Displaying POT Level with LEDs

Libero SoC and SoftConsole Flow Tutorial for a SmartFusion cSoC

Table of Contents

Introduction....................................3
Tutorial Requirements....................................3
Objective....................................4

Working with Libero SoC and SoftConsole....................................5
Step 1 - Creating a Libero SoC Project....................................5
Step 2 - Configuring MSS Peripherals....................................7
Step 3 - Generating the MSS Component....................................15
Step 4 - Generating the Program File....................................17
Step 5 - Programming SmartFusion Board Using FlashPro....................................18
Step 6 - Building the Software Application Through SoftConsole....................................21
Step 7 - Configuring Serial Terminal Emulation Program....................................26
Step 8 - Installing Drivers for the USB to RS232 Bridge....................................29
Step 9 - Debugging the Application Project using SoftConsole....................................30
Step 10 - Building Executable Image in Release mode....................................36

Appendix A – Libero SoC Catalog Settings....................................37

Appendix B – Firmware Catalog Settings....................................39

Appendix – C....................................41

List of Changes....................................49

Product Support....................................51
Customer Service....................................51
Customer Technical Support Center....................................51
Technical Support....................................51
Website....................................51
Contacting the Customer Technical Support Center....................................51
ITAR Technical Support....................................52

Introduction

This tutorial shows you how to use Microsemi tools to develop an application that can be implemented on SmartFusion® customized system-on-chip (cSoC) device. After completing this tutorial you will be familiar with the following:

- Creating and implementing a Libero® system-on-chip (SoC) v10.0 project using SmartFusion cSoC device.
- Configure the peripherals using SmartDesign.
- Configuring the analog compute engine (ACE).
- Generating the microcontroller subsystem (MSS) Component.
- Generating the programming file to program the SmartFusion cSoC device.
- Opening the project in SoftConsole from Libero SoC and writing application code.
- Compiling application code.
- Creating and launching a debug session.
- Debugging and running the code using SoftConsole.

Tutorial Requirements

Software Requirements

This tutorial requires the following software installed on your PC:

- Libero SoC v10.0 (or later) can be downloaded from: www.microsemi.com/soc/download/software/libero/default.aspx.
- Microsemi SoftConsole v3.3 (or later), which is installed as a part of Libero SoC installation or can be downloaded from www.microsemi.com/soc/download/software/softconsole/default.aspx.

Hardware Requirements

You will need the following hardware:

- SmartFusion Evaluation Kit Board or SmartFusion Development Kit Board
- Two USB cables (programming and communication) — one for connecting the programmer to your PC and the other to connect the universal asynchronous receiver/transmitter (UART) interface on the board to PC.

Associated Project Files

You can download the associated project files for this tutorial from the Microsemi website: www.microsemi.com/soc/download/rsc/?f=SmartFusion_LiberoSC_POTlevel_tutorial_DF

Note: When launching the SoftConsole from Libero SoC, Softconsole may open without any workspace or display an error. Refer to the following KB article for the workaround: www.microsemi.com/soc/kb/article.aspx?id=KI8879.

You can download the programming file (*.stp) in release for this tutorial from the Microsemi website: www.microsemi.com/soc/download/rsc/?f=SmartFusion_LiberoSC_POTlevel_tutorial_PF

MSS Components Used

- ARM® Cortex™-M3 processor
- Clock conditioning circuitry (CCC)
- General purpose input/output (GPIO)
- UART_0
- Analog compute engine (ACE)

Target Board

SmartFusion Evaluation Kit Board (A2F-EVAL-KIT) or SmartFusion Development Kit Board (A2F-DEV-KIT).

Objective

The objective of this tutorial is instruct how to configure SmartFusion cSoC analog channels and ACE that is used to monitor the voltage across the potentiometer (POT). The UART is used to send the ADC results to a terminal program.

Design Steps

- Create a Libero SoC project and use the SmartFusion cSoC MSS Configurator to configure the ACE, adding a voltage monitor with flags.
- Perform synthesis and layout, and generate a programming file to program the SmartFusion cSoC device.
- Open the software project in SoftConsole and write the application program.
- Run an application to monitor the voltage across the POT on the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board.

The hardware configuration has four flags:

- Over 1.0 V
- Over 1.5 V
- Over 2.0 V
- Over 2.5 V

The design monitors the voltage across a POT and four flags are included for the voltage monitoring. These flags are used to drive the four LEDs on the board.

Working with Libero SoC and SoftConsole

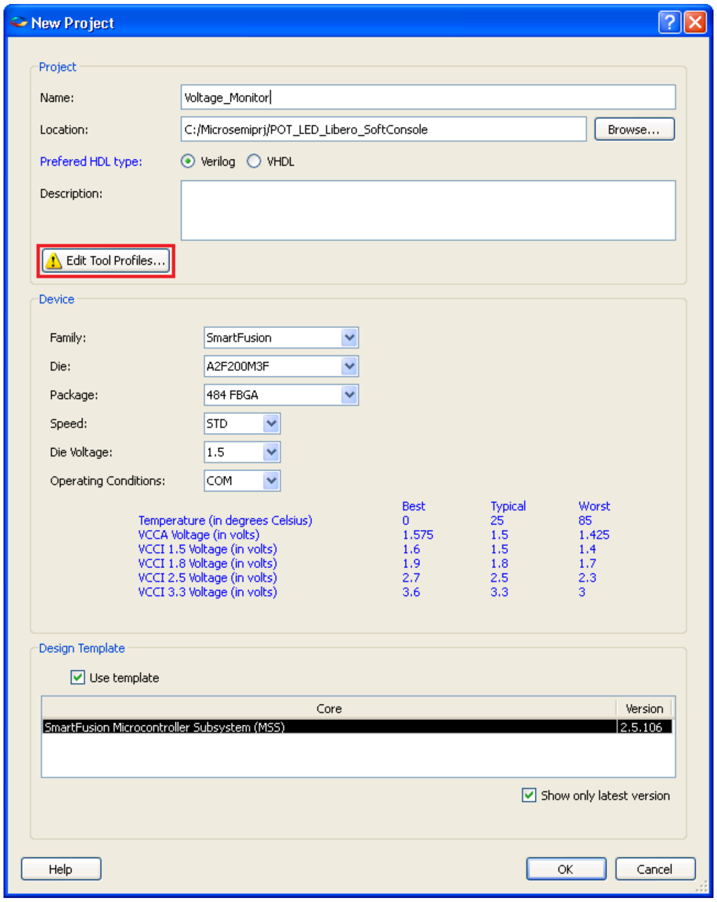
This section describes how to create a Libero SoC project, configure the microcontroller subsystem (MSS), program the design on the SmartFusion board, and run an application program in the SoftConsole IDE.

Step 1 - Creating a Libero SoC Project

1. Launch Libero SoC v10.0 (or later).
2. From the **Project** menu, select **New Project**. Enter the information as displayed in Figure 1 · .

- Name: Voltage_Monitor
- Location: <..> (For example, C:\Microsemiprj\POT_LED_Libero_SoftConsole)
- Family: SmartFusion
- Die: If you are using SmartFusion Evaluation Kit Board, enter A2F200M3F; if you are using SmartFusion Development Kit Board, enter A2F500M3F.
- Package: 484 FBGA
- Speed: STD

Leave others as default.



Name: Voltage_Monitor

Location: C:/Microsemiprj/POT_LED_Libero_SoftConsole

Prefered HDL type: ● Verilog ○ VHDL

Description:

Family: SmartFusion

Die: A2F200M3F

Package: 484 FBGA

Speed: STD

Die Voltage: 1.5

Operating Conditions: COM

	Best	Typical	Worst
Temperature (in degrees Celsius)	0	25	85
VCCA Voltage (in volts)	1.575	1.5	1.425
VCCI 1.5 Voltage (in volts)	1.6	1.5	1.4
VCCI 1.8 Voltage (in volts)	1.9	1.8	1.7
VCCI 2.5 Voltage (in volts)	2.7	2.5	2.3
VCCI 3.3 Voltage (in volts)	3.6	3.3	3

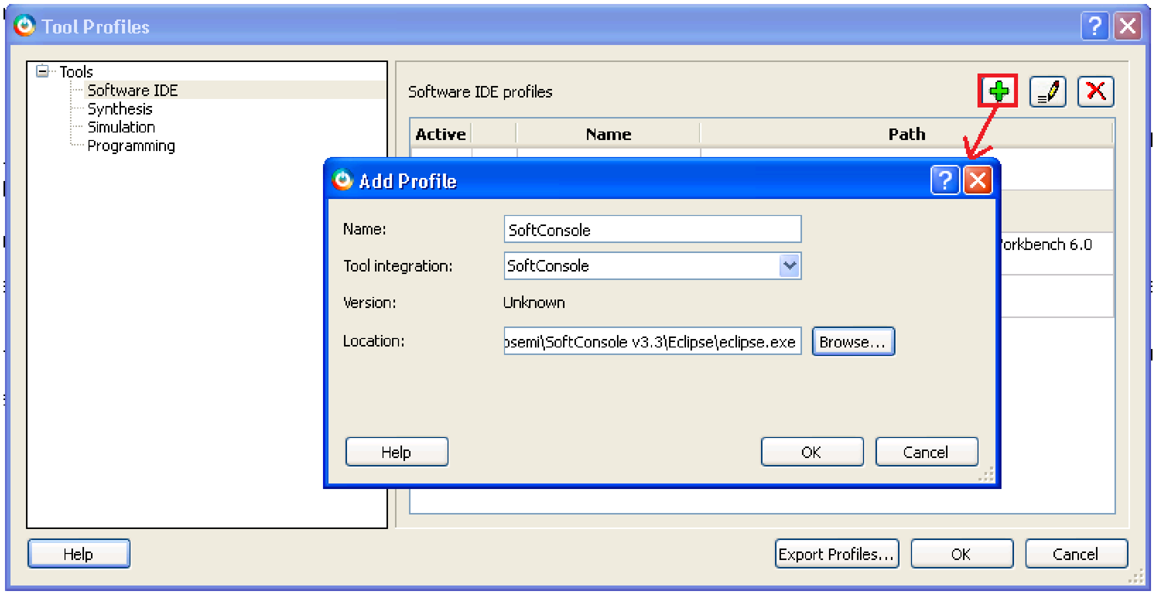
☑ Use template

Core	Version
SmartFusion Microcontroller Subsystem (MSS)	2.5.106

☑ Show only latest version

Figure 1 · New project Dialog Box

3. Click **Edit Tool Profiles** and add SoftConsole by clicking on Software IDE as shown in Figure 2 · .



Name: SoftConsole
Tool integration: SoftConsole
Version: Unknown
Location: osemi\SoftConsole v3.3\Eclipse\eclipse.exe

Figure 2 · Selecting SoftConsole as Software IDE

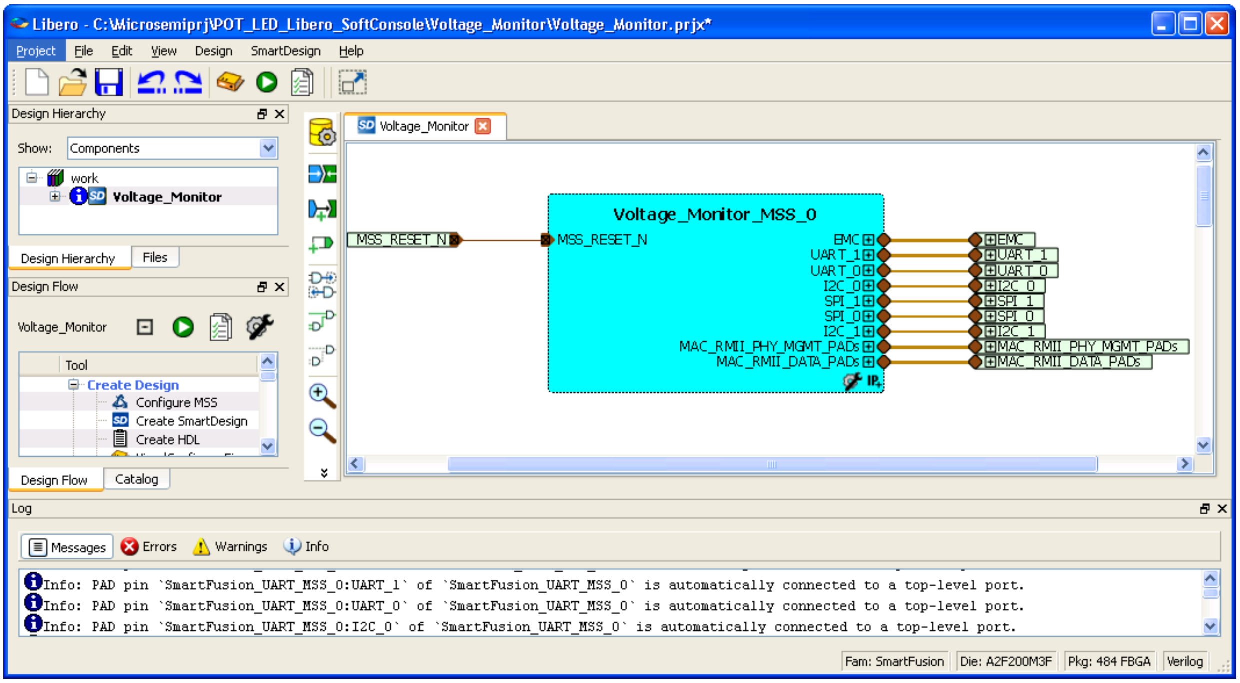
4. After adding the Profile, click **OK** to close the **Add Profile** dialog window.

   Repeat the steps (3 and 4) above for Synthesis, Simulation, and Programming and then click **OK** to close the Tool Profiles dialog window.

5. Select the MSS core in **New Project** dialog box and click **OK.**

   Note: If SmartFusion cSoC MSS does not appear in the list, refer to Appendix A – Libero SoC Catalog Settings to find out how to set the repositories. If your vault does not have the MSS core, then download the core by double clicking on the core name in **Design template** in **New Project** dialog box.

6. The project is created and the Libero SoC window appears as shown in Figure 3 · . The SmartDesign “Voltage_Monitor” is created with the instantiation of MSS component.



```
Info: PAD pin `SmartFusion_UART_MSS_0:UART_1` of `SmartFusion_UART_MSS_0` is automatically connected to a top-level port.
Info: PAD pin `SmartFusion_UART_MSS_0:UART_0` of `SmartFusion_UART_MSS_0` is automatically connected to a top-level port.
Info: PAD pin `SmartFusion_UART_MSS_0:I2C_0` of `SmartFusion_UART_MSS_0` is automatically connected to a top-level port.
```

Fam: SmartFusion | Die: A2F200M3F | Pkg: 484 FBGA | Verilog

Figure 3 · The Libero Window After Completing New Project Wizard

Step 2 - Configuring MSS Peripherals

1. Double click on **Voltage_Monitor_MSS_0** component to configure the MSS**.** The MSS is displayed in the SmartDesign Canvas opens in a new tab, as shown in Figure 4 · .

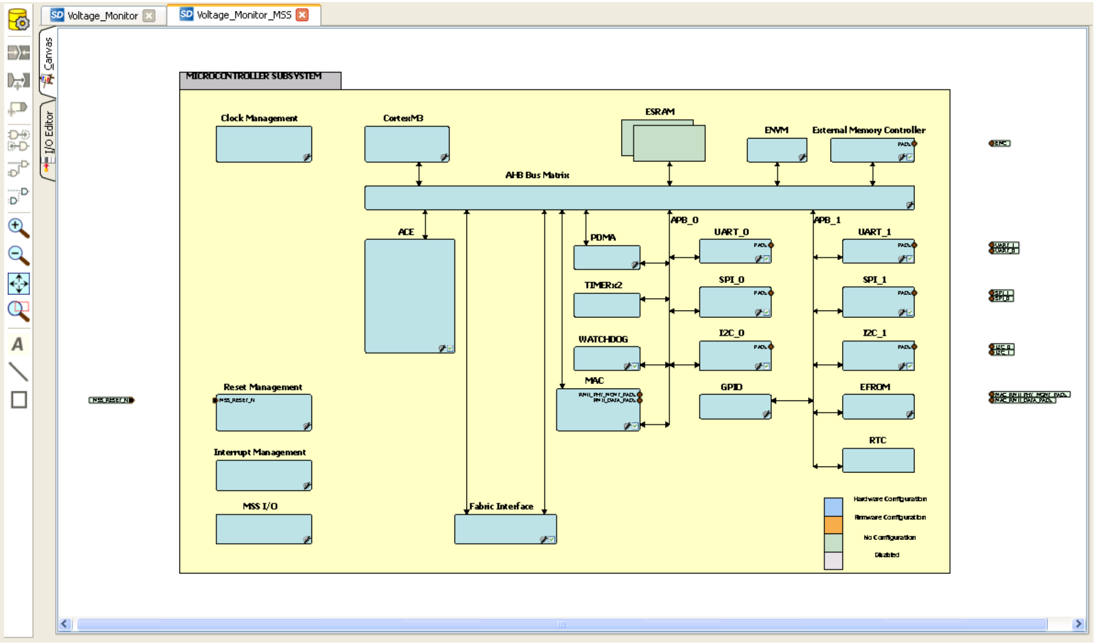


Figure 4 · MSS in the SmartDesign Canvas

The enabled MSS peripherals are highlighted in blue, and can be configured in the hardware. The disabled peripherals are shown in gray.

To disable a peripheral that is not required, select the peripheral, right-click, and clear the **Enabled** check box or, or clear the check box in the lower right corner of the peripheral box. The box turns grey to indicate that the peripheral has been disabled. Disabled peripherals can be enabled by repeating the procedure.

An enabled peripheral looks as shown in Figure 5 · .

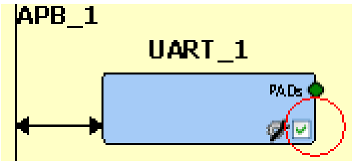


Figure 5 · Enabling the Peripheral

This example uses only the clock management, ACE, GPIO, and UART_0 peripherals.

2. Disable the following peripherals: MAC, WATCHDOG, Fabric Interface, SPI0, SPI1, I2C0, I2C1, UART1, and EMC.

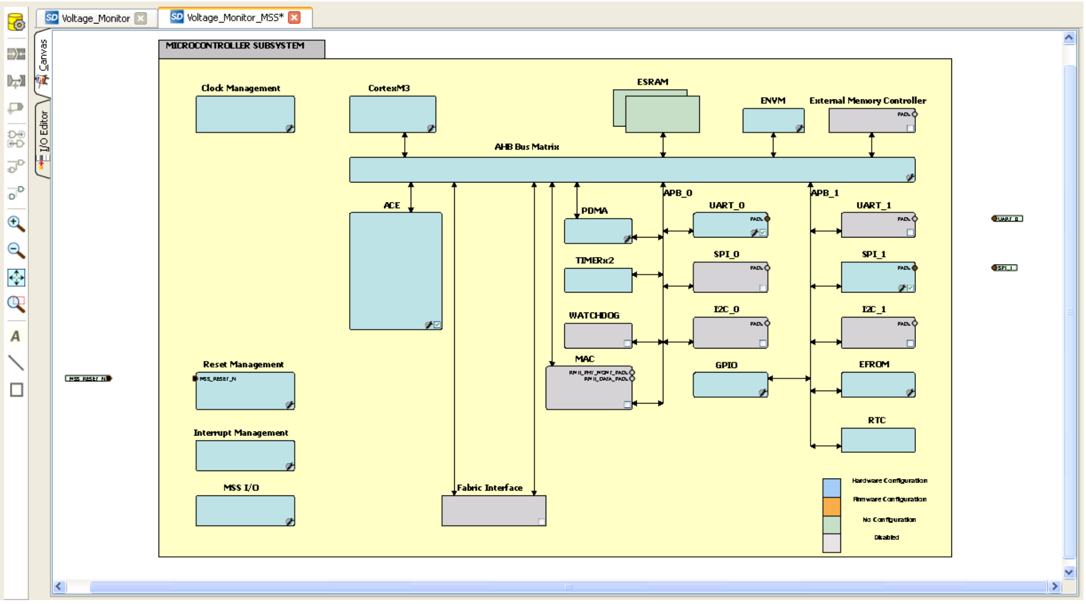


Figure 6 · Used MSS Peripherals

3. Double-click the **Clock Management** block and configure as shown below:
   - CLKA: On-Chip RC Oscillator
   - MSS clock source: PLL output
   - MSS clock frequency: 80 MHz

   Use default settings for all other fields.

   After completing the configuration, click **OK**.

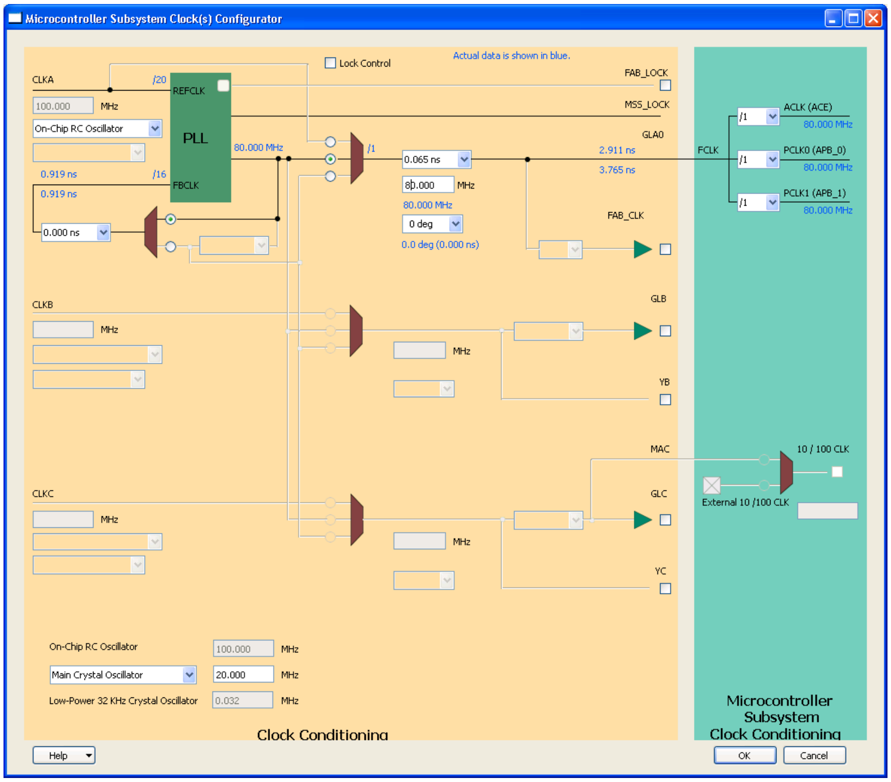
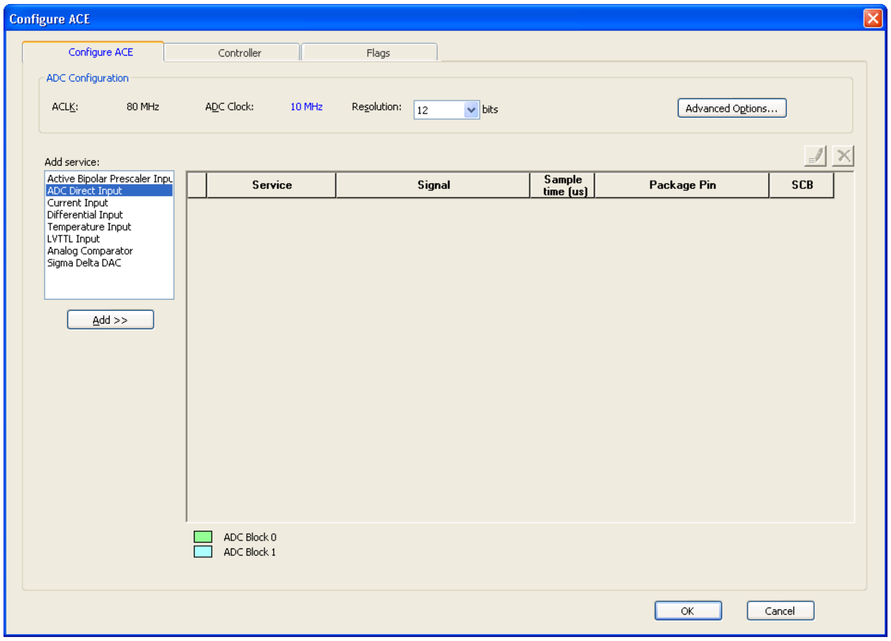


Figure 7 · MSS Clock Configuration

Configuring ACE

1. To configure ACE, double-click the **ACE** peripheral block and configure as follows:

   Connect TM0 to the POT on the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board. Configure a voltage monitor to measure the voltage across the POT and also to create flags to indicate when the voltage is greater than 1.0 V, 1.5 V, 2.0 V, and 2.5 V. These flags are used to illuminate the LEDs on the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board.


Configure ACE

ADC Configuration

ACLK: 80 MHz   ADC Clock: 10 MHz   Resolution: 12 bits

Add service:

- Active Bipolar Prescaler Inpu
- ADC Direct Input
- Current Input
- Differential Input
- Temperature Input
- LVTTL Input
- Analog Comparator
- Sigma Delta DAC

	Service	Signal	Sample time (us)	Package Pin	SCB

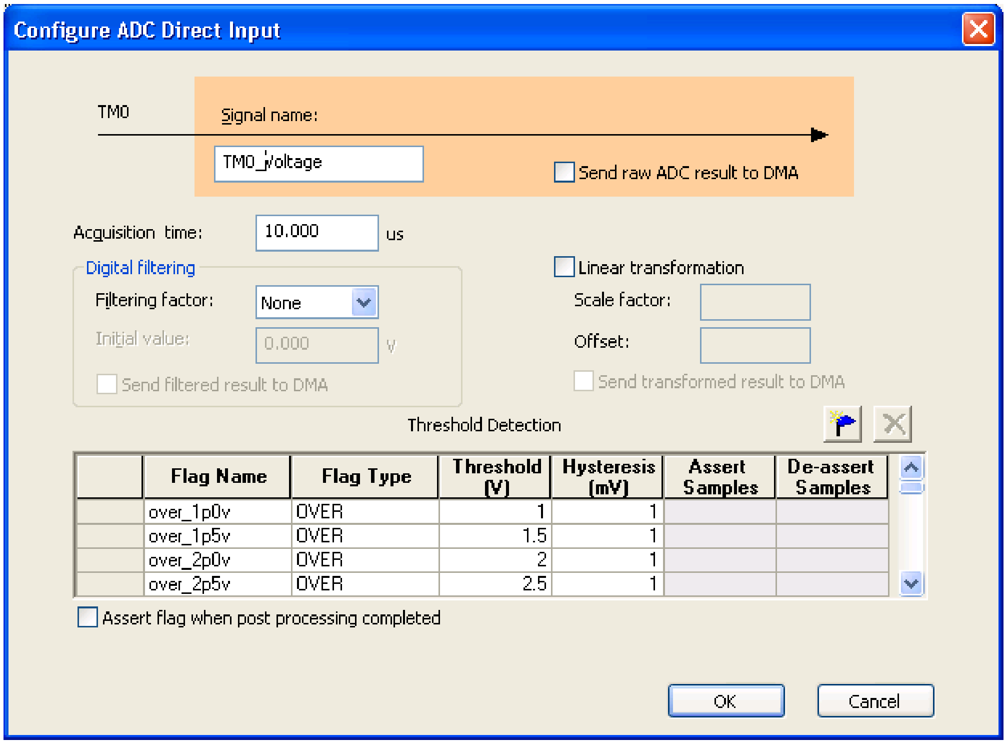
🟩 ADC Block 0
🟦 ADC Block 1

Figure 8 · MSS ACE Configuration

2. Select **ADC Direct Input** > **Add** (or, double-click **ADC Direct Input**) and enter the parameters as shown in Figure 9 · .
   - Signal name: TM0_Voltage
   - Send raw results to DMA: Cleared check box
   - Acquisition time: 10 µs
   - Filtering factor: None
3. Next, add the flags as shown in Table 1:

Table 1 · Flag Definitions

Flag Name	Flag Type	Threshold (V)	Hysteresis (mV)
over_1p0v	OVER	1	1
over_1p5v	OVER	1.5	1
over_2p0v	OVER	2	1
over_2p5v	OVER	2.5	1



Signal name: TM0_Voltage

□ Send raw ADC result to DMA

Acquisition time: 10.000 us

Filtering factor: None

Initial value: 0.000 V

□ Send filtered result to DMA

□ Linear transformation

Scale factor:

Offset:

□ Send transformed result to DMA

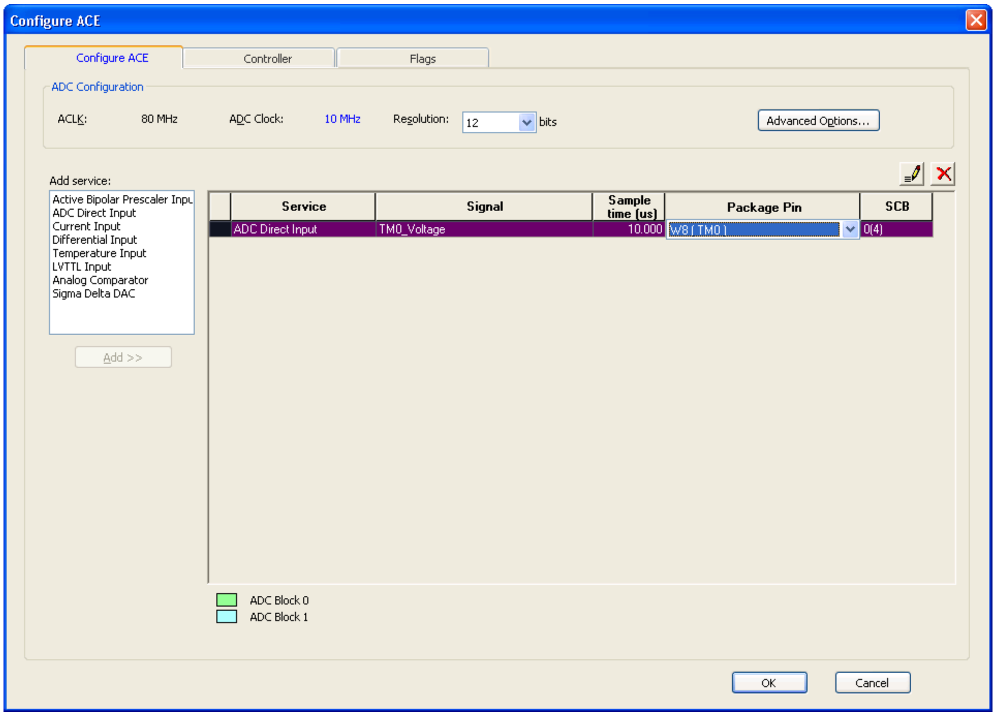
	Flag Name	Flag Type	Threshold (V)	Hysteresis (mV)	Assert Samples	De-assert Samples
	over_1p0v	OVER	1	1		
	over_1p5v	OVER	1.5	1		
	over_2p0v	OVER	2	1		
	over_2p5v	OVER	2.5	1		

□ Assert flag when post processing completed

Figure 9 · MSS ADC Direct Input Configuration

4. Click **OK.**

5. Assign the ADC Direct Input Signal to package pin W8 in the Configure ACE dialog box. The **Configure ACE** tab is displayed, as shown in Figure 10 · :



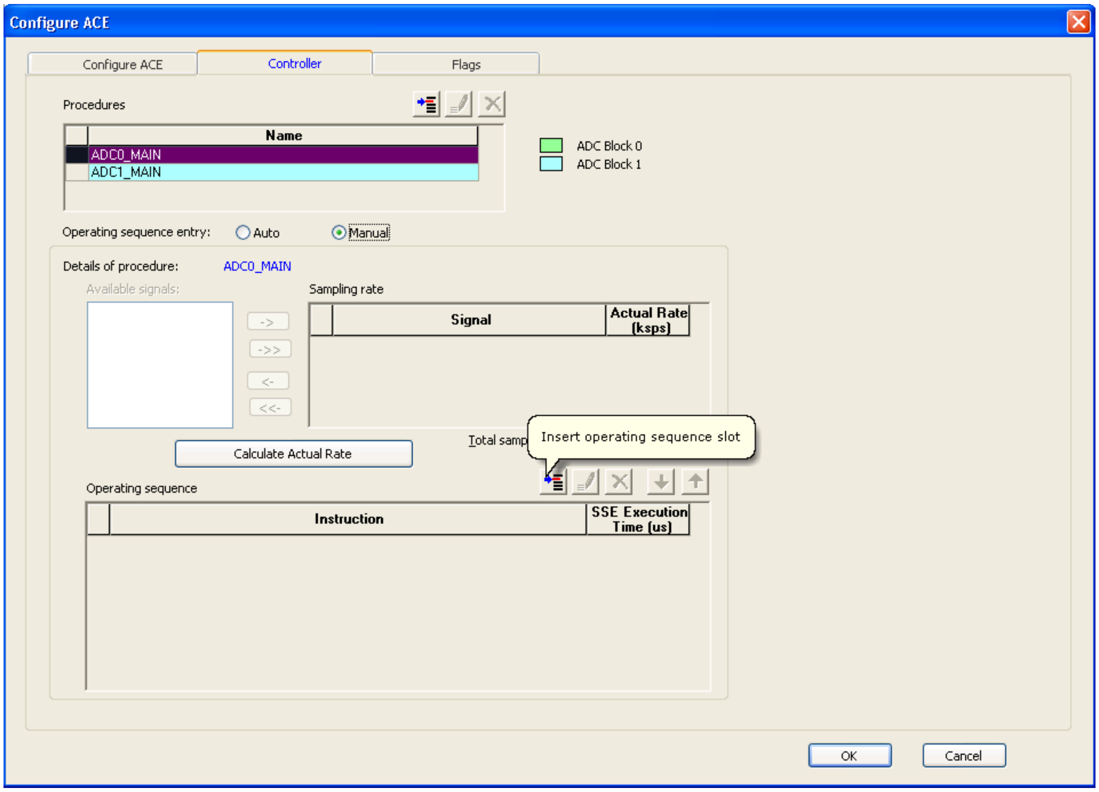
ACLK: 80 MHz ADC Clock: 10 MHz Resolution: 12 bits

Service	Signal	Sample time (us)	Package Pin	SCB
ADC Direct Input	TM0_Voltage	10.000	W8 (TM0)	0(4)

🟩 ADC Block 0
🟦 ADC Block 1

Figure 10 · MSS ACE Configuration With ADC Direct Input

6. The next step in configuring the ACE is to enable the sampling sequence. This configuration dialog is launched by clicking on the **Controller** tab (next to the **Configure ACE** tab).
7. Select **Manual** as the **Operating sequence entry** in the **Controller** tab.



Name
ADC0_MAIN
ADC1_MAIN

Operating sequence entry: ○ Auto ● Manual

Details of procedure: ADC0_MAIN

Figure 11 · MSS ACE Configuration to Enable Sampling Sequence

8. Click **Insert operating sequence slot** as shown in Figure 11 · .
9. Select **SAMPLE**.

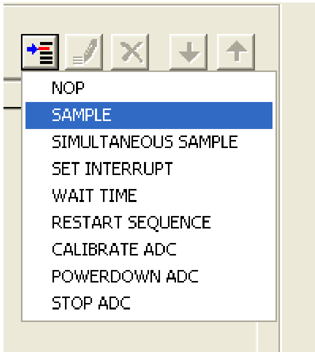
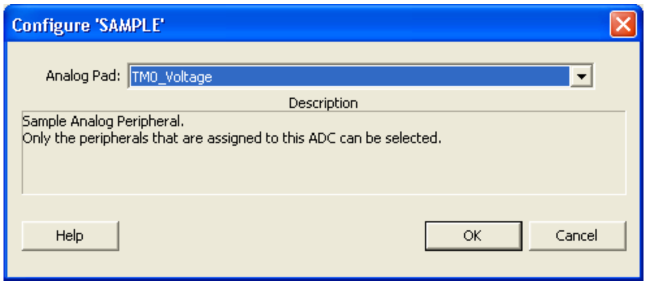


Figure 12 · Select SAMPLE

10. The **Configure SAMPLE** window is displayed. Select **TM0_voltage** as shown in Figure 13 · and click **OK**.



Analog Pad: TM0_Voltage

Sample Analog Peripheral.
Only the peripherals that are assigned to this ADC can be selected.

Figure 13 · Configure SAMPLE

11. Click the Insert operating sequence slot again and select RESTART SEQUENCE.

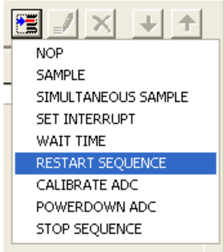
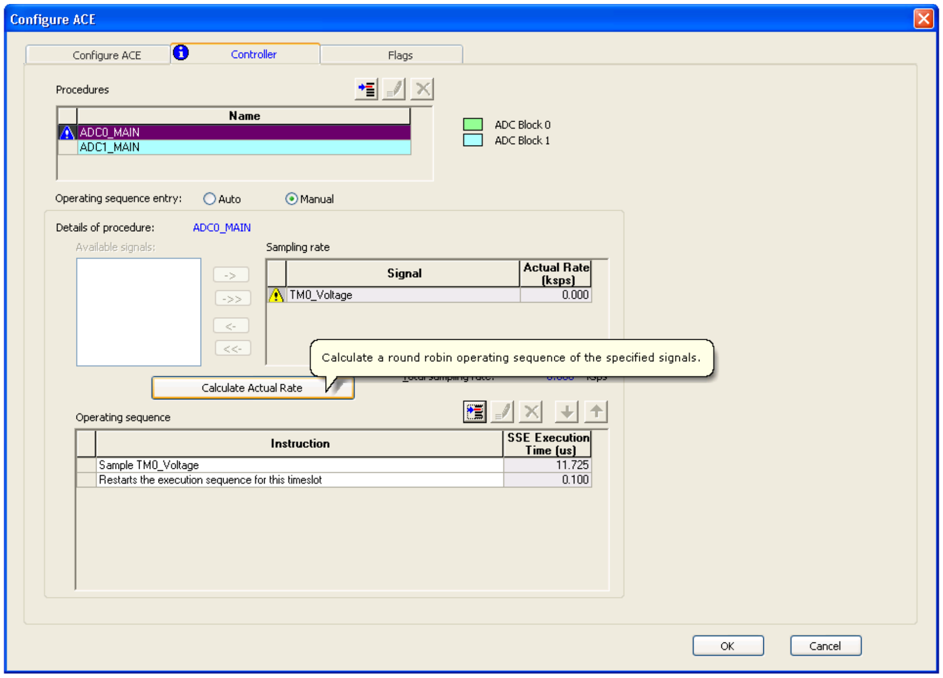


Figure 14 · Select Restart Sequence

12. Click **Calculate Actual Rate**.



Procedures

Name
ADC0_MAIN
ADC1_MAIN

🟩 ADC Block 0
🟦 ADC Block 1

Operating sequence entry: ○ Auto ● Manual

Details of procedure: ADC0_MAIN

Available signals:

Sampling rate

Signal	Actual Rate (ksps)
TM0_Voltage	0.000

Calculate a round robin operating sequence of the specified signals.

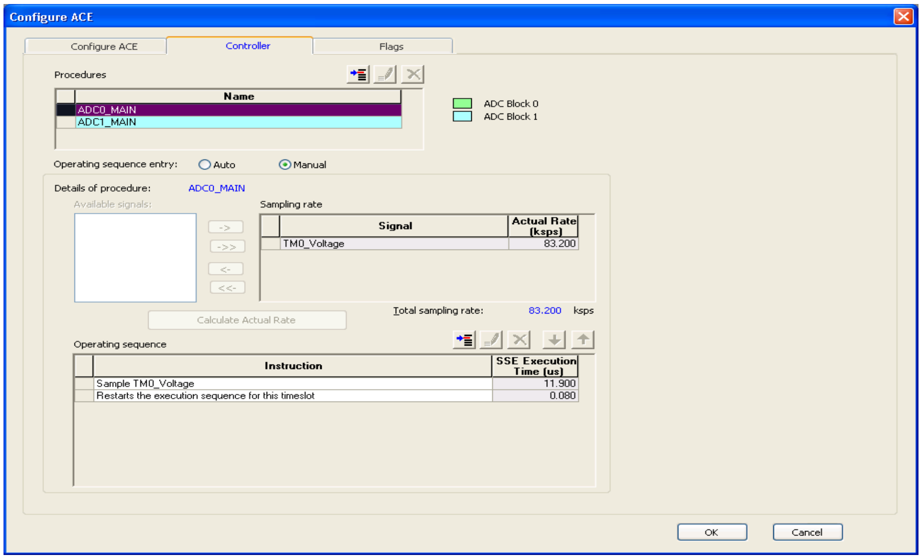
Calculate Actual Rate

Operating sequence

Instruction	SSE Execution Time (us)
Sample TM0_Voltage	11.725
Restarts the execution sequence for this timeslot	0.100

Figure 15 · MSS ACE Configuration: Final Controller Tab

13. The **Controller** tab window is displayed as shown in Figure 16 · :



Procedures

Name
ADC0_MAIN
ADC1_MAIN

🟩 ADC Block 0
🟦 ADC Block 1

Operating sequence entry: ○ Auto ● Manual

Details of procedure: ADC0_MAIN

Available signals:

Sampling rate

Signal	Actual Rate (ksps)
TM0_Voltage	83.200

Total sampling rate: 83.200 ksps

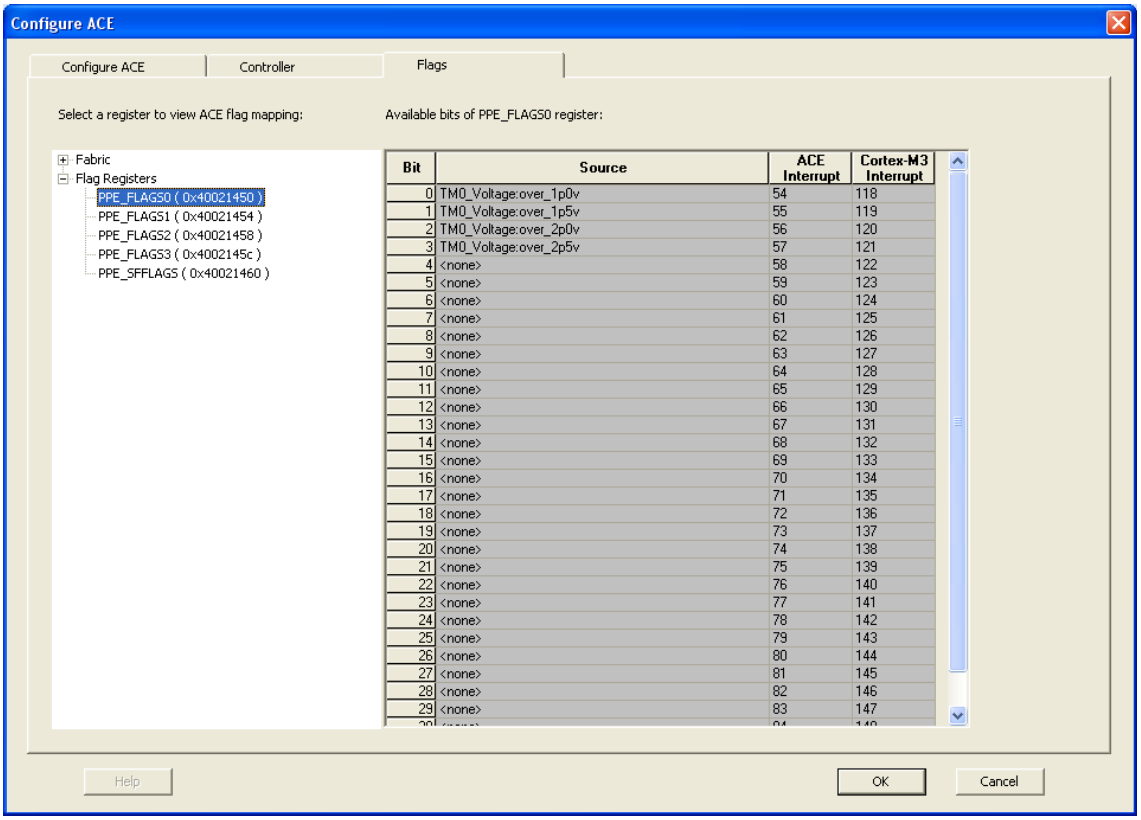
Operating sequence

Instruction	SSE Execution Time (us)
Sample TM0_Voltage	11.900
Restarts the execution sequence for this timeslot	0.080

Figure 16 · MSS ACE Configuration: Controller Tab

14. Click the **Flags** tab in the Configure ACE window. This tab lists the flags set from PPE registers.

15. Click the **+** sign to expand the Flag registers group. The PPE_FLAGSn registers contain the user-defined flags.
16. Select **PPE_FLAGS0** (FLAGBANK0). PPE_FLAGS0 contains the 4 threshold flags assigned earlier. These are the flags that were defined when the direct input voltage service was configured. The flag register can be read by the Cortex-M3 processor. The flags also generate interrupts to the Cortex-M3 processor.



Select a register to view ACE flag mapping:

- Fabric
- Flag Registers
  - PPE_FLAGS0 (0x40021450)
  - PPE_FLAGS1 (0x40021454)
  - PPE_FLAGS2 (0x40021458)
  - PPE_FLAGS3 (0x4002145c)
  - PPE_SFFLAGS (0x40021460)

Available bits of PPE_FLAGS0 register:

Bit	Source	ACE Interrupt	Cortex-M3 Interrupt
0	TM0_Voltage:over_1p0v	54	118
1	TM0_Voltage:over_1p5v	55	119
2	TM0_Voltage:over_2p0v	56	120
3	TM0_Voltage:over_2p5v	57	121
4	<none>	58	122
5	<none>	59	123
6	<none>	60	124
7	<none>	61	125
8	<none>	62	126
9	<none>	63	127
10	<none>	64	128
11	<none>	65	129
12	<none>	66	130
13	<none>	67	131
14	<none>	68	132
15	<none>	69	133
16	<none>	70	134
17	<none>	71	135
18	<none>	72	136
19	<none>	73	137
20	<none>	74	138
21	<none>	75	139
22	<none>	76	140
23	<none>	77	141
24	<none>	78	142
25	<none>	79	143
26	<none>	80	144
27	<none>	81	145
28	<none>	82	146
29	<none>	83	147
30	<none>	84	148

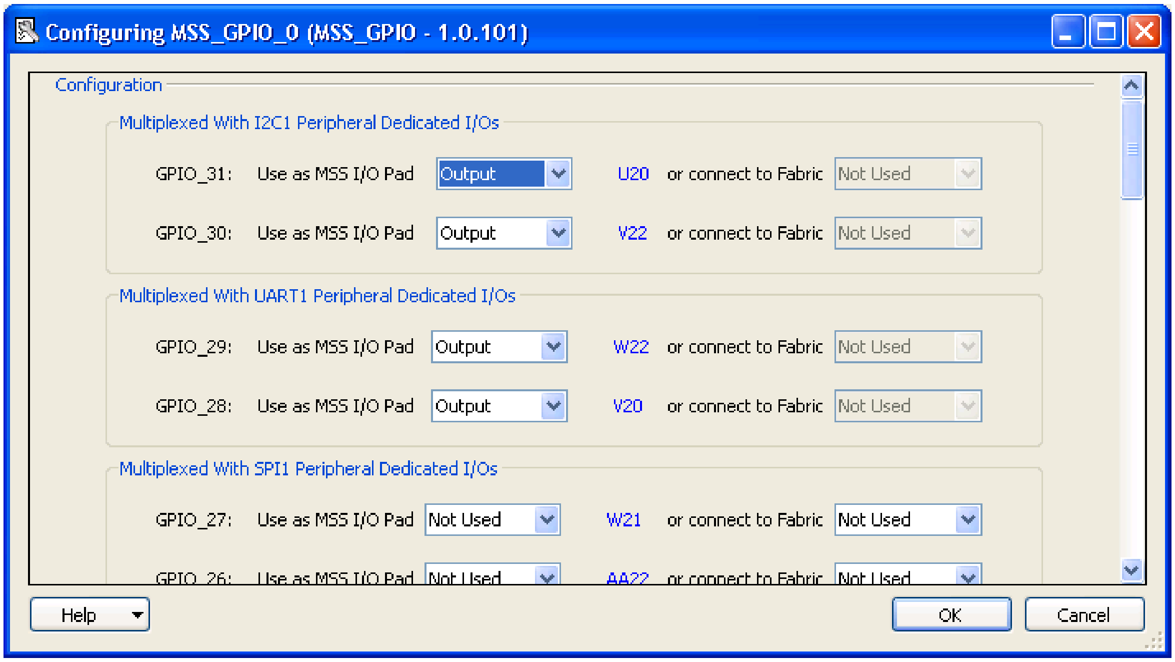
Figure 17 · ACE Flag Mapping - PPE Flag Registers

17. Click **OK** to close the ACE configuration window.

Configuring the GPIO Peripheral

Note: If you are not using the SmartFusion Evaluation Kit Board Revision 2 or later, or using the SmartFusion Development Kit Board, follow Appendix – C. Skip Step 3 - Generating the MSS Component and Step 4 - Generating the Program File.

1. Double-click the **GPIO block** in the **MSS component**, configure as shown in Figure 18 · , and click **OK**.



Configuring MSS_GPIO_0 (MSS_GPIO - 1.0.101)

Configuration

Multiplexed With I2C1 Peripheral Dedicated I/Os

GPIO_31: Use as MSS I/O Pad Output U20 or connect to Fabric Not Used

GPIO_30: Use as MSS I/O Pad Output V22 or connect to Fabric Not Used

Multiplexed With UART1 Peripheral Dedicated I/Os

GPIO_29: Use as MSS I/O Pad Output W22 or connect to Fabric Not Used

GPIO_28: Use as MSS I/O Pad Output V20 or connect to Fabric Not Used

Multiplexed With SPI1 Peripheral Dedicated I/Os

GPIO_27: Use as MSS I/O Pad Not Used W21 or connect to Fabric Not Used

GPIO_26: Use as MSS I/O Pad Not Used AA22 or connect to Fabric Not Used

Figure 18 · Configure MSS_GPIO_0

2. This example requires GPIO_31, GPIO_30, GPIO_29, and GPIO_28 to be connected to LED_8 to LED_5 on the SmartFusion Evaluation Kit Board (A2F-EVAL-KIT Rev 2).
3. Click **File > Save** to save the Voltage_Monitor_MSS.

Step 3 - Generating the MSS Component

1. Right-click on **Voltage_Monitor_MSS_0** component on the **Voltage_Monitor** tab and select **Update Instance(s) with Latest Component** as shown in Figure 19 · .

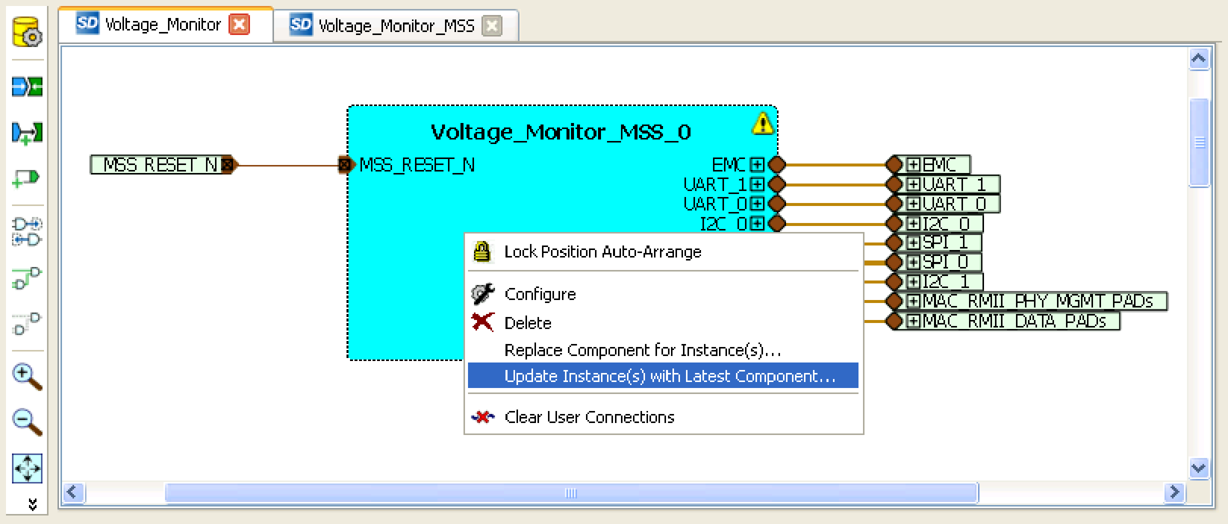


Figure 19 · Updating the MSS

2. Click **Design > Configure Firmware** as shown in Figure 20 · .

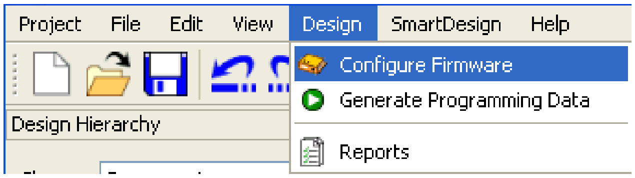
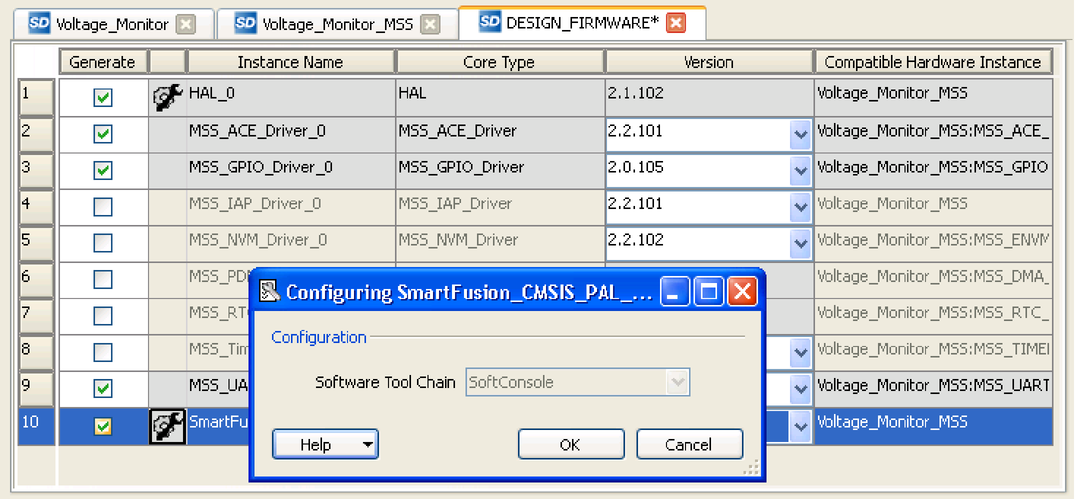


Figure 20 · Opening Design_Firmware

3. On the **DESIGN_FIRMWARE** tab, clear the **Generate** check boxes for all the peripherals for which you do not need to generate the firmware. Click **Configuration** on the SmartFusion_CMSIS_PAL_0 instance and select SoftConsole as the configuration.


Voltage_Monitor | Voltage_Monitor_MSS | DESIGN_FIRMWARE*

	Generate		Instance Name	Core Type	Version	Compatible Hardware Instance
1	☑		HAL_0	HAL	2.1.102	Voltage_Monitor_MSS
2	☑		MSS_ACE_Driver_0	MSS_ACE_Driver	2.2.101	Voltage_Monitor_MSS:MSS_ACE_
3	☑		MSS_GPIO_Driver_0	MSS_GPIO_Driver	2.0.105	Voltage_Monitor_MSS:MSS_GPIO
4	☐		MSS_IAP_Driver_0	MSS_IAP_Driver	2.2.101	Voltage_Monitor_MSS
5	☐		MSS_NVM_Driver_0	MSS_NVM_Driver	2.2.102	Voltage_Monitor_MSS:MSS_ENVM
6	☐		MSS_PD			Voltage_Monitor_MSS:MSS_DMA_
7	☐		MSS_RT			Voltage_Monitor_MSS:MSS_RTC_
8	☐		MSS_Tim			Voltage_Monitor_MSS:MSS_TIMEI
9	☑		MSS_UA			Voltage_Monitor_MSS:MSS_UART
10	☑		SmartFu			Voltage_Monitor_MSS

Configuring SmartFusion_CMSIS_PAL_...

Configuration

Software Tool Chain: SoftConsole

Figure 21 · Configuring SmartFusion_CMSIS_PAL_0

4. Check whether or not you are able to see the latest version of the drivers without any warning or error indicating that firmware is missing from the Vault. If missing, refer to Appendix A – Libero SoC Catalog Settings.
5. Click **File > Save** to Save the Design_Firmware.
6. **Save** the design and generate the component by clicking **Generate Component** or by selecting **SmartDesign > Generate Component**.

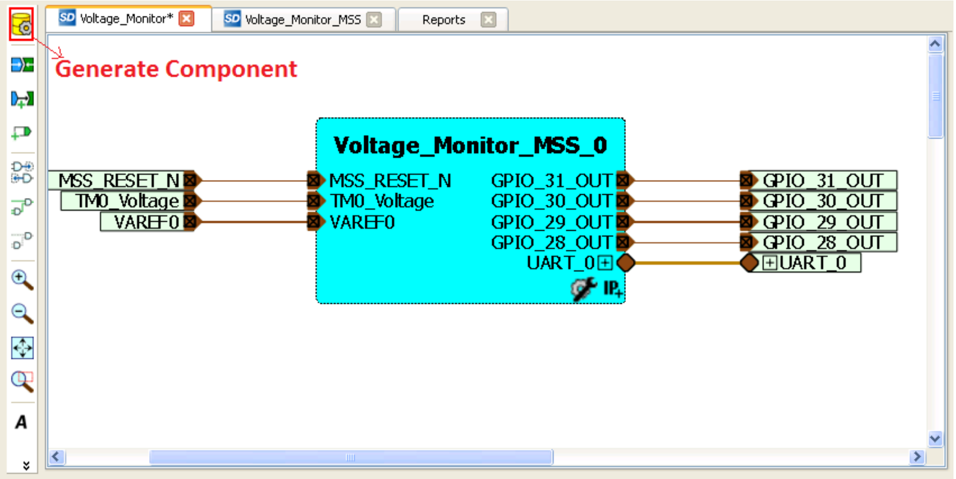
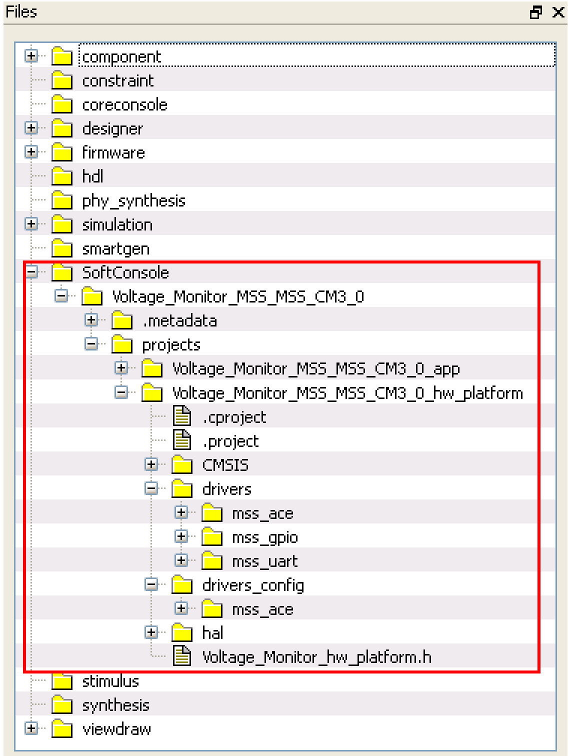


Figure 22 · Generating the MSS Component

7. After successful generation of project the log window displays the message "**Info: 'Voltage_Monitor' was successfully generated. Open datasheet for details**". The datasheet has the Project information like Generated files, used IO's, and Memory map etc.
8. Confirm that the SoftConsole folder is created with the folders and files as shown in Figure 23 · .



Files

```
component
constraint
coreconsole
designer
firmware
hdl
phy_synthesis
simulation
smartgen
SoftConsole
    Voltage_Monitor_MSS_MSS_CM3_0
        .metadata
        projects
            Voltage_Monitor_MSS_MSS_CM3_0_app
            Voltage_Monitor_MSS_MSS_CM3_0_hw_platform
                .cproject
                .project
                CMSIS
                drivers
                    mss_ace
                    mss_gpio
                    mss_uart
                drivers_config
                    mss_ace
                hal
                Voltage_Monitor_hw_platform.h
stimulus
synthesis
viewdraw
```

Figure 23 · Files Window

Step 4 - Generating the Program File

Libero SoC provides the push-button flow for Generating programming data of the project in a single step. By clicking **Generate Programming Data**, you can complete the synthesis, place and route, verify timing and generating the programming file. You can also complete the flow by running the synthesis and place and route tools in interactive mode (step-by-step), for more information refer *Libero SoC Quick Start Guide*.

Push-button Design Flow

1. Click **Generate Programming Data** as shown in Figure 24 · to complete the place and route, verify timing, and generate the programming file. This completes the.fdb file generation.

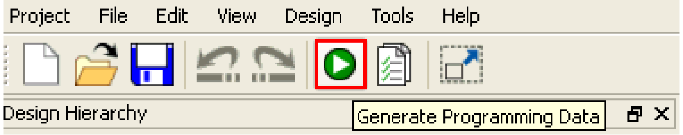


Figure 24 · Generating Programming Data

2. The **Design Flow** window looks as shown in Figure 25 · .

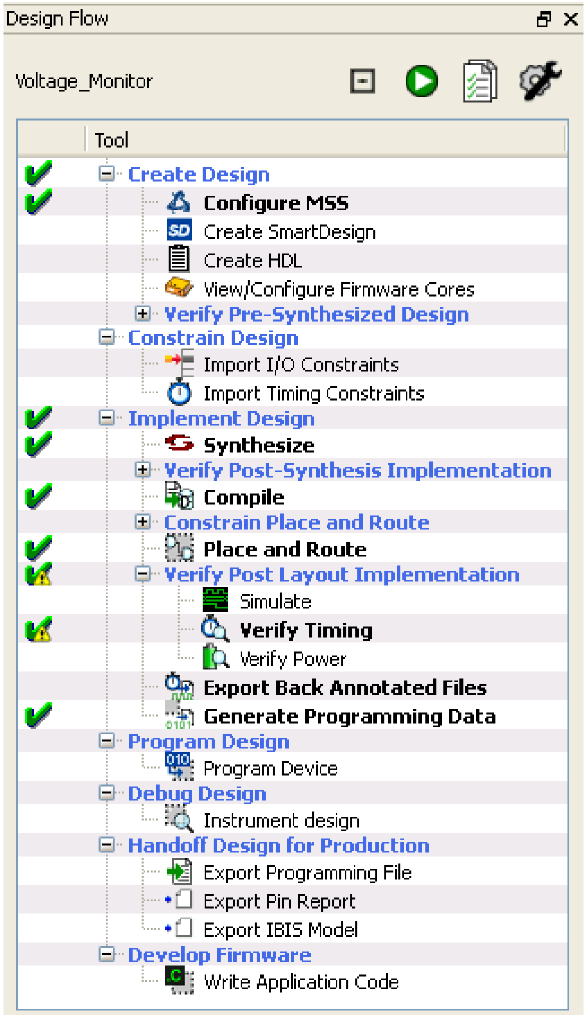


Figure 25 · Design Flow Window After Building the Project

Step 5 - Programming SmartFusion Board Using FlashPro

Before you proceed with programming the device, ensure that the low cost programming stick (LCPS) or FlashPro4 is properly connected to the board. Use the following details to ensure the correct jumper settings. Refer to the *SmartFusion Evaluation Kit User's Guide* and the *SmartFusion Development Kit User's Guide* for additional information.

Jumper Settings for SmartFusion Evaluation Kit Board

- JP10: Short pin 1 and 2 using a jumper.
- JP7: Short pin 1 and 2 using a jumper for LCPS mode.
- J6: Connect pin 1 and 2 using the jumper.
- JP6: Connect pin 2 and 3 using the jumper.
- J13: Connect the USB cable to J13 connector. Install the FlashPro4 or FlashPro drivers if they are not already installed.
- J14: Connect second USB cable for power.
- JP11, JP12, JP13, and JP14: Short pin 2 and 3 using a jumper (in A2F - EVAL - REV 2).

Jumper Settings for SmartFusion Development Kit Board

SW9 must be off (JTAGSEL = H) in order to program the SmartFusion device. SW9 remains in the off position for Libero SoC and SoftConsole programming. Make the jumper settings as shown in the following table:

Table 2 · Jumper Settings for Development Kit Board

Factory Default	Factory Default	Factory Default
JP1: 1–2	JP12: 1–2	JP21: 1–2
JP2: 1–2	JP13: 1–2	JP22: 2–3
JP4: 1–3; 7–9	JP14: 1–2	JP23: 1–2
JP5: 1–2; 3–4	JP15: 1–2	JP24: 1–2
JP6: 2–3	JP16: 2–3	JP27: 1–2
J7: 2–3; 6–7; 10–11; 14–15	JP17: 2–3	JP28: 1-2
JP7: 1–2	JP18: 1–2	J32: 1–2; 3–4; 5–6
JP8: 3-4; 7-8; 11-12; 15-16	JP19: 2–3	–
JP11: 1-2	JP20: 1–2	–

Programming the Device

Double click **Program Device** under **Program Design** in the **Design Flow** window to program the SmartFusion cSoC device. Click **Yes** when it prompts that the timing constraints are not yet set.

Note: Do not interrupt the programming sequence; it may damage the device or the programmer. If you face any problems, contact Microsemi Tech Support at soc_tech@microsemi.com.

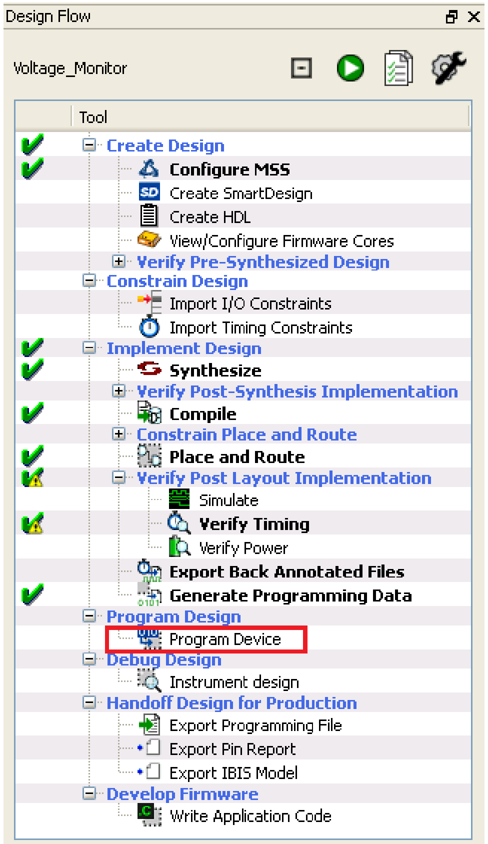
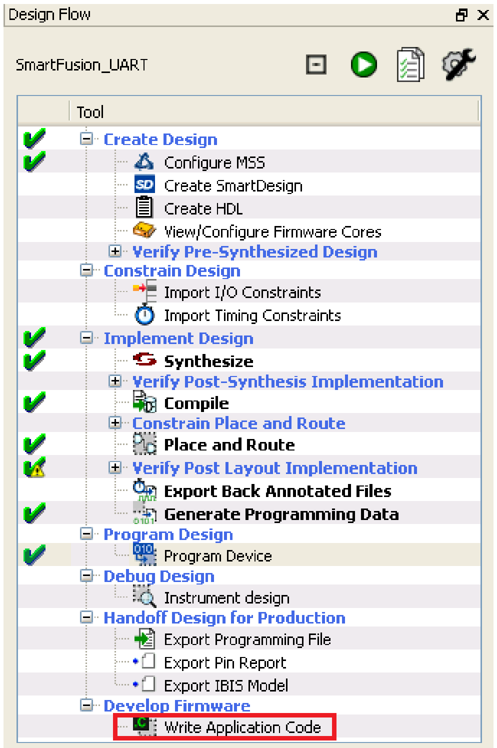


Figure 26 · Design Flow Window

You can also run flash pro interactively by right clicking on **Program Device** in **Design Flow** window and selecting **Open Interactively**. For more information on FlashPro refer *FlashPro user’s guide*.

Step 6 - Building the Software Application Through SoftConsole

1. From the Libero SoC open the SoftConsole project by double clicking on **Write Application Code** under **Develop Firmware** in **Design Flow** window.



Design Flow

SmartFusion_UART

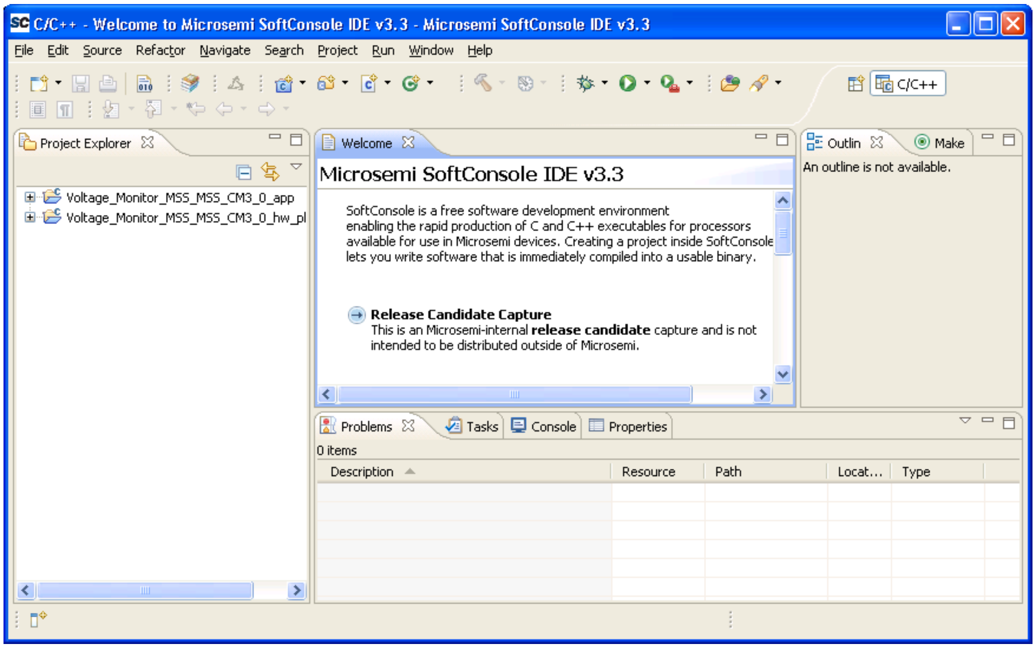
Tool

- ✓ Create Design
  - ✓ Configure MSS
  - Create SmartDesign
  - Create HDL
  - View/Configure Firmware Cores
  - Verify Pre-Synthesized Design
- Constrain Design
  - Import I/O Constraints
  - Import Timing Constraints
- ✓ Implement Design
  - ✓ Synthesize
  - Verify Post-Synthesis Implementation
  - ✓ Compile
  - Constrain Place and Route
  - ✓ Place and Route
  - ✓ Verify Post Layout Implementation
  - Export Back Annotated Files
  - ✓ Generate Programming Data
- Program Design
  - ✓ Program Device
- Debug Design
  - Instrument design
- Handoff Design for Production
  - Export Programming File
  - Export Pin Report
  - Export IBIS Model
- Develop Firmware
  - Write Application Code

Figure 27 · Invoking SoftConsole from Libero SoC

Note: If you are using the provided design files and if SoftConsole opens without a workspace or displays an error when opening the project then refer to www.microsemi.com/soc/kb/article.aspx?id=KI8879.

2. The SoftConsole perspective looks similar to Figure 28 · :



C/C++ - Welcome to Microsemi SoftConsole IDE v3.3 - Microsemi SoftConsole IDE v3.3

Voltage_Monitor_MSS_MSS_CM3_0_app

Voltage_Monitor_MSS_MSS_CM3_0_hw_pl

An outline is not available.

Microsemi SoftConsole IDE v3.3

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.

Release Candidate Capture
This is an Microsemi-internal **release candidate** capture and is not intended to be distributed outside of Microsemi.

0 items

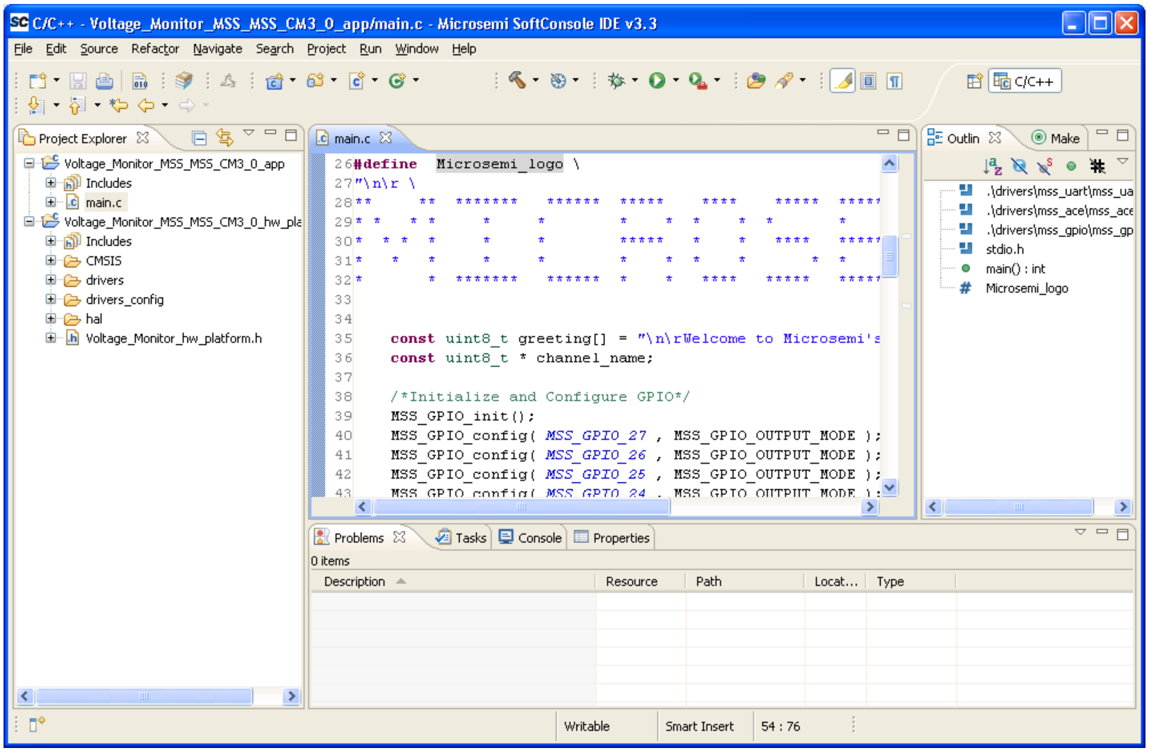
Description	Resource	Path	Locat...	Type

Figure 28 · SoftConsole Workspace

3. Copy the code provided below and paste it in **main.c** file under Voltage_Monitor_MSS_MSS_CM3_0_app project in the SoftConsole editor and delete the existing code.

```
#include "mss_uart.h"
#include "mss_ace.h"
#include "mss_gpio.h"
#include <stdio.h>

#define  Microsemi_logo \
"\n\r \
**     **  *******   ******  *****    ****    *****  ******  **     **  ******* \n\r \
* *   * *     *     *        *    *  *    *  *       *       * *   * *     *    \n\r \
*  * *  *     *     *        *****   *    *   ****   ******  *  * *  *     *    \n\r \
*   *   *     *     *        *    *  *    *       *  *       *   *   *     *    \n\r \
*       *  *******   ******  *    *   ****   *****   ******  *       *  ******* "

int main()
{

        const uint8_t greeting[] = "\n\rWelcome to Microsemi's SmartFusion Voltage
Monitor\n\n\r";
    const uint8_t * channel_name;

    /*Initialize and Configure GPIO*/
    MSS_GPIO_init();
    MSS_GPIO_config( MSS_GPIO_31 , MSS_GPIO_OUTPUT_MODE );
    MSS_GPIO_config( MSS_GPIO_30 , MSS_GPIO_OUTPUT_MODE );
    MSS_GPIO_config( MSS_GPIO_29 , MSS_GPIO_OUTPUT_MODE );
    MSS_GPIO_config( MSS_GPIO_28 , MSS_GPIO_OUTPUT_MODE );

    /*Initialize UART_0*/
    MSS_UART_init(
        &g_mss_uart0,
        MSS_UART_57600_BAUD,
        MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT );

    /*Initialize ACE*/
    ACE_init( );

    MSS_UART_polled_tx_string( &g_mss_uart0, (const uint8_t*)Microsemi_logo );
    MSS_UART_polled_tx( &g_mss_uart0, greeting, sizeof(greeting) );

    channel_name = ACE_get_channel_name( TM0_Voltage );
    for (;;)
    {
        uint8_t display_buffer[32];
        uint16_t adc_result;
        int32_t adc_value_mv;

        adc_result   = ACE_get_ppe_sample( TM0_Voltage );
        adc_value_mv = ACE_convert_to_mV( TM0_Voltage, adc_result );

        if ( adc_value_mv < 0 )
               {
                      snprintf( (char *)display_buffer, sizeof(display_buffer),
               "%s : -%.3fV\r", channel_name, ((float)(-adc_value_mv) / (float)(1000)));
               }
        else
            {
                      snprintf( (char *)display_buffer, sizeof(display_buffer),
               "%s : %.3fV\r", channel_name, ((float)(adc_value_mv) / (float)(1000)));
            }

        MSS_UART_polled_tx_string( &g_mss_uart0, display_buffer );

        /* Checking the status of Voltage flags */
        int32_t flag_status_2p5v = ACE_get_flag_status(TM0_Voltage_over_2p5v);
               int32_t flag_status_2p0v = ACE_get_flag_status(TM0_Voltage_over_2p0v);
               int32_t flag_status_1p5v = ACE_get_flag_status(TM0_Voltage_over_1p5v);
               int32_t flag_status_1p0v = ACE_get_flag_status(TM0_Voltage_over_1p0v);

               /* Voltage flags are displayed on the LEDs through GPIO */
               uint32_t gpio_output;
               if ( flag_status_2p5v == FLAG_ASSERTED )
                      gpio_output = ~(
                                     MSS_GPIO_28_MASK |
                      MSS_GPIO_29_MASK |
                      MSS_GPIO_30_MASK |
                      MSS_GPIO_31_MASK );
               else
               if ( flag_status_2p0v == FLAG_ASSERTED )
                      gpio_output = ~(
                                     MSS_GPIO_28_MASK |
                      MSS_GPIO_29_MASK |
                      MSS_GPIO_30_MASK );
               else
if ( flag_status_1p5v == FLAG_ASSERTED )
                      gpio_output = ~(
                                     MSS_GPIO_28_MASK |
                      MSS_GPIO_29_MASK );
               else
               if ( flag_status_1p0v == FLAG_ASSERTED )
                      gpio_output = ~(
                                     MSS_GPIO_28_MASK );
               else
                      gpio_output = (
                                     MSS_GPIO_28_MASK |
                      MSS_GPIO_29_MASK |
                      MSS_GPIO_30_MASK |
                      MSS_GPIO_31_MASK );

            MSS_GPIO_set_outputs( gpio_output );
      }
    return 0;
}
/**************************************************************************/
```

4. The SoftConsole window looks as shown in Figure 29 · .

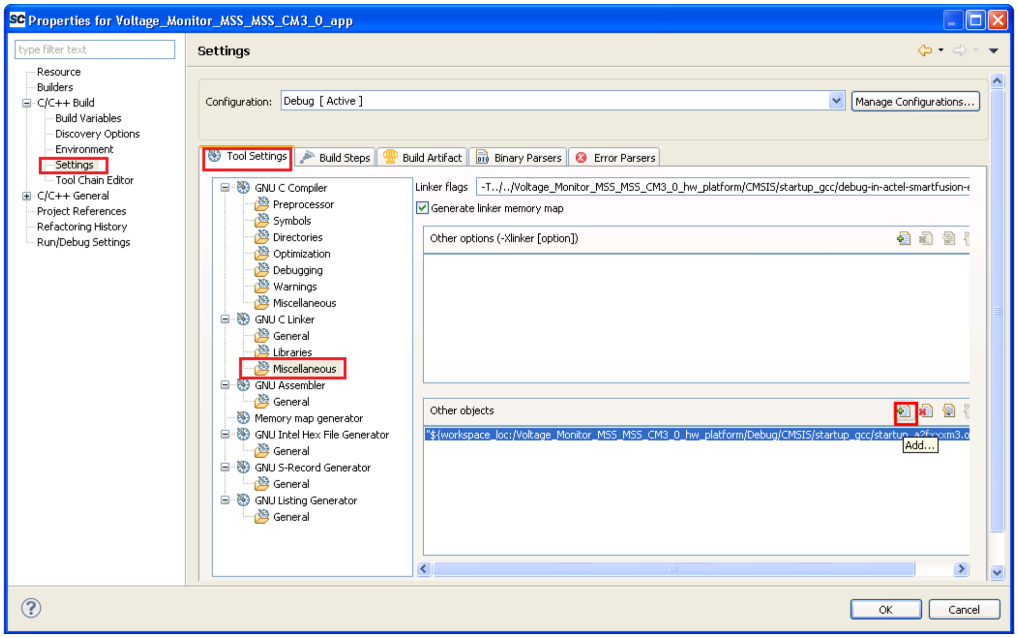


```
#define  Microsemi_logo \
"\n\r \
**     **   *******   ******  *****    ****    *****  *****
* *   * *     *     *        *    *  *    *  *       *
*  * *  *     *     *        *****   *    *   ****   *****
*   *   *     *     *        *    *  *    *       *  *
*       *  *******   ******  *    *   ****   *****   *****


    const uint8_t greeting[] = "\n\rWelcome to Microsemi's
    const uint8_t * channel_name;

    /*Initialize and Configure GPIO*/
    MSS_GPIO_init();
    MSS_GPIO_config( MSS_GPIO_27 , MSS_GPIO_OUTPUT_MODE );
    MSS_GPIO_config( MSS_GPIO_26 , MSS_GPIO_OUTPUT_MODE );
    MSS_GPIO_config( MSS_GPIO_25 , MSS_GPIO_OUTPUT_MODE );
    MSS_GPIO_config( MSS_GPIO_24 , MSS_GPIO_OUTPUT_MODE );
```

Figure 29 · SoftConsole Workspace

5. Right click on Voltage_Monitor_MSS_MSS_CM3_0_app project and select **Properties**.
6. Click **Settings** under **C/C++ Build**.

7. Select **Miscellaneous** under **GNU C Linker** in **Tool Settings** tab as shown in Figure 30 · .



Configuration: Debug [Active]

Linker flags: -T../../Voltage_Monitor_MSS_MSS_CM3_0_hw_platform/CMSIS/startup_gcc/debug-in-actel-smartfusion-

☑ Generate linker memory map

Other objects: "${workspace_loc:/Voltage_Monitor_MSS_MSS_CM3_0_hw_platform/Debug/CMSIS/startup_gcc/startup_a2fxxxm3.o

Figure 30 · Properties Window

8. Click **Add** in the **Other objects** section. The **Add file path** dialog window appears as shown in Figure 31 · .

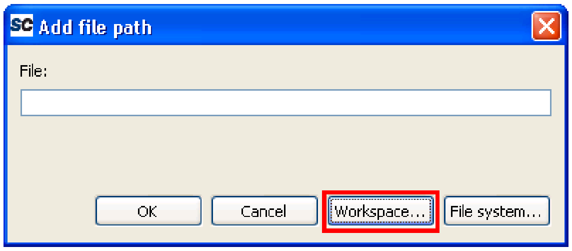
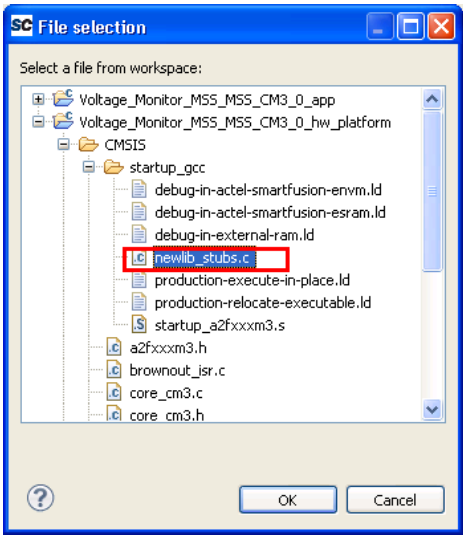


Figure 31 · Add File Path Window

9. Click **Workspace** and add the **newlib_stubs.c** file under **Voltage_Monitor_MSS_MSS**_CM3_0_**hw_platform> CMSIS > startup_gcc> newlib_stubs.c** as shown in Figure 32 · .



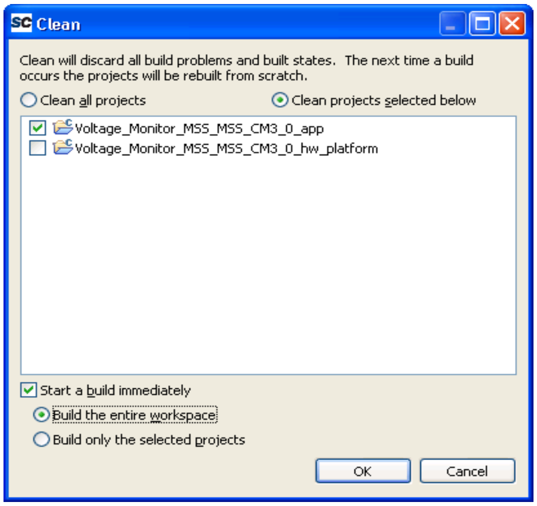
File selection

Select a file from workspace:

- Voltage_Monitor_MSS_MSS_CM3_0_app
- Voltage_Monitor_MSS_MSS_CM3_0_hw_platform
  - CMSIS
    - startup_gcc
      - debug-in-actel-smartfusion-envm.ld
      - debug-in-actel-smartfusion-esram.ld
      - debug-in-external-ram.ld
      - newlib_stubs.c
      - production-execute-in-place.ld
      - production-relocate-executable.ld
      - startup_a2fxxxm3.s
    - a2fxxxm3.h
    - brownout_isr.c
    - core_cm3.c
    - core_cm3.h

Figure 32 · File Selection Window

10. Click **OK** in properties window.
11. Perform a clean build by selecting **Project** > **Clean**. Accept the default settings in the **Clean** dialog box and click **OK**.



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

☑ Voltage_Monitor_MSS_MSS_CM3_0_app
□ Voltage_Monitor_MSS_MSS_CM3_0_hw_platform

☑ Start a build immediately
● Build the entire workspace
○ Build only the selected projects

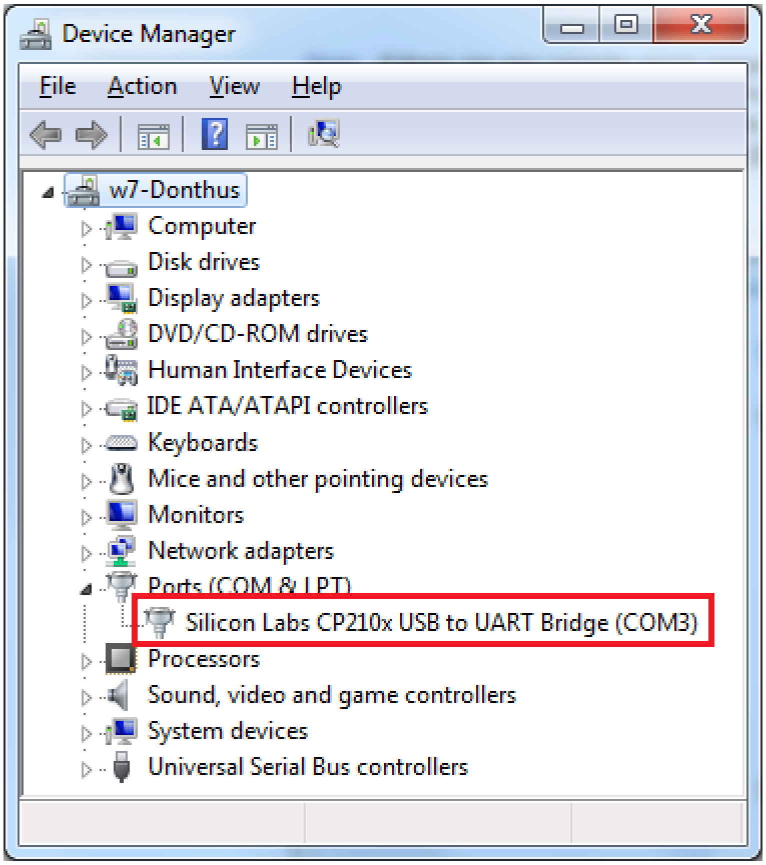
Figure 33 · Settings for a Clean Build

12. Make sure there are no errors and warnings. Use the next steps to configure the HyperTerminal.

Step 7 - Configuring Serial Terminal Emulation Program

Prior to running the application program, you need to configure the terminal emulator program (HyperTerminal, included with Windows®) on your PC. Perform the following steps to use the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board:

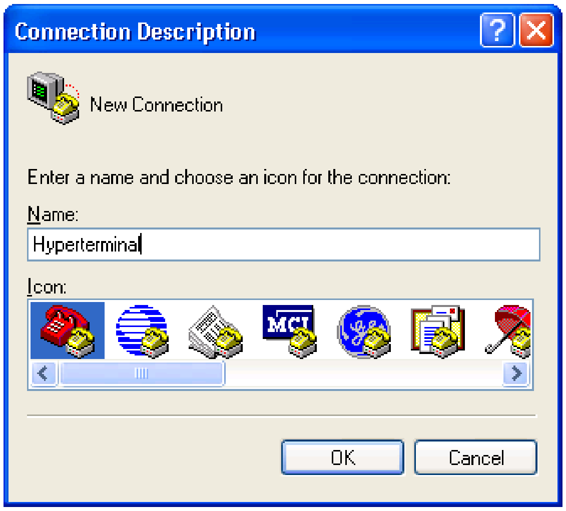
1. Connect a second mini USB cable between the USB connector on the SmartFusion Evaluation Kit Board (or the SmartFusion Development Kit Board) and a USB port of your PC. If Windows prompts you to connect to Windows Update, select **No**, **not at this time** and click **Next**.
2. If the Silicon Labs CP210x USB to UART Bridge drivers are automatically detected (this can be verified in Device Manager), as shown in Figure 34 · , proceed to the next step; otherwise follow the
3. Step 8 - Installing Drivers for the USB to RS232 Bridge to install drivers for USB to RS232 Bridge.



w7-Donthus
- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - Silicon Labs CP210x USB to UART Bridge (COM3)
- Processors
- Sound, video and game controllers
- System devices
- Universal Serial Bus controllers

Figure 34 · Device Manager Listing Silicon Labs CP210x USB to UART Bridge Drivers

4. From the Windows **Start** menu, select **Programs > Accessories > Communications > HyperTerminal.** This opens HyperTerminal. If your computer does not have HyperTerminal, use any free serial terminal emulation program like PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.
5. Enter **Hyperterminal** in the **Name** field in the **Connection Description** dialog box and click **OK**.



New Connection

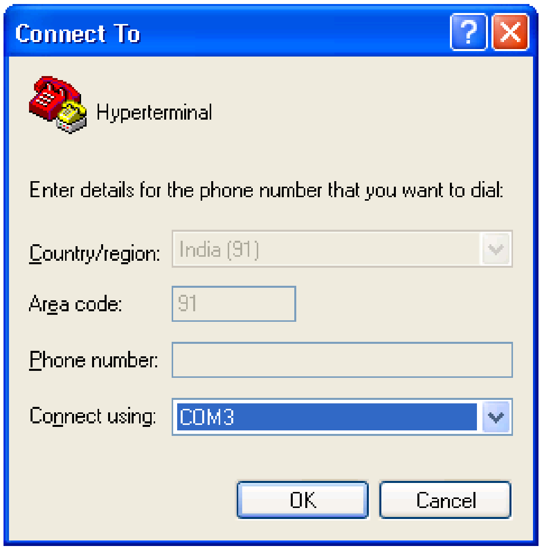
Enter a name and choose an icon for the connection:

Name: Hyperterminal

Icon:

Figure 35 · New Connection

6. Select the appropriate COM port (to which USB-Rs232 drivers are pointed) from the **Connect using** drop-down list and click **OK**.



Hyperterminal

Enter details for the phone number that you want to dial:

Country/region: India (91)

Area code: 91

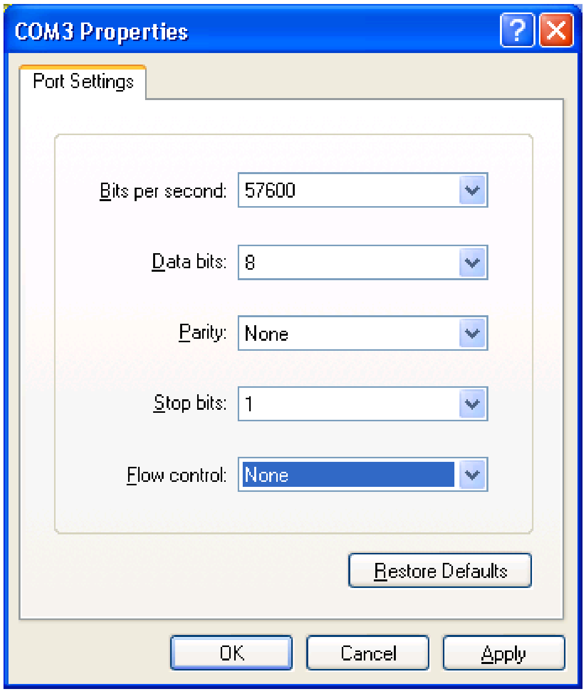
Phone number:

Connect using: COM3

Figure 36 · Selecting the COM Port

7. Set the following in the **COM Properties** window and click **OK**:

- Bits per second: 57600
- Data bits: 8
- Parity: None
- Stop Bits: 1
- Flow control: None



Bits per second: 57600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Figure 37 · Setting the COM Properties

8. Click **OK** to close the Hyperterminal Properties dialog box.

Next time you can directly open HyperTerminal (without configuring) by selecting **Programs > Accessories > Communications > HyperTerminal > Hyperterminal**.

Step 8 - Installing Drivers for the USB to RS232 Bridge

Note: You must have full administrative rights for your system to install the USB-RS232 drivers.

1. Download the USB to RS232 bridge drivers from

www.microsemi.com/soc/documents/CP2102_driver.zip.

2. Unzip the CP2102_driver.zip file.

3. Double-click (Run) the CP210x_VCP_Win_XP_S2K3_Vista_7.exe file.

4. Accept the default installation location and click Install.

5. Click Continue Anyway if prompted.

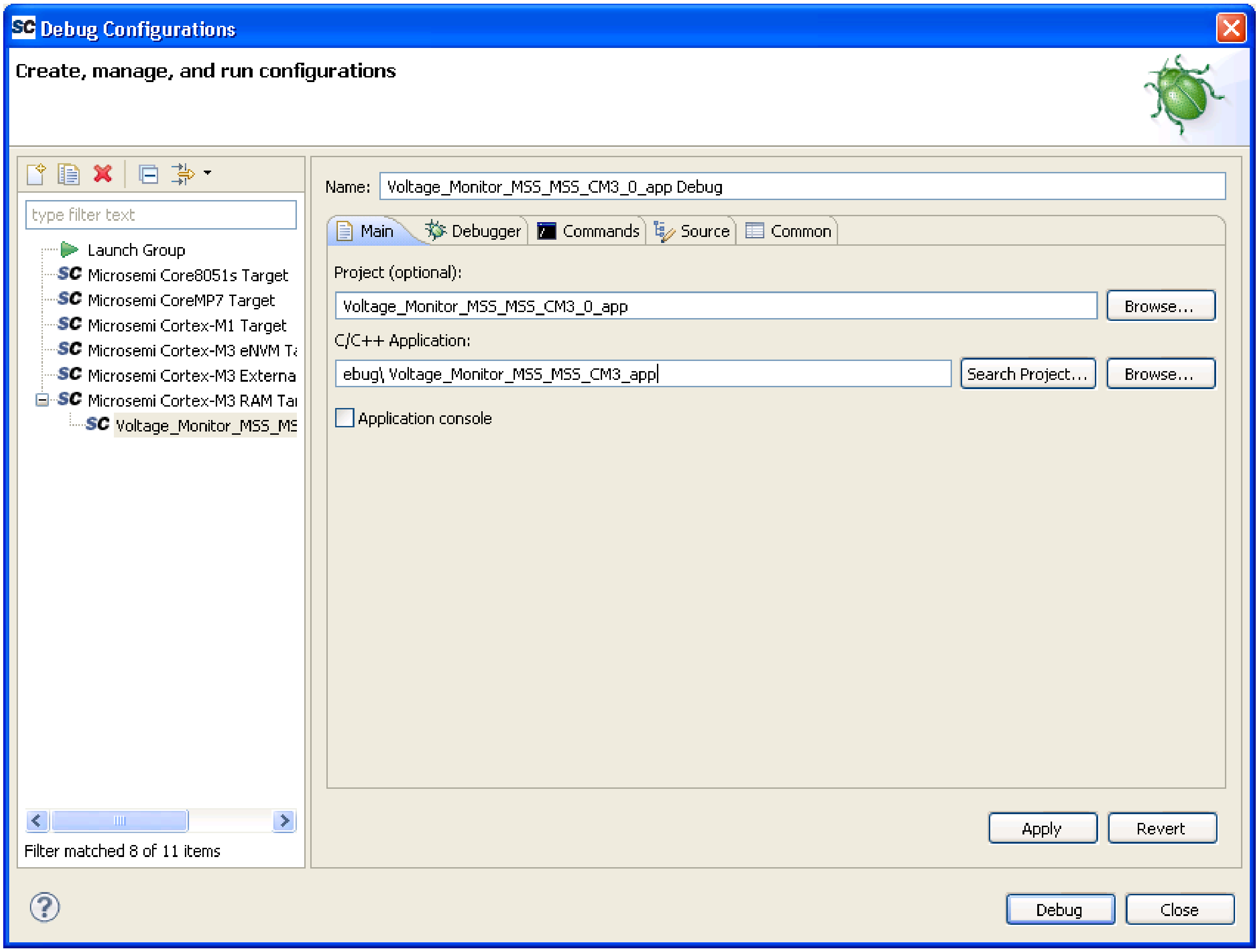
6. When the installation is complete, click OK. The Ports (COM & LPT) section of the Device Manager lists Silicon Labs CP210x USB to UART Bridge under the Ports section of Device Manager.

Use the following steps to install drivers for the USB to RS232 Bridge:

Step 9 - Debugging the Application Project using SoftConsole

Use the following steps to debug the application project using SoftConsole:

1. Select **Debug Configurations** from the **Run** menu of the SoftConsole. The Debug dialog is displayed.
2. Double clicking on **Microsemi Cortex-M3 RAM target** displays an image similar to Figure 38 · :



Debug Configurations

Create, manage, and run configurations

Name: Voltage_Monitor_MSS_MSS_CM3_0_app Debug

Project (optional): Voltage_Monitor_MSS_MSS_CM3_0_app

C/C++ Application: ebug\ Voltage_Monitor_MSS_MSS_CM3_app

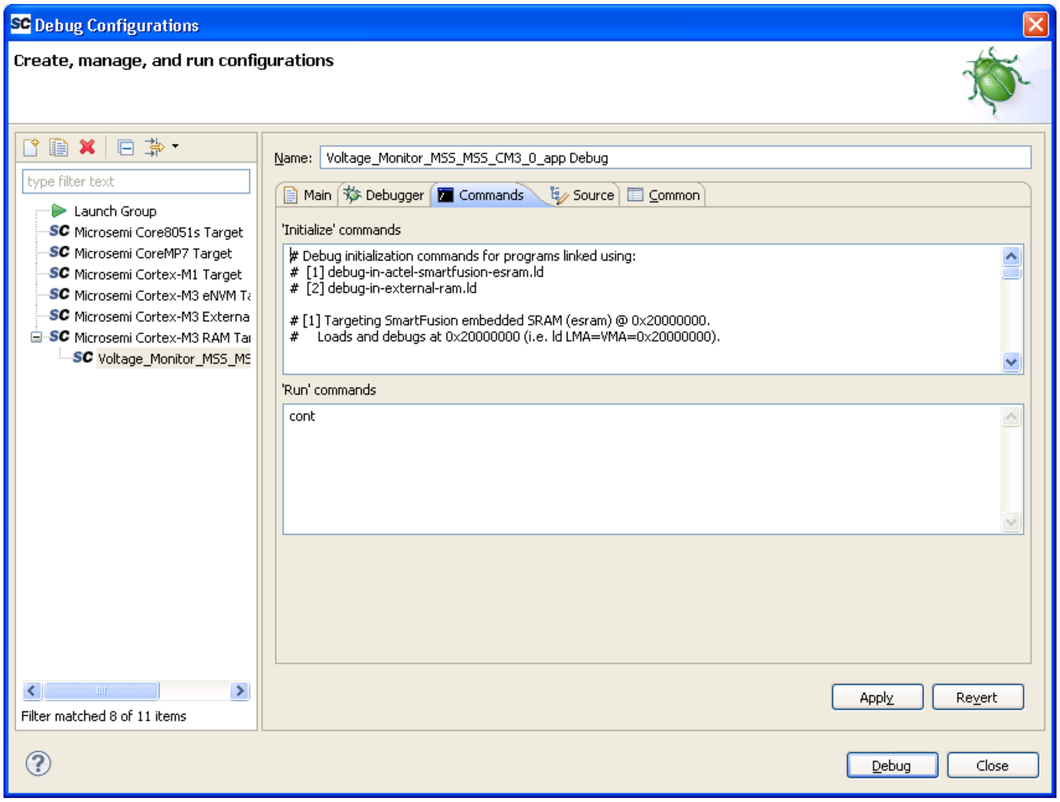
□ Application console

Filter matched 8 of 11 items

Figure 38 · Debug Window

3. Confirm that the following appear on the **Main** tab in the Debug window:
   - Name: Voltage_Monitor_MSS_MSS_CM3_0_app Debug
   - Project: Voltage_Monitor_MSS_MSS_CM3_0_app
   - C/C++ application: Debug\ Voltage_Monitor_MSS_MSS_CM3_0_app

4. Select the **Commands** tab. Confirm that commands appear in the Initialize and Run command sections, as shown in Figure 39 · :



Launch Group
Microsemi Core8051s Target
Microsemi CoreMP7 Target
Microsemi Cortex-M1 Target
Microsemi Cortex-M3 eNVM Ta
Microsemi Cortex-M3 Externa
Microsemi Cortex-M3 RAM Tar
Voltage_Monitor_MSS_MS

Filter matched 8 of 11 items

Name: Voltage_Monitor_MSS_MSS_CM3_0_app Debug

'Initialize' commands

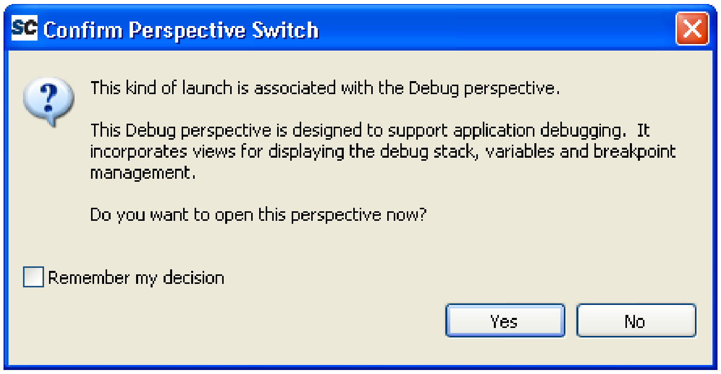
```
# Debug initialization commands for programs linked using:
#  [1] debug-in-actel-smartfusion-esram.ld
#  [2] debug-in-external-ram.ld

# [1] Targeting SmartFusion embedded SRAM (esram) @ 0x20000000.
#     Loads and debugs at 0x20000000 (i.e. ld LMA=VMA=0x20000000).
```

'Run' commands

```
cont
```

Figure 39 · Debugger Commands

5. Click **Apply** and **Debug**.
6. Click **Yes** when prompted for **Confirm Perspective Switch**. This displays the debug view mode.



This kind of launch is associated with the Debug perspective.

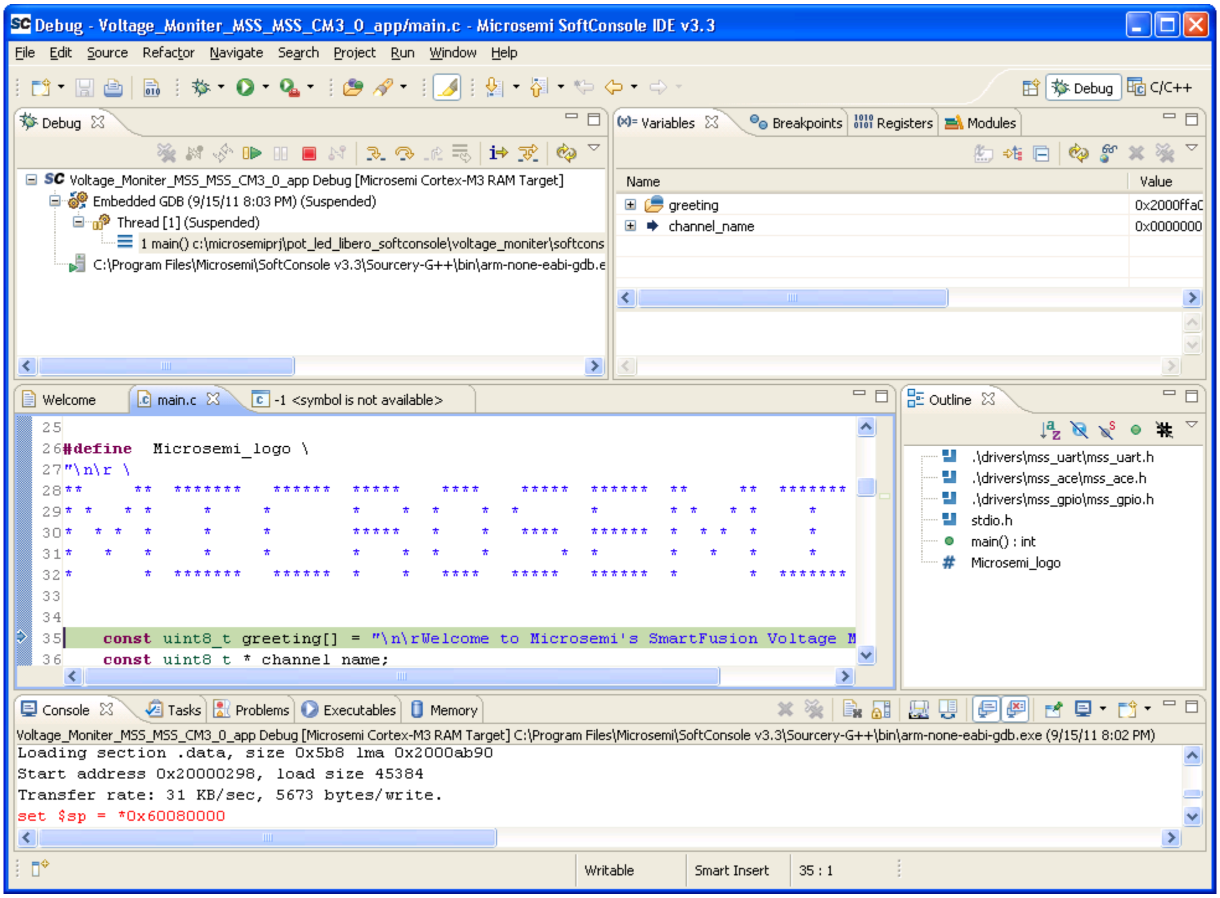
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

□ Remember my decision

Figure 40 · Confirm Perspective Switch

7. Your Debug Prospective should resemble Figure 41 · :



Debug - Voltage_Moniter_MSS_MSS_CM3_0_app/main.c - Microsemi SoftConsole IDE v3.3
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
C/C++
Debug
Variables
Breakpoints
Registers
Modules
Welcome
main.c
-1 <symbol is not available>
Outline
.\drivers\mss_uart\mss_uart.h
.\drivers\mss_ace\mss_ace.h
.\drivers\mss_gpio\mss_gpio.h
stdio.h
main() : int
Microsemi_logo
Console
Tasks
Problems
Executables
Memory
Writable
Smart Insert
35 : 1

Voltage_Moniter_MSS_MSS_CM3_0_app Debug [Microsemi Cortex-M3 RAM Target]
Embedded GDB (9/15/11 8:03 PM) (Suspended)
Thread [1] (Suspended)
1 main() c:\microsemiprj\pot_led_libero_softconsole\voltage_moniter\softcons
C:\Program Files\Microsemi\SoftConsole v3.3\Sourcery-G++\bin\arm-none-eabi-gdb.e

Name	Value
greeting	0x2000ffaC
channel_name	0x0000000

```
#define  Microsemi_logo \
"\n\r \
**     **  *******   ******  *****    ****    *****   ******  **     **  *******
* *   * *     *     *        *    *   *    *  *       *       * *   * *     *
*  * *  *     *     *        *****    *    *  ****    ******  *  * *  *     *
*   *   *     *     *        *    *   *    *       *  *       *   *   *     *
*       *  *******   ******  *    *   ****    *****   ******  *       *  *******


    const uint8_t greeting[] = "\n\rWelcome to Microsemi's SmartFusion Voltage M
    const uint8_t * channel_name;
```

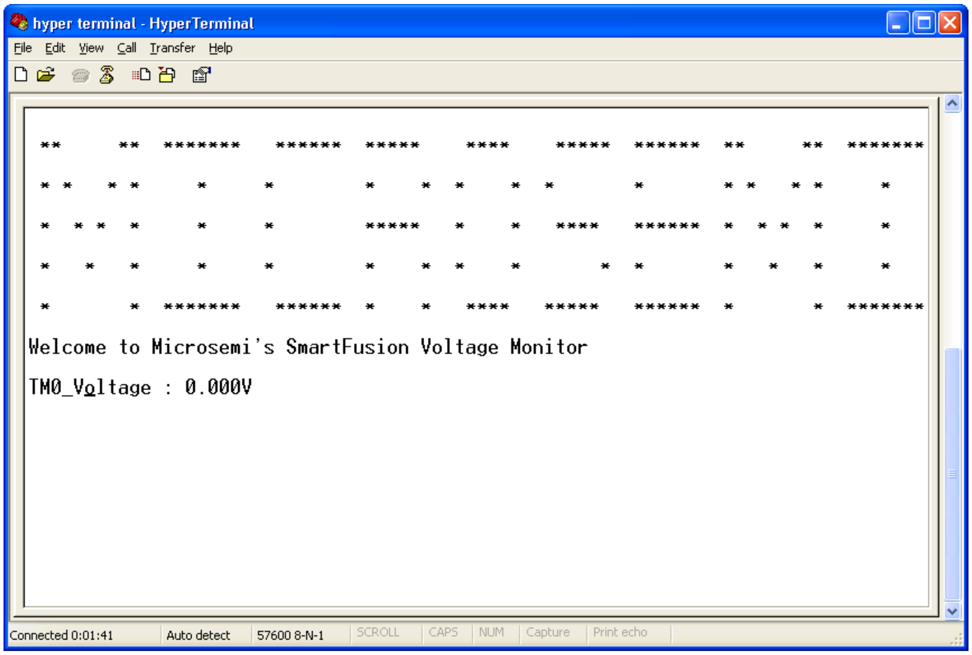
```
Voltage_Moniter_MSS_MSS_CM3_0_app Debug [Microsemi Cortex-M3 RAM Target] C:\Program Files\Microsemi\SoftConsole v3.3\Sourcery-G++\bin\arm-none-eabi-gdb.exe (9/15/11 8:02 PM)
Loading section .data, size 0x5b8 lma 0x2000ab90
Start address 0x20000298, load size 45384
Transfer rate: 31 KB/sec, 5673 bytes/write.
set $sp = *0x60080000
```

Figure 41 · Debug Perspective

8. Run the application by clicking **Run > Resume** or by clicking the **Run** icon on the SoftConsole toolbar. The voltage measurement along with the greeting message is displayed in the terminal program window.

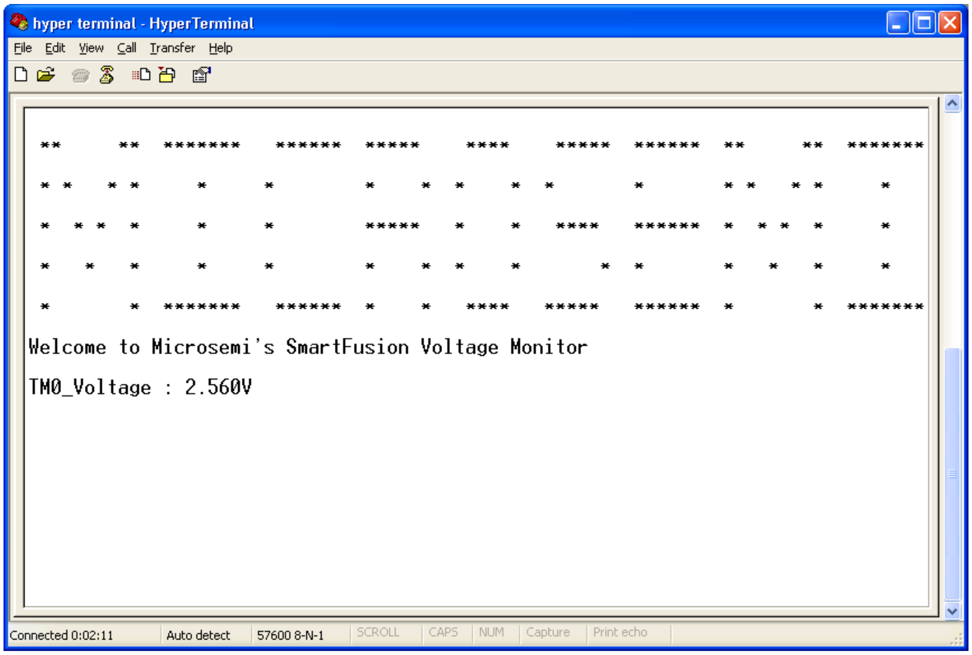
9. Turn the potentiometer (POT) on the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board. The voltage measurement will be displayed on HyperTerminal and the LEDs on the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board will illuminate when one of the voltage monitor flags is asserted.

10. Adjust the **POT** and observe that the voltage measurement is continuously updated.



```
 **     **  *******   ******  *****    ****    *****  ******  **     **  *******

 * *   * *     *     *        *    *  *    *  *       *       * *   * *     *

 *  * *  *     *     *        *****   *    *   ****   ******  *  * *  *     *

 *   *   *     *     *        *    *  *    *       *  *       *   *   *     *

 *       *  *******   ******  *    *   ****   *****   ******  *       *  *******

Welcome to Microsemi's SmartFusion Voltage Monitor

TM0_Voltage : 0.000V
```

Figure 42 · Voltage Monitor

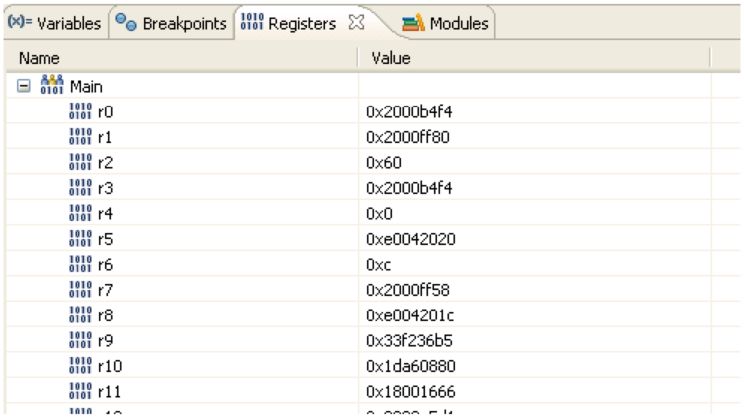


```
 **     **  *******   ******  *****    ****    *****  ******  **     **  *******

 * *   * *     *     *        *    *  *    *  *       *       * *   * *     *

 *  * *  *     *     *        *****   *    *   ****   ******  *  * *  *     *

 *   *   *     *     *        *    *  *    *       *  *       *   *   *     *

 *       *  *******   ******  *    *   ****   *****   ******  *       *  *******

Welcome to Microsemi's SmartFusion Voltage Monitor

TM0_Voltage : 2.560V
```

Figure 43 · Voltage Measurement Continuous Update

11. Observe the state of the LEDs as the POT is adjusted. Confirm that the flags work as specified in the ACE configurator.
12. Suspend the software application by clicking **Run > Suspend** from the SoftConsole menu.

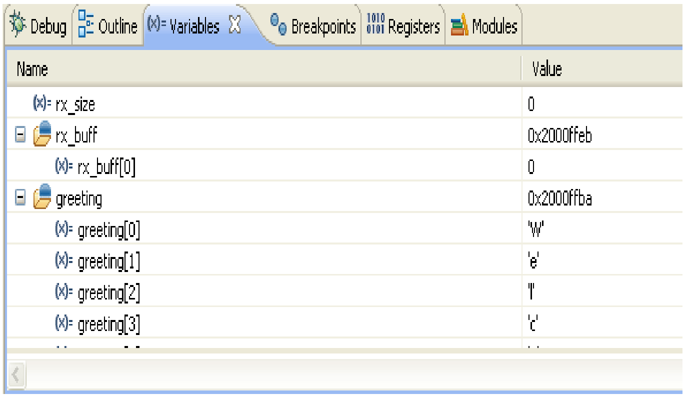
13. Select the **Registers** tab on the upper window pane to view the value of the Cortex-M3 processor internal registers.



Name	Value
Main	
r0	0x2000b4f4
r1	0x2000ff80
r2	0x60
r3	0x2000b4f4
r4	0x0
r5	0xe0042020
r6	0xc
r7	0x2000ff58
r8	0xe004201c
r9	0x33f236b5
r10	0x1da60880
r11	0x18001666

Figure 44 · The Registers Tab

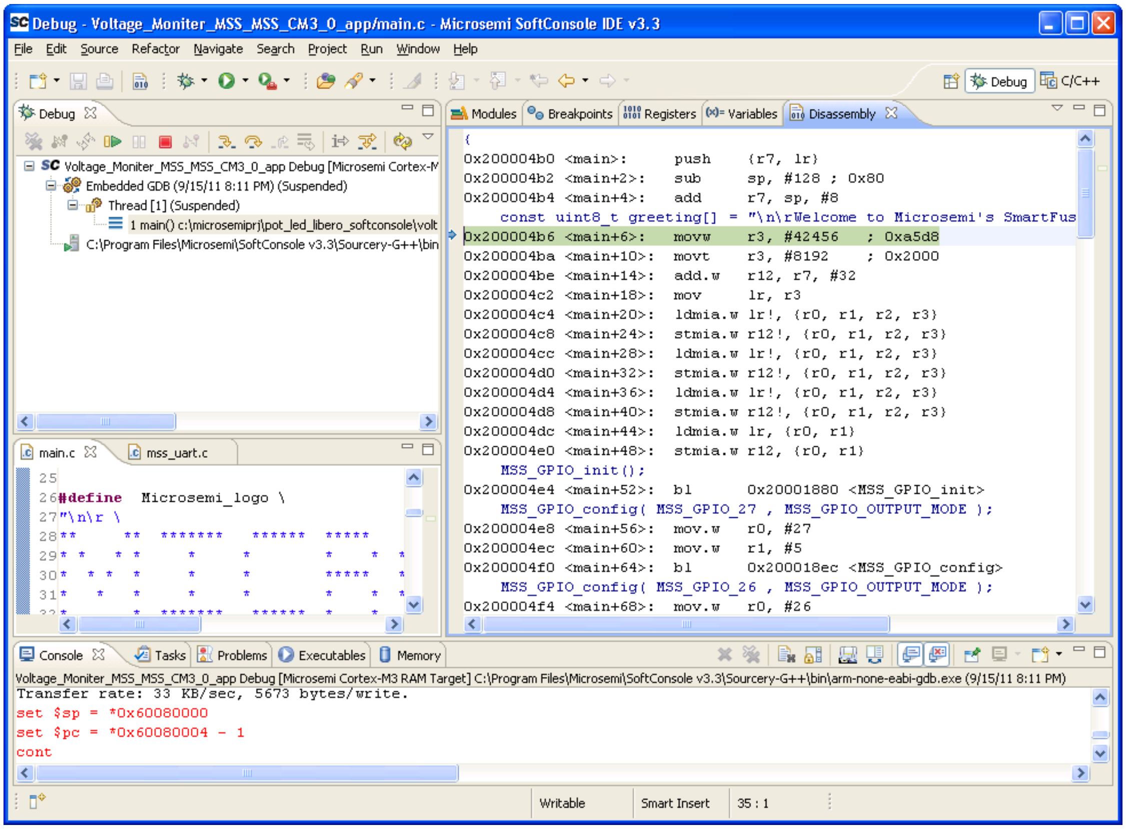
14. Select the **Variables** tab in the upper left window pane to view the value of variables in the source code.



Name	Value
rx_size	0
rx_buff	0x2000ffeb
rx_buff[0]	0
greeting	0x2000ffba
greeting[0]	'W'
greeting[1]	'e'
greeting[2]	'l'
greeting[3]	'c'

Figure 45 · The Variables Tab

15. Choose **Window > Show View > Disassembly** to display the assembly level instructions. The Assembly window is displayed on the right side of the Debug perspective.


Debug - Voltage_Moniter_MSS_MSS_CM3_0_app/main.c - Microsemi SoftConsole IDE v3.3
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
C/C++
Debug
Modules
Breakpoints
Registers
Variables
Disassembly
main.c
mss_uart.c
Console
Tasks
Problems
Executables
Memory
Writable
Smart Insert
35 : 1

```
Voltage_Moniter_MSS_MSS_CM3_0_app Debug [Microsemi Cortex-M
  Embedded GDB (9/15/11 8:11 PM) (Suspended)
    Thread [1] (Suspended)
      1 main() c:\microsemiprj\pot_led_libero_softconsole\volt
  C:\Program Files\Microsemi\SoftConsole v3.3\Sourcery-G++\bin
```

```
#define  Microsemi_logo \
"\n\r \
**     **   *******   ******  *****
* *   * *     *     *        *    *  *
*  * *  *     *     *        *****   *
*   *   *     *     *        *    *  *
```

```
{
0x200004b0 <main>:      push    {r7, lr}
0x200004b2 <main+2>:    sub     sp, #128 ; 0x80
0x200004b4 <main+4>:    add     r7, sp, #8
    const uint8_t greeting[] = "\n\rWelcome to Microsemi's SmartFus
0x200004b6 <main+6>:    movw    r3, #42456    ; 0xa5d8
0x200004ba <main+10>:   movt    r3, #8192     ; 0x2000
0x200004be <main+14>:   add.w   r12, r7, #32
0x200004c2 <main+18>:   mov     lr, r3
0x200004c4 <main+20>:   ldmia.w lr!, {r0, r1, r2, r3}
0x200004c8 <main+24>:   stmia.w r12!, {r0, r1, r2, r3}
0x200004cc <main+28>:   ldmia.w lr!, {r0, r1, r2, r3}
0x200004d0 <main+32>:   stmia.w r12!, {r0, r1, r2, r3}
0x200004d4 <main+36>:   ldmia.w lr!, {r0, r1, r2, r3}
0x200004d8 <main+40>:   stmia.w r12!, {r0, r1, r2, r3}
0x200004dc <main+44>:   ldmia.w lr, {r0, r1}
0x200004e0 <main+48>:   stmia.w r12, {r0, r1}
    MSS_GPIO_init();
0x200004e4 <main+52>:   bl      0x20001880 <MSS_GPIO_init>
    MSS_GPIO_config( MSS_GPIO_27 , MSS_GPIO_OUTPUT_MODE );
0x200004e8 <main+56>:   mov.w   r0, #27
0x200004ec <main+60>:   mov.w   r1, #5
0x200004f0 <main+64>:   bl      0x200018ec <MSS_GPIO_config>
    MSS_GPIO_config( MSS_GPIO_26 , MSS_GPIO_OUTPUT_MODE );
0x200004f4 <main+68>:   mov.w   r0, #26
```

```
Voltage_Moniter_MSS_MSS_CM3_0_app Debug [Microsemi Cortex-M3 RAM Target] C:\Program Files\Microsemi\SoftConsole v3.3\Sourcery-G++\bin\arm-none-eabi-gdb.exe (9/15/11 8:11 PM)
Transfer rate: 33 KB/sec, 5673 bytes/write.
set $sp = *0x60080000
set $pc = *0x60080004 - 1
cont
```

Figure 46 · Assembly Window

16. You can single-step through the source code by choosing **Run > Step Into** or **Run > Step Over** or by clicking the **Step Into** or **Step Over** icons. Observe the changes in the source code window and Disassembly view. Performing a Step Over allows for stepping over functions. The entire function is executed but there is no need to single step through each instruction contained in the function.
17. Click the **Instruction Stepping** icon and then perform **Step Into** operations. Observe that **Step Into** now executes a single line of assembly code.
18. Click the **Instruction Stepping** icon to exit the instruction stepping mode. Single-step through the application and observe the instruction sequence in the source code window in the middle of the Debug perspective, and the values of the variables and registers.
19. Resume execution of the code by choosing **Run > Resume** or by clicking the **Resume** icon. You can even add breakpoints in the application for further debugging.

20. Once you made the required changes, terminate the debugger by selecting Voltage_Monitor_MSS_MSS_CM3_0_app Debug in the Debug view, then right-clicking and selecting **Terminate and Remove**.

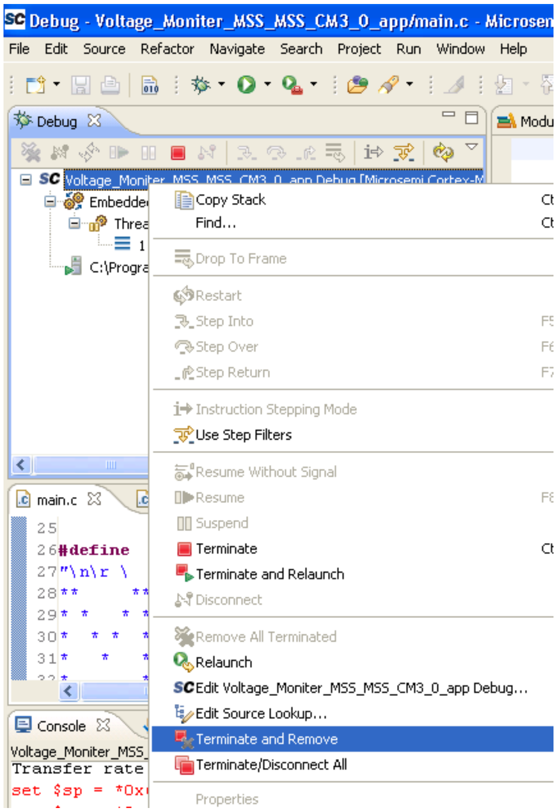


Figure 47 · Terminating the Program

21. Close the Debug perspective by selecting **Close Perspective** from the **Window** menu.
22. Close the voltage monitor project by selecting the project name in the SoftConsole Project Explorer view, right–clicking, and selecting **Close Project**.
23. Close SoftConsole using **File > Exit**.
24. Close HyperTerminal using **File > Exit**. Click **Yes** when prompted for closing immediately.

Step 10 - Building Executable Image in Release mode

You can build an application executable image in “release mode” and load it into eNVM for executing code in eNVM of SmartFusion cSoC device. You can load the application executable image into eNVM with the help of eNVM data storage client from SmartDesign MSS Configurator and in-application programming (IAP) or FlashPro programming software. In release mode, you cannot use SoftConsole debugger to load the executable image into eNVM.

For steps to build an executable image for our application refer the tutorial SmartFusion: Building Executable Image in Release Mode and Loading into eNVM.

This concludes the tutorial.

Appendix A – Libero SoC Catalog Settings

Listed below are the steps to show how to configure your vault location and set up the repositories in Libero SoC.

1. On the **Catalog** window, click **Options**.

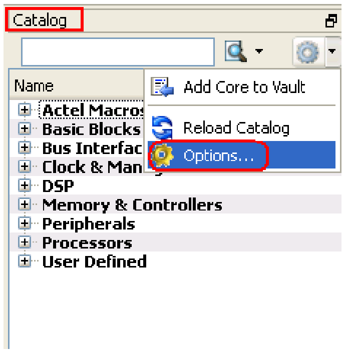
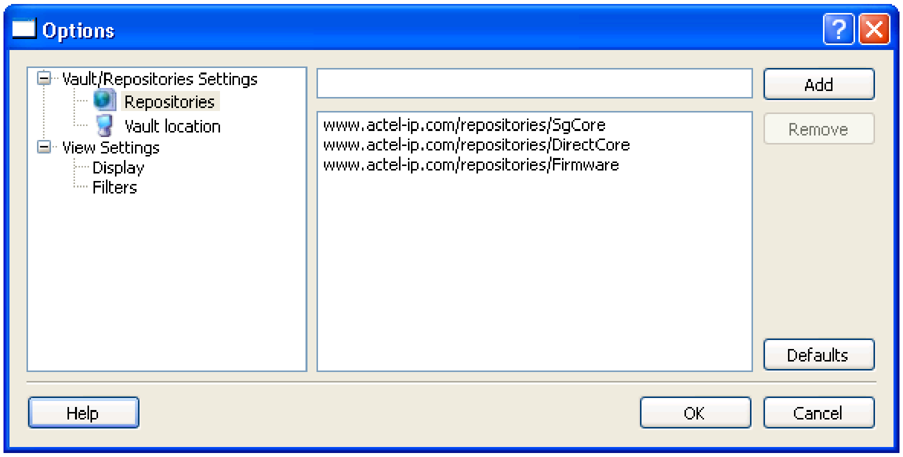


Figure 48 · Catalog – Options

2. The **Options** window is displayed. Click **Repositories** under **Vault/Reposieteries Settings** and add the following in the address field:
   - www.actel-ip.com/repositories/SgCore
   - www.actel-ip.com/repositories/DirectCore
   - www.actel-ip.com/repositories/Firmware

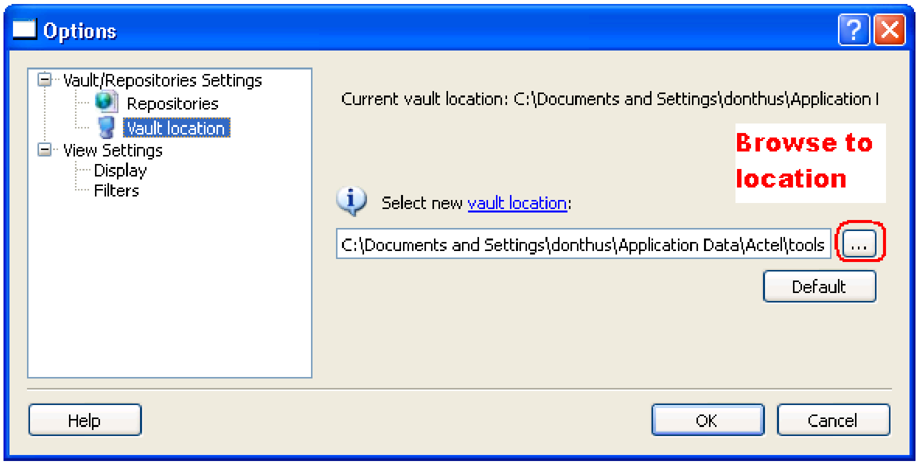
Note: Click **Add** after entering each path.



www.actel-ip.com/repositories/SgCore
www.actel-ip.com/repositories/DirectCore
www.actel-ip.com/repositories/Firmware

Figure 49 · Setting Repositories

3. Click on **Vault location** under **Vault/Repositeries Settings** the **Options** window. Browse to a location on your PC to set the vault location where the IPs can be downloaded from the repositories.



Current vault location: C:\Documents and Settings\donthus\Application I

Select new vault location:

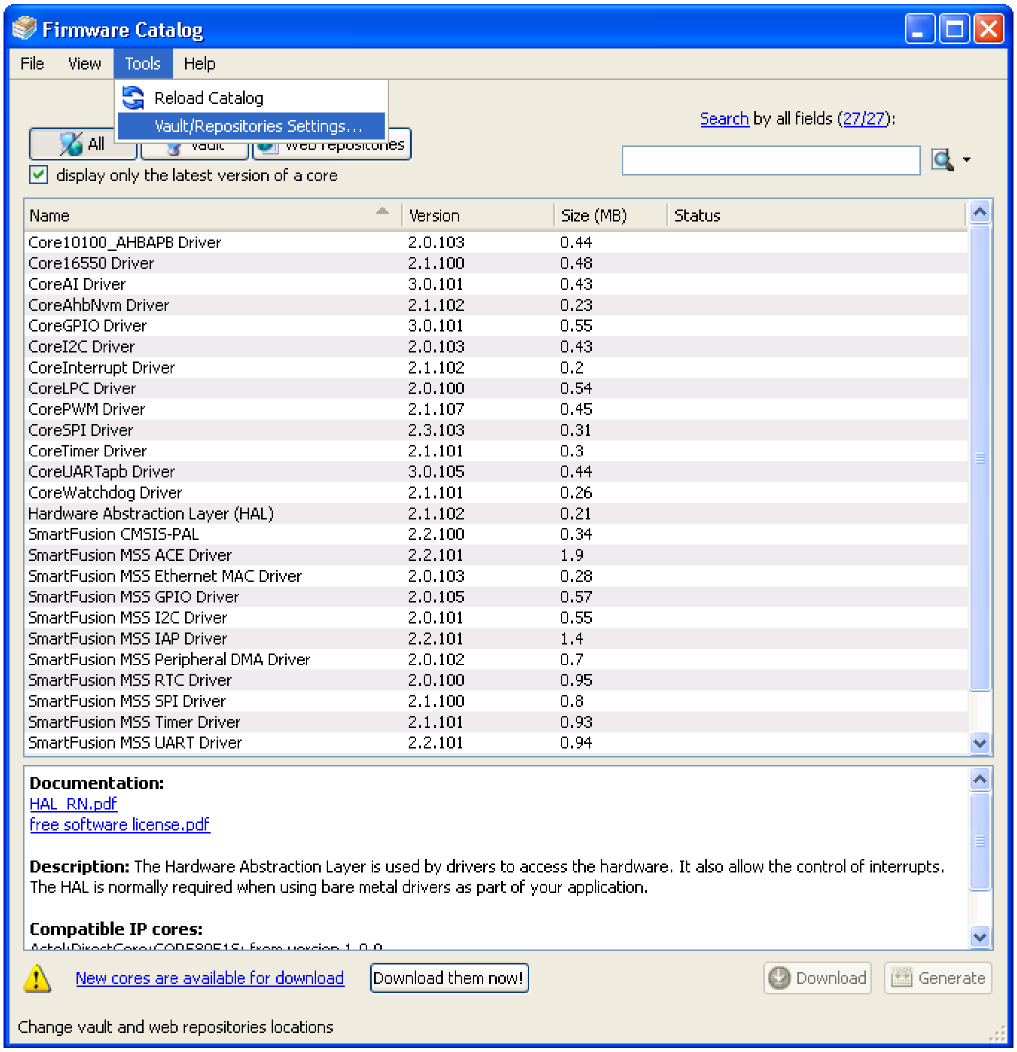
C:\Documents and Settings\donthus\Application Data\Actel\tools

Figure 50 · Setting the Vault Location

4. Click **OK**.

Appendix B – Firmware Catalog Settings

1. Open the **<Libero Installation directory>\Designer\bin\catalog.exe**.
2. Select **Tools > Vault/Repositories Settings**, from the Firmware Catalog widow.



☑ display only the latest version of a core

Name	Version	Size (MB)	Status
Core10100_AHBAPB Driver	2.0.103	0.44	
Core16550 Driver	2.1.100	0.48	
CoreAI Driver	3.0.101	0.43	
CoreAhbNvm Driver	2.1.102	0.23	
CoreGPIO Driver	3.0.101	0.55	
CoreI2C Driver	2.0.103	0.43	
CoreInterrupt Driver	2.1.102	0.2	
CoreLPC Driver	2.0.100	0.54	
CorePWM Driver	2.1.107	0.45	
CoreSPI Driver	2.3.103	0.31	
CoreTimer Driver	2.1.101	0.3	
CoreUARTapb Driver	3.0.105	0.44	
CoreWatchdog Driver	2.1.101	0.26	
Hardware Abstraction Layer (HAL)	2.1.102	0.21	
SmartFusion CMSIS-PAL	2.2.100	0.34	
SmartFusion MSS ACE Driver	2.2.101	1.9	
SmartFusion MSS Ethernet MAC Driver	2.0.103	0.28	
SmartFusion MSS GPIO Driver	2.0.105	0.57	
SmartFusion MSS I2C Driver	2.0.101	0.55	
SmartFusion MSS IAP Driver	2.2.101	1.4	
SmartFusion MSS Peripheral DMA Driver	2.0.102	0.7	
SmartFusion MSS RTC Driver	2.0.100	0.95	
SmartFusion MSS SPI Driver	2.1.100	0.8	
SmartFusion MSS Timer Driver	2.1.101	0.93	
SmartFusion MSS UART Driver	2.2.101	0.94	

Documentation:
HAL_RN.pdf
free software license.pdf

Description: The Hardware Abstraction Layer is used by drivers to access the hardware. It also allow the control of interrupts. The HAL is normally required when using bare metal drivers as part of your application.

Compatible IP cores:
Actel:DirectCore:CORE8051S: from version 1.0.0

⚠ New cores are available for download Download them now!

Change vault and web repositories locations

Figure 51 · Firmware Catalog Settings

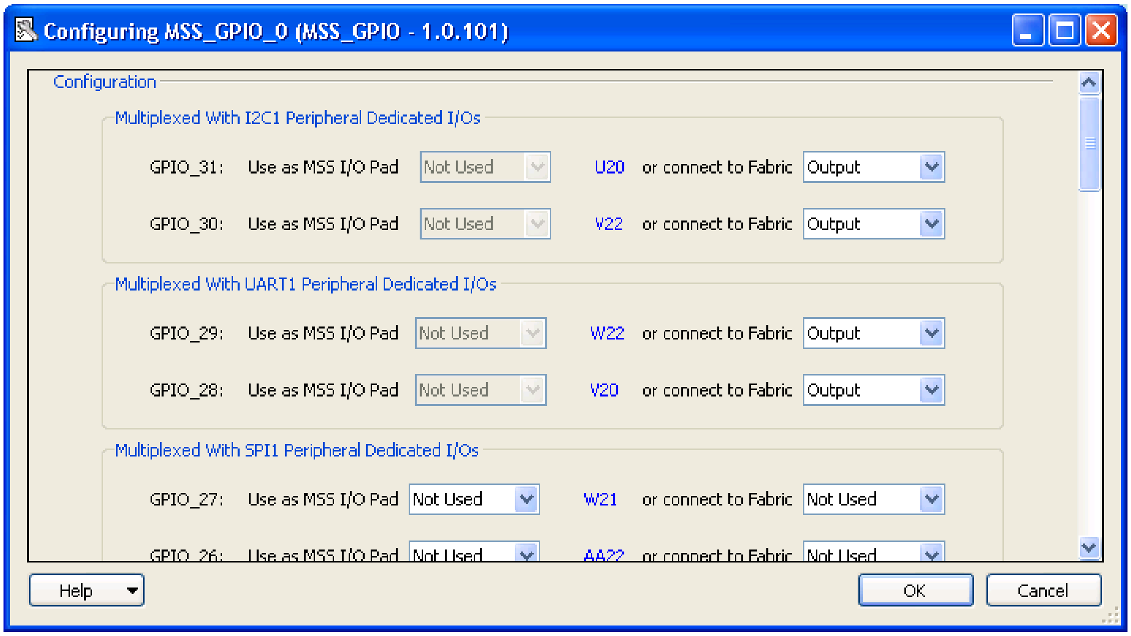
3. Select **Repositories** under **Vault/Repositories Settings** in the **Options** dialog box.
4. Confirm that the following repositories are displayed (add them if needed):
   - www.actel-ip.com/repositories/SgCore
   - www.actel-ip.com/repositories/DirectCore
   - www.actel-ip.com/repositories/Firmware
5. Add the above mentioned paths in the address field if required by selecting the repository and clicking **Add**.

If new cores are available for download, click **Download them now!** to download the new cores to the vault.

Appendix – C

Configuring the GPIO Peripheral

1. Double-click the **GPIO block** in the **MSS component**, configure as shown in Figure 52 · , and click **OK**.



Configuring MSS_GPIO_0 (MSS_GPIO - 1.0.101)

Configuration

Multiplexed With I2C1 Peripheral Dedicated I/Os

GPIO_31: Use as MSS I/O Pad Not Used U20 or connect to Fabric Output

GPIO_30: Use as MSS I/O Pad Not Used V22 or connect to Fabric Output

Multiplexed With UART1 Peripheral Dedicated I/Os

GPIO_29: Use as MSS I/O Pad Not Used W22 or connect to Fabric Output

GPIO_28: Use as MSS I/O Pad Not Used V20 or connect to Fabric Output

Multiplexed With SPI1 Peripheral Dedicated I/Os

GPIO_27: Use as MSS I/O Pad Not Used W21 or connect to Fabric Not Used

GPIO_26: Use as MSS I/O Pad Not Used AA22 or connect to Fabric Not Used

Figure 52 · Configure MSS_GPIO_0

This example requires GPIO_31, GPIO_30, GPIO_29, and GPIO_28 to be connected to LED_4 to LED_1 on the SmartFusion Evaluation Kit Board, and D4 to D1 on the SmartFusion Development Kit Board. These signals will be routed through the fabric to I/O pins H17, C19, B20, and B19, respectively.

2. Click **File > Save** to save the Voltage_Monitor_MSS.

Generating the MSS Component

1. Right-click on Voltage_Monitor_MSS_0 component on the **Voltage_Monitor** tab and select Update Instance(s) with Latest Component as shown in Figure 53 · .

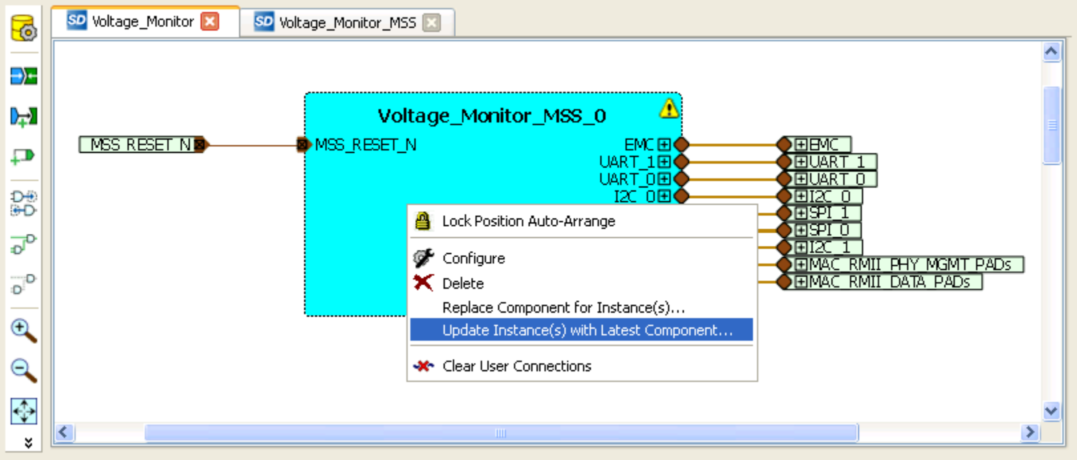


Figure 53 · Updating the MSS

2. Promote the M2F_GPIO [31:28] pins to top level.

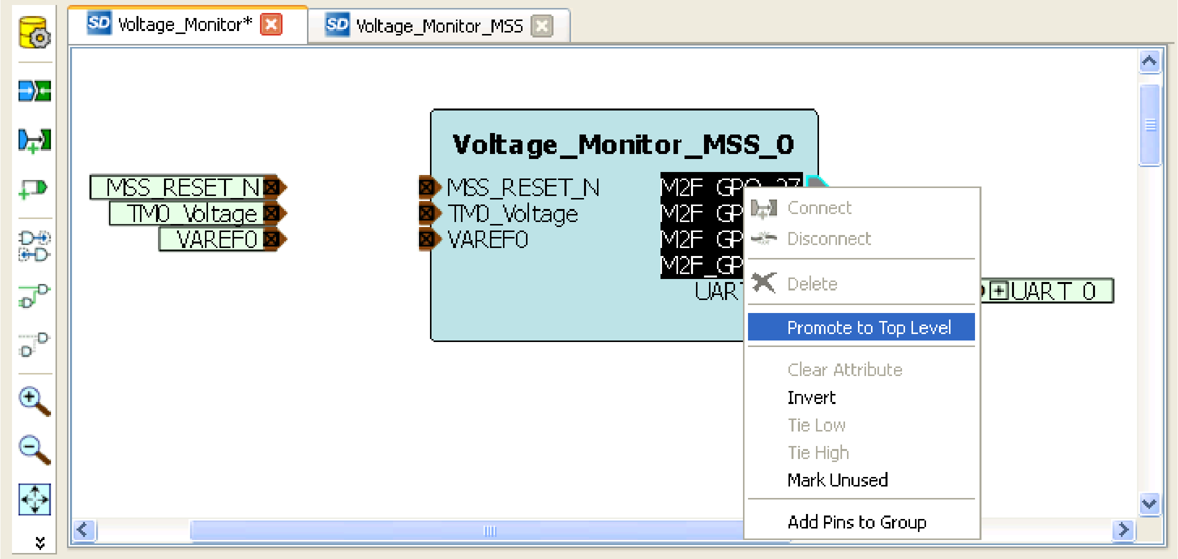


Figure 54 · GPIO Pins Promoted to Top Level

3. Click **Design > Configure Firmware** as shown in Figure 55 · .

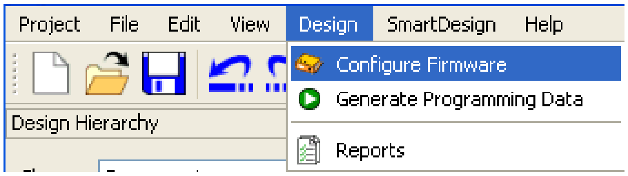
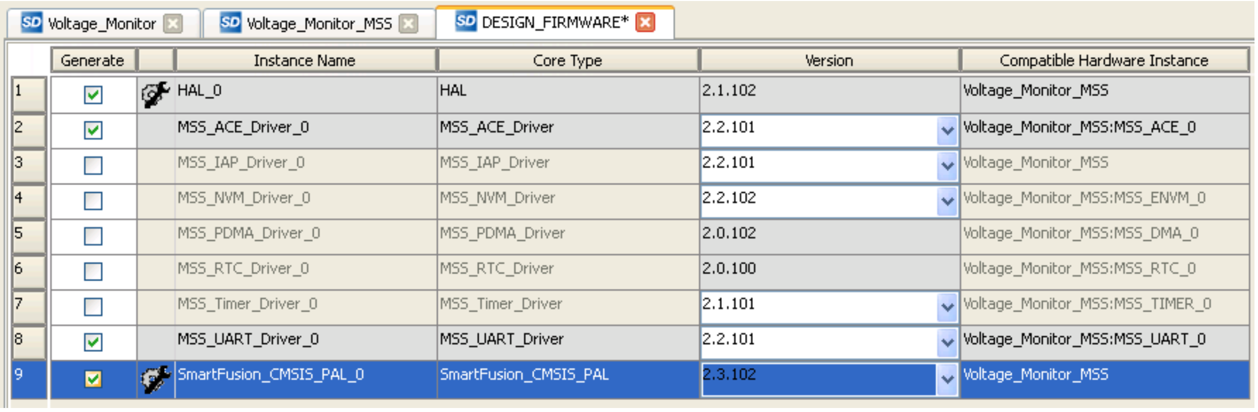


Figure 55 · Opening Design_Firmware

4. On the **DESIGN_FIRMWARE** tab, clear the **Generate** check box for all the peripherals for which you do not need to generate the firmware. Click Configuration on the SmartFusion_CMSIS_PAL_0 instance and select SoftConsole as the configuration.



	Generate		Instance Name	Core Type	Version	Compatible Hardware Instance
1	☑		HAL_0	HAL	2.1.102	Voltage_Monitor_MSS
2	☑		MSS_ACE_Driver_0	MSS_ACE_Driver	2.2.101	Voltage_Monitor_MSS:MSS_ACE_0
3	□		MSS_IAP_Driver_0	MSS_IAP_Driver	2.2.101	Voltage_Monitor_MSS
4	□		MSS_NVM_Driver_0	MSS_NVM_Driver	2.2.102	Voltage_Monitor_MSS:MSS_ENVM_0
5	□		MSS_PDMA_Driver_0	MSS_PDMA_Driver	2.0.102	Voltage_Monitor_MSS:MSS_DMA_0
6	□		MSS_RTC_Driver_0	MSS_RTC_Driver	2.0.100	Voltage_Monitor_MSS:MSS_RTC_0
7	□		MSS_Timer_Driver_0	MSS_Timer_Driver	2.1.101	Voltage_Monitor_MSS:MSS_TIMER_0
8	☑		MSS_UART_Driver_0	MSS_UART_Driver	2.2.101	Voltage_Monitor_MSS:MSS_UART_0
9	☑		SmartFusion_CMSIS_PAL_0	SmartFusion_CMSIS_PAL	2.3.102	Voltage_Monitor_MSS

Figure 56 · Configuring SmartFusion_CMSIS_PAL_0

5. Check whether or not you are able to see the latest version of the drivers without any warning or error indicating that firmware is missing from the Vault. If missing, refer to Appendix B – Firmware Catalog Settings.
6. Click **File > Save** to save the Design_Firmware.
7. Save the design and generate the component by clicking **Generate Component** or by selecting **SmartDesign > Generate Component**.

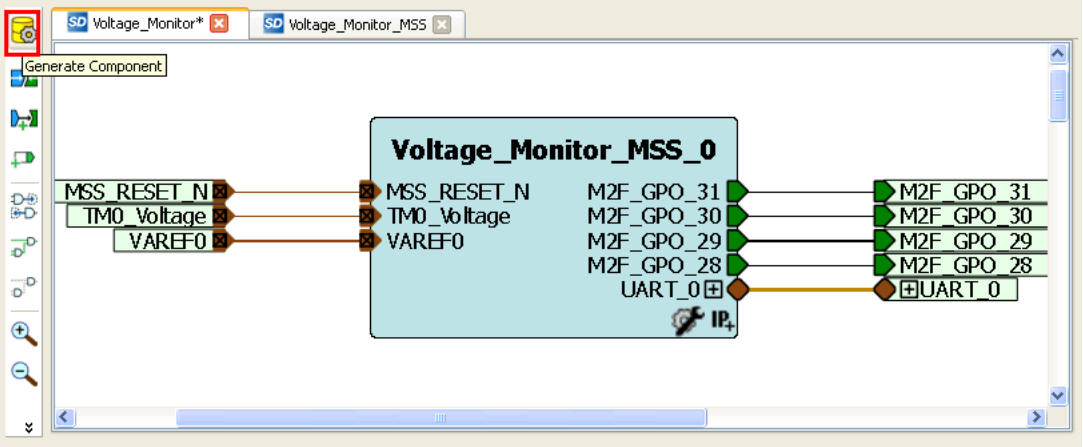
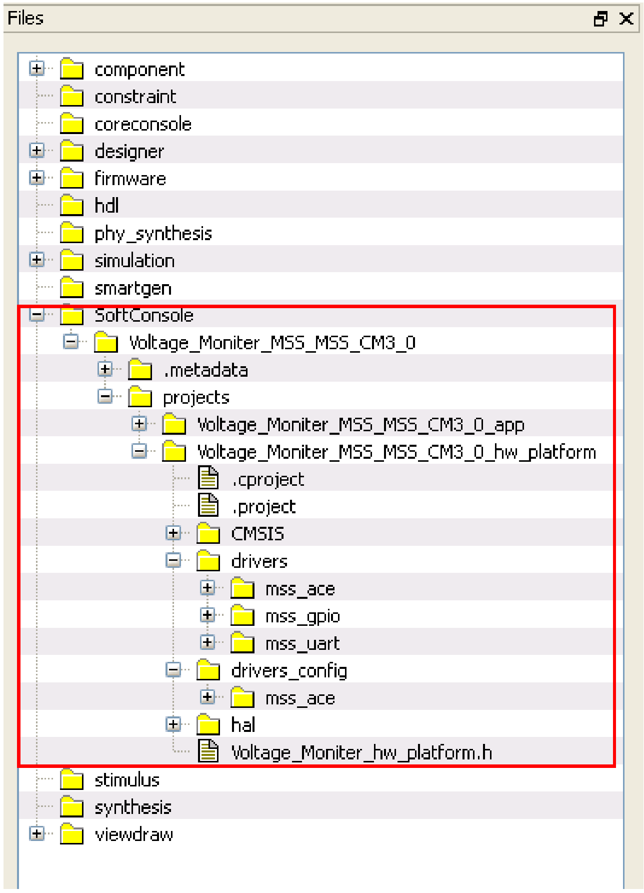


Figure 57 · Generating the MSS Component

8. After successful generation of project the log window displays the message “**Info: 'Voltage_Monitor' was successfully generated. Open datasheet for details**”. The datasheet has the Project information like Generated files, used IO’s, and Memory map etc.

9. Confirm that the SoftConsole folder is created with the folders and files as shown in Figure 58 · .



```
Files
component
constraint
coreconsole
designer
firmware
hdl
phy_synthesis
simulation
smartgen
SoftConsole
    Voltage_Moniter_MSS_MSS_CM3_0
        .metadata
        projects
            Voltage_Moniter_MSS_MSS_CM3_0_app
            Voltage_Moniter_MSS_MSS_CM3_0_hw_platform
                .cproject
                .project
                CMSIS
                drivers
                    mss_ace
                    mss_gpio
                    mss_uart
                drivers_config
                    mss_ace
                hal
                Voltage_Moniter_hw_platform.h
stimulus
synthesis
viewdraw
```

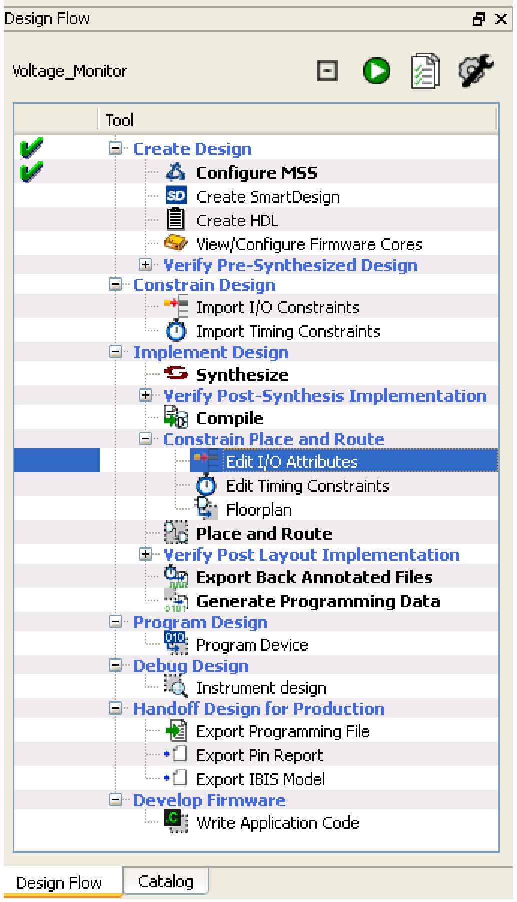
Figure 58 · Files Window

Generating the Program File

Libero SoC provides the push-button flow for Generating programming data of the project in a single step. By clicking **Generating Programming Data**, you can complete synthesis, place and route, verify timing and generate the programming file. You can also complete the flow by running the synthesis and place and route tools in interactive mode (step-by-step). For additional information, refer to the *Libero SoC Quick Start Guide*.

Push-button Design Flow

1. Click **Edit I/O Attributes** under **Constrain place and route** in the **Design Flow** window.



Design Flow

Voltage_Monitor

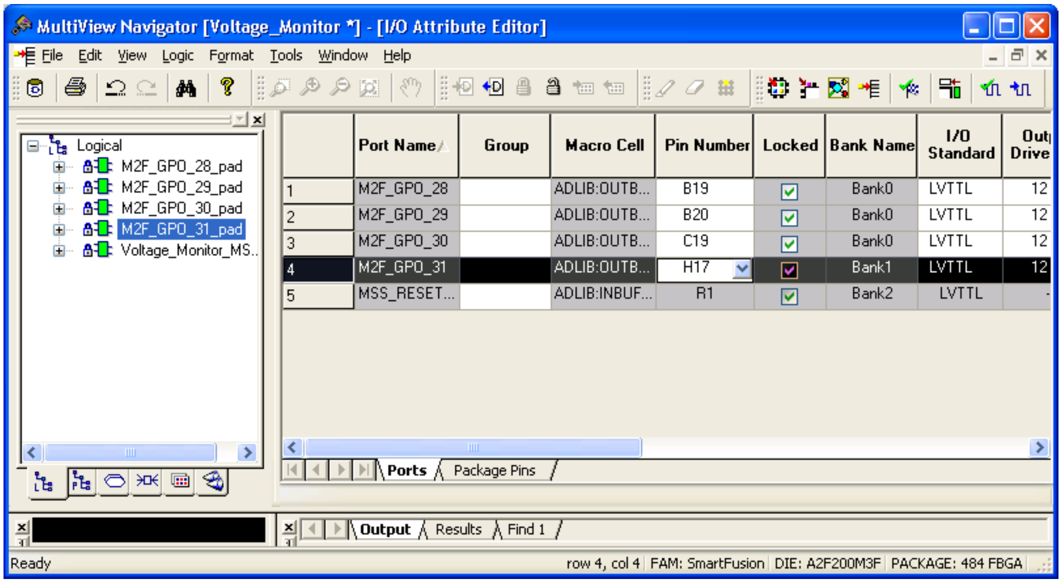
Tool

- ✓ Create Design
  - ✓ Configure MSS
  - Create SmartDesign
  - Create HDL
  - View/Configure Firmware Cores
  - Verify Pre-Synthesized Design
- Constrain Design
  - Import I/O Constraints
  - Import Timing Constraints
- Implement Design
  - Synthesize
  - Verify Post-Synthesis Implementation
  - Compile
  - Constrain Place and Route
    - Edit I/O Attributes
    - Edit Timing Constraints
    - Floorplan
  - Place and Route
  - Verify Post Layout Implementation
  - Export Back Annotated Files
  - Generate Programming Data
- Program Design
  - Program Device
- Debug Design
  - Instrument design
- Handoff Design for Production
  - Export Programming File
  - Export Pin Report
  - Export IBIS Model
- Develop Firmware
  - Write Application Code

Design Flow | Catalog

Figure 59 · Edit I/O Attributes

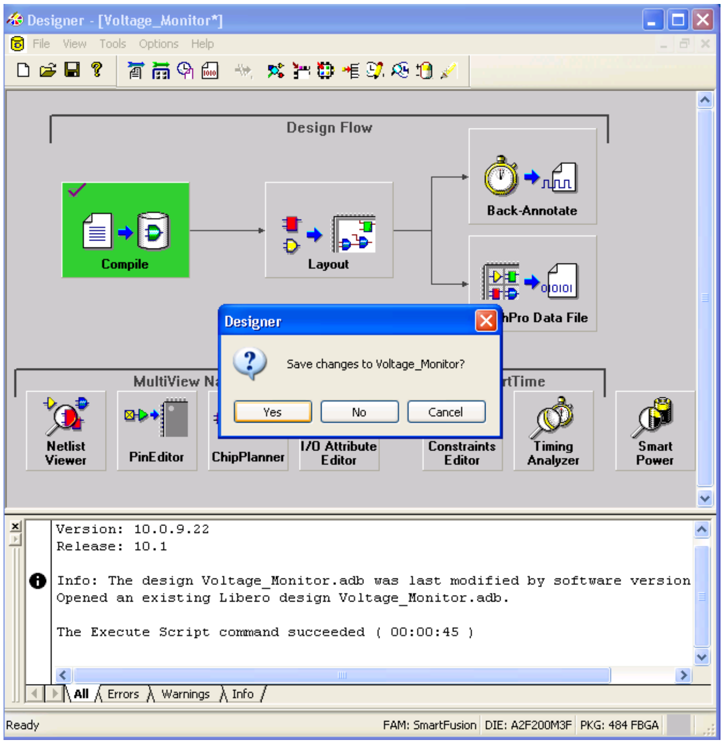
2. Make the following pin assignments in **MultiView Navigator** window as shown in Figure 60 · :
   - GPO_28 to B19
   - GPO_29 to B20
   - GPO_30 to C19
   - GPO_31 to H17



	Port Name	Group	Macro Cell	Pin Number	Locked	Bank Name	I/O Standard	Out Drive
1	M2F_GPO_28		ADLIB:OUTB...	B19	☑	Bank0	LVTTL	12
2	M2F_GPO_29		ADLIB:OUTB...	B20	☑	Bank0	LVTTL	12
3	M2F_GPO_30		ADLIB:OUTB...	C19	☑	Bank0	LVTTL	12
4	M2F_GPO_31		ADLIB:OUTB...	H17	☑	Bank1	LVTTL	12
5	MSS_RESET...		ADLIB:INBUF...	R1	☑	Bank2	LVTTL	-

Figure 60 · MultiView Navigator GUI

3. Commit and check the edits using **File > Commit and Check**. Connect any errors that are reported in the MVN log window.
4. Close the **MultiView Navigator** using **File > Exit**.
5. Close the **Designer** window and select **Yes** when it prompts to save changes.



Save changes to Voltage_Monitor?

```
Version: 10.0.9.22
Release: 10.1

Info: The design Voltage_Monitor.adb was last modified by software version
Opened an existing Libero design Voltage_Monitor.adb.

The Execute Script command succeeded ( 00:00:45 )
```

Figure 61 · Designer Window

6. Click **Generate Programming Data** to complete the place and route, verify timing and generate the programming file. This completes the.fdb file generation.

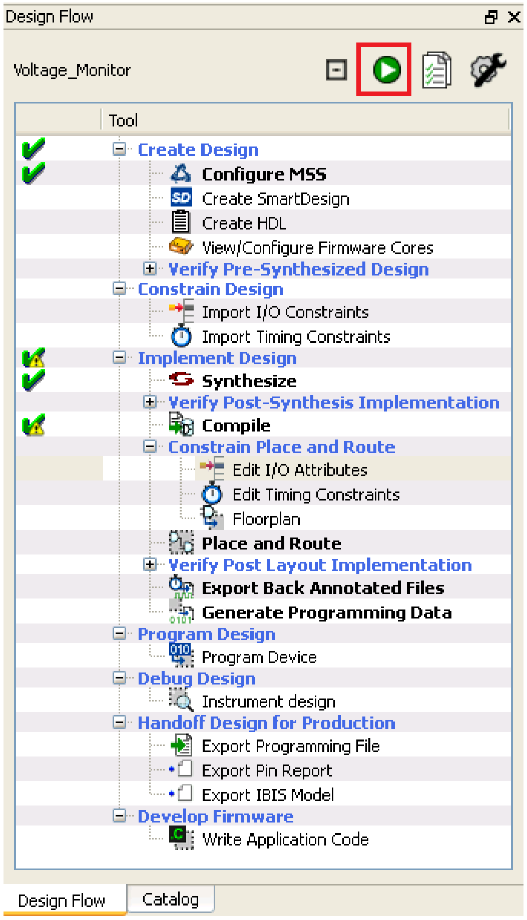
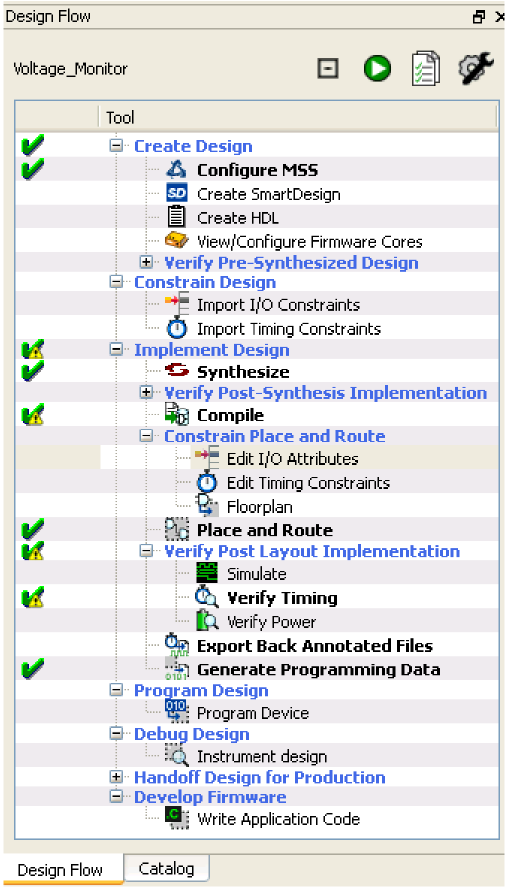


Figure 62 · Generating Programming Data

7. The **Design Flow** window looks similar to Figure 63 · .



Design Flow

Voltage_Monitor

Tool

- ✓ Create Design
  - ✓ Configure MSS
  - Create SmartDesign
  - Create HDL
  - View/Configure Firmware Cores
  - Verify Pre-Synthesized Design
- Constrain Design
  - Import I/O Constraints
  - Import Timing Constraints
- ✓ Implement Design
  - ✓ Synthesize
  - Verify Post-Synthesis Implementation
  - ✓ Compile
  - Constrain Place and Route
    - Edit I/O Attributes
    - Edit Timing Constraints
    - Floorplan
  - ✓ Place and Route
  - ✓ Verify Post Layout Implementation
    - Simulate
    - ✓ Verify Timing
    - Verify Power
  - Export Back Annotated Files
  - ✓ Generate Programming Data
- Program Design
  - Program Device
- Debug Design
  - Instrument design
- Handoff Design for Production
- Develop Firmware
  - Write Application Code

Design Flow | Catalog

Figure 63 · Design Flow Window After Building The Project

8. Follow Step 5 - Programming SmartFusion Board Using FlashPro.

List of Changes

Revision	Changes	Page
Revision 5 (February 2012)	Modified Associated Project Files section (SAR 36900).	3
	Modified Step 6 - Building the Software Application Through SoftConsole section. (SAR 36900).	25
	Modified Step 7 - Configuring Serial Terminal Emulation Program section (SAR 36900).	26
	Updated Figure 34 (SAR 36900).	27
	Modified Step 8 - Installing Drivers for the USB to RS232 Bridge section (SAR 36900).	29
	Modified Step 9 - Debugging the Application Project using SoftConsole section.	30
Revision 4 (January 2012)	Modified Step 2 - Configuring MSS Peripherals section. (SAR 36492)	8 and 15
	Modified Step 6 - Building the Software Application Through SoftConsole section. (SAR 36492)	25
	Modified Step 9 - Debugging the Application Project using SoftConsole section. (SAR 36492)	30
Revision 3 (November 2011)	Updated the document for Libero SoC v10.0 (SAR 35045).	

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

Product Support

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

Customer Service

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

From North America, call **800-262-1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world **650. 318.8044**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, common design cycle questions, 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 Microsemi SoC Products Group 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 website.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at http://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 dropdown list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise Drive, Aliso Viejo CA 92656
Within the USA: (800) 713-4113
Outside the USA: (949) 221-7100
Fax: (949) 756-0308 · www.microsemi.com

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

© 2012 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.