# Libero® SoC Quick Start Guide

## for Software v10.0

Microsemi

# Table of Contents

Introduction and Design Overview . . . 3
Tutorial Requirements . . . 3
Design Overview . . . 4
Interrupt Generator Block Description . . . 5

1 Step 1 - Creating a Libero SoC Project and Configuring the SmartFusion Microcontroller Subsystem . 7
Libero SoC Interface Description . . . 10

2 Step 2 - Configuring the SmartFusion MSS . . . 11

3 Step 3 - Configuring Timers, Connecting to the MSS and Generating the Design . . . 17

4 Step 4 - Performing Pre-Synthesis Simulation . . . 21

5 Step 5 - Implementing the Design . . . 29

6 Software Implementation . . . 45
Step 1 – Invoking the SoftConsole from Libero SoC . . . 45
Step 2 - Configuring Serial Terminal Emulation Program . . . 50
Step 3 - Installing Drivers for the USB to RS232 Bridge . . . 52
Step 4 - Debugging the Application Project Using SoftConsole . . . 52
Step 5 - Building an Executable Image in Release mode . . . 55

A Product Support . . . 57
Customer Service . . . 57
Customer Technical Support Center . . . 57
Technical Support . . . 57
Website . . . 57
Contacting the Customer Technical Support Center . . . 57
ITAR Technical Support . . . 58

# Introduction and Design Overview

This tutorial introduces you to the Microsemi® system-on-chip (SoC) development flow using Libero® SoC. It is a starting point for any FPGA design engineer who is new to Microsemi FPGAs, or just wants to learn more about Libero SoC. It demonstrates the basics on how to use Libero SoC and its tools to create a simple design.The sample design incorporates the Libero SoC Catalog IP core macros and hard embedded microcontroller subsystem (MSS) and guides you through synthesis, simulation and programming.

The tutorial was developed using the SmartFusion® Evaluation Kit Board and A2F200M3F device and covers the following tools and features:

- Libero SoC
  - Design flow
  - SmartDesign
  - Catalog
- Mentor Graphics® ModelSim® AE
- Synopsys Synplify Pro AE
- Microsemi Designer
  - Compile
  - Place and Route
  - I/O Attribute Editor
  - Pin Editor
  - SmartTime
- FlashPro

## Tutorial Requirements

### Software Requirements

This tutorial requires that the following software is installed on your computer:

- Libero SoC v10.0 or higher, which can be downloaded from http://www.microsemi.com/soc/download/software/libero/default.aspx
- FlashPro v10.0 or higher, which is installed as part of the Microsemi Libero SoC installation and can be launched from within Libero SoC or standalone.
- SoftConsole v3.3 or higher, which is installed as part of the Microsemi Libero SoC installation and can be launched from within Libero SoC or standalone.
- HyperTerminal or similar software (PuTTy or Tera Term), normally under Start > Programs > Accessories > Communications > HyperTerminal.

  For other alternatives to HyperTerminal see http://www.windowsreference.com/windows-7/alternatives-to-hyperterminal-in-windows-7/
- USB Drivers for USB to UART connection http://www.microsemi.com/soc/documents/CP2102_driver.zip

### Hardware Requirements

You will need the following hardware:

- SmartFusion Evaluation Kit Board

- Two USB cables for programming and communication (provided with the SmartFusion Evaluation Kit)

## Extracting Source Files

Download the Design Files from http://www.microsemi.com/soc/download/rsc/?f=LiberoSoC_QS_DF. Please extract design files in the root directory of your local drive (e.g. C :\) using WinRAR .

You may seen an issue when launching SoftConsole from Libero SoC when you click Write Application Code. SoftConsole may open without any workspace or display an error. Refer to the following KB article for the workaround: http://www.microsemi.com/soc/kb/article.aspx?id=KI8879.

# Design Overview

SmartFusion cSoC FPGA devices contain a 100 MHz ARM® Cortex™-M3 processor, Ethernet MAC, DMA engine, real-time counter (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), programmable analog circuitry, and FPGA fabric consisting of programmable logic tiles, static random access memory (SRAM), and phase locked loops (PLLs). The Cortex-M3 processor includes an interrupt controller called the nested vectored interrupt controller (NVIC). There are 151 interrupts available in the NVIC, of which 32 are general purpose interrupts that can be connected to the fabric or outside world. There is a dedicated fabric interrupt, FABINT, which is connected to the FPGA.

The design example explains how to use a general purpose input/output (GPIO) and FABINT to interrupt the MSS from the FPGA fabric, and how to implement the interrupt handler on the SmartFusion Evaluation Kit Board Cortex-M3 processor. It consists of an interrupt generator block that has two timer blocks. Figure 1 illustrates how to interface an interrupt generator with the MSS to generate GPIO and FABINT interrupts. The universal asynchronous receiver/transmitter (UART) in MSS is used for printing interrupt messages on the HyperTerminal.

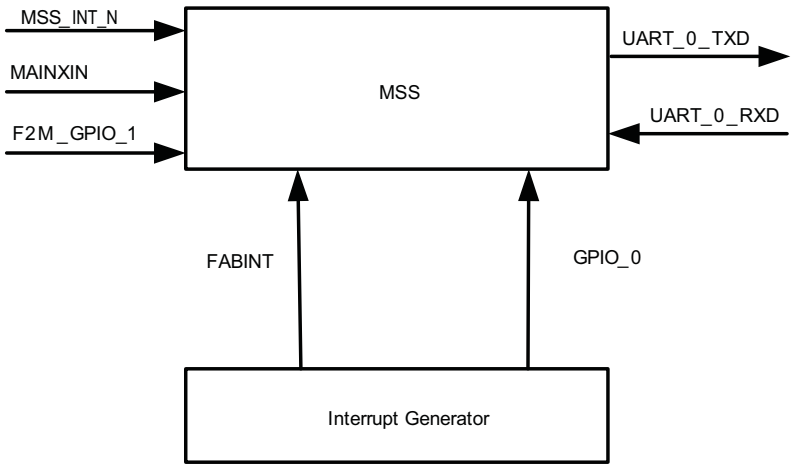


***Figure 1* • Tutorial Design Block Diagram**

The example design contains the following blocks:

- MSS
- Interrupt generator

The example design contains the following inputs and outputs:

- Inputs: Clock, Reset and External interrupt signal
- Outputs: HyperTerminal

# Interrupt Generator Block Description

The interrupt generator (Figure 2) has two timer blocks. Two different timers are implemented in the FPGA fabric and run with the same clock source. The clock conditioning circuit (CCC) in the MSS generates a 10 MHz clock and acts as the clock source for the timers. This design example is implemented on the SmartFusion Evaluation Kit Board and uses three interrupts.

Timer 1 and Timer 2 are internal interrupts; the third interrupt is generated by SW1 that is connected to FPGA I/O pin G19. An interrupt is generated asynchronously whenever SW1 is pressed. The Timer 1 block is connected to FABINT of the MSS and the Timer 2 block is connected to the GPIO_0 of the MSS, which is configured to connect with the fabric. Figure 2 shows the block diagram of the interrupt generator.

Whenever any of the interrupts occur (Timer 1, Timer 2, or the switch interrupt), the processor executes the corresponding interrupt service routine and the source of the interrupt is printed on HyperTerminal.

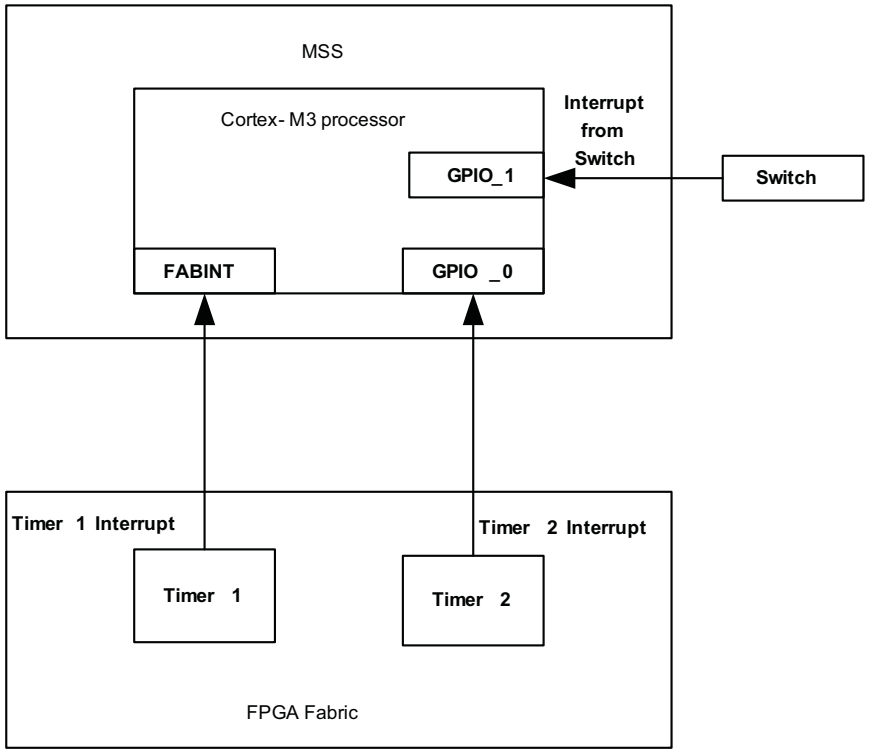


***Figure 2* • Interrupt Generator Block Diagram**

This tutorial provides step-by-step instructions on how to build MSS and timer blocks, simulate the timer block, synthesize, place and route, and generate a programming file for the entire design. It also guides you on how to program the design on the SmartFusion Evaluation Kit Board.

# 1 – Step 1 - Creating a Libero SoC Project and Configuring the SmartFusion Microcontroller Subsystem

1. Click **Start > Programs > Microsemi Libero SoC v10.0 > Libero SoC v10.0**. The Libero SoC window appears (Figure 1-1).


***Figure 1-1 •*** **Libero SoC Project Manager**

2. From the **Project** menu choose **New Project**. The **New Project** dialog box appears (Figure 1-2).

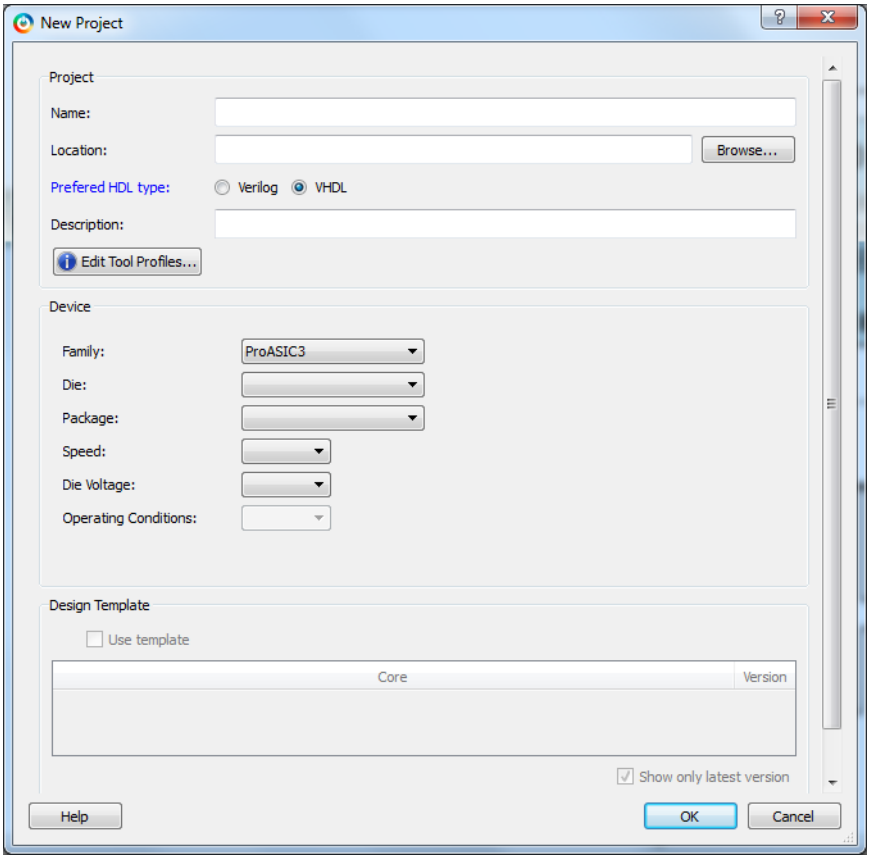


***Figure 1-2* • Libero SoC New Project Dialog Box**

3. Enter the information as shown in Figure 1-3. If a value in the dialog box is not specified below you may use the default.

- Name: Interrupt_MSS_GPIO
- Location: C:/Microsemiprj
- Preferred HDL type: VHDL (for VHDL projects) or Verilog (for Verilog projects)
- Family: SmartFusion
- Die: A2F200M3F
- Package: 484 FBGA
- Speed: STD
- Core: SmartFusion microcontroller subsystem (MSS) 2.5.106

If the SmartFusion MSS text (circled in Figure 1-3) is in italics the core is not in your IP vault. Double-click the core name to download the latest MSS core.

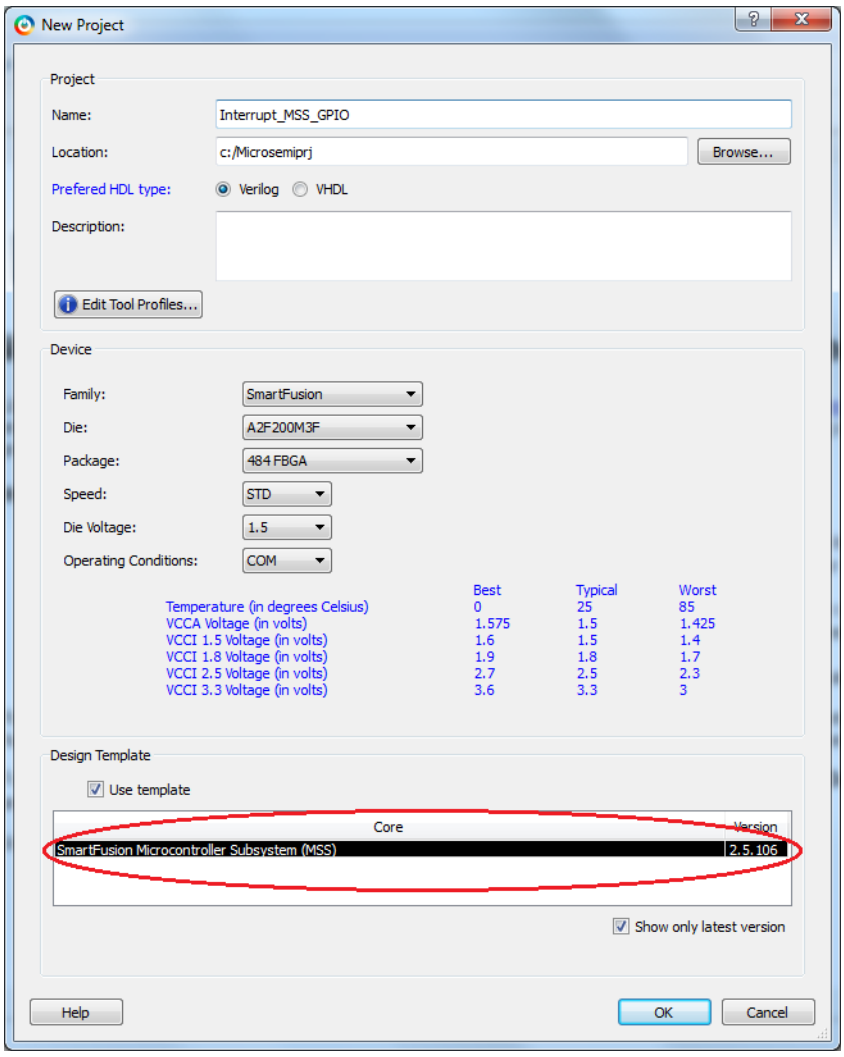


***Figure 1-3* • Libero New Project Dialog Box**

4. Click the **Edit Tool Profiles** button to open the **Tool Profiles** dialog box (Figure 1-4).

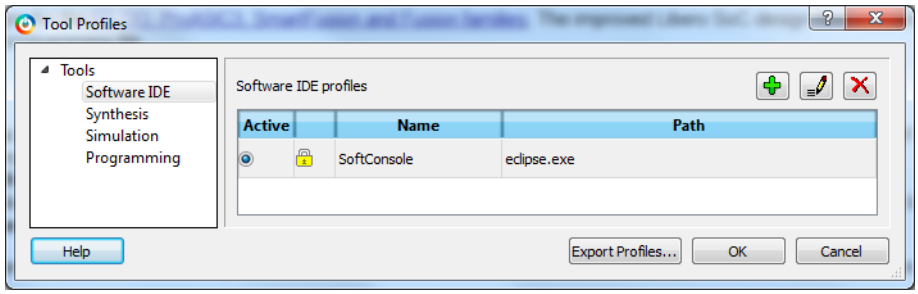


***Figure 1-4* • Tool Profiles Dialog Box**

5. Select the **Software IDE Profile** (SoftConsole, IAR or Keil) so that the Libero SoC generates the corresponding project.

   If the tools are not listed then click the **Add Profile** button to add the required tool. (If SoftConsole v3.3 was not installed in default location then "?" appears on SoftConsole.)
6. After selecting correct path to the tools, click **OK** to close the **Tool Profiles** dialog box.
7. Click **OK** to close the **New Project** dialog box.

Note: Click Yes if the software prompts you to download the MSS. This occurs when the vault does not have the selected MSS core.

# Libero SoC Interface Description

The Libero SoC interface enables a push-button design flow via several tabs and an expanded work area.

- **Work Window** - Displays the SmartDesign canvas, HDL editor or Report view. Click the **Maximize/Restore Work Area** button to show/hide the other interface elements.
- **Design Hierarchy Tab** - Lists components and modules in your design. Use it to manage your design files.
- **Files Tab** - Lists all your project files by directory; use it to manage your project files directly.
- **Design Flow Window** - Enables you to execute the push-button design flow or, if you prefer, open the tools interactively and specify custom settings.
- **Catalog** - Lists the cores available for use in your design. Click and drag them onto your canvas and add them to your design.
- **HDL Templates** - Lists HDL templates for common constructs; double-click a template to copy it to the clipboard, then paste it into your HDL to use it in your design.
- **Log Window** - Lists all messages, errors, warnings, and info for the SoC tools. Click each type to filter accordingly.

# 2 – Step 2 - Configuring the SmartFusion MSS

If you completed the steps in the previous chapter, the **SmartDesign** Canvas opens with the SmartFusion MSS component.

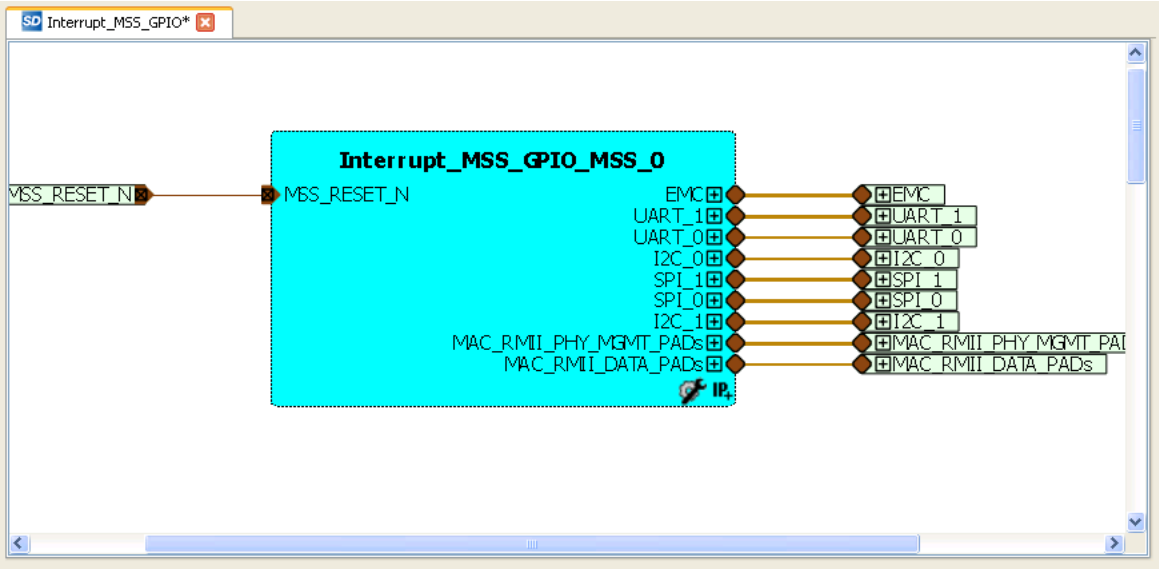


***Figure 2-1* • SmartFusion MSS on the Canvas**

1. Double-click **Interrupt_MSS_GPIO_MSS_0** to open the SmartFusion MSS configurator.

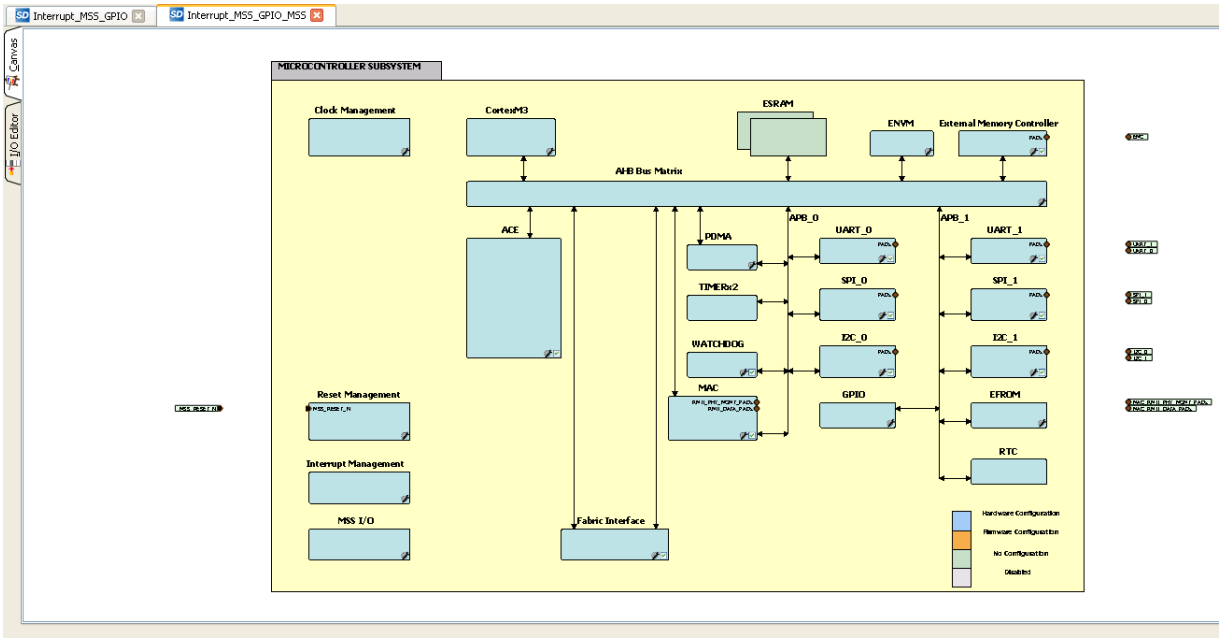


***Figure 2-2* • SmartFusion MSS Configurator**

MSS peripherals that can be disabled have a small wrench symbol and checkbox in the right corner (Figure 2-3). To disable a peripheral, select the peripheral, right-click and choose **Disable**, or click the checkbox. The peripheral turns grey to indicate it has been disabled.

Disabled peripherals can be enabled by repeating the procedure.

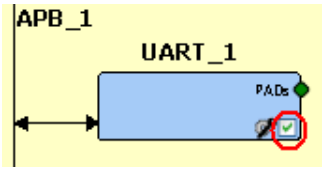


***Figure 2-3* • Disabling the Peripheral**

2. Disable the following peripherals:
   - External Memory Controller
   - ACE
   - MAC
   - UART_1
   - SPI_0
   - SPI_1
   - I2C_0
   - I2C_1
   - Fabric Interface
   - WATCHDOG

   The MSS appears (Figure 2-4).

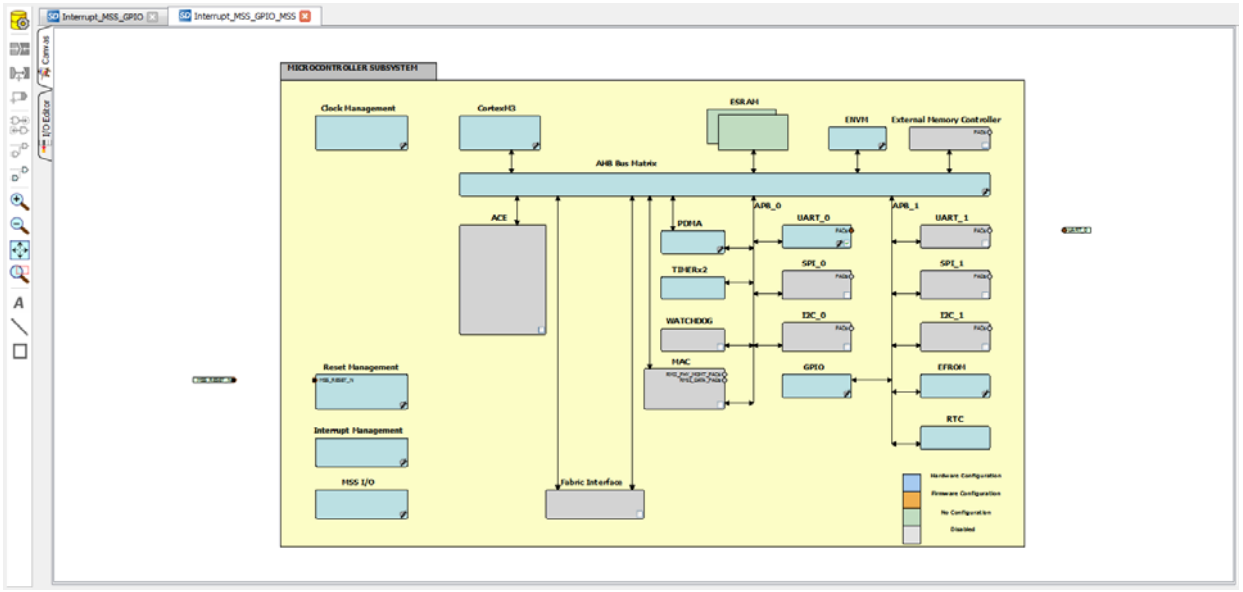


***Figure 2-4* • SmartDesign MSS in the SmartDesign Canvas**

3. Double-click the **Clock Management** block to configure the MSS CCC.

4. Set the Microcontroller Subsystem Clock Configurator options as follows (shown in Figure 2-5):
   - **CLKA**: On-chip RC Oscillator
   - **FCLK source**: PLL output (VCO output)
   - **FCLK frequency**:40 MHz
   - **FAB_CLK**: Enable and select /4
   - **FAB_LOCK**: Enable

Use the default for all other settings.

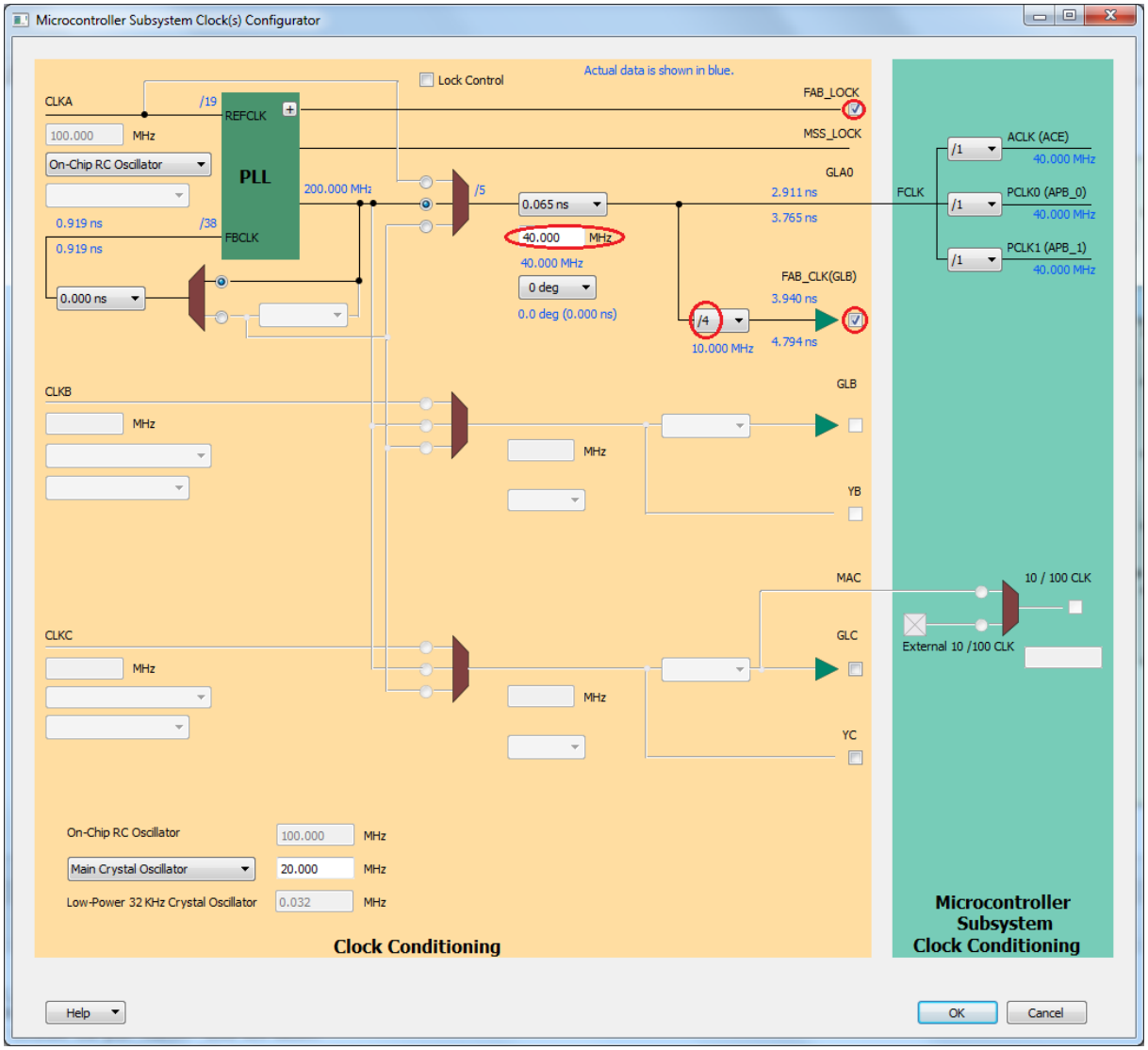


***Figure 2-5* • MSS Clock Management Configuration**

5. Click **OK** to close the **Microcontroller Subsystem Clock Configurator** dialog box.

6. Double-click Interrupt Management to open and configure the MSS Interrupts. Check the option **Use Fabric to MSS Interrupt (FABINT)** (Figure 2-6). Click OK to continue.

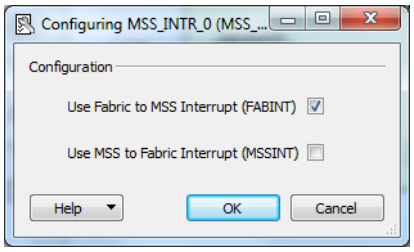


***Figure 2-6* • MSS Interrupts Management Configuration**

7. Double-click the **GPIO** block in MSS to configure it. Configure GPIO_0 and GPIO_1 as an **Input** (as shown in Figure 2-7); leave the rest of the ports at their default settings. Click **OK** to close the GPIO configuration dialog box. The F2M_GPI_[1:0] ports are promoted to the top level automatically.

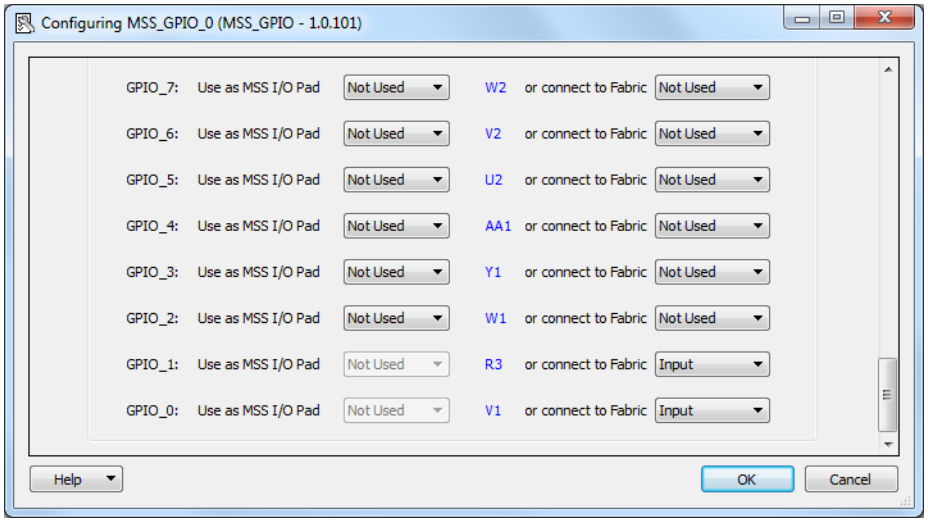


***Figure 2-7* • MSS Interrupt Management Configuration**

Keep the default settings fort he Reset Management and UART_0 blocks.

8. Save the Interrupt_MSS_GPIO component (**File > Save Interrupt_MSS_GPIO**).

   The MSS component (Interrupt_MSS_GPIO_MSS_0) is shown in the SmartDesign Canvas. The warning symbol indicates that the port list for the SmartFusion cSoC component has changed.

9. Right-click and choose **Update Instance(s) with Latest Component** (Figure 2-8).

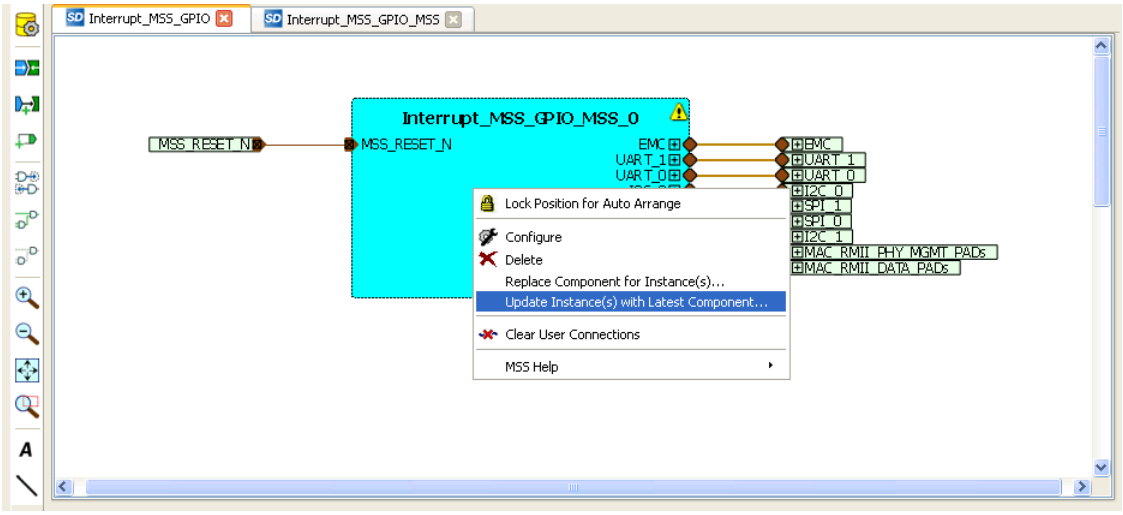


***Figure 2-8 •*** **Update the MSS Component**

# 3 – Step 3 - Configuring Timers, Connecting to the MSS and Generating the Design

The timer blocks required in the design are created with the SmartGen core generator. SmartGen cores are listed in the Catalog; the Counter is under **Basic Blocks**, as shown in Figure 3-1.

Make sure that you are adding the counters to your Interrupt_MSS_GPIO Canvas.

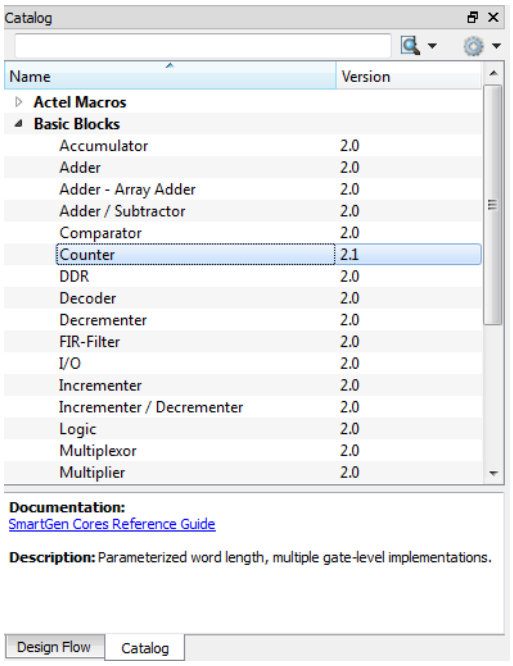


***Figure 3-1* • IP Catalog**

1. Click and drag the counter from the Catalog to the Interrupt_MSS_GPIO Canvas. The Create Core dialog box opens and enables you to configure the Counter. Modify the settings as shown in Figure 3-2.
2. Click **Generate** to add the counter to your Canvas. Name the 20-bit counter **Timer_0**.
3. Click and drag another counter from the Catalog to the Interrupt_MSS_GPIO Canvas. Use the settings shown in Figure 3-2 with the exception of Width. **Set the width to 21**.
4. Click **Generate** to add the counter to your Canvas. Name the 21-bit counter **Timer_1**.

Double-click any component on the Canvas to modify the settings. For example, if you wish to change the counter width of Timer_1, double-click **Timer_1_0** on your Canvas and enter a new value for **Width**. Click **Generate** to continue and save your changes.

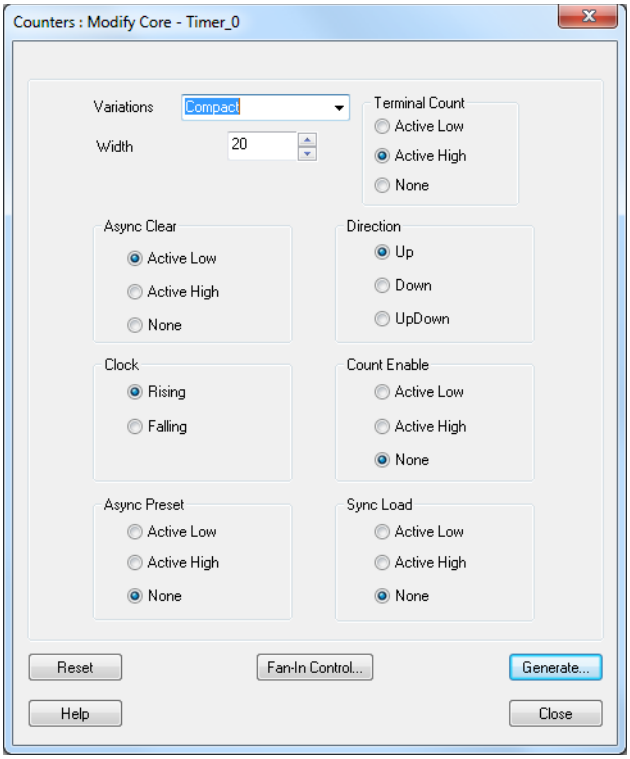


***Figure 3-2* • 20 bit Counter Configuration**

5. Connect the **Aclr** port of **Timer_0_0** to the FAB_LOCK port of Interrupt_MSS_GPIO_MSS_0 component as follows:

- Click the Connection Mode button to enable it. You can use Connection Mode to click and drag between valid connections.
- Click the **Aclr** port of **Timer_0_0**and drag to the **FAB_LOCK** port of **Interrupt_MSS_GPIO_MSS_0** component.

6. Repeat the procedure above to make the following connections:

- **Aclr** port of Timer_1_0: **FAB_LOCK** port of Interrupt_MSS_GPIO_MSS_0.
- **Clock** port of Timer_0_0: **FAB_CLK** port of Interrupt_MSS_GPIO_MSS_0.
- **Clock** port of Timer_1_0: **FAB_CLK** port of Interrupt_MSS_GPIO_MSS_0.
- **Tcnt** port of Timer_0_0: **FABINT** port of Interrupt_MSS_GPIO_MSS_0.
- **Tcnt** port of Timer_1_0: **F2M_GPI_0** port of Interrupt_MSS_GPIO_MSS_0.

7. Right-click **F2M_GPI_1** of **Interrupt_MSS_GPIO_MSS_0** and choose **Promote to Top Level**.
8. Disconnect the Q [19:0] and Q[20:0] ports; To do so, right-click each port and choose **Mark Unused**.

After making all the connections the **Interrupt_MSS_GPIO** appears as shown in Figure 3-3. Pad ports and nets are displayed in brown.

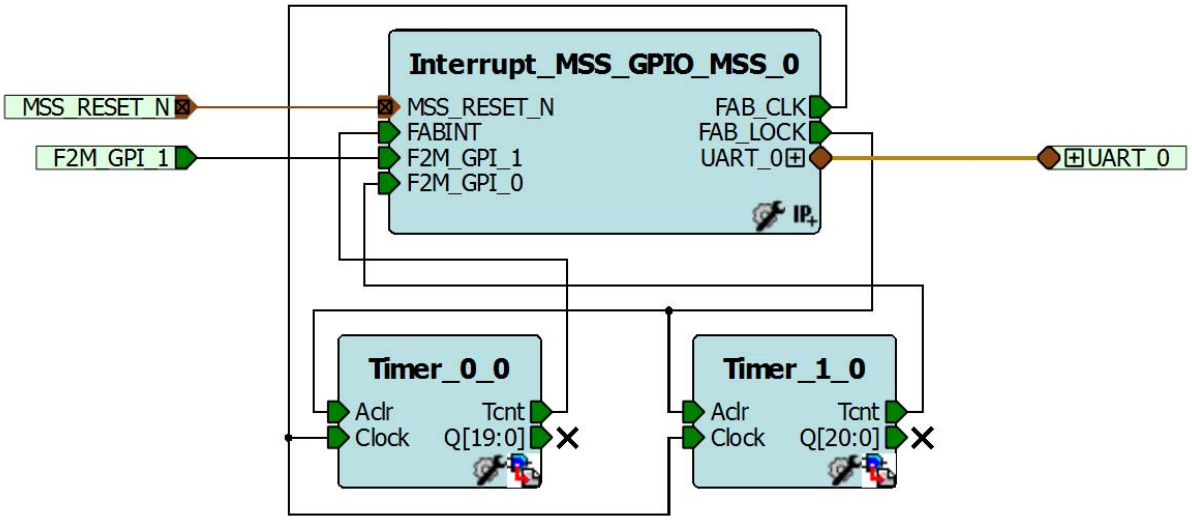


***Figure 3-3* • SmartDesign Canvas After Making Connections**

9. From the **Design** menu choose **Configure Firmware**.
10. In the DESIGN_FIRMWARE tab select the drivers that need to be generated. If a green button is shown next to your driver, click the button to download the driver.
11. Select **HAL_0**, **MSS_GPIO_Driver_0**, **MSS_UART_Driver_0**, **SmartFusion_CMSIS_PAL_0** and uncheck the Generate check box for other drivers, as shown in Figure 3-4. Save the DESIGN_FIRMWARE tab.

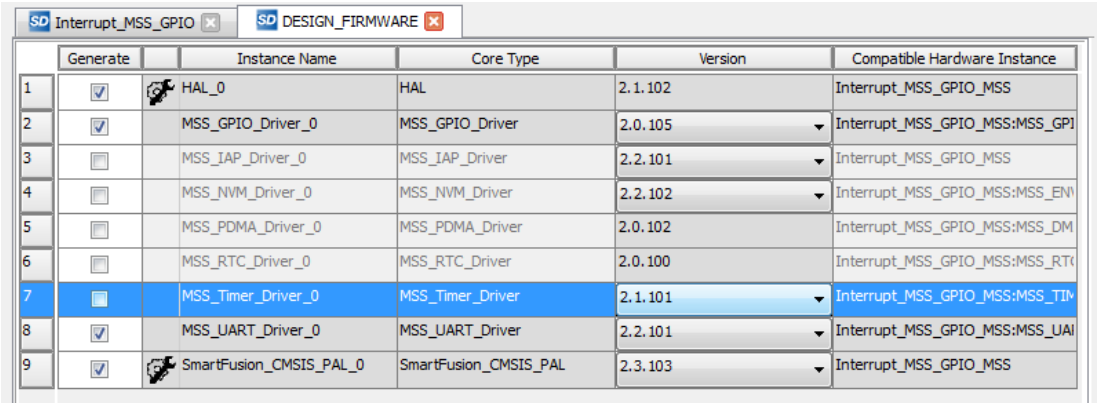


***Figure 3-4* • Configure Firmware**

12. From the **SmartDesign** menu choose **Generate Component**.

    If you do not see **Generate Component** in the SmartDesign menu, make sure that you are viewing the Interrupt_MSS_GPIO tab.

A green check mark under Create Design in the Libero SoC design flow window indicates the design was created without any errors (Figure 3-5).

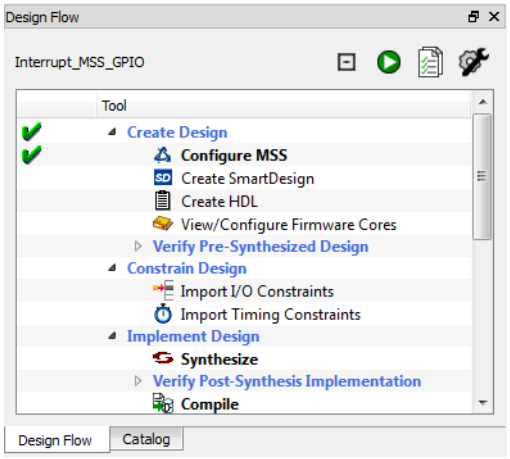


***Figure 3-5* • Successful Generation of SmartDesign Component**

After you generate the MSS Component successfully the Log window displays the message **"Info: 'Interrupt_MSS_GPIO' was successfully generated. Open datasheet for details"** (Figure 3-6). Scroll in the Interrupt_MSS_GPIO datasheet to familiarize yourself with the Generated Files, Firmware and Memory Map sections (click the hyperlink at the top of the datasheet to move to the section of interest).

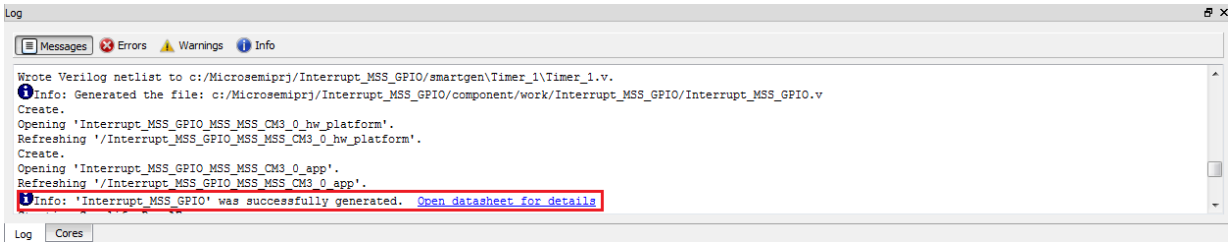


***Figure 3-6* • Log Window**

# 4 – Step 4 - Performing Pre-Synthesis Simulation

SmartDesign creates a testbench that can be used to simulate the design. In this step you will perform pre-synthesis simulation on the design.

1. Open the Libero SoC project settings (**Project > Project Settings**) and choose **Do File** under **Simulation options**. Under Simulation Runtime enter a value of **1ms** (as shown in Figure 4-1), Save, and Close.

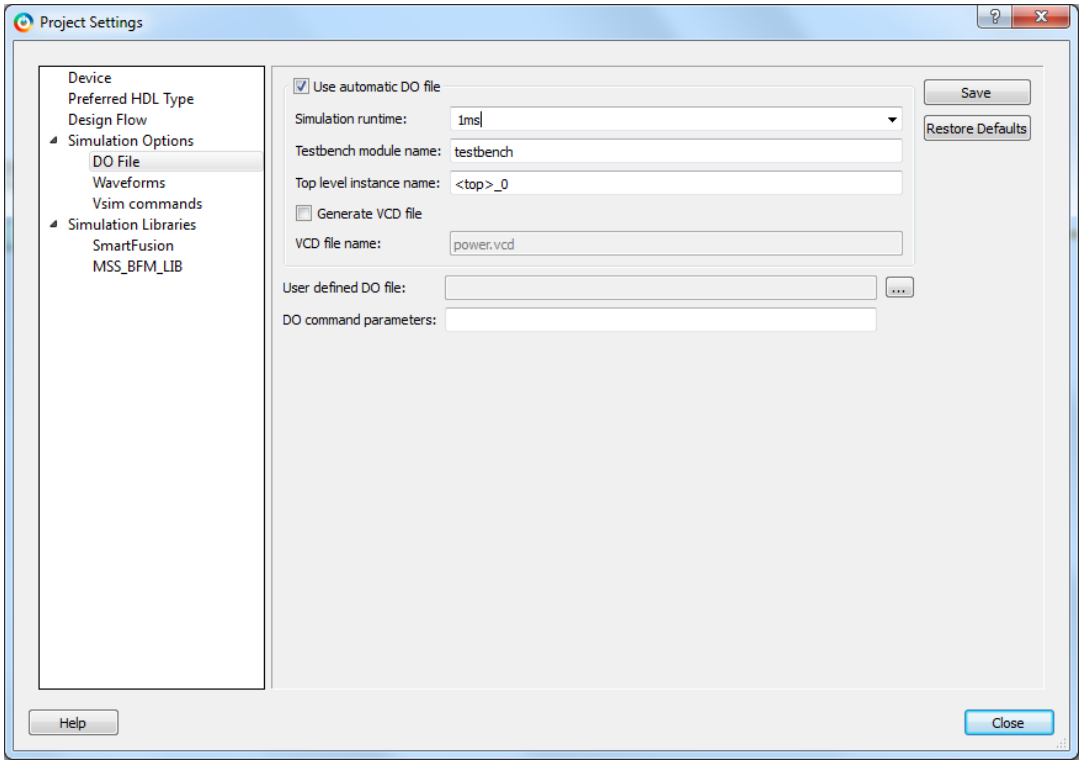


***Figure 4-1* • Project Settings Dialog Box**

2. Under **Simulation Options**, click **Waveforms** and click the checkbox to enable the option **Log all signals in the design** (Figure 4-2).

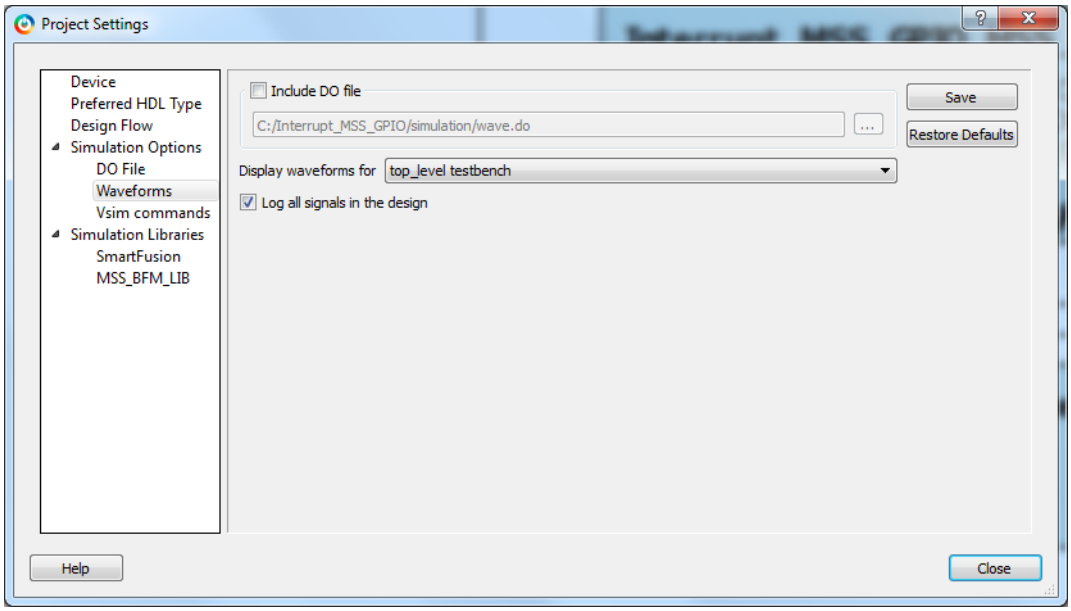


***Figure 4-2* • Changing Waveform Options**

3. Click **Save** to save the changes to the project settings. Click **Close** to close the Project Settings dialog box.
4. In the Design Flow window, expand Verify Pre-Synthesized Design and right-click **Simulate** and choose **Open Interactively** to launch ModelSim in GUI mode.

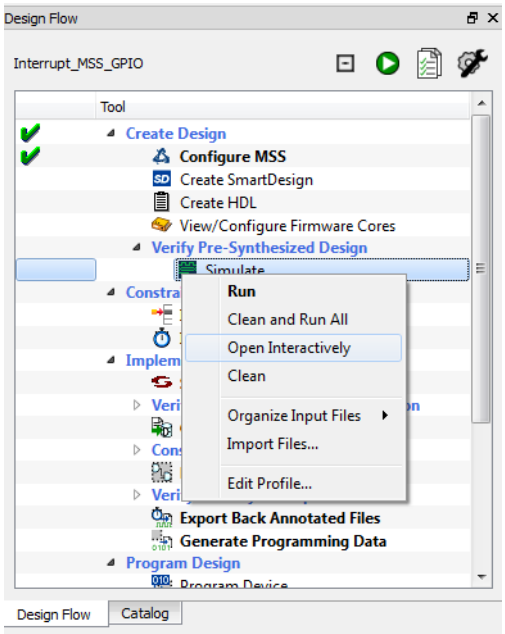


***Figure 4-3* • Invoking ModelSim**

ModelSim opens and automatically imports a **run.do** macro file that contains the links to the design files and gives simulation commands. The simulator compiles the source files and loads the design. The wave window appears as shown in Figure 4-4.

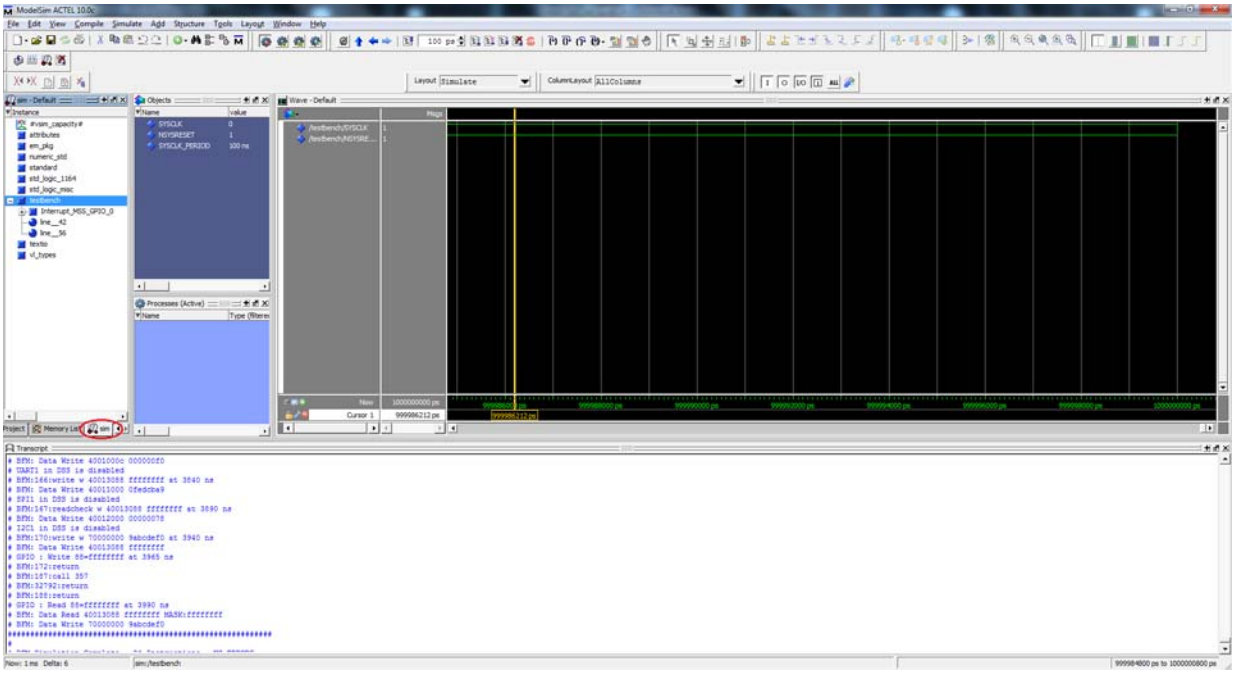

***Figure 4-4* • ModelSim Wave Window**

You must add additional signals to the wave window in order to confirm the design is functioning properly.

5. Select the ModelSim **sim** tab (circled in Figure 4-4). Expand the Design Hierarchy and select **Timer_1_0** (Figure 4-5).

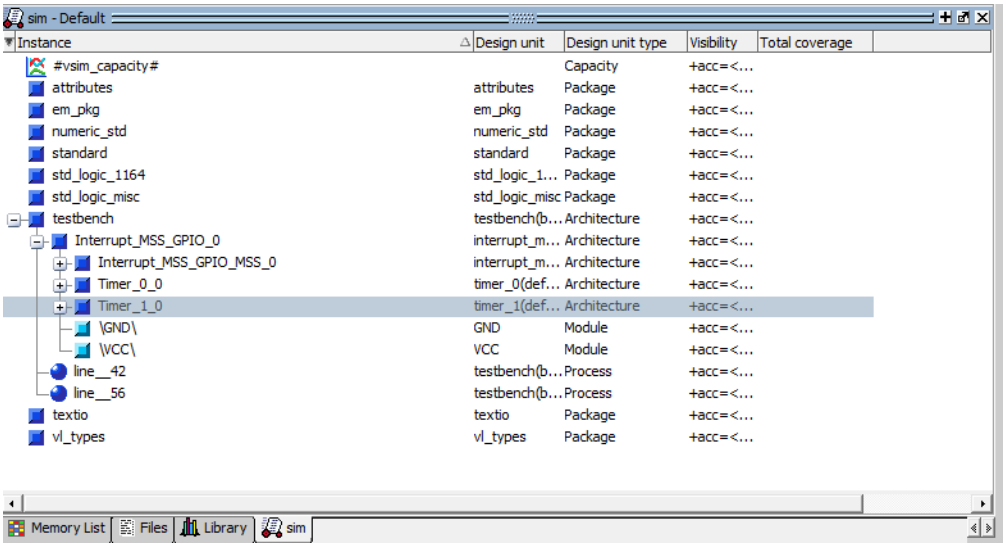


***Figure 4-5* • Timer_1_0 Selected in the ModelSim sim Tab**

6. Click the **Objects** tab. Ctrl+click to select **Aclr**, **Clock**, **Q** and **Tcnt**.

7. Add the signals to the Wave window: from the **ModelSim** menu, choose **Add > To Wave > Selected Signals** (Figure 4-6).

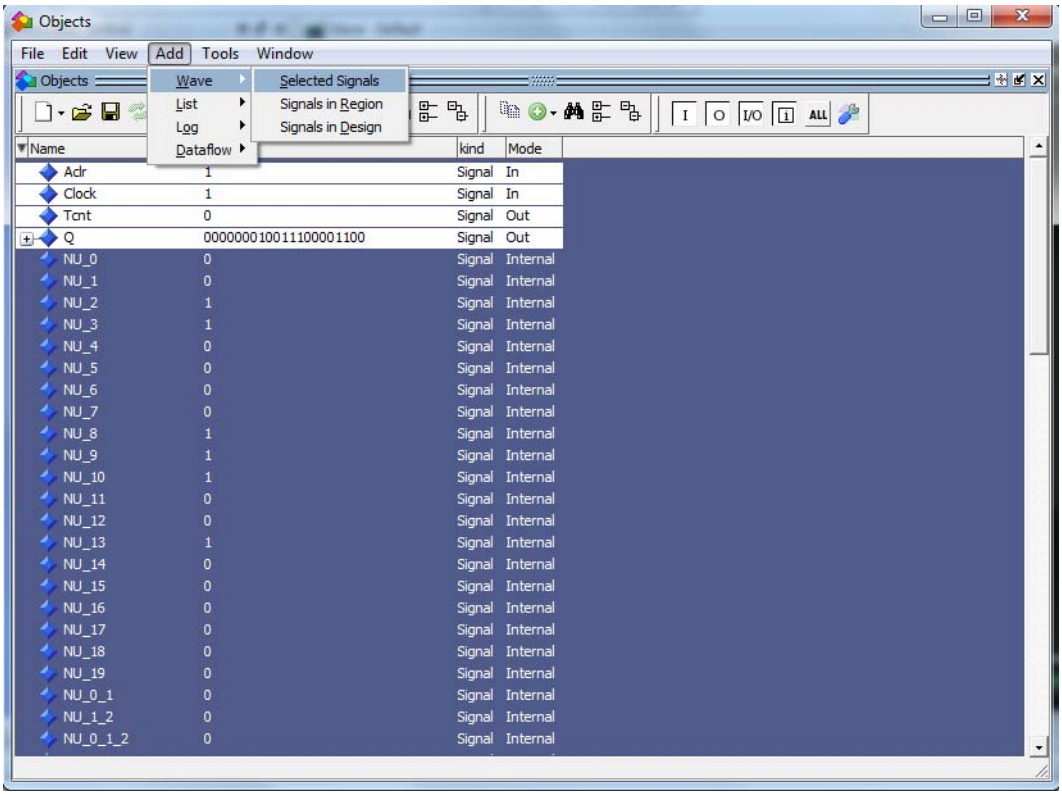


***Figure 4-6* • Adding Signals to the Wave Window**

8. Select the ModelSim **sim** tab (circled in Figure 4-4). Expand the Design Hierarchy and select **Timer_0_0**.
9. Click the **Objects** tab. Ctrl+click to select **Aclr**, **Clock**, **Q** and **Tcnt**.
10. Add the signals to the Wave window: from the **ModelSim** menu, choose **Add > To Wave > Selected Signals**.

    After the signals are added the Wave window should look like Figure 4-7.

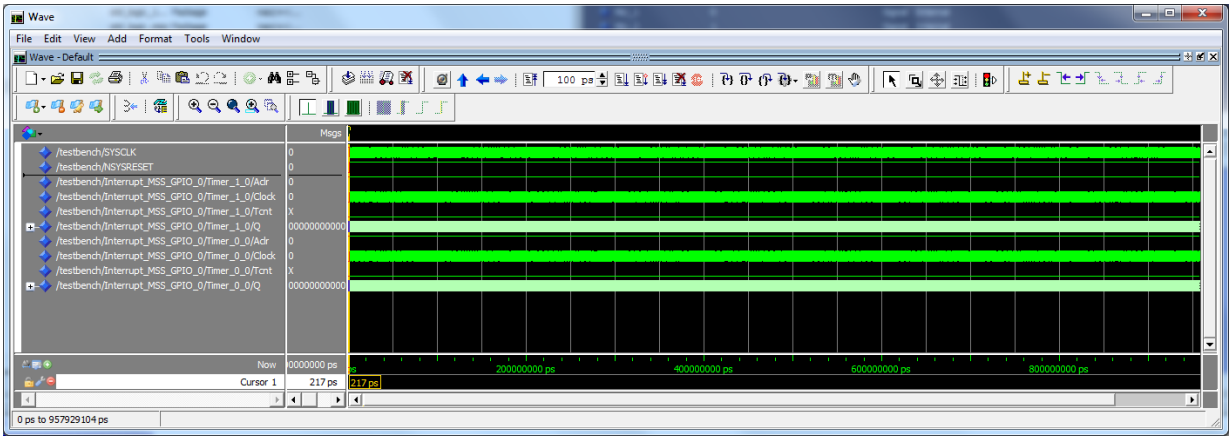


***Figure 4-7* • ModelSim Wave Window After Adding Timer_1_0 and Timer_0_0 Signals**

11. Observe the operation of the counters to confirm the design is working. Use the zoom buttons to zoom in and out as necessary (Figure 4-8).

    Un-dock the Wave window to make it easier to observe the signals.

    Change the radix of the Timer Q output to hex to make it easier to view the values.

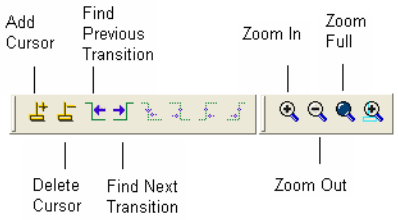


***Figure 4-8* • Wave Window Zoom Controls**

ModelSim displays waveforms of the Timers, as shown in Figure 4-9.

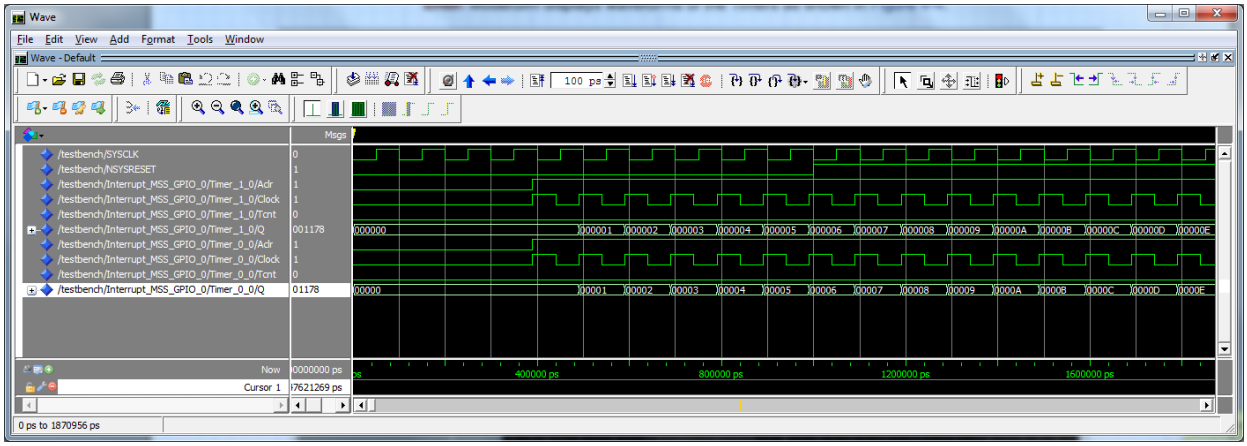


***Figure 4-9* • ModelSim Wave Window**

12. From the ModelSim toolbar **File** menu, choose **Quit** to exit the simulator. Click **Yes** when asked if you want to quit.

## Push-button Design Flow

Click **Generate Programming Data** to 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 interactively (step-by-step).

## Synthesizing the Design using Synplify® Pro

Synplify Pro compiles and synthesizes the design into an EDIF (*.edn) file. Your EDIF Netlist is then automatically translated by Libero SoC into an HDL Netlist. The resulting *.edn and *.vhd files are visible in the **Files** tab under **Synthesis**. Synthesis can be run in batch mode and GUI mode. If you double-click the Synthesis tool it defaults to batch mode.

1. In the Design Flow window right-click **Synthesis** and choose **Open Interactively** to launch Synplify Pro. Double-click Synthesis in the Design Flow window to run it automatically.

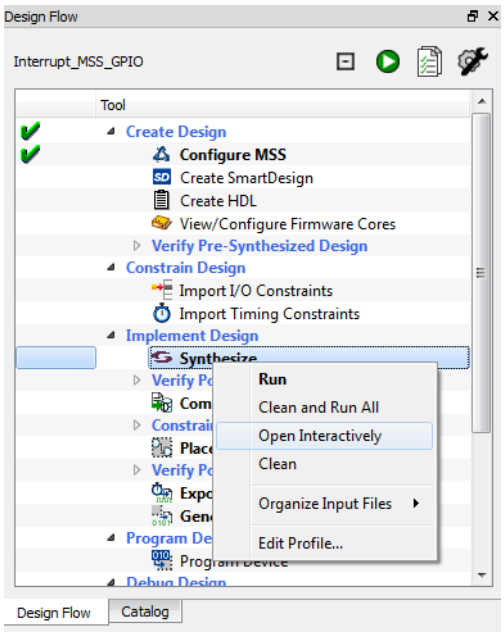


***Figure 4-10 •*** **Invoking the Synthesis Tool**

Change the frequency in the Synplify Pro GUI to any frequency depending on your design requirements (as shown in Figure 5-3 on page 31). In this design we are not changing the frequency.

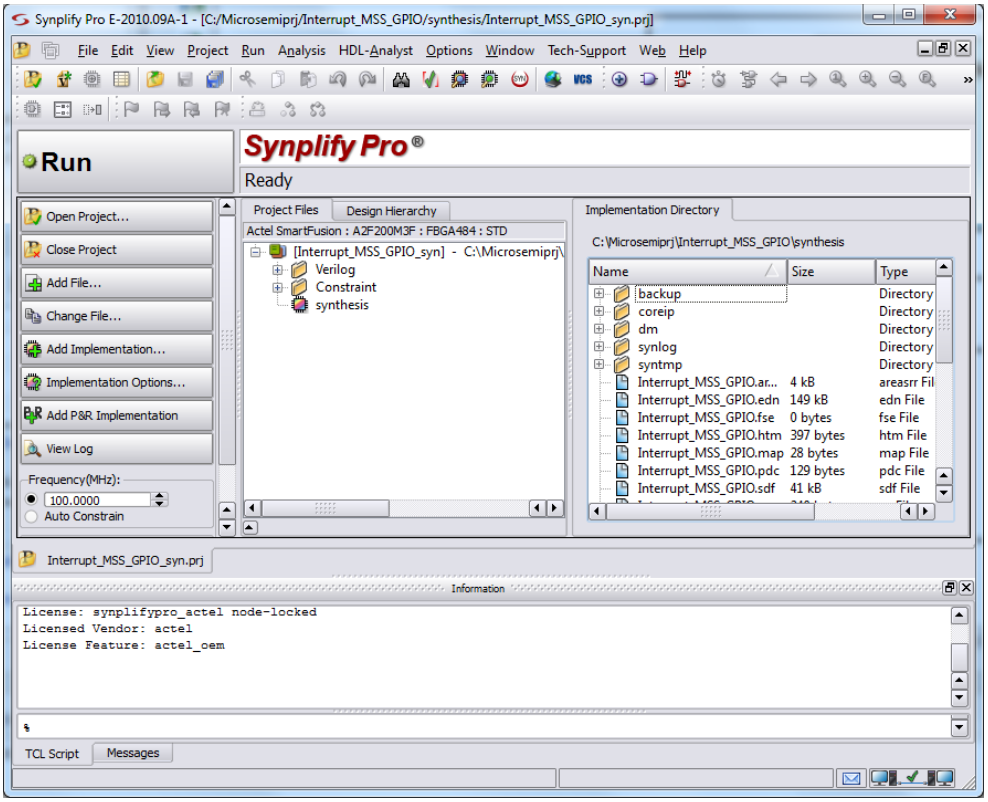


***Figure 4-11 •*** **Synthesis Tool GUI**

2. Click **Run** to map the design (Figure 4-11). When **Ready** in Synplify Pro changes to **Done**, the design has been mapped successfully.
3. From the File menu, choose Exit to close Synplify Pro. Click Yes if prompted to save changes to the project.

A green check mark adjacent to Synthesis in the Libero SoC design flow window indicates that the design was created without any errors.

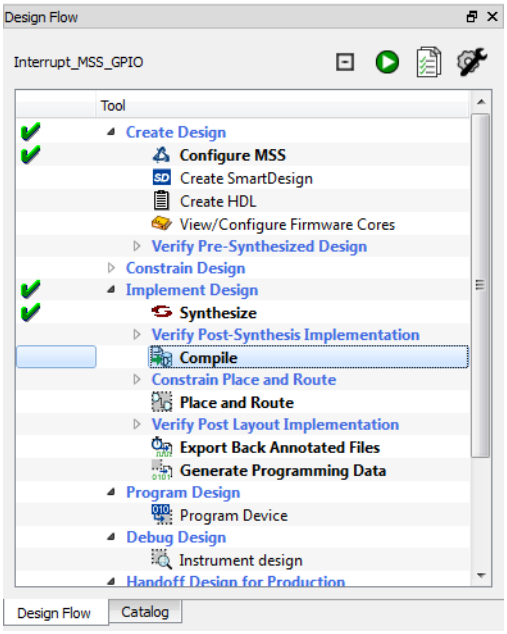


***Figure 4-12 •*** **Successful Synthesis**

# 5 – Step 5 - Implementing the Design

The next step is to implement the design. We need to compile the design, assign I/O and timing constraints, and run layout. Once the design is completely placed and routed, we will perform the static timing analysis and power analysis.

1. Right-click **Compile** in the Design Flow window and choose **Open Interactively** to invoke **Designer**.

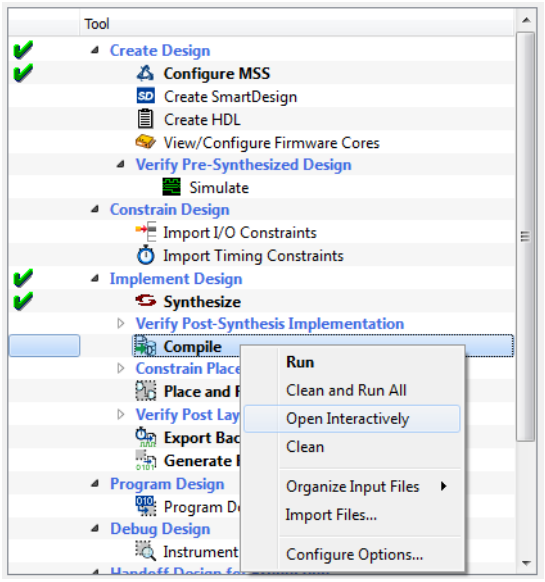


***Figure 5-1* • Opening Designer in GUI Mode**

2. Click **Compile** in **Designer** to compile the design. Compile contains a variety of functions that perform legality checking and basic Netlist optimization. It also calculates and displays the utilization of the design for the selected device.

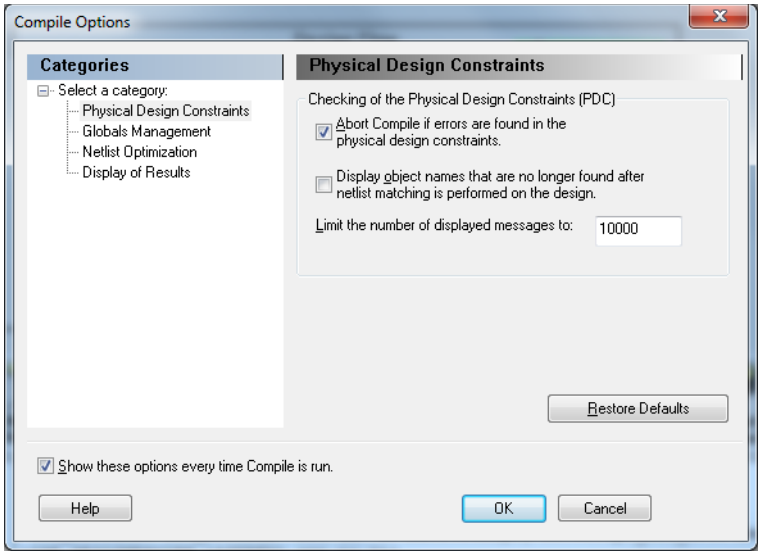


***Figure 5-2 •*** **Compile Options**

3. Click **OK** to close the **Compile Options** dialog box. The **Compile** button turns green to indicate that the design was compiled without any errors (Figure 5-3).

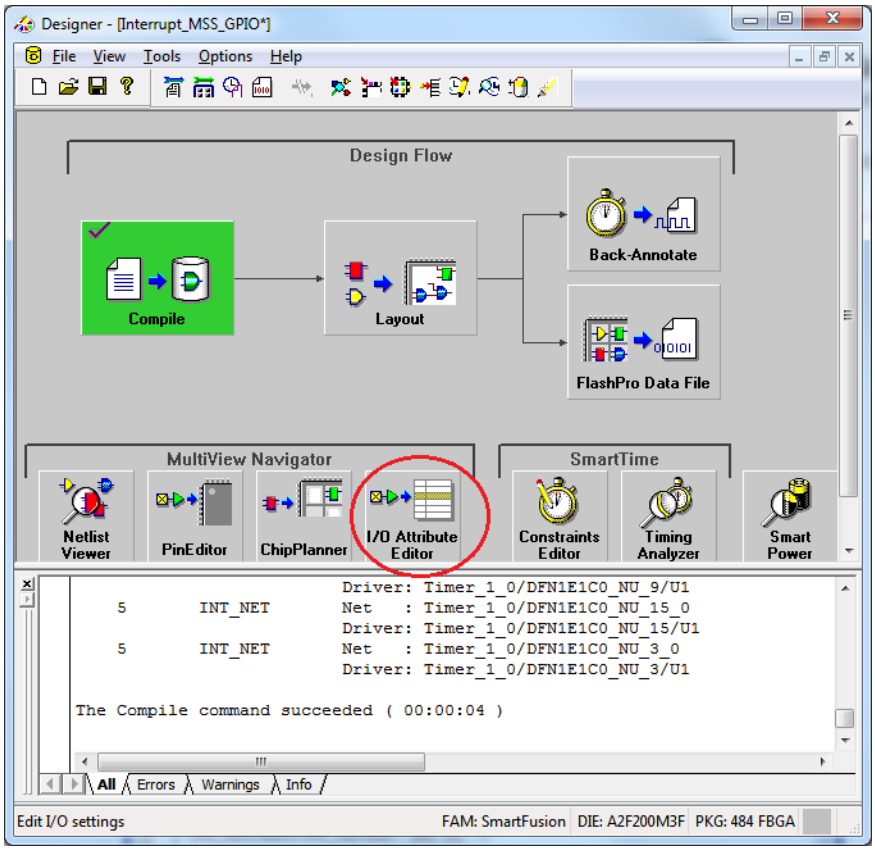


***Figure 5-3* • Designer - Compile Complete**

Designer includes an **I/O Attribute Editor** that enables you to make pin assignments and set I/O configurations for your design.

4. Click **I/O Attribute Editor** in Designer to open the I/O Attribute Editor in the MultiView Navigator.

   All ports are hard-wired except for F2M_GPI_1.

5. Click the **Pin Number** drop down menu and connect the **F2M_GPI_1** port to pin number **G19** (Figure 5-4).

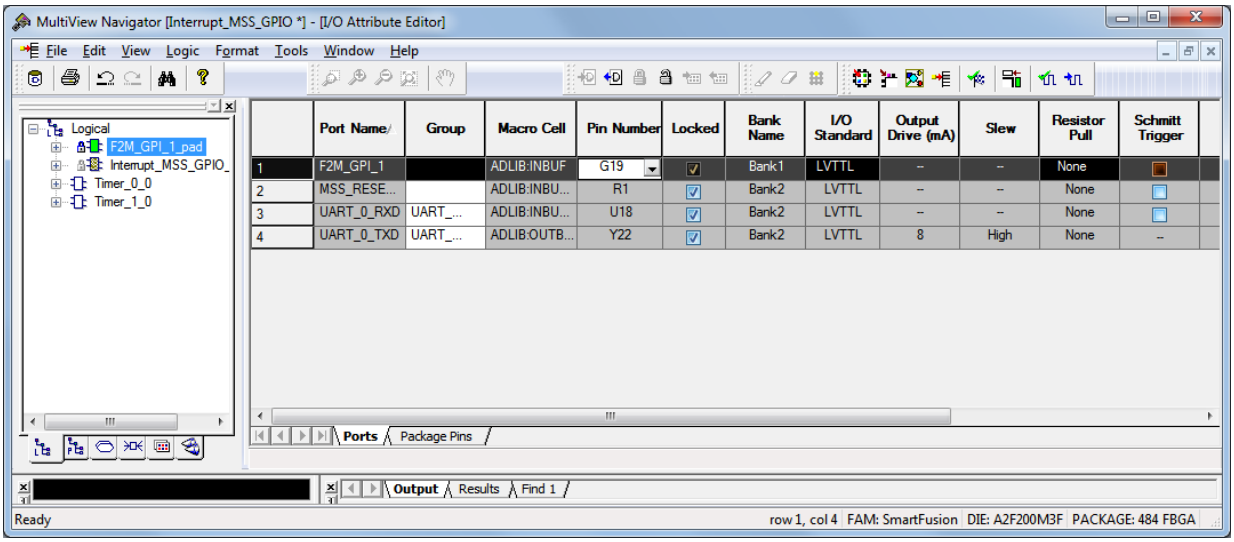


***Figure 5-4* • I/O Attribute Editor in MultiView Navigator**

6. In the **File** menu, choose **Commit and Check** to save the pin assignment. Correct any errors that are reported in the MVN Log window.
7. From the **File** menu, choose **Exit** to close MultiView Navigator.
8. Click **Layout** in Designer to run layout on the design. Click **OK** to accept the default Layout options (Figure 5-5)**.**

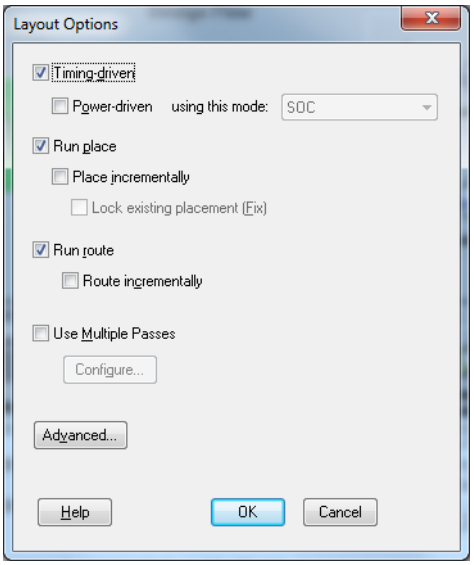


***Figure 5-5* • Layout Options**

The **Layout** button turns green to indicate that the design completed layout without any errors (Figure 5-6).

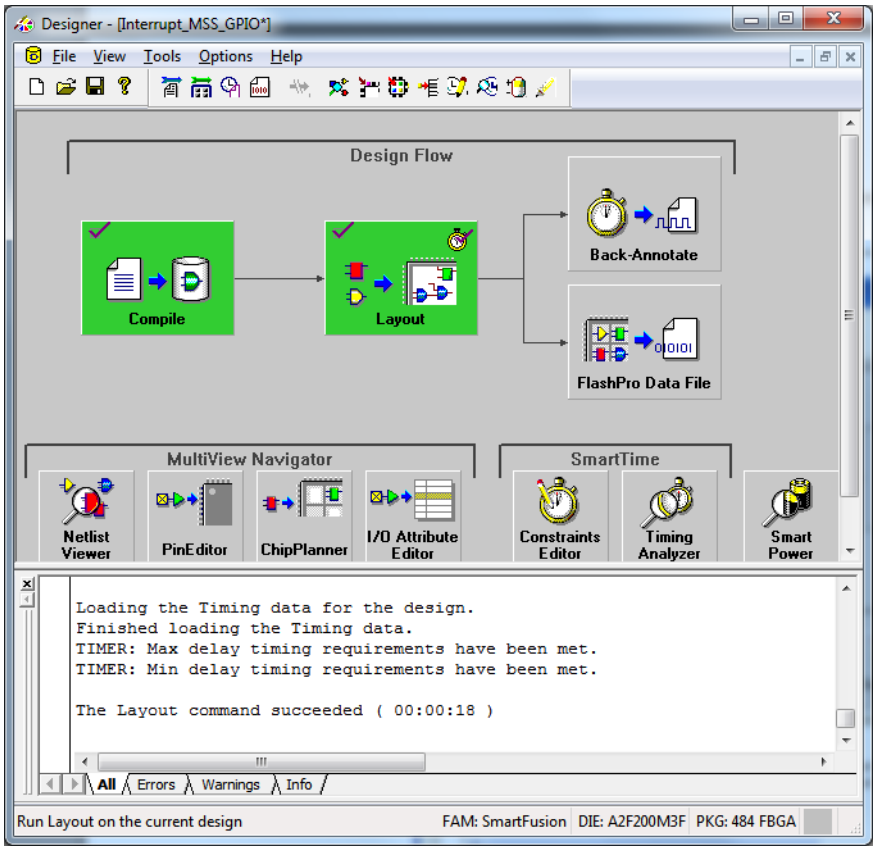


***Figure 5-6* • Successful Layout**

## Timing Analysis

The next step is to perform static timing analysis using SmartTime. SmartTime reads your design and displays post layout timing information (pre-layout, if invoked before place and route). SmartTime includes a **Constraints Editor** and a **Timing Analyzer**. By default the timing constraints from Synthesis would be imported into designer.

1. In the Designer GUI, click **Timing Analyzer** to open SmartTime. Confirm that the fabric clock (mss_ccc_glb) meets the timing requirement (10 MHz).

2. Select the **Register to Register** path set for the mss_ccc_glb clock domain (Figure 5-7).

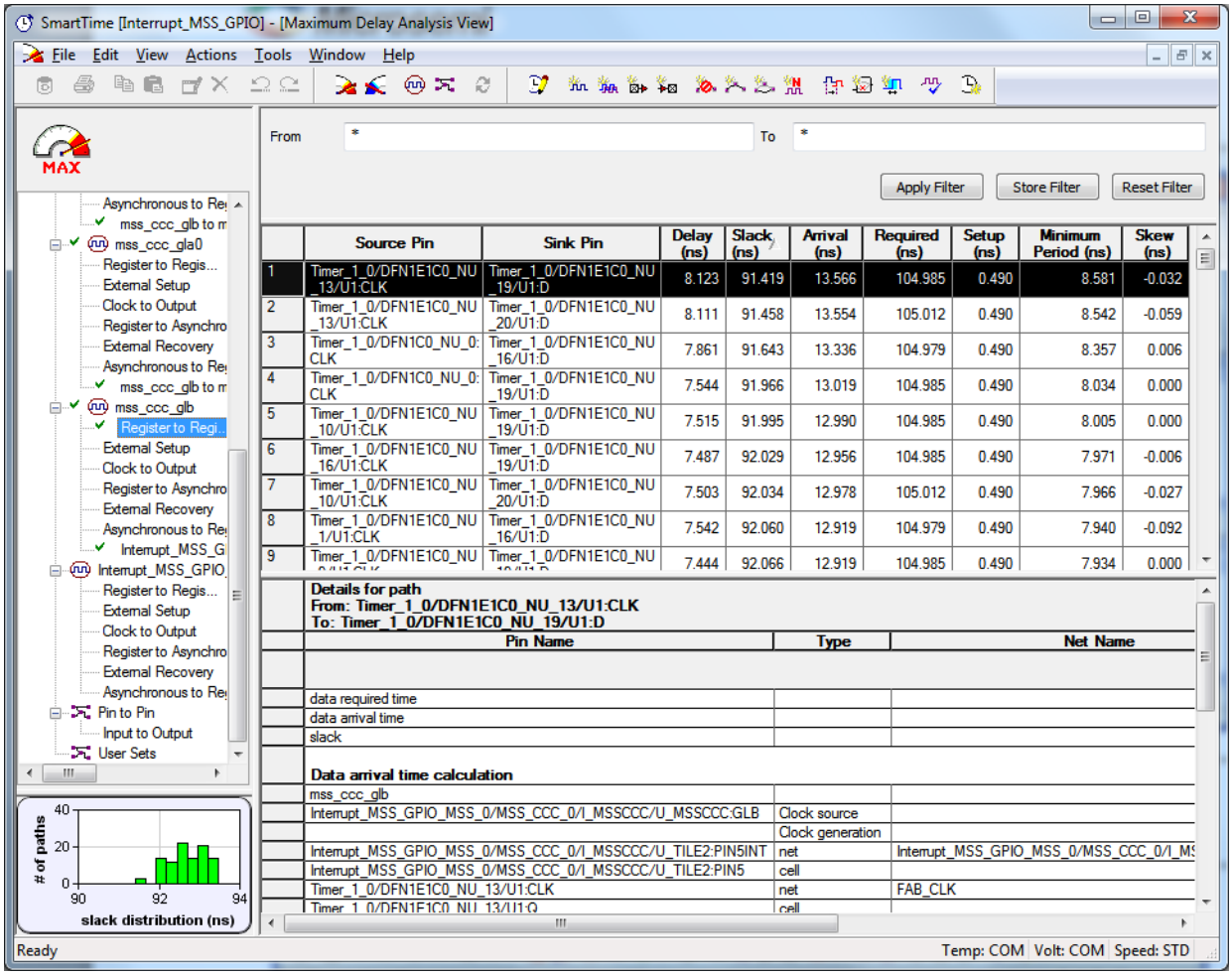


***Figure 5-7* • SmartTime Max Delay Timing Analysis**

3. Double-click one of the source pins to view detailed timing analysis for the selected path in the path details (as shown in Figure 5-8).

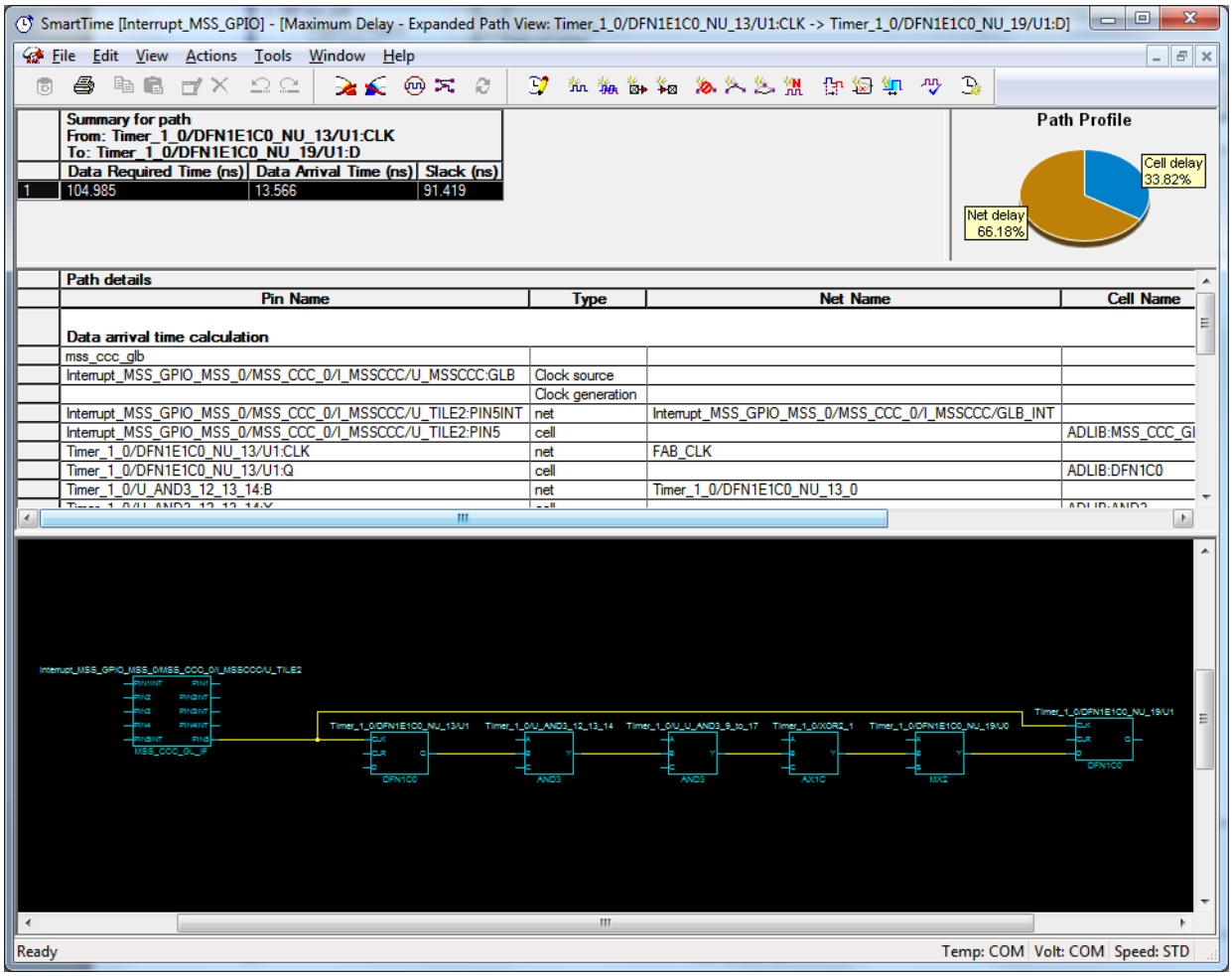


***Figure 5-8* • Detailed View of the Register Path in SmartTime**

4. From the **Tools** menu, choose **Timing Analyzer > Minimum Delay Analysis** to view the paths that have minimum delay.
5. From the **File** menu, choose **Exit** to close SmartTime.

## Power Analysis

SmartPower enables you to estimate the power consumption in your design.This enables you to make adjustments to reduce power consumption. Click the **SmartPower** button in Designer to start power analysis.

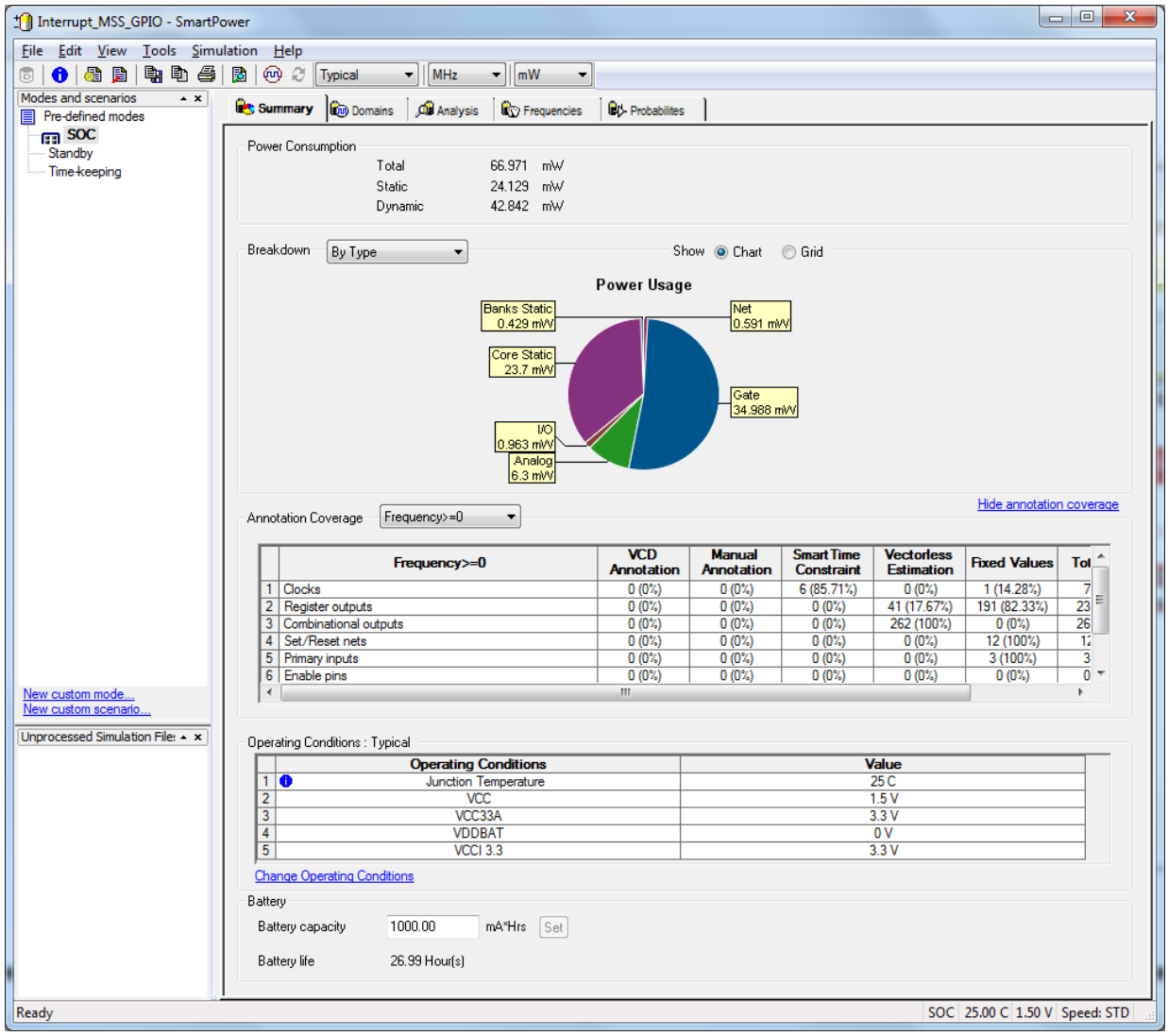


***Figure 5-9 •*** **SmartPower Analysis Window**

The constraints that we set in SmartTime are imported into SmartPower. SmartPower includes the following tabs (as shown in Figure 5-9:

- **Summary** - Displays the total power consumption, temperature and voltage operating conditions, battery capacity in mA/hr, and the projected battery life.
- **Domains** - Displays a list of existing domains with their corresponding clock and data frequencies. Use the **Domains** tab to set different clock frequencies and observe the effect on power consumption.
- **Analysis** - Displays detailed hierarchical reports of the power consumption.
- **Activity** - Used to attach switching activity attributes to the interconnects of the design.

## Back-Annotation

Back-Annotation generates a *.sdf file that contains timing information for your design. It is used for post-layout and timing-accurate simulation.

1. Click **Back-Annotate** to extract post layout timing delay from your design for simulation. The **Back-Annotate** dialog box opens as shown in Figure 5-10.

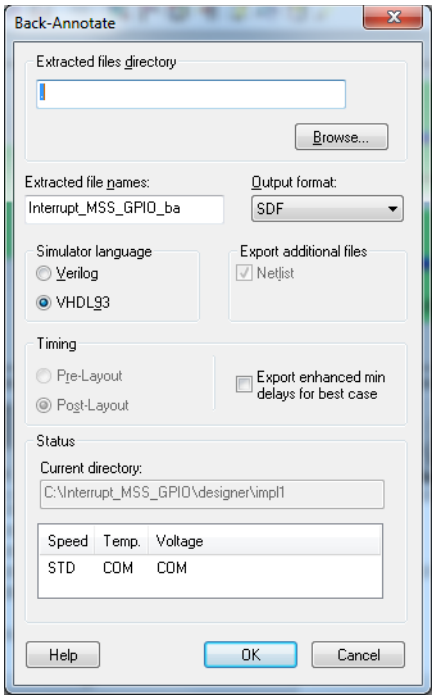


***Figure 5-10* • Back-Annotate Dialog Box**

2. Leave the default settings and click **OK** to continue. The Designer Log window shows whether or not the command succeeded. The Back-Annotate button in Designer turns green.
3. Save and Close Designer.
4. In the Libero **Design Flow** window, expand **Verify Post Layout Implementation**, right-click **Simulate** and choose **Open Interactively** to open ModelSim.
5. Add the signal **Timer_0_0/U_AND3_Tcnt/Y** to the Wave window, as shown in Figure 5-11. To do so:
   – In the ModelSim **Instance** window, click to select **Timer_0_0/U_AND3_Tcnt/Y**.

– Right-click **Y** in the **Objects** window and choose **Add > To Wave > Selected Signals**.

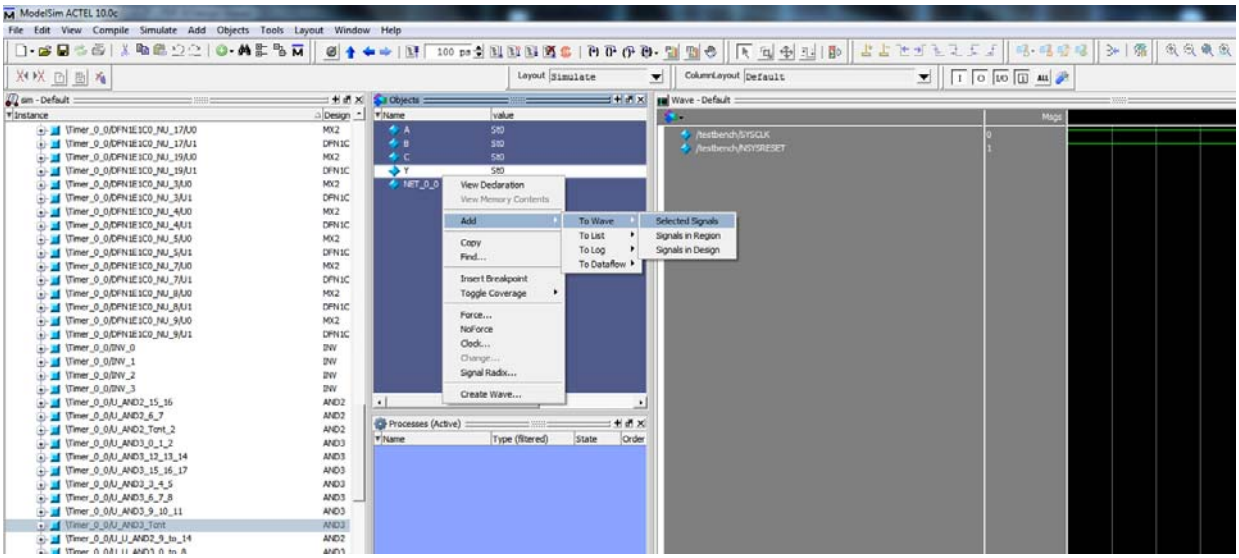


***Figure 5-11* • Adding Signals to the Wave Window in ModelSim**

6. Add the signal **Timer_1_0/U_AND3_Tcnt/Y** to the Wave window. To do so:
   – In the ModelSim **Instance** window, click to select **Timer_1_0/U_AND3_Tcnt/Y**.
   – Right-click **Y** in the **Objects** window and choose **Add > To Wave > Selected Signals**.
7. From the **Simulate** menu choose **Run > Run -All**.

   In ModelSim, let the simulation run for 100 - 200 ms of run time. You can stop the simulation from the ModelSim menu (**Simulate > Break**). Use the zoom buttons to change the scale and view details in the waveforms (Figure 5-12).

   Note that this simulation may take the full 200 ms of run time to show the interrupts.
8. Close ModelSim.

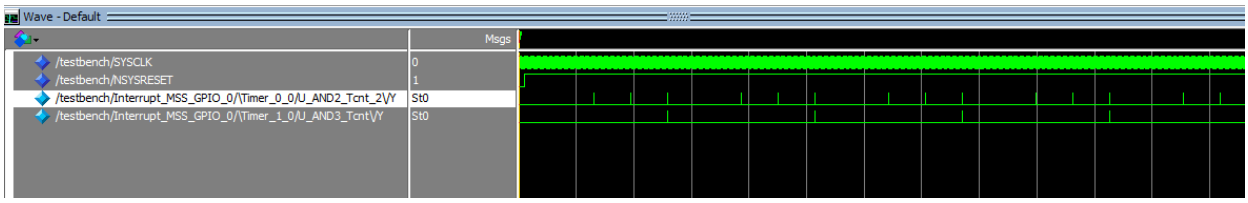


***Figure 5-12* • Post-Layout Simulation Window**

## *Generating a Programming File*

In the Design Flow window, double-click **Generate Programming Data**.

A green check mark indicates successful programming file generation, as shown in Figure 5-13. .

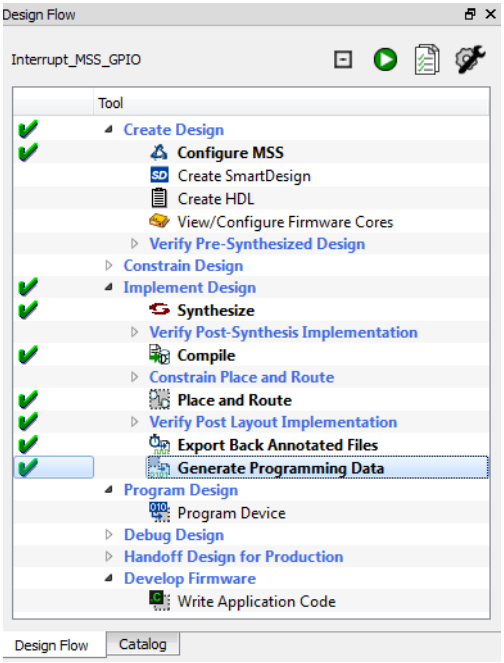


***Figure 5-13 •*** **Design Flow Window - Generate Programming Data**

## Programming

In this step you will launch FlashPro and program the SmartFusion device on the evaluation kit.

Please visit your specific board's documentation for instructions on how to set up the hardware for programming. Although the programming hardware may differ, the programming software is the same for FlashPro3 or FlashPro4.

1. Prior to programming (and powering up the board), verify that the jumpers are installed correctly on the SmartFusion Evaluation Kit Board:
   - JP7:Connect pins 1 and 2 (USB programming)
   - JP10: Connect pins 1 and 2 (FPGA)
   - J6: Connect pins 1 and 2 with the jumper
   - JP6: Connect pins 2 and 3 with the jumper (external 1.5 V mode)
2. Connect one of the mini USB cables between the USB PROG connector (J13) and a USB port on your PC. Install the FlashPro4 drivers if prompted. The drivers are located in the <drive>:\<Actel Libero SoC v10.0 installation>\Designer\Drivers folder.
3. Connect the second USB cable between the USB/UART connector (J14) and a USB port on your PC. Install the drivers for the USB to RS-232 Bridge if prompted.
4. In the Design Flow window expand **Program Design**.
   - To program the device directly double-click **Program Device**.

– To begin programming interactively, right-click **Program Device** and choose **Open Interactively (Figure 5-14)**.

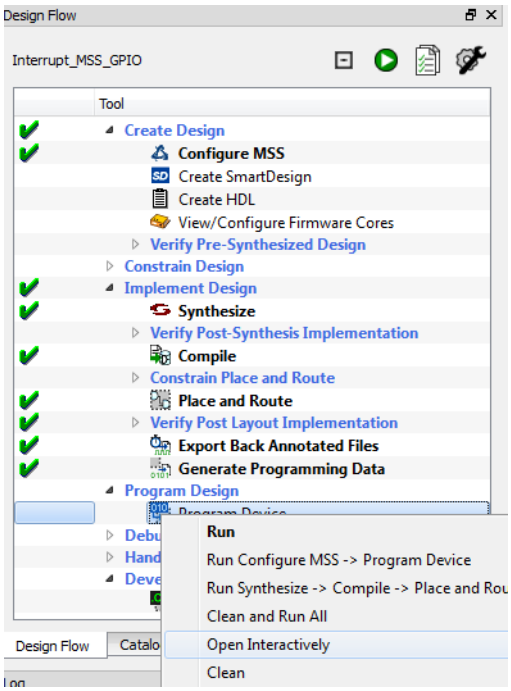


***Figure 5-14* • Invoking FlashPro in Interactive Mode**

FlashPro starts, as shown in Figure 5-15. When it opens the programmer loads the device and programming file automatically.

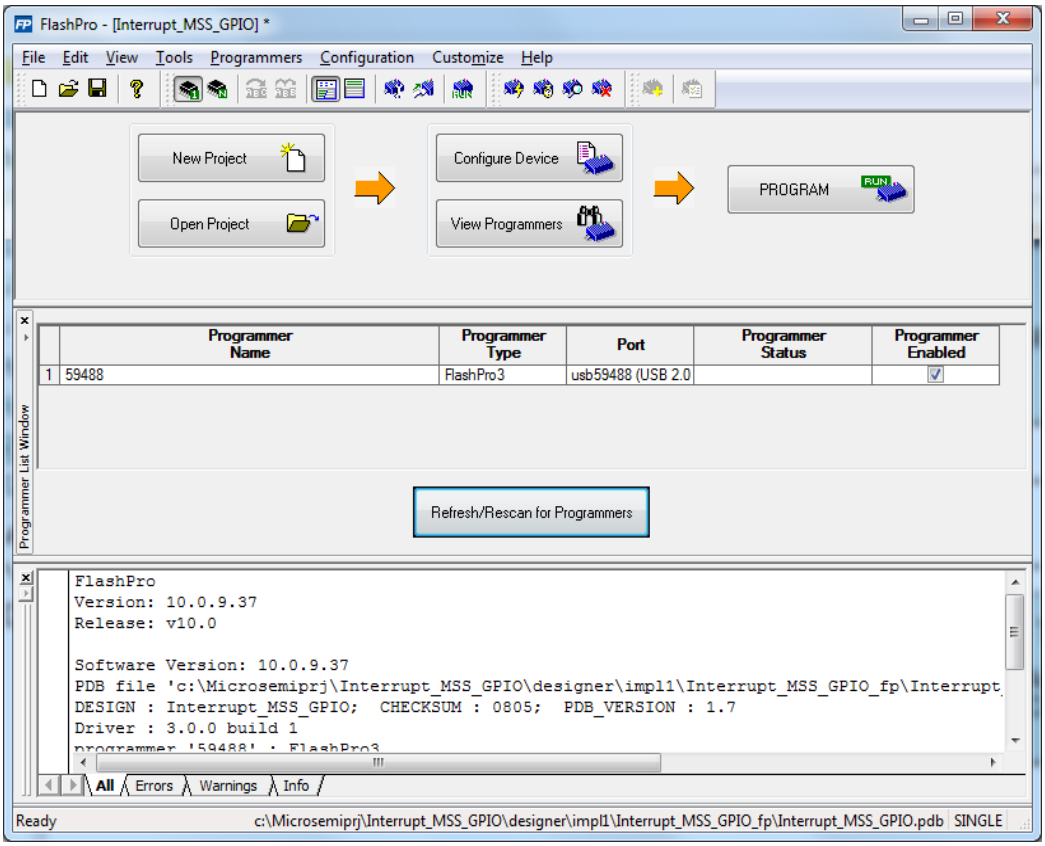


***Figure 5-15 •*** **FlashPro GUI**

5. Click **PROGRAM** and the programmer programs the device. Programming messages will be visible in the Libero SoC log window.

Do not interrupt the programming sequence; it may damage the device or the programmer.

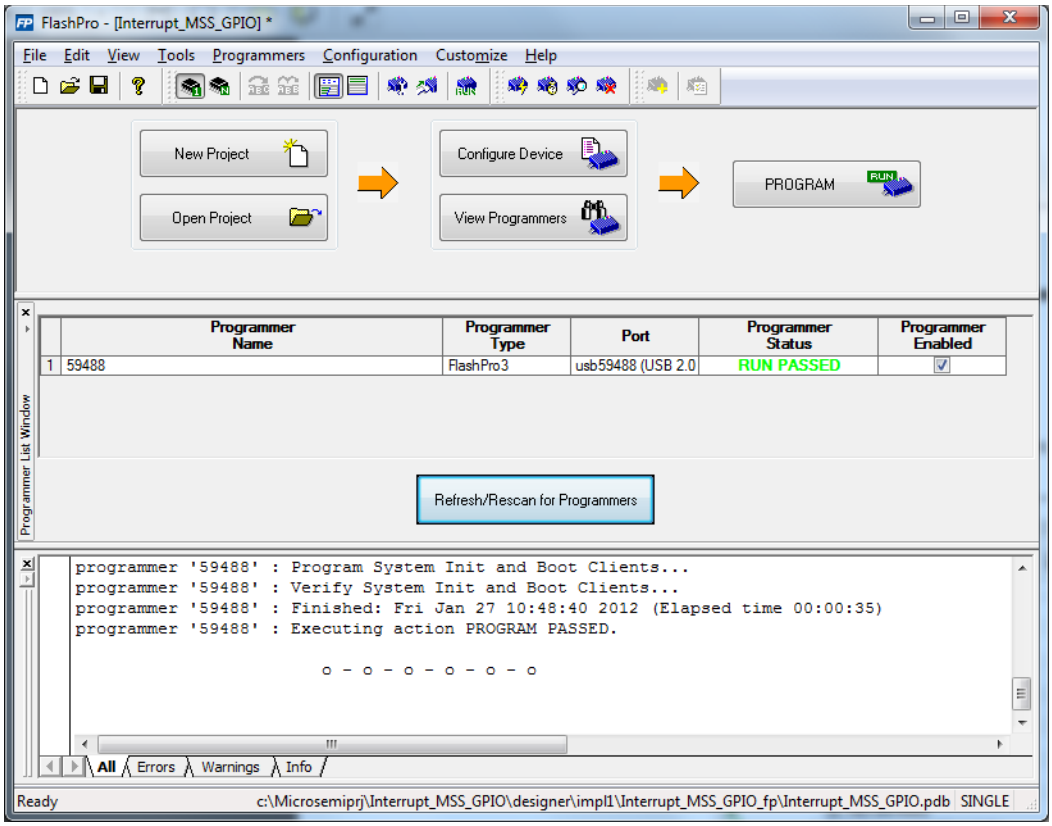


***Figure 5-16 •*** **Programming Messages in the Libero SoC Log Window**

The following message is visible in the Libero SoC log window when the device is programmed successfully: **Executing action PROGRAM PASSED.**

A green check mark next to **Program Design** and **Program Device** in the Design Flow window indicates that the programming completed successfully (Figure 5-17).

6. From the **File** menu, choose **Exit** to close FlashPro when programming is complete. Click **Yes** if prompted to save the project.

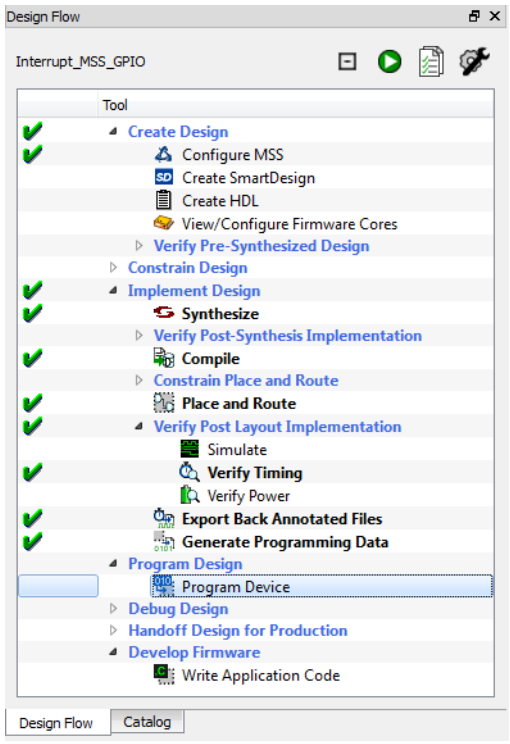


***Figure 5-17* • Design Flow Window After Programming**

# 6 – Software Implementation

## Step 1 – Invoking the SoftConsole from Libero SoC

1. In the Libero SoC Design Flow window, expand **Develop Firmware** and double-click **Write Application Code** (as shown in Figure 6-1).

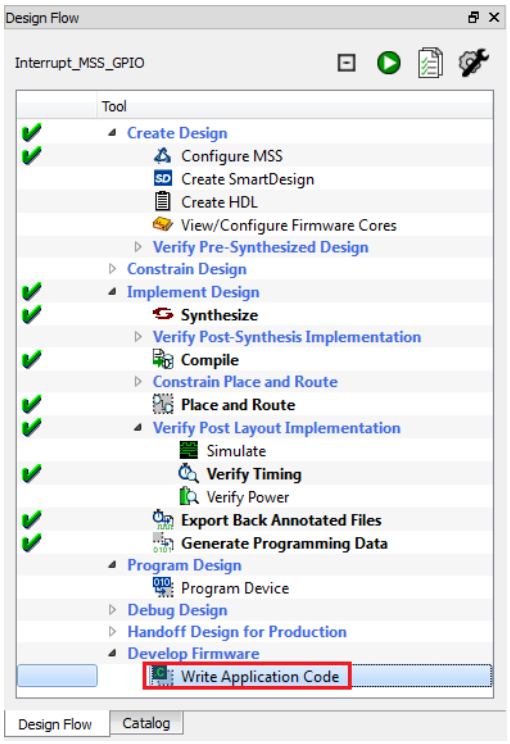


***Figure 6-1* • Invoking SoftConsole from Libero SoC**

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

Your SoftConsole looks like Figure 6-2.

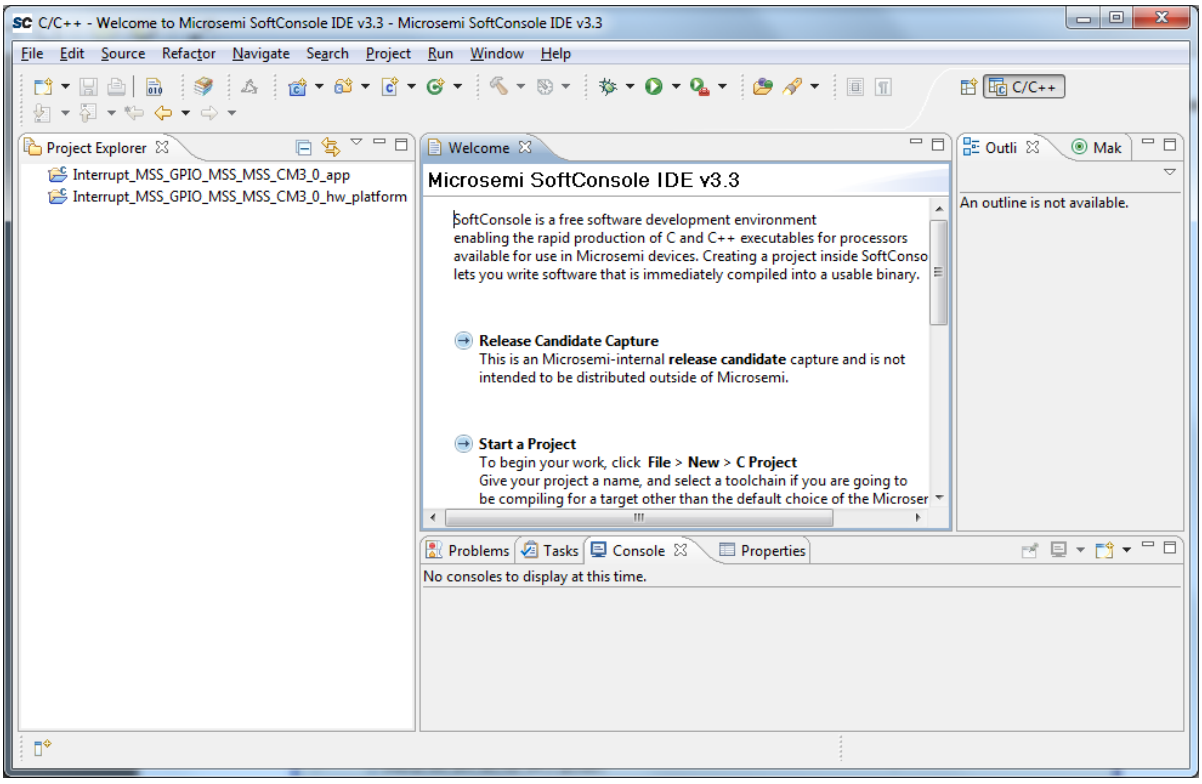


***Figure 6-2 •*** **SoftConsole Workspace**

2. Expand **Interrupt_MSS_GPIO_MSS_MSS_CM3_0_app** and double-click **main.c** to open it.
3. Delete the existing code in main.c and copy the code below and paste it in the **main.c** file.

```
/*******************************************************************************
***********/

// Including Directories
/*******************************************************************************
**********/
#include<stdio.h>
#include "mss_uart.h"
#include "mss_gpio.h"

/*******************************************************************************
**********
 Interrupt Handlers
*******************************************************************************
**********/

/* GPIO 0 Interrupt Handler */
/* This Interrupt handler executes upon the occurrence of
   GPIO 0 interrupt, which is from the timer 2 in the FPGA
   Fabric. This function prints source of the interrupt to the
   hyperterminal */
void GPIO0_IRQHandler( void )
  {


const uint8_t tim2[] = "\n\r Timer 2 Interrupt occurred - GPIO 0 \n\r";
```

```
MSS_UART_polled_tx (&g_mss_uart0, tim2, sizeof(tim2));
MSS_GPIO_clear_irq( MSS_GPIO_0 );
    NVIC_ClearPendingIRQ( GPIO0_IRQn );
  }


/* GPIO 1 Interrupt Handler */
/* This Interrupt handler executes upon the occurrence of
   GPIO 1 interrupt, which is from the Switch.
   This function prints source of the interrupt to the
   hyperterminal */

void GPIO1_IRQHandler( void )
  {
const uint8_t sw1[] = "\n\r Switch Interrupt occurred - GPIO 1 \n\r";
MSS_UART_polled_tx (&g_mss_uart0, sw1, sizeof(sw1));
MSS_GPIO_clear_irq( MSS_GPIO_1 );
    NVIC_ClearPendingIRQ( GPIO1_IRQn );

  }
/* Fabric Interrupt Handler */
/* This Interrupt handler executes upon the occurrence of
   fabric interrupt, which is from the timer 1.
   This function prints the source of the interrupt to the
   hyperterminal */
void Fabric_IRQHandler( void )
{
const uint8_t tim1[] = "\n\r Timer 1 Interrupt occurred - FABINT \n\r";
MSS_UART_polled_tx (&g_mss_uart0, tim1, sizeof(tim1));
NVIC_ClearPendingIRQ( Fabric_IRQn );

  }

/*******************************************************************************
**********
 Function Main
*******************************************************************************
**********/

int main()
{

/*UART initialization*/
MSS_UART_init(
&g_mss_uart0,
MSS_UART_57600_BAUD,
MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT );
/* Enabling of GPIO 0, GPIO 1 and Fabric Interrupts*/
    NVIC_EnableIRQ(GPIO0_IRQn);
MSS_GPIO_config( MSS_GPIO_0, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_POSITIVE );
MSS_GPIO_enable_irq( MSS_GPIO_0 );

NVIC_EnableIRQ(GPIO1_IRQn);
MSS_GPIO_config( MSS_GPIO_1, MSS_GPIO_INPUT_MODE | MSS_GPIO_IRQ_EDGE_NEGATIVE );
MSS_GPIO_enable_irq( MSS_GPIO_1 );
NVIC_EnableIRQ(Fabric_IRQn);

for(;;)
{

}
return 0;
}
/*******************************************************************************
**********
```

```
 End of function Main
******************************************************************************
**********/
```

4. Right-click the **Interrupt_MSS_GPIO_MSS_MSS_CM3_0_app** project and choose **Properties**.
5. Expand **C/C++ Build** and click **Settings**.
6. In the Tool Settings tab, under GNU C Linker click **Miscellaneous**, as shown in Figure 6-3.

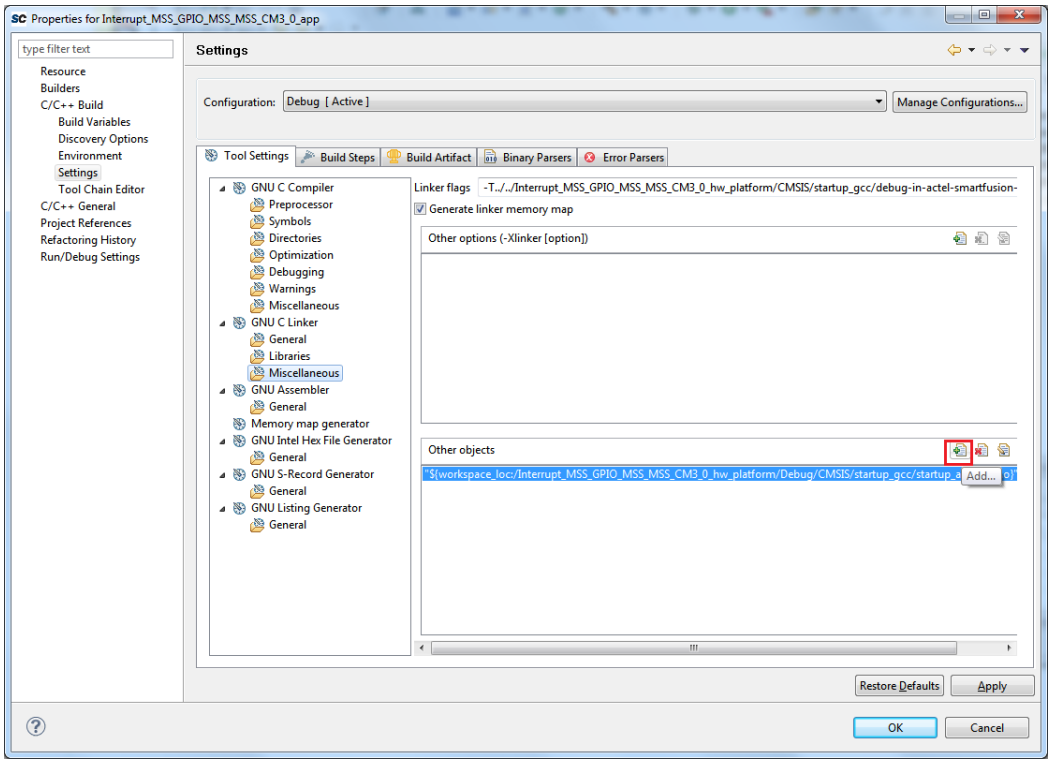


***Figure 6-3 •*** **Properties Window**

7. In the Other Objects window click **Add**. The **Add file path** dialog window appears, as shown in Figure 6-4.

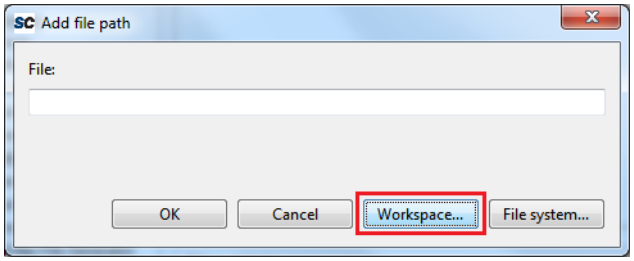


***Figure 6-4 •*** **Add File Path Window**

8. Click **Workspace** and add the **newlib_stubs.c** file under **Interrupt_MSS_GPIO_MSS_MSS_CM30_hw_platform > CMSIS > startup_gcc > newlib_stubs.c,** as shown in Figure 6-5.

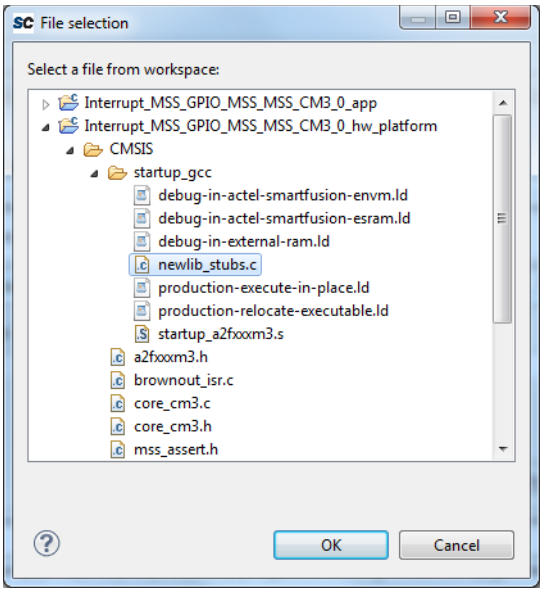


***Figure 6-5* • File Selection Window**

9. Click **OK**.
10. In the Properties window click **Apply**, then click **OK**.
11. From the **Project** menu choose **Clean** to perform a clean build. Accept the default settings in the **Clean** dialog box and click **OK**.

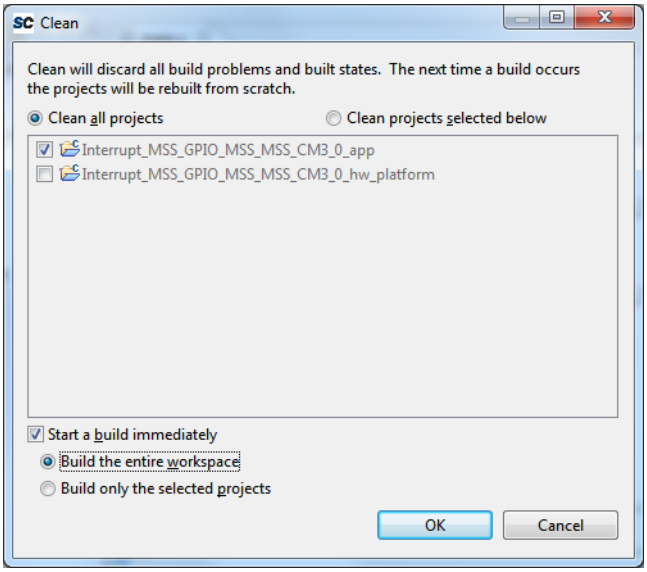


***Figure 6-6* • Settings for a Clean Build**

12. View the Problems tab to make sure there are no errors and warnings. Use the following steps to configure the HyperTerminal.

# Step 2 - 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:

1. Connect a second mini USB cable between the USB connector on the SmartFusion Evaluation Kit Board and a USB port of your computer. 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 6-7, proceed to the next step; otherwise follow the "Step 3 - Installing Drivers for the USB to RS232 Bridge" to install drivers for USB to RS232 Bridge.

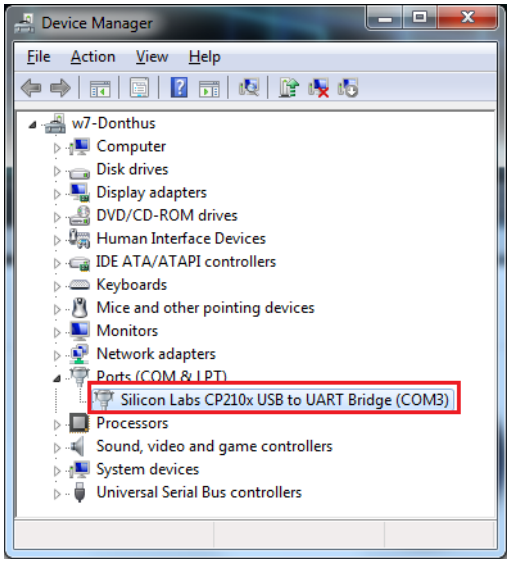


***Figure 6-7* • Device Manager Listing Silicon LabsCP210x USB to UART Bridge**

3. From the Windows **Start** menu, choose **Programs > Accessories > Communications > HyperTerminal**. This opens HyperTerminal. If your PC does not have HyperTerminal, use any free serial terminal emulation program, such as PuTTY or Tera Term. Refer to the *Configuring Serial Terminal Emulation Programs* tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.

4. Enter **Hyperterminal** in the **Name** field in the **Connection Description** dialog box and click **OK**.


***Figure 6-8* • New Connection**

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


***Figure 6-9* • Selecting the COM Port**

6. 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

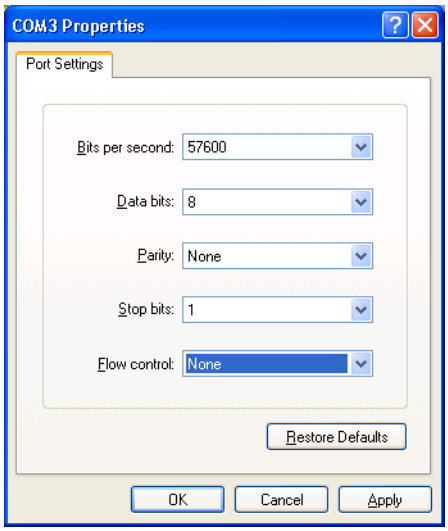


***Figure 6-10* • Setting the COM Properties**

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

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

# Step 3 - Installing Drivers for the USB to RS232 Bridge

Note: You must have Admin privileges on your PC to install the USB-RS232 drivers.

To install drivers for the USB to RS232 Bridge:

1. Download the USB to RS232 bridge drivers from www.microsemi.com/soc/documents/CP2102_driver.zip.
2. Unzip the **cp2102_drivers.zip** file.
3. Double-click (**Run**) the ***.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.

# Step 4 - Debugging the Application Project Using SoftConsole

To debug the application project using SoftConsole:

1. From the **Run** menu in SoftConsole choose **Debug Configurations**. The Debug dialog box appears.

2. Double-click **Microsemi Cortex-M3 RAM target** to display the Options for the RAM target (Figure 6-11).

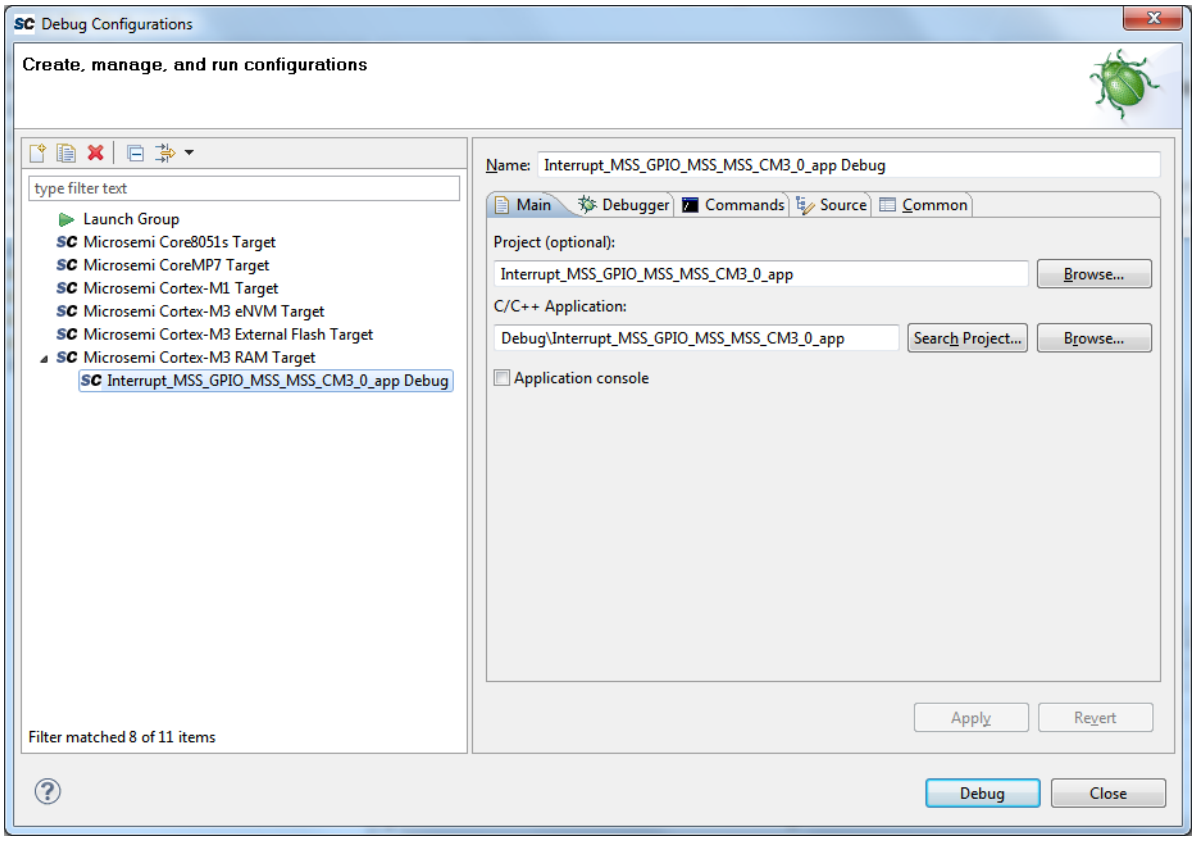


***Figure 6-11* • Debug Configurations Dialog Box**

3. Confirm that the following appear on the **Main** tab in the **Debug Configuration** dialog box:
   - **Name**: Interrupt_MSS_GPIO_MSS _MSS_CM3_app Debug
   - **Project**: Interrupt_MSS_GPIO_MSS _MSS_CM3_app
   - **C/C++ Application**: Debug\ Interrupt_MSS_GPIO_MSS _MSS_CM3_app
4. Click **Apply** and **Debug**.
5. Click **Yes** when prompted for **Confirm Perspective Switch** (Figure 6-12). This displays the Debug view mode.

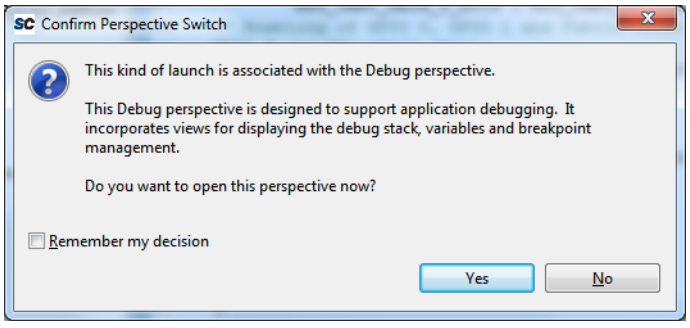


***Figure 6-12* • Confirm Perspective Switch**

Your Debug Perspective should resemble Figure 6-13:

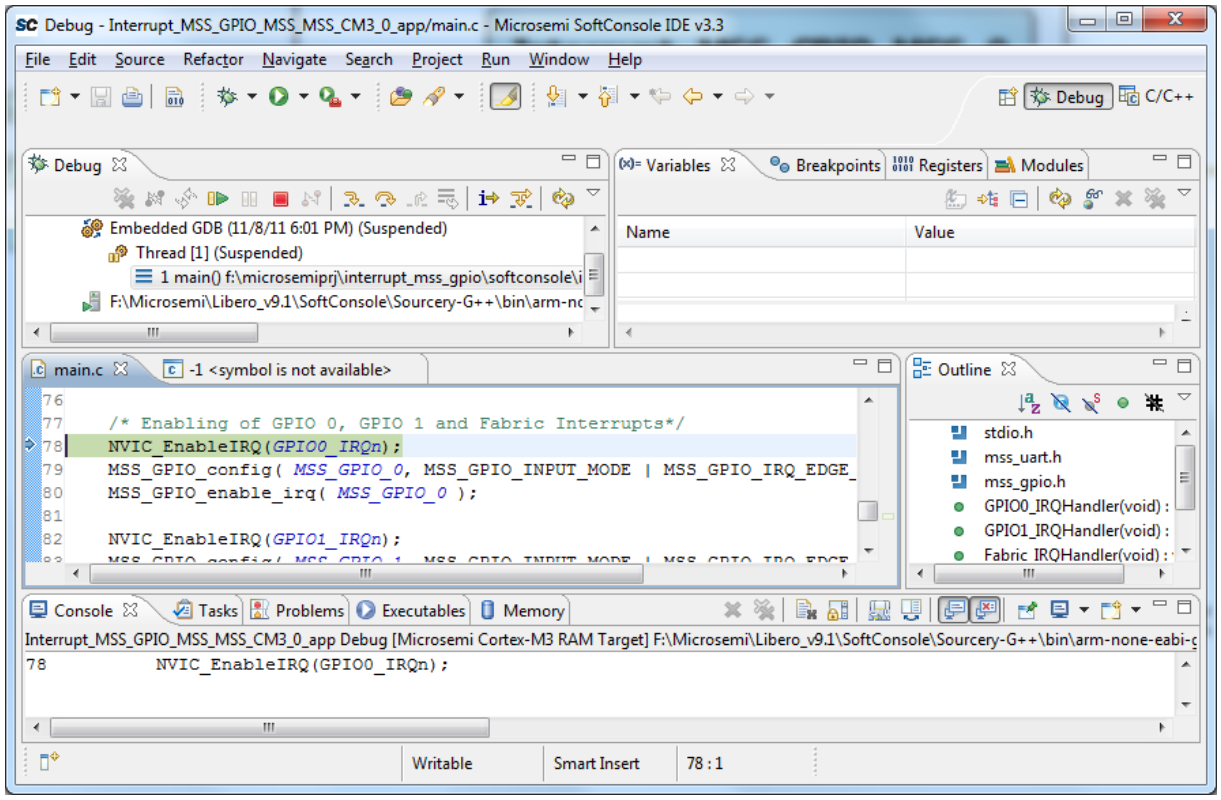


***Figure 6-13* • Debug Perspective**

6. Click the **Resume** icon on the SoftConsole toolbar to run the application.

The interrupt sequence messages are displayed in the terminal program window. Press the SW1 switch on the SmartFusion Evaluation Kit and observe the message in the serial terminal that indicates the switch interrupt occurred (Figure 6-14).

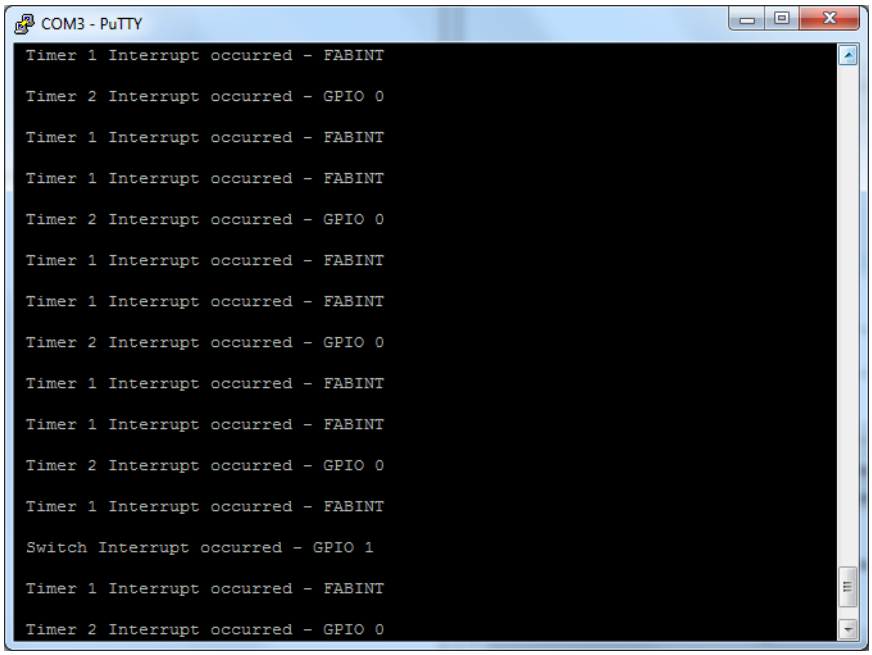


***Figure 6-14 •*** **PuTTy**

# Step 5 - Building an 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 the SmartFusion cSoC device. You can load the application executable image into eNVM with the help of the eNVM data storage client from SmartDesign MSS Configurator and in-application programming (IAP) or FlashPro programming software. In release mode, you cannot use the 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*.

Congratulations! You have completed the Libero SoC tutorial.

# A – Product Support

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

## Customer Service

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

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 650.318.8044

## Customer Technical Support Center

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

## Technical Support

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

## Website

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

## Contacting the Customer Technical Support Center

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

### Email

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

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

## My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

## Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

# ITAR Technical Support

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


**Microsemi Corporate Headquarters**
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

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**.



5-02-9123-18/01.12