

Using DSN, User Design Version, and NVM Data Integrity Check Services in IGLOO2 Devices - Libero SoC v11.6

Table of Contents

Purpose	1
Introduction	1
System Controller Block Overview	2
Device and Design Information Services	4
NVM Data Integrity Check Service	5
References	7
Design Requirements	8
Design Description	8
Hardware Implementation	9
Procedure for DSN, User Design Version, and NVM Data Integrity Check	10
Setting Up the Design	11
Running the Design	13
Conclusion	16
Appendix A: Design and Programming Files	17
List of Changes	18

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 IGLOO2[®] field programmable gate array (FPGA) devices.

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

System Controller Block Overview

Figure 1 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.

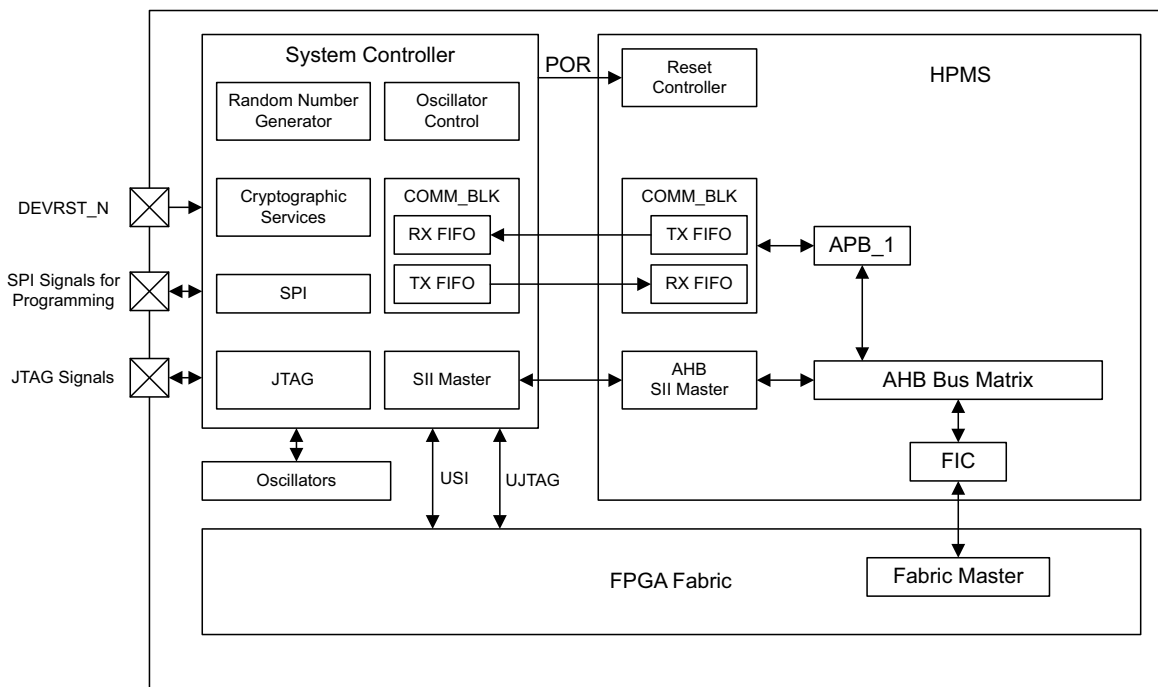


Figure 1 • Interfacing the System Controller with HPMS and FPGA Fabric

For more information about System Controller, refer to the [UG0450: SmartFusion2 SoC 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 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](#).

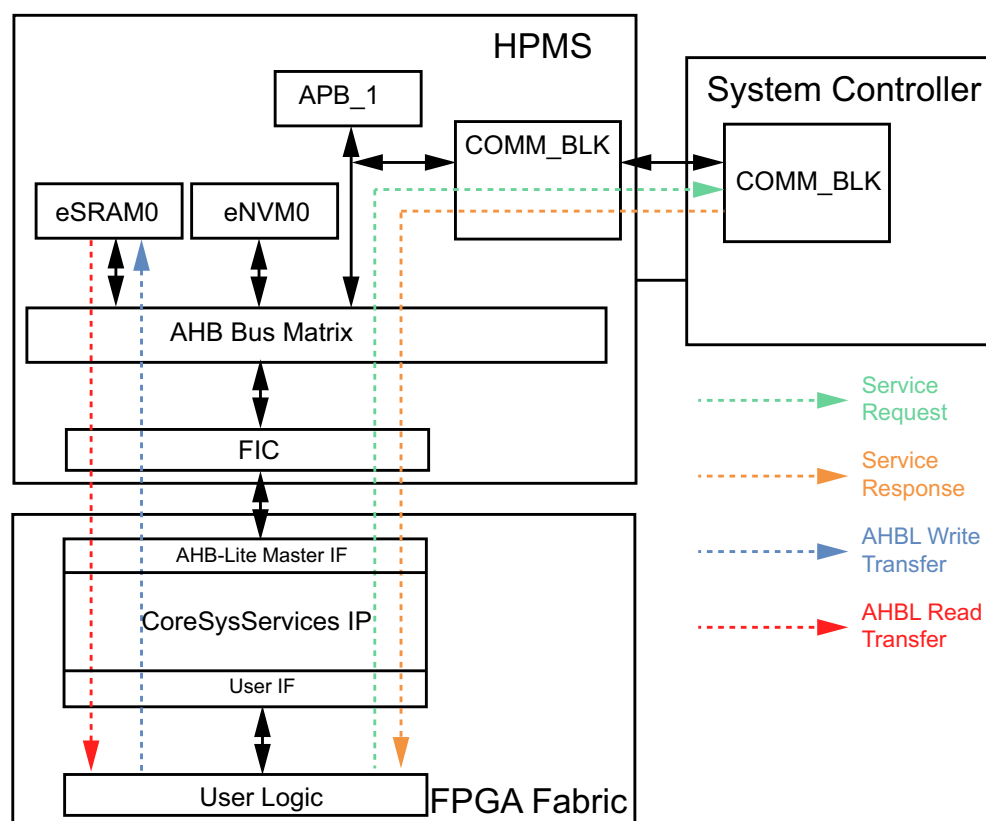


Figure 2 • CoreSysServices Data Flow Diagram

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" section](#) and ["User Design Version Service" section on page 4](#).

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.

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 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)

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. Refer to the [UG0443: SmartFusion2 and IGLOO2 FPGA Security and Reliability User Guide](#) for more information on design versioning.

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](#).

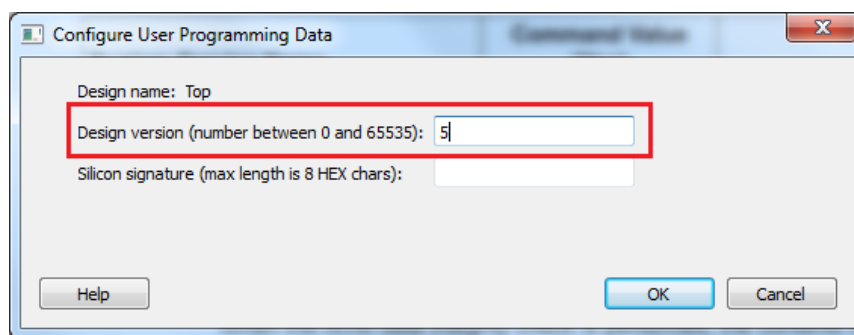


Figure 3 • Configure User Programming Data

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)

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 a failure is flagged. This digest test provides assurance against both natural and maliciously induced failures. In this application note, only 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](#).

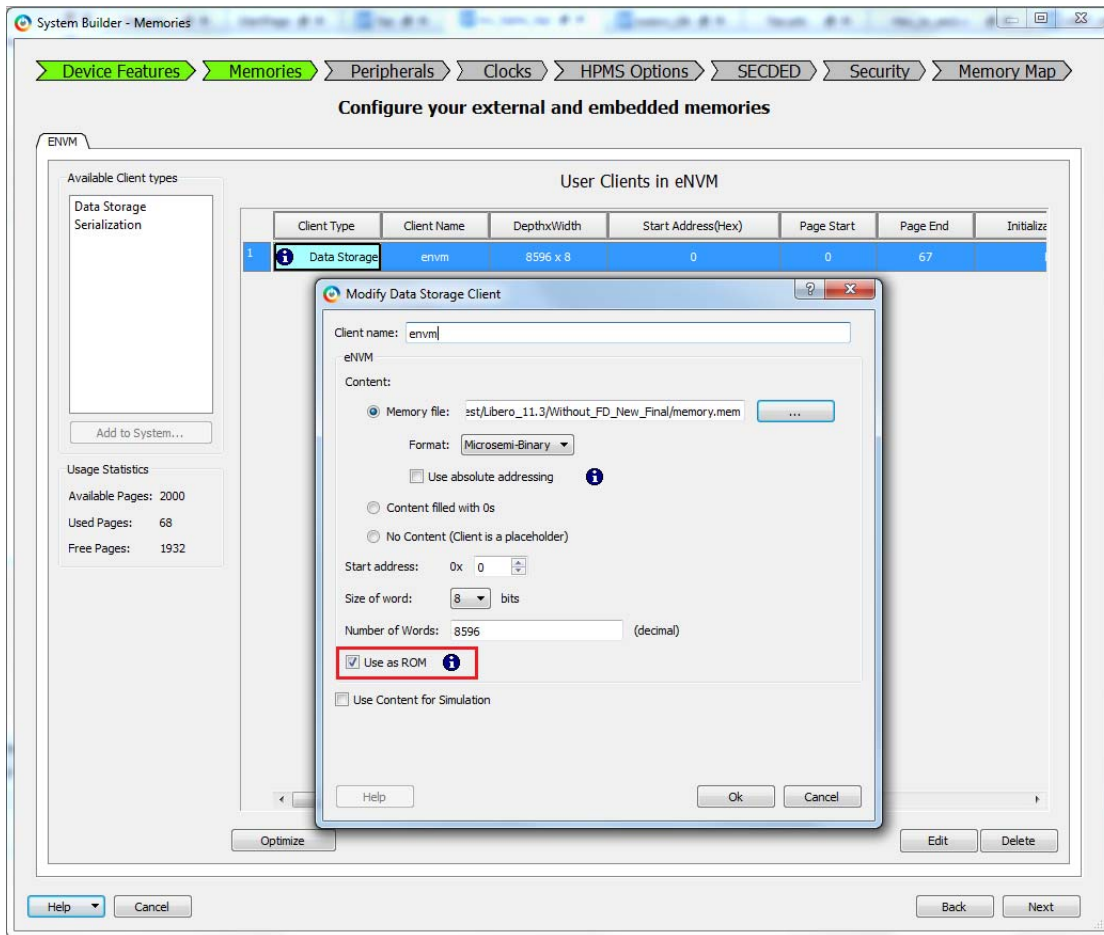


Figure 4 • System Builder Memories Tab

[Table 3 on page 7](#) 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

References

The following documents are referenced in this document.

- [UG0450: SmartFusion2 SoC 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](#)

Design Requirements

Table 4 shows the design requirements.

Table 4 • Design Requirements

Design Requirements	Description
Hardware Requirements	
IGLOO2 Evaluation Kit: <ul style="list-style-type: none">• 12 V adapter• FlashPro4 programmer• USB A to Mini-B cable	Rev D or later
Host PC	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.6
FlashPro programming software	v11.6
Host PC Drivers	USB to UART drivers
One of the following serial terminal emulation programs: <ul style="list-style-type: none">• Hyperterminal• TeraTerm• PuTTY	

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

Hardware Implementation

Figure 5 shows the design block diagram. A 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 and Figure 7 on page 10.

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 initiation of the SysService state control logic. CoreABC captures the ASCII data form APB data block and displays data on HyperTerminal using 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.

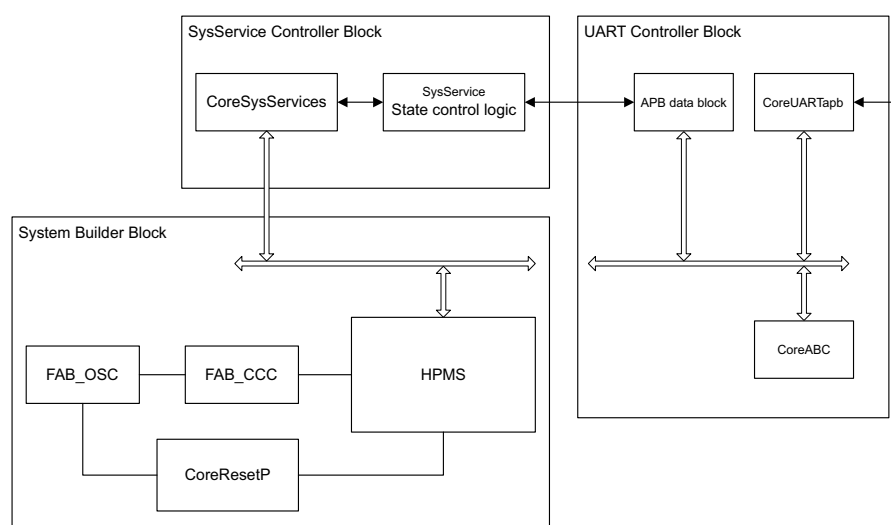


Figure 5 • Hardware Implementation Block Diagram

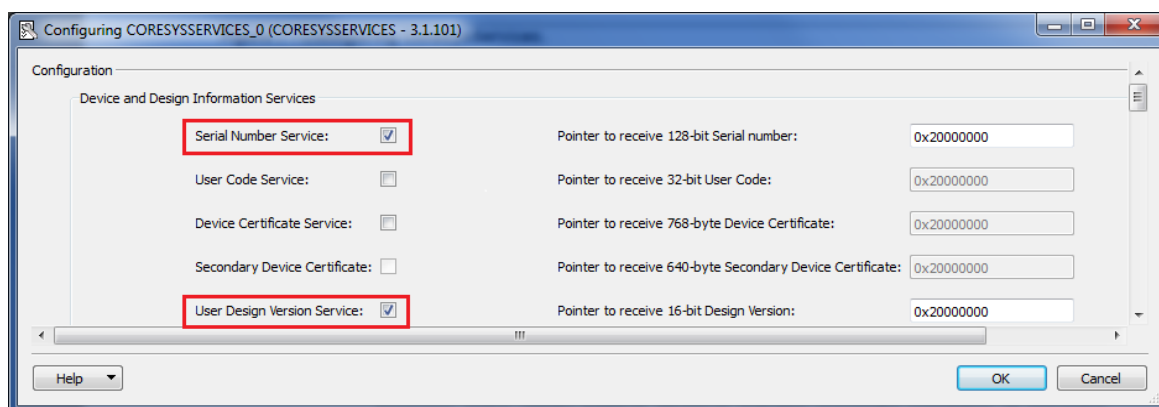


Figure 6 • Configuring CoreSysServices

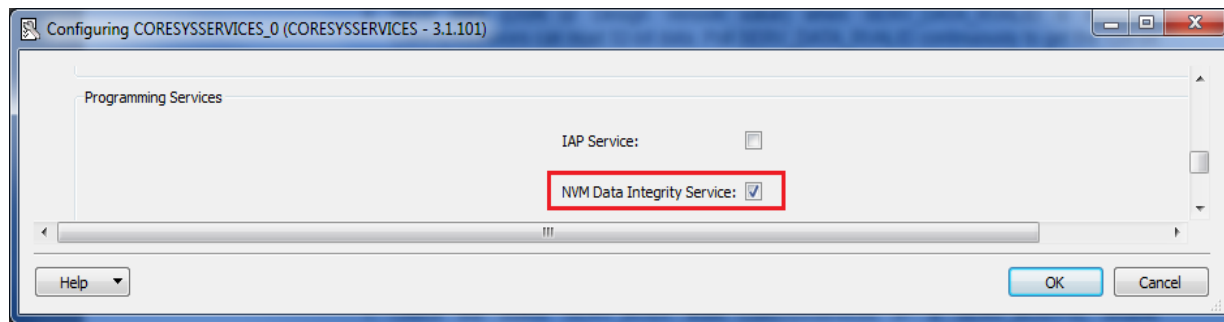


Figure 7 • Configuring CoreSysServices.

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 send it for display using 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 CoreUARTapb module.

Setting Up the Design

1. Connect the FlashPro4 ribbon cable to the J5 connector (JTAG Programming Header) of the IGLOO2 Evaluation Kit board.
2. Connect the mini USB cable between the FlashPro4 and the USB port of the host PC.
3. Connect the power supply to the J6 connector.
4. Connect one end of the USB mini cable to the J18 connector provided on the IGLOO2 Evaluation Kit.
5. 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. As shown in [Figure 8](#), COM10 is connected to USB Serial Converter D.
6. 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.

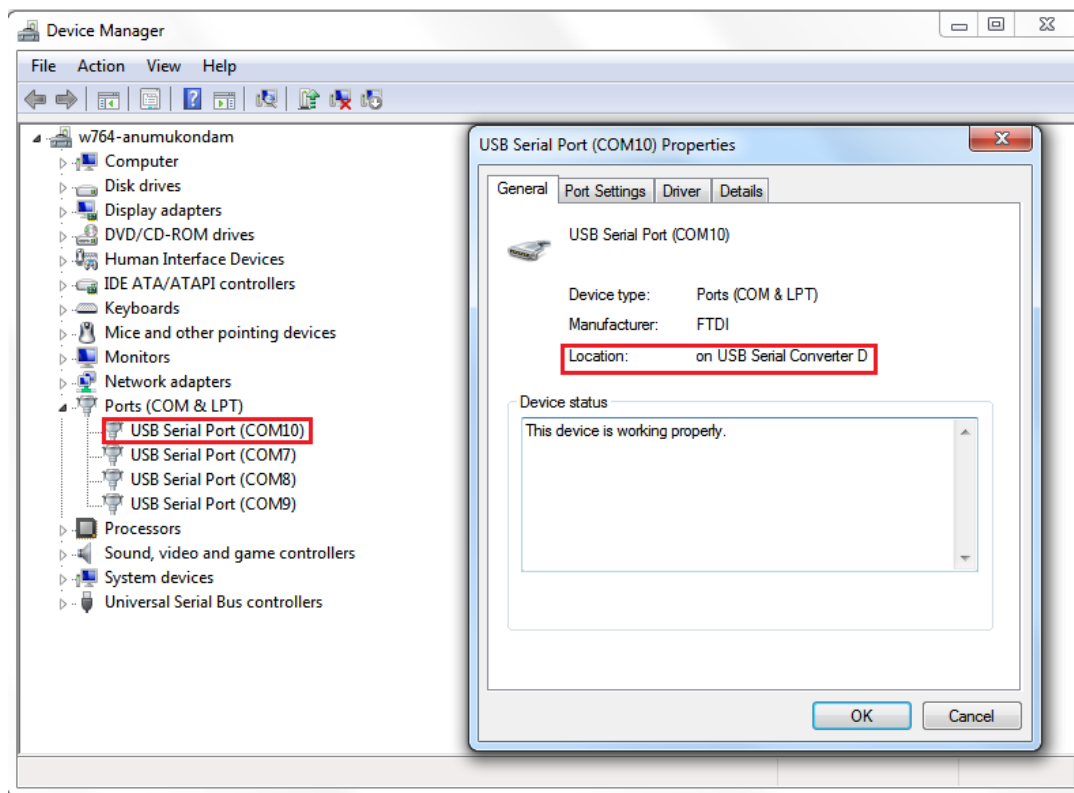


Figure 8 • USB Serial Port Properties Window

7. Connect the jumpers on the IGLOO2 Evaluation Kit, as shown in [Figure 5](#) on page 12.

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.

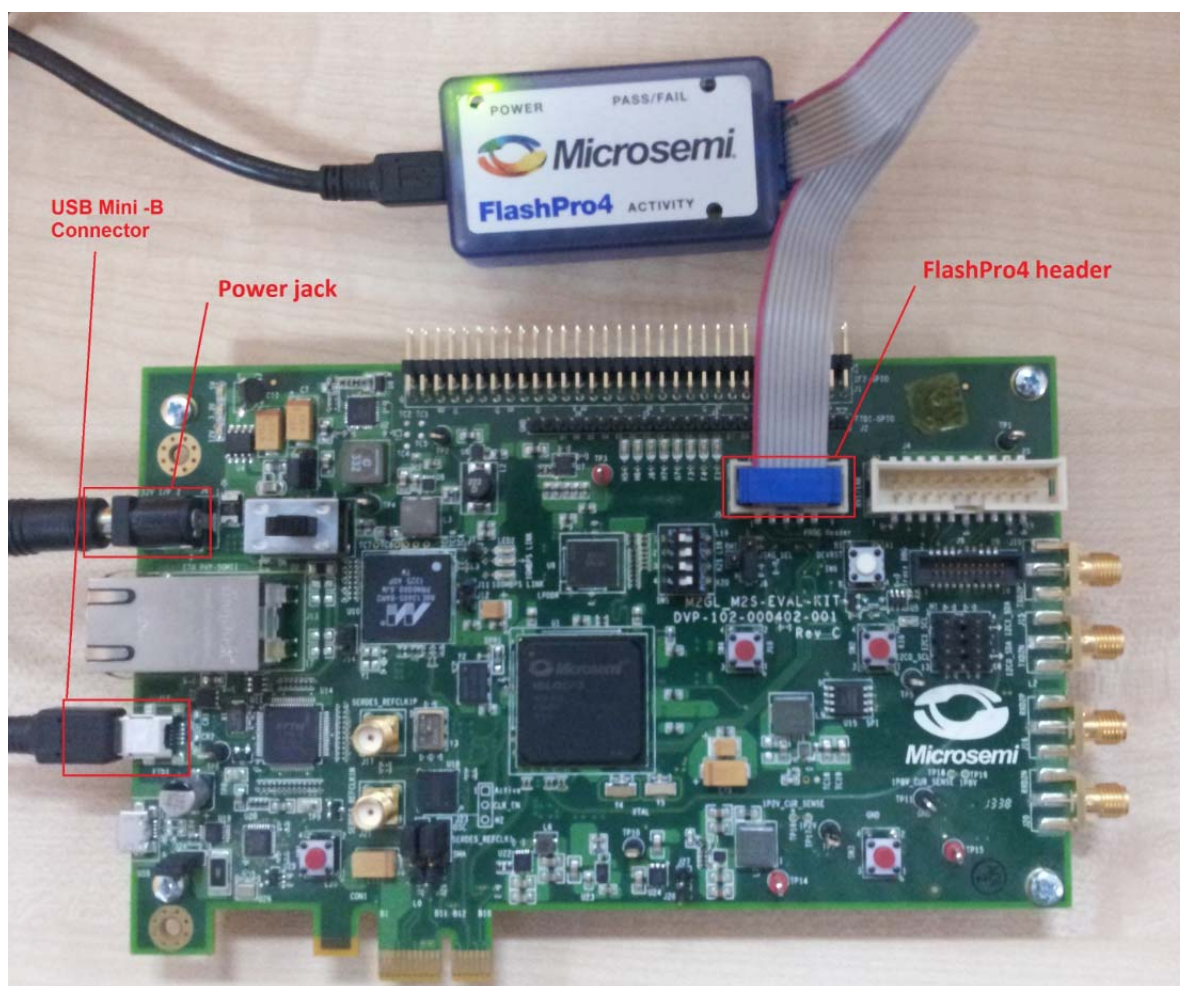


Figure 9 • IGLOO2 Evaluation Kit Board

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. Refer to the [Configuring Serial Terminal Emulation Programs Tutorial](#) for configuring HyperTerminal, TeraTerm, or PuTTY.

3. Program the IGLOO2 Evaluation Kit board with the STAPL file provided as part of the design files download using FlashPro software. Refer to "[Appendix: Design and Programming Files](#)" on page 17 for more information.
4. After programming, HyperTerminal displays a message to choose one of the following:
 - DSN
 - Design version
 - eNVM0 digest

Refer to [Figure 10](#).

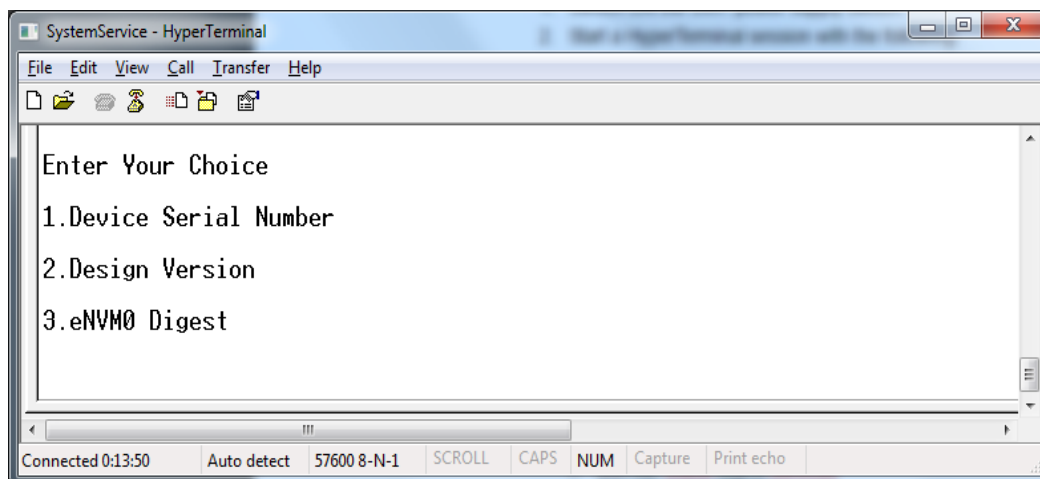


Figure 10 • HyperTerminal Window

5. Select option 1 to get DSN. [Figure 11](#) shows the HyperTerminal displaying the DSN value.

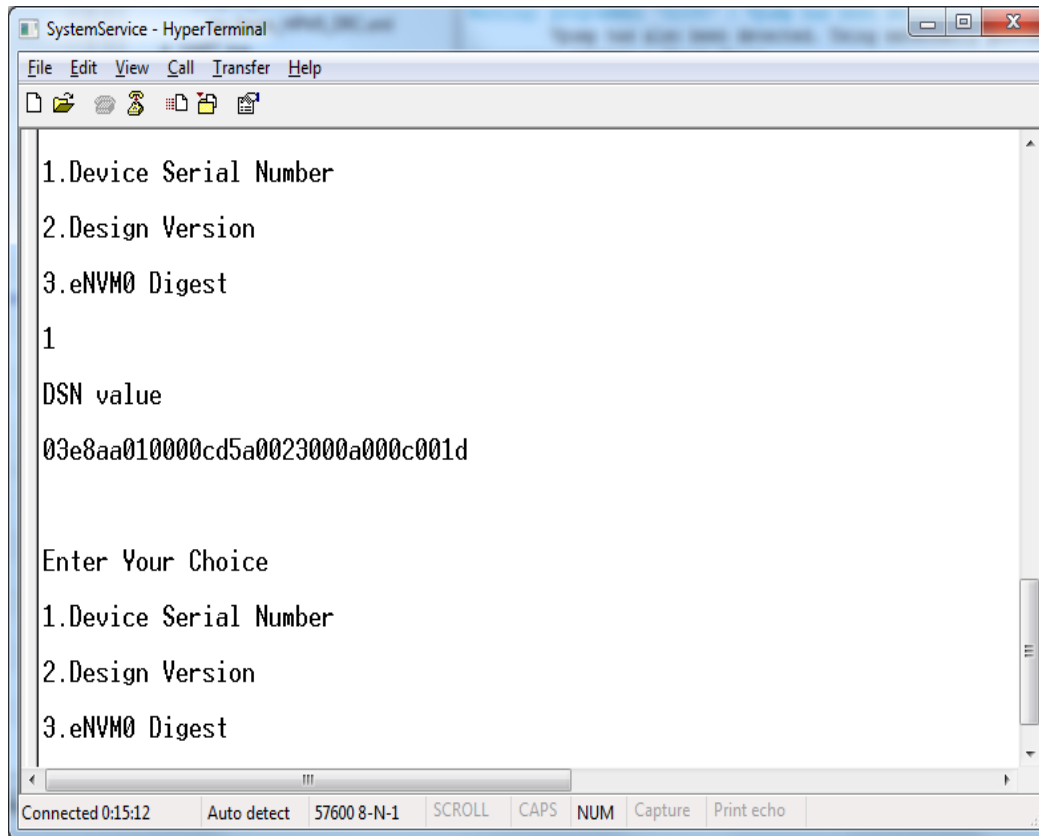


Figure 11 • Device Serial Number

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.

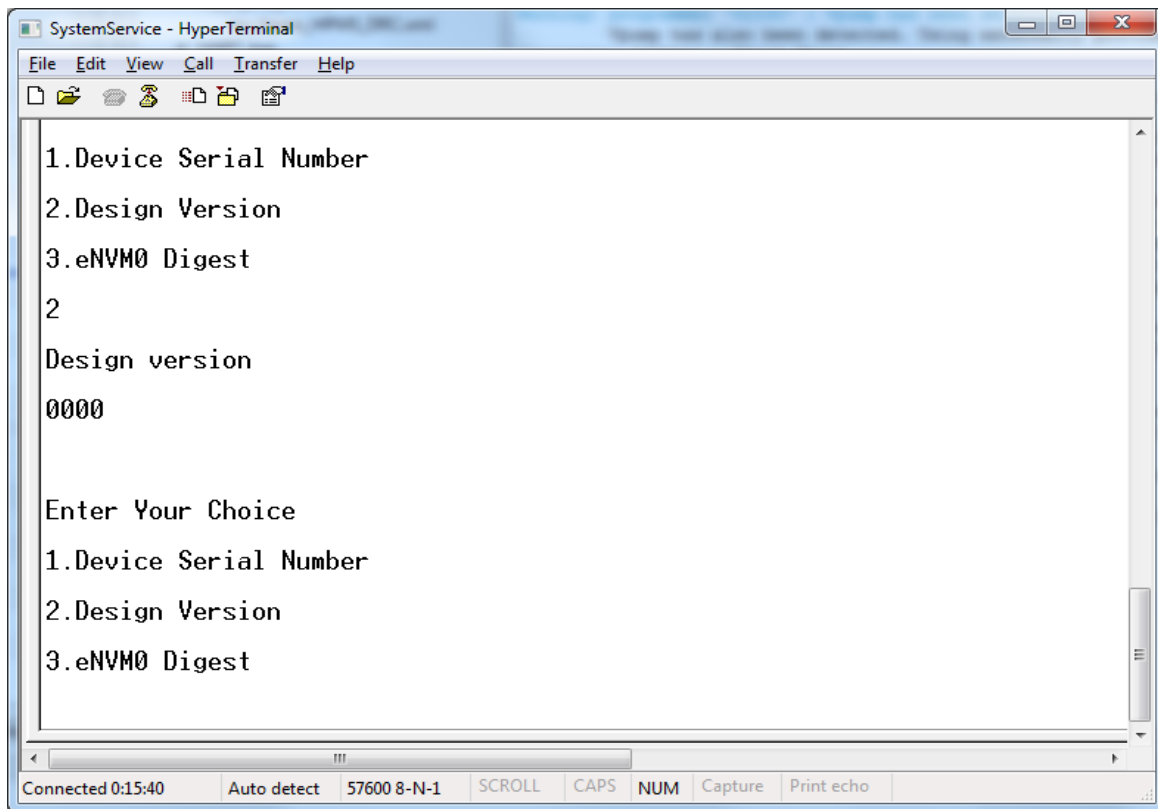


Figure 12 • Design Version

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

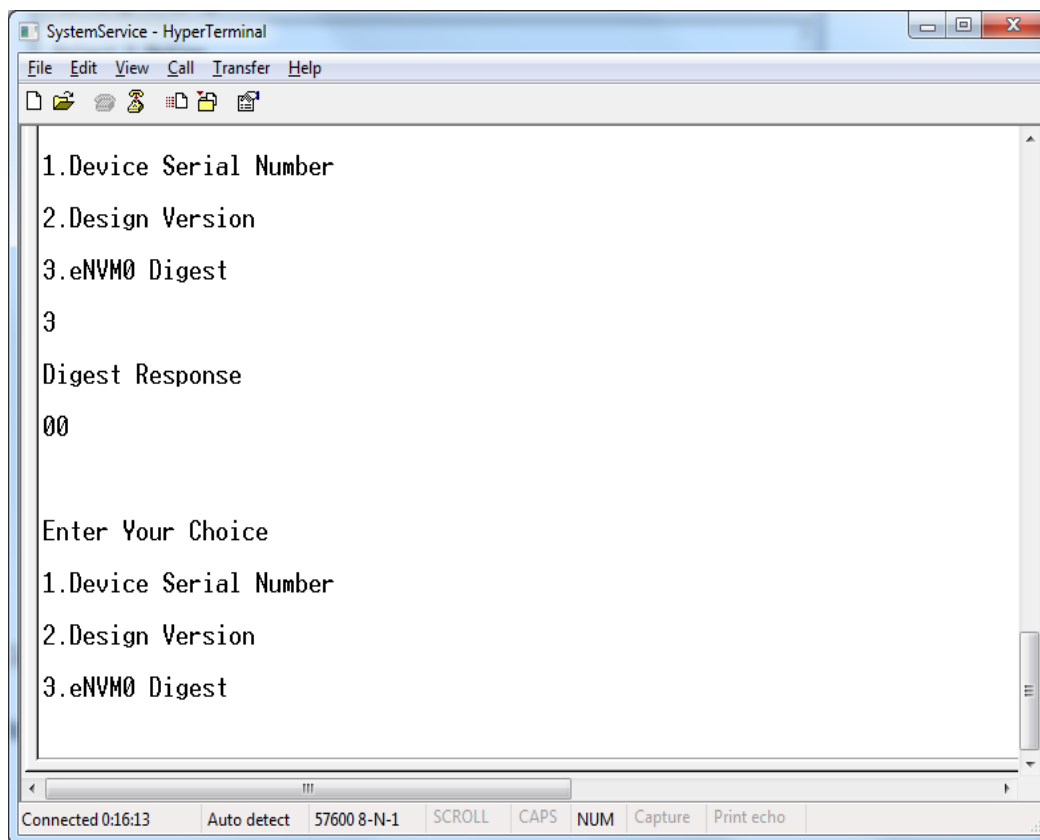


Figure 13 • eNVM0 Digest Response

Conclusion

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

Appendix: Design and Programming Files

Download the design files from Microsemi Website:

http://soc.microsemi.com/download/rsc/?f=m2gl_ac425_liberov11p6_df

The design files consist of a Libero Verilog project and programming files (*.stp) for the IGLOO2 Evaluation Kit. Refer to the `Readme.txt` file included in the design files for the directory structure and description.

Download the programming file (*.stp) in release mode from the Microsemi website:

http://soc.microsemi.com/download/rsc/?f=m2gl_ac425_liberov11p6_pf

The programming zip file consists of the STAPL programming file (*.stp) for IGLOO2 Evaluation Kit.

List of Changes

The following table shows important changes made in this document for each revision.

Date	Changes	Page
Revision 3 (October 2015)	Updated the document for Libero v11.6 software release (SAR 71802).	NA
Revision 2 (January 2015)	Updated the document for Libero v11.5 software release (SAR 63944).	NA
Revision 1 (November 2014)	Initial release.	NA

Note: The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.



Microsemi Corporate 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

E-mail: sales.support@microsemi.com

© 2015 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace 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; 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, Calif., and has approximately 3,600 employees globally. Learn more at 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.