TU0823
Tutorial
Secure Production Programming Solution using HSM

Microsemi
Power Matters.™

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

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) 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. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

50200823. 1.0 5/18

Contents

Tables

Figures

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 1.0

The first publication of this document.

2 Secure Production Programming Solution using HSM

2.1 Overview

Microsemi offers Secure Production Programming Solution (SPPS) to prevent overbuilding and cloning of the user designs. For implementing the SPPS, two Hardware Security Modules (HSM) are required. An HSM is a physical computing device connected to a PC via Interfaces like USB or PCIe. HSM protects and manages user key information and enables secured key injection at an untrusted manufacturing facility.

Using Thales e-Security FIPS140-2 level 3 certified HSMs, custom firmware and the state-of-the-art security protocols built into every Microsemi SmartFusion2 SoC FPGA and IGLOO2 FPGA, customers can automatically prevent overbuilding of their systems in any manufacturing facility anywhere in the world, saving millions of dollars in lost revenue.

This tutorial provides step-by-step instructions to use SPPS using tools provided by Microsemi and Thales nShield Edge module as HSM.

2.2 Terms and Definitions

Table 1 • **Terms and Definitions**

Term	Definition
HSM	Hardware Secure Module
U-HSM	User HSM
M-HSM	Manufacturer HSM
CM	Contract Manufacturer
OE	Operation Engineer
DSN	Device Serial Number
DFK-DB	Diversified Factory Key Database
IHP	In-House Programming
SPM	Security Policy Manager

2.3 References

1. SPPS User Guide: *Secure Production Programming Solution (SPPS) User Guide for Libero SoC v11.8 SP1*
2. User HSM Installation and Setup User Guide: *User HSM Installation and Setup User Guide for Libero SoC v11.8 SP1*
3. Manufacturer HSM Installation and Setup User Guide: *Manufacturer HSM Installation and Setup User Guide for Libero SoC v11.8 SP1*
4. Programming Job Manager User Guide: *Programming Job Manager User Guide for Libero SoC v11.8 SP1*
5. FlashPro Express User Guide: *FlashPro Express User Guide for Libero SoC v11.8 SP3*

2.4 Tutorial Requirements

2.4.1 Hardware Requirements

- Thales nShield Edge HSM (Part Number: **MSCNC4031U-10**) or Thales nShield Solo HSM (Part Number: **MSCNC4433E-500**) - 2 Modules
- Microsemi SmartFusion2 Security Evaluation Kits - 3 Kits
- FlashPro4/FlashPro5 Programmer

2.4.2 Software Requirements

- Libero SoC v11.8 SP2
- Programming & Debug v11.8 SP1[1]
- SPPS HSM Servers
  - User HSM Server v11.8 SP1
  - Manufacturer HSM Server v11.8 SP1

2.4.3 Source Files

- Sample SPPS scripts

2.5 Prerequisites

Before you start:

1. Download the design files from the following link:
   http://soc.microsemi.com/download/rsc/?f=m2s_tu0823_liberov11p8_df
2. Download and install Libero SoC v11.8 SP2 from the following location:
   https://www.microsemi.com/product-directory/soc-fpgas/1692-smartfusion2#documentation

It is mandatory for all users to install the required software and setup the HSM modules. For more information, refer:

- *User HSM Installation and Setup User Guide for Libero SoC v11.8 SP1*
- *Manufacturer HSM Installation and Setup User Guide for Libero SoC v11.8 SP1*

1.Programming & Debug Tools are installed automatically with Libero SoC. It is also available for standalone download for convenience (if needed for production programming and lab use). It includes FlashPro Express and Job Manager for secure production programming solution.

3 SPPS Flow

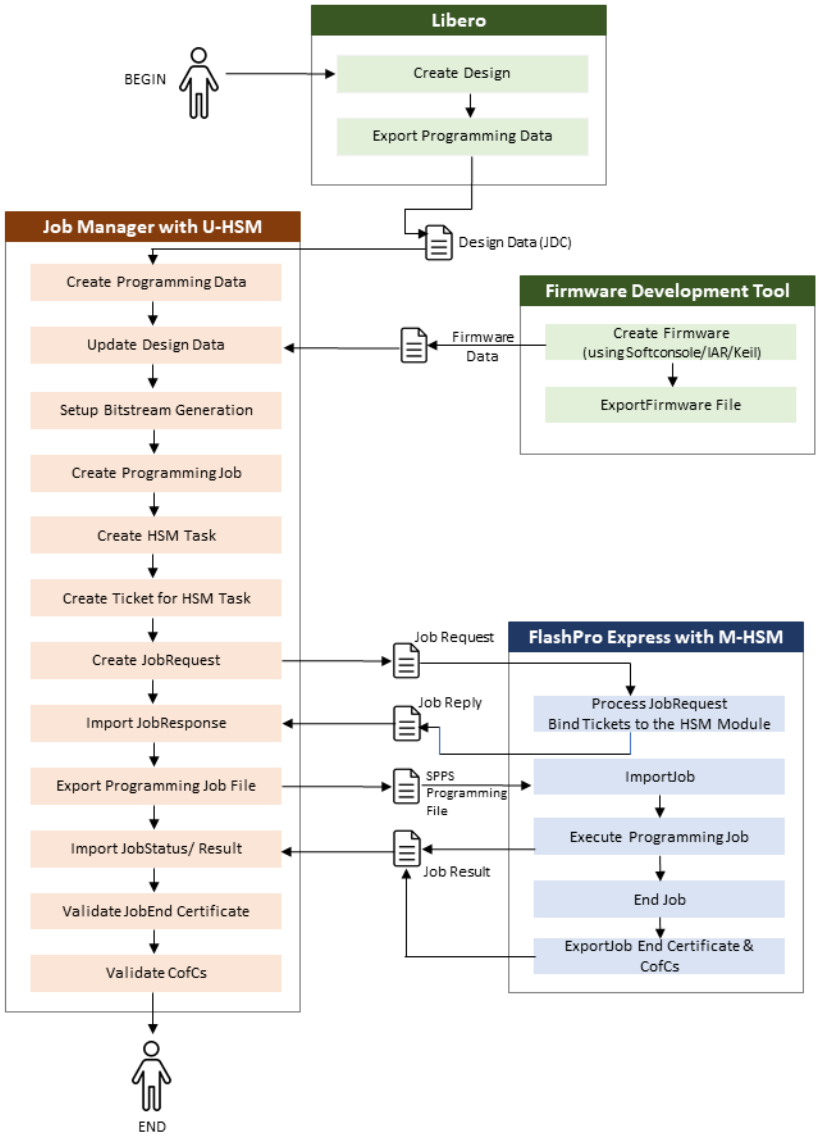
The Secured Production Programming Solution (SPPS) is designed to prevent overbuilding and cloning of the user design. In this solution, the User HSM (U-HSM) is used to generate and protect the user keys and a secured programming job file that can be safely sent to the Contract Manufacturer (CM). The job file contains the encrypted bitstream and job ticket. The job ticket contains the number of devices that the CM is authorized to program.

After receiving secured programming job file, the job must be loaded into the targeted Manufacturing HSM (M-HSM), through FlashPro Express. To program a device, an authorization code specific to the device must be obtained from the M-HSM and sent to the device, before sending in the bitstream. This process enables the M-HSM to track the device being programmed, as well as the number of devices which are programmed. M-HSM will stop issuing authorization codes, once the allowed device number has been reached.

Figure 1, page 10 shows the overall SPPS flowchart. For more detailed information on SPPS flow, refer *Secure Production Programming Solution (SPPS) User Guide for Libero SoC v11.8 SP1.* To understand the flow, this tutorial is split into following steps:

1. Design and Firmware Data Hand-off, page 10
2. U-HSM and M-HSM Parameters Configuration, page 12
3. Keyset Generation, page 14
4. Programming Data Creation and Bitstream Initialization, page 15
5. Programming Job Creation, page 16
6. Job Request-Reply Handshake Protocol, page 17
7. Export HSM Task, page 17
8. Production, page 18
9. HSM Job Completion or Termination, page 22
10. Post-Production, page 23

Figure 1 • **SPPS Flowchart**

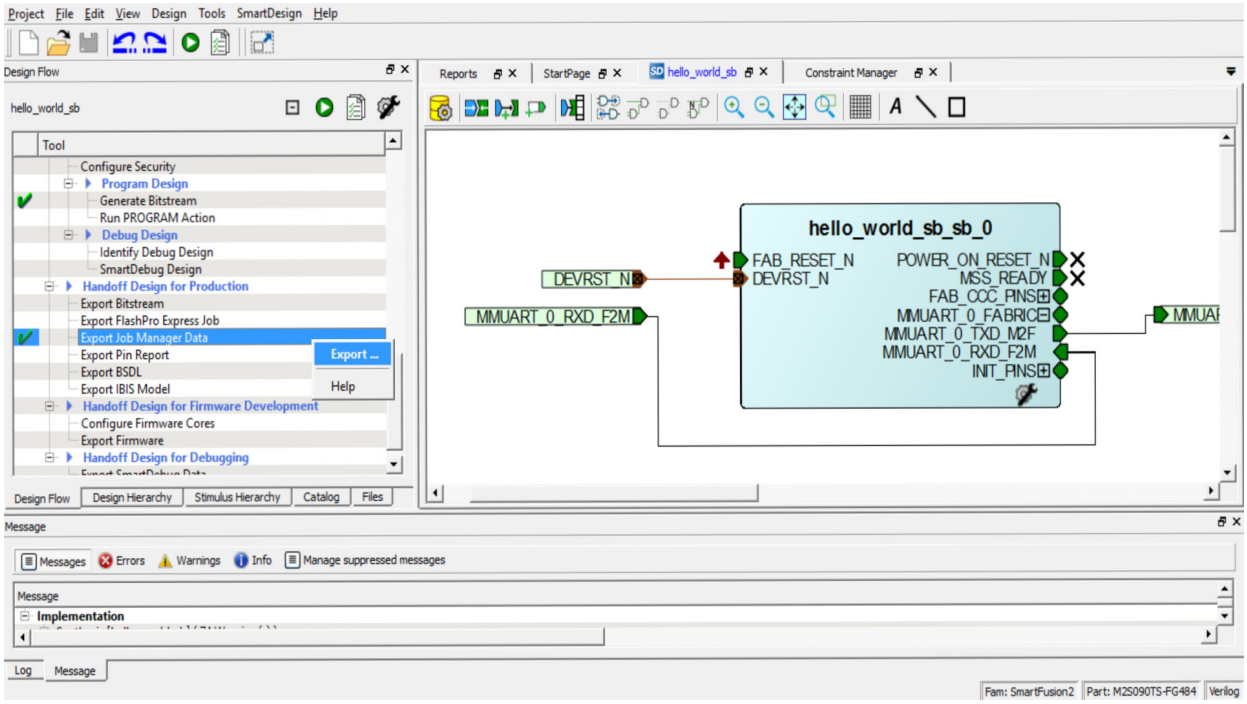


3.1 Design and Firmware Data Hand-off

A design is converted into Bitstream Data and is exported as JDC file to create secure production programming jobs using SPPS flow. The following steps explains design and firmware data hand-off:

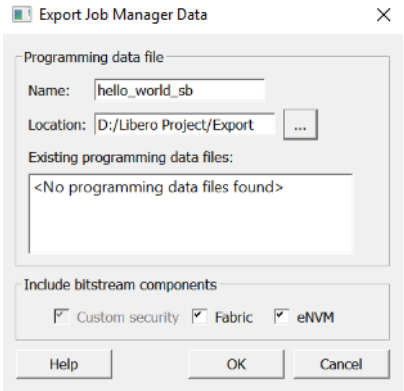
1. Using Libero SoC v11.8 SP2 software, open the Libero design for which the secure programming job needs to be created.
2. Configure design security features using Security Policy Manager. Open the Security Policy Manager by double-clicking on Configure Security in Libero Design Flow window. For more information on Configuration of Security features, refer *Libero SoC User Guide*.
3. In Libero Design Flow console, Right-click on Export Job Manager Data (under Hand-off Design for Production) and select Export option.

Figure 2 • **Libero SoC Console Window**



4. Enter the ***.jdc*** (Job Data Container) file name as shown in Figure 3, page 11 and select the Location by browsing for exporting the file. Bitstream components: Fabric, and eNVM; are given as options. Choose the required option while Exporting the Job Manager Data.

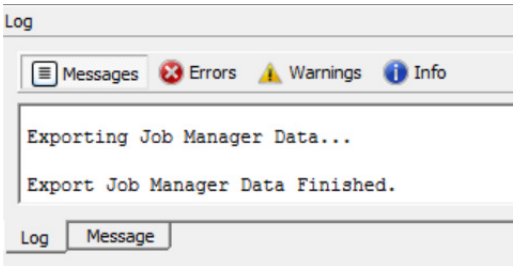
Figure 3 • **Exporting .jdc File**



5. Click OK.
6. Job Manager Data is exported. In the 'Export' folder of the Libero Project Location, a ***.jdc*** file is created.

Figure 4, page 11 display the Log Window with export job manager data finish message.

Figure 4 • **Libero Log displaying Finishing of Export of Job Manager Data**



7. The ***.jdc*** file is transferred to Operations team for creating secure programming job using SPPS flow.

3.2 U-HSM and M-HSM Parameters Configuration

The Job Manager and FlashPro Express tools are part of the SPPS ecosystem. The Job Manager is primarily intended for use by the Operation Engineer (OE), who is responsible for the manufacturing process organization and security. The Job Manager supports the creation of Programming Jobs for the secured production programming flow. The Job Manager supports only TCL scripts.

FlashPro Express is used during production, to program the device from programming jobs created by the Job Manager tool. The Manufacturer HSM (M-HSM) serves FlashPro Express's security protocol requests and enforces the overbuild protection policy and secured key injection. Proof of programming results such as Certificates of Conformance (CofC) generated by programmed devices and job end certifiers, are sent back to the OE and can be validated using the U-HSM.

The Job Manager and FlashPro Express are controlled using TCL script. With this script, the Project is created and the entire flow is run.

Firstly, the exported JDC file and the Job Manager scripts provided in the design files are moved to the workspace directory in OE's host PC. All the FlashPro Express scripts provided in the design files are moved to workspace directory in Manufacturer's host PC.

Note: It is assumed that the U-HSM server and M-HSM server are installed and running on OE's Host PC and Manufacturer's Host PC respectively.

Perform the following operations to set the U-HSM parameters and M-HSM parameters. The U-HSM parameters are set on OE and M-HSM parameters are set on CM.

8. To use Job Manager in the HSM flow, U-HSM parameters must be set. U-HSM configuration data specifies:
   - Name or IP address of the U-HSM server
   - U-HSM UUID assigned by Microsemi
   - U-HSM Master Key UUID
   - Default location of the keyset directory
   - M-HSM UUID assigned by Microsemi

   The `set_hsm_params` TCL command is used to configure HSM. This command saves the HSM parameters for the Job Manager application. This remains in effect until it is overridden using this same command.

9. Create a new Directory and name it *KeySet*.

   Use ***JobManager_Set_HSM_Parameters.tcl*** script as shown in the Figure 5, page 12 to set the U-HSM parameters. Edit this script to set your U-HSM configuration.

Note: All the sample scripts to used by OE and CM are kept in the ***Scripts_For_OE*** and ***Scripts_For_CM*** directories inside the design files respectively.

Figure 5 • **Script for U-HSM Parameter Setting**

```
# Save HSM parameters.
# Need to update parameters with real values before running this TCL command.
set_hsm_params -hsm_server_name {SJSOCPRGW7P1} -u_hsm_uuid {00000000000000000000000000000001}\
    -m_hsm_uuid {00000000000000000000000000000010}\
    -u_master_hsm_uuid {00000000000000000000000000000030}\
    -hsm_keyset_dir {./KeySet}
```

The *KeySet* directory contains the keyset file to be used in the HSM flow. New keyset files can be generated: randomly by the HSM, derived from the existing keyset files, or created from the imported plain text values.

10. Invoke Job Manager (**For Libero Users**: *Libero Installation Directory/Designer/bin/jobmgr.exe* or **For Standalone Tool Users:** *Programming & Debug Tool Installation Directory/bin/jobmgr.exe*) and run the updated ***JobManager_Set_HSM_Parameters.tcl*** script. This script sets the U-HSM Parameters and status can be observed in the generated script report.

Figure 6 • **Executing a Script in Job Manager**

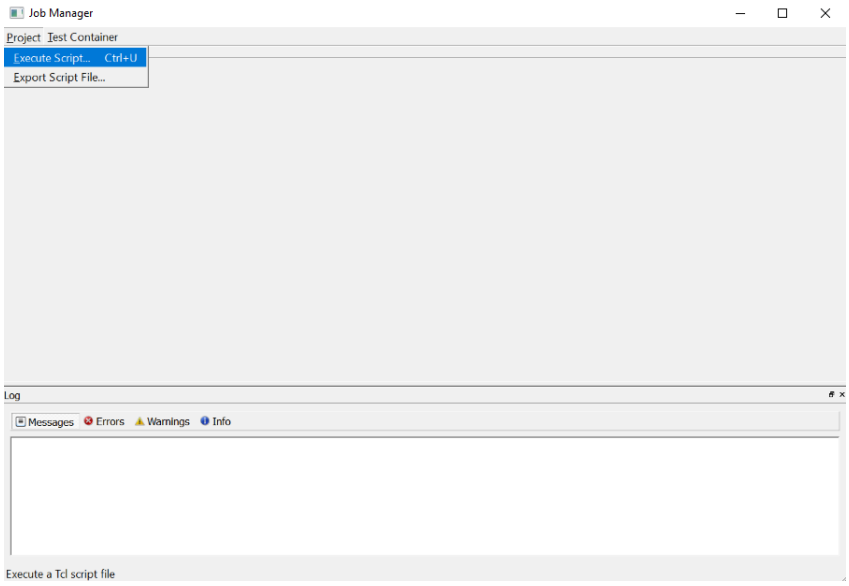
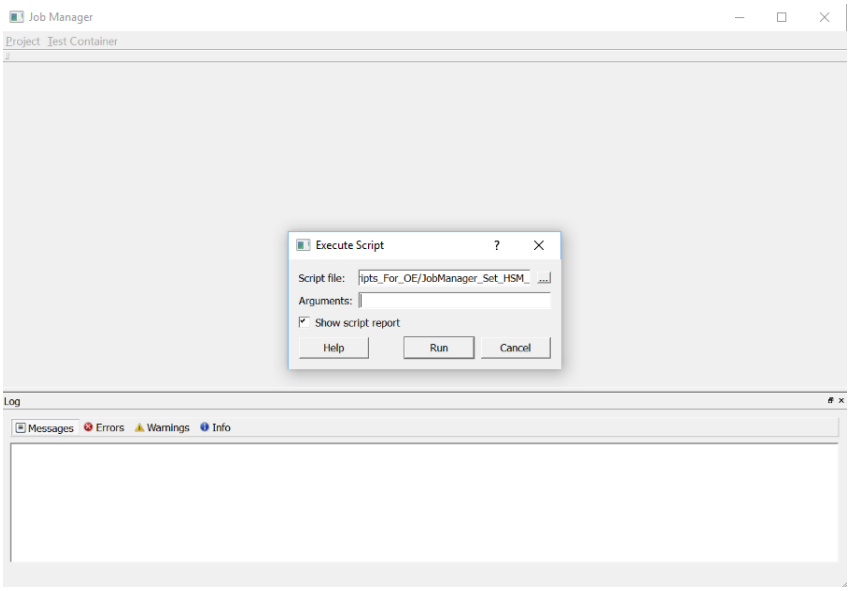


Figure 7 • **Browsing the Script for Execution in Job Manager**



11. To use FlashPro Express in the HSM flow, M-HSM parameters must be set. M-HSM configuration data specifies:
    - Name or IP address of the M-HSM server
    - M-HSM UUID assigned by Microsemi
    - HSM type
        - •TRUE - FlashPro Express will use the Manufacturer features of the User HSM
        - •FALSE - FlashPro Express will use a Manufacturer HSM
    - Username to access the HSM files via FTP server.
    - Password to access the HSM files via FTP server.

The `set_hsm_params` TCL command is used to configure HSM. This command saves the HSM parameters for the FlashPro Express application. This remains in effect until it is overridden using this same command.

Use the ***FlashProExpress_Set_HSM_Parameters.tcl*** script as shown in Figure 8, page 14, to set the M-HSM parameters. Edit this script to set your M-HSM configuration.

Figure 8 • **Script for M-HSM Parameter Setting**

```
# Save HSM parameters.
# Need to update parameters with real values before running this TCL command.
set_hsm_params -hsm_server_name {SJSOCPRGW7E1} -m_hsm_uuid {00000000000000000000000000000000000000010}\
    -hsm_type_u {FALSE} -ftp_username {hsm} -ftp_password {hsm}
```

12. Invoke FlashPro Express from Start menu and run the updated ***FlashProExpress_Set_HSM_Parameters.tcl*** script. This script sets the M-HSM Parameters and status can be observed in the generated script report.

Figure 9 • **Executing a Script in FlashPro Express**

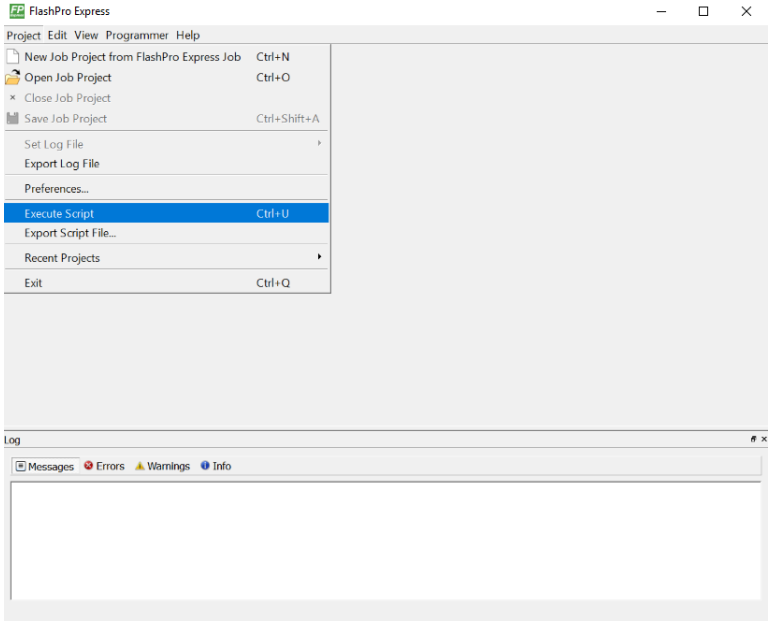
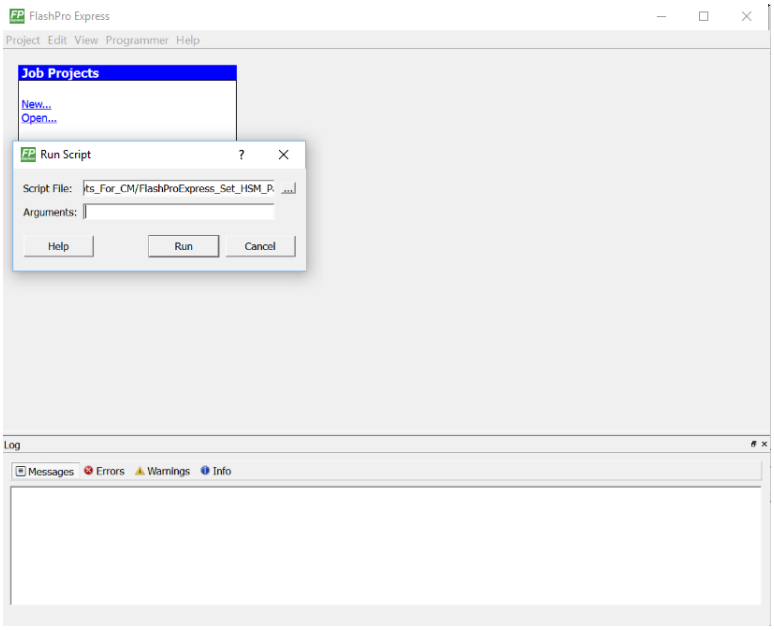


Figure 10 • **Browsing the Script for Execution in FlashPro Express**



3.3 Keyset Generation

In the HSM flow, all keys are protected by the U-HSM. Therefore, in this flow new programming data entry must be associated with a keyset file.

Figure 11, page 15 shows the sample command to generate keyset file:

Figure 11 • **Keyset File Generation**

```
# Create a new key set importing user keys in it.
# Note:
#     1. Import of the key values here is optional. For any key that is key not specified for import,
#        the HSM will use randomly generated value.
create_keyset -file {MyKeySet.txt}    -kip  {F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0}\
                                      -upk1 {A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0A0}\
                                      -uek1 {B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0B0}\
                                      -upk2 {C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0C0}\
                                      -uek2 {D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0D0}\
                                      -dpk  {E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0E0}
```

3.4 Programming Data Creation and Bitstream Initialization

Programming data is created by the OE within a Job Manager project from design data received from Libero via a ***.jdc*** file. Multiple programming data can be created by importing different ***.jdc*** files.

3.4.1 Programming Data Creation via JDC Import

Within the new/opened Job Manager project, a new programming data entry is created using the `new_prog_data` TCL command. While creating a new programming data entry, design data is copied from the JDC file into the current project. After this step, the external ***.jdc*** file can be deleted.

Figure 12, page 15 shows the sample commands to create a new Job Manager project (MyProject), and Figure 13, page 15 shows the sample command for new programming data entry using the JDC file.

Figure 12 • **Script for New Project Creation**

```
# Create new project
new_project -location {.} -name {MyProject}
```

Figure 13 • **Script for Loading Design Data in New Programming Data Entry**

```
# Load design data from the JDC file into new Programming Data entry
# Note: specifying key set parameter here will turn on HSM flow.
new_prog_data -data_name {MyProgData0} -import_file {./hello_world_sb.jdc} -keyset_file {MyKeySet.txt}
```

3.4.2 eNVM Client Modification

This is an optional step that allows the user to modify one or more eNVM clients found in the design, which are loaded into the programming data entry using `set_envm_update` TCL command. To add a client update, the design in the programming data entry must have an eNVM component that already contains the target eNVM client. The size of the update data must be equal to or smaller than the client size in Libero.

Figure 14, page 15 shows the sample commands for eNVM Client Modification.

Figure 14 • **eNVM Client Modification**

```
# Set eNVM update for the client named "MyClient1" - this can be repeated for any number of clients.
# set_envm_update -data_name {MyProgData0} -client_name {MyClient1} -file {./envm_initprog.hex} -overwrite {YES}
```

3.4.3 Initialization of Master Programming Bitstream

The Master programming bitstream is like the Master bitstream used in Libero, but it is based on secured key loading using the Authorization Code Protocol which makes initial key loading secure and provides overbuilding protection. The key mode for the authorization protocol must be set to DFK, KFP or KFPE. The OE configures the Master bitstream to include security and any other optional bitstream components (Fabric, eNVM) in the master bitstream.

Note: KFP and KFPE Authorization Protocols are only available for M2S060, M2S090, and M2S150.

Bitstream initialization is done with the `init_bitstream` TCL command. The sample command shown in Figure 15, page 16, generates a Master bitstream to program security settings and project keys from keyset file when the job file is created. The authorization protocol key mode is set to KFPE.

Figure 15 • **Script for Initialization of Bitstream Generation**

```
# Init master bitstream that uses Auth code protocol to program initial security, eNVM and FABRIC. Generate CoC.
# Note: that this bitstream will program all eNVM clients available in design.
#       If you need to program only specific client(s), use "-envm_clients" option
init_bitstream -data_name {MyProgData0} -bitstream_name {MyBistream} -bitstream_type {MASTER} \
    -features {SECURITY} -use_protocol {AUTH_CODE} -generate_coc -auth_keymode {KFPE}
```

13. Edit the ***JobManager_Create_Project.tcl*** script as per the application requirements and run it using Job Manager to create programming job entry and initialize bitstream.

3.5 Programming Job Creation

A Programming Job is a set of data used by programming systems for device programming in HSM flow. The Programming Jobs created by the Job Manager can be programmed into the device using FlashPro Express or In House Programming (IHP). A Programming Job contains the following data:

- Job type (FlashPro Express or IHP)
- Job origin
- Bitstream(s) for various programming actions (PROGRAM, ERASE, and VERIFY)
- Hardware setup information – for FlashPro Express job type
  - Type of hardware interface (JTAG in this version of the Job Manager)
  - Configuration
- Job device(s) – includes basic device information
- Encrypted Job Tickets authorizing programming actions and overbuild protection under control of the HSM
- Encrypted keys and security protocol data required by HSM protocols

A programming job is created in the Job Manager project using the `new_prog_job` TCL command. After creating a FlashPro Express Programming Job, the user specifies type of the hardware setup. Microsemi devices can be programmed using a bitstream generated by a Programming Data entry using `add_microsemi_prog_device` TCL command. A sample command to create JTAG chain with a single device is shown in the following figure.

Figure 16 • **Script for Programming Job Creation using Job Manager for FlashPro Express**

```
# Create Prog Job for FlashProExpress tool. These steps create JTAG chain with single device and sets MyBitstream to program it.
new_prog_job -job_name {MyProgJob} -job_type {FPExpress}
add_microsemi_prog_device -job_name {MyProgJob} -device_name {MyDevice} -device_hw_location {1}\
    -data_name {MyProgData0} -bitstream_name {MyBistream}
```

Refer *Programming Job Manager User Guide for Libero SoC v11.8 SP1* for more information on `add_microsemi_prog_device` TCL command.

3.5.1 Create HSM Task

The OE can prevent overbuilding by allowing the only specific amount of devices to be programmed. The OE manages this by adding HSM task to a Job.

For each HSM Task, the OE can specify programming actions which the CM is authorized to run and the number of devices the actions can be executed on.

This is accomplished by adding the Job Ticket to every authorized programming action (PROGRAM, ERASE, VERIFY) for the target device.

The Job Ticket is created as per the user-selected programming action. This allows the user to separately manage those actions. A new Job Ticket is created with the `new_hsmtask_ticket` TCL command. The `max_device` parameter is used to limit the number of devices a programming action can be executed on. If the `max_device` parameter is set to 'unlimited', overbuild protection is disabled.

Figure 17, page 17 shows the sample commands for HSM task creation and Job Tickets for programming actions for the target device.

Figure 17 • **Script for Task Creation and Job Tickets**

```
# Create HSM task with tickets to execute programming job. Same job can be executed by multiple CMs,
# if needed via separate HSM Tasks.
#add_hsmtask_to_job -job_name {MyProgJob} -hsmtask_name {MyHSMTask} -cm_request_type {INTERNAL}
add_hsmtask_to_job -job_name {MyProgJob} -hsmtask_name {MyHSMTask}

# Create ticket for each bitstream action: tickets contain encrypted keys and protocol data including optinal overbuild protection.
new_hsmtask_ticket -job_name {MyProgJob} -hsmtask_name {MyHSMTask} -ticket_name {MyTicket1} -device_name {MyDevice} -actions {PROGRAM}\
    -max_device {2}
new_hsmtask_ticket -job_name {MyProgJob} -hsmtask_name {MyHSMTask} -ticket_name {MyTicket2} -device_name {MyDevice} -actions {VERIFY}\
    -max_device {2}
new_hsmtask_ticket -job_name {MyProgJob} -hsmtask_name {MyHSMTask} -ticket_name {MyTicket3} -device_name {MyDevice} -actions {ERASE}\
    -max_device {2}
```

14. Edit the ***JobManager_Create_HSM_Task.tcl*** script per application requirements and run it using the Job Manager to create HSM task with tickets.

3.6 Job Request-Reply Handshake Protocol

The OE executes the Job Request-Reply handshake protocol between the U-HSM and the M-HSM to bind Job Tickets of the HSM Task to the specific physical M-HSM. This process ensures that the Job can only be executed by one HSM only, to ensure effective overbuild protection.

15. The OE exports the file with the Job Request from the Job Manager.
    Run the ***JobManager_Create_HSM_Job_Request.tcl*** script as shown in Figure 18, page 17 using the Job Manager to export the Job Request.

Figure 18 • **Script for Exporting Job Request**

```
# Export job request file and sent it to CM.
hsmtask_m_request -job_name {MyProgJob} -hsmtask_name {MyHSMTask} -request_file {./MyProgJobRequest.req}

# Close project
close_project -save {TRUE}

# END OF SCRIPT: load created request file in FlashProExpress and bring back generated job responce
```

16. The OE passes this Job Request to the Contract Manufacturer (CM). The OE and CM agree on the transport type for delivering the Job Request, depending on their specific case and security policies.
17. The CM loads the received Job Request with FlashPro Express & M-HSM, and exports a Job Reply with ticket IDs and HSM binding information for Job Manager.
    Run the ***FlashProExpress_Process_Job_Request.tcl*** script as shown in Figure 19, page 17 using the FlashPro Express to export Job Reply.

Figure 19 • **Script to Process Job Request**

```
# FlashProExpress serves job request: generates ticket IDs and HSM binding information for JobManager
process_job_request -request_file {./MyProgJobRequest.req} -reply_file {./MyProgJobReply.rep} -overwrite_reply {TRUE}
save_log -file {./MyProgJobRequestResponce.log}
```

18. The CM sends the Job Reply file back to the OE.
19. The OE imports the received Job Reply into the HSM Task on the Job Manager side.
    Run the ***JobManager_Import_Job_Reply.tcl*** script as shown in Figure 20, page 17 using the Job Manager. After performing this handshake protocol, the HSM Job can be exported from this HSM Task.

Figure 20 • **Script for Importing Job Reply**

```
open_project -project {./MyProject/MyProject.jprj}

# Import Job Replay with ticket IDs and HSM binding data
hsmtask_m_reply -job_name {MyProgJob} -hsmtask_name {MyHSMTask} -reply_file {./MyProgJobReply.rep}

close_project -save {TRUE}
```

3.7 Export HSM Task

An HSM Task (SPPS programming job) can be exported with the `export_hsmtask` TCL command. The HSM job can only be exported after importing the Job Reply. The HSM job can only be executed on the physical M-HSM that was used to generate the Job Reply.

20. Run the ***JobManager_Export_HSM_Task.tcl*** script as shown in Figure 21, page 18 using the Job Manager to export the HSM task.

Figure 21 • **Script for Exporting HSM Task**

```
open_project -project {./MyProject/MyProject.jprj}

# Exporting HSM Task
export_hsmtask -job_name {MyProgJob} -hsmtask_name {MyHSMTask} -location {./} -name {MyProgJob}

close_project -save {TRUE}
```

21. The OE passes the exported Job to the CM.

3.8 Production

The CM receives the SPPS Programming Job file from the OE and creates a FlashPro Express project with the HSM Job in the SPPS Programming Job file. For the HSM job, some ticket information, such as ticket IDs and overbuild protection data, gets loaded into the specific M-HSM module that participated in the Job Request-Reply handshake protocol.

22. Connect the **SmartFusion 2** evaluation board to a FlashPro device before the programming actions are executed.
23. Run the script ***FlashProExpress_Create_Job_Project.tcl*** script as shown in Figure 22, page 18 to create a FlashPro Express project from the Job file received from OE.

Figure 22 • **Script for Creating FlashPro Express Project from the Job File**

```
# Create FlashProExpress project from the JOB file.
# This will create a folder with same name as job_file name at the specified location.
# This will import tickets into the CM HSM
create_job_project \
        -job_project_location {.} \
        -job_file {./MyProgJob.job} \
        -overwrite 0

# Run Program/Verify/Erase, etc..
# ...
```

On successful execution of the script, a Programming Job folder is created as shown in Figure 23, page 18 and Programming Window is displayed in the FlashPro Express as shown in Figure 24, page 19.

Figure 23 • **Programming Job Folder (created on script execution)**

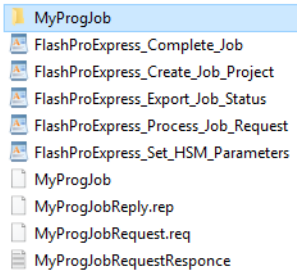
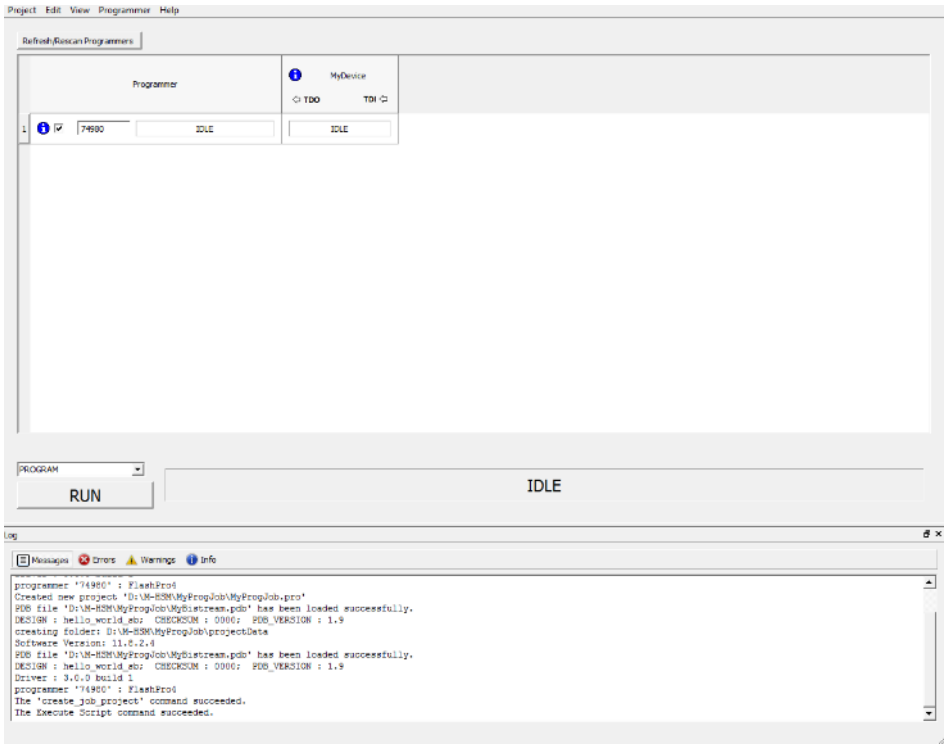
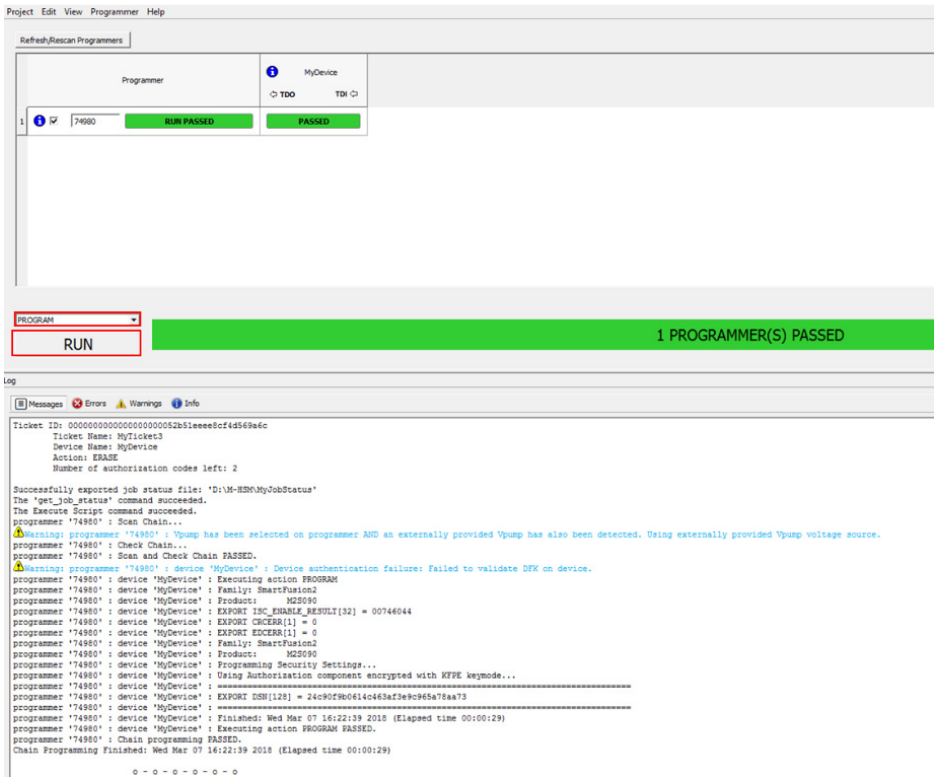


Figure 24 • **Programming Window**



24. Choose a Programming Option: **PROGRAM** and Click RUN (as shown in Figure 25, page 19).

Figure 25 • **Programming Window: PROGRAM option being RUN**



25. Job status can be requested in FlashPro Express during job execution. Job status can be printed out in the FlashPro Express log window or exported into the Job Status file and sent back to the OE.

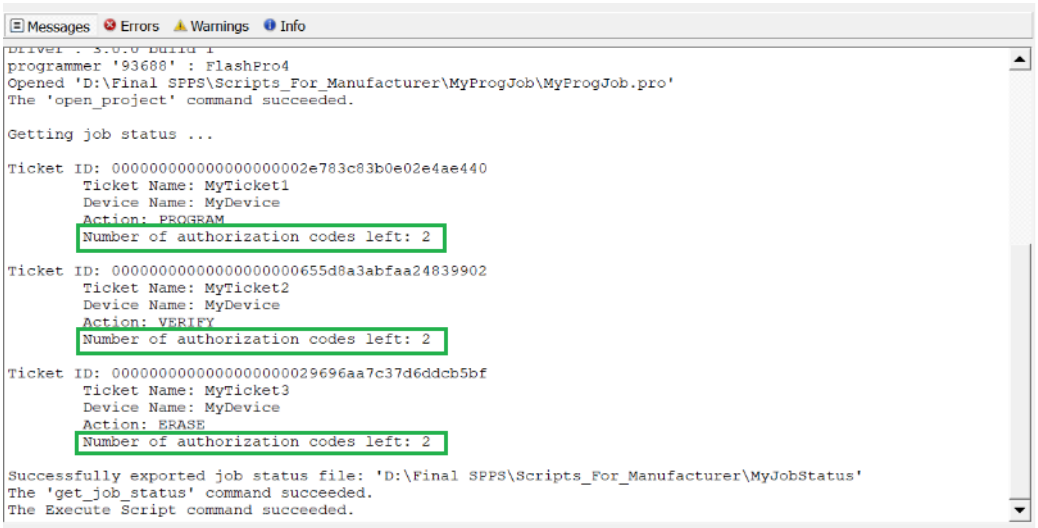
Run ***FlashProExpress_Export_Job_Status.tcl*** script as shown in Figure 26, page 20 in FlashPro Express to know the number of remaining authorization codes for various programming actions.

Figure 26 • **Script for Exporting Job Status**

```
# *optional step*: obtain intermediate job status to be sent to JobManger
# FlashProExpress project must be loaded first...
open_project -project {./MyProgJob/MyProgJob.pro}
get_job_status -job_status_file {./MyJobStatus}

# Load exported status file into JobManager to see job status..
```

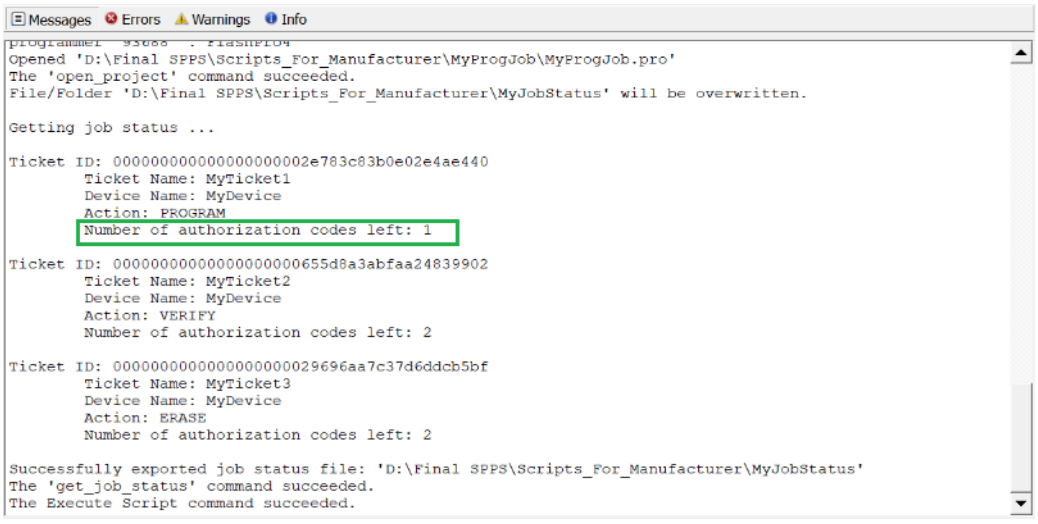
26. On successful execution of the script, the Job Status is displayed in the Message Box of FlashPro Express and exported as Job Status file.
    - Job Status before executing the Program (as shown in Figure 27, page 20):

Figure 27 • **Log - Before executing the Program**



- Job Status after executing the Program (as shown in Figure 28, page 20):

Figure 28 • **Log - After executing the Program**

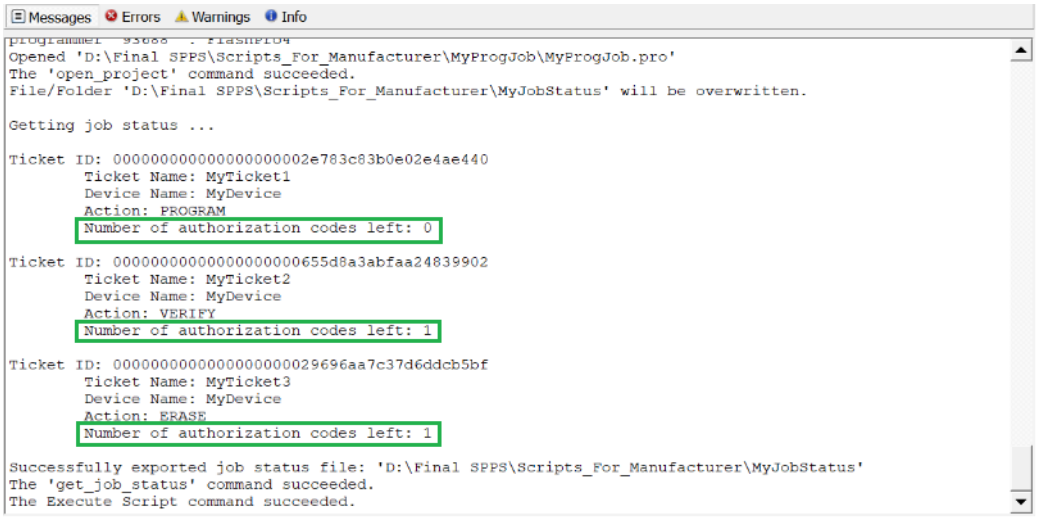


- Connect another **SmartFusion 2** evaluation board and Program it. Figure 29, page 21 shows the Job Status after executing the Program in the new board.

Figure 29 • **Log - After executing the Program in the new board**

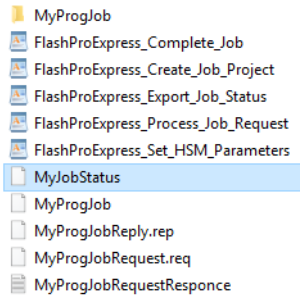
- Job Status after Verifying and Erasing actions (as shown in Figure 30, page 21):

Figure 30 • **Log - After executing Verify and Erase**



27. The Job Status file (as shown in Figure 31, page 21) can be transferred to OE for Intermediate Job Status check.

Figure 31 • ***MyJobStatus*** **File**



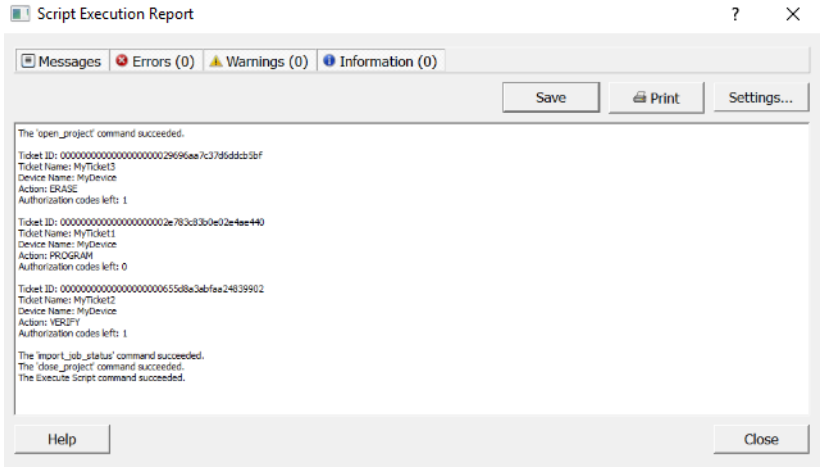
28. The OE can import the Job Status File and check the status of programming job.
Run ***JobManager_Check_Job_Status.tcl*** script as shown in Figure 32, page 22 in Job Manager to check the Intermediate Job Status.

Figure 32 • **Script for Job Status Check**

```
# *Optional Step* View job status recieved from FlashProExpress

# Open JobManager project
open_project -project {./MyProject/MyProject.jprj}

# read job status recieved from FlashProExpress..
import_job_status -job_status_file {./MyJobStatus}

close_project -save {TRUE}
```

On successful execution of the script, the Intermediate Job Status is displayed in the Message Box of the Job Manager as shown in Figure 33, page 22.

Figure 33 • **Job Status in Message Window of Job Manager**



3.9 HSM Job Completion or Termination

Job execution can be stopped in two ways:

- Normal job ending upon finishing programming of the target number of devices. For example, all job ticket overbuild protection counters are exhausted
- Job termination - If some job ticket overbuild protection counters still have devices to program.

Both types of job termination will do the following:

- Remove job tickets from the M-HSM and archive them in a dedicated folder on the M-HSM.
- Generate Job Status with the following data:
  - Device Certificates of Conformance (CofC) generated by devices. The OE can validate the CofC using U-HSM to ensure that only intended contents are programmed into the device.
  - Ticket End certifiers. This is a cryptographically validated proof of removing ticket data from the M-HSM module. Validation of this data can be done by the Job Manager using the U-HSM. Once the Job ticket is removed, the job ticket can no longer be used by the M-HSM.

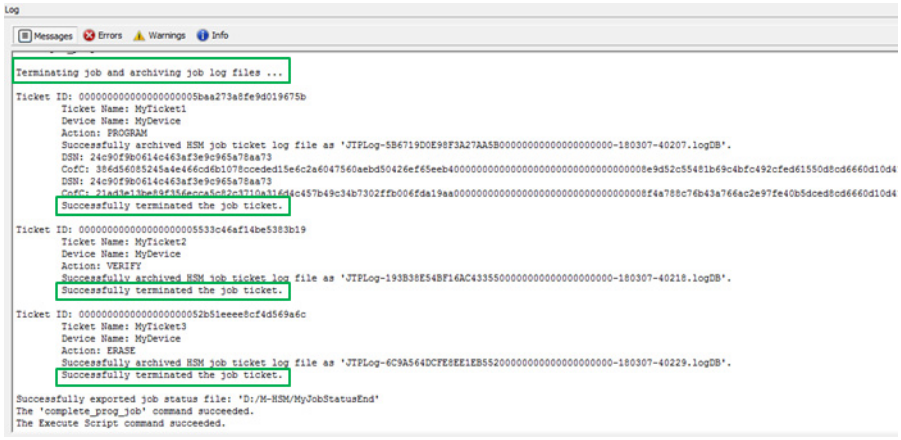
29. The job can be terminated using `complete_prog_job` command. It exports ticket information to a *JobStatusEnd* file. Use `-terminate` option to terminate the incomplete job.
    Run ***FlashProExpress_Complete_Job.tcl*** script as shown in Figure 35, page 23 in FlashPro Express to terminate all the job tickets.

Figure 34 • **Script for Terminating all Job Tickets**

```
# This will terminate all the job tickets and export ticket information into the export file
# NOte that the "-terminate" option must be specified if the job is incomplete (i.e. overbuild
# protection counters not reached).
open_project -project {./MyProgJob/MyProgJob.pro}
complete_prog_job -terminate -job_status_file {./MyJobStatusEnd}

# Load the exported file into JobManager to read job end/termination status..
```

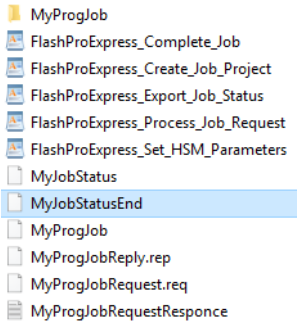
On successful execution of the above script, the Job Tickets are Terminated (as shown in Figure 35, page 23).

Figure 35 • **Job Competition - FlashPro Express Log**



30. Transfer the *JobStatusEnd* file (as shown in Figure 36, page 23) to OE for job status check.

Figure 36 • ***MyJobStatusEnd* File**



3.10 Post-Production

To analyze and cryptographically verify the CofC and Ticket End Verifier in Job Status received from the CM, the Post-Production Checks need to be done. This is done by importing the Job Status file into the HSM Task. The Job Manager uses the U-HSM to check all status validators.

31. Run ***JobManager_Check_Job_End_Status.tcl*** as shown in Figure 37, page 23 in Job Manager and the status can be checked in the Message Box of the Job Manager as shown in the Figure 38, page 24.

Figure 37 • **Script to Check Job End Status**

```
# Import and display job end status
# This will use CU HSM to validate CoCs and ticket end certifiers

# Open project
open_project -project {./MyProject/MyProject.jprj}

# Import and analyze job end/terminate status
import_job_status -job_status_file {./MyJobStatusEnd}

# Close project
close_project -save {TRUE}
```

Figure 38 • **Job End Status - Job Manager Log**