

Getting Started with Embedded System Development using MicroBlaze processor & Spartan-3A FPGAs

This tutorial is an introduction to Embedded System development with the MicroBlaze soft processor and low cost Spartan-3 FPGAs. In this tutorial you will design a custom Embedded system and develop your application software.

First, you will get started with the Base System Builder (BSB) wizard in Xilinx Platform Studio (XPS). You will create a custom Embedded Processing system (a custom Microcontroller) targeted at the Spartan 1800A FPGA board. Then you will use the Xilinx Software Development Kit (SDK – an Eclipse-based IDE) to compile and debug some embedded software that runs on this system.

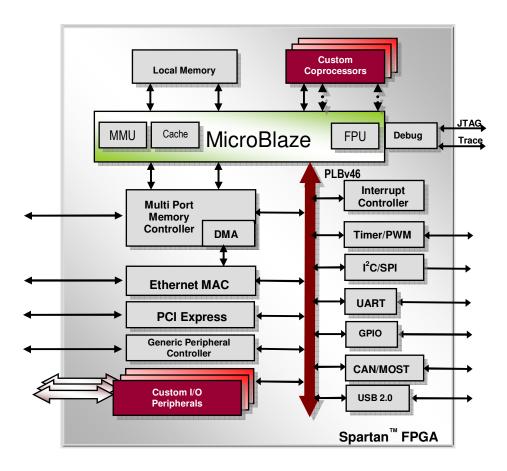


Figure 1 - MicroBlaze Processor System Architecture



Objectives

After completing this tutorial, you will be able to:

- Create an Embedded system with the MicroBlaze processor on Spartan-3 FPGA using Base System Builder (BSB) wizard
- Generate and Download the FPGA design bitstream on the board using Xilinx Platform Studio (XPS)
- Create a simple software application using Eclipse-based Platform Studio Software Development Kit (SDK)
- Debug the software application using the Eclipse IDE (SDK)

Requirements

Embedded Development HW/SW Kit – Spartan-3A DSP S3D1800A MicroBlaze Processor Edition

Hardware Requirements

- Xilinx MicroBlaze Development Kit (Spartan-3 1800A edition)
 - o Spartan-3 1800A DSP FPGA board
 - Xilinx Platform Cable USB
 - RS232 cable

Software Requirements

- ISE Design Suite 10.1 (including EDK 10.1)
- HyperTerminal or Tera Term terminal program connecting the PC and the FPGA board at 115,200 baud

Step 1 – Create Custom Embedded System

•

- 1. Open Xilinx Platform Studio:
- 2. You will create a new project:
 - Select "Base System Builder wizard" and Click OK



💠 Xilinx Platform Studio 🛛 🛛					
Create	new or open existing project O[Base System Builder wizard (recommended)]				
6	BSB O Blank XPS project				
	Open a recent project				
Browse for More Projects					
Browse installed EDK examples (projects) <u>here</u>					

- 3. Select the design directory for new project:
 - Click the **Browse** button and browse to the following location: C:\Workshop\MB_Workshop\mb_basic\
 - The file name should be: **system**
 - Click Save
 - Click **OK**

Create New XPS Project Using BSB Wizard New project Project file	Platform Studio Project
C-/Workshop/MB_Workshop/mb_basic/system.xmp Browse Advanced options (optional: F1 for help) Set Project Peripheral Repositories Browse OK Cancel	Save in: Imb_basic Image: Imag
	My Network File name: system Image: Save Places Save as type: Platform Studio Project (".xmp) Image: Cancel

- 4. Next the Wizard will give options on how to create the system:
 - Select "I would like to create a new design"
 - Click "Next"
- 5. Select the target development board:
 - Select Board vendor:"Xilinx"
 - Select Board name: "Spartan-3A DSP 1800A Starter Board"
 - Click "Next"



🕏 Base System	Builder - Select Board	X		
Select a target dev	velopment board:			
-Select board				
💿 I would like	to create a system for the following development board			
Board <u>v</u> endor:	Xilinx	•		
Board n <u>a</u> me:	Spartan-3A DSP 1800A Starter Board	/		
Board <u>r</u> evision:	Virtex-4 ML410 Evaluation Platform			
Note: Visit the v	Virtex 5 ML501 Evaluation Platform Virtex 5 ML505 Evaluation Platform			
Vendor's Websi	Virtex 5 ML506 Evaluation Platform			
	Spartan-3A DSP 1800A Starter Board Spartan-3E 1600E MicroBlaze Dev Board			
	Spartan-3 Starter Board			
 I would like 	Spartan-3AN Starter Kit			
-Board description	Spartan-3A Starter Kit Spartan-3E Starter Board	Ē.		
The MicroBlaze Spartan3A DSP 1800A Embedded Development Board utilizes Xilinx Spartan-3E XC3SD1800A-4FG676 device. The board includes 125MHz System Clock, RS232 serial ports, 8 DIP switches, 4 push buttons, 8 LEDs, VGA port, 4 SPI Ports, 10/100/1000 Ethernet port, 64MBit SPI flash, 16 MB of parallel NOR flash and 128MB DDR2 SDRAM. User I/D is supported with a 168 Pin EXP mezzanine connector, 2 Digilent 6 pin ports, and a general purpose 30 pin I/O connector				

- 6. Select the processor. (*Here you have no other option only the Virtex-5 FXT FPGA has a PowerPC processor*):
 - Click "Next"
- 7. Configure the processor options. You can modify options like local on-chip memory size, caches, FPU, etc. For now, you will leave the defaults and modify the system later.
 - Click "Next"

- 8. Configure the I/O interfaces of your embedded system. Here you can select the right set of Peripherals you need for your target system or microcontroller and also customize each peripheral:
 - Select the drop down for "Baudrate" and choose "115200"
 - Click "Next"

🗢 Base System Builder - Configure 10 Interfaces (1 of 3)	
The following external memory and IO devices were found on your board: Xilinx Spartan-3A DSP 1800A Starter Board Revision 1 Please select the IO devices which you would like to use: 10 devices	
 ✓ RS232_Uart_1 Peripheral: XPS UARTLITE ✓ Baudrate (bits per seconds): 9600 ✓ 2400 ✓ Data bits: 4800 9600 ✓ 	Data Sheet
Image: 19200 Image: 19200 38400 57600 Image: 19200 Image:	Data Sheet
 ✓ Push_Buttons Peripheral: XPS GPI0 ✓ ✓ Use interrupt 	Data Sheet
 ✓ DIP_Switches_8Bit Peripheral: XPS GPI0 ✓ Use interrupt 	Data Sheet
More Info	Cancel

XILINX®

- 9. Continue configuring the I/O interfaces. For now, you will unselect the Ethernet MAC and SPI Flash peripherals as we will not be using it in this tutorial. This will make the design smaller and also save some time when we create the FPGA netlist/bitstream for this tutorial:
 - Unselect Ethernet MAC peripheral
 - Click "Next"
 - Unselect SPI Flash peripheral
 - Click "Next"

🗢 Base System Builder - Configure 10 Interfaces (2 of 3) 👘	×
The following external memory and IO devices were found on your board: Xilimx Spartan-3A DSP 1800A Starter Board Revision 4 Please select the IO devices which user would like to use: 	
Ethernet_MAC	Data Sheet Noje
Base System Builder - Configure IO Interfaces (3 of 3) The following external memory and IO devices were found on your board: Xilinx Spatan-3A DSP 1800A Starter Board Revision 1 Please select the ID devices which year would like to use: IO devices SPL_FLASH	Data Sheet

10. Add some internal on-chip peripherals:

- Click "Add Peripheral"
- Select "XPS Timer"
- Click "Ok"

🚸 Base System Bu	iilder - Add Internal Peripherals (1 of 1)	×
"Add Peripheral" butto	that do not interact with off-chip components. Use the on to select from the list of available peripherals. dd any non-ID peripherals, click the "Next" button.	
Peripherals		Add Peripheral
	Add Peripheral	
	Select the peripheral you want to add:	
	XPS TIMER	
	XPS TIMEBASE WDT XPS TIMER	
		1

- 11. Customize the Timer by enabling Interrupts:
 - Click "**Use Interrupt**" for the timer peripheral
 - Click "Ok"

🗢 Base System Builder - Add Internal Peripherals (1 of 1)	X
Add other peripherals that do not interact with off-chip components. Use the "Add Peripheral" button to select from the list of available peripherals.	
If you do not wish to add any non-IO peripherals, click the "Next" button.	
	Add Peripheral
Peripherals	
_xps_timer_1	
Peripheral: XPS TIMER	<u>R</u> emove
Counter bit <u>w</u> idth: 32 🗸	Data Sheet
Timer mode	
⊙ <u>I</u> wo timers are present	
<u>D</u> ne timer is present	
Use interrupt	

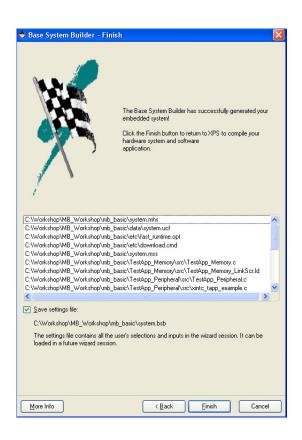
XILINX®

12. Next the wizard will help you create some sample programs that will do some diagnostics on your custom embedded system.

You can leave the defaults and modify the software later:

- Click "Next"
- Click "Next"
- Click "Next"
- 13. Now you have finished creating a basic system. The wizard will let you review your selections and generate the system:
 - Click "Generate"
 - Click "Finish"

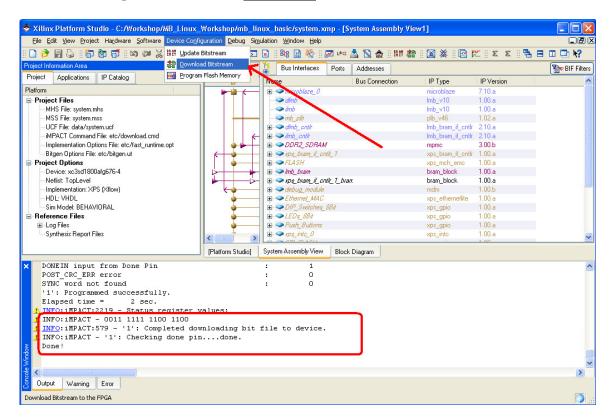
Base System Builder - System Created low is a summary of the system you have created. Please review the information below. If it is reect, hit Generate's to enter the information into the XPS data base and generate the system files. revise return to the previous page to make corrections. Processor: microblaze_0 System clock triguency: 62.50 MHz						
On Chip Memoy: 8 KB Total Off Chip Memoy: 128 MB - DDR2_SDRAM = 128 MB 						
editing features of X PLB Bus : PLB_1	PS. √46 Inst. name: mb_plt	Attached (Components:			
Core Name	Instance Name	Base Addr	High Addr			
xps_uartlite	RS232_Uart_1	0x84000000	0x8400FFFF			
xps_gpio	LEDs_8Bit	0x81400000	0x8140FFFF			
xps_gpio	Push_Buttons	0x81420000	0x8142FFFF			
xps_gpio	DIP_Switches_8Bit	0x81440000	0x8144FFFF			
mpmc	DDR2_SDRAM_C_MP	0x88000000	0x8FFFFFFF			
xps_timer	xps_timer_1	0x83C00000	0x83C0FFFF			
mdm	debug_module	0x84400000	0x8440FFFF			
xps_intc	xps_intc_0	0x81800000	0x8180FFFF			
LMB Bus : LMB_	V10 Inst. name: ilmb	Attached Cor	nponents:			
Core Name	Instance Name	Base Addr	High Addr			
lmb_bram_if_cntlr	ilmb_cntlr	0x00000000	0x00001FFF			
LMB Bus : LMB_	V10 Inst. name: dlmb	Attached Co	mponents:			
Core Name	Instance Name	Base Addr	High Addr			
lmb_bram_if_cntlr	dimb_cntir	0x00000000	0x00001FFF			
More Info		< <u>B</u> ack	<u>G</u> enerate Cancel			



Step 2 – Build and Download the Hardware system

- 1. From the Xilinx Platform Studio (XPS) IDE, create the FPGA bitstream and download the hardware design onto the Spartan FPGA board as follows:
 - Select the menu "Device Configuration-> Download Bitstream"

NOTE: This step will take almost <u>30 minutes</u>. Do not close the window !



This will create the FPGA design files, synthesize the netlist and also place and route the design creating the bitstream. The FPGA on the board will also be configured with this new bitstream containing your custom embedded system.

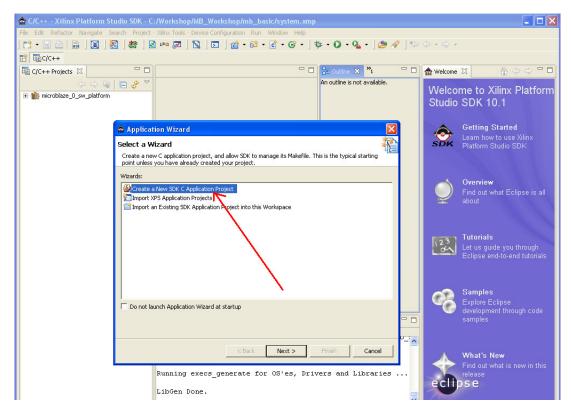
Your custom hardware is now ready. Next will write and program some software.

Step 3 – Develop Embedded software

- 1. From the Platform Studio IDE, launch Eclipse IDE for software development:
 - Select the menu "Software->Launch Platform Studio SDK"

Xilinx Platform Studio - C:\We	orks	hop\MB_Workshop\mb_basic\:	system. x	mp - [S	ystem Assem	ıbly View1]	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject Ha <u>r</u> dware	Soft	ware Device Co <u>n</u> figuration <u>D</u> ebug	Simulation	Window	Help		
i 🗋 🖻 🖥 🖧 i i 🛱 🔯 i 🛤	SOR	Launch Platform Studio SD <u>K</u>	1 🛛 🗄	88 🔡	🍀 🛃 Llha	📥 🛐 🏫 🗄 🛤 🛔	書 🛙 🐹 💥
	<u>s</u>	Software Platform Settings					
Project Information Area		Assign Default <u>D</u> rivers	terfaces	Ports	Addresses		BIF Filters
Project Applications Catalog	Lip G	Generate Libraries and BSPs	_	Bus	Connection	IP Type	IP Version
Platform	1	Add Software Application Project	blaze_0			microblaze	7.10.a
📮 Project Files	*	Build All User Applications				lmb_v10	1.00.a
- MH2 File: system.mhs	-	Get Program Size				Imb_v10	1.00.a
- MSS File: system.mss	6.9		ıb			plb_v46	1.02.a
- UCF File: data/system.ucf		📓 Generate Linker Script	cntlr			Imb_bram_if_cntlr	2.10.a
		Clean Libraries	cntlr			Imb_bram_if_cntlr	2.10.a
			2_SDRAI	1		mpmc	4.00.a
Bitgen Options File: etc/bitgen.ut	9	Clean Programs	bram			bram_block	1.00.a
Project Options	2	Clean Software	g_module			mdm	1.00.b
Device: xc3sd1800afg676-4	_	è ⊞ <i>≫ DIF</i>	Switches	88 is		xps_gpio	1.00.a

- 2. In SDK, start creating a new software application using the Application Wizard:
 - Select "Create a New SDK C Application Project"
 - Click "Next"



3. Provide a name to the new project



- Type "hello"
- Click "Next"

Use the rest of the project defaults

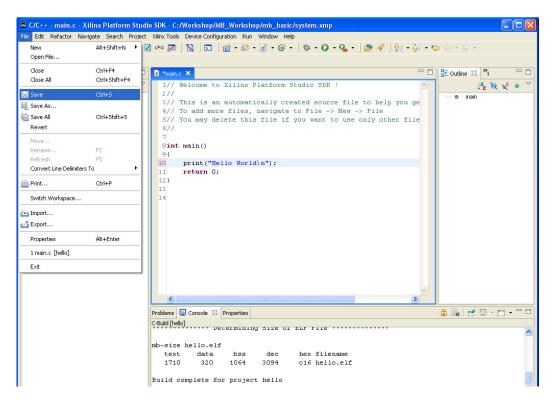
- Click "Next"
- Click "Finish"



- 4. In the Editor view, edit the default program "main.c" and save it:
 - Insert the following line after main() {
 print("Hello World");

NOTE: It is print() and not printf().

• Select the menu "File-> Save"



Saving the program will also compile it and result in "hello.elf" as you see in the console view.

Now you are ready to download and debug the program on the board.

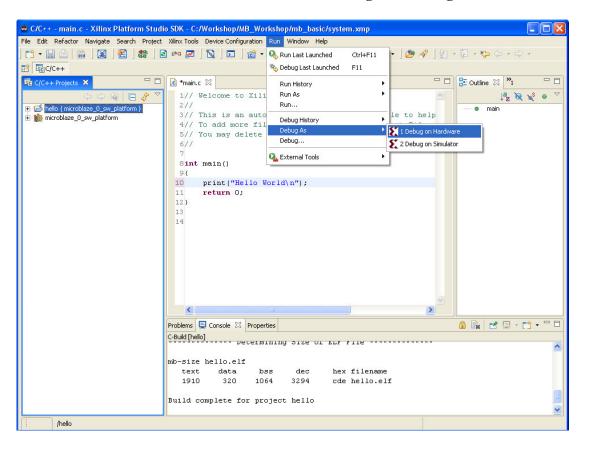
Step 4 – Debugging software program

- 1. Before debugging the Hello world program, you have to open the RS-232 serial communication with the board using the Tera Term console program.
 - Double Click on the Tera Term Icon on the Windows Desktop



The output of your Hello world program can be seen in this console.

- 2. From SDK, start debugging the program as follows:
 - Select the Menu item "Run->Debug As->Debug on Hardware"



SDK will now change to the Debug Perspective. SDK will also connect to the FPGA board via the USB cable, connect to the MicroBlaze processor and download the program to memory.

11

Now you are ready to debug the program using the Eclipse Debug perspective.

XILINX°

3. In the Debug perspective, click on the resume button (looks like *Play* button):
Click the "**Resume**" button

NOTE: You can alternatively select "Run->Resume"

🏟 Debug - main.c - Xilinx Platform Studio SDK - C:/Workshop/MB_Works	shop/mb_basic	/system.xmp				
File Edit Refactor Navigate Search Project Xilinx Tools Device Configuration Run Window Help						
🗂 • 🔚 👜 📓 🦉 🎇 🔮 🔤 🔤 🔽 🐚 - 🖸	- 💁 - 🛛 🕭	1 - 1	• 🐤 🗇 •	⇒ -		
☆Debug 旺C/C++						
🇚 Debug 🛛 🚺 🗈 🕼 🖏 🕄 🕀 🚓 🛒 🖬 🏹	(×)= Variable	es 🖾 Breakpoir	its Modules Regi	isters 🏭 🏭 🕂	°° × % · − ⊔	
Amb (arget debug agend (b) 1/06 9:13 PM) (Suspended)						
= 1 main() at\main.c:10 0x000001b8						
Debugger Process (6/1/08 9:13 PM)	ana an					
C:\Workshop\MB_Workshop\mb_Besic\SDK_projects\hello\Debug\hello.elf (6/1/	38 9:1					
					1000	
					2	
	> <				×	
			- 8			
🖸 main.c 🗙				🗄 Outline 🖾 🗍		
1// Welcome to Xilinx Platform Studio SDK !			~	····· • main		
2// 3// This is an automatically created source file to B	als you get	startad				
4// To add more files, navigate to File -> New -> File		starteu.				
5// You may delete this file if you want to use only		for your	project.			
6//						
7						
Sint main()						
9{ ⇒ 10 print("Hello World\n");						
11 return 0;			-			
E Console 🛛 Tasks Memory				🔳 💥 🖹 🔒	🛃 🖻 • 📬 • 🗖 🗖	
hello_HW [Xilinx C/C++ ELF] C:\Workshop\MB_Workshop\mb_basic\SDK_projects\hello\Debi	ug\hello.elt (6/1/08	9:13 PM)				
					\sim	
<u><</u>					>	
	Writable	Smart Insert	10:1			

You should now see the output of the program on the Serial console (TeraTerm program) as follows:

Hello World	
Hello World	
-	
· ·	

You can now use the Eclipse IDE to modify your program and debug it further. Here are some steps to try out on your own:

- 1. In the SDK IDE source code perspective,
 - a. Use the Application Wizard to import Test_App_Memory and Test_App_Peripheral programs.
- 2. While debugging,
 - a. View Registers, Memory
 - b. Single Step through the program using Step, Next
 - c. Toggle between assembly and source views
- 3. In the SDK IDE source code perspective,
 - a. Modify the linker script and change the location where the program is running from