# AC454 Application Note RTG4 SRAM Initialization After Power-up Using µPROM





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

©2021 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners. Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document or to any products and services at any time without notice.

#### **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.



# Contents

1	1.1 1.2 1.3 1.4 1.5	Dn History Revision 6.0 Revision 5.0 Revision 4.0 Revision 3.0 Revision 2.0	1 1 1 1
2	RTG4	Revision 1.0       SRAM Initialization After Power-up Using µPROM         Design Requirements       Design Requirements	2
	2.3 2.4 2.5	Prerequisites Embedded SRAM and µPROM Blocks Data Storage in RTG4 µPROM Design Description 2.5.1 Fabric APB Master	4 4 5
	2.6 2.7 2.8 2.9 2.10 2.11	2.5.2       CoreµPROMIF_APB         Hardware Implementation       Clocking Structure         Reset Structure       1         Simulating the Design       1         Setting Up the Demo Design       1         Running the Design       1         Conclusion       1	9 9 0 2 3
3		dix 1: Programming the Device Using FlashPro Express	
4	Append	dix 2: Running the TCL Script1	7
5	Append	dix 3: Design Files	8
6	Append	dix 4: Customizing RAM Wrapper Interface1	9
7	Append	dix 5: How to Reset RAM Block Contents Using µPROM	0



# **Figures**

Figure 1	Top-Level Block Diagram	. 3
Figure 2	µPROM Memory Configurator GUI	
Figure 3	APB3 State Diagram	. 6
Figure 4	Fabric Master State Diagram	. 7
Figure 5	CoreµPROMIF_APB and RTG4µPROM SmartDesign Connection	. 8
Figure 6	SmartDesign Top-Level Diagram	. 9
Figure 7	Clocking Structure	10
Figure 8	Reset Structure	10
Figure 9	Waveform of RTG4 LSRAM Initialization Using µPROM	11
Figure 10	TPSRAM Write Data and Write Address	11
Figure 11	TPSRAM Read Data and Read Address	12
Figure 12	RTG4 Development Kit Board	12
Figure 13	SmartDebug LSRAM Read Data	13
Figure 14	FlashPro Express Job Project	14
Figure 15	New Job Project from FlashPro Express Job	15
Figure 16	Programming the Device	15
Figure 17	FlashPro Express—RUN PASSED	16
Figure 18	APB Master Wrapper SmartDesign	19
Figure 19	Catalog Window	20
Figure 20	Sample SmartDesign Block	20
Figure 21	uPROM Configurator	21
Figure 22	uPROM Configurator—Initialization client	21
Figure 23	SmartDebug Window	22
Figure 24	Debug FPGA Array Window	23
Figure 25	RAM Contents Reset to Zero	23





Table 1	Design Requirements	3
	SRAM and µPROM Blocks in the RT4G150 Device	
Table 3	Fabric APB Master Interface Signals	8



# 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 6.0

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

- Updated the design and document for Libero SoC v2021.2.
- Updated Figure 1, page 3.
- Updated Hardware Implementation, page 9.
- Updated Clocking Structure, page 9.
- Updated Reset Structure, page 10.

### 1.2 **Revision 5.0**

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

- Added Setting Up the Demo Design, page 12.
- Added Appendix 1: Programming the Device Using FlashPro Express, page 14.
- Added Appendix 2: Running the TCL Script, page 17.
- Removed the references to Libero version numbers.

### 1.3 Revision 4.0

The document was updated for Libero SoC v11.9 SP1.

### 1.4 Revision 3.0

The document was updated for Libero SoC v11.8 SP2.

### 1.5 Revision 2.0

The procedure to reset RAM block contents using µPROM was added. For more information, refer to Appendix 5: How to Reset RAM Block Contents Using µPROM, page 20.

### 1.6 Revision 1.0

The first publication of this document.



# 2 RTG4 SRAM Initialization After Power-up Using µPROM

This application note describes how to initialize the static random access memory (SRAM) blocks of Microsemi RTG4<sup>™</sup> field programmable gate array (FPGA) with user data after power-up. The design for this application note uses a large SRAM (LSRAM) block, which is initialized by an FPGA fabric master through the Advanced Microcontroller Bus Architecture Advanced Peripheral Bus interface (AMBA APB bus).

RTG4 FPGA devices have embedded SRAM blocks (LSRAM and micro SRAM ( $\mu$ SRAM)) in the fabric. Both LSRAM and  $\mu$ SRAM blocks are placed in multiple rows within the FPGA fabric, and they can be accessed through the fabric routing architecture. Table 2, page 4 lists the number of LSRAM,  $\mu$ SRAM, and micro programmable read-only memory ( $\mu$ PROM) blocks available in the RT4G150 device.

LSRAMs are used for larger data storage (up to 24,576 bits), whereas  $\mu$ SRAMs are used for smaller data storage (up to 1536 bits). Both LSRAM and  $\mu$ SRAM are volatile. As a result, data is lost after the device power-down. After power-up, the state of SRAM is unknown.

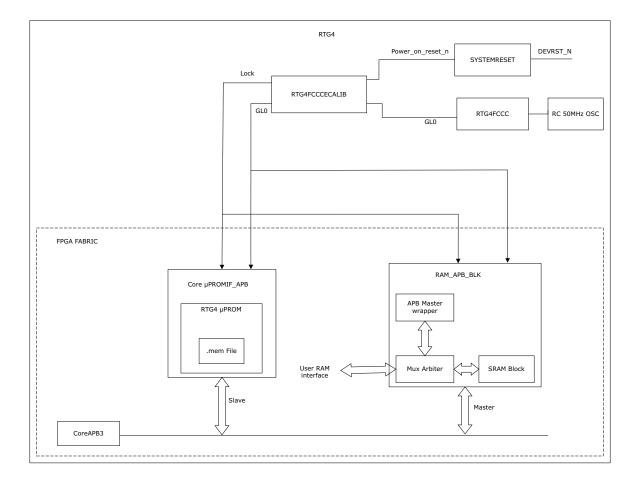
 $\mu$ PROM can be used to store programmable data for initializing the LSRAM and  $\mu$ SRAM blocks.  $\mu$ PROM cells are located at the bottom of the FPGA fabric, and they can be accessed through the fabric interface. This application note implements a design that uses  $\mu$ PROM to initialize the LSRAM block.  $\mu$ PROM 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. For more information about  $\mu$ PROM features and architecture, refer to the  $\mu$ PROM section in UG0574: RTG4 FPGA Fabric User Guide.

Figure 1, page 3 shows the top-level block diagram of the design used in this application note.

The LSRAM initialization data (.mem file) is stored in the  $\mu$ PROM during programming. The .mem file is a data storage client that contains the  $\mu$ PROM memory content. The fabric APB bus master wrapper reads the  $\mu$ PROM and stores the initialization data in LSRAM. The Core $\mu$ PROMIF\_APB IP core implements the APB bus slave wrapper logic to provide read-only access to the  $\mu$ PROM. This design is simulated and validated using the RTG4 Development Kit.



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



## 2.1 Design Requirements

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

#### Table 1 • Design Requirements

Requirement	Version						
Hardware							
RTG4 Development Kit	Rev B or later						
<ul> <li>12 V adapter (provided with the kit)</li> </ul>							
Host PC or laptop	64-bit Windows 7 and 10						
Software Requirements							
Libero <sup>®</sup> System-on-Chip (SoC)	Note: Refer to the readme.txt file provided in						
FlashPro Express	the design files for the software versions used with this reference design.						
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:

- 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
- For demo design files download link: http://soc.microsemi.com/download/rsc/?f=rtg4\_ac454\_df

## 2.3 Embedded SRAM and µPROM Blocks

The following table lists the number of SRAM blocks (including LSRAM and  $\mu$ SRAM) and  $\mu$ PROM blocks available in the RT4G150 device.

Table 2 • S	SRAM and µPROM	Blocks in the	RT4G150 Device
-------------	----------------	---------------	----------------

Туре	Number of blocks in RT4G150
LSRAM 24.5 Kb blocks	209
µSRAM 1.5 Kb blocks	210
µPROM 381 Kb block	1

LSRAM is configured in a two-port mode in this design. One port is dedicated for write operations and the other for read operations. The read and write operations are synchronous and require a clock edge. LSRAM supports both pipelined read and non-pipelined read (flow-through) operations. µSRAM blocks have two read data ports (port A and port B) and one write data port (port C). Read operations are executed in synchronous and asynchronous modes. Write operation is performed only in synchronous mode.

µPROM supports only read operation during normal device operation. Read operation is performed using the fabric interface in synchronous mode only.

For more information about the features and use models of LSRAM, µSRAM, and µPROM, refer to UG0574: RTG4 FPGA Fabric User Guide.

### 2.4 Data Storage in RTG4 µPROM

This design uses the  $\mu$ PROM memory content for initializing the LSRAM block after power-up. Core $\mu$ PROMIF\_APB IP provides easy access to  $\mu$ PROM memory for APB bus masters. It performs address translation, allowing APB bus masters to address  $\mu$ PROM using word-aligned addressing. To access  $\mu$ PROM memory using Core $\mu$ PROMIF\_APB, RTG4 $\mu$ PROM core is instantiated along with Core $\mu$ PROMIF\_APB in SmartDesign. These cores are available in the Libero SoC Catalog. For more information about instantiating Core $\mu$ PROMIF\_APB and mapping RTG4 $\mu$ PROM, refer to the System Integration section in CoreUPROMIF\_APB\_HB.pdf.

In this design, a data storage client is created to store LSRAM initialization data. It is configured to store 36-bit words in 64 locations during the configuration of  $\mu$ PROM. The data storage client file is provided with the design file (refer to Appendix 3: Design Files, page 18). The RTG4 $\mu$ PROM core memory configurator GUI in Libero SoC is used for creating multiple data storage clients. The following figure shows how to create a single data storage client for this design. To allow  $\mu$ PROM content for simulation, select the **Use content for simulation** check box.



Figure 2 • µPROM Memory Configurator GUI

Edit Data Storage Client ? ×
Client name: test1
Content from file:
Format: Microsemi Binary 💌
C Content filled with 0s
Start address: 0x 0 📩
Number of 36-bit words: 64 Decimal
$\label{eq:linear} Imported \ Memory \ file \ location \ is \ E:\ Projects\ Updates\ 11_9\ t4g\_ac454\_liberov\ 11p9\_df\ Source \ Files$
✓ Use content for simulation
Help   Cancel

## 2.5 Design Description

This design includes:

- Fabric APB Master, page 5
- CoreµPROMIF\_APB, page 8

### 2.5.1 Fabric APB Master

The fabric master performs the following functions:

- 1. Acts as AMBA Advanced Peripheral Bus 3 (APB3) bus interface that reads data from the µPROM memory using CoreµPROMIF\_APB slave interface.
- 2. Loads data into LSRAM via the APB bus I/F.

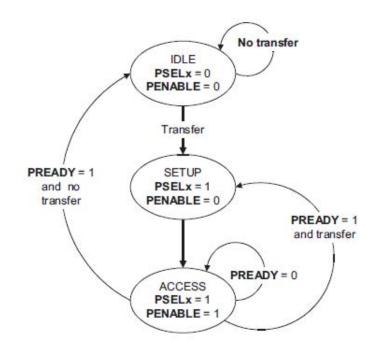
The state machine operates through the following states of an APB bus cycle:

- IDLE: This is the default state of the APB bus.
- SETUP: When a transfer is required the state machine moves to this state. In this state, the required PSEL signal is asserted. The APB bus remains in this state for one clock cycle and moves to the ACCESS state on the rising edge of PCLK.
- ACCESS: The PENABLE signal is asserted in this state. The address, write, and select signals are
  asserted and must remain stable during the transition from SETUP to ACCESS state. The PREADY
  signal controls the exit from this state in the following ways:
  - If PREADY is held low by the slave, the state machine remains in this state.
  - If PREADY is held high by the slave, this state is exited and moved back to the IDLE state if no more transfers happen. If another transition happens, the state machine moves to the SETUP state.



The following illustration shows the states in the APB bus cycle.

#### Figure 3 • APB3 State Diagram



After the fabric master interface reads data from µPROM memory and loads that data into fabric LSRAM, it asserts a switch flag (init\_done) to the mux arbiter block. This block lets the LSRAM ports to be used for initialization on design startup and then releases them for user access once initialization is complete.

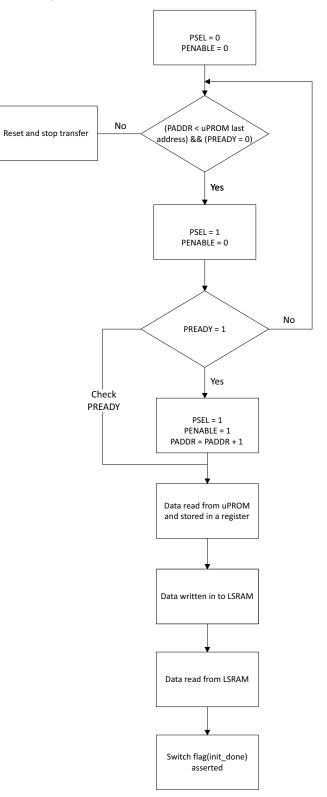
The LSRAM block is configured as a two-port with a depth of 512 and a width of 36. The APB3 master wrapper logic generates the required read and write operations for LSRAM using the PREADY signal from the slave interface. The PREADY signal is also used for inserting wait states.

The APB fabric master interface generates address and controls the signals on the bus after the rising edge of PCLK. If PREADY is HIGH, the APB master enters the data phase to perform a read or write operation. During the data phase, if PREADY is LOW, then the APB slave extends the data phase. The APB fabric master must hold the data throughout extended cycles. The APB master will only read and write the APB slaves when PREADY is HIGH.



The following illustration shows the states of the fabric master interface.

#### Figure 4 • Fabric Master State Diagram





### 2.5.1.1 Fabric APB Master Interface Description

The following table lists the fabric APB master interface signals.

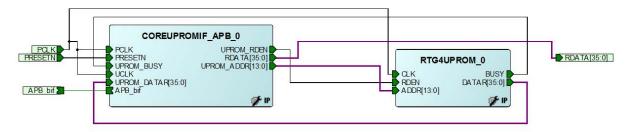
#### Table 3 • Fabric APB Master Interface Signals

Signal	Direction	Description
PCLK	Input	APB bus clock
PRESETn	Input	APB bus active low reset
PRDATA[31:0]	Input	APB bus input 32 bit read data
RDATA[35:0]	Input	36 bit µPROM read data
PREADY	Input	APB bus ready signal
PSLVERR	Input	APB bus error reporting signal
PWRITE	Output	APB bus write access signal
PWDATA[31:0]	Output	APB bus 32 bit wide write data
PENABLE	Output	APB bus enable signal
PSEL	Output	APB bus slave select signal
PADDR[15:2]	Output	APB bus slave address signal
Mem_data_out[35:0]	Output	Output data for LSRAM
rd_en	Output	LSRAM read enable signal
wr_en	Output	LSRAM write enable signal
raddr[8:0]	Output	LSRAM read address signal
waddr[8:0]	Output	LSRAM write address signal
Init_done	Output	Initialization done signal

### 2.5.2 CoreµPROMIF\_APB

RTG4 µPROM is accessed using RTG4 µPROM core or by using CoreµPROMIF\_APB IP core. The CoreµPROMIF\_APB IP core makes the RTG4 µPROM block appear as a transparent memory on the APB bus interface. In this design, the CoreµPROMIF\_APB IP core is used to perform address translation to allow APB bus masters to directly address µPROM using word aligned addressing. The CoreµPROMIF\_APB IP core also prevents reading an invalid address space in µPROM. The CoreµPROMIF\_APB IP core must be instantiated along with the RTG4µPROM core and connected as shown in the following figure.

#### Figure 5 • CoreµPROMIF\_APB and RTG4µPROM SmartDesign Connection



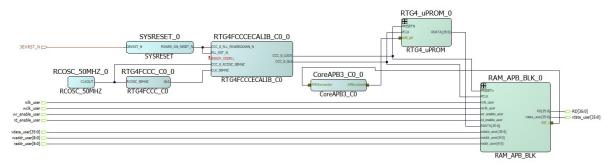
For more information about other features and advantages of using CoreµPROMIF\_APB, refer to *CoreUPROMIF\_APB\_HB.pdf*.



## 2.6 Hardware Implementation

This design uses the Sysreset signal, a 50 MHz RC oscillator, the RTG4 fabric clock conditioning circuit (FCCC), the RAM\_APB\_BLK\_0 block (LSRAM block with master wrapper), and the CoreµPROMIF\_APB and RTG4µPROM IP cores, as shown in the following figure. The IP cores, along with the LSRAM wrapper, are used to initialize the fabric SRAM by moving data from µPROM to fabric LSRAM via the APB interface. This design uses the 50 MHz RC oscillator as a reference clock for the fabric CCC. The fabric CCC is connected to RTG4FCCCECALIB which generates a 30 MHz clock, which is used as the system clock.

#### Figure 6 • SmartDesign Top-Level Diagram



The RAM\_APB\_BLK\_0 module contains LSRAM as a two-port memory with the depth and width configured as 512 × 36. This module can be modified to initialize LSRAM blocks configured in any of the various aspect ratios and operating modes supported by RTG4. For more information about using variations of LSRAM, refer to Appendix 4: Customizing RAM Wrapper Interface, page 19.

The RTG4 $\mu$ PROM core imposes a maximum frequency constraint of 30 MHz on the  $\mu$ PROM clock. Core $\mu$ PROMIF\_APB has pre-scalar clock logic that generates  $\mu$ PROM clock from PCLK frequency with the condition of  $\mu$ PROM clock frequency not exceeding 30 MHz. In this design, the  $\mu$ PROM read operation frequency is 30 MHz.

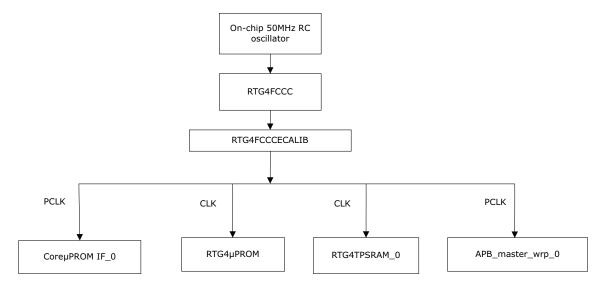
For more information about operating µPROM at a clock frequency other than 30 MHz, refer to the *Design Constraints* section in *CoreUPROMIF\_APB\_HB.pdf*.

## 2.7 Clocking Structure

The on-chip 50 MHz oscillator gives the reference frequency to RTG4FCCC\_0, which is connected to RTG4FCCCECALIB generates a 30 MHz clock (GL0) and drives COREUPROMIF\_0, RTG4UPROM, RTG4TPSRAM\_0, and APB\_master\_wrp\_0 blocks. The following figure shows the reset structure of the design.



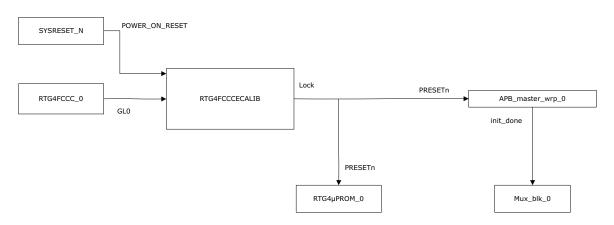
#### Figure 7 • Clocking Structure



### 2.8 Reset Structure

The POWER\_ON\_RESET and GL0 signals are given to RTG4FCCCECALIB core and the output signal is used to reset the RTG4uPROM\_0 and APB\_master\_wrp\_0 block. After reset, the APB\_master\_wrp\_0 block generates the init\_done signal to reset the mux\_blk\_0. The following figure shows the reset structure of the design.

#### Figure 8 • Reset Structure



## 2.9 Simulating the Design

The design files mentioned in Appendix 3: Design Files, page 18 include testbench files that are required for simulating the design. As shown in the following figure, data is read from  $\mu$ PROM using APB interface and loaded to LSRAM. This data can be seen using the RD[35:0] output signal of the TPSRAM block (highlighted in the figure).



Figure 9 • Waveform of RTG4 LSRAM Initialization Using µPROM

THE LANSAUL         000           THE LANSAUL BURKED MID. ARTICLENESS (END ADDRES)         000           THE LANSAUL BURKED MID. ARTICLENESS (END ADDRES)         0           THE LANSAUL BURKED MID. ARTICLENESS (END ADDRES)         0           THE LANSAUL BURKED MID. ARTICLENESS (END ADDRES)         0           THE LANSAUL BURKED MID.	•	. M	aus -
Projuginos         Projugi		0000	
		1	
		+	
implication busined and and an analysis of the second and and an analysis of the second busined and and an analysis of the second and analysis of the second analysis of the second and analysis of the s		e e	
implication based as plants                  implication based as plants               implication based as plants                implication based as plants               implication based as plants                implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based as plants               implication based plants               implication bas p		e e	
		ő	
		00000000	
Apply and business bu		1	
		No Data-	uP8024 memory data
VPC/VPC/VPC         PVC/VPC/VPC         PVC/VPC/VPC         PVC/VPC/VPC/VPC/VPC/VPC/VPC/VPC/VPC/VPC/			
in the function basis of the sense determine of a bit sense of the sense			
A PU-Jacobi Bulkovi bo Statu Jacobi			
Pro-Juscie Bulkander Jasse Bulkander Jasse         Die Juscie Bulkander Jasse Bulkander Jasse Bulkander Jasse         Die Juscie Bulkander Jasse Bulkander Bu		000000000	
//psi/set/set/set/set/set/set/set/set/set/set			
Angle Jacket, Backet, Backet, Jacket, Jacket, Jacket, Backet, Jacket, Backet, B			
μ. μ. μ. μ. μ. μ. μ. μ. μ. μ			
μου ματοχ δριστού μου ζριστό μου οι ζειτό μου		000000000	
Visite Visite         Visite V		0	
J. Proj. Vision, Bulkarda, Iso, Davi, Vision, Bara, Janaka, Jana, Davi, Jana, D		000000000	
		0	
Angl. Jarding Busing Line (Jung Jung Jung Jung Jung Jung Jung Jung			
A registration (and two back) (and gas called material) and (and two back)         1           A registration (and two back) (and gas called material) (and two back)         00           A registration (and two back) (and gas called material) (and gas called material)         00           A registration (and two back) (and gas called material) (and gas called material)         00           A registration (and two back) (and gas called material) (and gas called material)         00           A registration (and two back) (and gas called material) (and gas called material)         00           A registration (and registration (and gas called material))         00           A registration (and registration (and gas called material))         00           A			
April (MR) Buildon (MR) (MR) (MR) (MR) (MR) (MR) (MR) (MR)			
		000	
// hote/set/set/set/set/set/set/set/set/set/s		000	
2224 воды		0	
/ ну числе зыяком но дели не да датоглязии дила: / ну числе зыяком но дели не да датоглязии дила: / ну числе зыяком но дели не да датоглязии дила: / ну числе зыяком но дели не да датоглязии дел: / ну числе зыяком но дели не да датоглязии дел: 000000000000000000000000000000000000		182	TRSPAN write operation
Instrument         Instrum	Ato4 UPROM ID/UPROM IDD O/RAM APB BLK O/RTG4TPSRAM O/WEN		inconstances de constances a constances de constances de constances de constances de constances de constances d
/ № 4 и и и и и и и и и и и и и и и и и и		000	
//pu_index_balandu iso (baka jing back)atteriminani (ban) 0 //pu_index_balandu iso (baka jing back)atteriminani (basa) 0 00 00		000000000	
		0	
		000	
TPSRAM read operation		xxxxxxxxxx	
TPSRAM read operation			
			TPSRAM read operation

The following figure shows the write data and write address of TPSRAM.

Figure 10 • TPSRAM Write Data and Write Address

A http4_UPROM_tb/UPROM_top_0/RAM_APB_BLK_0/RTG4TPSRAM_0/WCLK & http4_UPROM_tb/UPROM_top_0/RAM_APB_BLK_0/RTG4TPSRAM_0/WEN	0	whww	·www	huu	nnn	www	nnn	mm	mm	mm	mm	mhnn	nn
/tg4_uPROM_tb/uPROM_top_0/RAM_APB_BLK_0/RTG4TPSRAM_0/WADDR	03f	200	10	01	002	1003	004	(005	006	1007	1008	1009	100
j /rtg4_uPROM_tb/uPROM_top_0/RAM_APB_BLK_0/RTG4TPSRAM_0/WD	00000000f	000000000	10	00000001	000000002	000000003	000000004	000000005	(000000006	1000000007	100000008	00000009	100
				TPSRA	M write	data an	d addre						



The following figure shows the read data and read address of TPSRAM.

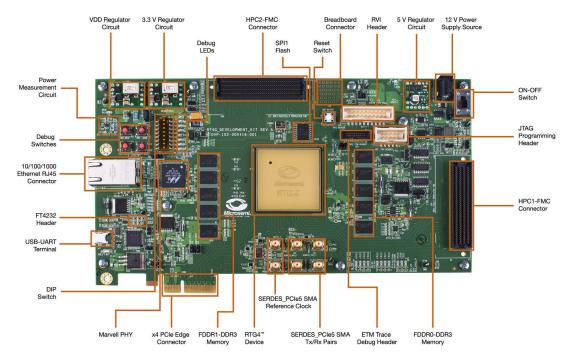
#### Figure 11 • TPSRAM Read Data and Read Address



## 2.10 Setting Up the Demo Design

The following figure shows the RTG4 Development Kit board.

#### Figure 12 • RTG4 Development Kit Board





# 2.11 Running the Design

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.

After the RTG4 Development kit is programmed successfully, open SmartDebug in Libero SoC and check for the LSRAM content. The LSRAM content must match with the memory file that is loaded into µPROM through data client configurator, as shown in the following figure.

For more information about running SmartDebug to view LSRAM memory block content, refer to *TU0530: SmartFusion2 and IGLOO2 SmartDebug Hardware Design Debug Tools Tutorial.* 

Figure 13 • SmartDebug LSRAM Read Data

femory Blocks:			v debug data bes Active	Probes Me	nory Blocks	Probe Insertion	a		uPRO	M conter	nt in LSRA	M							
RAM_APB_BLK_0/RTG4TPSRAM_	0/RAM_APB_BL	Current I Data bit I			K_O/RTG4TPSR	AM_0/RAM_APE	LBLK_RTG4TPS		sram_roco/# ation 1	IST_RAM1K18_R	T_IP	Locat	ion 2			Locatio	n 3		
		0000	000	000	000	000	001	000	000	000	002	000	000	000	003	000	000	000	1
		0010	004	000	000	000	005	000	000	000	006	000	000	000	007	000	000	000	1
		0020	008	000	000	000	009	000	000	000	00A	000	000	000	008	000	000	000	1
		0030	00C	000	000	000	000	000	000	000	OOE	000	000	000	OOF	000	000	000	
		0040	000	000	000	000	001	000	000	000	002	000	000	000	003	000	000	000	
		0050	004	000	000	000	005	000	000	000	006	000	000	000	007	000	000	000	
		0060	008	000	000	000	009	000	000	000	00A	000	000	000	008	000	000	000	
		0070	000	000	000	000	000	000	000	000	00E	000	000	000	OOF	000	000	000	
		0080	000	000	000	000	001	000	000	000	002	000	000	000	003	000	000	000	
		0090	004	000	000	000	005	000	000	000	006	000	000	000	007	000	000	000	
		OAOO	008	000	000	000	009	000	000	000	00A	000	000	000	006	000	000	000	
		0080	000	000	000	000	000	000	000	000	00E	000	000	000	00F	000	000	000	
		00C0	000	000	000	000	001	000	000	000	002	000	000	000	003	000	000	000	
		0000	004	000	000	000	005	000	000	000	006	000	000	000	007	000	000	000	
		00E0	008	000	000	000	009	000	000	000	00A	000	000	000	008	000	000	000	
		00F0	000	000	000	000	00D	000	000	000	00E	000	000	000	00F	000	000	000	-
* []									F	ead Block	Write Block	1				Locatio	on 63		

## 2.12 Conclusion

This application note describes how to initialize the RTG4 FPGA LSRAM with user data programmed into the  $\mu$ PROM. It provides an interface that can be instantiated in a user design to perform LSRAM initialization using  $\mu$ PROM. It also explains how to simulate and validate the design on RTG4 Development Board.



# 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 14 • FlashPro Express Job Project

	New Job Project from FlashPro Express Job	Ctrl+N
	🚰 Open Job Project	Ctrl+0
	× Close Job Project	
	🕍 Save Job Project	Ctrl+ <mark>Shi</mark> ft+A
or	Set Log File	
	Export Log File	
	Preferences	
	Execute Script	Ctrl+U
	Export Script File	
	Recent Projects	,
	Exit	Ctrl+Q
	or	Cor Set Log File Export Log File Export Script File Execute Script Export Script File Recent Projects

- 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\_ac454\_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	$\times$
Import HashPro Express job file     3_PCIe_SGDMA\rtg4_dg0713_df\Programming_Job\top.job     Browse	
Construct automatically (developer mode) Connected programmers: Programming interface: JTAG	
FlashPro Express job project name:     top       FlashPro Express job project location:     C:\JUNK\RTG4   Browse	
Help OK Cancel	

- 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

	Programmer	<b>1</b> RT4G150	۲
		¢ TDO	TDI 🗢
1 🚺 🗹  \$20:	IQVPTI IDLE	IDL	E
		31	

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

ashPro Express C:\Work\Projects\AC444_Space t Edit View Programmer Help	Wire\Nov_2020_update\QR_12p6\test_FPExpress\top\top.prc - JTAG Programming Interface*	
resh/Rescan Programmers		
Programmer	RT4G150     TOI     TOI	
S201QVPTI RUN PASSED	PASSED	
RAM I	1 PROGRAMMER(S) PASSED	
RUN	1 PROGRAMMER(S) PASSED	
RUN	14G150' : Frogramming FFGA Array	
RUN	14GISU' : Frogramming FFGA Array 14GISU' :	
RUN lessages Frors A Warnings Info frammer 'S201QVPTI' : device 'k grammer 'S201QVPTI' : device 'k grammer 'S201QVPTI' : device 'k grammer 'S201QVPTI' : device 'k grammer 'S201QVPTI' : device 'k	14GISO' : Programming FFGA Array T4GISO' : T4GISO' : EXPORT DSN[128] = 00000000000000000000000000000000000	
RUN essages Errors A Warning Info rrammer 'S201QVPTI' : device 'K rrammer 'S201QVPTI' : device 'R rrammer 'S201QVPTI' : device 'R rrammer 'S201QVPTI' : device 'R	TWGISU' : Frogramming FFGA Array T4GISO' : T4GISO' : EXFORT DSN[128] = 00000000000000000000000000000000000	

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\_ac454\_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: Design Files

You can download the design files from the following location on the Microsemi website: http://soc.microsemi.com/download/rsc/?f=rtg4\_ac454\_df

The design files consist of a Verilog version of Libero project folder, source file (data storage memory client), and programming file (\*.job) for RTG4 Development Kit. Refer to the <code>readme.txt</code> file included in the design files for the directory structure and description.

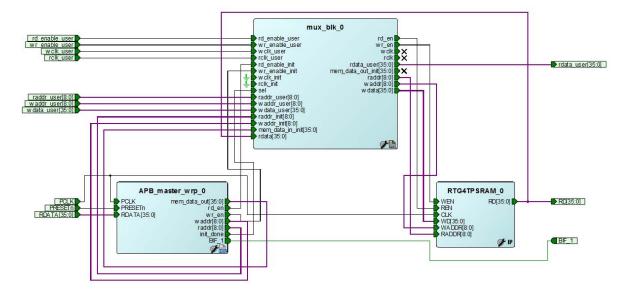


# 6 Appendix 4: Customizing RAM Wrapper Interface

This section describes how to customize a RAM wrapper interface according to the LSRAM variation configuration. The Fabric APB master code needs to be modified to support other LSRAM configurations that are different from those shown in this application note. The following figure shows the RAM APB wrapper block, which consists of the following blocks:

- RTG4TPSRAM\_0: RTG4 LSRAM configured as two-port mode with 512 depth and 36 width
- **APB\_master\_wrp\_0:** Fabric APB master interface
- Mux\_blk\_0: mux arbiter block to switch SRAM ports

#### Figure 18 • APB Master Wrapper SmartDesign



The RTG4TPSRAM\_0 setting must be updated based on the specific variation used. Also, the fabric APB master RTL code must be modified, changing the DATA\_WIDTH and ADDR\_WIDTH parameters as necessary. After the modifications are made, SmartDesign must be connected and regenerated. This design supports a data width of 36.



# 7 Appendix 5: How to Reset RAM Block Contents Using µPROM

In Libero SoC, the RTG4 uPROM Configurator allows the addition of a client for resetting the contents of all RAM blocks. At device power-up or when the DEVRST\_N signal goes active, the RAM initialization client initializes all RAM blocks to zero. After device power-up, the data read from all µSRAM and LSRAM address locations is zero until the RAM blocks are written to.

**Note:** When you read the initial zero from an address in a RAM block that has ECC enabled, its SB\_CORRECT and DB\_DETECT flags get asserted. The flags for any given RAM address location reset once that address location is written to.

The following steps describe how to initialize RAM blocks using UPROM Configurator in Libero SoC. The design must consist of RTG4 dual-port LSRAM, two-port LSRAM, or micro SRAM blocks.

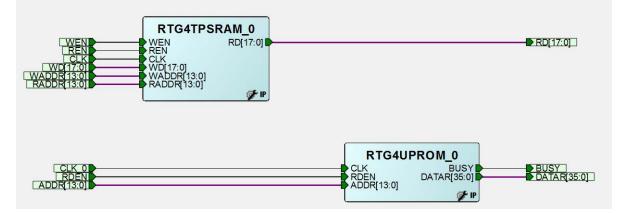
1. From the Libero Catalog, select the RTG4 uPROM macro, as shown in the following figure, and drag it on to the SmartDesign canvas.

#### Figure 19 • Catalog Window

Catalog					8	x
prom	۹	•	Γ	Simulation Mode	Ô	Ŧ
Name		- 7	7	Version		
- Memory & Controllers						
CoreUPROMIF_APB				2.0.107		
RTG4 uPROM				1.0.106		

The following figure shows a sample SmartDesign block consisting of the RTG4  $\mu$ PROM Configurator and a two-port large SRAM instance configured as 16 1K × 18.

#### Figure 20 • Sample SmartDesign Block





2. Open the **uPROM Configurator**, click **Add** ..., and select **Add init client to system**, as shown in the following figure.

Figure 21 • uPROM Configurator

UPROM Configurator	?	×
Add       Image: Client Sine User Clients in uPROM         Add client to system       Client Name       Start Address       36-bit words         Add init client to system :       0         Used memory(36-bit words):       0         Free memory(36-bit words):       10400		
Used space Free space Edit Delete		
Help • OK	Canc	el

The read only data client **MSCC\_RAM\_INITIALIZATION\_TO\_ZERO** gets added at address location 0x0, as shown in the following figure. This initializes all RAMs in the device using broadcast feature.

Note: This client cannot be edited, but it can be deleted.



IPROM Configurator		?	×
Add 🔻	User clients in uPROM		
Usage statistics Available memory(36-bit words): 10400	Client Name Start Address 36-bit words		_
Used memory(36-bit words): 820	1 MSCC_RAM_INITIALIZATION_TO_ZERO 0x0 820		
Free memory(36-bit words) : 9580			
Used space			
Free space	View Delete		
Help 👻	ОК	Cance	el

3. After adding the initialization client, run the design flow till the Run PROGRAM Action step.



- 4. After the device gets programmed, double-click the **SmartDebug** design in the **Design Flow** tab to check whether the RAM block contents are reset.
- 5. In SmartDebug, click **Debug FPGA Array...** as shown in the following figure.

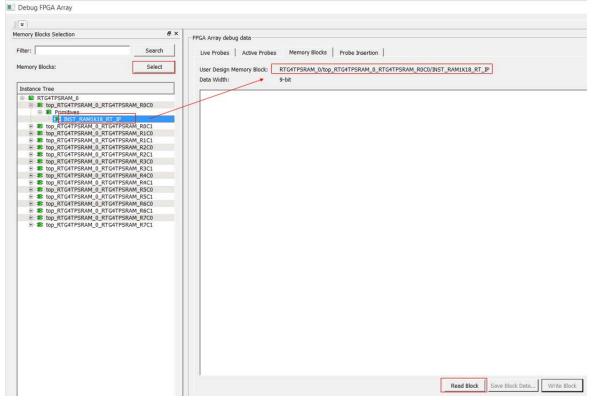
#### Figure 23 • SmartDebug Window

<ul> <li>SmartDebug</li> </ul>	- 🗆 X	
File View Help		
Device: RT4G150 (RT4G150)	Programmer: E201EXXCD (E201EXXCD)	
ID code read from device: 207011CF		
View Device Status	Debug FPGA Array	
	Debug SERDES	
Log	£	P ×
E Messages 🕹 Errors 🗼 Warnings 🌒 Info		



 In the Debug FPGA Array window, click the Memory Blocks tab, select the INST\_RAM1K18\_RT\_IP instance, and click Read Block to read its content, as shown in the following figure.

Figure 24 • Debug FPGA Array Window



The entire RAM block content is reset to zero, as shown in following figure. You can select the other instance and read its content, which will also be reset to zero.

Figure 25 • RAM Contents Reset to Zero

<b>v</b>																		
emory Blocks Selection & X	- FPGA Array	debug data -																
ilter: Search					1 .													
	Live Prob	es Active	Probes N	femory Blocks	Probe Inser	tion												
lemory Blocks:Select	User Desi Data Wid	ign Memory B th:	lock: RTG4 9-bit	TPSRAM_0/top	_RTG4TPSRAM	1_0_RTG4TPSR	AM_R0C0/INS	r_RAM1K18_R1	_IP									
Instance Tree								-										_
B S RTG4TPSRAM_0		0	1	2	3	4	5	6	7	8	9	Α	B	C	D	E	F	
# top_RTG4TPSRAM_0_RTG4TPSRAM_R0C0	0000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	-
Primitives     INST RAMIK18 RT IP	0000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
INST_RAMINIE_RT_P      E top_RTG4TPSRAM_0_RTG4TPSRAM_R0C1	0010	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
* 12 top RTG4TPSRAM 0 RTG4TPSRAM RICO																		
top_RTG4TPSRAM_0_RTG4TPSRAM_R1C1	0020	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	_
top_RTG4TPSRAM_0_RTG4TPSRAM_R2C0	0030	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
top_RTG4TPSRAM_0_RTG4TPSRAM_R2C1																		4
top_RTG4TPSRAM_0_RTG4TPSRAM_R3C0 top_RTG4TPSRAM_0_RTG4TPSRAM_R3C1	0040	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
E top_RTG4TPSRAM_0_RTG4TPSRAM_R4C0	0050	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
# # top_RTG4TPSRAM_0_RTG4TPSRAM_R4C1	0030																	4
# top_RTG4TPSRAM_0_RTG4TPSRAM_R5C0	0060	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
B top_RTG4TPSRAM_0_RTG4TPSRAM_R5C1	0070	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
top_RTG4TPSRAM_0_RTG4TPSRAM_R6C0 top_RTG4TPSRAM_0_RTG4TPSRAM_R6C1	0070	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
	0080	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
* Stop RTG4TPSRAM 0 RTG4TPSRAM R7C1									000			000						
	0090	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
	00A0	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
	0080	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
		000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	-
	00C0																	-
	00D0	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
	00E0	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
	00F0	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
	0100	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	1
	0110	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
																		4
	0120	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
	0130	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	
	0140	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	1