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 LX9 and Nexys™-3 boards, both of which are based on Spartan®-6 FPGAs. 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.

4. **Nexys-3 and LX9 Differences.** Most of the instructions in this document refer to both the Nexys-3 board and the LX9. Instances where a procedure is specific to one board or other will be denoted by the following indicators:

   LX9 Nexys-3

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

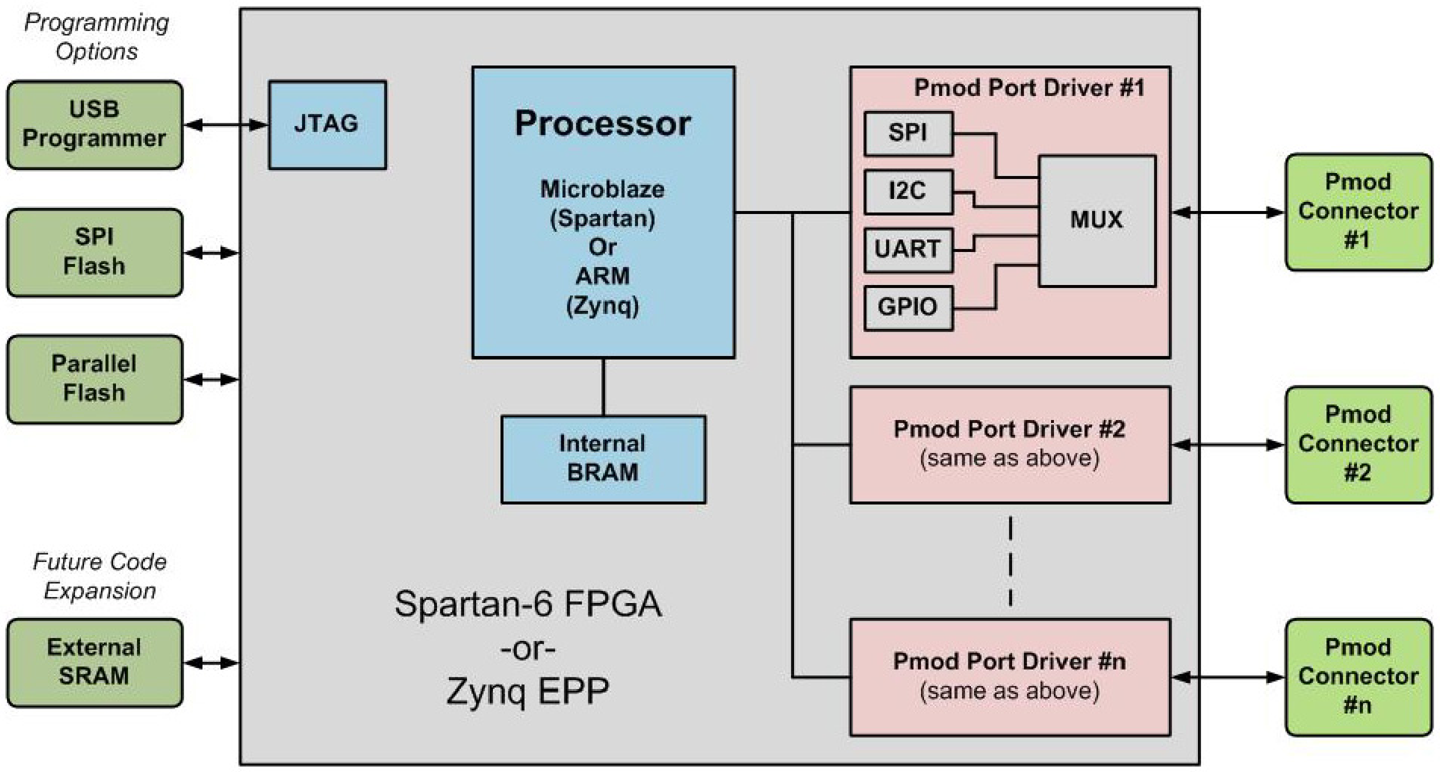
The top level of the hardware design is a Xilinx ISE® Project Navigator Project (.xise) for Xilinx ISE version 13.4. The Verilog-based HDL design instantiates the Microblaze™ core, the support hardware required to run the Microblaze, and the peripherals (I^2C, SPI, GPIO, and UART) which interface to the Pmod ports. This 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 ISE13.4 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.maxim-ic.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

Nexys-3

Digilent Nexys-3: This FPGA board contains 4 Pmod connectors, all of which are driven by the logic/hardware resources as shown in the above diagram.

LX9

Avnet LX9: Although the LX9 board contains 2 Pmod connectors, the FPGA does not contain a sufficient number of gates to implement two complete copies of the drivers as shown above. Therefore, only one of the Pmod ports on the LX9 (“J5”) is driven as shown. Pmod connector “J4” is driven by an octal GPIO interface block and not utilized by the supplied software.

Getting Started

Introduction

With small Microblaze FPGA designs, it is typical for the binary executable program to fit entirely within the internal block RAM of the FPGA. In these cases, the FPGA can be programmed by simply downloading a single bit-stream file, which contains both the FPGA logic and the binary executable for the running software on the processor. However, the size of the sample programs for the Analog Essentials collection exceeds the capacity of the internal block RAM memory. Therefore, external memory (outside the FPGA) must be utilized for both program and data storage. Because of this, loading the program is a multistep process. The first step loads, then executes a boot loader. The boot loader then loads the .elf (executable software file for the Microblaze). While this is not as simple as downloading a single file, it represents the situation encountered with many real-world applications.

The first section of this guide leads the user through the process of loading the project into the FPGA board 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 that corresponds to your FPGA board. The code will be delivered as a .zip file. Visit www.maxim-ic.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 13.4 > EDK > Xilinx Software Development Kit**).

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

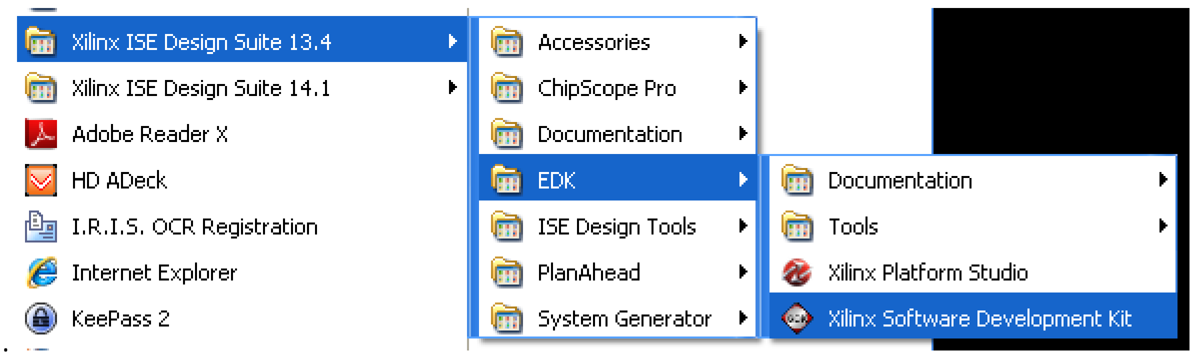
5. Open Hyperterminal or a similar communications program to communicate with FPGA board.

6. Use Xilinx SDK/XMD debugger console to download and run executable file (.elf) on the Microblaze.

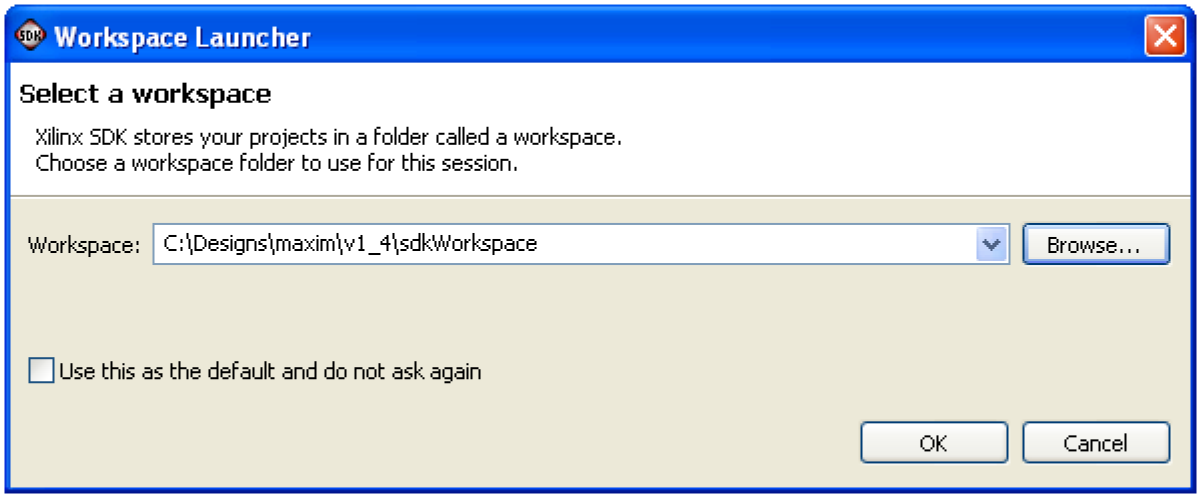
Detailed Steps: Downloading the Example/Demonstration Application

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

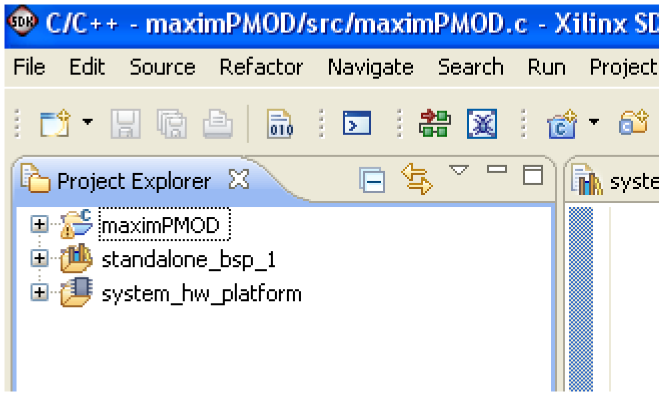
Select a workspace

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

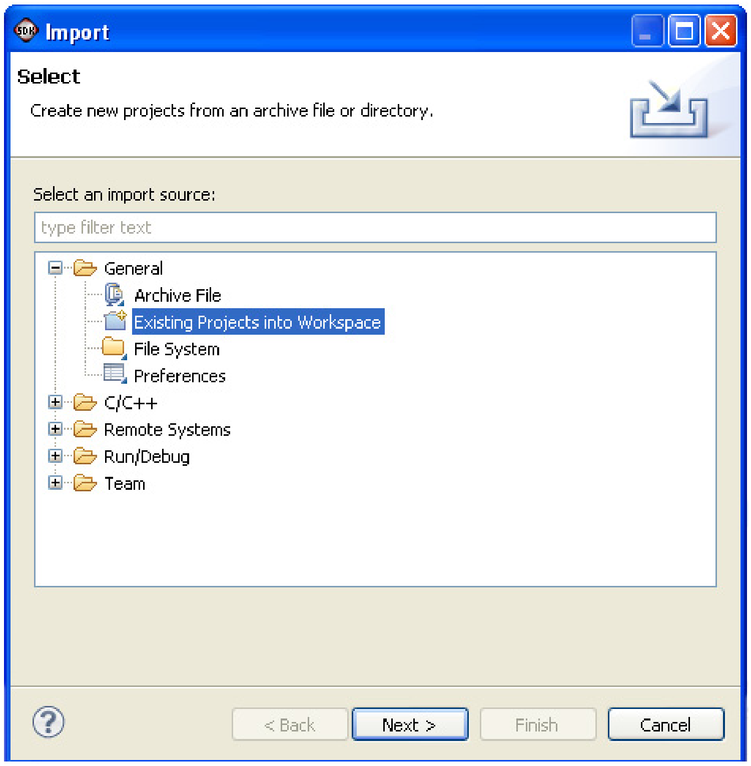
Workspace: C:\Designs\maxim\v1_4\sdkWorkspace

   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.


C/C++ - maximPMOD/src/maximPMOD.c - Xilinx SD
File Edit Source Refactor Navigate Search Run Project
Project Explorer
maximPMOD
standalone_bsp_1
system_hw_platform
syste

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.


Import

Select

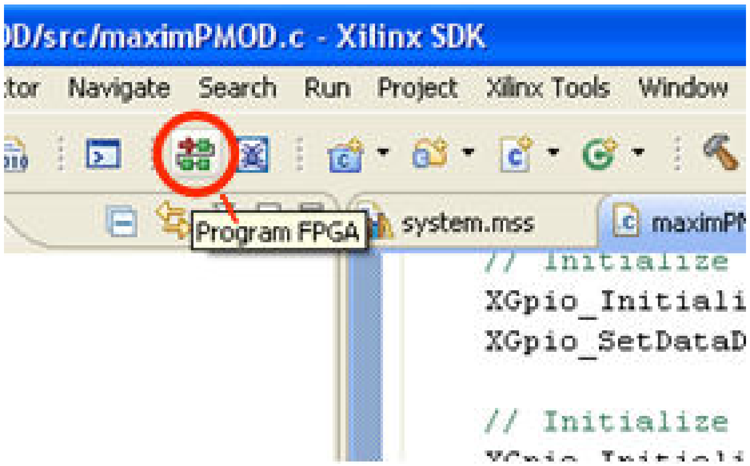
Create new projects from an archive file or directory.

Select an import source:

type filter text

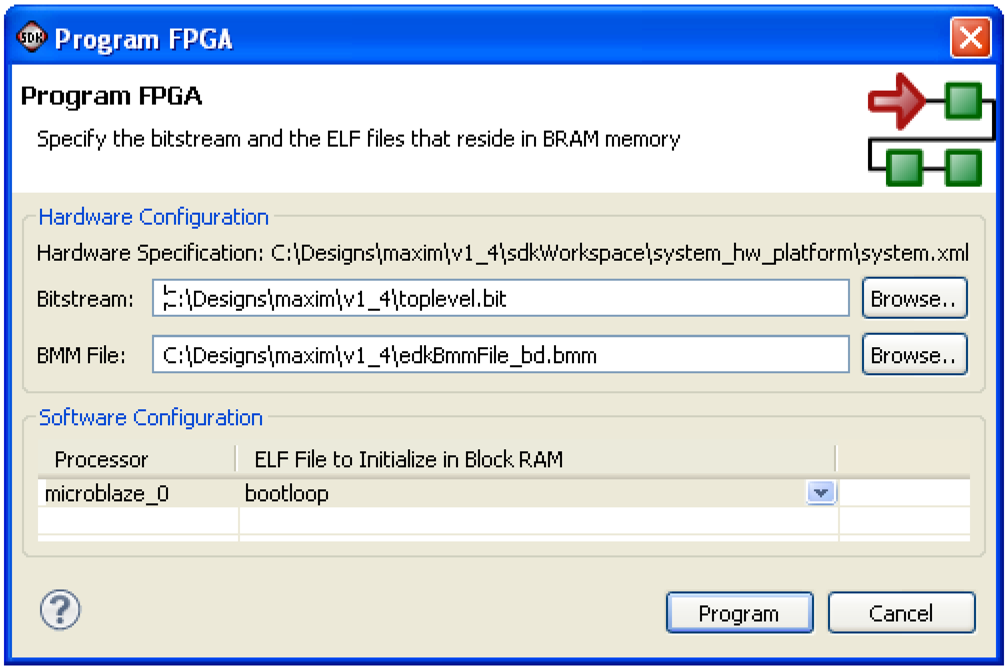
2. Downloading the bitstream (.bit) file to the board.

    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
maximPI

```
// 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 Microblaze memory as shown below. Be sure to select the toplevel.bit file, the edkBmmFile_bd.bmm, and the bootloop file as shown, then select **Program**.



Program FPGA

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

Hardware Configuration

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

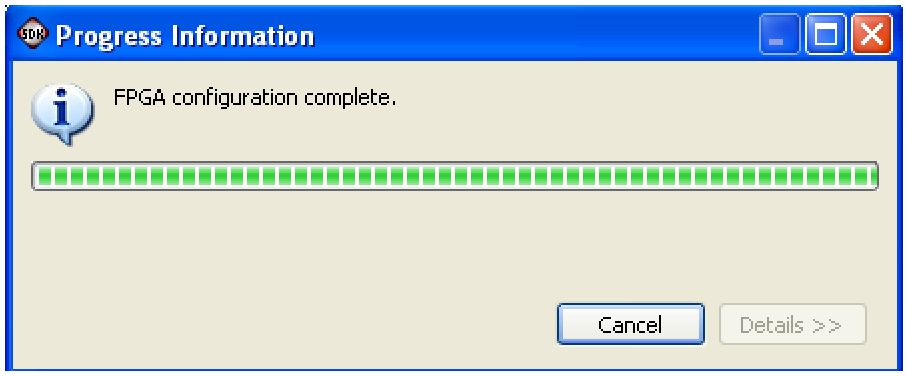
Bitstream: C:\Designs\maxim\v1_4\toplevel.bit

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

Software Configuration

Processor	ELF File to Initialize in Block RAM
microblaze_0	bootloop

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



FPGA configuration complete.

A brief note summarizing the previous steps:

As discussed earlier in this document, the code size of this program exceeds that available in block memory internal to the FPGA, necessitating the use of the external RAM. This requires a multi-stage loading process implemented with a boot-loader. This operates a bit differently from smaller programs where the entire FPGA configuration data and executable code is contained in one .bit file and is loaded in one step

The bitstream (.bit) file is used to configure the FPGA and initiate the bootloading process. This bitstream contains all the configuration information necessary to initialize the FPGA with the gate level hardware design, including block memories, which are used for the base Microblaze memory.

The (.bmm) file provides the location of block RAM memories (internal to the FPGA) that are used as codespace for the Microblaze. The (.elf) file contains the instructions that represent the Microblaze executable program. That file will be loaded into external pseudostatic RAM (PSRAM) using the Xilinx debugger in a subsequent step, right after we set up PC communications in the next step.

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. While the drivers for both the LX9 and Nexys-3 are functionally equivalent, the boards use different USB-UART bridge ICs. Therefore, the boards use different drivers. Also, a suitable communications program such as HyperTerminal should be invoked.

a. The Nexys-3 utilizes the FTDI USB UART, so you need to install FTDI's virtual COM port drivers. These may be obtained from the FTDI website. (www.ftdichip.com/Drivers/VCP.h) Note: the Windows® operating system generally performs this step automatically the first time the USB on the host is connected to the PC's USB port, so the driver *might* already be installed.

Nexys-3

The LX9 also utilizes a USB UART in a similar fashion. However, the USB UART IC used on the LX9 is the CP2102 from Silicon Labs. Follow the same instructions as above but install the driver from the Silicon Labs website. (http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx)

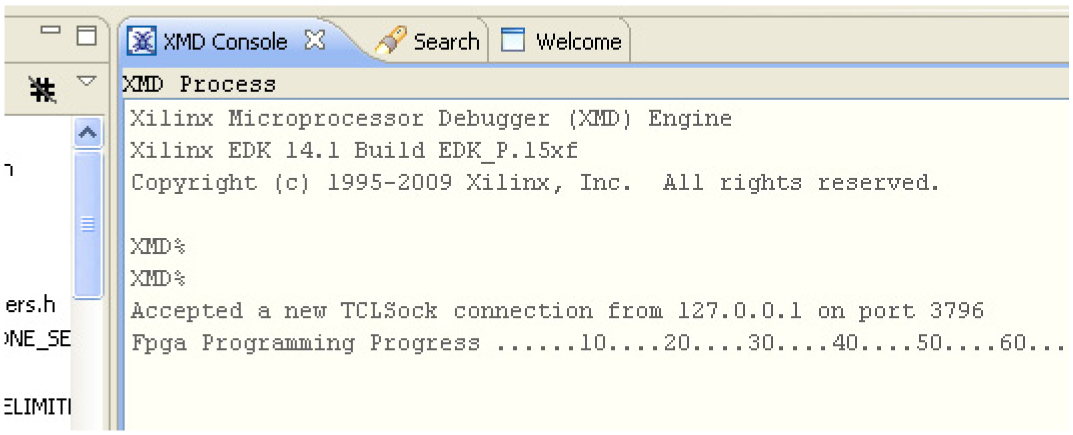
LX9

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.

b. 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/XMD to download and run executable file (.elf) on the Microblaze.

   a. Click the **XMD Console** tab if it isn't already showing, as shown below.

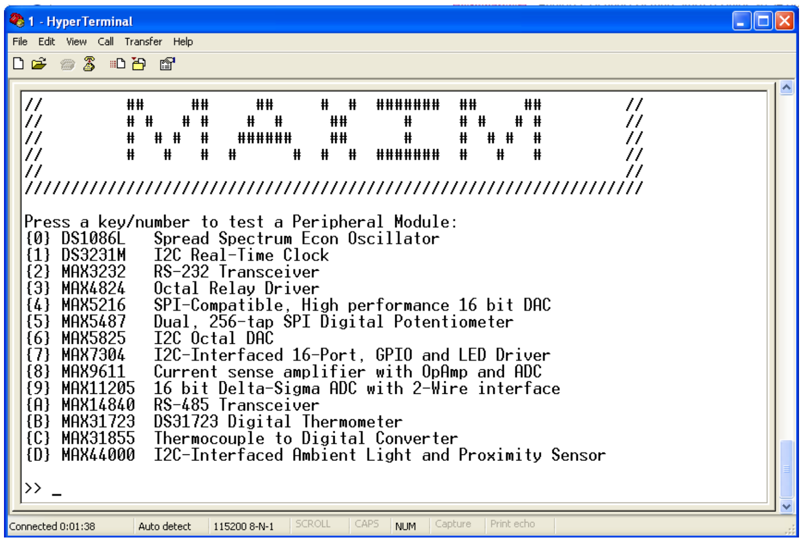


```
XMD Process
Xilinx Microprocessor Debugger (XMD) Engine
Xilinx EDK 14.1 Build EDK_P.15xf
Copyright (c) 1995-2009 Xilinx, Inc.  All rights reserved.

XMD%
XMD%
Accepted a new TCLSock connection from 127.0.0.1 on port 3796
Fpga Programming Progress ......10....20....30....40....50....60...
```

   b. At the **XMD%** prompt at the bottom of the XMD console, issue the following commands.

      1. Use "pwd" and "cd" to navigate to the following directory:
         C:\designs\maxim\v1_4\sdkWorkspace\maximPMOD\Debug
      2. connect mb mdm
      3. dow maximPMOD.elf
      4. run

   c. At this point, the application will be running on the Microblaze and the USB UART/Hyperterminal should show a menu like the one below. This is the top level menu for the module example programs.

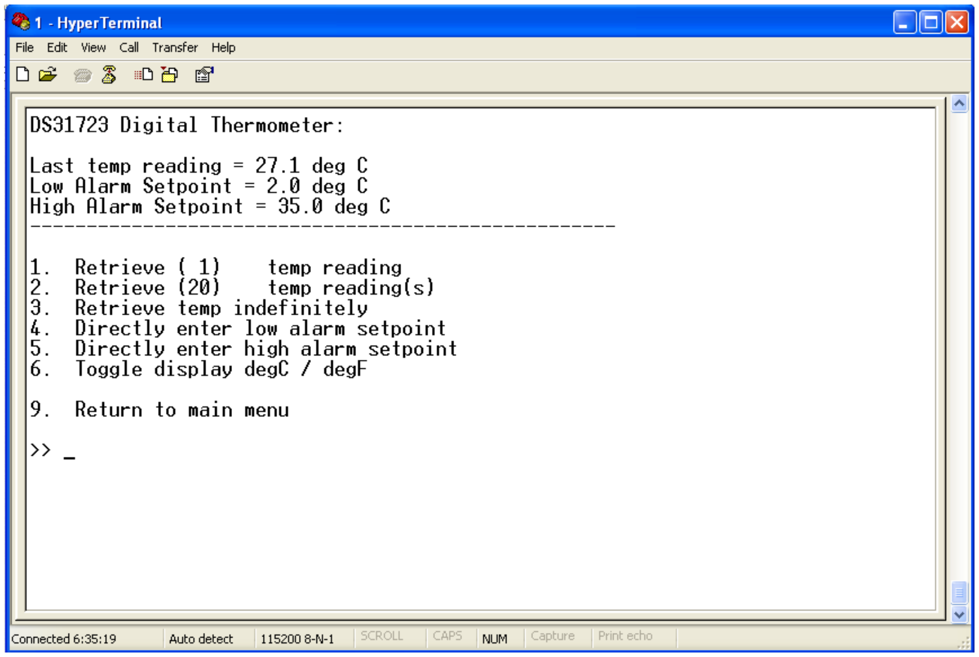


```
//         ##     ##     ##     #  #  #######  ##     ##         //
//         # #   # #    #  #     ##      #     # #   # #         //
//         #  # #  #   ######    ##      #     #  # #  #         //
//         #   #   #  #      #  #  #  #######  #   #   #         //
//                      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

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  DS31723 Digital Thermometer
{C} MAX31855  Thermocouple to Digital Converter
{D} MAX44000  I2C-Interfaced Ambient Light and Proximity Sensor

>> _
```

5. Running the demonstration programs

Peripheral modules can now be placed into the appropriate PMOD port (Ports A..D) and a menu selection made to exercise the module's functionality. For example, place the MAX31723 module into the PMOD B port*. When prompted for a Pmod port, select "**B"**.

The submenu for the MAX31723 peripheral module will appear and 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.

*Note: The Nexys-3 board uses all 4 available Pmod ports which are considered PortA - PortD running left to right across the top of the board. The LX9 board only uses the Pmod port labeled "J5" so the port selection step does not apply when using this board.

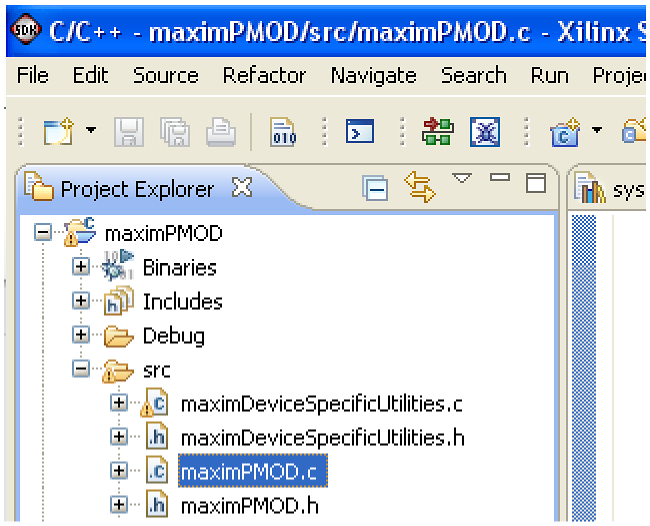
The remaining sections of this document are intended to serve only as brief outlines of and introductions to the steps necessary to modify the hardware and software for this project. Detailed instructions on these procedures and the design tools are beyond the scope of this document.

Modifying the Project C Code

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

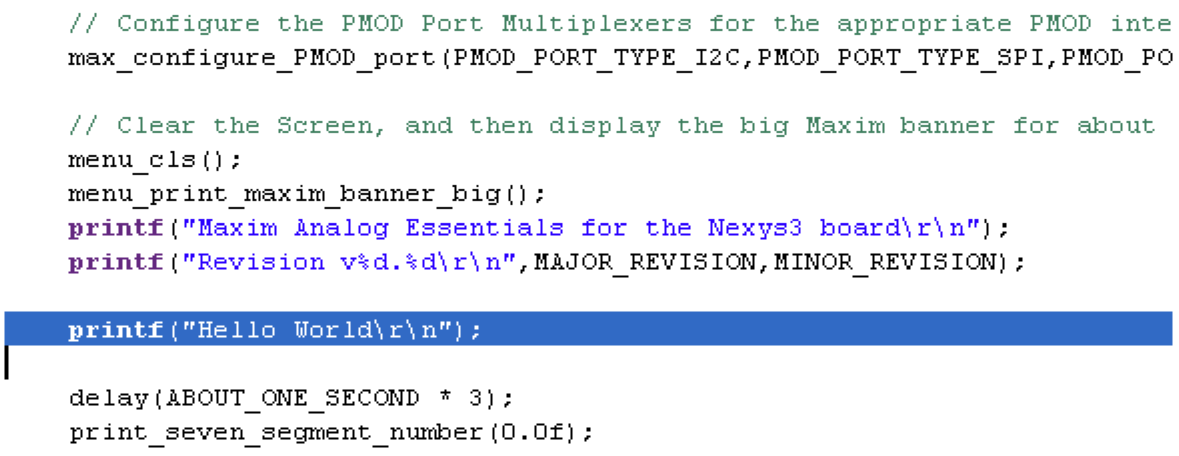
1. Modifying the C code and recompiling

   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

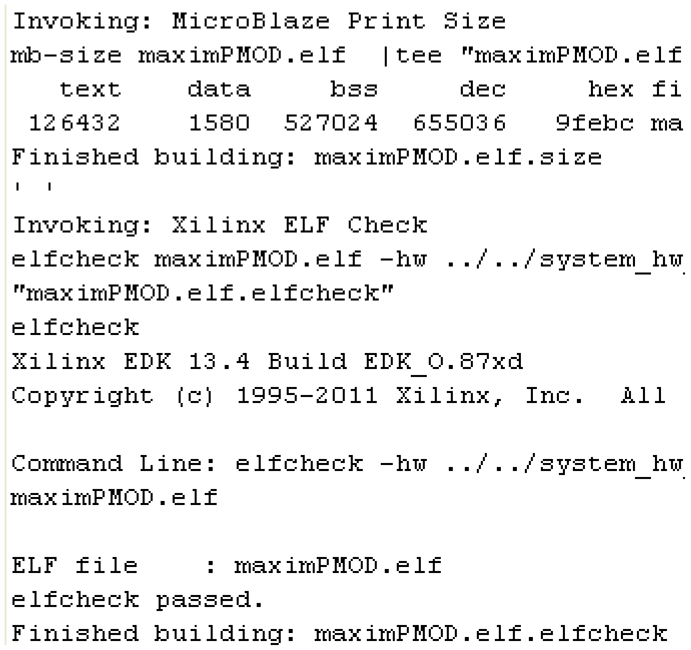
   b. Double-click on maximPMOD.c. This will open the file in the source code editor. The file maximPMOD.c contains the main() function for the demonstration application. The main() function initializes several board peripherals, then prints a menu to the USB UART on the host board. Each menu choice calls 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 appropriate PMOD inte
    max_configure_PMOD_port(PMOD_PORT_TYPE_I2C,PMOD_PORT_TYPE_SPI,PMOD_PO

    // Clear the Screen, and then display the big Maxim banner for about
    menu_cls();
    menu_print_maxim_banner_big();
    printf("Maxim Analog Essentials for the Nexys3 board\r\n");
    printf("Revision v%d.%d\r\n",MAJOR_REVISION,MINOR_REVISION);

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

    delay(ABOUT_ONE_SECOND * 3);
    print_seven_segment_number(0.0f);
```

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.



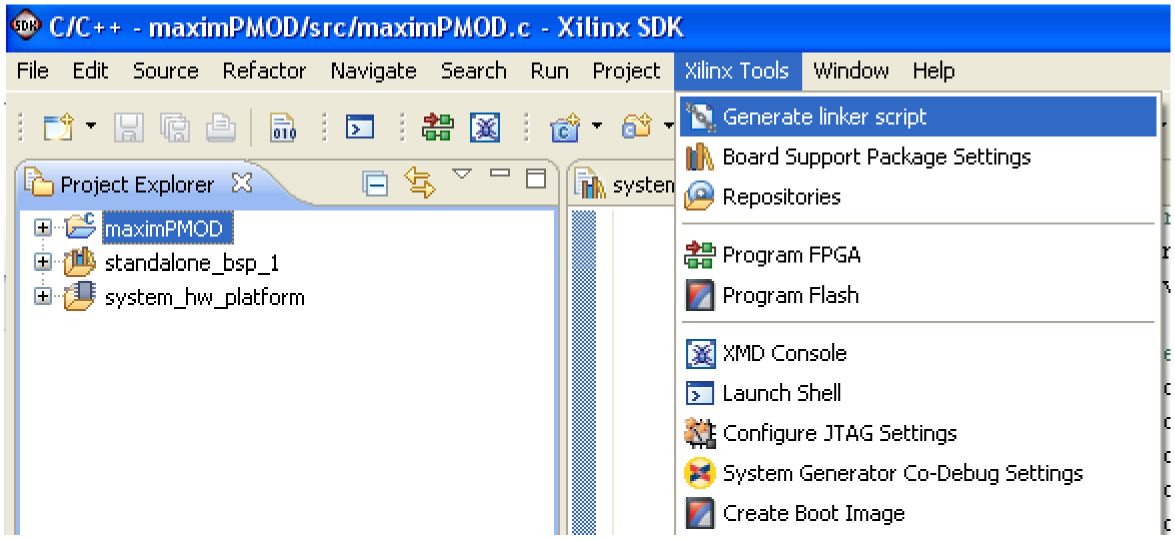
```
Invoking: MicroBlaze Print Size
mb-size maximPMOD.elf  |tee "maximPMOD.elf
   text    data     bss     dec     hex fi
 126432    1580  527024  655036   9febc ma
Finished building: maximPMOD.elf.size

Invoking: Xilinx ELF Check
elfcheck maximPMOD.elf -hw ../../system_hw
"maximPMOD.elf.elfcheck"
elfcheck
Xilinx EDK 13.4 Build EDK_O.87xd
Copyright (c) 1995-2011 Xilinx, Inc.  All

Command Line: elfcheck -hw ../../system_hw
maximPMOD.elf

ELF file    : maximPMOD.elf
elfcheck passed.
Finished building: maximPMOD.elf.elfcheck
```

e. The last two lines are critical. If elfcheck passes, then the (.elf) executable for the Microblaze has been built successfully and is ready for download via the XMD window and “dow” command. Note: if the Microblaze is already running, the XMD command “stop” can be issued to stop the Microblaze. “Run” will then redownload the Microblaze and run the program.

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

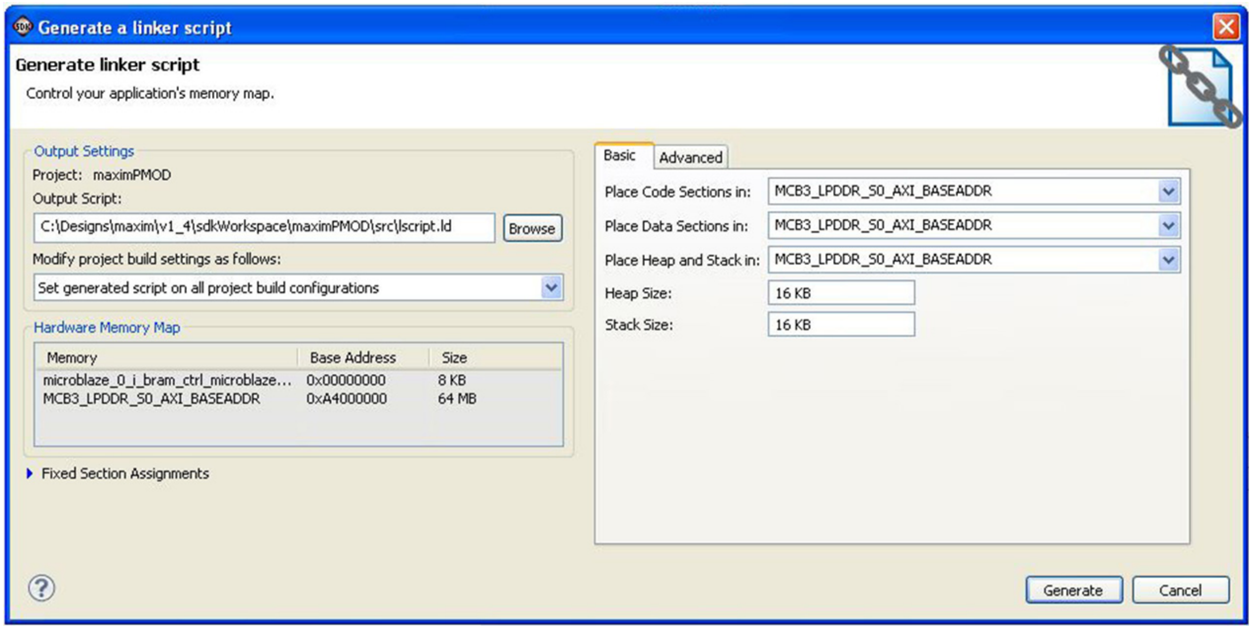
In small software designs, the source code can be merged directly into the FPGA bitstream and downloaded with normal FPGA bitstream tools. However, larger software designs (like this one) that exceed the internal code space must be downloaded to and run from external memory. Using the linker script allows the user to select the target memory location.

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



b. [LX9 version. Skip to step "c" for Nexys-3.] This will bring up the linker script menu as shown below.

LX9


Generate a linker script

Generate linker script

Control your application's memory map.

Output Settings

Project: maximPMOD

Output Script: C:\Designs\maxim\v1_4\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
microblaze_0_i_bram_ctrl_microblaze...	0x00000000	8 KB
MCB3_LPDDR_S0_AXI_BASEADDR	0xA4000000	64 MB

Fixed Section Assignments

Basic | Advanced

Place Code Sections in: MCB3_LPDDR_S0_AXI_BASEADDR

Place Data Sections in: MCB3_LPDDR_S0_AXI_BASEADDR

Place Heap and Stack in: MCB3_LPDDR_S0_AXI_BASEADDR

Heap Size: 16 KB

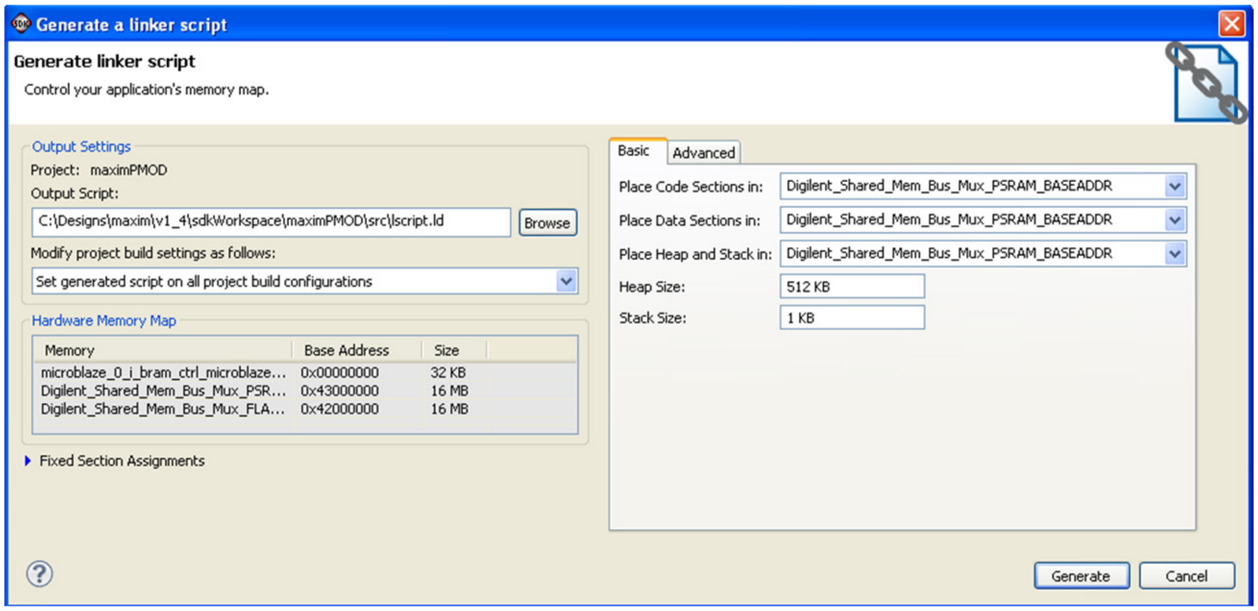
Stack Size: 16 KB

From this window the code, data, heap/stack section placement can be chosen. For the application demonstration, this should be set to MCB3_LPDDR_S0_AXI_BASEADDR.

Skip to step "d".

c. [Nexys-3. See above for LX9] This will bring up the linker script menu as shown below.

Nexys-3


Generate a linker script

Generate linker script

Control your application's memory map.

Output Settings

Project: maximPMOD

Output Script: C:\Designs\maxim\v1_4\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
microblaze_0_i_bram_ctrl_microblaze...	0x00000000	32 KB
Digilent_Shared_Mem_Bus_Mux_PSR...	0x43000000	16 MB
Digilent_Shared_Mem_Bus_Mux_FLA...	0x42000000	16 MB

Fixed Section Assignments

Basic | Advanced

Place Code Sections in: Digilent_Shared_Mem_Bus_Mux_PSRAM_BASEADDR

Place Data Sections in: Digilent_Shared_Mem_Bus_Mux_PSRAM_BASEADDR

Place Heap and Stack in: Digilent_Shared_Mem_Bus_Mux_PSRAM_BASEADDR

Heap Size: 512 KB

Stack Size: 1 KB

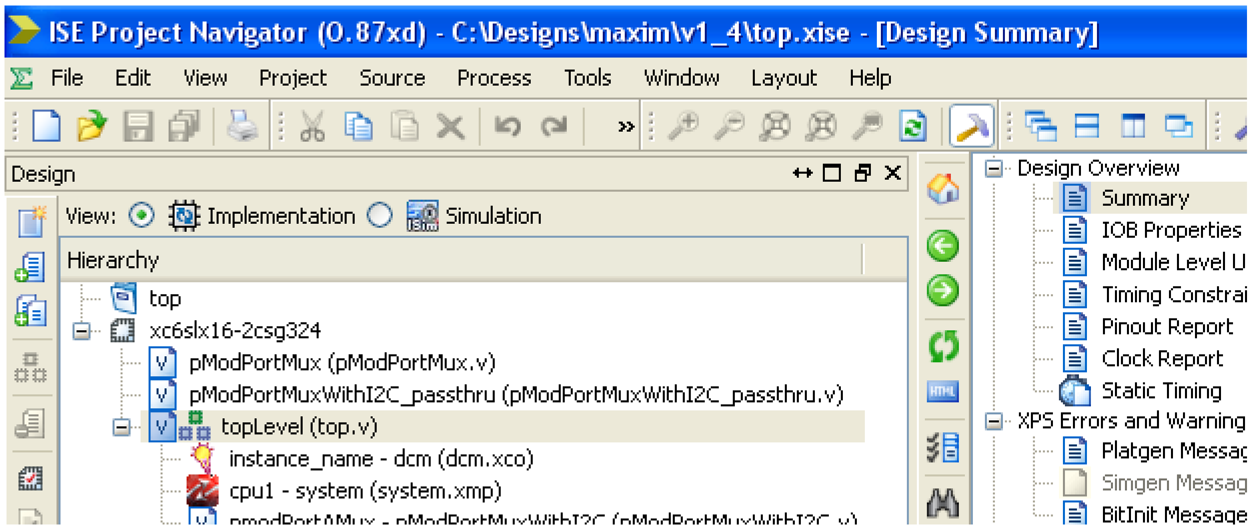
From this window the code, data, heap/stack section placement can be chosen. For the application demonstration, this should be set to Digilent_Shared_Mem_Bus_Max_PSRAM_BASEADDR.

d. 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 13.4

   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**).

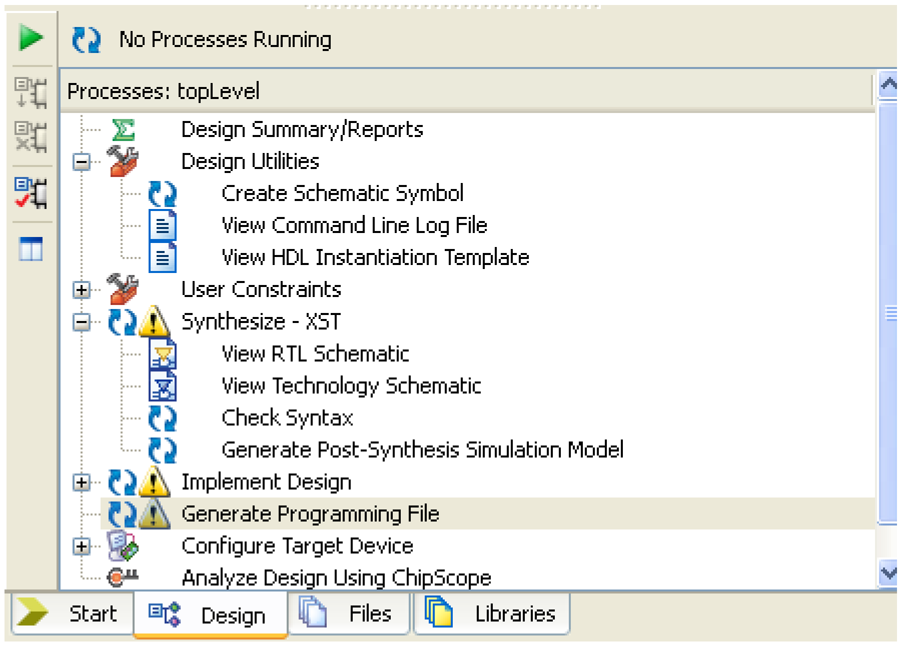


ISE Project Navigator (O.87xd) - C:\Designs\maxim\v1_4\top.xise - [Design Summary]

```
top
xc6slx16-2csg324
    pModPortMux (pModPortMux.v)
    pModPortMuxWithI2C_passthru (pModPortMuxWithI2C_passthru.v)
    topLevel (top.v)
        instance_name - dcm (dcm.xco)
        cpu1 - system (system.xmp)
```

   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.

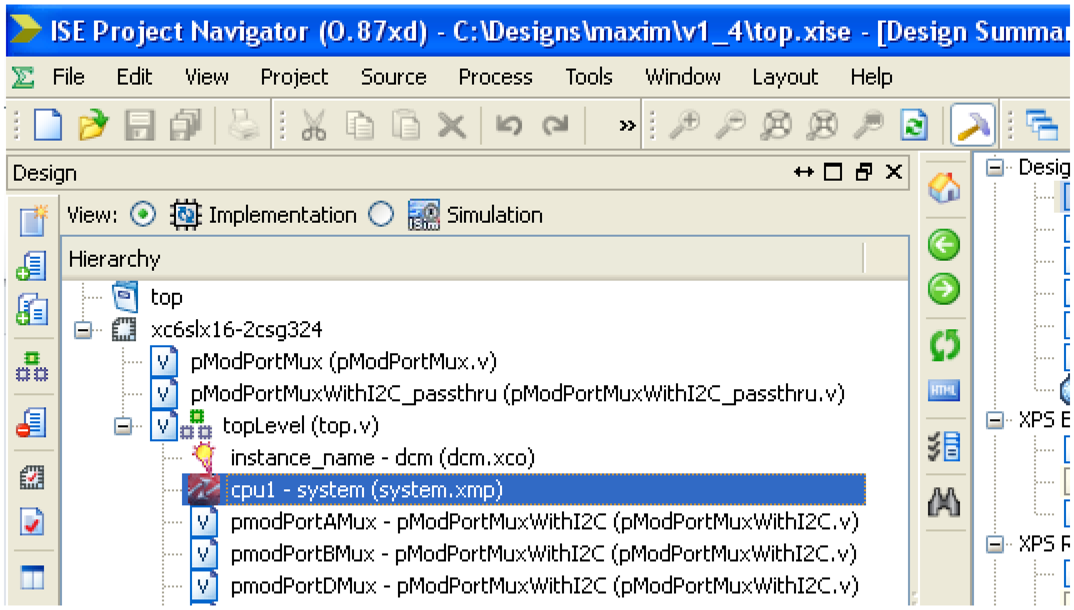

No Processes Running

Processes: topLevel

- Design Summary/Reports
- Design Utilities
  - Create Schematic Symbol
  - View Command Line Log File
  - View HDL Instantiation Template
- User Constraints
- Synthesize - XST
  - View RTL Schematic
  - View Technology Schematic
  - Check Syntax
  - Generate Post-Synthesis Simulation Model
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

2. Open the EDK/XMP project (Microblaze Core/Peripherals)

   a. Highlight and double-click on "cpu1 – system(system.xmp)" as shown below. Changes to the Microblaze core and 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.

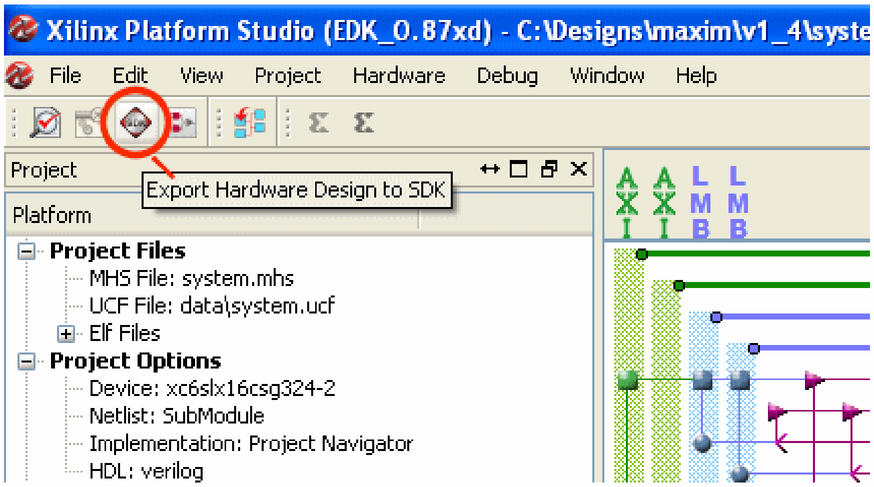


ISE Project Navigator (O.87xd) - C:\Designs\maxim\v1_4\top.xise - [Design Summar

```
top
xc6slx16-2csg324
  pModPortMux (pModPortMux.v)
  pModPortMuxWithI2C_passthru (pModPortMuxWithI2C_passthru.v)
  topLevel (top.v)
    instance_name - dcm (dcm.xco)
    cpu1 - system (system.xmp)
    pmodPortAMux - pModPortMuxWithI2C (pModPortMuxWithI2C.v)
    pmodPortBMux - pModPortMuxWithI2C (pModPortMuxWithI2C.v)
    pmodPortDMux - pModPortMuxWithI2C (pModPortMuxWithI2C.v)
```

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

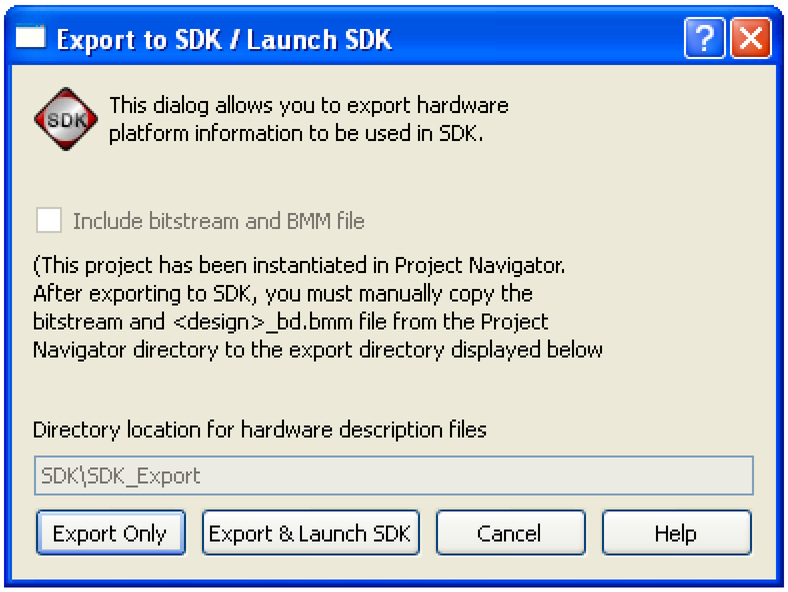
   c. Re-export the design to SDK by clicking on the Export to SDK icon in the upper left menu area.



Xilinx Platform Studio (EDK_O.87xd) - C:\Designs\maxim\v1_4\syste

```
Project Files
  MHS File: system.mhs
  UCF File: data\system.ucf
  Elf Files
Project Options
  Device: xc6slx16csg324-2
  Netlist: SubModule
  Implementation: Project Navigator
  HDL: verilog
```

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.



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

Include bitstream and BMM file

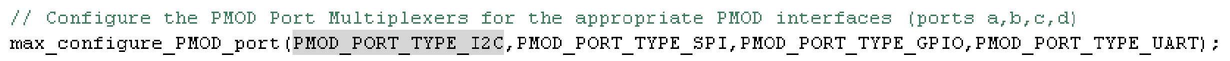
(This project has been instantiated in Project Navigator. After exporting to SDK, you must manually copy the bitstream and <design>_bd.bmm file from the Project Navigator directory to the export directory displayed below

Directory location for hardware description files

SDK\SDK_Export

Appendix A: Project Notes

1. Each active Pmod port has a dedicated I^{2}C port, SPI port UART and octal GPIO. These communications peripherals are multiplexed to the physical ports 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);
   ```

   Also, note that all 4 Pmod ports on the Nexys-3 board are driven in this manner. However, due to gate count limitation in the LX9, only one of its Pmod ports is actively driven by this configuration.

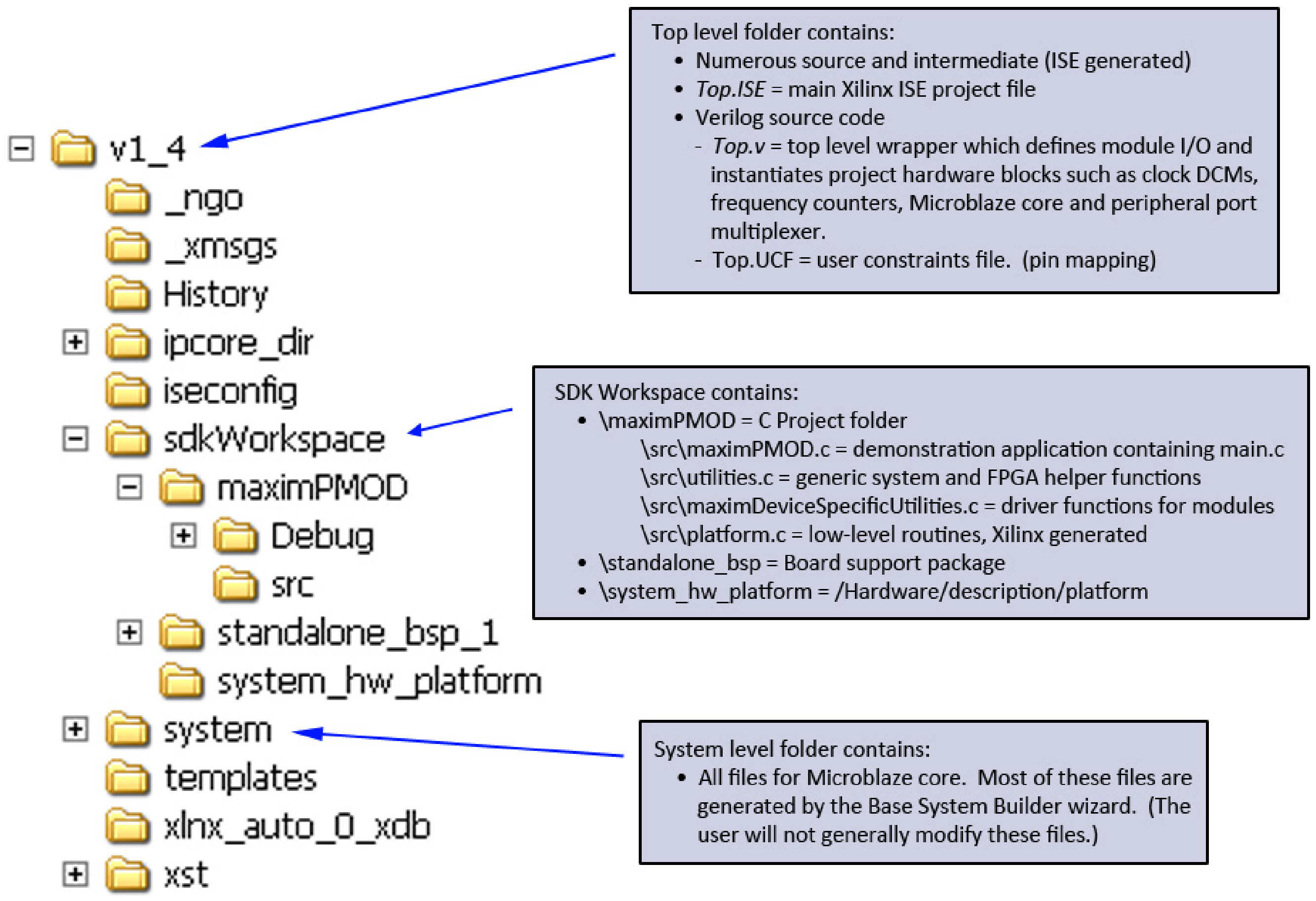
2. The functions in maximDeviceSpecificUtilities.c are used to access the major functionality provided by the Maxim ICs. In general, these functions return an integer value (TRUE/FALSE) that describes whether or not the function successfully completed. Although it is not required to check this return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

3. Functions will typically require a peripheral port address. These ports are assigned sequentially from the sets of I^{2}C, SPI, UART and GPIO peripherals in the design, and are named from ports 0..3. (Again, the Nexys-3 design includes 4 ports and the LX9 design includes one port.) Examples include:
   a. XPAR_IIC_0_BASEADDR, XPAR_IIC_1_BASEADDR, XPAR_IIC_2_BASEADDR, etc.
   b. XPAR_SPI_0_BASEADDR, XPAR_SPI_1_BASEADDR, etc
   c. XPAR_AXI_UARTLITE_0_BASEADDR, XPAR_AXI_UARTLITE_0_BASEADDR, etc

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, it is passed back to the calling function via a pointer.

Appendix B: Project Structure and Key Filenames



Top level folder contains:

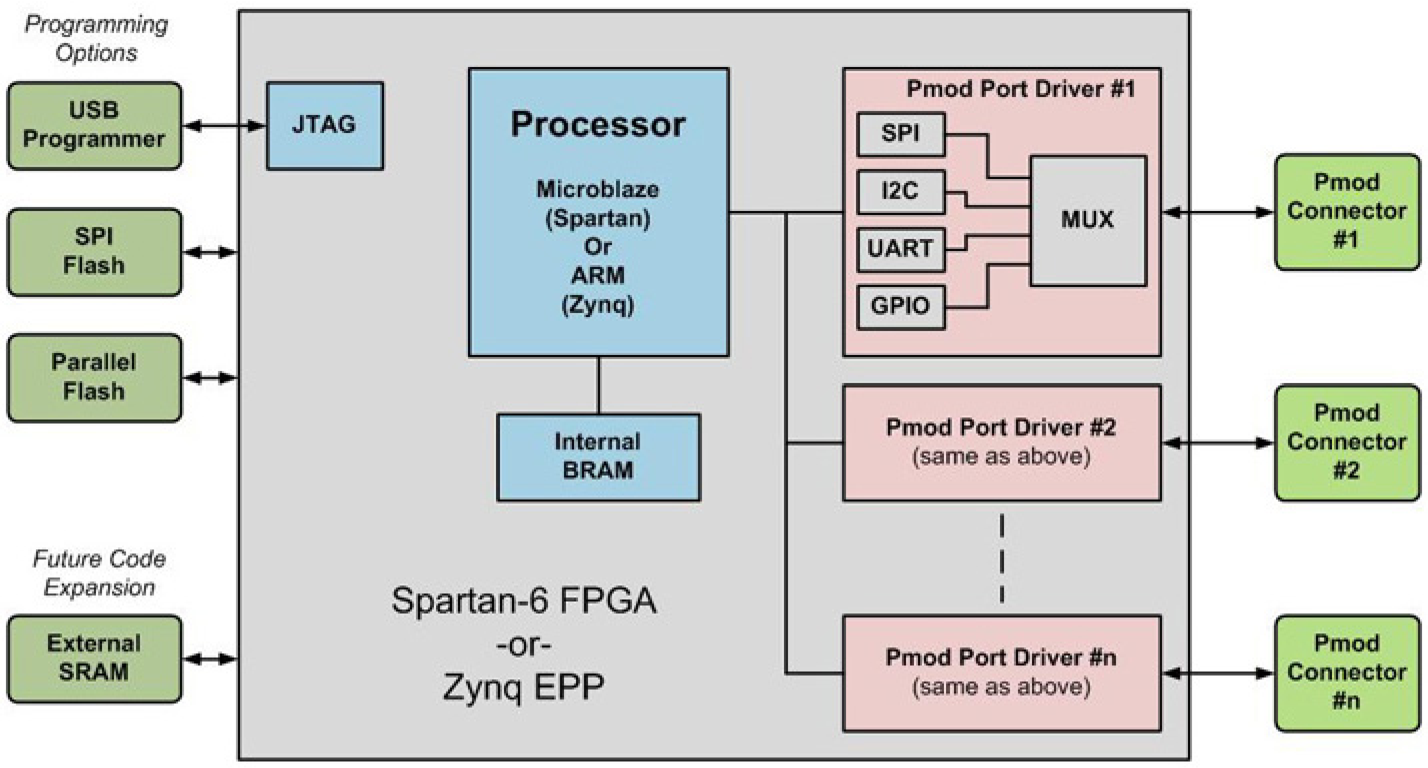
- Numerous source and intermediate (ISE generated)
- *Top.ISE* = main Xilinx ISE project file
- Verilog source code
  - *Top.v* = top level wrapper which defines module I/O and instantiates project hardware blocks such as clock DCMs, frequency counters, Microblaze core and peripheral port multiplexer.
  - 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 Microblaze core. Most of these files are generated by the Base System Builder wizard. (The user will not generally modify these files.)


Programming Options
USB Programmer
SPI Flash
Parallel Flash
Future Code Expansion
External SRAM
JTAG
Processor
Microblaze (Spartan) Or ARM (Zynq)
Internal BRAM
Spartan-6 FPGA -or- Zynq EPP
Pmod Port Driver #1
SPI
I2C
UART
GPIO
MUX
Pmod Port Driver #2 (same as above)
Pmod Port Driver #n (same as above)
Pmod Connector #1
Pmod Connector #2
Pmod Connector #n