SmartFusion2 Controller Area Network (CAN) - Libero SoC v11.5

TU0559 Tutorial

Microsemi®

Table of Contents

SmartFusion2 Controller Area Network (CAN)

Introduction

SmartFusion®2 system-on-chip (SoC) field programmable gateway arrays (FPGAs) have an integrated controller area network (CAN) peripheral. The CAN controller is an advanced peripheral bus (APB_1) slave in the microcontroller subsystem (MSS) AHB bus matrix. Refer to the *SmartFusion2 Microcontroller Subsystem User's Guide* for more information. The ARM® Cortex™-M3 processor master or a master in the FPGA fabric configures the CAN controller using the APB slave interface in the SmartFusion2 SoC FPGAs.

The CAN controller in the SmartFusion2 device supports the concept of mailboxes. It is compliant to the CAN standard defined in the ISO 11898-1 standard. The CAN controller contains 32-bit receive buffers and 32-bit transmit buffers. Each buffer has its own message filter with prioritized arbitration scheme for optimal support of higher-layer protocols (HLP) such as DeviceNet. The message filter also covers the first two data bytes of the message payload. The transmit and receive message buffers are single error corrected, double error detected (SECDED) using the integrated error detection and correction (EDAC) controller. The functional behavior of the CAN peripheral must be defined at the application level using the SmartFusion2 MSS CAN firmware drivers provided by Microsemi®. Refer to the *CAN firmware drivers' user guide* in Libero® System-on-Chip (SoC) design software for more details. Figure 1 shows the CAN controller block diagram.

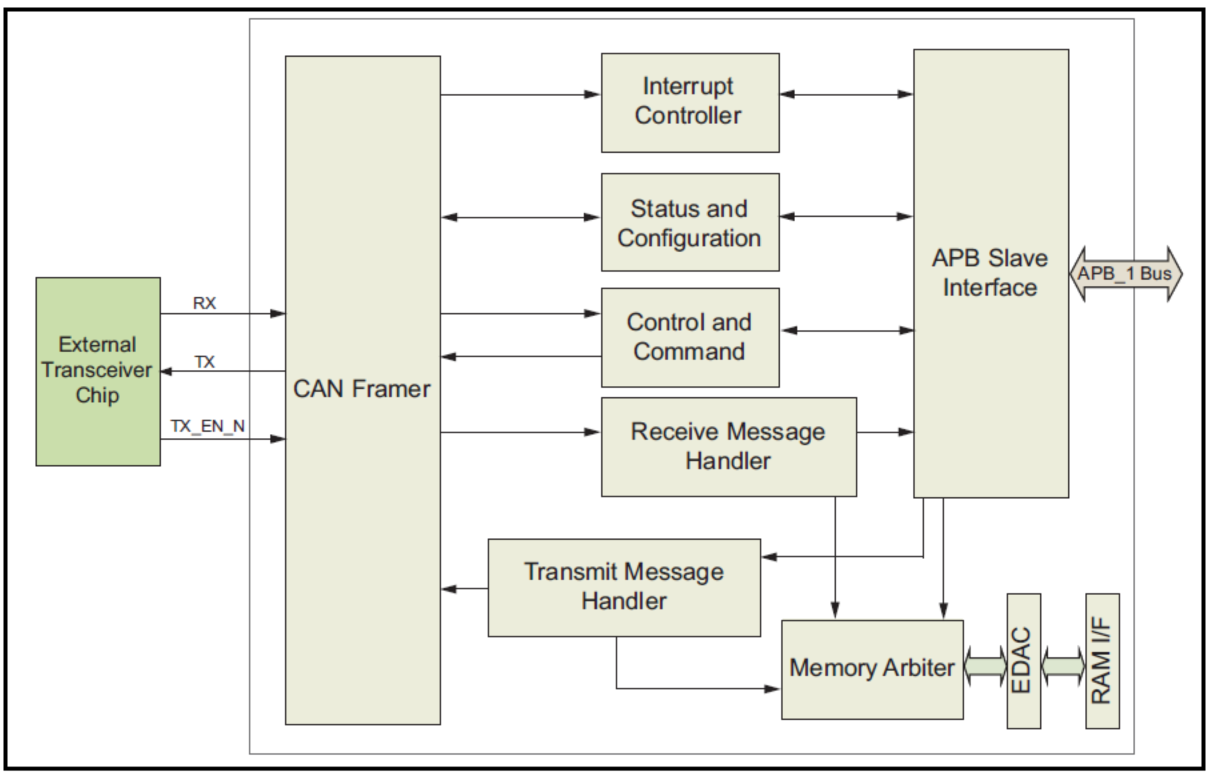

Figure 1 **• CAN Controller Block Diagram**

This tutorial design focuses on using the ARM Cortex-M3 processor as a master that configures the CAN controller. It uses the firmware driver functions and APIs that Microsemi provides to use different sets of CAN functions.

After completing this tutorial, you should be familiar with the following tasks:

1. Creating a Libero® SoC v11.5 project using the SmartFusion2 SoC FPGAs System Builder tools.
2. Configuring and generating various hardware blocks and clocking systems using the MSS.
3. Opening the project in SoftConsole and writing the application code.
4. Validating the application design on the SmartFusion2 Development Kit Board.

Tutorial Requirements

Table 1 lists the design requirements of CAN.

Table 1 • **Design Requirements**

Design Requirements	Description
Hardware Requirements	
PCAN-USB Adapter	Optionally recommended for evaluating some of the CAN features, *http://gridconnect.com/can-usb.html*
DB9 female-to-female adapter	To connect the PCAN-USB to the SmartFusion2 Development Kit Board
SmartFusion2 Development Kit	Rev C or later
Software Requirements	
Libero SoC	v11.5
FlashPro programming software	v11.5
PCAN-View software	Download from GridConnect, *http://gridconnect.com/can-usb.html*
SoftConsole	v3.4SP1
One of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	-

Associated Project Files

Extract the

www.microsemi.com/soc/download/rsc/?f=m2s_tu0559_can_liberov11p5_df

Libero SoC project along with the ReadMe and programming (.stp) file to a folder on the HDD of your PC (For example, *C:\Microsemiprj*).

Design Overview

The CAN controller tutorial design describes the usage of MSS CAN drivers and APIs to use different SmartFusion2 CAN features. The design is created using the System Builder with the following configurations:

- The M3_CLK and the MSS Main clock are configured to generate a 32 MHz clock, which is generated using the MSS clock conditioning circuit (MSS_CCC). The 32 MHz is used for the demonstration purposes.
- The CLK_BASE of the MSS_CCC is sourced from the fabric CCC (FCCC). The FCCC is sourced from the 25/50 MHz RC Oscillator.
- The MSS is configured to use a UART peripheral instance (MMUART_1). The MMUART_1 is used as an interface for reading and writing the messages from and to the HyperTerminal, and is clocked by APB_1_CLK on the APB bus1 (APB_1). The APB_1_CLK is derived from M3_CLK.
- The CAN peripheral instance is enabled.

Figure 2 shows the CAN top-level block diagram.

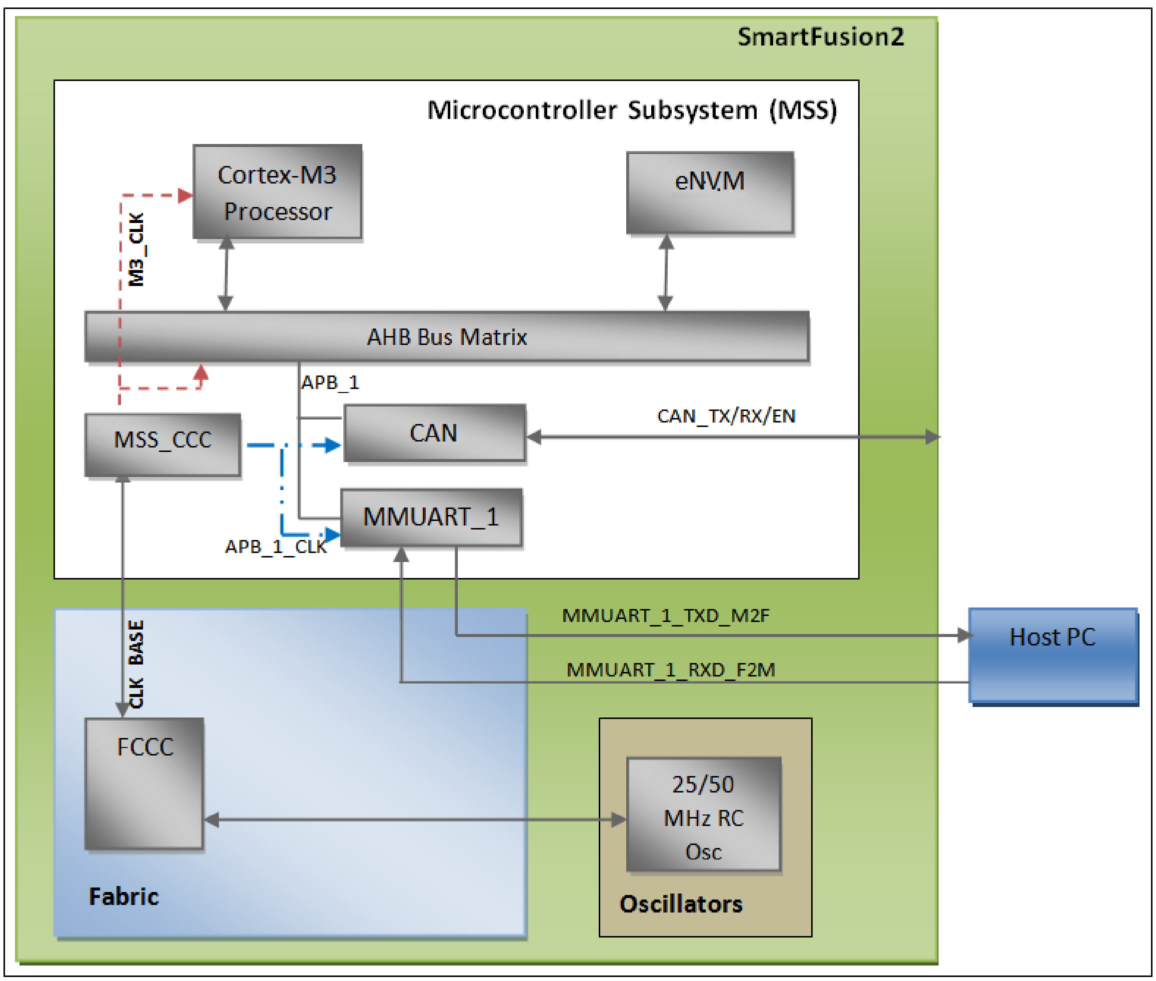

***Figure 2* • CAN Top-Level Block Diagram**

Design Creation

Step 1: Creating a Libero SoC Project

1. Launch Libero SoC v11.5.
2. From the **Project** menu, select **New Project**. In the **Project Details** window, enter the information displayed in Figure 3.
   – **Project Name:** CAN_SmartFusion2_Tutorial
   – **Project Location:** Navigate to an appropriate location to save the new project (for example, C:/Microsemi_prj)
   – **Preferred HDL Type:** Leave as Verilog
   – **Enable Block Creation:** Unchecked

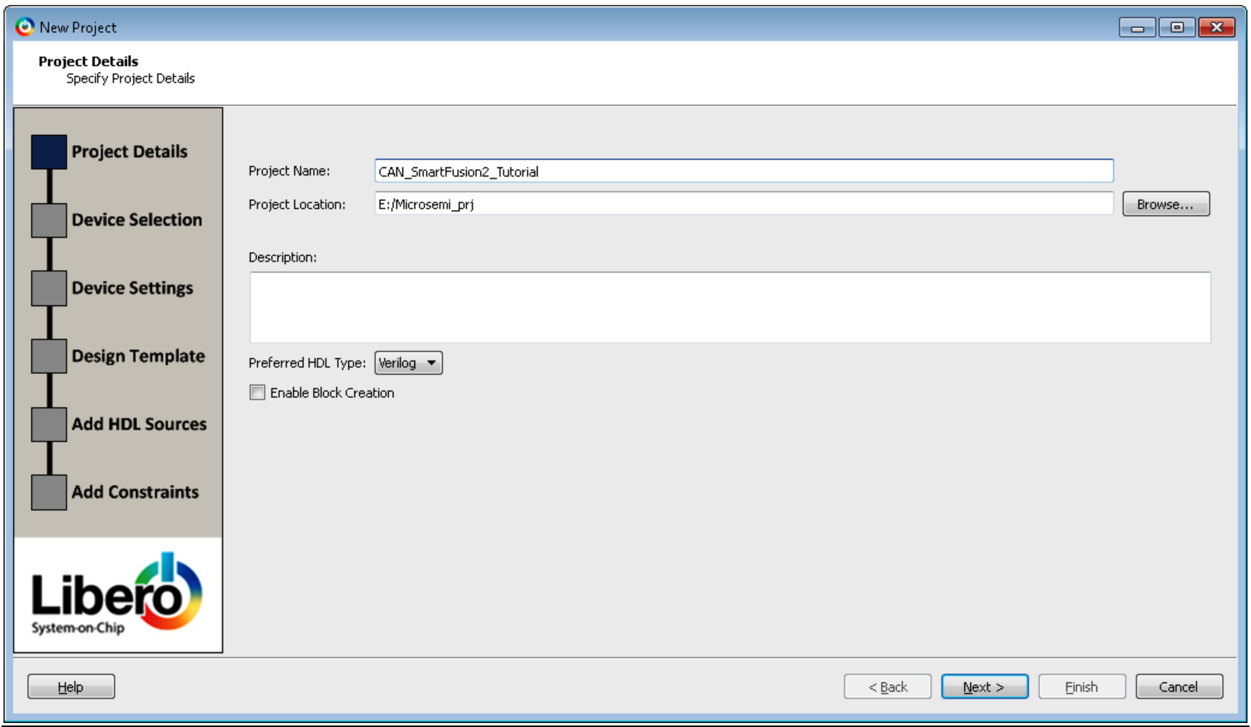

***Figure 3* • New Project Details Window**

3. Click Next. In the Device Selection window, select the information displayed in Figure 4 on page 7.
   – **Family:** SmartFusion2
   – **Die:** M2S050T
   – **Package:** 896 FBGA
   – **Part Number:** M2S050T-FG896

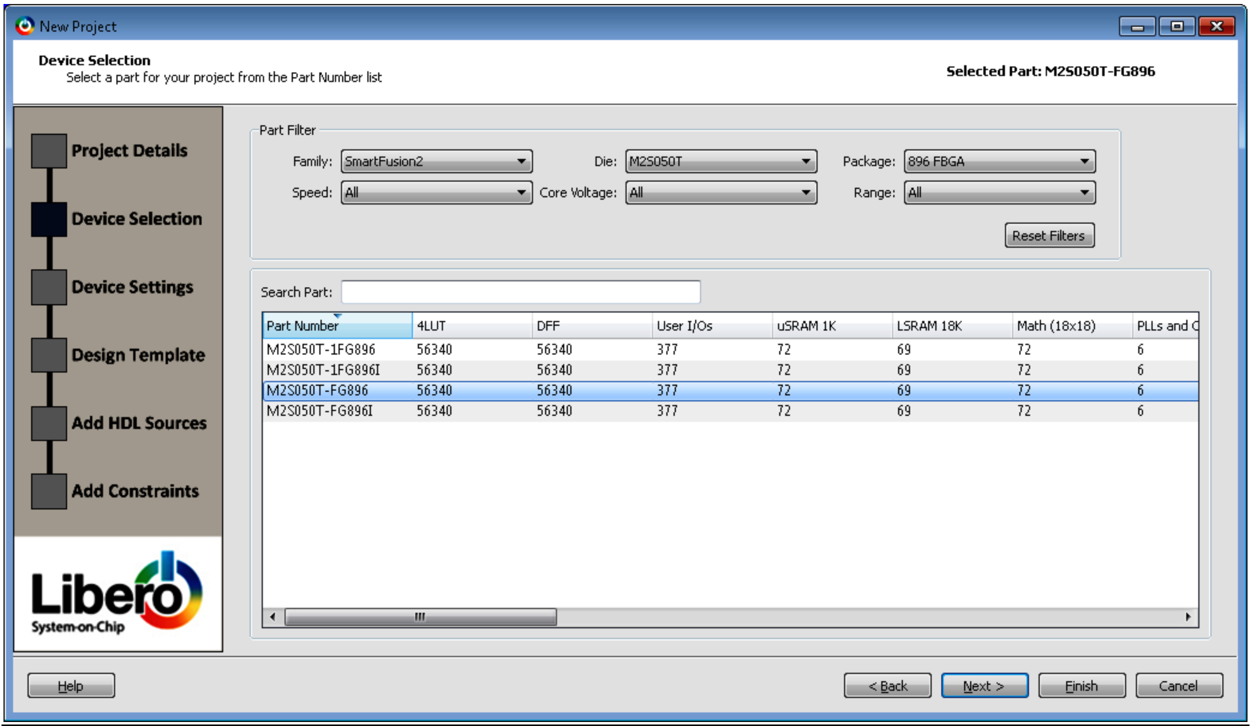

***Figure 4* • Device Selection Window**

4. Click **Next**. In the **Device Settings** window, select the information displayed in Figure 5 on page 8.
   – **Default I/O Technology:** LVSMO2.5v
   – **PLL Supply Voltage (V):** 2.5
   – **Maximum Core Voltage Rail Ramp Up Time:** 100ms Minimum
   – **System Controller Suspend Mode:** Unchecked

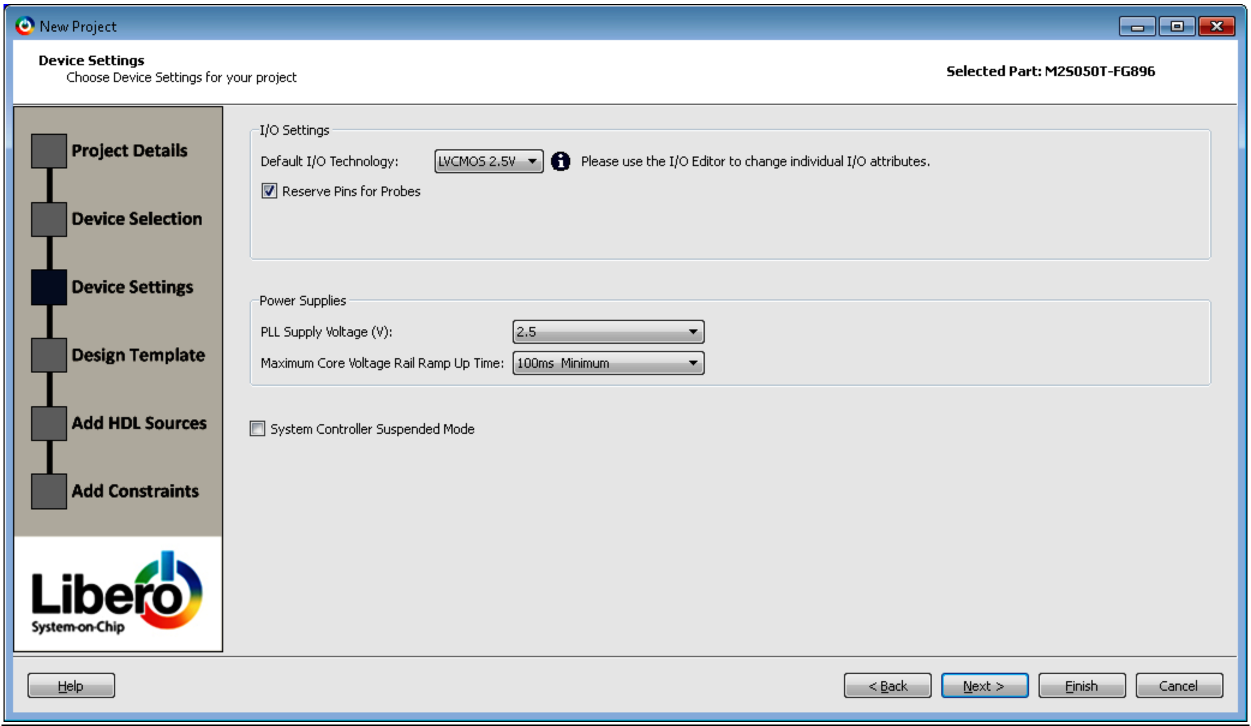

Figure 5 **• Device Settings Window**

5. Click **Next**. In the **Design Template** window, select **Create a System Builder base design** under **Design Templates and Creators** as shown in Figure 6.

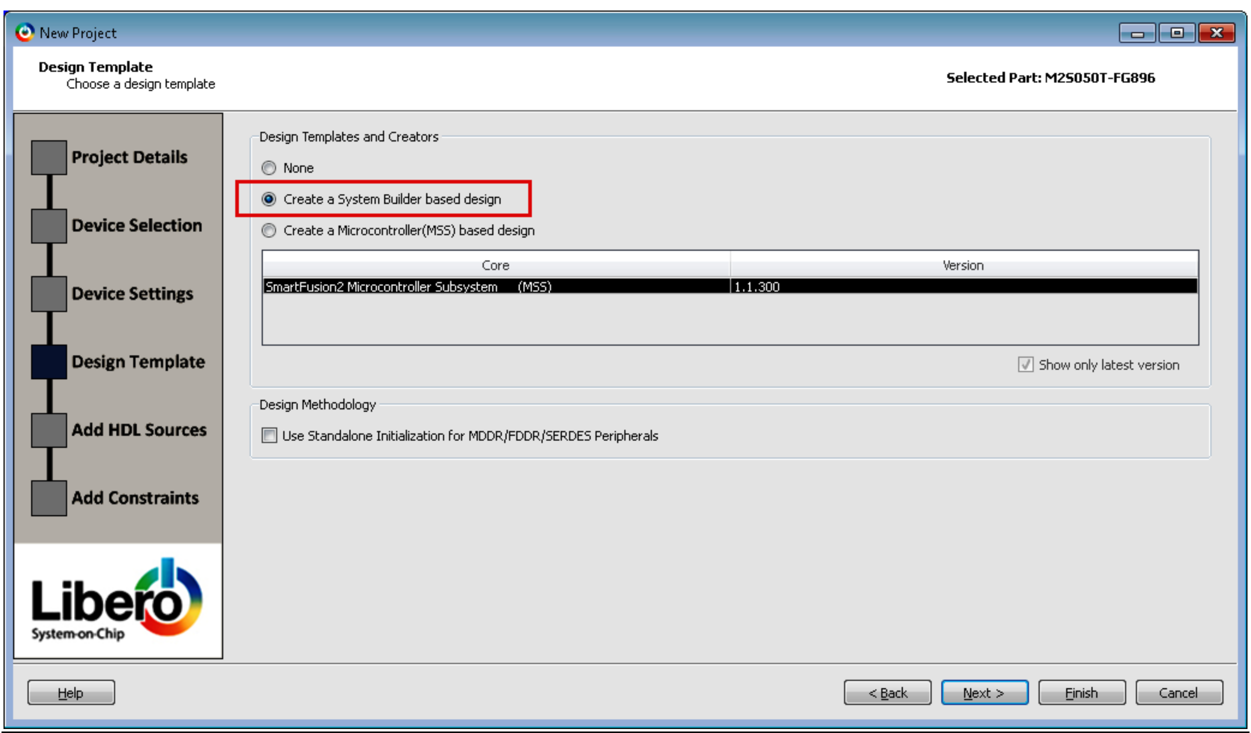

Figure 6 **• Device Template Window**

6. Click **Finish.**
7. Enter **CAN_SB** as name in the **System Builder** dialog box as shown in Figure 7.

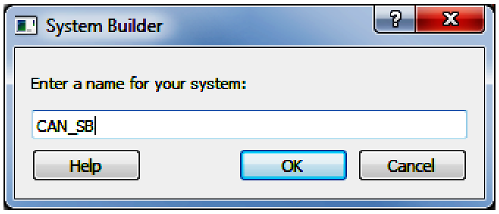

***Figure 7* • Create New System Builder**

8. Click **OK**.

The **System Builder- Device Features** window is displayed as shown in Figure 8.

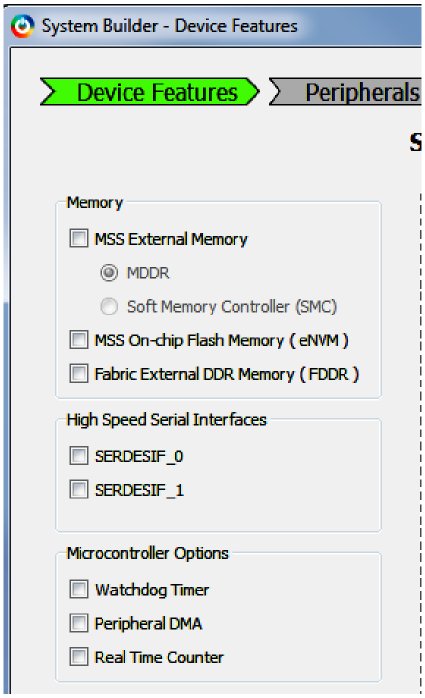

***Figure 8* • System Builder - Device Features Window**

9. Click **Next**. The **System Builder - Peripherals** window is displayed.

10. Select the **MSS_CAN** and **MM_UART_1** MSS peripheral check boxes and clear all the other MSS peripheral check boxes as shown in Figure 9.

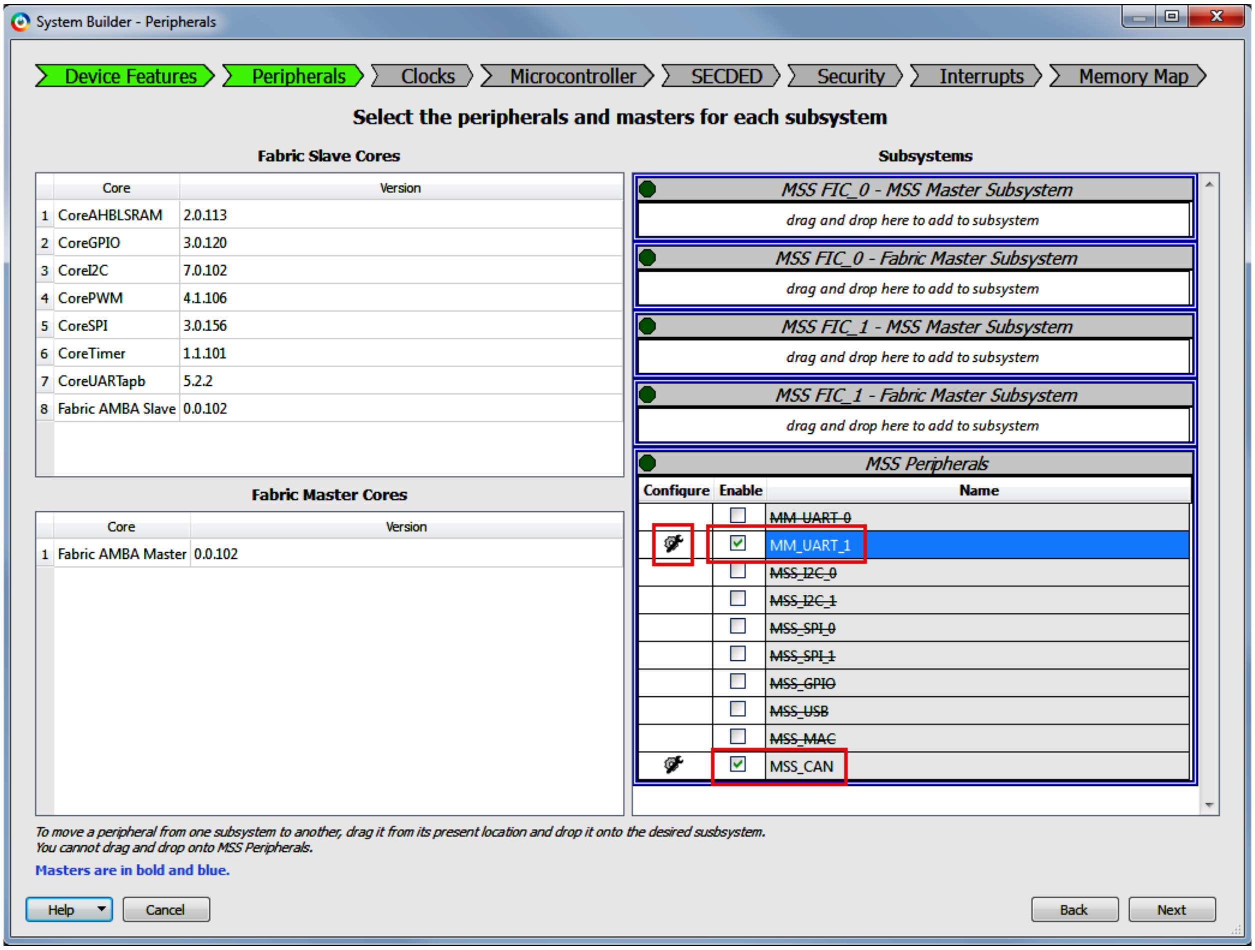

Figure 9 **• System Builder - Peripherals Window**

11. Double-click the settings icon next to **MM_UART_1** peripheral. Figure 9 highlights the MM_UART_1 and MSS_CAN peripherals and the settings icon. The **Configuring MM_UART_1_0** window is displayed.

12. From the MMUART configuration window, select **Fabric** from the **Connect To** drop down list under **Configuration** and leave the other settings default as shown in Figure 10. The MMUART signals are routed through the fabric and connected to the R29 and R24 package pins. The R29 and R24 pins are connected to the FTDI or mini-USB interface using the J129 and J133 jumpers.

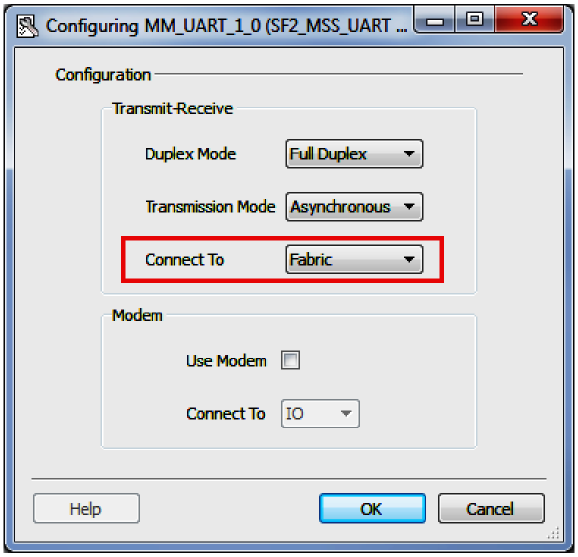

***Figure 10* • MMUART Configuration - Fabric Connection**

13. Click **OK**.
14. Click **Next**. The **System Builder - Clock Settings** window is displayed as shown in Figure 11 on page 12.
15. Select the following options:
    – **System Clock**: set to **50 MHz** and select **On-chip 25/50 MHz RC Oscillator** from the drop down list. The System Builder automatically instantiates the oscillator and configures it.
    – **M3_CLK**: set to **32 MHz**. This is derived from the System Clock.
    – **APB_1_CLK**: set to **32 MHz**. The CAN clock is the APB_1_CLK, which is derived from M3_CLK.

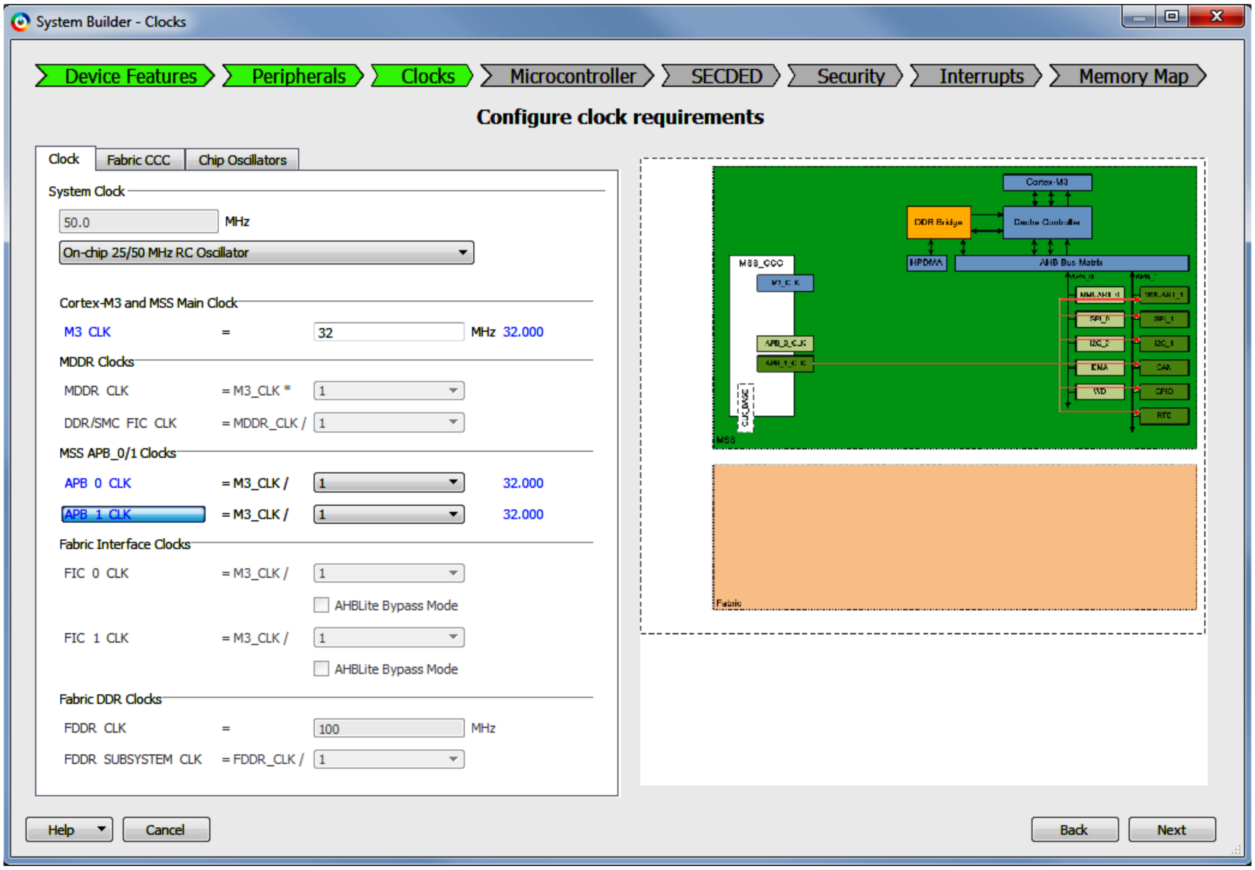

***Figure 11* • System Builder - Clocks Window**

Note: By clicking the blue clock name (For example, APB_1_CLK), the clock and the different blocks that the clock drives are shown.

16. Click **Next**. The **System Builder - Microcontroller Options** window is displayed.

Note: From steps 16 to 18, keep the default settings.

17. Click **Next**. The **System Builder - SECDED Options** window is displayed.
18. Click **Next**. The **System Builder - Security Options** window is displayed.
19. Click **Next**. The **System Builder - Interrupts Options** window is displayed.
20. Click **Next**. The **System Builder - Memory Map Options** window is displayed.

21. Click **Finish**. The System Builder generates the system based on the selected options. The System Builder block is created and added to the Libero SoC project as shown in Figure 12.

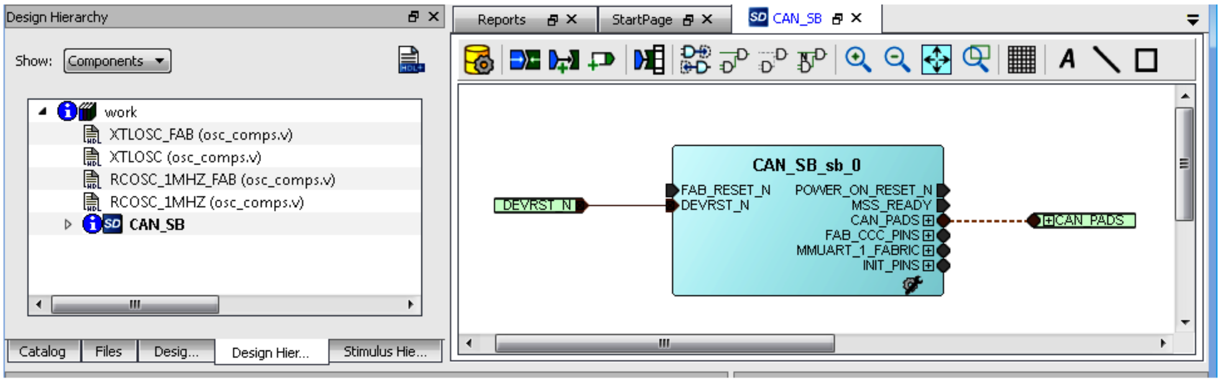

***Figure 12* • System Builder Top-Level Block Diagram**

22. Connect the pins as follows:
    - Right-click **FAB_RESET_N** and select **Tie High**. This is an active low reset input that comes from the user logic to fabric. As this signal is not used in this tutorial, set it High.
    - Right-click **POWER_ON_RESET_N** and select **Mark Unused**.
    - Right-click **MSS_READY** and select **Mark Unused**.
    - Right-click **FAB_CCC_GL0**, part of **FAB_CCC_PINS** and select **Mark Unused**.
    - Right-click **INIT_DONE**, part of **INIT_PINS** and select **Mark Unused**.
    - Right-click **MMUART_1_FABRIC** and select **Promote to Top Level**.

After making all the connections, the System Builder block displays as shown in Figure 13.

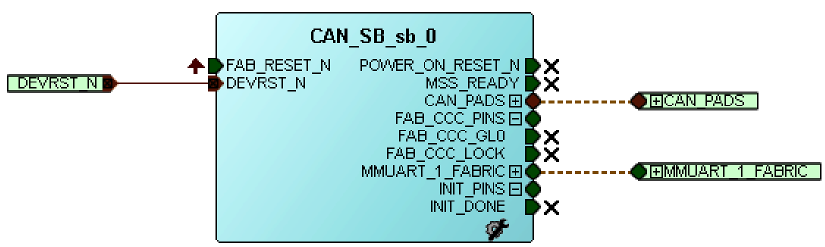

***Figure 13* • System Builder Connections**

23. Choose **Generate Component** from the **SmartDesign** menu to generate the final system. Or click (**Generate Component**). The System Builder generates the system based on the selected options. After successful generation of the system, the message "Info: 'CAN_SB' was successfully generated" is displayed in the log window.

Step 2: Generating the Programming File

1. Click the **Design Flow** tab and double-click the **I/O Constraints** in the **Design Flow** window as shown in Figure 14. The **I/O Editor** window is displayed when Synthesize and Compile stages are complete.

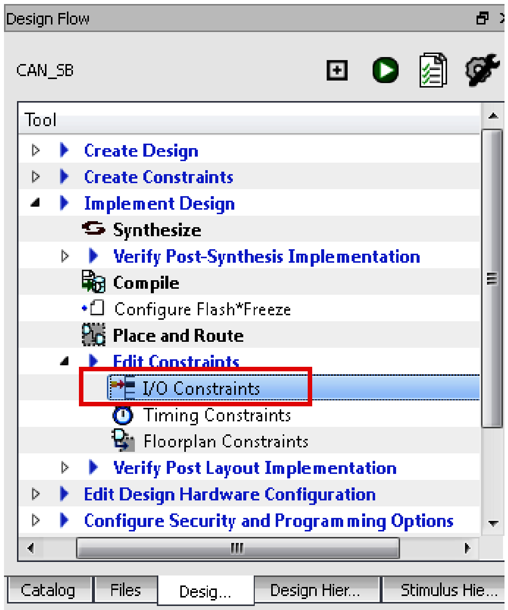

Figure 14 • **I/O Constraints Editor**

2. Assign the MMUART pins as shown in Table 2.

Table 2 • **Pin Names and Pin Numbers**

Pin Names	Pin Numbers
MMUART_1_RXD_F2M	R29
MMUART_1_TXD_M2F	R24

The **I/O Editor - CAN_SmartFusion2_Tutorial** is displayed as shown in Figure 15.

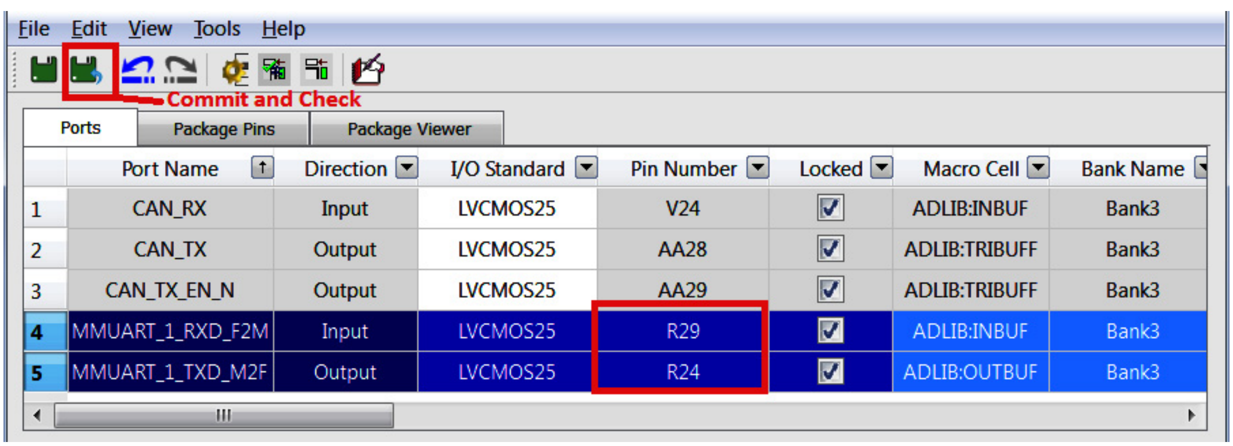

Figure 15 • **Launch I/O Constraints Editor**

3. Click the **Commit and Check** icon after updating the I/O editor. The I/O PDC constraint file is generated automatically and used for compile stage. The message "Info: Generated IOPDC file 'E:/Microsemi_prj/CAN_SmartFusion2_Tutorial/constraint/io/CAN_SmartFusion2_Tutorial.io.pdc'; marked as Use for Compile" is displayed in the **I/O Editor Log** window.
4. Close the **I/O Editor** window.
5. Click the **Generate Bitstream** icon to complete the Place and Route process and generate the programming file, as shown in Figure 16.

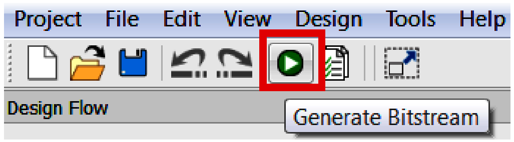

***Figure 16* • Generate Programming Data**

Step 3: Programming the Device

Before proceeding with programming the device, ensure that FlashPro4 programmer is properly connected to the Flash Pro Header J59 connector of the SmartFusion2 Development Kit board. Use the following to ensure the correct jumper settings:

Board Jumper Settings

1. Connect the Jumpers on the SmartFusion2 Development Kit Board as described in Table 3. When connecting the jumper setting, switch OFF the power supply on the board, SW7.

***Table 3* • Jumper Settings**

Jumper	Connection	Descriptions
J129	2-3	Connects the MMUART1 RXD to the FTDI Interface
J133	2-3	Connects the MMUART1 TXD to the FTDI Interface
CAN BUS		
J114	1-2	CANTXBUS1
J111	1-2	CANRXBUS1
J115	1-2	CANTXEBL1
J36	Use jumper	-

Note: Refer to the *SmartFusion2 Development Kit User Guide* for more information on Board Jumper Settings.

2. Click the **Design Flow** tab and double-click the **Run PROGRAM Action** under **Program Design** in the **Design Flow** window to program the SmartFusion2 SoC device as shown in Figure 17.

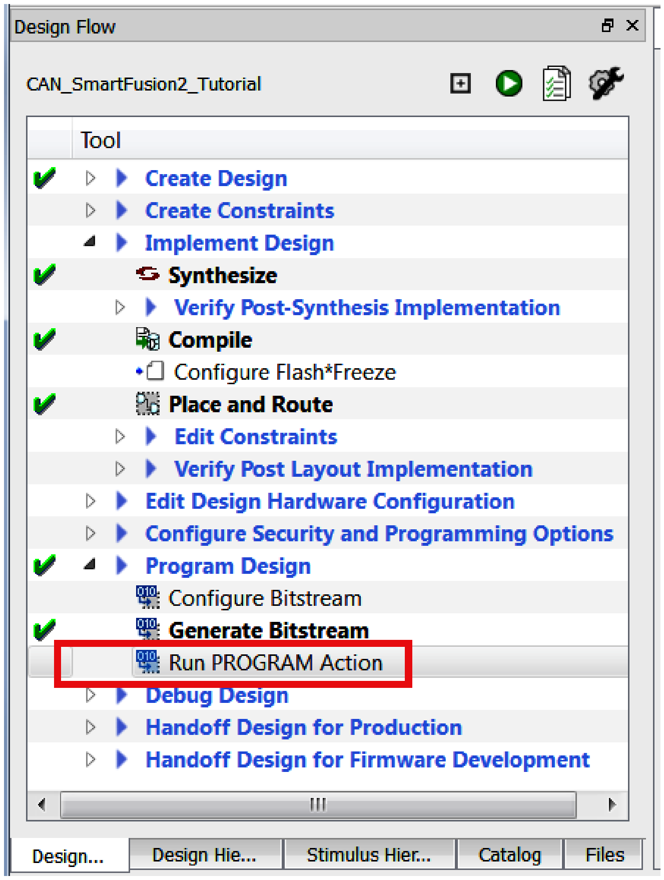

Figure 17 **• Program the Device**

Step 4: Building the Software Application using SoftConsole

From Libero SoC, configure and export the Firmware cores used in the project.

To export the SoftConsole project,

1. Double-click **Configure Firmware Cores** under **Handoff Design for Firmware Development** as shown in Figure 18. Inspect the design's firmware core versions to confirm the latest versions of the core drivers are selected.
2. Double-click **Export Firmware** under **Handoff Design for Firmware Development** to export the Firmware files used in the project as shown in Figure 18

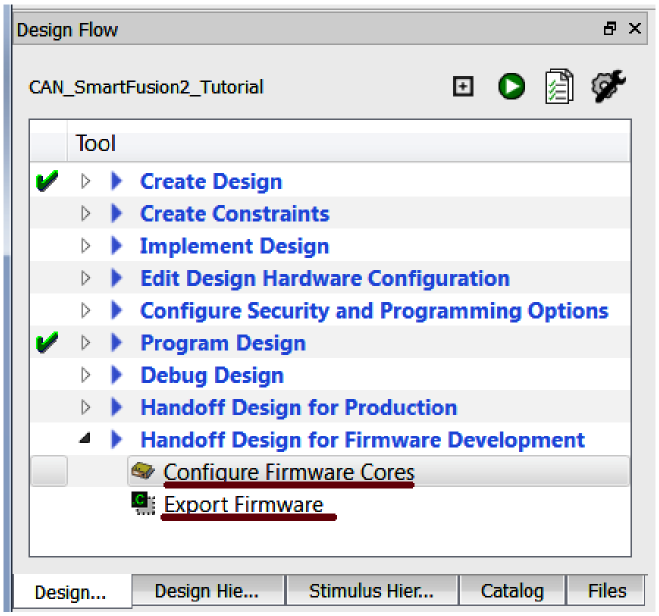

***Figure 18* • Configure and Export Firmware**

3. Select the **Create project** check box and select **SoftConsole3.4** from the drop-down list as shown in Figure 19. By default, the firmware and SoftConsole project files are created in the same directory where the Libero project is created.
4. Browse and change the location as required.

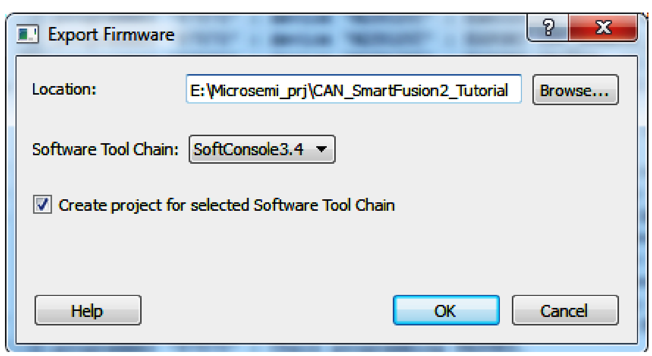

***Figure 19* • Export Firmware**

5. Click **OK**. The Firmware and SoftConsole folders are created at the selected location. The folders contain the required file sets for SoftConsole IDE. After exporting the Firmware and SoftConsole successfully, the message "Info: Firmware project was successfully exported to 'E:\Microsemi_prj\CAN_SmartFusion2_Tutorial\firmware'." is displayed.
6. Launch the **SoftConsole v3.4SP1**.
7. Browse and select the Workspace of Softconsole folder where the Libero project is located, as shown in Figure 20.

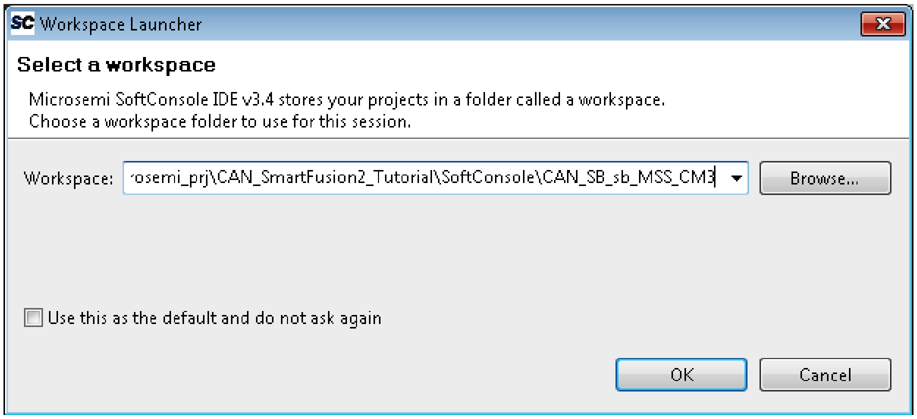

Figure 20 **• Specify the SoftConsole Workspace Location**

8. Click **OK**.

Note: The specified SoftConsole Workspace should be the path where the SoftConsole folder is created. The **SoftConsole** window is displayed with the application and HW projects loaded automatically.

9. Go to the source folder in the downloaded design files folder and copy the code from the **source_main.c** file.
10. Click the **Project Explorer** tab on the left pane and double-click the **CAN_SB_sb_MSS_CM3_app** folder in SoftConsole.
11. Double-click the **main.c** file in the left pane. The `main.c` file is opened on the right pane.

12. Delete the existing code and paste the code from `source_main.c` in the main.c file as shown in Figure 21.

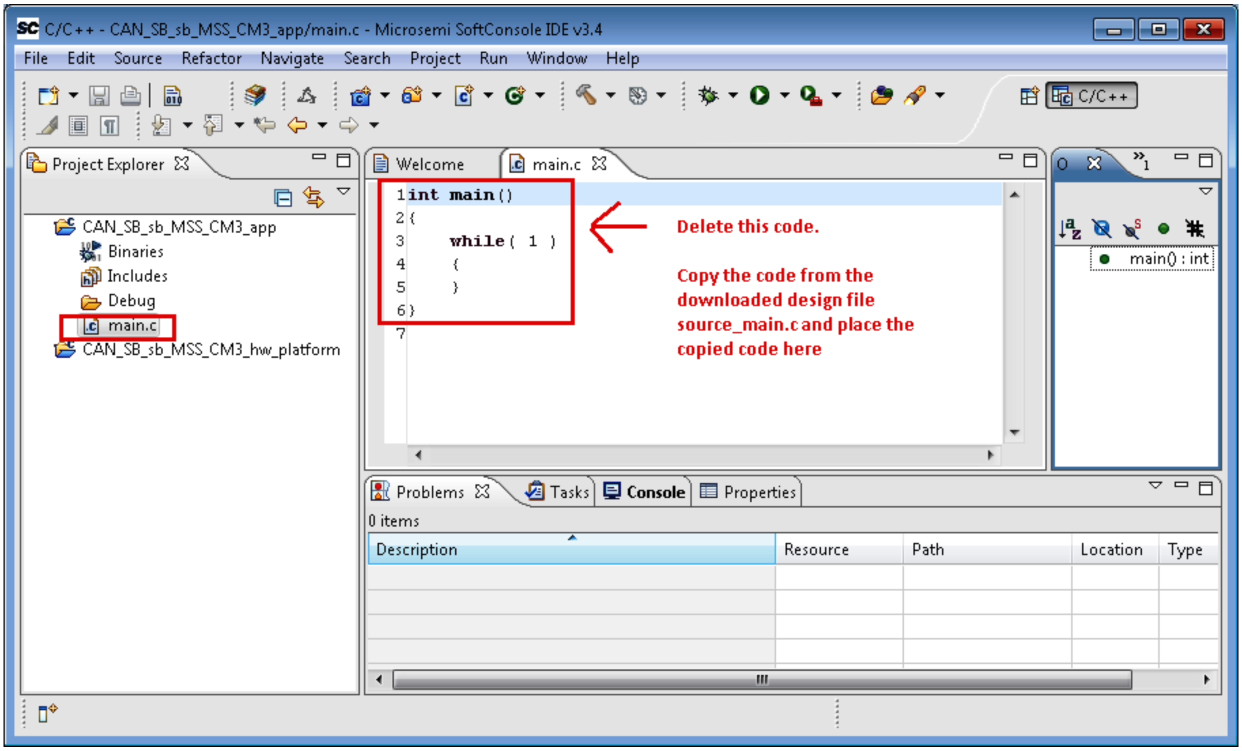

Figure 21 **• Update the main.c with the Provided Code**

13. Choose **Project** > **Clean** to perform a clean build as shown in Figure 22.

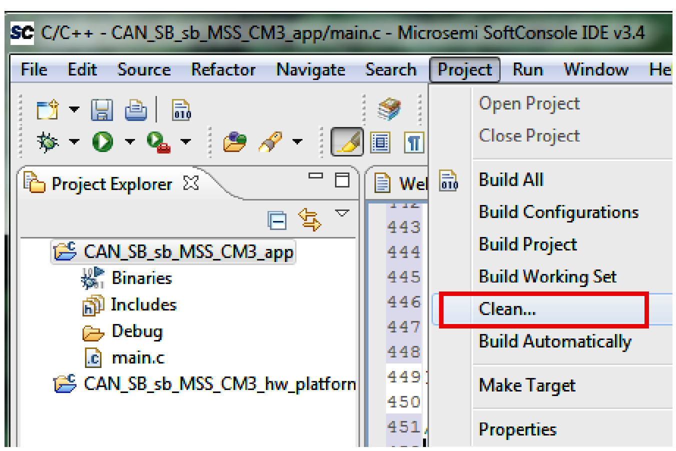

Figure 22 **• Clean Project Build**

14. Accept the default settings in the **Clean** window, and click **OK** as shown in Figure 23.

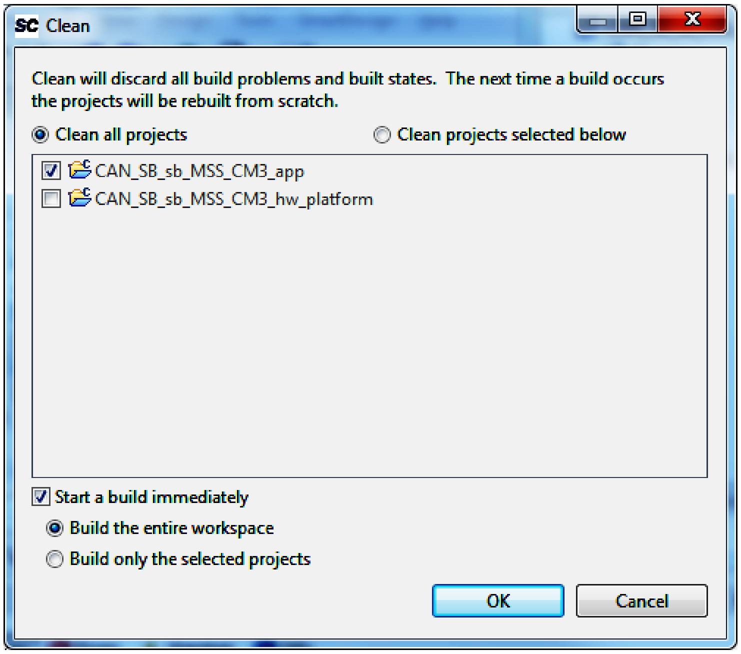

***Figure 23* • Clean and Build Window**

Note: Ensure that no errors are displayed throughout the design configuration and build flow.

Step 5: Configuring the Serial Terminal Emulation Program

Before running the application program, configure the terminal emulator program (HyperTerminal) in the system. Follow the below steps to use the SmartFusion2 Development Kit Board:

1. Connect one end of the USB mini-B (FTDI interface) cable to the J24 connector provided on the SmartFusion2 Development Kit Board.
2. Connect the other end of the USB cable to the host PC.
3. Ensure that the USB to UART bridge drivers are automatically detected.
4. Download and install the drivers from *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip* if the USB to UART bridge drivers are not installed.
5. Start a HyperTerminal with the baud rate set as **57600**, **8 data bits**, **1 stop bit**, **no parity**, and **no flow control**.

Refer to the *Configuring Serial Terminal Emulation Programs Tutorial* for configuring HyperTerminal, Tera Term, and PuTTY.

Step 6: Setting and Using an External CAN Transceiver

An external transceiver is used to generate and receive CAN transactions, and is also used to demonstrate the sent and received features of the SmartFusion2 CAN controller. The CAN USB Adapter (PCAN-USB) from GridConnect is a hardware that is used in this tutorial shown in Figure 24. The PCAN-View software is also used for user interface along with the PCAN-USB hardware.

Refer to the *GridConnect* site for more information.

***Figure 24* • PCAN-USB Adapter**

To connect the PCAN USB Adapter,

1. Connect the PCAN-USB adapter to the SmartFusion2 CAN header DB9-CAN1 (J42) on the SmartFusion2 Development Kit Board using a DB9 female-to-female adapter.
2. Connect the other USB end cable to the host PC.
3. Ensure that the drivers are automatically detected.
4. Download and install the drivers from *http://gridconnect.com/pcan/can-adapters/can-usb.html* if PCAN-USB drivers are not installed.
5. Download the PCAN-View software executable from *http://gridconnect.com/media/documentation/peak_system/pcanview.zip*

6. Launch the **PCAN-View** software. The **PCAN-View** window is displayed as shown in Figure 25.

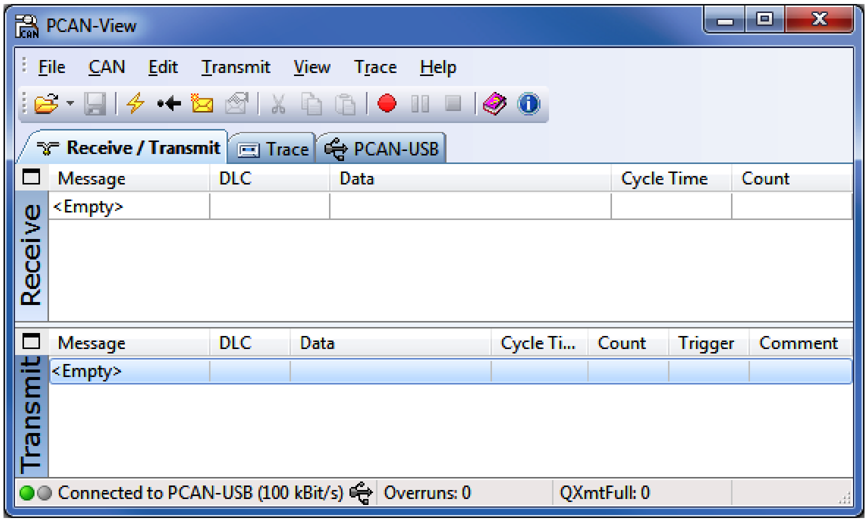

***Figure 25* • PCAN-View**

7. Choose **Connect** from the **CAN** menu to connect to the PCAN-USB adapter. The **Connect** window is displayed as shown in Figure 26.

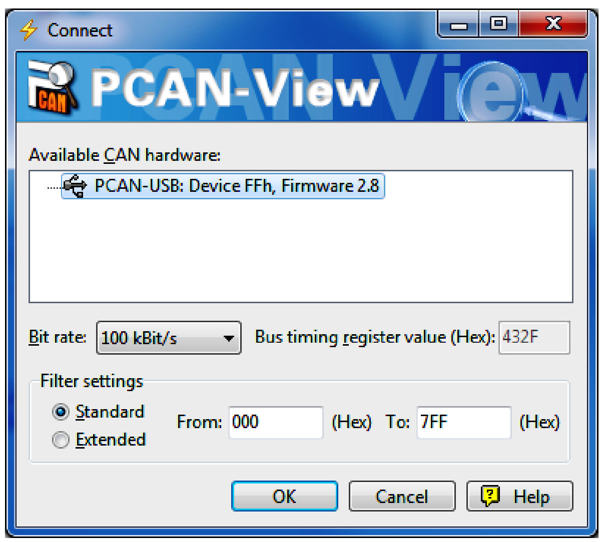

***Figure 26* • Connect to the CAN Hardware**

The PCAN-USB device is displayed under Available CAN hardware if PCAN-USB adapter drivers are installed correctly.

8. Select the **Bit rate** as **100 kBit/s** as shown in Figure 26.
9. Click **OK**.

Step 7: Debugging and Running Application Project using SoftConsole

Follow the steps below to debug and run application project using SoftConsole:

1. Select the **CAN_SB_sb_MSS_CM3_app** in Project Explorer of SoftConsole.
2. Choose **Run** > **Debug Configurations** in the **SoftConsole** window. The **Debug Configurations** window is displayed.
3. Double-click the **Microsemi Cortex-M3 Target** and click **CAN_SB_sb_MSS_CM3_app Debug** as shown in Figure 27.

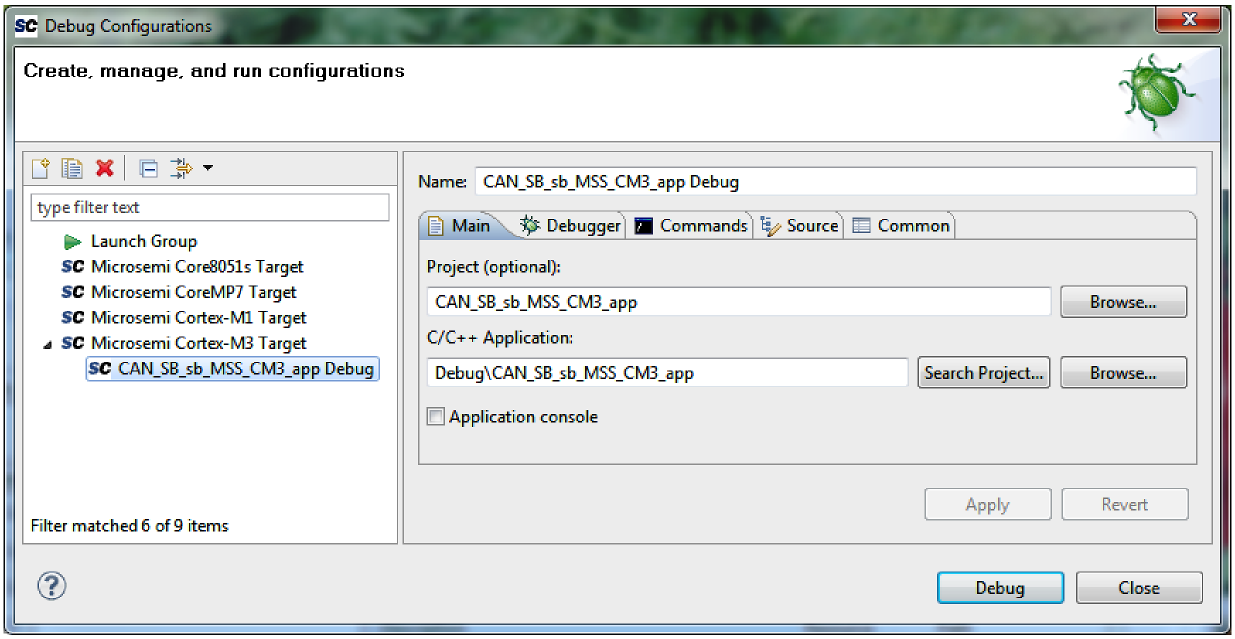

Figure 27 • **Debug Configurations**

4. Ensure that the following information appear in the **Main** tab of **Debug Configurations** window:
   – **Name**: CAN_SB_sb_MSS_CM3_app Debug
   – **Project (optional)**: CAN_SB_sb_MSS_CM3_app
   – **C/C++ Application**: Debug\CAN_SB_sb_MSS_CM3_app
5. Click **Apply** if active, and then click **Debug**.
6. Click **Yes**, when prompted for Confirm Perspective Switch as shown in Figure 28

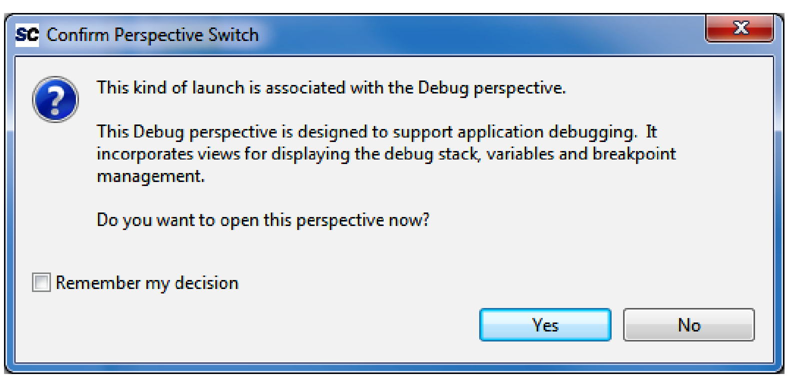

Figure 28 • **Confirm Perspective Switch**

The Debug view mode is displayed as shown in Figure 29.

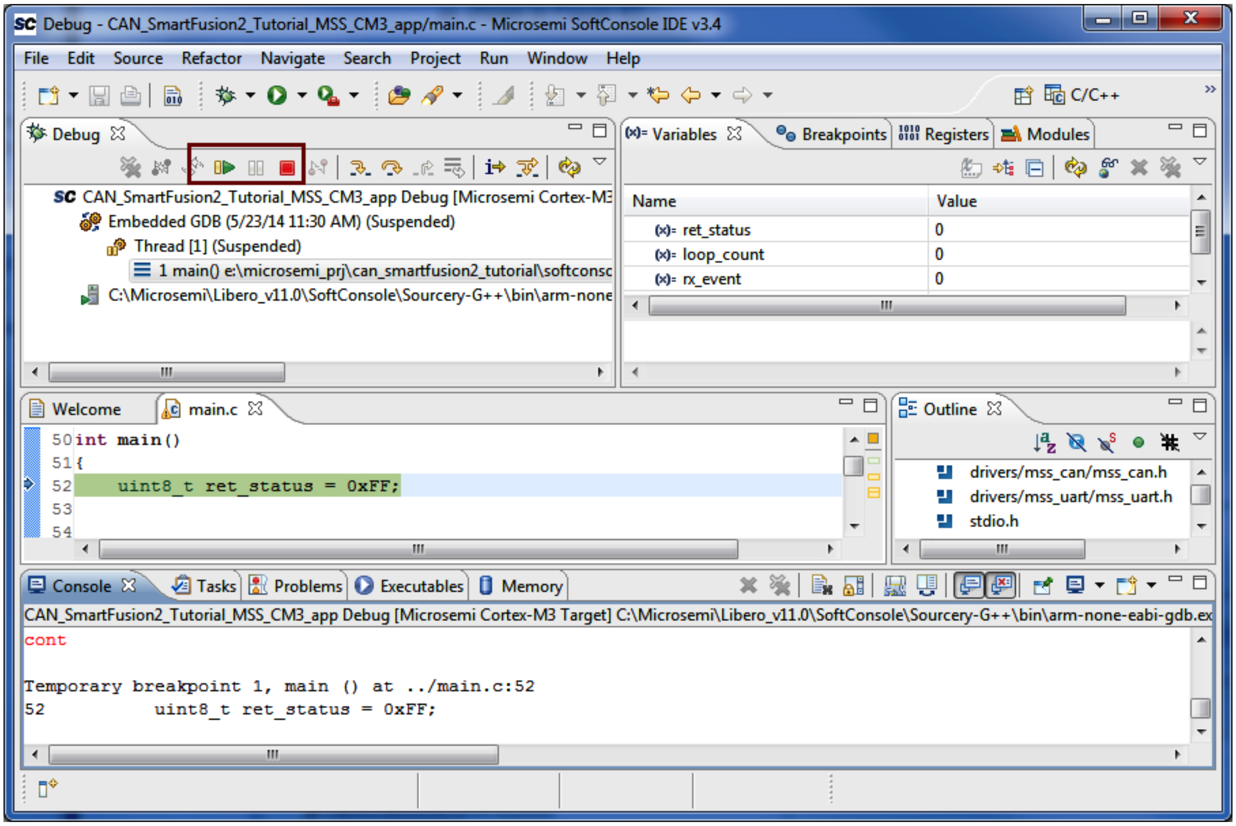

***Figure 29* • Debug Perspective**

7. Choose **Run** > **Resume** to run the application or click **Run** on the SoftConsole toolbar as shown in Figure 29.

The application options are displayed in the terminal program window as shown in Figure 30.

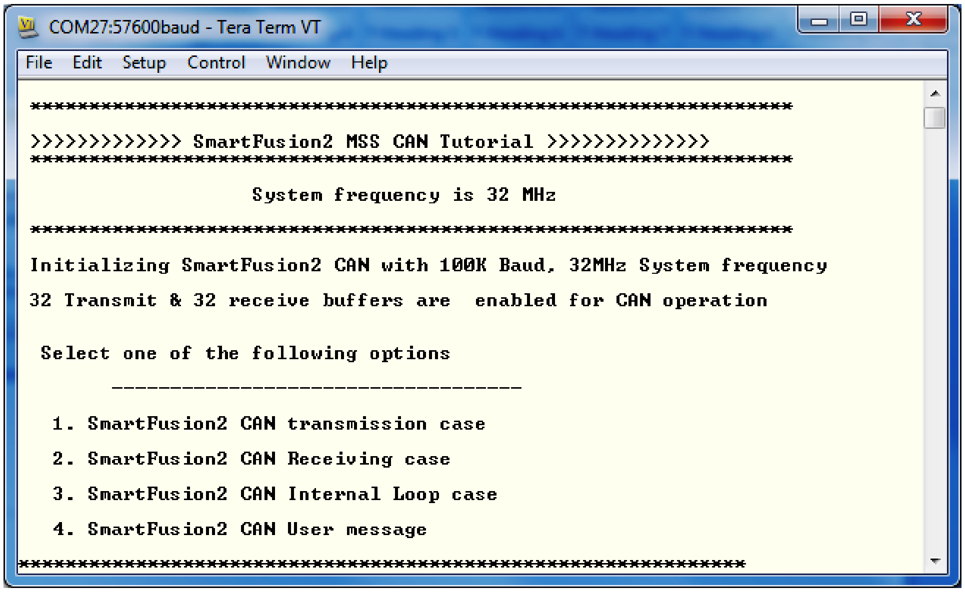

***Figure 30* • Greeting Messages**

Table 4 summarizes the HyperTerminal options and its usages.

***Table 4* • Options and Description**

Options	Description
1	When selected, the predefined message is sent to the CAN bus, and the same is reflected in CAN Analyzer.
2	When selected, the message that is received on the CAN bus is displayed in HyperTerminal. The message(s) are sent by the CAN Analyzer are displayed in HyperTerminal.
3	When selected, the message that is sent from CAN controller is looped back internally and reflected in HyperTerminal. It demonstrates the internal loop feature of the CAN controller.
4	When selected, the user enters the message to send to the CAN bus from HyperTerminal. The message is sent to the CAN bus, and the same message is reflected in CAN Analyzer.

Option 1: Transmitting a Message

The MSS CAN driver must be initialized and the mode of the operation must also be selected before performing the data transfers on the CAN bus.

The following functions are used to transmit the messages:

- **MSS_CAN_init()**: initializes the CAN controller and driver
- **MSS_CAN_set_mode()**: selects the operating mode of CAN
- **MSS_CAN_start()**: starts the actual data transfers
- **MSS_CAN_stop()**: stops the CAN controller
- **MSS_CAN_set_config_reg()**: changes the MSS CAN configuration after initializing the normal mode operation

A sample message configuration is displayed below.

```
/* configure a message */
pMsg.ID=0x120;
pMsg.DATALOW = 0x1A2B3C4D;
pMsg.DATAHIGH = 0xF5E6C7D9;
pMsg.L =((1<<20)| 0x00080000);
```

Refer to the *mss_can.h generated file* for more information on the CAN message structure.

- To set the mode of the CAN bus, the MSS_CAN_set_mode() function is used as follows:
  – MSS_CAN_set_mode(&g_can0, CANOP_MODE_NORMAL);
- To send a message to the CAN bus, the MSS_CAN_send() function is used as follows:
  – MSS_CAN_send_message(&g_can0, &pMsg);

Refer to the **SmartFusion2 MSS CAN Driver User Guide** for more information, which can be accessed from the Libero SoC Firmware Configuration option.

- Enter **1**. The message is sent to the CAN bus and the same message is reflected in the HyperTerminal as shown in Figure 31.

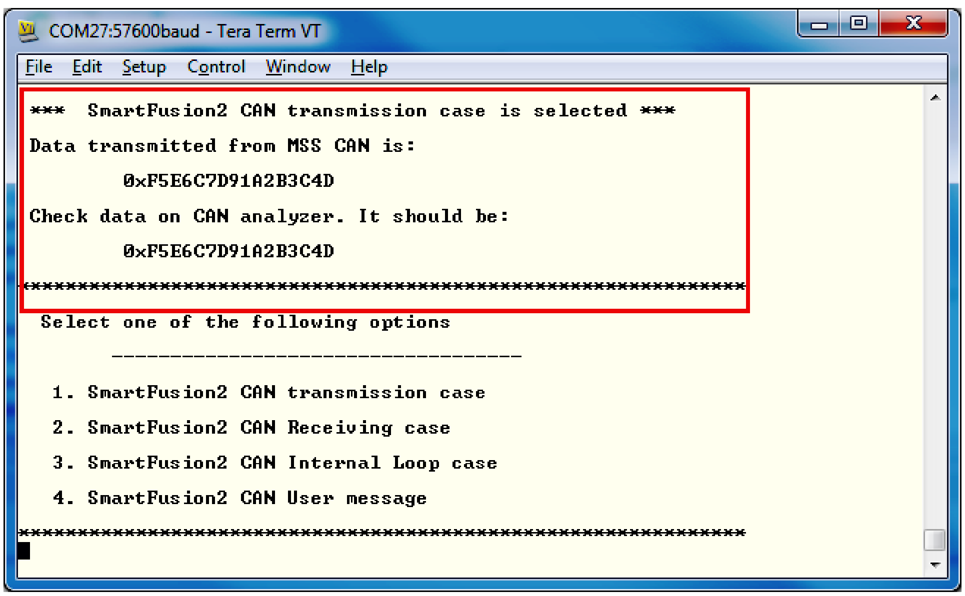

Figure 31 **• SmartFusion2 CAN Transmission Case Option**

The same message is also displayed in the PCAN-View window as shown in Figure 32.

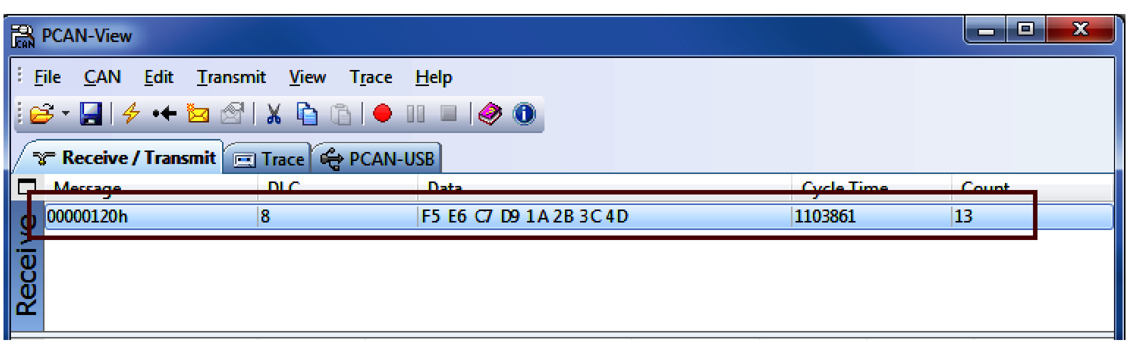

Figure 32 **• PCAN-View Transmitted Message**

Option 2: Receive a Message

1. Choose **Transmit** > **New Message** to enter a new transmit message using the PCAN-View window as shown in Figure 33. The **New Transmit Message** window is displayed.

***Figure 33* • Transmit New Message**

2. Enter the message in **ID (Hex)**, **DLC**, and **Data: (Hex)**. Figure 34 shows an example message.

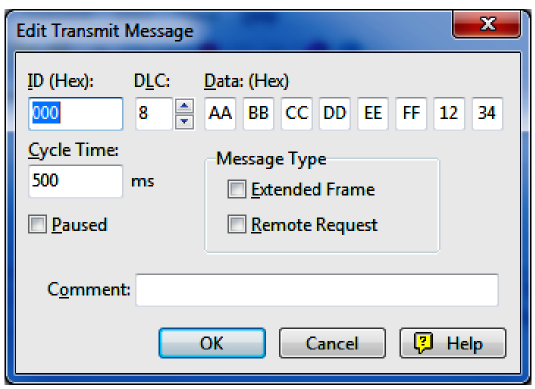

***Figure 34* • Edit Transmit Message**

3. Enter the **Cycle Time** that specifies the duration of an interval (in Milliseconds) for a periodic message transmission.
4. Or enter **0** in **Cycle Time** to transmit the message manually.
5. Click **OK**.

6. Enter **2** in the HyperTerminal. The CAN controller receives the sent message, and the same message is displayed in the HyperTerminal as shown in Figure 35. The message is sent to the CAN bus periodically as specified in **Cycle Time**.

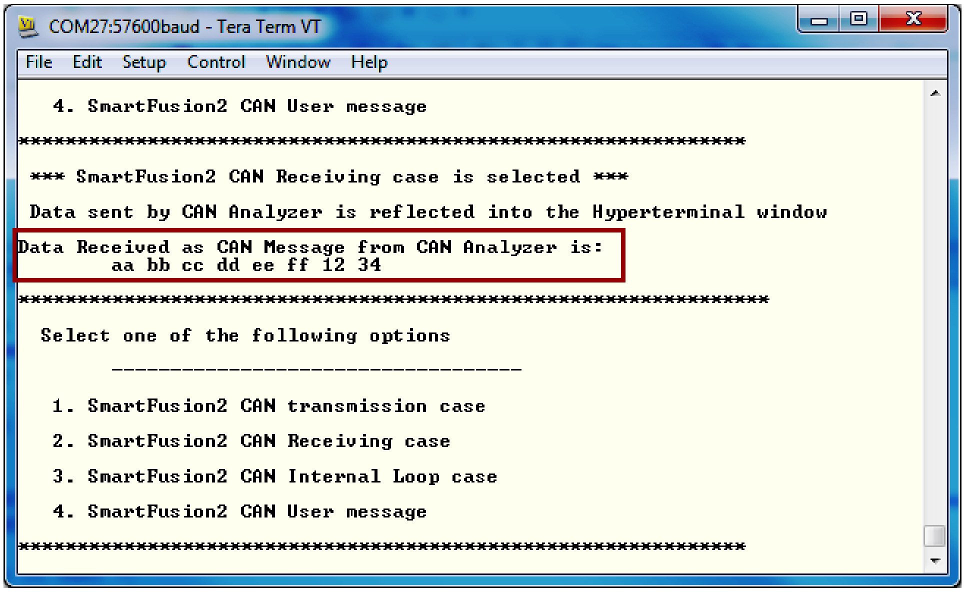

***Figure 35* • Sent Message**

Option 3: Internal Loop Case

The CAN controller is configured in one of the available test modes where the CAN controller can perform. In this configuration, the CAN controller receives the messages, but they are not sent to the network. Instead, the data is sent internally to the CAN. Refer to the Test Modes Table 11-2 of the *SmartFusion2 Microcontroller Subsystem User Guide* for more information on the CAN Test mode operations.

The internal mode function is set by using the driver as follows:

```
MSS_CAN_set_mode(&g_can0, CANOP_MODE_INT_LOOPBACK);
```

- Enter **3**. The message is displayed in the HyperTerminal itself. The **PCAN-View** window does not receive the messages as the messages are looped internally, and they are not sent to the CAN bus as shown in Figure 36 and Figure 37.

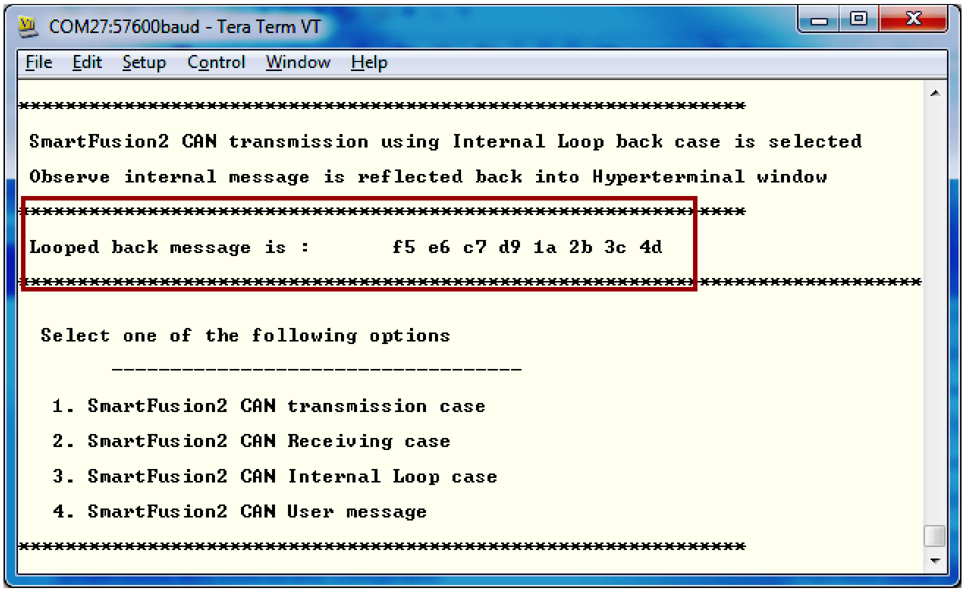

Figure 36 **• Looped Back Message**

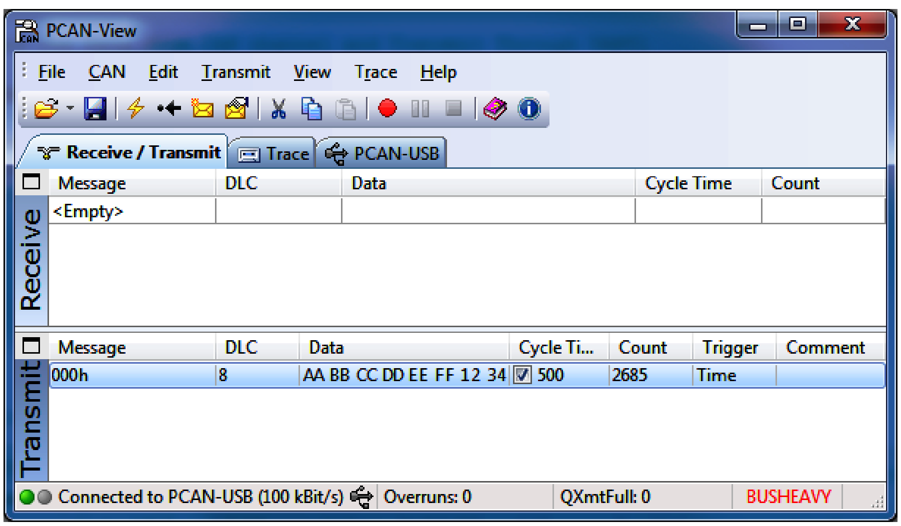

Figure 37 **• PCAN-View**

Option 4: User Specified Message

In this selection, enter any messages to send to the CAN bus from HyperTerminal. The message is sent to the CAN bus and the same message is reflected in the CAN Analyzer.

1. Enter **4** in the HyperTerminal.
2. Enter the message to transmit using CAN controller. (For example, enter aabbccddeeff12).

3. Press **ENTER**. The sent message is reflected in the **PCAN-View** window as shown in Figure 38 and Figure 39.

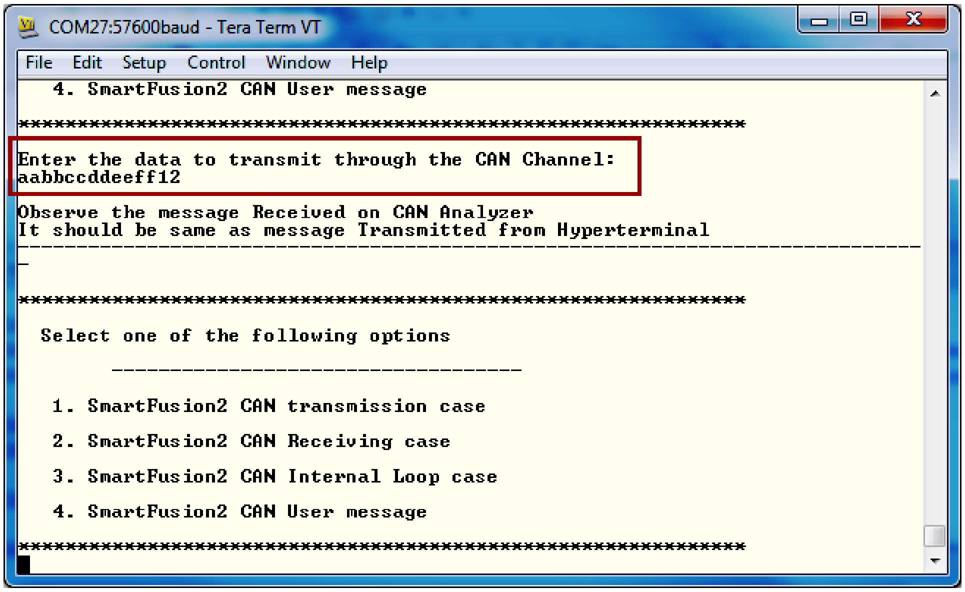

Figure 38 **• Enter Data to Transmit to CAN Bus**

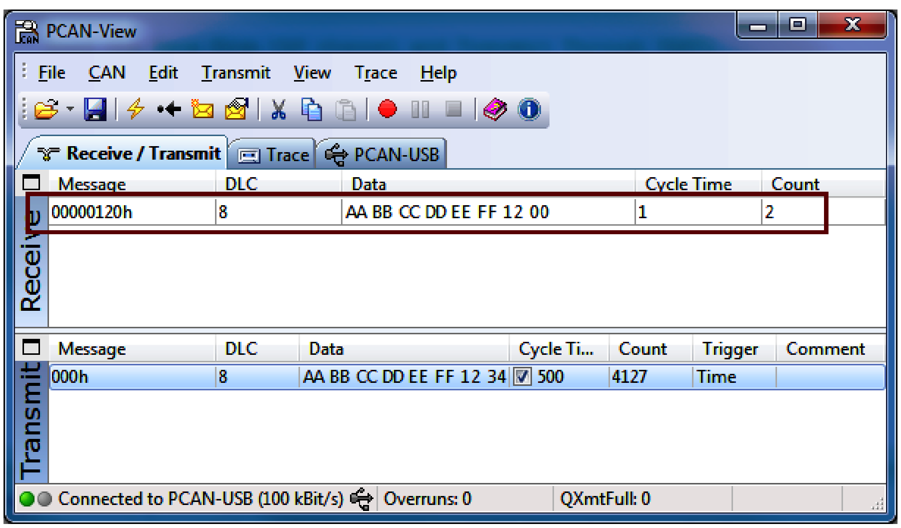

Figure 39 **• PCAN-View Received Data**

Conclusion

This tutorial presents a step-by-step instructions on how to create a new CAN project in Libero SoC, configure and generate various hardware blocks and clocking system using System Builder, open the project in SoftConsole, and write the application code. The example design describes how to validate the application design on SmartFusion2 Development Kit Board using SoftConsole, PCAN-Adapter, and the PCAN-View software.

A – List of Changes

The following table lists the critical changes that were made in each revision

Date	Changes	Page
Revision 2 (February 2015)	Updated the document for Libero SoC v11.5 software release (SAR 63034).	NA
Revision 1 (October 2014)	First Release.	NA
The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.		

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 Microsemi SoC home page.

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:

Technical support email address: 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 at: soc_tech@microsemi.com or contact a local Sales office listing at Sales.Support@Microsemi.com.

ITAR Technical Support

Contact technical support at: soc_tech_itar@microsemi.com for RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR). 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 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 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

50200559-2/02.15