# Table of Contents

Introduction ................................................................................................................... 3

Design Description ....................................................................................................... 4
  Components Used........................................................................................................ 4
  Software Requirements ............................................................................................... 5
  System Requirements ................................................................................................. 5
  Hardware Requirements ............................................................................................. 5
  Preparing for the Lab.................................................................................................. 5

Step 1 – Creating the Design ....................................................................................... 6
  Launching Libero SoC................................................................................................ 6
  Creating the Top-Level in the Canvas........................................................................ 24

Step 2 – Importing a Physical Constraint File .......................................................... 28

Step 3 – Synthesis and Layout .................................................................................... 30

Step 4 – Programming ................................................................................................ 33

Running the Application............................................................................................. 37

Conclusion .................................................................................................................. 39

List of Changes ........................................................................................................... 40

Product Support .......................................................................................................... 41

Customer Service ....................................................................................................... 41

Customer Technical Support Center ......................................................................... 41

Technical Support ....................................................................................................... 41

Website ........................................................................................................................ 41

Contacting the Customer Technical Support Center ............................................... 41
  Email............................................................................................................................. 41
  My Cases....................................................................................................................... 41
  Outside the U.S. ........................................................................................................... 42

ITAR Technical Support ............................................................................................... 42
Introduction

SmartFusion®2 and IGLOO2® field programmable gate array (FPGA) devices have many new security features compared to previous Microsemi or competitive FPGA offerings. From supply chain assurances to enhanced protection of the valuable intellectual property, SmartFusion2 and IGLOO2 FPGA are loaded with unique features, making advanced cryptographic applications easier. SmartFusion2 and IGLOO2 FPGAs add many unique design and data security features and use new models to the PLD industry. The system controller block in the SmartFusion2 and IGLOO2 device manages programming of the device and also handles system service requests to allow design and data security. The system controller serves as the base on which the system services are made available with the SmartFusion2 and IGLOO2 FPGAs. The following are the list of various system services:

- Device and design information services
- Flash*Freeze services
- Cryptographic services
- Differential power analysis (DPA)-resistant key tree services
- Non-deterministic random bit generator services
- Zeroization service
- Programming services

The system services block can be accessed through the communication block (COMM_BLK). There are two COMM_BLK instances: one in the microcontroller subsystem (MSS) or high-performance memory subsystem (HPMS) that the user interfaces with and one that communicates with the first one that is located in the system controller.

The COMM_BLK consists of an advanced peripheral bus (APB) interface, eight byte transmit FIFO, and an eight byte receive FIFO. The COMM_BLK provides a bidirectional message passing facility between the MSS/HPMS and the system controller. The system services are initiated by the user using the COMM_BLK in MSS/HPMS, which can be read or written by any master on the advanced high-performance bus (AHB) matrix; typically either the Cortex™-M3 (in SmartFusion2) or a design in the FPGA fabric (in SmartFusion2/IGLOO2).

The system controller receives the command through the COMM_BLK in the system controller. The system controller then uses the SII Master, a bus master controlled by the system controller, to get the additional details and options at an address supplied in the original COMM_BLK command, pointing where this structured data is stored in memory by the user prior to invoking the command. The system services output bytes returned to the user by the system controller is written to a memory address specified in this data structure. Upon completion of the requested service, the system controller returns a status message through the COMM_BLK.
Design Description
The design consists HPMS, on-chip 50 MHz RC oscillator, Fabric clock conditioning circuitry (CCC), CoreSysServices IP, CoreRESETP, CoreABC, CoreUART_apb, SysServices State Control block, and APB data block. Figure 1 shows the block diagram of the design.

The 50 MHz RC oscillator is used as a main clock. It is used with CCC to provide 100 MHz reference clock to the HPMS. This 100 MHz clock is also used as the main clock for the fabric blocks. The HPMS is configured to use CoreRESETP to generate the reset signals for all the blocks.

The CoreSysServices IP is configured to use only the NRBG. It sends various NRBG commands to the system controller through COMM_BLK in the HPMS. The Sysservice state control logic, controls the sequence of the system service command and capture of the NRBG data from the CoreSysservice. The UART Controller sends the data from the Sysservice state control logic to the HyperTerminal. The APB data block, inside the UART Controller block, also converts the NRBG Hex data to ASCII Hex data to display the NRBG data in the correct format to the HyperTerminal. The CoreABC program controls initiating the SysService Controller Block from the HyperTerminal and displaying the data through CoreUARTapb. The Fabric logic also consists of a counter block (not shown in Figure 1) to display counter value through LEDs that display the design is up and running.

Components Used
This tutorial uses the following SmartFusion2 devices:

- SmartFusion2 FPGA fabric
- On-chip 25/50 MHz RC oscillator
- Fabric CCC.
Software Requirements

- Libero® System-on-Chip (SoC) v11.7 SP1 for viewing design files
- FlashPro programming software (v11.7 SP1)
- USB to serial drivers

System Requirements

Any 64-bit Windows operating system

Hardware Requirements

- SmartFusion2 Development Kit, Rev C that has:
  - FlashPro4 programmer
  - USB A- to Mini-B cable
  - 12 V adapter
- USB-RS232 Serial adapter or RS232 cable

Preparing for the Lab

The demo design files are available for download from the following path in the Microsemi website:
http://soc.microsemi.com/download/rsc/?f=m2s_tu0503_core_sys_svc_liberov11p7_df
Step 1: Creating the Design

In this step create the fabric design using the system builder, SmartDesign and with the supplied source files. The source files are provided in the C:\Coresyssvc_lab\Source_files folder.

Launching Libero SoC

The following steps describe how to launch Libero SoC:

1. Click Start > Programs > Microsemi Libero SoC v11.7 > Libero SoC v11.7, or click the shortcut on your desktop to open the Libero 11.7 Project Manager.

2. Create a new project by selecting New on the Start Page tab (as shown in Figure 2), or by clicking Project > New Project from the Libero SoC menu. This opens the New Project dialog box.

![Figure 2: Libero SoC Project Manager](image)
3. Enter the following information in the **New Project** dialog box.
   - **Project Name:** Sysservice_NRBG
   - **Project Location:** C:\Actelprj\`
   - **Preferred HDL type:** Verilog
   - **Family:** IGLOO2
   - **Die:** M2GL010TS
   - **Package:** 484 FBGA
   - **Speed:** –1
   - **Core Voltage:** 1.2
   - **Range:** IND
   - **Use Design Tool:** checked (as shown in Figure 3)

![Figure 3 Libero SoC New Project Parameters](image)

4. Click **OK** to close the new Project dialog box.
5. Enter `my_hpms` as the name of the system in the **System Builder** dialog box, as shown in Figure 4.

![System Builder Name Dialog Box](image)

**Figure 4** System Builder Name Dialog Box

6. Click **OK**. This opens the **System Builder – Device Features**.

7. In the **System Builder – Device Features** tab, select the **HPMS System Services** check box under **System Services**, as shown in Figure 5.

![System Builder – Device Features](image)

**Figure 5** System Builder – Device Features

**Note:** The IGLOO2 HPMS system builder offers a variety of services including Random Number Generation, Encryption, and Flash“Freeze. To access these services through the HPMS, check the HPMS System Services checkbox in the System Builder – Device Features page. This action exposes a Fabric Master port on the System Builder generated block.
8. Click Next. This opens the System Builder – Peripherals tab as shown in Figure 6. Use the default settings.

![System Builder – Peripherals](image)

**Figure 6** System Builder – Peripherals
9. Click **Next**. This opens the System Builder – Clocks tab, select **On-chip 25/50 Mhz RC Oscillator** as the System clock and **FIC_0_CLK = HPMS/1= 100Mhz**, as shown in **Figure 7**.

![System Builder – Clocks](image)

**Figure 7** System Builder – Clocks

10. Click **Next**. This opens the **System Builder – HPMS Options** tab. Accept the default settings.

11. Click **Next**. This opens the **System Builder – SECDED** tab. Accept the default settings.

12. Click **Next**. This opens the **System Builder – Security** tab. Accept the default settings.

13. Click **Next**. This opens the **System Builder – Memory Map** tab. Accept the default settings.
14. **System Builder** generates the memory, as shown in Figure 8. Verify the memory map settings and click **Finish** to generate the System Builder blocks.
Step 1: Creating the Design

The System Builder block my_hpms is added to the SmartDesign Canvas as my_hpms_top, as shown in Figure 9.

15. Select The Master BIF on my_hpms_top block (mirroredMaster port M on my_hpms_top block), right-click and select Promote to Top Level to promote the signal to top-level, as shown in Figure 10.
16. Promote the other ports on `my_hpms_top` block to top-level, as shown in Figure 11.

**Figure 11**  `my_hpms_0` block in the SmartDesign With All Ports Connected

17. Click `File > Save my_hpms_top` to save `my_hpms_top` smartdesign block, as shown in Figure 12.

**Figure 12**  Saving `my_hpms_top` SmartDesign Block

18. Click `SmartDesign > Generate Component` or select `Generate Component` to generate the `my_hpms_top` block, as shown in Figure 13.

**Figure 13**  Generating `my_hpms_top` SmartDesign Block
Step 1: Creating the Design

After generating the design, end up with two new components in your Design Hierarchy. There is a my_hpms_top, which is a SmartDesign that instantiates system builder generated component my_hpms. The my_hpms_top is a regular SmartDesign that contains an instance of system builder. Double-click the my_hpms_top component in Design Hierarchy to reconfigure the system builder, if needed.

Use several rtl source files in this tutorial. Next, import those source files.

19. Go to the Files tab, right-click on the hdl and select Import Files to open the Import files dialog box, or File>Import > HDL Source Files, as shown in Figure 14.

20. Browse to C:\Coresyssvc_lab\Source_files_blocks\hdl folder and import all four source files: APB_register_blk.v, Count28.v, Hex_to_ascii.v, and NRBG_sysservice_state.v files, as shown in Figure 15.
After successful import, the files appear in the hdl folder and also in the Design Hierarchy tab, as shown in Figure 16.

Next, create a block that instantiates CoreSysServices (configuring CoreSysServices for the NRBG service).

21. Move to the Design Hierarchy tab and click Create SmartDesign in the Design Flow window to create a new SmartDesign block, as shown in Figure 17.
Step 1: Creating the Design

22. This opens the new SmartDesign dialog box. Enter syssvc_blk as the name, as shown in Figure 18.

![Create New SmartDesign dialog box](image)

**Figure 18  Create a New SmartDesign**

The SmartDesign canvas opens. Add CoreSysServices to design the canvas, as shown in Figure 19.

23. Go to the Catalog tab and expand Peripherals in the IP catalog.

![IP catalog](image)

**Figure 19  Create a New SmartDesign Dialog Box**

24. Drag an instance of the CoreSysServices v3.1.101 component into the SmartDesign canvas. SmartDesign uses the CORESYSSERVICES _0 instance name.

25. Double-click the CoreSysServices _0 component in the SmartDesign canvas to open the CoreSysServices Configurator.

26. Select the NRBG under the Data Security Services and enter the following settings:

   - Pointer to instantiate structure: 0x20001000
   - Pointer to RBG personalization string in MSS address space: 0x20002000
   - Pointer to DRBG generate structure: 0x20003000
   - Pointer to buffer to receive generated random data: 0x20004000
   - Pointer to DRBGINSTANTIATE structure: 0x20005000
   - Pointer to additional input data: 0x20006000
The pointer address points to various locations in the eSRAM, CoreSystemServices, and COMM_BLK. Use these locations to transfer data during various NRBG services operation.

27. Click **OK** to close the CoreSysServices.
28. Drag `my_sysservice_state` instance from the Design Hierarchy tab to the SmartDesign canvas. SmartDesign adds `my_sysservice_state` and use `my_sysservice_state_0` as the instance name, as shown in Figure 21.

Figure 21  Adding NRBG State Machine Code in the SmartDesign Canvas
29. Next, connect the signals between CORESYSERVICES_0 and my_sysservice_state_0. There are several ways to connect signals in the SmartDesign.

- Select SmartDesign > Connection mode, enter Connection Mode, and click and drag from one pin to another.
- Select the two signals you want to connect using the CTRL key, then right-click and select Connect to connect them, as shown in Figure 22.
30. Connect the following signal between CORESYSERVICES_0 and my_sysservice_state_0 block, as shown in Table 1.

<table>
<thead>
<tr>
<th>CoreSysServices_0</th>
<th>my_sysservice_state</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERV_BUSY</td>
<td>SERV_BUSY</td>
</tr>
<tr>
<td>SERV_DATA_WRDY</td>
<td>SERV_DATA_WRDY</td>
</tr>
<tr>
<td>SERV_DATA_RVALID</td>
<td>SERV_DATA_RVALID</td>
</tr>
<tr>
<td>SERV_DATA_R[31:0]</td>
<td>SERV_DATA_R[31:0]</td>
</tr>
<tr>
<td>SERV_STATUS_RESP[7:0]</td>
<td>SERV_STATUS_RESP[7:0]</td>
</tr>
<tr>
<td>SERV_STATUS_VALID</td>
<td>SERV_STATUS_VALID</td>
</tr>
<tr>
<td>SERV_DATA_W[31:0]</td>
<td>SERV_DATA_W[31:0]</td>
</tr>
<tr>
<td>SERV_DATA_WVALID</td>
<td>SERV_DATA_WVALID</td>
</tr>
<tr>
<td>SERV_ENABLE_REQ</td>
<td>SERV_ENABLE_REQ</td>
</tr>
<tr>
<td>SERV_CMDBYTE_REQ</td>
<td>SERV_CMDBYTE_REQ</td>
</tr>
<tr>
<td>SERV_NRBG_LENGTH</td>
<td>SERV_NRBG_LENGTH</td>
</tr>
<tr>
<td>SERV_NRBG_HANDLE</td>
<td>SERV_NRBG_HANDLE</td>
</tr>
<tr>
<td>SERV_NRBG_ADDLENGTH</td>
<td>SERV_NRBG_ADDLENGTH</td>
</tr>
<tr>
<td>SERV_NRBG_PRREQ</td>
<td>SERV_NRBG_PRREQ</td>
</tr>
</tbody>
</table>

31. Promote RESETn, START, CLK, SELF_TEST_SERVICE, INSTANTIATE_SERVICE, GENERATE_SERVICE, NRBG_SVC_DONE, NRBG_HANDLE[7:0], NRBG_DATA[31:0], and fsm port from my_sysservice_state_0 to the top-level, as shown in Figure 24.
32. Promote COMM_BLK_INT and AHBL_MASTER interface (M) to top-level, as shown in Figure 24.
33. Connect the following ports between my_syssservice_state and CoreSysServices_0 as shown in Table 2.

<table>
<thead>
<tr>
<th>my_syssservice_state</th>
<th>CoreSysServices_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>CLK and HCLK</td>
</tr>
<tr>
<td>RESETn</td>
<td>RESETN and HRESETN</td>
</tr>
</tbody>
</table>

Figure 24 shows the final syssvc_blk.

34. Select the Generate Component icon ( ) to generate syssvc_blk.
35. Select File > Close syssvc_blk to close the syssvc_blk design canvas.

Next, import the UART_top SmartDesign block. This block converts the NRBG_DATA (Hex value) to ascii hex value and then sends it to the HyperTerminal.
Step 1: Creating the Design

36. Go to the Files tab and select Component > Import Files and this opens the File Import dialog box, as shown in Figure 25.

37. Ensure the File name drop-down is selected as Components (*.cxf) in the Import Files dialog box.

38. Browse to C:\Coresyssvc_lab\Source_files_blocks\UART_top folder and select UART_top.cxf. Click Open to import UART_top SmartDesign block, as shown in Figure 26.

Ignore any warning or error message in the Libero SoC during import. Regenerate the UART_top to update all the components used in UART_top.
39. Select UART_top in the Design Hierarchy tab. Right-click and select Generate Component to generate the UART_top block, as shown in Figure 27.

![Figure 27 Generating the Component in the Libero SoC](image)

40. Open the UART_top block and see the following components:

- **CoreABC_0**: Initiate the system service by sending START_sysservice signal and the type of service through GENERATE_SERVICE, INSTANTIATE_SERVICE, and SELF_TEST_SERVICE ports. It also captures the system service output from APB_register_blk_0 and sends it to the CoreUARTapb_0 block.
- **APB_register_blk_0**: Capture the NRBG data and convert the regular Hex to ascii Hex.
- **CoreUARTapb_0**: Interact with the HyperTerminal. The UART block is configured with the following settings: baud rate 57600, data 8-bit, and parity none, as shown in Figure 28.

![Figure 28 UART_top Block in the SmartDesign Canvas](image)

Next, create the top-level block and connect the three sub-blocks in the SmartDesign canvas to complete the design. As mentioned, the SmartDesign in the Libero SoC has a connection mode that supports click, drag and release to make connections.
Creating the Top-Level in the Canvas

The following steps describe how to create the top-level in the canvas:

1. Double-click Create SmartDesign in the Design Flow tab, the Create New SmartDesign dialog box opens, as shown in Figure 29.

2. Enter Top as the Name and click OK to open the SmartDesign canvas, as shown in Figure 30.
3. Drag an instance of the UART_top, syssvc_blk, my_hpms_top, and counter28 components into the SmartDesign canvas, as shown in Figure 31.

![Figure 31: SmartDesign Canvas After Adding the Components](image)

4. Next, connect the components in the SmartDesign canvas to complete the design.

   Tip: Expand the canvas area by selecting View > Maximize Work Area, or click the icon on the tool bar ( ). Also, expand all the BUS signals.
5. Connect the various signals between the block, as shown in Table 3.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Block 1</th>
<th>Block 2</th>
<th>Block 3</th>
<th>Block 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_DONE</td>
<td>my_hpms_top_0</td>
<td>syssvc_blk_0</td>
<td>UART_top_0</td>
<td>counter28_0</td>
</tr>
<tr>
<td>FIC_0_CLK</td>
<td>RESETn</td>
<td>NSYSRESET</td>
<td>nreset</td>
<td></td>
</tr>
<tr>
<td>COMM_BLK_INT</td>
<td>CLK</td>
<td>PCLK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMM_BLK_INT</td>
<td>–</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>START</td>
<td></td>
<td>START_sysservice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPMS_FIC_0_SYS_SERVICES_MASTER</td>
<td>AHBL_MASTER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRBG_DATA[31:0]</td>
<td></td>
<td>NRBG_DATA[31:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRBG_HANDLE[7:0]</td>
<td></td>
<td>NRBG_HANDLE[7:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRBG_SVC_DONE</td>
<td></td>
<td>NRBG_SVC_DONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GENERATE_SERVICE</td>
<td></td>
<td>GENERATE_SERVICE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INSTANTIATE_SERVICE</td>
<td></td>
<td>INSTANTIATE_SERVICE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELF_TEST_SERVICE</td>
<td>SELF_TEST_SERVICE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. Tie FAB_RESET_N to high. To tie a signal to high, select the signal, right-click and select Tie High, as shown in Figure 32.

7. Promote TX and RX ports on UART_apb_0, cntout[27:24] port on counter28_0, and fsm[3:0] on syssvc_blk_0 to the top-level.
8. After making all the connections listed, top-level Smart Design canvas TOP appears, as shown in Figure 33. Drag the components or use the SmartDesign Auto Arrange feature to improve the appearance of the canvas.

![Figure 33 Tie Signal to the Top Using SmartDesign](image_url)

9. Ensure Top is set as the root in the Design Hierarchy. If it is not set as the root, select Top block in the Design Hierarchy, right-click and select Set As Root.

10. Save the design (File > Save Top).

11. Generate the design by clicking SmartDesign > Generate Component or by clicking the Generate Component icon on the SmartDesign toolbar ( ).

12. Restore the work area (View > Restore Work Area), if you expanded the work area earlier.

13. Confirm that the message Top was successfully generated appears in the Libero Log window.

14. Close the design (File > Close Top).
Step 2: Importing a Physical Constraint File

In this step use a Physical Constraint file to make pin assignments and I/O attribute assignments for the layout. There are multiple ways to make I/O assignments. In this section, import a PDC file that is provided to make the I/O attribute and pin assignments.

1. Expand **Create Constraints** in the **Design Flow** window. Right-click **I/O Constraints** and select **Import Files**, as shown in Figure 34.

2. Enter the following information in the **Import Files** dialog box, then click **Open**:
   - Look in: C:\Actelprj\Coresyssvc_lab\Source_files_blocks\constraint
   - File name: Top.pdc
   - Files of type: Physical Design Constraint Files (*.pdc)

3. Click **Yes** in Information window, as shown in Figure 35.
4. The file is visible under **I/O Constraints** and Create Constraints, as shown in Figure 36.

![Figure 36 PDC Constraint File in the Libero SoC Project](image)

5. Right-click on `constraint\io\Top.pdc` and select **Use for Compile** to use the file for Compile, as shown in Figure 37.

![Figure 37 Applying the Imported PDC File for Compile Operation](image)

6. Double-click the PDC file name on the Files tab to open it in the Libero SoC editor. Scroll in the file to become familiar with the syntax.

A description of the Designer PDC constraints is available in the Libero SoC Help (Help > Implement Design > Constrain Place and Route > Assigning Design Constraints > Design Constraints Guide > Constraints by File Format > PDC Command Reference).
Step 3: Synthesis and Layout

In this step use the push-button flow to synthesize the design with Synplify Pro, run layout, and generate the programming file.

1. Click the Generate Bitstream icon in the Data Flow window (as shown in Figure 38) or select Design > Generate Fabric Programming Data to synthesize the design, run compile and layout using the I/O constraints that are created and generate the programming file.

![Design Flow](image)

**Figure 38** Generate Icon
The design implementation tools run in the batch mode. Successful completion of a design step is indicated by a green check mark next to the **Implement Design in the Design Flow** window, as shown in Figure 39.

![Successful Completion of a Design Implementation](image_url)
Step 3: Synthesis and Layout

The Reports tab displays reports for the tools used to implement the design, as shown in Figure 40.

2. Select the Compile report (Top_compile.xml) under Compile on the Reports tab to view the resource usage. Record the number of sequential and combinatorial cells used in the design below.

- Combinatorial cells (COMB)  ___________________________
- Sequential cells (SEQ)  ___________________________

Figure 40  Reports Tab After Implementing the Design
Step 4: Programming

In this step run FlashPro in batch mode to program the M2GL010TS device in the IGLOO2 Evaluation Kit board.

1. Prior to programming and powering up the IGLOO2 Evaluation Kit board, confirm that the jumpers are positioned, as shown in Table 4.

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Location</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>J3</td>
<td>Above the On/Off Switch</td>
<td>1-2 installed</td>
</tr>
<tr>
<td>J8</td>
<td>Below the JTAG Programming Header (J5)</td>
<td>1-2 installed</td>
</tr>
</tbody>
</table>

2. Plug the FlashPro5 ribbon cable into connector J5 (JTAG Programming Header) on the IGLOO2 Evaluation Kit board.

3. Connect the mini USB cable between the FlashPro4 and the USB port of your PC.

4. Install the FlashPro5 drivers, if prompted. The drivers are located in the <FlashPro Installation Directory>\Drivers folder.
5. Expand Program Design in the **Design Flow** window. Right-click **Run Programming Action** and select **Run** to start programming, as shown in **Figure 42**.

![Figure 42](image)

**Figure 42**  Launching Programming Software From the Design Flow Window

6. FlashPro runs in the batch mode and programs the device. Programming messages is visible in the **Libero SoC Log** window (the programmer number differs).

**CAUTION**: Do not interrupt the programming sequence; it may damage the device or the programmer.
Step 4: Programming

The following message, shown in Figure 43, must be visible in the Reports view under Program Device when the device is programmed successfully (programmer number differs).

programmer '82427': device 'M2GL010T': Executing action PROGRAM PASSED.

Figure 43 Programming Messages in the Libero SoC Log
A green check mark appears next to the **Program Design** and Program Device in the **Design Flow** window, to indicate programming is completed successfully, as shown in Figure 44.

![Design Flow After Programming](image)

**Figure 44** Design Flow After Programming

7. Close the Libero SoC (**Project > Exit** window).
Running the Application

The following steps describe how to run the application:

1. Connect one end of the USB mini-B cable to the J18 connector provided on the IGLOO2 Evaluation Kit board. Connect the other end of the USB cable to the host PC. Ensure that the USB to UART bridge drivers are automatically detected which can be verified in the Device Manager, as shown in Figure 45. If the USB to UART bridge drivers are not installed, download and install the drivers from: http://www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip.

2. From the detected four COM ports, select the one with Location on its properties window as on USB Serial Converter D. Note the COM port number for the serial port configuration.

Figure 45 USB to UART Bridge Drivers
3. Connect the HyperTerminal, select the right COMM port, then select the following settings, as shown in Figure 46.
   - **Bits per second**: 57600
   - **Data bits**: 8
   - **Parity**: None
   - **Stop bits**: 1
   - **Flow control**: None

![COM14 Properties](image)

**Figure 46  Design Flow Window After Programming**

If you do not have a HyperTerminal installed, copy the HyperTerminal folder to local pc from Coresyssvc_lab\Source_files_blocks and double-click hypertrm.exe to launch it.

4. Cycle the power on the board. This is not needed if the HyperTerminal program is running before programming operation is done.
5. Enter 1 to start the DRBG generation. The DRBG value is displayed in the HyperTerminal. Enter 2 to run a self test on the NRBG block in the system controller, as shown in Figure 47.

![HyperTerminal GUI Shows the NRBG Generation](image)

6. When finished, remove power from the board.

**Conclusion**

This tutorial shows how to use CoreSysServices for Non-deterministic random bit generator services in an IGLOO2 design. The design displays the non-deterministic random number generator service (NRBG) values in the HyperTerminal and also describes the following:

- Creating an IGLOO2 design in the Libero SoC
- Using the System Builder to configure HPMS for the System Service
- Using CoreSysServices soft IP for the IGLOO2 design
- Using CoreUART with CoreABC to transfer Hex data to the HyperTerminal
- Using the IGLOO2 Eval-Kit and displaying the NRBG values in the HyperTerminal.
### List of Changes

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Changes</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision 4</td>
<td>Updated the document for Libero v11.7 software release (SAR 75639).</td>
<td>N/A</td>
</tr>
<tr>
<td>(January 2016)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revision 3</td>
<td>No history available on the updates.</td>
<td>N/A</td>
</tr>
<tr>
<td>(October 2014)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world **650. 318.8044**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support


Website


Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.
Outside the U.S.
Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Visit About Us for sales office listings and corporate contacts.

ITAR Technical Support
For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech@microsemi.com. Alternatively, within My Cases, select Yes in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.
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.

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; 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, Calif., and has approximately 4,800 employees globally. Learn more at www.microsemi.com.