xios Hogeschool Limburg

Lab:

Implementing the Xilkernel on the Xilinx Spartan 3E Starter Kit

Keywords: EDK, Xilinx Platform Studio, Microblaze, Xilinx SPARTAN3E Starter Kit, Real-Time Operating System (RTOS), pthreads,…

xios Hogeschool Limburg

2009 Vincent Claes

Introduction

These labs are created by Vincent Claes. If you encounter problems using this labs or want some advice/consultancy on Embedded Systems, FPGA's, LabVIEW and especially LabVIEW FPGA you can always contact the author.

These labs are free to use however to show respect to the author please email him when you use them with your contact details (feedback is also welcome).

Contact Information:
Vincent Claes
claesvincent@gmail.com
http://www.linkedin.com/in/vincentclaes

Software Requirements:

- Xilinx EDK 10.1 SP3
- Xilinx EDK (SDK) 10.1 SP3

Hardware Requirements:

- Xilinx Spartan3E Starter kit: http://www.xilinx.com/products/devkits/HW-SPAR3E-SK-US-G.htm
- User manual: www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf

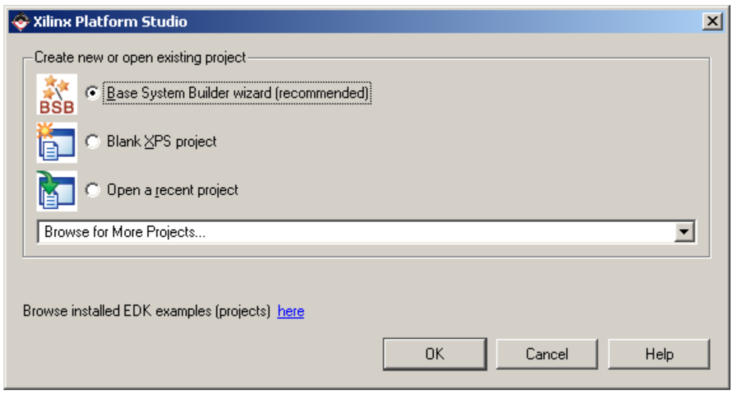
Getting Started

When you want to use this labs you have to setup your board. This labs are written for the Xilinx SPARTAN3E Starter Kit so it is quite interesting to read the user manual of the board. Be sure to plug in the USB cable, plug in the Power cord and Switch the board on before starting the lab.

Step 1: Create a Base System

The first step is creating a Base System for the Spartan3E starter board. For more information if you need please visit the Xilinx website.

Start Xilinx Platform Studio (EDK) 10.1
When the "Create new or open existing project" window appears select "Base System Builder wizard" and click the OK button.



Xilinx Platform Studio

Create new or open existing project

[✓ Base System Builder wizard (recommended)]

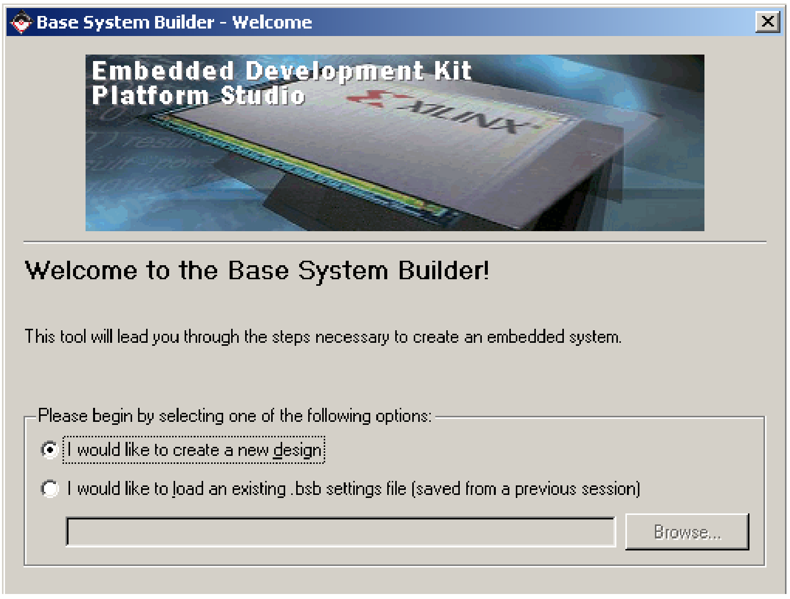
Blank XPS project

Open a recent project

Browse for More Projects...

Browse installed EDK examples (projects) here

Create a new *.xmp file in a directory on your HDD. Remember to not use spaces in your filenames or directory name.



Base System Builder - Welcome

Welcome to the Base System Builder!

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

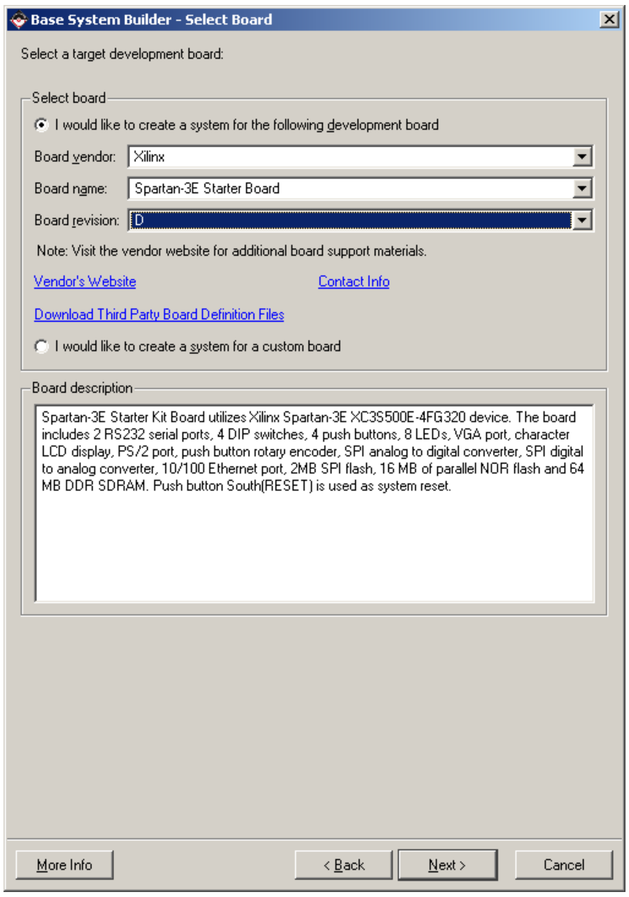
Please begin by selecting one of the following options:

[✓ I would like to create a new design]

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

Check the Option "I would like to create a new design" in the next window and click on the "Next" button.

This lab is written for the Spartan 3E Starter board Revision D so we have to select that board and click on the "Next" button.


Base System Builder - Select Board
Select a target development board:
Select board
Note: Visit the vendor website for additional board support materials.
Vendor's Website
Contact Info
Download Third Party Board Definition Files
Board description
More Info
< Back
Next >
Cancel

[✓ I would like to create a system for the following development board]

Board vendor: Xilinx

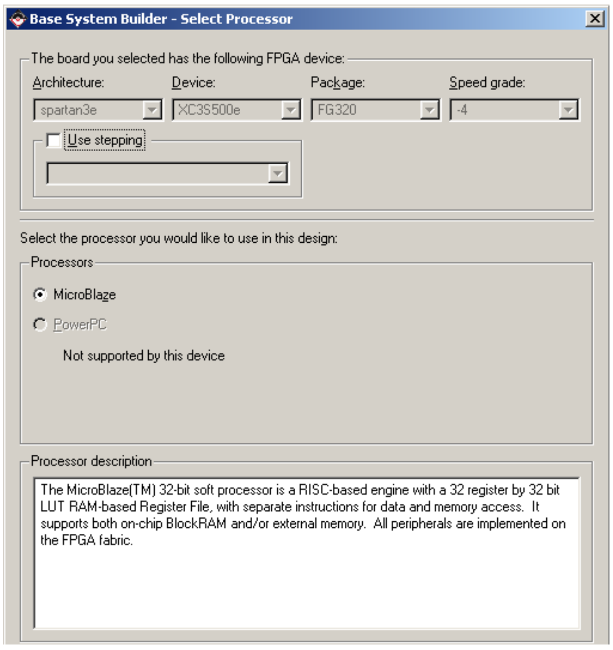
Board name: Spartan-3E Starter Board

Board revision: D

I would like to create a system for a custom board

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 64 MB DDR SDRAM. Push button South(RESET) is used as system reset.

On this Spartan 3E starter board there is no PowerPC core implemented in the FPGA so we can only add the Xilinx Microblaze processor as the processor that will be included in the embedded design. Select "Microblaze" and click the "Next" button.


Base System Builder - Select Processor

The board you selected has the following FPGA device:

Architecture:	Device:	Package:	Speed grade:
spartan3e	XC3S500e	FG320	-4

☐ Use stepping

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

Processors

● MicroBlaze

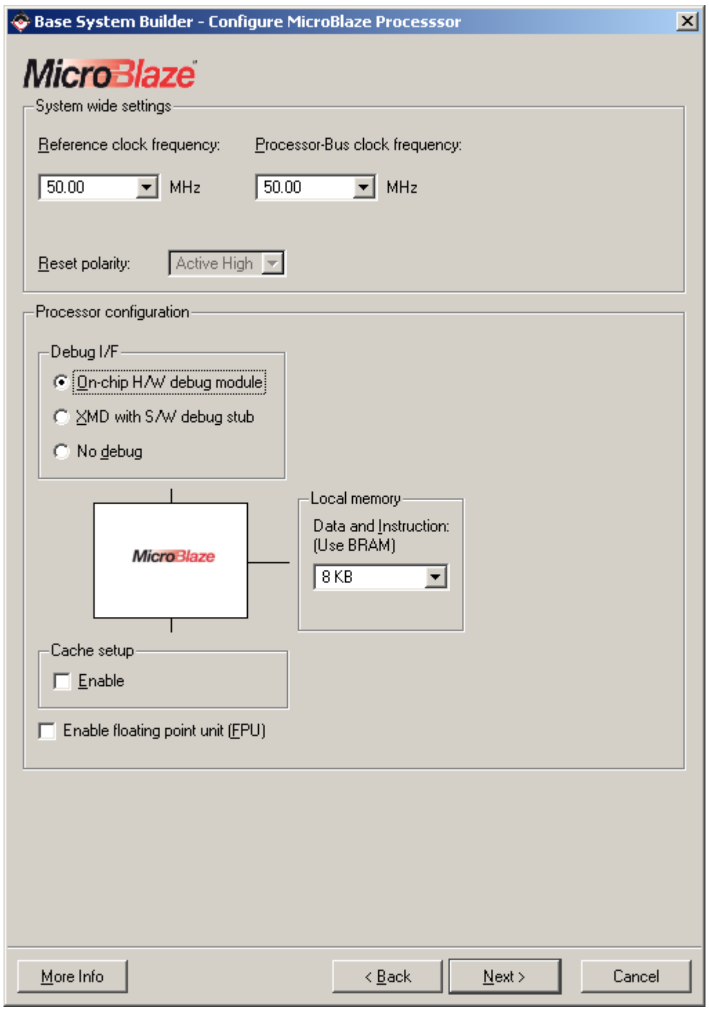
○ PowerPC

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.

Leave all the settings for the microblaze processor like they are proposed to you and click next.


Base System Builder - Configure MicroBlaze Processsor

MicroBlaze

System wide settings

Reference clock frequency: 50.00 MHz

Processor-Bus clock frequency: 50.00 MHz

Reset polarity: Active High

Processor configuration

Debug I/F

● On-chip H/W debug module

○ XMD with S/W debug stub

○ No debug

Local memory

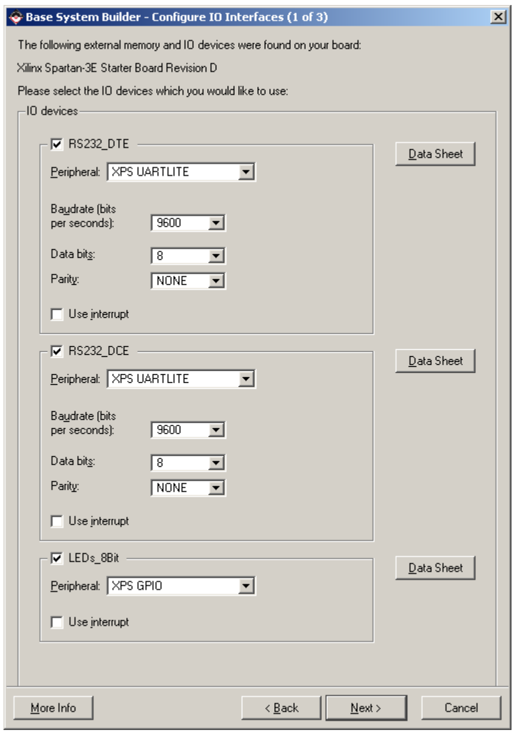
Data and Instruction: (Use BRAM) 8 KB

Cache setup

☐ Enable

☐ Enable floating point unit (FPU)

Now it is time to Configure all the IO Interfaces that are needed in this project. I will add all the existing IO interfaces to have a platform that can be used during more labs. So look at the next screens for the configuration of the IO Interfaces.


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

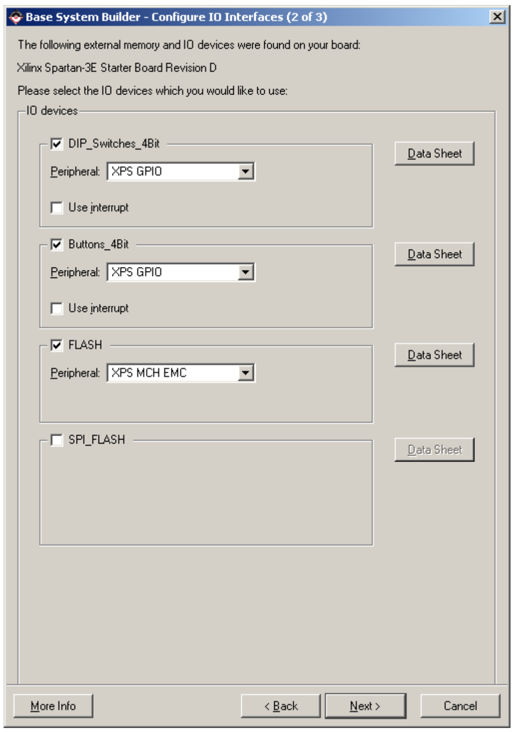
The following external memory and IO devices were found on your board:
Xilinx Spartan-3E Starter Board Revision D
Please select the IO devices which you would like to use:

IO devices

[✓] RS232_DTE
Peripheral: XPS UARTLITE
Baudrate (bits per seconds): 9600
Data bits: 8
Parity: NONE
[ ] Use interrupt

[✓] RS232_DCE
Peripheral: XPS UARTLITE
Baudrate (bits per seconds): 9600
Data bits: 8
Parity: NONE
[ ] Use interrupt

[✓] LEDs_8Bit
Peripheral: XPS GPIO
[ ] Use interrupt


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

The following external memory and IO devices were found on your board:
Xilinx Spartan-3E Starter Board Revision D
Please select the IO devices which you would like to use:

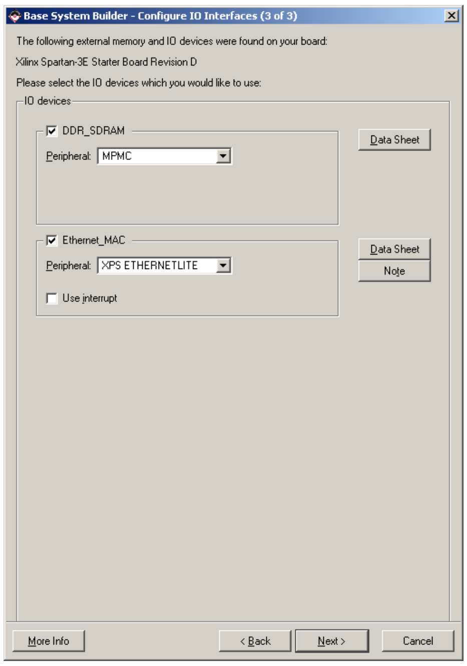
IO devices

[✓] DIP_Switches_4Bit
Peripheral: XPS GPIO
[ ] Use interrupt

[✓] Buttons_4Bit
Peripheral: XPS GPIO
[ ] Use interrupt

[✓] FLASH
Peripheral: XPS MCH EMC

[ ] SPI_FLASH


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

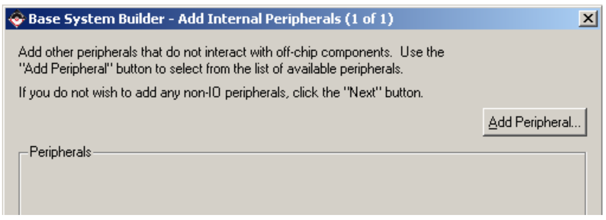
The following external memory and IO devices were found on your board:
Xilinx Spartan-3E Starter Board Revision D
Please select the IO devices which you would like to use:

IO devices

[✓] DDR_SDRAM
Peripheral: MPMC

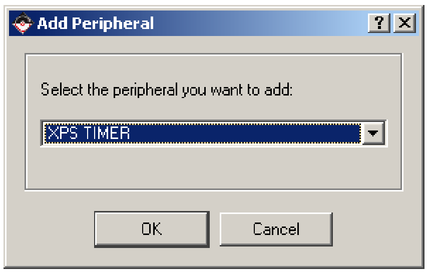
[✓] Ethernet_MAC
Peripheral: XPS ETHERNETLITE
[ ] Use interrupt

For the implementation of the Xilkernel we are in need of a timer. This is added as an "Internal Peripheral" during the next step of the "Base System Builder". So click the "Add Peripheral" button and then select the XPS_TIMER from the drop-down menu. Click the "OK" button.


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.

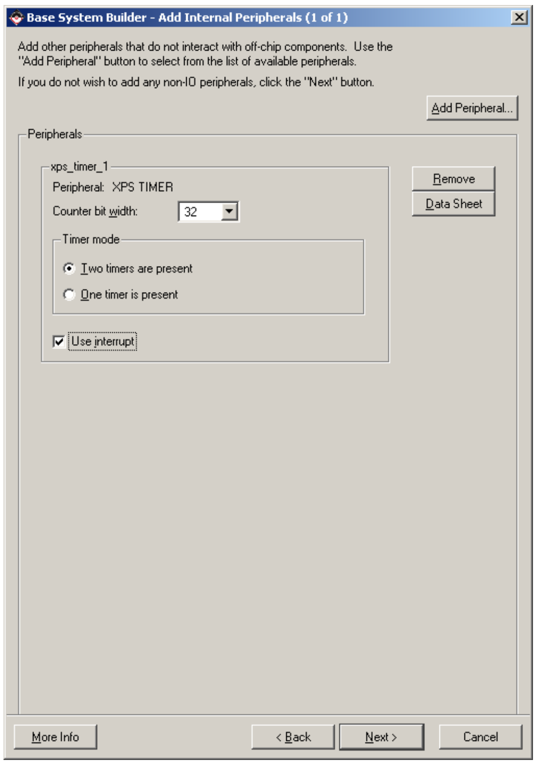
If you do not wish to add any non-IO peripherals, click the "Next" button.


Add Peripheral

Select the peripheral you want to add:

XPS TIMER

Make sure your XPS_TIMER has the same settings as below, so check the box "Use interrupt".


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_timer_1

Peripheral: XPS TIMER

Counter bit width: 32

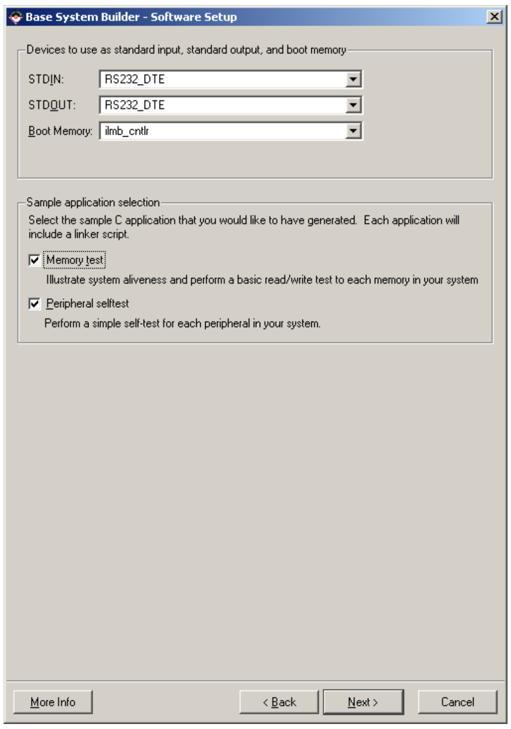
Timer mode

[✓ Two timers are present]

One timer is present

[✓ Use interrupt]

The next 3 steps are standard “Base System Builder” steps. Click the “Next button” 3 times. In this steps you tell which IO interfaces to use as STDIN (Standard input device) and STDOUT (Standard Output device). Also you set from which memory the system will boot (we will later a bootloop program in the blockram and execute our instruction from SDRAM).


Base System Builder - Software Setup

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

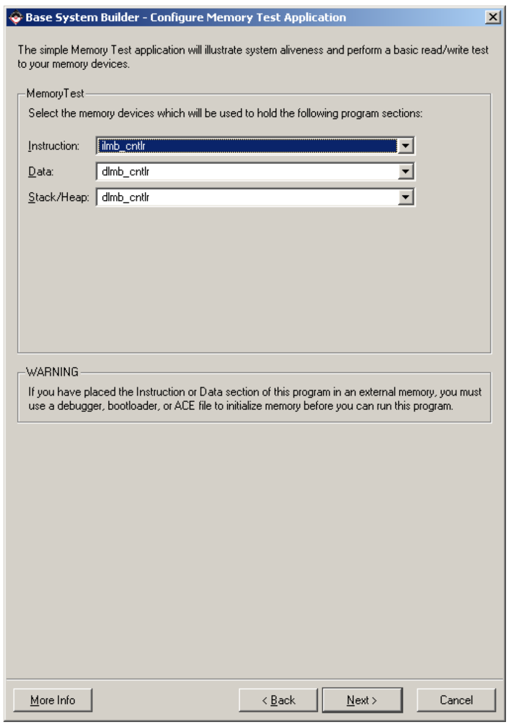
STDIN: RS232_DTE
STDOUT: RS232_DTE
Boot Memory: ilmb_cntlr

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.


Base System Builder - Configure Memory Test Application

The simple Memory Test application will illustrate system aliveness and perform a basic read/write test to your memory devices.

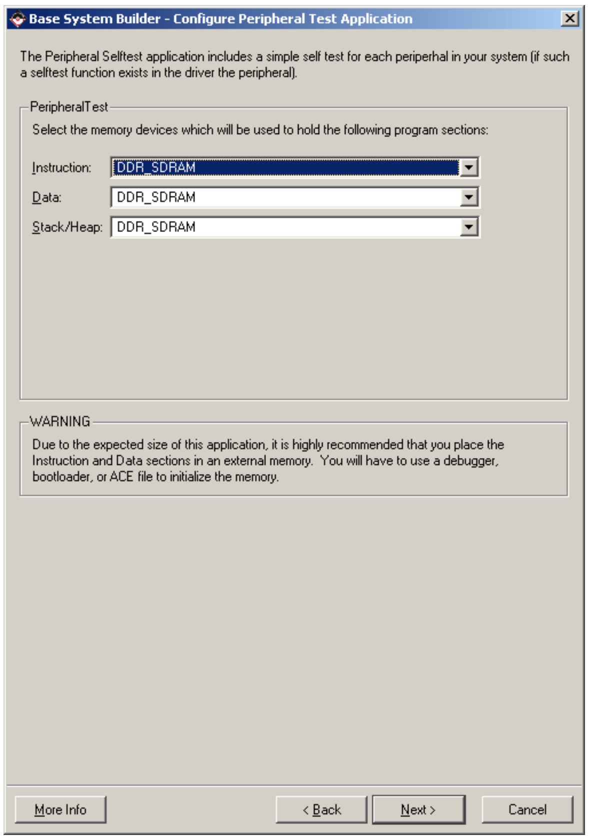
MemoryTest

Select the memory devices which will be used to hold the following program sections:

Instruction: ilmb_cntlr
Data: dlmb_cntlr
Stack/Heap: dlmb_cntlr

WARNING

If you have placed the Instruction or Data section of this program in an external memory, you must use a debugger, bootloader, or ACE file to initialize memory before you can run this program.



The Peripheral Selftest application includes a simple self test for each periperhal in your system (if such a selftest function exists in the driver the peripheral).

PeripheralTest

Select the memory devices which will be used to hold the following program sections:

Instruction: DDR_SDRAM

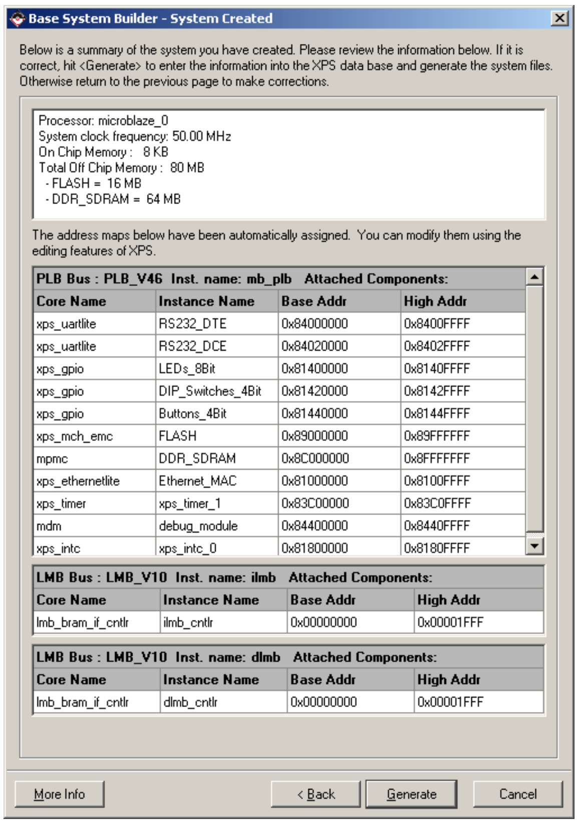
Data: DDR_SDRAM

Stack/Heap: DDR_SDRAM

WARNING

Due to the expected size of this application, it is highly recommended that you place the Instruction and Data sections in an external memory. You will have to use a debugger, bootloader, or ACE file to initialize the memory.

The next window is a summary of the system you have created. You can see all the cores you have implemented in your SoC (System-On-Chip). You can also have a look at the memory map of your system. Click the "Generate" button to have your system generated. After this a new screen appears; click the "Finish" button.



Below is a summary of the system you have created. Please review the information below. If it is correct, hit <Generate> to enter the information into the XPS data base and generate the system files. Otherwise return to the previous page to make corrections.

```
Processor: microblaze_0
System clock frequency: 50.00 MHz
On Chip Memory :   8 KB
Total Off Chip Memory :  80 MB
  - FLASH =  16 MB
  - DDR_SDRAM =  64 MB
```

The address maps below have been automatically assigned. You can modify them using the editing features of XPS.

PLB Bus : PLB_V46 Inst. name: mb_plb Attached Components:

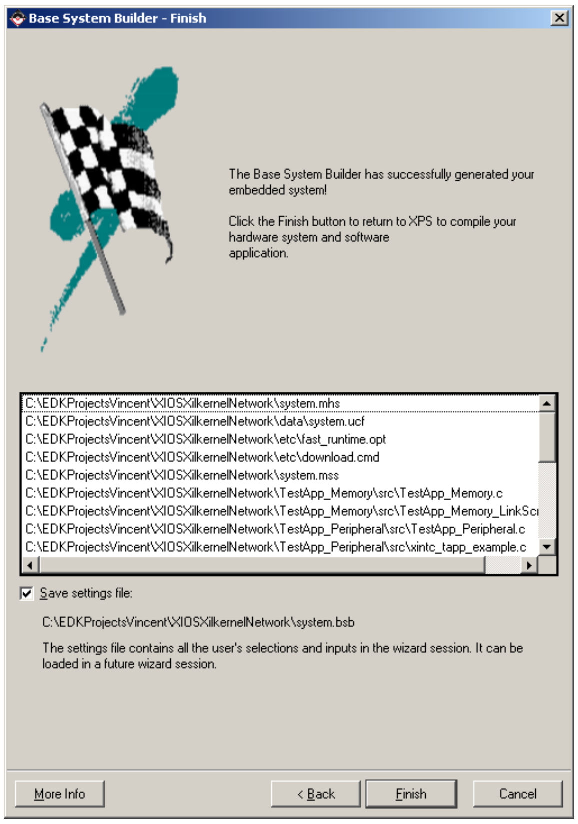
Core Name	Instance Name	Base Addr	High Addr
xps_uartlite	RS232_DTE	0x84000000	0x8400FFFF
xps_uartlite	RS232_DCE	0x84020000	0x8402FFFF
xps_gpio	LEDs_8Bit	0x81400000	0x8140FFFF
xps_gpio	DIP_Switches_4Bit	0x81420000	0x8142FFFF
xps_gpio	Buttons_4Bit	0x81440000	0x8144FFFF
xps_mch_emc	FLASH	0x89000000	0x89FFFFFF
mpmc	DDR_SDRAM	0x8C000000	0x8FFFFFFF
xps_ethernetlite	Ethernet_MAC	0x81000000	0x8100FFFF
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 Components:

Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	ilmb_cntlr	0x00000000	0x00001FFF

LMB Bus : LMB_V10 Inst. name: dlmb Attached Components:

Core Name	Instance Name	Base Addr	High Addr
lmb_bram_if_cntlr	dlmb_cntlr	0x00000000	0x00001FFF



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.

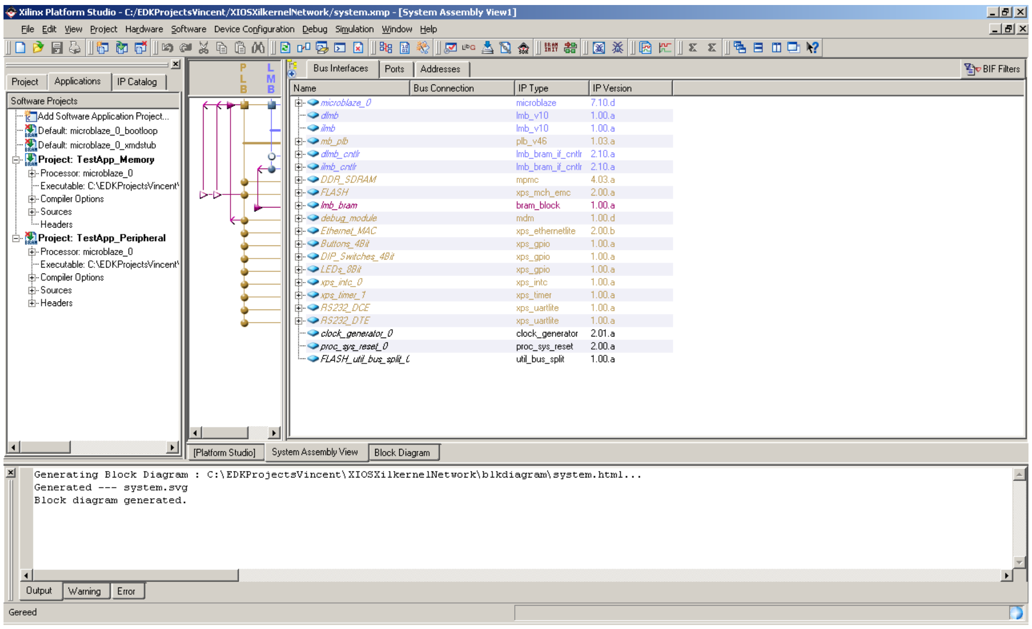
```
C:\EDKProjectsVincent\XIOSXilkernelNetwork\system.mhs
C:\EDKProjectsVincent\XIOSXilkernelNetwork\data\system.ucf
C:\EDKProjectsVincent\XIOSXilkernelNetwork\etc\fast_runtime.opt
C:\EDKProjectsVincent\XIOSXilkernelNetwork\etc\download.cmd
C:\EDKProjectsVincent\XIOSXilkernelNetwork\system.mss
C:\EDKProjectsVincent\XIOSXilkernelNetwork\TestApp_Memory\src\TestApp_Memory.c
C:\EDKProjectsVincent\XIOSXilkernelNetwork\TestApp_Memory\src\TestApp_Memory_LinkScr
C:\EDKProjectsVincent\XIOSXilkernelNetwork\TestApp_Peripheral\src\TestApp_Peripheral.c
C:\EDKProjectsVincent\XIOSXilkernelNetwork\TestApp_Peripheral\src\xintc_tapp_example.c
```

[✓] Save settings file:

C:\EDKProjectsVincent\XIOSXilkernelNetwork\system.bsb

The settings file contains all the user's selections and inputs in the wizard session. It can be loaded in a future wizard session.

Congratulations, you just have designed an Xilinx Base system that can run an Real-Time Operating System (RTOS) called Xilkernel. You can have a look at your block diagram in EDK.

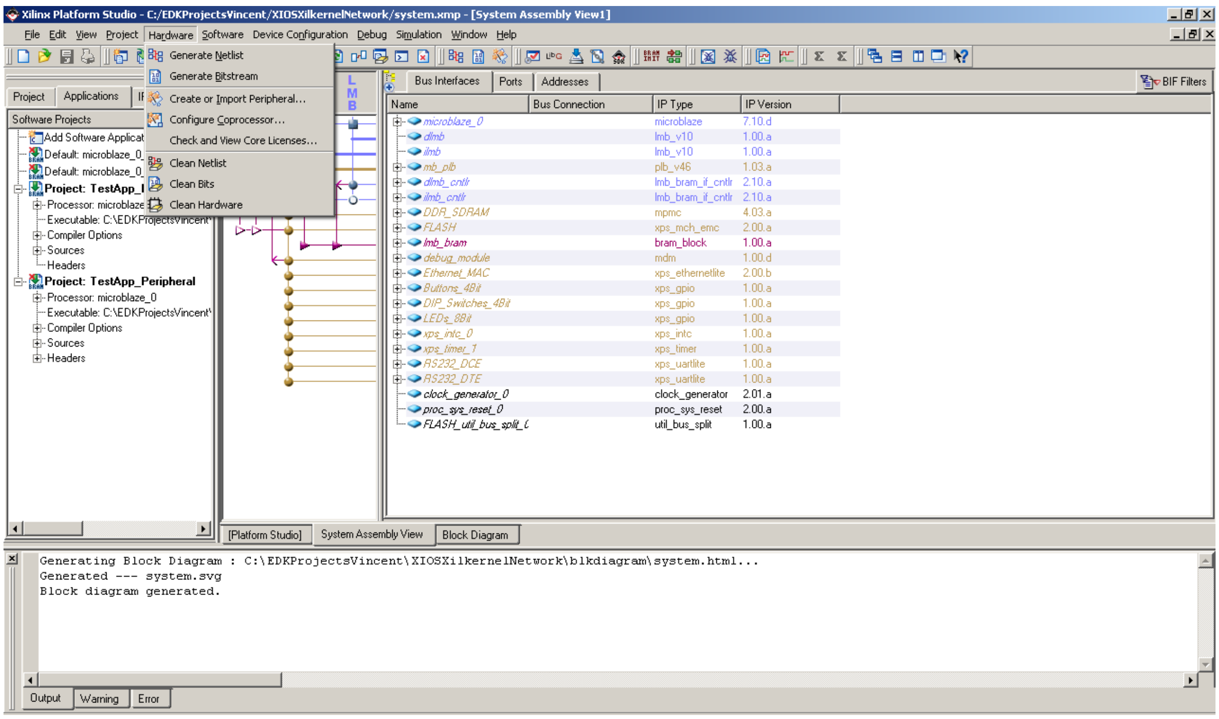

Xilinx Platform Studio - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp - [System Assembly View1]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Project Applications IP Catalog
Software Projects
Add Software Application Project...
Default: microblaze_0_bootloop
Default: microblaze_0_xmdstub
Project: TestApp_Memory
Processor: microblaze_0
Executable: C:\EDKProjectsVincent\
Compiler Options
Sources
Headers
Project: TestApp_Peripheral
Processor: microblaze_0
Executable: C:\EDKProjectsVincent\
Compiler Options
Sources
Headers
Bus Interfaces Ports Addresses
BIF Filters
[Platform Studio] System Assembly View Block Diagram
Output Warning Error
Gereed

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.10.d
dlmb		lmb_v10	1.00.a
ilmb		lmb_v10	1.00.a
mb_plb		plb_v46	1.03.a
dlmb_cntlr		lmb_bram_if_cntlr	2.10.a
ilmb_cntlr		lmb_bram_if_cntlr	2.10.a
DDR_SDRAM		mpmc	4.03.a
FLASH		xps_mch_emc	2.00.a
lmb_bram		bram_block	1.00.a
debug_module		mdm	1.00.d
Ethernet_MAC		xps_ethernetlite	2.00.b
Buttons_4Bit		xps_gpio	1.00.a
DIP_Switches_4Bit		xps_gpio	1.00.a
LEDs_8Bit		xps_gpio	1.00.a
xps_intc_0		xps_intc	1.00.a
xps_timer_1		xps_timer	1.00.a
RS232_DCE		xps_uartlite	1.00.a
RS232_DTE		xps_uartlite	1.00.a
clock_generator_0		clock_generator	2.01.a
proc_sys_reset_0		proc_sys_reset	2.00.a
FLASH_util_bus_split_(		util_bus_split	1.00.a

```
Generating Block Diagram : C:\EDKProjectsVincent\XIOSXilkernelNetwork\blkdiagram\system.html...
Generated --- system.svg
Block diagram generated.
```

Step 2: Software Platform Settings

Now that you have created your hardware project you only have to generate the bitstream to be able to download later this hardware project. In EDK click the "Hardware" menu and then select "Generate Bitstream".


Xilinx Platform Studio - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp - [System Assembly View1]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Generate Netlist
Generate Bitstream
Create or Import Peripheral...
Configure Coprocessor...
Check and View Core Licenses...
Clean Netlist
Clean Bits
Clean Hardware
Project Applications
Software Projects
Add Software Applicat
Default: microblaze_0_
Default: microblaze_0_
Project: TestApp_I
Processor: microblaze
Executable: C:\EDKProjectsVincent\
Compiler Options
Sources
Headers
Project: TestApp_Peripheral
Processor: microblaze_0
Executable: C:\EDKProjectsVincent\
Compiler Options
Sources
Headers
Bus Interfaces Ports Addresses
BIF Filters
[Platform Studio] System Assembly View Block Diagram
Output Warning Error

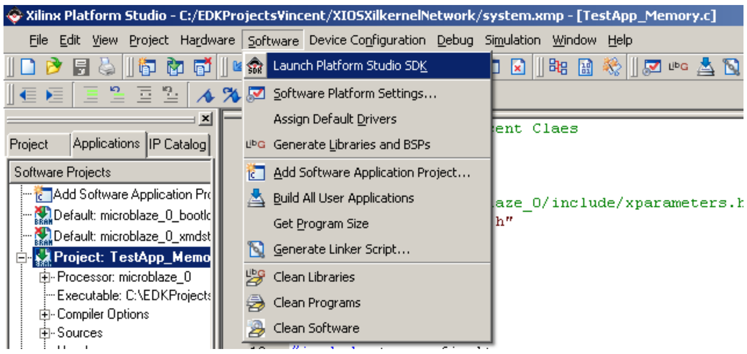
Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.10.d
dlmb		lmb_v10	1.00.a
ilmb		lmb_v10	1.00.a
mb_plb		plb_v46	1.03.a
dlmb_cntlr		lmb_bram_if_cntlr	2.10.a
ilmb_cntlr		lmb_bram_if_cntlr	2.10.a
DDR_SDRAM		mpmc	4.03.a
FLASH		xps_mch_emc	2.00.a
lmb_bram		bram_block	1.00.a
debug_module		mdm	1.00.d
Ethernet_MAC		xps_ethernetlite	2.00.b
Buttons_4Bit		xps_gpio	1.00.a
DIP_Switches_4Bit		xps_gpio	1.00.a
LEDs_8Bit		xps_gpio	1.00.a
xps_intc_0		xps_intc	1.00.a
xps_timer_1		xps_timer	1.00.a
RS232_DCE		xps_uartlite	1.00.a
RS232_DTE		xps_uartlite	1.00.a
clock_generator_0		clock_generator	2.01.a
proc_sys_reset_0		proc_sys_reset	2.00.a
FLASH_util_bus_split_(		util_bus_split	1.00.a

```
Generating Block Diagram : C:\EDKProjectsVincent\XIOSXilkernelNetwork\blkdiagram\system.html...
Generated --- system.svg
Block diagram generated.
```

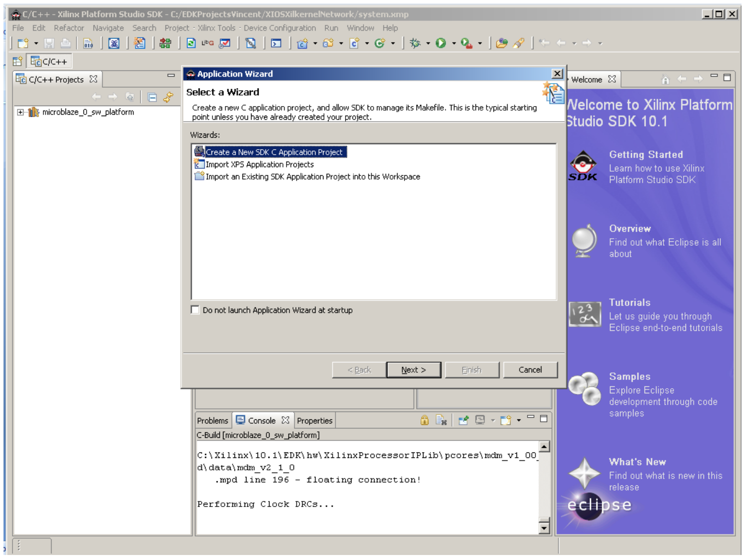
Generating the bitstream will take some time so I suggest you go to the coffee machine and have a cup of coffee and return after some time.

When you come back and see in the "Output" window of EDK the following message:
"Saving bit stream in "system.bit"
Bitstream generation is complete.
Done!

You are lucky and can go further in the development. Select the "Software" → "Launch Plaform Studio SDK".


Xilinx Platform Studio - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp - [TestApp_Memory.c]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
Launch Platform Studio SDK
Software Platform Settings...
Assign Default Drivers
Generate Libraries and BSPs
Add Software Application Project...
Build All User Applications
Get Program Size
Generate Linker Script...
Clean Libraries
Clean Programs
Clean Software
Project
Applications
IP Catalog
Software Projects
Add Software Application Pr
Default: microblaze_0_bootlo
Default: microblaze_0_xmdst
Project: TestApp_Memo
Processor: microblaze_0
Executable: C:\EDKProjects
Compiler Options
Sources
ent Claes
aze_0/include/xparameters.h
h"

When you have started Xilinx Platform Studio SDK a wizard is launched. The first step is selecting the "Create a New SDK C Application Project" and click the "Next" button.

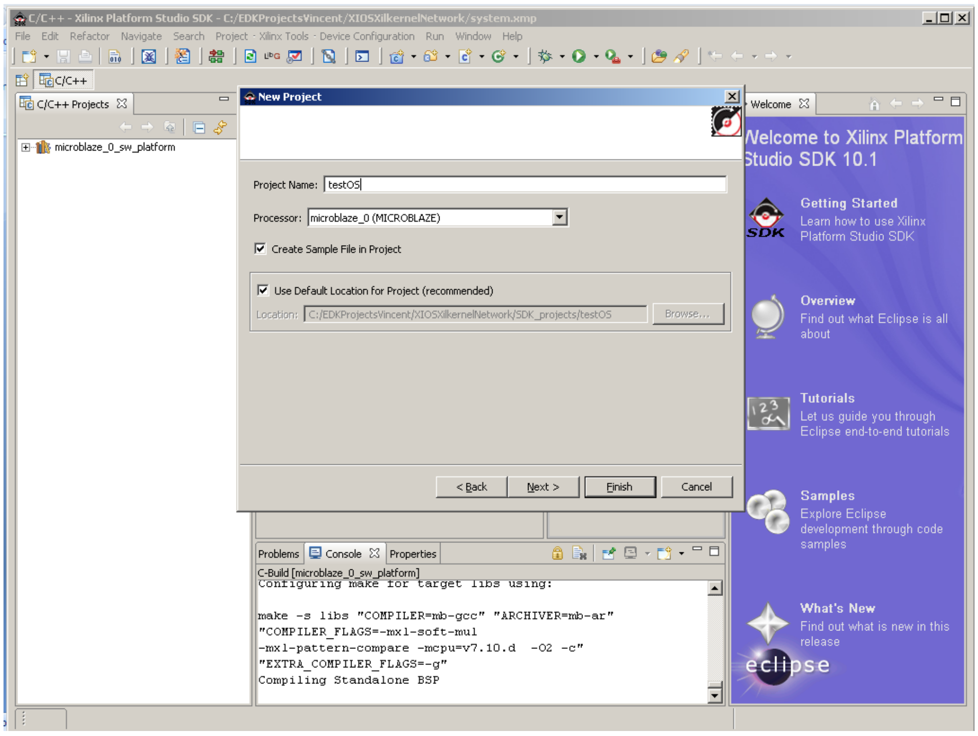

C/C++ - Xilinx Platform Studio SDK - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp
File Edit Refactor Navigate Search Project Xilinx Tools Device Configuration Run Window Help
C/C++
C/C++ Projects
microblaze_0_sw_platform
Application Wizard
Select a Wizard
Wizards:
Create a New SDK C Application Project
Import XPS Application Projects
Import an Existing SDK Application Project into this Workspace
Do not launch Application Wizard at startup
< Back
Next >
Finish
Cancel
Problems
Console
Properties
C-Build [microblaze_0_sw_platform]
Welcome
Welcome to Xilinx Platform Studio SDK 10.1
Getting Started
Learn how to use Xilinx Platform Studio SDK
Overview
Find out what Eclipse is all about
Tutorials
Let us guide you through Eclipse end-to-end tutorials
Samples
Explore Eclipse development through code samples
What's New
Find out what is new in this release
eclipse

Create a new C application project, and allow SDK to manage its Makefile. This is the typical starting point unless you have already created your project.

[ ] Do not launch Application Wizard at startup

```
C:\Xilinx\10.1\EDK\hw\XilinxProcessorIPLib\pcores\mdm_v1_00_
d\data\mdm_v2_1_0
   .mpd line 196 - floating connection!

Performing Clock DRCs...
```

Put in the "Project Name" box a name that you wish to give to your project. Be sure you select the correct microblaze instance for your project, normally there is only one if you are working in a design with only one processor core. Leave all the other options like they are presented to you and click on the "Next" button.


C/C++ - Xilinx Platform Studio SDK - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp
File Edit Refactor Navigate Search Project Xilinx Tools Device Configuration Run Window Help
C/C++
C/C++ Projects
microblaze_0_sw_platform
Welcome
Welcome to Xilinx Platform Studio SDK 10.1
Getting Started
Learn how to use Xilinx Platform Studio SDK
Overview
Find out what Eclipse is all about
Tutorials
Let us guide you through Eclipse end-to-end tutorials
Samples
Explore Eclipse development through code samples
What's New
Find out what is new in this release
eclipse
Problems
Console
Properties
&lt; Back
Next &gt;
Finish
Cancel
Browse...

New Project

Project Name: testOS

Processor: microblaze_0 (MICROBLAZE)

☑ Create Sample File in Project

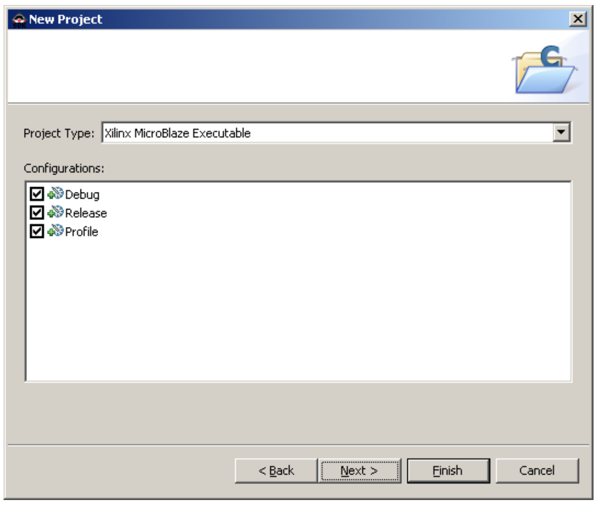
☑ Use Default Location for Project (recommended)

Location: C:/EDKProjectsVincent/XIOSXilkernelNetwork/SDK_projects/testOS

C-Build [microblaze_0_sw_platform]

```
Configuring make for target libs using:

make -s libs "COMPILER=mb-gcc" "ARCHIVER=mb-ar"
"COMPILER_FLAGS=-mxl-soft-mul
-mxl-pattern-compare -mcpu=v7.10.d  -O2 -c"
"EXTRA_COMPILER_FLAGS=-g"
Compiling Standalone BSP
```

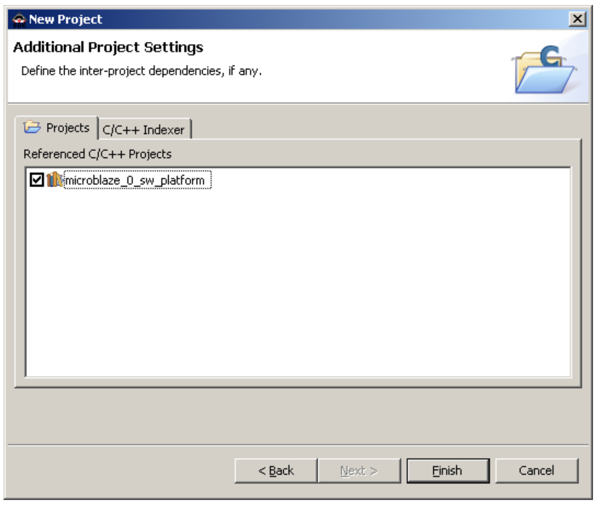
Finish this "New project wizard" by clicking the "Next" button on the next screen that appears and then checking the box for "microblaze_0_sw_platform" to use this as a referenced C/C++ project. Now press the "Finish" button.



Project Type: Xilinx MicroBlaze Executable

Configurations:

- [x] Debug
- [x] Release
- [x] Profile



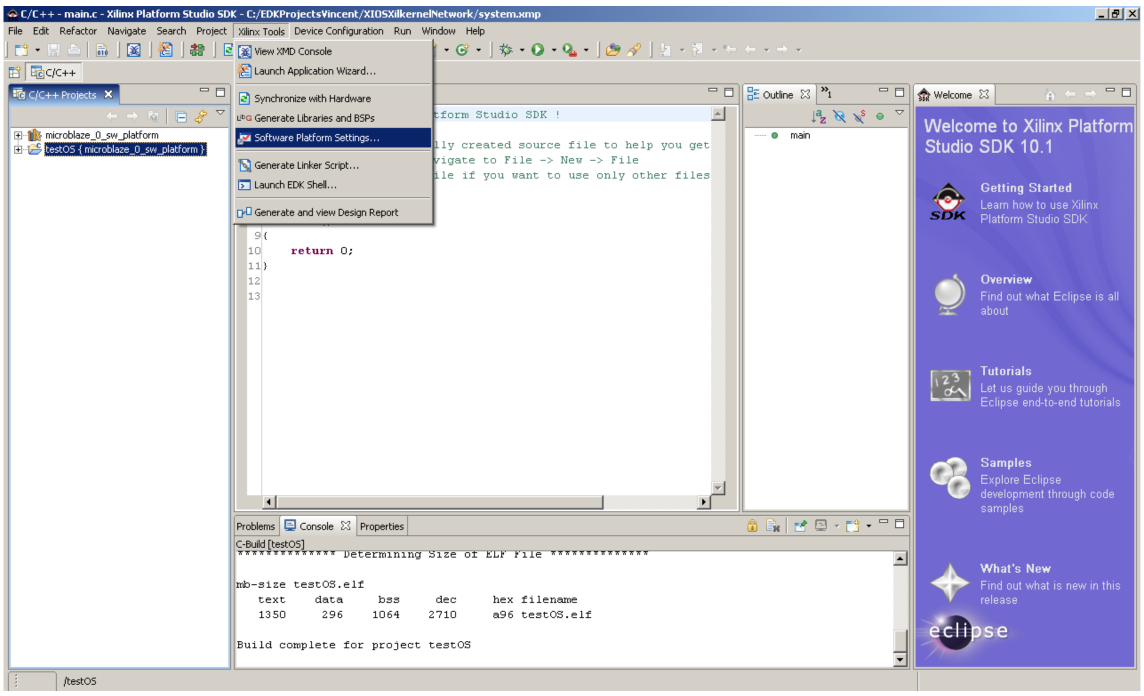
Additional Project Settings

Define the inter-project dependencies, if any.

Referenced C/C++ Projects

- [x] microblaze_0_sw_platform

Now you have generated a "New SDK project". The first thing we are going to do is setting all the "Software Platform Settings" to the values we want. Remember we want to get a xilkernel platform on the Xilinx Spartan 3E Starter board. So for now select "Xilinx Tools" and "Software Platform Settings" from the top-down menu.

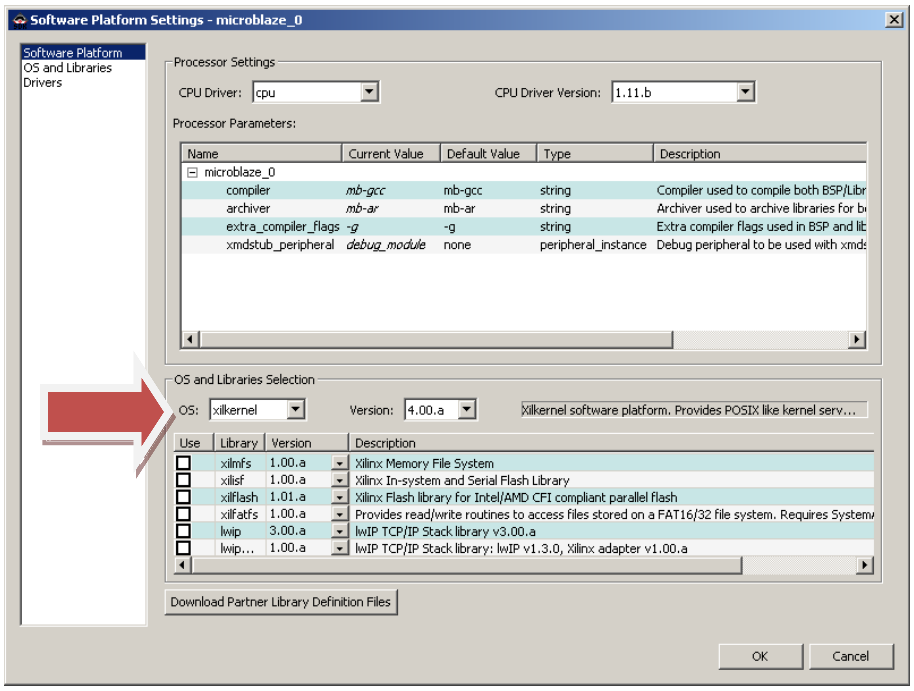


C/C++ - main.c - Xilinx Platform Studio SDK - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp

```
************** Determining Size of ELF File **************

mb-size testOS.elf
   text    data     bss     dec     hex filename
   1350     296    1064    2710     a96 testOS.elf

Build complete for project testOS
```

File
Edit
Refactor
Navigate
Search
Project
Xilinx Tools
Device Configuration
Run
Window
Help
View XMD Console
Launch Application Wizard...
Synchronize with Hardware
Generate Libraries and BSPs
Software Platform Settings...
Generate Linker Script...
Launch EDK Shell...
Generate and view Design Report
C/C++ Projects
microblaze_0_sw_platform
testOS { microblaze_0_sw_platform }
Outline
main
Welcome
Welcome to Xilinx Platform Studio SDK 10.1
Getting Started
Learn how to use Xilinx Platform Studio SDK
Overview
Find out what Eclipse is all about
Tutorials
Let us guide you through Eclipse end-to-end tutorials
Samples
Explore Eclipse development through code samples
What's New
Find out what is new in this release
Problems
Console
Properties
C-Build [testOS]
/testOS

In the "Software Platform Settings" we have to configure the software part of our platform.
On the second part of the window you must select "xilkernel".



Software Platform Settings - microblaze_0

Processor Settings

CPU Driver: cpu   CPU Driver Version: 1.11.b

Processor Parameters:

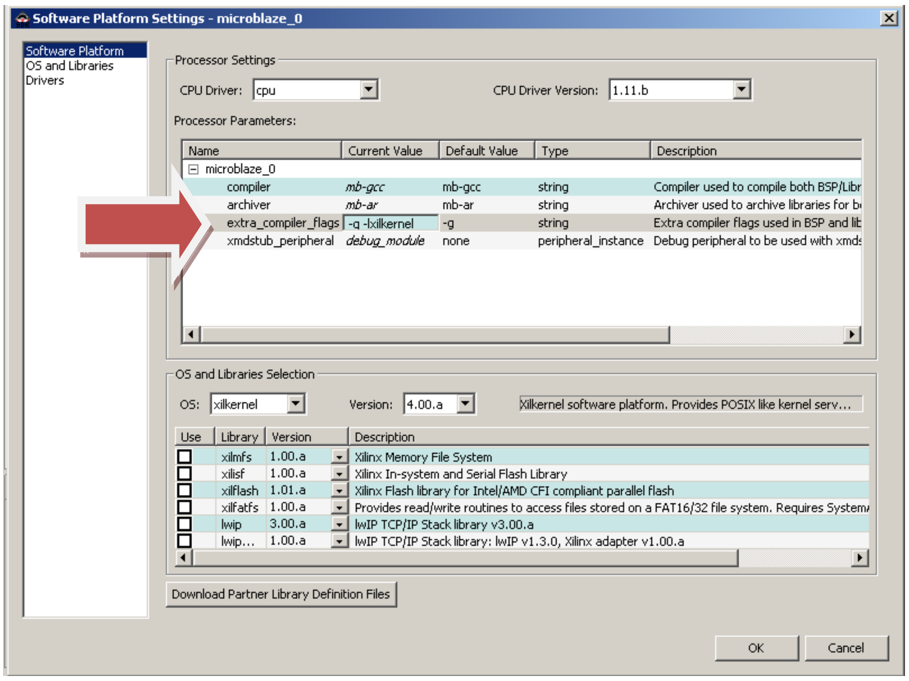
Name	Current Value	Default Value	Type	Description
microblaze_0				
compiler	mb-gcc	mb-gcc	string	Compiler used to compile both BSP/Libr
archiver	mb-ar	mb-ar	string	Archiver used to archive libraries for b
extra_compiler_flags	-g	-g	string	Extra compiler flags used in BSP and lib
xmdstub_peripheral	debug_module	none	peripheral_instance	Debug peripheral to be used with xmds

OS and Libraries Selection

OS: xilkernel   Version: 4.00.a   Xilkernel software platform. Provides POSIX like kernel serv...

Use	Library	Version	Description
☐	xilmfs	1.00.a	Xilinx Memory File System
☐	xilisf	1.00.a	Xilinx In-system and Serial Flash Library
☐	xilflash	1.01.a	Xilinx Flash library for Intel/AMD CFI compliant parallel flash
☐	xilfatfs	1.00.a	Provides read/write routines to access files stored on a FAT16/32 file system. Requires System
☐	lwip	3.00.a	lwIP TCP/IP Stack library v3.00.a
☐	lwip...	1.00.a	lwIP TCP/IP Stack library: lwIP v1.3.0, Xilinx adapter v1.00.a

When we have developed our application and would like to use the features of the xilkernel we have to add "-lxilkernel" as an extra_compiler_flag in the "Software Platform Settings".



Software Platform Settings - microblaze_0

Processor Settings

CPU Driver: cpu   CPU Driver Version: 1.11.b

Processor Parameters:

Name	Current Value	Default Value	Type	Description
microblaze_0				
compiler	mb-gcc	mb-gcc	string	Compiler used to compile both BSP/Libr
archiver	mb-ar	mb-ar	string	Archiver used to archive libraries for b
extra_compiler_flags	-g -lxilkernel	-g	string	Extra compiler flags used in BSP and lib
xmdstub_peripheral	debug_module	none	peripheral_instance	Debug peripheral to be used with xmds

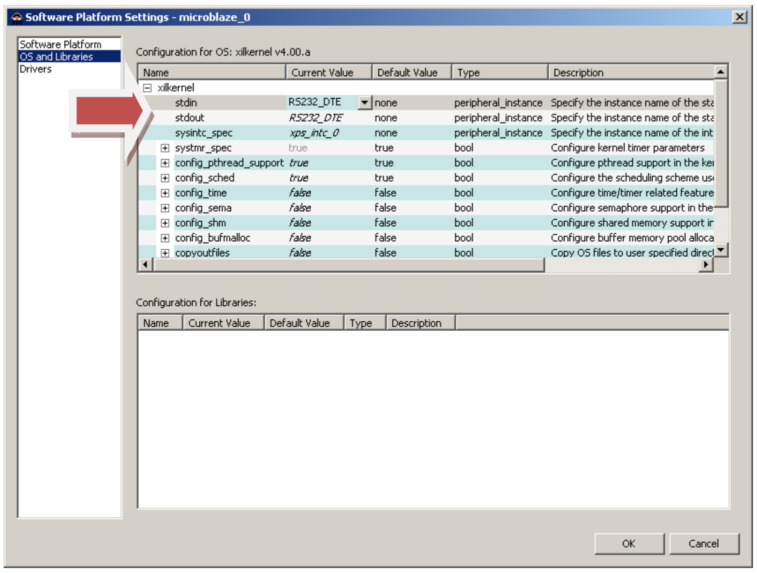
OS and Libraries Selection

OS: xilkernel   Version: 4.00.a   Xilkernel software platform. Provides POSIX like kernel serv...

Use	Library	Version	Description
☐	xilmfs	1.00.a	Xilinx Memory File System
☐	xilisf	1.00.a	Xilinx In-system and Serial Flash Library
☐	xilflash	1.01.a	Xilinx Flash library for Intel/AMD CFI compliant parallel flash
☐	xilfatfs	1.00.a	Provides read/write routines to access files stored on a FAT16/32 file system. Requires System
☐	lwip	3.00.a	lwIP TCP/IP Stack library v3.00.a
☐	lwip...	1.00.a	lwIP TCP/IP Stack library: lwIP v1.3.0, Xilinx adapter v1.00.a

After this you select the “OS and libraries” option from the left part of the “Software Platform Settings” window. Here we have to set some configuration parameters of the OS (xilkernel v4.00.a).

1. Check that you put in the stdin and stdout configuration parameters “RS232_DTE” as current value if you want to use this serial port as your standard input and output parameters.
2. Check that you select xps_intc_0 as the sysintc_spec.



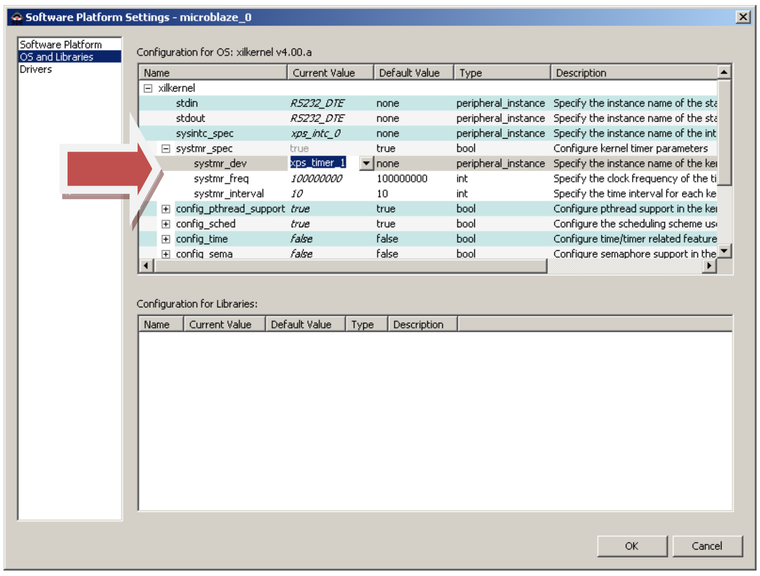
Configuration for OS: xilkernel v4.00.a

Name	Current Value	Default Value	Type	Description
xilkernel				
stdin	RS232_DTE	none	peripheral_instance	Specify the instance name of the sta
stdout	RS232_DTE	none	peripheral_instance	Specify the instance name of the sta
sysintc_spec	xps_intc_0	none	peripheral_instance	Specify the instance name of the int
systmr_spec	true	true	bool	Configure kernel timer parameters
config_pthread_support	true	true	bool	Configure pthread support in the ker
config_sched	true	true	bool	Configure the scheduling scheme us
config_time	false	false	bool	Configure time/timer related feature
config_sema	false	false	bool	Configure semaphore support in the
config_shm	false	false	bool	Configure shared memory support ir
config_bufmalloc	false	false	bool	Configure buffer memory pool alloca
copyoutfiles	false	false	bool	Copy OS files to user specified direc

Configuration for Libraries:

Name	Current Value	Default Value	Type	Description

3. For the systmr_spec set the systmr_dev to xps_timer_1.



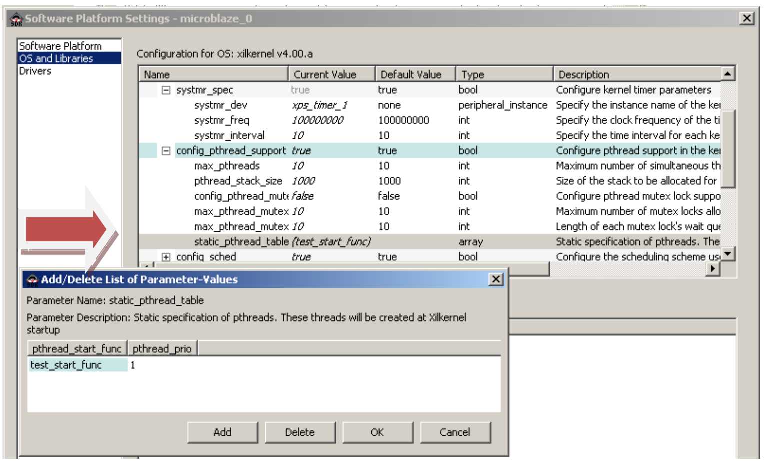
Configuration for OS: xilkernel v4.00.a

Name	Current Value	Default Value	Type	Description
xilkernel				
stdin	RS232_DTE	none	peripheral_instance	Specify the instance name of the sta
stdout	RS232_DTE	none	peripheral_instance	Specify the instance name of the sta
sysintc_spec	xps_intc_0	none	peripheral_instance	Specify the instance name of the int
systmr_spec	true	true	bool	Configure kernel timer parameters
systmr_dev	xps_timer_1	none	peripheral_instance	Specify the instance name of the ker
systmr_freq	100000000	100000000	int	Specify the clock frequency of the ti
systmr_interval	10	10	int	Specify the time interval for each ke
config_pthread_support	true	true	bool	Configure pthread support in the ker
config_sched	true	true	bool	Configure the scheduling scheme us
config_time	false	false	bool	Configure time/timer related feature
config_sema	false	false	bool	Configure semaphore support in the

Configuration for Libraries:

Name	Current Value	Default Value	Type	Description

4. Go to the config_pthread_support parameter and select the static_pthread_table option, you do this by clicking on the "Current Value" of this option.
   Be sure to put in this table the value "test_start_func" as pthread_start_func and put 1 in the column for the pthread_prio value of it.



Software Platform Settings - microblaze_0

Software Platform
OS and Libraries
Drivers

Configuration for OS: xilkernel v4.00.a

Name	Current Value	Default Value	Type	Description
⊟ systmr_spec	true	true	bool	Configure kernel timer parameters
systmr_dev	xps_timer_1	none	peripheral_instance	Specify the instance name of the ker
systmr_freq	100000000	100000000	int	Specify the clock frequency of the ti
systmr_interval	10	10	int	Specify the time interval for each ke
⊟ config_pthread_support	true	true	bool	Configure pthread support in the ker
max_pthreads	10	10	int	Maximum number of simultaneous th
pthread_stack_size	1000	1000	int	Size of the stack to be allocated for
config_pthread_mute	false	false	bool	Configure pthread mutex lock suppo
max_pthread_mutex	10	10	int	Maximum number of mutex locks allo
max_pthread_mutex	10	10	int	Length of each mutex lock's wait que
static_pthread_table	(test_start_func)		array	Static specification of pthreads. The
⊞ config_sched	true	true	bool	Configure the scheduling scheme us

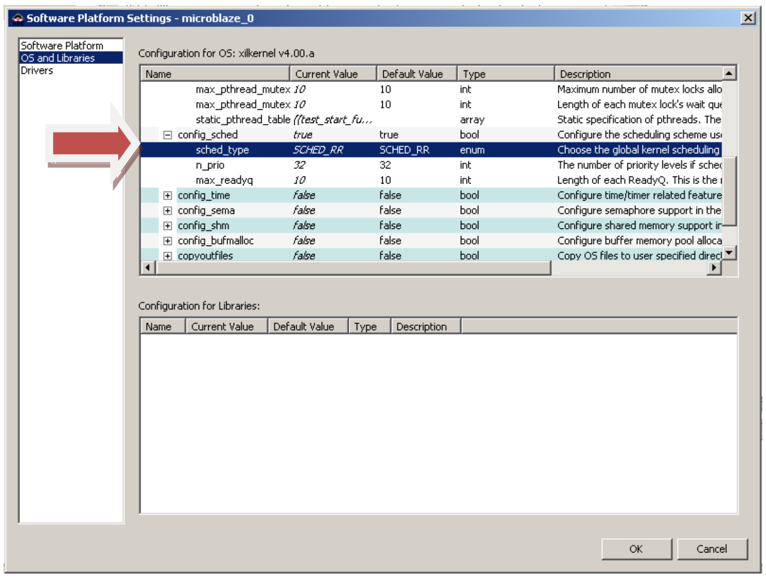
Add/Delete List of Parameter-Values

Parameter Name: static_pthread_table

Parameter Description: Static specification of pthreads. These threads will be created at Xilkernel startup

pthread_start_func	pthread_prio
test_start_func	1

5. Be sure that the sched_type option which you can find under config_sched is set to use Round Robin (SCHED_RR) otherwise you will get some errors while compiling your code.



Software Platform Settings - microblaze_0

Software Platform
OS and Libraries
Drivers

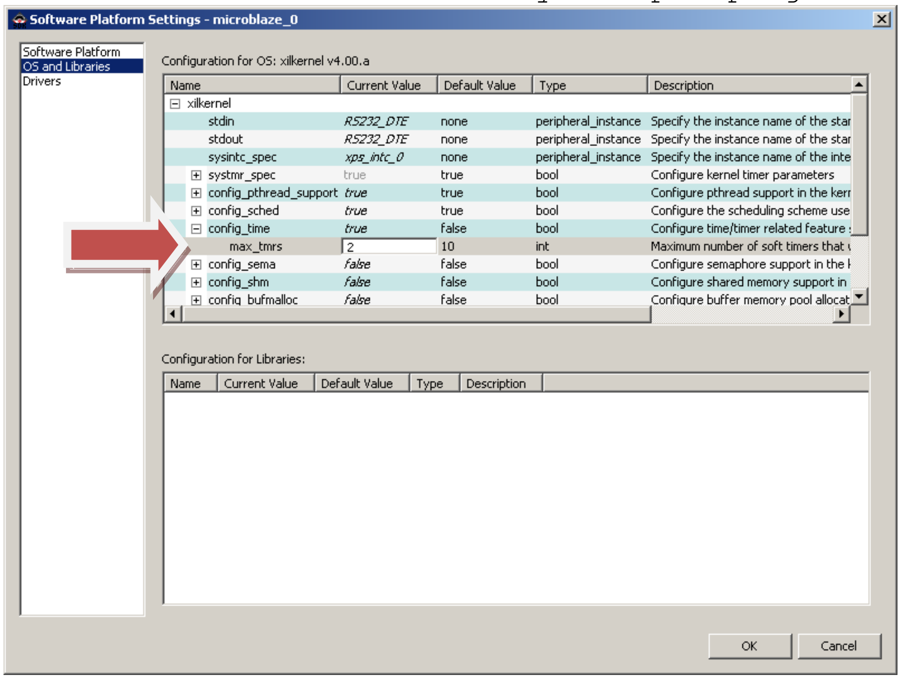
Configuration for OS: xilkernel v4.00.a

Name	Current Value	Default Value	Type	Description
max_pthread_mutex	10	10	int	Maximum number of mutex locks allo
max_pthread_mutex	10	10	int	Length of each mutex lock's wait que
static_pthread_table	((test_start_fu...		array	Static specification of pthreads. The
⊟ config_sched	true	true	bool	Configure the scheduling scheme us
sched_type	SCHED_RR	SCHED_RR	enum	Choose the global kernel scheduling
n_prio	32	32	int	The number of priority levels if sched
max_readyq	10	10	int	Length of each ReadyQ. This is the i
⊞ config_time	false	false	bool	Configure time/timer related feature
⊞ config_sema	false	false	bool	Configure semaphore support in the
⊞ config_shm	false	false	bool	Configure shared memory support ir
⊞ config_bufmalloc	false	false	bool	Configure buffer memory pool alloca
⊞ copyoutfiles	false	false	bool	Copy OS files to user specified direc

Configuration for Libraries:

Name	Current Value	Default Value	Type	Description

6. The xilkernel can deliver you some software timers. But if you want to use this option of the xilkernel your have to make the max_tmrs option to a value that is greater than 1. Before you can do this you have to put the config_time option to the value "true". I have used the value 2 in my example project.



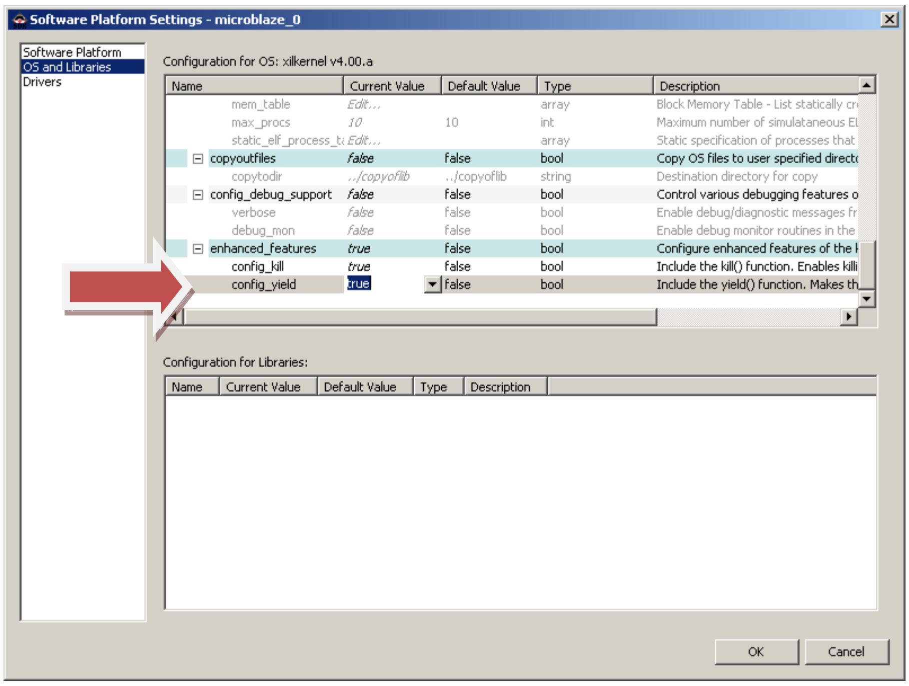
Configuration for OS: xilkernel v4.00.a

Name	Current Value	Default Value	Type	Description
xilkernel				
stdin	R5232_DTE	none	peripheral_instance	Specify the instance name of the star
stdout	R5232_DTE	none	peripheral_instance	Specify the instance name of the star
sysintc_spec	xps_intc_0	none	peripheral_instance	Specify the instance name of the inte
systmr_spec	true	true	bool	Configure kernel timer parameters
config_pthread_support	true	true	bool	Configure pthread support in the kerr
config_sched	true	true	bool	Configure the scheduling scheme use
config_time	true	false	bool	Configure time/timer related feature :
max_tmrs	2	10	int	Maximum number of soft timers that (
config_sema	false	false	bool	Configure semaphore support in the k
config_shm	false	false	bool	Configure shared memory support in
config_bufmalloc	false	false	bool	Configure buffer memory pool allocat

Configuration for Libraries:

Name	Current Value	Default Value	Type	Description

7. In the example software application we will kill a pthread on runtime. If you want to be able to do this you have to put the value of enhanced_features to true and set config_kill and config_yield also to true.



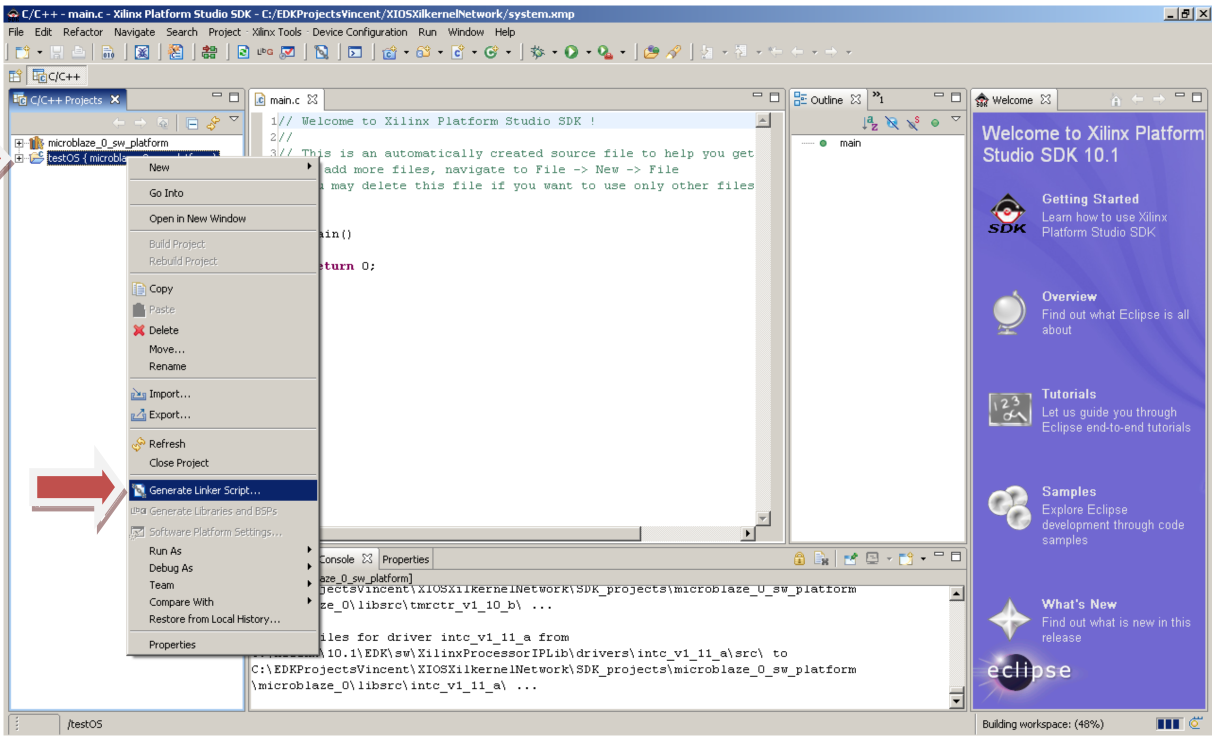
Configuration for OS: xilkernel v4.00.a

Name	Current Value	Default Value	Type	Description
mem_table	Edit...		array	Block Memory Table - List statically cr
max_procs	10	10	int	Maximum number of simulataneous El
static_elf_process_t	Edit...		array	Static specification of processes that
copyoutfiles	false	false	bool	Copy OS files to user specified directo
copytodir	../copyoflib	../copyoflib	string	Destination directory for copy
config_debug_support	false	false	bool	Control various debugging features o
verbose	false	false	bool	Enable debug/diagnostic messages fr
debug_mon	false	false	bool	Enable debug monitor routines in the
enhanced_features	true	false	bool	Configure enhanced features of the k
config_kill	true	false	bool	Include the kill() function. Enables killi
config_yield	true	false	bool	Include the yield() function. Makes th

Configuration for Libraries:

Name	Current Value	Default Value	Type	Description

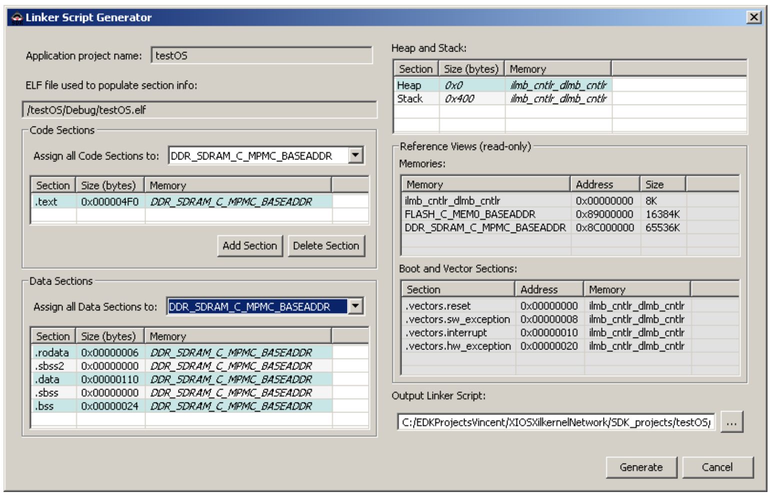
After this you can close this "Software Platform Settings" window by clicking on the "OK" button. After this you will be back in the Xilinx Platform Studio SDK.
We are going to manipulate the "Linker Script" used by our project in such a way that we run our software directly from DDR-SDRAM. We do this by clicking on the software project you have generated with the right mouse button and then selecting "Generate Linker Script".


C/C++ - main.c - Xilinx Platform Studio SDK - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp
File Edit Refactor Navigate Search Project Xilinx Tools Device Configuration Run Window Help
C/C++ Projects
microblaze_0_sw_platform
testOS { microbla
New
Go Into
Open in New Window
Build Project
Rebuild Project
Copy
Paste
Delete
Move...
Rename
Import...
Export...
Refresh
Close Project
Generate Linker Script...
Generate Libraries and BSPs
Software Platform Settings...
Run As
Debug As
Team
Compare With
Restore from Local History...
Properties
main.c
Outline
main
Console
Properties
Welcome
Welcome to Xilinx Platform Studio SDK 10.1
Getting Started
Learn how to use Xilinx Platform Studio SDK
Overview
Find out what Eclipse is all about
Tutorials
Let us guide you through Eclipse end-to-end tutorials
Samples
Explore Eclipse development through code samples
What's New
Find out what is new in this release
eclipse
/testOS
Building workspace: (48%)

```
// Welcome to Xilinx Platform Studio SDK !
//
// This is an automatically created source file to help you get
  add more files, navigate to File -> New -> File
  may delete this file if you want to use only other files

ain()

eturn 0;
```

```
aze_0_sw_platform]
jectsVincent\XIOSXilkernelNetwork\SDK_projects\microblaze_0_sw_platform
ze_0\libsrc\tmrctr_v1_10_b\ ...

iles for driver intc_v1_11_a from
\10.1\EDK\sw\XilinxProcessorIPLib\drivers\intc_v1_11_a\src\ to
C:\EDKProjectsVincent\XIOSXilkernelNetwork\SDK_projects\microblaze_0_sw_platform
\microblaze_0\libsrc\intc_v1_11_a\ ...
```

After this a new window appears titled "Linker Script Generator". In this window you have to assign all the Code Sections and Data Sections of the ELF file to "DDR_SDRAM_C_MPMC_BASEADDR".



Application project name: testOS

ELF file used to populate section info: /testOS/Debug/testOS.elf

Code Sections

Assign all Code Sections to: DDR_SDRAM_C_MPMC_BASEADDR

Section	Size (bytes)	Memory
.text	0x000004F0	DDR_SDRAM_C_MPMC_BASEADDR

Data Sections

Assign all Data Sections to: DDR_SDRAM_C_MPMC_BASEADDR

Section	Size (bytes)	Memory
.rodata	0x00000006	DDR_SDRAM_C_MPMC_BASEADDR
.sbss2	0x00000000	DDR_SDRAM_C_MPMC_BASEADDR
.data	0x00000110	DDR_SDRAM_C_MPMC_BASEADDR
.sbss	0x00000000	DDR_SDRAM_C_MPMC_BASEADDR
.bss	0x00000024	DDR_SDRAM_C_MPMC_BASEADDR

Heap and Stack:

Section	Size (bytes)	Memory
Heap	0x0	ilmb_cntlr_dlmb_cntlr
Stack	0x400	ilmb_cntlr_dlmb_cntlr

Reference Views (read-only)

Memories:

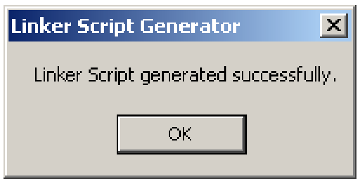
Memory	Address	Size
ilmb_cntlr_dlmb_cntlr	0x00000000	8K
FLASH_C_MEM0_BASEADDR	0x89000000	16384K
DDR_SDRAM_C_MPMC_BASEADDR	0x8C000000	65536K

Boot and Vector Sections:

Section	Address	Memory
.vectors.reset	0x00000000	ilmb_cntlr_dlmb_cntlr
.vectors.sw_exception	0x00000008	ilmb_cntlr_dlmb_cntlr
.vectors.interrupt	0x00000010	ilmb_cntlr_dlmb_cntlr
.vectors.hw_exception	0x00000020	ilmb_cntlr_dlmb_cntlr

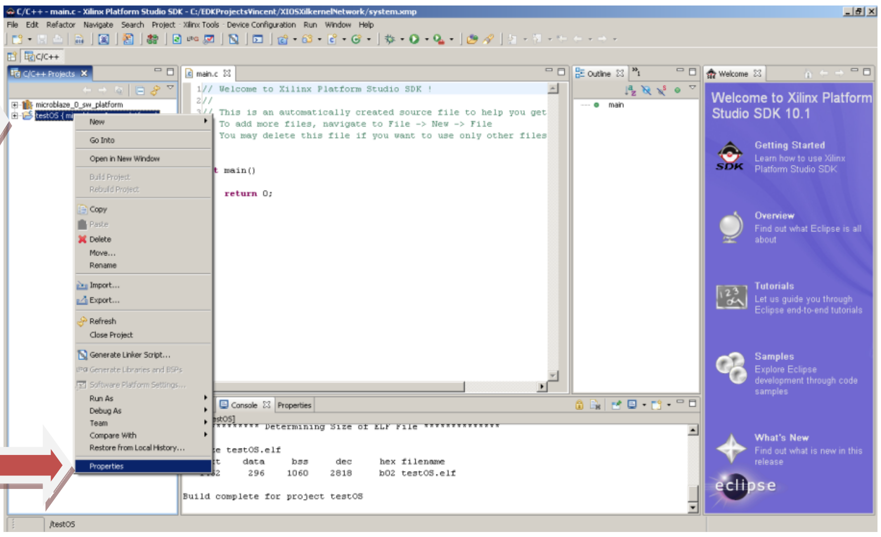
Output Linker Script: C:/EDKProjectsVincent/XIOSXilkernelNetwork/SDK_projects/testOS/

Click on the “Generate” button.


Linker Script Generator

Linker Script generated successfully.

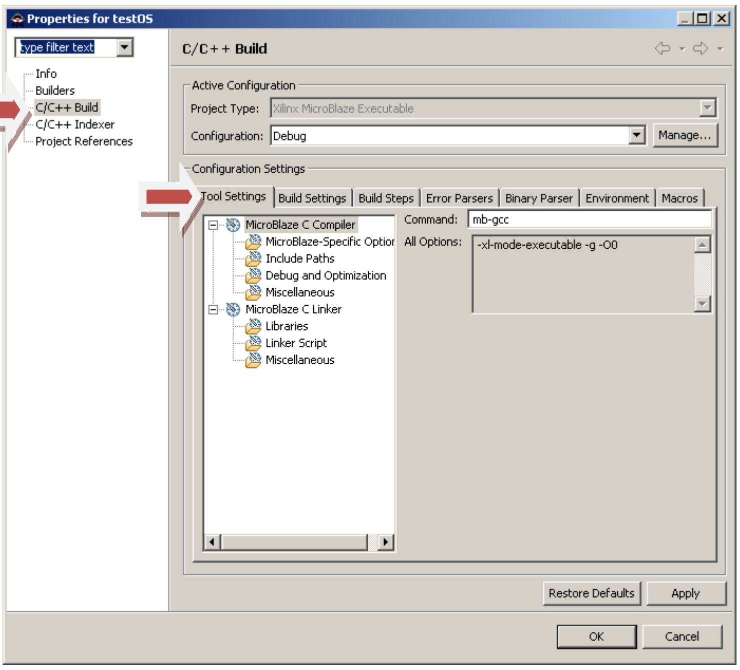
After this we have to tell the gcc linker to link against the xilkernel library. We do this by clicking with the "right" mouse button on your project and selecting "properties".


C/C++ - main.c - Xilinx Platform Studio SDK - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp

```
// Welcome to Xilinx Platform Studio SDK !
//
// This is an automatically created source file to help you get
   To add more files, navigate to File -> New -> File
   You may delete this file if you want to use only other files

t main()

 return 0;
```

```
******** Determining Size of ELF File **************

e testOS.elf
t      data     bss      dec      hex filename
2       296     1060     2818     b02 testOS.elf

Build complete for project testOS
```

Go to C/C++ Build and select the "Tool Settings" tab.


Properties for testOS

C/C++ Build

Active Configuration

Project Type: Xilinx MicroBlaze Executable

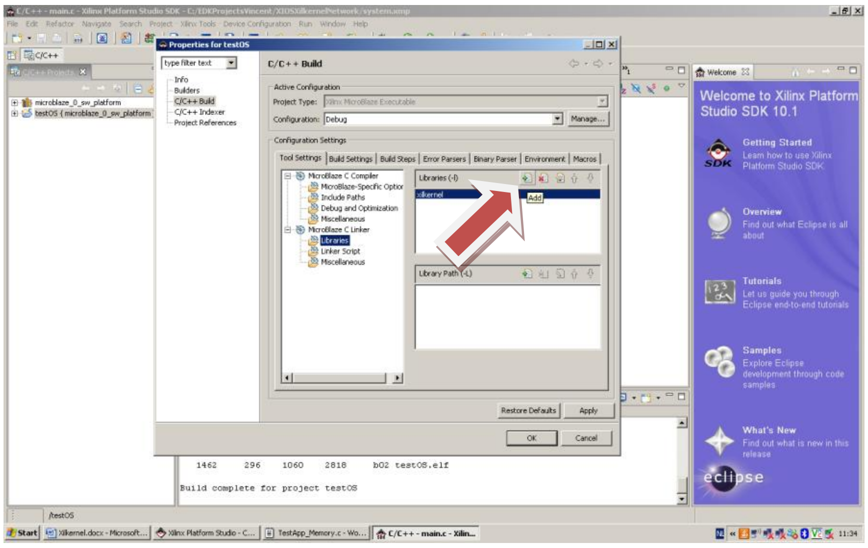
Configuration: Debug

Configuration Settings

Command: mb-gcc

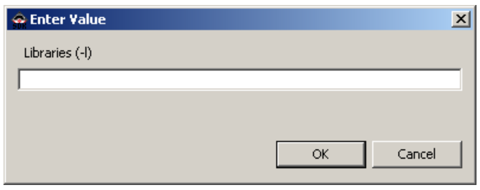
All Options: -xl-mode-executable -g -O0

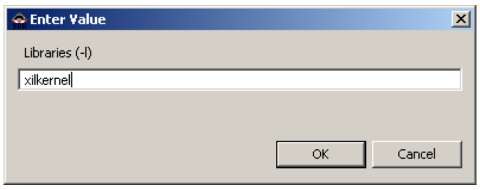
Click on the “Add” icon.



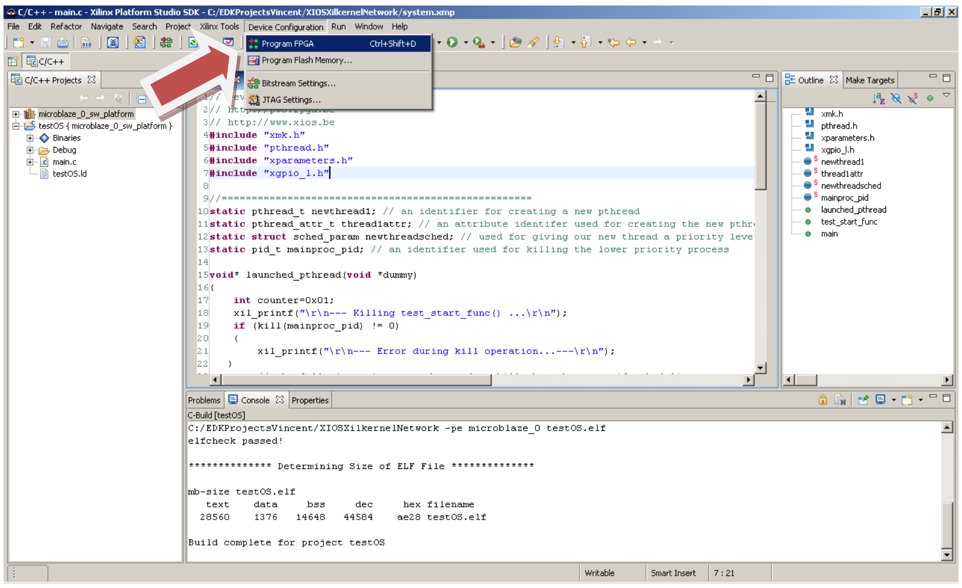
```
   1462    296   1060   2818    b02 testOS.elf

Build complete for project testOS
```

A new messagebox appears, you have to write xilkernel as library in this box and click the “OK” button.





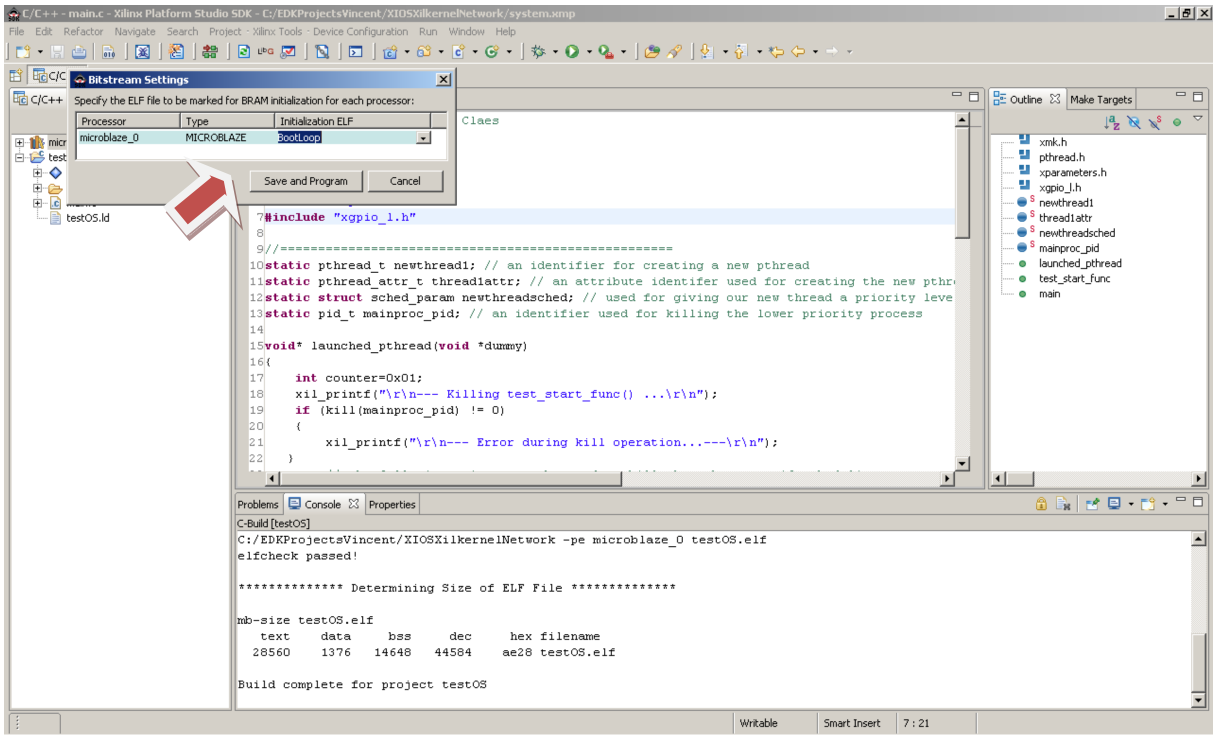
Now it is time to program your FPGA with the generated bitstream. Do this by clickin on the “Device Configuration” menuitem and select “Program FPGA”. This will download the .bit file to your Spartan3E FPGA.



```
// http://www.xios.be
#include "xmk.h"
#include "pthread.h"
#include "xparameters.h"
#include "xgpio_l.h"

//====================================================
static pthread_t newthread1; // an identifier for creating a new pthread
static pthread_attr_t thread1attr; // an attribute identifer used for creating the new pthr
static struct sched_param newthreadsched; // used for giving our new thread a priority leve
static pid_t mainproc_pid; // an identifier used for killing the lower priority process

void* launched_pthread(void *dummy)
{
    int counter=0x01;
    xil_printf("\r\n--- Killing test_start_func() ...\r\n");
    if (kill(mainproc_pid) != 0)
    {
        xil_printf("\r\n--- Error during kill operation...---\r\n");
    }
```

```
C:/EDKProjectsVincent/XIOSXilkernelNetwork -pe microblaze_0 testOS.elf
elfcheck passed!

************** Determining Size of ELF File **************

mb-size testOS.elf
   text    data     bss     dec     hex filename
  28560    1376   14648   44584    ae28 testOS.elf

Build complete for project testOS
```

The "Program FPGA" selection will start up a new window where you have to select the "BootLoop" ELF to be used in BRAM. This will place a small boot program into the BRAM that point to the start place in DDR-SDRAM where we will place our application software. So click the "Save and Program" button.



Specify the ELF file to be marked for BRAM initialization for each processor:

Processor	Type	Initialization ELF
microblaze_0	MICROBLAZE	BootLoop

```
Claes

#include "xgpio_l.h"

//====================================================
static pthread_t newthread1; // an identifier for creating a new pthread
static pthread_attr_t thread1attr; // an attribute identifer used for creating the new pthr
static struct sched_param newthreadsched; // used for giving our new thread a priority leve
static pid_t mainproc_pid; // an identifier used for killing the lower priority process

void* launched_pthread(void *dummy)
{
    int counter=0x01;
    xil_printf("\r\n--- Killing test_start_func() ...\r\n");
    if (kill(mainproc_pid) != 0)
    {
        xil_printf("\r\n--- Error during kill operation...---\r\n");
    }
```

C-Build [testOS]

```
C:/EDKProjectsVincent/XIOSXilkernelNetwork -pe microblaze_0 testOS.elf
elfcheck passed!

************** Determining Size of ELF File **************

mb-size testOS.elf
   text    data     bss     dec     hex filename
  28560    1376   14648   44584    ae28 testOS.elf

Build complete for project testOS
```

Step 3: Write Software Application

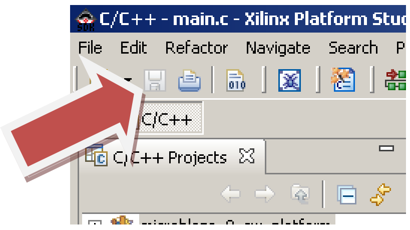
Now it is time to program our C-application that we would like to debug on our HW platform. This C-application we use the xilkernel as Real-Time Operating System (RTOS)

In your Xilinx Platform Studio SDK click on the "main.c" file and remove all the code that is in the code window with the code below (explanation of the code is as comment included):

```
// Development by Vincent Claes
// http://pwo.fpga.be
// http://www.xios.be

#include "xmk.h"              // Xilinx Micro Kernel library
#include "pthread.h"          // pthread library
#include "xparameters.h"      // Platform parameters
#include "xgpio.h"
#include "xgpio_l.h"

//=====================================================
static pthread_t newthread1; // an identifier for creating a new pthread
static pthread_attr_t thread1attr;
// an attribute identifier used for creating the new pthread
static struct sched_param newthreadsched;
// used for giving our new thread a priority level
static pid_t mainproc_pid;
// an identifier used for killing the lower priority process

XGpio GpioOutput;
/* The driver instance for GPIO Device configured as O/P */

void* launched_pthread(void *dummy)
{
      int counter=0x01;
      int Status;
      xil_printf("\r\n--- Killing test_start_func() ...\r\n");

      // next code used to kill the other proc if scheduling
      // is Round Robin (RR-based) instead of Priority based.

      if (kill(mainproc_pid) != 0)
      {
            xil_printf("\r\n--- Error during kill operation...---\r\n");
   }
    Status = XGpio_Initialize(&GpioOutput,XPAR_LEDS_8BIT_BASEADDR);
      XGpio_SetDataDirection(&GpioOutput,1,0x0);

      while(1)
      {
            XGpio_DiscreteWrite(&GpioOutput,1,counter);
            xil_printf("\r\n%d",counter);
            sleep(5000); // This is why we need a software timer!
            counter++;
      }

}

void* test_start_func(void *dummy)
{
      int i=0,temp;

      xil_printf("--- In test_start_funct() ---");
      mainproc_pid = get_currentPID();    // parameter needed to kill this
                                          //process in other thread

      while(1)
      {
            i++;
            if(i==25)
            // spawn the thread after some time
            {
                  pthread_attr_init(&thread1attr);
                  newthreadsched.sched_priority = 1;
                  temp = pthread_create(&newthread1, &thread1attr,
(void*)launched_pthread, NULL);
// NULL: It's used if the thread function must take in an argument
            }
      }
}

int main (void) {
      xil_printf("-- Entering main() --\r\n");
```

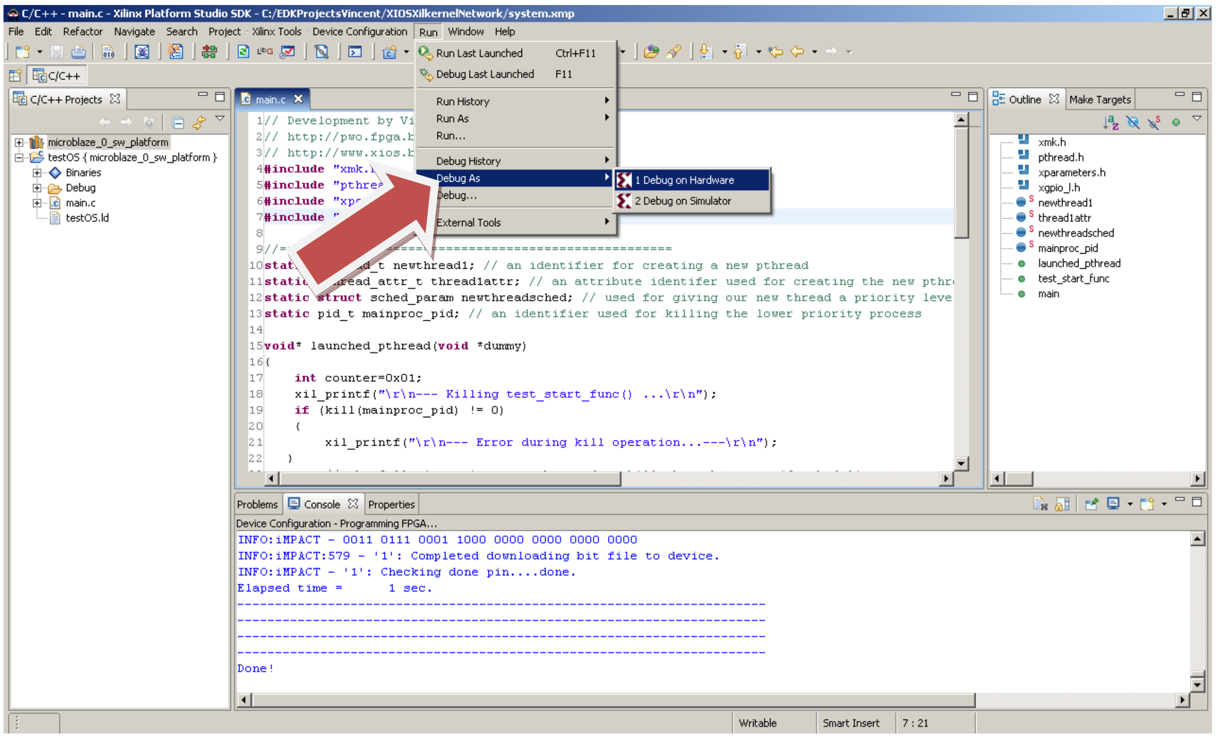
```c
        //Wait for user input to start xilkernel...
        xilkernel_main();
        //We never reach this line...
        xil_printf("-- Exiting main() --\r\n");
    return 0;
}
```

Click now on the "save" icon; the software then will be automatically compiled by the mb-gcc (microblaze port of the gcc compiler) and linked.



Step 4: Debug Application

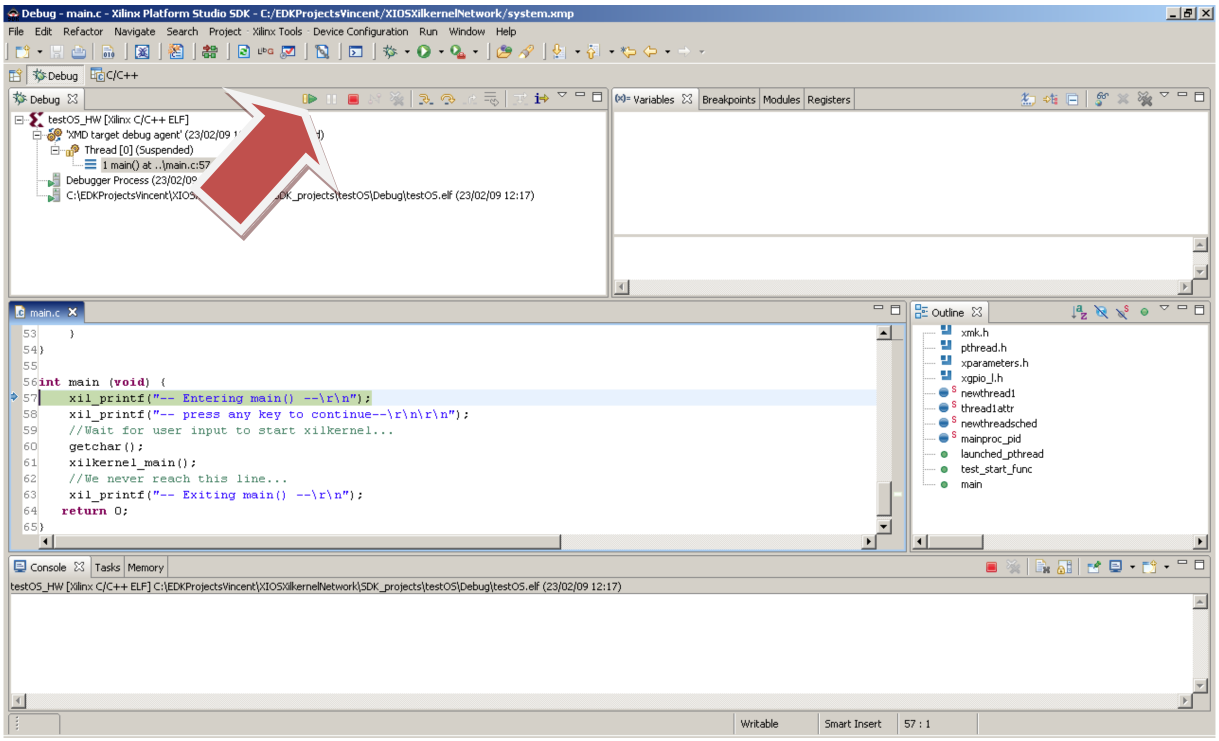
Now it's time for the final step: debug the application from DDR-SDRAM. Be sure you have connected your board correctly to the computer and it is powered on. Also connect your serial DTE port to a RS-232 port of your computer. Open a serial connection from your computer to the Spartan 3E board; for instance by use of "Hyperterminal" and use 9600 bps as baud rate.
Click on "Debug on Hardware" this option you can find from the menu-item "Run" and then selecting "Debug As".



```c
// Development by Vi
// http://pwo.fpga.b
// http://www.xios.b
#include "xmk.
#include "pthre
#include "xp
#include "

//=
stat         d_t newthread1; // an identifier for creating a new pthread
static      read_attr_t thread1attr; // an attribute identifer used for creating the new pthr
static struct sched_param newthreadsched; // used for giving our new thread a priority leve
static pid_t mainproc_pid; // an identifier used for killing the lower priority process

void* launched_pthread(void *dummy)
{
    int counter=0x01;
    xil_printf("\r\n--- Killing test_start_func() ...\r\n");
    if (kill(mainproc_pid) != 0)
    {
        xil_printf("\r\n--- Error during kill operation...---\r\n");
    }
```

```
Device Configuration - Programming FPGA...
INFO:iMPACT - 0011 0111 0001 1000 0000 0000 0000 0000
INFO:iMPACT:579 - '1': Completed downloading bit file to device.
INFO:iMPACT - '1': Checking done pin....done.
Elapsed time =       1 sec.
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------
Done!
```

The next screen will appear; the final step is clicking the "Resume" button! (the debugger will always pause the application on the first line of the main() function of your application.


Debug - main.c - Xilinx Platform Studio SDK - C:/EDKProjectsVincent/XIOSXilkernelNetwork/system.xmp

```
testOS_HW [Xilinx C/C++ ELF]
  'XMD target debug agent' (23/02/09 1
    Thread [0] (Suspended)
      1 main() at ..\main.c:57
  Debugger Process (23/02/09
  C:\EDKProjectsVincent\XIOS        DK_projects\testOS\Debug\testOS.elf (23/02/09 12:17)
```

```c
    }
}

int main (void) {
    xil_printf("-- Entering main() --\r\n");
    xil_printf("-- press any key to continue--\r\n\r\n");
    //Wait for user input to start xilkernel...
    getchar();
    xilkernel_main();
    //We never reach this line...
    xil_printf("-- Exiting main() --\r\n");
  return 0;
}
```

```
testOS_HW [Xilinx C/C++ ELF] C:\EDKProjectsVincent\XIOSXilkernelNetwork\SDK_projects\testOS\Debug\testOS.elf (23/02/09 12:17)
```

Enjoy.

Vincent Claes
XIOS Hogeschool Limburg
Department of Industrial Sciences and Technology
Universitaire Campus - Agoralaan – Gebouw H
B-3590 Diepenbeek
Belgium
vincent.claes@xios.be
tel.: +32 11 26 00 39
fax: +32 11 26 00 54
mobile: +32 478 35 38 49