PolarFire FPGA Implementing Data Security Using User Cryptoprocessor Application Note

AN4591

Introduction (Ask a Question)

PolarFire® FPGAs represent the industry's most advanced security programmable FPGAs. Data security protects application data—stored, communicated, or computed at run-time—from being copied, altered, or corrupted. PolarFire devices have a dedicated crypto processor, called User Cryptoprocessor, for data security applications.

This application note describes User Cryptoprocessor features and how to build a Libero® and SoftConsole project to perform cryptographic operations using the User Cryptoprocessor as a coprocessor to a host general purpose processor.

Table of Contents

1. User Cryptoprocessor (Ask a Question)

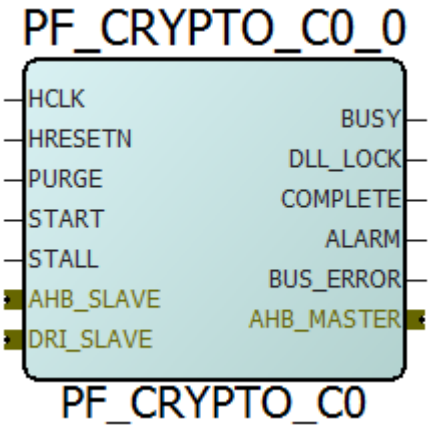
The User Cryptoprocessor is an Athena TeraFire® EXP-F5200B cryptography microprocessor. It provides complete support for the Commercial National Security Algorithm (CNSA) Suite and beyond and includes Side-Channel Analysis (SCA) resistant cryptography using patented leakage reduction countermeasures. These countermeasures provide strong resistance against SCA attacks such as Differential Power Analysis (DPA) and Simple Power Analysis (SPA). The User Cryptoprocessor is available in PolarFire® "S" grade devices.

The User Cryptoprocessor supports numerous cryptographic algorithms, including the following:

- AES with 128-bit, 192-bit, and 256-bit key sizes in ECB, CBC, CFB, OFB, CTR, and GCM modes
- AES key wrap and unwrap
- SHA1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
- AES-CMAC and AES-GMAC
- HMAC-SHA
- True random number generation (non-deterministic random bit generator plus NIST SP800-90A deterministic random bit generator)
- RSA, DSA, and modular exponentiation (Diffie-Hellman) with key sizes up to 3072-bits
- EC key pair generation, point validation, point multiplication (EC Diffie-Hellman), and ECDSA for
  - NIST P-curves: P-192, P-224, P-256, P-384, and P-521
  - Brainpool curves: P-256, P-384, and P-512
- Key-tree function

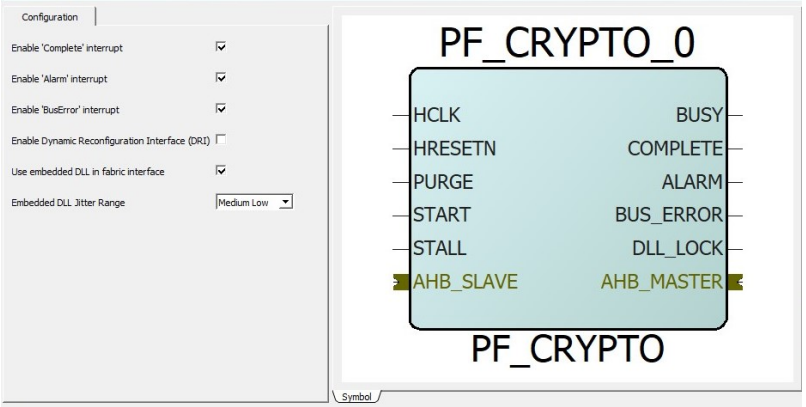
The User Cryptoprocessor is a hard block in PolarFire FPGAs, and its maximum operating frequency is 189 MHz. It is accessible to a soft processor in the FPGA fabric through the AHB-Lite slave interface for control and primary data input and output. The User Cryptoprocessor has built-in DMA to offload the main processor from data transfers between the User Cryptoprocessor and the user memory. The DMA functionality is accessible through an AMBA AHB-Lite master interface. The Libero® SoC design suite provides a PF_CRYPTO macro in Catalog, which must be integrated with the user design to use the User Cryptoprocessor. The following figure shows the input and output ports of the PF_CRYPTO macro.

Figure 1-1. PolarFire CRYPTO Macro



The following figure shows the crypto configurator used to enable Complete interrupt, Alarm interrupt, BusError interrupt, and DRI. If the frequency of the crypto block is greater than or equal to 125 MHz, select the Use embedded DLL in the fabric interface option for removing clock insertion delay. The Embedded DLL Jitter Range can be set to Low, Medium_Low, Medium_High, or High according to the crypto clock jitter specification. For more information, see PolarFire FPGA Datasheet for embedded DLL jitter tolerance ranges.

Figure 1-2. Crypto Configurator



The following table lists and describes the PF_CRYPTO ports. The control and status signals initiate action and obtain status. Corresponding control and status signals are also accessible through the dynamic reconfiguration interface (DRI). Contact Microchip technical support for information on how to use DRI.

Table 1-1. PF_CRYPTO Port List

Port Name	Direction	Description
AHB_SLAVE	—	AHB-Lite slave interface.
AHB_MASTER	—	AHB-Lite master interface.
DRI_SLAVE	—	Control and status signals are accessible through the DRI.
HCLK	Input	AHB bus clock.
HRESETN	Input	AHB bus reset. Asserts the functional reset of the User Cryptoprocessor block and zeroizes all the internal RAM and registers as PURGE signal.
PURGE	Input	When the signal is set to '1', it initializes the Zeroization of User Cryptoprocessor internal RAM and registers. For normal operation, this signal must be tied low. The PURGE input is level sensitive, and if the PURGE pin is still asserted when a purge operation completes, another purge operation is initiated.
START	Input	External execution initiation input when the User Cryptoprocessor operates in the stand-alone configuration without a host processor connected to the bus interface. Asserting the START signal causes the User Cryptoprocessor to initiate execution. During execution, the status of the User Cryptoprocessor is reflected by the BUSY and DLL_LOCK ports. This signal must be tied low when the User Cryptoprocessor is used as a coprocessor.
STALL	Input	Stalls the User Cryptoprocessor for a clock cycle to introduce variance in the external signatures. The STALL input is expected to be generated by an LFSR circuit in the fabric and asserted randomly for a single cycle to achieve the required stall rates. The STALL input must not be asserted until at least three clock cycles after the HRESETN is de-asserted and the DLL has indicated LOCK for three cycles.
BUSY	Output	Execution status signal.
COMPLETE	Output	Asserted to indicate that the User Cryptoprocessor has completed an operation. This signal can be connected to the host microprocessor as an interrupt request signal, enabling the User Cryptoprocessor to interrupt the processor when it completes an operation.

Table 1-1. PF_CRYPTO Port List (continued)

Port Name	Direction	Description
ALARM	Output	Asserted to indicate an uncorrectable memory error condition. An uncorrectable memory error causes the crypto core to perform a reset and purge. This reset terminates any in-progress operation. For most CAL operations, the CALPKTrfRes() function is used to complete the operation and generates a hardware Fault code in the event of an alarm.
BUS_ERROR	Output	Asserted when a HRESP response error is detected by the User Cryptoprocessor AHB master. Once set, a reset is required to clear.
DLL_LOCK	Output	DLL lock status.

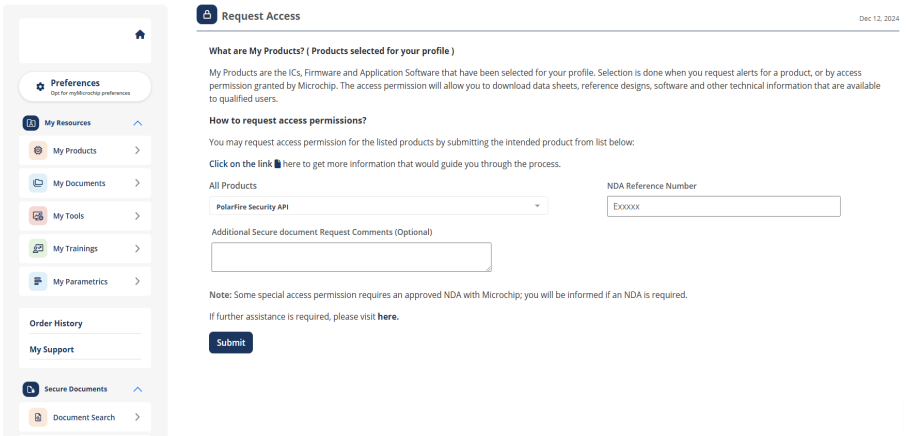
Microchip provides an Athena TeraFire Cryptographic Applications Library (CAL) to access User Cryptoprocessor functions. TeraFire CAL offers functionalities for accessing symmetric key, elliptic curve, public key, hash, random number generation, and message authentication code algorithms. For CAL project access and download, refer to the Steps to Access PolarFire Security CAL and Documentation section. The user application running on the main processor must include CAL APIs to perform the cryptographic operations on the User Cryptoprocessor.

1.1 Steps to Access PolarFire Security CAL and Documentation (Ask a Question)

To request the project design files, perform the following steps:

1. Go to the Microchip Sign-In Page and log in to your My Microchip account.
2. Locate the **Secure Documents** section in the left-hand panel. To expand the menu, click the downward arrow. Within the expanded **Secure Documents** menu, click **Request Access** option. The request access page is displayed on the right side of the panel.
3. From the **All Products** dropdown menu, choose **PolarFire Security API**, and click **Submit**.

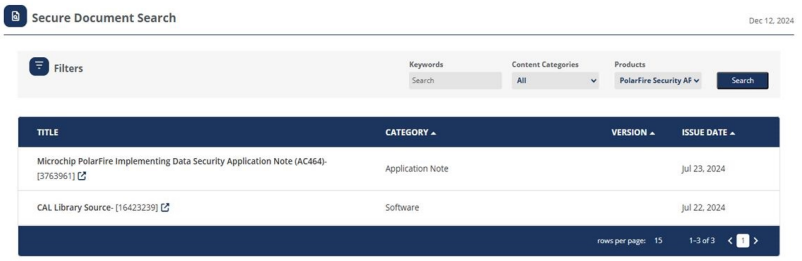
Figure 1-3. Request Access



Please allow 24-48 hours for your access request to be processed and approved.

4. Once the request is approved, to verify access and retrieve the documentation return to the **Secure Documents** section and click **Document Search**.
5. From the **Products** dropdown menu on the right side of the panel, choose **PolarFire Security API**. To initiate the search, click **Search**.
6. After the search is completed, view and download the necessary documentation and the reference design files required for AN4591: PolarFire FPGA Implementing Data Security using User Cryptoprocessor Design.

Figure 1-4. Secure Document Search



In case of any issues during this process, please contact your designated Field Applications Engineer (FAE) or submit a technical support case through My Microchip.

2. Design Requirements (Ask a Question)

The following table lists the hardware and software required to perform cryptographic operations using the User Cryptoprocessor.

Table 2-1. Resource Requirements

Requirement	Version
Host PC Operating System	Windows® 11
Hardware	
PolarFire® Evaluation Kit (MPF300TS-1FCG1152I)	Rev D or later
Software	
Libero® SoC **Note:** Libero SoC Gold, Silver, and Platinum support the use of PolarFire 'S' grade devices.	See the `readme.txt` file provided in the design files for all software versions needed to create this reference design.
SoftConsole	
TeraTerm	5.3

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

3. Prerequisites (Ask a Question)

- Download the design files from the following location: www.microchip.com/en-us/application-notes/an4591. This design is targeted for PolarFire Evaluation kit only.
- Download and install Libero SoC on the host PC from the following location: Libero SoC Documentation.

Important:

- To create Libero project using TCL script, see Appendix 3: Running the TCL Script section.

4. Design Description (Ask a Question)

Microchip offers a freely available RISC-V processor IP core called Mi-V and a software tool-chain to support Mi-V processor-based designs. In this reference design, a Mi-V soft processor core is used as the main processor and the User Cryptoprocessor is used as the coprocessor.

RISC-V, a standard open Instruction Set Architecture (ISA) under the governance of the RISC-V Foundation, offers numerous benefits, including enabling the open source community to test and improve cores at a faster pace than closed ISAs.

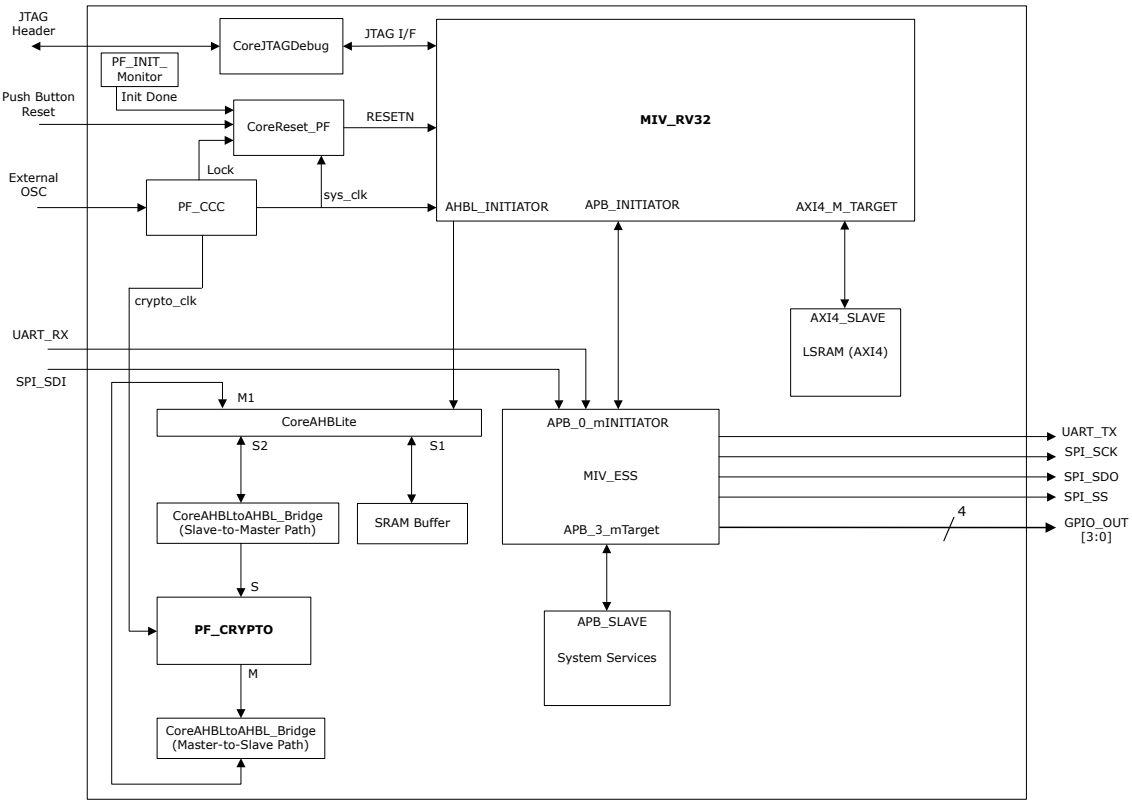
The Libero design provided with this application note shows how to integrate the User Cryptoprocessor in a Mi-V processor subsystem. The SoftConsole project shows how to integrate and build a TeraFire CAL driver into a Mi-V processor application project. A similar process can be used to integrate the User Cryptoprocessor and its CAL driver into other general-purpose processor subsystem environments.

The Mi-V application provided with the reference design demonstrates the Advanced Encryption Standard (AES) algorithm features of the User Cryptoprocessor. It provides a user interface on the host PC using the UART. The user can download and run the other User Crypto sample projects available in the GitHub to explore using the User Cryptoprocessor cryptographic algorithms.

Each PolarFire FPGA has 56 KB of secure Non-Volatile Memory (sNVM), which can be used for storing cryptographic keys. The sNVM pages are accessible through system services to read/write. The reference design integrates PF_SYSTEM_SERVICES IP for sNVM read/write.

The following figure shows a block diagram of the reference design.

Figure 4-1. Reference Design Block Diagram



4.1 Clocking Structure (Ask a Question)

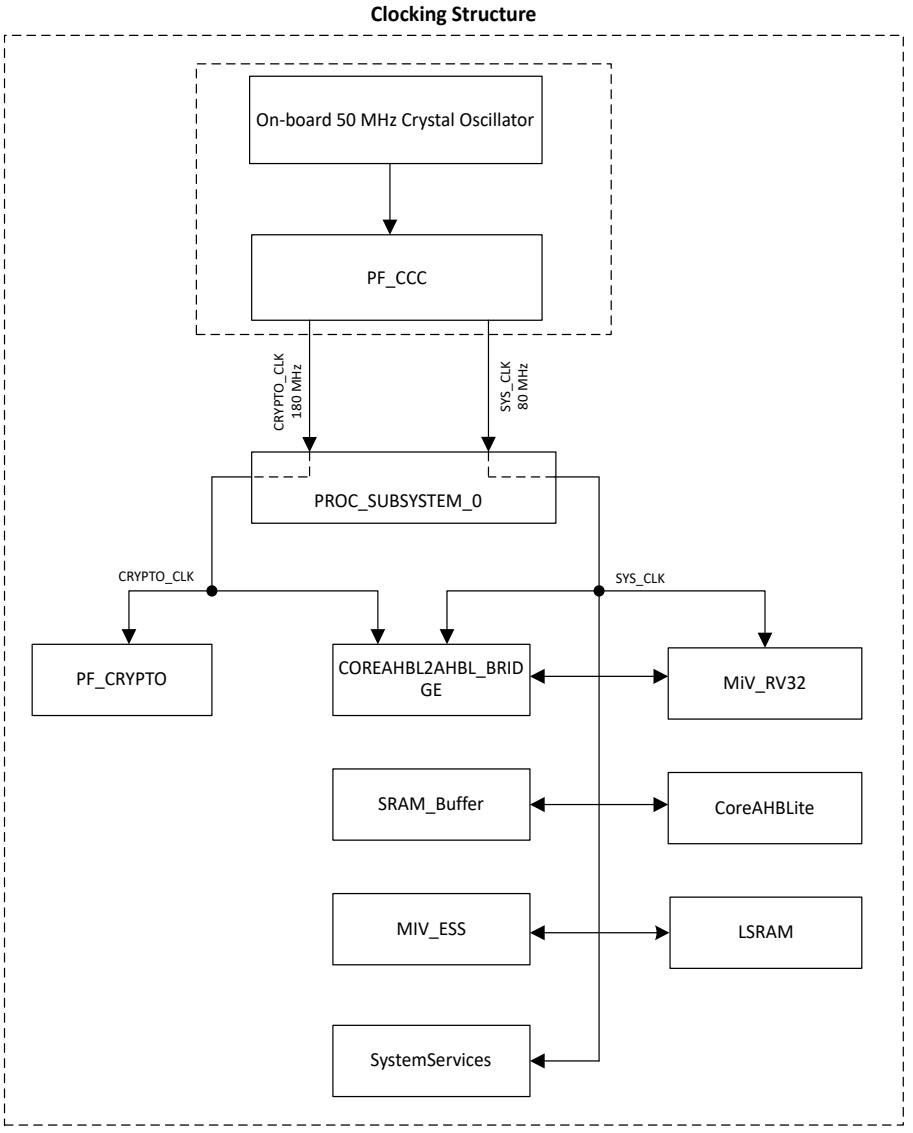
In the following reference design, there are two clock domains. The on-board 50 MHz crystal oscillator is connected to the PF_CCC block, which generates 80 MHz and 180 MHz clocks.

The 180 MHz crypto clock is connected to PF_CRYPTO and COREAHBL2AHBL_BRIDGE blocks.

The 80 MHz system clock is connected to COREAHBL2AHBL_BRIDGE, MIV_RV32, SRAM_Buffer, LSRAM, Core_JTAG_Debug, CoreAHBLite, System Services, and MIV_ESS blocks.

The following figure shows the clocking structure.

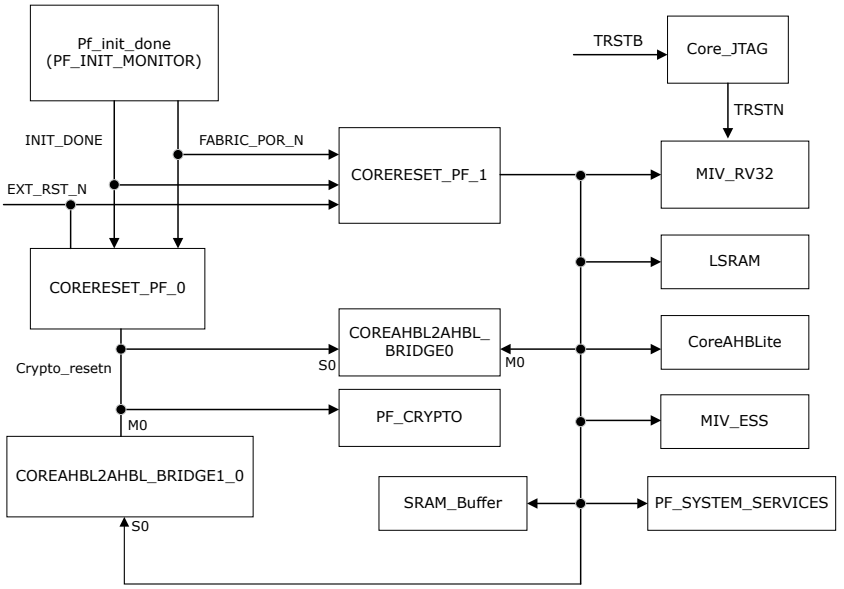
Figure 4-2. Clocking Structure



4.2 Reset Structure (Ask a Question)

The following reference design has two reset domains. The CORERESET_PF_0 is connected to the PF_CRYPTO and COREAHBL2AHBL_BRIDGE blocks. The CORERESET_PF_1 is connected to the COREAHBL2AHBL_BRIDGE, MIV_RV32, SRAM_Buffer, LSRAM, Core_JTAG_Debug, CoreAHBLite, System Services, and MIV_ESS blocks. The following figure shows the reset structure.

Figure 4-3. Reset Structure



4.3 Hardware Implementation (Ask a Question)

To build a Mi-V subsystem for PolarFire FPGAs, use the Libero SoC design suite to create an FPGA design using the Mi-V processor core and peripheral cores. The following table lists the IP cores used in the reference design.

Table 4-1. IP Cores Used in the Reference Design

IP Core	Description
MIV_RV32	Mi-V soft processor
Core_JTAG_DEBUG	Facilitates the connection of Joint Test Action Group (JTAG) compatible soft core processors to the JTAG header for debugging. It provides fabric access to the JTAG interface using the UJTAG macro.
PF_INIT_MONITOR	The System Controller uses this macro to check the status of device initialization. The device initialization includes SRAM initialization from µPROM, sNVM, or SPI Flash. The DEVICE_INIT_DONE signal is used as a reset.
CoreAHBLite	Multi-master AHB-Lite bus
COREAHBL2AHBL_BRIDGE	AHB to AHB bridge connects two AHB-Lite buses, clocked by asynchronous clocks. This is required because the User Cryptoprocessor clock is different from the rest of the Mi-V processor subsystem clock.
PF_CRYPTO	Macro to access hard User Cryptoprocessor
PF_SRAM_AHBL_AXI	PolarFire LSRAM; used as system memory for Mi-V processor. SRAM buffer; used as memory buffer for User Cryptoprocessor.
PF_CCC	Macro to access PolarFire Clock Conditioning Circuit (CCC) block. It is used to synthesize 80 MHz and 180 MHz clock frequencies from the CCC with an on-board 50 MHz reference clock.
PF_SYSTEM_SERVICES	The PF_SYSTEM_SERVICES provides access to the System Services supported by the PolarFire device. These are System Controller actions initiated from the user design, using the PF_SYSTEM_SERVICES.
MIV_ESS	Multi-featured, highly-configurable Extended SubSystem (ESS) which supports both bootstrap and base peripherals. It is specifically designed to use with MIV_RV32 soft processor. It has APB interface to access subsystem memory-mapped peripherals and other cores like SPI, GPIO and UART.

Important: In this design, the CoreAHBLite v. 5.6.105 is used as the latest CoreAHBLite v. 6.1.101 has some known issue and is being updated.

To create a programming file, configure and connect the IP cores listed in Table 4-1 and then run the Libero design flow. For more information about how to build a Mi-V processor subsystem for PolarFire devices, see AN4997: PolarFire FPGA Building a Mi-V Processor Subsystem. This Mi-V processor subsystem can be extended to integrate a User Cryptoprocessor into the system.

The MIV_RV32 processor core comprises an instruction fetch unit, an execution pipeline, and a data memory system. In the Mi-V processor memory map, the 0x8000_0000 to 0x8FFF_FFFF range is defined for the AXI4 interface to access the LSRAM; the 0x7000_0000 to 0x7FFF_FFFF range is defined for the MIV_ESS's APB interface; and the 0x6000_0000 to 0x6FFF_FFFF range is defined for the AHBLite I/O interface. The MIV_RV32 processor's reset vector address is set to 0x80000000. The processor's reset vector address is configurable. The MIV_RV32's reset is an active-low signal, which must be de-asserted in sync with the system clock through a reset synchronizer. For more information about the MIV_RV32 core, see MIV_RV32 User Guide from Libero Catalog.

The MIV_RV32 processor accesses the application execution memory using the AXI4 interface. The AXI4 bus interface is configured to provide a 256 MB memory slot beginning at the address 0x80000000. The PolarFire LSRAM memory block (LSRAM_0 instance) is connected to this slot, and the LSRAM acts as an application execution memory for the Mi-V processor.

The MIV_RV32 processor directs the data transactions between addresses 0x60000000 and 0x6FFF_FFFF to the AHBL interface. The AHBL interface is connected to the CoreAHBLite_0 bus to communicate with peripherals connected to its slave slots. The CoreAHBLite_0 bus instance is configured to provide 16 slave slots, each of size 64 KB. The sixteen 64 KB slots consume a total address space of 16 × 64 × 1024 = 2^20 bytes, and can be addressed using a 20-bit address bus. The CoreAHBLite_0 bus maps the connected peripherals within the address range, using only the lower 20-bits of the MMIO bus address.

The following table summarizes the memory map of the reference design.

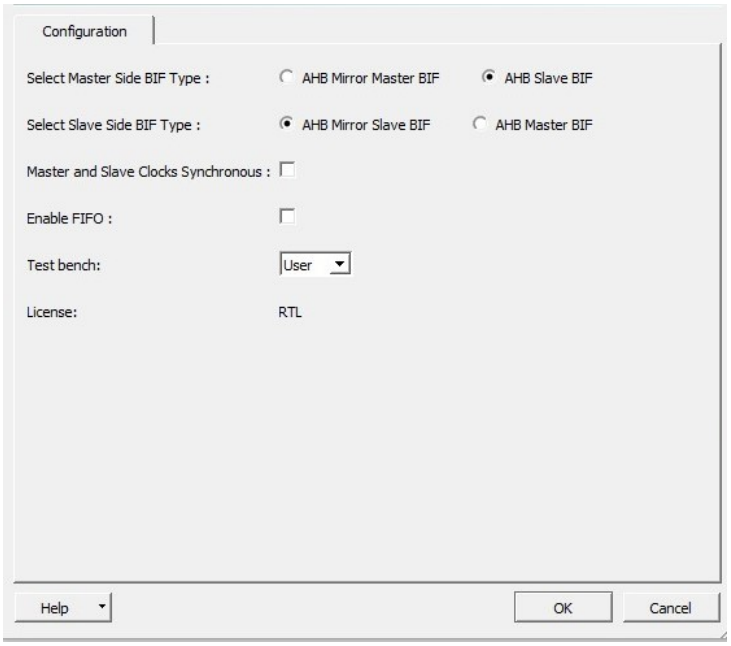
Table 4-2. System Memory Map

Component	Description	Memory Map
PF_CRYPTO_proc_0	User Cryptoprocessor	0x62000000 to 0x6200FFFF
PF_SYSTEM_SERVICES_C0_0	System Services	0x73000000 to 0x73FFFFFF
MIV_ESS_UART	UART peripheral	0x71000000 to 0x71FFFFFF
SRAM_Buffer_0	Memory buffer for User Cryptoprocessor	0x61000000 to 0x61FFFFFF
MIV_ESS_GPIO	GPIO peripheral	0x75000000 to 0x75FFFFFF
MIV_ESS_SPI	SPI peripheral	0x76000000 to 0x76FFFFFF
LSRAM_0	Mi-V processor system memory	0x80000000 to 0x8FFFFFFF

In this reference design, the User Cryptoprocessor clock (crypto_clk) is configured to operate at 180 MHz, and the clock for the rest of the Mi-V subsystem (sys_clk) operates at 80 MHz. This reference design uses the CoreAHBL2AHBL_Bridge IP to provide clock domain crossing between sys_clk and crypto_clk.

The CoreAHBL2AHBL_Bridge IP functions as a bridge between the AHB master and AHB slave, where master and slave operate in two clock domains that are asynchronous in nature. The following figure shows the CoreAHBL2AHBL_Bridge IP Configurator.

Figure 4-4. CoreAHBL2AHBL_Bridge Configurator

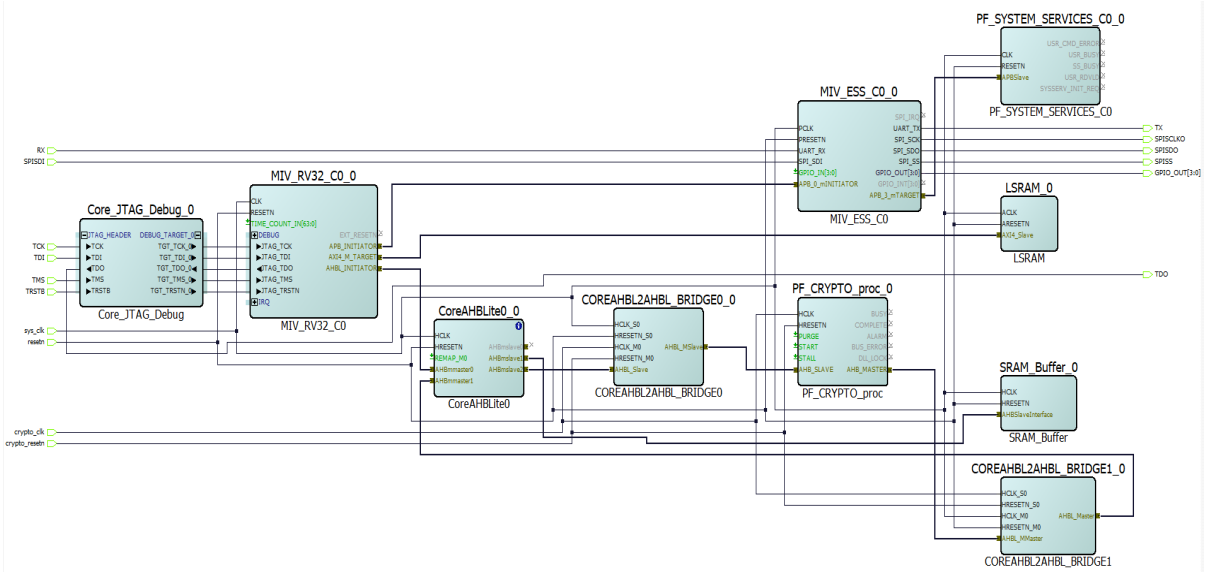


The CoreAHBL2AHBL_Bridge_0 is configured to connect the User Cryptoprocessor to the Mi-V processor peripheral bus for control and primary data input and output. In this configuration, the sys_clk must be connected to the bridge's slave interface clock (HCLK_S0), and the crypto_clk must be connected to the bridge's master interface clock (HCLK_M0).

The CoreAHBL2AHBL_Bridge_1 is configured to connect the User Cryptoprocessors AHB master port to the Mi-V processors peripheral bus for DMA transactions. In this configuration, the sys_clk must be connected to the bridge's master interface clock (HCLK_M0), and the crypto_clk must be connected to the bridge's slave interface clock (HCLK_S0).

The following figure shows the SmartDesign view of the Mi-V processor subsystem with a User Cryptoprocessor.

Figure 4-5. Mi-V Processor Subsystem with User Cryptoprocessor



For more details on component configurations and connections, see the Libero project created using provided TCL scripts.

4.3.1 FPGA Resource Utilization (Ask a Question)

The following figure shows the reference design resource utilization report under **Synthesize** in **Design Flow** window.

Figure 4-6. Design Resources Utilization-Evaluation Kit

Module Name	Fabric 4LUT	Fabric DFF	Interface 4LUT	Interface DFF	Single-Ended I/O	uSRAM (64x12)	LSRAM (20K)	Chip Globals	PLL
Top	10880	3499	6012	6012	12	21	160	5	1
Primitives	1	0	0	0	12	0	0	0	0
CCC_0_0	0	0	0	0	0	0	0	2	1
CORERESET_PF_C0_0	2	16	0	0	0	0	0	0	0
CORERESET_PF_C1_0	0	17	0	0	0	0	0	1	0
PROC_SUBSYSTEM_0	10877	3466	6012	6012	0	21	160	2	0
COREAHBL2AHBL_BRI...	38	135	0	0	0	0	0	0	0
COREAHBL2AHBL_BRI...	34	140	0	0	0	0	0	0	0
CoreAHBLite0_0	394	107	0	0	0	0	0	0	0
Core_JTAG_Debug_0	250	17	0	0	0	0	0	2	0
LSRAM_0	1701	168	4608	4608	0	0	128	0	0
MIV_ESS_C0_0	580	330	24	24	0	2	0	0	0
MIV_RV32_C0_0	7013	2148	228	228	0	19	0	0	0
PF_SYSTEM_SERVICES_...	412	375	0	0	0	0	0	0	0
SRAM_Buffer_0	455	46	1152	1152	0	0	32	0	0

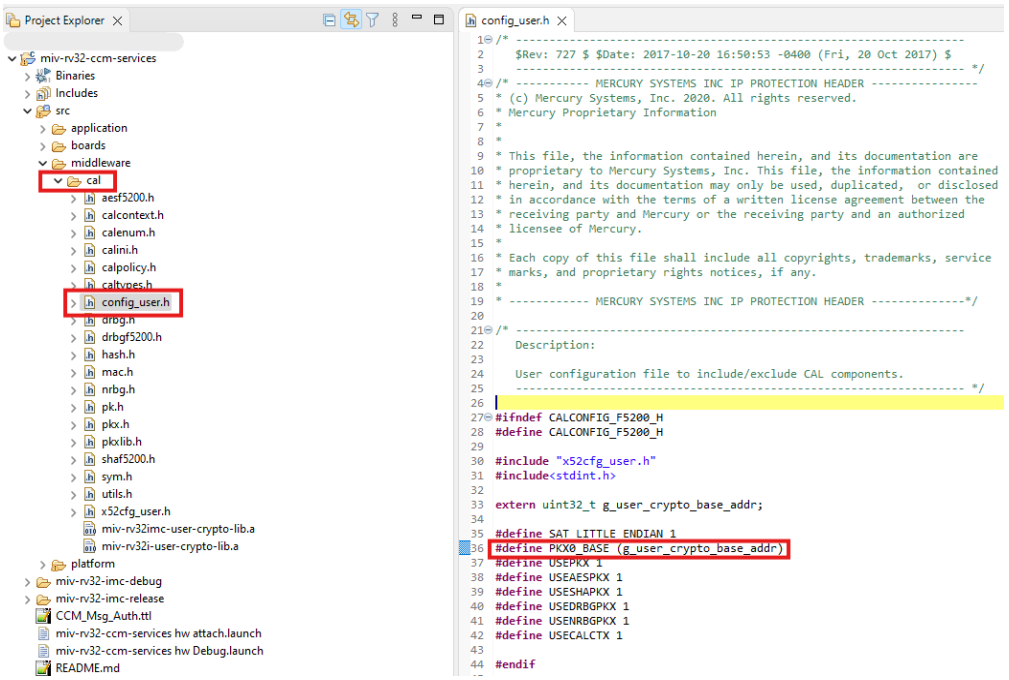
4.4 Software Implementation (Ask a Question)

For information about creating and building a SoftConsole project for the Mi-V processor subsystem, see AN4997: PolarFire FPGA Building a Mi-V Processor Subsystem.

To use the User Cryptoprocessor services in the application, the user needs to download the SoftConsole example project firmware from the GitHub, which also contains the pre-compiled PolarFire Crypto CAL library. The User Crypto CAL folder contains `config_user.h` file for driver configuration. In the `config_user.h file`, define the PKX0_BASE macro as the base address of the User Cryptoprocessor, according to the Libero design.

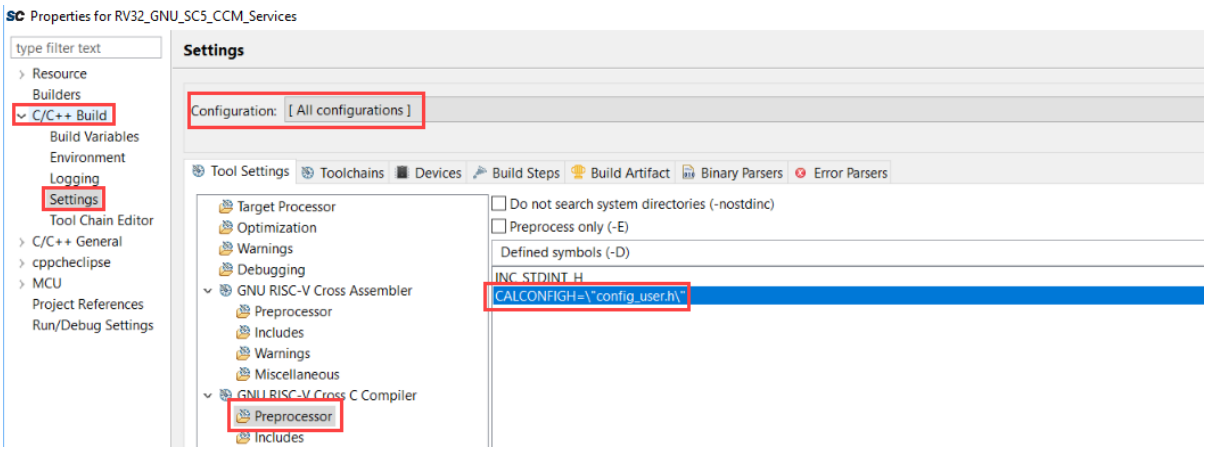
Important: To access the source code of the PolarFire User Crypto CAL library, the user can refer to the Steps to Access PolarFire Security CAL and Documentation section.

Figure 4-7. User Crypto CAL Driver



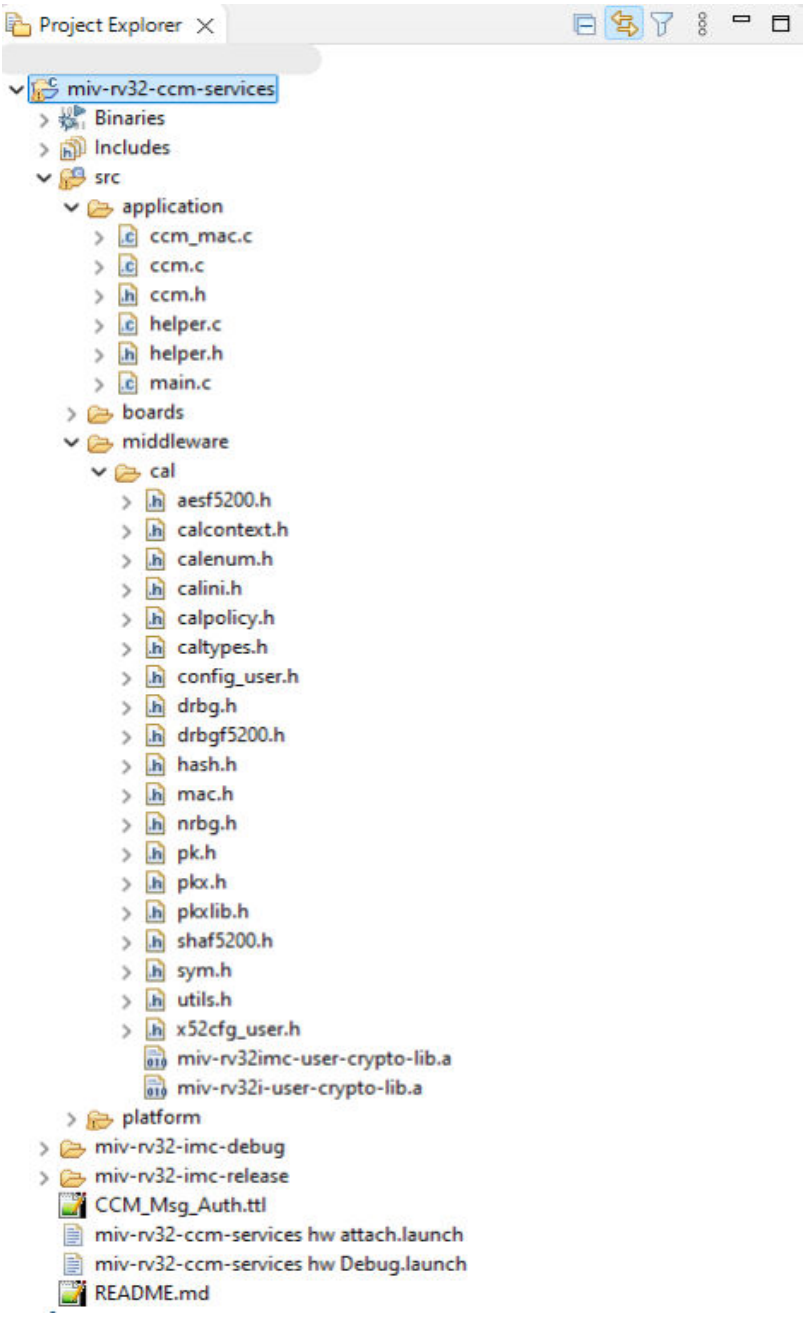
For both debug and release configurations, browse and add the `config_user.h` file to **GNU RISC-V Cross C Compiler** > **Preprocessor**, as shown in the following figure.

Figure 4-8. CALCONFIGH



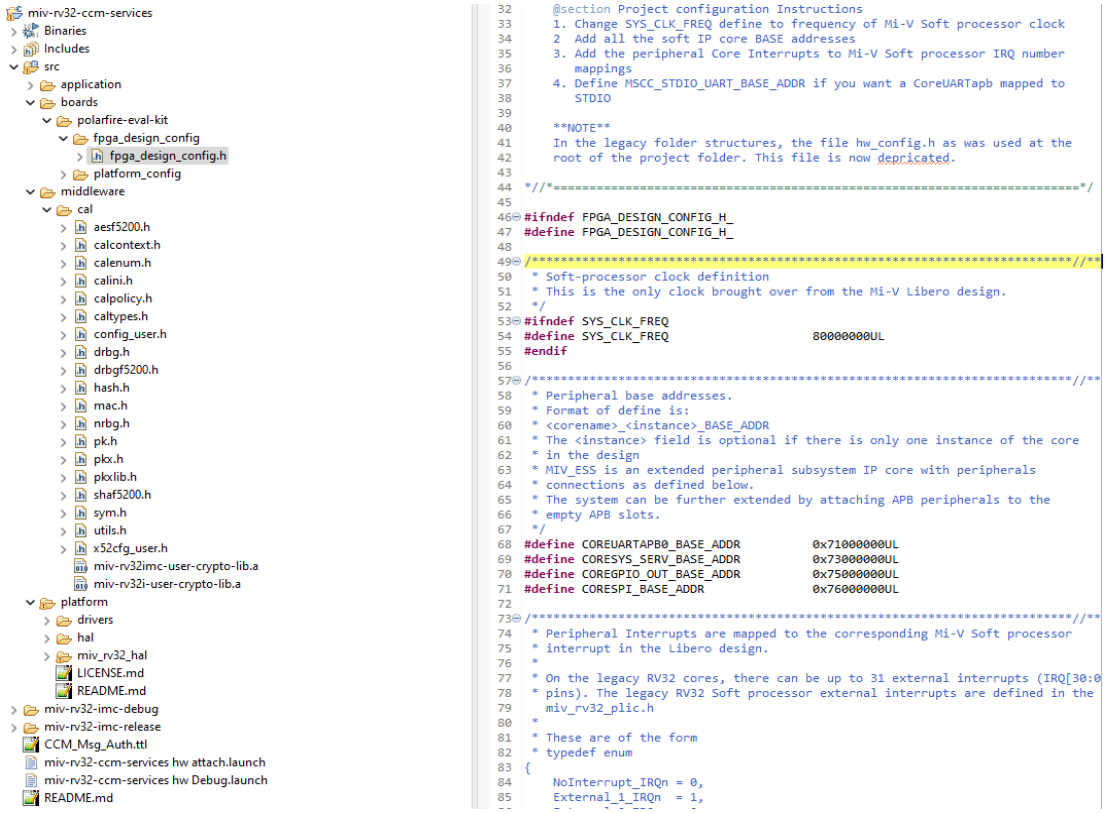
The following figure shows the intended directory structure for a project based on MIV_RV32, using SoftConsole.

Figure 4-9. SoftConsole Project Directory Structure



The UART base address, SPI base address, GPIO base address, System Services base address, and system clock frequency are provided in the `fpga_design_config.h` file.

Figure 4-10. Peripheral Base Address



4.4.1 Linker Script Update (Ask a Question)

A sample linker script file (`miv-rv32-ram.ld`) is provided along with the `MIV_RV32 HAL` files. This linker script assumes that the SRAM is connected to the Mi-V processor memory space. The start address and size of the memory space must correspond with the Libero design.

A crypto_ram user section is defined in the linker script (see the following figure) to map the User Crypto input and output buffers to a common memory located on the Mi-V processor AHBL interface. The common memory is located at address 0x61000000.

Figure 4-11. Linker Script

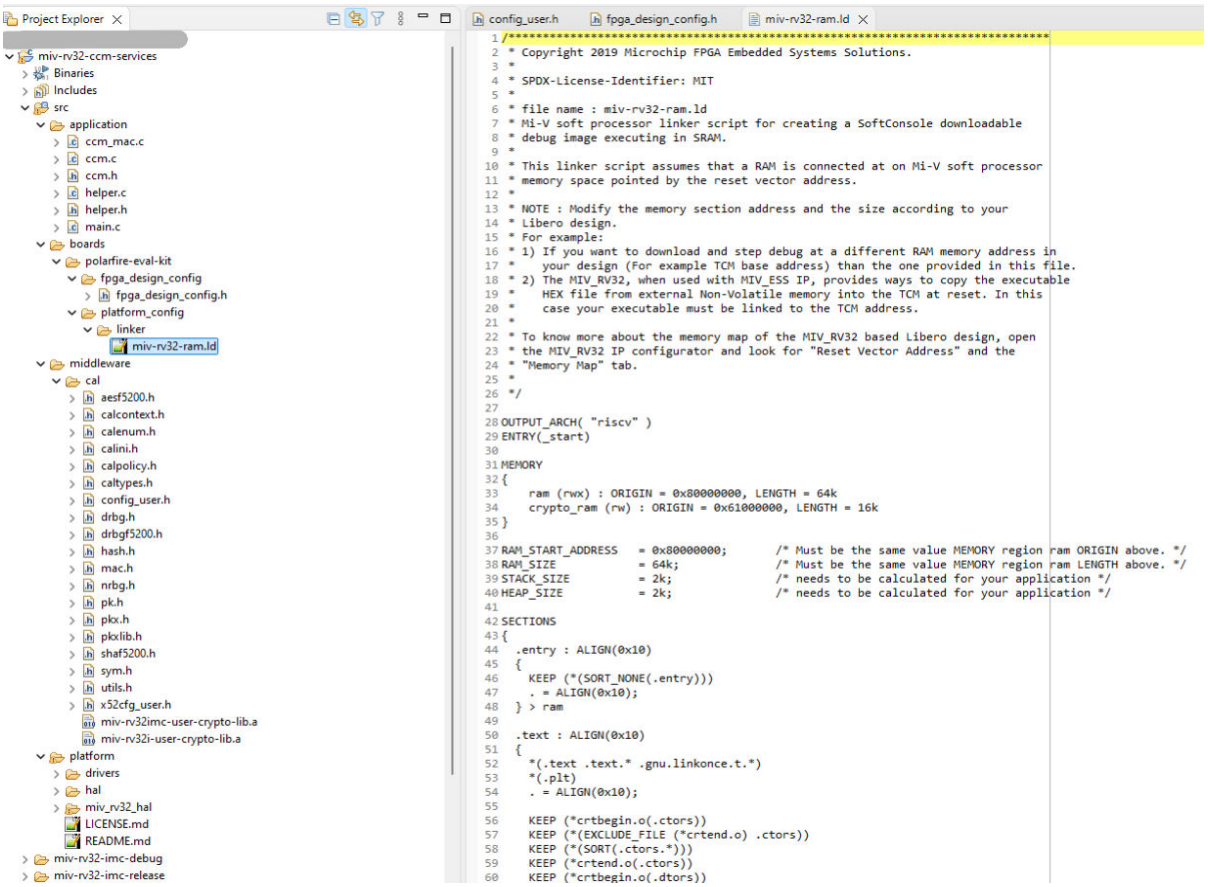
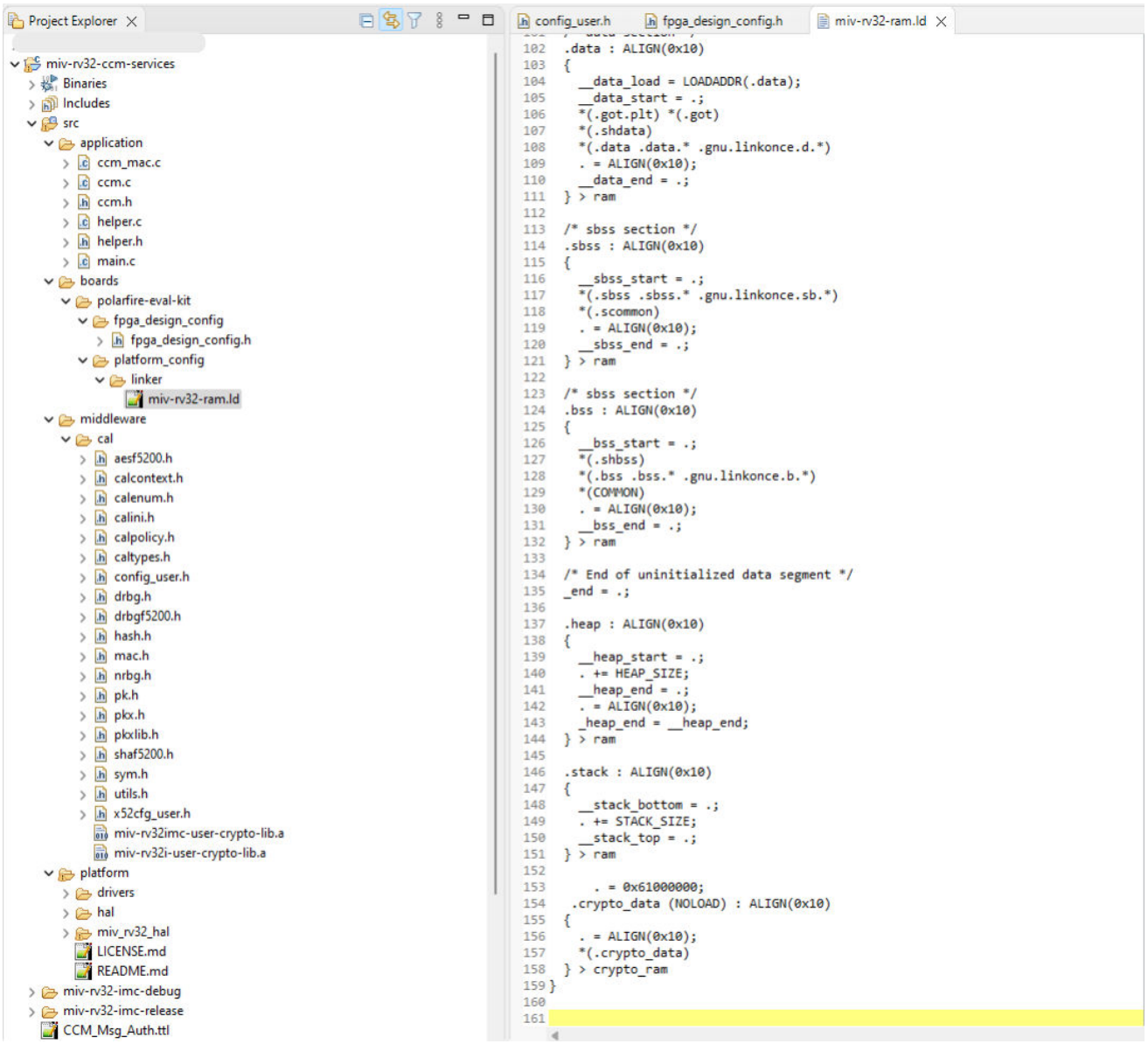


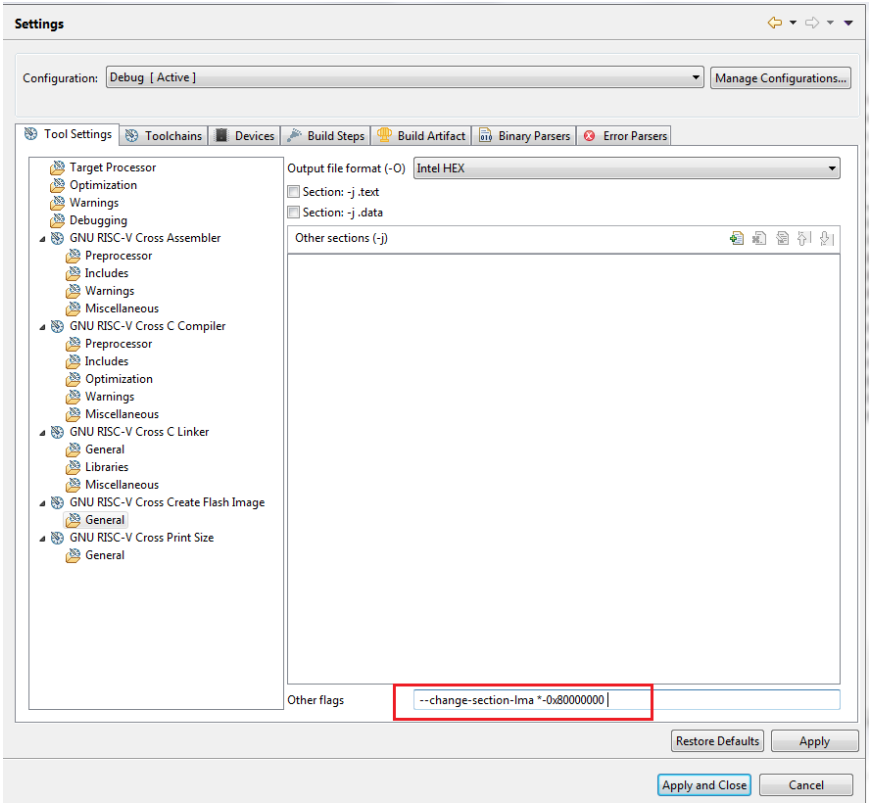
Figure 4-12. Linker Script (Continued)



4.4.2 RISC-V Flash Image Setting (Ask a Question)

As shown in the following figure, add the `--change-section-lma *-0x80000000` command to remove the non-linear address (the first line) of the HEX file.

Figure 4-13. RISC-V Flash Image Setting



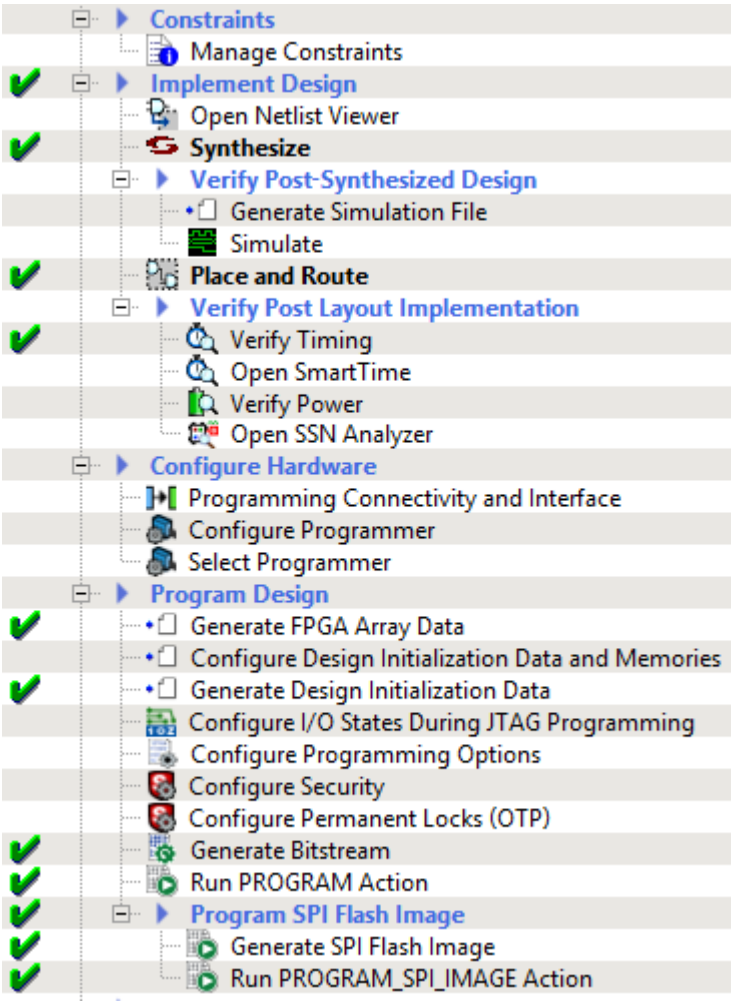
4.5 LSRAM Initialization from SPI Flash (Ask a Question)

This section describes how to load the user application hex image file into the LSRAM from SPI flash using System Controller.

To configure Design Initialization Data and Memories, perform the following steps:

1. Generate the Libero_Project using the TCL script. See Appendix 4: Running the TCL Script section.
2. Double click **Configure Design Initialization Data and Memories** from the **Design Flow** window, see the following figure.

Figure 4-14. Configure Design Initialization Data and Memories

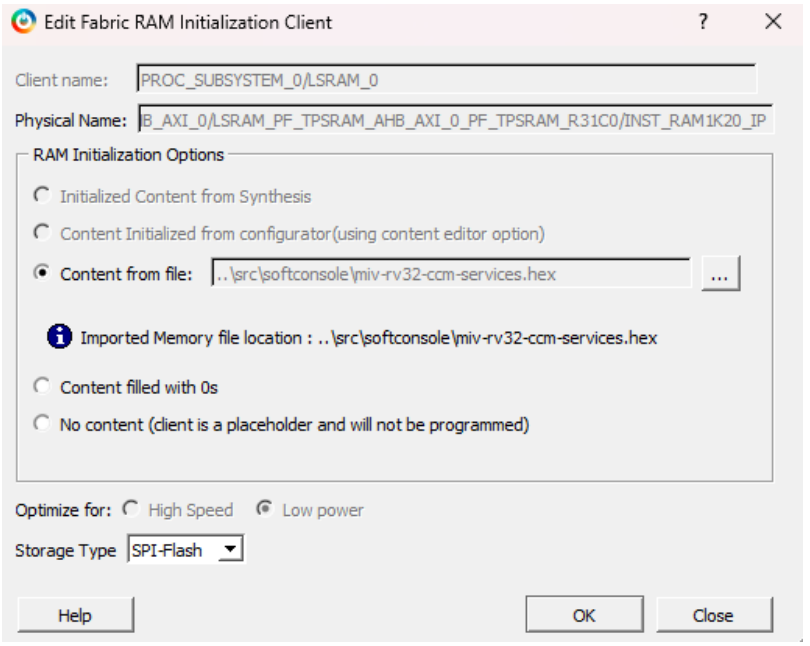


3. Click the **Fabric RAMs** tab and select the LSRAM instance that needs to be initialized and click **Edit**, as shown in the following figure. In this design, LSRAM instance must be initialized with the user application.

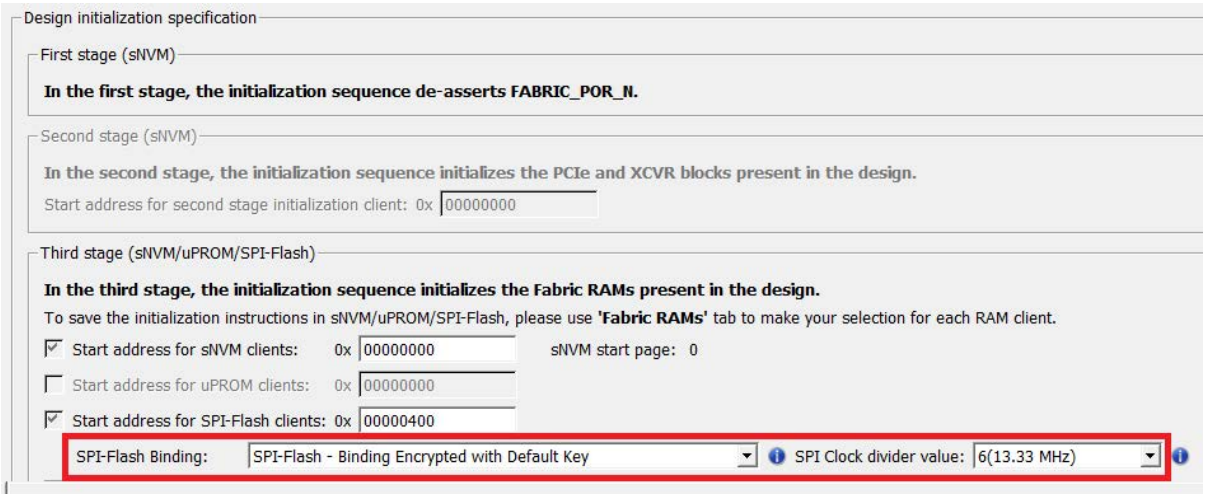
Figure 4-15. Fabric RAMs

4. Double click the LSRAM instance to add the initialization client and storage location. In the **Edit Fabric RAM Initialization Client dialog**, select the **Content from file** option, locate the `miv-rv32-ccm-services.hex` file from the `mpf_an4591_df/HW/src/softconsole` folder, and select **Storage Type as SPI-Flash**, and then click **OK**, as shown in the following figure.

Figure 4-16. Edit Fabric RAM Initialization Client



5. Click the **Design Initialization** tab, and ensure **SPI-Flash - Binding Encrypted with Default Key** is selected for **SPI-Flash Binding** type as shown in the following figure. SPI clock divider value must be set to 6. Refer to the note given below for more information. In this reference design, the initialization client is stored in the SPI flash in an encrypted format with authentication. When this option is selected, the design initialization client file (<root>_uic.bin) is encrypted with the default encryption key. When the Default key is selected, the user does not need to specify any other details. If required, enable user security protection using UEK1/UEK2 for the SPI-Flash client..



Important: The SPI clock divider value specifies the required SPI SCK frequency to read the initialization data from SPI Flash. The SPI Clock divider value must be selected based on the external SPI Flash operating frequency range.

Table 4-3. SPI Clock Divider Value

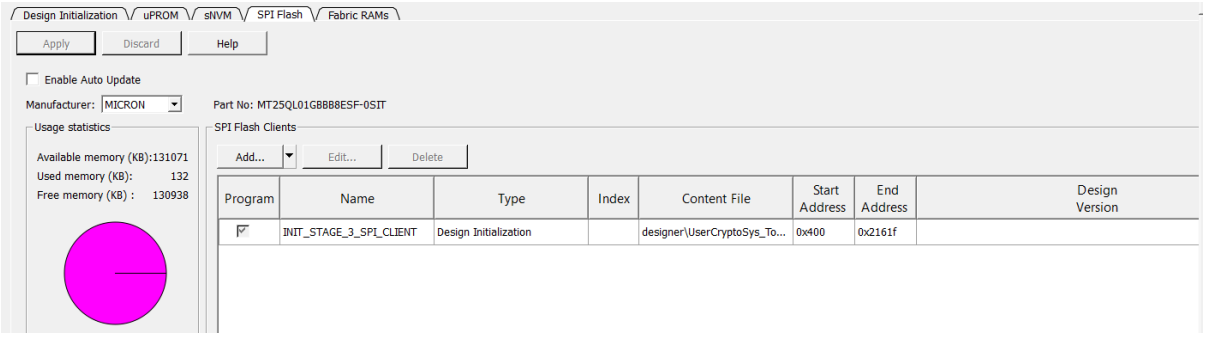
SPI Clock Divider Value	SCK Frequency
1	80 MHz
2	40 MHz
4	20 MHz
6	13.3 MHz

6. Click **Generate Initialization** clients under the **Design Initialization** tab to generate the External
SPI-Flash (Non-authenticated) client.
7. When the initialization clients are generated, the Generate Initialization clients status window is displayed, see the following figure.

Figure 4-17. Generate Design Initialization Data

8. Select the **SPI Flash** tab to verify that the SPI flash client is generated, see the following figure.

Figure 4-18. SPI Flash Client Verification



Configuration of Design Initialization Data and Memories is successfully completed.

Important: In the scenario where a PolarFire device is first programmed with a design with an sNVM client and then reprogrammed with a (different) design without an sNVM client, upon completion of programming with the second design, the sNVM client will not be erased. In such a case, if there are locked sNVM pages, writing to those pages using a system service call fails. For that enable, **Sanitize all sNVM pages in ERASE action** in **Generate Bistream Configure Options**.

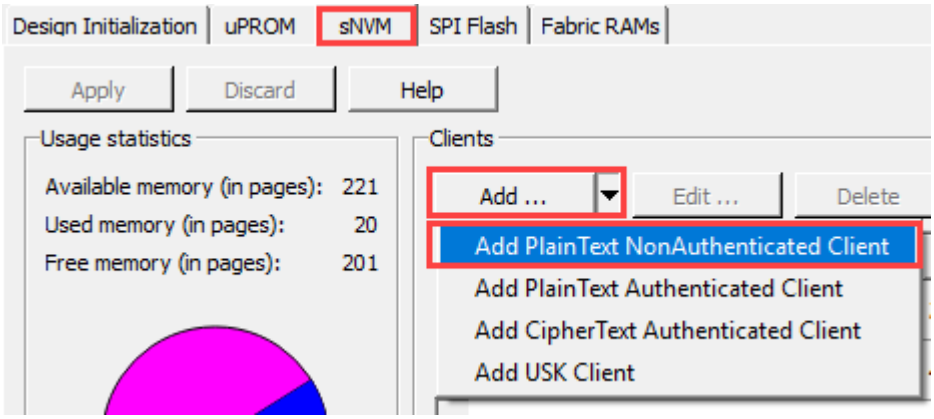
4.5.1 Adding a Dummy Client to Unlock the sNVM Pages (Ask a Question)

In this design, the AES key provided by the user is stored in the sNVM on page#4 using a system service call. Hence, a dummy client to clear the sNVM page needs to be added to the sNVM configurator. To add a dummy client, perform the following steps:

1. In the Libero **Design Flow** window, double click **Configure Design Initialization Data and Memories**.
2. In the Design and Memory Initialization page, select the **sNVM** tab.

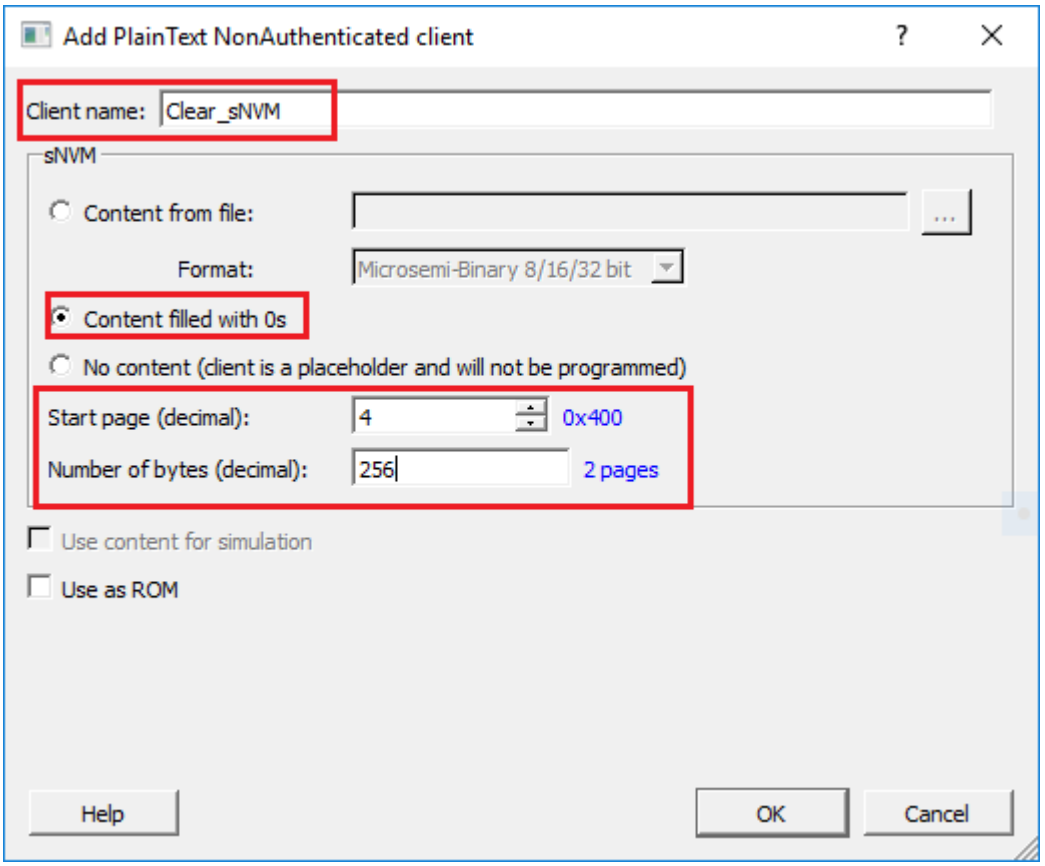
3. In the sNVM Clients pane, click **Add...** > **Add PlainText NonAuthenticated Client.**, see the following figure.

Figure 4-19. Adding Dummy Client



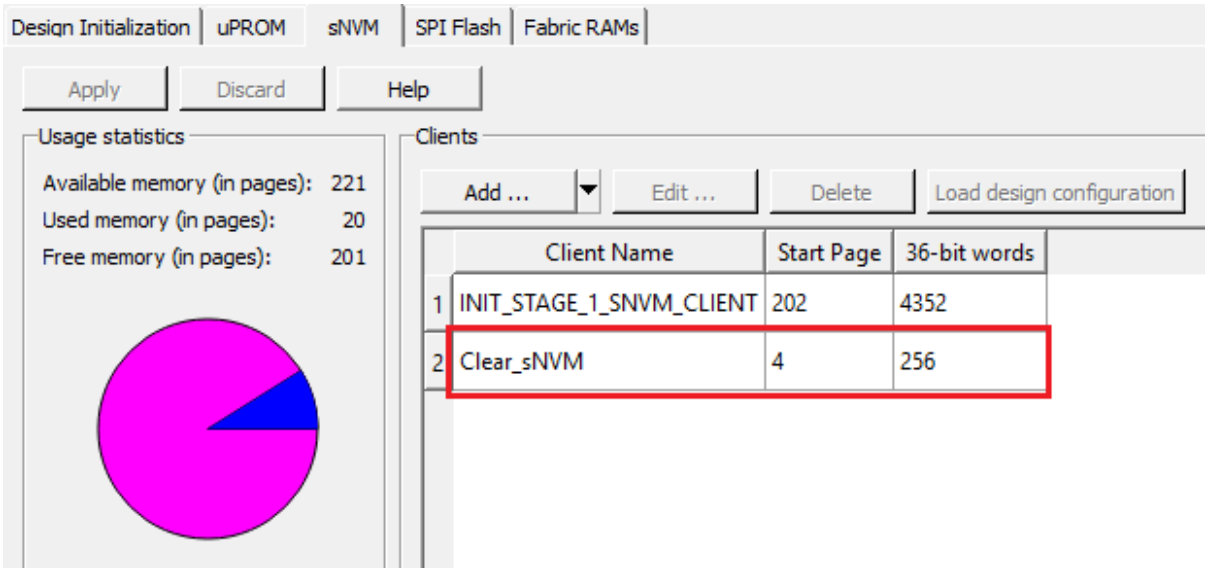
4. In the **Add PlainText NonAuthenticated client** window, select **Content filled with 0s** radio button and enter:
   - **Client name**: Clear_sNVM
   - **Start page (decimal)**: 4
   - **Number of bytes (decimal)**: 256
5. Click **OK** to create a dummy client.

Figure 4-20. Dummy Client Parameters



The following figure shows the dummy client added to the sNVM Clients pane.

Figure 4-21. Dummy Client



5. Programming the PolarFire Device and SPI Flash (Ask a Question)

This section describes how to program the PolarFire device and SPI Flash.

To program the PolarFire device, perform the following steps:

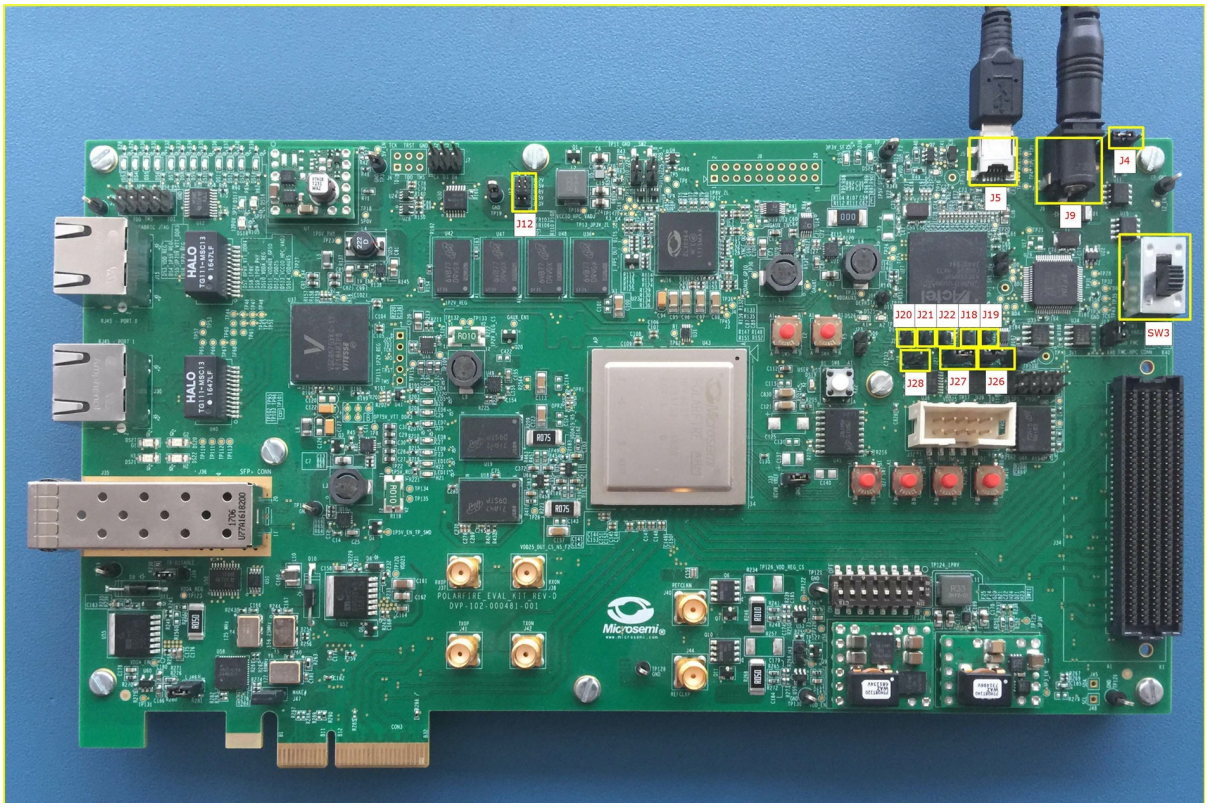
1. Ensure that the jumpers on the evaluation board are set as specified in the following table.

Table 5-1. Jumper Settings—Evaluation Board

Jumper	Description
J18, J19, J20, J21, and J22	Close pin 2 and 3 for programming the PolarFire FPGA through FTDI
J28	Close pin 1 and 2 for programming through the on-board FlashPro5
J26	Close pin 1 and 2 for programming through the FTDI SPI
J27	Close pin 1 and 2 for programming through the FTDI SPI
J23	Open pin 1 and 2 for programming SPI Flash
J4	Close pin 1 and 2 for manual power switching using SW3
J12	Close pin 3 and 4 for 2.5 V

2. Connect the power supply cable to the J9 connector on the evaluation board.
3. Connect the USB cable from the host PC to J5 (FTDI port) on the evaluation board.
4. Power on the evaluation board using the SW3 slide switch.
   The following figure shows the PolarFire Evaluation Kit set up to be programmed.

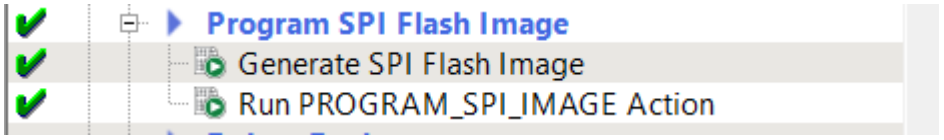
Figure 5-1. Board Setup-Evaluation



5. Refer Appendix 4: Running the TCL Script to create Libero Project and open the **project**.
6. Click **Run PROGRAM Action** to program the device.

7. Double click **Run Program_SPI_IMAGE Action** to program the SPI flash. A green tick mark is displayed after the successful generation, see the following figure.

Figure 5-2. Program SPI Flash Image



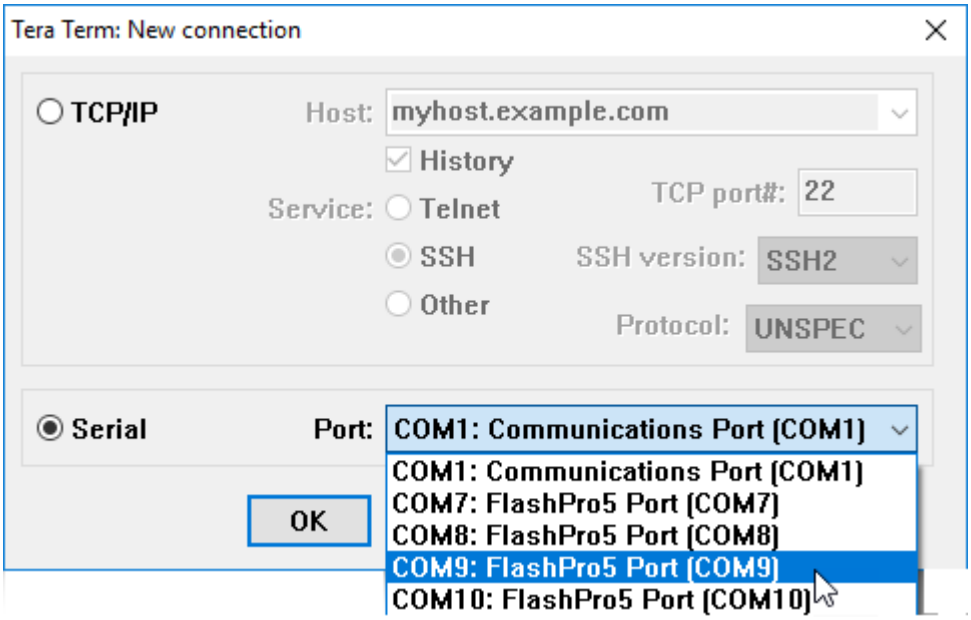
5.1 TeraTerm Setup (Ask a Question)

The user application provides a user interface on the TeraTerm terminal through the UART interface.

To set up the TeraTerm program, perform the following steps:

1. Ensure that the USB cable connects the host PC to the **J5** (USB) port on the PolarFire Evaluation board.
2. Start TeraTerm.
3. Select **Serial** as the Connection type.
4. Set the Serial **Port** to the second highest COM port number from the drop-down list, see the following figure. For example, **COM9: FlashPro5 Port [COM9]** in this instance.

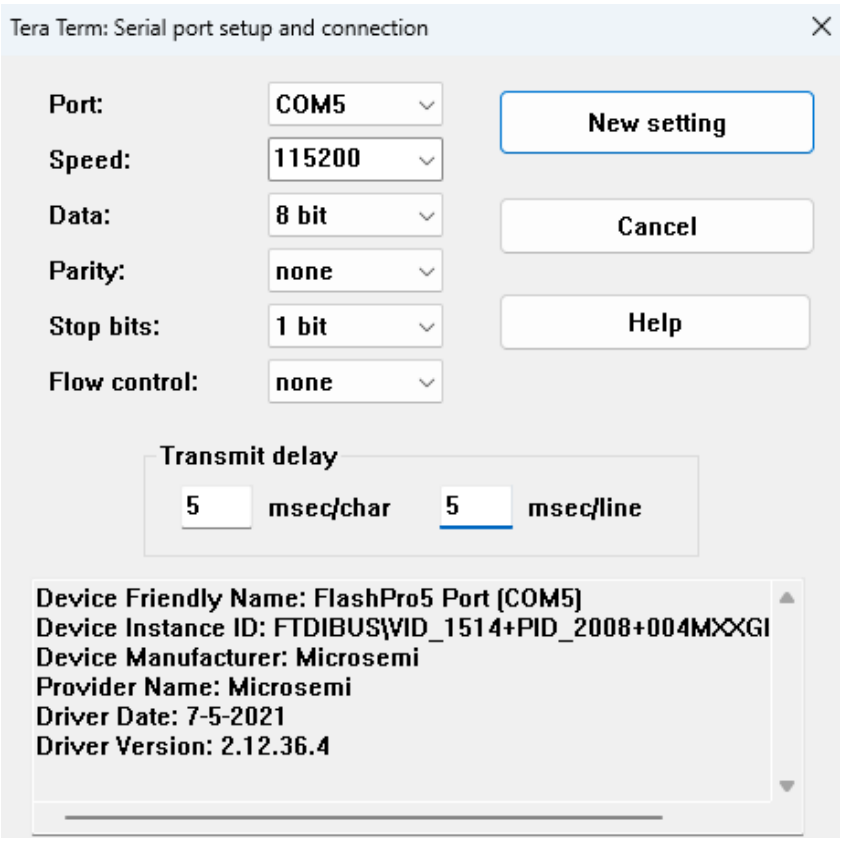
Figure 5-3. Select Serial as the Connection Type



Important: The COM port number is system specific.

5. In the **TeraTerm** window, navigate to **Setup** > **Serial port...**, set;
   - **Baud rate**: 115200
   - **Transmit delay**: 5 msec/char and 5 msec/line
     Rest of the serial port settings must be at default state, see the following figure, and click **OK**.

Figure 5-4. TeraTerm Configuration

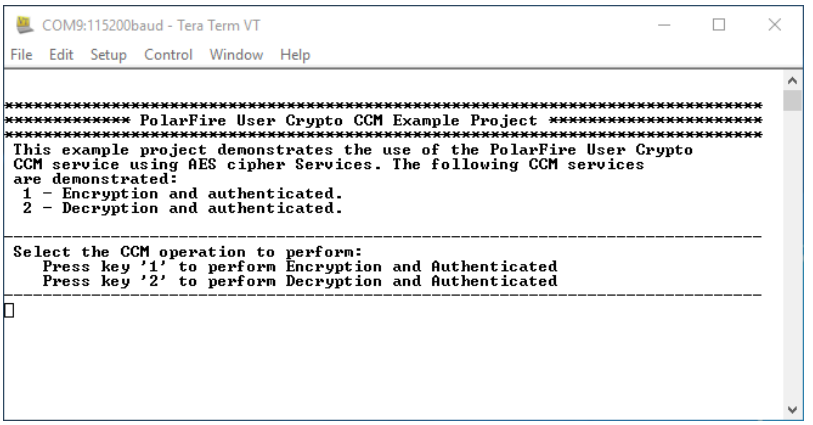


This completes the TeraTerm program setup.

6. Running the Demo (Ask a Question)

After the device is programmed, power cycle the board. The application prints the menu on the Tera Term program through the UART interface, as shown in following figure.

Figure 6-1. Application Menu



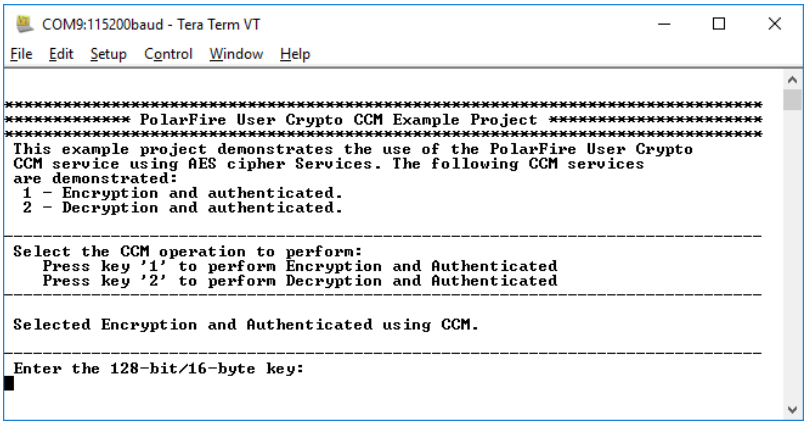
Use the following test vector to verify the AES CCM encryption and decryption operation:

- AES Key = 404142434445464748494A4B4C4D4E4F
- Nonce = 10111213141516171819 1A1B
- Additional authentication data (AAD) = 000102030405060708090A0B0C0D0E0F10111213
- Input data to Encrypt and authenticate = 20212223242526272829 2A2B2C2D2E2F3031323334353637

To run the demo, perform the following steps:

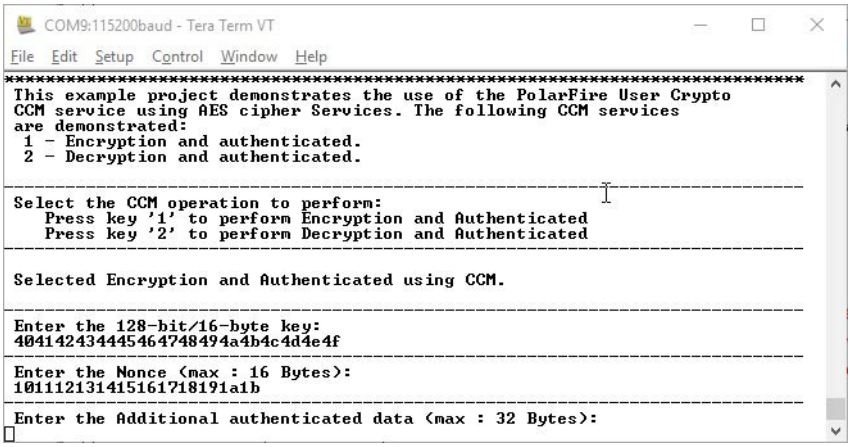
1. Press '1' to perform CCM AES-128 encryption using the User Cryptoprocessor. The application prompts to enter a 128-bit key, see the following figure. The application securely stores the AES key into sNVM using the System Service function call.

   Figure 6-2. Application Menu—Enter 128-bit Key



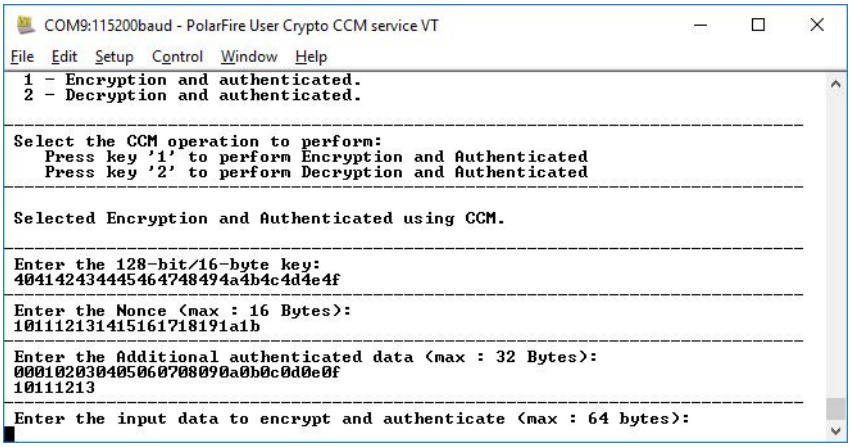
2. Enter the AES key provided in the test vector and press Enter. The application prompts for Nonce.
3. Enter the Nonce provided in the test vector and press Enter. The application prompts to enter AAD, as shown in the following figure.

Figure 6-3. Application Menu—Additional Authentication Data



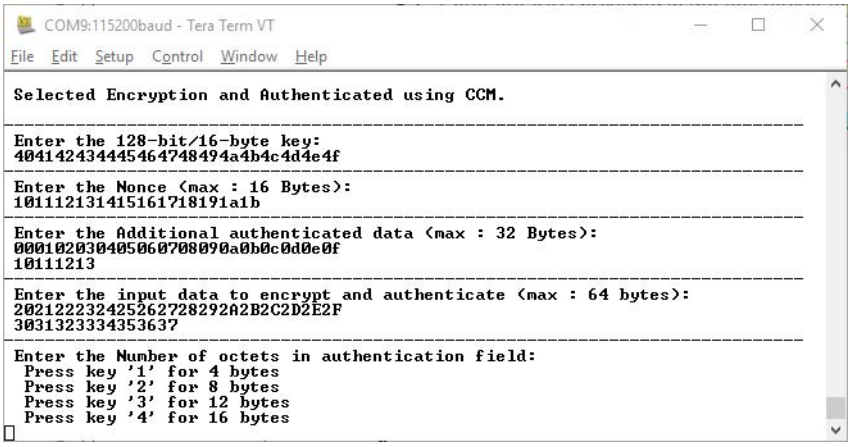
4. Enter the AAD provided in the test vector and press Enter. The application prompts to enter input data to encrypt and authenticate, as shown in the following figure.

Figure 6-4. Application Menu—Input Data to Encrypt and Authenticate



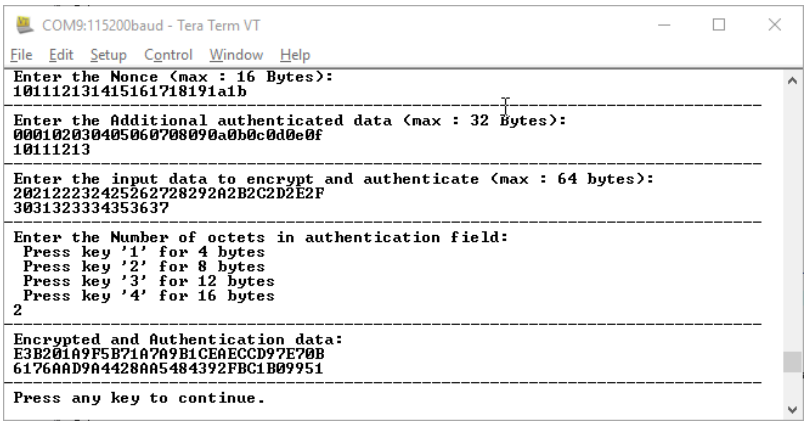
5. Enter the input data to encrypt and authenticate provided in the test vector and press Enter. The application prompts to enter the number of octets in the authentication field, as shown in the following figure.

Figure 6-5. Application Menu—Number of Octets in authentication Field



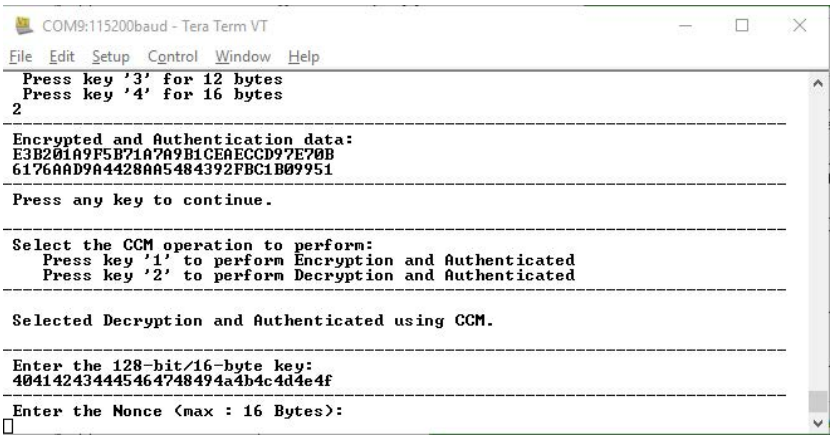
6. Press '2' for Encrypted and Authentication data. The result of the CCM AES encryption is printed on the Tera Term program, as shown in the following figure. Observe that the result is matched with the test vector, as shown in the following figure.

Figure 6-6. Application Menu—Encrypted and Authentication Data



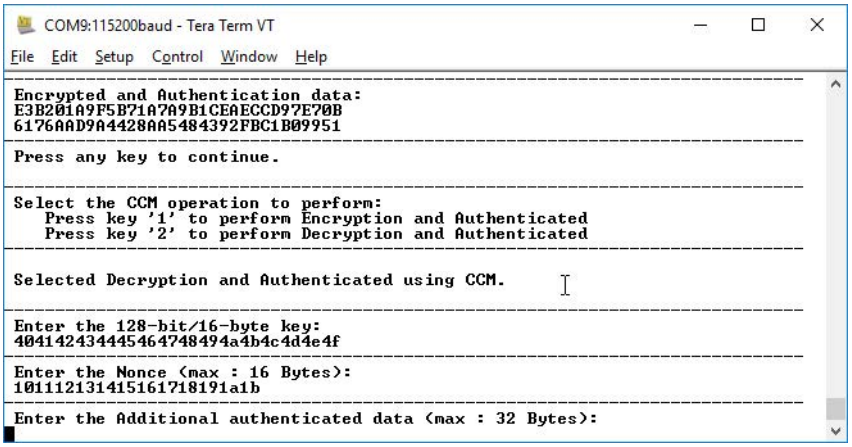
7. Press any key to continue to evaluate the AES operation. The application prints the AES encryption/decryption menu again on Tera Term program.
8. Press '2' to perform AES CCM decryption using User Cryptoprocessor. The application prompts to enter 128-bit key.
9. Enter the key provided in the preceding test vector and press Enter. The application prompts for Nonce.

Figure 6-7. Application Menu Decryption—Nonce



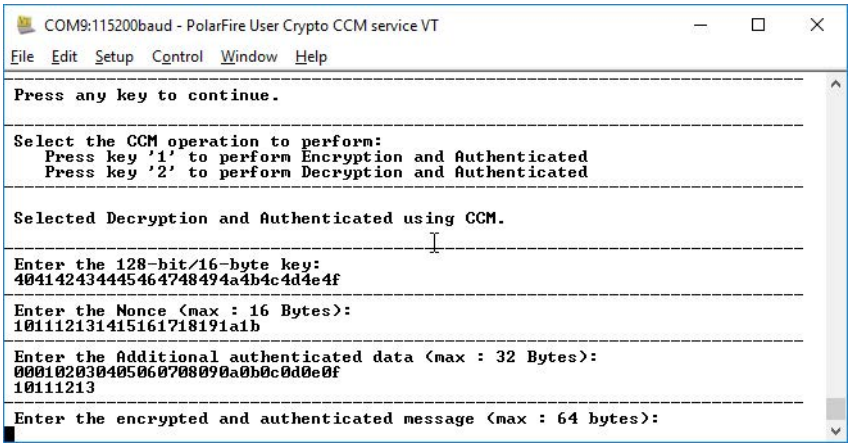
10. Enter the Nonce provided in the test vector and press Enter. The application prompts to enter AAD, as shown in the following figure.

Figure 6-8. Application Menu—Additional Authentication Data



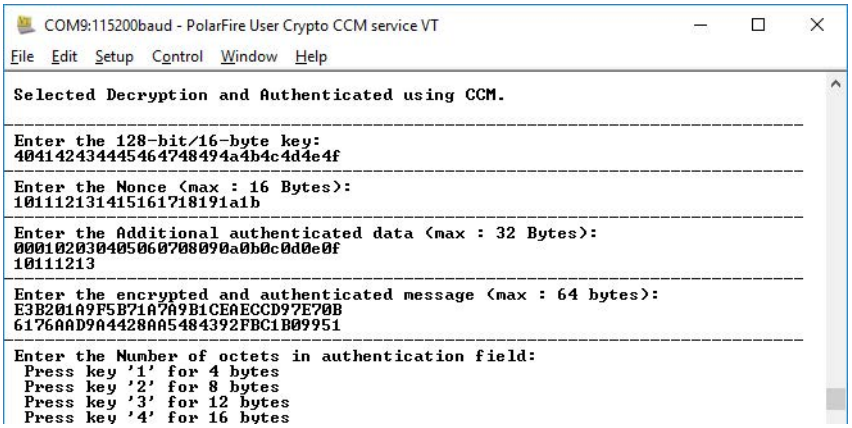
11. Enter the AAD provided in the test vector and press Enter. The application prompts to enter the encrypted and authenticated message, as shown in the following figure.

Figure 6-9. Application Menu—Encrypted and Authenticated Data



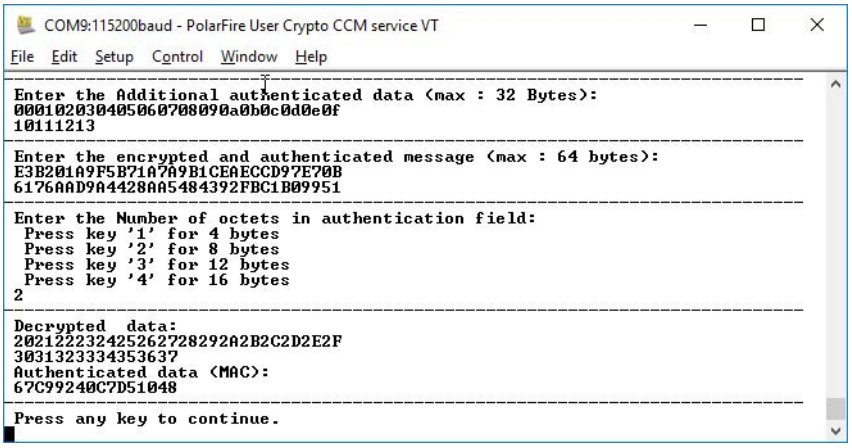
12. Enter the encrypted and authentication data provided in the test vector and press Enter. The application prompts to enter the number of octets in authentication field, as shown in the following figure.

Figure 6-10. Application Menu—Encrypted and Authentication Data



13. Enter '2' to proceed. The result of the AES CCM decryption is printed on the Tera Term program, as shown in the following figure. Observe that the result is matched with the test vector.

Figure 6-11. Application Menu—AES CCM Decryption Result



14. There are other sample firmware projects available on GitHub. The user can download and execute them on this hardware. To run the sample projects, see Appendix 1: Running UserCrypto Sample Projects section.

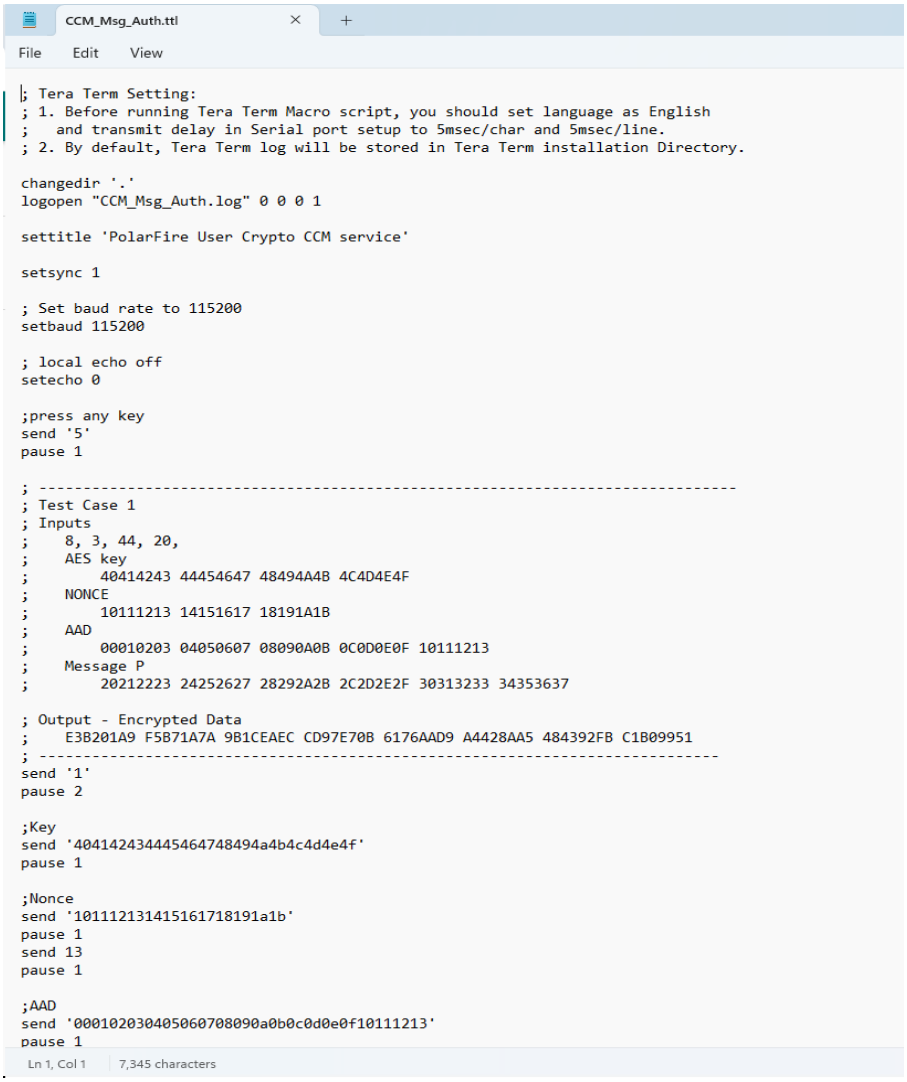
6.1 Running TeraTerm Macro Script (Ask a Question)

To avoid the necessity of entering the input test vectors manually, TeraTerm supports macros which can be used to pass the required inputs during an example project execution. TeraTerm macros use `.ttl` file for this. Each sample project contains a TeraTerm macro script (`.ttl`) file to test the cryptographic algorithms used in the sample project. The TeraTerm macro script has commands to pass required input test vectors. It also contains the expected output for each input test vector for user verification. The following figure shows the TeraTerm macro script in the sample projects provided in the design file.

Important:

- TeraTerm Macros in this design do not work with Windows® 10 build 14393.0. You must update to Windows® 10 build 14393.0.105 or later.
- This is an alternative step to validate the result.

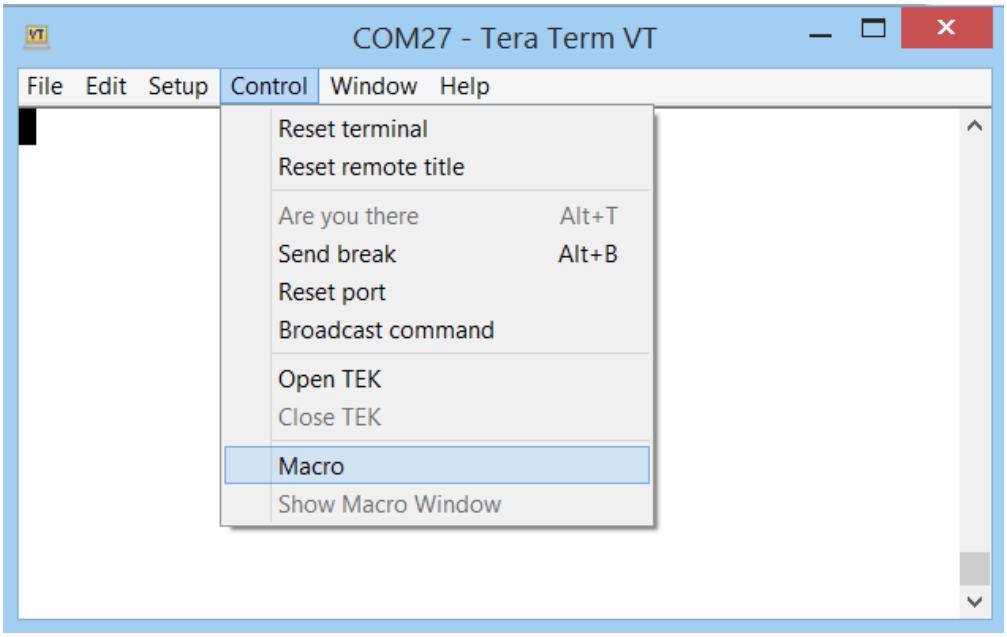
Figure 6-12. TeraTerm Macro Script to Test AES Service



To run the TeraTerm script, perform the following steps:

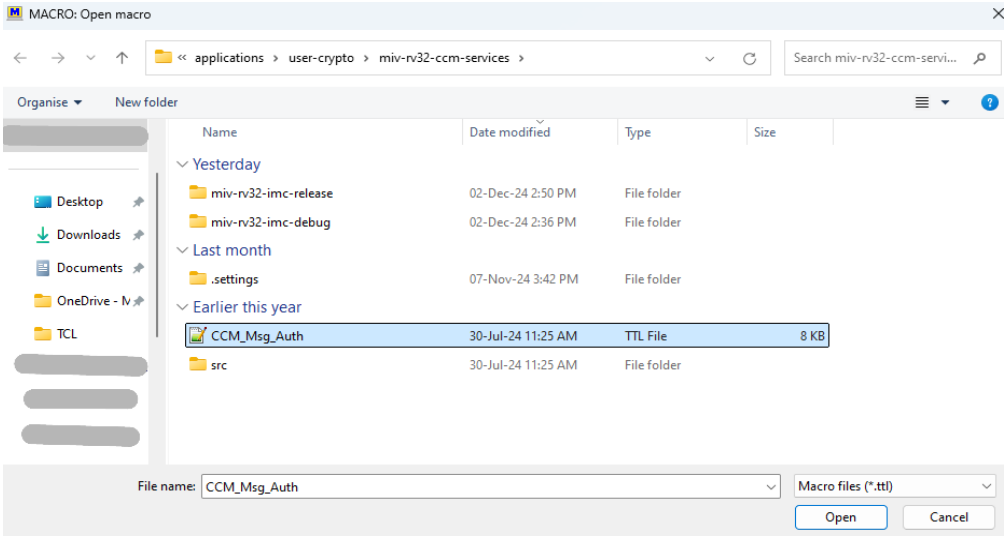
1. On the **Control** menu, click **Macro**, see the following figure.

Figure 6-13. Executing Macro Script



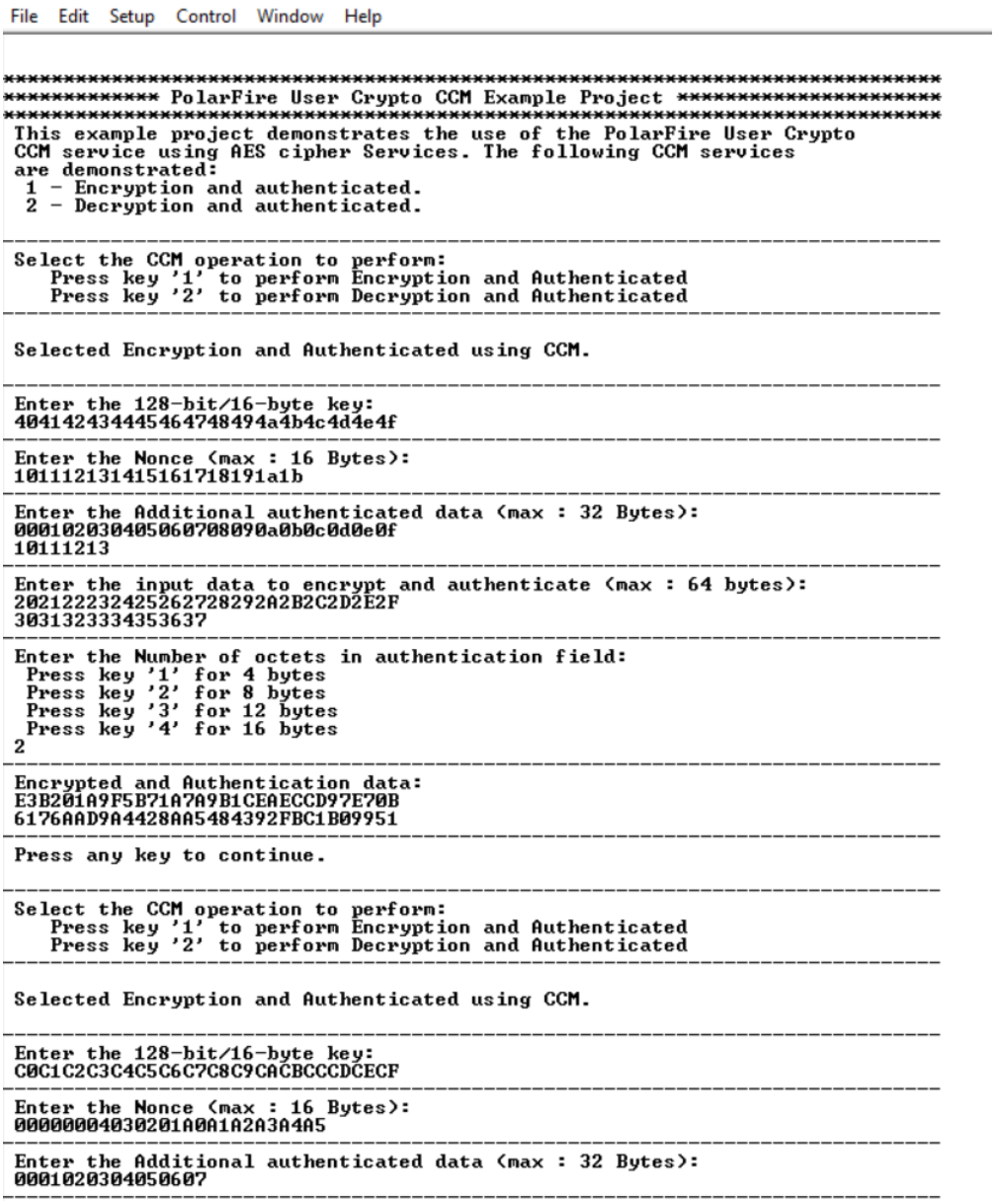
2. In the **Open macro** window, select the `CCM_Msg_Auth.ttl` file available in the sample project provided with the design files and click **Open**, see the following figure.

Figure 6-14. Selecting Macro Script File



TeraTerm executes the script, and the results are printed on the TeraTerm. Compare the output of the AES encryption and decryption service with the test vectors provided in the macro script. The following figure shows the result of the macro script execution.

Figure 6-15. Macro Script Execution Result



This concludes the demo of User Crypto AES service. Terminate the SoftConsole debugger session.

To run other User Crypto sample projects, see Appendix 1: Running UserCrypto Sample Projects.

7. Appendix 1: Running UserCrypto Sample Projects (Ask a Question)

The GitHub contains sample projects to show how to use User Crypto (CAL) APIs in the user application. You can use the sample projects to evaluate various cryptographic features supported by the User Cryptoprocessor on PolarFire devices. This section provides instructions on how to run UserCrypto sample projects with the provided Libero design.

The following table list the UserCrypto algorithms demonstrated in the sample projects. For more information about the sample projects, see `README.txt` file available in the sample projects.

Table 7-1. Algorithm

Algorithm	Parameters and Modes
AES	AES-ECB-256 encrypt
	AES-ECB-256 encrypt
	AES-CCM-128
GMAC	AES-GCM-256, 128-bit tag
HMAC	HMAC-SHA-256, 256-bit key
CMAC	AES-CMAC-256
KEY TREE	128-bit nonce + 8-bit optype
SHA	SHA-256
ECC	ECDSA SigGen, P-384/SHA-384, DPA
IFC (RSA)	Encrypt, RSA-3072, e=65537
	Decrypt, RSA-3072, CRT, DPA
FFC (DH)	SigGen, DSA-3072/SHA-384, DPA
	Key Agreement (KAS), DH-3072
NRBG	Instantiate: strength, s = 256, 384-bit nonce, 384-bit personalization string
	Generate: (no add input, no prediction resistance) s = 256

Important: CCM is used to assure the confidentiality and authenticity of computer data by combining the Counter (CTR) mode techniques and the Cipher Block Chaining-Message Authentication Code (CBC-MAC) algorithm.

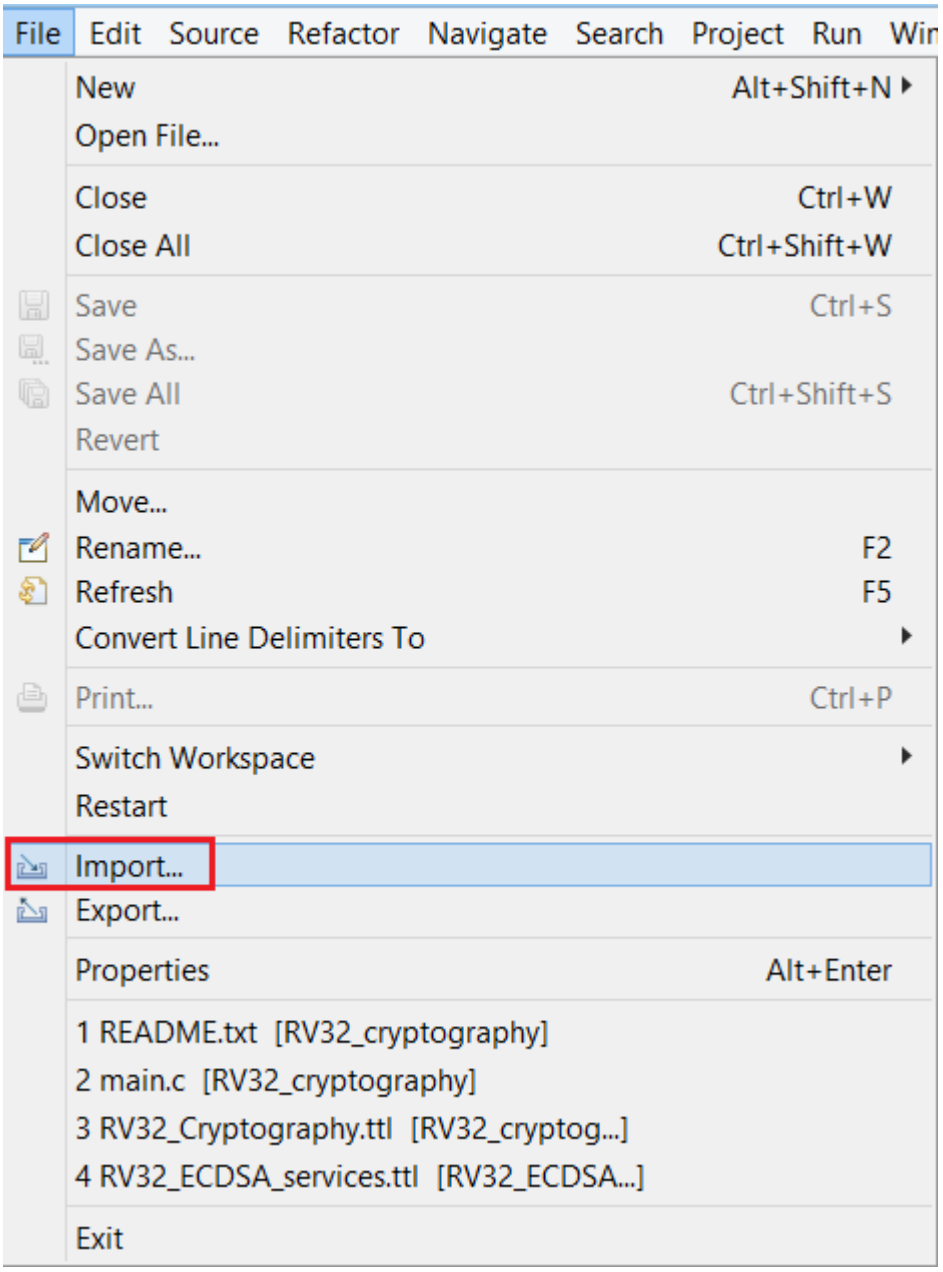
To run the sample projects, perform the following steps:

1. Download the required sample project from GitHub to a required project folder/location.

   Important: CCM Services sample project is provided with the design files. Generate other sample projects and import into the SoftConsole workspace.

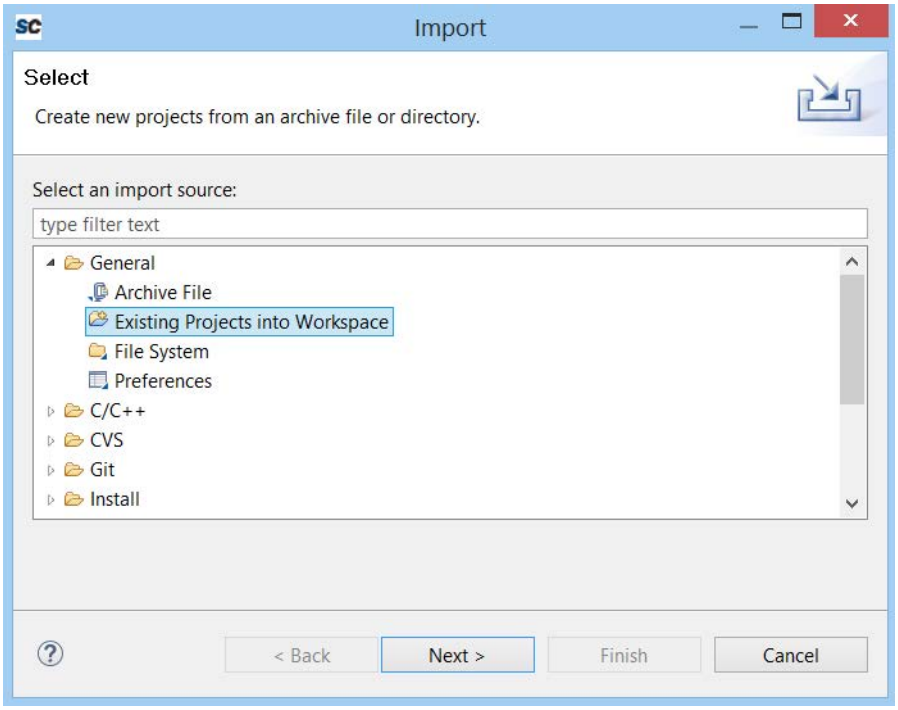
2. To import the sample project into the SoftConsole, navigate to **File** and select **Import**, see the following figure.

Figure 7-1. Import Options



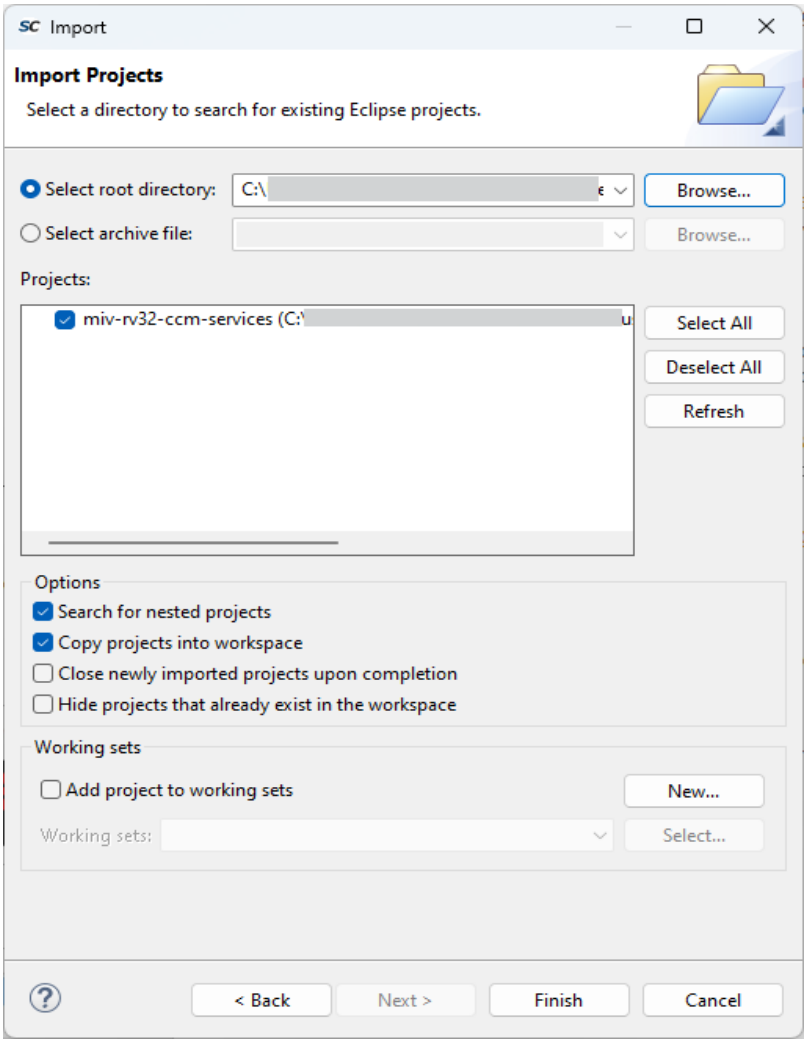
3. Select import source as **Existing Projects into Workspace** and click **Next**, as shown in the following figure.

Figure 7-2. Select An Import Source



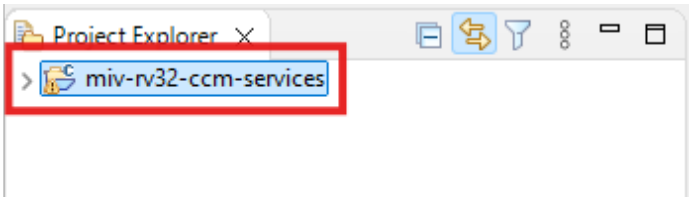
4. In the Import dialog, click **Browse..** to locate the generated sample project in the local PC folder and click **OK**.
5. Ensure that the sample project is selected and click **Finish** to import the generated sample project in a SoftConsole workspace, as shown in the following figure.

Figure 7-3. Importing RV32_Message_Authentication Project



6. The new sample project is imported in the SoftConsole workspace, as shown in the following figure.

Figure 7-4. SoftConsole Workspace—Sample Projects



7. See Software Implementation to make necessary changes to the imported sample project.
8. After making necessary changes, right-click on the imported sample project and click **Build Project** to build the project.
9. Start the SoftConsole debugger to run the project. See Running TeraTerm Macro Script for running the macro script provided in the sample project.

8. Appendix 2: User Cryptoprocessor Simulation (Ask a Question)

Microchip Libero SoC provides a PLI simulation library for the User Cryptoprocessor to show the functional behavior of the User Cryptoprocessor. The user Cryptoprocessor Simulation library is a model that depicts the user Cryptoprocessor's functional behavior, and it is not cycle accurate as real hardware.

The PLI library for User Cryptoprocessor is available at `<Libero_Installation_Directory>/Designer/lib/modelsimpro/pli`. The PLI library must be passed to the ModelSim, using the VSIM command, for User Cryptoprocessor simulation. The VSIM command can be set in Libero Project settings under Simulation options.

- VSIM command for Windows:

```
-pli <$Libero_Installation_Directory>/Designer/lib/modelsimpro/pli/pf_crypto_win_me_pli.dll
```

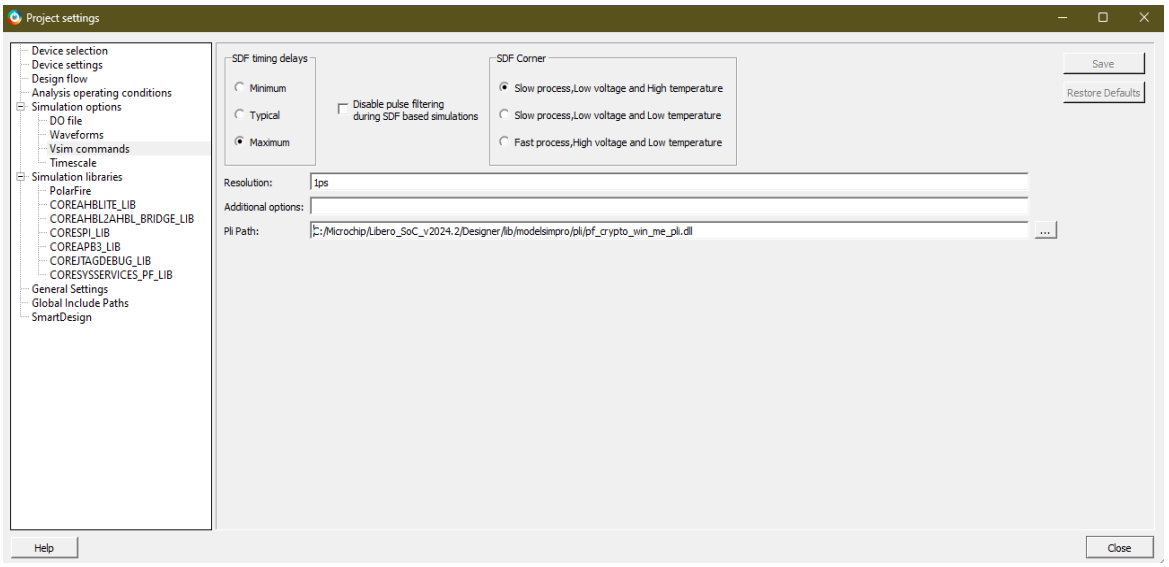
- VSIM command for Linux®:

```
-pli <$Libero_Installation_Directory>/Designer/lib/modelsimpro/pli/pf_crypto_lin_me_pli.so
```

Edit `<Libero_Installation_Directory>` to match the location of Libero SoC on the host PC.

The Libero installation folder is `C:/Microchip/Libero_SoC_v2024.2`, as shown in the following figure.

Figure 8-1. Libero Project settings—VSIM Command for User Cryptoprocessor Simulation



The simulation steps are as follows:

1. Generate the top-level component which includes the Mi-V processor system with PF_CRYPTO core in it.
2. Build a Mi-V application with required User Cryptoprocessor functions. User Cryptoprocessor functions are accessible through Athena TeraFire CAL driver. Subsequently, create an application image or hex file and import it into the system memory (LSRAM).
3. Create a testbench for the complete processor system.
4. To execute the imported application image, simulate the complete processor system. You can observe that the Mi-V processor sends the commands and data to the User Cryptoprocessor, and the User Cryptoprocessor responds with the result.

Important: The preceding reference design uses UART interface as user interface to supply the test vectors and observe the results. To simulate the design, customers can integrate an UART TX and RX testbench with the given test vectors, by following the preceding steps.

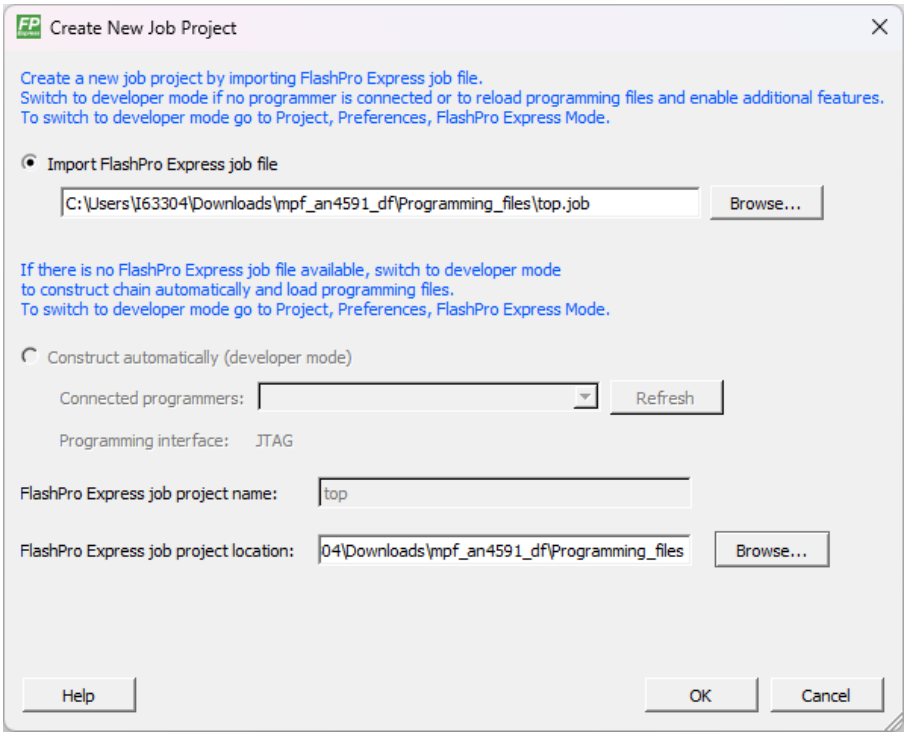
9. Appendix 3: Programming the Device Using FlashPro Express (Ask a Question)

This section describes how to program the PolarFire device with the `.job` programming file, using FlashPro Express. The `.job` file is available at the following design files folder location: `mpf_an4591_df\Programming_Job`.

To program the device, perform the following steps:

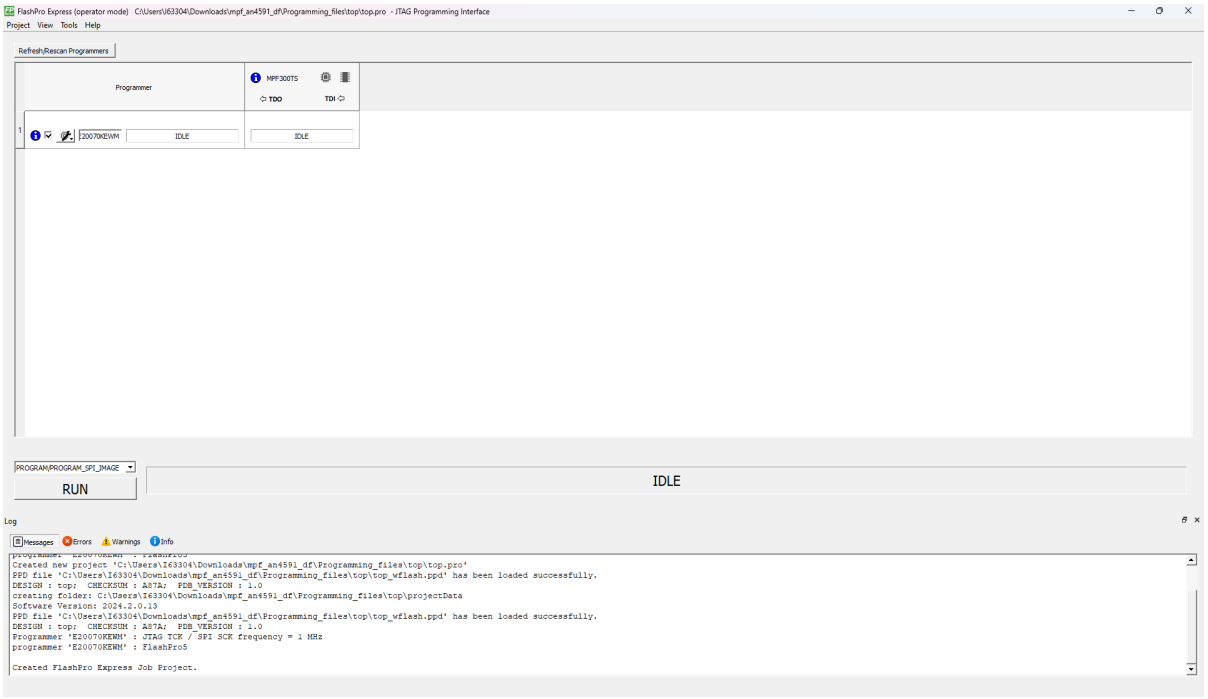
1. On the host PC, launch the FlashPro Express software.
2. To create a new job, click **New** or select **New Job Project from FlashPro Express Job** in the **Project** menu.
3. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:
   - **Programming job file**: Click **Browse**, navigate to the location where the `.job` file is located, and select the file. The default location is: `<download_folder>\mpf_an4591_df\Programming_Job`.
   - **FlashPro Express job project location**: Click **Browse** and navigate to the location where you want to save the project.

Figure 9-1. New Job Project from FlashPro Express Job



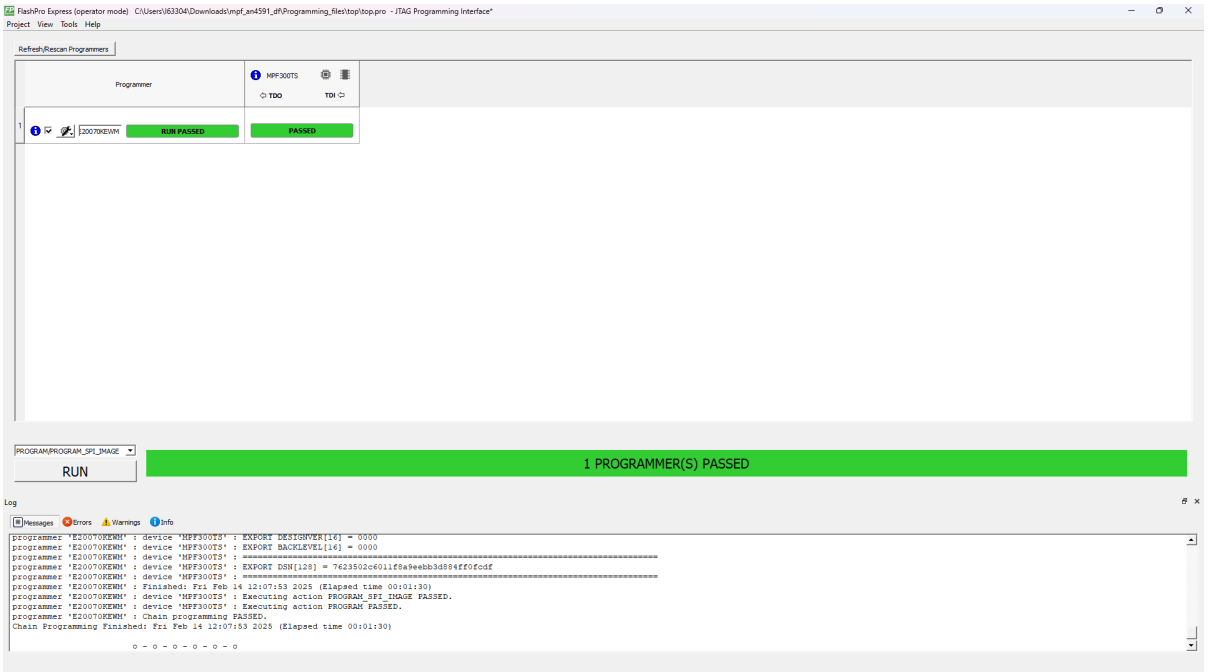
4. Click **OK**. The required programming file is selected and ready to be programmed in the device.
5. The **FlashPro Express** window appears, as shown in the following figure. Confirm that a programmer number appears in the **Programmer** field. If it does not, confirm the board connections and click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

Figure 9-2. Programming the Device



6. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed, as shown in the following figure.

Figure 9-3. FlashPro Express—RUN PASSED



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

10. 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 regenerated from Design Implementation till generation of job file.

To run the TCL, perform the following steps:

1. Launch the Libero software.
2. Select **Project** > **Execute Script…**.
3. Click **Browse** and select `script.tcl` from the downloaded `HW` directory.
4. Click **Run**.

After successful execution of TCL script, Libero project is created within `HW` directory. For more information about TCL scripts, see `mpf_an4591_df/HW/TCL_Script_readme.txt`.

For more details on TCL commands, see Tcl Commands Reference Guide. Contact Technical Support for queries encountered when running the TCL script.

11. 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 most current publication.

Table 11-1. Revision History

Revision	Date	Description
D	03/2025	Added Appendix 3: Programming the Device Using FlashPro Express section.
C	02/2025	The following is the list of changes in revision C of the document: • Updated the document for Libero v2024.2. • Updated the `.job` filepath and TCL script filepath throughout the document. • Updated Table 2-1 in the Design Requirements section. • Updated Figure 4-1 in the Design Description section. • Updated Figure 4-2 in the Clocking Structure section. • Added Reset Structure section. • Updated Hardware Implementation section as follows: – Added MIV_ESS to Table 4-1. – Added CoreGPIO_C0_0 and CORESPI_C0_0 to the Table 4-2. – Updated Figure 4-5. • Updated Figure 4-6 in the FPGA Resource Utilization section. • Updated Figure 4-7, Figure 4-9 and Figure 4-10 in the Software Implementation section. • Updated Figure 4-11 and Figure 4-12 in the Linker Script Update section. • Updated Figure 8-1 in the Appendix 2: User Cryptoprocessor Simulation section.
B	04/2023	In User Cryptoprocessor updated the text from "RSA, DSA, and modular exponentiation (Diffie-Hellman) with key sizes up to 4096-bits" to "RSA, DSA, and modular exponentiation (Diffie-Hellman) with key sizes up to 3072-bits".
A	08/2022	The following is the list of changes in revision A of the document: • The document was migrated to the Microchip template. • The document number was updated from 51900464 to DS00004591. • The document ID was updated from AC464 to AN4591. • Updated Appendix 2: User Cryptoprocessor Simulation. • Updated LSRAM Initialization from SPI Flash .
11.0	—	The following is a summary of the changes made in this revision. • Updated LSRAM Initialization from SPI Flash. • Updated Clocking Structure. • Updated Figure 4-2. • Updated Hardware Implementation. • Updated Table 4-2. • Updated Figure 4-5. • Updated Figure 4-6. • Updated Figure 4-15. • Updated Figure 4-16. • Updated Appendix 2: User Cryptoprocessor Simulation. • Updated Figure 8-1. • Updated Appendix 2: User Cryptoprocessor Simulation.
10.0	—	Added Appendix 4: Running the TCL Script.

Table 11-1. Revision History (continued)

Revision	Date	Description
9.0	—	In this revision, information about Prerequisites was updated.
8.0	—	The following is a summary of the changes made in this revision. • Updated the document for Libero SoC v12.2. • Removed the references to Libero version numbers. • Added information about enabling protection for SPI Flash client. For more information, refer to LSRAM Initialization from SPI Flash.
7.0	—	The following is a summary of the changes made in this revision. • Information about how to unlock the sNVM pages was added. See Adding a Dummy Client to Unlock the sNVM Pages. • Updated the document for Libero SoC v12.1.
6.0	—	Updated the document for Libero SoC v12.0.
5.0	—	The document was updated for Libero SoC PolarFire v2.3 release.
4.0	—	The document was updated for Libero SoC PolarFire v2.2 release.
3.0	—	The document was updated for Libero SoC PolarFire v2.1 release.
2.0	—	The following is a summary of the changes made in this revision. • Information about initializing the SRAM component with the HEX file was added, see LSRAM Initialization from SPI Flash. • The document was updated to include features and enhancements introduced in the Libero SoC PolarFire v2.0 release.
1.0	—	Initial release.

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: 979-8-3371-0921-3

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.

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.