Creating a Libero Project for Firmware Catalog Sample Project

Libero SoC v11.5 and SoftConsole Flow Tutorial for SmartFusion2
TU0487 Tutorial

Microsemi®

Table of Contents

Creating a Libero Project for Firmware Catalog Sample Project - Libero SoC v11.5 and SoftConsole Flow Tutorial for SmartFusion2

Introduction

Libero® System-on-Chip (SoC) Firmware catalog shows a list of available firmware cores. Sample projects for each firmware core can be generated from Firmware catalog. A sample project is an example of how the firmware core can be integrated in a project. This sample project contains firmware project using SoftConsole, IAR workbench, and Keil tools. This sample project does not have a Libero project for that generated firmware project. Each sample project folder contains a Readme text file which gives an overview of the design and hardware requirements. Using this information, the Libero SoC project can be generated.

This tutorial describes how to download the SoftConsole sample project from Firmware catalog and create a Libero SoC hardware design for the downloaded sample project. This tutorial provides an example design for System Services.

This tutorial describes the following:

- Downloading SoftConsole sample project from Firmware catalog
- Creating a Libero SoC project
- Generating the programming file
- Opening the project in SoftConsole
- Creating and launching a debug session
- Running the System Services application

Design Requirements

Table 1 • **Design Requirements**

Design Requirements	**Description**
Hardware Requirements	
SmartFusion®2 Security Evaluation Kit • FlashPro4 programmer • USB A to Mini-B cable • 12 V Adapter	Rev D or later
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.5
SoftConsole	v3.4 SPI
FlashPro programming software	v11.5
Host PC Drivers	USB to UART drivers
Any one of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	–

Project Files

The design files for this tutorial can be downloaded from the Microsemi® website: *http://soc.microsemi.com/download/rsc/?f=m2s_tu0487_creating_libero_project_libero11p5_df*

The design files include:

- Libero project
- Programming files
- Readme file

Refer to the `Readme.txt` file provided in the design files for the complete directory structure.

Target Board

SmartFusion2 Security Evaluation Kit Board, Rev D or later.

Design Overview

This tutorial demonstrates the following Device and Design Information services:

- **Serial Number Service**: Fetches the 128-bit device serial number (DSN) and is set during manufacturing.
- **USERCODE Service**: Fetches the programmed 32-bit JTAG USERCODE.
- **User Design Version Service**: Fetches the 16-bit user design version.
- **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.

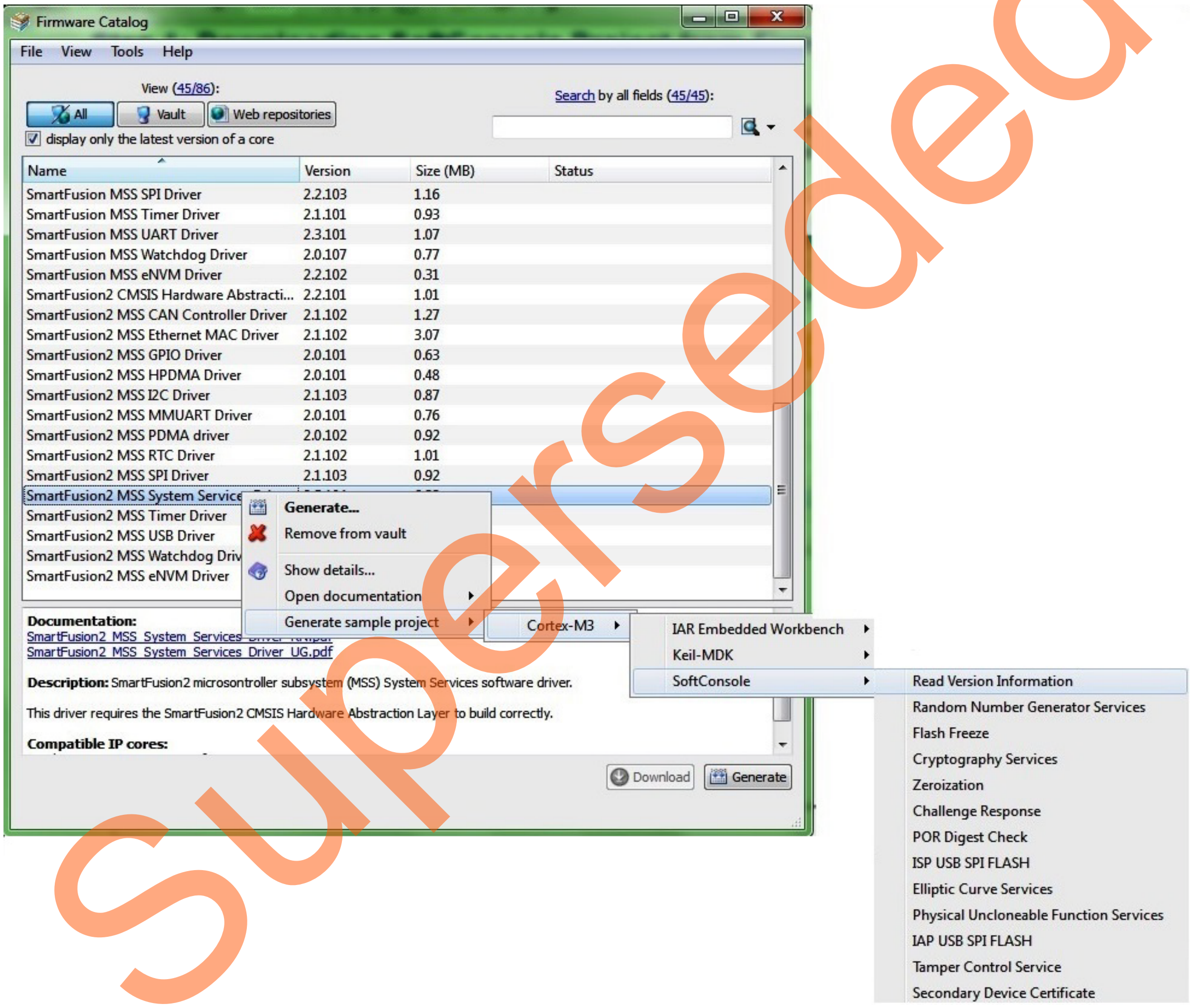
Note: In this tutorial, only fabric digest check is demonstrated.

System Services Information is displayed on HyperTerminal using MMUART_0 interface. For more information on System Services, refer to SmartFusion2 System Controller User Guide.

Downloading SoftConsole Project from Firmware Catalog

The following steps describe how to download the SoftConsole project from Firmware catalog:

1. Click **Start > Programs > Microsemi SoC Libero SoC 11.5 > Firmware Catalog v11.5 > Firmware Catalog**. This opens **Firmware Catalog** windows as shown in Figure 1.
2. Right-click on **SmartFusion2 MSS System Services Driver** and select **Generate Sample Project > Cortex-M3 > SoftConsole > Read Version Information** as shown in Figure 1.



Name	Version	Size (MB)	Status
SmartFusion MSS SPI Driver	2.2.103	1.16	
SmartFusion MSS Timer Driver	2.1.101	0.93	
SmartFusion MSS UART Driver	2.3.101	1.07	
SmartFusion MSS Watchdog Driver	2.0.107	0.77	
SmartFusion MSS eNVM Driver	2.2.102	0.31	
SmartFusion2 CMSIS Hardware Abstracti...	2.2.101	1.01	
SmartFusion2 MSS CAN Controller Driver	2.1.102	1.27	
SmartFusion2 MSS Ethernet MAC Driver	2.1.102	3.07	
SmartFusion2 MSS GPIO Driver	2.0.101	0.63	
SmartFusion2 MSS HPDMA Driver	2.0.101	0.48	
SmartFusion2 MSS I2C Driver	2.1.103	0.87	
SmartFusion2 MSS MMUART Driver	2.0.101	0.76	
SmartFusion2 MSS PDMA driver	2.0.102	0.92	
SmartFusion2 MSS RTC Driver	2.1.102	1.01	
SmartFusion2 MSS SPI Driver	2.1.103	0.92	
SmartFusion2 MSS System Service			
SmartFusion2 MSS Timer Driver			
SmartFusion2 MSS USB Driver			
SmartFusion2 MSS Watchdog Driv			
SmartFusion2 MSS eNVM Driver			

Documentation:
SmartFusion2_MSS_System_Services_Driver_RN.pdf
SmartFusion2_MSS_System_Services_Driver_UG.pdf

Description: SmartFusion2 microsontroller subsystem (MSS) System Services software driver.

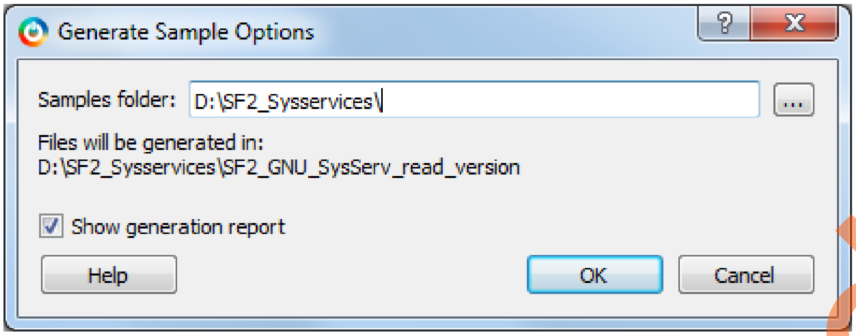
This driver requires the SmartFusion2 CMSIS Hardware Abstraction Layer to build correctly.

Compatible IP cores:

Figure 1 • **Downloading Sample Project from Firmware Catalog**

Note: Select the latest version of the SmartFusion2 MSS System Services driver.

Generate Sample Options window is displayed as shown in Figure 2.



Samples folder: D:\SF2_Sysservices\

Files will be generated in:
D:\SF2_Sysservices\SF2_GNU_SysServ_read_version

☑ Show generation report

Figure 2 • **Generate Sample Options Window**

3. Browse to a location to save System Services **Read Version Information** SoftConsole Project.
4. Open Readme file provided in the SF2_GNU_SysSer_read_version project folder. Readme file gives target hardware information. Refer to "Appendix 3: Readme File" on page 36 for the Readme file.

Creating a Libero SoC Project

Launching Libero SoC

The following steps describe how to launch Libero SoC:

1. Click **Start > Programs > Microsemi Libero SoC v11.5 > Libero SoC v11.5**, or click the shortcut on desktop to open the Libero 11.5 Project Manager.
2. Create a new project using one of the following options:
   - Select **New** on the **Start Page** tab as shown in Figure 3.
   - Click **Project > New Project** from the Libero SoC menu.

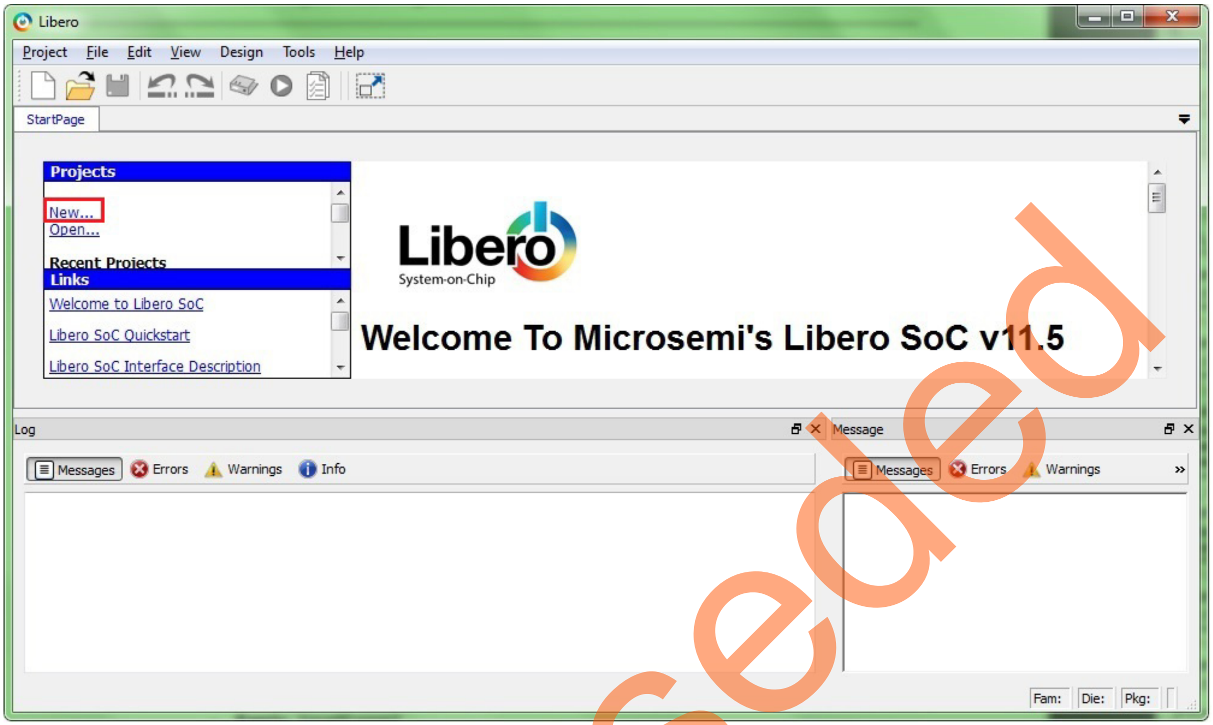
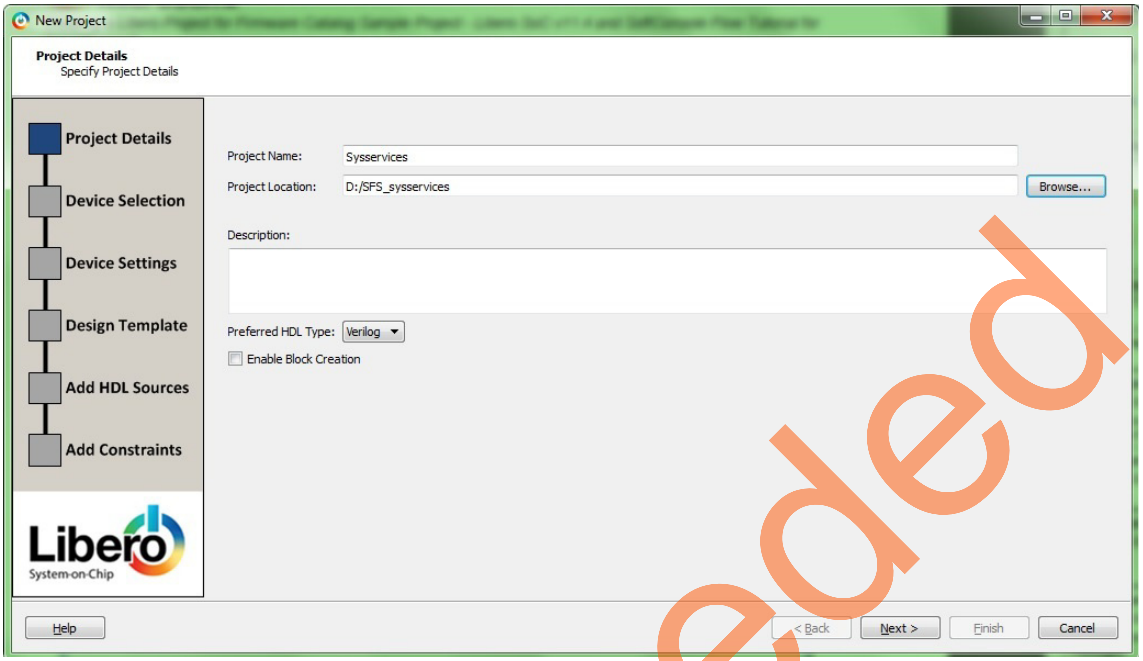


Figure 3 • **Libero SoC Project Manager**

3. Enter the following information in the **Project Details** page, as shown in Figure 4:
   - **Project Name**: Sysservices
   - **Project Location**: Select an appropriate location (for example, D:/SF2_Sysservices)
   - **Preferred HDL Type**: Verilog



New Project

Project Details
Specify Project Details

Project Name: Sysservices

Project Location: D:/SFS_sysservices

Description:

Preferred HDL Type: Verilog

☐ Enable Block Creation

Figure 4 • **Project Details Page**

4. Click **Next.** This opens **Device Selection** page as shown in Figure 5.

   Select the following values from the drop down list:

   – **Family**: SmartFusion2

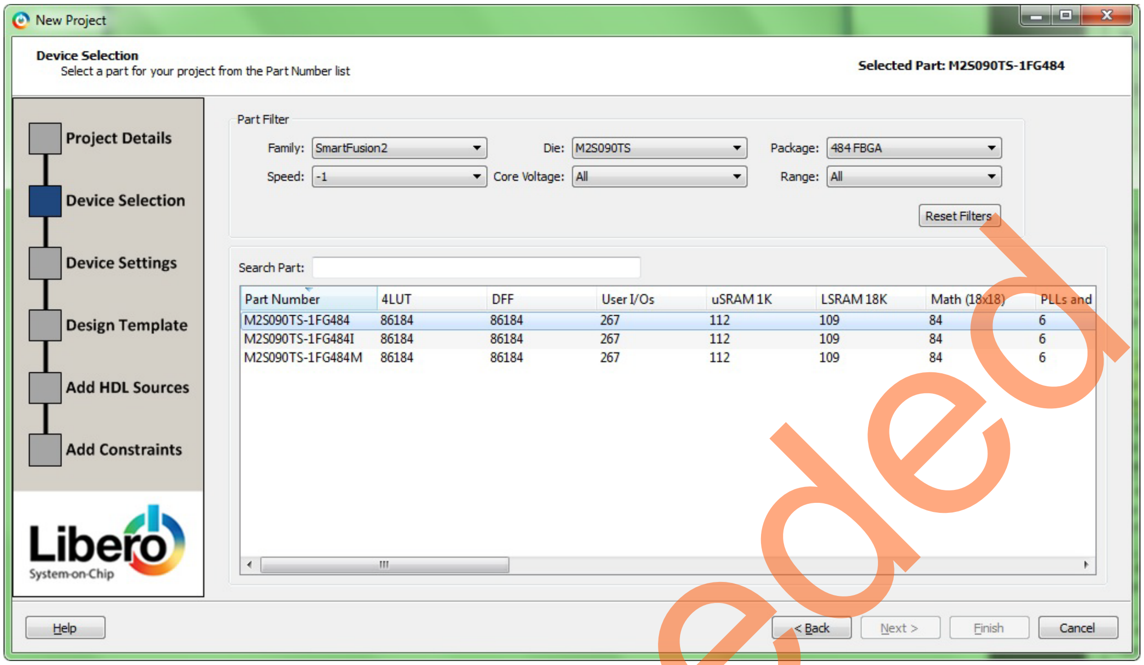
   – **Die**: M2S090TS

   – **Package**: 484 FBGA

   – **Speed**: -1

   – **Core Voltage**: All

   – **Range**: All



Device Selection
Select a part for your project from the Part Number list

Selected Part: M2S090TS-1FG484

Part Filter

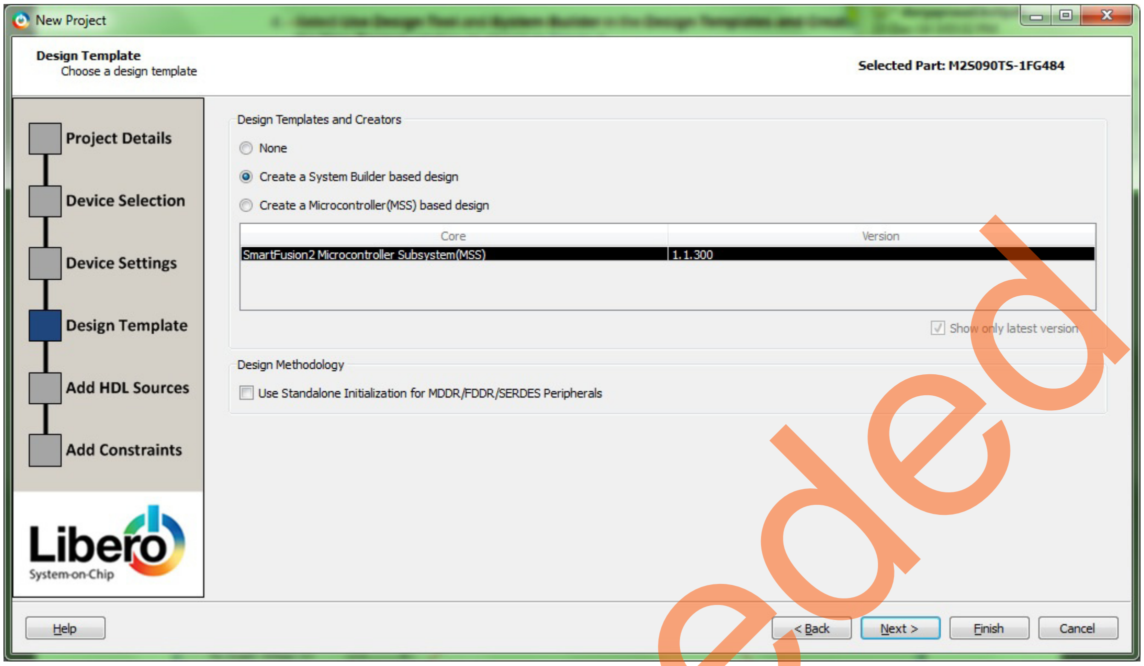
Family: SmartFusion2
Die: M2S090TS
Package: 484 FBGA
Speed: -1
Core Voltage: All
Range: All

Search Part:

Part Number	4LUT	DFF	User I/Os	uSRAM 1K	LSRAM 18K	Math (18x18)	PLLs and
M2S090TS-1FG484	86184	86184	267	112	109	84	6
M2S090TS-1FG484I	86184	86184	267	112	109	84	6
M2S090TS-1FG484M	86184	86184	267	112	109	84	6

Figure 5 • **Device Selection Page**

5. Click **Next**. This opens **Device Settings** page. Do not change the default settings.
6. Click **Next**. This opens **Design Template** page as shown in Figure 6. Under Design Templates and Creators, select **Create a System Builder based design**.



New Project

Design Template
Choose a design template

Selected Part: M2S090TS-1FG484

- Project Details
- Device Selection
- Device Settings
- Design Template
- Add HDL Sources
- Add Constraints

Design Templates and Creators

○ None
● Create a System Builder based design
○ Create a Microcontroller(MSS) based design

Core	Version
SmartFusion2 Microcontroller Subsystem(MSS)	1.1.300

☑ Show only latest version

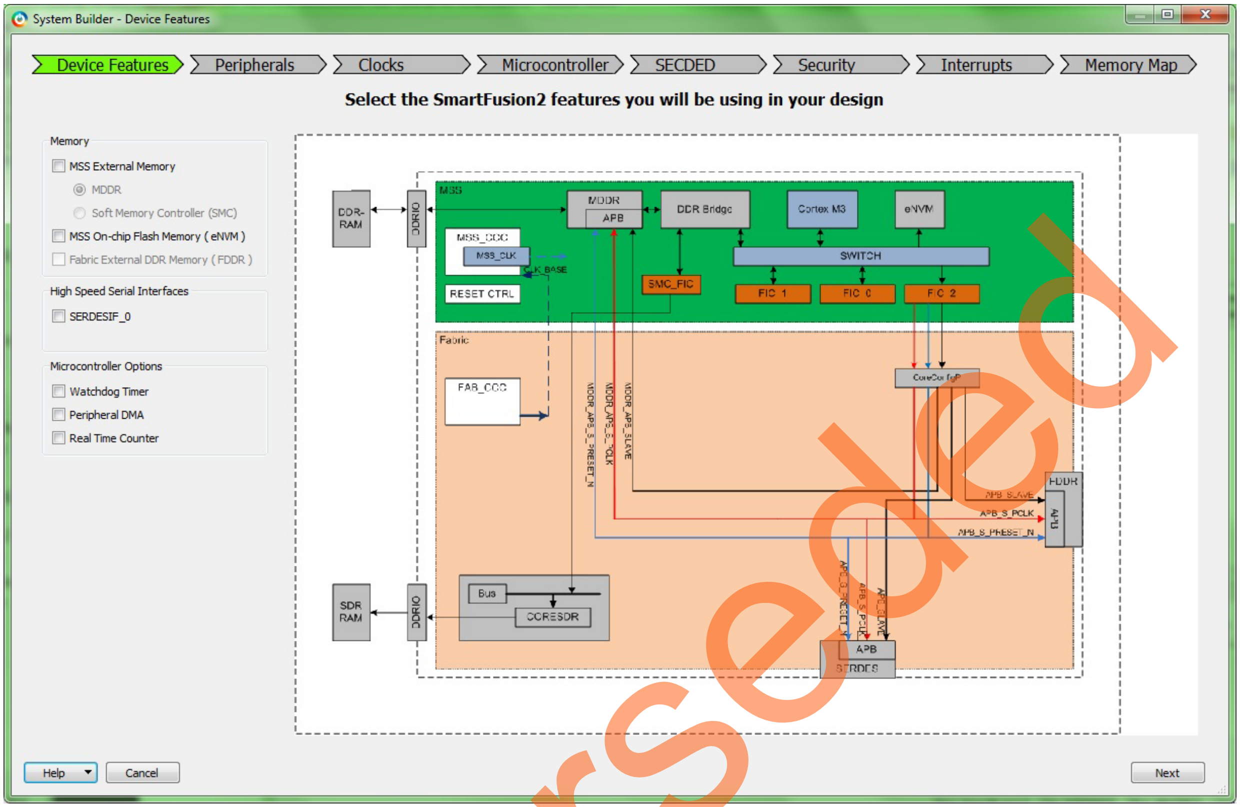
Design Methodology

□ Use Standalone Initialization for MDDR/FDDR/SERDES Peripherals

Figure 6 • **Device Template Page**

7. Click **Finish**. This opens **System Builder** window.
8. Enter the name of the system as **Sysservices** and click **OK**.

Figure 7 shows the **System Builder - Device Features** page.



Select the SmartFusion2 features you will be using in your design

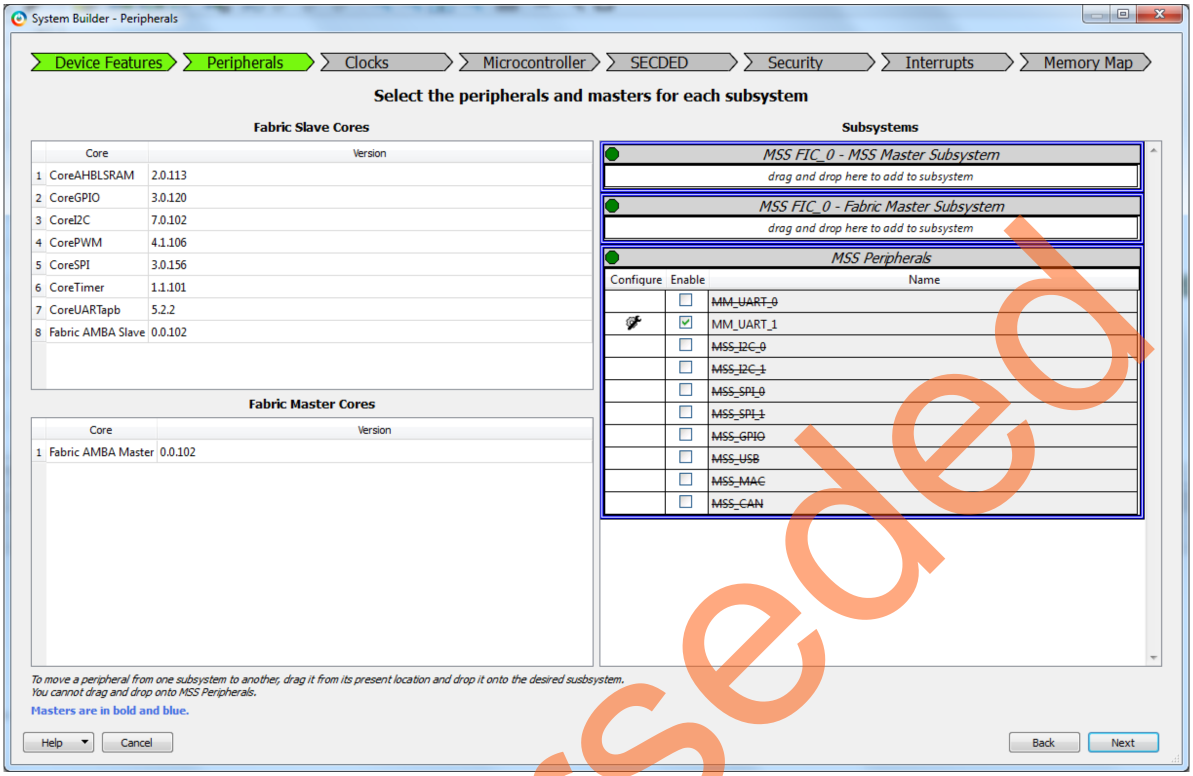
Memory
- □ MSS External Memory
  - ● MDDR
  - ○ Soft Memory Controller (SMC)
- □ MSS On-chip Flash Memory (eNVM)
- □ Fabric External DDR Memory (FDDR)

High Speed Serial Interfaces
- □ SERDESIF_0

Microcontroller Options
- □ Watchdog Timer
- □ Peripheral DMA
- □ Real Time Counter

Figure 7 • **System Builders - Device Features Page**

9. Click **Next**. This opens **System Builder - Peripherals** page as shown in Figure 8.
10. Disable the following peripherals on the **System Builder - Peripherals** page as shown in Figure 8:
    - MMUART_0
    - SPI_0 and SPI_1
    - I2C_0 and I2C_1
    - USB
    - Ethernet
    - CAN



System Builder - Peripherals

Device Features > Peripherals > Clocks > Microcontroller > SECDED > Security > Interrupts > Memory Map

Select the peripherals and masters for each subsystem

Fabric Slave Cores

	Core	Version
1	CoreAHBLSRAM	2.0.113
2	CoreGPIO	3.0.120
3	CoreI2C	7.0.102
4	CorePWM	4.1.106
5	CoreSPI	3.0.156
6	CoreTimer	1.1.101
7	CoreUARTapb	5.2.2
8	Fabric AMBA Slave	0.0.102

Fabric Master Cores

	Core	Version
1	Fabric AMBA Master	0.0.102

Subsystems

MSS FIC_0 - MSS Master Subsystem
drag and drop here to add to subsystem

MSS FIC_0 - Fabric Master Subsystem
drag and drop here to add to subsystem

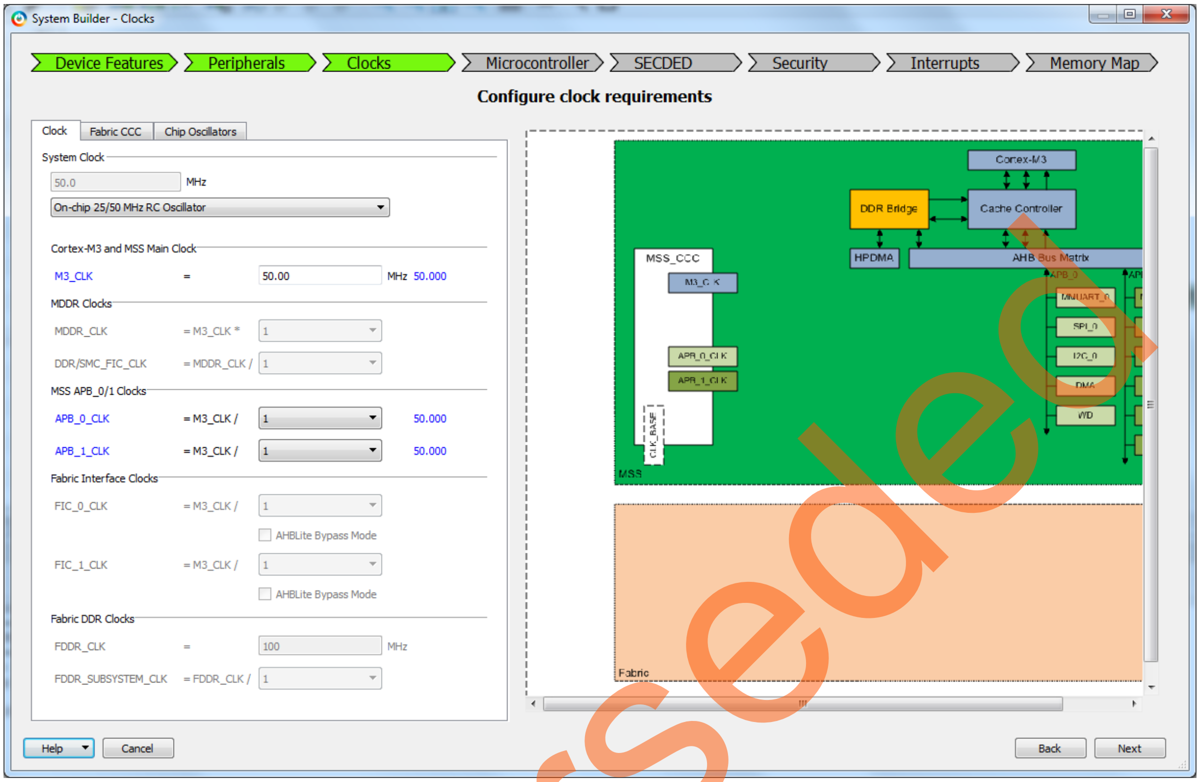
MSS Peripherals

Configure	Enable	Name
	□	~~MM_UART_0~~
	☑	MM_UART_1
	□	~~MSS_I2C_0~~
	□	~~MSS_I2C_1~~
	□	~~MSS_SPI_0~~
	□	~~MSS_SPI_1~~
	□	~~MSS_GPIO~~
	□	~~MSS_USB~~
	□	~~MSS_MAC~~
	□	~~MSS_CAN~~

To move a peripheral from one subsystem to another, drag it from its present location and drop it onto the desired susbsystem.
You cannot drag and drop onto MSS Peripherals.
Masters are in bold and blue.

Figure 8 • **System Builder - Peripherals Page**

11. Click **Next**. This opens **System Builder - Clocks** page as shown in Figure 9.
12. In the **System Builder - Clocks** page (see Figure 9):
    – Select **System Clock** frequency as **50 MHz** and clock source as **On-chip 25/50 MHz RC Oscillator**
    – Select **M3_CLK** as **50 MHz**
    – Select **APB_0_CLK** and **APB_1_CLK** frequency as **M3_CLK/1**
    – Do not change the default settings of remaining parameters.



Configure clock requirements

System Clock

50.0 MHz

On-chip 25/50 MHz RC Oscillator

Cortex-M3 and MSS Main Clock

M3_CLK = 50.00 MHz 50.000

MDDR Clocks

MDDR_CLK = M3_CLK * 1

DDR/SMC_FIC_CLK = MDDR_CLK / 1

MSS APB_0/1 Clocks

APB_0_CLK = M3_CLK / 1 50.000

APB_1_CLK = M3_CLK / 1 50.000

Fabric Interface Clocks

FIC_0_CLK = M3_CLK / 1

□ AHBLite Bypass Mode

FIC_1_CLK = M3_CLK / 1

□ AHBLite Bypass Mode

Fabric DDR Clocks

FDDR_CLK = 100 MHz

FDDR_SUBSYSTEM_CLK = FDDR_CLK / 1

Figure 9 • **System Builder - Clocks Page**

13. Click **Next**. This opens **System Builder - Microcontroller** page. Do not change the default selections.
14. Click **Next**. This opens **System Builder - SECDED** page. Do not change the default selections.
15. Click **Next**. This opens **System Builder - Security** page. Do not change the default selections.
16. Click **Next**. This opens **System Builder - Interrupts** page. Do not change the default selections.
17. Click **Next**. This opens **System Builder - Memory Map** page. Do not change the default selections.
18. Click **Finish**.

19. Select **File > Save** to save **Sysservices_sb_0**. Select the **Sysservices** tab on the Smart Design canvas, as shown in Figure 10.

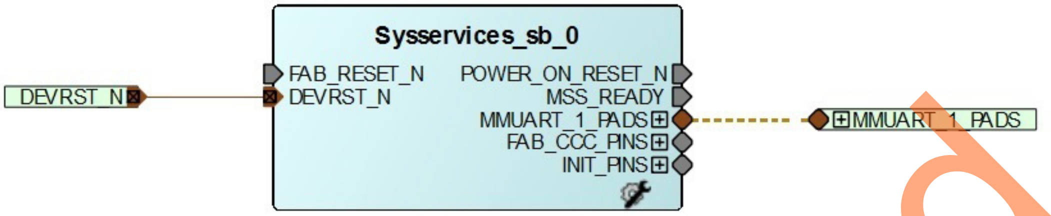


Figure 10 • **Updating Sysservices_sb_0**

Connecting Components in Sysservices_sb_0 SmartDesign

The following steps describe how to connect the components in the **Sysservices_sb_0** SmartDesign:

1. Right-click **POWER_ON_RESET_N** and select **Mark Unused**.
2. Right-click **MSS_READY** and select **Mark Unused**.
3. Expand **INIT_PINS**, right-click **INIT_DONE** and select **Mark Unused**.
4. Expand **FAB_CCC_PINS**, right-click **FAB_CCC_GL0** and select **Mark Unused**.
5. Right-click **FAB_CCC_LOCK** and select **Mark Unused**.
6. Right-click **FAB_RESET_N** and select **Tie High**.
7. Click **File** > **Save**.

The Sysservices_sb_0 design is displayed as shown Figure 11.

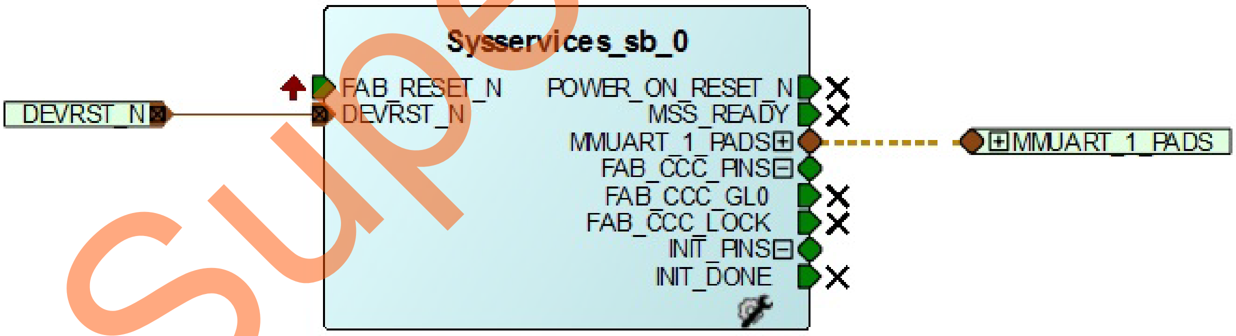
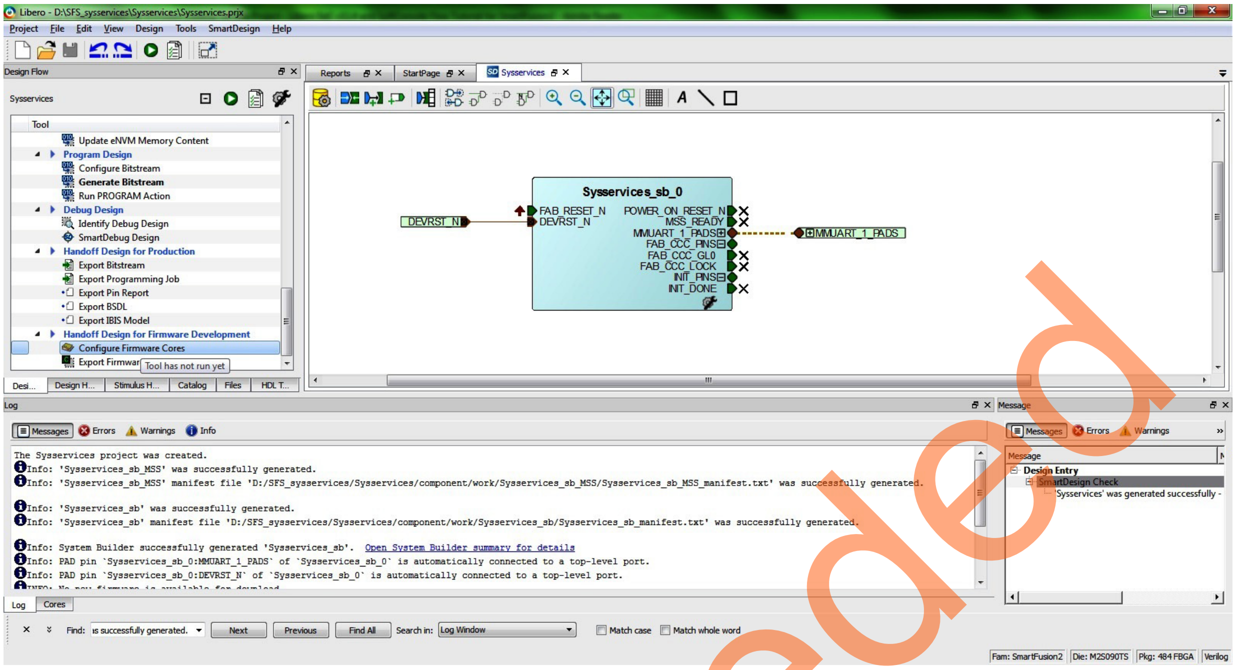


Figure 11 • **Sysservices_sb_0**

Configuring and Generating Firmware

The following steps describes how to configure and generate firmware:

1. In the **Design Flow** tab, double-click on **Configure Firmware Cores** under **Handoff design for Firmware Development**. This opens **DESIGN_FIRMWARE** window as shown in Figure 12.



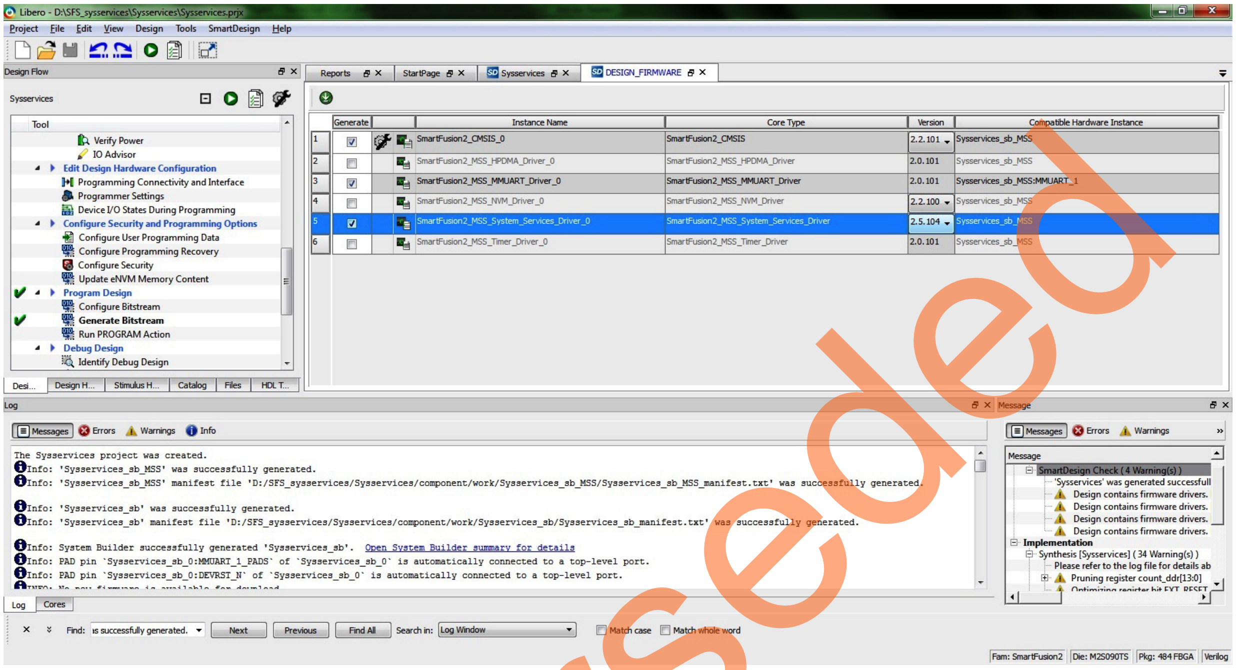
```
The Sysservices project was created.
Info: 'Sysservices_sb_MSS' was successfully generated.
Info: 'Sysservices_sb_MSS' manifest file 'D:/SFS_sysservices/Sysservices/component/work/Sysservices_sb_MSS/Sysservices_sb_MSS_manifest.txt' was successfully generated.

Info: 'Sysservices_sb' was successfully generated.
Info: 'Sysservices_sb' manifest file 'D:/SFS_sysservices/Sysservices/component/work/Sysservices_sb/Sysservices_sb_manifest.txt' was successfully generated.

Info: System Builder successfully generated 'Sysservices_sb'.  Open System Builder summary for details
Info: PAD pin `Sysservices_sb_0:MMUART_1_PADS` of `Sysservices_sb_0` is automatically connected to a top-level port.
Info: PAD pin `Sysservices_sb_0:DEVRST_N` of `Sysservices_sb_0` is automatically connected to a top-level port.
INFO: No new firmware is available for download
```

'Sysservices' was generated successfully -

Figure 12 • **Opening DESGIN_FIRMWARE Window**

2. Clear all drivers except CMSIS, MMUART_0, and System Services as shown in Figure 13.

Note: The SoftConsole sample project for System Services driver can also be downloaded from **DESIGN_FIRMWARE** window. Right-click **SmartFusion2_MSS_System_Services_Driver_0** and select **Read Version Information**.


Libero - D:\SFS_sysservices\Sysservices\Sysservices.prjx
Project File Edit View Design Tools SmartDesign Help
Design Flow
Sysservices
Tool
Verify Power
IO Advisor
Edit Design Hardware Configuration
Programming Connectivity and Interface
Programmer Settings
Device I/O States During Programming
Configure Security and Programming Options
Configure User Programming Data
Configure Programming Recovery
Configure Security
Update eNVM Memory Content
Program Design
Configure Bitstream
Generate Bitstream
Run PROGRAM Action
Debug Design
Identify Debug Design
Desi... Design H... Stimulus H... Catalog Files HDL T...
Reports StartPage Sysservices DESIGN_FIRMWARE
Log
Messages Errors Warnings Info
Log Cores
Message
Messages Errors Warnings

	Generate	Instance Name	Core Type	Version	Compatible Hardware Instance
1	☑	SmartFusion2_CMSIS_0	SmartFusion2_CMSIS	2.2.101	Sysservices_sb_MSS
2	□	SmartFusion2_MSS_HPDMA_Driver_0	SmartFusion2_MSS_HPDMA_Driver	2.0.101	Sysservices_sb_MSS
3	☑	SmartFusion2_MSS_MMUART_Driver_0	SmartFusion2_MSS_MMUART_Driver	2.0.101	Sysservices_sb_MSS:MMUART_1
4	□	SmartFusion2_MSS_NVM_Driver_0	SmartFusion2_MSS_NVM_Driver	2.2.100	Sysservices_sb_MSS
5	☑	SmartFusion2_MSS_System_Services_Driver_0	SmartFusion2_MSS_System_Services_Driver	2.5.104	Sysservices_sb_MSS
6	□	SmartFusion2_MSS_Timer_Driver_0	SmartFusion2_MSS_Timer_Driver	2.0.101	Sysservices_sb_MSS

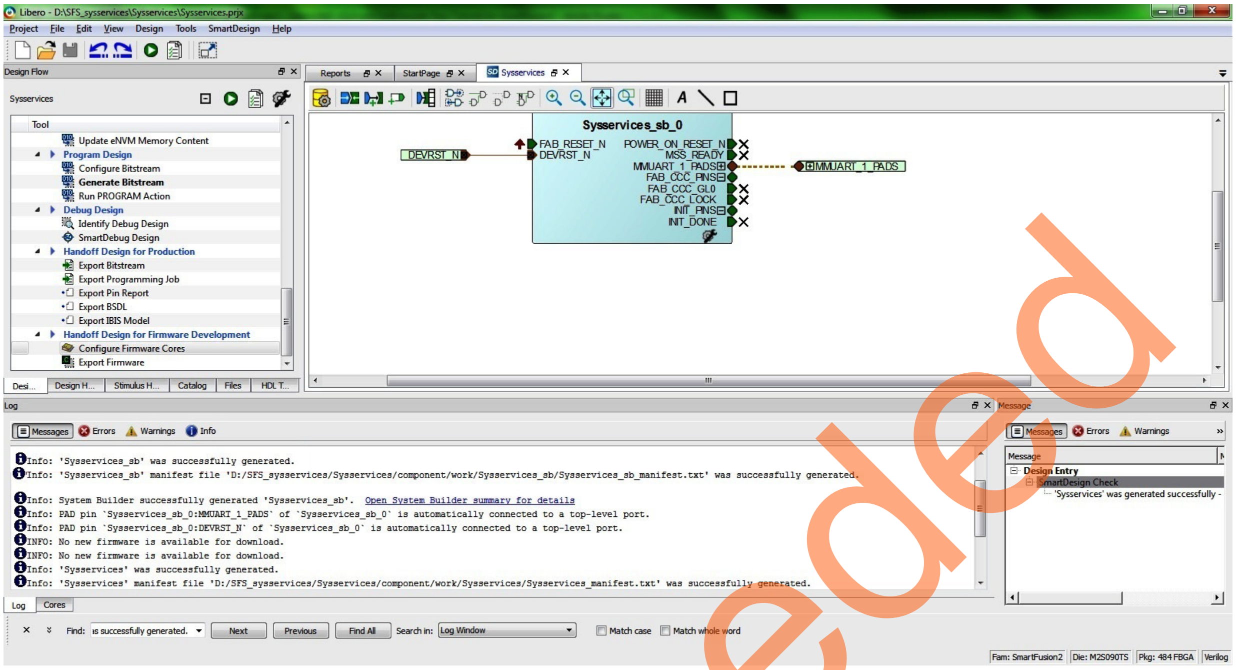
```
The Sysservices project was created.
Info: 'Sysservices_sb_MSS' was successfully generated.
Info: 'Sysservices_sb_MSS' manifest file 'D:/SFS_sysservices/Sysservices/component/work/Sysservices_sb_MSS/Sysservices_sb_MSS_manifest.txt' was successfully generated.

Info: 'Sysservices_sb' was successfully generated.
Info: 'Sysservices_sb' manifest file 'D:/SFS_sysservices/Sysservices/component/work/Sysservices_sb/Sysservices_sb_manifest.txt' was successfully generated.

Info: System Builder successfully generated 'Sysservices_sb'.  Open System Builder summary for details
Info: PAD pin `Sysservices_sb_0:MMUART_1_PADS` of `Sysservices_sb_0` is automatically connected to a top-level port.
Info: PAD pin `Sysservices_sb_0:DEVRST_N` of `Sysservices_sb_0` is automatically connected to a top-level port.
INFO: No new firmware is available for download
```

```
SmartDesign Check ( 4 Warning(s) )
    'Sysservices' was generated successfull
    Design contains firmware drivers.
    Design contains firmware drivers.
    Design contains firmware drivers.
    Design contains firmware drivers.
Implementation
  Synthesis [Sysservices] ( 34 Warning(s) )
    Please refer to the log file for details ab
    Pruning register count_ddr[13:0]
    Optimizing register bit EXT_RESET
```

Find: ıs successfully generated. Next Previous Find All Search in: Log Window □ Match case □ Match whole word

Fam: SmartFusion2 | Die: M2S090TS | Pkg: 484 FBGA | Verilog

Figure 13 • **DESIGN_FIRMWARE Window**


Libero - D:\SFS_sysservices\Sysservices\Sysservices.prjx
Project File Edit View Design Tools SmartDesign Help
Design Flow
Sysservices
Tool
Update eNVM Memory Content
Program Design
Configure Bitstream
Generate Bitstream
Run PROGRAM Action
Debug Design
Identify Debug Design
SmartDebug Design
Handoff Design for Production
Export Bitstream
Export Programming Job
Export Pin Report
Export BSDL
Export IBIS Model
Handoff Design for Firmware Development
Configure Firmware Cores
Export Firmware
Desi... Design H... Stimulus H... Catalog Files HDL T...
Reports StartPage Sysservices
Sysservices_sb_0
DEVRST_N
FAB_RESET_N
DEVRST_N
POWER_ON_RESET_N
MSS_READY
MMUART_1_PADS
FAB_CCC_PINS
FAB_CCC_GL0
FAB_CCC_LOCK
INIT_PINS
INIT_DONE
MMUART_1_PADS
Log
Messages Errors Warnings Info
Log Cores
Find: is successfully generated. Next Previous Find All Search in: Log Window □ Match case □ Match whole word
Message
Messages Errors Warnings
Design Entry
SmartDesign Check
'Sysservices' was generated successfully -
Fam: SmartFusion2 Die: M2S090TS Pkg: 484 FBGA Verilog

```
Info: 'Sysservices_sb' was successfully generated.
Info: 'Sysservices_sb' manifest file 'D:/SFS_sysservices/Sysservices/component/work/Sysservices_sb/Sysservices_sb_manifest.txt' was successfully generated.

Info: System Builder successfully generated 'Sysservices_sb'.  Open System Builder summary for details
Info: PAD pin `Sysservices_sb_0:MMUART_1_PADS` of `Sysservices_sb_0` is automatically connected to a top-level port.
Info: PAD pin `Sysservices_sb_0:DEVRST_N` of `Sysservices_sb_0` is automatically connected to a top-level port.
INFO: No new firmware is available for download.
INFO: No new firmware is available for download.
Info: 'Sysservices' was successfully generated.
Info: 'Sysservices' manifest file 'D:/SFS_sysservices/Sysservices/component/work/Sysservices/Sysservices_manifest.txt' was successfully generated.
```

Figure 14 • **Generate Component**

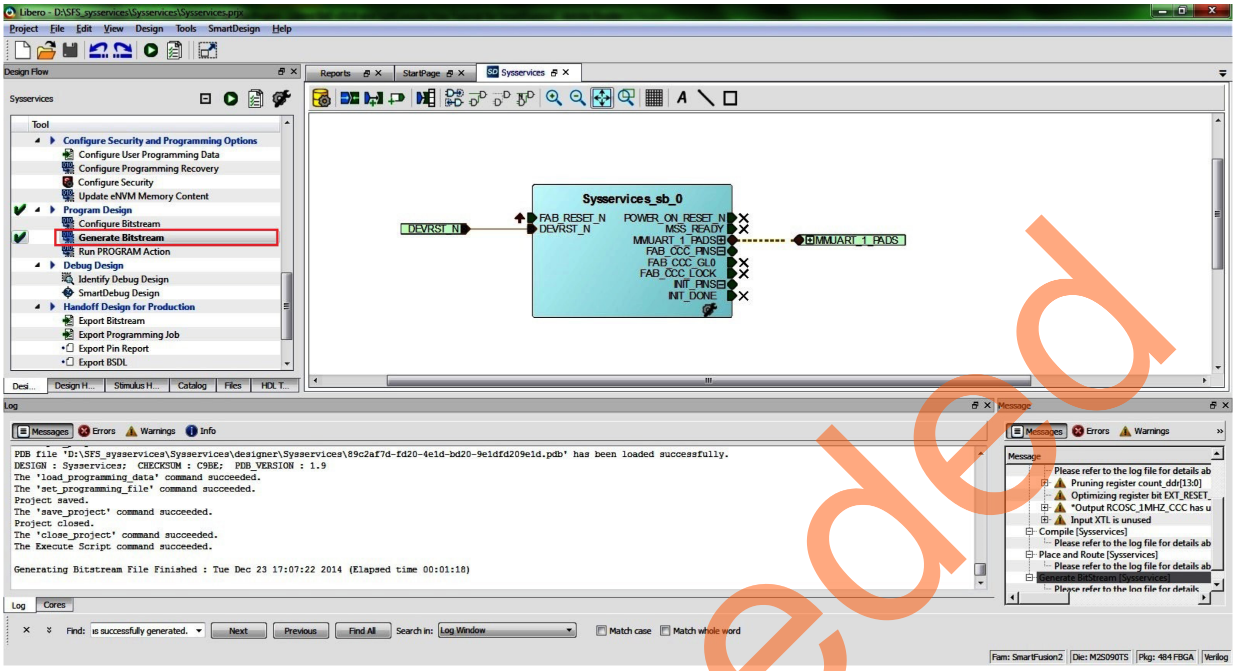
After successful generation of all the components, the following message is displayed on the log window, as shown in Figure 14.

```
Info: 'Sysservices' was successfully generated.
```

Generating the Program File

The following steps describe how to generate the program file:

1. Click **Generate Bitstream** as shown in Figure 15 to complete place and route, and generate the programming file.

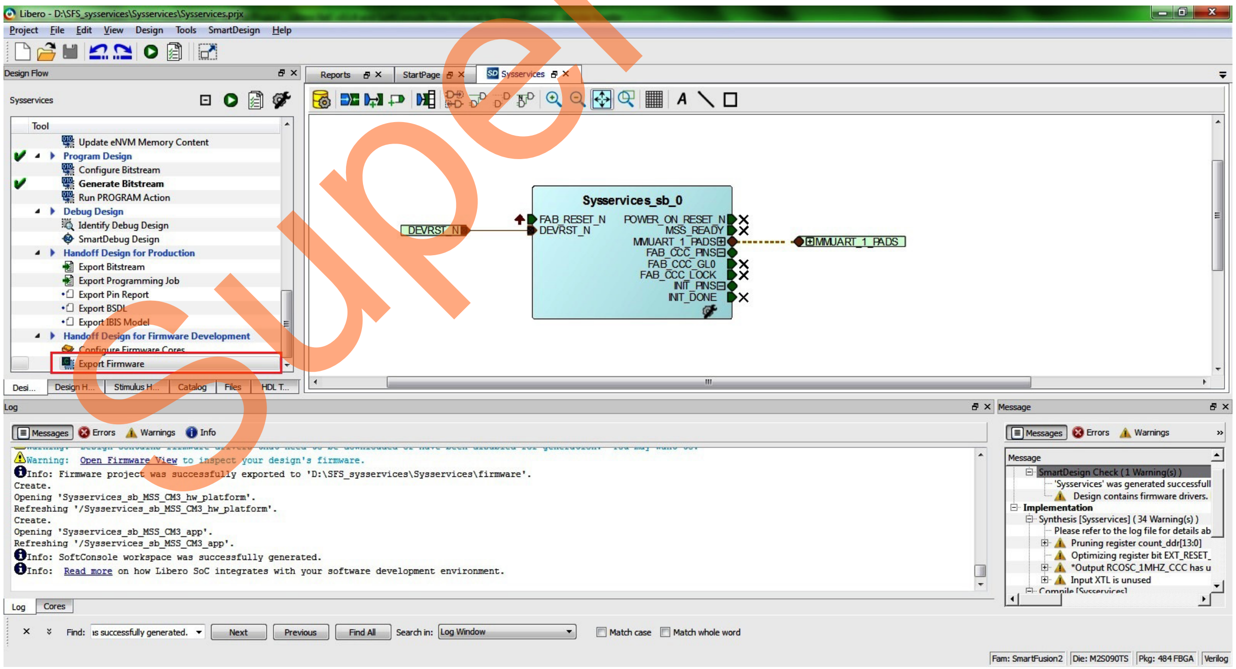

Libero - D:\SFS_sysservices\Sysservices\Sysservices.prjx
Project File Edit View Design Tools SmartDesign Help
Design Flow
Sysservices
Tool
Configure Security and Programming Options
Configure User Programming Data
Configure Programming Recovery
Configure Security
Update eNVM Memory Content
Program Design
Configure Bitstream
Generate Bitstream
Run PROGRAM Action
Debug Design
Identify Debug Design
SmartDebug Design
Handoff Design for Production
Export Bitstream
Export Programming Job
Export Pin Report
Export BSDL
Desi... Design H... Stimulus H... Catalog Files HDL T...
Reports StartPage Sysservices
Sysservices_sb_0
DEVRST_N
FAB_RESET_N
DEVRST_N
POWER_ON_RESET_N
MSS_READY
MMUART_1_PADS
FAB_CCC_PINS
FAB_CCC_GL0
FAB_CCC_LOCK
INIT_PINS
INIT_DONE
MMUART_1_PADS
Log
Messages Errors Warnings Info
Log Cores
Find: is successfully generated. Next Previous Find All Search in: Log Window
Match case Match whole word
Message
Fam: SmartFusion2 Die: M2S090TS Pkg: 484 FBGA Verilog

```
PDB file 'D:\SFS_sysservices\Sysservices\designer\Sysservices\89c2af7d-fd20-4e1d-bd20-9e1dfd209e1d.pdb' has been loaded successfully.
DESIGN : Sysservices;  CHECKSUM : C9BE;  PDB_VERSION : 1.9
The 'load_programming_data' command succeeded.
The 'set_programming_file' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.
The 'close_project' command succeeded.
The Execute Script command succeeded.

Generating Bitstream File Finished : Tue Dec 23 17:07:22 2014 (Elapsed time 00:01:18)
```

```
Message
    Please refer to the log file for details ab
  ⊞ Pruning register count_ddr[13:0]
    Optimizing register bit EXT_RESET_
  ⊞ *Output RCOSC_1MHZ_CCC has u
  ⊞ Input XTL is unused
⊟ Compile [Sysservices]
    Please refer to the log file for details ab
⊟ Place and Route [Sysservices]
    Please refer to the log file for details ab
⊟ Generate BitStream [Sysservices]
    Please refer to the log file for details
```

Figure 15 • Generate Bitstream Data

2. Click **Export Firmware**. This opens **Export Firmware** dialog box as shown in Figure 17.

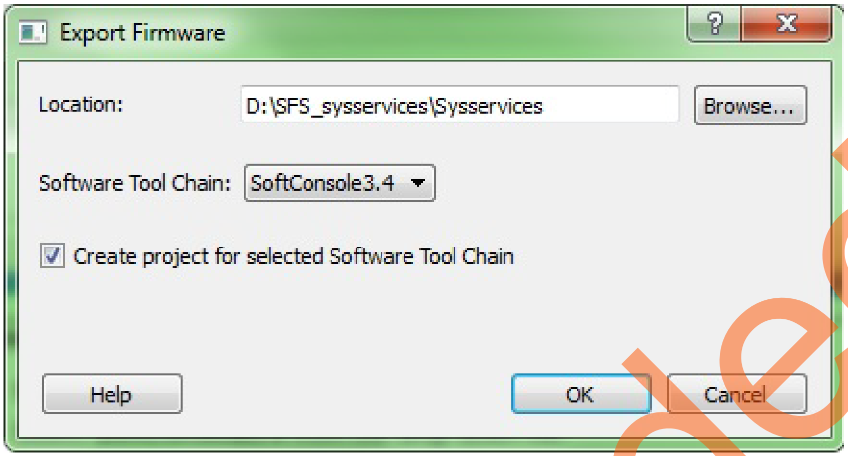

Libero - D:\SFS_sysservices\Sysservices\Sysservices.prjx
Project File Edit View Design Tools SmartDesign Help
Design Flow
Sysservices
Tool
Update eNVM Memory Content
Program Design
Configure Bitstream
Generate Bitstream
Run PROGRAM Action
Debug Design
Identify Debug Design
SmartDebug Design
Handoff Design for Production
Export Bitstream
Export Programming Job
Export Pin Report
Export BSDL
Export IBIS Model
Handoff Design for Firmware Development
Configure Firmware Cores
Export Firmware
Desi... Design H... Stimulus H... Catalog Files HDL T...
Reports StartPage Sysservices
Sysservices_sb_0
DEVRST_N
FAB_RESET_N
DEVRST_N
POWER_ON_RESET_N
MSS_READY
MMUART_1_PADS
FAB_CCC_PINS
FAB_CCC_GL0
FAB_CCC_LOCK
INIT_PINS
INIT_DONE
MMUART_1_PADS
Log
Messages Errors Warnings Info
Log Cores
Find: is successfully generated. Next Previous Find All Search in: Log Window
Match case Match whole word
Message
Fam: SmartFusion2 Die: M2S090TS Pkg: 484 FBGA Verilog

```
Warning:  Design contains firmware drivers that need to be downloaded or have been disabled for generation.  You may want to
Warning:  Open Firmware View to inspect your design's firmware.
Info: Firmware project was successfully exported to 'D:\SFS_sysservices\Sysservices\firmware'.
Create.
Opening 'Sysservices_sb_MSS_CM3_hw_platform'.
Refreshing '/Sysservices_sb_MSS_CM3_hw_platform'.
Create.
Opening 'Sysservices_sb_MSS_CM3_app'.
Refreshing '/Sysservices_sb_MSS_CM3_app'.
Info: SoftConsole workspace was successfully generated.
Info:  Read more on how Libero SoC integrates with your software development environment.
```

```
Message
  ⊟ SmartDesign Check ( 1 Warning(s) )
      'Sysservices' was generated successfull
      Design contains firmware drivers.
⊟ Implementation
  ⊟ Synthesis [Sysservices] ( 34 Warning(s) )
      Please refer to the log file for details ab
    ⊞ Pruning register count_ddr[13:0]
      Optimizing register bit EXT_RESET_
    ⊞ *Output RCOSC_1MHZ_CCC has u
    ⊞ Input XTL is unused
  ⊟ Compile [Sysservices]
```

Figure 16 • Export Firmware

3. In the **Export Firmware** dialog box:
   – Select **Create project for selected Software Tool Chain**.
   – Select **SoftConsole3.4** from the drop down list.



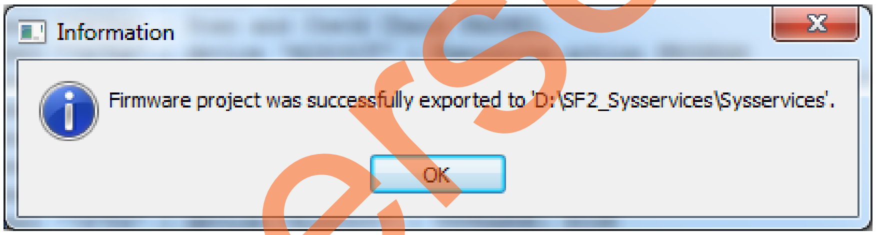
Location: D:\SFS_sysservices\Sysservices

Software Tool Chain: SoftConsole3.4

☑ Create project for selected Software Tool Chain

Figure 17 • **Export Firmware Dialog Box**

4. Click **OK**. The successful firmware generation window is displayed as shown in Figure 18.

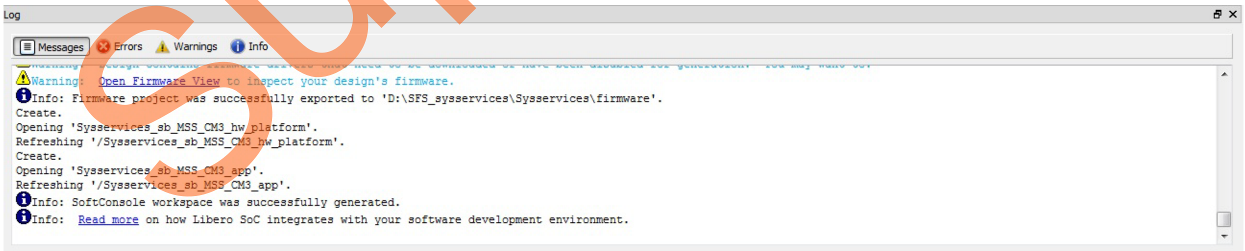


Firmware project was successfully exported to 'D:\SF2_Sysservices\Sysservices'.

Figure 18 • **Firmware Successfully Exported Message**

5. Click **OK**.

The log window is displayed as shown in Figure 19.



```
Warning:  Open Firmware View to inspect your design's firmware.
Info: Firmware project was successfully exported to 'D:\SFS_sysservices\Sysservices\firmware'.
Create.
Opening 'Sysservices_sb_MSS_CM3_hw_platform'.
Refreshing '/Sysservices_sb_MSS_CM3_hw_platform'.
Create.
Opening 'Sysservices_sb_MSS_CM3_app'.
Refreshing '/Sysservices_sb_MSS_CM3_app'.
Info: SoftConsole workspace was successfully generated.
Info:  Read more on how Libero SoC integrates with your software development environment.
```

Figure 19 • **Firmware Log Window**

Programming SmartFusion2 Security Evaluation Board Using FlashPro

The following steps describe how to program the SmartFusion2 Security Evaluation Board using FlashPro:

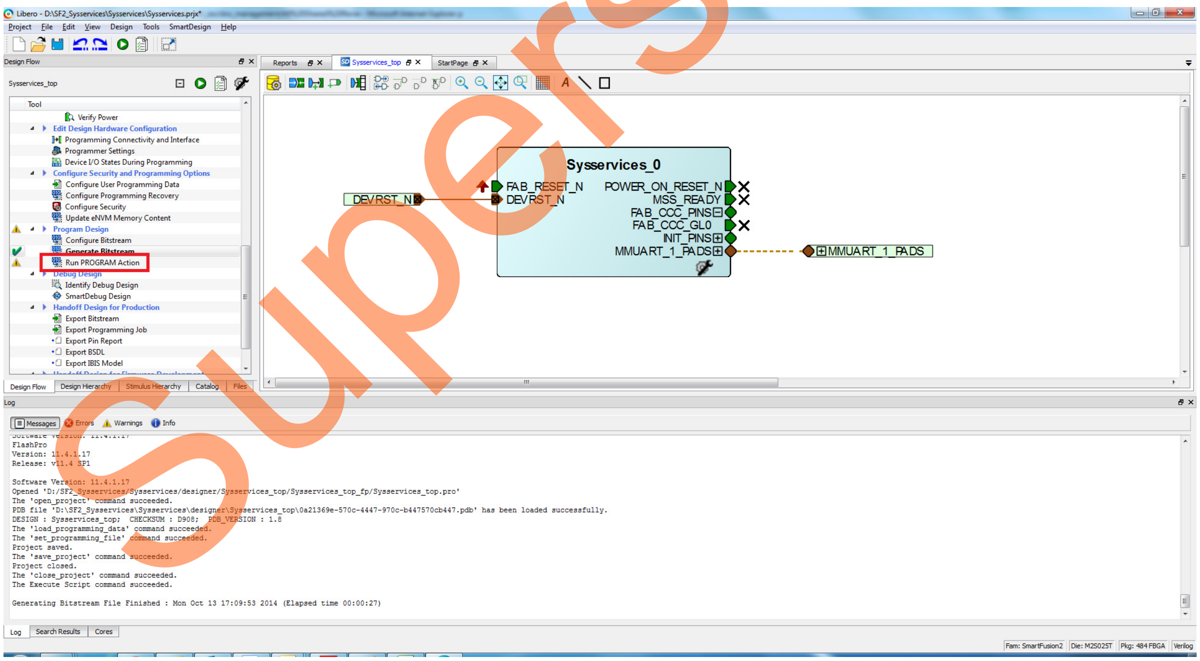
1. Connect the FlashPro4 programmer to the J5 connector of the SmartFusion2 Security Evaluation Kit.
2. Connect the jumpers on the SmartFusion2 Security Evaluation Kit board as per Table 2. For more information on jumper locations, refer to "Appendix 1: Jumper Locations" on page 34.

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

***Table 2 •* SmartFusion2 Security Evaluation Kit Jumper Settings**

Jumper Number	Pin (from)	Pin (to)	Comments
J22, J23, J24, J8, J3	1	2	These are the default jumper settings of the SmartFusion2 Security Evaluation Kit board. Ensure that these jumpers are set accordingly.

3. Connect the power supply to the J6 connector.
4. Switch ON the power supply switch, SW7. Refer to "Appendix 2: Board Setup for Running the Tutorial" on page 35 for information on the board setup for running the tutorial.
5. To program the SmarFusion2 device, double-click **Run PROGRAM Action** in the **Design Flow** tab as shown in Figure 20.



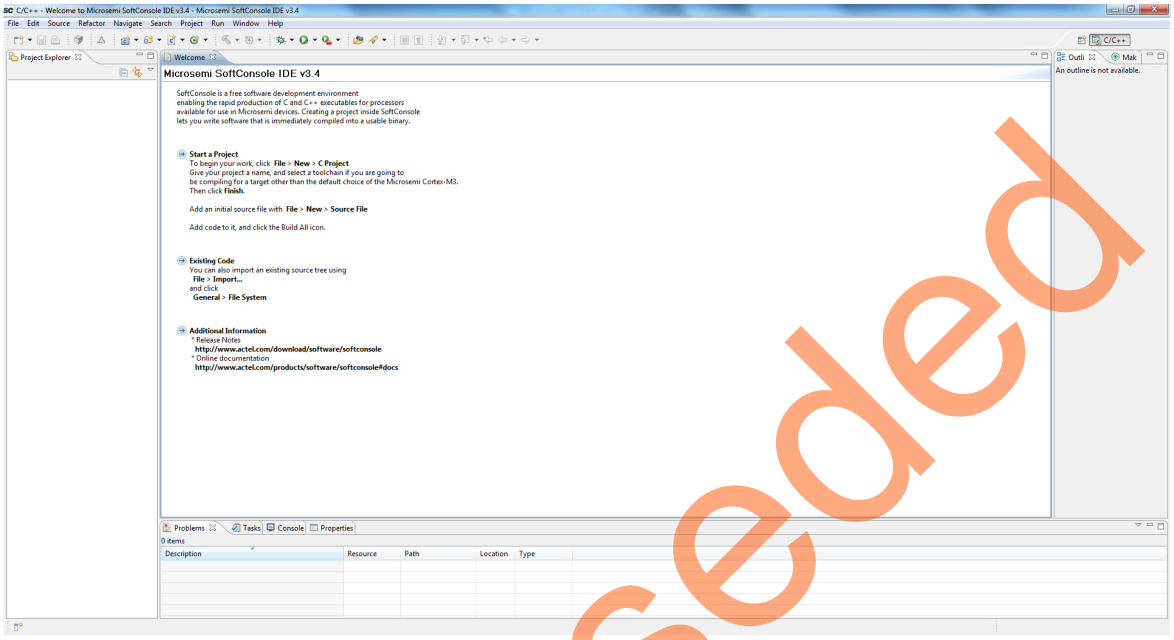
```
Software Version: 11.4.1.17
FlashPro
Version: 11.4.1.17
Release: v11.4 SP1

Software Version: 11.4.1.17
Opened 'D:/SF2_Sysservices/Sysservices/designer/Sysservices_top/Sysservices_top_fp/Sysservices_top.pro'
The 'open_project' command succeeded.
PDB file 'D:\SF2_Sysservices\Sysservices\designer\Sysservices_top\0a21369e-570c-4447-970c-b447570cb447.pdb' has been loaded successfully.
DESIGN : Sysservices_top;  CHECKSUM : D908;  PDB_VERSION : 1.8
The 'load_programming_data' command succeeded.
The 'set_programming_file' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.
The 'close_project' command succeeded.
The Execute Script command succeeded.

Generating Bitstream File Finished : Mon Oct 13 17:09:53 2014 (Elapsed time 00:00:27)
```

***Figure 20 •* Run Program Action**

Building Software Application Using SoftConsole

The following steps describe how to build a software application using SoftConsole:

1. Open the standalone SoftConsole IDE.



Microsemi SoftConsole IDE v3.4

SoftConsole is a free software development environment
enabling the rapid production of C and C++ executables for processors
available for use in Microsemi devices. Creating a project inside SoftConsole
lets you write software that is immediately compiled into a usable binary.

Start a Project
To begin your work, click **File** > **New** > **C Project**
Give your project a name, and select a toolchain if you are going to
be compiling for a target other than the default choice of the Microsemi Cortex-M3.
Then click **Finish**.

Add an initial source file with **File** > **New** > **Source File**

Add code to it, and click the Build All icon.

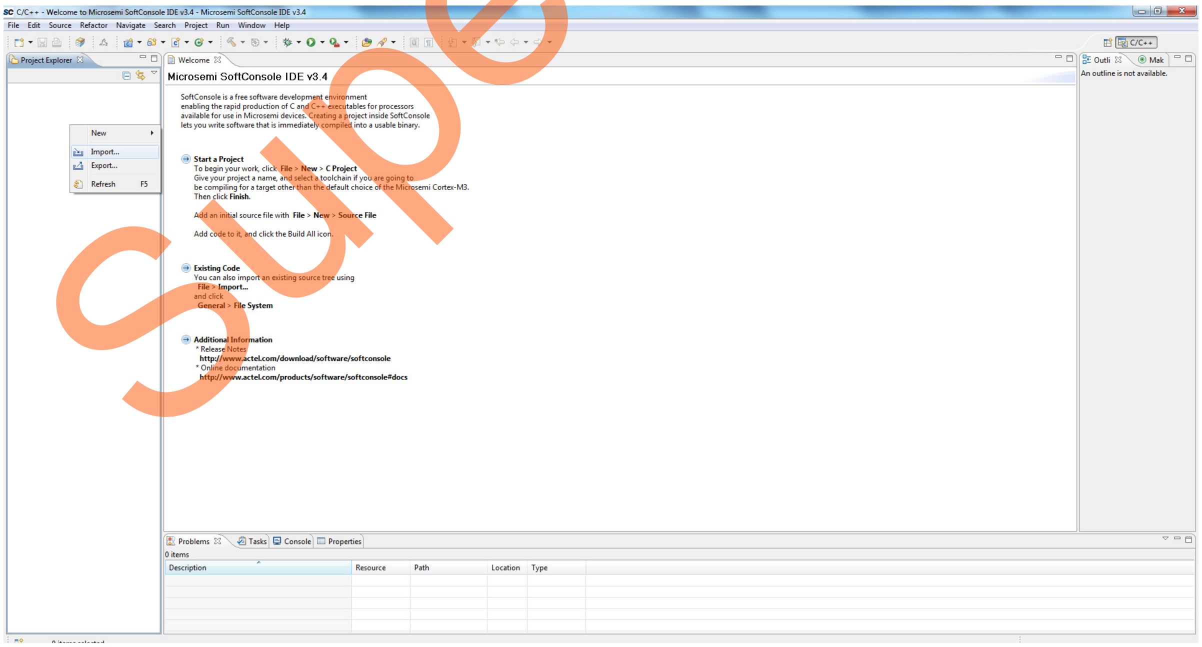
Existing Code
You can also import an existing source tree using
File > **Import...**
and click
General > **File System**

Additional Information
* Release Notes
http://www.actel.com/download/software/softconsole
* Online documentation
http://www.actel.com/products/software/softconsole#docs

An outline is not available.

Figure 21 • **Invoking SoftConsole IDE**

2. Right-click on the **Project Explorer** window and choose **Import** option as shown in Figure 22.



Microsemi SoftConsole IDE v3.4

SoftConsole is a free software development environment
enabling the rapid production of C and C++ executables for processors
available for use in Microsemi devices. Creating a project inside SoftConsole
lets you write software that is immediately compiled into a usable binary.

Start a Project
To begin your work, click **File** > **New** > **C Project**
Give your project a name, and select a toolchain if you are going to
be compiling for a target other than the default choice of the Microsemi Cortex-M3.
Then click **Finish**.

Add an initial source file with **File** > **New** > **Source File**

Add code to it, and click the Build All icon.

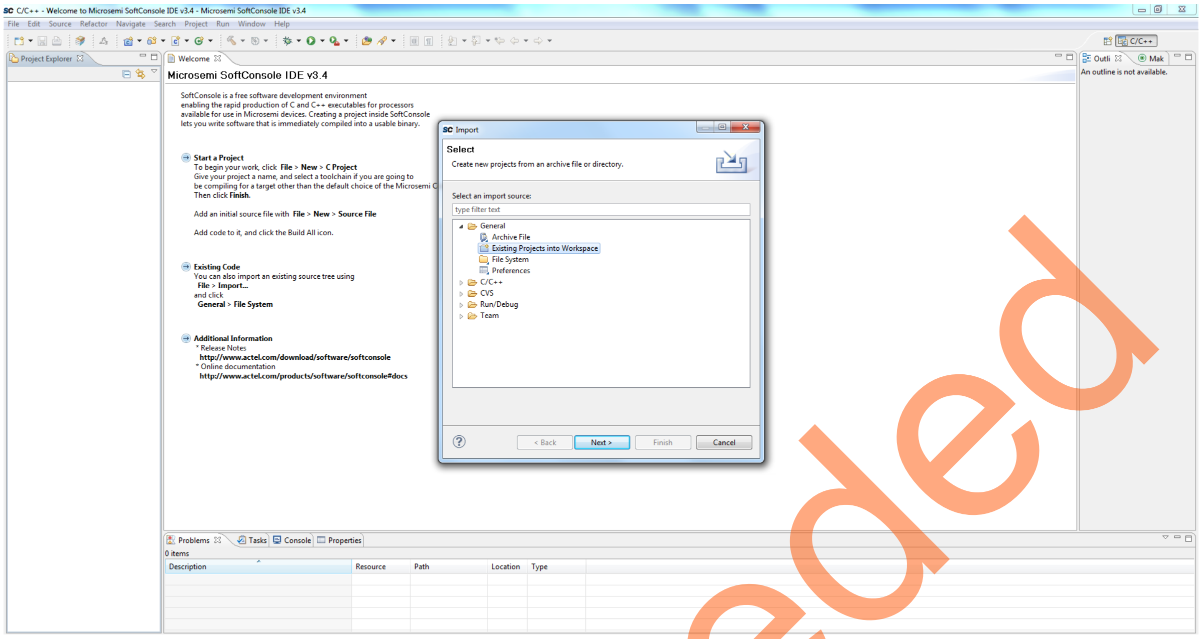
Existing Code
You can also import an existing source tree using
File > **Import...**
and click
General > **File System**

Additional Information
* Release Notes
http://www.actel.com/download/software/softconsole
* Online documentation
http://www.actel.com/products/software/softconsole#docs

An outline is not available.

Figure 22 • **Importing Projects**

3. Select **General > Existing Projects into Workspace** as shown in Figure 23.


SC C/C++ - Welcome to Microsemi SoftConsole IDE v3.4 - Microsemi SoftConsole IDE v3.4

File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Welcome
Outli
Mak
C/C++

An outline is not available.

Microsemi SoftConsole IDE v3.4

SoftConsole is a free software development environment enabling the rapid production of C and C++ executables for processors available for use in Microsemi devices. Creating a project inside SoftConsole lets you write software that is immediately compiled into a usable binary.

Start a Project
To begin your work, click **File** > **New** > **C Project**
Give your project a name, and select a toolchain if you are going to be compiling for a target other than the default choice of the Microsemi C
Then click **Finish**.

Add an initial source file with **File** > **New** > **Source File**

Add code to it, and click the Build All icon.

Existing Code
You can also import an existing source tree using
File > **Import...**
and click
General > **File System**

Additional Information
* Release Notes
http://www.actel.com/download/software/softconsole
* Online documentation
http://www.actel.com/products/software/softconsole#docs

SC Import

Select
Create new projects from an archive file or directory.

Select an import source:
type filter text

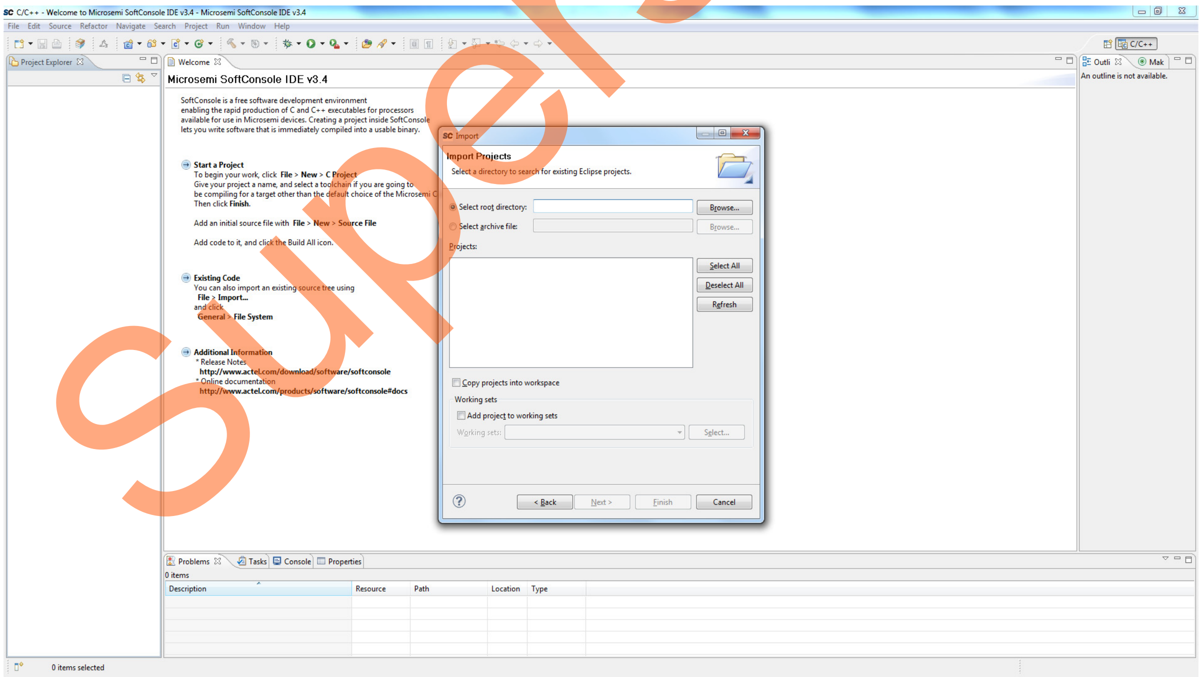
- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- C/C++
- CVS
- Run/Debug
- Team

0 items

Description	Resource	Path	Location	Type

Figure 23 • Importing Existing Projects

Import Window is displayed as shown in Figure 24.


SC C/C++ - Welcome to Microsemi SoftConsole IDE v3.4 - Microsemi SoftConsole IDE v3.4

File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Welcome
Outli
Mak
C/C++

An outline is not available.

Microsemi SoftConsole IDE v3.4

SoftConsole is a free software development environment enabling the rapid production of C and C++ executables for processors available for use in Microsemi devices. Creating a project inside SoftConsole lets you write software that is immediately compiled into a usable binary.

Start a Project
To begin your work, click **File** > **New** > **C Project**
Give your project a name, and select a toolchain if you are going to be compiling for a target other than the default choice of the Microsemi C
Then click **Finish**.

Add an initial source file with **File** > **New** > **Source File**

Add code to it, and click the Build All icon.

Existing Code
You can also import an existing source tree using
File > **Import...**
and click
General > **File System**

Additional Information
* Release Notes
http://www.actel.com/download/software/softconsole
* Online documentation
http://www.actel.com/products/software/softconsole#docs

SC Import

Import Projects
Select a directory to search for existing Eclipse projects.

● Select root directory:
○ Select archive file:

Projects:

□ Copy projects into workspace

Working sets
□ Add project to working sets
Working sets:

0 items

Description	Resource	Path	Location	Type

0 items selected

Figure 24 • Import Window

4. Browse through the Sysservices projects folder and select it as shown in Figure 25.

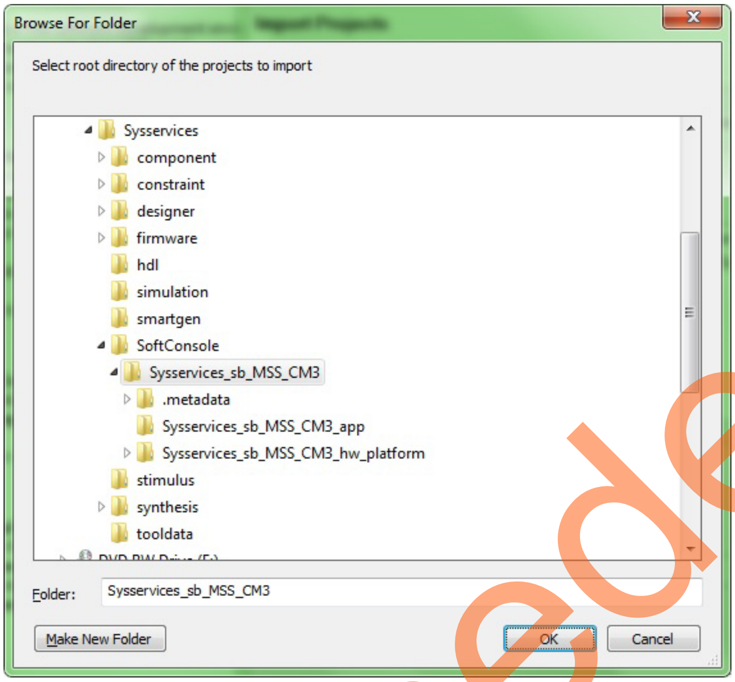
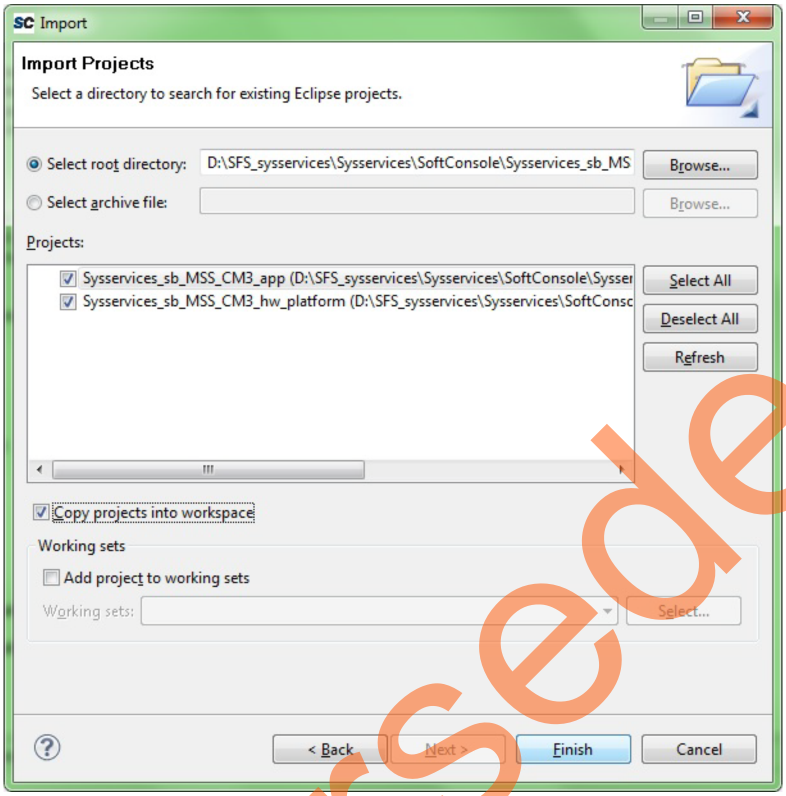


Figure 25 • **Selecting System Services**

5. Click **OK**.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: D:\SFS_sysservices\Sysservices\SoftConsole\Sysservices_sb_MS

○ Select archive file:

Projects:

☑ Sysservices_sb_MSS_CM3_app (D:\SFS_sysservices\Sysservices\SoftConsole\Sysser

☑ Sysservices_sb_MSS_CM3_hw_platform (D:\SFS_sysservices\Sysservices\SoftConsc

☑ Copy projects into workspace

Working sets

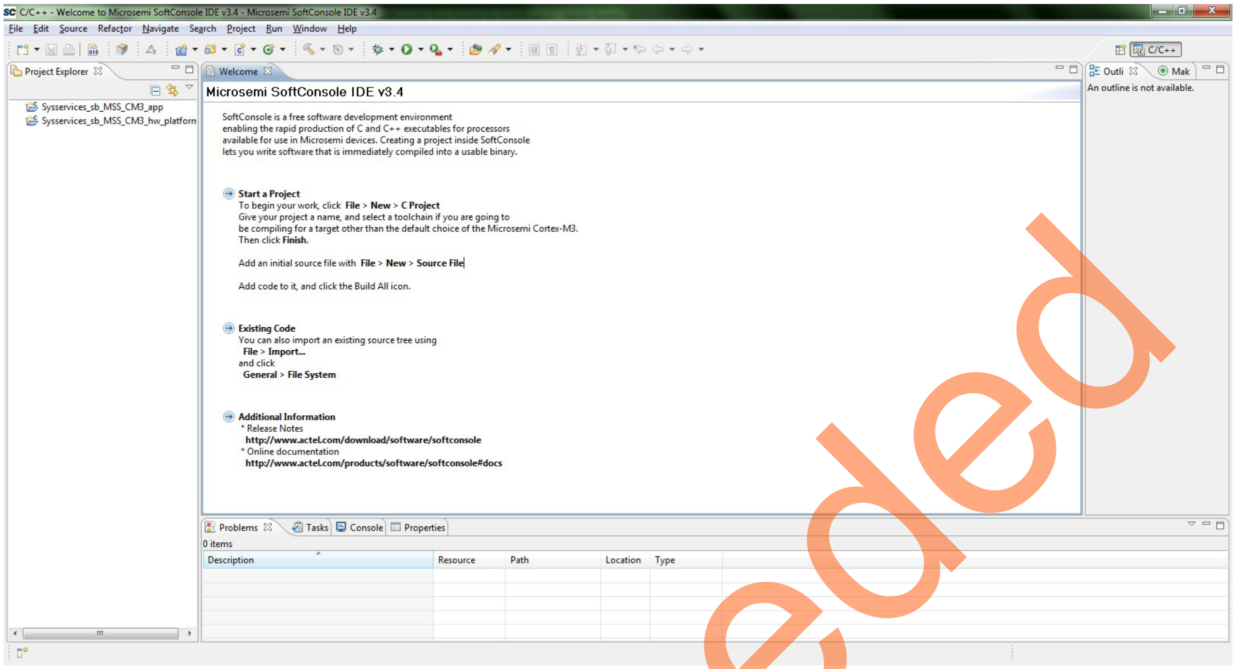
☐ Add project to working sets

Working sets:

Figure 26 • **Adding Projects to SoftConsole IDE**

6. Click **Finish**.

The SoftConsole perspective is displayed as shown in Figure 27.


C/C++ - Welcome to Microsemi SoftConsole IDE v3.4 - Microsemi SoftConsole IDE v3.4
File Edit Source Refactor Navigate Search Project Run Window Help
C/C++
Project Explorer
Sysservices_sb_MSS_CM3_app
Sysservices_sb_MSS_CM3_hw_platform
Welcome
Outli
Mak
An outline is not available.
Problems
Tasks
Console
Properties

Microsemi SoftConsole IDE v3.4

SoftConsole is a free software development environment
enabling the rapid production of C and C++ executables for processors
available for use in Microsemi devices. Creating a project inside SoftConsole
lets you write software that is immediately compiled into a usable binary.

Start a Project
To begin your work, click **File** > **New** > **C Project**
Give your project a name, and select a toolchain if you are going to
be compiling for a target other than the default choice of the Microsemi Cortex-M3.
Then click **Finish.**

Add an initial source file with **File** > **New** > **Source File**

Add code to it, and click the Build All icon.

Existing Code
You can also import an existing source tree using
File > **Import...**
and click
General > **File System**

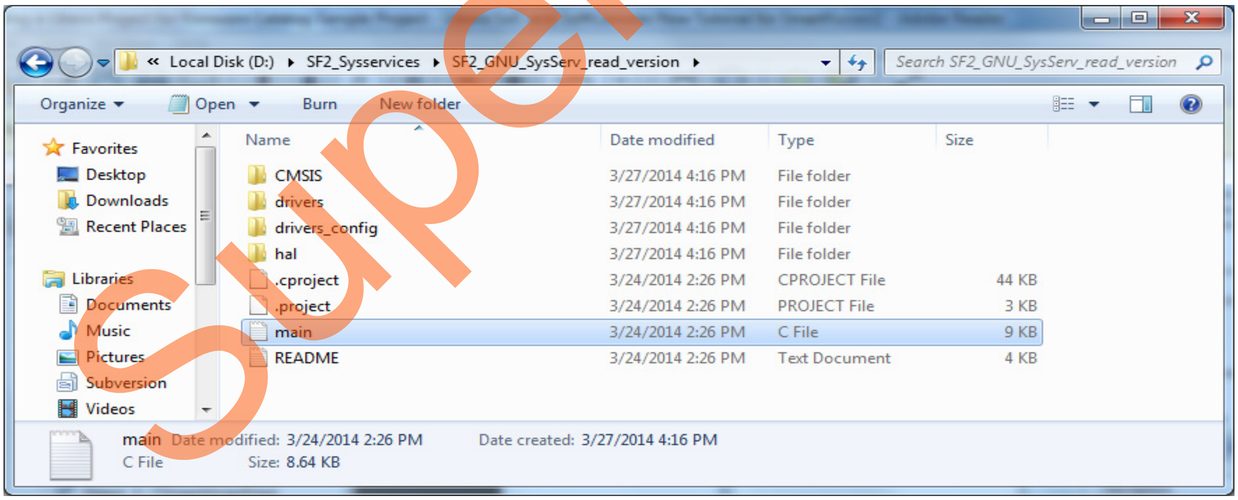
Additional Information
* Release Notes
http://www.actel.com/download/software/softconsole
* Online documentation
http://www.actel.com/products/software/softconsole#docs

0 items

Description	Resource	Path	Location	Type

Figure 27 • SoftConsole Workspace

7. Go to the location where the SoftConsole sample Firmware catalog project is saved, as shown in Figure 28.

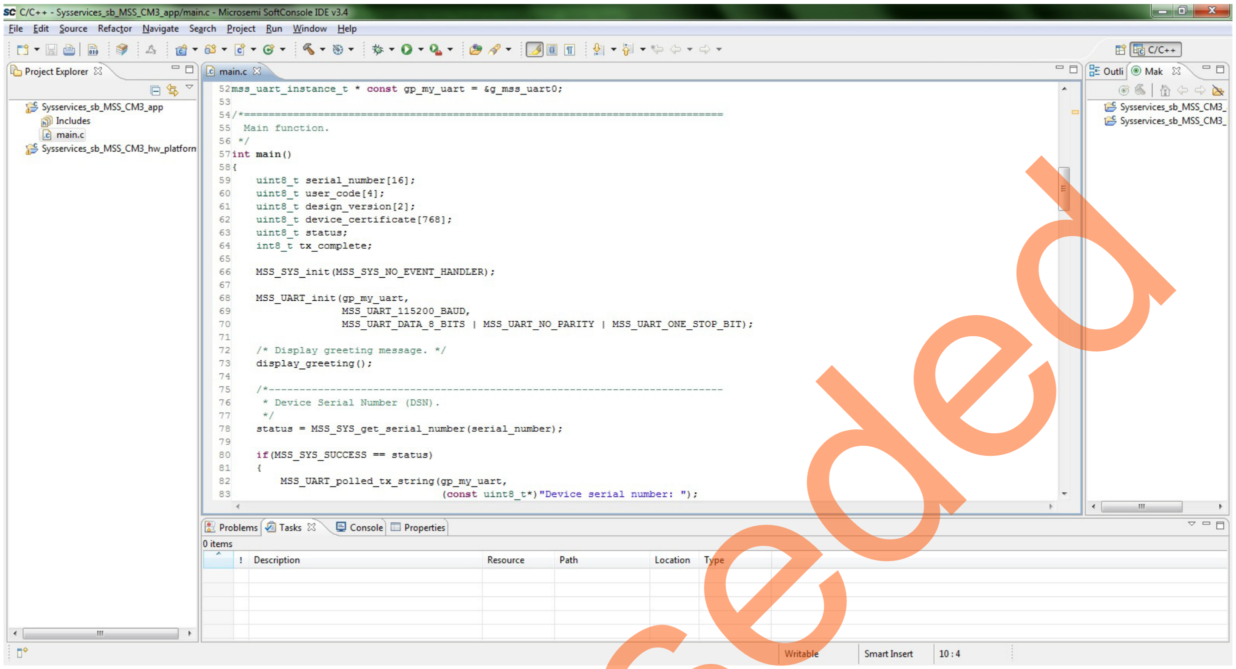


Name	Date modified	Type	Size
CMSIS	3/27/2014 4:16 PM	File folder	
drivers	3/27/2014 4:16 PM	File folder	
drivers_config	3/27/2014 4:16 PM	File folder	
hal	3/27/2014 4:16 PM	File folder	
.cproject	3/24/2014 2:26 PM	CPROJECT File	44 KB
.project	3/24/2014 2:26 PM	PROJECT File	3 KB
main	3/24/2014 2:26 PM	C File	9 KB
README	3/24/2014 2:26 PM	Text Document	4 KB

main Date modified: 3/24/2014 2:26 PM Date created: 3/27/2014 4:16 PM
C File Size: 8.64 KB

Figure 28 • Sample Project main. c File

8. Copy the `main.c file` and replace it with the existing `main.c` file under **Sysservices_sb_MSS_CM3** project in the SoftConsole workspace. The SoftConsole window is shown in Figure 29.


C/C++ - Sysservices_sb_MSS_CM3_app/main.c - Microsemi SoftConsole IDE v3.4
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Sysservices_sb_MSS_CM3_app
Includes
main.c
Sysservices_sb_MSS_CM3_hw_platform
main.c
C/C++
Outli
Mak
Sysservices_sb_MSS_CM3_
Sysservices_sb_MSS_CM3_
Problems
Tasks
Console
Properties
Writable
Smart Insert
10 : 4

```
mss_uart_instance_t * const gp_my_uart = &g_mss_uart0;

/*==============================================================================
  Main function.
 */
int main()
{
    uint8_t serial_number[16];
    uint8_t user_code[4];
    uint8_t design_version[2];
    uint8_t device_certificate[768];
    uint8_t status;
    int8_t tx_complete;

    MSS_SYS_init(MSS_SYS_NO_EVENT_HANDLER);

    MSS_UART_init(gp_my_uart,
                  MSS_UART_115200_BAUD,
                  MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT);

    /* Display greeting message. */
    display_greeting();

    /*--------------------------------------------------------------------------
     * Device Serial Number (DSN).
     */
    status = MSS_SYS_get_serial_number(serial_number);

    if(MSS_SYS_SUCCESS == status)
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"Device serial number: ");
```

0 items

	!	Description	Resource	Path	Location	Type

Figure 29 • **SoftConsole Workspace - main.c File**

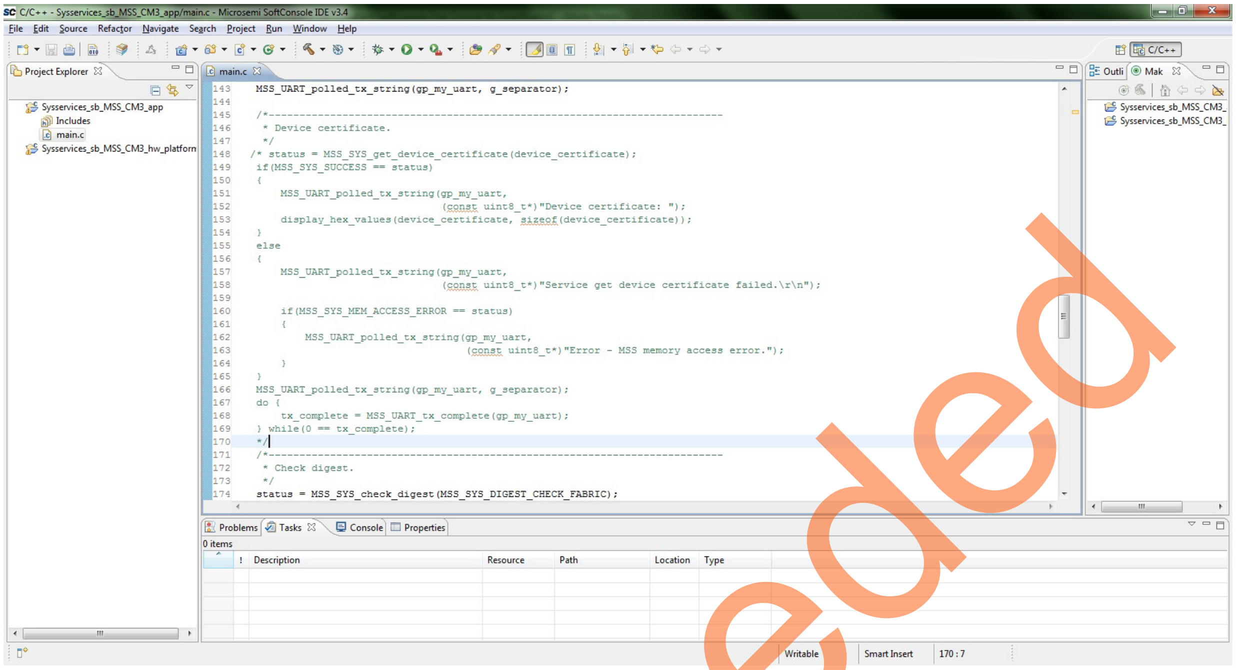
Device certificate service is not demonstrated in this tutorial. It will be available in future releases.

– Comment the lines in main.c file which execute the Device Certificate service

– update gp_my_uart to &g_mss_uart1 in `main.c` file

Modified `main.c` is available in "Appendix 4: main.c File" on page 37.

Figure 30 shows the SoftConsole workspace with modified `main.c` file.



```
    MSS_UART_polled_tx_string(gp_my_uart, g_separator);

    /*------------------------------------------------------------------------
     * Device certificate.
     */
   /* status = MSS_SYS_get_device_certificate(device_certificate);
    if(MSS_SYS_SUCCESS == status)
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"Device certificate: ");
        display_hex_values(device_certificate, sizeof(device_certificate));
    }
    else
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"Service get device certificate failed.\r\n");

        if(MSS_SYS_MEM_ACCESS_ERROR == status)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                      (const uint8_t*)"Error - MSS memory access error.");
        }
    }
    MSS_UART_polled_tx_string(gp_my_uart, g_separator);
    do {
        tx_complete = MSS_UART_tx_complete(gp_my_uart);
    } while(0 == tx_complete);
    */
    /*------------------------------------------------------------------------
     * Check digest.
     */
    status = MSS_SYS_check_digest(MSS_SYS_DIGEST_CHECK_FABRIC);
```

C/C++ - Sysservices_sb_MSS_CM3_app/main.c - Microsemi SoftConsole IDE v3.4
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Sysservices_sb_MSS_CM3_app
Includes
main.c
Sysservices_sb_MSS_CM3_hw_platform
main.c
Outli
Mak
Sysservices_sb_MSS_CM3_
Sysservices_sb_MSS_CM3_
Problems
Tasks
Console
Properties
0 items
Description
Resource
Path
Location
Type
Writable
Smart Insert
170 : 7

Figure 30 • SoftConsole Workspace - Modified main.c File

9. Right-click **Sysservices_sb_MSS_CM3** in the **Project Explorer** window of the SoftConsole project and select **Properties** as shown in Figure 31.

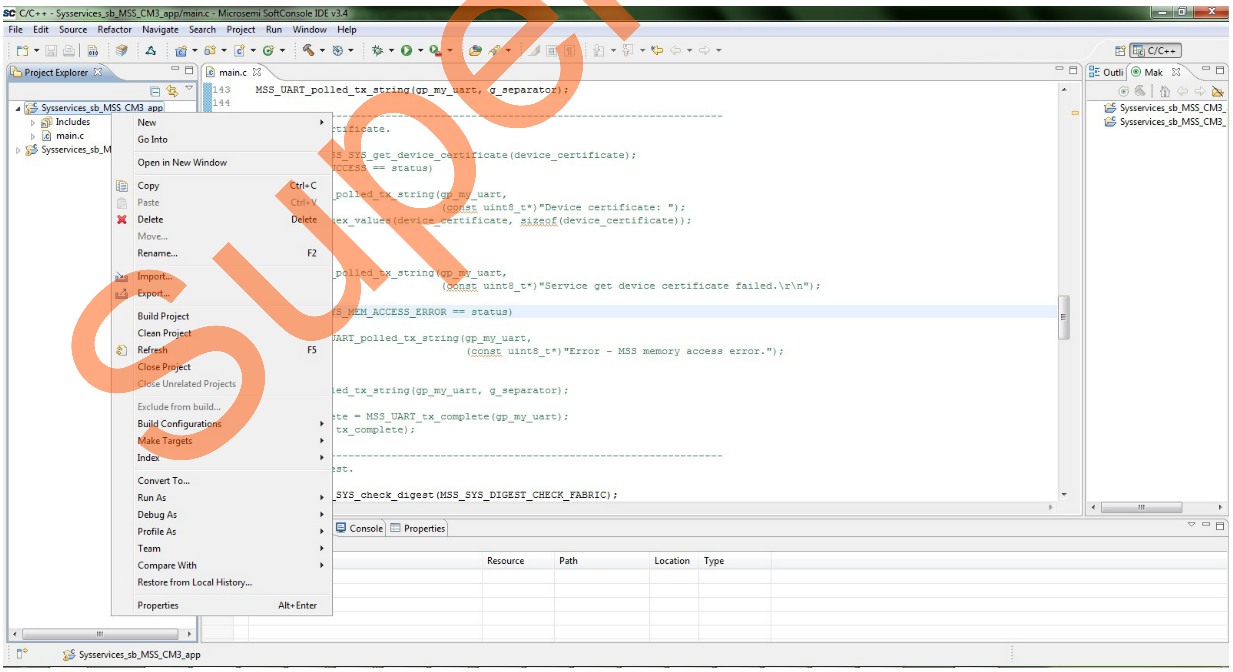
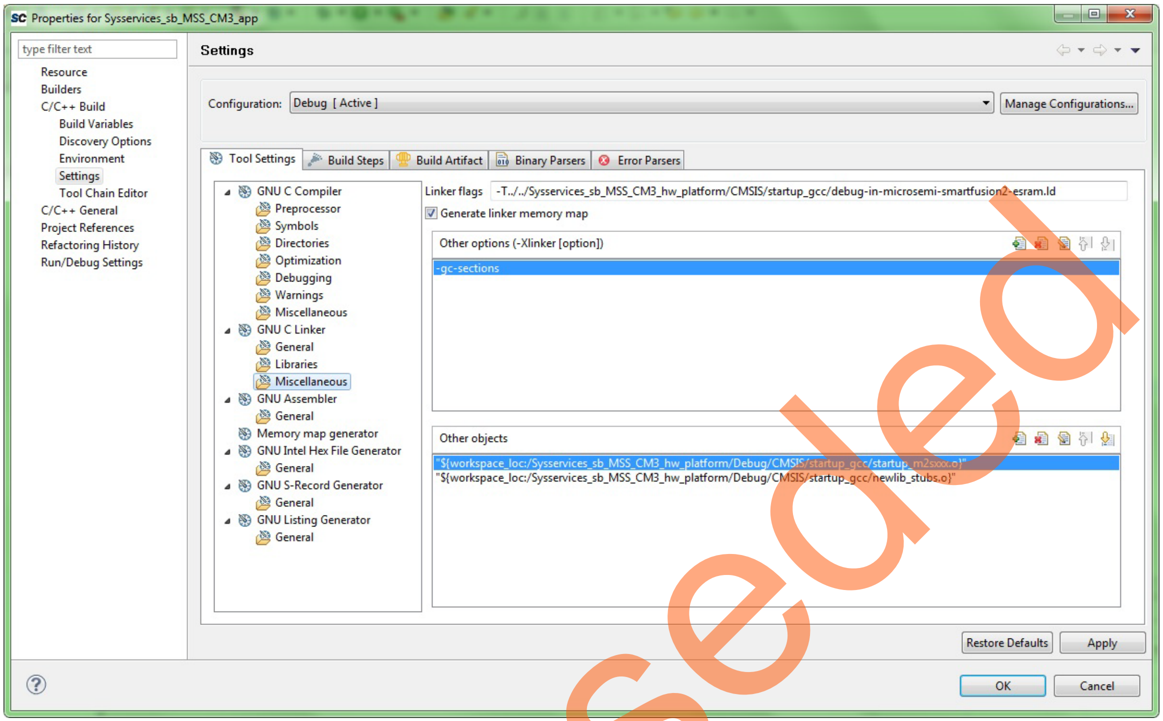


Figure 31 • Project Explorer Window - SoftConsole Project

10. In the **Properties** window, go to **Settings** under **C/C ++ Build** and select **GNU C linker** as **debug-in-microsemi-smartfusion2-esarm.ld** as shown in Figure 32. Click **Apply** and then **OK**.



Configuration: Debug [Active]

Linker flags -T../../Sysservices_sb_MSS_CM3_hw_platform/CMSIS/startup_gcc/debug-in-microsemi-smartfusion2-esram.ld

☑ Generate linker memory map

Other options (-Xlinker [option])

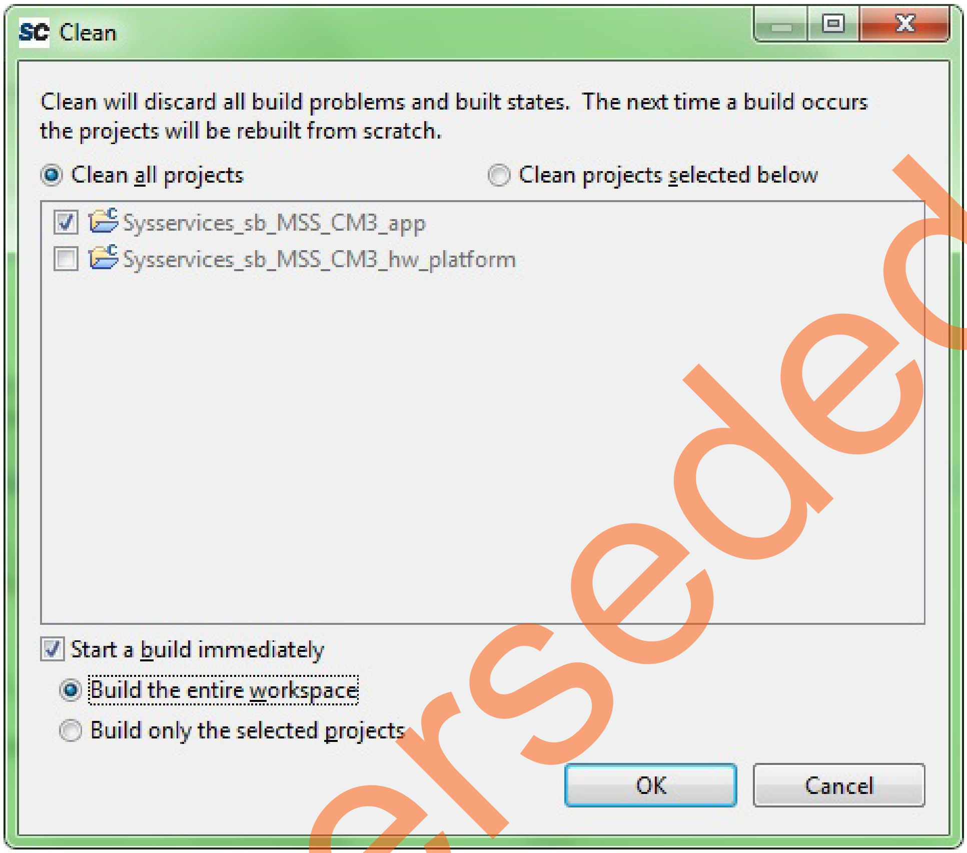
```
-gc-sections
```

Other objects

```
"${workspace_loc:/Sysservices_sb_MSS_CM3_hw_platform/Debug/CMSIS/startup_gcc/startup_m2sxxx.o}"
"${workspace_loc:/Sysservices_sb_MSS_CM3_hw_platform/Debug/CMSIS/startup_gcc/newlib_stubs.o}"
```

Figure 32 • **Sysservices_sb_MSS_CM3 Properties Window**

11. Perform a clean build by selecting **Project > Clean**. Accept the default settings in the **Clean** dialog box and click **OK,** as shown in Figure 33. The SoftConsole project must not have any errors.



SC Clean

Clean will discard all build problems and built states. The next time a build occurs the projects will be rebuilt from scratch.

● Clean all projects ○ Clean projects selected below

☑ Sysservices_sb_MSS_CM3_app
☐ Sysservices_sb_MSS_CM3_hw_platform

☑ Start a build immediately

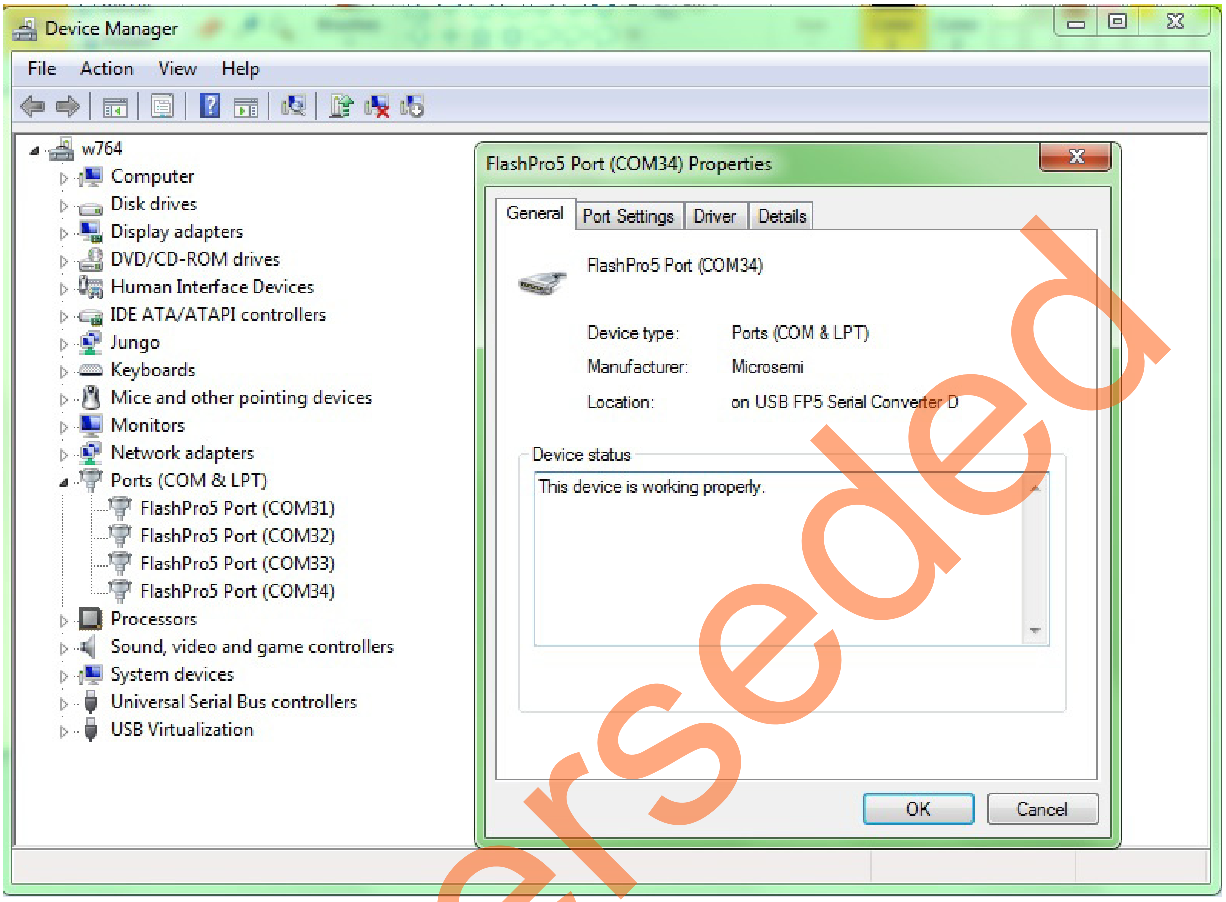
● Build the entire workspace

○ Build only the selected projects

Figure 33 • **Settings for Clean Build**

12. Install the USB driver. For serial terminal communication through the FTDI mini-USB cable, install the FTDI D2XX driver. Download the drivers and the installation guide from *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip*

13. Connect the host PC to the J18 connector using the USB min-B cable. The USB to UART bridge drivers are automatically detected. Verify if the detection is made in the device manager, as shown in Figure 34 on page 30.



FlashPro5 Port (COM34)

Device type: Ports (COM & LPT)

Manufacturer: Microsemi

Location: on USB FP5 Serial Converter D

Device status

This device is working properly.

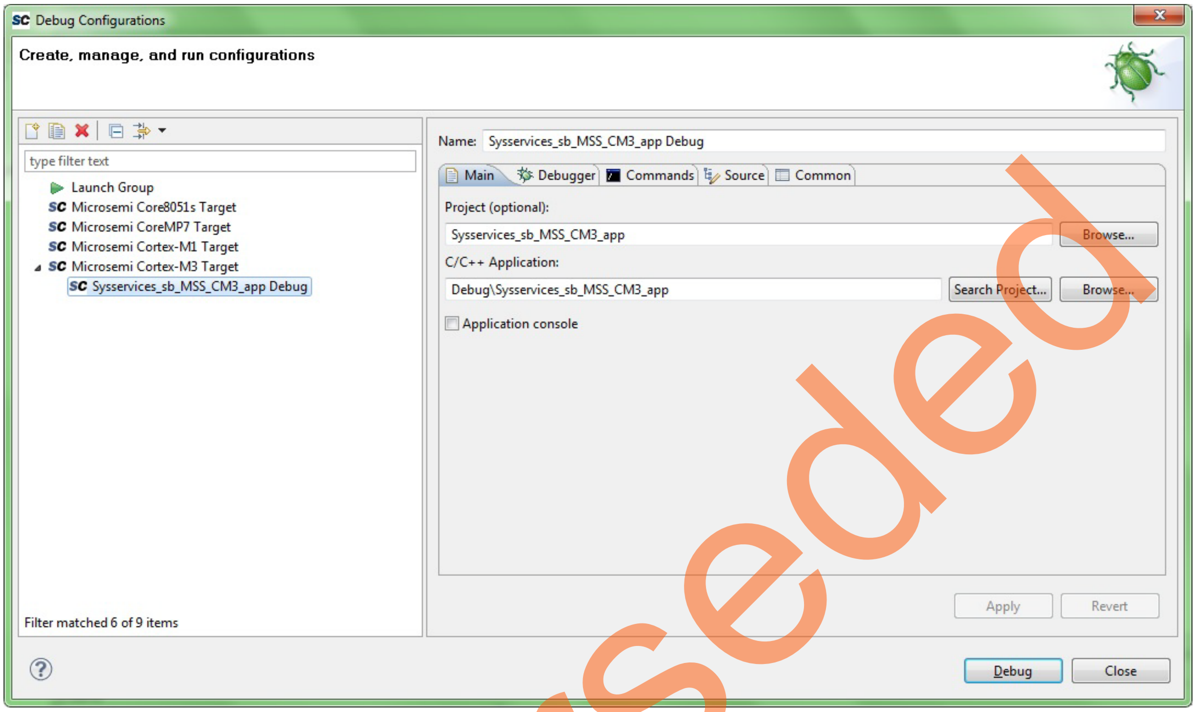
Figure 34 • **Device Manager Window**

14. Start the PuTTY session. If the PuTTY program is not available in the computer system, use any free serial terminal emulation program such as HyperTerminal or TeraTerm. Refer to the *Configuring Serial Terminal Emulation Programs Tutorial* for configuring the HyperTerminal, TeraTerm, or PuTTY.

    The PuTTY settings are as follows:

    – 115,200 baud rate
    – 8 data bits
    – 1 stop bit
    – No parity
    – No flow control

15. Select **Debug Configurations** from the **Run** menu of the SoftConsole. The **Debug** dialog box is displayed. Double-click on **Microsemi Cortex-M3 Target.**This displays a window similar to Figure 35.



Create, manage, and run configurations

- Launch Group
- SC Microsemi Core8051s Target
- SC Microsemi CoreMP7 Target
- SC Microsemi Cortex-M1 Target
- SC Microsemi Cortex-M3 Target
  - SC Sysservices_sb_MSS_CM3_app Debug

Filter matched 6 of 9 items

Name: Sysservices_sb_MSS_CM3_app Debug

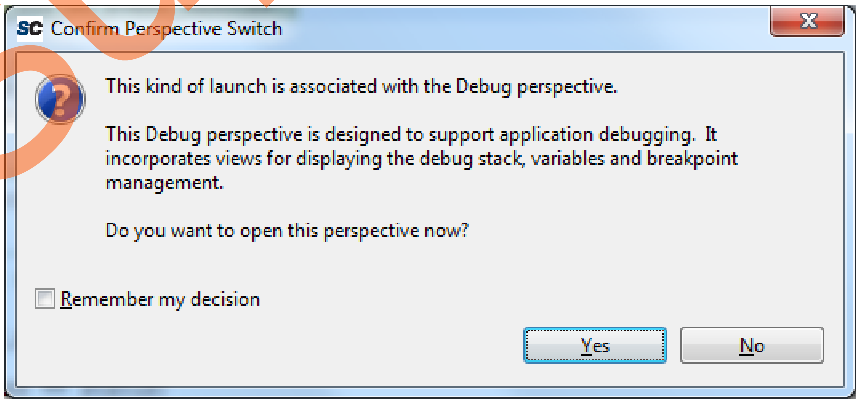
Project (optional): Sysservices_sb_MSS_CM3_app

C/C++ Application: Debug\Sysservices_sb_MSS_CM3_app

□ Application console

Figure 35 • **Debug Configurations Window**

16. Ensure that the following information appears on the **Main** tab in the **Debug Configurations** window, and click **Debug**:
    - **Name**: Sysservices_sb_MSS_ CM3 Debug
    - **Project**: Sysservices_sb_MSS_ CM3
    - **C/C++ Application**: Debug\Sysservices_MSS_MSS_CM3_app
17. Click **Yes** when prompted for the **Confirm Perspective Switch**, as shown in Figure 36. This displays the debug view mode.



This kind of launch is associated with the Debug perspective.

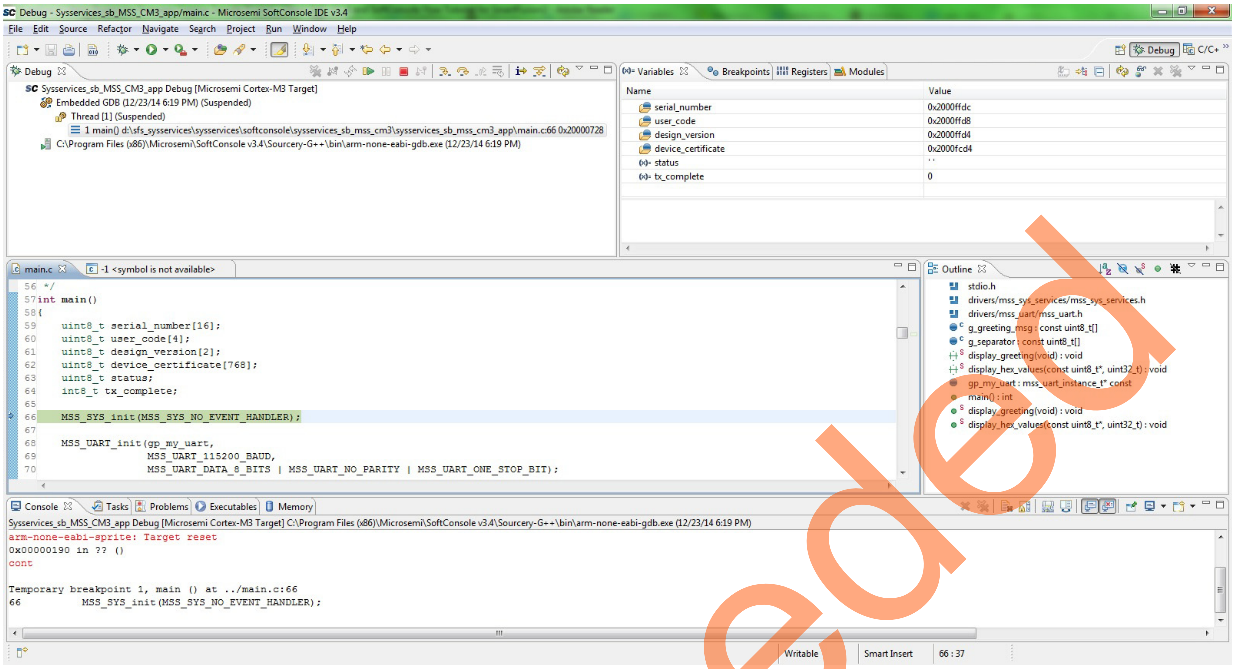
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

□ Remember my decision

Figure 36 • **Confirm Perspective Switch**

The **SoftConsole Debugger Perspective** window is opened, as shown in Figure 37.


SC Debug - Sysservices_sb_MSS_CM3_app/main.c - Microsemi SoftConsole IDE v3.4
File Edit Source Refactor Navigate Search Project Run Window Help
Debug C/C+
Debug
(x)= Variables
Breakpoints
Registers
Modules
main.c
-1 <symbol is not available>
Outline
Console
Tasks
Problems
Executables
Memory
Writable
Smart Insert
66 : 37

Sysservices_sb_MSS_CM3_app Debug [Microsemi Cortex-M3 Target]
- Embedded GDB (12/23/14 6:19 PM) (Suspended)
  - Thread [1] (Suspended)
    - 1 main() d:\sfs_sysservices\sysservices\softconsole\sysservices_sb_mss_cm3\sysservices_sb_mss_cm3_app\main.c:66 0x20000728
- C:\Program Files (x86)\Microsemi\SoftConsole v3.4\Sourcery-G++\bin\arm-none-eabi-gdb.exe (12/23/14 6:19 PM)

Name	Value
serial_number	0x2000ffdc
user_code	0x2000ffd8
design_version	0x2000ffd4
device_certificate	0x2000fcd4
status	' '
tx_complete	0

```
 */
int main()
{
    uint8_t serial_number[16];
    uint8_t user_code[4];
    uint8_t design_version[2];
    uint8_t device_certificate[768];
    uint8_t status;
    int8_t tx_complete;

    MSS_SYS_init(MSS_SYS_NO_EVENT_HANDLER);

    MSS_UART_init(gp_my_uart,
                  MSS_UART_115200_BAUD,
                  MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT);
```

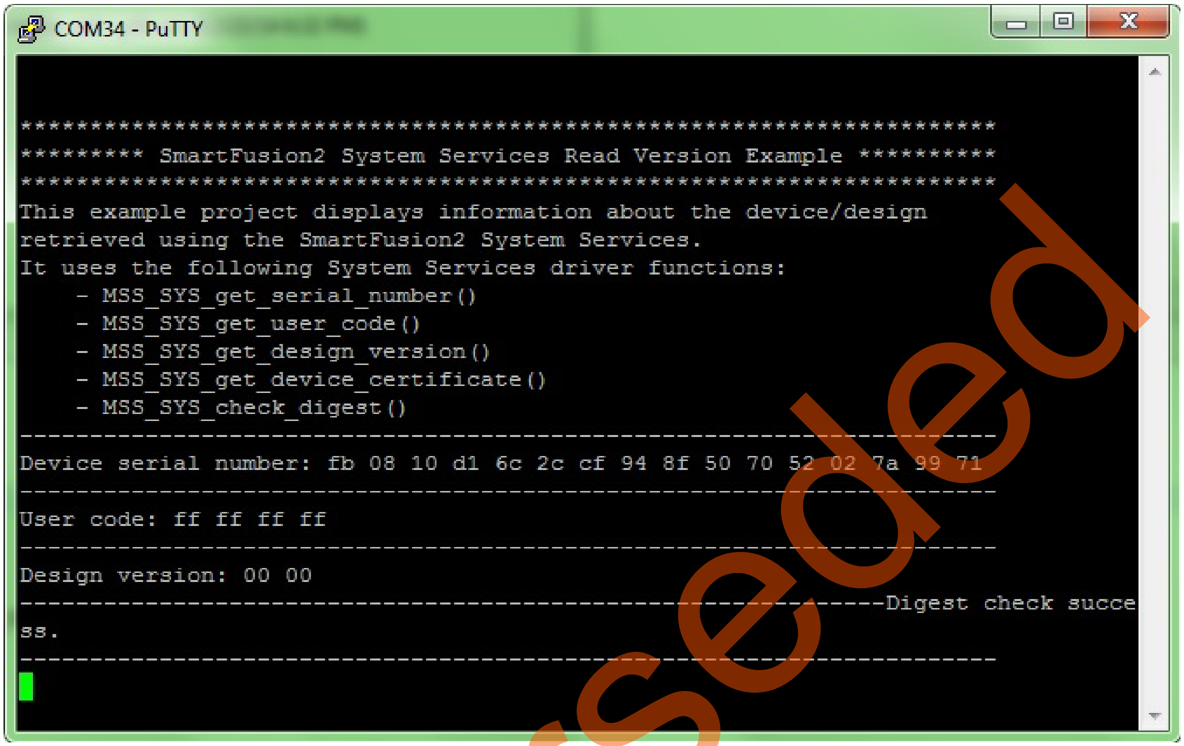
- stdio.h
- drivers/mss_sys_services/mss_sys_services.h
- drivers/mss_uart/mss_uart.h
- g_greeting_msg : const uint8_t[]
- g_separator : const uint8_t[]
- display_greeting(void) : void
- display_hex_values(const uint8_t*, uint32_t) : void
- gp_my_uart : mss_uart_instance_t* const
- main() : int
- display_greeting(void) : void
- display_hex_values(const uint8_t*, uint32_t) : void

Sysservices_sb_MSS_CM3_app Debug [Microsemi Cortex-M3 Target] C:\Program Files (x86)\Microsemi\SoftConsole v3.4\Sourcery-G++\bin\arm-none-eabi-gdb.exe (12/23/14 6:19 PM)

```
arm-none-eabi-sprite: Target reset
0x00000190 in ?? ()
cont

Temporary breakpoint 1, main () at ../main.c:66
66          MSS_SYS_init(MSS_SYS_NO_EVENT_HANDLER);
```

Figure 37 • **SoftConsole Debugger Perspective**

18. Run the application by clicking **Run > Resume**. The Information on the SmartFusion2 device and design, along with a greeting message is displayed on the PuTTY, as shown in Figure 38.



```
**********************************************************************
********* SmartFusion2 System Services Read Version Example **********
**********************************************************************
This example project displays information about the device/design
retrieved using the SmartFusion2 System Services.
It uses the following System Services driver functions:
    - MSS_SYS_get_serial_number()
    - MSS_SYS_get_user_code()
    - MSS_SYS_get_design_version()
    - MSS_SYS_get_device_certificate()
    - MSS_SYS_check_digest()
----------------------------------------------------------------------
Device serial number: fb 08 10 d1 6c 2c cf 94 8f 50 70 52 02 7a 99 71
----------------------------------------------------------------------
User code: ff ff ff ff
----------------------------------------------------------------------
Design version: 00 00
---------------------------------------------------------------Digest check succe
ss.
----------------------------------------------------------------------
```

Figure 38 • **PuTTY Window**

19. Terminate execution of the code by choosing **Run > Terminate**.
20. Close **Debug Perspective** by selecting **Close Perspective** from the **Window** menu.
21. Close SoftConsole using **File > Exit.**
22. Close the PuTTY. Click **Yes** when prompted for closing.

Conclusion

This tutorial describes how to download the SoftConsole Sample project from the Firmware catalog and how to create a Libero SoC project. It explains the procedure to generate the programming file and to run the SoftConsole project on the SmartFusion2 Security Evaluation Kit. A sample project for implementing System Services features is created to display the SmartFusion2 device and design information.

Appendix 1: Jumper Locations

Figure 39 shows the jumper locations in the SmartFusion2 Security Evaluation Kit Board.

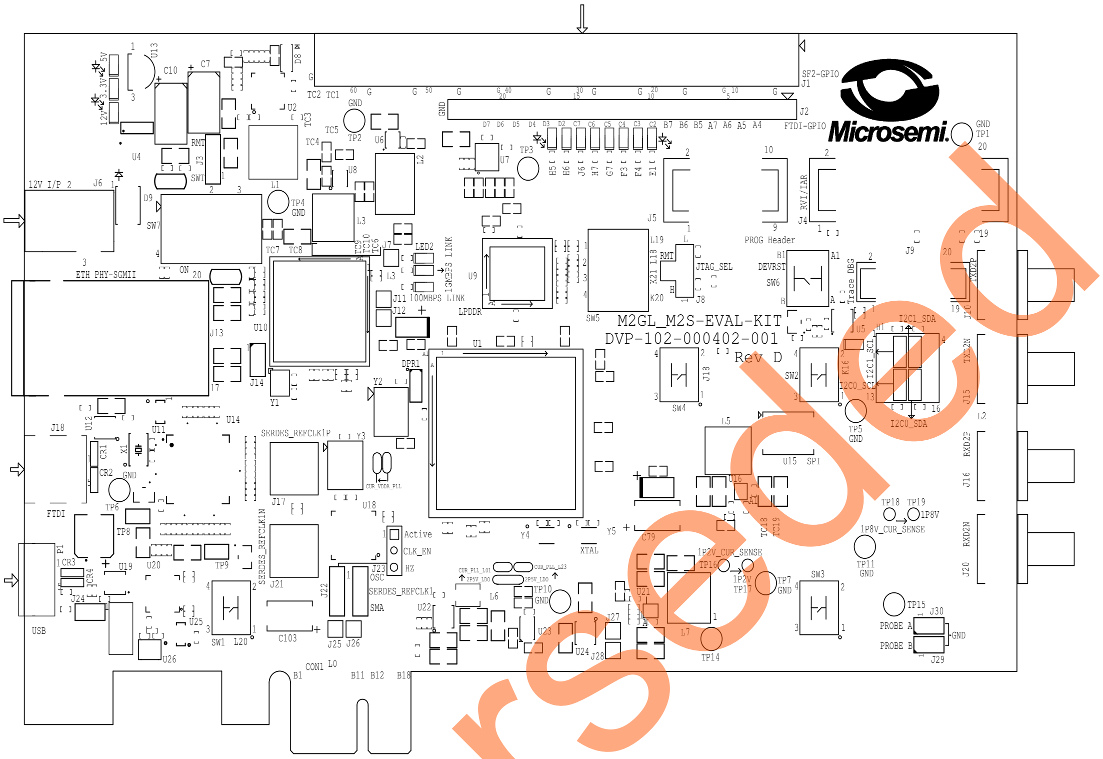


Figure 39 • Jumper Locations

Note: The location of the jumpers in Figure 39 are searchable.

Appendix 2: Board Setup for Running the Tutorial

Figure 40 shows the board setup for running the tutorial on the SmartFusion2 Security Evaluation Kit Board.

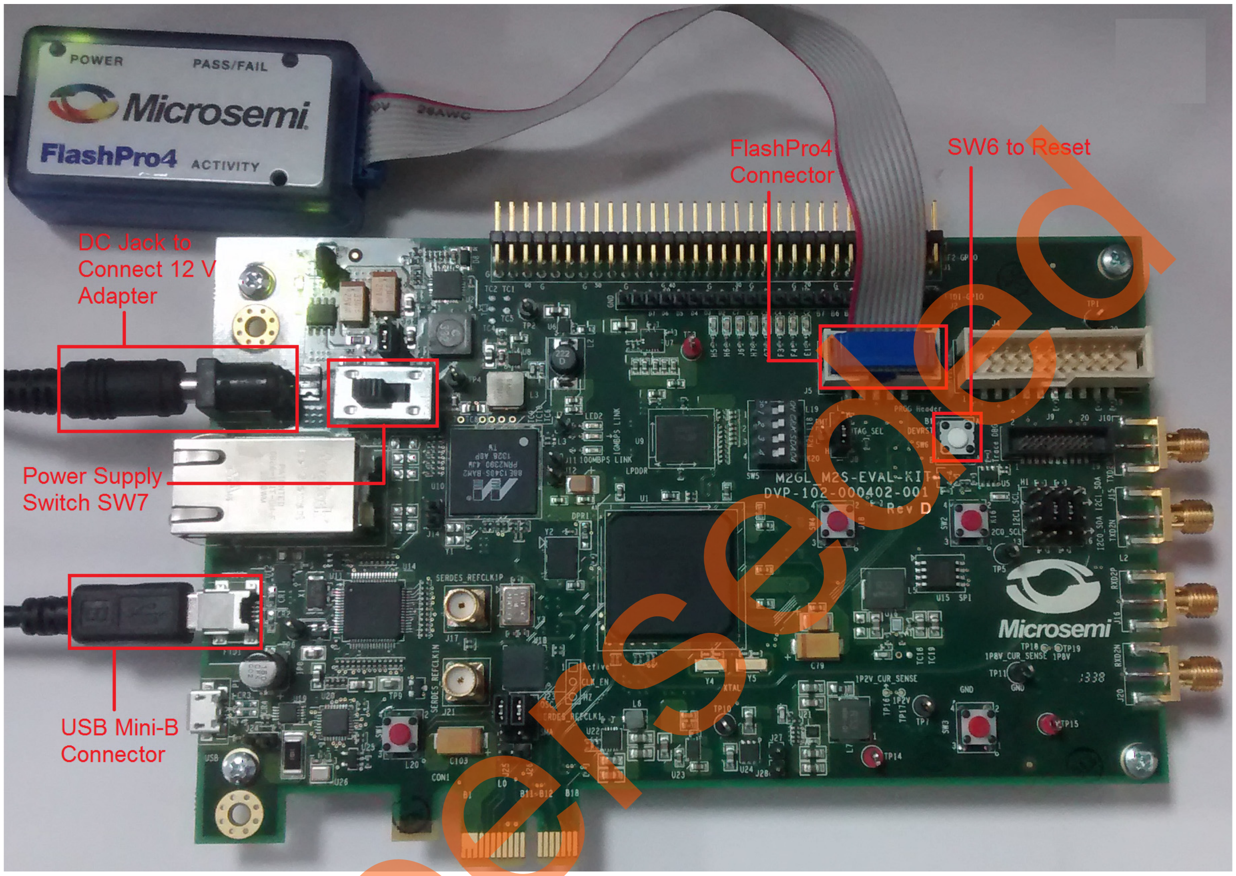


Figure 40 • **SmartFusion2 Security Evaluation Kit**

Appendix 3: Readme File

```
================================================================================
                SmartFusion2 System Services Read Version example
================================================================================

This example project demonstrates the use of the SmartFusion2 System Services
functions:
    - MSS_SYS_get_serial_number()
    - MSS_SYS_get_user_code()
    - MSS_SYS_get_design_version()
    - MSS_SYS_get_device_certificate()
    - MSS_SYS_check_digest()

--------------------------------------------------------------------------------
                            How to use this example
--------------------------------------------------------------------------------
This example project requires MMUART0 to be connected to a host PC. The host PC
must connect to the serial port using a terminal emulator such as HyperTerminal
or PuTTY configured as follows:
    - 115200 baud
    - 8 data bits
    - 1 stop bit
    - no parity
    - no flow control

The example project will display the following information about the
SmartFusion2 device on which it is executed:
    - device serial number
    - user code
    - design version
    - device certificate
    - digest status

--------------------------------------------------------------------------------
                                Target hardware
--------------------------------------------------------------------------------
This example project is targeted at a SmartFusion2 design which has MMUART0
enabled and connected to a host PC. The example project is built for a design
using a SmartFusion2 MSS APB clock frequency of 83MHz. Trying to execute this
example project on a different design will result in incorrect baud rate being
used by MMUART0 or no output if MMUART0 is not enabled and connected.

This example project can be used with another design using a different clock
configuration. This can be achieved by overwriting the content of this example
project's "drivers_config/sys_config" folder with the one generated by Libero
as part of your design's creation.

--------------------------------------------------------------------------------
     Redirecting MMUART0 to RS232 connector on SmartFusion2 Development Kit
--------------------------------------------------------------------------------
Please note that it is possible to redirect MMUART0 to the J198 RS232 connector
on the SmartFusion2 Development Kit despite J198 being connected to the MMUART1
SmartFusion2 pads. This can be done in your hardware design by selecting to
direct the MMUART0 TXD and RXD signals to the FPGA fabric and then connecting
these signals to top level ports assigned to pin H30 for TXD and pin G29 for
RXD.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                          Silicon revision dependencies
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This example is built to execute on an M2S050T die (M2S050T revision D). You
will need to overwrite this example project's "drivers_config/sys_config" and
"CMSIS" folders with the one generated by Libero for your hardware design if
```

```
using a newer silicon revision.
The "drivers_config/sys_config" folder contains information about your hardware
design. This information is used by the CMSIS to initialize clock frequencies
global variables which are used by the SmartFusion2 drivers to derive baud
rates. The CMSIS boot code may also complete the device's clock configuration
depending on silicon version. The "CMSIS" and "drivers_config/sys_config" for
your design can be found in the "firmware" folder of your Libero design.
```

Appendix 4: main.c File

```
/*******************************************************************************
 * (c) Copyright 2012-2014 Microsemi SoC Products Group.  All rights reserved.
 *
 * Retrieve device and design information using System Services.
 *
 * Please refer to file README.TXT for further details about this example.
 *
 * SVN $Revision: 6886 $
 * SVN $Date: 2014-09-03 14:22:17 +0530 (Wed, 03 Sep 2014) $
 */
#include <stdio.h>
#include "drivers/mss_sys_services/mss_sys_services.h"
#include "drivers/mss_uart/mss_uart.h"

/*==============================================================================
  Messages displayed over the UART.
 */
const uint8_t g_greeting_msg[] =
"\r\n\r\n\
**********************************************************************\r\n\
********* SmartFusion2 System Services Read Version Example **********\r\n\
**********************************************************************\r\n\
This example project displays information about the device/design\r\n\
retrieved using the SmartFusion2 System Services.\r\n\
It uses the following System Services driver functions:\r\n\
    - MSS_SYS_get_serial_number()\r\n\
    - MSS_SYS_get_user_code()\r\n\
    - MSS_SYS_get_design_version()\r\n\
    - MSS_SYS_get_device_certificate()\r\n\
    - MSS_SYS_check_digest()\r\n\
----------------------------------------------------------------------\r\n";

const uint8_t g_separator[] =
"\r\n----------------------------------------------------------------------\r\n";

/*==============================================================================
  Private functions.
 */
static void display_greeting(void);
static void display_hex_values
(
    const uint8_t * in_buffer,
    uint32_t byte_length
);

/*==============================================================================
  UART selection.
  Replace the line below with this one if you want to use UART1 instead of
  UART0:
  mss_uart_instance_t * const gp_my_uart = &g_mss_uart1;
 */
mss_uart_instance_t * const gp_my_uart = &g_mss_uart1;
```

```
/*==============================================================================
  Main function.
 */
int main()
{
    uint8_t serial_number[16];
    uint8_t user_code[4];
    uint8_t design_version[2];
    uint8_t device_certificate[768];
    uint8_t status;
    int8_t tx_complete;

    MSS_SYS_init(MSS_SYS_NO_EVENT_HANDLER);

    MSS_UART_init(gp_my_uart,
                  MSS_UART_115200_BAUD,
                  MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT);

    /* Display greeting message. */
    display_greeting();

    /*--------------------------------------------------------------------------
     * Device Serial Number (DSN).
     */
    status = MSS_SYS_get_serial_number(serial_number);

    if(MSS_SYS_SUCCESS == status)
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"Device serial number: ");
        display_hex_values(serial_number, sizeof(serial_number));
    }
    else
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"Service read device serial number
failed.\r\n");

        if(MSS_SYS_MEM_ACCESS_ERROR == status)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                       (const uint8_t*)"Error - MSS memory access error.");
        }
    }
    MSS_UART_polled_tx_string(gp_my_uart, g_separator);

    /*--------------------------------------------------------------------------
     * User code.
     */
    status = MSS_SYS_get_user_code(user_code);
    if(MSS_SYS_SUCCESS == status)
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"User code: ");
        display_hex_values(user_code, sizeof(user_code));
    }
    else
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                 (const uint8_t*)"Service read user code failed.\r\n");

        if(MSS_SYS_MEM_ACCESS_ERROR == status)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                   (const uint8_t*)"Error - MSS memory access error.");
        }
    }
    MSS_UART_polled_tx_string(gp_my_uart, g_separator);

    /*----------------------------------------------------------------------------
     * Design version.
     */
    status = MSS_SYS_get_design_version(design_version);
    if(MSS_SYS_SUCCESS == status)
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"Design version: ");
        display_hex_values(design_version, sizeof(design_version));
    }
    else
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                             (const uint8_t*)"Service get design version failed.\r\n");

        if(MSS_SYS_MEM_ACCESS_ERROR == status)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                   (const uint8_t*)"Error - MSS memory access error.");
        }
    }
    MSS_UART_polled_tx_string(gp_my_uart, g_separator);

    /*----------------------------------------------------------------------------
     * Device certificate.
     */
   /* status = MSS_SYS_get_device_certificate(device_certificate);
    if(MSS_SYS_SUCCESS == status)
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"Device certificate: ");
        display_hex_values(device_certificate, sizeof(device_certificate));
    }
    else
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                         (const uint8_t*)"Service get device certificate failed.\r\n");

        if(MSS_SYS_MEM_ACCESS_ERROR == status)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                   (const uint8_t*)"Error - MSS memory access error.");
        }
    }
    MSS_UART_polled_tx_string(gp_my_uart, g_separator);
    do {
        tx_complete = MSS_UART_tx_complete(gp_my_uart);
    } while(0 == tx_complete);
    */
    /*----------------------------------------------------------------------------
     * Check digest.
     */
    status = MSS_SYS_check_digest(MSS_SYS_DIGEST_CHECK_FABRIC);
    if(MSS_SYS_SUCCESS == status)
    {
        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"Digest check success.");
    }
    else
```

```
    {
        uint8_t fabric_digest_check_failure;
        uint8_t envm0_digest_check_failure;
        uint8_t envm1_digest_check_failure;
        uint8_t sys_digest_check_failure;
        uint8_t envmfp_digest_check_failure;
        uint8_t envmup_digest_check_failure;
        uint8_t svcdisabled_digest_check_failure;

        fabric_digest_check_failure = status & MSS_SYS_DIGEST_CHECK_FABRIC;
        envm0_digest_check_failure = status & MSS_SYS_DIGEST_CHECK_ENVM0;
        envm1_digest_check_failure = status & MSS_SYS_DIGEST_CHECK_ENVM1;
        sys_digest_check_failure = status & MSS_SYS_DIGEST_CHECK_SYS;
        envmfp_digest_check_failure = status & MSS_SYS_DIGEST_CHECK_ENVMFP;
        envmup_digest_check_failure = status & MSS_SYS_DIGEST_CHECK_ENVMUP;
        svcdisabled_digest_check_failure = status & MSS_SYS_DIGEST_CHECK_SVCDISABLED;

        MSS_UART_polled_tx_string(gp_my_uart,
                                  (const uint8_t*)"\r\nDigest check failure:");
        if(fabric_digest_check_failure)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                    (const uint8_t*)"\r\nFabric digest check failed.");
        }
        if(envm0_digest_check_failure)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                     (const uint8_t*)"\r\neNVM0 digest check failed.");
        }
        if(envm1_digest_check_failure)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                     (const uint8_t*)"\r\neNVM1 digest check failed.");
        }
        if(sys_digest_check_failure)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                    (const uint8_t*)"\r\n System Controller ROM digest
check failed.");
        }
        if(envmfp_digest_check_failure)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                     (const uint8_t*)"\r\n Private eNVM factory digest
check failed.");
        }
        if(envmup_digest_check_failure)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                      (const uint8_t*)"\r\n Private eNVM user digest
check failed.");
        }
        if(svcdisabled_digest_check_failure)
        {
            MSS_UART_polled_tx_string(gp_my_uart,
                                   (const uint8_t*)"\r\n Digest check service disabled
by the user lock.");
        }
    }
    MSS_UART_polled_tx_string(gp_my_uart, g_separator);

    for(;;)
    {
        ;
    }
```

```
}

/*==============================================================================
  Display greeting message when application is started.
 */
static void display_greeting(void)
{
    MSS_UART_polled_tx_string(gp_my_uart, g_greeting_msg);
}

/*==============================================================================
  Display content of buffer passed as parameter as hex values
 */
static void display_hex_values
(
    const uint8_t * in_buffer,
    uint32_t byte_length
)
{
    uint8_t display_buffer[128];
    uint32_t inc;

    if(byte_length > 16u)
    {
        MSS_UART_polled_tx_string( gp_my_uart,(const uint8_t*)"\r\n" );
    }

    for(inc = 0; inc < byte_length; ++inc)
    {
        if((inc > 1u) &&(0u == (inc % 16u)))
        {
            MSS_UART_polled_tx_string( gp_my_uart,(const uint8_t*)"\r\n" );
        }
        snprintf((char *)display_buffer, sizeof(display_buffer), "%02x ",
in_buffer[inc]);
        MSS_UART_polled_tx_string(gp_my_uart, display_buffer);
    }
}
```

A – List of Changes

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

Date	Changes	Page
Revision 3 (February 2015)	Updated the document for Libero v11.5 software release (SAR 64799).	NA
Revision 2 (October 2014)	Updated the document for Libero v11.4 software release (SAR 61636).	NA
Revision 1 (April 2014)	Initial release.	NA

B – 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, 408.643.6913

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

For Microsemi SoC Products Support, visit

http://www.microsemi.com/products/fpga-soc/designsupport/fpga-soc-support

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

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. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

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_itar@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 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

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; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,400 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.