

# **Spartan-6 LX9 MicroBoard Embedded Tutorial**

## **Tutorial 2**

### **Adding EDK IP to 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

Revision History.....	2
Table of Contents .....	2
Table of Figures.....	2
Overview.....	3
Objectives.....	3
Requirements .....	4
Software.....	4
Hardware .....	4
Setup .....	4
Recommended Reading .....	4
I. Adding the New Peripheral .....	5
II. Writing Code for the New Peripheral .....	8
III. Test the Generated System with the New Application .....	13
Getting Help and Support .....	14

## Table of Figures

Figure 1 - Hardware Platform .....	3
Figure 2 - IP Catalog .....	5
Figure 3 - GPIO Peripheral Configuration.....	5
Figure 4 - Connecting IP to PLB Bus.....	6
Figure 5 - Generate Addresses .....	6
Figure 6 - GPIO I/O Settings.....	7
Figure 7 - External Ports.....	7
Figure 8 - XPS GPIO Registers .....	8
Figure 9 - New C Project .....	9
Figure 10 - New Source File.....	10
Figure 11 - Generate Linker Script .....	11
Figure 12 - Connect LX9 MicroBoard to host PC.....	13

## Overview

This is the second 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 three main steps: adding a new peripheral, writing code for the peripheral and testing the system on the board. The test application will reside in Block memory inside the FPGA. Below is a block diagram of the hardware platform. We will add a GPIO core to make use of the DIP Switches on the board.

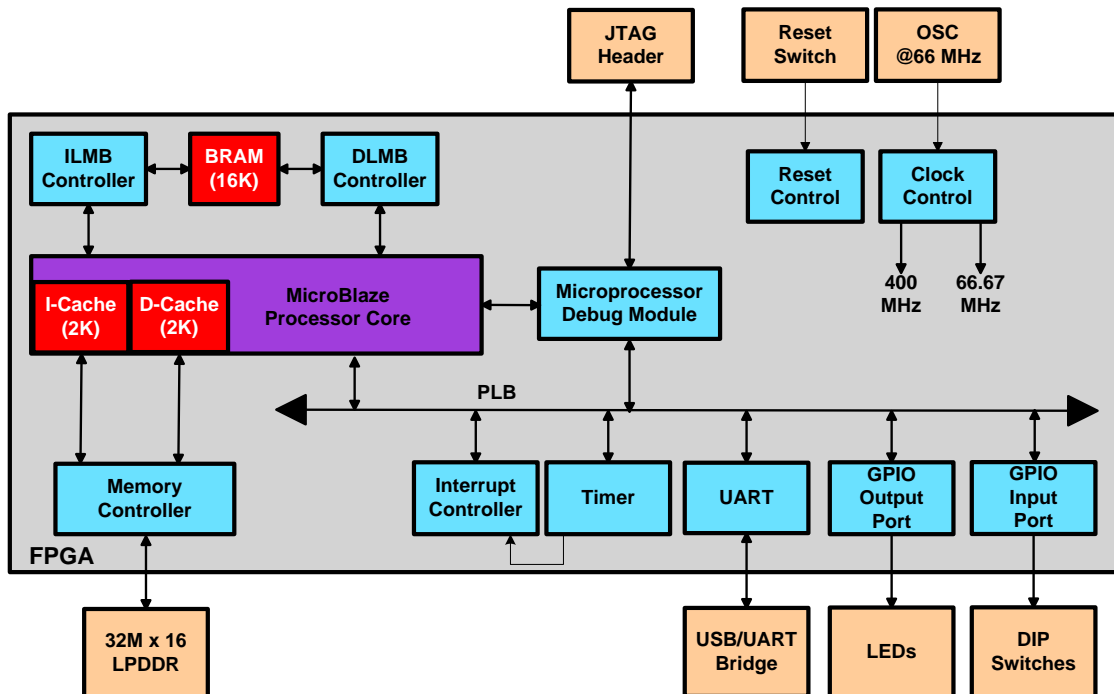


Figure 1 - Hardware Platform

## Objectives

This tutorial demonstrates how to add and modify peripherals to an existing MicroBlaze system using Xilinx Platform Studio (XPS). The system from the previous tutorial will be used as the starting point. The tutorial will show

- How to add an EDK peripheral
- How to connect to the existing system
- How to modify the peripheral options
- How to add constraints for the new peripheral

## Requirements

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

- Completion of the Creating an Embedded System 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 3<sup>rd</sup>-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<sup>1</sup>
- 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 3<sup>rd</sup>-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](#))

---

<sup>1</sup> Refer to [www.xilinx.com/ise/products/memory.htm](http://www.xilinx.com/ise/products/memory.htm)

## I. Adding the New Peripheral

We will use Xilinx Platform Studio (XPS) to add and connect a new peripheral to the existing system.

- 1) Start Project Navigator and open the **Tutorial\_01** project.
- 2) Double-click on the **mb\_system** module to open the system in XPS.
- 3) Select the **IP Catalog** tab in the project window. The IP catalog lists all the processor peripherals available with extended information. Expand the **General Purpose IO** option. The peripherals can be sorted by column field. Right click on the peripheral to view its datasheet or change log information.

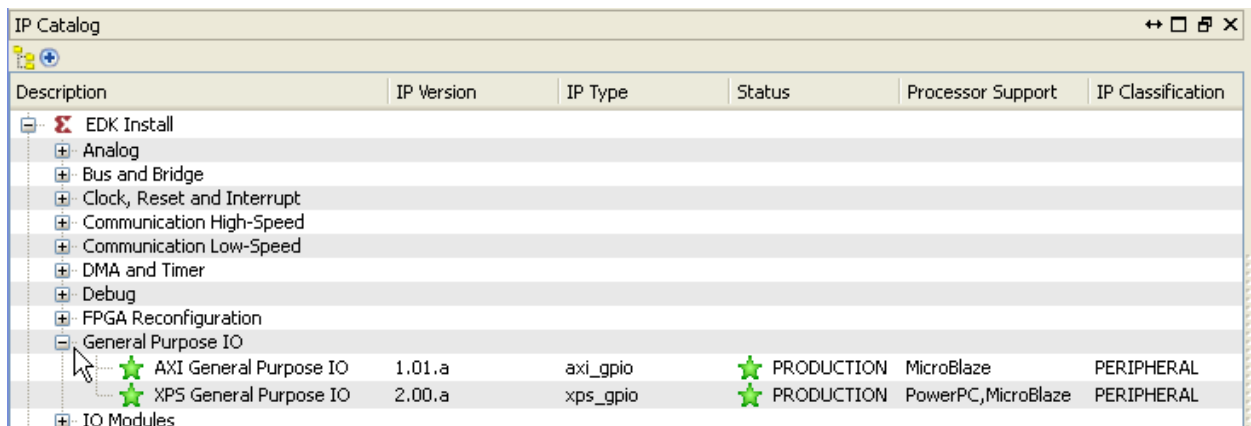


Figure 2 - IP Catalog

- 4) Select **xps\_gpio** version 2.00.a on the list then drag and drop it to the **System Assembly View** window.
- 5) The peripheral configuration window will open to configure the peripheral. Select Channel 1 and Change the **GPIO Data Channel Width** from 32 to 4. Change the **Channel 1 is Input Only** from FALSE to TRUE.

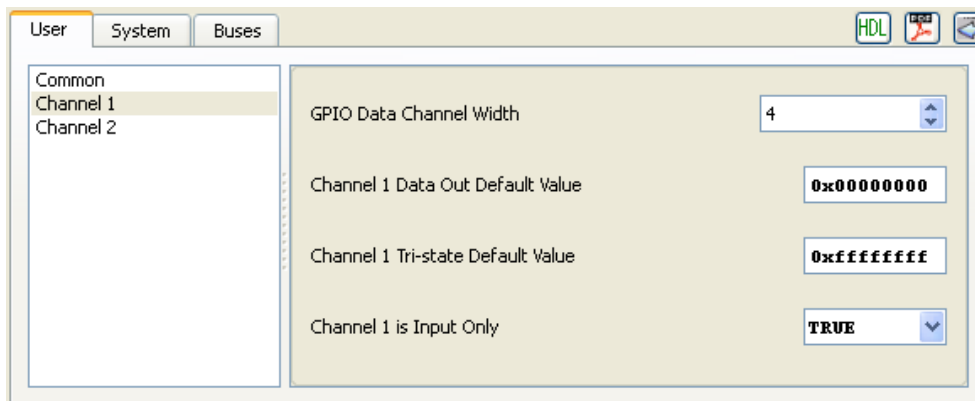


Figure 3 - GPIO Peripheral Configuration

- 6) Click **OK**.

- Click on **xps\_gpio\_0** in the *Name* column of the System Assembly window. Rename the instance to **DIP\_Switches**. To connect the peripheral to the PLB bus, click on the circle on the left of the peripheral name. The circle means that it is a slave on the bus.

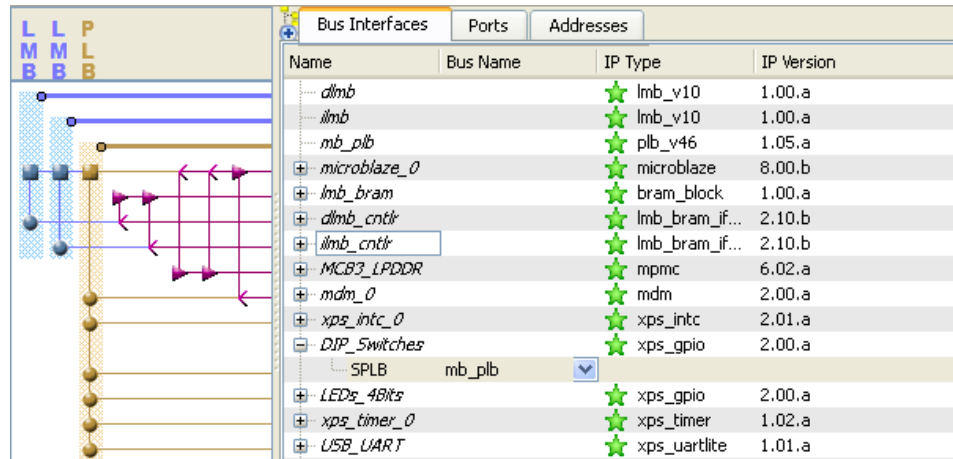


Figure 4 - Connecting IP to PLB Bus

- Click on the **Addresses** tab. The addresses view shows the address space for all the peripherals. The **Lock** box prevents the address for that peripheral from being changed when generating new addresses.
- Click on the **Generate Addresses** button to generate the address range for the new GPIO peripheral.

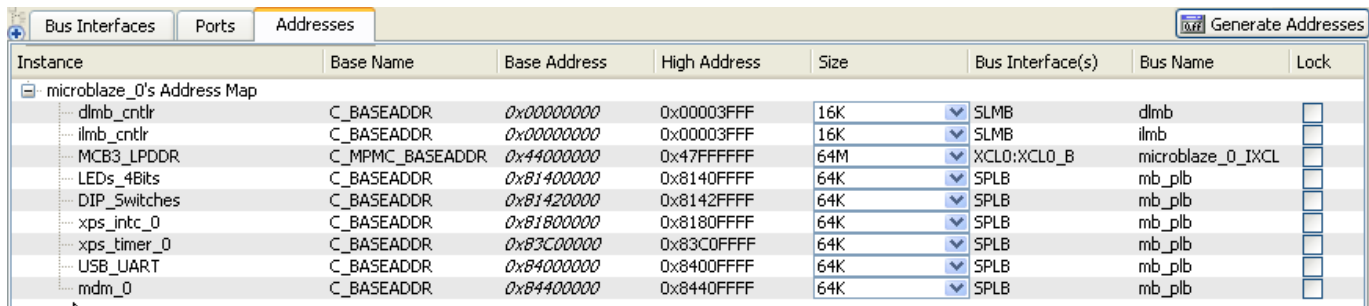


Figure 5 - Generate Addresses

- Click on the **Ports** tab. The Ports view shows the internal connections between the peripherals as well as the external ports connections.

- 11) Expand DIP\_Switches from the list. It will show the connections available for the peripheral.
- 12) Expand the **(IO\_IF) gpio\_0** selection. Look at the GPIO datasheet for a description of each port. The datasheet can be found by right-clicking on DIP\_Switches.
- 13) Select **GPIO\_IO\_I** since the DIP switches are only inputs. Click on **No Connection drop-down list** in the **Net** column. Select **Make External** to add it the external ports list.

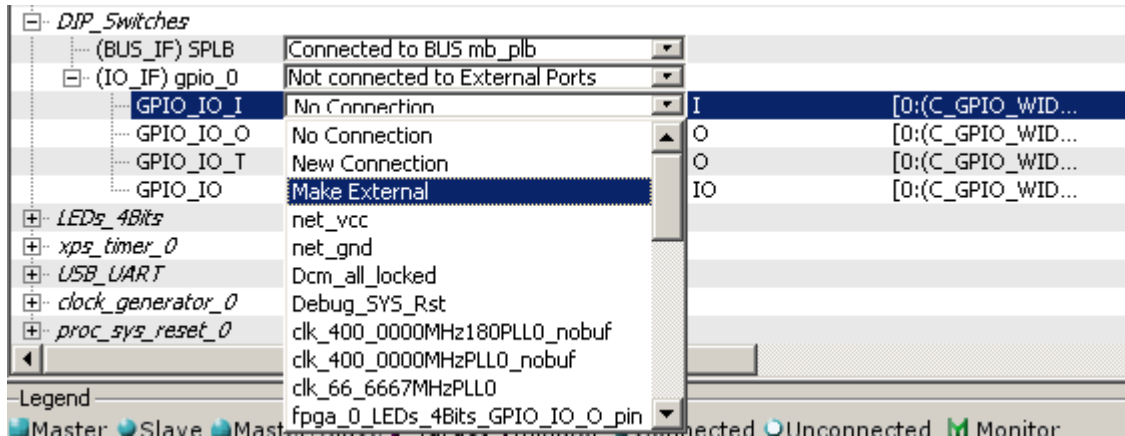


Figure 6 - GPIO I/O Settings

- 14) Expand the **External Ports** to view the new connection. The name of the new external port is **DIP\_Switches\_GPIO\_IO\_I\_pin** with a range of **[0:3]**.

Name	Net	Direction	Range
External Ports			
fpga_0_USB_UART_RX_pin	fpga_0_USB_UART_RX_pin	I	
fpga_0_USB_UART_TX_pin	fpga_0_USB_UART_TX_pin	O	
fpga_0_LEDs_4Bits_GPIO_IO_O_pin	fpga_0_LEDs_4Bits_GPIO_IO_O_pin	O	[0:3]
fpga_0_MCB3_LPDDR_mcbx_dram_addr_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	[12:0]
fpga_0_MCB3_LPDDR_mcbx_dram_ba_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	[1:0]
fpga_0_MCB3_LPDDR_mcbx_dram_ras_n_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	
fpga_0_MCB3_LPDDR_mcbx_dram_cas_n_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	
fpga_0_MCB3_LPDDR_mcbx_dram_we_n_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	
fpga_0_MCB3_LPDDR_mcbx_dram_cke_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	
fpga_0_MCB3_LPDDR_mcbx_dram_clk_n_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	
fpga_0_MCB3_LPDDR_mcbx_dram_dq_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	IO	[15:0]
fpga_0_MCB3_LPDDR_mcbx_dram_dqs_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	IO	
fpga_0_MCB3_LPDDR_mcbx_dram_udqs_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	IO	
fpga_0_MCB3_LPDDR_mcbx_dram_udm_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	
fpga_0_MCB3_LPDDR_mcbx_dram_idm_pin	fpga_0_MCB3_LPDDR_mcbx_dram...	O	
fpga_0_MCB3_LPDDR_rzq_pin	fpga_0_MCB3_LPDDR_rzq_pin	IO	
fpga_0_clk_i_sys_clk_pin	dcm_clk_s	I	
fpga_0_rst_i_sys_rst_pin	sys_rst_s	I	
DIP_Switches_GPIO_IO_I_pin	DIP_Switches_GPIO_IO_I	I	[0:3]

Figure 7 - External Ports

- 15) We need to update the design information for SDK. Go to **Project > Export Hardware Design to SDK...** Select **Export Only**.
- 16) Close XPS.

We need to update the constraint file to add the pinout information for the DIP switches.

- 17) In Project Navigator, expand the **mb\_system** module hierarchy view.
- 18) Select the **mb\_system.ucf** file, expand **User Constraints** in the Processes window and double-click on **Edit Constraints (Text)**. In the UCF file add the lines:

```
NET DIP_Switches_GPIO_IO_I_pin[0] LOC = "B3" | IOSTANDARD = "LVCMOS33";
NET DIP_Switches_GPIO_IO_I_pin[1] LOC = "A3" | IOSTANDARD = "LVCMOS33";
NET DIP_Switches_GPIO_IO_I_pin[2] LOC = "B4" | IOSTANDARD = "LVCMOS33";
NET DIP_Switches_GPIO_IO_I_pin[3] LOC = "A4" | IOSTANDARD = "LVCMOS33";
```

- 19) Save and close the UCF file.
- 20) Select the **mb\_system** module in the **Hierarchy** window.
- 21) Double-Click on **Generate Programming File** to update the bit file with the new peripheral.

## II. Writing Code for the New Peripheral

To test the new peripheral we will create a new software application in Platform Studio SDK and use the GPIO device drivers.

- 1) Start Xilinx SDK and select the Workspace from Tutorial\_01.
- 2) SDK will detect that the hardware system has changed. Click **Yes** to update the hardware platform and BSP.
  - a. If SDK does not auto-detect the new hardware, right-click on the hardware platform project and select **Change Hardware Platform Specification**. Browse to the XML file and click **OK**.
- 3) The peripheral datasheets and address map can be found under the hardware platform. Expand the **mb\_system\_hw\_platform** project and double-click on the **system.xml** file.
- 4) Open the xps\_gpio datasheet to view the GPIO register map. The GPIO Data Register is located at the base address of the peripheral, which is 0x81420000 for the DIP Switches.

Table 4: XPS GPIO Registers

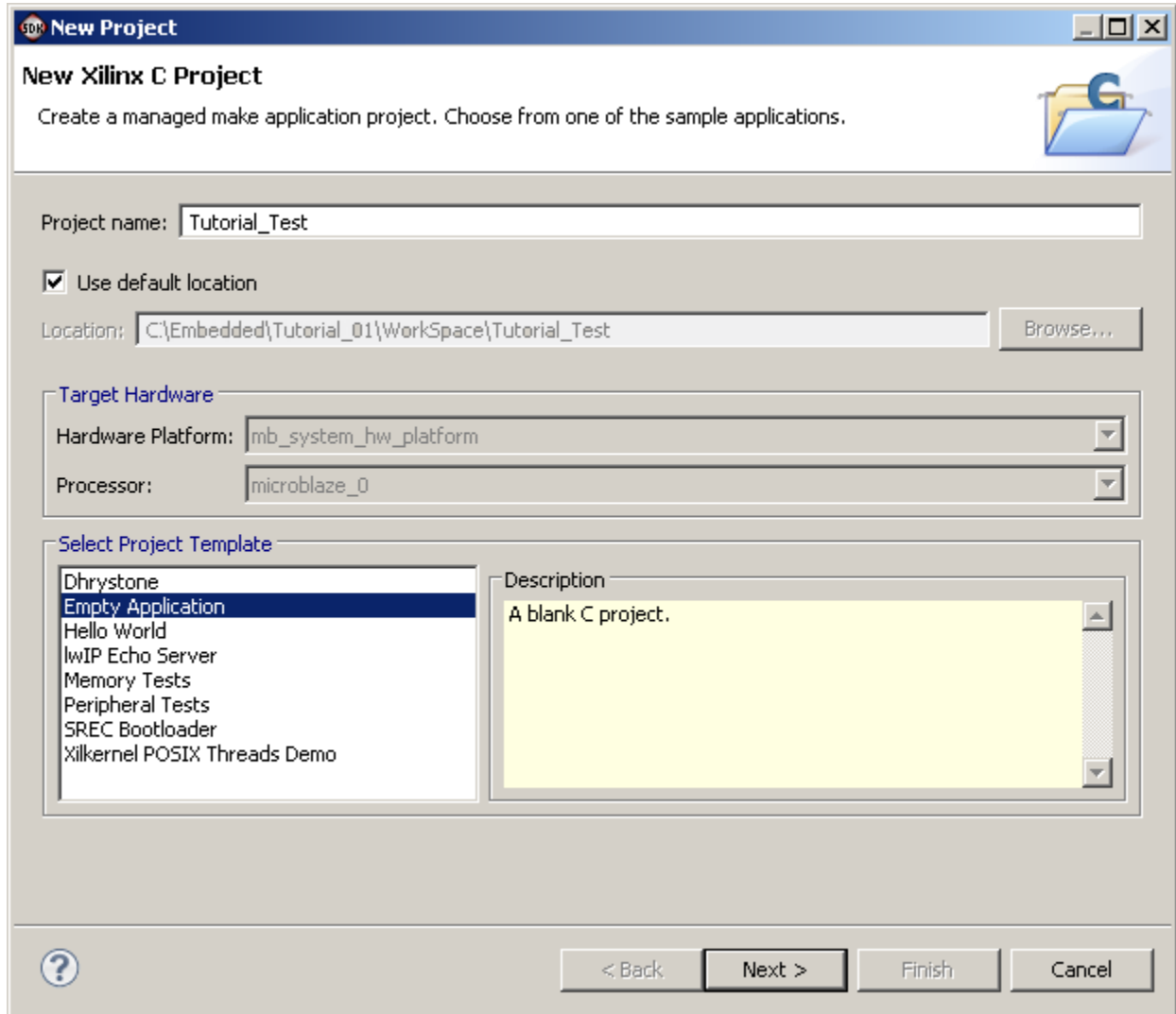
Register Name	Description	PLB Address	Access
GPIO_DATA	Channel 1 XPS GPIO Data Register	C_BASEADDR + 0x00	Read/Write
GPIO_TRI	Channel 1 XPS GPIO 3-state Register	C_BASEADDR + 0x04	Read/Write
GPIO2_DATA	Channel 2 XPS GPIO Data register	C_BASEADDR + 0x08	Read/Write
GPIO2_TRI	Channel 2 XPS GPIO 3-state Register	C_BASEADDR + 0x0C	Read/Write

Figure 8 - XPS GPIO Registers



We will create a new application to test the new peripheral.

- 5) Go to **File > New > Xilinx C Project**.
- 6) Name the project **Tutorial\_Test** and select Empty Application from the project templates. Click **Next**.



**Figure 9 - New C Project**

- 7) Select **Target an Existing Board Support Package** then click **Finish**.

- 8) We need to add a source file for the new empty C project. Select the **Tutorial\_Test\src** folder and go to **File > New > Source File**. Enter **main.c** for the file name. Click **Finish**.

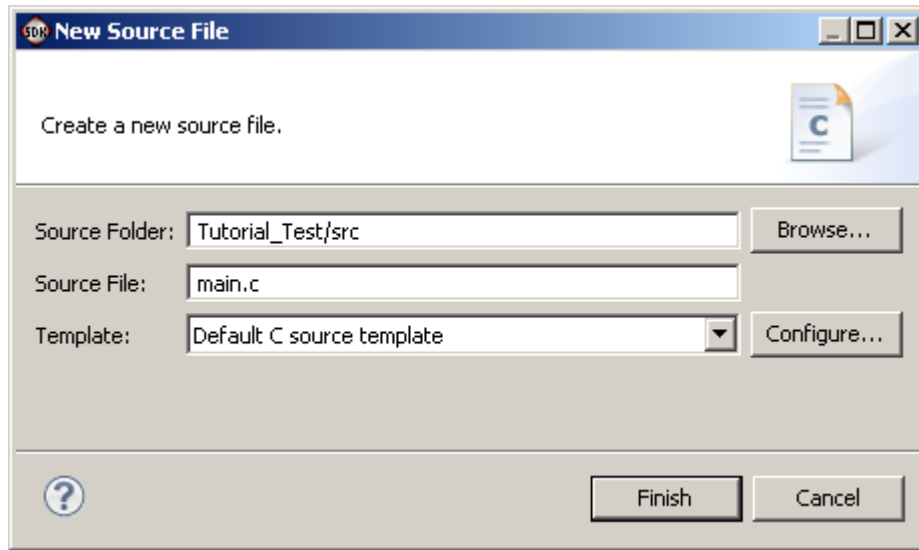


Figure 10 - New Source File

- 9) Inside main.c, after the comments, add:

```
#include "xparameters.h"
#include "stdio.h"
#include "xbasic_types.h"

//=====

int main (void) {

    print("-- Entering main() --\r\n");

    return 0;

}
```

- 10) Save the main.c file. The application will be compiled when saved. The **Project** menu gives options to change the behavior for building the application.

- 11) Create a Linker Script for the new application. Right-click on the **Tutorial\_Test** project and select **Generate Linker Script**. Select **ilmb\_cntlr\_dlmb\_cntlr** for all the code sections to place them in the internal BRAMs. Click on **Generate**. Click **Yes**.

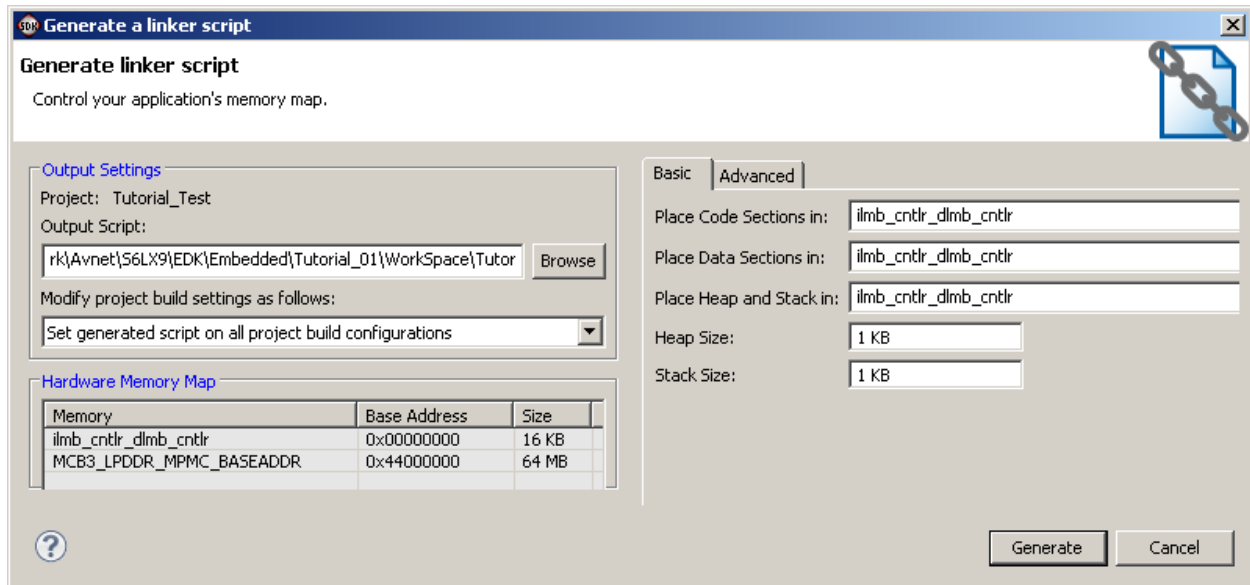


Figure 11 - Generate Linker Script

- 12) We will add code after the print statement to turn-on the LEDs when the DIP switches are asserted.
- 13) On the left side expand *the Standalone\_BSP* project. The **BSP Documentation** section contains the documentation for the device drivers. The **microblaze\_0** folder contains the header files, compiled libraries, and sources for the Board Support Package.
- 14) Expand **microblaze\_0** then expand the **include** directory. Double click on the **xparameters.h** file to view the hardware parameters for the system. Using the macros will isolate the software from the actual hardware.
- 15) Inside the expanded **include** directory for **microblaze\_0** are all the driver header files for the different peripherals. The **\_l.h** denotes a low level driver. Double-click on **xgpio\_l.h** to view the GPIO low level functions.
- 16) Click on **XGpio\_ReadReg** in the **Outline** window to view the format to read the GPIO registers. We will also use the function **XGpio\_WriteReg** to write to the LEDs. The Base Address for the device can be found in the **xparameters.h** file. The Data Register has an offset of 0x00.
- 17) Write code into **main.c** to read from the DIP Switches GPIO channel 1:

- a. Include the low level driver header for the GPIO after the other include statements

```
#include "xgpio_l.h"
```

- b. Declare a new global variable (u32 is defined in xbasic\_types.h) before *int main (void) {*

```
u32 DIP_Read;
```

- c. Add code to read from the DIP switches after the print statement:

```
while (1) {  
    DIP_Read = XGpio_ReadReg(XPAR_DIP_SWITCHES_BASEADDR, 0);  
}
```

- d. Add code to write the DIP switch value to the LEDs. Only add the WriteReg line inside the while loop. The final while loop should look like this:

```
while (1) {  
    DIP_Read = XGpio_ReadReg(XPAR_DIP_SWITCHES_BASEADDR, 0);  
  
    XGpio_WriteReg(XPAR_LEDS_4BITS_BASEADDR, 0, DIP_Read);  
  
}
```

18) Save and close the file. The application will be compiled automatically.

19) To view the changes made to main.c, right-click on **main.c** in the **Project Explorer** window and select **Compare > With Local History...** Click **OK** when finished.

The application is ready to be tested on the board.

### III. Test the Generated System with the New Application

- 1) Plug the MicroBoard board into the PC.
- 2) Plug the USB UART cable between the MicroBoard and the PC.

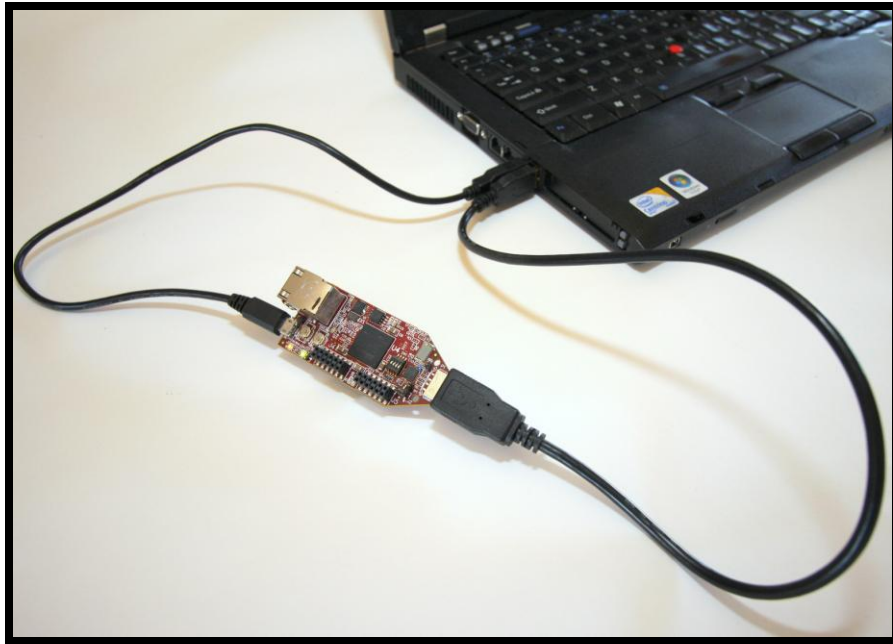




Figure 12 - Connect LX9 MicroBoard to host PC

- 3) 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**.
- 4) In the SDK Project Explorer View, right-click on the **Tutorial\_Test** project and select **Run As > Run Configurations...**
- 5) Select **Xilinx C/C++ ELF** and click on the **New Launch Configuration** icon 
- 6) In the SDK **Run Configurations** window, select the **STDIO Connection** tab.
- 7) Check the **Connect STDIO to Console** box.
  - a. Select the COM port for the USB UART and leave the BAUD Rate at 9600.
  - b. Click on **Run**.
- 8) Carefully modify the DIP switches positions to turn the LEDs on and off.
- 9) Close SDK. This concludes Tutorial #2.

## 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](http://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.