

Core8051s Debugging in Axcelerator and RTAX-S Device

Table of Contents

Introduction	1
Core8051s Debug Configuration	1
Design Example	2
Conclusion	10
Appendix A – Software and Hardware Tools Requirement	10
Appendix B – FS2 and Keil µVision3 Interface	10
Appendix C – Running Debugger Using FS2 Debugger	10

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 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 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 via a JTAG connection. In RTAXS or Axcelerator families, you can only run debug operation using user I/O option as these devices do not have UJTAG macro. This application note describes how to perform Core8051s debug operation using the user I/Os option in 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 these debug modes. From the Debug drop-down menu, you can choose the following options:

- Disabled: It excludes debug functionality.
- Enabled using UJTAG: It allows including the debug functionality and to use the dedicated JTAG pins of the device (via the UJTAG macro) for the debug connection.
- Enabled using I/Os: It allows including debug functionality and to use 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 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: It allows including a 256-byte deep trace RAM within Core8051s. Including the trace RAM increases the tile count for the processor and consumes RAM blocks on the device.
- Number of hardware triggers/breakpoints: It allows setting the maximum number of hardware triggers/breakpoints available 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. Please note that the UJTAG macro is only available for Microsemi Soc Products Group flash-based FPGA families. So 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 RTAX-S or Axcelerator device, you are restricted to “Enabled using I/Os” option for running 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 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 and Keil project. The design example displays a rotating LED pattern using push-button switch. It 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 3](#) shows Core8051s configuration setting. Please note that “Enable user I/Os” option is selected for debug.

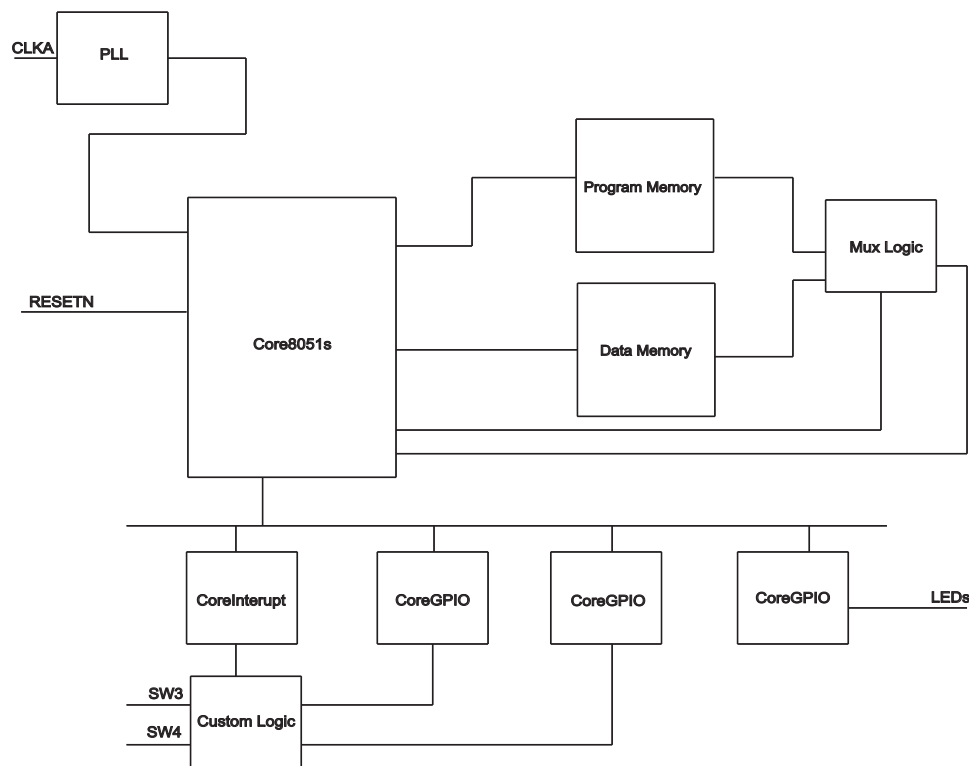


Figure 1 • Design Example Block Diagram

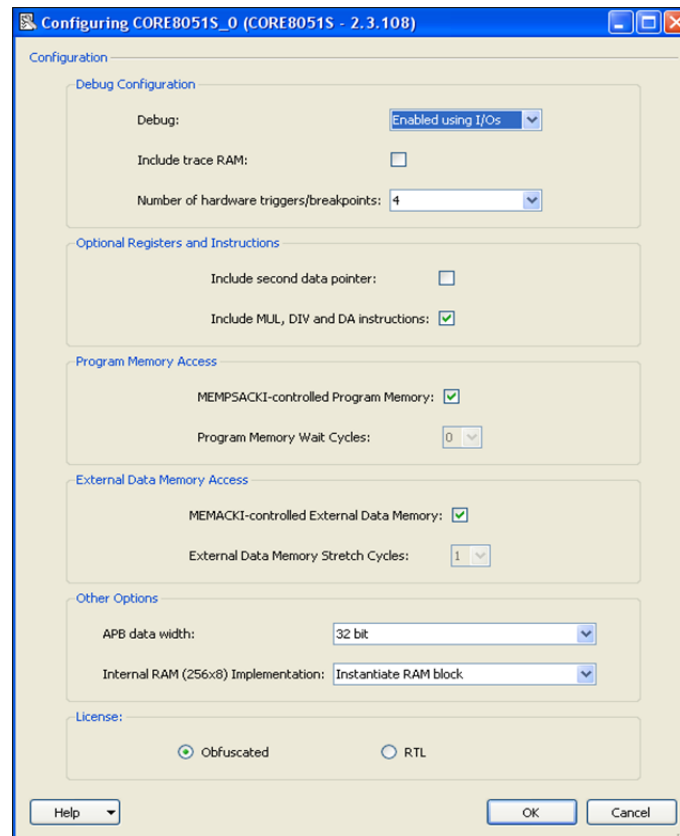


Figure 2 • Core8051s Configuration Settings

Figure 3 on page 4 shows Core8051s block connected to the program and data memory. It shows read enable and write enable signal polarity and how they are connected to Core8051s block. The memories are 2K deep and 8-bit 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 2 is used to control the correct data being passed to Core8051s.

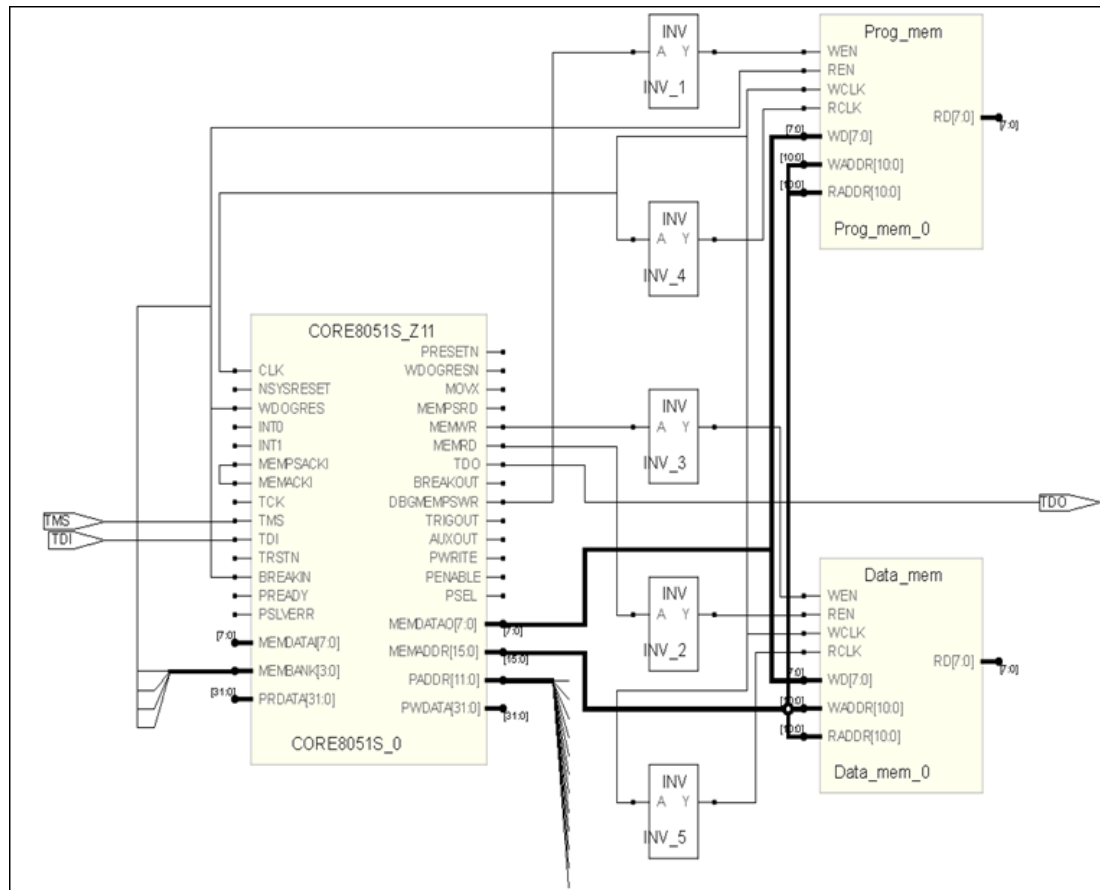


Figure 3 • Core8051s Connection to Program and Data Memory

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 AX starter Kit). It is used as CLKA for PLL used to generate 25 MHz clock for Core8051s block.
LED	8	LED outputs.
RESETN	1	Reset signal. It is connected to 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:

- **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 or RTAX-S devices are one time programmable (OTP), so you should run 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 routed clock network (RCLK) using 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 5 JTAG pins (TCK, TMS, TDI, TDO, and TRSTB) are brought to header 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 has a 40 MHz oscillator, 8 LEDs, several I/O headers, and a PQ208 socket. The design is implemented in AX250-PQ208 device and the debug operation can be run using Keil uVision3 IDE or FS2 ISA-Actel51 debugger. Make sure that you have 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 10 for hardware tools requirement 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 a USB cable between FlashPro3 and PC.

Step 2: Connect the other end of FlashPro3 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 also on FlashPro3 to the appropriate signals.

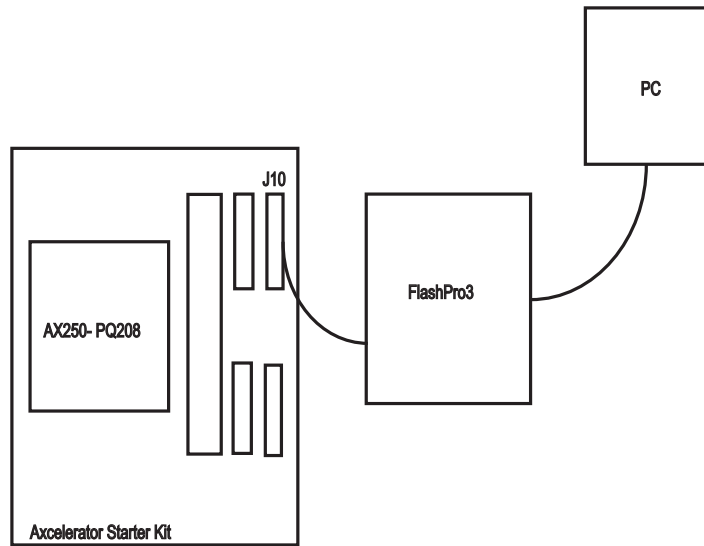


Figure 4 • Connection Between PC and AX Starter Kit

Step 3: Open the FS2 ISA-Actel51 Debugger console by selecting **Program > FS2 > ISA-Actel51** console to launch console as shown below.

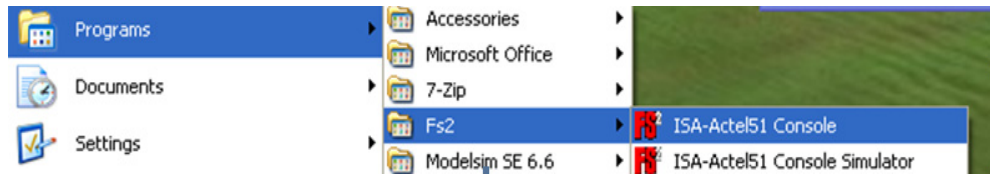
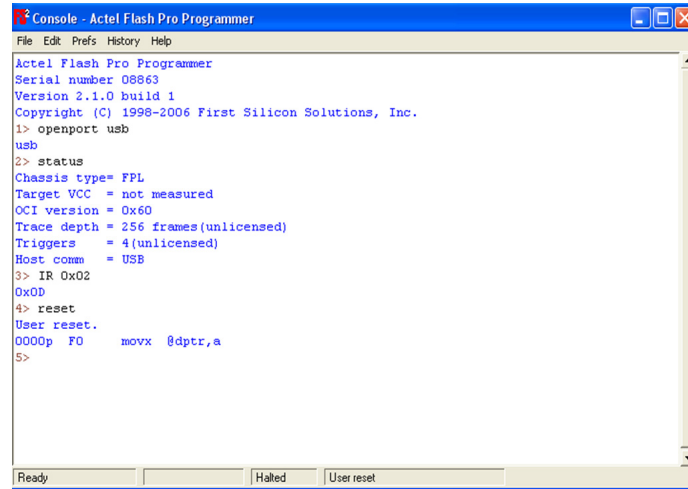


Figure 5 • Launching FS2 ISA-Actel51 Console

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

Step 5: Set the TCK frequency to 10Mhz using the command 'config TckRate 10000000'.

Step 6: In FS2 console, execute the basic commands like “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 7.



```
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(unlicensed)
Triggers = 4(unlicensed)
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 info. Note that the OCI version is 0x60 and Triggers is 4. These numbers match Core8051s configuration settings in Figure 2 on page 3.
- 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 on 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 on the Rebuild icon.

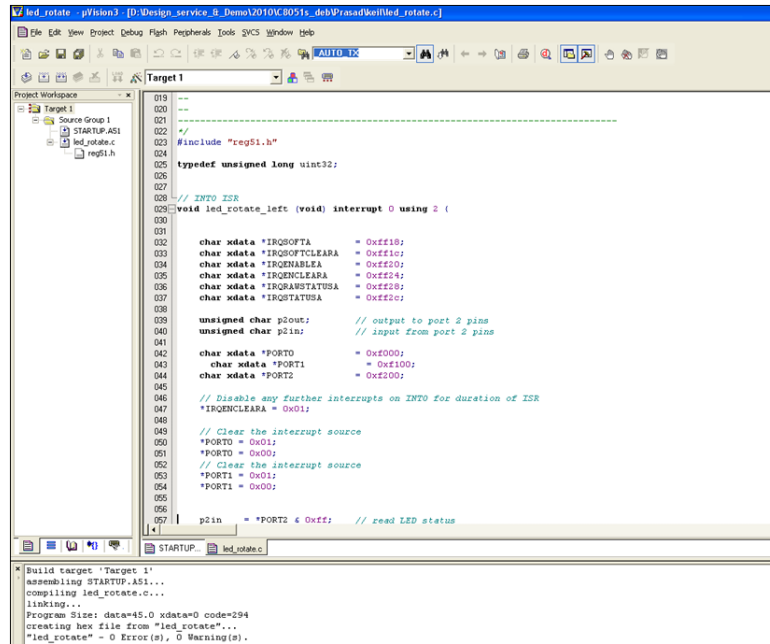


Figure 7 • Building Design in Kiel

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 on the **Debug** tab and make sure the debug setting is as shown in Figure 8.

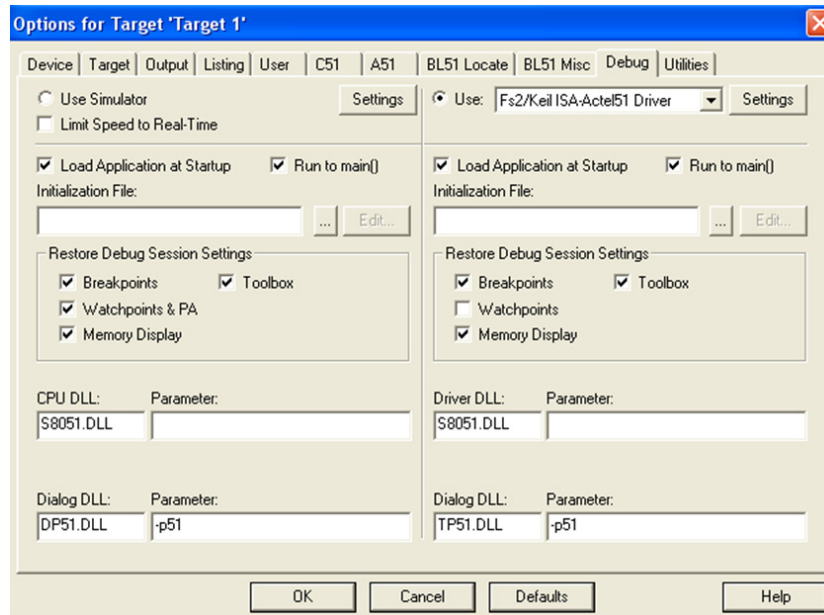


Figure 8 • Debug Setting in Kiel

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

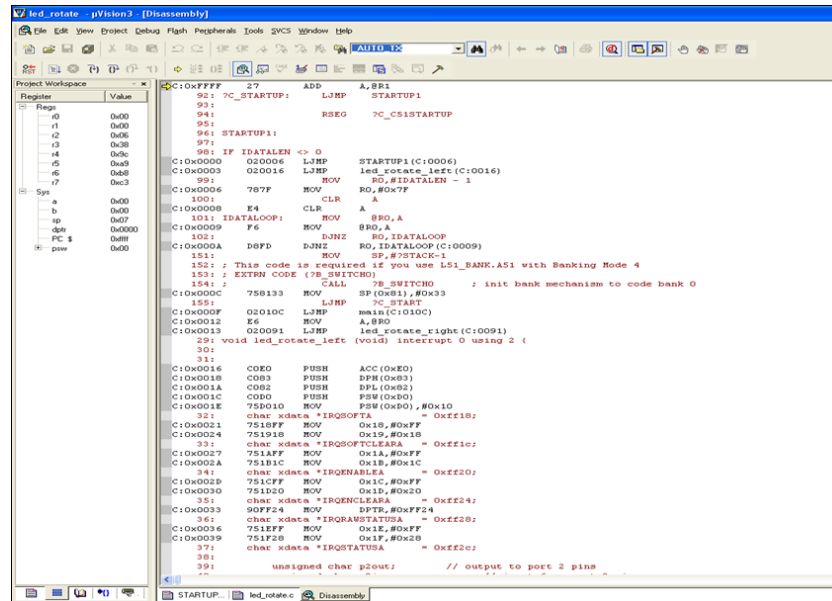


Figure 9 • Debug Session in Kiel

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

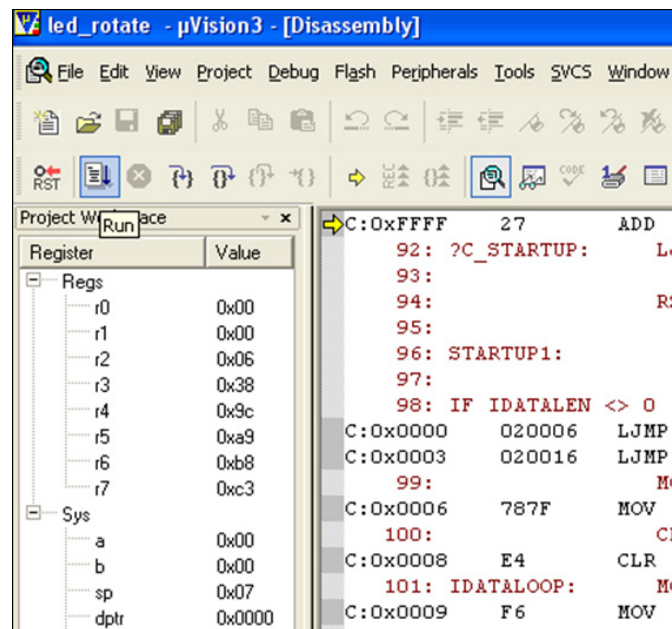


Figure 10 • Run Application in Kiel

Step 15: Press SW3 or SW4 on AX starter kit and you can observe the LED rotating.

Note: Instead of using Keil, you can also load the hex file from FS2 console and run the program. Refer to the "Appendix C – Running Debugger Using FS2 Debugger" section on page 10 for loading and running debugger from FS2 console.

Conclusion

Core8051s is a popular 8-bit microcontroller IP Core. It can be implemented both in Antifuse and Flash based FPGA. During design process, you can run your code in debug mode before creating the final code. In Antifuse FPGA's like Axcelerator or RTAX-S, you can run the debug using the user I/Os only. This application note provides detailed information on running debug operation on Core8051s using this mode in AX250-PQ208 device. This same procedure can be followed on other Axcelerator or 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 uVision3 IDE (available from Keil)*
- *FS2 ISA-Actel51 Debugger*

Hardware required:

- AX 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.

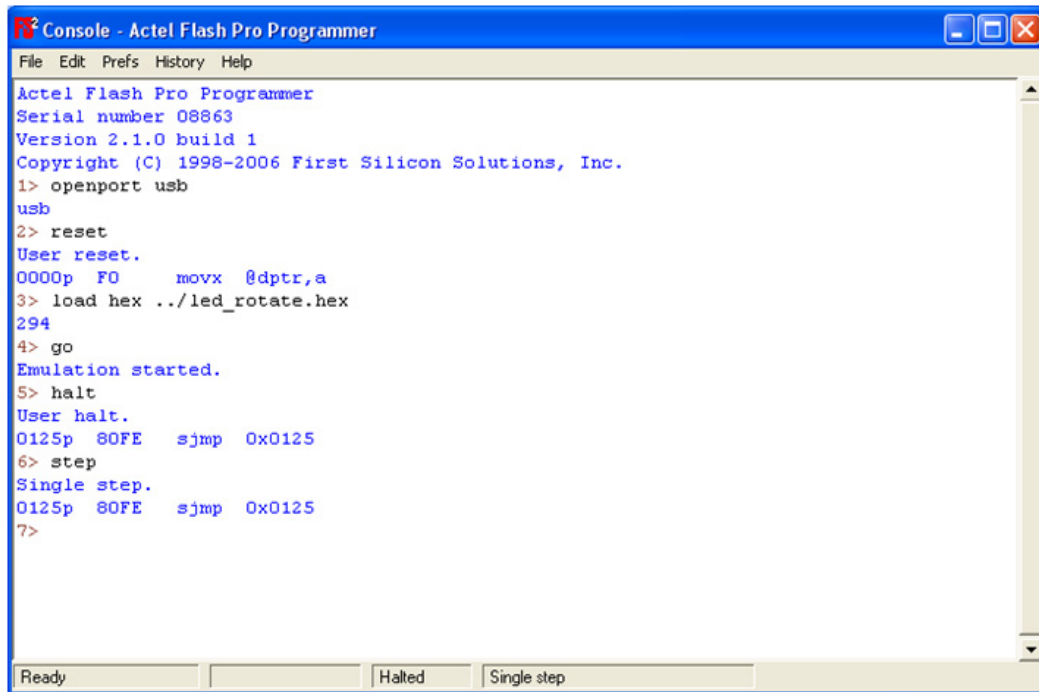
Once a session is started, new menu items appear in the Peripherals menu:

- 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.



```
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> 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 in Kiel



Microsemi Corporate Headquarters
One Enterprise Drive, Aliso Viejo CA 92656
Within the USA: (800) 713-4113
Outside the USA: (949) 221-7100
Fax: (949) 756-0308 · www.microsemi.com

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.

© 2011 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.