Zedboard Version

Analog Essentials Getting Started Guide

Overview

Maxim *Analog Essentials* are a series of plug-in peripheral modules that allow engineers to quickly test, evaluate, and integrate Maxim components into their hardware/software designs. The modules electrically and physically conform to the Digilent Pmod™ interface specification and are compatible with any Digilent Pmod-compatible header.

The Analog Essentials collection contains 15 peripheral module boards which are supported by prebuilt hardware and software projects for various FPGA development boards which are compatible. This version of the Getting Started Guide is applicable to the Zedboard which is based on the Xilinx Zynq® FPGA. Users of the other FPGA boards should refer to different versions of this guide.

Important notes – please read carefully

1. **ESD Sensitivity**. These modules are intended to be used in an electronic prototyping setting. They are sensitive to static charge and are subject to damage if proper precautions against ESD are not taken. When handling these modules, the user should take the same care and precautions as when working with other ESD sensitive prototypes.

2. **Plug – Connector mating.** Many plug-in modules contain only one row of six pins, while most Pmod connectors contain two rows of six contacts each. When plugging in a module with a single row of pins, *always plug the module into the top row of contacts.*

3. **Plug in with care.** Please exercise care when plugging the modules into the Pmod port. The connectors are not keyed, so it is possible to incorrectly plug in the module and misalign the pins. Although each module includes series resistors to provide current limiting, in some cases it is still possible to damage the module if it is incorrectly plugged in.

Additional components required

Interested users may wish to obtain additional components to evaluate the following modules:

MAX3231MPMB1 – this module requires a 10mm, 3V lithium primary type coin cell. The generic battery type is CR1025. Specific examples of this battery are Panasonic #CR1025 and Energizer #CR1025.

MAX31855PMB1 – this module requires a K-Type thermocouple with subminiature connector. Omega series 5SRTC are suitable (www.omega.com). Lower cost alternatives may be available via eBay.

Analog Essentials Collection modules

The table below lists the peripheral modules included in the collection, along with a description and interface type.

Part	Functionality	Interface
DS1086LPMB1#	I^2C spread-spectrum EconOscillator™	I^2C
DS3231MPMB1#	I^2C real-time clock	I^2C
MAX3232PMB1#	RS-232 transceiver	UART
MAX4824PMB1#	Octal relay driver	GPIO
MAX5216PMB1#	High-performance 16-bit DAC	SPI/GPIO
MAX5487PMB1#	Dual 256-tap digital potentiometer	SPI
MAX5825PMB1#	Octal 12-bit DAC	I^2C
MAX7304PMB1#	16-port GPIO and LED driver	I^2C
MAX9611PMB1#	Current-sense amplifier with op amp and ADC	I^2C
MAX11205PMB1#	16-bit delta-sigma ADC with 2-wire interface	GPIO
MAX14840PMB1#	RS-485 transceiver	UART/GPIO
MAX14850PMB1#	6-channel, 600V galvanic isolator	SPI/UART
MAX31723PMB1#	Digital thermometer	SPI
MAX31855PMB1#	Thermocouple-to-digital converter	SPI
MAX44000PMB1#	Ambient light and proximity sensor	I^2C

Included Files

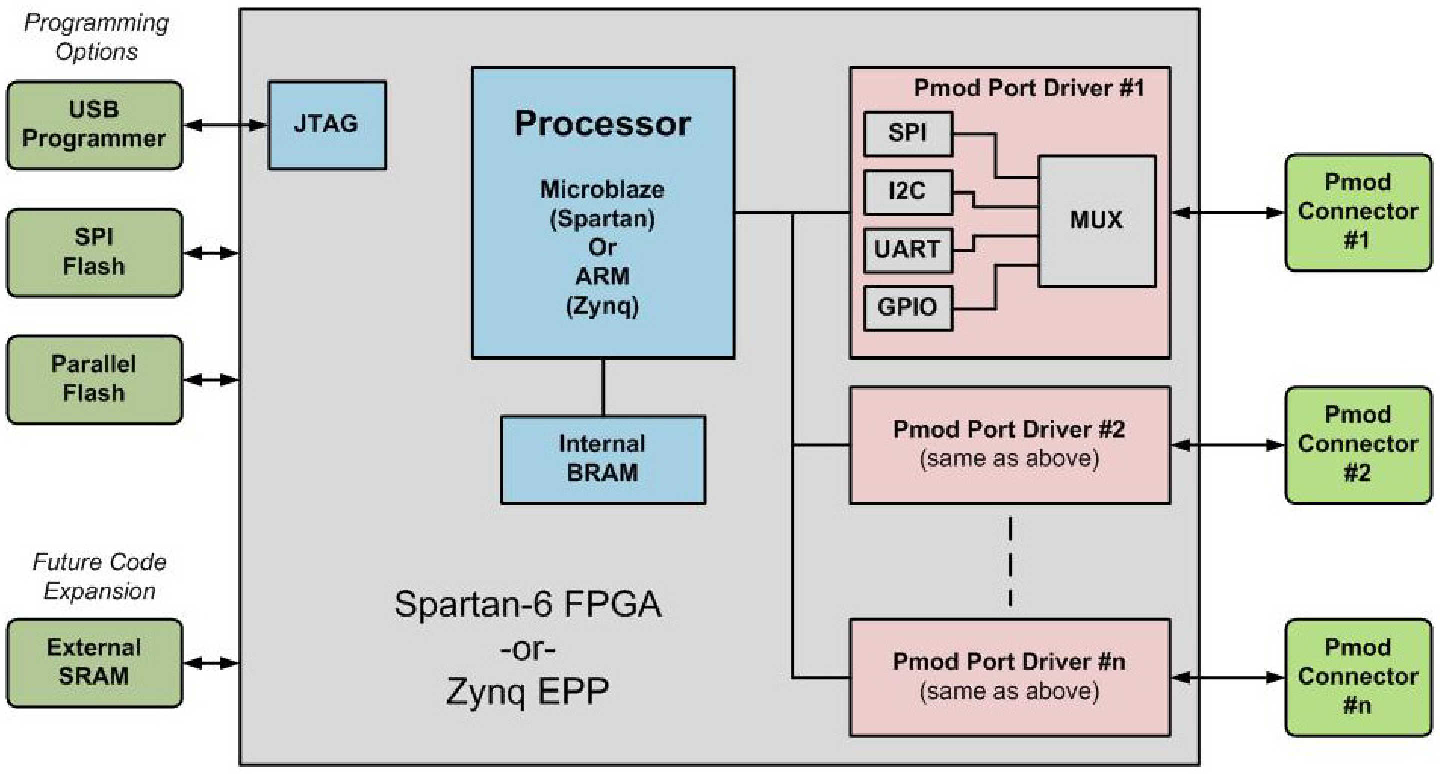
The top level of the hardware design is a Xilinx ISE® Project Navigator Project (.xise) for Xilinx ISE version 14.1 The Verilog-based top.v module provides FPGA/board net connectivity, allows HDL interaction with peripherals, and instantiates the wrapper that carries both the Zynq Processing System and (I2C, SPI, GPIO, UART) soft peripherals which interface to the PMOD ports. Software for the Zynq Processing System is supplied as a Xilinx software development kit (SDK) project which includes a demonstration software application to evaluate each module's functionality. The lower level c-code driver routines are portable to the user's own software projects.

Prerequisites

- Xilinx ISE14.1 or a later version must be installed on the development PC, along with a license for Xilinx EDK/SDK.
- To modify the FPGA code or the software, the user should have a basic understanding of Xilinx Project Navigator (ISE) development tools, I^2C, SPI, GPIO buses, ANSI C, and (ideally) the Xilinx Embedded Development Kit toolset.
- Maxim Analog Essential Project Source (downloadable from www.maximintegrated.com/FPGA-modules)
- One or more of the peripheral modules boards in the Analog Essentials collection.

Typical System Architecture

Below is a generalized block diagram for the architecture implemented in the FPGA project. One of the key elements is the logic block that drives each Pmod port. Each Pmod port driver contains a dedicated I^{2}C port, SPI port, UART, and octal GPIO port. Any of these can be quickly connected to the Pmod port via software control. (The multiplexer setting is controlled by another internal GPIO port.) This eliminates the need to reprogram the FPGA during testing and evaluation of various peripheral modules.



Board specific notes

The Avnet Zedboard contains (5) Pmod connectors. However, only connectors JA, JB, JC and JD are driven by the logic/hardware resources in the above diagram. Pmod connector JE is not connected to the logic associated with this project.

Analog Essentials Getting Started Guide

Getting Started

Introduction

The Xilinx Zynq FPGA provides a (nearly) standalone dual-core Arm Cortex-A9 CPU along with a large amount of FPGA fabric and I/O. The Maxim Analog Essentials project utilizes a combination of software on one of the Cortex-A9 CPUs combined with programmable logic based peripherals that connect through the FPGA programmable fabric to PMOD connector JA on the Zedboard. The project requires that both an FPGA bitstream and Cortex-A9 executable be downloaded to the board in order to operate.

The first section of this guide leads the user through the process of loading the project into the Zedboard for execution. The section following that outlines the basic steps in modifying the project.

Overview: Downloading the Example/Demonstration Application

Below is a high-level overview of the steps required to download and run the FPGA project. Detailed instructions for each step are provided in the following pages.

1. From Maxim's website, download the latest version of the Maxim Analog Essentials project. Be sure to download the version specified for operation with Zedboard. The code will be delivered as a .zip file. Visit www.maximintegrated.com/fpga-modules, then click on the link to MAXPMBAE which takes you to the Quick-View page for *Analog Essentials*. From there, click on the "Technical Documents" to access the software downloads.

2. Extract the .zip file to a directory on your machine (direction location is arbitrary, but for the purposes of this document, it will be **c:\designs\maxim\v1_4**). See Appendix B in this document for a complete description of the included files.

3. Open the Xilinx SDK *(***Windows Start Menu > Xilinx ISE 14.1 > EDK > Xilinx Software Development Kit**).

4. Download the bitstream (.bit) file to the board. This bitstream contains the FPGA hardware design and software bootloader.

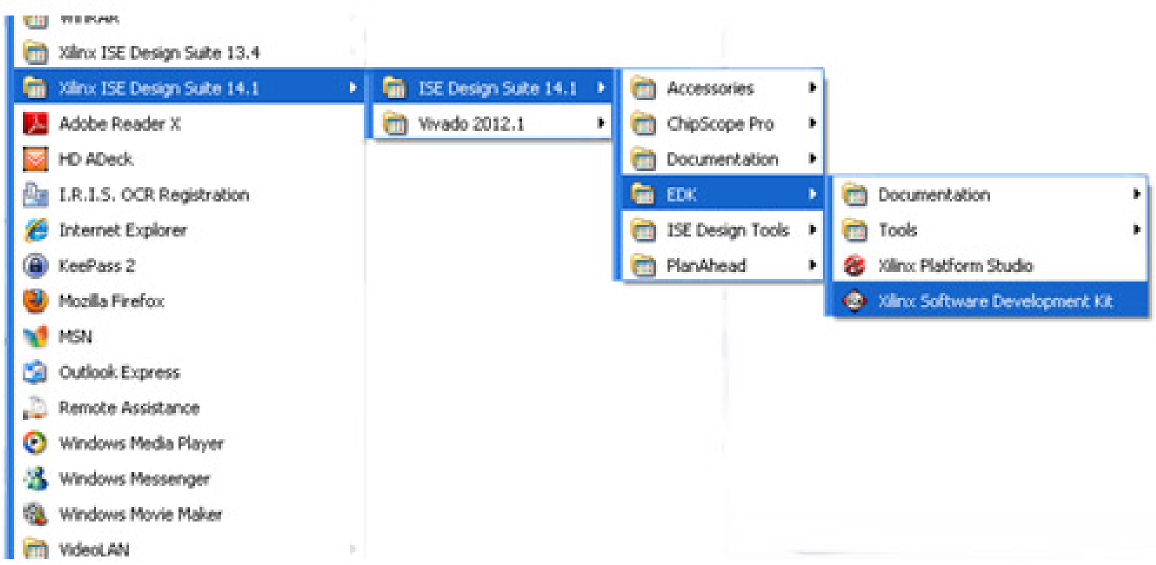
5. Open Hyperterminal or a similar communications program to communicate with FPGA board.

6. Use the SDK to download and run executable file (.elf) on the Cortex-A9.

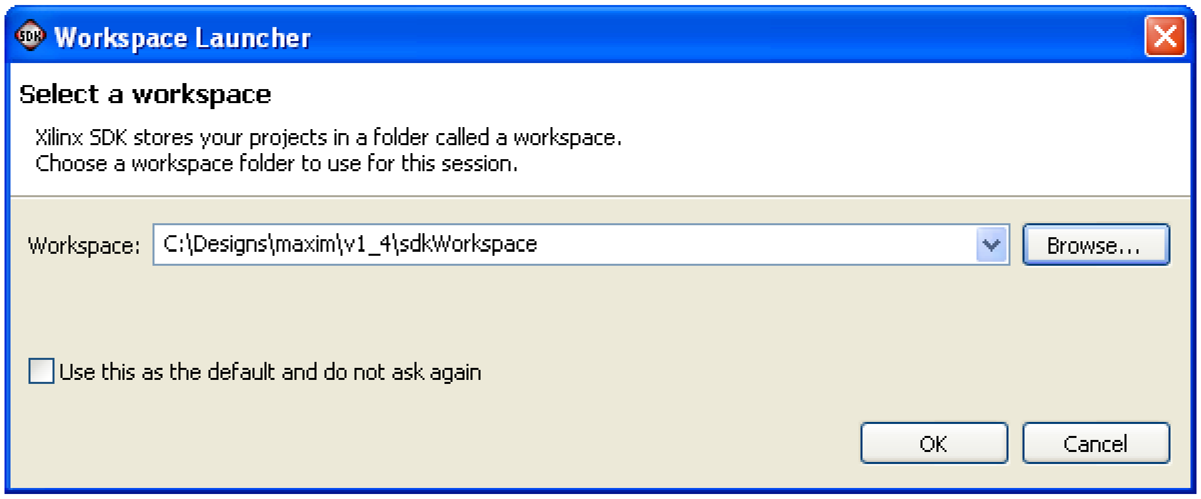
Detailed Steps: Downloading the Example/Demonstration Application

1. Opening the SDK

   a. From the **Windows Start Menu**, Open the Xilinx SDK as shown below.



   b. SDK will prompt for a workspace directory, the location where the software project is located. This directory is generally a subfolder within a Xilinx ISE/EDK project and will include board support files, as well as (.c) and (.h) files. Be certain to choose the correct Workspace directory. For this project, it will be **C:\Designs\maxim\v1_4\sdkWorkspace** (as shown below).


Workspace Launcher

Select a workspace

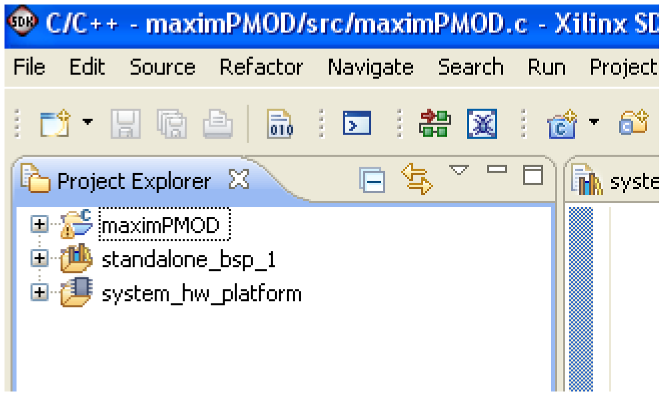
Xilinx SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Designs\maxim\v1_4\sdkWorkspace

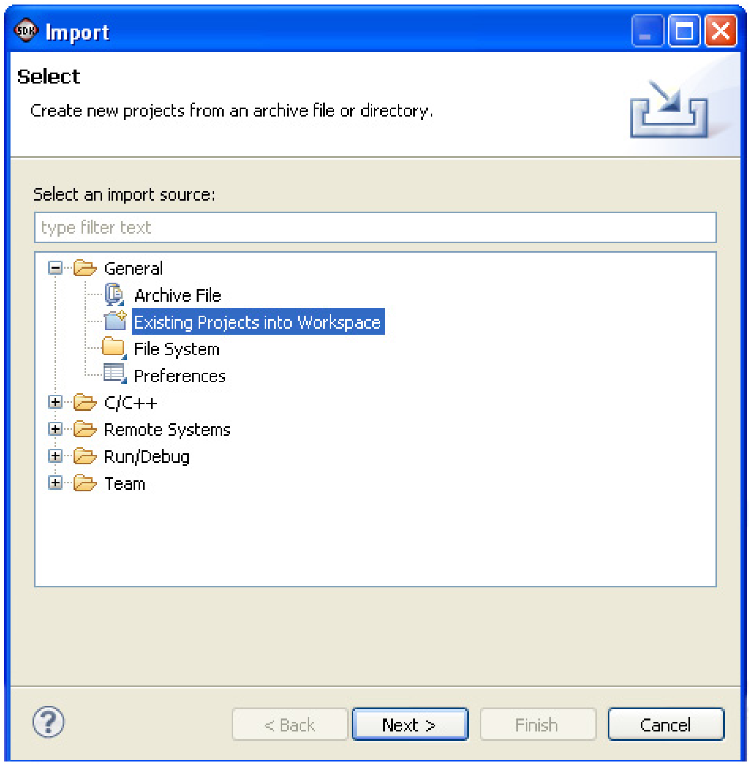
☐ Use this as the default and do not ask again

   c. Click **OK** to the prompt shown above and SDK will open. The Xilinx SDK is based on an Eclipse-based IDE, so it will be a familiar flow for many software developers.

d. Review the SDK IDE. The Project Explorer in the upper left tab should have three components as shown in the image below. If all three subfolders are present, you can skip the next step.

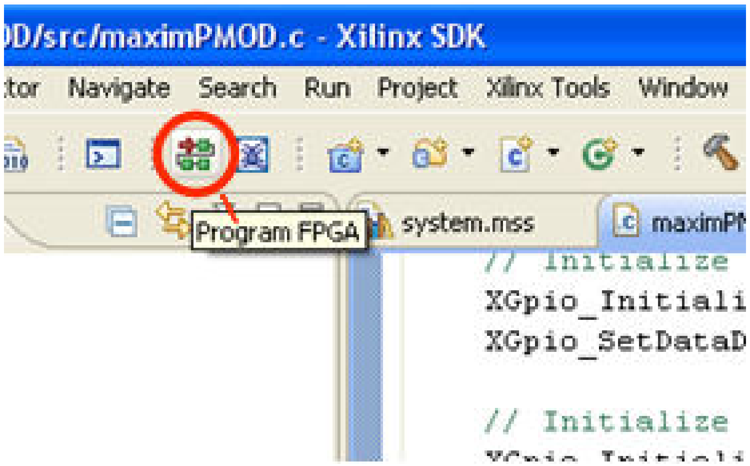


e. If the Project Explorer does not contain these three subfolders, launch the **File > Import menu**, expand the **General** tab and select **Existing Projects into Workspace**. Click **Next**. Set the root directory to **C:\designs\maxim\v1_4\sdkWorkspace**, and the missing projects should appear in SDK Project explorer with their checkboxes checked. Click **Finish** to import the projects.



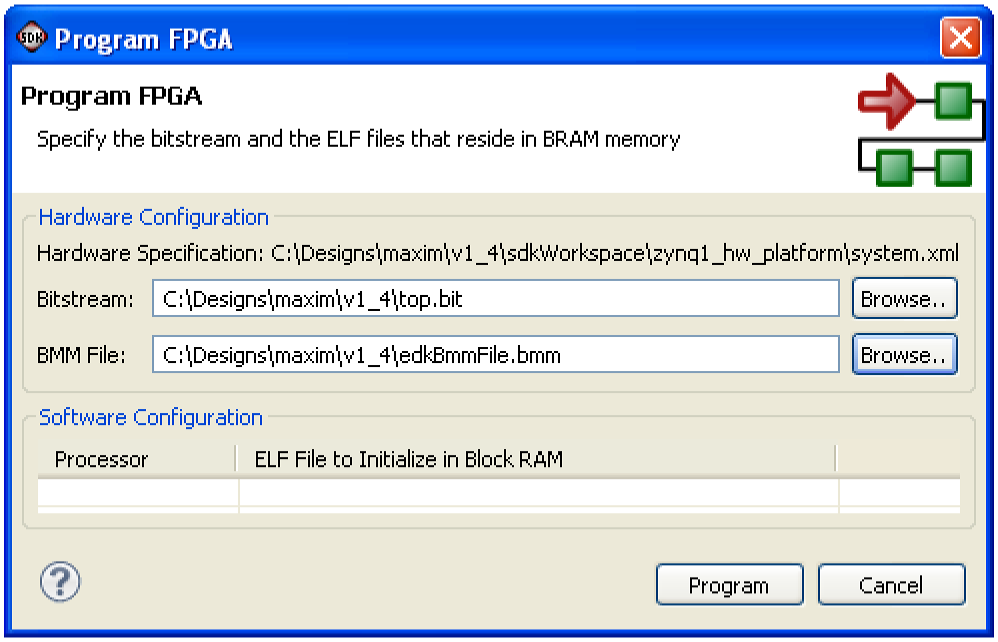
2. Downloading the bitstream (.bit) file to the board.

a. Click on the **Program FPGA** icon (which looks like a green chain of devices):


D/src/maximPMOD.c - Xilinx SDK
tor Navigate Search Run Project Xilinx Tools Window
Program FPGA
system.mss
maximPM

```
// Initialize
XGpio_Initiali
XGpio_SetDataD

// Initialize
```

b. The **Program FPGA** dialog box will appear. From here, an FPGA bitstream is selected as well as an FPGA bmm file and a (.elf) file to load into the Cortex-A9 memory as shown below. Be sure to select the toplevel.bit file, the edkBmmFile_bd.bmm, then select **Program**.



Specify the bitstream and the ELF files that reside in BRAM memory

Hardware Configuration

Hardware Specification: C:\Designs\maxim\v1_4\sdkWorkspace\zynq1_hw_platform\system.xml

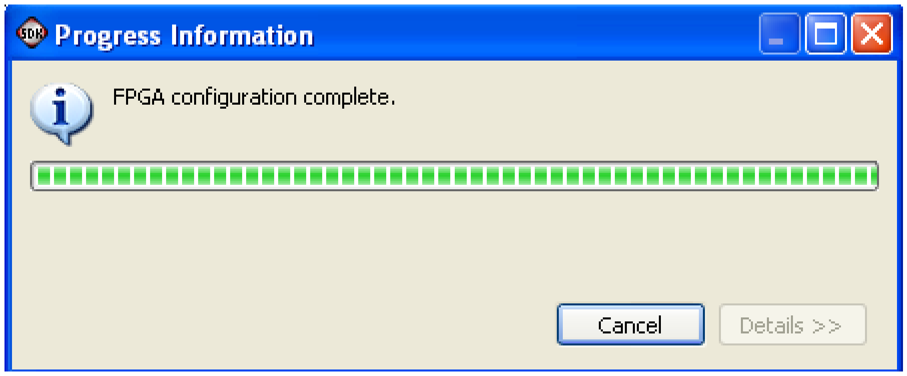
Bitstream: C:\Designs\maxim\v1_4\top.bit

BMM File: C:\Designs\maxim\v1_4\edkBmmFile.bmm

Software Configuration

Processor	ELF File to Initialize in Block RAM

It takes approximately 10 seconds to download the FPGA and a message box indicating “**FPGA configuration complete**” will appear.



FPGA configuration complete.

3. Setting Up the PC for Communication

Before loading the executable software file, the serial communications program should be opened so that the PC is ready to communicate with the FPGA board once the software begins executing. The example/demo software running on the host board communicates with the PC via a USB port set up to emulate a serial port (UART). To establish this communication link, the PC must be configured with the appropriate Windows drivers.

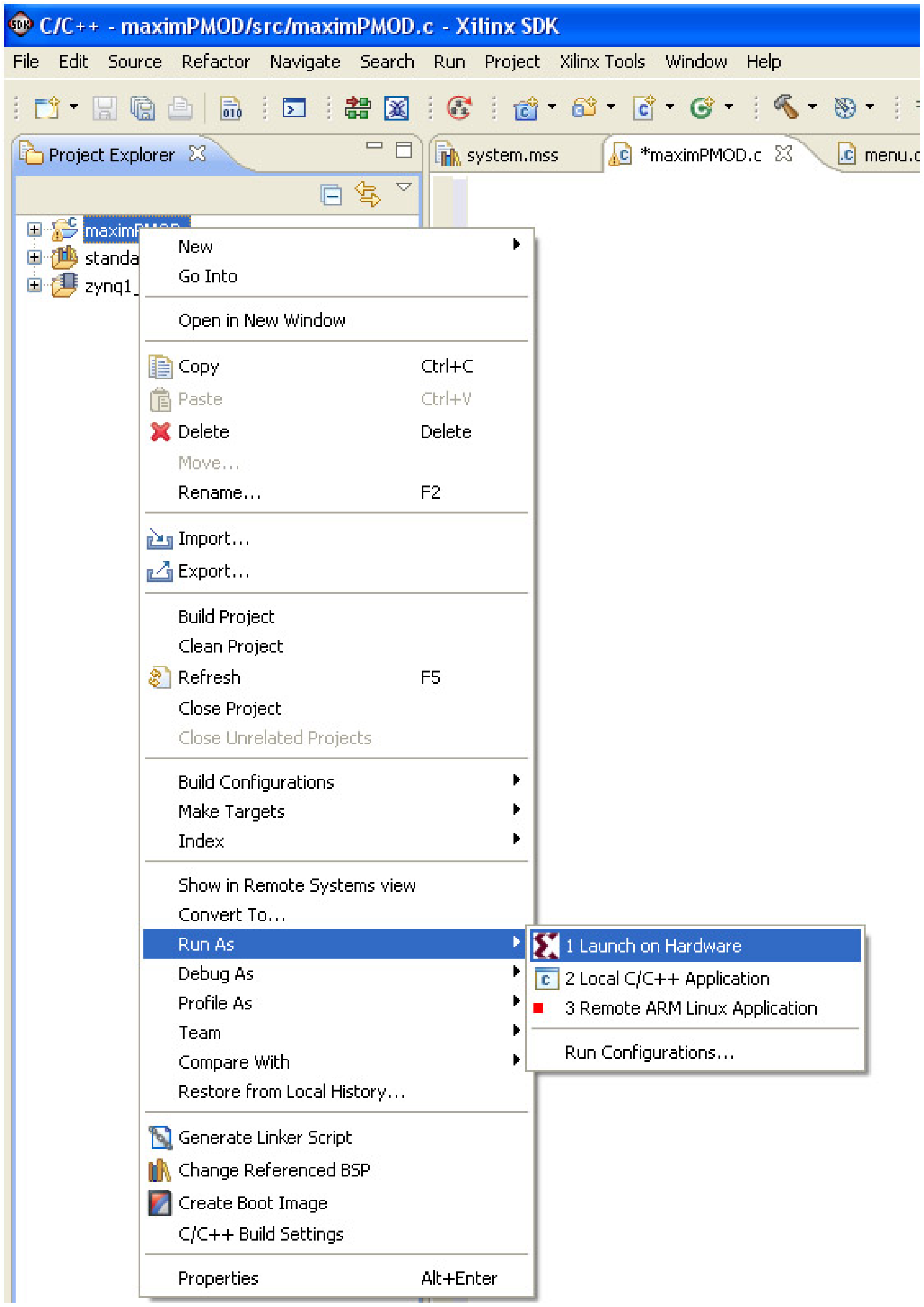
a. Zedboard implements a USB-UART bridge using the Cypress CY7C64225 chipset. In order to utilize this, the appropriate software drivers must be installed. If these are not already installed on your PC, follow the instructions available from www.zedboard.org/sites/default/files/CY7C64225_Setup_Guide_1_0.pdf The software drivers are available for download at http://www.cypress.com/?rID=63794

b. Once installed, Windows will assign a previously unused COM port. Use **Control Panel | System | Device Manager** to determine the COM port number. (It will be named *USB Serial Port*). Make a note of which COM port this is. That information is needed in the next step.

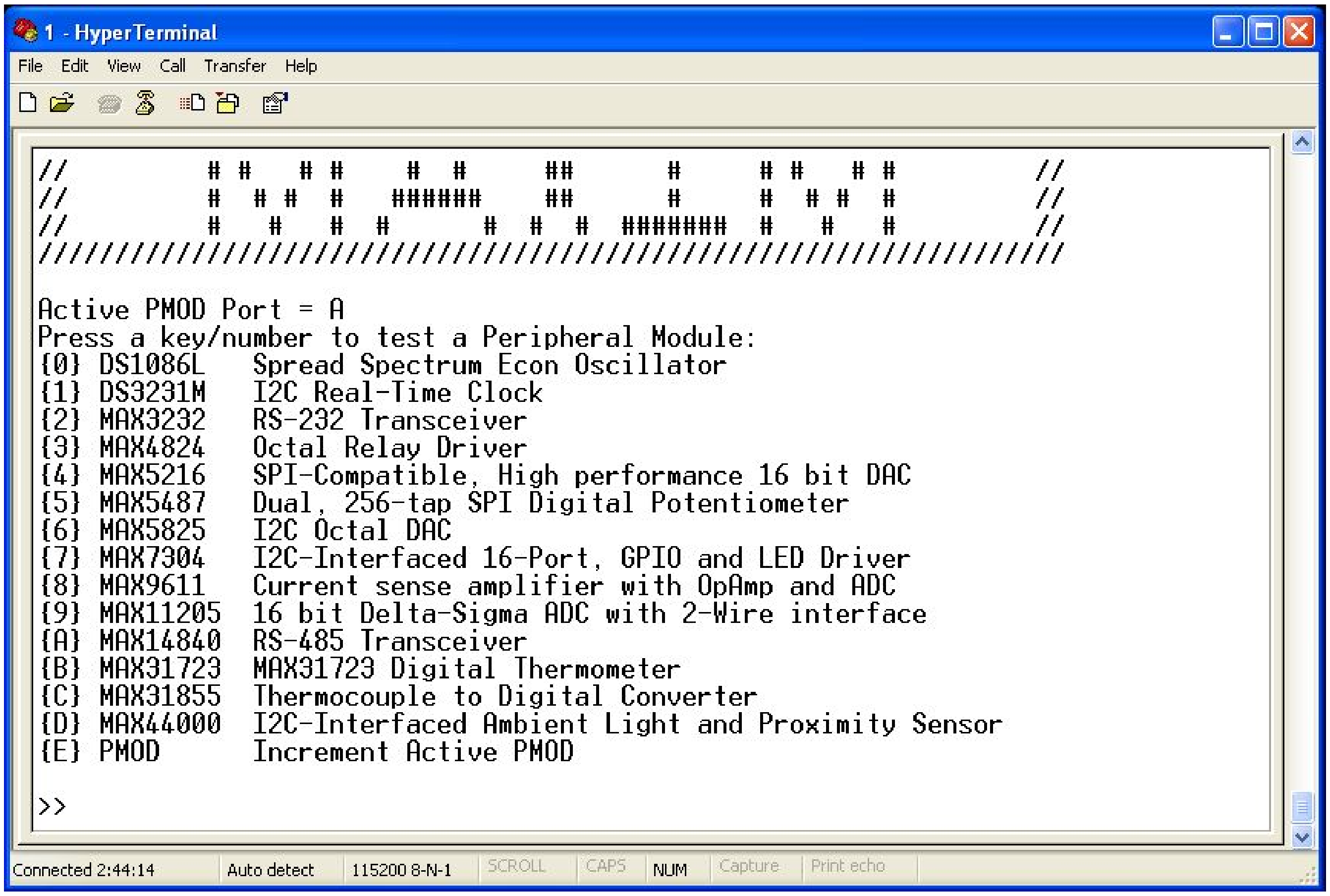
c. Next, a terminal emulation program needs to be installed and launched. For Windows XP® and earlier systems, the Hyperterminal program is the usual choice. However, since Hyperterminal was eliminated from Windows 7, it may be necessary to locate an alternative. Several are available, one of which is called "PuTTY" (www.putty.org). Whatever program you choose, the communication should be set up as follows: **bits per second:** 115,200; **data bits:** 8; **parity:** none; **stop bits:** 1; and **flow control:** none.

4. Using Xilinx SDK to download and run executable file (.elf) on the Cortex-A9.

   a. Under the Project Explorer (top left corner of SDK), highlight maximPMOD, and right click, and select **Run As->Launch on Hardware.**



b. After about 15 seconds of program downloading and initialization, the application will be running on the Cortex-A9 and the USB UART/Hyperterminal should show a menu like the one below. This is the top level menu for the module example programs.

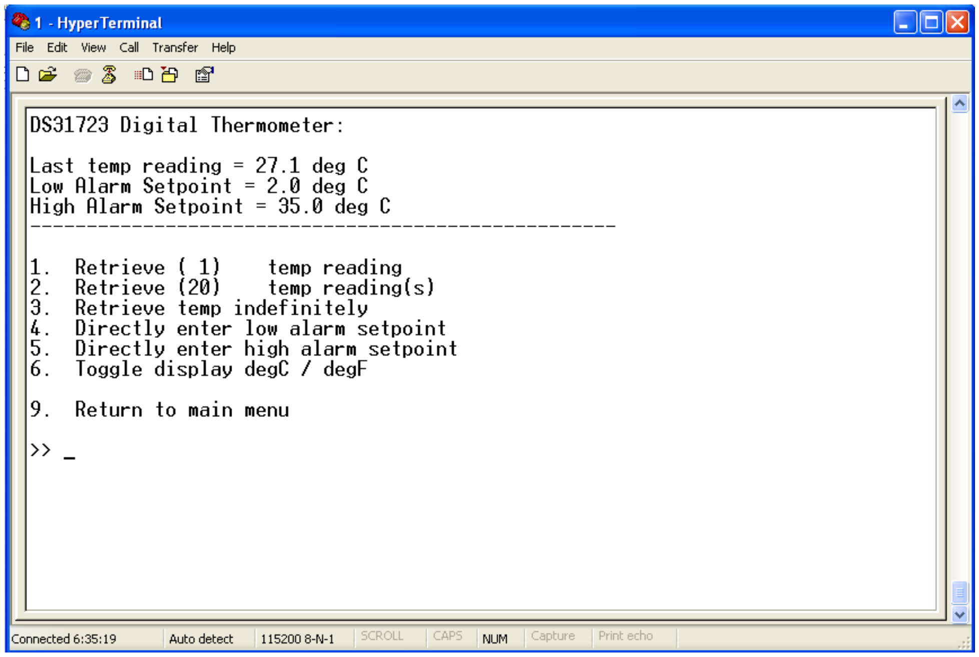


```
//         # #   # #    #  #     ##      #     # #   # #         //
//         #  # #  #   ######    ##      #     #  # #  #         //
//         #   #   #  #      #  #  #  #######  #   #   #         //
///////////////////////////////////////////////////////////////////

Active PMOD Port = A
Press a key/number to test a Peripheral Module:
{0} DS1086L   Spread Spectrum Econ Oscillator
{1} DS3231M   I2C Real-Time Clock
{2} MAX3232   RS-232 Transceiver
{3} MAX4824   Octal Relay Driver
{4} MAX5216   SPI-Compatible, High performance 16 bit DAC
{5} MAX5487   Dual, 256-tap SPI Digital Potentiometer
{6} MAX5825   I2C Octal DAC
{7} MAX7304   I2C-Interfaced 16-Port, GPIO and LED Driver
{8} MAX9611   Current sense amplifier with OpAmp and ADC
{9} MAX11205  16 bit Delta-Sigma ADC with 2-Wire interface
{A} MAX14840  RS-485 Transceiver
{B} MAX31723  MAX31723 Digital Thermometer
{C} MAX31855  Thermocouple to Digital Converter
{D} MAX44000  I2C-Interfaced Ambient Light and Proximity Sensor
{E} PMOD      Increment Active PMOD

>>
```

5. Running the demonstration programs

a. The Zedboard contains 5 Pmod ports labeled JA-JE. Only the first 4 are available for use by the demonstration software (JA-JD) and only 1 of these 4 ports is considered the 'active' port at one time. (User-written software can use any/all ports simultaneously.) The program defaults to Port A (Zedboard connector JA) begin active. Main menu entry (E) can be used to increment the active Pmod from A --> B --> C --> D. Of course, whichever port is selected in this manner should have a hardware module plugged into it.

b. Depending on which module was installed into the selected Pmod port, the corresponding menu item should be selected to run the demonstration program for that module. For example, if the MAX31723 module was inserted into the active Pmod port then menu item "B" would be selected and the submenu for the MAX31723 peripheral module will appear. The module may be exercised using the appropriate keyboard-selected commands. The submenu for the MAX31723 is shown below.



```
DS31723 Digital Thermometer:

Last temp reading = 27.1 deg C
Low Alarm Setpoint = 2.0 deg C
High Alarm Setpoint = 35.0 deg C
----------------------------------------------------

1.  Retrieve ( 1)    temp reading
2.  Retrieve (20)    temp reading(s)
3.  Retrieve temp indefinitely
4.  Directly enter low alarm setpoint
5.  Directly enter high alarm setpoint
6.  Toggle display degC / degF

9.  Return to main menu

>> _
```

Important note when running the demo programs:

Many of the programs allow incrementing and decrementing the primary registers or program values using the arrow keys. In these cases, the Up and Down arrows will increment and decrement the value in small steps (generally, but not always by an increment of one). The left and right arrow keys will increment and decrement the value in larger steps. This option is not indicated on the submenus.

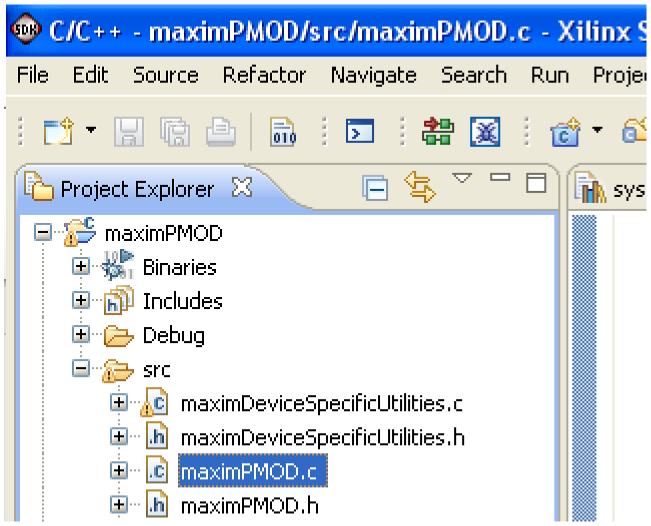
The remaining sections of this document are intended to serve only as brief outlines of and introductions to the steps necessary to modify the hardware and software for this project. Detailed instructions on these procedures and the design tools are beyond the scope of this document.

Modifying the Project C Code

This section provides detailed information on changing the project source code, recompiling, and redownloading.

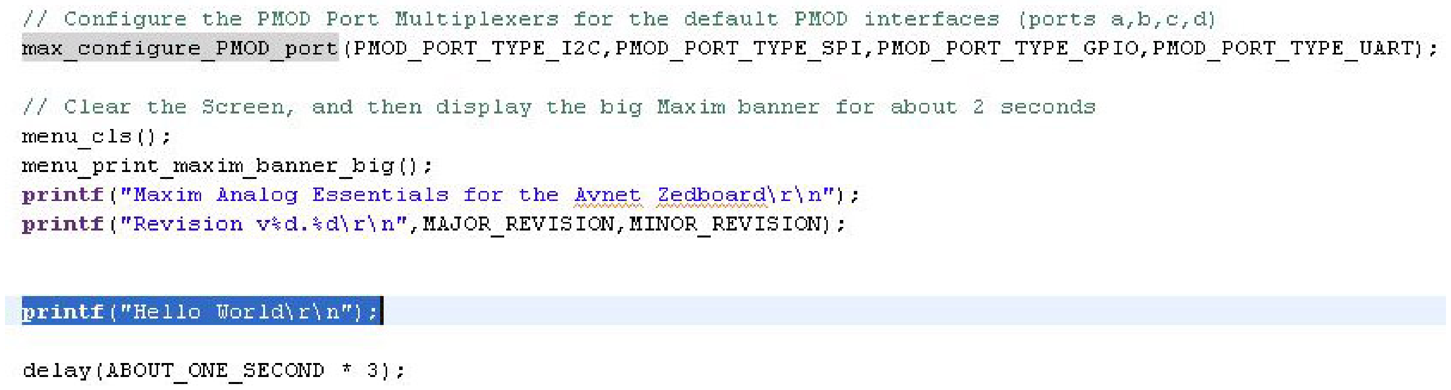
1. Modifying the C code and recompiling

a. To modify the C code, navigate in the Project Explorer to the maximPMOD.c file.


C/C++ - maximPMOD/src/maximPMOD.c - Xilinx S
File Edit Source Refactor Navigate Search Run Proje
Project Explorer
sys
maximPMOD
Binaries
Includes
Debug
src
maximDeviceSpecificUtilities.c
maximDeviceSpecificUtilities.h
maximPMOD.c
maximPMOD.h

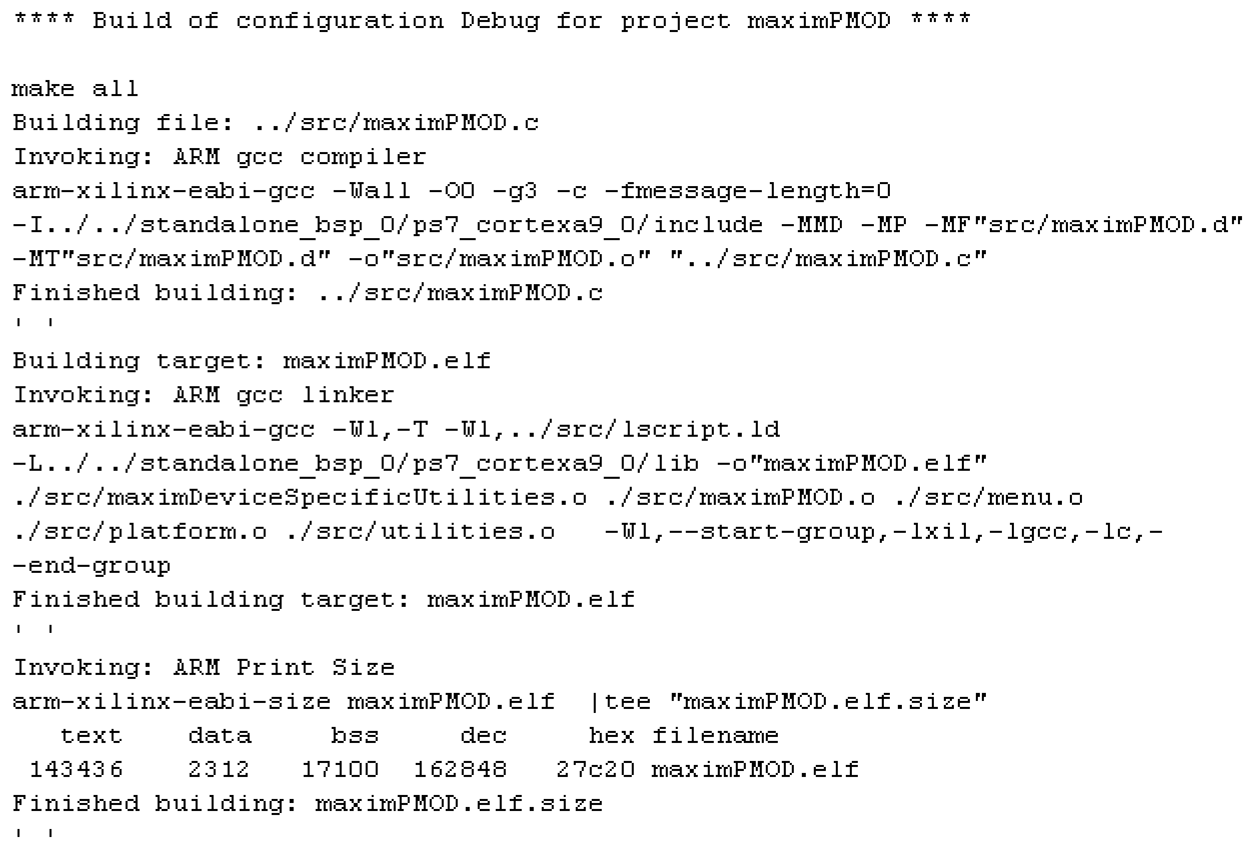
b. Double-click on maximPMOD.c. This will open the file in the source code editor. The file maximPMOD.c contains the main() function for the demonstration application. The main() function initializes several board peripherals, then prints a menu to the USB UART on the host board. Each menu choice calls an underlying menu function which allows functions of each peripheral module to be tested.

c. Make an edit to the maximPMOD.c file. We will uncomment a line of code to add a printf statement to the boot-up screen. This printf statement is located about half-way through the file maximPMOD.c.



```
// Configure the PMOD Port Multiplexers for the default PMOD interfaces (ports a,b,c,d)
max_configure_PMOD_port(PMOD_PORT_TYPE_I2C,PMOD_PORT_TYPE_SPI,PMOD_PORT_TYPE_GPIO,PMOD_PORT_TYPE_UART);

// Clear the Screen, and then display the big Maxim banner for about 2 seconds
menu_cls();
menu_print_maxim_banner_big();
printf("Maxim Analog Essentials for the Avnet Zedboard\r\n");
printf("Revision v%d.%d\r\n",MAJOR_REVISION,MINOR_REVISION);


printf("Hello World\r\n");

delay(ABOUT_ONE_SECOND * 3);
```

d. After un-commenting the “Hello World” line, select **File** > **Save** from the menu. This will force the IDE to recompile with the code change. The contents of the console should be as shown below.



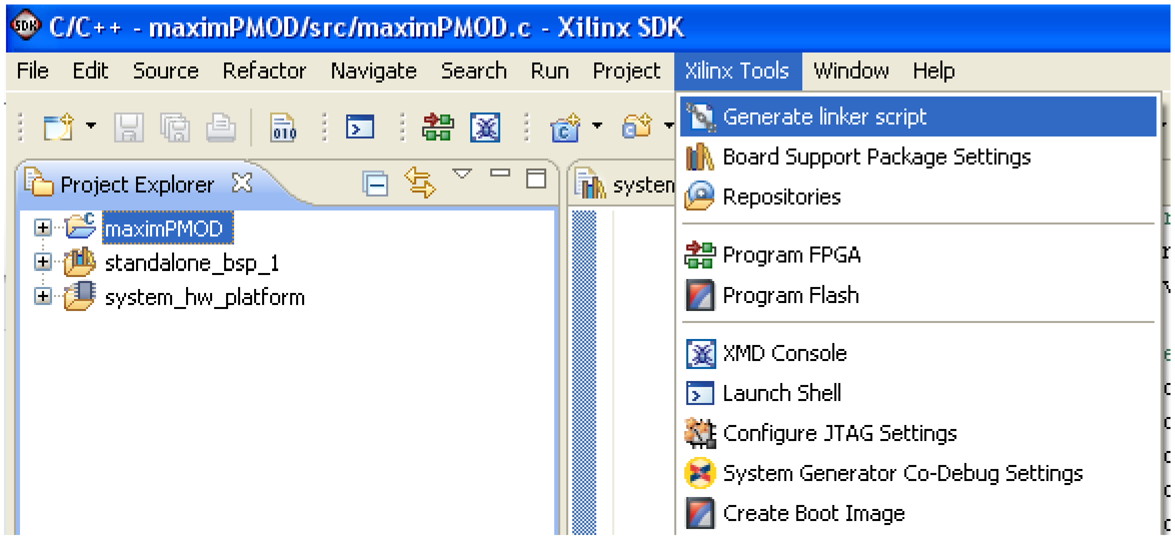
```
**** Build of configuration Debug for project maximPMOD ****

make all
Building file: ../src/maximPMOD.c
Invoking: ARM gcc compiler
arm-xilinx-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0
-I../../standalone_bsp_0/ps7_cortexa9_0/include -MMD -MP -MF"src/maximPMOD.d"
-MT"src/maximPMOD.d" -o"src/maximPMOD.o" "../src/maximPMOD.c"
Finished building: ../src/maximPMOD.c
' '
Building target: maximPMOD.elf
Invoking: ARM gcc linker
arm-xilinx-eabi-gcc -Wl,-T -Wl,../src/lscript.ld
-L../../standalone_bsp_0/ps7_cortexa9_0/lib -o"maximPMOD.elf"
./src/maximDeviceSpecificUtilities.o ./src/maximPMOD.o ./src/menu.o
./src/platform.o ./src/utilities.o   -Wl,--start-group,-lxil,-lgcc,-lc,-
-end-group
Finished building target: maximPMOD.elf
' '
Invoking: ARM Print Size
arm-xilinx-eabi-size maximPMOD.elf  |tee "maximPMOD.elf.size"
   text    data     bss     dec     hex filename
 143436    2312   17100  162848   27c20 maximPMOD.elf
Finished building: maximPMOD.elf.size
' '
```

e. If the ‘**Finished building: maximPMOD.elf’** line is received, the code has been rebuilt successfully and is ready to re-download to the Cortex-A9 via the **‘Run As->Launch on Hardware’** command.

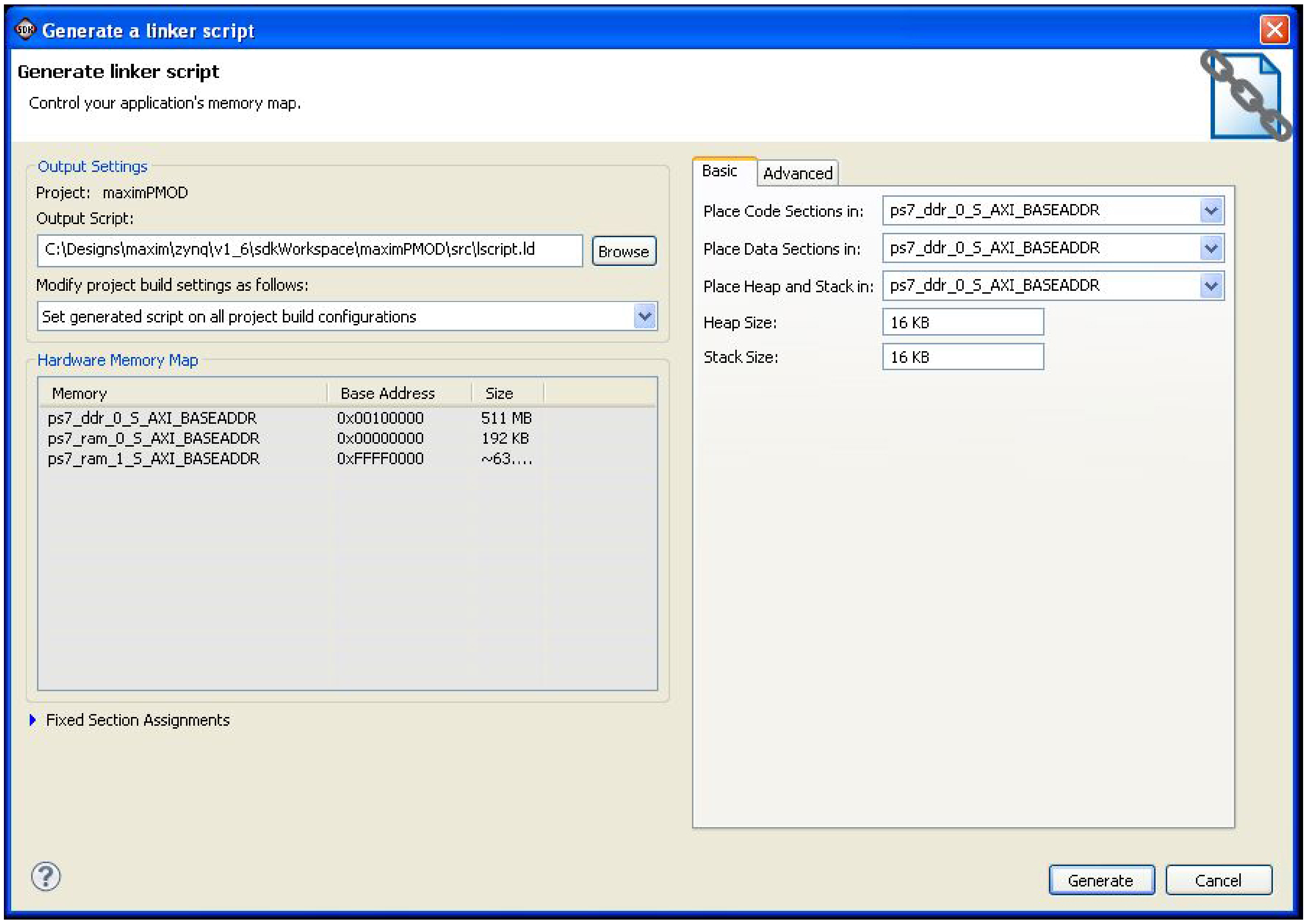
2. Optional: Setting the memory locations and generating a linker script

The Zynq IC includes 256k of internal SRAM which is sufficient to contain the Analog Essentials example/demo program. The linker script allows the user to select the target memory location for this program.

a. Highlight the **maximPMOD** project under Project Explorer, and choose **Menu** > **Xilinx Tools** > **Generate Linker Script** as follows.



b. This will bring up the linker script menu as shown below.



Generate linker script

Control your application's memory map.

Output Settings

Project: maximPMOD

Output Script: C:\Designs\maxim\zynq\v1_6\sdkWorkspace\maximPMOD\src\lscript.ld

Modify project build settings as follows: Set generated script on all project build configurations

Hardware Memory Map

Memory	Base Address	Size
ps7_ddr_0_S_AXI_BASEADDR	0x00100000	511 MB
ps7_ram_0_S_AXI_BASEADDR	0x00000000	192 KB
ps7_ram_1_S_AXI_BASEADDR	0xFFFF0000	~63....

Fixed Section Assignments

Place Code Sections in:	ps7_ddr_0_S_AXI_BASEADDR
Place Data Sections in:	ps7_ddr_0_S_AXI_BASEADDR
Place Heap and Stack in:	ps7_ddr_0_S_AXI_BASEADDR
Heap Size:	16 KB
Stack Size:	16 KB

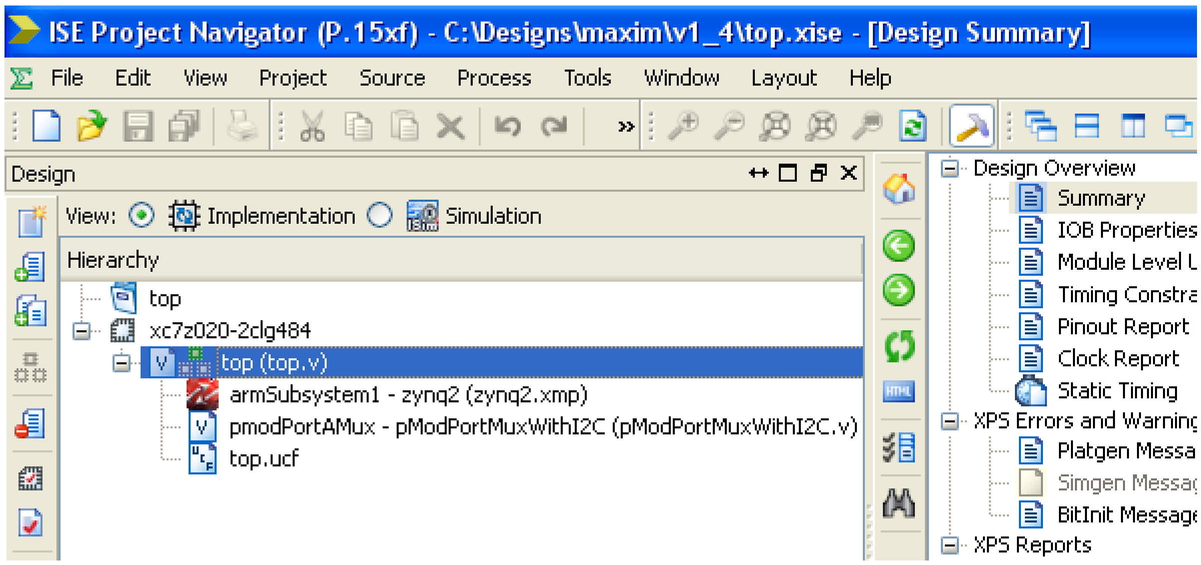
From this window the code, data, heap/stack section placement (16384 bytes each is suitable) can be chosen. For the application demonstration, this must be set to ps7_ddr_0_S_AXI_BASEADDR.

c. Clicking **Generate** will rebuild the (.elf) file with the memory locations set to the external PSRAM memory.

Detailed Steps: Opening the ISE Project to View the HDL and Launch EDK

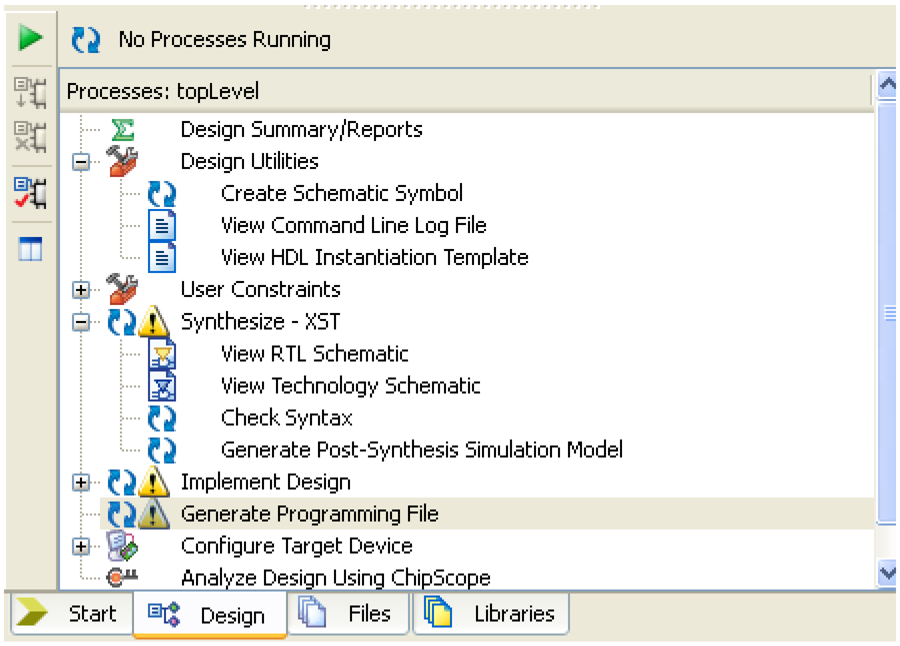
1. Open Xilinx Project Navigator 14.1

    a. Verify that the tools are using the correct directory by looking at the directory displayed at the top of the ISE screen. Xilinx Tools default to opening the last used directory. Be careful to make sure that you are not working on an older revision of the design. If ISE is not using the latest directory, then close the project and re-open the latest project (in this case, **C:\designs\maxim\v1_4\top.xise**).


ISE Project Navigator (P.15xf) - C:\Designs\maxim\v1_4\top.xise - [Design Summary]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
top
xc7z020-2clg484
top (top.v)
armSubsystem1 - zynq2 (zynq2.xmp)
pmodPortAMux - pModPortMuxWithI2C (pModPortMuxWithI2C.v)
top.ucf
Design Overview
Summary
IOB Properties
Module Level U
Timing Constra
Pinout Report
Clock Report
Static Timing
XPS Errors and Warning
Platgen Messa
Simgen Messag
BitInit Message
XPS Reports

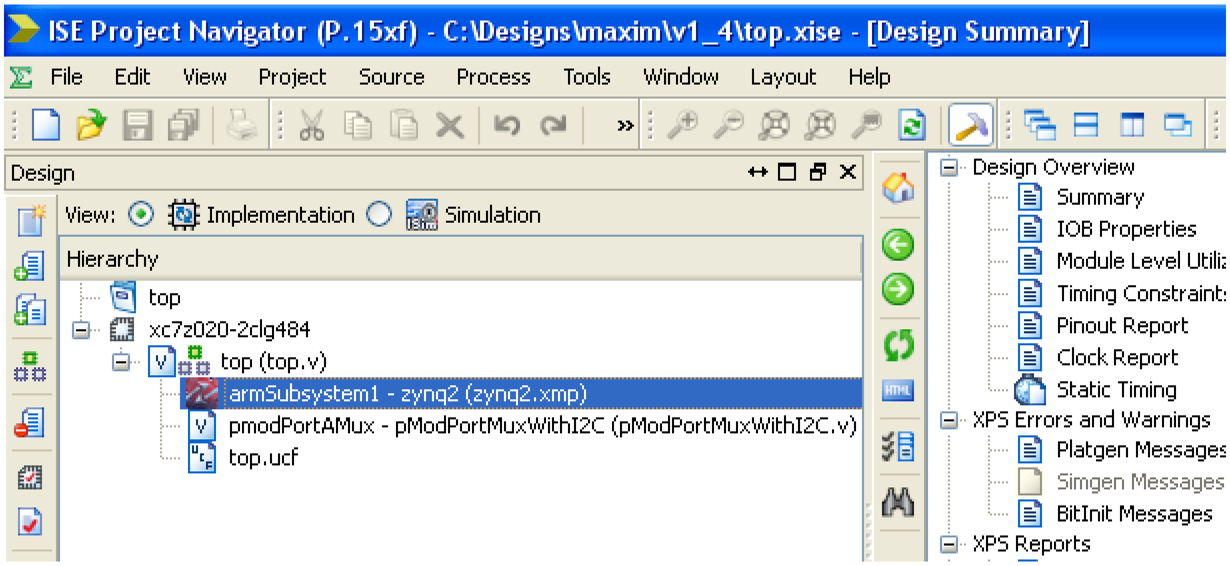
    b. The top level source code for this project is contained within top.v. Double-click on top.v to open the ISE Verilog editor.

c. Once any changes are made, the project can be rebuilt by selecting **Generate Programming File** from the Design tab (see below). This will generate a new toplevel.bit file.



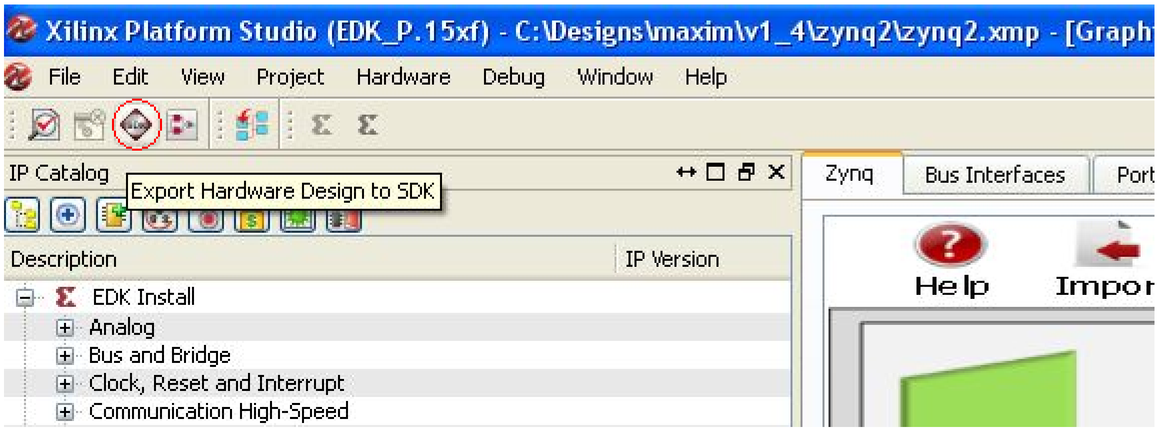
2. Open the EDK/XMP project (Zynq Processing System /Peripherals)

   a. Highlight and double-click on "armSubsystem1 – zynq2(zynq2.xmp)" as shown below. Changes to the internal peripherals can be configured here. Examples of this are adding additional I^{2}C ports or changing the baud rate on the UARTs.

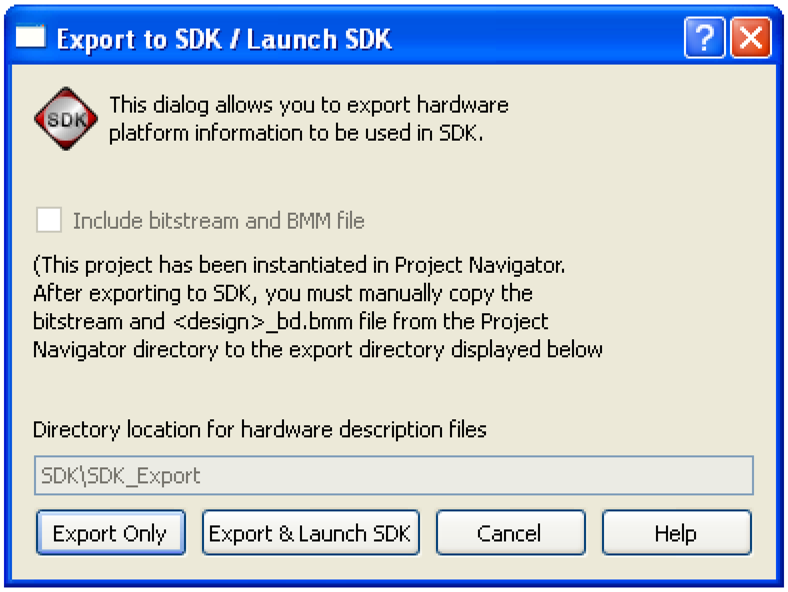


   b. After making any changes to the Zynq Processing System within EDK, run DRC and Generate netlist, then rebuild the ISE project to incorporate those changes into the base bitstream (toplevel.bit).

   c. Re-export the design to SDK by clicking on the Export to SDK icon in the upper left menu area.



d. This will launch a dialog box asking to export the design or export and launch SDK. At the dialog box, click the **Export & Launch SDK** button, or if the SDK is already running, choose **Export Only**. This process will bring the SDK project .mss/.xml files in sync with the EDK project.


Export to SDK / Launch SDK

This dialog allows you to export hardware platform information to be used in SDK.

☐ Include bitstream and BMM file

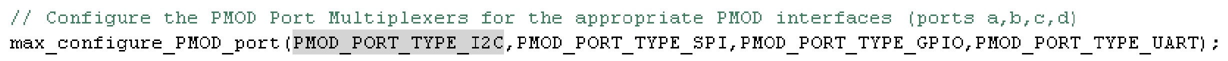
(This project has been instantiated in Project Navigator. After exporting to SDK, you must manually copy the bitstream and <design>_bd.bmm file from the Project Navigator directory to the export directory displayed below

Directory location for hardware description files

SDK\SDK_Export

Appendix A: Project Notes

1. Pmod port JA, JB, JC and JD each have a dedicated I^2C port, SPI port UART and octal GPIO. These communications peripherals are multiplexed to the physical port via an HDL designed multiplexer. (Refer to the diagram in the *Typical System Architecture* section.) An example of this can be seen in the maximPMOD.c file at about the midpoint, as shown in the clip below.

   ```
   // Configure the PMOD Port Multiplexers for the appropriate PMOD interfaces (ports a,b,c,d)
   max_configure_PMOD_port(PMOD_PORT_TYPE_I2C,PMOD_PORT_TYPE_SPI,PMOD_PORT_TYPE_GPIO,PMOD_PORT_TYPE_UART);
   ```

   (Note that in Pmod port JE is not driven by the logic within this project.)

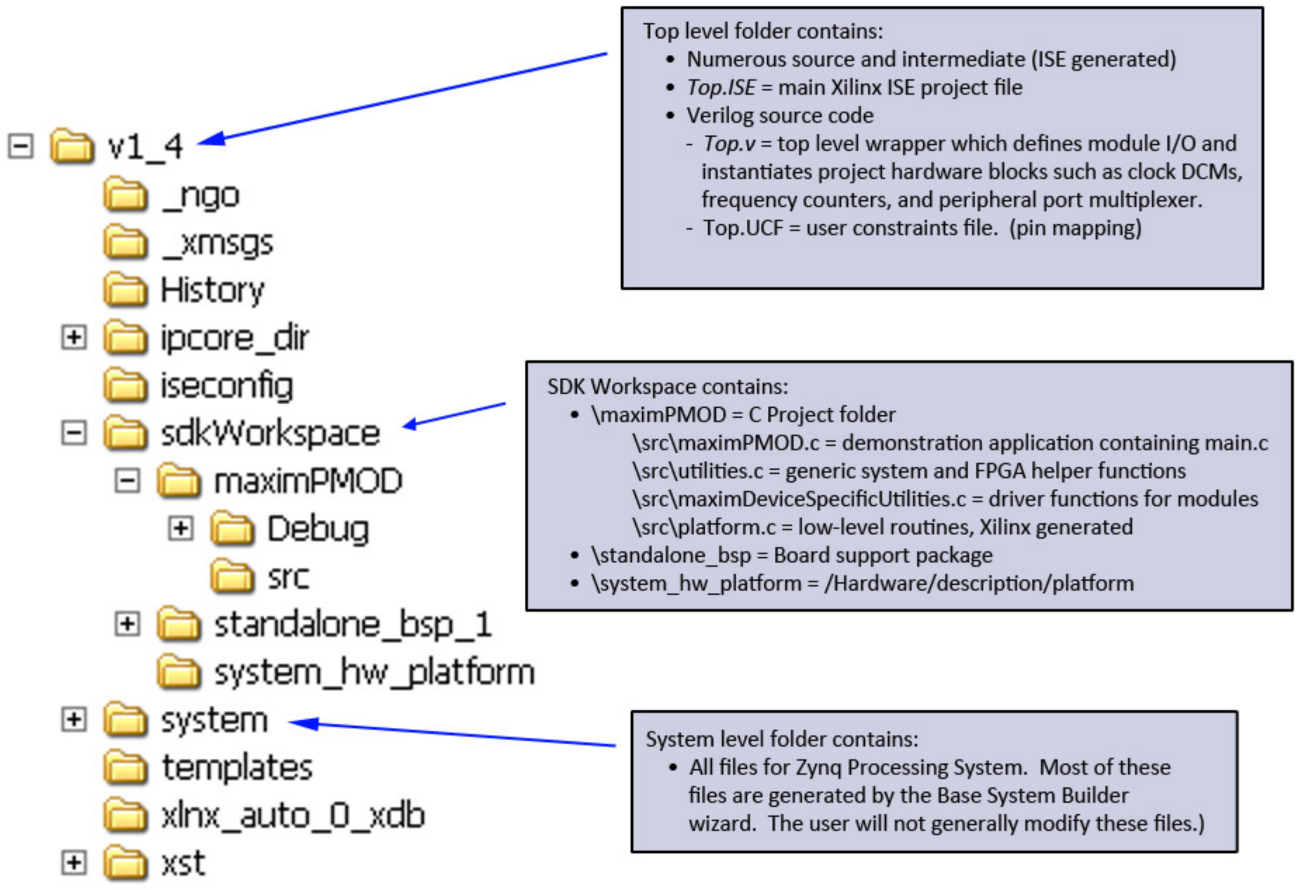
2. The functions in maximDeviceSpecificUtilities.c are used to access the major functionality provided by the Maxim ICs. In general, these functions return an integer value (TRUE/FALSE) that describes whether or not the function successfully completed. Although it is not required to check this return value, it is good programming practice to do so.

3. Functions will typically require a peripheral port address. These ports are named as follows:
   a. XPAR_IIC_0_BASEADDR
   b. XPAR_SPI_0_BASEADDR
   c. XPAR_AXI_UARTLITE_0_BASEADDR

4. GPIO calls receive a pointer to a Xilinx XGpio instance
   a. XGpio portA;
   b. XGpio_Initialize(&portA, XPAR_AXI_GPIO_0_DEVICE_ID);

Since the functions return TRUE/FALSE as a pass/fail indication, when the functions return a value, that value is passed back to the calling function via a pointer.

Appendix B: Project Structure and Key Filenames



Top level folder contains:

- Numerous source and intermediate (ISE generated)
- *Top.ISE* = main Xilinx ISE project file
- Verilog source code
  - *Top.v* = top level wrapper which defines module I/O and instantiates project hardware blocks such as clock DCMs, frequency counters, and peripheral port multiplexer.
  - Top.UCF = user constraints file. (pin mapping)

SDK Workspace contains:

- \maximPMOD = C Project folder
  - \src\maximPMOD.c = demonstration application containing main.c
  - \src\utilities.c = generic system and FPGA helper functions
  - \src\maximDeviceSpecificUtilities.c = driver functions for modules
  - \src\platform.c = low-level routines, Xilinx generated
- \standalone_bsp = Board support package
- \system_hw_platform = /Hardware/description/platform

System level folder contains:

- All files for Zynq Processing System. Most of these files are generated by the Base System Builder wizard. The user will not generally modify these files.)

All trademarks are property of their respective owners