches the stored data from the RTG4TPSRAM_0 and sends it to the COREUART IP. Figure 5 shows the UART_IF_0 SmartDesign component. For more information about RTG4TPSRAM, refer to *RTG4 FPGA Two-Port Large SRAM Configuration User Guide* from Libero.

Figure 5 • **UART_IF_0 SmartDesign Component**

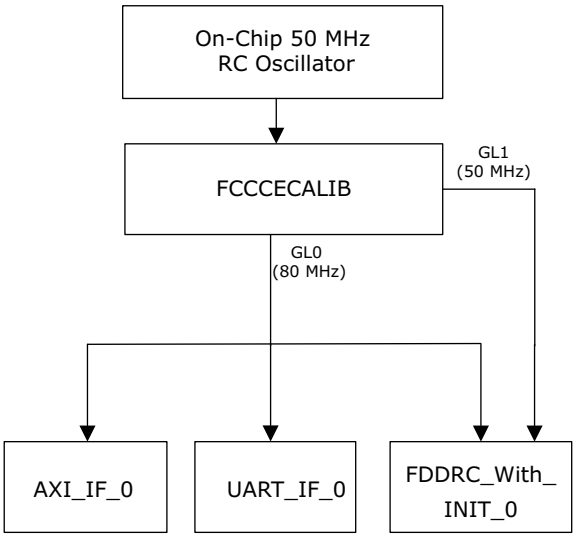


2.6 Clocking Structure

The on-chip 50 MHz oscillator provides the reference clock to the FCCC and FCCCECALIB blocks. The FCCCECALIB block provides two clocks, GL0 (80 MHz) and GL1 (50 MHz). The GL0 clock drives the AXI_IF_0, UART_IF_0, and FDDRC_With_INIT_0 blocks. The GL1 clock drives the CoreABC processor inside the FDDRC_With_INIT_0 block. The following figure shows the clocking structure of the design.

For more information about FCCCECALIB, refer to *UG0590: RTG4 FPGA Clock Conditioning Circuit with PLL Configuration User Guide* from Libero.

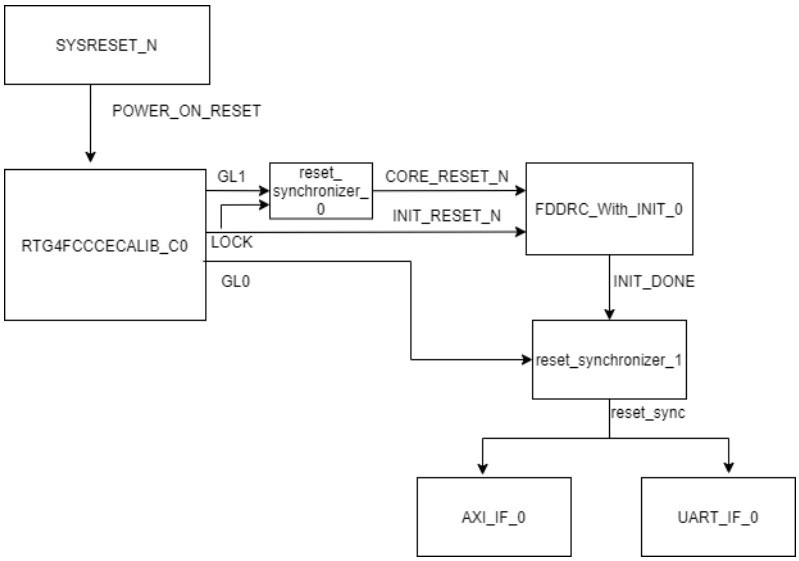
Figure 6 • **Clocking Structure**



2.7 Reset Structure

The POWER_ON_RESET signal is the input to the RTG4FCCCECALIB_C0 and the output signal is used in INIT_RESET_N and reset_synchronizer_0 to reset the FDDRC_With_INIT_0 block. After the reset of FDDRC_With_INIT_0, INIT_DONE signal is used in reset_synchronizer_1 to reset the AXI_IF_0 and UART_IF_0 blocks. The following figure shows the reset structure of the design.

Figure 7 • **Reset Structure**



2.8 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. For IP-wise utilization, refer to the respective handbooks.

Table 2 • **Resource Utilization**

Type	Used	Total	Percentage
4LUT	1120	151824	0.74
DFF	863	151824	0.57
I/O Register	0	2151	0.00
Logic Element	1343	151824	0.88

2.9 Demo Design Simulation

The demo design can be simulated using a testbench and Micron DDR3 memory model.

The simulation is set to run the following:

- Single AXI write and read operation
- 16-beat AXI burst write and read operation.

To run the simulation, ensure that the following files are present in the Libero SoC project:

- `ddr3.v`
- `ddr3_parameters.vh`
- `testbench.v`

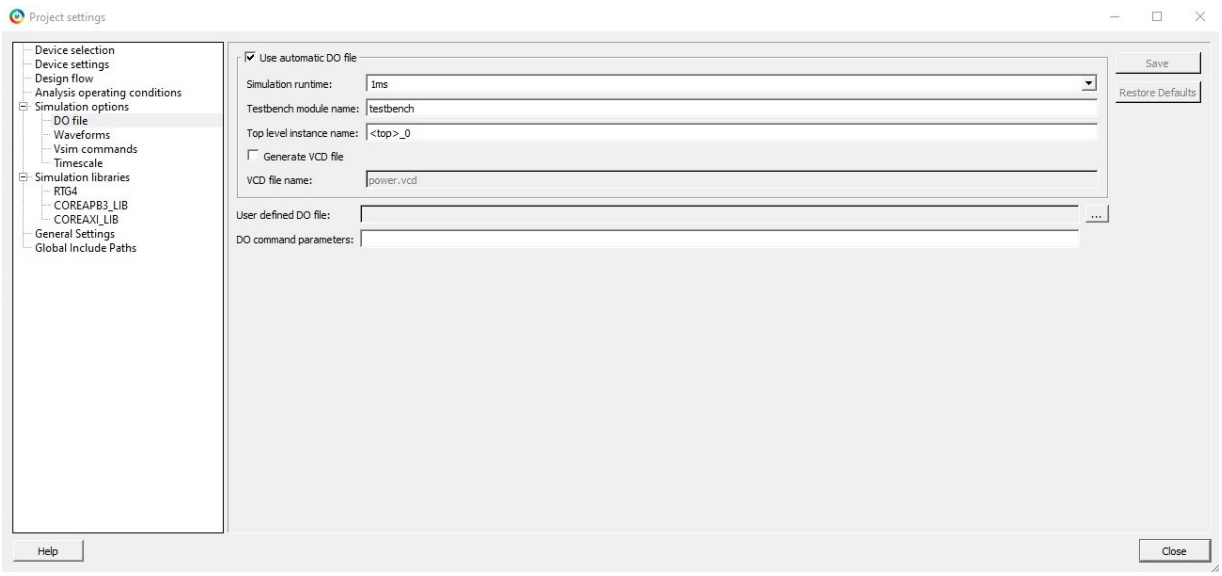
The default location of the files is:
<Download folder>\Libero_Project\stimulus

2.9.1 Simulation Setup

The following steps describe how to set up the simulation.

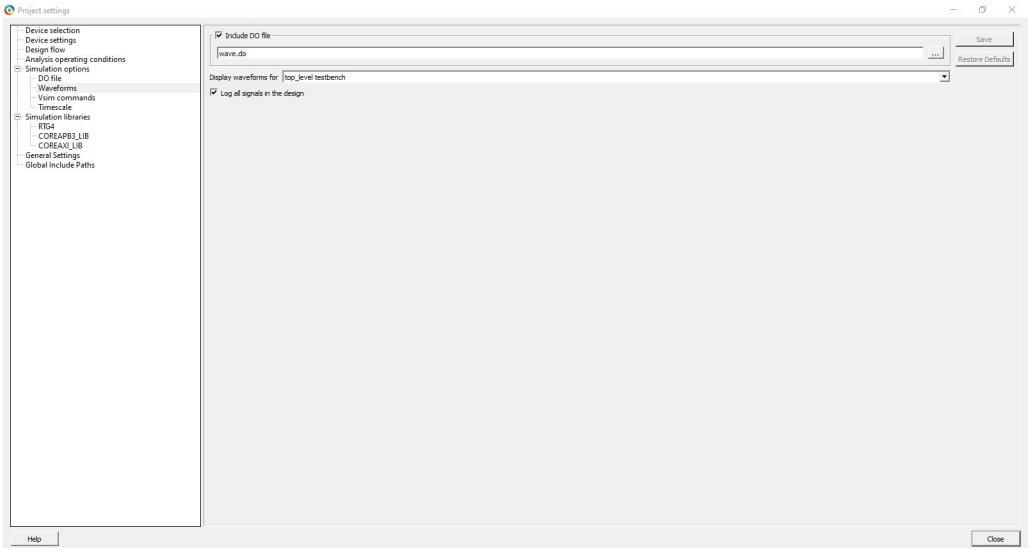
1. Launch the Libero SoC software.
2. Browse the `Libero_Project` project provided in the design file.
3. Choose **Project** > **Project Settings** > **Simulation Options**.
4. Ensure that the **DO File** tab has the configuration, as shown in Figure 8.

Figure 8 • **DO File Configuration**



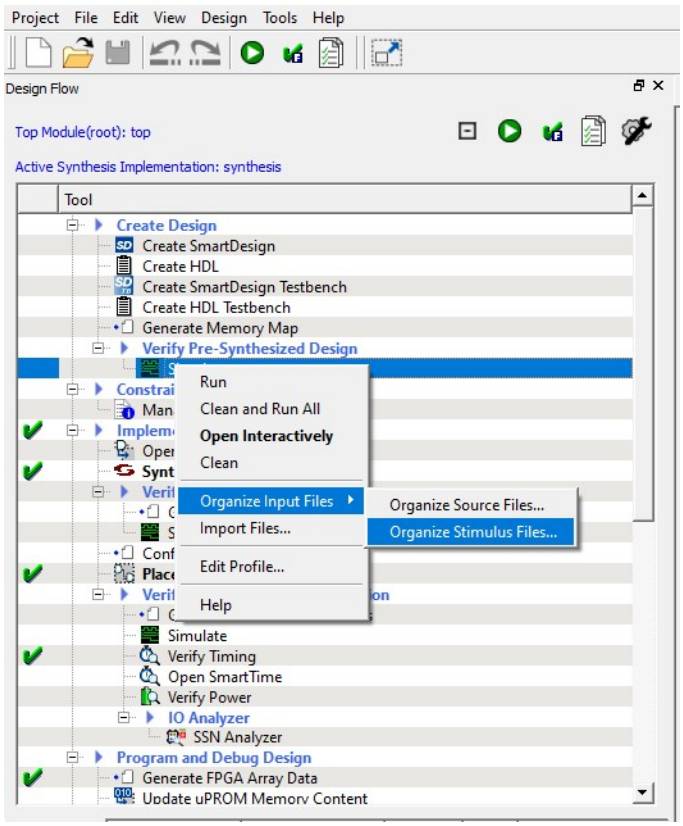
5. Ensure that the **Waveforms** tab has the configuration, as shown in Figure 9.

Figure 9 • **Waveforms Configuration**



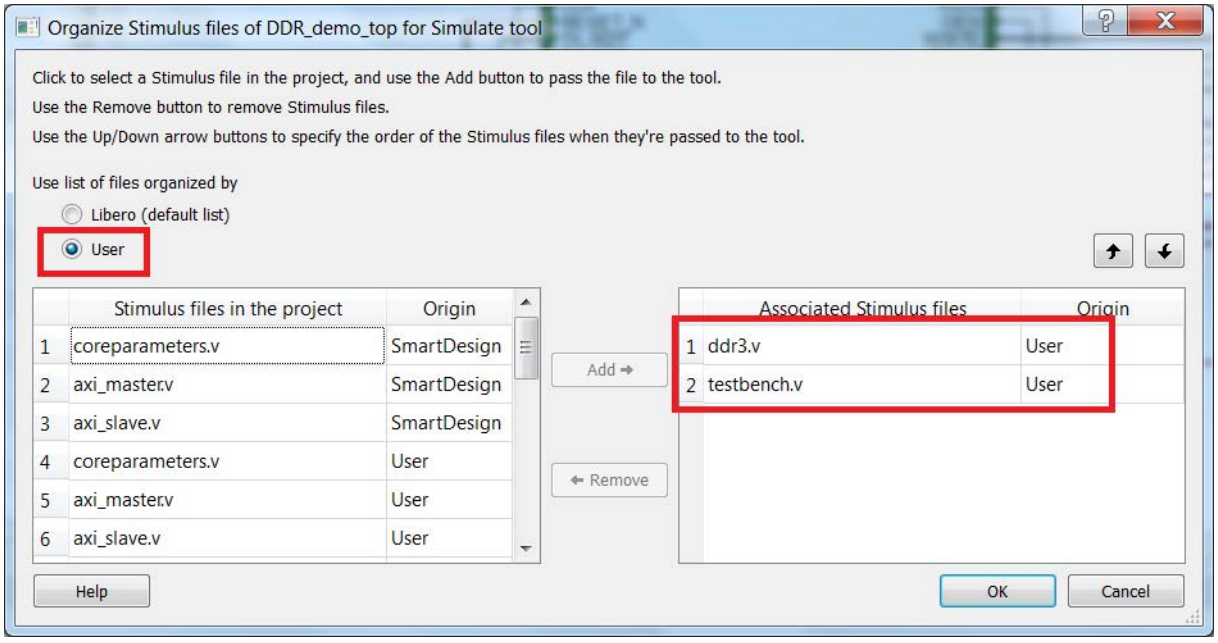
6. Go to **Design Flow** tab.
7. Right-click **Simulate** under **Verify Pre-Synthesized Design** and select **Organize Input Files** > **Organize Stimulus Files...,** as shown in Figure 10.

Figure 10 • **Design Flow Window - Simulate**



8. Ensure that the **Organize Stimulus files** window has the configuration, as shown in Figure 11.

Figure 11 • **Organize Stimulus Files Window**



2.9.2 Running the Simulation

The following steps describe how to run the simulation:

1. Right-click **Simulate** under **Verify Pre-Synthesized Design**.
2. Click **Open Interactively**.
   Simulation run time is 1 ms, as shown in Figure 8 on page 10. Figure 12 shows the transcript window of the simulation.

Figure 12 • **Transcript Window**

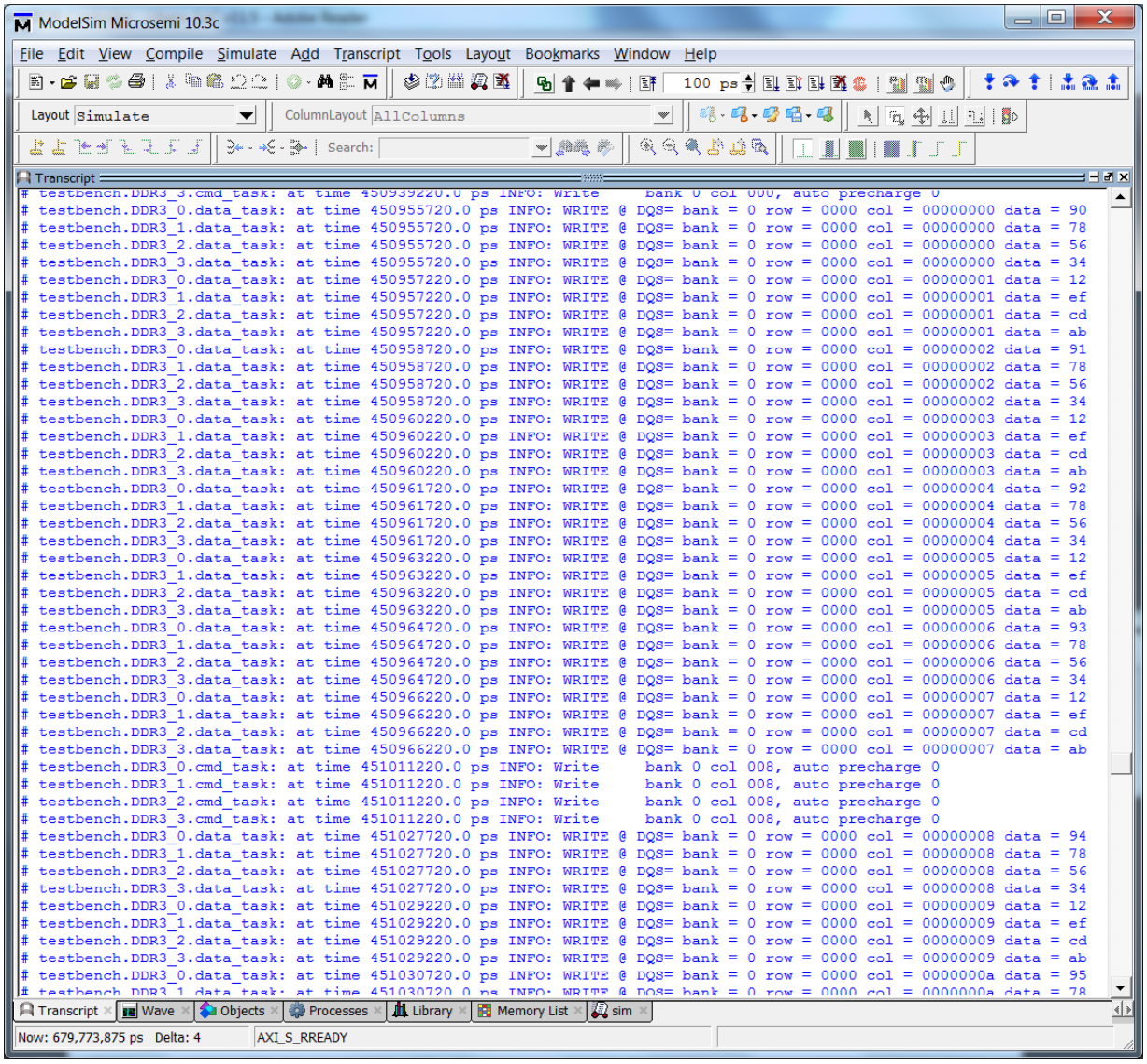


Figure 13 shows the single AXI write and AXI read operation as per AXI specification.

Figure 13 • **Single AXI Write and Read Operation**

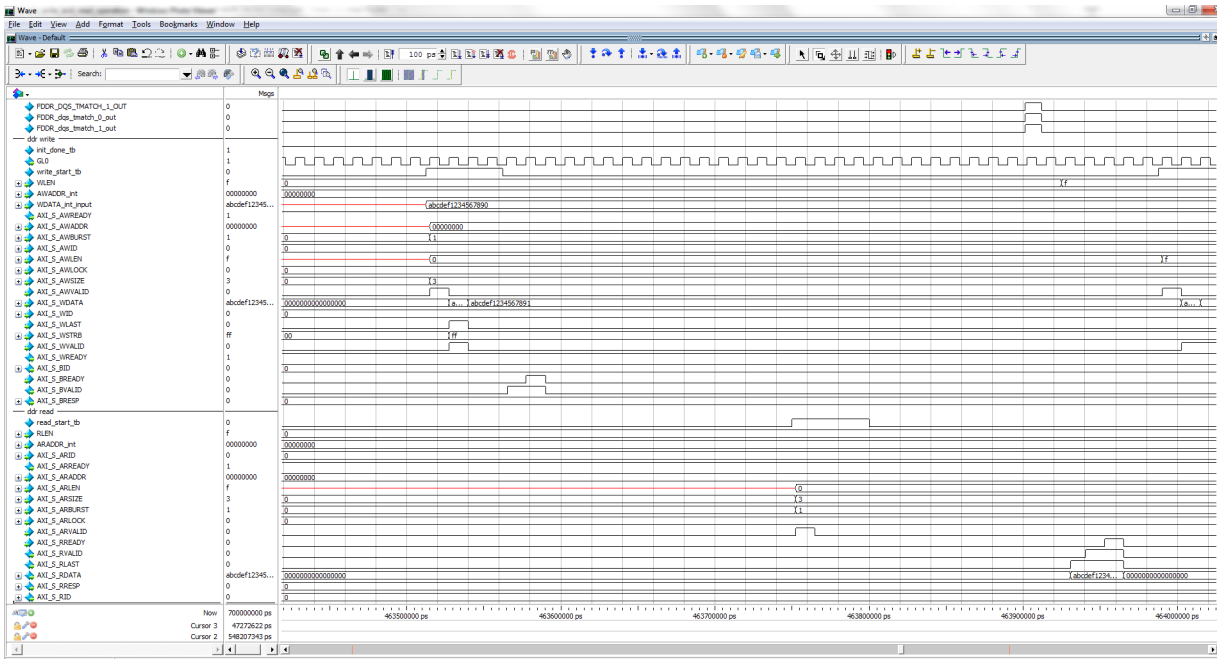
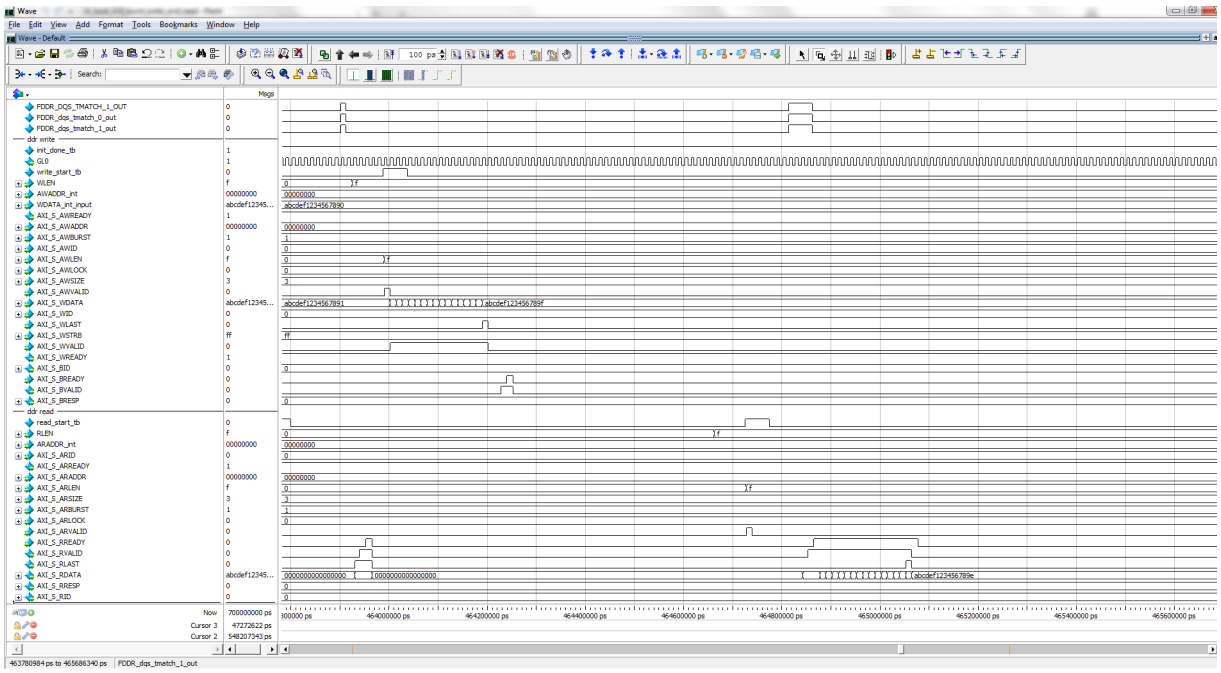


Figure 14 shows the 16-beat AXI burst write and read operation as per AXI specification.

Figure 14 • **16-Beat AXI Burst Write and Read**



These waveforms shows the AXI4 single and burst transfers initiated by AXI4 master. These AXI4 transactions are input to the FDDR, which converts them to read/write operations across external DDR3 memory.

2.10 Demo Setup

This demo design uses the RTG4 Development Kit. Figure 15 on page 15 shows the RTG4 Development Kit. The following steps describe how to set up the hardware demo:

1. Connect the jumpers on the RTG4 Development Kit, as shown in Table 3.

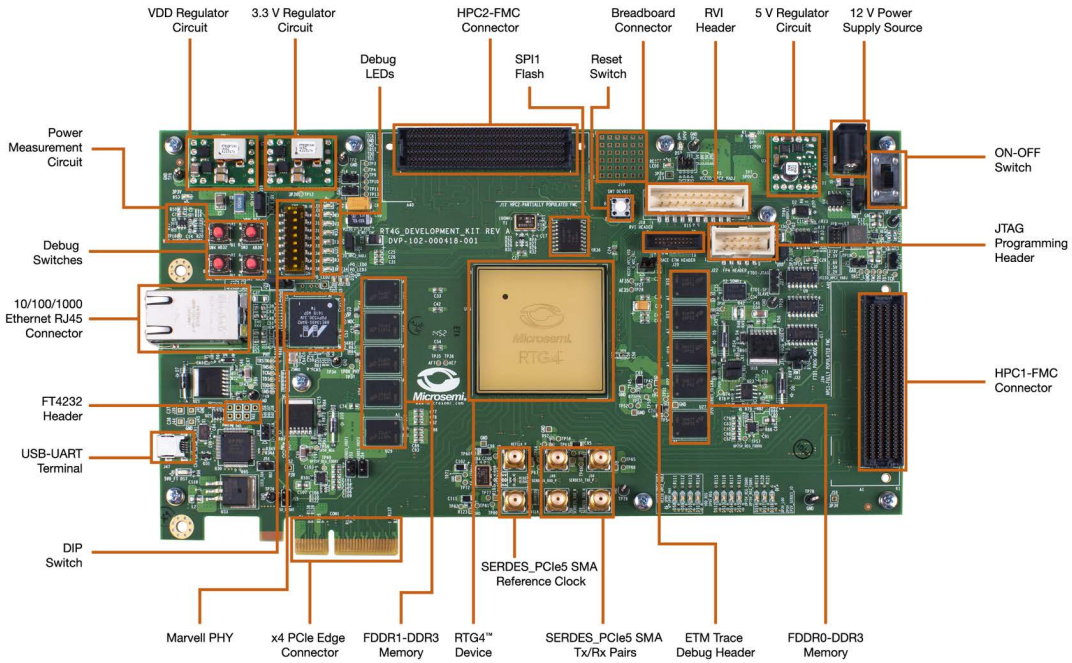
Table 3 • **RTG4 Development Kit Jumper Settings**

Jumper	Pin (from)	Pin (to)	Comments
J32	1	2	Default
J27	1	2	Default
J26	1	2	Default
J23	1	2	Default
J21	1	2	Default
J19	1	2	Default
J11	1	2	Default
J16	2	3	Default

CAUTION: Ensure that the power supply switch **SW6** is switched **OFF** while connecting the jumpers.

2. Connect the power supply to the J9 connector, switch **ON** the power supply switch, **SW6**.
3. Connect the host PC USB port to the RTG4 Development Kit J47 USB connector using the USB A to mini-B cable.

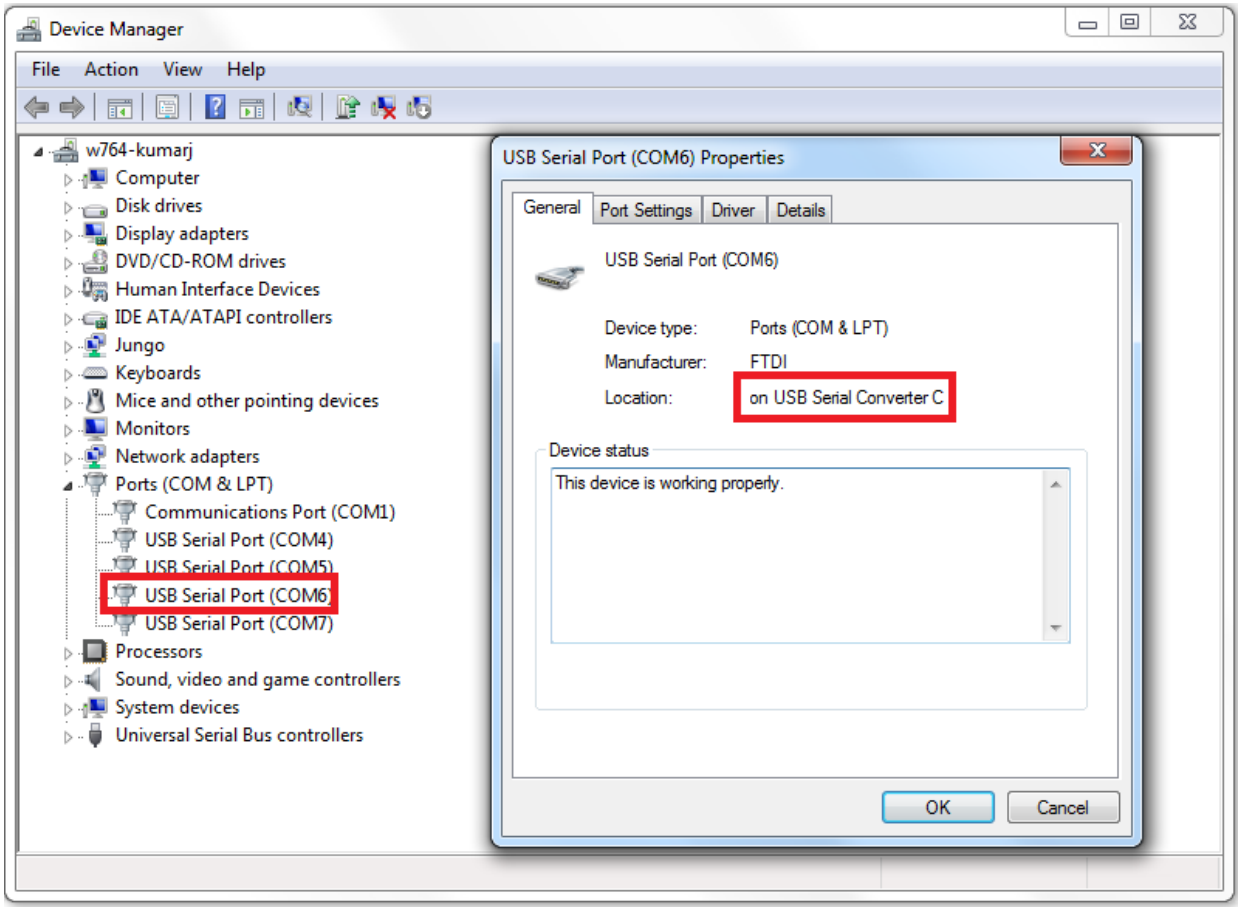
Figure 15 • **RTG4 Development Kit**



4. Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the **Device Manager** of the host PC. The future technology devices international (FTDI) USB to UART converter enumerates four COM ports. For USB 2.0, note down the USB Serial Converter C COM port number to use it in the GUI. Figure 16 shows the USB 2.0 serial port properties. As shown in

Figure 16, COM6 is connected to the USB serial converter C. Refer to "Appendix 4: Finding Correct COM Port Number when Using USB 3.0" on page 31 for finding the correct COM port in USB 3.0.

Figure 16 • **USB Serial 2.0 Port Properties**



5. If the USB to UART bridge drivers are not installed, download and install the drivers from *https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers*.

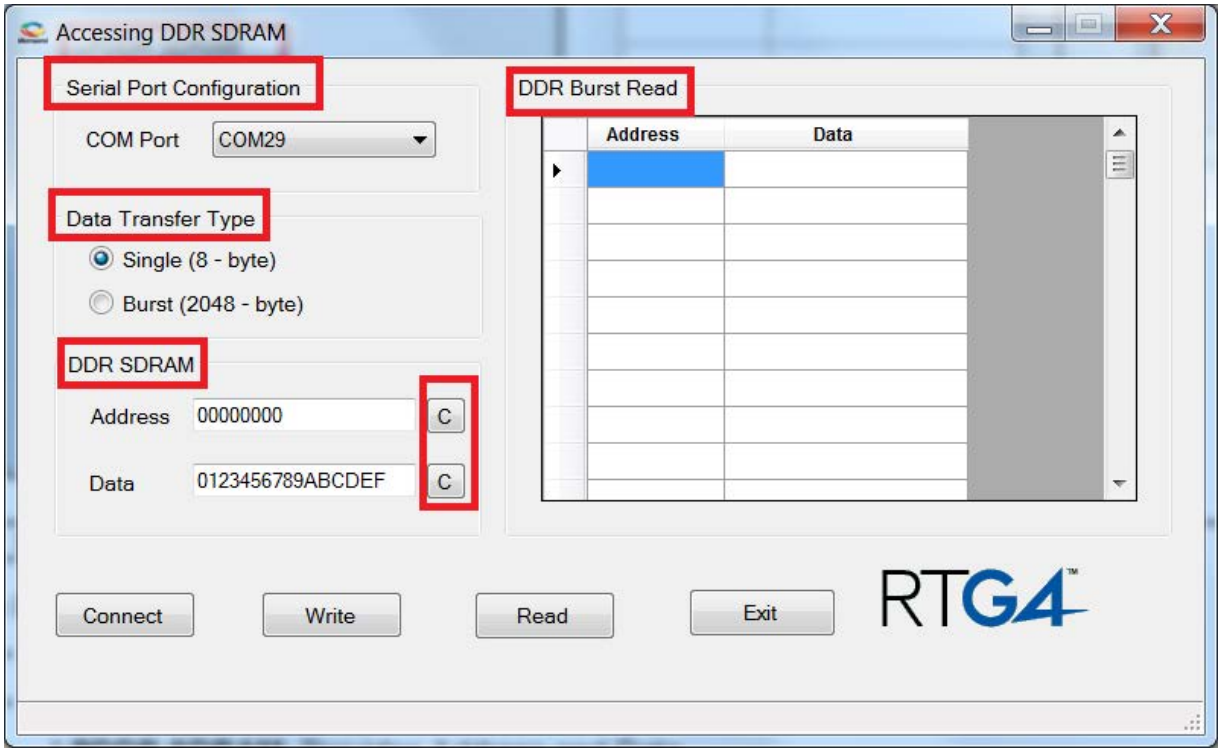
2.11 Programming the Device

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 23.

2.12 Running the Demo

The RTG4 DDR demo comes with a utility, `RTG4_DDR_Demo_Utility` that runs on the host PC to communicate with the RTG4 Development Kit. The UART protocol is used as the underlying communication protocol between the host PC and the RTG4 Development Kit. Figure 17 shows the initial screen of the `RTG4_DDR_Demo_Utility`.

Figure 17 • **RTG4_DDR_Demo_Utility**



The RTG4_DDR_Demo_Utility consists of the following sections:

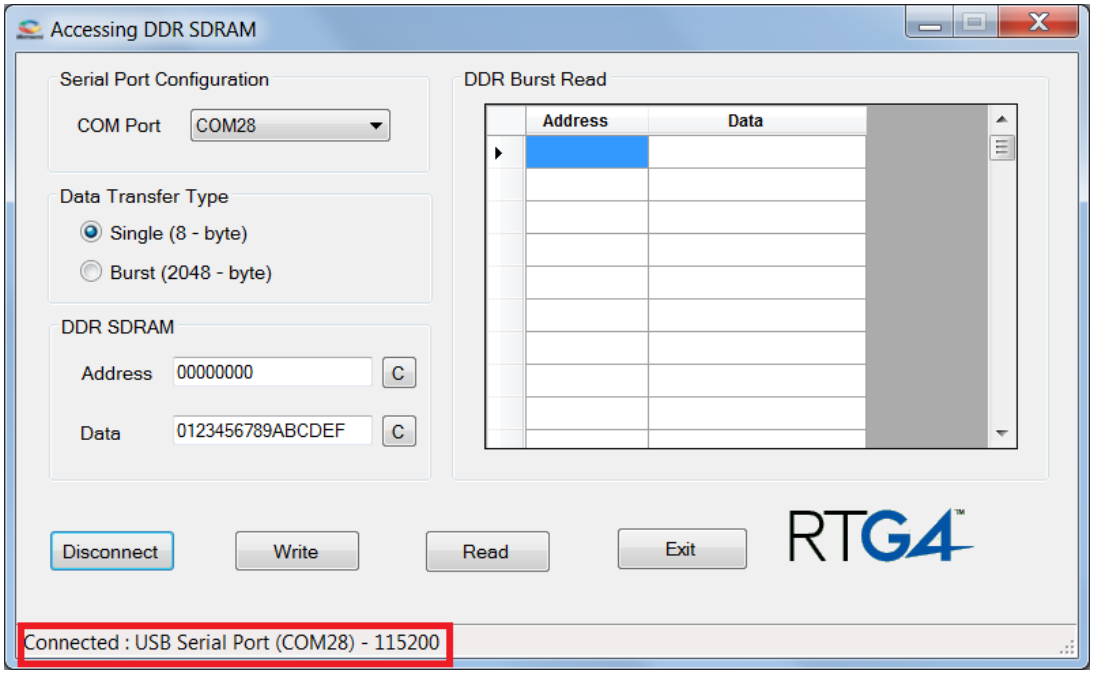
- **Serial Port Configuration**: Displays the serial port. Baud rate is fixed at 115200.
- **Data Transfer Type**: Single or Burst.
- **DDR SDRAM**: Provides Address and Data.
- **DDR Burst Read**: Displays the Burst Read values for the corresponding address.
- **C:** Clears the existing data.

2.12.1 Steps to Run the GUI

The following steps describe how to run the GUI:

1. Launch the utility. The default location is: *<download_folder>\Demo_Utility\RTG4_DDR.exe*.
2. Select the appropriate **COM Port** from the drop down menu. In this case, it is COM 28.
3. Click **Connect**. The connection status along with the COM Port and Baud rate is shown in the left bottom corner of the screen. Figure 18 shows the connection status of the utility.

Figure 18 • **RTG4_DDR_Demo - Connection Status**



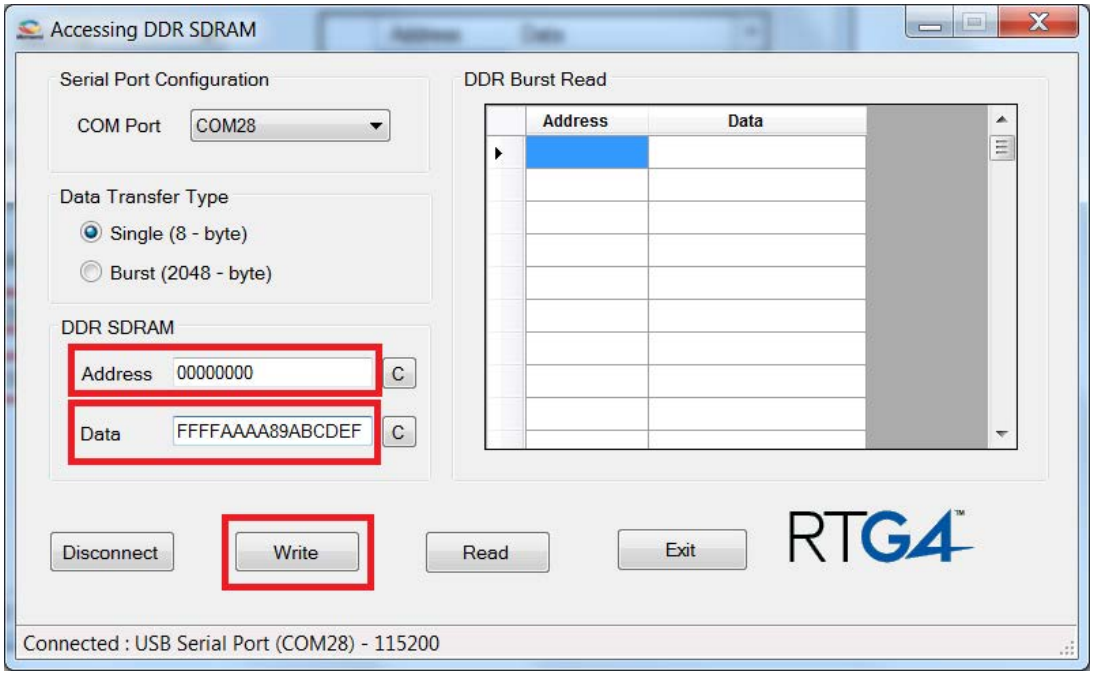
2.12.2 Performing a Single Data Transfer

For a single write or read operation, the AXI master logic is configured to transfer a burst length of 1 (that is, 8 bytes). For a write operation, the utility sends a 32-bit address and 64-bit (8 bytes) data. The data is then written to the DDR3 SDRAM. For a read operation, the utility sends a 32-bit address and receives 64-bit data from DDR3 and is displayed in the utility.

The following steps describe how to perform a single data transfer:

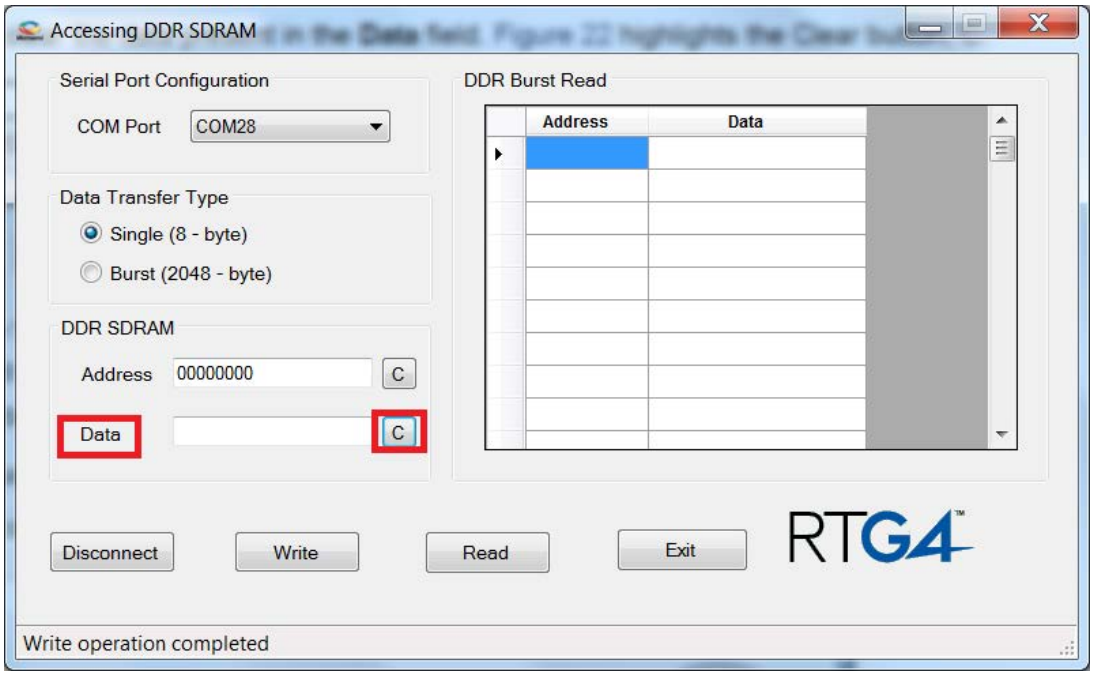
1. Select **Single (8-bytes)** as **Data Transfer Type**.
2. A 64-bit aligned address is required in the address field. Enter a 32-bit HEX Address in the range 0x00000000 - 0x03FFFFF8. When a non 64-bit aligned address is provided, the GUI converts it to a 64-bit aligned address and performs the write or read. See "Appendix 5: Performing Write or Read Operation when Non 64-bit Aligned Address is Provided" on page 33 to perform write or read operation when non 64-bit aligned address is provided.
3. In the **Data** field, enter 64-bit data in HEX format.
4. Click **Write**. The entered data is written to the DDR3 memory. Figure 19 shows the **Address** and **Data** values entered for a Single Write operation.

Figure 19 • **Single Write Operation**



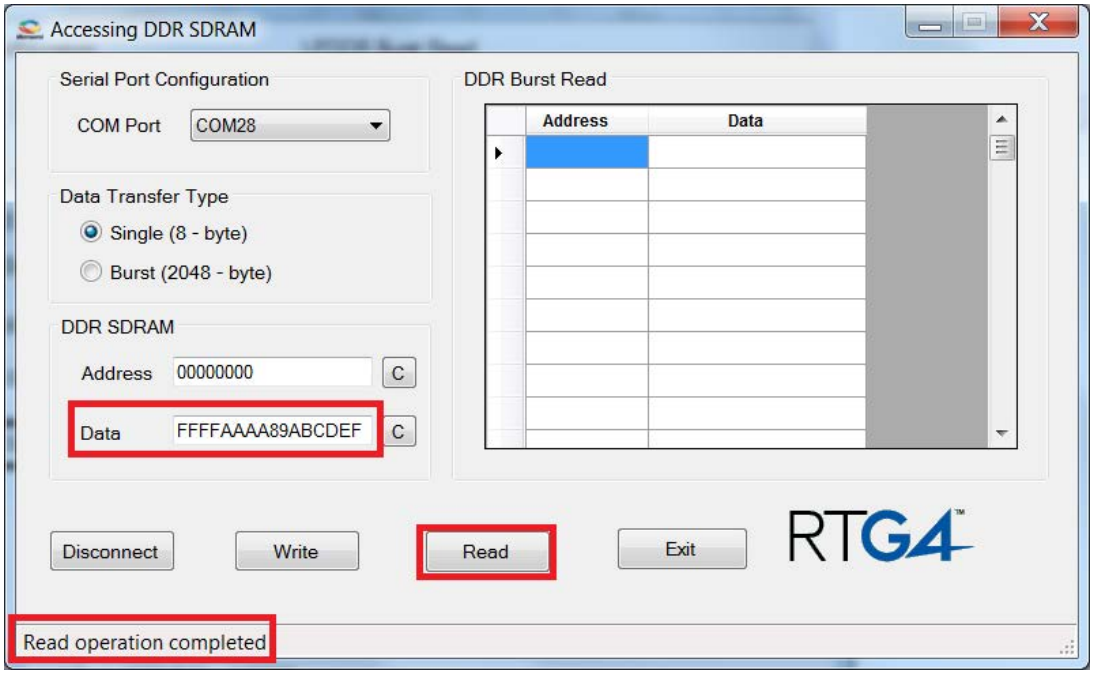
5. To verify the write operation, perform a read operation to the same address where the data is written.
6. Press **C** to clear the data present in the **Data** field. Figure 20 highlights the Clear button, C.

Figure 20 • **Clear Data Field**



7. Click **Read** to read the data from the DDR3 SDRAM. Figure 21 shows the data read from the DDR3 SDRAM.

Figure 21 • **Single Read Operation**



8. Compare the read and write data. The write and read data being the same establishes that the write and read operations to the DDR3 SDRAM are successful.

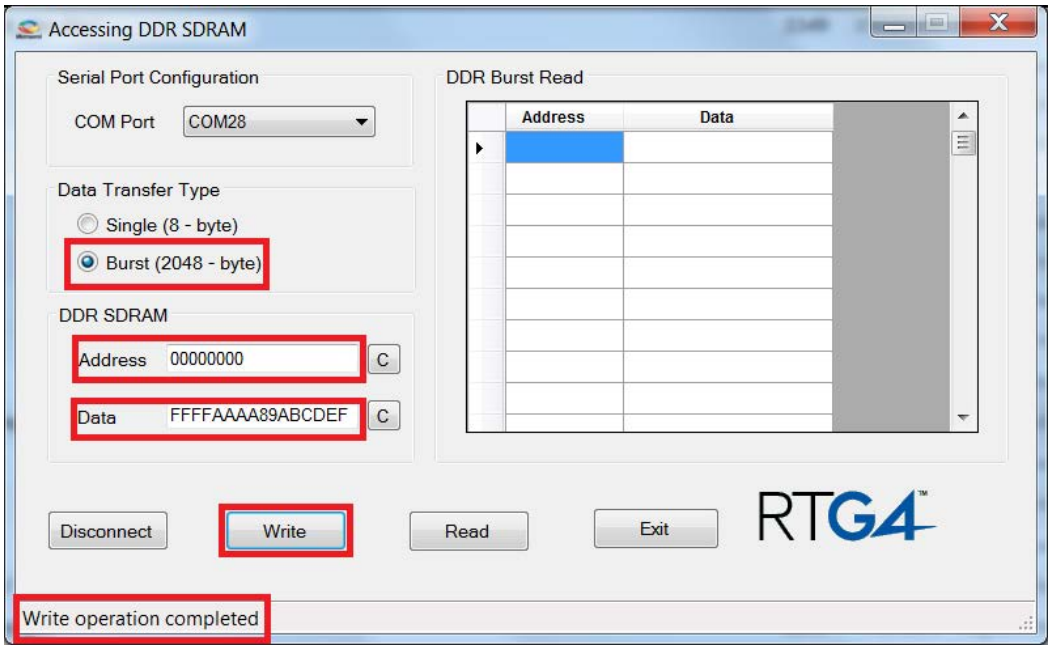
2.12.3 Performing Burst Data Transfer

For a burst write or read operation, the AXI master logic is configured to transfer a burst length of 16 (that is, 128 bytes). In this demo, 16 transfers of 16-beat burst operations are implemented (16 transfers x 16-beat burst data = 2048 bytes data). For a write operation, the utility sends a 32-bit initial address and 64-bit (8 bytes) initial data. After the initial write operation, incremental data is written. For a read operation, the utility sends a 32-bit address and receives 2048 bytes of data from the DDR3 SDRAM and the data is displayed in the utility.

The following steps describe how to perform a burst data transfer:

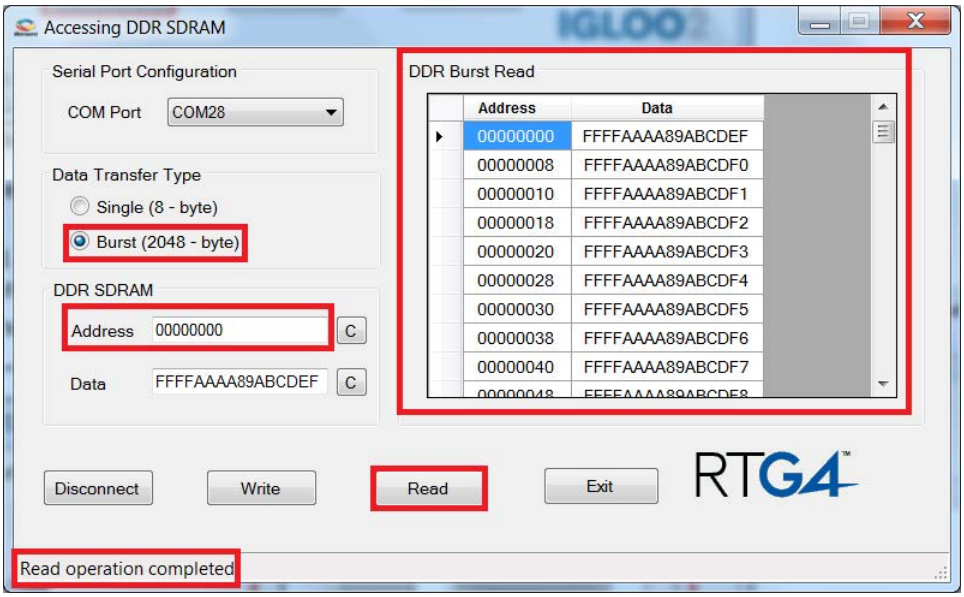
1. Select **Burst (2048-bytes)** as **Data Transfer Type**.
2. A 64-bit aligned address is required in the address field. Enter a 32-bit HEX Address in the range 0x00000000 - 0x03FFF7F8. When a non 64-bit aligned address is provided, the GUI converts it into a 64-bit aligned address and performs the write or read operation. Refer to "Appendix 5: Performing Write or Read Operation when Non 64-bit Aligned Address is Provided" on page 33 to perform write or read operation when non 64-bit aligned address is provided.
3. In the **Data** field, enter 64-bit data in HEX format.
4. Click **Write**. The entered data is written to the address location specified in the Address field and then the data is incremented by 1 and written to the next address location. This is repeated 256 times to write all the 2048 bytes of data. Figure 22 shows the **Address** and **Data** values entered for a Burst Write operation.

Figure 22 • **Burst Write Operation**



5. To verify the write operation, perform a read operation to the same address where the data is written.
6. Click **Read**. All the 2048 bytes of data written to the DDR3 SDRAM is read, and the read data is displayed on the **DDR Burst Read** panel. Figure 23 shows the burst read data.

Figure 23 • **Burst Read Operation**



7. Click **Exit** to exit the utility.

2.13 Conclusion

This demo shows how to perform read or write operations to DDR3 SDRAM using RTG4 FDDR. Options are provided to simulate the design using a testbench and validate the design on the RTG4 Development Kit using a GUI interface.

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.

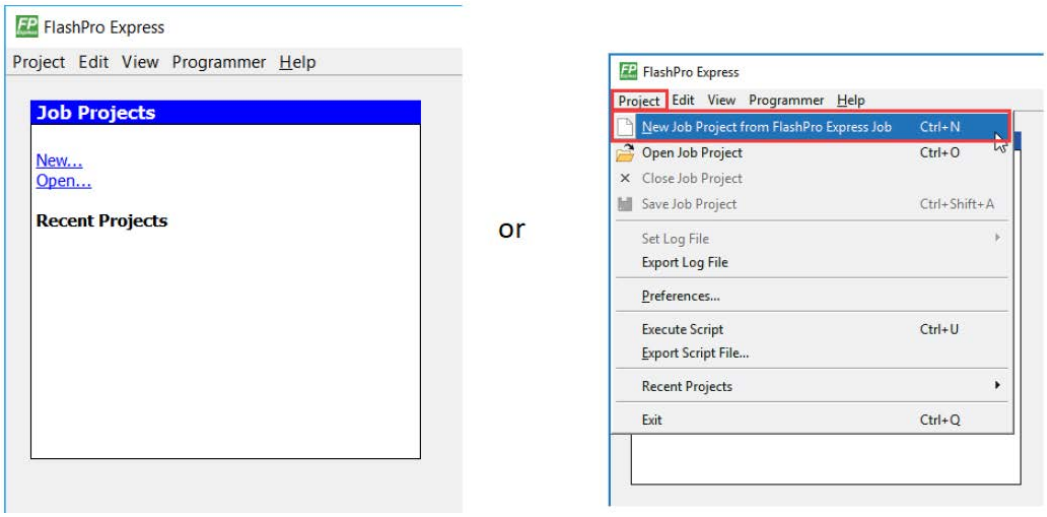
To program the device, perform the following steps:

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 Figure 24.

Figure 24 • **FlashPro Express Job Project**



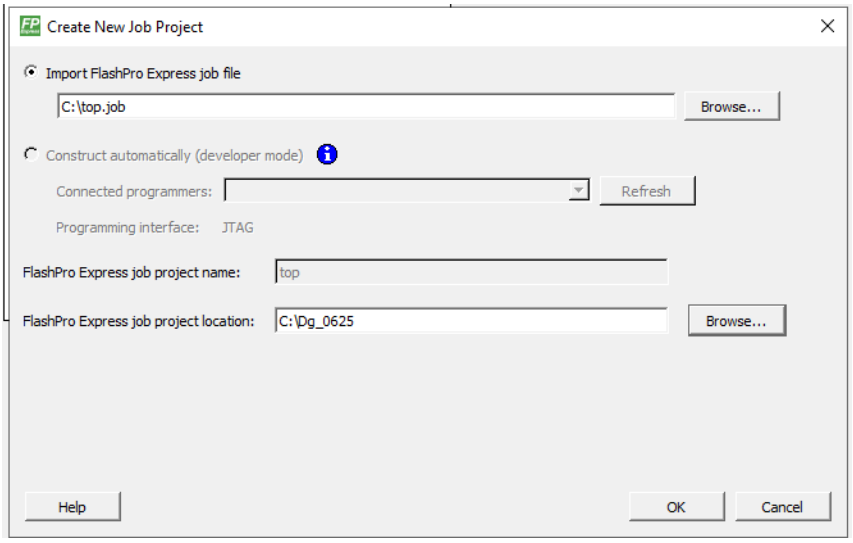
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

- **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_dg0625_df/Programming_File/top.job`

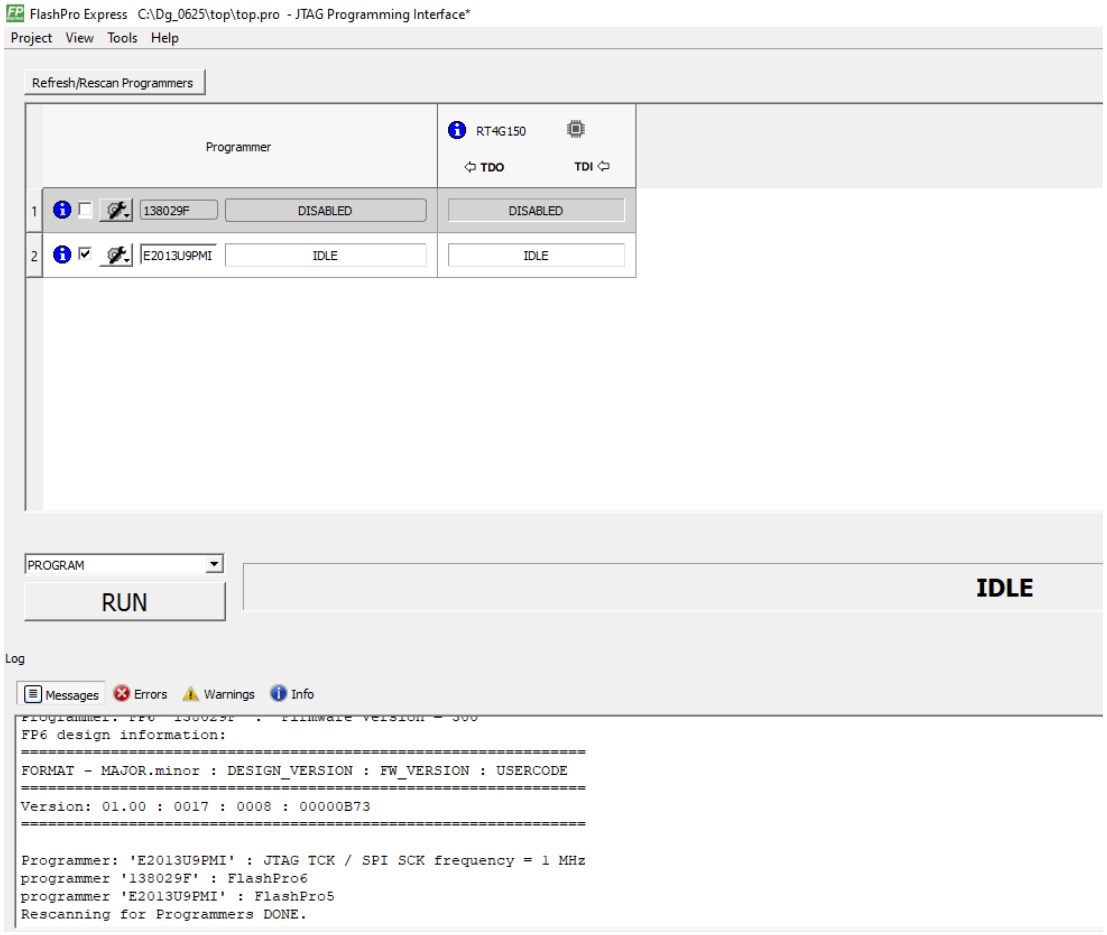
- **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 25 • **New Job Project from FlashPro Express Job**



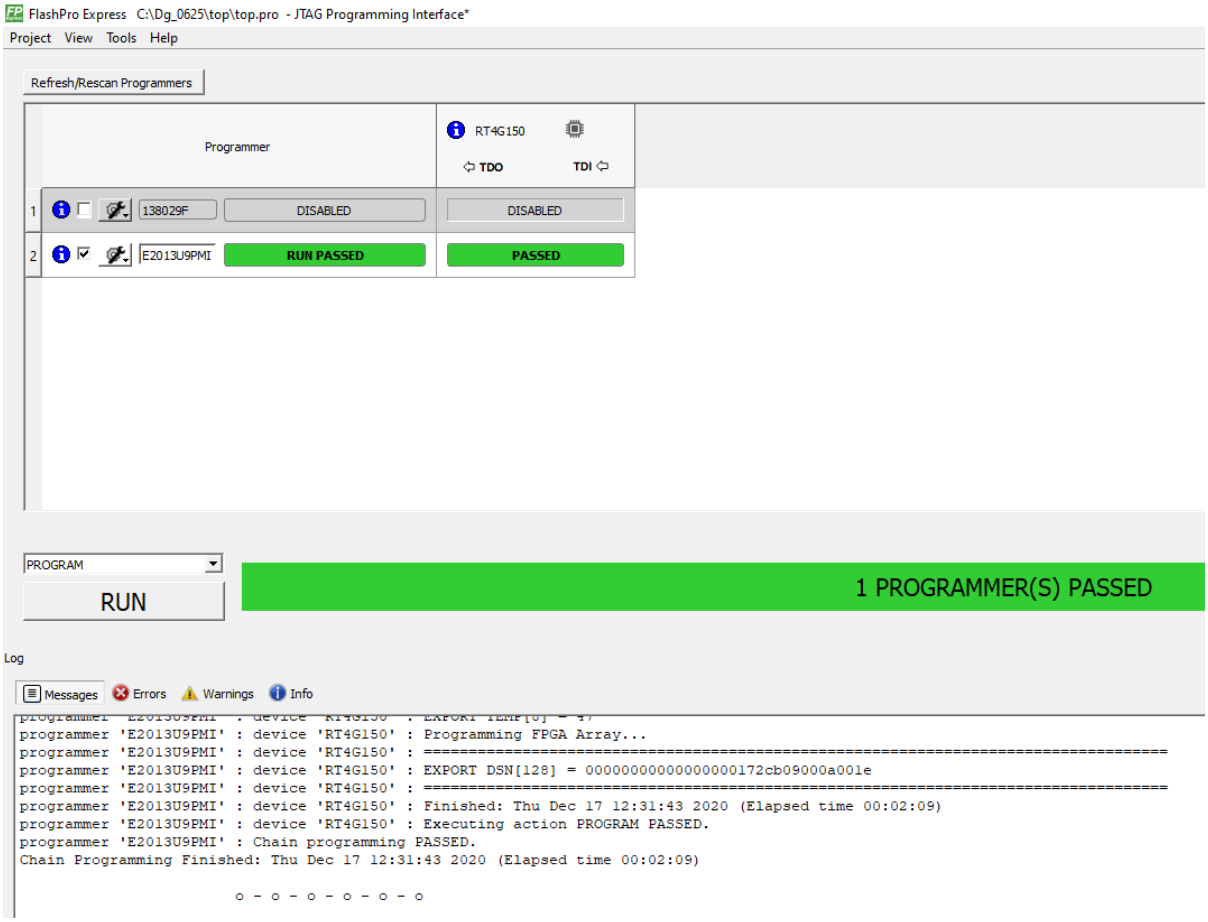
9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown Figure 26. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan** Programmers.

Figure 26 • **Programming the Device**



11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in Figure 27.

Figure 27 • **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_dg0625_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.

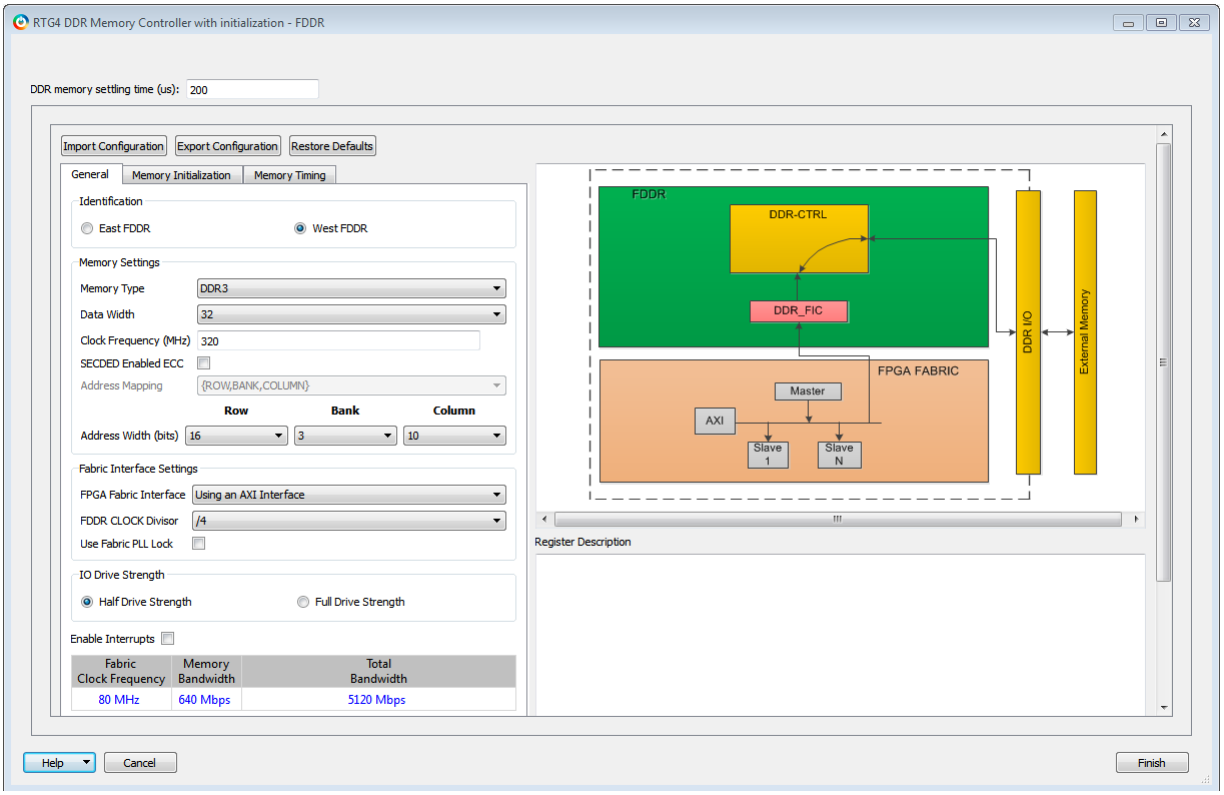
5 Appendix 3: RTG4 DDR Memory Controller Configuration and Initialization

This section describes how to configure and initialize the RTG4 DDR memory controller.

Launch **RTG4 DDR Memory Controller with Initialization** configurator using the **RTG4 DDR Memory Controller with initialization** SgCore in Libero. Use the RTG4 DDR Memory Controller Configurator to configure the FDDR, select its datapath bus interface (AXI or AMBA® high-performance bus (AHB)), and select the DDR clock frequency as well as the fabric datapath clock frequency.

1. In the **RTG4 DDR Memory Controller with Initialization** configurator, under the **General** tab, configure the settings as shown in Figure 28.

Figure 28 • **RTG4 FDDR Configuration Window**



2. Set the register values for the FDDR registers to match the external DDR memory characteristics. The FDDR has a set of registers that must be configured at runtime. The configuration values for these registers represent different parameters. Use the **Memory Initialization** tab, as shown in Figure 29 and the **Memory Timing** tab, as shown in Figure 30, to enter parameters that correspond to the DDR Memory and application. Consult the DDR Memory vendor's datasheet for values to enter in the **Memory Initialization** and **Memory Timing** tabs. Values entered in these tabs are automatically translated to the appropriate register values. When a specific parameter is clicked, a brief description of the corresponding register is described in the register description window of the **Fabric External Memory FDDR Configurator**. For more details on FDDR configuration registers, refer to *UG0573: RTG4 FPGA High Speed DDR Interfaces User Guide*.

Figure 29 • **FDDR Configurator - Memory Initialization tab**

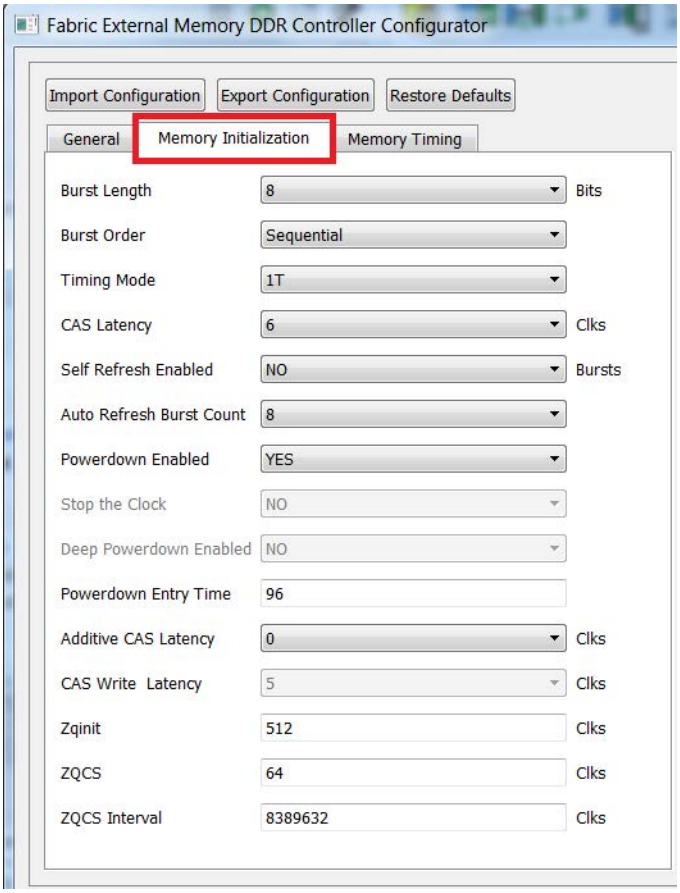
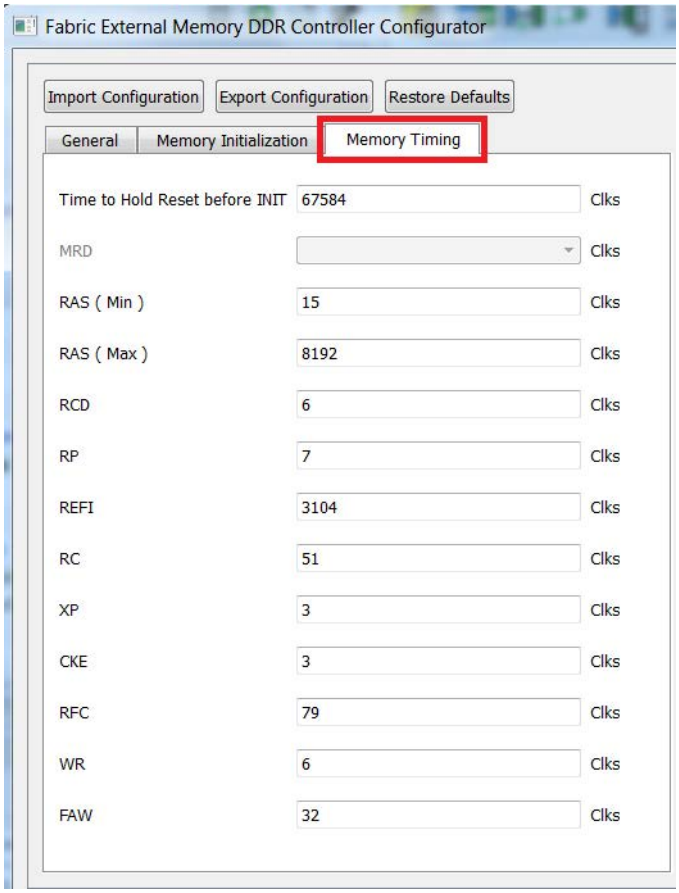


Figure 30 • **FDDR Configurator - Memory Timing tab**



3. Instantiate the FDDR as part of a user application and make the datapath connections.

5.1 Importing DDR Configuration Files

In addition to entering DDR Memory parameters using the Memory Initialization and Timing tabs, the DDR register values can be imported from a file. Click **Import Configuration** and navigate to the text file with the DDR register names and values. The DDR3 configuration file for the Micron memory available on the RTG4 Development Kit is provided along with the design files.

Note: If the register values are imported and not entered using the GUI, then all the required register values must be specified.

5.2 FDDR Initialization

The RTG4 DDR Memory Controller with initialization Sgcore has a built-in initialization state machine. On the assertion or de-assertion of the INIT_RESET_N (Active Low) signal, the FDDR block is initialized with the user configurations. When the configuration phase is complete, the INIT_DONE signal is asserted and the FDDR block is ready for normal operations. The FDDR initialization can start automatically at power up by connecting the INIT_RESET_N (Active Low) input of the FDDR block to the POWER_ON_RESET_N (Active Low) signal of the SYSRESET macro.

The clock used for initialization must be a 50 MHz clock which is connected to the INIT_CLK 50 MHz signal.

5.3 DDR Memory Settling Time

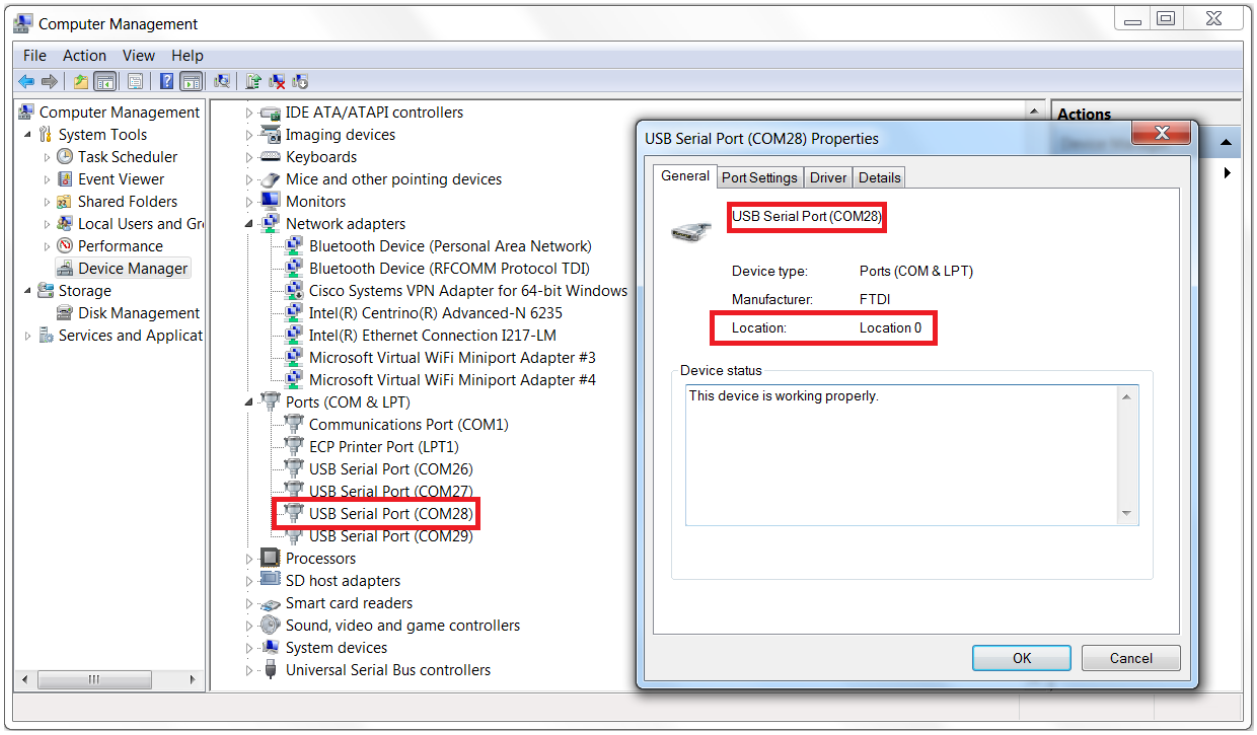
The RTG4 DDR memory controller block is hard-coded with a DDR memory settling time of 200 µs, assuming that the clock period of INIT_CLK is 20 ns (frequency 50 MHz). Microsemi recommends that the initialization frequency be kept at 50 MHz.

Refer to the DDR Memory vendor's datasheet for the correct memory settling time to use. An incorrect memory settling time may result in the failure of the DDR memory initialization, during operation. If a different memory settling time is required for the DDR memory or a different INIT_CLK frequency is chosen by the user, than the recommended 50 MHz, then the program code must be edited in the Program tab of **CoreABC** to change the load value of the register used to compute the settling time.

6 Appendix 4: Finding Correct COM Port Number when Using USB 3.0

FTDI USB to UART converter enumerates the four COM ports. In USB 3.0, the four available COM ports are in **Location 0**. Figure 31 shows the USB 3.0 Serial port properties.

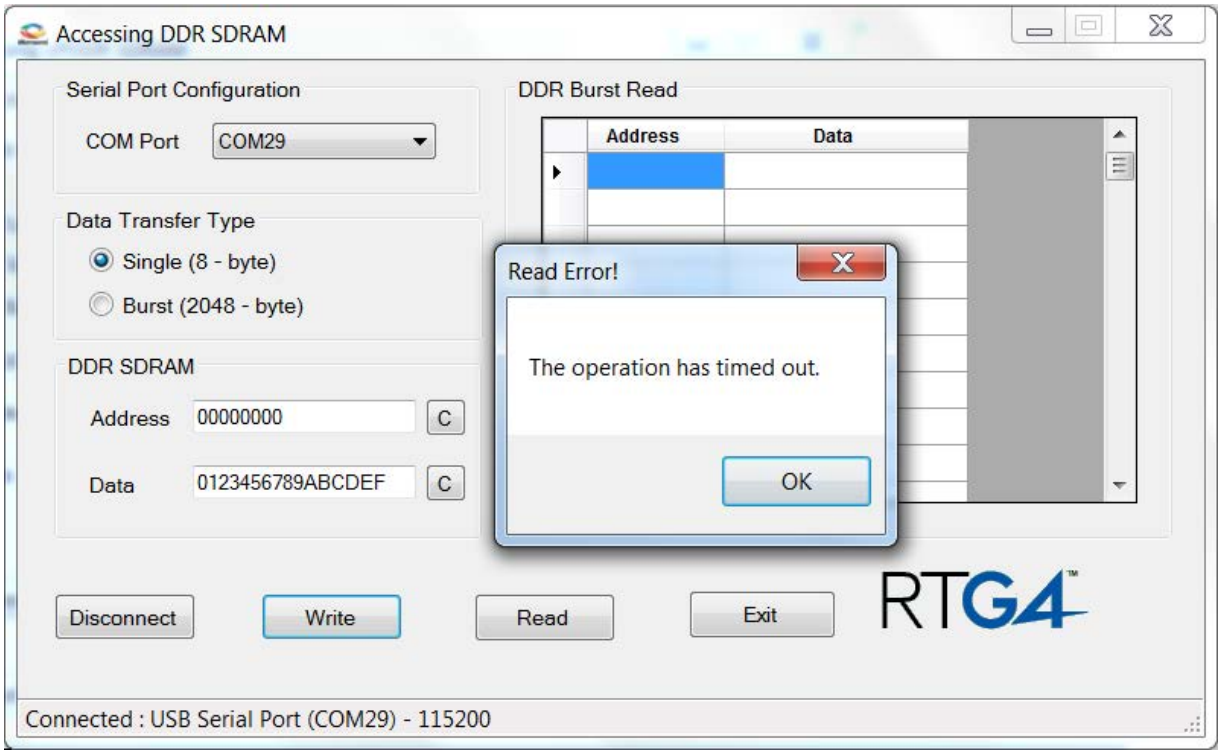
Figure 31 • **USB 3.0 Serial Port Properties**



The following steps describe how to find out the correct COM port:

1. Program the RTG4 Development Kit with the provided programming file.
2. Connect each available COM port and click **Write**.
   If a wrong COM port is selected, the GUI displays the **Read Error** message.
3. Try with all four available COM ports until this message disappears.
   Figure 32 shows the **Read Error** message.

Figure 32 • **Read Error**

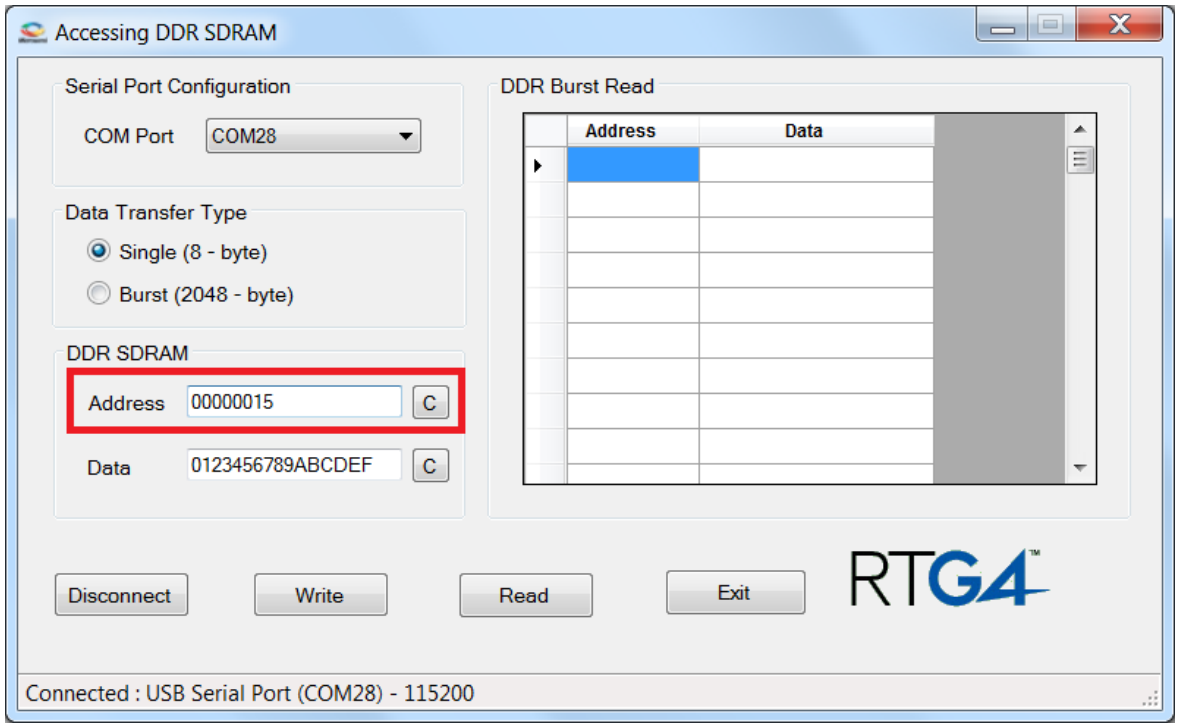


7 Appendix 5: Performing Write or Read Operation when Non 64-bit Aligned Address is Provided

When a non 64-bit aligned address is provided in the GUI, the GUI converts it into the 64-bit aligned address (0, 8, 10, 18, 20, 28, 30, 38 …) and performs the write or read operation.

1. Enter the non 64-bit aligned 32-bit address in HEX format.
2. Enter the 64-bit data in HEX format. Figure 33 shows the non 64-bit aligned **Address** entered in the GUI.

Figure 33 • **Non 64-bit Aligned Address**



3. Click **Write** to perform a write operation. GUI converts the address into a 64-bit aligned address and performs the write operation. Figure 34 shows the GUI pop-up information message and converted 64-bit aligned address.

Figure 34 • **Converted 64-bit Aligned Address**