

**Xilinx® PetaLinux  
for the  
Avnet Zynq®-7000 All Programmable SoC  
Mini-ITX Development Kit  
Reference Design**



**Version 4.0  
December 2015**

## Table of Contents

Introduction .....	2
Objectives .....	3
Reference Design Requirements .....	3
Software .....	3
Hardware .....	3
Supplied Files .....	4
Setting up the Zynq-7000 AP SoC Mini-ITX Development Kit .....	5
Network Configuration .....	7
Installing the UART Driver and Virtual COM Port .....	7
Installing a Serial Console on a Windows 7 Host .....	7
Running the Demo Files .....	8
Boot Linux from Micro SD Card .....	8
HTTP Server Demo .....	10
Telnet Server Demo .....	11
GPIO Demo .....	12
Create the Development Environment .....	13
Set up the TFTP Server .....	15
Install the PetaLinux Tools .....	16
Licensing .....	16
Setting PetaLinux Environment Variables .....	17
Vivado Hardware Platform and Block Diagram .....	18
Vivado Design Suite 2014 Tips .....	20
PetaLinux Board Bring-up Overview .....	22
Create a New PetaLinux Software Platform and BSP .....	22
Configure the PetaLinux Kernel (optional) .....	27
Configure the PetaLinux Root File System .....	28
Modify the Device Tree .....	30
Build the PetaLinux System Image .....	32
Create the BOOT.BIN File .....	33
Boot PetaLinux from the microSD Card .....	34
Appendix I: Installation of USB UART Driver .....	35
Download and Install the Required Software (for Windows) .....	35
Determining the Virtual COM Port (on Windows) .....	37
Appendix II: References .....	39
Appendix III: Useful Linux commands .....	40
Revision History .....	41

## Introduction

This document describes a Xilinx Zynq®-7000 All Programmable(AP) SoC Processor System (PS) integrated with Programmable Logic (PL) peripherals, implemented on the Avnet Zynq-7000 AP SoC Mini-ITX Development Kit, and running a Linux operating system created with the Xilinx PetaLinux tool chain.

The Linux development for this example design was performed with the PetaLinux toolchain installed on a 64-bit Ubuntu 14.04 Linux virtual machine within a Windows™ 7 Operating System host PC.

The reader is expected to possess some knowledge of Linux and Linux commands, and be comfortable working with Linux command-line tools. For instructions on creating a Virtual Machine with Linux, please see the **VirtualBox and Linux VM Installation Guide** link in **Appendix II: References**.

In any system level design, hardware and software must interact efficiently to create a winning solution. In a real-world system, many elements should be considered in your hardware platform specification. Using a Zynq-7000 AP SoC allows you to easily prototype many hardware combinations to see how they compare. Factors to consider include:

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

In this example a reliable hardware platform is supplied in order to demonstrate the Linux operating system built with the PetaLinux tool chain, including support for GPIO peripherals integrated into the Programmable Logic (PL). A Xilinx Vivado® Design Suite project with complete source for the platform is included, for hardware designers who wish to more closely inspect the PL or make additional customizations.

The kernel version of Linux used in the reference design is 3.19.

## Objectives

This reference design illustrates how to:

- Create a Linux BSP customized for the hardware platform.
- Use the PetaLinux tools to create customized second-stage boot loader and Linux kernel images.
- Customize the PetaLinux root file system to include a GPIO user application.
- Execute the resulting output products on a Zynq-7000 AP SoC Mini-ITX Development board to boot Linux from a microSD card.
- Demonstrate a few basic applications included by the PetaLinux tool chain (gpio-demo, web server, ssh).

## Reference Design Requirements

### Software

The software requirements for this reference design are:

- 64-bit Linux (can be Linux installed in a virtual machine on a 64-bit Windows host)
- PetaLinux Tools 2015.2.1
- TFTP Server (required by PetaLinux, but not used in the microSD boot)
- Silicon Labs CP2102 USB-UART bridge
- Serial terminal emulation software such as Tera Term

### Hardware

The hardware requirements for this reference design are:

- 64-bit host computer with at least 3/6 GB RAM (typical z7045/z7100) and up to 5/10 GB RAM (peak) available for Windows-7 or Linux operating systems with Vivado Design Suite
  - <http://www.xilinx.com/design-tools/vivado/memory.htm#zynq-7000>
- Avnet Zynq-7000 AP SoC Mini-ITX Development Kit (includes)
  - Zynq-7000 AP SoC Mini-ITX SoC 7045/7100 Development Board
  - ATX Power Supply
  - USB-A to USB-micro B cable for serial console
  - microSD card minimum 4 GB/class 4, formatted FAT32
- USB-A to USB-micro B cable for USB UART (Optional, used for JTAG)
- Cat-5 Ethernet cable
- Host PC with 10/100/1000 compatible Ethernet NIC
- Host PC or network router to act as DHCP server

## Supplied Files

The support files for the Zynq-7000 AP SoC Mini-ITX 7045/7100 PetaLinux reference design are each delivered in a single compressed zip file:

**Zynq\_Mini\_ITX\_7z045\_2015\_2\_PetaLinux\_Ref\_Design.zip**  
or  
**Zynq\_Mini\_ITX\_7z100\_2015\_2\_PetaLinux\_Ref\_Design.zip**

PetaLinux development is performed on a Linux host system, so the recommended development environment is to install the PetaLinux tools on a Linux host or VM. Development of the hardware platform used in this reference design was done in Vivado Design Suite, which can be installed on either a Windows or Linux host system. Once the hardware platform is exported as a hardware definition file (.hdf), it can be imported directly into a Petalinux project and thus Vivado not required for this reference design.

Depending on your preference, you may extract the archive file using a Windows or Linux host. All the steps can be performed on a Linux development system, but if you are more familiar with Windows and wish to copy files to the microSD card there, you can. You will just need to have the ability to copy files between your Windows and Linux development hosts. The PetaLinux tools must be installed on Linux.

The following directory structure is included with this reference design, where 7znnn is 7z045 or 7z100.

**Zynq\_Mini-ITX\_PetaLinux\_2015\_2.pdf:** this document.

**boot\_source:** Contains the files to create the BOOT.BIN image required to boot from the micro SD card:

**system-top.dts:** Device tree source file with the Zynq Mini-ITX Ethernet PHY added.

**u-boot.elf:** U-boot binary used as the second stage loader to boot Linux.

**zynq\_design\_1\_wrapper.bit:** The bitstream for the Vivado hardware design.

**zynq\_mini-itx\_fsbl.elf:** The golden ARM executable for the Zynq FSBL.

**sd\_card:** Contains image files needed to boot PetaLinux from microSD:

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

**image.ub:** Linux kernel, device tree blob and root file system.

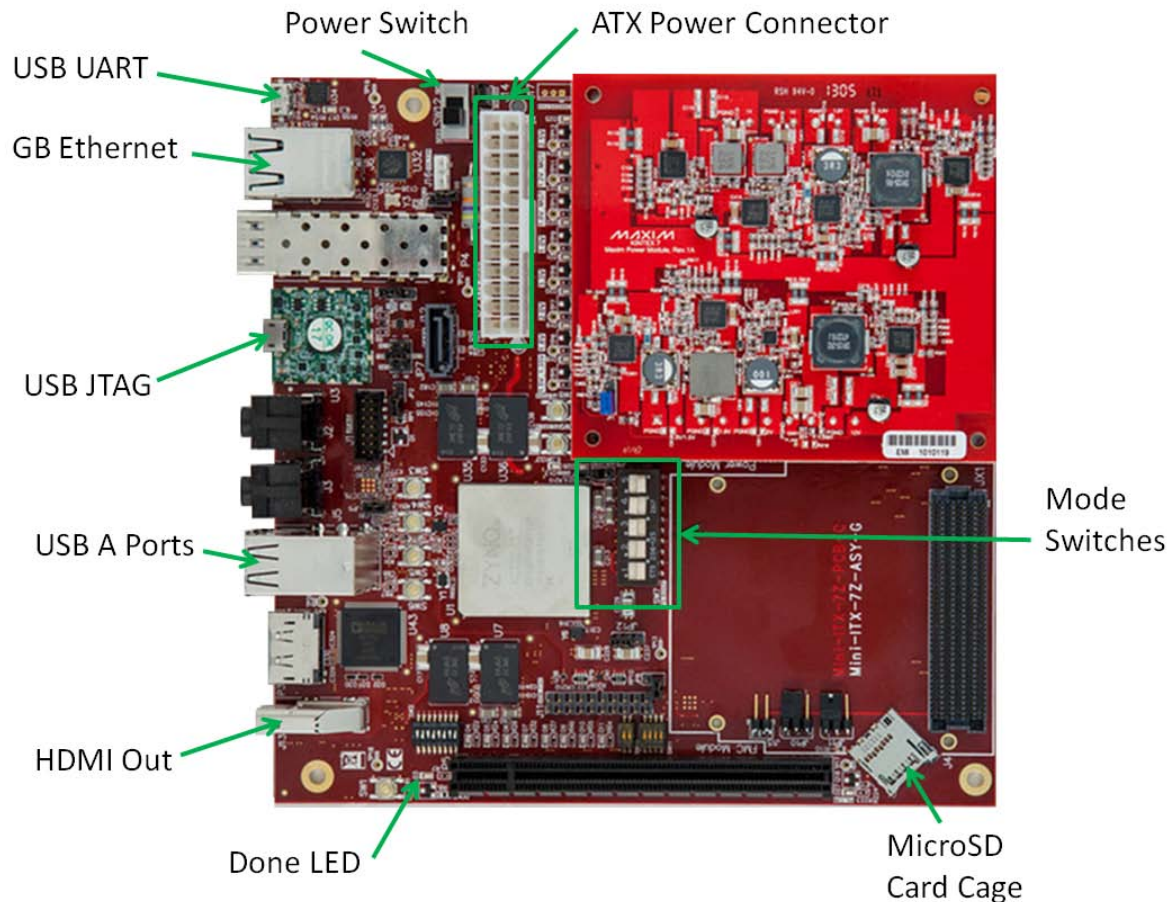
**vivado:** Compressed archive for the hardware platform for the Zynq-7000 AP SoC Mini-ITX 7045/7100.

**petalinux\_mitx\_design\_7znnn\_2015\_2.zip:** This archive is not required for this reference design, but is provided to allow hardware designers complete access to the original source. The platform can be modified and built in Vivado, and the resulting solution can be exported to a hardware definition file (make sure to include the bitstream).

**zynq\_design\_1\_7znnn.hdf:** Hardware description file exported from Vivado for import into the PetaLinux project.

## Setting up the Zynq-7000 AP SoC Mini-ITX Development Kit

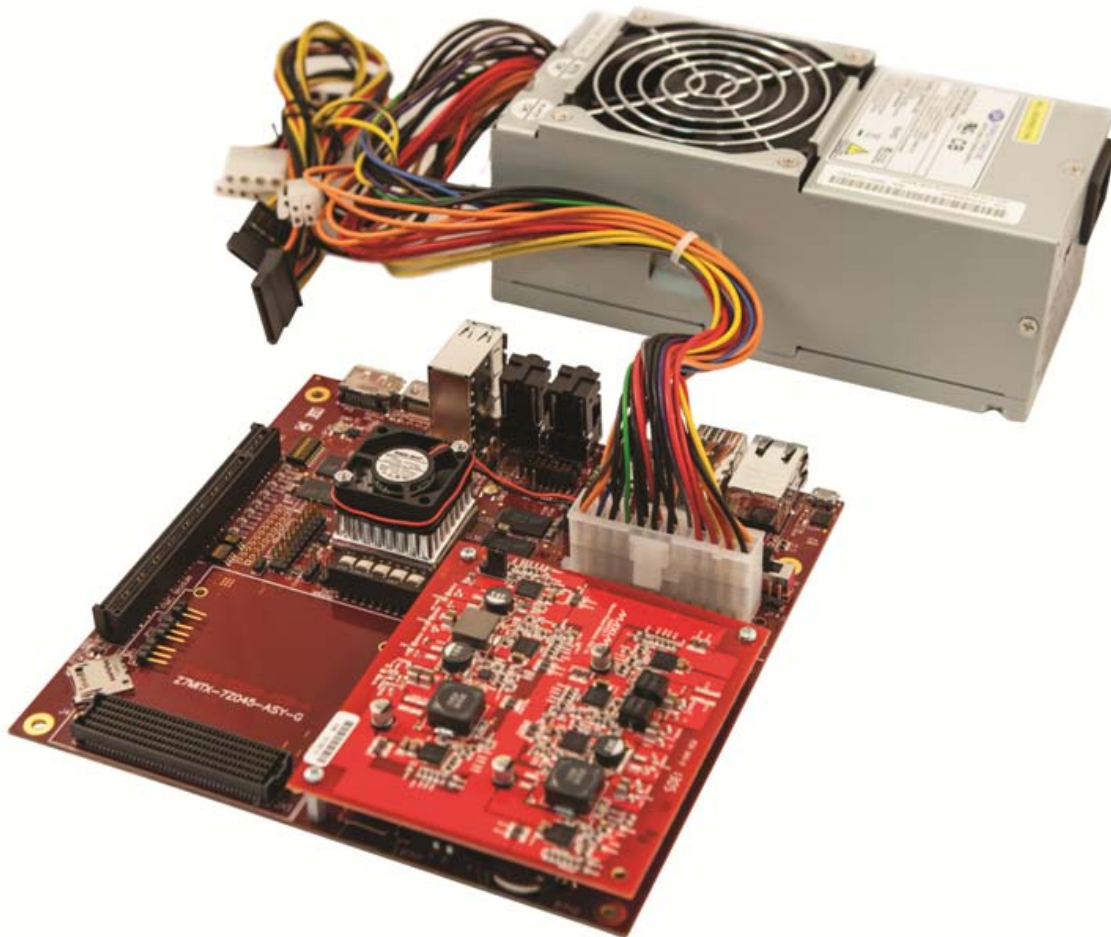
Refer to the following figure and perform the steps to set up the board for running applications from the microSD card.



**Figure 1 – Zynq-7000 AP SoC Mini-ITX Development Board**

1. Plug a USB A-micro B cable between the host PC and the USB UART port (J7) on the Zynq-7000 AP SoC Mini-ITX.
2. For JTAG connectivity (optional), plug a second USB A-micro B cable between the host PC and the USB JTAG port on the Zynq-7000 AP SoC Mini-ITX.
3. Set the boot mode switches (SW7[1:5]) to [off, off, on, on, off] to configure the Zynq-7000 AP SoC Mini-ITX to boot from the microSD card. Set the boot mode switches (SW7[1:5]) to [off, off, off, off, off] to configure the Zynq-7000 AP SoC Mini-ITX to boot from JTAG (shown in Figure above). On position is where the switch is closest to the number on the block (towards the cooling fan).
4. Connect a CAT-5 cable between the GB Ethernet Port (J6) on the Zynq-7000 AP SoC Mini-ITX and a NIC on your development host. Optionally, you may connect the board instead to a LAN router or switch, which may be more convenient for providing a DHCP address to PetaLinux.

5. Connect the ATX power supply to the power connector (P2). Insert the 4-wire connector first, on the end nearest the board edge, followed by the larger connector. Push the large connector down until the retainer clip snaps into place.



**Figure 2 – Zynq-7000 AP SoC Mini-ITX Development Board with ATX Power Supply**

6. Set the board power switch to the ON position. You must see 8 green power LEDs illuminate. If you do not, check that the ATX two-piece power connector is firmly seated, and that the tabs on the smaller portion (4 wire) of the connector is beneath the larger part. Also make sure the power module is firmly seated on the board, and that all the connector pins are in good condition. Do not proceed until you see all 8 power LEDs illuminate.
7. Set the board power switch to the OFF position.



## Network Configuration

Follow the figure below to configure the Ethernet NIC of the development host PC to establish an Ethernet connection directly with the Zynq-7000 AP SoC Mini-ITX Development Board. You may also plug the board directly into a router or switch on a LAN served by DHCP.

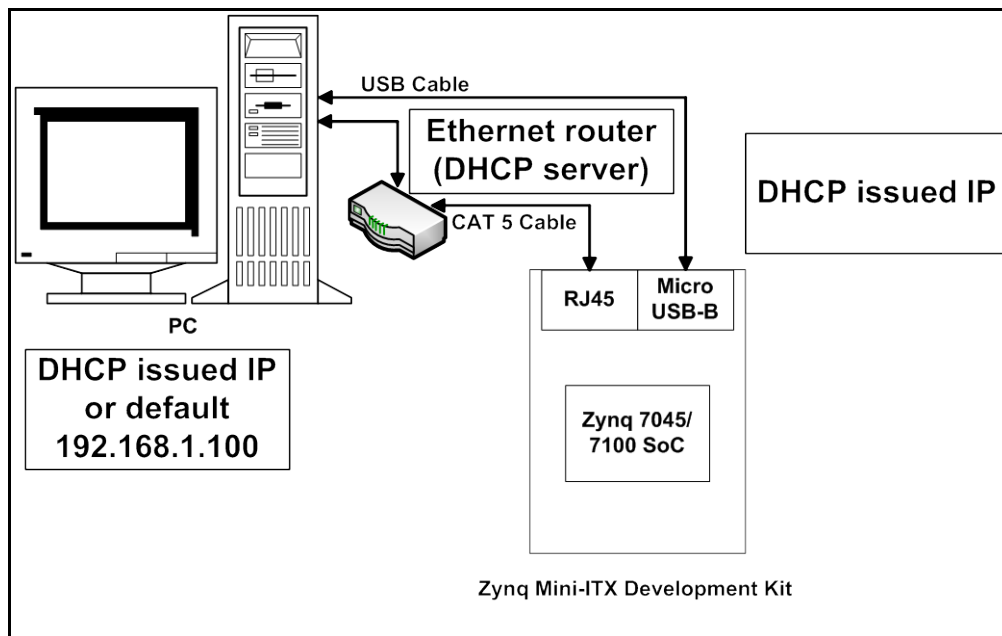


Figure 3 – Network Connection Setup

1. If you choose to connect the Zynq-7000 AP SoC Mini-ITX Development Board directly to the host PC and are not running a DHCP server you will need to configure the host PC with an IP address of **192.168.1.100** and a subnet mask of **255.255.255.0**.
2. If you choose to connect the Zynq-7000 AP SoC Mini-ITX Development Board directly to the host PC and your PC has a 1000 Mbps capable NIC, you may have to manually set the line rate to **100 Mbps full-duplex**. Gigabit Ethernet does not always auto-negotiate with a direct connection between the Zynq Mini-ITX board and host PC.

## Installing the UART Driver and Virtual COM Port

If the Zynq-7000 AP SoC Mini-ITX Development Kit has not been connected to the host PC before, it may be necessary to install the software driver for the virtual COM port. The driver installation for the Silicon Labs CP2102 USB-UART bridge on the Zynq-7000 AP SoC Mini-ITX Development Kit is described in detail in **Appendix I: Installation of USB UART Driver**.

## Installing a Serial Console on a Windows 7 Host

Starting with Windows 7, Microsoft no longer includes the HyperTerminal terminal emulator software. This example design requires use of terminal emulation software for a serial console connection to the Zynq-7000 AP SoC Mini-ITX Development Board. A suitable free and open-source replacement for HyperTerminal is Tera Term. A download link for Tera Term can be found in **Appendix II: References**.



## Running the Demo Files

You can configure the Zynq 7000 Programmable Logic (PL) and boot PetaLinux by copying the supplied files in the **sd\_card** folder for your target to a microSD card. You must have the hardware set up and connected as described in **Setting up the Zynq-7000 AP SoC Mini-ITX Development Kit**. Be sure to set the boot mode switches to SD card mode (SW[1:5]=off, off, on, on, off).

### Boot Linux from Micro SD Card

1. Extract the **Supplied Files** archive to any folder on your development system (Windows or Linux). The top level folder will be designated **<installation>** for this document.
2. Insert a microSD card into an available slot<sup>1</sup> on the host PC. The micro SD card should be formatted FAT32.
3. In the file browser of your host system, navigate to the **<installation>/sd\_card** folder and copy the **BOOT.BIN** and **image.ub** files in that folder to the microSD card.
4. In the file browser, eject/unmount the microSD drive so the card can be safely removed from your system.
5. Remove the microSD card from the host PC and insert it into the microSD card cage on the Zynq-7000 AP SoC Mini-ITX Development Board.
6. Verify the board is set up correctly as described in **Setting up the Zynq-7000 AP SoC Mini-ITX Development Kit**. Be sure to set the boot mode switches to SD card mode (SW[1:5]=off, off, on, on, off).
7. Slide the power switch to the ON position. You will see 8 green 'power good' LEDs lit near the power connector.
8. Start a serial terminal session and set the serial port parameters to **115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control.
9. Almost immediately you should see the blue DONE LED (D3) light, indicating the PL has been configured with the bitstream. You will then see U-boot start, and after a 4 second delay U-boot will then load and execute the Linux kernel.

---

<sup>1</sup> On some host systems, you may need to use a USB-to-SD card adapter.

10. Once the kernel has finished booting you should see the following in your serial terminal window. At startup PetaLinux is configured to fetch an IP address from a DHCP server. If there is no DHCP server present the DHCP request will timeout and the board IP address will have to be manually configured for network access. Note the address assigned for use in the demos that follow.

```

COM4:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
Sending discover...
Sending discover...
macb e000b000.ethernet eth0: link up (1000/Full)
IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
Sending discover...
Sending select for 192.168.1.110...
Lease of 192.168.1.110 obtained, lease time 86400
/etc/udhcpd.d/50default: Adding DNS 192.168.1.1
done.
Starting Dropbear SSH server: Generating key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCSNKngPQDvU1Z2xL1m9xn3BWtk/3QrUINzXNSSE/VJ
zylxB+ZL3PoFDzrSZO1Hmt/5BtN6qOyHejL2URnG8hftkr4mBa9xcqHY3JeomXG7yaGaJndG55Y5NI7N
2M8NLUWaWEJBCJZYVUyc1YD+PpfPIsSSyUoiDqosSujnQmGUmU6cgJhk9fTmxuLbz3s2icc3nYiJnv35
hOYRQqUS8IovLPXcHjRGZNecnP20z0ZFF/wrQUhOowsoNFKSwZL7/ZmLXiYyE0N5QdMW4fqJfeN2iKDa
n2dxhq69kBDVCMNidh5jSmLK9Nqe+mLup8YazDO9Y01S4KUE2EFYrVYiZAEB root@MITX_7045_2015
_2_1
Fingerprint: md5 17:5e:5d:f1:89:b3:1e:50:8c:81:68:dd:62:79:d0:17
dropbear.
Forcing USB OTG to host mode (enabling +5V power rail): OK
Starting tcf-agent: OK

Built with PetaLinux v2015.2.1 (Yocto 1.8) MITX_7045_2015_2_1 /dev/ttyPS0
MITX_7045_2015_2_1 login:

```

Figure 4 – PetaLinux Boot Console

11. Login to the kernel with the following credentials:
- User: **root**
- Password: **root**
12. Feel free to experiment with running Linux commands and utilities. Some common Linux commands are provided in **Appendix III: Useful Linux commands**. Various popular services and daemons, such as telnet and the µweb server, are running in the kernel.

## HTTP Server Demo

The web server implements only a subset of the HTTP 1.1 protocol and demonstrates the control and monitoring an embedded system via a browser.

1. Open an HTML browser and point to the URL [http://<board\\_IP\\_address>](http://<board_IP_address>), where [<board\\_IP\\_address>](http://<board_IP_address>) is the IP address of the board that was served by the DHCP server during boot or manually configured. The webserver demo should appear in your browser similar to the following figure.



Figure 5 – PetaLinux Web Server

2. Navigate the web pages to explore the implemented web page features. Close the browser when you are done.

## Telnet Server Demo

This is a simple application to echo whatever is sent to the program over the network.

1. Open a command window on your host system, and type:  
`telnet <target IP address>`
2. The telnet server will open the connection and present the user with the login prompt (user 'root', password 'root'). Although not secure, a telnet connection is often used to control an embedded system remotely during development. Try this out for yourself. Type `exit` to close the telnet session. Close the command window.



Figure 6 – Telnet Initiated from Windows

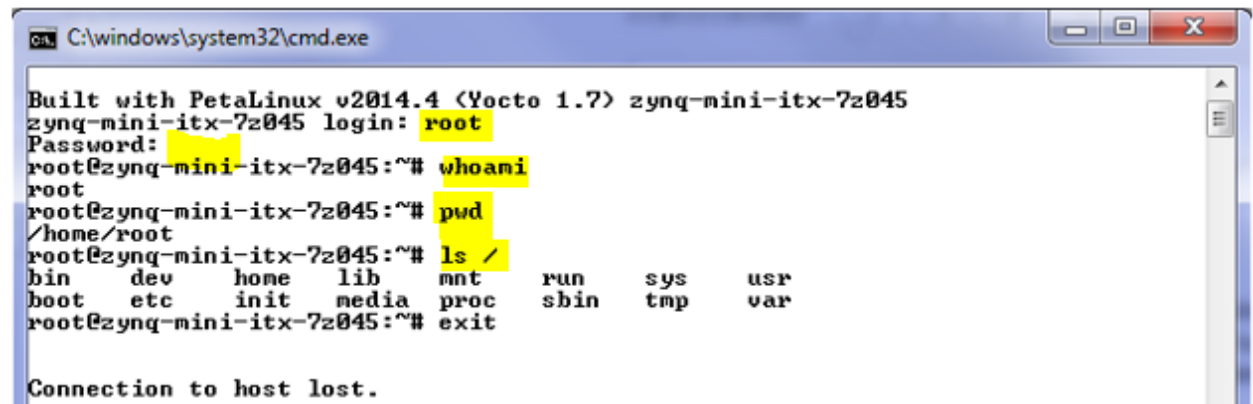


Figure 7 – Telnet Logged In to PetaLinux Zynq-7000 AP SoC Mini-ITX

## GPIO Demo

PetaLinux provides a simple GPIO demo example application that can optionally be configured as an application included in the root file system. It allows the interaction with the switches and LEDs on the Zynq-7000 AP SoC Mini-ITX board via the command line.

1. If you haven't already, login to PetaLinux using the provided credentials.
2. Before we can use the gpio-demo application we need to identify the location of the GPIO peripherals. Enter the following command to print the kernel boot and diagnostic messages and pipe the output through a grep filter to locate the GPIO peripherals in the system:

```
# dmesg | grep gpio
root@MITX_7045_2015_2_1:~# dmesg | grep gpio
XGpio: /amba_pl/gpio@41200000: registered, base is 903
XGpio: /amba_pl/gpio@41210000: registered, base is 895
XGpio: /amba_pl/gpio@41220000: registered, base is 887
root@MITX_7045_2015_2_1:~#
```

Figure 8 – GPIO Address Map

3. On the Zynq-7000 AP SoC Mini-ITX board there are 3 pushbutton switches (SW6, SW5, SW4), 8 red User LEDs, and 8 DIP switches. The locations of these peripherals can be inferred from the output of the dmesg command.

Peripheral	Location
Pushbutton Switches (x3)	903
DIP Switches (x8)	895
LEDs (x8)	887

4. You may experiment with the gpio-demo application to turn on the LEDs and read the switches. Enter the following command to turn on all the LEDs at once:  
# gpio-demo -g 887 -o 0xff
5. Use the application to read the pushbutton switches. Press and hold one or more of the pushbutton switches (SW4, SW5, SW6) before entering the command:  
# gpio-demo -g 903 -i
6. Use the application to read the DIP switches.  
# gpio-demo -g 895 -i

When you are satisfied with the applications, you may power off the board and close the serial terminal window.

## Create the Development Environment

*The Xilinx SDK is no longer required.*

This example design requires PetaLinux tools be installed and licensed on a 64-bit Linux host PC. This can be a 64-bit Linux virtual machine running on a 64-bit Windows host, or a native install. For instructions on installing VirtualBox and a CentOS virtual machine, see the Appendices.

The PetaLinux development environment requires that a number of packages be installed in the OS. Installations require root privilege, and you may elevate the access level within the current Terminal window with the **su** command and root password (or use **sudo** if your account is so configured). When you are operating with elevated privilege, the command prompt changes to a hash sign (#). To return to your original privilege level, type **exit** at the command prompt.

```
[training@localhost ~]$ su
Password: 
[root@localhost training]# exit
exit
[training@localhost ~]$
```

You will need Internet access from your Linux host to download packages. Further, the OS installed from the virtual disk was created at a specific moment in time. Since that point, many installed packages have been updated, and it is prudent to ensure your system is up to date before attempting to add specific development packages. If you do not perform this update step, you may find some of the required packages fail to install correctly.

During any package installation or update process, you may be prompted by the software to verify that the requested operation should proceed. To successfully install all required components, always respond **y** (yes) to the decisions, as shown below. Some packages in the list may have already been installed during the virtual machine creation – you will simply get a message indicating there is nothing to be done for such packages.

Package	Arch	Version	Repository	Size
Installing:				
dos2unix	x86_64	3.1-37.el6	base	16 k

```
Transaction Summary
=====
Install      1 Package(s)

Total download size: 16 k
Installed size: 18 k
Is this ok [y/N]: y
```

**Figure 9 – Accept Package Installation Option**

Install packages at the root user prompt in the Terminal window with the following command:

CentOS: `yum install <package-name>`<sup>2</sup>  
 Ubuntu: `apt-get install <package-name>`

Additional packages are shown in the columns below. Use the names from the column that corresponds to your host environment.

Tool/Library	YUM/RPM Package for RHEL/CentOS/Fedora	APT Package for Debian/Ubuntu	RPM Package for SuSE
dos2unix	dos2unix	tofrodos	dos2unix
ip	iproute	iproute	iproute2
gawk	gawk	gawk	gawk
gcc	gcc	gcc	gcc
git	git	git-core	git-core
make	gnutils-devel	make	make
netstat	net-tools	net-tools	net-tools
ncurses	ncurses-devel	ncurses-dev libncurses5-dev	ncurses-devel
open-ssl	openssl-devel	libssl-dev	
tftp server	tftp-server	tftpd	tftp-server
zlib	zlib-devel	zlib1g-dev	zlib-devel
flex	flex	flex	flex
bison	bison	bison	bison
32bit libs	libstdc++.i686, glibc.i686, libgcc.i686, libgomp.i386, ncurses-libs.i686, zlib.i686	setup 32bit libs support in ubuntu, lib32z1, lib32ncurses5, lib32bz2-1.0, ia32gcc1, lib32stdc++6, libselinux1	32-bit runtime environment

**Figure 10 – Required Software Packages**

CentOS only: In addition to the packages above, also install:

**libselinux.i686**  
**libgomp.i686**  
**ncurses-libs.i686**

All packages should install without error, or should indicate that the most recent version is already in place.

<sup>2</sup> You may include multiple packages on one line, and you may append `-y` to answer yes automatically to all questions.



## Set up the TFTP Server

The u-boot boot loader can be configured to use a TFTP server to download the PetaLinux kernel image to the Zynq-7000 AP SoC Mini-ITX Development Board for testing. PetaLinux automatically copies the image files to the root of the TFTP server folder, so it is useful to have this set up even if we are booting in this example from an SD card. Additional instructions on the usage of TFTP with PetaLinux can be found the Avnet PetaLinux Speedway in **Appendix II: References**.

1. Log in to your Linux host , open a Terminal window and use the **su** command to elevate your privilege level (or use **sudo** in front of each command).
2. Create a directory in the root of your Linux host where the PetaLinux build files will be copied. By default this is `\tftpboot`<sup>3</sup>. Be sure to give this directory full access permissions for all users:

```
# mkdir /tftpboot
# chmod 777 /tftpboot
```

3. By default the TFTP server is disabled. To enable the TFTP server, create/edit<sup>3</sup> `/etc/xinetd.d/tftp` file using root privileges, replacing the word **yes** on the *disable* line with the word **no**. Also verify the correct directory is specified as `/tftpboot` on the *server\_args* line. Save the file and exit the editor.

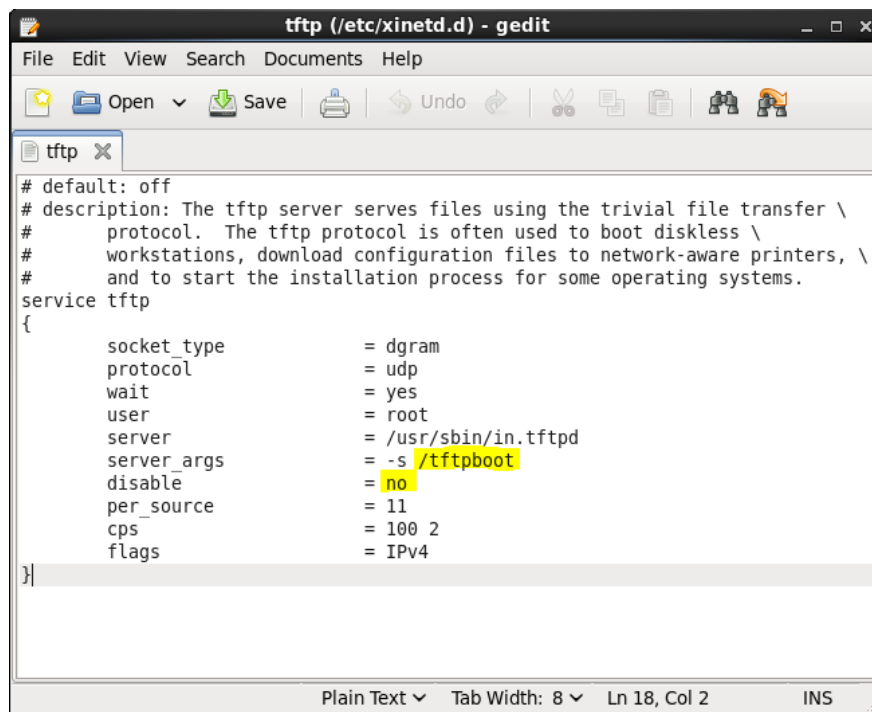


Figure 11 - TFTP Configuration File

4. Restart the daemon by invoking the run configuration script:

```
# /etc/init.d/xinetd restart
```

<sup>3</sup> You may use any installed editor to make the changes, or you may install another editor if you wish. Note that if you choose to use gedit, you may see warning messages as it is invoked, but these do not affect gedit operation.

5. This command will stop and then restart all of the services managed by the daemon on your Linux system. In addition to the **restart** command, you can also issue **stop** and **start** commands this way.

**Note:** If your Linux system is running Internet services on which other systems depend, restarting the daemon will cause a slight interruption in those services.

6. After executing this command, the TFTP server will be started on your system in response to incoming TFTP requests, and you can access any files you copied to `/tftpboot`.

Note that TFTP server requests may be blocked by Firewall software or router protection, and may not be accessible from outside the local sub-net without appropriate network configuration that is specific to your LAN. If you are using TFTP, it is best to have a DHCP server running on your Linux host, and to connect the Ethernet cable directly between the host and target.

## Install the PetaLinux Tools<sup>4</sup>

The **PetaLinux Tools Installation archive** can also be downloaded from the link in **Appendix II: References**. Install the PetaLinux Tools archive by entering the following commands in a Terminal window<sup>5</sup>.

```
sudo mkdir /opt/pkg
sudo <path to archive>/petalinux-v2015.2-final-installer.run /opt/pkg
```

## Licensing

The PetaLinux tools licensing is now handled as part of the installation process and there is no need to visit the Xilinx site for a no-cost license.

<sup>4</sup> For any large download, it is always a good idea to validate the MD5 checksum before using the archive.

<sup>5</sup> If you closed the Terminal window, open a new one and once again upgrade your privilege level to su.

## Setting PetaLinux Environment Variables

The PetaLinux tools require environment variables to be set before they can be used. Follow the instructions below to set these correctly.

### Notes

- PetaLinux recommends using the bash command shell. Set the default shell as bash or errors may occur building PetaLinux.
- You must source the settings scripts each time you open a new terminal window or shell.

1. Open a Terminal window on your Linux host.
2. Set the PetaLinux environment variables.
  - a. Open a command terminal window and navigate to the folder where the PetaLinux toolchain is installed:  

```
$ cd /<petalinux_install_path>/petalinux-v2015.2-final
```

(Default <petalinux\_install\_path> is /opt/pkg)

- b. Source the **settings.sh** file and verify the environment is correctly configured:  

```
$ echo $PETALINUX
```

```
training@training-VirtualBox:~$ cd /opt/pkg/petalinux-v2015.2-final/
training@training-VirtualBox:/opt/pkg/petalinux-v2015.2-final$ source settings.sh
PetaLinux environment set to '/opt/pkg/petalinux-v2015.2-final'
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
training@training-VirtualBox:/opt/pkg/petalinux-v2015.2-final$ echo $PETALINUX
/opt/pkg/petalinux-v2015.2-final
```

If you would like to make the environment settings automatic, instead of typing them each time you open a new terminal, you may edit the **.bashrc** file located in your home directory. In CentOS, you must be careful to ensure that the commands are activated only for a terminal shell and are NOT run during a login shell. If the latter occurs, your CentOS desktop environment will be compromised and you will need to remove the lines you added to **.bashrc** and reboot your Virtual Machine.

Add the lines highlighted below to your **.bashrc** file to check for a login shell, and execute the source commands only if the login shell condition is false.

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
shopt -q login_shell && echo 'Login shell' || source /opt/Xilinx/SDK/2014.4/settings64.sh
shopt -q login_shell && echo 'Login shell' || source /opt/pkg/petalinux-v2014.4-final/settings.sh
```

Figure 12 – CentOS .bashrc Example from 2014.4 (includes SDK – no longer required)

## Vivado Hardware Platform and Block Diagram

The hardware platform that underlies this software reference design was created using Vivado 2015.2. the hardware definition file required for the PetaLinux project is included and thus the Vivado tools are not required here. However, the complete Vivado design is available under the **vivado** folder in the **Supplied Files** archive, if you wish to build it yourself.

The hardware design contains the elements shown in the block diagram. Peripherals required by PetaLinux are:

- Xilinx Zynq 7Z045/7Z100 SoC
- 1GB of DDR3 SDRAM (32 MB minimum required by PetaLinux)
- Micro-SD Card (non-volatile boot memory required by PetaLinux)
- Gigabit Ethernet (Optional Network connection for PetaLinux)
- USB UART Port (Required by PetaLinux for the console)
- LEDs and Pushbutton Switches
- Interrupt Controller
- Triple Timer Counter (Required by PetaLinux)

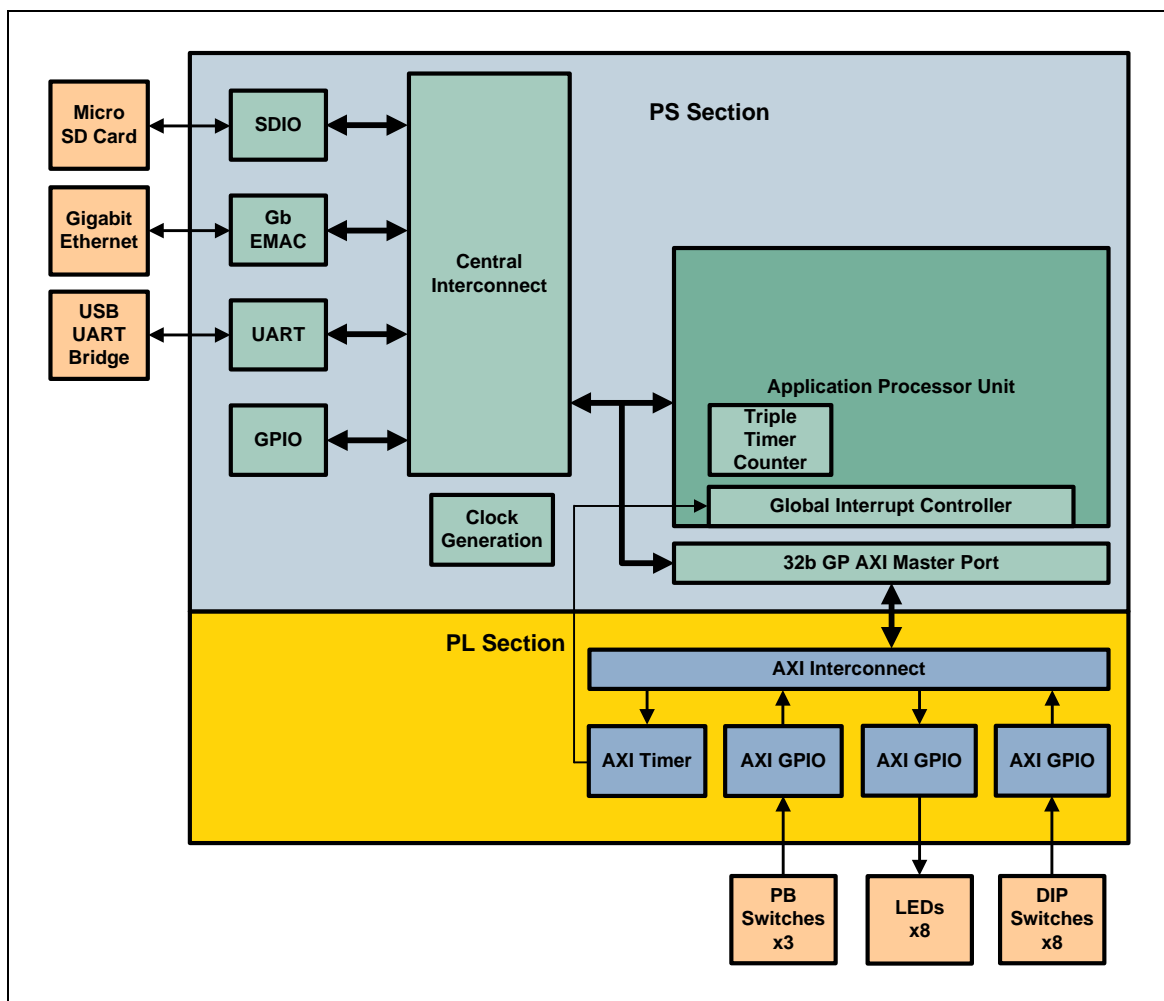


Figure 13 - Zynq-7000 AP SoC Mini-ITX Hardware Design Block Diagram

Below is the Block Diagram as it is seen in the Vivado Design Suite.

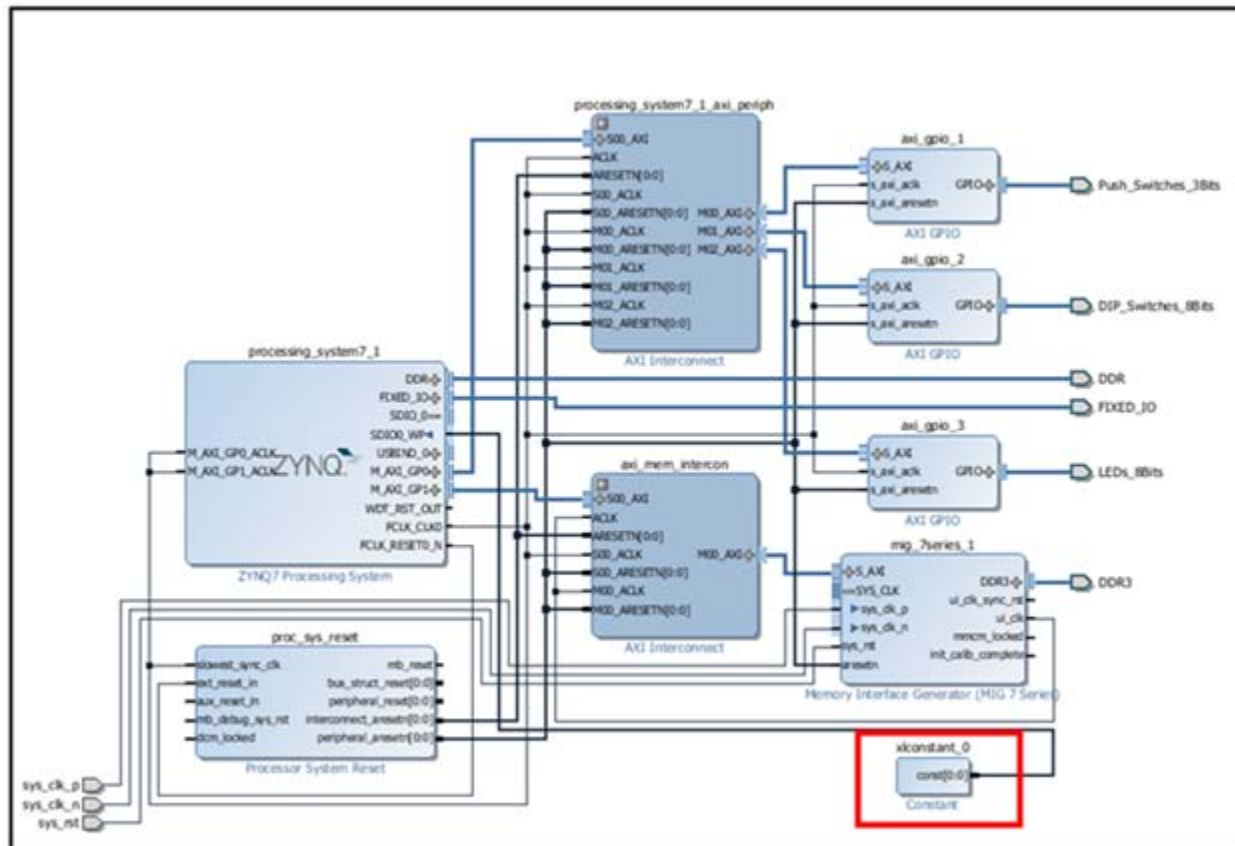


Figure 14 - Zynq-7000 AP SoC Mini-ITX Vivado Block Diagram

There is one additional peripheral not shown in the generic block diagram, labeled **xlconstant\_0**. This is a constant block tied to the write-protect pin of the SDIO peripheral in the Zynq PS. The constant drives the WP pin to active low, which emulates a write-protect switch set to the read/write position.

Linux requires that the memory containing the root file system be read/write, and the kernel checks the write-protect pin to ensure this is the case. A microSD card has no write-protect pin, so it is necessary to emulate this feature of a standard SD card to satisfy the Linux kernel and allow the boot to proceed.

## Vivado Design Suite 2014 Tips

### Windows 260 Character Path Limit

If you are using the Vivado Design Suite on a Windows-7 host, you may run into issues resulting from Vivado pathnames exceeding the maximum allowed. Vivado projects create a very deep file hierarchy, and it becomes very easy to violate the Windows limit if the project is not extracted near the root of the drive. This can even happen when the archive is decompressed, depending on where you choose to place the project in your existing file hierarchy.

It is not always convenient to place every Vivado project at the root of a drive. To work around this limitation, the recommended procedure is to place the Vivado archive in a shared folder on your host machine, then use Windows Explorer to map a network drive to the directory where the archive will be decompressed. This allows the Vivado project to be mapped to the root of the virtual (mapped) directory, eliminating any path issues.

Vivado also makes use of the Windows temp folder, which may be located several folders removed from the root drive, and this can also cause problems. You can create your own temporary directory in C:\temp, and force Vivado to use the new folder with the following TCL:

```
set_param "project.customTmpDirForArchive" C:/temp
```

For further information, see the Xilinx answer record at:

<http://www.xilinx.com/support/answers/52787.html>.

## Ensure the Triple-Timer Counter is Configured

The Vivado project used for PetaLinux is based on the Embedded Reference Design for the Zynq Mini-Module Plus, available from <http://www.zedboard.org>. The only change to the original project was to configure the Triple Timer in the processor system. If you omit the Triple Timer, you will not see any errors during the PetaLinux kernel build process, but when you try to execute your kernel on the target the boot will hang at **Starting kernel...**

```

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

Hash algo:    crc32
Hash value:   d4deb13d
Verifying Hash Integrity ... crc32+ OK
## Loading fdt from FIT Image at 01000000 ...
Using 'conf@1' configuration
Trying 'fdt@1' fdt subimage
  Description: Flattened Device Tree blob
  Type:        Flat Device Tree
  Compression: uncompressed
  Data Start:  0x01593320
  Data Size:   13414 Bytes = 13.1 KiB
  Architecture: ARM
  Hash algo:   crc32
  Hash value:  70b7bf27
  Hash algo:   sha1
  Hash value:  377946643197ce0bc0e30352ebf9ca317ab6e268
Verifying Hash Integrity ... crc32+ sha1+ OK
Booting using the fdt blob at 0x1593320
Uncompressing Kernel Image ... OK
Loading Device Tree to 07ff9000, end 07fff465 ... OK

Starting kernel ...

```

Figure 15 - PetaLinux System Missing Triple Timer

To ensure the Triple Timer Counter is configured, open the block diagram in your Vivado project<sup>6</sup> and double-click on the Zynq Processor System. You must see a checkmark beside the TTC entry, as shown below. Note that PetaLinux requires exclusive use of the first Triple Timer Counter in your system.

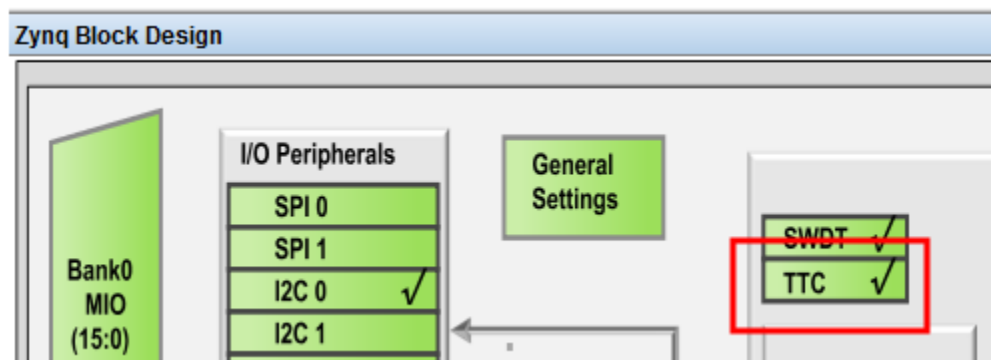


Figure 16 - Triple Timer Counter Configured in Vivado Project

<sup>6</sup> There is no need to do this with the supplied Vivado project, as the Triple Timer Counter has been configured properly. This applies only to your own custom projects.



## PetaLinux Board Bring-up Overview

The procedure to configure and boot PetaLinux on a Zynq target with a suitable hardware platform can be encapsulated by the following steps:

1. Create a PetaLinux Project and BSP for the target using the hardware description from Vivado.
  - a. Create new platform: **petalinux-create**
  - b. Configure software platform: **petalinux-config - - get-hw-description <options>**
  - c. Configure the kernel: **petalinux-config -c kernel**
  - d. Configure the RFS: **petalinux-config -c rootfs**
  - e. Build system image: **petalinux-build**
  - f. Generate the Zynq boot image: **petalinux-package**
2. Copy the **BOOT.BIN** and **image.ub** files to the top level of a FAT32-formatted microSD card.
3. Boot PetaLinux on the Zynq-7000 AP SoC Mini-ITX.
  - a. Connect the board cables and power.
  - b. Configure the board for booting.
  - c. Insert the microSD card.
  - d. Power the board.
  - e. Start a serial terminal for the PetaLinux console.
  - f. Log in to PetaLinux using the console.

## Create a New PetaLinux Software Platform and BSP

1. In the Terminal window, change to your desired working directory.

```
$ cd ~
```

```
training@training-VirtualBox:~$ cd ~
```

2. Copy the hardware description file appropriate to your target platform from the **vivado** folder of your **Supplied Files** to a directory on your Linux development system.

```
training@training-VirtualBox:~$ ls
BSP_Projects  Downloads  Pictures  Videos
Desktop       examples.desktop  Public   zynq_design_1_wrapper.hdf
Documents     Music       Templates
```

3. Create a new PetaLinux project folder ready for building a Linux system, fully customized for the bitstream created for your Zynq-7000 AP SoC Mini-ITX target. Since the project is based on the bitstream, it is target specific, so be sure to include the Zynq part number at the end of the name for reference.

```
$ petalinux-create --type project --template zynq --name <Project_Name>
```

```
training@training-VirtualBox:~$ petalinux-create --type project --template zynq --name zynq_mini-itx_7z045
INFO: Create project: zynq_mini-itx_7z045
INFO: New project successfully created in /home/training/zynq_mini-itx_7z045
```

Do not edit any of the files generated in the new directory. They will be overwritten each time you run the *petalinux-create* command.

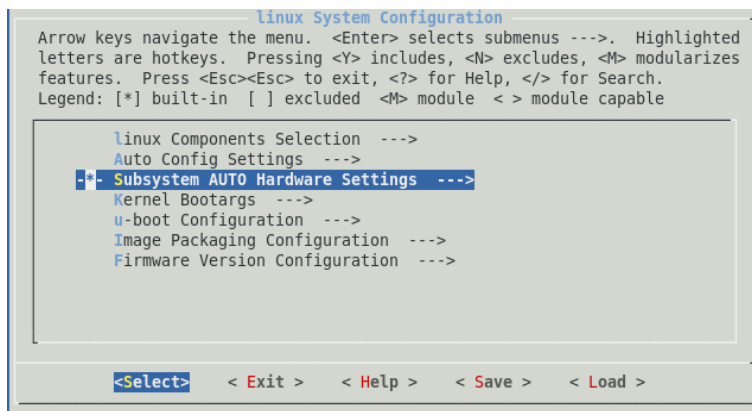
4. Import the hardware description into the PetaLinux project.

```
$ petalinux-config --get-hw-description -p <Project_Dir>
```

```
training@training-VirtualBox:~$ petalinux-config --get-hw-description -p zynq_mini-itx_7z045/
INFO: Checking component...
INFO: Getting hardware description...
INFO: Rename zynq_design_1_wrapper.hdf to system.hdf
```

```
***** hsi v (64-bit)
**** SW Build 1266856 on Fri Jun 26 16:35:25 MDT 2015
** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.
```

5. The PetaLinux menuconfig tool will launch automatically. This is where we can change project level configuration, including selecting the primary boot source. As of 2014.4, the SD card is the default, so this subsection simply illustrates how to change boot media if desired.
  - a. Use the *Down Arrow* key to scroll to select *Subsystem AUTO Hardware settings* and hit the *Enter* key.



- b. Scroll to *Advanced bootable images storage Settings* and if there is no asterisk between the square braces, use the *Space* bar to enable it. Hit the *Enter* key.

```

Subsystem AUTO Hardware Settings
Arrow keys navigate the menu. <Enter> selects submenus ---. Highlighted
letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module <> module capable

--- Subsystem AUTO Hardware Settings
  System Processor (ps7_cortexa9_0) --->
  Memory Settings --->
  Serial Settings --->
  Ethernet Settings --->
  Flash Settings --->
  SD/SDIO Settings --->
  [*] Advanced bootable images storage Settings --->

<Select> < Exit > < Help > < Save > < Load >

```

- c. Scroll to *boot image settings* and hit the *Enter* key.

```

Advanced bootable images storage Settings
Arrow keys navigate the menu. <Enter> selects submenus ---. Highlighted
letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module <> module capable

--- Advanced bootable images storage Settings
  boot image settings --->
  u-boot env partition settings --->
  kernel image settings --->
  jffs2 rootfs image settings --->
  dtb image settings --->

<Select> < Exit > < Help > < Save > < Load >

```

- d. Select *image storage media* and hit the *Enter* key.

```

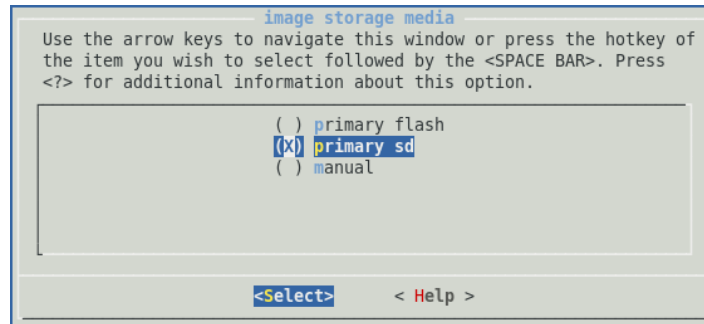
boot image settings
Arrow keys navigate the menu. <Enter> selects submenus ---. Highlighted
letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [ ] excluded <M> module <> module capable

  image storage media (primary flash) --->
    (boot) flash partition name (NEW)
    (BOOT.BIN) image name (NEW)

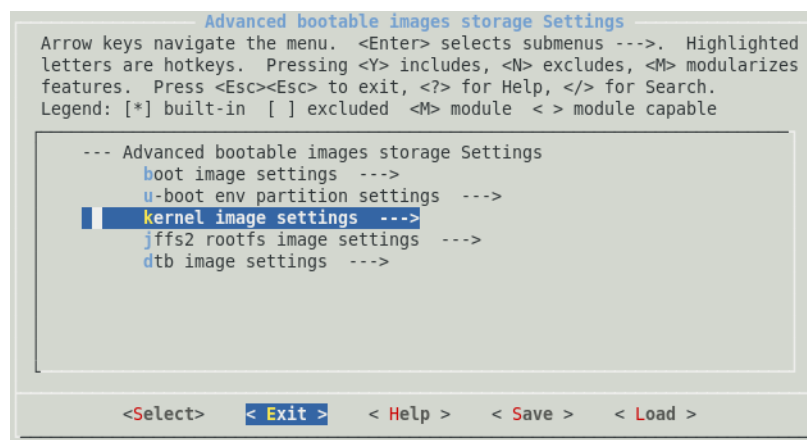
<Select> < Exit > < Help > < Save > < Load >

```

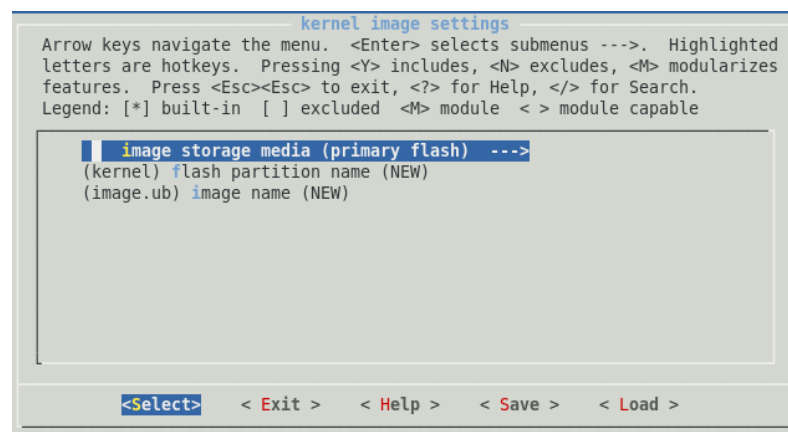
- e. Scroll down to *primary sd* and select it with the *Space* bar. This will automatically return to the previous screen, which will now show the *image storage media selection* as *primary sd*.



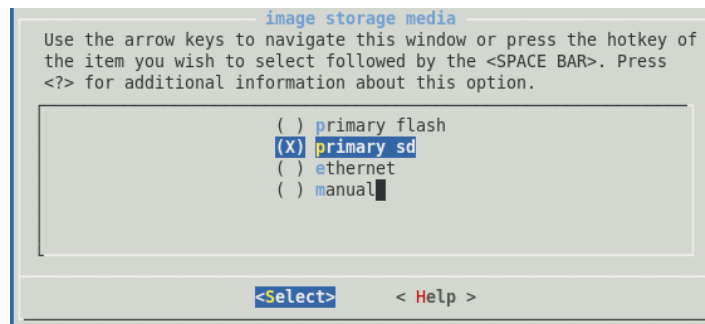
- f. Use the *Right Arrow* key to select **<EXIT>**. Hit the *Enter* key to move up one level in the configuration. Scroll down to *kernel image settings* and hit the *Enter* key.



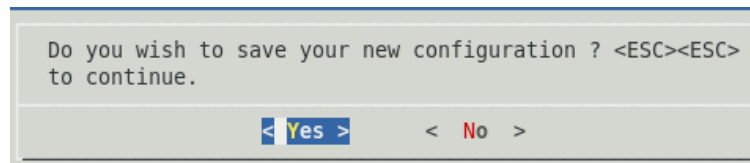
- g. Select *image storage media* and hit the *Enter* key.



- h. Scroll down to *primary sd* and select it with the *Space* bar. This will automatically return to the previous screen, which will now show the *image storage media selection* as **primary sd**.



- i. Select **<Exit>** (four times) until the option to save the new configuration appears. Hit the *Enter* key to accept **<Yes>** to save the file and exit the menuconfig application.



You will see that after exiting the configuration, a number of steps are performed by the Petalinux toolchain to fully customize the project:

1. Configuration file is saved
2. Device tree is generated
3. U-boot default configuration is created
4. First stage Zynq bootloader is created
5. Default configuration for kernel and root file system is written

```
[INFO ] config linux
configuration written to /home/training/zynq_mini-itx_7z045/subsystems/linux/config
```

```
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```

```
[INFO ] generate DTS to /home/training/zynq_mini-itx_7z045/subsystems/linux/configs/d
evice-tree
INFO: [Hsi 55-1698] elapsed time for repository loading 3 seconds
INFO: [Common 17-206] Exiting hsi at Mon Dec 21 17:51:15 2015...
[INFO ] generate linux/u-boot board header files
INFO: [Hsi 55-1698] elapsed time for repository loading 1 seconds
INFO: [Common 17-206] Exiting hsi at Mon Dec 21 17:51:19 2015...
[INFO ] generate /home/training/zynq_mini-itx_7z045/components/bootloader/zynq_fsbl
INFO: [Hsi 55-1698] elapsed time for repository loading 0 seconds
INFO: [Common 17-206] Exiting hsi at Mon Dec 21 17:51:33 2015...
[INFO ] generate BSP for zynq_fsbl
INFO: [Hsi 55-1698] elapsed time for repository loading 1 seconds
INFO: [Common 17-206] Exiting hsi at Mon Dec 21 17:51:46 2015...
INFO: Config linux/kernel
[INFO ] oldconfig linux/kernel
INFO: Config linux/rootfs
[INFO ] oldconfig linux/rootfs
```

## Configure the PetaLinux Kernel (optional)

The BSP customization is now complete, but depending on your requirements there may still be additional kernel configuration needed if your system differs from the defaults. In this example, there are no mandatory changes. However, if you wish to change the kernel configuration (advanced users only!), you may access the configuration menu using these steps.

1. Change to the PetaLinux project folder created earlier.

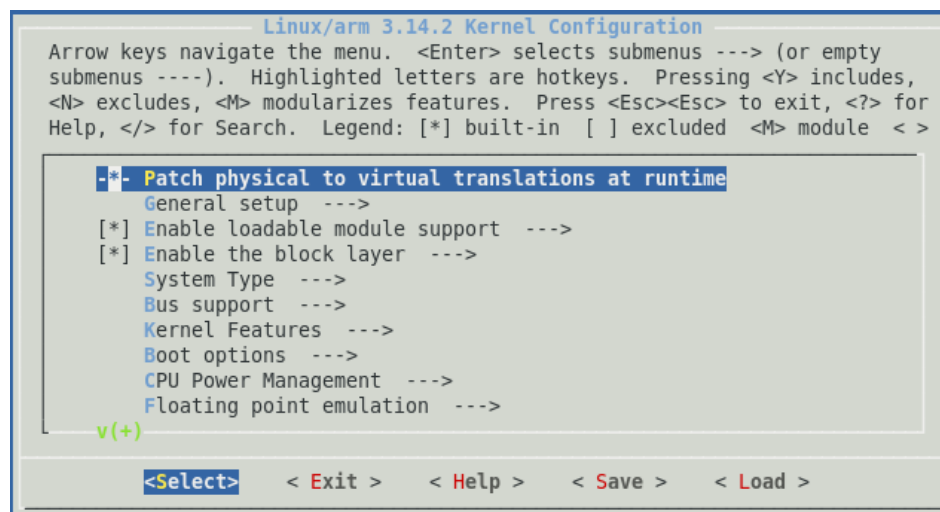
```
$ cd <Project Folder>
```

```
training@training-VirtualBox:~$ cd zynq_mini-itx_7z045/
```

2. Launch the menuconfig tool to customize the PetaLinux kernel.

```
$ petalinux-config -c kernel
```

```
training@training-VirtualBox:~/zynq_mini-itx_7z045$ petalinux-config -c kernel
INFO: Checking component...
INFO: Config linux/kernel
[INFO ] config linux/kernel
```



3. If you make kernel changes, save and exit in the usual manner (or discard any changes by selecting No when prompted to save). Exit the configuration application when you are done.

## Configure the PetaLinux Root File System

The gpio-demo is not included as a standard application in the default PetaLinux root file system configuration. This, and others, including custom applications you build yourself, can be added using the following steps.

1. Change to the PetaLinux project folder.

```
$ cd ~/<Project Folder>
```

```
training@training-VirtualBox:~/zynq_mini-itx_7z045$ cd ~/zynq_mini-itx_7z045/
```

2. Launch the menuconfig tool to customize the PetaLinux root file system.

```
$ petalinux-config -c rootfs
```

```
training@training-VirtualBox:~/zynq_mini-itx_7z045$ petalinux-config -c rootfs
INFO: Checking component...
INFO: Config linux/rootfs
[INFO ] config linux/rootfs
```

3. Scroll down to the *Apps* entry and hit the *Enter* key.

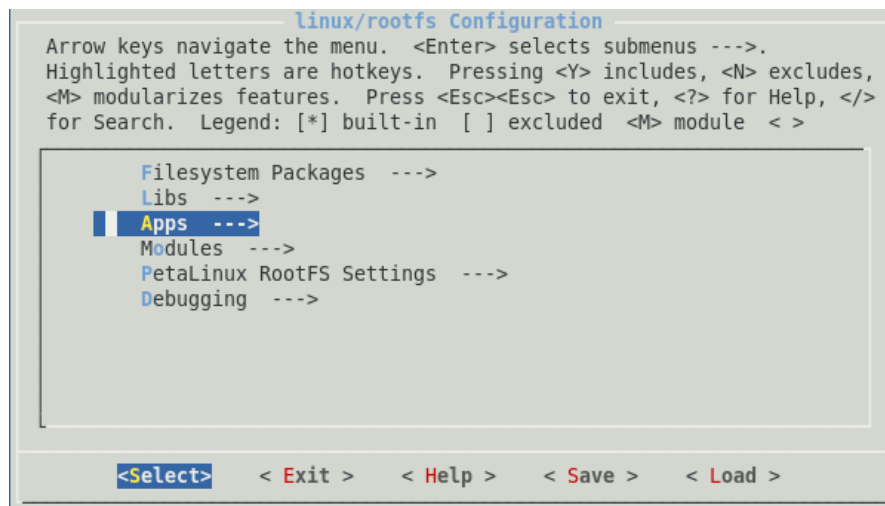


Figure 17 – Configure PetaLinux Applications



4. Scroll down to **gpio-demo** and use the *Space* bar to select it.

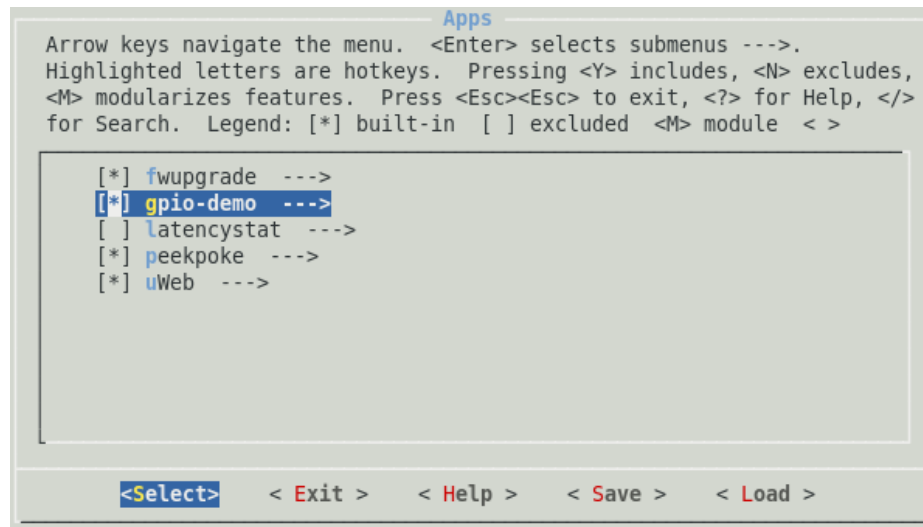


Figure 18 – Add the gpio-demo Application

5. Use the *Right Arrow* key to select **<Exit>** and hit the *Enter* key. Do this again at the top menu level. Hit the *Enter* key once more to save the new configuration file.

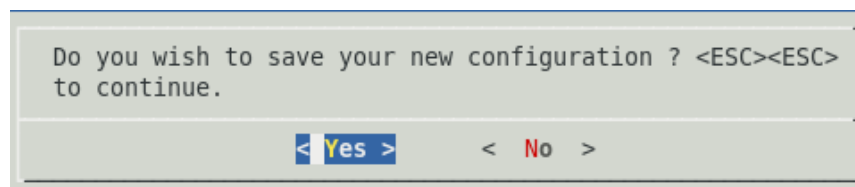


Figure 19 – Update PetaLinux Root File System Configuration

## Modify the Device Tree

There are a number of changes to be applied to the default device tree generated by the PetaLinux tools:

1. The *system.hdf* file only contains information for the ENET0 (eth0) MAC device, and for target boards with multiple PHYs there is no way to determine an association automatically. Since the networking subsystem in Linux must have specific information about the physical connections between MAC and PHY, an entry for the PHY must be added to the device tree source before the kernel is compiled.

```
&gem0 {
    phy-handle = <&phy0>;
    phy-mode = "rgmii-id";

    ps7_ethernet_0_mdio: mdio {
        #address-cells = <1>;
        #size-cells = <0>;
        phy0: phy@0 {
            compatible = "marvell,88e1510";
            device_type = "ethernet-phy";
            reg = <0x0>;
            marvell,reg-init = <3 16 0xff00 0x1e 3 17 0xffff0 0x00>;
        };
    };
};
```

2. Since the PetaLinux 2014.2 release, an association for QSPI must be specified to allow proper mapping of the Memory Technology Device (MTD) partitions. If the QSPI entry is omitted, no partitions will be visible.

```
&qspi {
    flash0: flash@0 {
        compatible = "micron,n25q128a13";
    };
};
```

3. The 2015.2 toolchain introduced a new driver stack for USB devices. There is an issue with USB-OTG not mapping to the correct driver entry that requires correction in the device tree file.

```
/{
    usb_phy0:usb_phy@0 {
        compatible="usb-nop-xceiv";
        #phy-cells = <0>;
    };
};

&usb0 {
    dr_mode = "otg";
    usb-phy = <&usb_phy0>;
};
```

4. The I2C chain includes a mux which is non-standard for the device driver stack, so it must be correctly mapped to allow connected devices to be accessed.

```
&i2c0 {
    #address-cells = <1>;
    #size-cells = <0>;
    i2cswitch@70 {
        compatible = "nxp,pca9548";
        #address-cells = <1>;
        #size-cells = <0>;
        reg = <0x70>;

        i2c@1 {
            #address-cells = <1>;
            #size-cells = <0>;
            reg = <1>;
            adv7511: adv7511 {
                compatible = "adi,adv7511";
                reg = <0x39>;
                adi,input-style = <0x02>;
                adi,input-id = <0x01>;
                adi,input-color-depth = <0x3>;
                adi,sync-pulse = <0x03>;
                adi,bit-justification = <0x01>;
                adi,up-conversion = <0x00>;
                adi,timing-generation-sequence = <0x00>;
                adi,vsync-polarity = <0x02>;
                adi,hsync-polarity = <0x02>;
                adi,tdms-clock-inversion;
                adi,clock-delay = <0x03>;
            };
        };

        fmc_i2c: i2c@3 {
            #size-cells = <0>;
            #address-cells = <1>;
            reg = <3>;
        };

        i2c_adau1761: i2c@5 {
            #size-cells = <0>;
            #address-cells = <1>;
            reg = <5>;

            adau1761: adau1761@3b {
                compatible = "adi,adau1761";
                reg = <0x3b>;
            };
        };
    };
};
```

The device tree source files are located under your PetaLinux project directory at:

**subsystems/linux/configs/device-tree**

You may use the **boot\_source/system-top.dts** file included in the **Supplied Files** to replace the existing file in your PetaLinux project.

## Build the PetaLinux System Image

Invoke the PetaLinux build command to compile the configured kernel.

1. In the Terminal window, verify you are in the PetaLinux project directory for your target.

```
$ pwd
```

```
training@training-VirtualBox:~/zynq_mini-itx_7z045$ pwd  
/home/training/zynq_mini-itx_7z045
```

2. Build the kernel. This step begins with the output shown below, and will remain static for 5-10 minutes or more, depending on the capabilities of your development system. The final completion output is shown in the subsequent Figure.

```
$ petalinux-build7
```

```
training@training-VirtualBox:~/zynq_mini-itx_7z045$ petalinux-build  
INFO: Checking component...  
INFO: Generating make files and build linux  
INFO: Generating make files for the subcomponents of linux  
INFO: Building linux  
[INFO ] pre-build linux/rootfs/fwupgrade  
[INFO ] pre-build linux/rootfs/peekpoke  
[INFO ] pre-build linux/rootfs/uWeb  
[INFO ] build system.dtb  
[INFO ] build linux/kernel
```

---

<sup>7</sup> If you encounter errors on the build after running it more than once, try first cleaning the installation by typing: **petalinux-build -x distclean**. For more detailed information on a failed build, open **build/build.log**.

```

training@training-VirtualBox: ~/zynq_mini-itx_7z045
[INFO ] install linux/kernel
[INFO ] update linux/u-boot source
[INFO ] generate linux/u-boot configuration files
[INFO ] build linux/u-boot
[INFO ] install linux/u-boot
[INFO ] Expanding rootfs
[INFO ] install sys_init
[INFO ] install linux/rootfs/fwupgrade
[INFO ] install linux/rootfs/peekpoke
[INFO ] install linux/rootfs/uWeb
[INFO ] install kernel in-tree modules
[INFO ] modules_install linux/kernel
[INFO ] post-install linux/rootfs/fwupgrade
[INFO ] post-install linux/rootfs/peekpoke
[INFO ] post-install linux/rootfs/uWeb
[INFO ] package rootfs.cpio to /home/training/zynq_mini-itx_7z045/images/linux
[INFO ] Update and install vmlinux image
[INFO ] vmlinux linux/kernel
[INFO ] install linux/kernel
[INFO ] package zImage
[INFO ] zImage linux/kernel
[INFO ] install linux/kernel
[INFO ] Package HDF bitstream
training@training-VirtualBox:~/zynq_mini-itx_7z045$ █

```

Figure 20 – PetaLinux Build Completion

The system image is created at:

`<Petalinux_project_dir>/images/linux/image.ub`

## Create the BOOT.BIN File

All the components have been assembled at this point, and the last step is to create a Zynq boot image. The PetaLinux packager will create a boot image for the Zynq 7000 AP SoC, combining the FSBL ELF, bitstream and u-boot ELF into a single BOOT.BIN file. This file is used in conjunction with the image.ub image created in the previous section to boot the target device.

1. In the Terminal window, verify you are in the PetaLinux project directory for your target.

```
$ pwd
```

```

[training@VBCentOS6 zynq_mini-itx_2014_4_7z045]$ pwd
/home/training/Zynq_Mini-ITX/PetaLinux/zynq_mini-itx_2014_4_7z045

```

2. Enter the package command to create the Zynq boot image.

```

$ petalinux-package \
--boot \
--fsbl images/linux/zynq_fsbl.elf \
--fpga images/linux/zynq_design_1_wrapper.bit \
--uboot

```

```

training@training-VirtualBox:~/zynq_mini-itx_7z045$ petalinux-package --boot --fsbl i
images/linux/zynq_fsbl.elf --fpga images/linux/zynq_design_1_wrapper.bit --uboot
INFO: Generating zynq binary package BOOT.BIN...
INFO: Binary is ready.

```

The BOOT.BIN image is created in the current directory and copied to the **images/linux** directory.

## Boot PetaLinux from the microSD Card

The image creation process is now complete. All that remains is to transfer the image files to the boot media and power up the board to see PetaLinux in action.

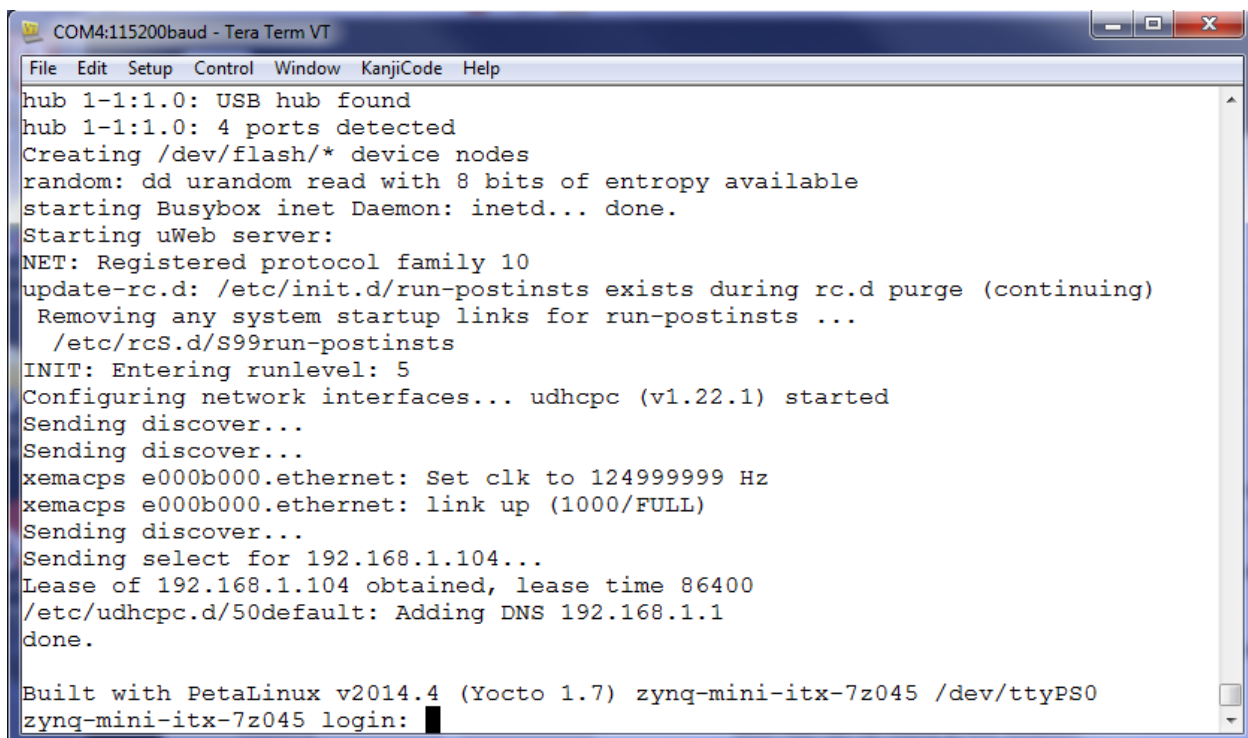
The PetaLinux kernel image, device tree blob and root file system are contained in a single file named **image.ub**. This file was created in the System Image build step.

The boot image (**BOOT.BIN**) created by the PetaLinux packager contains the Zynq FSBL, the target bitstream and u-boot executable.

Both files can be found at this location:

**<PetaLinux project directory>/images/linux/**

1. Copy the **BOOT.BIN** and the **image.ub** files to the top level directory of a FAT32 formatted microSD card.
2. Make the cable connections to your target board as described **Setting up the Zynq-7000 AP SoC Mini-ITX Development Kit**. Set the mode switches for SD boot (off, off, on, on, off) and power the board. Start your serial console to view the PetaLinux boot messages and login prompt.



```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
hub 1-1:1.0: USB hub found
hub 1-1:1.0: 4 ports detected
Creating /dev/flash/* device nodes
random: dd urandom read with 8 bits of entropy available
starting Busybox inet Daemon: inetd... done.
Starting uWeb server:
NET: Registered protocol family 10
update-rc.d: /etc/init.d/run-postinsts exists during rc.d purge (continuing)
  Removing any system startup links for run-postinsts ...
    /etc/rcS.d/S99run-postinsts
INIT: Entering runlevel: 5
Configuring network interfaces... udhcpc (v1.22.1) started
Sending discover...
Sending discover...
xemacps e000b000.ethernet: Set clk to 124999999 Hz
xemacps e000b000.ethernet: link up (1000/FULL)
Sending discover...
Sending select for 192.168.1.104...
Lease of 192.168.1.104 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.1.1
done.

Built with PetaLinux v2014.4 (Yocto 1.7) zynq-mini-itx-7z045 /dev/ttyPS0
zynq-mini-itx-7z045 login: █
```

Figure 21 – PetaLinux Boot Console

## Appendix I: Installation of USB UART Driver

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

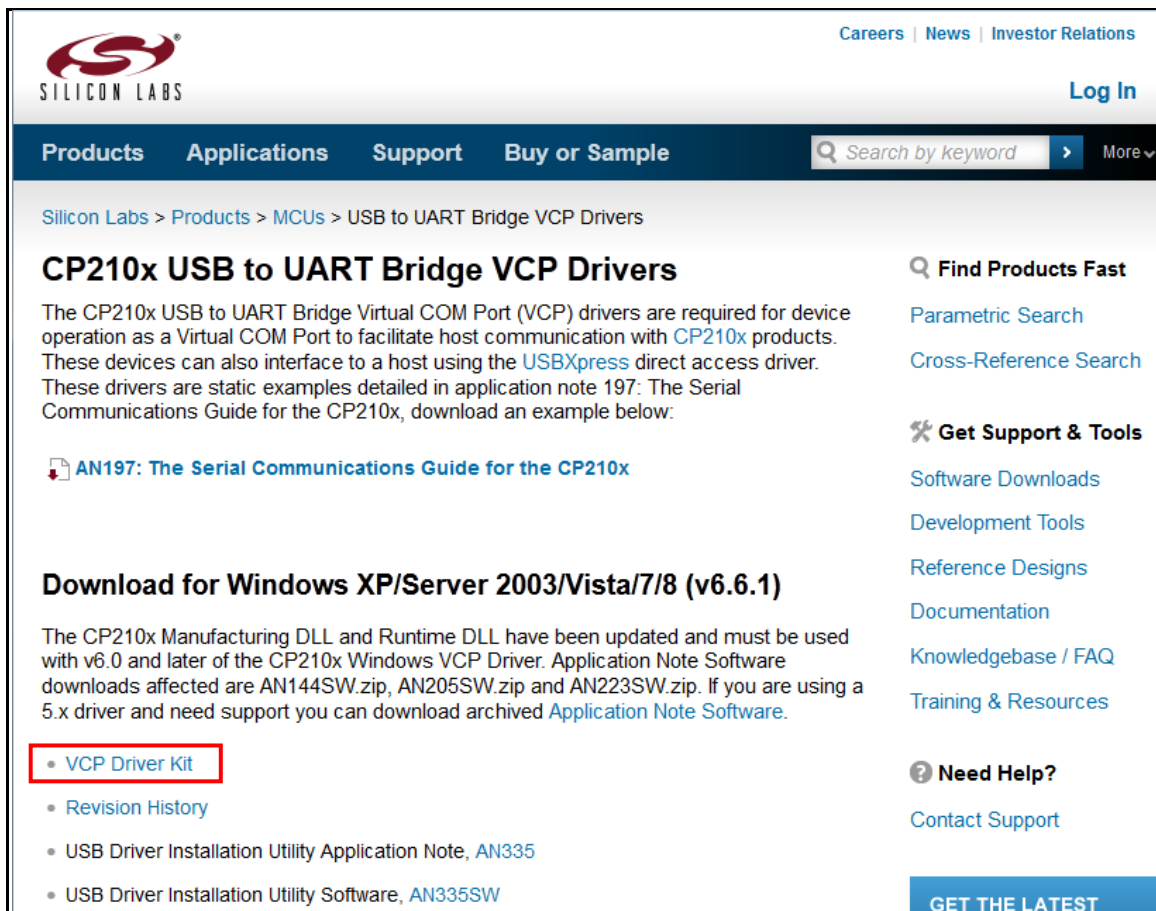
To install the Silicon Labs drivers follow the instructions listed below.

### Download and Install the Required Software (for Windows)

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

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

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



The screenshot shows the Silicon Labs website's product page for CP210x USB to UART Bridge VCP Drivers. The page layout includes a header with the Silicon Labs logo and navigation links (Careers, News, Investor Relations, Log In). Below the header is a navigation bar with links to Products, Applications, Support, and Buy or Sample, along with a search bar. The main content area features the product title 'CP210x USB to UART Bridge VCP Drivers' and a description of the drivers. A link to 'AN197: The Serial Communications Guide for the CP210x' is provided. A section titled 'Download for Windows XP/Server 2003/Vista/7/8 (v6.6.1)' contains a list of download links. The 'VCP Driver Kit' link is highlighted with a red box. A sidebar on the right contains links for 'Find Products Fast', 'Get Support & Tools', and 'Need Help?'. A 'GET THE LATEST' button is located at the bottom right of the page.

Figure 22 – USB to UART Download Site



- Once the file is downloaded, extract the **CP210x VCP Driver Kit** archive. For example, for Windows XP/Vista/7 the file is CP210x\_VCP\_Windows.zip. Once the archive is extracted, open the folder where the archive was extracted and choose the correct installer for a 32-bit (CP210xVCPInstaller\_x86.exe) or 64-bit (CP210xVCPInstaller\_x64.exe) PC. The installer will guide you through the setup. Accept the license agreement and install the software on your PC. Click **FINISH** when completed.

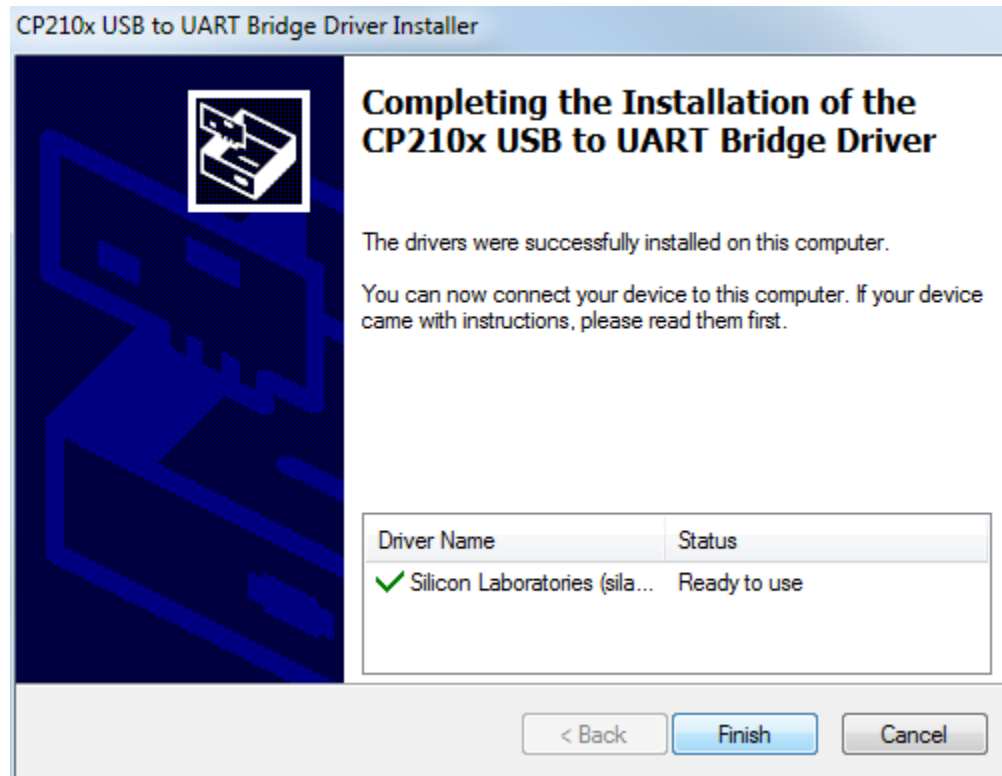


Figure 23 – USB to UART Driver Installed

## Determining the Virtual COM Port (on Windows)

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

1. Attach power to your Zynq Mini-ITX, and connect a USB A-micro B cable between the UART port of the board and an open USB port on your host computer. Set the board power switch to the ON position. You must see 8 green power LEDs illuminate. If you do not, check that the ATX two-piece power connector is firmly seated, and that the tabs on the smaller portion (4 wire) of the connector is beneath the larger part. Also make sure the power module is firmly seated on the board, and that all the connector pins are in good condition. Do not proceed until you see all 8 power LEDs illuminate.
2. In Windows Explorer, right-click on **Computer** and select **Manage** from the pop-up menu. Select **Device Manager** from the *Computer Management* window.

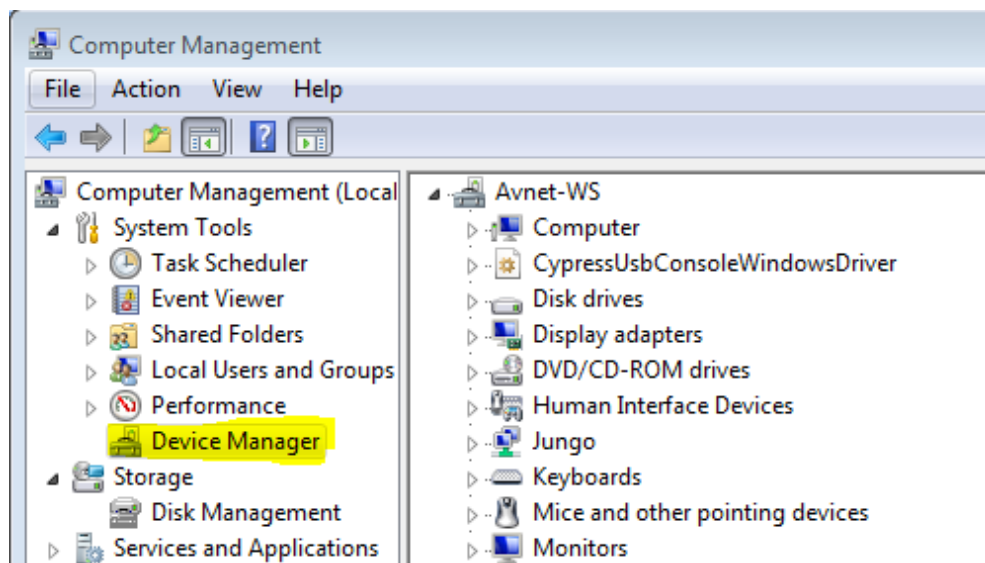


Figure 24 – Windows Device Manager

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

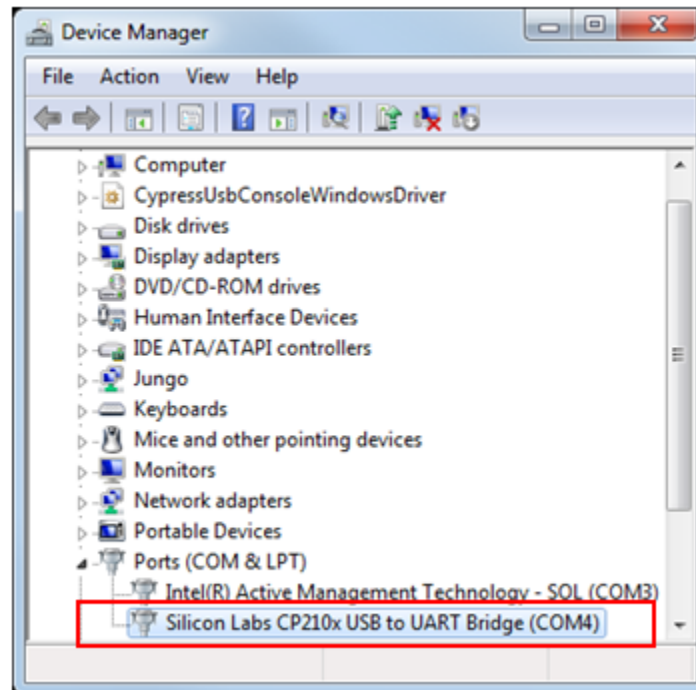


Figure 25 – UART Bridge COM Port

## Appendix II: References

### ZedBoard.org

**Embedded Standalone OS (Bare-Metal) Reference Design for the Zynq-7000 AP SoC Mini-ITX**

**Vivado 2015.2 (7z045 or 7z100)**

<http://zedboard.org/support/design/2056/17>

**VirtualBox and Linux VM Installation Guide**

<http://zedboard.org/support/design/2056/17>

### **Avnet Speedway – PetaLinux for the Zynq-7000 All Programmable SoC**

Explore additional PetaLinux capabilities and features using the Avnet ZedBoard or Avnet MicroZed. The same techniques for network booting, application development and debugging can be used for the Avnet Zynq-7000 AP SoC Mini-ITX, once you have the basic PetaLinux from this reference design operating.

<http://zedboard.org/support/trainings-and-videos>

### Xilinx Website

**Xilinx Vivado/SDK 2015.2 Installation Archive**

<http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools.html>

**PetaLinux 2015.2 Toolchain Installation Archive**

<http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools/2015-2.html>

**PetaLinux Tools Reference Guide (UG1144)**

[http://www.xilinx.com/support/documentation/sw\\_manuals/petalinux2015\\_2/ug1144-petalinux-tools-reference-guide.pdf](http://www.xilinx.com/support/documentation/sw_manuals/petalinux2015_2/ug1144-petalinux-tools-reference-guide.pdf)

**PetaLinux Tools Command Line Reference Guide (UG1157)**

[http://www.xilinx.com/support/documentation/sw\\_manuals/petalinux2015\\_2/ug1157-petalinux-tools-command-line-guide.pdf](http://www.xilinx.com/support/documentation/sw_manuals/petalinux2015_2/ug1157-petalinux-tools-command-line-guide.pdf)

**PetaLinux Tools Workflow Tutorial (UG1156)**

[http://www.xilinx.com/support/documentation/sw\\_manuals/petalinux2015\\_2/ug1156-petalinux-tools-workflow-tutorial.pdf](http://www.xilinx.com/support/documentation/sw_manuals/petalinux2015_2/ug1156-petalinux-tools-workflow-tutorial.pdf)

**PetaLinux Tools First Boot Checklist (UG1155)**

[http://www.xilinx.com/support/documentation/sw\\_manuals/petalinux2015\\_2/ug1155-petalinux-tools-boot-checklist.pdf](http://www.xilinx.com/support/documentation/sw_manuals/petalinux2015_2/ug1155-petalinux-tools-boot-checklist.pdf)

### Other Internet Resources

Configuring a TFTP Server for Linux: [https://linuxlink.timesys.com/docs/linux\\_tftp](https://linuxlink.timesys.com/docs/linux_tftp)

Tera Term Pro: <http://en.sourceforge.jp/projects/ttssh2>

CentOS ISO Image: <http://www.centos.org/download/>

Ubuntu ISO Image: <http://www.ubuntu.com/download/desktop>

## Appendix III: Useful Linux commands

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

All commands are case-sensitive.

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

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

## Revision History

Version	Date	Details
1.0	March 21, 2014	Vivado 2013.4, Xilinx SDK 2013.4, PetaLinux SDK 2013.10
1.1	April 3, 2014	Updated for Zynq Mini-ITX Rev C/D, split packages to separate 7z045 and 7z100
2.0	July 21, 2014	Vivado/SDK 2014.2, PetaLinux 2014.2
3.0	February 5, 2015	Vivado/SDK 2014.4, PetaLinux 2014.4
3.1	February 11, 2015	yum update issue resolved
4.0	December 21, 2015	Vivado 2015.2, PetaLinux 2015.2