
Core8051s Debugging in Axcelerator and RTAX-S Devices

Table of Contents

Introduction	1
Core8051s Debug Configuration	1
Design Example	2
Conclusion	11
Appendix A – Software and Hardware Tools Requirement	11
Appendix B – FS2 and Keil µVision3 Interface	11
Appendix C – Running Debugger Using FS2 Debugger	11

Introduction

This application note describes Core8051s debug operation in Axcelerator[®] and RTAX-S devices. Core8051s is a high-performance, 8-bit microcontroller IP Core. It is a fully functional 8-bit embedded controller that executes all ASM51 instructions and has the same instruction set as the 80C31. Core8051s can run programs written for the 8051. It contains only the 8051 core logic without any peripherals. Debug operation for Core8051s can be done using the UJTAG macro available in Microsemi[®] SoC Products Group flash devices or user I/Os. The core configurator in Libero[®] Integrated Design Environment (IDE) allows you to choose the right option and run a debug operation. When the debug option is selected, the on-chip instrumentation (OCI) debug functionality is enabled and you can connect a debugger to the processor through a JTAG connection. In RTAXS-S and Axcelerator families, you can only run debug operation using the user I/O option because these devices do not support the UJTAG macro. This application note describes how to perform Core8051s debug operation using the user I/Os option in the AX250-PQ208 device with a design example.

Core8051s Debug Configuration

Core8051s is a processor core and is compatible with the instruction set of the 8051 microcontroller. There are three debug-related configuration options. Refer to the [Core8051s Handbook](#) for details. When configuring Core8051s, you can choose from several debug modes. From the Debug drop-down menu, you can choose the following options:

- Disabled: Core8051s excludes debug functionality.
- Enabled using UJTAG: Core8051s allows inclusion of debug functionality and use of the dedicated JTAG pins of the device (through the UJTAG macro) for the debug connection.
- Enabled using I/Os: Core8051s allows inclusion of debug functionality and use of general purpose I/O pins for the debug connection. This option should be used if the UJTAG macro is either not present on your device or is already in use and not available for the Core8051s debug connection.

When the Debug option is enabled using UJTAG or user I/Os, two additional debug options are made available for added control over the debug functionality:

- Include trace RAM: A 256-byte deep trace RAM within Core8051s can be included. Including the trace RAM increases the tile count for the processor and consumes RAM blocks on the device.
- Number of hardware triggers/breakpoints: The maximum number of hardware triggers/breakpoints available can be set when debugging a Core8051s system. The options are 0, 1, 2, or 4. Increasing the number of hardware triggers/breakpoints increases the tile count of the processor.

You should choose the debug option based on the family or design being used. Note that the UJTAG macro is only available for Microsemi SoC Products Group flash-based FPGA families. You have the choice of using either option in Microsemi SoC Products Group flash-based FPGAs because of the availability of the UJTAG macro. However, there is no UJTAG macro available on RTAX-S or Axcelerator devices – hence use of the dedicated JTAG interface for soft processor debug is not an option. When implementing Core8051s on an RTAX-S or Axcelerator device, you are restricted to the “Enabled using I/Os” option for running the debugger. In addition, when the normal user I/Os are used for the debug connection, a soft JTAG test access port (TAP) is created in the FPGA fabric, consuming a few more tiles.

The following sections explain how to run debug mode (OCI Block with user I/O) with a design example in Axcelerator or RTAX-S devices.

Design Example

The design files referred to in this application note are available for downloading. The design example can be downloaded from www.microsemi.com/soc/download/rsc?f=CORE8051s_AC354_DF and it includes complete Libero IDE and Keil project. The design example displays a rotating LED pattern using a push-button switch. The design uses Core8051s, program and data memory, CoreAPB3, CoreInterrupt, CoreGPIO, and PLL.

Figure 1 shows a high-level block diagram of the design and Figure 2 on page 4 shows the Core8051s configuration setting. The **Enable user I/Os** option is selected for debug.

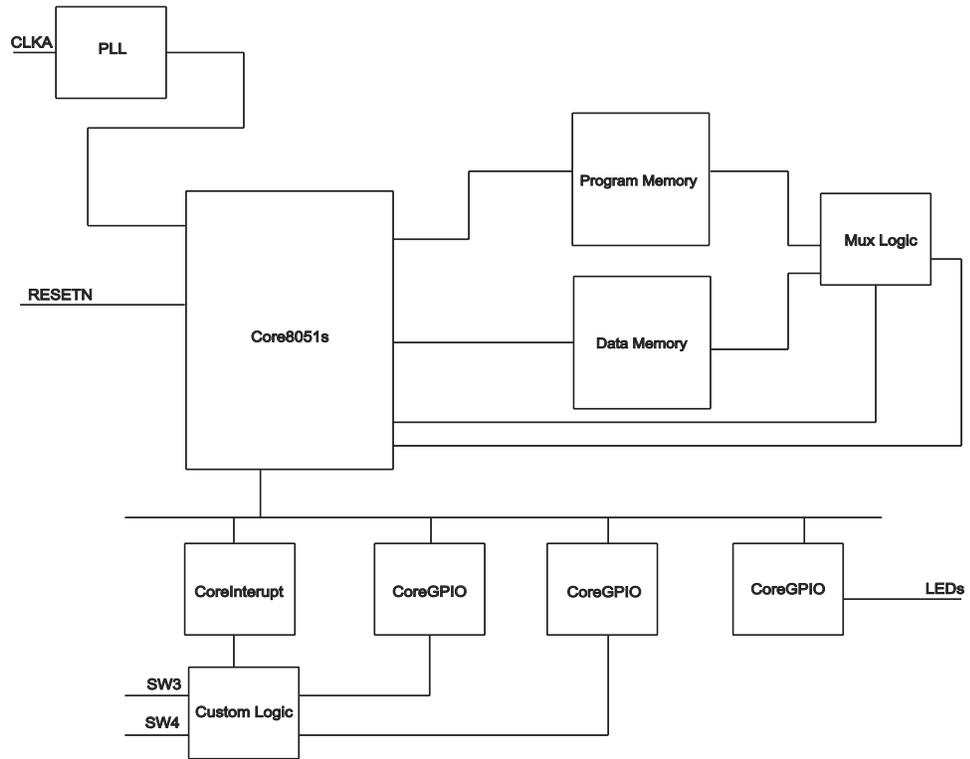


Figure 1 • Design Example Block Diagram

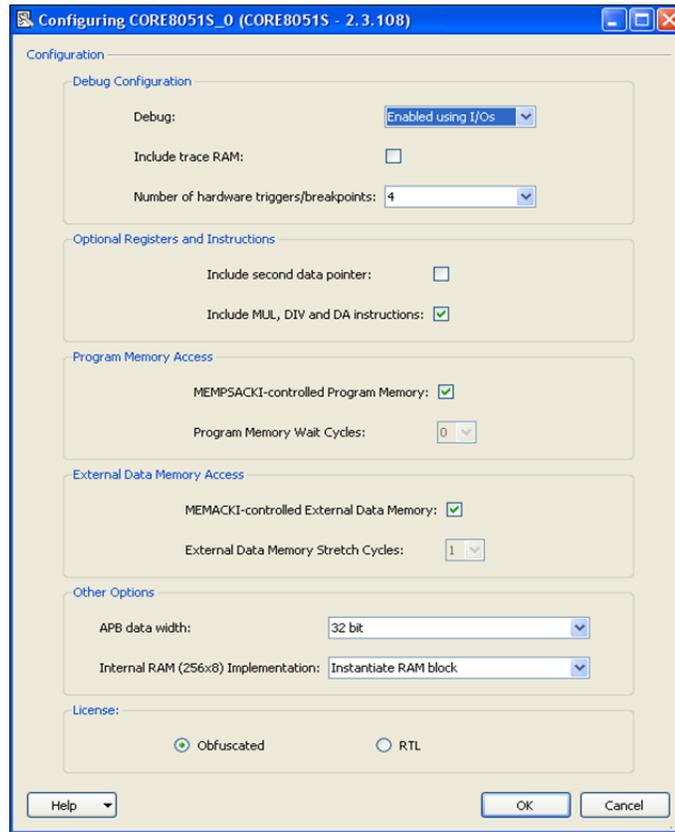


Figure 2 • Core8051s Configuration Settings

Figure 3 on page 5 shows the Core8051s block connected to the program and data memory. Read enable and write enable signal connections to the Core8051s block are shown. The memories are 2K deep and 8 bits wide. The program memory is accessed using program store memory acknowledge input (MEMPSACKI) and data memory is accessed using data memory acknowledge input (MEMACKI). The MUX logic block in Figure 1 on page 3 is used to control the data being passed to Core8051s.

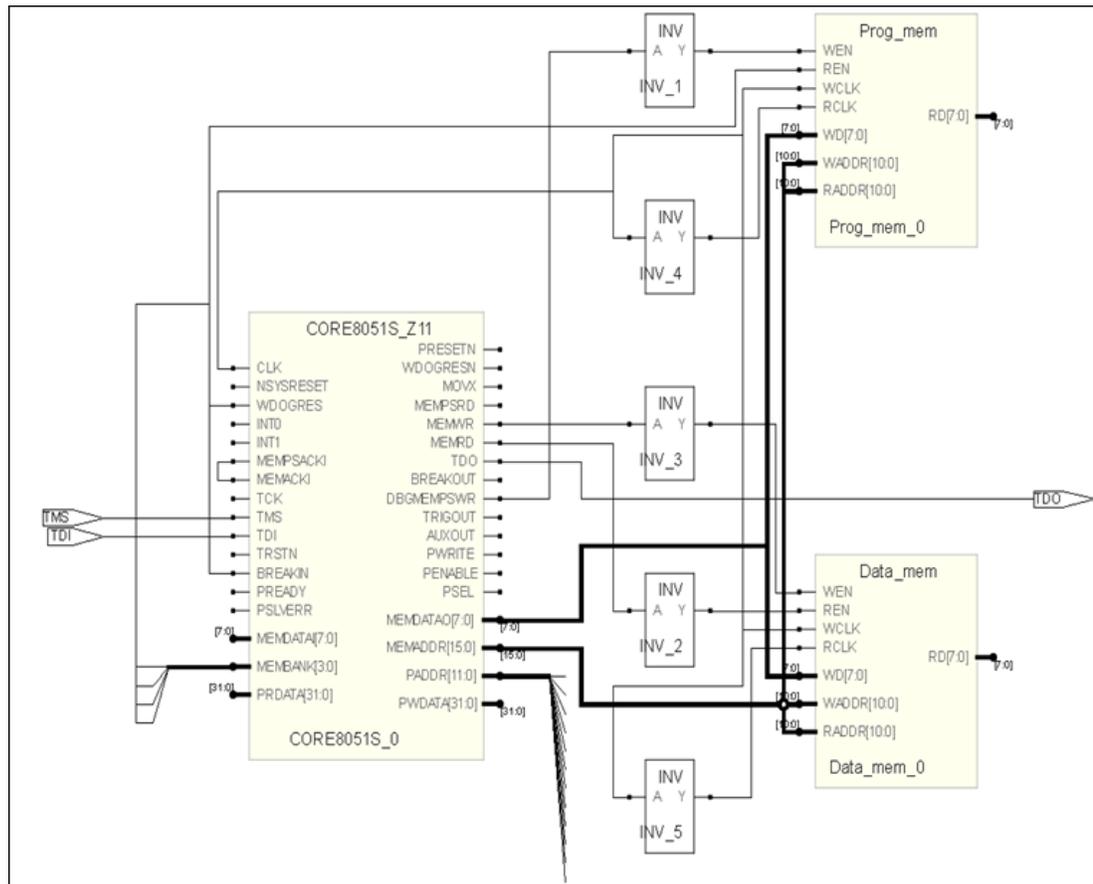


Figure 3 • Core8051s Connection to Program and Data Memory

Core8051s is also connected to CoreGPIO and CoreInterrupt via the APB3 bus. One of the CoreGPIO blocks is used to drive the LEDs. SW3 is connected to INT0 and SW4 is connected to INT1 of CoreInterrupt. When the program is running, pressing SW3 and SW4 causes the illuminated LED to be shifted one position to the left or right.

Design Implementation

The design example is implemented using Libero IDE. Table 1 shows the top-level I/O ports:

Table 1 • Top Level I/O Ports

Name	Width	Description
CLKA	1	40 MHz clock (available in Axcelerator starter Kit). It is used as CLKA for the PLL used to generate a 25 MHz clock for Core8051s block.
LED	8	LED outputs.
RESETN	1	Reset signal. It is connected to the push-button reset input on the starter kit and is logical-ANDed with PLL Lock signal.
SW3	1	Switch input. When the program is running, pressing SW3 rotates the LED to left.
SW4	1	Switch input. When the program is running, pressing SW4 rotates the LED to right.
TCK	1	JTAG test clock, connected to user I/O 132.

Table 1 • Top Level I/O Ports

Name	Width	Description
TDI	1	JTAG test data in, connected to user I/O 138.
TDO	1	JTAG test data out, connected to user I/O 141.
TMS	1	JTAG test mode select, connected to user I/O 133.
TRSTN	1	JTAG test reset, connected to user I/O 140.

Use the following guidelines during design implementation in Libero IDE:

- **Simulation:** To run simulation with application code, the program memory block should be loaded with the hex file generated for the application from Keil. This simulation ensures that the design works as expected with preloaded memory. Note that Axcelerator and RTAX-S devices are one time programmable (OTP), so you should verify the design in simulation before testing it in the hardware.
- **Global assignment:** Make sure to use global for the critical signals such as clocks and resets to avoid any clock skew issues. In this design example, PCLK, TCLK, and TRSTn are promoted to a routed clock network (RCLK) using the CLKINT macro. This is done to allow flexible I/O assignment, which is not possible when CLKBUF is used.
- **Synopsis design constraints (SDC):** Apply the appropriate clock constraints. In this design example, a 40 MHz timing constraint is applied to CLKA. SmartTime automatically creates a 25 MHz generated clock constraint on the PLL output.
- **Pin assignment:** During layout, make sure that the pins are connected properly. This includes CLKA, RESETN, LEDs, and switch connections. Also make sure the five JTAG pins (TCK, TMS, TDI, TDO, and TRSTB) are brought to headers that are easy to access and connect to FlashPro3/FlashPro4 hardware.

Running the Hardware Debug

The design example has been targeted for the *Axcelerator Starter Kit*. The kit includes a 40 MHz oscillator, 8 LEDs, several I/O headers, and a PQ208 socket. The design is implemented in an AX250-PQ208 device and the debug operation can be run using Keil uVision3 IDE or FS2 ISA-Actel51 debugger. Ensure that you have the FS2 ISA-Actel51 Debugger and Keil uVision3 IDE software installed in your PC before running the design example. Refer to the "[Appendix A – Software and Hardware Tools Requirement](#)" section on page 11 for hardware tools required and download link.

The following section explains the hardware debug operation using step by step information. Make sure that the Axcelerator starter kit has the right device and design programmed.

Step 1: Connect the FlashPro3 programmer to the PC with the USB cable.

Step 2: Connect the other end of the FlashPro3 programmer to the JTAG user I/Os in the Axcelerator Starter Kit. Note that the JTAG pins from Core8051s are connected to the user I/Os under header J10 in the Axcelerator Starter Kit.

Make sure to connect GND and VJTAG pins on FlashPro3 to the appropriate signals.

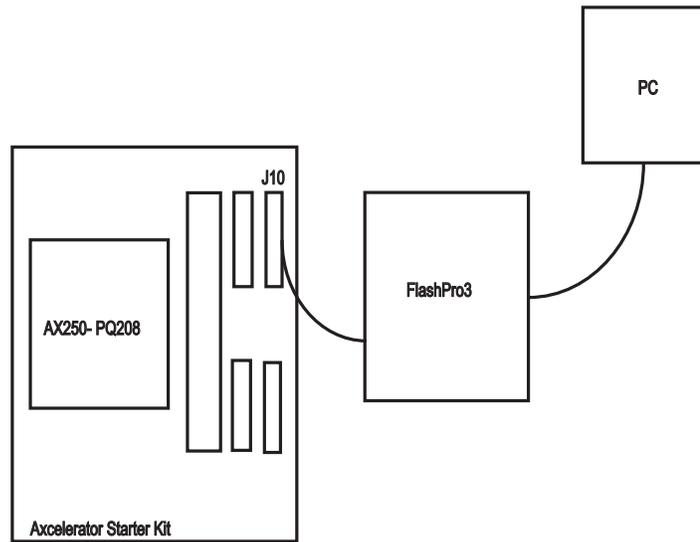


Figure 4 • Connection Between PC and Axcelerator Starter Kit

Step 3: Open the FS2 ISA-Actel51 Debugger console by selecting **Program > FS2 > ISA-Actel51** console to launch the console, as shown in [Figure 5](#).

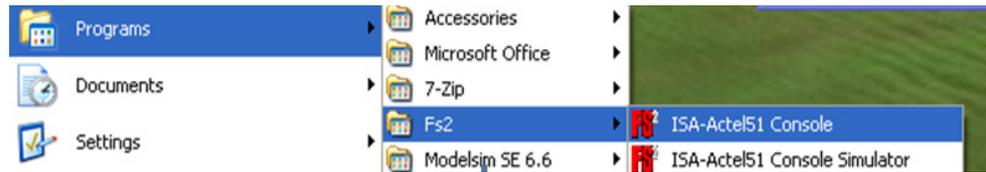
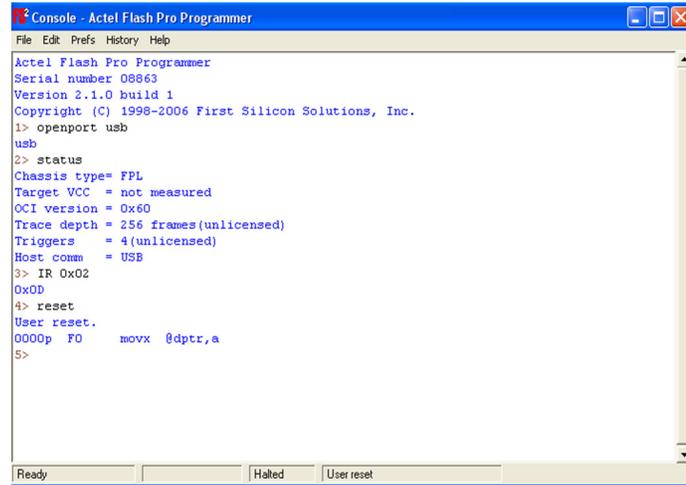


Figure 5 • Launching FS2 ISA-Actel51 Console

Step 4: In the FS2 console, type **openport lpt1** to select USB communication in the System Analyzer software.

Step 5: Set the TCK frequency to 10 MHz using the command **config TckRate 10000000**.

Step 6: In FS2 console, execute the basic commands such as status, IR 0x02, and reset to make sure the FS2 console is able to interact with Core8051s via Flashpro3, as shown in [Figure 6 on page 8](#).



```
Console - Actel Flash Pro Programmer
File Edit Prefs History Help
Actel Flash Pro Programmer
Serial number 08863
Version 2.1.0 build 1
Copyright (C) 1998-2006 First Silicon Solutions, Inc.
1> openport usb
usb
2> status
Chassis type= FFL
Target VCC = not measured
OCI version = 0x60
Trace depth = 256 frames(ungicensed)
Triggers = 4(ungicensed)
Host comm = USB
3> IR 0x02
0x0D
4> reset
User reset.
0000p F0 movx @dptr,a
5>
```

Figure 6 • Basic Commands in ISA-Actel51 Console

- Status: Read the status information. Note that the OCI version is 0x60 and Triggers is 4. These numbers match the Core8051s configuration settings in [Figure 2 on page 4](#).
- IR 0x02: Read the IR register 0x02. Note that 0x0D is the expected value according to Core8051s Handbook.
- Reset: Reset the Core8051S controller.

Step 7: Close FS2 console.

Step 8: Start μ Vision3 by clicking its desktop icon.

Step 9: Select Project > Open Project from the main menu and open the file led_rotate.Uv2.

Step 10: Compile and build the project by clicking the **Rebuild** icon.

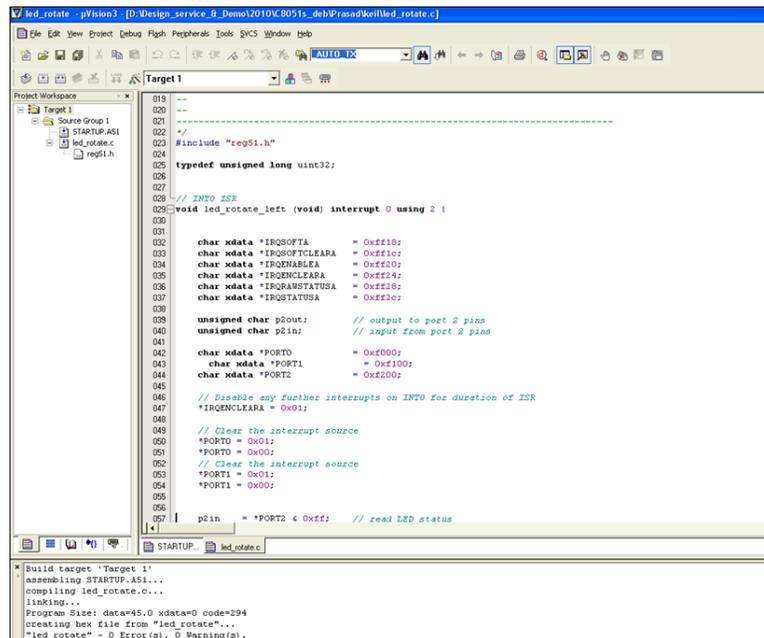


Figure 7 • Building Design in Kiel

- Step 11: Right-click on Target 1 and choose "Options for Target 'Target 1'". The "Option for 'Target 1'" dialog box is displayed.
- Step 12: Click the **Debug** tab and make sure the debug setting is as shown in Figure 8.

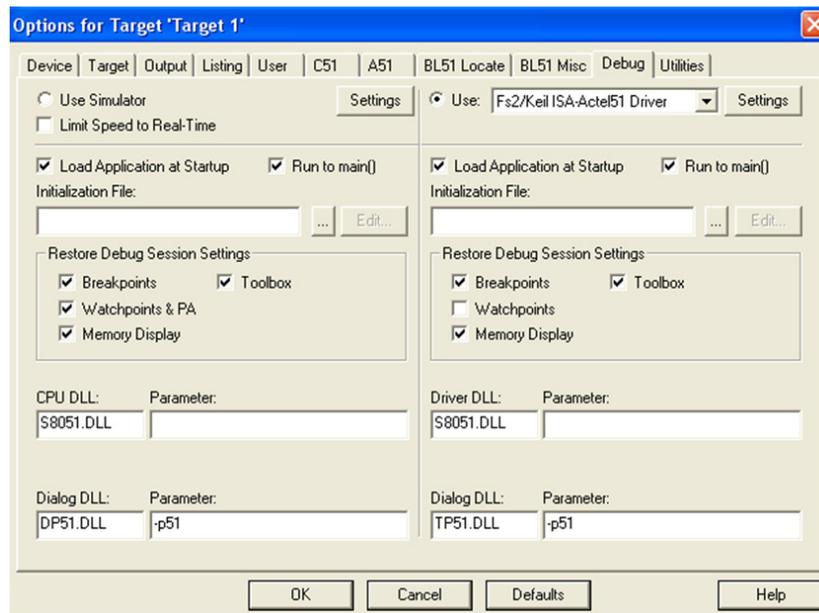


Figure 8 • Debug Settings

Step 13: Click the **Start/Stop debug** icon. The debug window is displayed.

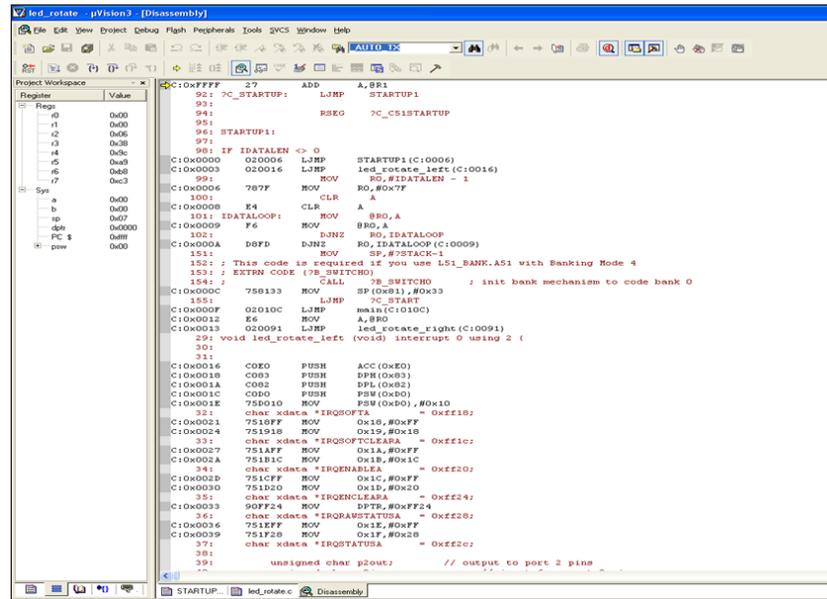


Figure 9 • Debug Session

Step 14: Click **Run** to run the application.

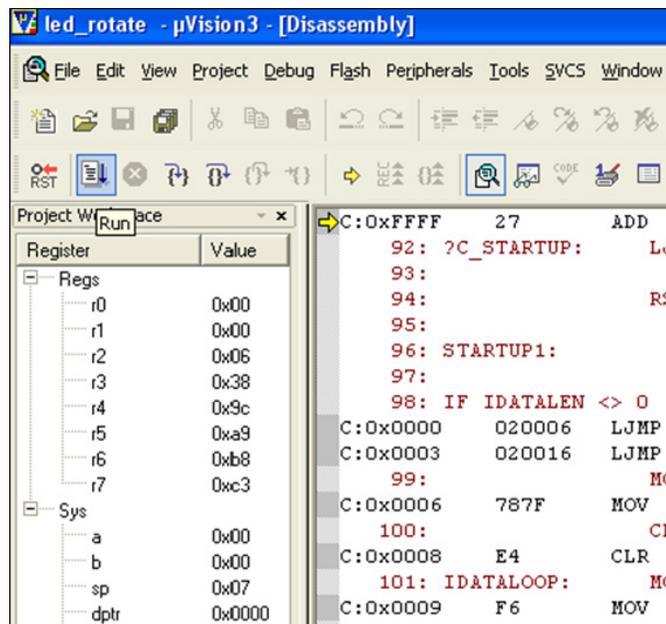


Figure 10 • Run Application in Kiel

Step 15: Press SW3 or SW4 on the Axcelerator Starter Kit and you can observe the LED rotating.

Note: Instead of using Keil, you can load the hex file from the FS2 console and run the program. Refer to the "Appendix C – Running Debugger Using FS2 Debugger" section on page 11 for loading and running the debugger from the FS2 console.

Conclusion

Core8051s is a popular 8-bit microcontroller IP Core. It can be implemented both in antifuse and flash-based FPGA. During the design process, you can run your code in debug mode before creating the final code. In antifuse FPGAs such as Axcelerator or RTAX-S, you can run debug using the user I/Os option only. This application note provides detailed information on running a debug operation on Core8051s using user I/O mode in an AX250-PQ208 device. This same procedure can be followed on other Axcelerator and RTAX-S devices.

Appendix A – Software and Hardware Tools Requirement

The following software and hardware are required to run the design example:

Software required:

- *Libero IDE v9.0 IDE*
- *Keil μ Vision3 IDE (available from Keil)*
- *FS2 ISA-Actel51 Debugger*

Hardware required:

- Axcelerator Starter Kit with AX250-PQ208 device
- Microsemi SoC Products Group FlashPro3 Programmer

Note: The FlashPro4 programmer will not work for this demo design.

Appendix B – FS2 and Keil μ Vision3 Interface

The Core8051 System Analyzer interfaces with Keil μ Vision3 software by adding a hardware driver and a simulator driver to the list of available tools in Keil's tools.ini file. After initiating the μ Vision3 software, these drivers can be selected and used to debug code with the FS2 debugger. All Keil source level debug features work with the Core8051 System Analyzer – go, step, halt, view/edit registers, and view/edit memory.

To start a debug session in Keil:

- Select the Debug menu.
- Select the Start/Stop Debug Session menu item.

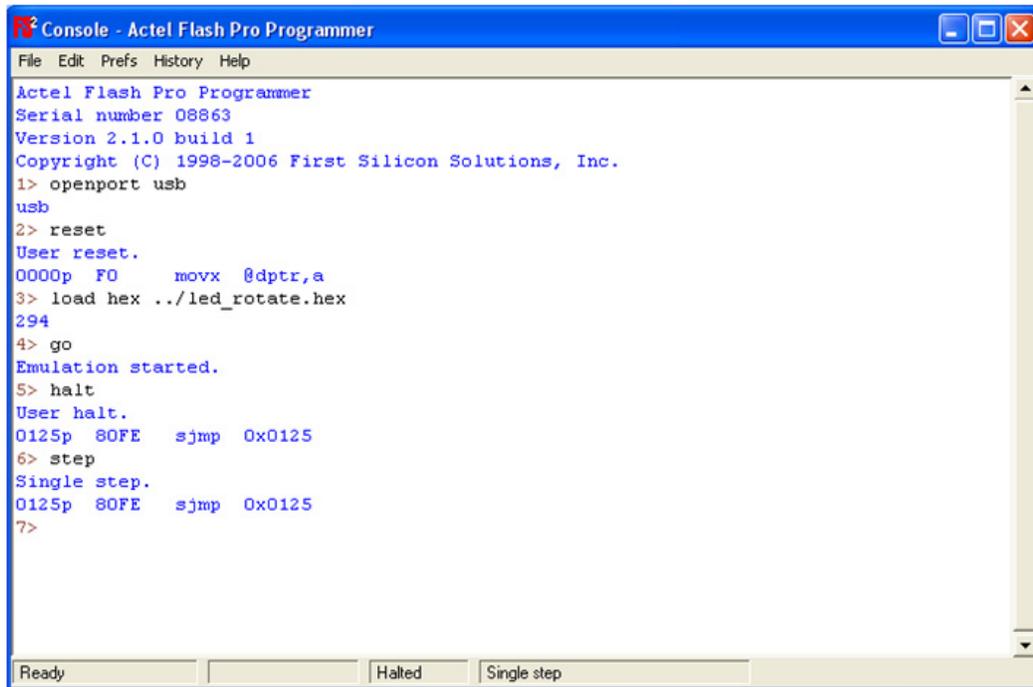
Once a session is started, new menu items appear in the Peripherals menu:

- Target Settings - Used to change communications ports.
- Trace - Used to view the trace for the last emulation session.
- Triggers - Used to set complex triggers and view current settings.
- Console Interface - Gives full access to the FS2 debugger with macros. Use the Help menu for details. Memory and register changes in the Console are reflected in the Keil windows, and vice-versa.

Appendix C – Running Debugger Using FS2 Debugger

You can load the hex file and run the debugger using the FS2 debugger.

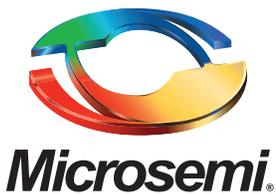
- Locate the hex file and type **Load hex <path_to_hex_file>** to load the hex file.
- Type **go** to start emulation at the current execution address.
- Type **halt** to stop emulation immediately. This displays the next execution address and one instruction disassembly.
- Type **step** to execute 1 instruction and to display next execution address and one instruction disassembly.



```
Actel Flash Pro Programmer
Serial number 08863
Version 2.1.0 build 1
Copyright (C) 1998-2006 First Silicon Solutions, Inc.
1> openport usb
usb
2> reset
User reset.
0000p F0 movx @dptr,a
3> load hex ../led_rotate.hex
294
4> go
Emulation started.
5> halt
User halt.
0125p 80FE sjmp 0x0125
6> step
Single step.
0125p 80FE sjmp 0x0125
7>
```

Ready Halted Single step

Figure 11 • Run Application



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

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

© 2012 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.