Rate This Document

Core8051s Embedded Processor Software Development Tutorial

for Fusion Mixed-Signal FPGAs

Actel®
POWER MATTERS

Actel Corporation, Mountain View, CA 94043

© 2009 Actel Corporation. All rights reserved.

Printed in the United States of America

Part Number: 50200153-1

Release: March 2009

Table of Contents

Introduction

This tutorial shows how to develop a simple application program for an 8051-based embedded processor system using Actel tools. This design is suitable as a starting point for developing an embedded system. It is assumed that the reader is familiar with the C programming language. After completing this tutorial you will be familiar with the software design process for creating an 8051-based embedded system using SoftConsole. This includes the following steps:

1. Configuring the SoftConsole complier/linker settings
2. Compiling your code
3. Debugging your code
4. Programming your code into the FPGA

Requirements for Tutorial

This tutorial requires that SoftConsole and FlashPro have been installed on your computer. FlashPro is often installed as part of the Actel Libero® Integrated Design Environment (IDE) installation and can be launched from within Libero IDE or standalone.

Supported Development Boards

This tutorial is designed to support the following three development board designs:

- ARM® Cortex™-M1–Enabled Fusion Development KIT (M1AFS-DEV-KIT-SCS) board
- Fusion Embedded Development KIT (M1ASF-EMBEDDED-KIT) board
- Fusion Advanced Development KIT (M1AFS-ADV-DEV-KIT) board

In general, most tutorial steps apply to all three target boards. Steps which are specific to a given target board are clearly indicated.

Programming Requirements

Programming of the application code in the target application requires one of the target boards and a Flashpro3 debugger/programmer.

The M1AFS-DEV-KIT-SCS-SA board includes a FlashPro3 in the board design.

The M1ASF-EMBEDDED-KIT and the M1AFS-ADV-DEV-KIT are supplied with an LC Programmer board which contains the FlashPro3 debugger/programmer.

Software Versions

This tutorial uses the following software versions:

- SoftConsole version 2.2
- FlashPro version 8.5

Tutorial Firmware Source Code

The C compiler and linker used by SoftConsole needs information about the hardware resources used in your project.

This includes information about the memory map for code memory, external data memory, and the peripherals. This information is included in the header files for this tutorial.

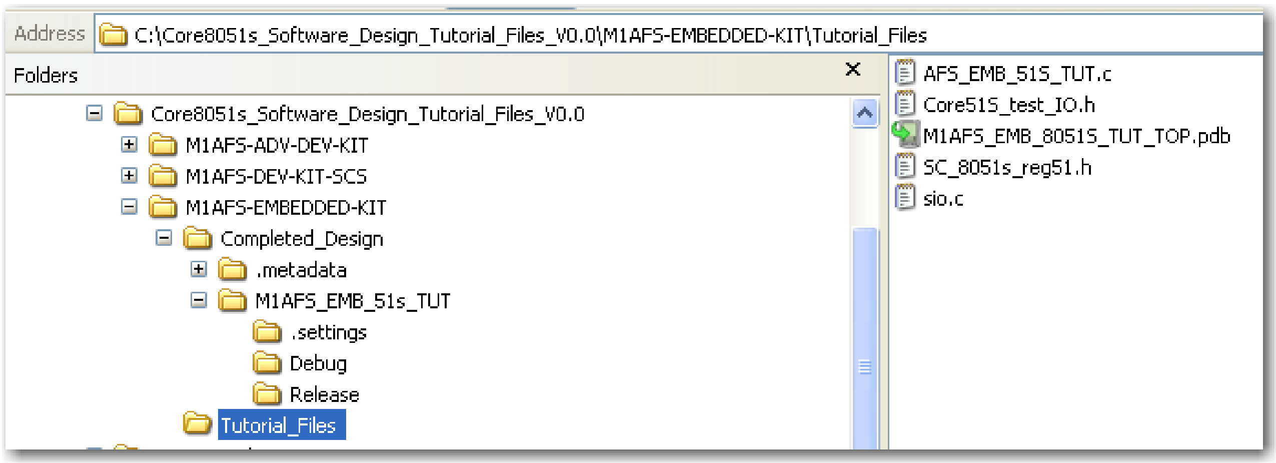
Source code and header files used in this tutorial may be downloaded from the Actel website:

http://www.actel.com/documents/Core8051s_SW_DesignTutorial_DF.zip.

Note: This file should be unzipped in a root directory (e.g., c:/ or d:/) of your hard drive. If you unzip the file to a directory path that is too long, you will be asked for a password and you will not be able to extract the files.

This folder contains the *Core8051s_Software_Design_Tutorial_Files* folder. This folder contains three folders (M1AFS…), one for each targeted board (Figure 1). Each board folder contains a *Completed_Design* folder and a *Tutorial_Files* folder.

The *Completed_Design* folder is a SoftConsole workspace containing a complete project using the same code that will be used in this tutorial. The *Tutorial_Files* folder contains source code and header files that you will use in this tutorial. This folder also contains a M1AFS…TUT_TOP.pdb file that is used for programming the FPGA fabric (the exact file name depends on the target board). Further references to the *Tutorial_Files* folder refer to the folder specific to your board.



```
Address  C:\Core8051s_Software_Design_Tutorial_Files_V0.0\M1AFS-EMBEDDED-KIT\Tutorial_Files

Core8051s_Software_Design_Tutorial_Files_V0.0
  M1AFS-ADV-DEV-KIT
  M1AFS-DEV-KIT-SCS
  M1AFS-EMBEDDED-KIT
    Completed_Design
      .metadata
      M1AFS_EMB_51s_TUT
        .settings
        Debug
        Release
    Tutorial_Files

AFS_EMB_51S_TUT.c
Core51S_test_IO.h
M1AFS_EMB_8051S_TUT_TOP.pdb
SC_8051s_reg51.h
sio.c
```

Figure 1 · Zip File Contents

Software Project

This chapter describes how to create and build the software project, preparing both the debug and release version of the output files.

Description of Code

This demo code initializes the serial port, the analog quads, and the ADC. It then prints a sign-on message through the serial port and prompts the user to hit any key. The code then sits in a loop waiting for a key to be pressed.

M1AFS-DEV-KIT-SCS-SA Board

If a key is pressed, the code reads and displays the following information:

- The 3.3 V supply voltage (using AV6)
- The 3.3 V supply current (using AV6 and AC6)
- The 1.5 V supply voltage (using AV7)
- The 1.5 V supply current (using AV7 and AC7)
- The ambient temperature (using an NPN transistor and AT8)
- The 2.5 V supply voltage (using AV9)
- The 2.5 V supply current (using AV9 and AC9)
- The ambient temperature (using an NPN transistor and AT9)
- The wiper voltage of potentiometer, RV1 (using AV8)

M1AFS-EMBEDDED-KIT and M1AFS-ADV-DEV-KIT Boards

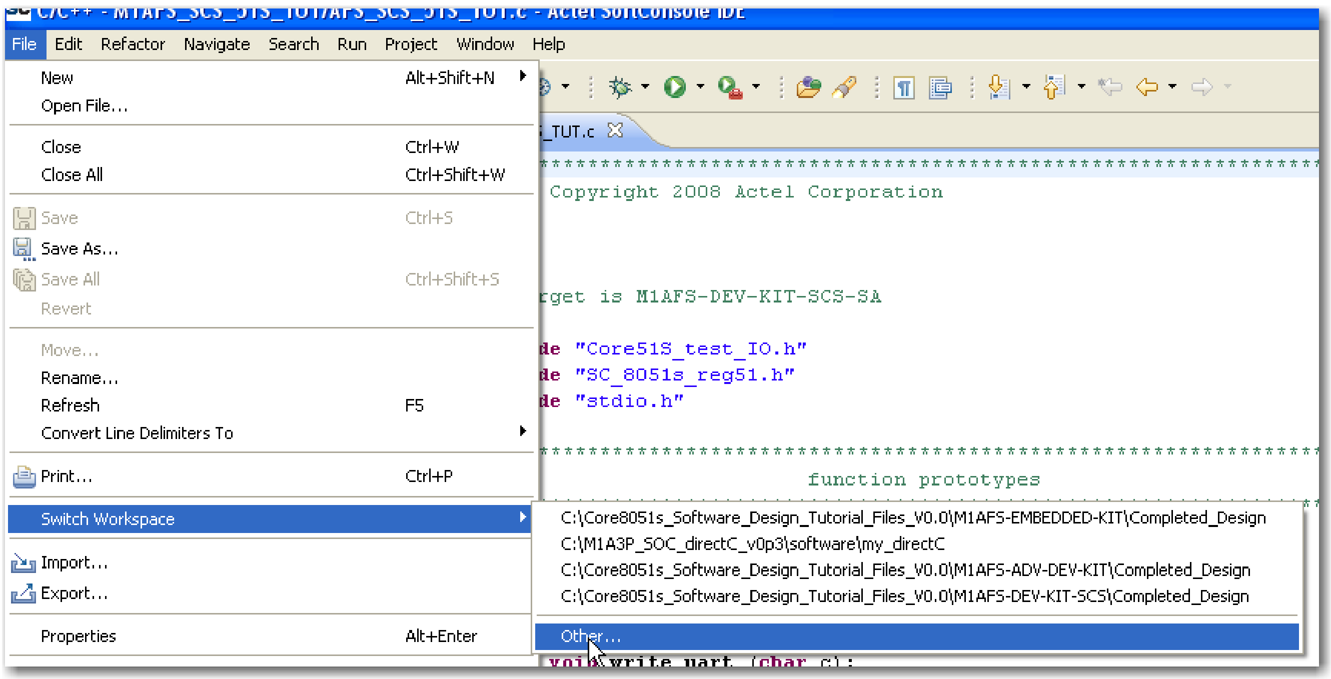
These boards use the same firmware. If a key is pressed, the code reads and displays the following information for the board:

- The 3.3 V supply voltage (using AV0)
- The 3.3 V supply current (using AV0 and AC0)
- The 1.5 V supply voltage (using AV1)
- The 1.5 V supply current (using AV1 and AC1)
- The ambient temperature (using an NPN transistor and AT2)
- The wiper voltage of potentiometer, RV1 (using AC4)

Creating the SoftConsole Project

SoftConsole organizes files into projects. Each project exists within a workspace. A workspace can contain more than one project and can be located anywhere within the file system. Actel highly recommends that you allow SoftConsole to organize your project for you by using a Managed Make Project. Each time you launch SoftConsole, you might be prompted to select a workspace, depending on how SoftConsole has been previously configured. In such a case, browse and select or create a workspace to use.

If not prompted, you can either use an existing workspace or use the **File** > **Switch Workspace** command to select or create a new workspace (Figure 1-1).


C/C++ - M1AFS_SCS_51S_TUT/AFS_SCS_51S_TUT.c - Actel SoftConsole IDE
File
Edit
Refactor
Navigate
Search
Run
Project
Window
Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename...
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Import...
Export...
Properties Alt+Enter
C:\Core8051s_Software_Design_Tutorial_Files_V0.0\M1AFS-EMBEDDED-KIT\Completed_Design
C:\M1A3P_SOC_directC_v0p3\software\my_directC
C:\Core8051s_Software_Design_Tutorial_Files_V0.0\M1AFS-ADV-DEV-KIT\Completed_Design
C:\Core8051s_Software_Design_Tutorial_Files_V0.0\M1AFS-DEV-KIT-SCS\Completed_Design
Other...
_TUT.c

```
*****************************************************************
 Copyright 2008 Actel Corporation


rget is M1AFS-DEV-KIT-SCS-SA

de "Core51S_test_IO.h"
de "SC_8051s_reg51.h"
de "stdio.h"

*****************************************************************
                                function prototypes
*****************************************************************
void write_uart (char c);
```

Figure 1-1 · Switch Workspace

Create a new workspace. You can choose any name for this workspace. You might want to choose a name that reminds you that the workspace folder is a workspace, such as my_workspace.

In this tutorial you will build the software project that is identical to the project in the *Completed_Design* folder for your board, following the same file-naming conventions used for the completed design (see "Tutorial Firmware Source Code" on page 5).

1. From the menu choose **File** > **New** > **C Project** (Figure 1-2 on page 9). The C Project window will open.
   - If your target is the M1AFS-DEV-KIT-SCS board, name the project M1AFS_SCS_51S_TUT in the Project name box.
   - If your target is the M1AFS-EMBEDDED-KIT board, name the project M1AFS_EMB_51S_TUT.
   - If your target is the M1AFS-ADV-DEV-KIT board, name the project M1AFS_ADV_51S_TUT.

2. Choose **Executable (Managed Make)** for the project type.

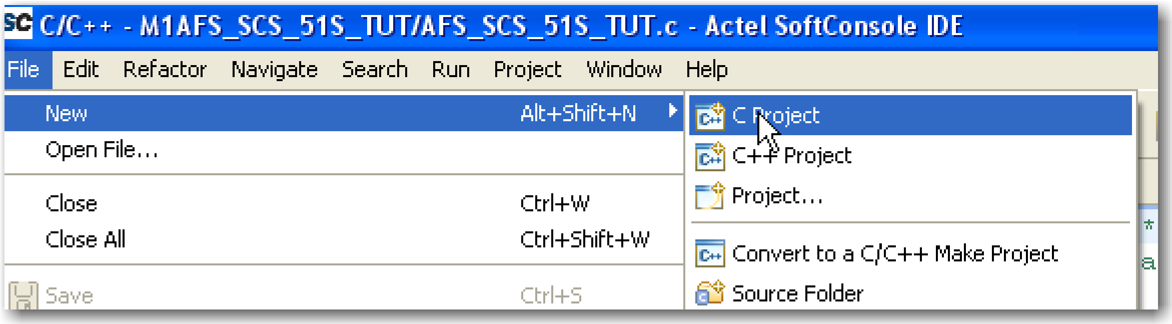
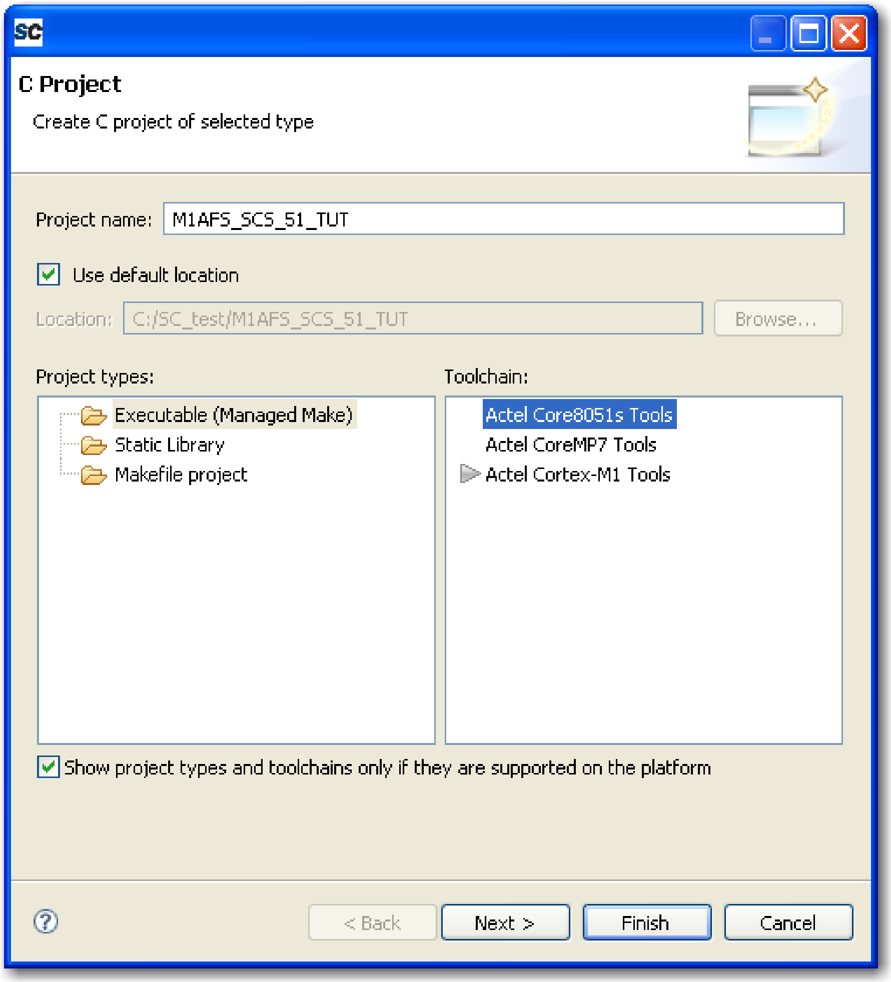

SC C/C++ - M1AFS_SCS_51S_TUT/AFS_SCS_51S_TUT.c - Actel SoftConsole IDE
File Edit Refactor Navigate Search Run Project Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
C Project
C++ Project
Project...
Convert to a C/C++ Make Project
Source Folder

Figure 1-2 · Selecting Project Type

3. Select **Actel Core8051s Tools** for the toolchain. Click **Finish** (Figure 1-3).



C Project

Create C project of selected type

Project name: M1AFS_SCS_51_TUT

Use default location ✓

Location: C:/SC_test/M1AFS_SCS_51_TUT

Project types: Executable (Managed Make)

Toolchain: Actel Core8051s Tools

Figure 1-3 · Selecting Target Device

The name of your project will now show in the Projects window. Some include files needed by the compiler will automatically be included in your project under the *Includes* subdirectory. At this point there are no user specified files. You can create your own C source files or header files using the text editor built into SoftConsole by using the **File** >**New** command. You can also import existing source or header files and include them in your project.

For this tutorial, you will import source files for the project.

Importing and Creating Source Files

You can copy existing source or header files and include them in your project.

1. Choose **File** > **Import** from the main menu. The Import dialog box will open. Expand the General item and select **File System** (Figure 1-4).

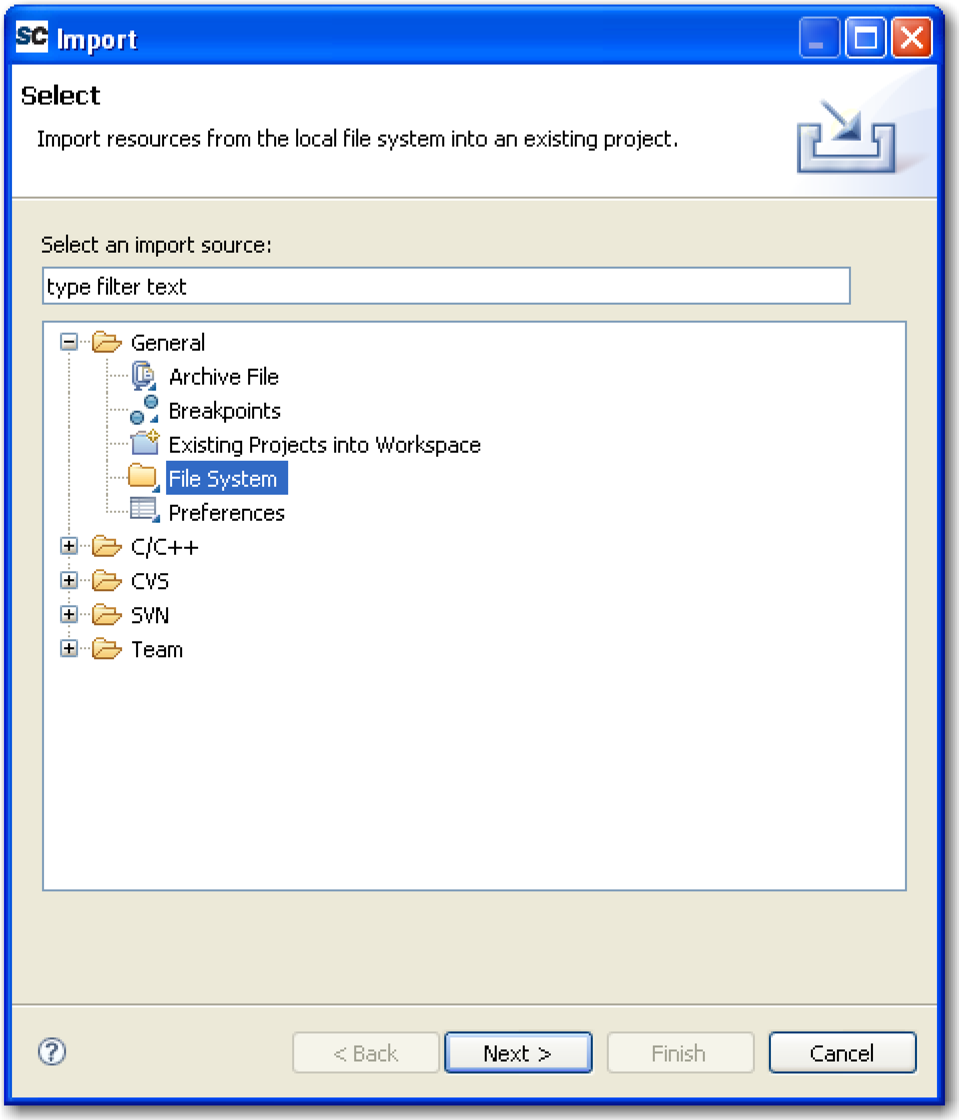


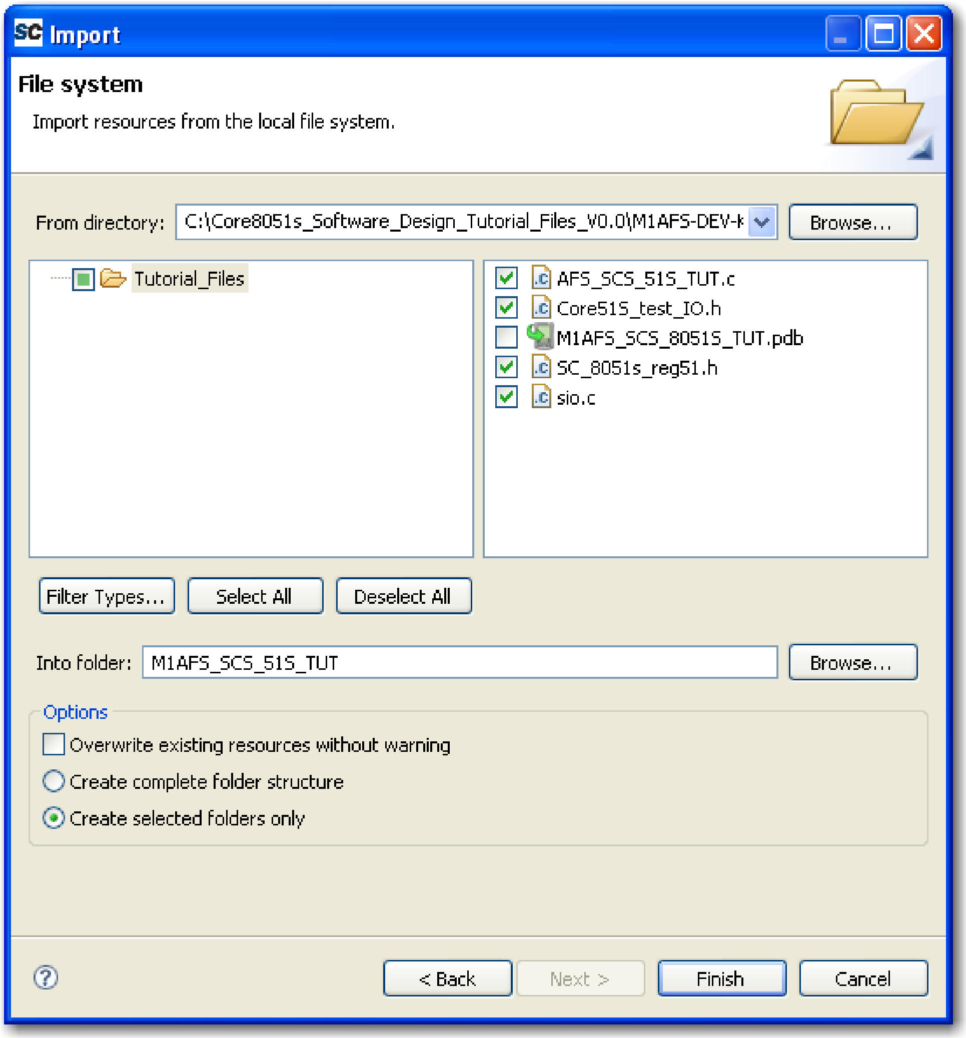
Figure 1-4 · Importing Files – Select File System

2. Click **Next**. Browse to the directory containing the files you want to import. Use the Browse button for the From directory.
3. Click **OK**. The available files will be shown.
4. Check all of the header (*.h) and source (*.c) files. Click the **Browse** button for the Into folder.
5. Click the project name in the Import into Folder box. Click **OK**. Make sure that under option, the **Create selected folders only** option is selected (Figure 1-5).
6. Click **Finish**. The files will then appear in your project. You can expand the project in the C\C++ Projects window by clicking the plus [+] sign and viewing the imported source files.

   If your target is the M1AFS-DEV-KIT-SCS board, the application source file is AFS_SCS_51S_TUT.C.

   If your target is the M1AFS-EMBEDDED-KIT board, the application source file is AFS_EMB_51S_TUT.C.

   If your target is the M1AFS-ADV-DEV-KIT board, the application source file is AFS_ADV_51S_TUT.C.
7. Click the application source file to open the file in the editor window.



From directory: C:\Core8051s_Software_Design_Tutorial_Files_V0.0\M1AFS-DEV-K

- [✓] Tutorial_Files

- [✓] AFS_SCS_51S_TUT.c
- [✓] Core51S_test_IO.h
- [ ] M1AFS_SCS_8051S_TUT.pdb
- [✓] SC_8051s_reg51.h
- [✓] sio.c

Into folder: M1AFS_SCS_51S_TUT

Options
- [ ] Overwrite existing resources without warning
- ○ Create complete folder structure
- [✓ Create selected folders only]

Figure 1-5 · Importing the Files from a Project

Memory Models

Storage Classes

In addition to the ANSI storage classes, SoftConsole allows the following MCS51-specific storage classes:

- data – Variables declared with this storage class are allocated in the directly addressable portion of the internal RAM.
- idata – Variables declared with this storage class are allocated into the indirectly addressable portion of the internal RAM.
- xdata – Variables declared with this storage class are placed in the external data RAM.
- code – Variables declared with this storage class are placed in the code memory.
- bit – This is a data type and a storage class specifier. When a variable is declared as a bit, it is allocated into the bit addressable memory. The bit addressable memory consists of 128 bits which are located from 0x20 to 0x2F in data memory, in addition to bits in specific SFRs.

SoftConsole Memory Models

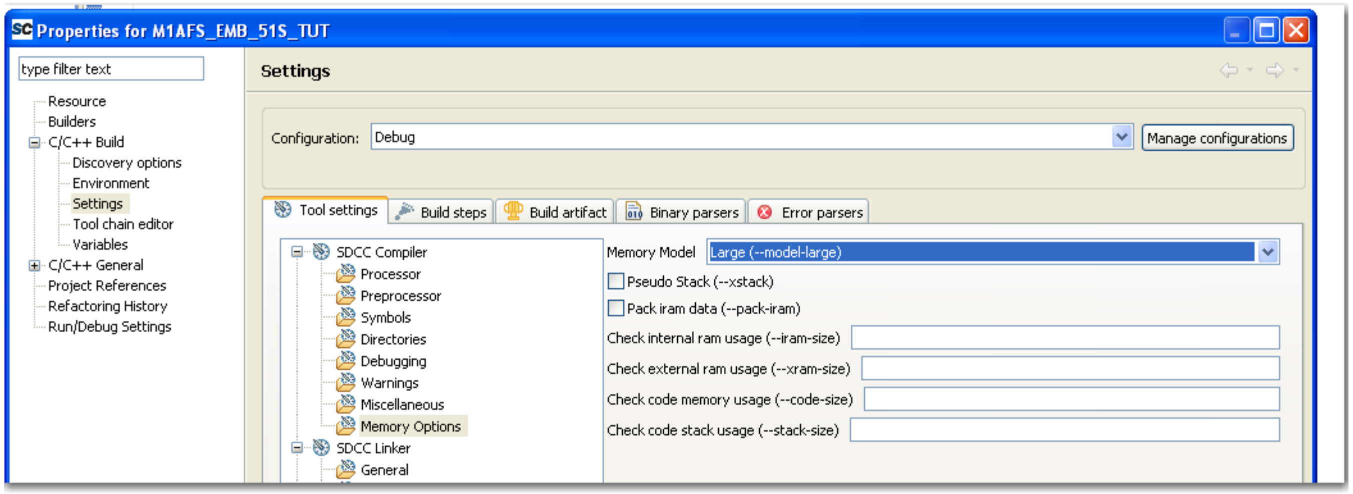
SoftConsole supports two memory models:

- Large – When no storage class is specified, a variable will have a default storage class of xdata.
- Small – When no storage class is specified, a variable will have a default storage class of data.

Thus all variables declared without a storage class will be allocated into external RAM for the large model and into internal RAM for the small model.

Selecting Memory Model

1. Select the memory model by right-clicking on the project name and selecting **Properties** from the pull-down menu. Expand C/C++ Build. Click Settings. Click on **Memory Options** under SDCC Compiler. In the Memory Model box, select **Large (--model-large)** for this tutorial project (Figure 1-6).

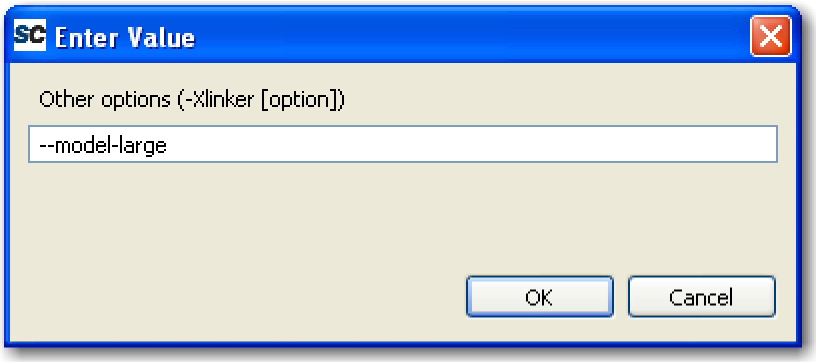


Configuration: Debug

Memory Model: Large (--model-large)
Pseudo Stack (--xstack)
Pack iram data (--pack-iram)
Check internal ram usage (--iram-size)
Check external ram usage (--xram-size)
Check code memory usage (--code-size)
Check code stack usage (--stack-size)

Figure 1-6 · Setting Memory Model for SDCC Compiler Memory Options

2. In the same box, click **Miscellaneous** under SDCC Linker. In the Other Options box, click on the icon with the plus [+] sign. The Enter Value box will open. Type **--model-large** for the large memory model for this tutorial. Click **OK**. (Figure 1-7).

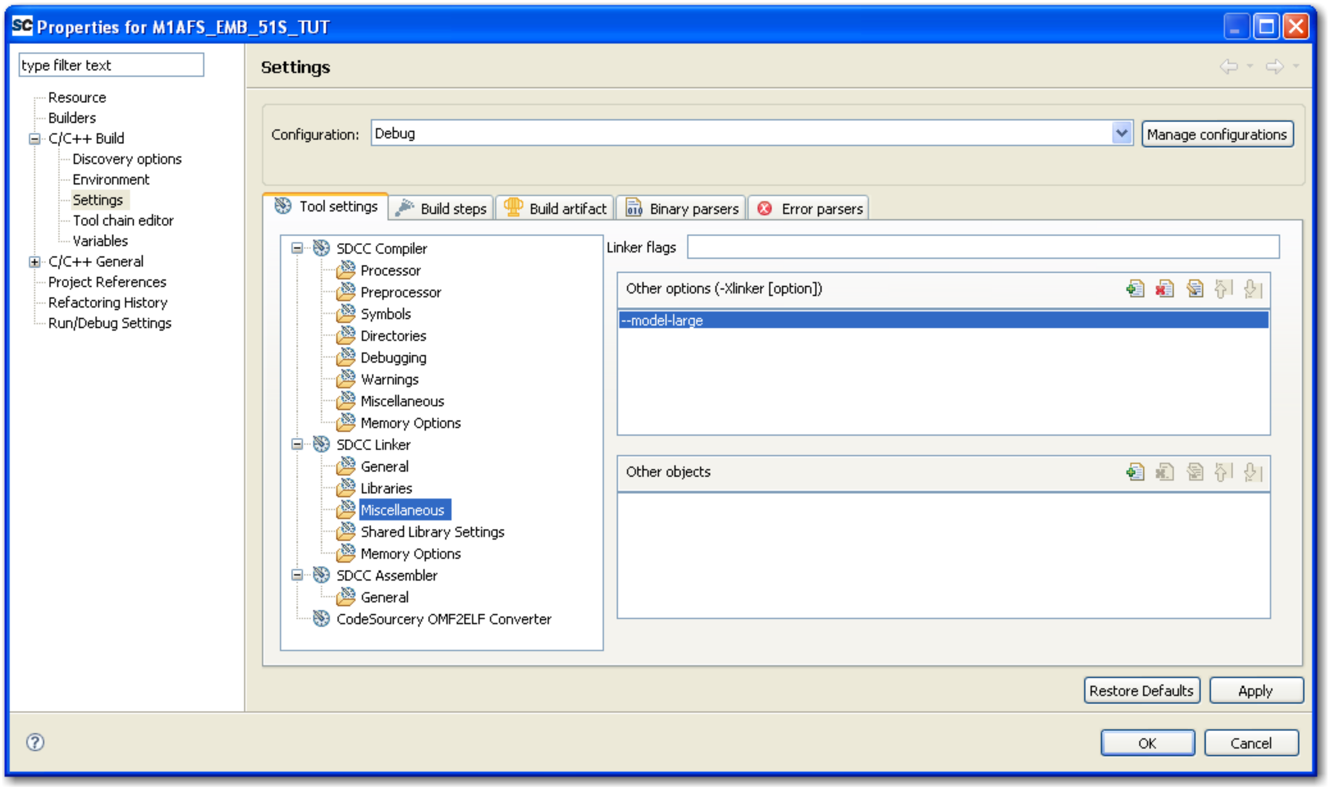


Other options (-Xlinker [option])

--model-large

Figure 1-7 · Setting Linker to Large Memory Model

3. Click **Apply**. Click **OK**.



Configuration: Debug

Linker flags

Other options (-Xlinker [option])

--model-large

Figure 1-8 · SDCC Linker Set to Large Memory Mode

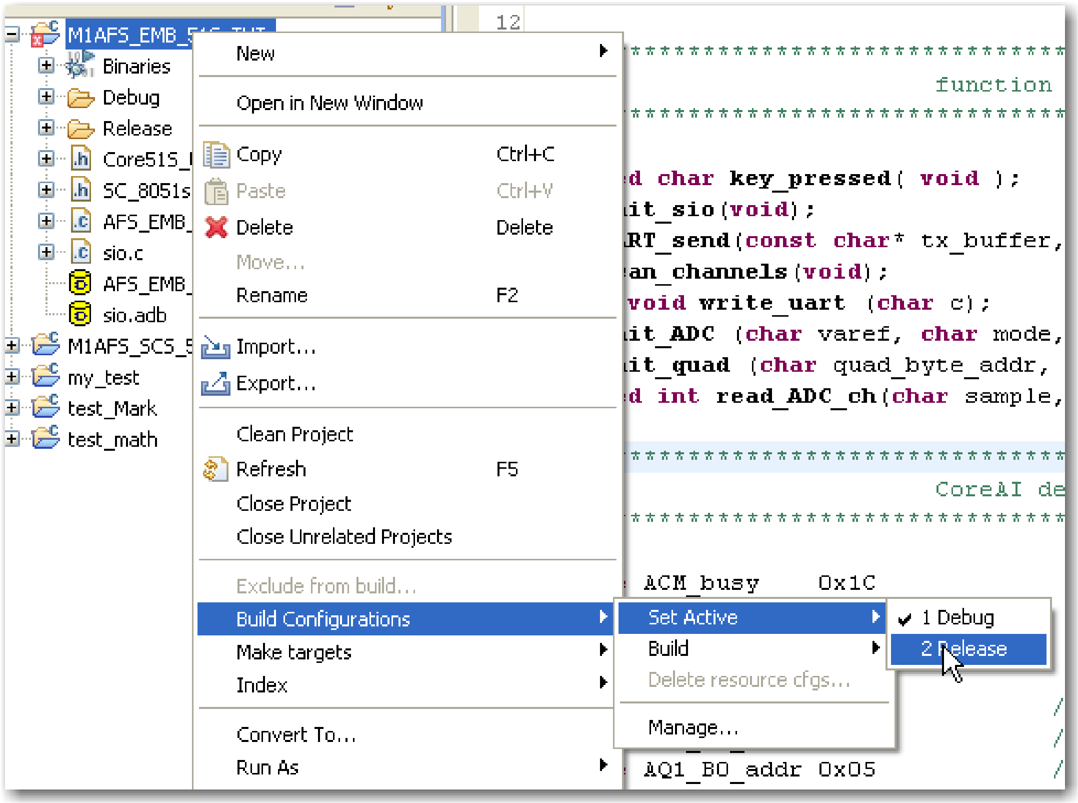
The large memory model is the default for SoftConsole and will be used for this tutorial. For reference small memory model applications are created by selecting the Small setting in the Memory options pull-down menu and typing **--model-small** for the SDCC Linker Miscellaneous options.

Creating the Debug Version

There are two types of output files that the compiler can create: a debug version and a release version. The debug file includes the user's application code along with additional information used by the debugger. The debug option creates a file, default.elf, and is placed in the *Debug* folder of the project. This code is used along with the FlashPro3 debugger to run your application in debug mode.

To create the debug version:

Right-click on the project name and scroll down to Build Configurations. Scroll to Set Active. Select **Debug** to select a debug build. (Figure 1-9).



```
12
*****************************
                     function
*****************************

d char key_pressed( void );
it_sio(void);
RT_send(const char* tx_buffer,
an_channels(void);
void write_uart (char c);
it_ADC (char varef, char mode,
it_quad (char quad_byte_addr,
d int read_ADC_ch(char sample,

*****************************
                     CoreAI de
*****************************

ACM_busy     0x1C

AQ1_BO_addr 0x05
```

Figure 1-9 · Setting the Build Configuration

Creating the Release Version

The release file contains the user's application code in hex format and has the extension *.ihx. This file is placed in the *Release* folder of the project and should be used to program NVM code memory. The release file will have the name of the project with the extension *.ihx.

To create the release version:

Right-click on the project name and scroll down to Build Configurations. Scroll to Set Active. Select **Release** to select a release build.

Building the Project

There are three options for building the project (in addition to changing from release to debug, etc.). These are Build Project, Clean, or Build Automatically. These options are available from the Project pull-down menu.

- Build Automatically builds the project when you save a file. If this option is checked, the Build Project option is grayed out. Select Build Automatically for the tutorial.
- Clean provides an option to build a single project or build all projects in the workspace.
- Build Project builds the current project. This option is grayed out if Build Automatically is checked.

1. Set the build configuration to **Release** (see "Creating the Release Version").
2. Perform a project Clean by selecting **Project** from the SoftConsole menu, then selecting **Clean** (Figure 1-10).

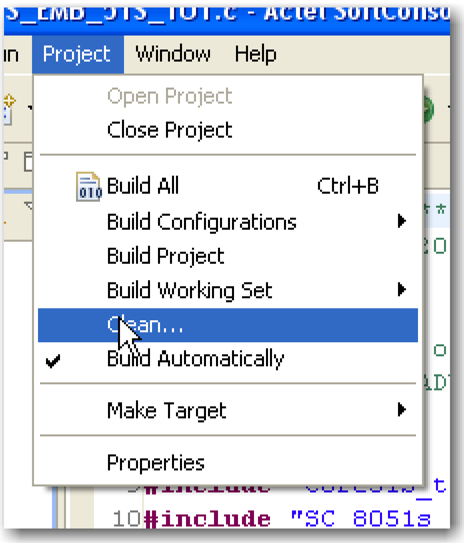
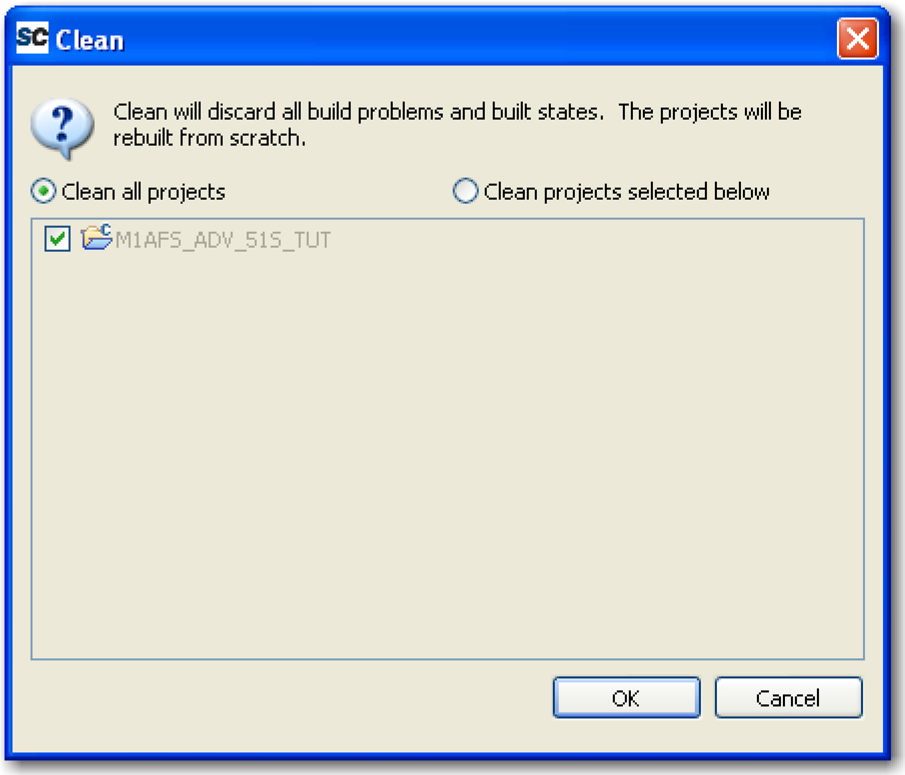


Figure 1-10 · Selecting a Clean Build

3. The Clean box will open. You have an option to clean all the projects in your workspace or only selected projects. In the case of this tutorial, there is only one project in the workspace so either option will work. Click **OK**.



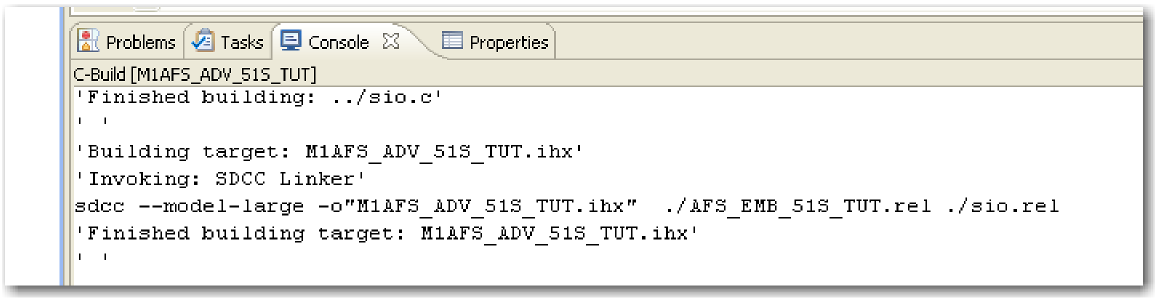
Clean will discard all build problems and built states. The projects will be rebuilt from scratch.

[✓ Clean all projects] Clean projects selected below

✓ M1AFS_ADV_51S_TUT

Figure 1-11 · Cleaning Options

4. The project will be compiled and linked with results indicated in the Console window.



```
C-Build [M1AFS_ADV_51S_TUT]
'Finished building: ../sio.c'
' '
'Building target: M1AFS_ADV_51S_TUT.ihx'
'Invoking: SDCC Linker'
sdcc --model-large -o"M1AFS_ADV_51S_TUT.ihx"  ./AFS_EMB_51S_TUT.rel ./sio.rel
'Finished building target: M1AFS_ADV_51S_TUT.ihx'
' '
```

Figure 1-12 · Successful Compilation of the Release Version

2

Programming Using FlashPro

The *Tutorial_Files* folder for your target board contains a PDB file used to program the FPGA fabric. When a hardware design contains a Fusion NVM module, the hardware designer is prompted to specify a *.hex file containing the code for this NVM. This hex code is merged into the PDB file by the hardware design tools. In this tutorial, the NVM functions as the code memory for the microcontroller and the hex code is the microcontroller's application code. The FlashPro software uses the PDB file to program the FPGA fabric. However, FlashPro also allows for the contents of the NVM to be changed without the need to regenerate the hardware design and create a new PDB file. This allows the software engineer to easily update the application code.

Prepare the Hardware Platform for Programming

1. Install the FlashPro software if not previously installed. The FlashPro software package includes drivers in addition to the FlashPro application. The FlashPro hardware functions as both an FPGA fabric programmer and as a JTAG debugger. There will be two drivers associated with the FlashPro hardware (one driver for the FPGA programmer and the other for the debugger).
2. Connect FlashPro3.

   If your target is the M1AFS-DEV-KIT-SCS board:

   - Connect a USB cable from a USB port on your PC to the USB PROG connector on the board. This is the connector for the FlashPro programmer and debugger integrated into this board design.

   If your target is the M1AFS-EMBEDDED-KIT board:

   - Plug the LC Programmer board into connector J1 of the M1AFS-EMBEDDED-KIT board. The LC Programmer board is functionally equivalent to a FlashPro3 programmer.
   - Connect a USB cable from a USB port on your PC to the USB connector on the LC Programmer board.

   If your target is the M1AFS-ADV-DEV-KIT board:

   - Plug the LC Programmer board into connector J1 of the M1AFS-ADV-DEV-KIT board. The LC Programmer board is functionally equivalent to a FlashPro3 programmer.
   - Connect a USB cable from a USB port on your PC to the USB connector on the LC Programmer board.
3. Connect the power.

   If your target is the M1AFS-DEV-KIT-SCS board, connect power to the power connector on the board.

   If your target is the M1AFS-EMBEDDED-KIT board, place jumper J40 in the USB position (this position supplies power to the board from the USB port and is the default). Connect a USB cable from a USB port on your PC to the USB connector J2 on your board. If prompted by the operating system for drivers, you can download the drivers for the CP2102 USB serial port from the Actel website:

   http://www.actel.com/products/hardware/devkits_boards/fusion_embedded.aspx

   If your target is the M1AFS-ADV-DEV-KIT board:

   - Connect power to the power connector on the board.
   - Slide switch SW7 to its ON position.

Launching FlashPro

The FlashPro Programmer will program both the FPGA array and the Embedded Flash Memory module. The FPGA array pattern implements the logic elements of the design. The Embedded Flash Memory module contains the code pattern for the application program for the microcontroller.

The FlashPro software can be launched from either Windows or from within Libero IDE. When launched from within Libero IDE, the FlashPro project will be created automatically and FlashPro will be given the path to the *.pdb file. When launched from Windows, however, you will be prompted to create the FlashPro project and you will need to provide FlashPro with the path to your *.pdb file.

Launching FlashPro from within Libero IDE

1. Open the Libero project for your board and verify that the Place & Route box is green. Click the **Programming** box in the Libero IDE.
2. Maximize the FlashPro dialog box. A FlashPro project is automatically created.

Launching FlashPro from Windows

When launched from Windows, FlashPro will remember the last FlashPro project that was opened. You must create a new FlashPro project for this application.

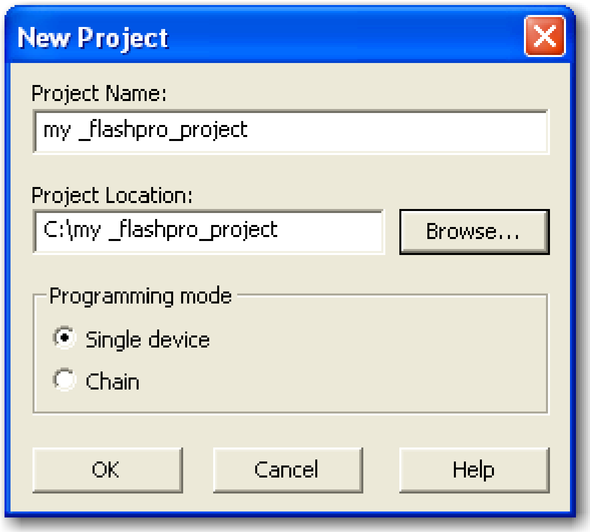
You will also need to import the *.pdb file created in the Libero IDE if you have completed the *Core8051s Embedded Processor Hardware Development Tutorial*. The *Tutorial_Files* folder for your target board contains a *.pdb file for this project that you can import and use.

1. Launch FlashPro from Windows. After FlashPro opens, click the **New Project** button.

   If your target is the M1AFS-DEV-KIT-SCS board, name the project M1AFS-DEV-KIT-SCS.

   If your target is the M1AFS-EMBEDDED-KIT board, name the project M1AFS-Embedded-KIT.

   If your target is the M1AFS-ADV-DEV-KIT board, name the project M1AFS-ADV-DEV-KIT.
2. Browse to the location where you would like to save the Flashpro project. Verify that programming mode is set to Single device. Click **OK** (Figure 2-1),


New Project

Project Name: my _flashpro_project

Project Location: C:\my _flashpro_project

Programming mode: [✓ Single device] Chain

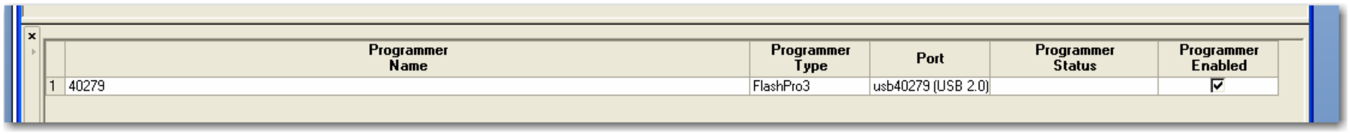
Figure 2-1 · Creating a FlashPro Project

Programming Using FlashPro

1. Verify that a programmer has been identified in the Programmer Name list (Figure 2-2). If no programmer name is shown, click the **Refresh/Rescan for Programmers** button. If no programmer is found, you might not have connected a USB cable between your PC and the FlashPro device.

   This is the connector marked USB PROG for the M1AFS-DEV-KIT-SCS board.

   This is the USB connector on the LC Programmer board used with the M1AFS-EMBEDDED-KIT and the M1AFS-ADV-DEV-KIT.



	Programmer Name	Programmer Type	Port	Programmer Status	Programmer Enabled
1	40279	FlashPro3	usb40279 (USB 2.0)		✓

Figure 2-2 · Programmer Identified

2. Click the **Configure Device** button (Figure 2-3).

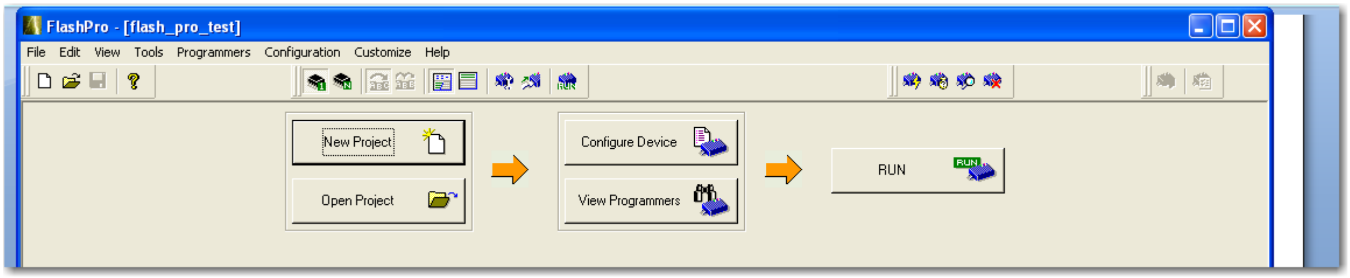
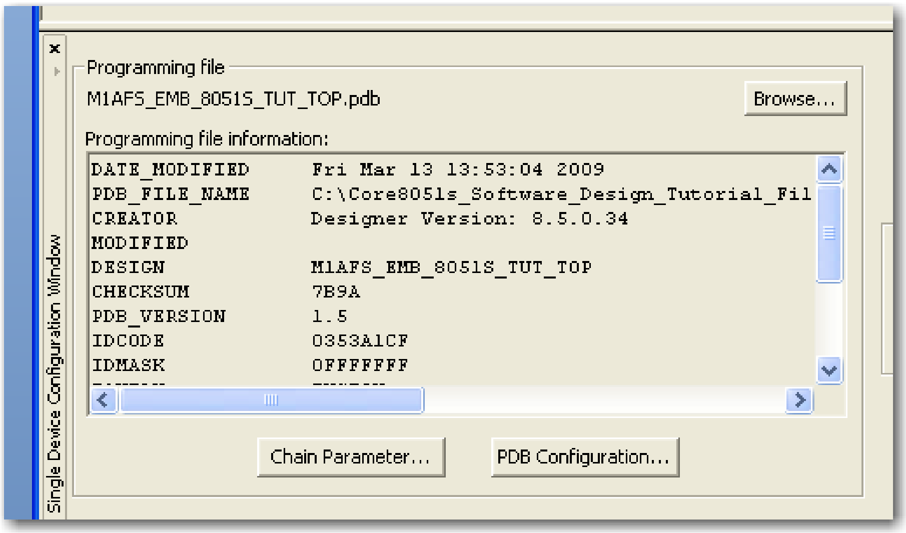

FlashPro - [flash_pro_test]
File Edit View Tools Programmers Configuration Customize Help
New Project
Open Project
Configure Device
View Programmers
RUN

Figure 2-3 · Configure Device Button

3. If FlashPro was launched from within Libero IDE, FlashPro received the path to the *.pdb file and will read and display information about the FPGA design in the programming file box (Figure 2-4 on page 20).

   If FlashPro was launched from Windows, you will need to provide FlashPro with the path to the *.pdb file. Click the **Browse** button in the Programming File box. Navigate to the *.pdb file for your hardware design if you have completed the *Core8051s Embedded Processor Hardware Development Tutorial*. If you have not completed this tutorial, navigate to the *Tutorial_Files* folder for your board and select the *.pdb file in this folder. Click **OPEN**.

FlashPro will read and display information about the FPGA design in the Programming File box (Figure 2-4).



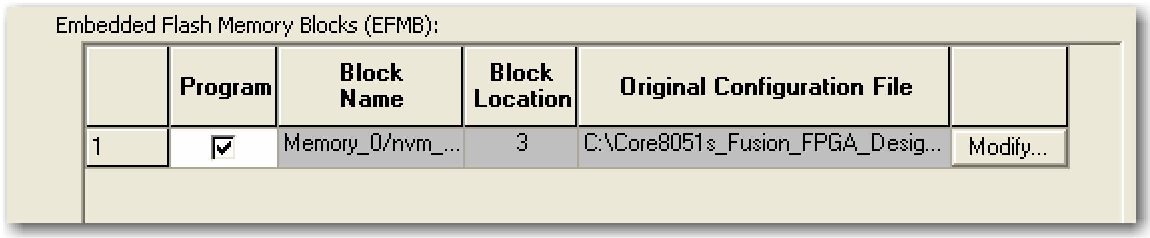
M1AFS_EMB_8051S_TUT_TOP.pdb

Programming file information:

```
DATE_MODIFIED     Fri Mar 13 13:53:04 2009
PDB_FILE_NAME     C:\Core8051s_Software_Design_Tutorial_Fil
CREATOR           Designer Version: 8.5.0.34
MODIFIED
DESIGN            M1AFS_EMB_8051S_TUT_TOP
CHECKSUM          7B9A
PDB_VERSION       1.5
IDCODE            0353A1CF
IDMASK            0FFFFFFF
```

Figure 2-4 · Programming File Box

4. The *.pdb file contains hex code for the NVM code memory block that was linked to the *.pdb file at the time that the Libero project was built. It is the correct code for this tutorial. However, the steps below illustrate how to update this file without the need to regenerate the FPGA design. This is useful when you want to make a firmware change.
   - Click the **PDB Configuration** button in the programming file box (Figure 2-4 on page 20). This will open the Programming File Generator box.
   - Click the **Modify** button in the Programming File Generator box (Figure 2-5).The embedded Flash Memory Block box will open.
   - Click the **Import content** button for the block content (Figure 2-6 on page 21). Navigate to the Tutorial_Files folder and click the *.ihx file. Click **Import**. Click **OK** to close the Modify Embedded Flash Memory Block box.


Embedded Flash Memory Blocks (EFMB):

	Program	Block Name	Block Location	Original Configuration File	
1	✓	Memory_0/nvm_...	3	C:\Core8051s_Fusion_FPGA_Desig...	Modify...

Figure 2-5 · Modify Button

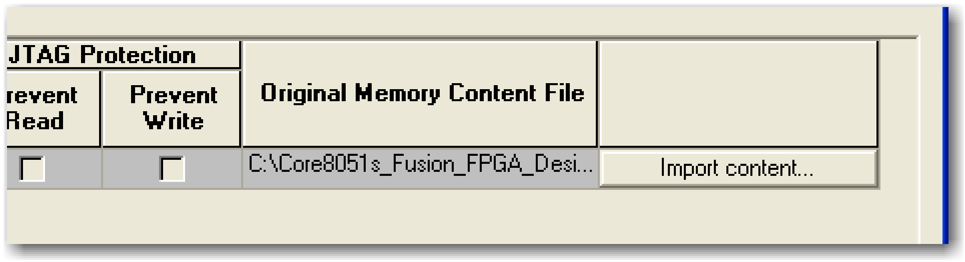
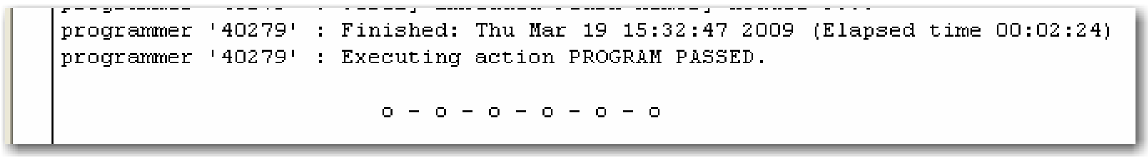

JTAG Protection
revent Read
Prevent Write
Original Memory Content File
C:\Core8051s_Fusion_FPGA_Desi...
Import content...

Figure 2-6 · Click this Button to Import the *.ihx File

- Click **Finish** to close the Programming File Generator window.

Subsequent changes in the code will only require re-importing the content for the flash memory block.

5. Click the **Program** button. FlashPro will erase, program, and verify the FPGA and the Embedded Flash Memory Module. Do not disturb power during the erase or program process as this could potentially damage the device. FlashPro will indicate that the programming process has completed by printing a "PROGRAM PASSED" message in the FlashPro console window (Figure 2-7).



```
programmer '40279' : Finished: Thu Mar 19 15:32:47 2009 (Elapsed time 00:02:24)
programmer '40279' : Executing action PROGRAM PASSED.

                             o - o - o - o - o - o
```

Figure 2-7 · Programming Completed

Running the Application from NVM Code Memory

1. Disconnect the FlashPro3 hardware.

   If your target is the M1AFS-DEV-KIT-SCS board, disconnect the USB cable from the USB PROG connector on the board.

   If your target is the M1AFS-EMBEDDED-KIT board, unplug the LC Programmer board from connector J1.

   If your target is the M1AFS-ADV-DEV-KIT board, unplug the LC Programmer board from connector J1.

2. Connect the serial port:

   If your target is the M1AFS-DEV-KIT-SCS board, connect a USB cable from a USB port on your PC to the USB PROG connector on the board.

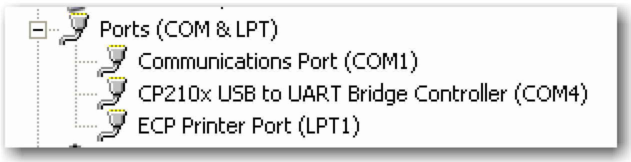
   If your target is the M1AFS-EMBEDDED-KIT board, the serial port should still be connected from the FPGA programming step.

   If your target is the M1AFS-ADV-DEV-KIT board, connect a USB cable from a USB port on your PC to the USB SERIAL connector on the board.

   If prompted by the operating system for drivers, you can download the drivers for the CP2102 USB serial port from http://www.actel.com/products/hardware/devkits_boards/fusion_embedded.aspx.

3. Determine the COM port assigned to the USB interface:

   Open the Windows Control Panel and double-click the **System** icon. Click the **Hardware** tab. Click the **Device Manager** button. Expand the Ports (COM & LPT) item in Device Manager. Look for the CP2102 USB to UART Bridge Controller in the list of ports (Figure 2-8). The COM port in parentheses next to it is the COM port assigned to this device.



Ports (COM & LPT)
Communications Port (COM1)
CP210x USB to UART Bridge Controller (COM4)
ECP Printer Port (LPT1)

Figure 2-8 · Determining the USB Serial Port's COM Port

4. Close the Device Manager. Close the Control Panel.

5. Selecting NVM code memory:

   If your target is the M1AFS-DEV-KIT-SCS board, set position 10 of switch SW1 to the OFF position.

   If your target is the M1AFS-EMBEDDED-KIT board or the M1AFS-ADV-DEV-KIT board, remove the jumper, if any, between pins 2 and 4 of connector P2.

6. Cycle power to the target board:

   If your target is the M1AFS-DEV-KIT-SCS board, unplug the power connector. Wait 5 seconds. Plug the power connector back in. Note: the serial port interface is powered separately by the USB cable.

   If your target is the M1AFS-EMBEDDED-KIT board, unplug the serial port connect. Wait 5 seconds. Plug the serial port connector back in.

   If your target is the M1AFS-ADV-DEV-KIT board, slide switch SW7 to the OFF position.

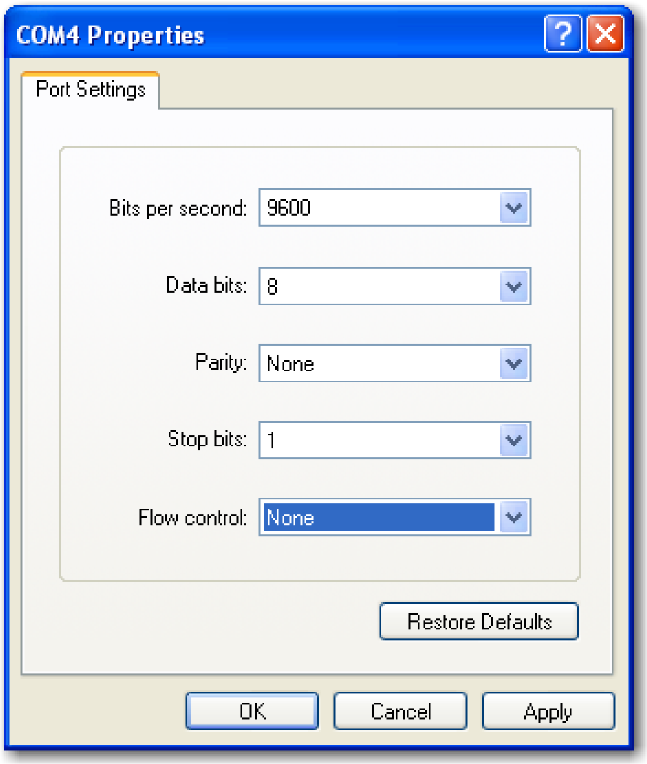
7. Prepare the terminal emulator:

   A terminal emulation program running on your host PC is used with the software in this project, whether running the application from NVM or using the debugger. The serial communications parameters used in this software are as follows:

   - 9,600 baud
   - 8 data bits
   - 1 stop bit

- No parity
- No flow control (handshaking)

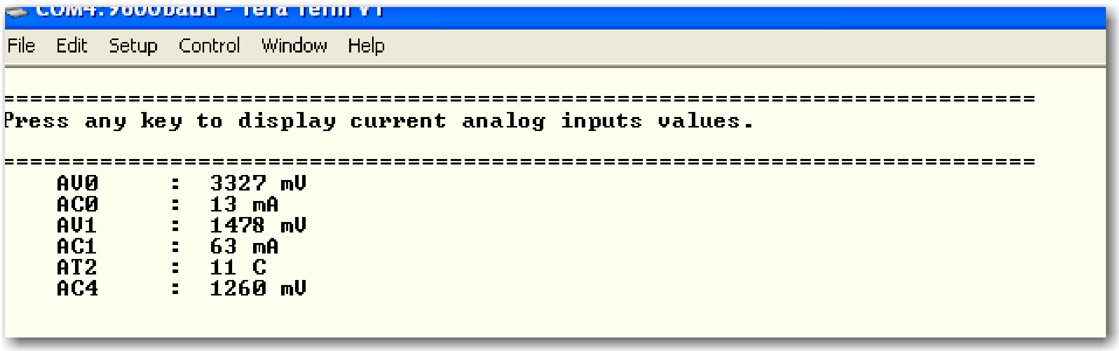
Open a terminal emulator program and set the communications parameters (Figure 2-9). Set the COM port to match the USB serial port of the board. Consult the documentation for your terminal emulation program for information on configuring your terminal emulation.



Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

Figure 2-9 · Communication Settings

8. Press the reset push-button on your target board. The terminal program should show a sign-on message. Click any key on your keyboard to send a character to the board. The board should respond by printing a message similar to that shown in Figure 2-10.


COM4:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
==========================================================================
Press any key to display current analog inputs values.

==========================================================================
    AV0      :  3327 mV
    AC0      :  13 mA
    AV1      :  1478 mV
    AC1      :  63 mA
    AT2      :  11 C
    AC4      :  1260 mV
```

Figure 2-10 · Application Output

Debugging

Debugging requires that SRAM be used to store program code since the debugger uses soft breakpoints. The design used in the Core8051s Fusion FPGA Design Tutorial meets this requirement.

Preparing for Debugging

1. This portion of the tutorial assumes that you have programmed the FPGA fabric with FlashPro3 and still have the FlashPro3 hardware connected to your target board and to you computer's USB port.
2. Connect the serial port:

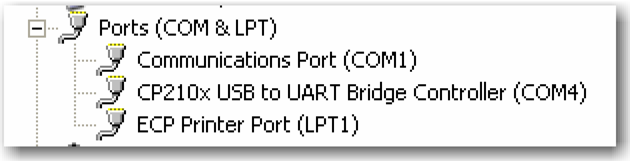
   If your target is the M1AFS-DEV-KIT-SCS board, connect a USB cable from a USB port on your PC to the USB PROG connector on the board.

   If your target is the M1AFS-EMBEDDED-KIT board, the serial port should still be connected from the FPGA programming step.

   If your target is the M1AFS-ADV-DEV-KIT board, connect a USB cable from a USB port on your PC to the USB SERIAL connector on the board.

   If prompted by the operating system for drivers, you can download the drivers for the CP2102 USB serial port from http://www.actel.com/products/hardware/devkits_boards/fusion_embedded.aspx.
3. Determine the COM port assigned to the USB interface:

   Open the Windows Control Panel and double-click the **System** icon. Click on the **Hardware** tab. Click the **Device Manager** button. Expand the Ports (COM & LPT) item in Device Manager. Look for the CP2102 USB to UART Bridge Controller in the list of ports (Figure 3-1). The COM port in parentheses next to it is the COM port assigned to this device.



Ports (COM & LPT)
Communications Port (COM1)
CP210x USB to UART Bridge Controller (COM4)
ECP Printer Port (LPT1)

Figure 3-1 · Determining the USB Serial Port's COM Port

4. Close the Device Manager. Close the Control Panel.
5. Selecting SRAM code memory:

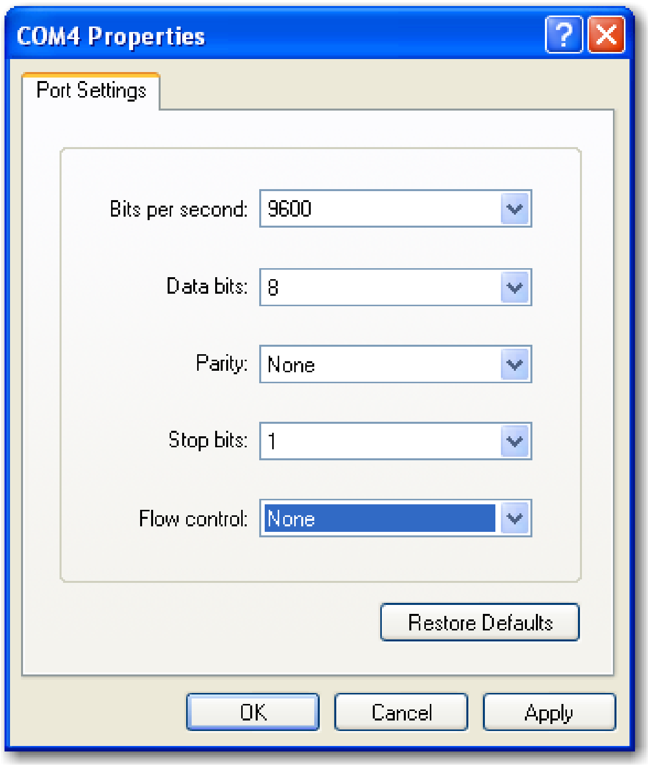
   If your target is the M1AFS-DEV-KIT-SCS board, set position 10 of switch SW1 to the ON position.

   If your target is the M1AFS-EMBEDDED-KIT board or the M1AFS-ADV-DEV-KIT board, place a jumper between pins 1 and 2 of connector J5.
6. Prepare the terminal emulator:

   A terminal emulation program running on your host PC is used with the software in this project whether running the application from NVM or using the debugger. The serial communications parameters used in this software are as follows:

   - 9,600 baud
   - 8 data bits
   - 1 stop bit
   - No parity
   - No flow control (handshaking)

Open a terminal emulator program and set the communications parameters (Figure 3-2). Set the COM port to match the USB serial port of the board. Consult the documentation for your terminal emulation program for information on configuring your terminal emulation.

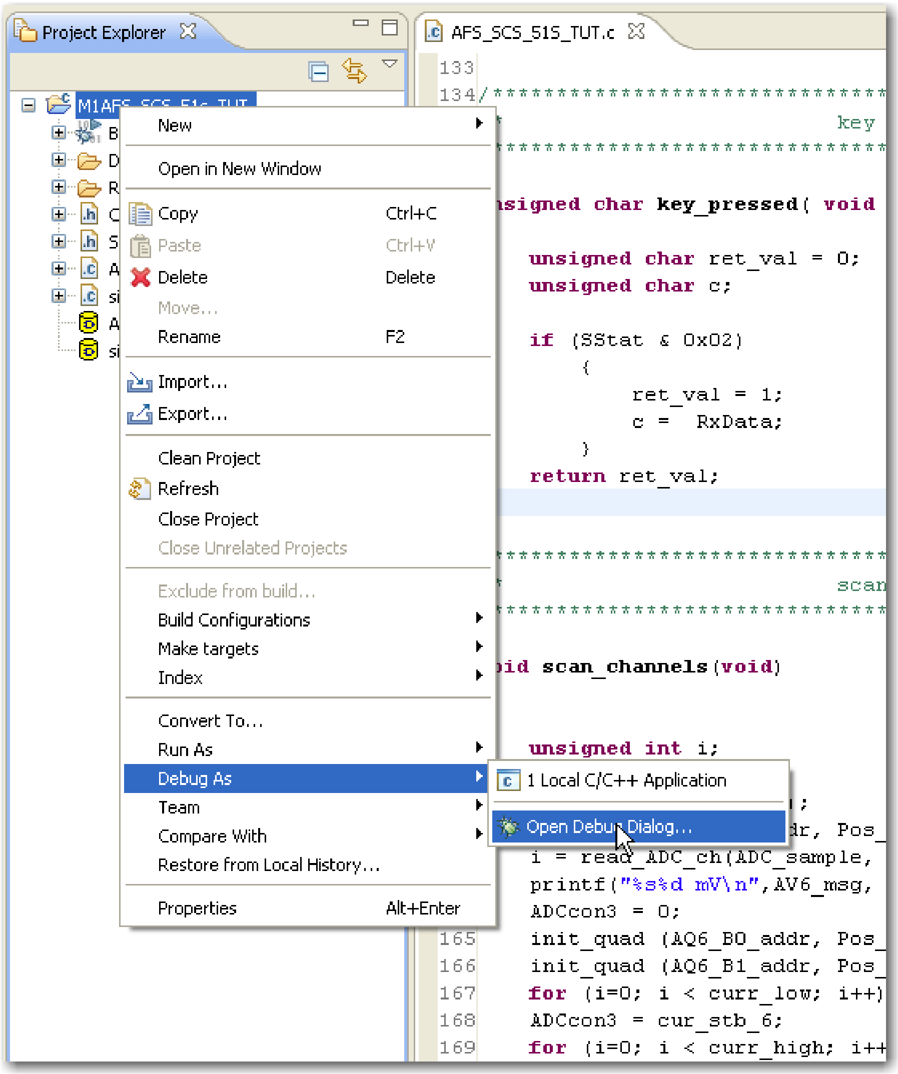


Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

Figure 3-2 · Communications Settings

Debugging

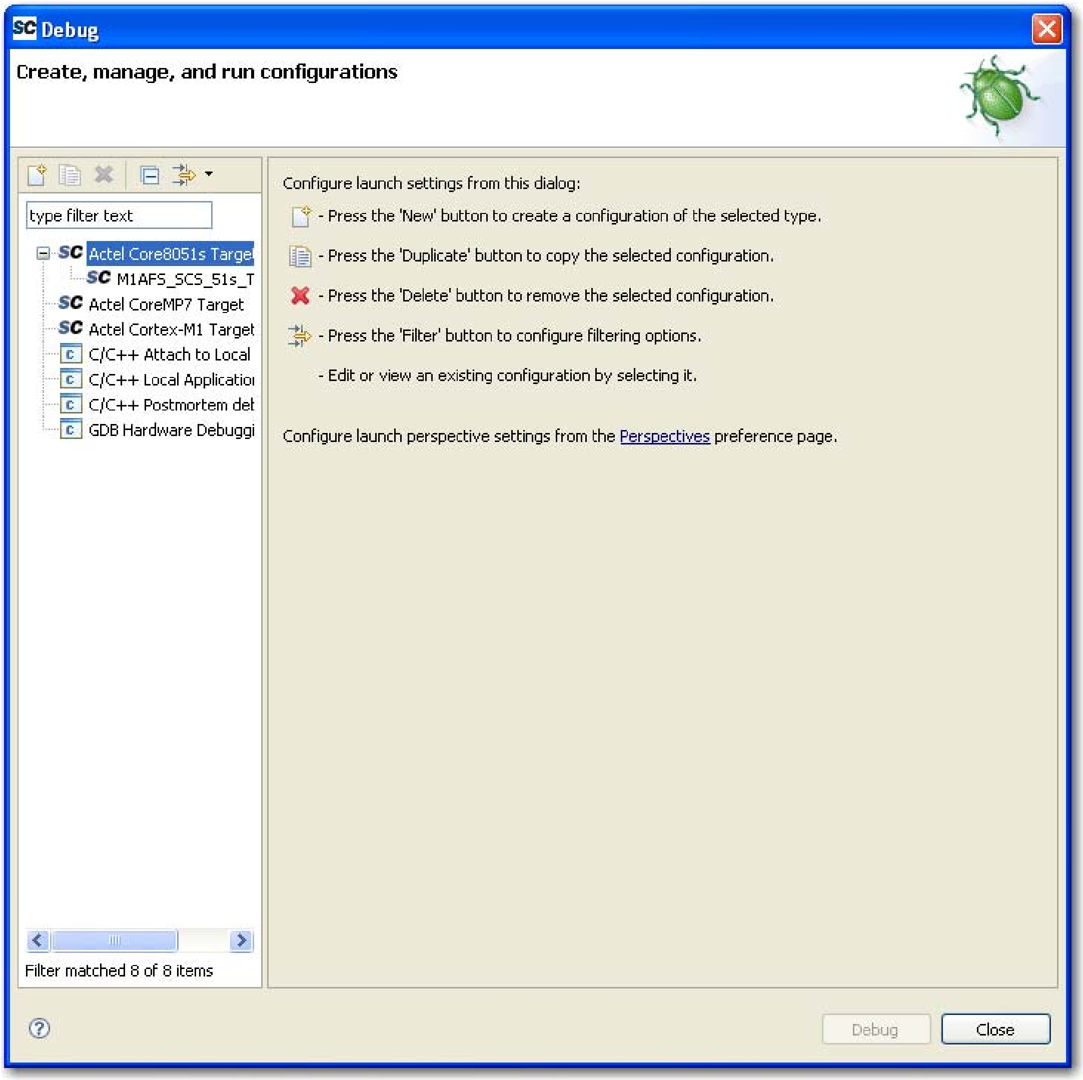
1. In Project Explorer, right-click on the project and choose **Debug As**. Select **Open Debug Dialog** (Figure 3-3).



```
/*********************************
*                            key
*********************************
nsigned char key_pressed( void

   unsigned char ret_val = 0;
   unsigned char c;

   if (SStat & 0x02)
       {
           ret_val = 1;
           c =  RxData;
       }
   return ret_val;


*********************************
*                            scan
*********************************

oid scan_channels(void)


   unsigned int i;
                              ;
                           dr, Pos_
   i = read_ADC_ch(ADC_sample,
   printf("%s%d mV\n",AV6_msg,
   ADCcon3 = 0;
   init_quad (AQ6_B0_addr, Pos_
   init_quad (AQ6_B1_addr, Pos_
   for (i=0; i < curr_low; i++)
   ADCcon3 = cur_stb_6;
   for (i=0; i < curr_high; i++
```

Figure 3-3 · Opening the Debug Dialog Box

2. Right-click on the relevant target (Actel Core8051s Target) and choose **New** (Figure 3-4). SoftConsole will automatically configure the debug utility and open the Debug window. Click the **Debug** button. The Debug window will close.



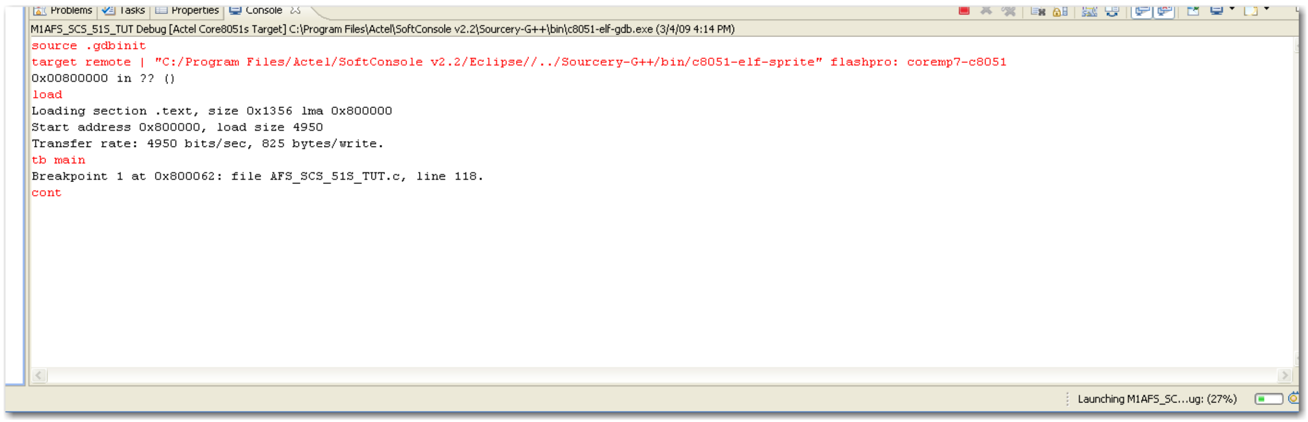
Configure launch settings from this dialog:

- 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 3-4 · Selecting the Debug Target

3. You should see some communications in the console window and see your file loading status in the lower right of the console window (Figure 3-5).

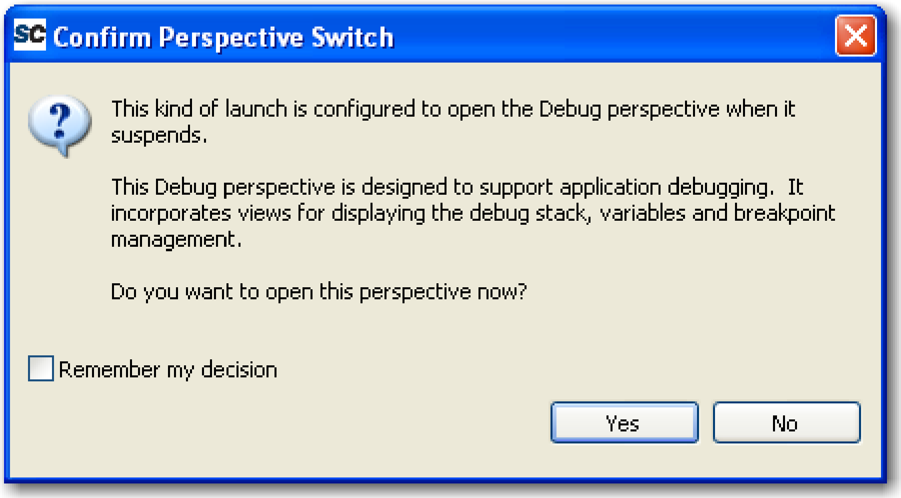


```
M1AFS_SCS_51S_TUT Debug [Actel Core8051s Target] C:\Program Files\Actel\SoftConsole v2.2\Sourcery-G++\bin\c8051-elf-gdb.exe (3/4/09 4:14 PM)
source .gdbinit
target remote | "C:/Program Files/Actel/SoftConsole v2.2/Eclipse//../Sourcery-G++/bin/c8051-elf-sprite" flashpro: coremp7-c8051
0x00800000 in ?? ()
load
Loading section .text, size 0x1356 lma 0x800000
Start address 0x800000, load size 4950
Transfer rate: 4950 bits/sec, 825 bytes/write.
tb main
Breakpoint 1 at 0x800062: file AFS_SCS_51S_TUT.c, line 118.
cont
```

Launching M1AFS_SC...ug: (27%)

Figure 3-5 · Console Communications

4. After the file loads, the Debug window will disappear and the Confirm Perspective Switch window will open. Click the **Yes** button (Figure 3-6). The debug perspective will open.



Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

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 3-6 · Confirming the Perspective Switch

5. Start the application by clicking **Run** > **Resume** or by clicking the **Resume** button (Figure 3-7). The terminal program should show a sign-on message. Click any key on your keyboard to send a character to the board. The board should respond by printing a message similar to that shown in Figure 3-8 on page 30. The exact text depends on the target board.

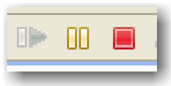
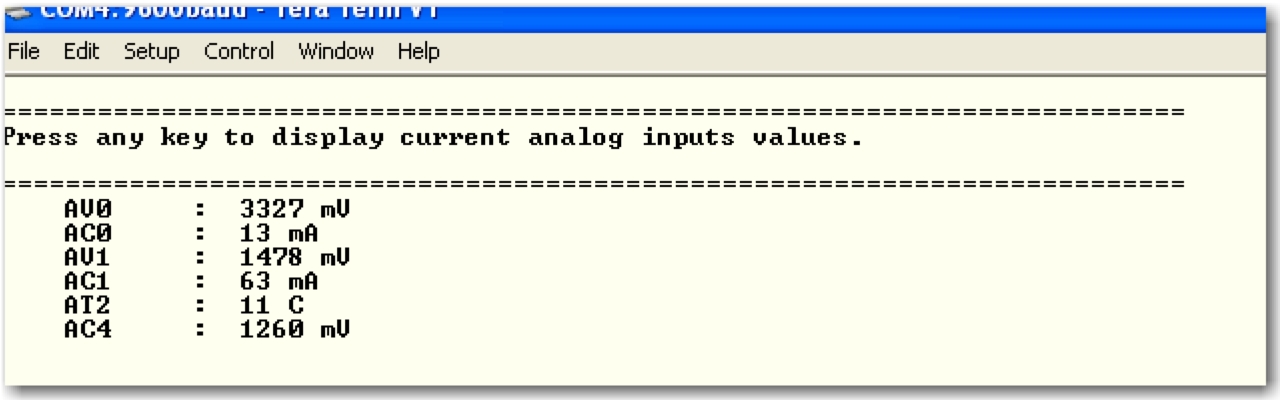


Figure 3-7 · Resume, Suspend, and Terminate Button


COM4:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
=========================================================================
Press any key to display current analog inputs values.

=========================================================================
    AV0      :  3327 mV
    AC0      :  13 mA
    AV1      :  1478 mV
    AC1      :  63 mA
    AT2      :  11 C
    AC4      :  1260 mV
```

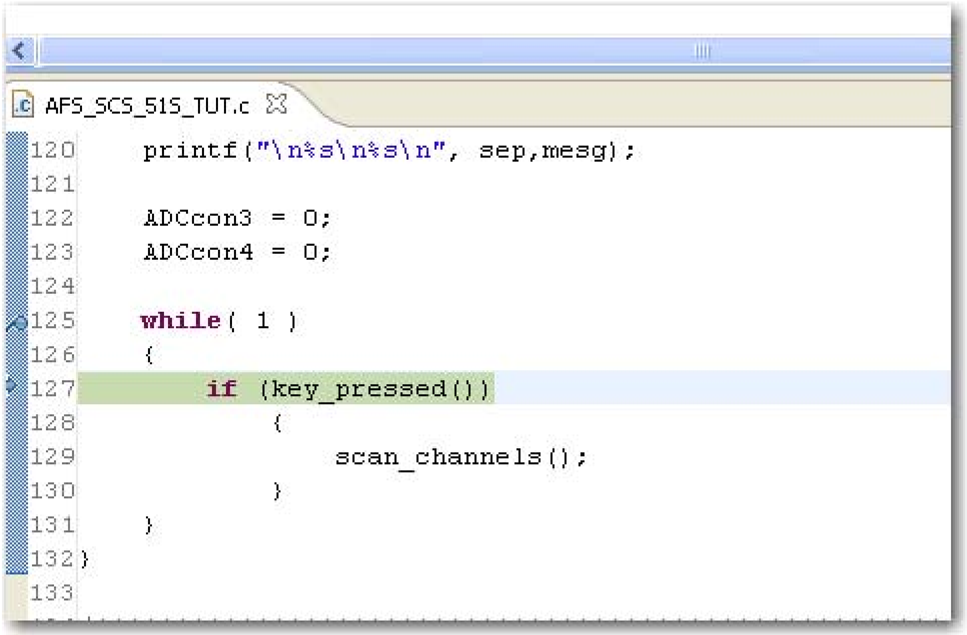
Figure 3-8 · Application Output

6. Breakpoints can be implemented by placing the cursor on a line of code and selecting **Run** > **Toggle Breakpoin**t. You can also double-click on the line of code to the left of the line number and toggle the breakpoint.

   Click the **Suspend** button or select **Run**> **Suspend** from the debugger menu.

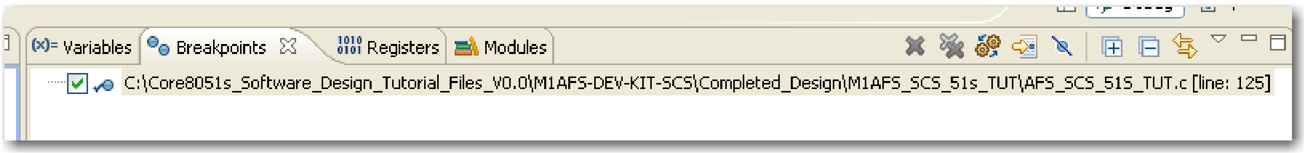
   Find the “while (1)” statement in “main” function In the source code. Click on this line of code and select **Run** > **Toggle Breakpoint** for the debugger menu. The line number for this line of code will show in the breakpoint tab in the upper right window pane (Figure 3-9 and Figure 3-10 on page 31).

   Click the **Resume** button. The code will break right before it checks for a character received by the serial port. The Resume and Terminate buttons are active and only the Suspend button is inactive (“grayed out”).



```
120     printf("\n%s\n%s\n", sep,mesg);
121
122     ADCcon3 = 0;
123     ADCcon4 = 0;
124
125     while( 1 )
126     {
127         if (key_pressed())
128         {
129             scan_channels();
130         }
131     }
132 }
133
```

Figure 3-9 · Breakpoint set at 'while (1)'



C:\Core8051s_Software_Design_Tutorial_Files_V0.0\M1AFS-DEV-KIT-SCS\Completed_Design\M1AFS_SCS_51s_TUT\AFS_SCS_51S_TUT.c [line: 125]

Figure 3-10 · Breakpoint Set at 'while (1)'

7. Click the **Registers** tab in the upper right pane of SoftConsole. Expand the "main" icon in the Registers window to see various 8051 registers.
8. In the SoftConsole menu, select **Window** > **Show View** > **Disassembly**. The middle right-hand pane will show the assembly code for the current function.
9. Click the **Step Into** icon or select **Run** > **Step Into** from the SoftConsole menu. You will see the debugger step to the first line of code in the key_pressed function. Click **Step Into** again. Since to character is waiting in the serial port's receiver, the code will proceed to the return instruction.
10. Click **Step Into** until the code exits the "key_pressed" function and returns to the "while (1)" loop.
11. This time, click the **Step Over** icon or select **Run** > **Step Over** from the SoftConsole menu. The code will execute the function in real time and stop and the next line of code to be executed in the "while (1)" loop. In this case it is function call to "key_pressed".
12. Click in your terminal program and type and key. Return to the debugger and click the **Step Over** icon or select **Run** > **Step Over** from the SoftConsole menu. This time the debugger will stop at the "scan_channels" line of code, indicating that the character from the terminal program was received by the UART.
13. Click the **Step Over** icon or select **Run** > **Step Over** from the SoftConsole menu. The ADC readings will be displayed in your terminal program and debugger will stop at the call to "key_pressed".
14. Click in your terminal program and type and key.
15. Return to the debugger and click the **Step Over** icon or select **Run** > **Step Over** from the SoftConsole menu. The debugger will stop at the "scan_channels" line of code indicating that the character from the terminal program was received by the UART.
16. Click the **Step Into** icon or select **Run** > **Step Into** from the SoftConsole menu. The debugger will stop at the first line of code inside the "scan_channels" function.

Summary

In this tutorial you have created a simple application program for an 8051-based embedded processor system using Actel tools. This design can serve as the basic starting point for other Core8051s designs. This process included the following steps:

1. Creating and configuring a SoftConsole project
2. Compiling your code
3. Programming your code into Fusion NVM
4. Launching the debugging and performing basic debugger operations from external SRAM

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel 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 Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650.318.8044**

Actel Customer Technical Support Center

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

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

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 tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/contact/offices/index.html.

Index

50200153-1/3.09