

Implementing Linux on the Zynq™-7000 SoC

Lab 1.1

Creating the Zynq First Stage Boot Loader (FSBL)



September 2012
Version 05

Table of Contents

Table of Contents.....	2
Lab Setup.....	3
Lab Instruction Notes.....	4
Technical Support	4
Lab 1.1 Overview.....	5
Lab 1.1 Objectives	5
Experiment 1: Workspace and Hardware Platform.....	6
Experiment 2: Create the FSBL Project and Board Support Package	13
Exploring Further	21
Revision History	21
Resources.....	21
Answers.....	22

Lab Setup

To complete the SpeedWay labs, the following software and hardware setups are recommended.

Software

The recommended software for SpeedWay labs is:

- Xilinx ISE WebPACK 14.2 (Free license and download from Xilinx website)
 - Check Xilinx [AR51895](#) for required installation work-arounds
- Cypress CY7C64225 USB-to-UART Bridge Driver
- Tera Term (Exact version used for this SpeedWay is V4.75)
- VMware Player V5.0.0 (Exact version used for this SpeedWay is VMware-player-5.0.0-812388.exe)
- CentOS V6.3 64-bit installer image (Exact version used for this SpeedWay is CentOS-6.3-x86_64-bin-DVD1.iso)
- Sourcery CodeBench Zynq toolchain (Exact version used for this SpeedWay is xilinx-2011.09-50-arm-xilinx-linux-gnueabi.bin)
- Git SCM toolset (Exact version used for this SpeedWay is V1.7.1)
- Adobe Reader for viewing PDF content (Exact version used for this SpeedWay is Adobe Reader X 10.1.4)

Hardware

The targeted hardware consists of the following:

- PC workstation with at least 5 GB RAM, 30GB free hard disk space, and Windows 7 64-bit operating system
- Available SD card slot on PC or external USB-based SD card reader
- Avnet ZedBoard (**AES-Z7EV-7Z020-G**) – *included in kit*
- USB cable (Type A to Micro-USB Type B) – *included in kit*
- 4GB SD card – *included in kit*
- CAT-5 Ethernet patch cable

Lab Instruction Notes

Throughout all the SpeedWay labs, a generalized instruction is given. If you're comfortable completing the task based on that instruction, feel free to do so. If not, step-by-step instructions are provided.

Technical Support

For technical support with any of the labs, please contact your local Avnet/Silica FAE or visit the ZedBoard.org support forum:

<http://www.zedboard.org/forum>

Additional technical support resources are listed below.

ZedBoard Kit support page with Documentation and Reference Designs

<http://www.zedboard.org/content/support>

For Xilinx technical support, you may contact your local Avnet/Silica FAE or Xilinx Online Technical Support at www.support.xilinx.com. On this site you will also find the following resources for assistance:

- Software, IP, and Documentation Updates
- Access to Technical Support Web Tools
- Searchable Answer Database with Over 4,000 Solutions
- User Forums
- Training - Select instructor-led classes and recorded e-learning options

Contact your Avnet/Silica FAE or Avnet Support for any additional questions regarding the ZedBoard reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design.

<http://www.em.avnet.com/techsupport>

Lab 1.1 Overview

This lab builds upon the skills covered in the Introduction to Zynq SpeedWay where the entire Zynq hardware design flow is covered. The hardware project from that SpeedWay will be leveraged to continue to develop the Linux system used in this SpeedWay. If you are interested in further details of how the hardware platform is created, you can refer back to the materials for the Introduction to Zynq SpeedWay.

Using the hardware design as the basis for our Linux system, we will export the hardware design into SDK. This will give us a platform description from which software can be built upon using the SDK New Project wizard.

A First Stage Boot Loader (FSBL) will be created from a project template and the platform description will be used to initialize our Zynq device.

Lab 1.1 Objectives

When you have completed Lab 1.1, you will know how to do the following:

- Export a hardware design into an SDK workspace
- Create a First Stage Boot Loader from a Zynq hardware design

Experiment 1: Workspace and Hardware Platform

This experiment shows how to export a hardware design to a SDK workspace.

Workspaces

SDK uses the concept of workspaces to hold your software development work. A workspace is a directory in your file system that SDK uses to hold meta-information about the projects you are working with. The workspace also contains your SDK settings, software project files, and logs.

For convenience, SDK can be instructed to remember your last used workspace. When you enable this option, you will not be prompted to choose a workspace every time you start SDK. You can always switch your current workspace in SDK.

Note: It is not safe to copy or move workspaces from one location to another. If you want to copy or move software components in your workspace, you must use the SDK Export/Import functions.

Hardware Platform

The Hardware Platform Specification file captures all the information and files from a Xilinx® PlanAhead (PA) hardware design that is required for a software developer to write, debug, and deploy software applications for that hardware.

Typically, a hardware designer who develops hardware using Xilinx Platform Studio (XPS) creates the Hardware Platform Specification in a directory. The software developer then uses this information in the Xilinx Software Development Kit (SDK). The main components of this specification are:

- A hardware description in an XML format
- An FPGA bitstream corresponding to the hardware description
- A series of source code files related to the Processing System (PS) initialization (ps7_init.c, ps7_init.h, ps7_init.tcl, and ps7_init.html) which are directly related to the Zynq PS configuration in XPS

SDK uses the hardware description file to detect the processor and memory mapped peripherals present in the hardware.

The FPGA bitstream is used to program the Programmable Logic (PL) subsystem with the hardware created by the hardware designer.

The ps7_init files are used to initialize the Zynq Processing System (PS) with the hardware settings that are specified in the XPS Zynq configuration tool.

Experiment 1 General Instruction:

Launch Xilinx PlanAhead and open the project in the Lab 1.1 directory then export the hardware for SDK.

Experiment 1 Step-by-Step Instructions:

1. Launch PlanAhead by selecting **Start → All Programs → Xilinx Design Tools → ISE Design Suite 14.2 → PlanAhead → PlanAhead**. Alternatively, PlanAhead can be launched using the **PlanAhead 14.2** Windows desktop shortcut.



Figure 1 – The PlanAhead Application Icon

2. Once PlanAhead is open, select the **Open Project** option under the **Getting Started** action list.

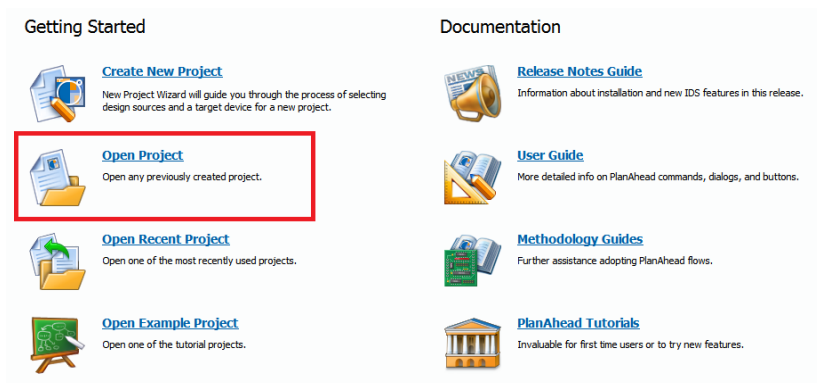


Figure 2 – Open Project as Shown in PlanAhead Action List

3. In the **Open Project** dialog, select the PlanAhead project **LED_Controller.ppr** from the **C:\Speedway\Fall_12\Zynq_Linux\lab1_1\LED_Controller** folder and click the **OK** button.

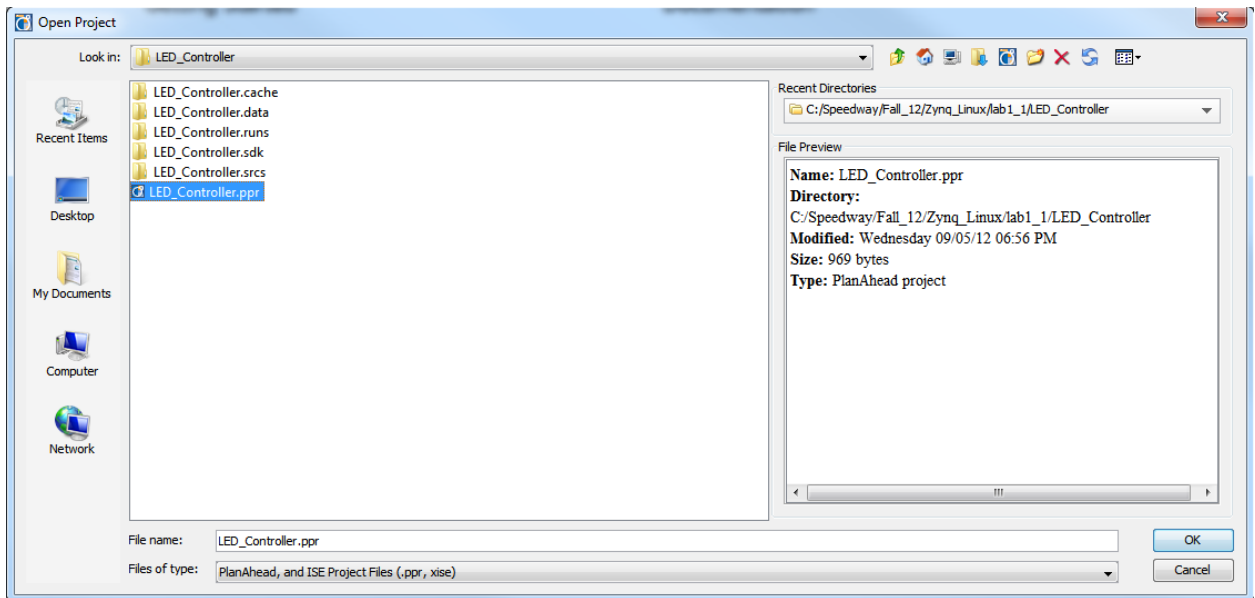


Figure 3 – Opening the LED_Controller PlanAhead Project

4. Expand the top-level **system_stub.v** entry in the PlanAhead Sources pane as seen in Figure 4.

The **system_stub.v** entry is the top level wrapper for the embedded subsystem that contains our PS7 configuration.

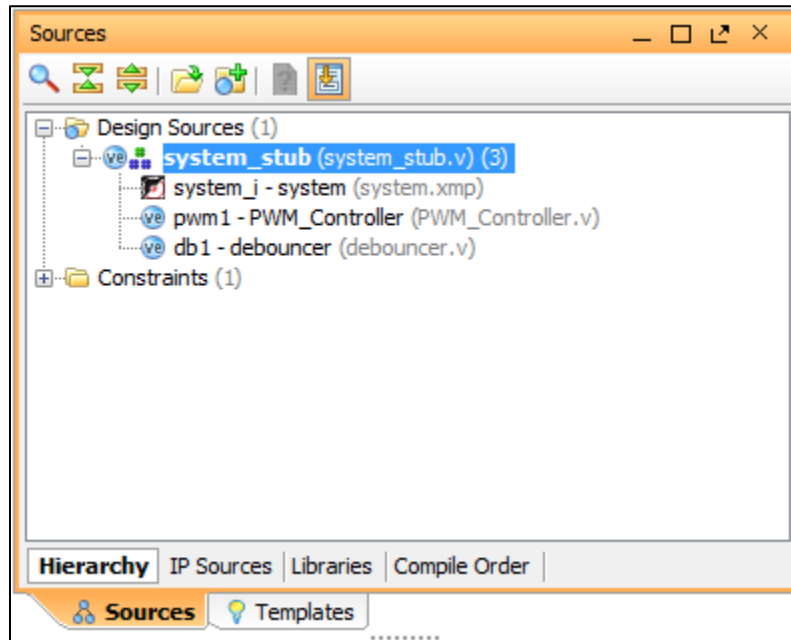


Figure 4 – Selecting the Embedded Subsystem in the PlanAhead Sources Pane

5. Export the selected hardware platform to SDK using the menu item **File→Export→Export Hardware for SDK...** to open the Export Hardware options.

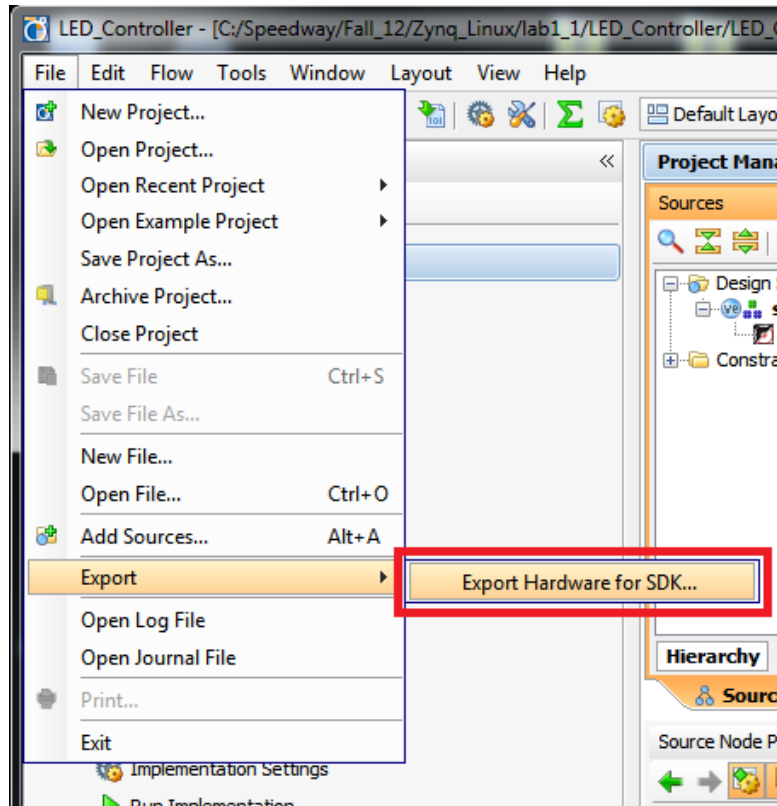


Figure 5 – Exporting the Embedded Subsystem Hardware for SDK

6. Use the default selection for each of the options except the **Launch SDK** option. Verify that the **Launch SDK** option is selected as seen in Figure 6.

Click the **OK** button and wait for SDK to launch.

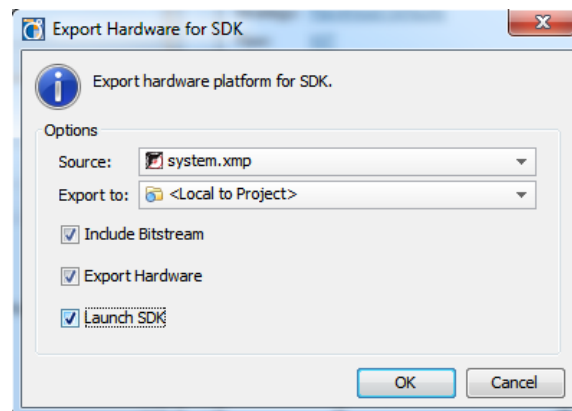


Figure 6 – Export Options

7. Once SDK has launched. Familiarize yourself with the SDK environment as we will revisit this environment in later labs. The **system.xml** file should be open in the main editor pane. The **system.xml** contains the memory map and associated IP blocks for each of the hardware peripherals that were connected to the processing system in XPS. To open this file at other times, double-click on **system.xml** in Project Explorer pane under the **system_hw_platform** project.

Notice that the **system_hw_platform** project also contains a bitstream (.bit) file as well as ps7_init.c, ps7_init.h, ps7_init.tcl, and ps7_init.html files. These files contain information critical to the Processing System initialization and were automatically generated based upon the XPS Processing System settings during the Hardware Export process.

Notice also that the software projects and BSP from the Zynq Introduction SpeedWay activities have been removed from the workspace beforehand since they are no longer needed for the Zynq Linux SpeedWay activities.

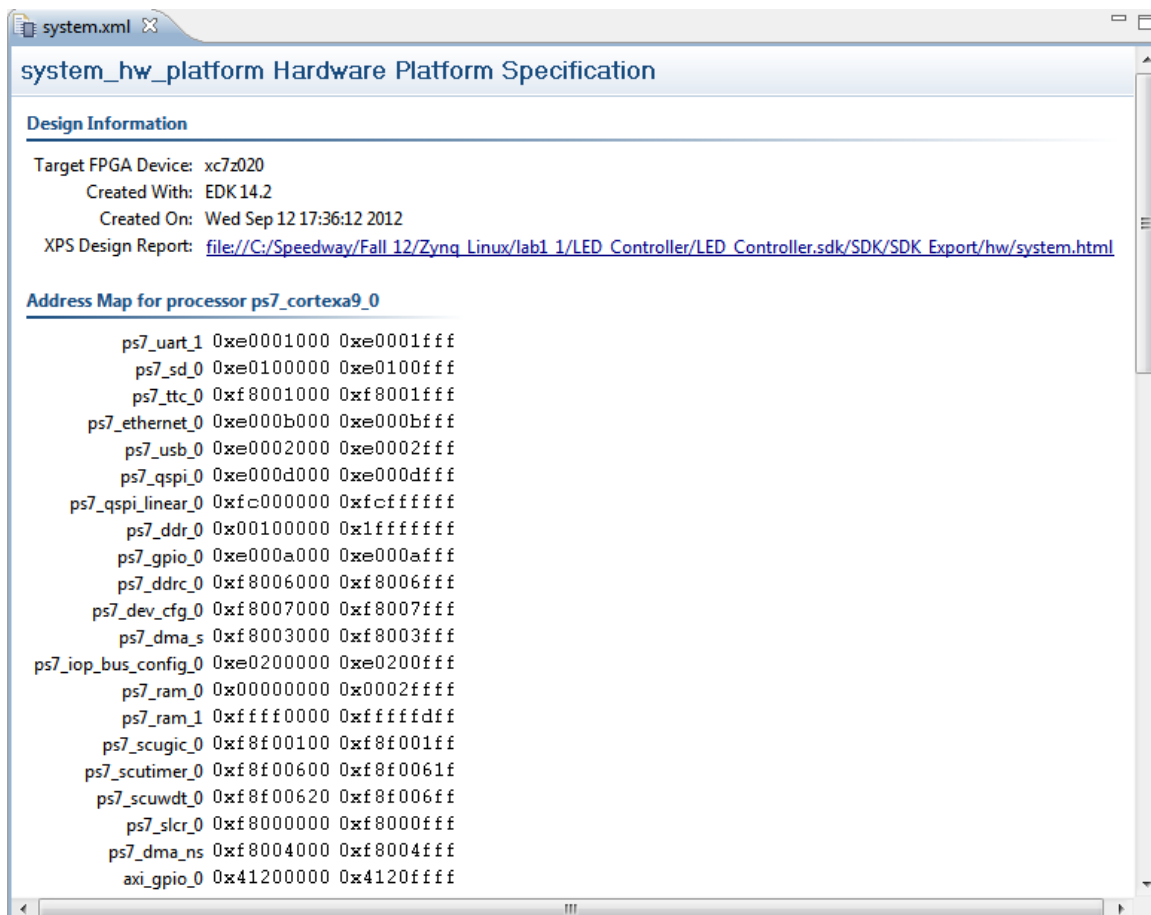


Figure 7 – The system.xml File Shown in SDK Main Editor Pane

Questions:

Answer the following questions:

- *What are the 3 required file sets that define a hardware platform?*

- *Where did the hardware platform in Experiment 1 come from and what tool/license was required to create it?*

- *Does the ZedBoard Kit include the license to create or modify hardware platforms?*

Experiment 2: Create the FSBL Project and Board Support Package

SDK is now ready for software development. In this experiment, a FSBL application project and Board Support Package is added to the workspace, the projects are built, and the executable is copied to the Lab 1.3 folder for later use.

Experiment 2 General Instruction:

Add the FSBL project and Board Support Package then copy the executable files to the Lab 1.3 folder.

Experiment 2 Step-by-Step Instructions:

1. In SDK, select the **File → New → Xilinx C Project** menu selection to launch the **New Xilinx C Project** wizard.

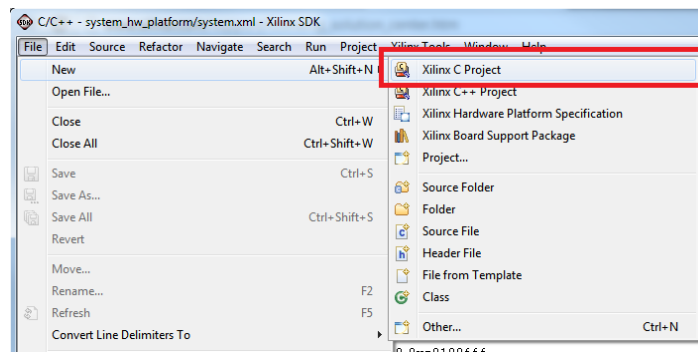


Figure 8 – Launching the New Xilinx C Project Wizard

2. Select **Zynq FSBL** from the **Select Project Template** field. This will automatically change the default project name to **zynq_fsbl_0**, which is acceptable.

Click the **Next** button.

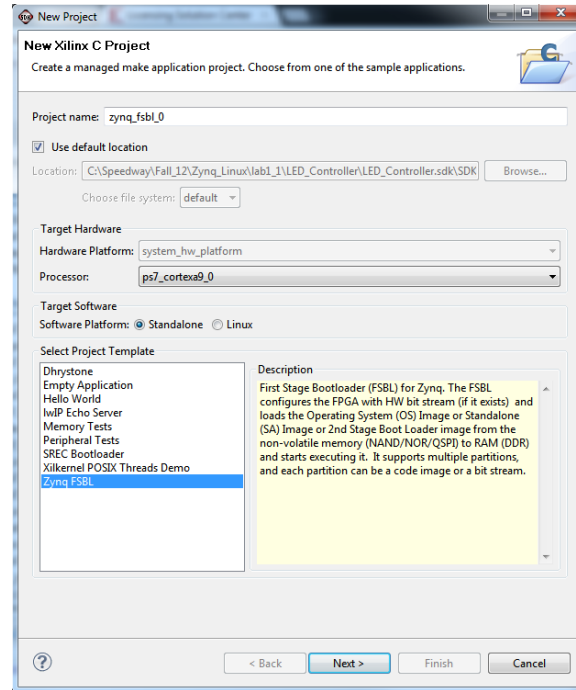


Figure 9 – Selecting the Zynq FSBL Project Template

3. Since there is not already a BSP in the project, the radio button for the **Create a new Board Support Package project** option is already selected. Even if there were a BSP already created in the workspace, this window to create a new Xilinx C project will still default to creating a new BSP specifically for the new FSBL software project.

The new BSP name **zynq_fsbl_bsp_0** is automatically suggested.

Click the **Finish** button.

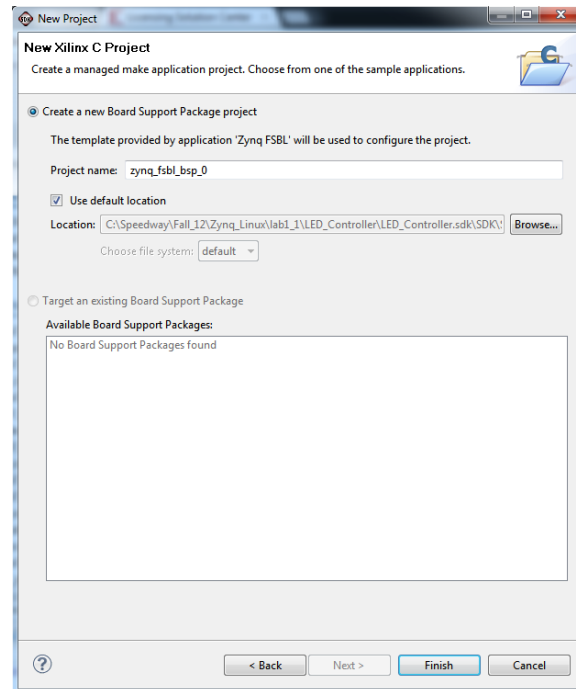


Figure 10 – Creating the BSP for the FSBL

4. The FSBL and BSP projects are now created within the workspace and automatically built by the compiler. By default, SDK will build the application project automatically after it is added or if any source files are modified.

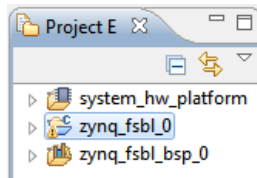


Figure 11 – FSBL Project Listing in SDK Project Explorer Pane

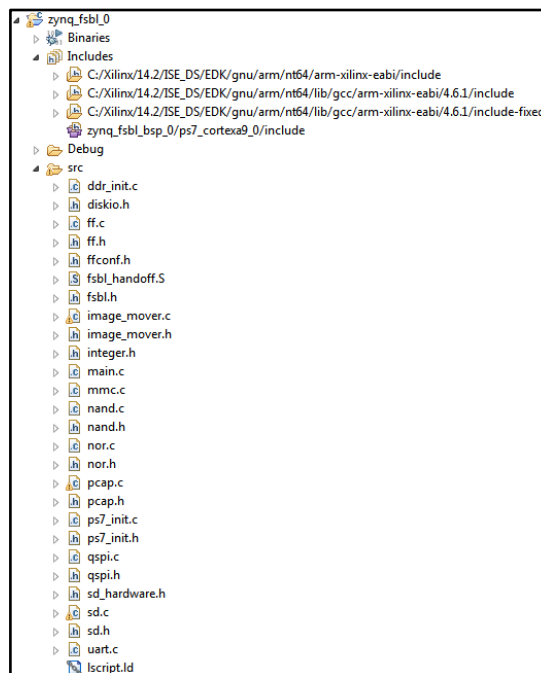
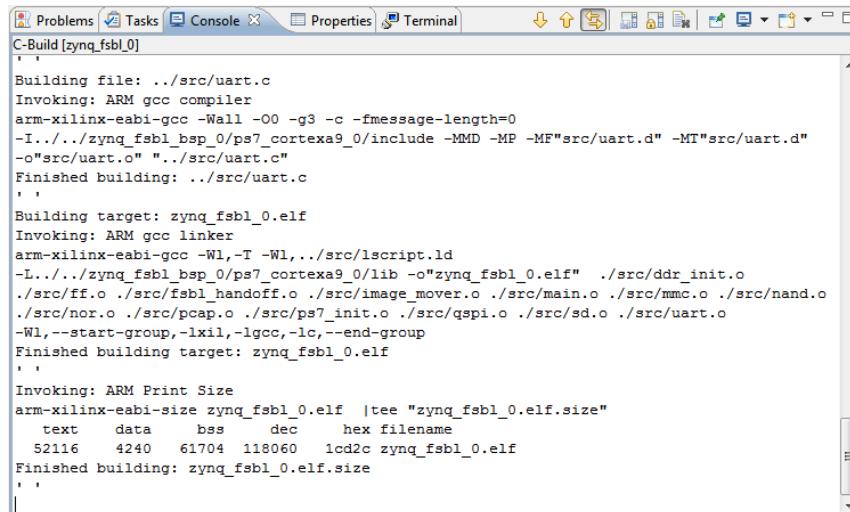


Figure 12 – zynq_fsbl_0 Application Expanded



```
C-Build [zynq_fsbl_0]

Building file: ../src/uart.c
Invoking: ARM gcc compiler
arm-xilinx-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0
-I../zynq_fsbl_bsp_0/ps7_cortexa9_0/include -MMD -MP -MF"src/uart.d" -MT"src/uart.d"
-o"src/uart.o" "../src/uart.c"
Finished building: ../src/uart.c

Building target: zynq_fsbl_0.elf
Invoking: ARM gcc linker
arm-xilinx-eabi-gcc -Wl,-T ../src/ldscript.ld
-L../zynq_fsbl_bsp_0/ps7_cortexa9_0/lib -o"zynq_fsbl_0.elf" ./src/ddr_init.o
./src/ff.o ./src/fsbl_handoff.o ./src/image_mover.o ./src/main.o ./src/mmc.o ./src/nand.o
./src/nor.o ./src/pcap.o ./src/ps7_init.o ./src/qspi.o ./src/sd.o ./src/uart.o
-Wl,--start-group,-lxil,-lgcc,-lc,--end-group
Finished building target: zynq_fsbl_0.elf

Invoking: ARM Print Size
arm-xilinx-eabi-size zynq_fsbl_0.elf |tee "zynq_fsbl_0.elf.size"
text    data    bss     dec      hex filename
52116    4240    61704   118060   1cd2c zynq_fsbl_0.elf
Finished building: zynq_fsbl_0.elf.size
```

Figure 13 – FSBL Application Automatically Built

5. Using the Project Explorer pane to the left of the screen, expand the **zynq_fsbl_0** project and the **src** sub-folder. Locate the linker script file **ldscript.ld** in the **src** sub-folder and open it by double clicking on the file entry.

Notice that in the Hardware Memory Map, two memories are listed. The **ps7_ram_0_S_AXI_BASEADDR** is the ARM local OCM memory location following a reset. This memory runs at the processor speed and is, therefore, one of the fastest types of memory available to the ARM Cortex-A9 cores.

The second memory segment is listed as **ps7_ram_1_S_AXI_BASEADDR**. This is part of the same ARM local OCM.

According to Xilinx UG585, V1.2, section 6.3.7: (upon BootROM exit) *192 KB of OCM is accessible starting at address 0x0 while 64 KB is accessible starting at address 0xFFFF0000*. Therefore the application linker script must map these same segments accordingly to match the OCM upper and lower partitions.

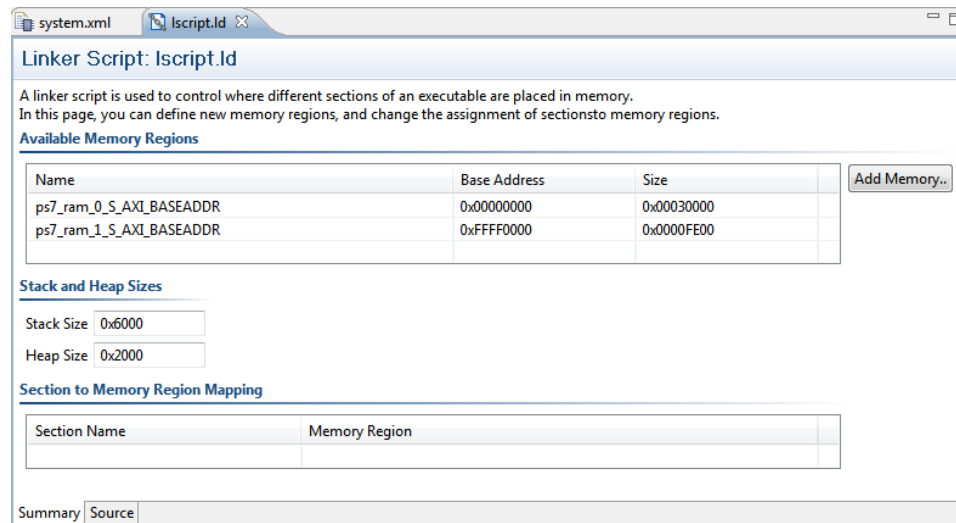


Figure 14 – Linker Script Definitions

- Now that the FSBL has been built, locate the executable file **zynq_fsbl_0.elf** in a Windows Explorer session. This file can be found in the following folder:

C:\Speedway\Fall_12\Zynq_Linux\lab1_1\LED_Controller\LED_Controller.sdk\SDK\SDK_Export\zynq_fsbl_0\Debug

This file is needed for a later lab so copy the file to the Lab 1.3 folder located in the following path:

**C:\Speedway\Fall_12\Zynq_Linux\lab1_3 **

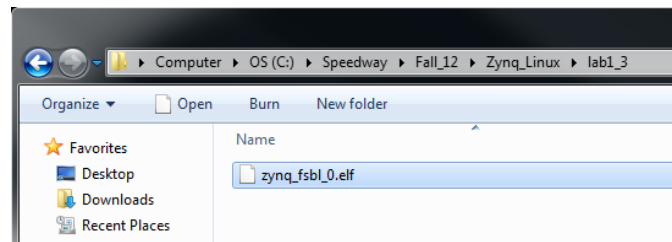


Figure 15 – FSBL Executable Copied to the Lab 1.3 Folder

- Locate the hardware bitstream file **system.bit** in a Windows Explorer session. This file can be found in the following folder:

C:\Speedway\Fall_12\Zynq_Linux\lab1_1\LED_Controller\LED_Controller.sdk\SDK\SDK_Export\system_hw_platform

This file is needed for a later lab so copy the file to the Lab 1.3 folder located in the following path:

C:\Speedway\Fall_12\Zynq_Linux\lab1_3

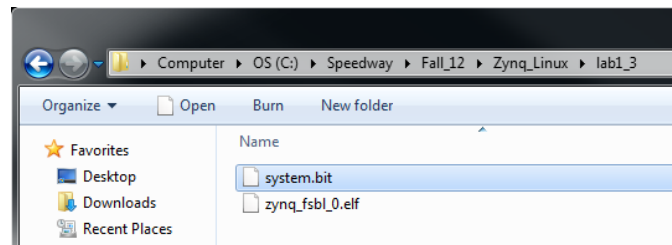


Figure 16 – Hardware Bitstream Copied to the Lab 1.3 Folder

Questions:

Answer the following questions:

- *From where did the code for the FSBL come from?*

- *Which address space is used when the FSBL first begins execution?*

Exploring Further

If you have additional time and would like to investigate more...

- Explore the Processing System initialization files (ps7_init.c, ps7_init.h, ps7_init.tcl, and ps7_init.html) and see what settings for the Processing System are inherited from the XPS hardware project.
- Read the *Tips & Tricks* available under **Help**.

This concludes Lab 1.1.

Revision History

Date	Version	Revision
07 Sep 12	00	Initial Draft
28 Sep 12	01	Revised Draft
18 Oct 12	02	Course Release
14 Jan 13	05	ZedBoard.org Training Course Release

Resources

<http://www.zedboard.org>

<http://www.xilinx.com/zyng>

<http://www.xilinx.com/planahead>

<http://www.xilinx.com/sdk>

Answers

Experiment 1

- *What are the 3 required file sets that define a hardware platform?*

The hardware description file (system.xml), hardware bitstream (.bit), and Processing System initialization files (ps7_init.c, ps7_init.h, ps7_init.tcl, and ps7_init.html)

- *Where did the hardware platform in Experiment 1 come from and what tool/license was required to create it?*

The Experiment 1 hardware platform was delivered with this training. It was previously designed and built using Xilinx Platform Studio (XPS) during the Introduction to Zynq SpeedWay training session.

Zynq hardware platform design requires XPS, which is part of the complete Xilinx Embedded Development Kit (EDK).

- *Does the ZedBoard Kit include the license to create or modify hardware platforms?*

The voucher included with the ZedBoard Kit will generate a WebPACK/ChipScope license via the Xilinx Licensing Solutions Center.

Experiment 2

- *From where did the code for the FSBL come from?*

The code is provided as an example within the SDK environment. SDK generates the code based on the memory controllers found within the hardware platform.

- *Which address space is used when the FSBL first begins execution?*

ps7_ram_0_S_AXI_BASEADDR 0x00000000 0x00030000