AC473
Application Note
PolarFire FPGA: Implementing Data Security Using User Cryptoprocessor - Splash Kit

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

51900473. 2.0 7/18

Contents

Figures

Tables

1 Revision History

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

1.1 Revision 2.0

The document was updated for Libero SoC PolarFire v2.2 release.

1.2 Revision 1.0

The first publication of this document.

2 Implementing Data Security Using User Cryptoprocessor

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, referred as User Cryptoprocessor, for data security applications.

This application note describes User Cryptoprocessor features, 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.

2.1 User Cryptoprocessor

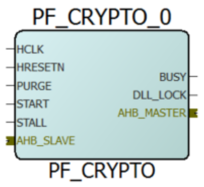
The User Cryptoprocessor is an Athena TeraFire® EXP-F5200B cryptography microprocessor. It provides complete support for the Commercial National Security Algorithm (CNSA) Suite1 and beyond, and includes side-channel analysis (SCA) resistant cryptography using patented leakage reduction countermeasures. 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-, 192-, 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 4096-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[1]. It is accessible to a soft processor in the FPGA fabric via 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 doing data transfers between the User Cryptoprocessor and the user memory. The DMA functionality is accessible via an AMBA AHB-Lite master interface. The Libero SoC PolarFire 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 • **PF_CRYPTO Macro**



1.Enable **Fabric Clock (HCLK) Frequency is less than 100 MHz** in PF_CRYPTO macro configurator if you want to operate the user cryptoprocessor at a frequency less than 100 MHz.

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 via the dynamic reconfiguration interface (DRI). Contact Microsemi tech-support for information on how to use DRI.

Table 1 • **PF_CRYPTO Port List**

Port Name	Direction	Description
AHB_SLAVE		AHB-Lite slave interface.
AHB_MASTER		AHB-Lite master interface.
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 standalone configuration without a host processor connected to the bus interface. Asserting 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 a 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		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.
ALARM		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		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.

Microsemi provides an Athena TeraFire cryptographic applications library (CAL) to access User Cryptoprocessor functions. TeraFire CAL is a C language library that provides functions to access symmetric key, elliptic curve, public key, hash, random number generation, and message authentication code algorithms. It is available for download in the Microsemi Firmware Catalog software. The user application running on the main processor must include CAL APIs to perform the cryptographic operations on the User Cryptoprocessor.

For information about User Cryptoprocessor, supported cryptographic functions and their CAL API descriptions, see *Athena TeraFire Cryptographic Algorithm Library (CAL) Users Guide*.

2.2 Design Requirements

The following table lists the hardware, software, and IPs required to create a Libero and SoftConsole project to perform cryptographic operations using the User Cryptoprocessor.

Table 2 • **Resource Requirements**

Requirement	**Version**
Host PC Operating system	Windows 7, 8.1, or 10
Hardware	
PolarFire Splash Kit (MPF300TS-1FCG484EES) – PolarFire Splash Board – 12 V/5 A wall-mounted power adapter – USB 2.0 A-male to mini-B cable	Rev 2 or later
Software	
Libero SoC PolarFire[1]	v2.2
SoftConsole	5.2
Tera Term	4.95
IP Cores and HDL Source Files	
MIV_RV32IMA_L1_AHB	2.0.100
CoreJTAGDEBUG	2.0.100
PF_INIT_Monitor	2.0.103
CoreAHBL2AHBL_Bridge	2.0.108
CoreAHBtoAPB3	3.1.100
CoreUARTapb	5.6.102
PF_SRAM_AHBL_AXI	1.1.125
CoreAHBLite	5.3.101
CoreAPB3	4.1.100
PF_CCC	1.0.113
PF_CRYPTO	1.0.105
reset_synchronizer.v	NA
CoreSysServices_PF	2.3.116

1. You need to have Evaluation, Gold, Platinum, or Standalone license to use PolarFire ‘S’ grade devices.

2.3 Design Files

Download the design files from the following location:
http://soc.microsemi.com/download/rsc/?f=mpf_ac473_liberosocpolarfirev2p2_df

2.4 Design Description

Microsemi offers a freely available RISC-V processor IP core called Mi-V and 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 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.

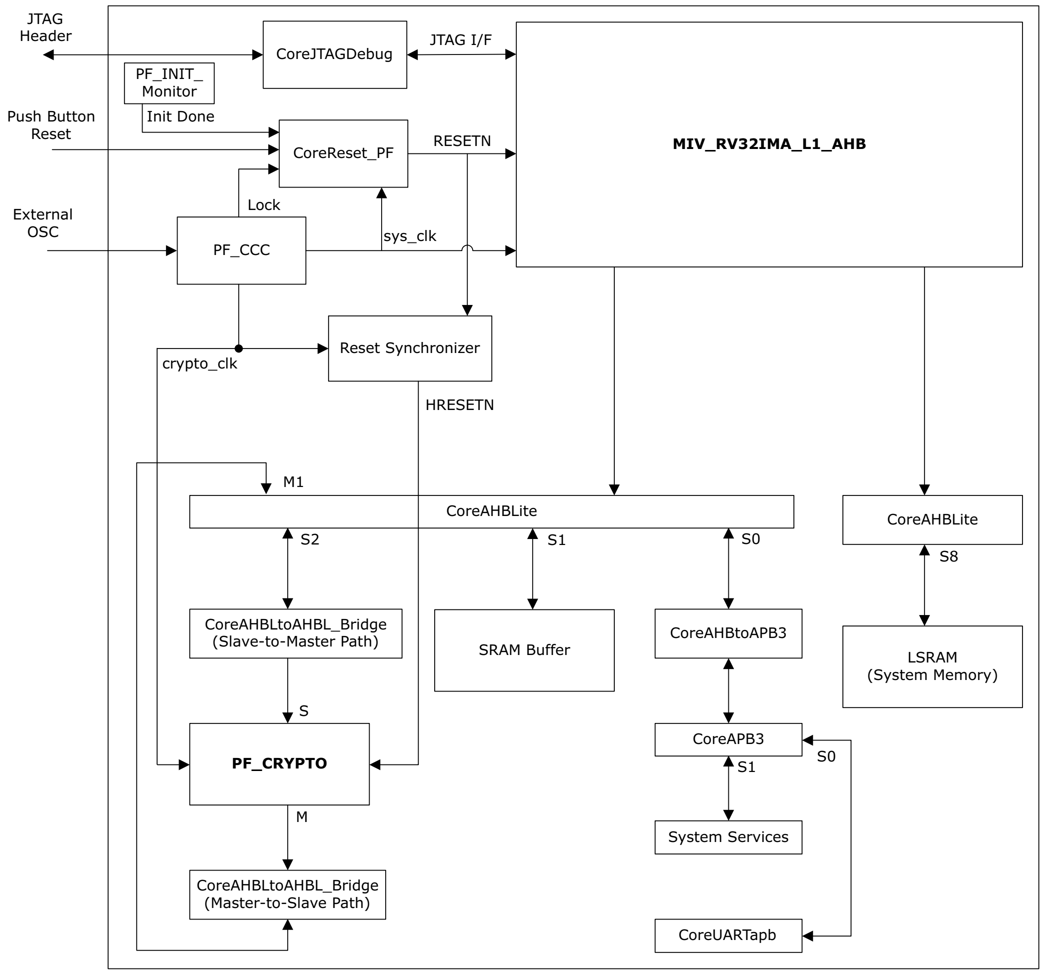
The Libero design provided with this application note shows how to integrate the User Cryptoprocessor in a Mi-V processor subsystem. The SoftConsole project shows how to integrate and build TeraFire CAL driver into a Mi-V processor application project. A similar process can be used to integrate the User 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 Firmware catalog to explore using the User Cryptoprocessor cryptographic algorithms.

Each PolarFire FPGA has 56 Kbytes of secure non-volatile memory (sNVM), which can be used for storing cryptographic keys. The sNVM pages are accessible through system services for read/write. The reference design integrates CoreSysServices_PF IP for sNVM read/write.

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

Figure 2 • **Reference Design Block Diagram**



2.4.1 Clocking Structure

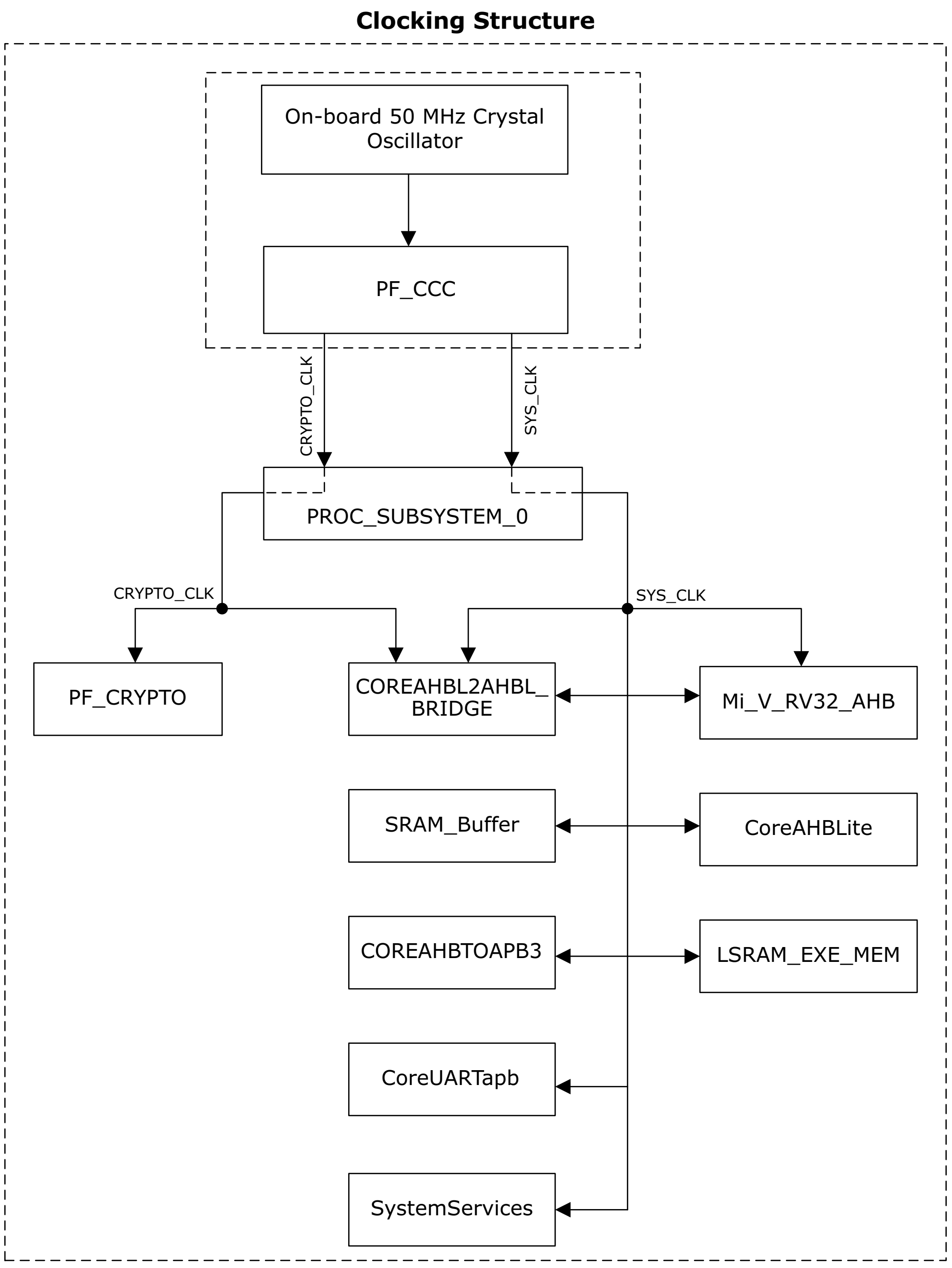
In the reference design, there is one clock domain. The on-board 50 MHz crystal oscillator is connected to the PF_CCC block which generates 100 MHz and 180 MHz clocks.

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

The 100 MHz system clock is connected to COREAHBL2AHBL_BRIDGE, Mi_V_RV32_AHB, SRAM_Buffer, CoreAHBLite, COREAHBTOAPB3, LSRAM_EXE_MEM, System Services, and CoreUARTapb blocks.

The following figure shows the clocking structure.

Figure 3 • **Clocking Structure**



2.4.2 Hardware Implementation

To build a Mi-V subsystem for PolarFire FPGAs, use the Libero SoC PolarFire 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 3 • **IP Cores Used in the Reference Design**

IP Core	Description
MIV_RV32IMA_L1_AHB	Mi-V soft processor
CoreJTAGDEBUG	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 UJTAG macro.
PF_INIT_monitor	System Controller uses this macro to check the status of device initialization. The device initialization includes SRAM initialization from µPROM/sNVM/SPI Flash. The DEVICE_INIT_DONE signal is used as reset.
CoreAHBLite	Multi-master AHB-Lite bus
COREAHBL2AHBL_BRIDGE	AHB to AHB bridge connect two AHB-Lite buses clocked by asynchronous clocks. This is required because the User Cryptoprocessor clock is different than 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.
CoreAHBtoAPB3	Bridge between AHB master and APB slave.
CoreUARTapb	UART Controller with APB interface
PF_CCC	Macro to access PolarFire CCC block. It is used to synthesize 100 MHz and 180 MHz clock frequencies from the CCC with an on-board 50 MHz reference clock.
reset_synchronizer.v	Reset synchronizer is used to synchronize the reset with system clock.
CoreSysServices_PF	The CoreSysServices_PF provides access to the System Services supported by PolarFire device. These are System Controller actions initiated from the user design using the CoreSysServices.

Configure and connect the IP cores listed in Table 3, page 7, then run the Libero design flow to create a programming file. For more information about how to build a Mi-V processor subsystem for PolarFire devices, see *TU0775: PolarFire FPGA: Building a Mi-V Processor Subsystem Tutorial*. This Mi-V processor subsystem can be extended to integrate a User Cryptoprocessor into the system.

The MIV_RV32IMA_L1_AHB processor core comprises of an instruction fetch unit, an execution pipeline, and a data memory system. The MIV_RV32IMA_L1_AHB processor memory system includes instruction cache and data cache. The MIV_RV32IMA_L1_AHB core includes two external AHB interfaces—the AHB memory (MEM) bus master interface and the AHB memory mapped I/O (MMIO) bus master interface. The cache controller uses AHB MEM interface to refill the instructions and the data caches. The AHB MMIO interface is used for an un-cached accesses to I/O peripherals.

The memory maps of the AHB MMIO interface and the MEM interface are 0x60000000 to 0x7F000000 and 0x80000000 to 0x8FFFFFFF, respectively. The MIV_RV32IMA_L1_AHB processor's reset vector address is set to 0x80000000. The processor’s reset vector address is configurable. The MIV_RV32IMA_L1_AHB's reset is an active-low signal, which must be de-asserted in sync with the system clock via a reset synchronizer. For more information about the MIV_RV32IMA_L1_AHB core, see *HB0801: MiV_RV32IMAF_L1_AHB V2.0 Handbook* from Libero Catalog.

The MIV_RV32IMA_L1_AHB processor accesses the application execution memory using the AHB MEM interface. The CoreAHBLite_1 bus instance is configured to provide 2 GB memory slot (Slot 16) beginning at address 0x80000000. The PolarFire LSRAM memory block (LSRAM_EXE_MEM_1 instance) is connected to this slot and the LSRAM acts as an application execution memory for the Mi-V processor.

The MIV_RV32IMA_L1_AHB processor directs the data transactions between addresses 0x60000000 and 0x7F000000 to the MMIO interface. The MMIO 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 20-bit address bus. The CoreAHBLite_0 bus, using only the lower 20-bits of the MMIO bus address, and maps the connected peripherals within the address range. The User Cryptoprocessor, UART peripheral, System Services, and LSRAM memory are connected to the CoreAHBLite_0 bus.

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

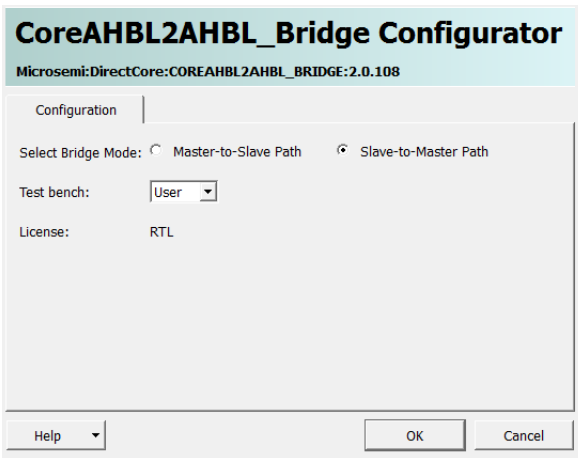
Table 4 • **System Memory Map**

Component	Description	Memory Map
PF_CRYPTO_0	User Cryptoprocessor	0x62000000 to 0x6200FFFF
SystemServices_0	System Services	0x60001000 to 0x60001FFF
CoreUARTapb_0	UART peripheral	0x60000000 to 0x60000FFF
RAM_Buffer_0	Memory buffer for User Cryptoprocessor	0x61000000 to 0x61FFFFFF
LSRAM_EXE_MEM_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 100 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. This IP can be configured by setting the **Select bridge Mode** option to either **Master-to-Slave Path** or **Slave-to-Master Path**.

Figure 4 • **CoreAHBL2AHBL_Bridge Configurator**


CoreAHBL2AHBL_Bridge Configurator

Microsemi:DirectCore:COREAHBL2AHBL_BRIDGE:2.0.108

Configuration

Select Bridge Mode: Master-to-Slave Path [✓ Slave-to-Master Path]

Test bench: User

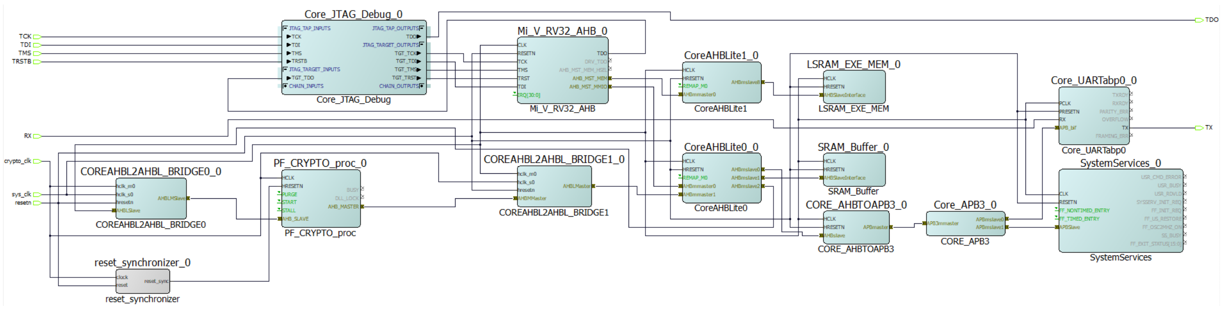
License: RTL

The CoreAHBL2AHBL_Bridge_0 is configured in the Slave-to-Master path to connect the User Cryptoprocessor to the Mi-V processors peripheral MMIO bus for control and primary data input and output. In this configuration, the sys_clk must be connected to the bridges 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 in the Master-to-Slave path 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 bridges master interface clock (hclk_m0) and the crypto_clk must be connected to the bridges slave interface clock (hclk_s0).

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

Figure 5 • **Mi-V Processor Subsystem with User Cryptoprocessor**

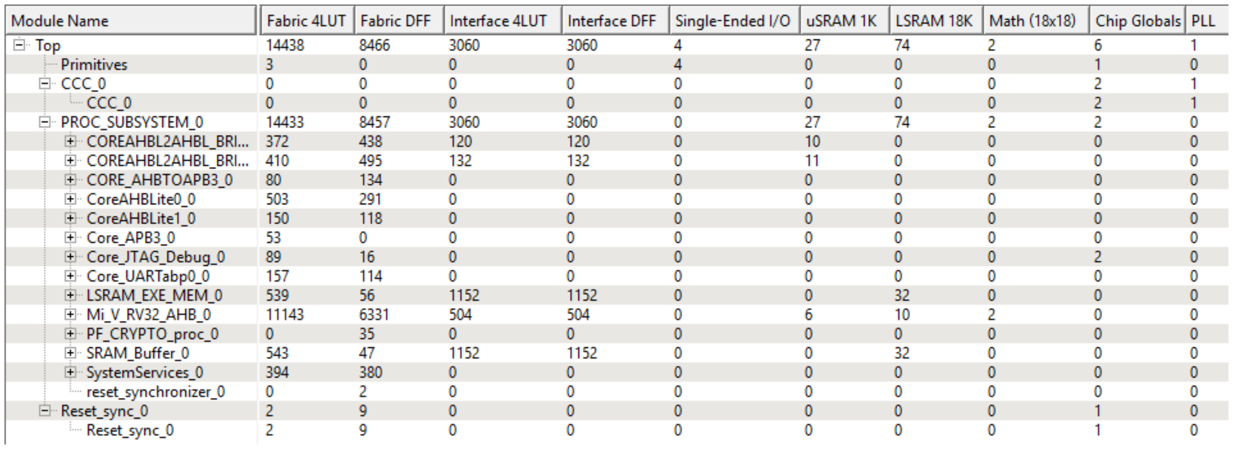


See the provided Libero project for more details on component configurations and connections.

2.4.2.1 FPGA Resource Utilization

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

Figure 6 • **Design Resource Utilization**



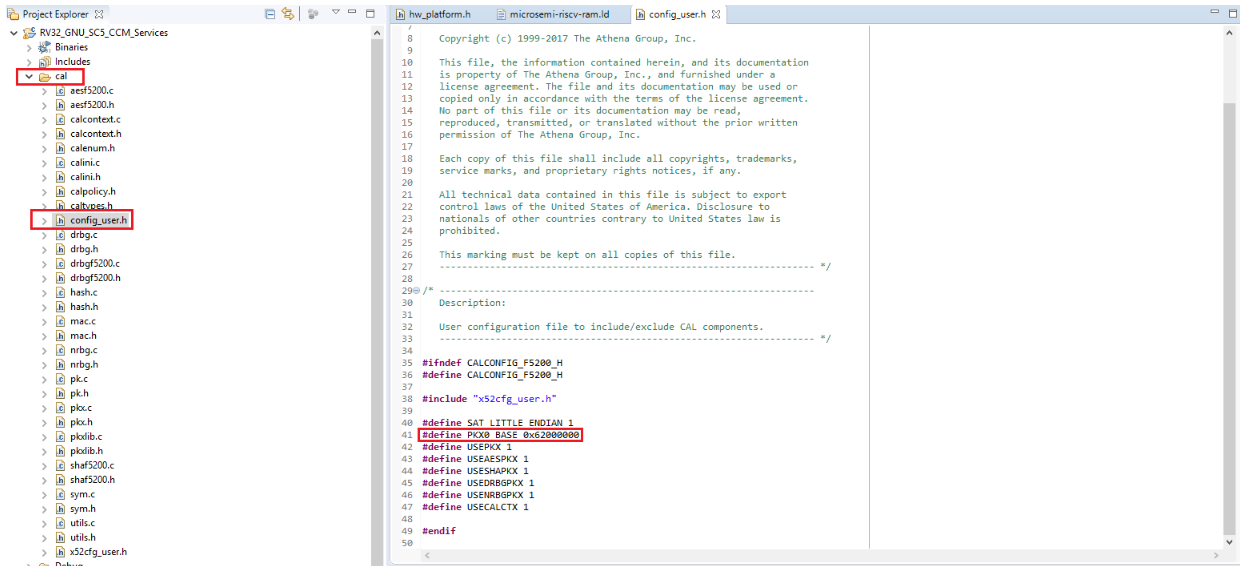
Module Name	Fabric 4LUT	Fabric DFF	Interface 4LUT	Interface DFF	Single-Ended I/O	uSRAM 1K	LSRAM 18K	Math (18x18)	Chip Globals	PLL
⊟ Top	14438	8466	3060	3060	4	27	74	2	6	1
Primitives	3	0	0	0	4	0	0	0	1	0
⊟ CCC_0	0	0	0	0	0	0	0	0	2	1
CCC_0	0	0	0	0	0	0	0	0	2	1
⊟ PROC_SUBSYSTEM_0	14433	8457	3060	3060	0	27	74	2	2	0
⊞ COREAHBL2AHBL_BRI...	372	438	120	120	0	10	0	0	0	0
⊞ COREAHBL2AHBL_BRI...	410	495	132	132	0	11	0	0	0	0
⊞ CORE_AHBTOAPB3_0	80	134	0	0	0	0	0	0	0	0
⊞ CoreAHBLite0_0	503	291	0	0	0	0	0	0	0	0
⊞ CoreAHBLite1_0	150	118	0	0	0	0	0	0	0	0
⊞ Core_APB3_0	53	0	0	0	0	0	0	0	0	0
⊞ Core_JTAG_Debug_0	89	16	0	0	0	0	0	0	2	0
⊞ Core_UARTapb0_0	157	114	0	0	0	0	0	0	0	0
⊞ LSRAM_EXE_MEM_0	539	56	1152	1152	0	0	32	0	0	0
⊞ Mi_V_RV32_AHB_0	11143	6331	504	504	0	6	10	2	0	0
⊞ PF_CRYPTO_proc_0	0	35	0	0	0	0	0	0	0	0
⊞ SRAM_Buffer_0	543	47	1152	1152	0	0	32	0	0	0
⊞ SystemServices_0	394	380	0	0	0	0	0	0	0	0
reset_synchronizer_0	0	2	0	0	0	0	0	0	0	0
⊟ Reset_sync_0	2	9	0	0	0	0	0	0	1	0
Reset_sync_0	2	9	0	0	0	0	0	0	1	0

2.4.3 Software Implementation

SoftConsole v5.2 is used to build the user application. For more information about how to create and build a SoftConsole project for the Mi-V processor subsystem, see *TU0775: PolarFire FPGA: Building a Mi-V Processor Subsystem Tutorial*.

To use User Cryptoprocessor services in the application, add the PolarFire User Crypto CAL driver to the SoftConsole project. The User Crypto CAL driver 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.

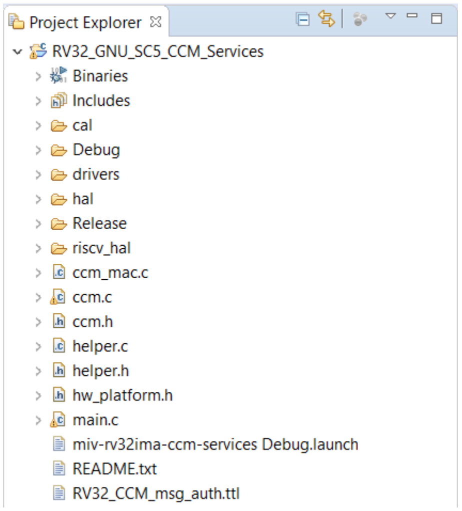
Figure 7 • User Crypto CAL Driver



```
   Copyright (c) 1999-2017 The Athena Group, Inc.

   This file, the information contained herein, and its documentation
   is property of The Athena Group, Inc., and furnished under a
   license agreement. The file and its documentation may be used or
   copied only in accordance with the terms of the license agreement.
   No part of this file or its documentation may be read,
   reproduced, transmitted, or translated without the prior written
   permission of The Athena Group, Inc.

   Each copy of this file shall include all copyrights, trademarks,
   service marks, and proprietary rights notices, if any.

   All technical data contained in this file is subject to export
   control laws of the United States of America. Disclosure to
   nationals of other countries contrary to United States law is
   prohibited.

   This marking must be kept on all copies of this file.
   ------------------------------------------------------------------- */

/* -------------------------------------------------------------------
   Description:

   User configuration file to include/exclude CAL components.
   ------------------------------------------------------------------- */

#ifndef CALCONFIG_F5200_H
#define CALCONFIG_F5200_H

#include "x52cfg_user.h"

#define SAT_LITTLE_ENDIAN 1
#define PKX0_BASE 0x62000000
#define USEPKX 1
#define USEAESPKX 1
#define USESHAPKX 1
#define USEDRBGPKX 1
#define USENRBGPKX 1
#define USECALCTX 1

#endif
```

The following figure shows the intended directory structure for a project based on MIV_RV32IMA_L1_AHB using SoftConsole. This project uses the User Crypto Library (CAL) and CoreUARTapb drivers. CoreUARTapb drivers rely on the RISC-V HAL to access the hardware. The content of the drivers directory result from generating the driver source files for each peripheral in the project using Firmware Catalog. The contents of the HAL and riscv_hal directories result from generating the source files for the RISC-V HAL from the Firmware Catalog.

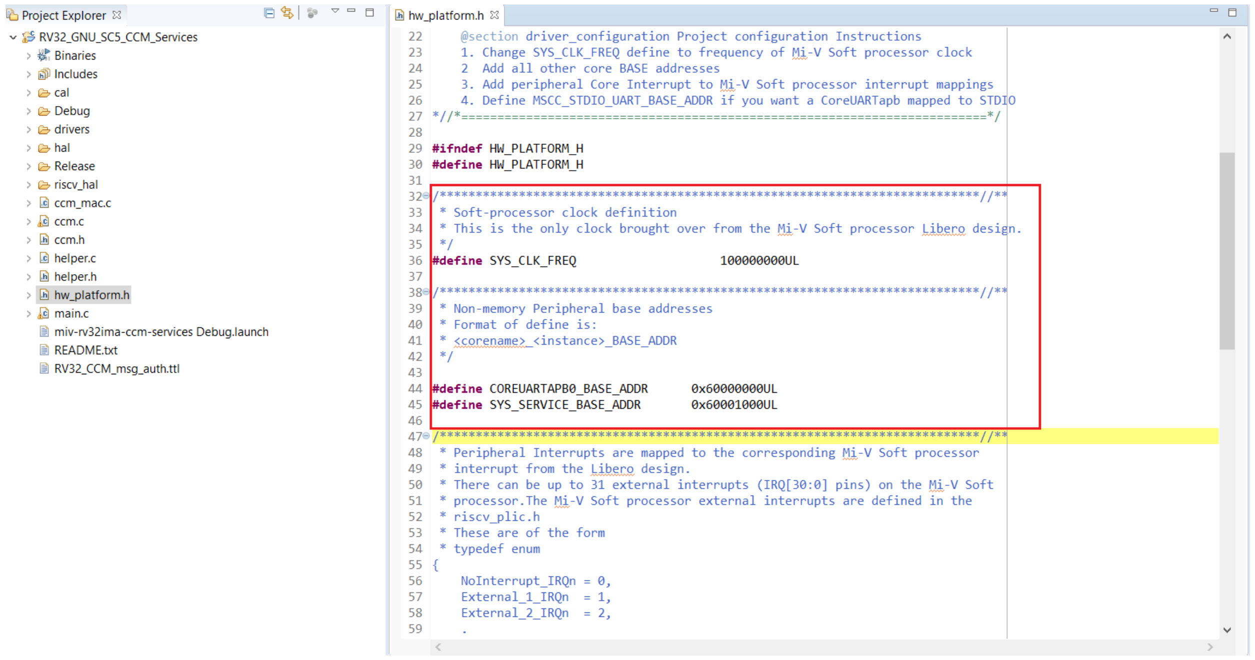
Figure 8 • SoftConsole Project Directory Structure



```
RV32_GNU_SC5_CCM_Services
  Binaries
  Includes
  cal
  Debug
  drivers
  hal
  Release
  riscv_hal
  ccm_mac.c
  ccm.c
  ccm.h
  helper.c
  helper.h
  hw_platform.h
  main.c
  miv-rv32ima-ccm-services Debug.launch
  README.txt
  RV32_CCM_msg_auth.ttl
```

The UART peripheral base address, System Services base address, and system clock frequency are provided in the `hw_platform.h` file.

Figure 9 • **Peripheral Base Address**



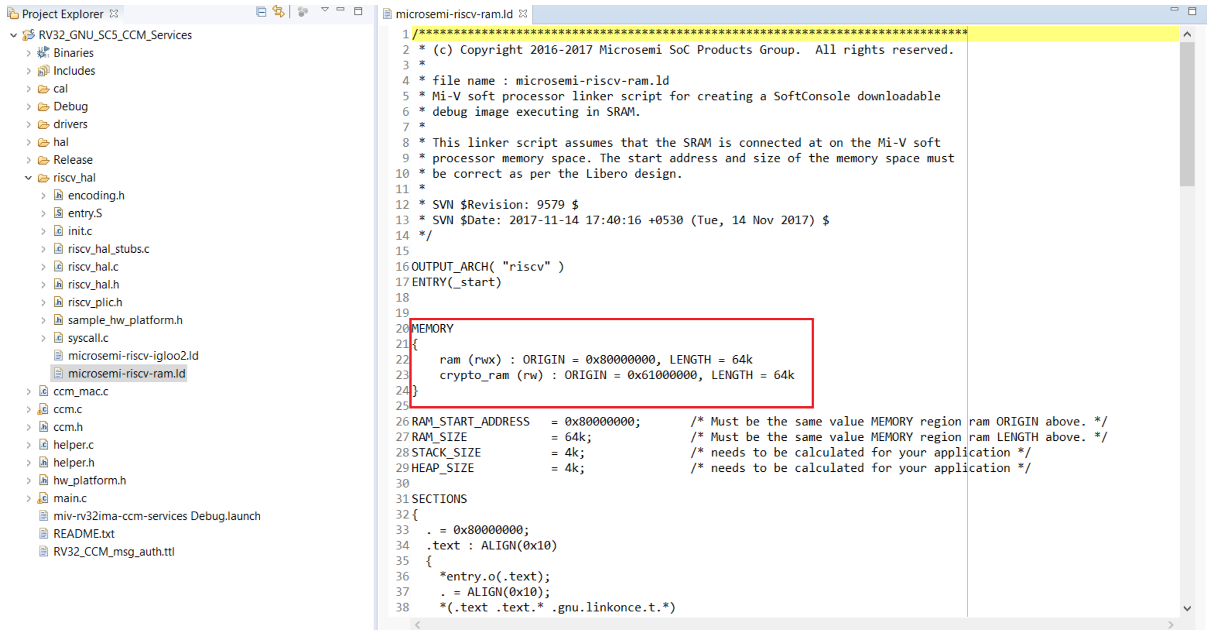
```
    @section driver_configuration Project configuration Instructions
    1. Change SYS_CLK_FREQ define to frequency of Mi-V Soft processor clock
    2  Add all other core BASE addresses
    3. Add peripheral Core Interrupt to Mi-V Soft processor interrupt mappings
    4. Define MSCC_STDIO_UART_BASE_ADDR if you want a CoreUARTapb mapped to STDIO
*//*==================================================================*/

#ifndef HW_PLATFORM_H
#define HW_PLATFORM_H

/***************************************************************************//**
 * Soft-processor clock definition
 * This is the only clock brought over from the Mi-V Soft processor Libero design.
 */
#define SYS_CLK_FREQ                    100000000UL

/***************************************************************************//**
 * Non-memory Peripheral base addresses
 * Format of define is:
 * <corename>_<instance>_BASE_ADDR
 */

#define COREUARTAPB0_BASE_ADDR      0x60000000UL
#define SYS_SERVICE_BASE_ADDR       0x60001000UL

/***************************************************************************//**
 * Peripheral Interrupts are mapped to the corresponding Mi-V Soft processor
 * interrupt from the Libero design.
 * There can be up to 31 external interrupts (IRQ[30:0] pins) on the Mi-V Soft
 * processor.The Mi-V Soft processor external interrupts are defined in the
 * riscv_plic.h
 * These are of the form
 * typedef enum
{
    NoInterrupt_IRQn = 0,
    External_1_IRQn  = 1,
    External_2_IRQn  = 2,
    .
```

2.4.3.1 Linker Script Update

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

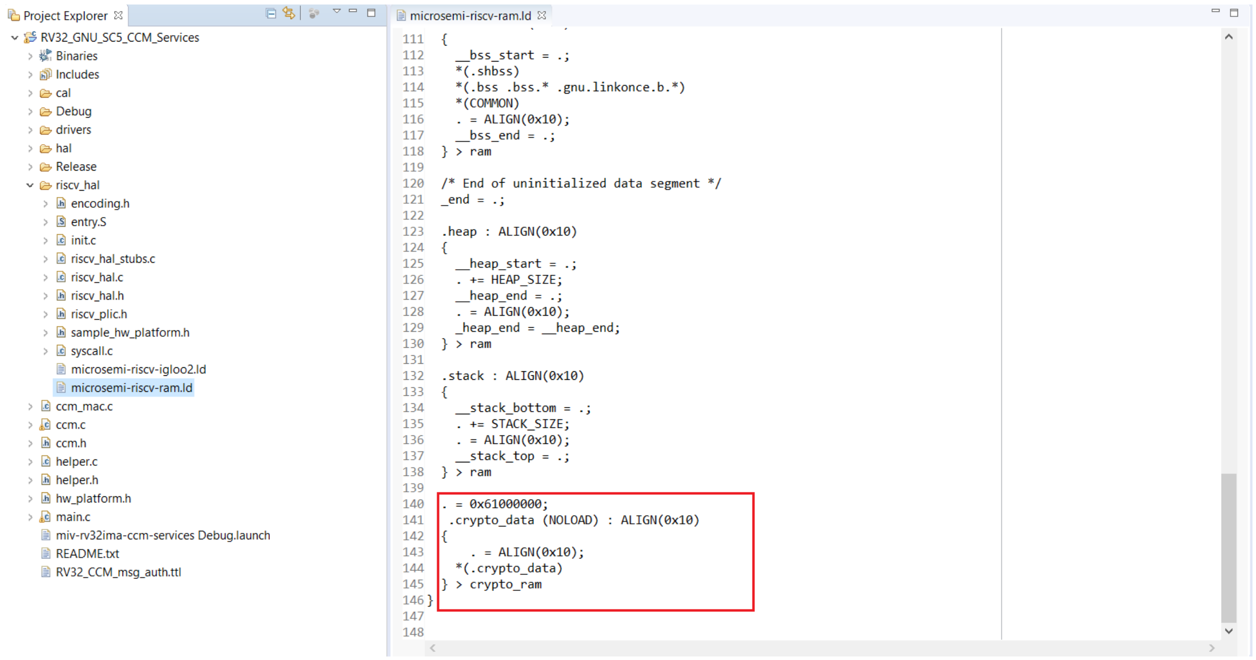
A *crypto_ram* user section is defined in the linker script (see the following figure) to map the User Crypto input and output buffers to a common memory located on the Mi-V processor MMIO interface. The common memory is located at address 0x61000000.

Figure 10 • **Linker Script**



```
/*******************************************************************************
 * (c) Copyright 2016-2017 Microsemi SoC Products Group.  All rights reserved.
 *
 * file name : microsemi-riscv-ram.ld
 * Mi-V soft processor linker script for creating a SoftConsole downloadable
 * debug image executing in SRAM.
 *
 * This linker script assumes that the SRAM is connected at on the Mi-V soft
 * processor memory space. The start address and size of the memory space must
 * be correct as per the Libero design.
 *
 * SVN $Revision: 9579 $
 * SVN $Date: 2017-11-14 17:40:16 +0530 (Tue, 14 Nov 2017) $
 */

OUTPUT_ARCH( "riscv" )
ENTRY(_start)


MEMORY
{
    ram (rwx) : ORIGIN = 0x80000000, LENGTH = 64k
    crypto_ram (rw) : ORIGIN = 0x61000000, LENGTH = 64k
}

RAM_START_ADDRESS   = 0x80000000;       /* Must be the same value MEMORY region ram ORIGIN above. */
RAM_SIZE            = 64k;              /* Must be the same value MEMORY region ram LENGTH above. */
STACK_SIZE          = 4k;               /* needs to be calculated for your application */
HEAP_SIZE           = 4k;               /* needs to be calculated for your application */

SECTIONS
{
  . = 0x80000000;
  .text : ALIGN(0x10)
  {
    *entry.o(.text);
    . = ALIGN(0x10);
    *(.text .text.* .gnu.linkonce.t.*)
```

Figure 11 • Linker Script (Continued)



```
{
    __bss_start = .;
    *(.shbss)
    *(.bss .bss.* .gnu.linkonce.b.*)
    *(COMMON)
    . = ALIGN(0x10);
    __bss_end = .;
} > ram

/* End of uninitialized data segment */
_end = .;

.heap : ALIGN(0x10)
{
    __heap_start = .;
    . += HEAP_SIZE;
    __heap_end = .;
    . = ALIGN(0x10);
    _heap_end = __heap_end;
} > ram

.stack : ALIGN(0x10)
{
    __stack_bottom = .;
    . += STACK_SIZE;
    . = ALIGN(0x10);
    __stack_top = .;
} > ram

. = 0x61000000;
 .crypto_data (NOLOAD) : ALIGN(0x10)
{
      . = ALIGN(0x10);
    *(.crypto_data)
} > crypto_ram
}
```

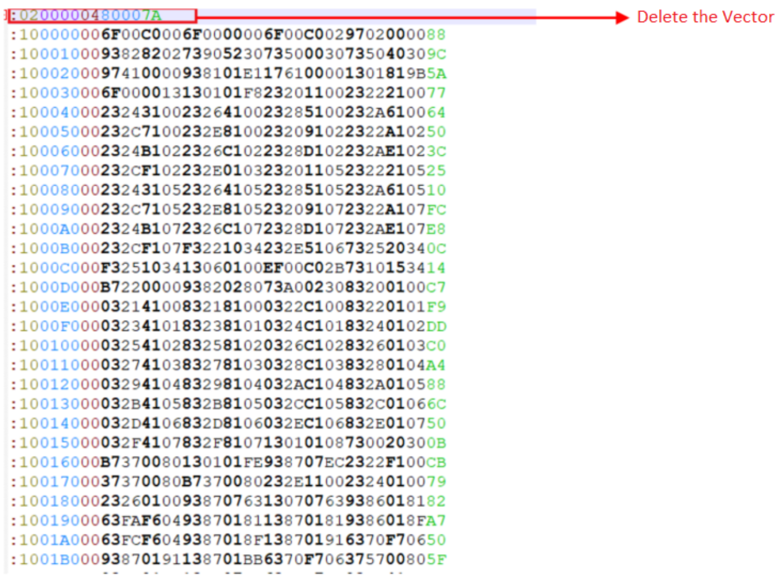
2.4.4 LSRAM Initialization from SPI Flash

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, complete the following steps:

1. Copy the `RV32_GNU_SC5_CCM_Services.hex` file from *<Directory_Structure>\mpf_ac473_liberosocpolarfirev2p2_df\SoftConsole\RV32_GNU_SC5_CCM_Services\Debug folder* and place it in *<Directory_Structure>\mpf_ac473_liberosocpolarfirev2p2_df\\Libero_Project\mpf_riscv_crypto folder*, remove the vectors to make it compatible with LSRAM as shown in the following figure and then save the file.

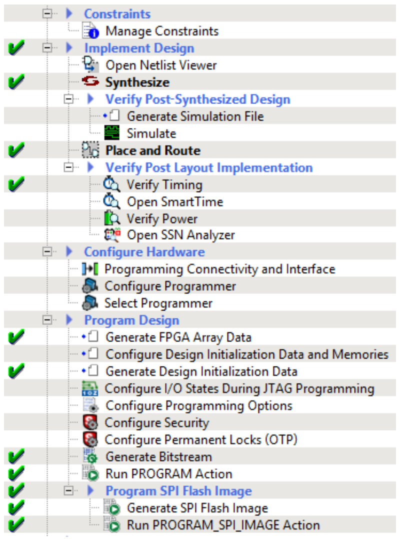
Figure 12 • Example of Hex File



```
:0200000480007A
:100000006F00C0006F0000006F00C0029702000088
:10001000938202273905230735000307350403309C
:1000200097410000938101E1176100001301819B5A
:100030006F000013130101F8232011002322210077
:10004000232431002326410023285100232A610064
:10005000232C7100232E81002320910223222A10250
:100060002324B1022326C1022328D102232AE1023C
:10007000232CF102232E0103232011052322210525
:10008000232431052326410523285105232A610510
:10009000232C7105232E81052320910723222A107FC
:1000A0002324B1072326C1072328D107232AE107E8
:1000B000232CF107F3221034232E5106732520340C
:1000C000F325103413060100EF00C02B7310153414
:1000D000B72200009382028073A0023083200100C7
:1000E00003214100832181000322C10083220101F9
:1000F00003234101832381010324C10183240102DD
:1001000003254102832581020326C10283260103C0
:1001100003274103832781030328C10383280104A4
:100120000329410483298104032AC104832A010588
:10013000032B4105832B8105032CC105832C01066C
:10014000032D4106832D8106032EC106832E010750
:10015000032F4107832F810713010108730020300B
:10016000B7370080130101FE938707EC2322F100CB
:1001700037370080B7370080232E11002324010079
:100180002326010093870763130707639386018182
:1001900063FAF60493870181138701819386018FA7
:1001A00063FCF6049387018F138701916370F70650
:1001B00093870191138701BB6370F706375700805F
```

2. Double-click **Configure Design Initialization Data and Memories** from the **Design Flow** window as shown in the following figure.

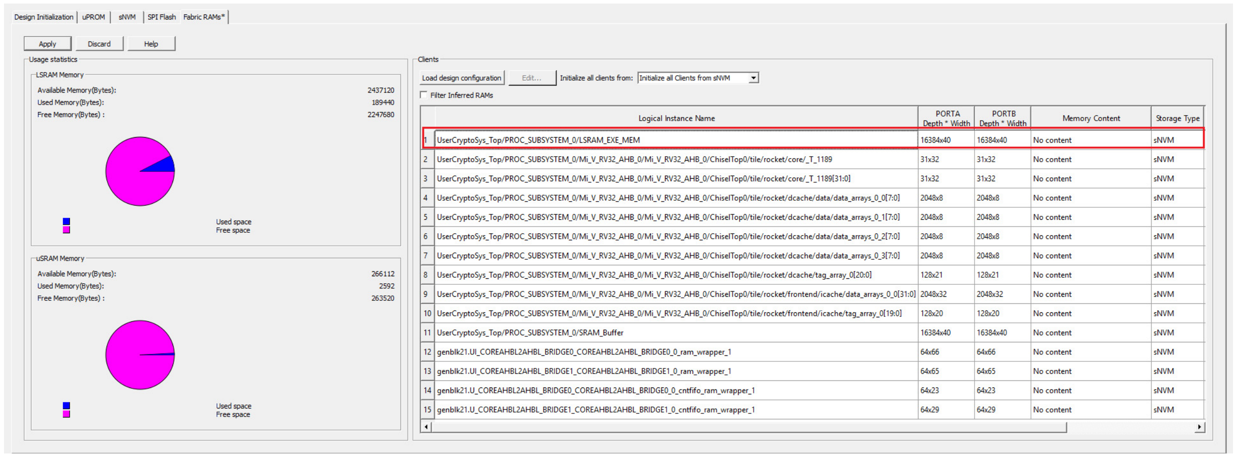
Figure 13 • **Configure Design Initialization Data and Memories**



Constraints
Manage Constraints
✓ Implement Design
Open Netlist Viewer
✓ Synthesize
Verify Post-Synthesized Design
Generate Simulation File
Simulate
✓ Place and Route
Verify Post Layout Implementation
✓ Verify Timing
Open SmartTime
Verify Power
Open SSN Analyzer
Configure Hardware
Programming Connectivity and Interface
Configure Programmer
Select Programmer
Program Design
✓ Generate FPGA Array Data
Configure Design Initialization Data and Memories
✓ Generate Design Initialization Data
Configure I/O States During JTAG Programming
Configure Programming Options
Configure Security
Configure Permanent Locks (OTP)
✓ Generate Bitstream
✓ Run PROGRAM Action
✓ Program SPI Flash Image
✓ Generate SPI Flash Image
✓ Run PROGRAM_SPI_IMAGE Action

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_EXE_MEM instance must be initialized with the user application.

Figure 14 • **Fabric RAMs**



Usage statistics

LSRAM Memory

Available Memory(Bytes):	2437120
Used Memory(Bytes):	189440
Free Memory(Bytes) :	2247680

🟦 Used space
🟪 Free space

uSRAM Memory

Available Memory(Bytes):	266112
Used Memory(Bytes):	2592
Free Memory(Bytes) :	263520

🟦 Used space
🟪 Free space

Clients

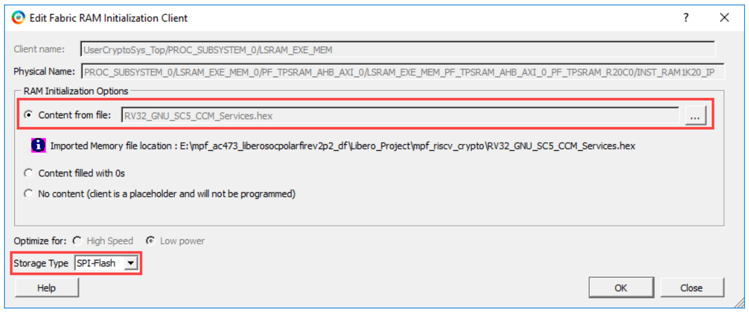
Initialize all clients from: Initialize all Clients from sNVM

☐ Filter Inferred RAMs

	Logical Instance Name	PORTA Depth * Width	PORTB Depth * Width	Memory Content	Storage Type
1	UserCryptoSys_Top/PROC_SUBSYSTEM_0/LSRAM_EXE_MEM	16384x40	16384x40	No content	sNVM
2	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/core/_T_1189	31x32	31x32	No content	sNVM
3	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/core/_T_1189[31:0]	31x32	31x32	No content	sNVM
4	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/dcache/data/data_arrays_0_0[7:0]	2048x8	2048x8	No content	sNVM
5	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/dcache/data/data_arrays_0_1[7:0]	2048x8	2048x8	No content	sNVM
6	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/dcache/data/data_arrays_0_2[7:0]	2048x8	2048x8	No content	sNVM
7	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/dcache/data/data_arrays_0_3[7:0]	2048x8	2048x8	No content	sNVM
8	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/dcache/tag_array_0[20:0]	128x21	128x21	No content	sNVM
9	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/frontend/icache/data_arrays_0_0[31:0]	2048x32	2048x32	No content	sNVM
10	UserCryptoSys_Top/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/Mi_V_RV32_AHB_0/ChiselTop0/tile/rocket/frontend/icache/tag_array_0[19:0]	128x20	128x20	No content	sNVM
11	UserCryptoSys_Top/PROC_SUBSYSTEM_0/SRAM_Buffer	16384x40	16384x40	No content	sNVM
12	genblk21.UI_COREAHBL2AHBL_BRIDGE0_COREAHBL2AHBL_BRIDGE0_0_ram_wrapper_1	64x66	64x66	No content	sNVM
13	genblk21.UI_COREAHBL2AHBL_BRIDGE1_COREAHBL2AHBL_BRIDGE1_0_ram_wrapper_1	64x65	64x65	No content	sNVM
14	genblk21.U_COREAHBL2AHBL_BRIDGE0_COREAHBL2AHBL_BRIDGE0_0_cntfifo_ram_wrapper_1	64x23	64x23	No content	sNVM
15	genblk21.U_COREAHBL2AHBL_BRIDGE1_COREAHBL2AHBL_BRIDGE1_0_cntfifo_ram_wrapper_1	64x29	64x29	No content	sNVM

4. Double-click the **LSRAM_EXE_MEM** instance to add initialization client and storage location. In the **Edit Fabric RAM Initialization Client** dialog, select the **Content from file** option, locate the `RV32_GNU_SC5_CCM_Services.hex` file from Libero Project folder, select **Storage Type** as **SPI-Flash**, and then click **OK** as shown in the following figure.

Figure 15 • **Edit Fabric RAM Initialization Client**



Client name: UserCryptoSys_Top/PROC_SUBSYSTEM_0/LSRAM_EXE_MEM

Physical Name: PROC_SUBSYSTEM_0/LSRAM_EXE_MEM_0/PF_TPSRAM_AHB_AXI_0/LSRAM_EXE_MEM_PF_TPSRAM_AHB_AXI_0_PF_TPSRAM_R20C0/INST_RAM1K20_IP

RAM Initialization Options

[✓ Content from file:] RV32_GNU_SC5_CCM_Services.hex

Imported Memory file location : E:\mpf_ac473_liberosocpolarfirev2p2_df\Libero_Project\mpf_riscv_crypto\RV32_GNU_SC5_CCM_Services.hex

Content filled with 0s

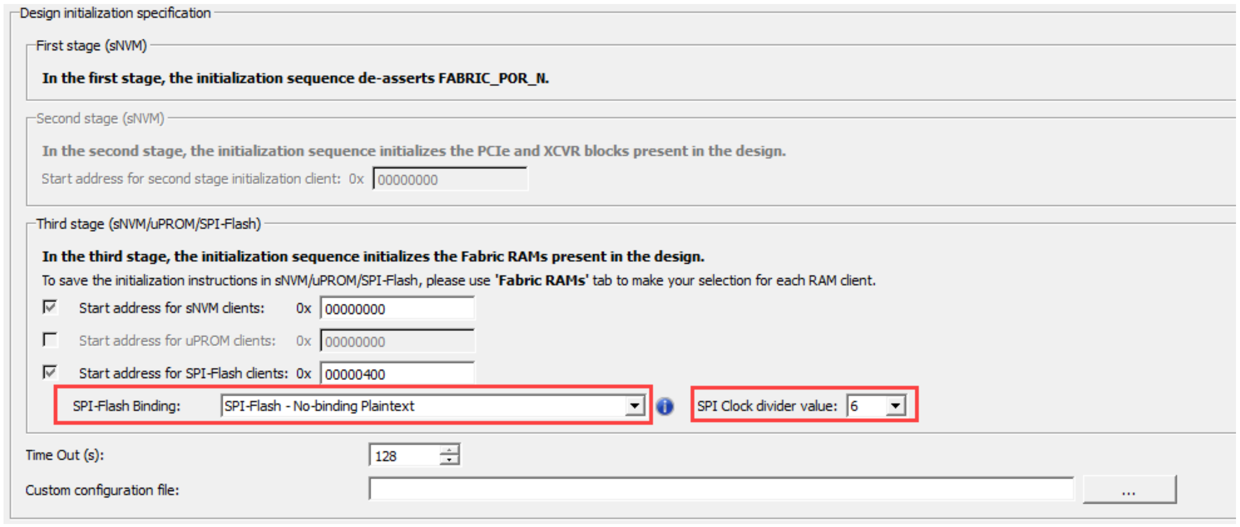
No content (client is a placeholder and will not be programmed)

Optimize for: High Speed [✓ Low power]

Storage Type SPI-Flash

5. Click the **Design Initialization** tab, ensure **SPI-Flash - No-binding Plaintext** is selected for **SPI-Flash Binding** type as shown in the following figure. **SPI clock divider value** must be set to 6. In this reference design, the initialization client is stored in the SPI flash in plaintext format without authentication. Enable security protection for the initialization SPI-Flash client if required.

Figure 16 • **Design Initialization**



Design initialization specification

First stage (sNVM)

In the first stage, the initialization sequence de-asserts FABRIC_POR_N.

Second stage (sNVM)

In the second stage, the initialization sequence initializes the PCIe and XCVR blocks present in the design.

Start address for second stage initialization client: 0x 00000000

Third stage (sNVM/uPROM/SPI-Flash)

In the third stage, the initialization sequence initializes the Fabric RAMs present in the design.

To save the initialization instructions in sNVM/uPROM/SPI-Flash, please use **'Fabric RAMs'** tab to make your selection for each RAM client.

- [✓] Start address for sNVM clients: 0x 00000000
- [ ] Start address for uPROM clients: 0x 00000000
- [✓] Start address for SPI-Flash clients: 0x 00000400

SPI-Flash Binding: SPI-Flash - No-binding Plaintext

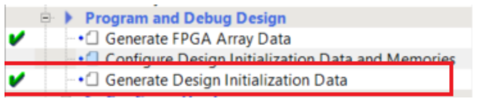
SPI Clock divider value: 6

Time Out (s): 128

Custom configuration file:

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 as shown in the following figure.

Figure 17 • **Generate Design Initialization Data**



Program and Debug Design

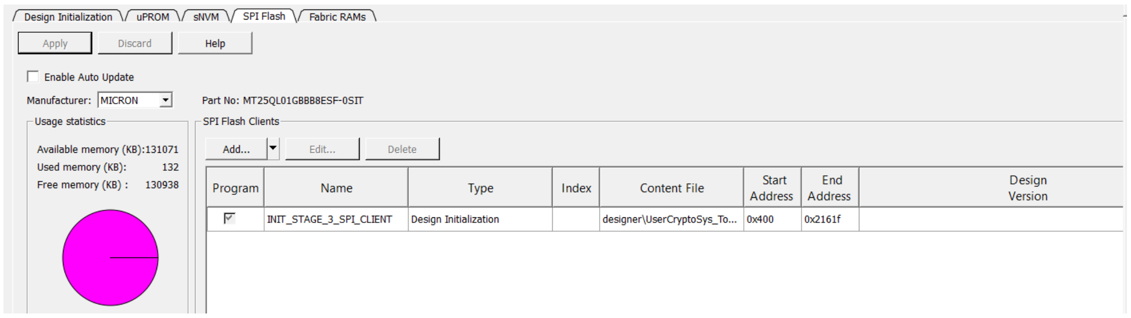
✓ Generate FPGA Array Data

Configure Design Initialization Data and Memories

✓ Generate Design Initialization Data

8. Select the **SPI Flash** tab to verify that the SPI flash client is generated as shown in the following figure.

Figure 18 • **sNVM Client Verification**



[ ] Enable Auto Update

Manufacturer: MICRON

Part No: MT25QL01GBBB8ESF-0SIT

Usage statistics

Available memory (KB):131071

Used memory (KB): 132

Free memory (KB) : 130938

SPI Flash Clients

Program	Name	Type	Index	Content File	Start Address	End Address	Design Version
✓	INIT_STAGE_3_SPI_CLIENT	Design Initialization		designer\UserCryptoSys_To...	0x400	0x2161f	

Configuration of Design Initialization Data and Memories is successfully completed.

2.5 Programming the PolarFire Device and SPI Flash

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

To program the PolarFire device, complete the following steps:

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

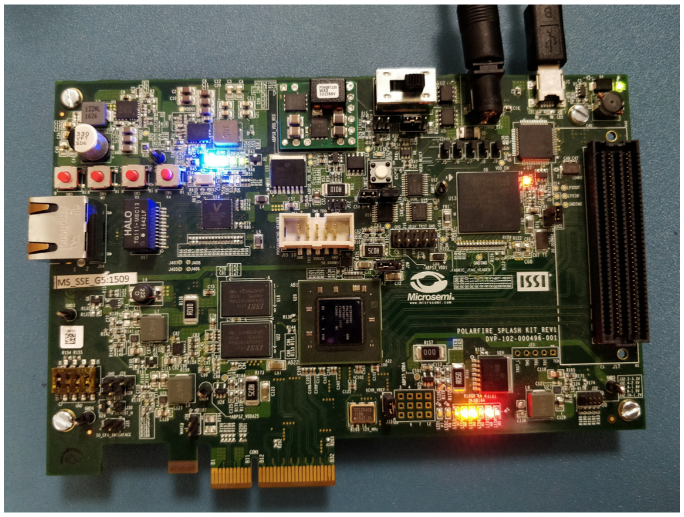
Table 5 • **Jumper Settings**

Jumper	Description
J5, J6, J7, J8, J9	Close pin 2 and 3 for programming the PolarFire FPGA through FTDI
J11	Close pin 1 and 2 for programming through FTDI chip
J10	Close pin 1 and 2 for programming through FTDI SPI
J4	Close pin 1 and 2 for manual power switching using SW1
J3	Open pin 1 and 2 for 1.0 V
J35	Open pin 1 and 2 to enable SPI Flash programming
J12	Short pin 3 and 4 for 2.5 V

2. Connect the power supply cable to the **J2** connector on the board.
3. Connect the USB cable from the host PC to **J1** (FTDI port) on the board.
4. Power on the board using the **SW1** slide switch.

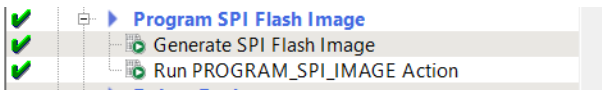
The following figure shows the PolarFire Splash Kit set up to be programmed.

Figure 19 • **Board Setup**



5. Open the Libero 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 as shown in the following figure.

Figure 20 • **Program SPI Flash Image**



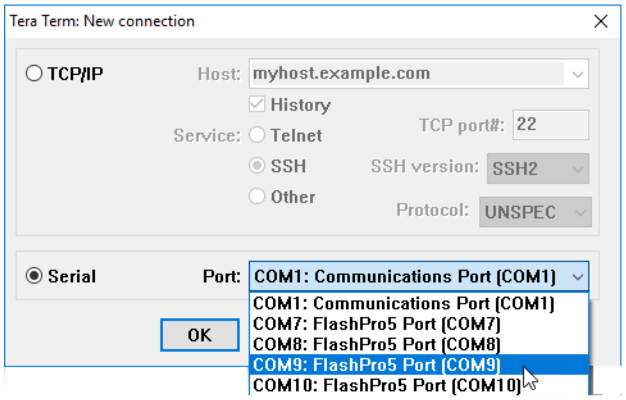
2.5.1 Tera Term Setup

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

To set up the Tera Term program:

1. Ensure that the USB cable connects the host PC to the **J1** (USB) port on the PolarFire Splash board.
2. Start Tera Term.
3. Select **Serial** as the Connection type.
4. Set the Serial **Port** to the second highest COM port number from the drop-down list as shown in the following figure. For example, **COM9: FlashPro5 Port [COM9]** in this instance.

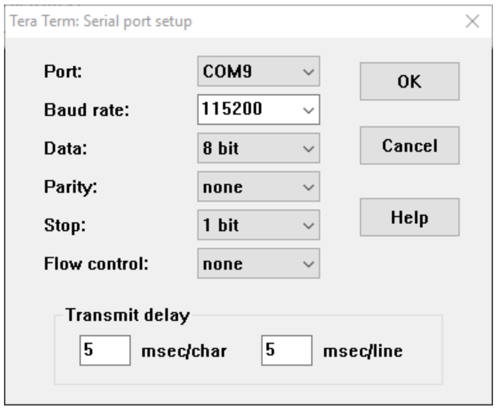
Figure 21 • **Select Serial as the Connection Type**


Tera Term: New connection
TCP/IP
Host: myhost.example.com
History
Service: Telnet
SSH
Other
TCP port#: 22
SSH version: SSH2
Protocol: UNSPEC
Serial
Port: COM1: Communications Port (COM1)
COM1: Communications Port (COM1)
COM7: FlashPro5 Port (COM7)
COM8: FlashPro5 Port (COM8)
COM9: FlashPro5 Port (COM9)
COM10: FlashPro5 Port (COM10)
OK

5. In the **Tera Term** window, go 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 as shown in the following figure and click **OK**.

Figure 22 • **Tera Term Configuration**

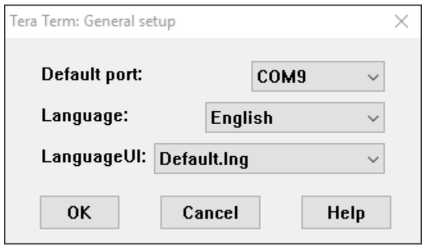

Tera Term: Serial port setup

Port:	COM9
Baud rate:	115200
Data:	8 bit
Parity:	none
Stop:	1 bit
Flow control:	none

Transmit delay: 5 msec/char 5 msec/line

6. In the **Tera Term** window, go to **Setup > General...**, set the **Language** to **English** and click **OK**, as shown in the following figure. This setup is required for running the Tera Term macro script.

Figure 23 • **Tera Term General Setup**

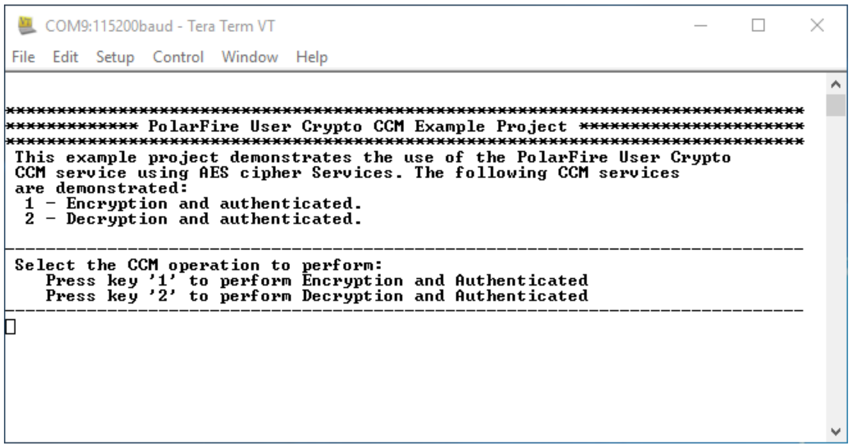

Tera Term: General setup

Default port:	COM9
Language:	English
LanguageUI:	Default.lng

This completes the Tera Term program setup.

2.6 Running the Demo

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.



COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
********************************************************************************
************* PolarFire User Crypto CCM Example Project ***********************
********************************************************************************
 This example project demonstrates the use of the PolarFire User Crypto
 CCM service using AES cipher Services. The following CCM services
 are demonstrated:
  1 - Encryption and authenticated.
  2 - Decryption and authenticated.

--------------------------------------------------------------------------------
 Select the CCM operation to perform:
    Press key '1' to perform Encryption and Authenticated
    Press key '2' to perform Decryption and Authenticated
--------------------------------------------------------------------------------
```

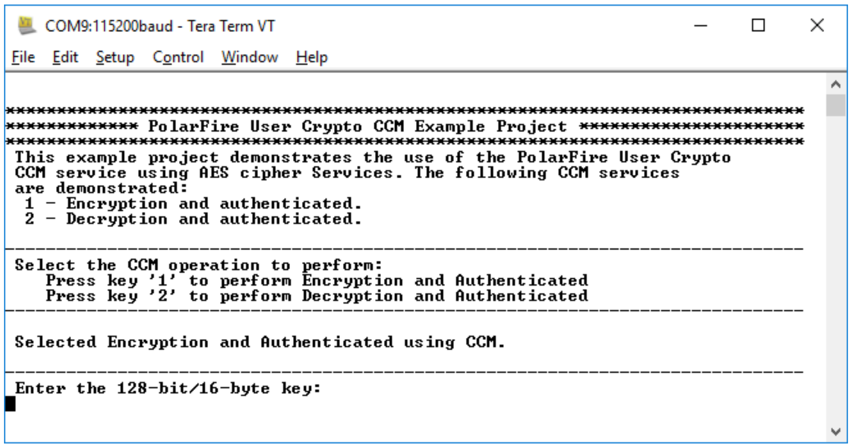
Figure 24 • **Application Menu**

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

- AES Key = 404142434445464748494A4B4C4D4E4F
- Nonce = 101112131415161718191A1B
- Additional authentication data (AAD) = 00010203 0405060708090A0B0C0D0E0F10111213
- Input data to Encrypt and authenticate = 202122232425262728292A2B2C2D2E2F3031323334353637

To run the demo:

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



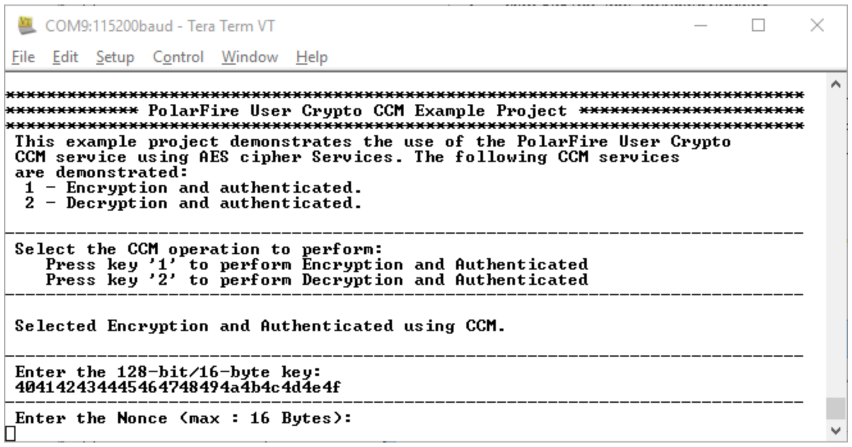
COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
********************************************************************************
************* PolarFire User Crypto CCM Example Project ***********************
********************************************************************************
 This example project demonstrates the use of the PolarFire User Crypto
 CCM service using AES cipher Services. The following CCM services
 are demonstrated:
  1 - Encryption and authenticated.
  2 - Decryption and authenticated.

--------------------------------------------------------------------------------
 Select the CCM operation to perform:
    Press key '1' to perform Encryption and Authenticated
    Press key '2' to perform Decryption and Authenticated
--------------------------------------------------------------------------------

 Selected Encryption and Authenticated using CCM.

--------------------------------------------------------------------------------
 Enter the 128-bit/16-byte key:
```

Figure 25 • **Application Menu—Enter 128-bit Key**

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

Figure 26 • **Application Menu—Nonce**

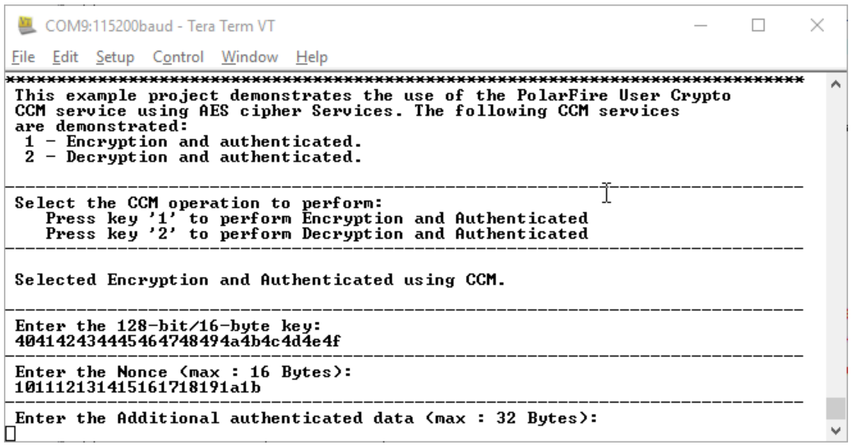


COM9:115200baud - Tera Term VT

```
*******************************************************************************
************* PolarFire User Crypto CCM Example Project ***********************
*******************************************************************************
 This example project demonstrates the use of the PolarFire User Crypto
 CCM service using AES cipher Services. The following CCM services
 are demonstrated:
  1 - Encryption and authenticated.
  2 - Decryption and authenticated.

-------------------------------------------------------------------------------
 Select the CCM operation to perform:
    Press key '1' to perform Encryption and Authenticated
    Press key '2' to perform Decryption and Authenticated
-------------------------------------------------------------------------------

 Selected Encryption and Authenticated using CCM.

-------------------------------------------------------------------------------
 Enter the 128-bit/16-byte key:
 404142434445464748494a4b4c4d4e4f
-------------------------------------------------------------------------------
 Enter the Nonce (max : 16 Bytes):
```

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 27 • **Application Menu—Additional Authentication Data**

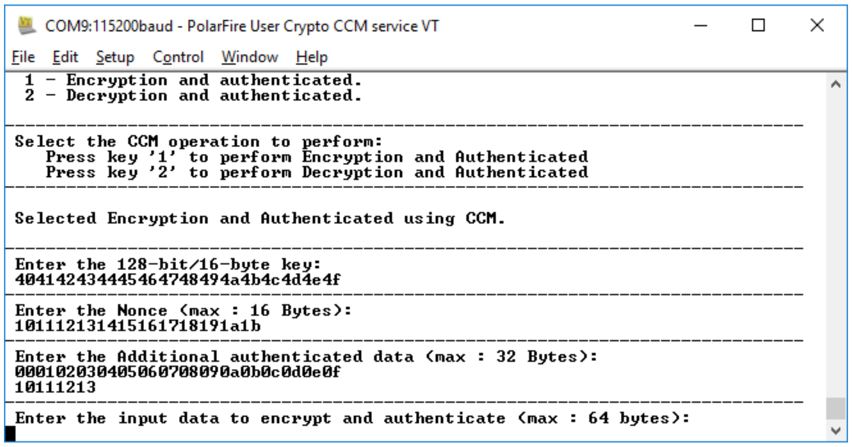


COM9:115200baud - Tera Term VT

```
*******************************************************************************
 This example project demonstrates the use of the PolarFire User Crypto
 CCM service using AES cipher Services. The following CCM services
 are demonstrated:
  1 - Encryption and authenticated.
  2 - Decryption and authenticated.

-------------------------------------------------------------------------------
 Select the CCM operation to perform:
    Press key '1' to perform Encryption and Authenticated
    Press key '2' to perform Decryption and Authenticated
-------------------------------------------------------------------------------

 Selected Encryption and Authenticated using CCM.

-------------------------------------------------------------------------------
 Enter the 128-bit/16-byte key:
 404142434445464748494a4b4c4d4e4f
-------------------------------------------------------------------------------
 Enter the Nonce (max : 16 Bytes):
 101112131415161718191a1b
-------------------------------------------------------------------------------
 Enter the Additional authenticated data (max : 32 Bytes):
```

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 28 • **Application Menu—Input Data to Encrypt and Authenticate**

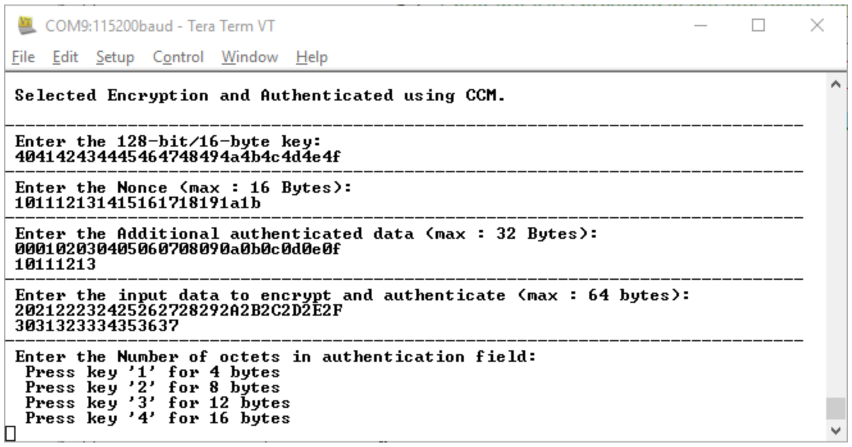


COM9:115200baud - PolarFire User Crypto CCM service VT

```
  1 - Encryption and authenticated.
  2 - Decryption and authenticated.

-------------------------------------------------------------------------------
 Select the CCM operation to perform:
    Press key '1' to perform Encryption and Authenticated
    Press key '2' to perform Decryption and Authenticated
-------------------------------------------------------------------------------

 Selected Encryption and Authenticated using CCM.

-------------------------------------------------------------------------------
 Enter the 128-bit/16-byte key:
 404142434445464748494a4b4c4d4e4f
-------------------------------------------------------------------------------
 Enter the Nonce (max : 16 Bytes):
 101112131415161718191a1b
-------------------------------------------------------------------------------
 Enter the Additional authenticated data (max : 32 Bytes):
 000102030405060708090a0b0c0d0e0f
 10111213
-------------------------------------------------------------------------------
 Enter the input data to encrypt and authenticate (max : 64 bytes):
```

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 authentication field, as shown in the following figure.

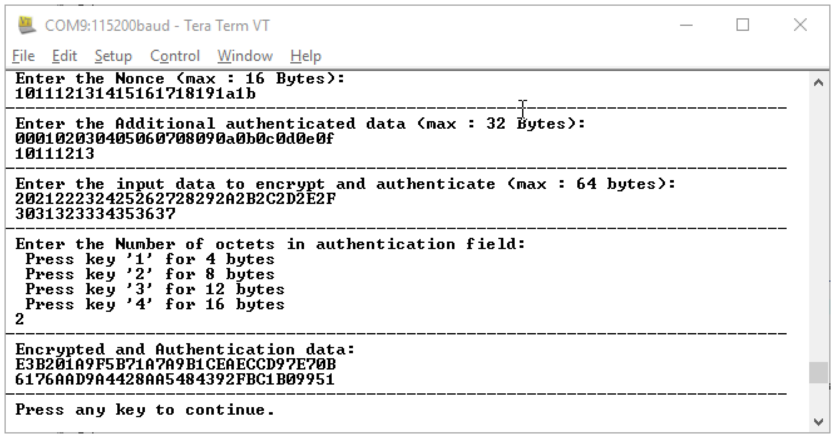
Figure 29 • **Application Menu—Number of Octets in authentication Field**


COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
Selected Encryption and Authenticated using CCM.
--------------------------------------------------------------------------------
Enter the 128-bit/16-byte key:
404142434445464748494a4b4c4d4e4f
--------------------------------------------------------------------------------
Enter the Nonce (max : 16 Bytes):
101112131415161718191a1b
--------------------------------------------------------------------------------
Enter the Additional authenticated data (max : 32 Bytes):
000102030405060708090a0b0c0d0e0f
10111213
--------------------------------------------------------------------------------
Enter the input data to encrypt and authenticate (max : 64 bytes):
202122232425262728292A2B2C2D2E2F
3031323334353637
--------------------------------------------------------------------------------
Enter the Number of octets in authentication field:
 Press key '1' for 4 bytes
 Press key '2' for 8 bytes
 Press key '3' for 12 bytes
 Press key '4' for 16 bytes
```

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 30 • **Application Menu—Encrypted and Authentication Data**

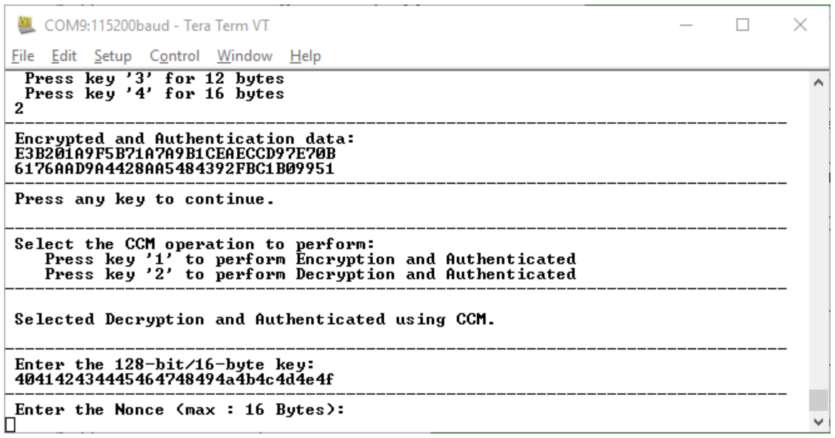

COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
Enter the Nonce (max : 16 Bytes):
101112131415161718191a1b
--------------------------------------------------------------------------------
Enter the Additional authenticated data (max : 32 Bytes):
000102030405060708090a0b0c0d0e0f
10111213
--------------------------------------------------------------------------------
Enter the input data to encrypt and authenticate (max : 64 bytes):
202122232425262728292A2B2C2D2E2F
3031323334353637
--------------------------------------------------------------------------------
Enter the Number of octets in authentication field:
 Press key '1' for 4 bytes
 Press key '2' for 8 bytes
 Press key '3' for 12 bytes
 Press key '4' for 16 bytes
2
--------------------------------------------------------------------------------
Encrypted and Authentication data:
E3B201A9F5B71A7A9B1CEAECCD97E70B
6176AAD9A4428AA5484392FBC1B09951
--------------------------------------------------------------------------------
Press any key to continue.
```

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, as shown in the following figure.

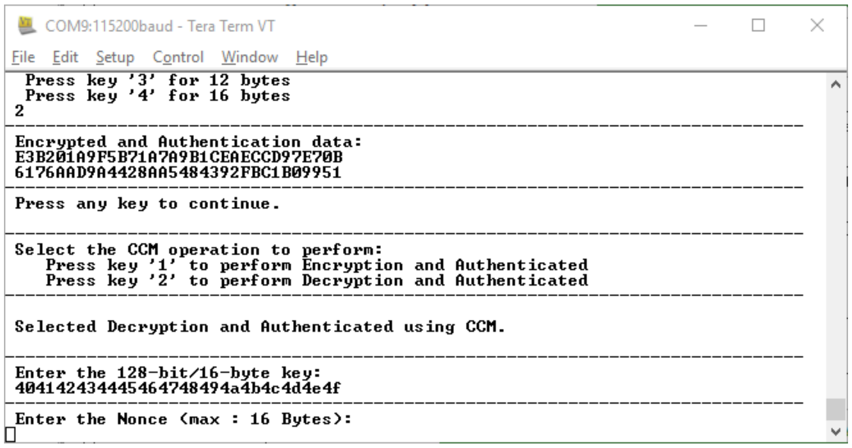
Figure 31 • **Application Menu Decryption—Enter 128-bit Key for AES Decryption**


COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
 Press key '3' for 12 bytes
 Press key '4' for 16 bytes
2
--------------------------------------------------------------------------------
Encrypted and Authentication data:
E3B201A9F5B71A7A9B1CEAECCD97E70B
6176AAD9A4428AA5484392FBC1B09951
--------------------------------------------------------------------------------
Press any key to continue.

--------------------------------------------------------------------------------
Select the CCM operation to perform:
   Press key '1' to perform Encryption and Authenticated
   Press key '2' to perform Decryption and Authenticated
--------------------------------------------------------------------------------

Selected Decryption and Authenticated using CCM.

--------------------------------------------------------------------------------
Enter the 128-bit/16-byte key:
404142434445464748494a4b4c4d4e4f
--------------------------------------------------------------------------------
Enter the Nonce (max : 16 Bytes):
```

9. Enter the key provided in the preceding test vector and press Enter. The application prompts for Nonce.

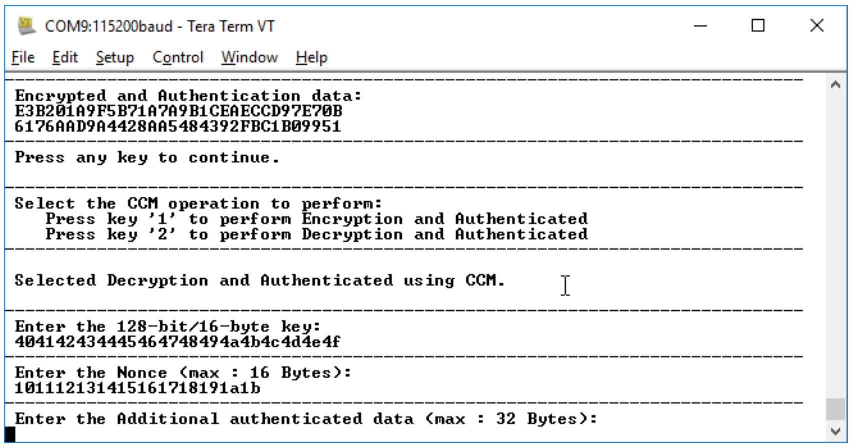
Figure 32 • **Application Menu Decryption—Nonce**


COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
 Press key '3' for 12 bytes
 Press key '4' for 16 bytes
2
--------------------------------------------------------------------------------
Encrypted and Authentication data:
E3B201A9F5B71A7A9B1CEAECCD97E70B
6176AAD9A4428AA5484392FBC1B09951
--------------------------------------------------------------------------------
Press any key to continue.

--------------------------------------------------------------------------------
Select the CCM operation to perform:
   Press key '1' to perform Encryption and Authenticated
   Press key '2' to perform Decryption and Authenticated
--------------------------------------------------------------------------------

Selected Decryption and Authenticated using CCM.

--------------------------------------------------------------------------------
Enter the 128-bit/16-byte key:
404142434445464748494a4b4c4d4e4f
--------------------------------------------------------------------------------
Enter the Nonce (max : 16 Bytes):
```

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

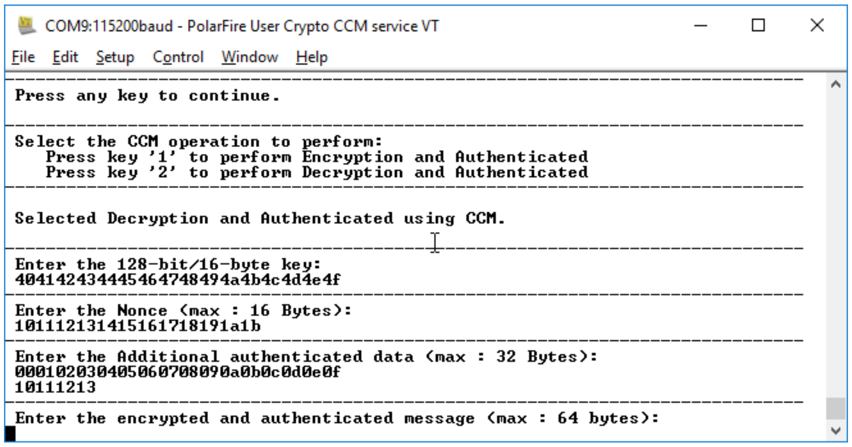
Figure 33 • **Application Menu—Additional Authentication Data**


COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
--------------------------------------------------------------------------------
Encrypted and Authentication data:
E3B201A9F5B71A7A9B1CEAECCD97E70B
6176AAD9A4428AA5484392FBC1B09951
--------------------------------------------------------------------------------
Press any key to continue.

--------------------------------------------------------------------------------
Select the CCM operation to perform:
   Press key '1' to perform Encryption and Authenticated
   Press key '2' to perform Decryption and Authenticated
--------------------------------------------------------------------------------

Selected Decryption and Authenticated using CCM.

--------------------------------------------------------------------------------
Enter the 128-bit/16-byte key:
404142434445464748494a4b4c4d4e4f
--------------------------------------------------------------------------------
Enter the Nonce (max : 16 Bytes):
101112131415161718191a1b
--------------------------------------------------------------------------------
Enter the Additional authenticated data (max : 32 Bytes):
```

11. 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.

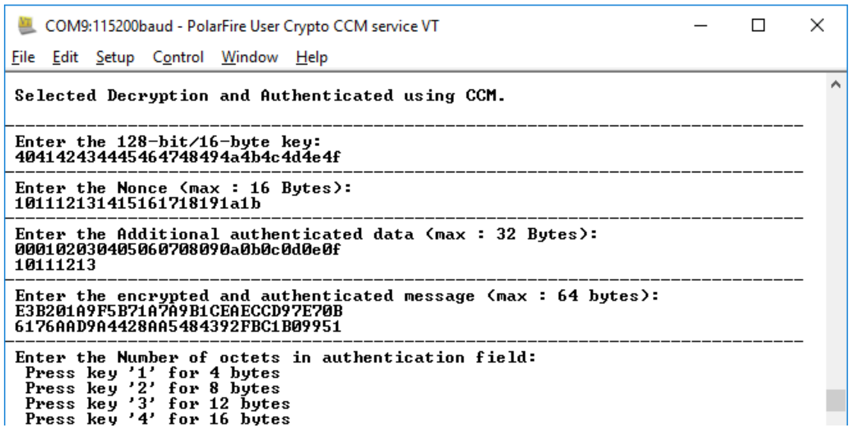
Figure 34 • **Application Menu—Input Data to Encrypt and Authenticate**


COM9:115200baud - PolarFire User Crypto CCM service VT
File Edit Setup Control Window Help

```
Press any key to continue.
--------------------------------------------------------------------------------
Select the CCM operation to perform:
   Press key '1' to perform Encryption and Authenticated
   Press key '2' to perform Decryption and Authenticated
--------------------------------------------------------------------------------
Selected Decryption and Authenticated using CCM.
--------------------------------------------------------------------------------
Enter the 128-bit/16-byte key:
404142434445464748494a4b4c4d4e4f
--------------------------------------------------------------------------------
Enter the Nonce (max : 16 Bytes):
101112131415161718191a1b
--------------------------------------------------------------------------------
Enter the Additional authenticated data (max : 32 Bytes):
000102030405060708090a0b0c0d0e0f
10111213
--------------------------------------------------------------------------------
Enter the encrypted and authenticated message (max : 64 bytes):
```

12. 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 authentication field, as shown in the following figure.

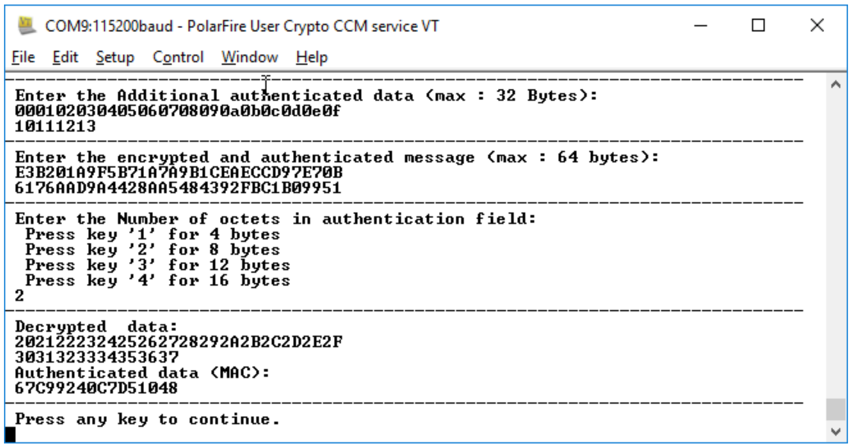
Figure 35 • **Application Menu—Encrypted and Authentication Data**


COM9:115200baud - PolarFire User Crypto CCM service VT
File Edit Setup Control Window Help

```
Selected Decryption and Authenticated using CCM.
--------------------------------------------------------------------------------
Enter the 128-bit/16-byte key:
404142434445464748494a4b4c4d4e4f
--------------------------------------------------------------------------------
Enter the Nonce (max : 16 Bytes):
101112131415161718191a1b
--------------------------------------------------------------------------------
Enter the Additional authenticated data (max : 32 Bytes):
000102030405060708090a0b0c0d0e0f
10111213
--------------------------------------------------------------------------------
Enter the encrypted and authenticated message (max : 64 bytes):
E3B201A9F5B71A7A9B1CEAECCD97E70B
6176AAD9A4428AA5484392FBC1B09951
--------------------------------------------------------------------------------
Enter the Number of octets in authentication field:
 Press key '1' for 4 bytes
 Press key '2' for 8 bytes
 Press key '3' for 12 bytes
 Press key '4' for 16 bytes
```

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 36 • **Application Menu—AES CCM Decryption Result**

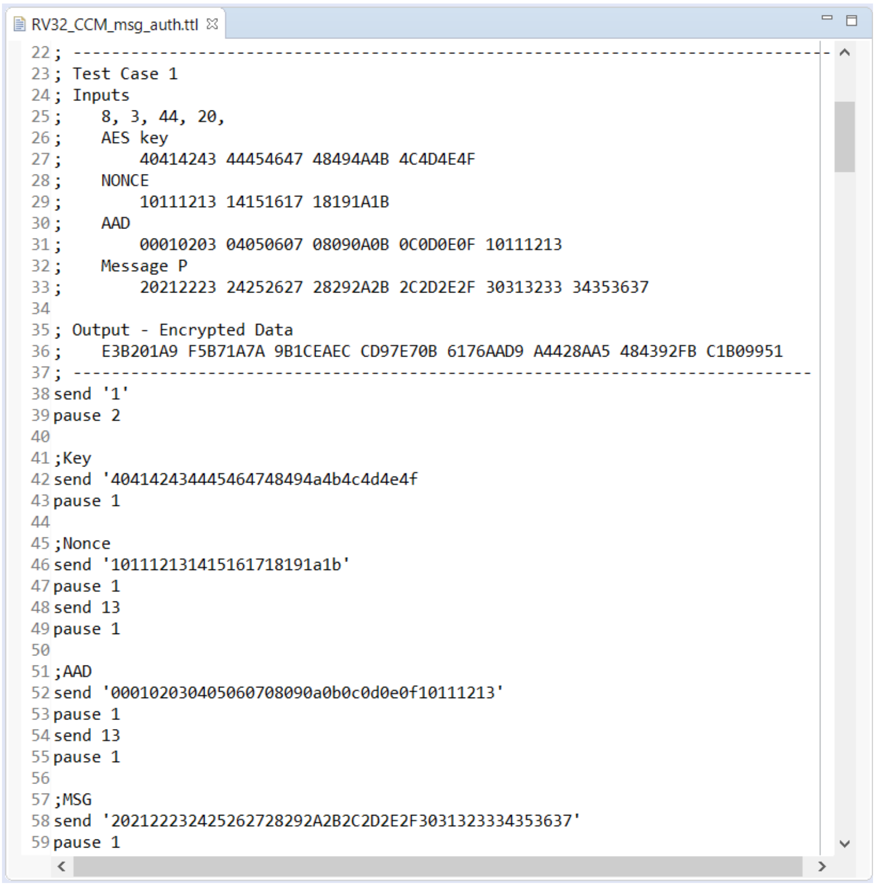

COM9:115200baud - PolarFire User Crypto CCM service VT
File Edit Setup Control Window Help

```
Enter the Additional authenticated data (max : 32 Bytes):
000102030405060708090a0b0c0d0e0f
10111213
--------------------------------------------------------------------------------
Enter the encrypted and authenticated message (max : 64 bytes):
E3B201A9F5B71A7A9B1CEAECCD97E70B
6176AAD9A4428AA5484392FBC1B09951
--------------------------------------------------------------------------------
Enter the Number of octets in authentication field:
 Press key '1' for 4 bytes
 Press key '2' for 8 bytes
 Press key '3' for 12 bytes
 Press key '4' for 16 bytes
2
--------------------------------------------------------------------------------
Decrypted  data:
202122232425262728292A2B2C2D2E2F
3031323334353637
Authenticated data (MAC):
67C99240C7D51048
--------------------------------------------------------------------------------
Press any key to continue.
```

2.6.1 Running Tera Term Macro Script

Each sample project contains Tera Term macro script (.ttl) file to test the cryptographic algorithms used in the sample project. The Tera Term macro script contain 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 Tera Term macro script available in the sample projects provided in the design file.

Figure 37 • Tera Term Macro Script to Test AES Service



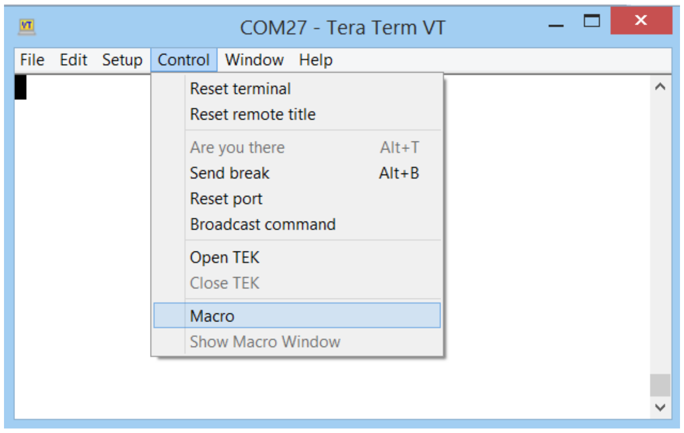
RV32_CCM_msg_auth.ttl

```
; ---------------------------------------------------------------------------
; Test Case 1
; Inputs
;    8, 3, 44, 20,
;    AES key
;        40414243 44454647 48494A4B 4C4D4E4F
;    NONCE
;        10111213 14151617 18191A1B
;    AAD
;        00010203 04050607 08090A0B 0C0D0E0F 10111213
;    Message P
;        20212223 24252627 28292A2B 2C2D2E2F 30313233 34353637

; Output - Encrypted Data
;    E3B201A9 F5B71A7A 9B1CEAEC CD97E70B 6176AAD9 A4428AA5 484392FB C1B09951
; ---------------------------------------------------------------------------
send '1'
pause 2

;Key
send '404142434445464748494a4b4c4d4e4f
pause 1

;Nonce
send '101112131415161718191a1b'
pause 1
send 13
pause 1

;AAD
send '000102030405060708090a0b0c0d0e0f10111213'
pause 1
send 13
pause 1

;MSG
send '202122232425262728292A2B2C2D2E2F3031323334353637'
pause 1
```

To run the Tera Term script:

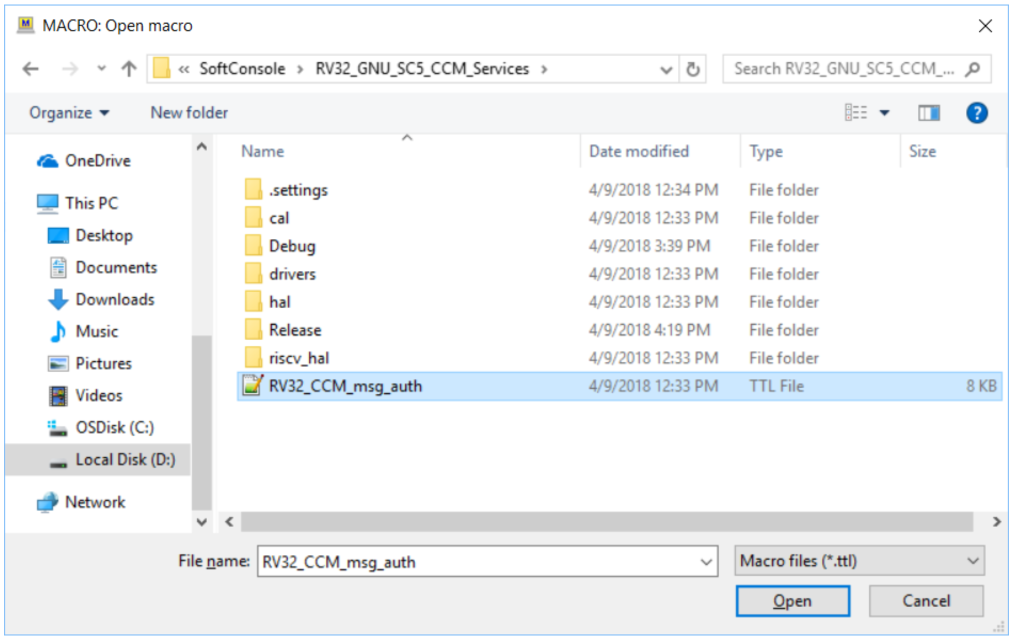
1. On the **Control** menu, click **Macro**, as shown in the following figure.

Figure 38 • **Executing Macro Script**



2. In the **Open macro** window, select the `RV32_CCM_msg_auth.ttl` file available in the sample project provided with the design files and click **Open** as shown in the following figure.

Figure 39 • **Selecting Macro Script File**

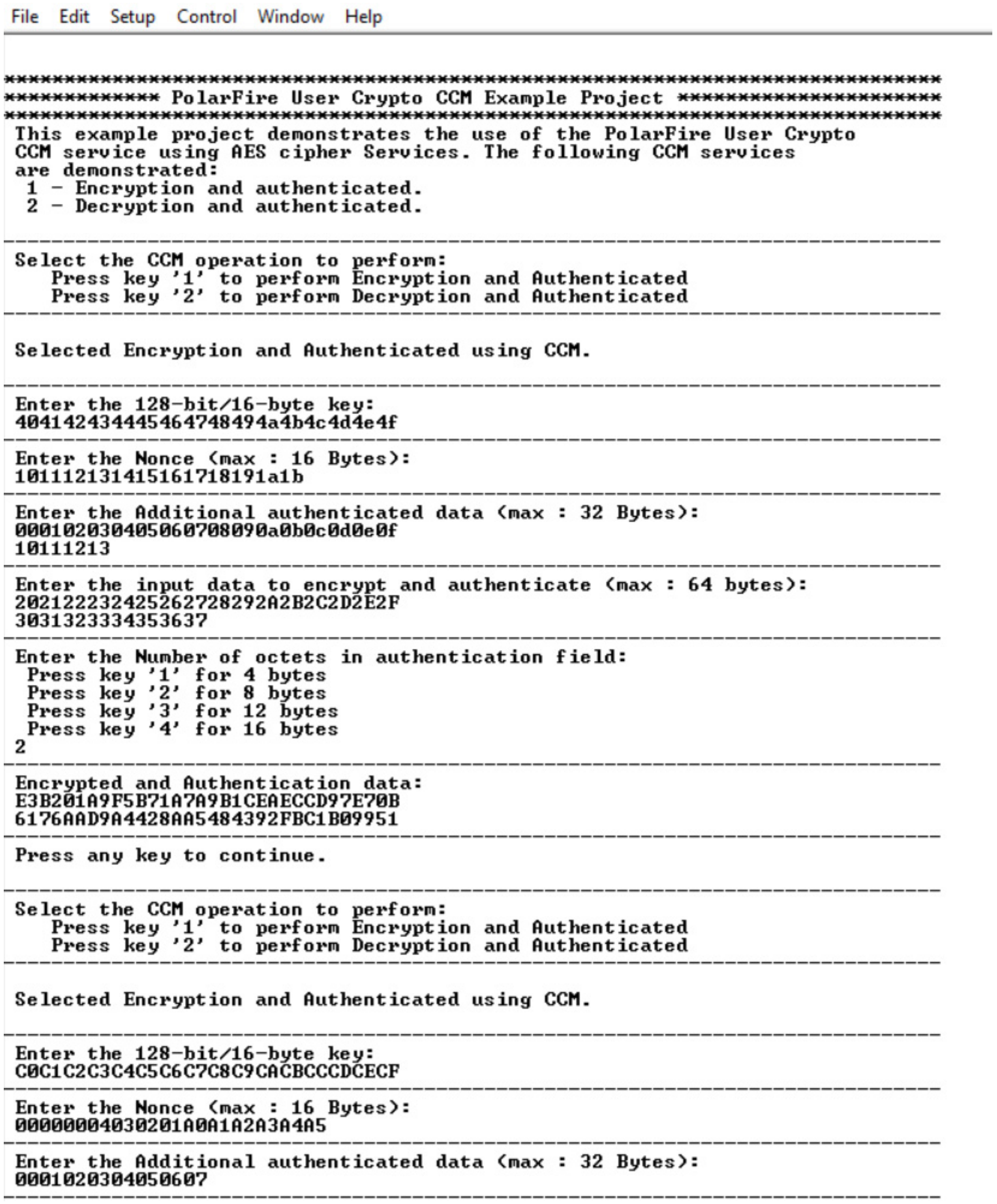


Name	Date modified	Type	Size
.settings	4/9/2018 12:34 PM	File folder	
cal	4/9/2018 12:33 PM	File folder	
Debug	4/9/2018 3:39 PM	File folder	
drivers	4/9/2018 12:33 PM	File folder	
hal	4/9/2018 12:33 PM	File folder	
Release	4/9/2018 4:19 PM	File folder	
riscv_hal	4/9/2018 12:33 PM	File folder	
RV32_CCM_msg_auth	4/9/2018 12:33 PM	TTL File	8 KB

File name: RV32_CCM_msg_auth

Tera Term executes the script and the results are printed on the Tera Term. 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 40 • **Macro Script Execution Result**



```
******************************************************************************
************* PolarFire User Crypto CCM Example Project **********************
******************************************************************************
 This example project demonstrates the use of the PolarFire User Crypto
 CCM service using AES cipher Services. The following CCM services
 are demonstrated:
  1 - Encryption and authenticated.
  2 - Decryption and authenticated.

------------------------------------------------------------------------------
 Select the CCM operation to perform:
    Press key '1' to perform Encryption and Authenticated
    Press key '2' to perform Decryption and Authenticated
------------------------------------------------------------------------------

 Selected Encryption and Authenticated using CCM.

------------------------------------------------------------------------------
 Enter the 128-bit/16-byte key:
 404142434445464748494a4b4c4d4e4f
------------------------------------------------------------------------------
 Enter the Nonce (max : 16 Bytes):
 101112131415161718191a1b
------------------------------------------------------------------------------
 Enter the Additional authenticated data (max : 32 Bytes):
 000102030405060708090a0b0c0d0e0f
 10111213
------------------------------------------------------------------------------
 Enter the input data to encrypt and authenticate (max : 64 bytes):
 202122232425262728292A2B2C2D2E2F
 3031323334353637
------------------------------------------------------------------------------
 Enter the Number of octets in authentication field:
  Press key '1' for 4 bytes
  Press key '2' for 8 bytes
  Press key '3' for 12 bytes
  Press key '4' for 16 bytes
 2
------------------------------------------------------------------------------
 Encrypted and Authentication data:
 E3B201A9F5B71A7A9B1CEAECCD97E70B
 6176AAD9A4428AA5484392FBC1B09951
------------------------------------------------------------------------------
 Press any key to continue.

------------------------------------------------------------------------------
 Select the CCM operation to perform:
    Press key '1' to perform Encryption and Authenticated
    Press key '2' to perform Decryption and Authenticated
------------------------------------------------------------------------------

 Selected Encryption and Authenticated using CCM.

------------------------------------------------------------------------------
 Enter the 128-bit/16-byte key:
 C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF
------------------------------------------------------------------------------
 Enter the Nonce (max : 16 Bytes):
 00000003020100A0A1A2A3A4A5
------------------------------------------------------------------------------
 Enter the Additional authenticated data (max : 32 Bytes):
 0001020304050607
------------------------------------------------------------------------------
```

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

To run other UserCrypto sample projects, see Appendix: Running UserCrypto Sample Projects, page 25.

3 Appendix: Running UserCrypto Sample Projects

The Firmware Catalog contains sample projects to show how to use UserCrypto (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, refer to the README.txt, file available in the sample projects.

Table 6 • **Algorithm**

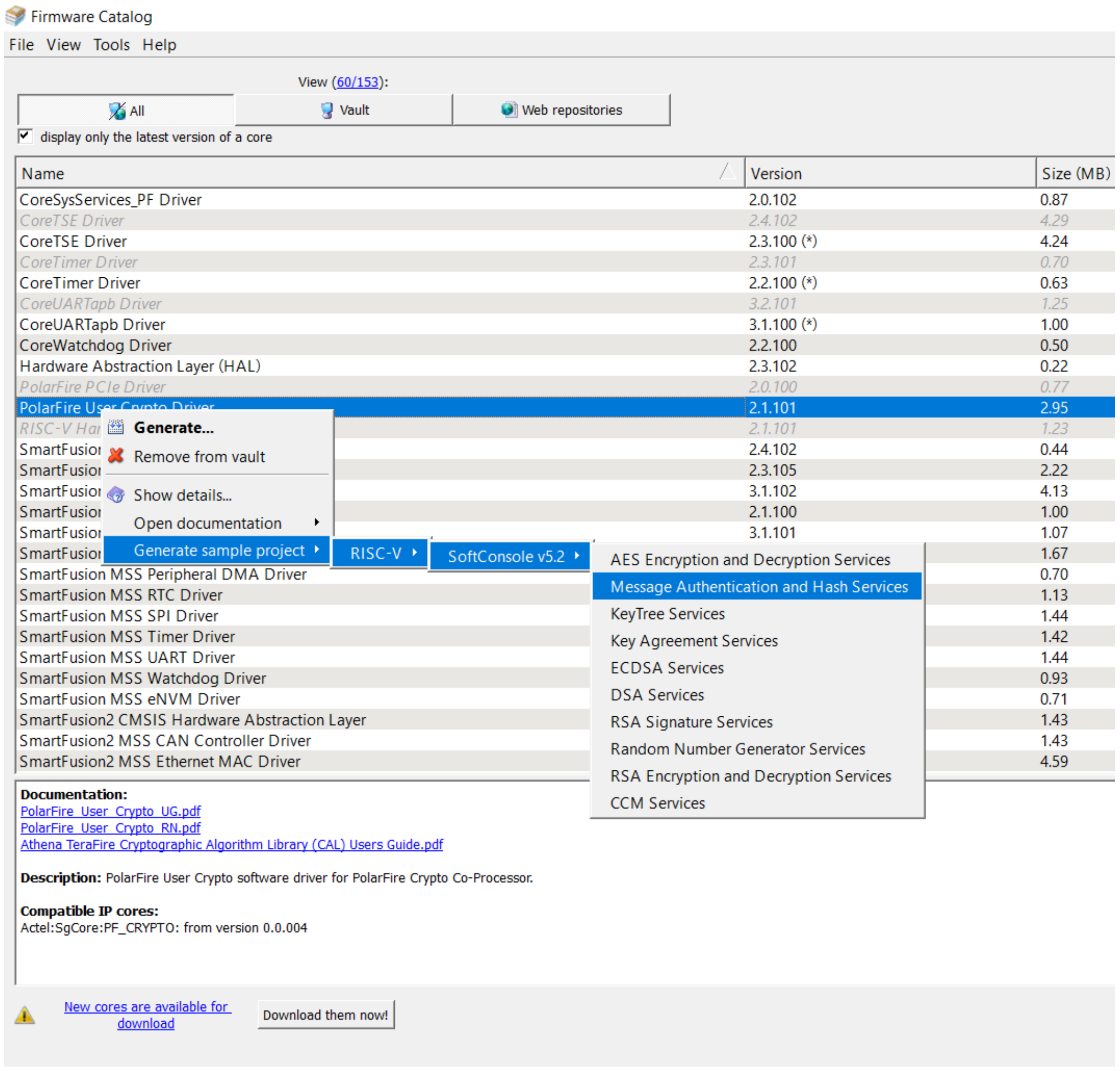
Algorithm	**Parameters and Modes**
AES	AES-ECB-256 encrypt
	AES-ECB-256 encrypt
	AES-CCM-128
GMAC	AES-GCM-256, 128-bit tag
HMAC	HMAC-SHA-256, 256-bit key
CMAC	AES-CMAC-256
KEY TREE	128-bit nonce + 8-bit optype
SHA	SHA-256
ECC	ECDSA SigGen, P-384/SHA-384, DPA
IFC (RSA)	Encrypt, RSA-3072, e=65537
	Decrypt, 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

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

To run the sample projects:

1. Download the sample project from Firmware Catalog by right-clicking on **PolarFire UserCrypto Driver** and select a sample project available through **Generate Sample Project > RISC-V > SoftConsole 5.2 > project name**, as shown in the following figure.

Figure 41 • **Firmware Catalog—Sample Projects**



Name	Version	Size (MB)
CoreSysServices_PF Driver	2.0.102	0.87
CoreTSE Driver	2.4.102	4.29
CoreTSE Driver	2.3.100 (*)	4.24
CoreTimer Driver	2.3.101	0.70
CoreTimer Driver	2.2.100 (*)	0.63
CoreUARTapb Driver	3.2.101	1.25
CoreUARTapb Driver	3.1.100 (*)	1.00
CoreWatchdog Driver	2.2.100	0.50
Hardware Abstraction Layer (HAL)	2.3.102	0.22
PolarFire PCIe Driver	2.0.100	0.77
PolarFire User Crypto Driver	2.1.101	2.95
RISC-V Har	2.1.101	1.23
SmartFusio	2.4.102	0.44
SmartFusio	2.3.105	2.22
SmartFusio	3.1.102	4.13
SmartFusio	2.1.100	1.00
SmartFusio	3.1.101	1.07
SmartFusio		1.67
SmartFusion MSS Peripheral DMA Driver		0.70
SmartFusion MSS RTC Driver		1.13
SmartFusion MSS SPI Driver		1.44
SmartFusion MSS Timer Driver		1.42
SmartFusion MSS UART Driver		1.44
SmartFusion MSS Watchdog Driver		0.93
SmartFusion MSS eNVM Driver		0.71
SmartFusion2 CMSIS Hardware Abstraction Layer		1.43
SmartFusion2 MSS CAN Controller Driver		1.43
SmartFusion2 MSS Ethernet MAC Driver		4.59

Documentation:
PolarFire_User_Crypto_UG.pdf
PolarFire_User_Crypto_RN.pdf
Athena TeraFire Cryptographic Algorithm Library (CAL) Users Guide.pdf

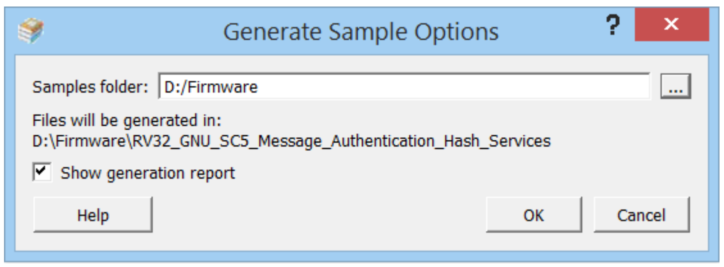
Description: PolarFire User Crypto software driver for PolarFire Crypto Co-Processor.

Compatible IP cores:
Actel:SgCore:PF_CRYPTO: from version 0.0.004

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

2. Click **OK** to generate the selected sample project to a local folder on the host PC.

Figure 42 • **Generate Sample Project**



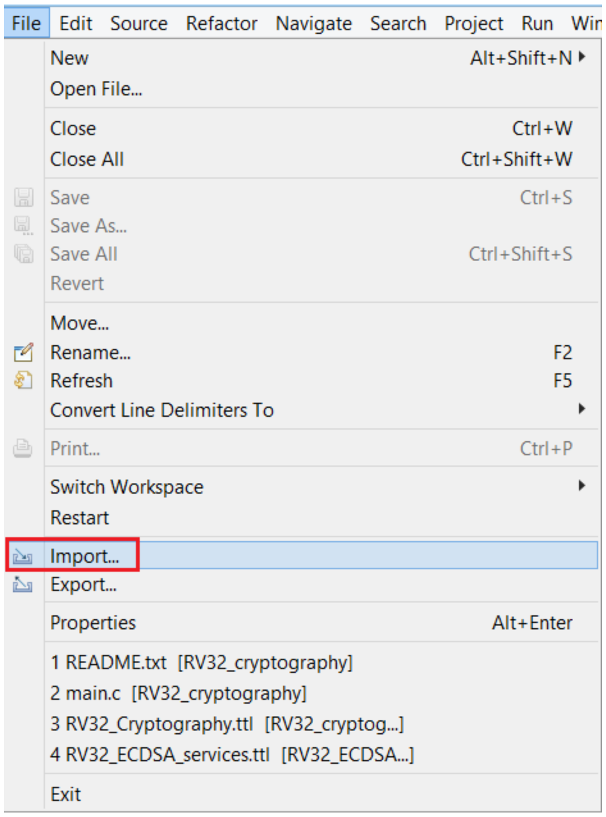
Samples folder: D:/Firmware

Files will be generated in:
D:\Firmware\RV32_GNU_SC5_Message_Authentication_Hash_Services

☑ Show generation report

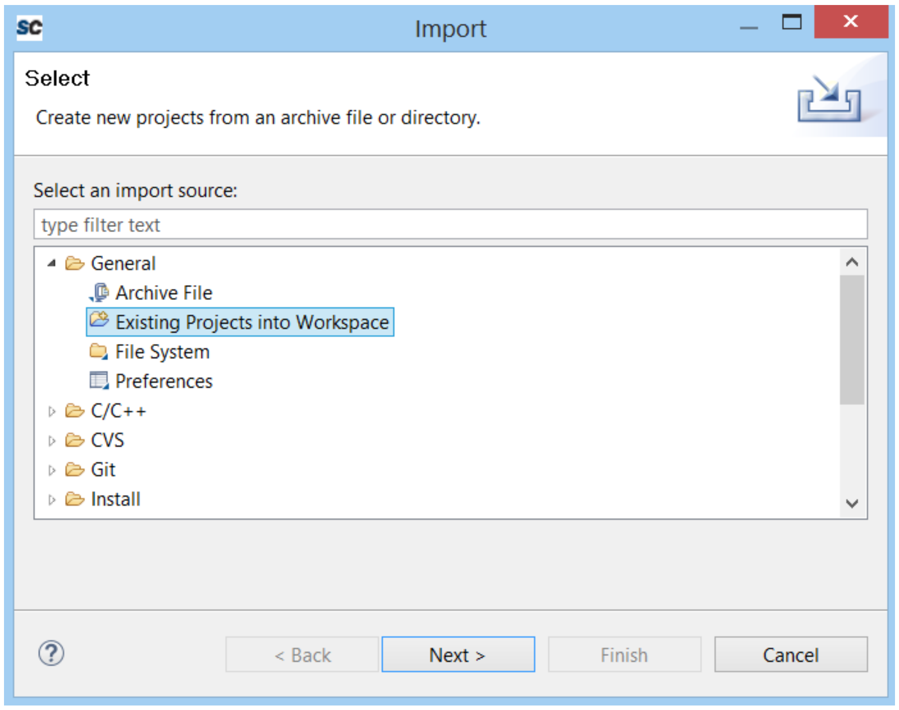
3. In the SoftConsole, go to **File** and select **Import** as shown in the following figure.

Figure 43 • **Import Options**



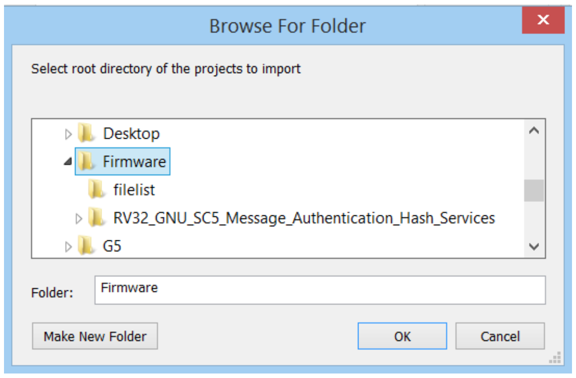
4. Select import source as **Existing Projects into Workspace** and click **Next**, as shown in the following figure.

Figure 44 • **Select An Import Source**



5. In the Import dialog, click **Browse..** to locate the generated sample project in the local PC folder and click **OK**, as shown in the following figure.

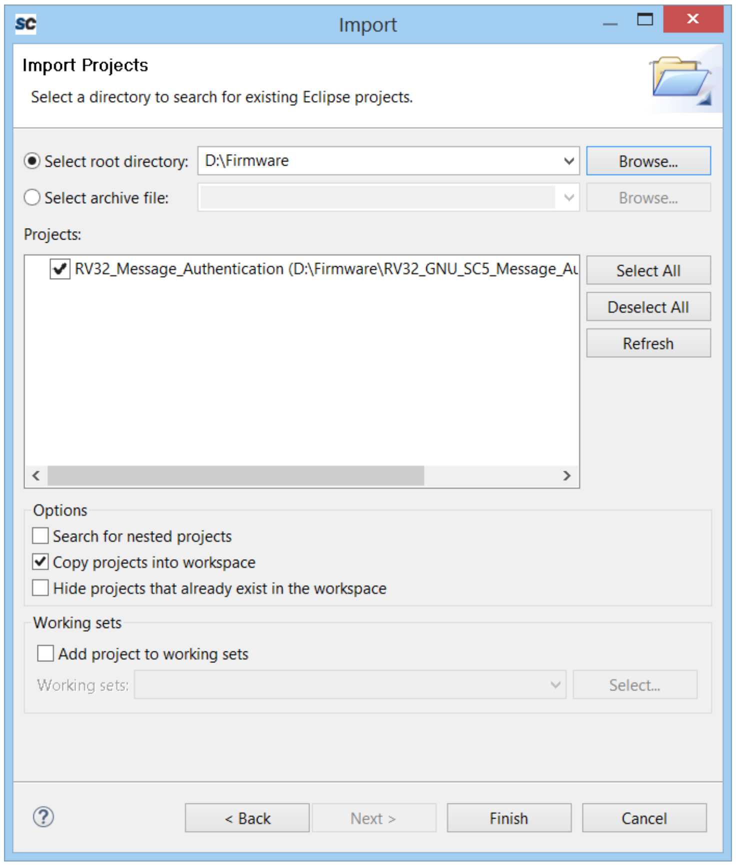
Figure 45 • **Import Dialog**


Browse For Folder
Select root directory of the projects to import
Desktop
Firmware
filelist
RV32_GNU_SC5_Message_Authentication_Hash_Services
G5
Make New Folder
OK
Cancel

Folder: Firmware

6. Ensure that the generated project is selected and click **Finish** to import the generated sample project in a SoftConsole workspace, as shown in the following figure.

Figure 46 • **Importing RV32_Message_Authentication Project**



Select a directory to search for existing Eclipse projects.

(●) Select root directory: D:\Firmware
( ) Select archive file:

Projects:

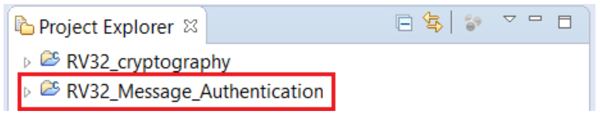
[✓] RV32_Message_Authentication (D:\Firmware\RV32_GNU_SC5_Message_Au

Options
[ ] Search for nested projects
[✓] Copy projects into workspace
[ ] Hide projects that already exist in the workspace

Working sets
[ ] Add project to working sets
Working sets:

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

Figure 47 • **SoftConsole Workspace—Sample Projects**



8. 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 Tera Term Macro Script, page 22 for running the macro script provided in the sample project.

4 Appendix: User Cryptoprocessor Simulation

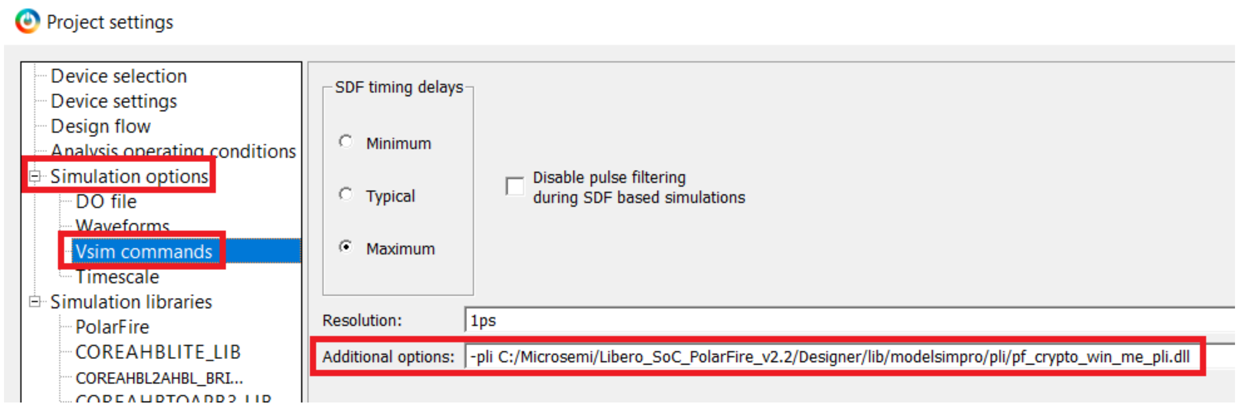
Microsemi Libero SoC PolarFire v2.2 software provides PLI simulation library for the User Cryptoprocessor to show the functional behavior of the User Cryptoprocessor. The PLI library for User Cryptoprocessor is available at *<Libero_Installation_Directory>/Designer/lib/modelsimpro/pli*. The PLI library must be passed to the ModelSim using 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 PolarFire v2.2 on the host PC.

In the following figure, the Libero installation folder is C:/Microsemi/Libero_SoC_PolarFire_v2.2.

Figure 48 • **Libero Project settings—VSIM Command for User Cryptoprocessor Simulation**



Project settings

- Device selection
- Device settings
- Design flow
- Analysis operating conditions
- Simulation options
  - DO file
  - Waveforms
  - Vsim commands
  - Timescale
- Simulation libraries
  - PolarFire
  - COREAHBLITE_LIB
  - COREAHBL2AHBL_BRI...

SDF timing delays: Minimum / Typical / [✓ Maximum]

☐ Disable pulse filtering during SDF based simulations

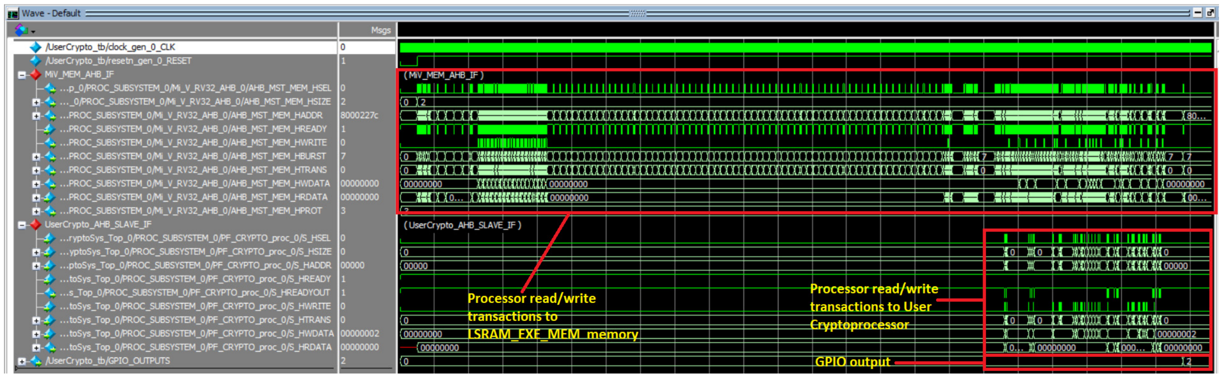
Resolution: 1ps

Additional options: -pli C:/Microsemi/Libero_SoC_PolarFire_v2.2/Designer/lib/modelsimpro/pli/pf_crypto_win_me_pli.dll

The simulation steps include:

1. Generating the top-level component, which includes Mi-V processor system with PF_CRYPTO core in it.
2. Build Mi-V application with required User Cryptoprocessor functions. User Cryptoprocessor functions are accessible through Athena TeraFire CAL driver.
3. Import the Mi-V application hex file into the designated LSRAMs for execution.
4. Simulate the complete processor system to execute the imported application image. 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, as shown in the following figure.

Figure 49 • **User Cryptoprocessor Simulation**



Wave - Default

	Msgs
/UserCrypto_tb/clock_gen_0_CLK	0
/UserCrypto_tb/resetn_gen_0_RESET	1
MiV_MEM_AHB_IF	
...p_0/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HSEL	0
..._0/PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HSIZE	2
...PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HADDR	8000227c
...PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HREADY	1
...PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HWRITE	0
...PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HBURST	7
...PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HTRANS	0
...PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HWDATA	00000000
...PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HRDATA	00000000
...PROC_SUBSYSTEM_0/Mi_V_RV32_AHB_0/AHB_MST_MEM_HPROT	3
UserCrypto_AHB_SLAVE_IF	
...ryptoSys_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HSEL	0
...yptoSys_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HSIZE	0
...ptoSys_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HADDR	00000
...toSys_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HREADY	1
...s_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HREADYOUT	1
...toSys_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HWRITE	0
...toSys_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HTRANS	0
...toSys_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HWDATA	00000002
...toSys_Top_0/PROC_SUBSYSTEM_0/PF_CRYPTO_proc_0/S_HRDATA	00000000
/UserCrypto_tb/GPIO_OUTPUTS	2

(MiV_MEM_AHB_IF)
(UserCrypto_AHB_SLAVE_IF)
Processor read/write transactions to LSRAM_EXE_MEM_memory
Processor read/write transactions to User Cryptoprocessor
GPIO output