# Debugging PolarFire FPGA Using SmartDebug Application Note

## AN4594

## Introduction (Ask a Question)

Design debugging is a critical phase of the FPGA design flow. SmartDebug enables the debugging of designs by providing verification and troubleshooting features at the hardware level. It provides access to probe points, Non-Volatile Memory (NVM), fabric and fabric RAM blocks, transceivers, and the DDR controller. These features enable designers to check the state of inputs and outputs in real-time, without any design modification. For more information about SmartDebug features, see SmartDebug User Guide.


© 2025 Microchip Technology Inc. and its subsidiaries

# Table of Contents

Introduction....1

1. Debugging PolarFire FPGA Designs Using SmartDebug....3
   - 1.1. Design Requirements....3
   - 1.2. Prerequisites....3
   - 1.3. Demo Design....4
   - 1.4. Clocking Structure....8
   - 1.5. Reset Structure....8
   - 1.6. Enabling FPGA Hardware Breakpoint (FHB)....9
   - 1.7. Programming the Device....9
   - 1.8. Debugging Using SmartDebug....11
   - 1.9. Conclusion....42
2. Appendix 1: Known Issues....43
   - 2.1. Data Traffic Errors on XCVR Lanes in CDR Mode....43
3. Appendix 2 : Synthesis....44
4. Appendix 3: Place and Route....45
5. Appendix 4: Running the TCL Script....48
6. Appendix 5: References....49
7. Revision History....50

Microchip FPGA Support....52

Microchip Information....52
- Trademarks....52
- Legal Notice....52
- Microchip Devices Code Protection Feature....53


© 2025 Microchip Technology Inc. and its subsidiaries

# 1. Debugging PolarFire FPGA Designs Using SmartDebug (Ask a Question)

This application note provides a demo design to demonstrate how SmartDebug is used for debugging Transceiver, DDR Memory, and Dual-Port SRAM (DPSRAM) in a PolarFire FPGA design.

## 1.1. Design Requirements (Ask a Question)

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

**Table 1-1.** Design Requirements

| Requirement | Description |
|---|---|
| Operating system | 64-bit Windows® 10 and 11 |
| **Hardware** | |
| PolarFire® Evaluation Kit (MPF300T-1FCG1152I) | Rev D or later |
| 2 SMA-to-SMA cables with 5 Gbps support | Only for Evaluation Kit |
| **Software** | |
| Libero® SoC | See the `Readme.txt` file provided in the design files for all software versions needed to create this reference design. |

**Important:** Libero® SmartDesign and configuration screenshots shown in this guide are for illustration only. To see the latest updates, open the Libero design.

## 1.2. Prerequisites (Ask a Question)

Before you begin:

1. Download and install Libero SoC (as indicated in the website for this design) from the following location:
   Libero SoC Documentation.
2. Demo design files download link: www.microchip.com/en-us/application-notes/AN4594

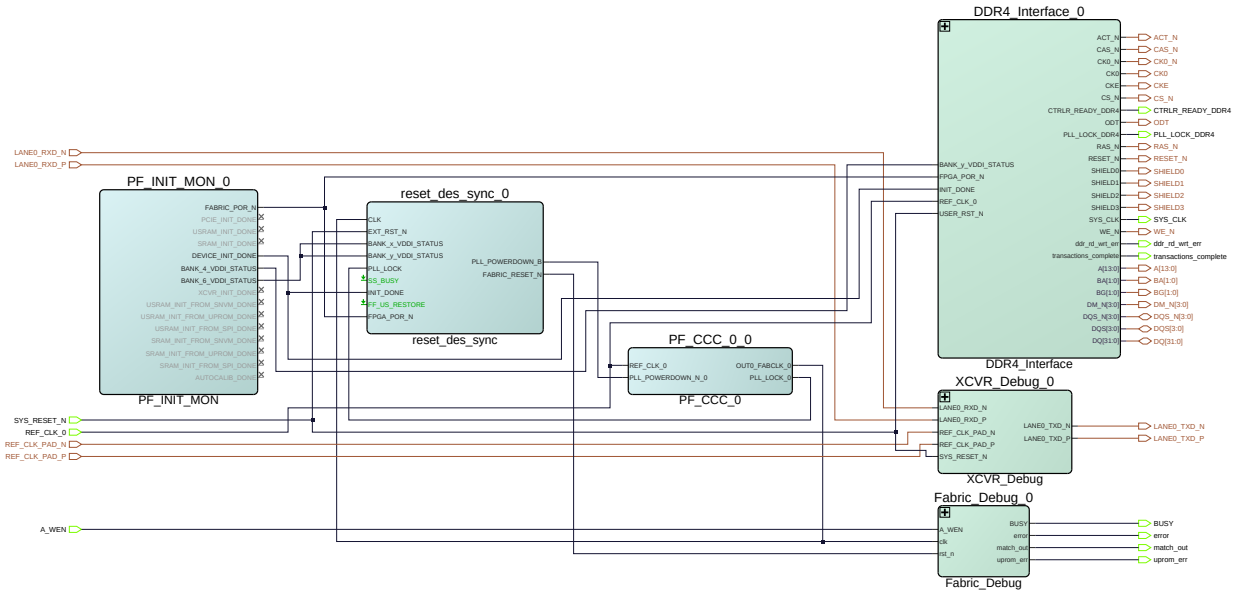

© 2025 Microchip Technology Inc. and its subsidiaries

## 1.3. Demo Design (Ask a Question)

This section describes the fabric, DDR interface, and XCVR design blocks implemented in Libero SoC.

The following figure shows the top-level blocks of SmartDebug.

**Figure 1-1.** SmartDebug Top-Level Blocks




**Important:** The FHB feature is not enabled in the demo design. To enable FHB debugging, provide the "FHB_ENABLE" argument with the given TCL script. This creates the design without the DDR controller block, because FHB debugging is currently not supported for designs that include a DDR controller. For more information about enabling FHB debugging using the TCL script, see Appendix 4: Running the TCL Script.

The top level block contains the following blocks:

- PF_CCC
- PF_INIT_MON
- reset_des_sync
- XCVR_Debug
- Fabric_Debug
- DDR Interface

### 1.3.1. PF_CCC (Ask a Question)

The PF_CCC block generates 125 MHz clock. Fabric_Debug logic works on this clock.

### 1.3.2. PF_INIT_MON (Ask a Question)

The PF_INIT_MON block checks the status of device initialization. When the initialization of SRAM and µPROM is completed, the IP asserts DEVICE_INIT_DONE signal.


© 2025 Microchip Technology Inc. and its subsidiaries

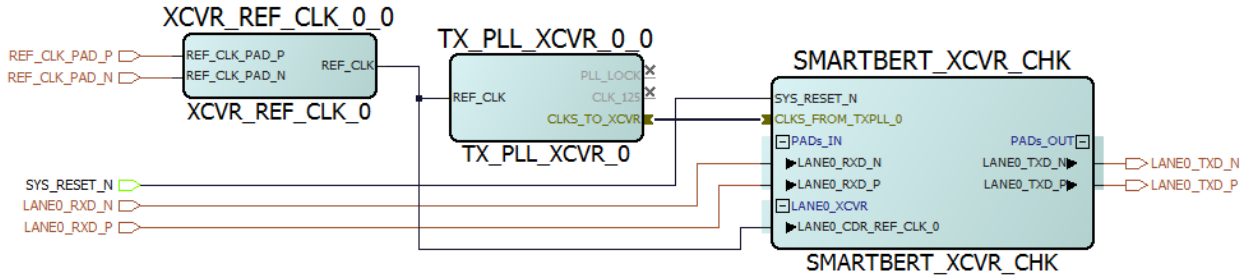
### 1.3.3. reset_des_sync (Ask a Question)

The reset_des_sync_0 block is an instantiation of CoreRESET_PF IP. It synchronizes the de-assertion of asynchronous reset.

### 1.3.4. XCVR_Debug (Ask a Question)

The following figure shows the IP blocks inside the XCVR_Debug block. The XCVR_Debug block demonstrates SmartDebug's real-time Signal Integrity (SI) testing and debugging capabilities to test and debug the PolarFire transceiver. This block contains CoreSmartBERT, TX_PLL, and XCVR_REF_CLK IP cores. CoreSmartBERT implements the PolarFire transceiver in the PMA mode.

**Figure 1-2.** XCVR_Debug Overall Design Blocks

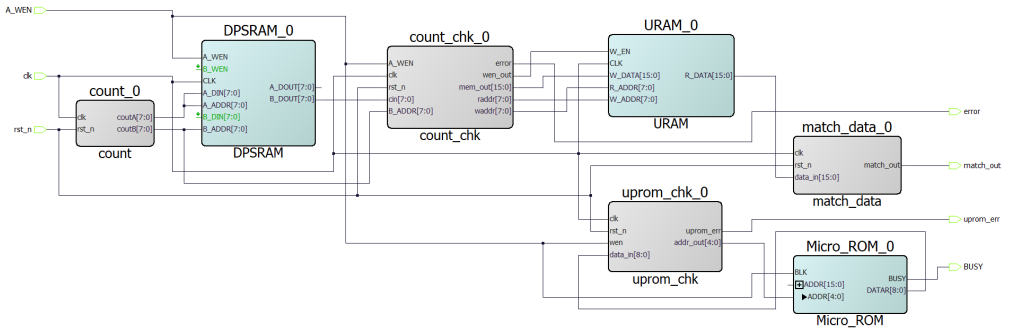



**Important:** SmartBERT includes the PolarFire Transceiver, which interfaces to the SmartDEBUG tool through a user control GUI to run the hardened PRBS generator and checkers. It also has fabric pattern generators and checks with more features like error injection.

### 1.3.5. Fabric_Debug (Ask a Question)

The following figure shows the IP blocks inside the Fabric_Debug block.

**Figure 1-3.** Fabric_Debug Overall Design Blocks




The Fabric_Debug block demonstrates the following FPGA fabric debug features of SmartDebug.

- FPGA array debugging capabilities using a counter that loads a counting pattern into the DPSRAM instance. The data value of the DPSRAM block is the same as the address value of the block. On the read side of the DPSRAM, a count checker (count_chk) ensures that the count progresses as expected. The output (error) is driven high if there is an error.
- µPROM debugging feature of SmartDebug using a µPROM instance.


© 2025 Microchip Technology Inc. and its subsidiaries

- Live probes to monitor an internal user-selected point on the device in real-time, and how to set active probes for dynamic, asynchronous read and write to a flip-flop or probe point. These features help to quickly observe the output of the logic internally or experiment to determine how the logic is affected by writing to a probe point.
- Capabilities to read and modify fabric SRAM content in real-time.

**µPROM**: This is the embedded non-volatile PROM arranged in a single row at the bottom of the fabric and is read-only through the fabric interface. µPROM is programmed with the FPGA bitstream during fabric programming. µPROM stores the initialization data for DPSRAM and µSRAM and other user data. µPROM is initiated with the `uprom.mem` file.

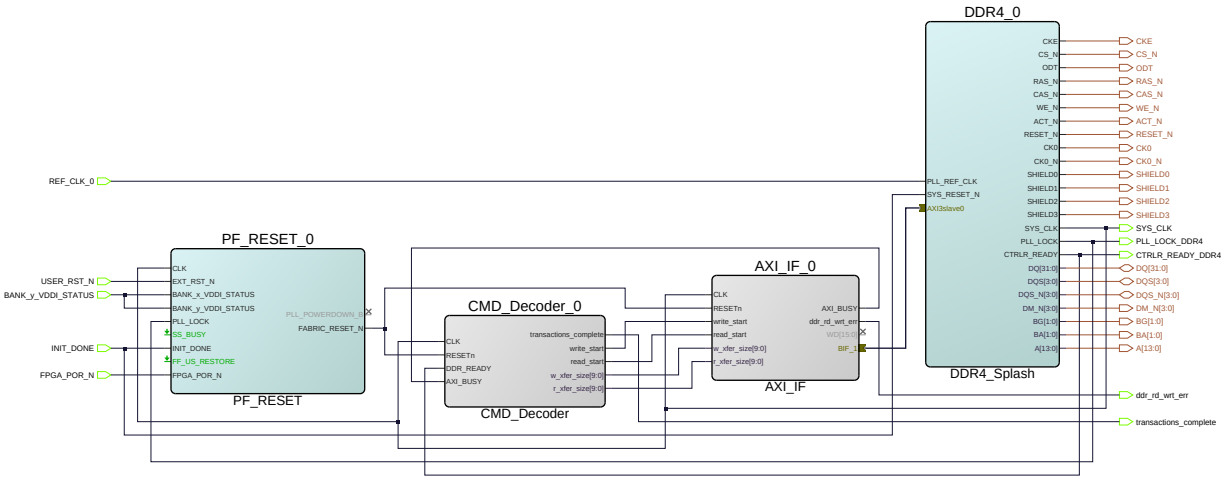
**µSRAM**: This is the fabric RAM block that is accessed using the PF_SRAM_AHBL_AXI IP. Generally, µSRAM is initialized with a user application executable at device power-up. In the example design, µSRAM is initialized with the `sram.hex` file.

### 1.3.6. DDR Interface (Ask a Question)

The DDR_Interface block demonstrates Debug DDR IO Margin feature in SmartDebug. Select the **Debug DDR Memory** option in the main SmartDebug window. **Debug DDR Memory** is available only for DDR3/DDR4/LPDDR3 memory configurations on PolarFire devices. This option is not visible when DDR memory is not used in the design.

**Important:** For detailed information on DDR Debug, see SmartDebug User Guide.

**Figure 1-4.** DDR Interface Overall Blocks



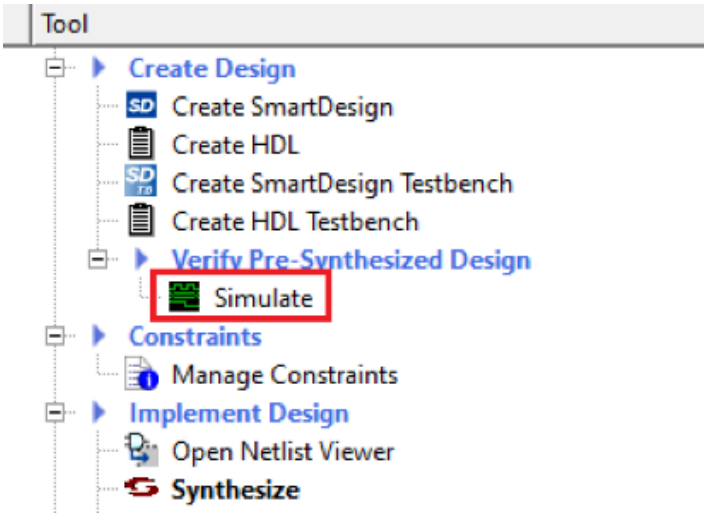

### 1.3.6.1. Simulation Using Micron DDR4 SDRAM Model (Ask a Question)

To simulate the DDR4 model, perform the following steps:

1. The PolarFire® Evaluation Kit features the DDR4 SDRAM from Micron with the MT40A1G8WE083E part number. The DDR4 simulation model files are available at the design files path `mpf_an4594_df\HW\src\stimulus`.
2. To run the simulation, navigate to **Design Flow** > **Verify Pre-Synthesized Design** and double-click the **Simulate** option, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

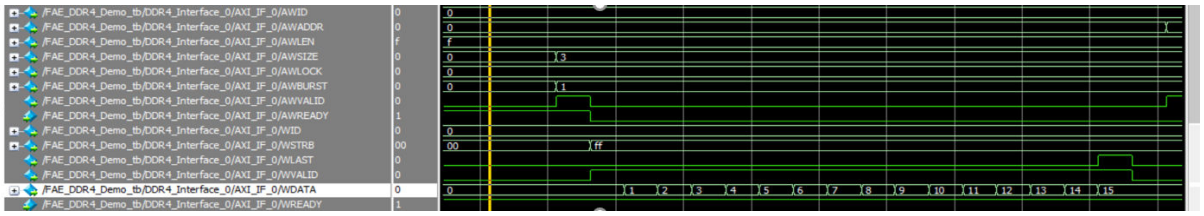
**Figure 1-5.** Simulating Pre-Synthesized Design




The AXI_IF block initiates 16 bursts (each burst has 16 operations of read and write) of reads and writes to DDR4 memory through the CMD_Decoder block.

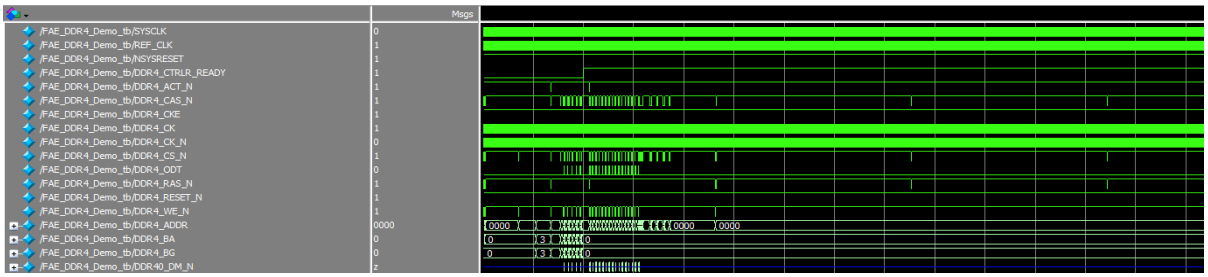
The following figure shows the AXI Master signal write operation.

**Figure 1-6.** AXI Master Signal Write Operation



The following figure shows the DDR4 signals.

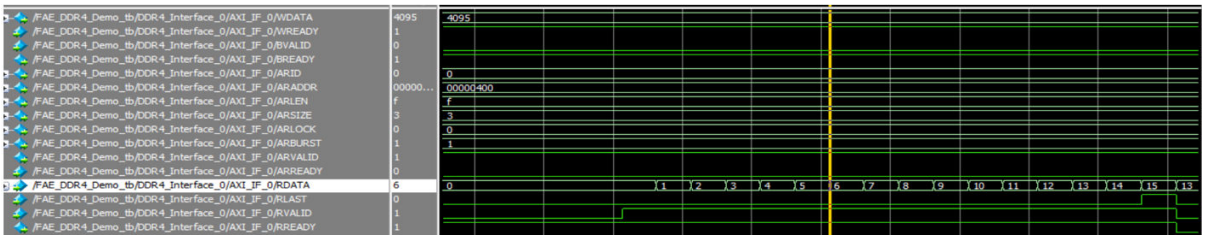
**Figure 1-7.** DDR4 Signals



The following figure shows the AXI Master signal read operation.


© 2025 Microchip Technology Inc. and its subsidiaries

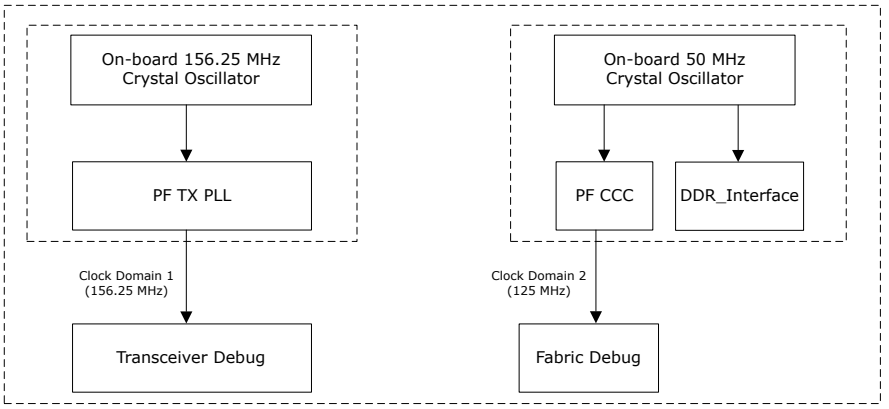
**Figure 1-8.** AXI Master Signal Read Operation




## 1.4. Clocking Structure (Ask a Question)

The reference design has two clock domains. Clock domain 1 is used for transceiver debugging and runs at 156.25 MHz. Clock domain 2 is used for fabric debugging and runs at 125 MHz, as shown in the following figure.

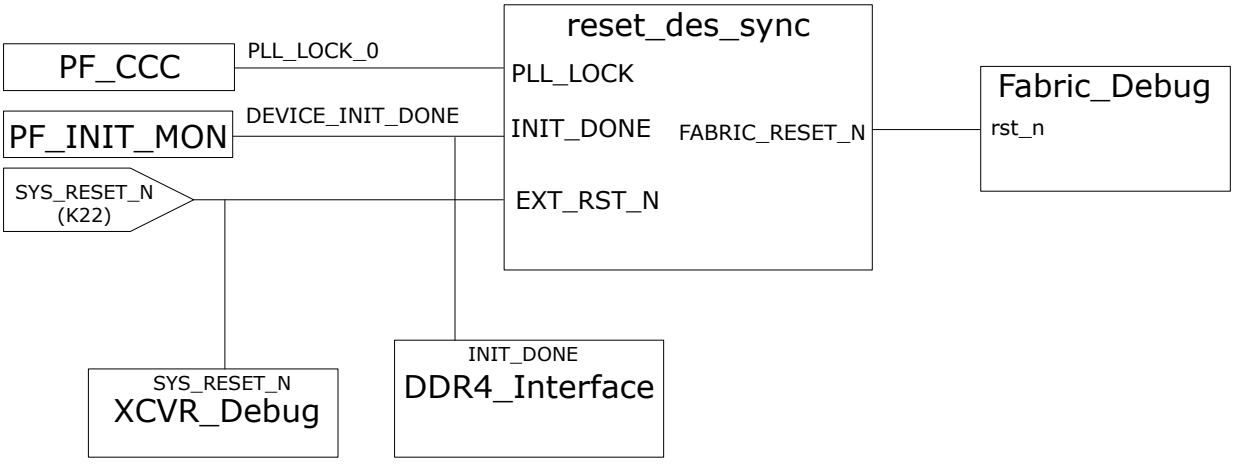
**Figure 1-9.** Clocking Structure




## 1.5. Reset Structure (Ask a Question)

The following figure shows the reset structure used in the design.

**Figure 1-10.** Reset Structure





© 2025 Microchip Technology Inc. and its subsidiaries

## 1.6. Enabling FPGA Hardware Breakpoint (FHB) (Ask a Question)

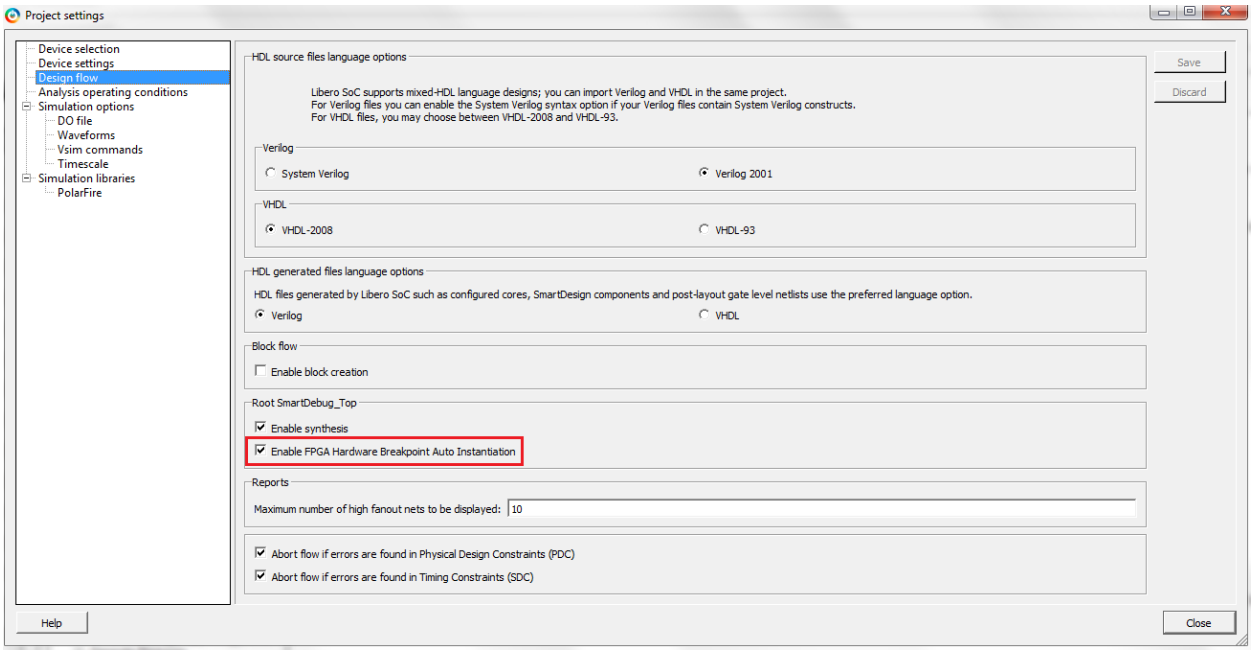
A Libero® SoC project with FHB enabled is created using the TCL script. A TCL script is provided in the design files folder in the `HW` directory to create a Libero SoC Project with FHB enabled.

To create the Libero project with FHB enabled, perform the following steps:

1. Launch Libero SoC.
2. Select **Project** > **Execute Script…**.
3. Select **Browse** and then select `script.tcl` from the downloaded `HW` directory.
4. In the **Argument** tab, provide the `FHB_ENABLE` argument.
5. Click **Run**.

After successfully executing the TCL script, the Libero SoC project is created within the `HW` directory. To confirm that the FHB is enabled, navigate to **Project** > **Project Settings** and verify that the **Enable FPGA Hardware Breakpoint Auto Instantiation** option is selected, as shown in the following figure.

**Figure 1-11.** Enabling FHB




## 1.7. Programming the Device (Ask a Question)

The following steps describe how to program the device on a PolarFire Evaluation Kit.

1. Ensure that the following jumper settings are followed.

> **Important:** Power-down the board before making the jumper connections.

**Table 1-2.** Jumper Settings For Evaluation Kit

| Jumper | Description |
|---|---|
| J46 | Short pin 1 and 2 to set the Reference Clock to 125 MHz on-board oscillator |
| J18, J19, J20, J21, and J22 | Short pin 2 and 3 to program the PolarFire FPGA through FTDI |
| J28 | Short pin 1 and 2 to program through the on-board FlashPro5 |

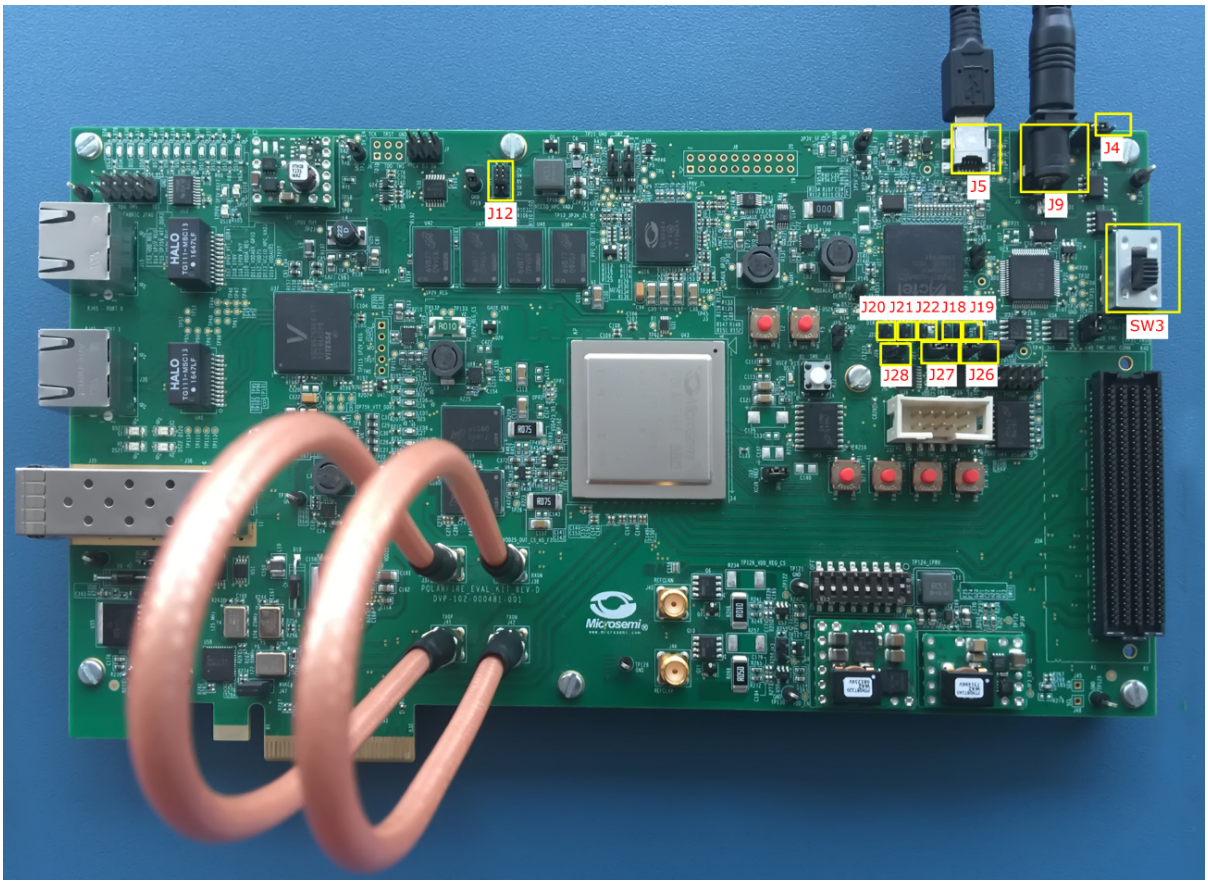

© 2025 Microchip Technology Inc. and its subsidiaries

**Table 1-2.** Jumper Settings For Evaluation Kit (continued)

| Jumper | Description |
|---|---|
| J4 | Short pin 1 and 2 for manual power switching using SW3 |
| J17 | Short pin 1 and 2 |
| J12 | Short pin 3 and 4 for 2.5V |

2. Connect the power supply cable to the J9 connector on the board.
3. Connect the USB cable from the Host PC to the J5 connector (FTDI port) on the board.
4. Power on the board using the SW3 slide-switch.
5. Switch OFF the DIP1 switch.
6. Connect **TXN** to **RXN** and **TXP** to **RXP** using two SMA to SMA cables, as shown in the following figure. The following figure shows the board setup.

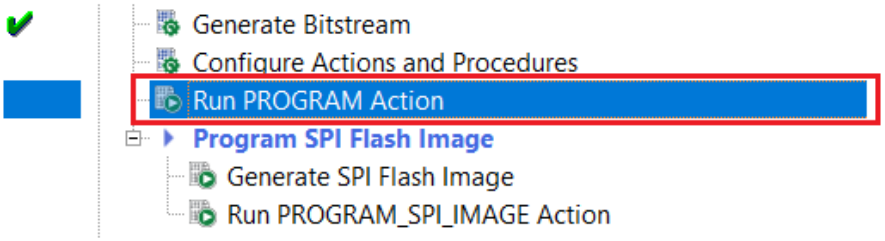
**Figure 1-12.** Board Setup (Evaluation kit)




7. In the **Design Flow** window, select **Run PROGRAM Action**, as shown in the following figure. This programs the design into the device.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-13.** Programming the Device




## 1.8. Debugging Using SmartDebug (Ask a Question)

To debug the device using SmartDebug, follow these steps:

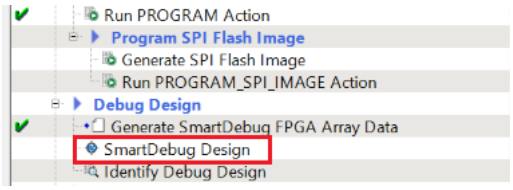
- Launch SmartDebug from Libero
- View Device Status
- Debug FPGA Array
- Using FHB
- Debug µPROM
- sNVM Debug
- Debug TRANSCEIVER
- Debug DDR IO Margin

### 1.8.1. Launch SmartDebug from Libero (Ask a Question)

On the **Design Flow** window:

1. Select **Generate SmartDebug FPGA Array Data** to generate data for SmartDebug Design. Once the data is generated, a green tick mark is seen on the left side of the option indicating that the data generation is successful.
2. Open **SmartDebug Design**.

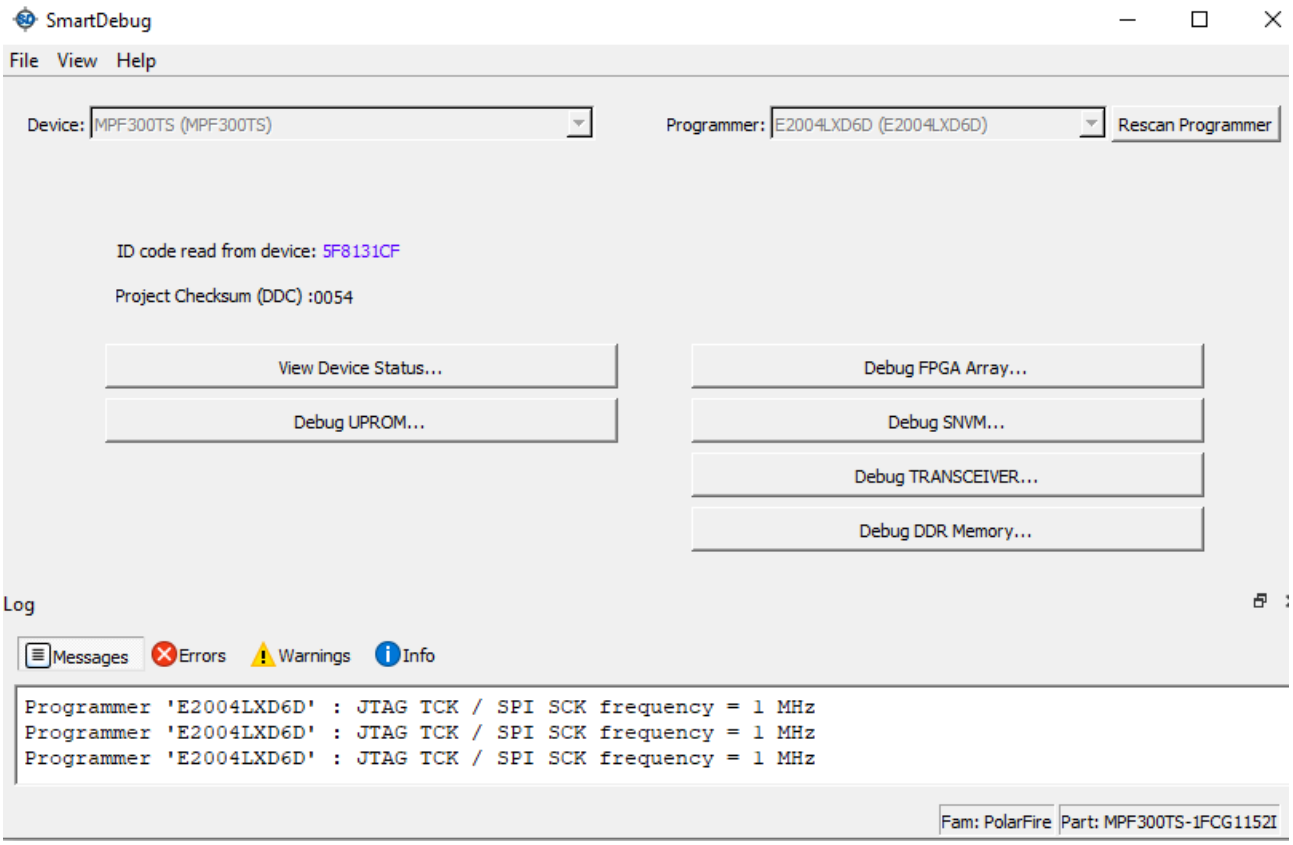
**Figure 1-14.** Launching SmartDebug Design




The **SmartDebug** window is displayed, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-15.** SmartDebug Window Debug Options

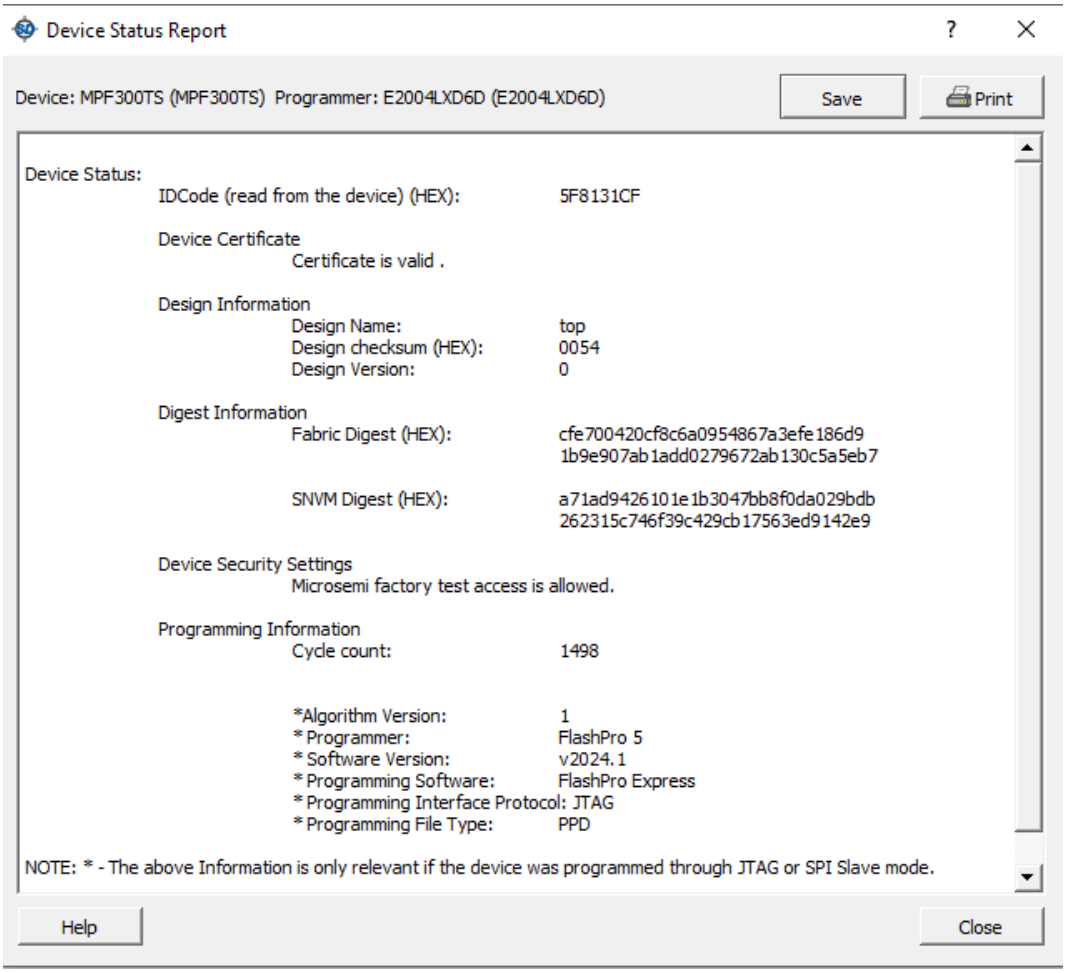



### 1.8.2. View Device Status (Ask a Question)

The View Device Status option provides the device status report. It summarizes device information, programmer information, design information, the factory serial number, and any security information set. To view the device status report, click **View Device Status** in the **SmartDebug** window. The following figure shows a sample of the device status information.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-16.** Device Status Report Sample




### 1.8.3. Debug FPGA Array (Ask a Question)

The Debug FPGA Array provides an interface to probe the user logic implemented in the FPGA's Logic Elements (LEs) using active and live probes, and read-write access to the fabric flip-flops, and read-write access to the memories implemented using DPSRAMs/URAMs. Probe insertion allows the assignment of the internal signals to the assigned or unassigned pins. These signals can be monitored using the oscilloscope in real-time. The Debug FPGA Array supports the following four features:

- Live Probes
- Active Probes
- Memory Blocks
- Probe Insertion

#### 1.8.3.1. Live Probes (Ask a Question)

Live Probes monitor two internal signals simuntaneously in the design without repeating the place and route. PolarFire devices have two dedicated live probe channels (for example, pin H6 and G6 of PolarFire MPF300TS device). For more information about Live Probes, see SmartDebug User Guide.

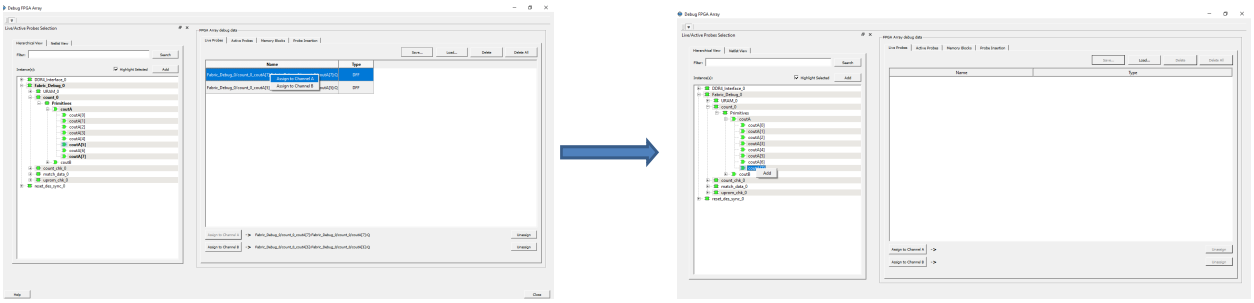
The following steps explain the procedure of adding a probe point to a list:

1. Select the **Live Probes** tab in the right pane. The probe signals are displayed in the left pane.


© 2025 Microchip Technology Inc. and its subsidiaries

2. Select the probe points you want to add from the **Hierarchical View** or **Netlist View** in the left pane.
3. Right-click on the selected points and click **Add** to add them to the **Live Probes**. You can also add the selected probe points by clicking **Add** in the top-right corner of the left pane. The probe signals can be filtered with the **Filter** option.
4. Select any of the added probes and assign them to either Channel A or Channel B (by clicking on 'Assign to Channel A' or 'Assign to Channel B'), as shown in the following figure.
5. When the assignment is complete, the probe name appears to the right of the button for that channel, and SmartDebug configures Channel A and Channel B I/Os to monitor the desired probe points.
6. Once the probe points are assigned, the probes can be monitored by connecting the probe points (for example, pin H6 and G6) to the oscilloscope.

**Figure 1-17.** Debug FPGA Array-Active Probes



### 1.8.3.2. Active Probes (Ask a Question)

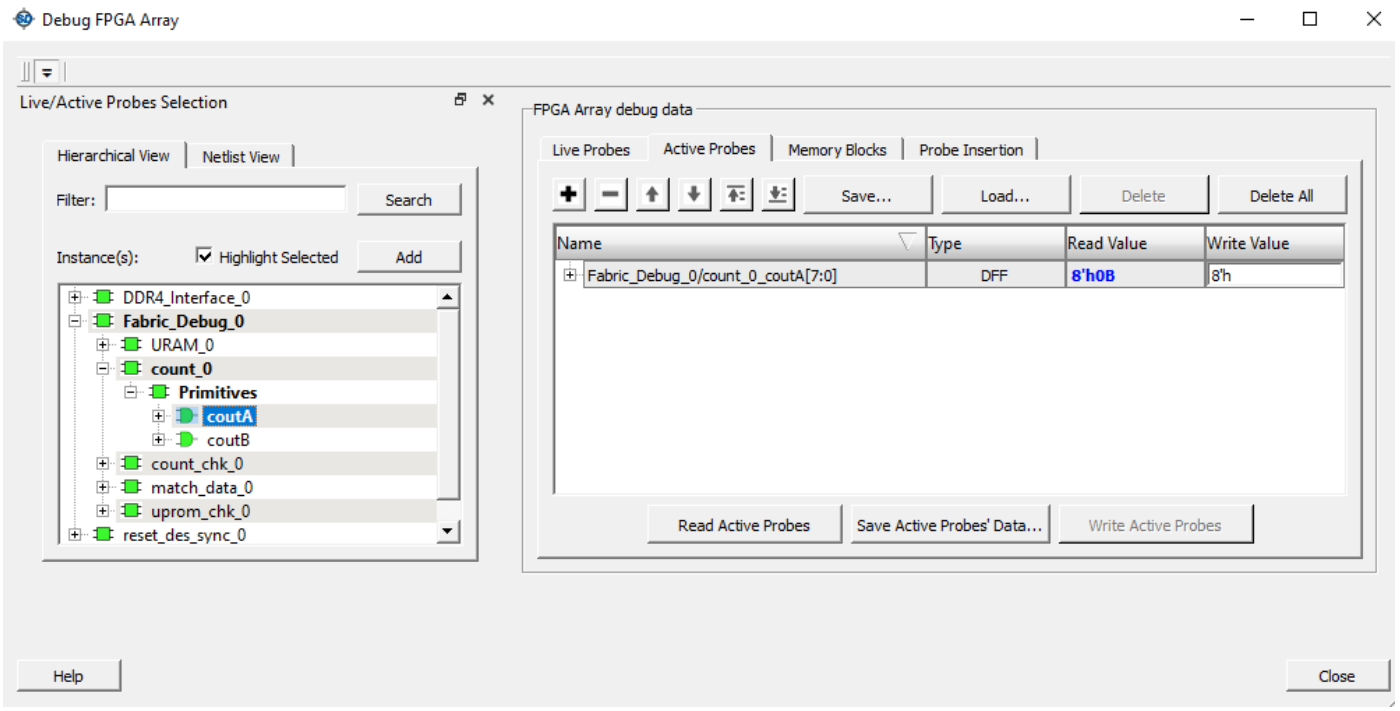
Active Probes enable reading or changing the values of probe points in a design through JTAG. Active Probes dynamically and asynchronously read or write to any logic element register bit. Active probes are useful for quick observation of an internal signal. For more information about Active Probes, see SmartDebug User Guide.

To add probe points to a list, perform the following steps:

1. Select the **Active Probes** tab in the right pane. The probe signals are displayed in the left pane.
2. Select the probe points you want to add from the **Hierarchical View** or **Netlist View** in the left pane.
3. Right-click the selected points and click **Add** to add them to the Active Probes. You can also add the selected probe points by clicking **Add** in the top-right corner of the left pane. The probes signals can be filtered with the Filter option.
4. Click **Read Active Probes** to read the content of the registers added to the window.

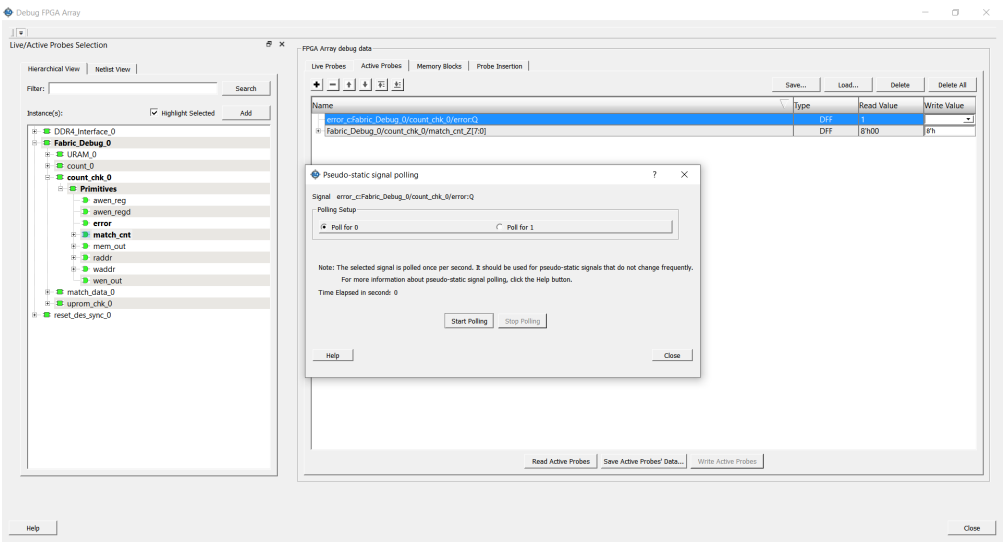

© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-18.** Debug FPGA Array—Active Probes




5. To use pseudo-static signal polling, right-click any probe point on the **Active Probes** tab and select **Poll**, as shown in the figure below.
   Static signal polling is used to check whether the logical bit value is changed to the expected polled value. For more information on polling, see SmartDebug User Guide.

**Figure 1-19.** Pseudo-static Signal Polling



### 1.8.3.3. Memory Blocks (Ask a Question)

SmartDebug provides the **Memory Blocks** tab to dynamically and asynchronously read from and write to a selected FPGA fabric SRAM block. For more information about the **Memory Blocks** tab, see SmartDebug User Guide. Using the **Memory Blocks** tab, the user can select the required memory block to perform the following:

- Reading

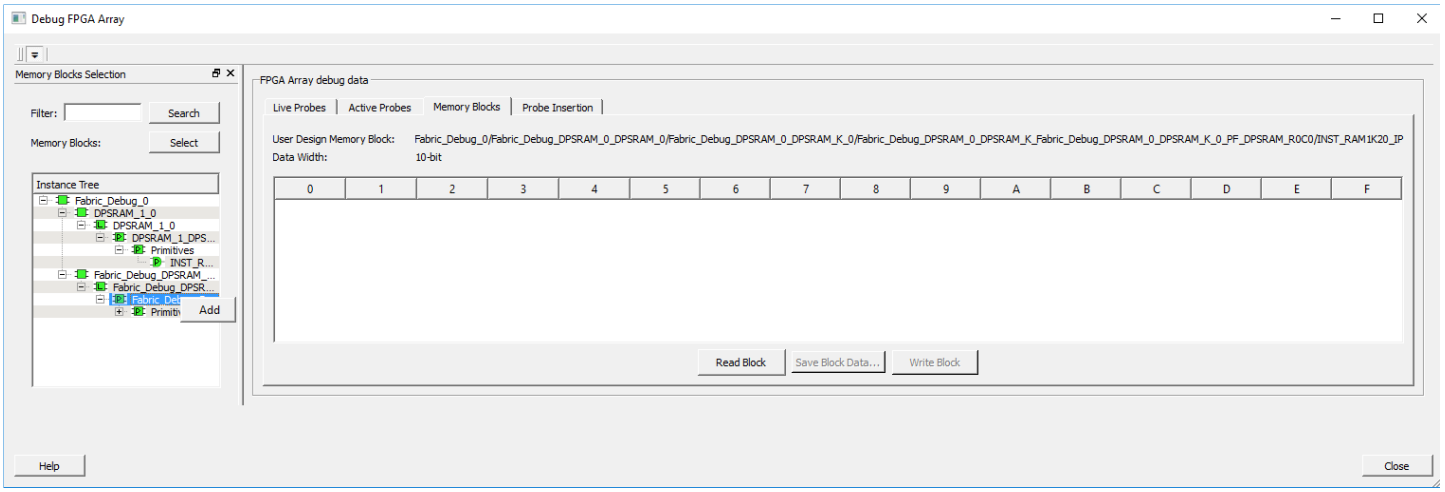

© 2025 Microchip Technology Inc. and its subsidiaries

- Capturing a snapshot of the memory
- Modifying memory values, and then write the values back to that block

To read and write memory blocks, follow these steps:

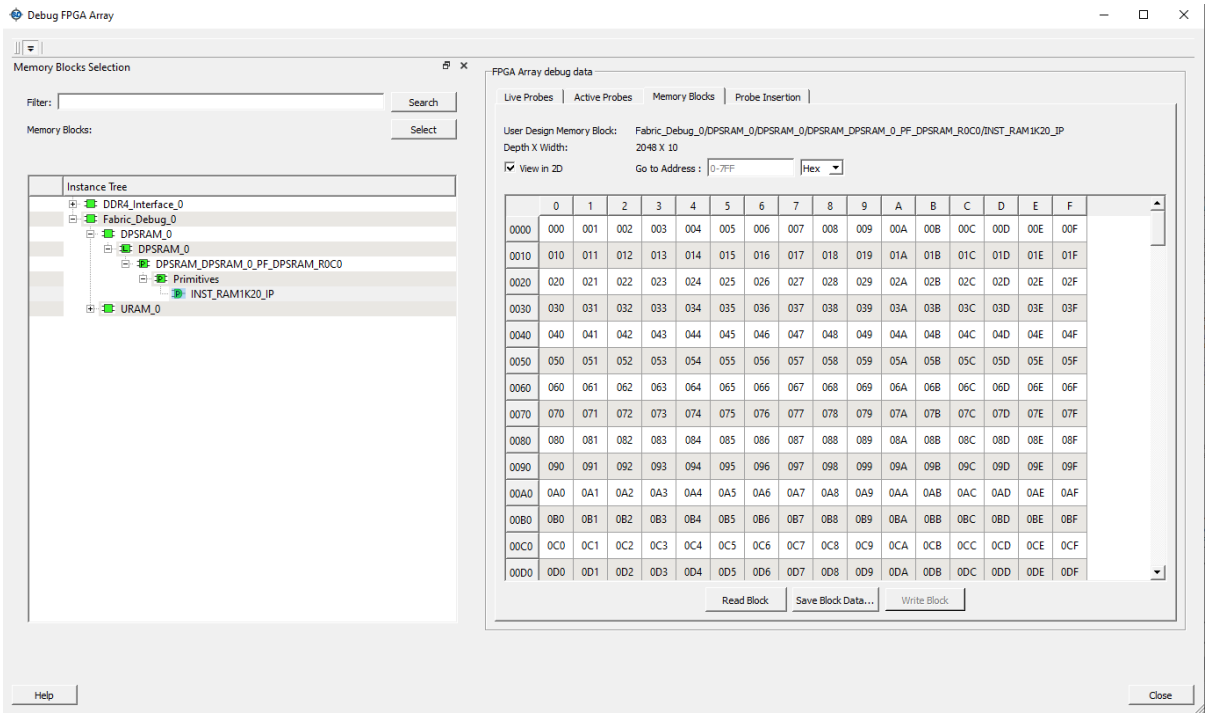
1. Select the **Memory Blocks** tab in the right pane of the **SmartDebug** window.
2. View the memory blocks in the left pane in the **Hierarchical View**.
3. Select the memory block in the left pane and click select in the top-right corner of the pane.
4. Right click the selected memory block and click **Add**.
   The following figure shows the **Memory Blocks** tab in **Debug FPGA Array** window.

**Figure 1-20.** Debug FPGA Array—Memory Blocks




5. Click **Read Block**. The specified memory block is read, as shown in the following figure.

**Figure 1-21.** Memory Blocks-Read Block





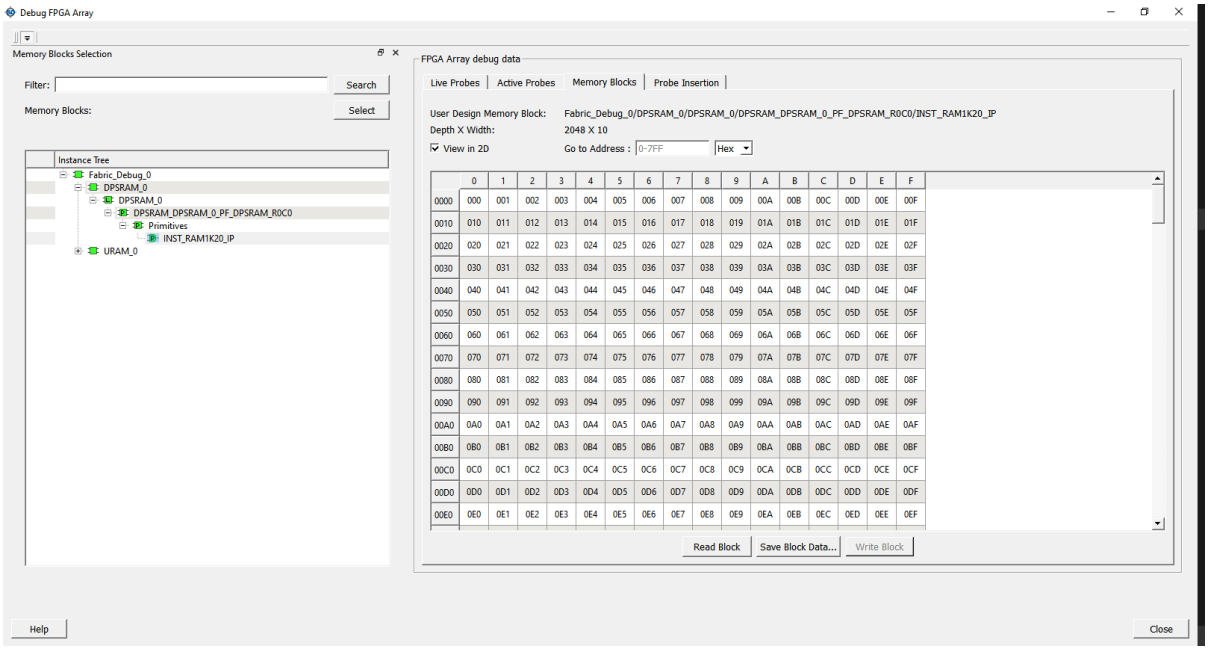
© 2025 Microchip Technology Inc. and its subsidiaries

6. Enter a hexadecimal value in the memory block locations and click **Write Block** to write content into memory.

> **Important:** The counter logic writes to SRAM constantly. Before you write to SRAM using SmartDebug, ensure that the A_WEN signal (DIP1 of SW11) is low (ON state). This prevents SRAM from being overwritten by the counter logic.

7. Switch On DIP1, enter a hexadecimal value in the memory block location(s) and click **Write Block** to write the modified value to the SRAM, as shown in the following figure.

**Figure 1-22.** Memory Blocks-Write Block



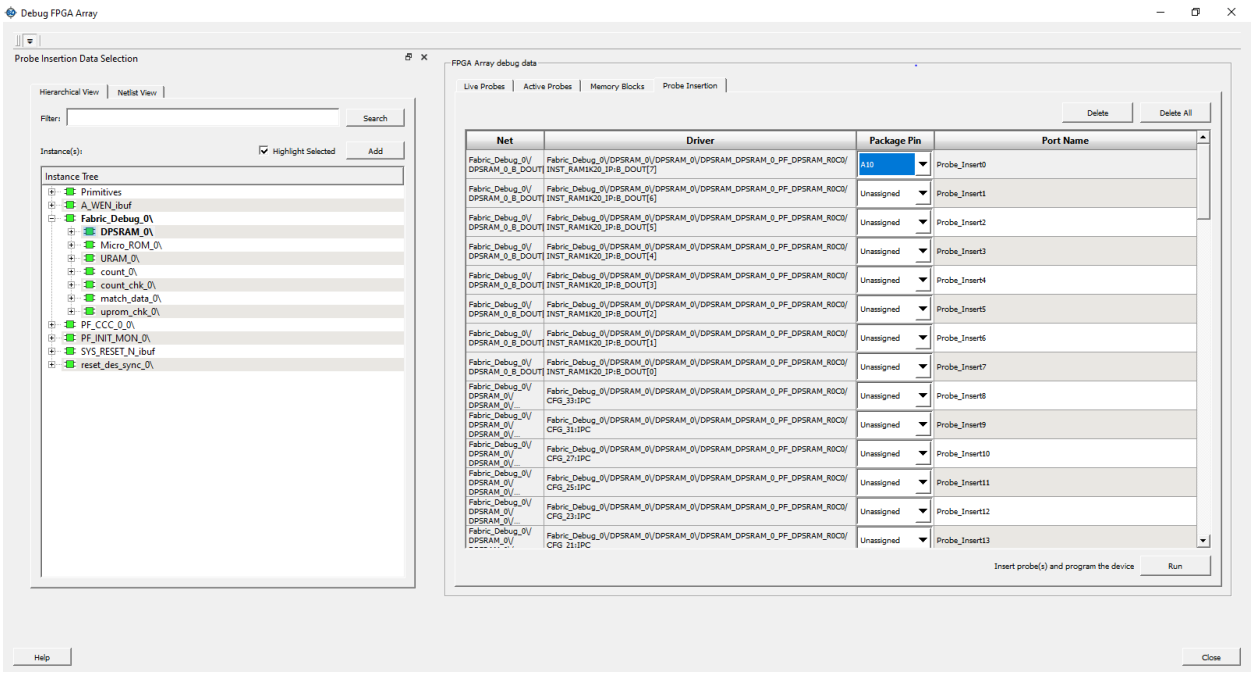

8. The error LED4 (F22) light turns on, indicating an error in the counting pattern.
9. Go to **Active Probes** tab, read the value of error signal, it must show '1'. To use static signal polling, right-click **error_c:Fabric_Debug_0/count_chk_0/error:Q** and select **Poll (Poll for 0)**, as shown in Figure 1-19.
10. Move DIP1 to off state to resume the write operation from the counter to the SRAM. This overwrites the error that was injected into the SRAM. Check the status of the LED, and it must turn off. Hit the **Poll for 0, User value match** message must appear on the polling window. Close the **Pseudo-static signal polling** window.
11. The content of the SRAM can be rechecked by clicking **Read Block** in the **Memory Blocks** tab.

### 1.8.3.4. Probe Insertion (Ask a Question)

Probe insertion is a post-layout debug process that enables internal nets in the FPGA design to be routed to unused or used I/Os. Nets are selected and assigned to probes using the **Probe Insertion** tab in SmartDebug. The rerouted design is reprogrammed automatically by Libero into the FPGA, where an external logic analyzer or oscilloscope can be used to view the activity of the probed signal. The following figure shows the **Probe Insertion** tab in the **Debug FPGA Array** window.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-23.** Debug FPGA Array—Probe Insertion




## 1.8.4. Using FHB (Ask a Question)

When FHB is enabled, an FHB instance is created on each clock domain of the design. Each FHB instance gates its associated clock domain. You can add a trigger signal (**countB[0]**) to a live probe and halt the design on the positive edge of the trigger.

When FHB is enabled, the following options are enabled on the Live Probes tab:

- **Event Counter**—Counts the transition of signals assigned to Channel A or Channel B through the Live Probe feature. This feature tracks events from the board. When the Event Counter is activated and a signal is assigned to Channel A, the counter starts counting the rising edge transitions. The counter must be stopped to get the final signal transition count. During the count, you cannot assign another signal to Channel A/Channel B or go to any other tab on the window.
- **Frequency Monitor**—Calculates the frequency of any signal in the design that can be assigned to Live Probe Channel A or Channel B. You can enter the duration of monitoring the signal. The accuracy of results increases as the monitor time increases. The unit of measurement is displayed in MegaHertz (MHz). During the run, progress is displayed.
- **User Clock Frequencies**—Shows the clock frequencies from the Clock Conditioning Circuit (CCC) block.

This section describes the following procedures:

- Use Event Counter
- Use Frequency Monitor
- Use User Clock Frequency
- Select FHB

### 1.8.4.1. Use Event Counter (Ask a Question)

This section describes the procedure to add a trigger signal (**coutA[7]**) to Live Probe to activate the event counter to count the rising edge transitions of that signal.

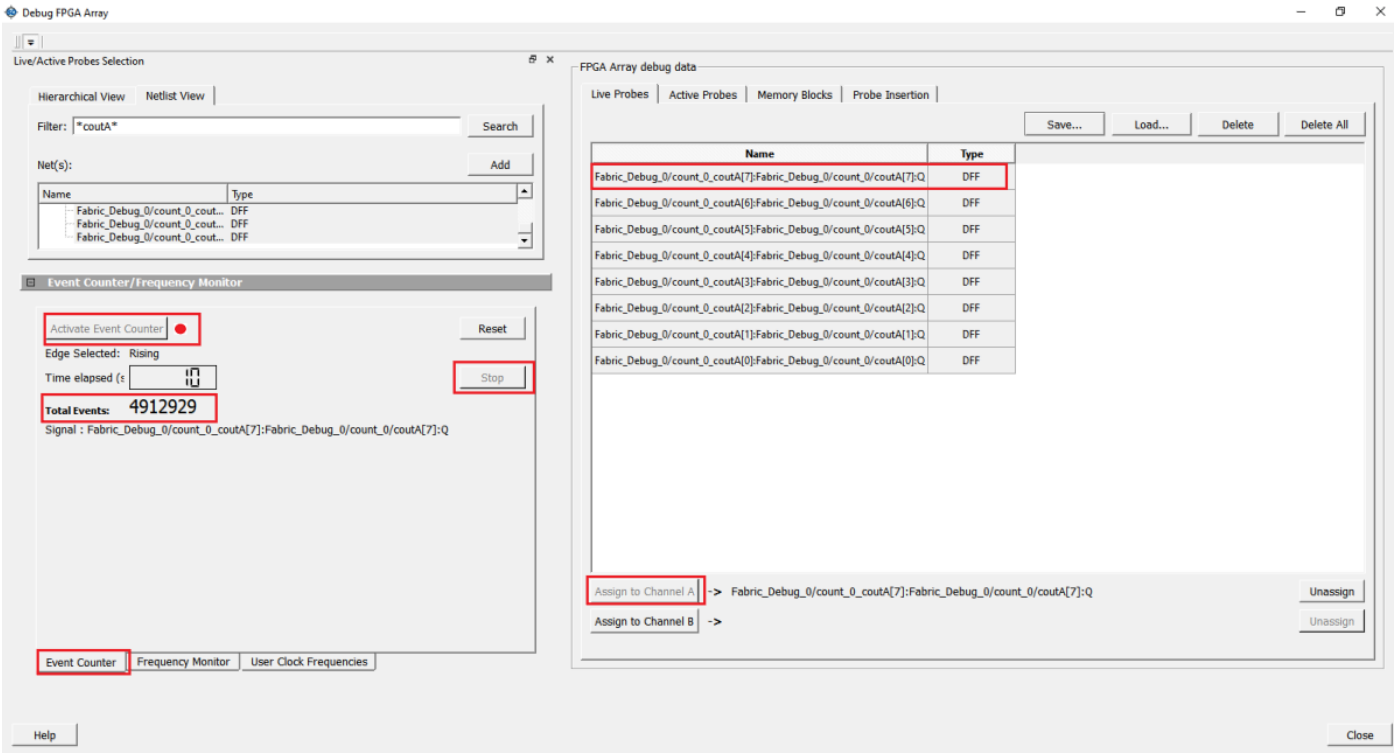
Follow these steps to activate the trigger counter:


© 2025 Microchip Technology Inc. and its subsidiaries

1. Go to the **Live Probes** tab, and enter ***coutA*** in the filter box, then click **Search**.
2. Select **coutA [7]** to **Cout[0]** and click **Add**, as shown in the following figure.
   **coutA [7]** is used as the hardware break point trigger.
3. Select **coutA [7]: Q**, then click **Assign to Channel A**.
4. Click **Activate Event Counter**.
5. The count is updated every second and is displayed as **Total Events**.
6. Click **Stop** button to stop counting, as shown in the following figure.

**Important:** When the Event Counter is running, only the Stop button is enabled.

**Figure 1-24.** Event Counter



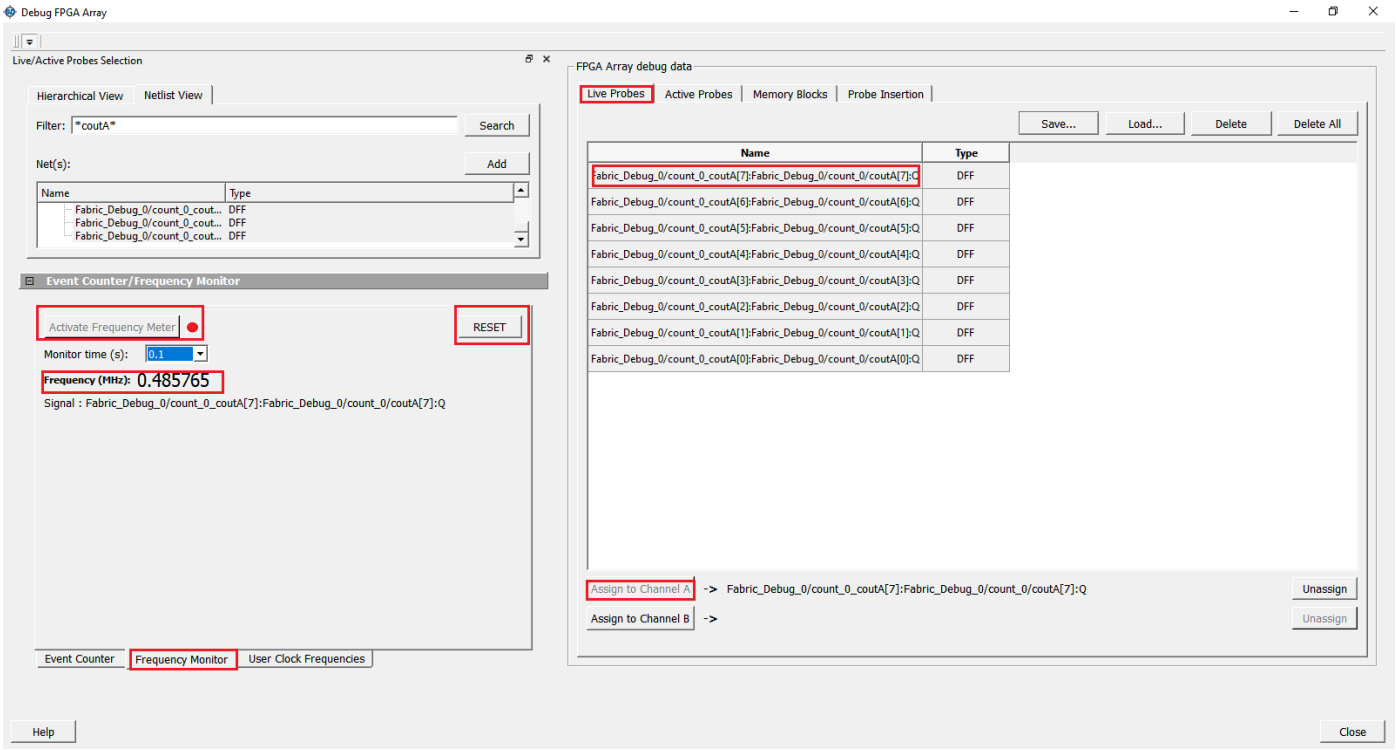

### 1.8.4.2. Use Frequency Monitor (Ask a Question)

To use Frequency Monitor, perform the following steps:

1. Click the **Live Probe** tab, assign a signal to Channel A and then click the **Frequency Monitor** tab.
2. Set Monitor Time(s) at 0.1 and select the **Activate Frequency Monitor** check box.
3. The Frequency Monitor stops when the specified monitor time is over. The result is displayed as Frequency (MHz). The window and the tabs on the control panel are enabled. The Reset button is also enabled to reset the Frequency to 0 to start over the next iteration. The progress bar is hidden when the Frequency Monitor stops.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-25.** Frequency Monitor

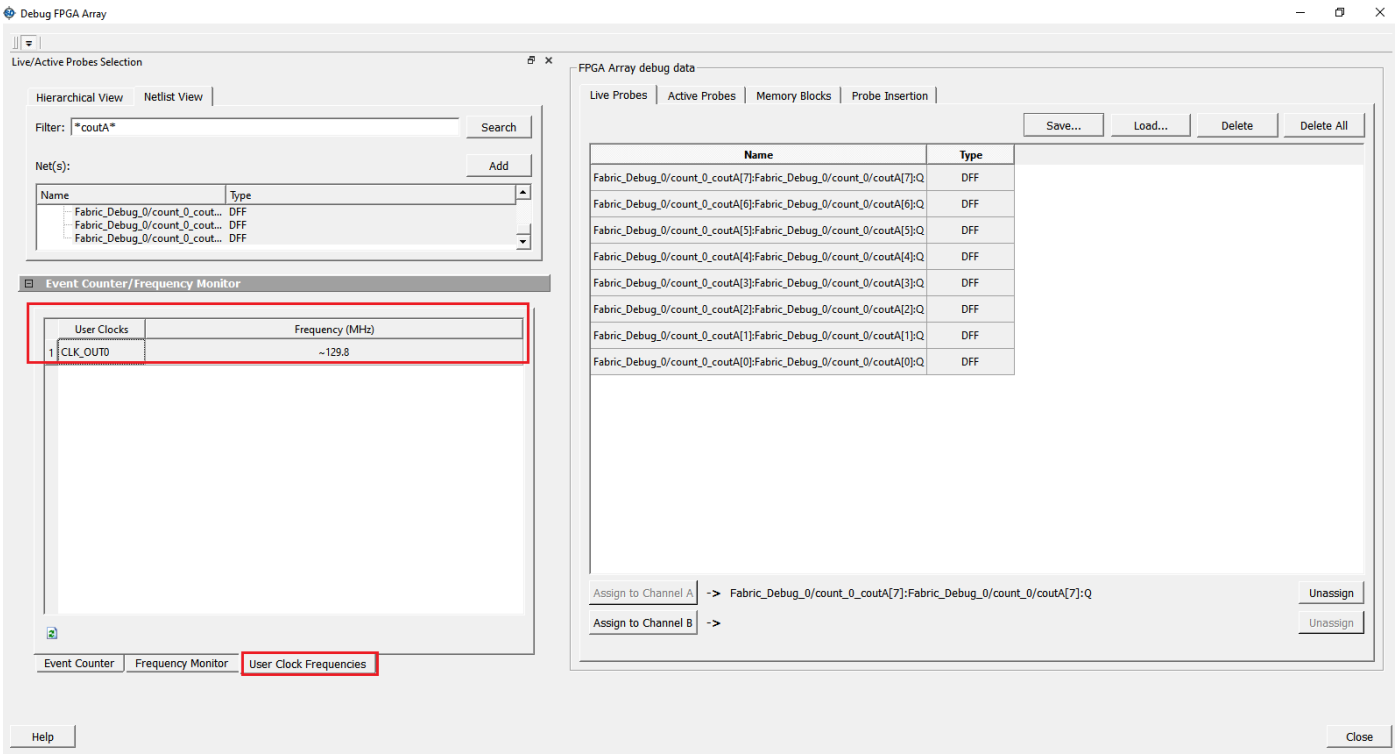



### 1.8.4.3. Use User Clock Frequency (Ask a Question)

All of the CCC clock frequencies are calculated by selecting the User Clock Frequencies tab, as shown in the following figure. Live probes are temporarily unavailable until all the user clock frequencies are calculated and displayed.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-26.** User Clock Frequencies




**Important:** The design includes one clock frequencies from PF_CCC component.

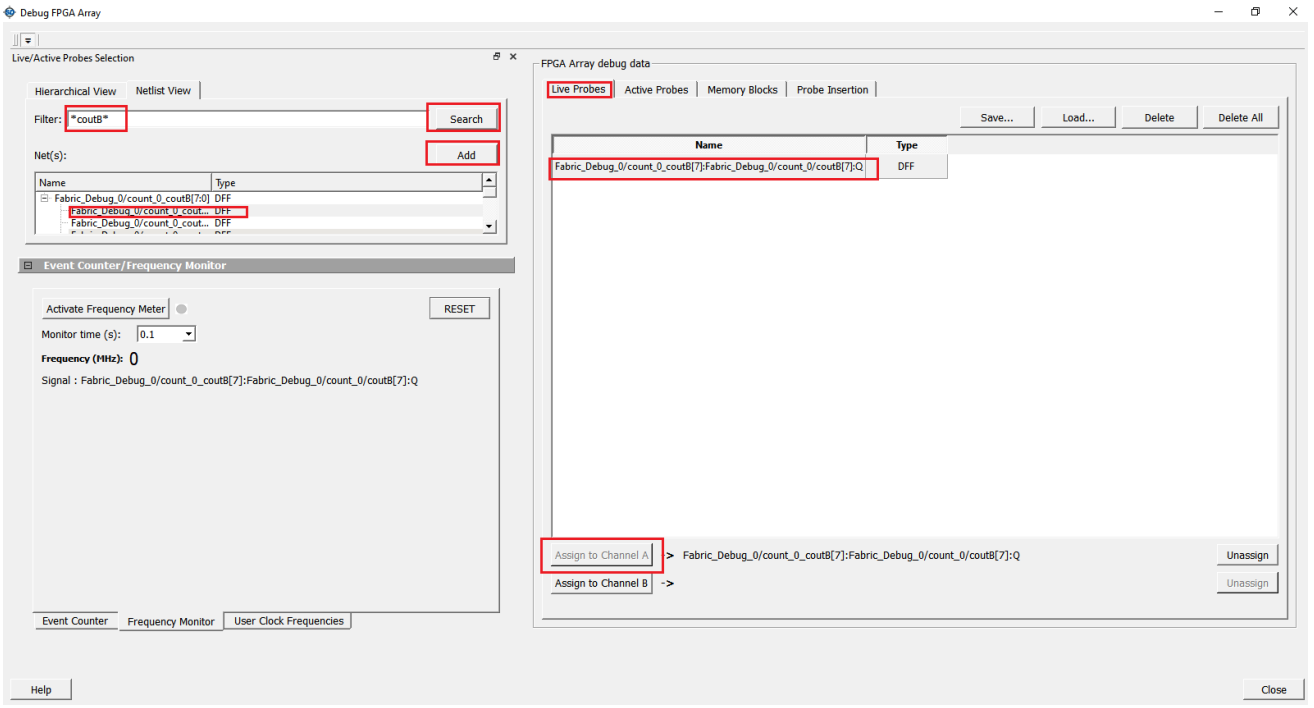
### 1.8.4.4. Select FHB (Ask a Question)

To select an FHB, perform the following steps:

1. Go to the **Live Probes** tab and enter ***coutB*** in the filter box, then click **Search**.
2. Select **coutB [0]** and click **Add**, as shown in the following figure.
   **coutB [0]** is used as the hardware break point trigger.
3. Select **coutB [0]: Q**, and click **Assign to Channel A**, as shown in the following figure.

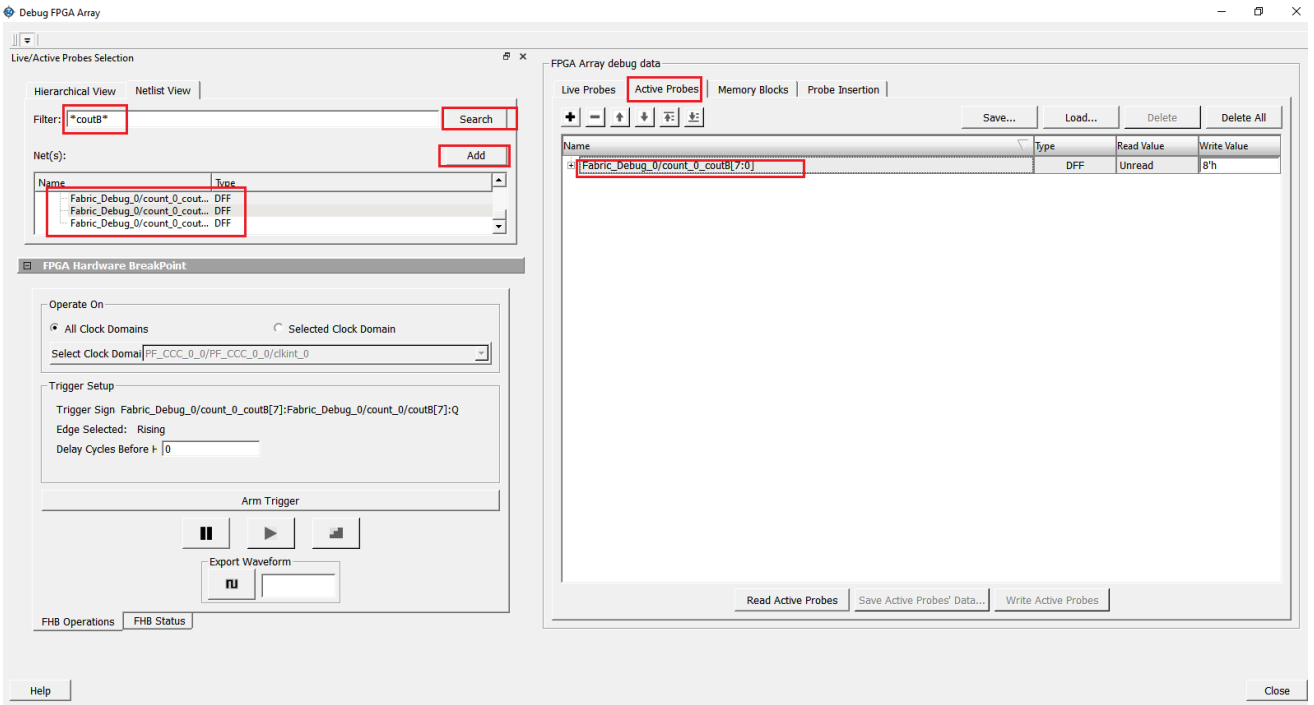

© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-27.** Select an FHB




4. Select the **Active Probes** tab, and search for ***coutB***.
5. Select **coutB [0]** to **coutB [7]** by holding the Shift key and click **Add**, as shown in the following figure.

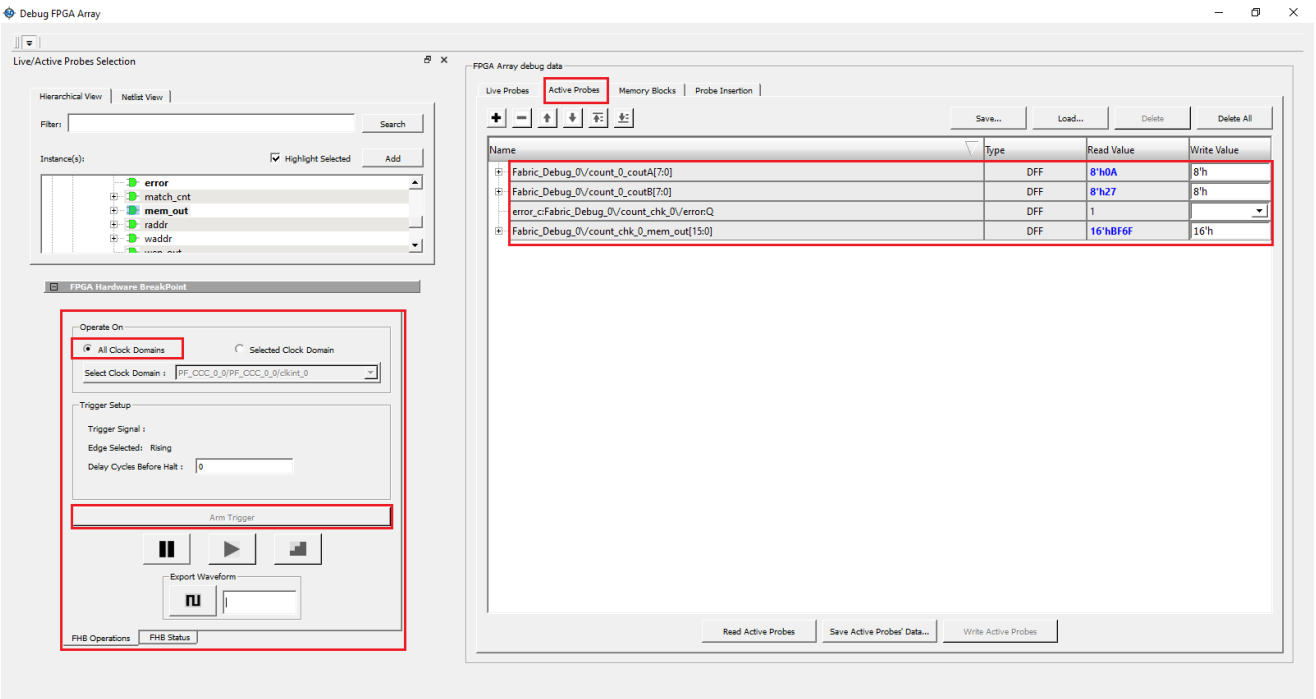
**Figure 1-28.** Adding FHB Trigger to Active Probes




6. Select **Operate on All Clock Domains**, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-29.** Selecting Clock Domains




7. Click **Arm Trigger**, as shown in the preceding figure. The counter halts on the next positive edge that occurs on the signal connected to Channel A (**coutB [0]**) in Live Probes.

**Important:** If you require a certain number of clock cycles before halting the clock domain after triggering, a value between 0 and 255 must be entered in **Delay Cycles Before Halt** before you click **Arm Trigger**. This sets the FHBs to trigger after the specified delay from the rising edge trigger.

The FHB controls are highlighted in the preceding figure, the following actions can be performed using these FHB controls:

1. Provide custom delay cycles before the halt.
2. Force a selected clock domain or all clock domains to halt without waiting for a trigger from a live probe signal by clicking the **Halt** button.
3. When the clock domain is in the halted state (live probe halt or force halt), resume the clock domain by clicking the **Play/Resume** button.
4. When the clock domain is in the halted state (live probe halt or force halt), advance the clock domain by one clock cycle and hold the state of the clock domain by clicking the **Step** button.
5. Save the waveform view of the selected active probes by specifying the number of clock cycles to capture in the Export Waveform text box and then click the **Capture Waveform** button. The waveform is saved as a `.vcd` file.
6. View the saved waveforms by importing the `.vcd` file. The waveform file can be viewed in a waveform viewer that supports the Value Change Dump (vcd) format.
7. Click **Close** to close the **Debug FPGA Array** window. Click **No** when prompted to save the active probes to a file.

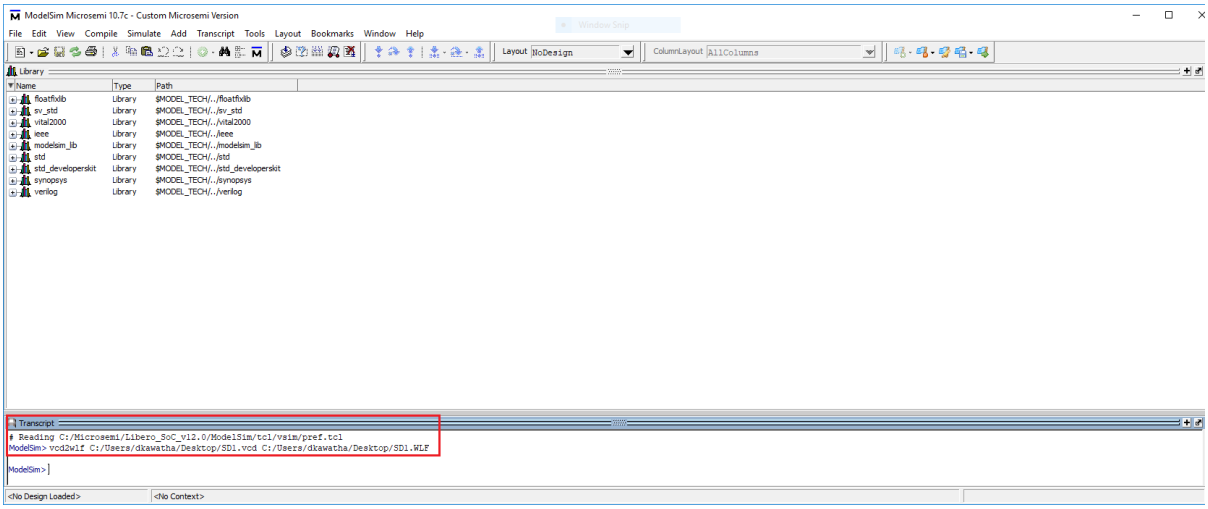
### 1.8.4.5. Opening VCD File in ModelSim (Ask a Question)

To view the signals exported by the SmartDebug in the `.vcd` file, use the Modelsim Waveform window:


© 2025 Microchip Technology Inc. and its subsidiaries

1. Open ModelSim.
2. Go to the Transcript window and convert VCD to WLF format using the following the `vcd2wlf <file1.vcd> <file2.wlf>` command as shown in the following figure.

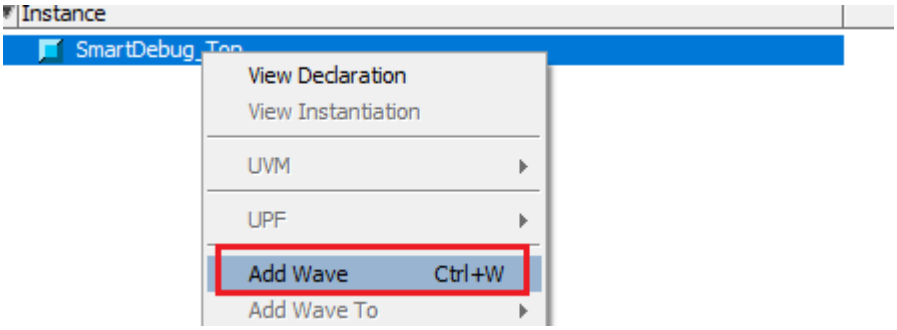
**Figure 1-30.** VCD to WLF Conversion




**Important:** Conversion errors are most often caused by a nonexistent instance path. Ensure that the instance paths specified are correct.

3. Open the WLF file created using File menu Open -> file2.wlf.
4. Select window with wlf file name and add signals to the Waveform window, as shown in the following figure.

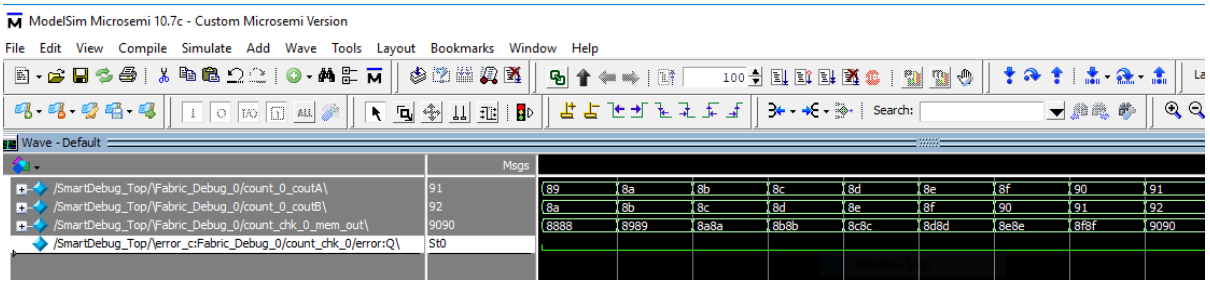
**Figure 1-31.** Adding Signals




5. Open the Wave window. Observe that the error signal is not asserted indicating data written and date read from DPSRAM is same, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-32.** Wave Window



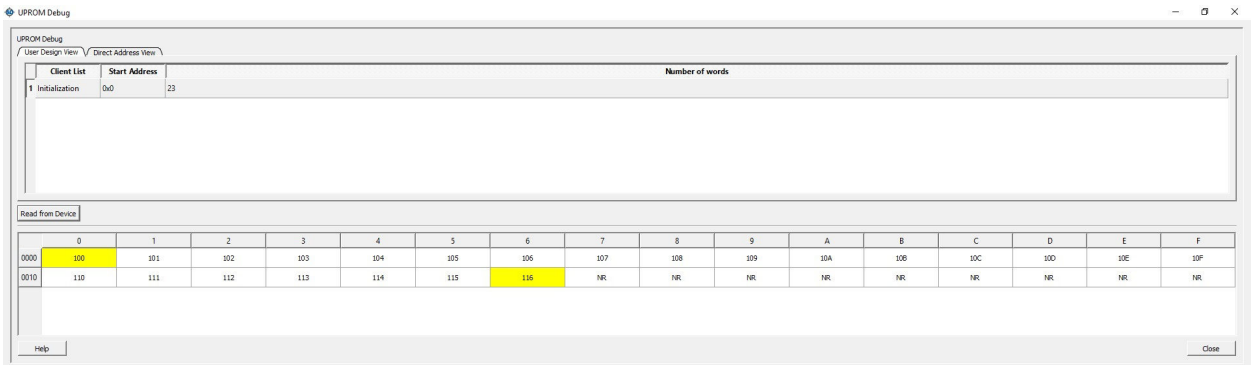

**Important:** The Fabric_Debug block includes the match_out signal, which must always be high, indicating that the data expected from DPSRAM and the data read from DPSRAM matches. But the provided design has a bug due to which the match_out signal always toggles. Debug the match_data.v logic using the FHB feature and find the route cause. Add match_out and mem_out of count_chk_0 block to Active probe and export the signals. Observe when the match_out signal is asserted to 1'b0.

### 1.8.5. Debug µPROM (Ask a Question)

SmartDebug enables debugging µPROM and reading its µPROM contents. The clients added in the design can be debugged using the SmartDebug Debug µPROM feature.

1. Click **Debug µPROM** in the **SmartDebug** window. The µPROM Debug window is shown in the Figure 1-33.
2. Select **MicroPROM_0** in the **User Design View** tab and then click **Read from Device** to read the µPROM content. Check whether the content provided in uprom.mem file (part of design stimulus files) matches with the data read from µPROM. You can check the highlighted locations 100 and 116 in Figure 1-33 to verify the content.

**Figure 1-33.** µPROM Debug




**Important:** PolarFire devices have a single user programmable read only memory (µPROM) row located at the bottom of the fabric, providing up to 459 Kb of non-volatile, read-only memory. The address bus is 16 bits wide, and the read data bus is 9-bit wide. µPROM is used to store the configuration data, which is used by Fabric logic to process.


© 2025 Microchip Technology Inc. and its subsidiaries

### 1.8.6. sNVM Debug (Ask a Question)

sNVM Debug feature enables reading from the sNVM during debug. Debug Pass Key is required to carry out SNVM_DEBUG instruction. This feature supports debugging of non-authenticated plain text, authenticated plain text, and clients cipher authenticated.

1. Click **Debug SNVM** in the **SmartDebug** window.
2. Click **Client View** tab. The client view details are listed—Client Names, Start Page, End Page, Number of Bytes, Write Cycles, Page Type, Used as ROM, and USK Status.
3. Select a client from the list in the **Client View** and click **Read from Device**, as shown in the following figure.

**Figure 1-34.** sNVM Debug

sNVM Debug

Client View | Page View

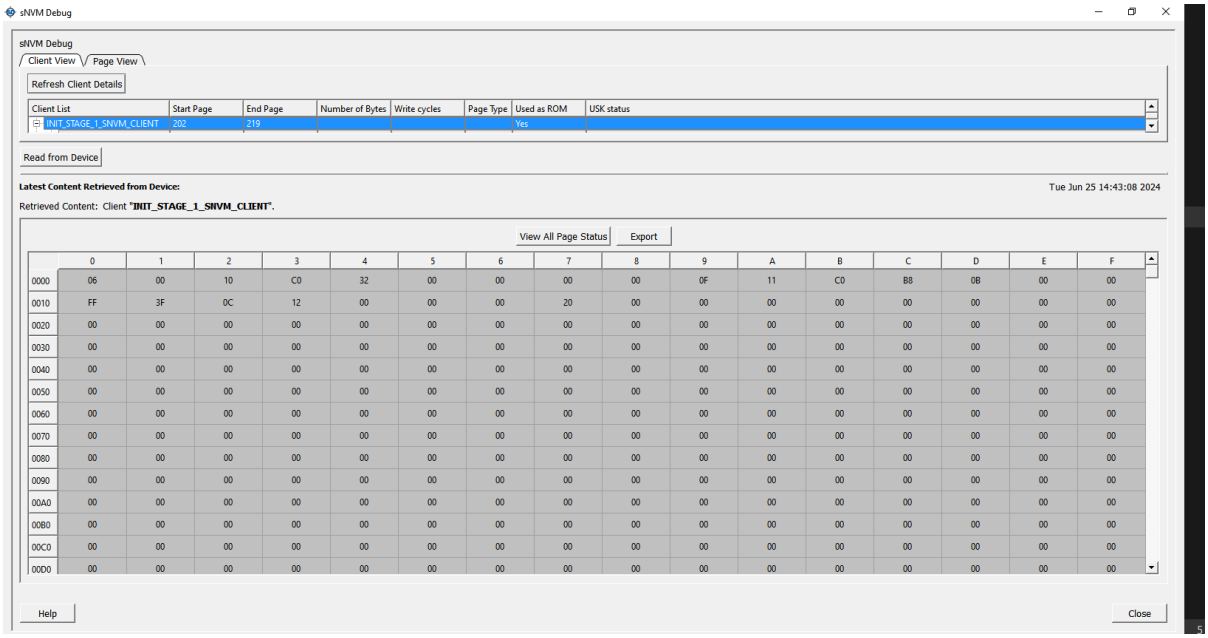
Refresh Client Details

| Client List | Start Page | End Page | Number of Bytes | Write cycles | Page Type | Used as ROM | USK status |
|---|---|---|---|---|---|---|---|
| INIT_STAGE_1_SNVM_CLIENT | 202 | 219 | | | | Yes | |
| Page 202 | 202 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 203 | 203 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 204 | 204 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 205 | 205 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 206 | 206 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 207 | 207 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 208 | 208 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 209 | 209 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 210 | 210 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 211 | 211 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 212 | 212 | | 252 | 117 | Plain Text | Yes | N/A |
| Page 213 | 213 | | 252 | 78 | Plain Text | Yes | N/A |
| Page 214 | 214 | | 252 | 78 | Plain Text | Yes | N/A |
| Page 215 | 215 | | 252 | 78 | Plain Text | Yes | N/A |
| Page 216 | 216 | | 252 | 78 | Plain Text | Yes | N/A |
| Page 217 | 217 | | 252 | 78 | Plain Text | Yes | N/A |
| Page 218 | 218 | | 252 | 101 | Plain Text | Yes | N/A |
| Page 219 | 219 | | 252 | 713 | Plain Text | Yes | N/A |
| INIT_STAGE_2_3_SNVM_CLIENT | 0 | 7 | | | | Yes | |
| Page 0 | 0 | | 252 | 634 | Plain Text | Yes | N/A |
| Page 1 | 1 | | 252 | 524 | Plain Text | Yes | N/A |
| Page 2 | 2 | | 252 | 524 | Plain Text | Yes | N/A |
| Page 3 | 3 | | 252 | 524 | Plain Text | Yes | N/A |
| Page 4 | 4 | | 252 | 451 | Plain Text | Yes | N/A |
| Page 5 | 5 | | 252 | 340 | Plain Text | Yes | N/A |
| Page 6 | 6 | | 252 | 339 | Plain Text | Yes | N/A |
| Page 7 | 7 | | 252 | 331 | Plain Text | Yes | N/A |

The following figure shows the Client View window.


© 2025 Microchip Technology Inc. and its subsidiaries

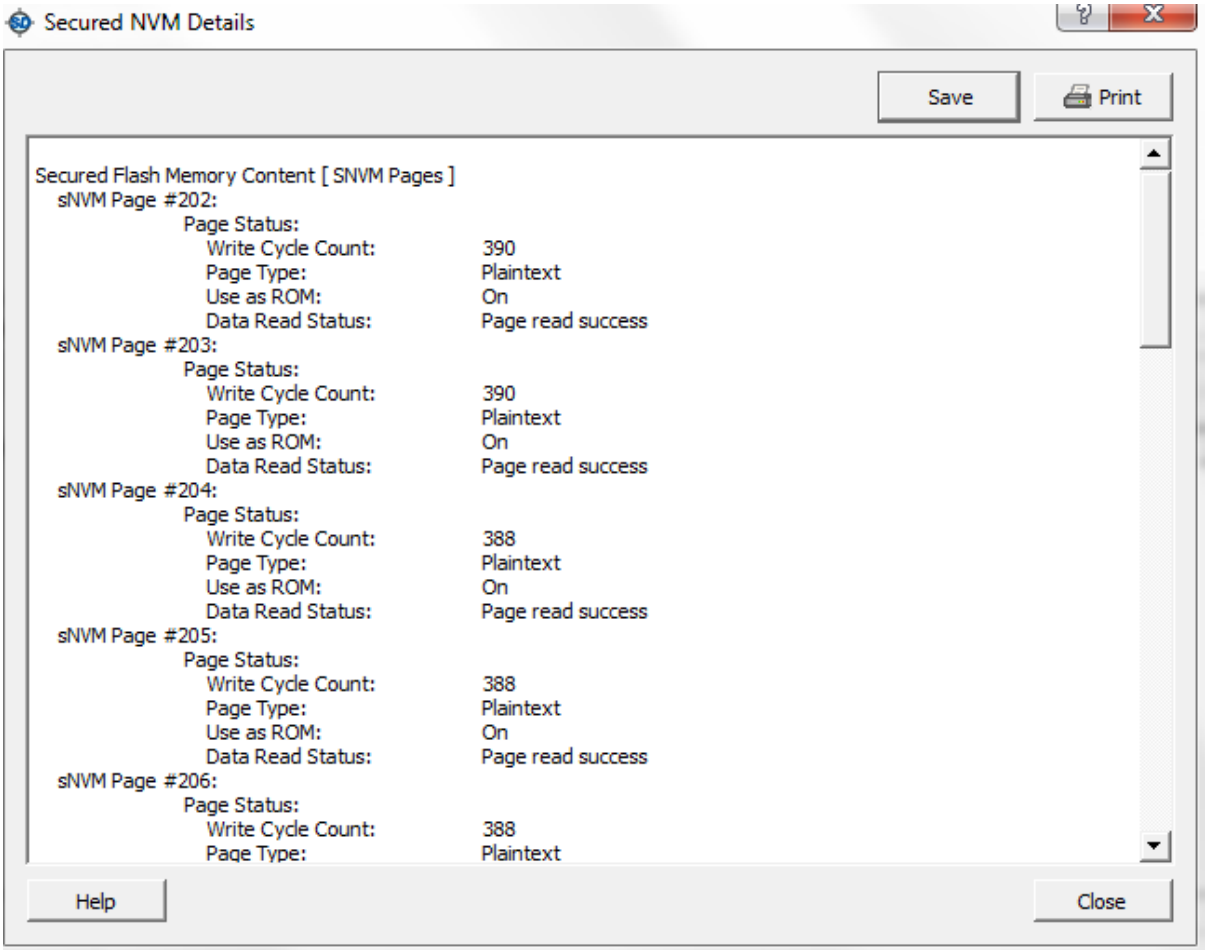
**Figure 1-35.** sNVM Debug-Client View




4. Click **View All Page Status** to view the page status such as Write Cycle Count, Page Type, Use as ROM, and Data Read Status, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

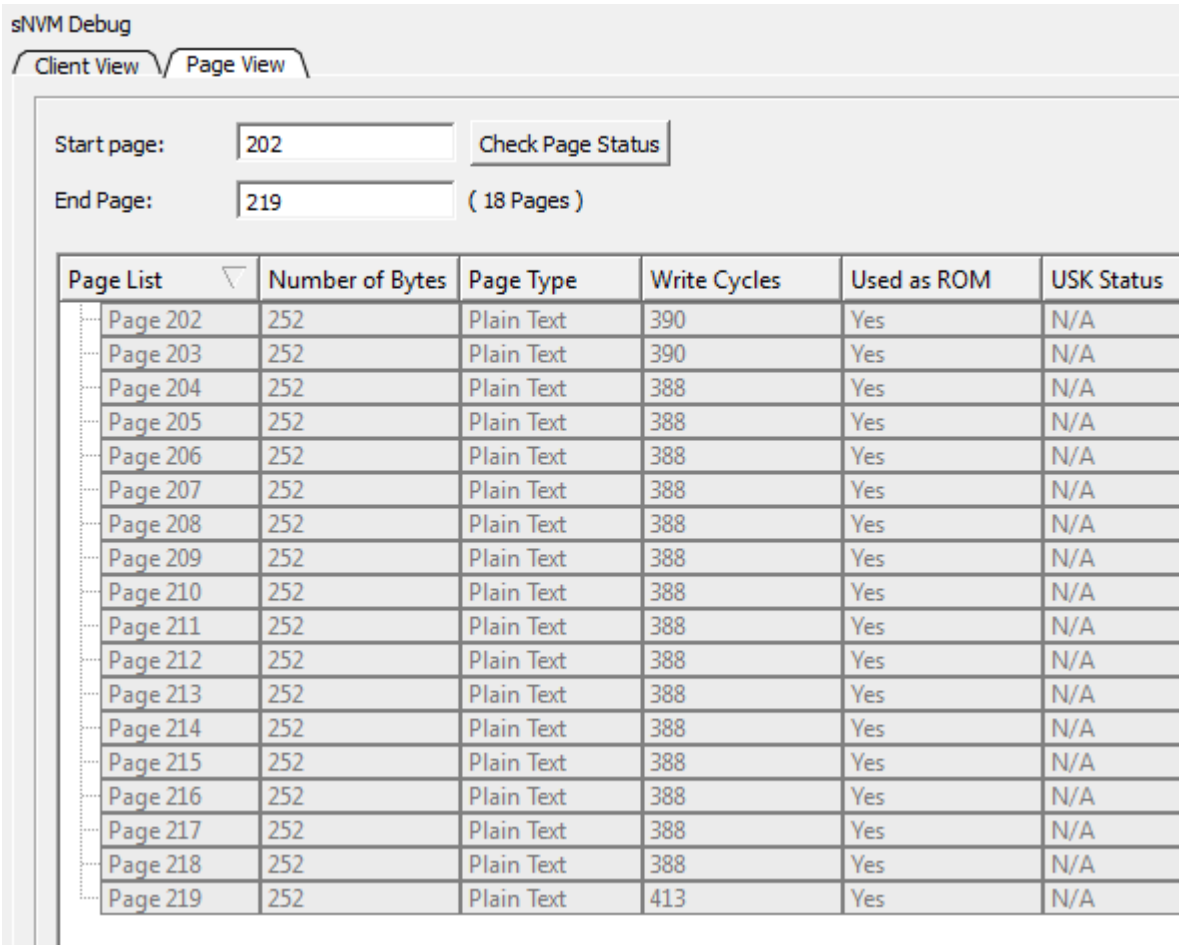
**Figure 1-36.** Secured NVM Details




5. Click **Page View** tab in the **sNVM Debug** window, Page view displays the client details of the required pages. You can read pages from 0-220 in the page view.
6. Enter the **Start page** and **End page** in the respective boxes.
7. Click **Check Page Status**. The page status information is displayed, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-37.** sNVM Debug—Page View




| Page List | Number of Bytes | Page Type | Write Cycles | Used as ROM | USK Status |
|---|---|---|---|---|---|
| Page 202 | 252 | Plain Text | 390 | Yes | N/A |
| Page 203 | 252 | Plain Text | 390 | Yes | N/A |
| Page 204 | 252 | Plain Text | 388 | Yes | N/A |
| Page 205 | 252 | Plain Text | 388 | Yes | N/A |
| Page 206 | 252 | Plain Text | 388 | Yes | N/A |
| Page 207 | 252 | Plain Text | 388 | Yes | N/A |
| Page 208 | 252 | Plain Text | 388 | Yes | N/A |
| Page 209 | 252 | Plain Text | 388 | Yes | N/A |
| Page 210 | 252 | Plain Text | 388 | Yes | N/A |
| Page 211 | 252 | Plain Text | 388 | Yes | N/A |
| Page 212 | 252 | Plain Text | 388 | Yes | N/A |
| Page 213 | 252 | Plain Text | 388 | Yes | N/A |
| Page 214 | 252 | Plain Text | 388 | Yes | N/A |
| Page 215 | 252 | Plain Text | 388 | Yes | N/A |
| Page 216 | 252 | Plain Text | 388 | Yes | N/A |
| Page 217 | 252 | Plain Text | 388 | Yes | N/A |
| Page 218 | 252 | Plain Text | 388 | Yes | N/A |
| Page 219 | 252 | Plain Text | 413 | Yes | N/A |

### 1.8.7. Debug TRANSCEIVER (Ask a Question)

SmartDebug enables transceiver debugging, which includes checking lane functionality and health for different settings of lane parameters. Select **Debug TRANSCEIVER** in the **SmartDebug** window to access the debug transceiver feature. Debug Transceiver supports the following features:

- Configuration Report
- SmartBERT
- Loopback Modes
- Static Pattern Transmit
- Eye Monitor

#### 1.8.7.1. Configuration Report (Ask a Question)

The Configuration Report feature creates a report that shows the physical location, Tx and Rx PLL lock status, and data width of all enabled transceiver lanes. This report includes the following lane parameters:

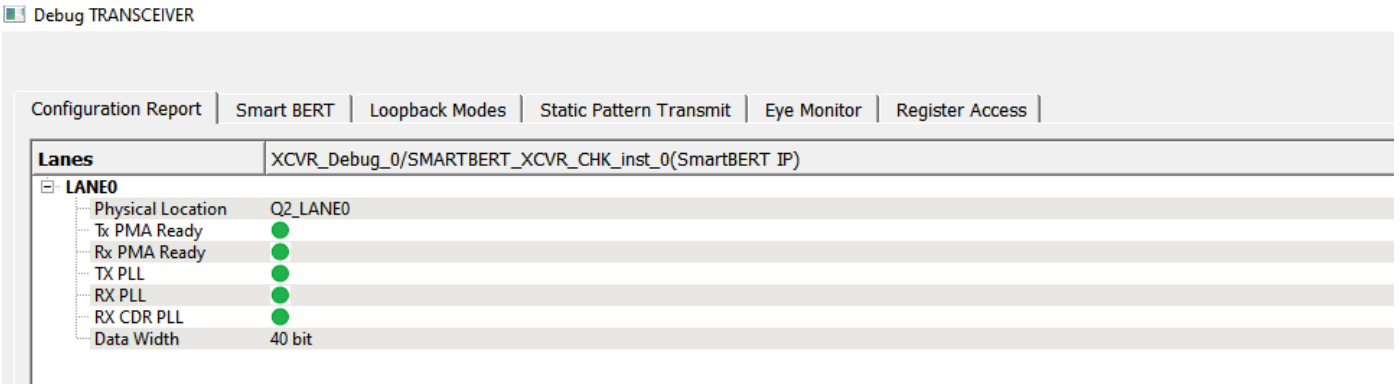
- **Physical Location**: Physical location of the transceiver lanes in the system.
- **Tx PMA Ready**: Tx lane of the transceiver is powered up and ready for transactions.
- **Rx PMA Ready**: Rx lane is powered up and ready for transactions.
- **TX PLL**: TX PLL of the transceiver is locked.
- **RX PLL**: RX PLL of the transceiver is locked.


© 2025 Microchip Technology Inc. and its subsidiaries

- **Data Width**: Configured data width of the corresponding lanes in the transceiver.

The following figure shows the **Configuration Report** tab.

**Figure 1-38.** Configuration Report




**Important:** The initial status of RX PLL and RX CDR PLL status is inactive. The status changes to active when the data is sent. SmartBERT is configured in CDR mode, so the data must be sent to get the PLL Locked. Go to the Smart BERT tab, select the XCVR instance and start the data transmission using the default PRBS pattern. Then, the status changes to active.

### 1.8.7.2. SmartBERT (Ask a Question)

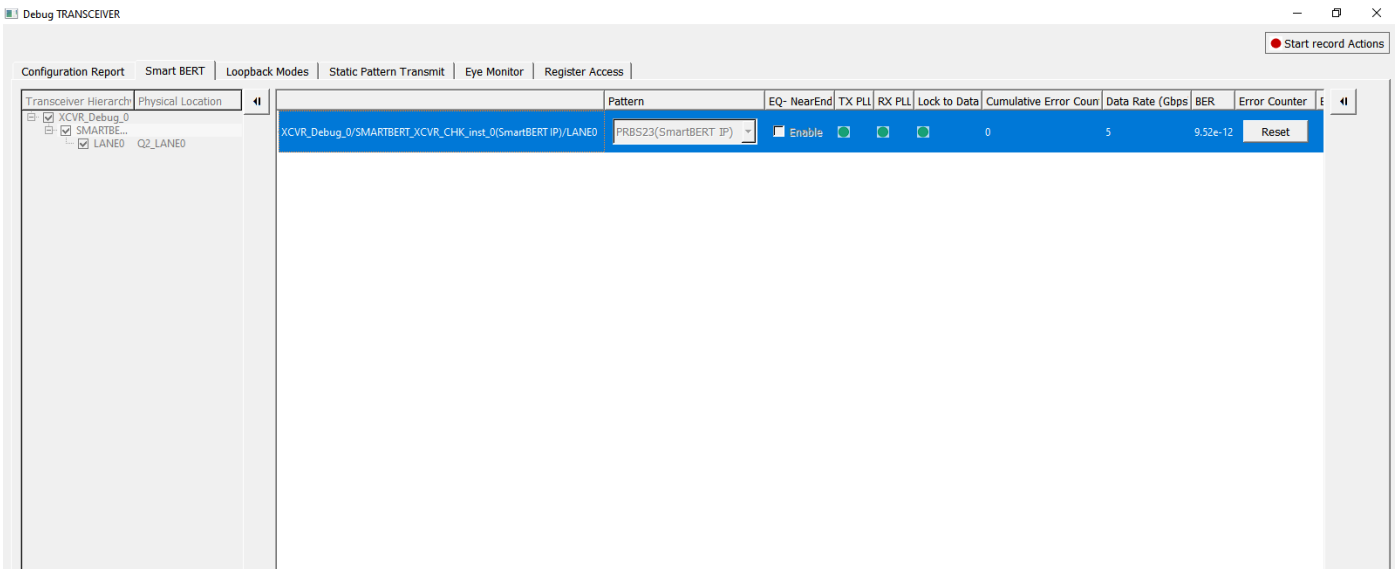
SmartBERT enables you to run diagnostic tests on the transceiver lanes. SmartBERT uses the PRBS generator and checker functionality available in each transceiver lane to determine a lane's bit error rate (BER). The various PRBS patterns supported are PRBS7(SmartBERT IP), PRBS9(SmartBERT IP), PRBS15(SmartBERT IP), PRBS23(SmartBERT IP), and PRBS31(SmartBERT IP). Near-end loopback can be performed using one of these PRBS patterns.

To run SmartBERT in Debug TRANSCEIVER, follow these steps:

1. Select the **SmartBERT** tab in the **Debug TRANSCEIVER** window.
2. Select **LANE** in the left pane.
3. Select the **Pattern** from the drop-down list.
4. Select the **EQ-NearEnd** check box to enable internal loop-back, (this step can be ignored if external loop-back is enabled).
5. Click **Start**. This enables both transmitter and the receiver for a particular lane and for a particular PRBS pattern. The following figure shows the Debug TRANSCEIVER window and the PRBS pattern options for SmartBERT.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-39.** Debug TRANSCEIVER—Smart BERT




When a SmartBERT IP lane is added, the **Error Injection** column is displayed in the right pane. The error injection feature is provided to inject an error while running a PRBS pattern. This feature is unavailable if regular lanes are added. Also, this feature is enabled only for SmartBERT IP supported PRBS patterns.

6. Select **Reset** to clear the error count under **Cumulative Error Counter**. Error Count is displayed when the lane is added.

Figure 1-39 shows the **Smart BERT** tab and status of the TXPLL, RXPLL, Lock to Data, Data rate, and the BER.

### 1.8.7.3. Loopback Modes (Ask a Question)

Loopback modes perform the following types of loopback tests:

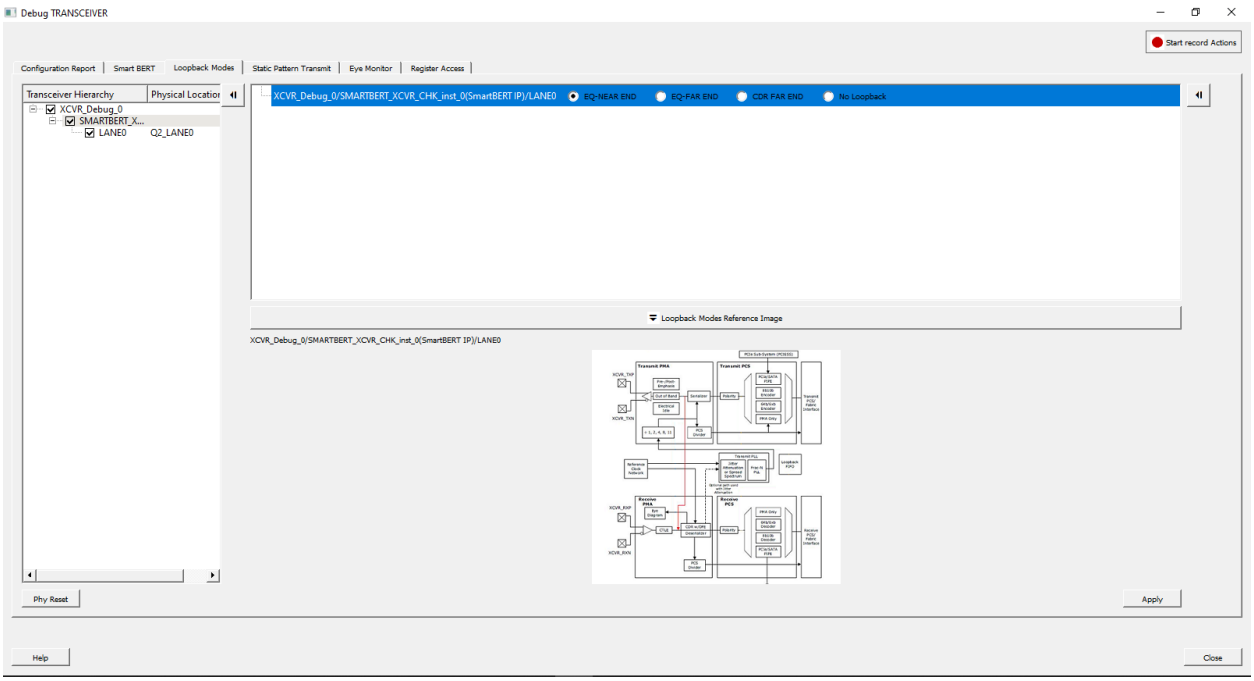
- **EQ-Near End Loopback**: Serialized data from PMA is looped from Tx to Rx internally before the transmit buffer. This is called near-end serial loopback. EQ-Near End loopback supports data transmission rates of up to 10.315 Gbps.
- **EQ-Far End Loopback**: Serialized data from Rx is looped back to Tx in PMA. This is called far-end serial loopback. EQ-Far End loopback supports data transmission rates of up to 1.25 Gbps.
- **CDR-Far End Loopback**: De-serialized data from PCS Rx channel is looped back to Tx.
- **No Loopback**: Data is not looped internally.

To select Loopback mode, perform the following steps:

1. Select **LANE** in the left pane.
2. Select **Loopback Mode** and click on **Apply** to apply the loopback.
3. Go to **SmartBERT** tab, select **LANE** and choose any prbs pattern and click **Start**.
4. Check the status of TX PLL, RX PLL, Lock to Data.
5. Click **Stop** to stop the pattern transmission for the selected lane.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-40.** Debug TRANSCEIVER—Loopback Modes




### 1.8.7.4. Static Pattern Transmit (Ask a Question)

Static Pattern Transmit enables the selection of pattern to be transmitted on a specific transceiver (Tx) lane. It supports the following pattern:

- Fixed pattern
- Max run length pattern
- User pattern

The user pattern is defined in the value column. It must be hex numbers and not greater than the configured data width.

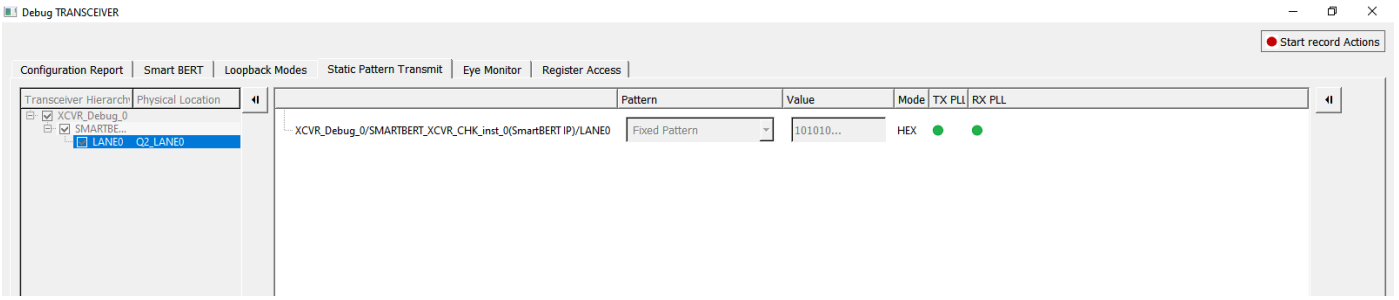
TX-PLL indicates lane-lock onto TX PLL when a static pattern is transmitted. RX-PLL indicates RX PLL lock when a static pattern is transmitted. Data Width displays the data width configured for a transceiver lane.

To view static pattern transmit, perform the following steps:

1. Select the **Static Pattern Transmit** tab.
2. Select the **Transceiver Hierarchy** in the left pane of the window. The selected lane data is displayed in the right pane. Select a pattern from the **Pattern** drop-down list.
3. Click **Start**. The static pattern for the selected lanes is transmitted.
4. The static pattern for the selected lanes is transmitted. Status of TX PLL and RX PLL should be green.
5. Click **Stop**. The static pattern transmission is stopped for the selected lanes.
   The following figure shows the **Static Pattern Transmit** tab.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-41.** Static Pattern Transmit




### 1.8.7.5. Eye Monitor (Ask a Question)

Eye Monitor enables visualizing the eye diagram present within the receiver. This feature plots the receiving eye after the CTLE and Receiver functions. The diagram representation provides vertical and horizontal measurements of the eye measurement and BER performance measurements. Whenever PRBS/static pattern transmission is in progress, click the **Eye Monitor** tab in the **Debug TRANSCEIVER** window to see the eye monitor representation within the receiver.

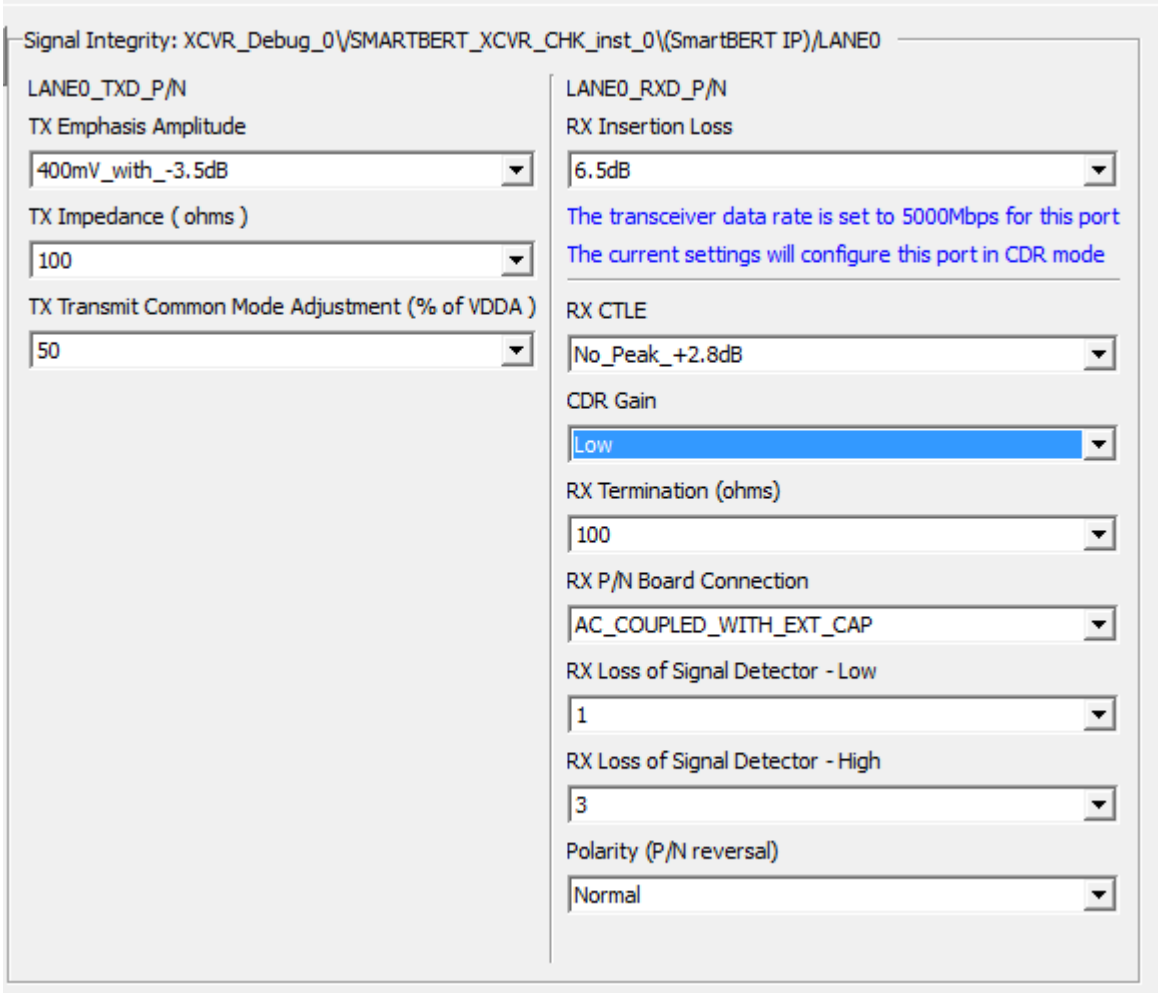
In Libero SoC, the following types of Eye Scan modes are supported:

- **Normal mode—**In this mode, Eye Monitor performs a single eye scan and displays the Eye Diagram on the Eye Monitor plot.
- **Infinite Persistent Mode—**In this mode, the Plot Eye button changes to Start Plot Eye. Select Start Plot Eye to start infinite persistent eye monitoring. The Start Plot Eye button changes to Stop Plot Eye, and the infinite scanning and accumulation process begins. In every iteration, the eye is cumulated with all previous eyes to make a single cumulative eye. This cumulative eye is displayed with a color scheme on the Eye Monitor plot. The completed iteration number and the cumulative BER are updated and displayed after every iteration, along with the cumulative eye. To stop cumulative eye monitoring, click the **Stop Plot Eye** button. The process halts after the current iteration completes.
- **Design Initiated Eye Plots—**In this mode, the Select Eye Output drop-down is enabled when an Eye Plot log file is browsed and loaded on the Eye Monitor page. Click **Browse File** to load the Eye Plot output files. If the loaded Design Initiated Eye Plot log file does not contain any eye output, it is disabled. After selecting Eye output from the Select Eye Output drop-down, click Plot Eye to start eye monitoring for the lane. Then the Eye diagram displays the selected log file.

The following figure shows the recommended SI settings for the demo design. These settings are for the short reach and less lossy cables.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-42.** Recommended Settings for Eye Monitor

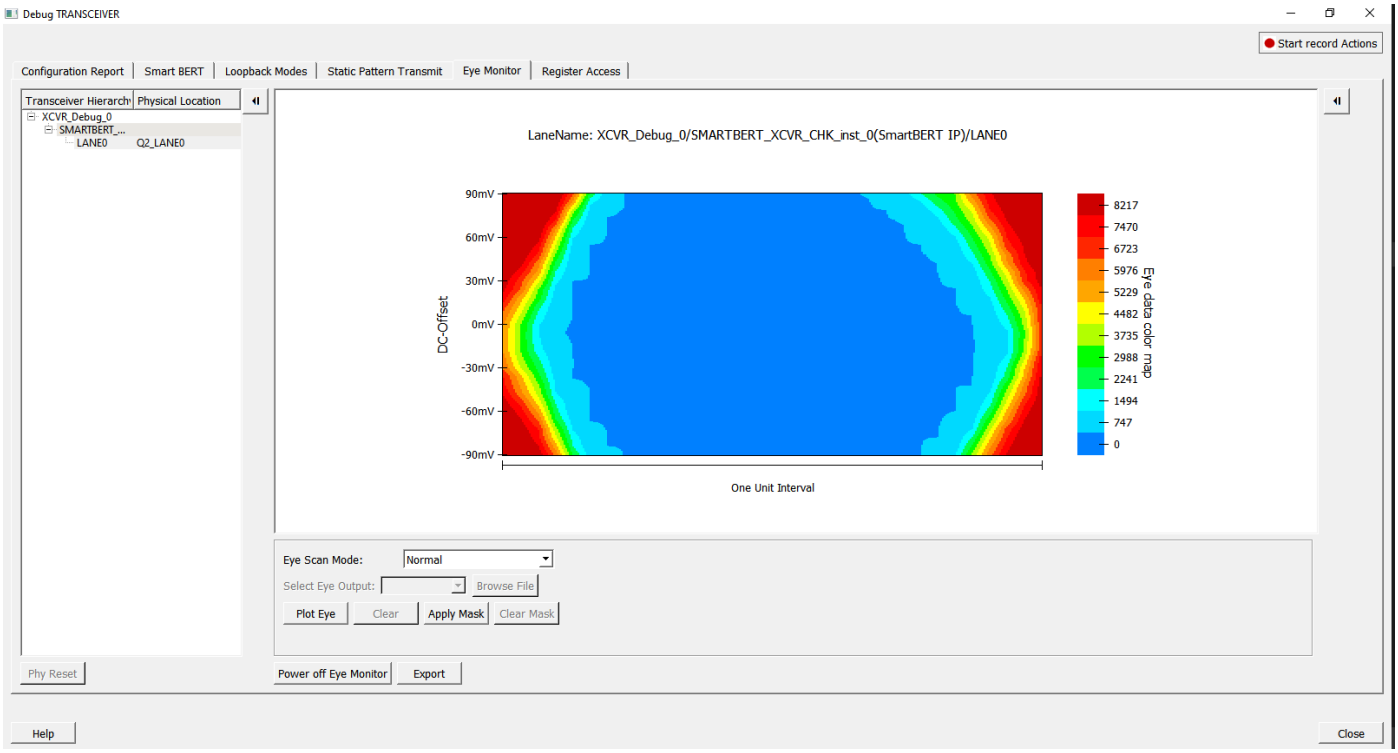



To plot the Eye Diagram, perform the following steps:

1. Go to **EYE Monitor** tab and Select **LANE0**.
2. Click **Power** on **Eye Monitor**.
3. Go to **SmartBERT** tab, select **LANE0** and choose any probes pattern and click **Start**.
4. Go back to **Eye Monitor** tab and click **Plot Eye** to plot the eye.
   The Eye Plot of the Signal in **LANE0** is plotted.
   The following figure shows the Eye Monitor tab.


© 2025 Microchip Technology Inc. and its subsidiaries

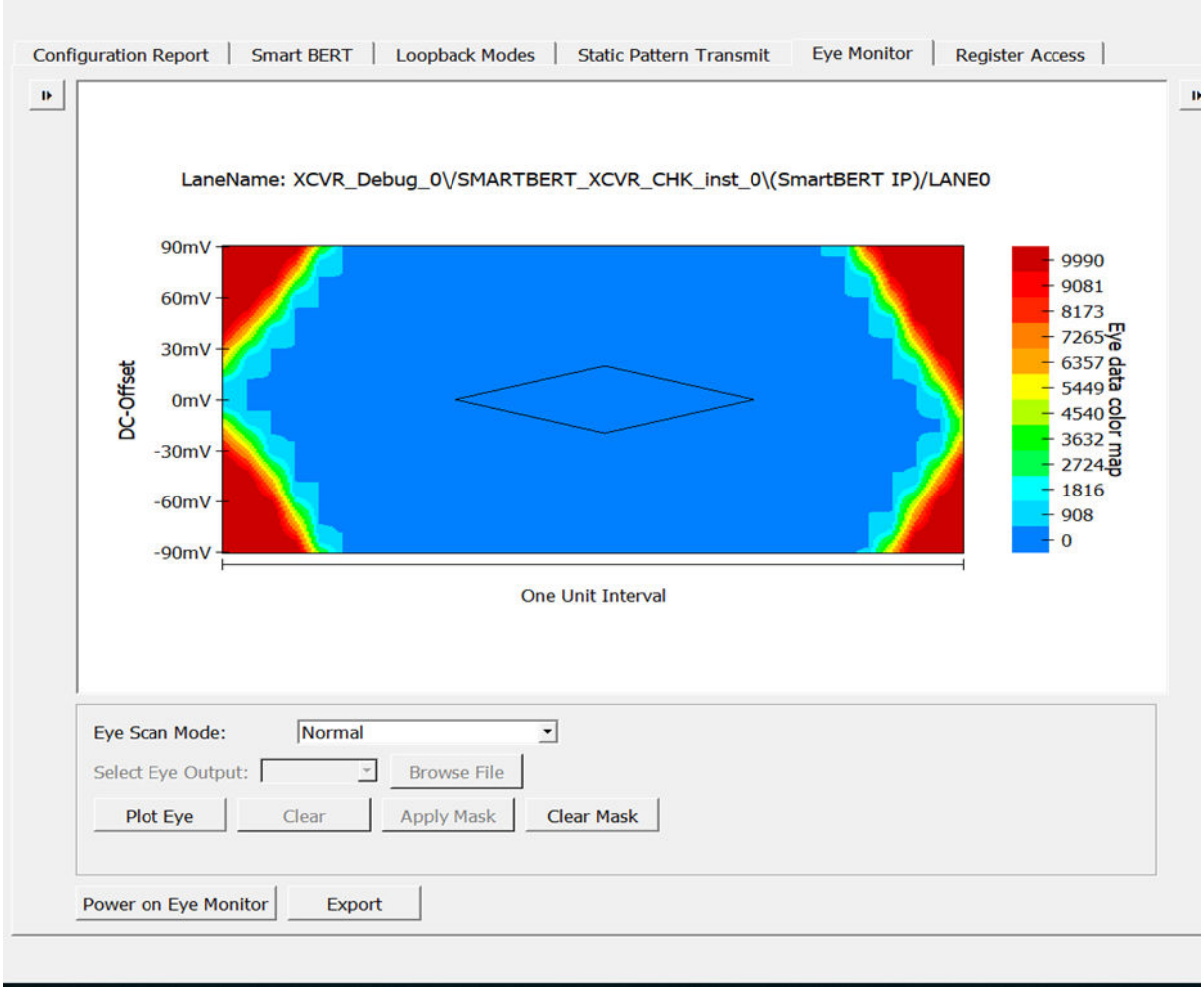
**Figure 1-43.** Debug TRANSCEIVER—Eye Monitor




5. After plotting the Eye diagram, the user can use the **Apply Mask** option to know the best eye-opening with the least errors. The **Apply Mask** and the **Clear Mask** options are disabled in the Default View. Click **Plot Eye** to enable the **Apply Mask** option.
6. After selecting the **Apply Mask** option, the Clear Mask option is enabled, and the Eye Mask for the Eye Plot appears, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-44.** Eye Monitor GUI After Applying the Mask Using Apply Mask Button



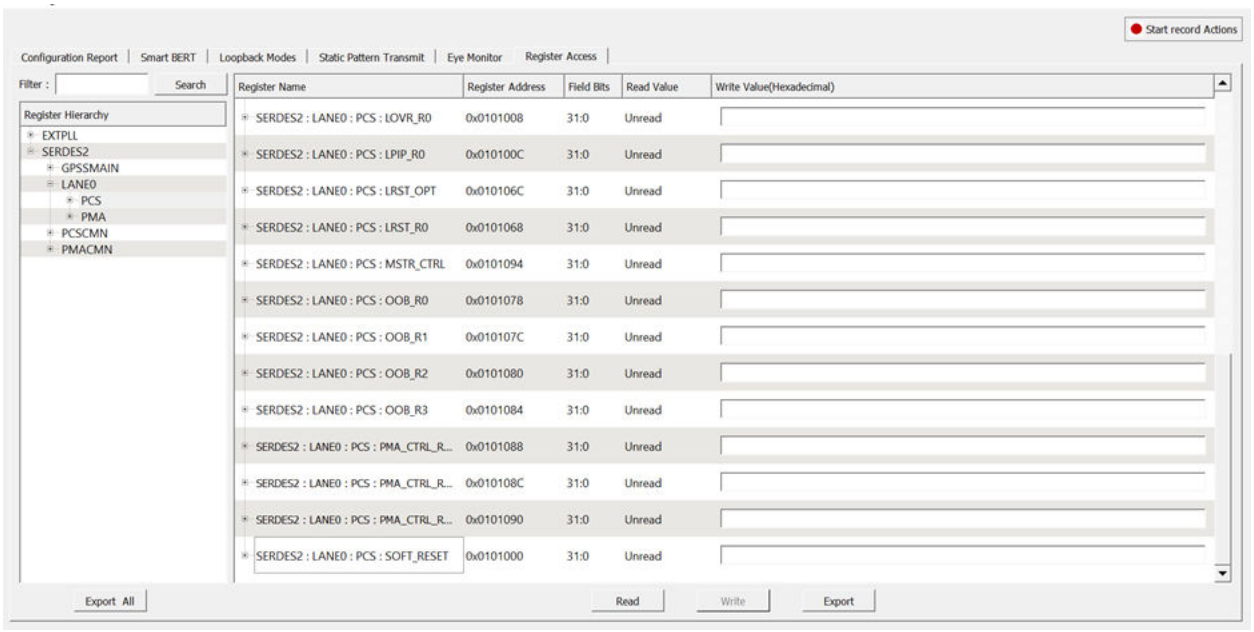

### 1.8.7.6. Register Access (Ask a Question)

The **Register Access** tab enables the following operations:

- Register read and write
- Export all register operations. The exported register details are saved to a `.csv` file.
- Register hide


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-45.** Register Access Tab



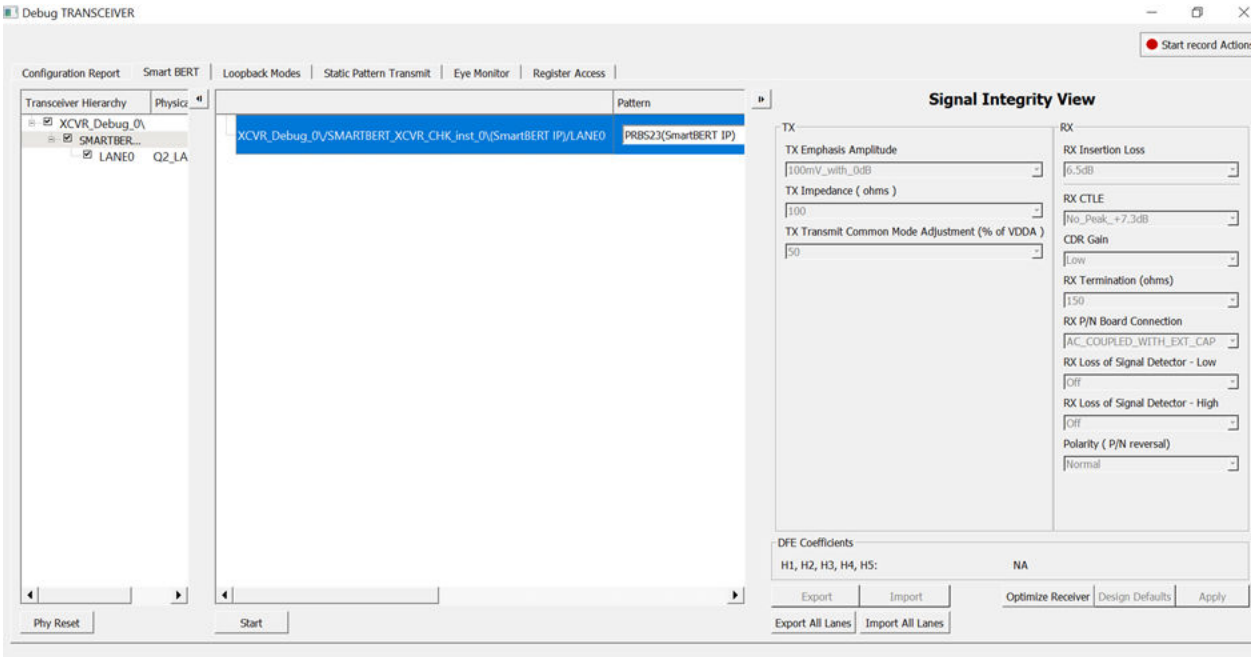

The preceding figure shows the register access tab.

For detailed information about Register Access tab, see SmartDebug User Guide.

### 1.8.7.7. Start Record Actions (Ask a Question)

The **Start Record Actions** option is used to record the register sequence of XCVR operations into a file. This option is available at the top-right on the Debug Transceiver window, as shown in the following figure.

**Figure 1-46.** Start Record Actions

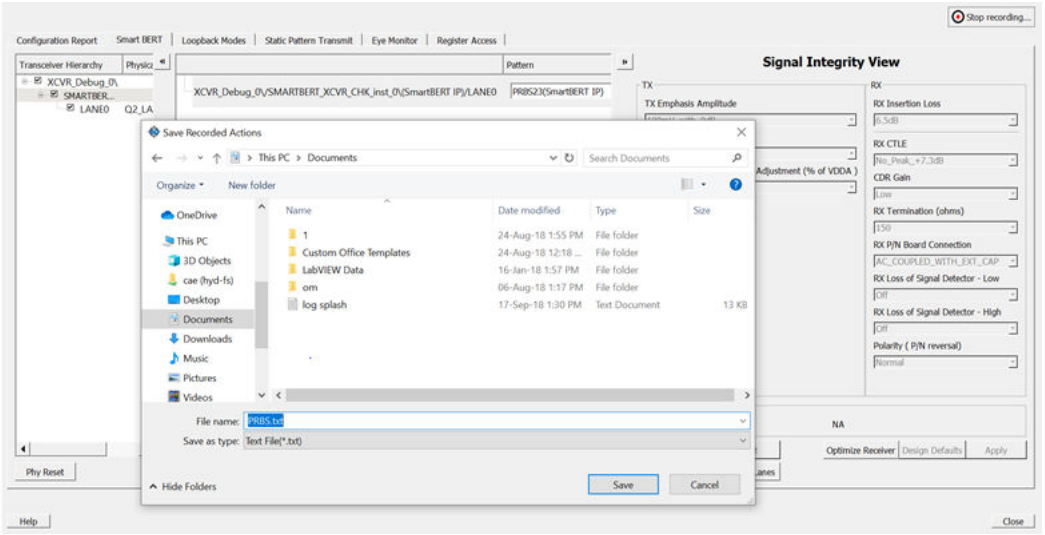




© 2025 Microchip Technology Inc. and its subsidiaries

This option is hidden in the Demo Mode. When this option is selected, the recording starts, and the option changes to **Stop recording** for stopping the recording. When **Stop recording** is selected, a window
pop-up prompts the user for the output to be saved to a `.txt` file as shown in. After saving the file, the **Debug TRANSCEIVER** window goes to default state.

**Figure 1-47.** Saving the Recording




For more information about this option, see SmartDebug User Guide.

### 1.8.7.8. Signal Integrity (Ask a Question)

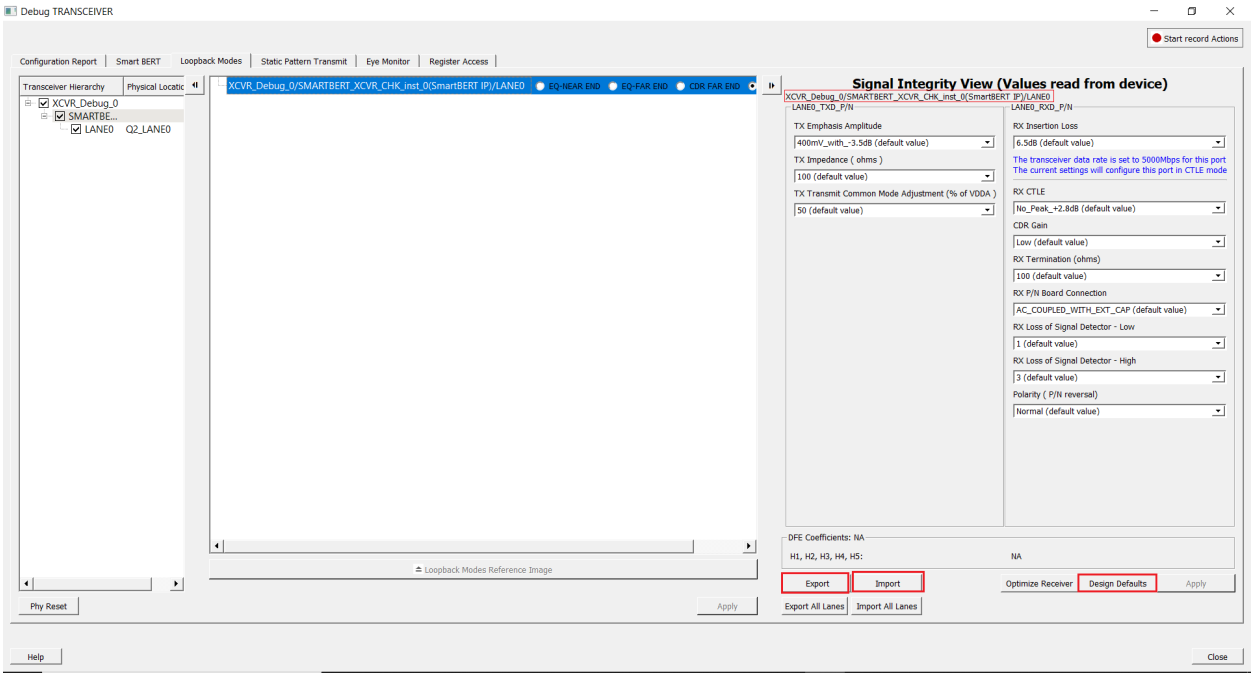
The Signal Integrity feature in SmartDebug works with Signal Integrity in the I/O Editor, allowing the import and export of PDC files. The Signal Integrity pane appears in the following SmartDebug pages:

- SmartBERT
- Loopback Modes
- Static Pattern Transmit
- Eye Monitor

When a lane is selected in the SmartBERT, Loopback Modes, Static Pattern Transmit, or Eye Monitor pages, the corresponding Signal Integrity parameters (configured in the I/O Editor or changed in SmartDebug) are enabled, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-48.** Signal Integrity




#### 1.8.7.8.1. Design Defaults (Ask a Question)

Click **Design Defaults** to load the signal integrity parameter options for the selected lane instance. These are the signal integrity settings selected in the Libero design flow and reside in the STAPL file.

#### 1.8.7.8.2. Export (Ask a Question)

Click **Export** to export the selected parameter options and other physical information to an external PDC file. A popup box prompts you to choose the location where you want the PDC file to be exported.

The exported content is in two set_io commands form—TXP and RXP ports of the selected lane instance.

### 1.8.7.9. Optimize Receiver (Ask a Question)

SmartDebug uses the Receiver coefficients to optimize the settings for the overall signal integrity at the receiver.

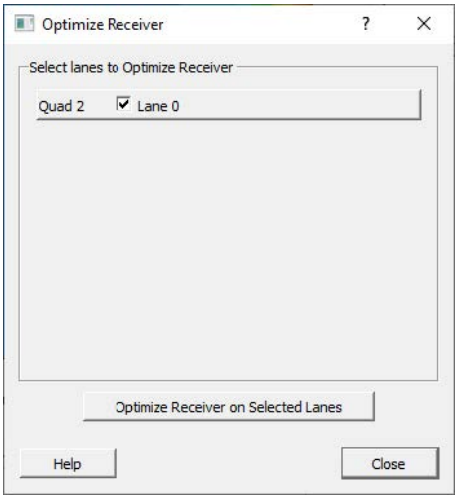
To run Optimize Receiver, perform the following steps:

1. In Debug Transceiver, go to **Signal Integrity** and click on **Optimize Receiver**.
2. In Optimize Receiver window the following settings
   - Select **Lanes** to Optimize Receiver: Lane0
3. Click **Optimize Receiver** on selected Lanes.

Software based Optimizing Receiver process is successful on all selected lanes.


© 2025 Microchip Technology Inc. and its subsidiaries

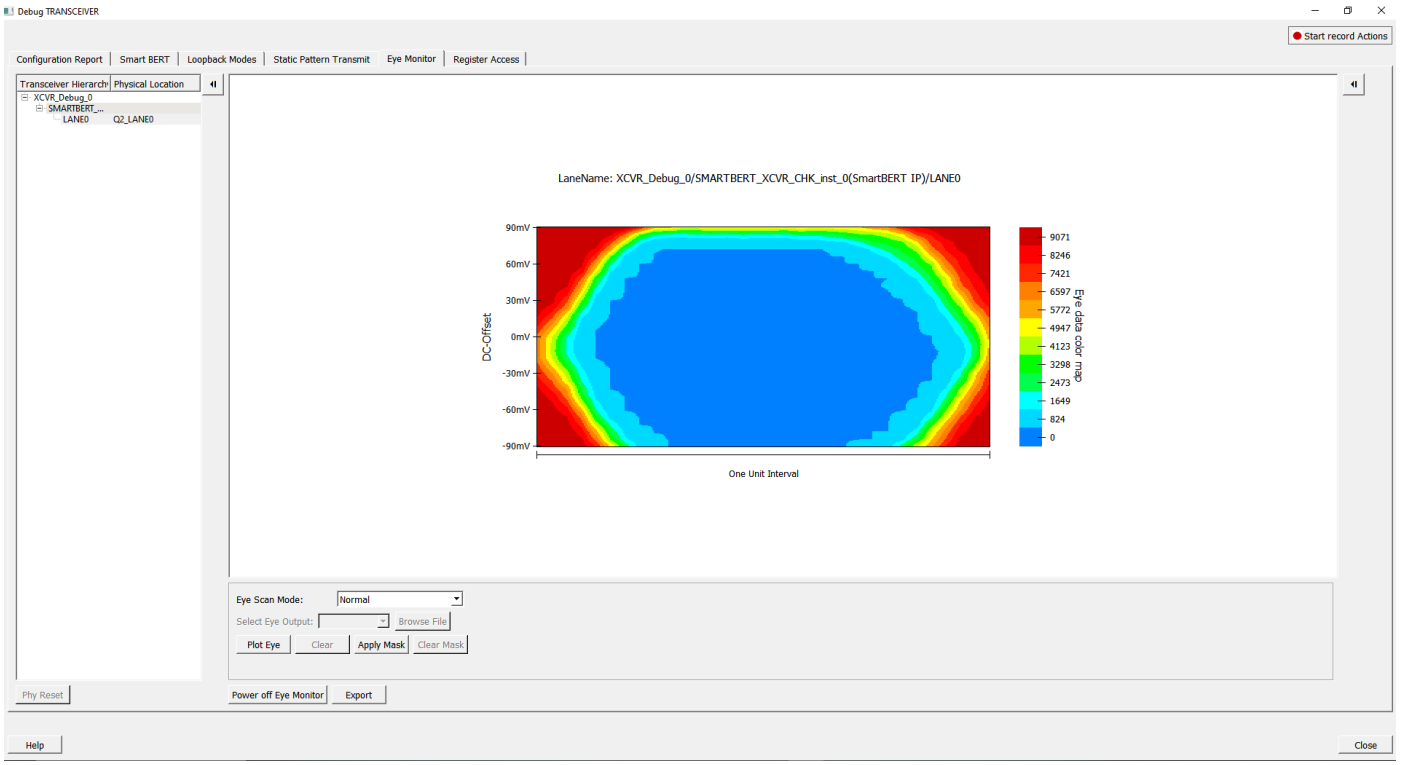
**Figure 1-49.** Optimize Receiver




### 1.8.7.9.1. Eye Monitor after Optimizing Receiver (Ask a Question)

After Optimizing Receiver, follow the steps mentioned in section Eye Monitor to plot the Eye Diagram.

**Figure 1-50.** Eye Diagram after using Optimize Receiver




## 1.8.8. Debug DDR IO Margin (Ask a Question)

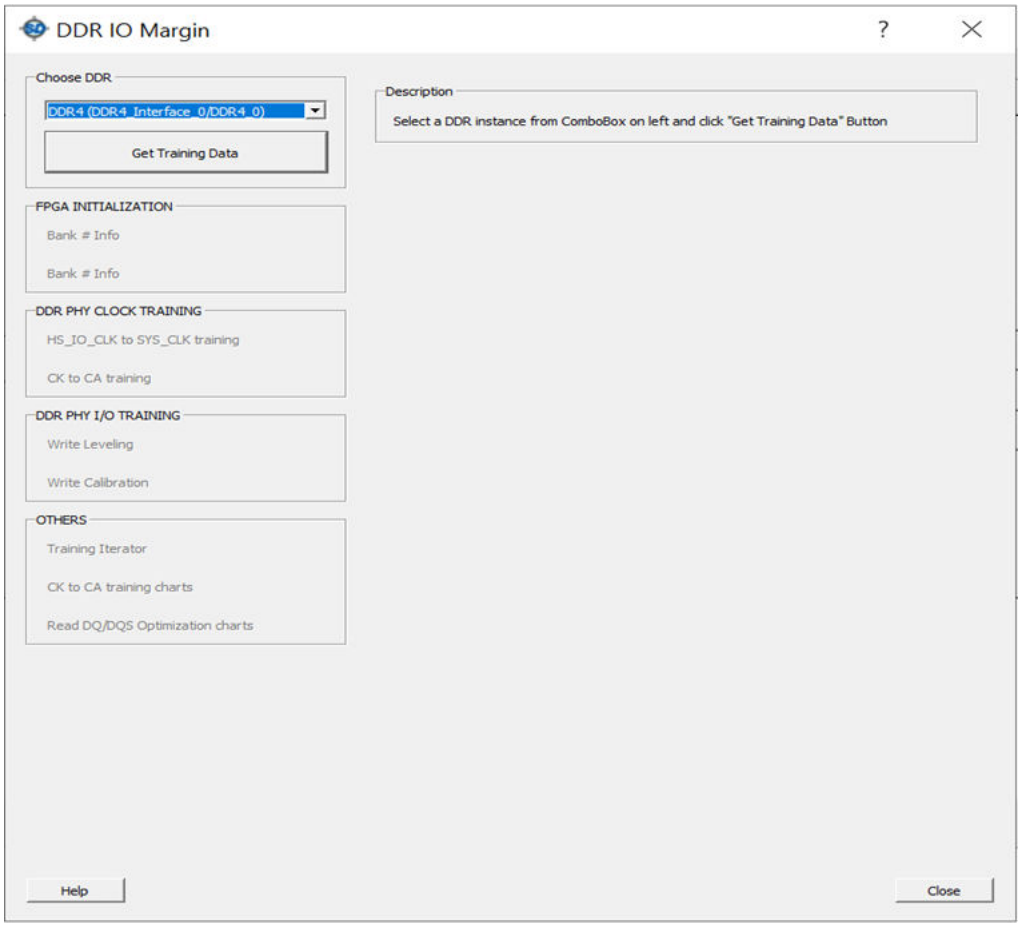
To access the Debug DDR IO Margin feature, select **Debug DDR Memory** from the main **SmartDebug** window. This option is available only for DDR3/DDR4/LPDDR3 memory configurations. This option is not visible when DDR memory is not used in the design. The default view of the **DDR IO Margin** window.


© 2025 Microchip Technology Inc. and its subsidiaries

**Important:** After programming the device, the LED11 glows, indicating the completion of DDR transactions.

**Important:** To debug DDR in the SmartDebug, run the tcl script without the FHB_ENABLE argument.

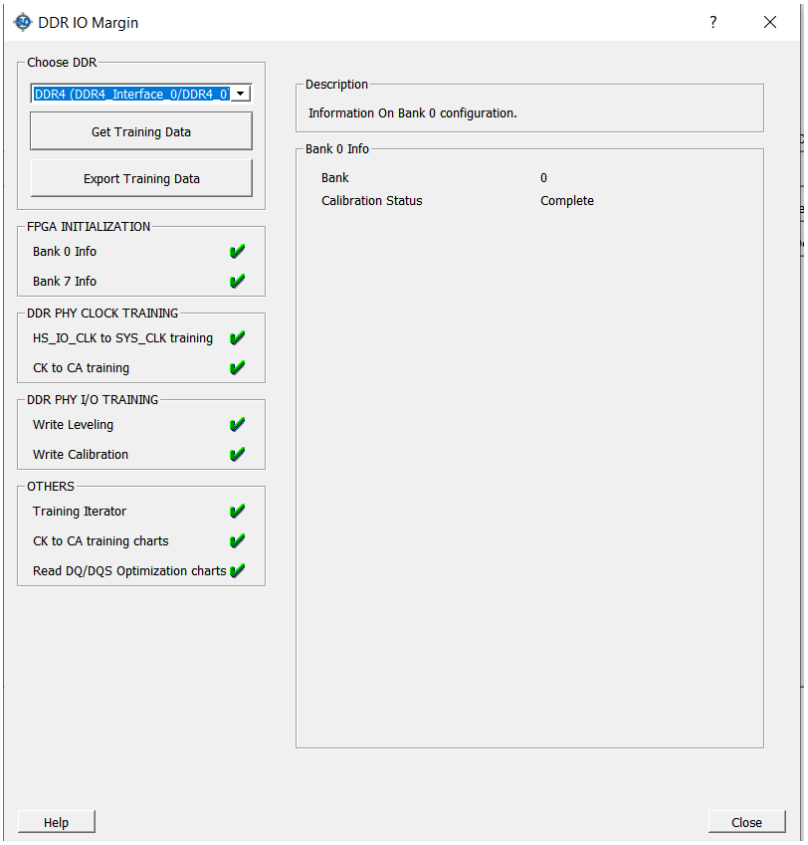
**Figure 1-51.** DDR IO Margin Window




Initially, all options in the DDR IO Margin GUI are disabled. Select the required DDR instances and click **Get Training Data**. After this, a script is run for fetching the training data. After around two minutes the fetched information of the selected DDR Instance is displayed as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 1-52.** Training Data




## 1.9. Conclusion (Ask a Question)

This application note demonstrated the capabilities of SmartDebug to observe and analyze many embedded device features. Live probes give real-time access to device test points, and internal logic states can be accessed using active probes. The SmartDebug TRANSCEIVER utility assists FPGA and board designers in validating signal integrity of high-speed serial links in a system and improves board bring-up time. This can be done in real-time without any design modifications. The PMA analog settings can be tuned to optimize link performance and to match the design to the system.


© 2025 Microchip Technology Inc. and its subsidiaries

# 2. Appendix 1: Known Issues (Ask a Question)

This chapter lists known issues related to SmartDebug hardware design debug and provides workarounds for each issues.

## 2.1. Data Traffic Errors on XCVR Lanes in CDR Mode (Ask a Question)

While plotting the eye using an eye monitor, errors are introduced in data traffic on transceiver lanes configured to use the CDR receiver path. The errors are introduced when Receiver and EM blocks are turned off during normal operation to save power. This issue does not impact functionality. The cumulative error count and BER values can be ignored when plotting the eye. A software update will be provided in future Libero releases to fix the issue.


© 2025 Microchip Technology Inc. and its subsidiaries

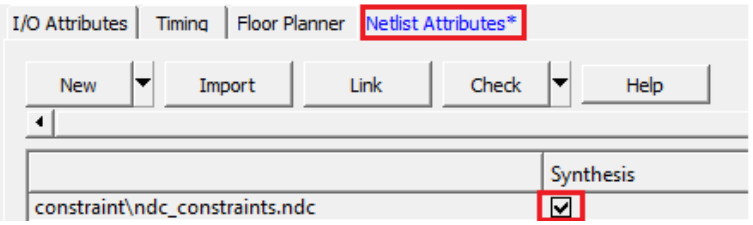
# 3. Appendix 2 : Synthesis (Ask a Question)

The synthesis process requires the following steps to be completed:

**Important:** Selecting the already imported `ndc_constraints.ndc` in case of FHB_ENABLED

To complete the synthesis process, on the **Netlist Attributes** tab, select the check box next to the `ndc_constraints.ndc` as shown in the following figure. The .ndc constraint file consists of instructions to automatically instantiate FHB at the specified instance path to manage timing.

**Figure 3-1.** Synthesis





© 2025 Microchip Technology Inc. and its subsidiaries

# 4. Appendix 3: Place and Route (Ask a Question)

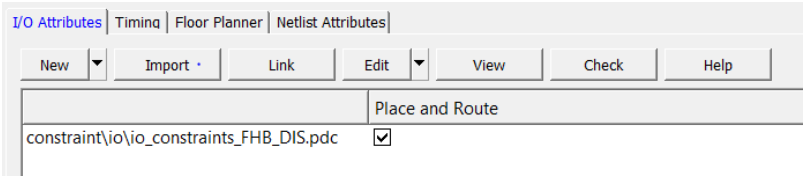
The place and route process requires the following steps to be completed:

1. Selecting the already imported `io_constraints_FHB_DIS.pdc or io_constraints_FHB_EN.pdc` file.
2. Placing the XCVR_Debug_0 block using the I/O Editor.
3. Ensuring all the I/Os are locked.

To complete the place and route process, follow these steps:

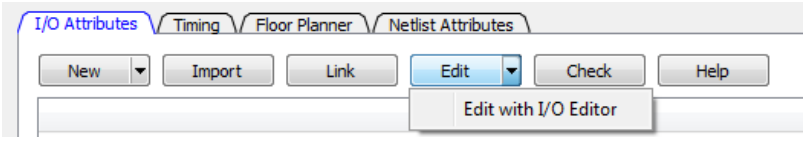
1. On the **I/O Attributes** tab, select the check box next to the `io_constraints_FHB_DIS.pdc or io_constraints_FHB_EN.pdc` file, as shown in the following figure. The `io_constraints_FHB_DIS.pdc or io_constraints_FHB_EN.pdc` file contains the I/O assignment for reference clock, switches and XCVR Lanes.

**Figure 4-1.** I/O Attributes Tab



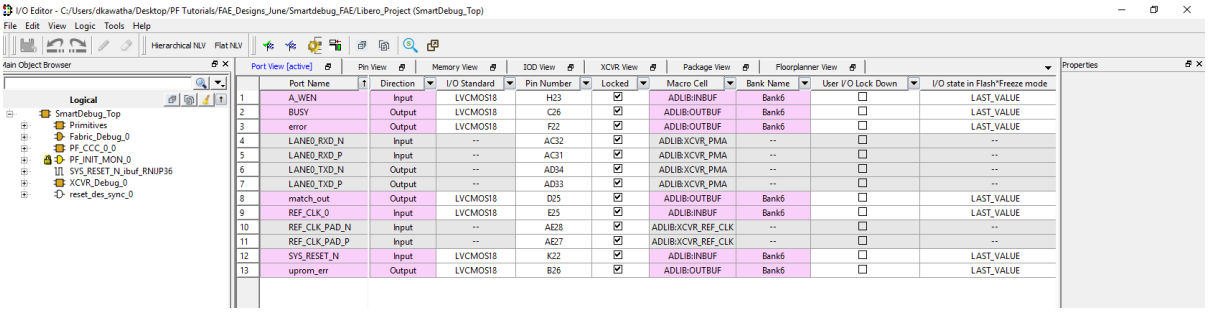

2. From the Edit drop-down list, select Edit with I/O Editor, as shown in the following figure.

**Figure 4-2.** Editing with I/O Editor




3. The I/O Editor opens, as shown in the following figure.

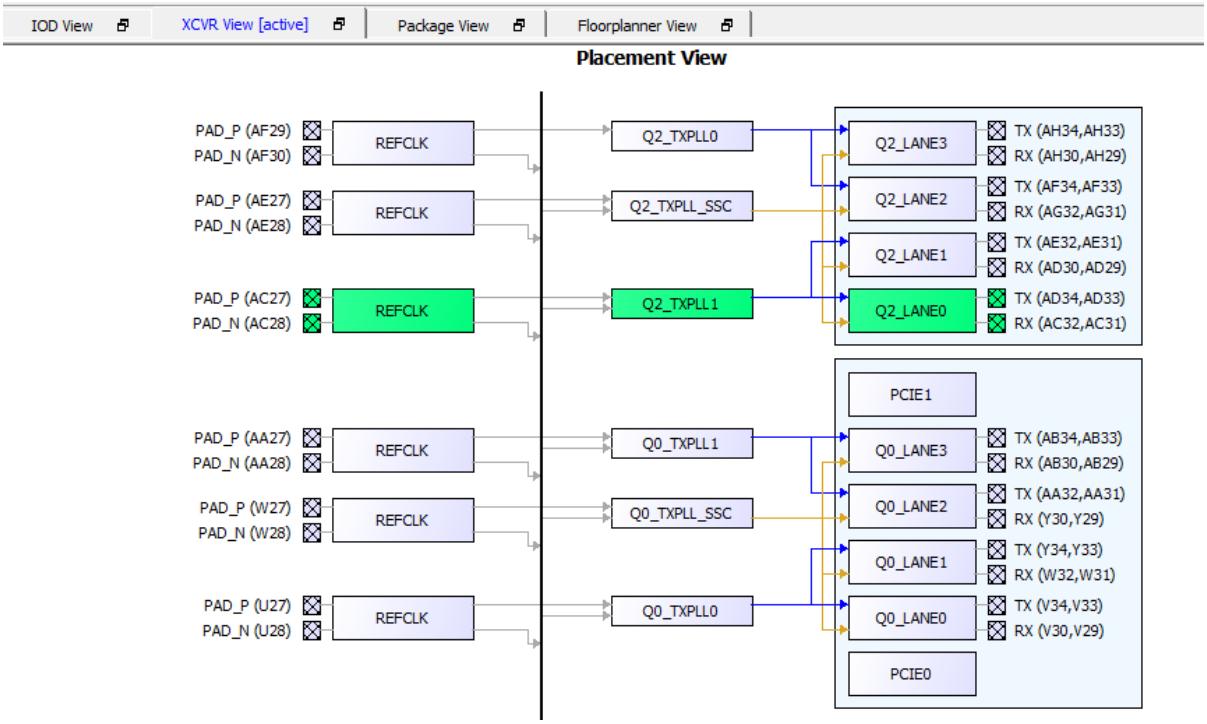
**Figure 4-3.** I/O Editor




4. Select the **XCVR View** in the **I/O Attribute editor**. This view allows you to assign IO locations to XCVR and reference clock.
5. Place TX_PLL, XCVR_REF_CLK, and XCVR_Debug, as shown in Figure 4-4 for Evaluation kit.

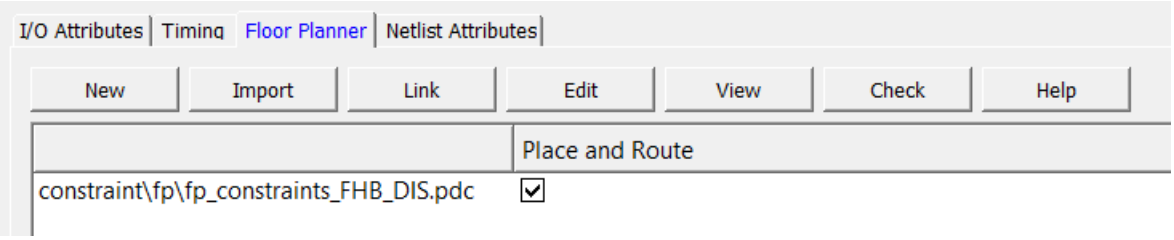

© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 4-4.** Placing the TX_PLL, XCVR_REF_CLK, and PF_XCVR Components (Evaluation kit)




6. Select **File > Commit** to save the placement, then close the I/O Editor (**File > Exit**).
7. Select the **Constraint Manager Floor Planner**. The constraint file (`fp_constraints_FHB_DIS.pdc or fp_constraints_FHB_EN.pdc`) must be visible. Ensure that the file is checked to be used for Place and Route.

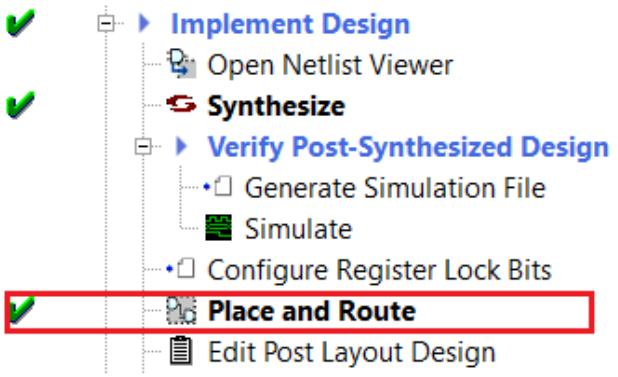
**Figure 4-5.** Constraint Files on the Floor Planner Attributes Tab




8. Double-click **Place and Route** from the **Design Flow** tab. When place and route is successful, a green tick mark appears next to Place and Route, as shown in the following figure.


© 2025 Microchip Technology Inc. and its subsidiaries

**Figure 4-6.** Place and Route





© 2025 Microchip Technology Inc. and its subsidiaries

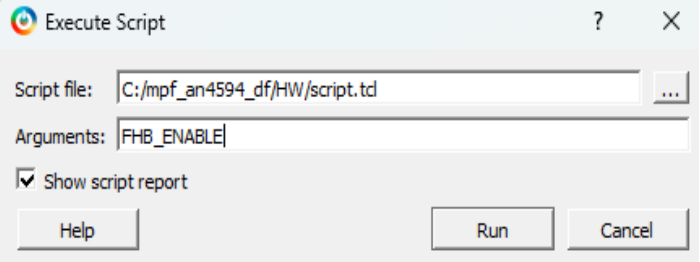
# 5. Appendix 4: Running the TCL Script (Ask a Question)

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

To run the TCL script, perform the following steps:

1. Launch the Libero® SoC.
2. Select **Project** > **Execute Script...**
3. Click **Browse** and select `script.tcl` from the downloaded `HW` directory.
4. In order to enable FHB provide argument `FHB_ENABLE` in the **Arguments** tab.

**Figure 5-1.** Enabling FHB




5. Click **Run**.

After successful execution of the TCL script, Libero project is created within `HW` directory. For more information about TCL scripts, see `mpf_an4594_df/HW/TCL_Script_readme.txt`.

For more information about TCL commands, see Tcl Commands Reference Guide. For any queries encounter while running the TCL script, contact Technical Support.


© 2025 Microchip Technology Inc. and its subsidiaries

# 6. Appendix 5: References (Ask a Question)

The following documents provide more information about the SmartDebug and IP cores used in the reference design:

- For more information about SmartDebug, see SmartDebug User Guide.
- For more information about PolarFire transceiver blocks, see PolarFire Family Transceiver User Guide.
- Fore more information about PF_CCC, see PolarFire Family Clocking Resources User Guide.
- For more information about Libero SoC, see the Libero SoC Documentation web page.
- For more information about PolarFire FPGA Evaluation Kit, see UG0747: PolarFire FPGA Evaluation Kit User Guide.
- For more information about PF_UPROM, PF_USRAM, and PF_DPSRAM, see Libero catalog.
- For more information about Identify RTL, see Synopsys Identify RTL User Guide.


© 2025 Microchip Technology Inc. and its subsidiaries

# 7. Revision History (Ask a Question)

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

**Table 7-1.** Revision History

| Revision | Date | Description |
|---|---|---|
| C | 07/2025 | The following is the list of changes in revision B of the document:<br>• Updated the document for Libero® v2025.1.<br>• Updated the `.job` filepath and TCL script filepath throughout the document.<br>• Updated Figure 1-7 in the Simulation Using Micron DDR4 SDRAM Model section.<br>• Updated Figure 1-11 in the Enabling FPGA Hardware Breakpoint (FHB) section. |
| B | 08/2024 | The following is the list of changes in revision B of the document:<br>• Updated the document for Libero SoC v2024p1<br>• Updated the Figure 1-9 in the section Clocking Structure<br>• Updated the Figure 1-10 in the section Reset Structure<br>• Updated the Figure 1-13 in the section Programming the Device<br>• Updated the Figure 1-15 in the section Launch SmartDebug from Libero<br>• Updated the Figure 1-16 in the section View Device Status<br>• Updated the Figure 1-18 in the section Active Probes<br>• Updated the Figure 1-21 and Figure 1-22 in the section Memory Blocks<br>• Updated the Figure 1-24 in the section Use Event Counter<br>• Updated the Figure 1-25 in the section Use Frequency Monitor<br>• Updated the Figure 1-26 in the section Use User Clock Frequency<br>• Updated the Figure 1-27 and Figure 1-28 in the section Select FHB<br>• Updated the Figure 1-35 in the section sNVM Debug<br>• Updated the Figure 1-38 in the section Configuration Report<br>• Updated the Figure 1-39 in the section SmartBERT<br>• Updated the Figure 1-41 in the section Static Pattern Transmit<br>• Updated the Figure 1-43 in the section Eye Monitor<br>• Updated the Figure 1-50 in the section Eye Monitor after Optimizing Receiver<br>• Updated the Figure 4-6 in the section Appendix 3: Place and Route<br>• Updated the Figure 5-1 in the section Appendix 4: Running the TCL Script<br>• Added the Appendix 2 : Synthesis section |
| A | 08/2022 | The following is the list of changes in revision A of the document:<br>• The document was migrated to Microchip template.<br>• The document number was updated to DS00004594A from 51900479.<br>• The document ID was updated to AN4594 from AC479.<br>• Updated note in Memory Blocks.<br>• Updated sNVM Debug steps in sNVM Debug.<br>• Updated Loopback mode steps in Loopback Modes.<br>• Updated Signal Integrity.<br>• Updated place and route steps in Appendix 3: Place and Route. |


© 2025 Microchip Technology Inc. and its subsidiaries

**Table 7-1.** Revision History (continued)

| Revision | Date | Description |
|---|---|---|
| 8.0 | _ | The following is a summary of the changes made in this revision.<br>• Updated for Libero SoC v2021.2.<br>• Updated Design Requirements.<br>• Updated DDR Interface in Demo Design.<br>• Updated Clocking Structure and Reset Structure.<br>• Updated Enabling FPGA Hardware Breakpoint (FHB).<br>• Updated Programming the Device.<br>• Added information about applying eye mask on plotted eye diagram, see Eye Monitor.<br>• Added Register Access and Start Record Actions.<br>• Added Debug DDR IO Margin. |
| 7.0 | _ | Added Appendix 4: Running the TCL Script. |
| 6.0 | _ | The following is a summary of the changes made in this revision.<br>• Updated the document for Libero® SoC v12.2.<br>• Removed the references to Libero version numbers. |
| 5.0 | _ | The following is a summary of the changes made in this revision.<br>• Updated the document for Libero® SoC v12.0.<br>• The new FHB feature of SmartDebug was added, see Using FHB.<br>• Updated the supported Eye Scan Modes in Eye Monitor. |
| 4.0 | _ | The following is a summary of the changes made in this revision.<br>• Converted this document from a tutorial (TU0804) to an application note (AC479).<br>• Updated the document for both Evaluation and SPLASH kits.<br>• Included the information from UG0743: PolarFire FPGA Debug User Guide.<br>• Updated the document for Libero® SoC PolarFire v2.3. |
| 3.0 | _ | The document was updated for Libero SoC PolarFire v2.2. |
| 2.0 | _ | The document was updated for Libero SoC PolarFire v2.1. |
| 1.0 | _ | The first publication of this document. |


© 2025 Microchip Technology Inc. and its subsidiaries

# Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

# Microchip Information

## Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-1684-6

## Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.


© 2025 Microchip Technology Inc. and its subsidiaries

## Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.


© 2025 Microchip Technology Inc. and its subsidiaries