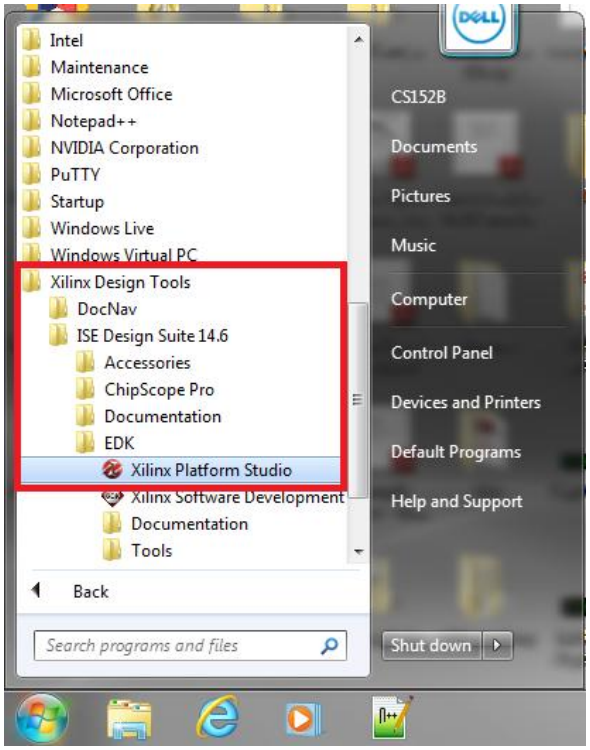
Paaras Chand

MicroBlaze and iRobot Tutorial

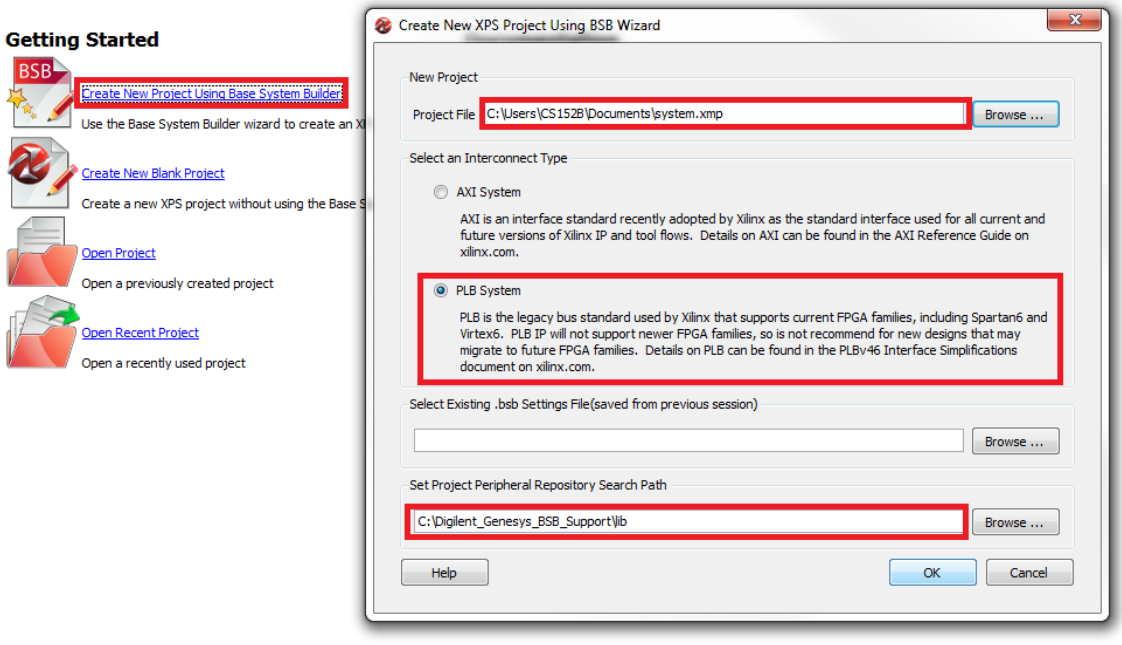
Setting up the EDK and SDK

1. Launch the Xilinx Platform Studio

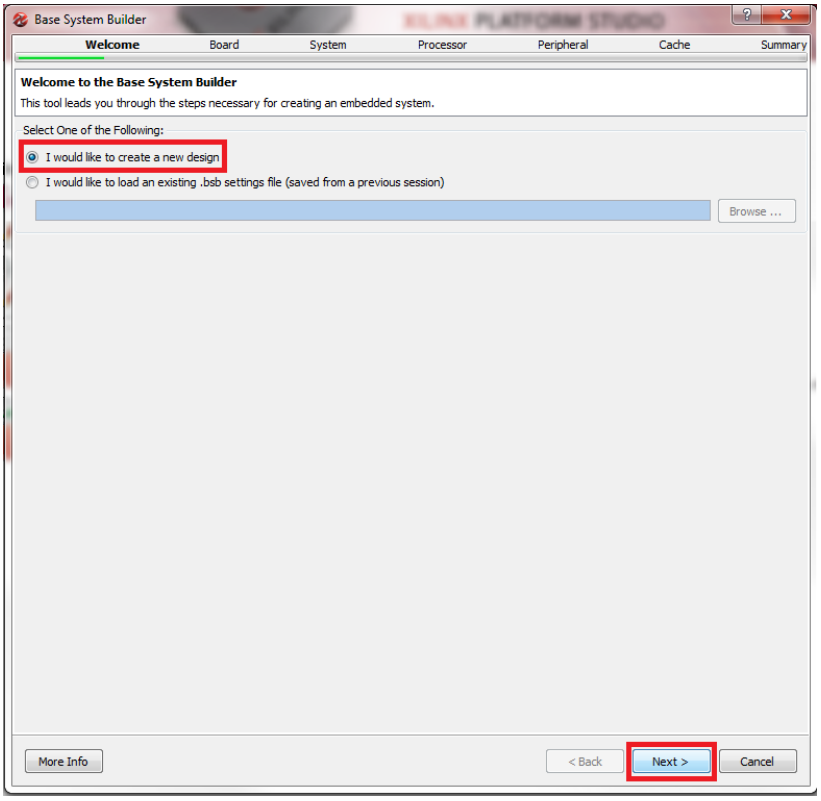


2. Select the "Create New Project Using Base System Builder" Option.

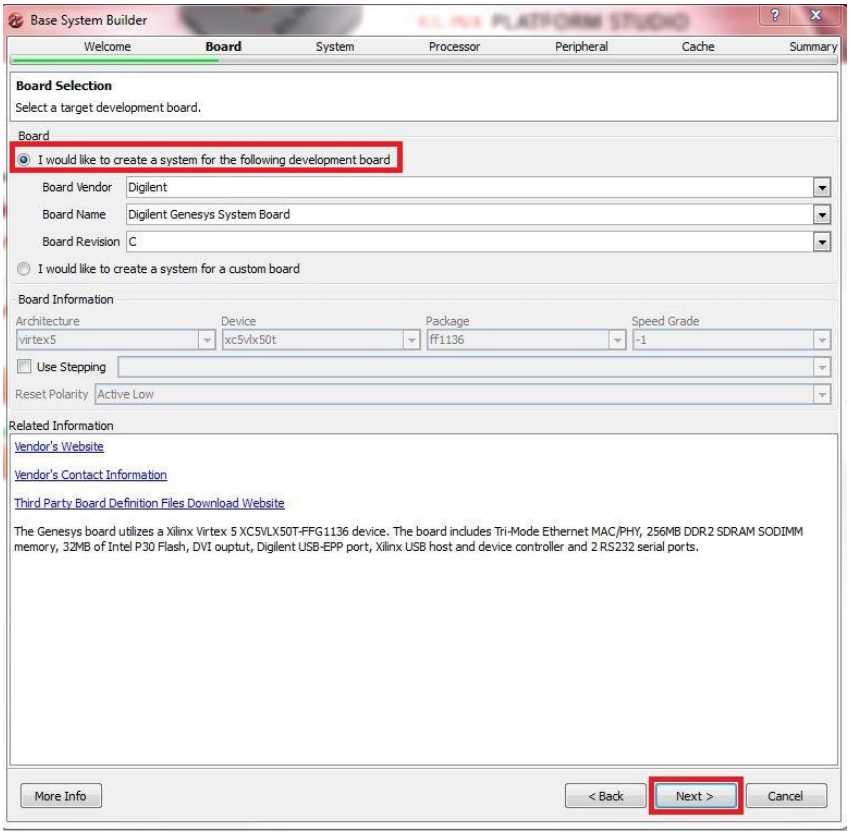
Make sure you set the correct "Project Peripheral Repository Search Path." If you can't find the file on your computer you can download it from http://www.digilentinc.com/Products/Detail.cfm?Prod=GENESYS (select DOC# DSD-0000275)



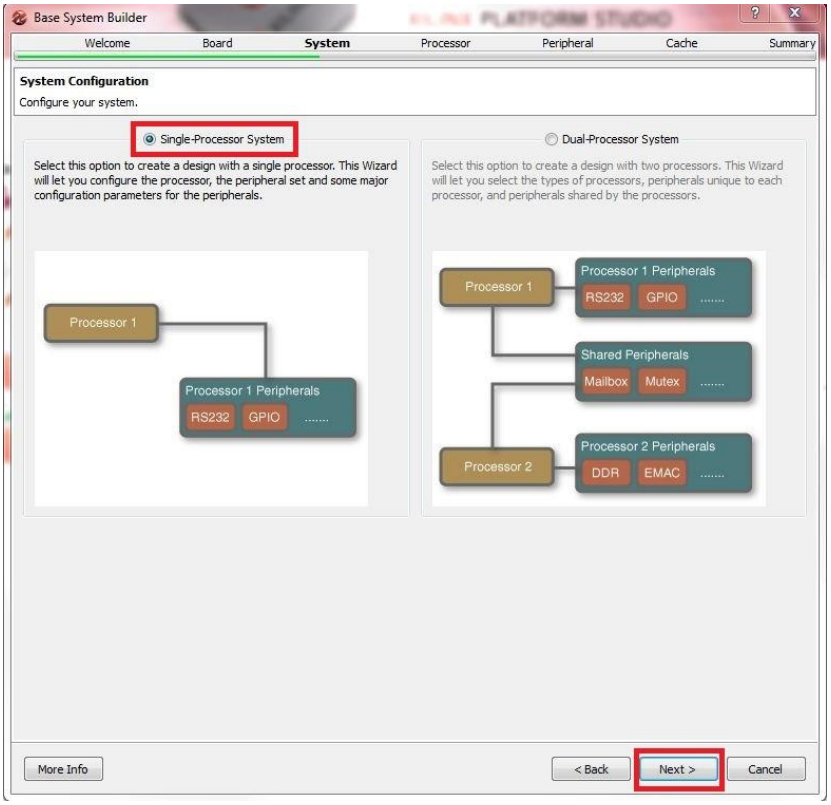
3. Select New Design and press Next.



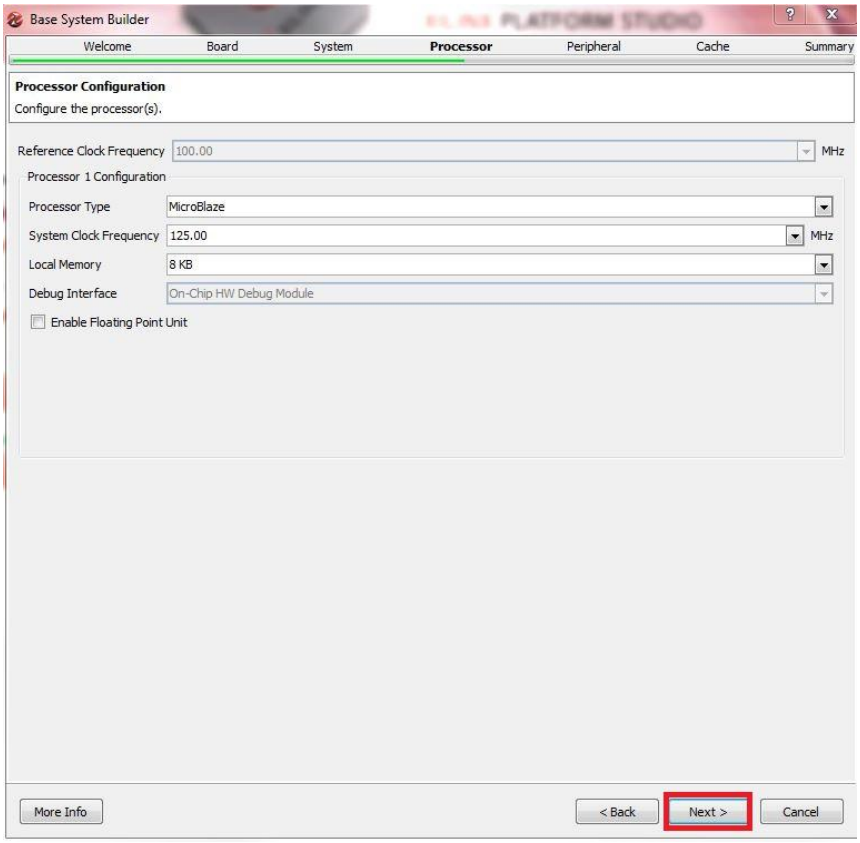
4. Make sure the correct board is selected and press Next.



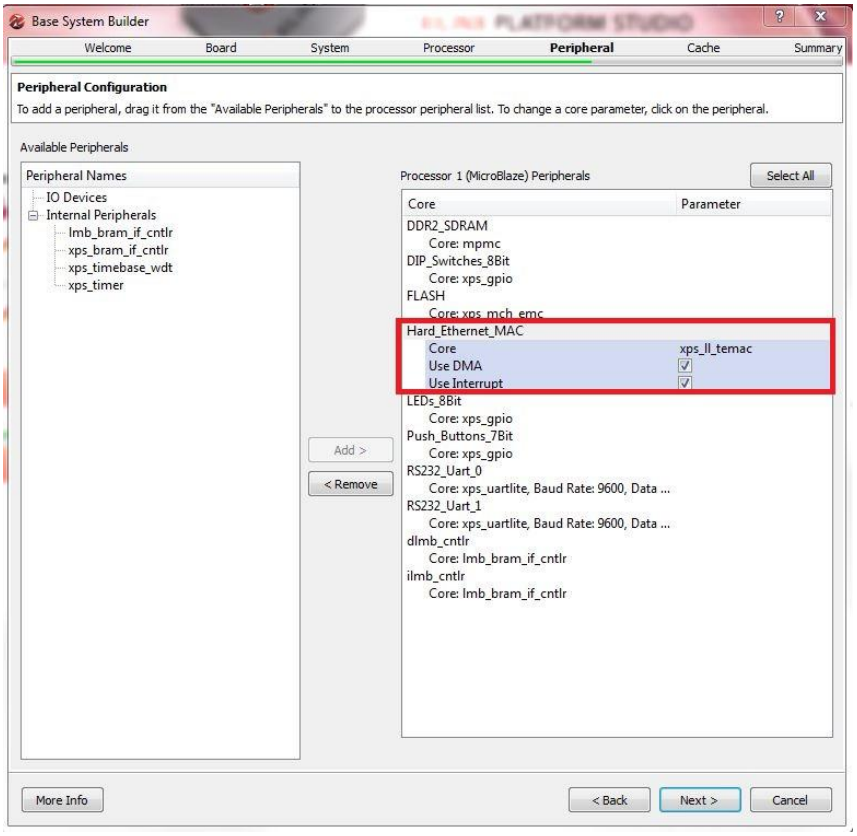
5. Select Single-Processor System



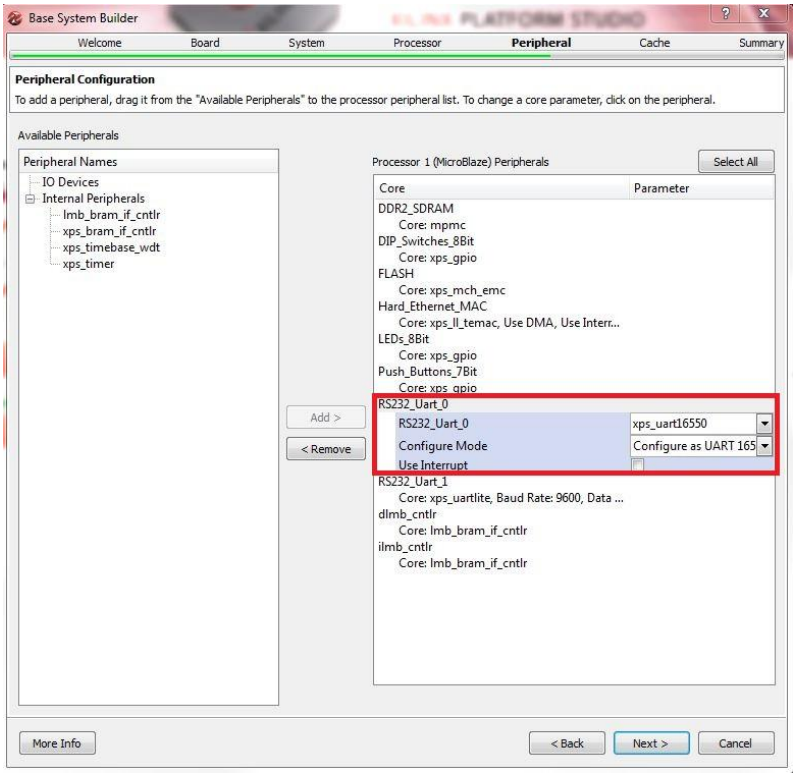
6.



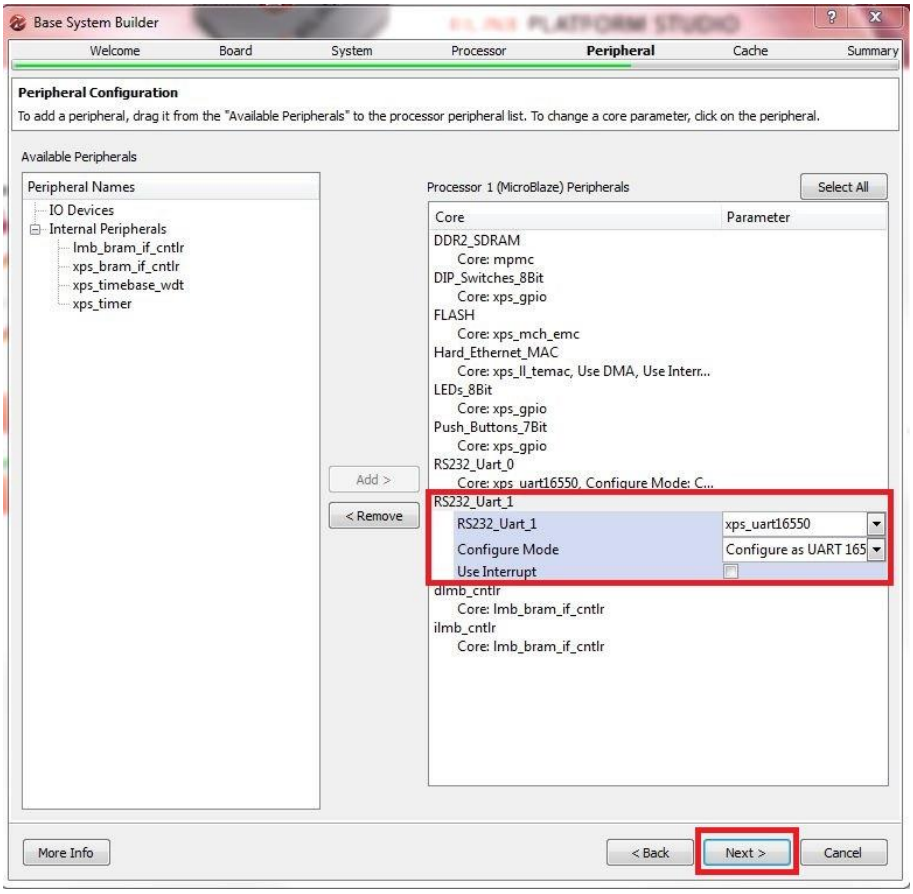
7. Configure the Ethernet port, DO NOT press Next yet



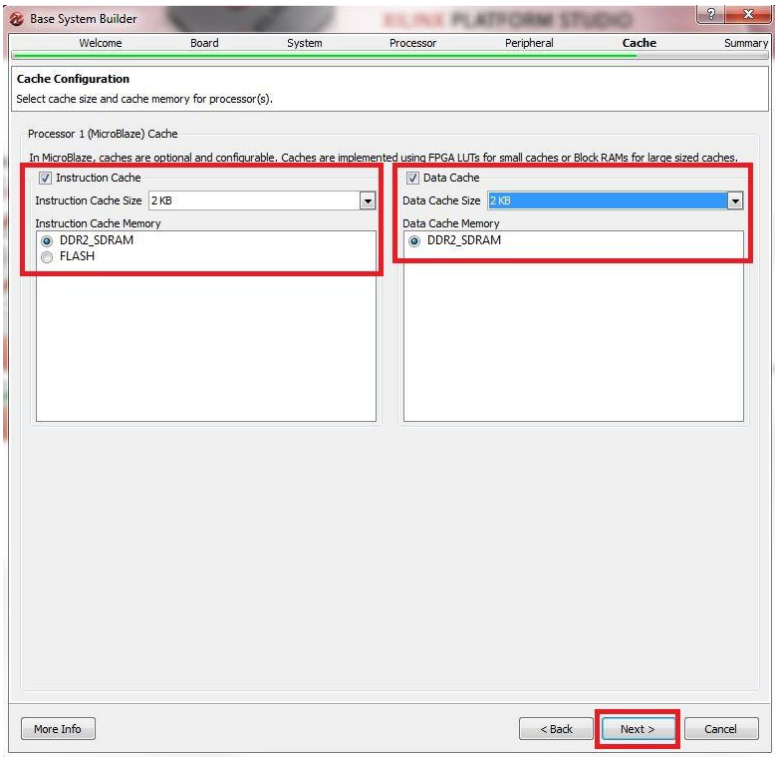
8. Configure the RS232 serial port, DO NOT press Next yet



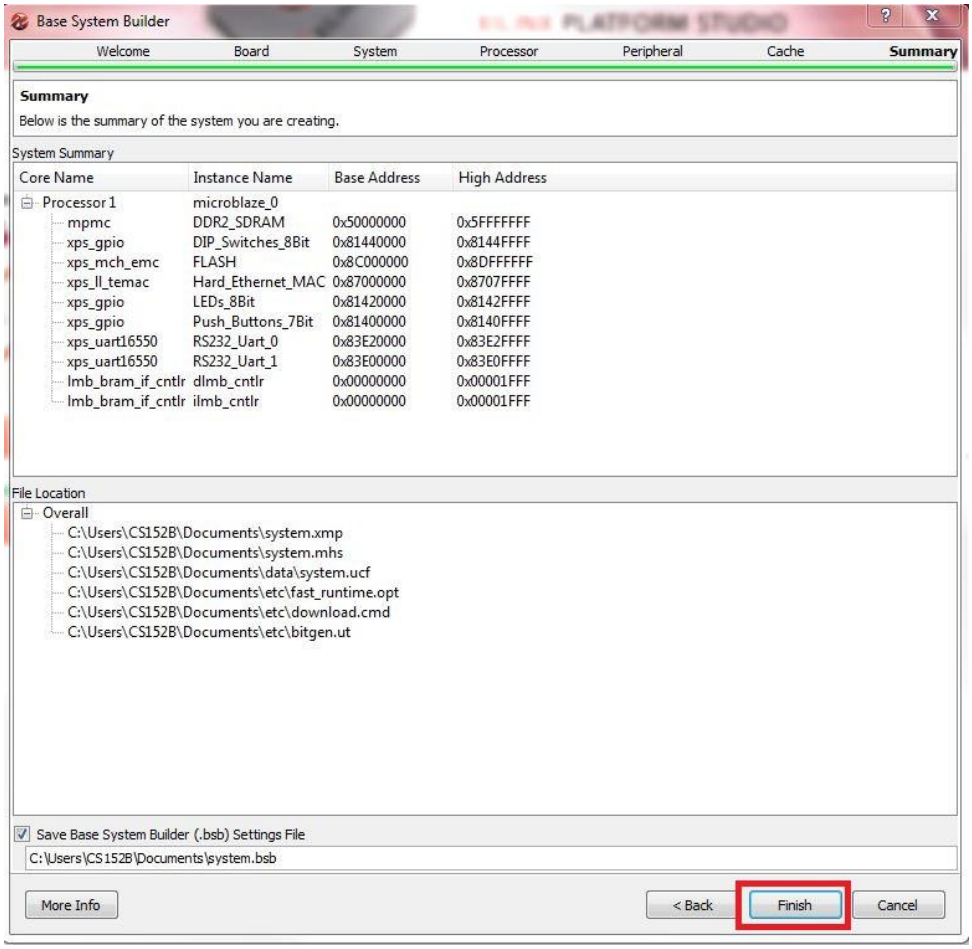
9. Configure the second RS232 UART, now you can press Next



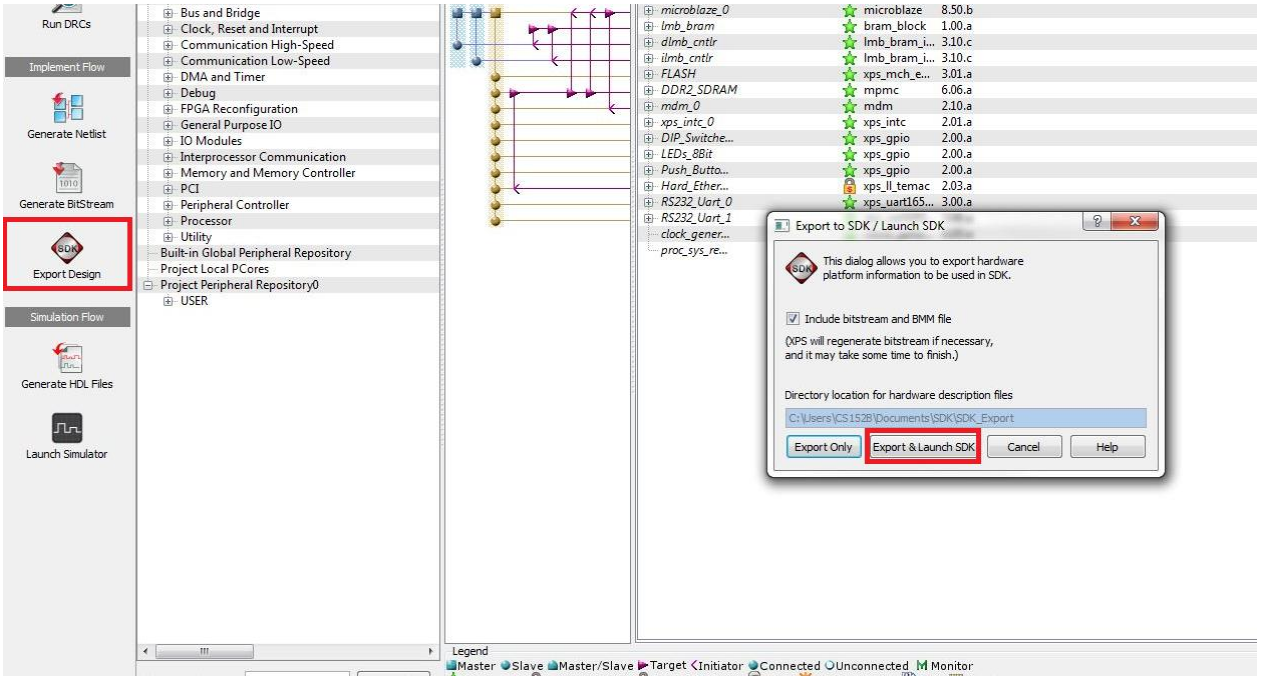
10. Configure Data and Instruction Cache



11. Verify your settings and press Finish

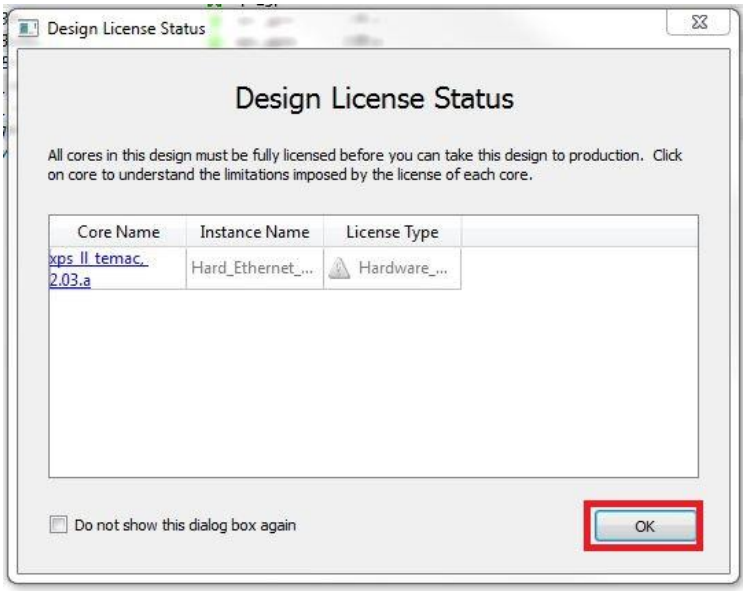


12. First press the Export Design Button, then press Export and Launch SDK

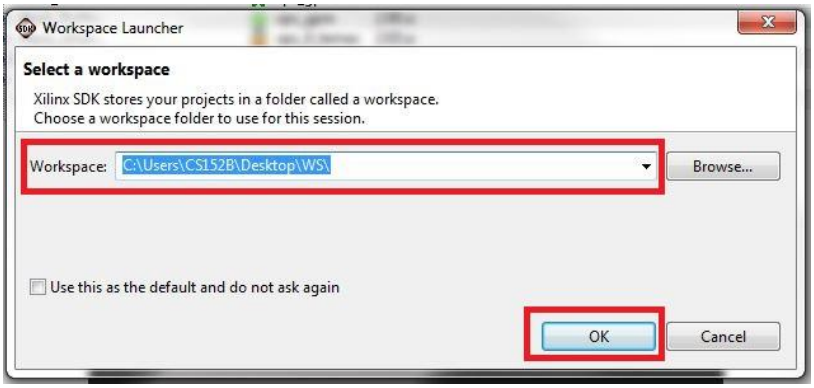


13. Export the Design takes about 30 minutes the first time, so just sit tight and wait for it to complete. Afterwards you'll be able to get right into the SDK without waiting. If you ever make changes inside the EDK (like adding a bus or port) you'll have to export the design (and 30 minute process) again.

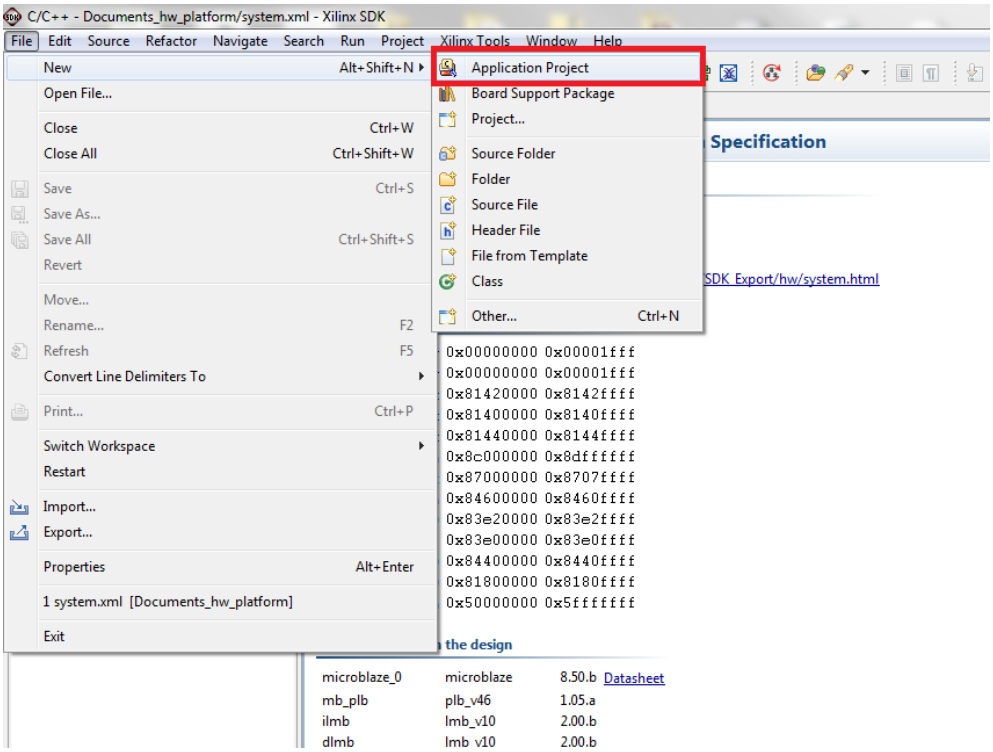
You might see messages like this while you're waiting, just press OK.



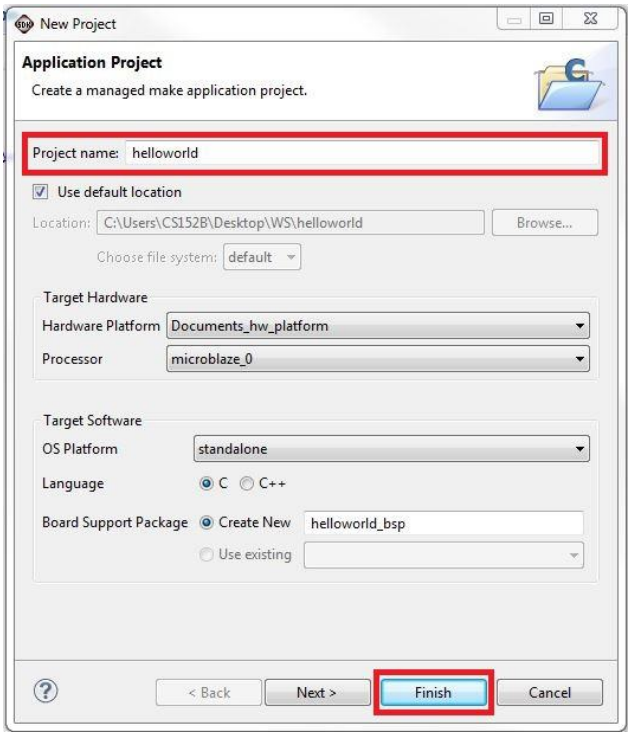
14. Now select a folder for your Workspace



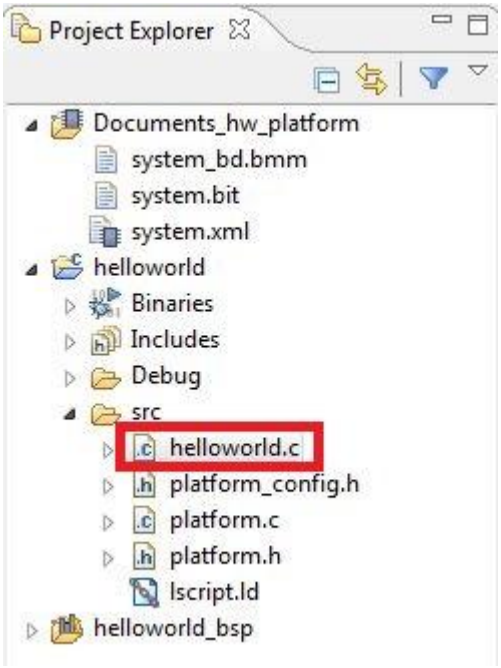
15. Now the Xilinx SDK will open. Select New Application Project



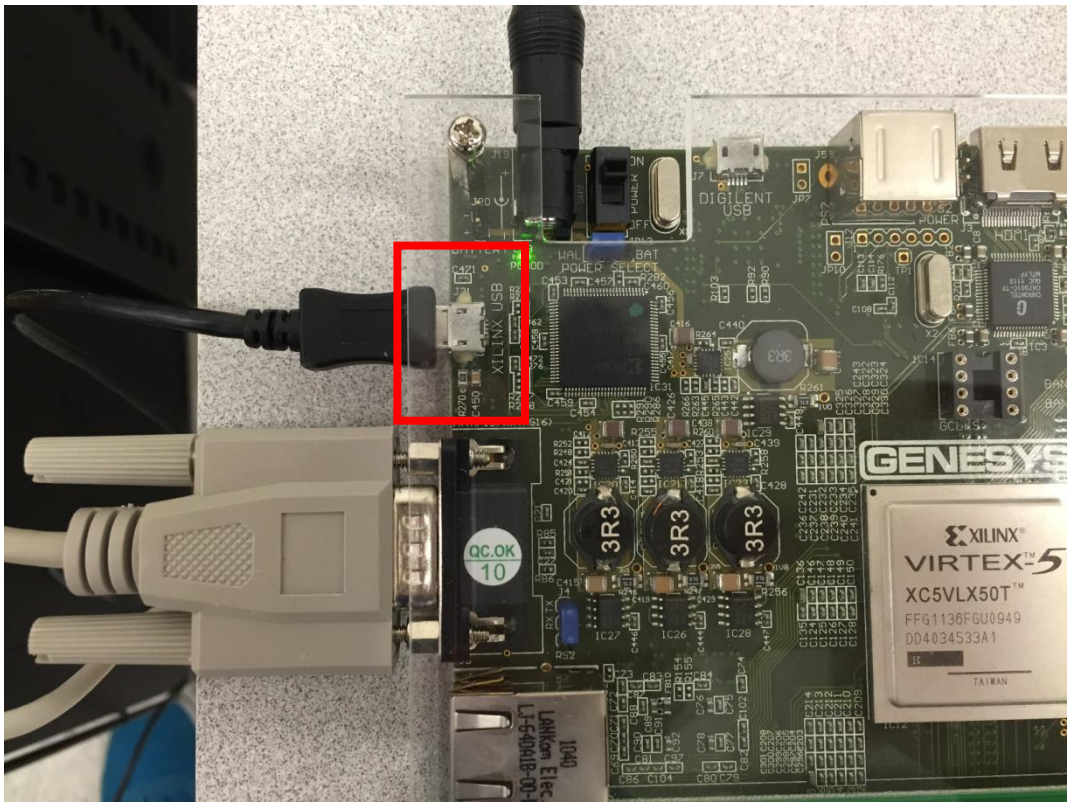
16. Give your project a name and select Finish



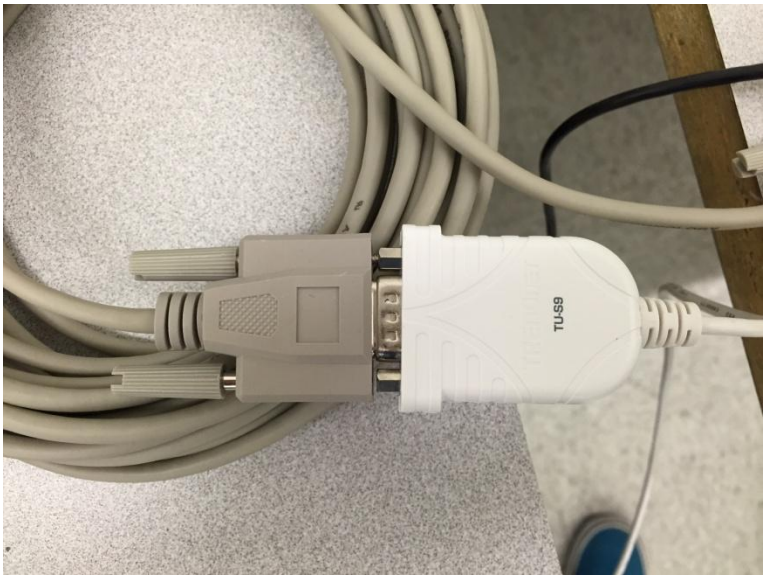
17. Take a look at your Project Explorer Window and expand the folder until you find your helloworld.c file. Double Click on it and examine the code.



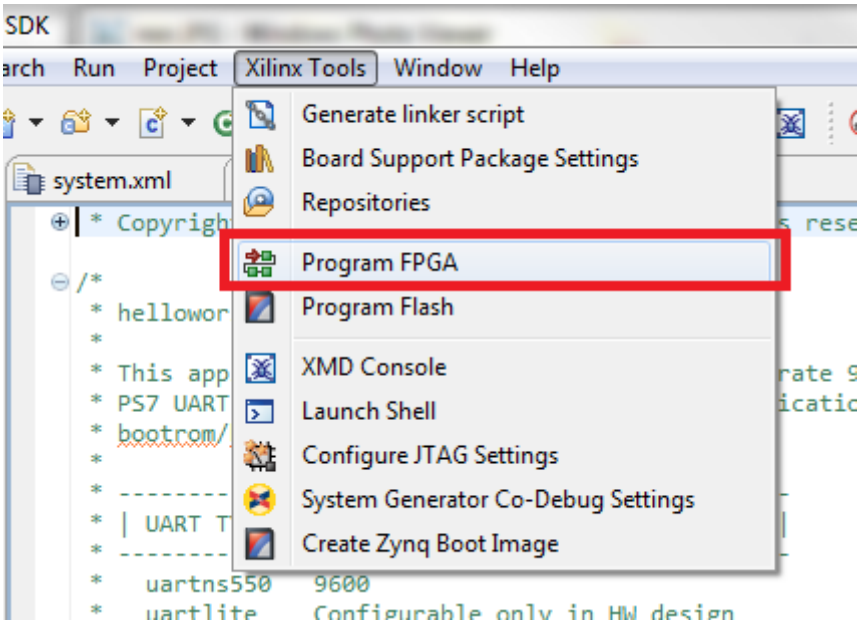
18. Now it's time to connect your board to the computer. Make sure you connect the micro USB cable to the Xilinx USB port, not the Diligent one. Connect an RS232 cable to the serial port as shown.



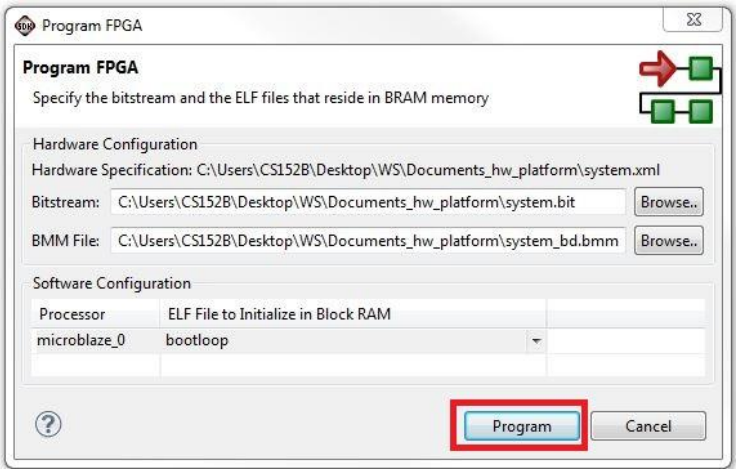
19. You'll also need an serial to usb adapter.



20. Now that you're connected press Xilinx Tools >> Program FPGA

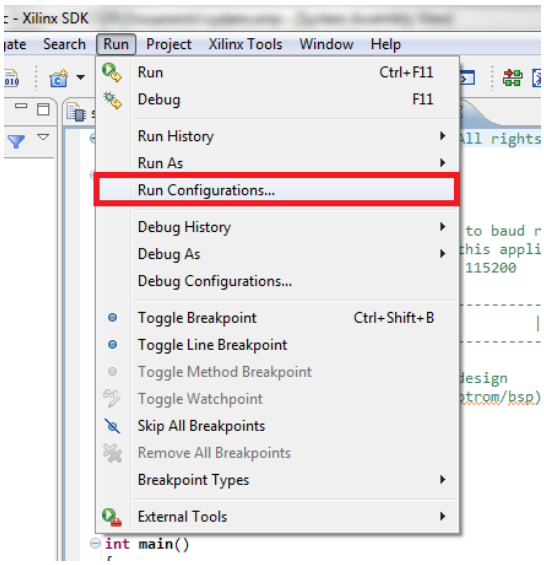


21. Now press Program.

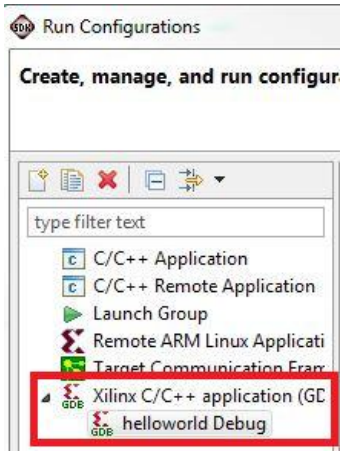


22. If you run into some error here, check to see if you're connected to the Xilinx USB port. If the error still persists, try relaunching the SDK and EDK (close the program and run it again).

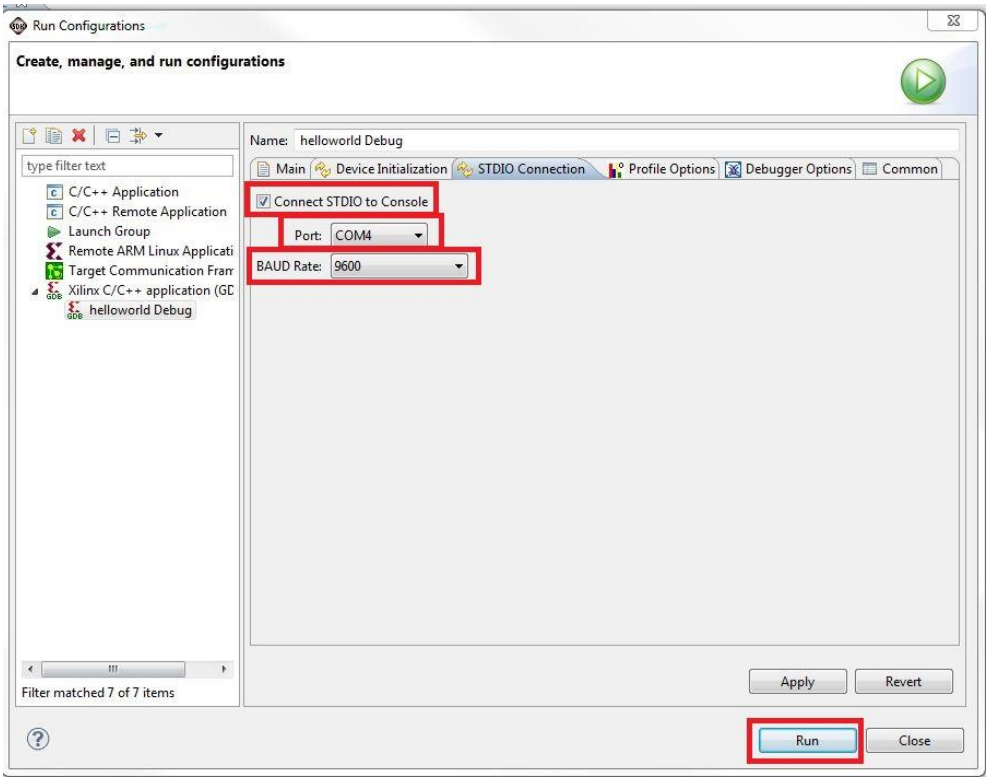
23. Now go to the Run menu and select Run Configurations.



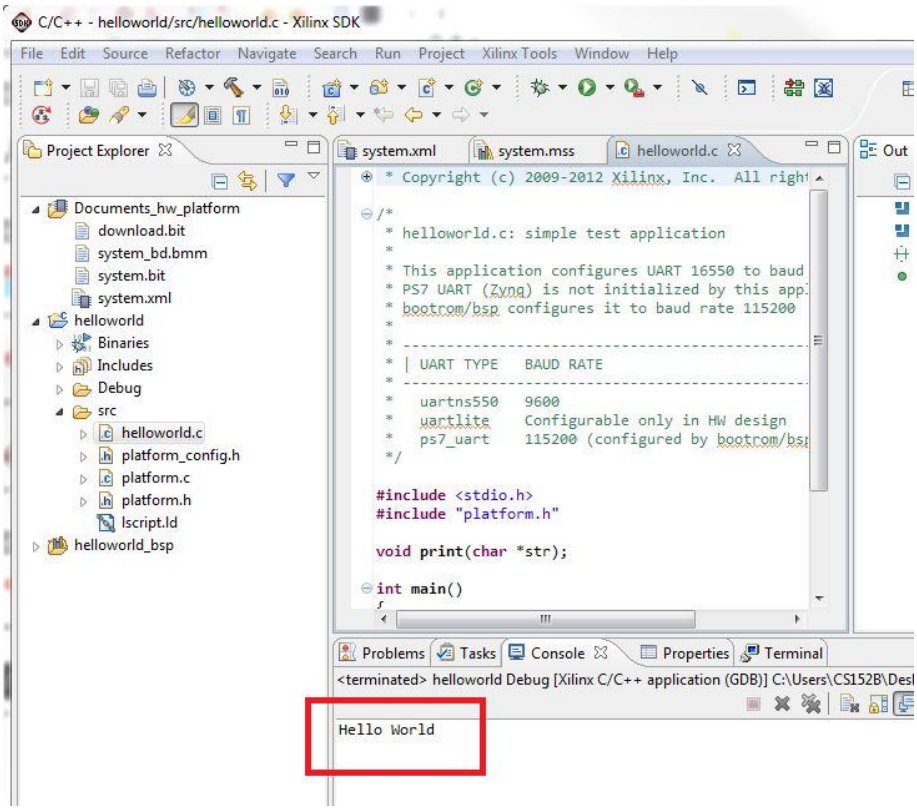
24. In the side window Double Click Xilinx C/C++ application (GDB).



25. Go to the STDIO Connection Tab and select Connect STDIO to Console. Select the port and set the BAUD rate to 9600. Then press Run.



26. You should see the output in your console now.



27. If you are having difficulties then ensure that you have connected the serial cable to your PC and have set the baud rate to 9600.

Now we can finally get started on the iRobot!

1. The iRobot can communicate with other peripherals over a serial connection.

The communication occurs at a BAUD rate of 5760. We can change the Baud Rate in microblaze using the following code:

```
XUartNs550_SetBaud(XPAR_RS232_UART_0_BASEADDR, XPAR_XUARTNS550_CLOCK_HZ, 57600);
```

Each transmission is 1 byte (8 bits) which we can set using:

```
XUartNs550_SetLineControlReg(XPAR_RS232_UART_0_BASEADDR, XUN_LCR_8_DATA_BITS);
```

2. Before you can communicate with the iRobot you need to send 2 bytes (each byte being 8 bits) of data. The first byte is the start command which has an opcode of 128 (0x80). The second byte puts the iRobot into Full mode (0x84).

```
XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x80);

XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x84);
```

3. Be careful not to start sending commands right after the initialization step. If you do the iRobot will just ignore them. To remedy this issue we need to implement some sort of sleep function:

```
int i, j;

for (j=0; j<2; j++)

     for (i=0; i<25000000; i++)

         asm("nop");
```

This function is also useful for when you want the iRobot to do something for a certain duration of time (like go forward for 2 seconds).

5. To drive the iRobot we must send 5 consecutive bytes.

The first byte 137 (0x89) specifies that the iRobot is in drive mode.

```
XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x89);
```

You can specify the velocity of the iRobot in mm/s. This information is encoded in the next two bytes (The high byte comes first). In this example we'll run it at 128 m/s (0x0080).

```
XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x00);

XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x80);
```

6. The next two bytes specify the radius at which the iRobot will move (you can use these bits to implement turning). If you just want to go straight, send a signal of 32767 (0x7FFF).

```
XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x7f);

XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0xff);
```

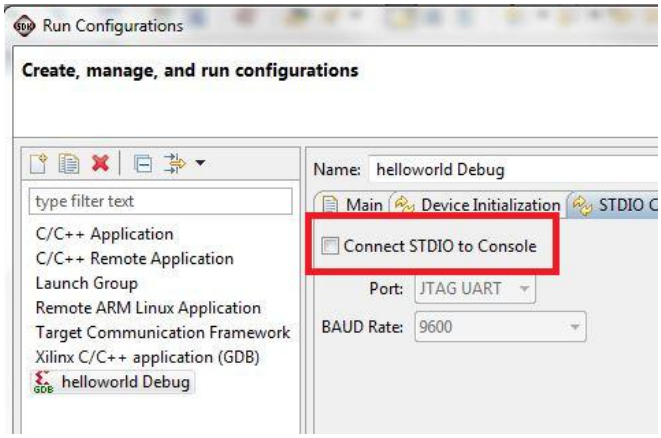
7. The full source code is provided below:

```
#include <stdio.h>
#include <stdlib.h>
#include <xparameters.h>
#include <xgpio.h>
#include <xuartns550_l.h>
#include <xio.h>
#include <xenv_standalone.h>
#include <xil_printf.h>
#include <xstatus.h>
#include <xuartlite_l.h>
#include <time.h>
#include "platform.h"
void print(char *str);

int main()
{
   init_platform();

   XUartNs550_SetBaud(XPAR_RS232_UART_0_BASEADDR, XPAR_XUARTNS550_CLOCK_HZ, 57600);
   XUartNs550_SetLineControlReg(XPAR_RS232_UART_0_BASEADDR, XUN_LCR_8_DATA_BITS);

// Initialize
   XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x80);
   XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x84);

// Sleep for 2 seconds
   int i,j;
   for (j=0;j<2;j++)
       for (i=0;i<25000000;i++)
               asm("nop");

// Forward
   XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x89);
   XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x00);
   XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x80);
   XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0x7f);
   XUartNs550_SendByte(XPAR_RS232_UART_0_BASEADDR,0xff);

   return 0;
}
```

8. Paste in the code above onto your Hello World microblaze project from earlier. Select Xilinx Tools >> Program FPGA. Then Press the Program button, just like before.

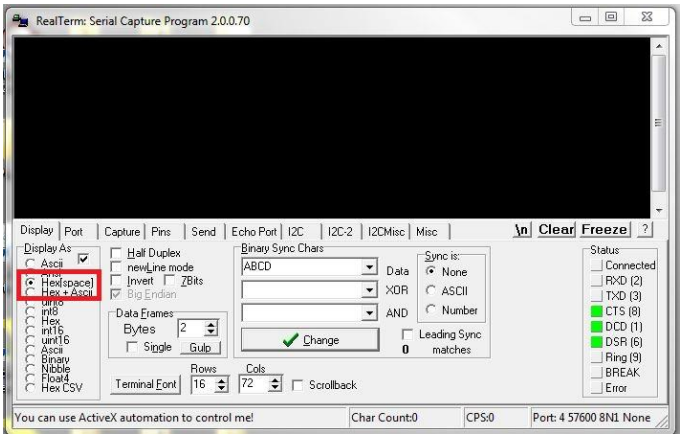
IMPORTANT: Xilinx SDK does not auto save your file when you press program FPGA. Ensure you have pressed the save button before programming FPGA.

9. Go to Run >> Run Configurations and deselect the Connect STDIO to Console Option, but DO NOT press Run yet.

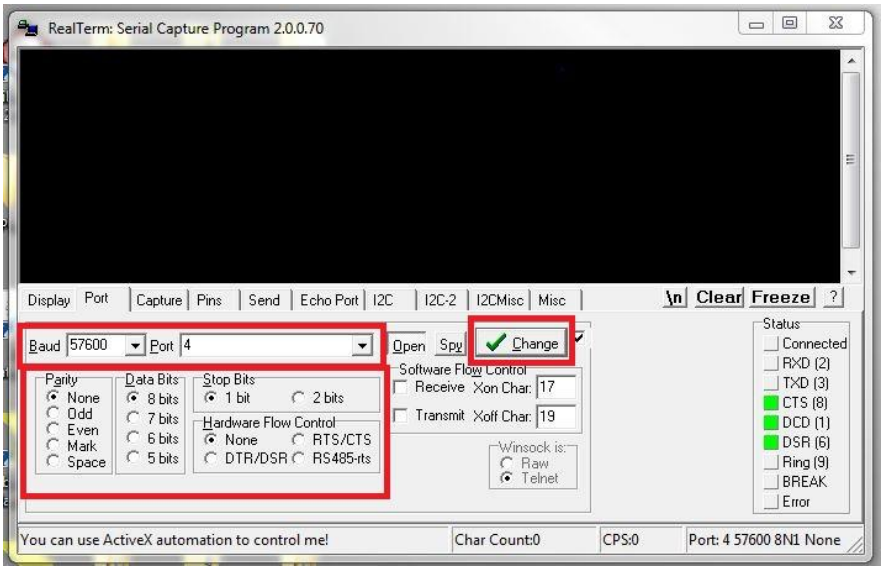


10. Before we start interfacing with the iRobot, we must ensure that the signals are correctly being transmitted from the FPGA. You can do so using RealTerm. If you don't have it installed on your computer, you can get it from here: http://realterm.sourceforge.net/

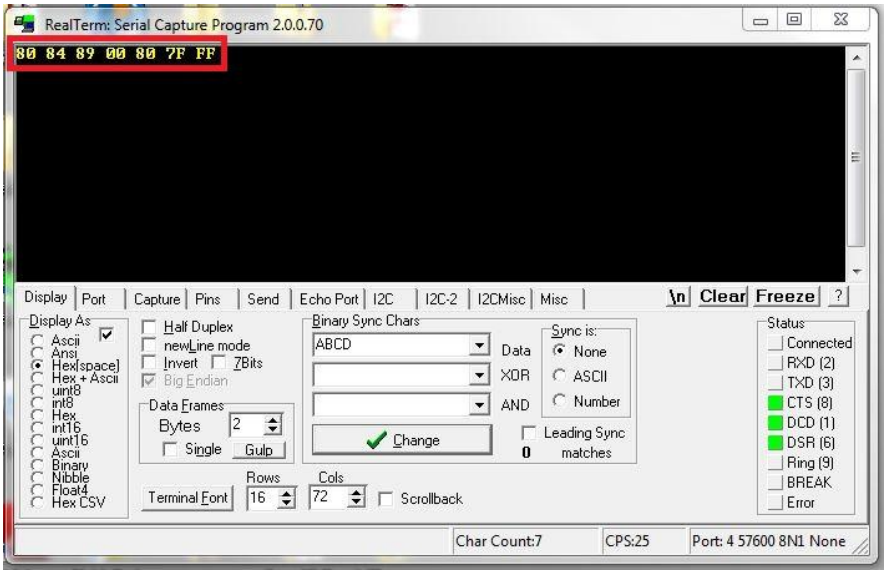
Open up Realterm and ensure that you have set the display to output Hex.



Then go to the Port tab and ensure that you have set the BAUD rate to 57600. Make sure your settings match the ones below.



10. Now press Run from the SDK and you should see the following signals on RealTerm:



11. Now that we can see the signals are being successfully transmitted, we can connect the iRobot to the FPGA board. You'll need a male adapter to connect the serial cable from the FPGA to the iRobot.

12. Turn the iRobot on and run the program from the SDK. If everything went as planned, you should see your iRobot move forward after 2 seconds. If you've verified your transmission through RealTerm and can't get the iRobot to respond, ensure that the iRobot is charged and and the iRobot cable is functioning. There are LEDS on cable which should light up after you press run.

13. The iRobot can do more than just move around. Check out the documentation for a full list of commands at http://www.irobot.com/filelibrary/pdfs/hrd/create/Create%20Open%20Interface_v2.pdf

Good Luck on Your Project!