Spartan-6 LX9 MicroBoard Embedded Tutorial

Tutorial 1

Creating an Embedded System

Version 12.4.01

Revision History

Version	Description	Date
12.4.01	Initial release for EDK 12.4	3/7/2011

Table of Contents

Table of Figures

Overview

This is the first tutorial in a series of training material dedicated to introducing engineers to creating their first embedded designs. These tutorials will cover all the required steps for creating a complete MicroBlaze design in the Spartan-6 LX9 MicroBoard. While dedicated to this platform, the information learned here can be used with any Xilinx FPGA.

The tutorial is divided into four main steps: creating the hardware platform, understanding what was created, using SDK to compile a software application and testing the system on the FPGA. The test application will reside in Block memory inside the FPGA. Below is a block diagram of the hardware platform. The GPIO core will be used for the LEDs on the board.

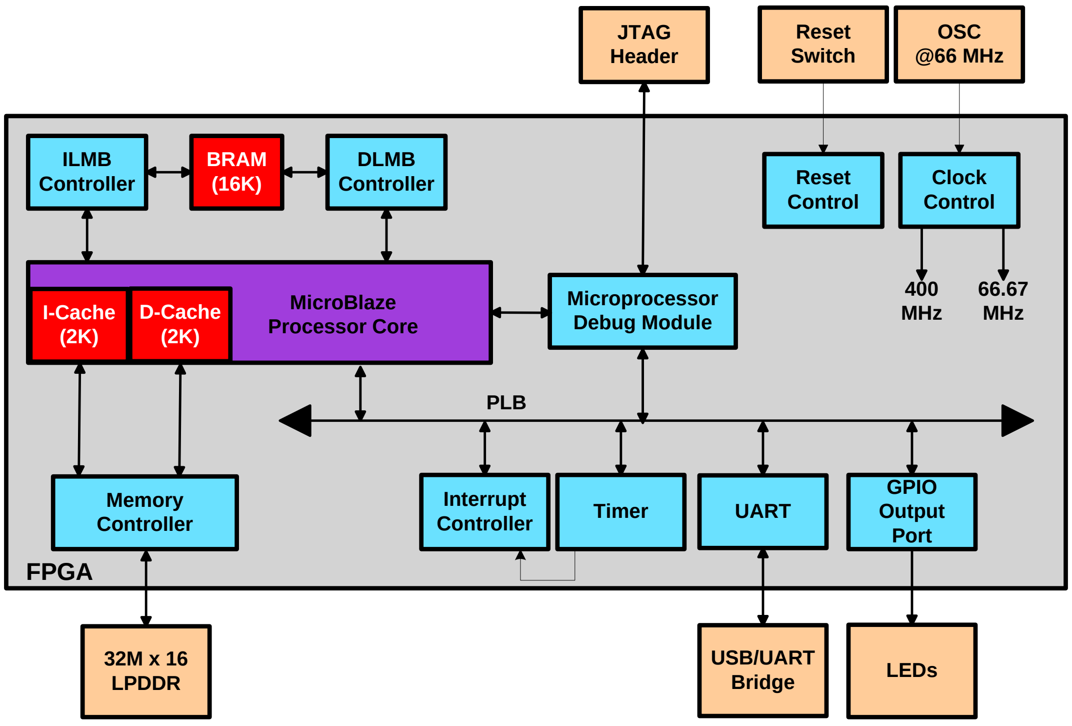


Figure 1 - Hardware Platform

Objectives

This tutorial demonstrates how to create an embedded hardware platform for MicroBlaze using the Xilinx Platform Studio (XPS) Base System Builder (BSB). The lab will show

- How start Base System Builder from Project Navigator
- How to generate a MicroBlaze embedded hardware platform
- What files are generated
- How to use the software environment (SDK) to test a simple application

Requirements

The following items are required for proper completion of this tutorial.

Software

The following software setup was used to test this reference design:

- WindowsXP 32-bit Service Pack 2
- Xilinx ISE WebPack with the SDK add-on or ISE Embedded Edition version 12.4
- Installed Digilent Adept and Xilinx 3rd-party USB Cable driver
- Installed Silicon Labs CP210x USB-to-UART Bridge Driver
- Installation of the Spartan-6 LX9 MicroBoard XBD files

Hardware

The hardware setup used by this reference design includes:

- Computer with a minimum of 300-900 MB (depending on O/S) to complete an XC6SLX9 design[1]
- Avnet Spartan-6 LX9 MicroBoard Kit
  - Avnet Spartan-6 LX9 MicroBoard
  - USB Extension cable (if necessary)
  - USB A-to-MicroB cable

Setup

- Install ISE Embedded Edition or ISE WebPack with the EDK add-on.
- Install Digilent Adept and Xilinx 3rd-party USB Cable driver (see Installation Guide on the DRC)

Recommended Reading

Available from Avnet: http://em.avnet.com/s6microboard

- The hardware used on the Spartan-6 LX9 MicroBoard is described in detail in Avnet document, *Spartan-6 LX9 MicroBoard User Guide*.
- An overview of the configuration options available on the Spartan-6 LX9 MicroBoard, as well as Digilent driver installation instructions can be found in the Avnet document, *Spartan-6 LX9 MicroBoard Configuration Guide.*
- Instructions on installing the Silicon Labs CP210x USB-to-UART drivers can be found in the Avnet document, *Silicon Labs CP210x USB-to-UART Setup Guide*.

Available from Xilinx: http://www.xilinx.com/support/documentation/spartan-6.htm

- Details on the Spartan-6 FPGA family are included in the following Xilinx documents:
  - *Spartan-6 Family Overview (DS160)*
  - *Spartan-6 FPGA Data Sheet (DS162)*
  - *Spartan-6 FPGA Configuration User Guide (UG380)*
  - *Platform Studio Help (available in tool menu)*
  - *Platform Studio SDK Help (available in tool menu)*
  - *MicroBlaze Reference Guide v.12.4 (UG081)*
  - *Embedded System Tools Reference Manual v.12.4 (UG111)*

[1] Refer to www.xilinx.com/ise/products/memory.htm

I. Create the Hardware Platform

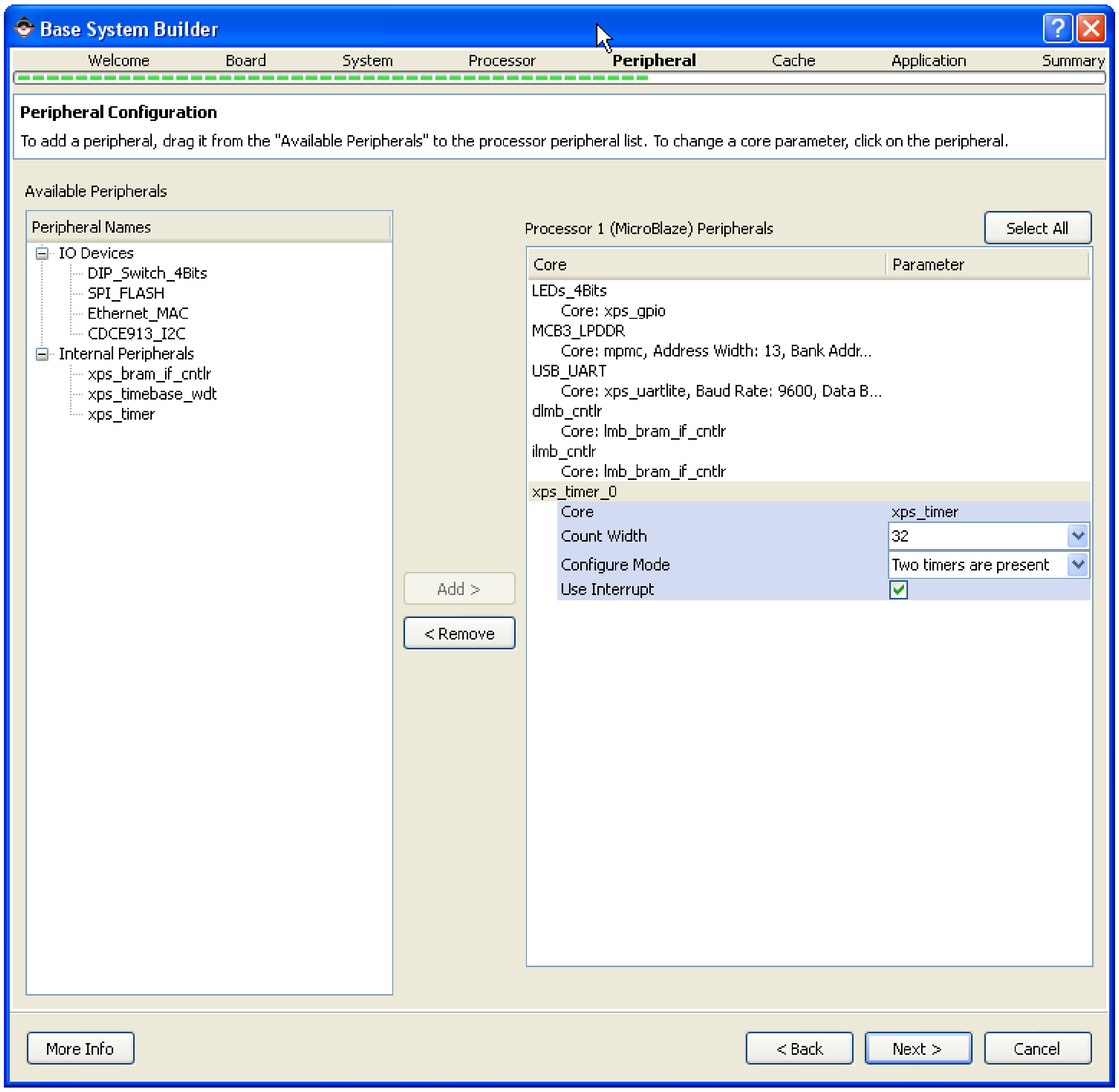
We will use Base System Builder to create the MicroBlaze embedded system. The Wizard can quickly create baseline systems for current development boards or for custom boards. The system will include a timer, a UART, a GPIO controller for the LEDs, external DDR memory, and internal memory to store the application.

1. Start **Xilinx ISE Project Navigator,** ***Start → Programs → Xilinx ISE Design Suite 12.4 → ISE Design Tools → Project Navigator*** and create a new project: ***File > New Project…***
2. Set the **Project Location** to **C:\Xilinx\Embedded\** and the **Project Name** to **Tutorial_01**. Click ***Next****.*
3. Select **Spartan6, XC6SLX9, CSG324, -2**. For this tutorial, pick **VHDL** as the Preferred Language. Click ***Next.***
4. Click ***Finish.***
5. Go to ***Project > New Source…*** then select ***Embedded Processor***. Type ***mb_system*** for the File name. Click ***Next****.* Click ***Finish****.*
6. A message will appear asking if you want to create a Base System using the BSB wizard, click ***Yes****.* **Note**: the **Base System Builder** screen may be minimized to the application tray at the bottom of the screen.
7. Select the **PLB system** (might already be selected) then click ***OK***.
8. In the **Welcome** window click ***Next*** to create a new design.
9. Based on the part selected in Project Navigator, the wizard automatically selected a board with a matching part, the Avnet Spartan-6 LX9 MicroBoard. Select Board Revision **B**, Click ***Next***
10. In the **System Configuration** window click ***Next*** to select a single processor system.
11. Configure the processor:
    - **66.67 MHz** (already selected)
    - **16KB** of Local Memory.

    Click ***Next***

12. Configure the peripherals
    - Remove the **CDCE913_I2C** core
    - Remove the **DIP_Switch_4Bits** core
    - Remove the **Ethernet_MAC** core
    - Remove the **SPI_FLASH** core
    - Select the **xps_timer** core on the left side and **add** it to the system
    - Click on the **Use Interrupt** checkbox. The interrupt controller will be added automatically when the use interrupt option is selected.

    Click ***Next***.



Peripheral Configuration

To add a peripheral, drag it from the "Available Peripherals" to the processor peripheral list. To change a core parameter, click on the peripheral.

Processor 1 (MicroBlaze) Peripherals

Core	Parameter
LEDs_4Bits	
Core: xps_gpio	
MCB3_LPDDR	
Core: mpmc, Address Width: 13, Bank Addr...	
USB_UART	
Core: xps_uartlite, Baud Rate: 9600, Data B...	
dlmb_cntlr	
Core: lmb_bram_if_cntlr	
ilmb_cntlr	
Core: lmb_bram_if_cntlr	
xps_timer_0	
Core	xps_timer
Count Width	32
Configure Mode	Two timers are present
Use Interrupt	✓

Figure 2 - Base System Builder Peripherals

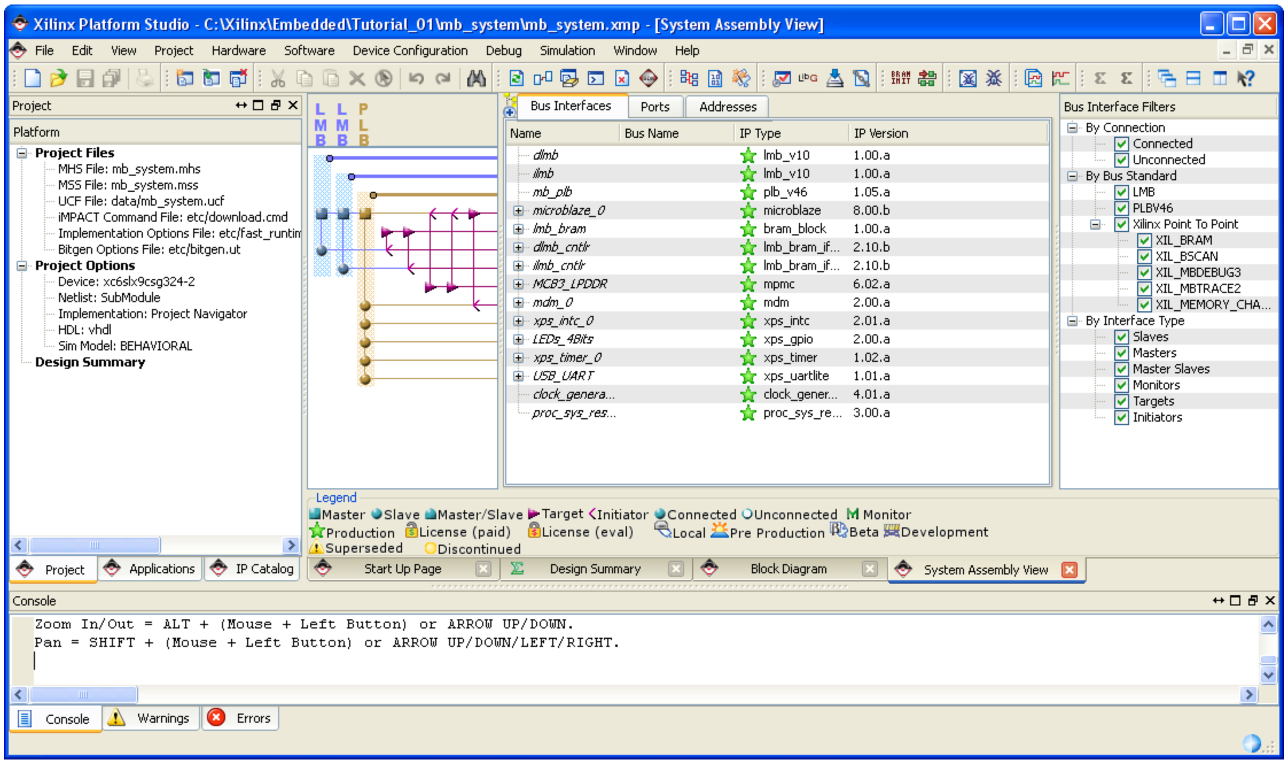
13. In the **Cache Configuration** window, click on the checkboxes for the **Instruction Cache** and the **Data Cache**. Using the drop-down list, select **2 KB** for each. Click ***Next***.
14. In the **Application Configuration Window**, click ***Next***.

15. In the Summary Window, click on the ***Finish*** button to create the project. Click ***OK*** in the constraints pop-up window.

II. Understanding the System

The Base System Builder wizard created all the files needed to get started with an embedded MicroBlaze system. Once finished the Xilinx Platform Studio IDE has all the information for the project. We will look at the interface and the files created. Xilinx Platform Studio provides an integrated environment for creating an embedded hardware platform. We will explore all of the windows and necessary files as shown in the figure below.

.


Xilinx Platform Studio - C:\Xilinx\Embedded\Tutorial_01\mb_system\mb_system.xmp - [System Assembly View]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project
Platform
LMB LMB PLB
Bus Interfaces Ports Addresses
Bus Interface Filters
Legend
Project Applications IP Catalog
Start Up Page Design Summary Block Diagram System Assembly View
Console Warnings Errors

Project Files
- MHS File: mb_system.mhs
- MSS File: mb_system.mss
- UCF File: data/mb_system.ucf
- iMPACT Command File: etc/download.cmd
- Implementation Options File: etc/fast_runtim
- Bitgen Options File: etc/bitgen.ut

Project Options
- Device: xc6slx9csg324-2
- Netlist: SubModule
- Implementation: Project Navigator
- HDL: vhdl
- Sim Model: BEHAVIORAL

Design Summary

Name	Bus Name	IP Type	IP Version
dlmb		lmb_v10	1.00.a
ilmb		lmb_v10	1.00.a
mb_plb		plb_v46	1.05.a
microblaze_0		microblaze	8.00.b
lmb_bram		bram_block	1.00.a
dlmb_cntlr		lmb_bram_if...	2.10.b
ilmb_cntlr		lmb_bram_if...	2.10.b
MCB3_LPDDR		mpmc	6.02.a
mdm_0		mdm	2.00.a
xps_intc_0		xps_intc	2.01.a
LEDs_4Bits		xps_gpio	2.00.a
xps_timer_0		xps_timer	1.02.a
USB_UART		xps_uartlite	1.01.a
clock_genera...		clock_gener...	4.01.a
proc_sys_res...		proc_sys_re...	3.00.a

Bus Interface Filters
- By Connection
  - ✓ Connected
  - ✓ Unconnected
- By Bus Standard
  - ✓ LMB
  - ✓ PLBV46
  - ✓ Xilinx Point To Point
    - ✓ XIL_BRAM
    - ✓ XIL_BSCAN
    - ✓ XIL_MBDEBUG3
    - ✓ XIL_MBTRACE2
    - ✓ XIL_MEMORY_CHA...
- By Interface Type
  - ✓ Slaves
  - ✓ Masters
  - ✓ Master Slaves
  - ✓ Monitors
  - ✓ Targets
  - ✓ Initiators

Legend: Master, Slave, Master/Slave, Target, Initiator, Connected, Unconnected, Monitor, Production, License (paid), License (eval), Local, Pre Production, Beta, Development, Superseded, Discontinued

```
Zoom In/Out = ALT + (Mouse + Left Button) or ARROW UP/DOWN.
Pan = SHIFT + (Mouse + Left Button) or ARROW UP/DOWN/LEFT/RIGHT.
```

Figure 3 - System Assembly View

1. The **System Assembly View** shows each peripheral used and the connections between the peripherals when the **Bus Interface** tab is selected. We will learn more about the System Assembly View as well as the IP Catalog tab in the next tutorial.

2. The **Project** window provides information on the project options used, gives access to the main project files, and log files. The main Project Files are:

   - **MHS File**. The Microprocessor Hardware Specification file contains the hardware specification of the entire system. The MHS file contains the bus architecture, list of peripherals, connectivity for the system, interrupt request priorities, and address space. The MHS file can also call out custom IP.

   - **MSS File**. The Microprocessor Software Specification file defines the software drivers associated with peripherals, standard input/output devices, interrupt handler routines, RTOS and libraries used, and other related software features.

- **UCF File**. The User Constraints File contains the timing and placement constraints for the FPGA.
- **Download.cmd** - iMPACT Command File contains JTAG chain information to download the design to the FPGA on the development board.
- **Fast_runtime.opt** - Implementation Options File lists implementation options for all the phases of the FPGA hardware implementation. The options are used to run the standard ISE tools.
- **Bitgen.ut** - Bitgen Options File provides options when generating the bit file for the FPGA.

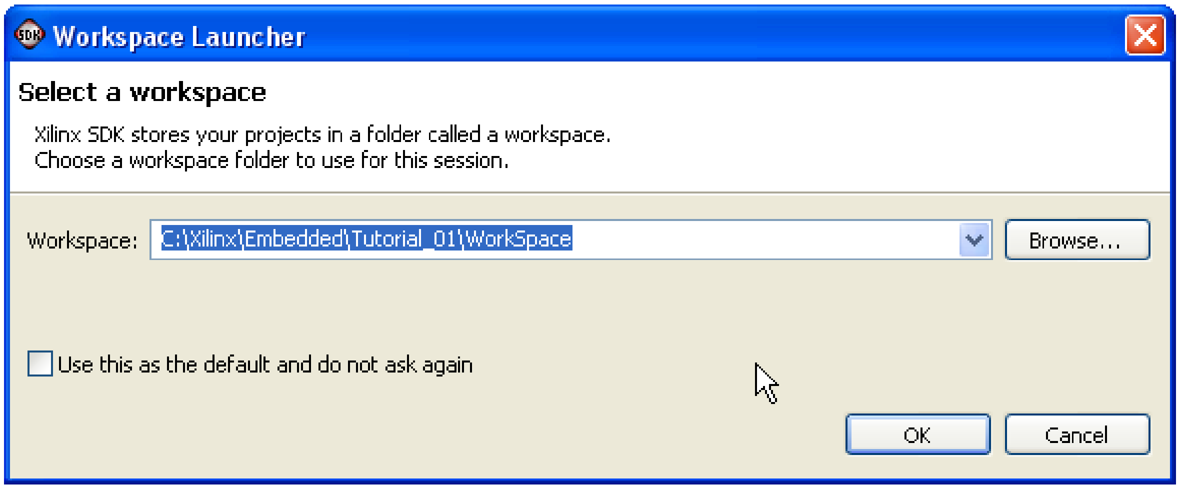
3. The **Applications** tab shows the software applications available for the processor. For this lab, the Xilinx Software Development Kit (SDK) will be used to edit and compile the software.
4. The **Addresses** tab shows the address map for the system.
5. ***Click*** on the **Block Diagram** tab to view the block diagram for the project. The block diagram shows the connections between the different busses and components in the system. Also, you can export this Block Diagram to a jpeg image by clicking ***Project > Generate Block Diagram Image.*** This image gets saved in your under your project in a folder named *blockdiagram*
6. A datasheet of the system can also be generated. Go to ***Project > Generate and View Design Report*** to view the design report. This is an html file that is generated in a *report* subfolder.
7. To view the general project options go to ***Project > Project Options***… Click ***Cancel*** to close the window.
8. To start a software project in SDK, the hardware design information needs to be exported. Go to ***Project > Export Hardware Design to SDK***… Select ***Export Only***.
9. The exported files are located in the *SDK\SDK_Export\hw* directory inside the XPS project directory. We will use the files to create a new C application for our system.
10. ***Close*** XPS when finished. This could take several minutes depending on your PC.
11. Return to Project Navigator and ***Go to Project > Add Copy of Source*** to add the FPGA constraints to the project.
12. Select **mb_system.ucf** in the *mb_system\data* directory. Click **OK**.
13. Select **mb_system** in the hierarchy window.
14. ***Double-click*** on ***Generate Programming File*** to implement the system.

III. Compiling a Test Application Using SDK

Xilinx SDK is an Eclipse based software development environment to create and debug software applications. Features include project management, multiple build configurations, a feature-rich C/C++ code editor, error navigation, a debugging and profiling environment, and source code version control. More time will be spent on the different features of the SDK in the next tutorials. The steps below can be done while Project Navigator is building the hardware.

1. ***Start*** Xilinx SDK. ***Start → Programs → Xilinx ISE Design Suite 12.4 → EDK → Xilinx Software Development Kit.***

2. Create a Workspace named **WorkSpace** in the *Tutorial_01* directory. Click ***OK***.



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:\Xilinx\Embedded\Tutorial_01\WorkSpace

☐ Use this as the default and do not ask again

Figure 4 - Workspace Launcher

3. Close the Welcome window.

4. Go to ***File > New > Xilinx C Project*** to create a new C project.

5. Click on ***Specify*** to indicate the hardware design we will be using. A Xilinx C application needs to be associated with a valid MicroBlaze hardware design.

6. Click on ***Browse*** in the **Target Hardware Specification** section. Browse to the *Tutorial_01\mb_system\SDK\SDK_Export\hw* directory and select the **mb_system.xml** file. Click ***Finish***.

7. Select the **Peripheral Tests** application from the project templates then click ***Next***.

8. Change the Board Support Package project name to **Standalone_BSP**.

9. Click ***Finish***.

10. The Project Explorer View in SDK contains 3 projects: the hardware platform, the Board Support Package, and the C application as shown below.

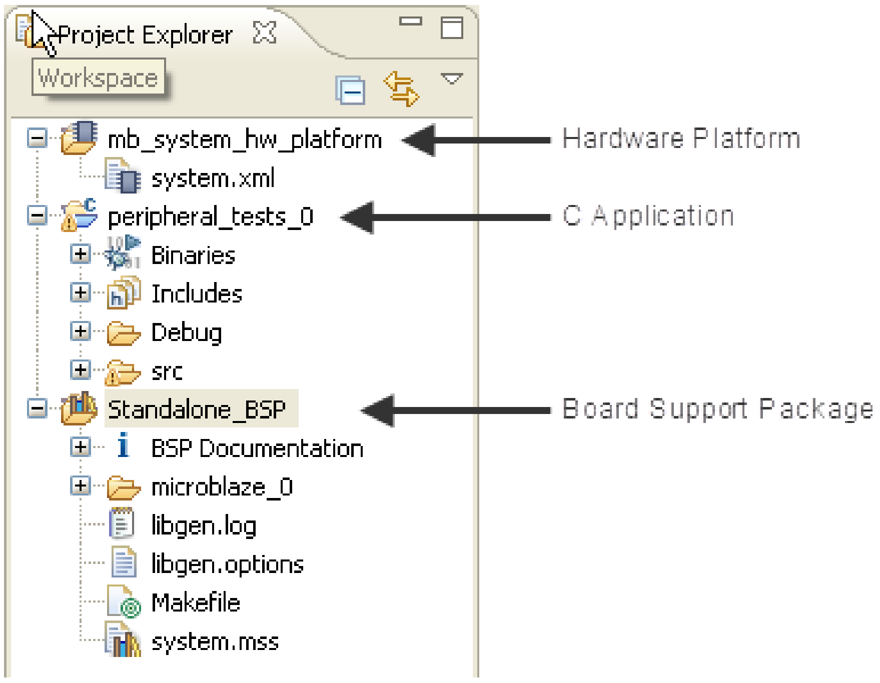


Figure 5 - SDK Project Explorer

11. Within the Board Support Package, all the libraries and include files associated with the hardware project are available to browse. The *microblaze_0\include* folder contains all the header files applicable for the current project.

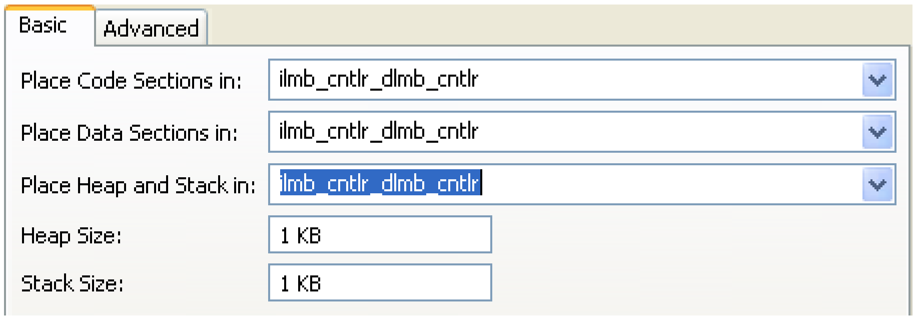
12. The **peripheral_tests_0** project contains C source files to test each peripheral. Expand the project *src* folder to view the sources. The project is compiled automatically after being created.

13. Double click on the **testperiph.c** file to view the main application.

14. To view the project properties right click on the **peripheral_tests_0** project and select ***Properties***. **Expand C/C++ Build** and select ***Settings*** to view all the build options. Click ***Cancel*** to exit.

15. The system contains internal BRAM memory as well as external DDR memory. We can select where the code will be physically located through a linker script. Right-click on the **peripheral_tests_0** project and select ***Generate Linker Script*.**

16. Use the drop-down list to select the internal BRAM memory, **ilmb_cntlr_dlmb_cntlr**, for all the code sections. Click ***Generate*** then ***Yes***.



Place Code Sections in: ilmb_cntlr_dlmb_cntlr
Place Data Sections in: ilmb_cntlr_dlmb_cntlr
Place Heap and Stack in: ilmb_cntlr_dlmb_cntlr
Heap Size: 1 KB
Stack Size: 1 KB

Figure 6 - Linker Script Configuration

17. You can ignore the warning regarding code size. The test application is small enough to fit. Wait for Project Navigator to finish implementing the hardware.

IV. Test the Generated System with the Test Application

The MicroBoard uses a USB-to-UART Bridge controller which is mapped to a COM port on the Windows machine. We will need to check which COM port is being used. The COM port was specified when installing the drivers for the Silicon Labs USB-to-UART Bridge.

1. Plug the MIcroBoard board into the PC. The included USB extension cable can be used if USB access is not convenient.

2. Plug the USB-UART cable between the MicroBoard and the PC.

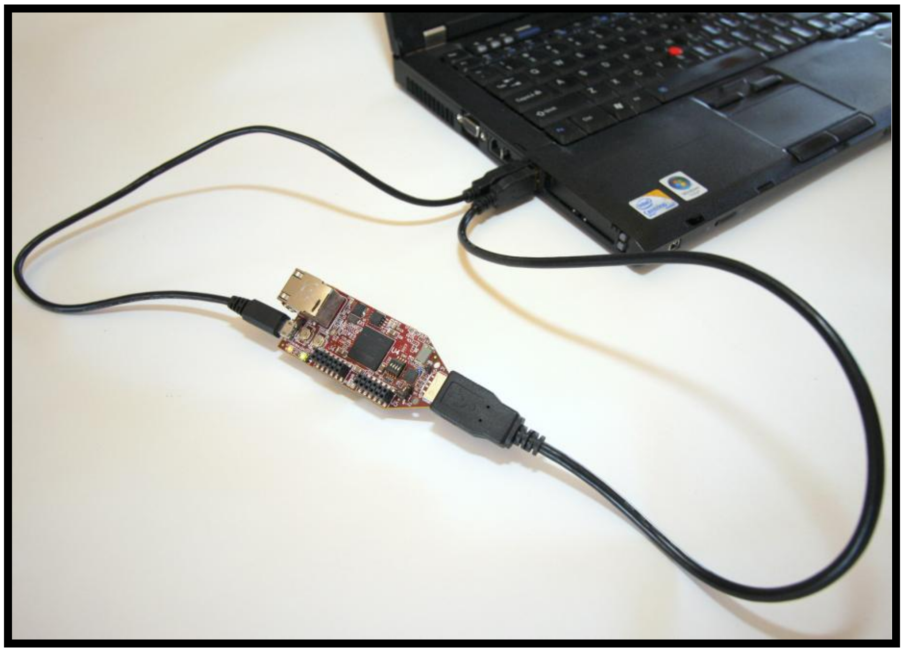
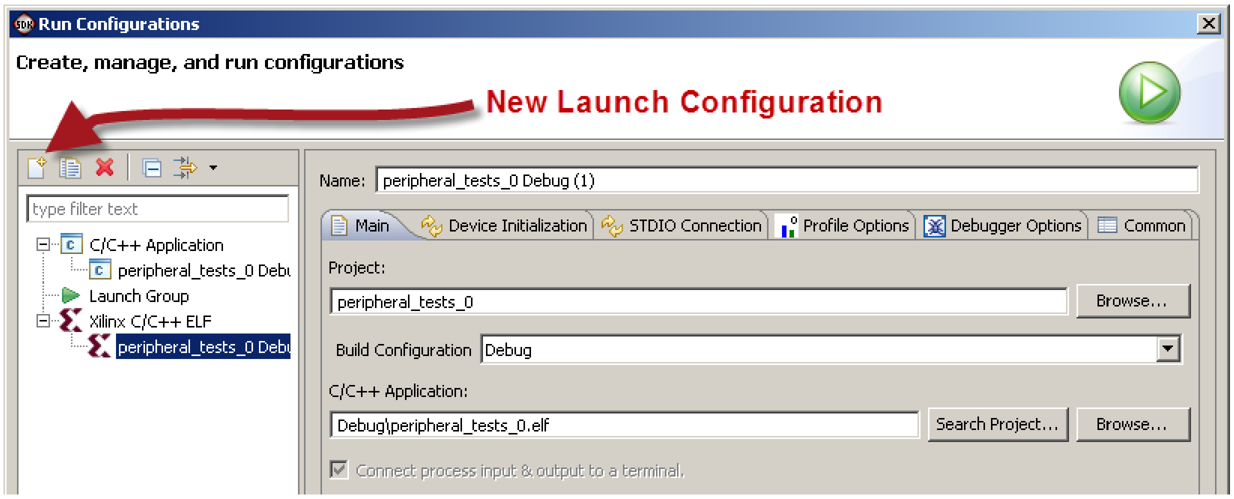


Figure 7 - Connect LX9 MicroBoard to host PC

3. On your desktop, right-click on **My Computer** and select ***Properties***.
4. Select the **Hardware** tab.
5. Click on **Device Manager** and expand the **Ports** option.
6. Write-down the COM port used for the **Silicon Labs USB to UART Bridge**.
7. Close the Device Manager and the System Properties window.
8. In SDK go to ***Xilinx Tools > Configure JTAG Settings***.
    a. For the JTAG Cable, select **3rd Party Cable**, Xilinx Plug-in.
    b. For the Other Options, type **-cable type xilinx_plugin modulename digilent_plugin**
    c. Click ***OK***.
9. In SDK, click on the **Program FPGA** icon
    a. For the Bitstream, browse to the *Tutorial_01* directory and **select mb_system.bit**
    b. For the BMM File, browse to the *Tutorial_01* directory and select **edkBmmFile_bd.bmm**
    c. Click on ***Program***.
10. In the SDK Project Explorer View, right-click on the **peripheral_tests_0** project and select ***Run As > Run Configurations.***
11. Select **Xilinx C/C++ ELF** and click on the ***New Launch Configuration*** icon.



Name: peripheral_tests_0 Debug (1)

Project: peripheral_tests_0

Build Configuration: Debug

C/C++ Application: Debug\peripheral_tests_0.elf

☑ Connect process input & output to a terminal.

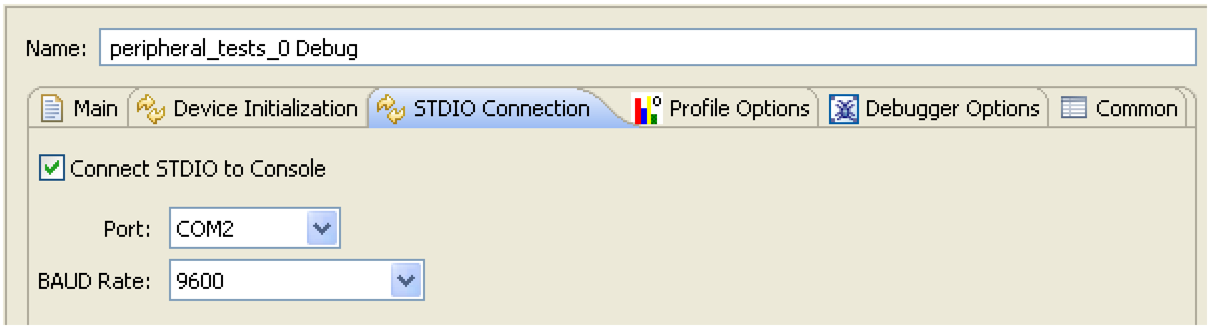
Figure 8 - Run Configurations

The Run Configuration contains settings to run the application on the target board. We will change the default settings to enable STDIO inputs and outputs on the SDK console.

12. In the **SDK Run Configurations** window, select the ***STDIO Connection tab***.

13. Check the **Connect STDIO to Console** box.

14. Select the COM port from step 5 above and leave the BAUD Rate at **9600**.



Name: peripheral_tests_0 Debug

☑ Connect STDIO to Console

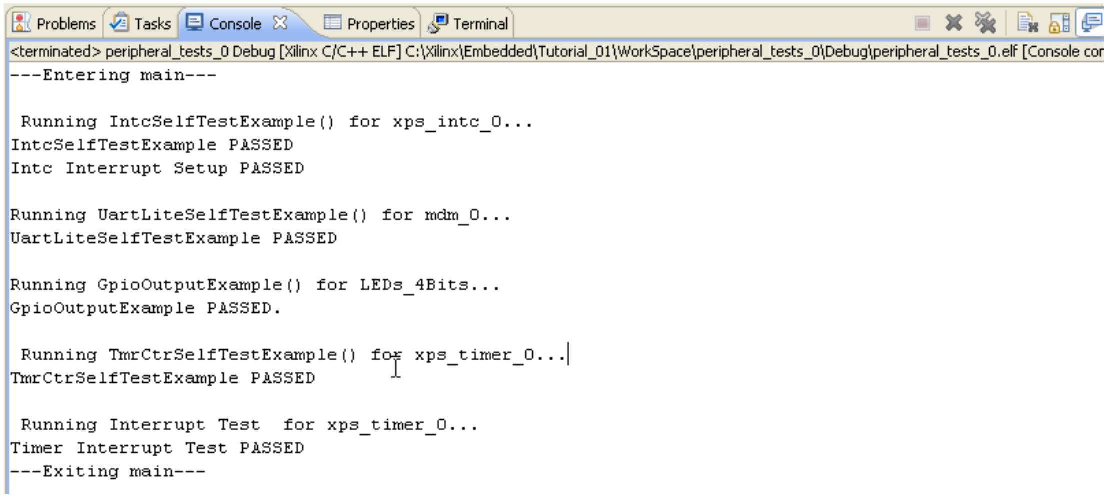
Port: COM2

BAUD Rate: 9600

Figure 9 - STDIO Configuration

15. Click on ***Run***.

16. Verify that the different peripheral print statements appear on the Hyperterminal window and that the LEDs are blinking.



```
<terminated> peripheral_tests_0 Debug [Xilinx C/C++ ELF] C:\Xilinx\Embedded\Tutorial_01\WorkSpace\peripheral_tests_0\Debug\peripheral_tests_0.elf [Console cor
---Entering main---

 Running IntcSelfTestExample() for xps_intc_0...
IntcSelfTestExample PASSED
Intc Interrupt Setup PASSED

Running UartLiteSelfTestExample() for mdm_0...
UartLiteSelfTestExample PASSED

Running GpioOutputExample() for LEDs_4Bits...
GpioOutputExample PASSED.

 Running TmrCtrSelfTestExample() for xps_timer_0...
TmrCtrSelfTestExample PASSED

 Running Interrupt Test  for xps_timer_0...
Timer Interrupt Test PASSED
---Exiting main---
```

Figure 10 - Console View

17. Close SDK. This completes Tutorial #1 of the Spartan-6 LX9 MicroBoard Embedded Tutorial.

Getting Help and Support

Evaluation Kit home page with Documentation and Reference Designs

http://em.avnet.com/s6microboard

Avnet Spartan-6 LX9 MicroBoard forum:

http://community.em.avnet.com/t5/Spartan-6-LX9-MicroBoard/bd-p/Spartan-6LX9MicroBoard

For Xilinx technical support, you may contact your local Avnet/Silica FAE or Xilinx Online Technical Support at www.support.xilinx.com. On this site you will also find the following resources for assistance:

- Software, IP, and Documentation Updates
- Access to Technical Support Web Tools
- Searchable Answer Database with Over 4,000 Solutions
- User Forums
- Training - Select instructor-led classes and recorded e-learning options

Contact Avnet Support for any questions regarding the Spartan-6 LX9 MicroBoard reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design.

- http://www.em.avnet.com/techsupport

You can also contact your local Avnet/Silica FAE.