

Using Serial Flash on the Xilinx Spartan-3E Starter Board

Version 8.1 February 23, 2006 Bryan H. Fletcher

Overview

The Xilinx Spartan-3E FPGA features the ability to configure from standard serial flash over a built-in Serial Peripheral Interface (SPI). Being general-purpose flash, the SPI serial flash can also be used for any other non-volatile storage that the user may need. One such non-volatile purpose is the storage of MicroBlaze processor application code for bootloading.

Objectives

The Xilinx Spartan-3E Starter Board features an STMicro M25P16 serial flash memory. This reference design demonstrates several aspects of using this serial flash and the Spartan-3E FPGA on the Xilinx Spartan-3E Starter Board, including:

- FPGA configuration over SPI
 - Store bitstream to serial flash
 - Configure FPGA from serial flash
- MicroBlaze test application utilizing the STMicro M25P16
 - Read manufacturer's ID
 - Perform a bulk erase
 - Write to all locations in the flash
 - Read from all locations in the flash
- MicroBlaze interactive user application utilizing the STMicro M25P16 for data program data storage
 - Read from a designated sector
 - Perform a sector erase
 - Write to a sector
- MicroBlaze bootloader application
 - Merge a configuration bitstream and binary MicroBlaze application
 - o Store merged file to serial flash
 - Copy application image from serial flash to external memory
 - Run from external memory

^{© 2006} Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.



Experiment Setup

Software Prerequisites

The recommended software setup for this reference design is:

- Windows2000 or WindowsXP
- Xilinx ISE 8.1i (Foundation or WebPack) with the latest Service Pack¹
- Xilinx EDK 8.1 with the latest Service Pack²
- XAPP445 Design Files, including the XSPI utility³

Hardware Prerequisites

The hardware setup used by this reference design includes:

- Computer with a minimum of 256 MB RAM and 256 MB Virtual Memory⁴
- Xilinx Spartan-3E Starter Board Rev C or D (Tested on Rev C)
- Digilent low-cost Parallel Cable III or Xilinx Parallel Cable IV (PC4) with flyleads (USB not currently supported for XSPI)
- (optional) USB device cable (for on-board Platform USB connection to JTAG chain)
- Serial Cable

Setup

- Jumper settings:
 - Install J30 jumper in position 3-4, which is M1 (M[2:0] = 101, Boundary Scan Mode). An additional jumper for M2 is needed but do not install it at this time.
 - Install two jumpers on J11, positions 1-2 and 3-4.
 - Install a jumper on J9, positions 2-3 (3.3V)
 - Install jumpers on JP6 and JP7
 - All other jumpers NOT installed
- Cables:
 - Do one of the following, but not both:
 - Plug in the Parallel JTAG Cable to the PC and the JTAG port on the board (J28).
 - Plug in the USB cable between the PC and the USB port (J18) on the board
 - Plug in the serial cable between the PC RS232 port and the DCE RS232 connector (J9) on the board. (The DTE RS232 connector (J10) can also be used for the serial connection but the STDIN/STDOUT setting must be modified in the Software Platform Settings.)

¹ ISE 8.1i latest Service Pack is available at <u>www.support.xilinx.com/swupdate</u>

² EDK 8.1 latest Service Pack is available at <u>www.xilinx.com/ise/embedded/edk_download.htm</u>

³ Xilinx Application Notes home page: <u>www.xilinx.com/xapp</u>

⁴ Refer to the ISE 8.1i Release Notes and Installation Guide <u>http://toolbox.xilinx.com/docsan/xilinx8/books/docs/irn/irn.pdf</u>



- Files:
 - Unzip the Xil3S500E_Serial_Flash_v81.zip file to a folder of your choosing, making sure there are no spaces in the pathname.
 - The version of XSPI included with this project is dated January 26, 2006. If you would like to update XSPI, unzip XAPP445 design file install_xspi.zip into the Xil3S500E_Serial_Flash_v81\FLASH_BURN directory. Run the self-extracting archive spi_setup.exe in this directory, extracting xspi.exe, the database .xdv file, and the readme file xspi_man.pdf in this directory. It is assumed that you have read and are familiar with the information contained in XAPP445. The FLASH_BURN directory contents are shown in Figure 1.



Figure 1 – FLASH_BURN Directory Contents after Extracting XSPI

- Project:
 - An XPS project is included with this reference design. For details regarding how to build this hardware platform yourself, please refer to Appendix A.

Experiment 1: Create and test a bitstream

This experiment uses a simple "Hello World" MicroBlaze design that prints to the UART. This experiment will ensure the software, jumpers, and cables are all installed properly. During this experiment, you will create a bitstream for this project and test it by downloading the bitstream directly to the FPGA via JTAG.

- 1. Launch Xilinx Platform Studio (XPS) by selecting Start → Programs → Xilinx Platform Studio 8.1 → Xilinx Platform Studio
- Select File → Open Project. Browse to the Xil3S500E_Serial_Flash_v81 folder, select system.xmp, and click Open.
- 3. Select the *Applications* tab. Select the **hello_world** project for BRAM initialization by right-clicking on the project and selecting **Mark to Initialize BRAMs**. No other applications should be marked for BRAM initialization. The *Applications* tab should look like Figure 2.





Figure 2 – Project: hello_world Selected for BRAM Initialization

4. Select **Device Configuration** → **Update Bitstream**. This will generate the MicroBlaze hardware platform, build the Board Support Package (BSP), compile the project, and initialize the bitstream with the application code. The result of this operation is an FPGA bitstream located at:

Xil3S500E_Serial_Flash_v81\implementation\download.bit

- 5. Launch a HyperTerminal connected to the RS232 COM port with the settings 115200 bps, 8 data bits, 1 stop bit, no parity, and no flow control. Alternatively, double-click on the com1_115200_8nln.ht file in the project directory to launch HyperTerminal with the appropriate settings.
- 6. Plug power into the Spartan-3E starter board. Turn the Power Switch to the ON position. The POWER LED should light.
- 7. In XPS, select **Device Configuration** → **Download Bitstream** which will download the download.bit file to the Spartan-3E FPGA. When the download is complete, the XC-DONE LED should light, and the HyperTerminal shows Hello World.





Figure 3 – Hello World Downloaded to FPGA

Experiment 2: Configure from serial flash

This experiment stores the bitstream from Experiment 1 in the serial flash and then configures the Spartan-3E FPGA from that flash.

When using iMPACT for configuration, the Start-Up Clock can be set to either CCLK or JTAG Clock, regardless of whether you are using JTAG or PROM for configuration. iMPACT is smart enough to infer the correct Start-Up clock based on the operation you are trying to perform. However, when using a non-iMPACT configuration method, such as is used for SPI or BPI configuration, you must take care to select CCLK as the Start-Up Clock, or the bitstream configuration will appear to hang at the end of the process.

XAPP445 states:

Before converting a Spartan-3E bitstream into a SPI-formatted PROM file, the designer must verify the bitstream was generated with the **bitgen** -g **StartupClk:Cclk** option.



- 1. In the XPS *Project* tab, under **Project Files**, open **Bitgen Options File:** etc\bitgen.ut.
- 2. Verify that the StartUpClk is set to CCLK

Now the project is ready for storing to the serial flash using XSPI.

- 3. Turn Power OFF.
- 4. If plugged in previously, disconnect the USB cable from the board.
- 5. Plug the Parallel Cable into the PC and the SPI connector (J12). If flyleads are used, match the signals as follows:

FLYLEADS	BOARD J12
VREF	VCC
GND	GND
TCK	SCK
TDO	SDO
TDI	SDI
TMS	SEL

- 6. Verify the MODE header (J30) is set to Boundary Scan mode (M1 jumper only M[2:0] = 101)
- 7. Turn power ON.

The bitstream must be prepared for SPI programming and then programmed.

8. If not already there, use Windows Explorer and browse to the Xil3S500E_Serial_Flash_v81\FLASH_BURN directory.

Several files exist in this directory, which are the command-line utilities and scripts required to program the SPI Flash. The SPI programming utility xspi.exe should have previously been unzipped into this directory by the user (see the Experiment Setup section).

Besides xspi.exe, the Xilinx utility **promgen** is also used, but this is available to the script through the PATH settings from the Xilinx installation, so it is not included in the directory.

Typically, these utilities are called through a Command Prompt. To simplify entering the required commands, a few batch files have been created. The xil3SE_store.bat batch file performs the following procedures:

- Generate a HEX file from the Hello_World download.bit (located in the project's implementation directory), naming it xil_3se.hex
- Erase, program, and verify the STMicro serial flash with xil_3se.hex



9. Double-click the xil3SE_store.bat batch file to launch the programming process.

A successful programming sequence shows 0 mismatches in the command window at the end of the operation, as shown in Figure 4.

w C:\WINDOWS\system32\cmd.exe - 🗆 × C:\tutorials\Xilinx\3S500E_Starter\Xil3S500E_Serial_Flash_v81\FLASH_BURN>cd . C:\tutorials\Xilinx\3S500E_Starter\Xil3S500E_Serial_Flash_v81\FLASH_BURN>echo_of r Release 8.1.02i - Promgen I.26 Copyright (c) 1995-2005 Xilinx, Inc. All rights reserved. 0x45480 (283776) bytes loaded up from 0x0 Using user-specified prom size of 512K Writing file "xil_3se.hex". Writing file "xil_3se.prm". ==> Checking SPI database [____xspi_database.xdv] - version [001 . 07 < 2006 January 25 >]: OK xspi(tm) Version 1.22 Copyright (c) 2003-2005 Xilinx, Inc. All rights reserved. Xilinx SPI Programming Utility exe | | ee l NOTICE: XSPI SOFTWARE FOR XILINX PROTOTYPE DEVELOPMENT USE ONLY жж ×× SOFTWARE PROVIDED "AS IS". ALL WARRANTIES, EXPRESS OR IMPLIED, ARE HEREBY DISCLAIMED. SOFTWARE NOT AUTHORIZED FOR USE IN PRODUCTION ENVIRONMENTS OR FOR USE IN OR WITH LIFE-SUPPORT OR MISSION-CRITICAL APPLIANCES, SYSTEMS, OR DEVICES. This software is for use with SPI devices listed in the XSPI device database; results when used with SPI devices from other manufacturers are unknown. Please contact Xilinx support for help with technical questions: ж× ×× . ** exe 🕴 http://www.xilinx.com/support/services/contact_info.htm ж× //** ===[Program notice/license accepted via -accept_notice command option]=== Start : Thu Feb 23 13:40:58 2006 Device ID code(s) check ====> [OK] -=> Operation: Erase => Operation: Program and Verify using file [xil_3se.hex] Programmed [283776] of [283776] bytes (w/ polling) Verified [283776] of [283776] bytes (0 errors) --> Total byte mismatches [0] (see [verify.txt]) Finish : Thu Feb 23 13:41:44 2006 Elapsed clock time (00:00:46) = 46 seconds 1 file(s) copied. Press any key to continue . . .



10. Press any key to close the command window.



- 11. Turn power OFF.
- 12. Unplug the Parallel Cable from J12 (required for proper SPI access by the FPGA).
- 13. Add a jumper to M2 (J30.5-6), setting the MODE to SPI Configuration Mode (M[2:0] = 0:0:1).

Note that the Vendor Select pins for the FPGA are strapped by default on the Xilinx Spartan-3E Starter board to the proper value for the STMicro M25P16 flash, which is VS[2:0] = 1:1:1.

- 14. If previously closed, re-open the HyperTerminal (com1_115200_8n1n.ht)
- 15. Turn power on. DONE should light indicating the FPGA is configured.

You should see the same application that we saw previously in Experiment 1, but now the bitstream is stored and configured from low-cost, SPI Flash!

16. Turn power OFF.

Experiment 3: Exercise serial flash from MicroBlaze

A sample application to test the serial flash is included. This application uses the same driver code that is used in Experiment 4's user application. The results are shown in HyperTerminal. The test application does the following:

- Read the manufacturer's ID
- Perform a bulk erase
- Write to all locations in the flash
- Read from all locations in the flash

*NOTE: Because of the jumper placement on J11 (1-2 and 3-4), the serial flash select signal (ROM_CS) is tied to both CSO_B (U3) and SEL (R12). This jumper placement is necessary to allow either the programming cable (using the SEL signal) or the FPGA (using the CSO_B signal) to access the serial flash. However, SEL is an unused pin in the MicroBlaze design. The default for unused pins in ISE is to pull them down (see bitgen defaults), which causes a conflict with the SPI controller. Therefore, the following statement was previously added to the bitgen.ut file to prevent SEL from interfering with the serial flash operation:

-g UnusedPin:Pullnone

1. In XPS, mark the **stmicro_spi_test** for BRAM initialization and unmark all others, as shown in Figure 5.



Project	Applications	IP Catalog	
Software	Projects		
- Carl	ld Software Appli	cation Project	
🛛 🔤 🎆 De	alt: microblaze	_0_bootloop	
📔 🔤 🎆 De	efault: microblaze	_0_xmdstub	
🗄 🎇 Project: hello_world			
🗄 🌄 Project: stmicro_spi_test			
🖶 🎇 Project: spi_user_app			
🗄 🎇 Project: spi_bootloader			
1			

Figure 5 – Project: stmicro_spi_test Marked for BRAM Initialization

- 2. Take a moment to browse the code to become familiar with what the application is doing.
- 3. Select **Device Configuration** → **Update Bitstream** to compile this project and create a new download.bit bitstream with this application.

Before downloading, a few changes are made to the board.

- 4. Return the J30 jumpers to Boundary Scan Mode (M1 only installed).
- 5. Do one of the following:
 - a. Plug the Parallel Cable to the JTAG connector (J28).
 - b. Unplug the Parallel Cable from the back of the computer and plug in the USB cable.
- 6. Turn power ON.
- 7. If previously closed, re-open the HyperTerminal (com1_115200_8n1n.ht)
- 8. Download the bitstream to the board.

The complete test takes less than one minute to run. Results are shown in Figure 6.





Figure 6 – Serial Flash Test Results

9. Turn power OFF.

Experiment 4: Bootload MicroBlaze from serial flash

This experiment shows how a MicroBlaze application is stored in serial flash device and then bootloaded after configuration. The process to be completed in this experiment is:

- Configure the FPGA from serial flash with a bitstream containing a MicroBlaze hardware platform and BRAM contents initialized with a bootloader application.
- The bootloader accesses a pre-determined location in the serial flash and copies a stored user application from flash to DDR.
- The bootloader application then jumps to the user application in DDR and begins running.

The user application does the following:

- Reads and displays a 16-character location in serial flash and displays the user string.
- Prompts the user to enter a new string



- Accepts a new 16-character string from the UART
- Stores the string in serial flash
- Instructs you to reconfigure to show that your new string was stored in flash

This experiment uses the serial flash for three separate functions: FPGA configuration, MicroBlaze application bootloading, and user data storage.

- 1. Take a moment to browse through the code for the two applications that are used during this experiment:
 - **Project: spi_user_app**. A binary version of this application will be stored in serial flash and later bootloaded.
 - **Project: spi_bootloader**. This application is stored in BRAM and launches immediately after the MicroBlaze is configured.
- 2. In XPS, mark the **Project: spi_bootloader** for BRAM Initialization and unmark all others.



Figure 7 – SPI Bootloader Marked for BRAM Initialization

3. Right-click on **Project: spi_user_app** and select **Set Compiler Options...** Note that Program Start Address is set to 0x2200000 which is the base address of the DDR memory. Click **Cancel** to close this window.



📀 Set Compiler Options	×
Compiler Tools: mb-gcc	
Environment Debug and Optimization Paths Advanced	
Application Mode Executable C XmdStub xmdstub_peripheral: debug_module	
Output ELF file 3S500E_Serial_Flash_v81\spi_user_app\executable.elf Browse	
Linker Script Use Custom Linker Script Browse Browse	
Program Start Address 0x22000000	
Heap Size	
OK Cancel	

Figure 8 – spi_user_app Compiler Options

4. Right-click on **Project: spi_user_app** and select **Build Project**. This creates the ELF file: xil3s500E_Serial_Flash_v81\spi_user_app\executable.elf

This application must be converted from ELF to binary format for use with the programming utilities.

5. Open a cygwin shell window by selecting Project \rightarrow Launch EDK Shell...

A script is provided for ease of use. The script uses the mb-objcopy utility to convert spi_user_app\executable.elf to FLASH_BURN\spi_user_app.b. Options are included to exclude several initialization vectors that will be included as part of the bootloader. The syntax format to perform this on the command line of the shell window is:

mb-objcopy -0 binary <options> <ELF file input> <binary file to output>



6. In the shell window, type the following: ./make_bin.sh <enter>

🔤 /cygdrive/c/tutorials/Xilinx/35500E_Starter/Xil35500E_Serial_Flash_v81	
EDK Shell Xilinx EDK 8.1.01 Build EDK_I.19.5 Copyright (c) 1995-2006 Xilinx, Inc. All rights reserved.	
Analyzing Cygwin versions Xilinx EDK detected Cygwin installation v1.5.11(0.116/4/2) on your machine. This Cygwin (c:\\EDK71\\cygwin\\bin) will be used to run Xilinx EDK tools.	
024974@USSDG01NB024974 /cygdrive/c/tutorials/Xilinx/3S500E_Starter/Xil3S500E ial_Flash_v81 \$./make_bin.sh ************************************	_Ser

0249740USSDG01NB024974 /cygdrive/c/tutorials/Xilinx/3S500E_Starter/Xil3S500E ial_Flash_v81 5	_Ser

Figure 9 – Creating Binary of the SPI User Application

7. Type '**exit**' to close the command shell.

The file spi_user_app.b should be approximately 7 KB. If the options to exclude the initialization vectors are not included, this file will be over 500 MB, and the rest of this experiment will fail.

8. Browse to the FLASH_BURN directory. Make sure that spi_user_app.b is 7KB in size.

Now the bitstream with the hardware platform and bootloader application is created.

9. Select Tools → Update Bitstream to compile the spi_bootloader project and create a new implementation/download.bit bitstream with the bootloader application initialized into BRAM.

Similar to Experiment 2, command-line utilities are used to program this information into the serial flash. The bitstream as well as the **spi_user_app** application binary must be stored in the flash.

10. Browse to the Xil3S500E_Serial_Flash_v81\FLASH_BURN directory

The top-level script used in this experiment is called Xil3SE_bootload.bat. This script first creates an MCS image for download.bit. Then, it calls another script, prog_multiple.bat, which is called with four variables: the bitstream, the user



application binary file, the address where to put the user application in the SPI flash, and the type of flash.

11. In Explorer, right-click on Xil3SE_bootload.bat and select Edit.

Note that the **spi_user_app** will be stored at address 0x60000 in the serial flash. This is the beginning of Sector #6, which matches the location from which the bootloader application will read (see BOOT_SECTOR constant in the spi_bootloader project's bootload.c file).

12. Close the script.

The prog_multiple.bat does the following:

- Converts the user application binary to the MCS format
- Combines the bitstream MCS and the user application MCS into a single MCS.
- Erase, program, and verify the flash
- 13. Unplug the USB from the board.
- 14. Plug the Parallel Cable into the PC and the SPI port (J12).
- 15. Make sure the MODE (J32) jumpers are in Boundary Scan Mode (M1 installed).
- 16. Turn power ON.
- 17. Double-click the Xil3SE_bootload.bat script. The files are programmed into the flash. In the command window, make sure that the total byte mismatches are 0:

--> Total byte mismatches [0] (see [verify.txt])

- 18. Press any key to close the command window.
- 19. Turn off power to the board.
- 20. Remove the parallel cable.
- 21. If closed, launch HyperTerminal (com1_115200_8n1n.ht).
- 22. Change the MODE (J32) to SPI (M1 and M2 installed, M0 uninstalled).
- 23. Turn board power on. The DONE light should go on, and you will see HyperTerminal display as shown in Figure 10.



🗞 com1_115200_8n1n - HyperTerminal	
File Edit View Call Transfer Help	
	<u>^</u>
	1
BootingDone!	
SPI User Application	
User String stored in flash =	
Enter 18-byte string to be stored in flash	
<u> </u>	v
Connected 0:00:07 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo	.:

Figure 10 - Configured and Bootloaded from SPI Flash

24. Enter exactly 16 characters. For example, "0123456789abcdef" is exactly 16 characters. The data is then written in the SPI flash.



Figure 11 – Initial User String Written to Flash



25. Press the PROGRAM button. The FPGA will again configure from SPI flash. This time, however, the user application finds the string previously typed (see Figure 12).



Figure 12 – User String Retrieved from SPI Flash

26. Turn the board power off.

Revision History

Date	Version	Revision
02/23/06	8.1	Initial release.



Appendix A: Building the Hardware Platform

This appendix details how to build the MicroBlaze hardware platform used in this example.

- 1. Launch XPS.
- 2. Select Base System Builder wizard. Click OK.

۶	ilinx Platform Studio	
Г	reate new or open existing project	l
	Base System Builder wizard (recommended)	
	Blank XPS project	
	C Open a recent project	
	Browse for More Projects	l
В	wse installed EDK examples (projects) <u>here</u>	
	OK Cancel	

Figure 13 – Entering Base System Builder

3. Browse to an acceptable location to build this project. Since the XBD file for this board is included in the default EDK installation, there is no need to provide a User Repository. Click **OK**.



📀 Create New XPS Project using BSB Wizard	×
New project	1
Project file	
500E_Starter/Xil3S500E_Serial_Flash_v81/system.xmp Browse	
Advanced options (optional)]
Browse	
OK Cancel	

Figure 14 – BSB Project Name

4. Select the option to create a new design. Click Next >.



Figure 15 – Creating a New Design



5. Select the Xilinx Spartan-3E Starter Board, either Rev C or D (depending on which board you have – this project is built based on Rev C). Click Next >.

ase System Builder - Select Board	🔶 Bas
act a target development board:	Selec
	C -1
elect board	
	Boi
eard vendel. Jointh	- Doc
	Doc
	Boa
rote, visit the vendor website for additional board support materials.	
	ver
ownload Third Party Board Definition Files	Dou
I would like to create a system for a custom board	<u> </u>
oard description	Boa
Spartan-3E Starter Kit Board utilizes Xilinx Spartan-3E XC3S500E-4FG320 device. The board includes 2 RS232 serial ports, 4 DIP switches, 4 push buttons, 8 LEDs, VGA port, character LCD display, PS/2 port, push button rotary encoder, SPI analog to digital converter, SPI digital to analog converter, 10/100 Ethernet port, 2MB SPI flash, 16 MB of parallel NOR flash and 32 MB DDR SDRAM. Push button South(RESET) is used as system reset.	Sp ind LC MI
fore Info < Back Next > Cancel	Мо

Figure 16 – Target Development Board

6. Since this is a MicroBlaze design, no changes need to be made to the *Select Processor* screen. Click **Next>**.



spartan3e XC33500e FG320 4 elect the processor you would like to use in this design: Processors Image: PowerPC Not supported by this device Image: Processor description The MicroBlaze(TM) 32-bit soft processor is a RISC-based engine with a 32 register by 32 bit LUT RAM-based Register File, with separate instructions for data and memory access. It supports both on-chip peripheral memory. All peripherals are implemented on the FPGA fabric and operate off the on-chip peripheral bus (OPB).	Architecture:	Device:	Package:	Speed grad	de:
elect the processor you would like to use in this design: Processors PowerPC Not supported by this device Processor description The MicroBlaze[TM] 32-bit soft processor is a RISC-based engine with a 32 register by 32 bit LUT RAM-based Register File, with separate instructions for data and memory access. It supports both on-chip BlockRAM and/or external memory. All peripherals are implemented on the FPGA fabric and operate off the on-chip peripheral bus (OPB).	spartan3e	XC3S500e	▼ FG320	-4	*
PowerPC by this device OPB OPB <t< td=""><td>elect the process Processors</td><td>or you would like to u</td><td>use in this design:</td><td></td><td></td></t<>	elect the process Processors	or you would like to u	use in this design:		
Processor description The MicroBlaze(TM) 32-bit soft processor is a RISC-based engine with a 32 register by 32 bit LUT RAM-based Register File, with separate instructions for data and memory access. It supports both on-chip BlockRAM and/or external memory. All peripherals are implemented on the FPGA fabric and operate off the on-chip peripheral bus (OPB).	MicroBlaze PowerPC Not suppor by this devi	ted Cepo Arbiter W100 E-Net			
	Processor descrip The MicroBlaze LUT RAM-base supports both o the FPGA fabric	ption e(TM) 32-bit soft proc ed Register File, with on-chip BlockRAM ar c and operate off the	essor is a RISC-based e separate instructions for nd/or external memory. / on-chip peripheral bus (ngine with a 32 register data and memory acces All peripherals are implen OPB).	by 32 bit ss. It nented on

Figure 17 – MicroBlaze Processor Selected

7. Change the *Local Memory* setting to 16 KB. All other settings' defaults are appropriate. Click **Next>**.



Base System Builder - Configure MicroBlaze MicroBlaze System wide settings Reference clock frequency: Processor-Bus clock frequency: 50.00 MHz Beset polarity: Active High	?×
Processor configuration Debug I/F © On-chip H/W debug module C XMD with S/W debug stub No debug Local memory Data and Instruction: (Use BRAM) IGKE Cache setup © No Cache © Enable OPB cache © Enable cache link	
Enable floating point unit (FPU)	
More Info < Back Next > Can	cel

Figure 18 – Configure MicroBlaze

Note that the following screens may not appear the same on your machine, depending on your screen resolution setting.

8. Change the *Baudrate* setting for both RS232 peripherals to 115200. Only one RS232 peripheral is required, but this project was built with both. Click **Next>**.



📀 Base System Builder - Configure IO Interfaces	<u>? ×</u>
The following external memory and IO devices were found on your board:	
Xilinx Spartan-3E Starter Board Revision C	
Please select the IO devices which you would like to use:	
-10 devices	
- ▼ RS232_DCE	
Peripheral: OPB UARTLITE	Data Sheet
Baudrate (bits per seconds): 115200	
Data bits: 8	
Parity: NONE	
Use interrupt	
RS232_DTE	Data Sheet
Peripheral: OPB UARTLITE	
Baudrate (bits per seconds):	
Data bits: 8	
Parity: NONE 💌	
Use interrupt	
More Info < Back Nex	t > Cancel

Figure 19 – Configure UARTs

9. Deselect the LEDs_8Bit, DIP_Switches_4Bit, Buttons_4Bit, and Flash_16Mx8. Deselecting the LEDs, DIPs, and Buttons is done simply to make the project simpler. Deselecting the flash is a requirement since the parallel flash and serial flash physically share a pin. They cannot be connected through this wizard, although it is possible to do it with user logic. Click **Next>**.



Base System Builder - Configure Additional IO Interfaces		<u>? ×</u>
The following external memory and 10 devices were found on your board: Xilinx Spartan-3E Starter Board Revision C	:	
Please select the IO devices which you would like to use: IO devices		
LEDs_8Bit	Da	ata Sheet
DIP_Switches_4Bit	Da	ata Sheet
Buttons_4Bit	Da	ita Sheet
FLASH_16Mx8	De	ata Sheet
More Info < Back No	ext >	Cancel

Figure 20 – Deselecting Peripherals

10. Check the box for *SPI_FLASH*. Deselect the *Ethernet_MAC* (for project simplification only). Click Next>.



The following external memory and IO devices were found on your board: Xilinx Spartan-3E Starter Board Revision C Please select the IO devices which you would like to use: IO devices IO devices Data Sheet Peripherat: OPB SPI I include receive and transmit FIFOs Data Sheet IV use interrupt Data Sheet Note Note	📀 Base System Builder - Configure Additional IO Interfaces	<u>? ×</u>
SPI_FLASH Peripheral: OPB SPI Include receive and transmit FIFOs Use interrupt Data Sheet Note Data Sheet Note Data Sheet Note	The following external memory and IO devices were found on your board: Xilinx Spartan-3E Starter Board Revision C Please select the IO devices which you would like to use: 10 devices	
Use interrupt Use interrupt DDR_SDRAM_16Mx16 Data Sheet Note Ethernet_MAC Data Sheet Note	Include receive and transmit FIFOs	Data Sheet
Ethernet_MAC Data Sheet Note	Use interrupt DDR_SDRAM_16Mx16 Peripheral: OPB DDR	Data Sheet Note
	Ethernet_MAC	Data Sheet Note

Figure 21 – Select SPI_FLASH and DDR_SDRAM

11. Click **Next>** since no additional peripherals are added.



Base Syster	n Builder - Add Internal Pe	ripherals		<u>? ×</u>
Add other periph "Add Peripheral	nerals that do not interact with o " button to select from the list of	ff-chip component available periphe	ts. Use the rals.	
If you do not wi:	sh to add any non-10 peripherals	s, click the "Next"	button.	
				Add Peripheral
-Peripherals—				
More Info		< Back	Next >	Cancel

Figure 22 – No Internal Peripherals Added

12. Set STDIN/STDOUT to the RS232 peripheral that you would like to use. Click **Next>**.

Note that the project was built with the Memory and Peripheral Tests included. These applications still reside in the project directory, but they have been deleted from the *Applications* tab for project simplification purposes.



📀 Base System Builder - Software Setup 🔹 🤶
Devices to use as standard input and standard output
STDIN: RS232_DCE
STDOUT: RS232_DCE
Courses and the time as leasting
Select the sample C application that you would like to have generated. Each application will include a linker script.
J♥ Memory test
Illustrate system aliveness and perform a basic read/write test to each memory in your system
Perinheral selftest
Perform a simple self-test for each peripheral in your system.
More Info < Back Next > Cancel

Figure 23 – Set STDIN/OUT and Applications

13. Click Next>.



🔶 Base Systen	n Builder - Configure Mem	ory Test Applicat	ion	<u>?</u> ×
The simple Mem to your memory o	ory Test application will illustrat Jevices.	e system aliveness a	ind perform a basic	read/write test
-MemoryTest-				
Select the me	mory devices which will be use	d to hold the followin	ng program sections	:
Instruction:	ilmb_cntlr			
Data:	dlmb_cntlr		•	
Stack/Heap:	dlmb_cntlr		•	
If you have pl	aced the Instruction or Data se	ction of this program	in an external mem	ory, you must
use a debugg	er, bootloader, or ACE file to ini	tialize memory before	e you can run this p	rogram!
More Info		< Back	Next >	Cancel

Figure 24 – Configure Memory Test

14. Click Next>.



📀 Base System	n Builder - Configure Perip	heral Test Applic	cation	? X
The Peripheral S a selftest function	elftest application includes a sin n exists in the driver the periph	imple self test for ea eral)	ach periperhal in ye	our system (if such
-PeripheralTes				
Select the me	mory devices which will be use	d to hold the follow	ing program sectio	ins:
Instruction:	DDR_SDRAM_16Mx16			3
Data:	DDR_SDRAM_16Mx16		-	- E
Stack/Heap:	DDR_SDRAM_16Mx16			3
-WARNING-				
Instruction an bootloader, or	pected size or this application, d Data sections in an external r ACE file to initialize the memory	nemory. You will ha y.	nded that you plac ave to use a debu	gger,
More Info		< Back	Next >	Cancel

Figure 25 – Configure Peripheral Test

15. Click Generate.



System clock frequi Debug interface: O On Chip Memory : Total Off Chip Mem - DDR_SDRAM_1	ency: 50.000000 MHz h-Chip HW Debug Module 16 KB ory : 32 MB 6Mx16 = 32 MB elow have been automatica	illy assigned. You c	an modify them using the
OPB Bus : OPB	V20 Inst. name: mb o	pb Attached Co	omponents:
 Core Name	Instance Name	Base Addr	High Addr
opb_mdm	debug_module	0x41400000	0x4140FFFF
opb_uartlite	RS232_DCE	0x40600000	0x4060FFFF
opb_uartlite	RS232_DTE	0x40620000	0x4062FFFF
opb_spi	SPI_FLASH	0x40A00000	0x40A0FFFF
opb_ddr	DDR_SDRAM_16Mx1	0x22000000	0x23FFFFFF
LMB Bus : LMB_	V10 Inst. name: ilmb	Attached Comp	onents:
Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	ilmb_ontlr	0x00000000	0x00003FFF
LMB Bus : LMB_	V10 Inst. name: dlmb	Attached Comp	onents:
 Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	dimb_cntir	0x00000000	0x00003FFF

Figure 26 – System Summary: Ready to Generate

16. Click **Finish** to close the BSB wizard.



📀 Base System Builder - Finish	<u>? ×</u>
	The Base System Builder has successfully generated your embedded system! Click the Finish button to return to XPS to compile your hardware system and software application.
C:\tutorials\Xilinx\3S500E_Starter\ C:\tutorials\Xilinx\3S500E_Starter\ C:\tutorials\Xilinx\3S500E_Starter\ C:\tutorials\Xilinx\3S500E_Starter\ C:\tutorials\Xilinx\3S500E_Starter\ C:\tutorials\Xilinx\3S500E_Starter\ C:\tutorials\Xilinx\3S500E_Starter\ C:\tutorials\Xilinx\3S500E_Starter\ C:\tutorials\Xilinx\3S500E_Starter\ Save settings file:	XI3S500E_Serial_Flash_v81\system.mhs XiI3S500E_Serial_Flash_v81\data\system.ucf XiI3S500E_Serial_Flash_v81\etc\fast_runtime.opt XiI3S500E_Serial_Flash_v81\etc\download.cmd XiI3S500E_Serial_Flash_v81\system.mss XiI3S500E_Serial_Flash_v81\TestApp_Memory\src\TestApp_ XiI3S500E_Serial_Flash_v81\TestApp_Memory\src\TestApp_ XiI3S500E_Serial_Flash_v81\TestApp_Peripheral\src\TestAp
C:\tutorials\Xilinx\3S500E_Start The settings file contains all the loaded in a future wizard sessior	er\XiI3S500E_Serial_Flash_v81\system.bsb user's selections and inputs in the wizard session. It can be n
More Info	< Back Finish Cancel

Figure 27 – Embedded System Complete

17. A couple minor modifications need to be made prior to building the system, so select *Start using Platform Studio* and click **OK**.



📀 The Next Step	X
What would you like to do next?	
C Configure drivers and libraries (Software Platform)	
C Download the design to the board and test it	
C Edit the test application generated by BSB	
Start using Platform Studio	
Remember my selection and don't show this again	OK Cancel

Figure 28 – Start using Platform Studio

Platform Studio will appear as shown in the screen below.



Figure 29 – Project View in XPS

The file that needs to be edited is the etc/bitgen.ut file that sets the bitstream generation options. However, this file isn't copied to the project until after the first build.



18. Select **Hardware** → **Generate Bitstream**. You should immediately see the following message in the Console Window:

```
Copied c:\EDK81/data/xflow/bitgen_spartan3e.ut to etc directory
```

- 19. Once you see this message, click **Project** -> **Terminate Running Process**.
- 20. Open the bitgen.ut file by double-clicking on it under the *Project* tab, *Project Files*, as shown in the image below.

Project	Applications IP Catalog
Platform	
🚊 Proje	ct Files
М М	HS File: system.mhs
М М	SS File: system.mss
UU	CF File: data/system.ucf
l …iN	1PACT Command File: etc/download.cmd
In	nplementation Options File: etc/fast_runtime.opt
Bi	itgen Options File: etc/bitgen.ut
🗄 Proje	ect Options
D	evice: xc3s500efg320-4
- N	etlist: TopLevel
In	nplementation: XPS
н	DL: VHDL
Si 🗠 Si	im Model: BEHAVIORAL
🛓 🗄 🛛 Refe	rence Files
l ⊡-Lo	og Files
́	eport Files
1	

Figure 30 – Opening the bitgen.ut file

- 21. One option is modified, and one option is added. Change the **StartUpClk** setting from **JTAGCLK** to **CCLK**. Add the **UnusedPin** option with a setting of **PULLNONE**. The syntax for these two lines is shown below.
 - -g StartUpClk:CCLK
 - -g UnusedPin:PULLNONE

This completes the design of the hardware platform.