Accessing Serial Flash Memory Using SPI Interface - Libero SoC v11.7 and IAR Embedded Workbench Flow Tutorial for SmartFusion2

TU0547 Tutorial

Microsemi®

Power Matters.™

Contents

Figures

Tables

1 Preface

1.1 Purpose

This document contains the following sections:

- Accessing Serial Flash Memory using SPI Interface - Libero SoC v11.7 and IAR Embedded Workbench Flow Tutorial for SmartFusion2

1.2 Intended Audience

Triton devices are used by:

- FPGA designers
- Embedded designers
- System-level designers

2 Accessing Serial Flash Memory using SPI Interface - Libero SoC v11.7 and IAR Embedded Workbench Flow Tutorial for SmartFusion2

2.1 Introduction

The Libero® System-on-Chip (SoC) software generates firmware projects using IAR, Keil, and SoftConsole tools. This tutorial describes the process to build an IAR application that can be implemented and validated using the SmartFusion®2 SoC field programmable gate array (FPGA) Security Evaluation Kit.

The same firmware project can be built using SoftConsole and Keil tools. Refer to the respective tutorials:

- *TU0546: SoftConsole v4.0 and Libero SoC v11.7*
- *TU0548: Accessing Serial Flash Memory Using SPI Interface - Libero SoC and Keil uVision Flow Tutorial for SmartFusion2 SoC FPGA*

After completing this tutorial, you will be able to perform the following tasks:

- Creating a Libero SoC project using System Builder
- Generating the programming file to program the SmartFusion2 device
- Opening the project in IAR Embedded Workbench from Libero SoC
- Compiling application code
- Debugging and run code using IAR Embedded Workbench

2.2 Design Requirements

Table 1 • **Design Requirements**

Design Requirements	**Description**
Hardware Requirements	
SmartFusion2 Security Evaluation Kit • FlashPro4 programmer • J-Link programmer • USB A to Mini-B cable • 12 V Adapter	Rev D or later
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.7
FlashPro programming software	v11.7
IAR Embedded Workbench for ARM	v6.4
Host PC Drivers	USB to UART drivers
SoftConsole	v3.4 SP1* *Note:* **For this application note, SoftConsole v3.4 SP1 is used. For using SoftConsole v4.0, see the TU0546: SoftConsole v4.0 and Libero SoC v11.7 Tutorial.*
One of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	–

2.2.1 Project Files

The design files for this tutorial can be downloaded from the Microsemi® website:
http://soc.microsemi.com/download/rsc/?f=m2s_tu0547_liberov11p7_df

The design files include:

- Liberoproject
- Programmingfiles
- Source files
- SPI_Flash_Drivers
- Readme files

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

2.3 Design Overview

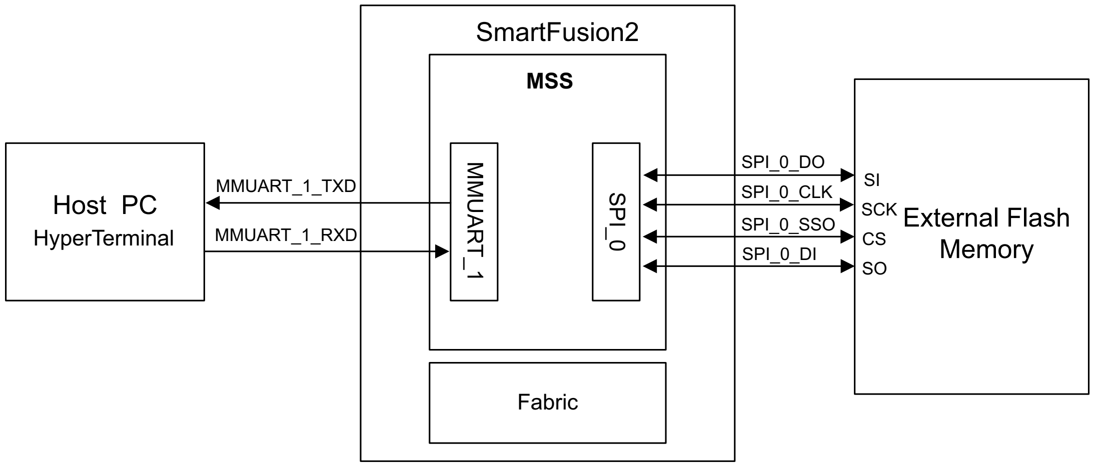
This design example demonstrates the execution of basic read and write operations on the SPI flash present on the SmartFusion2 Security Evaluation Kit board. This kit has a built-in winbond SPI flash memory, named W25Q64FVSSIG, which is connected to the SmartFusion2 microcontroller subsystem (MSS) through the dedicated MSS SPI_0 interface.

Read and write data information is displayed using HyperTerminal, which communicates to the SmartFusion2 MSS using the MMUART_1 interface.

For more information on SPI, refer to the *UG0331: SmartFusion2 Microcontroller Subsystem User Guide*.

Figure 1 shows interfacing the external SPI flash to MSS SPI_0.

Figure 1 • **SPI Flash Interfacing Block Diagram**



2.4 Step 1: Creating a Libero SoC Project

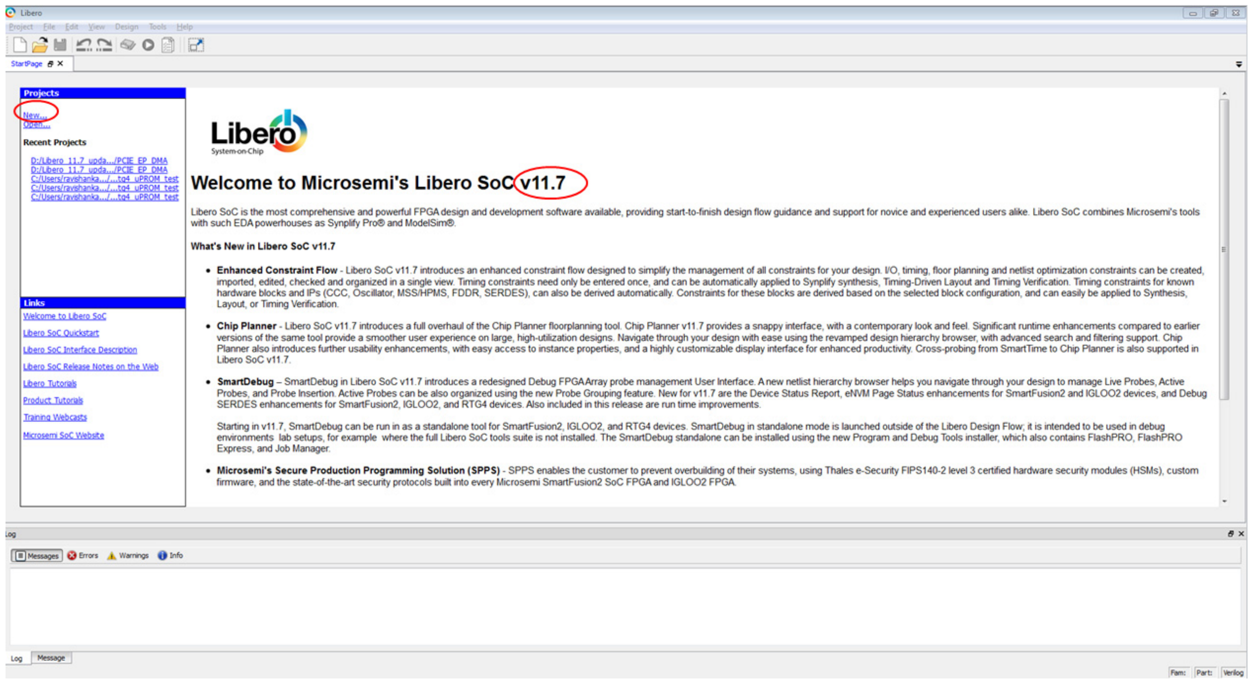
The following steps describe how to create a Libero SoC project:

2.4.1 Launching Libero SoC

The following steps describe how to launch Libero SoC:

1. Click **Start > Programs > Microsemi Libero SoC v11.7 > Libero SoC v11.7**, or double-click the shortcut on desktop to open the Libero SoC v11.7 Project Manager.
2. Create a new project using one of the following options:
   - Select **New** on the **Start Page** tab as shown in Figure 2.
   - Click **Project > New Project** from the Libero SoC menu.

Figure 2 • **Libero SoC Project Manager**


Libero
Project File Edit View Design Tools Help
StartPage
Projects
New...
Open...
Recent Projects
D:/Libero_11.7_upda.../PCIE_EP_DMA
D:/Libero_11.7_upda.../PCIE_EP_DMA
C:/Users/ravishanka.../...tg4_uPROM_test
C:/Users/ravishanka.../...tg4_uPROM_test
C:/Users/ravishanka.../...tg4_uPROM_test
Links
Welcome to Libero SoC
Libero SoC Quickstart
Libero SoC Interface Description
Libero SoC Release Notes on the Web
Libero Tutorials
Product Tutorials
Training Webcasts
Microsemi SoC Website
Libero System-on-Chip
Log
Messages
Errors
Warnings
Info
Log
Message
Fam:
Part:
Verilog

Welcome to Microsemi's Libero SoC v11.7

Libero SoC is the most comprehensive and powerful FPGA design and development software available, providing start-to-finish design flow guidance and support for novice and experienced users alike. Libero SoC combines Microsemi's tools with such EDA powerhouses as Synplify Pro® and ModelSim®.

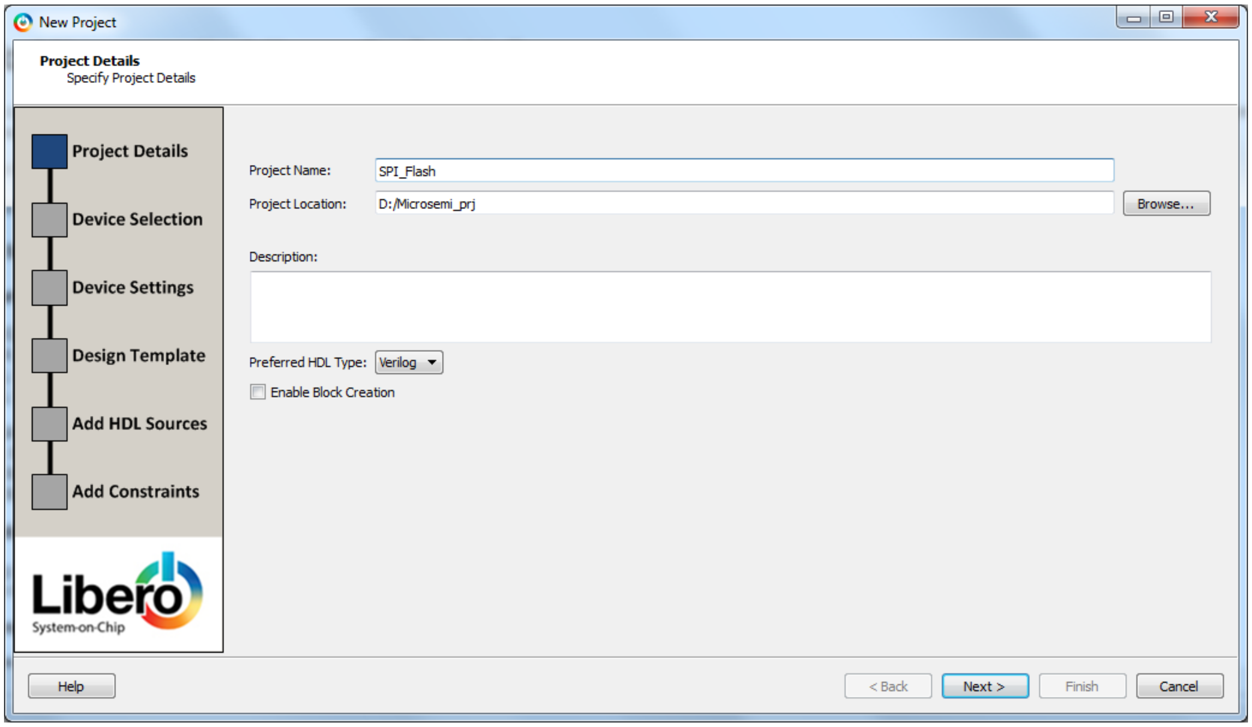
What's New in Libero SoC v11.7

- **Enhanced Constraint Flow** - Libero SoC v11.7 introduces an enhanced constraint flow designed to simplify the management of all constraints for your design. I/O, timing, floor planning and netlist optimization constraints can be created, imported, edited, checked and organized in a single view. Timing constraints need only be entered once, and can be automatically applied to Synplify synthesis, Timing-Driven Layout and Timing Verification. Timing constraints for known hardware blocks and IPs (CCC, Oscillator, MSS/HPMS, FDDR, SERDES), can also be derived automatically. Constraints for these blocks are derived based on the selected block configuration, and can easily be applied to Synthesis, Layout, or Timing Verification.
- **Chip Planner** - Libero SoC v11.7 introduces a full overhaul of the Chip Planner floorplanning tool. Chip Planner v11.7 provides a snappy interface, with a contemporary look and feel. Significant runtime enhancements compared to earlier versions of the same tool provide a smoother user experience on large, high-utilization designs. Navigate through your design with ease using the revamped design hierarchy browser, with advanced search and filtering support. Chip Planner also introduces further usability enhancements, with easy access to instance properties, and a highly customizable display interface for enhanced productivity. Cross-probing from SmartTime to Chip Planner is also supported in Libero SoC v11.7.
- **SmartDebug** – SmartDebug in Libero SoC v11.7 introduces a redesigned Debug FPGAArray probe management User Interface. A new netlist hierarchy browser helps you navigate through your design to manage Live Probes, Active Probes, and Probe Insertion. Active Probes can be also organized using the new Probe Grouping feature. New for v11.7 are the Device Status Report, eNVM Page Status enhancements for SmartFusion2 and IGLOO2 devices, and Debug SERDES enhancements for SmartFusion2, IGLOO2, and RTG4 devices. Also included in this release are run time improvements.

  Starting in v11.7, SmartDebug can be run in as a standalone tool for SmartFusion2, IGLOO2, and RTG4 devices. SmartDebug in standalone mode is launched outside of the Libero Design Flow; it is intended to be used in debug environments lab setups, for example where the full Libero SoC tools suite is not installed. The SmartDebug standalone can be installed using the new Program and Debug Tools installer, which also contains FlashPRO, FlashPRO Express, and Job Manager.
- **Microsemi's Secure Production Programming Solution (SPPS)** - SPPS enables the customer to prevent overbuilding of their systems, using Thales e-Security FIPS140-2 level 3 certified hardware security modules (HSMs), custom firmware, and the state-of-the-art security protocols built into every Microsemi SmartFusion2 SoC FPGA and IGLOO2 FPGA.

3. Enter the following information in the **Project Details** page, as shown in Figure 3.
   - **Project Name**: SPI_Flash
   - **Project Location**: Select an appropriate location (for example, *D:/Microsemi_prj*)
   - **Preferred HDL Type**: Verilog
   - **Enable Block Creation**: Unchecked

Figure 3 • **Project Details Page**



Project Name: SPI_Flash

Project Location: D:/Microsemi_prj

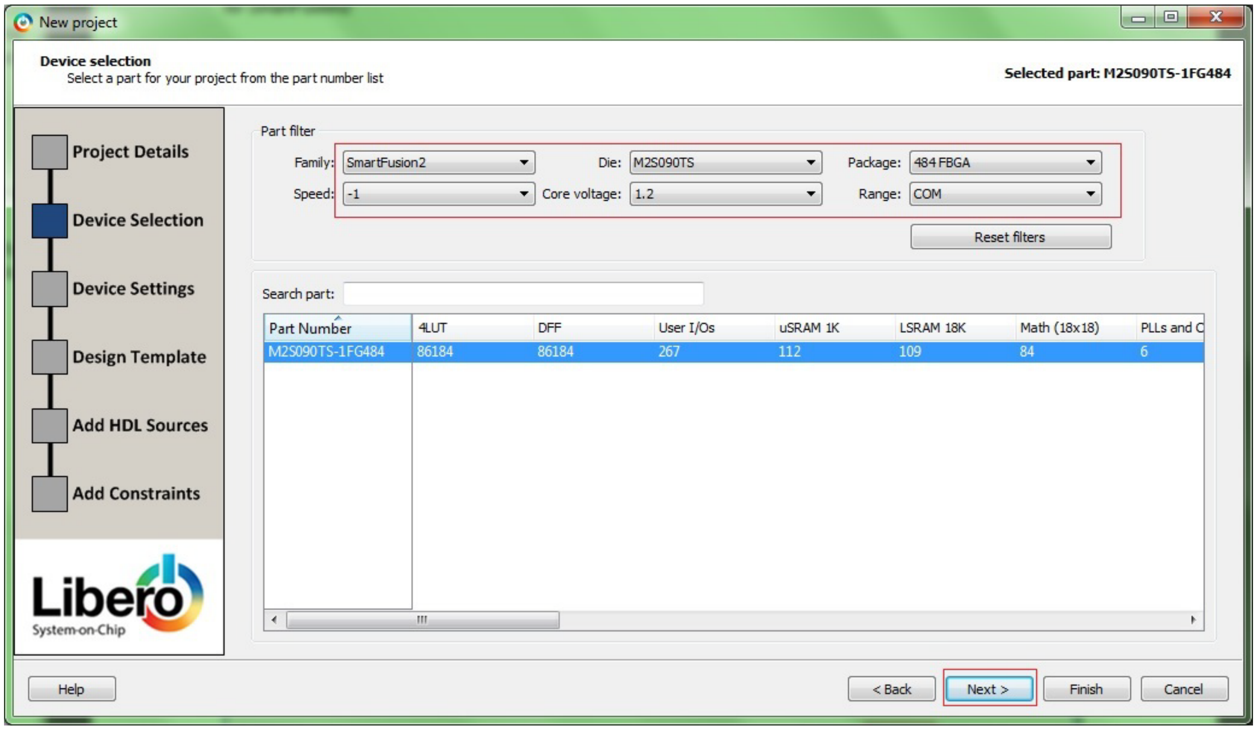
Description:

Preferred HDL Type: Verilog

☐ Enable Block Creation

4. Click **Next**. This opens **Device Selection** page as shown in Figure 4.
5. Select the following values from the drop down lists:
   - **Family**: SmartFusion2
   - **Die**: M2S090TS
   - **Package**: 484 FBGA
   - **Speed**: -1
   - **Core voltage**: 1.2
   - **Range**: COM

Figure 4 • **Device Selection Page**



Device selection
Select a part for your project from the part number list

Selected part: M2S090TS-1FG484

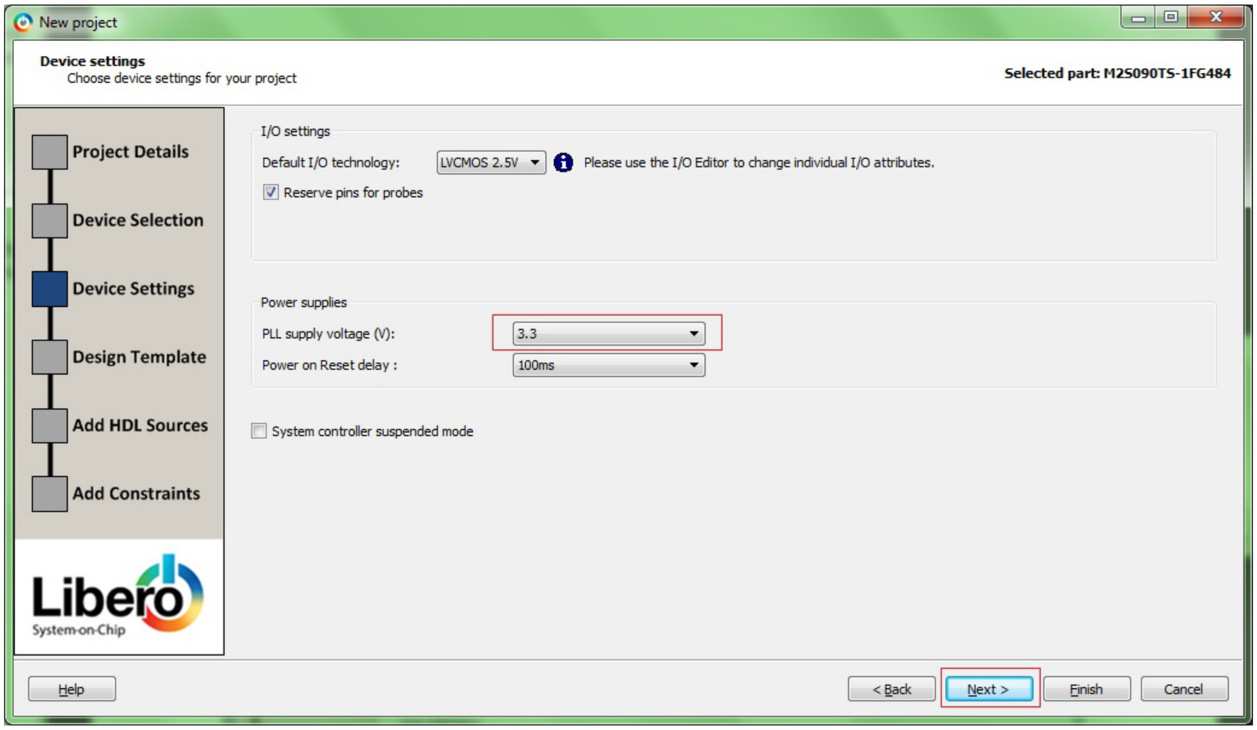
Part filter
Family: SmartFusion2 Die: M2S090TS Package: 484 FBGA
Speed: -1 Core voltage: 1.2 Range: COM

Search part:

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

6. Click **Next**. This opens **Device Settings** page as shown in Figure 5. Select PLL supply voltage as 3.3.

Figure 5 • **Device Settings**


New project

Device settings
Choose device settings for your project

Selected part: M2S090TS-1FG484

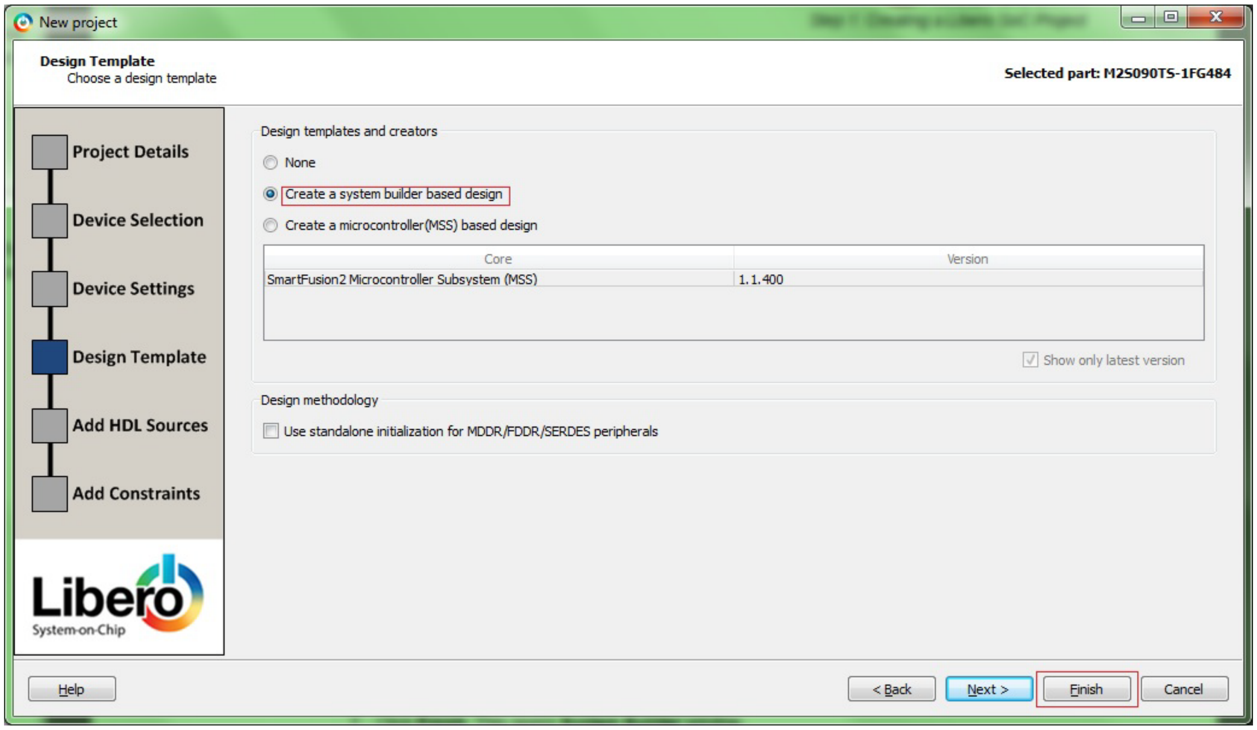
I/O settings
Default I/O technology: LVCMOS 2.5V
Please use the I/O Editor to change individual I/O attributes.
☑ Reserve pins for probes

Power supplies
PLL supply voltage (V): 3.3
Power on Reset delay : 100ms

☐ System controller suspended mode

7. Click **Next**. This opens **Design Template** page as shown in Figure 6, Under Design Templates and Creators, select **Create a system builder based design**.

Figure 6 • **Device Template Page**


New project

Design Template
Choose a design template

Selected part: M2S090TS-1FG484

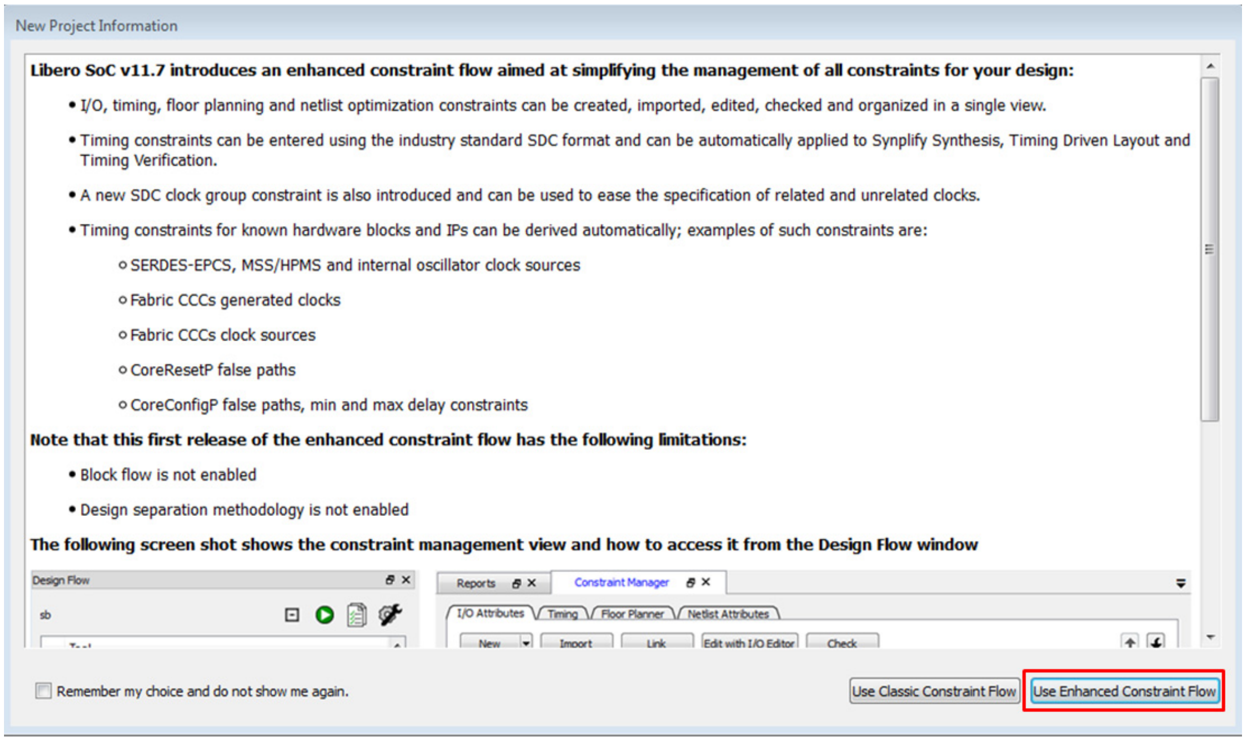
Design templates and creators
○ None
● Create a system builder based design
○ Create a microcontroller(MSS) based design

Core	Version
SmartFusion2 Microcontroller Subsystem (MSS)	1.1.400

☑ Show only latest version

Design methodology
☐ Use standalone initialization for MDDR/FDDR/SERDES peripherals

Figure 7 • **New Project Information Window**


New Project Information

Libero SoC v11.7 introduces an enhanced constraint flow aimed at simplifying the management of all constraints for your design:

- I/O, timing, floor planning and netlist optimization constraints can be created, imported, edited, checked and organized in a single view.
- Timing constraints can be entered using the industry standard SDC format and can be automatically applied to Synplify Synthesis, Timing Driven Layout and Timing Verification.
- A new SDC clock group constraint is also introduced and can be used to ease the specification of related and unrelated clocks.
- Timing constraints for known hardware blocks and IPs can be derived automatically; examples of such constraints are:
  - SERDES-EPCS, MSS/HPMS and internal oscillator clock sources
  - Fabric CCCs generated clocks
  - Fabric CCCs clock sources
  - CoreResetP false paths
  - CoreConfigP false paths, min and max delay constraints

Note that this first release of the enhanced constraint flow has the following limitations:

- Block flow is not enabled
- Design separation methodology is not enabled

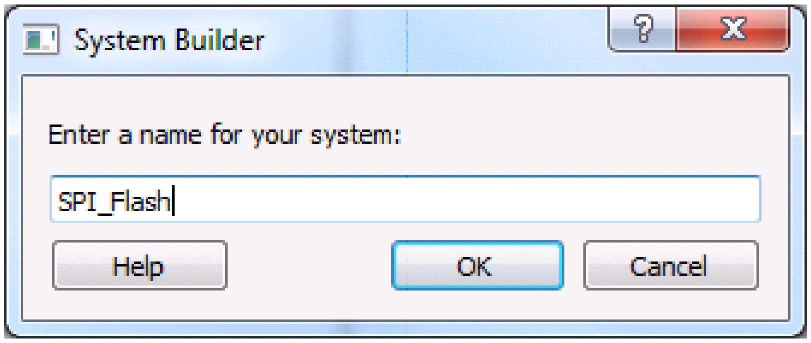
The following screen shot shows the constraint management view and how to access it from the Design Flow window

Remember my choice and do not show me again.

8. Click **Finish. New Project Information** window is displayed. Select **Use Enhanced Constraint Flow** option to use the new constraint flow as shown in Figure 7.
9. **System Builder** window is displayed. Enter the name of the system as **SPI_Flash** and click **OK**, as shown in Figure 8.

Note: System Builder is a graphical design wizard. It creates a design based on high-level design specifications by taking the user through a set of high-level questions that define the intended system.

Figure 8 • **System Builder Window**

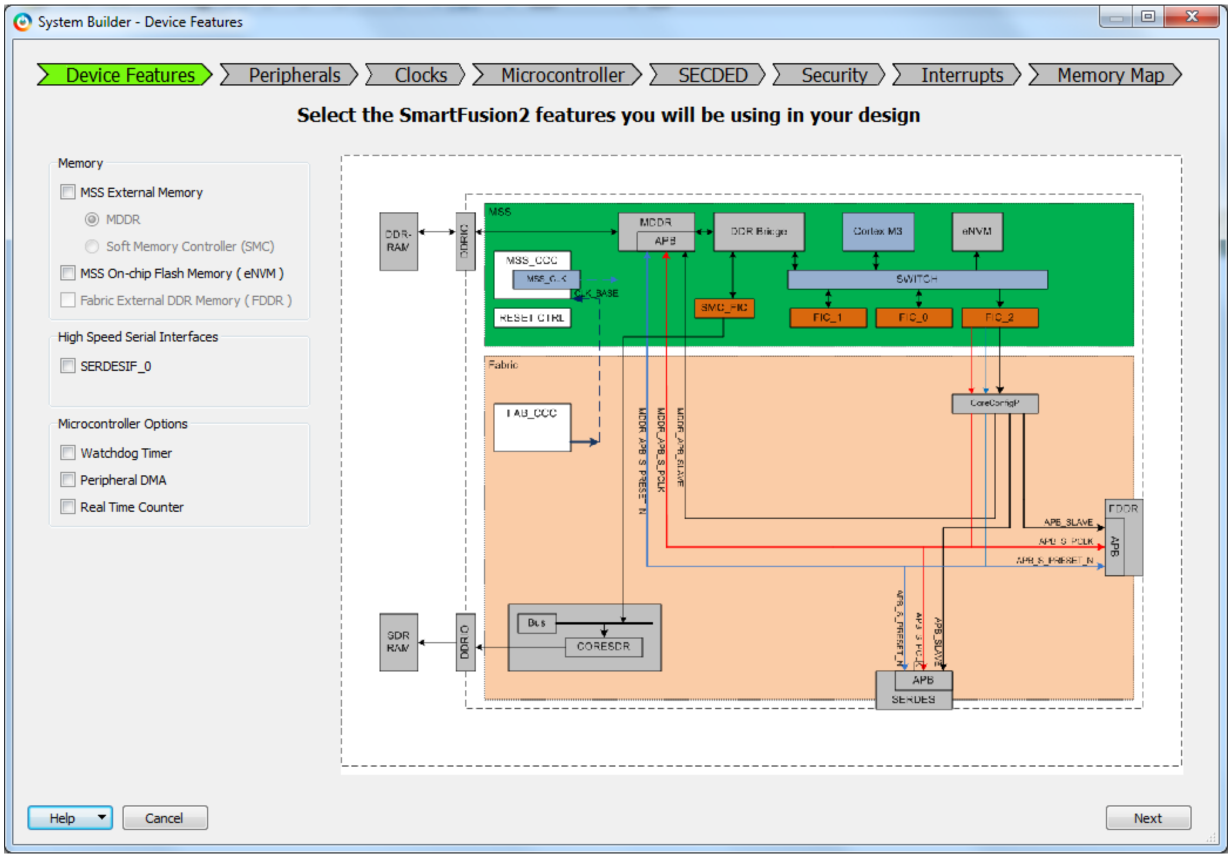


Enter a name for your system:

SPI_Flash

Figure 9 shows the **System Builder – Device Features** page.

***Figure 9* • System Builder – Device Features Page**



Select the SmartFusion2 features you will be using in your design

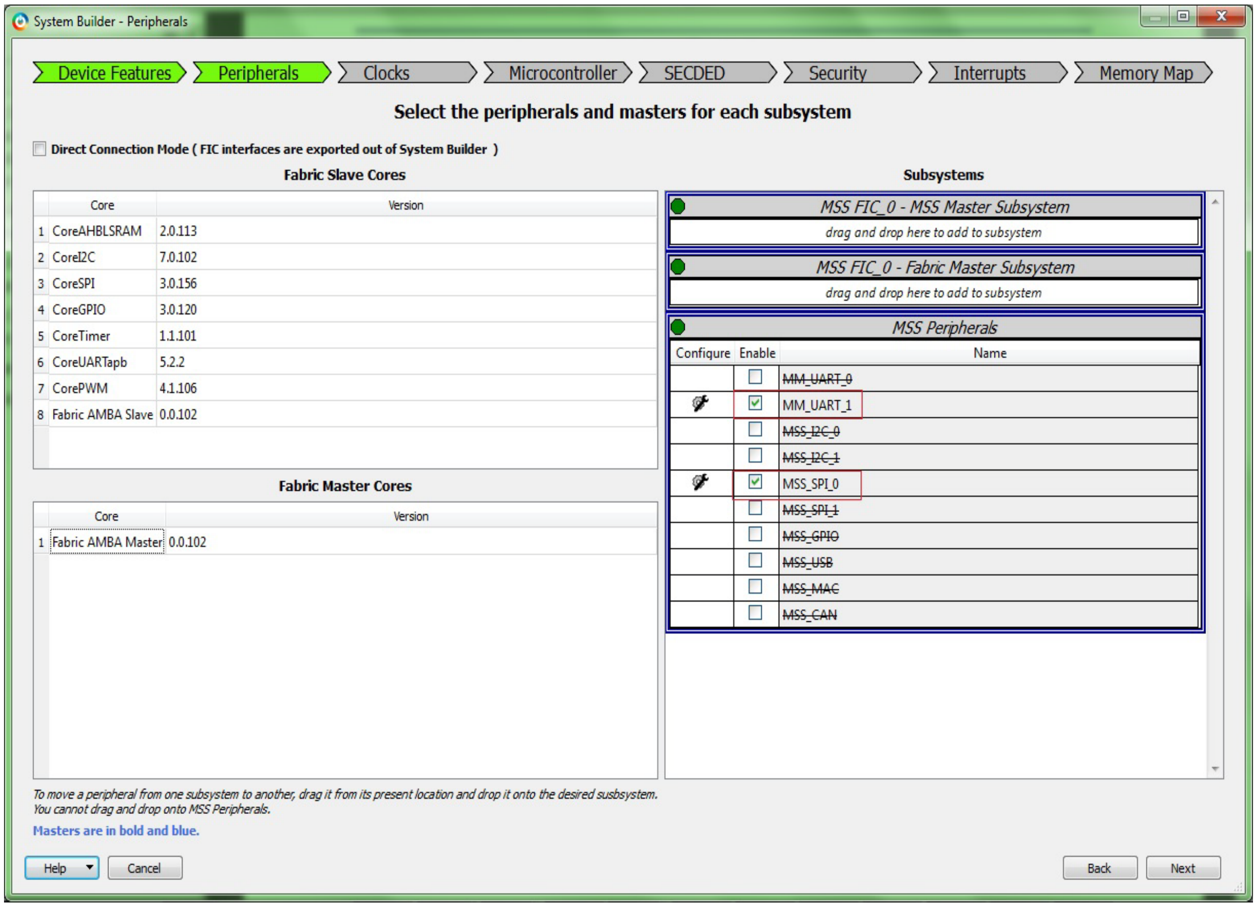
Memory
- [ ] MSS External Memory
  - ( ) MDDR
  - ( ) Soft Memory Controller (SMC)
- [ ] MSS On-chip Flash Memory (eNVM)
- [ ] Fabric External DDR Memory (FDDR)

High Speed Serial Interfaces
- [ ] SERDESIF_0

Microcontroller Options
- [ ] Watchdog Timer
- [ ] Peripheral DMA
- [ ] Real Time Counter

10. Click **Next**. This opens **System Builder - Peripherals** page as shown in Figure 10.
11. Under the MSS Peripherals section, clear all the check boxes except **MM_UART_1** and **MSS_SPI_0**, as shown in Figure 10.

Figure 10 • **System Builder – Peripherals Page**



Select the peripherals and masters for each subsystem

☐ Direct Connection Mode (FIC interfaces are exported out of System Builder)

Fabric Slave Cores

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

Fabric Master Cores

	Core	Version
1	Fabric AMBA Master	0.0.102

Subsystems

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

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

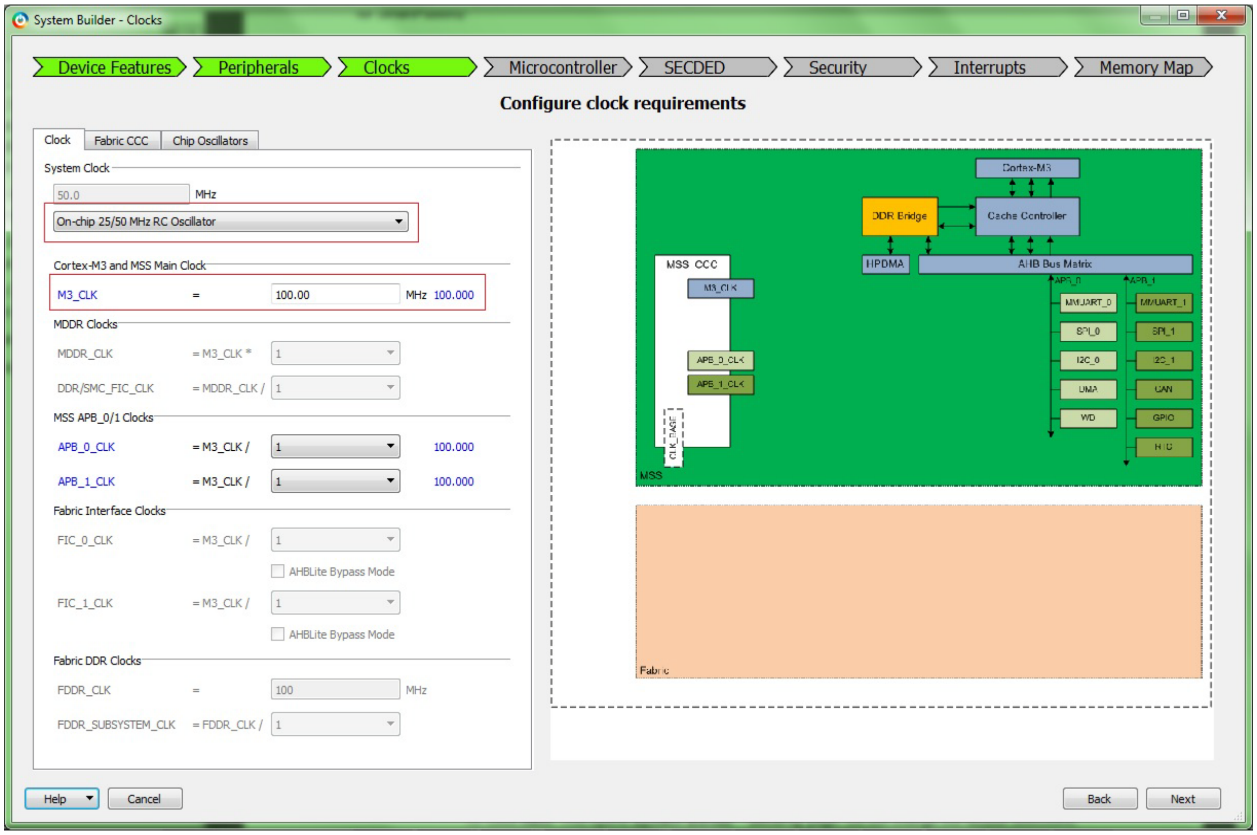
MSS Peripherals

Configure	Enable	Name
	☐	~~MM_UART_0~~
	✓	MM_UART_1
	☐	~~MSS_I2C_0~~
	☐	~~MSS_I2C_1~~
	✓	MSS_SPI_0
	☐	~~MSS_SPI_1~~
	☐	~~MSS_GPIO~~
	☐	~~MSS_USB~~
	☐	~~MSS_MAC~~
	☐	~~MSS_CAN~~

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

12. Click **Next**. This opens **System Builder - Clocks** page as shown in Figure 11.
13. In the **System Builder - Clocks** page (Refer to Figure 11):
    - Select **System Clock** frequency as **50 MHz** and clock source as **On-chip 25/50 MHz RC Oscillator**
    - Select **M3_CLK** as **100 MHz**
    - Select **APB_0_CLK** and **APB_1_CLK** frequency as **M3_CLK/1**
    - Do not change the default settings of remaining parameters.

Figure 11 • **System Builder – Clock Page**


System Builder - Clocks

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

Configure clock requirements

Clock | Fabric CCC | Chip Oscillators

System Clock

50.0 MHz

On-chip 25/50 MHz RC Oscillator

Cortex-M3 and MSS Main Clock

M3_CLK = 100.00 MHz 100.000

MDDR Clocks

MDDR_CLK = M3_CLK * 1

DDR/SMC_FIC_CLK = MDDR_CLK / 1

MSS APB_0/1 Clocks

APB_0_CLK = M3_CLK / 1 100.000

APB_1_CLK = M3_CLK / 1 100.000

Fabric Interface Clocks

FIC_0_CLK = M3_CLK / 1

AHBLite Bypass Mode

FIC_1_CLK = M3_CLK / 1

AHBLite Bypass Mode

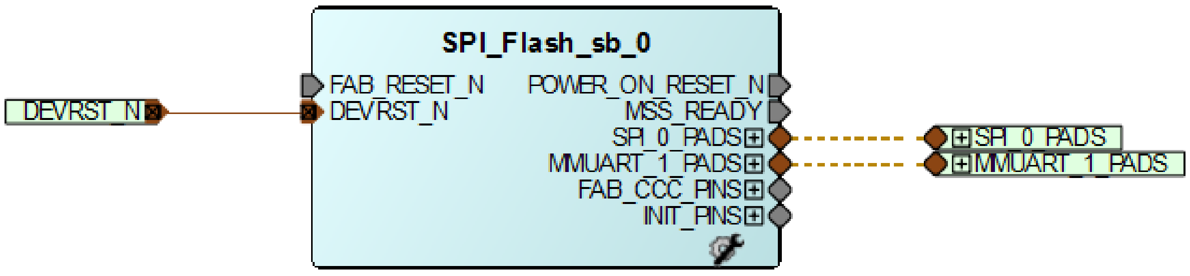
Fabric DDR Clocks

FDDR_CLK = 100 MHz

FDDR_SUBSYSTEM_CLK = FDDR_CLK / 1

14. Click **Next**. This opens **System Builder - Microcontroller** page. Do not change the default selections.
15. Click **Next**. This opens **System Builder - SECDED** page. Do not change the default selections.
16. Click **Next**. This opens **System Builder - Security** page. Do not change the default selections.
17. Click **Next**. This opens **System Builder - Interrupts** page. Do not change the default selections.
18. Click **Next**. This opens **System Builder - Memory Map** page. Do not change the default selections.
19. Click **Finish**.
20. Select **File > Save** to save **SPI_Flash**. Select the **SPI_Flash** tab on the Smart Design canvas, as shown in Figure 12.

Figure 12 • **SPI_Flash Smart Design**

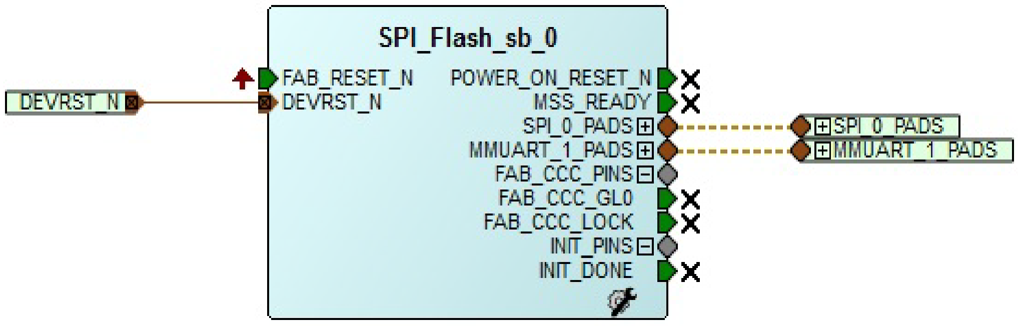


2.4.2 Connecting Components in SPI_Flash SmartDesign

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

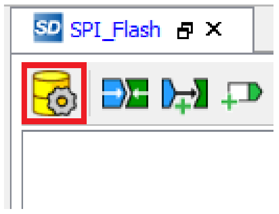
1. Right-click **POWER_ON_RESET_N** and select **Mark Unused**.
2. Right-click **MSS_READY** and select **Mark Unused**.
3. Expand **INIT_PINS**, right-click **INIT_DONE** and select **Mark Unused**.
4. Expand **FAB_CCC_PINS**, right-click **FAB_CCC_GL0** and select **Mark Unused**.
5. Right-click **FAB_CCC_LOCK** and select **Mark Unused**.
6. Right-click **FAB_RESET_N** and select **Tie High**.
7. Click **File > Save**.
   The SPI_Flash design is displayed, as shown in Figure 13.

Figure 13 • **SPI_Flash Smart Design**



8. Generate the SPI_Flash Smart Design by clicking **SmartDesign > Generate Component** or by clicking **Generate Component** on the SmartDesign toolbar, as shown in Figure 14.

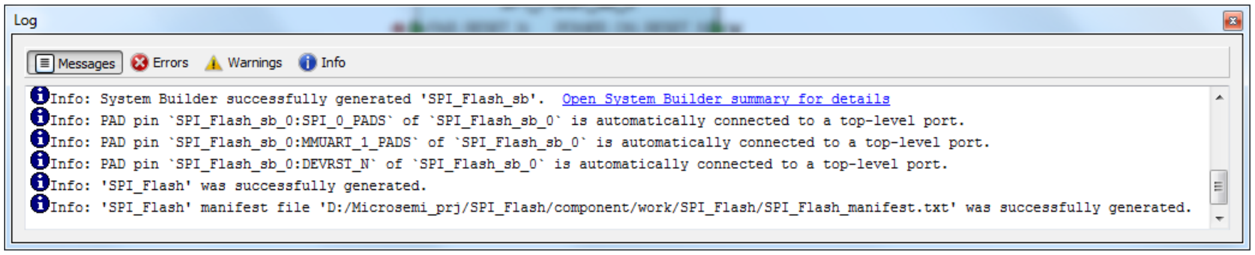
Figure 14 • **Generate Component**



After successful generation of all the components, the following message is displayed on the log window, as shown in Figure 15.

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

Figure 15 • **Log Window**



```
Info: System Builder successfully generated 'SPI_Flash_sb'.  Open System Builder summary for details
Info: PAD pin `SPI_Flash_sb_0:SPI_0_PADS` of `SPI_Flash_sb_0` is automatically connected to a top-level port.
Info: PAD pin `SPI_Flash_sb_0:MMUART_1_PADS` of `SPI_Flash_sb_0` is automatically connected to a top-level port.
Info: PAD pin `SPI_Flash_sb_0:DEVRST_N` of `SPI_Flash_sb_0` is automatically connected to a top-level port.
Info: 'SPI_Flash' was successfully generated.
Info: 'SPI_Flash' manifest file 'D:/Microsemi_prj/SPI_Flash/component/work/SPI_Flash/SPI_Flash_manifest.txt' was successfully generated.
```

2.5 Step 2: Generating the Program File

The following step describes how to generate the program file:

1. Double-click **Manage Constraints** and click **Derive Constraints** option under **Timing** tab to generate SDC file for root module.
2. Click **Yes** to associate SDC file to Synthesis, Place and Route and Timing Verification stages, as shown in Figure 16, Figure 17 on page 20, and Figure 18 on page 20.

Figure 16 • **Manage Constraints**

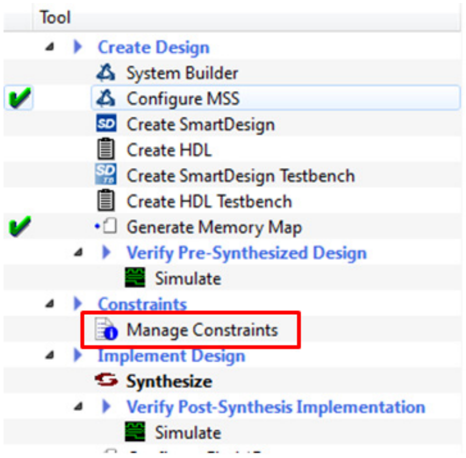


Figure 17 • **Timing - Derive Constraints Tab**

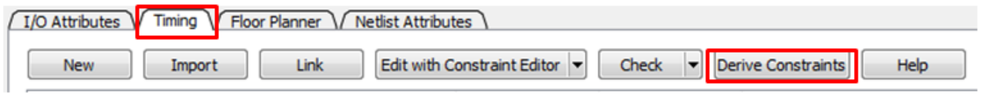
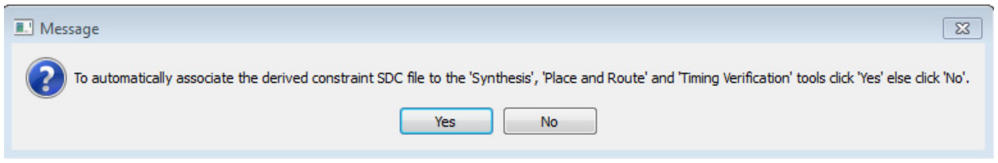


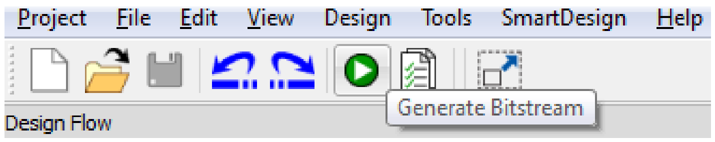
Figure 18 • **Associate the Derive Constraint SDC file - Message**



To automatically associate the derived constraint SDC file to the 'Synthesis', 'Place and Route' and 'Timing Verification' tools click 'Yes' else click 'No'.

3. Click **Generate Bitstream** as shown in Figure 19 to complete place and route, and generate the programming file.

Figure 19 • **Generate Bitstream**



2.6 Step 3: Programming SmartFusion2 Security Evaluation Board Using FlashPro

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

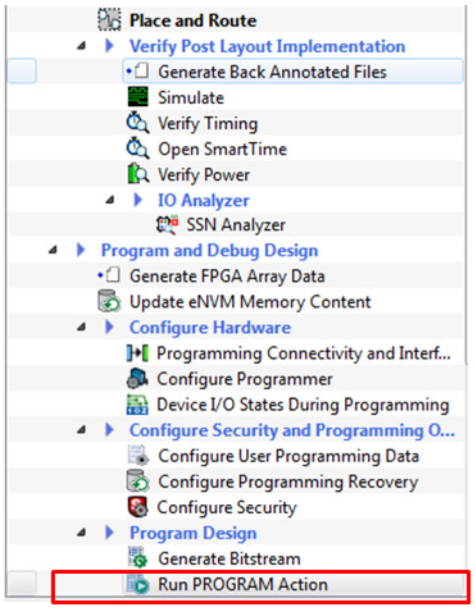
1. Connect the FlashPro4 programmer to the **J5** connector of the SmartFusion2 Security Evaluation Kit.
2. Connect the jumpers on the SmartFusion2 Security Evaluation Kit board as per Table 2.
   For more information on jumper locations, refer to Appendix: SmartFusion2 Security Evaluation Kit Board Jumper Locations.
   CAUTION: Ensure that the power supply switch, **SW7** is switched OFF while connecting the jumpers on the SmartFusion 2 Security Evaluation Kit.

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

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

3. Connect the power supply to the **J6** connector.
4. Switch ON the power supply switch, **SW7**. Refer to Appendix: Board Setup for Programming the Tutorial for information on the board setup for running the tutorial.
5. To program the SmartFusion2 device, double-click **Run PROGRAM Action** in the **Design Flow** tab as shown in Figure 20.

Figure 20 • **Run Program Action**



Place and Route
Verify Post Layout Implementation
Generate Back Annotated Files
Simulate
Verify Timing
Open SmartTime
Verify Power
IO Analyzer
SSN Analyzer
Program and Debug Design
Generate FPGA Array Data
Update eNVM Memory Content
Configure Hardware
Programming Connectivity and Interf...
Configure Programmer
Device I/O States During Programming
Configure Security and Programming O...
Configure User Programming Data
Configure Programming Recovery
Configure Security
Program Design
Generate Bitstream
Run PROGRAM Action

2.7 Step 4: Configuring and Generating Firmware

The Design Firmware window displays compatible firmware drivers based on the peripherals configured in the design. Following drivers are used in this tutorial:

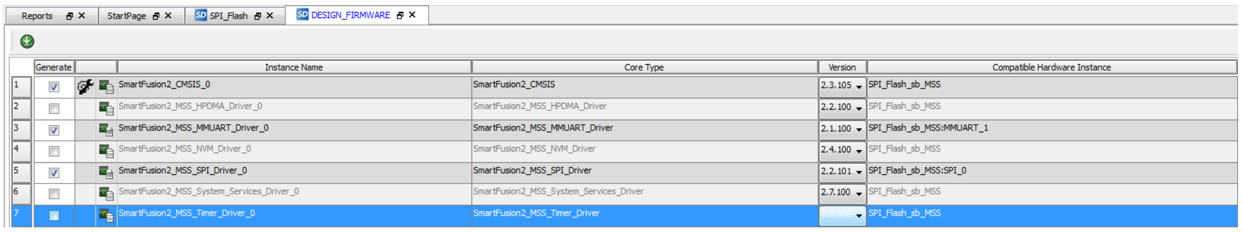
- CMSIS
- MMUART
- SPI

To generate the required drivers:

1. Double-click **Configure Firmware Cores** in **Handoff design for Firmware Development** in **Design Flow** window.
2. Clear all the drivers check boxes, except **CMSIS**, **MMUART**, and **SPI**, as shown in Figure 21.

Note: Select the latest version of the drivers.

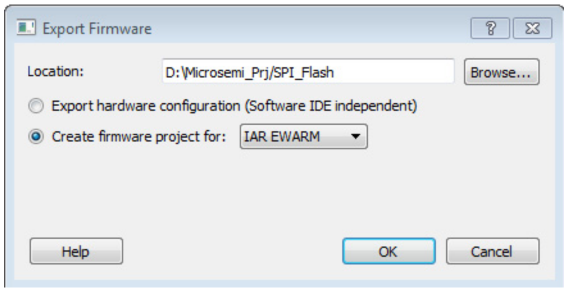
Figure 21 • **Configuring Firmware**



	Generate	Instance Name	Core Type	Version	Compatible Hardware Instance
1	✓	SmartFusion2_CMSIS_0	SmartFusion2_CMSIS	2.3.105	SPI_Flash_sb_MSS
2		SmartFusion2_MSS_HPDMA_Driver_0	SmartFusion2_MSS_HPDMA_Driver	2.2.100	SPI_Flash_sb_MSS
3	✓	SmartFusion2_MSS_MMUART_Driver_0	SmartFusion2_MSS_MMUART_Driver	2.1.100	SPI_Flash_sb_MSS:MMUART_1
4		SmartFusion2_MSS_NVM_Driver_0	SmartFusion2_MSS_NVM_Driver	2.4.100	SPI_Flash_sb_MSS
5	✓	SmartFusion2_MSS_SPI_Driver_0	SmartFusion2_MSS_SPI_Driver	2.2.101	SPI_Flash_sb_MSS:SPI_0
6		SmartFusion2_MSS_System_Services_Driver_0	SmartFusion2_MSS_System_Services_Driver	2.7.100	SPI_Flash_sb_MSS
7		SmartFusion2_MSS_Timer_Driver_0	SmartFusion2_MSS_Timer_Driver	2.2.100	SPI_Flash_sb_MSS

3. Double-click **Export Firmware** in **Handoff design for Firmware Development** in **Design Flow** window.
   The Export Firmware dialog box is displayed, as shown in Figure 22.

Figure 22 • **Export Firmware Dialog Box**



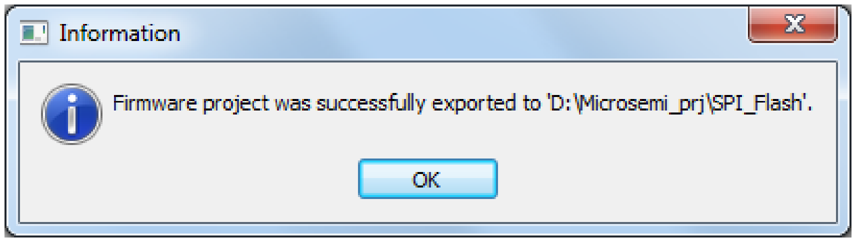
Location: D:\Microsemi_Prj/SPI_Flash

( ) Export hardware configuration (Software IDE independent)

(●) Create firmware project for: IAR EWARM

4. In the **Export Firmware** dialog box:
   - Select **Create firmware project for**.
   - Select **IAR EWARM** from the drop down list.
5. Click **OK**. The successful firmware generation window is displayed as shown in Figure 23.

Figure 23 • **Firmware Successfully Exported Message**



Firmware project was successfully exported to 'D:\Microsemi_prj\SPI_Flash'.

6. Click **OK**.

The SmartFusion2 Security Evaluation Kit is ready for running and debugging the IAR Embedded Workbench application through J-Link Debugger.

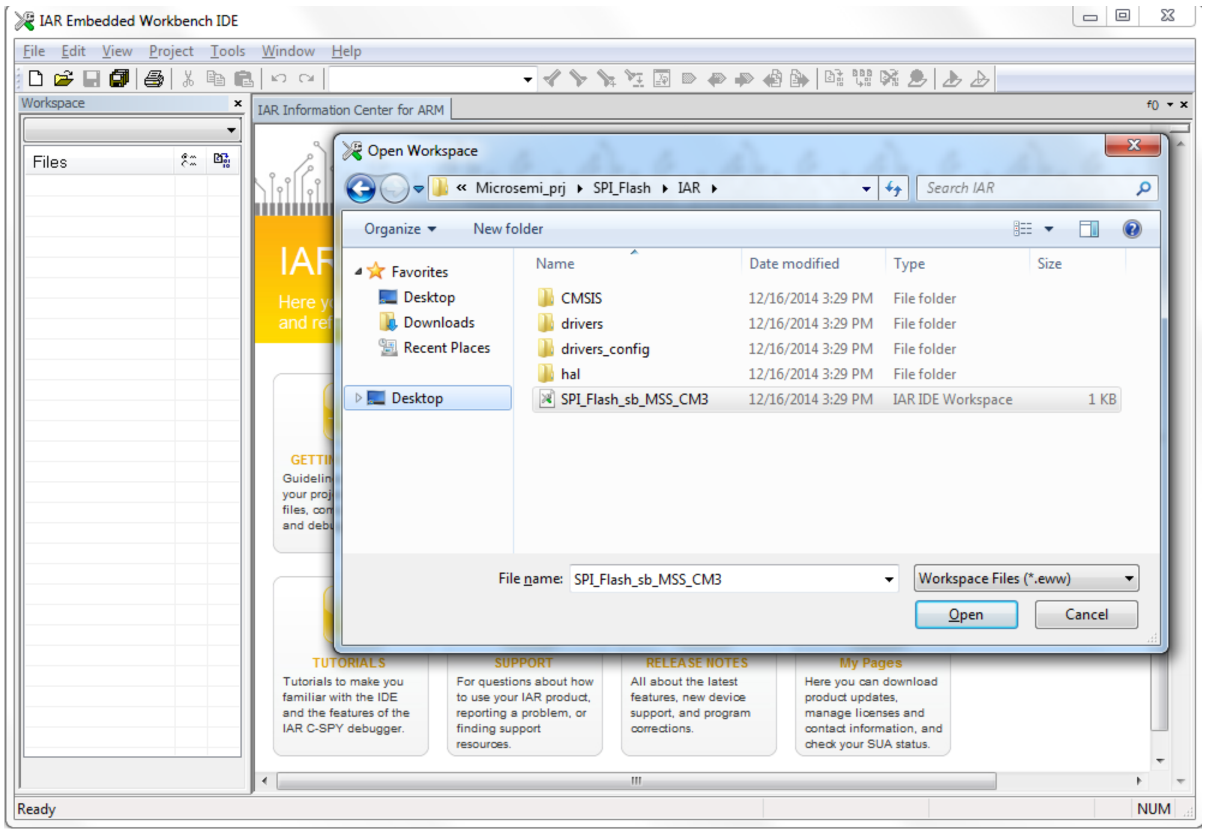
2.8 Step 5: Building Software Application using IAR Embedded Workbench

The following steps describe how to build a software application using IAR embedded workbench:

1. Connect the J-Link programmer to **J4 connector** of SmartFusion2 Security Evaluation Kit. Refer to "Appendix: SmartFusion2 Security Evaluation Kit Board Jumper Locations" on page 48 for information on the board setup for running and debugging the IAR software application.

   Ensure that the SmartFusion2 Security Evaluation Kit Jumper **J8** is in **2-3 closed** position for IAR Embedded Workbench and J-Link communication.
2. Open the IAR project by double-clicking **SPI_Flash_sb_MSS_CM3 IAR** project as shown in Figure 24.

Figure 24 • **Invoking IAR Embedded Workbench from Libero SoC Software**


IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
Workspace
Files
IAR Information Center for ARM
Open Workspace
« Microsemi_prj ▸ SPI_Flash ▸ IAR ▸
Search IAR
Organize ▾ New folder
Favorites
Desktop
Downloads
Recent Places
Desktop
Name
Date modified
Type
Size

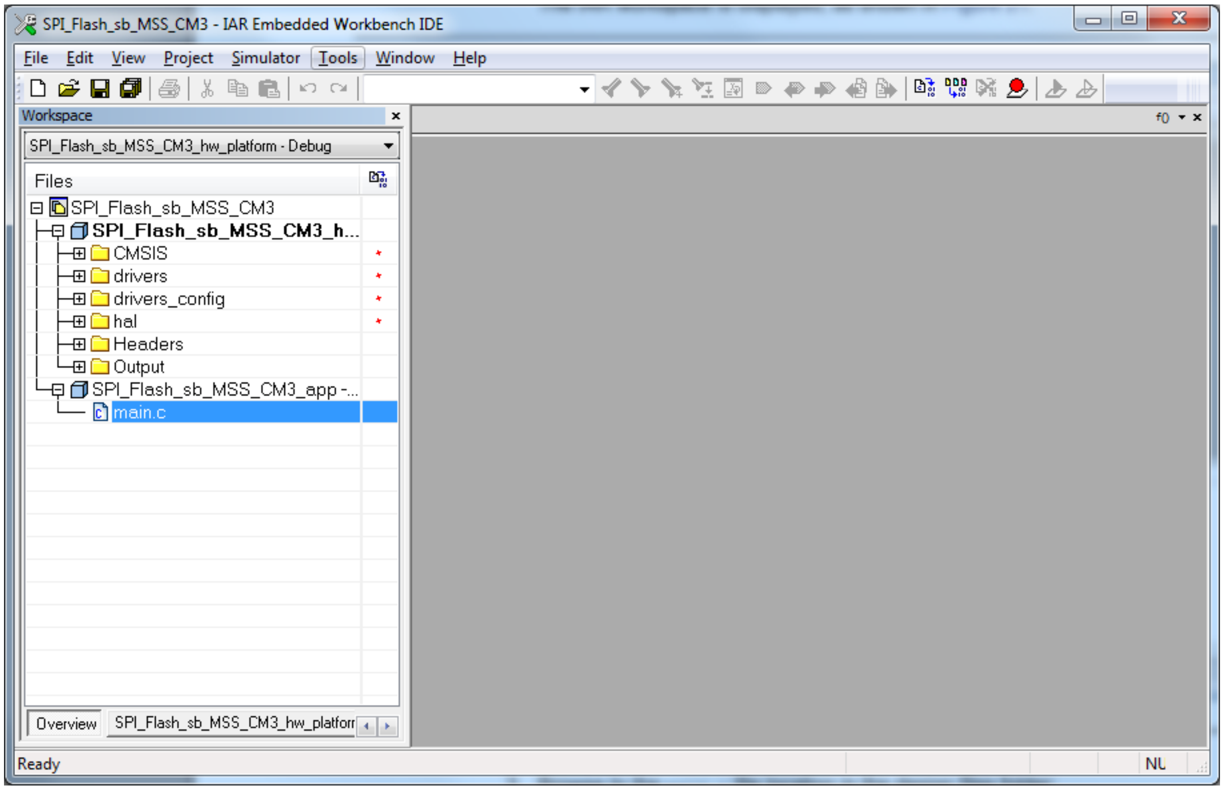
Name	Date modified	Type	Size
CMSIS	12/16/2014 3:29 PM	File folder	
drivers	12/16/2014 3:29 PM	File folder	
drivers_config	12/16/2014 3:29 PM	File folder	
hal	12/16/2014 3:29 PM	File folder	
SPI_Flash_sb_MSS_CM3	12/16/2014 3:29 PM	IAR IDE Workspace	1 KB

File name: SPI_Flash_sb_MSS_CM3
Workspace Files (*.eww)

Open
Cancel
TUTORIALS
Tutorials to make you familiar with the IDE and the features of the IAR C-SPY debugger.
SUPPORT
For questions about how to use your IAR product, reporting a problem, or finding support resources.
RELEASE NOTES
All about the latest features, new device support, and program corrections.
My Pages
Here you can download product updates, manage licenses and contact information, and check your SUA status.
Ready
NUM

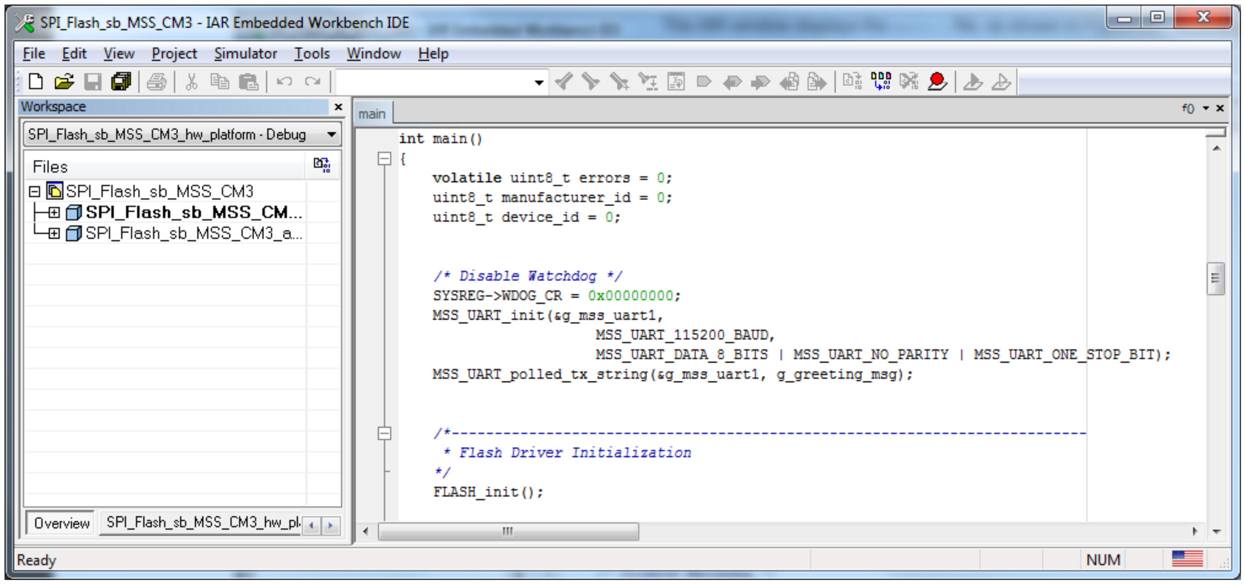
The IAR workspace is displayed, as shown in Figure 25.

Figure 25 • **IAR Workspace**


SPI_Flash_sb_MSS_CM3 - IAR Embedded Workbench IDE
File Edit View Project Simulator Tools Window Help
Workspace
SPI_Flash_sb_MSS_CM3_hw_platform - Debug
Files
SPI_Flash_sb_MSS_CM3
SPI_Flash_sb_MSS_CM3_h...
CMSIS
drivers
drivers_config
hal
Headers
Output
SPI_Flash_sb_MSS_CM3_app -...
main.c
Overview
SPI_Flash_sb_MSS_CM3_hw_platforr
Ready
NL

3. Browse to the `main.c` file location in the design files folder: *<download folder>\SF2_SPI_Flash_IAR_Tutorial_DF\SourceFiles*.
4. Copy the `main.c` file and replace the existing `main.c` file under **SPI_Flash_sb_MSS_CM3_app** project in the IAR workspace.
5. The IAR window displays the `main.c` file, as shown in Figure 26.

Figure 26 • **IAR Workspace main.c file**


SPI_Flash_sb_MSS_CM3 - IAR Embedded Workbench IDE
File Edit View Project Simulator Tools Window Help
Workspace
SPI_Flash_sb_MSS_CM3_hw_platform - Debug
Files
SPI_Flash_sb_MSS_CM3
SPI_Flash_sb_MSS_CM...
SPI_Flash_sb_MSS_CM3_a...
Overview
SPI_Flash_sb_MSS_CM3_hw_pl
main
Ready
NUM

```
int main()
{
    volatile uint8_t errors = 0;
    uint8_t manufacturer_id = 0;
    uint8_t device_id = 0;


    /* Disable Watchdog */
    SYSREG->WDOG_CR = 0x00000000;
    MSS_UART_init(&g_mss_uart1,
                  MSS_UART_115200_BAUD,
                  MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT);
    MSS_UART_polled_tx_string(&g_mss_uart1, g_greeting_msg);


    /*---------------------------------------------------------------------------
     * Flash Driver Initialization
    */
    FLASH_init();
```

6. The winbondflash SPI flash drivers are not included in the Libero generated IAR workspace. To include the drivers in the IAR workspace, browse to the location of the winbondflash drivers in the design files folder: *<download_folder>\SF2_SPI_Flash_IAR_Tutorial_DF\SPI_Flash_Drivers*.
7. Copy the **winbondflash** folder to the drivers folder of SPI_Flash_sb_MSS_CM3_hw_platform project in the IAR workspace: *projectdirectory\IAR\drivers*.
8. Right-click and add the driver files (winbondflash.c & winbondflash.h) to the SPI_Flash_sb_MSS_CM3_hw_platform project in the IAR workspace, as shown in Figure 27.

Figure 27 • **IAR Workspace Window - Add winbondflash SPI Driver Files**

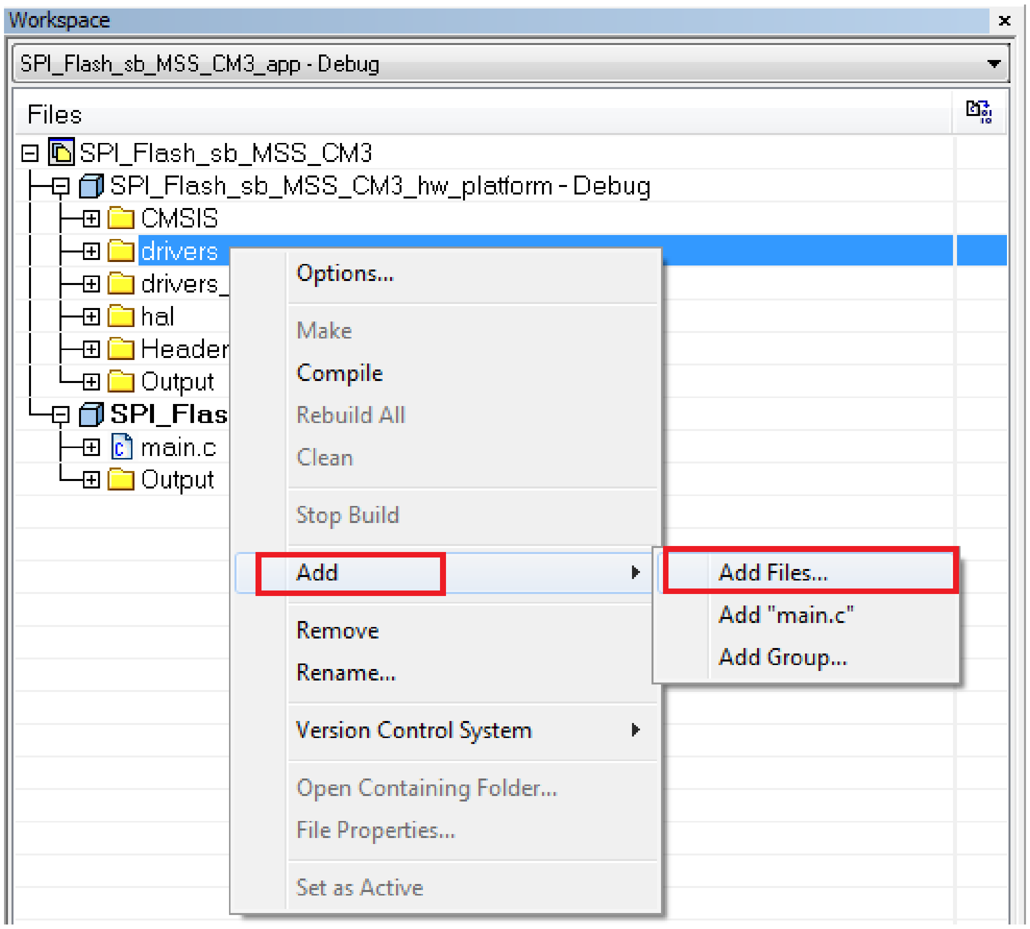
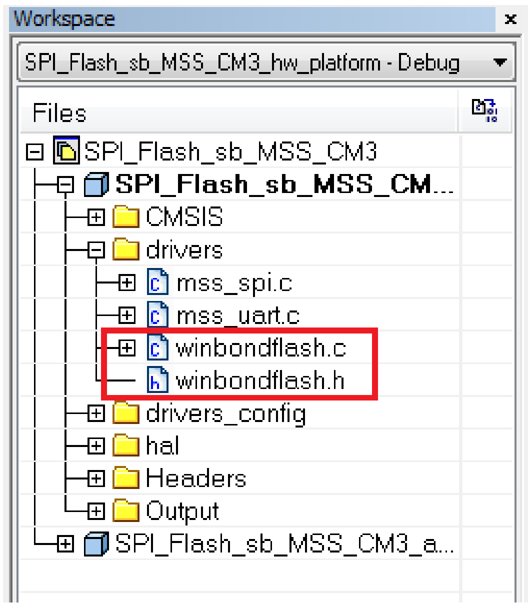


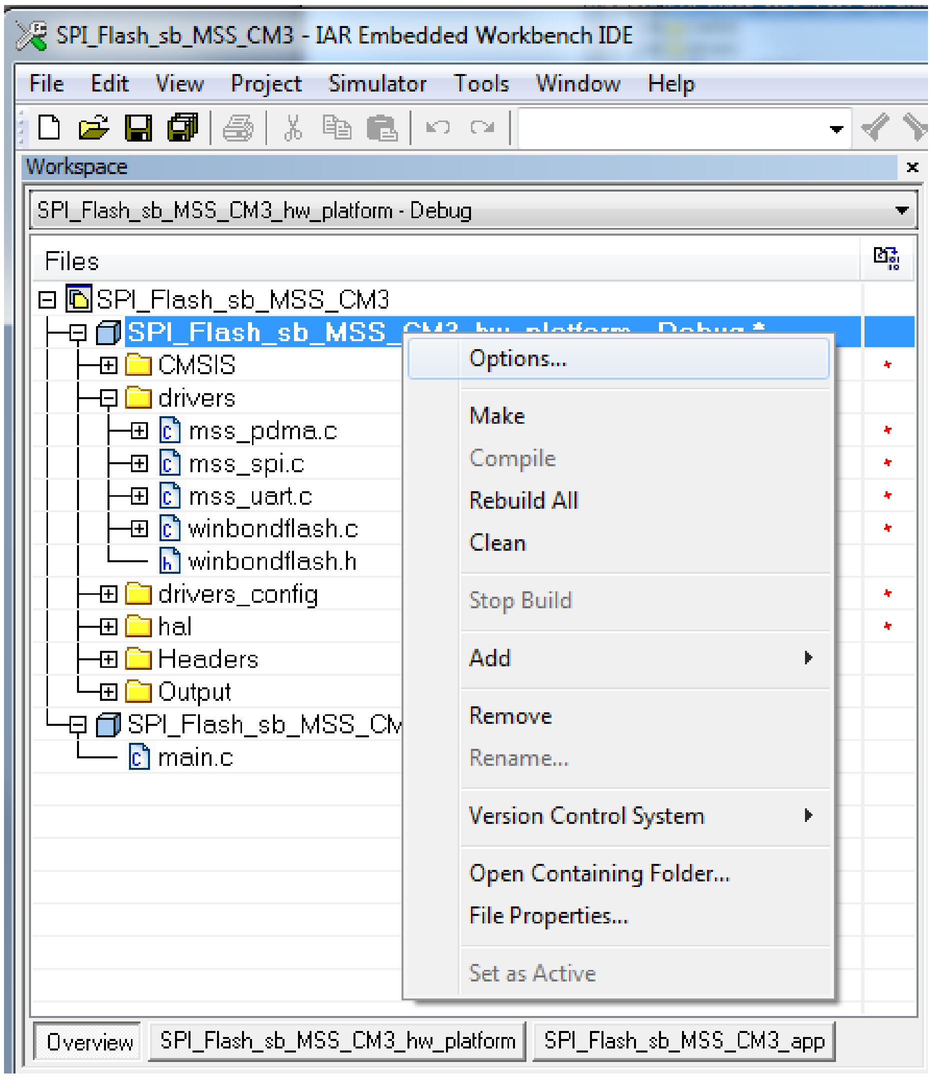
Figure 28 shows the IAR workspace window displaying winbondflash SPI Driver Files.

Figure 28 • **IAR Workspace Window - Display winbondflash SPI Driver Files**



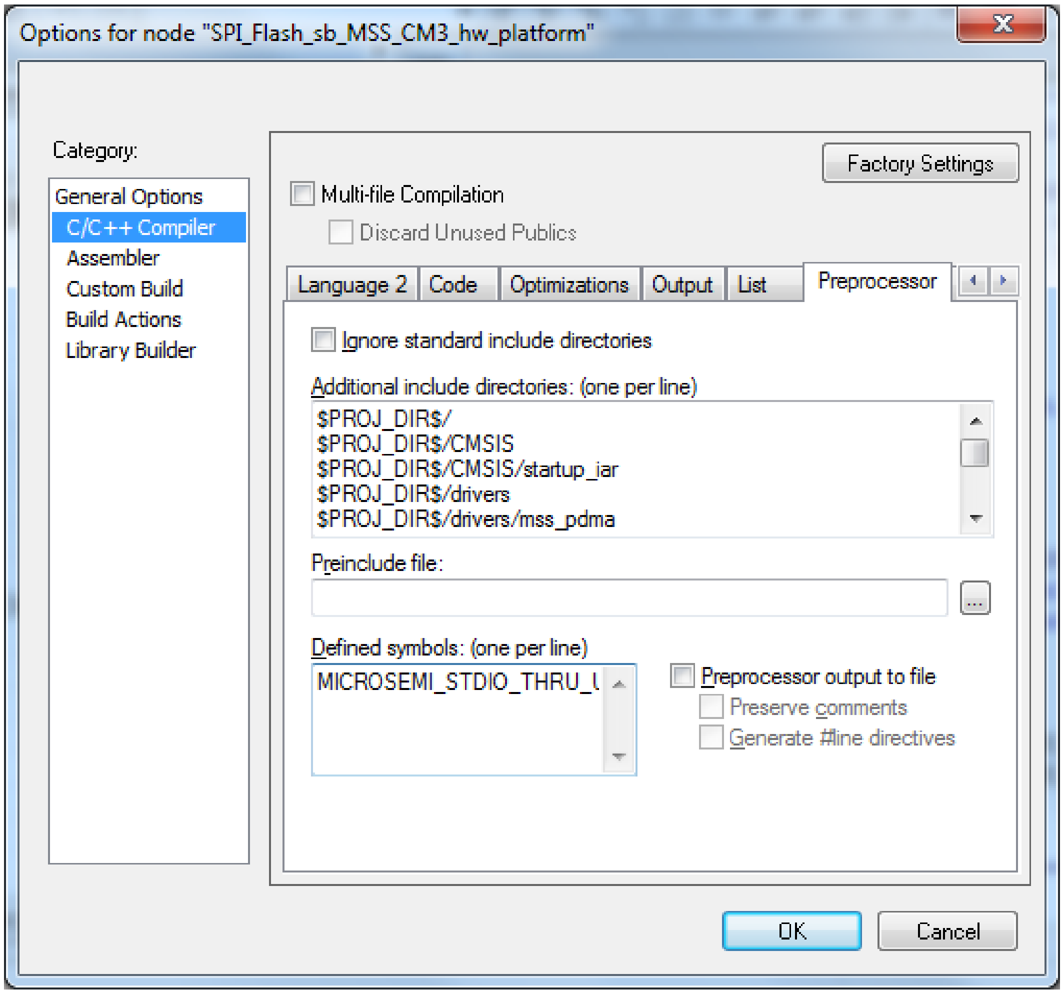
9. To configure the project, right-click the project name (SPI_Flash_sb_MSS_CM3_hw_platform) and click **Options**, as shown in Figure 29.

Figure 29 • **IAR Workspace Window - Choose Options**



10. This tutorial uses `printf` statements to display memory read data. Redirection of the output of `printf()` to a UART is enabled by adding the **MICROSEMI_STDIO_THRU_UART** symbol.
11. In **Options** window, click **C/C ++ Compiler**.
12. Click **Preprocessor** tab.
13. Under **Defined symbols** enter **MICROSEMI_STDIO_THRU_UART** and click **OK**, as shown in Figure 30.

Figure 30 • **IAR Workspace Window - Adding Symbol**

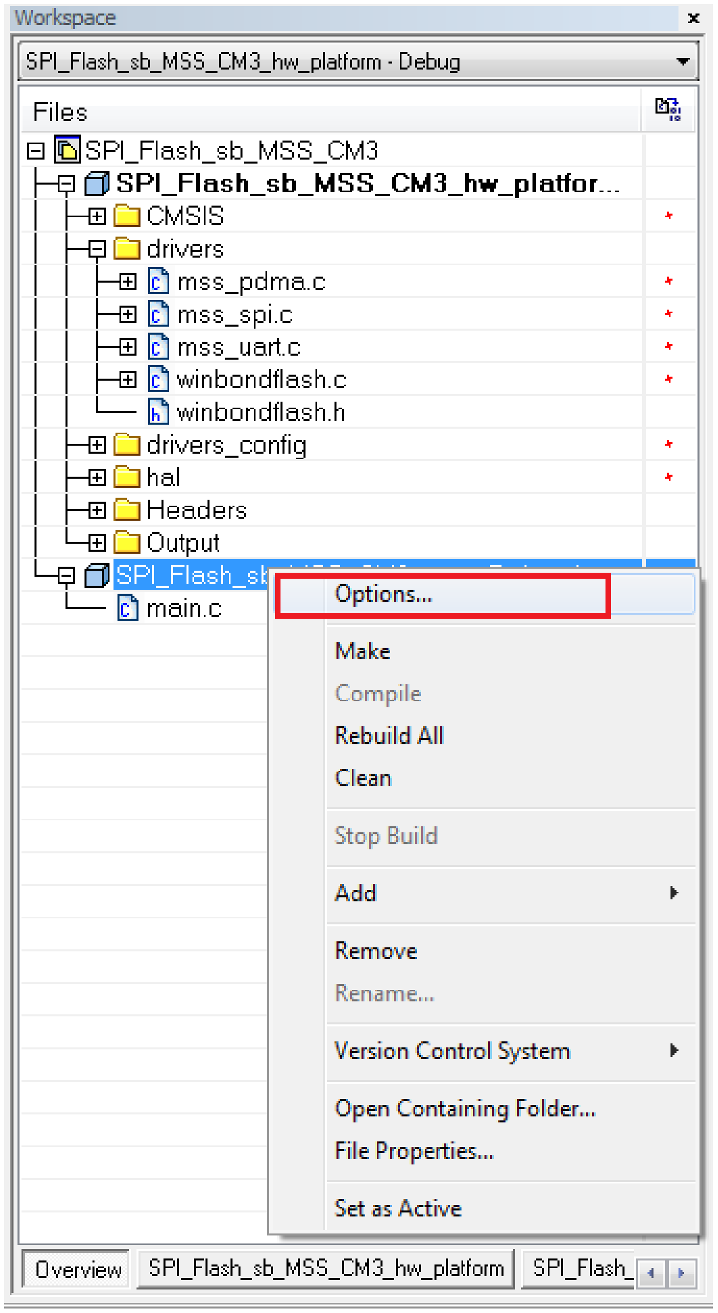


```
$PROJ_DIR$/
$PROJ_DIR$/CMSIS
$PROJ_DIR$/CMSIS/startup_iar
$PROJ_DIR$/drivers
$PROJ_DIR$/drivers/mss_pdma
```

Defined symbols: MICROSEMI_STDIO_THRU_U

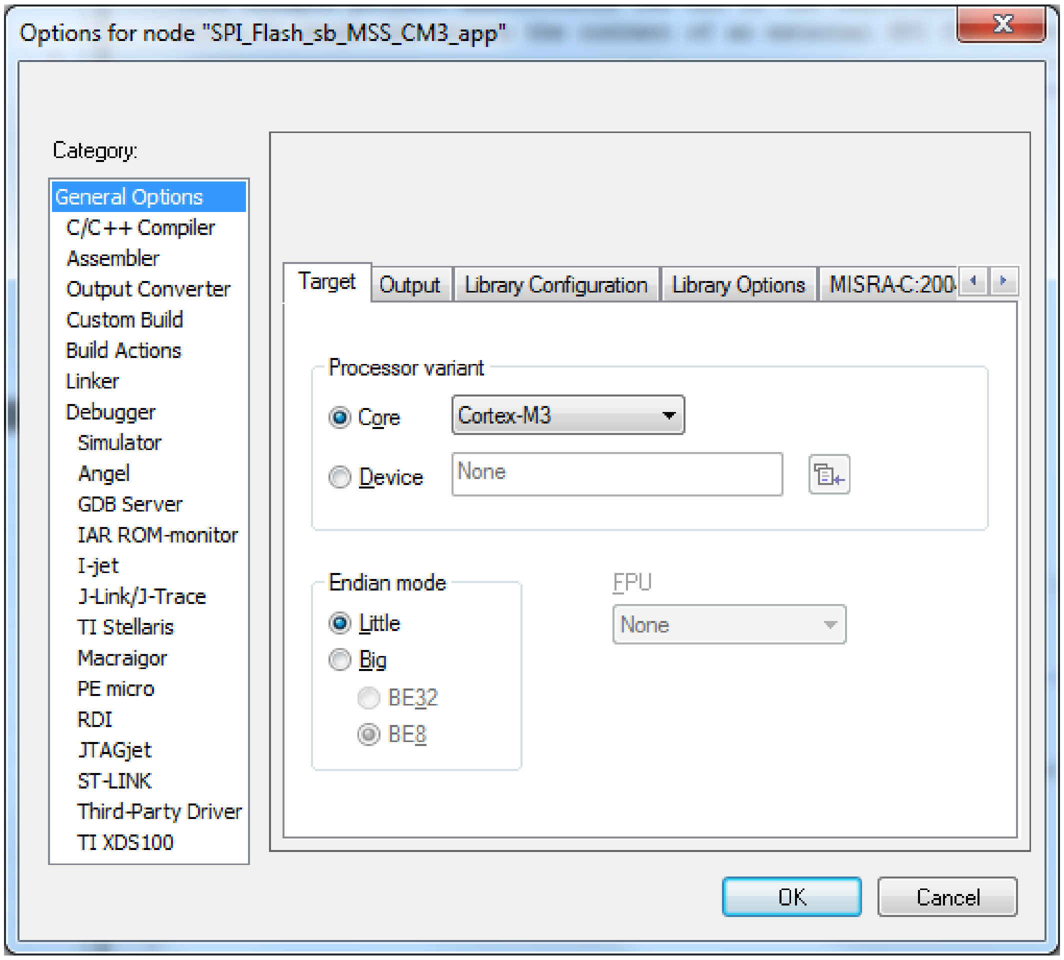
14. To configure the project, right-click the project name (SPI_Flash_sb_MSS_CM3_app), and click **Options**, as shown in Figure 31.

Figure 31 • **IAR Workspace Window - Choose Options**



The **Options for node SPI_Flash_sb_MSS_CM3_app** window is displayed, as shown in Figure 32.

Figure 32 • **IAR Node Options**


Options for node "SPI_Flash_sb_MSS_CM3_app"

Category:
- General Options
- C/C++ Compiler
- Assembler
- Output Converter
- Custom Build
- Build Actions
- Linker
- Debugger
  - Simulator
  - Angel
  - GDB Server
  - IAR ROM-monitor
  - I-jet
  - J-Link/J-Trace
  - TI Stellaris
  - Macraigor
  - PE micro
  - RDI
  - JTAGjet
  - ST-LINK
  - Third-Party Driver
  - TI XDS100

Target | Output | Library Configuration | Library Options | MISRA-C:200

Processor variant
- [✓ Core] Cortex-M3
- Device None

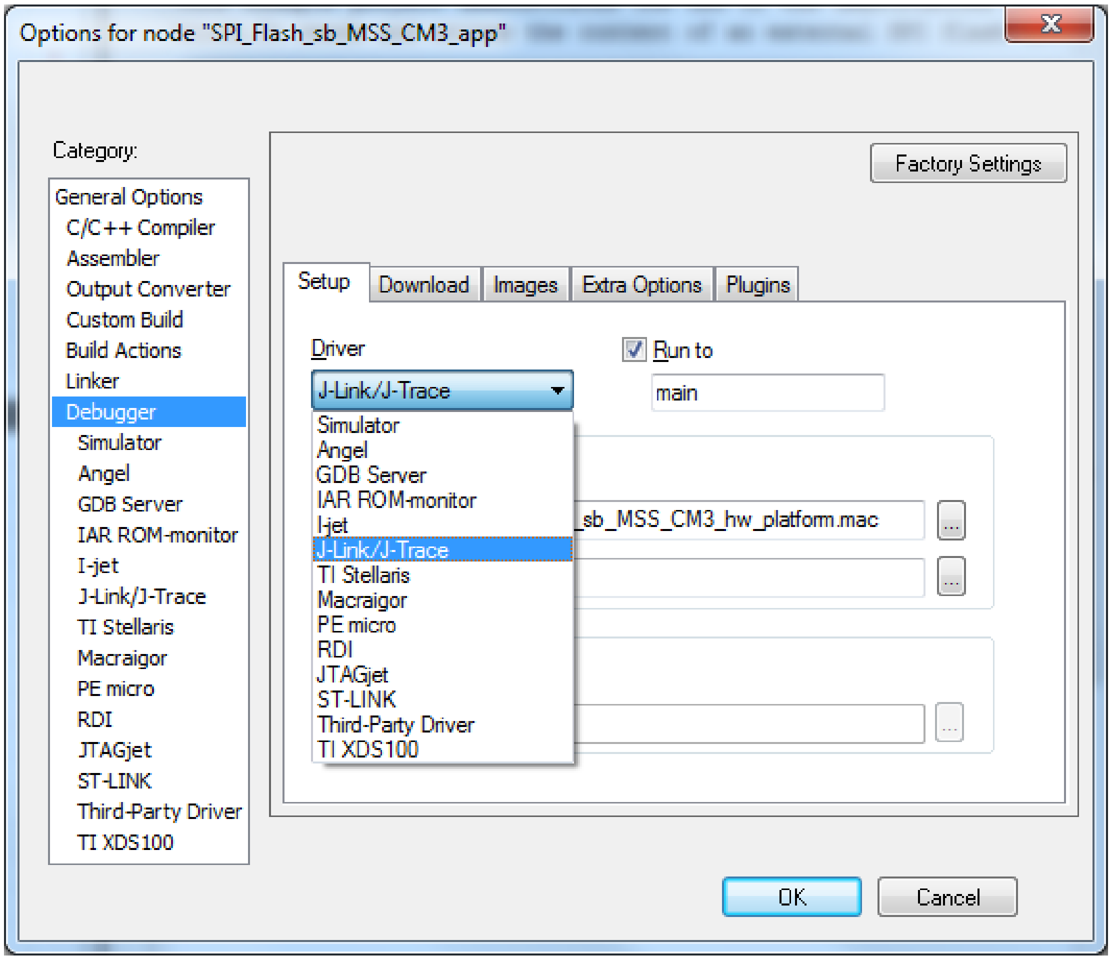
Endian mode
- [✓ Little]
- Big
  - BE32
  - BE8

FPU
None

OK | Cancel

15. Click **Debugger**. Under the **Setup** tab, select **J-Link/J-Trace** from the driver the drop-down list, refer to Figure 33.

Figure 33 • **IAR Debugger Options - Selecting Driver**



Options for node "SPI_Flash_sb_MSS_CM3_app"

Category:

- General Options
  - C/C++ Compiler
  - Assembler
  - Output Converter
  - Custom Build
  - Build Actions
  - Linker
  - Debugger
    - Simulator
    - Angel
    - GDB Server
    - IAR ROM-monitor
    - I-jet
    - J-Link/J-Trace
    - TI Stellaris
    - Macraigor
    - PE micro
    - RDI
    - JTAGjet
    - ST-LINK
    - Third-Party Driver
    - TI XDS100

Factory Settings

Setup | Download | Images | Extra Options | Plugins

Driver: J-Link/J-Trace

Simulator
Angel
GDB Server
IAR ROM-monitor
I-jet
J-Link/J-Trace
TI Stellaris
Macraigor
PE micro
RDI
JTAGjet
ST-LINK
Third-Party Driver
TI XDS100

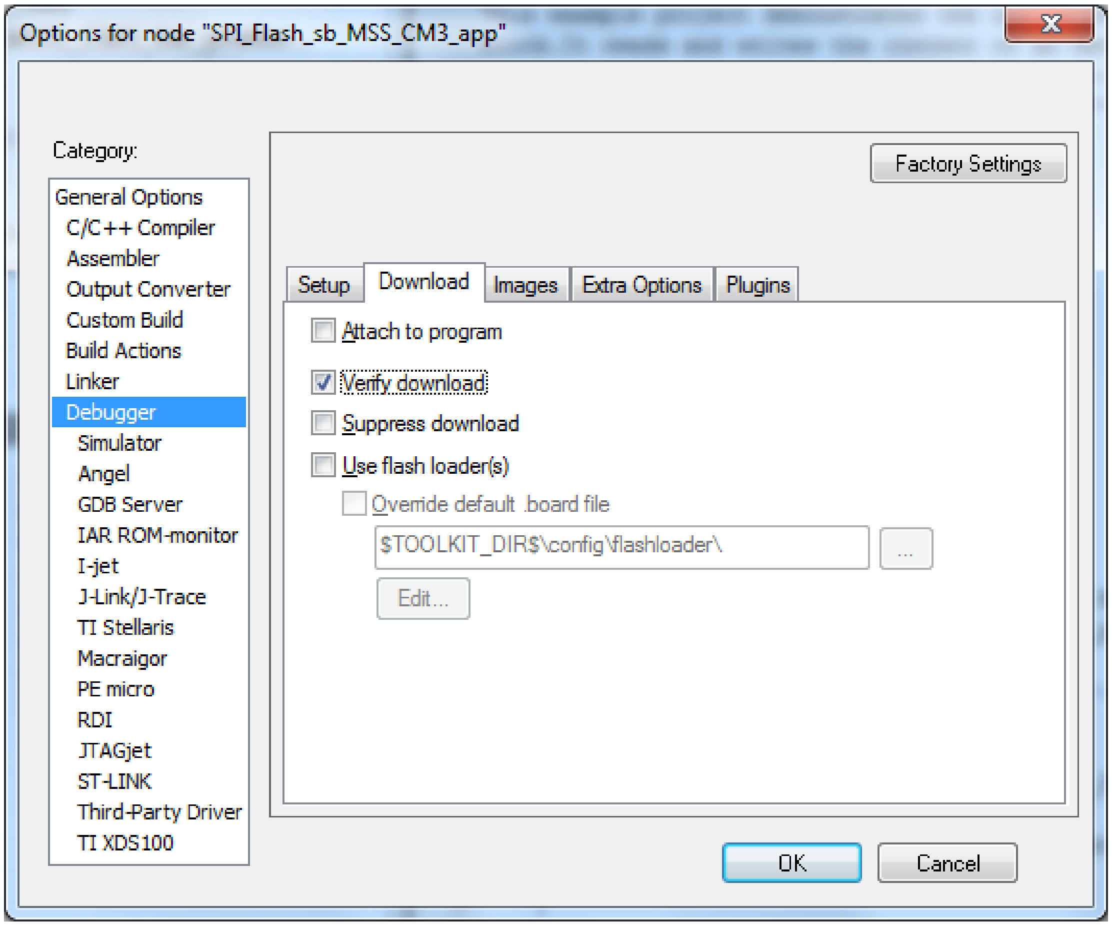
[✓] Run to: main

_sb_MSS_CM3_hw_platform.mac

OK | Cancel

16. Click **Download** tab and select the **Verify download** check box, as shown in Figure 34.

Figure 34 • **IAR Debugger Options - Download**



Options for node "SPI_Flash_sb_MSS_CM3_app"

Category:

- General Options
  - C/C++ Compiler
  - Assembler
  - Output Converter
  - Custom Build
  - Build Actions
  - Linker
  - Debugger
    - Simulator
    - Angel
    - GDB Server
    - IAR ROM-monitor
    - I-jet
    - J-Link/J-Trace
    - TI Stellaris
    - Macraigor
    - PE micro
    - RDI
    - JTAGjet
    - ST-LINK
    - Third-Party Driver
    - TI XDS100

Factory Settings

Setup | Download | Images | Extra Options | Plugins

- [ ] Attach to program
- [x] Verify download
- [ ] Suppress download
- [ ] Use flash loader(s)
  - [ ] Override default .board file

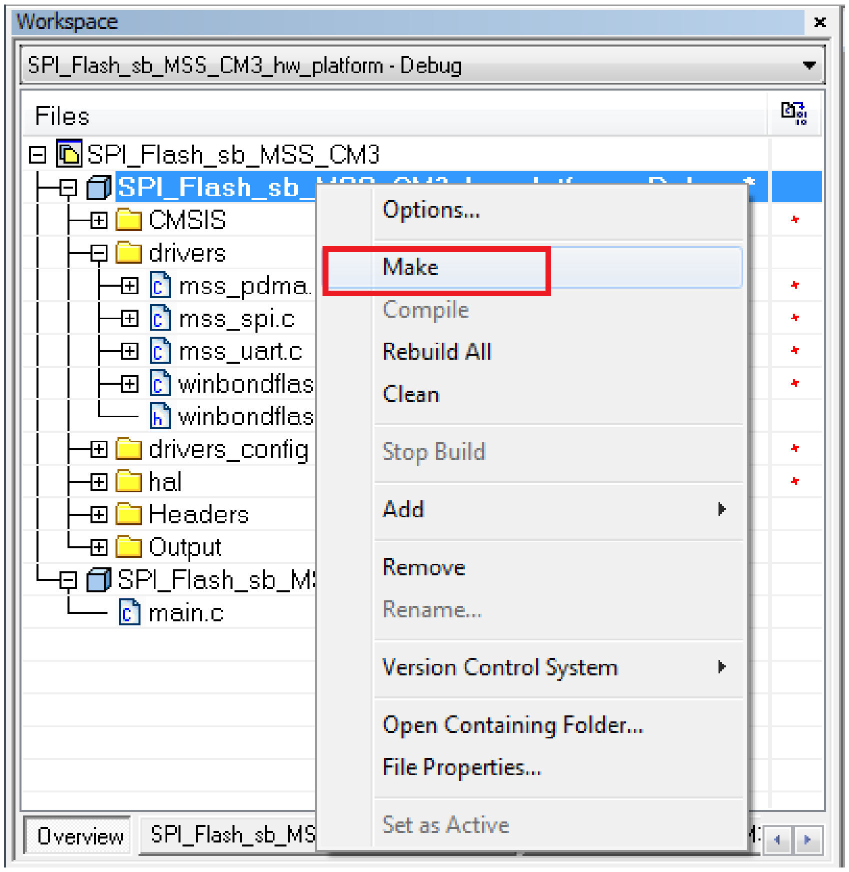
$TOOLKIT_DIR$\config\flashloader\

Edit...

OK | Cancel

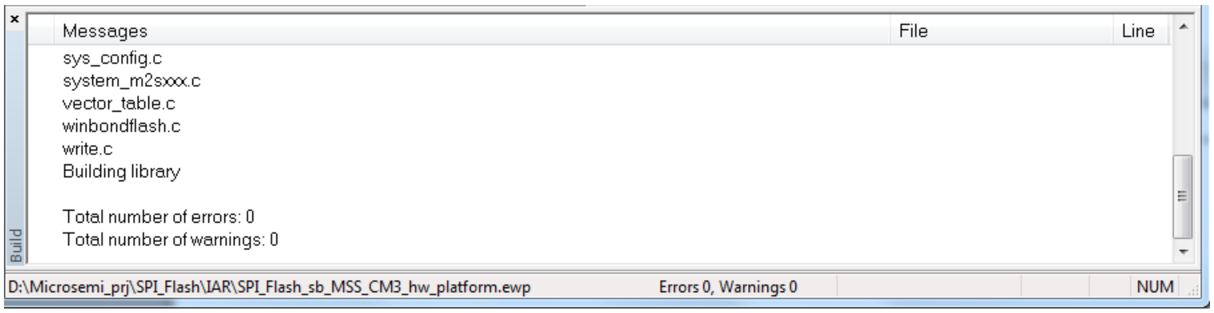
17. Click **OK** to close the **Options** window and build the project.
18. Right-click **SPI_Flash_sb_MSS_CM3_hw_platform - Debug** and select **Make**, as shown in Figure 35 and Figure 36.

Figure 35 • **IAR Workspace - Hardware Platform Code Compilation using Make**



Successful Hardware Platform Code Compilation page is displayed as shown in Figure 36.

Figure 36 • **IAR Workspace - Successful Hardware Platform Code Compilation using Make**

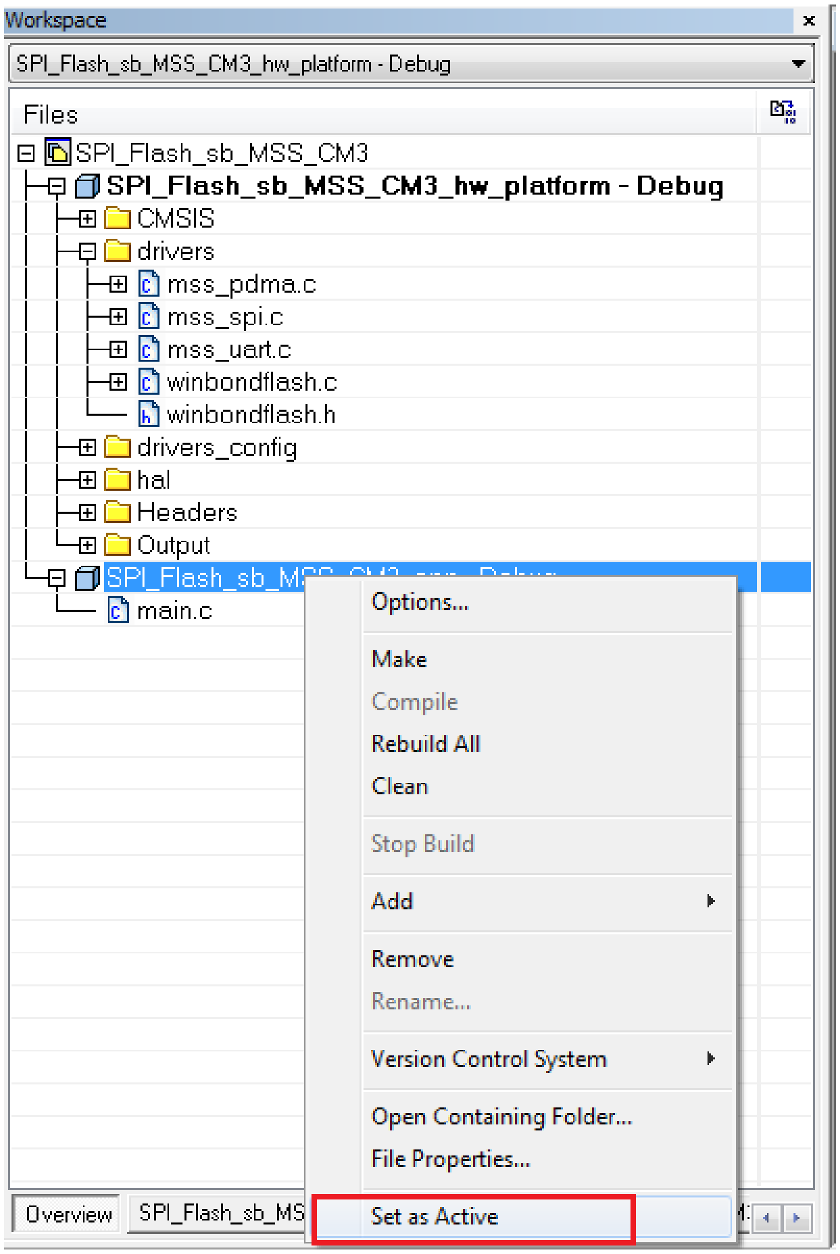


Messages	File	Line
sys_config.c		
system_m2sxxx.c		
vector_table.c		
winbondflash.c		
write.c		
Building library		
Total number of errors: 0		
Total number of warnings: 0		

```
D:\Microsemi_prj\SPI_Flash\IAR\SPI_Flash_sb_MSS_CM3_hw_platform.ewp    Errors 0, Warnings 0    NUM
```

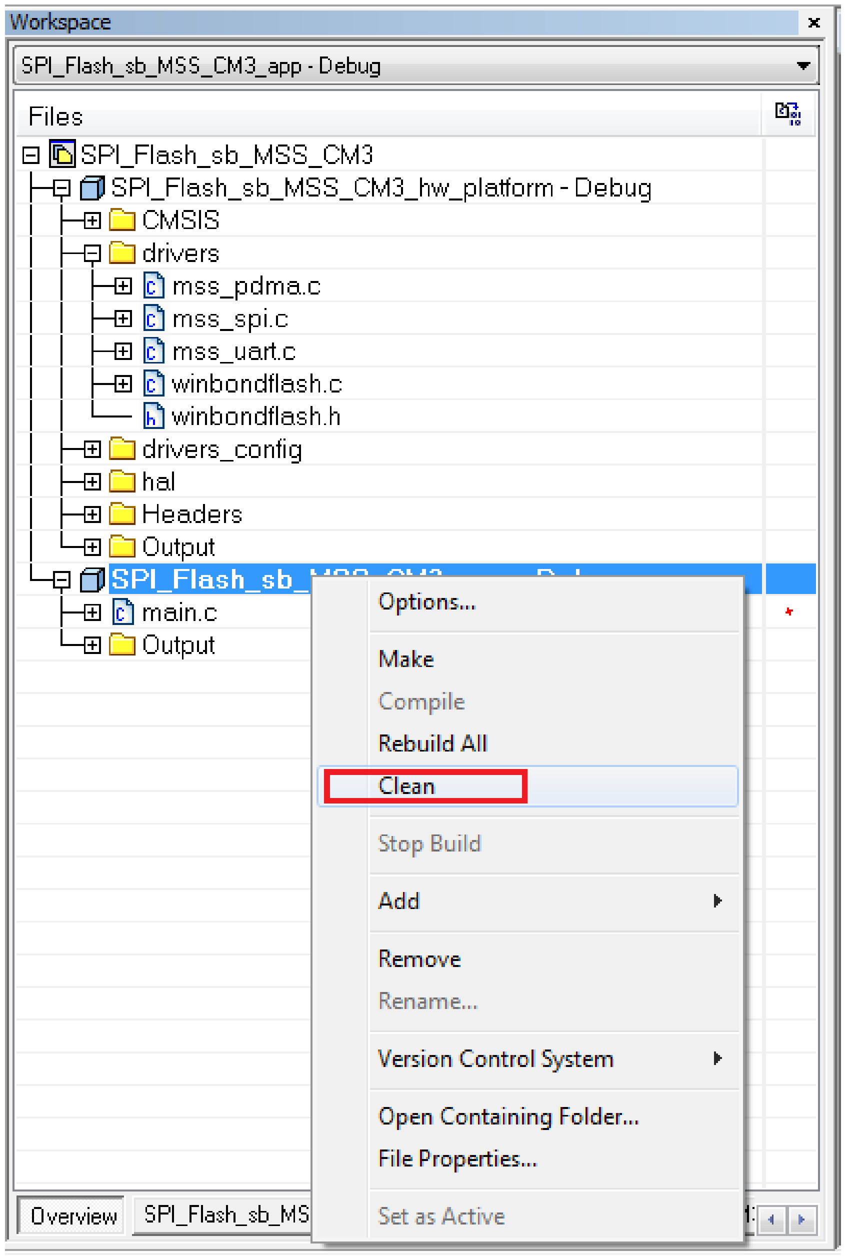
19. Right-click **SPI_Flash_sb_MSS_CM3_app - Debug** project name and select **Set as Active**, as shown in Figure 37.

Figure 37 • **IAR Workspace - SPI_Flash_sb_MSS_CM3_app Set as Active**



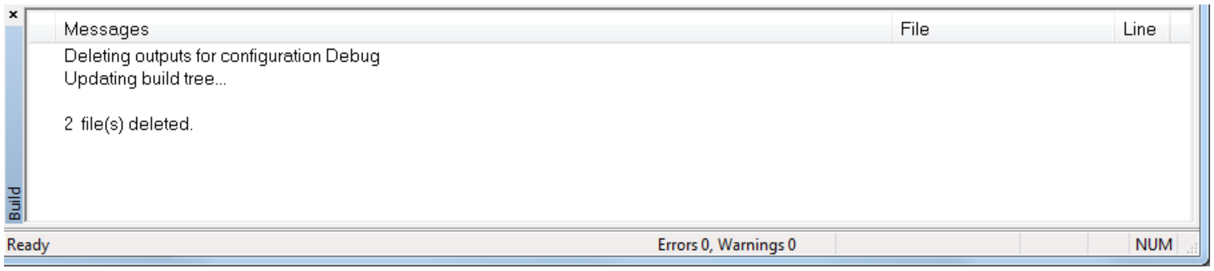
20. Right-click **SPI_Flash_sb_MSS_CM3_app - Debug** project name and select **Clean**, as shown in Figure 38.

Figure 38 • **IAR Workspace - Execute Clean on SPI_Flash_sb_MSS_CM3_app Project**



21. After cleaning the project, the **Messages** log section shows that some files are deleted, as shown in Figure 39.

Figure 39 • **IAR Workspace - Deleted Files**



```
Messages                                          File          Line
Deleting outputs for configuration Debug
Updating build tree...

2  file(s) deleted.

Ready                         Errors 0, Warnings 0              NUM
```

22. Right-click **SPI_Flash_sb_MSS_CM3_app - Debug** project name and click **Rebuild All**, as shown in Figure 40.

Figure 40 • **IAR Workspace - Select Rebuild All**

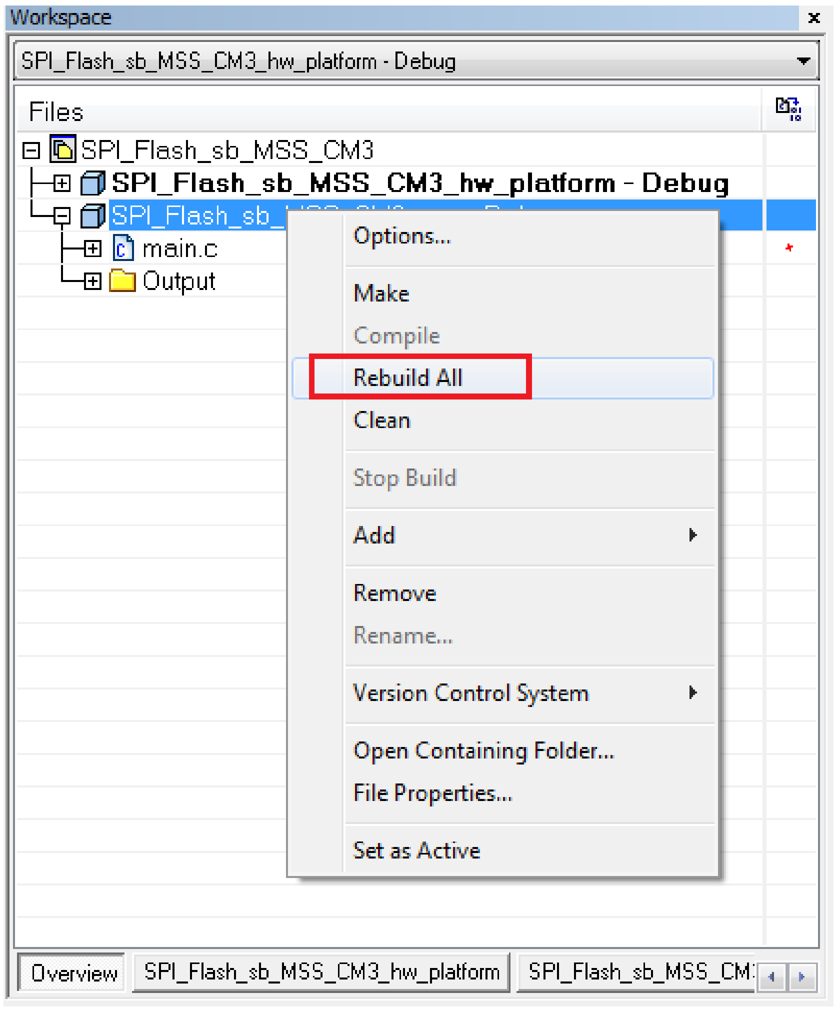
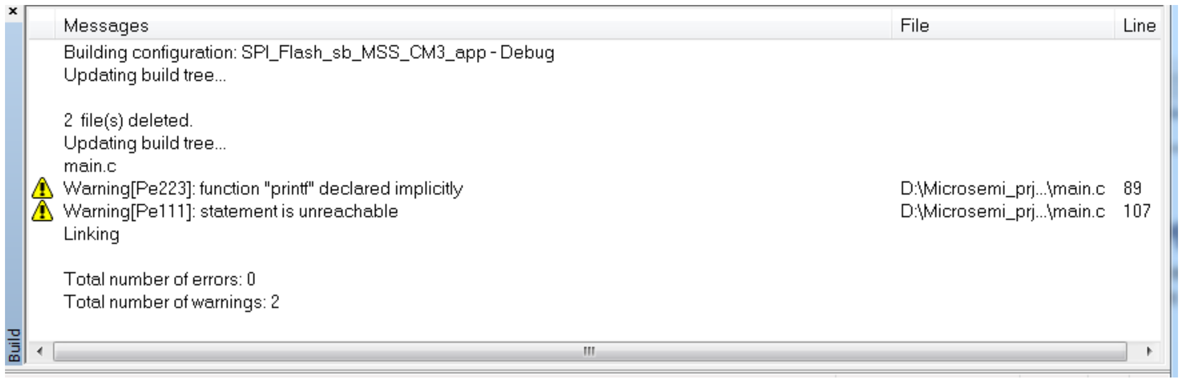


Figure 41 • **IAR Workspace - Rebuild All**



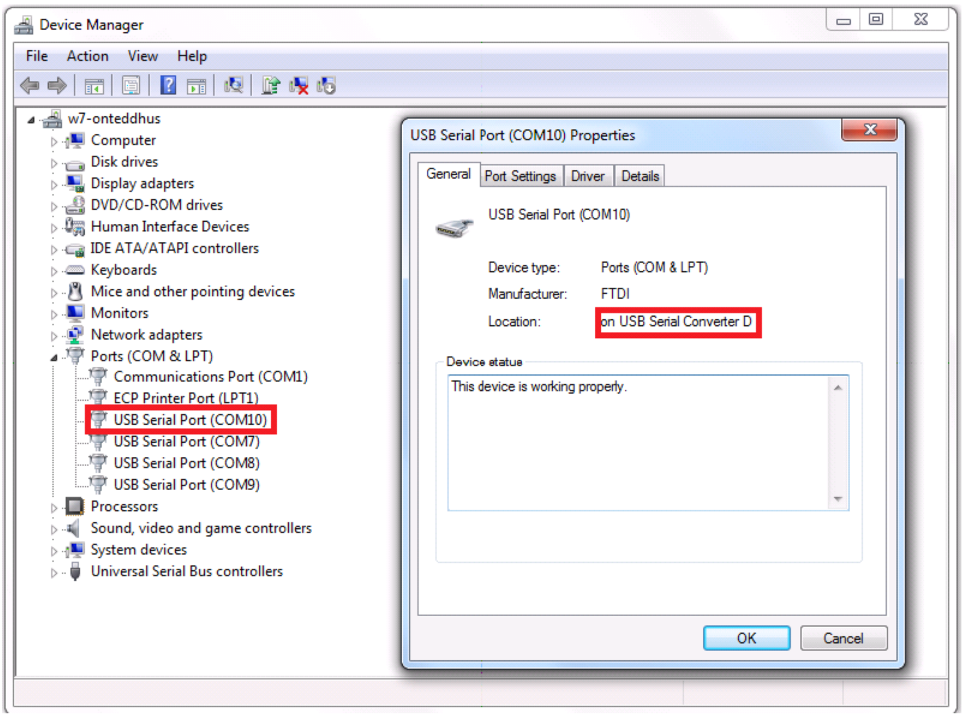
```
Messages                                                                  File                    Line
Building configuration: SPI_Flash_sb_MSS_CM3_app - Debug
Updating build tree...

2 file(s) deleted.
Updating build tree...
main.c
Warning[Pe223]: function "printf" declared implicitly                    D:\Microsemi_prj...\main.c  89
Warning[Pe111]: statement is unreachable                                 D:\Microsemi_prj...\main.c  107
Linking

Total number of errors: 0
Total number of warnings: 2
```

2.9 Step 6: Configuring Serial Terminal Emulation Program

The following steps describe how to configure serial terminal emulation program:

1. Install the USB driver. For serial terminal communication through the FTDI mini USB cable, install the FTDI D2XX driver. Download the drivers and the installation guide from: *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip*.
2. Connect the host PC to the **J18** connector using the USB Mini-B cable. The USB to UART bridge drivers are automatically detected. From the four COM ports, select the one with Location as **on USB Serial Converter D**. Figure 42 shows an example **Device Manager** window.

Figure 42 • **Device Manager Window**



USB Serial Port (COM10)

Device type:	Ports (COM & LPT)
Manufacturer:	FTDI
Location:	on USB Serial Converter D

Device status

This device is working properly.

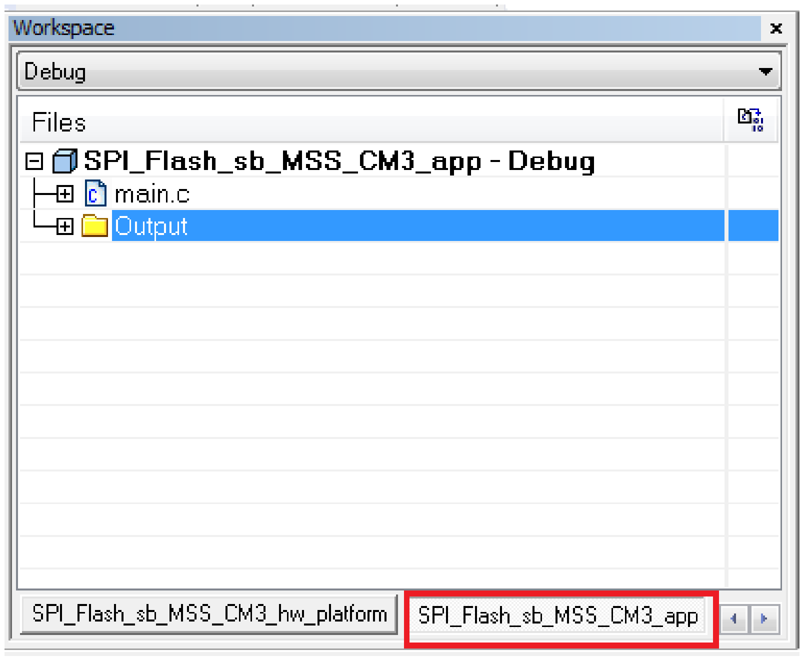
3. Start the HyperTerminal session. If the HyperTerminal program is not available in the computer, any free serial terminal emulation program such as PuTTY or TeraTerm can be used. Refer to the *Configuring Serial Terminal Emulation Programs Tutorial* for configuring the HyperTerminal, TeraTerm, or PuTTY.
   The HyperTerminal settings are as follows:
   - 115200 baud rate
   - 8 data bits
   - 1 stop bit
   - No parity
   - No flow control

2.10 Step 7: Debugging the Application Project using IAR Workbench

The following steps describe how to debug the application project using IAR Workbench:

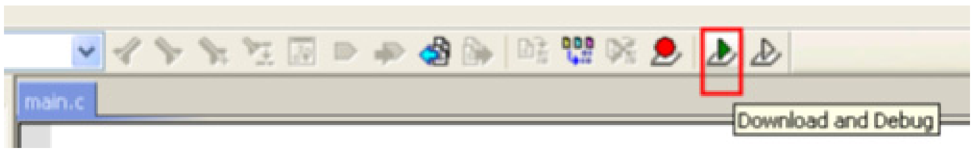
1. Switch to **SPI_Flash_sb_MSS_CM3_app - Debug** tab from Overview tab as shown in Figure 43.

Figure 43 • **Debug Window**


Workspace
Debug
Files
SPI_Flash_sb_MSS_CM3_app - Debug
main.c
Output
SPI_Flash_sb_MSS_CM3_hw_platform
SPI_Flash_sb_MSS_CM3_app

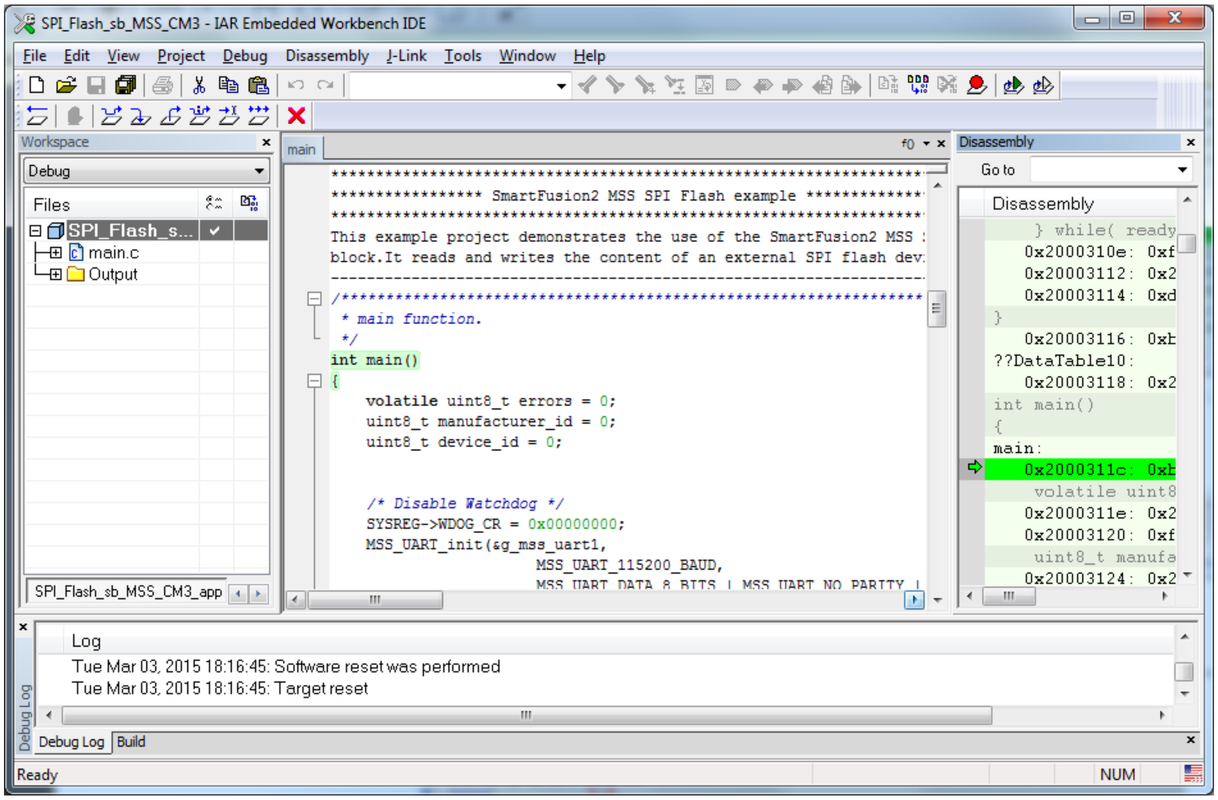
2. In the IAR Workbench, click **Download and Debug** as shown in Figure 44.

Figure 44 • **IAR Workbench - Download and Debug Option**



IAR Debugger Perspective window is opened, as shown in Figure 45.

Figure 45 • **IAR Workbench - Debugger Perspective**


SPI_Flash_sb_MSS_CM3 - IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly J-Link Tools Window Help
Workspace
Debug
Files
SPI_Flash_s...
main.c
Output
SPI_Flash_sb_MSS_CM3_app
main
f()
Disassembly
Go to
Debug Log
Build
Ready
NUM

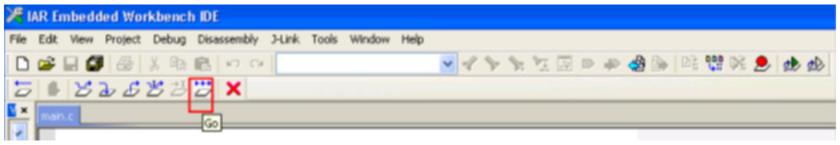
```
*******************************************************************
***************** SmartFusion2 MSS SPI Flash example *************
*******************************************************************
This example project demonstrates the use of the SmartFusion2 MSS :
block.It reads and writes the content of an external SPI flash dev:
-------------------------------------------------------------------
/******************************************************************
 * main function.
 */
int main()
{
    volatile uint8_t errors = 0;
    uint8_t manufacturer_id = 0;
    uint8_t device_id = 0;


    /* Disable Watchdog */
    SYSREG->WDOG_CR = 0x00000000;
    MSS_UART_init(&g_mss_uart1,
                  MSS_UART_115200_BAUD,
                  MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY |
```

```
Disassembly
    } while( ready
   0x2000310e: 0xf
   0x20003112: 0x2
   0x20003114: 0xd
 }
   0x20003116: 0xb
??DataTable10:
   0x20003118: 0x2
 int main()
 {
main:
   0x2000311c: 0xb
    volatile uint8
   0x2000311e: 0x2
   0x20003120: 0xf
    uint8_t manufa
   0x20003124: 0x2
```

```
Log
Tue Mar 03, 2015 18:16:45: Software reset was performed
Tue Mar 03, 2015 18:16:45: Target reset
```

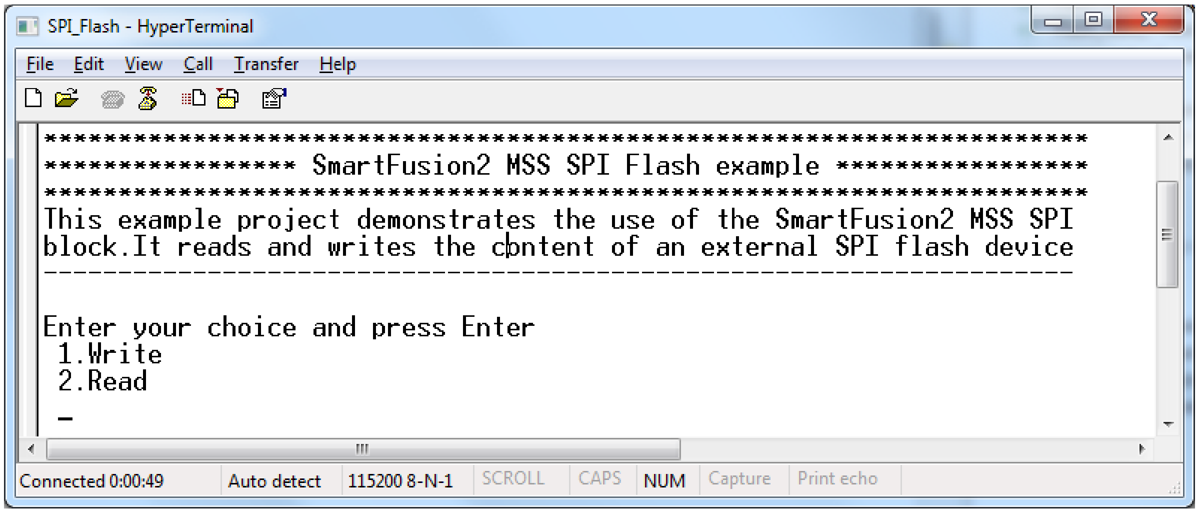
3. Click **Go** on IAR workbench to run the application, as shown in Figure 46.

Figure 46 • **IAR Workbench - Go Option**


IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly J-Link Tools Window Help
main.c
Go

4. On successful operation, the HyperTerminal window displays a message, as shown in Figure 47.

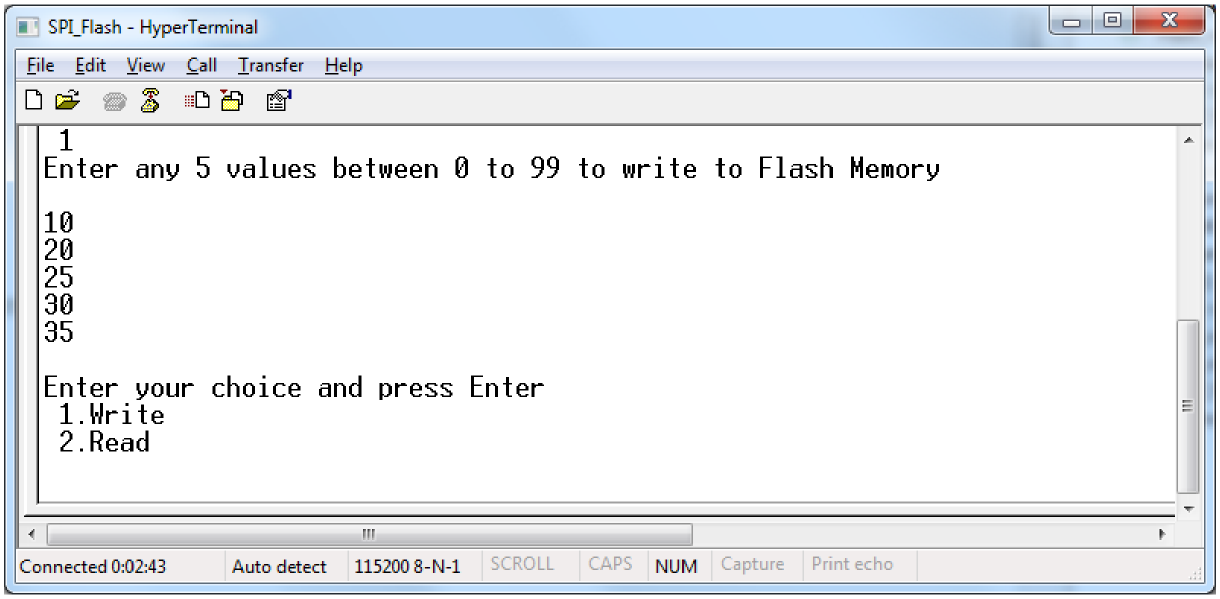
Figure 47 • **HyperTerminal Window**



```
*********************************************************************
***************** SmartFusion2 MSS SPI Flash example *****************
*********************************************************************
This example project demonstrates the use of the SmartFusion2 MSS SPI
block.It reads and writes the content of an external SPI flash device
--------------------------------------------------------------------

Enter your choice and press Enter
 1.Write
 2.Read
 _
```

5. Select option **1** and enter values to write to the SPI Flash Memory, as shown in Figure 48.

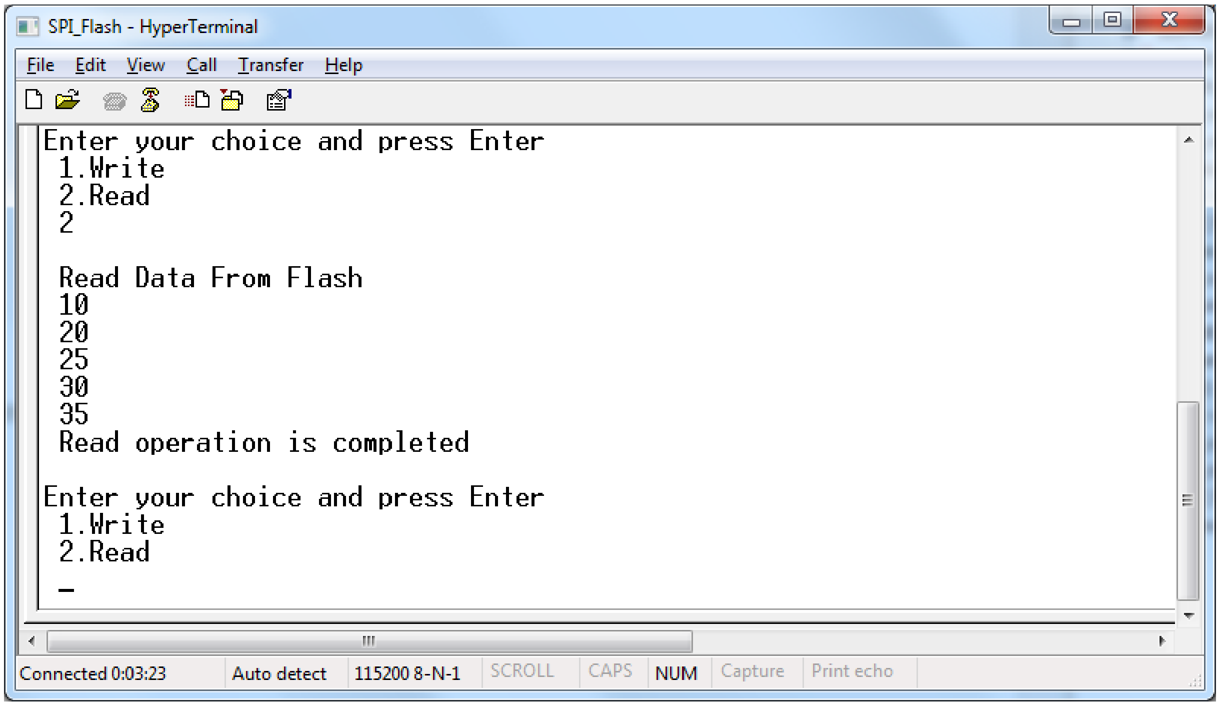
Figure 48 • **HyperTerminal Window - Option 1**



```
 1
Enter any 5 values between 0 to 99 to write to Flash Memory

10
20
25
30
35

Enter your choice and press Enter
 1.Write
 2.Read
```

6. Select option **2** to read data from SPI Flash Memory, as shown in Figure 49.

Figure 49 • **HyperTerminal Window - Option 2**

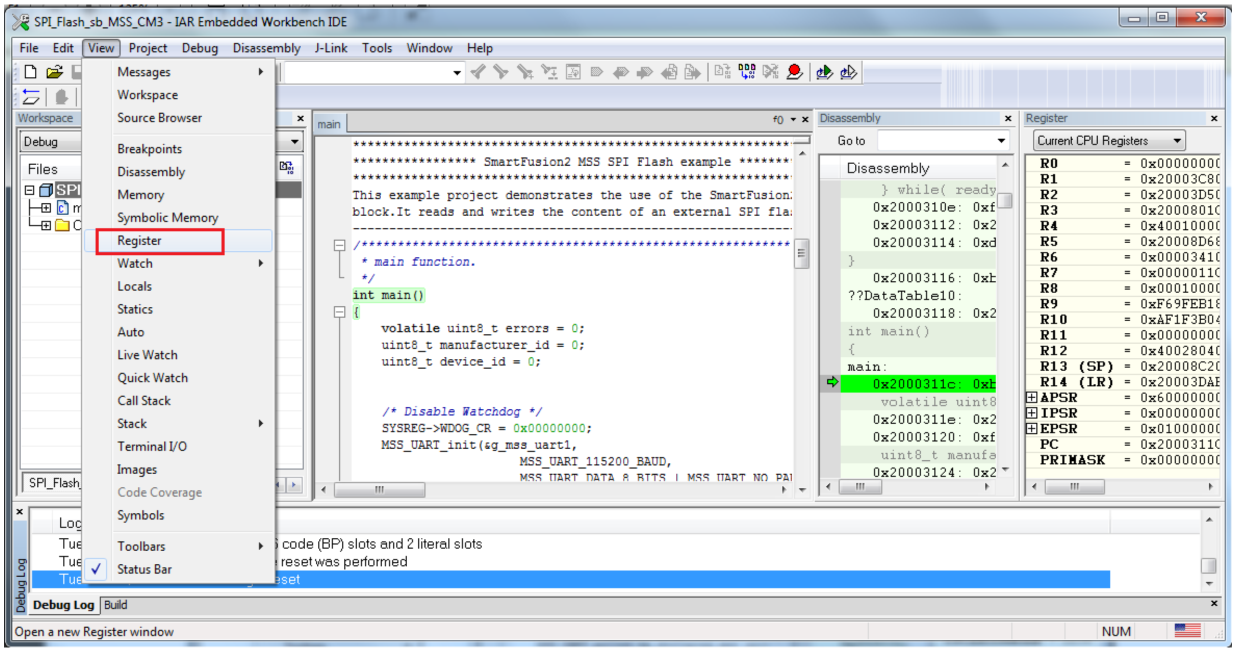


```
Enter your choice and press Enter
 1.Write
 2.Read
 2

 Read Data From Flash
 10
 20
 25
 30
 35
 Read operation is completed

Enter your choice and press Enter
 1.Write
 2.Read

 _
```

Connected 0:03:23 | Auto detect | 115200 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

7. Click **View > Register** to view the values of the ARM® Cortex®-M3 processor internal registers, as shown in Figure 50.

Figure 50 • **Values of Cortex-M3 Internal Registers**



```
*****************************************************************
**************** SmartFusion2 MSS SPI Flash example *******
*****************************************************************
This example project demonstrates the use of the SmartFusion
block.It reads and writes the content of an external SPI fla
-----------------------------------------------------------------
/****************************************************************
 * main function.
 */
int main()
{
    volatile uint8_t errors = 0;
    uint8_t manufacturer_id = 0;
    uint8_t device_id = 0;


    /* Disable Watchdog */
    SYSREG->WDOG_CR = 0x00000000;
    MSS_UART_init(&g_mss_uart1,
                  MSS_UART_115200_BAUD,
                  MSS_UART_DATA_8_BITS | MSS_UART_NO_PAI
```

```
Disassembly
    } while( ready
    0x2000310e: 0xf
    0x20003112: 0x2
    0x20003114: 0xd
}
    0x20003116: 0xb
??DataTable10:
    0x20003118: 0x2
int main()
{
main:
    0x2000311c: 0xb
    volatile uint8
    0x2000311e: 0x2
    0x20003120: 0xf
    uint8_t manufa
    0x20003124: 0x2
```

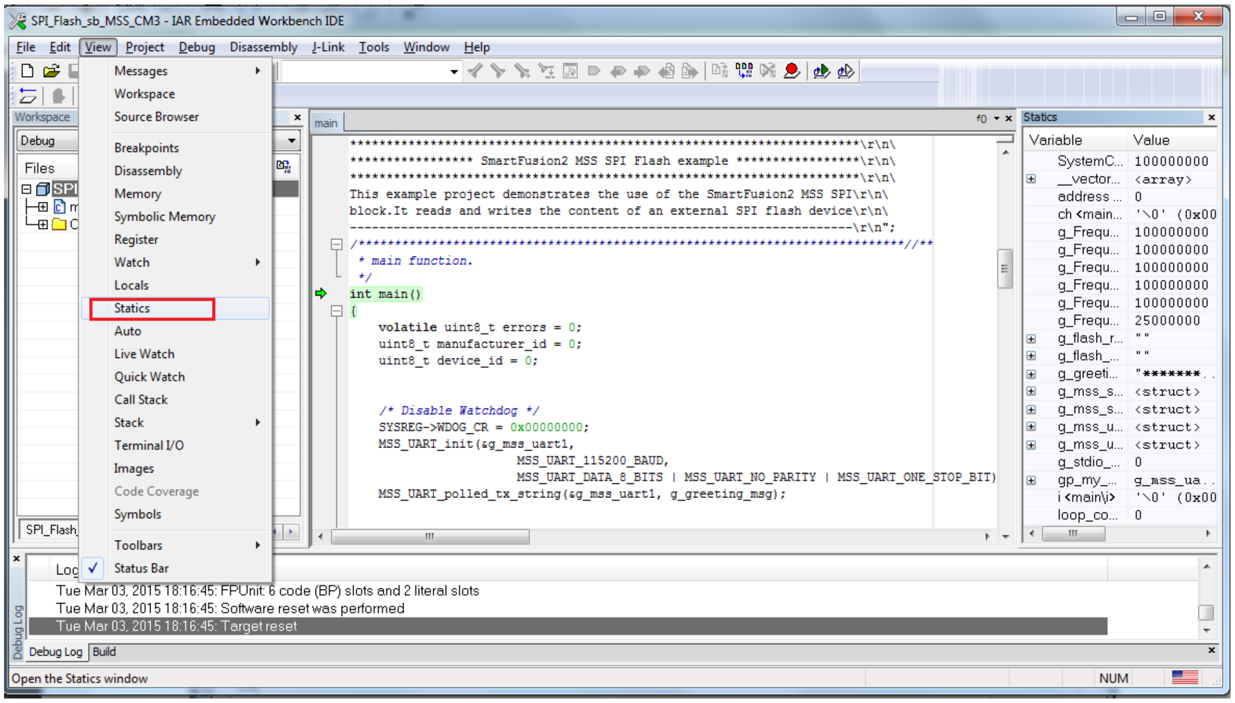
```
R0         = 0x0000000C
R1         = 0x20003C8C
R2         = 0x20003D5C
R3         = 0x2000801C
R4         = 0x4001000C
R5         = 0x20008D68
R6         = 0x0000341C
R7         = 0x0000011C
R8         = 0x0001000C
R9         = 0xF69FEB18
R10        = 0xAF1F3B04
R11        = 0x0000000C
R12        = 0x4002804C
R13 (SP)   = 0x20008C2C
R14 (LR)   = 0x20003DAE
APSR       = 0x6000000C
IPSR       = 0x0000000C
EPSR       = 0x0100000C
PC         = 0x2000311C
PRIMASK    = 0x0000000C
```

```
Log
Tue  code (BP) slots and 2 literal slots
Tue  reset was performed
Tue  set
```

Open a new Register window

8. Click **View > Statics** to view the values of variables in the source code, as shown in Figure 51.

Figure 51 • **Values of Source Code Variables**



```
*******************************************************************\r\n\
**************** SmartFusion2 MSS SPI Flash example ****************\r\n\
*******************************************************************\r\n\
This example project demonstrates the use of the SmartFusion2 MSS SPI\r\n\
block.It reads and writes the content of an external SPI flash device\r\n\
-------------------------------------------------------------------\r\n";
/*********************************************************************//**
 * main function.
 */
int main()
{
    volatile uint8_t errors = 0;
    uint8_t manufacturer_id = 0;
    uint8_t device_id = 0;


    /* Disable Watchdog */
    SYSREG->WDOG_CR = 0x00000000;
    MSS_UART_init(&g_mss_uart1,
                  MSS_UART_115200_BAUD,
                  MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT)
    MSS_UART_polled_tx_string(&g_mss_uart1, g_greeting_msg);
```

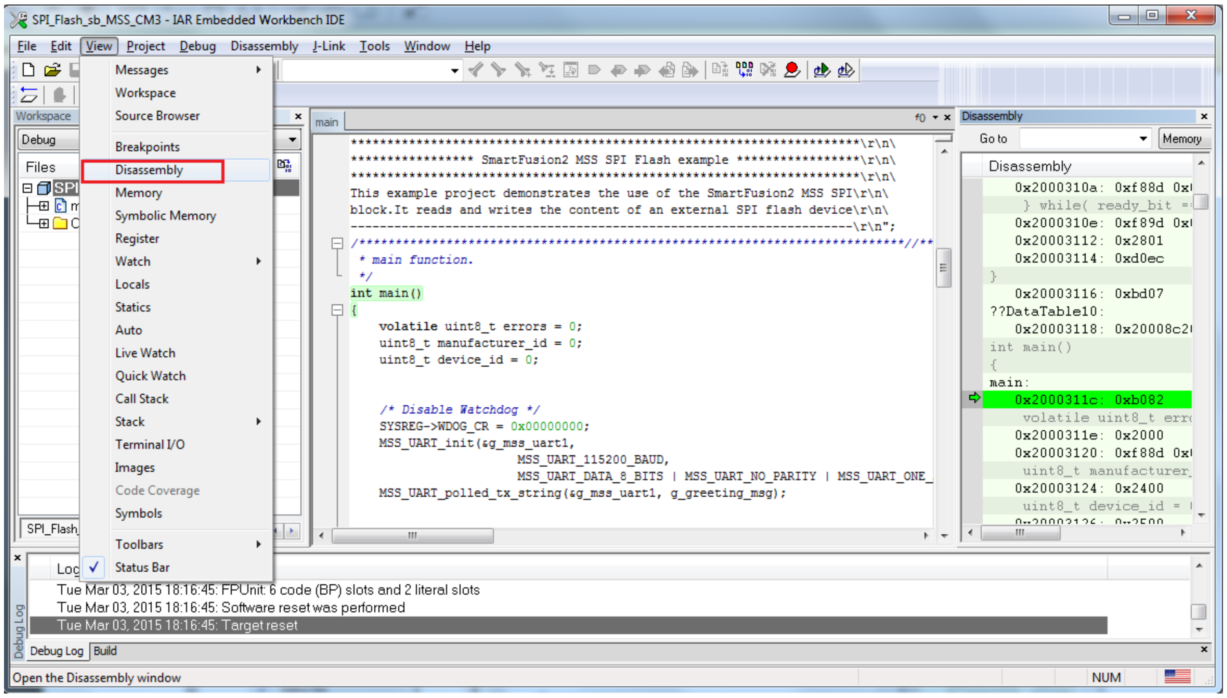
Variable	Value
SystemC...	100000000
__vector...	<array>
address ...	0
ch <main...	'\0' (0x00
g_Frequ...	100000000
g_Frequ...	100000000
g_Frequ...	100000000
g_Frequ...	100000000
g_Frequ...	100000000
g_Frequ...	25000000
g_flash_r...	""
g_flash_...	""
g_greeti...	"******* ..
g_mss_s...	<struct>
g_mss_s...	<struct>
g_mss_u...	<struct>
g_mss_u...	<struct>
g_stdio_...	0
gp_my_...	g_mss_ua..
i <main\i>	'\0' (0x00
loop_co...	0

```
Log
Tue Mar 03, 2015 18:16:45: FPUnit: 6 code (BP) slots and 2 literal slots
Tue Mar 03, 2015 18:16:45: Software reset was performed
Tue Mar 03, 2015 18:16:45: Target reset
```

Open the Statics window

9. Click **View > Disassembly** to view the values of variables in the source code, as shown in Figure 52.

Figure 52 • **Assembly Level Instructions**



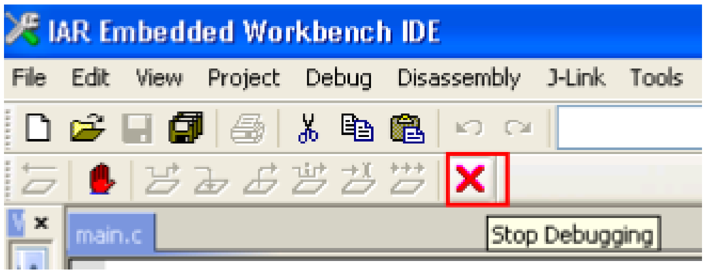
```
*****************************************************************\r\n\
**************** SmartFusion2 MSS SPI Flash example ****************\r\n\
*****************************************************************\r\n\
This example project demonstrates the use of the SmartFusion2 MSS SPI\r\n\
block.It reads and writes the content of an external SPI flash device\r\n\
-----------------------------------------------------------------\r\n";
/*****************************************************************//**
 * main function.
 */
int main()
{
    volatile uint8_t errors = 0;
    uint8_t manufacturer_id = 0;
    uint8_t device_id = 0;


    /* Disable Watchdog */
    SYSREG->WDOG_CR = 0x00000000;
    MSS_UART_init(&g_mss_uart1,
                  MSS_UART_115200_BAUD,
                  MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_
    MSS_UART_polled_tx_string(&g_mss_uart1, g_greeting_msg);
```

```
Disassembly
    0x2000310a: 0xf88d 0x
     } while( ready_bit =
    0x2000310e: 0xf89d 0x
    0x20003112: 0x2801
    0x20003114: 0xd0ec
}
    0x20003116: 0xbd07
??DataTable10:
    0x20003118: 0x20008c2
int main()
{
main:
    0x2000311c: 0xb082
     volatile uint8_t err
    0x2000311e: 0x2000
    0x20003120: 0xf88d 0x
     uint8_t manufacturer_
    0x20003124: 0x2400
     uint8_t device_id =
```

```
Log
Tue Mar 03, 2015 18:16:45: FPUnit: 6 code (BP) slots and 2 literal slots
Tue Mar 03, 2015 18:16:45: Software reset was performed
Tue Mar 03, 2015 18:16:45: Target reset
```

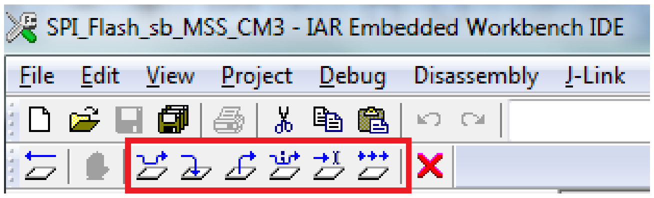
10. When debug process is finished, terminate execution of the code by choosing **Debug > Stop Debugging**, as shown in Figure 53.

Figure 53 • **IAR Workbench - Stop Debugging Option**


IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly J-Link Tools
main.c
Stop Debugging

11. The Step Level Debugging can be performed before running the application using Go. This can be accessed from the Debug menu or on the IAR workbench, as shown in Figure 54.

Figure 54 • **IAR Workbench - Step Level Debugging**


SPI_Flash_sb_MSS_CM3 - IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly J-Link

- Source code can be single-stepped by selecting from the Debug menu **Debug > Step Into**, **Debug > Step Out**, **Debug > Step Over** or selecting the respective options from the IAR workbench as shown in Figure 54. Observe the changes in the source code window and Disassembly view. Performing a Step Over provides an option for stepping over functions. The entire function is run but there is no need to single-step through each instruction contained in the function.

12. Close **Debug Perspective** by selecting **Close Perspective** from the Window menu.
13. Close IAR Embedded Workbench using **File > Exit**.
14. Close the HyperTerminal using **File > Exit**.

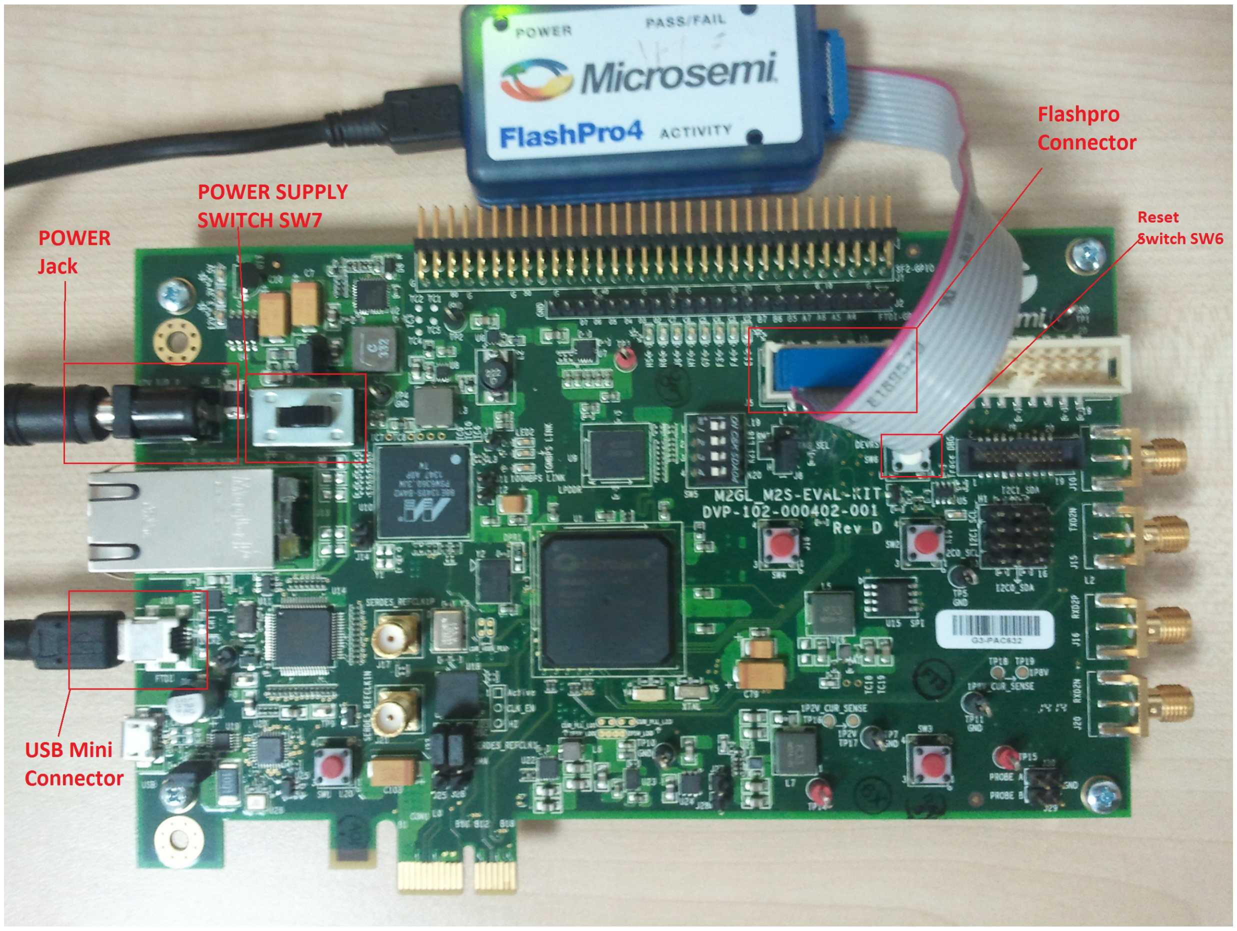
2.11 Conclusion

This tutorial provides steps to create a Libero SoC design using System Builder. It describes the procedure to build, debug, and run an IAR Embedded Workbench application. It also provides a simple design to access the SPI flash.

3 Appendix: Board Setup for Programming the Tutorial

Figure 55 on page 46 shows the board setup for programming the tutorial on the SmartFusion2 Security Evaluation Kit board.

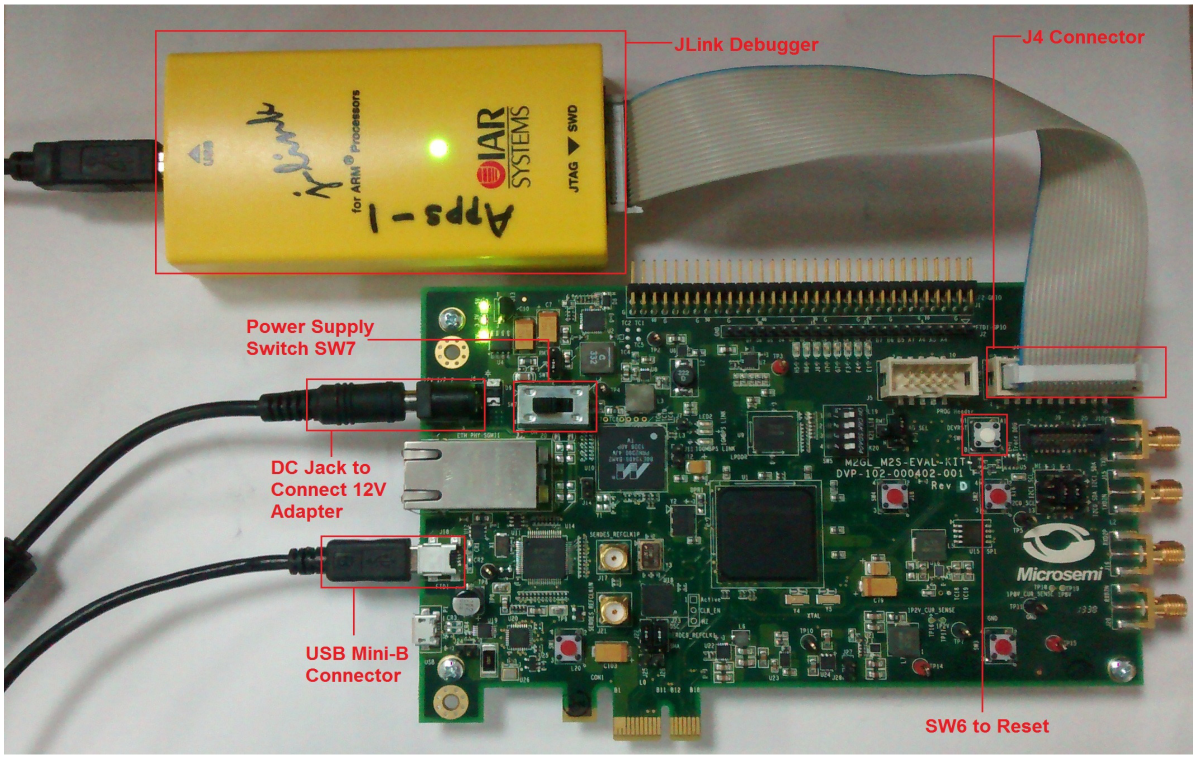
Figure 55 • **SmartFusion2 Security Evaluation Kit Setup**



4 Appendix: Board Setup for Running the IAR Tutorial

Figure 56 on page 47 shows the board setup for running and debugging the tutorial on the SmartFusion2 Security Evaluation Kit board.

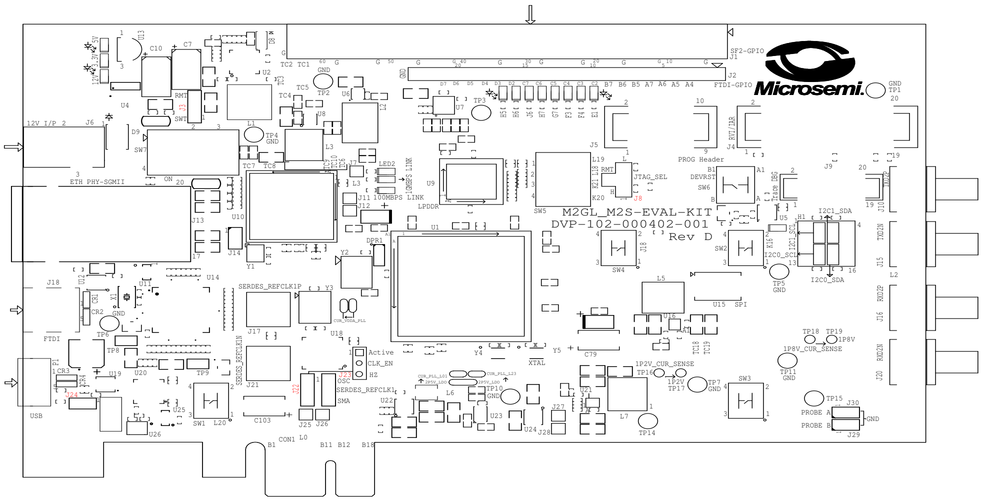
Figure 56 • **SmartFusion2 Security Evaluation Kit J-Link Programmer Connection**



5 Appendix: SmartFusion2 Security Evaluation Kit Board Jumper Locations

Figure 57 on page 48 shows the jumper locations on the SmartFusion2 Security Evaluation Kit board.

Figure 57 • **SmartFusion2 Security Evaluation Kit Board Jumper Locations**



Note:

- Jumpers highlighted in red (J22, J23, J24, J8, and J3) are set by default.
- The location of the jumpers in Figure 57 on page 48 are searchable.

6 Revision History

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

Revision	Changes
Revision 5 (April 2016)	Updated the document for Libero SoC v11.7 software release (SAR 77812)
Revision 4 (October 2015)	Updated the document for Libero SoC v11.6 software release (SAR 72826)
Revision 3 (March 2015)	Updated the document for Libero SoC v11.5 software release (SAR 64188).
Revision 2 (November 2014)	Updated the document for Libero SoC v11.4 software release (SAR 61628).
Revision 1 (April 2014)	Initial release.

7 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.

7.1 Customer Service

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

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

7.2 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.

7.3 Technical Support

For Microsemi SoC Products Support, visit
http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support.

7.4 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/products/fpga-soc/fpga-and-soc.

7.5 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.

7.5.1 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.

7.5.2 My Cases

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

7.5.3 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. Visit About Us for sales office listings and corporate contacts.

7.6 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@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 Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif, and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com

50200547-5/4.16