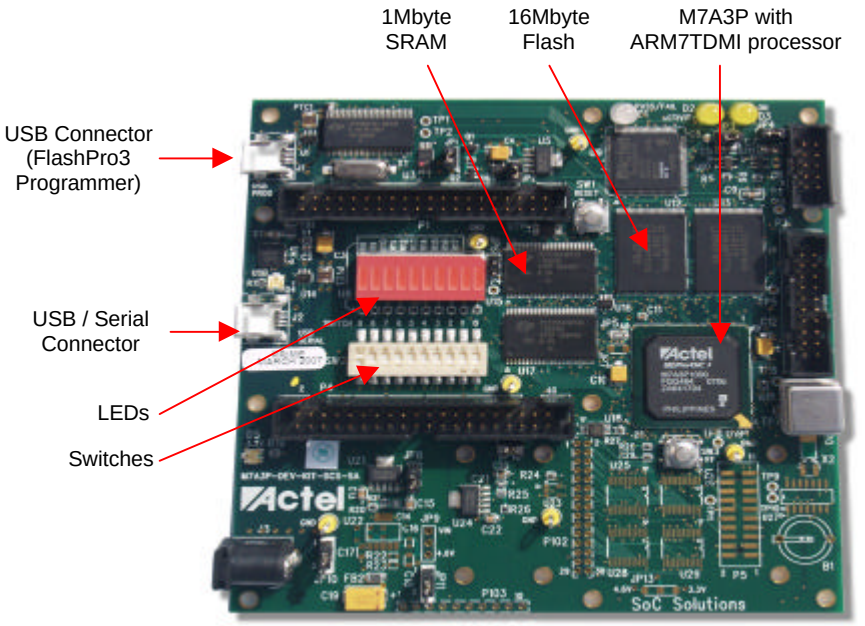
Version 1.0

Introduction

Thank you for purchasing the Actel ARM7-Enabled ProASIC3™ Development Kit.

This guide provides the information required to easily design ARM7 systems using ProASIC3 devices.



M7A3P Development Board

Table of Contents

Document Contents

Chapter 1 – Contents and System Requirements

describes the contents of the ProASIC3™ Development Kit.

Chapter 2 – Hardware Components

describes the components of the ProASIC3™ Development Board.

Chapter 3 – Setup and Self Test

describes how to set up the M7A3P Development Board and how to perform a self test. Also describes how to set up and use the SwiftTrax™ Integrated Monitor.

Chapter 4 – Description of Pre-loaded Design

describes the preloaded design on the ProASIC3 Development Board.

Chapter 5 – Sample Design Tutorial

gives a tutorial on using the included demo design.

Appendix A – FG484 Package Connections for M7A3P FPGAs

provides a table listing the board connections.

Appendix B – Board Schematics

provides illustrations of the ProASIC3 Development Board.

Appendix C – Product Support

describes Actel support services.

Document Assumptions

This user's guide assumes the following:

- You intend to use Actel Libero® Integrated Design Environment (IDE) software.
- You intend to use Actel CoreConsole® and SoftConsole® software.
- You have installed and are familiar with:
    - Actel Libero IDE v8.0 or later software.
    - Actel CoreConsole® v1.3 or later software.
    - Actel SoftConsole® v1.1 or later software.
- You are familiar with PCs and the Windows® operating system.

Contents and System Requirements

This chapter details the contents of the ARM7-Enabled ProASIC3 Development Kit and lists the power supply and software system requirements.

ARM7-Enabled ProASIC3 Development Kit Contents

The ARM7-Enabled ProASIC3 Development Kit includes the following:

- M7A3P Development Board
- Libero IDE v8.0 CD (in DVD case)
- Core Console v1.2.1 / Soft Console 4.1 (in DVD case)
- Install CD with sample design
- 2 USB A to Mini-B Cables
- Quick Start Guide

System Requirements

The ARM7-Enabled ProASIC3 Development Kit requires the following:

- PC or Laptop running Windows XP or Windows 2000
- 2 USB ports (connectors) on the PC or Laptop

Hardware Components

This chapter describes the hardware components of the ARM7-Enabled ProASIC3 Development Kit.

ARM7-Enabled ProASIC3 Development Kit

The ARM7-Enabled ProASIC3 Development Kit consists of the following:

- 2 USB cables
- M7A3P Development Board
  - Actel A3P1000 ProASIC3
  - 1 MByte SRAM
  - 16 MByte Flash
  - USB–RS232 converter chip
  - GPIO connectors
  - Low Power (Can be powered with USB)
  - On-board FlashPro3 circuitry
  - 20-Pin ARM7 JTAG connector
  - Socketed Crystal Oscillator
  - Pushbutton power-on reset circuit
  - 10 test LEDs
  - 10 test switches
  - Expansion connectors

For further information, refer to the following appendices:

Appendix A – “Package Connections”

Appendix B – “Board Schematics”

Ideal Uses for the development kit

Ideal uses for the ARM7-Enabled ProASIC3 Development Kit are the following:

- development and verification of embedded microprocessor based systems or subsystems
- product development platform
- algorithm development

Applications

The ARM7-Enabled ProASIC3 Development Kit is ideal for use in the following applications:

- smart controllers
- Digital Signal Processing
  - CPU or DSP (inside FPGA)
- wireless baseband processors
- communications
- display controllers
- sensor controllers

Detailed Board Description and Usage

The ARM7-Enabled ProASIC3 Development Kit has various advanced features that are covered in later sections of this chapter. The architecture provides access to a one-chip FPGA solution containing an ARM7TDMI 32bit RISC processor, and digital peripheral components.

Note that the Actel FPGA is soldered directly to the board. The development board is available only in a directly soldered configuration. A socketed configuration is not available.

Full schematics are available on the Install CD supplied with the development kit. See Appendix B for M7A3P Development Board schematics contents.

Block Diagram

The following simplified block diagram shows the main features of the M7A3P Development Board. The blocks in dashed lines are not normally installed.

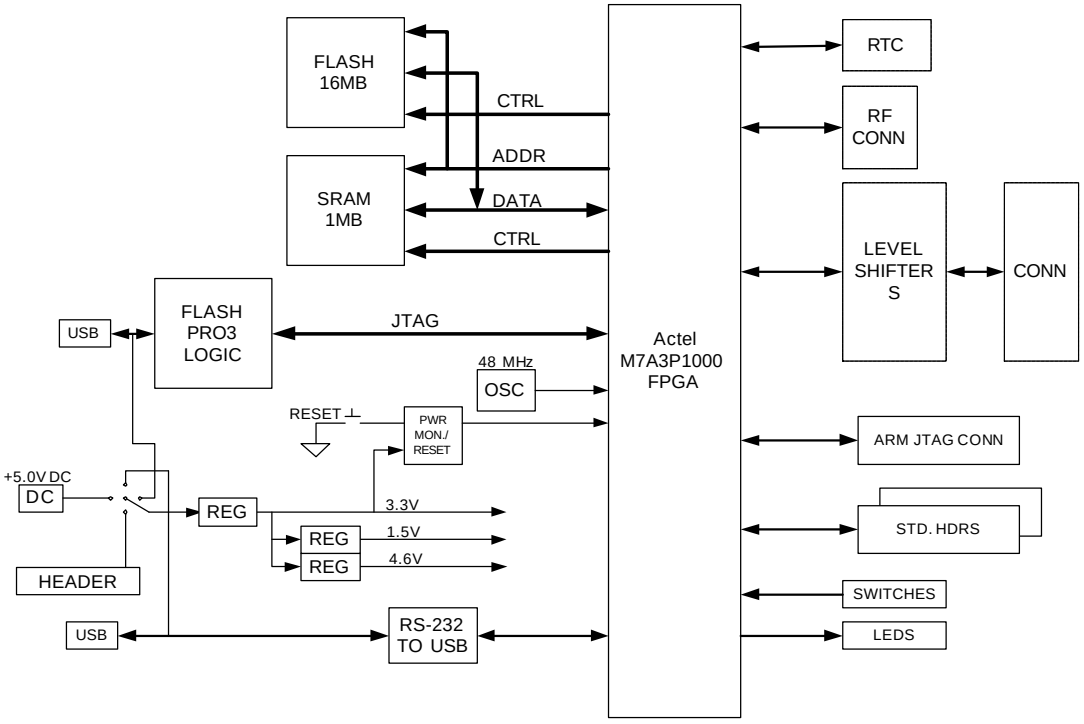


Figure 2.1 - M7A3P Development Board Block Diagram

Power

No power supply is provided with the development kit. An external 5 V power supply can be used but the preferred method to power the board is via the USB (FlashPro3) connector. A USB “mini-B to A” cable is provided to allow this. A second cable is also provided for the USB-to-RS232 interface. Both USB ports can be used simultaneously.

To program the ProASIC3 FPGA, the USB cable should be plugged into the top, or J1 connector and jumper JP10 should be installed in the 1-5 position. In this configuration the circuitry along the top of the board emulates the Actel FlashPro3 programming adaptor. The FlashPro software is then used to program the device.

There are two USB power rails on the board. Please refer to Figure 2-1. The FlashPro3 logic uses 3 voltage regulators. A 3.3V regulator provides power to the USB interface. The USB interface can then enable 2.5 V and 3.3 V regulators for the APA150 FPGA which implements the JTAG programming logic for the ProASIC3 FPGA.

During programming, JP10 is used to connect the 2 USB power rails. Q1, a P-channel MOSFET device, is used to hold off powering up most of the board until the USB interface has had a chance to ask for an increase from 100 mA to 500 mA of 5V current.

Aside from the regulators for the FlashPro3 circuit, there are two regulator components on the board to provide 1.5 V and 3.3 V to the ProASIC3 FPGA. There is an optional regulator that can supply another voltage for applications such as GPS.

The board can be powered by an external +5.0 V 2.1 mm positive-center power supply via connector J3, by the USB-to-serial interface via J2, or from a motherboard via P103. Any one of these inputs can replace the USB connector as the power source using jumper JP10.

Note that with JP10 installed in the 1-5 position, it is possible to use both USB connections at the same time. Also note that the J1 USB is designed to power up the entire board, whereas the J2 USB is not. Only very low power designs (below 100 mA) should use only J2 to power the board.

In programming mode, the "ON" LED (D3) illuminates to indicate that an external supply has been connected to the board. LED D5 indicates that the 3.3V regulator is operating.

The 3.3 V supply is used to provide the VPUMP programming voltage.

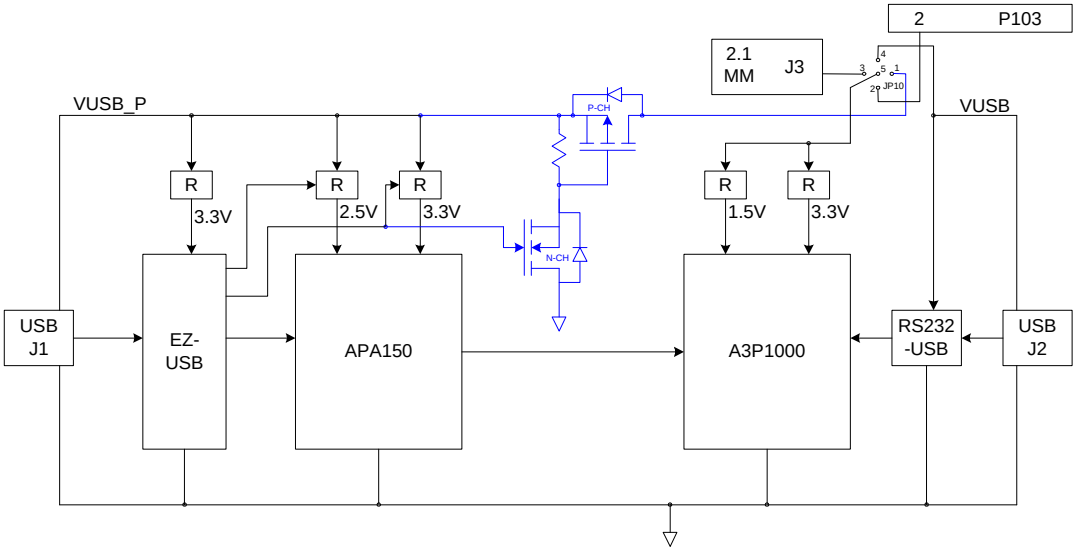


Figure 2.2- M7A3P Development Board Power Configuration

Accessory Card Power Supply Connections

Limited power may be supplied by the ProASIC3 Development Kit to an accessory card. Connectors for accessory cards (headers P1 & P4) are shown on page 2 of the schematics. The 5 V input voltage and the 3.3 V regulated voltage are provided to the accessory card connectors.

Note that it may be necessary to use J1 or J3 to power the board if your configuration uses more than 100 mA of 5 V current. If you need more than 500 mA, you must NOT power the main board logic from USB through J10. Both USB ports may still be used. It is also possible to use no USB after the M7A3P1000 is programmed. Note that the FlashPro3 logic is only powered when USB power is applied through J1. Similarly, the serial USB interface is only powered when USB power is applied through J2.

Current (5V USB)	Options for Powering Main Logic	J10 Jumper Setting	Comments
0 to 100 mA	J1	1-5	FlashPro3 provides power.
	J2	4-5	Serial USB provides power.
	J3	3-5	2.1mm barrel connector
	P103	2-5	Optional header
101 to 500 mA	J1	1-5	J2 USB port can still be used.
	J2	4-5	J1 USB port can still be used.
	P103	2-5	Optional header
> 500 mA	J3	3-5	J1 & J2 USB ports can still be used.
	P103	2-5	Optional header

Table 2.1- M7A3P Development Board Power Options

ARM7TDMI Support

A standard 20-pin ARM7TDMI JTAG header (P3) is supplied to enable use of any ARM standard emulator and debug environment. Alternatively USB port J1 and the on-board FlashPro3 circuitry may be used as a debug path to the ARM7TDMI's internal debug interface. Note that the FPGA design must instantiate the CoreMP7Sd component to include the debug logic for software development.

Connected to the FPGA for use by the ARM7TDMI are Flash and SRAM memories. The SRAM configuration is a byte-addressable 256Kx32, or 1MB. The Flash configuration is 4Mx32 or 16 MB (expandable to 32 MB). The Flash is not byte-addressable but each 16-bit word may be individually addressed. All memories are asynchronous. Access times may vary due to component availability so check the datasheets for the memories installed.

The clock for the FPGA logic can come from U20, a Socketed 48 MHz 3.3 V, 50% duty cycle crystal oscillator. This frequency may be modified inside the FPGA.

Two power-on resets are provided to the FPGA. One has a pushbutton feature to provide a warm reset. This pushbutton reset is also used to reset the optional Real Time Clock chip. The non-push-button reset is usually used for the ARM7TDMI's JTAG nTRST (DBGnTRST) signal. Other reset schemes may also be used.

USB Serial Interface

USB Connector J2 not only provides a way to power up the board, but also provides a USB-to-RS-232 interface through U4. In this configuration, the ProASIC3 FPGA should contain a UART core. This USB interface may be used in conjunction with the J1 USB interface for programming the ProASIC3 FPGA, ARM7 debugging, etc.

Other Features

The development kit also contains a 10 position DIP switch bank and a 10 LED module for general purpose use. All signals are connected to the FPGA. The LEDs and switches are active high and the switches have 10K pull-down resistors.

Programming or Re-Programming the Pre-Loaded Design

On the ARM7-Enabled ProASIC3 Development Kit Install CD, you will find a *Pre-Loaded Design* folder containing a STAPL file for programming the target design. Select the *.STP* file from the CD and use that as the STAPL file in the FlashPro software. Selecting **PROGRAM** will erase, program, and verify the part. The total programming time is approximately 2 minutes 30 seconds.

M7A3P Development Board Jumper Descriptions

Jumper	Development Kit Function	Notes
JP1		This jumper was removed and is no longer part of the design.
JP2	Provides 3.3V to Prog. USB interface	Current can be measured at this point.
JP3	Provides 2.5V to FlashPro3 FPGA	Current can be measured at this point.
JP4	Provides 3.3V to FlashPro3 FPGA	Current can be measured at this point.
JP5	Connects On-Board 3.3V supply to VPUMP	Zero ohm 1206 resistor
JP6	Connects pushbutton reset to system reset input of ARM7 JTAG connector	Not installed; this feature is usually not necessary
JP7	Connects 1.5V to VPLL for A3PE FPGAs	Not installed
JP8	Provides 3.3V to ProASIC3 FPGA	Current can be measured at this point.
JP9	Allows optional 4.66V supply to be bypassed with Vin (5V)	Not installed. Used for GPS application.
JP10	Selects input power to the main board logic from one of 4 sources	Factory installed between pins 1 & 5 to select USB power from FlashPro3 USB interface.
JP11		This jumper was removed and is no longer part of the design.
JP12	Provides 1.5V to ProASIC3 FPGA	Current can be measured at this point.
JP13	Selects level shifter voltage as either 4.66V (pin 1) or 3.3V (pin 3)	Zero ohm 1206 resistor is not installed. Level shifters used for GPS application.

Table 2.2- M7A3P Development Board Jumper Descriptions

Test Points

All ground test points on the board are fitted with small test loops. They are labeled only as “GND”. Signal test points are labeled on the silkscreen as TP1, TP2, etc. The test points have holes that a scope probe can access. Power voltages may be probed at the jumpers that connect them to the circuitry that they power. Voltages will be 5.0 V, 3.3 V, 2.5 V, 1.5 V, or GND. When measuring the voltage at a test point with a DVM (digital voltage multimeter) the ground lead should be connected to a test point labeled GND and the voltage lead should be connected to the voltage to be tested. All voltage labels on the board are relative to a 0 V ground reference or GND.

Physical Characteristics of Board

The printed circuit board assembly, including all components, is completely lead-free RoHS compliant.

The board is fabricated with six copper layers. The layers are arranged as follows from top to bottom:

- Layer 1 – Top Signal Layer
- Layer 2 – Signal Layer 2
- Layer 3 – Ground Plane
- Layer 4 – Power Plane
- Layer 5 – Signal Layer 3
- Layer 6 – Bottom Signal Layer

Optional GPS Application

The ARM7-Enabled ProASIC3 Development Kit was designed to support both embedded and PC-based GPS applications. For more information please contact info@socsolutions.com for further information.

3

Setup and Self Test

This chapter outlines how to set up and test the ARM7-Enabled ProASIC3 Development Kit.

Software Installation

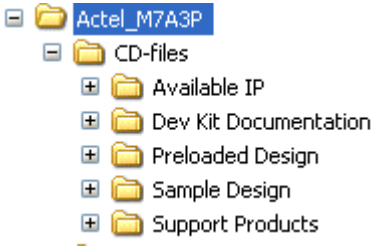
The ARM7-Enabled ProASIC3 Development Kit includes three CDs:

- Install CD v1.0
- Libero IDE version 8.0 CD
- CoreConsole v1.3 / SoftConsole v1.1 CD.

Installing ARM7-Enabled Development Kit using the "Install CD v1.0".

Place the "Install CD" in the CD Drive on your PC (PC refers to either your Personal Computer or Laptop). The CD should automatically start an auto-run session. Follow the instructions (prompts) on the "Install" dialog box.

The "Install" application will properly place all the documentation and sample project files in the C:\Actel_M7A3P folder (default) or the user selected folder.



The "Install" application will also set up the USB drivers on your PC for the USB-RS232 chip.

Note: The Libero IDE CD Install procedure will set up the USB drivers for the on-board FlashPro3 Programmer circuitry.

The Install CD contains the following documention:

- This ARM7-Enabled Development Kit User Guide.
- ARM7-Enabled Development Kit Quick Start Guide.
- Available IP from Actel and also 3rd Party IP Vendors.
- Support Products such as SwifTrax embedded monitor.
- M7A3P Board Schematics.

Installing Libero IDE v8.0

Place the Libero CD in the CD Drive on your Personal Computer or Laptop. The CD should automatically start an auto-run session. At this point, follow the instructions (prompts) on the "Libero IDE" dialog box.

For more Libero IDE v8.0 software installation instructions, please refer to the documentation supplied in the Libero IDE DVD case or refer to the *Actel Libero IDE / Designer Installation and Licensing Guide for Software v8.0*.

Installing CoreConsole v1.3 and SoftConsole v1.1

Place the CoreConsole v1.3 – SoftConsole 1.1 CD in the CD Drive on your Personal Computer or Laptop. The CD should automatically start an auto-run session. At this point, follow the instructions (prompts) on the dialog box.

For futher information on how to use CoreConsole and SoftConsole tools, please refer to the documentation supplied in the Libero IDE DVD case.

Note: Libero IDE , CoreConsole and SoftConsole tools will be used in for the Sample Design in Chapter 5 – Sample Design Tutorial.

Testing the M7A3P Development Board

Powering up the board

Before powering up the M7A3P Development board for the first time, please make sure the switches and jumpers are in the following, factory set, positions:

SW2: All switches (0-9) are in the ON position.

JP2, **JP3**, **JP4**, **JP8**, and **JP12** are installed.

JP10 connection from pin 1 to pin 5 (star)

All others are not installed.

To power up the board, connect one end of a supplied USB cable to a USB port (connector) on your PC or Laptop. Connect the other end to M7A3P port J1. After a few seconds, you should see the big yellow "ON" LED at the top right of the board illuminate.

The PC will detect that new hardware is installed. The "add new hardware wizard" will take care of the Actel FlashPro3 driver installation. The wizard will ask for a location for the drivers. See the Actel FlashPro3 documentation for the location of these drivers. These are usually located in the "<FlashPro install location>\Drivers".

Now connect one end of the second supplied USB cable to a second USB port (connector) on your PC or Laptop. Connect the other end to M7A3P Development Board port J2. You should see the LED closest to the J2 connector illuminate.

The PC will detect that new hardware is installed. The "add new hardware wizard" will take care of the USB driver installation. The wizard will ask for a location for the drivers. The drivers are located at ".\CD-Files\Preloaded Design\Software\USB_Drivers".

Step by Step guide for running "User Tests"

The M7A3P Development Board is shipped with a preloaded embedded controller design from *SoC Solutions*. See Chapter 4 – Description of Pre-loaded Design for design architecture details.

*Note: If the embedded controller design needs to be re-loaded, use the Actel FlashPro3 programming tool to program the FPGA with the preloaded embedded controller design (**socTop.stp** in the **Preloaded Design\ Hardware** folder). However, reloading the design should not be necessary.*

1. Verify the PC COM port that was enumerated with the RS-232 USB chip connected to J2.
    a. On the PC, click the Start button and open up the "Device Manager" by navigating to "Control Panel -> System". Then click the "Hardware" tab, then the "Device Manager" button as shown in figure 3.1.

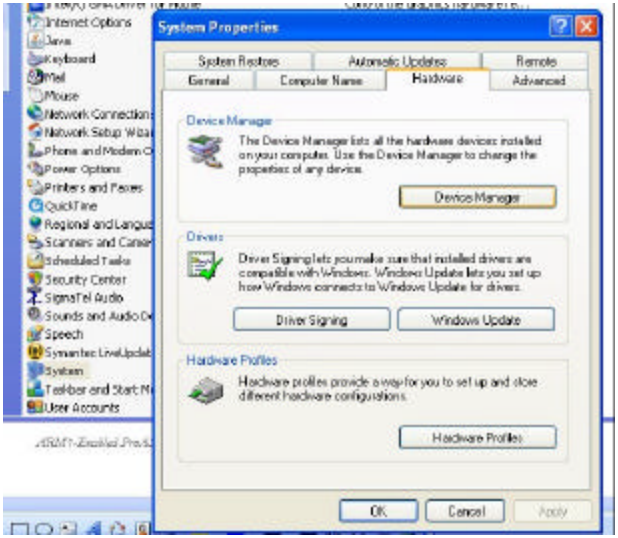


Figure 3.1- Finding the COM Port

b. Expand the "Ports (COM & LPT)" tree in the view.

Figure 3.2- Finding the COM Port

c. Note the COM port associated with the "**SFE USB to RS232 Controller**". This information will be needed later for running the M7A3P Development Board tests using the "**M7A3P Board – User Tests**" program (GUI).

2. Run the **M7A3P Board – User Tests** by navigating to the "Preloaded Design\Software\M7A3P_User_Tests" folder and double clicking the M7a3p_test.exe file.

3. The first dialog box will be a COM configuration dialog. Select the COM port from step 3, and click OK.

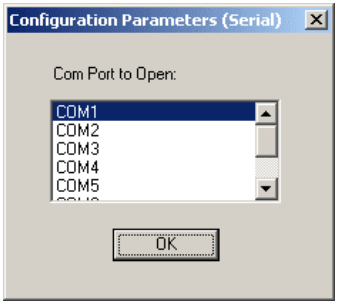


Figure 3.3- COM Port Dialog

4. Now run all the tests in the **M7A3P Board – User Tests** GUI.

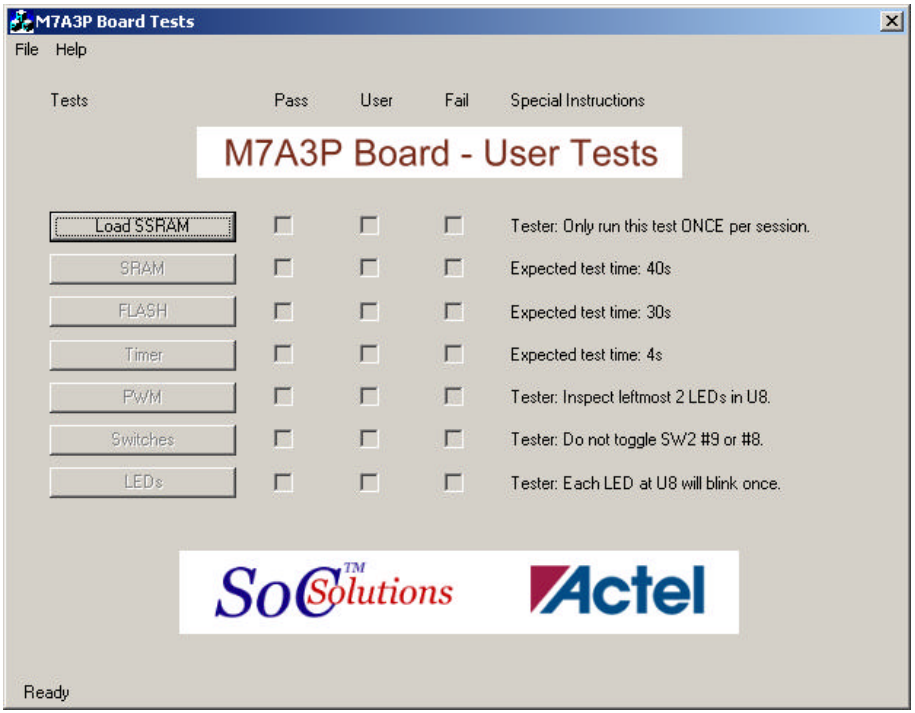


Figure 3.3- M7A3P Board – User Tests GUI

a. Run the **Load SSRAM** test first by clicking the Load SSRAM button. This will load the embedded test software into the internal SSRAM of the preloaded embedded controller design on the FPGA via the serial port.

b. If the **Load SSRAM** test passes, all other tests will become available and can be run in any order.

- Clicking the **"SRAM"** button performs a complete check of the external SRAM chips at U15 and U17.
- Clicking the **"FLASH"** button performs a continuity check of the external FLASH chips at U12 and U13.
- Clicking the **"Timer"** button tests the interrupt capability of the ARM7 processor core in the FPGA.
- Clicking the **"PWM"** button tests internal PWM logic, and the top two LEDs at U8.
- Clicking the **"Switches"** button reads the state of the switch bank SW2 then outputs the result in a text format. IMPORTANT: Do not toggle SW2 #9 or SW2 #8 while running the board checkout program. (#9 and #8 are indicated on the board silkscreen.) Doing so will cause the board checkout system to function incorrectly.
- Clicking the **"LEDs"** button will blink each of the 10 LEDs at U8 on and off individually.

c. If the **"Load SSRAM"** test fails, then do the following sequence:

- Close down application.
- Power down the board by unplugging USB cables at J1, J2.
- Verify the jumpers and switches correspond to factory defaults.
- Repeat steps 1-5 above.

d. If "**Load SSRAM"** test still fails, contact *SoC Solutions* at support@socsolutions.com

.

Using the SwifTrax™ Integrated Monitor

The **SwifTrax™ Integrated Monitor**, supplied by *SoC Solutions*, is a software module that can be integrated into any microprocessor based embedded system design containing a UART.

The monitor allows the user to view and control the FPGA's internal registers and memories connected to the system bus without having to use software development tools and JTAG debuggers. In fact any datum that can be accessed from the system's memory map (or address space) can be monitored.

SwifTrax™ Integrated Monitor features:

- Write internal memory or registers.
- Write with read back verify of memory or registers
- Fill memory with data patterns
- Read single or multiple data from any address location
- Selectable addressing and offsets
- Log data from transactions
- Run batch program

Step by Step guide for running the SwifTrax™ Integrated Monitor

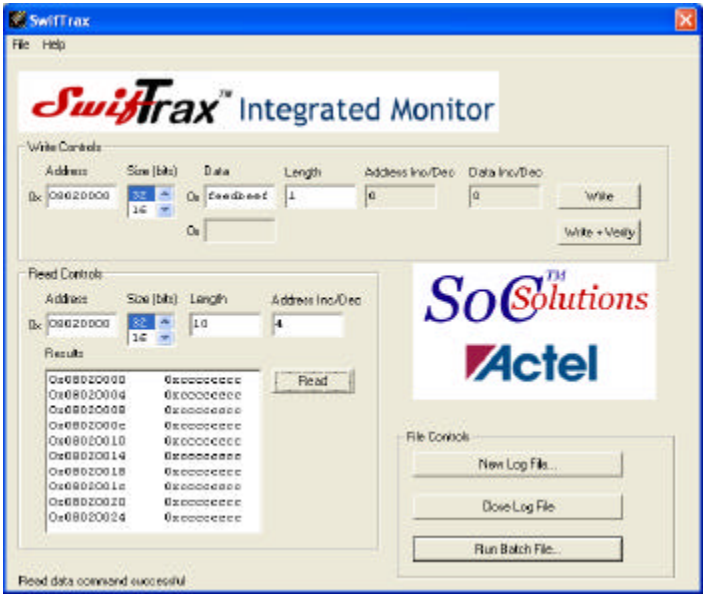


Figure 3.4- SwifTrax Integrated Monitor GUI

1. Configuring the M7A3P Development Board for the **SwifTrax™ Integrated Monitor**
    a. Power down the board by unplugging the USB connectors at J1 and J2.
    b. Configure SW2 as follows: #8 (as indicated on silkscreen) OFF, all others ON.
    c. Power up the board by plugging in the USB connectors at J1 and J2.
2. Run the **SwifTrax™ Integrated Monitor** by navigating to the "Preloaded Design\Software\SwifTrax" folder and double clicking the SwifTrax-Ser.exe file.
3. The first dialog box will be a com configuration dialog. Select the COM port found from step 3 in the "Step by Step guide for running "User Tests"" above, and then click OK.

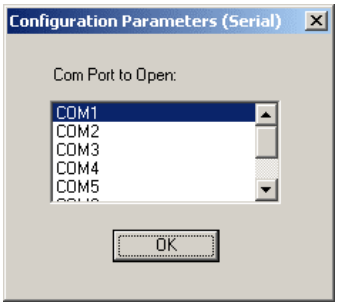


Figure 3.5- COM Port Dialog

4. The **SwifTrax™ Integrated Monitor**, shown in Figure 3.3, should now appear.
5. Now run some of the monitor's functions.
    a. Note that reads and writes can be done through the GUI, or through a scripting interface.
    b. To run an example **SwifTrax™** script, click the "**Run Batch File**" button. This brings up a standard windows file dialog box.

Figure 3.6- SwifTrax Batch File Selection

Select "**leds.txt**" from the "Preloaded Design\Software\SwifTrax" folder and click the open button. Note the bank of LEDs at U8 reflect the counting pattern that is programmed in the batch file.

c. Next, use the GUI to read the first 10 32-bit values at address 0. Notice the ARM7TDMI opcodes in the Results pane.

See *SoC Solutions'* PiP-EC02 datasheet, supplied with the Install CD, for information about the register map of the embedded design used in this example.

Also refer to the "**memory regions.txt**" file for warnings about writing address locations that will cause the PiP-EC02 to stop functioning.

4

Description of the Pre-loaded Design

This is a general description of the embedded controller design, **PiP-EC02,** that is pre-loaded in the M7A3P Development Board. The **PiP-EC02,** Pre-integrated IP platform, is provided by *SoC Solutions*. The design can be licensed directly from *SoC Solutions* or CAST, Inc. For more information, contact sales@socsolutions.com .

PiP-EC02 System Overview

Pre-integrated IP Embedded Controller with AMBA Bus System

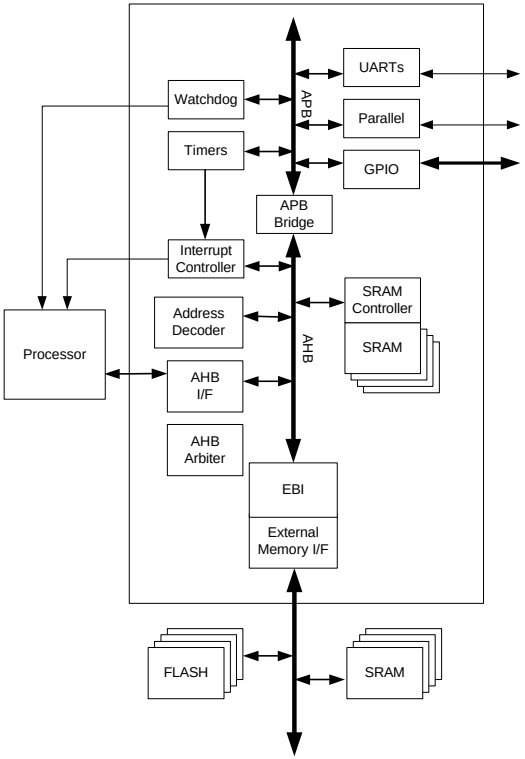


Figure 4.1- PiP-EC02 Embedded Controller Block Diagram

The **PiP-EC02** embedded controller design, as shown in Figure 4.1, is an integrated ARM7TDMI™ based IP Platform. The **PiP-EC02** provides the basic infrastructure for most of the popular SoC applications such as smart controllers, wireless baseband processors, communications devices, display controllers, sensor controllers, GPS, as well as other electronic devices.

The architecture of the **PiP-EC02** is based on the popular AMBA AHB and APB bus system. The PiP-EC02 is a multi-master AHB system that supports architectures with multiple processors or co-processors, DMA channels, burst mode peripherals or bus masters. As shown in Figure 4.1, the AMBA bus is partitioned into two sections:

1. A high speed AHB bus used for fast memories, DMAs and peripherals.
2. A low speed APB bus used for slower peripherals such as UARTs, GPIO, PWMs, etc.

Peripheral Overview

Peripheral Name	Verilog Module Name	Bus	Description
AHB Arbiter	socAhbArb	AHB	AHB Arbiter for up to three bus masters
Address Decoder	socAddrDecRemap	AHB	Generates block select signals for each system block and provides an address remap utility.
Interrupt Controller	socIntrCtrl	AHB	Monitors all system interrupts and issues interrupt requests to the processor
Internal Memory Interface	socIntMemIf	AHB	Interface to zero wait-state internal synchronous SRAM
External Bus Interface (EBI)	socEbi	AHB	Provides a configurable interface to external devices such as FLASH and RAM
APB Bridge	socApbBridge	AHB	Bridge from AHB to APB
Timer	socTimer	APB	Time base generator for the system and general-purpose counter. Two used in PiP-EC02.
UART	socUart	APB	Provides a means of asynchronous serial communication with external devices. Two used in PiP-EC02.
Parallel Port	socParPort	APB	PC compatible parallel port
GPIO	socGpio	APB	Configurable general purpose parallel I/O module
PWM	SocPwm	APB	Generates a Pulse-Width-Modulated output

Table 4.1- PiP-EC02 Block descriptions

For more information regarding the pre-loaded embedded controller design, see the **PiP-ECO2** datasheet provided in the “Available IP\SoC Solutions” folder or contact *SoC Solutions* at info@socsolutions.com.

5

Sample Design Tutorial

This sample design is created specifically for the Actel ARM-Enabled ProASIC3 Development Kit. This tutorial will guide you through designing, synthesizing, loading and testing using the CoreConsole and SoftConsole tools.

Block Diagram

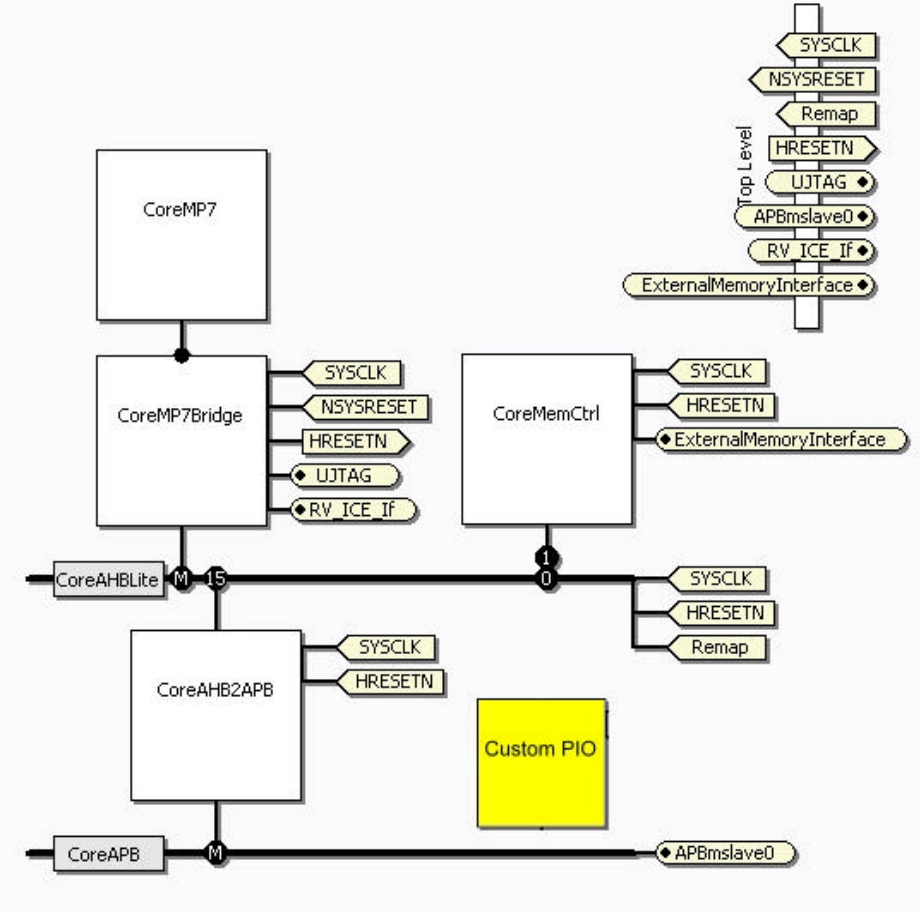


Figure 5.1- Sample Design Block Diagram

Functional Description

The following blocks and functionality are exercised in the sample design:

- CoreMP7
- CoreMP7 to AHB Lite Bridge
- AHB to APB Bridge
- Internal SRAM (Memory) Controller on AHB bus
- Parallel IO (PIO) on the APB bus

This sample design will demonstrate the CoreMP7 system executing software to blink LEDs on U8. While this design is rather simple, it will demonstrate the basics of creating an ARM7TDMI (CoreMP7) based system FPGA design using CoreConsole and the embedded software to run the on the hardware using SoftConsole.

The sample design will execute software instructions from external SRAM via the CoreMemCtrl block, AHB Lite bus and the CoreMP7Bridge block. The external SRAM will contain the software program, data variables, and software stacks.

The CoreMP7 software program will periodically write data to the Custom PIO via the CoreAHB2APB block to the APB bus. The periodic data transactions will blink the LEDs which are tied directly to the Custom PIO. While the program is running, Switches SW2 #0 thru #7 can be toggled to change the state of the LEDs. Two LEDs are providing a “heartbeat” to let the user know that the program is running.

Hardware Implementation

For this sample design project there are three components that comprise the design.

- CoreConsole IP – the CoreMP7 and subsystem
- PLL – used for the clock source.
- Parallel IO (Custom Verilog Source)

Step 1 – View the CoreMP7 Subsystem using CoreConsole

For simplicity, the CoreMP7 Subsystem has been created and can be loaded into CoreConsole to be edited.

To open the CoreMP7 Subsystem in CoreConsole for editing, do the following sequence:

- Navigate to the “Sample Designs\Harware\CoreConsole\ConsoleDesigns” folder.
- Copy the “Example” folder and its entire contents to “<CoreConsole Install Folder>\ConsoleDesigns” folder.
    - This creates a new “<CoreConsole Install Folder>\ConsoleDesigns\Example” folder.
- Launch CoreConsole tool.
    - Click “Open” from the “File” memu.
    - On the Open Design Dialog box, select “Example”

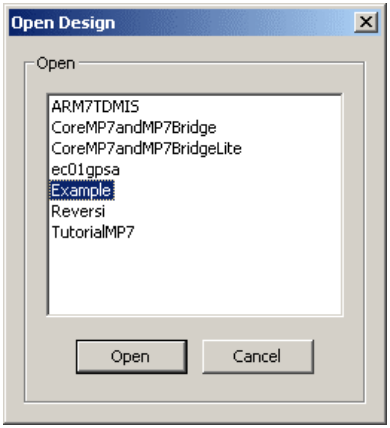


Figure 5.2- CoreConsole Open Design Dialog

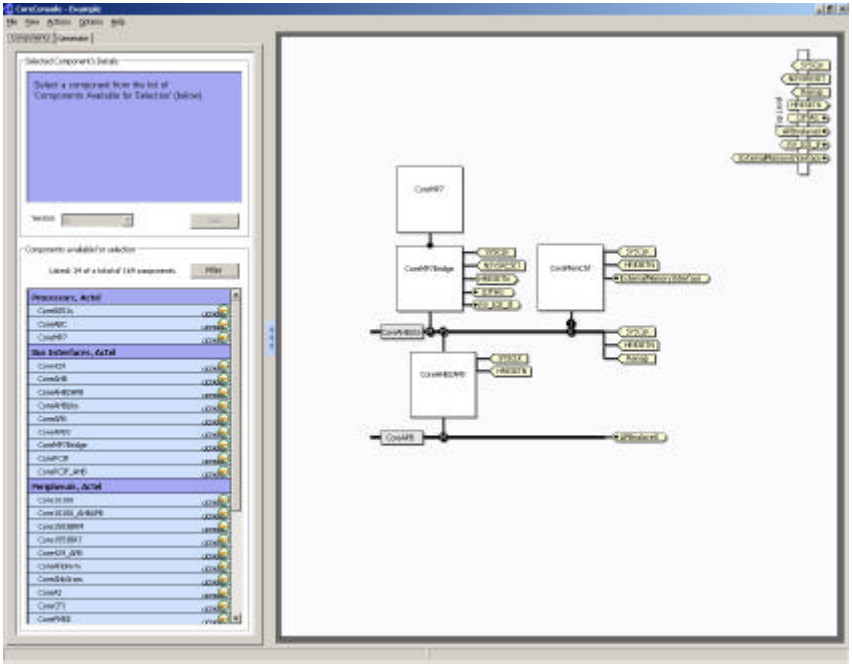


Figure 5.3- CoreConsole with CoreMP7 Subsystem view.

CoreConsole produces a Verilog HDL module of the entire CoreMP7 Subsystem, except the CoreMP7, which is an encrypted netlist. The output files from running CoreConsole are saved in the "..\LiberoExport" folder.

For more information on creating, configuring and editing CoreConsole projects, please refer to CoreConeole documentation supplied on the CoreConsole v1.2.1 CD.

Step 2 – Generate the PLL using SmartGen

For simplicity, the PLL has been created using the SmartGen tool. SmartGen produces a synthesizable Verilog HDL file that can be instantiated in the top level Verilog design.

The parameters for the PLL used in the sample design are:

- 48Mhz input
- 16Mhz output
- Hardwired IO

For more information on creating SmartGen components, please refer to SmartGen documentation supplied on the Libero IDE v8.0 CD.

Step 3 – Create Top Level Design

After generating the major functional blocks, you need to create the top level design. This top level Verilog design can be built using any text editor.

The socTop.v design is supplied in the "Sample Designs\Hardware\Source" folder. This design instantiates the PLL, the CoreMP7 Subsystem and the Custom PIO block.

```
////////////////////////////////
//      CoreMP7 SAMPLE DESIGN
////////////////////////////////
module socTop (
    pbRstN,
    poRstN,

...

// I/O definitions:

    // Resets:
    input  pbRstN;  // external reset connected to pushbutton
    input  poRstN;  // external reset not connected to pushbutton

...

// Instantiations:

    // Global resources for clock, reset
    actel_pll_48MHz_16MHz pll1 (
        .POWERDOWN(poRstNi),
        .CLKA(sysClk_48MHz),
        .LOCK(pllLock),
        .GLA(sysClk_16MHz)
    );

    // Instantiate ARM7 Subsystem:
    Example example0
    (
        // Inputs
        .APBmslave0_PRDATA      (APBmslave0_PRDATA),
        .NSYSRESET              (pbRstNi),
...

    // Instantiate Parallel Port (APB)
    socParPort socParPort0
    (
        .PRESETn    (HRESETN),
        .PCLK       (sysClk_16MHz),
```

You may close the Core Console application when you are finished viewing it.

Step 4 – Use Libero IDE for Synthesis and Place-and-Route

Libero IDE is an "Integrated Development Environment" that helps the user build an FPGA design. The IDE provides a framework in which the design parameters, constraints, source code, etc can be specified using the Libero GUI. The IDE also provides a portal to both the Synthesis and Place-and-Route tools.

Launch Libero IDE (a.k.a. Project Manager).

Run the New Project Wizard by selecting File->New Project… from the menu. The Wizard will guide you through the steps in setting up a Libero project.

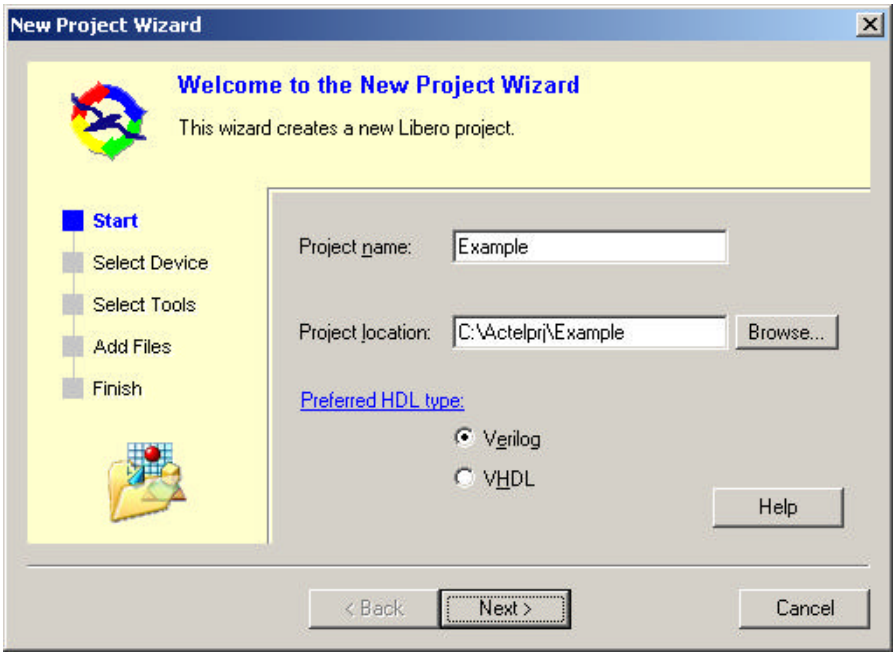


Figure 5.4- Libero IDE New Project Wizard - Start

In the Project Name text box, type **Example**.

Select the **Verilog** radio button under the **Preferred HDL** type.

Click **Next** button.

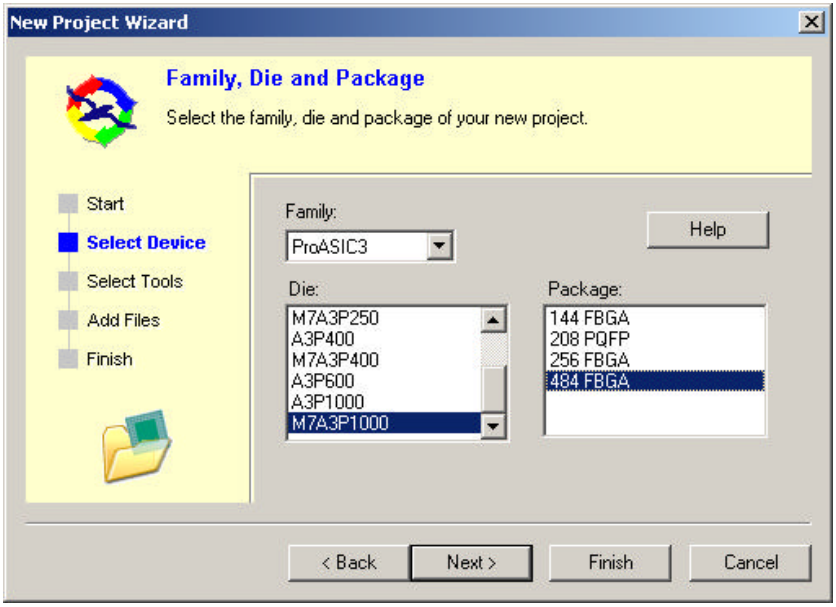


Figure 5.5- Libero IDE New Project Wizard – Select Device

Select **ProASIC3** from the **Family** dropdown list.

Select **M7A3P1000** from the **Die** select box.

Select **484 FPGA** from the **Package** select box.

Click **Next** button.

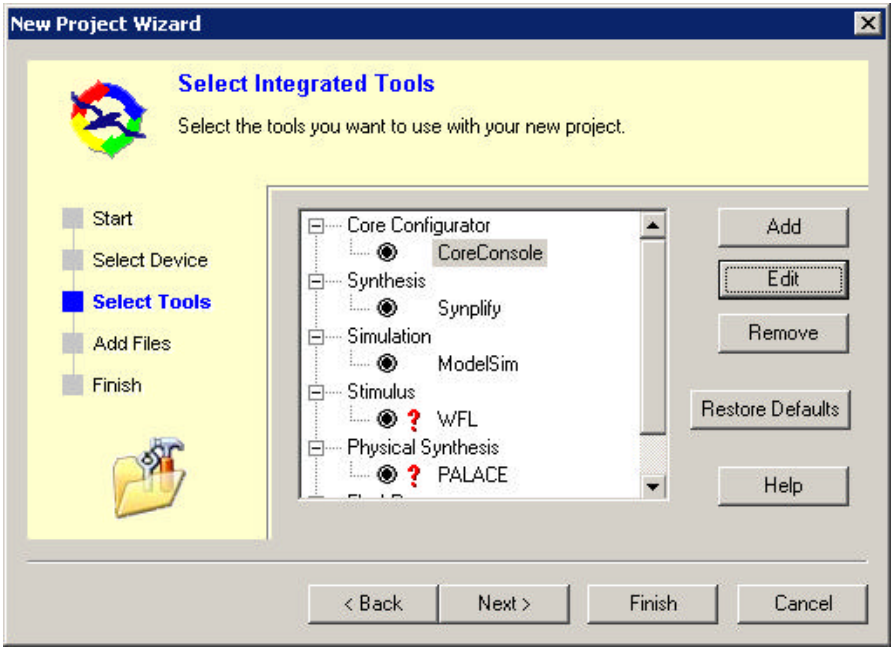


Figure 5.6- Libero IDE New Project Wizard – Select Tools

Select your preferred tools from the tools tree radio buttons. You may need to select Edit to specify the locations of your tools.

Click **Next** button.

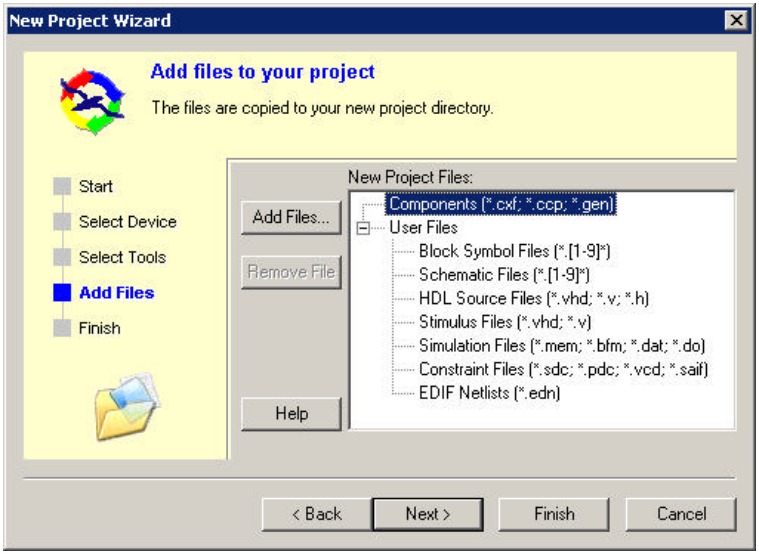


Figure 5.7- Libero IDE New Project Wizard – Add Files

Now add all the necessary files for the Sample Design by clicking the **Add Files** button. The Windows Explorer window will open so you can navigate to the file to be added.

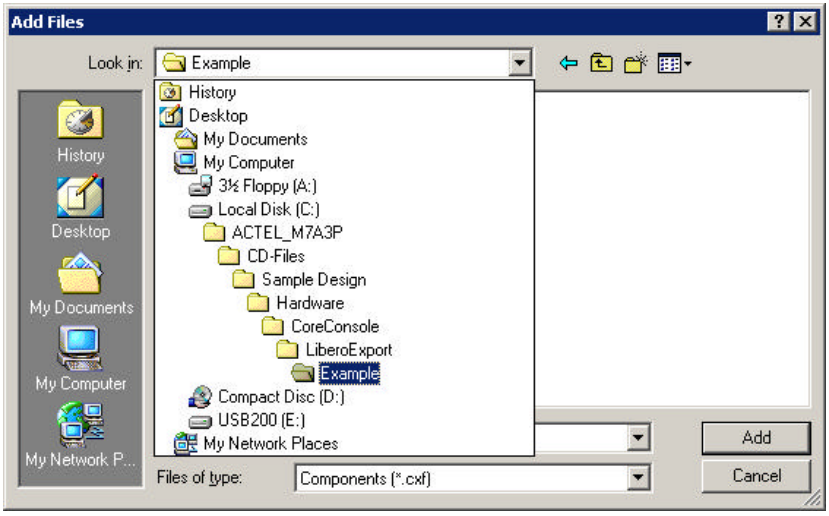


Figure 5.8- Libero IDE New Project Wizard – Browse to Examples Folder

Navigate to the Sample Design\Hardware\CoreConsole\LiberoExport\Example folder as shown.

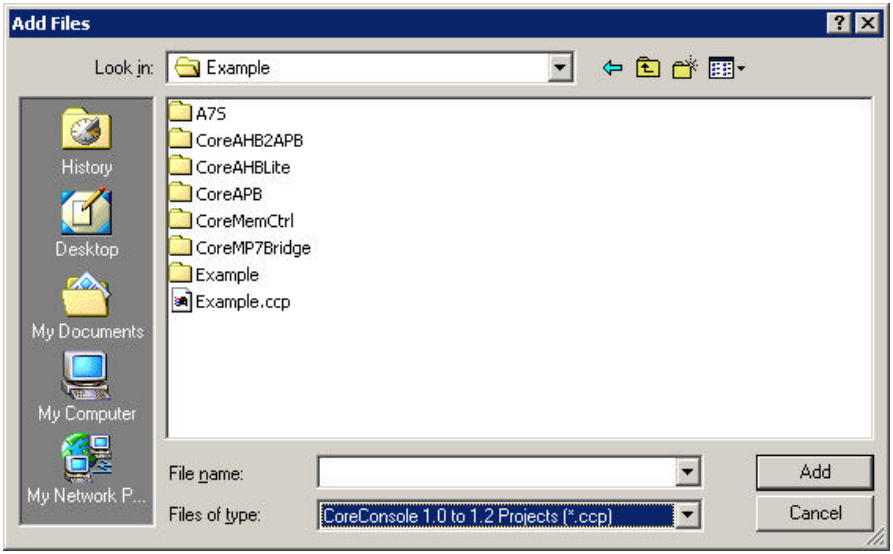


Figure 5.9- Libero IDE New Project Wizard – Select Files to Add

Now add all the following files:

Folder Path	Filename	File of Type
Sample Design\Hardware\CoreConsole\LiberoExport\Example		
	Example.ccp	CoreConsole 1.0 to 1.2 Projects (*.ccp)
Sample Design\Hardware\Source		
	socTop.v	HDL Source Files (*.vhd; *.v; *.h)
	socParPort.v	
	socParPort.h	
	actel_pll_48MHz_16MHz.v	
Sample Design\Hardware\Constraints		
	constraints.sdc	SDC File (*.sdc)
	socTop.pdc	Physical Design Constraint File (*.pdc)

Table 5.1- Libero IDE New Project Files

Click **Next** button.

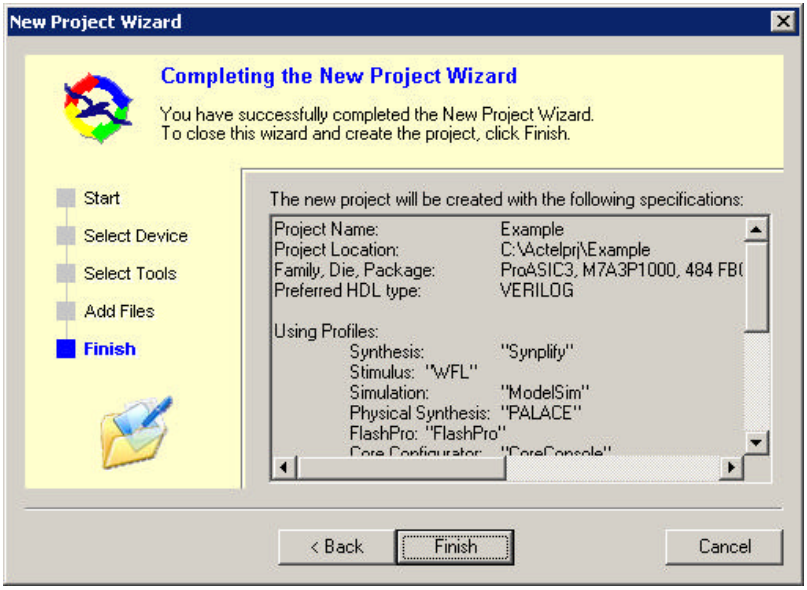


Figure 5.10 - Libero IDE Now Project Wizard - Finish

Now check the parameters and then click the **Finish** button.

Step 4 – Synthesis, Place-and-Route and Generate Programming File

After the functional simulation is done, you can proceed to synthesize and place-and-route the design. The Place&Route tool will generate a .pdb file and/or a “STAPL File” (.stp) for use in programming the ProASIC3 FPGA on the M7A3P Development Board. Be sure to check the box in the Programming File dialog box if you want to create the .stp file.

For more information, see Libero IDE documentation.

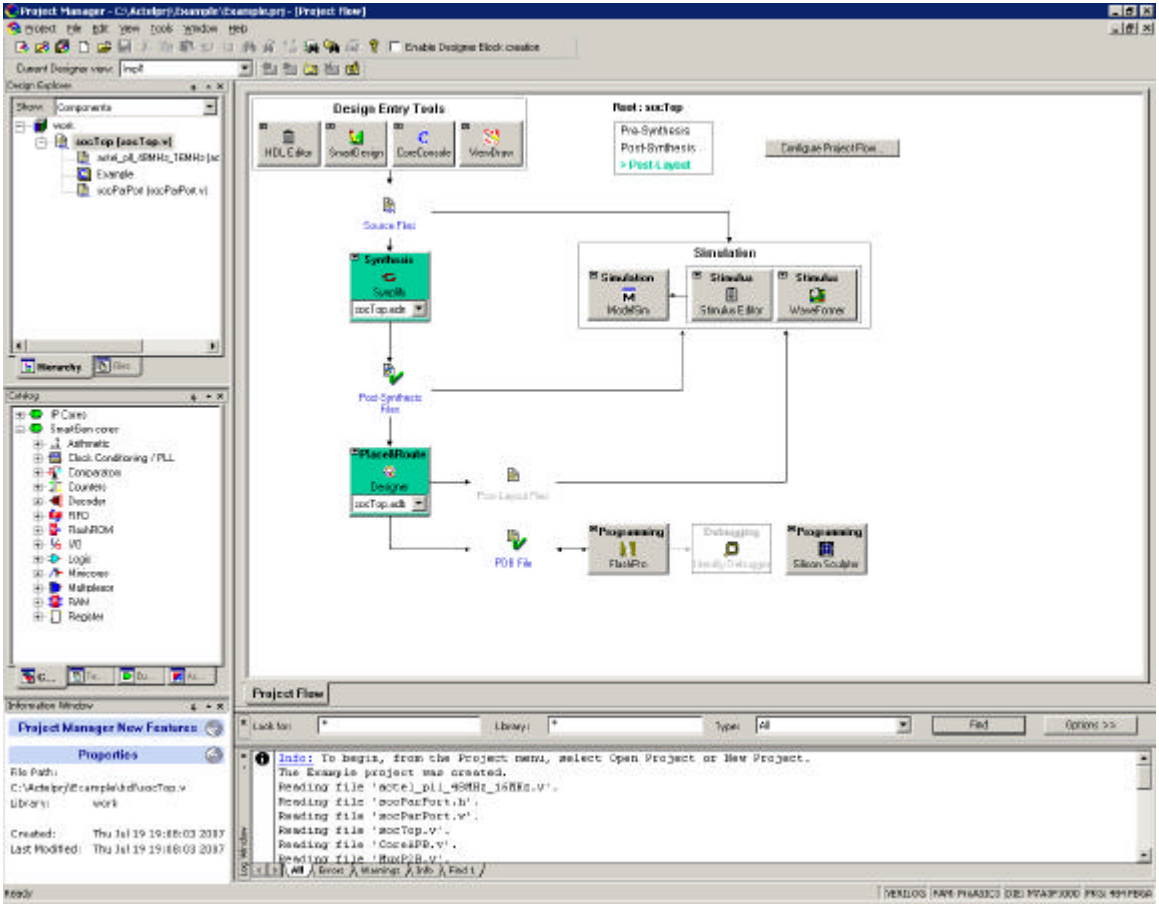


Figure 5.11- Libero IDE

Step 5 – Program the M7A3P1000 on the M7A3P Development Board

Connect the USB cable to the PC and the J1, USB (PROG), connector on the M7A3P Development Board to power up the board. Make sure that switch SW2 is set for switch 9 OFF and all other switches ON.

Follow the FlashPro User Guide to program the M7A3P1000. For reference, the socTop.stp STPL file is in the "Sample Design\Hardware\Device" folder.

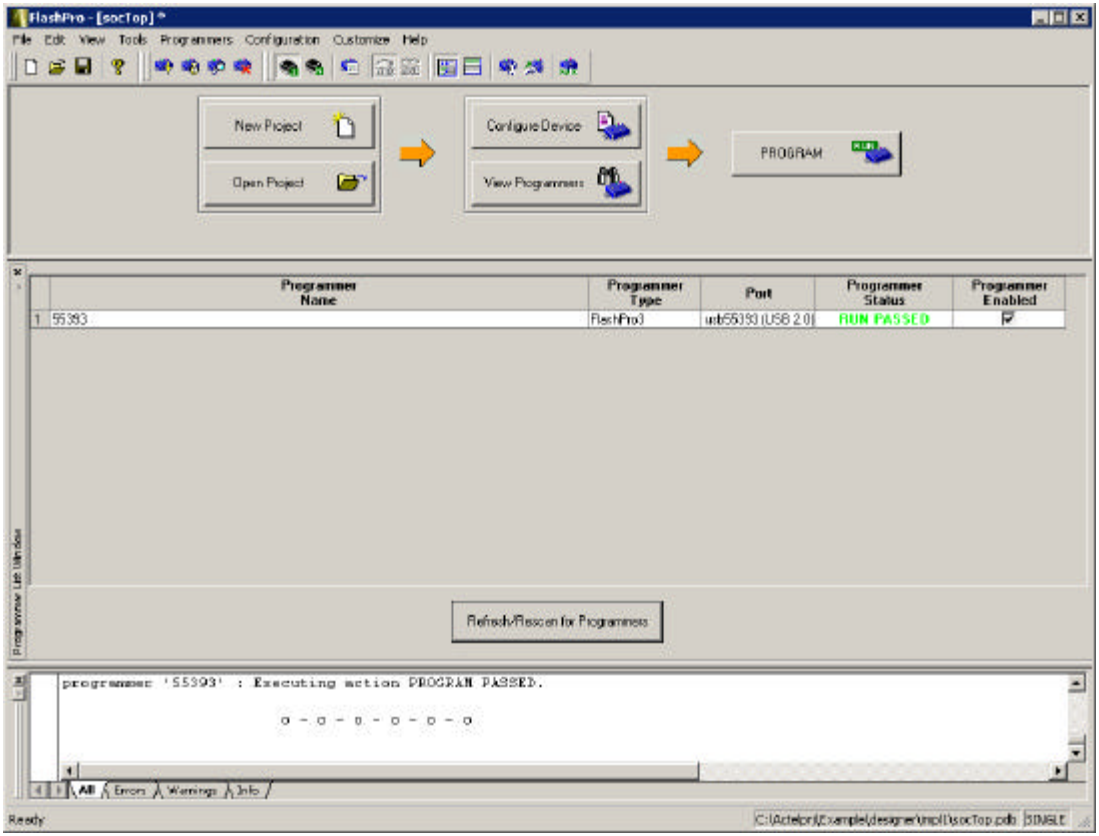


Figure 5.12 - FlashPro3 Programmer GUI

Software

"QuickStart" Example

The QuickStart Sample software periodically samples the switch inputs via the parallel port, then drives the LED outputs again via the parallel port to reflect the state of the switches. Additionally, a separate parallel register drives the top two LEDs to provide a "heartbeat", which indicates that the program is executing.

The following code is the **main()** function.

```
/*****************************************************************************
 * Copyright Actel Corporation.
 *
 * Simple test program flashing LEDs
 *
 */

#include "socParPort.h"
#include "platform.h"
#include "arm.h"

#define PUT_UINT32(addr, data)    *((volatile unsigned int *) (addr)) = (data)
#define GET_UINT32(addr, pData)   (*pData) = *((volatile unsigned int *) (addr))

/*****************************************************************************/
void main( void )
{
    unsigned int switchValue;
    unsigned int i, loopCount;

    //The lower two control bits (otherwise unused) are connected to the top two LEDs
    unsigned int extraControlBits = 0x1;

    DisableIRQ();
    DisableFIQ();

        //Initialize the loopCount
    loopCount = 0;

    for (;;)
    {
       //Configure Parallel Port to accept switch inputs
       PUT_UINT32(PARALLEL_BASE_ADDR+PARALLEL_CONTROL_OFFSET, PARALLEL_CONTROL_DATA_INPUT
| extraControlBits);
       //Read the switch settings through the data register location
       GET_UINT32(PARALLEL_BASE_ADDR+PARALLEL_DATA_OFFSET, &switchValue);
```

```
        //Configure Parallel Port to drive outputs
        PUT_UINT32(PARALLEL_BASE_ADDR+PARALLEL_CONTROL_OFFSET, PARALLEL_CONTROL_DATA_OUTPUT
| extraControlBits);
        //Drive the ouptuts (to LEDs) based on the switch settings
        PUT_UINT32(PARALLEL_BASE_ADDR+PARALLEL_DATA_OFFSET, switchValue & 0xFF);

        //Wait
        for (i = 0; i < 80000; i++ ) { }

        if (loopCount++ == 3)
        {
            //Reset loopCount
                        loopCount = 0;
                        //Toggle lower two control bits.
                        //This produces the "heartbeat" on the top 2 LEDs when program is
executing.
                        extraControlBits = ~extraControlBits & 0x3;
        }
    }
}
```

Step 1 – Load the provided "QuickStart" SoftConsole Project

Launch the SoftConsole compiler/debugger software and choose "Sample Design\Software\SoftConsole" as the Workspace for this session.

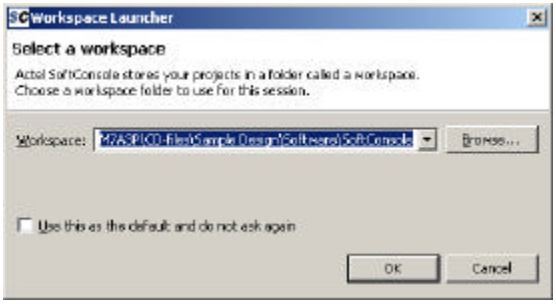


Figure 5.13 - SoftConsole Workspace Dialog

The sample "QuickStart" project should be opened automatically.

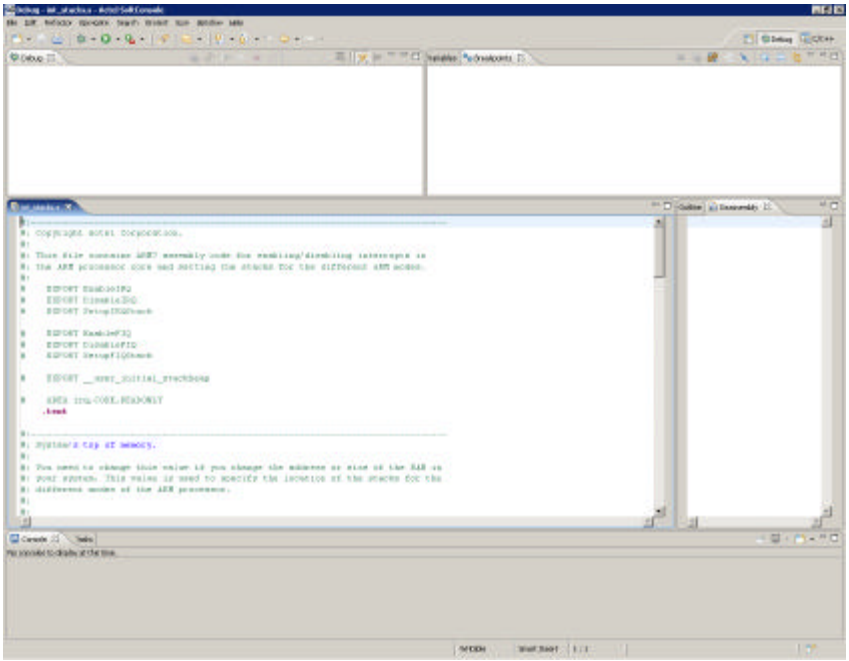


Figure 5.14 - SoftConsole Project

Step 2 – Run a software debugging session

Run the **FS2** Low-Level debugging tool by selecting “Run->External Tools->FlashPro3 On-Chip Debugging for CoreMP7” from the menu.

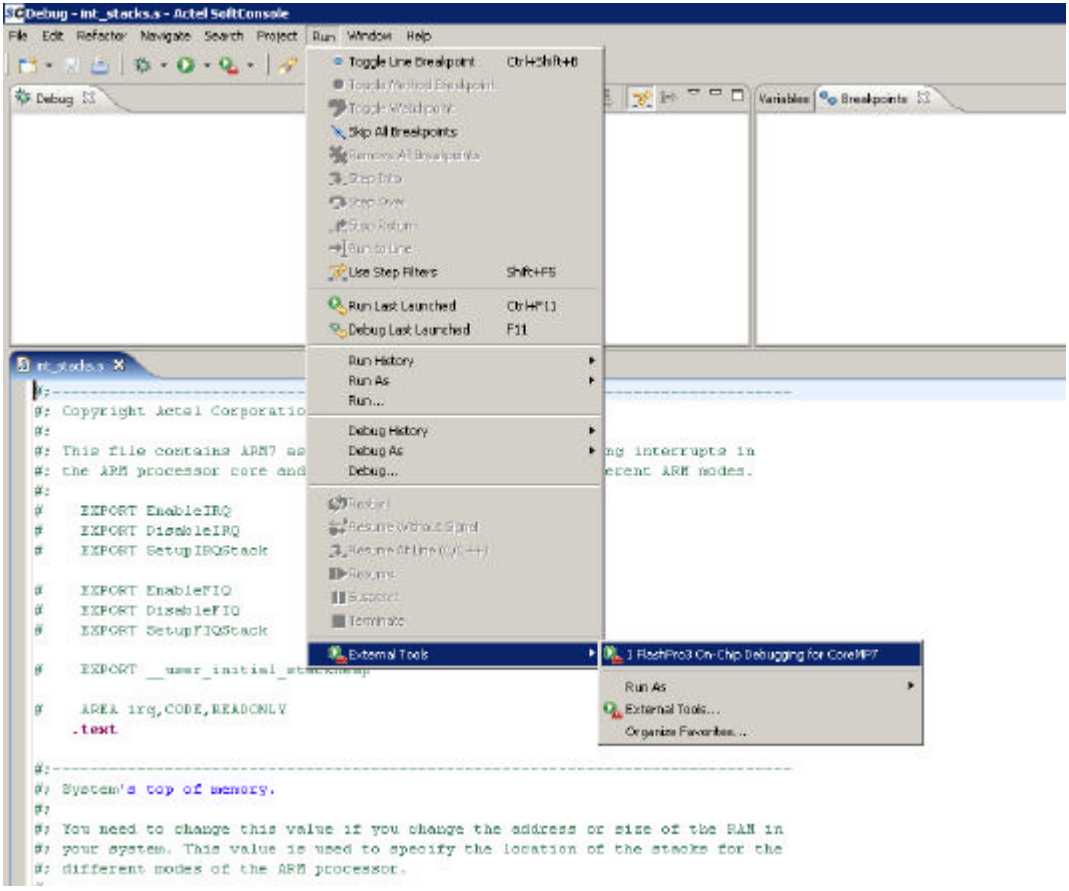


Figure 5.15 - SoftConsole – Debug Start

Verify that the **FS2** Debugging console window appears.

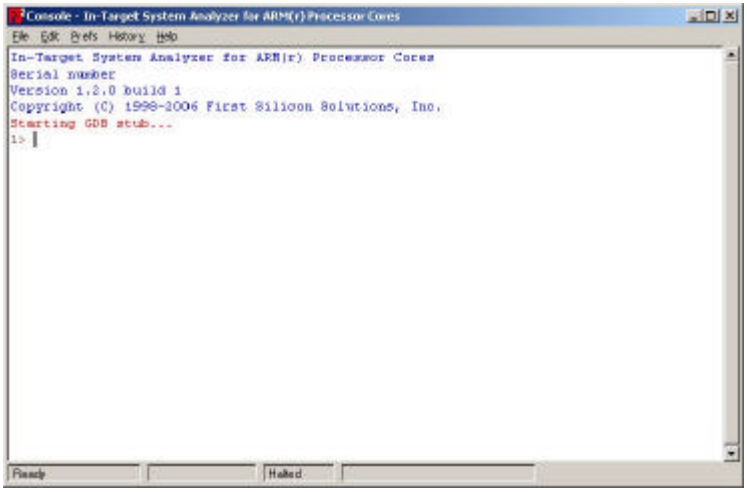


Figure 5.16 - FS2 Console Window

Debug the “QuickStart” example by clicking the Debug Icon. Note that the design is compiled automatically.

Execution should stop once the program reaches the **main()** function.

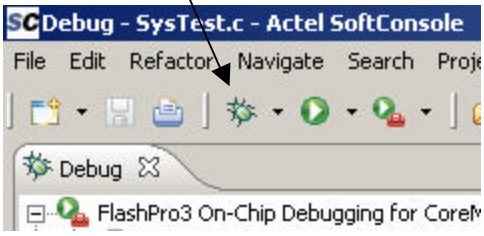


Figure 5.18 - SoftConsole Entering Debug

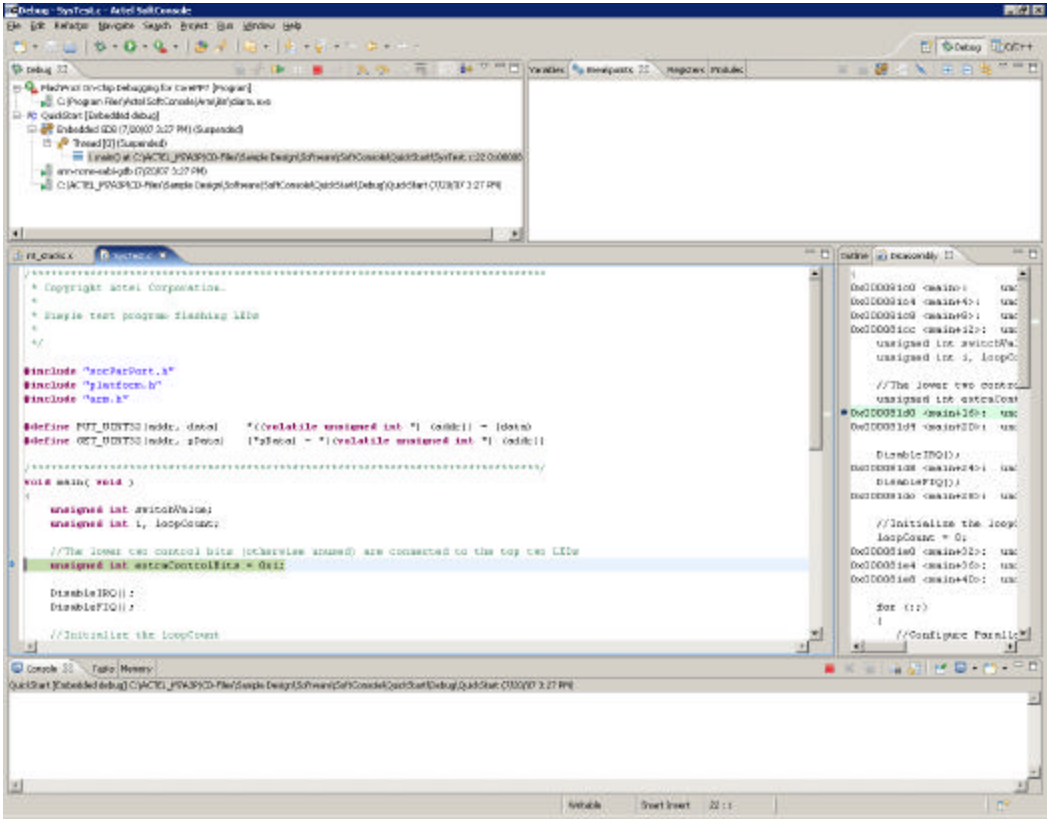


Figure 5.18 - SoftConsole Debug Session

Continue program execution by selecting the Resume Icon. Alternatively, step through code using SoftConsole's single-step or explore other debugging capabilities.

See SoftConsole documentation for more information.

A

FG484 Package for the M7A3P FPGAs

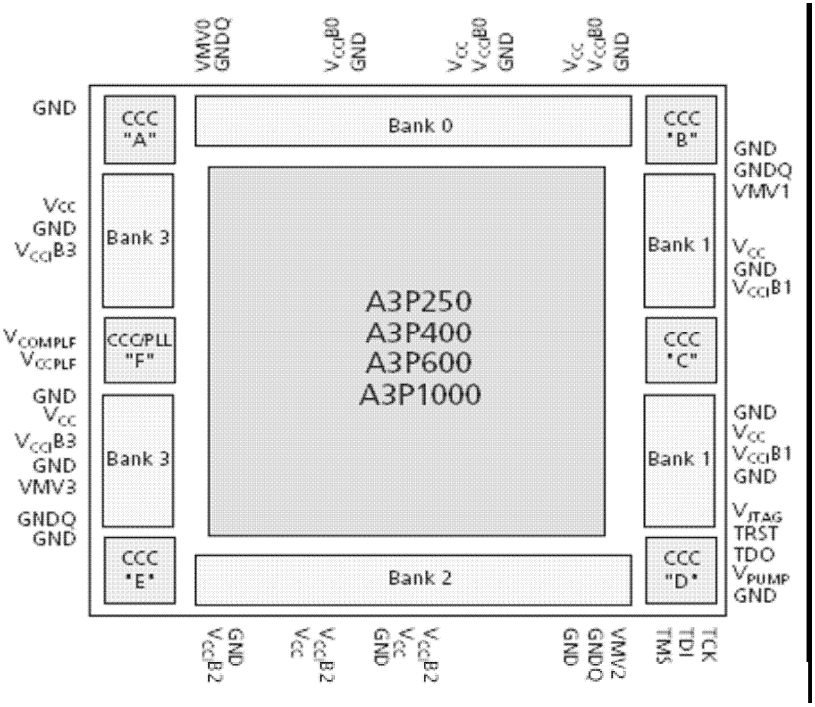


Figure A.1- A3P1000 Layout

484-Pin FBGA (Bottom View)

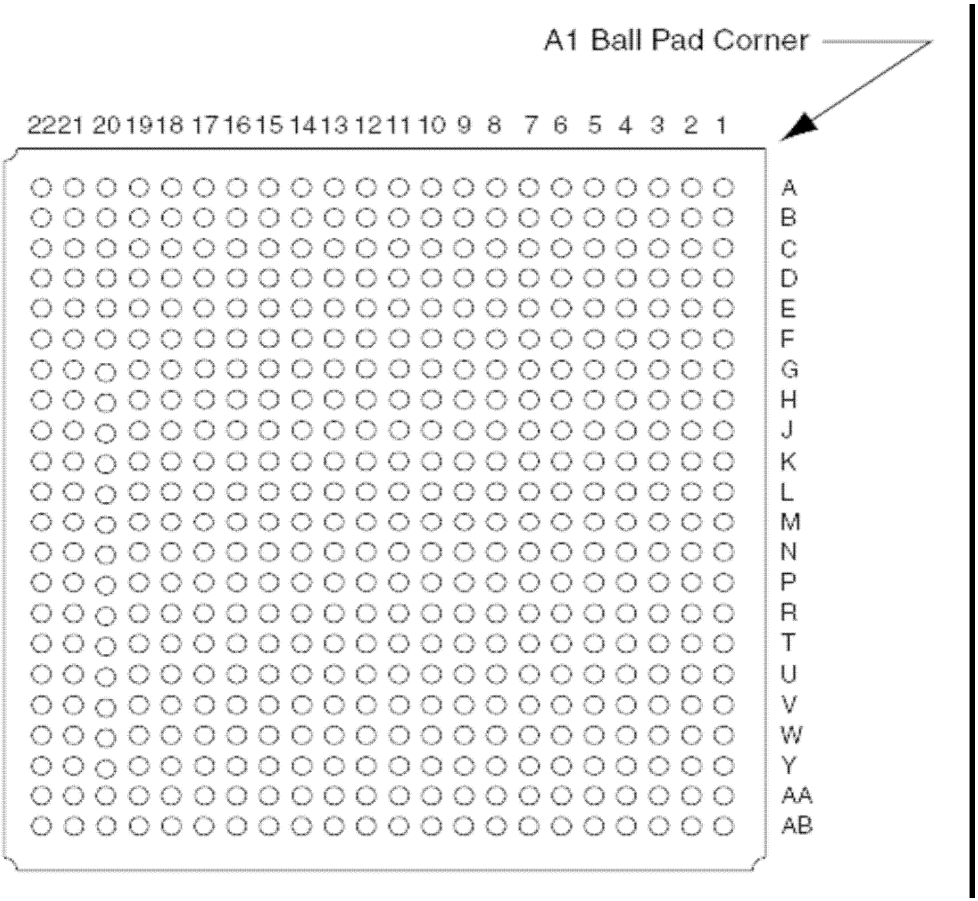


Figure A.2- 484 Pin FBGA Package

Note: For package manufacturing and environmental information, visit the Resource Center at http: / / www.actel.com / products / rescenter / package / index.html.

Due to the comprehensive and flexible nature of M7A3P FPGA device user I / Os, a naming scheme is used to show the details of the I / O. The name identifies to which I / O bank it belongs, as well as the pairing and pin polarity for differential I / Os.

I / O Nomenclature	=	Gmn / IOuxwBy
Gmn is only used for I / Os that also have CCC access – i.e., global pins.		
G	=	Global
m	=	Global pin location associated with each CCC on the device: A (northwest corner), B (northeast corner), C (east middle), D (southeast corner), E (southwest corner), and F (west middle)
n	=	Global input MUX and pin number of the associated Global location m, either A0, A1, A2, B0, B1, B2, C0, C1, or C2
u	=	I / O pair number in the bank, starting at 00 from the northwest I / O bank in a clockwise direction
x	=	P (Positive) or N (Negative) for differential pairs, or R (Regular – single-ended) for the I / Os that support single-ended and voltage-referenced I / O standards only. U (Positive-LVDS only) or V (Negative-LVDS only) restrict the I / O differential pair from being selected as LVPECL pair.
w	=	D (Differential Pair) or P (Pair) or S (Single-Ended). D (Differential Pair) if both members of the pair are bonded out to adjacent pins or are separated only by one GND or NC pin; P (Pair) if both members of the pair are bonded out but do not meet the adjacency requirement; or S (Single-Ended) if the I / O pair is not bonded out. For Differential (D) pairs, adjacency for ball grid packages means only vertical or horizontal. Diagonal adjacency does not meet the requirements for a true differential pair.
B	=	Bank
y	=	Bank number [0..3]. Bank number starting at 0 from the northwest I / O bank in a clockwise direction

Table A.1 - I/O Nomenclature

The following table shows the FPGA pin connections to the M7A3P Development Board signals and the pre-loaded PiP-EC02 design signals.

FPGA Ball	Pre-loaded Design Signal	FPGA Signal	Schematic Signal	I / O	Dev. Kit Function
T19	pbRstN	GDA1 / IO113PDB1	BUF1_PBRST_N	I	Push Button Reset
E18	poRstN	GBA2 / IO78PDB1	PORESET_N	I	Power on Reset
J6	flashRstN	IO216PDB3	FLASH_RST_N	0	Reset to Flash Chips
E4	sysClk	GAB2 / IO224PDB3	OSC_CLK	I	System Clock
N6	ICE_TCK	IO203NDB3	PROC_TCK	I	Jtag Clock
N5	ICE_TDI	IO204NDB3	PROC_TDI	I	Jtag Data In
N7	ICE_TMS	IO203PDB3	PROC_TMS	I	Jtag Mode Select
M7	ICE_TDO	IO205NDB3	PROC_TDO	O	Jtag Data Out
L5	ICE_TRTCK	GFA0 / IO207NDB3	PROC_RTCK	I	Jtag sync
G18	rs232Atx	IO79NDB1	RS232_RX	O	UART Transmit
F18	rs232Arx	IO78NDB1	RS232_TX	I	UART Receive
T6	memAddr[2]	GEC1 / IO190PPB3	MEM_ADDR2	O	SRAM / FLASH Address
R7	memAddr[3]	GEC0 / IO190NPB3	MEM_ADDR3	O	SRAM / FLASH Address
U4	memAddr[4]	GEB1 / IO189PDB3	MEM_ADDR4	O	SRAM / FLASH Address
U5	memAddr[5]	GEB0 / IO189NDB3	MEM_ADDR5	O	SRAM / FLASH Address
V4	memAddr[6]	GEA1 / IO188PDB3	MEM_ADDR6	O	SRAM / FLASH Address
V5	memAddr[7]	GEA0 / IO188NDB3	MEM_ADDR7	O	SRAM / FLASH Address
F5	memAddr[8]	IO225NDB3	MEM_ADDR8	O	SRAM / FLASH Address
F4	memAddr[9]	IO224NDB3	MEM_ADDR9	O	SRAM / FLASH Address
G5	memAddr[10]	IO222PDB3	MEM_ADDR10	O	SRAM / FLASH Address
G4	memAddr[11]	IO222NDB3	MEM_ADDR11	O	SRAM / FLASH Address
H7	memAddr[12]	IO221PDB3	MEM_ADDR12	O	SRAM / FLASH Address
H6	memAddr[13]	IO221NDB3	MEM_ADDR13	O	SRAM / FLASH Address
H5	memAddr[14]	IO218PDB3	MEM_ADDR14	O	SRAM / FLASH Address
J5	memAddr[15]	IO218NDB3	MEM_ADDR15	O	SRAM / FLASH Address
N4	memAddr[16]	GFC2 / IO204PDB3	MEM_ADDR16	O	SRAM / FLASH Address
K7	memAddr[17]	GFC1 / IO209PPB3	MEM_ADDR17	O	SRAM / FLASH Address
L8	memAddr[18]	GFC0 / IO209NPB3	MEM_ADDR18	O	SRAM / FLASH Address
M8	memAddr[19]	GFB2 / IO205PDB3	MEM_ADDR19	O	SRAM / FLASH Address
L6	memAddr[20]	GFB1 / IO208PPB3	MEM_ADDR20	O	SRAM / FLASH Address
L4	memAddr[21]	GFB0 / IO208NPB3	MEM_ADDR21	O	SRAM / FLASH Address
J7	memAddr[22]	IO216NDB3	MEM_ADDR22	O	SRAM / FLASH Address
K6	memAddr[23]	IO213PDB3	MEM_ADDR23	O	SRAM / FLASH Address
K5	memAddr[24]	IO213NDB3	MEM_ADDR24	O	SRAM / FLASH Address

K4	memAddr[25]	IO210PPB3	MEM_ADDR25	O	SRAM / FLASH Address
W10	memData[0]	IO158RSB2	MEM_DATA0	I / O	SRAM / FLASH Data
V10	memData[1]	IO157RSB2	MEM_DATA1	I / O	SRAM / FLASH Data
AB9	memData[2]	IO156RSB2	MEM_DATA2	I / O	SRAM / FLASH Data
V9	memData[3]	IO163RSB2	MEM_DATA3	I / O	SRAM / FLASH Data
W9	memData[4]	IO164RSB2	MEM_DATA4	I / O	SRAM / FLASH Data
U9	memData[5]	IO165RSB2	MEM_DATA5	I / O	SRAM / FLASH Data
AB8	memData[6]	IO162RSB2	MEM_DATA6	I / O	SRAM / FLASH Data
W8	memData[7]	IO170RSB2	MEM_DATA7	I / O	SRAM / FLASH Data
V8	memData[8]	IO168RSB2	MEM_DATA8	I / O	SRAM / FLASH Data
U8	memData[9]	IO171RSB2	MEM_DATA9	I / O	SRAM / FLASH Data
AB7	memData[10]	IO167RSB2	MEM_DATA10	I / O	SRAM / FLASH Data
AB6	memData[11]	IO173RSB2	MEM_DATA11	I / O	SRAM / FLASH Data
W7	memData[12]	IO172RSB2	MEM_DATA12	I / O	SRAM / FLASH Data
V7	memData[13]	GEC2 / IO185RSB2	MEM_DATA13	I / O	SRAM / FLASH Data
T9	memData[14]	GEA2 / IO187RSB2	MEM_DATA14	I / O	SRAM / FLASH Data
W5	memData[15]	IO183RSB2	MEM_DATA15	I / O	SRAM / FLASH Data
V15	memData[16]	IO125RSB2	MEM_DATA16	I / O	SRAM / FLASH Data
U14	memData[17]	IO128RSB2	MEM_DATA17	I / O	SRAM / FLASH Data
T13	memData[18]	IO129RSB2	MEM_DATA18	I / O	SRAM / FLASH Data
W14	memData[19]	IO130RSB2	MEM_DATA19	I / O	SRAM / FLASH Data
V14	memData[20]	IO131RSB2	MEM_DATA20	I / O	SRAM / FLASH Data
U13	memData[21]	IO134RSB2	MEM_DATA21	I / O	SRAM / FLASH Data
W13	memData[22]	IO135RSB2	MEM_DATA22	I / O	SRAM / FLASH Data
R12	memData[23]	IO136RSB2	MEM_DATA23	I / O	SRAM / FLASH Data
U12	memData[24]	IO137RSB2	MEM_DATA24	I / O	SRAM / FLASH Data
V13	memData[25]	IO138RSB2	MEM_DATA25	I / O	SRAM / FLASH Data
T12	memData[26]	IO141RSB2	MEM_DATA26	I / O	SRAM / FLASH Data
W12	memData[27]	IO142RSB2	MEM_DATA27	I / O	SRAM / FLASH Data
V12	memData[28]	IO143RSB2	MEM_DATA28	I / O	SRAM / FLASH Data
R11	memData[29]	IO147RSB2	MEM_DATA29	I / O	SRAM / FLASH Data
V11	memData[30]	IO149RSB2	MEM_DATA30	I / O	SRAM / FLASH Data
W11	memData[31]	IO153RSB2	MEM_DATA31	I / O	SRAM / FLASH Data
J4	flashHiCeN	IO217NDB3	FLASH_HCE_N	O	Flash Chip Enable (high chip)
H4	flashLoCeN	IO217PDB3	FLASH_LCE_N	O	Flash Chip Enable (low chip)
AB12	flashWeN	IO144RSB2	FLASH_WE_N	O	Flash Write Enable
AB13	flashOeN	IO132RSB2	FLASH_OE_N	O	Flash Output Enable
AB17	sramCeN	IO121RSB2	SRAM_CE_N	O	SRAM Chip Enable
W17	sramBsN[0]	GDA2 / IO114RSB2	SRBS0_N	O	SRAM Byte Select 0

V16	sramBsN[1]	GDB2 / IO115RSB2	SRBS1_N	O	SRAM Byte Select 1
W15	sramBsN[2]	GDC2 / IO116RSB2	SRBS2_N	O	SRAM Byte Select 2
W16	sramBsN[3]	IO120RSB2	SRBS3_N	O	SRAM Byte Select 3
AB16	sramWeN	IO123RSB2	SRAM_WE_N	O	SRAM Write Enable
T14	sramOeN	IO124RSB2	SRAM_OE_N	O	SRAM Output Enable
R17	ledOut[0]	GDB1 / IO112PPB1	LED0	O	Drives LED 0
P16	ledOut[1]	GDB0 / IO112NPB1	LED1	O	Drives LED 1
U19	ledOut[2]	GDA0 / IO113NDB1	LED2	O	Drives LED 2
M15	ledOut[3]	GCB2 / IO95PPB1	LED3	O	Drives LED 3
L16	ledOut[4]	GCB1 / IO92PPB1	LED4	O	Drives LED 4
L19	ledOut[5]	GCB0 / IO92NPB1	LED5	O	Drives LED 5
M19	ledOut[6]	GCA2 / IO94PPB1	LED6	O	Drives LED 6
M16	ledOut[7]	GCA1 / IO93PPB1	LED7	O	Drives LED 7
L17	ledOut[8]	GCA0 / IO93NPB1	LED8	O	Drives LED 8
N16	ledOut[9]	IO95NPB1	LED9	O	Drives LED 9
K21	switchIn[0]	IO94PPB1	SWITCH0	I	Switch Input 0
K20	switchIn[1]	IO94NPB1	SWITCH1	I	Switch Input 1
J18	switchIn[2]	IO90PPB1	SWITCH2	I	Switch Input 2
K17	switchIn[3]	IO90NPB1	SWITCH3	I	Switch Input 3
K18	switchIn[4]	IO88PDB1	SWITCH4	I	Switch Input 4
K19	switchIn[5]	IO88NDB1	SWITCH5	I	Switch Input 5
H19	switchIn[6]	IO87PDB1	SWITCH6	I	Switch Input 6
J19	switchIn[7]	IO87NDB1	SWITCH7	I	Switch Input 7
J17	switchIn[8]	IO86NPB1	SWITCH8	I	Switch Input 8
H18	switchIn[9]	IO86PPB1	SWITCH9	I	Switch Input 9
D7	stnLoad	GAB0 / IO02RSB0	LS_ALE	O	STN LCD Control
E7	stnFrame	GAB1 / IO03RSB0	LS_IORD	O	STN LCD Control
F8	stnCp	GAC0 / IO04RSB0	LS_IOWR	O	STN LCD Clock
F9	stnDisp	GAC1 / IO05RSB0	LS_IOEN	O	STN LCD Display Enable
A13	stnOut[0]	IO53RSB0	LS_D0	O	Drives STN Data 0
D14	stnOut[1]	IO55RSB0	LS_D1	O	Drives STN Data 1
F14	stnOut[2]	IO56RSB0	LS_D2	O	Drives STN Data 2
E14	stnOut[3]	IO57RSB0	LS_D3	O	Drives STN Data 3
G14	stnOut[4]	IO60RSB0	LS_D4	O	Drives STN Data 4
D15	stnOut[5]	IO61RSB0	LS_D5	O	Drives STN Data 5
A16	stnOut[6]	IO65RSB0	LS_D6	O	Drives STN Data 6
A17	stnOut[7]	IO67RSB0	LS_D7	O	Drives STN Data 7
A6	gpio0[0]	IO13RSB0	LS_D8	I / O	General Purpose IO
D8	gpio0[1]	IO16RSB0	LS_D9	I / O	General Purpose IO

E8	gpio0[2]	IO17RSB0	LS_D10	I / O	General Purpose IO
A7	gpio0[3]	IO18RSB0	LS_D11	I / O	General Purpose IO
E9	gpio0[4]	IO21RSB0	LS_D12	I / O	General Purpose IO
D9	gpio0[5]	IO22RSB0	LS_D13	I / O	General Purpose IO
G9	gpio0[6]	IO23RSB0	LS_D14	I / O	General Purpose IO
F10	gpio0[7]	IO25RSB0	LS_D15	I / O	General Purpose IO
D5	gpio0[8]	GAA0 / IO00RSB0	RF_CLK	I / O	General Purpose IO
D6	gpio0[9]	GAA1 / IO01RSB0	RF_SCLK	I / O	General Purpose IO
G6	gpio0[10]	GAC2 / IO223PDB3	RTC_SCL	I / O	General Purpose IO
T5	gpio0[11]	IO192PPB3	RTC_SDA	I / O	General Purpose IO
E10	gpio0[12]	IO27RSB0	LS_A2	I / O	General Purpose IO
D10	gpio0[13]	IO28RSB0	LS_A3	I / O	General Purpose IO
G10	gpio0[14]	IO29RSB0	LS_A4	I / O	General Purpose IO
A10	gpio0[15]	IO32RSB0	LS_A5	I / O	General Purpose IO
G11	gpio0[16]	IO33RSB0	LS_A6	I / O	General Purpose IO
E11	gpio0[17]	IO34RSB0	LS_A7	I / O	General Purpose IO
D11	gpio0[18]	IO35RSB0	LS_A8	I / O	General Purpose IO
F11	gpio0[19]	IO36RSB0	LS_A9	I / O	General Purpose IO
A12	gpio0[20]	IO41RSB0	LS_OE_N	I / O	General Purpose IO
D17	gpio0[21]	GBA0 / IO76RSB0	LS_INT_N	I / O	General Purpose IO
B6	gpio0[22]	IO12RSB0	RF_MAG	I / O	General Purpose IO
B7	gpio0[23]	IO15RSB0	RF_SGN	I / O	General Purpose IO
F19	gpio0[24]	IO81NDB1	DIFFA1N	I / O	General Purpose IO
E19	gpio0[25]	IO81PDB1	DIFFA1P	I / O	General Purpose IO
R19	gpio0[26]	IO107NDB1	DIFFA2N	I / O	General Purpose IO
P19	gpio0[27]	IO107PDB1	DIFFA2P	I / O	General Purpose IO
P17	gpio0[28]	IO106NDB1	DIFFB1N	I / O	General Purpose IO
P18	gpio0[29]	IO106PDB1	DIFFB1P	I / O	General Purpose IO
T18	gpio0[30]	GDC0 / IO111NDB1	DIFFB2N	I / O	General Purpose IO
R18	gpio0[31]	GDC1 / IO111PDB1	DIFFB2P	I / O	General Purpose IO
L15	gpio1[0]	GCC0 / IO91NPB1	GPIOA_0	I / O	General Purpose IO
K16	gpio1[1]	GCC1 / IO91PPB1	GPIOA_1	I / O	General Purpose IO
M17	gpio1[2]	GCC2 / IO96PPB1	GPIOA_2	I / O	General Purpose IO
N17	gpio1[3]	IO100NPB1	GPIOA_3	I / O	General Purpose IO
N18	gpio1[4]	IO102NDB1	GPIOA_4	I / O	General Purpose IO
N19	gpio1[5]	IO102PDB1	GPIOA_5	I / O	General Purpose IO
U11	gpio1[6]	IO151RSB2	GPIOA_6	I / O	General Purpose IO
T11	gpio1[7]	IO155RSB2	GPIOA_7	I / O	General Purpose IO
U10	gpio1[8]	IO159RSB2	GPIOA_8	I / O	General Purpose IO

T10	gpio1[9]	IO161RSB2	GPIOA_9	I / O	General Purpose IO
R22	gpio1[10]	IO105PDB1	GPIOA_10	I / O	General Purpose IO
E5	gpio1[11]	GAA2 / IO225PDB3	GPIOA_11	I / O	General Purpose IO
P4	gpio1[12]	IO202NDB3	GPIOA_12	I / O	General Purpose IO
P5	gpio1[13]	IO202PDB3	GPIOA_13	I / O	General Purpose IO
M4	gpio1[14]	GFA2 / IO206PDB3	GPIOA_14	I / O	General Purpose IO
M5	gpio1[15]	GFA1 / IO207PDB3	GPIOA_15	I / O	General Purpose IO
AA16	gpio1[16]	IO122RSB2	GPIOA_16	I / O	General Purpose IO
AA13	gpio1[17]	IO133RSB2	GPIOA_17	I / O	General Purpose IO
AA10	gpio1[18]	IO152RSB2	GPIOA_18	I / O	General Purpose IO
AA7	gpio1[19]	IO169RSB2	GPIOA_19	I / O	General Purpose IO
AA6	gpio1[20]	IO175RSB2	GPIOA_20	I / O	General Purpose IO
R2	gpio1[21]	IO197PPB3	GPIOA_21	I / O	General Purpose IO
T2	gpio1[22]	IO198NDB3	GPIOA_22	I / O	General Purpose IO
P2	gpio1[23]	IO199PDB3	GPIOA_23	I / O	General Purpose IO
M2	gpio1[24]	IO200NDB3	GPIOA_24	I / O	General Purpose IO
L2	gpio1[25]	IO200PDB3	GPIOA_25	I / O	General Purpose IO
N1	gpio1[26]	IO201NDB3	GPIOA_26	I / O	General Purpose IO
K1	gpio1[27]	IO211PDB3	GPIOA_27	I / O	General Purpose IO
J1	gpio1[28]	IO212NDB3	GPIOA_28	I / O	General Purpose IO
J2	gpio1[29]	IO212PDB3	GPIOA_29	I / O	General Purpose IO
G1	gpio1[30]	IO214NDB3	GPIOA_30	I / O	General Purpose IO
G2	gpio1[31]	IO214PDB3	GPIOA_31	I / O	General Purpose IO
H11	gpio2[0]	IO38RSB0	GPIOB_0	I / O	General Purpose IO
F12	gpio2[1]	IO42RSB0	GPIOB_1	I / O	General Purpose IO
E12	gpio2[2]	IO44RSB0	GPIOB_2	I / O	General Purpose IO
D12	gpio2[3]	IO45RSB0	GPIOB_3	I / O	General Purpose IO
G12	gpio2[4]	IO46RSB0	GPIOB_4	I / O	General Purpose IO
H12	gpio2[5]	IO47RSB0	GPIOB_5	I / O	General Purpose IO
F13	gpio2[6]	IO49RSB0	GPIOB_6	I / O	General Purpose IO
D13	gpio2[7]	IO50RSB0	GPIOB_7	I / O	General Purpose IO
E13	gpio2[8]	IO51RSB0	GPIOB_8	I / O	General Purpose IO
G13	gpio2[9]	IO52RSB0	GPIOB_9	I / O	General Purpose IO
D18	gpio2[10]	GBA1 / IO77RSB0	GPIOB_10	I / O	General Purpose IO
E16	gpio2[11]	GBB0 / IO74RSB0	GPIOB_11	I / O	General Purpose IO
D16	gpio2[12]	GBB1 / IO75RSB0	GPIOB_12	I / O	General Purpose IO
F15	gpio2[13]	GBC0 / IO72RSB0	GPIOB_13	I / O	General Purpose IO
E15	gpio2[14]	GBC1 / IO73RSB0	GPIOB_14	I / O	General Purpose IO
G17	gpio2[15]	GBB2 / IO79PDB1	GPIOB_15	I / O	General Purpose IO

G19	gpio2[16]	IO82NPB1	GPIOB_16	I / O	General Purpose IO
H16	gpio2[17]	GBC2 / IO80PDB1	GPIOB_17	I / O	General Purpose IO
J16	gpio2[18]	IO83NPB1	GPIOB_18	I / O	General Purpose IO
H17	gpio2[19]	IO83PPB1	GPIOB_19	I / O	General Purpose IO
J22	gpio2[20]	IO89NDB1	GPIOB_20	I / O	General Purpose IO
J21	gpio2[21]	IO89PDB1	GPIOB_21	I / O	General Purpose IO
L22	gpio2[22]	IO99NPB1	GPIOB_22	I / O	General Purpose IO
K22	gpio2[23]	IO98PDB1	GPIOB_23	I / O	General Purpose IO
M21	gpio2[24]	IO99PPB1	GPIOB_24	I / O	General Purpose IO
N21	gpio2[25]	IO101NPB1	GPIOB_25	I / O	General Purpose IO
P22	gpio2[26]	IO103NDB1	GPIOB_26	I / O	General Purpose IO
R21	gpio2[27]	IO104NDB1	GPIOB_27	I / O	General Purpose IO
B17	gpio2[28]	IO68RSB0	GPIOB_28	I / O	General Purpose IO
B16	gpio2[29]	IO66RSB0	GPIOB_29	I / O	General Purpose IO
B13	gpio2[30]	IO54RSB0	GPIOB_30	I / O	General Purpose IO
B10	gpio2[31]	IO31RSB0	GPIOB_31	I / O	General Purpose IO

Table A.2- M7A3P FPGA Signals

B

Board Schematics

For detailed Schematic, please refer to the **“M7A3P Schematics.pdf”** in the **“Dev Kit Documentation”** folder.

There are 8 pages to the Schematics, titled as follows:

1. POWER SUPPLIES
2. PWR, CLOCK, ETC.
3. SRAM & FLASH
4. RESET, TEST, USB
5. GPS APP. SUPPORT
6. FPGA
7. FPGA
8. FLASHPRO3

The schematic number is SOC-M7A3P-S-002. Please reference this number and the schematic sheet when contacting support@socsolutions.com for M7A3P Development Board and schematic support.

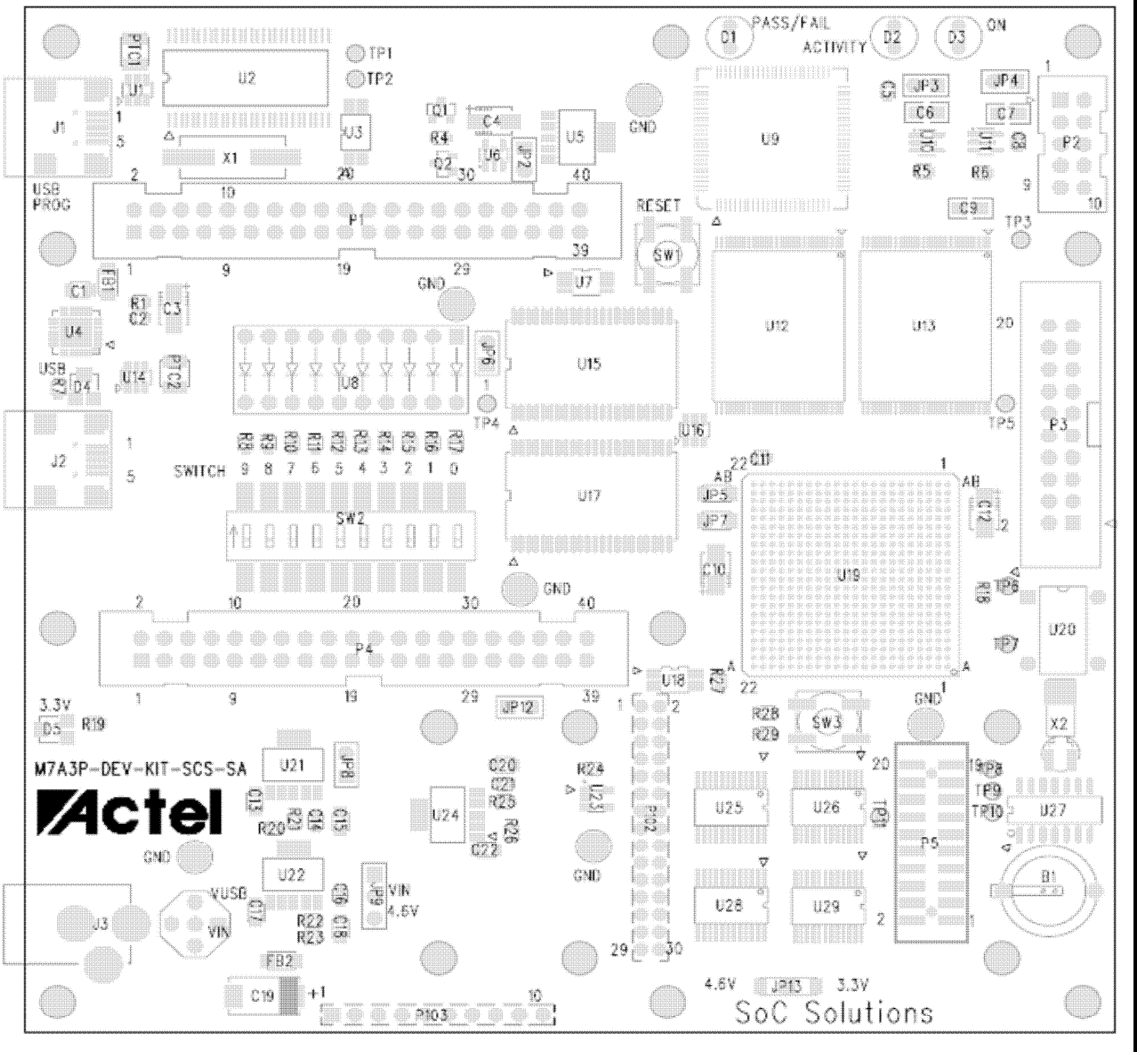


Table B..1 - M7A3P Development Board Top Silkscreen

Product Support

SoC Solutions is providing technical and non-technical support for this product. The product is distributed through Actel and its distributors. For pricing information, please contact Actel. For all other support, please contact SoC Solutions by email or phone.

Technical Support

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 support@socsolutions.com.

Phone

Our Support Staff can help you with hardware, software and installation issues. SoC Solutions will retrieve information, such as your name, company name, phone number and your question, and then issues a case number. The phone hours are from 10:00 A.M. to 6:00 P.M., Eastern Time, Monday through Friday. The Technical Support numbers are:

770-680-2500 Ask for technical support for Actel.

Customers needing assistance outside the US time zones can either contact technical support via email support@socsolutions.com.