



PicoZed FPGA Mezzanine Connector (FMC) Carrier Card Getting Started Guide



Version 1.0
08 February 2015

Revision History

Version	Description	Date
1.0	Initial release	08 Feb 2015

Table of Contents

Revision History	2
Table of Contents	3
Getting Started with the PicoZed FMC Carrier Card	4
What's Inside the Box?	5
PicoZed PZCC-FMC Kit contents.....	5
What's on the Web?	6
Official Documentation:.....	6
Tutorials and Reference Designs:	6
Trainings and Videos:.....	6
PicoZed PZCC-FMC Key Features	7
PicoZed PZCC-FMC Basic Setup and Operation	9
Mounting the PicoZed.....	11
Example Design	12
Hardware Setup	13
Running the Example	16
File System	19
Interact with GPIO (LED and push button)	22
Ethernet Operations	29
USB-Host and microSD Card	35
Poweroff.....	38
Getting Help and Support	39
Avnet Support	39
Xilinx Support.....	41
Appendix A: Format the microSD Card	42
Appendix B: Host PC Networking Configuration	45
Appendix C: Installing and Licensing Xilinx Software.....	48
Install Vivado Design Suite, WebPack Edition	48

Getting Started with the PicoZed FMC Carrier Card

The Avnet PicoZed FMC Carrier Card (PZCC-FMC) enables hardware and software developers to explore the capabilities of the PicoZed System-on-Module (SOM). Coupled together, the PicoZed SOM and PZCC-FMC allow designers to create or evaluate Zynq™-7000 All Programmable SoC designs for both the Processor Subsystem (PS) and the Programmable Logic (PL) fabric.

The PZCC-FMC powers the PicoZed and connects the peripheral PHYs to I/O connectors. The PZCC-FMC exposes the PL I/Os, while also providing system, I/O, and transceiver power through the mezzanine MicroHeaders. The PicoZed PL I/Os are connected on the PZCC-FMC to a Low-Pin-Count (LPC) FPGA Mezzanine Connector (FMC) based on the [Vita 57](#) standard. A 2nd Ethernet circuit, 1080p HDMI, PCIe, SFP+, Digilent Pmod™ Compatible headers, LEDs, and push-buttons are additional features on the board.

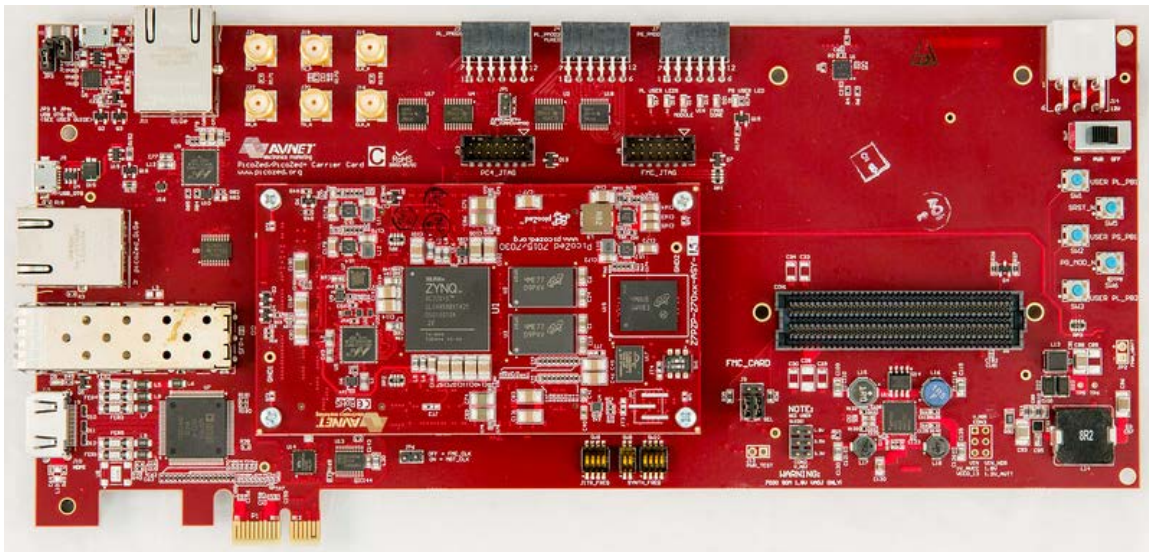


Figure 1 – PicoZed FMC Carrier Card Board shown with PicoZed SOM Mounted

This *Getting Started Guide* will outline the steps to setup the PicoZed SOM and PZCC-FMC hardware. It documents the procedure to run a PetaLinux design running on the ARM® dual-core Cortex™-A9 MPCore™ Processing System (PS).

What's Inside the Box?

PicoZed PZCC-FMC Kit contents

- PicoZed FMC Carrier Card (PZCC-FMC)
- 12V @ 5A AC/DC adapter
- 3 power adapter plugs for international use
- microUSB cable
- USB Adapter: Male Micro-B to Female Standard-A
- microSD Card 4GB
- PicoZed mounting hardware – 4 stand-offs and 8 screws (not shown)
- Documentation (not shown)
 - Quick Start Instruction card
 - Welcome Letter



Figure 2 – PZCC-FMC Kit Contents

What's on the Web?

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

Official Documentation:

- Getting started guide
- Hardware user guide
- Schematics
- Bill of materials
- Layout
- PCB net lengths
- Mechanical drawing
- 3D Model
- Board definition files for Vivado integration
- Programmable logic (PL) master user constraints

Tutorials and Reference Designs:

- Introduction to Zynq Design Tutorials
- PetaLinux BSP
- Booting PicoZed using QSPI and eMMC
- Transceiver IBERT tutorial
- PCIe tutorial
- Community projects

Trainings and Videos:

- Introduction to Zynq Software Design
- Introduction to Zynq Hardware Design

PicoZed PZCC-FMC Key Features

- Expansion connectors
 - Low pin count (LPC) FMC with 72 PL I/Os (36 differential pairs)
 - Three Digilent Pmod™ compatible interfaces
 - Access to 24 user I/O
 - One (8 I/O) connected to PS MIO (shared with eMMC on SOM)
 - One (8 I/O) connected to Bank 13 (supported with 7Z015, 20, and 30 PicoZed only)
 - One (8 I/O) connected to Bank 13 (supported with 7Z015 and 30 PicoZed only)
- Configuration and Debug
 - Xilinx Platform Cable JTAG connector for SOM
 - Xilinx Platform Cable JTAG connector for FMC
- General Purpose I/O
 - 2 user LEDs
 - 3 push buttons
- Memory
 - bootable microSD card slot with 4GB microSD card
- Communications
 - x1 PCIe Gen 2
 - SFP+ cage
 - SMA port for GTX/GTP
 - 10/100/1000 Ethernet connector
 - 10/100/1000 Ethernet PHY and connector
 - USB 2.0 connector
 - USB UART
- Other
 - HDMI output port
 - SMA reference clock input
 - Adjustable bank voltage power supply

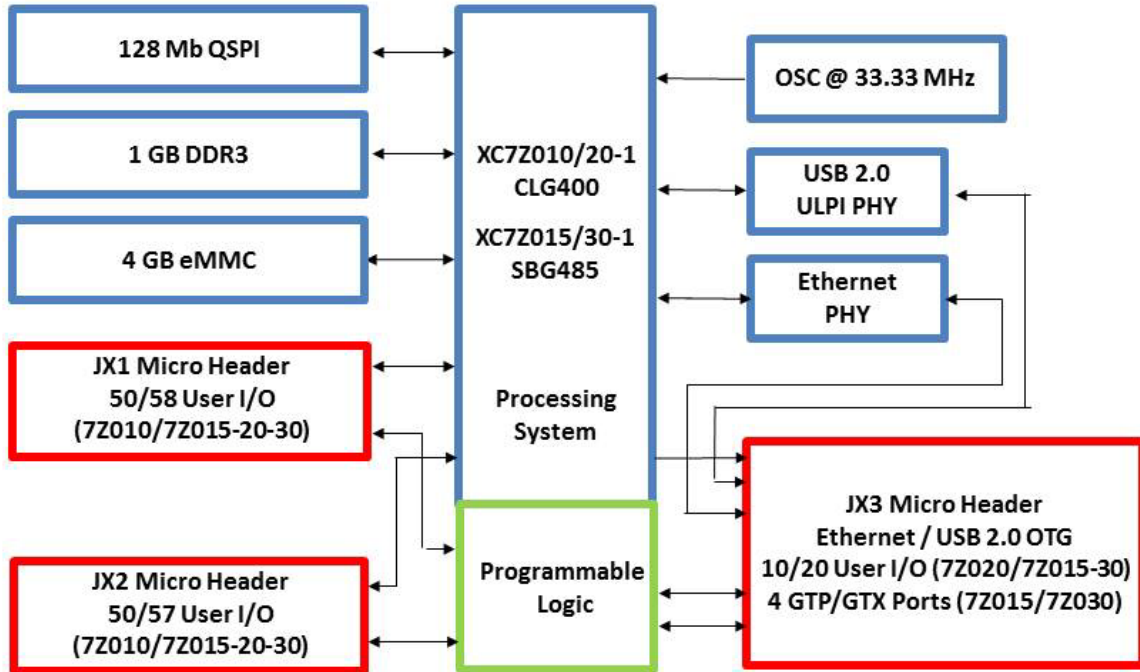


Figure 3 – PicoZed Block Diagram

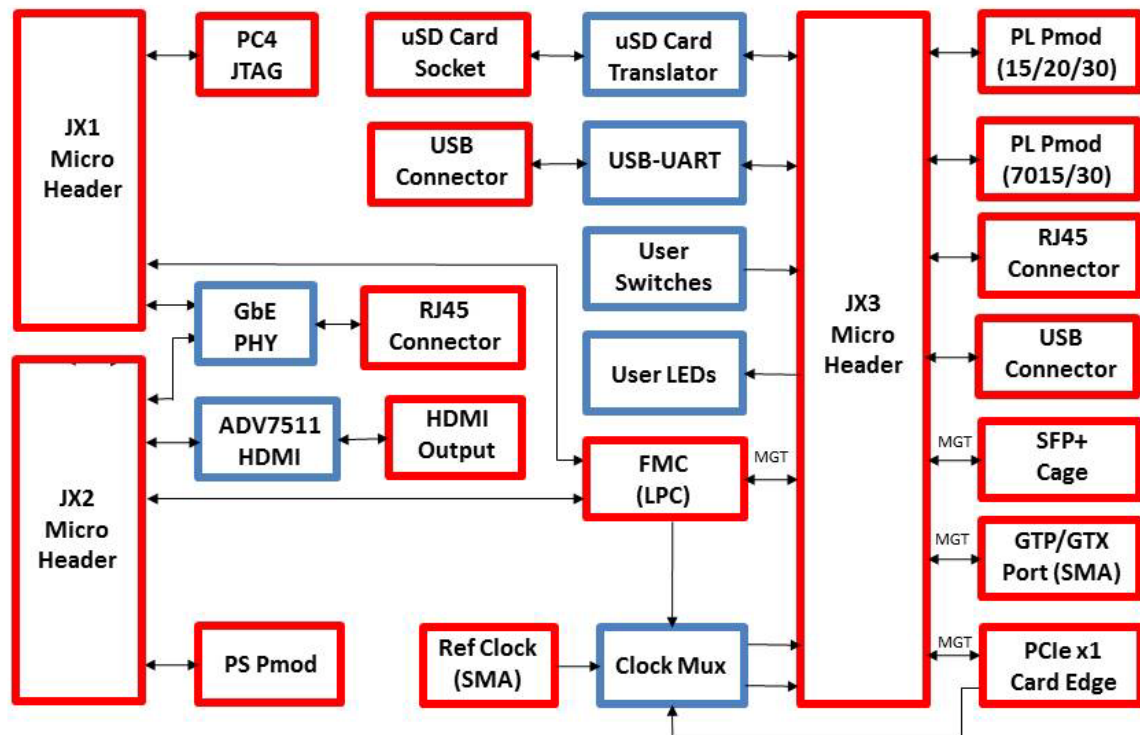


Figure 4 – PZCC-FMC Block Diagram

PicoZed PZCC-FMC Basic Setup and Operation

The PZCC-FMC and PicoZed module only operate together. Neither will function stand-alone. The functionality of both the PicoZed and the PZCC-FMC is determined by the application booted from the selected non-volatile memory – whether that be the QSPI and eMMC on the PicoZed or the microSD card on the PZCC-FMC. Since the SD Card shipped with the PZCC-FMC is not programmed prior to shipment, the PZCC-FMC does not ship with any pre-configured design. However, the PicoZed modules are tested in manufacturing with a PZCC-FMC, so the default image stored in the PicoZed QSPI is fully compatible with the PZCC-FMC.

Of course, the primary purpose of the PZCC-FMC is to allow both a PicoZed and an FMC module to be connected together. In this case, the application is still controlled by the PicoZed while the FMC may add enhanced functionality with additional circuitry.

This *Getting Started Guide* offers system developers examples of how to do several things with the PicoZed and PZCC-FMC together:

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

In addition to the items included in the kit, you will also need the following to complete the exercises in this tutorial.

- **PicoZed module**
- **Ethernet cable**
- **microSD card reader/adaptor**

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

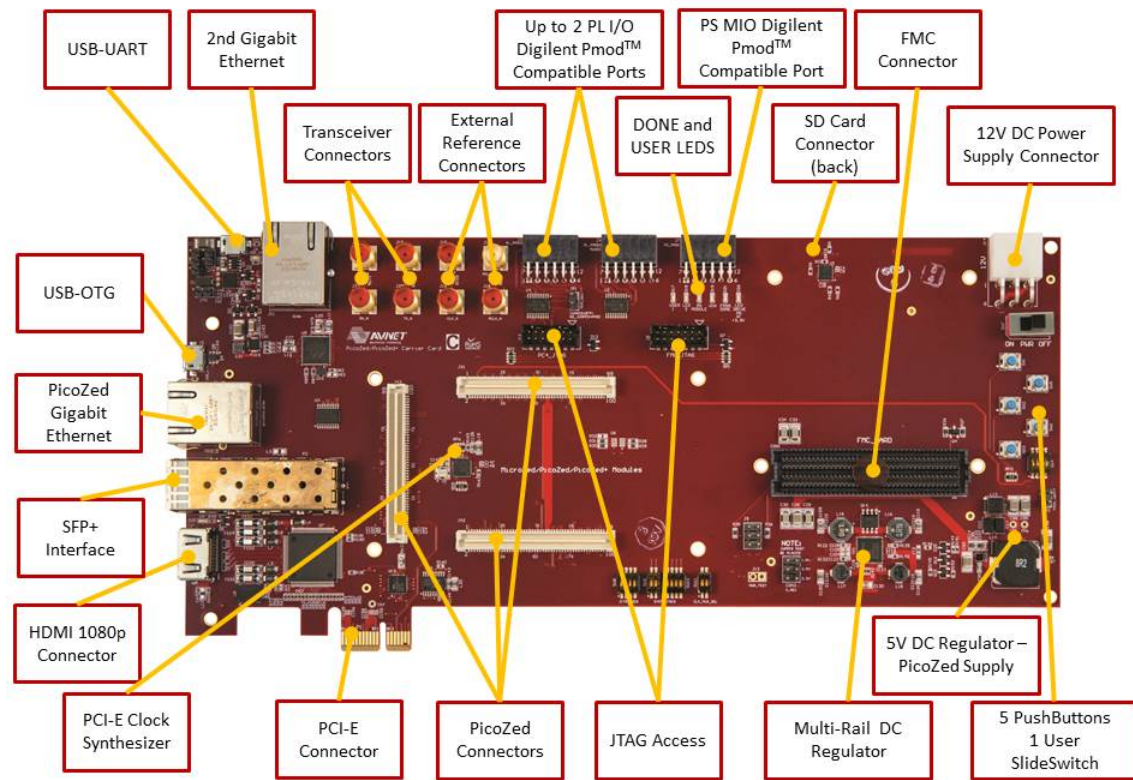


Figure 5 – PZCC-FMC Topology

Mounting the PicoZed

The PZCC-FMC Kit includes mounting hardware that allows you to more permanently secure your PicoZed to the PZCC-FMC. This can be done now, but it is not required.

1. Insert one of the screws through the top of one of the mounting holes on the PicoZed.
2. Twist a stand-off onto the screw.
3. Repeat for the other three mounting holes.
4. Plug the PicoZed onto the PZCC-FMC.
5. From the bottom-side of the PZCC-FMC, use the screws to attach to the standoffs through the PZCC-FMC mounting holes.

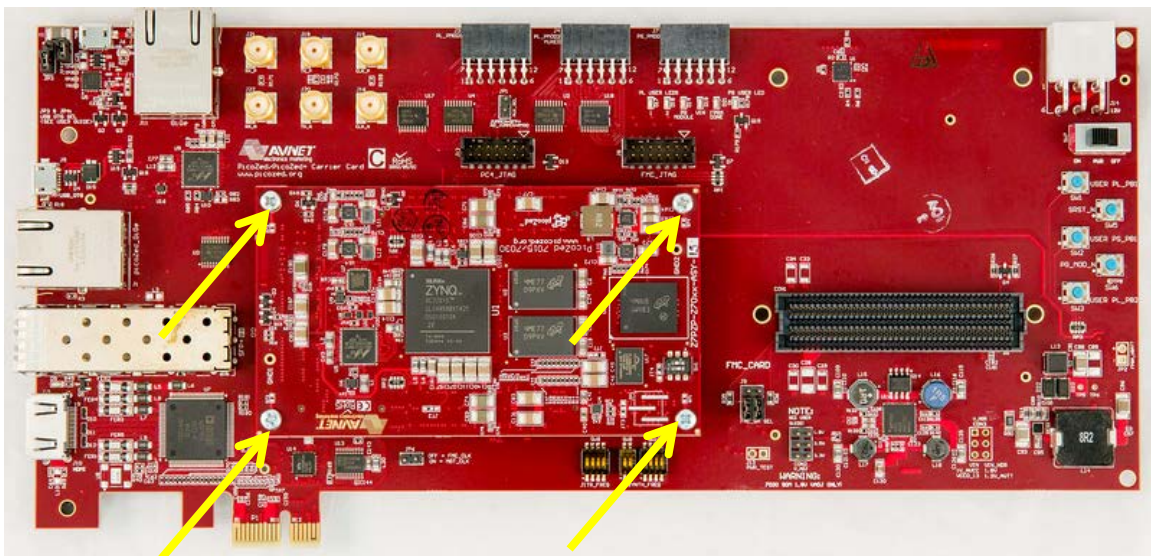


Figure 6 – Location of Four Mounting Holes

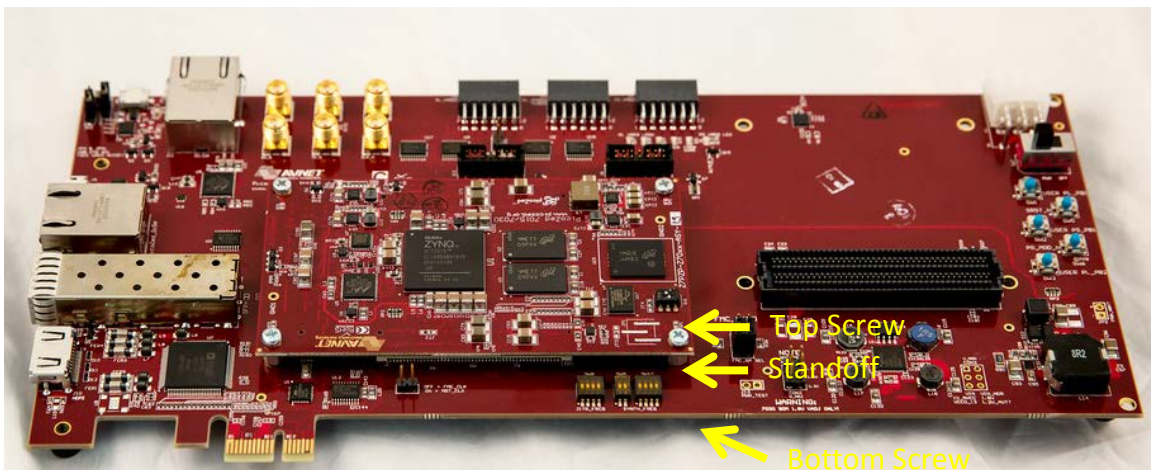


Figure 7 – View Showing Top Screw, Standoff, and Bottom Screw

Example Design

The example PZCC-FMC design is based on the initial PetaLinux design that ships with the PicoZed SOM in the QSPI. If the QSPI has been erased or reprogrammed, then use the tutorial available at www.picozed.org to restore the original factory image.

A block diagram for the hardware platform is shown below.

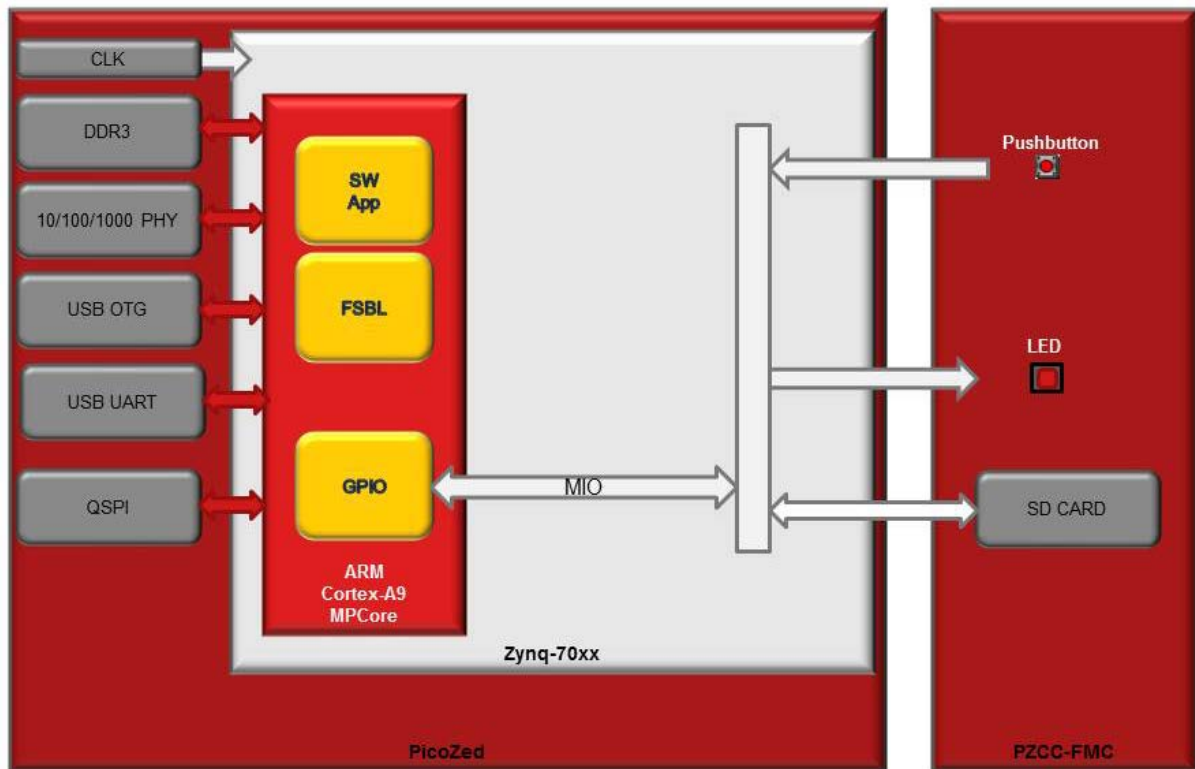
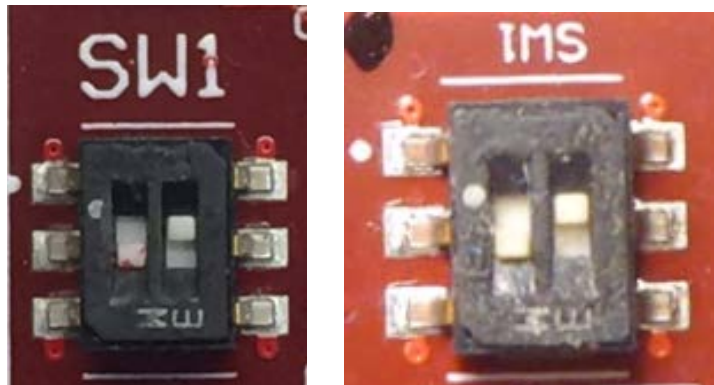


Figure 8 – PicoZed/PZCC-FMC Hardware Design

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 PZCC-FMC. Refer to [Appendix B: Host PC Networking Configuration](#) for instructions to accomplish this.
3. A terminal program is required. Windows 7 or 8 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.PicoZed.org to download instructions for installing the Silicon Labs CP2104 USB-to-UART driver.
<http://picozed.org/support/documentation/4701>
Silicon Labs CP210x USB-to-UART Setup Guide
5. Set the PicoZed boot mode switch SW1 to QSPI mode as shown below.



**Figure 9 – PicoZed SOM SW1 Set to QSPI Boot
7010/20 on the Left; 7015/30 on the Right**

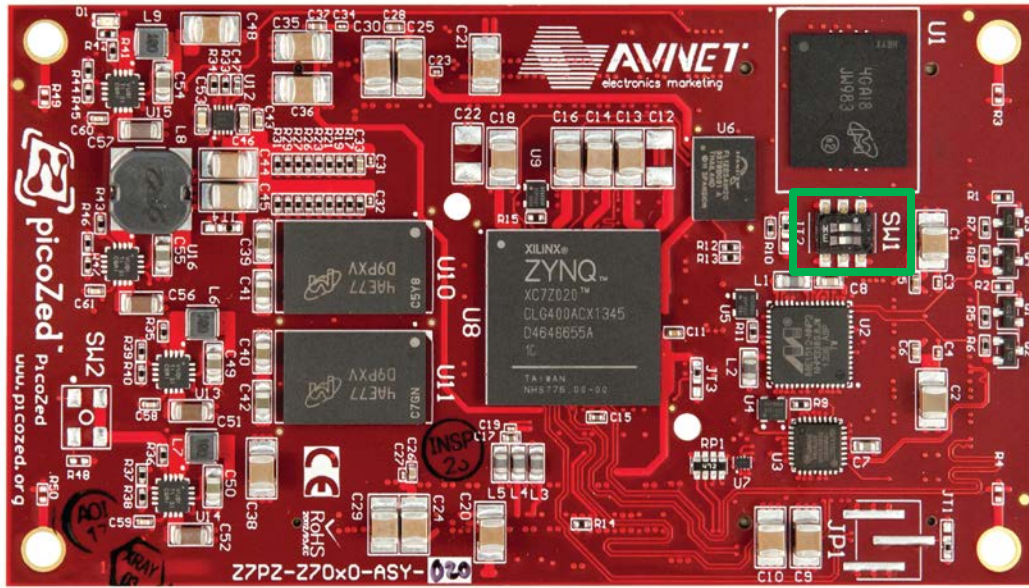


Figure 10 – PicoZed 7010/20 SW1 Location

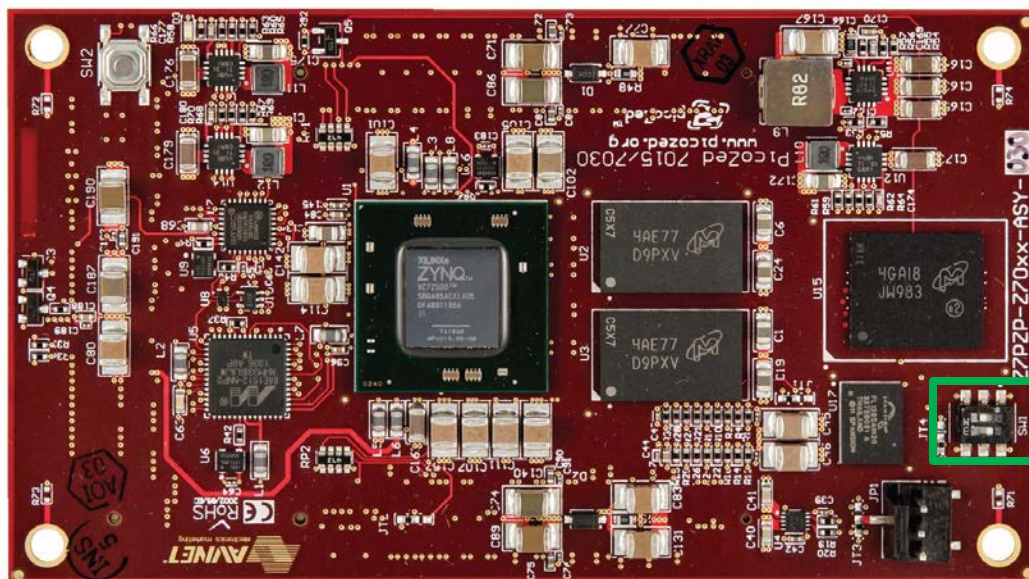


Figure 11 – PicoZed 7015/30 SW1 Location

6. Make sure the PZCC-FMC power switch is in the OFF position.
7. Insert the PicoZed module onto the PZCC-FMC.

8. Insert the blank formatted 4GB microSD card included with PZCC-FMC into the microSD card slot (J2) located on the underside of PZCC-FMC (see Figure 5 for location).
9. Set the on-board jumpers as follows
 - JP1 is open
 - JP3 is closed in position 1-2
 - JP4 is closed
 - JP6 is open
 - J9 is closed in positions 3-5 and 4-6
 - CON2 is open, which sets V_ADJ to 1.8V
10. Plug in the micro-USB cable between the host PC and the PZCC-FMC USB-UART port (J6).
11. Plug in an Ethernet cable between the host PC and the PZCC-FMC **picoZed GiGe (J1)** port.
12. Insert the appropriate country plug into the 12V AC/DC adapter. Plug it into the J14 2x3 power connector. (NOTE – this 2x3 connector is NOT compatible with ATX power supplies.)

Running the Example

13. Turn the power switch on the PZCC-FMC to the ON position. After 1-2 seconds, you will notice four LEDs that are lit:
- D1 (green) on PicoZed, indicating Power Good
 - D19 (green) on PZCC-FMC, indicating Vin is on
 - D14 (green) on PZCC-FMC, which is the PG_MODULE handshake between the SOM and the Carrier indicating that the SOM power is good
 - D21 (blue) on PZCC-FMC indicating that the Zynq PL configuration is DONE
 - D6 (amber) indicating the USB-UART is connected

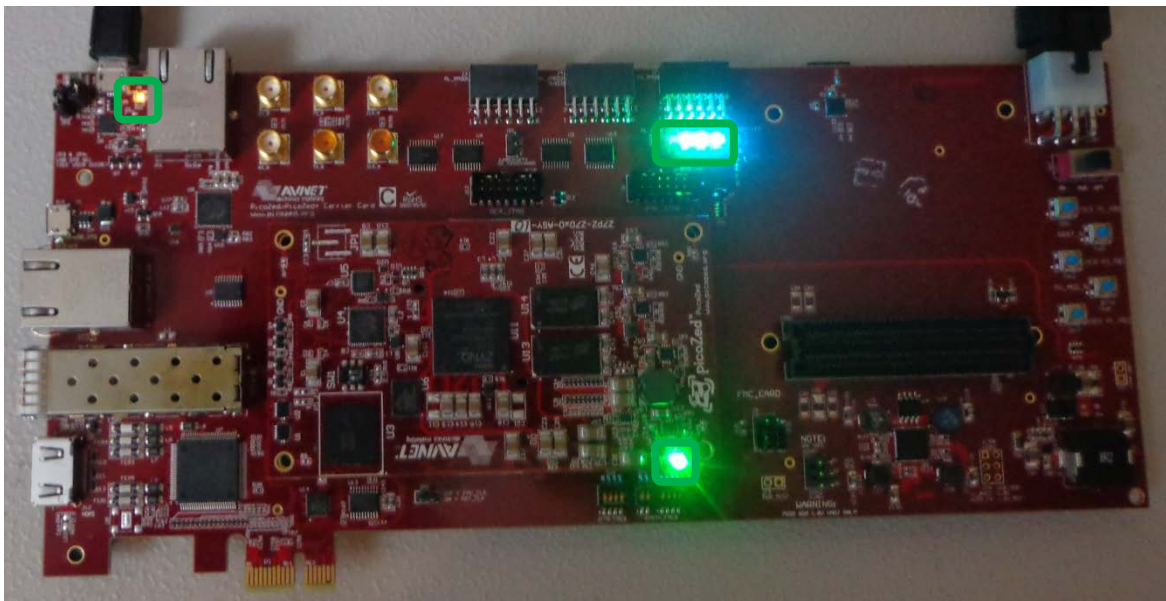


Figure 12 – PicoZed / PZCC-FMC Powered On with LEDs

14. 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 13, using the appropriate COM port that you discover on your own machine.

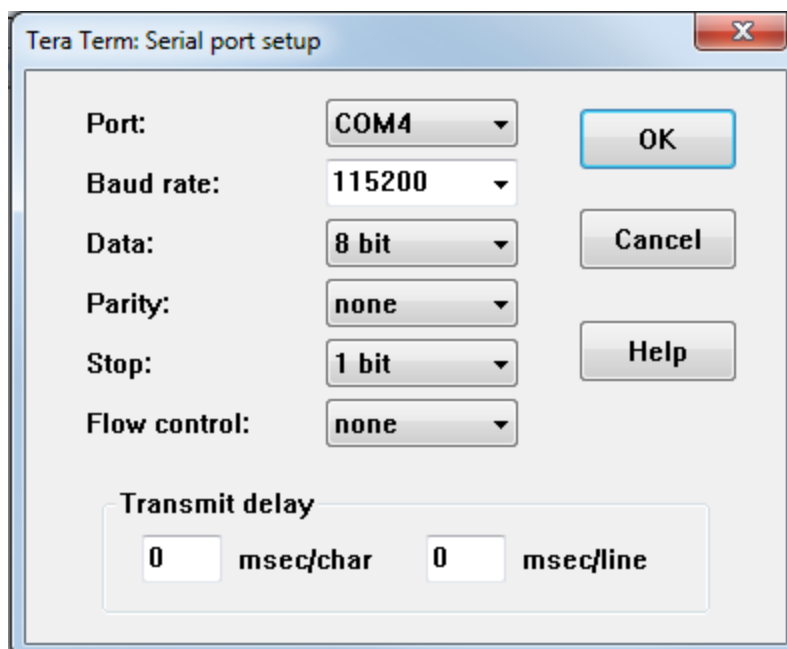


Figure 13 – Connect Tera Term to the proper COMx port

15. Perform a System Processor Reset by pushing the SRST_N button (SW5) on the PZCC-FMC.
16. When the terminal output from U-Boot and a countdown is observed, allow the countdown to expire.
17. The terminal output shows that the Zynq boots to U-boot and then Linux, concluding with the **zynq>** prompt.

```

COM7:115200baud - Tera Term VT
File Edit Setup Control Window Help
io scheduler noop registered
io scheduler deadline registered
io scheduler cfg registered (default)
dma-pl330 f8003000.ps7-dma: Loaded driver for PL330 DMAC-267056
dma-pl330 f8003000.ps7-dma: DBUFF-128x8bytes Num_Chans-8 Num_Peri-4 Num_Events-16
e0001, rk [ttyPS0] enabled, bootconsole disabled
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":
0x000000000000-0x00000001000000 : "qspi-fsbl-uboot"
0x00000001000000-0x00000006200000 : "qspi-linux"
0x00000006200000-0x00000006200000 : "qspi-device-tree"
0x00000006200000-0x0000000c000000 : "qspi-rootfs"
0x0000000c000000-0x00000010000000 : "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 USB320 ULPI transceiver.
ULPI integrity check: passed.
xushbps-ehci xushbps-ehci.0: Xilinx PS USB EHCI Host Controller
xushbps-ehci xushbps-ehci.0: new USB bus registered, assigned bus number 1
xushbps-ehci xushbps-ehci.0: irq 53, io mem 0x00000000
xushbps-ehci xushbps-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 usbhcd
usbhid: USB HID core driver
TCP: cubic registered
NET: Registered protocol family 17
UFP 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
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
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=255.255.255.0, 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
UFS: Mounted root (ext2 filesystem) on device 1:0.
devtmpfs: mounted
Freeing init memory: 164K
Starting rcS...
** Mounting filesystem
mount: mounting /dev/mmchlk0p1 on /mnt failed: No such file or directory
mount: mounting /dev/mmchlk0 on /mnt failed: No such file or directory
** Setting up mdev
** Starting telnet daemon
** Starting http daemon
** Starting ftp daemon
** Starting ssh daemon
rcS Complete
zynq>

```

Figure 14 – PicoZed and PZCC-FMC Example Design

File System

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

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

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

```
zynq> pwd
```

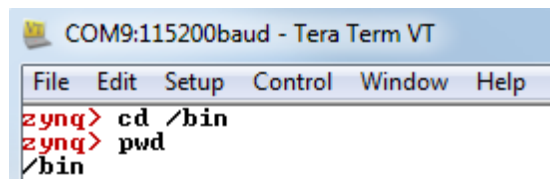


Figure 15 – Print Working Directory

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

```
zynq> ls
```



Figure 16 – List Contents

21. 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
lrwxrwxrwx 1 root root 7 Nov 27 2012 df -> busybox
```

Figure 17 – Detailed List Contents

22. 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 18 – Disk Usage

23. 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        3863040         4    3863036   0% /mnt
zynq>
```

Figure 19 – Disk Free

24. 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"
```

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

Figure 20 – Find a File

25. 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
```

```
zynq> cd /
zynq> which iperf
/usr/bin/iperf
```

Figure 21 – 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 PicoZed booted to the Linux command prompt, the MIO GPIO hardware can be accessed directly via the generic sysfs GPIO driver.

26. 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.

```
zynq> ls /sys/class/gpio/
```

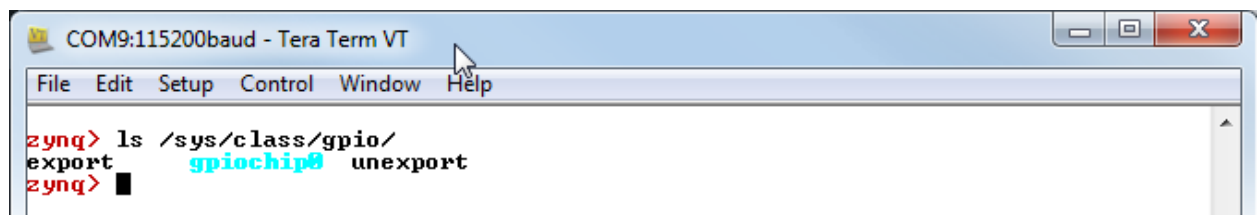


Figure 22 – Exploring the Sysfs Subsystem

27. Take a look at Sheets 7 of the PZCC-FMC schematic and determine which IO pin the LED tied to MIO (PS_LED1 – D1) is connected to.

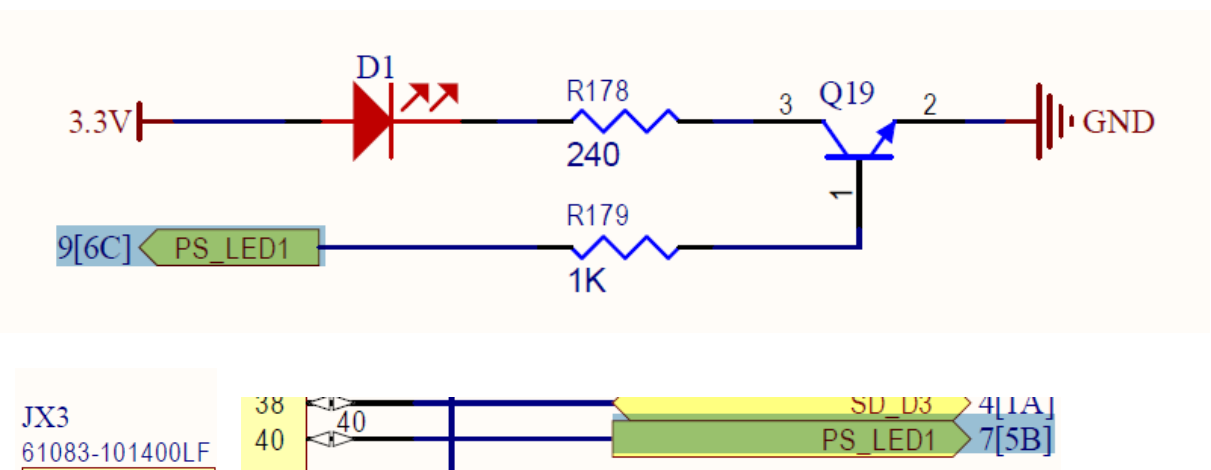


Figure 23 – PZCC-FMC Schematic Snippets Relating to PS_LED1 (D1)

28. In looking at the schematic, you should have determined that the MIO LED **D1** is connected to pin **JX3.pin40**.
29. Now use sheets 9 and 5 PicoZed schematic to determine how JX3.pin40 is connected.

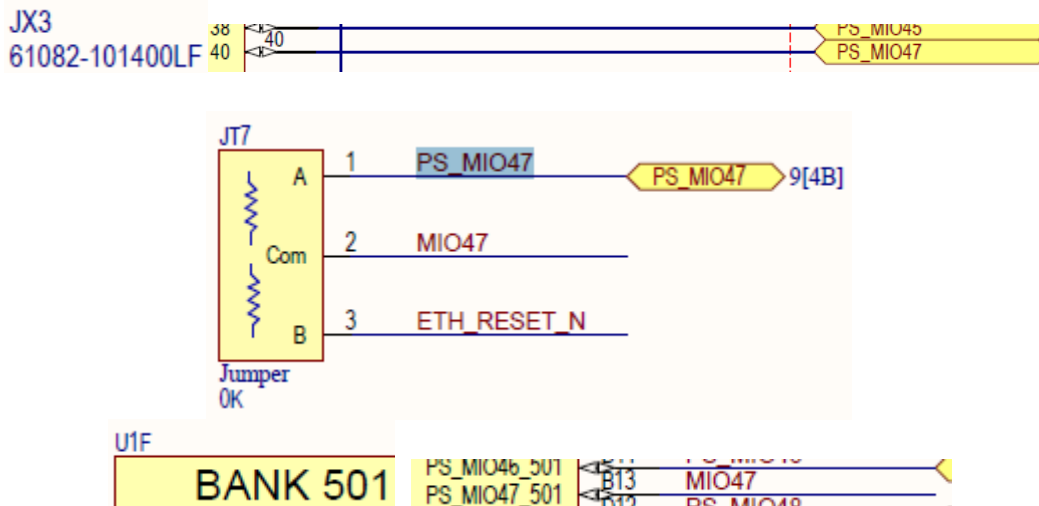


Figure 24 – PicoZed Schematic Snippets Relating to PS_LED1 (D1)

30. In looking at the schematic, you should have determined that pin **JX3.pin40** is connected to JT7 (default jumper at 1-2), which corresponds to **MIO47** on Bank 501 of the Zynq.
31. Using MIO number 47, 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.

```
zynq> echo 47 > /sys/class/gpio/export
zynq> 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.



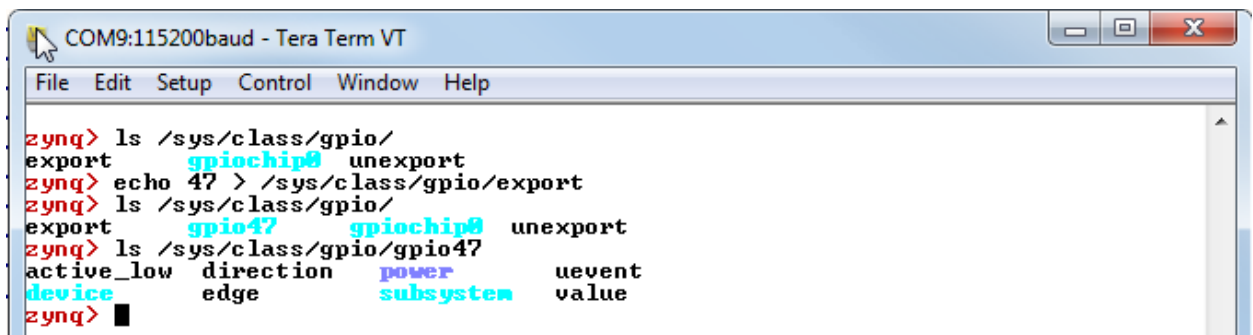
```
COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help
zynq> ls /sys/class/gpio/
export      gpiochip8  unexport
zynq> echo 47 > /sys/class/gpio/export
zynq> ls /sys/class/gpio/
export      gpio47    gpiochip8  unexport
zynq>
```

Figure 25 – Exporting GPIO47 Controls Via the Sysfs Subsystem

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

```
zynq> ls /sys/class/gpio/gpio47
```

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



```
COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help
zynq> ls /sys/class/gpio/
export      gpiochip8  unexport
zynq> echo 47 > /sys/class/gpio/export
zynq> ls /sys/class/gpio/
export      gpio47    gpiochip8  unexport
zynq> ls /sys/class/gpio/gpio47
active_low  direction  power      uevent
device      edge       subsystem  value
zynq>
```

Figure 26 – GPIO47 Control Properties Via the Sysfs Subsystem

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**.

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

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

34. Modify the value property of the gpio47 node and watch the red PicoZed D1 LED as the command input is entered.

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

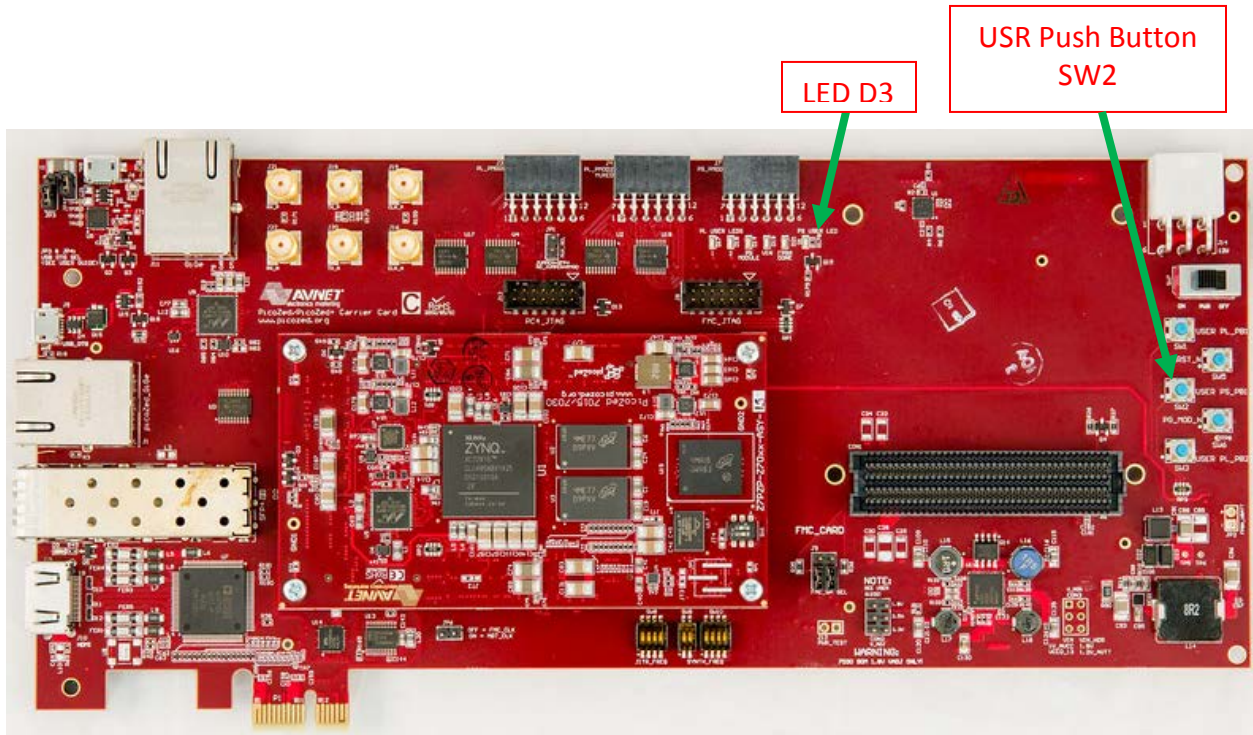


Figure 27 – PicoZed LED and Push Button

Did you observe a change in state on D1 LED?

Modify the value property of the gpio47 node again and watch the PicoZed D1 LED as the command input is entered.

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

35. 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?

```

zynq> ls /sys/class/gpio/
export      gpiochip8  unexport
zynq> echo 47 > /sys/class/gpio/export
zynq> ls /sys/class/gpio/
export      gpio47  gpiochip8  unexport
zynq> ls /sys/class/gpio/gpio47
active_low  direction  power      uevent
device      edge        subsystem  value
zynq> echo out > /sys/class/gpio/gpio47/direction
zynq> echo 1 > /sys/class/gpio/gpio47/value
zynq> echo 0 > /sys/class/gpio/gpio47/value
zynq>

```

Figure 28 – Modifying the GPIO47 value Property

36. Perform a similar exercise using MIO push button **USER_PS_PB1** (Net Name =PS_PB1, REFDES=SW2) as an input device. Take a look at the PZCC-FMC and PicoZed schematics and determine to which IO pin the MIO push button **USER_PS_PB1** is connected.

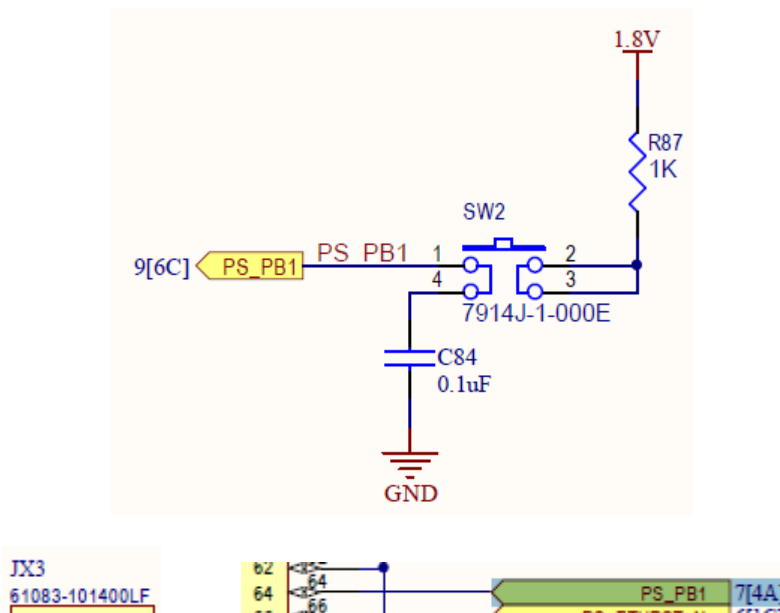


Figure 29 – PZCC-FMC Schematic Snippets Related to PS_PB1

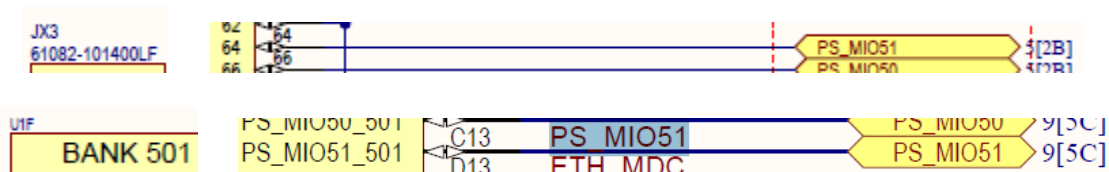


Figure 30 – PicoZed Schematic Snippets Related to PS_PB1

37. In looking at the schematics, you should have determined that the MIO push button **PS_PB1** is connected to signal **PS_MIO51**. Using this MIO number, export the corresponding GPIO device for use and evaluate the GPIO folder again.

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

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

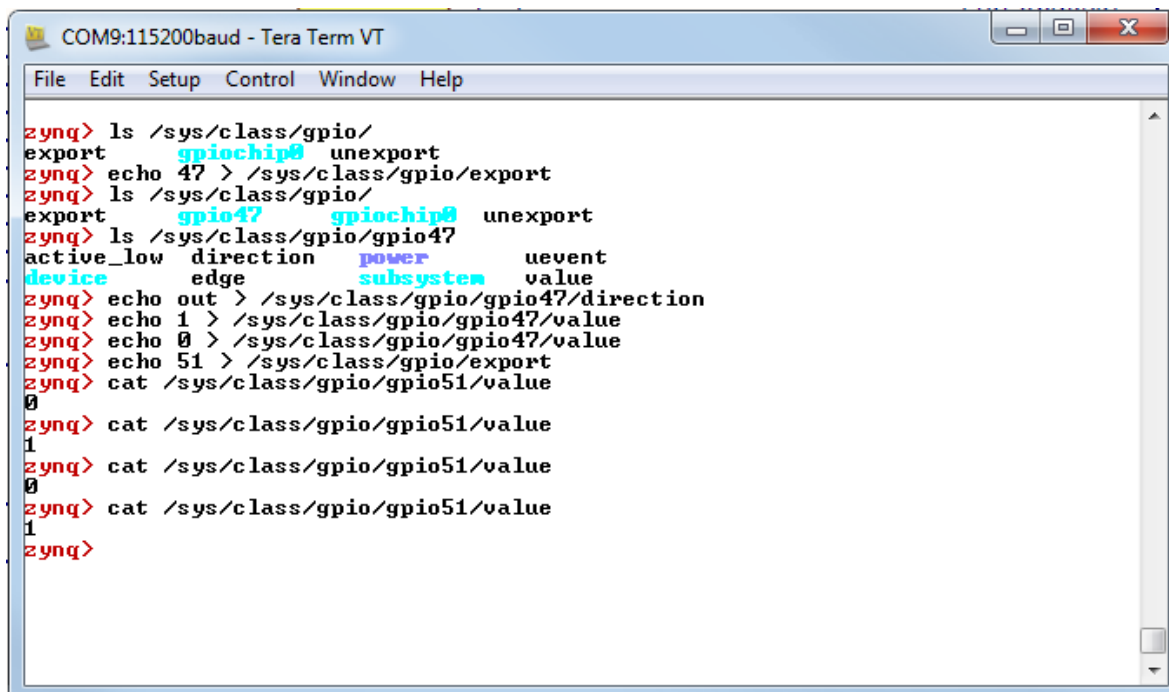
```
zynq> echo in > /sys/class/gpio/gpio51/direction
```

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

```
zynq> cat /sys/class/gpio/gpio51/value
```

40. 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?

41. 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?



```
COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

zynq> ls /sys/class/gpio/
export      gpiochip0  unexport
zynq> echo 47 > /sys/class/gpio/export
zynq> ls /sys/class/gpio/
export      gpio47  gpiochip0  unexport
zynq> ls /sys/class/gpio/gpio47
active_low  direction  power      uevent
device      edge        subsystem  value
zynq> echo out > /sys/class/gpio/gpio47/direction
zynq> echo 1 > /sys/class/gpio/gpio47/value
zynq> echo 0 > /sys/class/gpio/gpio47/value
zynq> echo 51 > /sys/class/gpio/export
zynq> cat /sys/class/gpio/gpio51/value
0
zynq> cat /sys/class/gpio/gpio51/value
1
zynq> cat /sys/class/gpio/gpio51/value
0
zynq> cat /sys/class/gpio/gpio51/value
1
zynq>
```

Figure 31 – Reading the GPIO51 value Property

42. 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.

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

43. 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_leds_led.sh`.

```
zynq> cd /  
zynq> echo while : > pb_leds_led.sh  
zynq> echo do >> pb_leds_led.sh  
zynq> echo "cat /sys/class/gpio/gpio51/value >  
/sys/class/gpio/gpio47/value" >> pb_leds_led.sh  
zynq> echo done >> pb_leds_led.sh  
zynq> chmod 755 pb_leds_led.sh  
zynq> ./pb_leds_led.sh
```

44. 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 PicoZed 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.

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

```
zynq> ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:01:22
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:54 Base address:0xb000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

zynq> □
```

Figure 32 – PicoZed IP Address Revealed with `ifconfig` Command

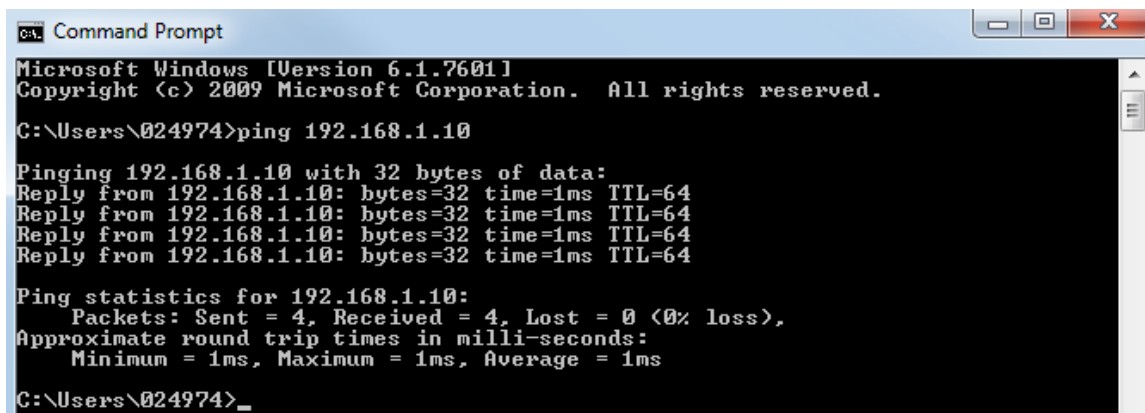
46. The most simple connectivity test is to use the 'ping' command. Try pinging your host PC with the following command (assuming you used the address given in the setup section of this document). Hit Ctrl-C when you are satisfied.

```
zynq> ping 192.168.1.100
```

```
zynq> ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: seq=0 ttl=128 time=1.124 ms
64 bytes from 192.168.1.100: seq=1 ttl=128 time=1.530 ms
64 bytes from 192.168.1.100: seq=2 ttl=128 time=0.774 ms
64 bytes from 192.168.1.100: seq=3 ttl=128 time=0.996 ms
64 bytes from 192.168.1.100: seq=4 ttl=128 time=0.885 ms
64 bytes from 192.168.1.100: seq=5 ttl=128 time=1.105 ms
64 bytes from 192.168.1.100: seq=6 ttl=128 time=2.359 ms
^C
--- 192.168.1.100 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0.774/1.253/2.359 ms
zynq> □
```

Figure 33 – Ping the Host PC

47. Likewise, you can ping the PicoZed from the Host PC. Open a Windows command prompt, and enter command 'ping 192.168.1.10'



```
CA: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\024974>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: bytes=32 time=1ms TTL=64
Reply from 192.168.1.10: bytes=32 time=1ms TTL=64
Reply from 192.168.1.10: bytes=32 time=1ms TTL=64
Reply from 192.168.1.10: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\Users\024974>_
```

Figure 34 – Ping the PicoZed

48. To view the PetaLinux embedded webpage, open a web browser (such as Firefox) and browse to the PicoZed IP address <http://192.168.1.10/> as the URL. The 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.

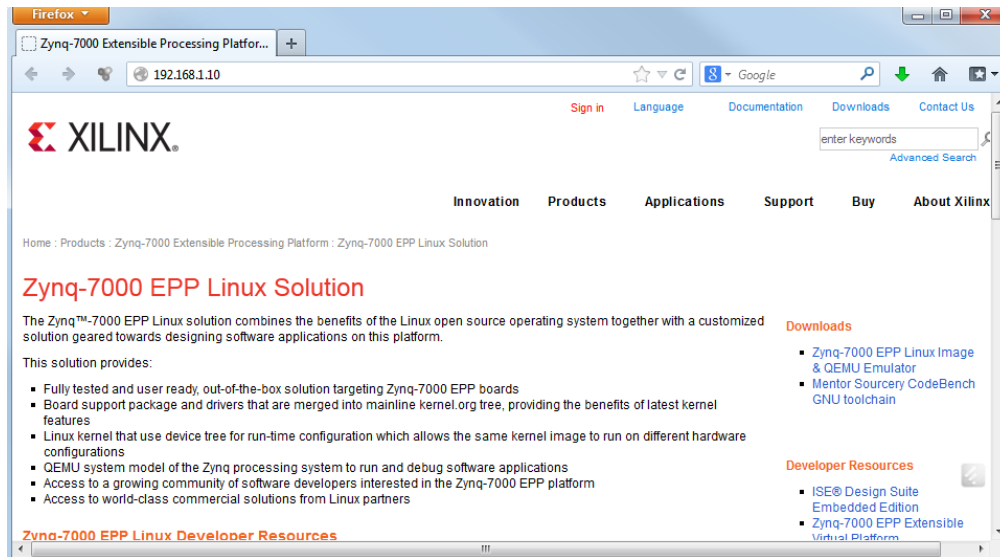


Figure 35 – PicoZed Webpage Shown In PC Host Browser

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

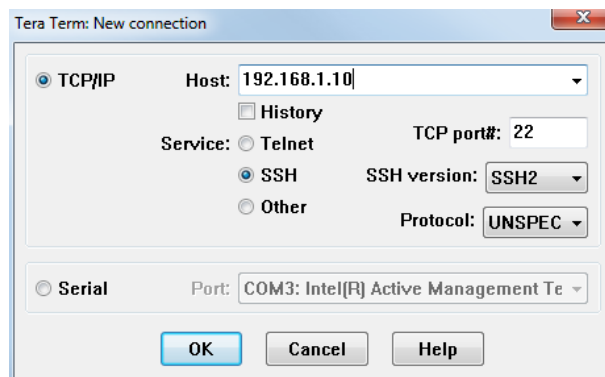


Figure 36 – Setup SSH Connection in Tera Term

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

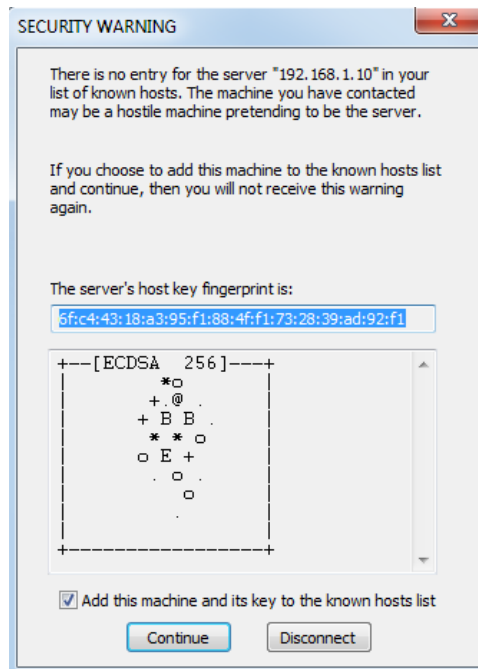


Figure 37 – Tera Term SSH Security Warning

55. 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. Click **OK**.

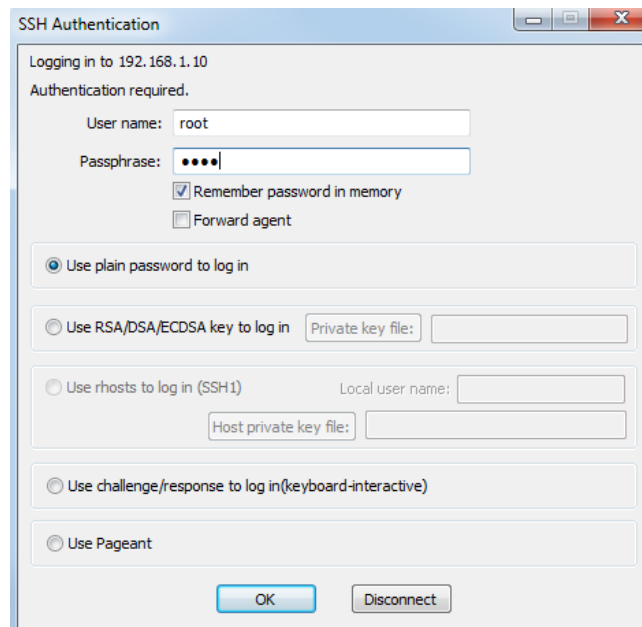


Figure 38 – Login as root

56. 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'

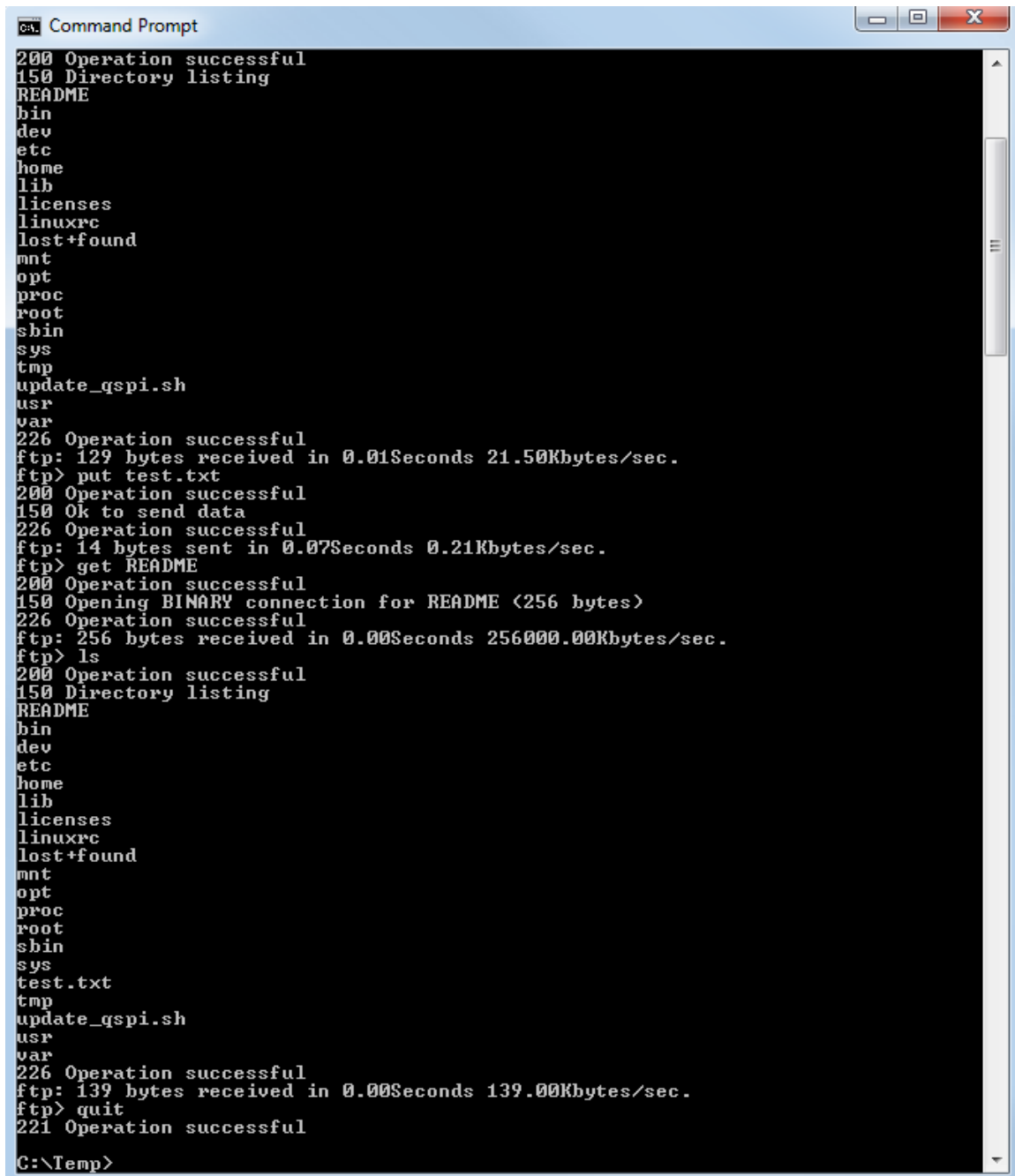
```

zynq> cd /
zynq> ls -l
total 841
-rw-r--r--      1 root      root           256 Nov 27  2012 README
drwxr-xr-x      2 root      root        2048 Nov 27  2012 bin
drwxr-xr-x      6 root      root       2580 Jan  1  00:00 dev
drwxr-xr-x      5 root      root       1024 Jan  1  00:00 etc
drwxrwxrwx      3 root      root       1024 Nov 27  2012 home
drwxr-xr-x      3 root      root       1024 Nov 27  2012 lib
drwxr-xr-x     12 root      root       1024 Nov 27  2012 licenses
lrwxrwxrwx      1 root      root         11 Nov 27  2012 linuxrc -> bin/busybox
drwx-----      2 root      root    838656 Nov 27  2012 lost+found
drwxr-xr-x      2 root      root       4096 Jan  1  00:00 mnt
drwxr-xr-x      2 root      root       1024 Nov 27  2012 opt
dr-xr-xr-x     64 root      root         0 Jan  1  00:00 proc
drwxr-xr-x      2 root      root       1024 Jan  1  00:57 root
drwxr-xr-x      2 root      root       1024 Nov 27  2012/sbin
dr-xr-xr-x     12 root      root         0 Jan  1  00:00 sys
drwxrwxrwt      2 root      root         40 Jan  1  00:00 tmp
-rwxr-xr-x      1 root      root       481 Nov 27  2012 update_qspi.sh
drwxr-xr-x      6 root      root       1024 Nov 27  2012 usr
drwxr-xr-x      5 root      root       1024 Nov 27  2012 var
zynq>

```

Figure 39 – Remote PicoZed Terminal via SSH Session

57. Logout and close the remote session with the `exit` command.
58. Open a Windows Command Prompt.
59. 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 PicoZed.
60. 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 40 – PicoZed 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 PicoZed. The PZCC-FMC / PicoZed USB 2.0 is designed to be configured as Host, Device, or OTG, with the default jumper settings for JP3 and JP4 configuring it for Host.

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

PicoZed only has one USB 2.0 port. To connect multiple USB devices with the PZCC-FMC, connect a powered hub to the USB-Host port. USB devices attached to this hub can then also be accessed in Linux.

61. 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.
62. Connect the USB memory stick to the included Male Micro-B to Female Standard-A USB adapter. Then connect the adapter to the PZCC-FMC USB_OTG microUSB connector (J5).
63. The USB memory stick should enumerate and the device indication should display on the serial console. As shown in Figure 41, the primary partition of the USB memory stick is enumerated as device `/dev/sda`.

```
zynq> usb 1-1: new high-speed USB device number 3 using xusbps-ehci
scsi1 : usb-storage 1-1:1.0
scsi 1:0:0:0: Direct-Access    SanDisk Cruzer           8.02 PQ: 0 ANSI: 0 CCS
sd 1:0:0:0: Attached scsi generic sg0 type 0
sd 1:0:0:0: [sda] Attached SCSI removable disk
```

Figure 41 – USB Drive Enumerated as `/dev/sda`

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
```

64. 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
```

Figure 42 – Nothing Mounted at `/mnt`

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

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

66. 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 43 – SD Card Successfully Mounted

67. 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 one file (test.txt) that was 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
```

```
zynq> mount /dev/sda1 /mnt/memstick
zynq> df
Filesystem            1K-blocks      Used Available Use% Mounted on
none                   516800         0    516800  0% /tmp
/dev/mmcblk0p1        3801088         4   3801084  0% /mnt/sdcard
/dev/sda1             514760         8    514752  0% /mnt/memstick
zynq> ls /mnt/memstick
test.txt
zynq>
```

Figure 44 – SD Card and Memory Stick 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.

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

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

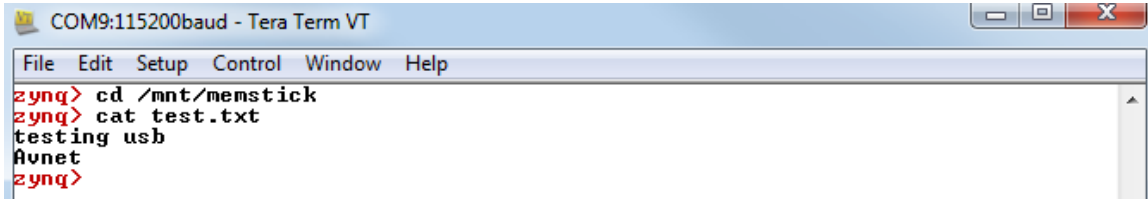


Figure 45 – Reading a Text File from Memory Stick

69. 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 "PicoZed is Awesome" > new.txt
zynq> ls
zynq> cat new.txt
```

```
zynq> cd /mnt/memstick
zynq> ls
test.txt
zynq> echo "PicoZed is Awesome" > new.txt
zynq> ls
new.txt  test.txt
zynq> cat new.txt
PicoZed is Awesome
zynq> █
```

Figure 46 – Writing a Text File to a Memory Stick

70. 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.

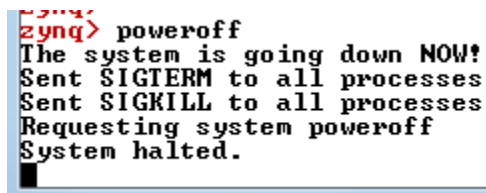
71. Remove the memory stick. Plug it into the PC and verify the `new.txt` file is there.
72. Repeat steps 69 through 71 for the microSD card and mount point `/mnt/sdcard`.

Poweroff

When you are done experimenting, power off Linux and the boards.

73. Linux should be properly shut-down.

```
zynq> poweroff
```



```
zynq> poweroff
The system is going down NOW!
Sent SIGTERM to all processes
Sent SIGKILL to all processes
Requesting system poweroff
System halted.
```

Figure 47 – PicoZed Linux Shutdown

74. Turn the power switch (SW7) to the OFF position.

To further examine PicoZed and the PicoZed FMC Carrier, please go to www.picozed.org
→ Support → Reference Designs/Tutorials → PicoZed FMC Carrier

To complete the tutorials, you will need to install Xilinx development tools. For instructions on installing the Xilinx software, please refer to Appendix A: Format the microSD Card.

Getting Help and Support

Avnet Support

The PZCC-FMC is a versatile development kit that allows evaluation of the PicoZed SOM, which can help you adopt PicoZed into your next design. All technical support is offered through the PicoZed.org website support forums. PicoZed users are encouraged to participate in the forums and offer help to others when possible.

<http://picozed.org/forums/>

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

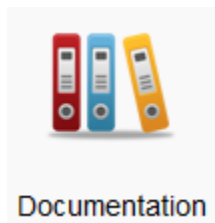
PicoZed.org Web Master – webmaster@PicoZed.org

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

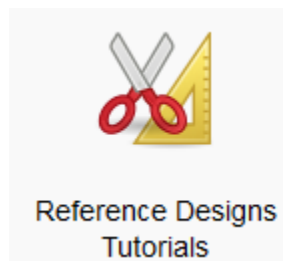
www.PicoZed.org/content/support

Once on the PicoZed.org support page:

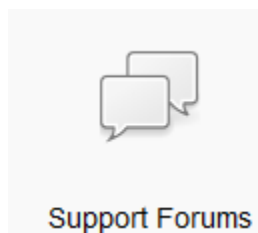
To access the latest PicoZed documentation, click on the Documentation link:



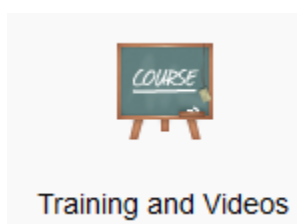
To access the latest reference designs for PicoZed, click on the following link:



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



To view online training and videos, 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 PicoZed 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.

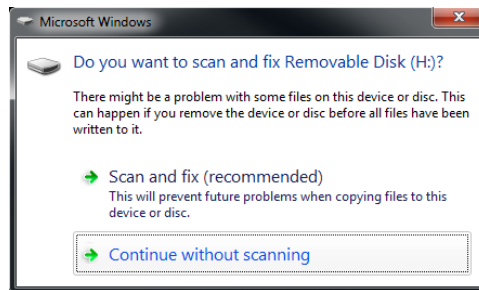


Figure 48 – Windows Prompt for Scanning and Fixing an SD Card

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**.

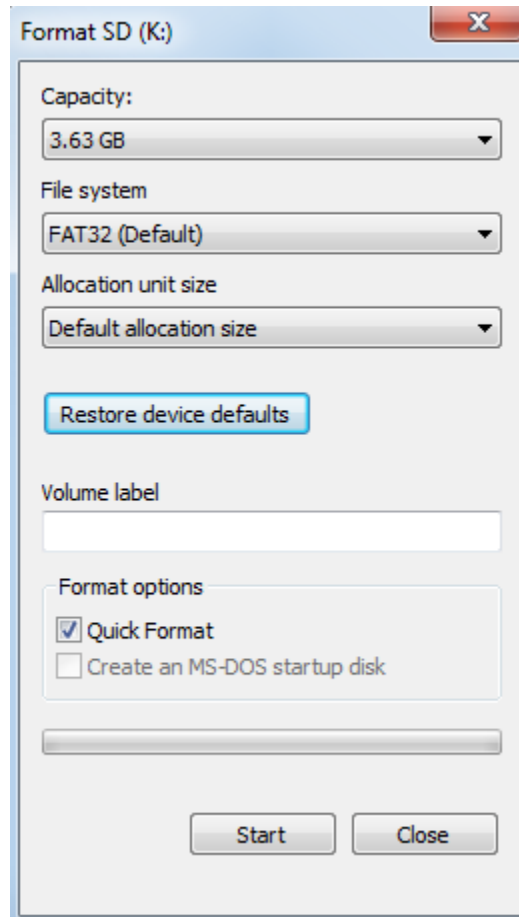


Figure 49 – Format the microSD Card

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

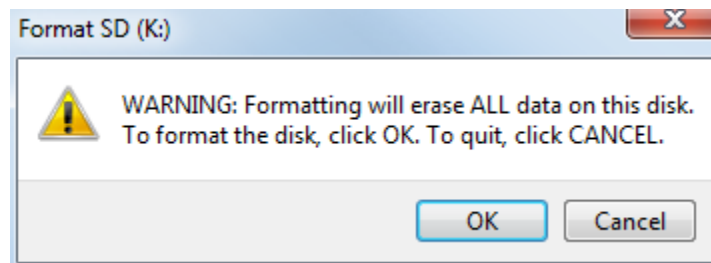


Figure 50 – Formatting Will Erase

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

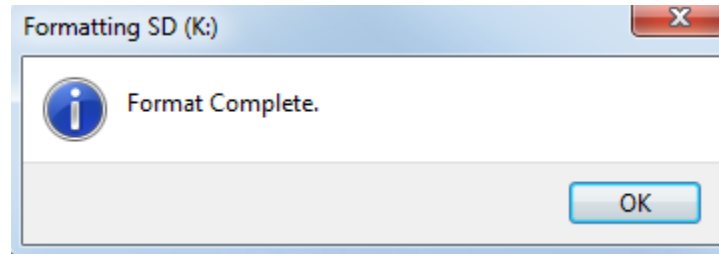


Figure 51 – Format Complete

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.

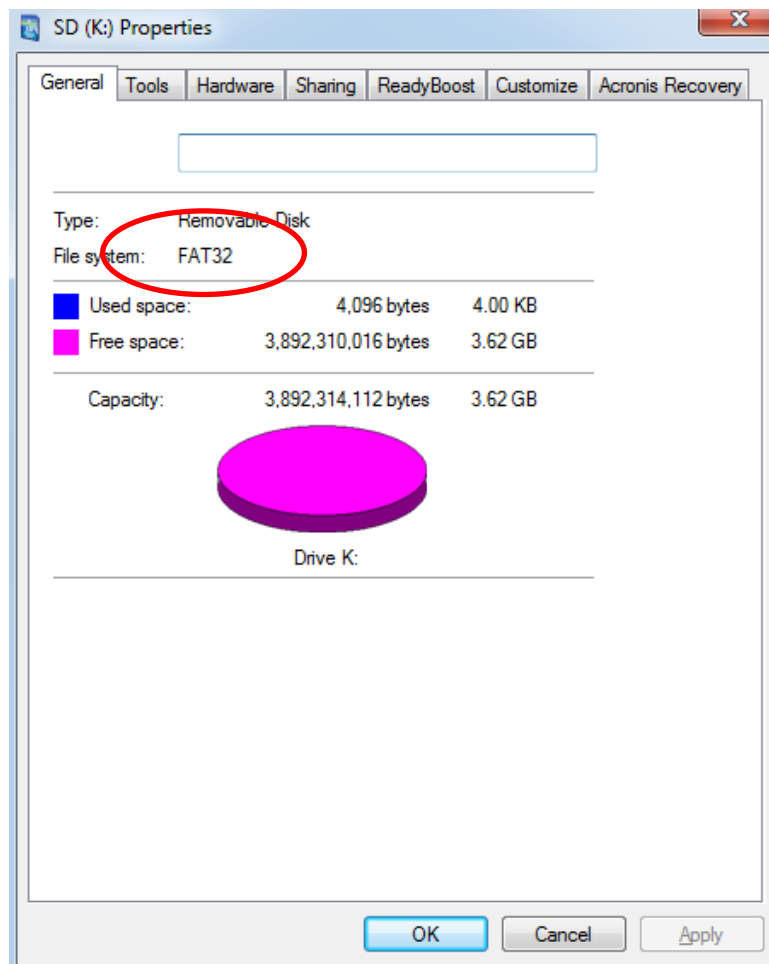


Figure 52 – Properties of the microSD Drive

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**.

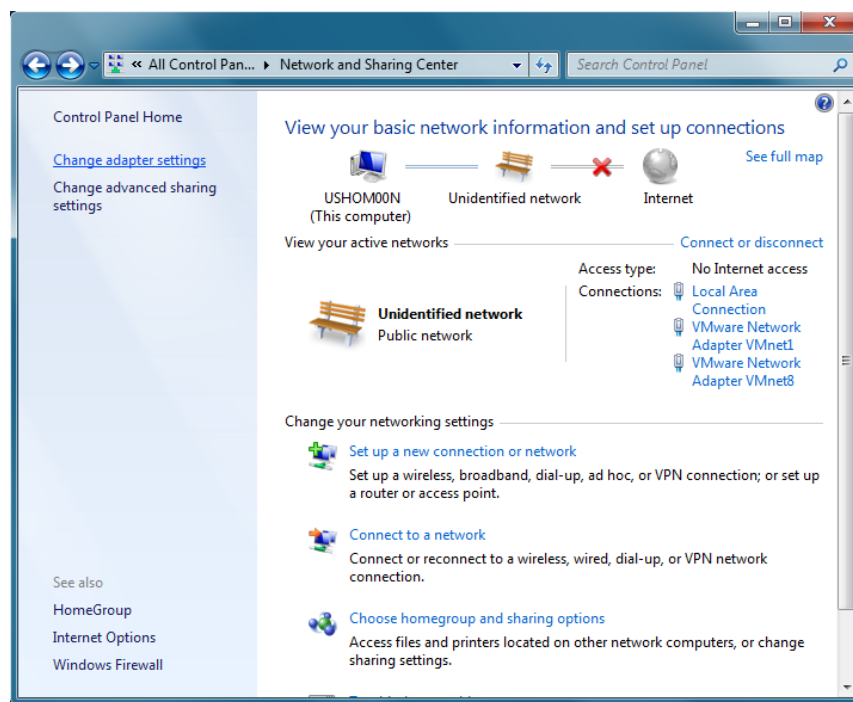


Figure 53 – Network and Sharing Center

3. 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**.

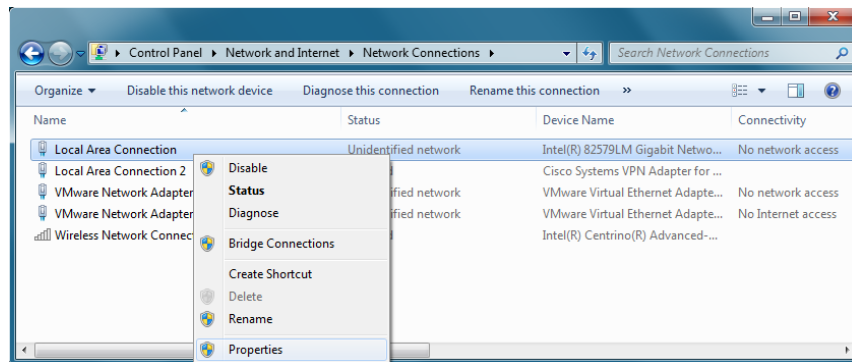


Figure 54 – Network Connections

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

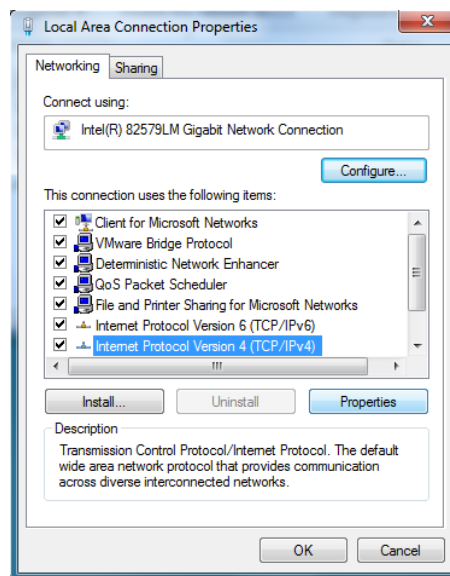


Figure 55 – Local Area Connection Properties

5. 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.

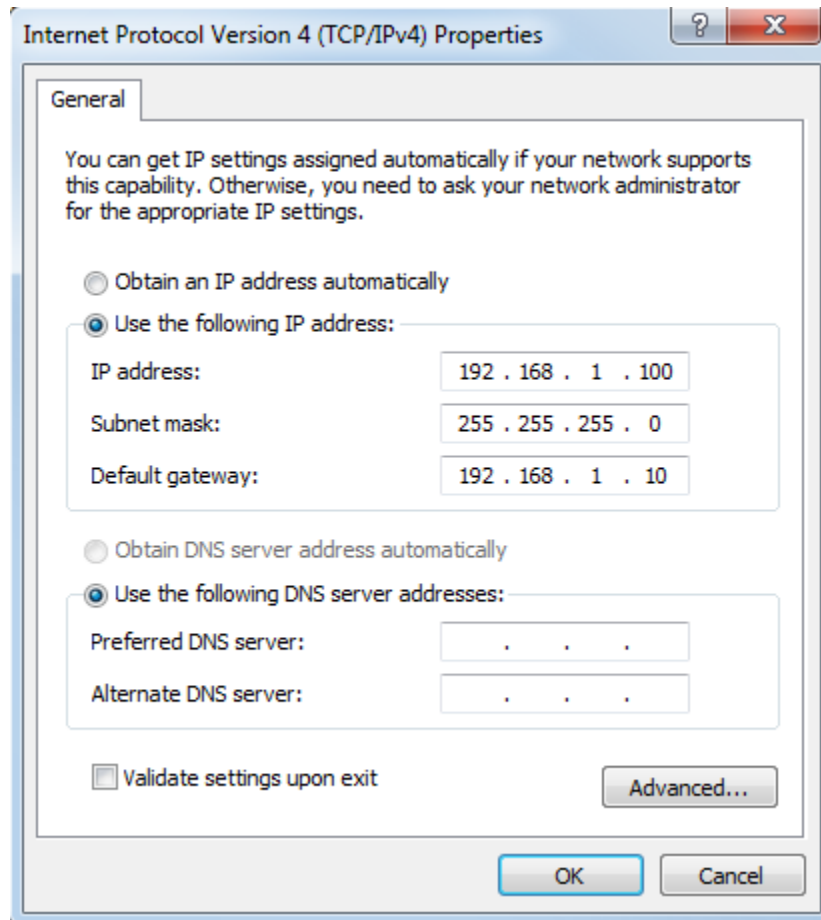


Figure 56 – Internet Protocol Version 4 (TCP/IPv4) Properties

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.

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 Suite, WebPack Edition

The four Zynq devices available within the PicoZed SOM family are all supported in Vivado Design Suite, WebPack Edition. 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

Although free, WebPack still must be licensed. To obtain your free license, visit

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

If a full set of Vivado System or Design 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 latest version of **Vivado Design Suite User Guide Release Notes, Installation, and Licensing (UG973)**. The 2014.4 version is available on the Xilinx website at:

http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_4/ug973-vivado-release-notes-install-license.pdf