Accessing Serial Flash Memory Using SPI Interface

Libero SoC v11.5 and Keil uVision Flow Tutorial for SmartFusion2 TU0548 Tutorial

Microsemi®

Table of Contents

Accessing Serial Flash Memory using SPI Interface - Libero SoC v11.5 and Keil uVision Flow Tutorial for SmartFusion2

Introduction

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

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

- *Accessing Serial Flash Memory using SPI Interface - Libero SoC and SoftConsole Flow Tutorial for SmartFusion2 SoC FPGA*
- *Accessing Serial Flash Memory using SPI Interface - Libero SoC and IAR Embedded Workbench Flow Tutorial for SmartFusion2 SoC FPGA*

This tutorial describes the following:

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

Design Requirements

Table 1 • **Design Requirements**

Design Requirements	**Description**
Hardware Requirements	
SmartFusion2 Security Evaluation Kit • FlashPro4 programmer • USB A to Mini-B cable • 12 V Adapter	Rev D or later
Keil debugger	-
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.5
Keil uVision	v5
FlashPro programming software	v11.5
Host PC Drivers	USB to UART drivers
Any one of the following serial terminal emulation programs: • HyperTerminal • TeraTerm • PuTTY	-

Project Files

The design files for this tutorial can be downloaded from the Microsemi® website: *http://soc.microsemi.com/download/rsc/?f=m2s_tu0548_liberov11p5_df*

The design files include:

- Libero project
- Programming files
- Source Files
- SPI_Flash_Drivers
- Readme file

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

Target Board

SmartFusion2 Security Evaluation Kit Board, Rev D (or later).

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 W25Q64FVSSIG, which is connected to the SmartFusion2 microcontroller subsystem (MSS) through 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 *SmartFusion2 Microcontroller Subsystem User Guide*.

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

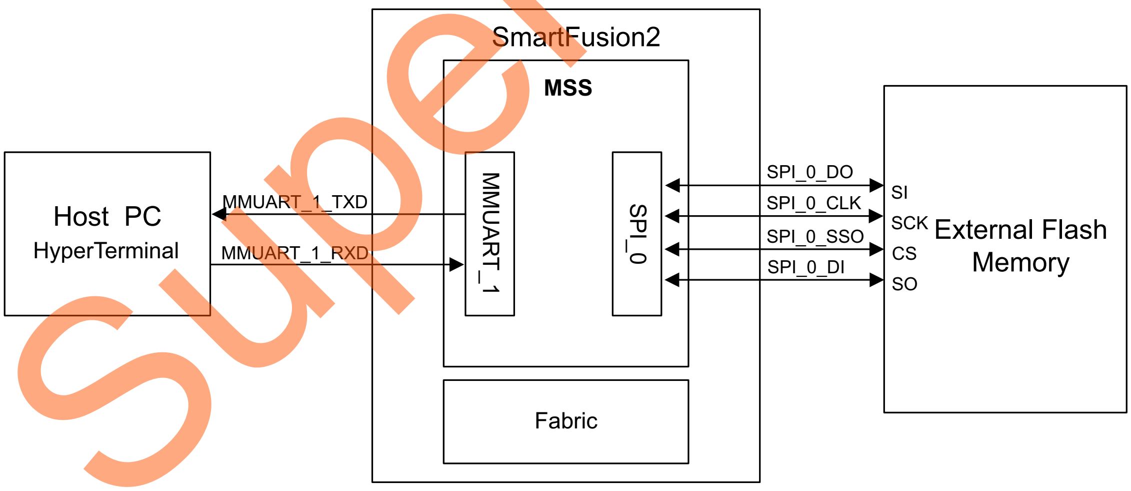


***Figure 1* • SPI Flash Interfacing Block Diagram**

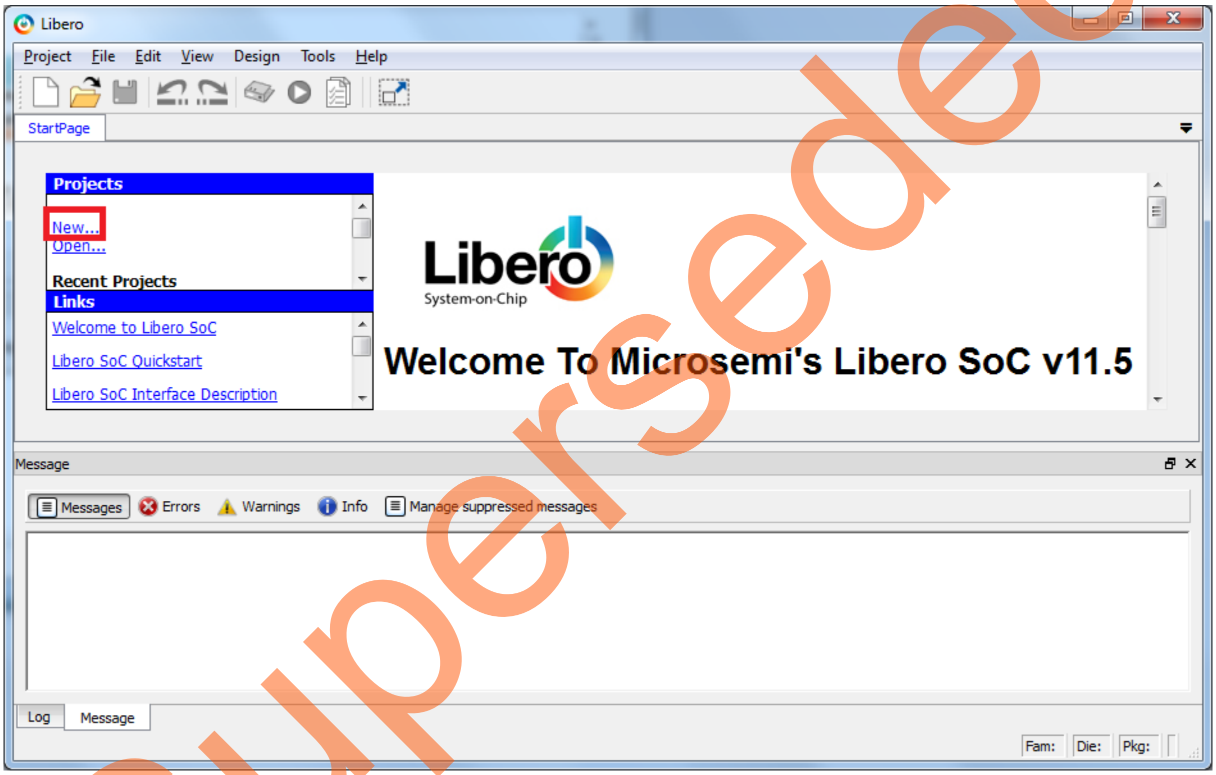
Step 1: Creating a Libero SoC Project

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

Launching Libero SoC

The following steps describe how to launch Libero SoC:

1. Click **Start > Programs > Microsemi Libero SoC v11.5 > Libero SoC v**11.5, or click the shortcut on desktop to open the Libero SoC v11.5 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.

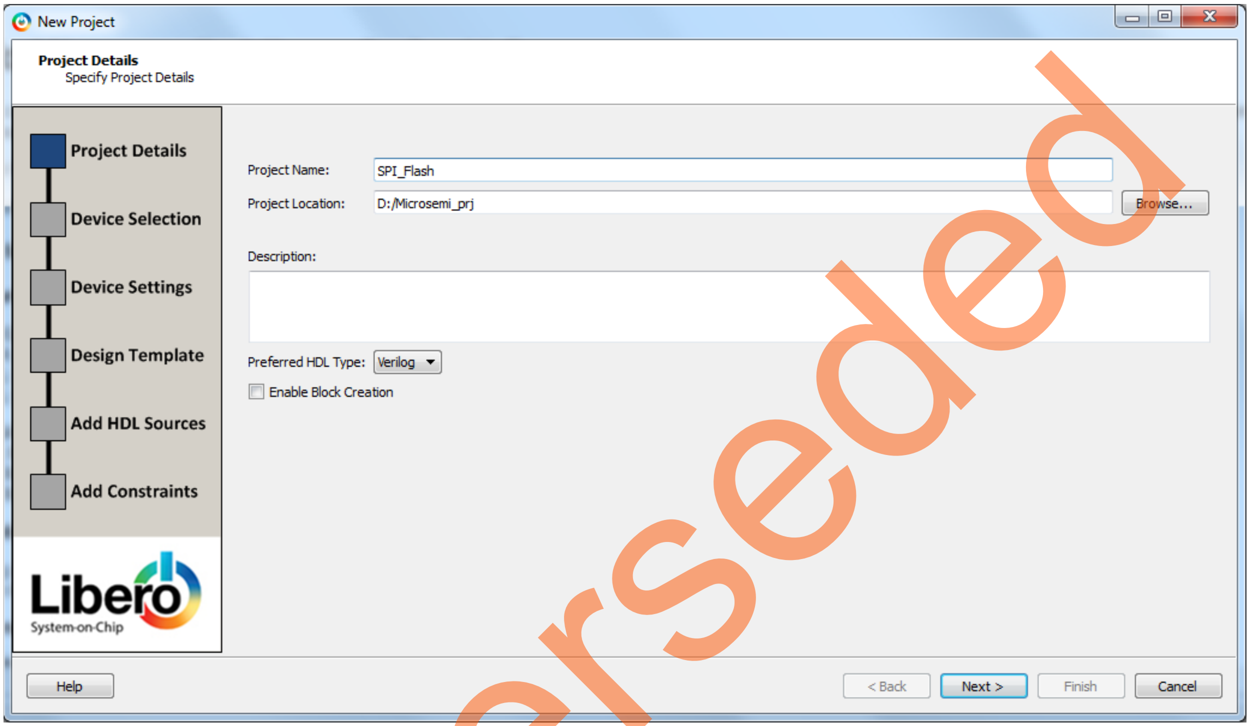


Welcome To Microsemi's Libero SoC v11.5

***Figure 2* • Libero SoC Project Manager**

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



Project Details
Specify Project Details

Project Name: SPI_Flash

Project Location: D:/Microsemi_prj

Description:

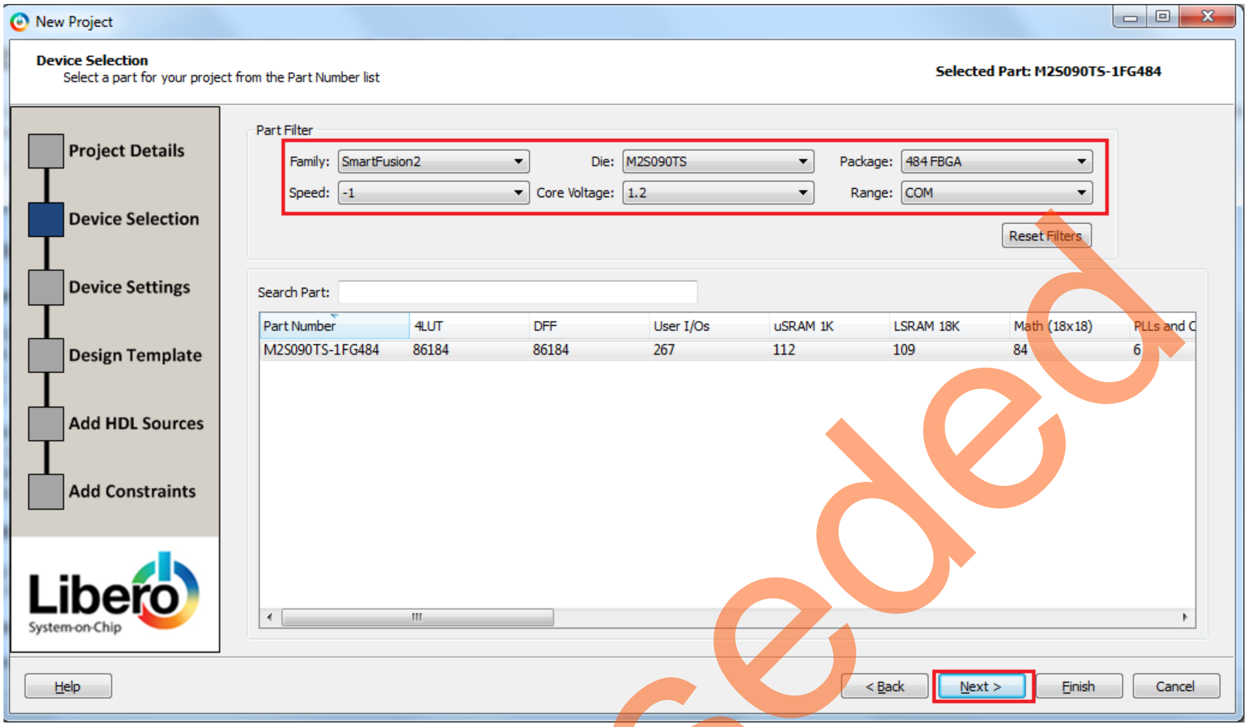
Preferred HDL Type: Verilog

☐ Enable Block Creation

***Figure 3* • Project DetailsPage**

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

– **Range**: COM



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

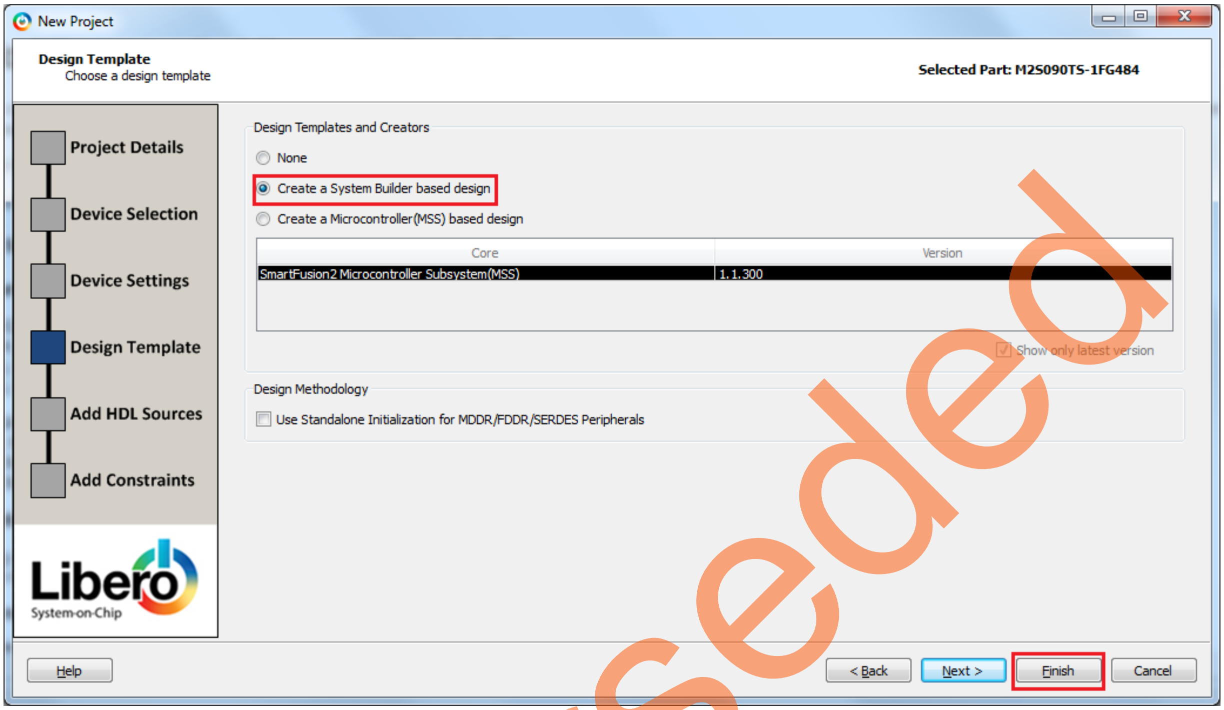
Search Part:

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

Figure 4 **• Device Selection Page**

5. Click **Next.** This opens **Device Settings** page. Do not change the default settings.

6. Click **Next.** This opens **Design Template** page as shown in Figure 5, Under Design Templates and Creators, select **Create a System Builder based design.**



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)	1.1.300

☑ Show only latest version

Design Methodology

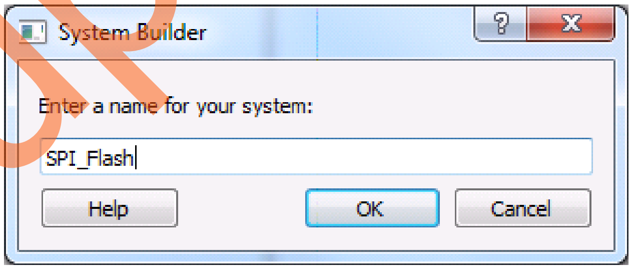
□ Use Standalone Initialization for MDDR/FDDR/SERDES Peripherals

Figure 5 • **Design Template Window**

7. Click **Finish.** This opens **System Builder** window.

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 will define the intended system.

8. Enter the name of the system as **SPI_Flash** and click **OK.**, as shown in Figure 6.



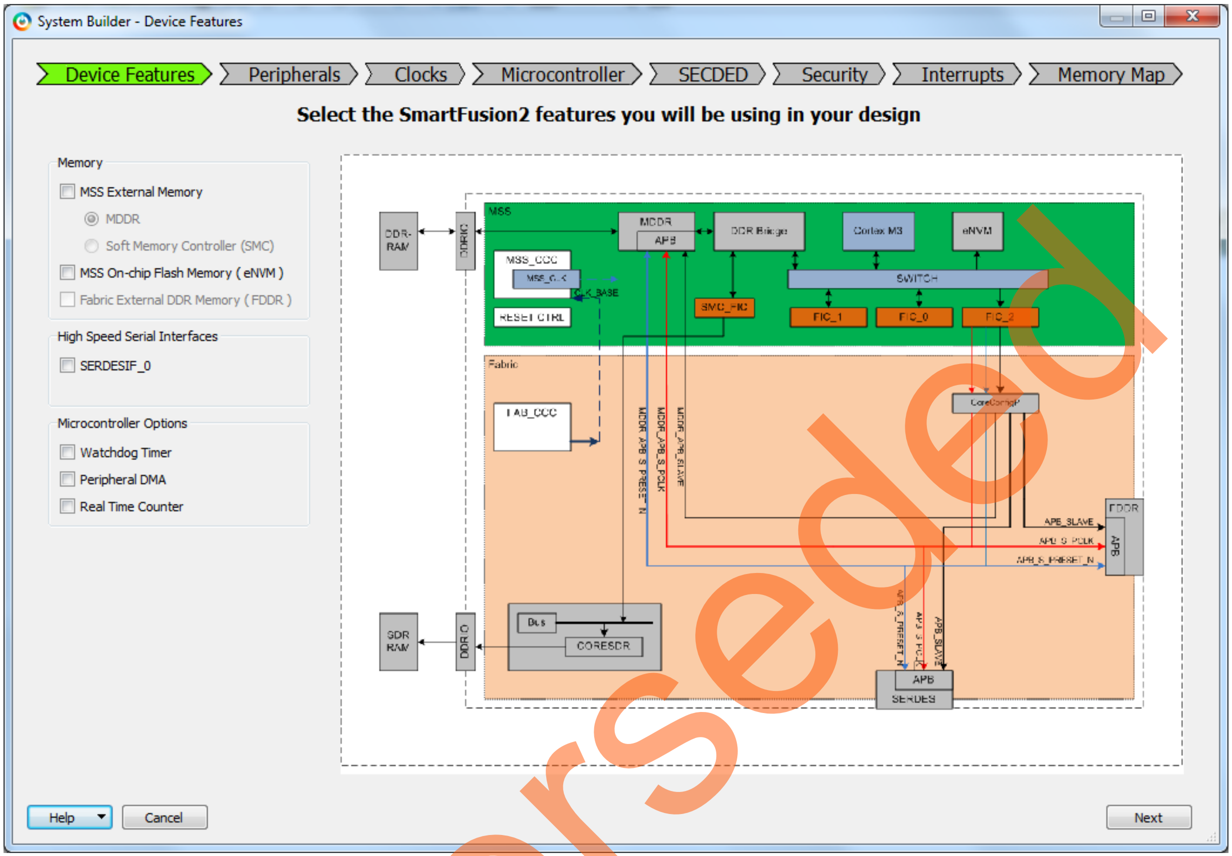
System Builder

Enter a name for your system:

SPI_Flash

Figure 6 • **System Builder Window**

9. Figure 7 shows the **System Builder – Device Features** page.



System Builder - Device Features

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

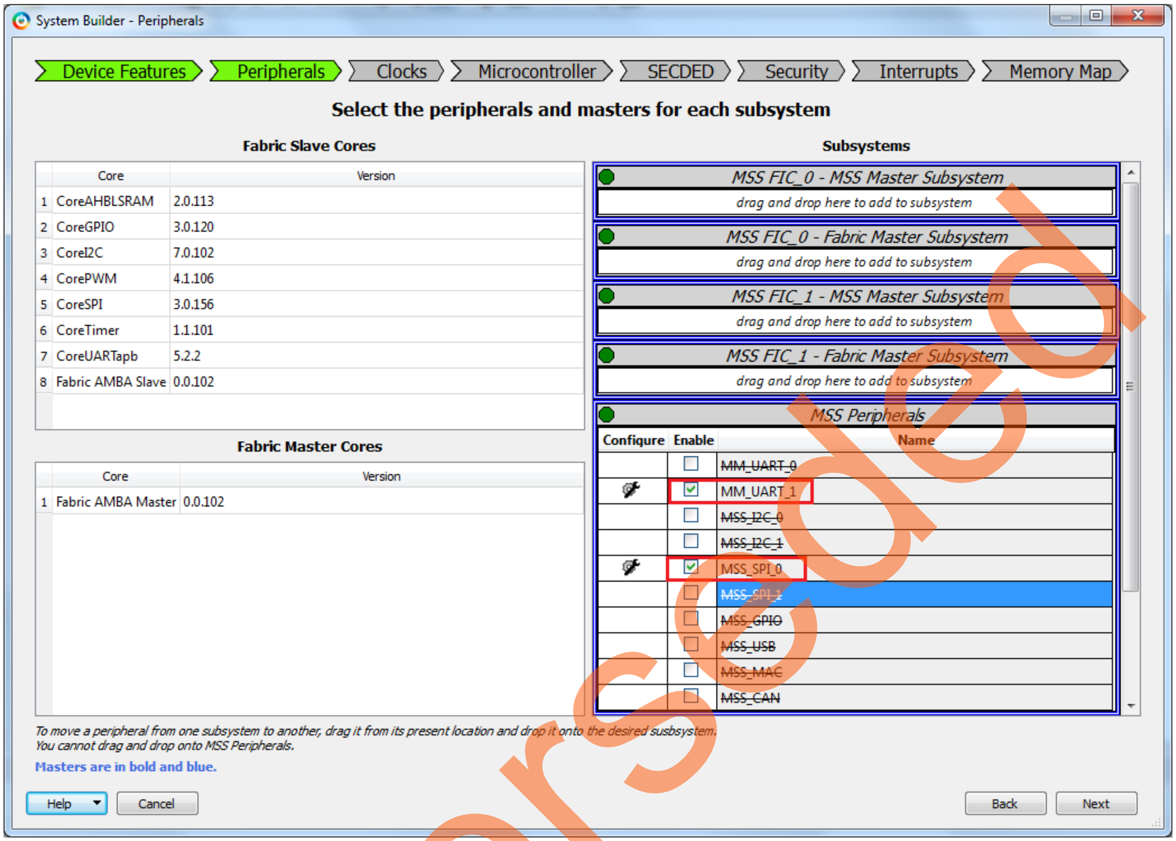
□ Peripheral DMA

□ Real Time Counter

Figure 7 • **System Builder – Device Features Page**

10. Click **Next**. This opens **System Builder - Peripherals** page as shown in Figure 8.

11. Under the MSS Peripherals section, clear all the check boxes except **MM_UART_1** and **MSS_SPI_0**, as shown in Figure 8.



Select the peripherals and masters for each subsystem

Fabric Slave Cores

	Core	Version
1	CoreAHBLSRAM	2.0.113
2	CoreGPIO	3.0.120
3	CoreI2C	7.0.102
4	CorePWM	4.1.106
5	CoreSPI	3.0.156
6	CoreTimer	1.1.101
7	CoreUARTapb	5.2.2
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 FIC_1 - MSS Master Subsystem
drag and drop here to add to subsystem

MSS FIC_1 - 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~~

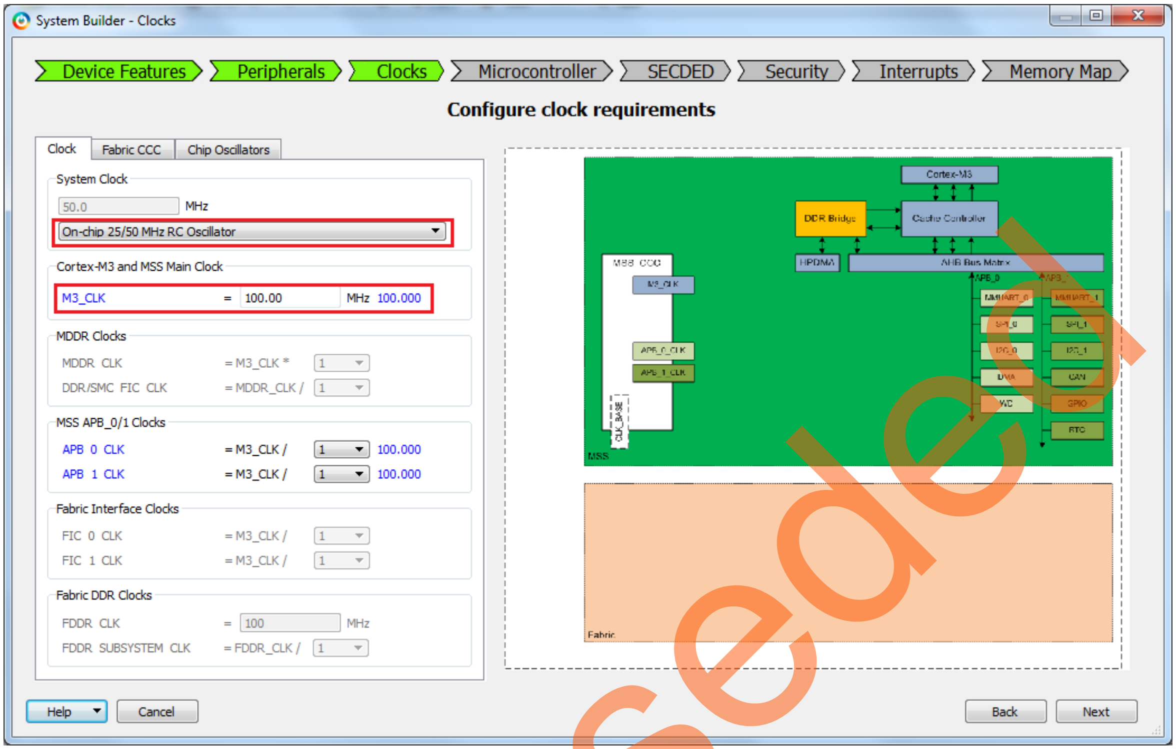
To move a peripheral from one subsystem to another, drag it from its present location and drop it onto the desired susbsystem.
You cannot drag and drop onto MSS Peripherals.

Masters are in bold and blue.

***Figure 8* • System Builder – Peripherals Page**

12. Click **Next**. This opens **System Builder - Clocks** page as shown in Figure 9.

13. In the **System Builder - Clocks** page (see Figure 9):
   – 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**



System Builder - Clocks

Configure clock requirements

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
FIC 1 CLK = M3_CLK / 1

Fabric DDR Clocks
FDDR CLK = 100 MHz
FDDR SUBSYSTEM CLK = FDDR_CLK / 1

Figure 9 • **System Builder – Clocks Page**

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 10.

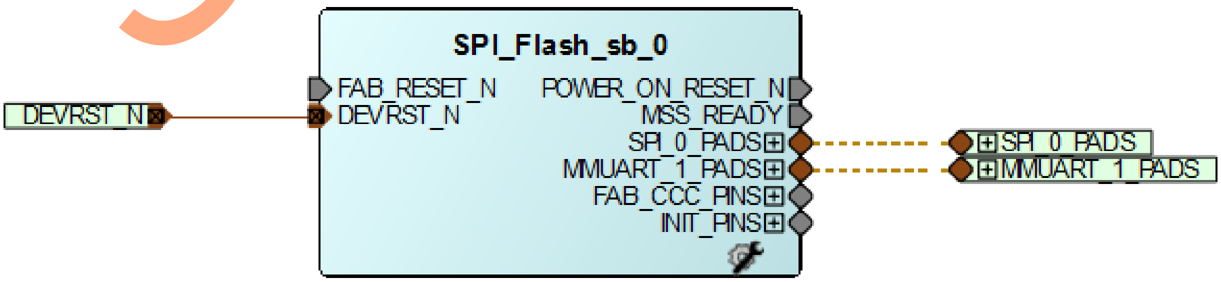


Figure 10 • **SPI_Flash SmartDesign**

Connecting Components in SPI_Flash_0 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 11.

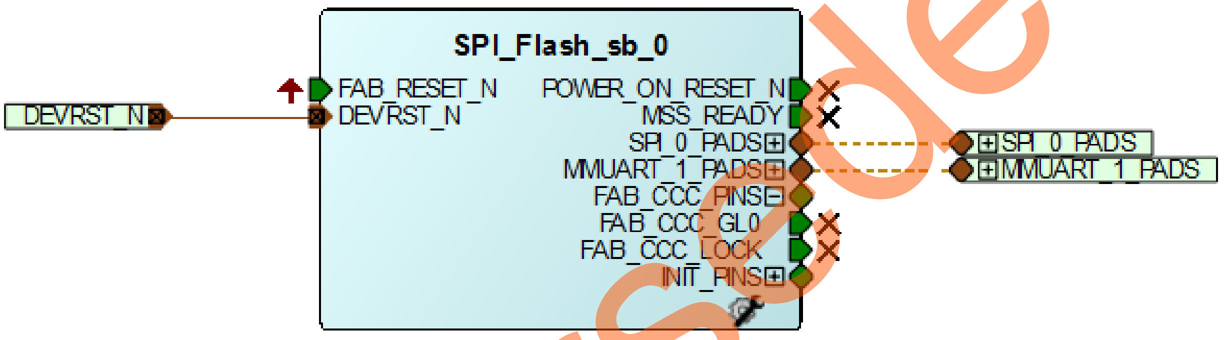


Figure 11 **• SPI_Flash SmartDesign**

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 12

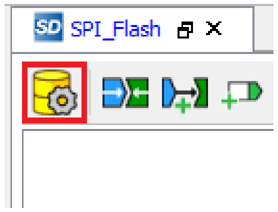
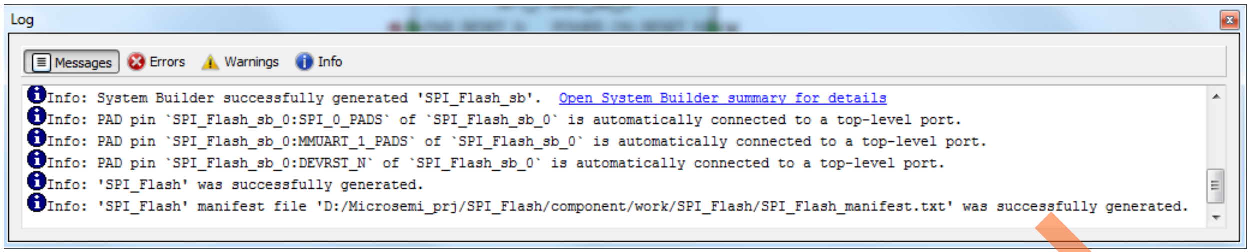


Figure 12 **• Generate Component**

After successful generation of all the components, the following message is displayed on the log window, as shown in Figure 13.

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



```
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.
```

Figure 13 **• Log Window**

Step 2: Generating the Program File

The following step describe how to generate the program file:

Click **Generate Bitstream** as shown in Figure 14 to generate the programming file.

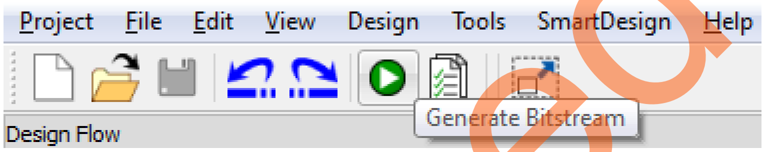


Figure 14 **• Generate Bitstream**

Step 3: Programming the 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 listed in Table 2. For more information on jumper locations, refer to the "Appendix C- SmartFusion2 Security Evaluation Kit Board Jumper Locations" on page 46.

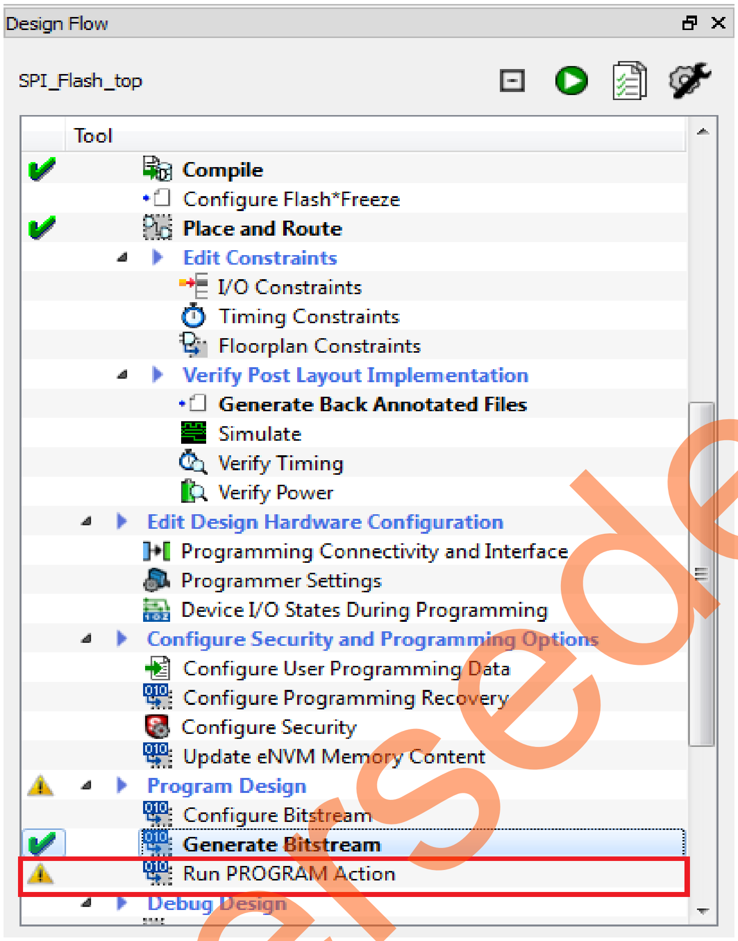
   CAUTION: Ensure that the power supply switch, **SW7** is switched OFF while connecting the jumpers on the SmartFusion 2 Security Evaluation Kit.

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.

   Switch **ON** the power supply switch, SW7. Refer to "Appendix B - Board Setup for Programming the Tutorial" on page 45 for information on the board setup for running the tutorial.
4. To program the SmartFusion2 device, double-click **Run PROGRAM Action** in the **Design Flow** tab as shown in Figure 15.



Design Flow

SPI_Flash_top

Tool

- ✓ Compile
- Configure Flash*Freeze
- ✓ Place and Route
  - Edit Constraints
    - I/O Constraints
    - Timing Constraints
    - Floorplan Constraints
  - Verify Post Layout Implementation
    - Generate Back Annotated Files
    - Simulate
    - Verify Timing
    - Verify Power
- Edit Design Hardware Configuration
  - Programming Connectivity and Interface
  - Programmer Settings
  - Device I/O States During Programming
- Configure Security and Programming Options
  - Configure User Programming Data
  - Configure Programming Recovery
  - Configure Security
  - Update eNVM Memory Content
- ⚠ Program Design
  - Configure Bitstream
  - ✓ Generate Bitstream
  - ⚠ Run PROGRAM Action
- Debug Design

Figure 15 **• Run Program Action**

Step 4: Configuring and Generating Firmware

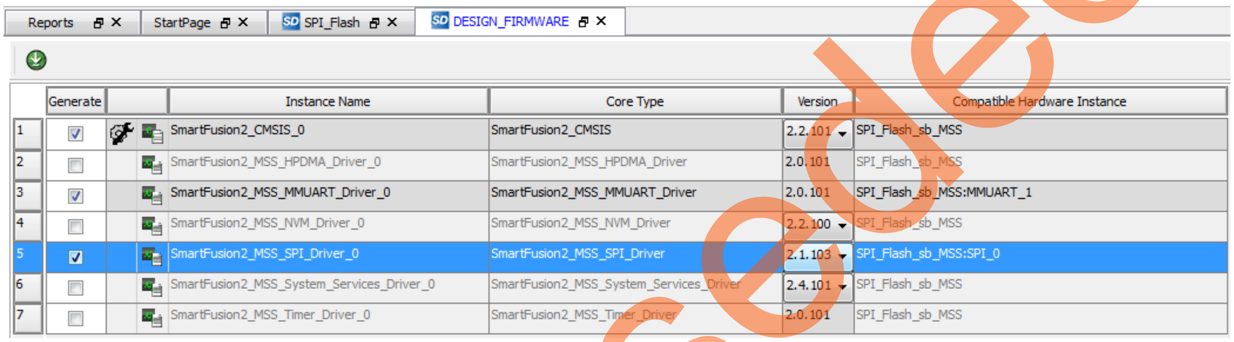
The Design Firmware window displays compatible firmware drivers based on peripherals configured in the design. Following drivers are used in this tutorial:

- CMSIS
- MMUART
- SPI

To generate the required drivers:

1. Double-click on **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 16.

Note: Select the latest version of the drivers.

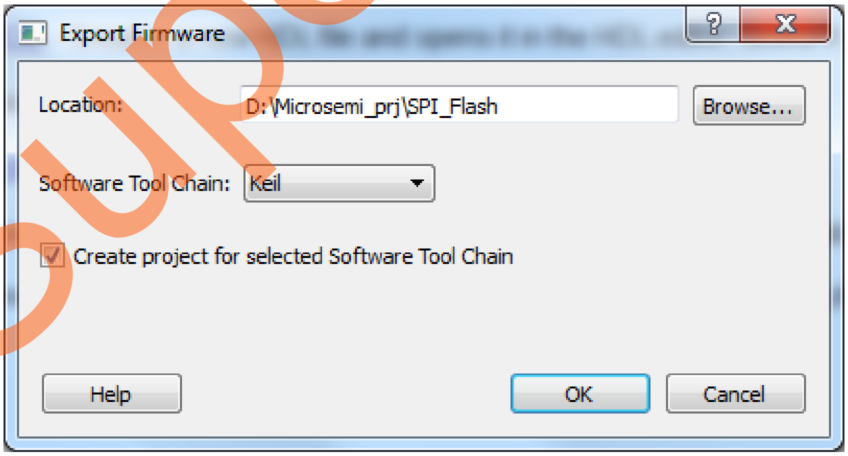


	Generate	Instance Name	Core Type	Version	Compatible Hardware Instance
1	☑	SmartFusion2_CMSIS_0	SmartFusion2_CMSIS	2.2.101	SPI_Flash_sb_MSS
2	□	SmartFusion2_MSS_HPDMA_Driver_0	SmartFusion2_MSS_HPDMA_Driver	2.0.101	SPI_Flash_sb_MSS
3	☑	SmartFusion2_MSS_MMUART_Driver_0	SmartFusion2_MSS_MMUART_Driver	2.0.101	SPI_Flash_sb_MSS:MMUART_1
4	□	SmartFusion2_MSS_NVM_Driver_0	SmartFusion2_MSS_NVM_Driver	2.2.100	SPI_Flash_sb_MSS
5	☑	SmartFusion2_MSS_SPI_Driver_0	SmartFusion2_MSS_SPI_Driver	2.1.103	SPI_Flash_sb_MSS:SPI_0
6	□	SmartFusion2_MSS_System_Services_Driver_0	SmartFusion2_MSS_System_Services_Driver	2.4.101	SPI_Flash_sb_MSS
7	□	SmartFusion2_MSS_Timer_Driver_0	SmartFusion2_MSS_Timer_Driver	2.0.101	SPI_Flash_sb_MSS

Figure 16 • **Configuring Firmware**

3. Double-click on **Export Firmware** in **Handoff design for Firmware Development** in **Design Flow** window.

   Export Firmware dialog box is displayed as shown in Figure 17.



Location: D:\Microsemi_prj\SPI_Flash

Software Tool Chain: Keil

☑ Create project for selected Software Tool Chain

Figure 17 • **Export Firmware Dialog Box**

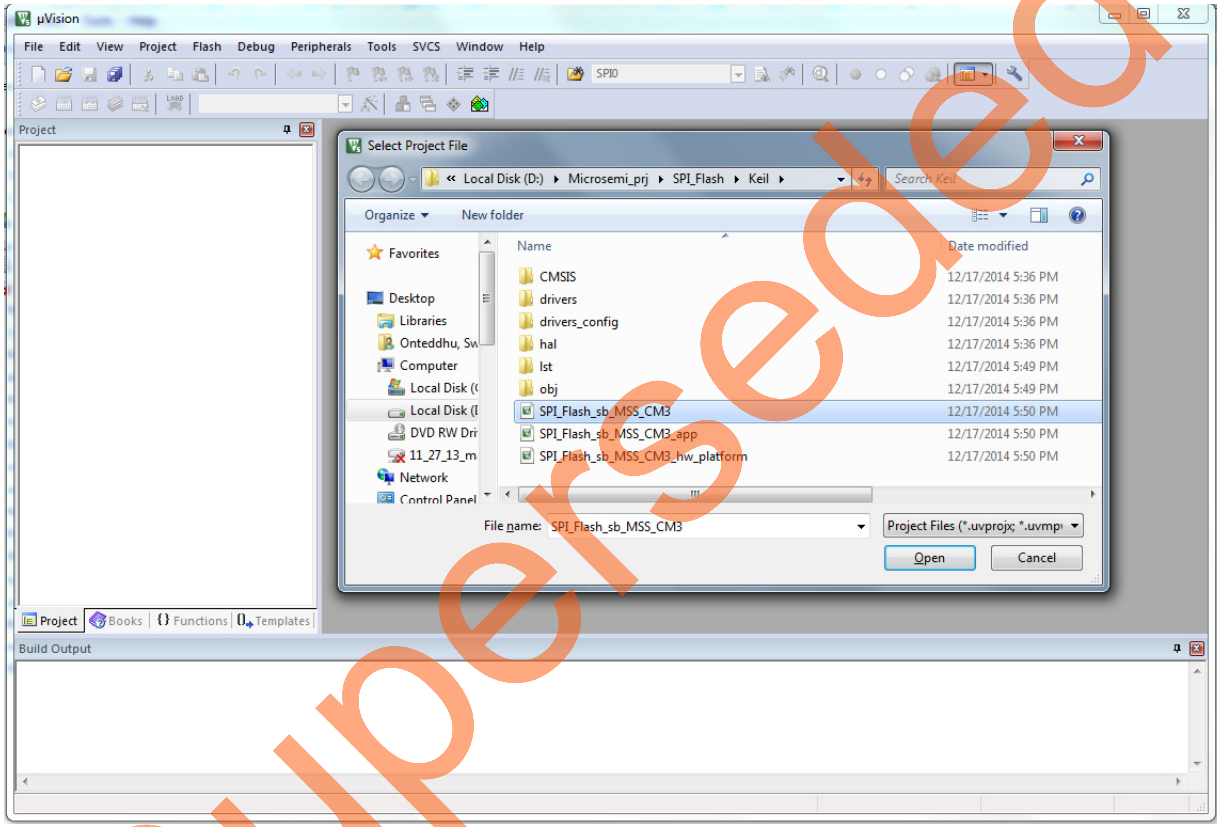
4. In the **Export Firmware** dialog box:
   – Select **Create project for selected Software Tool Chain**.
   – Select **Keil** from the drop down list.
5. Click **OK**. The successful firmware generation window is displayed.

The SmartFusion2 Security Evaluation Kit is ready for running and debugging the Keil application through ULINK-ME Debugger.

Step 5: Building the Software Application Using Keil uVision 5 IDE

The following steps describe how to build a software application using Keil uVision 5 IDE:

1. Launch the Keil IDE.Open the Keil project by double-clicking SPI_Flash_sb_MSS_CM3 Keil project as shown in Figure 18.



« Local Disk (D:) ▸ Microsemi_prj ▸ SPI_Flash ▸ Keil ▸

Search Keil

Name	Date modified
CMSIS	12/17/2014 5:36 PM
drivers	12/17/2014 5:36 PM
drivers_config	12/17/2014 5:36 PM
hal	12/17/2014 5:36 PM
lst	12/17/2014 5:49 PM
obj	12/17/2014 5:49 PM
SPI_Flash_sb_MSS_CM3	12/17/2014 5:50 PM
SPI_Flash_sb_MSS_CM3_app	12/17/2014 5:50 PM
SPI_Flash_sb_MSS_CM3_hw_platform	12/17/2014 5:50 PM

File name: SPI_Flash_sb_MSS_CM3

Project Files (*.uvprojx; *.uvmpı

Figure 18 **• Keil Homepage**

2. The Keil workspace is displayed, as shown in Figure 19.

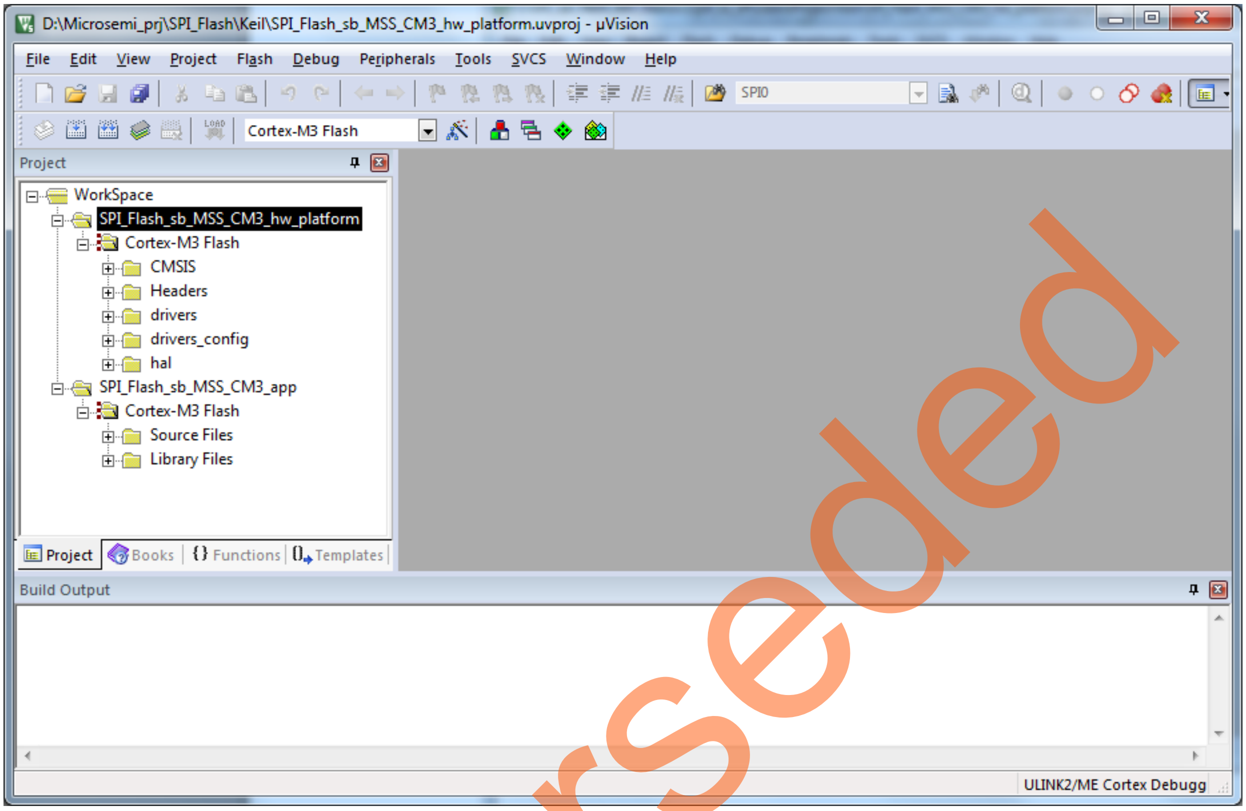
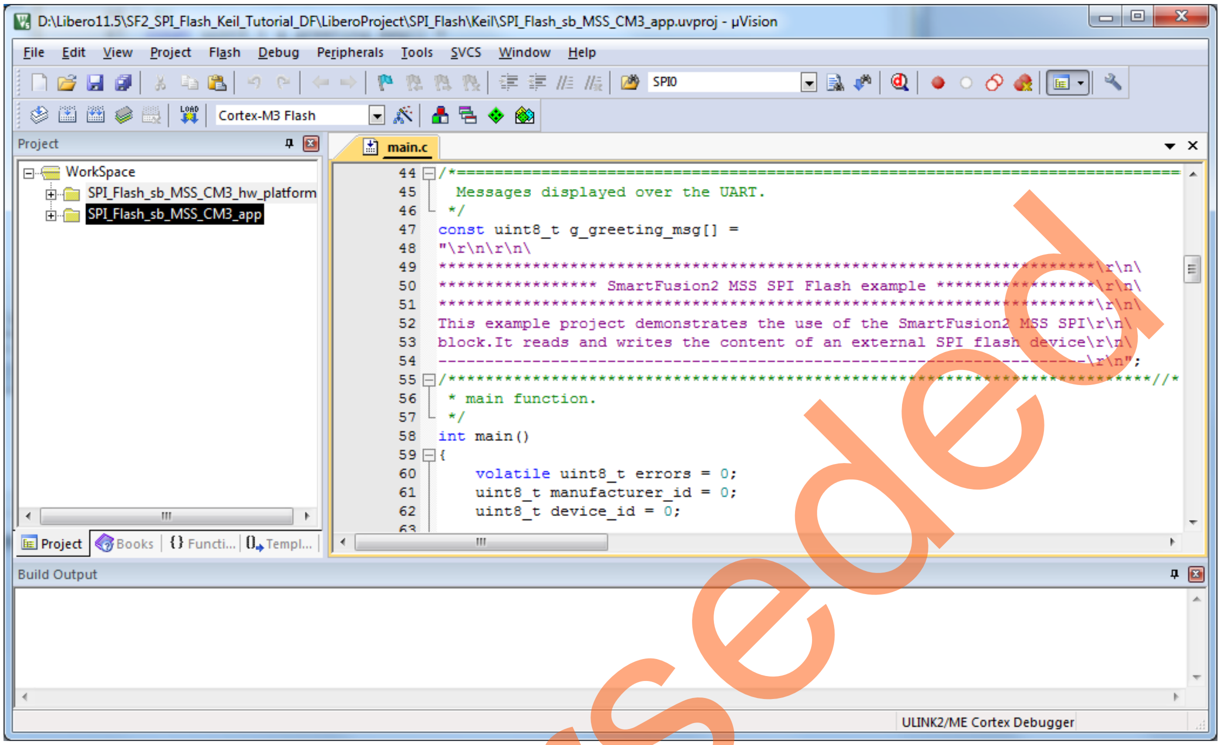


Figure 19 **• uVision Workspace**

3. Browse to the `main.c` file location in the design files folder: *<download_folder>*/SF2_SPI_Flash_Keil_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 uVision workspace.

The uVision window displays the `main.c` file, as shown in Figure 20.

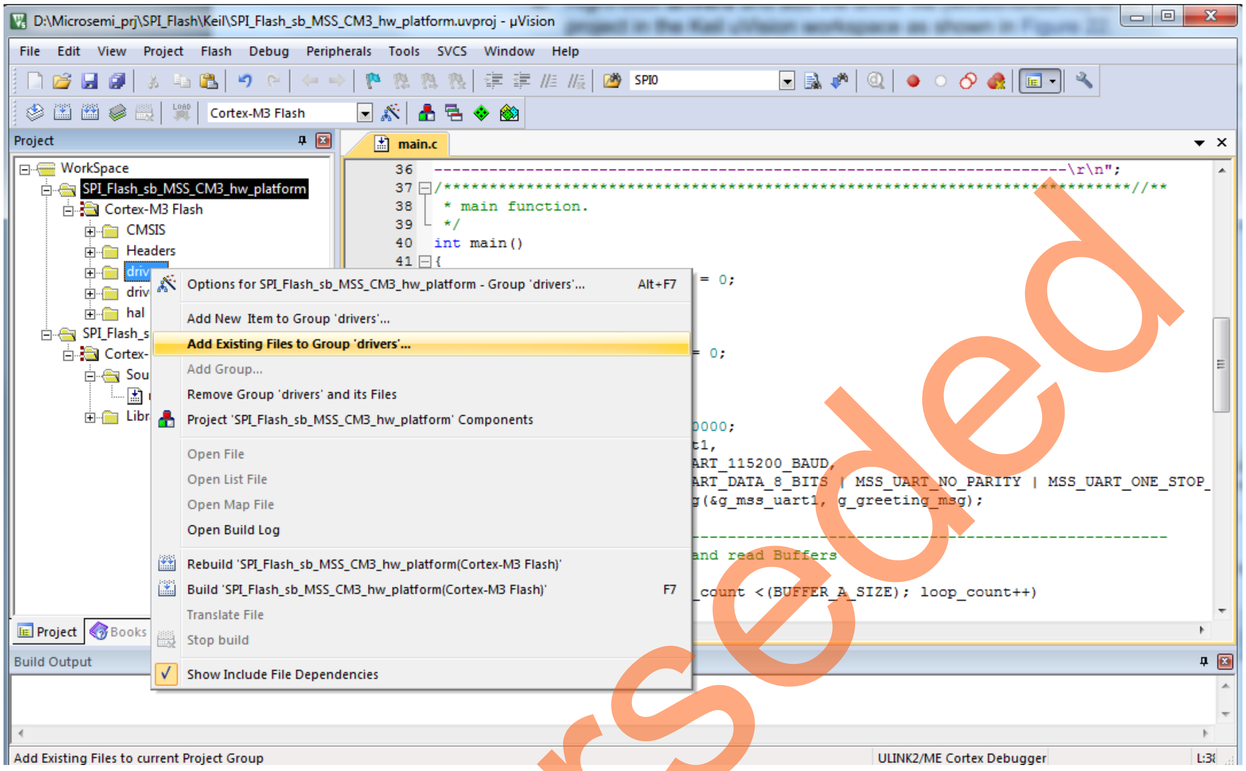

D:\Libero11.5\SF2_SPI_Flash_Keil_Tutorial_DF\LiberoProject\SPI_Flash\Keil\SPI_Flash_sb_MSS_CM3_app.uvproj - µVision
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
SPI0
Cortex-M3 Flash
Project
WorkSpace
SPI_Flash_sb_MSS_CM3_hw_platform
SPI_Flash_sb_MSS_CM3_app
main.c
Project
Books
{} Functi...
0→ Templ...
Build Output
ULINK2/ME Cortex Debugger

```
/*==============================================================================
  Messages displayed over the UART.
 */
const uint8_t g_greeting_msg[] =
"\r\n\r\n\
**********************************************************************\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;
```

Figure 20 **• uVision Workspace main.c file**

5. winbondflash drivers are not included in the Libero generated uVision workspace. To include the drivers in the uVision workspace, browse to the location of the winbondflash drivers in the design files folder:
   <download_folder>\SF2_SPI_Flash_Keil_Tutorial_DF*SPI_Flash_Drivers*.
6. Copy the **winbond flash** folder to the drivers folder of SPI_Flash_sb_MSS_CM3_hw_platform project in the uVision workspace.

7. Right-click and add the driver file (winbondflash.c) to SPI_Flash_sb_MSS_CM3_hw_platform project in the Keil uVision workspace as shown in Figure 21.


D:\Microsemi_prj\SPI_Flash\Keil\SPI_Flash_sb_MSS_CM3_hw_platform.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
SPI0
Cortex-M3 Flash
Project
WorkSpace
SPI_Flash_sb_MSS_CM3_hw_platform
Cortex-M3 Flash
CMSIS
Headers
driv
driv
hal
SPI_Flash_s
Cortex-
Sou
Libr
Project
Books
main.c
Options for SPI_Flash_sb_MSS_CM3_hw_platform - Group 'drivers'... Alt+F7
Add New Item to Group 'drivers'...
Add Existing Files to Group 'drivers'...
Add Group...
Remove Group 'drivers' and its Files
Project 'SPI_Flash_sb_MSS_CM3_hw_platform' Components
Open File
Open List File
Open Map File
Open Build Log
Rebuild 'SPI_Flash_sb_MSS_CM3_hw_platform(Cortex-M3 Flash)'
Build 'SPI_Flash_sb_MSS_CM3_hw_platform(Cortex-M3 Flash)' F7
Translate File
Stop build
Show Include File Dependencies
Build Output
Add Existing Files to current Project Group
ULINK2/ME Cortex Debugger
L:3

```
--------------------------------------------------------------------------\r\n";
/**********************************************************************************//**
 * main function.
 */
int main()
{
                                             = 0;

                                             = 0;

                                             0000;
                                             t1,
                                             ART_115200_BAUD,
                                             ART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_
                                             g(&g_mss_uart1, g_greeting_msg);

                                             --------------------------------------------------------
                                             and read Buffers

                                             _count <(BUFFER_A_SIZE); loop_count++)
```

Figure 21 • **uVision Workspace Window - Add winbondflash SPI Driver Files**

8. Change **SPI_Flash_sb_MSS_CM3_hw_platform** debug mode to **Cortex-M3_SRAM** by selecting **Cortex-M3_SRAM** from the drop-down list, as shown in Figure 22.

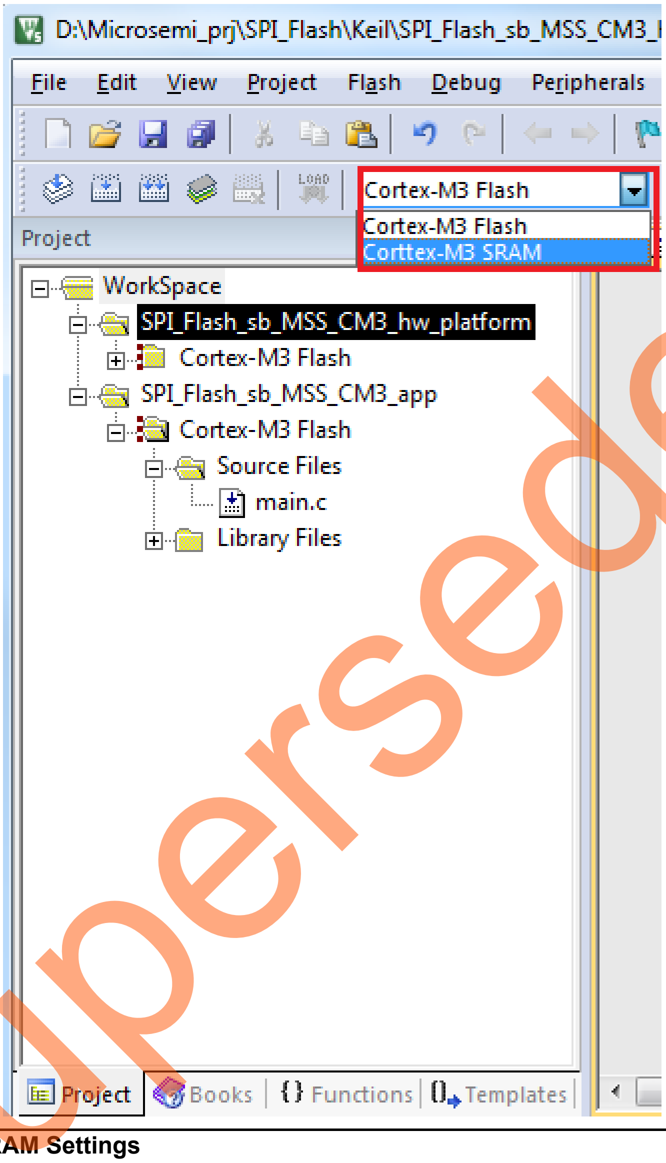


Figure 22 **• Cortex-M3_SRAM Settings**

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.

Follow the steps given below to add MICROSEMI_STDIO_THRU_UART symbol:

a. Right-click **Cortex - M3 SRAM** under **SPI_Flash_sb_MSS_CM3_hw_platform** and click **Options for SPI_Flash_sb_MSS_CM3_hw_platform - Target Cortex - M3 SRAM**.

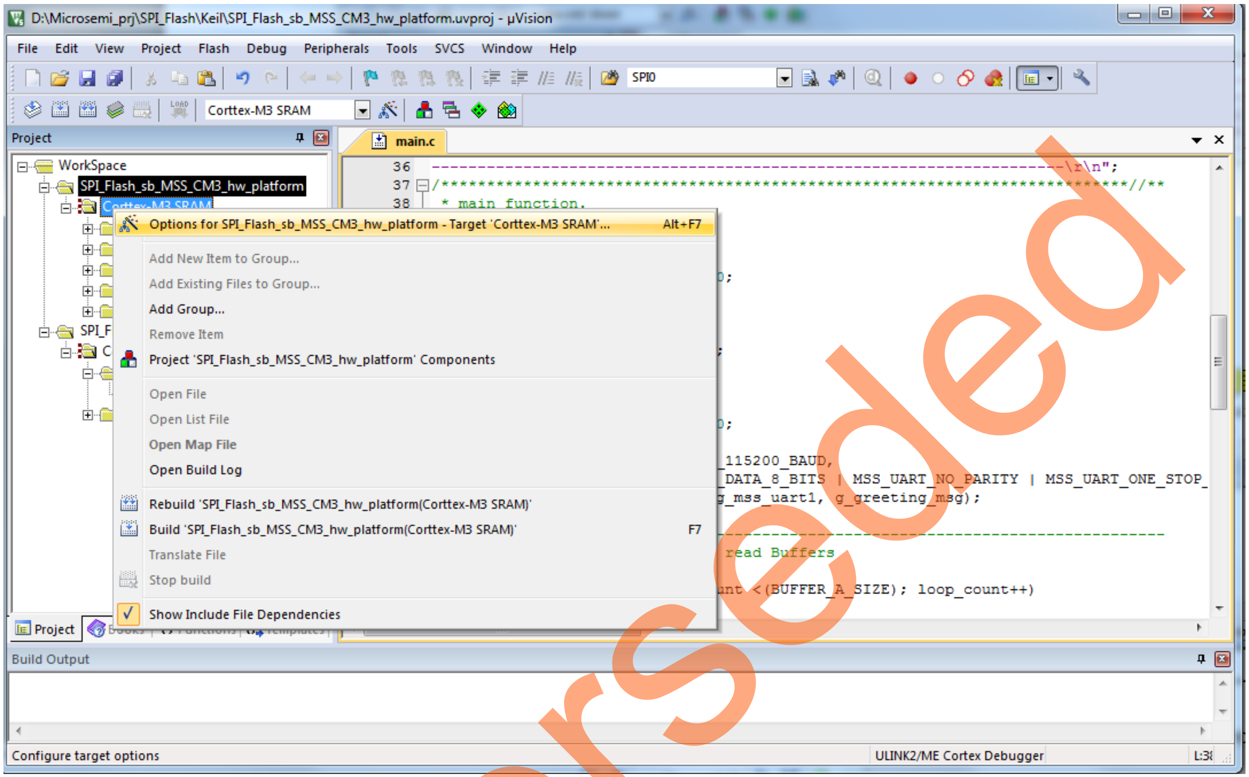
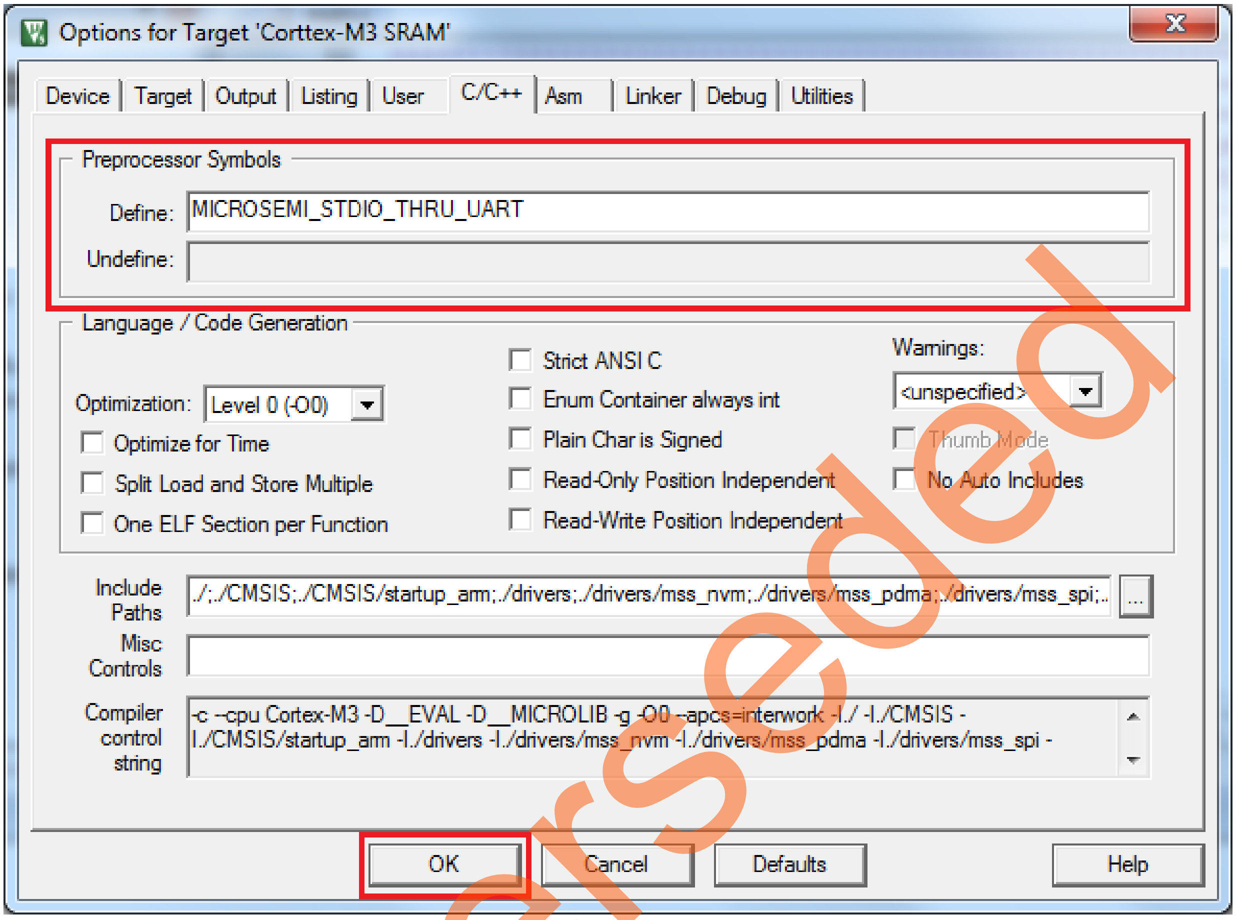

D:\Microsemi_prj\SPI_Flash\Keil\SPI_Flash_sb_MSS_CM3_hw_platform.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
SPI0
Cortex-M3 SRAM
Project
WorkSpace
SPI_Flash_sb_MSS_CM3_hw_platform
Cortex-M3 SRAM
main.c
Options for SPI_Flash_sb_MSS_CM3_hw_platform - Target 'Cortex-M3 SRAM'... Alt+F7
Add New Item to Group...
Add Existing Files to Group...
Add Group...
Remove Item
Project 'SPI_Flash_sb_MSS_CM3_hw_platform' Components
Open File
Open List File
Open Map File
Open Build Log
Rebuild 'SPI_Flash_sb_MSS_CM3_hw_platform(Cortex-M3 SRAM)'
Build 'SPI_Flash_sb_MSS_CM3_hw_platform(Cortex-M3 SRAM)' F7
Translate File
Stop build
Show Include File Dependencies
Project
Build Output
Configure target options
ULINK2/ME Cortex Debugger

Figure 23 **• Target Options**

b. Go to **C/C++** tab and enter **MICROSEMI_STDIO_THRU_UART** at **Define** under Preprocessor Symbols as shown in Figure 24 on page 23.

c. Click **OK**.



Define: MICROSEMI_STDIO_THRU_UART

Undefine:

Optimization: Level 0 (-O0)

□ Optimize for Time

□ Split Load and Store Multiple

□ One ELF Section per Function

□ Strict ANSI C

□ Enum Container always int

□ Plain Char is Signed

□ Read-Only Position Independent

□ Read-Write Position Independent

Warnings: <unspecified>

□ Thumb Mode

□ No Auto Includes

Include Paths: ./;./CMSIS;./CMSIS/startup_arm;./drivers;./drivers/mss_nvm;./drivers/mss_pdma;./drivers/mss_spi;..

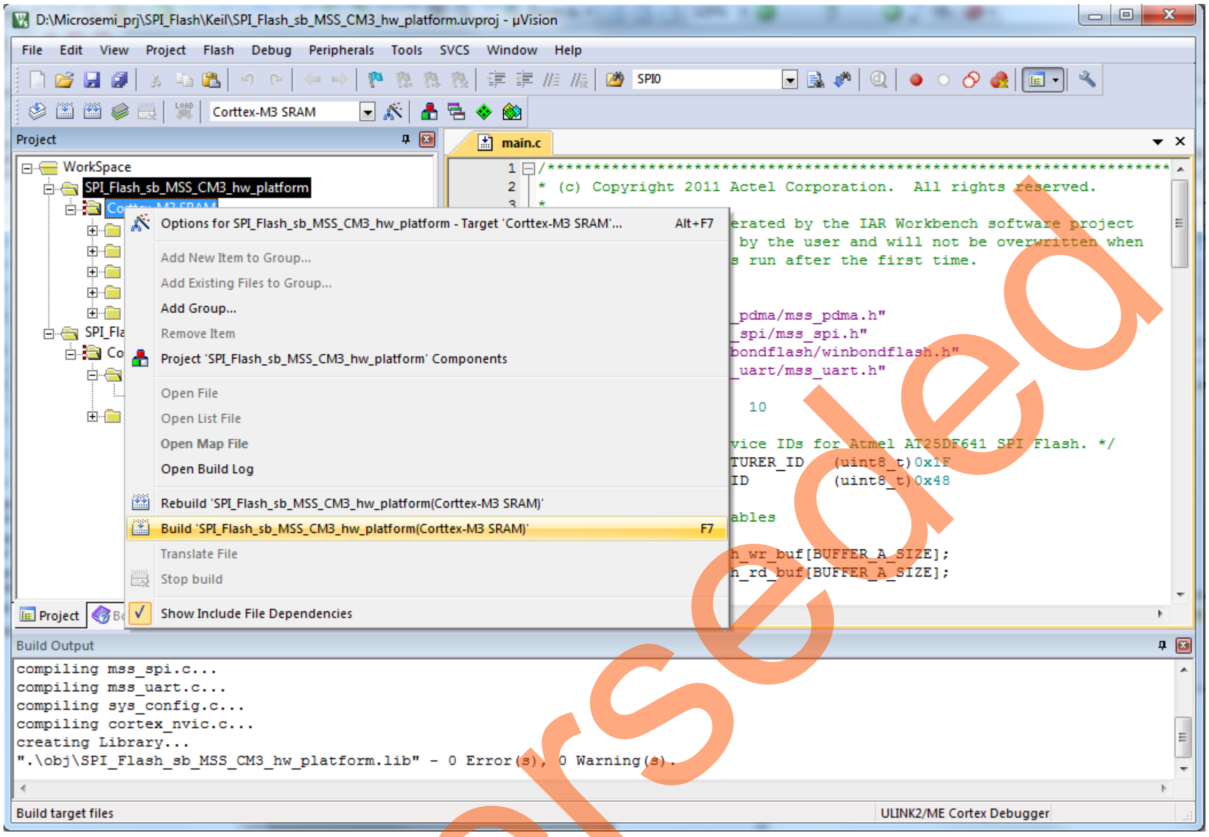
Misc Controls:

Compiler control string:

```
-c --cpu Cortex-M3 -D__EVAL -D__MICROLIB -g -O0 --apcs=interwork -I./ -I./CMSIS -
I./CMSIS/startup_arm -I./drivers -I./drivers/mss_nvm -I./drivers/mss_pdma -I./drivers/mss_spi -
```

Figure 24 **• Target Options-Adding Symbols**

9. Right-click **Cortex-CM3_SRAM** under SPI_Flash_sb_MSS_CM3_hw_platform and select **Build SPI_Flash_sb_MSS_CM3_hw_platform (Cortex-CM3 SRAM)** as shown in Figure 25.


D:\Microsemi_prj\SPI_Flash\Keil\SPI_Flash_sb_MSS_CM3_hw_platform.uvproj - µVision
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
SPI0
Cortex-M3 SRAM
Project
WorkSpace
SPI_Flash_sb_MSS_CM3_hw_platform
Cortex-M3 SRAM
SPI_Fla
Co
main.c
Options for SPI_Flash_sb_MSS_CM3_hw_platform - Target 'Corttex-M3 SRAM'... Alt+F7
Add New Item to Group...
Add Existing Files to Group...
Add Group...
Remove Item
Project 'SPI_Flash_sb_MSS_CM3_hw_platform' Components
Open File
Open List File
Open Map File
Open Build Log
Rebuild 'SPI_Flash_sb_MSS_CM3_hw_platform(Corttex-M3 SRAM)'
Build 'SPI_Flash_sb_MSS_CM3_hw_platform(Corttex-M3 SRAM)' F7
Translate File
Stop build
✓ Show Include File Dependencies
Project
Bo
Build Output
Build target files
ULINK2/ME Cortex Debugger

```
/*************************************************************************
 * (c) Copyright 2011 Actel Corporation.  All rights reserved.
 *
erated by the IAR Workbench software project
 by the user and will not be overwritten when
s run after the first time.

_pdma/mss_pdma.h"
_spi/mss_spi.h"
bondflash/winbondflash.h"
_uart/mss_uart.h"

 10

vice IDs for Atmel AT25DF641 SPI Flash. */
TURER_ID    (uint8_t)0x1F
ID          (uint8_t)0x48

ables

h_wr_buf[BUFFER_A_SIZE];
h_rd_buf[BUFFER_A_SIZE];
```

```
compiling mss_spi.c...
compiling mss_uart.c...
compiling sys_config.c...
compiling cortex_nvic.c...
creating Library...
".\obj\SPI_Flash_sb_MSS_CM3_hw_platform.lib" - 0 Error(s), 0 Warning(s).
```

Figure 25 **• Build HW Platform Window**

10. Right-click **SPI_Flash_sb_MSS_CM3_app** and select **Set as Active Project**.

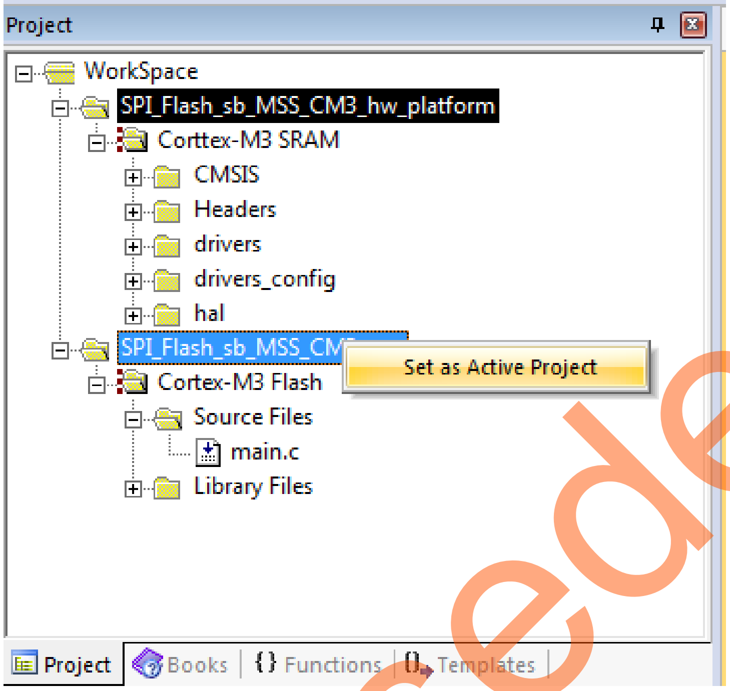


Figure 26 • **Set as Active Project**

11. Change **SPI_Flash_sb_MSS_CM3_app** debug mode to **Cortex-M3_SRAM** by selecting **Cortex-M3_SRAM** from the drop-down menu as shown in Figure 27.

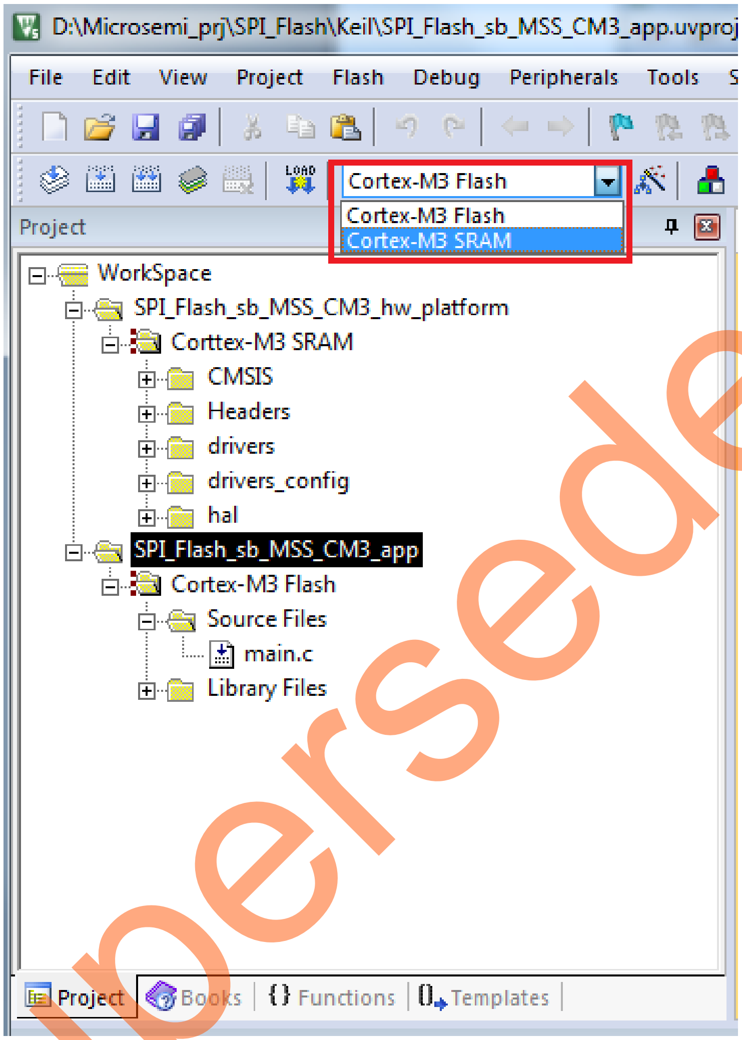


Figure 27 • **Cortex-M3_SRAM Settings**

12. Right-click **Cortex-M3 SRAM** under SPI_Flash_sb_MSS_CM3_app and click **Options for project**.

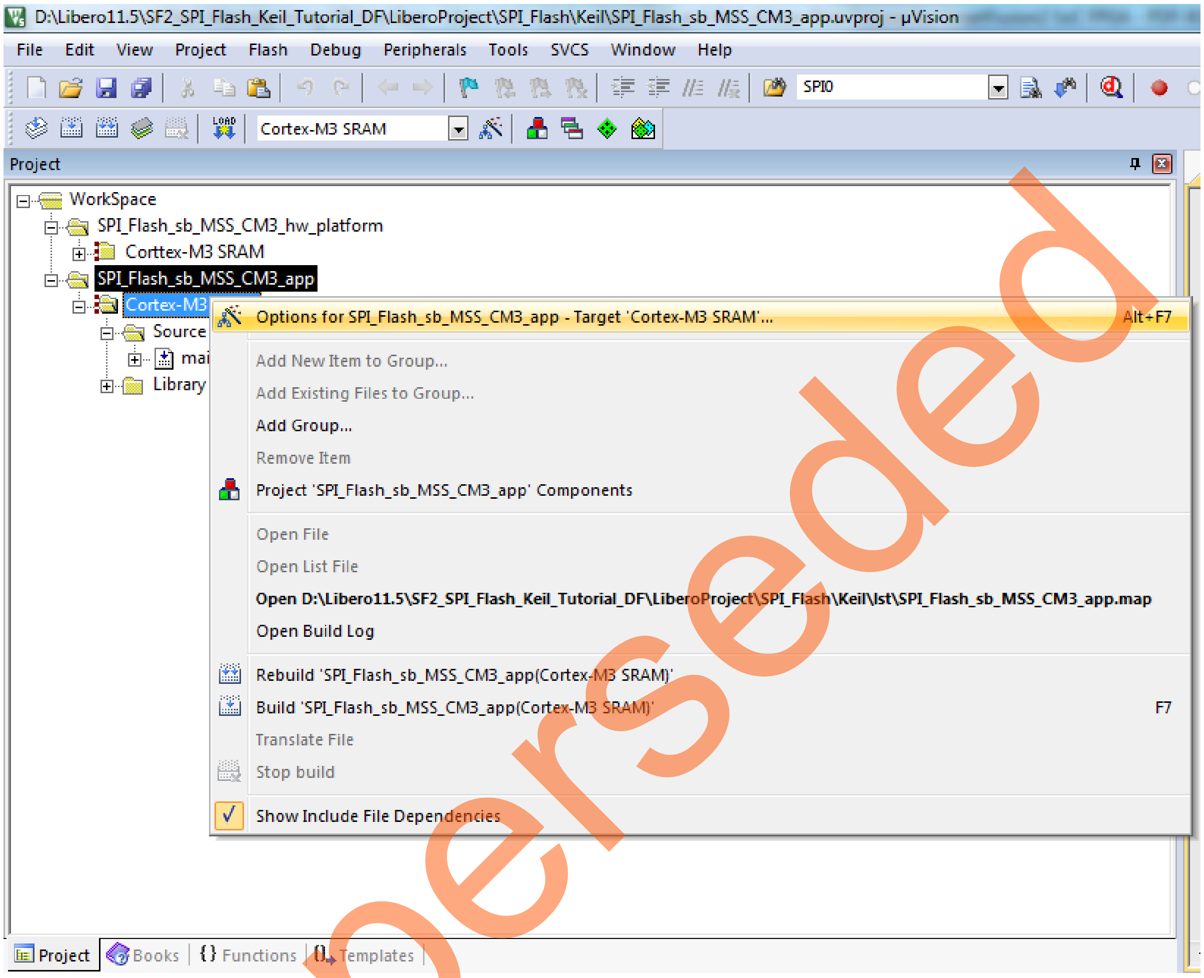
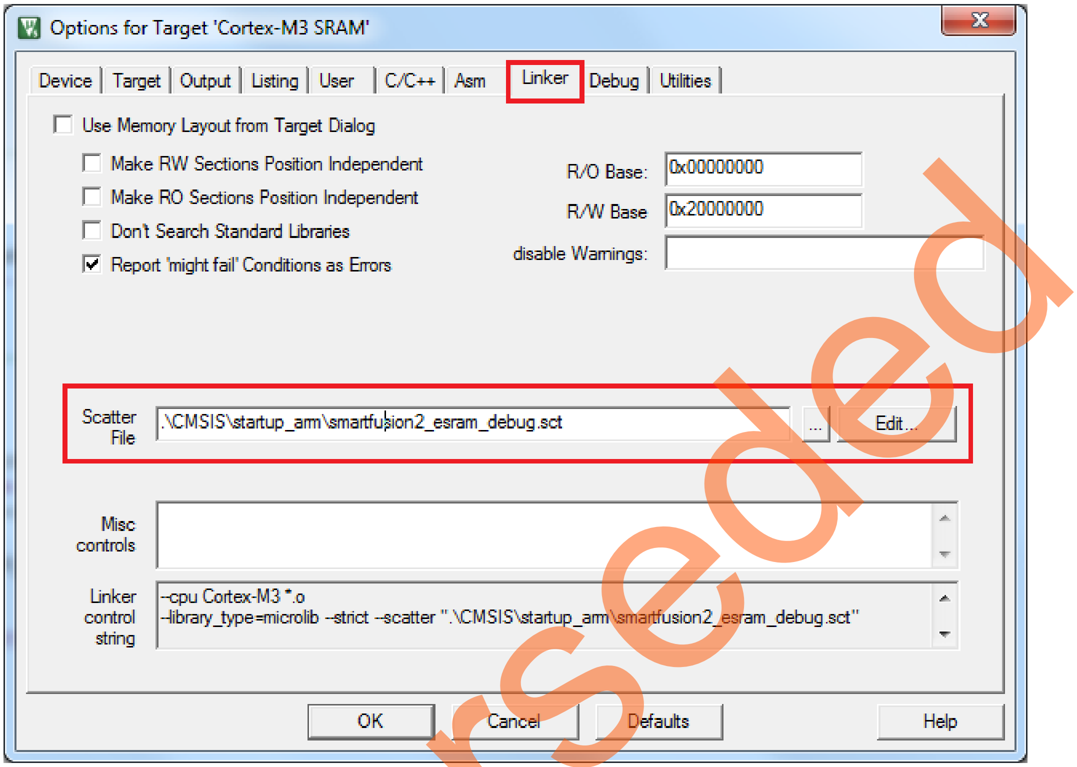

D:\Libero11.5\SF2_SPI_Flash_Keil_Tutorial_DF\LiberoProject\SPI_Flash\Keil\SPI_Flash_sb_MSS_CM3_app.uvproj - µVision
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
SPI0
Cortex-M3 SRAM
Project
WorkSpace
SPI_Flash_sb_MSS_CM3_hw_platform
Corttex-M3 SRAM
SPI_Flash_sb_MSS_CM3_app
Cortex-M3
Source
mai
Library
Options for SPI_Flash_sb_MSS_CM3_app - Target 'Cortex-M3 SRAM'...
Alt+F7
Add New Item to Group...
Add Existing Files to Group...
Add Group...
Remove Item
Project 'SPI_Flash_sb_MSS_CM3_app' Components
Open File
Open List File
Open D:\Libero11.5\SF2_SPI_Flash_Keil_Tutorial_DF\LiberoProject\SPI_Flash\Keil\lst\SPI_Flash_sb_MSS_CM3_app.map
Open Build Log
Rebuild 'SPI_Flash_sb_MSS_CM3_app(Cortex-M3 SRAM)'
Build 'SPI_Flash_sb_MSS_CM3_app(Cortex-M3 SRAM)'
F7
Translate File
Stop build
Show Include File Dependencies
Project
Books
{} Functions
Templates

Figure 28 • **Target Options**

13. Click the **Linker** tab and navigate to the SF2_SPI_Flash_Keil_Tutorial_DF\LiberoProject\Keil\CMSIS\startup_arm folder to select the **Scatter File** as `smartfusion2_esram_debug.sct`, as shown in Figure 29.



□ Use Memory Layout from Target Dialog

□ Make RW Sections Position Independent

□ Make RO Sections Position Independent

□ Don't Search Standard Libraries

☑ Report 'might fail' Conditions as Errors

R/O Base: 0x00000000

R/W Base 0x20000000

disable Warnings:

Scatter File: .\CMSIS\startup_arm\smartfusion2_esram_debug.sct

Misc controls:

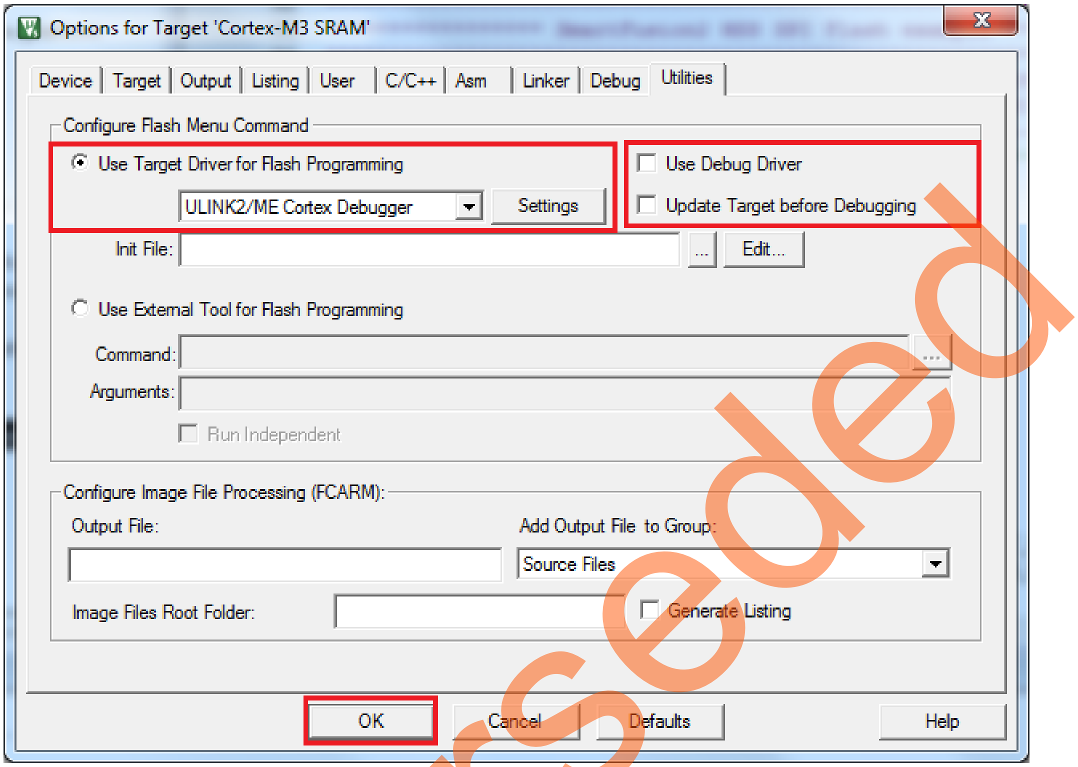
Linker control string:

```
--cpu Cortex-M3 *.o
--library_type=microlib --strict --scatter ".\CMSIS\startup_arm\smartfusion2_esram_debug.sct"
```

***Figure 29* • Target Options - Scatter File**

14. Click the **Utilities** tab and clear **Use Debug Driver** and **Update Target before Debugging** check boxes.

15. Select **ULINK2/ME Cortex Debugger** from the drop-down list and click **OK** as shown in Figure 30.



Configure Flash Menu Command

● Use Target Driver for Flash Programming

ULINK2/ME Cortex Debugger

□ Use Debug Driver

□ Update Target before Debugging

Init File:

○ Use External Tool for Flash Programming

Command:

Arguments:

□ Run Independent

Configure Image File Processing (FCARM):

Output File:

Add Output File to Group: Source Files

Image Files Root Folder:

□ Generate Listing

Figure 30 **• Target Options - Utilities Settings**

16. Right-click **Cortex-M3 SRAM** under **SPI_Flash_sb_MSS_CM3_app** and select **Build SPI_Flash_sb_MSS_CM3_app (Cortex-M3 SRAM)** as shown in Figure 31. It compiles all of the source files and links the object files into an AXF file to debug. Make sure that there are no errors. Correct any syntax errors and rebuild if necessary.

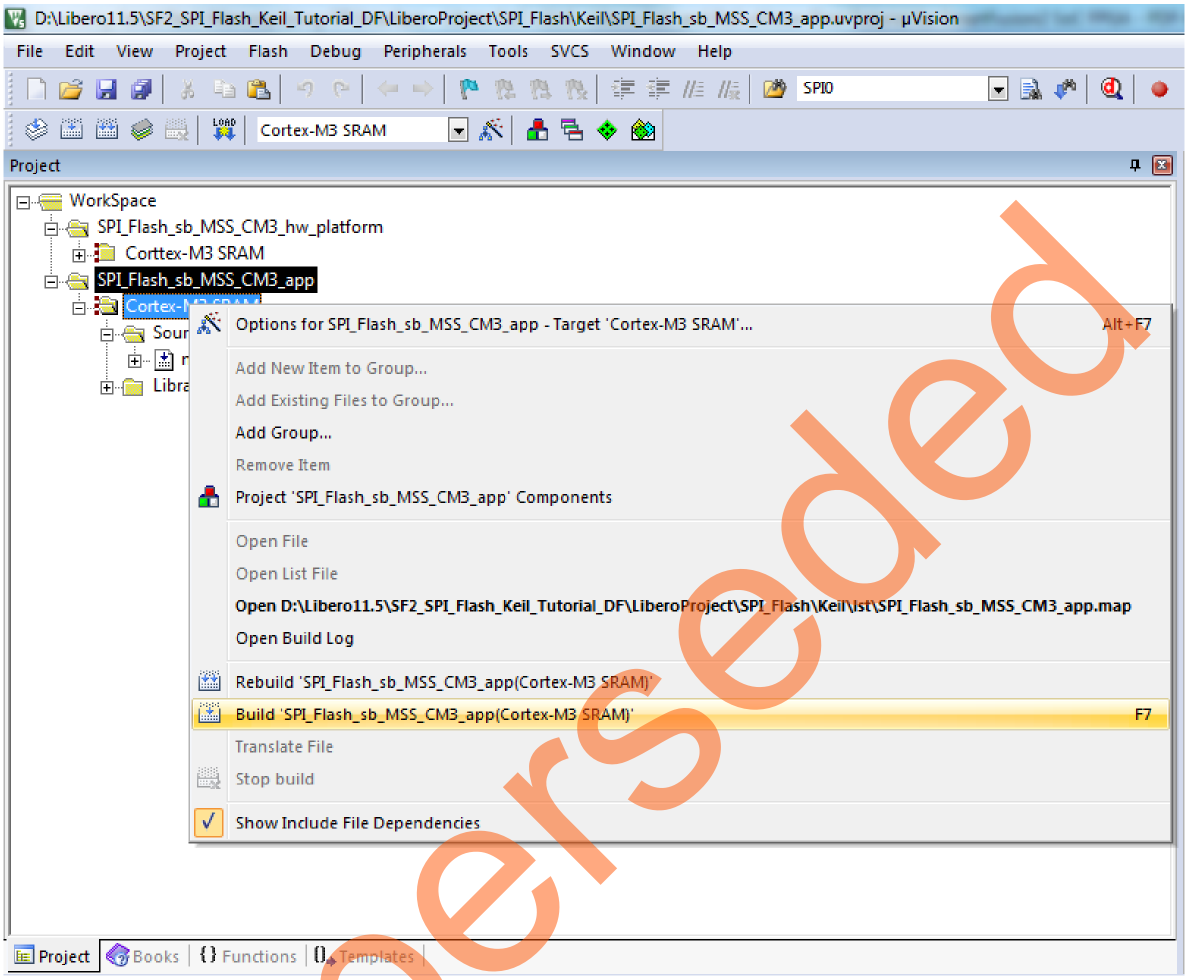
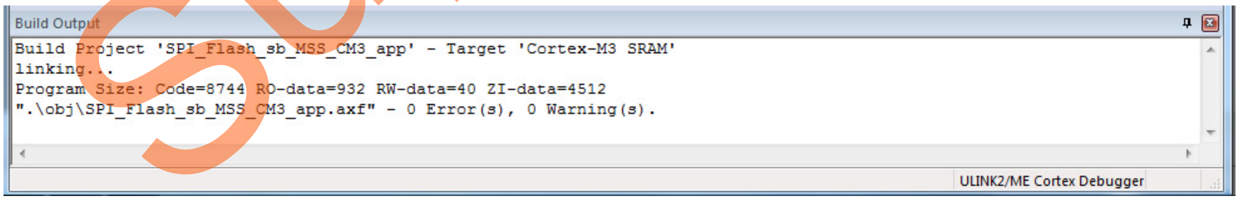

D:\Libero11.5\SF2_SPI_Flash_Keil_Tutorial_DF\LiberoProject\SPI_Flash\Keil\SPI_Flash_sb_MSS_CM3_app.uvproj - µVision
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
SPI0
Cortex-M3 SRAM
Project
WorkSpace
SPI_Flash_sb_MSS_CM3_hw_platform
Cortex-M3 SRAM
SPI_Flash_sb_MSS_CM3_app
Cortex-M3 SRAM
Sour
Libra
Options for SPI_Flash_sb_MSS_CM3_app - Target 'Cortex-M3 SRAM'... Alt+F7
Add New Item to Group...
Add Existing Files to Group...
Add Group...
Remove Item
Project 'SPI_Flash_sb_MSS_CM3_app' Components
Open File
Open List File
Open D:\Libero11.5\SF2_SPI_Flash_Keil_Tutorial_DF\LiberoProject\SPI_Flash\Keil\lst\SPI_Flash_sb_MSS_CM3_app.map
Open Build Log
Rebuild 'SPI_Flash_sb_MSS_CM3_app(Cortex-M3 SRAM)'
Build 'SPI_Flash_sb_MSS_CM3_app(Cortex-M3 SRAM)' F7
Translate File
Stop build
✓ Show Include File Dependencies
Project
Books
{} Functions
{}. Templates

Figure 31 **• Build Application Window**

Figure 32 shows the messages that are displayed in the console after the build.



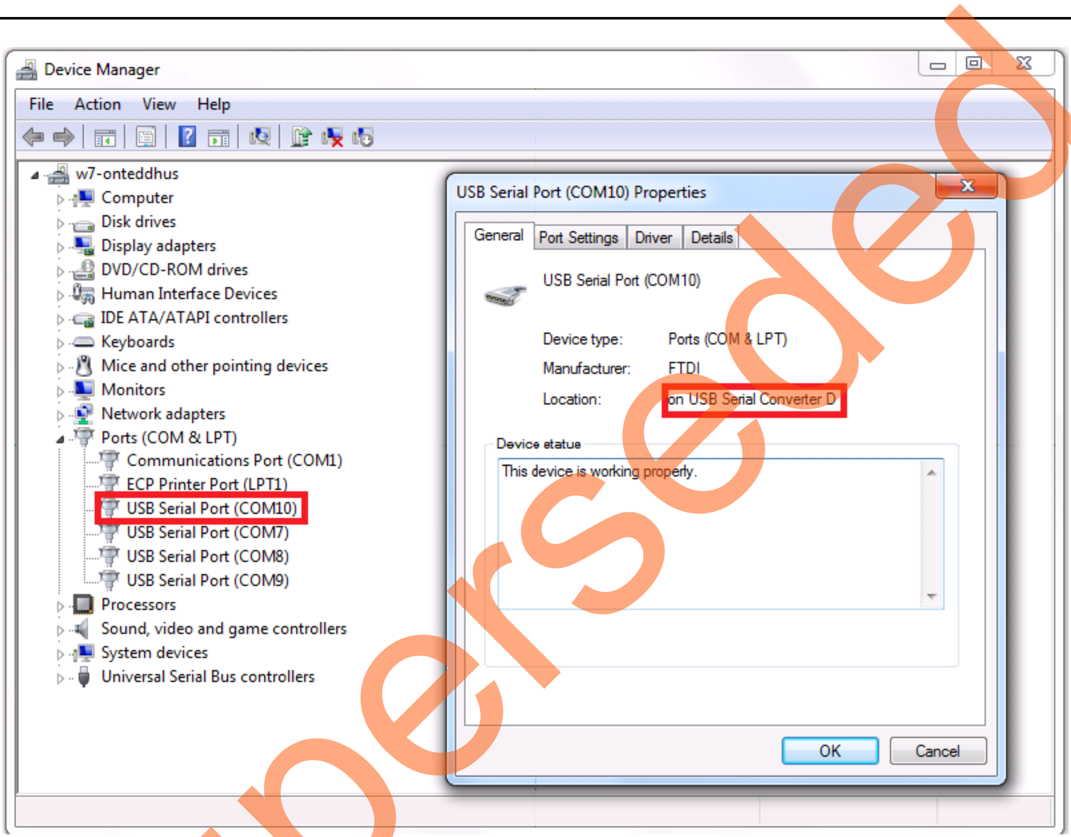
```
Build Project 'SPI_Flash_sb_MSS_CM3_app' - Target 'Cortex-M3 SRAM'
linking...
Program Size: Code=8744 RO-data=932 RW-data=40 ZI-data=4512
".\obj\SPI_Flash_sb_MSS_CM3_app.axf" - 0 Error(s), 0 Warning(s).
```

Figure 32 **• Build Output**

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. Of the four COM ports, select the one with Location as **on USB Serial Converter D**. Figure 33 shows an example Device Manager window.



w7-onteddhus
- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - Communications Port (COM1)
  - ECP Printer Port (LPT1)
  - USB Serial Port (COM10)
  - USB Serial Port (COM7)
  - USB Serial Port (COM8)
  - USB Serial Port (COM9)
- Processors
- Sound, video and game controllers
- System devices
- Universal Serial Bus controllers

USB Serial Port (COM10)

Device type: Ports (COM & LPT)

Manufacturer: FTDI

Location: on USB Serial Converter D

Device status

This device is working properly.

Figure 33 • Device Manager Window

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

Step 7: Connecting the ULINK-ME to the Board and PC

The following steps describe the connection between the SmartFusion2 Security Evaluation Kit board, ULINK-ME, and host PC. Use the appropriate settings for the board that is in use.

1. Connect Pin 2 and Pin 3 on the jumper J8 on the SmartFusion2 Security Evaluation Kit board.
2. Connect the USB A-Mini B cable between the host PC and the SmartFusion2 Security Evaluation Kit board. This is used to display the HyperTerminal communications.
3. Verify that the ULINK-ME debugger is connected to the SmartFusion2 Security Evaluation Kit board RVI Header as shown in Figure 34 and also to the host PC through a USB A-Mini B cable. The ULINK-ME adapter has one LED that indicates connection status in the following ways:
   – Blinking slowly indicates that ULINK-ME is ready to communicate with the debugger.
   – Blinking speedily indicates that the target board is executing the program under debugger control.
   – Remaining **ON** during debugging indicates that the debugger has halted the target board.
   – Remaining **ON** during download indicates that target download and verification is in progress.
4. Switch **ON** the SW7 power supply switch.

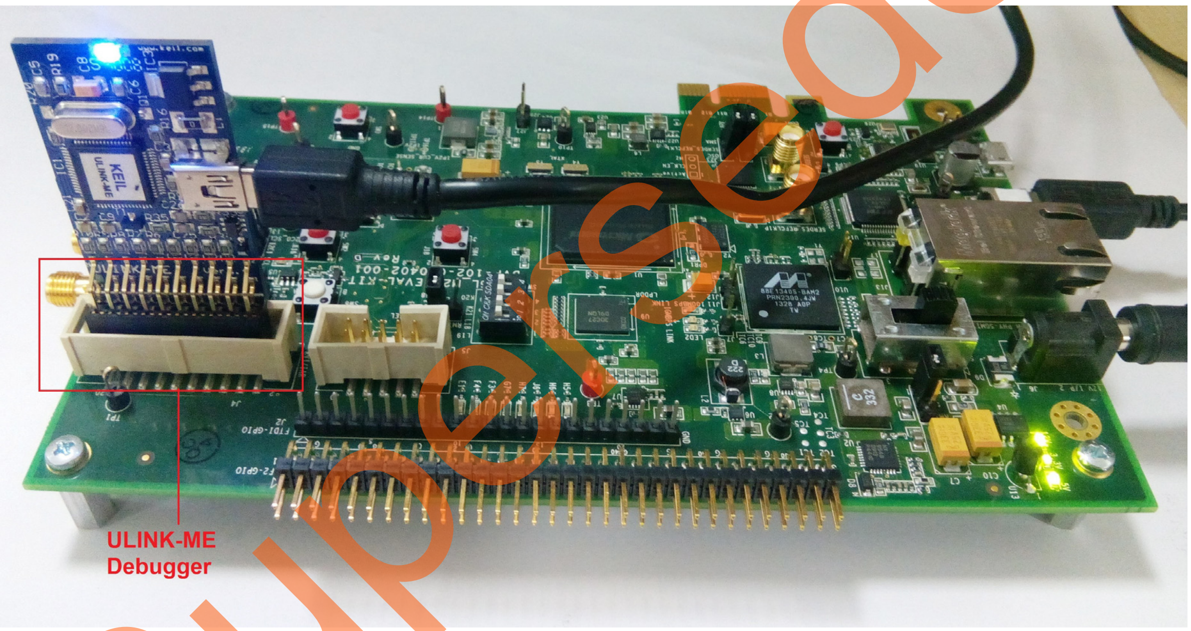


Figure 34 • **ULINK-ME Connections**

Refer to "Appendix A - Board Setup for Debugging from Keil uVision" on page 44 for information on the board setup for running the tutorial.

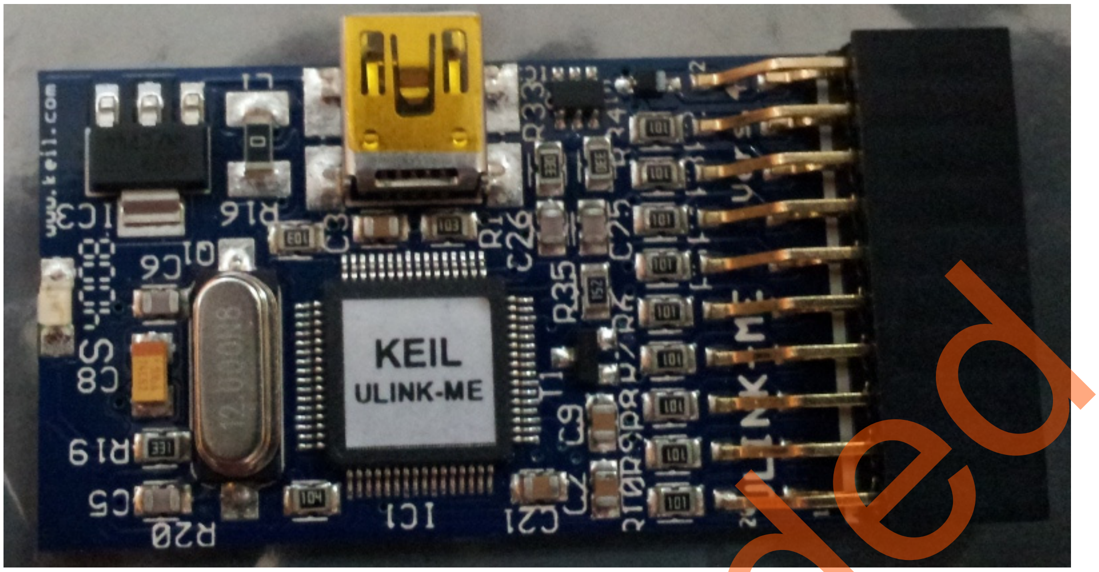
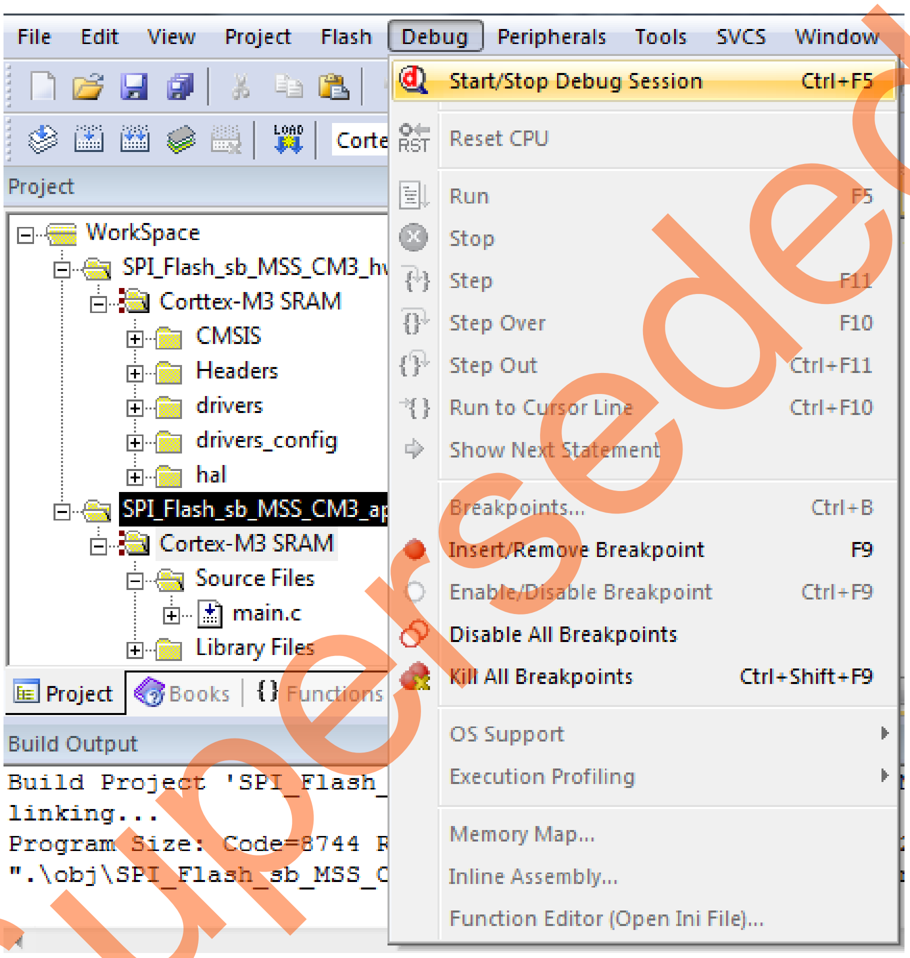


Figure 35 • **ULINK-ME Debugger**

Step 8: Debugging the Application Project using Keil uVision 5

The following steps describe how to debug the application project using Keil uVision:

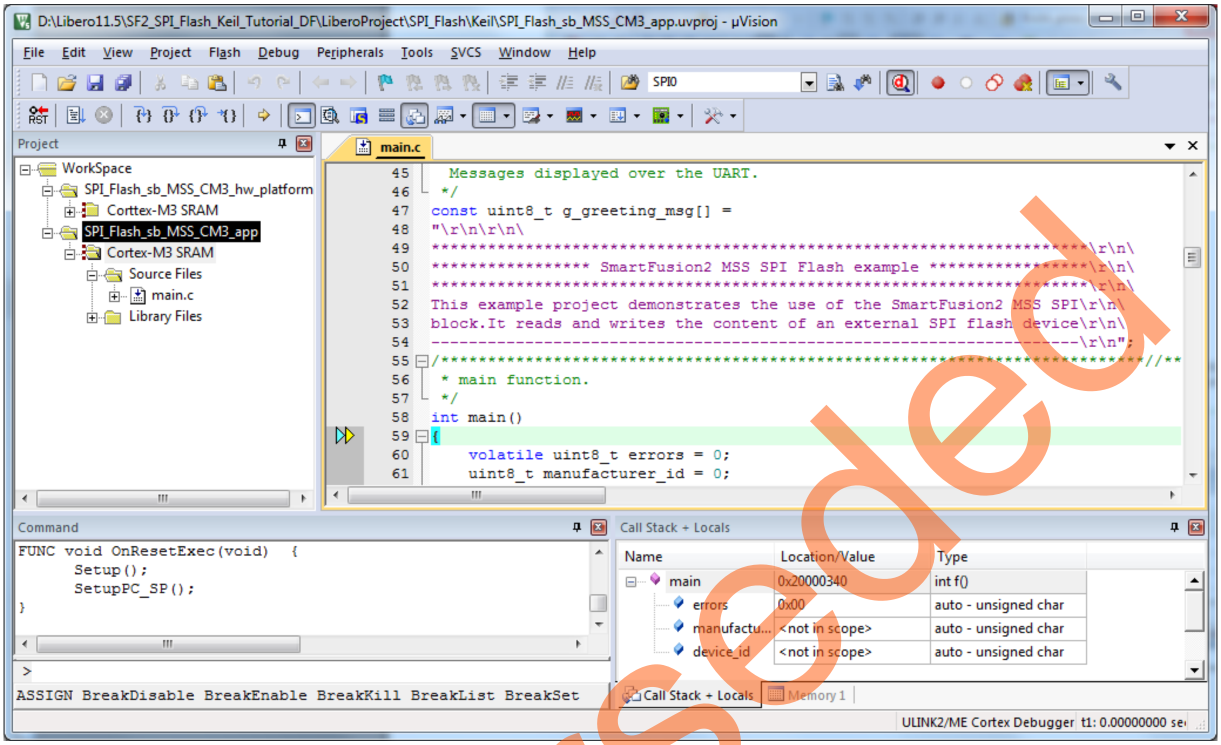
1. Select **Start/Stop Debug Session** from the **Debug** menu in the uVision window to run it through the debug hardware as shown in Figure 36. The processor code will be downloaded to the SmartFusion2 eSRAM.



```
Build Project 'SPI_Flash_
linking...
Program Size: Code=8744 R
".\obj\SPI_Flash_sb_MSS_C
```

Figure 36 **• Selecting Start/Stop Debug Session**

The code will automatically 'run to main' and then stop as shown in Figure 37.


D:\Libero11.5\SF2_SPI_Flash_Keil_Tutorial_DF\LiberoProject\SPI_Flash\Keil\SPI_Flash_sb_MSS_CM3_app.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
SPI0
Project
WorkSpace
SPI_Flash_sb_MSS_CM3_hw_platform
Cortex-M3 SRAM
SPI_Flash_sb_MSS_CM3_app
Cortex-M3 SRAM
Source Files
main.c
Library Files
main.c
Command
Call Stack + Locals
Memory 1

```
  Messages displayed over the UART.
 */
const uint8_t g_greeting_msg[] =
"\r\n\r\n\
*************************************************************************\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;
```

```
FUNC void OnResetExec(void)  {
       Setup();
       SetupPC_SP();
}
```

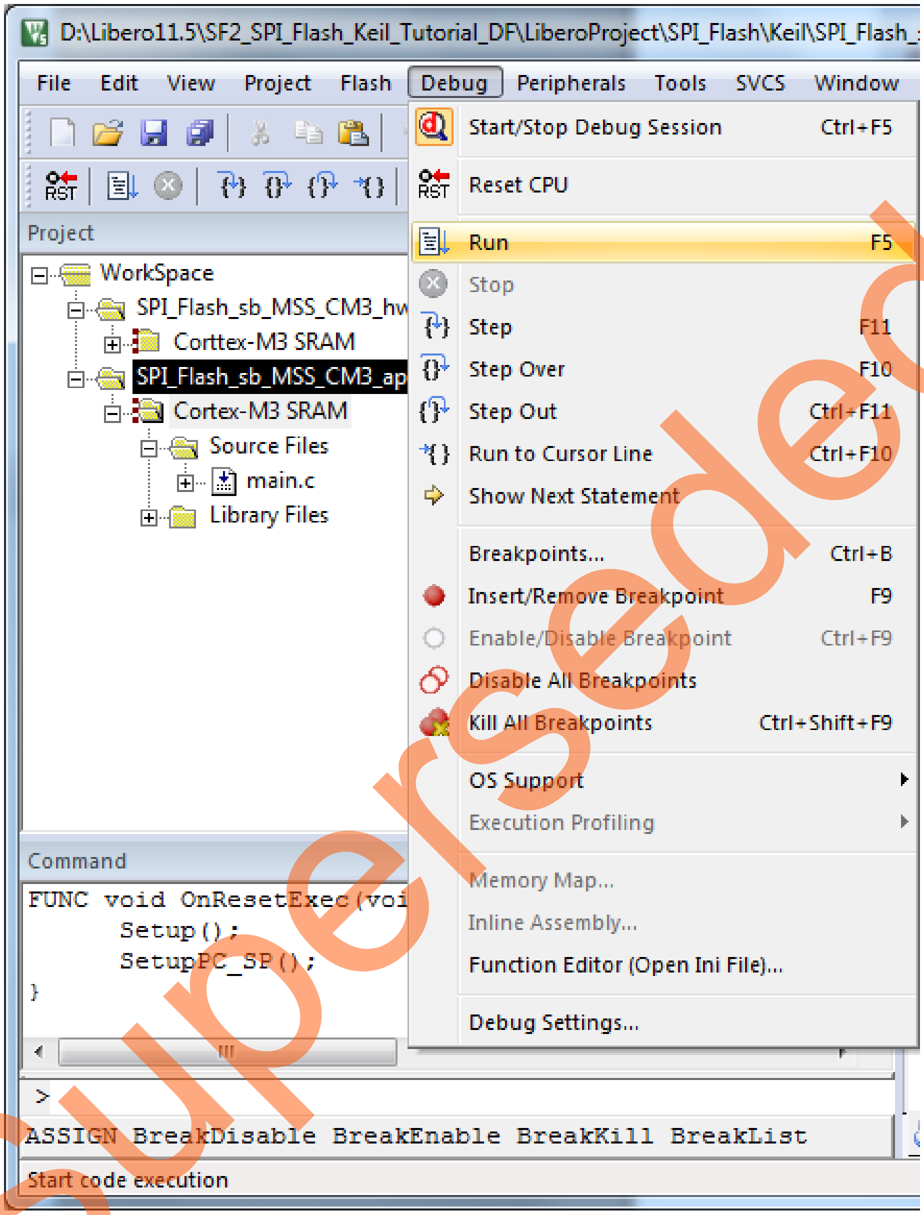
```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Name	Location/Value	Type
main	0x20000340	int f()
errors	0x00	auto - unsigned char
manufactu...	<not in scope>	auto - unsigned char
device_id	<not in scope>	auto - unsigned char

ULINK2/ME Cortex Debugger t1: 0.00000000 se

Figure 37 **• Debug Menu**

2. Click **Run** from the **Debug** menu as shown in Figure 38.

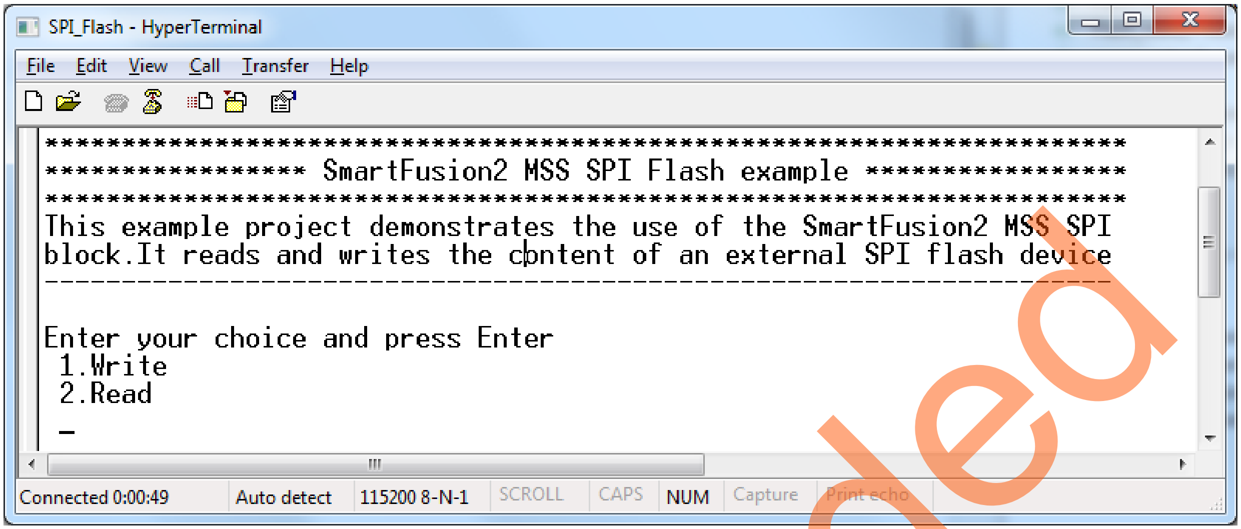


```
FUNC void OnResetExec(voi
      Setup();
      SetupPC_SP();
}
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Figure 38 • **Selecting Run from the Debug Menu**

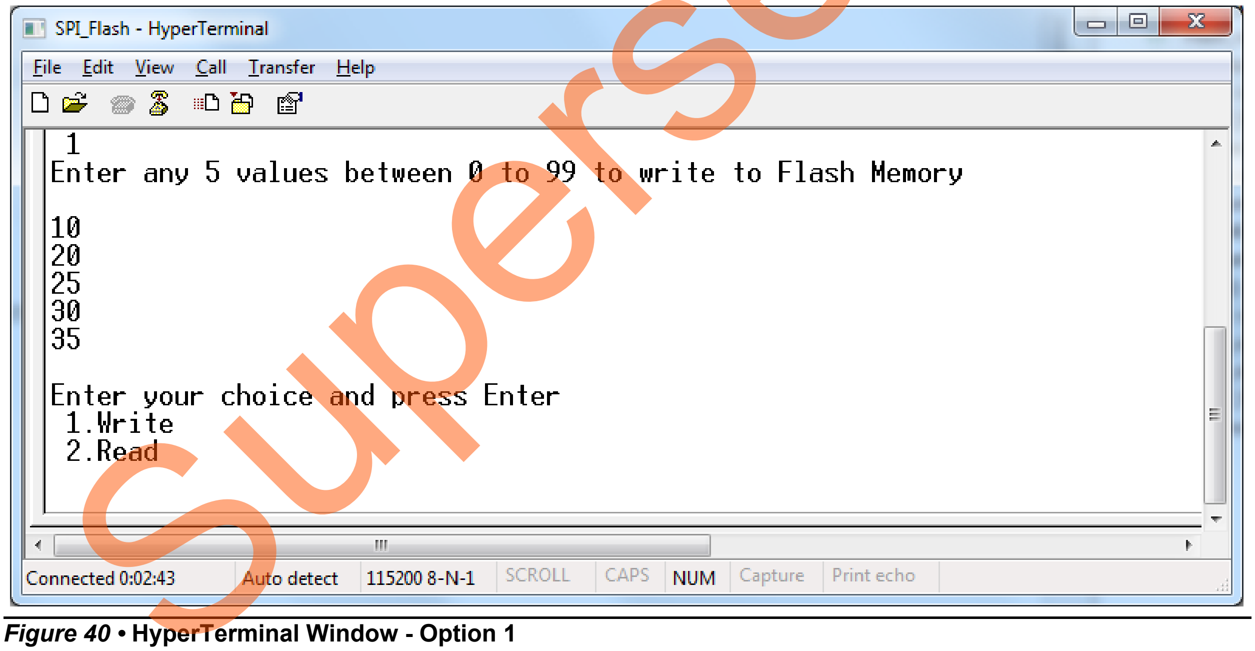
3. On successful operation, the HyperTerminal window displays a message as shown in Figure 39.



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

 _
```

Figure 39 **• HyperTerminal Window**

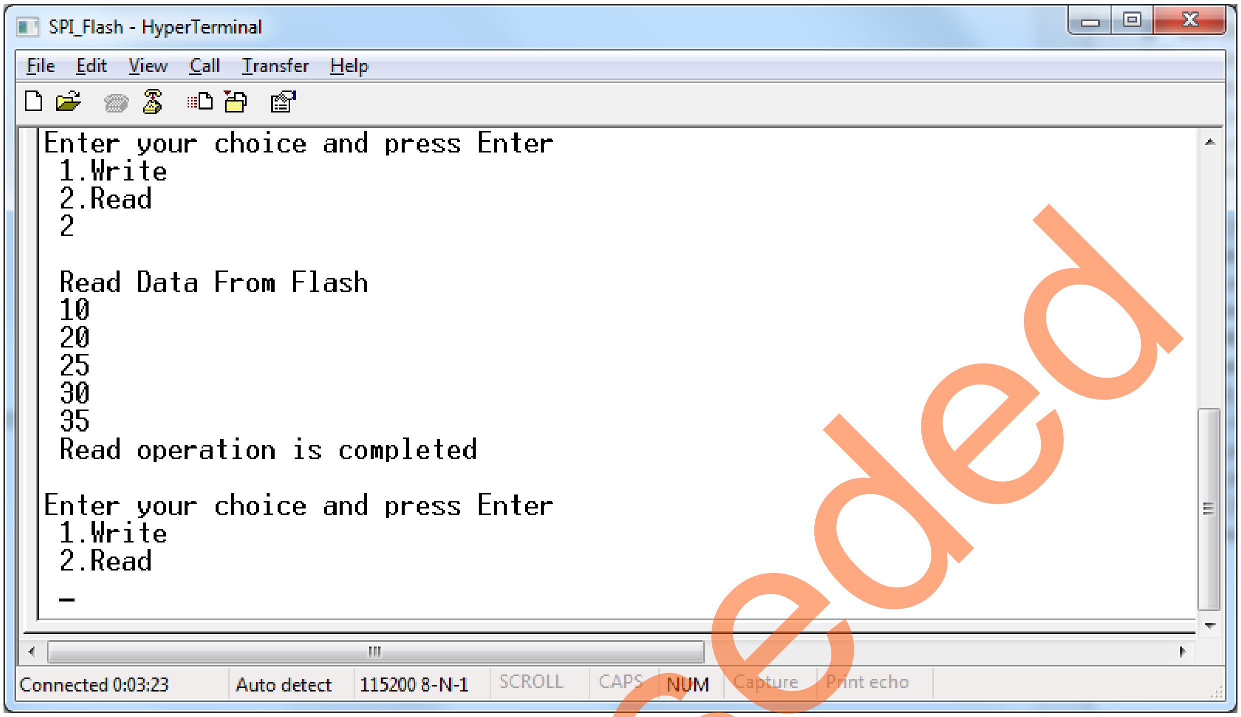
4. Select option 1 and enter values to write to the SPI Flash Memory as shown in Figure 40.



```
 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
```

Figure 40 **• HyperTerminal Window - Option 1**

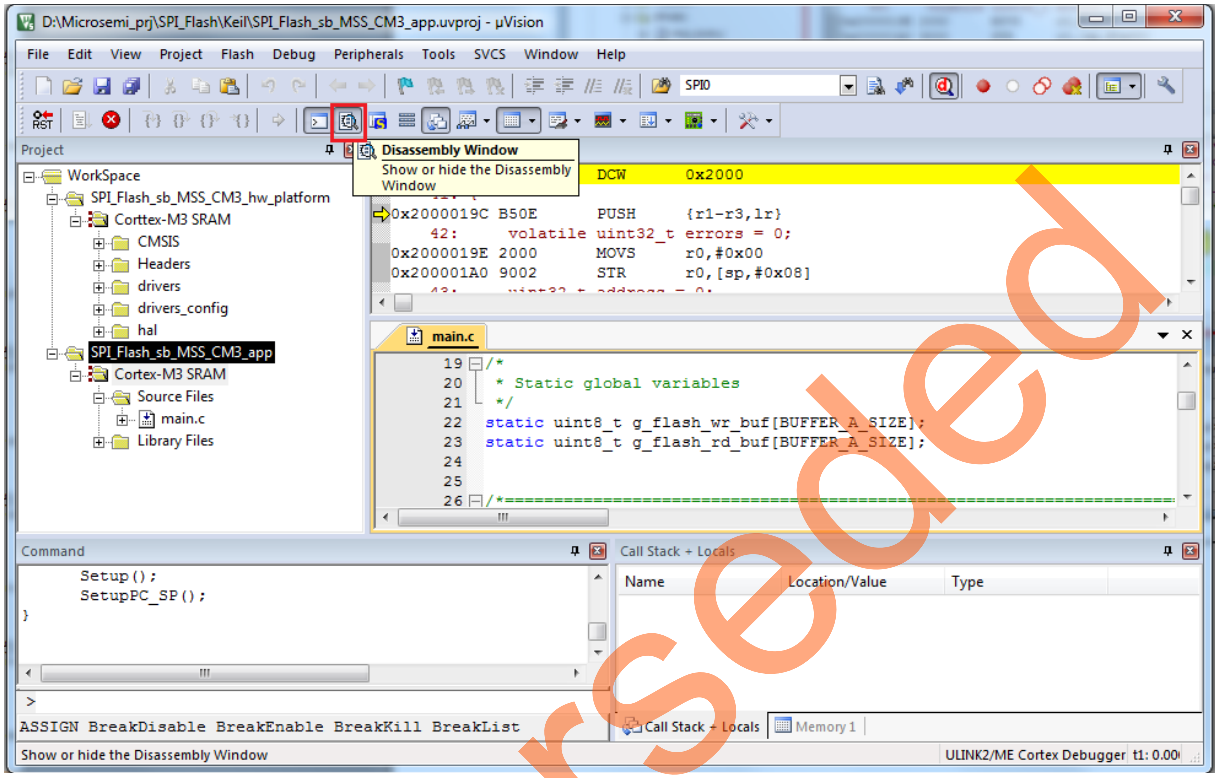
5. Select option 2 to read data from SPI Flash Memory as shown in Figure 41.



```
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
 _
```

Figure 41 • **HyperTerminal Window - Option 2**

The **Disassembly** window is displayed in the middle of the **Debug** section as shown in Figure 42. If not, click the **Disassembly** icon to display the **Disassembly** section.



```
                      DCW      0x2000
0x2000019C B50E       PUSH     {r1-r3,lr}
    42:       volatile uint32_t errors = 0;
0x2000019E 2000       MOVS     r0,#0x00
0x200001A0 9002       STR      r0,[sp,#0x08]
```

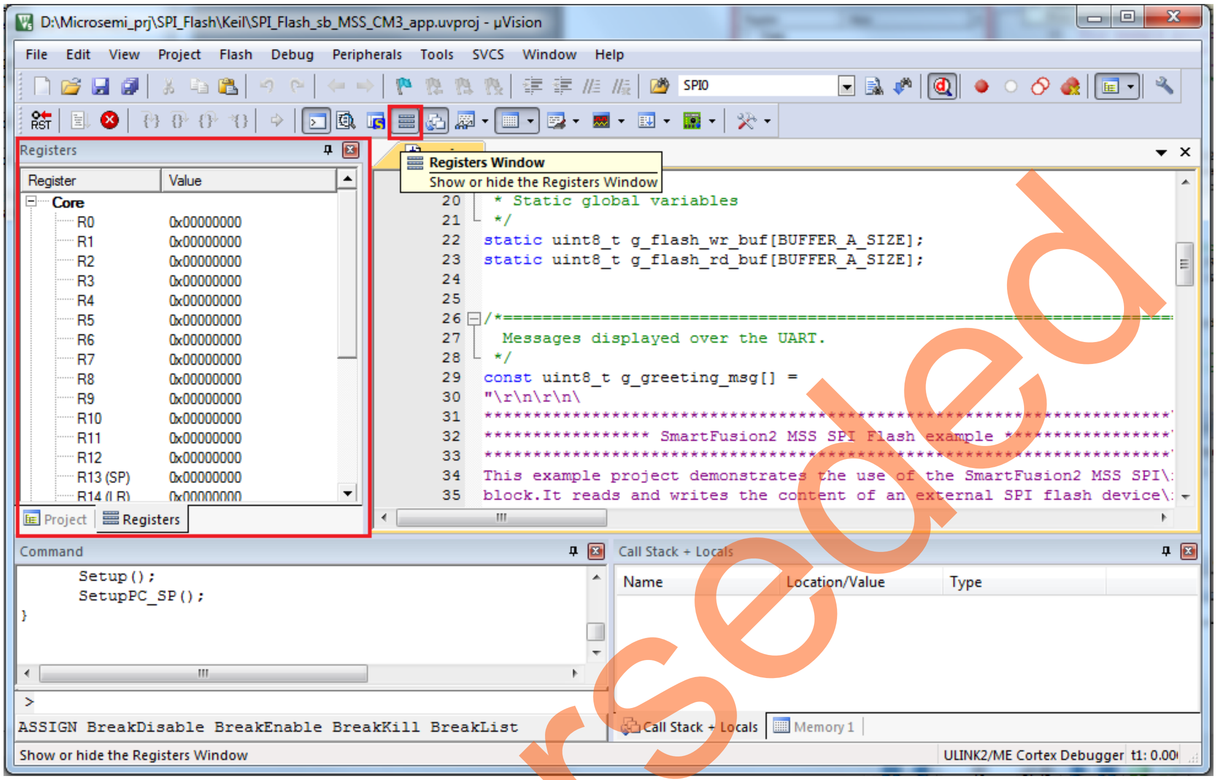
```
/*
 * Static global variables
 */
static uint8_t g_flash_wr_buf[BUFFER_A_SIZE];
static uint8_t g_flash_rd_buf[BUFFER_A_SIZE];


/*=====================================================================
```

```
      Setup();
      SetupPC_SP();
}
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Figure 42 **• Disassembly Window**

6. Click **Registers Window** to view the values of the ARM® Cortex®-M3 processor internal registers as shown in Figure 42.



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000

```
 * Static global variables
 */
static uint8_t g_flash_wr_buf[BUFFER_A_SIZE];
static uint8_t g_flash_rd_buf[BUFFER_A_SIZE];


/*==========================================================================
  Messages displayed over the UART.
 */
const uint8_t g_greeting_msg[] =
"\r\n\r\n\
*****************************************************************************
***************** 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\:
```

```
        Setup();
        SetupPC_SP();
}
```

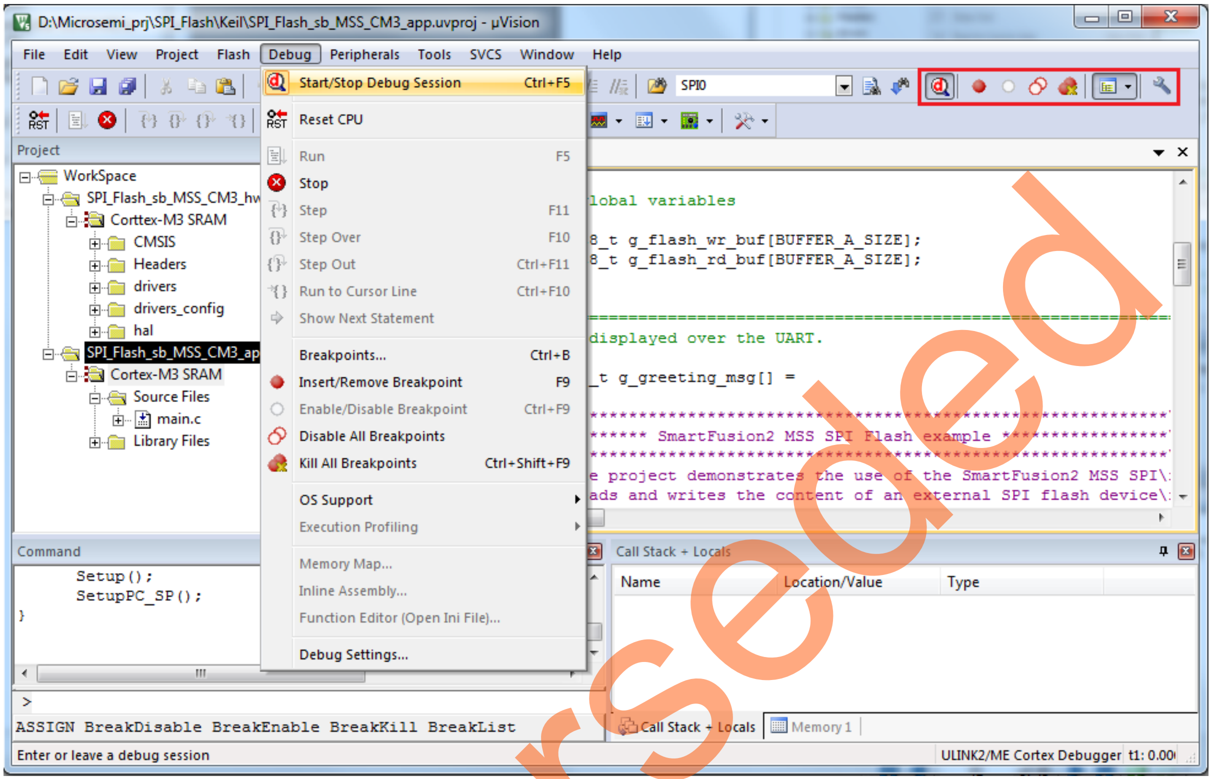
```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Show or hide the Registers Window

ULINK2/ME Cortex Debugger t1: 0.00

***Figure 43* • Values of the Cortex-M3 Internal Registers**

7. When debug process is finished, terminate execution of the code by choosing **Debug > Start/Stop Debug Session** as shown in Figure 44.



```
lobal variables

8_t g_flash_wr_buf[BUFFER_A_SIZE];
8_t g_flash_rd_buf[BUFFER_A_SIZE];

================================================================
displayed over the UART.

_t g_greeting_msg[] =

****************************************************************
****** SmartFusion2 MSS SPI Flash example ***********************
****************************************************************
e project demonstrates the use of the SmartFusion2 MSS SPI\
ads and writes the content of an external SPI flash device\
```

```
      Setup();
      SetupPC_SP();
}
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Enter or leave a debug session   ULINK2/ME Cortex Debugger t1: 0.00(

Figure 44 **• Keil uVision Workbench - Stop Debug Option**

8. The Step Level Debugging can be performed before running the application using **Run**. These can be accessed from the Debug menu or on the Keil uVision workbench as shown in Figure 45:

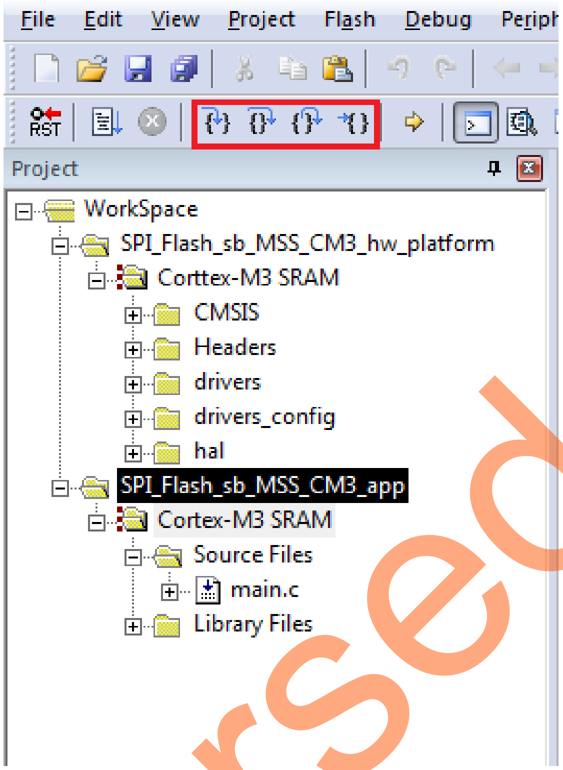


Figure 45 • **Keil uVision Workbench - Step Level Debugging**

- Source code can be single-stepped by selecting from the Debug menu **Debug > Step**, **Debug > Step Over**, **Debug > Step Out** or by selecting the respective options from the Keil uVision workbench as shown in Figure 45. Observe the changes in the source code window and Disassembly section. 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.
- Select **Debug > Step Out** to exit the instruction in stepping mode.

9. Add breakpoints from the **Debug** menu in workbench to force the code to halt, start Debug session, and then single-step and observe the instruction sequence.
10. Close uVision using **File > Exit**.
11. Close the HyperTerminal using **File > Exit**.

Conclusion

This tutorial provides steps to create a Libero SoC software design using the System Builder. It describes how to build, debug, and run Keil uVision application. It also provides a simple design to access the SPI flash.

Appendix A - Board Setup for Debugging from Keil uVision

Figure 1 shows the board setup for debugging the Keil uVision on the SmartFusion2 Security Evaluation Kit board.

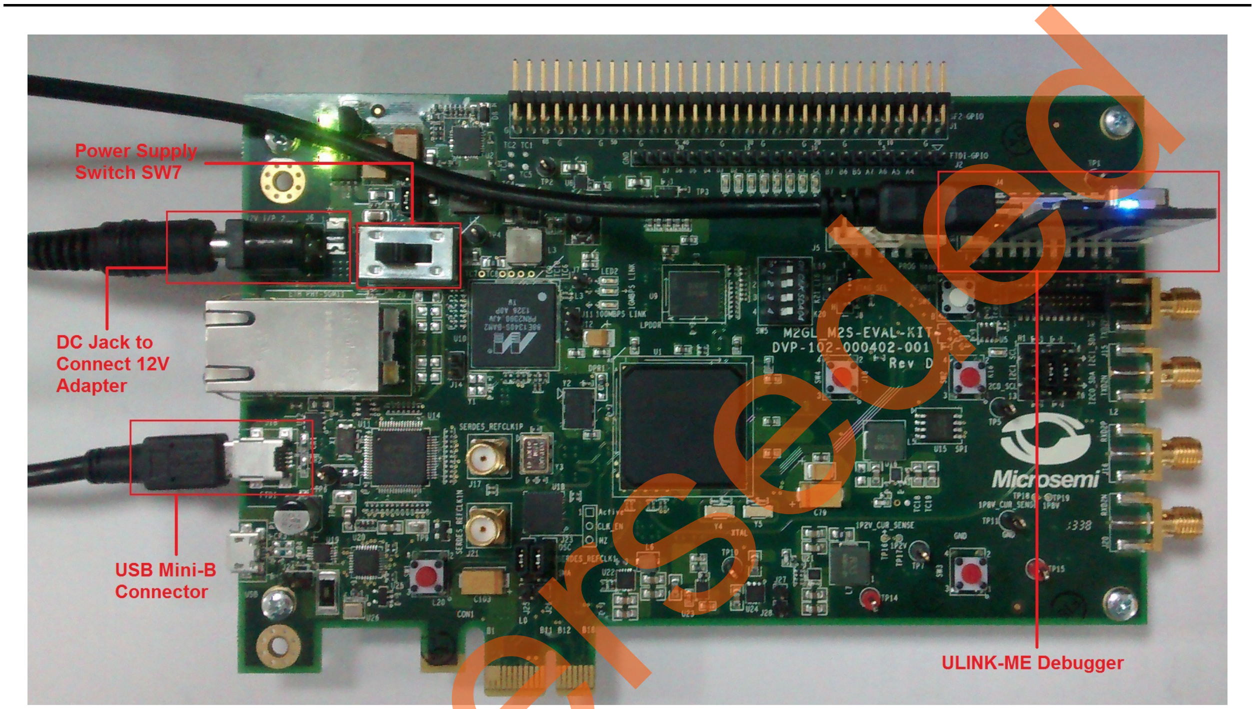


Figure 1 • **SmartFusion2 Security Evaluation Kit in Debug Mode using Keil uVision**

Appendix B - Board Setup for Programming the Tutorial

Figure 1 shows the board setup for running the tutorial on the SmartFusion2 Security Evaluation Kit board.

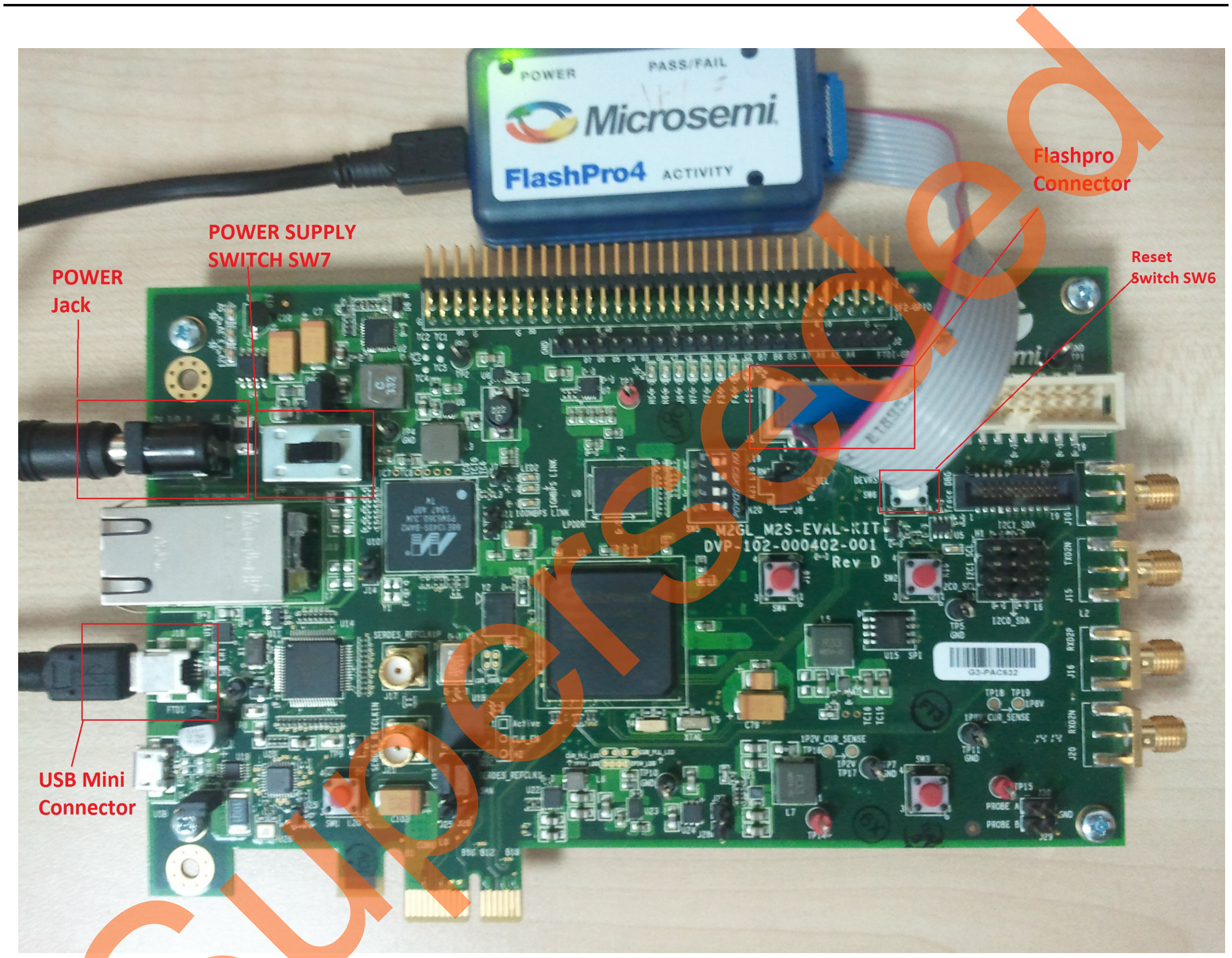


Figure 1 **• SmartFusion2 Security Evaluation Kit in Programming Mode**

Appendix C- SmartFusion2 Security Evaluation Kit Board Jumper Locations

Figure 1 shows the jumper locations on the SmartFusion2 Security Evaluation Kit board.

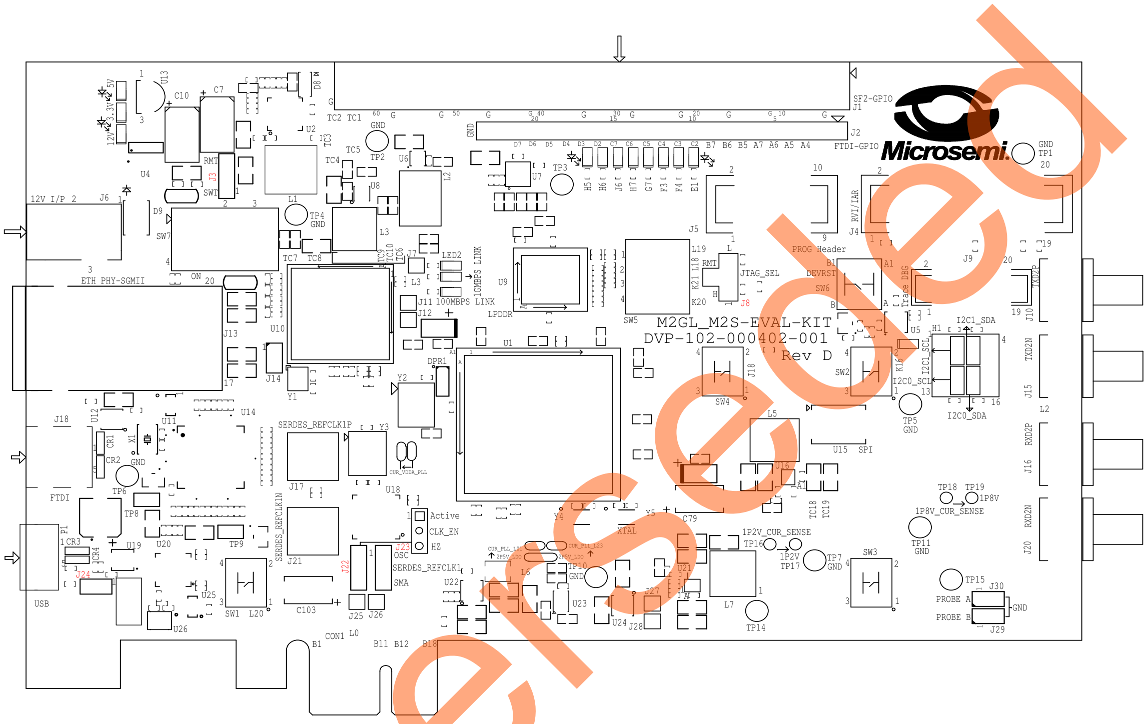


Figure 1 • SmartFusion2 Security Evaluation Kit Board Jumper Locations

Notes:

- Jumpers highlighted in red (J22, J23, J24, J3, J8) are set by default.
- The location of the jumpers in Figure 1 are searchable.

List of Changes

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

Revision*	Changes	Page
Revision 3 (March 2015)	Updated the document for Libero SoC v11.5 software release (SAR 64189).	N/A
Revision 2 (November 2014)	Updated the document for Libero SoC v11.4 software release (SAR 61938).	N/A
Revision 1 (April 2014)	Initial release.	N/A

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

Product Support

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

Customer Service

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

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

Customer Technical Support Center

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

Technical Support

For Microsemi SoC Products Support, visit

http://www.microsemi.com/products/fpga-soc/designsupport/fpga-soc-support

Website

You can browse a variety of technical and non-technical information on the 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 Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

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

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.