Developing an Application on Core8051s IP-Based Embedded Processor System Using Firmware Catalog Drivers

User’s Guide

Table of Contents

Introduction

This tutorial shows how to use Firmware Catalog drivers to develop a sample Core8051s IP-based application on the Microsemi® Fusion Advanced Development Kit.

Design Overview

This application is developed for the Fusion Advanced Development Kit. A Core8051s IP-based system is developed with peripherals like CoreGPIO, CoreUARTapb, CoreWatchdog, CoreTimer, and CoreInterrupt on Microsemi Fusion field programmable gate array (FPGA) using Libero® System-on-Chip (cSoC). Drivers for these peripherals are generated from the Firmware Catalog. A firmware project is created by importing the generated drivers into Microsemi's SoftConsole software. This tutorial illustrates the usage of the drivers. A sample application is developed for blinking on board LED at a specified time interval. This project is build such that it debugs the code from the external static random access memory (SRAM).

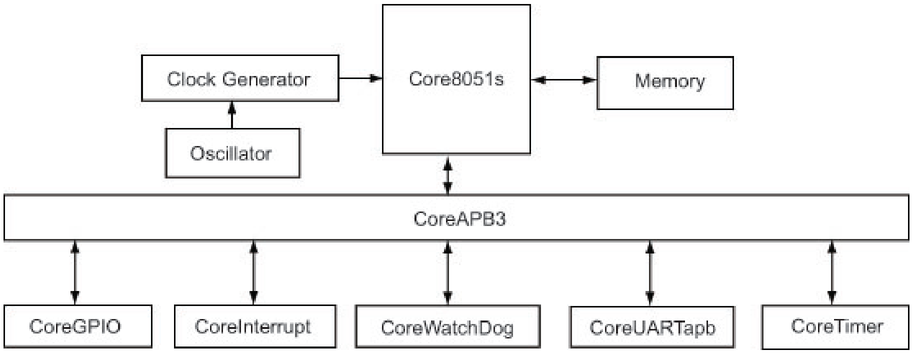


Figure 1 Block Diagram

Tutorial Requirements

Software Requirements

This tutorial requires the following software installed on your PC:

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

Hardware Requirements

You need the following hardware:

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

Associated Project Files

You can download the associated project files for this tutorial from the Microsemi website: www.microsemi.com/soc/download/rsc/?f=Core8051s_EmbProc_tutorial_DF.

Working with Libero SoC and SoftConsole

Step 1 - Creating Hardware Design Using Libero

Microsemi provides tutorials for Libero SoC on the Microsemi website in addition to training classes.

1. Unzip the M1AFS_Core8051s_DF.rar file to a root directory (C:\).
2. Open the Core8051s_Ext_SRAM project folder, as shown in Figure 2.

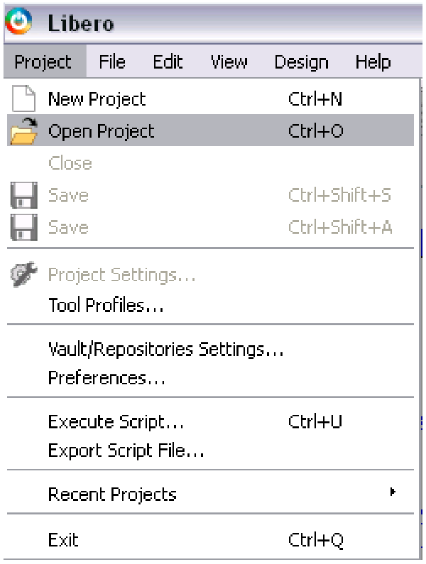
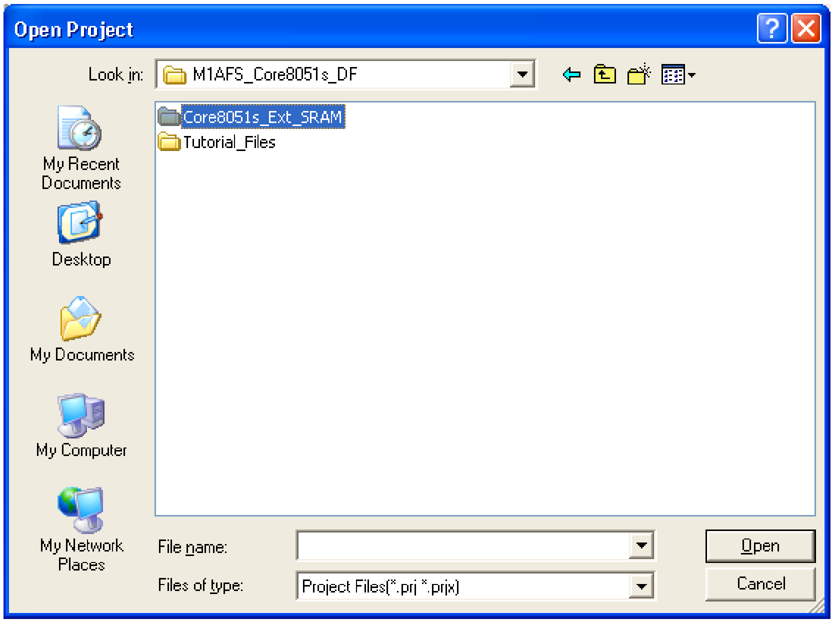


Figure 2 Opening Libero Project

3. Select the folder (Figure 3) and click **Open**.



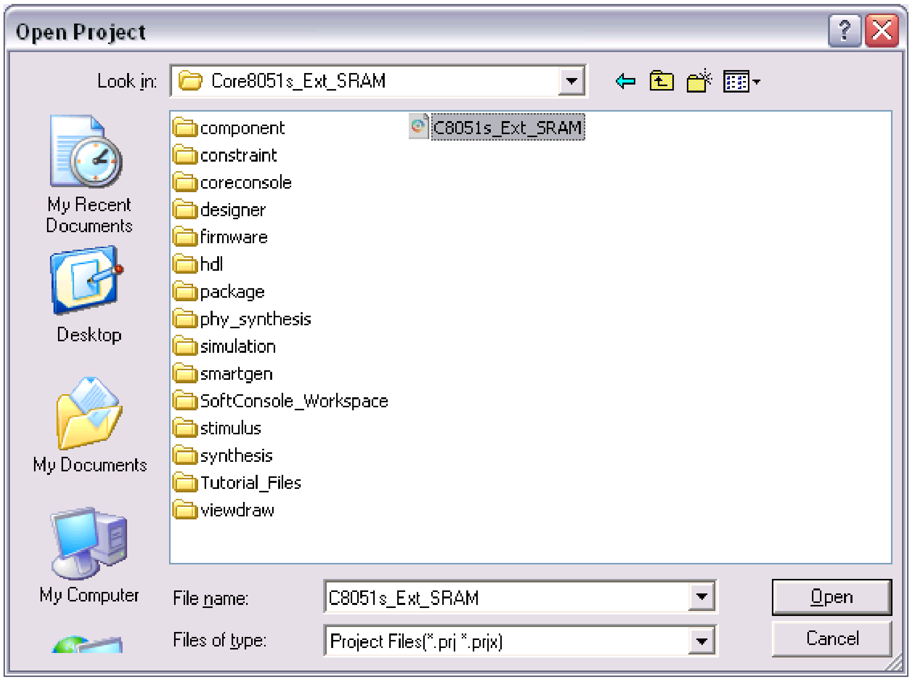
Look in: M1AFS_Core8051s_DF

File name:

Files of type: Project Files(*.prj *.prjx)

Figure 3 Selecting Project Folder

4. Select C8051s_EXT_SRAM, as shown in Figure 4.



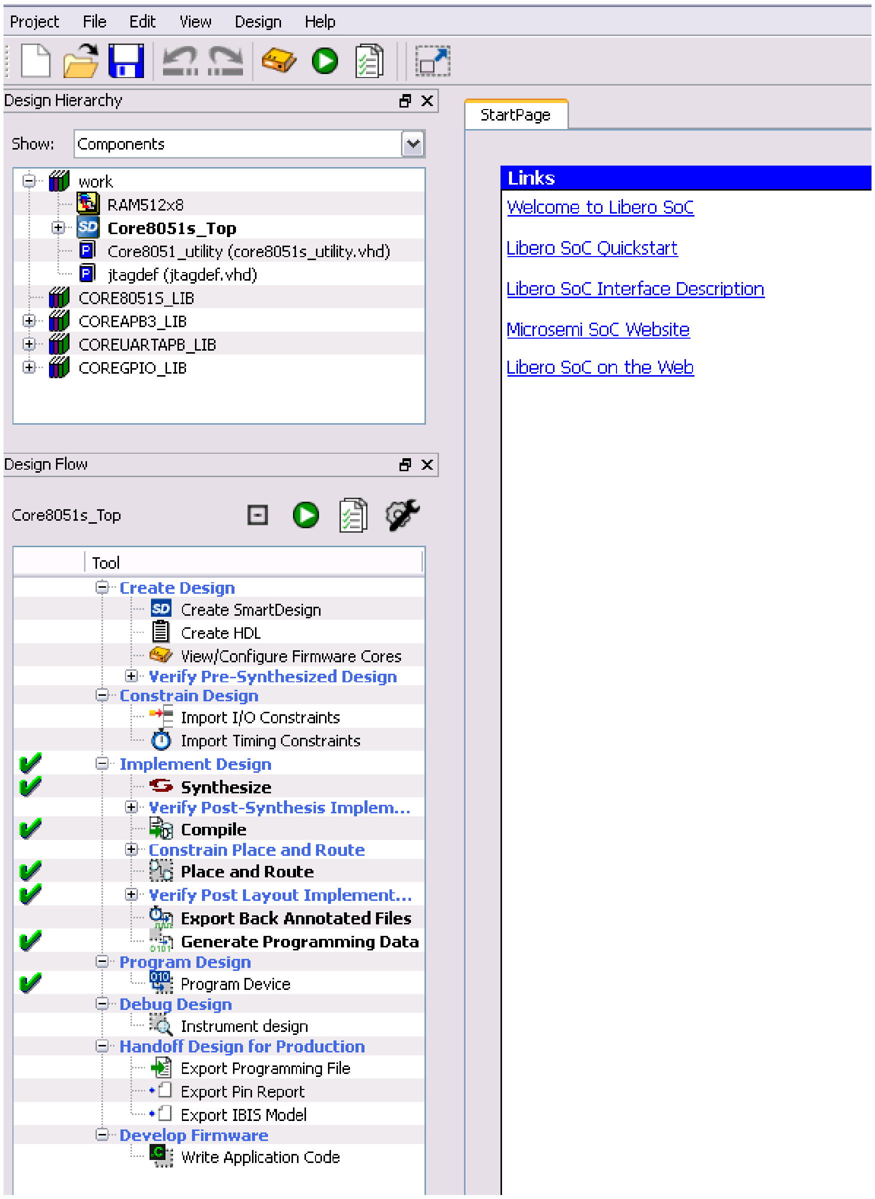
Look in: Core8051s_Ext_SRAM

File name: C8051s_Ext_SRAM

Files of type: Project Files(*.prj *.prjx)

Figure 4 Selecting Project File

5. The project flow window is displayed, as shown in Figure 5.



Show: Components

Figure 5 Project Flow Window

6. Right-click the top level SmartDesign component (for example, Core8051s_Top) and select **Open Component**, as shown in Figure 6.

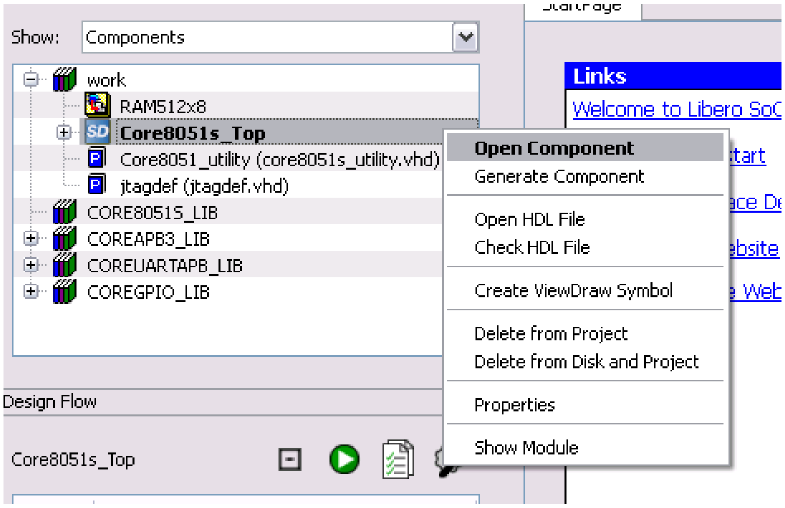


Figure 6 Open Top Level Component

7. The top level SmartDesign window is displayed, as shown in Figure 7.

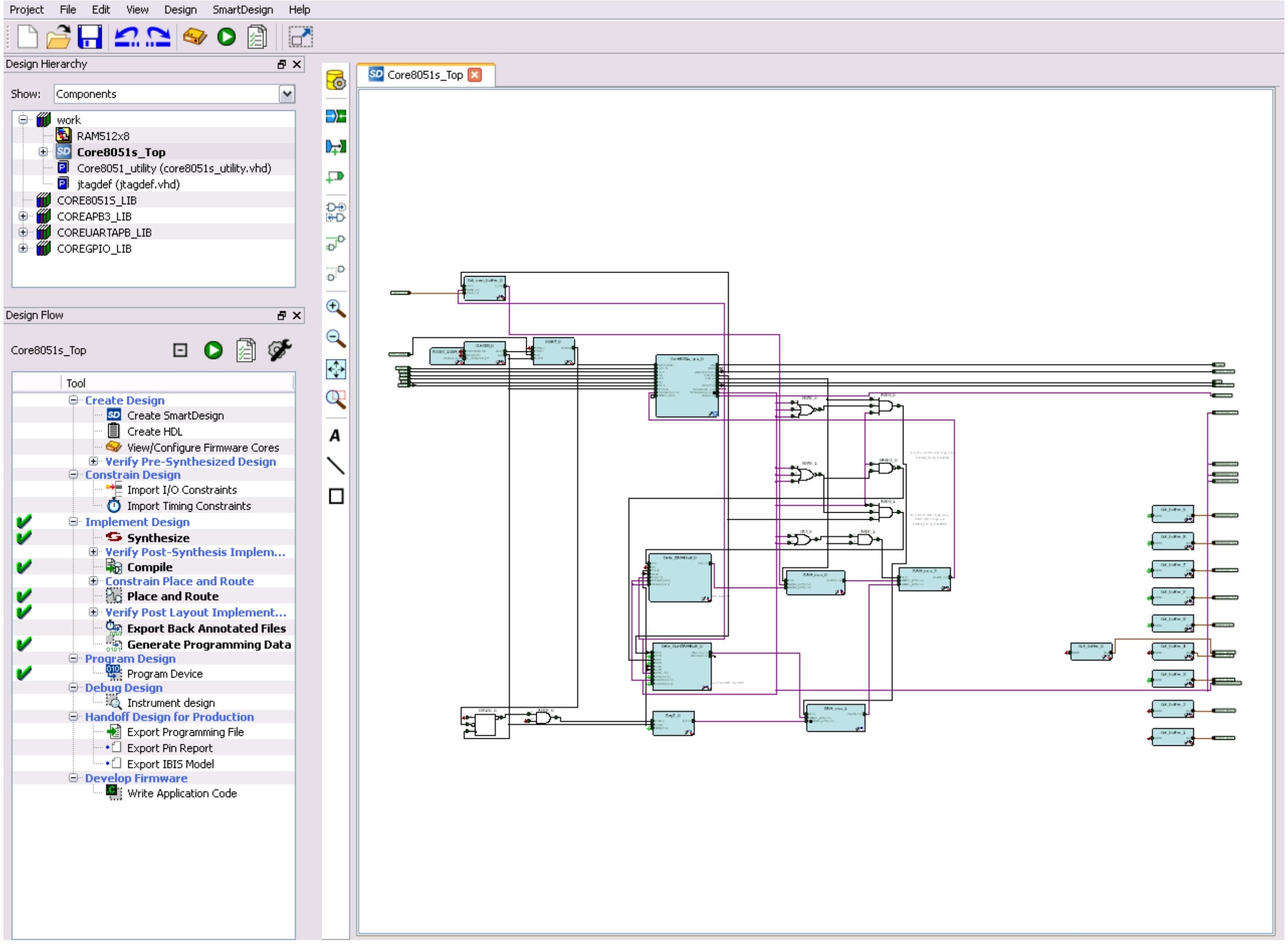


Figure 7 Top Level SmartDesign Window

8. Click **Maximize Work Area**, as shown in Figure 8.

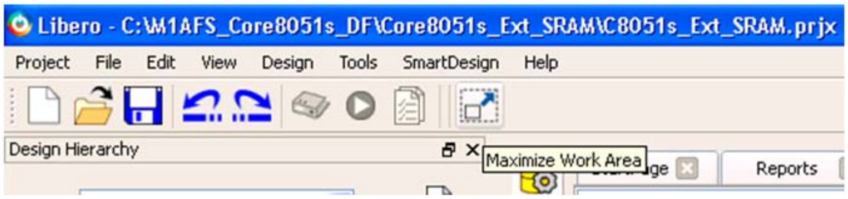


Figure 8 Maximize Work Area

9. You can view how different peripherals are connected to Core8051s processor using CoreAPB3 interface. Refer to the Core8051s based hardware tutorial for step-by-step procedures to create a Libero design.

   Program memory space is implemented using 64 kbytes of the external SRAM on the Fusion Advanced Development Kit.

   The peripherals used in this design are assigned at the following addresses:

• COREGPIO_0:	0x0000F000
• CORETIMER_0:	0x0000F100
• COREWATCHDOG_0:	0x0000F200
• COREUARTAPB_0:	0x0000F500
• COREINTERRUPT_0:	0x0000F600

   Note: To check the base addresses of the peripherals, view the memory map of the design.

   You can refer to these address allocations in Core8051s_Sys_hw_platform.h file provided in the SoftConsole project.

   Note: These addresses are specific to this project. It can vary from one design to another.

   The Timer interrupt is connected to INT0 of Core8051s through Core Interrupt.

10. Connect the LC Programmer to the Fusion Advanced Development Kit.
11. Connect a USB cable from a USB port on your PC to the USB connector on the LC Programmer. Power cycle the board.
12. If prompted by the operating system for drivers, you can download the drivers for the CP2102 USB serial port from www.microsemi.com/soc/products/hardware/devkits_boards/fusion_embedded.aspx.

    Note: Refer to the Core8051s based hardware tutorial for more information.

13. Double-click **Program Device** under Program Design in the Design Flow window to program the device, as shown in Figure 9.

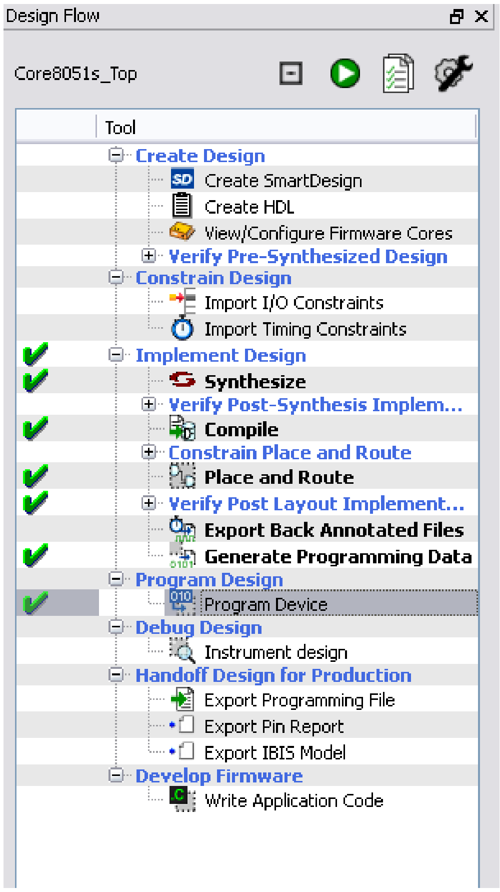
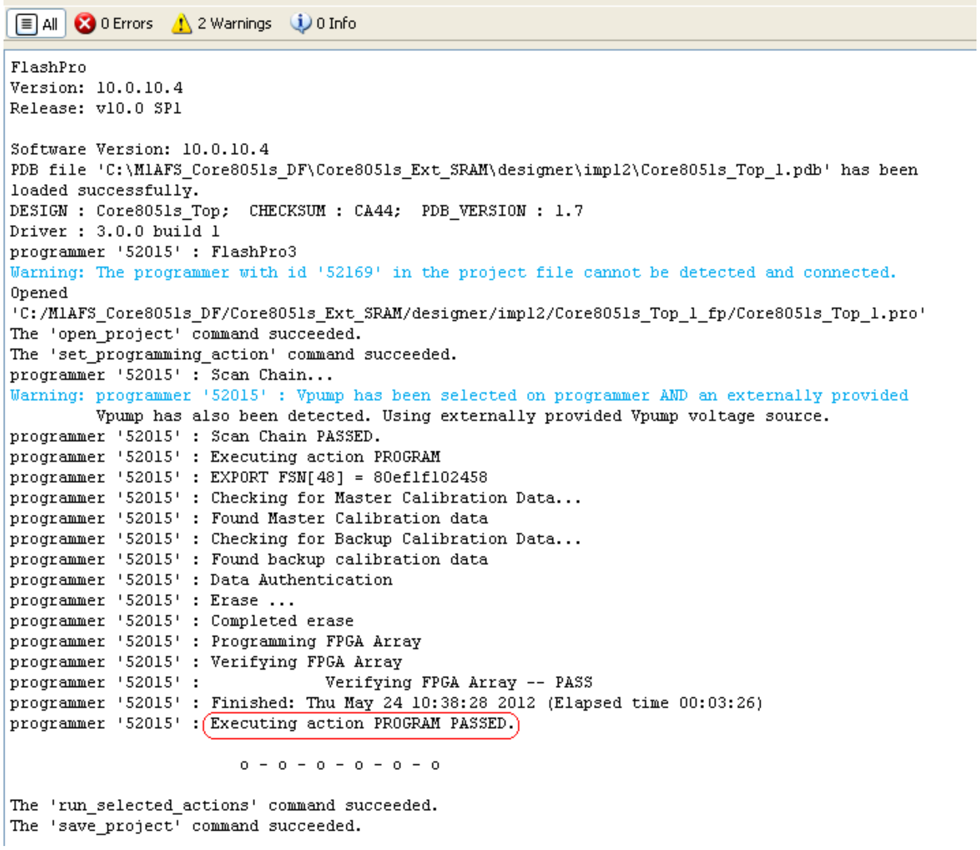


Figure 9 Program Device

14. You can see the message “PROGRAM PASSED”, as shown in Figure 10.



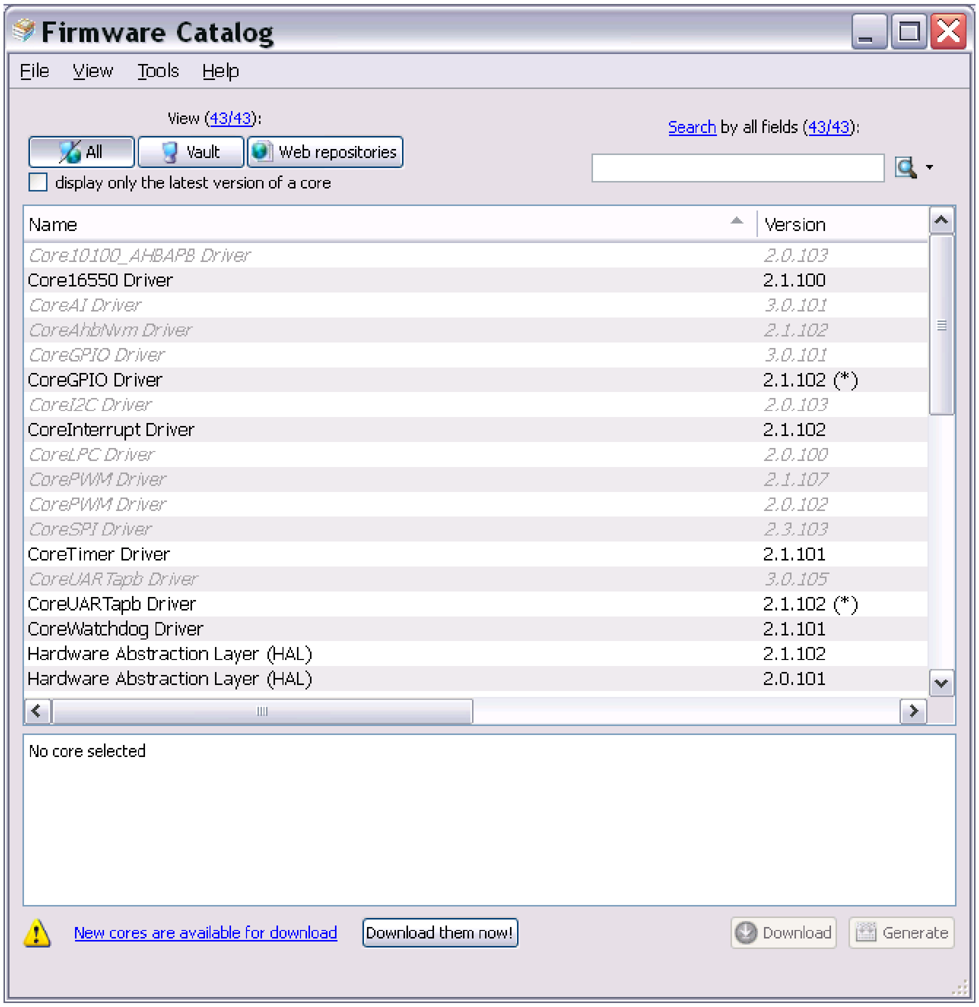
```
FlashPro
Version: 10.0.10.4
Release: v10.0 SP1

Software Version: 10.0.10.4
PDB file 'C:\M1AFS_Core8051s_DF\Core8051s_Ext_SRAM\designer\impl2\Core8051s_Top_1.pdb' has been
loaded successfully.
DESIGN : Core8051s_Top;  CHECKSUM : CA44;  PDB_VERSION : 1.7
Driver : 3.0.0 build 1
programmer '52015' : FlashPro3
Warning: The programmer with id '52169' in the project file cannot be detected and connected.
Opened
'C:/M1AFS_Core8051s_DF/Core8051s_Ext_SRAM/designer/impl2/Core8051s_Top_1_fp/Core8051s_Top_1.pro'
The 'open_project' command succeeded.
The 'set_programming_action' command succeeded.
programmer '52015' : Scan Chain...
Warning: programmer '52015' : Vpump has been selected on programmer AND an externally provided
         Vpump has also been detected. Using externally provided Vpump voltage source.
programmer '52015' : Scan Chain PASSED.
programmer '52015' : Executing action PROGRAM
programmer '52015' : EXPORT FSN[48] = 80ef1f102458
programmer '52015' : Checking for Master Calibration Data...
programmer '52015' : Found Master Calibration data
programmer '52015' : Checking for Backup Calibration Data...
programmer '52015' : Found backup calibration data
programmer '52015' : Data Authentication
programmer '52015' : Erase ...
programmer '52015' : Completed erase
programmer '52015' : Programming FPGA Array
programmer '52015' : Verifying FPGA Array
programmer '52015' :             Verifying FPGA Array -- PASS
programmer '52015' : Finished: Thu May 24 10:38:28 2012 (Elapsed time 00:03:26)
programmer '52015' : Executing action PROGRAM PASSED.

                        o - o - o - o - o - o

The 'run_selected_actions' command succeeded.
The 'save_project' command succeeded.
```

Figure 10 Program Passed

Step 2 - Generating Peripheral Drivers From Firmware Catalog

This section explains how to generate firmware drivers from Firmware Catalog.

1. Launch Firmware Catalog using **Start → All Programs → Microsemi Libero SoC v 10.0 → Firmware Catalog v 10.0 → Firmware Catalog**. The Firmware Catalog is displayed, as shown in Figure 11.



□ display only the latest version of a core

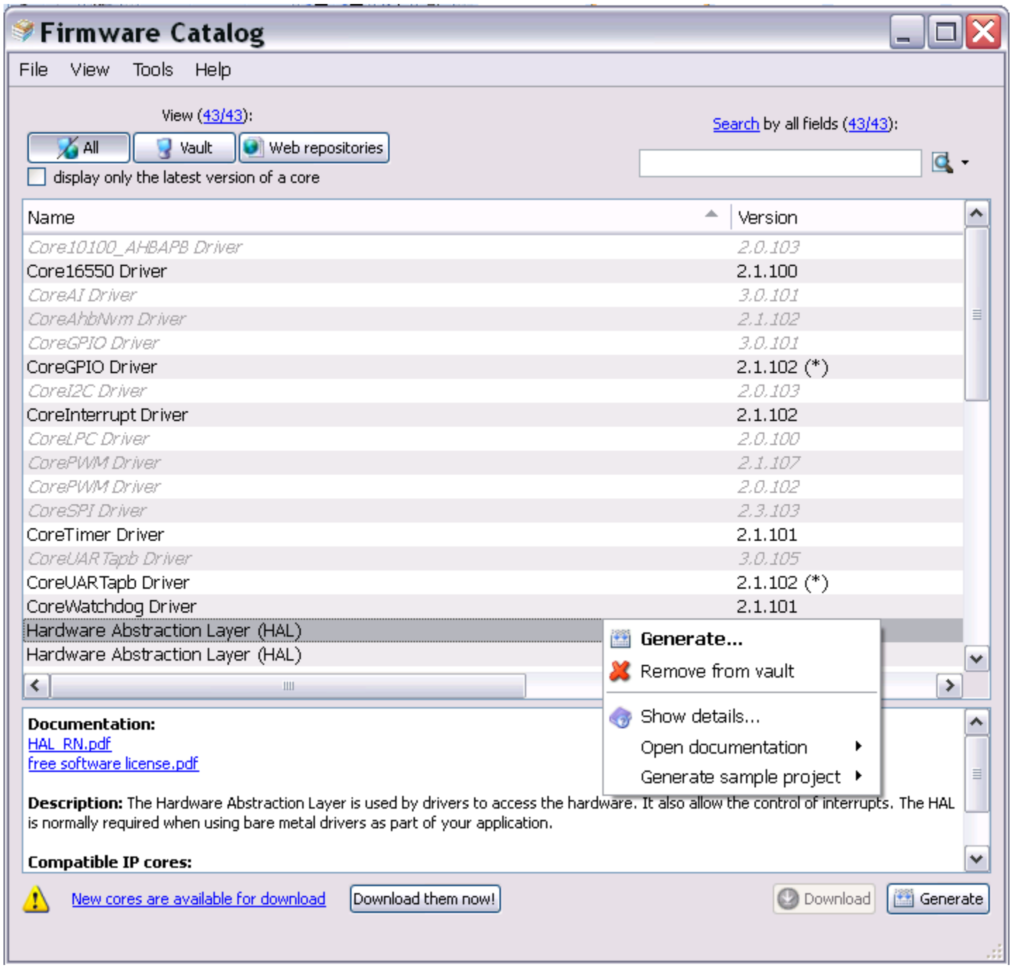
Name	Version
Core10100_AHBAPB Driver	2.0.103
Core16550 Driver	2.1.100
CoreAI Driver	3.0.101
CoreAhbNvm Driver	2.1.102
CoreGPIO Driver	3.0.101
CoreGPIO Driver	2.1.102 (*)
CoreI2C Driver	2.0.103
CoreInterrupt Driver	2.1.102
CoreLPC Driver	2.0.100
CorePWM Driver	2.1.107
CorePWM Driver	2.0.102
CoreSPI Driver	2.3.103
CoreTimer Driver	2.1.101
CoreUARTapb Driver	3.0.105
CoreUARTapb Driver	2.1.102 (*)
CoreWatchdog Driver	2.1.101
Hardware Abstraction Layer (HAL)	2.1.102
Hardware Abstraction Layer (HAL)	2.0.101

No core selected

New cores are available for download

Figure 11 Firmware Catalog

2. To generate the driver, right-click the required driver and click **Generate**, as shown in Figure 12.



Name	Version
Core10100_AHBAPB Driver	2.0.103
Core16550 Driver	2.1.100
CoreAI Driver	3.0.101
CoreAhbNvm Driver	2.1.102
CoreGPIO Driver	3.0.101
CoreGPIO Driver	2.1.102 (*)
CoreI2C Driver	2.0.103
CoreInterrupt Driver	2.1.102
CoreLPC Driver	2.0.100
CorePWM Driver	2.1.107
CorePWM Driver	2.0.102
CoreSPI Driver	2.3.103
CoreTimer Driver	2.1.101
CoreUARTapb Driver	3.0.105
CoreUARTapb Driver	2.1.102 (*)
CoreWatchdog Driver	2.1.101
Hardware Abstraction Layer (HAL)	
Hardware Abstraction Layer (HAL)	

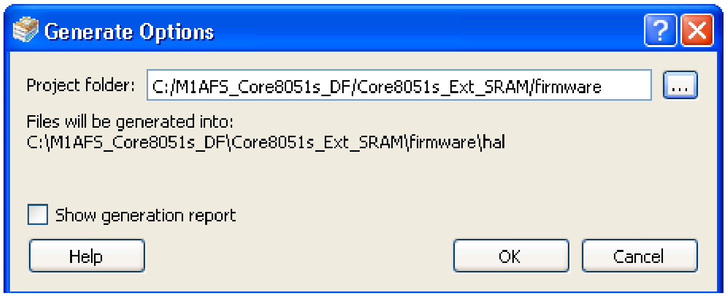
Documentation:
HAL_RN.pdf
free software license.pdf

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

Compatible IP cores:

Figure 12 Generate Driver

3. Save the generated drivers at: C:\M1AFS_Core8051s_DF\Core8051s_Ext_SRAM\firmware, as shown in Figure 13.



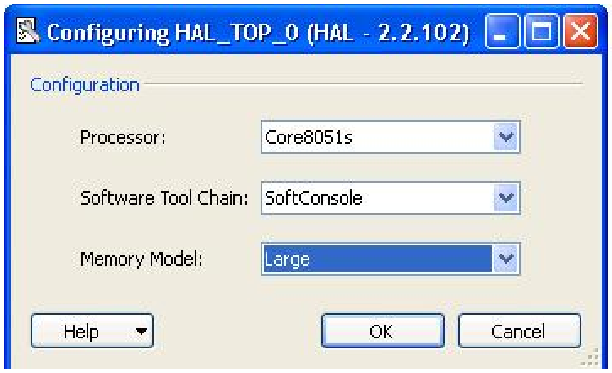
Project folder: C:/M1AFS_Core8051s_DF/Core8051s_Ext_SRAM/firmware

Files will be generated into:
C:\M1AFS_Core8051s_DF\Core8051s_Ext_SRAM\firmware\hal

□ Show generation report

Figure 13 Generate Options

4. Configure the HAL, as shown in Figure 14. The hardware abstraction layer (HAL) is used by the drivers to access the hardware. It also allows the control of interrupts. The HAL is required when using bare metal drivers as part of the application.



Configuring HAL_TOP_0 (HAL - 2.2.102)

Configuration

Processor: Core8051s

Software Tool Chain: SoftConsole

Memory Model: Large

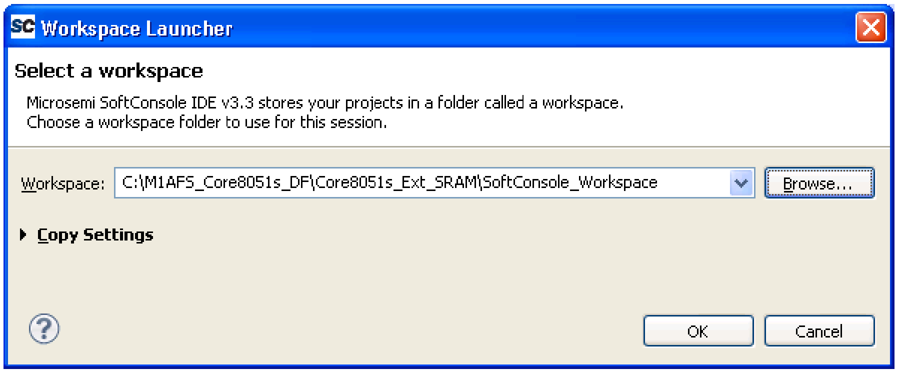
Figure 14 Configure HAL

5. Repeat the same procedure to generate the required drivers—CoreUARTapb, CoreGPIO, CoreWatchdog, CoreInterrupt, and CoreTimer.

Step 3 - Creating a SoftConsole Project

This section explains how to create a SoftConsole project using peripheral drivers generated from the Firmware Catalog and launch the debugger to run the application.

1. Launch SoftConsole IDE using **Start → All Programs → Microsemi SoftConsole v3.3 → Microsemi SoftConsole IDE v3.3.0.14**.
2. Open **SoftConsole** and launch new workspace, as shown in Figure 15.



Workspace Launcher

Select a workspace

Microsemi SoftConsole IDE v3.3 stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\M1AFS_Core8051s_DF\Core8051s_Ext_SRAM\SoftConsole_Workspace

Figure 15 Selecting SoftConsole Workspace

3. Click **OK**.

4. Go to **File → New → C project**, as shown in Figure 16.

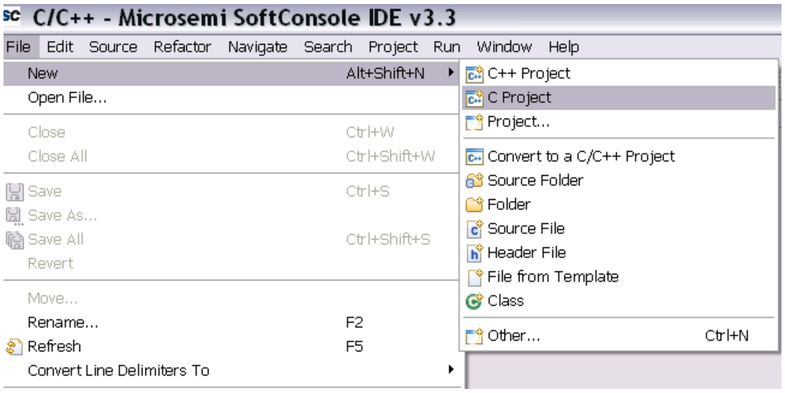
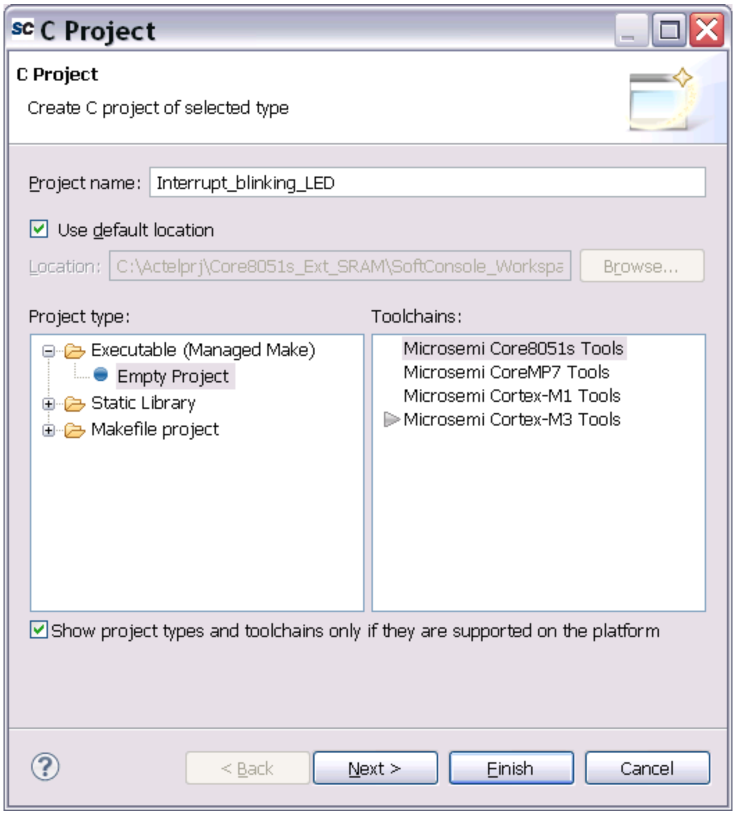


Figure 16 Creating New C Project

5. Go to **Project Type → Executable (Managed Make)** and select **Empty Project**.
6. Go to ToolChains and select **Microsemi Core8051s** Tools.
7. Name the project as Interrupt_blinking_LED, as shown in Figure 17.


C Project

Create C project of selected type

Project name: Interrupt_blinking_LED

☑ Use default location

Location: C:\Actelprj\Core8051s_Ext_SRAM\SoftConsole_Workspa

Project type: Executable (Managed Make) > Empty Project

Toolchains: Microsemi Core8051s Tools

☑ Show project types and toolchains only if they are supported on the platform

Figure 17 Selecting Project Type and ToolChains

8. Click **Next** and click **Finish**.

9. The created new project is displayed, as shown in Figure 18.

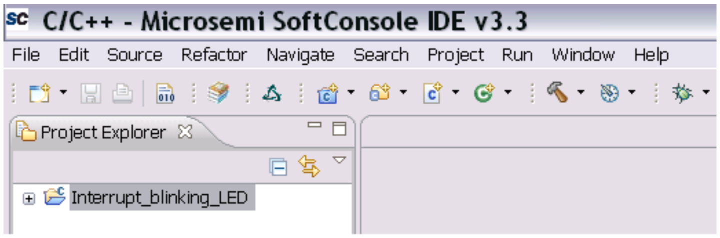

C/C++ - Microsemi SoftConsole IDE v3.3
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Interrupt_blinking_LED

Figure 18 New Project Created

Step 4 - Importing Generated Drivers From Firmware Catalog to the SoftConsole Project

Importing the Drivers

1. Right-click Interrupt_blinking_LED and click **Import**, as shown in Figure 19.

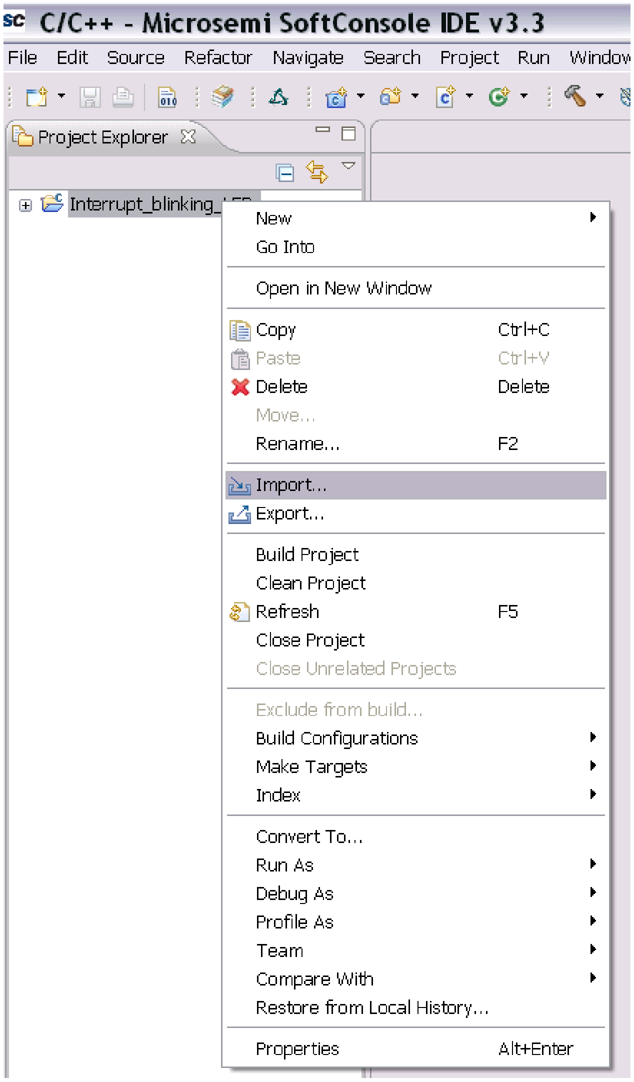
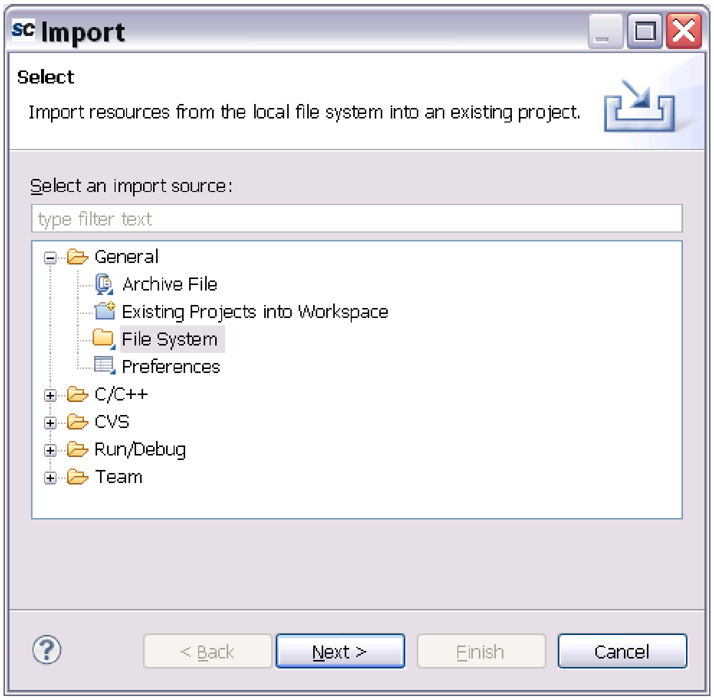


Figure 19 Importing Files into Project

2. Click **General**, select the **File System** and click **Next**, as shown in Figure 20.



Import

Select

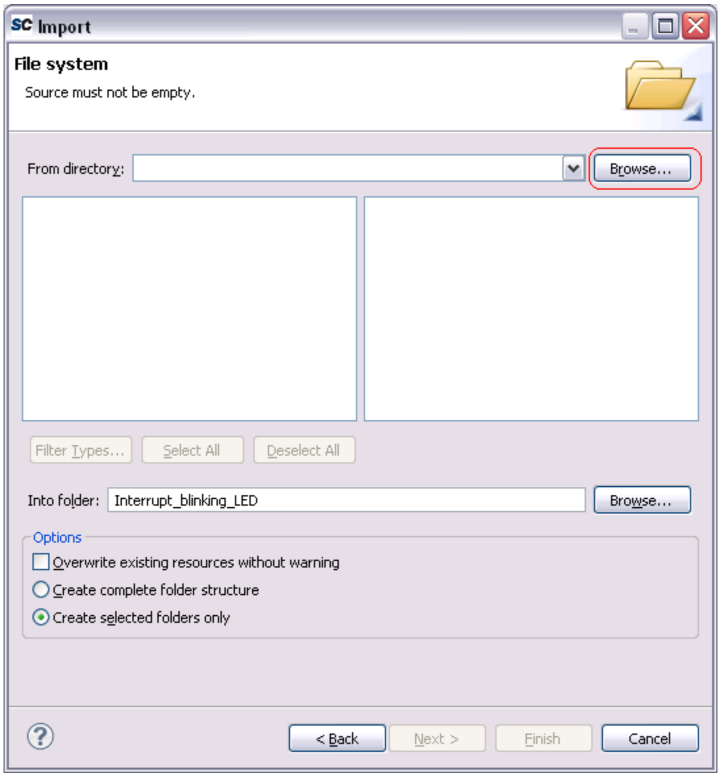
Import resources from the local file system into an existing project.

Select an import source:

type filter text

Figure 20 Select an Import Source

3. Click **Browse** to open the project path, as shown in Figure 21.



Import

File system

Source must not be empty.

From directory:

Into folder: Interrupt_blinking_LED

Options

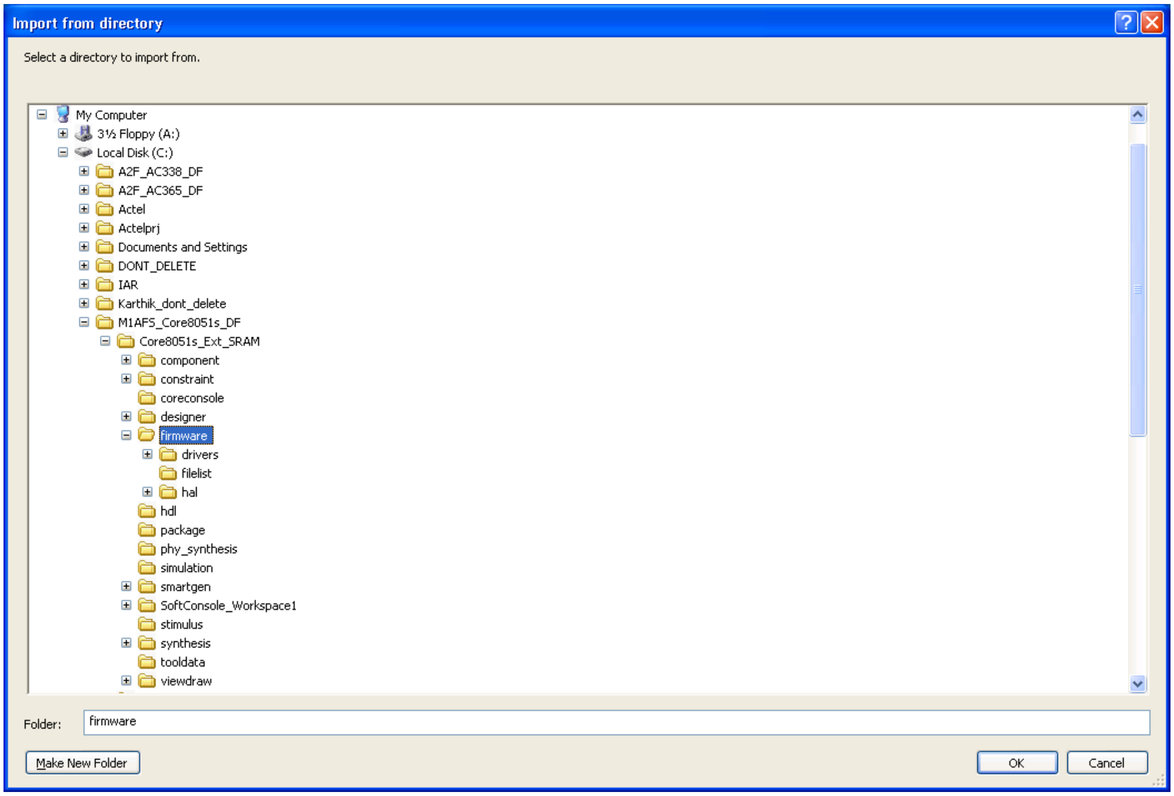
☐ Overwrite existing resources without warning

○ Create complete folder structure

● Create selected folders only

Figure 21 Open the Project Path

4. Go to the path where the firmware drivers are saved, that is C:\M1AFS_Core8051s_DF\Core8051s_Ext_SRAM\firmware (Figure 22).

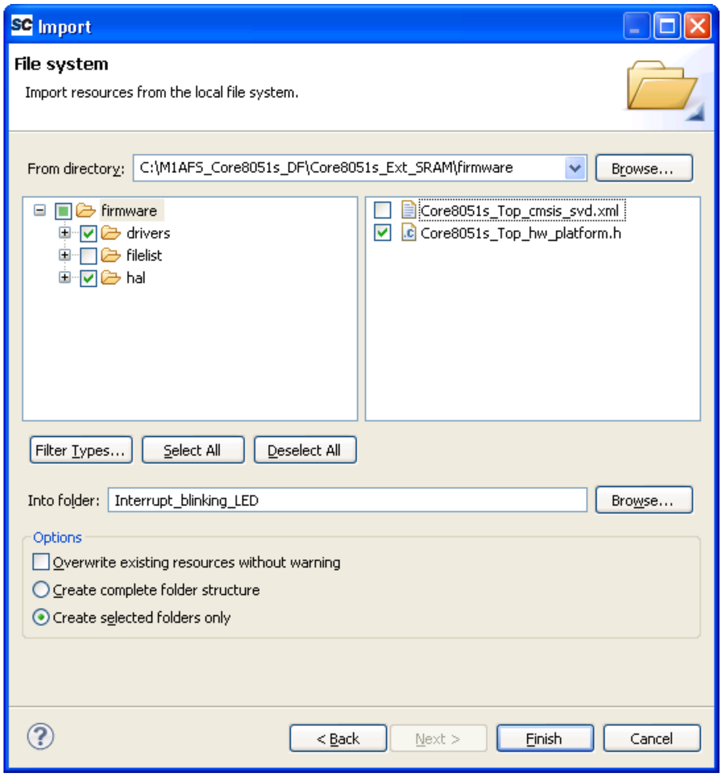


Folder: firmware

Figure 22 Firmware Drivers

5. Click **OK**.

6. Select the drivers, HAL and Core8051s_Top_hw_platform.h file, as shown in Figure 23.



Import

File system

Import resources from the local file system.

From directory: C:\M1AFS_Core8051s_DF\Core8051s_Ext_SRAM\firmware

- ⊟ ■ firmware
  - ⊞ ☑ drivers
  - ⊞ □ filelist
  - ⊞ ☑ hal

- □ Core8051s_Top_cmsis_svd.xml
- ☑ Core8051s_Top_hw_platform.h

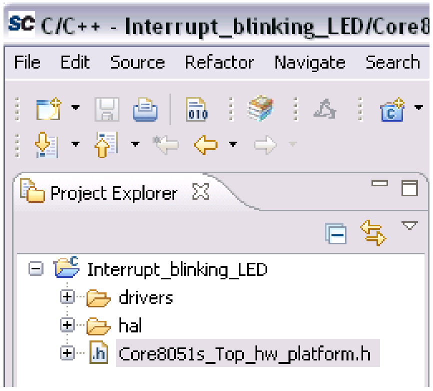
Into folder: Interrupt_blinking_LED

Options

- □ Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

Figure 23 Select the Files to Import

7. Click **Finish**.
8. The project explorer is displayed, as shown in Figure 24.



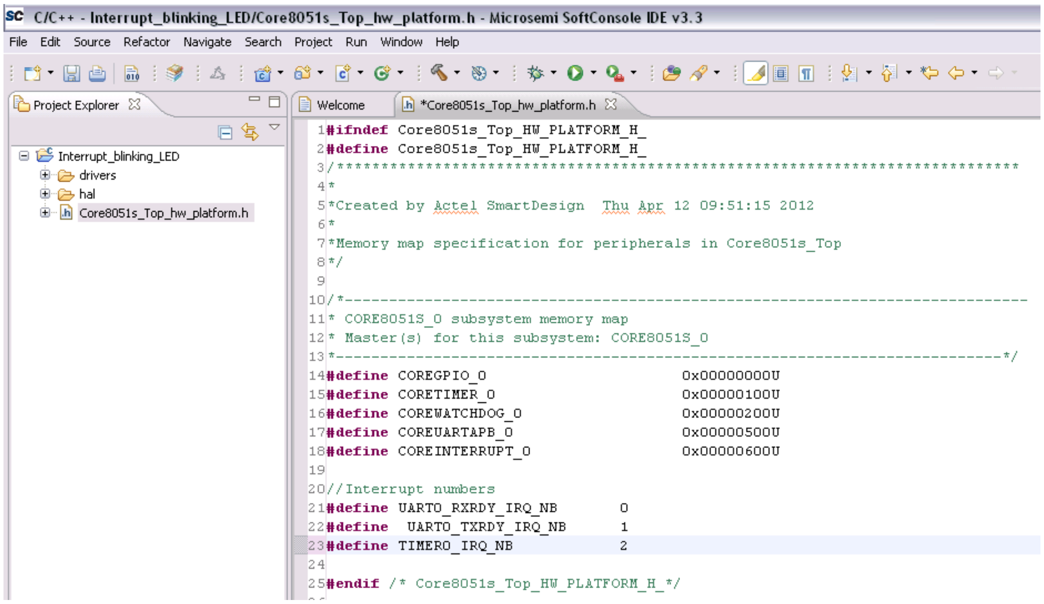
C/C++ - Interrupt_blinking_LED/Core8

- ⊟ Interrupt_blinking_LED
  - ⊞ drivers
  - ⊞ hal
  - ⊞ Core8051s_Top_hw_platform.h

Figure 24 Project Explorer

9. Open Core8051s_sys_hw_platform.h file and add the following definitions (Figure 25).

```
//Interrupt numbers
#define UART0_RXRDY_IRQ_NB 0
#define UART0_TXRDY_IRQ_NB 1
#define TIMER0_IRQ_NB            2
```


C/C++ - Interrupt_blinking_LED/Core8051s_Top_hw_platform.h - Microsemi SoftConsole IDE v3.3
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Interrupt_blinking_LED
drivers
hal
Core8051s_Top_hw_platform.h
Welcome
*Core8051s_Top_hw_platform.h

```
#ifndef Core8051s_Top_HW_PLATFORM_H_
#define Core8051s_Top_HW_PLATFORM_H_
/******************************************************************************
*
*Created by Actel SmartDesign  Thu Apr 12 09:51:15 2012
*
*Memory map specification for peripherals in Core8051s_Top
*/

/*-----------------------------------------------------------------------------
* CORE8051S_0 subsystem memory map
* Master(s) for this subsystem: CORE8051S_0
*---------------------------------------------------------------------------*/
#define COREGPIO_0                      0x00000000U
#define CORETIMER_0                     0x00000100U
#define COREWATCHDOG_0                  0x00000200U
#define COREUARTAPB_0                   0x00000500U
#define COREINTERRUPT_0                 0x00000600U

//Interrupt numbers
#define UART0_RXRDY_IRQ_NB       0
#define  UART0_TXRDY_IRQ_NB      1
#define TIMER0_IRQ_NB            2

#endif /* Core8051s_Top_HW_PLATFORM_H_*/
```

Figure 25 Modified Hardware Platform File

10. Create a new C source file, as shown in Figure 26.

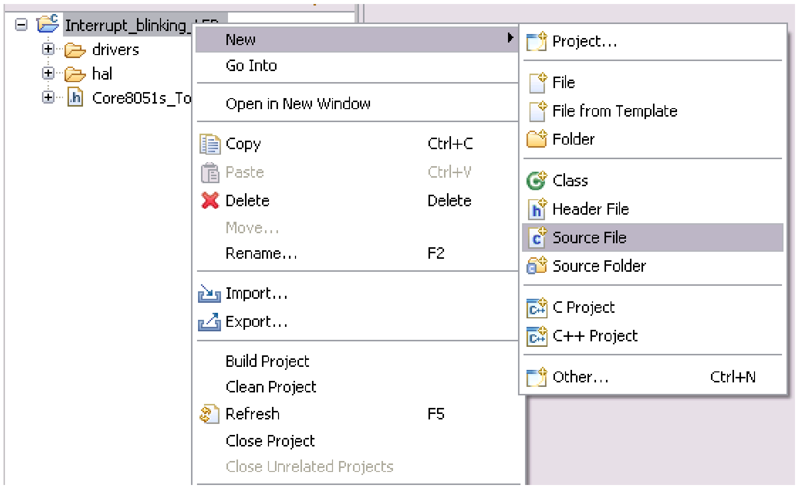
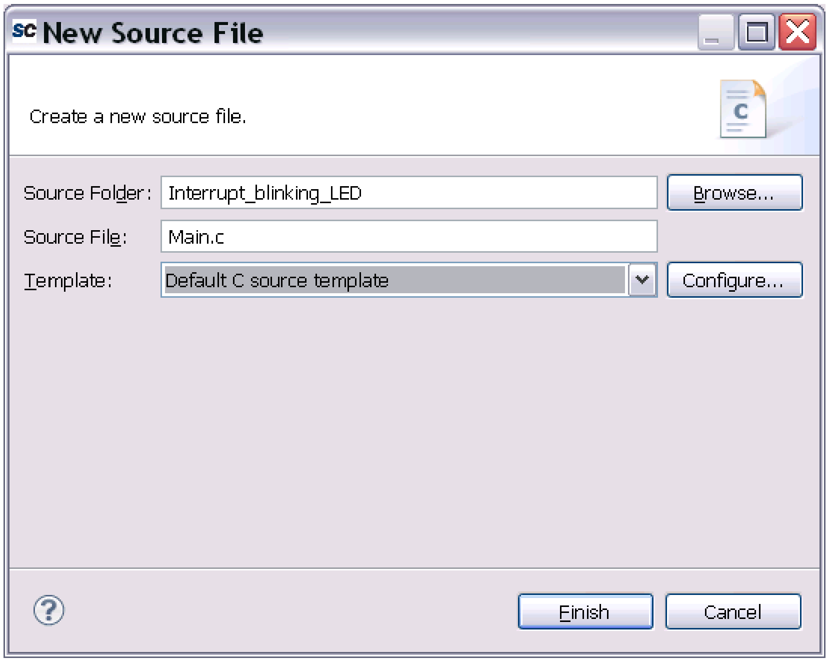


Figure 26 Creating New Source File

11. Name it as **Main.c**, as shown in Figure 27.



Create a new source file.

Source Folder: Interrupt_blinking_LED

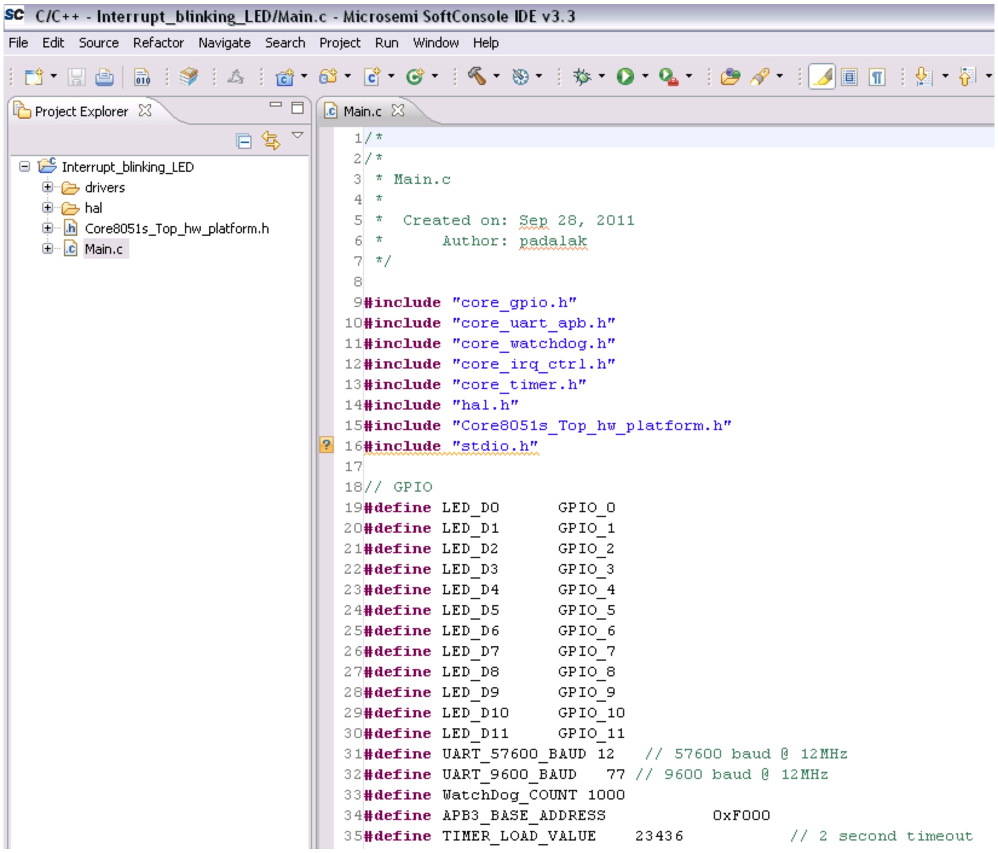
Source File: Main.c

Template: Default C source template

Figure 27 Name the Source File

12. Open the Main.c file which is given in the Tutorial_Files folder. Copy the code and paste in the new Main.c file source file.

13. Save **Main.c** file. The SoftConsole project is displayed, as shown in Figure 28.



```
/*
/*
 * Main.c
 *
 *  Created on: Sep 28, 2011
 *      Author: padalak
 */

#include "core_gpio.h"
#include "core_uart_apb.h"
#include "core_watchdog.h"
#include "core_irq_ctrl.h"
#include "core_timer.h"
#include "hal.h"
#include "Core8051s_Top_hw_platform.h"
#include "stdio.h"

// GPIO
#define LED_D0       GPIO_0
#define LED_D1       GPIO_1
#define LED_D2       GPIO_2
#define LED_D3       GPIO_3
#define LED_D4       GPIO_4
#define LED_D5       GPIO_5
#define LED_D6       GPIO_6
#define LED_D7       GPIO_7
#define LED_D8       GPIO_8
#define LED_D9       GPIO_9
#define LED_D10      GPIO_10
#define LED_D11      GPIO_11
#define UART_57600_BAUD 12   // 57600 baud @ 12MHz
#define UART_9600_BAUD   77 // 9600 baud @ 12MHz
#define WatchDog_COUNT 1000
#define APB3_BASE_ADDRESS           0xF000
#define TIMER_LOAD_VALUE    23436            // 2 second timeout
```

Figure 28 SoftConsole Project with the Source File

Step 5 - Firmware Drivers Modification

CoreGPIO

Core8051s is an 8-bit processor. The functions like **GPIO_get_inputs** and **GPIO_get_outputs**, which are contained in the file core_gpio.c, should be modified to work with Core8051s.

The below given lines have to be commented in the coregpio.c file:

Changes made to GPIO_get_inputs

```
case GPIO_APB_16_BITS_BUS:
//      gpio_in |= (HAL_get_16bit_reg( this_gpio->base_addr, GPIO_IN1) << 16);
      break;

case GPIO_APB_8_BITS_BUS:
  //   gpio_in |= (HAL_get_8bit_reg( this_gpio->base_addr, GPIO_IN2) << 16);
  //   gpio_in |= (HAL_get_8bit_reg( this_gpio->base_addr, GPIO_IN3) << 24);
     break;
```

Changes made to GPIO_get_outputs

```
        case GPIO_APB_16_BITS_BUS:
//        gpio_out |= (HAL_get_16bit_reg( this_gpio->base_addr, GPIO_OUT1) << 16);
        break;

        case GPIO_APB_8_BITS_BUS:
   //      gpio_out |= (HAL_get_16bit_reg( this_gpio->base_addr, GPIO_OUT2) << 16);
    //     gpio_out |= (HAL_get_16bit_reg( this_gpio->base_addr, GPIO_OUT3) << 24);
        break;
```

The modified GPIO driver that is core_gpio.c is available in the Tutorial_files folder.

8051s HAL

The HAL drivers do not have the code to enable 8051s interrupts. Hence, the following function is included in Main.c to enable the 8051s interrupts.

```
void enable_8051_interrupts(void)
{
       __asm
       setb ex0 ; if using INT0
setb ex1 ; if using INT1
       setb ea
       __endasm;
}
```

The HAL does not provide a function to enable edge or level interrupts. The below function is included in Main.c to accomplish this.

```
void C8051_interrupt_type(void)
{
       __asm
       setb it0 ; if set INT0 for edge operation
       setb it1 ; if set INT1 for edge operation
       __endasm;
}
```

The function CIC_irq_handler contained in core_irq_ctrl.c (part of the CoreInterrupt firmware driver) can be used to handle interrupts from CoreInterrupt.

Note: These changes are already added in the Main.c file.

Modifying Assembler Settings

In SoftConsole v3.3, the assembler settings are omitted from the Core8051s project default build settings. Due to this, the HAL's assembly file (hal_large.s or hal_small.s) is not compiled and the build fails. The following steps are required to fix this:

1. Select the Project.
2. Open the project properties window (Project → Properties), as shown in Figure 29.

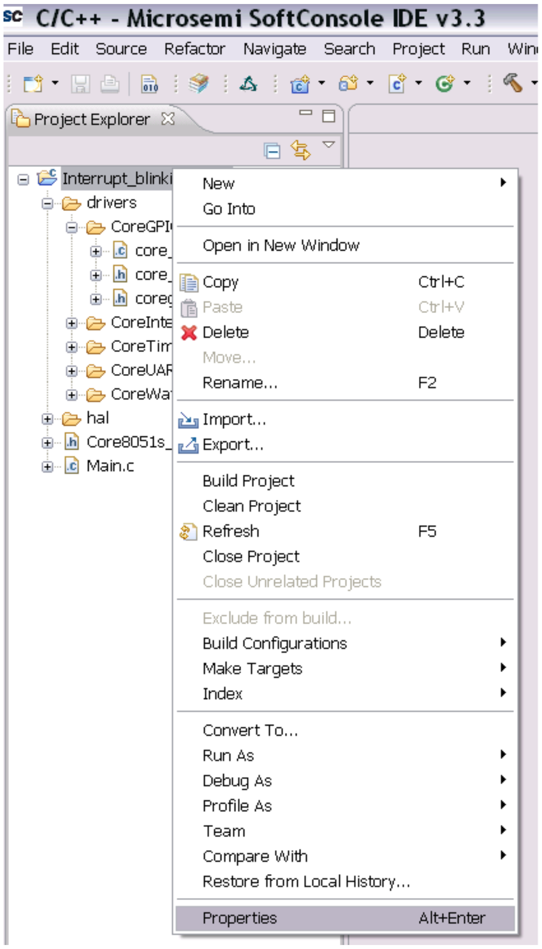
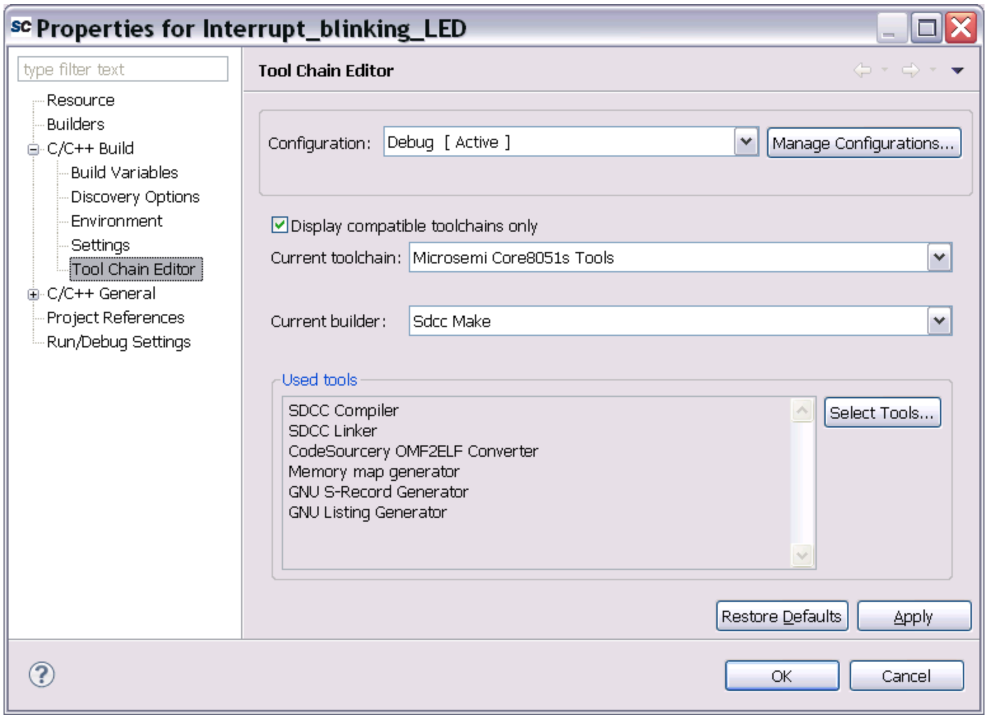


Figure 29 Setting the Project Properties

3. Navigate to **C/C++ Build → Tool Chain Editor**, as shown in Figure 30.



Properties for Interrupt_blinking_LED

Tool Chain Editor

Configuration: Debug [Active]

☑ Display compatible toolchains only

Current toolchain: Microsemi Core8051s Tools

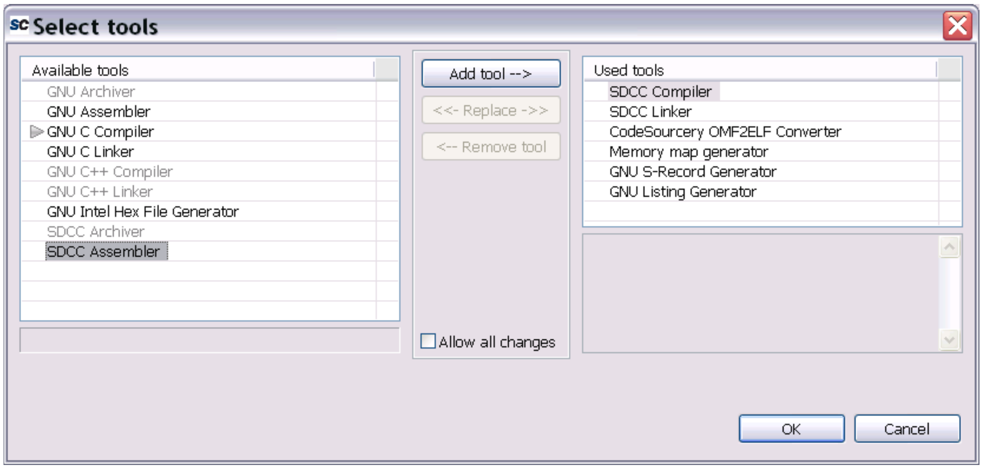
Current builder: Sdcc Make

Used tools

SDCC Compiler
SDCC Linker
CodeSourcery OMF2ELF Converter
Memory map generator
GNU S-Record Generator
GNU Listing Generator

Figure 30 Tool Chain Editor

4. Click **Select Tools**. The Select tools window is displayed, as shown in Figure 31.



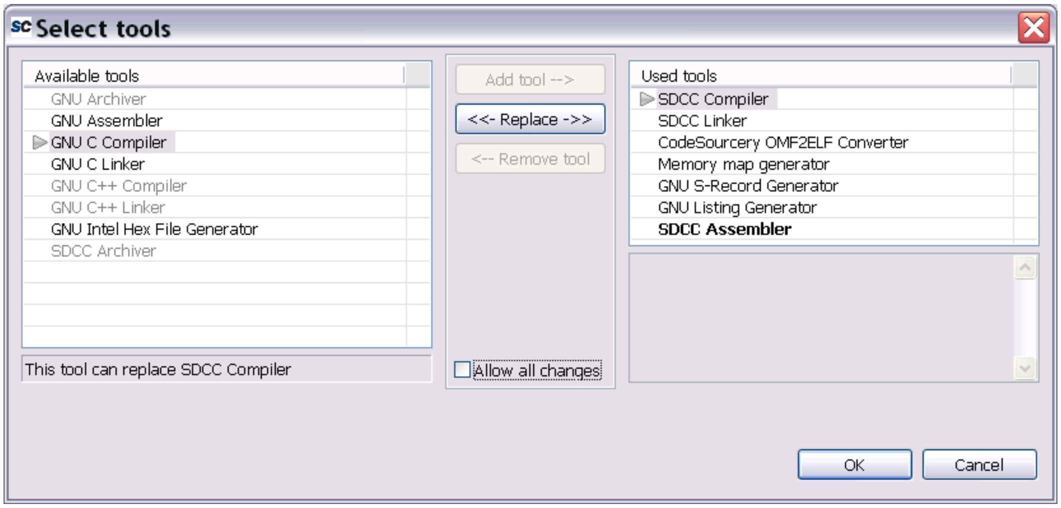
Select tools

Available tools
GNU Archiver
GNU Assembler
GNU C Compiler
GNU C Linker
GNU C++ Compiler
GNU C++ Linker
GNU Intel Hex File Generator
SDCC Archiver
SDCC Assembler

Used tools
SDCC Compiler
SDCC Linker
CodeSourcery OMF2ELF Converter
Memory map generator
GNU S-Record Generator
GNU Listing Generator

Figure 31 Select Tools Window

5. Under Available Tools, select **SDCC Assembler** and click **Add tool -->**, as shown in Figure 32.



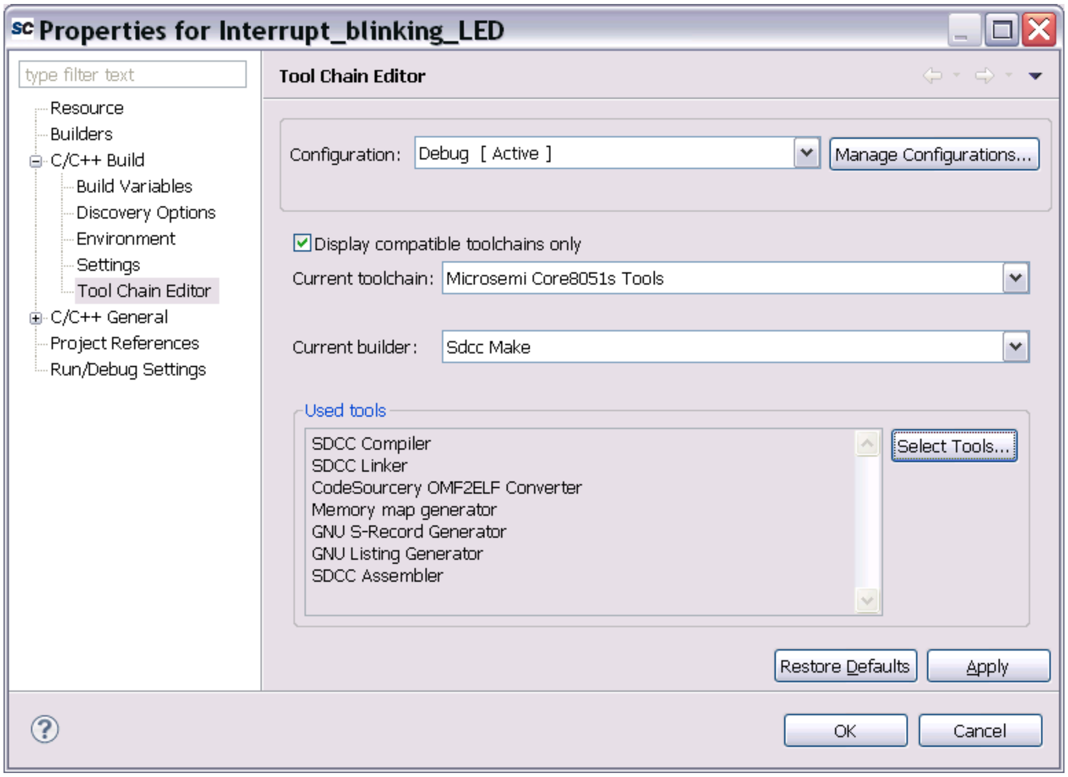
Available tools
GNU Archiver
GNU Assembler
GNU C Compiler
GNU C Linker
GNU C++ Compiler
GNU C++ Linker
GNU Intel Hex File Generator
SDCC Archiver

This tool can replace SDCC Compiler

Used tools
SDCC Compiler
SDCC Linker
CodeSourcery OMF2ELF Converter
Memory map generator
GNU S-Record Generator
GNU Listing Generator
SDCC Assembler

Figure 32 Adding Tools

6. The SDCC Assembler is moved to **Used tools** on the right side. Click **OK**.
7. In the **Tool Chain Editor** window, the SDCC Assembler should be listed under **Used tools**. Click **Apply** and click **OK**, as shown in Figure 33.



Configuration: Debug [Active]

☑ Display compatible toolchains only

Current toolchain: Microsemi Core8051s Tools

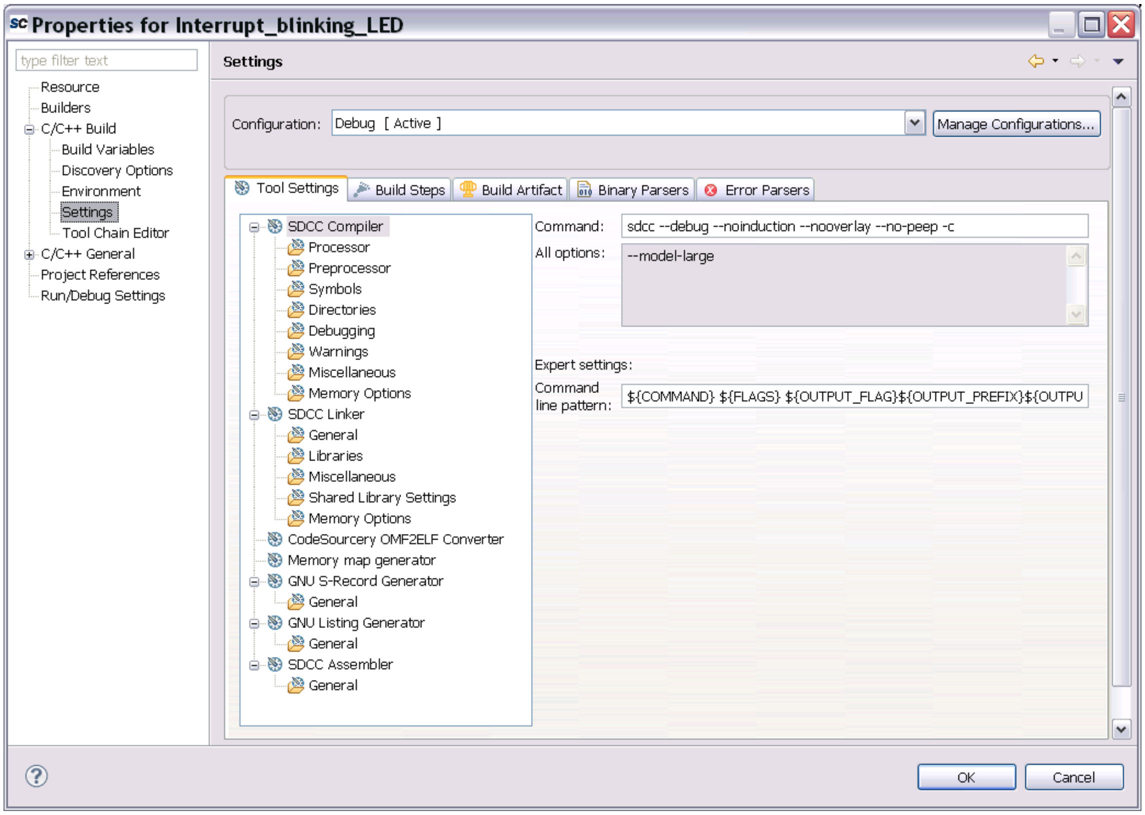
Current builder: Sdcc Make

Used tools
SDCC Compiler
SDCC Linker
CodeSourcery OMF2ELF Converter
Memory map generator
GNU S-Record Generator
GNU Listing Generator
SDCC Assembler

Figure 33 Project Properties

Adding Directories

1. Go to **C/C++ Build → Settings → SDCC Compiler → Directories**, click **Add**. The window is displayed, as shown in Figure 34.



Properties for Interrupt_blinking_LED

Settings

Configuration: Debug [Active]

Command: sdcc --debug --noinduction --nooverlay --no-peep -c

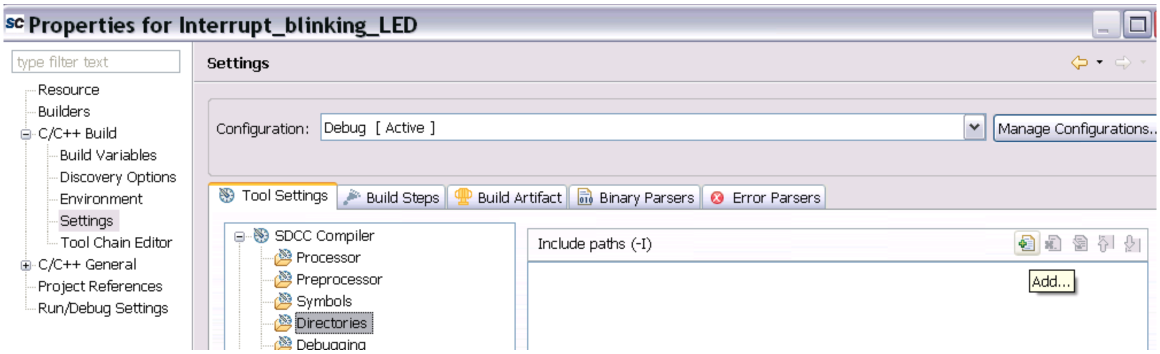
All options: --model-large

Expert settings:

Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPU

Figure 34 Adding Directories

2. Go to the **Directories**, as shown in Figure 35 and click **Add**.



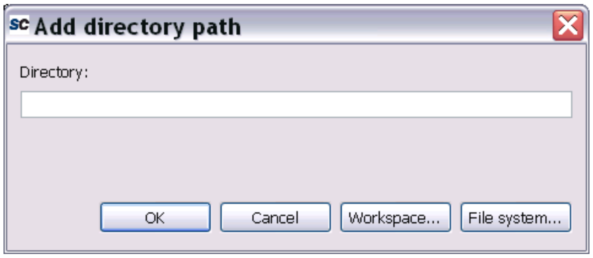
Properties for Interrupt_blinking_LED

Settings

Configuration: Debug [Active]

Figure 35 Add Directories

3. The window is displayed, as shown in Figure 36. Click **Workspace**.



Add directory path

Directory:

Figure 36 Select Workspace

4. Select CoreGPIO and click **OK**, as shown in Figure 37.

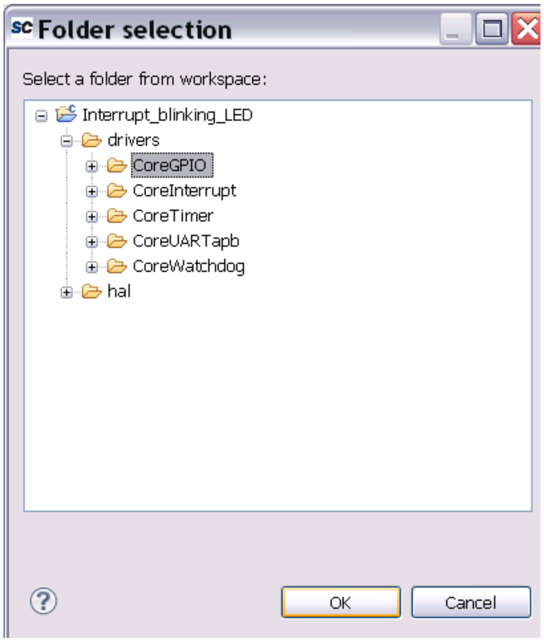
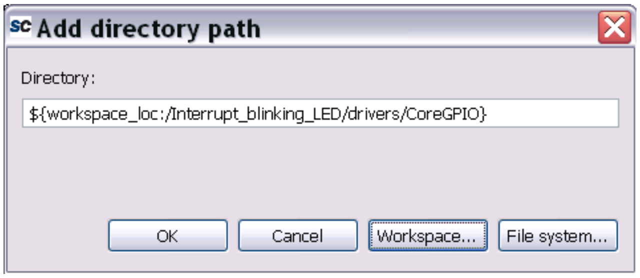


Figure 37 Select CoreGPIO Driver

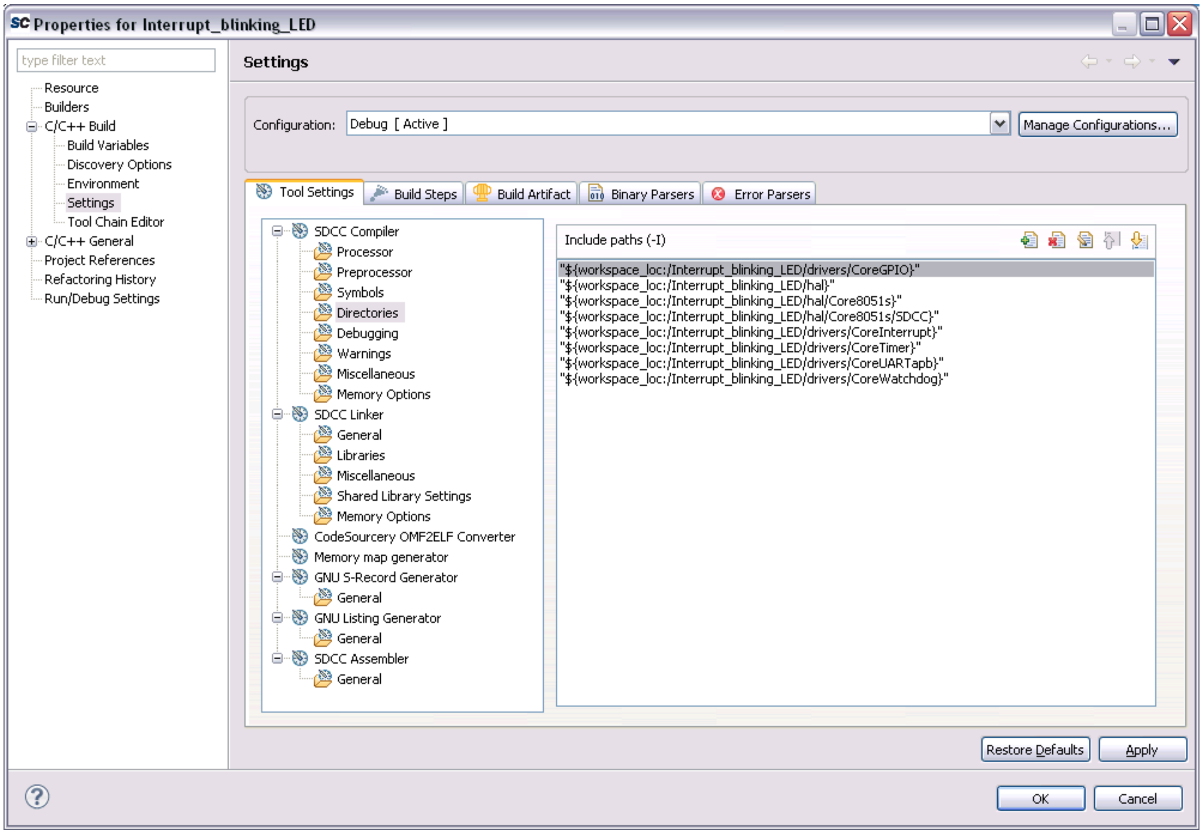
5. The window is displayed as shown in Figure 38. Click **OK**.



Directory:
${workspace_loc:/Interrupt_blinking_LED/drivers/CoreGPIO}

Figure 38 Add Directory Path

6. Similarly add CoreUARTapb, CoreWatchdog, CoreInterrupt, CoreTimer, HAL, Core8051s, and SDCC. Finally the settings window is displayed, as shown in Figure 39.



Configuration: Debug [Active]

Include paths (-I)

```
"${workspace_loc:/Interrupt_blinking_LED/drivers/CoreGPIO}"
"${workspace_loc:/Interrupt_blinking_LED/hal}"
"${workspace_loc:/Interrupt_blinking_LED/hal/Core8051s}"
"${workspace_loc:/Interrupt_blinking_LED/hal/Core8051s/SDCC}"
"${workspace_loc:/Interrupt_blinking_LED/drivers/CoreInterrupt}"
"${workspace_loc:/Interrupt_blinking_LED/drivers/CoreTimer}"
"${workspace_loc:/Interrupt_blinking_LED/drivers/CoreUARTapb}"
"${workspace_loc:/Interrupt_blinking_LED/drivers/CoreWatchdog}"
```

Figure 39 Include Paths

7. The SoftConsole project is ready to build. Right-click the project and select **Build Project**, as shown in Figure 40.

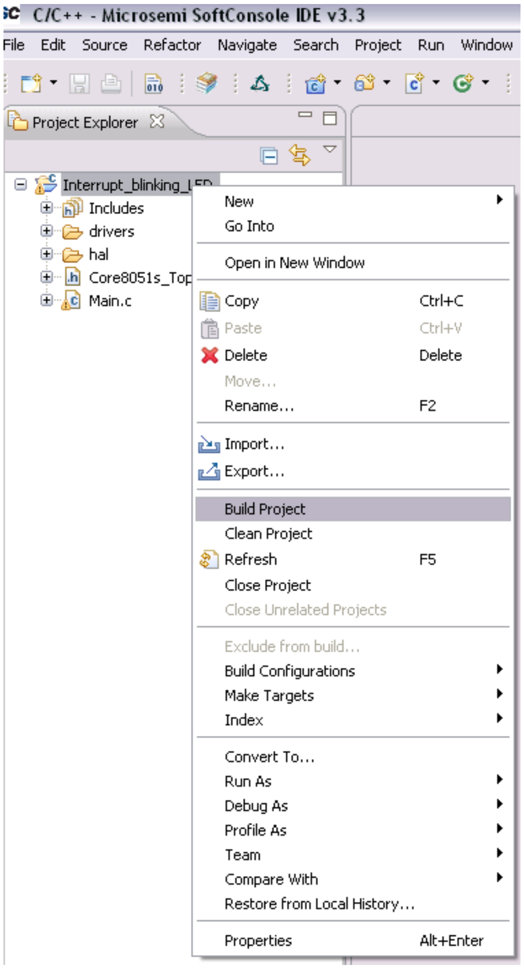
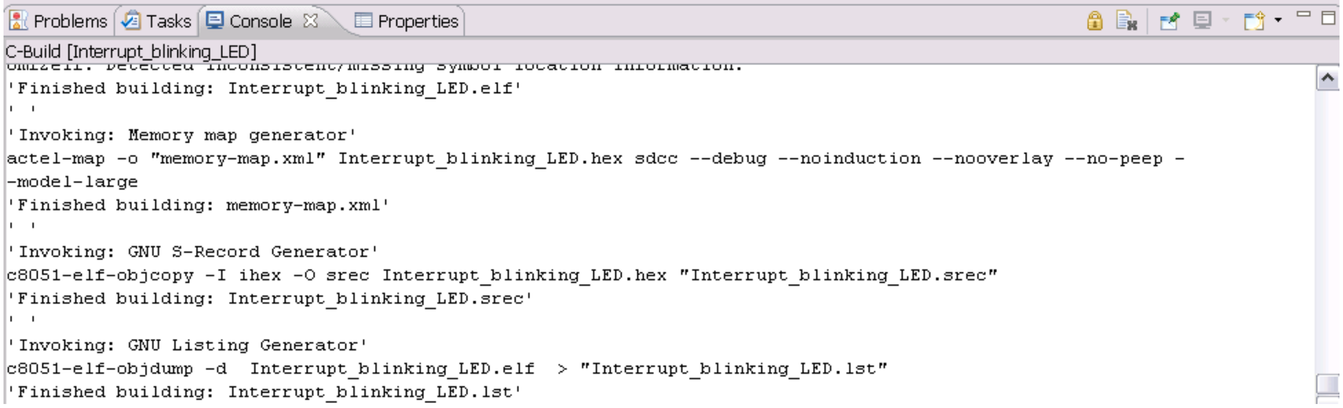


Figure 40 Build Project

8. The **Console** tab is displayed with no errors, as shown in Figure 41.



C-Build [Interrupt_blinking_LED]

```
omizer: Detected inconsistent/missing symbol location information.
'Finished building: Interrupt_blinking_LED.elf'
' '
'Invoking: Memory map generator'
actel-map -o "memory-map.xml" Interrupt_blinking_LED.hex sdcc --debug --noinduction --nooverlay --no-peep -
-model-large
'Finished building: memory-map.xml'
' '
'Invoking: GNU S-Record Generator'
c8051-elf-objcopy -I ihex -O srec Interrupt_blinking_LED.hex "Interrupt_blinking_LED.srec"
'Finished building: Interrupt_blinking_LED.srec'
' '
'Invoking: GNU Listing Generator'
c8051-elf-objdump -d  Interrupt_blinking_LED.elf  > "Interrupt_blinking_LED.lst"
'Finished building: Interrupt_blinking_LED.lst'
```

Figure 41 The Console Tab

Step 6 - Configuring Serial Terminal Emulation Program

1. Go to **Start** and select **HyperTerminal**, as shown in Figure 42.

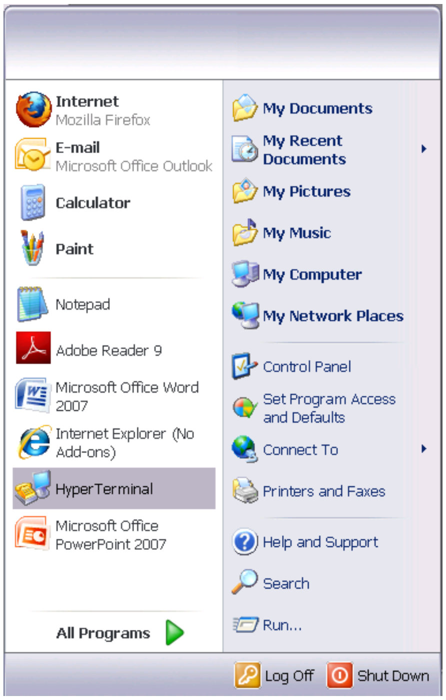
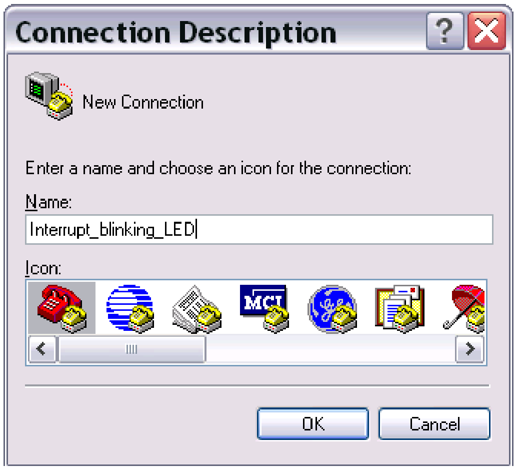


Figure 42 Selecting HyperTerminal

Note: Refer to www.microsemi.com/soc/documents/Configuring_Serial_Terminal_Emulation_Programs.pdf for other versions as Tera Term Pro or PuTTY.

2. Name the connection, as shown in Figure 43. Click **OK**.



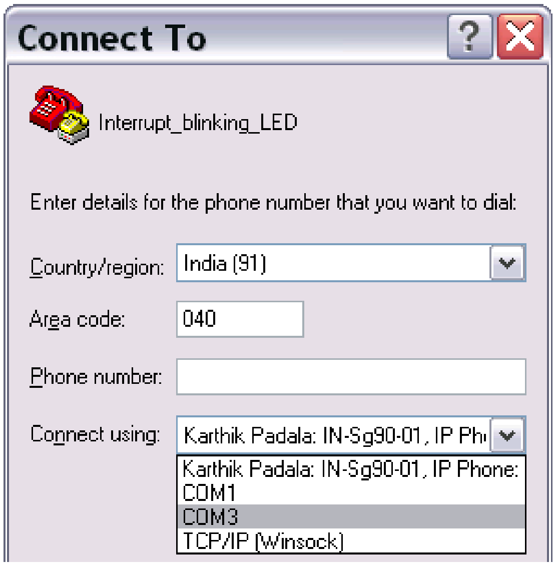
Enter a name and choose an icon for the connection:

Name: Interrupt_blinking_LED

Figure 43 Connection Description

Note: Select the COM port to which **Silicon labs CP210x USB to UART Bridge** is connected. This is accessed by using **My Computer Properties→ Hardware → Device Manager**.

3. Select **COM3**, as shown in Figure 44.



Connect To

Interrupt_blinking_LED

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

Country/region: India (91)

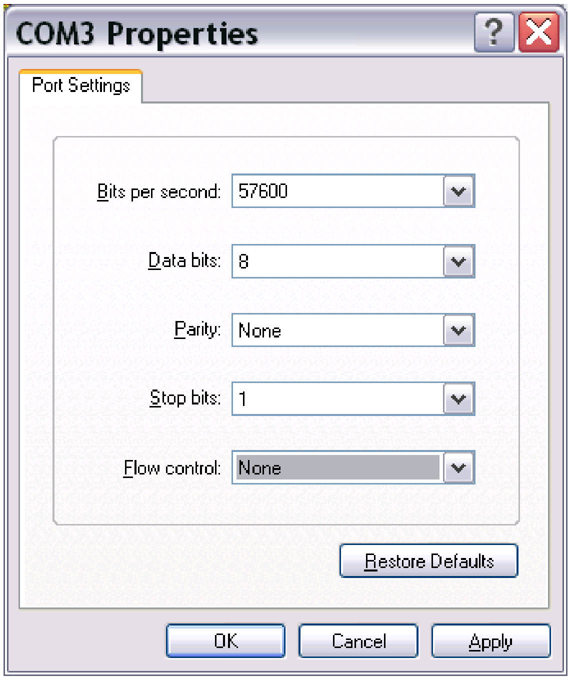
Area code: 040

Phone number:

Connect using: Karthik Padala: IN-Sg90-01, IP Ph

Figure 44 Selecting COM Port

4. Give the port settings, as shown in Figure 45 and click **OK**.



COM3 Properties

Port Settings

Bits per second: 57600

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

Figure 45 Port Settings

5. The HyperTerminal with name Interrupt_blinking_LED is displayed, as shown in Figure 46.

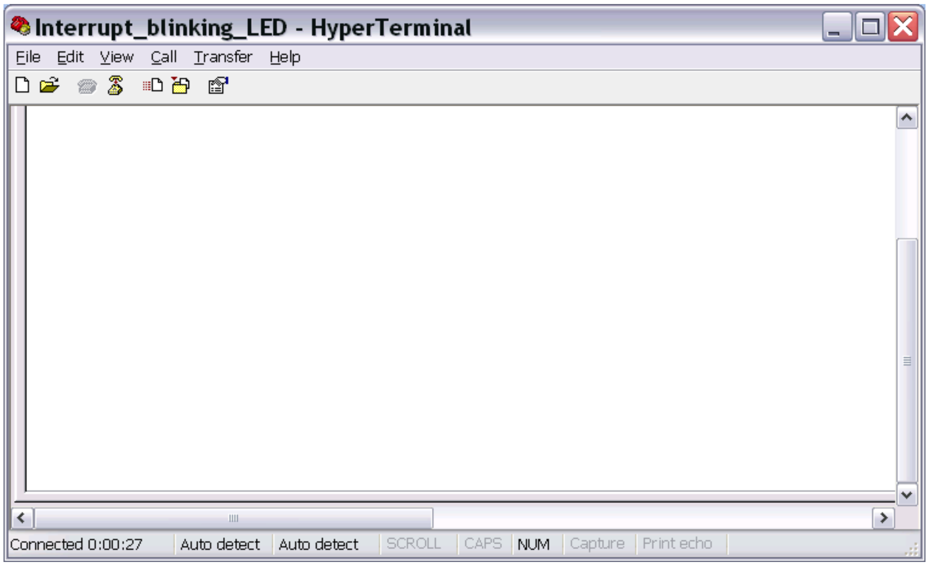

Interrupt_blinking_LED - HyperTerminal
File Edit View Call Transfer Help
Connected 0:00:27
Auto detect
Auto detect
SCROLL
CAPS
NUM
Capture
Print echo

Figure 46 HyperTerminal

Step 7 - Debugging the Application Project using SoftConsole

1. Right-click **Interrupt_blinking_LED** and Launch a debug session, as shown in Figure 47.

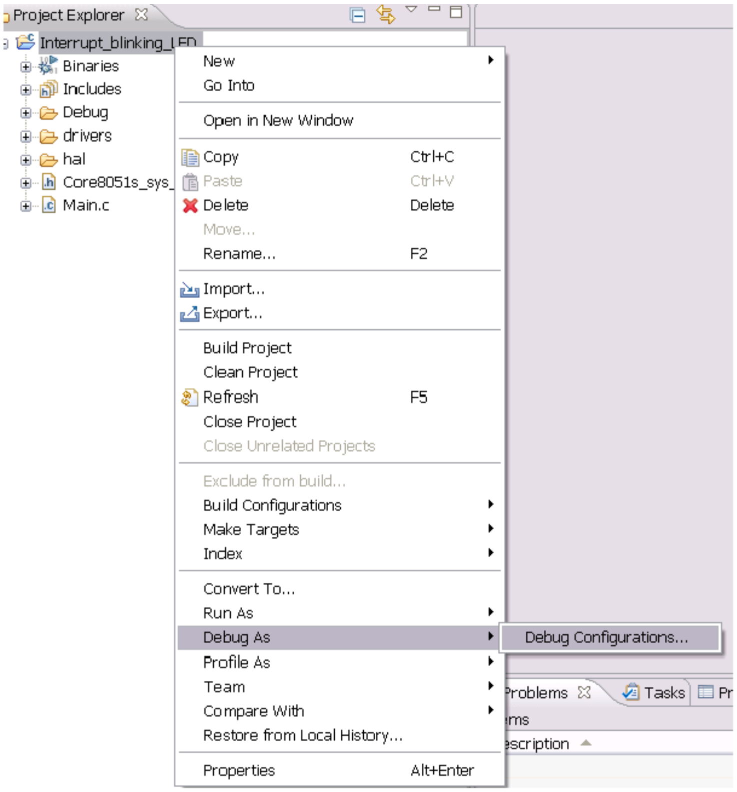
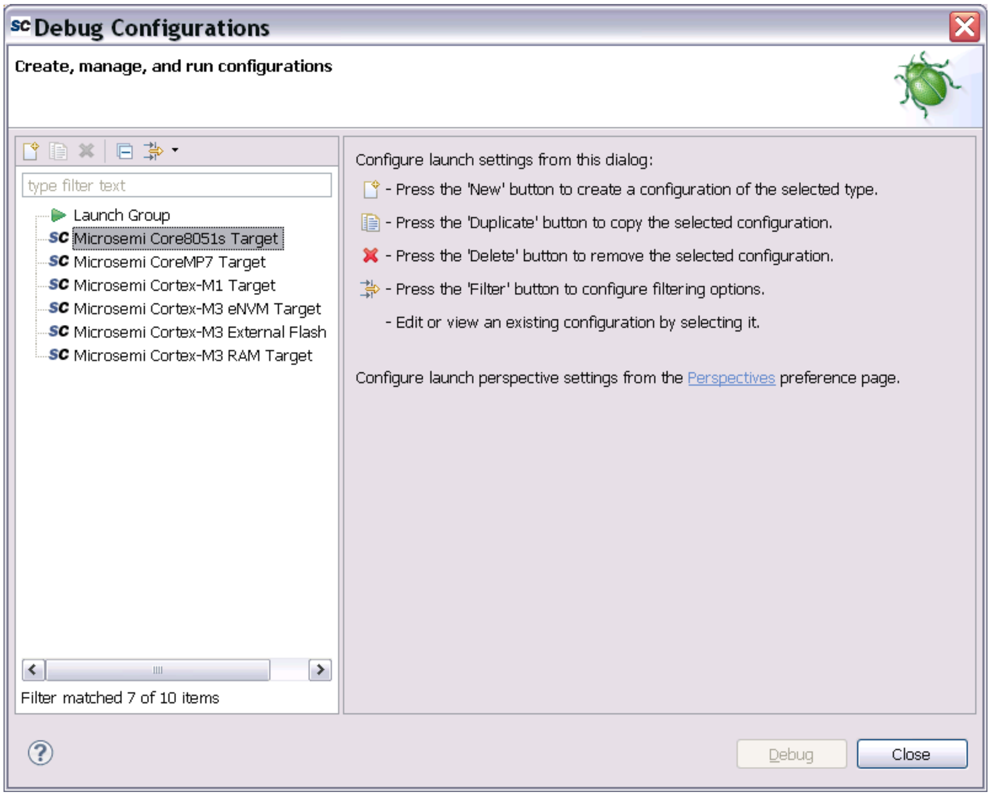


Figure 47 Selecting Debug Configurations

2. Right-click **Microsemi Core8051s Target** and select **New**, as shown in Figure 48.



Debug Configurations

Create, manage, and run configurations

type filter text

- Launch Group
- Microsemi Core8051s Target
- Microsemi CoreMP7 Target
- Microsemi Cortex-M1 Target
- Microsemi Cortex-M3 eNVM Target
- Microsemi Cortex-M3 External Flash
- Microsemi Cortex-M3 RAM Target

Filter matched 7 of 10 items

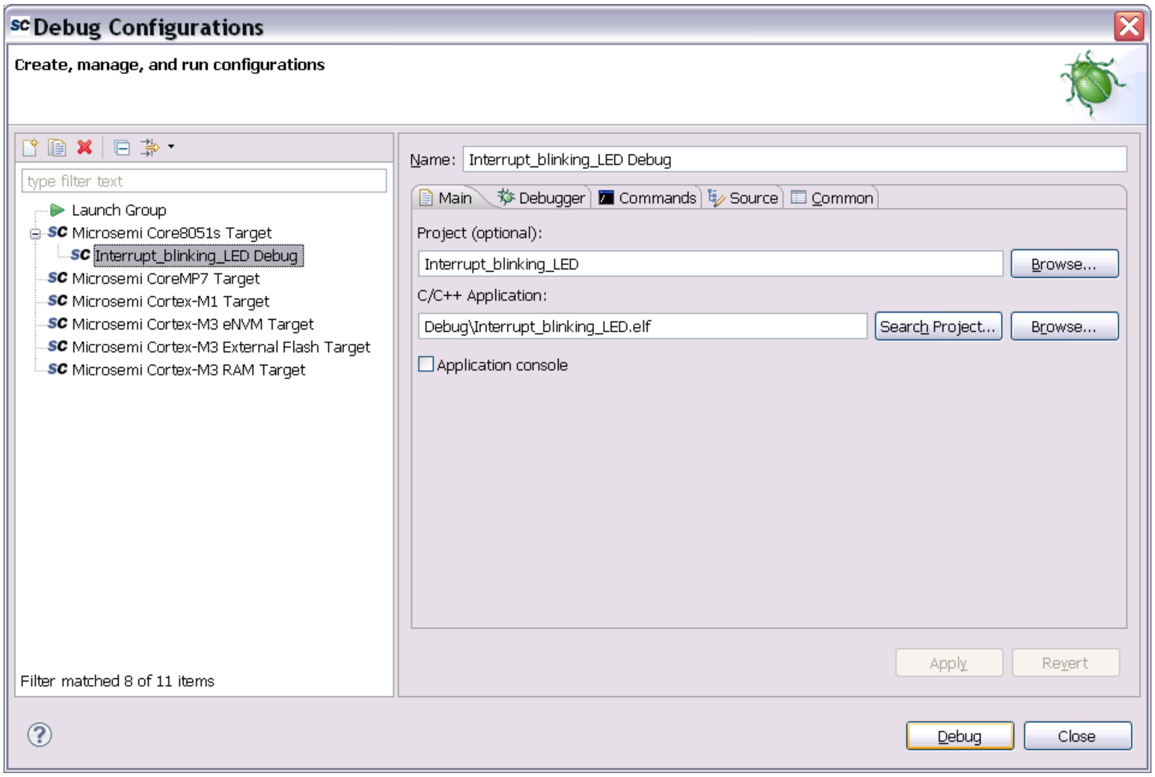
Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

Figure 48 Create Debug Configuration

3. The Debug Configuration window is displayed, as shown in Figure 49. Click **Debug**.



Debug Configurations

Create, manage, and run configurations

type filter text

- Launch Group
- Microsemi Core8051s Target
  - Interrupt_blinking_LED Debug
- Microsemi CoreMP7 Target
- Microsemi Cortex-M1 Target
- Microsemi Cortex-M3 eNVM Target
- Microsemi Cortex-M3 External Flash Target
- Microsemi Cortex-M3 RAM Target

Filter matched 8 of 11 items

Name: Interrupt_blinking_LED Debug

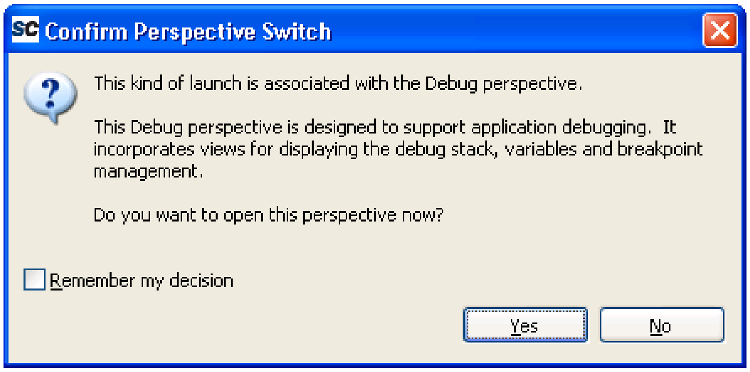
Project (optional): Interrupt_blinking_LED

C/C++ Application: Debug\Interrupt_blinking_LED.elf

□ Application console

Figure 49 Launching Debug Session

4. Click **Yes** to confirm, as shown in Figure 50.



Confirm Perspective Switch

This kind of launch is associated with the Debug perspective.

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

Do you want to open this perspective now?

□ Remember my decision

Figure 50 Confirm Perspective Switch

5. From **Run** menu select **Resume**, as shown in Figure 51.

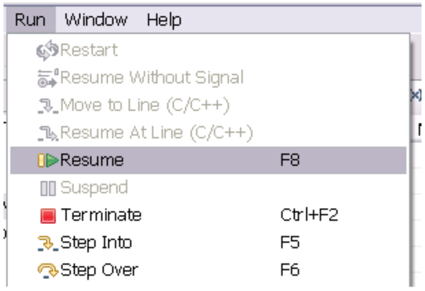
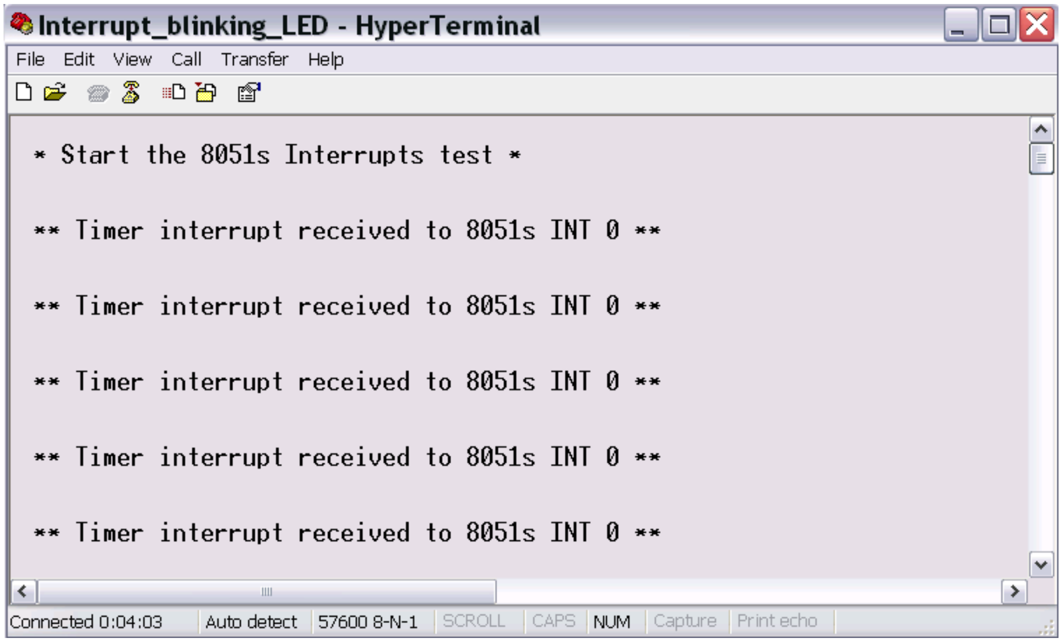


Figure 51 Select Resume

6. You can see the LED glowing at the specified time interval on the Fusion Advanced Development Kit. The HyperTerminal messages are displayed as shown in Figure 52.



Interrupt_blinking_LED - HyperTerminal

```
* Start the 8051s Interrupts test *

** Timer interrupt received to 8051s INT 0 **

** Timer interrupt received to 8051s INT 0 **

** Timer interrupt received to 8051s INT 0 **

** Timer interrupt received to 8051s INT 0 **

** Timer interrupt received to 8051s INT 0 **
```

Connected 0:04:03 | Auto detect | 57600 8-N-1

Figure 52 Application Running

Conclusion

In this tutorial you have created a simple application program for Core8051s IP-based Embedded Processor System using Microsemi Tools. This design can serve as a starting point to other Core8051s IP-based designs.

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

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

Website

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

Contacting the Customer Technical Support Center

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

Email

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

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

My Cases

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

Outside the U.S.

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

ITAR Technical Support

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

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

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