

Implementing Linux on the Zynq™-7000 SoC

Lab 3.2

Device Drivers and PL Devices on AXI-Lite Interconnects



September 2012
Version 05

Table of Contents

Table of Contents	2
Lab 3.2 Overview	3
Lab 3.2 Objectives	3
Experiment 1: Implement the Custom Device Driver	4
<i>Questions:</i>	12
Experiment 2: Modify the Device Tree Source File	13
<i>Questions:</i>	19
Experiment 3: Test the Custom Device Driver	20
<i>Questions:</i>	26
Exploring Further	27
Revision History	27
Resources	27
Answers	28
Experiment 1	28
Experiment 2	28
Experiment 3	28

Lab 3.2 Overview

This lab builds upon the skills covered in the previous labs.

This lab will demonstrate traditional driver implementation with the purpose of controlling a Programmable Logic (PL) device that has been added to the AXI-Lite interconnect.

This lab will also illustrate the role of the device tree and its importance in validating device drivers.

Lab 3.2 Objectives

When you have completed Lab 3.2, you will know how to do the following:

- Implement a driver to access PL hardware connected to AXI-Lite interconnects
- Modify a device tree source file to include hardware resources for the custom AXI-Interconnect system device
- Compile the modified device tree source file to obtain a flattened device tree which can be used to test the custom driver
- Dynamically insert the driver into the Linux kernel

Experiment 1: Implement the Custom Device Driver

This experiment shows how to implement the custom driver for the LED brightness controller that exists in the Programmable Logic design. Two design pieces are needed for our custom driver to take advantage of this hardware device. The first design piece needed is the device driver module and the second piece is an updated device tree file which describes the Programmable Logic hardware resource.

Device Driver Module

The custom device driver will be delivered as a patch to the kernel. Once the patch is applied, the kernel build process will be followed again to obtain a kernel driver module or kernel object. This driver module (recognizable with a file extension of *.ko) can later be loaded into the kernel dynamically at runtime using the insmod command.

Device Tree

The device tree is used to specify where devices are located on the AXI interconnects and which resources are attached to those devices. This information is used by the kernel to load and enumerate the necessary drivers for the devices found in the device tree.

Experiment 1 General Instruction:

Implement the custom driver by patching the kernel and rebuilding to obtain the custom driver module.

Experiment 1 Step-by-Step Instructions:

1. If the CentOS virtual machine is not already open, launch the VMware Player application from by selecting **Start → All Programs → VMware → VMware Player**. If the CentOS virtual machine is already open, skip ahead to Step 4.



Figure 1 – The VMware Player Application Icon

2. Select the virtual machine named **CentOS-6.3-amd64-ZedBoard-linux** from the selections on the left and then click on the **Play virtual machine** button to the right.

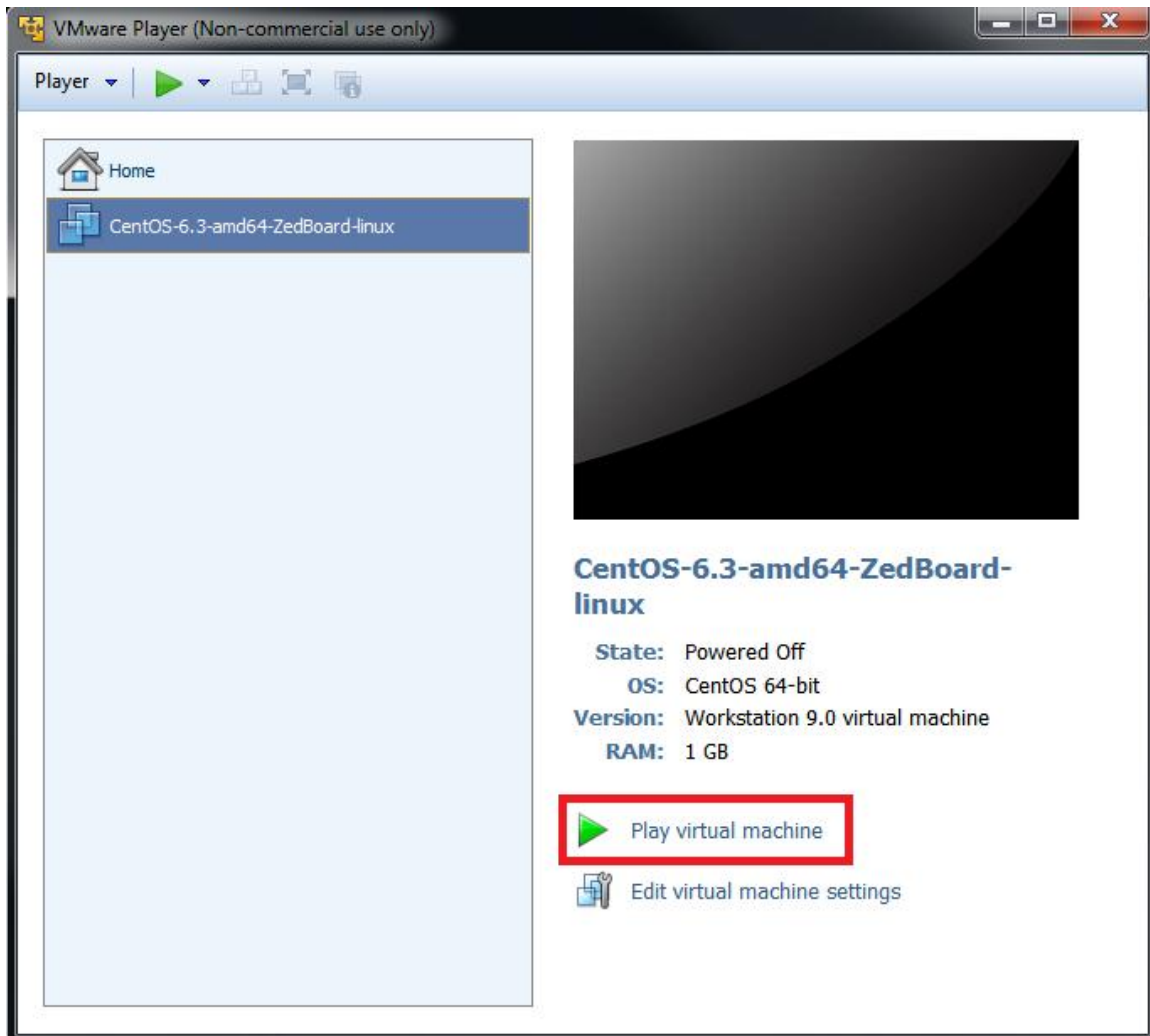


Figure 2 – The VMware Player Application

3. If prompted for a workstation login, click on the user entry **training** and enter the password **Avnet** in order to log into the system.

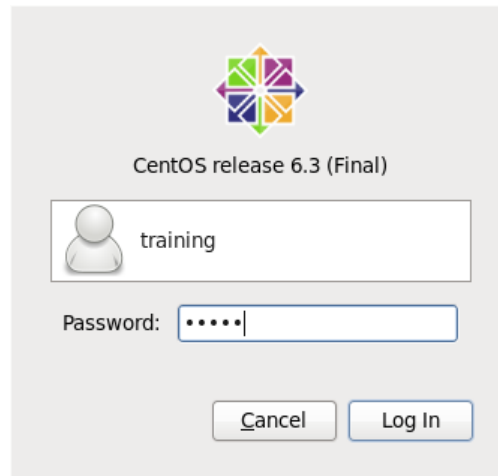


Figure 3 – The CentOS Workstation Login

4. If a terminal is not already open on the desktop of the CentOS guest operating system, open a terminal window through the **Applications→System Tools→Terminal** menu item. If a terminal is already open, bring that terminal session into focus on the desktop.

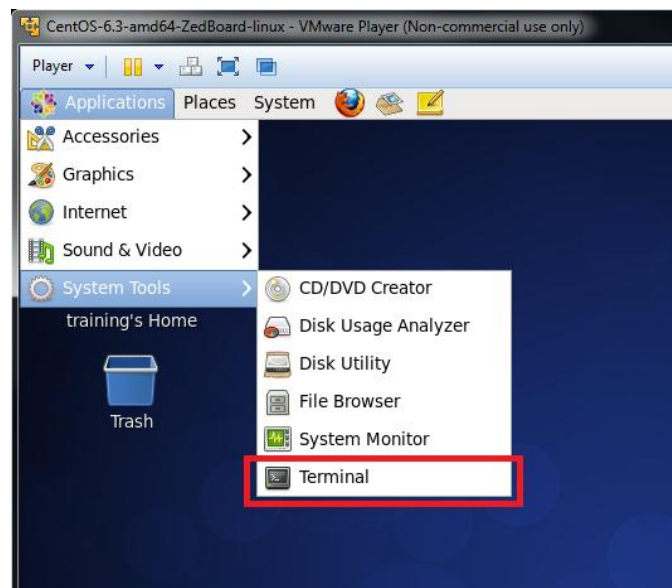


Figure 4 – Launching the CentOS Terminal from the Desktop

5. Change from the home directory into the Xilinx Linux kernel source directory.

```
$ cd ~/linux-xlnx/
```

6. Locate the **.patch** file in the Lab 3.2 folder on the host operating system by using Windows explorer to navigate to the following folder:

C:\Speedway\Fall_12\Zynq_Linux\lab3_2

Right click on this file and select the **Copy** option which will place the selected file in the Virtual Machine clipboard.

This patch includes the custom device driver code which is a simplified way of delivering a code change to the Linux kernel source tree or any code tree for that matter. The driver code contained in this patch is needed for the following labs 4.1, and 4.2 to be completed successfully.

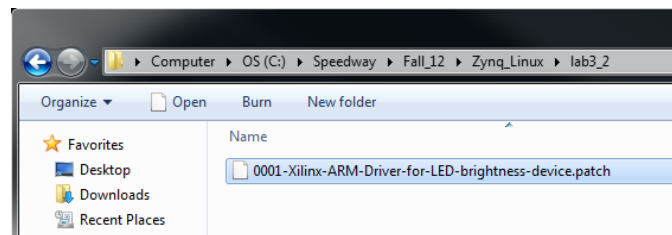


Figure 5 – Kernel Device Driver Patch File

7. If a CentOS file browser window is not already open from a previous exercise, open a file browser window through the **Applications→System Tools→File Browser** menu item.

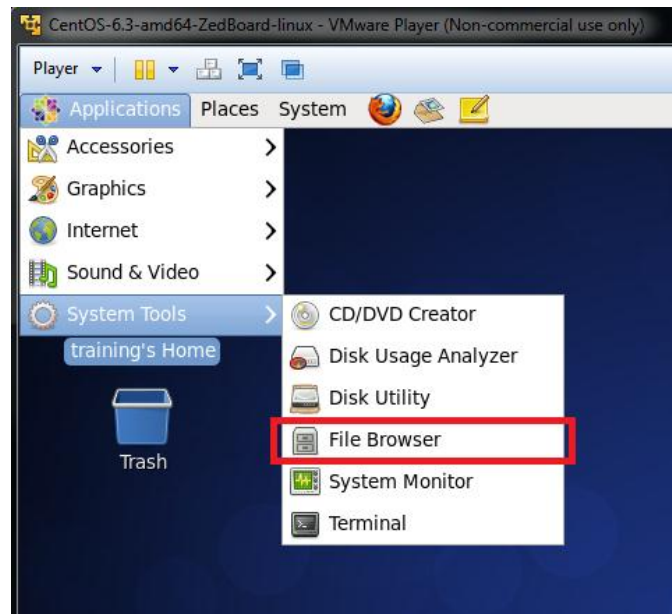


Figure 6 – CentOS File Browser

8. Using the CentOS File Browser on the guest operating system, right click in open white space between folder icons and select the **Paste** option to paste the patch file into the Linux kernel source code tree **/home/training/linux-xlnx/** folder.

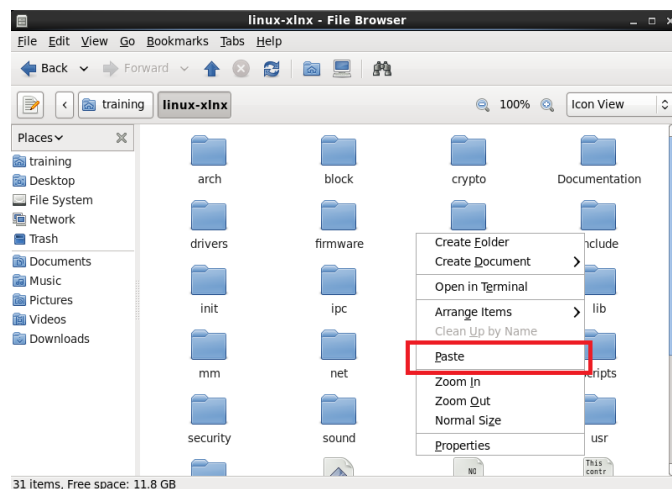


Figure 7 – Kernel Patch Pasted to the Guest Machine

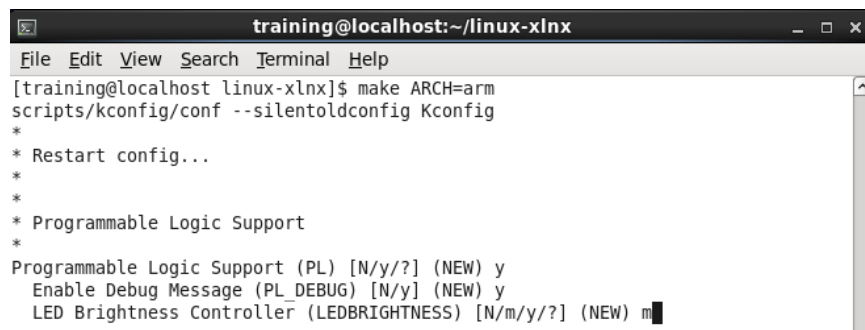
9. Using the CentOS terminal, apply the patch to the kernel tree.

```
$ git apply 0001-Xilinx-ARM-Driver-for-LED-brightness-device.patch
```

10. Build the kernel source with the make command to create the kernel driver module target.

```
$ make ARCH=arm
```

If prompted for additional configuration options for Programmable Logic Support and Debug Message simply enter the character **y** and press **<Enter>** for each option. For the LED Brightness controller, enter the character **m** and press **<Enter>** to build the driver as a module.



```
training@localhost:~/linux-xlnx
File Edit View Search Terminal Help
[training@localhost linux-xlnx]$ make ARCH=arm
scripts/kconfig/conf --silentoldconfig Kconfig
*
* Restart config...
*
* Programmable Logic Support
*
Programmable Logic Support (PL) [N/y/?] (NEW) y
Enable Debug Message (PL_DEBUG) [N/y] (NEW) y
LED Brightness Controller (LEDBRIGHTNESS) [N/m/y/?] (NEW) m
```

Figure 8 – Linux Configuration for New Device Driver

If you mistakenly enter the wrong options when prompted above and the led-brightness.ko module is not built, simply open the kernel configuration file with gedit text editor.

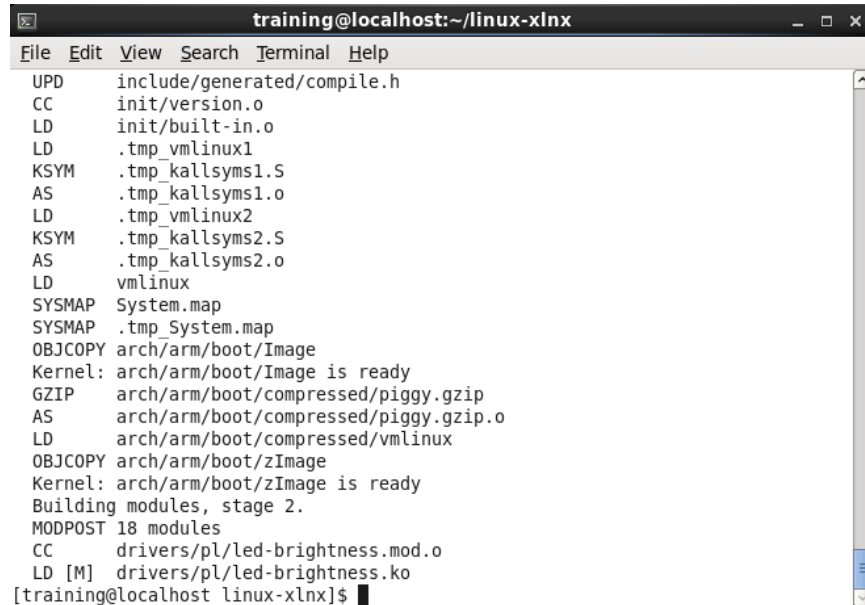
```
$ gedit .config
```

Then check to see that the following entries exist and are assigned to the expected settings.

```
CONFIG_PL=y
CONFIG_PL_DEBUG=y
CONFIG_LEDBRIGHTNESS=m
```

Save the any changes to the configuration file before exiting gedit. Try to build again by repeating the **make** command from above.

Since the kernel build process has already been performed previously as part of Lab 2.1, this time the build process should only take about 1-2 minutes to complete and should end up looking similar to Figure 9.



```
training@localhost:~/linux-xlnx
File Edit View Search Terminal Help
UPD    include/generated/compile.h
CC     init/version.o
LD     init/built-in.o
LD     .tmp_vmlinux1
KSYM   .tmp_kallsyms1.S
AS     .tmp_kallsyms1.o
LD     .tmp_vmlinux2
KSYM   .tmp_kallsyms2.S
AS     .tmp_kallsyms2.o
LD     vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
GZIP   arch/arm/boot/compressed/piggy.gz
AS     arch/arm/boot/compressed/piggy.o
LD     arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
Building modules, stage 2.
MODPOST 18 modules
CC     drivers/pl/led-brightness.mod.o
LD [M] drivers/pl/led-brightness.ko
[training@localhost linux-xlnx]$
```

Figure 9 – Linux Kernel Build Completed

11. Now that the driver module has been built, it must be copied to the Windows host machine so that it can later be transferred to the SD card in a later experiment.

Using the CentOS File Browser, locate the file **/home/training/linux-xlnx/drivers/pl/led-brightness.ko** which is the target driver module needed for the next lab experiment. Right click on this file and select the **Copy** option which will place the selected file in the Virtual Machine clipboard.

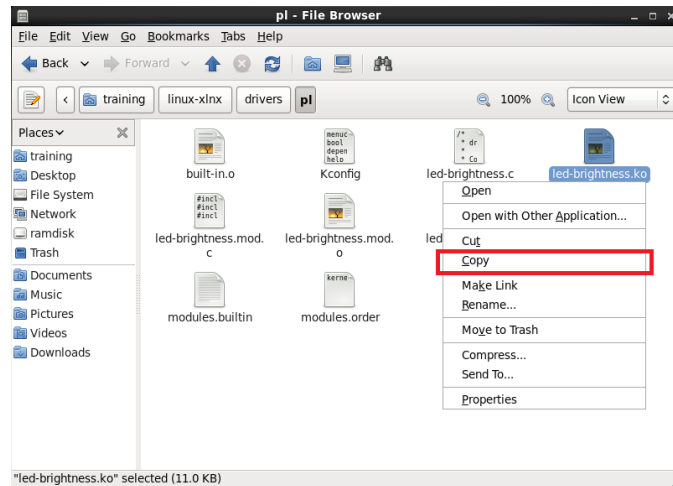


Figure 10 – Copying Kernel Image to Virtual Machine Clipboard

12. Paste the **led-brightness.ko** driver module into the Lab 3.2 folder under the host operating system by using Windows Explorer to navigate to the following folder:

C:\Speedway\Fall_12\Zynq_Linux\lab3_2

You may have noticed some other files and folders in the Lab 3.2 folder. The **driver_test_bench** folder will later be used to verify that the driver is functioning properly and affects the behavior of the hardware device as expected. Ignore the **.dts** file for now, this file will be used in the next experiment.

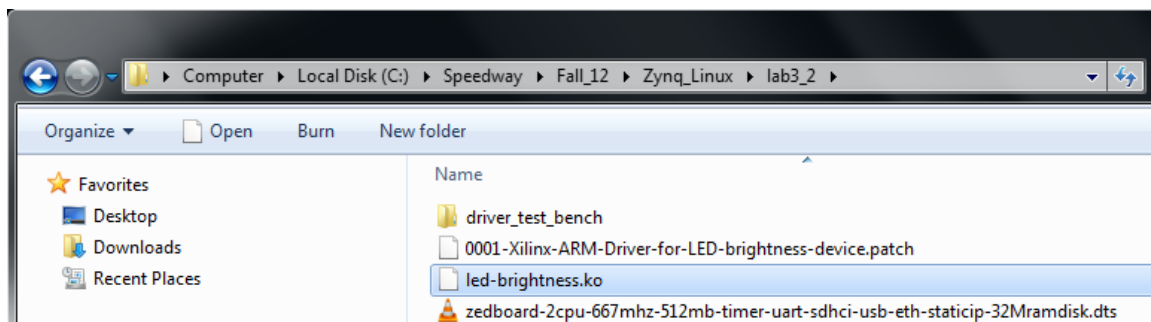


Figure 11 – Device Driver Module Copied to the Windows Host Machine

Questions:

Answer the following questions:

- *What is the purpose of the configuration update that is performed in Step 10?*

Experiment 2: Modify the Device Tree Source File

This experiment shows how to fulfill the implementation of the custom driver for the LED brightness controller that exists in the Programmable Logic design. In the lecture, we mentioned that two design pieces are needed for our custom driver to take advantage of this hardware device. The first design piece needed is the device driver module and the second piece is an updated device tree file which describes the Programmable Logic hardware resource. Although the device driver was properly implemented in Experiment 1, the device tree has not been modified to accommodate the device we are attempting to add. As a result, we will not see the device driver behavior we were hoping for until these device tree changes are made.

In this exercise, the device tree will be used to specify where the LED brightness controller is located on the AXI interconnects and which resources are attached to that device. The new information in the device tree will then be used by the kernel to load the necessary driver properties for the new device found in the device tree.

Experiment 2 General Instruction:

Modify the device tree to add the needed resources for the LED brightness controller which exists on an AXI interconnect in the Programmable Logic fabric.

Experiment 2 Step-by-Step Instructions:

1. Change from the home directory into the Xilinx Linux kernel source directory.

```
$ cd ~/linux-xlnx/
```

2. Locate the **.dts** file in the Lab 3.2 folder on the Windows host operating system by using Windows Explorer to navigate to the following folder:

C:\Speedway\Fall_12\Zynq_Linux\lab3_2

Right click on this file and select the **Copy** option which will place the selected file in the Virtual Machine clipboard.

This device tree source file includes the basic system configuration that we have been using to boot Linux throughout all the previous exercises. This device tree needs to be modified for the LED brightness driver in order for the following labs 4.1 and 4.2 to be completed successfully.

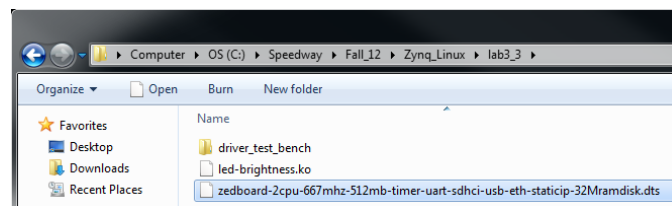


Figure 12 – Device Tree Source File

3. Create a copy of the device tree source file on the Linux guest virtual machine.

Using the CentOS File Browser on the guest operating system, right click in open white space between folder icons and select the **Paste** option to paste the file into kernel source **/home/training/linux-xlnx/** folder.

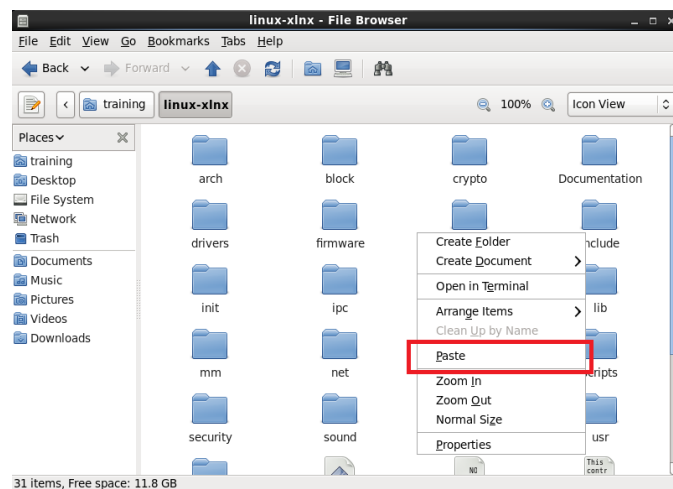


Figure 13 – Device Tree Source Pasted to the Guest Machine

4. Using the CentOS terminal, move the device tree source file to a more manageable file named **devicetree.dts** within the same folder.

```
$ mv zedboard-2cpu-667mhz-512mb-timer-uart-sdhci-usb-\  
eth-staticip-32Mramdisk.dts devicetree.dts
```

5. Modify the **devicetree.dts** file by opening the file with the gedit text editor. This is easily done from the command line using the following command line:

```
$ gedit devicetree.dts
```

Copy the following lines from this document into the VM clipboard by right-clicking in Adobe Reader and selecting the **Copy** option:

```
led-brightness@41200000 {  
    compatible = "avnet,led-brightness";  
    reg = <0x41200000 0x20>;  
};
```

Paste contents from the VM clipboard near the end of the **devicetree.dts** file by right-clicking in the gedit editor window and selecting the **Paste** option. Format the text using tabs on the left side so that it looks similar to Figure 14.

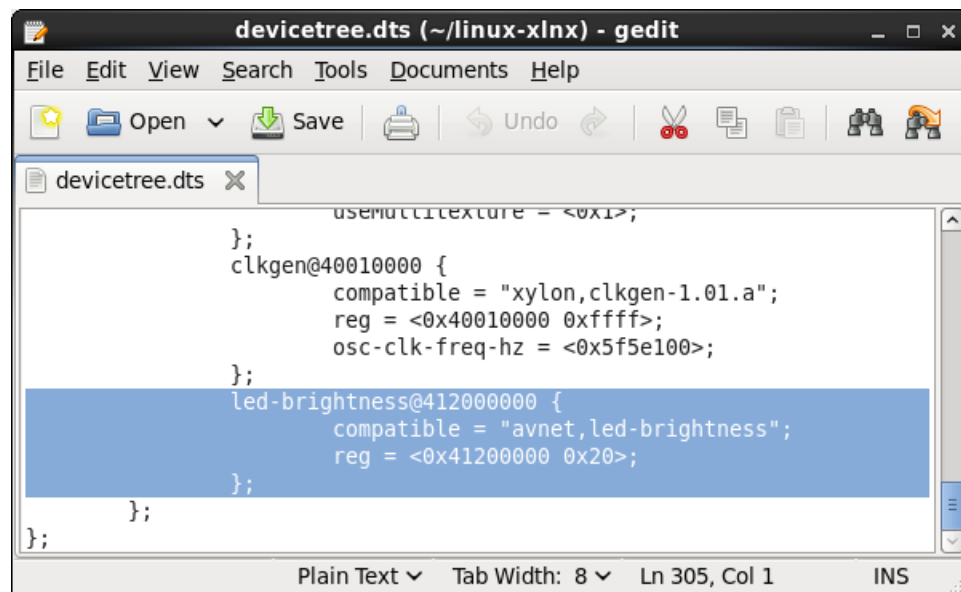


Figure 14 – Using gedit Text Editor to Modify the devicetree.dts File

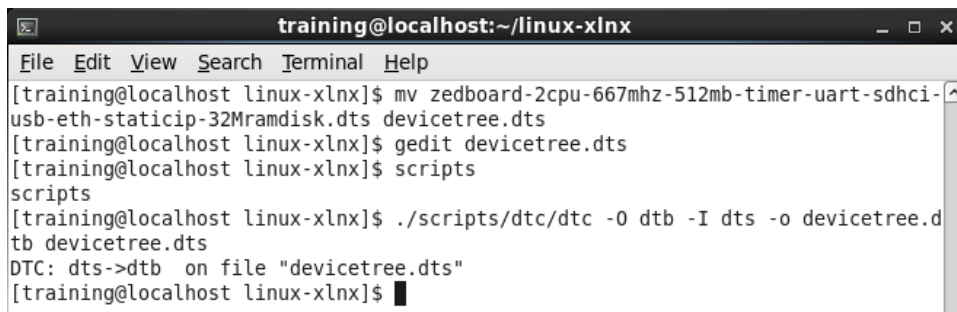
The new device tree entry specifies a **led-brightness** device attached to the AXI interconnect at memory address 0x41200000. This is the memory address reported by the system.xml file within SDK (0x41200000) which represents the first channel of the **axi_gpio** device. Refer to Lab 1.1 for information on how to open this file for further inspection.

Exit the gedit text editor using the **File→Quit** menu option and save changes to the **devicetree.dts** file.

6. Use the Linux kernel device tree compiler **dtc** to convert the source file into a binary representation that the kernel is able to interpret at boot time.

```
$ ./scripts/dtc/dtc -O dtb -I dts -o devicetree.dtb \
devicetree.dts
```

Here the device tree compiler uses the **-O** argument to specify the output format as the device tree binary and the **-I** argument to specify the input format as the device tree source. The output filename is specified using the **-o** argument and the input filename is passed in as a plain command argument.



```
training@localhost:~/linux-xlnx
File Edit View Search Terminal Help
[training@localhost linux-xlnx]$ mv zedboard-2cpu-667mhz-512mb-timer-uart-sdhci-
usb-eth-staticip-32Mramdisk.dts devicetree.dts
[training@localhost linux-xlnx]$ gedit devicetree.dts
[training@localhost linux-xlnx]$ scripts
scripts
[training@localhost linux-xlnx]$ ./scripts/dtc/dtc -O dtb -I dts -o devicetree.d
tb devicetree.dts
DTC: dts->dtb on file "devicetree.dts"
[training@localhost linux-xlnx]$
```

Figure 15 – Using the Linux Device Tree Compiler Script

- Now that the device tree has been modified and compiled to binary format, it must be copied to the Windows host machine so that it can later be transferred to the SD card in a later experiment.

Using the CentOS file browser, locate the file **/home/training/linux-xlnx/devicetree.dtb** which is the binary representation of the device tree needed for the next experiment. Right click on this file and select the **Copy** option which will place the selected file in the Virtual Machine clipboard.

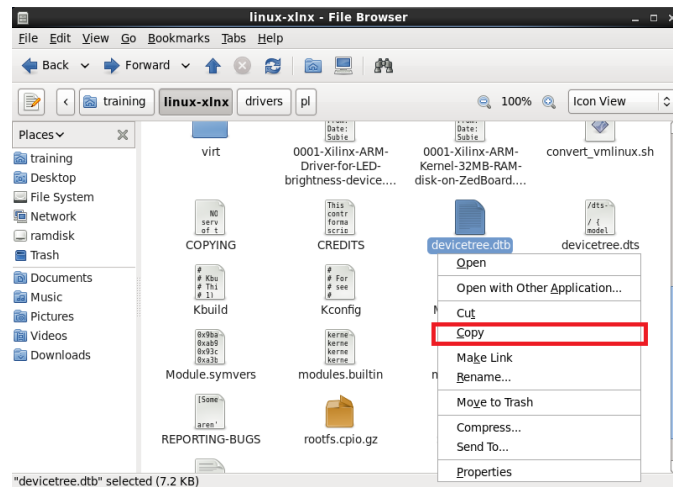


Figure 16 – Copying the devicetree.dtb File to Virtual Machine Clipboard

- Paste the **devicetree.dtb** file into the Lab3.2 folder under the host operating system by using Windows Explorer to navigate to the following folder:

C:\Speedway\Fall_12\Zynq_Linux\lab3_2

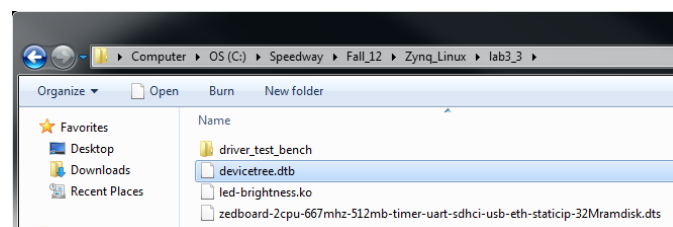


Figure 17 – Device Tree Binary File Copied to the Host Machine

9. Power down ZedBoard using power switch SW8 and remove the SD card from J12.

Insert the SD card into the PC or SD card reader and wait for it to enumerate as a Windows drive. If prompted by Windows when inserting the SD card, select the **Continue without scanning** option.

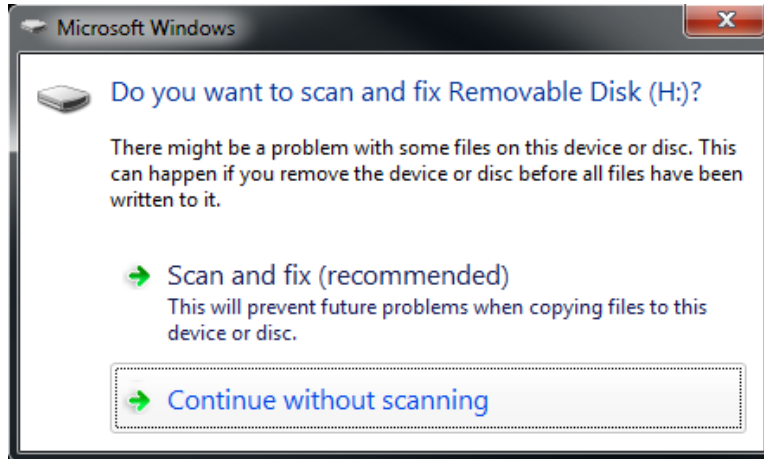


Figure 18 – Windows Prompt for Scanning and Fixing an SD Card

10. Copy the **led-brightness.ko** and **devicetree.dtb** files as well as the **driver_test_bench** folder from the Lab 3.2 folder to the top level of the SD card. Replace any existing versions of the **led-brightness.ko** and **devicetree.dtb** files as well as the **driver_test_bench** folder that may be on the SD card.

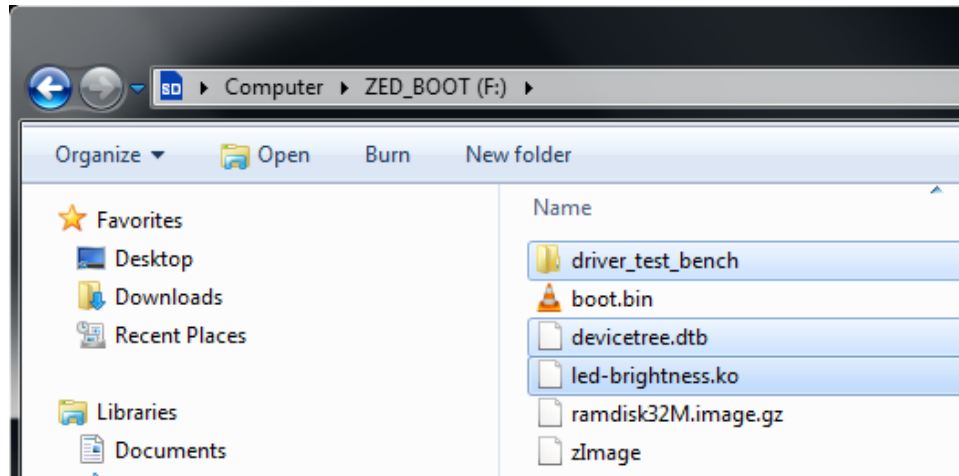


Figure 19 – Driver Module, Device Tree, and Test Bench Copied to the SD Card

Questions:

Answer the following questions:

- Why must the device tree source file be compiled to a binary format?

Experiment 3: Test the Custom Device Driver

ZedBoard can now be booted and the custom device driver inserted into the kernel using the updated device tree binary. We will look for the driver debug statements to be output to the console.

Experiment 3 General Instruction:

Boot ZedBoard using the SD card and insert the custom device driver into the kernel for testing using the updated device tree binary.

Experiment 3 Step-by-Step Instructions:

1. Connect 12 V power supply to ZedBoard barrel jack (J20).

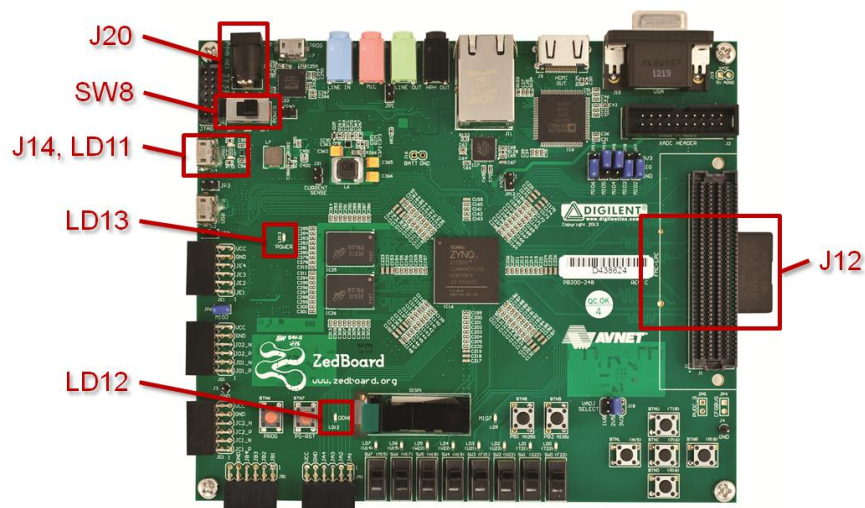


Figure 20 – ZedBoard Hardware Reference

2. Connect the USB-UART port of ZedBoard (J14) which is labeled UART to a PC using the MicroUSB cable.
3. Insert the 4GB SD card included with ZedBoard into the SD card slot (J12) located on the underside of ZedBoard PCB.

4. Verify the ZedBoard boot mode (JP7-JP11) and MIO0 (JP6) jumpers are set to SD card mode as described in the Hardware Users Guide:

http://www.zedboard.org/sites/default/files/ZedBoard_HW_UG_v1_6.pdf

A copy of the Hardware Users Guide is also located in the Speedway
C:\Speedway\Fall_12\Zynq_Linux\support_documents folder for your convenience.

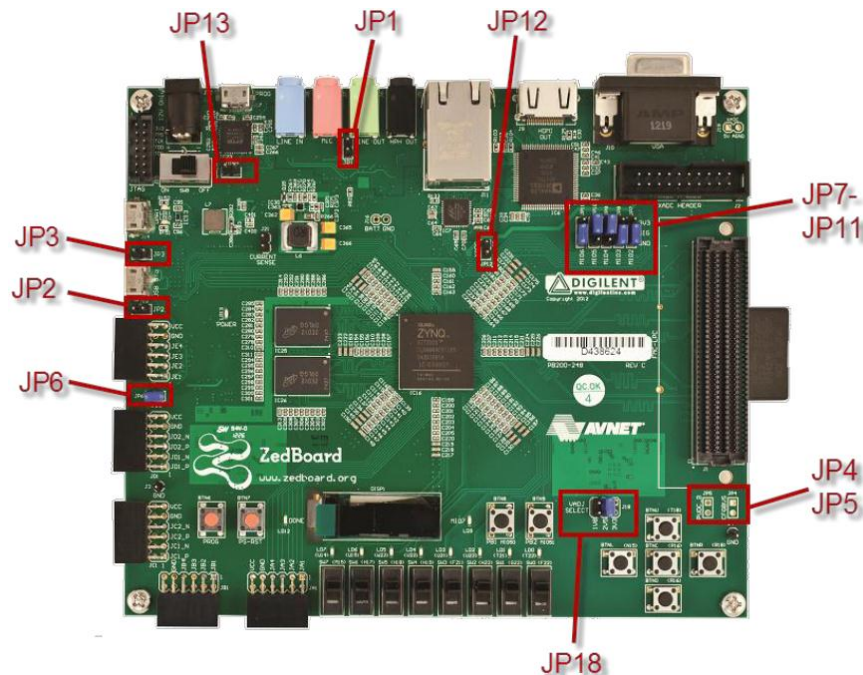


Figure 21 – ZedBoard Jumper Settings

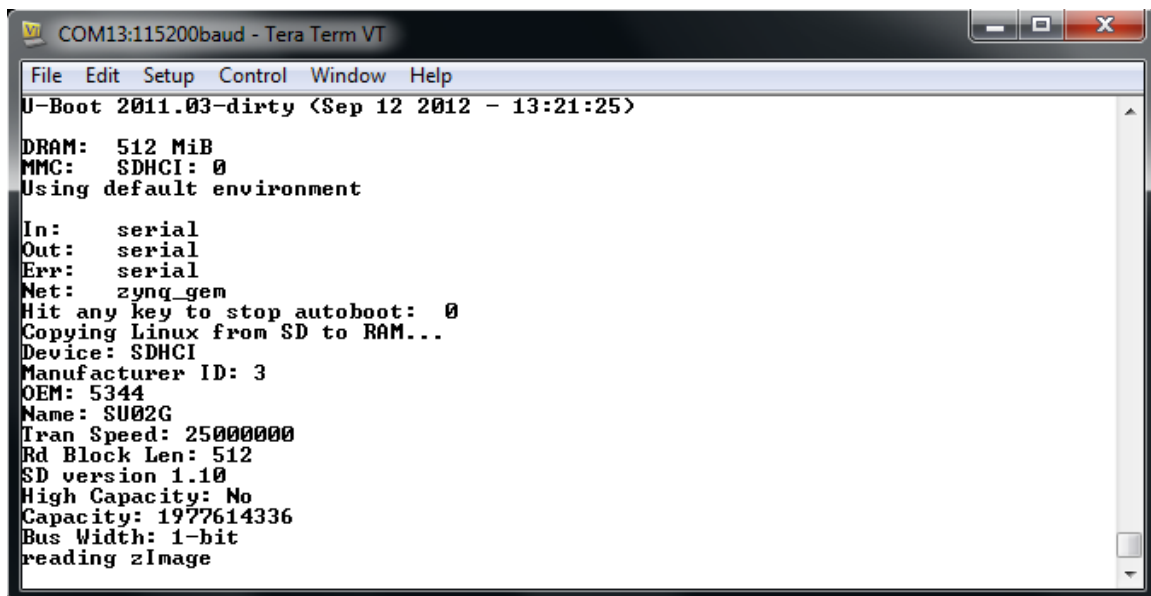
5. Turn power switch (SW8) to the ON position. ZedBoard will power on and the Green Power Good LED (LD13) should illuminate.

6. Wait approximately 15 seconds. The blue Done LED (LD12) should illuminate.
7. On the PC, if a serial terminal session is not already open, open a serial terminal program. Tera Term was used to show the example output for this lab document.



Figure 22 – Tera Term Icon

8. If the amber USB-Link Status (LD11) does not flicker during boot to indicate activity, check the driver installation to determine if the device driver is recognized and enumerated successfully and that there are no errors reported by Windows.
9. Power cycle the ZedBoard and monitor the Tera Term window. When the terminal output from U-Boot and a countdown is observed, allow the countdown to expire.



```
COM13:115200baud - Tera Term VT
File Edit Setup Control Window Help
U-Boot 2011.03-dirty (Sep 12 2012 - 13:21:25)

DRAM: 512 MiB
MMC: SDHCI: 0
Using default environment

In: serial
Out: serial
Err: serial
Net: zynq_gem
Hit any key to stop autoboot: 0
Copying Linux from SD to RAM...
Device: SDHCI
Manufacturer ID: 3
OEM: 5344
Name: SU02G
Tran Speed: 25000000
Rd Block Len: 512
SD version 1.10
High Capacity: No
Capacity: 1977614336
Bus Width: 1-bit
reading zImage
```

Figure 23 – ZedBoard U-Boot Booting Linux

10. When the Linux command prompt is reached, begin by mounting the FAT32 file system on the SD card.

The first partition on the SD card shows up as mmcblk0p1 in the device tree. This device represents MMC block device 0, partition 1. We will mount this

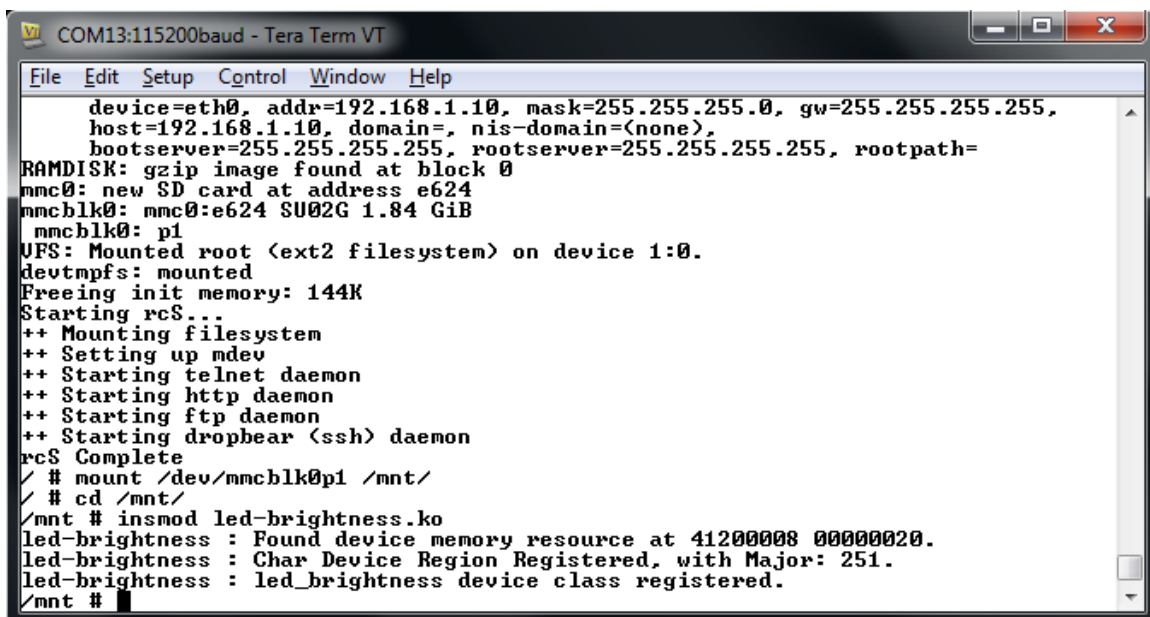
device to the mount point /mnt and change the working directory to that folder so that the FAT32 file system can be accessed.

```
# mount /dev/mmcblk0p1 /mnt/  
  
# cd /mnt/
```

11. Now that the FAT32 partition on the SD card has been mounted, the led-brightness driver module that was placed in Experiment 2 can be loaded into the kernel using the insmod command.

```
# insmod led-brightness.ko
```

Notice that a set of debug statements are sent from our driver to the console (via the printk() kernel call) and that these are now shown on the terminal.



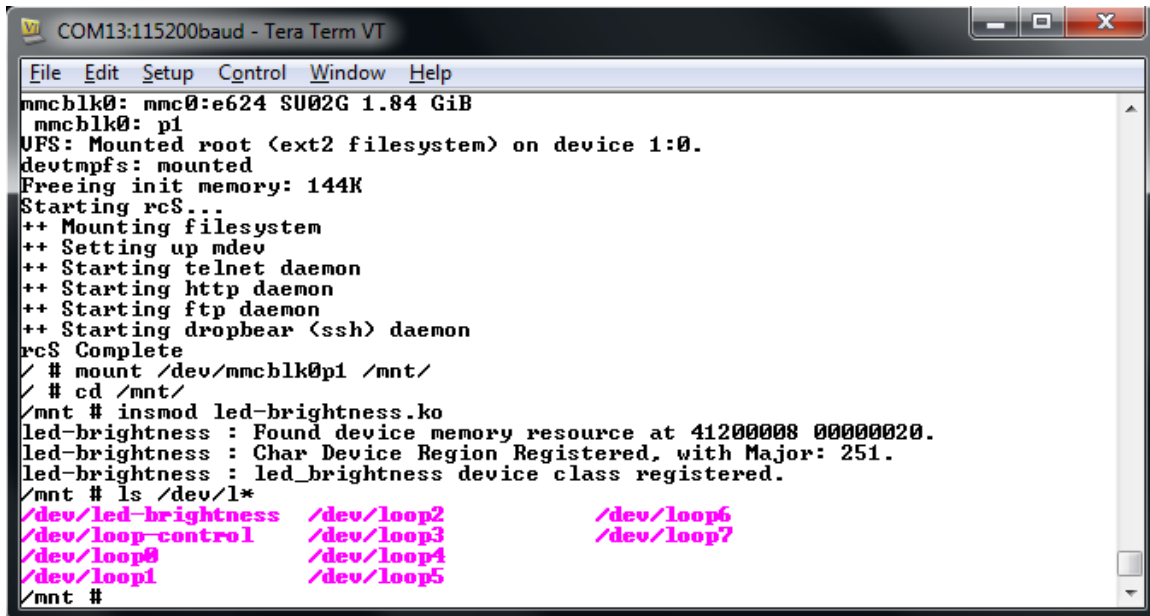
```
COM13:115200baud - Tera Term VT  
File Edit Setup Control Window Help  
device=eth0, addr=192.168.1.10, mask=255.255.255.0, gw=255.255.255.255,  
host=192.168.1.10, domain=, nis-domain=(none),  
bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=  
RAMDISK: gzip image found at block 0  
mmc0: new SD card at address e624  
mmcblk0: mmc0:e624 SU02G 1.84 GiB  
mmcblk0: p1  
UFS: Mounted root (ext2 filesystem) on device 1:0.  
devtmpfs: mounted  
Freeing init memory: 144K  
Starting rcS...  
++ Mounting filesystem  
++ Setting up mdev  
++ Starting telnet daemon  
++ Starting http daemon  
++ Starting ftp daemon  
++ Starting dropbear (ssh) daemon  
rcS Complete  
/ # mount /dev/mmcblk0p1 /mnt/  
/ # cd /mnt/  
/mnt # insmod led-brightness.ko  
led-brightness : Found device memory resource at 41200008 00000020.  
led-brightness : Char Device Region Registered, with Major: 251.  
led-brightness : led_brightness device class registered.  
/mnt #
```

Figure 24 – Inserting the Driver Module Into the Kernel

12. Take a look in the /dev folder for the led-brightness character device.

```
# ls /dev/l*
```

Notice that the led-brightness character device is now shown in the /dev folder. So far, the driver is working as we expected and is ready to be tested.



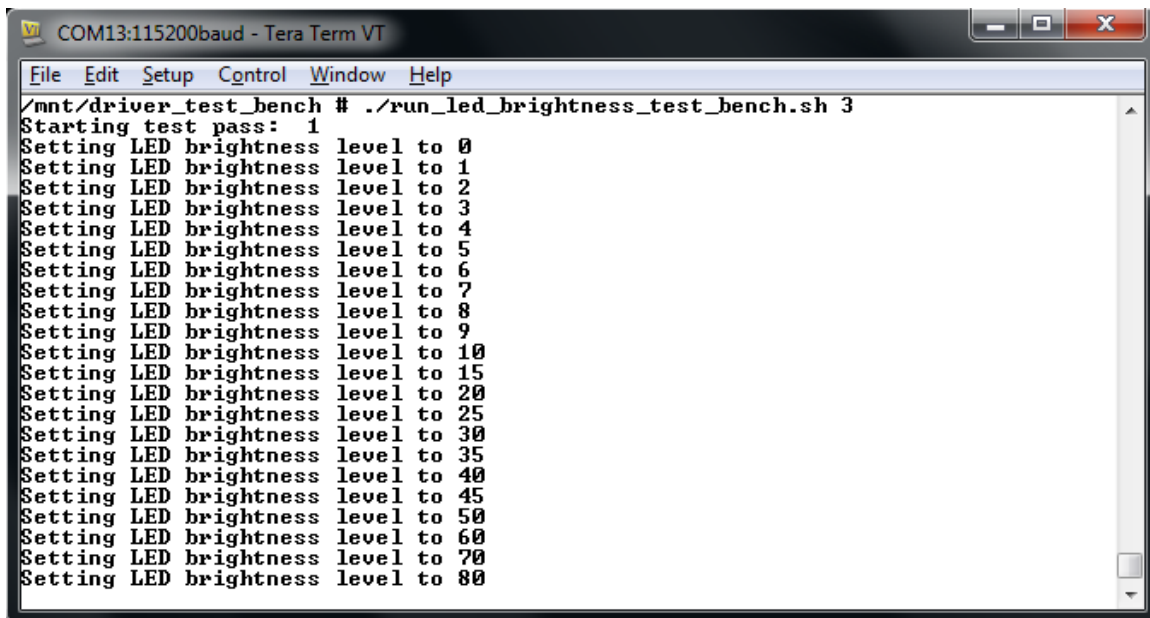
```
COM13:115200baud - Tera Term VT
File Edit Setup Control Window Help
mmcblk0: mmc0:e624 SU02G 1.84 GiB
mmcblk0: p1
UFS: Mounted root (ext2 filesystem) on device 1:0.
devtmpfs: mounted
Freeing init memory: 144K
Starting rcS...
++ Mounting filesystem
++ Setting up mdev
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting dropbear (ssh) daemon
rcS Complete
/ # mount /dev/mmcblk0p1 /mnt/
/ # cd /mnt/
/mnt # insmod led-brightness.ko
led-brightness : Found device memory resource at 41200008 00000020.
led-brightness : Char Device Region Registered, with Major: 251.
led-brightness : led_brightness device class registered.
/mnt # ls /dev/l*
/dev/led-brightness /dev/loop2 /dev/loop6
/dev/loop-control /dev/loop3 /dev/loop7
/dev/loop0 /dev/loop4
/dev/loop1 /dev/loop5
/mnt #
```

Figure 25 – Device Driver Module Inserted and led-brightness Device Listed

13. Run the driver test bench by changing the working directory to the **driver_test_bench/** folder on the SD card and executing the **run_led_brightness_test_bench.sh** shell script.

```
# cd driver_test_bench/  
# ./run_led_brightness_test_bench.sh 3
```

The numerical argument to the script represents the number of times that the test bench is run against our new device. Closely watch LEDs LD0-LD7 and wait for the brightness level to increase as the test bench script executes.



```
COM13:115200baud - Tera Term VT  
File Edit Setup Control Window Help  
/mnt/driver_test_bench # ./run_led_brightness_test_bench.sh 3  
Starting test pass: 1  
Setting LED brightness level to 0  
Setting LED brightness level to 1  
Setting LED brightness level to 2  
Setting LED brightness level to 3  
Setting LED brightness level to 4  
Setting LED brightness level to 5  
Setting LED brightness level to 6  
Setting LED brightness level to 7  
Setting LED brightness level to 8  
Setting LED brightness level to 9  
Setting LED brightness level to 10  
Setting LED brightness level to 15  
Setting LED brightness level to 20  
Setting LED brightness level to 25  
Setting LED brightness level to 30  
Setting LED brightness level to 35  
Setting LED brightness level to 40  
Setting LED brightness level to 45  
Setting LED brightness level to 50  
Setting LED brightness level to 60  
Setting LED brightness level to 70  
Setting LED brightness level to 80
```

Figure 26 – Device Driver Test Bench Underway

14. Once the test bench has completed execution. Un-mount the SD card from the **/mnt** mount point.

The test bench is simply a script sends raw binary data from test pattern files to the LED brightness driver and verifies that the driver is working correctly enough to begin writing a user space application which will be explored in the next lab.

```
# cd /  
# umount /mnt/
```

Questions:

Answer the following questions:

- *What is the device file name of the LED brightness driver after the driver module is inserted?*

Exploring Further

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

- Explore the driver source code further. How does the device driver identify device tree resource information?

This concludes Lab 3.2.

Revision History

Date	Version	Revision
17 Sep 12	00	Initial Draft
27 Sep 12	01	Revised Draft
19 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 is the purpose of the configuration update that is performed in Step 10?*

The kernel configuration update is performed because the build scripts detected a change in the Linux source tree which contained build options for which an existing configuration was not found in the existing .config file.

Experiment 2

- *Why must the device tree source file be compiled to a binary format?*

The kernel is capable of interpreting the binary format of the device tree at boot time and it is unable to interpret the plain-text format of the device tree source file.

Experiment 3

- *What is the device file name of the LED brightness character driver after the driver module is inserted?*

/dev/led-brightness