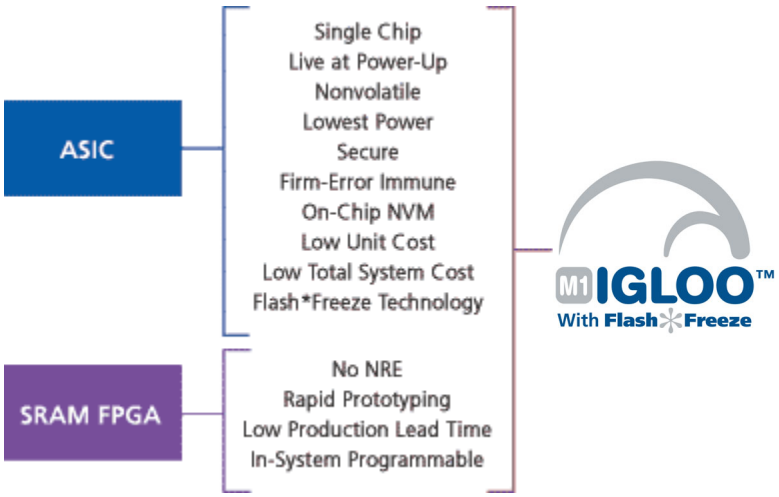
Version 1.5

Introduction

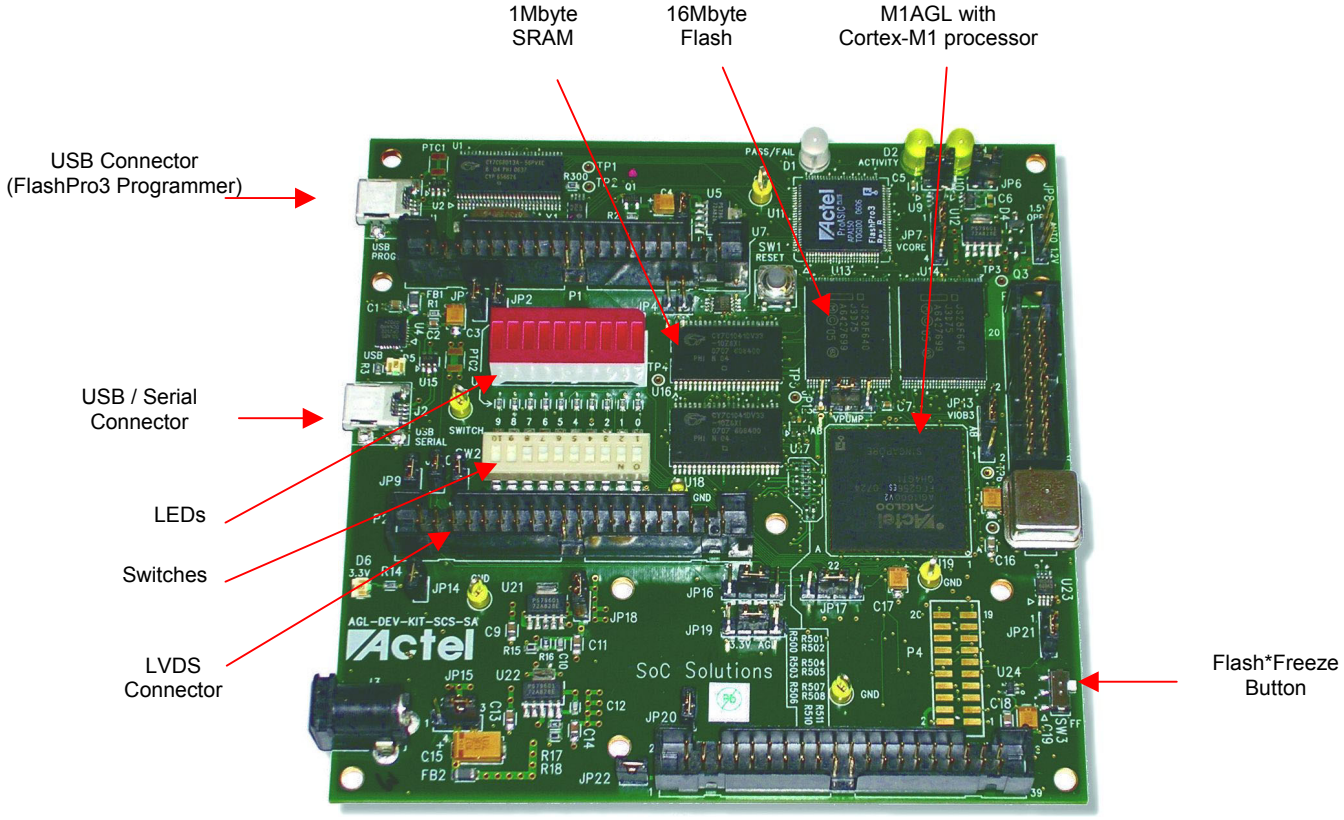
Thank you for purchasing the Actel Cortex-M1 Enabled IGLOO™ Development Kit.

Key Features

- Ultra-low power in Flash*Freeze mode
- Low power active capability
- Small footprint packages
- Reprogrammable flash technology
- 1.2 V or 1.5 V operation
- High capacity, advanced I/O
- Clock conditioning circuit (CCC) and PLL
- Embedded SRAM and nonvolatile memory (NVM)
- In-system programming (ISP) and security
- Cortex-M1 processor



The Cortex-M1 Enabled IGLOO ™ Development Kit is an advanced microprocessor based FPGA development and evaluation kit. The purpose of the kit is to help the user become familiar with the IGLOO FPGA features by providing a useful Sample Design, with a "How To" tutorial for implementing the FPGA hardware design using Libero Project Manager and CoreConsole. The tutorial also shows how to implement Cortex-M1 embedded software using SoftConsole.



M1AGL Development Board

Table of Contents

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 are familiar with:
    - Actel Libero IDE v8.1 or later software.
    - Actel CoreConsole® v1.4 or later software.
    - Actel SoftConsole® v2.0 or later software.
- You are familiar with PCs and the Windows® operating system.

Contents and System Requirements

This chapter details the contents of the Cortex-M1 Enabled IGLOO Development Kit and lists the system requirements.

Cortex-M1 Enabled IGLOO Development Kit Contents

The Cortex-M1 Enabled IGLOO Development Kit includes the following:

- M1AGL Development Board
- Libero IDE v8.x DVD (in DVD case)
- Cortex-M1 Enabled IGLOO Development Kit Install CD with sample design
- +5.0 V external power supply with international adaptors
- 2 USB A to Mini-B Cables
- 4 Self-Adhesive Rubber Pads
- Quick Start Guide

System Requirements

The Cortex-M1 Enabled IGLOO Development Kit requires the following:

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

Additional Information

For further information, refer to the following appendices:

Appendix A – "Package Connections"

Appendix B – "Board Schematics"

Appendix C – "User Tests"

Appendix D – "Support"

Hardware Components

This chapter describes the hardware components of the Cortex-M1 Enabled IGLOO Development Kit.

Ideal Uses for the development kit

Ideal uses for the Cortex-M1 Enabled IGLOO Development Kit are the following:

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

Applications

The Cortex-M1 Enabled IGLOO Development Kit is ideal for use in the following applications:

- Smartphones, GPS, DCAM, PDA
- Portable industrial & medical equipment
- PC laptops, PCMCIA
- Any ultra-low power devices

M1AGL Development Board

The M1AGL Development Board has these features:

- Actel M1AGL600 IGLOO FPGA
- 1 MByte SRAM
- 16 MByte Flash
- USB–RS232 converter chip
- GPIO connectors
- LVDS
- Ultra Low Power with Flash*Freeze technology
- On-board FlashPro3 circuitry
- 20-Pin Cortex-M1 JTAG connector
- Socketed Crystal Oscillator
- Pushbutton power-on reset circuit
- 10 test LEDs
- 10 test switches
- Expansion connectors

Detailed Board Description and Usage

The Cortex-M1 Enabled IGLOO 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 a Cortex-M1 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 M1AGL Development Board schematics contents.

Block Diagram

The following simplified block diagram shows the main features of the M1AGL Development Board. The blocks in dashed lines are not normally installed.

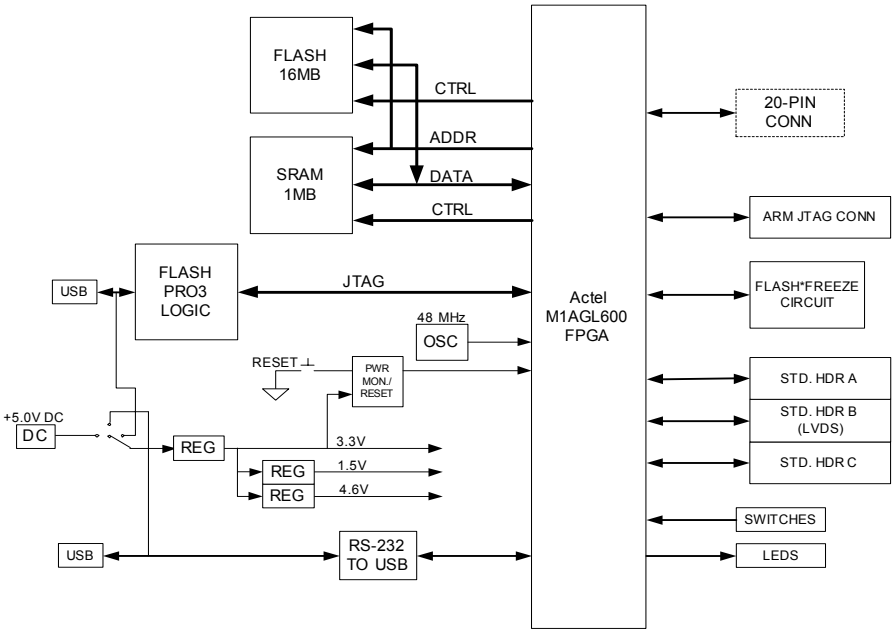


Figure 2.1 - M1AGL Development Board Block Diagram

Power

The development kit may only be powered through J3 with the external +5.0 V 2.1 mm positive-center power supply that is included with the kit. Jumper JP22 should be installed in the 3-5 position to allow this. This jumper is a convenient place to measure the input current. The board was originally designed to allow USB powering options; however, the USB inrush current was too high so these options were removed.

2 USB "mini-B to A" cables are supplied but are only used to power the FlashPro3 and Serial-USB circuitry. Both USB ports can be used simultaneously. 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. It is also possible to use no USB after the M1AGL600 is programmed.

There are two USB power rails on the board. Please refer to Figure 2-2. 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 IGLOO FPGA.

Q1, a P-channel MOSFET device, is used to hold off powering up most of the FlashPro3 logic 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 three regulator components on the board to provide 1.5 V and/or 1.2 V, 2.5 V, and 3.3 V to the IGLOO FPGA.

LED D6 indicates that the 3.3V regulator is operating. The 3.3 V supply is used to provide the VPUMP programming voltage.

The IGLOO FPGA's core voltage is provided by voltage regulator U12. The output voltage of this regulator depends on the setting of jumper JP5. This voltage can be either fixed 1.5V, fixed 1.2V, or can automatically switch to 1.5V during FPGA configuration, then switch to 1.2V for operation. See jumper settings below in Table 2-1.

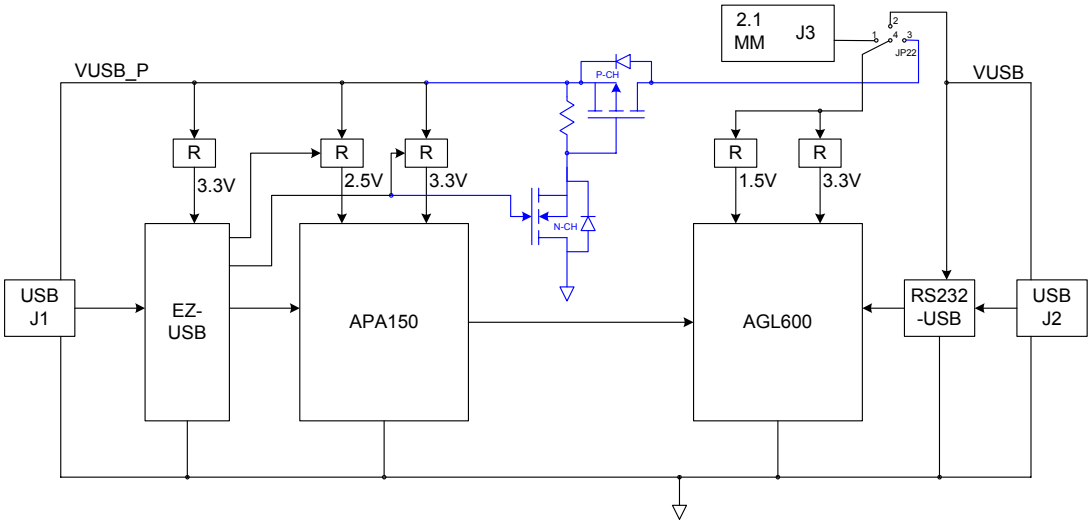


Figure 2.2- M1AGL Development Board Power Configuration

Accessory Card Power Supply Connections

Limited power may be supplied by the IGLOO Development Kit to an accessory card. Connectors for accessory cards (headers P1, P2, & P5) are shown on page 2 of the schematics. The 5 V input voltage and the 2.5 V and 3.3 V regulated voltages are provided to the accessory card connectors.

Cortex-M1 Support

A standard 20-pin Cortex-M1 JTAG header (P3) is supplied to enable use of any ARM standard emulator and debug environment. Alternatively USB connector J1 (PROG) and the on-board FlashPro3 circuitry may be used as a debug path to the Cortex-M1's internal debug interface. Note that the FPGA design must instantiate the Cortex-M1 CoreConsole component to include the debug logic for software development.

Connected to the FPGA for use by the Cortex-M1 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. The non-push-button reset is usually used for the Cortex-M1's JTAG nTRST signal. Other reset schemes may also be used.

USB Serial Interface

USB Connector J2 (SERIAL) provides a USB-to-RS-232 interface through U4. In this configuration, the IGLOO FPGA should contain a UART core.

Measuring Current

The FPGA I/O banks are powered separately through jumpers. See the schematic excerpt below. The jumpers are 4 pins instead of 2 to allow connection of an ammeter on the outer 2 pins and subsequent removal of the shunt. This way current may be measured without shutting down power. Note that the DMM must have a microammeter setting.

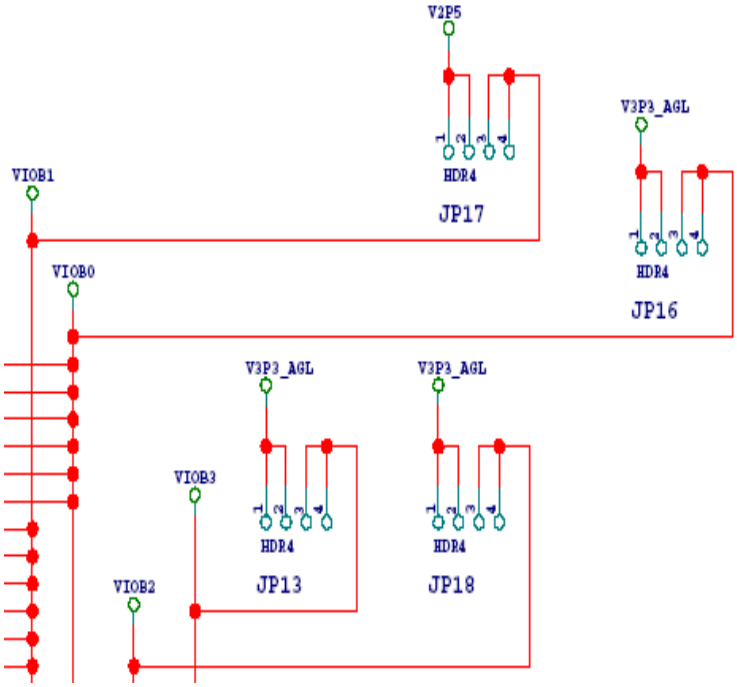


Figure 2.3- M1AGL Development Board Power Configuration

If you are not using the PLL, be sure that jumper JP20 is installed on pins 2 and 3. This will disable power to the PLL.

Be sure to measure current in Flash*Freeze mode also by sliding switch SW3 toward the bottom of the board. You can also use the pushbutton reset switch (SW1) to stop the flip-flops from toggling and measure idle current. See the Sample Design Tutorial section below for more about power modes and measuring current.

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 Sample Design

On the Cortex-M1 Enabled IGLOO Development Kit Install CD, you will find a *Sample 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.

M1AGL Development Board Jumper Descriptions

Jumper	Development Kit Function	Factory Default	Notes
JP1	Provides 3.3V to Prog. USB interface	Installed	Current can be measured at this point.
JP2	Provides 2.5V to FlashPro3 FPGA	Installed	Current can be measured at this point.
JP3	Provides 1.2V and/or 1.5V core voltage to IGLOO	Installed 2-3	Current can be measured at this point.
JP4	Provides 3.3V to FlashPro3 FPGA	Installed	Current can be measured at this point.
JP5	Selects 1.2V and/or 1.5V core voltage for IGLOO FPGA	Depends on whether FPGA is V2 or V5. V2: Installed 2-3 V5: Not installed (auto switch mode)	No jumper installed = 1.5V fixed (use this for V5 parts) Jumper pins 2 to 3 = 1.2V fixed Jumper pins 1 to 2 = 1.5V during configuration, then 1.2V for operation
JP6	Connects 3.3V to pin 2 of P1 connector	Installed	Current can be measured at this point.
JP7	Connects VIN (5V) to pin 1 of P1 connector	Installed	Current can be measured at this point.
JP8	Connects pushbutton reset to P3	Not installed	This functionality is usually not required and can add noise to the reset.
JP9	Connects 3.3V to VPUMP pin on FPGA	Installed 2-3	Current can be measured at this point.
JP10	Connects 2.5V to pin 2 of P2 connector	Installed	Current can be measured at this point.
JP11	Connects RS232_TX signal from FPGA to RXD input of serial-to-USB chip.	Installed	Jumper can be removed so FPGA I/O can be used for another purpose.
JP12	Connects RS232_RX signal from FPGA to TXD input of serial-to-USB chip.	Installed	Jumper can be removed so FPGA I/O can be used for another purpose.
JP13	Connects 3.3V to bank 3 of IGLOO FPGA	Installed 2-3	Current can be measured at this point.
JP14	Connects VIN (5V) to pin 1 of P2 connector	Installed	Current can be measured at this point.
JP15	Provides 3.3V to non-FlashPro3 portion of board	Installed	Current can be measured at this point.
JP16	Connects 3.3V to bank 0 of IGLOO FPGA	Installed 2-3	Current can be measured at this point.
JP17	Connects 2.5V to bank 1 of IGLOO FPGA	Installed 2-3	Current can be measured at this point.
JP18	Connects 3.3V to bank 2 of IGLOO FPGA	Installed 2-3	Current can be measured at this point.
JP19	Connects 3.3V to IGLOO FPGA	Installed 2-3	Current can be measured at this point.
JP20	Supplies voltage to PLL	Installed 1-2 to enable PLL	1-2 connects core voltage to PLL 2-3 shorts VCCPLF to GND to disable PLL and insure it doesn't consume power.
JP21	Selects source of Flash*Freeze pin.	Installed 2-3	1-2 connects GPIOB_0 to FF pin. 2-3 connects pushbutton circuit with RC & Schmitt trigger buffer.
JP22	Selects input power (5V) to the main board logic	Installed 1-4	Factory installed between pins 1 & 4 to select power from 2.1mm external power supply connector. Other jumper positions have been removed and are no longer supported.

Jumper	Development Kit Function	Factory Default	Notes
JP23	Connects VIN (5V) to pin 1 of P5 connector	Installed	Current can be measured at this point.
JP24	Connects 3.3V to pin 2 of P5 connector	Installed	Current can be measured at this point.

Table 2.1- M1AGL 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 – Ground Plane
- Layer 3 – Signal & Ground Ref. for LVDS
- Layer 4 – Signal Layer – LVDS, etc.
- Layer 5 – Ground Plane, Power Plane cutouts
- Layer 6 – Power Plane, LVDS signals
- Layer 7 – Signal & Ground Ref. for LVDS
- Layer 8 – Signal & Local Powers & Grounds
- Layer 9 – Ground Plane
- Layer 10 – Bottom Signal Layer

Installation and Setup

This chapter outlines how to set up the Cortex-M1 Enabled IGLOO Development Kit using the Install CD and the Libero DVD. This chapter also describes the initial M1AGL Development Board configuration and Power sequence.

Installing Libero IDE v8.x

Place the Libero DVD in the DVD Drive on your Personal Computer or Laptop. The DVD 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.x 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.x*.

Note: Libero IDE , CoreConsole and SoftConsole tools will be used in for the Sample Design Tutorial in Chapter 4.

Installing Cortex-M1 Enabled Development Kit using the "Install CD".

Place the Development Kit "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_M1AGL folder (default) or the user selected folder. Figure 3.1 shows the installed directory structure

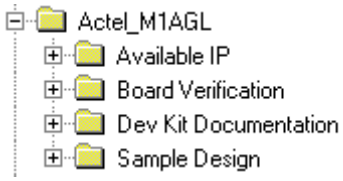


Figure 3.1 – Installed directory structure

The "Install" application will also install the SFE USB to RS232 Controller driver set for the Serial USB connector J2 (SERIAL). The drivers are copied to: C:\Program Files\SparkFunElectronics\USB (default location).

Note: The FlashPro3 USB drivers for the J1 USB (PROG) interface are located in the Libero IDE Installation location.

The Install CD contains the following documention:

- This Cortex-M1 IGLOO Enabled Development Kit User Guide.
- Cortex-M1 IGLOO Enabled Development Kit Quick Start Guide.
- M1AGL Board Schematics.
- Available IP marketing briefs from Actel and also 3rd Party IP Vendors..

Initial M1AGL Development Board configuration

Before powering up the M1AGL 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.

JP1, JP2, JP3(2-3), JP4, JP6, JP7, JP9(2-3), JP10, JP11,JP12, JP13(2-3), JP14, JP15, JP16(2-3), JP17(2-3), JP18(2-3), JP19(2-3), JP20(1-2), JP21(2-3), JP22(1-4) JP23, JP24 are installed.

All others are not installed.

Powering up the M1AGL Development Board

Apply power to the board by connecting one end of the 5-Volt power supply to the J3 connector on the M1AGL board and the other end to a power outlet.

Next, connect one end of a supplied USB cable to a USB port (connector) on your PC or Laptop. Connect the other end to M1AGL connector J1 (PROG). 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.

Note: 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 M1AGL Development Board port J2 (SERIAL). You should see the LED closest to the J2 connector illuminate. This USB port is used to transport a serial RS-232 interface over USB 2.0. The PC will recognize this interface as a COM port.

The PC will detect that new hardware is installed. The "add new hardware wizard" will take care of the USB driver installation. The wizard may ask for a location for the drivers. The drivers are located at "<M1AGL Install Location>\Sample Design\USB_Drivers".

Determining the Serial COM Port

Verify the PC COM port that was enumerated with the RS-232 USB chip at J2.

1. 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 3.2.

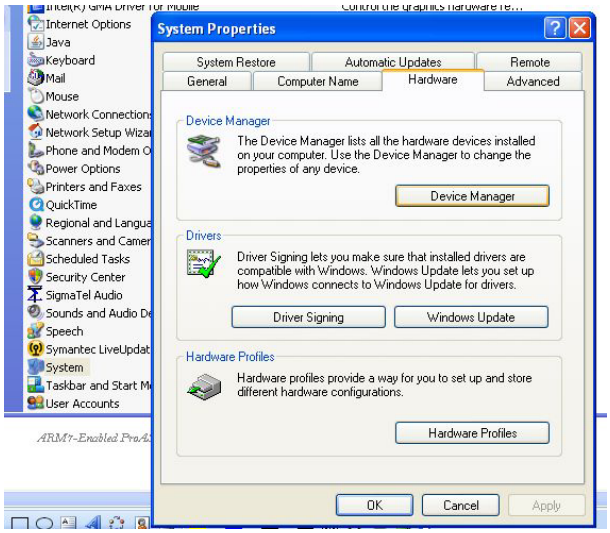


Figure 3.2- Running the Device Manager

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

Figure 3.3- Finding the COM Port

3. Figure 3.3 shows the COM port associated with the "**SFE USB to RS232 Controller**". This is the COM port associated with the J2 USB (Serial) Connector.

Note: Concerning the SFE USB to RS232 Controller, if the board is powered down while any PC application is using this resource, then the PC application will need to close the COM port and reopen the COM port after the board has been powered up again. In general, adhere to the following sequence: close COM port, power down board, power up board, open COM port. Otherwise, the communication channel will not function if the COM port is not restarted after a board powerdown/powerup sequence.

Factory Configuration

The M1AGL Development Board is shipped from the manufacturer with the Sample Design loaded in the M1AGL FPGA. Also the Traffic Light Controller embedded software image is loaded into external FLASH. *See Chapter 4 – Sample Design Tutorial.*

After powering up the M1AGL Development Board for the first time, the user will be able to see the Traffic Light Sample Design executing by observing the timed sequence of LEDs illuminating on U8.

Sample Design Tutorial

This sample design is created specifically for the Actel Cortex-M1 Enabled IGLOO Development Kit. This tutorial will guide you though the following sections:

Sample System Hardware Overview

Libero Project

- Top Level File
- Additional Source Files
- Constraints Files
- Device Options

CoreConsole Design

- Sample Design Hardware Overview
- Functional Description
- Memory Map

Programming the FPGA on the M1AGL Development Board

SoftConsole Designs

- Importing and compiling the SoftConsole designs.
- Debugging the Traffic Light design example
- Using the Memory Loader Utility to program the on board Flash with the Traffic Light embedded code image
- Rebooting the board to run the Traffic Light software from Flash

Sample System Hardware Overview

The Sample system hardware design was produced using two tools in the standard Actel toolchain: Libero Project Manager and CoreConsole.

The Libero design contains the top-level file, other source files, the CoreConsole subsystem, constraints files, and build scripts. These are used by Libero to compile, synthesize, and place-and-route the sample design. The Libero project used in the sample hardware design is provided for reference in the development kit installation.

The CoreConsole component contains the processor, memory controller(s), UART, Timer, and Interrupt Controller, as well as instantiations of the AHB and APB buses. The CoreConsole component contains the bulk of the functionality and logic of the sample design. The CoreConsole project used in the sample hardware design is provided for reference in the development kit installation.

Note: It is not necessary to synthesize, place-and-route, or generate a programming file for the sample hardware design. Also, it is not necessary to regenerate the CoreConsole subsystem. These files are included in the M1AGL Development Kit installation.

Libero Project

Start Libero Project Manager by selecting Actel Libero IDE 8.x -> Project Manager …from the Start Menu.

Open the Libero project file Example_M1AGL_UJTAG.prj by selecting the Project -> Open Project… menu item and then selecting Example_M1AGL_UJTAG.prj. The project is shown in Figure 4.1.

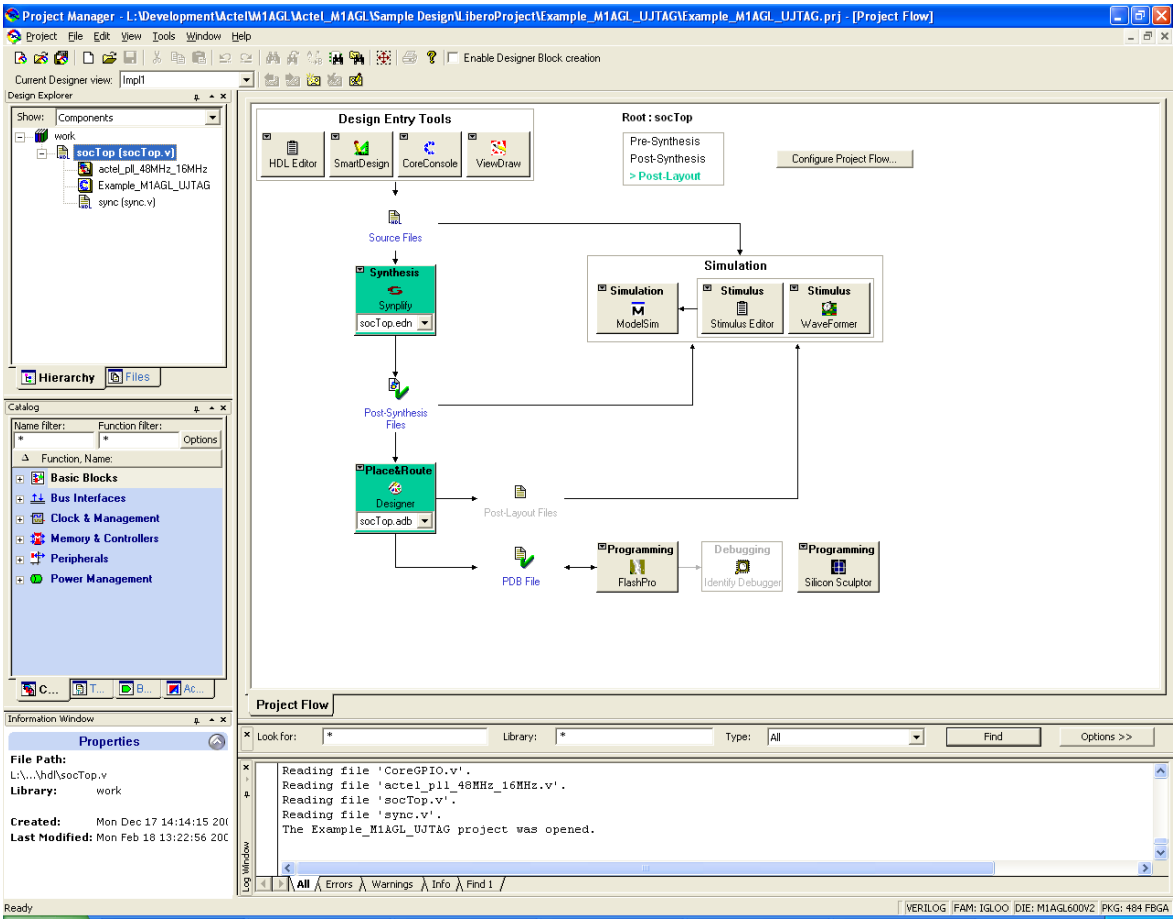


Figure 4.1 - Libero Project

This project file is located in the <install folder>\Sample Design\LiberoProject\Example_M1AGL_UJTAG\ folder.

Top Level Design

The socTop.v design, supplied in the "Sample Designs\Hardware\Source" folder, instantiates the following modules:

- PLL Module (SmartGen component)
- Clock Buffers
- Cortex-M1 Subsystem (CoreConsole component)
- Synchronizing module (for the switch inputs)

The following code is from the socTop Verilog module.

```
////////////////////////////////
//      Cortex-M1  SAMPLE DESIGN
////////////////////////////////

`define N_EXT_ADDR 26
module socTop (
    pbRstN,
    poRstN,

...

  // I/O definitions:

    // System clocks:
    input  sysClk;

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

    // RS232 connections:
    output rs232Atx;                // serial transmit line
    input  rs232Arx;                // serial receive line

  ...


// Instantiations:


    // Instantiate PLL:
    actel_pll_48MHz_16MHz pll1 (.POWERDOWN(poRstNi), .CLKA(sysClk_48MHz), .LOCK(pllLock), .GLA(sysClk_16MHz));
    CLKINT ci0 (.A(sysClk),      .Y(sysClk_48MHz));
    CLKINT ci1 (.A(poRstN),      .Y(poRstNi));
    CLKINT ci2 (.A(pbRstN),      .Y(pbRstNi));


    // Instantiate Cortex-M1 Subsystem:
    Example_M1AGL_UJTAG Example_M1AGL_UJTAG_00(
        // Inputs
        .APBmslave0_PRDATA      (APBmslave0_PRDATA),
        .CoreUARTapbRX          (rs232Arx),

...
    // Instantiate Synchronizers
    sync #(10) sync0
    (
        .outClk      (sysClk_16MHz),
...
```

Additional Source Files

The following is the list of additional Verilog HDL files need for the Libero:

- sync.v

  Switch input synchronizers

sync.v is located in the <install folder>\Sample Design\LiberoProject\Example_M1AGL_UJTAG\HDL folder.

Constraints Files

The following is the list of the Libero Constraint files for the Sample Design:

- constraints.sdc

  Timing constraints for synthesis

- socTop.pdc

  Physical design constraints for place and route (ie IO placement)

These files are located in the <install folder>\Sample Design\LiberoProject\Example_M1AGL_UJTAG\Constraints folder.

Device Options

Select Project -> Settings … from the menu to view the Libero device options as shown in Figure 4.2.

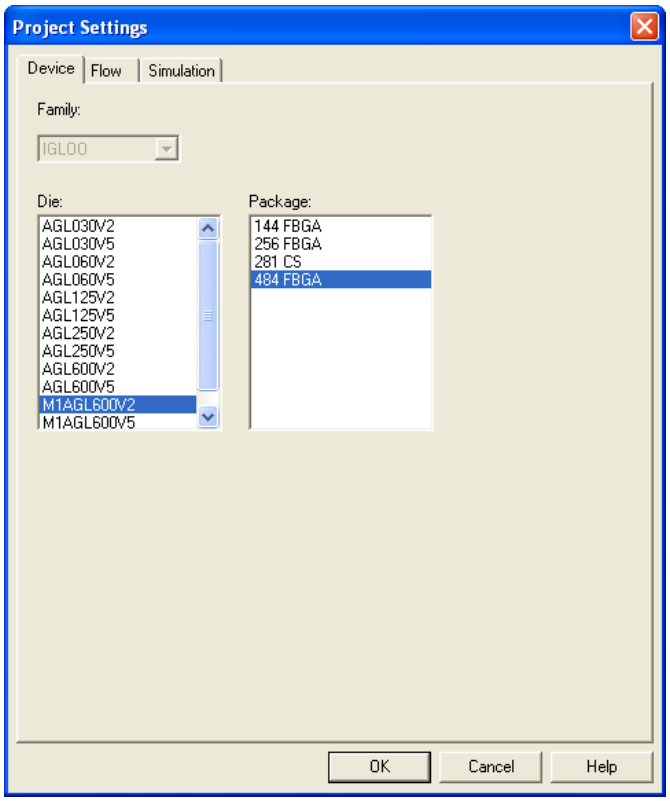


Figure 4.2 - Libero Device Options

CoreConsole Design

To open the CoreConsole portion of the Example_ M1AGL_UJTAG project, expand the top level of the design in Libero Project Manager (Hierarchy Window), right click on Example_ M1AGL_UJTAG, and select “Open Component”. The project is shown in Figure 4.3.

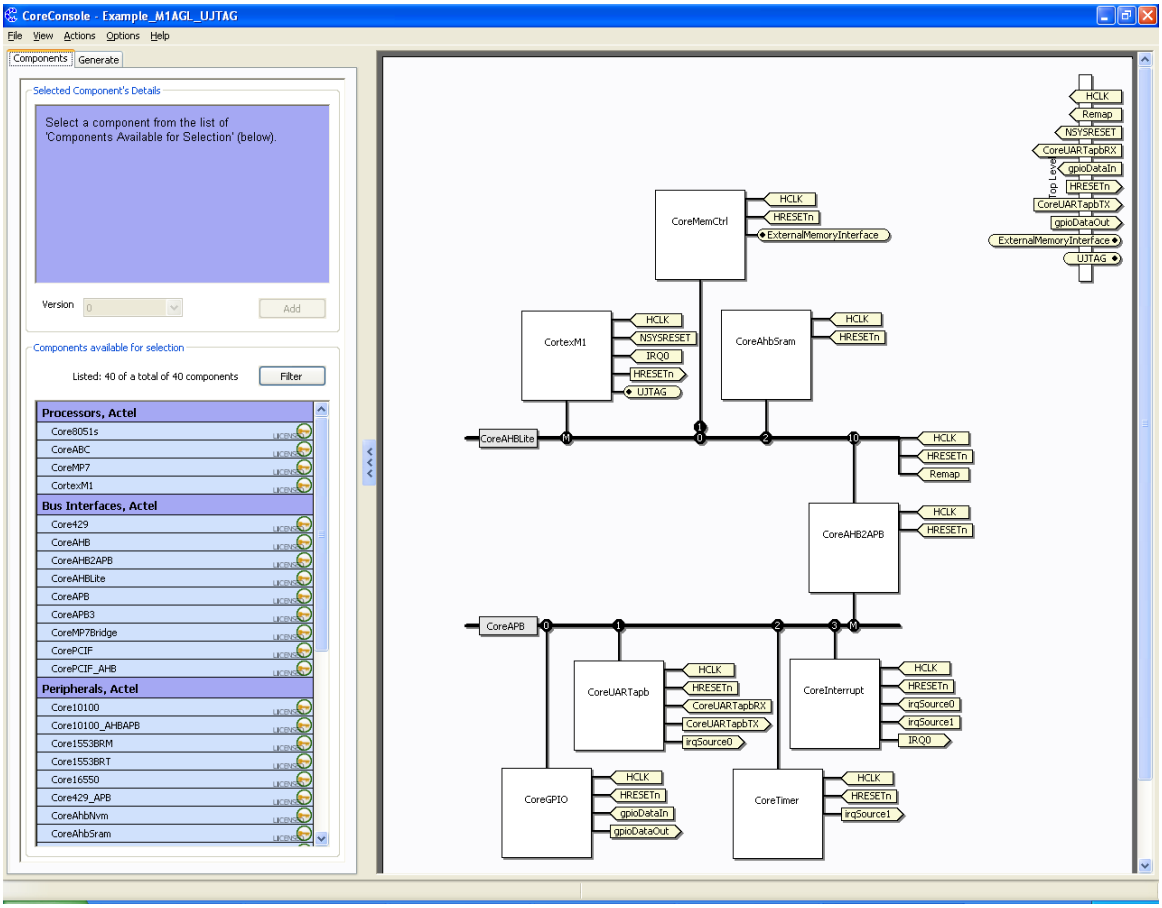


Figure 4.3 - CoreConsole Example_M1AGL_UJTAG Project

Sample Design Hardware Overview

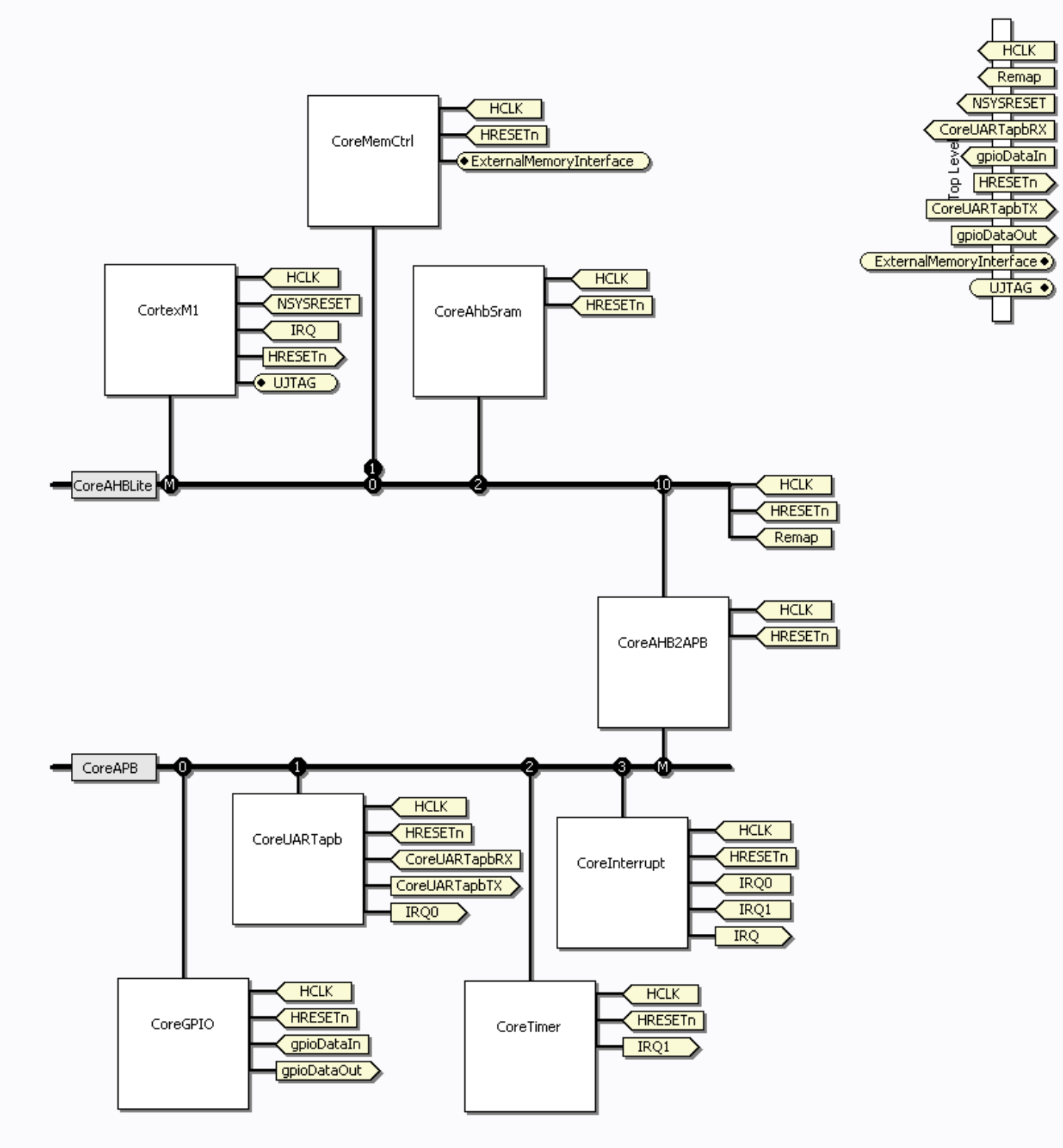


Figure 4.4 - Sample Design Block Diagram

SmartGen Components

The Sample Hardware design contains this SmartGen component:

- actel_pll_48MHz_16MHz (Phase Locked Loop module to convert 48 MHz input clock to 16 MHz)

To explore this module and its options, expand the top level of the design in Libero Project Manager (Hierarchy Window), right click on actel_pll_48MHz_16MHz, and select “Open Component”.

Functional Description

The Sample Hardware design is a basic microcontroller subsystem design. This design contains hardware that is typically used for supporting software, such as Real Time Operating Systems (RTOS) or basic controller software. This design specifically supports a Cortex-M1 processor based system.

The following blocks are contained in the CoreConsole sample design:

- Cortex_M1 (Processor)
- CoreAHB2APB (AHB to APB Bridge)
- CoreMemCtrl (External SRAM and Flash Controller)
- CoreAhbSram (Internal Memory Controller)
- CoreGPIO
- CoreUARTapb
- CoreTimer
- CoreInterrupt
- CoreAHBLite (bus)
- CoreAPB (bus)

The CortexM1 processor can be configured to use from 0 to 32 interrupts. In this example, the CortexM1 is configured to use one interrupt input (IRQ0). There are two interrupt sources in the design that share the same CortexM1 interrupt via the CoreInterrupt (interrupt controller module). The two interrupt sources are the "Receive Ready" interrupt from the CoreUARTapb peripheral, and the "Timer Expired" interrupt from the CoreTimer peripheral.

The Sample subsystem interfaces to both internal SRAM memory and external SRAM and Flash memory. The internal SRAM is connected to the CoreAHBLite bus through the CoreAhbSram internal memory controller. The external (off chip) SRAM and Flash are also connected to the CoreAHBLite bus through the CoreMemCtrl module.

The slower peripherals, CoreUARTapb, CoreTimer, CoreInterrupt, and CoreGPIO modules, are connected to the CoreAPB bus. The CoreAPB bus is connected to the CoreAHBLite bus through the CoreAHB2APB bridge.

Memory Map

The memory map is largely determined by what slot on the AHB/APB bus a particular module occupies. A Remap switch toggles the mapping of SRAM and FLASH at AHB slots 0 and 1.

The following memory locations are determined by the position of switch SW2 as follows:

With SW2 #9 in the ON position:

- External FLASH 0x00000000
- External SRAM 0x10000000

With SW2 #9 in the OFF position:

- External SRAM 0x00000000
- External FLASH 0x10000000

The rest of the memory map is fixed:

- Internal SSRAM 0x20000000
- CoreGPIO 0xA0000000
- CoreUARTapb 0xA1000000
- CoreTimer 0xA2000000
- CoreInterrupt 0xA3000000

See the CoreConsole project for additional settings and connections.

Programming the FPGA on the M1AGL Development Board

This section describes how to load the FPGA with the sample hardware design. It is necessary for the FPGA to be programmed with the Sample Design load before running the example software described later in the SoftConsole Designs section.

Note: The M1AGL Development Board is shipped from the manufacturer with the Sample Design loaded in the M1AGL FPGA.

Apply power to the board by connecting one end of the external 5-Volt power supply to the J3 connector on the M1AGL board and the other end to a power outlet.

Next, connect one end of a supplied USB cable to a USB port (connector) on your PC or Laptop. Connect the other end to M1AGL connector J1 (PROG). After a few seconds, you should see the big yellow "ON" LED at the top right of the board illuminate.

Make sure that switch **SW2** is set for switches 0-8 in the ON position, and switch 9 in the OFF position. This places RAM at address 0 for the debug session.

Follow the FlashPro User Guide to program the M1AGL600. The socTop.stp STPL file is in the <install folder>\Sample Design\LiberoProject\Example_M1AGL_UJTAG\designer\impl1 folder.

Run the FlashPro Programmer from the Start Menu under Actel Libero IDE v8.x -> FlashPro v6.x -> FlashPro v6.x. Use the FlashPro window to program the FPGA as shown in Figure 4.5.

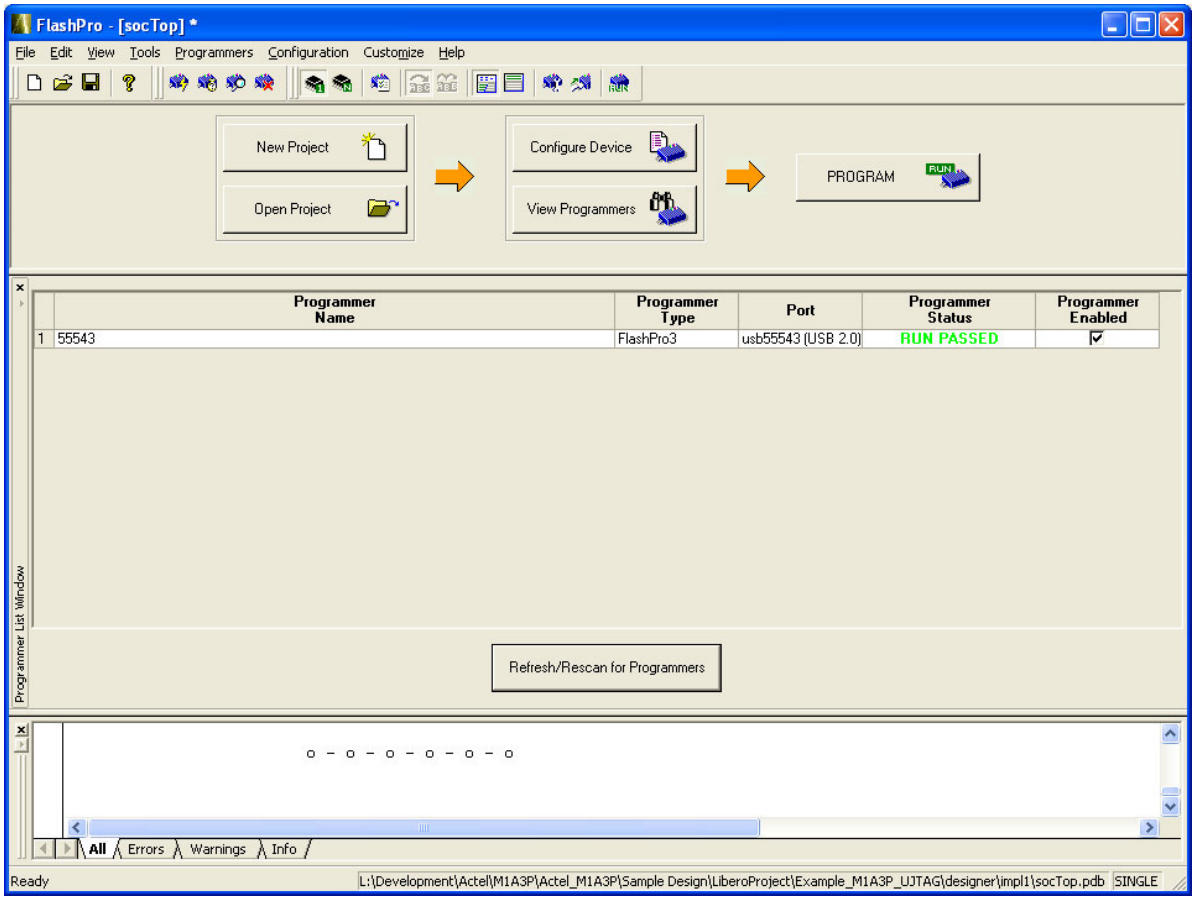


Figure 4.5 –FlashPro3 Programmer Application

SoftConsole Designs

There are two SoftConsole embedded software designs provided with the development kit: a simple traffic light example and a memory loader utility for programming on-board Flash.

The traffic light example will demonstrate the Cortex-M1 system executing a traffic light sequence to blink LEDs on U8. The traffic light example will also transmit control characters through the UART (Com Port) to an optional Traffic Light **Application** running on a PC. While this design is rather simple, it will demonstrate the basics of creating a Cortex-M1 based system FPGA design using CoreConsole and the embedded software to run the on the hardware using SoftConsole.

The memory loader utility provides a convenient method for programming the on-board Flash.

For the traffic light example, the Cortex-M1 software will periodically write data to the CoreGPIO via the CoreAHB2APB module to the APB bus. The periodic data transactions will blink the LEDs which are tied directly to the CoreGPIO.

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

Importing and compiling the SoftConsole designs

Start SoftConsole by selecting Actel SoftConsole v2.x -> Actel SoftConsole… from the Start Menu. Browse to your SoftConsole "workspace" folder as shown in Figure 4.6. Click OK.

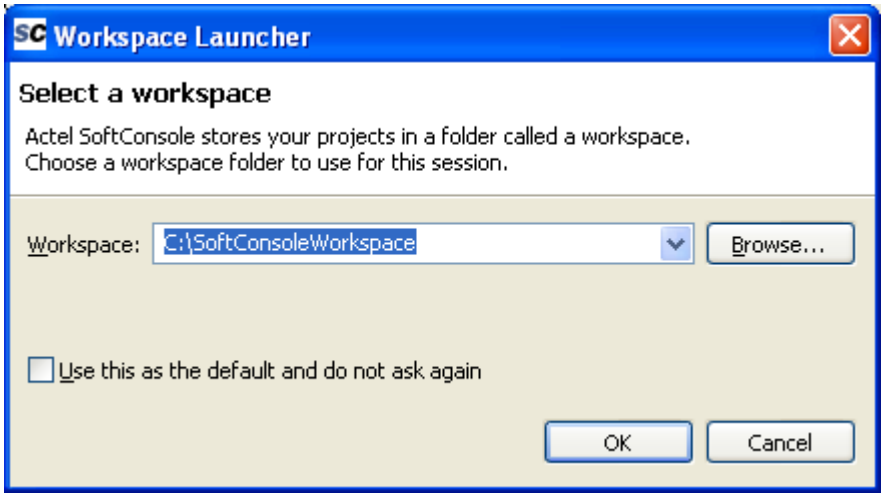


Figure 4.6 - SoftConsole Workspace Launcher

Now import the Traffic Light Example and the Memory Loader Utility project into SoftConsole by selecting File -> Import… item from the menu.

Figure 4.7 shows the resulting window.

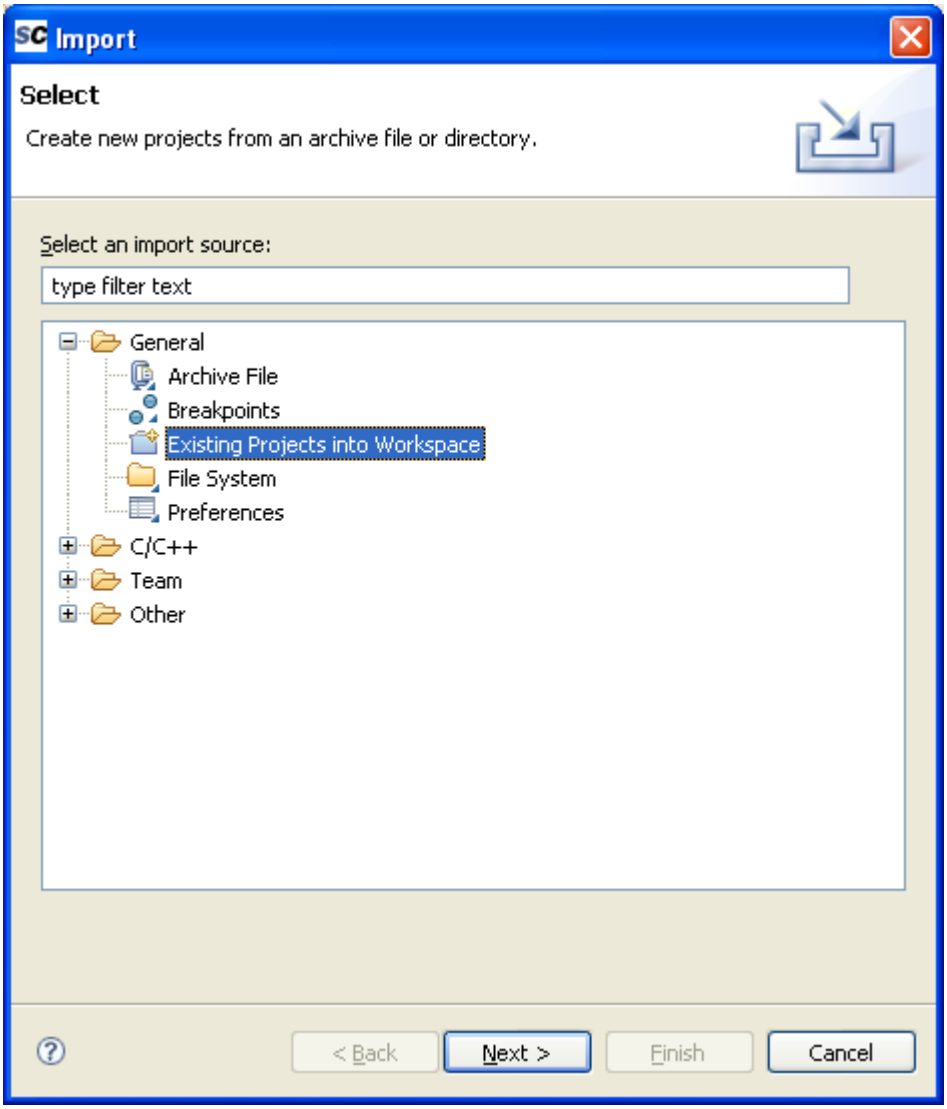


Figure 4.7 - SoftConsole Import Window

Select “Existing Projects into Workspace” in the SoftConsole Import Window and then click “Next”.

Figure 4.8 shows the next window.

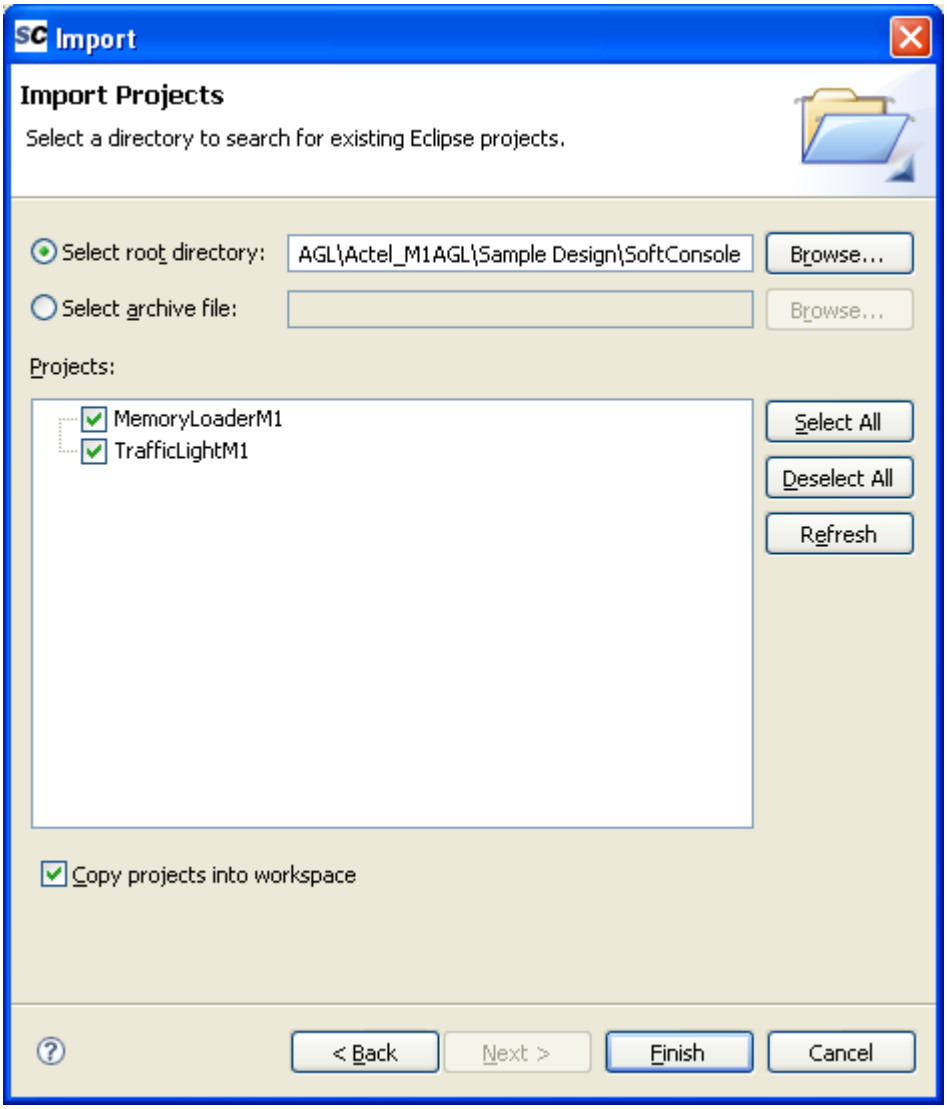


Figure 4.8 - Next SoftConsole Import Window

Click the “Select root directory” radio button and browse to the <install folder> \Sample Design\SoftConsole folder.

Make sure the MemoryLoaderM1 and TrafficLightM1 checkboxes are checked.

Click “Copy projects into workspace” checkbox and then click “Finish”.

By clicking Finish, the projects will be copied into the “Workspace” and then compiled automatically.

Figure 4.9 shows the SoftConsole “Workspace” after these projects have be imported and compiled.

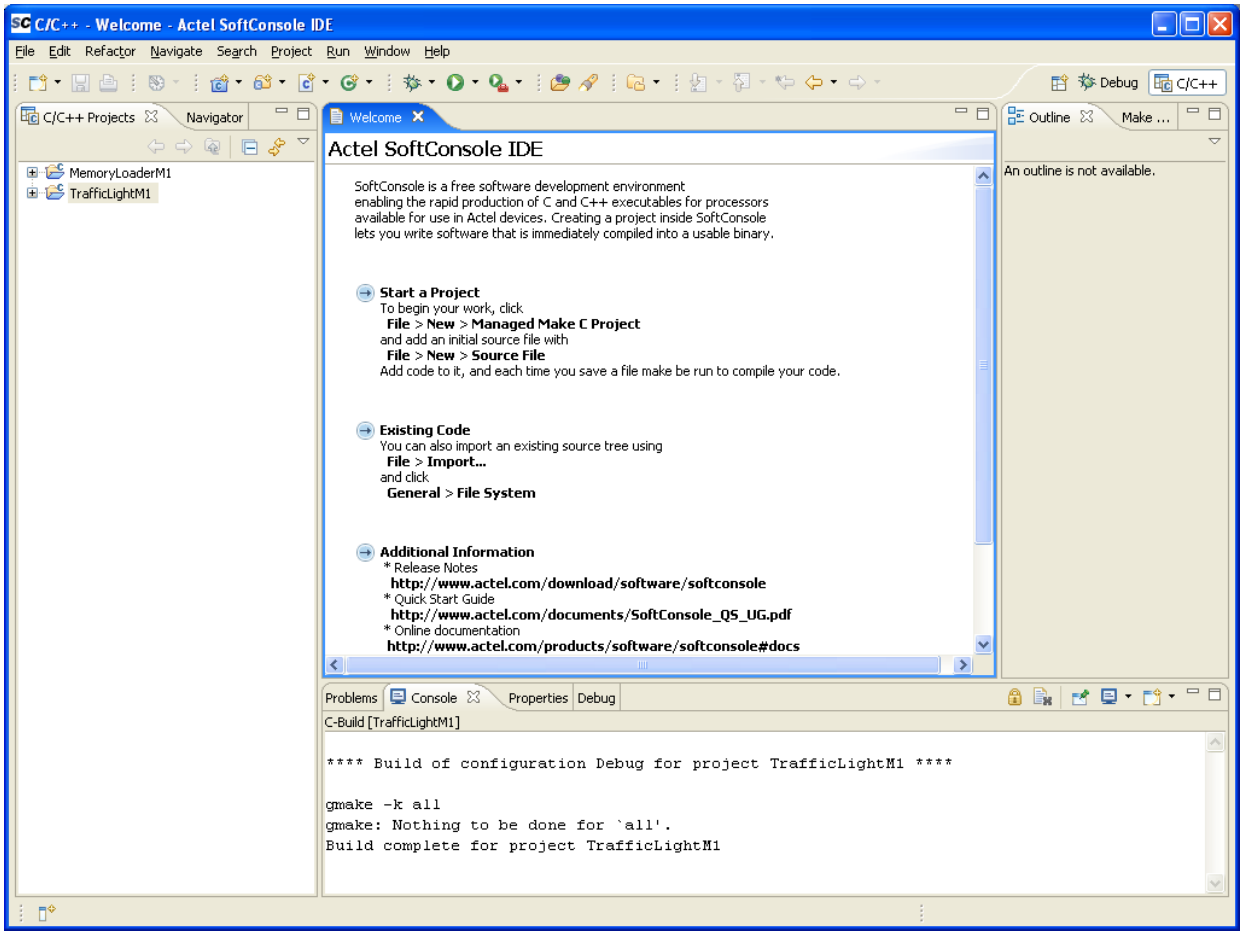


Figure 4.9 - SoftConsole "Workspace" with new projects added

Debugging the Traffic Light design example

Set up the Cortex-M1 Companion Debugger by selecting Run->External Tools->External Tools… from the menu. Create a new configuration and fill in the fields on the External Tools (Main tab) as shown in Figure 4.10.

Name: FlashPro3 On-Chip Debugger for Cortex-M1

Location: <SoftConsole install folder>\Sourcery-G++\bin\arm-none-eabi-sprite.exe

Working Directory: <SoftConsole install folder>\Sourcery-G++\bin

Arguments: -m -l :3000 flashpro: coremp7-cm1

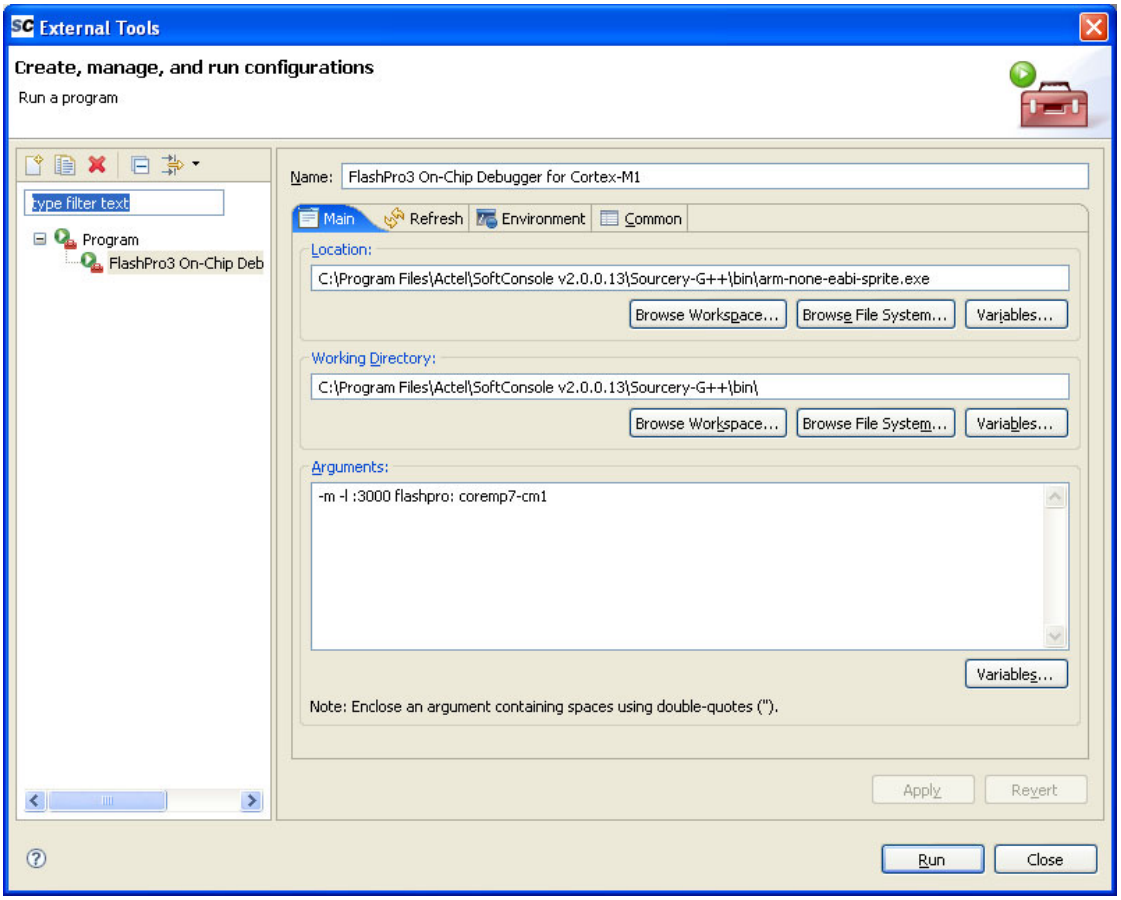


Figure 4.10 - Companion Debugger External Tools Window

Click the "Common" tab and click the "External Tools" checkbox in the "Display in favorites menu" section as shown in Figure 4.11. This creates a shortcut to run the configured "FlashPro3 On-Chip Debugger for Cortex-M1" directly from the Run->External Tools menu.

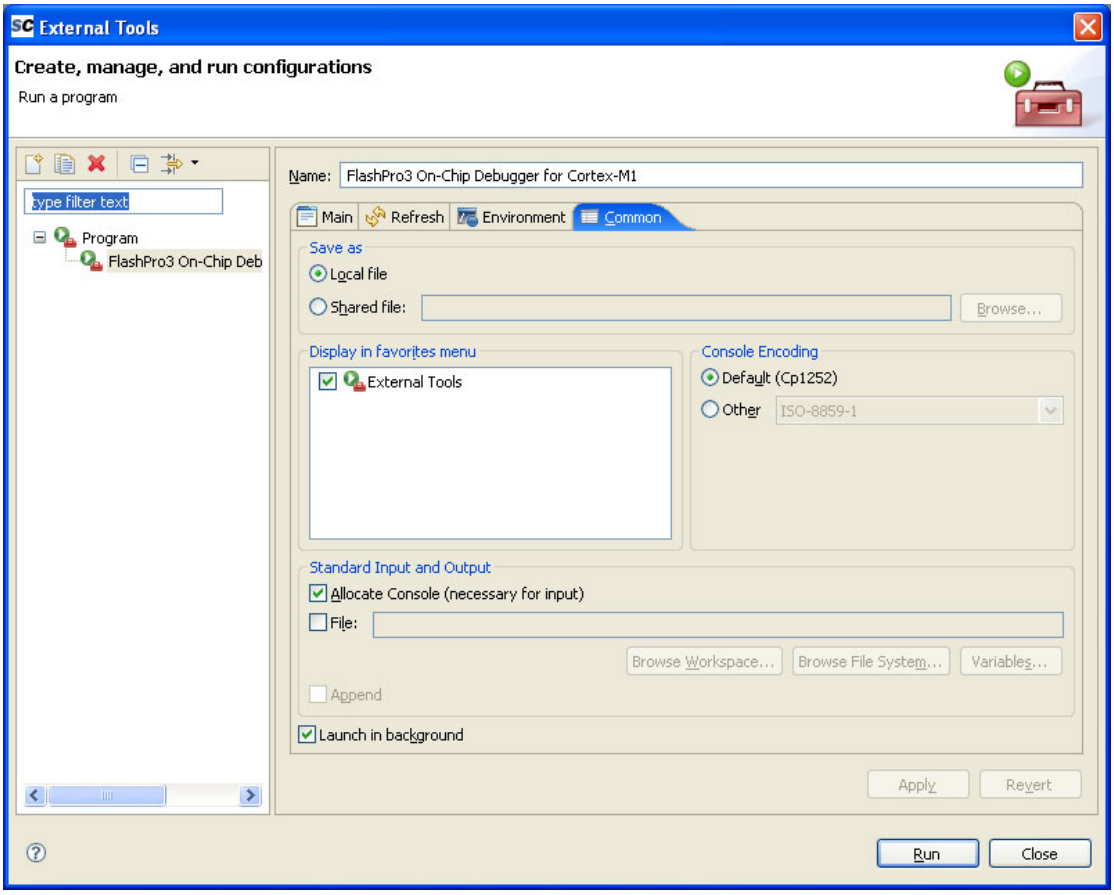


Figure 4.11 – SoftConsole External Tools Window

Click Apply to save these settings, then click Run to Run the FlashPro3 Debugger for the Cortex-M1.

Note: This companion debugger must be launched before starting any debugging session. The companion debugger is available from the Run->External Tools->FlashPro3 Debugger for the Cortex-M1 menu.

Configure the Debug session for the TrafficLightM1 example by selecting Run->Debug… from the menu. Create a new configuration and configure the settings on the "Main" tab as shown in Figure 4.12:

Name: TrafficLightM1

Project: TrafficLightM1C/C++ Application: Debug/TrafficLightM1

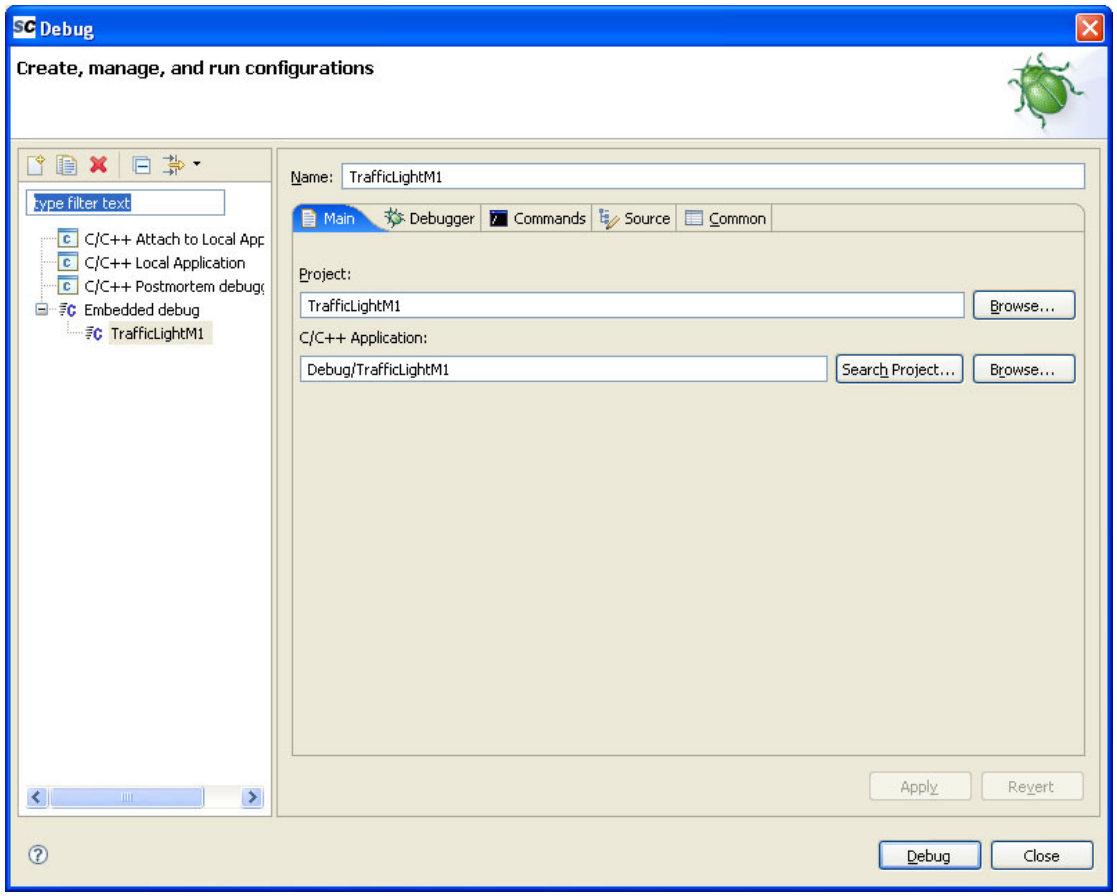


Figure 4.12 SoftConsole Debug Window

Click the “Commands” tab, and configure the settings as shown in Figure 4.13:

‘Initialize’ commands:

target remote :3000

load

‘Run’ commands:

cont

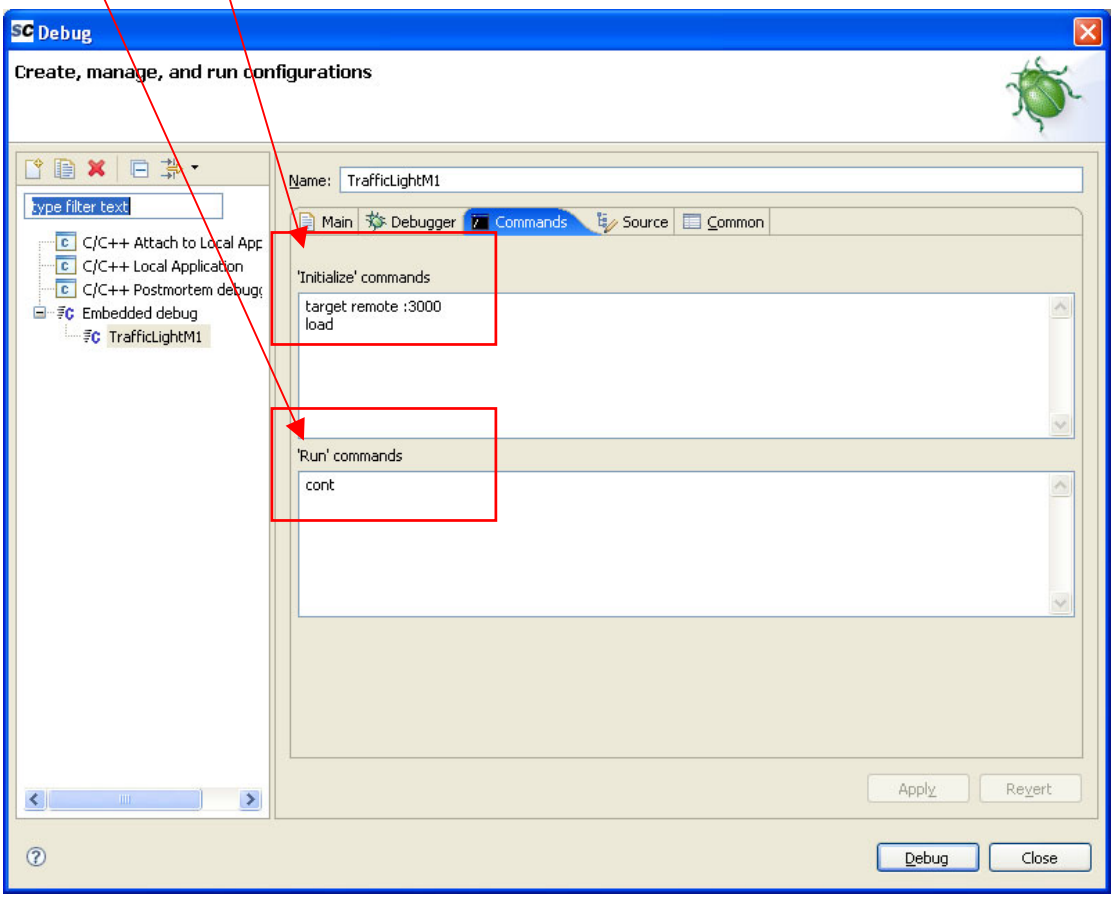


Figure 4.13 – SoftConsole Debug Window – Commands Tab

Click the “Common” tab, and configure the settings as shown in Figure 4.14:

Ensure the “Debug” checkbox in the “Display in favorites menu” section is checked.

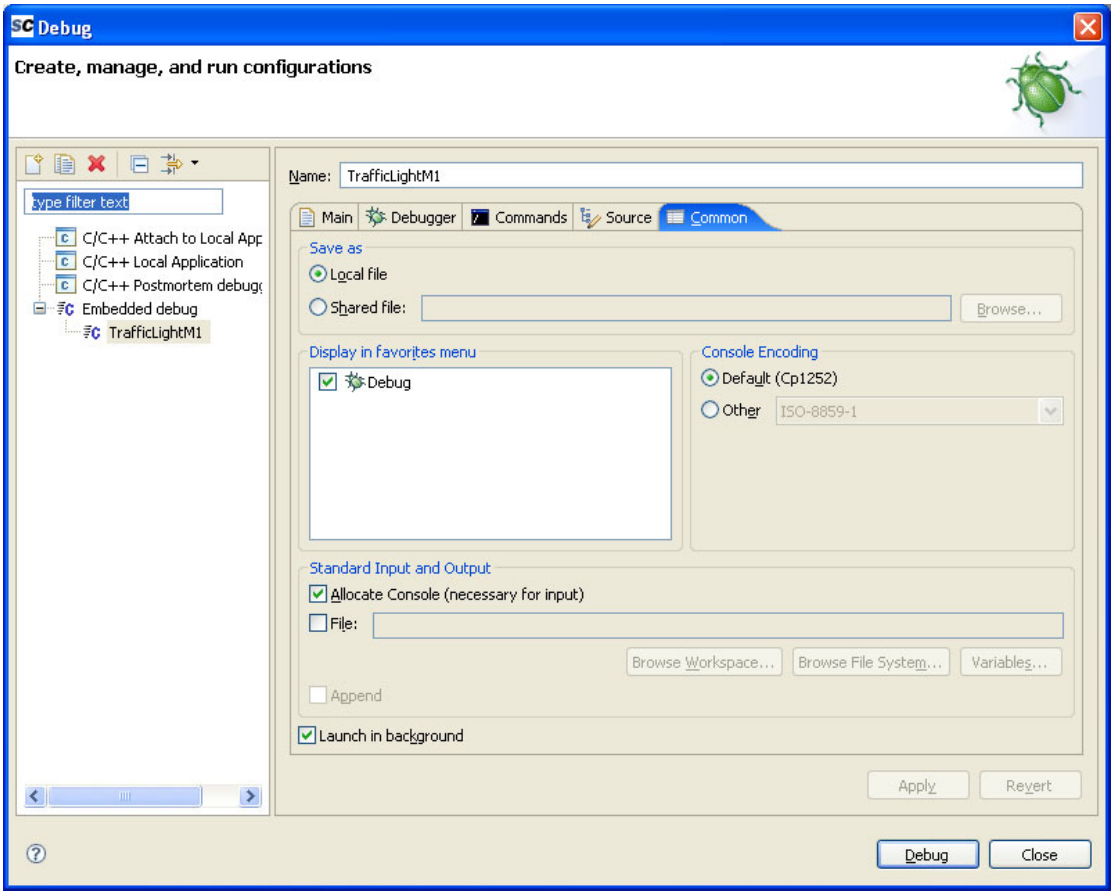


Figure 4.14 – SoftConsole Debug Window – Common Tab

Click Apply to save these settings, then click Debug to start the debugging session for the TrafficLightM1 example. Note that the companion debugger must already be running before starting any debugging session. The TrafficLightM1 debug session is available from the Run->Debug menu.

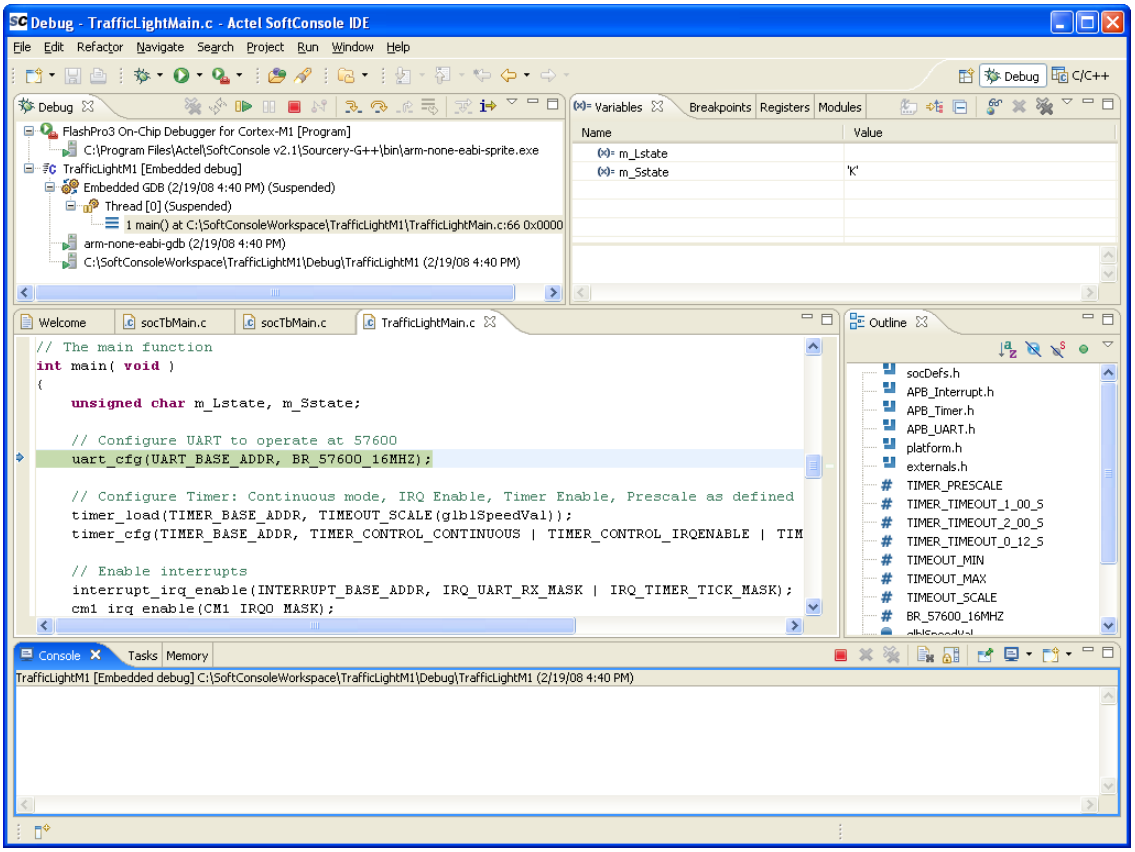


Figure 4.15 – SoftConsole Main Debug Screen

Program execution stops automatically at the beginning of the “main()” function. It is now possible to set breakpoints, single-step through the program or simply run the program.

The TrafficLightM1 example program has a forever loop inside main() function which executes the following:

- sequences the state of the traffic light and
- then outputs the traffic light state to the LEDs and to the UART.

The frequency at which this loop is executed is governed by a timer interrupt.

As an option, run the Traffic Light Application on the PC by running the Traffic_Light.exe application located in the folder <install folder>\ Sample Design\PC_Software\TrafficLight Figure 4.16 shows the PC Traffic Light Application. Ensure that a USB cable is connected from the J2 USB (SERIAL) connector to the PC. Reference the *Determining the Serial COM Port* section in *Chapter 3* in this user guide for determining which COM port to use. Click the “Run” button to connect to the M1AGL board. The PC application will now read the state of the traffic light from the embedded software.

Figure 4.16 – Traffic Light Application

Close the TrafficLight Application and terminate the SoftConsole debugging session

Using the Memory Loader Utility

Ensure that the Cortex-M1 companion debugger is running. The companion debugger is available from the Run->External Tools->FlashPro3 Debugger for the Cortex-M1 menu.

Create a SoftConsole debug session similar to the one created for the TrafficLightM1 Example by selecting Run->Debug… from the menu. Create a new configuration and configure the settings on the "Main" tab as shown in Figure 4.17:

Name: MemoryLoaderM1

Project: MemoryLoaderM1

C/C++ Application: Debug/MemoryLoaderM1

Click the "Commands" tab and configure the settings as follows:

'Initialize' commands:

target remote :3000

load

'Run' commands:

cont

Click the "Common" tab and configure the settings as follows:

Ensure the "Debug" checkbox in the "Display in favorites menu" section is checked.

Click Apply to save these settings, then click Debug to start the debugging session for the MemoryLoader example. Note that the companion debugger must already be running before starting any debugging session. The MemoryLoaderM1 debug session is available from the Run->Debug menu.

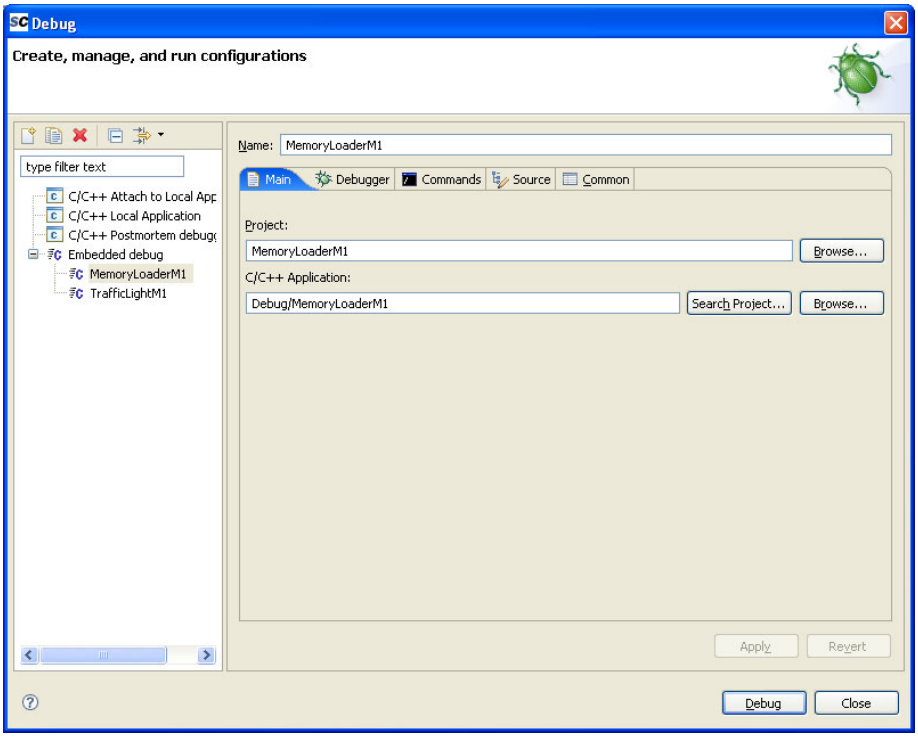


Figure 5-17 – Memory Loader Utility SoftConsole setup

Run the MemoryLoaderM1 embedded application by selecting Run->Resume from the menu. The MemoryLoaderM1 application configures the UART in the example system, then enters a forever loop listening for commands from the MemoryLoader PC application.

Now we are ready to execute the MemoryLoader PC Application by running Memory_Loader.exe from the <install folder>\Sample Design\PC_Software\MemoryLoader folder. Figure 4.18 shows the Memory_Loader PC Application.

Ensure that a USB cable is connected from the J2 USB (SERIAL) connector to the PC. Reference the *Determining the Serial COM Port* section in *Chapter 3* in this user guide for determining which COM port to use. Select the appropriate COM port from the drop-down box, and click the "Connect to Target" button to connect to the M1AGL board. If the connection to the M1AGL board is successful, a status message will appear in the lower left of the MemoryLoader Utility.

In the Load Memory section, set the Base Address to 10000000 (Hex), and click the "Select File to Load" button. Browse to the file TrafficLightM1.bin located in the <SoftConsole workspace folder>\ TrafficLightM1\Release folder.

Note: TrafficLightM1.bin was created by a post-build step for the Release version of the TrafficLightM1 SoftConsole project.

Click the "Load Memory" button to erase and load the external Flash with the TrafficLightM1 image for the Cortex-M1.

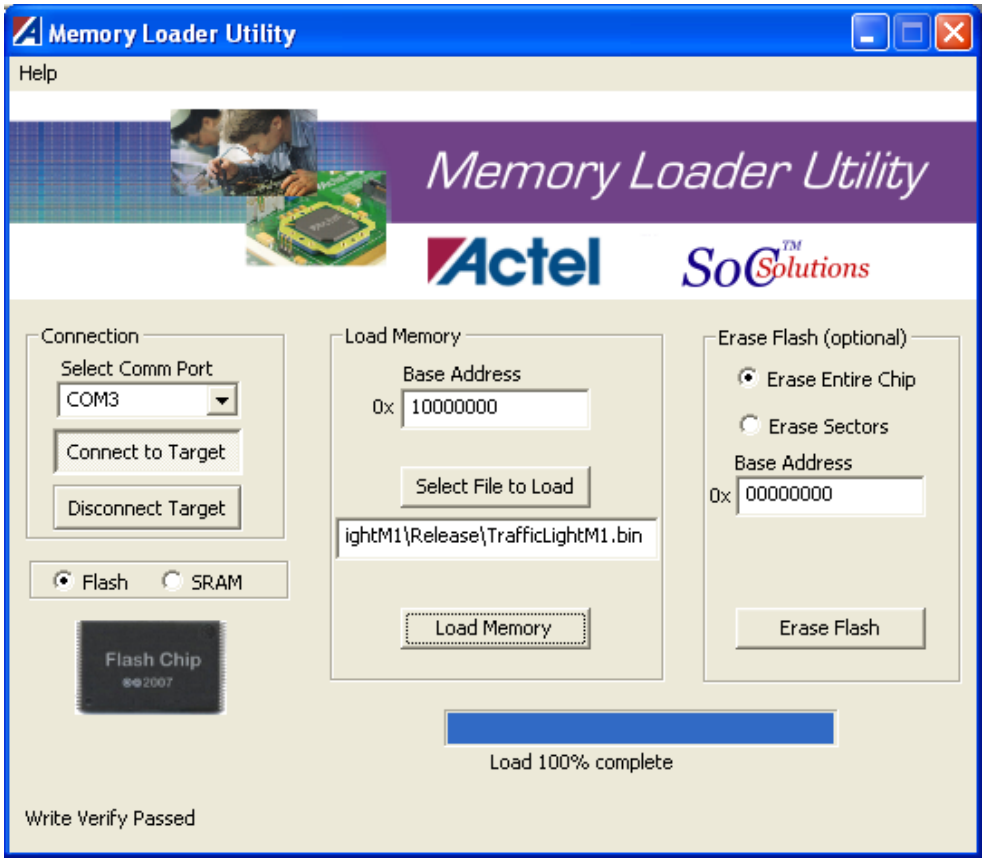


Figure 4.18 – Memory Loader Application

For more information about the Memory Loader Utility for the PC, select Help->Help from the menu.

Close the Memory Loader Utility, terminate the SoftConsole debugging session, and terminate the companion debugger for the Cortex-M1.

Rebooting the board to run the Traffic Light software from Flash

The Release version of the TrafficLightM1 example software has been loaded into external Flash. To run the TrafficLightM1 application, toggle SW2 #9 to the ON position. This remaps external Flash to memory address 0x00000000. Then press and release the system reset button at SW1 to start program execution. Now observe the LEDs at U8 sequence through the traffic light states as in the debug session for the TrafficLightM1 example. Optionally, run the Traffic Light PC application.

Restoring factory default settings

The completion of the procedure documented in this chapter restores factory default settings for the M1AGL FPGA and the on-board FLASH.

FG484 Package for the M1AGL FPGAs

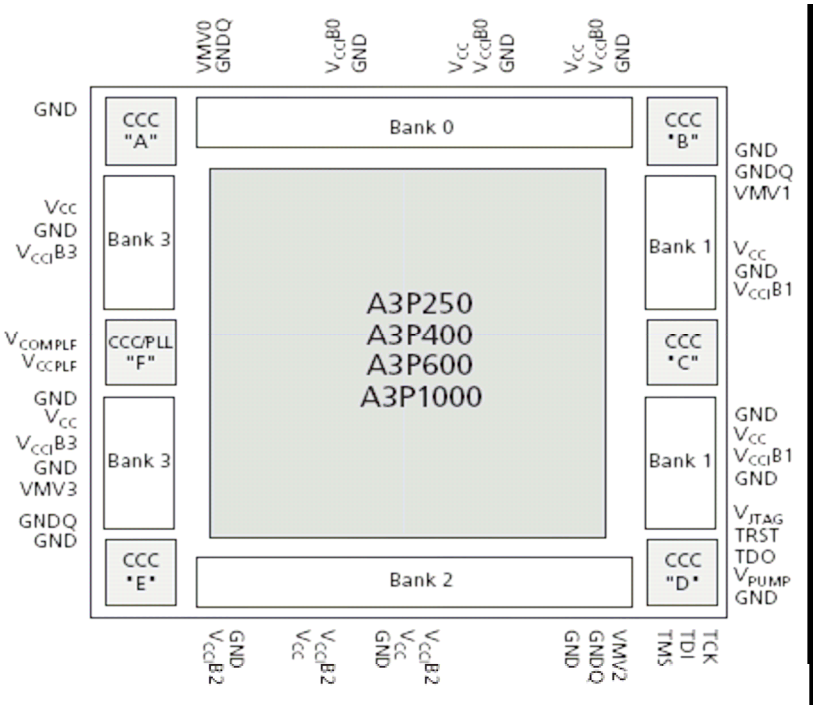


Figure A.1- M1AGL**600** Layout

Figure A.2 shows the bottom view of the 484-Pin FBGA.

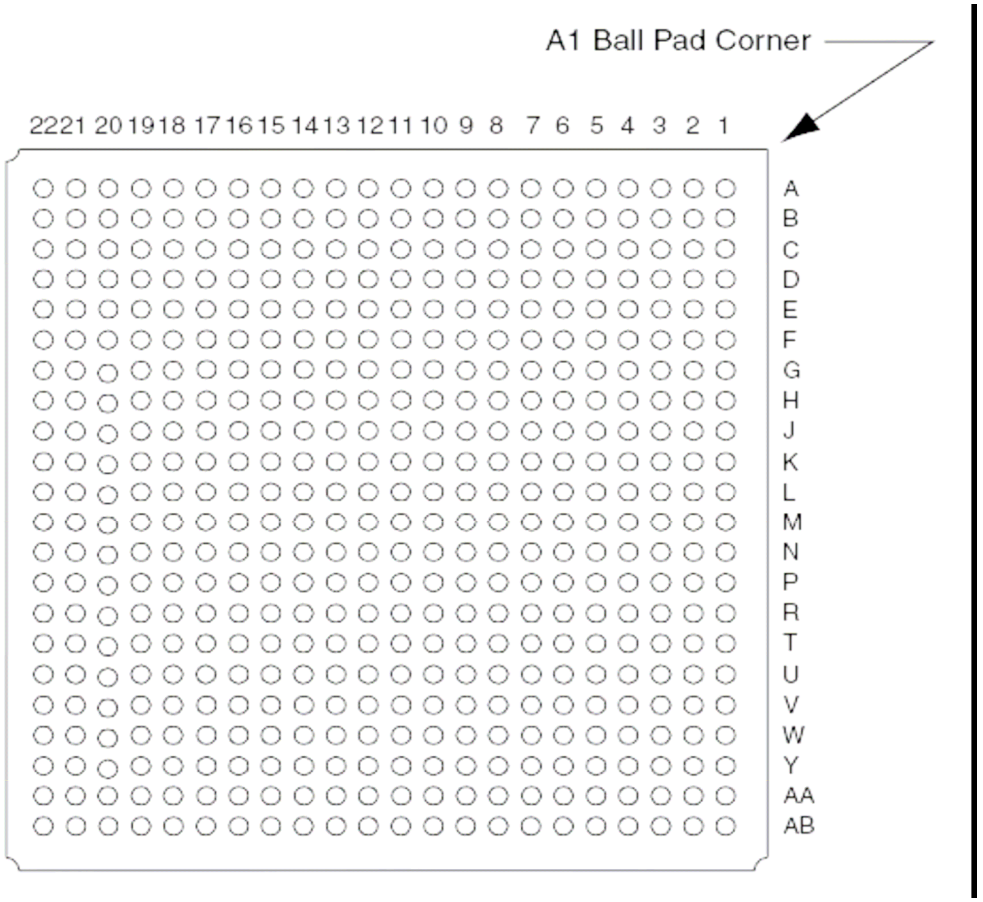


Figure A.2- 484 Pin FBGA Package

Note: For package manufacturing and environmental information, visit the Packaging Solutions page:

http://www.actel.com/products/solutions/package/default.aspx.

Due to the comprehensive and flexible nature of M1AGL 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 M1AGL Development Board signals and the Sample Design signals. Note that the board also allows use of the FG256 BGA package although the FG484 is supplied.

FG256 Ball	FG484 Ball	Pre-loaded Design Signal	Schematic Signal	I / O	Dev. Kit Function
N6	T9	pbRstN	BUF2_PBRST_N	I	Push Button Reset
R4	V7	poRstN	PORESET_N	I	Power on Reset
N4	T7	flashRstN	FLASH_RST_N	O	Reset to Flash Chips
B1	E4	extClk	OSC_CLK	I	System Clock
D6	G9	rs232Atx	RS232_TX	O	UART Transmit
A6	D9	rs232Arx	RS232_RX	I	UART Receive
N3	T6	memAddr[2]	MEM_ADDR2	O	SRAM / FLASH Address
M4	R7	memAddr[3]	MEM_ADDR3	O	SRAM / FLASH Address
C5	F8	memAddr[4]	MEM_ADDR4	O	SRAM / FLASH Address
C6	F9	memAddr[5]	MEM_ADDR5	O	SRAM / FLASH Address
B4	E7	memAddr[6]	MEM_ADDR6	O	SRAM / FLASH Address
A4	D7	memAddr[7]	MEM_ADDR7	O	SRAM / FLASH Address
C2	F5	memAddr[8]	MEM_ADDR8	O	SRAM / FLASH Address
C1	F4	memAddr[9]	MEM_ADDR9	O	SRAM / FLASH Address
D4	G7	memAddr[10]	MEM_ADDR10	O	SRAM / FLASH Address
D2	G5	memAddr[11]	MEM_ADDR11	O	SRAM / FLASH Address
D1	G4	memAddr[12]	MEM_ADDR12	O	SRAM / FLASH Address
E4	H7	memAddr[13]	MEM_ADDR13	O	SRAM / FLASH Address
E3	H6	memAddr[14]	MEM_ADDR14	O	SRAM / FLASH Address
E2	H5	memAddr[15]	MEM_ADDR15	O	SRAM / FLASH Address
F2	J5	memAddr[16]	MEM_ADDR16	O	SRAM / FLASH Address
K1	N4	memAddr[17]	MEM_ADDR17	O	SRAM / FLASH Address
G4	K7	memAddr[18]	MEM_ADDR18	O	SRAM / FLASH Address
H5	L8	memAddr[19]	MEM_ADDR19	O	SRAM / FLASH Address
J5	M8	memAddr[20]	MEM_ADDR20	O	SRAM / FLASH Address
H3	L6	memAddr[21]	MEM_ADDR21	O	SRAM / FLASH Address
H1	L4	memAddr[22]	MEM_ADDR22	O	SRAM / FLASH Address
J1	M4	memAddr[23]	MEM_ADDR23	O	SRAM / FLASH Address
J2	M5	memAddr[24]	MEM_ADDR24	O	SRAM / FLASH Address
M2	R5	memAddr[25]	MEM_ADDR25	O	SRAM / FLASH Address
T8	W11	memData[0]	MEM_DATA0	I / O	SRAM / FLASH Data
P8	U11	memData[1]	MEM_DATA1	I / O	SRAM / FLASH Data
R8	V11	memData[2]	MEM_DATA2	I / O	SRAM / FLASH Data
R7	V10	memData[3]	MEM_DATA3	I / O	SRAM / FLASH Data
T7	W10	memData[4]	MEM_DATA4	I / O	SRAM / FLASH Data
P7	U10	memData[5]	MEM_DATA5	I / O	SRAM / FLASH Data
N8	T11	memData[6]	MEM_DATA6	I / O	SRAM / FLASH Data
T6	W9	memData[7]	MEM_DATA7	I / O	SRAM / FLASH Data
R6	V9	memData[8]	MEM_DATA8	I / O	SRAM / FLASH Data
P6	U9	memData[9]	MEM_DATA9	I / O	SRAM / FLASH Data
N7	T10	memData[10]	MEM_DATA10	I / O	SRAM / FLASH Data
T5	W8	memData[11]	MEM_DATA11	I / O	SRAM / FLASH Data
R5	V8	memData[12]	MEM_DATA12	I / O	SRAM / FLASH Data
B5	E8	memData[13]	MEM_DATA13	I / O	SRAM / FLASH Data

T4	W7	memData[14]	MEM_DATA14	I / O	SRAM / FLASH Data
B6	E9	memData[15]	MEM_DATA15	I / O	SRAM / FLASH Data
N10	T13	memData[16]	MEM_DATA16	I / O	SRAM / FLASH Data
T11	W14	memData[17]	MEM_DATA17	I / O	SRAM / FLASH Data
R11	V14	memData[18]	MEM_DATA18	I / O	SRAM / FLASH Data
P10	U13	memData[19]	MEM_DATA19	I / O	SRAM / FLASH Data
T10	W13	memData[20]	MEM_DATA20	I / O	SRAM / FLASH Data
M9	R12	memData[21]	MEM_DATA21	I / O	SRAM / FLASH Data
P9	U12	memData[22]	MEM_DATA22	I / O	SRAM / FLASH Data
R10	V13	memData[23]	MEM_DATA23	I / O	SRAM / FLASH Data
N9	T12	memData[24]	MEM_DATA24	I / O	SRAM / FLASH Data
T9	W12	memData[25]	MEM_DATA25	I / O	SRAM / FLASH Data
R9	V12	memData[26]	MEM_DATA26	I / O	SRAM / FLASH Data
M8	R11	memData[27]	MEM_DATA27	I / O	SRAM / FLASH Data
A5	D8	memData[28]	MEM_DATA28	I / O	SRAM / FLASH Data
C4	F7	memData[29]	MEM_DATA29	I / O	SRAM / FLASH Data
L3	P6	memData[30]	MEM_DATA30	I / O	SRAM / FLASH Data
M1	R4	memData[31]	MEM_DATA31	I / O	SRAM / FLASH Data
N2	T5	flashHiCeN	FLASH_HCE_N	O	Flash Chip Enable (high chip)
M3	R6	flashLoCeN	FLASH_LCE_N	O	Flash Chip Enable (low chip)
L4	P7	flashWeN	FLASH_WE_N	O	Flash Write Enable
N1	T4	flashOeN	FLASH_OE_N	O	Flash Output Enable
N11	T14	sramCeN	SRAM_CE_N	O	SRAM Chip Enable
T14	W17	sramBsN[0]	SRBS0_N	O	SRAM Byte Select 0
R13	V16	sramBsN[1]	SRBS1_N	O	SRAM Byte Select 1
T12	W15	sramBsN[2]	SRBS2_N	O	SRAM Byte Select 2
T13	W16	sramBsN[3]	SRBS3_N	O	SRAM Byte Select 3
R12	V15	sramWeN	SRAM_WE_N	O	SRAM Write Enable
P11	U14	sramOeN	SRAM_OE_N	O	SRAM Output Enable
B7	E10	ledOut[0]	LED0	O	Drives LED 0
C7	F10	ledOut[1]	LED1	O	Drives LED 1
P5	U8	ledOut[2]	LED2	O	Drives LED 2
T2	W5	ledOut[3]	LED3	O	Drives LED 3
P4	U7	ledOut[4]	LED4	O	Drives LED 4
R3	V6	ledOut[5]	LED5	O	Drives LED 5
P2	U5	ledOut[6]	LED6	O	Drives LED 6
P1	U4	ledOut[7]	LED7	O	Drives LED 7
R1	V4	ledOut[8]	LED8	O	Drives LED 8
R2	V5	ledOut[9]	LED9	O	Drives LED 9
A15	D18	switchIn[0]	SWITCH0	I	Switch Input 0
A14	D17	switchIn[1]	SWITCH1	I	Switch Input 1
B14	E17	switchIn[2]	SWITCH2	I	Switch Input 2
C13	F16	switchIn[3]	SWITCH3	I	Switch Input 3
A12	D15	switchIn[4]	SWITCH4	I	Switch Input 4
D11	G14	switchIn[5]	SWITCH5	I	Switch Input 5
B11	E14	switchIn[6]	SWITCH6	I	Switch Input 6
C11	F14	switchIn[7]	SWITCH7	I	Switch Input 7
D10	G13	switchIn[8]	SWITCH8	I	Switch Input 8
A11	D14	switchIn[9]	SWITCH9	I	Switch Input 9
Unused	Unused	gpio0[9:0]	Unused	I / O	General Purpose IO

N/A	P3	gpio0[10]	IOS_P5 (unused)	I / O	General Purpose IO
N16	T19	gpio0[11]	DIFFB2PRX	I / O	General Purpose IO
P16	U19	gpio0[12]	DIFFB2NRX	I / O	General Purpose IO
J14	M17	gpio0[13]	DIFFA1P	I / O	General Purpose IO
K15	N18	gpio0[14]	DIFFA1N	I / O	General Purpose IO
N/A	K1	gpio0[15]	IOS_P6 (unused)	I / O	General Purpose IO
L15	P18	gpio0[16]	DIFFA2P	I / O	General Purpose IO
L14	P17	gpio0[17]	DIFFA2N	I / O	General Purpose IO
L16	P19	gpio0[18]	DIFFB1P	I / O	General Purpose IO
M16	R19	gpio0[19]	DIFFB1N	I / O	General Purpose IO
Unused	Unused	gpio0[31:20]	Unused	I / O	General Purpose IO
A2	D5	gpio1[0]	GPIOA_0	I / O	General Purpose IO
A3	D6	gpio1[1]	GPIOA_1	I / O	General Purpose IO
A7	D10	gpio1[2]	GPIOA_2	I / O	General Purpose IO
D7	G10	gpio1[3]	GPIOA_3	I / O	General Purpose IO
D8	G11	gpio1[4]	GPIOA_4	I / O	General Purpose IO
B8	E11	gpio1[5]	GPIOA_5	I / O	General Purpose IO
A8	D11	gpio1[6]	GPIOA_6	I / O	General Purpose IO
C8	F11	gpio1[7]	GPIOA_7	I / O	General Purpose IO
E8	H11	gpio1[8]	GPIOA_8	I / O	General Purpose IO
C9	F12	gpio1[9]	GPIOA_9	I / O	General Purpose IO
B9	E12	gpio1[10]	GPIOA_10	I / O	General Purpose IO
A9	D12	gpio1[11]	GPIOA_11	I / O	General Purpose IO
D9	G12	gpio1[12]	GPIOA_12	I / O	General Purpose IO
E9	H12	gpio1[13]	GPIOA_13	I / O	General Purpose IO
C10	F13	gpio1[14]	GPIOA_14	I / O	General Purpose IO
A10	D13	gpio1[15]	GPIOA_15	I / O	General Purpose IO
B10	E13	gpio1[16]	GPIOA_16	I / O	General Purpose IO
B13	E16	gpio1[17]	GPIOA_17	I / O	General Purpose IO
A13	D16	gpio1[18]	GPIOA_18	I / O	General Purpose IO
C12	F15	gpio1[19]	GPIOA_19	I / O	General Purpose IO
B12	E15	gpio1[20]	GPIOA_20	I / O	General Purpose IO
K4	N7	gpio1[21]	GPIOA_21	I / O	General Purpose IO
K2	N5	gpio1[22]	GPIOA_22	I / O	General Purpose IO
J4	M7	gpio1[23]	GPIOA_23	I / O	General Purpose IO
G1	K4	gpio1[24]	GPIOA_24	I / O	General Purpose IO
G2	K5	gpio1[25]	GPIOA_25	I / O	General Purpose IO
G3	K6	gpio1[26]	GPIOA_26	I / O	General Purpose IO
F4	J7	gpio1[27]	GPIOA_27	I / O	General Purpose IO
F3	J6	gpio1[28]	GPIOA_28	I / O	General Purpose IO
F1	J4	gpio1[29]	GPIOA_29	I / O	General Purpose IO
E1	H4	gpio1[30]	GPIOA_30	I / O	General Purpose IO
B2	E5	gpio1[31]	GPIOA_31	I / O	General Purpose IO
E13	H16	gpio2[0]	GPIOB_0	I / O	2.5V IO
C16	F19	gpio2[1]	GPIOB_1	I / O	LVDS IO Transmit Pair with termination
D16	G19	gpio2[2]	GPIOB_2	I / O	
G15	K18	gpio2[3]	GPIOB_3	I / O	LVDS IO Transmit Pair with termination
G16	K19	gpio2[4]	GPIOB_4	I / O	
E16	J18	gpio2[5]	GPIOBRX_5	I / O	LVDS IO Receive Pair with termination
F16	K17	gpio2[6]	GPIOBRX_6	I / O	

B15	E18	gpio2[7]	GPIOB_7	I / O	LVDS IO Transmit Pair with termination
B16	E19	gpio2[8]	GPIOB_8	I / O	
D14	G17	gpio2[9]	GPIOB_9	I / O	LVDS IO Transmit Pair with termination
C15	F18	gpio2[10]	GPIOB_10	I / O	
E14	H17	gpio2[11]	GPIOBRX_11	I / O	LVDS IO Receive Pair with termination
H15	L18	gpio2[12]	GPIOBRX_12	I / O	
E15	H18	gpio2[13]	GPIOB_13	I / O	LVDS IO Transmit Pair with termination
F14	J17	gpio2[14]	GPIOB_14	I / O	
F15	J18	gpio2[15]	GPIOB_15	I / O	LVDS IO Transmit Pair with termination
G14	K17	gpio2[16]	GPIOB_16	I / O	
J15	M18	gpio2[17]	GPIOBRX_17	I / O	LVDS IO Receive Pair with termination
K14	N17	gpio2[18]	GPIOBRX_18	I / O	
G13	K16	gpio2[19]	GPIOB_19	I / O	LVDS IO Transmit Pair with termination
H12	L15	gpio2[20]	GPIOB_20	I / O	
H13	L16	gpio2[21]	GPIOB_21	I / O	LVDS IO Transmit Pair with termination
H16	L19	gpio2[22]	GPIOB_22	I / O	
J13	M16	gpio2[23]	GPIOBRX_23	I / O	LVDS IO Receive Pair with termination
H14	L17	gpio2[24]	GPIOBRX_24	I / O	
J16	M19	gpio2[25]	GPIOB_25	I / O	LVDS IO Transmit Pair with termination
K16	N19	gpio2[26]	GPIOB_26	I / O	
M14	R17	gpio2[27]	GPIOB_27	I / O	LVDS IO Transmit Pair with termination
L13	P16	gpio2[28]	GPIOB_28	I / O	
M15	R18	gpio2[29]	GPIOBRX_29	I / O	LVDS IO Receive Pair with termination
N15	T18	gpio2[30]	GPIOBRX_30	I / O	
F13	J16	gpio2[31]	GPIOB_31	I / O	2.5V IO
N/A	A6	P5_ODD_35_5[0]	GPIOC_1	I / O	General Purpose IO
N/A	A7	P5_ODD_35_5[1]	GPIOC_3	I / O	General Purpose IO
N/A	A10	P5_ODD_35_5[2]	GPIOC_5	I / O	General Purpose IO
N/A	A11	P5_ODD_35_5[3]	GPIOC_7	I / O	General Purpose IO
N/A	B6	P5_ODD_35_5[4]	GPIOC_9	I / O	General Purpose IO
N/A	B7	P5_ODD_35_5[5]	GPIOC_11	I / O	General Purpose IO
N/A	B10	P5_ODD_35_5[6]	GPIOC_13	I / O	General Purpose IO
N/A	AB17	P5_ODD_35_5[7]	GPIOC_15	I / O	General Purpose IO
N/A	AB16	P5_ODD_35_5[8]	GPIOC_17	I / O	General Purpose IO
N/A	AB13	P5_ODD_35_5[9]	GPIOC_19	I / O	General Purpose IO
N/A	AB12	P5_ODD_35_5[10]	GPIOC_21	I / O	General Purpose IO
N/A	AA16	P5_ODD_35_5[11]	GPIOC_23	I / O	General Purpose IO
N/A	Y2	P5_ODD_35_5[12]	GPIOC_25	I / O	General Purpose IO
N/A	U1	P5_ODD_35_5[13]	GPIOC_27	I / O	General Purpose IO
N13	T16	P5_ODD_35_5[14]	GPIOC_29	I / O	General Purpose IO
N/A	M3	P5_ODD_35_5[15]	GPIOC_31	I / O	General Purpose IO
N/A	A12	P5_EVEN_34_4[0]	GPIOC_0	I / O	General Purpose IO
N/A	A13	P5_EVEN_34_4[1]	GPIOC_2	I / O	General Purpose IO
N/A	A16	P5_EVEN_34_4[2]	GPIOC_4	I / O	General Purpose IO
N/A	A17	P5_EVEN_34_4[3]	GPIOC_6	I / O	General Purpose IO
N/A	B13	P5_EVEN_34_4[4]	GPIOC_8	I / O	General Purpose IO

N/A	B16	P5_EVEN_34_4[5]	GPIOC_10	I / O	General Purpose IO
N/A	B17	P5_EVEN_34_4[6]	GPIOC_12	I / O	General Purpose IO
N/A	AB9	P5_EVEN_34_4[7]	GPIOC_14	I / O	General Purpose IO
N/A	AB8	P5_EVEN_34_4[8]	GPIOC_16	I / O	General Purpose IO
N/A	AB7	P5_EVEN_34_4[9]	GPIOC_18	I / O	General Purpose IO
N/A	AB6	P5_EVEN_34_4[10]	GPIOC_20	I / O	General Purpose IO
N/A	AA7	P5_EVEN_34_4[11]	GPIOC_22	I / O	General Purpose IO
N/A	AA6	P5_EVEN_34_4[12]	GPIOC_24	I / O	General Purpose IO
N/A	W2	P5_EVEN_34_4[13]	GPIOC_26	I / O	General Purpose IO
N/A	T1	P5_EVEN_34_4[13]	GPIOC_28	I / O	General Purpose IO
M13	R16	P5_EVEN_34_4[15]	GPIOC_30	I / O	General Purpose IO
N/A	J21	P5_38_37[0]	DIFFC1P	I / O	General Purpose IO
N/A	J22	P5_38_37[1]	DIFFC1N	I / O	General Purpose IO
N/A	K22	P5_40_39[0]	DIFFC2P	I / O	General Purpose IO
N/A	K21	P5_40_39[1]	DIFFC2N	I / O	General Purpose IO

Table A.2- M1AGL FPGA Signals

Board Schematics

For detailed Schematic, please refer to the **"M1AGL Schematics.pdf"** in the **"Dev Kit Documentation"** folder.

There are 7 pages to the Schematics, titled as follows:

1. POWER SUPPLIES
2. PWR, HDRS, LVDS
3. SRAM & FLASH
4. CLK, RESET, USB, ETC.
5. FPGA
6. FPGA
7. FLASHPRO3

The schematic number is SOC-AGL-S-002. Please reference this number and the schematic sheet when contacting support@socsolutions.com for M1AGL Development Board and schematic support.

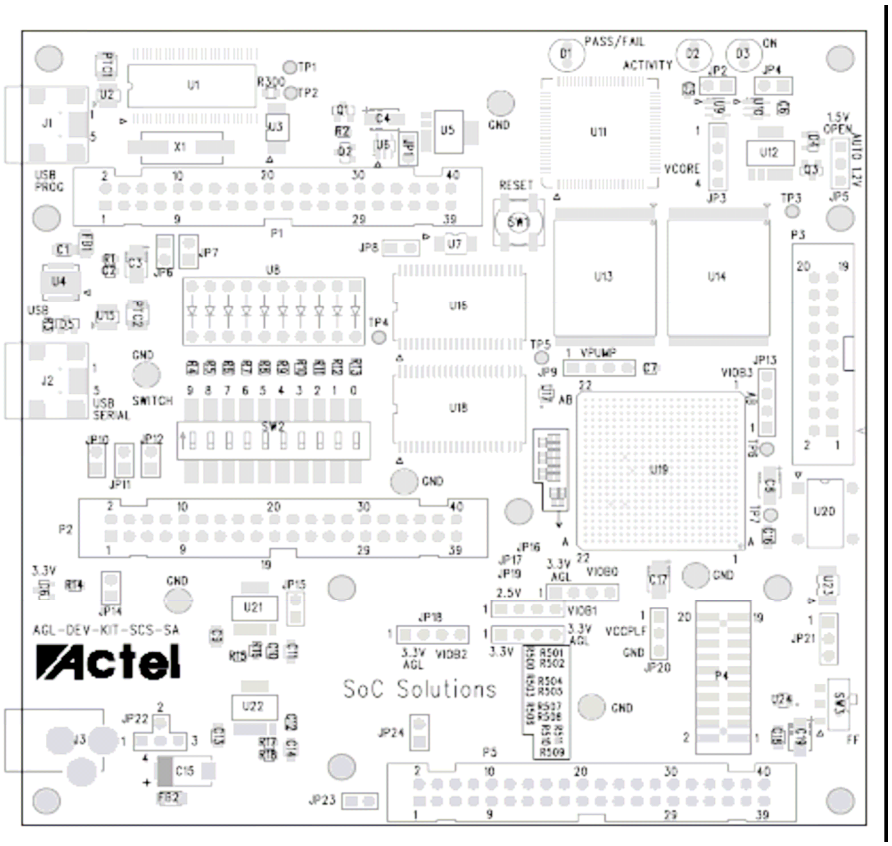


Figure B.1 - M1AGL Development Board Top Assembly Drawing

M1AGL Development Board User Tests

This appendix outlines how to test the M1AGL Development Board.

Introduction

The Cortex-M1 Enabled IGLOO Development Kit is supplied with the design and procedure used for manufacturing tests. The following procedure is a subset of the actual procedure used by the manufacturer to fully test the M1AGL Development Board. This test procedure is useful for verifying the functionality of the board as well as observing some of the Cortex-M1 Enabled IGLOO Development Kit features in action.

Board Configuration

Please refer to *Chapter 3 – Initial M1AGL Development Board Configuration* section for configuring the M1AGL Development Board.

Also refer to the *Chapter 3 – Powering up the M1AGL Development Board* section for powering and cabling the M1AGL Development Board.

Load the "User Tests" design

Use the Actel FlashPro programming tool to program the FPGA with the embedded controller design (**socTop.stp** in the **Board Verification\Hardware** folder). The "User Tests" design architecture is described later in this appendix.

Procedure for running "User Tests"

1. Run the **M1AGL Board – User Tests** by navigating to the "Board Verification\PC_Software\M1AGL_User_Tests" folder and double clicking the M1AGL_Tests.exe file.

2. The first dialog box will be a COM configuration dialog. Reference the *Determining the Serial COM Port* section in *Chapter 3* in this user guide for determining which COM port to use. Select the appropriate COM port and click OK.

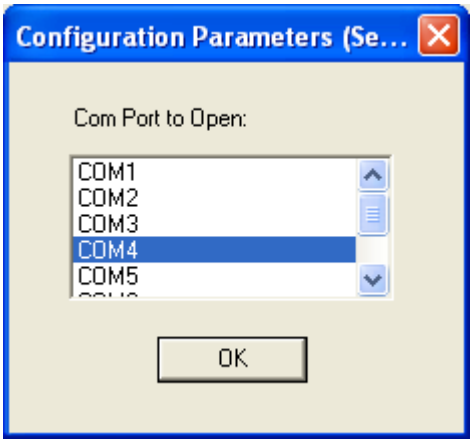


Figure C.1- COM Port Dialog

3. Now run all the tests in the **M1AGL Board – User Tests Application**.

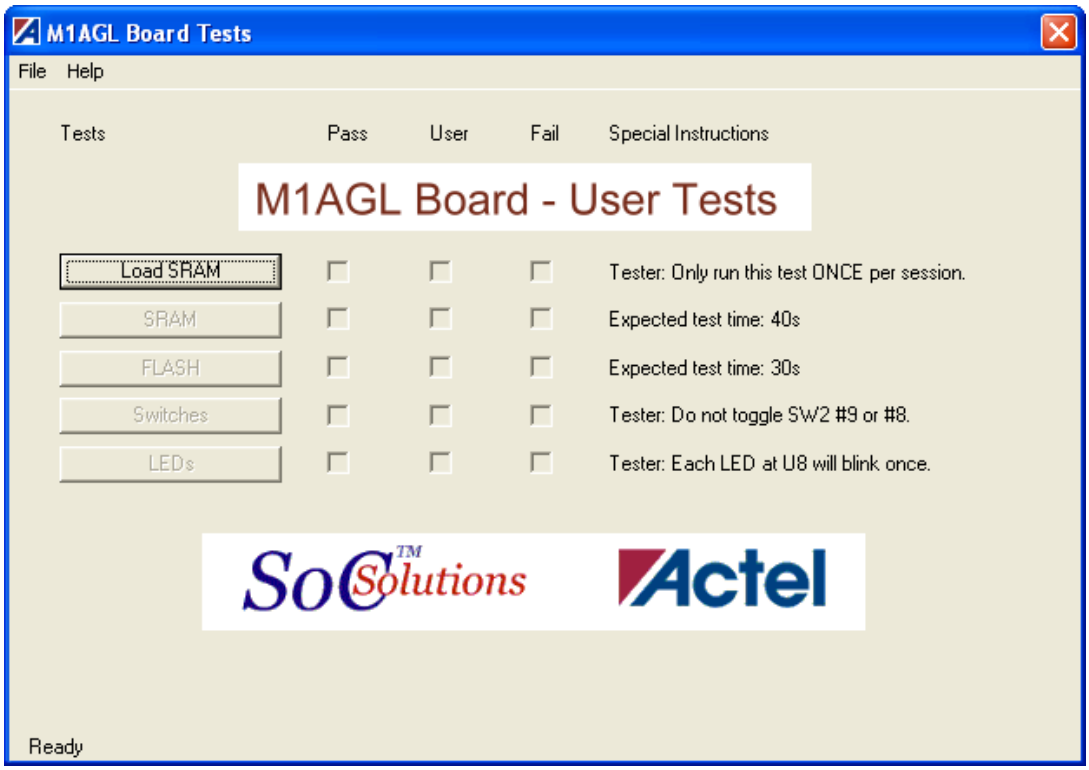


Figure C.2- M1AGL Board – User Tests Application

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 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 **"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 the external power supply from J3.
- Unplug USB cables at J1, J2.
- Verify the jumpers and switches correspond to factory defaults.
- Repeat steps 1-3 above.

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

"User Tests" Design

The "User Test" design is based on the **PIP-EC02,** Pre-integrated IP platform. This design is provided by *SoC Solutions*. For more information, contact sales@socsolutions.com .

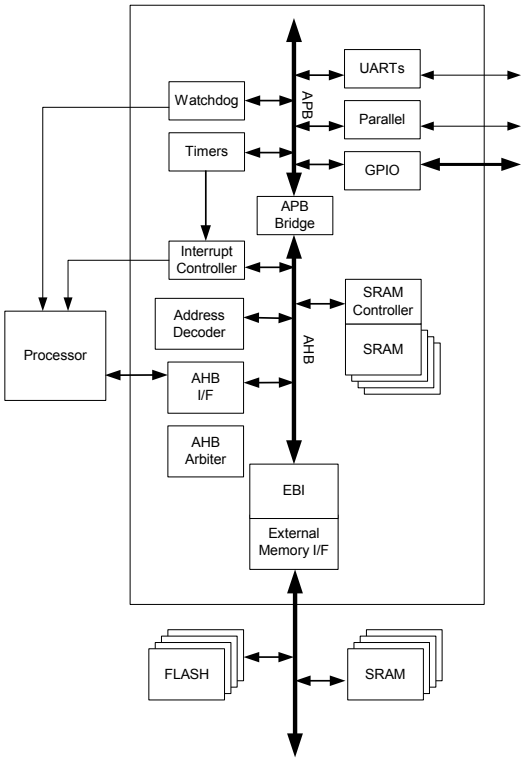


Figure C.3- "User Tests" Design

.

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.

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.