AC434
Application Note
Using SRAM PUF System Service in SmartFusion2

Microsemi

a Microchip company

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

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer’s responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided “as is, where is” and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

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

Contents

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 most current publication.

1.1 Revision 5.0

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

- Updated the document for Libero SoC v2021.1.
- Removed the references to Libero version numbers.

1.2 Revision 4.0

Updated the document for Libero SoC v11.7 software release (SAR 77009).

1.3 Revision 3.0

Updated the document for Libero SoC v11.6 software release (SAR 71675).

1.4 Revision 2.0

Updated the document for Libero SoC v11.5 software release (SAR 63969).

1.5 Revision 1.0

The first publication of this document.

2 Using SRAM PUF System Service in SmartFusion2

2.1 Purpose

This application note explains how to use the Static Random-Access Memory (SRAM) Physical Unclonable Functions (PUF) services in the SmartFusion®2 System-on-Chip (SoC) Field Programmable Gate Array (FPGA) devices.

2.2 Introduction

One of the main assumptions in cryptography is that the participants possess a secret key, in order to differentiate themselves from potential attackers. As a consequence, encrypted messages can only be read by the person knowing the key. However, it is not easy to store a secret key. Many devices operate in environments where physical attacks can be applied. By accessing a system, an attacker can easily read the digital key. To guarantee the security of keys, even if an attacker has physical access to a system, PUF is used.

PUFs are based on the internal randomness present in physical systems. If an attacker knows all the details of the system, it is impossible to generate the same key or to clone the device. Another advantage of using a PUF is that additional physical security is achieved without any special manufacturing steps. Moreover, as the process variations are beyond the control of manufacturers, no two systems are equal.

The initial power-up values of SRAM behave randomly and independently due to manufacturing differences such as:

- Thickness of the gate dielectric
- Number of atoms diffused into the channel region
- Other random process variations

SRAM PUF can be used for the generation of keys in any data security application, because of its randomness and device individual fingerprint. The primary advantage of SRAM PUF for generating keys is that the keys are dynamically reconstructed without storing in memory. Keys are generated only when needed on-the-fly. Also, SRAM PUF is easy to implement in hardware. The SRAM PUF generates a device-individual fingerprint using the startup behavior of SRAM. It can serve as a root of trust and provides a key that cannot be easily reverse engineered.

SRAM PUF is unique for each chip as the physical characteristics are unique for each chip, difficult to predict, easy to evaluate, and reliable.

SRAM PUF feature is available in the larger SmartFusion2 devices, such as:

- M2S060TS
- M2S090TS
- M2S150TS

In the M2S060TS, M2S090TS, and M2S150TS devices, SRAM PUF is accessible through system services. The system services are system controller actions initiated by asynchronous events from the ARM® Cortex®-M3 processor or a fabric master in the SmartFusion2 devices.

To use SRAM PUF for key generation, the following steps are involved:

- Enrollment Process: Used for creating an activation code.
- Key Code Generation: Uses activation code and intrinsic or extrinsic keys to generate key codes.
- Key Reconstruction: Uses activation code and key codes to reconstruct the intrinsic or extrinsic keys.

2.2.1 Enrollment Process

The initial step for key generation using SRAM PUF is the enrollment process. The enrollment process creates an activation code using the startup states of the SRAM PUF. The activation code size is 1192 bytes. It is stored in the embedded Non-Volatile Memory (eNVM). The activation code is used for key code generation and key reconstruction. Enrollment is required only for the first time. However, the device can be enrolled multiple times, producing a new activation code for each enrollment. In this case, previous key codes generated using the older activation codes become invalid, making the key reconstruction not possible.

For more information about Enrollment Process, refer to the SRAM-PUF Enrollment Service section in the *UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide*.

2.2.2 Key Code Generation

Key codes are generated using intrinsic or extrinsic keys. Intrinsic keys are automatically generated by the SRAM PUF core. The extrinsic key option allows user to supply the required keys manually for KeyCode generation.

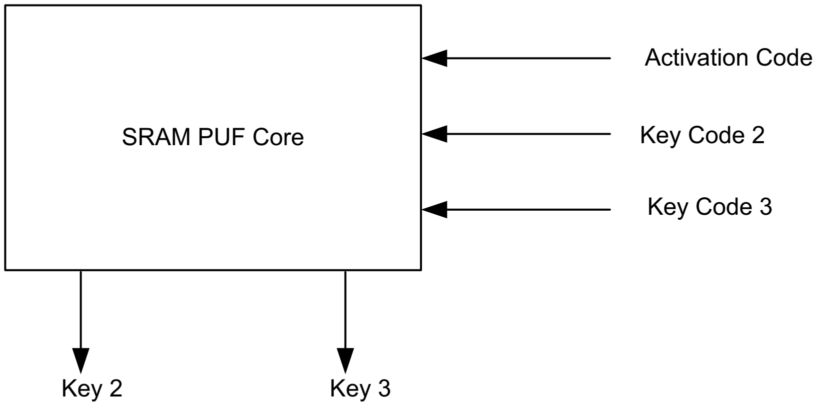
For each intrinsic key or extrinsic key, a key code is generated. Multiple key codes can be generated from multiple intrinsic or extrinsic keys.

2.2.3 Key Reconstruction

The keys have to be reconstructed because they are not stored in memory. To reconstruct the keys, activation code and key codes are required, as shown in Figure 1.

SRAM PUF core reconstructs the original intrinsic or extrinsic keys.

Figure 1 • **Key Reconstruction**



This application note also provides the design example to access the following SRAM-PUF services. For more information on SRAM-PUF services, refer to SRAM PUF Services, page 5.

- Create user AC (Activation Code)
- Delete user AC
- Get number of KC (Key Code)
- Create user KC for an intrinsic key
- Create user KC for an extrinsic key
- Export all KC
- Import all KC
- Delete user KC
- Fetch a user PUF key
- Get a PUF seed

2.2.4 System Controller Block in SmartFusion2 Device

The SRAM PUF services provide access to the system controller's PUF core. SRAM PUF core block is accessed through the communication block (COMM_BLK).

There are two COMM_BLK instances located in:

- Microcontroller Subsystem (MSS)
- System controller

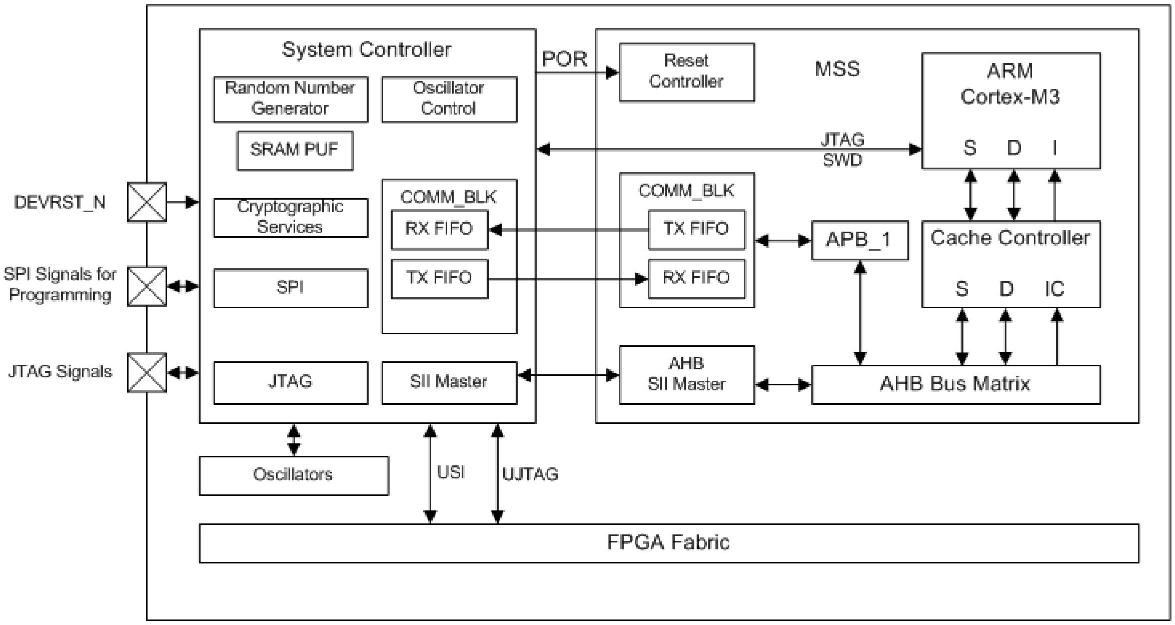
The COMM_BLK consists of an APB interface, eight byte transmit FIFO, and eight byte receive FIFO. The COMM_BLK provides a bi-directional message passing facility between the MSS and the system controller.

The PUF system services are initiated using the COMM_BLK in the MSS, which can be read or written by any master on the AMBA® high performance bus (AHB) matrix; typically either the Cortex-M3 processor or a design in the FPGA fabric (also known as a fabric master).

The system controller receives the command through the COMM_BLK in the system controller. On completion of the requested service, the system controller returns a status message through the COMM_BLK. The responses generated are based on the selected command.

Figure 2 shows the system controller block in the SmartFusion2 device.

Figure 2 • **System Controller Block in SmartFusion2 Device**



For more information about system controller, refer to the *UG0450: SmartFusion2 SoC and IGLOO2 FPGA System Controller User Guide*.

For more information about "COMM_BLK", refer to the "Communication Block" chapter in the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*.

2.2.5 SRAM PUF Services

2.2.5.1 Creating or Deleting the User Activation Code

- CREATE_AC sub command enrolls a new user activation code. The 1192 byte AC is stored in eNVM for future use in generating user keys.
- DELETE_AC sub command deletes AC with all user key codes.

Table 1 shows the command value to create or delete the user activation code and response status.

Table 1 • **Create or Delete the User Activation Code Service Command**

System Service Name	Command Value	Sub Command	Response Status
Create or Delete the User Activation Code	25	0: CREATE_AC 1: DELETE_AC	0: Success completion 1: eNVM MSS error 2: PUF error, when creating 3: Invalid subcmd 4: eNVM program error 7: eNVM verify error 127: HRESP error occurred during MSS transfer 253: License not available in the device 254: Service disabled by factory security 255: Service disabled by user security

2.2.5.2 Create or Delete User Key Code and Export/Import All

This system service is used to generate or delete user key code with the existing activation code from eNVM. It is also used to get the number of keys. User can export and import the generated keycodes using this service.

- GET_NUMBER_OF_KC sub command returns the total number of keys. All keys valid and invalid are counted up to the last valid key. The invalid key is one that got deleted.
- A minimum of two keys are required, (KC#0 and KC#1).
- A maximum of 58 keys are allowed, (KC#0 through KC#57).
- CREATE_INT_KC or CREATE_EXT_KC sub commands generate the new user key code using the existing activation code for an intrinsic and extrinsic key respectively.
- The user key code is stored in eNVM for future use in generating the user keys.
- Supports: 64 bits to 4096 bits key size range.

Note: In this application note, only a 64 bit key size is used.

- EXPORT_ALL_KC sub command export key codes 0 to 57 in encrypted form. The stored user AC and all KCs are first XORed with the one-time pad and copied to a contiguous memory space specified by the user. User must take care of the exported memory space (that is, the size of the memory space can be calculated using the following formula) and the original key memory space. User can only export once.
- The Key Code size (KC_size) depends on the key size and can be calculated with the following formula: KC_size = 96 + ceiling256 (Key Size), ceiling256 () rounds up to the next multiple of 256.
- IMPORT_ALL_KC sub command reads user AC and all KCs from a contiguous memory space used at the time of export.
- The individual private keys are regenerated from the PUF and are copied into the individual memory address spaces defined by the CREATE_EXT_KC or CREATE_INT_KC sub command and stored in private eNVM.
- Import operation can only be successful if an EXPORT operation has been successful earlier.
- DELETE_KC sub command deletes the KC corresponding to the key number provided. User cannot delete keys 0 and 1.

Table 2 shows the command value and response status to Create, Delete User Key Code, and Export/Import All.

Table 2 • **Create or Delete User Key Code and Export/Import All Service Command**

System Service Name	Command Value	Sub Command	Response Status
Create or Delete User Key Code (User KC) and Export or Import all	26	0: GET_NUMBER_OFKC 1: CREATE_EXT_KC 2: CREATE_INT_KC 3: EXPORT_ALL_KC 4: IMPORT_ALL_KC 5: DELETE_KC	0: Success completion 1: eNVM MSS error 2: PUF error, when creating 3: Invalid request or KC, when exporting or importing 4: eNVM program error 5: Invalid hash 6: Invalid user AC 7: eNVM verify error 8: Incorrect key size for renewing a KC 10: Private eNVM user digest mismatch 11: Invalid subcmd 12: DRBG error 127: HRESP error occurred during MSS transfer 253: License not available in device 254: Service disabled by factory security 255: Service disabled by user security

2.2.5.3 Fetch a User PUF Key

Fetch a User PUF Key regenerates the key using the existing activation code and key code located in the eNVM memory. User cannot use this function after EXPORT.

Table 3 shows the command value and response status to fetch a User PUF Key.

Table 3 • **Fetch a User PUF Key service command**

System Service Name	Command Value	Response Status
Fetch a User PUF Key	27	0: Success completion 2: PUF error, when creating 3: Invalid key number or argument or exported or invalid key 5: Invalid hash 10: Private eNVM user digest mismatch 127: HRESP error occurred during MSS transfer 253: License not available in the device 254: Service disabled by factory security 255: Service disabled by user security

2.2.5.4 Get a PUF Seed

Get a PUF Seed generates a 256 bit seed. PUF Seed is a random number that is generated using the SRAM start up values.

Table 4 shows the command value and response status to get a PUF Seed.

Table 4 • **Get a PUF Seed service command**

System Service Name	Command Value	Response Status
Get a PUF Seed	29	0: Success completion 2: PUF error, when creating 127: HRESP error occurred during MSS transfer 253: License not available in device 254: Service disabled by factory security 255: Service disabled by user security

2.3 References

The following documents are referenced in this document. The references complement and help in understanding the relevant Microsemi SmartFusion2 device flows and features.

- *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*
- *UG0450: SmartFusion2 and IGLOO2 FPGA System Controller User Guide*
- *UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide*

2.4 Design Requirements

Table 5 lists the hardware and software design requirements for this demo design.

Table 5 • **Design Requirements**

Requirement	Version
Operating system	64 bit Windows 7 and 10
Hardware	
SmartFusion2 Security Evaluation Kit (M2S090TS): • FlashPro4 programmer • 12 V adapter • USB A to Mini-B cable	Rev D or later
Host PC or Laptop	
Software	
FlashPro Express	Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
Libero® System-on-Chip (SoC)	
SoftConsole	
USB to UART drivers	–
One of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	–

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.5 Prerequisites

Before you begin:

1. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location.

   https://www.microsemi.com/product-directory/design-resources/1750-libero-soc

2. For demo design files download link:

   http://soc.microsemi.com/download/rsc/?f=m2s_ac434_df

2.6 Design Description

The design is implemented on the SmartFusion2 Security Evaluation Kit board using the M2S090TS-1FGG484 device.

The design example consists of:

- RC oscillator
- Fabric CCC
- CORERESET
- MSS

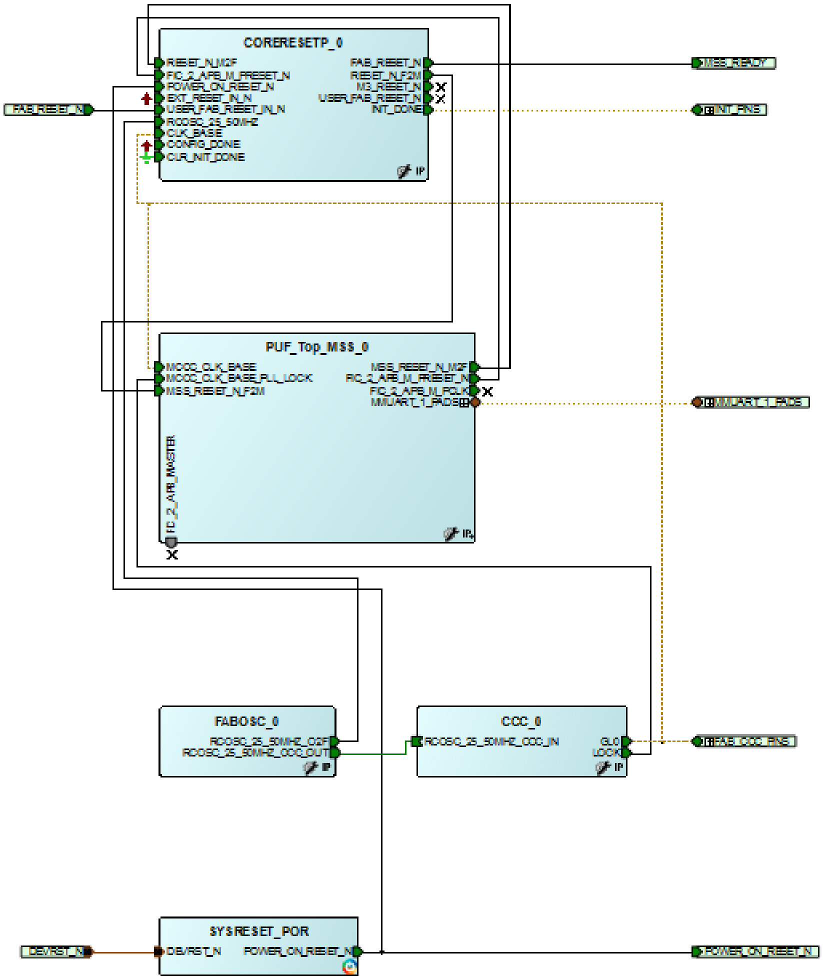
The fabric PLL is used to provide the base clock for the MSS. The system services are run using various C routines in the MSS. Also, a Universal Asynchronous Receiver or Transmitter (UART1) in the MSS is used to display the operation of the PUF system service.

2.7 Hardware Implementation

Figure 3 shows a block diagram of the design example. The RC oscillator generates a 50 MHz input clock and the fabric PLL generates a 100 MHz clock from the RC oscillator. This 100 MHz clock is used as the base clock for the MSS.

The MMUART_1 signals are for communicating with the serial terminal program.

Figure 3 • **Block Diagram of SmartFusion2 SRAM PUF Design Example**



2.8 Software Implementation

The software design example performs the following operations:

- Create or Delete the User Activation Code
- Get Number of the Key Code
- Create or Delete the User Key Code
- Export or Import All Key Codes
- Fetch a User PUF Key
- Fetch a PUF ECC Public Key
- Get a PUF Seed

2.8.1 Firmware Drivers

The following firmware drivers are used in this application:

- MSS MMUART driver: To communicate with serial terminal program on the host PC.
- MSS System Services driver: Provides access to SmartFusion2 system services.
- MSS eNVM driver: Provides access to SmartFusion2 eNVM.

2.8.2 List of APIs

Table 6 shows APIs used in software design to access the SRAM PUF services.

Table 6 • **APIs to Access the PUF System Services**

API	Description
MSS_SYS_puf_create_activation_code()	Create activation code
MSS_SYS_puf_delete_activation_code()	Delete activation code
MSS_SYS_puf_get_number_of_keys()	Returns total number of user keys
MSS_SYS_puf_enroll_key()	Enrolls a new user key code, for an intrinsic and extrinsic key
MSS_SYS_puf_fetch_key()	Retrieve a user PUF key
MSS_SYS_puf_delete_key()	Delete a previously enrolled key
MSS_SYS_puf_export_keycodes()	Export an encrypted copy of all the key codes
MSS_SYS_puf_import_keycodes()	Import a set of PUF key codes that was previously exported
MSS_SYS_puf_get_random_seed()	Generate a 256-bit random seed

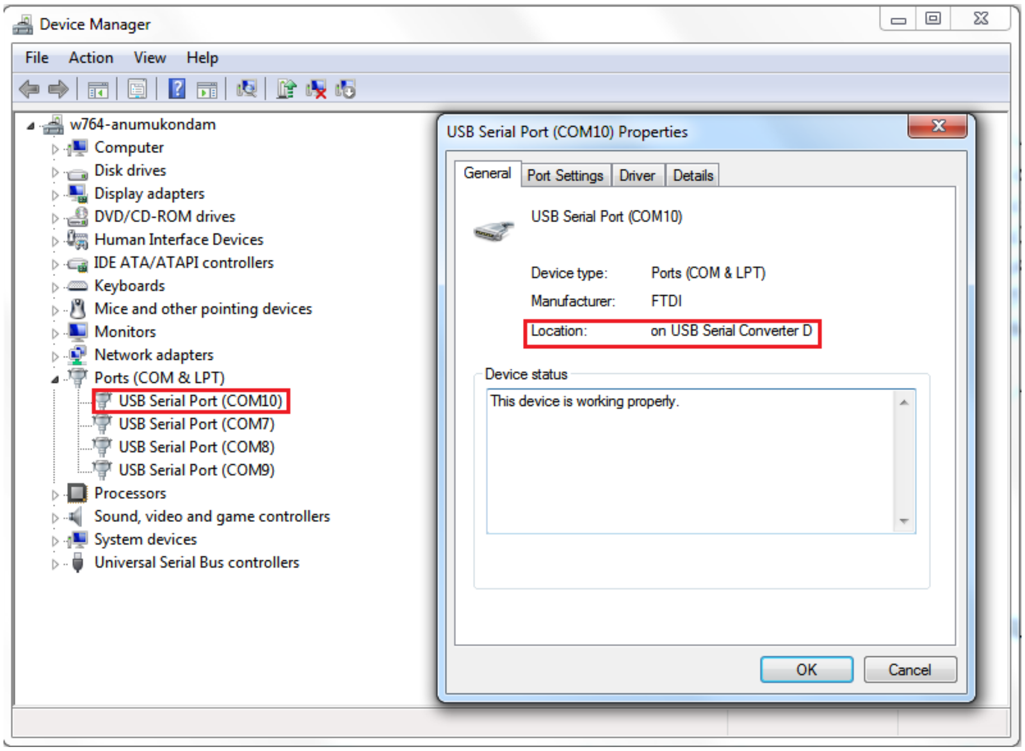
2.9 Setting Up the Design

The following steps describe how to set up the demo design for the SmartFusion2 Security Evaluation Kit Board. Plug the FlashPro4 ribbon cable into the connector J5 (JTAG Programming Header) on the SmartFusion2 Security Evaluation Kit board.

1. Connect the mini USB cable between the FlashPro4 and USB port of the host PC.
2. Connect the power supply to the J6 connector.
3. Connect one end of the USB mini cable to the J18 connector provided on the SmartFusion2 Security Evaluation Kit. Connect the other end of the USB cable to the host PC.
4. Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the Device Manager.

Figure 4 shows example Device Manager window. If USB to UART bridge drivers are not installed, download and install the drivers from: *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip*

Figure 4 • **Device Manager Window**



USB Serial Port (COM10)

Device type: Ports (COM & LPT)

Manufacturer: FTDI

Location: on USB Serial Converter D

Device status

This device is working properly.

5. Connect the jumpers on the SmartFusion2 Security Evaluation Kit, as shown in Table 7.

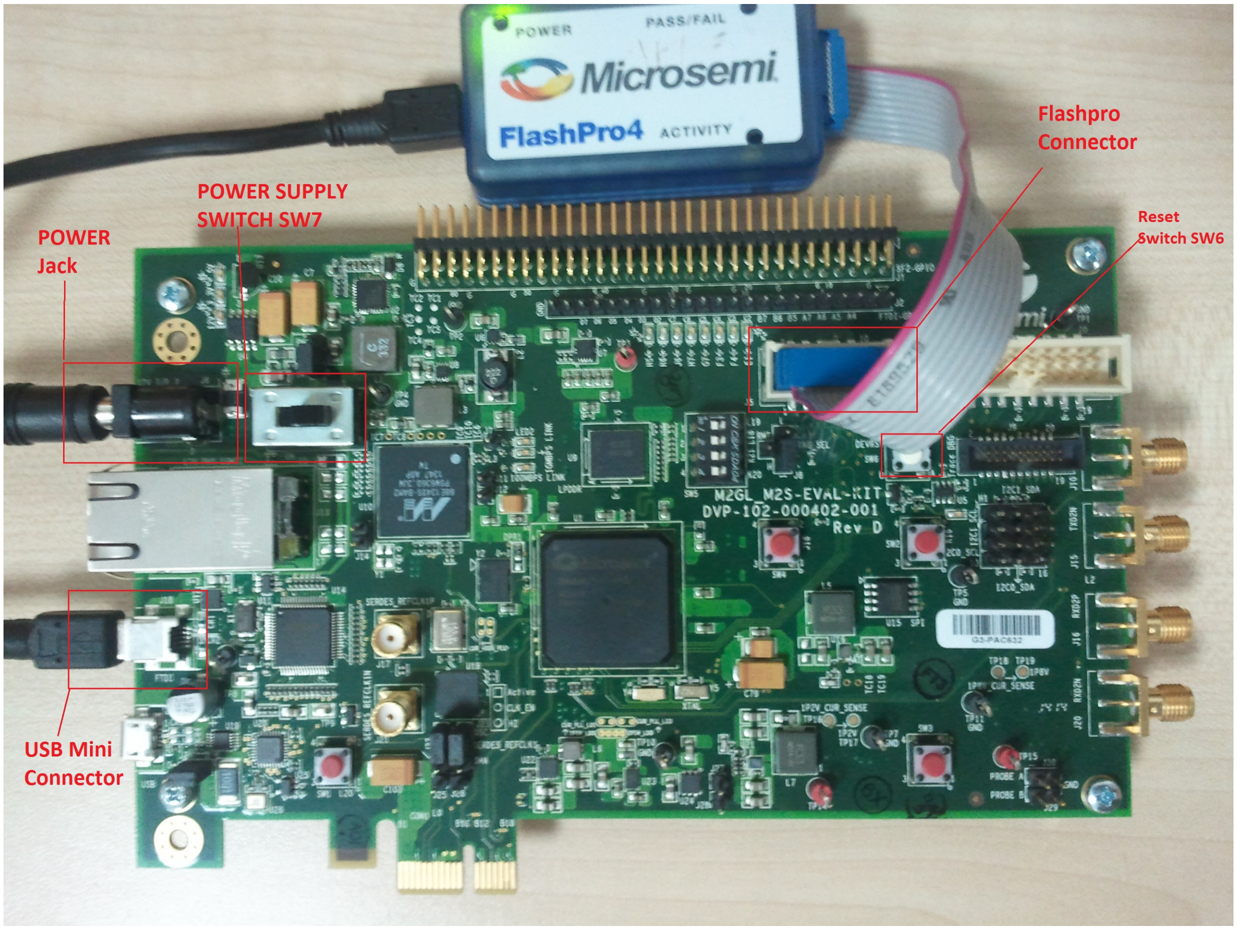
Note: Ensure that power supply switch **SW7** is switched OFF while connecting the jumpers on the SmartFusion2 Security Evaluation Kit.

Table 7 • **SmartFusion2 Security Evaluation Kit Jumper Settings**

Jumper	Pin (From)	Pin (To)	Comments
J22	1	2	Default
J23	1	2	Default
J24	1	2	Default
J8	1	2	Default
J3	1	2	Default

Figure 5 shows the board setup for running the PUF services design on the SmartFusion2 Security Evaluation Kit.

Figure 5 • **SmartFusion2 Security Evaluation Kit**

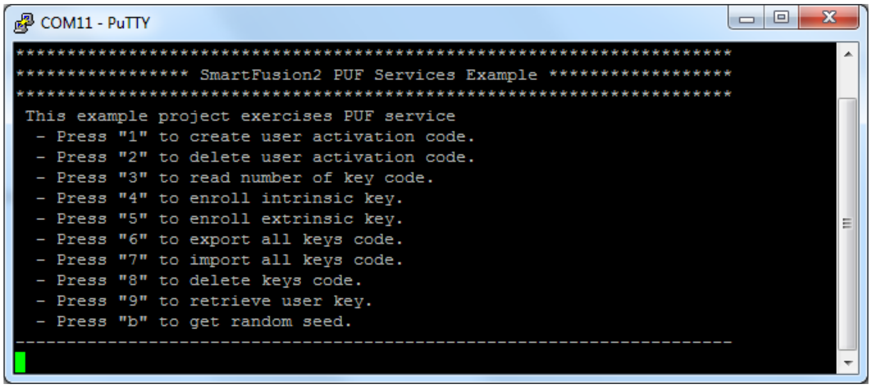


2.10 Running the Design

The following steps describes how to run the design on the SmartFusion2 Security Evaluation Kit board using the M2S090TS-1FGG484 device.

1. Switch ON the power supply switch, **SW7**.
2. Start a PuTTY session with 115200 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. Use any free serial terminal emulation program such as: HyperTerminal or Tera-Term, if the computer does not have the PuTTY program. For more information about configuring HyperTerminal, Tera-Term, or PuTTY, refer to the *Configuring Serial Terminal Emulation Programs Tutorial*.
3. Program the SmartFusion2 Security Evaluation Kit board with the job file provided as part of the design files using FlashPro Express software, refer to Appendix 1: Programming the Device Using FlashPro Express, page 16.
4. After programming, press switch **SW6** (DEVRST), PuTTY displays a message to run the PUF Services, as shown in Figure 6.

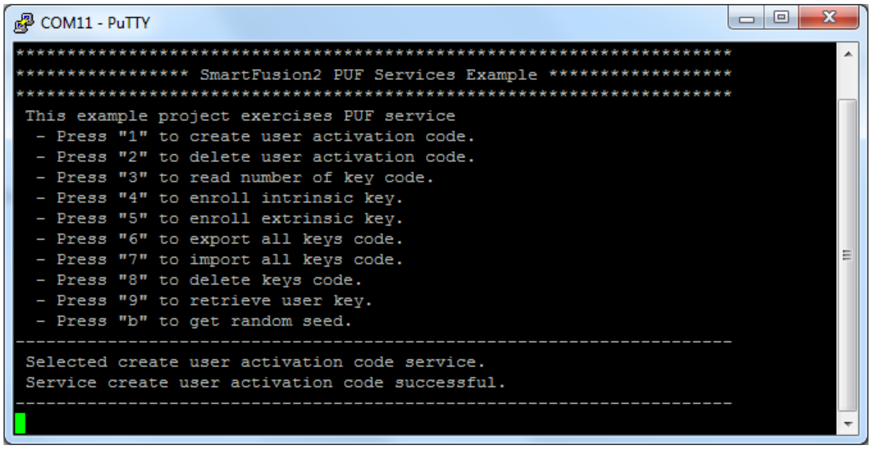
Figure 6 • **Welcome Message**



```
**********************************************************************
**************** SmartFusion2 PUF Services Example ******************
**********************************************************************
 This example project exercises PUF service
  - Press "1" to create user activation code.
  - Press "2" to delete user activation code.
  - Press "3" to read number of key code.
  - Press "4" to enroll intrinsic key.
  - Press "5" to enroll extrinsic key.
  - Press "6" to export all keys code.
  - Press "7" to import all keys code.
  - Press "8" to delete keys code.
  - Press "9" to retrieve user key.
  - Press "b" to get random seed.
----------------------------------------------------------------------
```

5. Enter **1** to enroll new activation code, as shown in Figure 7.

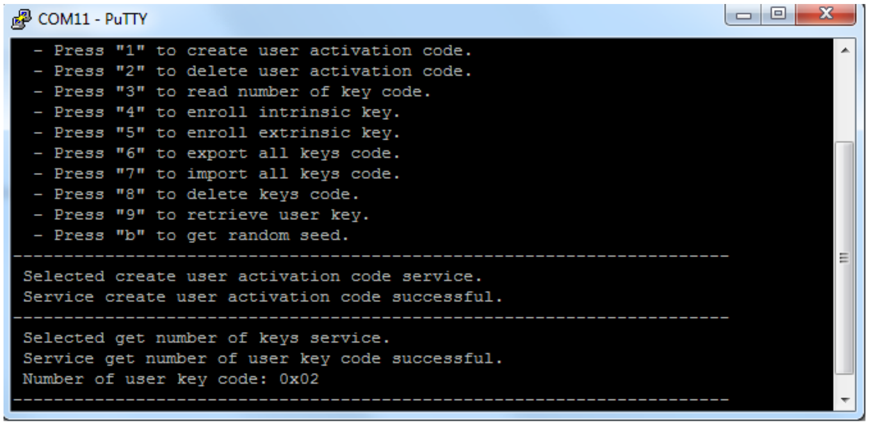
Figure 7 • **Creating the New Activation Code**



```
**********************************************************************
**************** SmartFusion2 PUF Services Example ******************
**********************************************************************
 This example project exercises PUF service
  - Press "1" to create user activation code.
  - Press "2" to delete user activation code.
  - Press "3" to read number of key code.
  - Press "4" to enroll intrinsic key.
  - Press "5" to enroll extrinsic key.
  - Press "6" to export all keys code.
  - Press "7" to import all keys code.
  - Press "8" to delete keys code.
  - Press "9" to retrieve user key.
  - Press "b" to get random seed.
----------------------------------------------------------------------
 Selected create user activation code service.
 Service create user activation code successful.
----------------------------------------------------------------------
```

6. Enter **3** to read number of keys. No of keys is displayed as 2, as shown in Figure 8. These are design security keys: **KC0** and **KC1**.

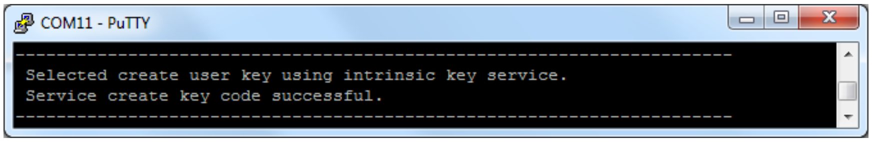
Figure 8 • **Reading the Number of the Key Code**



```
  - Press "1" to create user activation code.
  - Press "2" to delete user activation code.
  - Press "3" to read number of key code.
  - Press "4" to enroll intrinsic key.
  - Press "5" to enroll extrinsic key.
  - Press "6" to export all keys code.
  - Press "7" to import all keys code.
  - Press "8" to delete keys code.
  - Press "9" to retrieve user key.
  - Press "b" to get random seed.
----------------------------------------------------------------------
 Selected create user activation code service.
 Service create user activation code successful.
----------------------------------------------------------------------
 Selected get number of keys service.
 Service get number of user key code successful.
 Number of user key code: 0x02
----------------------------------------------------------------------
```

7. Enter **4** to enroll intrinsic key, as shown in Figure 9.

Figure 9 • **Enrolling an Intrinsic Key**

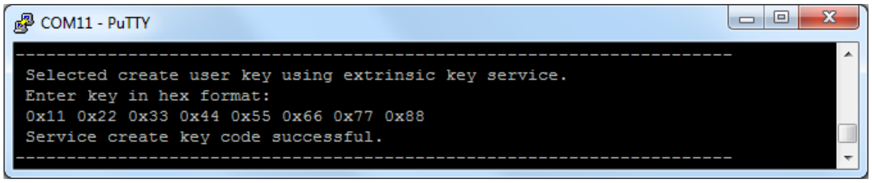


```
----------------------------------------------------------------------
 Selected create user key using intrinsic key service.
 Service create key code successful.
----------------------------------------------------------------------
```

8. Enter **5** to enroll extrinsic key. Enter 64 bit key (1122334455667788), as shown in Figure 10.

Note: PuTTY displays 0×11, if you enter 11.

Figure 10 • **Enrolling an Extrinsic Key**

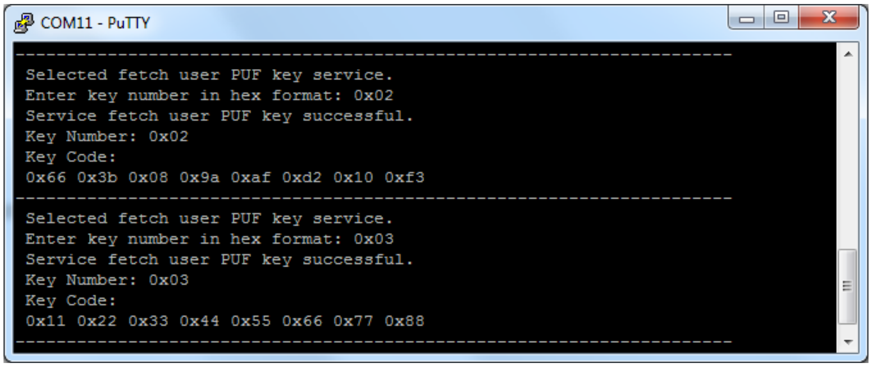


```
-----------------------------------------------------------------------
Selected create user key using extrinsic key service.
Enter key in hex format:
0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
Service create key code successful.
-----------------------------------------------------------------------
```

9. Enter **9** to retrieve the key. Enter Key Numbers, as shown in Figure 11. The Fetch User PUF Key service reconstructs the intrinsic or extrinsic keys, which are used by the user application.

Note: PuTTY displays 0×03, if you enter 03.

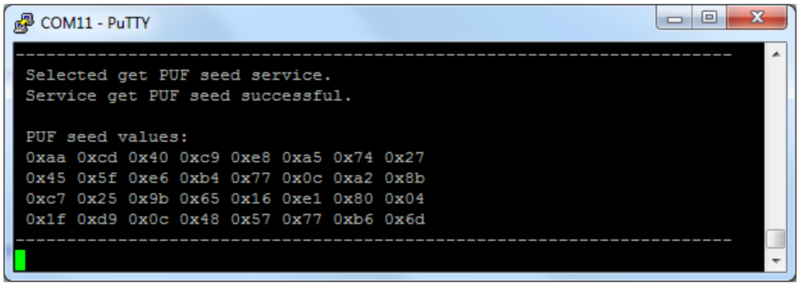
Figure 11 • **Retrieving the Key**



```
-----------------------------------------------------------------------
Selected fetch user PUF key service.
Enter key number in hex format: 0x02
Service fetch user PUF key successful.
Key Number: 0x02
Key Code:
0x66 0x3b 0x08 0x9a 0xaf 0xd2 0x10 0xf3
-----------------------------------------------------------------------
Selected fetch user PUF key service.
Enter key number in hex format: 0x03
Service fetch user PUF key successful.
Key Number: 0x03
Key Code:
0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
-----------------------------------------------------------------------
```

10. Enter **b** to get a PUF seed, as shown in Figure 12.

Figure 12 • **PUF Seed Service**

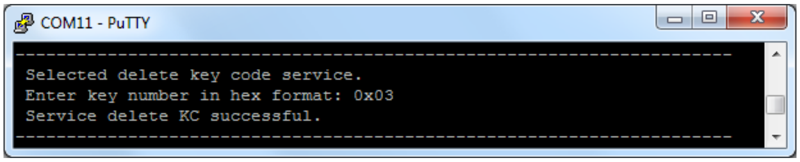


```
-----------------------------------------------------------------------
Selected get PUF seed service.
Service get PUF seed successful.

PUF seed values:
0xaa 0xcd 0x40 0xc9 0xe8 0xa5 0x74 0x27
0x45 0x5f 0xe6 0xb4 0x77 0x0c 0xa2 0x8b
0xc7 0x25 0x9b 0x65 0x16 0xe1 0x80 0x04
0x1f 0xd9 0x0c 0x48 0x57 0x77 0xb6 0x6d
-----------------------------------------------------------------------
```

11. Enter **8** to delete the key. Enter the key number to delete the key, as shown in Figure 13. An error message is displayed, if the user attempts to fetch a key after the deletion of the corresponding key code.

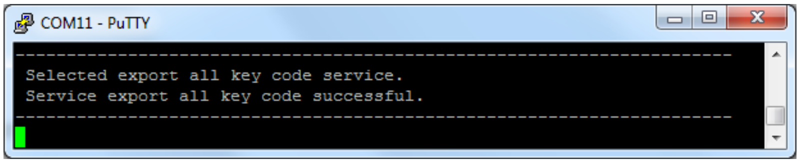
Figure 13 • **Deleting Key Code**



```
-----------------------------------------------------------------------
Selected delete key code service.
Enter key number in hex format: 0x03
Service delete KC successful.
-----------------------------------------------------------------------
```

12. Enter **6** to export all key codes, as shown in Figure 14. All key codes and activation codes are exported to the memory location that was specified by the user in an encrypted format. In case, when the export is unsuccessful, an error message is displayed from the system services.

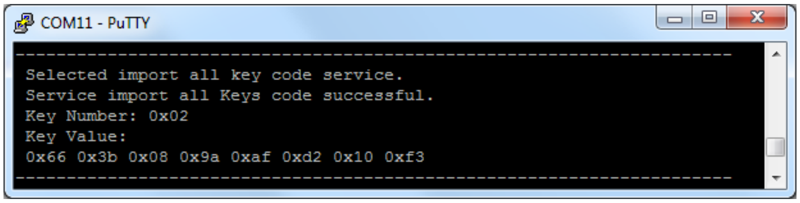
Figure 14 • **Exporting all Key Codes**



```
----------------------------------------------------------------------
 Selected export all key code service.
 Service export all key code successful.
----------------------------------------------------------------------
```

13. Enter **7** to import all key codes, as shown in Figure 15. All key codes and activation codes are imported from the location that was specified at the time of exporting in a decrypted format. In case, when the import is unsuccessful, then an error message is displayed from the system services.

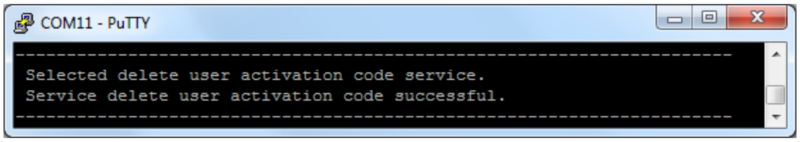
Figure 15 • **Importing all Key Codes**



```
----------------------------------------------------------------------
 Selected import all key code service.
 Service import all Keys code successful.
 Key Number: 0x02
 Key Value:
 0x66 0x3b 0x08 0x9a 0xaf 0xd2 0x10 0xf3
----------------------------------------------------------------------
```

14. Enter **2** to delete activation code, as shown in Figure 16. This command deletes activation code with all the user key codes.

Figure 16 • **Deleting Activation Code**



```
----------------------------------------------------------------------
 Selected delete user activation code service.
 Service delete user activation code successful.
----------------------------------------------------------------------
```

Note: In SmartFusion2 M2S090 and M2S150 devices, the system controller does not release the eNVM1 access after the execution of any of the following SRAM-PUF system services:

- Create user activation code service
- Delete user activation code service
- Create user key code service (for both intrinsic and extrinsic keys)
- Delete user key code service

All the above mentioned system services are executed successfully. However, the eNVM1 becomes inaccessible to other masters such as the Cortex-M3 processor or an FPGA fabric master.

The workaround is to execute the SRAM-PUF system service to read the number of enrolled user keys (GET_NUMBER_OF_KC) immediately after the above-mentioned services to release the eNVM1 access from the system controller.

To request any of the above mentioned SRAM-PUF system services using the Cortex-M3 processor, related firmware code with the above mentioned workaround must be executed only from the eNVM0, eSRAM, or DDR memories. These SRAM-PUF system services cannot be executed from eNVM1 because the Cortex-M3 processor does not have access to the eNVM1 after the execution of any of the above mentioned services. As a result, the Cortex-M3 processor stalls the execution.

2.11 Conclusion

This application note explains how to access the SRAM PUF services in SmartFusion2 devices.

3 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the SmartFusion2 device with the programming job file using FlashPro Express.

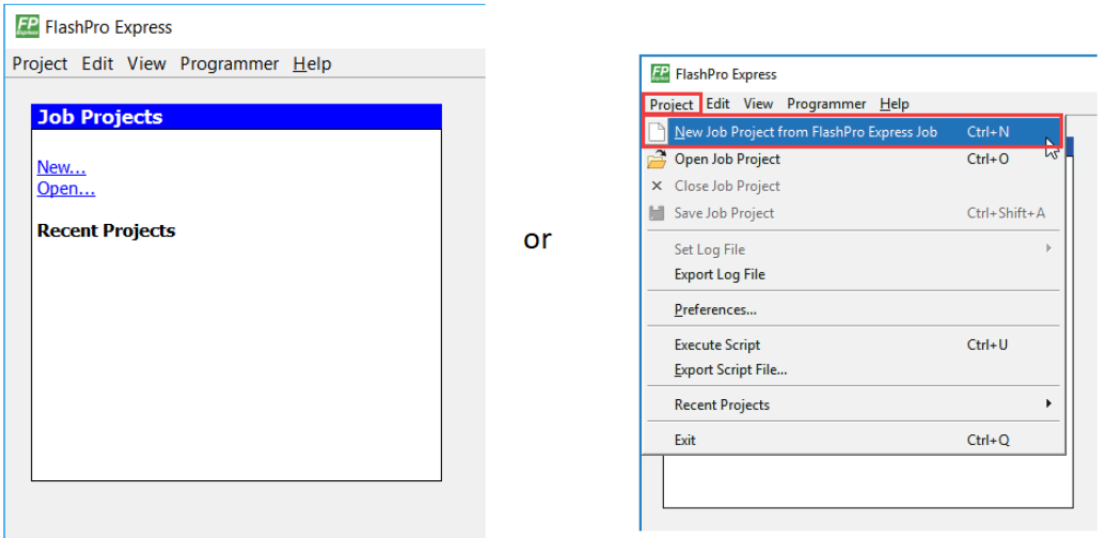
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in Table 7, page 11.

Note: The power supply switch must be switched off while making the jumper connections.

2. Connect the power supply cable to the **J6** connector on the board.
3. Power **ON** the power supply switch **SW7**.
4. On the host PC, launch the **FlashPro Express** software.
5. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in Figure 17.

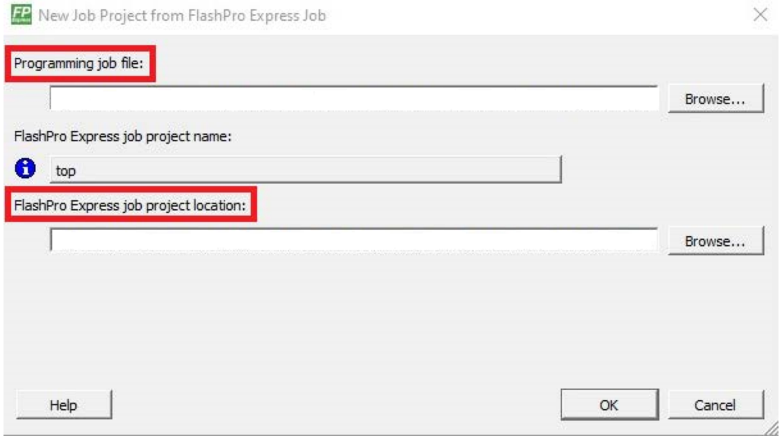
Figure 17 • **FlashPro Express Job Project**



6. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

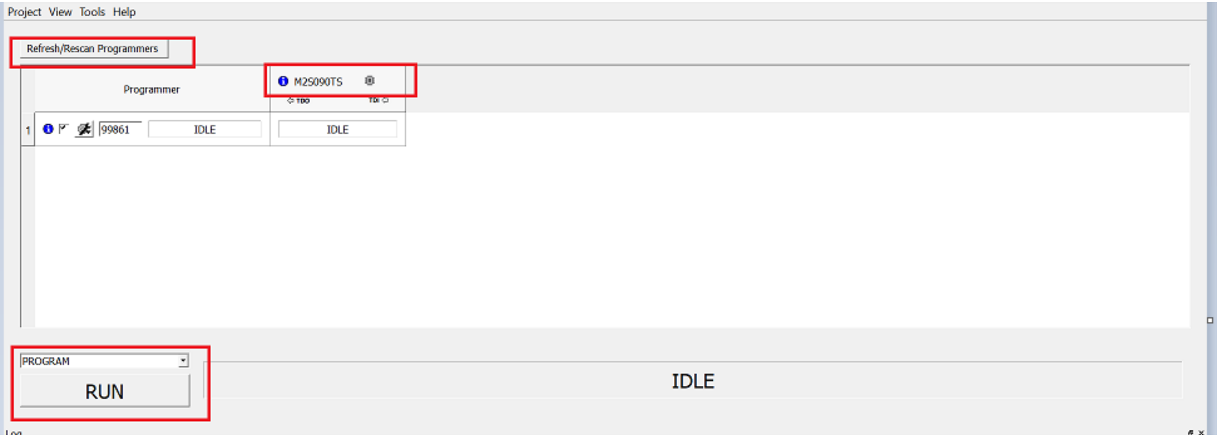
- **Programming job file:** Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is:
  `<download_folder>\m2s_ac434_df\Programming_Job`
- **FlashPro Express job project name:** Click **Browse** and navigate to the location where you want to save the project.

Figure 18 • **New Job Project from FlashPro Express Job**


New Job Project from FlashPro Express Job
Programming job file:
Browse...
FlashPro Express job project name:
top
FlashPro Express job project location:
Browse...
Help
OK
Cancel

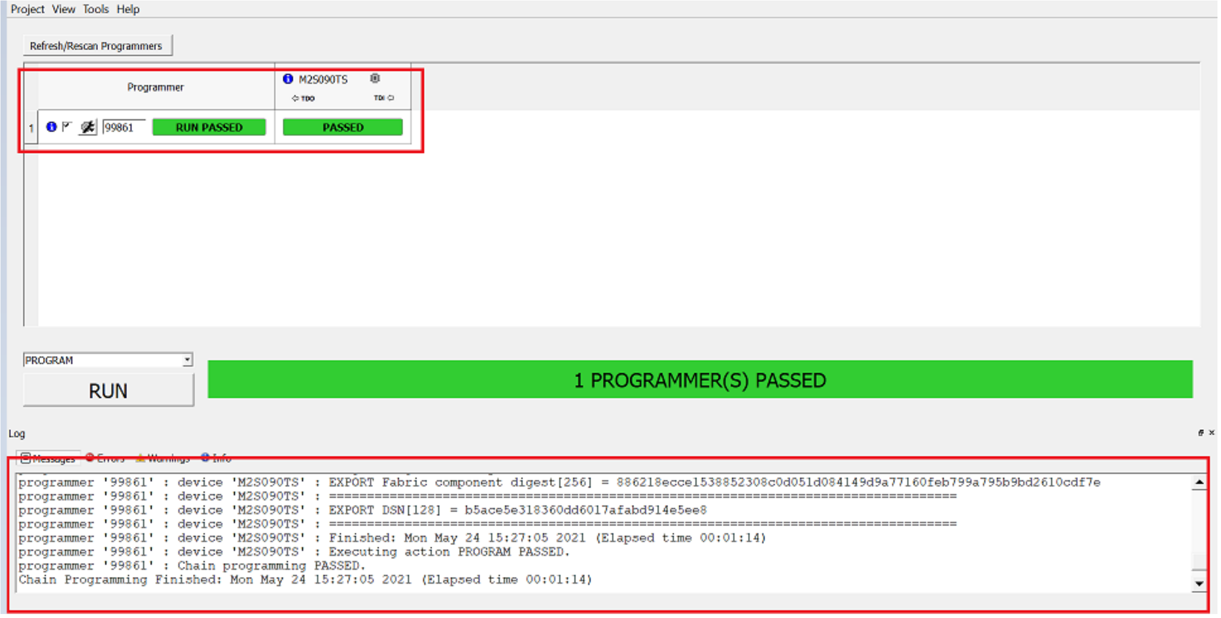
7. Click **OK**. The required programming file is selected and ready to be programmed in the device.
8. The FlashPro Express window appears as shown in Figure 19. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan**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

Figure 19 • **Programming the Device**



9. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in Figure 20.

Figure 20 • **FlashPro Express—RUN PASSED**



```
programmer '99861' : device 'M2S090TS' : EXPORT Fabric component digest[256] = 886218ecce1538852308c0d051d084149d9a77160feb799a795b9bd2610cdf7e
programmer '99861' : device 'M2S090TS' : ===================================================================================
programmer '99861' : device 'M2S090TS' : EXPORT DSN[128] = b5ace5e318360dd6017afabd914e5ee8
programmer '99861' : device 'M2S090TS' : ===================================================================================
programmer '99861' : device 'M2S090TS' : Finished: Mon May 24 15:27:05 2021 (Elapsed time 00:01:14)
programmer '99861' : device 'M2S090TS' : Executing action PROGRAM PASSED.
programmer '99861' : Chain programming PASSED.
Chain Programming Finished: Mon May 24 15:27:05 2021 (Elapsed time 00:01:14)
```

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