

---

# ***ARM<sup>®</sup> Cortex<sup>™</sup>-M1 Embedded Processor Software Development Tutorial for Fusion Mixed-Signal FPGAs***

© 2009 Actel Corporation. All rights reserved.

Printed in the United States of America

Part Number: 50200162-1

Release: November 2009

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Actel.

Actel makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability or fitness for a particular purpose. Information in this document is subject to change without notice. Actel assumes no responsibility for any errors that may appear in this document.

This document contains confidential proprietary information that is not to be disclosed to any unauthorized person without prior written consent of Actel Corporation.

### **Trademarks**

Actel, IGLOO, Actel Fusion, ProASIC, Libero, Pigeon Point and the associated logos are trademarks or registered trademarks of Actel Corporation. All other trademarks and service marks are the property of their respective owners.

# Table of Contents

Introduction .....	5
Requirements for the Tutorial .....	5
Tutorial Files .....	5
<b>1 Before You Get Started .....</b>	<b>7</b>
Set Up the USB-to-UART Driver .....	7
Set Up HyperTerminal .....	9
Install Firmware Catalog .....	10
Setup Firmware Catalog to Run in SoftConsole .....	11
<b>2 Tutorial – Create and Build Software Project .....</b>	<b>17</b>
Open SoftConsole .....	17
Create and Build the Software Project .....	17
<b>3 Tutorial – Programming and Debug .....</b>	<b>27</b>
Program the FPGA .....	27
Create the Cortex-M1 Debug Configuration .....	29
Launch Debug Session .....	30
Continue Debug Session for Fusion-Based Kits .....	32
<b>A Appendix .....</b>	<b>37</b>
Debugging Features in SoftConsole .....	37
Updating Contents of NVM in PDB file .....	37
<b>B List of Document Changes .....</b>	<b>39</b>
<b>A Product Support .....</b>	<b>41</b>
Customer Service .....	41
Actel Customer Technical Support Center .....	41
Actel Technical Support .....	41
Website .....	41
Contacting the Customer Technical Support Center .....	41
<b>Index .....</b>	<b>43</b>



# Introduction

This tutorial shows you how use the Actel tools to develop a software application for a Cortex™-M1-based embedded processor system. After completing this tutorial you will be familiar with the software design process which includes the following steps:

- Creating the SoftConsole project
- Configuring the SoftConsole compiler/linker settings
- Compiling your code
- Creating and launching a debug session
- Debugging your code using SoftConsole

This design is suitable as a starting point for developing an embedded system.

## Requirements for the Tutorial

This tutorial requires SoftConsole v2.2 (or newer) and FlashPro v8.6 (or newer) 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. You will also need the following hardware:

- One of the following Actel development boards:
  - Fusion Embedded Development Kit (M1AFS-EMBEDDED-KIT)
  - Fusion Advanced Development Kit (M1AFS-ADV-DEV-KIT) along with power supply
- Programming hardware (low-cost programming stick or FlashPro3)
- 2 USB cables (programming and communication)

Programming of the application code to the target board requires a FlashPro3 programmer (or low-cost programming stick). Development kits contain a low-cost programming stick (LCPS) or FlashPro3 programmer. You will also need two USB cables; one for connecting the programmer to your PC and the other to connect the UART interface on the board to your PC.

## Tutorial Files

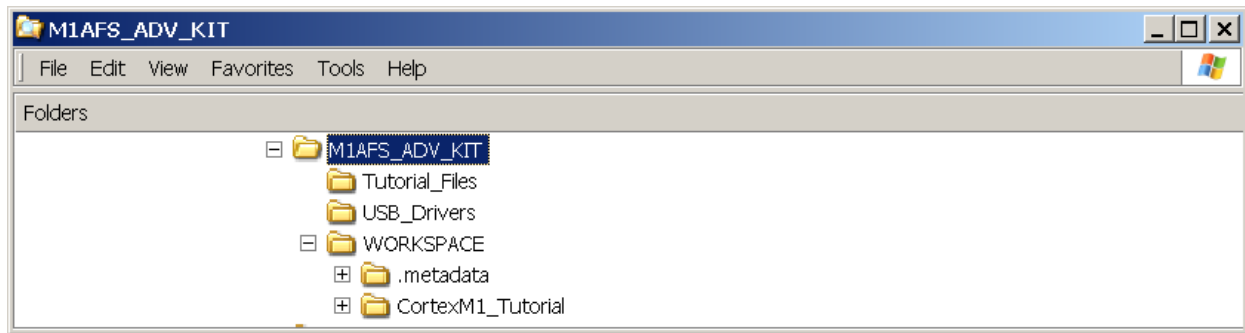
Before starting the tutorial, you need to download the tutorial files from the website for the board you are targeting. Go to Actel's Design Hardware web page:

(<http://www.actel.com/products/hardware/default.aspx>)

Click the link for the board you are targeting. On the web page of the target board, click the zip file under the *ARM Cortex-M1 Embedded Processor Software Development Tutorial* document. Download and unzip the file to your hard drive. The files are placed in a folder called *CortexM1\_Fusion\_SW\_Tutorial*.

If you browse the contents of the zip file, you will see a subdirectory for every board this tutorial supports. The directory structure is similar for each board. As an example, [Figure 1](#) shows a picture of the directory structure for the M1AFS-EMBEDDED-KIT board. The WORKSPACE is the software

tutorial solution directory. If you change the workspace in SoftConsole to this location (**File < Switch Workspace**), you can use this software database directly.



**Figure 1 • Directory Structure for M1AFS-EMBEDDED-KIT Board**

The *Completed\_Design* subdirectory contains the complete Cortex-M1 project for this tutorial design for your reference. The *Tutorial\_Files* subdirectory contains the source files you need to complete this tutorial. [Table 1](#) below has a description of these files.

**Table 1 • Tutorial Files Descriptions**

File Name	Description
M1<FPGA>_TUT_TOP.STP	FPGA programming file containing Cortex-M1 FPGA design tutorial (hardware design)
tutorial.h	Header file with memory map and interrupt assignments targeted for Cortex-M1 FPGA design tutorial
main.c	Top C source file
boot-from-actel-coreahbnvm.ld	Linker script modified to work with Fusion-based tutorial designs

The *drivers* subfolder contains the driver code for all the peripherals in the system. The *USB Drivers* subdirectory contains the driver for the USB-to-UART chip on the target board. This driver allows you to communicate with the target board over USB by treating the USB port as a COM port. Therefore, you can use HyperTerminal to connect to a UART in the design running on the board.

Before debugging software for a Cortex-M1 system, you need the \*.STP (program database) file, which is used to program the FPGA with the hardware design. The \*.STP file provided in this tutorial is from the design created in the Cortex-M1 <FPGA Family> FPGA Design Tutorial.

# 1 – Before You Get Started

## Set Up the USB-to-UART Driver

At the end of the tutorial, you program the Fusion FPGA and communicate with the target board. To complete the tutorial, you need to make sure that the USB-to-UART drivers are installed on your machine and that you know which COM port the USB port is associated with. Follow the steps below before starting the tutorial to make sure your machine is properly configured and connected to the target board.

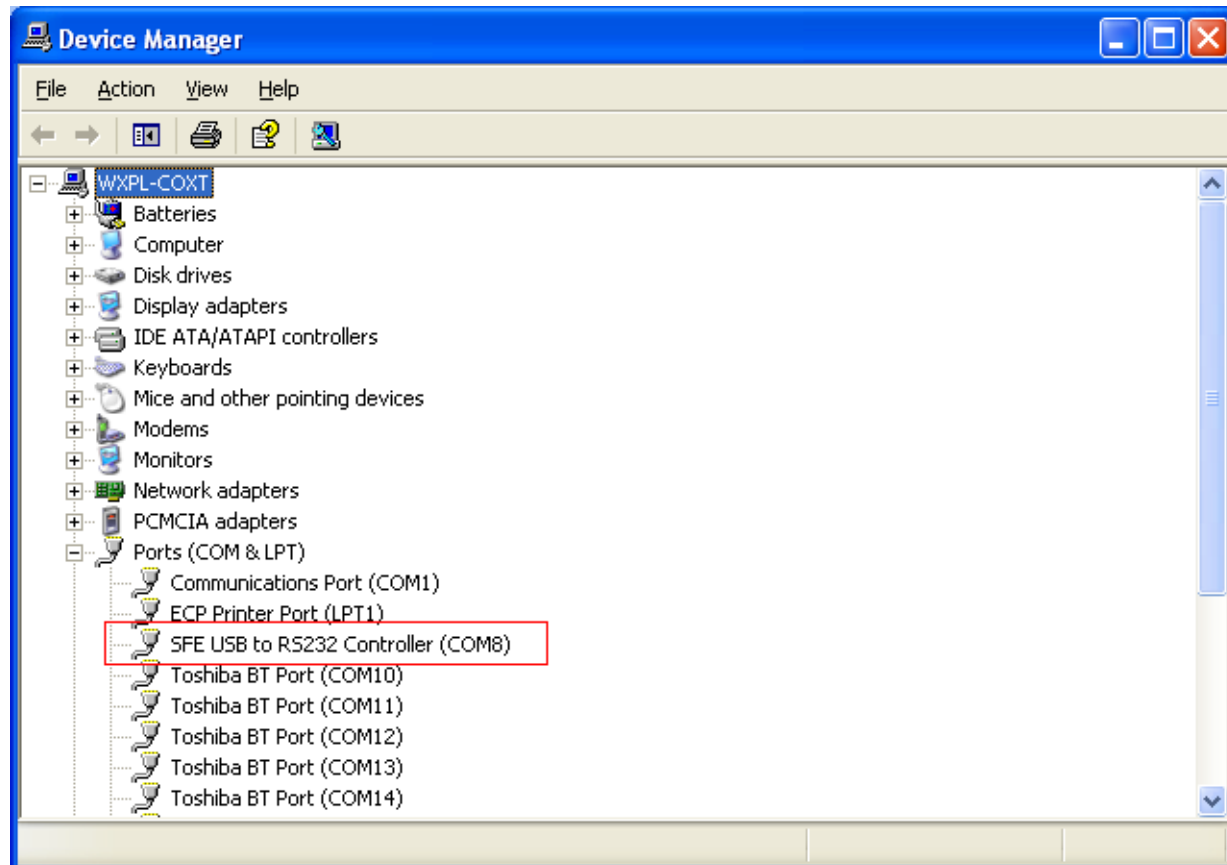
1. Install the drivers for the USB to RS-232 Bridge by double clicking on the **Preinstaller.exe** executable located in the *USB\_Drivers* subfolder. Accept the default installation folder and press the **Install** button. Press the **Continue Anyway** button if prompted.
2. Connect USB cables and power cables according to [Table 1-1](#).

**Table 1-1 • Board Connections**

Board	Power and USB Connections
M1AFS-EMBEDDED-KIT	Make sure that you have J40 set to use USB power (not V5IN). Connect a USB cable to J2, which provides power and the USB-to-UART communication.
M1AFS-ADV-DEV-KIT	Connect the 9 V power supply provided with the kit to J3 on the board. Connect a USB cable to J2. This provides the USB-to-UART communication and make sure SW7 is in the ON position to supply power to the board.

3. If Windows prompts you to connect to Windows Update, select **No, not at this time** and press **Next**. Next, select **Install the software automatically (recommended)** and press **Next**. Once installation has completed, press **Finish**. Repeat the driver installation steps a second time (if prompted). Press the **Continue Anyway** button if prompted.
4. Open the Windows® **Device Manager** by selecting **Start > Control Panel > System > Hardware > Device Manager**. Expand the **Ports (COM & LPT)** section and take note of the COM port assignment for the SFE USB to RS232 Controller.

The example shown in Figure 1-1 is assigned to COM8.



**Figure 1-1 • Device Manager**

If you do not see SFE USB to RS232 Controller in the Device Manager, you may need to reboot your machine.



## Set Up HyperTerminal

1. Open the HyperTerminal application (**Start > Programs > Accessories > Communications > HyperTerminal**). Enter **CortexM1\_Tutorial** in the Name field in the Connection Description dialog box and click **OK**.
2. Select the COM port you identified in "Set Up the USB-to-UART Driver" on page 1-7 and click **OK**.
3. Enter the following values for the properties (Figure 1-2):
  - Bits per second: 57600
  - Data bits: 8
  - Parity: None
  - Stop bits: 1
  - Flow control: None

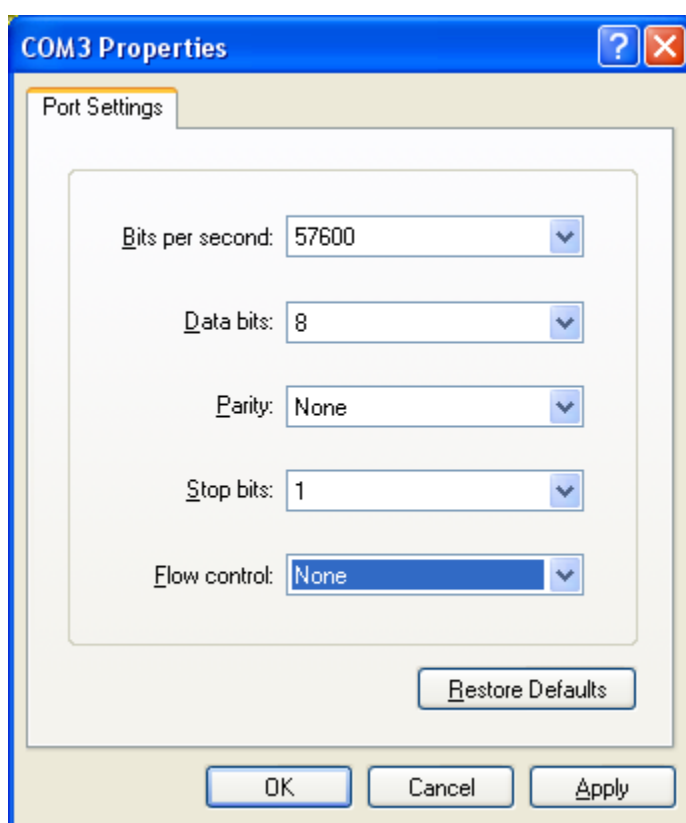


Figure 1-2 • COM Port Properties

4. Click **OK**. HyperTerminal is now connected with the appropriate settings to communicate with the UART on the target design.

## Install Firmware Catalog

Firmware Catalog is a tool that allows you to create firmware for software development. Typically, you use the tool to select the firmware that you need and Firmware Catalog places the firmware into your software development project (SoftConsole, IAR EWARM, or Keil). For this tutorial, you will use Firmware Catalog to add the drivers and HAL code to your SoftConsole project.

Firmware Catalog is installed by default when Libero IDE is installed (v8.6 or newer). If you do not want to install Libero IDE, you can download the Firmware Catalog installation only from Actel's Downloads web page:

[www.actel.com/download/default.aspx](http://www.actel.com/download/default.aspx)

Regardless of which installation method you use, Firmware Catalog is a standalone tool with its own executable.

## Setup Firmware Catalog to Run in SoftConsole

Use the following steps to link Firmware Catalog to SoftConsole.

1. Invoke SoftConsole and define the directory called **Workspace** for software project data (Figure 1-3).

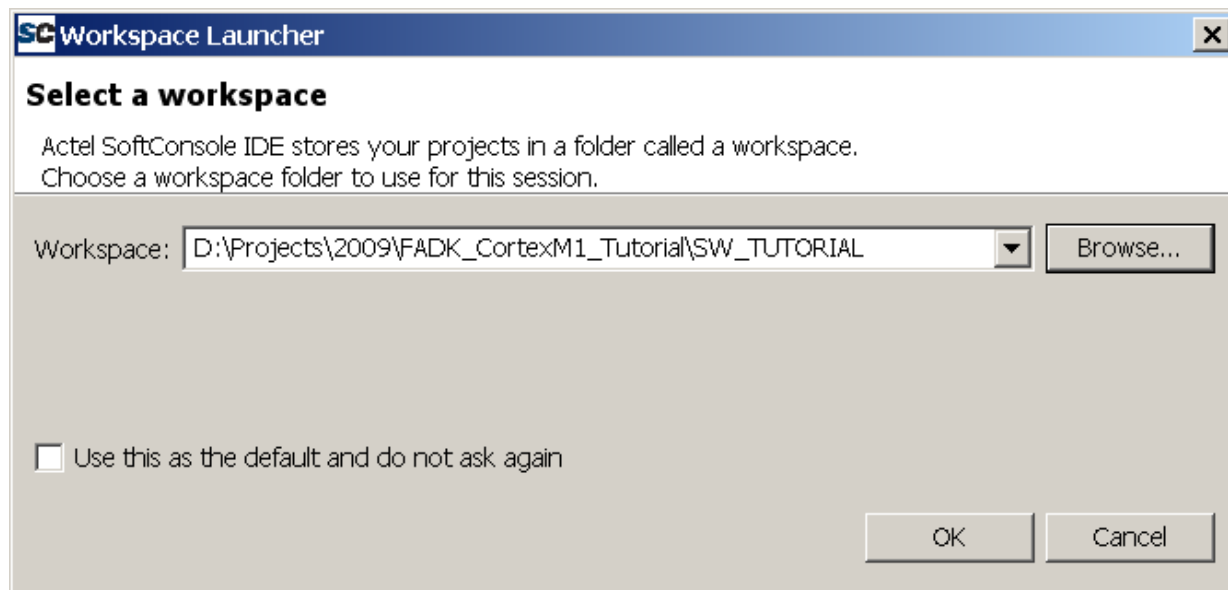


Figure 1-3 • Workspace Launcher

**Note:** Do not use folders that have spaces in the folder name.

- From the SoftConsole main GUI, select Run from the toolbar and select **External Tools > Open External Tools Dialog** (Figure 1-4).

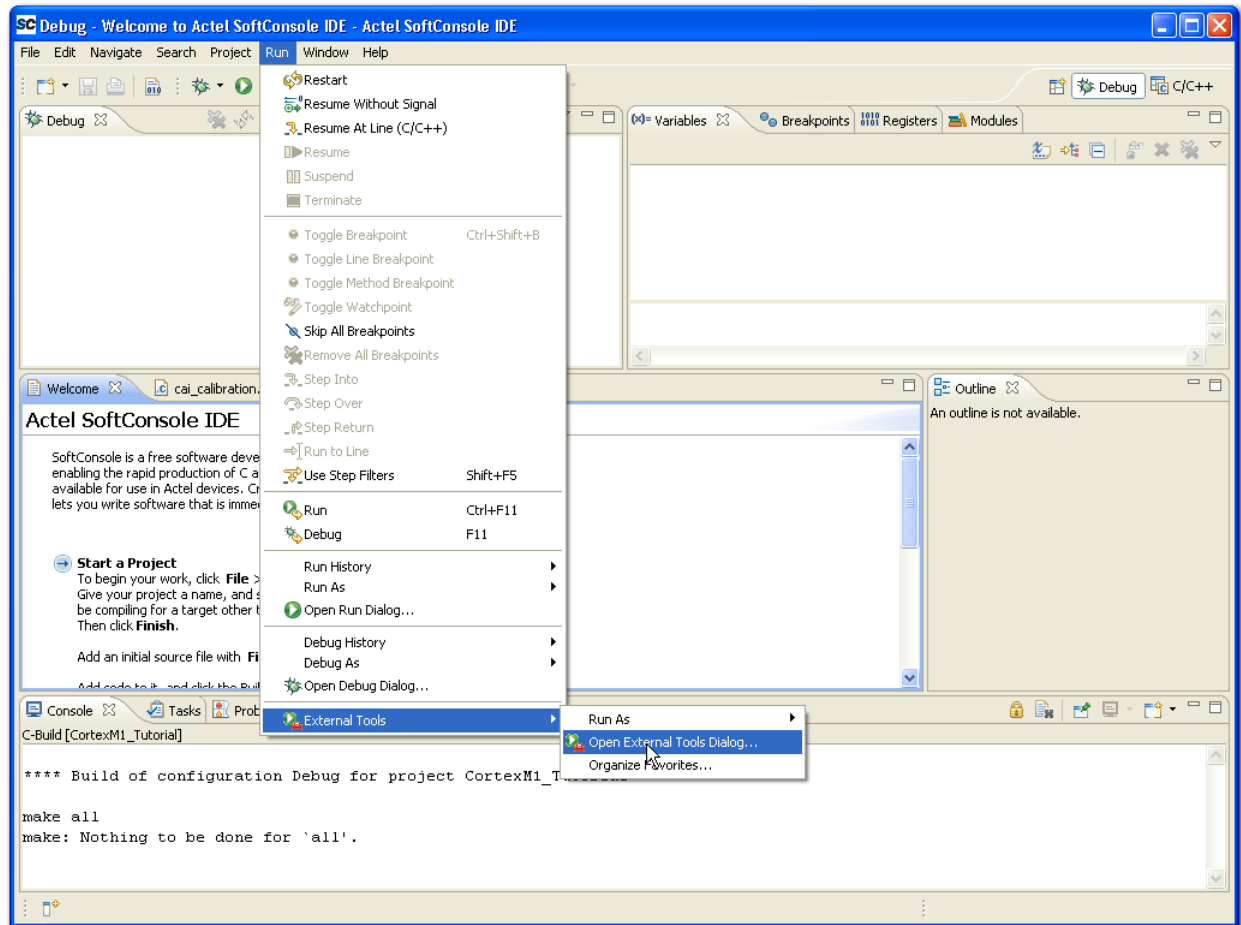
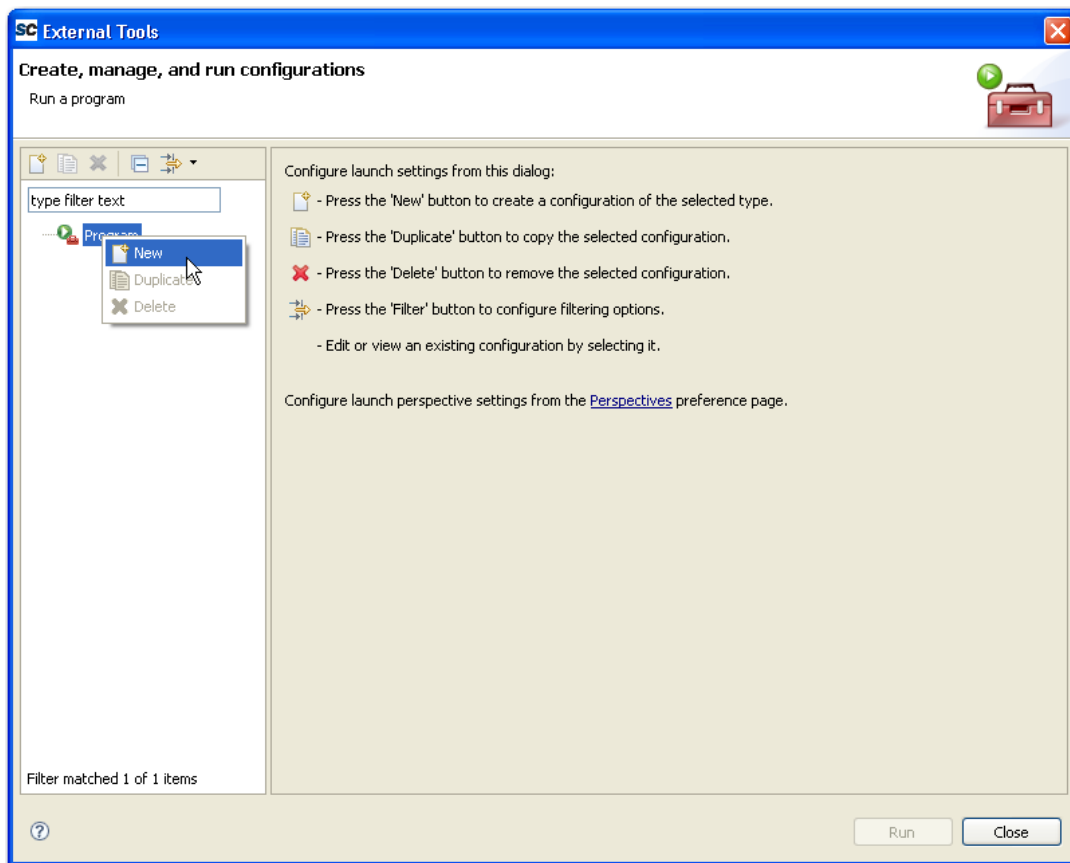


Figure 1-4 • Open External Tools Catalog

3. The External Tools dialog will open. Right-click Program, and click **New** (Figure 1-5).

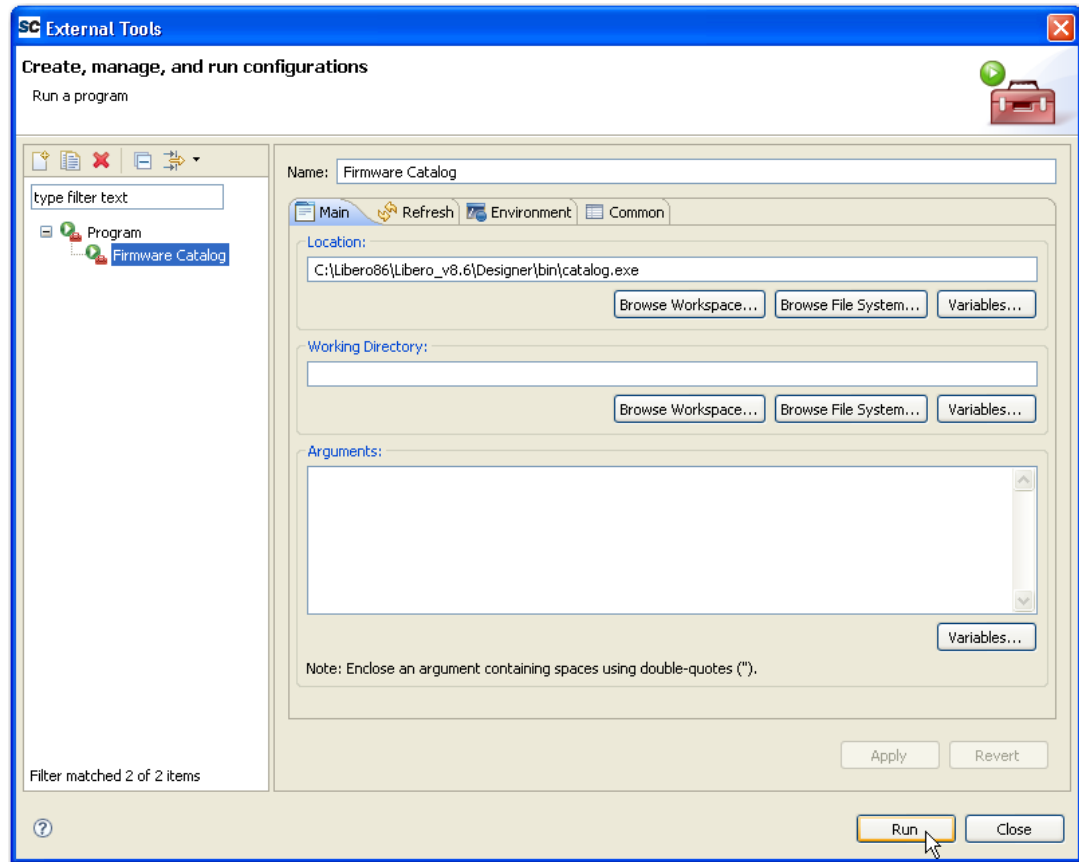


**Figure 1-5 • Create New Program**

The External Tools “Create, manage, and run configurations” dialog will open.

1. In the Name field, enter **Firmware Catalog**.
2. Using the Browse File System button, browse to the catalog.exe file. If you have installed Libero IDE, unless you are instructed otherwise, the catalog.exe file is placed in the path `c:\Libero86\Libero_v8.6\Designer\bin\catalog.exe`. If you downloaded and installed the Firmware Catalog independently, the executable will be at the following location: `<drive>:\Firmware_Catalog\bin\catalog.exe`.

3. After entering the path, click **Run** to save the configuration (Figure 1-6). The Firmware Catalog will also open.



**Figure 1-6 • Save Configuration**

4. Once the Firmware Catalog has opened, it can be closed. This is necessary to save the configuration in SoftConsole.

Close the Firmware Catalog (Figure 1-7) by clicking the X on the window.

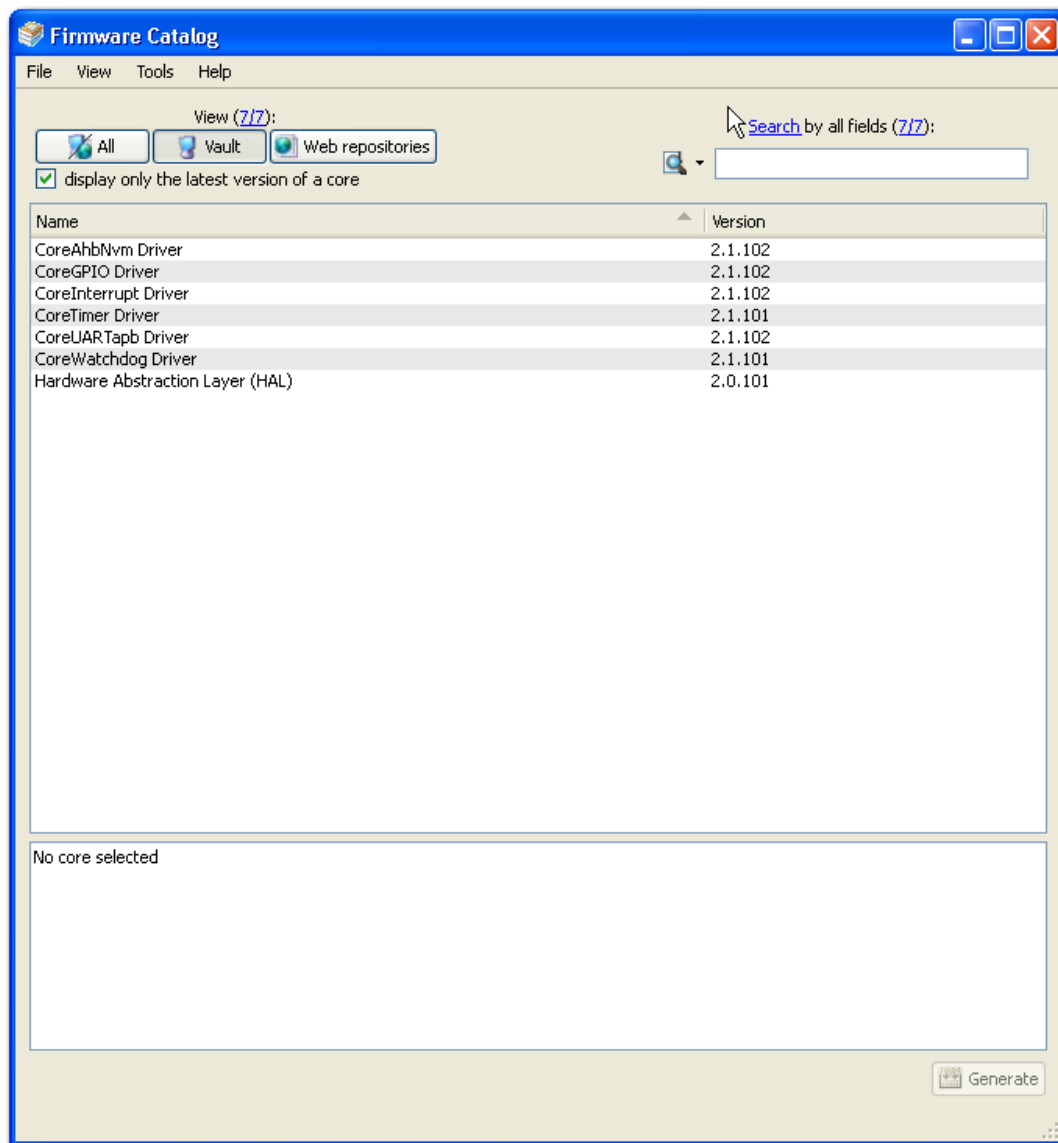
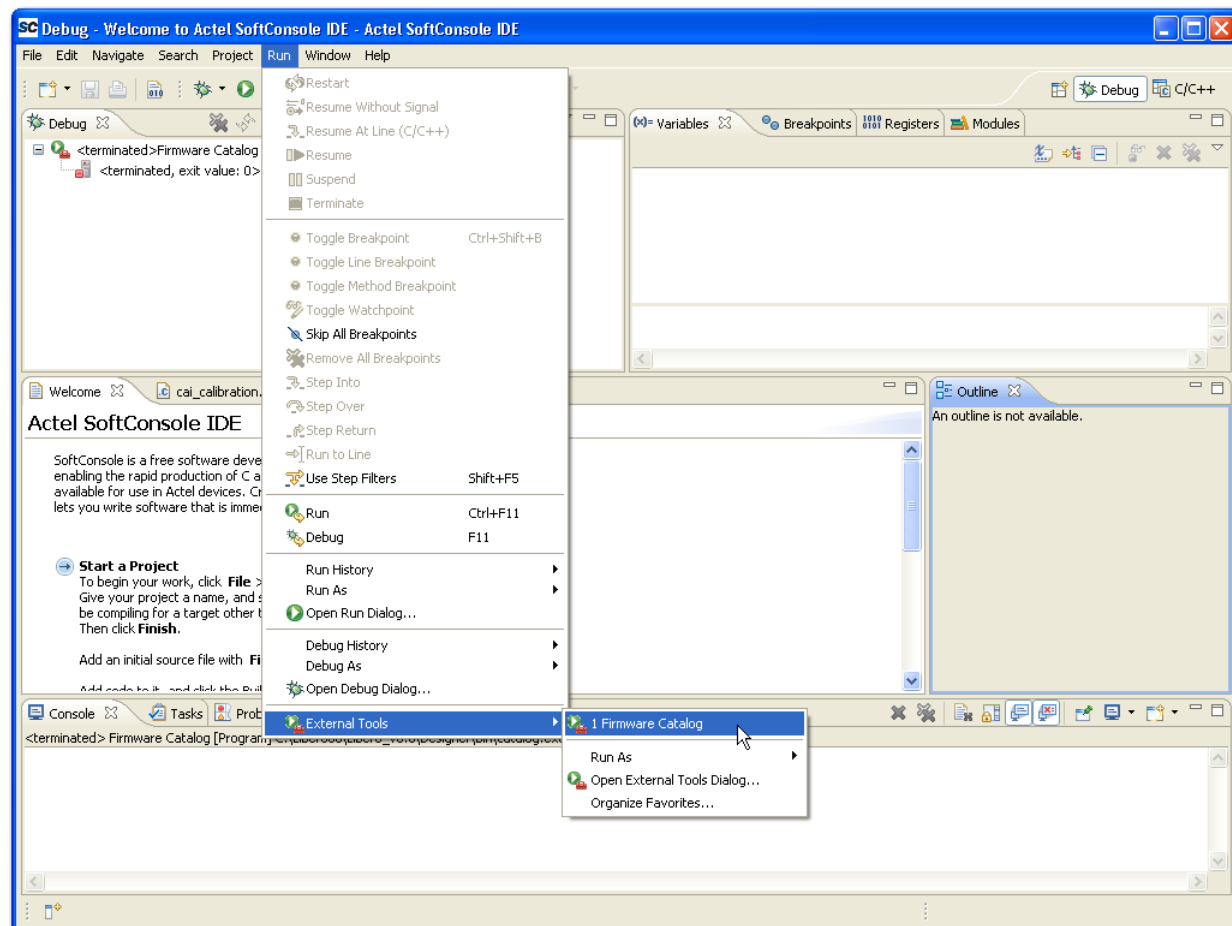


Figure 1-7 • Firmware Catalog

- To verify that the Firmware Catalog is linked to SoftConsole, click **Run** in SoftConsole again. You should see Firmware Catalog at the top of the list of External Tools (Figure 1-8).



**Figure 1-8 • Verify Firmware Catalog is Linked to SoftConsole**



---

## 2 – Tutorial – Create and Build Software Project

---

### Open SoftConsole

When you open SoftConsole for the first time, it asks you to specify the location of the workspace. You can select any location on your machine, keeping in mind that the workspace directory will contain any SoftConsole projects that you create during this session.

By default, SoftConsole builds your projects every time a source file is saved. For this tutorial, change this behavior by going to **Window > Preferences** and selecting **General > Workspace**. Clear the check box for Build automatically and click **OK**.

### Create and Build the Software Project

In this section you will create and build a SoftConsole project targeting a Cortex-M1 embedded processor system. Since this tutorial focuses on the tool flow, the source code is provided. Therefore, you will import the application source code, drivers, and hardware abstraction layer (HAL) into the project.

1. Invoke SoftConsole and open the workspace for the software project data (same workspace set up during step 1 of the ["Setup Firmware Catalog to Run in SoftConsole" section on page 1-11](#)). If some other workspace location is opened by default because the default location was selected, use **File < Switch Workspace** and navigate to the desired workspace.
2. Build Automatically is not enabled by default in SoftConsole v2.3. Use **Project < Build Automatically** to enable or disable Build Automatically. Make sure a check mark appears next to Build Automatically so that it is enabled.

3. Choose **File > New > Project**. The **New Project** window appears (Figure 2-1).

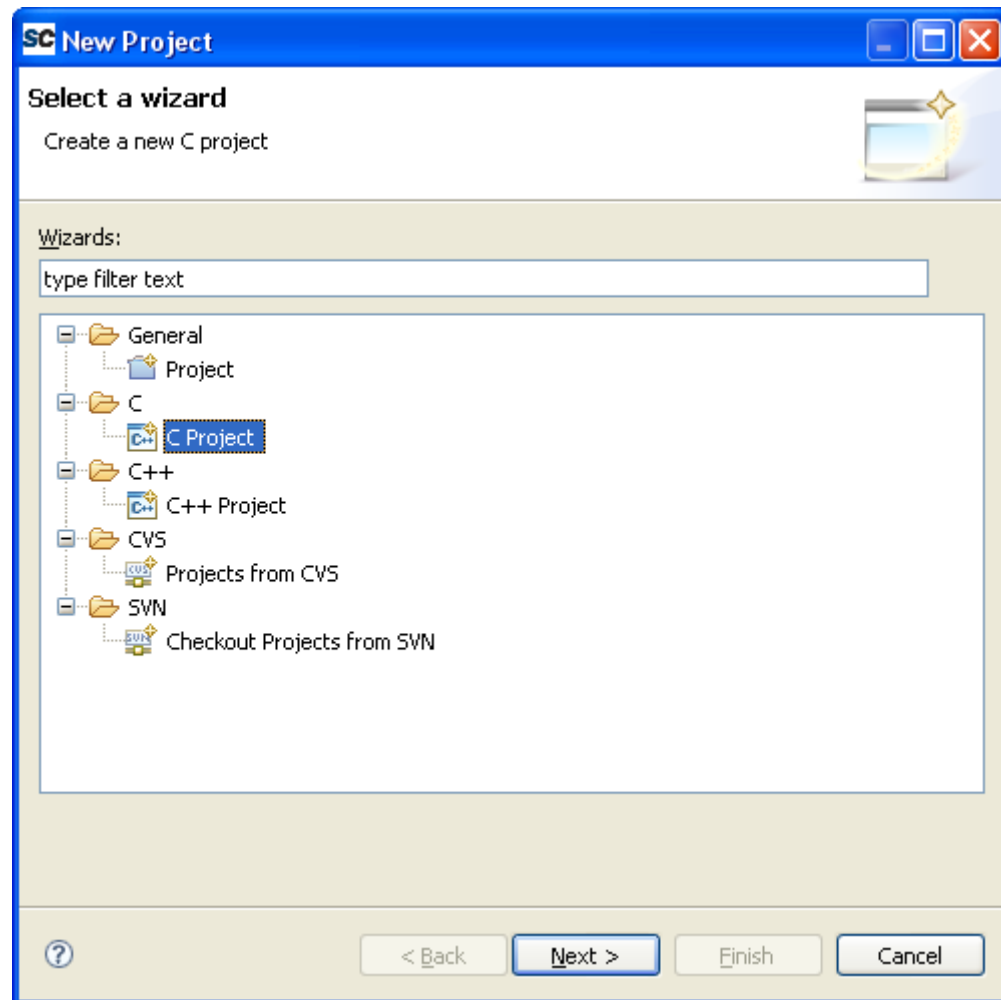


Figure 2-1 • New Project Window

4. Select **C Project** and click **Next**. The C Project window appears (Figure 2-2).

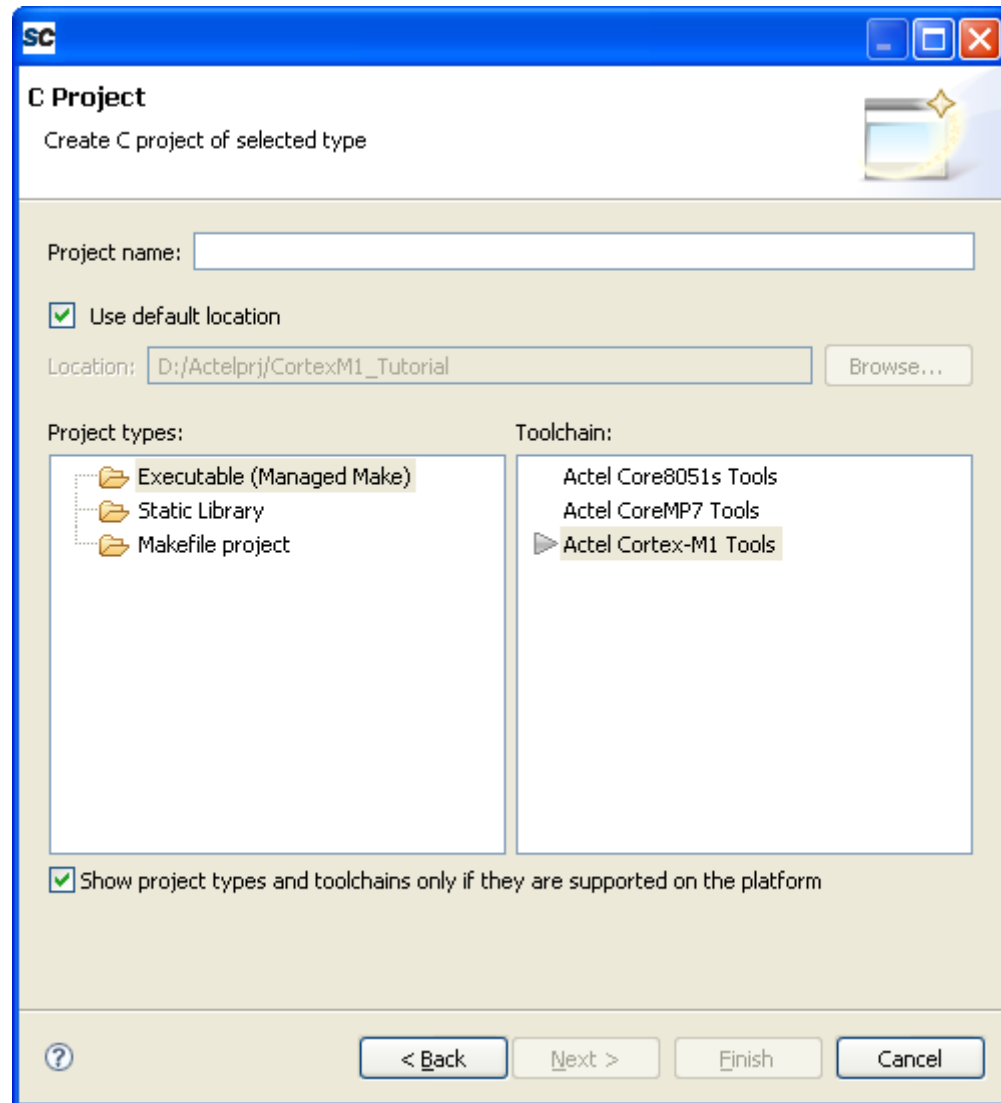


Figure 2-2 • C Project Window

5. Type CortexM1\_Tutorial in the Project name field, select **Actel Cortex-M1 Tools**, and click **Finish**. The CortexM1\_Tutorial project appears in the Navigator view on the left (Figure 2-3).

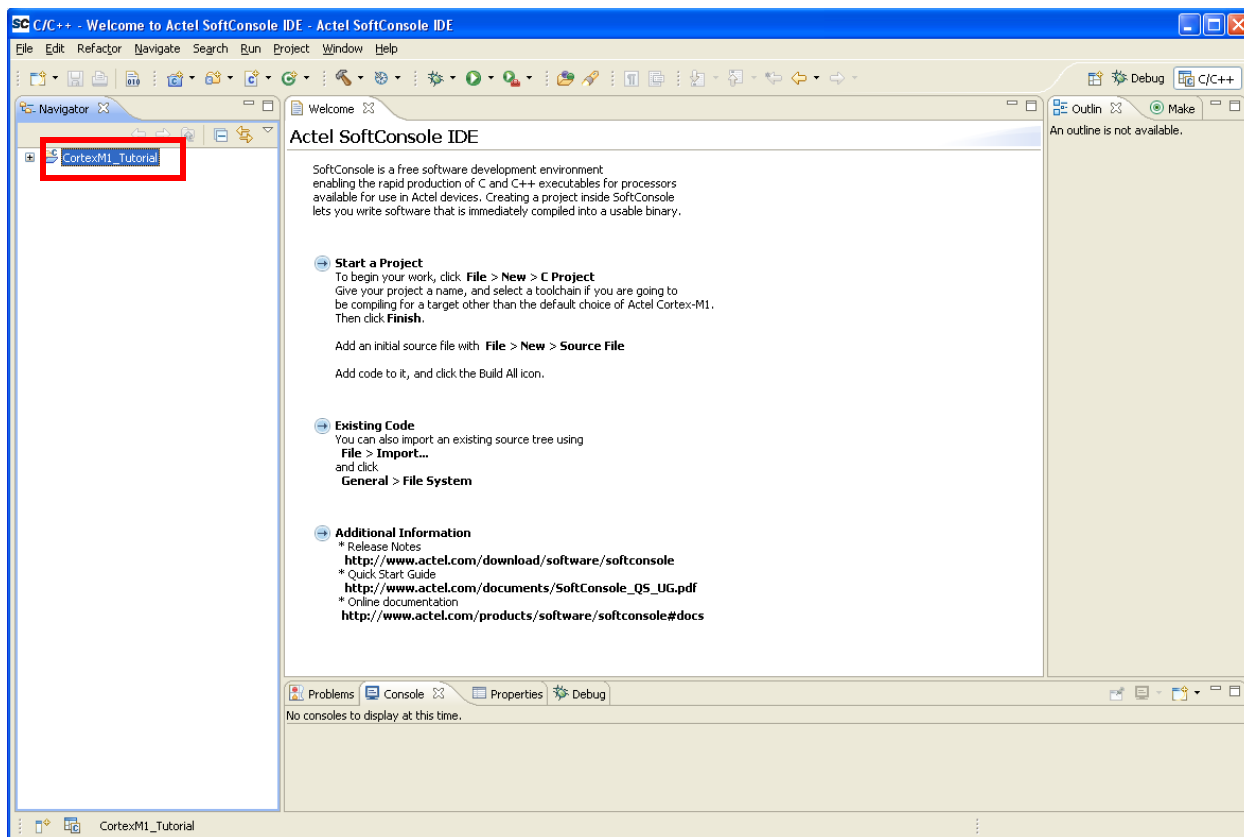


Figure 2-3 • Navigator View

6. Right-click on the project name and choose **Import**. You see the Import window appear.
7. Select **General > File System** and click **Next**.
8. Click the **Browse** button.
9. Browse to the location of the tutorial source files:  
`<zip file location>/CortexM1_Fusion_SW_Tutorial/<KIT PART NUMBER>/Tutorial_Files.`
10. Click **OK**.

[illegible]

11. Check the boxes next to `main.c`, `tutorial.h`, and `boot-from-actel-coreahbnvm.ld`.
12. Make sure that the **Create selected folders only** option is selected.
13. Click **Finish**. If you expand the project in the Navigator, you see the new files that have been imported into the project.
14. Go to **Run > External Tools > Firmware Catalog**.

The Firmware Catalog tool will launch and you should see a list of firmware cores in italics, indicating that these cores have not been downloaded to the local vault on your machine (Figure 2-5).

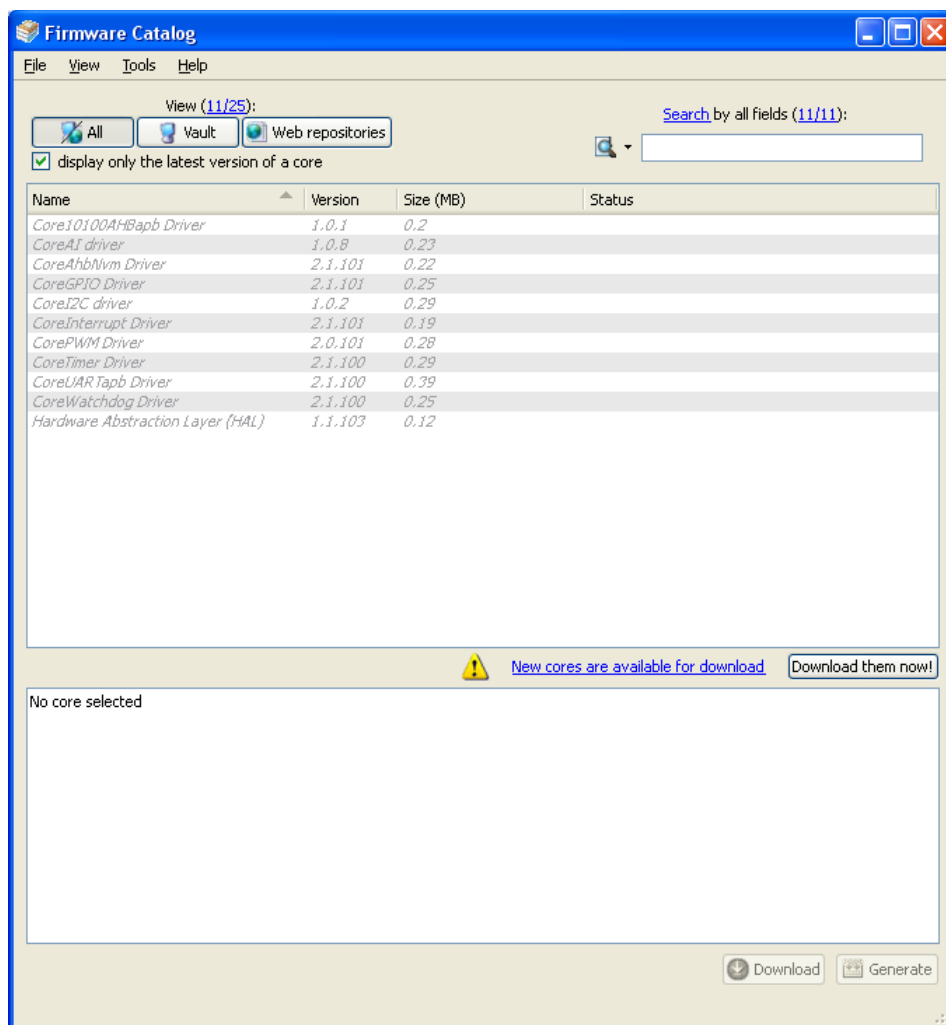
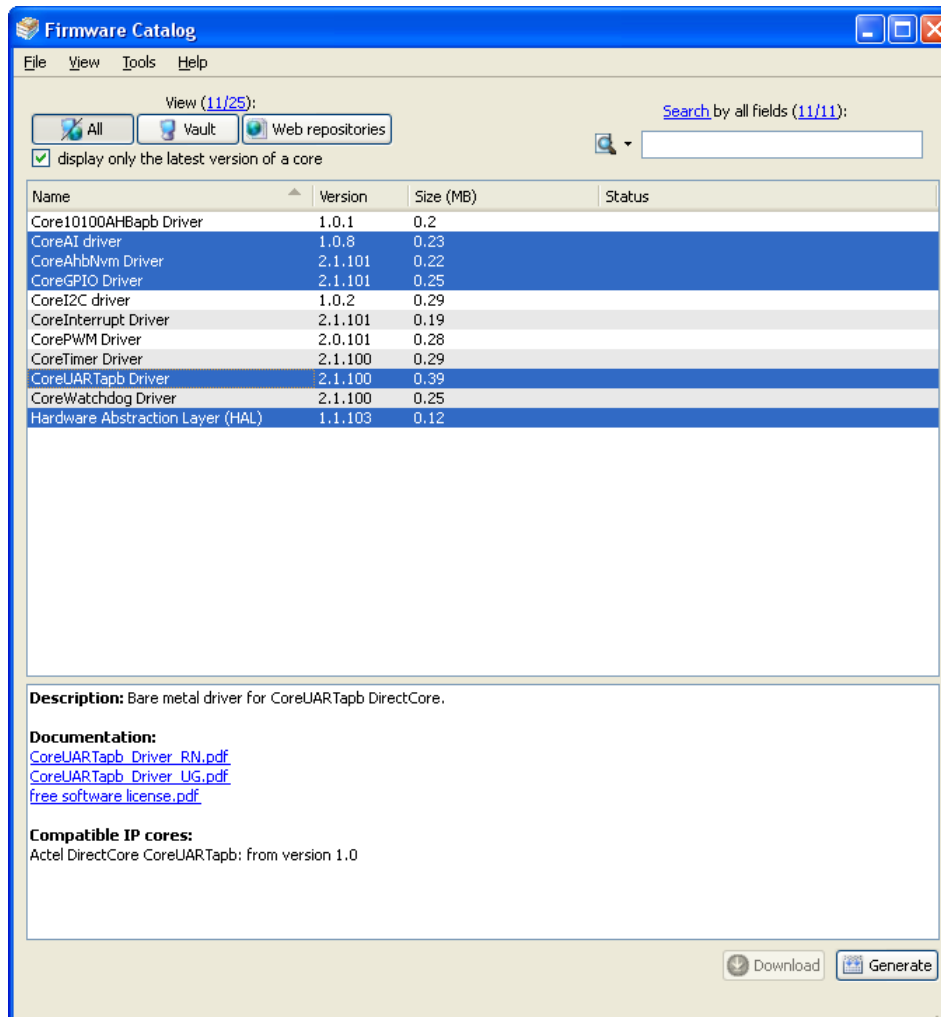


Figure 2-5 • Firmware Catalog

- Click the **Download them now!** button to download all of the firmware cores.

The Firmware Catalog will proceed to download each core to your local vault. The Vault location is common to all Actel software (Libero IDE Project Manager, Firmware Catalog, etc.). Changing your Vault location here updates the Vault location for all Actel tools you use on this machine. For more information, refer to the "Vault/Repositories Settings Dialog Box" entry in the Firmware Catalog help.

- Highlight the following firmware cores by using CTRL + right-click (Figure 2-6):
  - CoreAI Driver
  - CoreAhbNvm Driver
  - CoreGPIO Driver
  - CoreUARTapb Driver
  - Hardware Abstraction Layer (HAL)



**Figure 2-6 • Firmware Catalog Firmware Cores**

17. Click the **Generate** button. The Generate Options window appears.
18. Click the **Browse** button (with the ...) to the right of the Project folder text box.
19. Browse to your SoftConsole project directory (<...>/WORKSPACE/CortexM1\_Tutorial>). This tells Firmware Catalog to copy the firmware cores to this SoftConsole project.
20. Click **OK** two times.
21. The Configuring CoreAI\_Driver configuration window appears, which looks very similar to the CoreAI configuration window in SmartDesign (for configuring the hardware IP core). This driver must be modified based on the configuration of the IP core. Therefore, the same parameters must be entered in this window as were entered in the CoreAI configuration window in SmartDesign. This ensures that the driver configuration will match the hardware configuration. For this design, follow the steps below to achieve this.
22. Select Target device as **AFS1500**.
23. Select Calibration method as **None**.

24. Configure the quads as listed below while leaving the rest **Disabled**.
  - AV0 input: 0 to 4 Volts
  - AC0 input: Current monitor
  - AV1 input: 0 to 2 Volts
  - AC1 input: Current monitor
  - AT2 input: Temperature monitor
  - AC4 input: 0 to 4 Volts
25. Click **OK** to generate the firmware cores and place them into your SoftConsole project. You will see a generation report showing you what was done.
26. The Configuring HAL dialog box opens. Confirm that the processor is Cortex-M1 and the Software tool chain is SoftConsole; then click **OK**.
27. Clicking **Close** closes the report. To close the Firmware catalog, click the X in the upper right corner.  
Firmware Catalog has created subfolders in your SoftConsole project containing the four drivers and the Cortex-M1 HAL.
28. Right-click on your project in SoftConsole and choose **Refresh**. You should see the new folders and files created by Firmware Catalog. Specifically, you have a new *hal* folder (with subfolders) and a new *drivers* folder with a subfolder for each driver (four total).
29. Right-click your project and choose **Properties**.
30. Expand **C/C++ Build** on the left and select **Settings**. You see a window similar to the one in Figure 2-7.

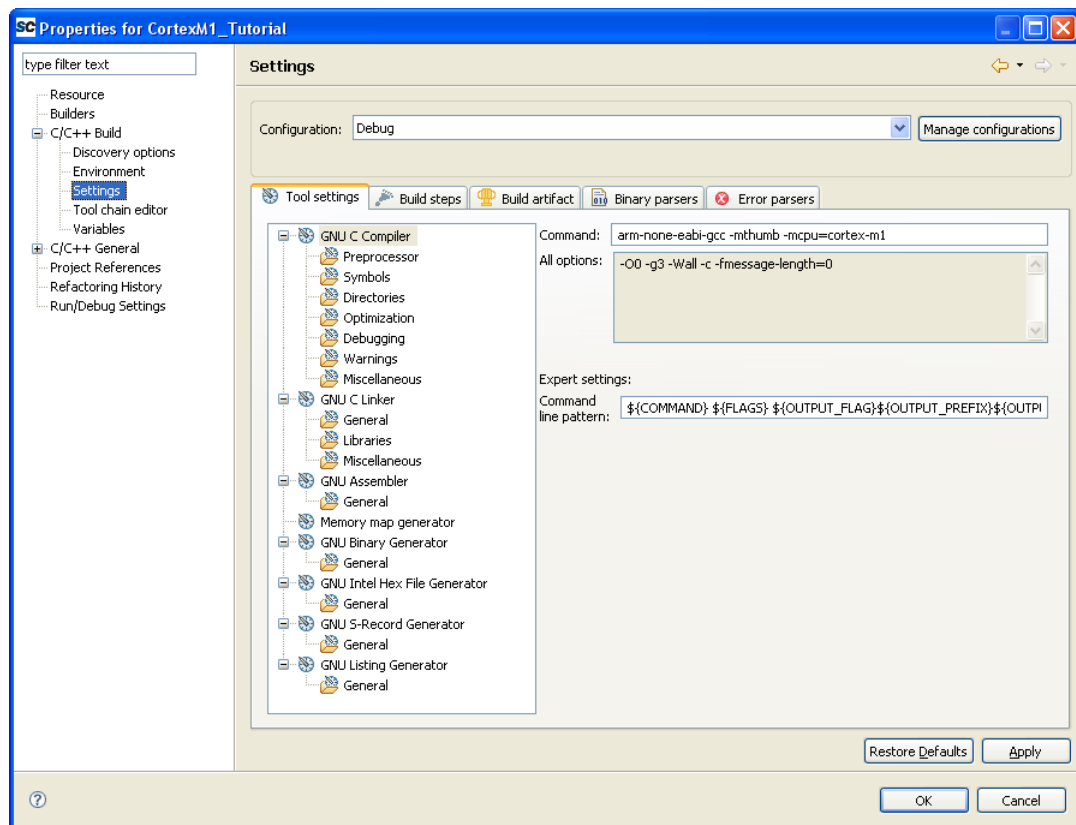

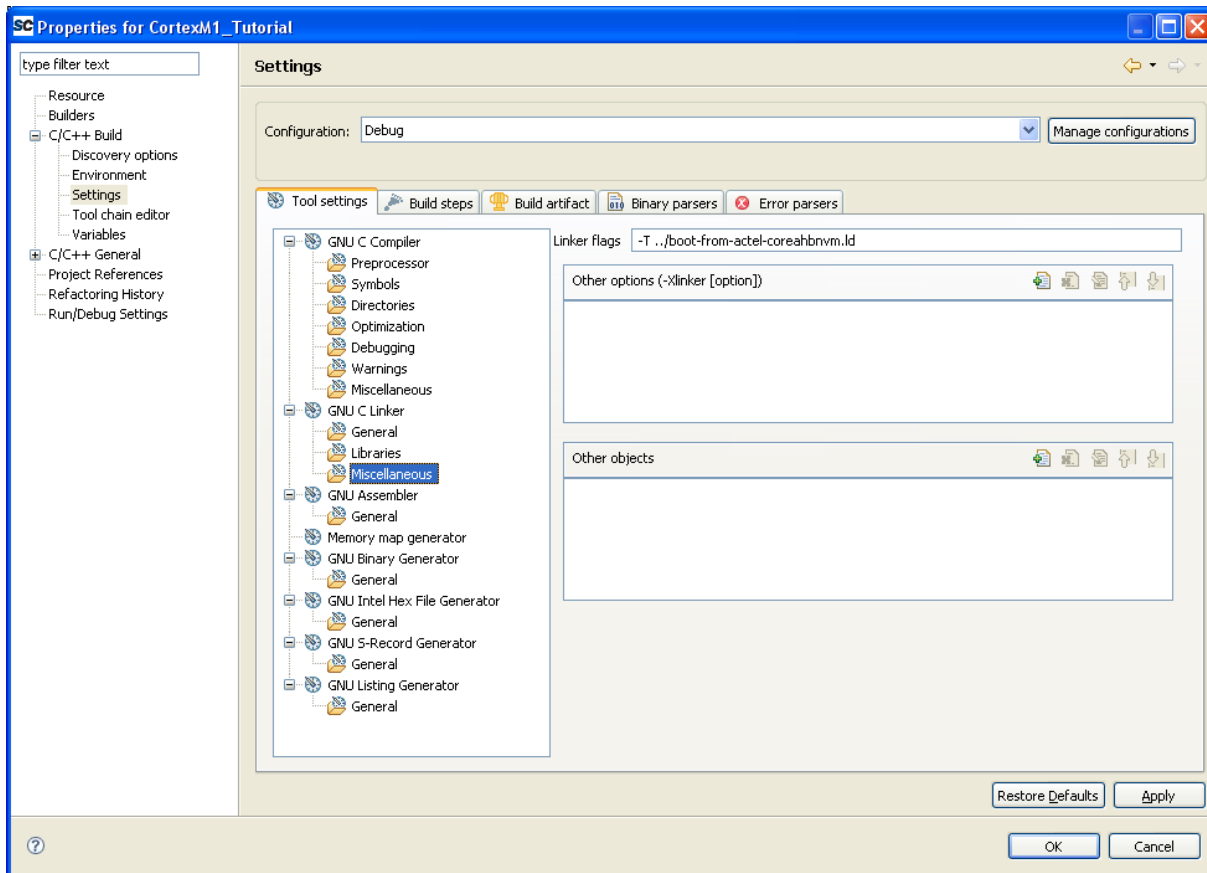


Figure 2-7 • Tutorial Properties



Notice the Configuration is set to Debug and the compiler options include the least optimization (-O0) and highest debug level (-g3). There is also a Release configuration which has the highest optimization (-O3) and least debugging (none). For this tutorial, you will debug the system, so you will only modify the Debug configuration.

31. Click **GNU C Compiler > Directories**.
32. Click the **Add** button .
33. Click the **Workspace** button.
34. Expand the CortexM1\_Tutorial project, select the `/CortexM1_Tutorial/hal` subdirectory, and click **OK** in the Folder Selection dialog box. Also click **OK** in the Add Directory Path dialog box.
35. Repeat the last three steps again for the following directories:
  - `CortexM1_Tutorial/hal/CortexM1`
  - `CortexM1_Tutorial/hal/CortexM1/GNU`
  - `CortexM1_Tutorial/drivers/CoreAhbNvm`
  - `CortexM1_Tutorial/drivers/CoreAI`
  - `CortexM1_Tutorial/drivers/CoreGPIO`
  - `CortexM1_Tutorial/drivers/CoreUARTApb`
36. Click **GNU C Linker > Miscellaneous**.
37. Enter `-T ../boot-from-actel-coreahbnvm.ld` in the Linker flags box, as shown in Figure 2-8.



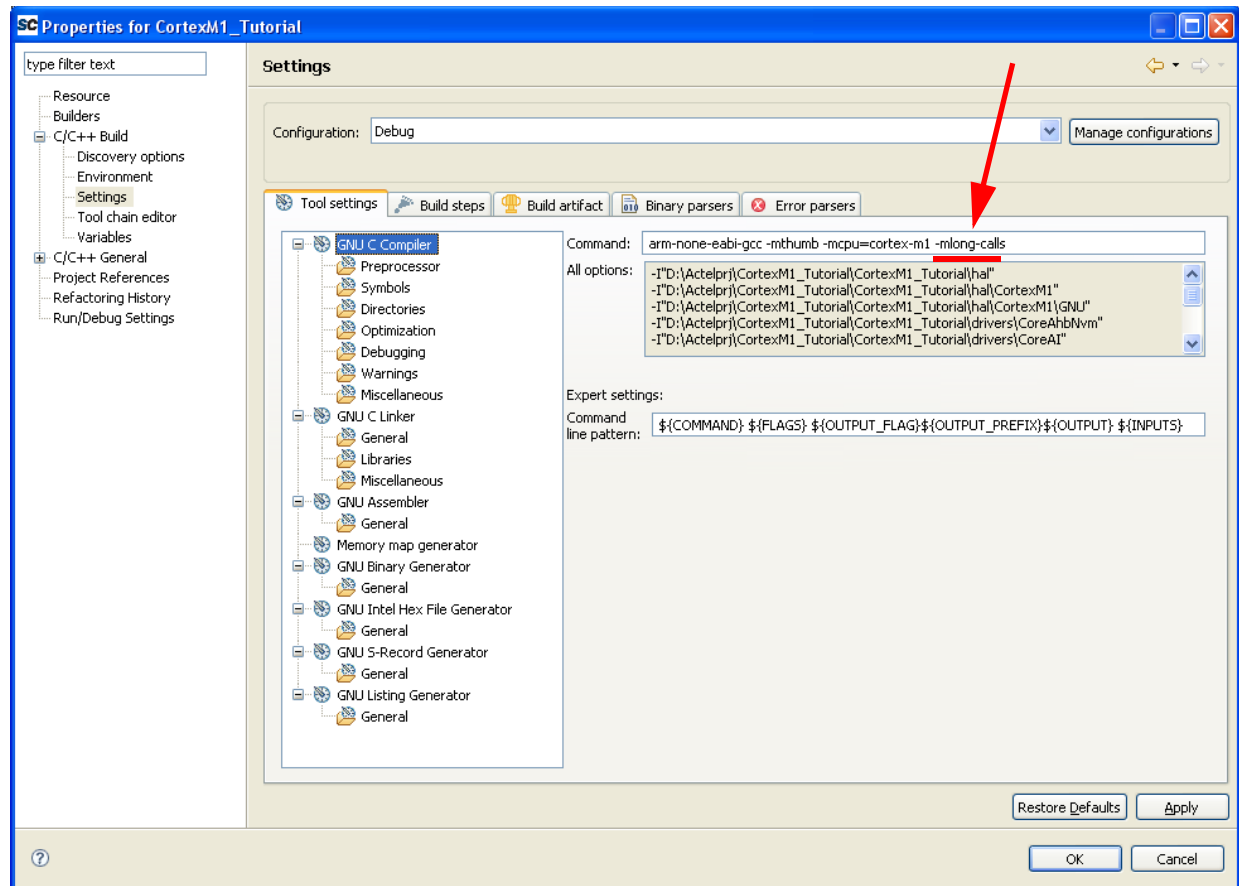
**Figure 2-8 • Linker Flags**

Example linker scripts are provided in the SoftConsole installation in the `<SoftConsole install>/src/Cortex-M1/linker-script-examples` directory. This linker script was copied from this

directory and modified for the target system. The system boots from on-chip NVM, copies the application code and data to external SRAM, and then runs the application code from external SRAM. For information about how to modify the linker script for your target system, refer to the “Start of board customization section” in the linker scripts.

38. Click **GNU C Compiler**.

39. Add **-mlong-calls** at the end of the Command box, as shown in [Figure 2-9](#).



**Figure 2-9 • GNU C Compiler**

**Note:** This flag is only necessary when booting from NVM and running from SRAM (for example, using `boot-from-actel-coreahbnvm.ld`). It is necessary to support the call to `main()` from `_start` since `_start` is in NVM, whose base address is `0x00000000`, and `main()` is being executed from SRAM, whose base address is `0x18000000`. For more information, refer to the GNU GCC documentation (**Start > Programs > Actel SoftConsole > Reference Documentation > GNU GCC Compiler Manual** or `<SoftConsole install>Sourcery-G++\share\doc\arm-arm-none-eabi\pdf\gcc\gcc.pdf`).

40. Click **OK** to save the project settings.

41. If Build Automatically is not selected, build the project by right-clicking on the project and choosing **Build Project**. You should not see any error in the build. The project is built with the Debug configuration settings and the corresponding files are created and placed in the Debug subfolder. You are now ready to debug your application.

---

## 3 – Tutorial – Programming and Debug

---

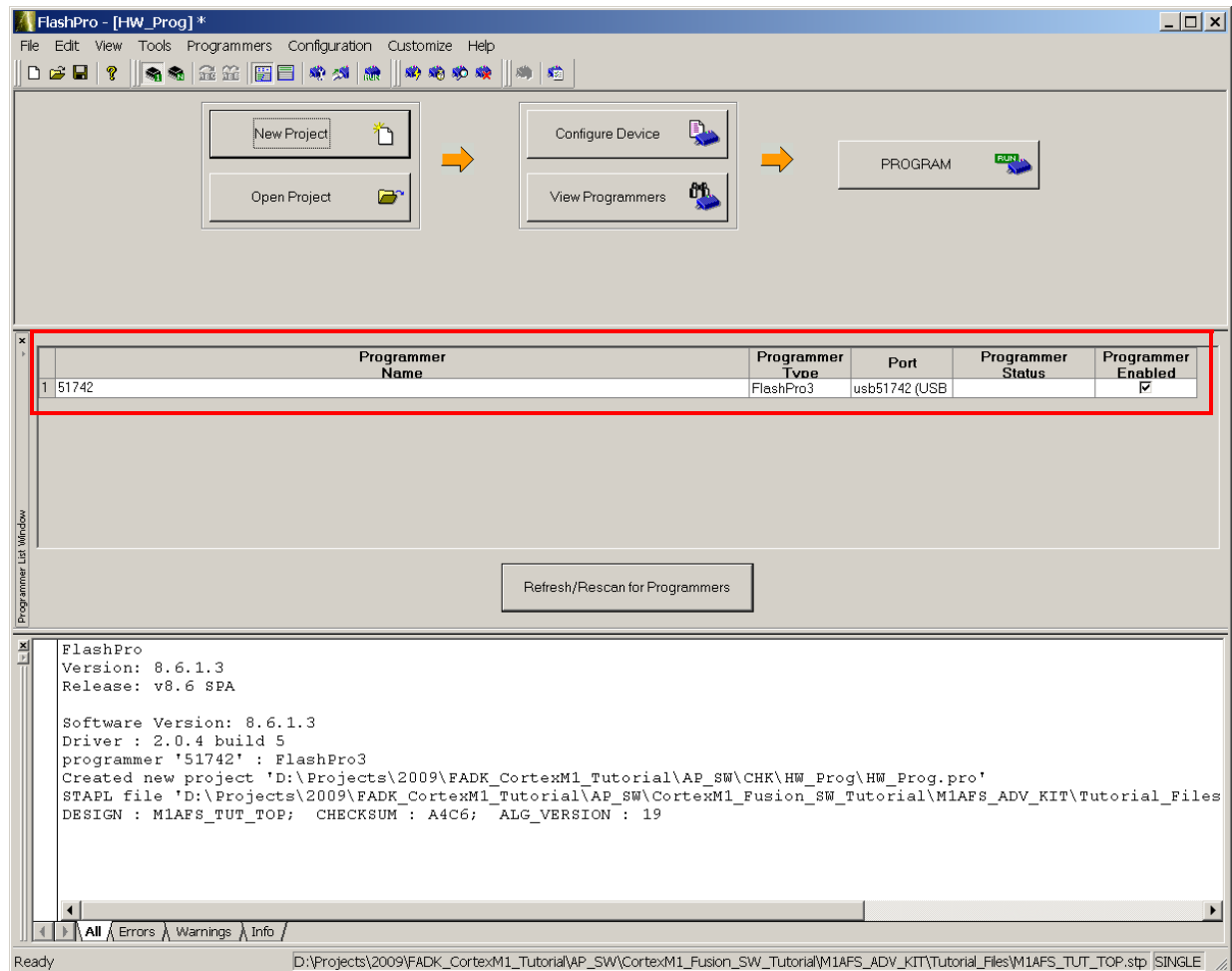
### Program the FPGA

Now you will use the standalone FlashPro software tool to program the hardware design into the target FPGA device. If you are familiar with launching FlashPro from within Libero IDE, as described in the [ARM Cortex-M1 Embedded Processor Hardware Development Tutorial](#), you can modify the steps below as necessary.

**Note:** Make sure the connections listed in the "Before You Get Started" section on page 1-7 have been made.

1. Launch the FlashPro software from the Windows Start menu.
2. Create a new FlashPro project by going to **File > New Project**.
3. Enter any project name and location you wish. Make sure there are no spaces in the folder names. Also, confirm that **Single device** is selected in the Programming mode field.
4. Click the **Configure Device** button.
5. Click the **Browse** button and select the <FPGA>\_TUT\_TOP.STP programming file. You will see the programming file information appear.

- Click the **View Programmers** button and verify that a programmer has been identified in the Programmer Name list, as shown in Figure 3-1.



**Figure 3-1 • FlashPro Programmer List**

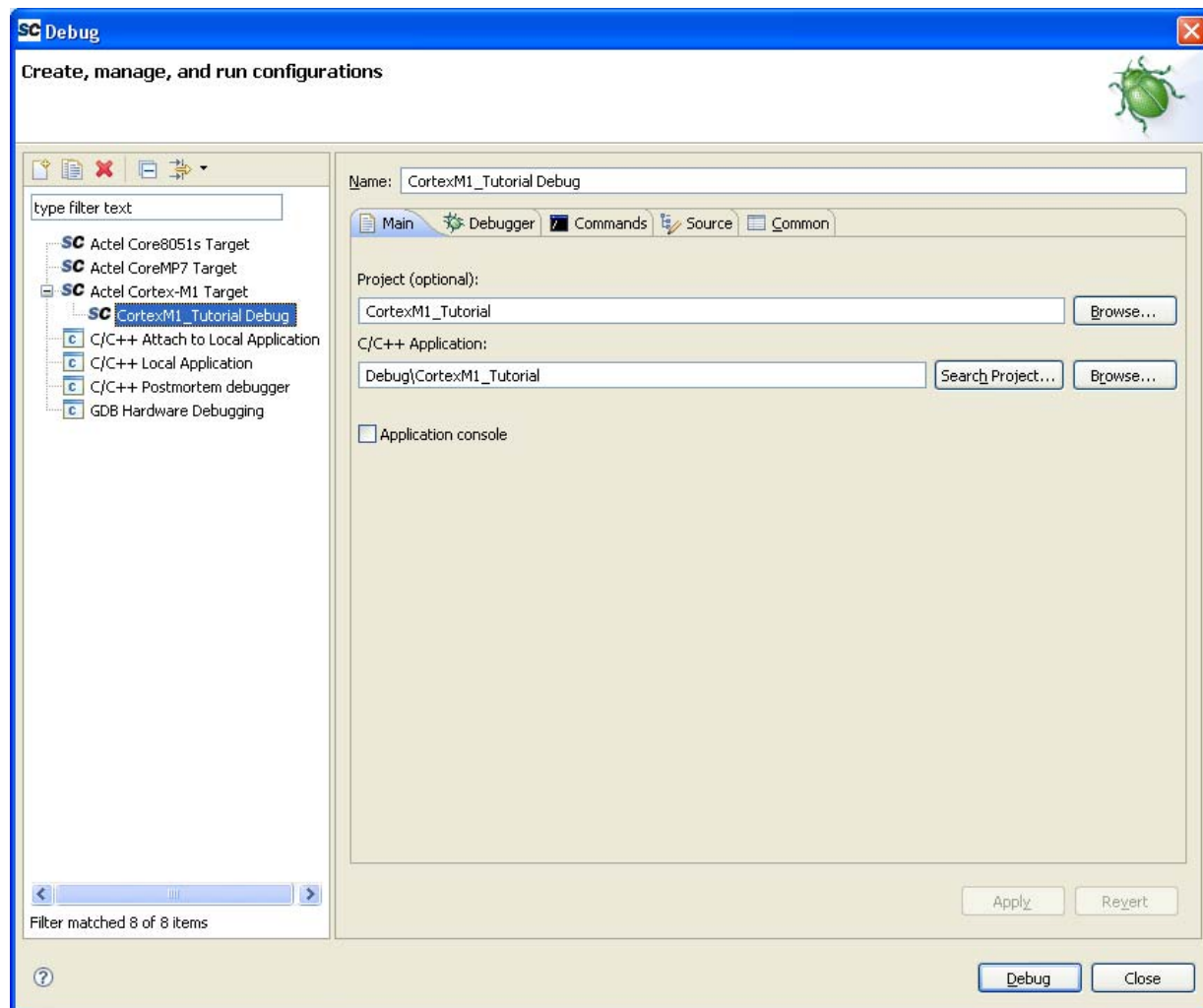
If no programmer name is shown, click the **Refresh/Rescan for Programmers** button. If no programmer is found, make sure that a USB cable is connected between your PC and the FlashPro3 programming stick and that you have the FlashPro3 programmer drivers installed.

- Click the **PROGRAM** button to download the FPGA design. The FlashPro Programmer will program the FPGA array.
- After successful programming a Run Passed message is displayed under Programmer Status. Press the reset switch (SW1) on the board.
- Close the FlashPro software. Answer **Yes** if prompted to save files. Next time you want to program the FPGA with this design you can open this project.

## Create the Cortex-M1 Debug Configuration

For each Cortex-M1 software application that you want to debug, you need to create a Cortex-M1 Debug configuration.

1. Right-click the CortexM1\_Tutorial project and select **Debug As > Open Debug Dialog**.
2. Right-click Actel Cortex-M1 Target and choose **New**.
3. Click **Apply**. You should see a window like the one in Figure 3-2.



**Figure 3-2 • Create New Debug Configuration**

A debug configuration called CortexM1\_Tutorial Debug is created, which is targeted for the Debug build configuration of your CortexM1\_Tutorial project.

## Launch Debug Session

SoftConsole uses the GNU GDB debugger, in conjunction with the Cortex-M1 Sprite, to support loading and debugging of programs for the Cortex-M1 processor. The executable file is loaded into program memory before debugging can take place. Before launching the debug, make sure the HyperTerminal setup is active and connected to the board (refer to the "Set Up HyperTerminal" section on page 1-9).

1. Click the **Debug** button in the Debug configurations window. Or, right-click the CortexM1\_Tutorial project, select **Debug As > Open Debug Dialog...** and click the **Debug** button.

The program code and data are downloaded to the target and the debug session starts. When the Fusion NVM is being programmed, you might receive messages in red, similar to those shown in [Figure 3-3](#). This indicates the data is being read from the NVM and compared. It is not an error message.

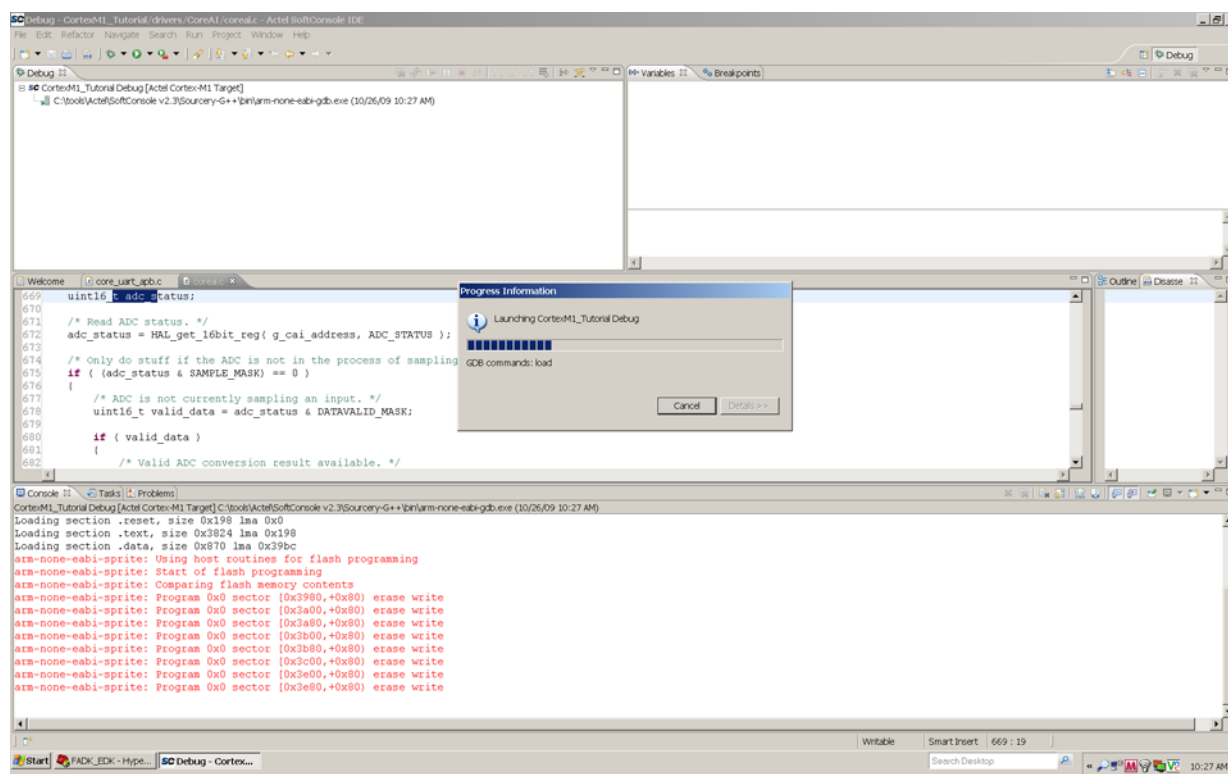


Figure 3-3 • NVM Programming Message

You will receive the following window (Figure 3-4).

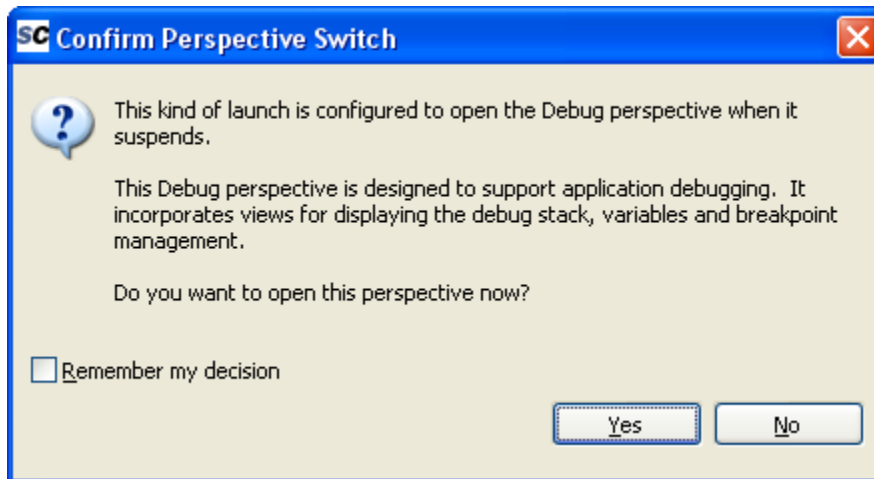


Figure 3-4 • Confirm Perspective Switch

- Click **Yes** to switch from the C/C++ perspective to the Debug perspective. SoftConsole switches to the Debug perspective and you should see something similar to Figure 3-5.

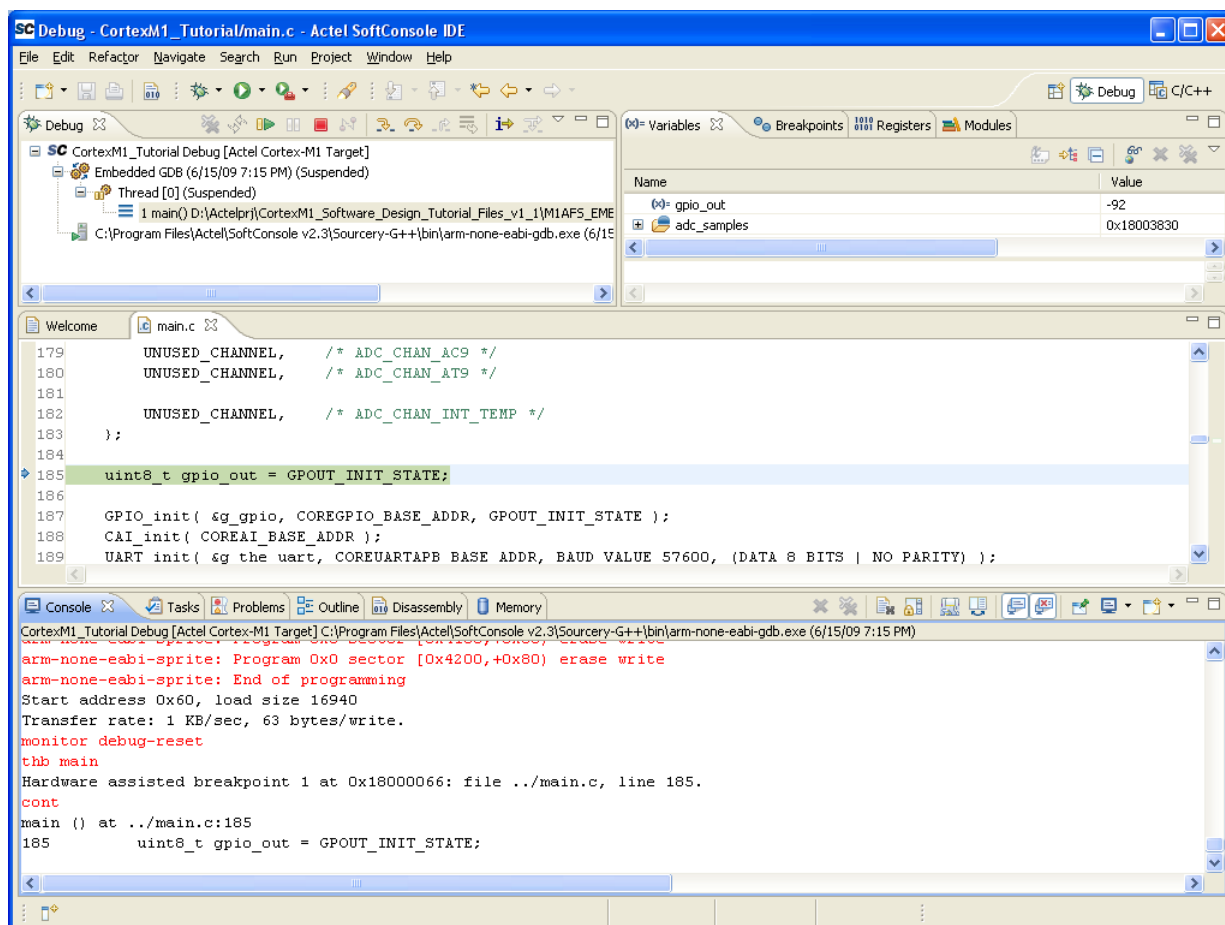





Figure 3-5 • Debug Perspective

The Debug View in the upper left shows the debug communication client running as well as the call stack in your application.

Note that the processor is currently suspended and the resume button  and step buttons   are enabled. A breakpoint is always set at the first executable line of code in `main()`. You see this line of code highlighted in `main.c`.

In the upper right there are views which show you various processor views, including the current variables, breakpoints, processor registers and modules. You can watch these values change as you step through the code.


Some views may not be displayed by default. You can go to **Window > Show View** to add them. If you add the Memory view, you will see a Memory tab at the bottom. Here you can view the values stored in memory, watch the values change, and modify the memory in the target. You can add a memory monitor by clicking the green plus sign, typing in the address, and clicking **OK**. You can add as many memory monitors as needed.

## Continue Debug Session for Fusion-Based Kits

The application code initializes the UART peripheral and the analog quads and ADC of the analog block of the Fusion FPGA. Then a message is sent to the UART, prompting the user to hit any key. Once the user hits a key, the Cortex-M1 processor reads and displays the following information from the analog sources on the board (analog channel in parentheses):

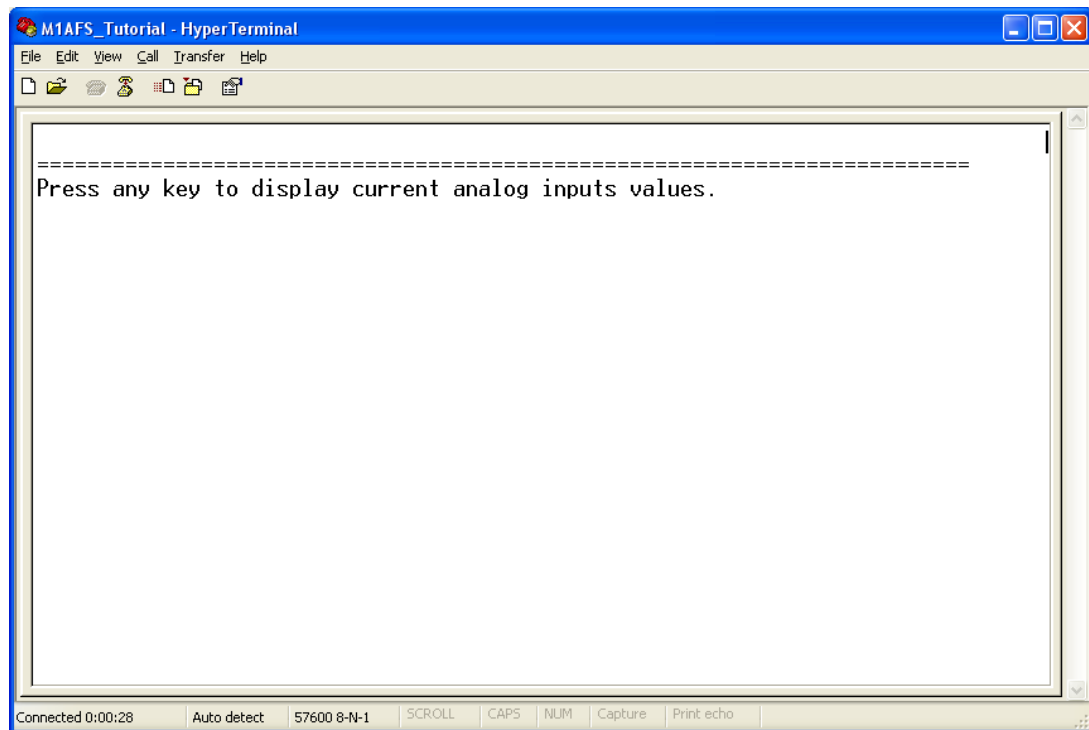
- 3.3 V supply voltage (AV0)
- 3.3 V supply current (AV0 and AC0)
- 1.5 V supply voltage (AV1)
- 1.5 V supply current (AV1 and AC1)
- Ambient temperature (AT2)
- Voltage of potentiometer RV1 (AC4)

The code then returns to the loop, waiting for a key to be pressed. Follow the steps below to run and debug the application.

1. Start your application by clicking the Resume button . The LEDs on the board should start counting to indicate that the application is running.

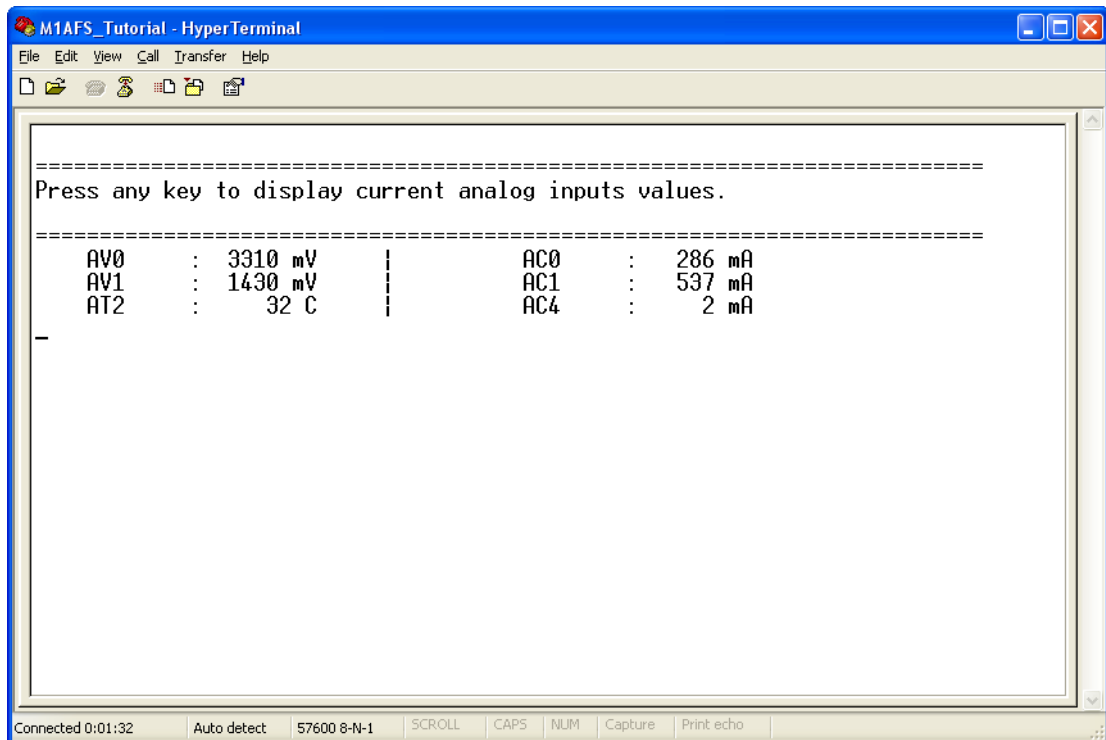


2. Go to your HyperTerminal window. You should see the message shown in [Figure 3-6](#) in the HyperTerminal window.




**Figure 3-6 • HyperTerminal Window Message**

3. Press any key. You should see something similar to [Figure 3-7](#) the in the HyperTerminal window.



**Figure 3-7 • HyperTerminal Analog Inputs Values**

4. Change the potentiometer (RV1) on the target board.
5. Go back to HyperTerminal and press any key. You should see AC4 change value, based on the potentiometer (RV1). The full range is about 0 V to 3.3 V.
6. Click the Pause button  to stop the processor from executing code. Notice that the call stack is updated and the next line of C code is highlighted.

7. Set a breakpoint at the line of code highlighted in [Figure 3-8](#) by double-clicking in the left margin next to the line of code. This line of code is near line 180.



```

188 CAI_init( COREAI_BASE_ADDR );
189 UART_init( &g_the_uart, COREUARTAPB_BASE_ADDR, BAUD_VALUE_57600, (DATA_8_BITS | NO_PARITY) );
190
191 display_start();
192
193 while( 1 )
194 {
195     GPIO_set_output( &g_gpio, ++gpio_out);
196     delay(0xFFFF);
197
198     CAI_round_robin( adc_samples );
199
200     if ( key_pressed() )
201     {
202         process_samples( adc_samples );
203     }
204 }
205

```

**Figure 3-8 • Set Breakpoint**

You will see a dot appear in the margin indicating that a breakpoint is set on that line of code.





8. Click the Resume button  to continue executing code. The debugger should stop at the line of code where you set a breakpoint.
9. Go to **Window > Show View > Registers**. If you do not see Registers as an option, then this view is already open (that is, you do not need to do anything for this step).
10. Expand the registers in the upper right window so you can see the values.
11. Click the Step Over button  and watch the values of the registers change. Any registers that changed value should be highlighted.

Congratulations! You have successfully created a Cortex-M1 software project and debugged it, running on an Actel FPGA development kit. For more information on features in SoftConsole, refer to the ["Appendix" on page A-37](#). The ["Appendix"](#) also includes instructions for updating the FPGA programming file (STP file) with new program code and data contents for the NVM block inside Fusion devices.



## A – Appendix

### Debugging Features in SoftConsole

- You can set a breakpoint using any of the following methods:
  - Place the cursor on the line of code and select **Run > Toggle Breakpoint**.
  - Place the cursor on the line of code, right-click, and choose **Toggle Breakpoint**.
  - Double-click the left margin next to the line of code.
- The upper right window has various tabs, including breakpoints, registers, and variables. Elements are highlighted if they have changed value since the last time the processor was stopped.
- You can single step through the source code by choosing **Run > Step Into** or **Run > Step Over** or clicking their corresponding icons,  and  respectively.
- The Step Return button  steps to the next instruction of the calling function. You can also use the menu by going to **Run > Step Return**.
- To view disassembly, choose **Window > Show View > Disassembly**. You can then click the Instruction Stepping Mode button  to step by processor instruction.

### Updating Contents of NVM in PDB file

In the tutorial we used an STP file for programming the hardware. If you want to update the hardware with the new application code inside the NVM block, you can update the content of NVM file in the PDB file, and program only the NVM.

The build process used for debug can also be used to build the release configuration. The \*.hex file generated in the release configuration is used to build NVM client content (refer to "Step 3 – Create Flash Memory System" in the [ARM Cortex-M1 Embedded Processor Hardware Tutorial](#)), which generates the \*.efc file needed by the programmer.

- Open the FlashPro tool.
- Click the **Configure Device** button.
- Click the **Browse** button and select the <FPGA>\_TUT\_TOP.PDB programming file. You will see the programming file information appear.
- Click the **PDB Configuration** button. The Flashpoint – Programming File Generator window opens. You use this window to modify the contents of the embedded flash memory (NVM) in the FPGA programming file.
- Click the **Modify** button. The Modify Embedded Flash Memory Block window opens. FlashPro will read the programming file and obtain the path for the (embedded flash memory configuration) file (\*.efc) and Intel hex file (\*.hex) for the NVM block. If either file has been modified since it was last imported, you will see a yellow triangle near the file.
- Click the **Import Configuration File** button.
- Browse to the location of your EFC file, which describes the configuration of the NVM block in your design. Libero IDE places it in the <Libero project folder>smartgen<name of NVM block> directory.
- Click **Import**.
- Click the **Import content** button and browse to the location of your \*.hex file, which contains the program code and/or data. SoftConsole automatically creates a \*.hex file containing the program code and data, and places it in the <SoftConsole project>/<Debug or Release> folder.
- Click **Import**.
- Click OK, and then click **Finish**.



## B – List of Document Changes

The following table lists critical changes that were made in the current version of the document.

Previous Version	Changes in Current Version (50200162-1)	Page
50200162-0 (March 2009)	The "Requirements for the Tutorial" section was revised. FlashPro v8.6 or newer is required. The M1AFS-ADV-DEV-KIT development board with power supply is now supported for this tutorial.	5
	The "Tutorial Files" section was revised. The folder name for the tutorial files changed. The names of the programming file and header file changed. Figure 1 • Directory Structure for M1AFS-EMBEDDED-KIT Board was replaced.	6
	Table 1-1 • Board Connections was revised to add the M1AFS-ADV-DEV-KIT board.	7
	Instructions for installing Firmware Catalog were added to the "Before You Get Started" section.	7
	The "Tutorial – Create and Build Software Project" section was revised to include use of Firmware Catalog.	17
	The "Tutorial – Programming and Debug" section was revised to conform with other changes in the tutorial made for the incorporation of Firmware Catalog.	27
	Figure 3-5 • Debug Perspective was replaced.	31
	Figure 3-8 • Set Breakpoint was replaced.	35
	The "Appendix" was revised to add some additional steps to the "Updating Contents of NVM in PDB file" section.	37





---

## A – 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/support/search/default.aspx](http://www.actel.com/support/search/default.aspx)) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

### Website

You can browse a variety of technical and non-technical information on Actel's home page, at [www.actel.com](http://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](mailto: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](mailto:tech@actel.com)) or contact a local sales office. Sales office listings can be found at [www.actel.com/company/contact/default.aspx](http://www.actel.com/company/contact/default.aspx).

---

# Index

---

## A

Actel  
    electronic mail 41  
    telephone 42  
    web-based technical support 41  
    website 41

## B

board connections 7  
breakpoint 35

## C

COM port 7, 9  
contacting Actel  
    customer service 41  
    electronic mail 41  
    telephone 42  
    web-based technical support 41  
Cortex-M1 debug configuration 29  
create new project 17  
customer service 41

## D

debug  
    Fusion-based kits 32  
    launch session 30  
debug configuration 29  
debug perspective 31  
debugging 37  
driver  
    installation 7  
driver setup 7

## F

Firmware Catalog 22  
firmware cores  
    download 22  
FlashPro 27

## G

GNU C Compiler 26

## H

HyperTerminal  
    analog inputs 34  
    setup 9

## P

PDB file  
    updating contents 37  
product support 42  
    customer service 41  
    electronic mail 41  
    technical support 41  
    telephone 42  
    website 41

## S

SoftConsole 17, 37

## T

technical support 41  
tutorial  
    build project 26  
    files 5  
    files, description 6  
    files, directory structure 5  
    import files 20  
    steps included 5

## U

USB driver 7  
USB-to-UART driver 7

## W

web-based technical support 41

---

Actel, IGLOO, Actel Fusion, ProASIC, Libero, Pigeon Point and the associated logos are trademarks or registered trademarks of Actel Corporation. All other trademarks and service marks are the property of their respective owners.



**Actel is the leader in low-power and mixed-signal FPGAs and offers the most comprehensive portfolio of system and power management solutions. Power Matters. Learn more at [www.actel.com](http://www.actel.com).**

**Actel Corporation**

2061 Stierlin Court  
Mountain View, CA  
94043-4655 USA

**Phone** 650.318.4200

**Fax** 650.318.4600

**Actel Europe Ltd.**

River Court, Meadows Business Park  
Station Approach, Blackwater  
Camberley Surrey GU17 9AB  
United Kingdom

**Phone** +44 (0) 1276 609 300

**Fax** +44 (0) 1276 607 540

**Actel Japan**

EXOS Ebisu Building 4F  
1-24-14 Ebisu Shibuya-ku  
Tokyo 150 Japan

**Phone** +81.03.3445.7671

**Fax** +81.03.3445.7668

<http://jp.actel.com>

**Actel Hong Kong**

Room 2107, China Resources Building  
26 Harbour Road  
Wanchai, Hong Kong

**Phone** +852 2185 6460

**Fax** +852 2185 6488

[www.actel.com.cn](http://www.actel.com.cn)