System Ace Tutorial

This is a basic System Ace tutorial that demonstrates two methods to produce a System ACE file; the use of the System Ace File Generator (GenACE) and through IMPACT. Also, the steps to Format the CF card to configure an FPGA with a bitstream and an executable (ELF) file will be described in detail.

The System ACE CF configures devices using the Boundary-Scan (JTAG) instructions and a Boundary-Scan Chain. System ACE CF is a two-chip solution that requires the System ACE controller, and either a CompactFlash card or a one-inch Microdrive disk drive. In this tutorial the System ACE CF, CompactFlash card solution will be used to configure the FPGA.

So, the first step is to generate a BSB that contains the blocks needed to carry this out. This tutorial will use the Base System Builder (BSB) to build the project. Follow the steps seen below to do this:

Step 1: Launching the Embedded Development Kit (EDK).

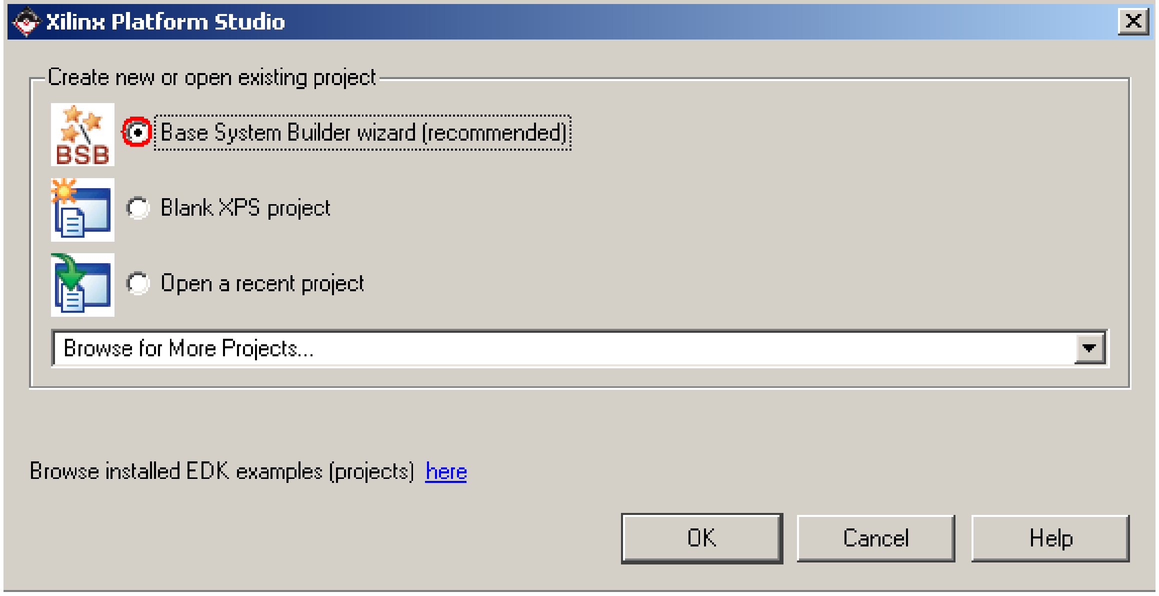
To launch EDK, go to *Start -> Programs -> Xilinx ISE Design Suite 10.1 -> EDK -> Xilinx Platform Studio*

Step 2: Creating BSB project

A Xilinx Platform Studio pop up box will appear on the screen. You have three options:

- Create a project using the Base System Builder (BSB)
- Create a blank project
- Open a recent project.

This tutorial will be using the Base System Builder, so select the BSB (shown below) and click “Next” to continue.



Xilinx Platform Studio

Create new or open existing project

● Base System Builder wizard (recommended)

○ Blank XPS project

○ Open a recent project

Browse for More Projects...

Browse installed EDK examples (projects) here

Step 3: Saving your project.

Save your project and select “Next” to continue.
Note: Make sure you path contains no spaces in it, as this may cause problems.

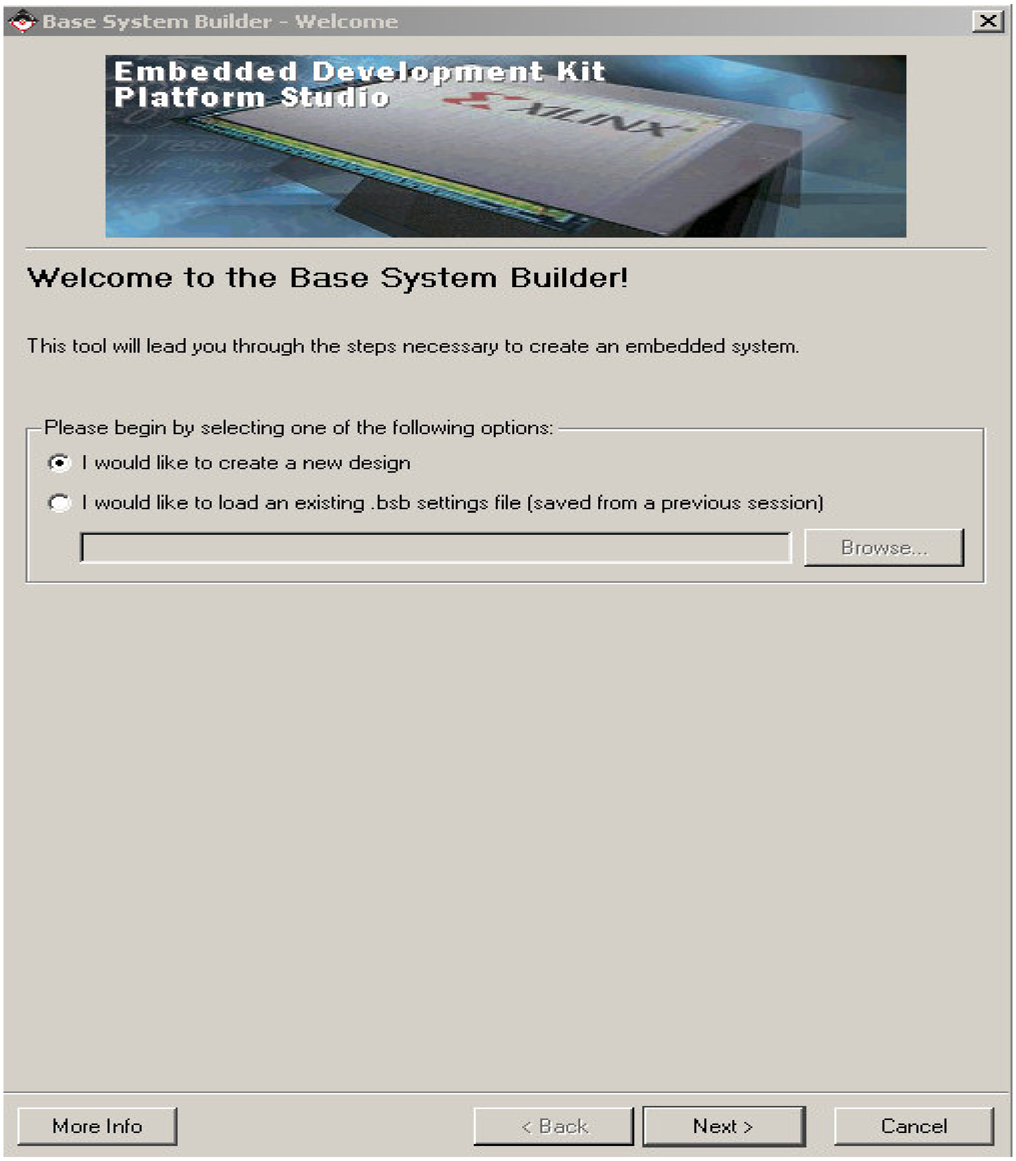
Step 4: Choosing your board in the BSB.

You have two options here:

- Create a new design
- Use an existing BSB design settings.

In this tutorial, a new BSB design will be created so click “I would like to create a new design” and click “Next” to continue. This is shown below:

Select "I would like to create a new design", Click Next to continue.



Welcome to the Base System Builder!

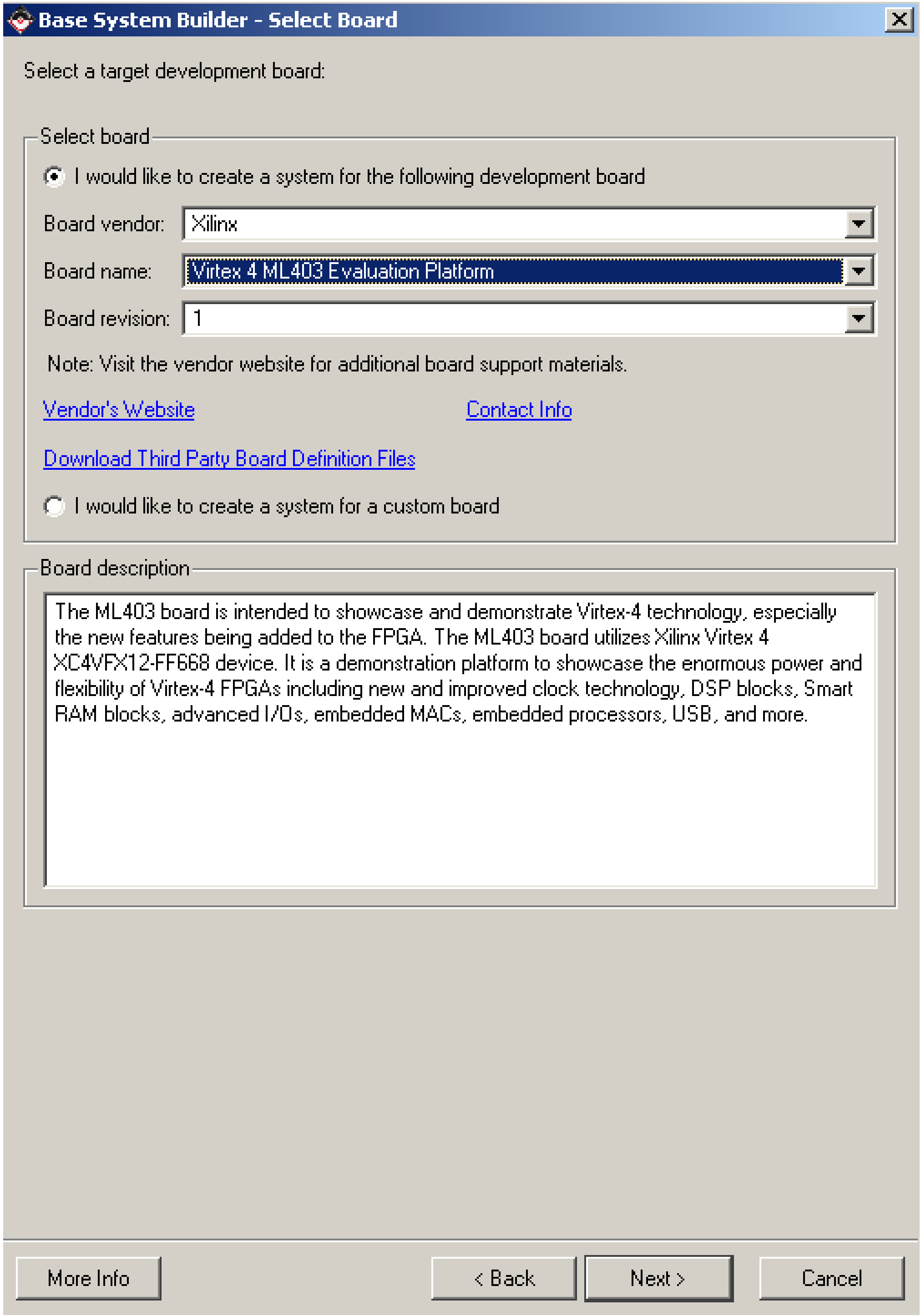
This tool will lead you through the steps necessary to create an embedded system.

Please begin by selecting one of the following options:

● I would like to create a new design

○ I would like to load an existing .bsb settings file (saved from a previous session)

Select the ML403 Board as shown below, Select Next to Continue



Base System Builder - Select Board

Select a target development board:

Select board

● I would like to create a system for the following development board

Board vendor: Xilinx

Board name: Virtex 4 ML403 Evaluation Platform

Board revision: 1

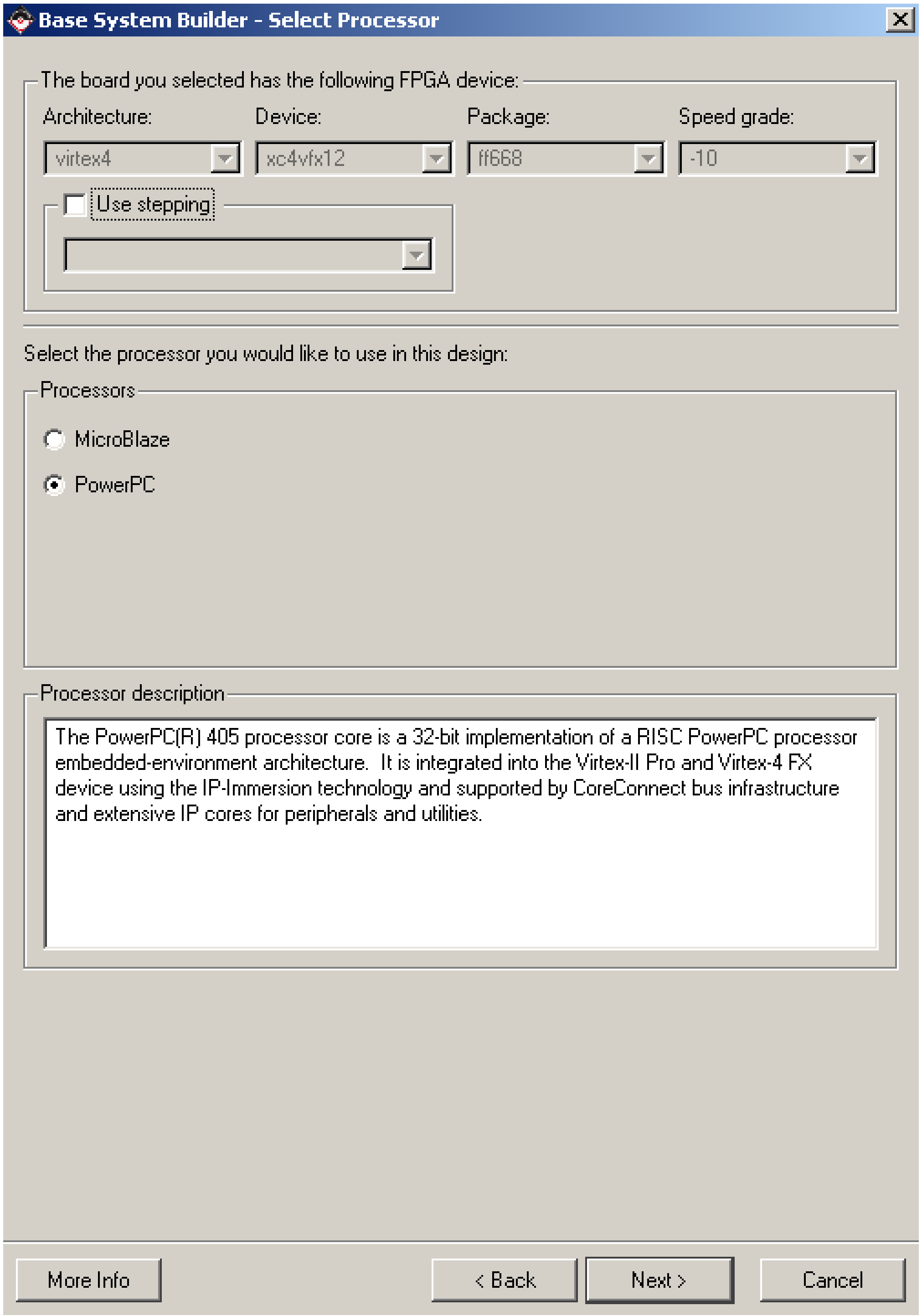
Note: Visit the vendor website for additional board support materials.

○ I would like to create a system for a custom board

Board description

The ML403 board is intended to showcase and demonstrate Virtex-4 technology, especially the new features being added to the FPGA. The ML403 board utilizes Xilinx Virtex 4 XC4VFX12-FF668 device. It is a demonstration platform to showcase the enormous power and flexibility of Virtex-4 FPGAs including new and improved clock technology, DSP blocks, Smart RAM blocks, advanced I/Os, embedded MACs, embedded processors, USB, and more.

Select the PowerPC, Press Next to Continue.



Base System Builder - Select Processor

The board you selected has the following FPGA device:

Architecture:	Device:	Package:	Speed grade:
virtex4	xc4vfx12	ff668	-10

□ Use stepping

Select the processor you would like to use in this design:

Processors

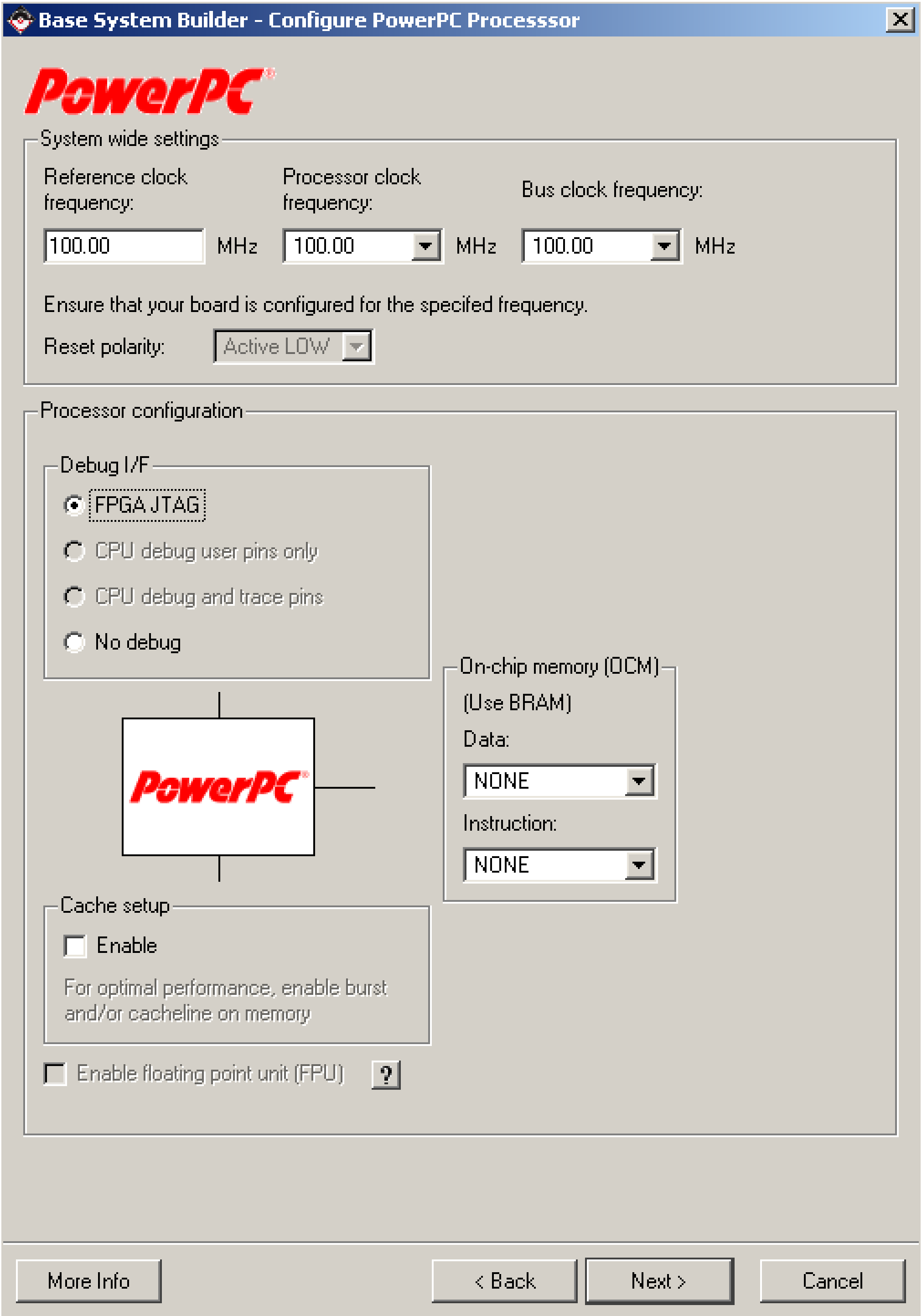
○ MicroBlaze

● PowerPC

Processor description

The PowerPC(R) 405 processor core is a 32-bit implementation of a RISC PowerPC processor embedded-environment architecture. It is integrated into the Virtex-II Pro and Virtex-4 FX device using the IP-Immersion technology and supported by CoreConnect bus infrastructure and extensive IP cores for peripherals and utilities.

Select "NONE" for the Local Memory, Press Next to continue.



Base System Builder - Configure PowerPC Processsor

System wide settings

Reference clock frequency: 100.00 MHz

Processor clock frequency: 100.00 MHz

Bus clock frequency: 100.00 MHz

Ensure that your board is configured for the specifed frequency.

Reset polarity: Active LOW

Processor configuration

Debug I/F

● FPGA JTAG

○ CPU debug user pins only

○ CPU debug and trace pins

○ No debug

On-chip memory (OCM)

(Use BRAM)

Data: NONE

Instruction: NONE

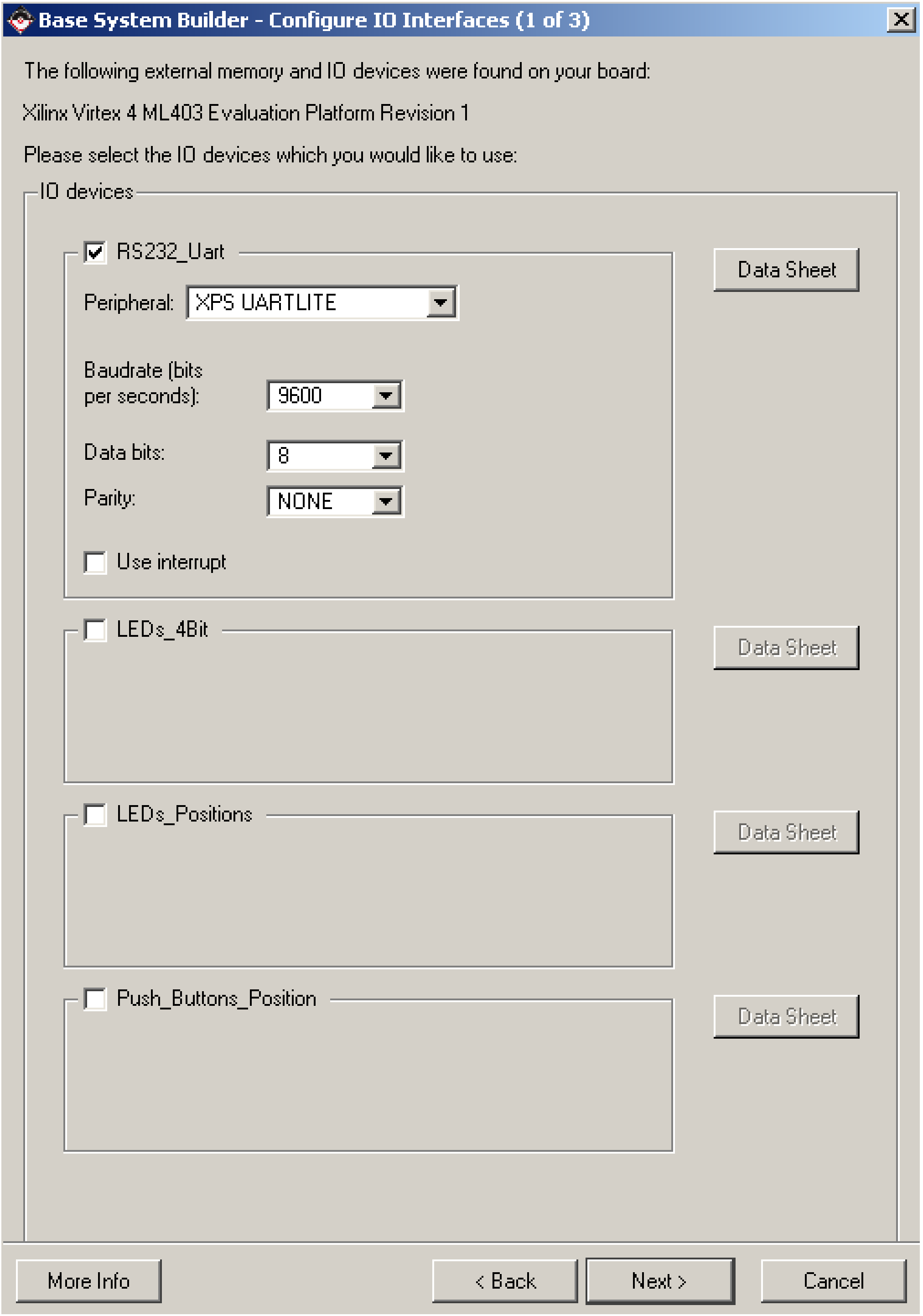
Cache setup

□ Enable

For optimal performance, enable burst and/or cacheline on memory

□ Enable floating point unit (FPU)

Select the peripherals as shown below, Press Next to Continue.



Base System Builder - Configure IO Interfaces (1 of 3)

The following external memory and IO devices were found on your board:

Xilinx Virtex 4 ML403 Evaluation Platform Revision 1

Please select the IO devices which you would like to use:

IO devices

☑ RS232_Uart

Peripheral: XPS UARTLITE

Baudrate (bits per seconds): 9600

Data bits: 8

Parity: NONE

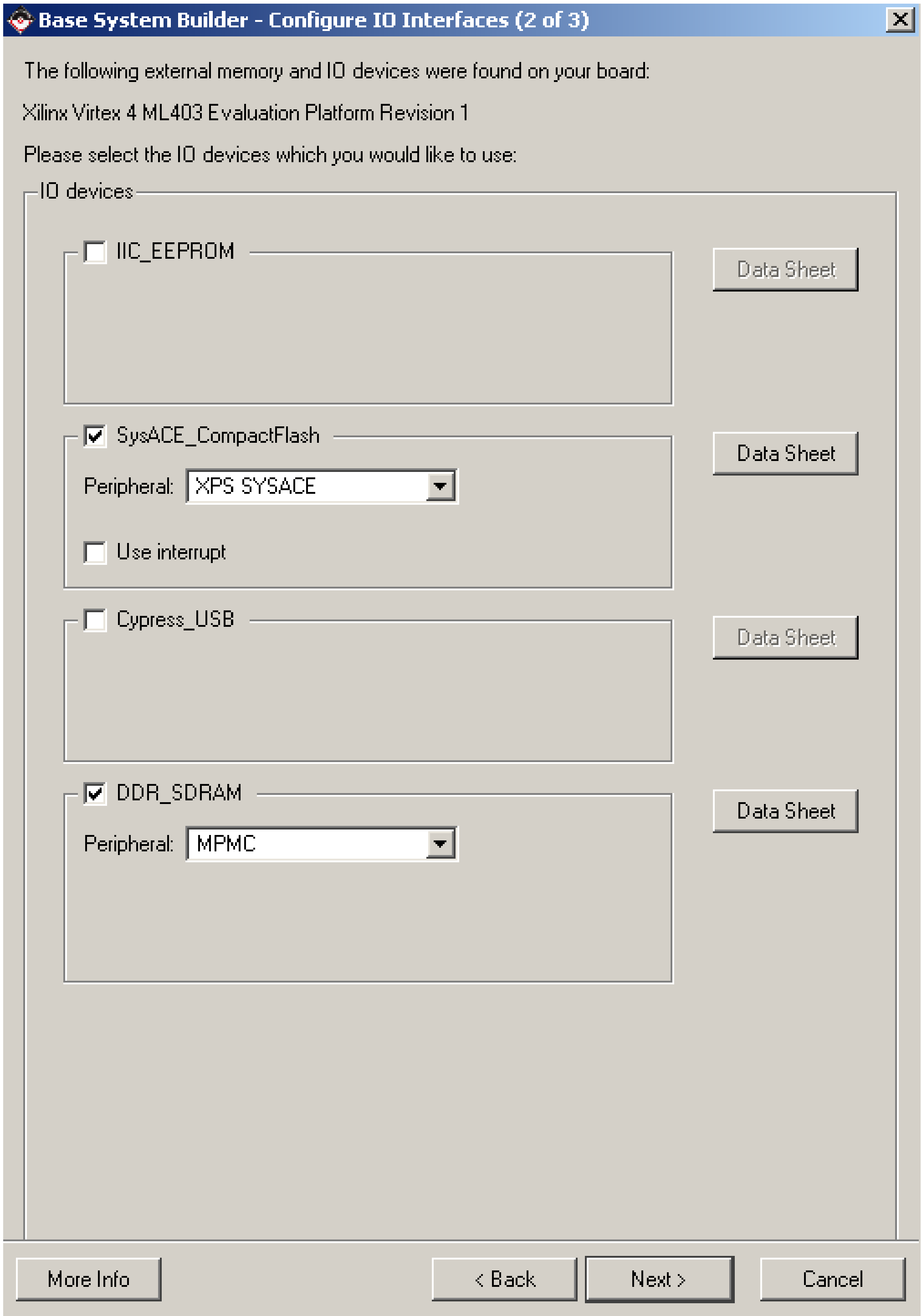
☐ Use interrupt

☐ LEDs_4Bit

☐ LEDs_Positions

☐ Push_Buttons_Position

Select the peripherals as shown below, Press Next to Continue.



Base System Builder - Configure IO Interfaces (2 of 3)

The following external memory and IO devices were found on your board:

Xilinx Virtex 4 ML403 Evaluation Platform Revision 1

Please select the IO devices which you would like to use:

IO devices

□ IIC_EEPROM

☑ SysACE_CompactFlash

Peripheral: XPS SYSACE

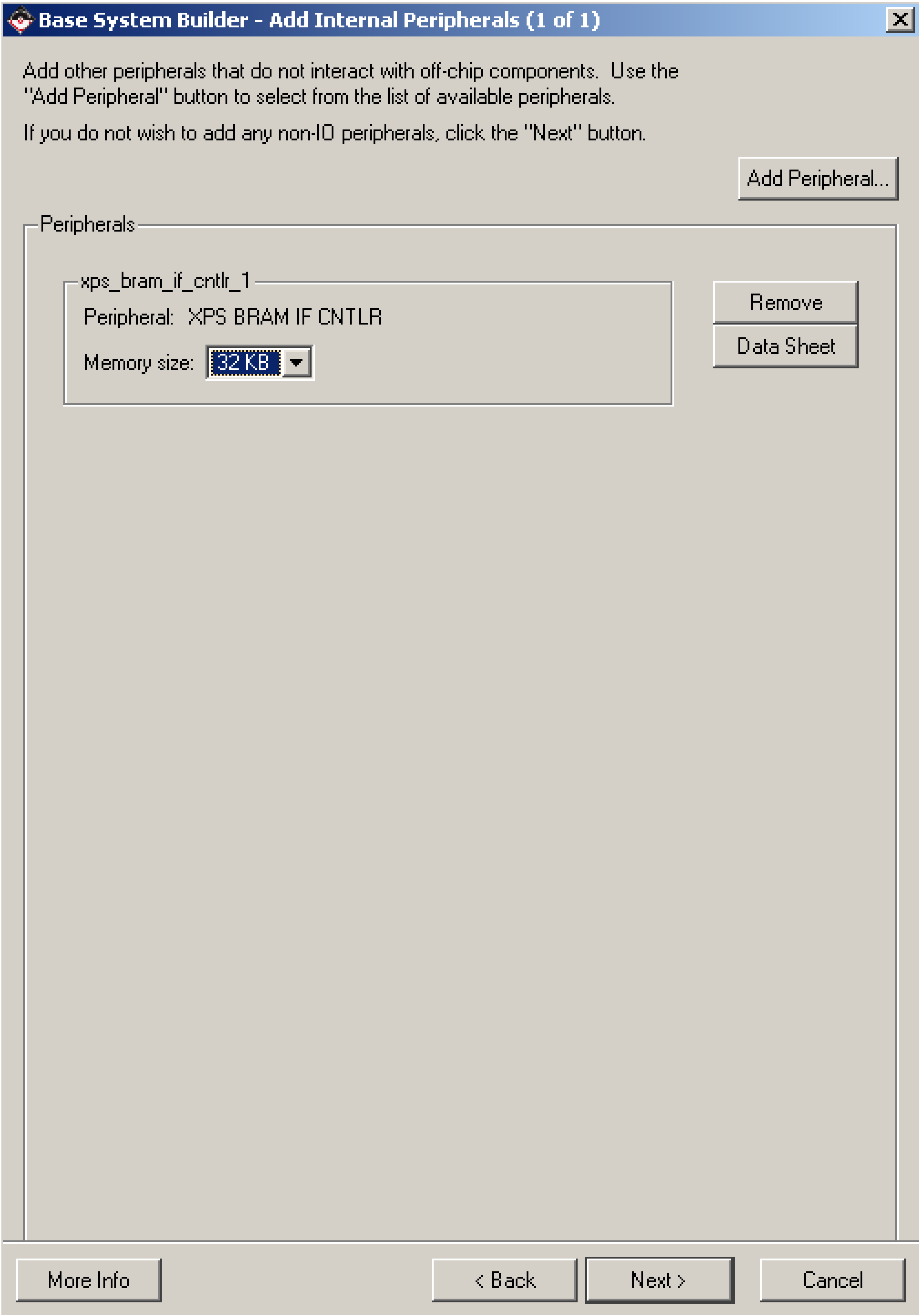
□ Use interrupt

□ Cypress_USB

☑ DDR_SDRAM

Peripheral: MPMC

Select memory size of 32K, Select "Next" to continue.



Base System Builder - Add Internal Peripherals (1 of 1)

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.

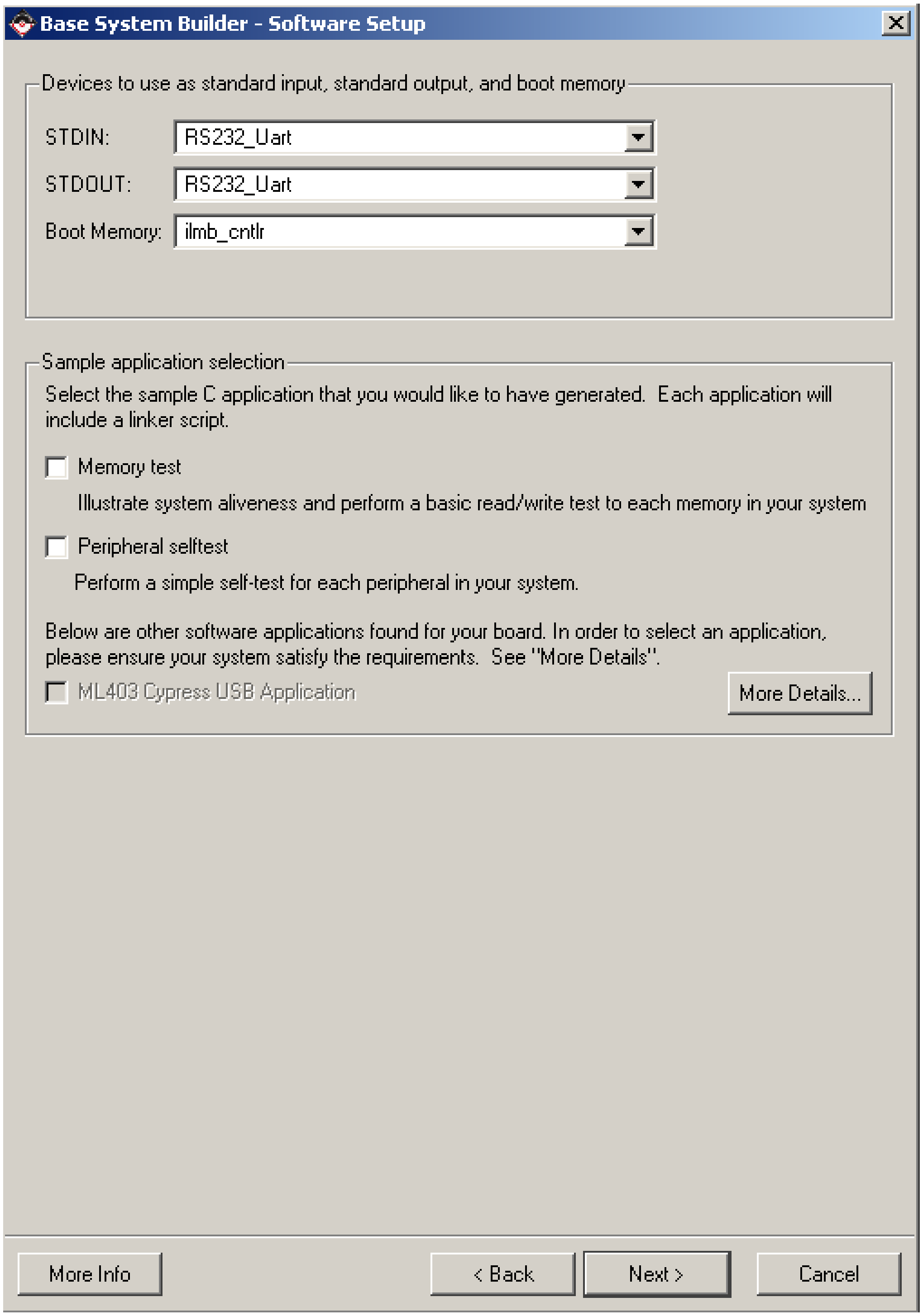
Peripherals

xps_bram_if_cntlr_1

Peripheral: XPS BRAM IF CNTLR

Memory size: 32 KB

In software setup, de-select the button for the two Application, shown below.
Press Next -> Generate -> Finish to finish



Base System Builder - Software Setup

Devices to use as standard input, standard output, and boot memory

STDIN: RS232_Uart

STDOUT: RS232_Uart

Boot Memory: ilmb_cntlr

Sample application selection

Select the sample C application that you would like to have generated. Each application will include a linker script.

□ Memory test

Illustrate system aliveness and perform a basic read/write test to each memory in your system

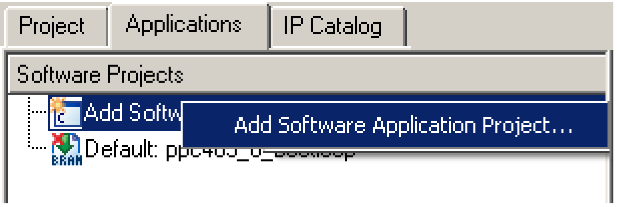
□ Peripheral selftest

Perform a simple self-test for each peripheral in your system.

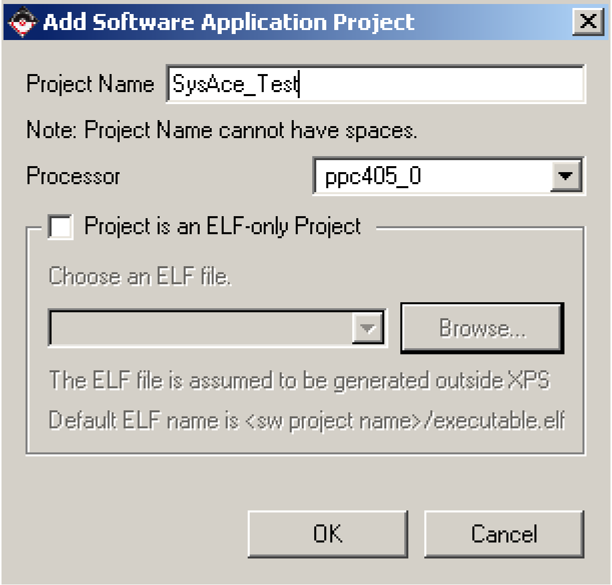
Below are other software applications found for your board. In order to select an application, please ensure your system satisfy the requirements. See "More Details".

□ ML403 Cypress USB Application

The next thing to do is to build the software application. This software application will be a simple application to show how to register the interrupt and to register the handler. To create a new application, Click on the Application tab on the left and right click on "Add Software Application Project" seen below:


Project
Applications
IP Catalog
Software Projects
Add Softw
Default: ppc405_0_bootloop
Add Software Application Project...

In the pop-up box, name your Application "SysAce_Test" as shown below. Select OK to continue:



Project Name SysAce_Test

Note: Project Name cannot have spaces.

Processor ppc405_0

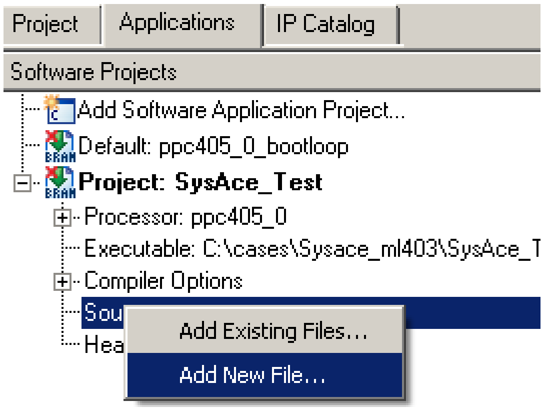
□ Project is an ELF-only Project

Choose an ELF file.

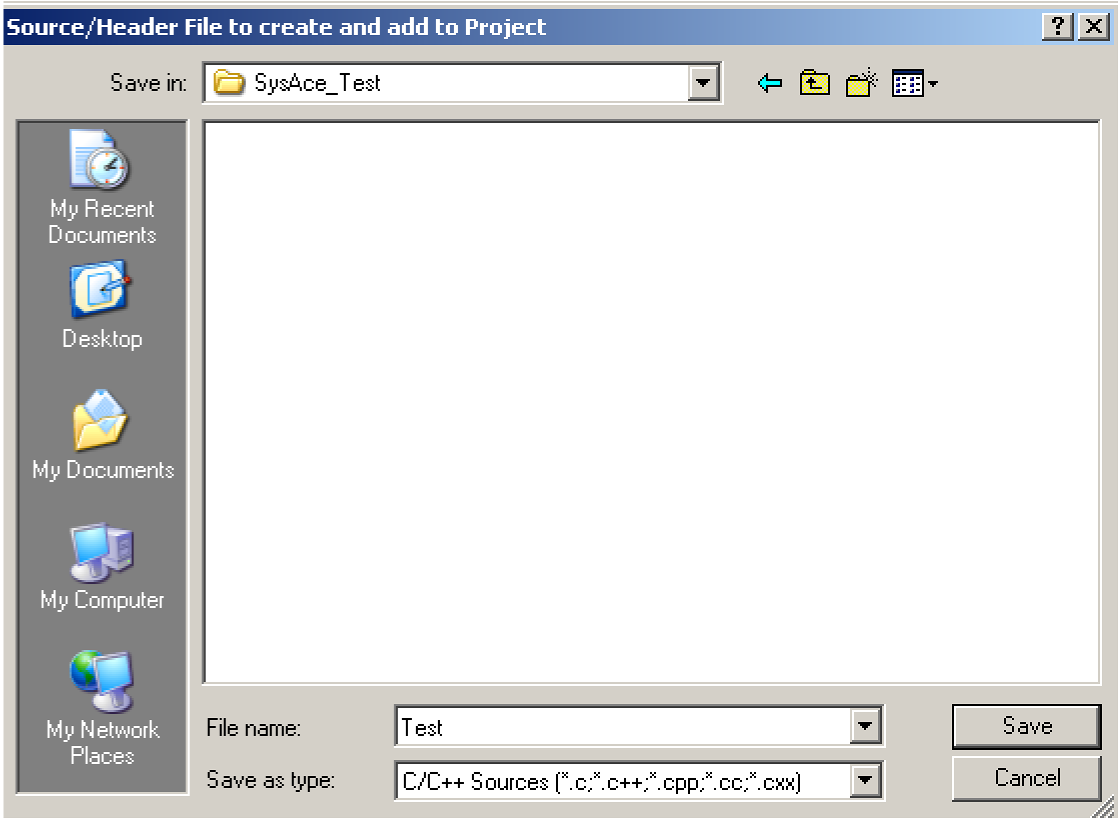
The ELF file is assumed to be generated outside XPS

Default ELF name is <sw project name>/executable.elf

This will create an empty project for you. To create the source code for the SysAce_Test application, right click on "source" in the application and select "Add New File…" This is seen below:



Browse to the SysAce_Test folder, if it isn't there create one making sure you name it correctly. Name the source file "Test.c", making sure you use a lower case c for the file. This is seen below:

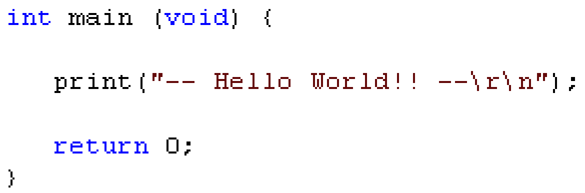

Source/Header File to create and add to Project

Save in: SysAce_Test

File name: Test

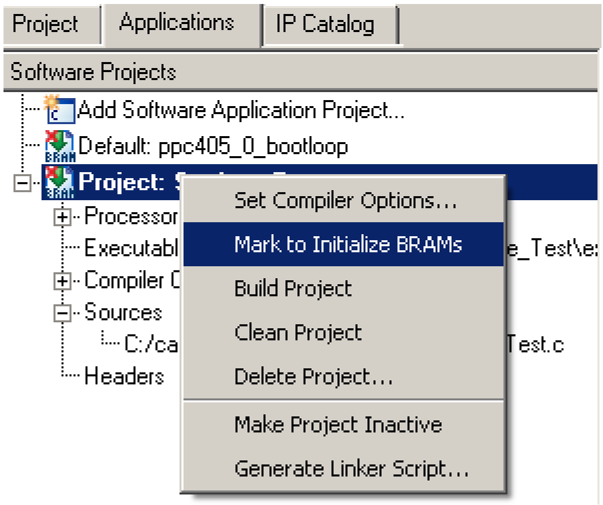
Save as type: C/C++ Sources (*.c;*.c++;*.cpp;*.cc;*.cxx)

Copy the “Hello World” code into your Test.c program.

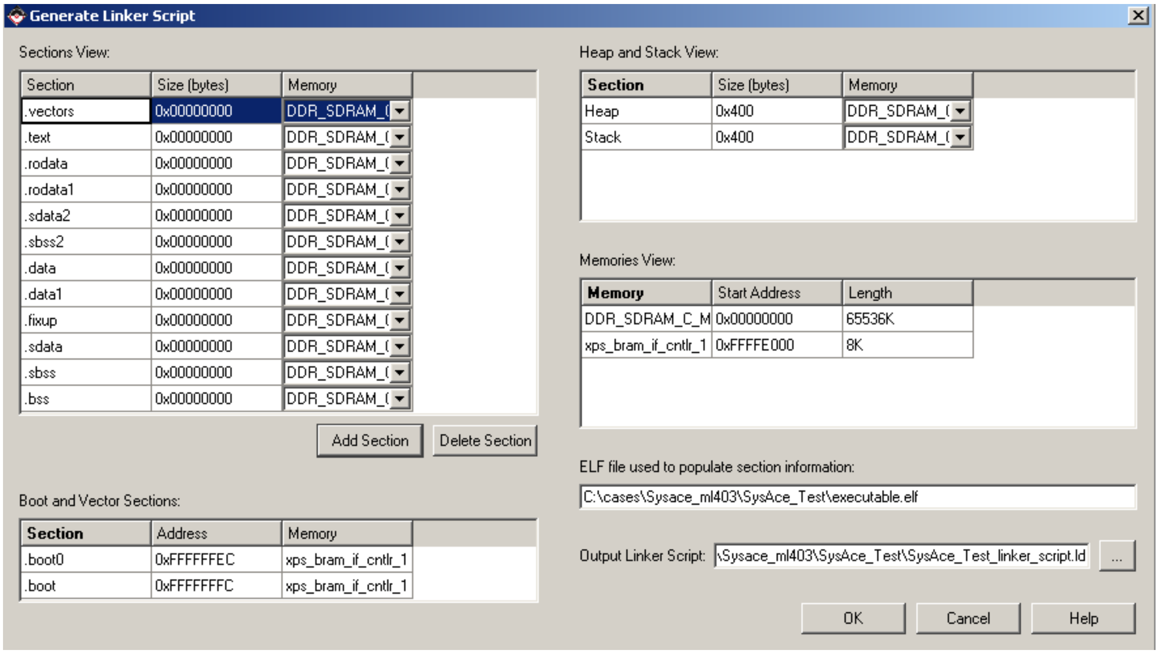


```
int main (void) {

   print("-- Hello World!! --\r\n");

   return 0;
}
```

Now that the code has been written, the project can be intialized into the BRAMS. To do this right click on the SysAce_Test application and select *“Mark to initialize brams…”* This is seen below:



Next Generate the Linker script for the SysAce_Test application. To do this right click on the SysAce_Test application and select *"Generate Linker Script"* Place all sections into the BRAM, this is seen below:



Sections View:

Section	Size (bytes)	Memory
.vectors	0x00000000	DDR_SDRAM_(
.text	0x00000000	DDR_SDRAM_(
.rodata	0x00000000	DDR_SDRAM_(
.rodata1	0x00000000	DDR_SDRAM_(
.sdata2	0x00000000	DDR_SDRAM_(
.sbss2	0x00000000	DDR_SDRAM_(
.data	0x00000000	DDR_SDRAM_(
.data1	0x00000000	DDR_SDRAM_(
.fixup	0x00000000	DDR_SDRAM_(
.sdata	0x00000000	DDR_SDRAM_(
.sbss	0x00000000	DDR_SDRAM_(
.bss	0x00000000	DDR_SDRAM_(

Boot and Vector Sections:

Section	Address	Memory
.boot0	0xFFFFFFEC	xps_bram_if_cntlr_1
.boot	0xFFFFFFFC	xps_bram_if_cntlr_1

Heap and Stack View:

Section	Size (bytes)	Memory
Heap	0x400	DDR_SDRAM_(
Stack	0x400	DDR_SDRAM_(

Memories View:

Memory	Start Address	Length
DDR_SDRAM_C_M	0x00000000	65536K
xps_bram_if_cntlr_1	0xFFFFE000	8K

ELF file used to populate section information:
C:\cases\Sysace_ml403\SysAce_Test\executable.elf

Output Linker Script: \Sysace_ml403\SysAce_Test\SysAce_Test_linker_script.ld

- The next step is to compile the Libraries and BSP's, to do this select Software -> Generate Libraries and BSP's.

- Then build the SysAce_Test application, to do this go to Software -> Build all user applications…

- Now build the bitstream, to do this go to Hardware -> generate bitstream.

- Now update the bitstream with the software, to do this go to Device Configuration -> Update bitstream.

- Finally, download the bitstream to the board, to do this go to Device Configuration -> Download bitstream.

- Open a Hyper-terminal and set the baud rate to 9600.

- Open the XMD debugger and download the elf file, this is done by typing the following command:
  dow SysAce_Test/executable.elf
  A Hello World message should be displayed on the Hyper-terminal.

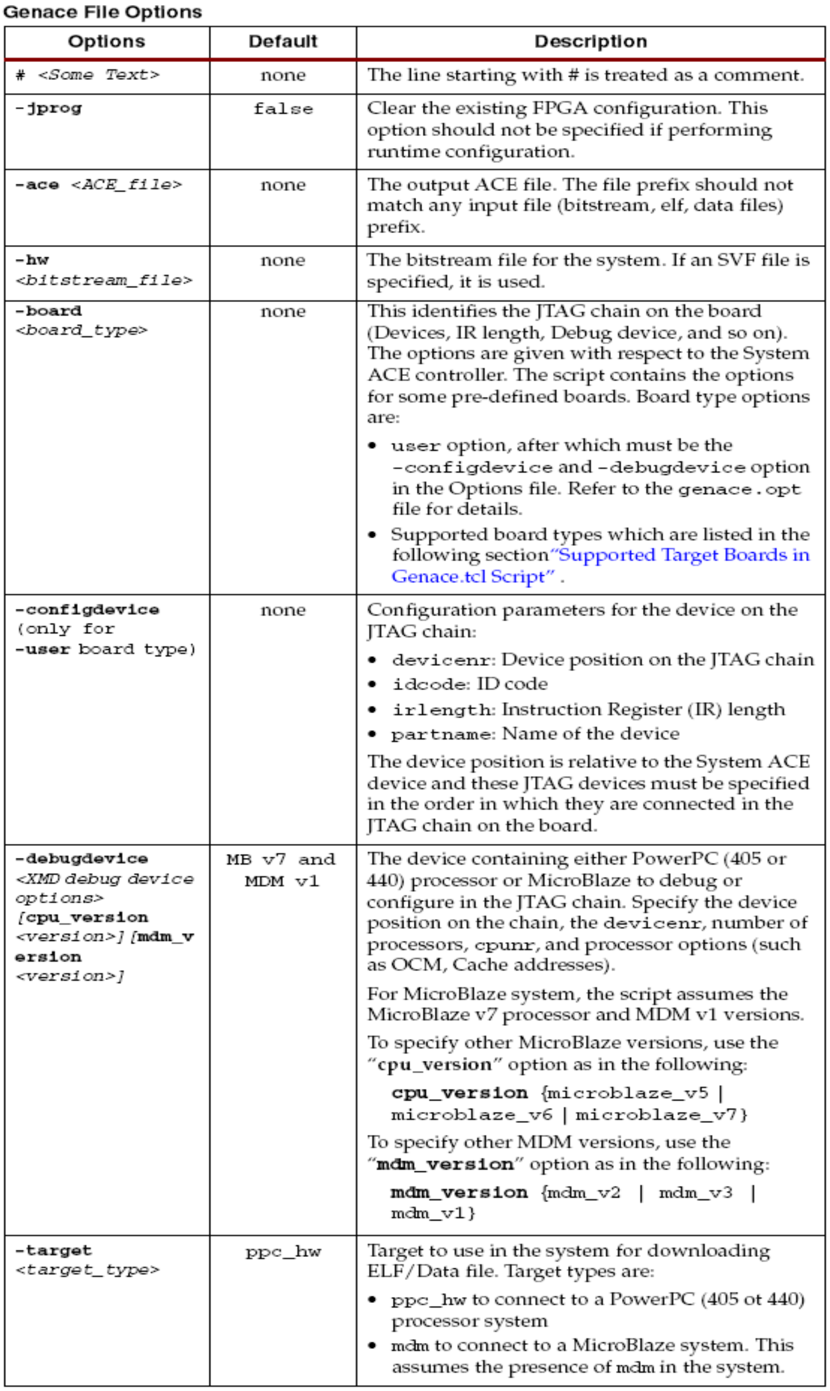
Formatting the Compact Flash Card

To format the CF card, download the freeware utility called “mkdosfs.zip” from the link below:
http://www1.mager.org/mkdosfs/mkdosfs.zip

Format the CF card by using the freeware utility called "mkdosfs" from the DOS prompt.
Inserted CF card into your PC using a card reader, lets call the drive “E”.
Extract the CF card utility “mkdosfs.zip” on root directory at “C” drive.
Issue a DOS command => *c:\> mkdosfs E*:
Next step is to copy the system.ace file onto the CompactFlash Card using the CompactFlash writer USB. Plug the CF writer into the USB port and copy and paste the system.ace file onto the CF card.

Generating System Ace File using the GenACE:

Now that the program has been verified in hardware the next step is to build the GenAce file for the CF card. However, first the GenAce options must be written. This can be passed to the genace.tcl in the form of a genace.opt file. The table of options is seen below:



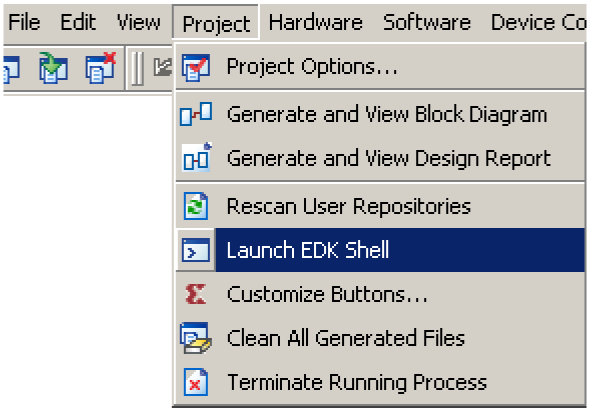
Genace File Options

Options	Default	Description
# <Some Text>	none	The line starting with # is treated as a comment.
-jprog	false	Clear the existing FPGA configuration. This option should not be specified if performing runtime configuration.
-ace <ACE_file>	none	The output ACE file. The file prefix should not match any input file (bitstream, elf, data files) prefix.
-hw <bitstream_file>	none	The bitstream file for the system. If an SVF file is specified, it is used.
-board <board_type>	none	This identifies the JTAG chain on the board (Devices, IR length, Debug device, and so on). The options are given with respect to the System ACE controller. The script contains the options for some pre-defined boards. Board type options are: • user option, after which must be the -configdevice and -debugdevice option in the Options file. Refer to the genace.opt file for details. • Supported board types which are listed in the following section"Supported Target Boards in Genace.tcl Script" .
-configdevice (only for -user board type)	none	Configuration parameters for the device on the JTAG chain: • devicenr: Device position on the JTAG chain • idcode: ID code • irlength: Instruction Register (IR) length • partname: Name of the device The device position is relative to the System ACE device and these JTAG devices must be specified in the order in which they are connected in the JTAG chain on the board.
-debugdevice <XMD debug device options> [cpu_version <version>] [mdm_version <version>]	MB v7 and MDM v1	The device containing either PowerPC (405 or 440) processor or MicroBlaze to debug or configure in the JTAG chain. Specify the device position on the chain, the devicenr, number of processors, cpunr, and processor options (such as OCM, Cache addresses). For MicroBlaze system, the script assumes the MicroBlaze v7 processor and MDM v1 versions. To specify other MicroBlaze versions, use the "cpu_version" option as in the following: cpu_version {microblaze_v5 | microblaze_v6 | microblaze_v7} To specify other MDM versions, use the "mdm_version" option as in the following: mdm_version {mdm_v2 | mdm_v3 | mdm_v1}
-target <target_type>	ppc_hw	Target to use in the system for downloading ELF/Data file. Target types are: • ppc_hw to connect to a PowerPC (405 ot 440) processor system • mdm to connect to a MicroBlaze system. This assumes the presence of mdm in the system.

Seen below is the genace.opt file to be passed to the genace.tcl. Create a file called genace.opt in your project directory and copy the options below into it.

```
-jprog
-hw implementation/download.bit
-ace system.ace
-board ml403
-target ppc_hw
-elf SysAce_Test/executable.elf
```

Next step is to open a open an EDK shell. To launch an EDK shell and go to Project -> Launch EDK Shell (seen below):



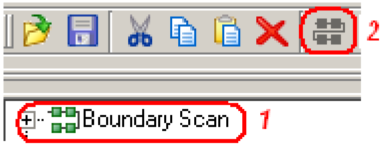
Run the command below in the shell:

```
xmd -tcl genace.tcl -opt genace.opt
```

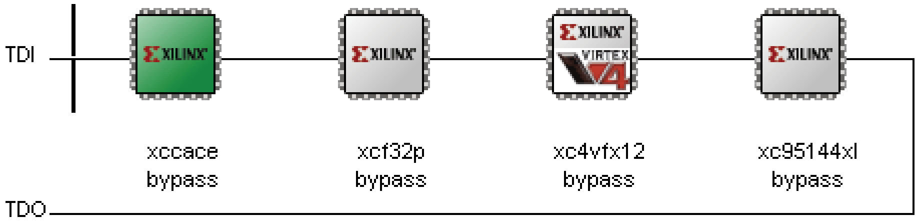
Generating System Ace File using through IMPACT:

The most common method to create a SysACE file is through IMPACT. Below, is the step to be taken to create the SysACE file in IMPACT. To launch the IMPACT tool with your board powered up.

The first thing to do is to make sure that the device chain is set up correctly. To view the chain in Impact, select “Boundary Scan” and the “Initialize chain” symbol, shown below:

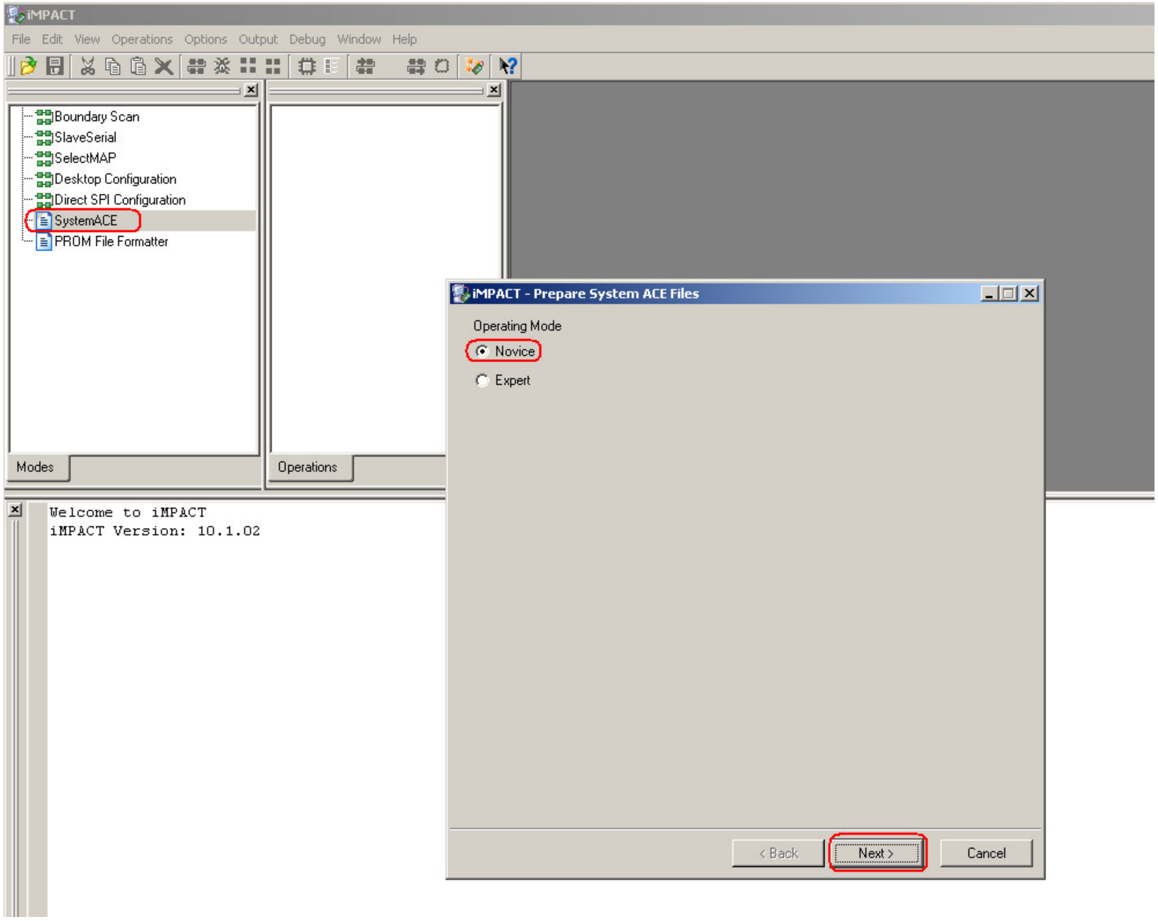


This will show you the chain seen below:



Setting up this chain will be discussed in the following steps.

Step1: Click SystemACE, Novice, Next to continue, shown below:



iMPACT - Prepare System ACE Files

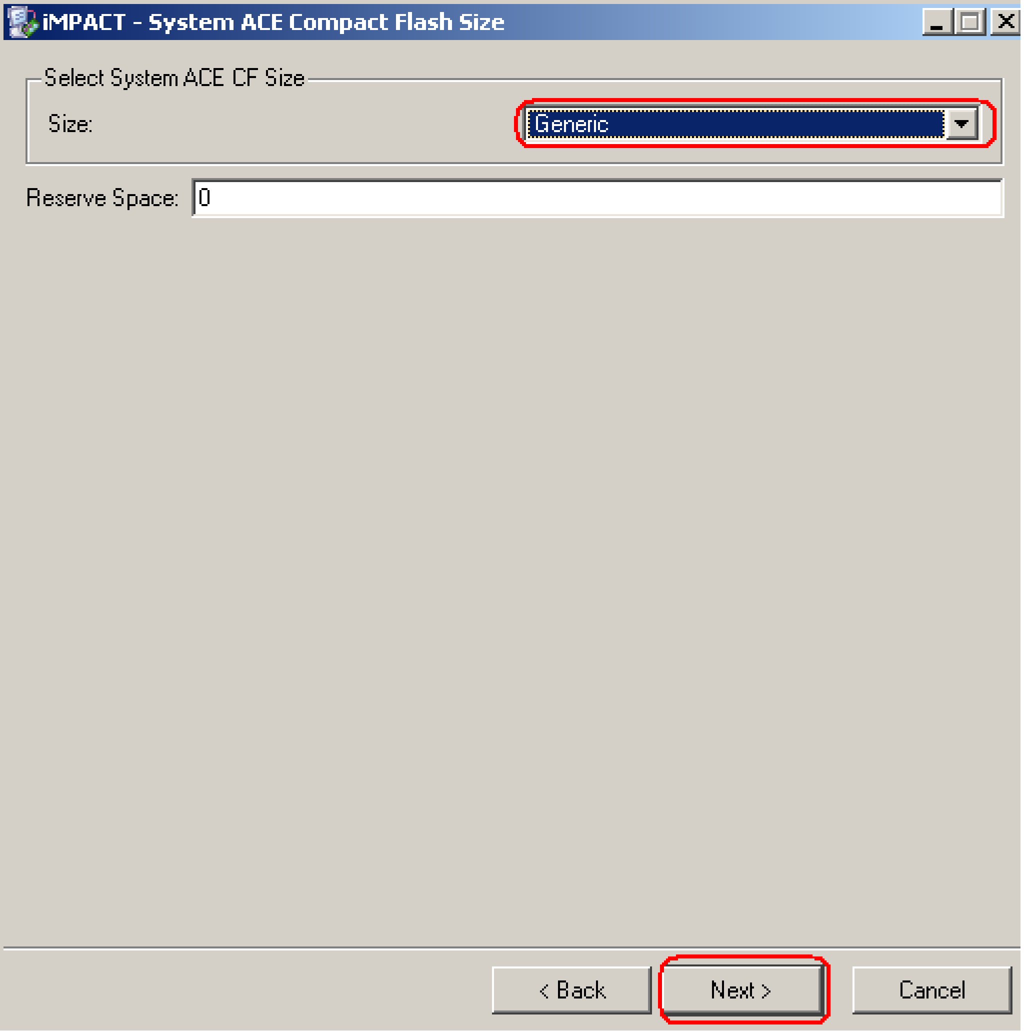
Operating Mode

● Novice

○ Expert

```
Welcome to iMPACT
iMPACT Version: 10.1.02
```

Step 2: Select size = Generic, Next to continue, shown below:



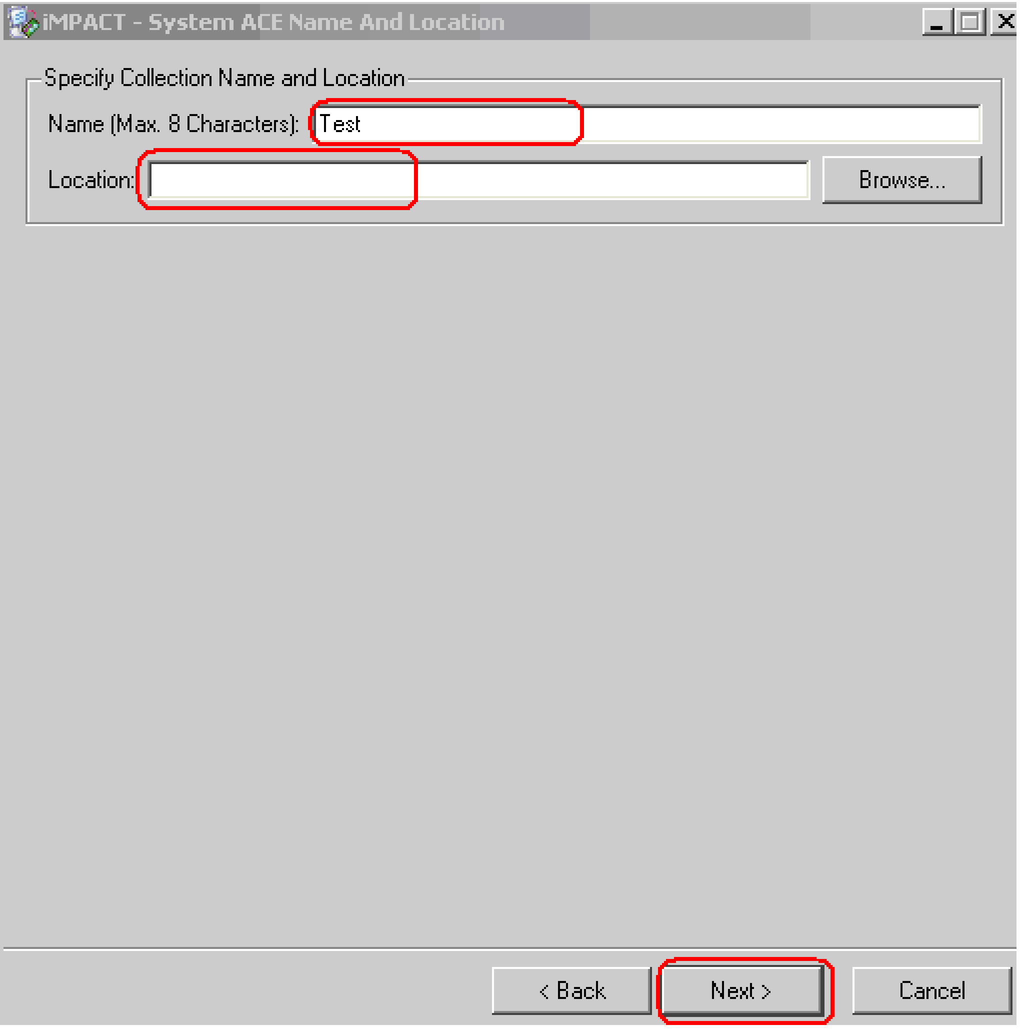
iMPACT - System ACE Compact Flash Size

Select System ACE CF Size

Size: Generic

Reserve Space: 0

Step 3: Give your files a name and location, Click next to continue, shown below:

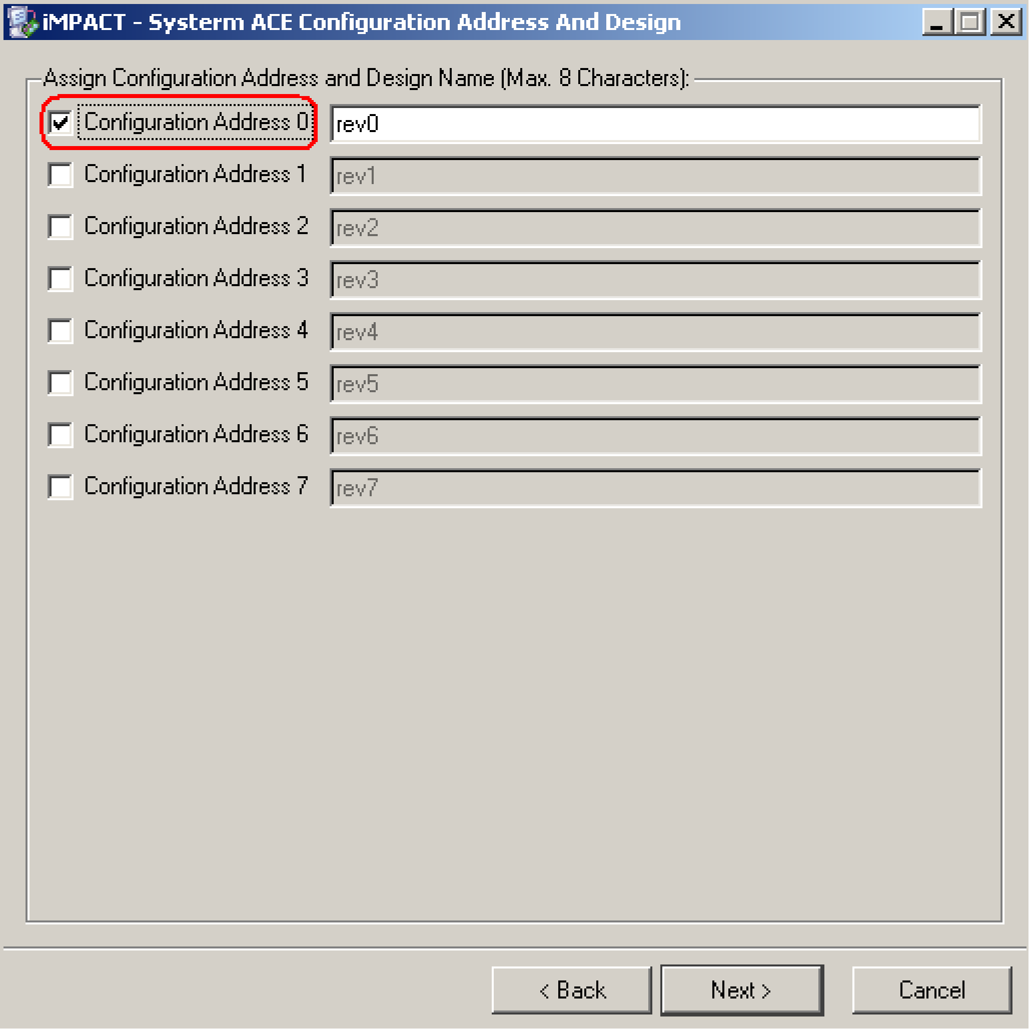


Specify Collection Name and Location

Name (Max. 8 Characters): Test

Location:

Step 4: Select configuration address as shown below. Click next to continue.



iMPACT - Systerm ACE Configuration Address And Design

Assign Configuration Address and Design Name (Max. 8 Characters):

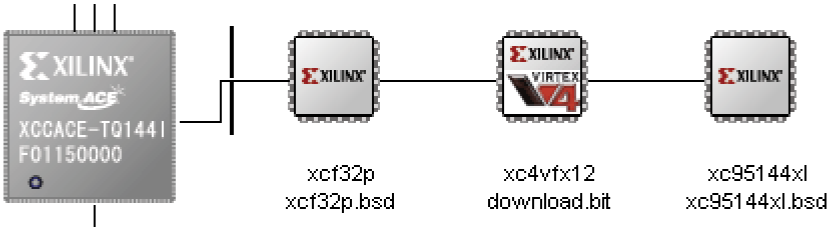
☑ Configuration Address 0 rev0
☐ Configuration Address 1 rev1
☐ Configuration Address 2 rev2
☐ Configuration Address 3 rev3
☐ Configuration Address 4 rev4
☐ Configuration Address 5 rev5
☐ Configuration Address 6 rev6
☐ Configuration Address 7 rev7

Step 5: Select Finish to finish.

Step 6: One page 18 the chain starts with a sysace => xcf32p => xc4fx12 => xc95144xl.The xcf32p will have to be added to the chain. This file can be found in the EDK install directory for example: C:\Xilinx\10.1\ISE\xcfp\data\xcf32p.bsd. You will be asked if you "Would like to add another design file" click yes and navigate to your download.bit file. Again, you will be asked to add another design file click yes and select the final device in the chain the xc95144xl this can be found in the EDK install directory C:\Xilinx\10.1\ISE\xc9500xl\data\ xc95144xl.bsd

Note: if the FPGA comes directly after the sysace as it does in the ML507, you do not have to do put the .bsd file in the chain.

You should have a device chain same as below. If not go back and start again as the chain needs to match or CF card wont configure the board.



Now that you CF files are generated you can either use windows to copy the CF files to your card or you can use impact.

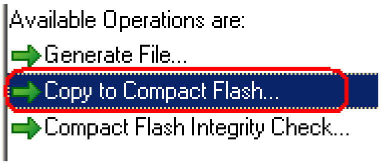
Copy to Compact Flash using windows

Step 1: Plug the CF card reader into your PC, copy and paste the following files:

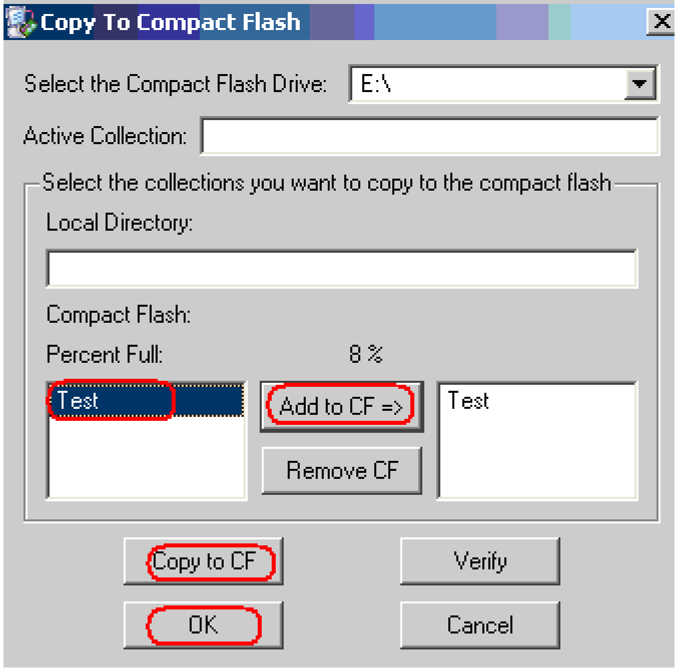
- *Test folder (assuming that this is what you called it in Step 3)*
- *xilinx.sys*

Copy to Compact Flash using impact

Step 1: Plug the CF card reader into your PC, and select "Copy to Compact Flash.." seen below:



Step 2: Navigate to the location where you saved the files (done in Step 3)
Select Test, Add to CF =>, Copy to CF, OK, Click OK to continue, these steps are seen below:



Select the Compact Flash Drive: E:\

Active Collection:

Select the collections you want to copy to the compact flash

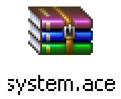
Local Directory:

Compact Flash:

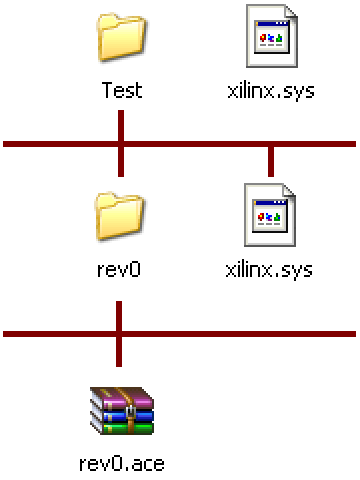
Percent Full: 8 %

Note: Make sure your board is configured correctly, this is seen in the user guide for your board.

If the file is generated through the GenACE flow this should create a system.ace file in your project directory (seen below):



If the file is generated through the IMPACT flow this should create the folder hierarchy seen below in the location directory mentioned in *Step 3*.



Testing the CF card

Open the Hyper-Terminal, baud rate = 9600.

Now simply power off the ML403 board and plug in the CF card into the CF card slot and power on the board. The "Hello World" message should be displayed on the hyper-terminal.