
Creating a Libero Project for Firmware Catalog Sample Project

***Libero SoC and SoftConsole Flow Tutorial for
SmartFusion2***

Superseded



Table of Contents

Creating a Libero Project for Firmware Catalog Sample Project - Libero SoC and SoftConsole Flow Tutorial for SmartFusion2	3
Introduction	3
Tutorial Requirements	4
Associated Project Files	4
Target Board	4
Design Overview	4
Step 1: Downloading SoftConsole Project from Firmware Catalog	5
Step 2: Creating a Libero SoC Project	6
Launching Libero SoC	6
Instantiating Clock Conditioning Circuitry (CCC) and Chip Oscillators in SysServices SmartDesign	14
Connecting Components in SysServices SmartDesign	16
Configuring and Generating Firmware	19
Step 3: Generating the Program File	21
Step 4: Programming the SmartFusion2 Board Using FlashPro	23
Step 5: Building the Software Application using SoftConsole	24
Conclusion	34
Appendix 1: Jumper Locations	35
Appendix 2: Board Setup for Running the Tutorial	36
Appendix 3: Readme File	37
Appendix 4: main.c File	38
Product Support	43
Customer Service	43
Customer Technical Support Center	43
Technical Support	43
Website	43
Contacting the Customer Technical Support Center	43
Email	43
My Cases	44
Outside the U.S.	44
ITAR Technical Support	44

Creating a Libero Project for Firmware Catalog Sample Project - Libero SoC 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. But a 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 downloading the SoftConsole sample project from Firmware catalog and creating a Libero SoC hardware design for the downloaded sample project. This tutorial provides an example design for System Services.

Upon completion of this tutorial, you will be familiar with 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

Tutorial Requirements

Table 1 • Reference Design Requirements and Details

Reference Design Requirements and Details	Description
Hardware Requirements	
SmartFusion2 Development Kit <ul style="list-style-type: none">FlashPro4 programmerUSB A to Mini-B cable12 V Adapter	Rev C or later
Host PC or Laptop	Any Windows 64-Bit Operating System
Software Requirements	
Libero SoC <ul style="list-style-type: none">SoftConsole v3.4FlashPro programming software v11.3	11.3
Host PC Drivers	USB to UART drivers
Any one of the following serial terminal emulation programs: <ul style="list-style-type: none">HyperTerminalTeraTermPuTTY	-

Associated Project Files

The design files for this tutorial can be downloaded from the Microsemi® website:
http://soc.microsemi.com/download/rsc/?f=SF2_SYS_SERVICES_Tutorial_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 Development Kit Board (SF2_DEV_KIT) Rev C 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](#).

Step 1: Downloading SoftConsole Project from Firmware Catalog

1. Click **Start > Programs > Microsemi SoC Libero SoC 11.3 > Firmware Catalog v11.3 > Firmware Catalog**. Right-click on **SmartFusion2 MSS System Services Driver** and select **Generate Sample Project > Cortex-M3 > SoftConsole > Read Version Information** as shown in Figure 1.

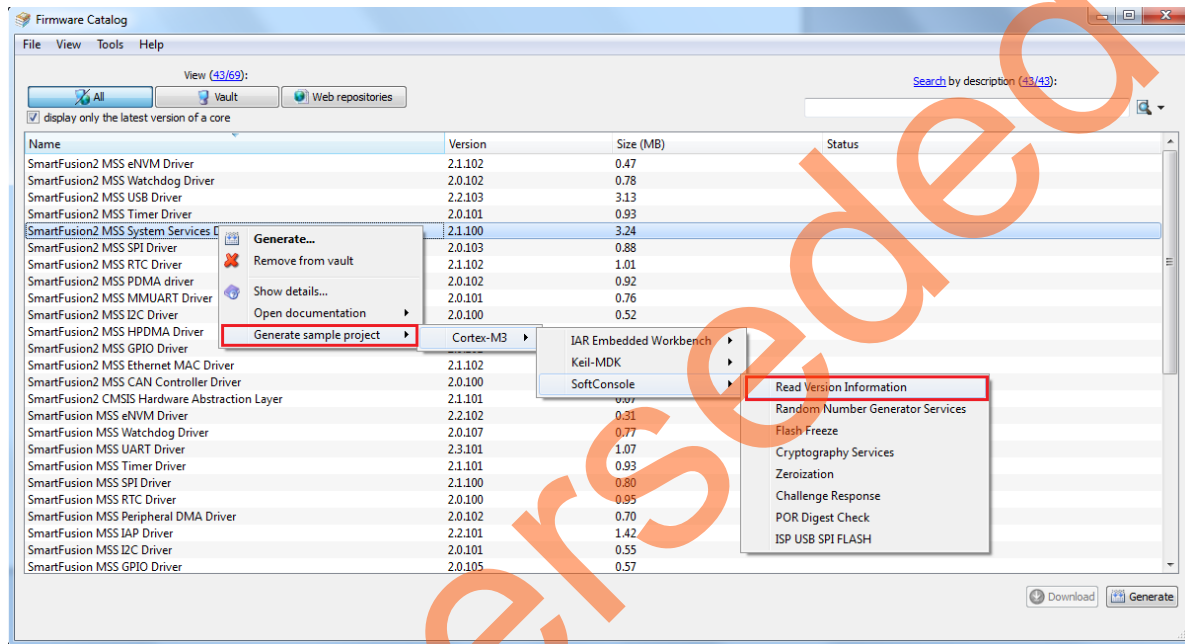


Figure 1 • Downloading Sample Project from Firmware Catalog

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

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

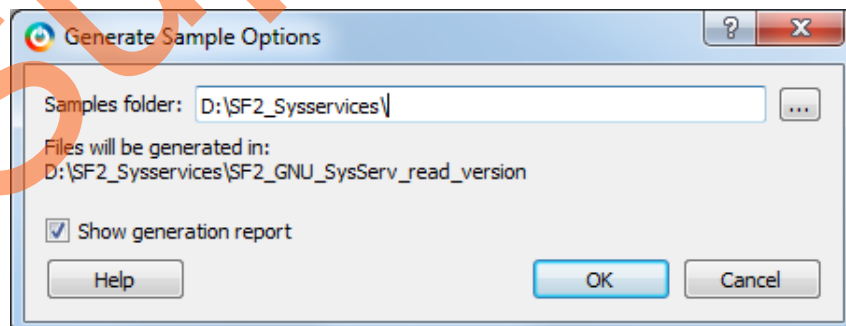


Figure 2 • Generate Sample Options Window

2. Browse to a location to save System Services **Read Version Information** SoftConsole Project.

3. 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 37 for the Readme file.

Step 2: Creating a Libero SoC Project

This step helps you create a SmartFusion2 System Services design using the Libero tool.

Launching Libero SoC

1. Click **Start > Programs > Microsemi Libero SoC v11.3 > Libero SoC v11.3**, or click the shortcut on desktop to open the Libero 11.3 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.

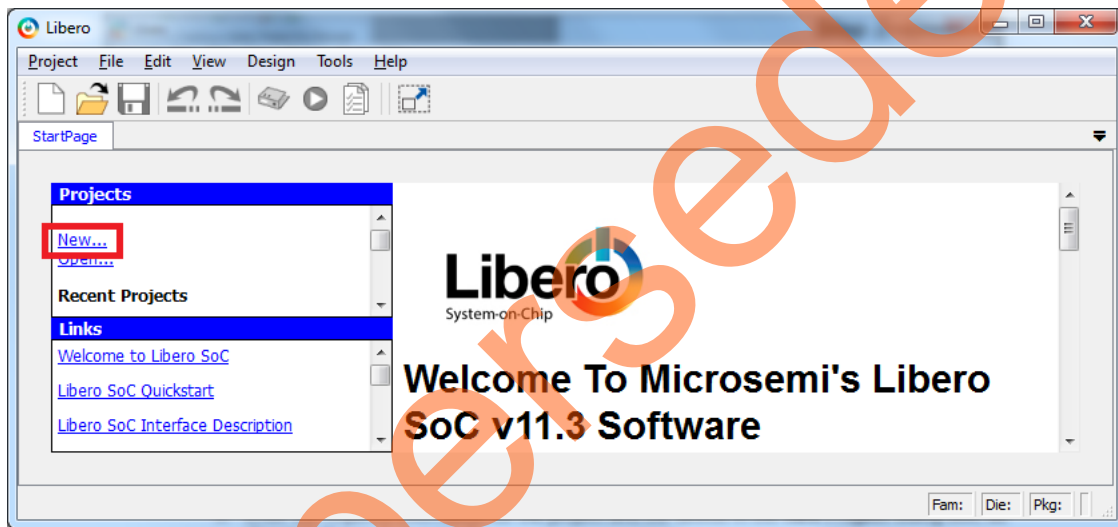


Figure 3 • Libero SoC Project Manager

3. Enter the required information for the project and the device in the **New Project** dialog box, as shown in [Figure 4](#) on page 7:
 - **Project Name:** Syssservices
 - **Project Location:** Select an appropriate location (for example, D:/SF2_Sysservices)
 - **Preferred HDL Type:** Verilog
 - **Family:** SmartFusion2
 - **Die:** M2S050T
 - **Package:** 896 FBGA
 - **Speed:** STD
 - **Die Voltage:** 1.2
 - **Operating Conditions:** COM

4. Select **Use Design Tool** and **SmartFusion2 Microcontroller Subsystem (MSS)** in the **Design Templates and Creators** section of the **New Project** window as shown in Figure 4.

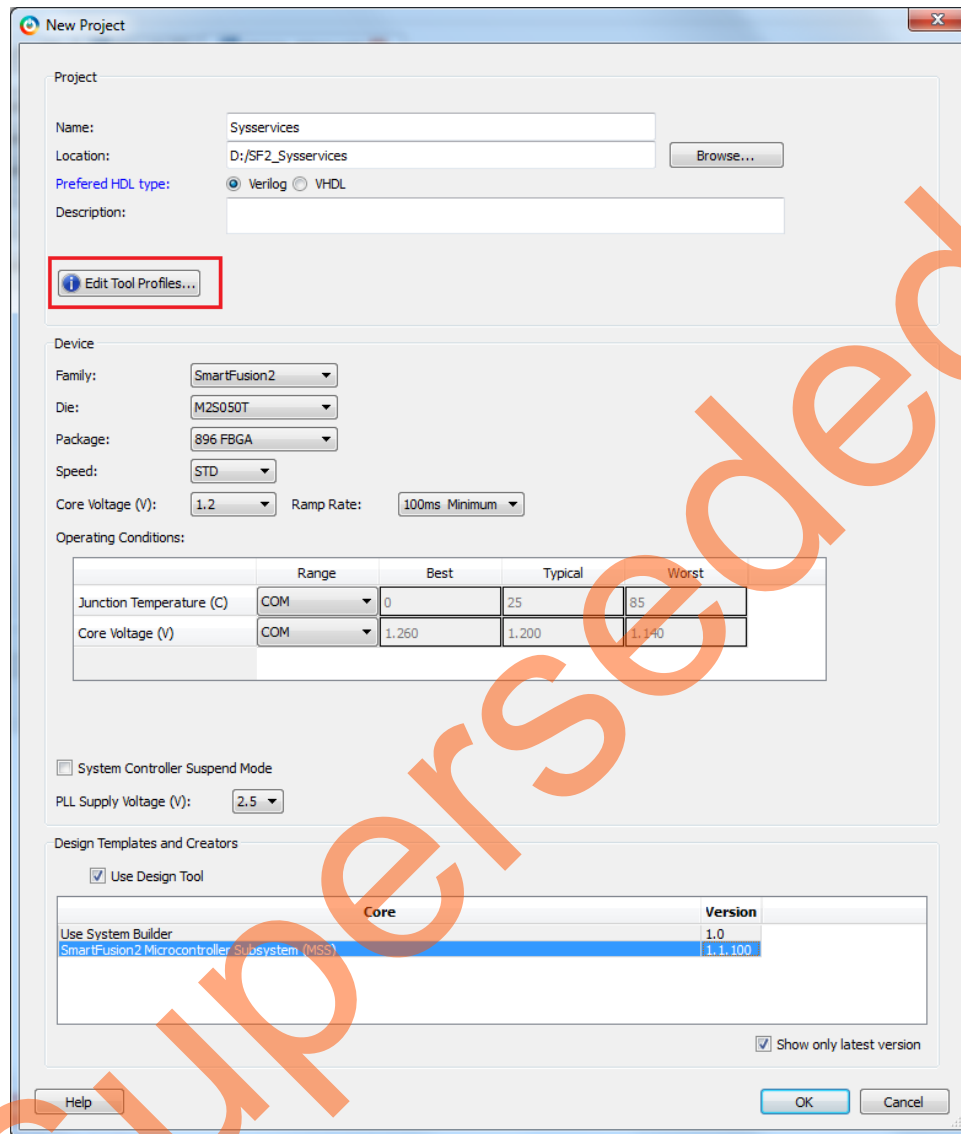


Figure 4 • Libero New Project Dialog Box

5. Click **Edit Tool Profiles** to display the **Tool Profiles** window as shown in Figure 5 on page 8.
6. In the **Tool Profiles** window, verify the following tool settings and click **OK**:
 - **Software IDE:** SoftConsole
 - **Synthesis:** Synplify Pro ME I-2013.09M-SP1
 - **Simulation:** ModelSim ME 10.2c

– **Programming:** FlashPro 11.3

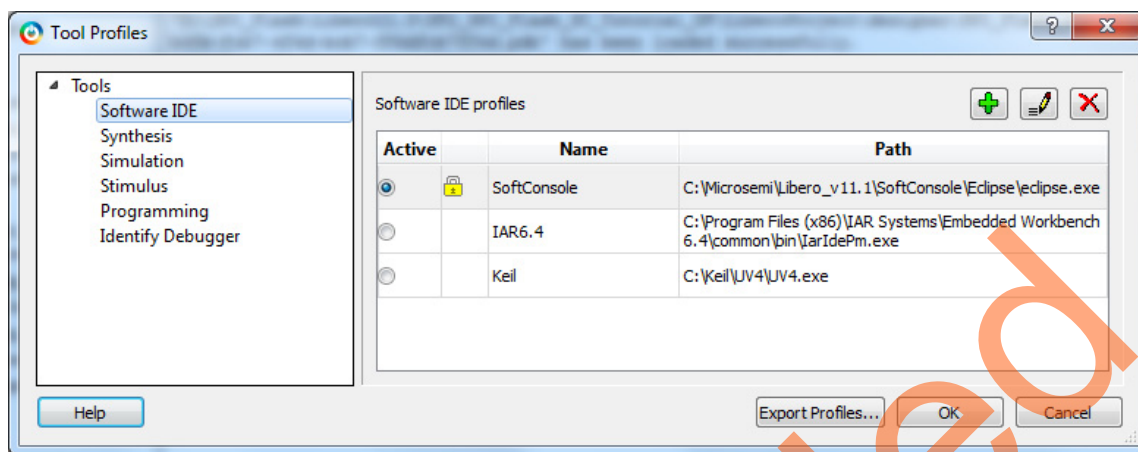


Figure 5 • Tool Profiles Window

- Click **OK** on the **New Project** window.

SmartDesign window with **Sysservices_MSS_0** component is displayed as shown in Figure 6.

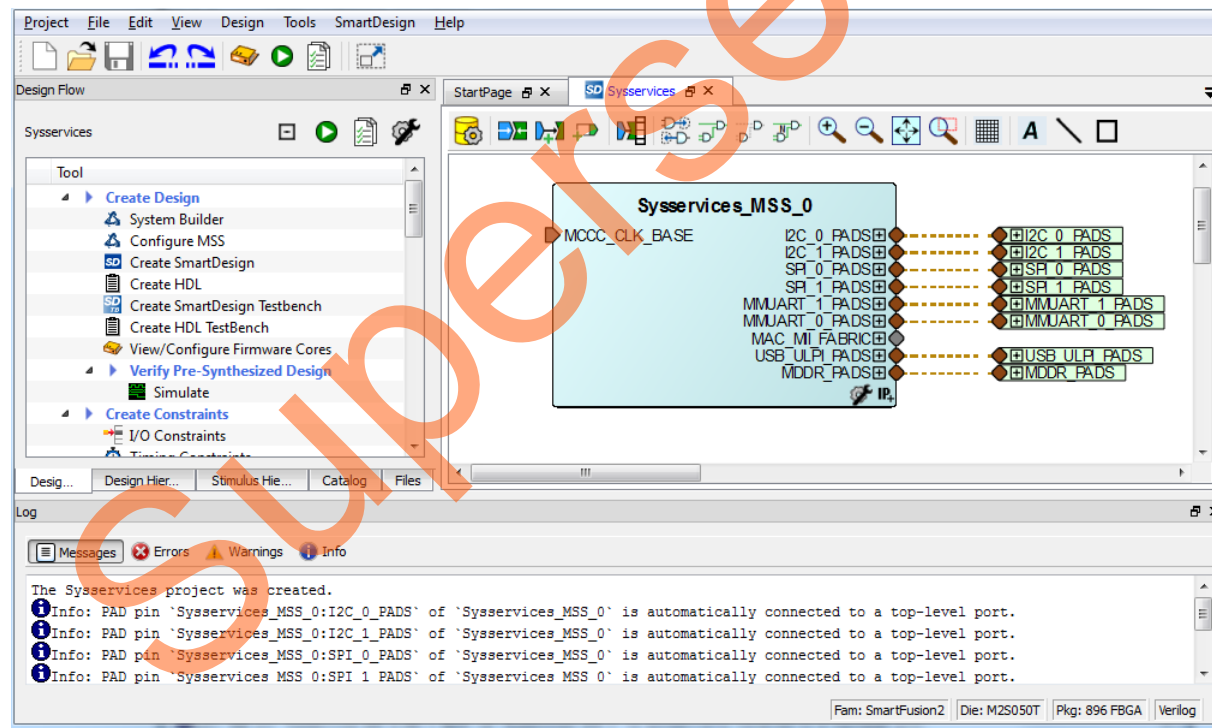


Figure 6 • SmartDesign Window with Sysservices_MSS_0 Component

- [illegible]

Figure 7 • MSS in the SmartDesign Canvas

The enabled MSS blocks are highlighted in blue. The disabled peripherals are shown in gray.

9. To disable a peripheral that is not required, right-click the peripheral block and clear the **Disable** check box, as shown in Figure 8, or clear the check box in the lower right corner of the peripheral box. The box turns gray to indicate that the peripheral has been disabled.

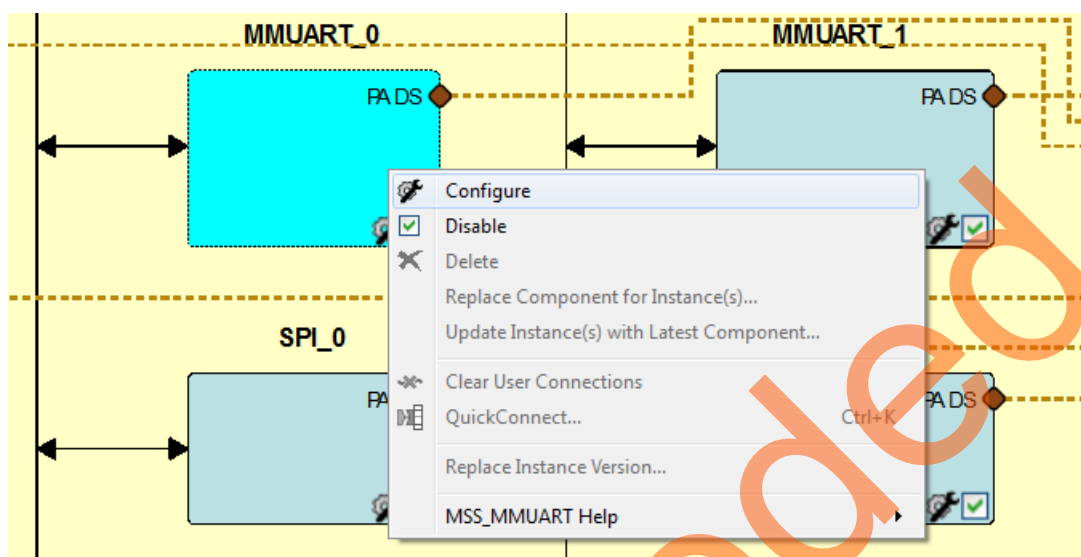


Figure 8 • Disabling a Peripheral Block

10. Disabled peripherals can be enabled by either selecting the check box in the lower right corner of the peripheral box or by right-clicking the peripheral block or selecting the **Enable** check box. An enabled peripheral is shown in Figure 9.

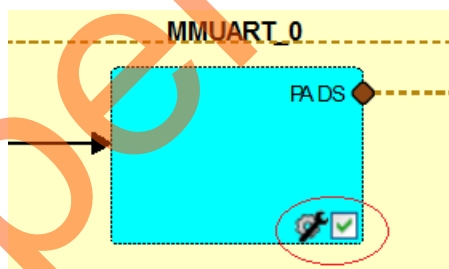


Figure 9 • Enabling the Peripheral

11. Disable the following peripherals on the MSS canvas:

- MMUART_1
- SPI_0 and SPI_1
- I2C_0 and I2C_1
- PDMA
- WATCHDOG
- FIC_0 and FIC_1
- USB
- Ethernet
- RTC

- MDDR PADS
- CAN
- GPIO

After disabling the above components, the MSS configuration window is displayed similar to Figure 10.

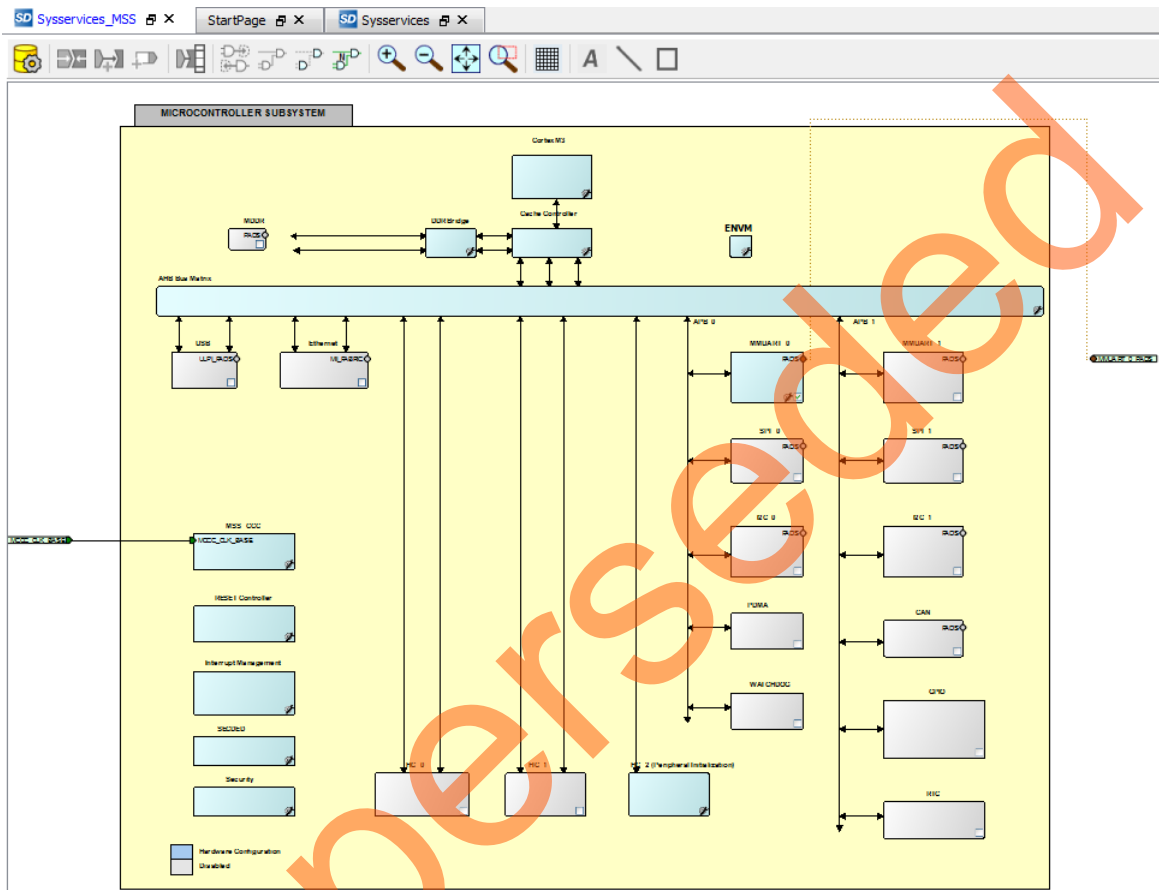


Figure 10 • Enabled and Disabled MSS Components

12. Double-click **MMUART_0** and configure it as shown in Figure 11.
 - Select **Main Connection** as **Fabric** for TXD and RXD Full Duplex MSIOs
 - Click **OK** on **MMUART_0 Configurator** to complete the configuration

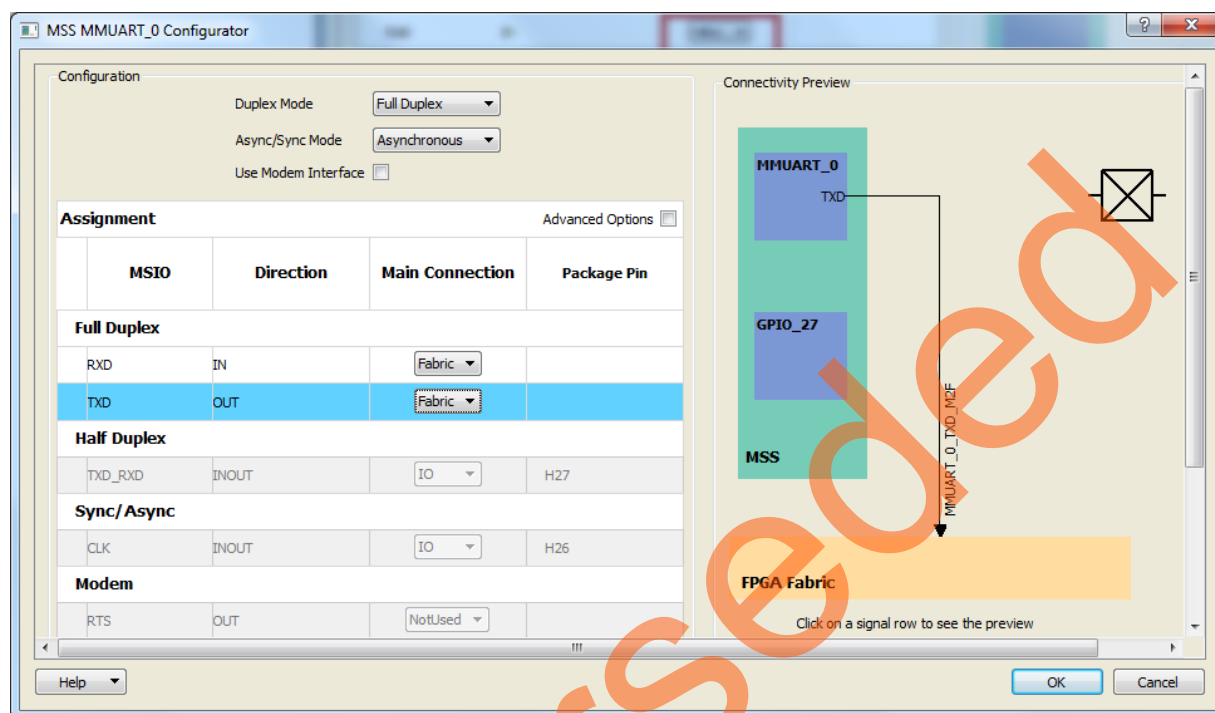


Figure 11 • MMUART_0 Configurator Window

13. Double-click **MSS_CCC** block to configure as shown in Figure 12 on page 13:
 - The clock input is by default selected as CLK_BASE, select **CLK_BASE** frequency as **50 MHz**
 - Select the check box for **Monitor FPGA Fabric PLL Lock (CLK_BASE_PLL_LOCK)**
 - Select **M3_CLK** as **50 MHz**
 - Select **APB_0_CLK** and **APB_1_CLK** frequency as **M3_CLK/1**
 - Leave the rest as default
 - Click **OK** on **MSS Clock Configurator** to complete the clock configuration

The above selection configures the MSS CCC to receive the input clock from the fabric CCC. The lock input of the MSS CCC is configured to be received from the fabric CCC block.

This completes the configuration of the MSS.

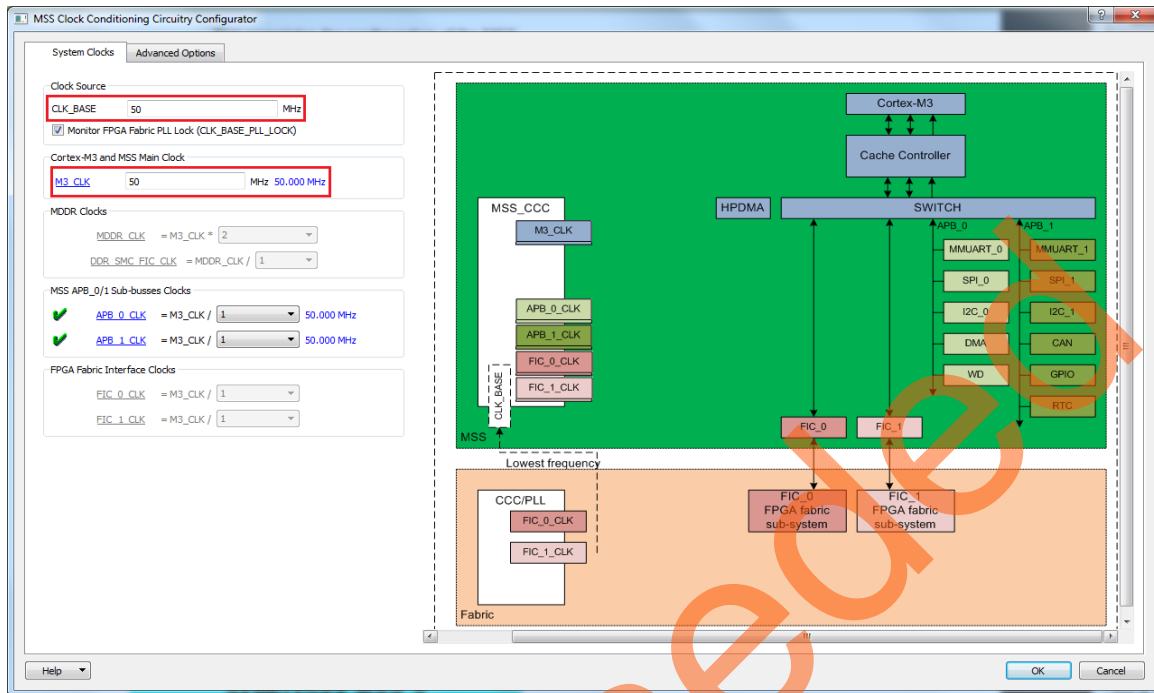


Figure 12 • MSS Clock Conditioning Circuitry Configurator

14. Select **File > Save** to save **Sysservices_MSS**. Select the **Sysservices** tab on the Smart Design canvas, right-click **Sysservices_MSS_0**, and select **Update Instance(s) with Latest Component**, as shown in Figure 13.

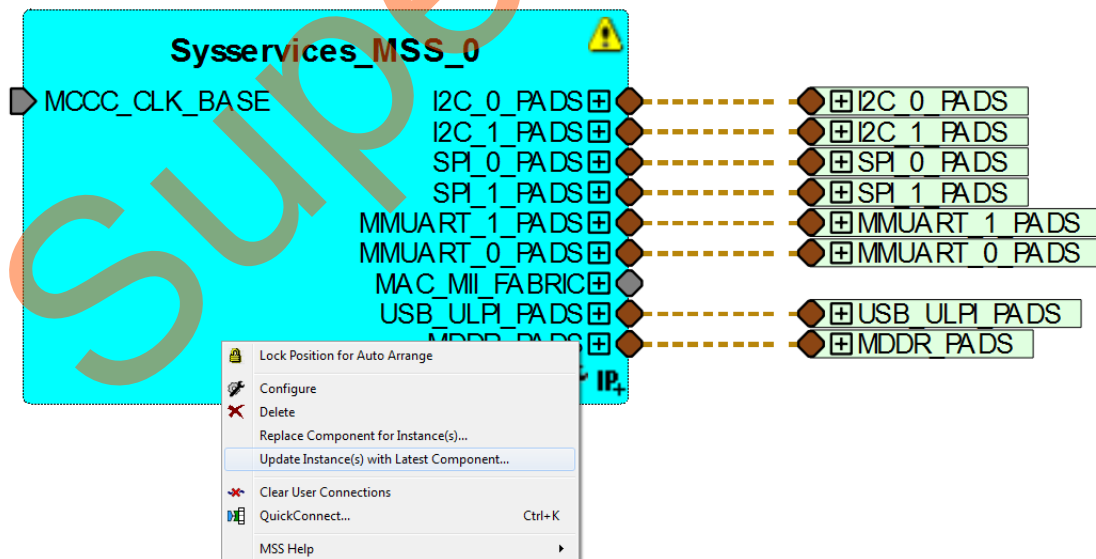


Figure 13 • Updating the Sysservices_MSS_0

After successfully updating, the **Sysservices_MSS_0** instance displays similar to [Figure 14](#).

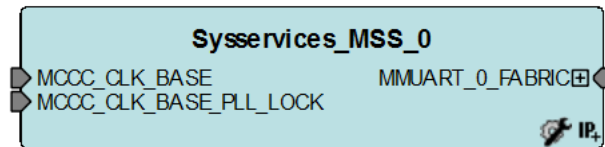


Figure 14 • Updated Sysservices_MSS_0

Instantiating Clock Conditioning Circuitry (CCC) and Chip Oscillators in Sysservices SmartDesign

The Libero SoC catalog provides IP cores that can be easily dragged and dropped into the SmartCanvas workspace. Many of these IPs are free to use while several require a license agreement.

1. Click on the **Catalog** tab and drag and drop the **Clock Conditioning Circuitry (CCC) and Chip Oscillators** block from the **Clock and Management** sub-section of the **IP Catalog** into **Sysservices SmartDesign** window as shown in [Figure 15](#).

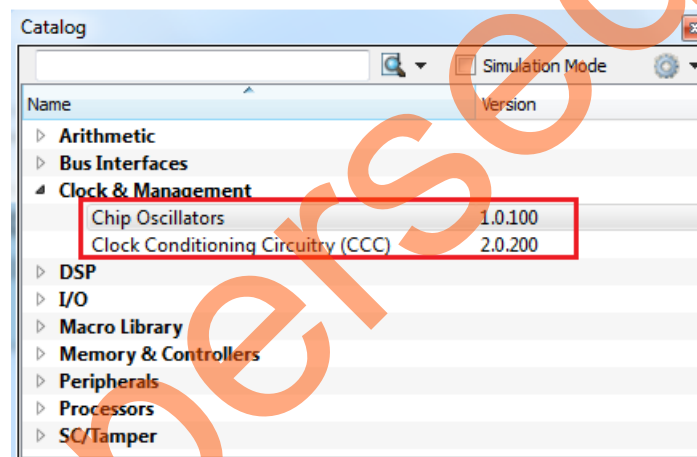


Figure 15 • Catalog Window

2. Double click **FCCC_0** to configure it. [Figure 16](#) on page 15 shows the FAB CCC Configurator window.
 - Select **Reference Clock** as **50 MHz** and **25/50 MHz Oscillator** from the drop-down list.
 - Select **GL0 Frequency** as **50 MHz**.
 - Click **OK** to complete the configuration.

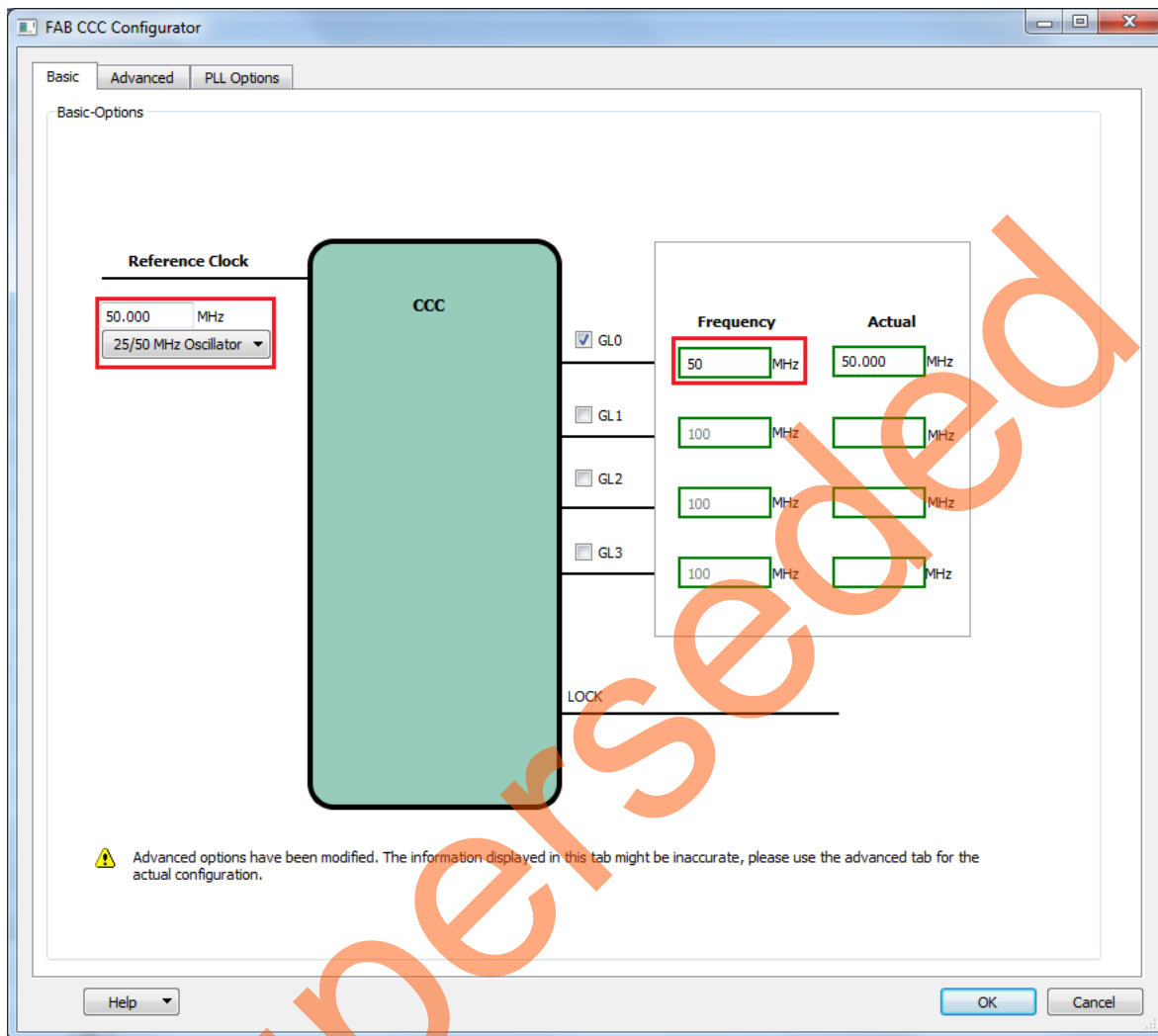


Figure 16 • FAB CCC Configurator

3. Double-click the **OSC_0** instance to open its **Configuration** window. Make the following selections, as shown in Figure 17 on page 16:
 - Select the check box for **On Chip 25/50 MHz RC Oscillator**
 - Select the check box for **Drives Fabric CCC(s)**
 - Leave the rest as default
 - Click **OK** to complete the configuration

On-chip 50 MHz RC oscillator drives the input of the fabric CCC block instantiated earlier.

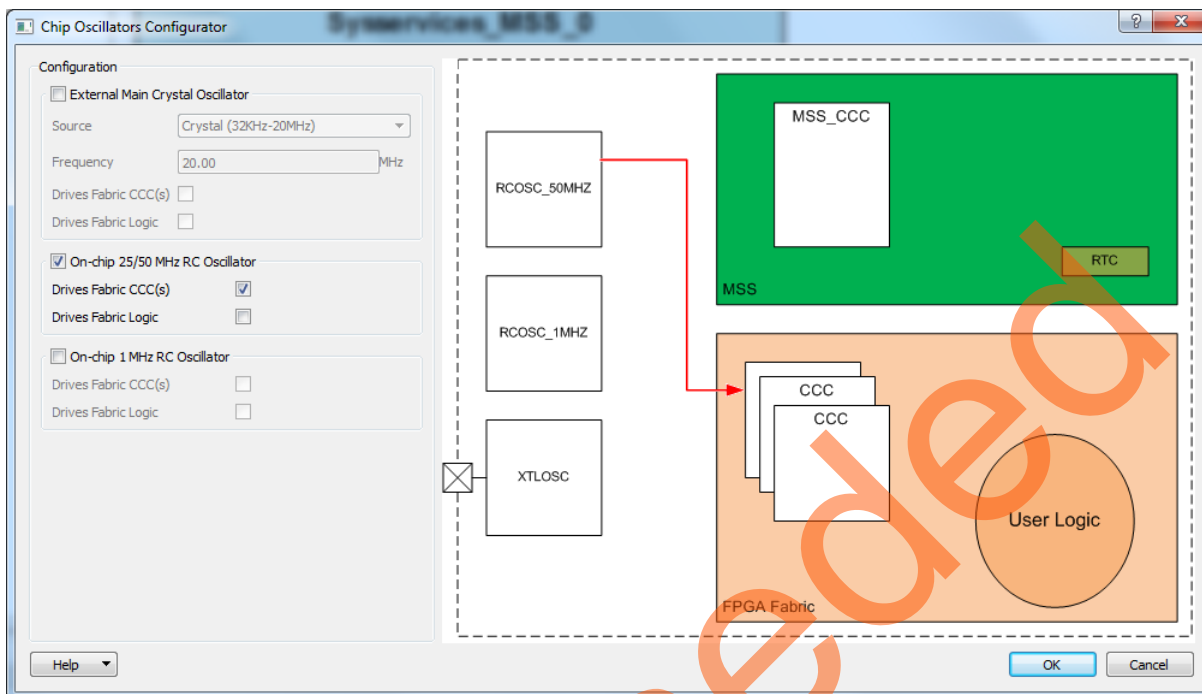


Figure 17 • Chip Oscillator Configurator

Connecting Components in SysServices SmartDesign

There are three methods for connecting components in SysServices SmartDesign:

- The first method is by using the **Connection Mode** option. To use this method, change the SmartDesign to connection mode by clicking the **Connection Mode** button on the SmartDesign window, as shown in Figure 18 on page 17. The cursor changes from the normal arrow shape to the connection mode icon shape. To make a connection in this mode, click on the first pin and drag-drop to the second pin that you want to connect.
- The second method is by selecting the pins to be connected together and selecting **Connect** from the context menu. To select multiple pins to be connected together, hold the **CTRL** key while selecting the pins. Right-click the input source signal and select **Connect** to connect all the signals together. In the same way, select the input source signal, right-click it, and select **Disconnect** to disconnect the signals that are already connected.
- The third method is by using the **Quick Connect** option. To use this method, change the SmartDesign to quick connect mode by clicking on **Quick Connect** mode on the SmartDesign window, as shown in Figure 18 on page 17. Quick connect window will be opened. Find the **Instance Pin** you want to connect and click to select it. In **Pins to Connect**, find the pin you wish to connect, right-click and choose **Connect** as shown in Figure 19 on page 18.

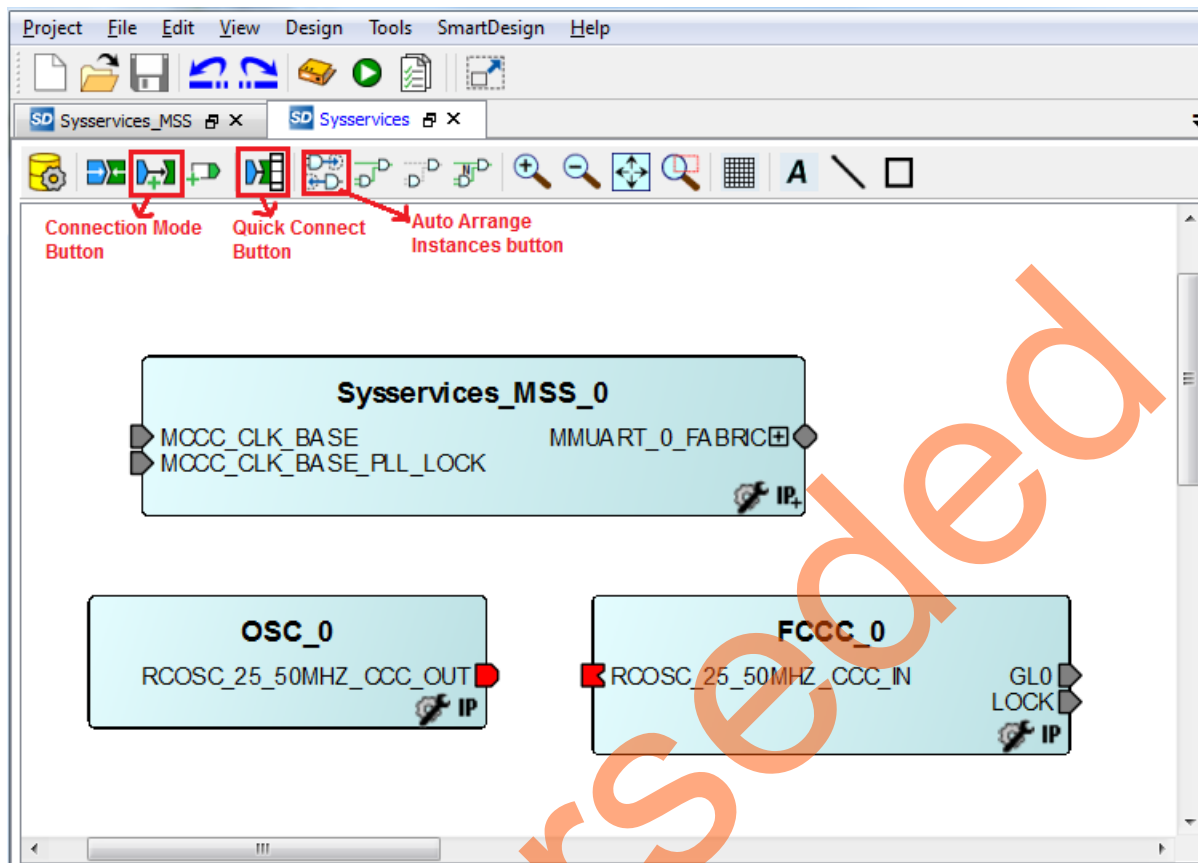


Figure 18 • Sysservices SmartDesign

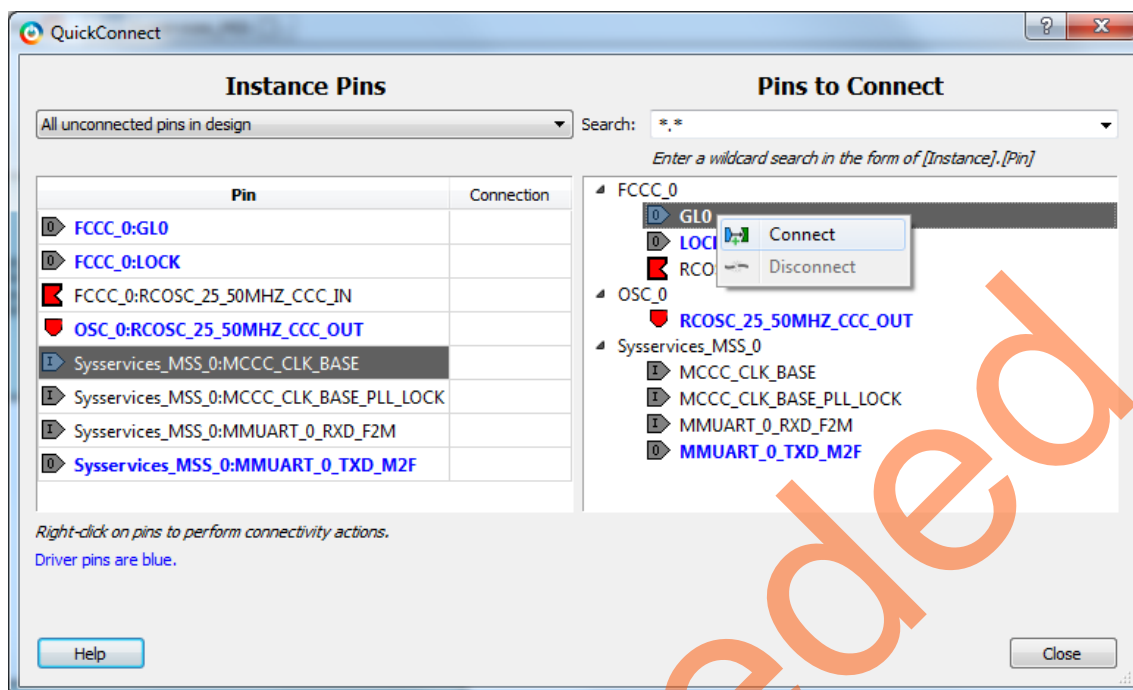


Figure 19 • Quick Connect Window

Use one of the three options described above and make the following connections:

1. Connect **ROSC_25_50MHZ_CCC_OUT** of **OSC_0** to **ROSC_25_50MHZ_CCC_IN** of **FCCC_0**.
2. Connect **GL0** of **FCCC_0** to **MCCC_CLK_BASE** of **Sysservices_MSS_0**.
3. Connect **LOCK** of **FCCC_0** to **MCCC_CLK_BASE_PLL_LOCK** input of **Sysservices_MSS_0**.
4. Right-click the **MMUART_0_FABRIC** of **Sysservices_MSS_0** and select **Promote to Top Level**.
5. Click **Auto Arrange Instances** to arrange the instances and click **File > Save**. Libero top-level design is displayed as shown in [Figure 20](#).

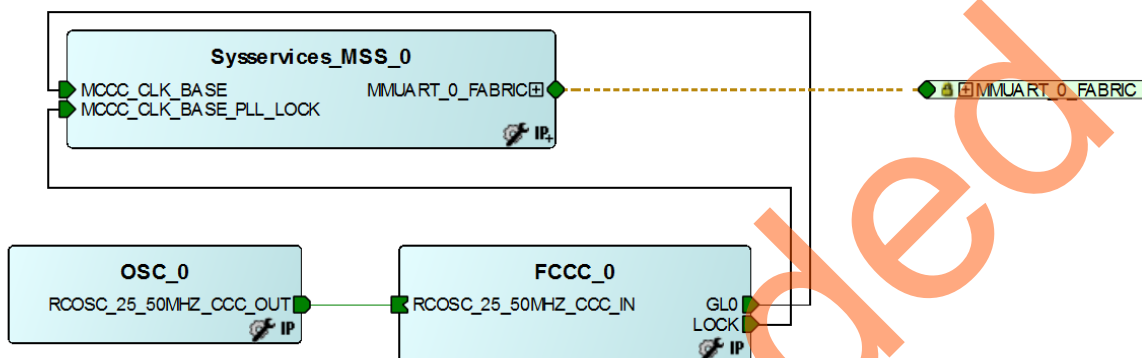


Figure 20 • Syservices SmartDesign

Configuring and Generating Firmware

1. Click **Design > Configure Firmware** in Libero SoC to open the **DESIGN_FIRMWARE** window, as shown in [Figure 21](#).

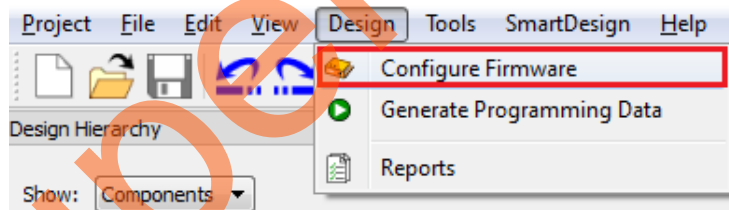


Figure 21 • Opening DESIGN_FIRMWARE Window

2. Clear all drivers except CMSIS, MMUART_0 and System Services as shown in Figure 22.

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

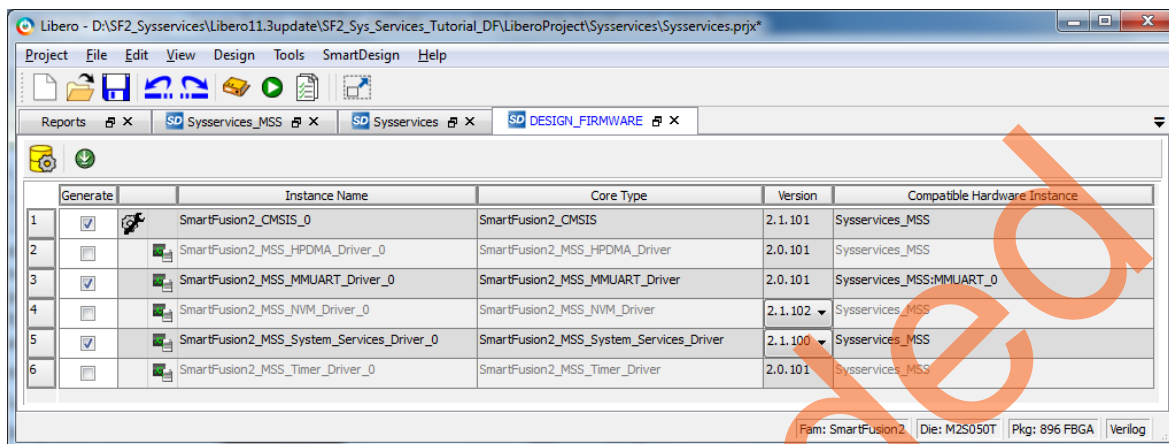


Figure 22 • DESIGN_FIRMWARE Window

3. Select **Sysservices** tab on the SmartDesign canvas and click **Generate Component** as shown in Figure 23.

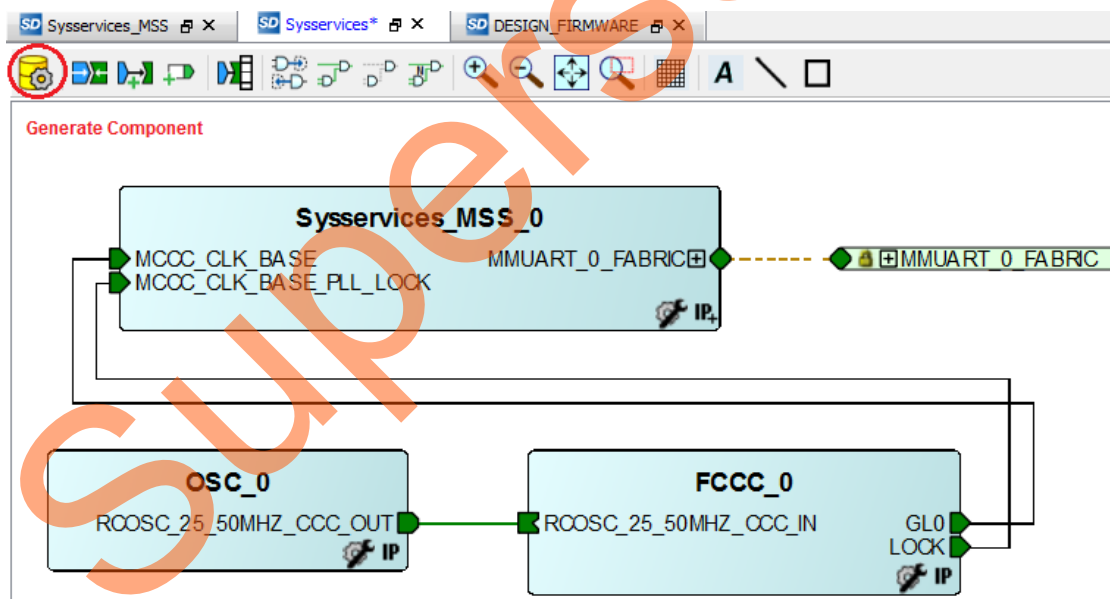


Figure 23 • Generate Component

After successful generation of all the components, the following message is displayed on the log window, as shown in Figure 24 on page 21.

Info: 'Sysservices' was generated.

Libero also generates the SoftConsole software project.

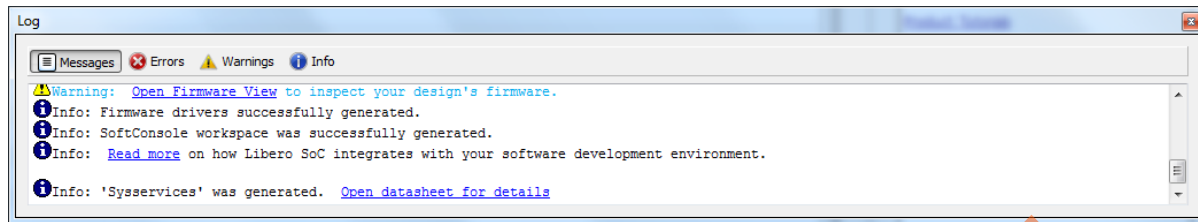


Figure 24 • Log Window

Step 3: Generating the Program File

1. Double-click **I/O Constraints** in the **Design Flow** window as shown in Figure 25. The **I/O Editor** window is displayed after completing **Synthesize** and **Compile**.

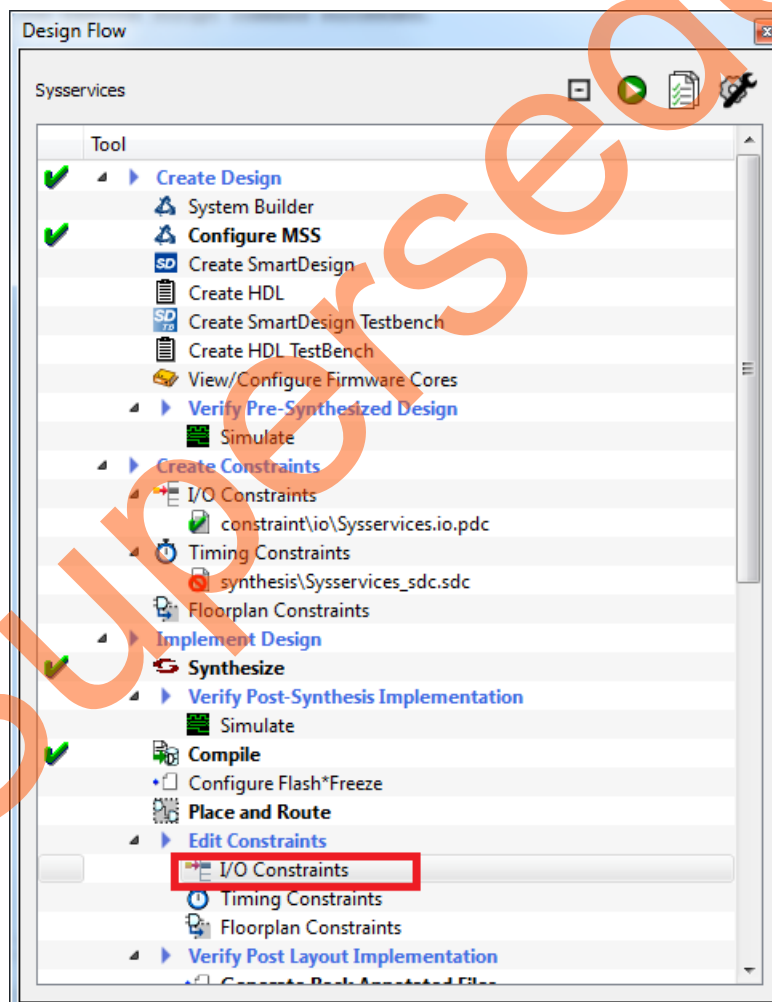


Figure 25 • Design Flow Window

2. Make the pin assignments shown in Table 2. After the pins have been assigned, the I/O Editor is displayed as shown in Figure 26.

Table 2 • Port-to-Pin Mapping

Port Name	Pin Number
MMUART0_0_RXD_F2M	R29
MMUART0_0_TXD_M2F	R24

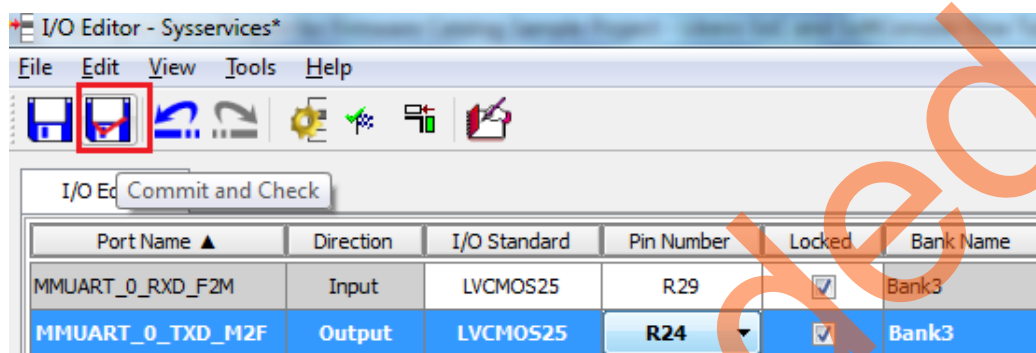


Figure 26 • I/O Editor

3. After updating I/O editor, click **Commit and Check**.
4. Close the I/O editor.
5. Click **Generate Programming Data** as shown in Figure 27 to complete place and route, and generate the programming file.

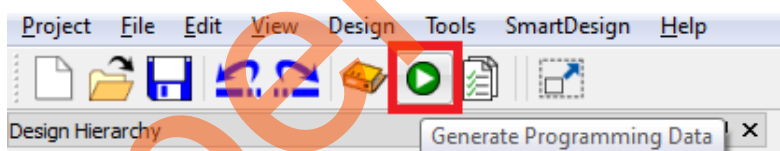


Figure 27 • Generate Programming Data

Step 4: Programming the SmartFusion2 Board Using FlashPro

1. Connect the FlashPro4 programmer to the J59 connector of SmartFusion2 SoC FPGA Development Kit.
2. Connect the jumpers on the SmartFusion2 Development Kit board, as described in [Table 3](#). For more information on jumper locations, refer to ["Appendix 1: Jumper Locations"](#) on page 35.

Caution: While making the jumper connections, the power supply SW7 switch on the board should be in **OFF** position.

Table 3 • SmartFusion2 SoC FPGA Development Kit Jumper Settings

Jumper Number	Settings	Notes
J70, J93, J94, J117, J123, J142, J157, J160, J167, J225, J226, J227	1-2 closed	These are the default jumper settings of the Development Kit. Make sure that these jumpers are set properly.
J2	1-3 closed	
J23	2-3 closed	

3. Connect the power supply to the J18 connector.
4. Switch the power supply switch SW7 to **ON** position. Refer to ["Appendix 2: Board Setup for Running the Tutorial"](#) on page 36 for information on the board setup for running the tutorial.
5. To program the SmartFusion2 device, double-click **Run Programming Action** in the **Design Flow** window as shown in [Figure 28](#).

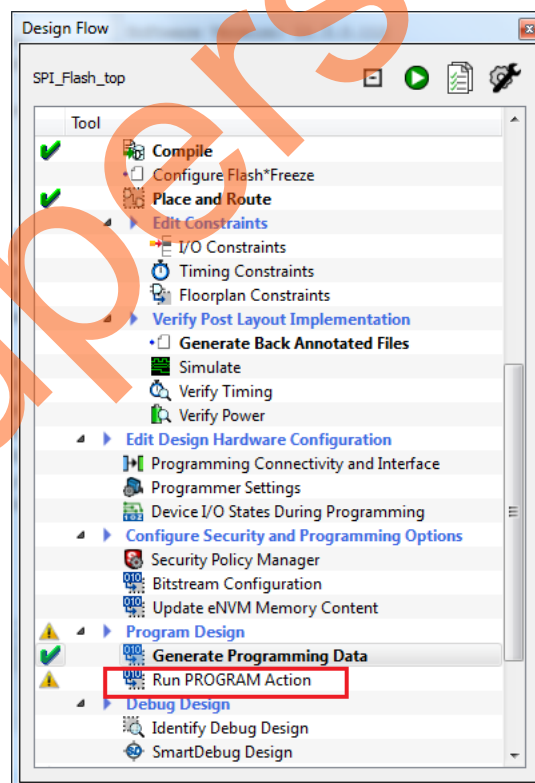


Figure 28 • Run Programming Action

Step 5: Building the Software Application using SoftConsole

1. After successful programming, open the SoftConsole project by double-clicking **Write Application Code** under **Develop Firmware** in **Design Flow** window as shown in Figure 29.

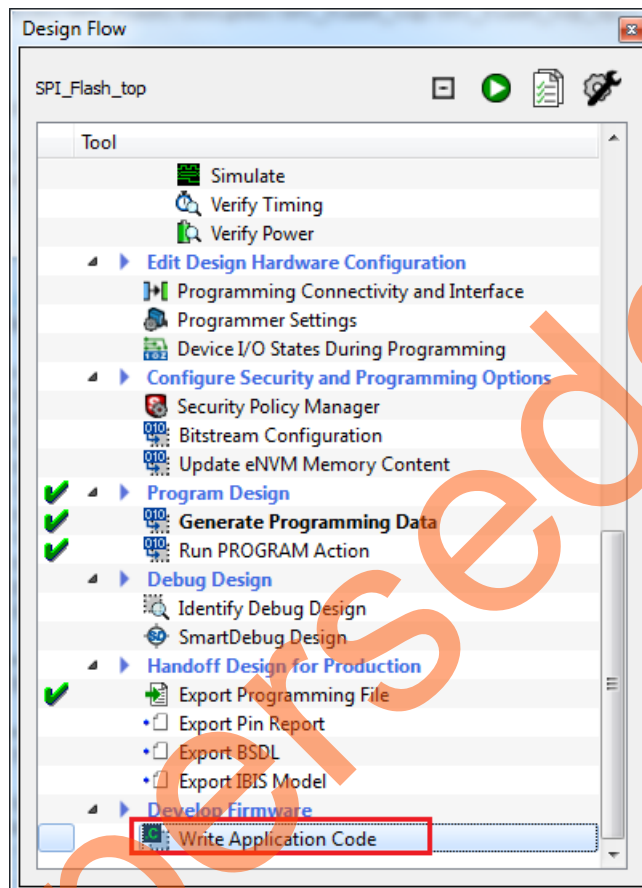


Figure 29 • Invoking SoftConsole from the Libero SoC

The SoftConsole perspective is displayed as shown in Figure 30.

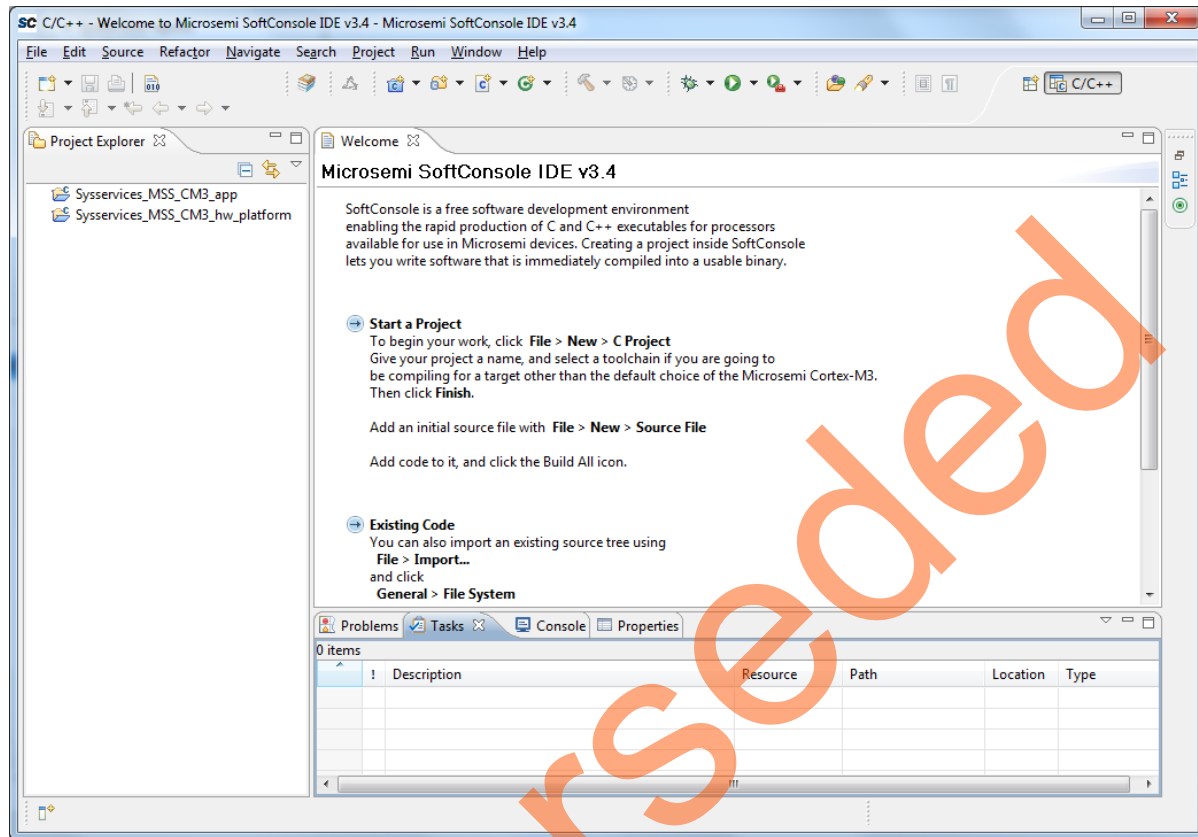


Figure 30 • SoftConsole Workspace

- Go to the location where the SoftConsole sample Firmware catalog project is saved, as shown in Figure 31.

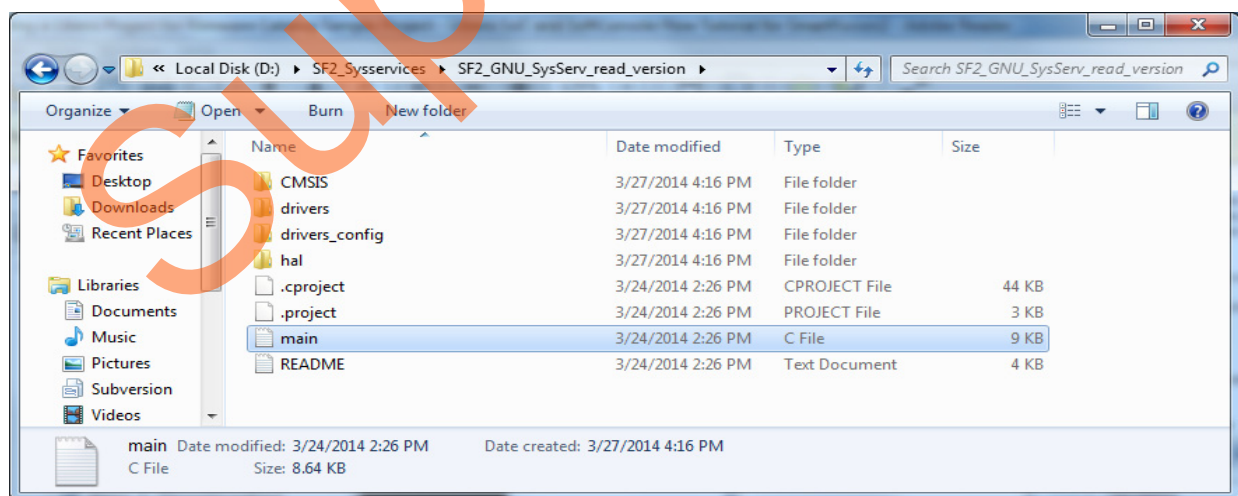


Figure 31 • Sample Project main. c File

- Copy the `main.c` file and replace it with the existing `main.c` file under **Sysservices_MSS_CM3_app** project in the SoftConsole workspace. The SoftConsole window looks as shown in [Figure 32](#).

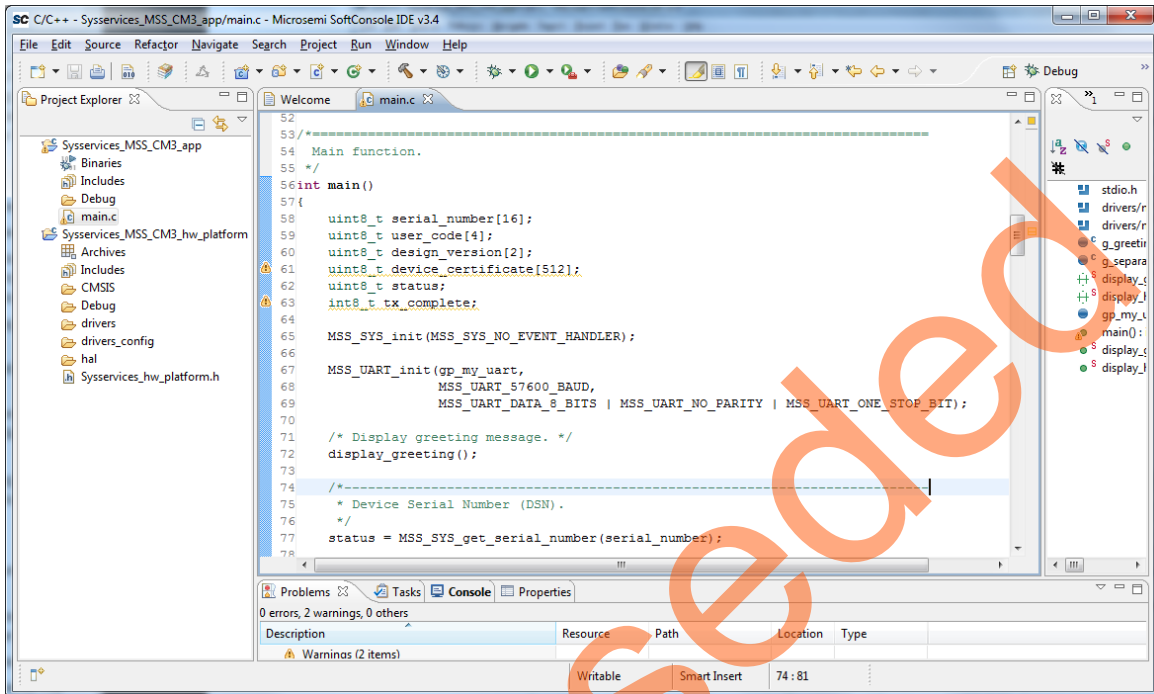


Figure 32 • SoftConsole Workspace - main.c File

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

Modified `main.c` is available in ["Appendix 4: main.c File"](#) on page 38.

[Figure 33](#) on page 27 shows the SoftConsole workspace with modified `main.c` file

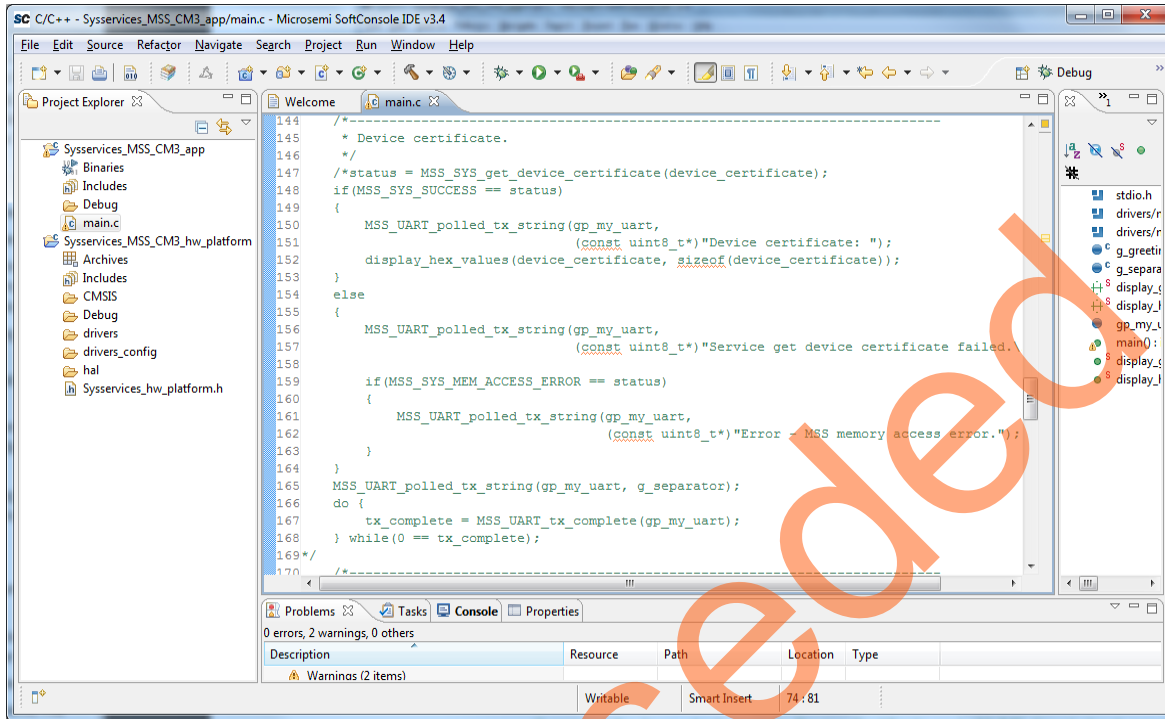


Figure 33 • SoftConsole Workspace - Modified main.c File

4. Right-click **Sysservices_MSS_CM3_app** in **Project Explorer** window of SoftConsole project and select **Properties** as shown in Figure 34.

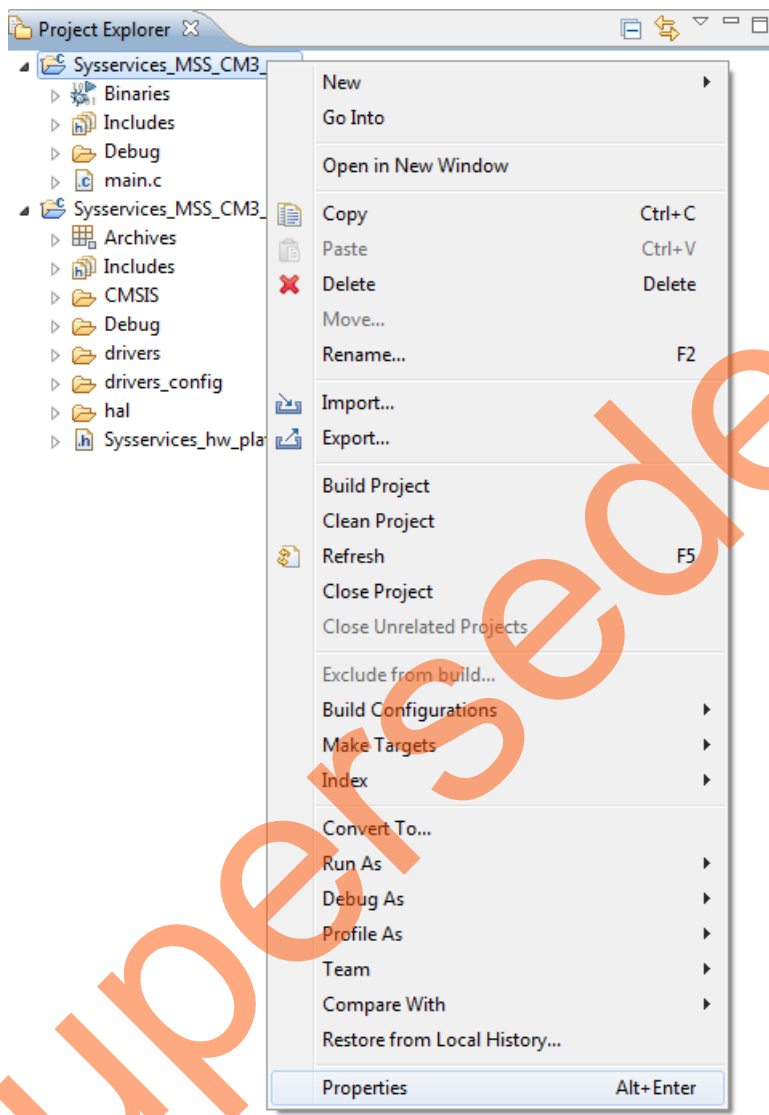


Figure 34 • Project Explorer Window of SoftConsole Project

5. 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 35](#). Click **Apply** and **OK**.

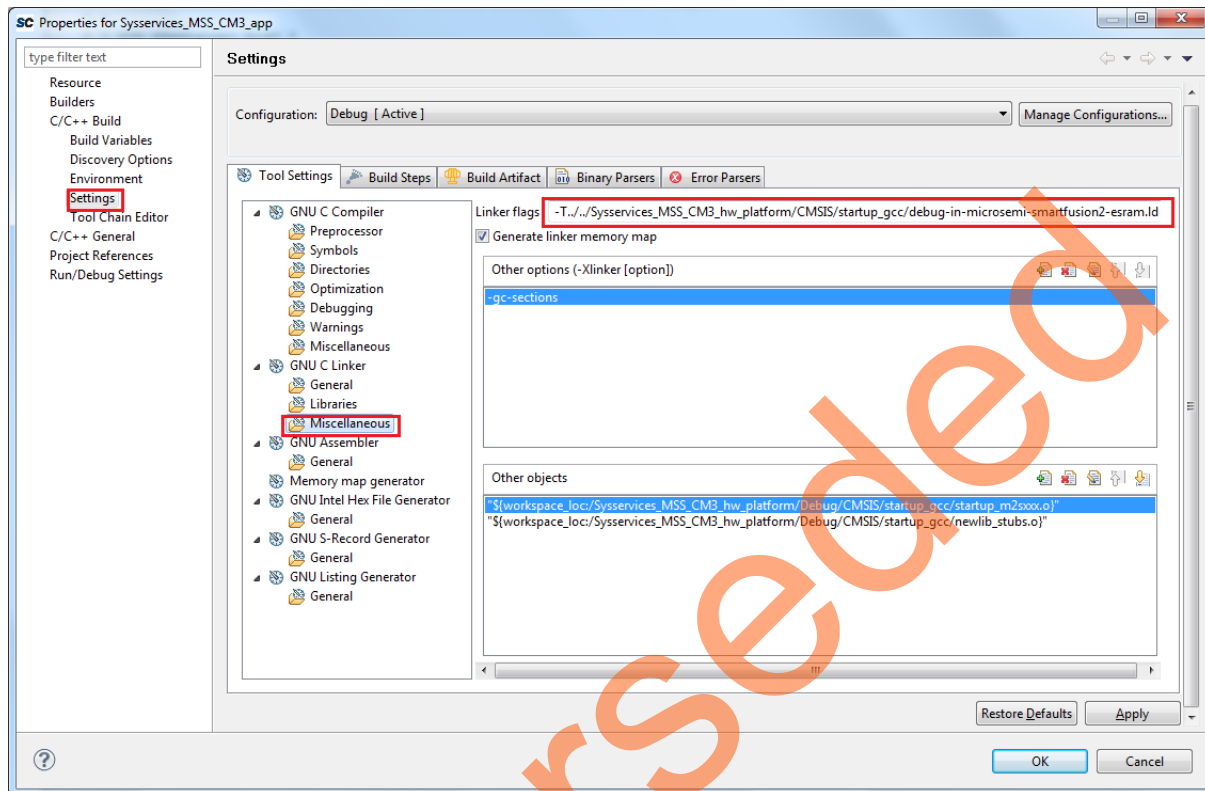


Figure 35 • Syssservices_MSS_CM3_app Properties Window

6. Perform a clean build by selecting **Project > Clean**. Accept the default settings in the **Clean** dialog box and click **OK**, as shown in Figure 36. SoftConsole project must not have any errors.

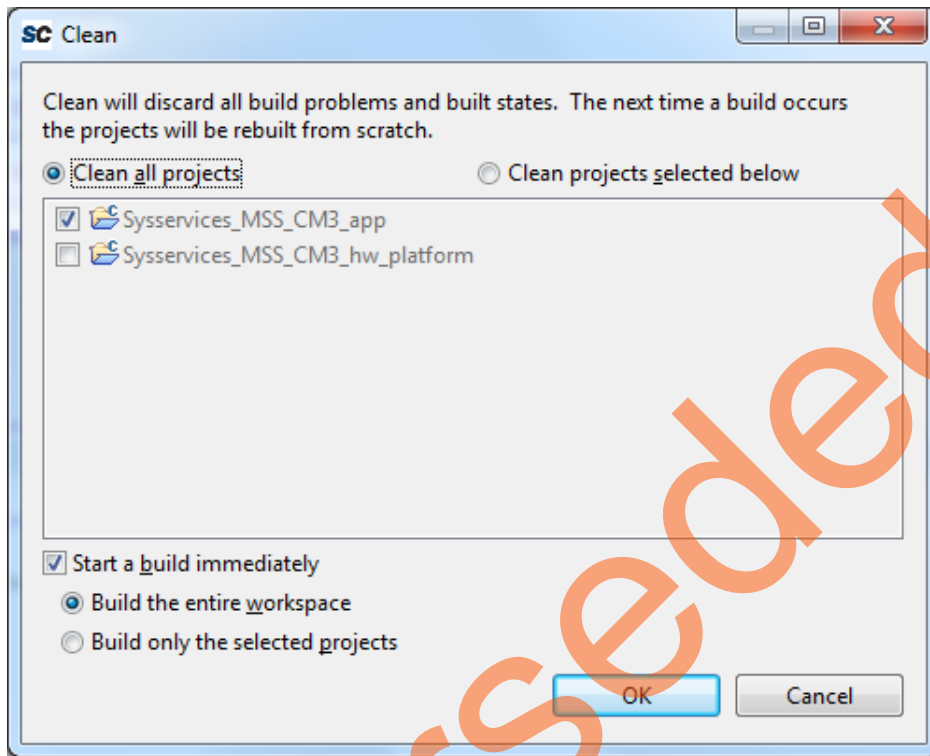


Figure 36 • Settings for a Clean Build

7. 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
8. Connect the host PC to the J24 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 37 on page 31.

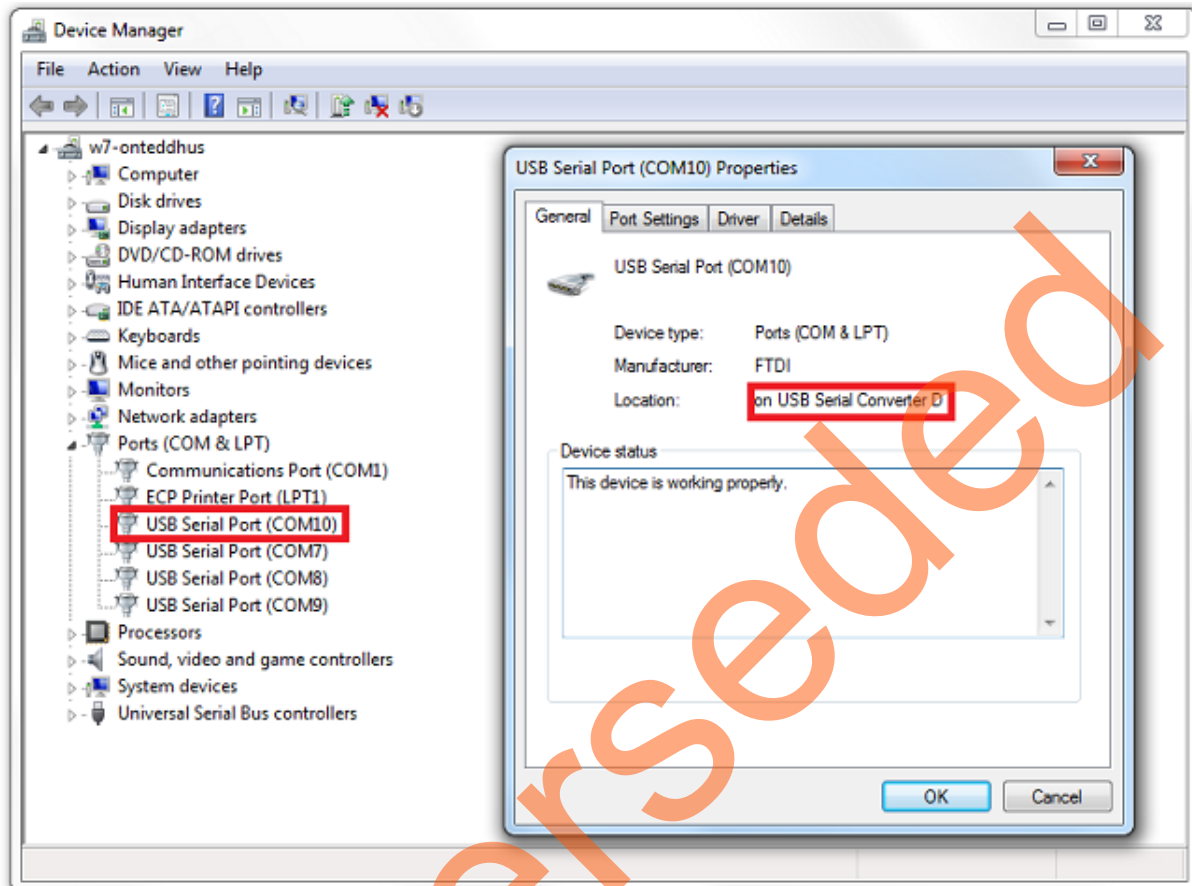


Figure 37 • Device Manager Window

9. Start the PuTTY session. If the PuTTY program is not available in the computer system, any free serial terminal emulation program such as HyperTerminal or TeraTerm can be used. Refer to the [Configuring Serial Terminal Emulation Programs Tutorial](#) for configuring the HyperTerminal, TeraTerm, or PuTTY.

The PuTTY settings are as follows:

- 57,600 baud rate
- 8 data bits
- 1 stop bit
- No parity
- No flow control

10. Select **Debug Configurations** from the **Run** menu of the SoftConsole. The **Debug** dialog box is displayed. Double-clicking the **Microsemi Cortex-M3 Target** displays a window similar to Figure 38.

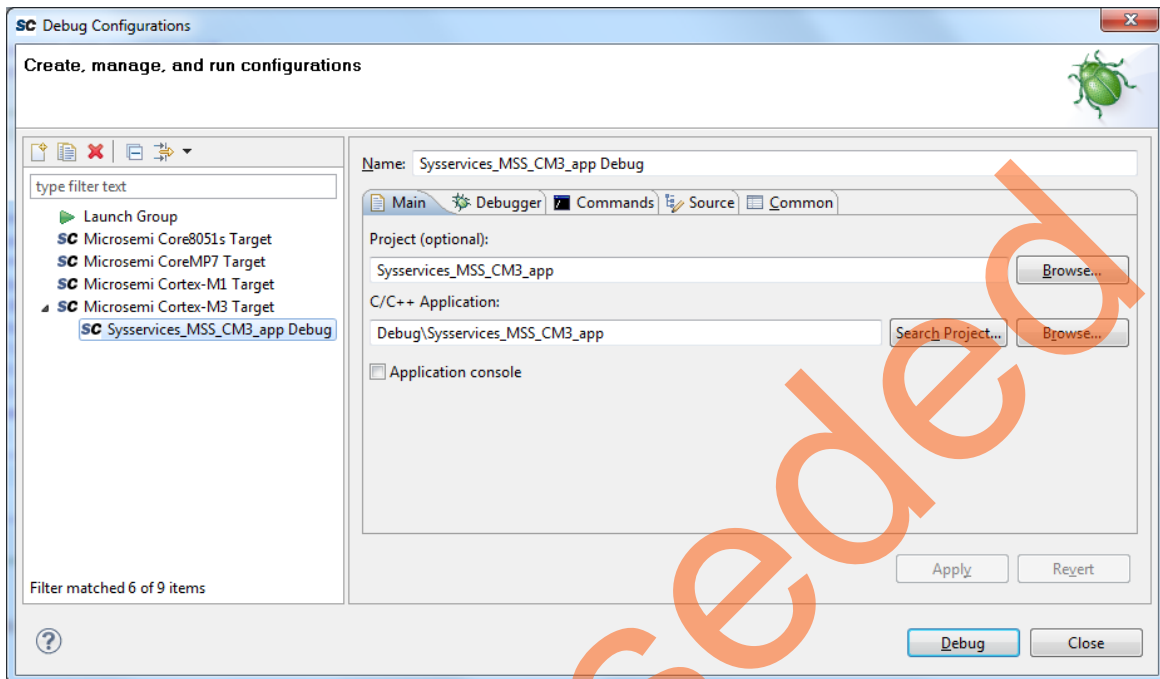


Figure 38 • Debug Configurations Window

11. Confirm that the following details appear on the **Main** tab in the **Debug** window and click **Debug**:
 - **Name:** Syssservices_MSS_CM3_app Debug
 - **Project:** Syssservices_MSS_CM3_app
 - **C/C++ Application:** Debug\Syssservices_MSS_CM3_app
12. Click **Yes** when prompted for the **Confirm Perspective Switch**, as shown in Figure 39. This displays the debug view mode.

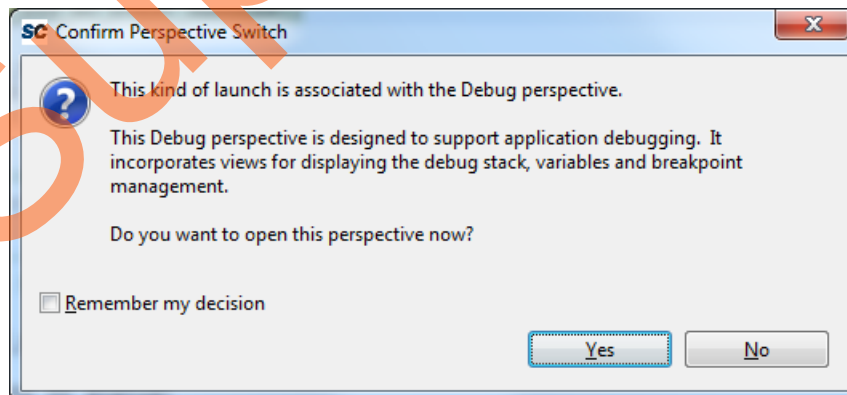


Figure 39 • Confirm Perspective Switch

The **SoftConsole Debugger Perspective** window is opened, as shown in Figure 40.

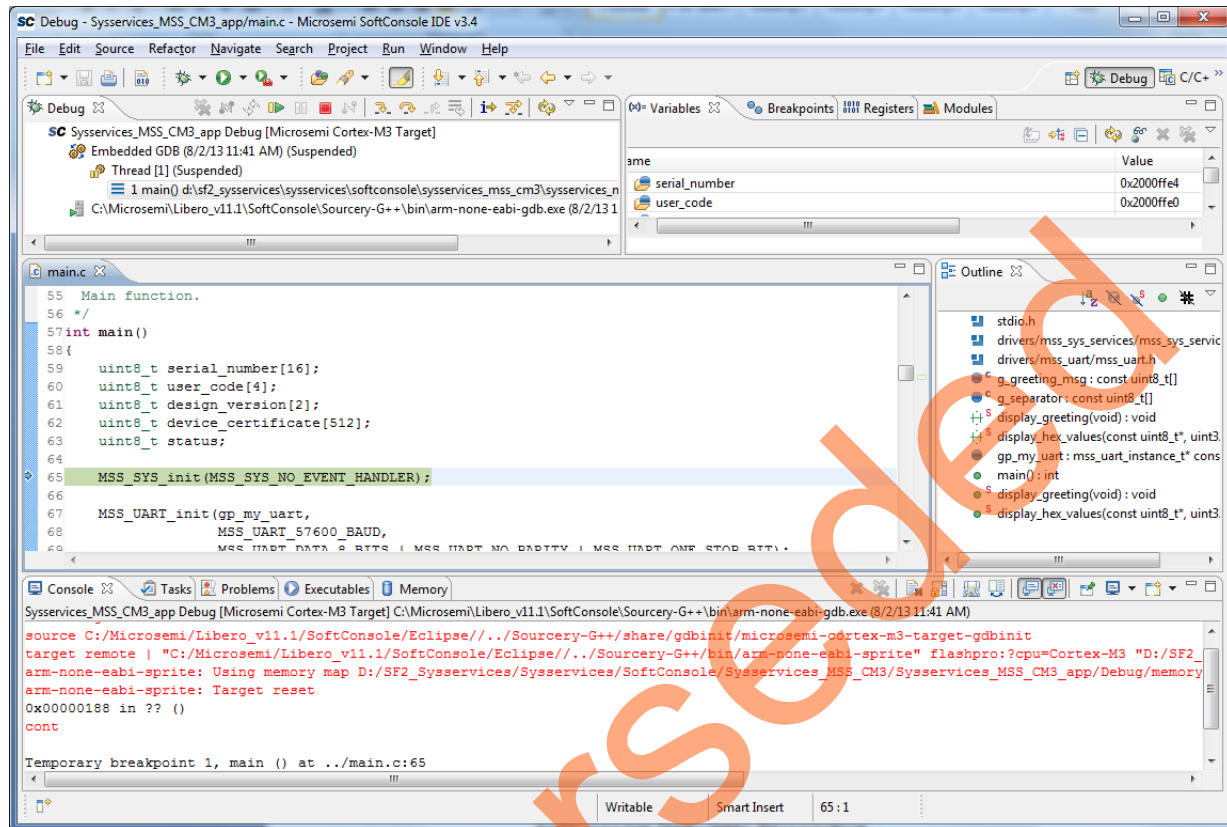
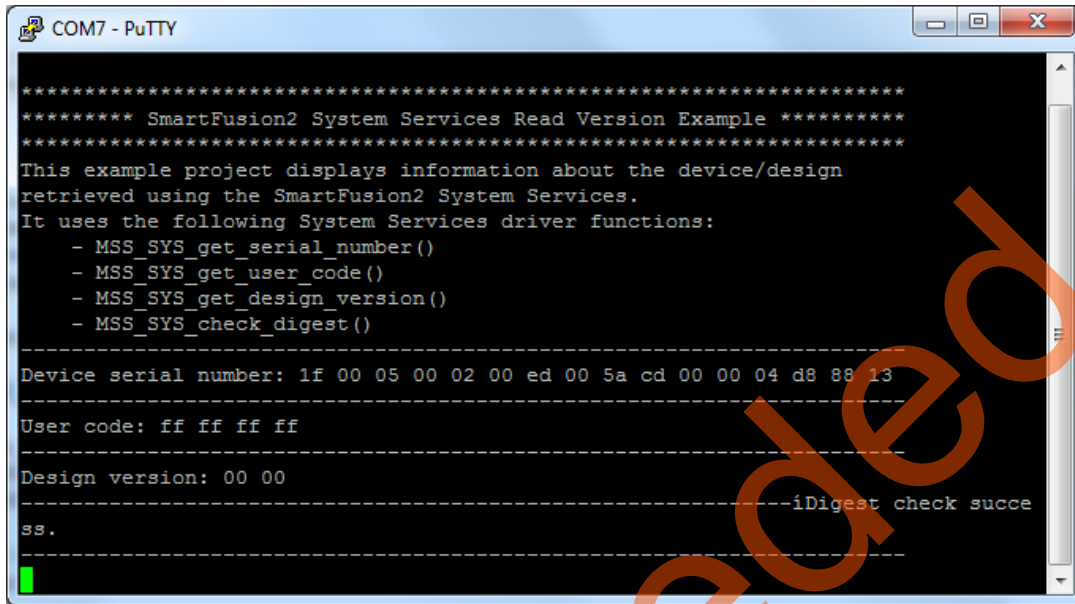


Figure 40 • SoftConsole Debugger Perspective

13. Run the application by clicking **Run > Resume**. Information on the SmartFusion2 device and the design, along with a greeting message is displayed on the PuTTY, as shown in [Figure 41](#).



```
***** 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_check_digest()

-----
Device serial number: 1f 00 05 00 02 00 ed 00 5a cd 00 00 04 d8 88 13
-----
User code: ff ff ff ff
-----
Design version: 00 00
-----
-----iDigest check succe
ss.
-----
```

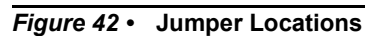
Figure 41 • PuTTY Window

14. Terminate execution of the code by choosing **Run > Terminate**.
15. Close **Debug Perspective** by selecting **Close Perspective** from the **Window** menu.
16. Close SoftConsole using **File > Exit**.
17. 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 Development Kit. A sample project for implementing System Services features is created to display the SmartFusion2 device and design information.

Figure 42 shows the jumper locations in SmartFusion2 Development Kit Board.



- Jumpers highlighted in red are set by default.
- The location of the jumpers in [Figure 42](#) are searchable.

Appendix 2: Board Setup for Running the Tutorial

Figure 43 shows the board setup for running the tutorial on the SmartFusion2 Development Kit Board.

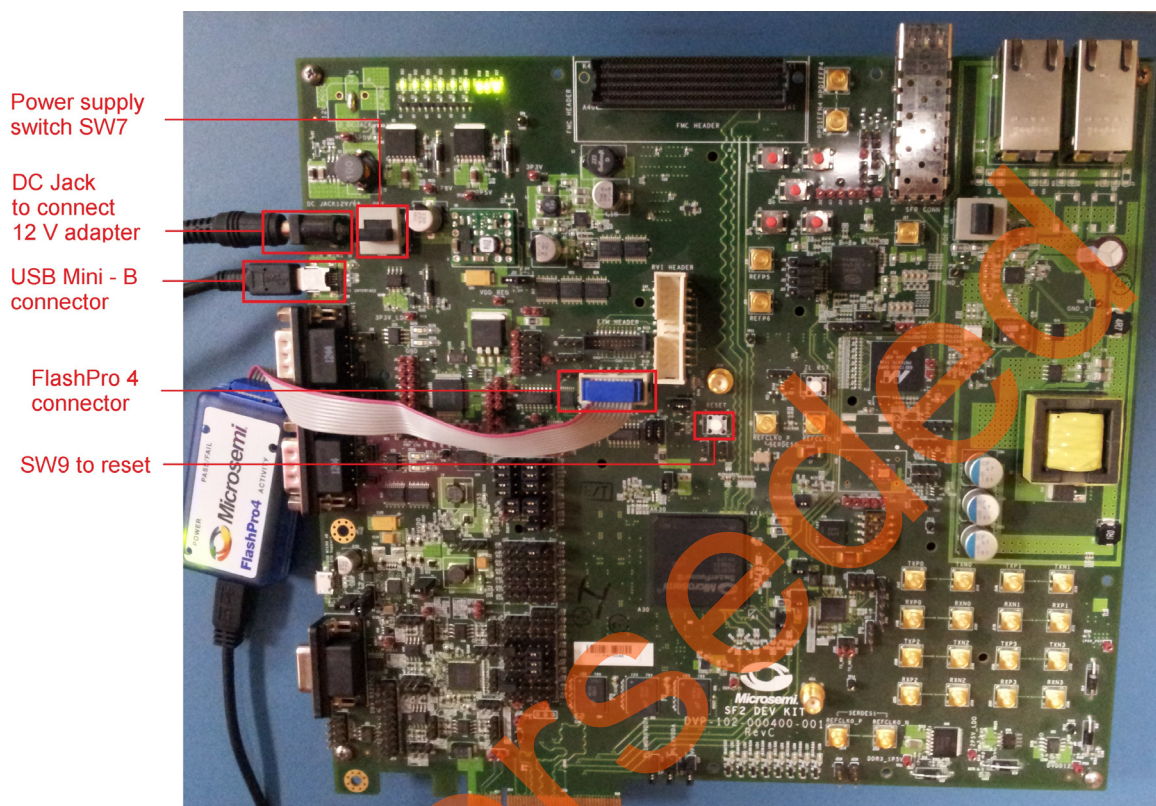


Figure 43 • SmartFusion2 Development Kit

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:

- 57600 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

NOTE: In Release mode (i.e. when the code is executed from eNVM0), Digest Check service will not work for eNVM0 option.

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_ES die (M2S050T revision B). 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 2012-2013 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: 5933 $
 * SVN $Date: 2013-11-04 15:00:37 +0530 (Mon, 04 Nov 2013) $
 */
#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_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_uart0;

/*=====
   Main function.
 */
```



```
int main()
{
    uint8_t serial_number[16];
    uint8_t user_code[4];
    uint8_t design_version[2];
    uint8_t device_certificate[512];
    uint8_t status;
    int8_t tx_complete;

    MSS_SYS_init(MSS_SYS_NO_EVENT_HANDLER);

    MSS_UART_init(gp_my_uart,
                  MSS_UART_57600_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;

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;

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.");
}
}
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);
    }
}
```

Superseded

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

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

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.

Superseded

Superseded



Microsemi®

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 and 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.

© 2014 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.