

**Xilinx Open Source Linux  
for the  
Avnet Zynq™ Mini-ITX Development Kit**

**ZYNQ**  <sup>®</sup>

**Version 1.0  
July 2014**

## Table of Contents

|   |    |
|---|----|
| Introduction .....                                    | 2  |
| Objectives .....                                      | 2  |
| Reference Design Requirements .....                   | 3  |
| Software .....  | 3  |
| Hardware .....  | 3  |
| Supplied Files .....                                  | 4  |
| Setting Up the Zynq Mini-ITX Development Kit .....    | 5  |
| PC Setup .....  | 7  |
| Installing the UART Driver and Virtual COM Port ..... | 7  |
| Installing a Serial Console on a Windows 7 Host ..... | 7  |
| Hardware Design Block Diagram .....                   | 8  |
| Running the Demo Files .....                          | 9  |
| Boot Linux from Micro SD Card .....                   | 11 |
| LED Brightness Controller Demo .....                  | 13 |
| Appendix I: Installation of USB UART Driver .....     | 16 |
| Download and Install the Required Software .....      | 16 |
| Determining the Virtual COM Port .....                | 18 |
| Appendix II: Where to Get More Information .....      | 20 |
| ZedBoard Website .....                                | 20 |
| Appendix III: Useful Linux commands .....             | 21 |
| Revision History .....                                | 23 |

## Introduction

This document describes a Zynq processor system design implemented and tested on the Avnet Zynq Mini-ITX development kit and running Linux.

The Linux development for this example design was performed with the Xilinx Linux, u-boot, and device tree git repositories installed in a 64-bit CentOS Linux virtual machine on a Windows 7 host PC. The Xilinx Vivado development tools were installed on the Windows 7 host PC. It is beyond the scope of this tutorial to describe how to create this virtual machine and install the Linux development host OS. The user is expected to possess some knowledge of Linux and Linux commands, and be comfortable working with Linux command-line tools.

In any system level design, hardware and software must interact efficiently to create a winning solution. In a real-world system, many elements should be considered in your hardware platform specification. Using a Zynq AP SoC allows you to easily prototype many hardware combinations to see how they compare. Factors to consider include:

1. Performance
2. Cost
3. Flexibility
4. System complexity
  - a. Size
  - b. Heat dissipation
  - c. Power consumption
  - d. Build time (development)
5. Scalability
6. Future upgrade requirements

In this example design our goal is to provide a reliable hardware platform to demonstrate Linux support for custom GPIO peripherals integrated into the Zynq Programmable Logic (PL). To that end we will not be pushing the performance thresholds of the Zynq device today so that we may minimize our development time.

## Objectives

This tutorial will demonstrate the following with the supplied files copied to a micro SD card. These files are also pre-installed on the micro SD card that is shipped with the Zynq Mini-ITX development kit.

- Run the First Stage Boot Loader (FSBL) to configure the Zynq PL and execute u-boot.
- Load the Linux kernel, device tree blob, and RAMdisk Root File System (RFS).
- Boot the Linux kernel and mount the RAMdisk root file system (RFS).
- Run the led\_dimmer.elf Linux application to interact with the LEDs and pushbutton switches on the baseboard.

## Reference Design Requirements

### Software

The software requirements for this reference design are:

- Linux, Windows XP, Windows 7  
[www.xilinx.com/ise/ossupport/index.htm](http://www.xilinx.com/ise/ossupport/index.htm)
- Xilinx Vivado Design Edition 2013.4 (Includes SDK)

### Hardware

The hardware setup for this reference design is:

- Computer with at least 3.7 GB RAM (recommended)  
[www.xilinx.com/ise/products/memory.htm](http://www.xilinx.com/ise/products/memory.htm)
- Avnet Zynq Mini-ITX Development Kit with the Zynq 7045 or 7100 AP SoC device
- USB-A to USB-micro B cable (2x)
  - JTAG
  - USB UART
- Cat-5 Ethernet cable
- Host PC with 10/100/1000 compatible Ethernet NIC
- Host PC or network router to act as DHCP server
- ATX Power supply
- Brand-name micro-SD card 1GB or larger and Class 4 or better

## Supplied Files

The following directory structure is included with this reference design:

**boot:** Contains the files to create the bootloader required to boot from the micro SD card:

**BOOT.BIN:** Boot image of First Stage Boot Loader (FSBL), PL bistream, and u-boot application.

**bootimage.bif:** Boot image information text file.

**cp\_from\_sdk.bat:** Batch file to copy hardware bitstream and software executables from the SDK workspace.

**makeboot.bat:** Batch file to run the command to create the boot image file.

**u-boot.elf:** Golden pre-built u-boot binary used to boot Linux.

**zmitx\_fsbl.elf:** The golden ARM executable for the Zynq FSBL.

**zynq\_system\_wrapper.bit:** The golden FPGA bitstream of the hardware design required to run the GPIO test application.

**doc:** Contains documentation for this design:

**ZMITX\_OOB\_Linux\_Guide\_v1\_0.pdf:** This document.

**hw\_sources:** Contains HDL and device constraints files.

**ps\_sources:** Contains Zynq Processing System definition file for the Zynq Mini-ITX board.

**sd\_card:** Contains the files to run Linux:

**BOOT.BIN:** Boot image of First Stage Boot Loader (FSBL), PL bistream, and u-boot application.

**devicetree.dtb:** Golden Device Tree Blob (DTB) file. This describes the hardware present in the system to the Linux kernel.

**ulmage:** Golden pre-built Linux kernel that includes the device driver required for the led\_dimmer software application.

**uramdisk.image.gz:** Golden pre-built RAMdisk that includes the led\_dimmer software application.

**sw\_sources:** Contains the sources for the Linux LED dimmer sw driver, sw application, and devicetree.

**Xil:** Contains the Vivado project and SDK workspace.

# Setting Up the Zynq Mini-ITX Development Kit

Refer to the following figure and perform the following steps to set up the board for booting Linux.

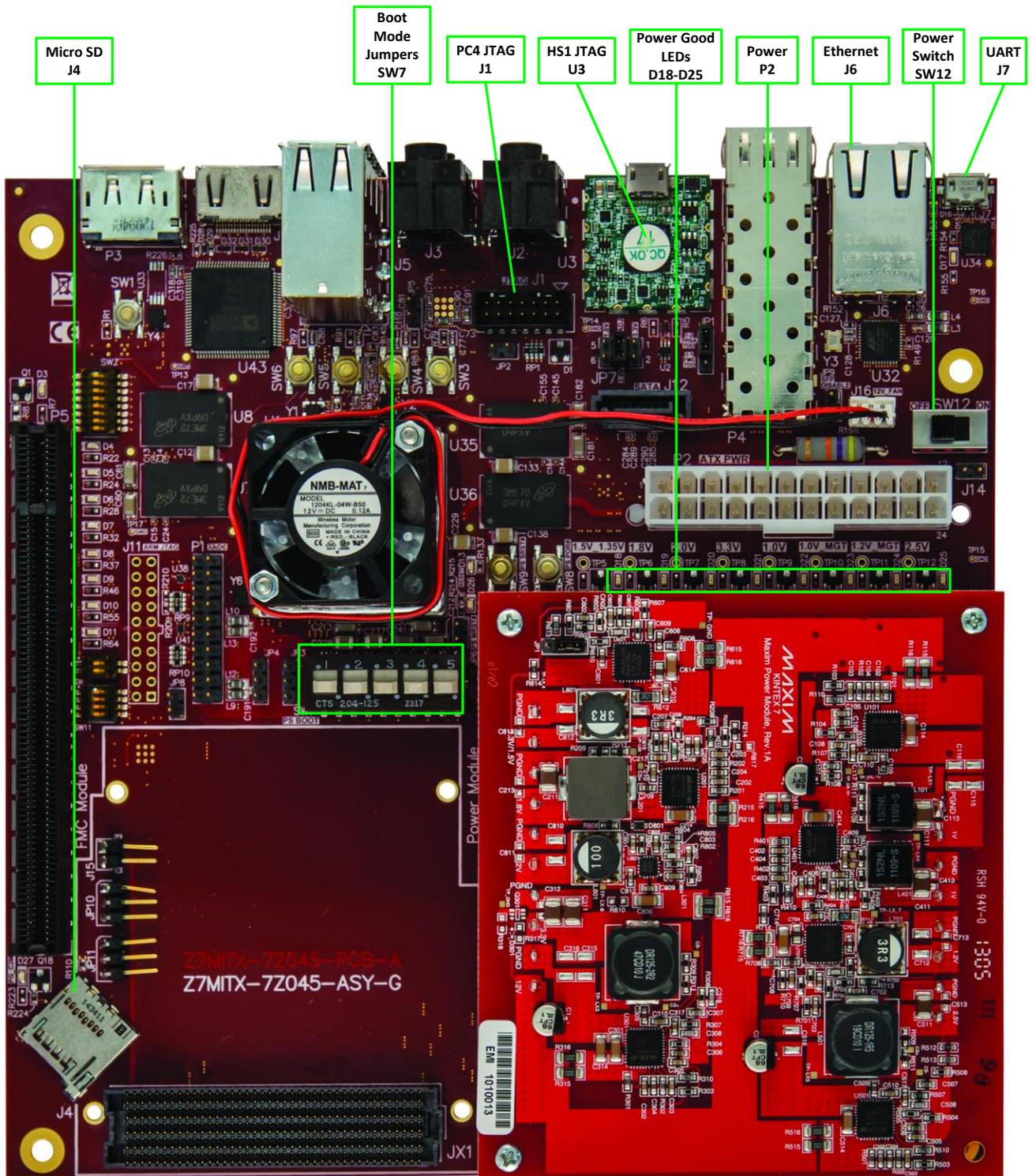


Figure 1 – Zynq Mini-ITX Development Kit

1. Plug a USB-micro B cable into the host PC and the USB UART port (J7) on the Mini-ITX Development Board. When the board is powered on the UART link LED (D17) will light indicating a USB link connection to the PC.
2. Connect the Cat-5 network cable to your Ethernet network router or host PC and Zynq Mini-ITX Ethernet jack (J6).
3. Set the boot mode switches (SW7[1:5]) to [off, off, on, on, off] to configure the Zynq Mini-ITX to boot from the micro SD card.
4. Connect the ATX power supply to the power connector (P2).

## PC Setup

Follow the steps below to configure the Ethernet NIC of the development host PC to establish an Ethernet connection directly with the Zynq Mini-ITX Development Board.

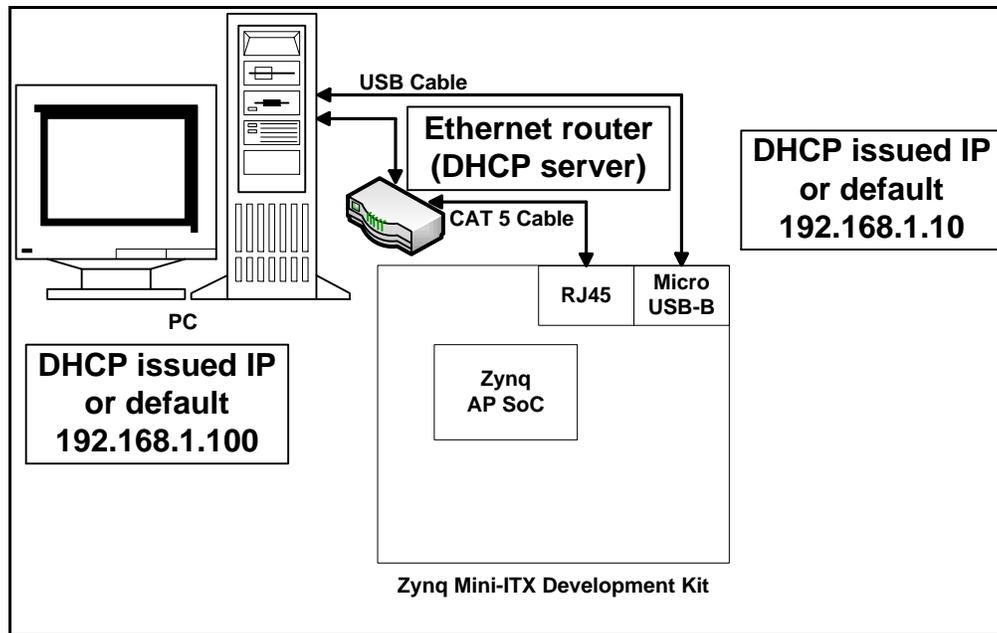


Figure 2 – Network Connection Setup

1. If you choose to connect the Zynq Mini-ITX Development Kit directly to the host PC and are not running a DHCP server you will need to configure the host PC with an IP address of **192.168.1.100** and a subnet mask of **255.255.255.0**.

## Installing the UART Driver and Virtual COM Port

If the Zynq Mini-ITX Development Kit has not been connected to the host PC before, it may be necessary to install the software driver for the virtual COM port. The driver installation for the Silicon Labs CP210x USB-UART bridge on the Zynq Mini-ITX Development Kit is described in detail in [Appendix I: Installation Of USB UART Driver](#).

## Installing a Serial Console on a Windows 7 Host

Starting with Windows 7, Microsoft no longer includes the HyperTerminal terminal emulator software. However, this example design requires use of terminal emulation software for a serial console connection to the development board. A suitable free and open-source replacement for HyperTerminal is TeraTerm. Download and install instructions for TeraTerm can be found at <http://en.sourceforge.jp/projects/ttssh2>. As an alternative the Terminal applet in the Xilinx SDK may also be used.

## Hardware Design Block Diagram

The hardware platform for this guide integrates a Zynq Processor System (PS) and Programmable Logic (PL) bitstream that includes a standard AXI GPIO peripheral and custom switch debouncer and Pulse Width Modulator (PWM) peripherals. The following figure shows a high-level block diagram of the hardware design. The design requires:

- Xilinx Zynq Z7045 or Z7100 AP SoC
- 1GB of DDR3 SDRAM
- Micro-SD Card
- Gigabit Ethernet
- Gigabit Ethernet
- USB UART Bridge
- LEDs and Pushbutton Switches
- Interrupt Controller
- Triple Timer Counter

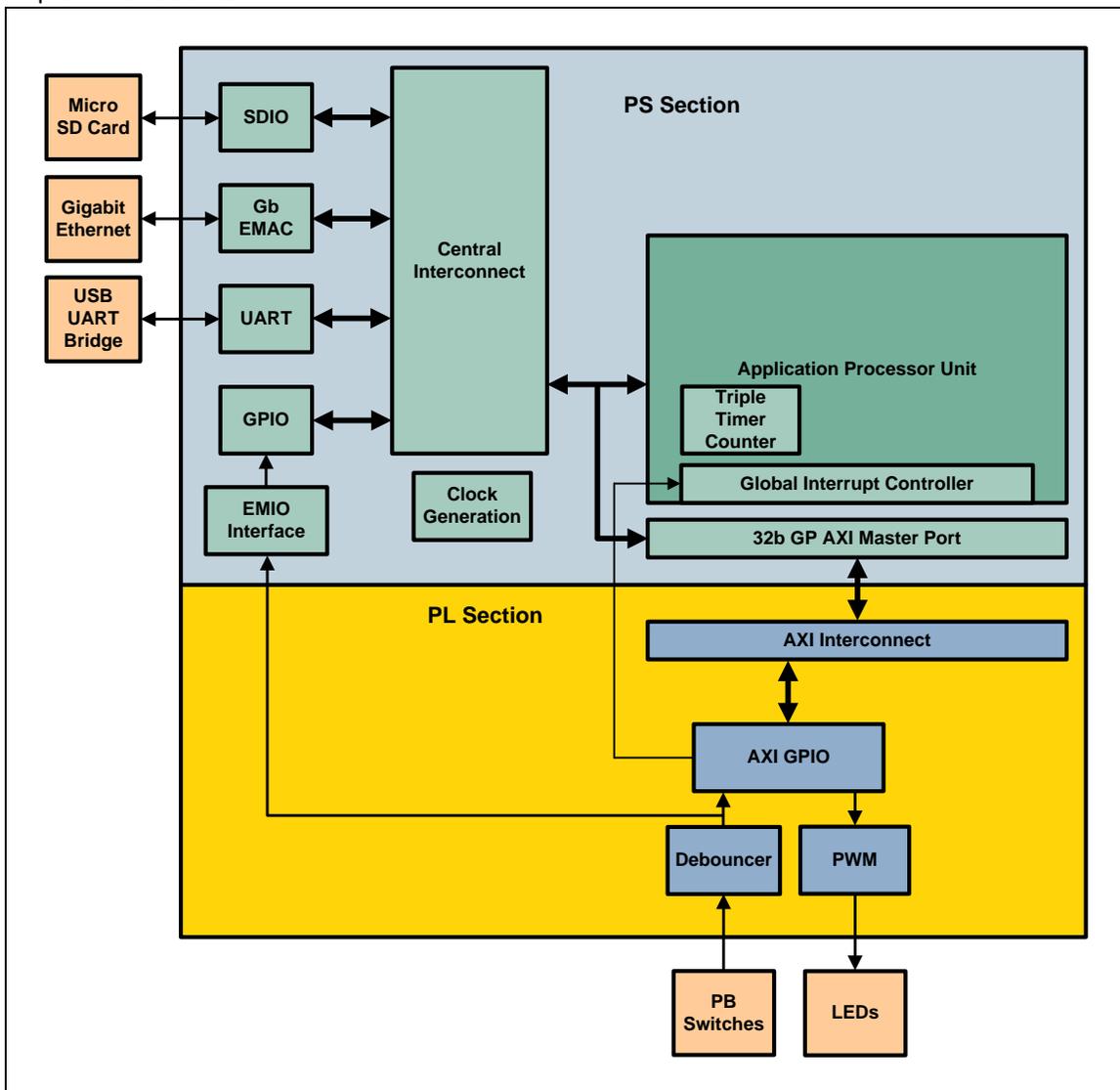


Figure 3 - Example Design Block Diagram

## Running the Demo Files

You can load the Zynq device and boot Linux by copying the supplied files to a micro SD card. You must have the hardware set up and connected as described in [Setting Up the Zynq Mini-ITX Development Kit](#).

The Zynq hardware platform and supplied Linux kernel for this example design are very similar to those used for the Avnet ZedBoard [Introduction to Zynq](#) and [Implementing Linux on the Zynq-7000 SoC](#) Speedway Training Courses posted on [www.zedboard.org/trainings-and-videos](http://www.zedboard.org/trainings-and-videos). If this is your first encounter with Zynq, or if you would like more information and instruction about how to build your own Zynq system with integrated custom peripherals in the PL, these Speedway courses are a very valuable resource.

The supplied Linux kernel includes a custom device driver to tie the LED controller software application to the LED outputs. The custom software application has been built into the RAMdisk. This is a simple application that demonstrates the full integration of custom PL hardware with the Zynq PS and using Linux to control and interact with the hardware. The software application polls the pushbutton switch inputs via the Linux sysfs virtual file system and writes data to the AXI GPIO peripheral to control the attached Pulse Width Modulator (PWM) via the custom device driver built into the kernel. The Device Tree Blob (DTB) informs the kernel at boot time about the name and address location for the LED hardware associated with the device driver. This is what tells the kernel to load this device driver so it is ready to use by the software application once Linux is booted.

Refer to the figure below when booting Linux and running the LED controller software application. Note the location of the pushbutton switches and LEDs. The micro SD card slot is located on the corner of the Zynq Mini-ITX in between the PCIe and FMC connectors.

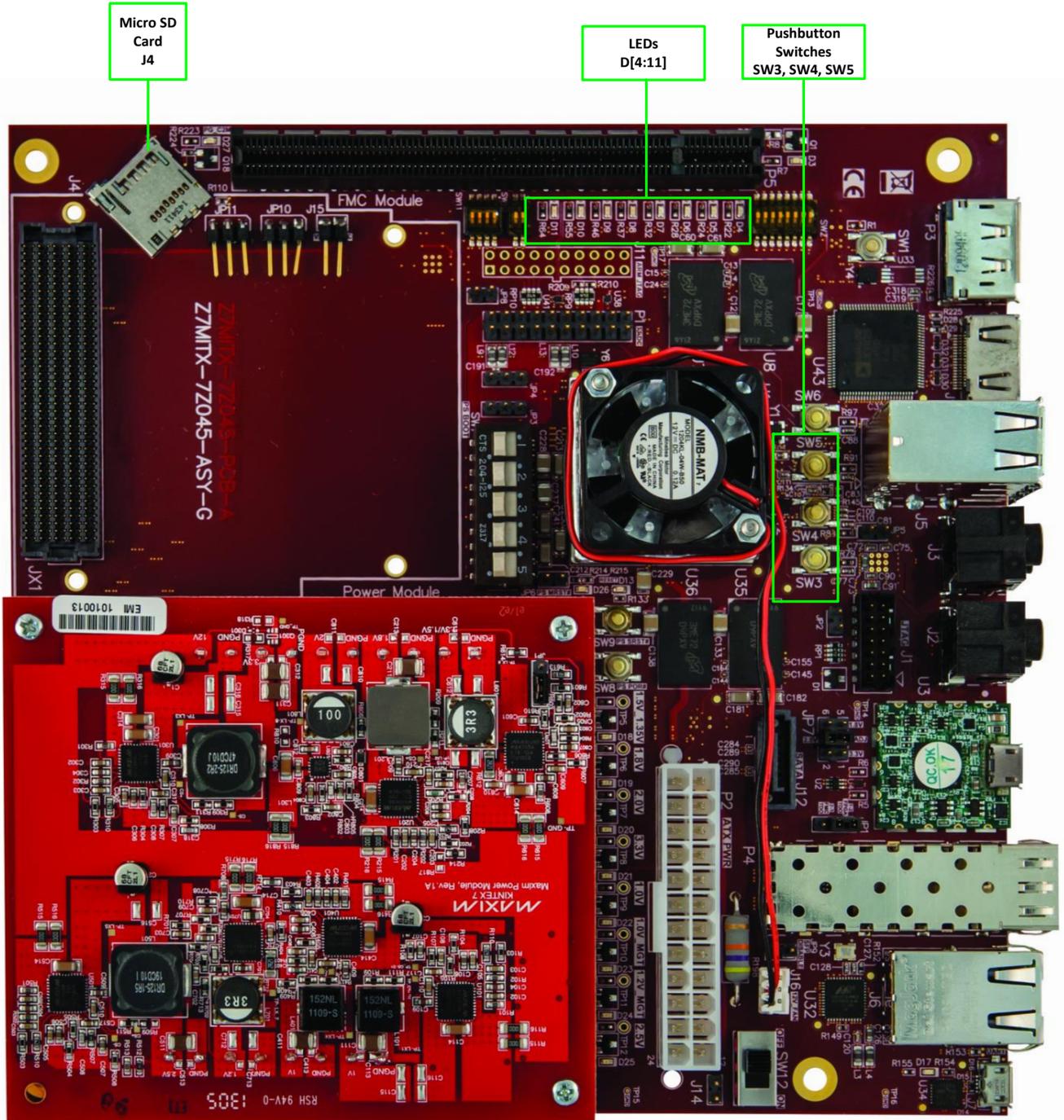
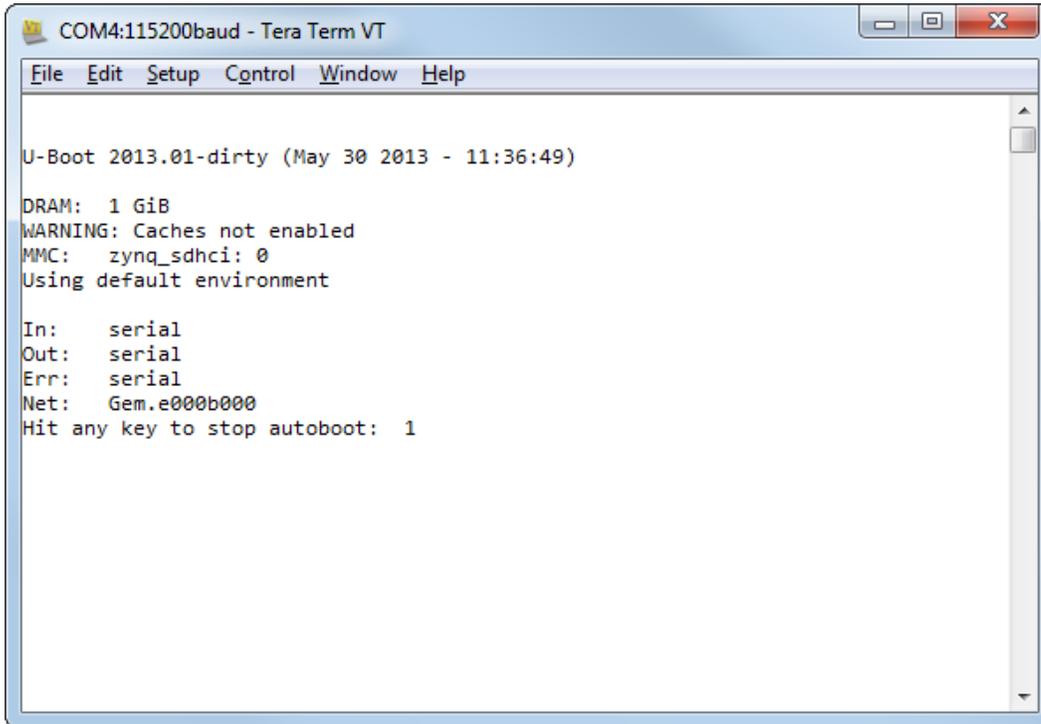


Figure 4 – Zynq Mini-ITX Linux Application GPIO and Micro SD Card

## Boot Linux from Micro SD Card

1. Insert a micro SD card into an available slot on the host PC. The micro SD card should be formatted FAT16 or FAT32. Though not strictly necessary, it may be best to delete or move any files currently on the micro SD card to a folder on the host PC.
2. Using Windows Explorer, navigate to the <installation>\sd\_card folder and copy the files in that folder to the micro SD card.
3. To make sure Windows unmounts the micro SD card correctly, navigate to the drive letter associated with the micro SD card in Windows Explorer. Right-click on the drive letter and select **Eject**.
4. Remove the micro SD card from the host PC and insert it into the micro SD card cage on the Zynq Mini-ITX.
5. Verify the board is setup correctly as described in [Setting Up the Zynq Mini-ITX Development Kit](#).
6. Slide the power switch (SW12) to the ON position. You will see 8 green 'power good' LEDs lit near the power supply module.
7. Start a serial terminal session and set the serial port parameters to **115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control.
8. After a few seconds you should see the blue DONE LED (D3) light, indicating the PL has been configured with the bitstream that includes that AXI GPIO and custom switch debouncer and PWM peripherals.

- Once the PL has been configured, u-boot will start and after a 3 second pause will load the kernel, RAMdisk, and DTB into DDR memory.



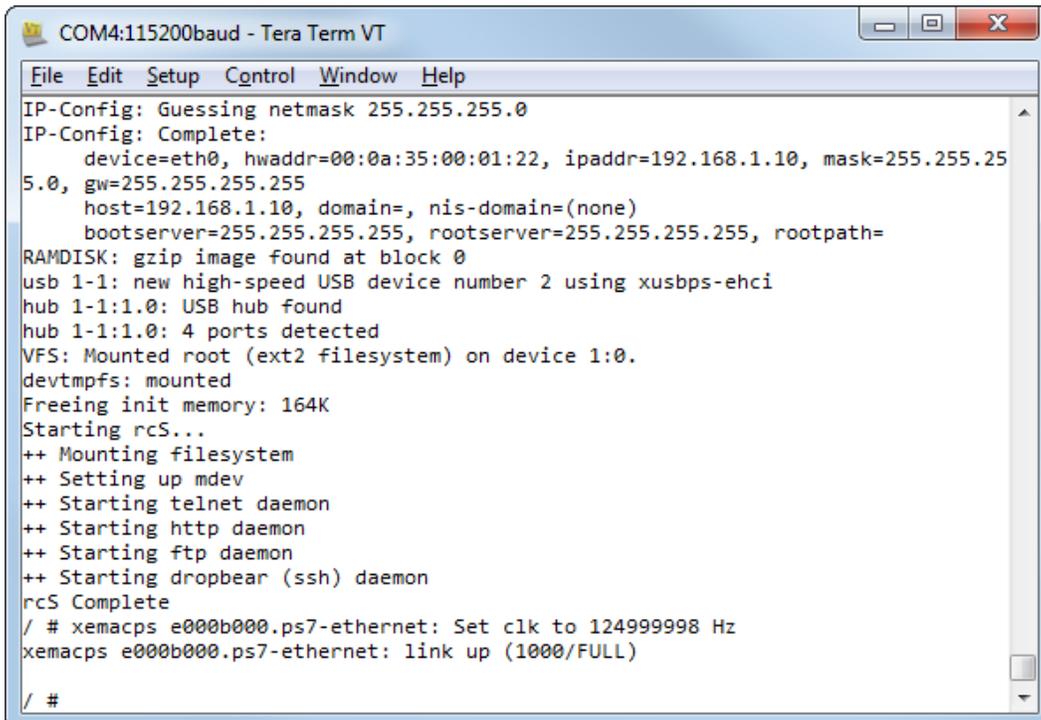
```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

U-Boot 2013.01-dirty (May 30 2013 - 11:36:49)

DRAM: 1 GiB
WARNING: Caches not enabled
MMC: zynq_sdhci: 0
Using default environment

In: serial
Out: serial
Err: serial
Net: Gem.e000b000
Hit any key to stop autoboot: 1
```

- The Linux kernel will then boot, read the DTB, and mount the RAMdisk. Once booted you should see a screen similar to below.



```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

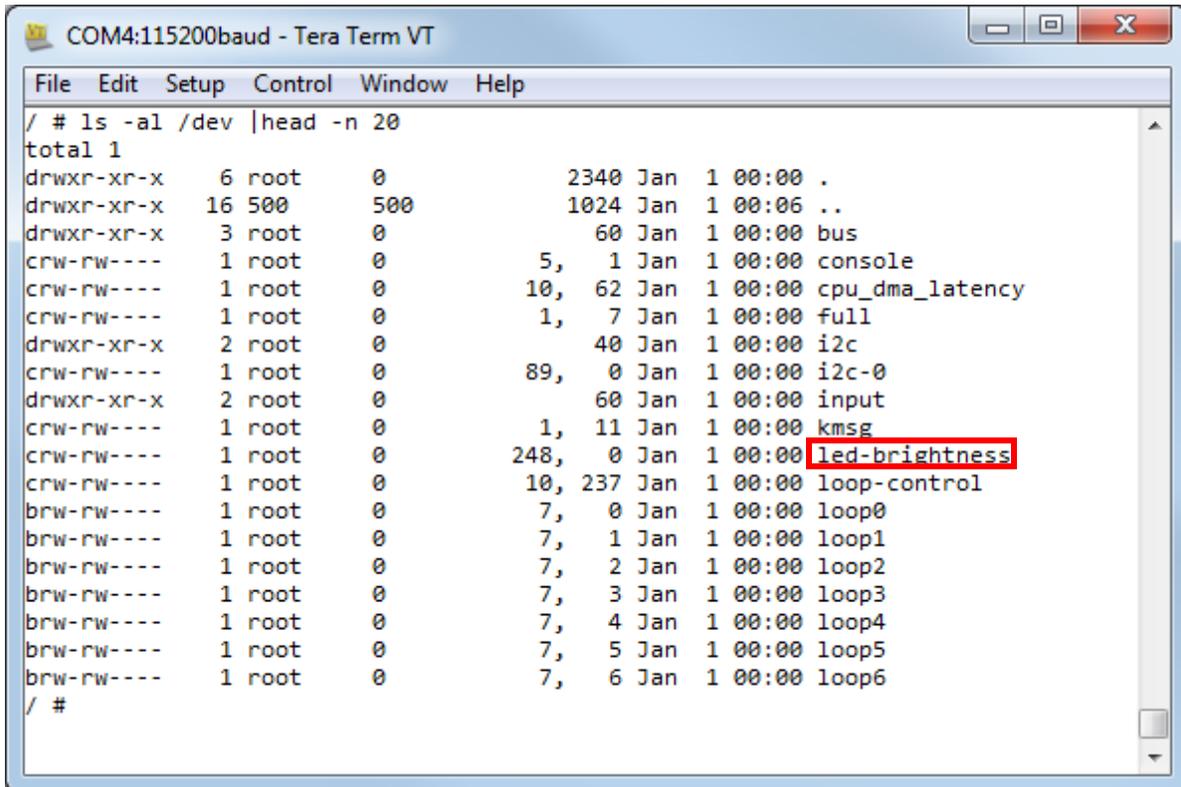
IP-Config: Guessing netmask 255.255.255.0
IP-Config: Complete:
    device=eth0, hwaddr=00:0a:35:00:01:22, ipaddr=192.168.1.10, mask=255.255.25
5.0, gw=255.255.255.255
    host=192.168.1.10, domain=, nis-domain=(none)
    bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
RAMDISK: gzip image found at block 0
usb 1-1: new high-speed USB device number 2 using xusbps-ehci
hub 1-1:1.0: USB hub found
hub 1-1:1.0: 4 ports detected
VFS: Mounted root (ext2 filesystem) on device 1:0.
devtmpfs: mounted
Freeing init memory: 164K
Starting rcS...
++ Mounting filesystem
++ Setting up mdev
++ Starting telnet daemon
++ Starting http daemon
++ Starting ftp daemon
++ Starting dropbear (ssh) daemon
rcS Complete
/ # xemacps e000b000.ps7-ethernet: Set clk to 124999998 Hz
xemacps e000b000.ps7-ethernet: link up (1000/FULL)

/ #
```

## LED Brightness Controller Demo

1. Verify the LED brightness device driver is loaded by checking for the presence of the led-brightness file in the /dev folder. Type the following command to list the first 20 files of the /dev folder:

```
/ # ls -al /dev |head -n 20
```



```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help
/ # ls -al /dev |head -n 20
total 1
drwxr-xr-x  6 root    0          2340 Jan  1 00:00 .
drwxr-xr-x 16 500    500        1024 Jan  1 00:06 ..
drwxr-xr-x  3 root    0           60 Jan  1 00:00 bus
crw-rw----  1 root    0           5,  1 Jan  1 00:00 console
crw-rw----  1 root    0          10,  62 Jan  1 00:00 cpu_dma_latency
crw-rw----  1 root    0           1,  7 Jan  1 00:00 full
drwxr-xr-x  2 root    0           40 Jan  1 00:00 i2c
crw-rw----  1 root    0          89,  0 Jan  1 00:00 i2c-0
drwxr-xr-x  2 root    0           60 Jan  1 00:00 input
crw-rw----  1 root    0           1, 11 Jan  1 00:00 kmsg
crw-rw----  1 root    0        248,  0 Jan  1 00:00 led-brightness
crw-rw----  1 root    0       10, 237 Jan  1 00:00 loop-control
brw-rw----  1 root    0           7,  0 Jan  1 00:00 loop0
brw-rw----  1 root    0           7,  1 Jan  1 00:00 loop1
brw-rw----  1 root    0           7,  2 Jan  1 00:00 loop2
brw-rw----  1 root    0           7,  3 Jan  1 00:00 loop3
brw-rw----  1 root    0           7,  4 Jan  1 00:00 loop4
brw-rw----  1 root    0           7,  5 Jan  1 00:00 loop5
brw-rw----  1 root    0           7,  6 Jan  1 00:00 loop6
/ #
```

2. The LED controller software application is named **led\_dimmer.elf** and is pre-installed in the **/opt/Avnet** folder. Type the following command to run the application:

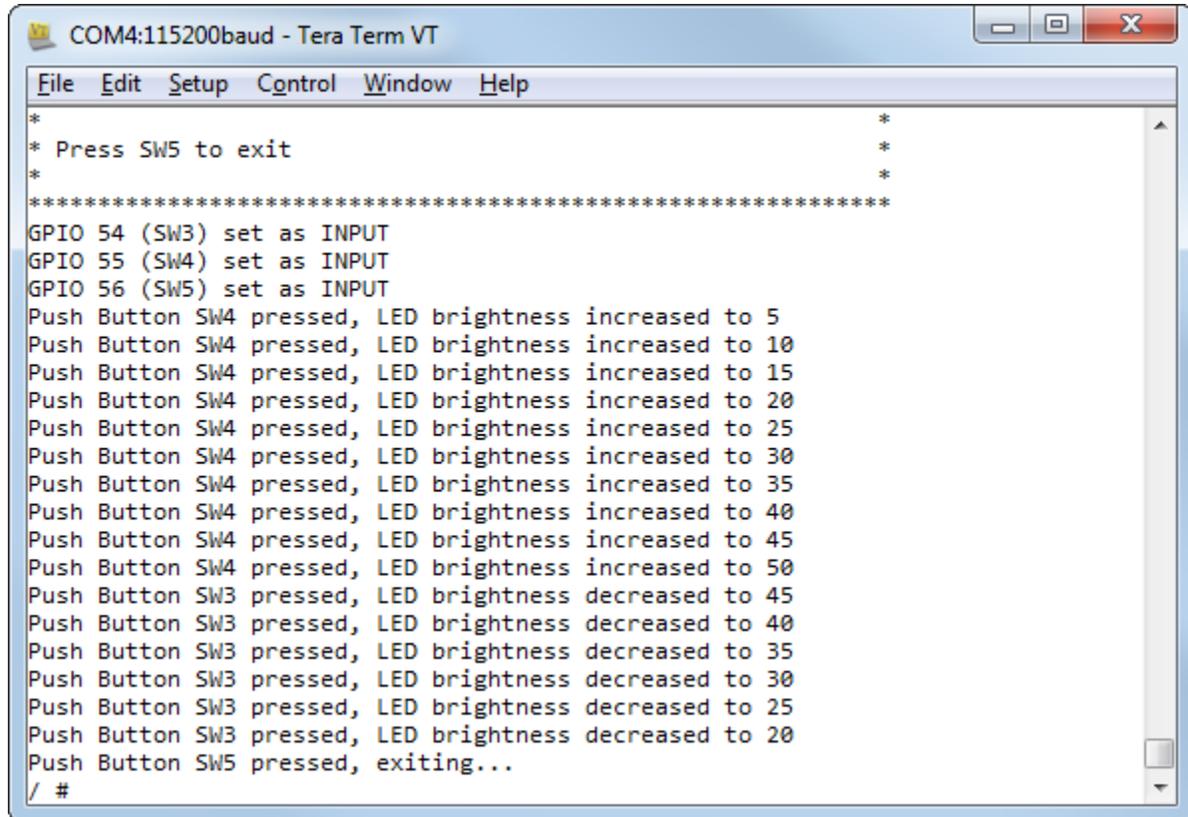
```
/ # /opt/Avnet/led_dimmer.elf
```

3. You will see the application banner shown below. GPIO 54, 55, and 56 are mapped to the pushbutton switch inputs. Linux sees all off the GPIO as a large array of 118 elements that include the 54 MIO bits mapped to PS peripherals and 64 bits mapped to EMIO. Note that GPIO 54, 55, and 56 assigned here are the first 3 GPIO above the MIO[0:53] bits.

```

COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help
brw-rw---- 1 root 0 7, 3 Jan 1 00:00 loop3
brw-rw---- 1 root 0 7, 4 Jan 1 00:00 loop4
brw-rw---- 1 root 0 7, 5 Jan 1 00:00 loop5
brw-rw---- 1 root 0 7, 6 Jan 1 00:00 loop6
/ #
/ #
/ # /opt/Avnet/led_dimmer.elf
*****
*
* Zynq Mini-ITX - LED Dimmer Application *
*
* Pushbutton switches are located on the Zynq Mini-ITX board *
* between the Zynq AP SOC and USB stack (J5) *
*
* Press SW3 to decrease LED brightness *
*
* Press SW4 to increase LED brightness *
*
* Press SW5 to exit *
*
*****
GPIO 54 (SW3) set as INPUT
GPIO 55 (SW4) set as INPUT
GPIO 56 (SW5) set as INPUT
    
```

4. Start by pressing **SW4** a few times to see the LEDs turn on and get progressively brighter as each button press tells the device driver to write a new value to the AXI\_GPIO to change the value of the PWM. Press **SW3** a few times to conversely see the LEDs get progressively dimmer using the same mechanism. Press **SW5** to turn the LEDs off and exit the application.



```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help
*
* Press SW5 to exit
*
*****
GPIO 54 (SW3) set as INPUT
GPIO 55 (SW4) set as INPUT
GPIO 56 (SW5) set as INPUT
Push Button SW4 pressed, LED brightness increased to 5
Push Button SW4 pressed, LED brightness increased to 10
Push Button SW4 pressed, LED brightness increased to 15
Push Button SW4 pressed, LED brightness increased to 20
Push Button SW4 pressed, LED brightness increased to 25
Push Button SW4 pressed, LED brightness increased to 30
Push Button SW4 pressed, LED brightness increased to 35
Push Button SW4 pressed, LED brightness increased to 40
Push Button SW4 pressed, LED brightness increased to 45
Push Button SW4 pressed, LED brightness increased to 50
Push Button SW3 pressed, LED brightness decreased to 45
Push Button SW3 pressed, LED brightness decreased to 40
Push Button SW3 pressed, LED brightness decreased to 35
Push Button SW3 pressed, LED brightness decreased to 30
Push Button SW3 pressed, LED brightness decreased to 25
Push Button SW3 pressed, LED brightness decreased to 20
Push Button SW5 pressed, exiting...
/ #
```

5. Feel free to experiment with running Linux commands and utilities. Various popular services and daemons, such as telnet, are running in the kernel.

## Appendix I: Installation of USB UART Driver

Many of the Avnet evaluation boards are equipped with the Silicon Labs CP2102 USB-to-UART Bridge IC. This connects a PC's USB port to the evaluation board and looks like a UART to the PC. A virtual COM port will be created on the PC by means of a Silicon Labs CP2102 USB-to-UART bridge driver.

To install the Silicon Labs drivers follow the instructions listed below.

### Download and Install the Required Software

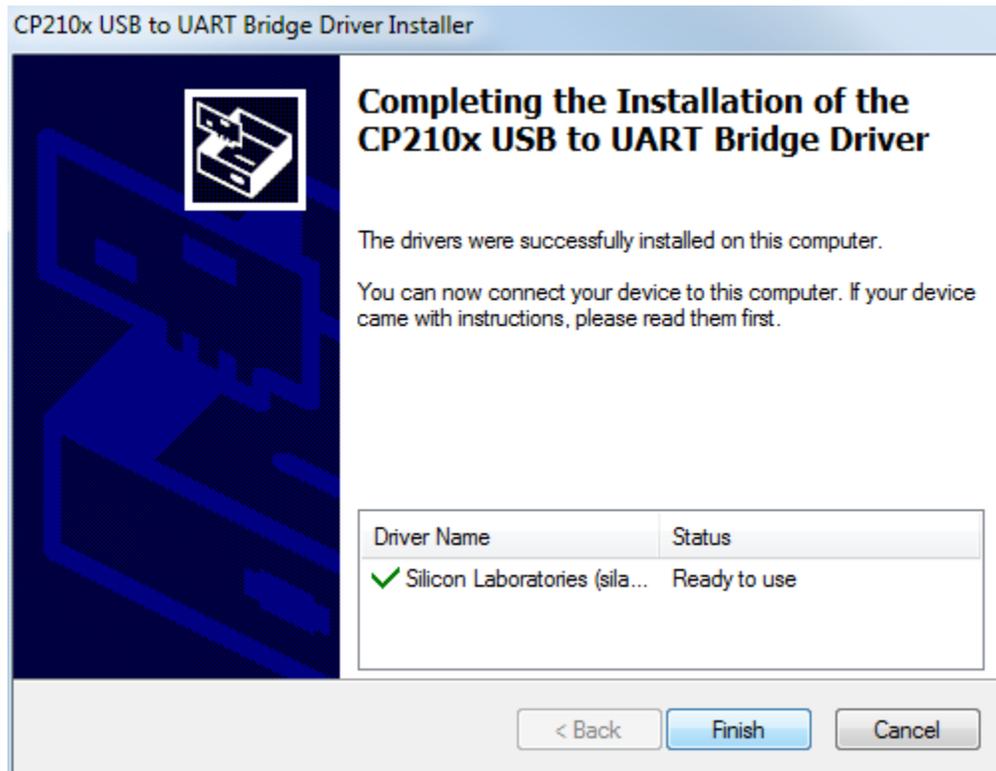
1. Using your web browser, navigate to the Silicon Labs website:

<http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx>

2. Download the **VCP Driver Kit** for your PC's operating system. Drivers for MacOS and Linux are also available.

The screenshot shows the Silicon Labs website interface. At the top, there is a navigation bar with links for 'Careers', 'News', and 'Investor Relations', and a 'Log In' button. Below this is a main navigation bar with 'Products', 'Applications', 'Support', and 'Buy or Sample' tabs, along with a search bar and a 'More' dropdown. The main content area displays the breadcrumb 'Silicon Labs > Products > MCUs > USB to UART Bridge VCP Drivers' and the title 'CP210x USB to UART Bridge VCP Drivers'. A brief description follows, mentioning that these drivers are required for device operation as a Virtual COM Port. A link to 'AN197: The Serial Communications Guide for the CP210x' is provided. Below this, a section titled 'Download for Windows XP/Server 2003/Vista/7/8 (v6.6.1)' contains a list of download links. The 'VCP Driver Kit' link is highlighted with a red box. Other links include 'Revision History', 'USB Driver Installation Utility Application Note, AN335', and 'USB Driver Installation Utility Software, AN335SW'. On the right side, there is a sidebar with 'Find Products Fast' (Parametric Search, Cross-Reference Search), 'Get Support & Tools' (Software Downloads, Development Tools, Reference Designs, Documentation, Knowledgebase / FAQ, Training & Resources), and 'Need Help?' (Contact Support). A 'GET THE LATEST' button is located at the bottom right of the main content area.

- Once the file is downloaded, extract the **CP210x VCP Driver Kit** archive. For example, for Windows XP/Vista/7 the file is `CP210x_VCP_Windows.zip`. Once the archive is extracted, open the folder where the archive was extracted and choose the correct installer for a 32-bit (`CP210xVCPInstaller_x86.exe`) or 64-bit (`CP210xVCPInstaller_x64.exe`) PC. The installer will guide you through the setup. Accept the license agreement and install the software on your PC. Click **FINISH** when completed.



## Determining the Virtual COM Port

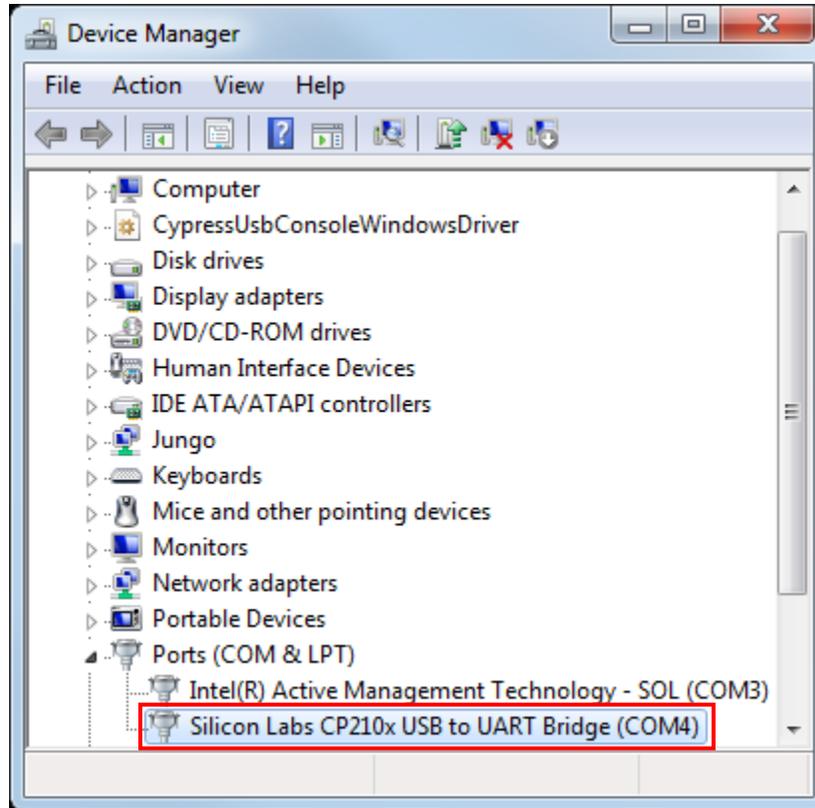
Now you can connect the evaluation board's USB-to-UART port to one of the USB ports on your PC. The new hardware detection will pop up and enumeration of the driver will be started. Once finished a virtual COMx port is created and you are ready to setup a connection using Windows HyperTerminal or comparable serial terminal emulation utility. Follow these instructions to determine the COMx port assigned to the USB-to-UART bridge:



1. Open the Device Manager by right-clicking on , select Properties, then click on the Device Manager.



2. In the Device Manager, scroll down to Ports and expand the list. You will see the Silicon Labs CP210x USB to UART Bridge and its assigned COM port. In the example below, it is COM4. Make note of this COM port number for use with the serial terminal you will use elsewhere in this design tutorial. This concludes these USB UART driver and virtual COM port installation instructions.



## Appendix II: Where to Get More Information

### ZedBoard Website

[www.zedboard.org/product/mini-itx](http://www.zedboard.org/product/mini-itx)

### Zynq Tutorials and Videos

[Introduction to Zynq](#)

[Implementing Linux on the Zynq-7000 SoC](#)

## Appendix III: Useful Linux commands

The table below contains a few Linux commands with commonly used options. It is not a complete list by any means, and there are often a great many more options for each command. However, for the Linux novice, it is hoped that the commands listed here may prove helpful to you. Not all commands are available under BusyBox, but all will be recognized by Red Hat Enterprise Linux or CentOS Linux.

All commands are case-sensitive.

| Command Format                         | Example                          | Purpose   |
|--|----------------------------------|---|
| bzip2 -d <file>                        | bzip2 -d linux.tar.bz2           | Bzip2 is a compression and decompression tool commonly used for linux archives. The -d option decompresses in place, leaving the file without the .bz2 extension.               |
| cat <file>                             | cat myFile.txt                   | Show file contents on console   |
| cd <path>                              | cd /bin/apps                     | Change directory. "." is the next directory up, "." is the current directory.   |
| chown <owner> <files>                  | chown root myFile                | Change the ownership of a file or files (. * works) to the specified user. Sufficient permission is required.   |
| df                                     | df                               | Show free disk space.   |
| du -h                                  | du -h                            | Estimate the bytes consumed by the current directory.   |
| env / printenv                         | env / printenv                   | Display the value of all environment variables on the console   |
| find -name <file> -print               | find -name *.c -print            | Recursively find all C source files starting at the current directory.  |
| ifconfig                               | ifconfig or /sbin/ifconfig       | List the TCP/IP parameters associated with each network device in your system.  |
| ln -s <dir> <link>                     | ln -s linux-2.4.20 linux         | Creates a symbolic link to a directory, to easily reference it from another location. Like a pointer.   |
| ls                                     | ls -a or ls -l                   | List the directory contents with various amounts of information.  |
| man <command>                          | man ls                           | List the documentation on a command, including all options. Once in the man page, hit the spacebar for a new page, and type q to quit.  |
| mount -t <type> <device> <mount point> | mount -t vfat /dev/hda5 /mnt/rfs | Create a mount point to access a FAT device from linux. Since FAT format is readable from both Windows and Linux, this can be used to transfer files between dual-boot systems. |

|  |                                     |   |
|--|-------------------------------------|---|
| <code>ps -dl</code>                    | <code>ps -dl</code>                 | List all running processes on the system.   |
| <code>rm -rf &lt;directory&gt;</code>  | <code>rm -rf myDir</code>           | Recursively delete the entire contents of a non-empty directory, without asking for confirmation for the removal of each file.  |
| <code>rpm -i &lt;file.rpm&gt;</code>   | <code>rpm -i linux.src.rpm</code>   | Installs the contents of an rpm file to your system.  |
| <code>rpm -qpl &lt;file.rpm&gt;</code> | <code>rpm -qpl linux.src.rpm</code> | A .rpm file is a type of self-extracting executable often used to install source patches. This format will dump the contents of the rpm without installing it.  |
| <code>tar -xf &lt;file&gt;</code>      | <code>tar -xf linux.tar</code>      | A tarball is the most common way to compress entire directories in Linux. It is not the most efficient compression, so you will often see a tar file compressed again with another utility. This form will extract the directory to the current location. |
| <code>uname -a</code>                  | <code>uname -a</code>               | Displays the kernel version on the console.   |

## Revision History

| Version | Date          | Author | Details              |
|---------|---------------|--------|----------------------|
| 1.0     | July 25, 2014 | TC     | Xilinx Vivado 2013.4 |
|         |               |        |                      |
|         |               |        |                      |
|         |               |        |                      |