AC425
Application Note
Using DSN, User Design Version, and NVM Data Integrity Check Services in IGLOO2 Devices

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.

51900425. 6.0 3/21

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 6.0

Updated the document for Libero SoC v12.6.

1.2 Revision 5.0

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

- Updated the document for Libero SoC v12.5.
- Removed the references to Libero version numbers.

1.3 Revision 4.0

Updated the document for Libero v11.7 software release (SAR 76991).

1.4 Revision 3.0

Updated the document for Libero v11.6 software release (SAR 71802).

1.5 Revision 2.0

Updated the document for Libero v11.5 software release (SAR 63944).

1.6 Revision 1.0

Initial release.

2 Using DSN, User Design Version, and NVM Data Integrity Check Services in IGLOO2 Devices

2.1 Purpose

This application note describes how to use Device Serial Number (DSN), user design version, and Non-Volatile Memory (NVM) data integrity check system services in the IGLOO®2 Field Programmable Gate Array (FPGA) devices.

2.2 Introduction

System services are system controller actions initiated by asynchronous events from a master in the FPGA fabric. Microsemi provides CoreSysServices soft IP to access the system services implemented by the system controller. The CoreSysServices soft IP provides a user interface to access each of the system services. This IP provides an Advanced High-Performance bus-Lite (AHB-Lite) master interface and communicates with the COMM_BLK through the Fabric Interface Controller (FIC) interface. For more information about communication block, refer to the *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide* and for more information about CoreSysServices soft IP, refer to the *CoreSysServices Handbook*. This application note describes how to use the following system services:

- Device and Design Information Services
  - DSN
  - User Design Version
- NVM Data Integrity Check Service

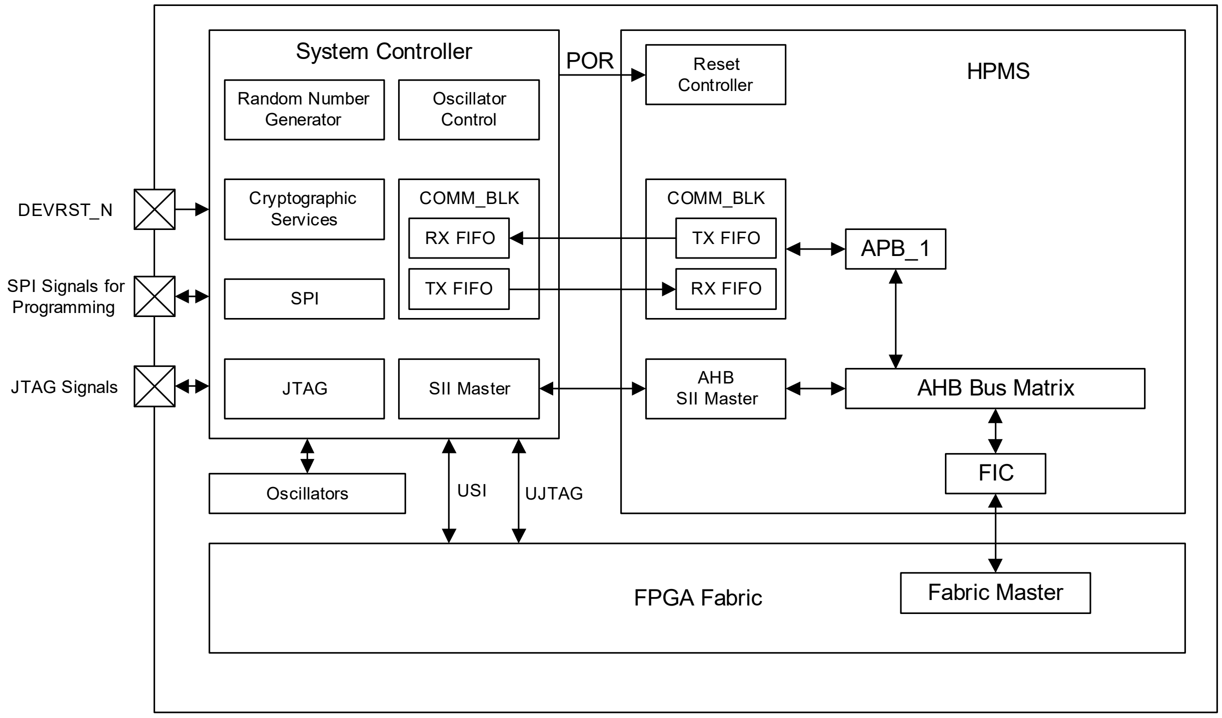
2.2.1 System Controller Block Overview

Figure 1, page 3 shows the IGLOO2 system controller block. There are two COMM_BLKs instantiated, one in High-Performance Memory Subsystem (HPMS) and the other in the system controller and they can communicate with each other.

The COMM_BLK consists of an Advanced Peripheral Bus (APB) interface, eight byte transmit-FIFO, and an eight byte receive-FIFO. It transfers data bi-directionally between the fabric master and the system controller.

System services are requested from the fabric master by sending a command byte describing the function to be performed. It is followed by command-specific sub-commands and/or data through the COMM_BLK interface attached to the HPMS. On completing the requested service, service responses and data are sent back to the fabric master using the COMM_BLK interface.

Figure 1 • **Interfacing the System Controller with HPMS and FPGA Fabric**



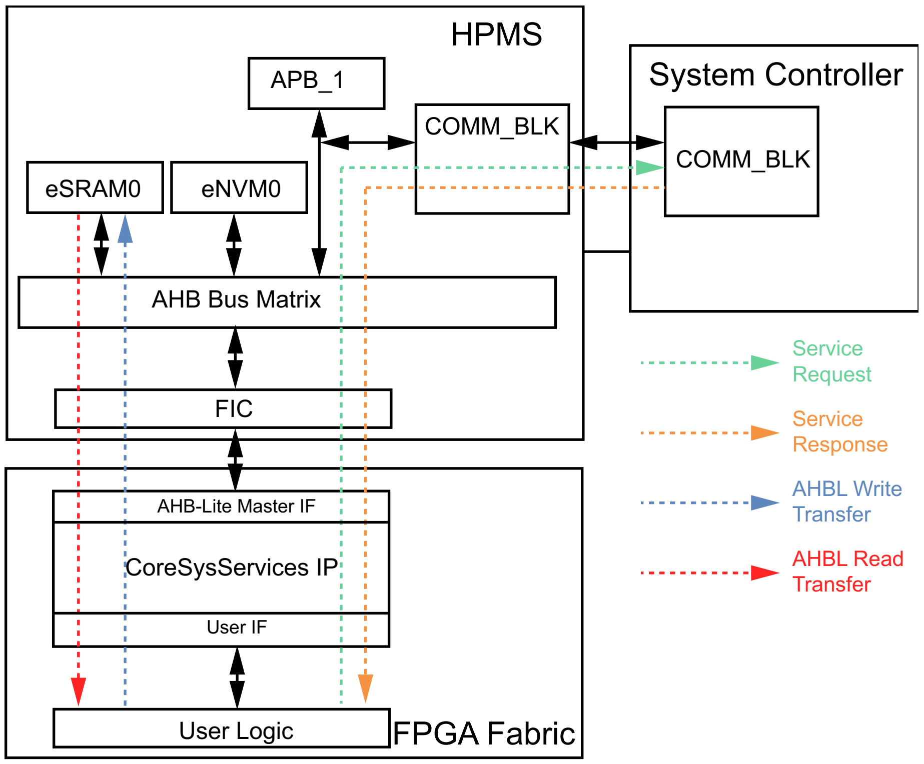
For more information about system controller, refer to the *UG0450: SmartFusion2 SoC FPGA and IGLOO2 FPGA System Controller User Guide*. For more information about COMM_BLK, refer to the Communication Block chapter in the *UG0448: IGLOO2 FPGA High Performance Memory Subsystem User Guide*.

Figure 2, page 4 shows the CoreSysServices data flow. This CoreSysServices data flow diagram shows the following transactions:

- Writing to eSRAM memory
- Communicating with the system controller through the FIC and COMM_BLK (service request and service response)
- Reading from the eSRAM memory

For more information about the data flow diagram, refer to the *CoreSysServices Handbook*.

Figure 2 • **CoreSysServices Data Flow Diagram**



2.2.2 Device and Design Information Services

The device and design information services return information about the device and current user design as described in Serial Number Service, page 4 and User Design Version Service, page 5.

The service request includes a service command and a pointer to a buffer in the HPMS memory to receive the result. The return status of these services can be either success or HPMS memory access error.

2.2.2.1 Serial Number Service

The Serial Number service fetches the 128-bit DSN. The DSN is unique to every device, set during manufacturing. Every device has its own DSN stored in a factory flash NVM. Applications can use DSN for authentication, and DSN enables robust, low-cost security solutions and addresses design security concerns such as design cloning and overbuilding.

DSN comprises of two parts:

- Factory Serial Number (FSN): Lower 64-bits - Uniquely identifies a device, cannot be modified.
- Serial Number Modifier (SNM): Higher 64-bits - Identifies, if a device is zeroized and a new set of factory keys are installed.

Table 1 shows the command value to get the serial number and response statuses.

Table 1 • **Serial Number Service command**

System Service Name	Command Value (Hex)	Response Status
Serial Number Service	01	0: Successful 127: HPMS memory access error (HRESP)

2.2.2.2 User Design Version Service

The User Design Version service fetches the 16-bit user design version stored in the user flash NVM.

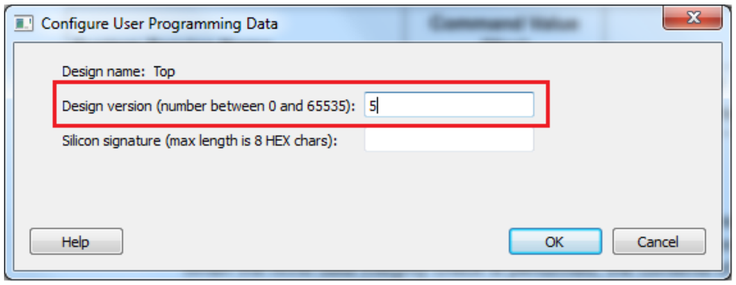
User design version service helps to prevent replay attacks using previously valid bit stream files. For more information about design versioning, refer to the *UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide.*

2.2.2.2.1 Configuring Design Version using Libero

Design version can be configured in the Libero® System-on-Chip (SoC) software using the **Configure Security and Programming** in the **Design Flow** window. Select **Configure User Programming Data**.

1. Enter **Design Version** in the **Configure User Programming Data** window, as shown in Figure 3.

Figure 3 • **Configure User Programming Data**



Design name: Top

Design version (number between 0 and 65535): 5

Silicon signature (max length is 8 HEX chars):

Table 2 shows the command value to get User Design Version and response status.

Table 2 • **User Design Version Service Command**

System Service Name	Command Value (Hex)	Response Status
Serial Number Service	05	0: Successful 127: HPMS memory access error (HRESP)

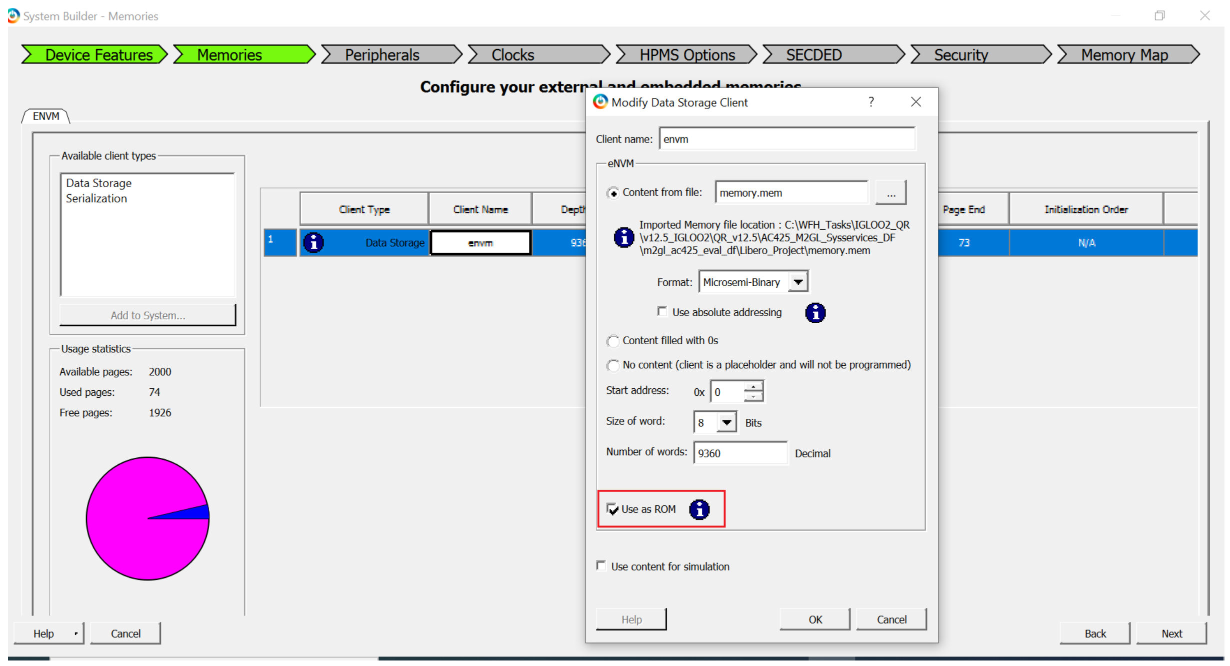
2.2.3 NVM Data Integrity Check Service

The NVM data integrity check service recalculates and compares cryptographic digests of the selected NVM component(s)—fabric, eNVM0, and eNVM1—to those previously computed and saved in NVM. When the NVM data integrity check is performed, the contents of embedded NVM (eNVM) are digested (hashed) using the secure hash algorithm (SHA)-256. The results are compared with the values stored in dedicated NVM located in each segment. If the contents are unchanged, that is, if the current and stored digests match, the digest test passes, otherwise failure is flagged. This digest test provides assurance against both natural and maliciously induced failures. In this application note, only the NVM data integrity check service is demonstrated. The final eNVM content is analyzed, and a digest is generated after programming is completed.

The eNVM digests are computed only on eNVM pages that are declared as ROM by the user. Pages that are not flagged as ROM are not included in the eNVM digest calculation.

To declare the eNVM pages as ROM, check **Use as ROM** in the **Modify Data Storage Client** window under the **Memories** tab of the **System Builder Configurator**, as shown in Figure 4, page 6.

Figure 4 • **System Builder Memories Tab**



Usage statistics
Available pages: 2000
Used pages: 74
Free pages: 1926

	Client Type	Client Name	Depth	Page End	Initialization Order
1	Data Storage	envm	936	73	N/A

Client name: envm
(●) Content from file: memory.mem
Imported Memory file location : C:\WFH_Tasks\IGLOO2_QR\v12.5_IGLOO2\QR_v12.5\AC425_M2GL_Sysservices_DF\m2gl_ac425_eval_df\Libero_Project\memory.mem
Format: Microsemi-Binary
☐ Use absolute addressing
( ) Content filled with 0s
( ) No content (client is a placeholder and will not be programmed)
Start address: 0x 0
Size of word: 8 Bits
Number of words: 9360 Decimal
☑ Use as ROM
☐ Use content for simulation

Table 3 shows the command value, options to perform NVM data integrity check service, and response statuses. The OPTIONS field in the NVM data integrity check service request selects the NVM components—fabric configuration, eNVM0, and eNVM1—for data integrity check.

- If the bit FABRIC is set to 1, then the FPGA fabric configuration digest test is performed.
- If the bits eNVM0 or eNVM1 are set to 1, then the corresponding eNVM digest tests are performed.
- If a digest mismatch occurs, DIGESTERR indicates which of the selected digests failed.

Table 3 • **NVM Data Integrity Check Service**

System Service Name	Command Value (Hex)	OPTIONS	Response Status
NVM Data Integrity Check Service	17	Bits [7:3] - Reserved 2: eNVM1 1: eNVM0 0: FABRIC	Digest error byte (DIGESTERR) Bits [7:3] - Reserved Bits [2] - eNVM1 Error 0: NVM1 data integrity check passed 1: NVM1 data integrity check mismatch Bits [1] - eNVM0 Error 0: NVM data integrity check passed 1: NVM data integrity check mismatch Bits [0] - Fabric Error 0: Fabric FPGA configuration data integrity check passed 1: Fabric FPGA configuration data integrity check mismatch

2.3 References

The following documents are referenced in this document.

- *UG0450: SmartFusion2 SoC FPGA and IGLOO2 FPGA System Controller User Guide*
- *UG0478: IGLOO2 FPGA Evaluation Kit User Guide*
- *UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide*
- *CoreSysServices Handbook*

2.4 Design Requirements

The following table lists the resources required to run the design.

Table 4 • **Design Requirements**

Requirement	**Version**
Operating System	64 bit Windows 7 and 10
Hardware	
IGLOO2 Evaluation Kit: • 12 V adapter • USB A to Mini-B cable	Rev C, Rev D, or later
Host PC or laptop	
Software	
Libero SoC	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	
Host PC Drivers	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 start:

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=m2gl_ac425_eval_df

2.6 Design Description

The design is implemented using the IGLOO2 Evaluation Kit board that has the M2GL010T-FG484 device.

This design example uses the following:

- IGLOO2 HPMS
- On-chip 50 MHz RC oscillator
- Fabric CCC
- CoreSysServices IP
- CoreRESET
- CoreABC
- CoreUARTapb
- Fabric state machine to control CoreSysServices IP
- An APB data block to capture DSN, Design Version, and NVM data integrity check response values.

2.7 Hardware Implementation

Figure 5 shows the design block diagram. 50 MHz RC oscillator is used as the clock source. It is used with CCC to provide a 100 MHz reference clock to the HPMS. This 100 MHz clock is used as the main clock for the fabric clocks. CoreRESETP generates reset signals for all the blocks. The CoreSysServices IP is configured to access the DSN, User Design Version, and NVM Data Integrity Check Services. Refer to Figure 6, page 8 and Figure 7, page 9.

The CoreSysServices IP sends commands requested by SysService state control logic to the system controller through the COMM_BLK block in the HPMS. The fabric SysService state control logic issues System Service commands to CoreSysServices IP and generates required control signals. It captures the data from CoreSysServices IP on completing the requested service. The APB data block captures the data values from SysService state control logic and converts the Hex data to ASCII format data. The CoreABC program controls the initiation of the SysService state control logic. CoreABC captures the ASCII data form the APB data block and displays data on HyperTerminal using the CoreUARTapb interface. The Fabric logic also consists of a counter block to display the counter value through the Light Emitting Diodes (LEDs) to indicate that the design is up and running.

Figure 5 • **Hardware Implementation Block Diagram**

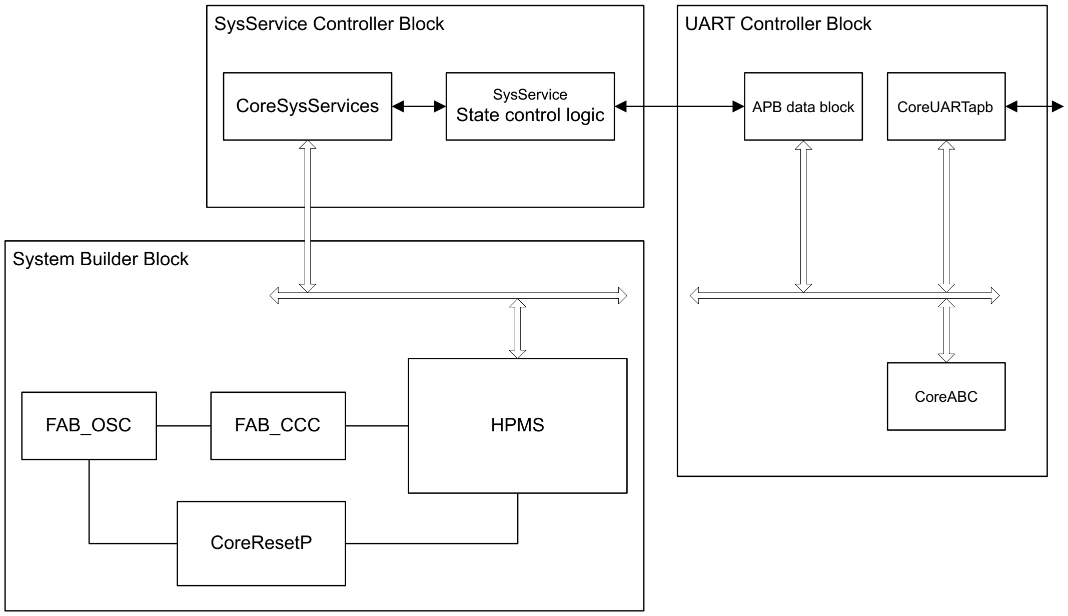
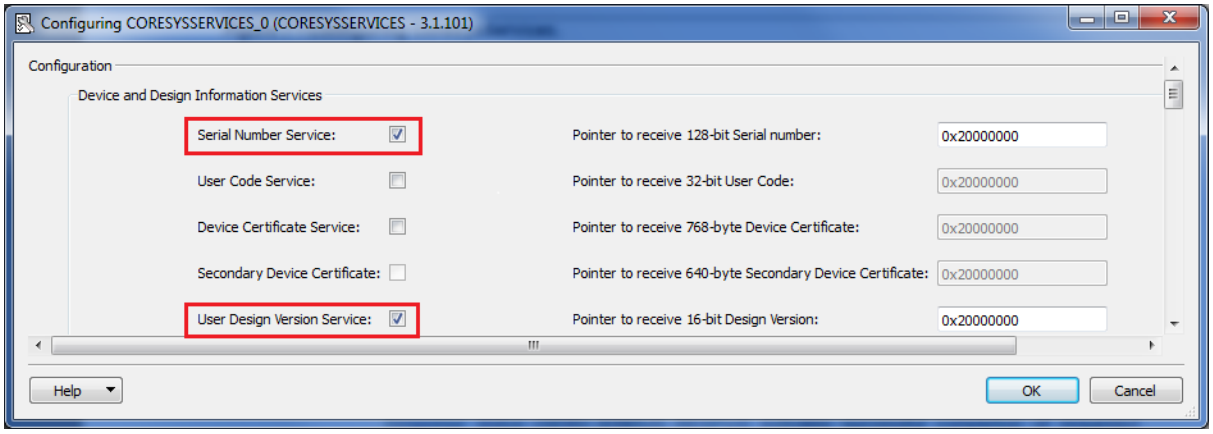


Figure 6 • **Configuring CoreSysServices**

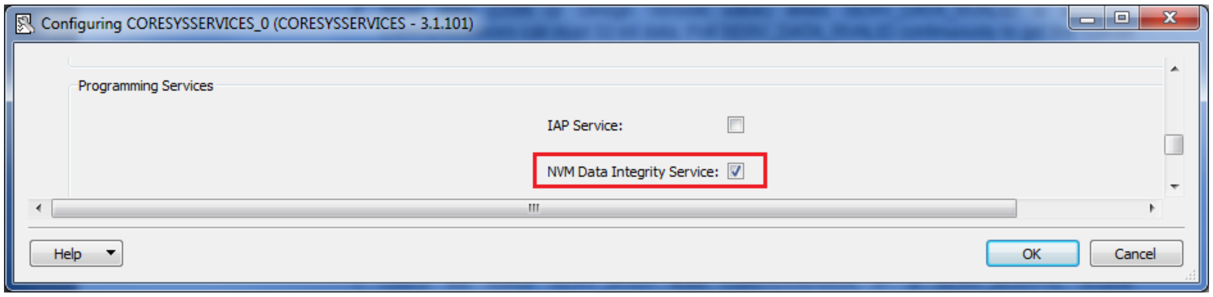

Configuring CORESYSSERVICES_0 (CORESYSSERVICES - 3.1.101)

Configuration

Device and Design Information Services

Serial Number Service:	☑	Pointer to receive 128-bit Serial number:	0x20000000
User Code Service:	☐	Pointer to receive 32-bit User Code:	0x20000000
Device Certificate Service:	☐	Pointer to receive 768-byte Device Certificate:	0x20000000
Secondary Device Certificate:	☐	Pointer to receive 640-byte Secondary Device Certificate:	0x20000000
User Design Version Service:	☑	Pointer to receive 16-bit Design Version:	0x20000000

Figure 7 • **Configuring CoreSysServices**



Programming Services

IAP Service: ☐

NVM Data Integrity Service: ☑

2.7.1 Procedure for DSN, User Design Version, and NVM Data Integrity Check

The following steps describe how to get the DSN and User Design Version using CoreSysServices IP:

1. Check the status of SERV_BUSY from CoreSysServices IP. If SERV_BUSY = 0, enable SERV_ENABLE_REQ.
2. Send command byte, SERV_CMDBYTE_REQ as 01 for DSN and 05 for User Design version.
3. Clear request enable, SERV_ENABLE_REQ.
4. Read data (DSN or Design Version value) when SERV_DATA_RVALID is asserted. CoreSysServices can read 32-bit data. Poll SERV_DATA_RVALID continuously to get the 128-bit DSN.
5. To confirm the completion of service request, check whether SERV_STATUS_VALID = 1 and SERV_STATUS_RESP = 0.

   Response status (SERV_STATUS_RESP) = 0 indicates successful completion of requested service.
6. Read data is converted from HexDecimal to ASCII format.
7. CoreABC reads the ASCII data and sends it for display using the CoreUARTapb module.

The following steps describe how to perform the NVM data integrity check using CoreSysServices IP:

1. Check the status SERV_BUSY from CoreSysServices IP. If SERV_BUSY = 0, enable SERV_ENABLE_REQ.
2. Send the command byte, SERV_CMDBYTE_REQ as 17, option SERV_OPTIONS_MODE as 2 to perform NVM data integrity check.
3. Clear request enable, SERV_ENABLE_REQ.
4. If SERV_STATUS_VALID = 1, read digest response, SERV_STATUS_RESP.

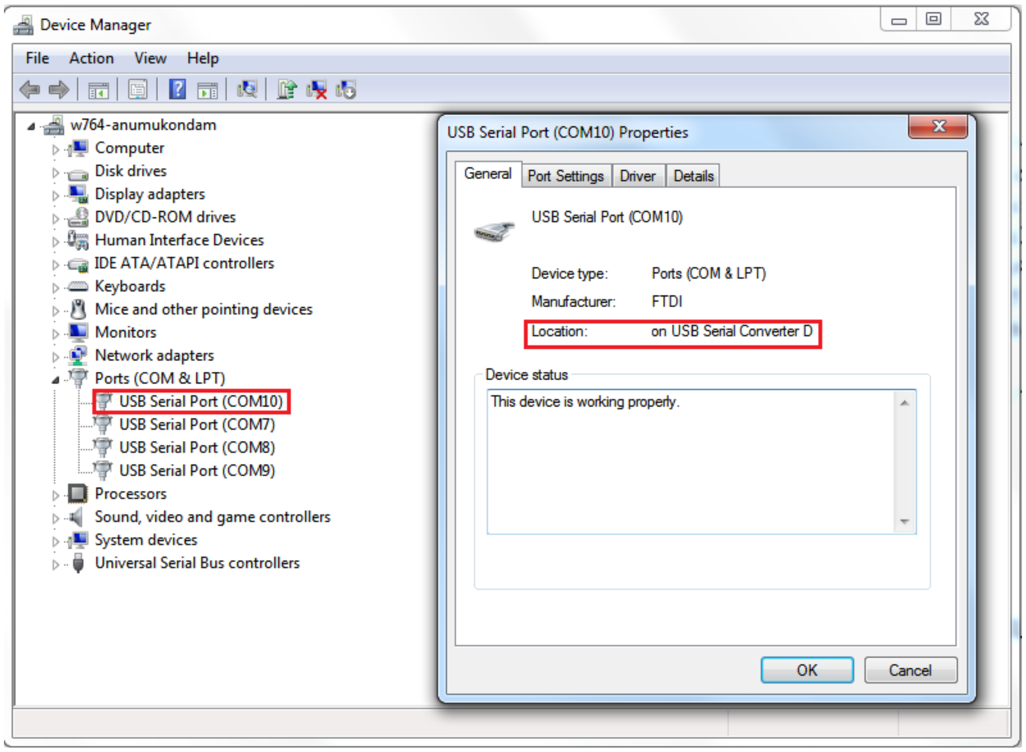
   If Response status from CoreSysServices IP SERV_STATUS_RESP = 0, data integrity is passed else digest mismatch.
5. Digest Response is converted from HexDecimal to ASCII format.
6. CoreABC reads the ASCII data and sends it for display using the CoreUARTapb module.

2.8 Setting Up the Design

The following steps describe how to set up a design:

1. Connect the power supply to the J6 connector.
2. Connect one end of the USB mini cable to the J18 connector provided on the IGLOO2 Evaluation Kit board.
3. Connect the other end of the USB cable to the host PC. Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the Device Manager of the host PC. The FTDI USB to UART converter enumerates four COM ports. Note down the USB Serial Converter D COM port number to use it in the HyperTerminal. Figure 8 shows the USB Serial port properties and COM10 connected to USB Serial Converter D.
4. 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 8 • **USB Serial Port Properties Window**



w764-anumukondam
- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - USB Serial Port (COM10)
  - USB Serial Port (COM7)
  - USB Serial Port (COM8)
  - USB Serial Port (COM9)
- Processors
- Sound, video and game controllers
- System devices
- Universal Serial Bus controllers

USB Serial Port (COM10) Properties

General | Port Settings | Driver | Details

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 IGLOO2 Evaluation Kit, as shown in Table 5.

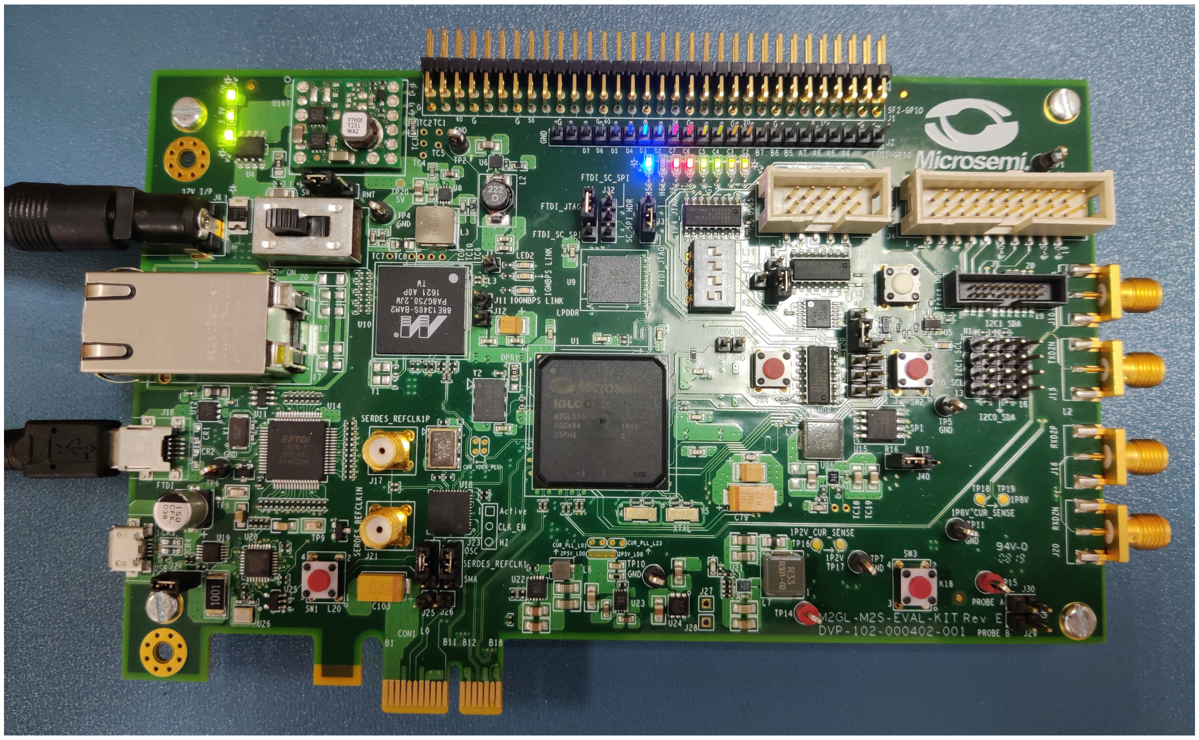
Note: While making the jumper connections the power supply switch SW7 on the board must be in OFF position.

Table 5 • **IGLOO2 Evaluation Kit Jumper Settings**

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

Figure 9 shows the board setup for running the system services design on the IGLOO2 Evaluation Kit board.

Figure 9 • **IGLOO2 Evaluation Kit Board**



2.9 Running the Design

The following steps describe running the design example on IGLOO2 Evaluation Kit board using the M2GL010T-FG484 device:

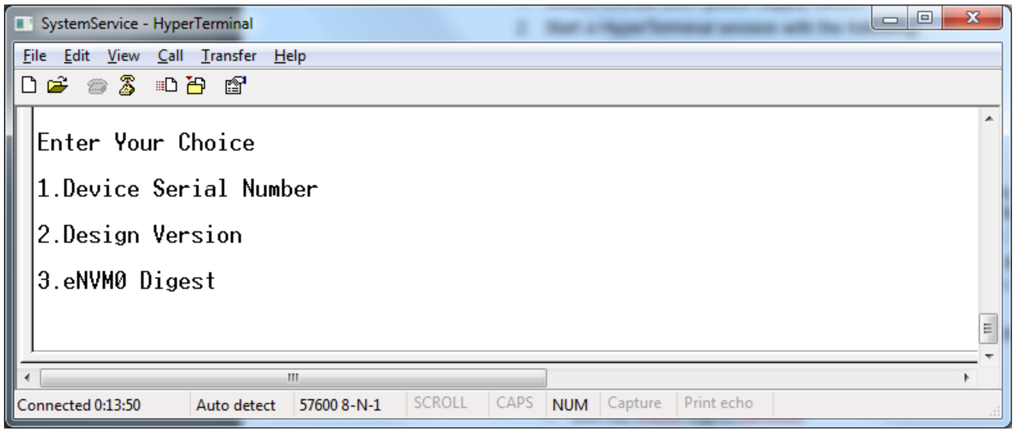
1. Switch ON the SW7 power supply switch.
2. Start a HyperTerminal session with the following:
   - 57600 baud rate
   - 8 data bits
   - 1 stop bit
   - No parity
   - No flow control.

If the computer does not have the HyperTerminal program, any free serial terminal emulation program such as PuTTY or TeraTerm can be used. For configuring HyperTerminal, TeraTerm, or PuTTY, refer to the *Configuring Serial Terminal Emulation Programs Tutorial*.

3. Program the IGLOO2 Evaluation Kit board with the job file provided as part of the design files using FlashPro Express software, refer to the Appendix: Programming the Device Using FlashPro Express, page 15.
4. After programming, HyperTerminal displays a message to choose one of the following:
   - DSN
   - Design version
   - eNVM0 digest

Figure 10 shows the HyperTerminal Window.

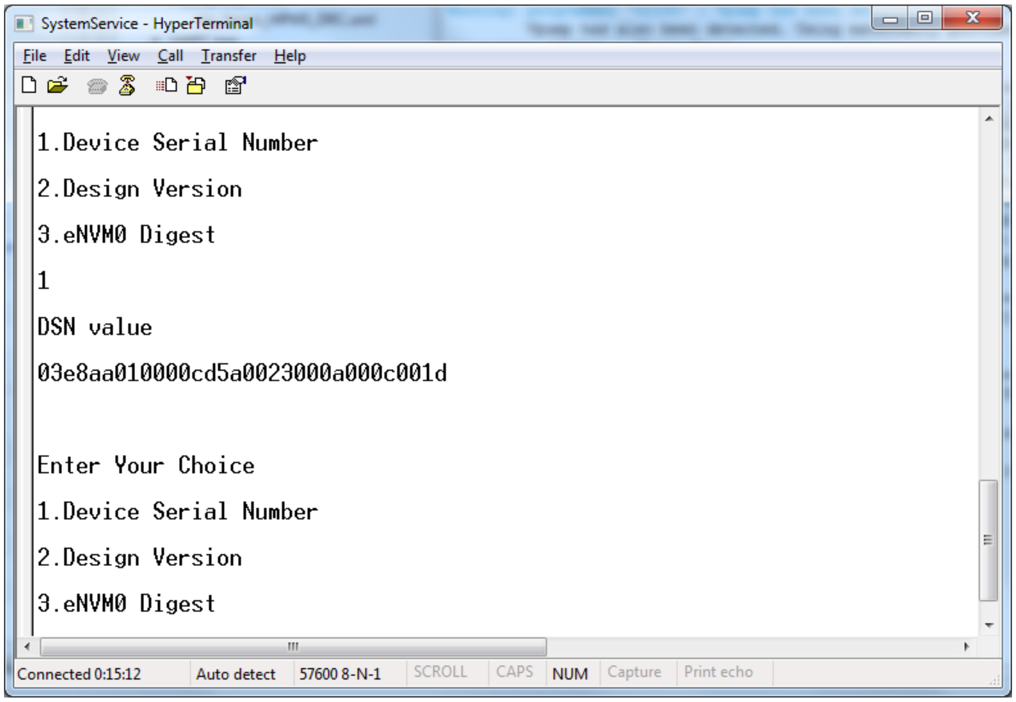
Figure 10 • **HyperTerminal Window**


SystemService - HyperTerminal
File Edit View Call Transfer Help

```
Enter Your Choice
1.Device Serial Number
2.Design Version
3.eNVM0 Digest
```

5. Select option 1 to get DSN. Figure 11 shows the HyperTerminal displaying the DSN value.

Figure 11 • **Device Serial Number**

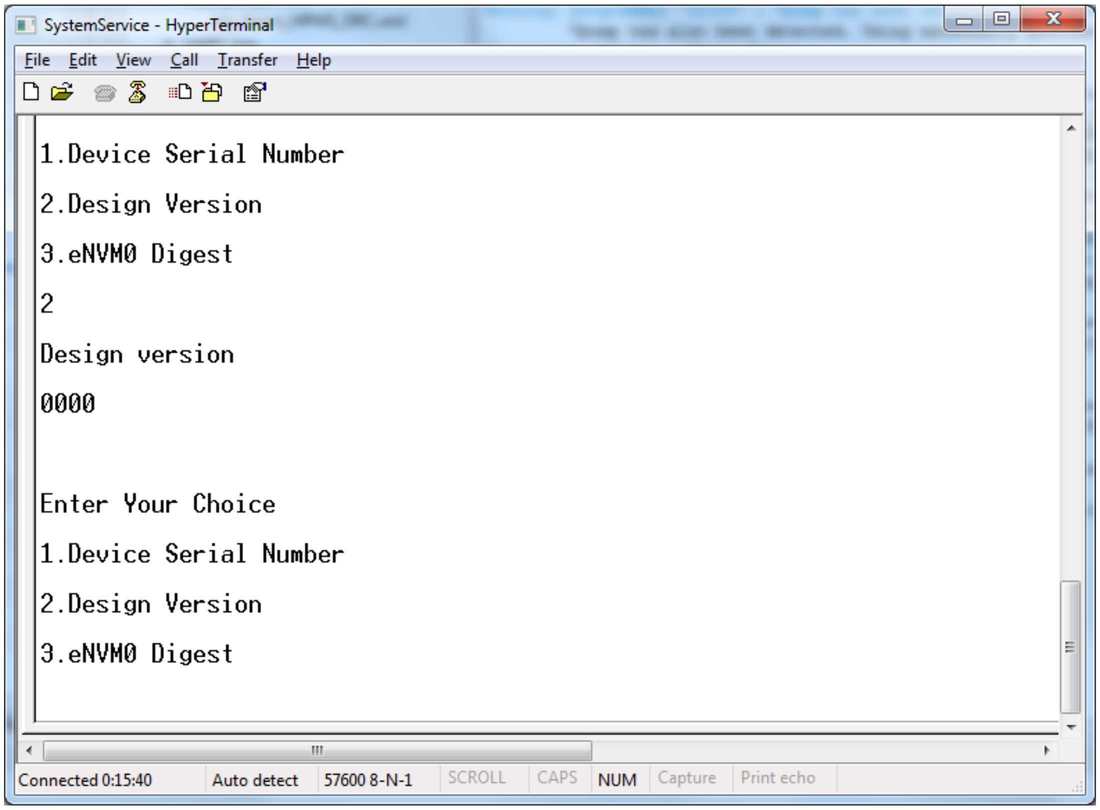


```
1.Device Serial Number
2.Design Version
3.eNVM0 Digest
1
DSN value
03e8aa010000cd5a0023000a000c001d


Enter Your Choice
1.Device Serial Number
2.Design Version
3.eNVM0 Digest
```

6. Select option 2 to get design version. 0000 is the default value of design version. Figure 12 shows the HyperTerminal displaying the Design Version.

Figure 12 • **Design Version**

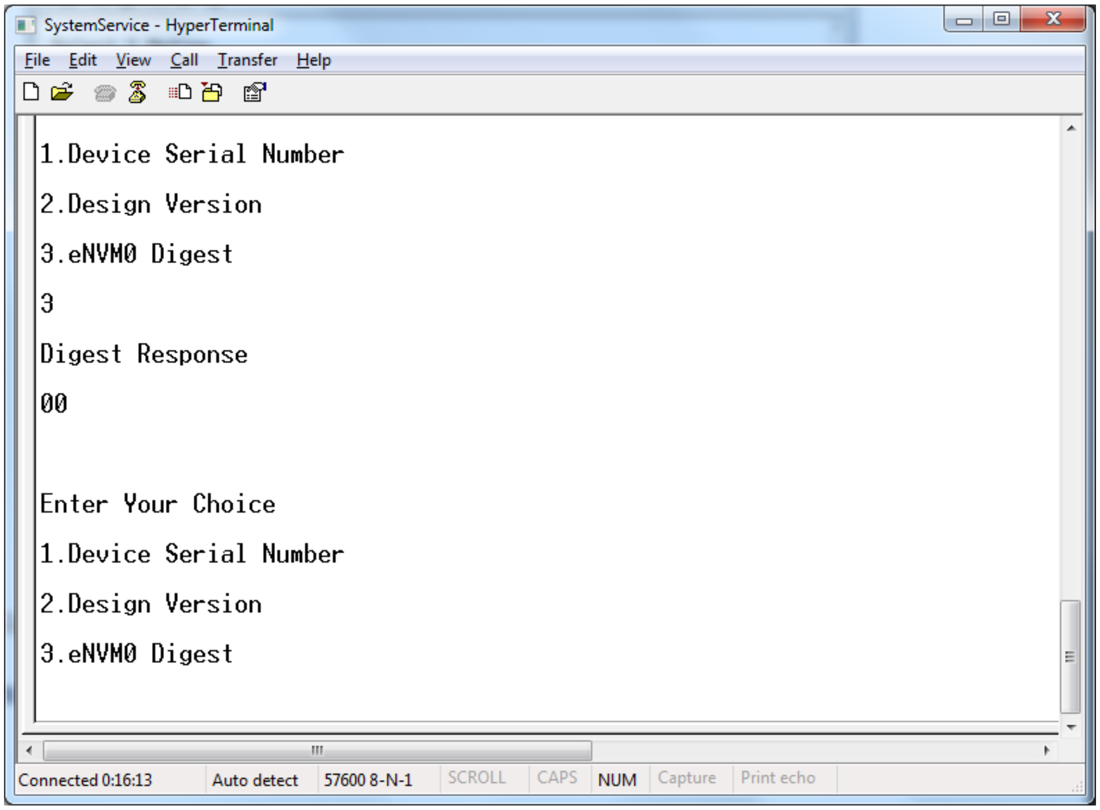


```
1.Device Serial Number
2.Design Version
3.eNVM0 Digest
2
Design version
0000


Enter Your Choice
1.Device Serial Number
2.Design Version
3.eNVM0 Digest
```

7. Select option 3 to get eNVM0 digest response. Figure 13 shows the HyperTerminal displaying the digest response. Digest Response 0 indicates digest passed.

Figure 13 • **eNVM0 Digest Response**



```
1.Device Serial Number
2.Design Version
3.eNVM0 Digest
3
Digest Response
00


Enter Your Choice
1.Device Serial Number
2.Design Version
3.eNVM0 Digest
```

2.10 Conclusion

This application note describes how to use DSN, User Design Version, and NVM data integrity check system services in the IGLOO2 FPGA devices.

3 Appendix: Programming the Device Using FlashPro Express

This section describes how to program the IGLOO2 device with the programming job file using FlashPro Express.

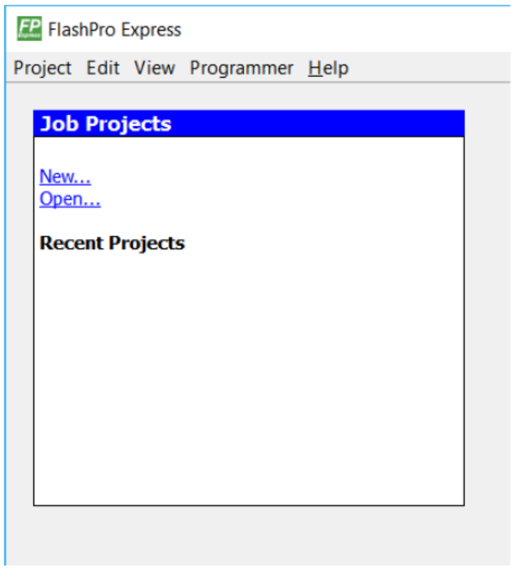
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in Table 5, page 11.

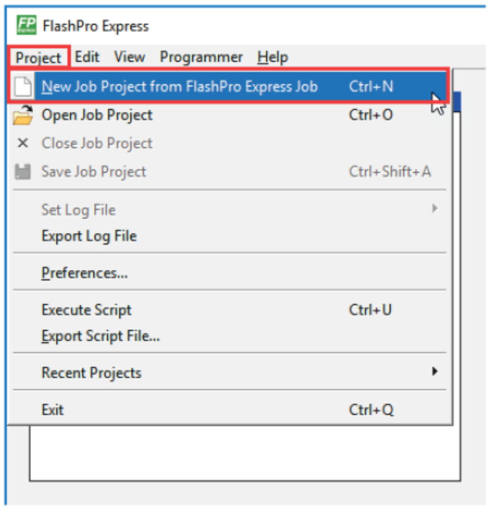
Note: The power supply switch must be switched off while making the jumper connections.

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 the following figure.

Figure 14 • **FlashPro Express Job Project**



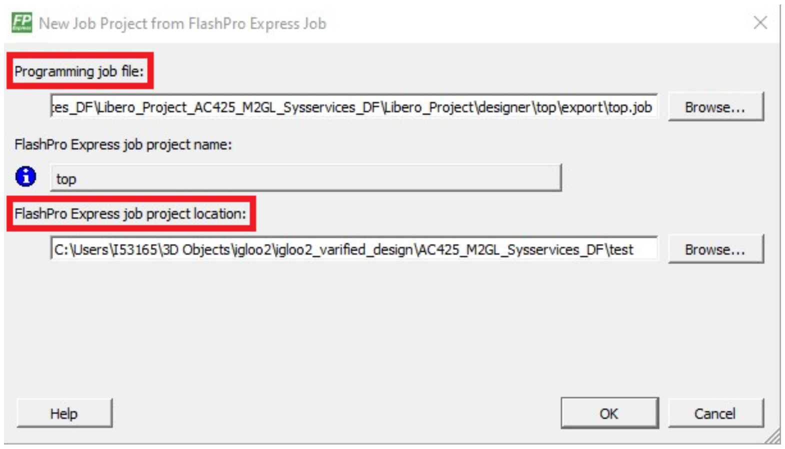
or



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>\m2gl_ac425_eval_df\Programming_Job
- **FlashPro Express job project name:** Click **Browse** and navigate to the location where you want to save the project.

Figure 15 • **New Job Project from FlashPro Express Job**



Programming job file:

es_DF\Libero_Project_AC425_M2GL_Sysservices_DF\Libero_Project\designer\top\export\top.job

FlashPro Express job project name:

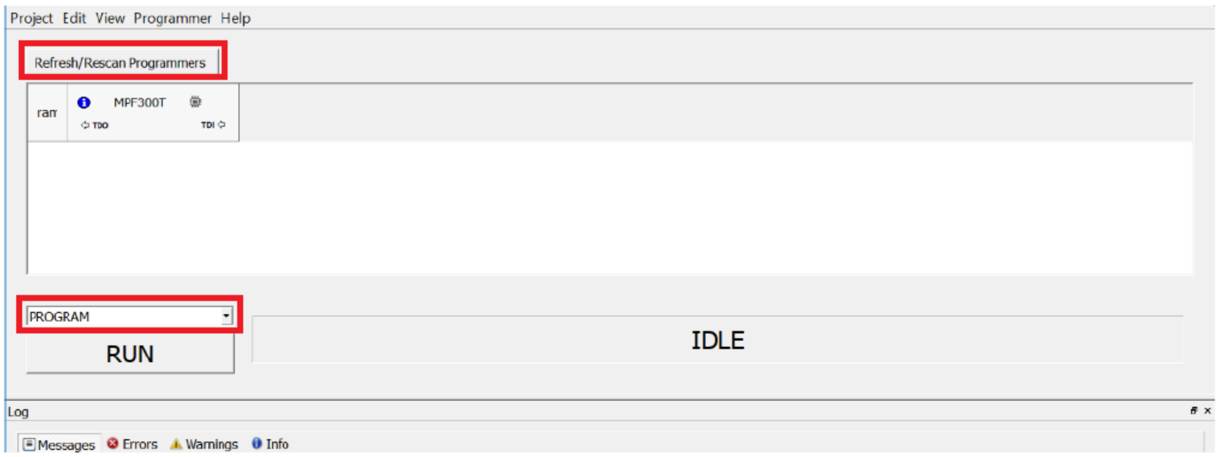
top

FlashPro Express job project location:

C:\Users\I53165\3D Objects\igloo2\igloo2_varified_design\AC425_M2GL_Sysservices_DF\test

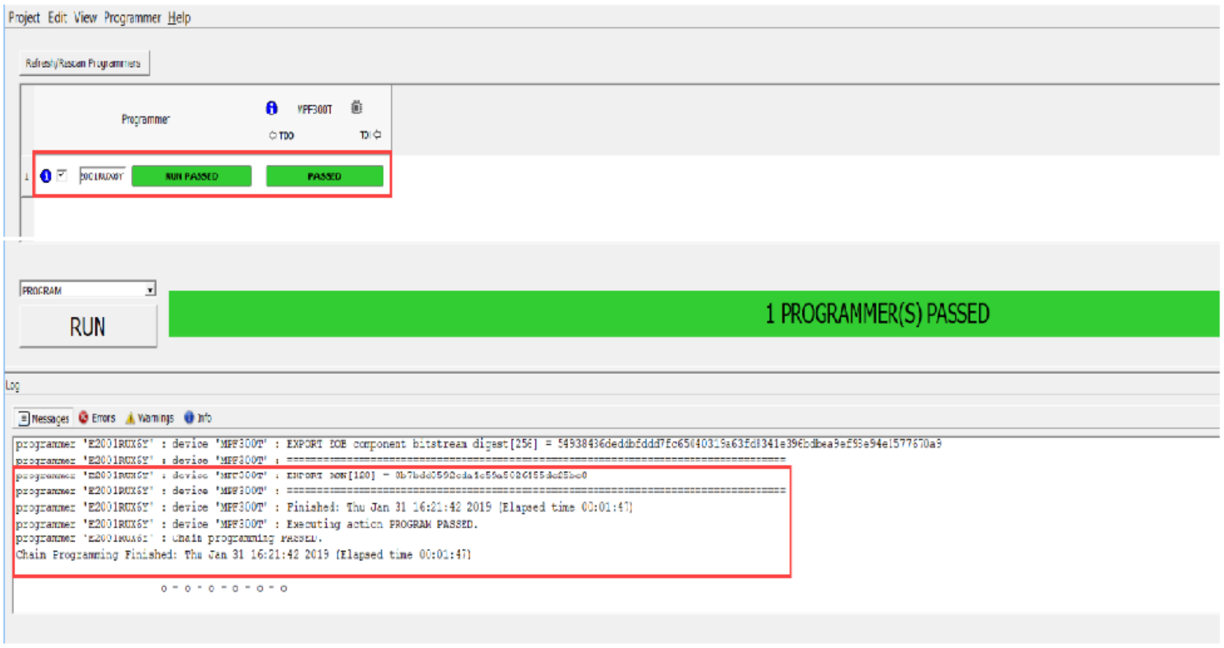
7. Click **OK**. The required programming file is selected and ready to be programmed in the device.
8. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan**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

Figure 16 • **Programming the Device**



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

Figure 17 • **FlashPro Express—RUN PASSED**



```
programmer 'E2001RUX6Y' : device 'MPF300T' : EXPORT DOE component bitstream digest[256] = 54938436deddbfddd7fc65040319a63fd8341e396bdbea9ef93e94e157767 0a9
programmer 'E2001RUX6Y' : device 'MPF300T' : ================================================================================
programmer 'E2001RUX6Y' : device 'MPF300T' : EXPORT DSN[128] = 0b7bdd0592cda1c59a5026155dcd5bc0
programmer 'E2001RUX6Y' : device 'MPF300T' : ================================================================================
programmer 'E2001RUX6Y' : device 'MPF300T' : Finished: Thu Jan 31 16:21:42 2019 (Elapsed time 00:01:47)
programmer 'E2001RUX6Y' : device 'MPF300T' : Executing action PROGRAM PASSED.
programmer 'E2001RUX6Y' : Chain programming PASSED.
Chain Programming Finished: Thu Jan 31 16:21:42 2019 (Elapsed time 00:01:47)
                    o - o - o - o - o - o
```

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