# MicroZed
# Getting Started Guide

Version 1.0
09 August 2013

# Revision History

| Version | Description | Date |
|---|---|---|
| 1.0 | Initial release | 10 Aug 2013 |

# Table of Contents

# Getting Started with MicroZed

The Avnet MicroZed enables hardware and software developers to create or evaluate Zynq™-7000 All Programmable SoC designs.

MicroZed has the unique ability to operate both standalone as well as a system-on-module (SOM). The MicroZed Evaluation Kit includes a standalone MicroZed that contains a fully functional Zynq Processing System (PS) with peripherals as well as enabling the Zynq Programmable Logic (PL) fabric. This PS system includes DDR3 memory, Flash memory, gigabit Ethernet, USB 2.0 Host, and a UART. The capabilities of the MicroZed can be enhanced by plugging it onto a carrier card, which then enables up to 108 I/Os for the user to define.

This *Getting Started Guide* will outline the steps to setup the MicroZed hardware. It documents the procedure to run a simple Linux design to show a Linux application running on the ARM® dual-core Cortex™-A9 MPCore™ Processing System (PS). Xilinx Vivado Design Edition tools are also introduced where the design can be built from scratch and customization options can be discovered. If Xilinx Vivado software is not already installed, further resources to install the software, get updated and generate a license are provided in Appendix C: Installing and Licensing Xilinx Software.

# What's Inside the Box?


## MicroZed Kit contents

- MicroZed
- USB-A to Micro-USB-B cable
- 4GB microSD card
- microSD to SD card adapter
- Software
    - Xilinx Vivado™ Design Edition License Voucher for MicroZed designs
- Documentation
    - Getting Started Instruction card

# What's on the Web?

MicroZed is a community-oriented kit, with all materials being made available through the MicroZed.org community website.

## Official Documentation:

- Schematics
- Bill of materials
- Layout
- Hardware manual
- Board definition files

## Tutorials and Reference Designs:

- Introductory material for beginners
  - Creating a Zynq hardware platform
  - Developing software in SDK
- Design examples
- Community projects

## Training and Videos:

- Overview of MicroZed
- Introduction to Zynq
- Implementing Linux on the Zynq-7000 SoC
- Software Defined Radio on Zynq
- Using XADC on Zynq for Thermal Analysis
- Embedded System Design Flow
- ZynqGeek Blog

# MicroZed Key Features

- Processor
  - Zynq™-7000 AP SoC XC7Z010-CLG400-1
- Memory
  - 1 GB DDR3
  - 128 Mb Quad-SPI Flash
  - 4 GB microSD card
- Communication
  - 10/100/1000 Ethernet
  - USB Host 2.0 and USB-UART
- Expansion connectors
  - 2 MicroHeader connectors (108 single-ended, 48 differential pairs, Agile Mixed Signaling (AMS))
  - Digilent Pmod™ Compatible header (8 MIO)
- Clocking
  - 33.33333 MHz clock source for PS
- Configuration and Debug
  - Xilinx Platform Cable JTAG connector
- General Purpose I/O
  - 1 user LED
  - 1 push button

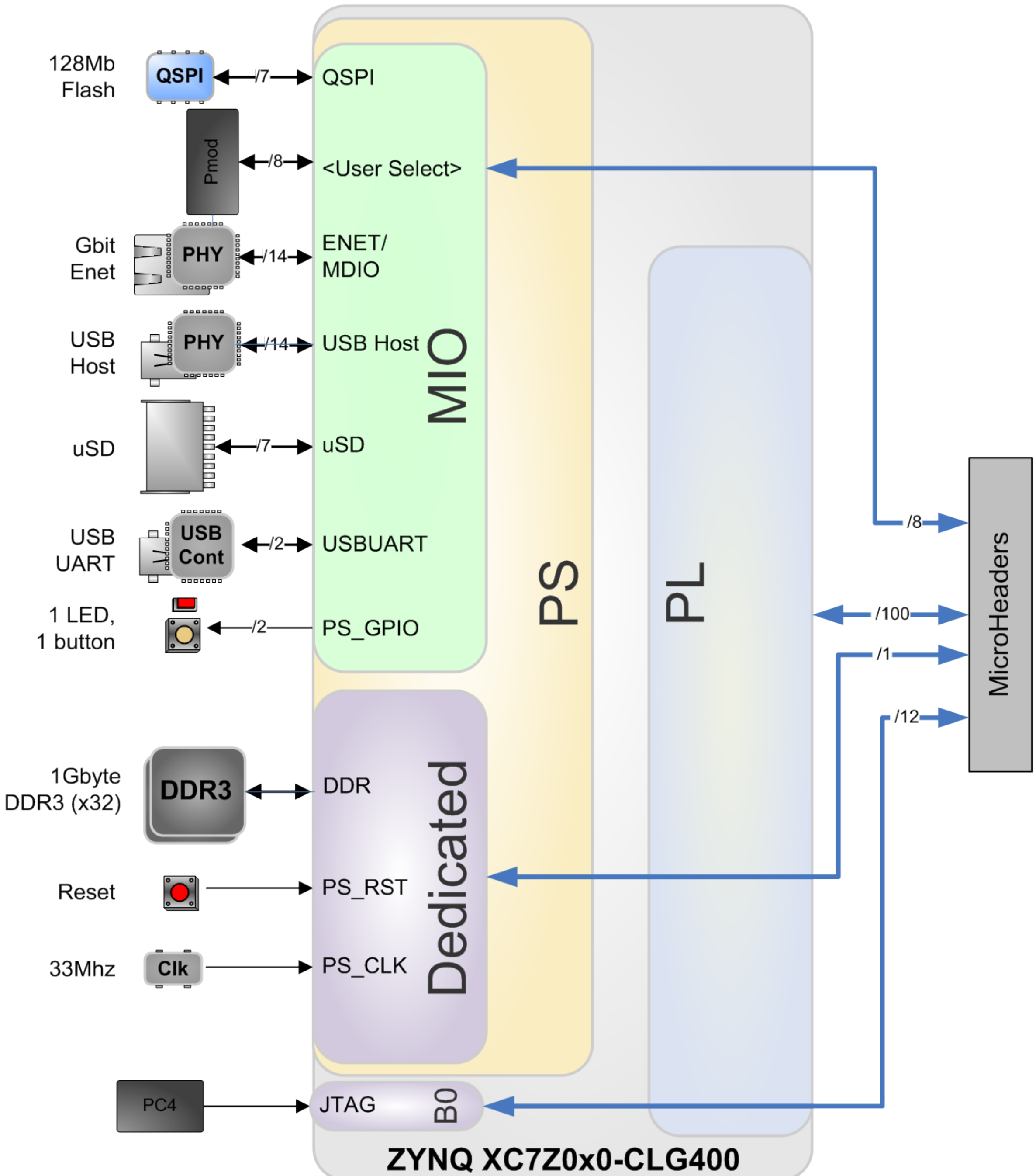



**Figure 1 – MicroZed Block Diagram**

# MicroZed Basic Setup and Operation

The MicroZed QSPI Flash is preloaded with an example open source Linux build with a RAMdisk file system. This document was created using a host PC running Windows 7 and the instructions apply directly to a Windows 7 host PC. It is recommended that the host PC also have a wired (RJ-45 connector) Network Interface Card (NIC) that can operate at 100 Mbps or 1000 Mbps.

This Getting Started Guide offers system developers examples of how to do several things within Linux:

1. Exercise the microSD card
2. Interact with GPIO (LED and push button)
3. Use Ethernet for webserver and file transfer
4. Mount and use a USB memory stick

In addition to the items included in the kit, you will also need a **CAT-5e Ethernet patch cable** and a **USB memory stick** to complete the exercises in this tutorial.

An image of the MicroZed in its expected out-of-box configuration is shown below along with the locations of several key components.

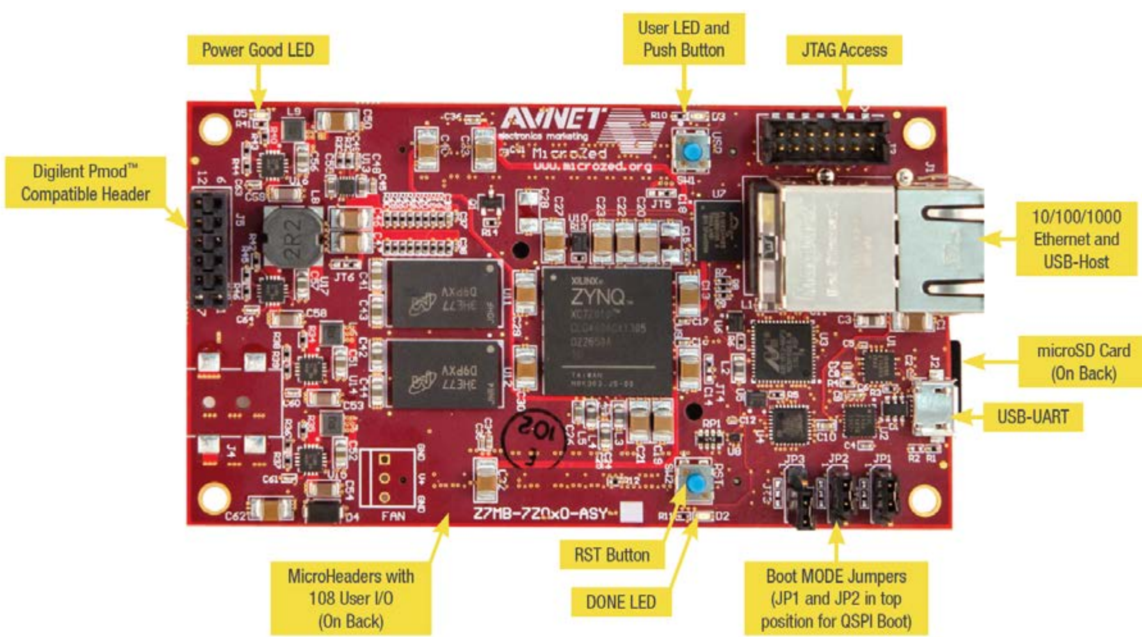



**Figure 2 – MicroZed Component Locations**

## Hardware Setup

1. The included microSD card must be formatted as FAT32. If this has not been previously done, please do that now. Refer to Appendix A: Format the microSD Card for specific instructions.

2. The PC network must be properly configured to communicate with the MicroZed. Refer to Appendix B: Host PC Networking Configuration for instructions to accomplish this.

3. A terminal program is required. Windows 7 does not come pre-installed with a terminal program. Tera Term was used in this example which can be downloaded from the Tera Term project on the SourceForge Japan page: ttssh2.sourceforge.jp Install Tera Term or another terminal program of your choice.

4. If not previously installed, go to www.microzed.org to download and install the Silicon Labs CP2104 USB-to-UART driver. www.microzed.org/sites/default/files/documentations/CP210x_Setup_Guide_1_2.pdf

5. Insert the 4GB microSD card included with MicroZed into the microSD card slot (J6) located on the underside of MicroZed module.

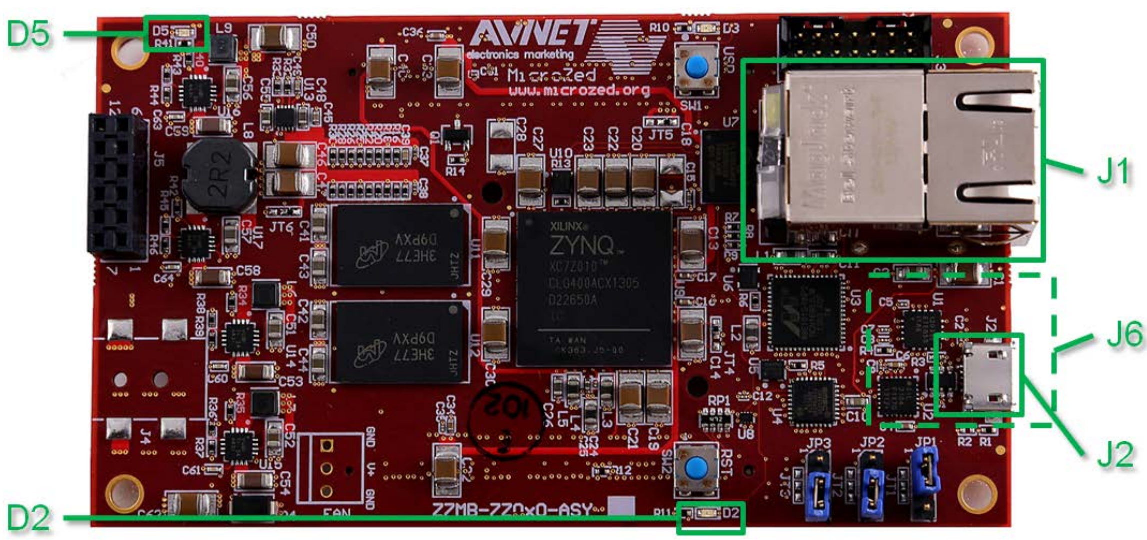



**Figure 3 – MicroZed Hardware Reference**

6. Verify the MicroZed boot mode (JP3-JP1) jumpers are set to SD card mode as described in the Hardware Users Guide.

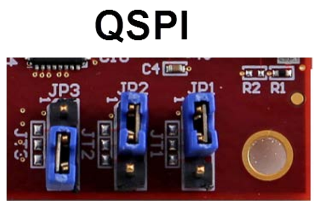



**Figure 4 – MicroZed Jumper Settings**

7. Connect the Ethernet cable between the MicroZed (J1) and the PC.

8. Connect the USB-UART port of MicroZed (J2) to a PC using the MicroUSB cable. MicroZed will power on and the Green Power Good LED (D5) should illuminate.

9. Wait approximately 7 seconds. The blue Done LED (D2) should illuminate.

10. On the PC, open a serial terminal program. Tera Term is used to show the example output for this lab document. Follow the instructions in the CP210x Setup Guide to set the terminal as shown in Figure 5.

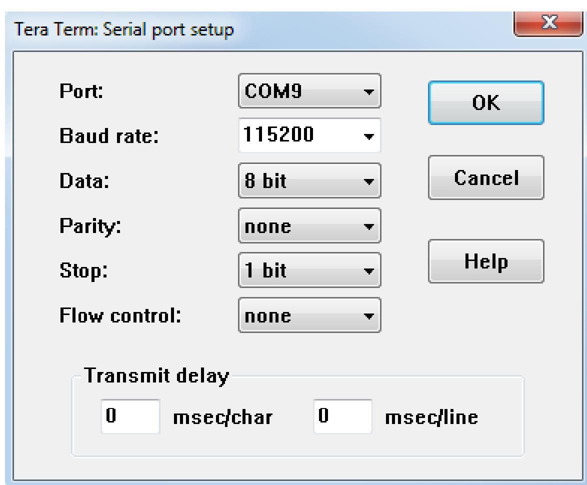



**Figure 5 – Connect Tera Term to the proper COMx port**

11. Reset the processor by pressing and releasing the RST button (SW2).

12. When the terminal output from U-Boot and a countdown is observed, allow the countdown to expire.

A successful boot is shown in the next figure.

```
COM9:115200baud - Tera Term VT
File  Edit  Setup  Control  Window  Help
console [ttyPS0] enabled, bootconsole disabled
xdevcfg f8007000.ps7-dev-cfg: ioremap f8007000 to f0062000 with size 1000
brd: module loaded
loop: module loaded
xqspips e000d000.ps7-qspi: master is unqueued, this is deprecated
m25p80 spi0.0: found s25fl129p1, expected n25q128
m25p80 spi0.0: s25fl129p1 (16384 Kbytes)
5 ofpart partitions found on MTD device spi0.0
Creating 5 MTD partitions on "spi0.0":
0x000000000000-0x000000100000 : "qspi-fsbl-uboot"
0x000000100000-0x000000600000 : "qspi-linux"
0x000000600000-0x000000620000 : "qspi-device-tree"
0x000000620000-0x000000c00000 : "qspi-rootfs"
0x000000c00000-0x000001000000 : "qspi-bitstream"
xqspips e000d000.ps7-qspi: at 0xE000D000 mapped to 0xF0064000, irq=51
e1000e: Intel(R) PRO/1000 Network Driver - 2.1.4-k
e1000e: Copyright(c) 1999 - 2012 Intel Corporation.
libphy: XEMACPS mii bus: probed
xemacps e000b000.ps7-ethernet: pdev->id -1, baseaddr 0xe000b000, irq 54
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
ehci-pci: EHCI PCI platform driver
ULPI transceiver vendor/product ID 0x0424/0x0007
Found SMSC USB3320 ULPI transceiver.
ULPI integrity check: passed.
xusbps-ehci xusbps-ehci.0: Xilinx PS USB EHCI Host Controller
xusbps-ehci xusbps-ehci.0: new USB bus registered, assigned bus number 1
xusbps-ehci xusbps-ehci.0: irq 53, io mem 0x00000000
xusbps-ehci xusbps-ehci.0: USB 2.0 started, EHCI 1.00
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
mousedev: PS/2 mouse device common for all mice
i2c /dev entries driver
xadcps f8007100.ps7-xadc: Failed to request memory region
sdhci: Secure Digital Host Controller Interface driver
sdhci: Copyright(c) Pierre Ossman
sdhci-pltfm: SDHCI platform and OF driver helper
mmc0: SDHCI controller on e0100000.ps7-sdio [e0100000.ps7-sdio] using ADMA
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
TCP: cubic registered
NET: Registered protocol family 17
VFP support v0.3: implementor 41 architecture 3 part 30 variant 9 rev 4
Registering SWP/SWPB emulation handler
dmatest: Started 1 threads using dma0chan0
dmatest: Started 1 threads using dma0chan1
dmatest: Started 1 threads using dma0chan2
dmatest: Started 1 threads using dma0chan3
dmatest: Started 1 threads using dma0chan4
dmatest: Started 1 threads using dma0chan5
dmatest: Started 1 threads using dma0chan6
dmatest: Started 1 threads using dma0chan7
mmc0: new high speed SDHC card at address 1234
mmcblk0: mmc0:1234 SA04G 3.63 GiB
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
 mmcblk0: p1
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
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 ssh daemon
rcS Complete
zynq> xemacps e000b000.ps7-ethernet: Set clk to 124999998 Hz
xemacps e000b000.ps7-ethernet: link up (1000/FULL)
```

**Figure 6 – MicroZed U-Boot Booting Linux**

# File System

1. This Linux image creates a file system on the DDR3 on MicroZed. Basic Linux commands are available as you might expect on any linux system. CD into the /bin directory.

```
zynq> cd /bin/
```

2. Check the current working directory by typing the command below

```
zynq> pwd
```

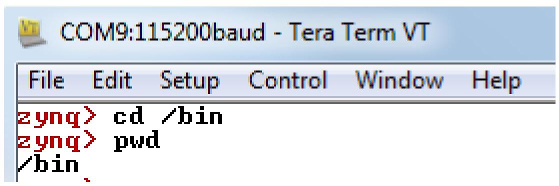



**Figure 7 – Print Working Directory**

3. List the contents of /mnt by typing the command below

```
zynq> ls
```


**Figure 8 – List Contents**

4. To see full details, use the command below

```
zynq> ls -l
```

```
zynq> ls -l
total 916
lrwxrwxrwx    1 root     root             7 Nov 27  2012 addgroup -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 adduser -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 ash -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 base64 -> busybox
-rwsr-xr-x    1 root     root        868364 Nov 27  2012 busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 cat -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 catv -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 chattr -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 chgrp -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 chmod -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 chown -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 cp -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 cpio -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 cttyhack -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 date -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 dd -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 delgroup -> busybox
lrwxrwxrwx    1 root     root             7 Nov 27  2012 deluser -> busybox
```

**Figure 9 – Detailed List Contents**

5. To see file sizes, use the command du

```
zynq> du *
```

```
zynq> du *
0       addgroup
0       adduser
0       ash
0       base64
854     busybox
0       cat
0       catv
0       chattr
```

**Figure 10 – Disk Usage**

6. To see how much free disk space is available, use the command df

```
zynq> df
```

```
zynq> df
Filesystem           1K-blocks      Used Available Use% Mounted on
none                    516800         0    516800   0% /tmp
/dev/mmcblk0p1         3801088         4   3801084   0% /mnt
zynq>
```

**Figure 11 – Disk Free**

7. To find a file in the file system, use the command 'find'. The command below searches from the root directory looking for a file called "iperf".

```
zynq> find / -name "iperf"
```

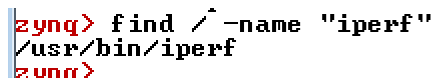



**Figure 12 – Find a File**

8. In the case with two executables with the same name, it might be useful to know which one is found without explicitly spelling out the path. Command 'which' will tell you the path of the executable to be run. Cd to the root directory then test if iperf is in the path.

```
zynq> cd /
zynq> which iperf
```

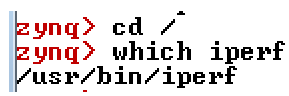



**Figure 13 – Which**

A short list of several more useful file- and directory-oriented commands include:

- mkdir
- rmdir
- rm
- chmod
- cp
- mv
- less <file>

# Interact with GPIO (LED and push button)

With MicroZed booted to the Linux command prompt, the MIO GPIO hardware can be accessed directly via the generic sysfs GPIO driver.

1. From the Linux command prompt, take a look at the GPIO driver class within **/sys** subfolders.

   Notice how the GPIO driver exports controls via sysfs. Here we see that GPIOs are available for export via the export property.

```
$ ls /sys/class/gpio/
```

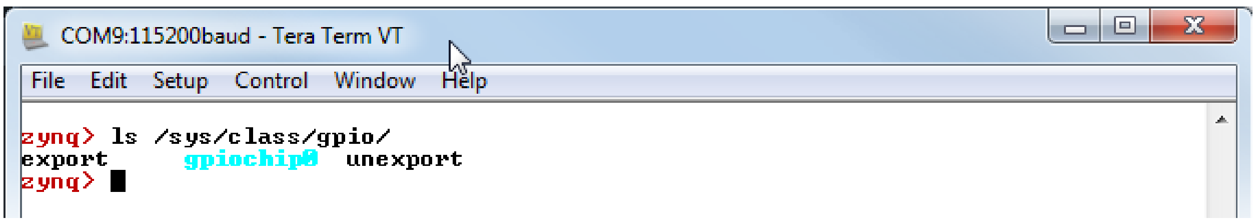



**Figure 14 – Exploring the Sysfs Subsystem**

2. Take a look at the MicroZed schematic and determine which IO pin the MIO LED D3 (sheet 5) is connected to.

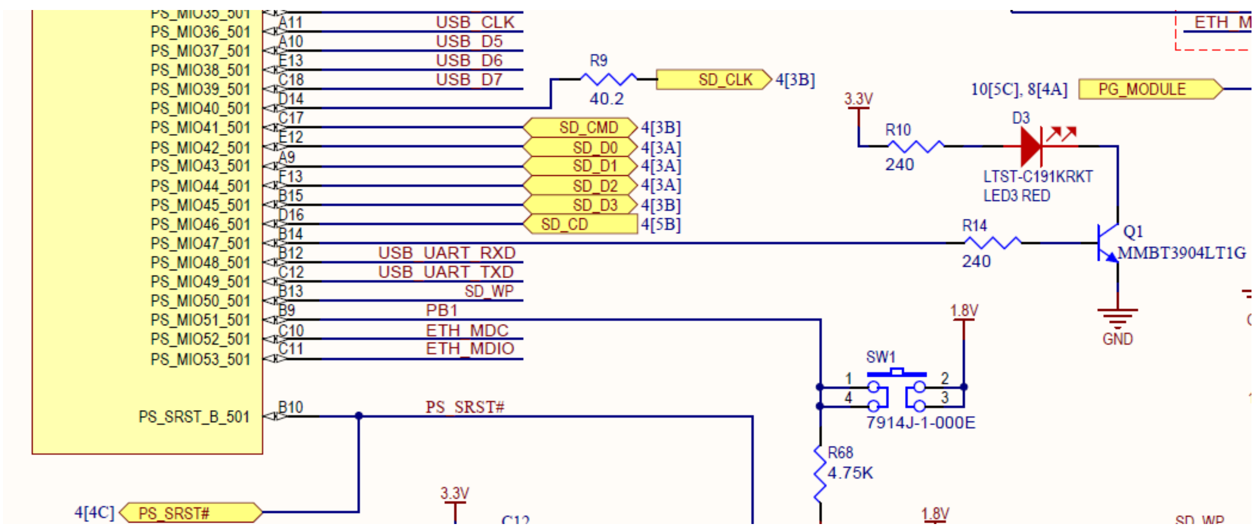



**Figure 15 – MicroZed Schematic Snippet Relating to MIO LED D3**

3. In looking at the schematic, you should have determined that the MIO LED **D3** is connected to pin **B14** which corresponds to **PS_MIO47**. Using MIO number 4**7**, export the corresponding GPIO device to the sysfs file system so that the GPIO controls for **PS_MIO47** can be used.

   This is done by using the echo command to send the number **47** to the gpio device class **export** property.

   Then evaluate the GPIO folder again to verify that the new **gpio47** device has been exported to the sysfs file system.

```
$ echo 47 > /sys/class/gpio/export
$ ls /sys/class/gpio/
```

   Notice that the export property has caused the gpio47 node to become available. Behind the scenes, the GPIO driver received a write call and used the 47 parameter entry to determine which GPIO channel to enable and export control properties for. In the next steps, we will explore the function of the properties of the newly enabled **gpio47** node.


**Figure 16 – Exporting GPIO47 Controls Via the Sysfs Subsystem**

4. Evaluate the new **gpio47** node that was exported in the previous step.

```
$ ls /sys/class/gpio/gpio47
```

   Notice that this node contains several properties which would normally be associated with a GPIO control.

   Two of these properties are useful for this lab: the **direction** property and the **value** property.

   The **direction** property is writable and controls whether the GPIO driver configures the controller for input or output. This property can be assigned either an **in** value or an **out** value.

The **value** property is read/writable and reflects either the output logic state of the GPIO when the **direction** property is set to **out** or reflects the input logic state of the GPIO when the **direction** property is set to **in**.

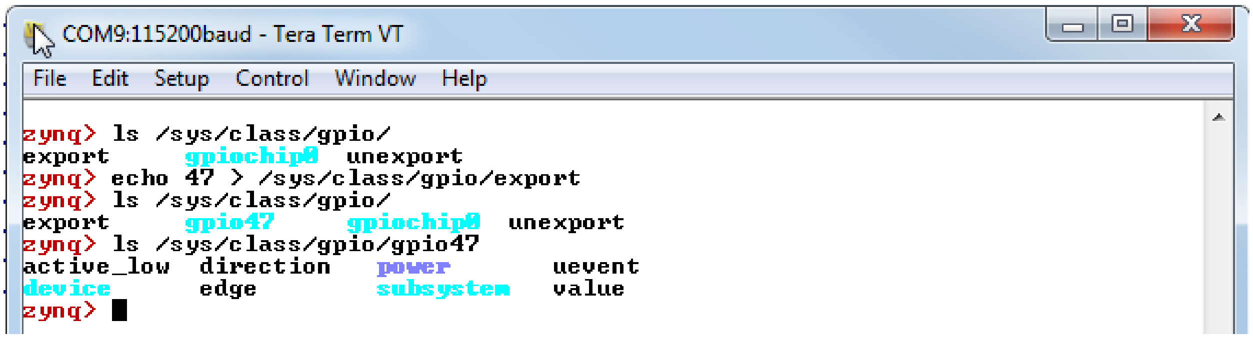



**Figure 17 – GPIO47 Control Properties Via the Sysfs Subsystem**

5. Modify the direction property of the gpio47 node and set it to an output.

```
$ echo out > /sys/class/gpio/gpio47/direction
```

6. Modify the value property of the gpio47 node and watch the MicroZed D3 LED as the command input is entered.

```
$ echo 1 > /sys/class/gpio/gpio47/value
```

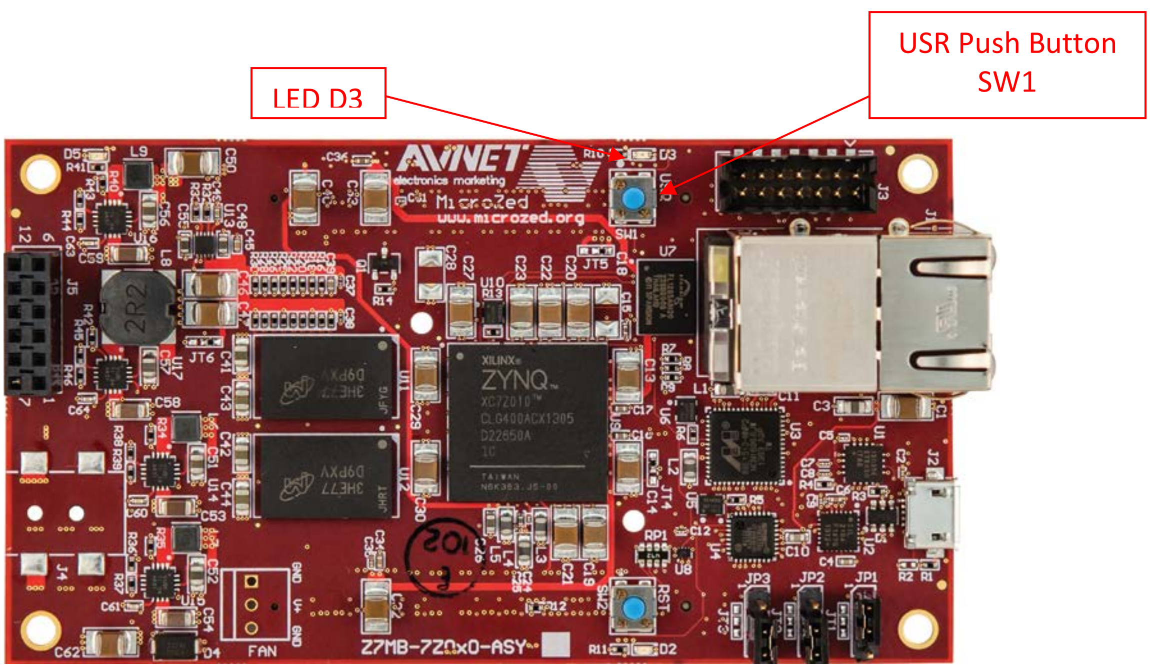



**Figure 18 – MicroZed LED and Push Button**

Did you observe a change in state on D3 LED?

Modify the value property of the gpio47 node again and watch the MicroZed D3 LED as the command input is entered.

```
$ echo 0 > /sys/class/gpio/gpio47/value
```

7. Continue experimenting with different inputs to the value. Which values are accepted, and which are ignored? How effective do you think it would be to implement a PWM control on this output using only software timing?

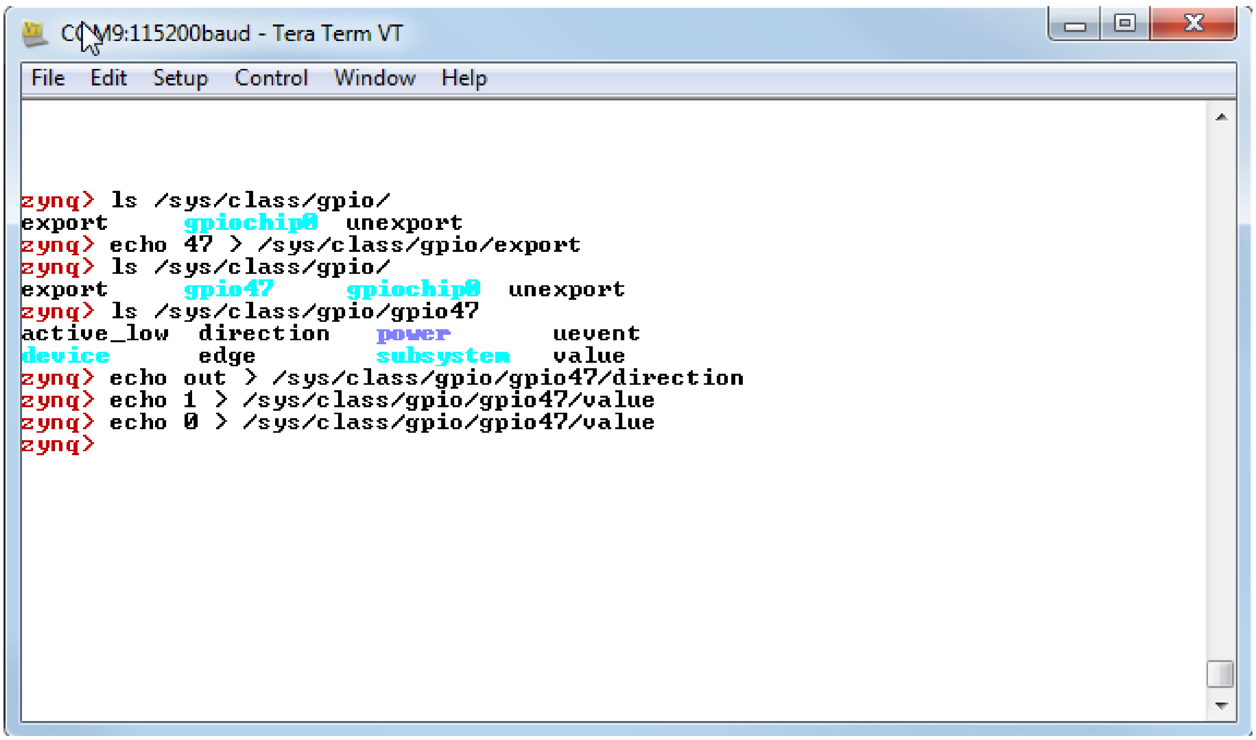



**Figure 19 – Modifying the GPIO47 value Property**

8. Perform a similar exercise using MIO push button **SW1** as an input device. Take a look at the MicroZed schematic (or Figure 15) and determine which IO pin the MIO push button **SW1** is connected to.

9. In looking at the schematic, you should have determined that the MIO push button **SW1** is connected to pin **B9** corresponding to **PS_MIO51**. Using this MIO number, export the corresponding GPIO device for use and evaluate the GPIO folder again.

```
$ echo 51 > /sys/class/gpio/export
```

10. Modify the direction property of the **gpio51** node and set it to input.

```
$ echo in > /sys/class/gpio/gpio51/direction
```

11. Read the value property of the **gpio51** node.

```
$ cat /sys/class/gpio/gpio51/value
```

12. Using the up arrow key on the keyboard to repeat a command in the command line history, repeat the above command while pressing the MIO push button. Did you observe a change in state of the value property read from the push button?

13. Continue experimenting with reading the different input states from the value properties. How effective do you think it would be to poll the push buttons for changes in state?

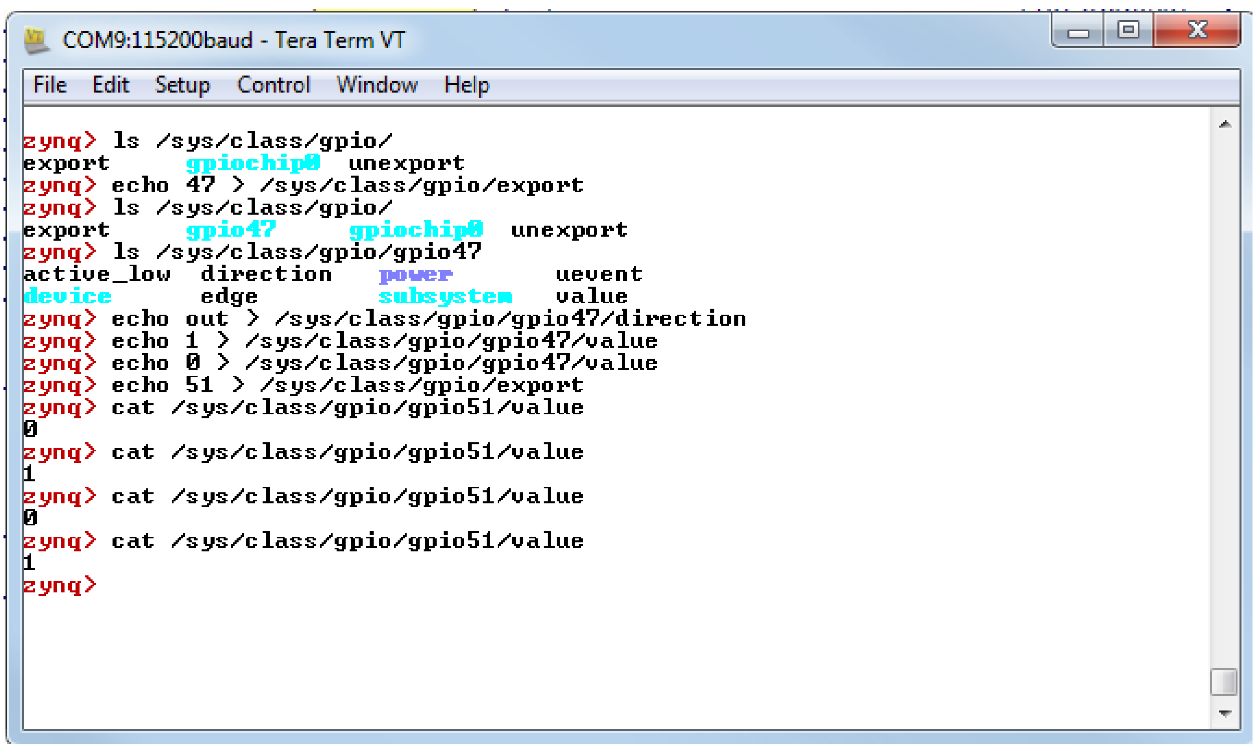



**Figure 20 – Reading the GPIO51 value Property**

14. Think how you might use the button to control the LED. When the button is pushed, it produces a '1' and when not pushed a '0'. Lighting the LED requires that you send it a '1' and to turn it off a '0'.

    Turn off the LED. Then, while holding down the push button, enter the command below.

```
$ echo 0 > /sys/class/gpio/gpio47/value
<now hold down the push button>
$ cat /sys/class/gpio/gpio51/value > /sys/class/gpio/gpio47/value
<now let off the push button>
$ cat /sys/class/gpio/gpio51/value > /sys/class/gpio/gpio47/value
```

15. Now create a script with an infinite loop that does this continuously. If you are comfortable using the vi editor, feel free to do so. Otherwise, the following set of commands will also do the job to create script `pb_lights_led.sh`.

```
$ cd /
$ echo while : > pb_lights_led.sh
$ echo do >> pb_lights_led.sh
$ echo “cat /sys/class/gpio/gpio51/value >
/sys/class/gpio/gpio47/value” >> pb_lights_led.sh
$ echo done >> pb_lights_led.sh
$ chmod 755 pb_lights_led.sh
$ ./pb_lights_led.sh
```

16. Hit Ctrl-C in the terminal window after you have enjoyed the satisfaction of seeing the LED light whenever you push the button.

# Ethernet Operations

The MicroZed example Linux system implements a Dropbear SSH server, ftpd FTP server, and Busybox httpd HTTP server at startup. Refer to the documentation on each of these server implementations if you are interested in using them beyond the scope of this document.

1. The default IP address of MicroZed Ethernet is set to 192.168.1.10. This can be verified with the output returned by the `ifconfig` command.

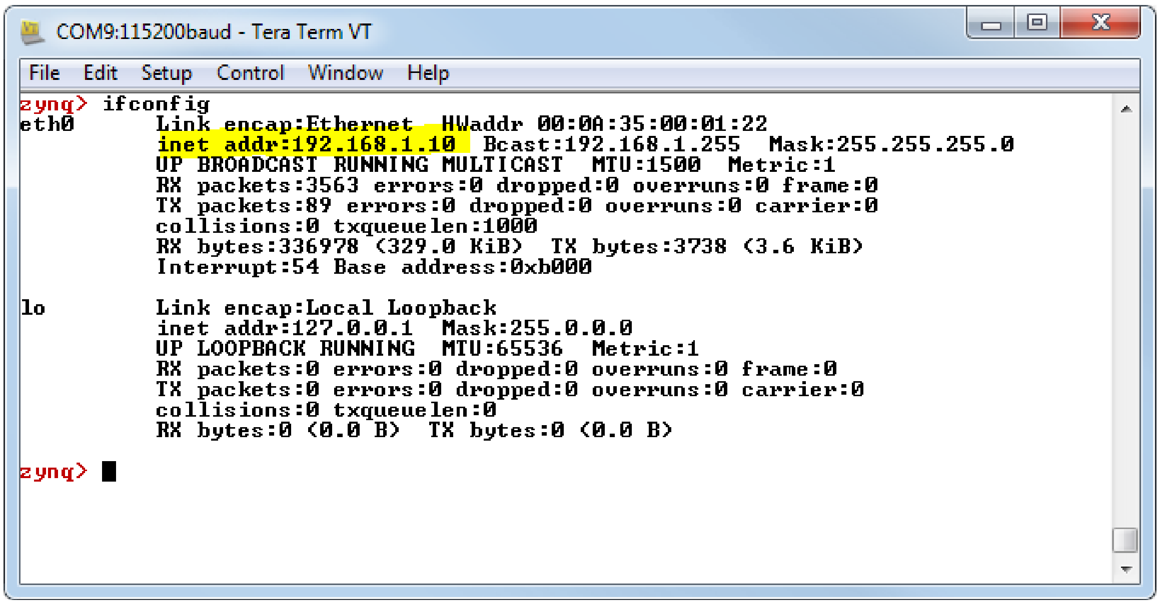



**Figure 21 – MicroZed IP Address Revealed with ifconfig Command**

2. The most simple connectivity test is to use the 'ping' command. Try pinging your laptop with the following command (assuming you used the address given in the setup section of this document). Hit Ctrl-C when you are satisfied.

```
zynq>  ping 192.168.1.100
```

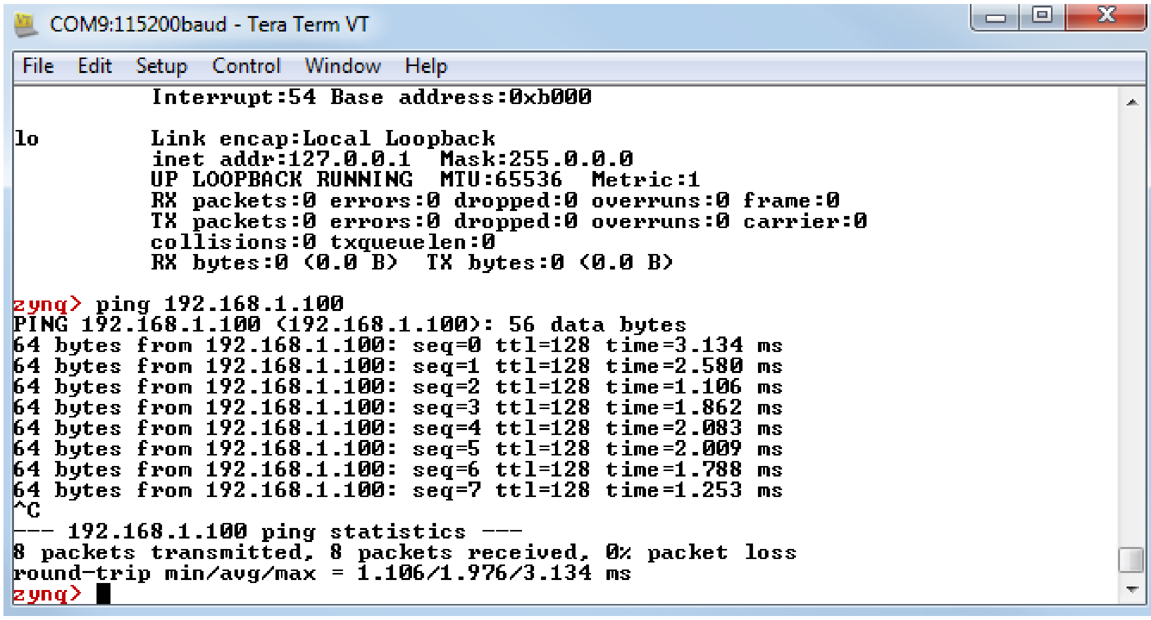



**Figure 22 – Ping the Laptop**

3. Likewise, you can ping the MicroZed from the Laptop. Open a Windows command prompt, and enter command 'ping 192.168.1.10'

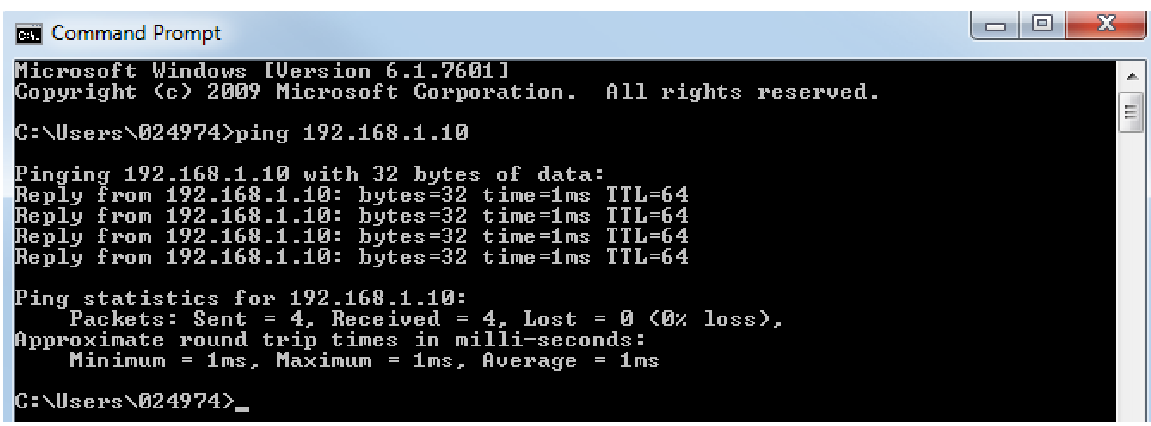



**Figure 23 – Ping the MicroZed**

4. To view the MicroZed embedded webpage, open a web browser (such as Firefox) and browse to the MicroZed IP address [http://192.168.1.10/](http://192.168.1.10/) as the URL. The MicroZed webpage should open in the browser. This is the default webserver provided through the Xilinx distribution. Note that many of the links point to internal Xilinx sites so they aren't all operational.

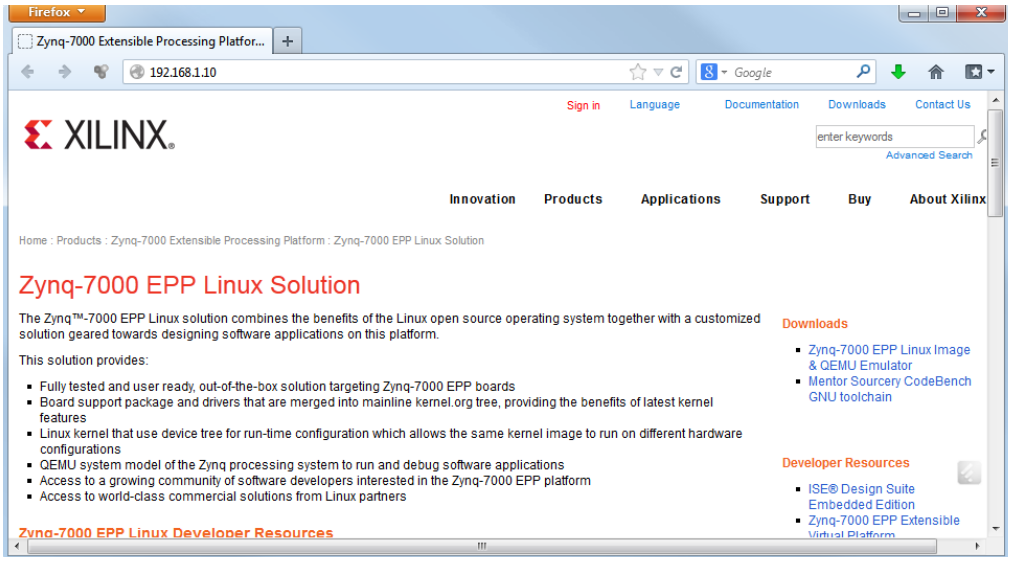



**Figure 24 –** MicroZed **Webpage Shown In PC Host Browser**

5. Using an SSH client, you can open a secure terminal connection to the target MicroZed using the 192.168.1.10 IP address. In Tera Term, select **File → New Connection**.
6. Select the radio button for *TCP/IP*.
7. Under *Service*, select the radio button for **SSH**.
8. Uncheck the *History* box.
9. In the Host: dialog, type '192.168.1.10' then click **OK**.

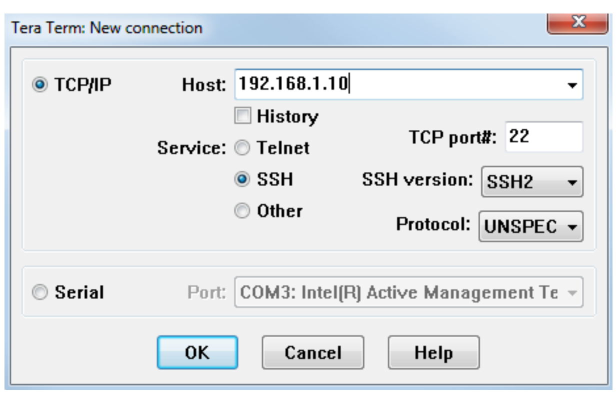



**Figure 25 – Setup SSH Connection in Tera Term**

10. A Security Warning may pop up. Click **Continue**.

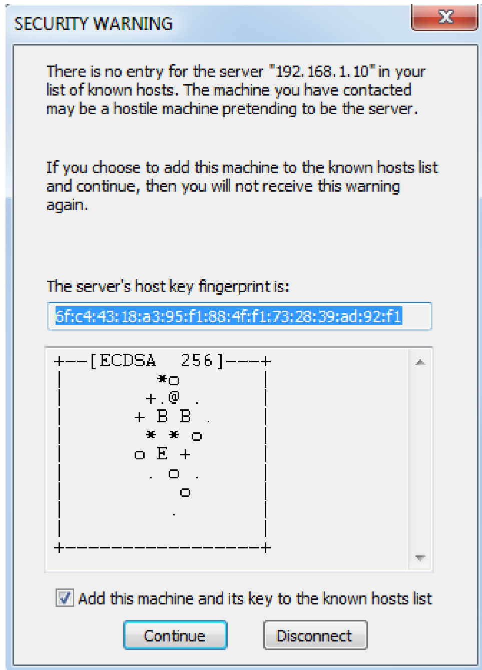



**Figure 26 – Tera Term SSH Security Warning**

11. Once the terminal connects, the remote system will prompt for a login. Use the user name **root** and the passphrase **root** to complete the connection.

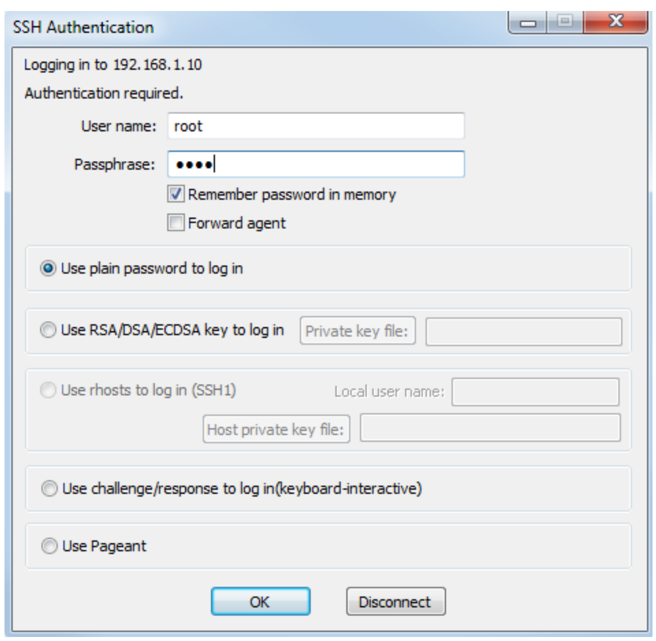



**Figure 27 – Login as root**

12. The session acts as a remote terminal and commands can be entered as you would on the local serial console, like ‘pwd’ or ‘ls’ or ‘cd’

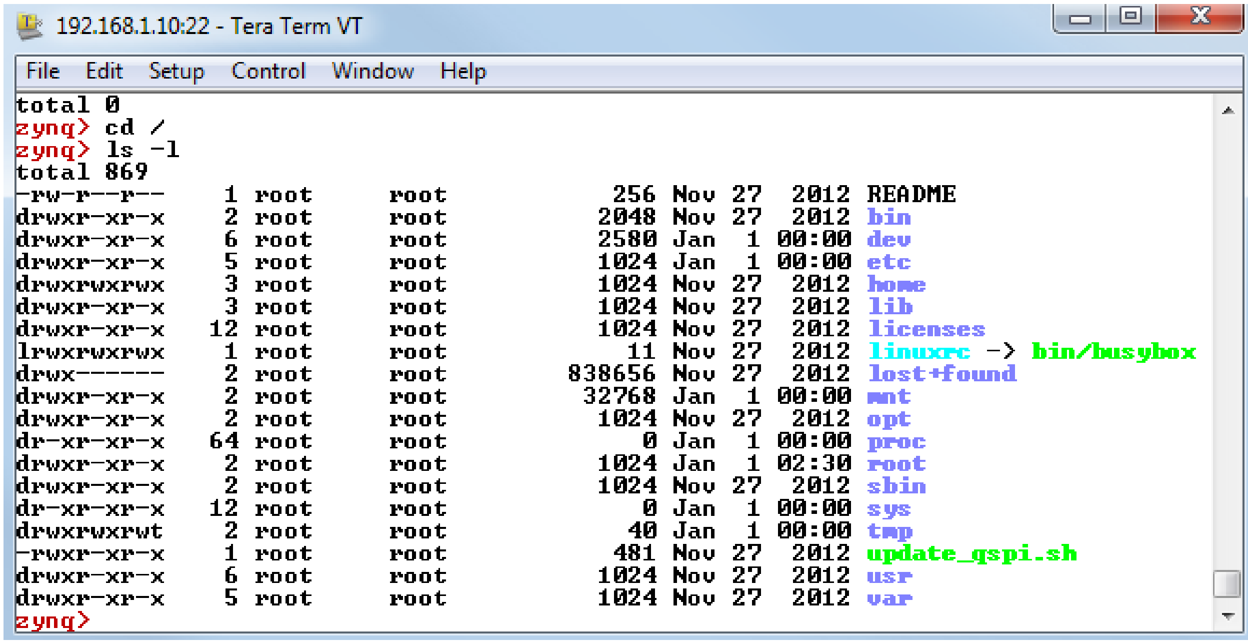
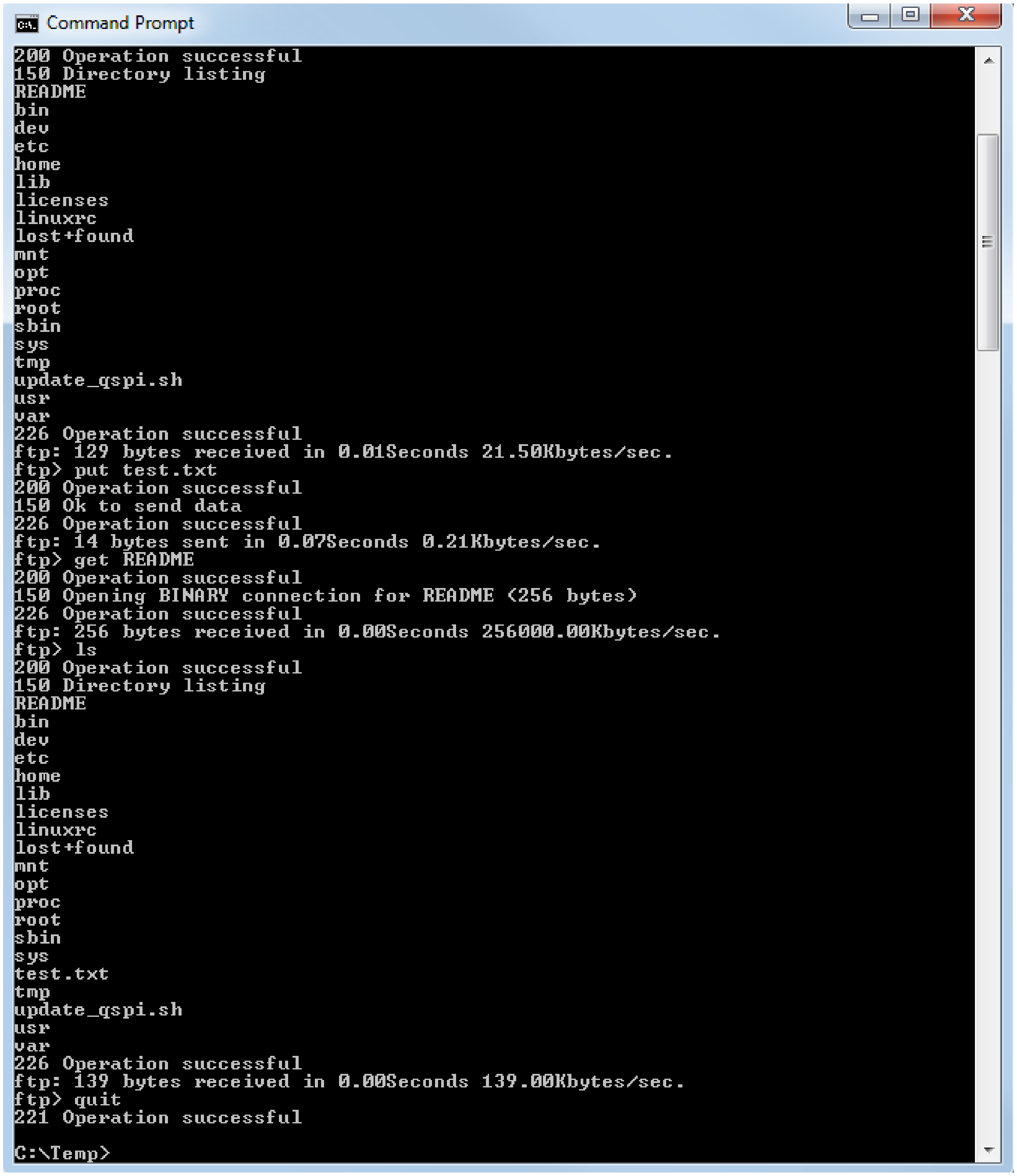



**Figure 28 – Remote MicroZed Terminal via SSH Session**

13. Logout and close the remote session with the `exit` command.
14. Open a Windows Command Prompt.
15. Connect an FTP session to the remote host with the command `ftp 192.168.1.10` and use the login **root**. You can use the ftp session to transfer files back and forth across the network to MicroZed.
16. Close the ftp session using the `quit` command.



```
Command Prompt
200 Operation successful
150 Directory listing
README
bin
dev
etc
home
lib
licenses
linuxrc
lost+found
mnt
opt
proc
root
sbin
sys
tmp
update_qspi.sh
usr
var
226 Operation successful
ftp: 129 bytes received in 0.01Seconds 21.50Kbytes/sec.
ftp> put test.txt
200 Operation successful
150 Ok to send data
226 Operation successful
ftp: 14 bytes sent in 0.07Seconds 0.21Kbytes/sec.
ftp> get README
200 Operation successful
150 Opening BINARY connection for README (256 bytes)
226 Operation successful
ftp: 256 bytes received in 0.00Seconds 256000.00Kbytes/sec.
ftp> ls
200 Operation successful
150 Directory listing
README
bin
dev
etc
home
lib
licenses
linuxrc
lost+found
mnt
opt
proc
root
sbin
sys
test.txt
tmp
update_qspi.sh
usr
var
226 Operation successful
ftp: 139 bytes received in 0.00Seconds 139.00Kbytes/sec.
ftp> quit
221 Operation successful

C:\Temp>
```

**Figure 29 – MicroZed FTP Session**

# USB-Host and microSD Card

This demo shows how a high speed USB communications peripheral connected to the Processing System (PS) of Zynq-7000 AP SoC can be used to extend the functionality of MicroZed. The MicroZed USB 2.0 is designed as Host only. With a bit of simple rework, it can be modified to be either Device or OTG as well.

At the same time, the microSD card will be mounted and exercised.

Since MicroZed is powered from the USB-UART, there is only a very limited amount of power to share with the USB-Host port. Additionally, there is only one port. To connect multiple USB devices with the MicroZed or to connect higher power USB peripherals, connect a powered hub to the USB-Host port. USB devices attached to this hub can then also be accessed in Linux.

1. Connect the USB memory stick to your PC. Format as FAT32 or NTFS. Create a simple text file on the memory stick then eject from the PC.
2. Connect the USB memory stick to the MicroZed Type A USB connector underneath the RJ45 on J1.
3. The USB memory stick should enumerate and the device indication should display on the serial console. Two examples are shown below. In Figure 30, the primary partition of the USB memory stick is enumerated as device `/dev/sda`. In Figure 31

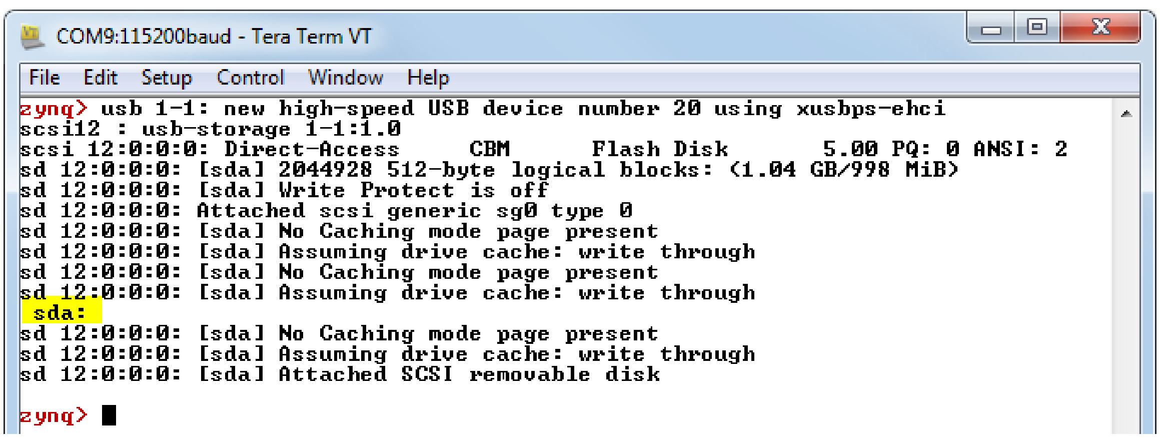



**Figure 30 – USB Drive Enumerated as /dev/sda**

```
COM9:115200baud - Tera Term VT
File  Edit  Setup  Control  Window  Help
zynq> usb 1-1: new high-speed USB device number 21 using xusbps-ehci
scsi13 : usb-storage 1-1:1.0
scsi 13:0:0:0: Direct-Access     Hisun    Flash Disk       2.10 PQ: 0 ANSI: 2
sd 13:0:0:0: [sda] 1030136 512-byte logical blocks: (527 MB/502 MiB)
sd 13:0:0:0: [sda] Write Protect is off
sd 13:0:0:0: Attached scsi generic sg0 type 0
sd 13:0:0:0: [sda] No Caching mode page present
sd 13:0:0:0: [sda] Assuming drive cache: write through
sd 13:0:0:0: [sda] No Caching mode page present
sd 13:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 13:0:0:0: [sda] No Caching mode page present
sd 13:0:0:0: [sda] Assuming drive cache: write through
sd 13:0:0:0: [sda] Attached SCSI removable disk

zynq>
```

**Figure 31 – USB Drive Enumerated as /dev/sda1**

4. The default Linux image mounts the SD Card at /mnt. First, we will unmount the SD Card with the following commands.

```
zynq> cd /
zynq> umount /mnt
```

5. Use ‘df’ to see that the device at /mnt is no longer there.

```
zynq> df
```

```
zynq> df
Filesystem           1K-blocks      Used Available Use% Mounted on
none                    516800         0    516800   0% /tmp
zynq>
```

**Figure 32 – Nothing Mounted at /mnt**

6. Now, we will create mount points for both the memory stick and the sdcard

```
zynq> cd /mnt
zynq> mkdir memstick
zynq> mkdir sdcard
```

7. Now re-mount the SD card and check to see if it mounted properly.

```
zynq> mount /dev/mmcblk0p1 /mnt/sdcard/
zynq> df
```

```
zynq> mount /dev/mmcblk0p1 /mnt/sdcard/
zynq> df
Filesystem           1K-blocks      Used Available Use% Mounted on
none                    516800         0    516800   0% /tmp
/dev/mmcblk0p1         3801088         4   3801084   0% /mnt/sdcard
zynq>
```

**Figure 33 – SD Card Successfully Mounted**

8. Mount the enumerated USB device to the `/mnt/memstick` mount point and check the contents. Depending on what you saw on the screen (sda or sda1), you will need to select the appropriate commands below. In this example, the memory stick has two files that were previously copied to the memory stick.

***For /dev/sda***

```
zynq> mount /dev/sda /mnt/memstick
zynq> ls /mnt/memstick
```

***For /dev/sda1***

```
zynq> mount /dev/sda1 /mnt/memstick
zynq> ls /mnt/memstick
```

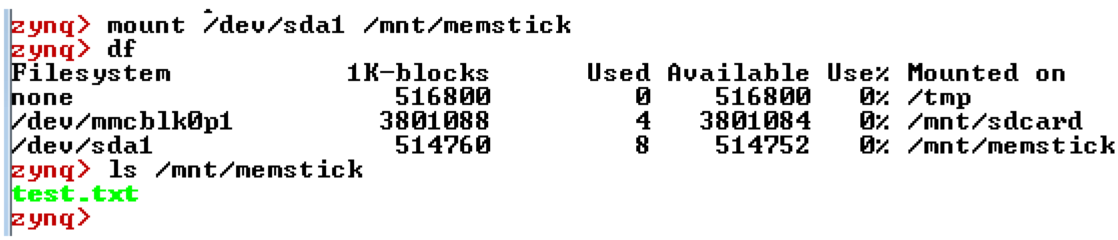



**Figure 34 – SD Card Successfully Mounted**

The microSD and USB drive are now mounted into the root file system at the mount points which enables read and write file operations to the device's file system.

9. Print the contents of a text file to test reading from the file system.

```
zynq> cd /mnt/memstick
zynq> cat test.txt
```

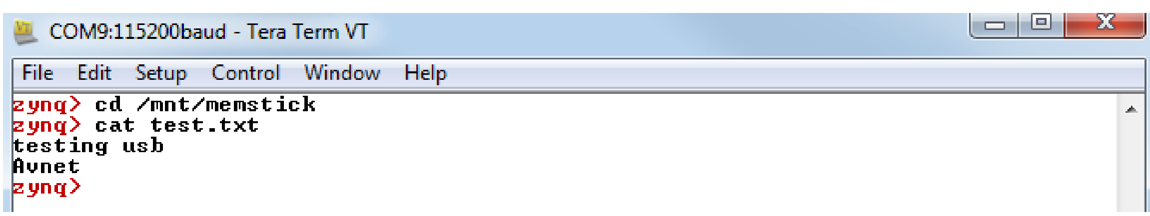



**Figure 35 – Reading a Text File from Memory Stick**

10. Now we'll test writing to the memory stick by creating a new text file. A Linux editor such as vi is fully functional on this system. You can use vi if you are comfortable. Otherwise, use the command below to write the file. Then print it back to make sure it worked.

```
zynq> echo "MicroZed is Awesome" > new.txt
zynq> ls
```

```
zynq> cat new.txt
```

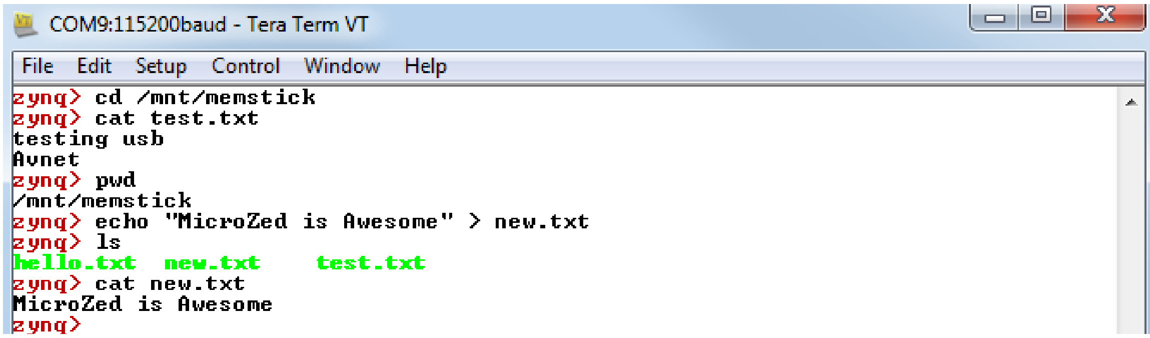



**Figure 36 – Writing a Text File to a Memory Stick**

11. The device should be cleanly un-mounted from the system before it is removed or the board powered off.

```
zynq> cd /mnt
zynq> umount memstick
```

Note: If the device cannot be un-mounted or if a “Device or resource busy” message is shown, make sure that no files or folders of the mounted file system are currently open or that the current working directory is not part of the mounted file system.

12. Remove the memory stick. Plug it into the PC and verify the `new.txt` file is there.

13. Repeat steps 10 through 12 for the microSD card and mount point /mnt/sdcard.

All of the previous experiments can be repeated after booting Linux from the microSD card. Refer to Appendix E: Boot MicroZed from the microSD Card for more information.

# Getting Help and Support

## Avnet Support

MicroZed is a versatile development kit, with all technical support being offered through the MicroZed.org website support forums.  MicroZed users are encouraged to participate in the forums and offer help to others when possible.

For questions regarding the MicroZed community website, please direct any questions to:

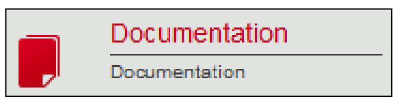
MicroZed.org Web Master – webmaster@microzed.org

To access the most current collateral for MicroZed please visit the community support page at:

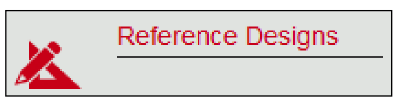
www.microzed.org/content/support

Once on the MicroZed.org support page:

To access the latest MicroZed documentation, click on the Documentation link:




To access the latest reference designs for MicroZed, click on the following link:




To access the MicroZed technical forums, click on the following link:


## Xilinx Support

For questions regarding products within the Product Entitlement Account, send an e-mail message to the Customer Service Representative in your region:

Canada, USA and South America - isscs_cases@xilinx.com

Europe, Middle East, and Africa - eucases@xilinx.com

Asia Pacific including Japan - apaccase@xilinx.com

For technical support including the installation and use of the product license file, contact Xilinx Online Technical Support at www.xilinx.com/support. The following assistance resources are also available on the website:

- Software, IP and documentation updates
- Access to technical support web tools
- Searchable answer database with over 4,000 solutions
- User forums

# Appendix A: Format the microSD Card

The MicroZed Evaluation Kit ships with a blank microSD card. To ensure it is ready to be used in Linux and later as a boot source, it must be properly formatted. To use the microSD card as a boot device, it must be formatted as FAT32.

The following operations were performed on a Windows 7 PC using a built-in SD Card slot. If an SD Card slot is not available on your PC, you will need to purchase an SD Card device or a USB-based microSD adapter.

1. Insert the microSD card into the included SD Adapter.
2. Insert the SD adapter into the SD Card slot 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.

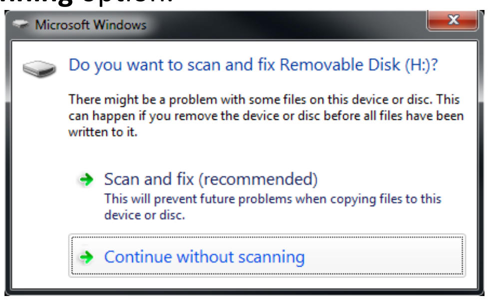



**Figure 37 – Windows Prompt for Scanning and Fixing an SD Card**

3. Find the assigned SD Drive in Windows Explorer.
4. Right-click and select **Format**.

5. Select the *File System* to be FAT32. The *Allocation unit size* can be set to **Default**. Click **Start**.

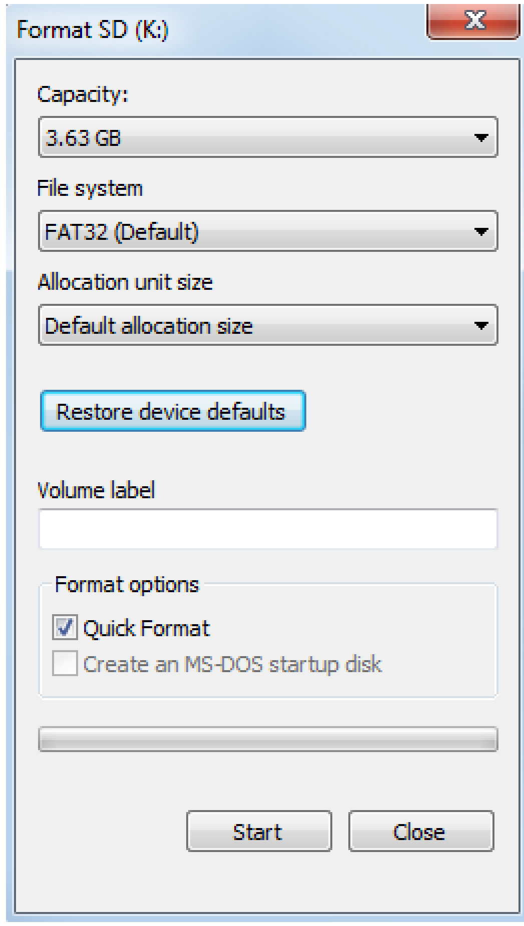



**Figure 38 – Format the microSD Card**

6. As stated in the warning dialog, formatting will erase all data on the disk. Click **OK**.

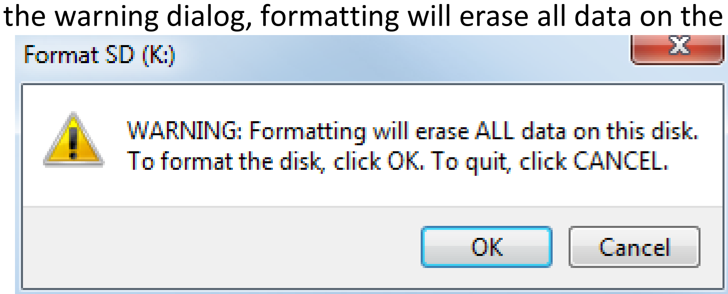



**Figure 39 – Formatting Will Erase**

7. If all goes well, you will get a message stating **Format Complete**. Click **OK**.

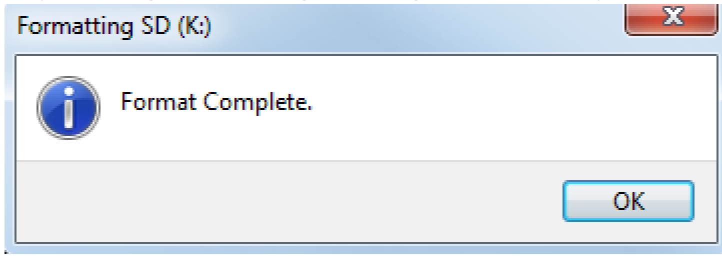



**Figure 40 – Format Complete**

8. Click **Close** in the Format dialog box.
9. To double-check your card, right-click on the drive in Windows Explorer and select **Properties**. Notice the *File system* displayed as **FAT32**. Click **OK** to close.

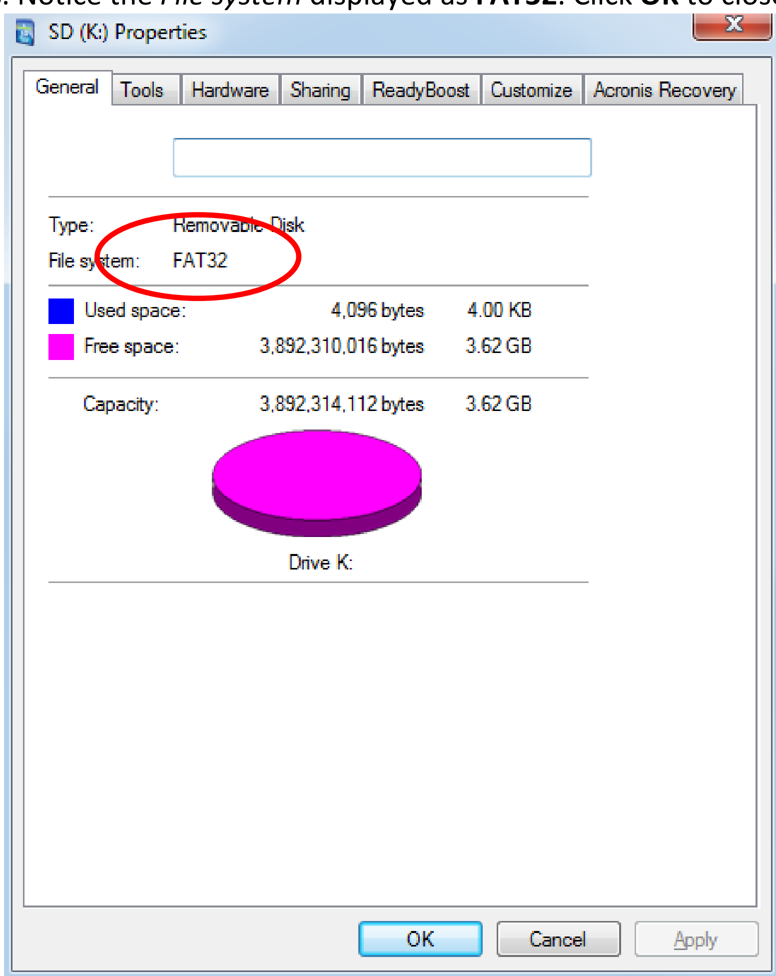



**Figure 41 – Properties of the microSD Drive**

# Appendix B: Host PC Networking Configuration

This tutorial utilizes the Gigabit Ethernet hardware and networking capability of MicroZed. To complete this tutorial, you may have to configure the network properties on your PC. The following steps will guide you through this process for a Windows 7 host PC.

1. Attach a standard Ethernet Cable between MicroZed Gigabit Ethernet Port (J1) and the host PC network interface adapter.
2. Open the **Change adapter settings** from the **Start→Control Panel→Network and Sharing Center**.

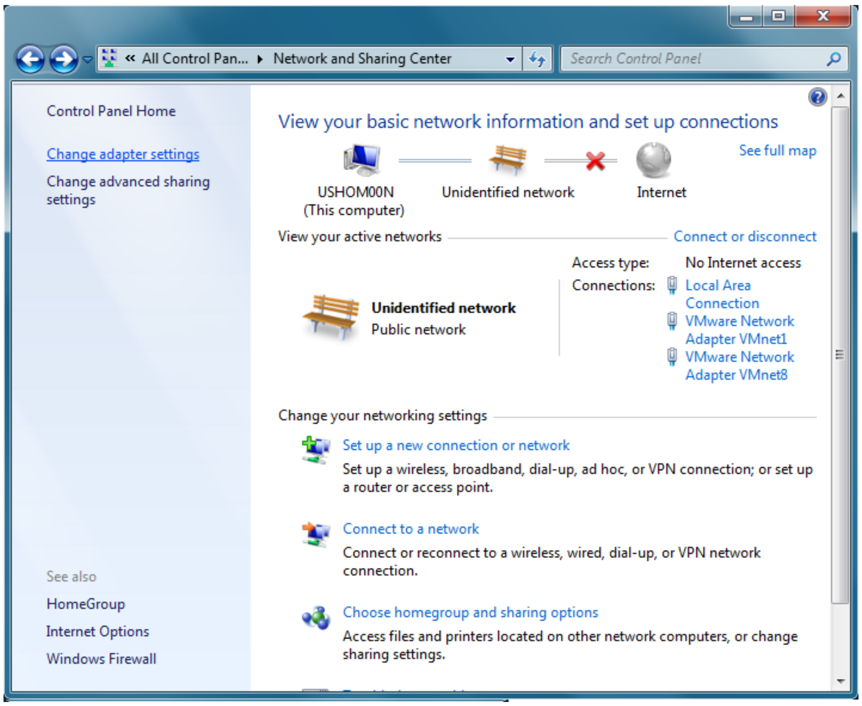



**Figure 42 – Network and Sharing Center**

1. In the **Network Connections** window, right-click on the Local Area Connection adapter entry corresponding to the network interface that is connected to MicroZed and select **Properties**.

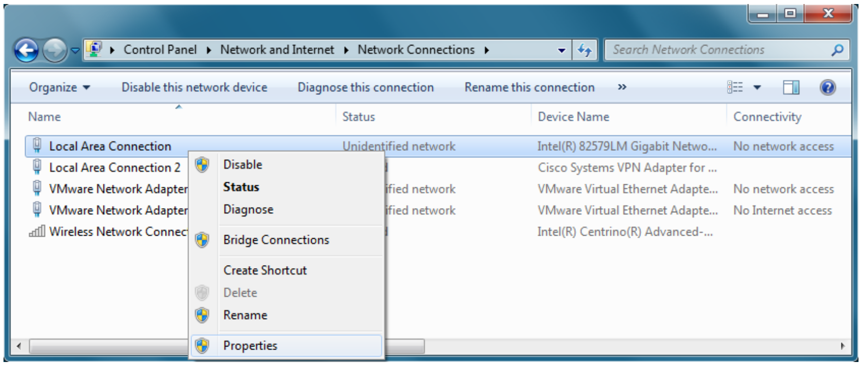



**Figure 43 – Network Connections**

2. In **Local Area Connection Properties**, select **Internet Protocol Version 4 (TCP/IPv4)**, then click the **Properties** button.

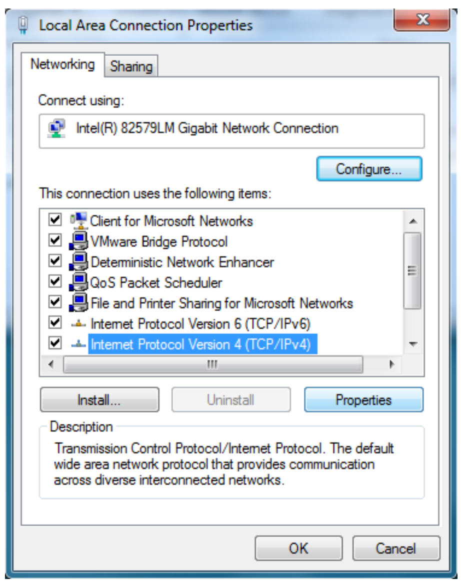



**Figure 44 – Local Area Connection Properties**

3. Set the IP address to 192.168.1.100, the Subnet mask to 255.255.255.0, and the Default gateway to 192.168.1.10 in the **Internet Protocol Version 4 (TCP/IPv4) Properties** window and then click the OK button.

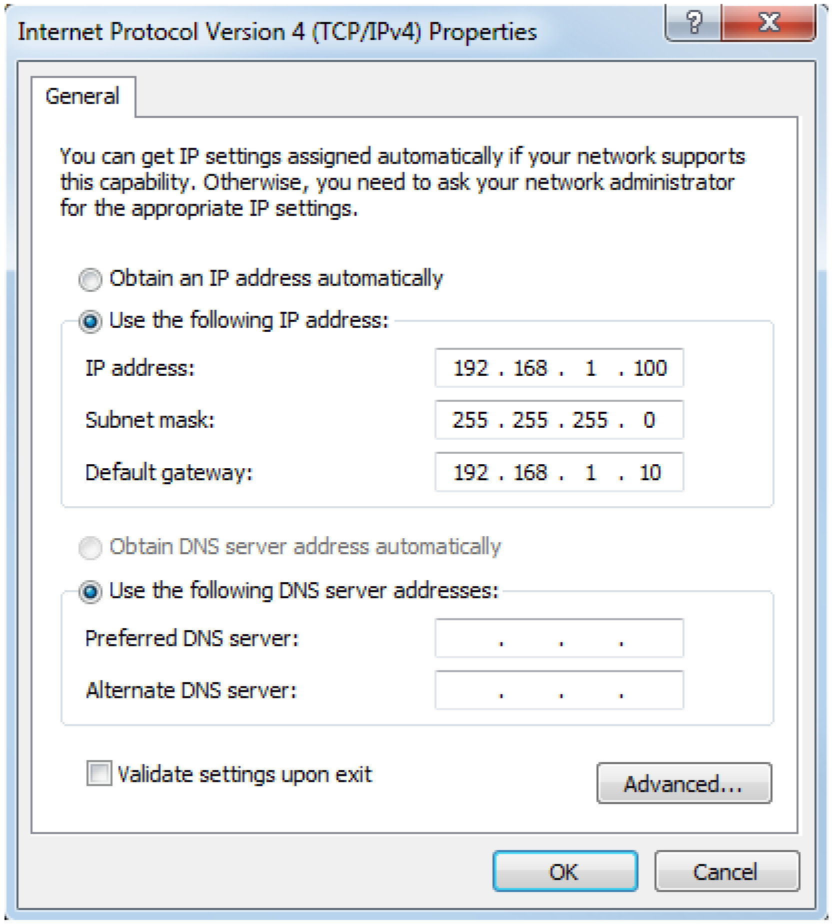



**Figure 45 – Internet Protocol Version 4 (TCP/IPv4) Properties**

The host PC networking is now configured and ready to proceed with the remainder of the tutorial.

# Appendix C: Installing and Licensing Xilinx Software

## Install Vivado Design Edition

The MicroZed XC7Z010-CLG400-1 Zynq-7000 AP SoC device development is supported by Vivado WebPACK licensing. MicroZed also comes with an entitlement voucher to a seat of Vivado Design Edition which is device locked to a XC7Z010-CLG400-1 Zynq-7000 AP SoC device. The Design Edition software is an advantage over WebPACK as it adds the Logic Analyzer capability. See
http://www.xilinx.com/products/design_tools/vivado/vivado-webpack.htm

This software can be downloaded online at:
www.xilinx.com/support/download/index.htm

You can also request a free DVD from
www.xilinx.com/onlinestore/dvd_fulfillment_request.htm

If a full seat of ISE Embedded or System Edition has already been installed, then no further software will be needed. Please check online for any updates at:
www.xilinx.com/support/download/index.htm

For detailed instructions on installing and licensing the Xilinx tools, please refer to the **Vivado Design Suite User Guide** ***Release Notes, Installation, and Licensing*** **(UG973)** available on the Xilinx website:
http://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_2/ug973-vivado-release-notes-install-license.pdf

# Appendix D: Troubleshooting

This section provides troubleshooting information for the MicroZed Open Source Linux Ethernet Performance Test Tutorial.

## Troubleshooting the MicroZed Network Connection

The Basic network configuration for the MicroZed Open Source Linux Ethernet Performance Test Tutorial is shown below:

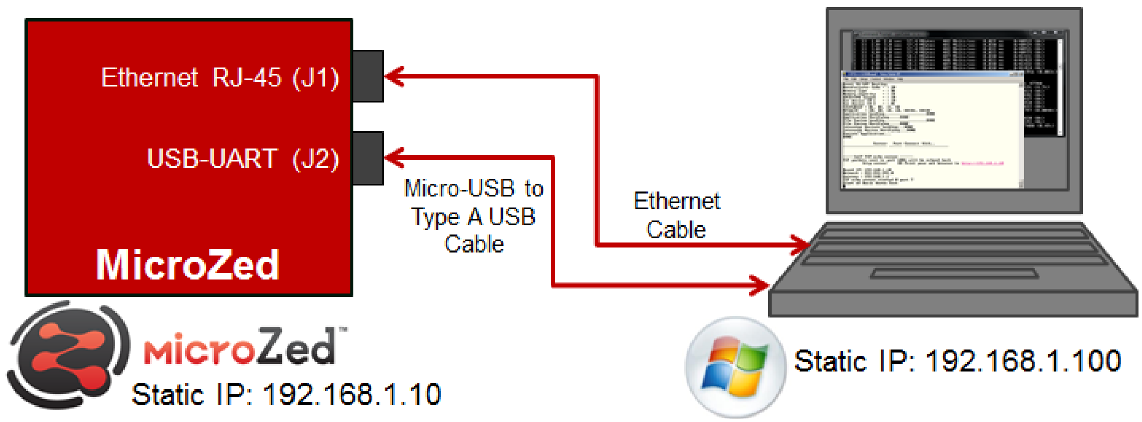



**Figure 46 – Connecting the MicroZed USB-UART and Ethernet Cable**

Make sure that Ethernet adapter on host PC Windows is configured as follows:

- IPv4 Address: 192.168.1.100
- Subnet Mask: 255.255.255.0
- Default Gateway: 192.168.1.10

Make sure the wireless internet adapter of the PC is disabled otherwise there may be a routing conflict that prevents the Zynq Linux host from being reached.

# Appendix E: Boot MicroZed from the microSD Card

This section of the tutorial demonstrates how to setup a MicroZed microSD card to boot into an open source Linux platform.

1. Download and unzip the **MicroZed_Linux_sd_image** archive from www.microzed.org.

2. If not previously completed, format the microSD Card as FAT32 as described in Appendix A: Format the microSD Card.

3. Copy the contents from the **MicroZed_Linux_sd_image** archive to the top level of the microSD card. Replace any existing versions of these files that may already be on the microSD card.

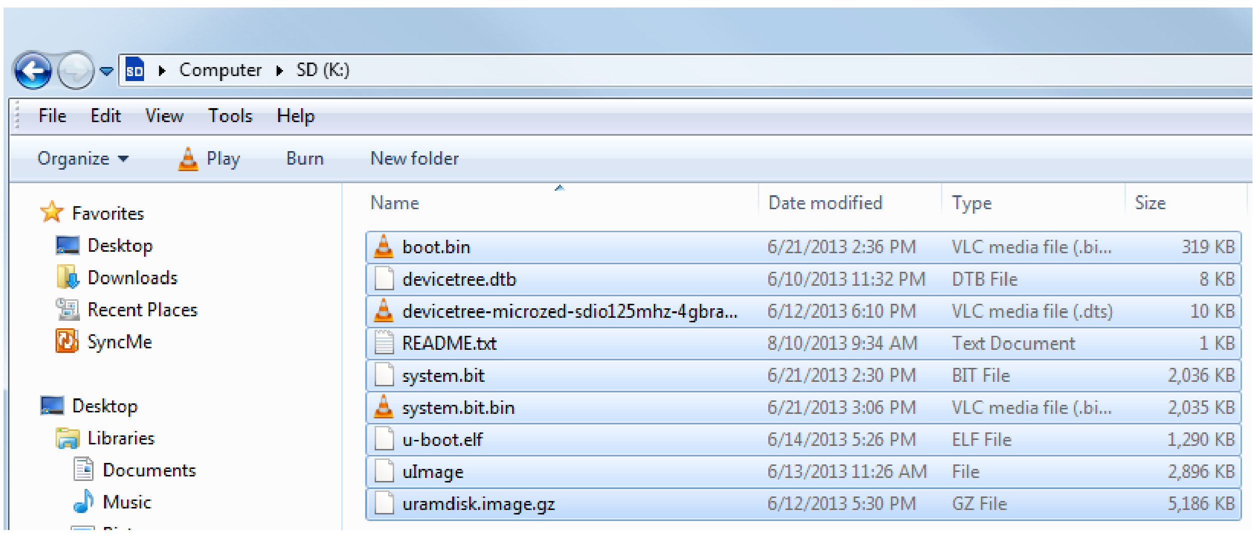



**Figure 47 – The MicroZed Linux Platform Files Copied to the microSD Card**

4. Once these files are copied to the microSD card, eject the microSD card from the PC or SD card reader.

5. Insert the 4GB microSD card included with MicroZed into the microSD card slot (J6) located on the underside of MicroZed module.

6. Verify the MicroZed boot mode (JP3-JP1) jumpers are set to SD card mode as described in the Hardware Users Guide.

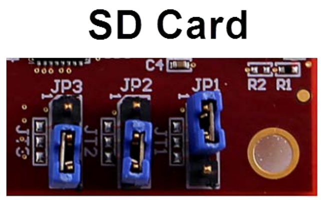



**Figure 48 – SD Card Boot Jumper Settings**

7. Return to Step 11 in section MicroZed Basic Setup and Operation and repeat the remaining experiments after having booted Linux from the microSD Card.