## AC491 Application Note PolarFire EDAC and Scrubbing of Fabric RAMs





а 🔨 Міскоснір 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	Revision History         1           .1         Revision 2.0
2	DAC and Scrubbing of Fabric RAMs       2         1       Design Requirements       2         2       Prerequisites       2         3       Application Note Design       3         2.3.1       Design Description       4         4       Clocking Structure       6         5       Reset Structure       6         6       Hardware Implementation       7
3	etting Up the Hardware
4	Running the Demo
5	onclusion
6	ppendix 1: Programming the Device Using FlashPro Express
7	ppendix 2: Running the TCL Script



# **Figures**

Figure 1	Block Diagram	3
Figure 2	CoreEDAC IP Configuration	5
Figure 3	Refresh Period Timer	5
Figure 4	Clocking Structure	6
Figure 5	Reset Structure	6
Figure 6	Hardware Implementation	7
Figure 7	Device Manager	8
Figure 8	PolarFire Board Setup	9
Figure 9	Selecting the COM Port 1	0
Figure 10	Device Connection Successful Message 1	1
Figure 11	Write Incremental Data to Memory 1	1
Figure 12	Single Bit Errors and Double Bit Errors Count1	2
Figure 13	Writing 0xAA at Address 0x48 1	2
Figure 14	Reading Value at 0xAA at Address 0x48 1	3
Figure 15	Single Error Injection at Address 0x48 1	3
Figure 16	Single Error count	4
Figure 17	Single Bit Error Memory Scrubbing	4
Figure 18	Double Error Injection at Address 0x45 1	5
Figure 19	Double Error count	5
Figure 20	Loop Test 1	6
Figure 21	FlashPro Express Job Project	8
Figure 22	New Job Project from FlashPro Express Job	8
Figure 23	Programming the Device	9
Figure 24	FlashPro Express—RUN PASSED	9



# **Tables**

Table 1	Design Requirements	. 2
Table 2	Jumper Settings	. 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 most current publication.

## 1.1 Revision 2.0

Added Appendix 2: Running the TCL Script, page 20.

## 1.2 Revision 1.0

The first publication of this document.



## 2 EDAC and Scrubbing of Fabric RAMs

The CoreEDAC IP generates EDAC circuitry for both internal (on-chip) and external RAM blocks. The user data is fed to the EDAC encoder, which calculates the parity bits and appends these to the user data, forming a codeword. The codeword is stored into the RAM. During user read, the read codeword is decoded first, which detects and corrects errors (if any), discards parity bits, and outputs the corrected user data word. Scrubbing periodically checks every memory location using the ECC decoder. If a location contains a corrupted word, the decoder detects and corrects the word. The scrubbing circuitry then writes the corrected word back to the same location. To provide normal access to the RAM and prevent decreasing performance, scrubbing is only done during idle periods. The scrubbing circuitry sets a proper write address and write enable signals, writing the corrected codeword back to the RAM. Writeback occurs only upon detecting an error.

The application note design can be programmed using the following option:

Using the job file: To program the device using the job file provided along with the design files, see Appendix 1: Programming the Device Using FlashPro Express.

### 2.1 Design Requirements

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

### Table 1 •Design Requirements

Requirement	Version
Operating system	64-bit Windows 7, 8, or 10
Hardware	
PolarFire Evaluation Kit (MPF300-EVAL-KIT) -12 V/5 A AC power adapter and cord -USB 2.0 A to mini-B cable for UART and programming	Rev D or later
Software	
Libero SoC	Note: Refer to the readme.txt file
FlashPro Express	provided in the design files for th software versions used with this reference design.

**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 the design files from: http://soc.microsemi.com/download/rsc/?f=mpf\_ac491\_df
- 2. Download and install Libero SoC from: https://www.microsemi.com/product-directory/design-resources/1750-libero-soc



## 2.3 Application Note Design

EDAC detects a 1-bit error or 2-bit error when data is read from the memory. If EDAC detects the 1-bit error, the EDAC controller corrects the same error bit. If EDAC is enabled for all the 1-bit and 2-bit errors, corresponding error counters will get incremented. To demonstrate this, an error is introduced manually and detection and correction is observed. The application note design shows how to use CoreEDAC IP with internal  $\mu$ SRAMs for the following:

- 1. Detect and correct single-bit errors
- 2. Detect double-bit errors
- 3. Perform auto memory scrubbing and on-demand memory scrubbing

The similar approach can be used for on-chip LSRAMs. This application note design does not demonstrate the use of EDAC with external memories.

The block diagram of the design is shown in Figure 1.

### Figure 1 • Block Diagram



- 1. Configure CoreEDAC IP to use internal µSRAM blocks and set the memory depth to 1kb. Enable error inject test port to induce an errors into the memory. Enable scrubbing logic to correct the errors.
- The user RTL logic is used to write and read 1kb data to and from the memory. This logic will also monitor the status flags set by the CoreEDAC controller for monitoring the scrubbing request and detecting single-bit and double-bit errors.
- 3. UART is used to interface with GUI to provide commands to the design and display the error counters and status flags.
- 4. The application note design performs the following operations using the GUI.
- Memory Write: Initialize 1kb µSRAM memory with incremental data.
- Memory Read: Read 1kb data from µSRAM memory to verify the memory is initialized. Initially, the memory is not corrupted; hence the error counters must be zero.
- Inject Error: Inject Single-bit or double-bit error in memory locations.
- Memory Read: Read the data from corrupted µSRAM memory.
  - **Single-bit error**: The relevant counter will be incremented and the corrected data will be available.
  - **Double-bit error**: The relevant counter will be incremented and the erroneous data will be available.
- Memory Scrub: Performs memory scrubbing and correct single bit errors in µSRAM memory.
- **Single Memory Write**: User can access the entire 1kb memory through this operation. User can provide the memory address and the data to be written during this operation.
- **Single Memory Read**: User can access the entire 1kb memory through this operation. User can provide the memory address and the data to be read during this operation.
- Loop Test: User can use this option to perform the operations from step to step in one click.



### 2.3.1 Design Description

This application note design involves implementation of following tasks:

- Writing data to µSRAM
- Reading data from µSRAM
- Corrupting one or two bits
- · Reading the data
- **Note:** In the case of a 1-bit error, the EDAC controller corrects the error. In the case of a 2-bit error, the EDAC controller does not correct the error.

The following tests are implemented in this application note design.

- Loop Test
- Manual Test

Note: These tests are applicable to both 1-bit and 2-bit errors.

### 2.3.1.1 Loop Test

Loop Test is executed when the PolarFire FPGA receives a loop test command from the GUI. Initially, all the error counters and EDAC related registers are placed in the RESET state.

The following steps are executed for each iteration:

- 1. 1kb incremental data is written into the memory.
- 2. 1kb data is read from the memory.
- 3. 50 single bit errors are injected into the memory.
- 4. The memory is read and the error count is displayed.
- 5. Scrubbing is performed on the memory and the scrubbing correction count is displayed.
- 6. 1-bit or 2-bit error detection and correction is sent to the GUI.

### 2.3.1.2 Manual Test

This method allows manual testing for enabling or disabling EDAC and write or read operation. Using this method, 1-bit or 2-bit errors can be introduced to any location within the µSRAM. Enable the EDAC and write data to the specified address using the GUI fields. Disable the EDAC and write 1-bit or 2-bit corrupted data to the same address location. Enable the EDAC and read the data from the same address location. The corresponding error counter is displayed on the GUI. The GUI Serial Console logs all the actions performed in PolarFire.

As shown in Figure 2, the CoreEDAC IP is configured with the following options:

- Internal µSRAM blocks are selected and the memory depth is set to 1Kb.
- Error injection is enabled for inducing errors into the memory.
- Scrubbing logic is enabled for correcting errors.
- Triple Redundancy is also enabled for the generation of three independent sets of the EDAC circuitry and the majority vote logic for protecting the EDAC from soft errors.
- Scrubber is enabled to detect data errors and repair them.



Figure 2 • CoreEDAC IP Configuration

Configuration	
RAM block : Internal V	
Width (bits) : 8	Encoded width (bits): 13
Depth (words) : 1024	Approx. Max RAM depth: 245760
Click on "Check Configuration" to check availability of Depth entered	
Use Triple Redundancy Use single Read/Write dock	Create error injection test port
Encoder / Decoder	
Limit encoder latency to : 0 Clock cycles $\sim$	Actual Latency : 0 Clock cycles
Limit decoder latency to : 0 Clock cycles $\sim$	Actual Latency: 0 Clock cycles
Provide RAM read address with the data	
Error Scrubber	
Use scrubber to detect data errors	epair detected errors
Min Refresh Period : 10240 Read Clocks	Refresh Period : 1.02E+006 Read Clocks
Start Address : 0	Binary Divider Width : 10
End Address : 1023	Timeout : 1000
Protected RAM (generated)	
RAM Type: Micro SRAM V	
Initialize RAM with 0 Use pipeline register for read data	3
	Check Configuration OK Cancel

### 2.3.1.3 Scrubbing Refresh Period

The refresh period defines how often the scrubbing session runs. The block diagram of the refresh period timer is shown in Figure 3. The timer is driven by the RCLK signal. The binary divider that has a configurable bitwidth of DIV\_WDTH, generates a relatively slow signal, *dec*. The frequency of the *dec* signal equals the frequency of the read clock divided by  $2^{\text{DIV}_{WDTH}}$ . The *dec* signal serves as an input to the configurable arbitrary divider. It divides *dec* frequency by arbitrary number TMOUT\_SET. As a result, the circuitry generates a timeout output signal once per TMOUT\_SET ×  $2^{\text{DIV}_{WDTH}}$  RCLK periods.

The refresh period must be more than 10 times the scrubbing time that is DIV\_WDTH, and TMOUT\_SET parameters must satisfy the following condition: TMOUT\_SET\*2<sup>DIV\_WDTH</sup> > 10\*(SCRUB\_AMAX - SCRUB\_AMIN). The timeout signal initiates another scrubbing session. As the user access takes priority over scrubbing, there might be an exception.

For more information about setting the refresh period timer, scrubbing, and scrubbing refresh period, see *HB0143:CoreEDAC IP Handbook*. Table 12 in this handbook explains DIV\_WDTH, TMOUT\_SET, SCRUB\_AMAX, and SCRUB\_AMIN parameters.

### Figure 3 • Refresh Period Timer





## 2.4 Clocking Structure

In this design, there is one clock domain. The on-board 50 MHz crystal oscillator is connected to the PF\_CCC block which generates 100 MHz clock that provides clock source to CORERESET\_PF, COREUART, UART\_IF, and COREEDAC modules. The following figure shows the clocking structure of the design.

### Figure 4 • Clocking Structure



### 2.5 Reset Structure

In this design, the reset signal of COREUART, UART\_IF, and COREEDAC blocks are issued using the CORERESET\_PF module. The CORERESET\_PF module releases active low reset when the PF\_CCC lock and PF\_INIT\_MONITOR INIT\_DONE are asserted. The following figure shows the reset structure of the design.

### Figure 5 • Reset Structure





## 2.6 Hardware Implementation

Figure 6 shows the PolarFire EDAC and scrubbing design implemented in Libero SoC.







# 3 Setting Up the Hardware

The following steps describe how to setup the hardware.

1. Ensure that the following Jumper Settings are set on the board.

### Table 2 • Jumper Settings

Description	Default
Short pin 2 and 3 for programming the PolarFire FPGA through FTDI	Closed
Short pin 1 and 2 for programming through the on-board FlashPro5	Open
Short pin 1 and 2 for programming through the FTDI SPI	Closed
Short pin 1 and 2 for manual power switching using SW3	Closed
Short pin 3 and 4 for 2.5 V	Closed
	DescriptionShort pin 2 and 3 for programming the PolarFire FPGA through FTDIShort pin 1 and 2 for programming through the on-board FlashPro5Short pin 1 and 2 for programming through the FTDI SPIShort pin 1 and 2 for manual power switching using SW3Short pin 3 and 4 for 2.5 V

- 2. Connect the power supply cable to the J9 connector on the board.
- 3. Connect the USB cable from the Host PC to **J5** (FTDI port) on the board.
- 4. Power on the board using the **SW3** slide switch.
- 5. Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the device manager of the host PC.
- 6. As shown in Figure 7, the port properties of COM13 show that it is connected to USB Serial Converter C. Hence, COM13 is selected in this example. The COM port number is system specific.

### Figure 7 • Device Manager



The PolarFire board setup is shown in Figure 8.









# 4 Running the Demo

The PolarFire EDAC application is a simple Graphic User Interface (GUI) that runs on the host PC to communicate with the PolarFire Device. Before running the demo, ensure Setting Up the Hardware and Appendix 1: Programming the Device Using FlashPro Express.

To run the EDAC demo:

1. Run the setup.exe file available at the following design files location:

<\$Download Directory>\mpf ac491 df\GUI\EDAC PF GUI.exe

- 2. Follow the installation wizard to install the GUI application.
- After successful GUI Installation. Invoke the EDAC GUI from All Programs > EDAC Demo > EDAC PF GUI
- 4. Open the PolarFire EDAC GUI and select the COM port.

#### Figure 9 • Selecting the COM Port

a Company	EDA	C Demo		PolarFire <sup>®</sup> FPGA
COM Connect	Memory Write Memo	ry Read Log	Read Data	
СОМ 🕌 🖃 🍕	Write R	ead		^
Error COM12 COM13				
Singl COM14 D	Oouble Error 0 Clea	r Counts		
Single Refresh				
Address: 0x 0	Vrite Data: 0x 0	Write		
Single Memory Read				
Address: 0x 0 R	ead Data: 0x 0	Read		
Inject Error	Loop Test			
Error Type Single 🔍	Loop T	est		
Address: 0x 0	Inject Error Memor	y Write		
Single Bit Error Memory Scr	ubbing Inject E	y Read		
	Memor	y Read	_	
Correction Count 0	Scrub Memor	y Scrub	0	Clear Log



5. After selecting the COM port, click **Connect** icon as highlighted in Figure 10. The log window will display connection successful message.

Figure 10 • Device Connection Successful Message

	1	EDAC De	PolarFire FPGA			
COM Connect	Memory Write	Memory Read	Log	Read Data		
сом 🖁 соміз 🔳 🔫	Write	Read	Device	e is Connected to	GUI Successfully	^
Error Counts						
Single Error 0 Do	ouble Error 0	Clear Counts				
Single Memory Write						
Address: 0x 0 W	rite Data: 0x 0	Write				
Single Memory Read						
Address: 0x 0 Re	ad Data: 0x 0	Read				
Inject Error	Loc	op Test				
Error Type Single 🗸	Inject Error	Loop Test				
Address: 0x 0	- gett Enter	Memory Write				
		Memory Read				
Single Bit Error Memory Scru	bbing	Inject Error				~
Correction Count 0	Scrub	Memory Kead			Clear Log	

6. Click **Write** as highlighted in Figure 11 to write 1kb incremental data into the memory. The log window will display the write successful message.

Figure 11 • Write Incremental Data to Memory

Містоsеті     «      «      Мисласни» сотралу	E	DAC Der	no		PolarFire <sup>®</sup> FPGA
COM Connect	Memory Write	Memory Read	Log	Read Data	
сом 🖁 соміз 💽 🔫	Write	Read	Device	e is Connected t	o GUI Successfully
Error Counts			Writin Succe	g 1KB Incremen ssful	tal Data to Memory is
Single Error 0 Do	uble Error 0	Clear Counts			
Single Memory Write					
Address: 0x 0 Wr	ite Data: 0x 0	Write			
Single Memory Read					
Address: 0x 0 Rea	ad Data: 0x 0	Read			
Inject Error	Loo	p Test			
Error Type Single 🔽	Inight From	Loop Test			
Address: 0x 0	inject Error	Memory Write			
Circula Dis Correctioners Correl		Memory Read			
Single bit Error Memory Scrut	bing	Memory Read			~
Correction Count 0	Scrub	Memory Scrub			Clear Log



7. Click Read as highlighted in Figure 12 to read 1kb data. The read data will be displayed in the Read Data tab and the single bit errors and double bit errors will be displayed in their respective fields. Also, the log window will display the read successful message.

Figure 12 • Single Bit Errors and Double Bit Errors Count

a <u> </u>	EDAC Demo				PolarFire <sup>®</sup> FPGA							
COM Connect	Memory Write	Memory Read	Log	Read	Data							
	Write	Read	00	00	01	02	03	04	05	06	07	^
		Neou	08	08	09	0A	0B	0C	0D	0E	OF	
			10	10	11	12	13	14	15	16	17	
Error Counts			18	18	19	1A	1B	1C	1D	1E	1F	
			20	20	21	22	23	24	25	26	27	
Single Error 0 D	ouble Error 0	Clear Counts	28	28	29	2A	2B	2C	2D	2E	2F	
			30	30	31	32	33	34	35	36	37	
Single Memory Write			38	38	39	3A	3B	3C	3D	3E	3F	
			40	40	41	42	43	44	45	46	47	
Address: 0x 0 V	/rite Data: 0x 0	Write	48	48	49	4A	4B	4C	4D	4E	4F	
			50	50	51	52	53	54	55	56	57	
Single Memory Read			58	58	59	5A	5B	5C	5D	5E	5F	
			60	60	61	62	63	64	65	66	67	
Address: 0x 0 R	ead Data: 0x 0	Read	68	68	69	6A	6B	6C	6D	6E	6F	
			70	70	71	72	73	74	75	76	77	
niect Error	1.0	an Tast	78	78	79	7A	7B	7C	7D	7E	7F	
inject Litor	LO	op Test	80	80	81	82	83	84	85	86	87	
Error Type Single		Loop Test	88	88	89	8A	8B	8C	8D	8E	8F	
Litor type Single	Inject Error	coop rest	90	90	91	92	93	94	95	96	97	
Address: 0x 0		Memory Write	98	98	99	9A	9B	9C	9D	9E	9F	
		Memory Read	A0	A0	A1	A2	A3	A4	A5	A6	A7	
Single Bit Error Memory Scru	ubbing	Inject Error	A8	A8	A9	AA	AB	AC	AD	AE	AF	
		Memory Read	BO	BO	B1	B2	<b>B</b> 3	B4	B5	B6	B7	
Correction Count 0	Scrub	Memory Scrub	B8	B8	B9	BA	BB	BC	BD	BE	BF	
			C0	<u></u>	C1	<b>C</b> 1	C2	CA	C5	C6	C7	

8. For a single memory location, enter the address and data to be written and click **Write** as highlighted in Figure 13.

Figure 13 • Writing 0xAA at Address 0x48

Microsemi a Microsemi	E	DAC Der	no		PolarF FPGA	TIRE" A
COM Connect	Memory Write	Memory Read	Log	Read Data		
сом 🖁 сом13 💽 🔫	Write	Read	Devic	e is Connected	I to GUI Successfully	^
Error Counts			Succe	ig TKB increme issful Asmony Pood S	ental Data to Memory is	
Single Error 0 Do	uble Error 0	Clear Counts	Writin	ng 0xAA at Ado	dress 0x48 is Successful	
Single Memory Write						
Address: 0x 48 Wr	ite Data: 0x 🗛	Write				
Single Memory Read						
Address: 0x 0 Rea	ad Data: 0x 0	Read				
Inject Error	Loo	p Test				
Error Type Single 🔍	Inight Frank	Loop Test				
Address: 0x 0	inject Error	Memory Write				
Single Bit Error Memory Scrub	obing	Memory Read Inject Error				~
Correction Count 0	Scrub	Memory Read Memory Scrub			Clear Log	



9. For reading a single location, provide the memory address and click **Read** as highlighted in Figure 14. The corresponding data will be displayed in the GUI.

Figure 14 • Reading Value at 0xAA at Address 0x48

Містоsеті     «     «     «     «     Міслосне сотралу	E	DAC De	mo		PolarFif FPGA	۶E
COM Connect	Memory Write	Memory Read	Log	Read Data		
сом 🖁 сом13 🔳 🔫	Write	Read	Device	e is Connected	to GUI Successfully	^
Error Counts			Succe 1KB M	g 1KB Increme ssful lemory Read S	ental Data to Memory is	
Single Error 0 D	ouble Error 0	Clear Counts	Readin	g OxAA at Add ng Value OxAA	at Address 0x48 is Successful at Address 0x48 is Successful	]
Single Memory Write						
Address: 0x 48 W	/rite Data: 0x AA	Write				
Single Memory Read						
Address: 0x 48 Re	ead Data: 0x AA	Read				
Inject Error	Loo	p Test				
Error Type Single 🗸	Initiat From	Loop Test				
Address: 0x 0	Inject Error	Memory Write				
Single Bit Error Memory Scru	bbing	Memory Read Inject Error				~
Correction Count 0	Scrub	Memory Read Memory Scrub			Clear Log	

10. For injecting a single bit error, provide the memory address and click **Inject Error** as highlighted in Figure 15.

Figure 15 • Single Error Injection at Address 0x48

Microsemi. a @ Microcele company	E	DAC Der	no		PolarFir FPGA	E
COM Connect	Memory Write	Memory Read	Log	Read Data		
сом 🎖 сом13 🔳 🔫	Write	Read	Devic	e is Connected t	o GUI Successfully	^
Error Counts			Succe 1KB N	ig 1KB Incremen ssful 1emory Read Su	tal Data to Memory is ccessful	
Single Error 0 Do	ouble Error 0	Clear Counts	Writin Readi Single	g 0xAA at Addr ng Value 0xAA a error Injection a	ess 0x48 is Successful t Address 0x48 is Successful at address 0x48 is Successful	1
Single Memory Write			-			1
Address: 0x 48 W	rite Data: 0x AA	Write				
Single Memory Read						
Address: 0x 48 Re	ad Data: 0x AA	Read				
Inject Error	Loo	p Test				
Error Type Single 🗸	Inject Error	Loop Test				
Address: 0x 48	inject citor	Memory Write				
Single Bit Error Memory Scru	bhing	Memory Read				
single bit ciror wemory scru	bollig	Memory Read				~
Correction Count 0	Scrub	Memory Scrub			Clear Log	



11. After Injecting a single bit error, click **Read** as highlighted in Figure 16. The Single Error count will be updated to 1.

Figure 16 • Single Error count

	I	EDAC Der	no		POLARFIF FPGA	E
COM Connect	Memory Write	Memory Read	Log	Read Data		
сом 🖁 соміз 🔳 💽	Write	Read	Devic	e is Connected	to GUI Successfully	^
Error Counts			Writin Succe	ig 1KB Increme Issful	ntal Data to Memory is	
Single Error 1	Double Error 0	Clear Counts	Writin Readi Single	nemory Read So ng 0xAA at Add ng Value 0xAA e error Injection	ress 0x48 is Successful at Address 0x48 is Successful at address 0x48 is Successful	
Single Memory Write			Readi 1KB N	ng Value 0xAA Memory Read Su	at Address 0x48 is Successful uccessful	
Address: 0x 48	Write Data: 0x AA	Write				
Single Memory Read						
Address: 0x 48	Read Data: 0x AA	Read				
Inject Error	Lo	op Test				
Error Type Single 🗸	the state	Loop Test				
Address: 0x 48	Inject Error	Memory Write				
Single Bit Error Memory S	crubbing	Memory Read				
single bit that memory s		Memory Read				-
Correction Count 0	Scrub	Memory Scrub			Clear Log	

12. The Memory Scrubbing detects and corrects single bit errors. Click **Scrub** as highlighted in Figure 17 to perform scrubbing on the memory. After the scrubbing is completed, the **Correction Count** will display the number of single bit errors corrected during scrubbing.

Figure 17 • Single Bit Error Memory Scrubbing

Microsemi a @Microsem company	E	DAC De	mo		POLARFIR FPGA	SE
COM Connect	Memory Write	Memory Read	Log	Read Data		
сом 🖁 сом13 🔳 🔫	Write	Read	Device	e is Connected	to GUI Successfully	
Error Counts Single Error 0 Di	ouble Error 0	Clear Counts	Writin Succe 1KB M Writin Readir Single Readir Single	ig 1KB Increme ssful Iemory Read S ig 0xAA at Add ng Value 0xAA error Injection ng Value 0xAA Bit Error Mem	ntal Data to Memory is uccessful ress 0x48 is Successful at Address 0x48 is Successful at Address 0x48 is Successful at Address 0x48 is Successful ory Scrubbing Successful	
Address: 0x 48 Re	ead Data: 0x AA	Read				
nject Error	Loo	p Test				
Error Type Single 🔽	Inject Error	Loop Test				
Address: 0x 48		Memory Write				
		Memory Read				
Single Bit Error Memory Scru	ubbing	Inject Error				
		Memory Read				

**Note:** One single bit error is introduced in step 11. Hence, during scrubbing, the single bit error is corrected, and the correction value is updated to "**1**".



13. For injecting Double bit error, select the **Error Type** to **Double**, provide the memory address and click **Inject Error** as highlighted in Figure 18.

Figure 18 • Double Error Injection at Address 0x45

		EDAC Dei	no		PolarFire <sup>®</sup> FPGA
COM Connect	Memory Write	Memory Read	Log	Read Data	
сом 🖁 соміз 🔳 🔫	Write	Read	Devic	e is Connected t	to GUI Successfully
Error Counts			Writin	ig 1KB Incremen issful	ntal Data to Memory is
Single Error 0 D	ouble Error 0	Clear Counts	Writin Readin Single	nemory Read Su ng 0xAA at Addr ng Value 0xAA a e error Injection	ecessrui ess 0x48 is Successful at Address 0x48 is Successful at address 0x48 is Successful
Single Memory Write			Readi	ng Value 0xAA a	at Address 0x48 is Successful
Address: 0x 48 V	/rite Data: 0x AA	Write	Doub	le error Injection	n at address 0x48 is Successful
Single Memory Read					
Address: 0x 48 R	ead Data: 0x AA	Read			
Inject Error	Loc	op Test			
Error Type Double 🖂		Loop Test			
Address: 0x 48	Inject Error	Memory Write			
40		Memory Read			
Single Bit Error Memory Scru	ubbing	Inject Error			~
		Memory Read			
Correction Count 1	Scrub	Memory Scrub			Clear Log

14. Click **Read** as highlighted in Figure 19. The **Double Error** will be updated to "1", indicating a double bit error exists in the memory. The **Single Error** is zero because the single bit error is corrected during scrubbing as mentioned in the previous step.

### Figure 19 • Double Error count

Містоsеті     а      «Міслосн» сопралу	1	EDAC Der	no		PolarFir FPGA	E,
COM Connect	Memory Write	Memory Read	Log	Read Data		
сом 🖁 сом13 🔳 🔫	Ø Write	Read	Device	e is Connected	I to GUI Successfully	^
Error Counts			Writin Succe	g 1KB Increme ssful	ental Data to Memory is	
Single Error 0 D	ouble Error 1	Clear Counts	1KB N Writin Readin Single	lemory Read S g 0xAA at Add ng Value 0xAA error Injectior	uccessful Iress 0x48 is Successful at Address 0x48 is Successful n at address 0x48 is Successful	
Single Memory Write			Readin	ng Value 0xAA Bit Error Mem	at Address 0x48 is Successful nory Scrubbing Successful	
Address: 0x 48 W	Vrite Data: 0x AA	Write	Doubl 1KB N	e error Injectio Iemory Read S	on át address Öx48 is Successful Juccessful	
Single Memory Read						
Address: 0x 48 R	ead Data: 0x AA	Read				
Inject Error	Lo	op Test				
Error Type Double 🗸		Loop Test				
Address: 0x 48	Inject Error	Memory Write				
		Memory Read				
Single Bit Error Memory Scru	ubbing	Inject Error				¥
Correction Count 1	Scrub	Memory Scrub			Clear Log	



- 15. Click Loop Test as highlighted in Figure 20. The loop test performs the following operations.
  - Writes 1kb incremental data into the memory.
  - Reads 1kb data from the memory.
  - Injects 50 single bit errors into the memory.
  - Reads the memory and displays the error count.
    - Performs scrubbing on the memory and displays the scrubbing correction count.

### Figure 20 • Loop Test

Microsemi.	E	DAC Der	no	POLAR FPG	i A
COM Connect	Memory Write	Memory Read	Log Read	Data	
сом 🖁 сом13 🔳 🤜	🖗 Write	Read	Device is Con	nected to GUI Successfully	
Error Counts			Loop Test Suc	cessful	
Single Error 50	Oouble Error 0	Clear Counts			
ingle Memory Write					
Address: 0x 0	Write Data: 0x 0	Write			
ingle Memory Read					
Address: 0x 0	Read Data: 0x 0	Read			
nject Error	Loo	p Test			
Error Type Single 🗸		Loop Test			
Address: 0x 0	Inject Error	/ Memory Write			
Single Bit Error Memory Scr	ubbing	/ Memory Read / Inject Error			
		/ Memory Read			
Correction Count 50	Scrub	/ Memory Scrub		Clear Log	

This concludes the PolarFire EDAC Demo.





This demo highlights the EDAC capabilities of the PolarFire µSRAM memories. The 1-bit error or 2-bit error are introduced manually. 1-bit error correction and 2-bit error detection is observed using a GUI.



## 6 Appendix 1: Programming the Device Using FlashPro Express

This chapter describes how to program the PolarFire device with the job file using Flashpro Express. The job file is available at the following design files folder location: <\$Download\_Directory>\mpf\_ac491\_df/Programming\_Job

Follow these steps:

- 1. On the host PC, start the FlashPro Express software from its installation directory.
- 2. Select **New or New Job Project from FlashPro Express Job** from Project menu to create a new job project, as shown in Figure 21, page 18.

#### Figure 21 • FlashPro Express Job Project

ect cut view Programmer <u>n</u> eip		E FlashPro Express	
Job Projects		Project Edit View Programmer <u>H</u> elp	
		New Job Project from FlashPro Express Job	Ctrl+N
New		🚰 Open Job Project	Ctrl+O
<u>Dpen</u>		× Close Job Project	
be an a Development		🔚 Save Job Project	Ctrl+Shift+A
tecent Projects	or	Set Log File	•
		Export Log File	
		Preferences	
		Execute Script	Ctrl+U
		Export Script File	
		Recent Projects	•
		Exit	Ctrl+Q

- 3. 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\_Directory>\mpf\_ac491\_liberosoc\_jb
  - FlashPro Express job project location: Select Browse and navigate to the location where you want to save the project.

Figure 22 • New Job Project from FlashPro Express Job

Rew Job Project from FlashPro Express	Job	×
Programming job file:		
AC_RTPF\V12.5\mpf_ac491_df\Programm	ing_Job\top.job	Browse
FlashPro Express job project name:		
<b>1</b> top		
FlashPro Express job project location:		
C:\Users\I52881\Documents\Custom Offic	e Templates	Browse
Help	OK	Cancel

- 4. Click **OK**. The required programming file is selected and ready to be programmed in the device.
- 5. The FlashPro Express window appears as shown in Figure 23, page 19. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan Programmers**.



### Figure 23 • Programming the Device

Programmer	<b>€</b> ™	1PF300T 🔘	10	 	 	 	
1 ₽ 2003G6YMO		IDLE					
DGRAM I							
RUN				IDLE			
DGRAM I				IDLE			
DGRAM T RUN Messages & Errors A Warnings ()	Info			IDLE			
OGRAM T RUN Messages SErrors A Warnings C edded FlashPro5 programmer dete rammer / ScuosferWo 1: FlashPro	Info			IDLE			

6. Click **RUN** to program the device. When the device is programmed successfully, a RUN PASSED status is displayed as shown in Figure 24, page 19. See Running the Demo, page 10.

Figure 24 • FlashPro Express—RUN PASSED

Programmer	Ф         мрезоот         (1)           ф тво         тоі ф	
E2003G6YMO RUN PASSED	PASSED	
GRAM		
GRAM I	1 PROGRAMMER(S) PASSED	
GRAM I	1 PROGRAMMER(S) PASSED	
GRAM I	1 PROGRAMMER(S) PASSED	
DGRAM T RUN	1 PROGRAMMER(S) PASSED	
OGRAM  RUN  Messages & Errors A Wamings 1 Info in Frogramming Finished: Wed Mar 20 14:433	1 PROGRAMMER(S) PASSED	

7. Close FlashPro Express (**Project > Exit**).

The PolarFire device is programmed.



# 7 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 mpf\_ac491\_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.