

**Using USB for Secondary Boot Images  
With the  
Avnet Zynq<sup>®</sup>-7000 All Programmable SoC  
Mini-ITX Development Kit**

**ZYNQ<sup>®</sup>** 

**Version 1.0  
May 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
Booting PetaLinux from USB.....	7
Build and Boot PetaLinux for the Zynq Mini-ITX .....	7
Build PetaLinux for USB Access .....	9
Manually Boot PetaLinux from a USB Image.....	12
Configure, Build and Boot PetaLinux from USB.....	14
Load Programmable Logic from USB .....	16
Modify the Bitstream for PCAP Loading .....	16
Manually Load the Bitstream from U-boot .....	17
Configure PetaLinux U-boot for Automatic PL Load.....	19
Revision History .....	22

## Introduction

This application note describes the procedure for using a USB storage device as persistent memory for boot images not required by the First Stage Boot Loader. When a Zynq PS system boots from power-up, it executes start-up code in the internal BOOTROM. The mode settings are sampled to select a flash memory controller to initialize in order to retrieve the primary boot image. The only memory types that can be used to hold a primary boot image on Zynq are:

1. SD/microSD card
2. QSPI
3. NAND
4. NOR

There are no such restrictions placed on secondary boot images, such as the Programmable Logic bitstream or software components required for Linux. By placing these images in secondary storage, the size of the primary flash memory can be reduced, providing a corresponding cost reduction for the platform.

These instructions are based on 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. For convenience, a PetaLinux project can be created using the current version of the PetaLinux Board Support Package (BSP) published under the Zynq Mini-ITX Reference Designs at:

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

**PetaLinux Board Support Packages**

Compressed PetaLinux BSPs for Avnet Zynq system platforms.

**Zynq Mini-ITX 7Z045 PetaLinux Compressed BSP v2014.4** 

**Zynq Mini-ITX 7Z100 PetaLinux Compressed BSP v2014.4** 

For information on using BSPs with PetaLinux, please consult the following Xilinx User Guides:

1. UG1156 – PetaLinux Tools Workflow Tutorial
2. UG1144 – PetaLinux Tools Reference Guide

These are available for download from the Xilinx Support site:

<http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools/2014-4.html>

## Objectives

The goal is to begin with a PetaLinux boot image on a microSD card, and first move the Linux FIT image from the card to a USB device, where it will be accessed by U-boot. Second, we will remove the bitstream from the primary boot image and create a standalone image on a USB device that can be used by U-boot to load the Programmable Logic via the Processor Configuration Access Port (PCAP).

This creates a reduction in size for required primary boot memory for a Zynq Mini-ITX 7z045 from approximately 20 MB to less than 400 KB.

This goal is to begin with a PetaLinux project created from a Zynq Mini-ITX BSP on a Linux (or Linux VM) development system. You should be familiar with the operation of the PetaLinux tool chain and how to deploy images on an SD/microSD card and boot the target device to a Linux prompt.

## 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 2014.4
- TFTP Server (required by PetaLinux, but not used in the microSD boot)
- Silicon Labs CP2102 USB-UART bridge
- Serial terminal emulation software such as TeraTerm

### 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 this application note are delivered in a single compressed zip file:

### **Zynq\_Mini\_ITX\_USB\_Boot\_PetaLinux\_2014\_4.zip**

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.

**Zynq\_Mini-ITX\_USB\_Boot\_PetaLinux\_2014\_4.pdf:** this document.

#### **usb\_bit:**

**platform-top.h:** Modified u-boot top level configuration file used for automatically loading the programmable logic and booting Linux. The kernel and bitstream files are stored on a USB device.

#### **usb\_boot:**

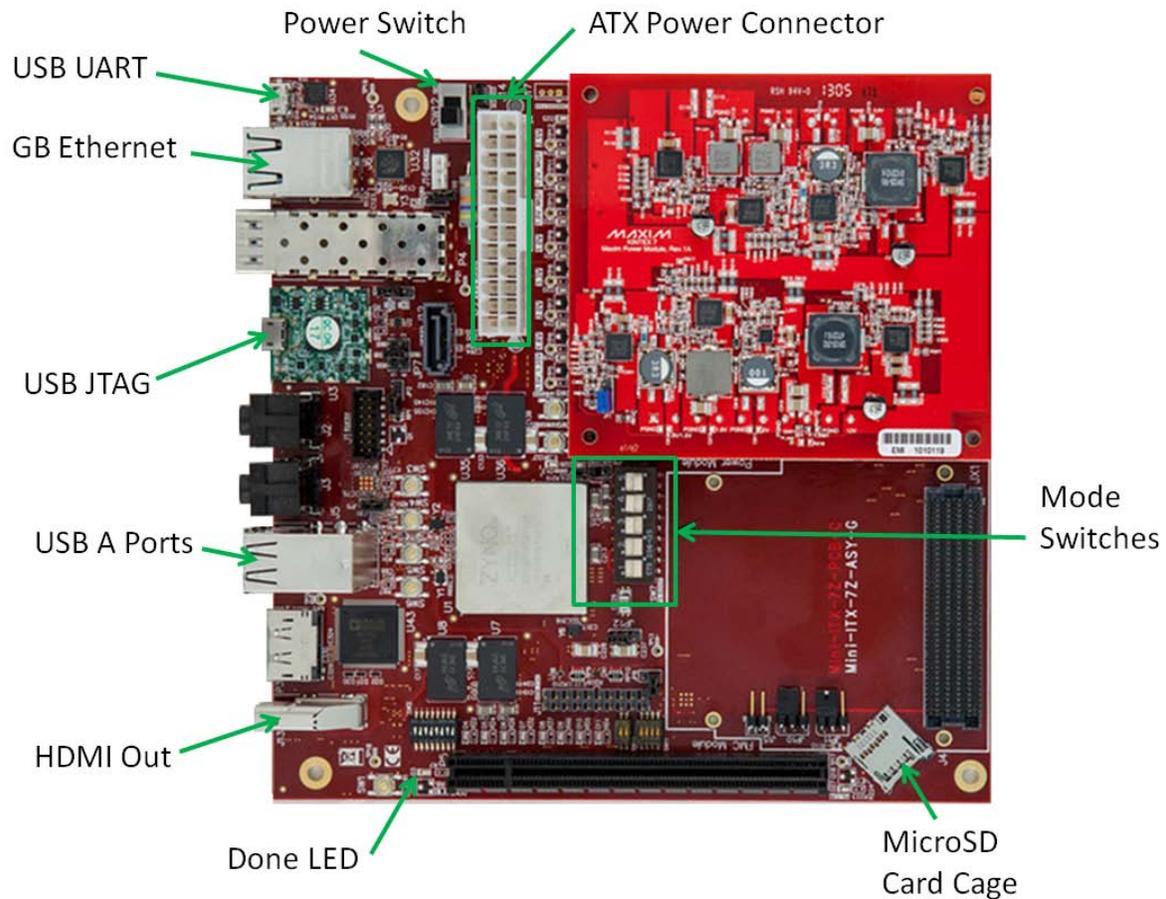
**platform-top.h:** Modified u-boot top level configuration file used for automatically booting Linux with the kernel stored on a USB device.

#### **usb\_config:**

**platform-top.h:** Modified u-boot top level configuration file used for including USB operations.

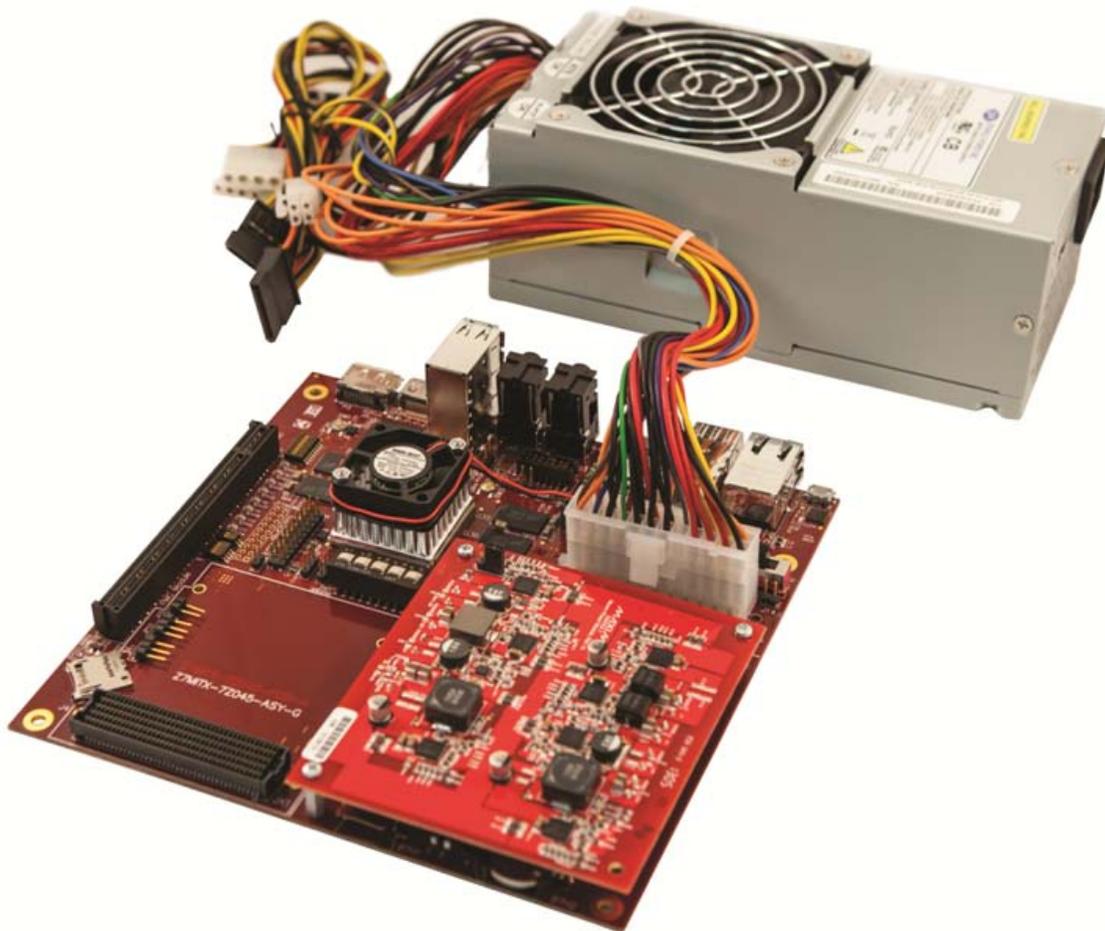
## Setting up the Zynq-7000 AP SoC Mini-ITX Development Kit

Refer to the following figure and perform the steps to set up the board connections.



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. Optionally 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. You may choose to connect the board to a LAN router or switch, which may be more convenient for providing a DHCP address to PetaLinux.

3. 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 1 – Zynq-7000 AP SoC Mini-ITX Development Board with ATX Power Supply**

4. 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.
5. Set the board power switch to the OFF position.

## Booting PetaLinux from USB

### Build and Boot PetaLinux for the Zynq Mini-ITX

The first goal is to start from a known good working platform. This section briefly describes the steps for using an existing BSP to boot the target from SD/microSD. If you have no prior experience working with PetaLinux BSPs and the PetaLinux tool chain, you may wish to consult more detailed instructions in the **PetaLinux Software Reference Design** on Zedboard.org.

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

#### High Level Procedure

1. Download the BSP for your target from Zedboard.org.
2. Create a PetaLinux project.
3. Copy the **BOOT.BIN** and **image.ub** files from the **pre-built** directory to a microSD card.
4. Set up the Zynq Mini-ITX with a USB serial connection to the development host (Ethernet is optional).
5. Set the mode switches for SD boot, power the board and start a serial console on the host.
6. Confirm that the system boots into Linux on your serial console.

#### Detailed Instructions

1. Download the BSP for your target from Zedboard.org.
  - a. Use your browser to access the URL above and download a compressed BSP to your PetaLinux development host.

##### PetaLinux Board Support Packages

Compressed PetaLinux BSPs for Avnet Zynq system platforms.

Zynq Mini-ITX 7Z045 PetaLinux Compressed BSP v2014.4



Zynq Mini-ITX 7Z100 PetaLinux Compressed BSP v2014.4



- b. Decompress the file on your development host.
 

```
unzip <downloaded file path>
```

```
[training@VBCent056 ~]$ cd ~/Desktop/
[training@VBCent056 Desktop]$ ls Zynq*
Zynq_Mini-ITX_z7045_PetaLinux_BSP_2014_4.zip
[training@VBCent056 Desktop]$ unzip Zynq_Mini-ITX_z7045_PetaLinux_BSP_2014_4.zip
Archive:  Zynq_Mini-ITX_z7045_PetaLinux_BSP_2014_4.zip
  inflating: zynq-mini-itx-7z045_v2014_4.bsp
```

2. Create a PetaLinux project.
  - a. Use the **petalinux-create** command to extract the BSP contents as a new PetaLinux project directory.

```

petalinux-create -t project -s <bsp path>
[training@VBCentOS6 Desktop]$ cd ~
[training@VBCentOS6 ~]$ petalinux-create -t project -s ~/Desktop/zynq-mini-itx-7z045_v2014_4.bsp
INFO: Create project:
INFO: Projects:
INFO: * zynq-mini-itx-7z045
INFO: has been successfully installed to /home/training/
INFO: New project successfully created in /home/training/

```

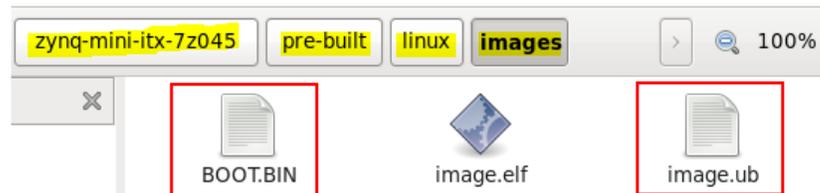
3. Copy the **BOOT.BIN** and **image.ub** files from the **pre-built** directory to a microSD card.
  - a. Navigate to the **pre-built** directory in your new PetaLinux project.

```

[training@VBCentOS6 ~]$ cd zynq-mini-itx-7z045/
[training@VBCentOS6 zynq-mini-itx-7z045]$ ls
BOOT.BIN  components  hardware  images  subsystems
build     config.project  hw-description  pre-built

```

- b. Inside your project, locate the directory **pre-built/linux/images** and copy the **BOOT.BIN** and **image.ub** files to the top level of a microSD card.



4. Set up the Zynq Mini-ITX with a USB serial connection to the development host (Ethernet is optional).
  - a. Follow the steps in **Setting up the Zynq-7000 AP SoC Mini-ITX Development Kit** to make the necessary cable connections for this application note.
5. Set the mode switches for SD boot and power the board.
  - a. 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. On position is where the switch is closest to the number on the block (towards the cooling fan).
  - b. Insert the microSD card containing your boot images into the card cage on the target.
  - c. Move the power switch to the ON position.

6. Confirm that the system boots into Linux on your serial console.
  - a. The system output shown is for a Zynq 7z045 Mini-ITX connected to a GB Ethernet router that includes a DHCP server.
  - b. You may log into the Linux system using the userid/password **root/root**.

```

COM4:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
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/udhcp.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: root
Password:
login[858]: root login on 'ttyPS0'
root@zynq-mini-itx-7z045:~#

```

## Build PetaLinux for USB Access

As of the 2014.4 release, the version of u-boot included in the PetaLinux tool chain does not include USB capability by default. We must edit the configuration files to add USB operations.

### High Level Procedure

1. Edit the **configs/u-boot/platform-top.h** file to add the USB configuration lines.
2. Build the PetaLinux kernel.
3. Create the **BOOT.BIN** as usual and update the microSD card.
4. Move the **image.ub** file from your microSD card to a USB device.

### Detailed Instructions

1. Edit the **configs/u-boot/platform-top.h** file to add the USB configuration lines. If you prefer, the **usb\_config/platform-top.h** file in the **Supplied Files** already has the changes required.
  - a. Within your PetaLinux project, navigate to the u-boot configuration directory. There are three files in this directory, but only **platform-top.h** file remains unchanged through subsequent configuration and build steps. To make persistent changes in the u-boot configuration we must edit only the **platform-top.h** file.

```

[training@VBCent056 zynq-mini-itx-7z045]$ cd subsystems/linux/configs/u-boot/
[training@VBCent056 u-boot]$ ls
config.mk platform-auto.h platform-top.h

```

- b. Open the **platform-top.h** file in your favorite editor. Add the lines shown below at the end of the file.

```
#define CONFIG_CMD_USB
#define CONFIG_USB_ULPI
#define CONFIG_USB_ULPI_VIEWPORT
#define CONFIG_USB_EHCI
#define CONFIG_USB_EHCI_ZYNQ
#define CONFIG_EHCI_IS_TDI
#define CONFIG_USB_MAX_CONTROLLER_COUNT 2

#define CONFIG_USB_STORAGE
#define CONFIG_DOS_PARTITION
#define CONFIG_SUPPORT_VFAT
#define CONFIG_CMD_FAT
#define CONFIG_FAT_WRITE
```

- c. Save the file and close the editor.

## 2. Build the PetaLinux kernel.

- a. Since this is a new project it is always a good idea to start fresh and rebuild all of the PetaLinux components, even though we have only changed u-boot. For subsequent changes we can rebuild only the affected component. Return to the top-level directory of the project and clean the generated files.

```
cd ../../../../..
```

```
petalinux-build -x distclean
```

```
[training@VBCent056 u-boot]$ cd ../../../../..
[training@VBCent056 zynq-mini-itx-7z045]$ pwd
/home/training/zynq-mini-itx-7z045
[training@VBCent056 u-boot]$ petalinux-build -x distclean
INFO: Checking component...
```

- b. Build all PetaLinux kernel components.

#### petalinux-build

```
[training@VBCentOS6 u-boot]$ 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/gpio-demo
[INFO ] pre-build linux/rootfs/peekpoke
[INFO ] pre-build linux/rootfs/uWeb
[INFO ] build system.dtb
[[INFO ] build linux/kernel
.
.
[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
```

3. Create the **BOOT.BIN** as usual and update the microSD card.
  - a. Create a new boot image that includes components from the pre-built directory for the first stage boot loader (FSBL)<sup>1</sup> and the bitstream, as well as the re-built Linux kernel and the second stage boot loader (SSBL) u-boot. The **force** option allows replacement of an existing **BOOT.BIN** file.

#### petalinux-package --boot --fpga <path> --uboot --force

```
[training@VBCentOS6 zynq-mini-itx-7z045]$ petalinux-package --boot --fpga pre-built
/linux/implementation/download.bit --uboot --force
INFO: Generating zynq binary package BOOT.BIN...
INFO: Binary is ready.
```

- b. Copy the **BOOT.BIN** file from the current directory to your microSD card, replacing the version used earlier to test-boot the target.
4. Move the **image.ub** file from your microSD card to a USB device.
  - a. Copy the **image.ub** file from the microSD card to your selected USB device using the standard tools on your development host.
  - b. Delete the **image.ub** file from your microSD card.
  - c. Insert the microSD card to the card cage on the target.

<sup>1</sup> If you specify the **--fsbl <path>** option, you may specify any Zynq FSBL ELF file you wish. If you omit the option, the tool chain will automatically include the **zynq\_fsbl.elf** from the **pre-built** hierarchy.

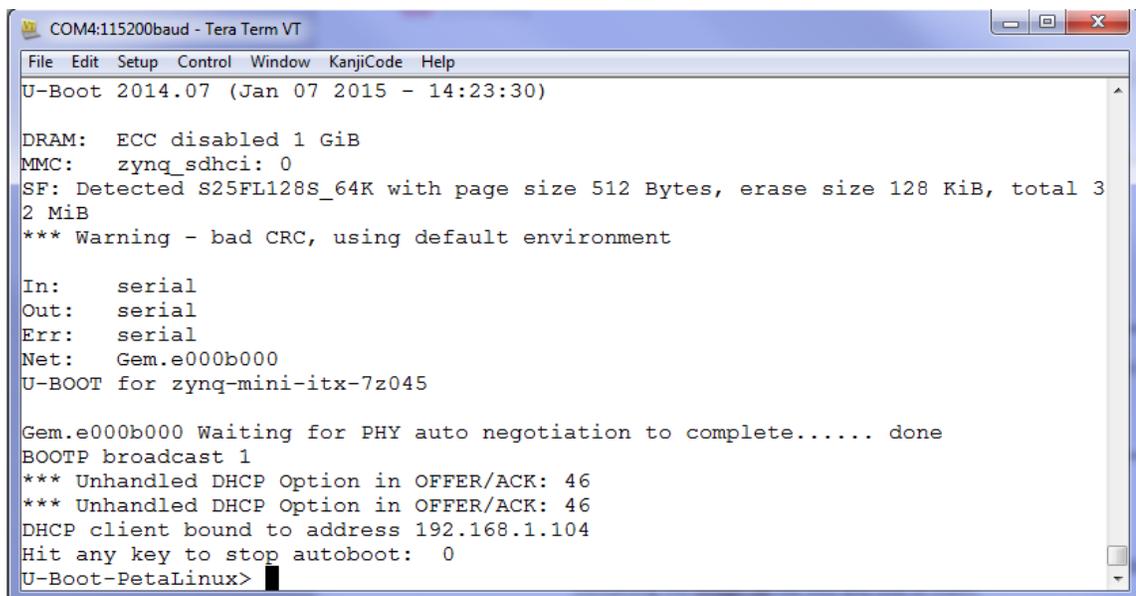
## Manually Boot PetaLinux from a USB Image

### High Level Procedure

1. Plug the USB device into one of the standard USB ports on the target.
2. Power the board. Hit any key in the terminal console during the boot process to interrupt u-boot during the countdown.
3. Test USB functionality.
4. Manually boot the system from u-boot using the image on USB.

### Detailed Instructions

1. Plug the USB device into one of the standard USB ports on the target.
  - a. The Zynq Mini-ITX has a tower of 4 USB-A ports. You may plug the USB storage device containing the **image.ub** file into any one of them.
2. Power the board. Hit any key in the terminal console during the boot process to interrupt u-boot during the countdown.
  - a. Immediately after powering the board, make the terminal console your primary window on the development host. When the u-boot countdown begins, the space bar on the keyboard is a convenient key to press to interrupt the boot process. The u-boot command prompt will be available in the console.



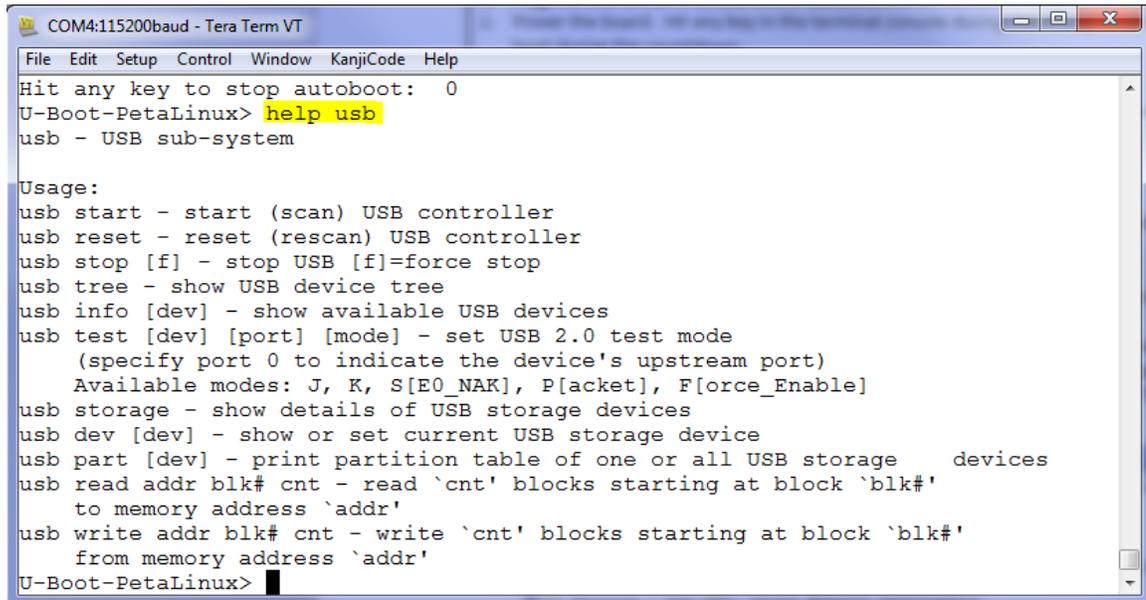
```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
U-Boot 2014.07 (Jan 07 2015 - 14:23:30)

DRAM:  ECC disabled 1 GiB
MMC:   zynq_sdhci: 0
SF: Detected S25FL128S_64K with page size 512 Bytes, erase size 128 KiB, total 3
2 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
U-BOOT for zynq-mini-itx-7z045

Gem.e000b000 Waiting for PHY auto negotiation to complete..... done
BOOTP broadcast 1
*** Unhandled DHCP Option in OFFER/ACK: 46
*** Unhandled DHCP Option in OFFER/ACK: 46
DHCP client bound to address 192.168.1.104
Hit any key to stop autoboot:  0
U-Boot-PetaLinux>
```

3. Test USB functionality.
  - a. Type **help usb** at the prompt to display the available USB commands in u-boot.



```

COM4:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help
Hit any key to stop autoboot: 0
U-Boot-PetaLinux> help usb
usb - USB sub-system

Usage:
usb start - start (scan) USB controller
usb reset - reset (rescan) USB controller
usb stop [f] - stop USB [f]=force stop
usb tree - show USB device tree
usb info [dev] - show available USB devices
usb test [dev] [port] [mode] - set USB 2.0 test mode
    (specify port 0 to indicate the device's upstream port)
    Available modes: J, K, S[E0_NAK], P[acket], F[orce_Enable]
usb storage - show details of USB storage devices
usb dev [dev] - show or set current USB storage device
usb part [dev] - print partition table of one or all USB storage devices
usb read addr blk# cnt - read `cnt' blocks starting at block `blk#'
    to memory address `addr'
usb write addr blk# cnt - write `cnt' blocks starting at block `blk#'
    from memory address `addr'
U-Boot-PetaLinux>
  
```

- b. Type **usb start** to scan the bus for USB devices.

```

U-Boot-PetaLinux> usb start
(Re)start USB...
USB0:  USB EHCI 1.00
scanning bus 0 for devices... 3 USB Device(s) found
USB1:  usb1 wrong num MIO: 0, Index 1
lowlevel init failed
scanning usb for storage devices... 1 Storage Device(s) found
  
```

USB 0: 2 Hubs, 1 Storage Device

USB 1: Nothing attached

4. Manually boot the system from u-boot using the image on USB.
  - a. The USB storage device was initialized in the previous step. Now we can use other USB commands to access and load the Linux image and boot the kernel to a command prompt. Type **fatls usb 0** to see the files present on the device.

```

U-Boot-PetaLinux> fatls usb 0
autorun/
36  autorun.inf
6835340  image.ub
  
```

- b. Load the Linux image from USB to DDR memory, taking care to place the image at an address well outside the executable space for the kernel.

**fatload usb 0 0x3000000 image.ub**

```

U-Boot-PetaLinux> fatload usb 0 0x3000000 image.ub
reading image.ub
6835340 bytes read in 579 ms (11.3 MiB/s)
  
```

- c. Execute the kernel loader using the boot-from-memory command. The kernel will boot to the command prompt.

**bootm 0x3000000**

```
U-Boot-PetaLinux> bootm 0x3000000
## Loading kernel from FIT Image at 03000000 ...
Using 'conf@1' configuration
Trying 'kernel@1' kernel subimage
  Description:  PetaLinux Kernel
  Type:         Kernel Image
  Compression:  gzip compressed
  Data Start:   0x030000f0
  Data Size:    6819167 Bytes = 6.5 MiB
  Architecture: ARM
  OS:           Linux
  Load Address: 0x00008000
  Entry Point:  0x00008000
  .
  .

Built with PetaLinux v2014.4 (Yocto 1.7) zynq-mini-itx-7z100 /dev/ttyPS0
zynq-mini-itx-7z100 login: █
```

## Configure, Build and Boot PetaLinux from USB

In the last section we confirmed that we can boot the Linux kernel from an attached USB device. In this section we will automate the process by modifying u-boot further.

### High Level Procedure

1. Modify the `platform-top.h` file in the u-boot configuration to automatically grab the FIT file from the USB device, instead of from the SD card.
2. Rebuild u-boot, create a new `BOOT.BIN` file and replace on the microSD.
3. Boot the target.

### Detailed Instructions

1. Modify the **platform-top.h** file in the u-boot configuration to automatically grab the FIT image from the USB device, instead of from the SD card.
  - a. Navigate to the u-boot configuration directory.
 

```
[training@VBCent0S6 zynq-mini-itx-7z045]$ cd subsystems/linux/configs/u-boot
```
  - b. Use your favorite editor to copy the `CONFIG_EXTRA_ENV_SETTINGS` definition from the **platform-auto.h** file to the end of the **platform-top.h** file. If you prefer, the **usb\_boot/platform-top.h** file in the **Supplied Files** already has the required changes.

- c. In the **platform-top.h** file, modify the `CONFIG_EXTRA_ENV_SETTINGS` block as follows:
  - i. Change the **cp\_kernel2ram** definition so that it initializes the usb subsystem and loads the kernel image from the usb device, instead of from the microSD card. The required command sequence is:

**usb start && fatload usb 0**

Change the highlighted portion of the `cp_kernel2ram` environment variable from:

```
"cp_kernel2ram=mmcinfo && fatload mmc 0 ${netstart} ${kernel_img}\0" \
```

to:

```
"cp_kernel2ram=usb start && fatload usb 0 ${netstart} ${kernel_img}\0" \
```

- d. Save the **platform-top.h** file and close the editor.
2. Rebuild u-boot, create a new **BOOT.BIN** file and replace on the microSD.
    - a. Since we have previously built the PetaLinux project, we can save time by only rebuilding the changed component u-boot. Change back to the top level directory of the project and enter:

**petalinux-build -c u-boot**

```
[training@VBCentOS6 u-boot]$ cd ../../../../..
[training@VBCentOS6 zynq-mini-itx-7z045]$ petalinux-build -c u-boot
INFO: Checking component...
INFO: Generating make files and build linux/u-boot
INFO: Generating make files for the subcomponents of linux/u-boot
INFO: Building linux/u-boot
[INFO ] update linux/u-boot source
[INFO ] generate linux/u-boot configuration files
[INFO ] build linux/u-boot
[INFO ] update linux/u-boot source
[INFO ] generate linux/u-boot configuration files
[INFO ] build linux/u-boot
[INFO ] install linux/u-boot
```

- b. Create a new boot image containing the FSBL, bitstream and modified u-boot.

**petalinux-package --boot --fpga <path> --uboot --force**

```
[training@VBCentOS6 zynq-mini-itx-7z045]$ petalinux-package --boot --fpga pre-built
/linux/implementation/download.bit --uboot --force
INFO: Generating zynq binary package BOOT.BIN...
INFO: Binary is ready.
```

3. Boot the target.
  - a. Copy the new **BOOT.BIN** file from the top level of your PetaLinux project to the microSD card, replacing the existing file of the same name.
  - b. Replace the microSD card in the target card cage.
  - c. Power the target. The system will automatically boot to the Linux command prompt, loading the kernel from the attached USB device.

## Load Programmable Logic from USB

Now that we have the Zynq system booting into Linux from a secondary image on a USB storage device, we can further reduce the required primary boot storage requirement by removing the bitstream from the BOOT.BIN file. The bitstream is usually the largest component by far in the primary boot image, and by placing it in secondary storage and configuring u-boot to load it to the programmable logic, we can reduce the BOOT.BIN size to 400KB or less.

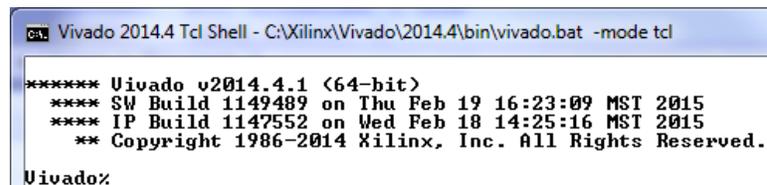
## Modify the Bitstream for PCAP Loading

### High Level Procedure

1. Use the Vivado TCL Shell to byte-swap the bitstream.
2. Copy the byte-swapped file to the USB device.
3. Create a new **BOOT.BIN** image with only the FSBL and u-boot.
4. Replace the **BOOT.BIN** on the microSD card.

### Detailed Instructions

1. Use the Vivado TCL Shell to byte-swap the bitstream.
  - a. When loading a bitstream via the PCAP, there is a requirement to byte-swap the original image. For this task only you will need access to a development machine that has Vivado installed. On a Vivado development machine, open a Vivado TCL Shell.



```

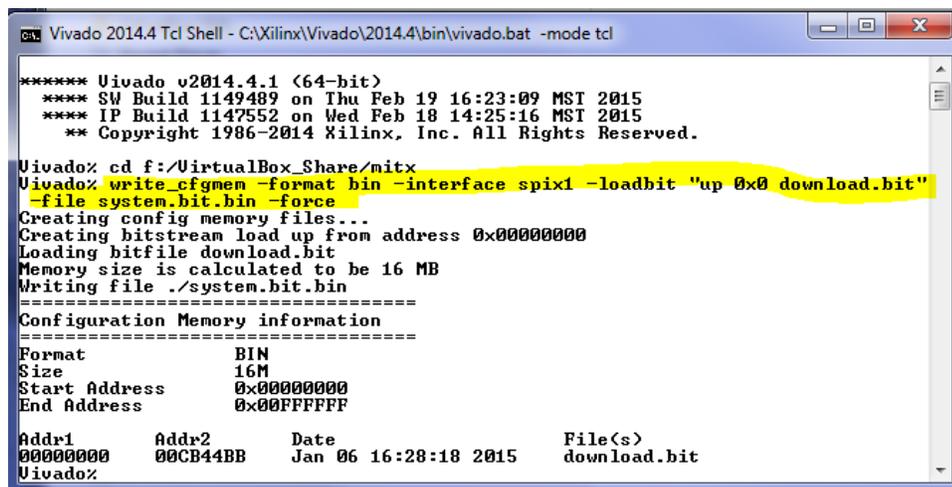
C:\Xilinx\Vivado\2014.4\bin\vivado.bat -mode tcl

***** Uivado v2014.4.1 (64-bit)
***** SW Build 1149489 on Thu Feb 19 16:23:09 MST 2015
***** IP Build 1147552 on Wed Feb 18 14:25:16 MST 2015
***** Copyright 1986-2014 Xilinx, Inc. All Rights Reserved.

Uivado%
  
```

- b. Use the `write_cfgmem` TCL command to perform the byte-swap. Change to the directory where your bitstream (`download.bit` in this example) is located. At the command prompt, enter:

```
write_cfgmem -format bin -interface spix1 -loadbit "up 0x0 <bitstream>" -file <result file> -force
```



```

C:\Xilinx\Vivado\2014.4\bin\vivado.bat -mode tcl

***** Uivado v2014.4.1 (64-bit)
***** SW Build 1149489 on Thu Feb 19 16:23:09 MST 2015
***** IP Build 1147552 on Wed Feb 18 14:25:16 MST 2015
***** Copyright 1986-2014 Xilinx, Inc. All Rights Reserved.

Uivado% cd f:/VirtualBox_Share/mitx
Uivado% write_cfgmem -format bin -interface spix1 -loadbit "up 0x0 download.bit"
         -file system.bit.bin -force
Creating config memory files...
Creating bitstream load up from address 0x00000000
Loading bitfile download.bit
Memory size is calculated to be 16 MB
Writing file ./system.bit.bin
=====
Configuration Memory information
=====
Format          BIN
Size            16M
Start Address   0x00000000
End Address     0x00FFFFFF

Addr1          Addr2          Date           File(s)
00000000      00CB44BB      Jan 06 16:28:18 2015  download.bit
Uivado%
  
```

2. Copy the result file to the USB device.
  - a. The name of the result file is significant in that it is used in the u-boot configuration in the following steps. For the purposes of this example, the result file name chosen is **system.bit.bin**.
3. Create a new **BOOT.BIN** image with only the FSBL and u-boot.
  - a. Use the `petalinux-package` command from the top-level directory of your PetaLinux project to create the new boot image. Simply omit the `fpga` option to exclude the bitstream. As before, the `zynq_fsbl.elf` file is automatically inserted from the pre-built directory.

```
petalinux-package --boot --uboot --force
```

```
[training@VBCent056 zynq-mini-itx-7z045]$ petalinux-package --boot --uboot --force  
INFO: Generating zynq binary package BOOT.BIN...  
INFO: Binary is ready.
```

4. Replace the **BOOT.BIN** on the microSD card.
  - a. Note that the size of the boot image is < 400 KB.

Name:	BOOT.BIN
Type:	unknown (application/octet-stream)
Size:	<b>399.1 KB</b> (408680 bytes)

## Manually Load the Bitstream from U-boot

### High Level Procedure

1. Boot the target, interrupt u-boot to get a command prompt.
2. Manually type the commands to program the FPGA using the USB image.
3. Complete the USB boot to the PetaLinux prompt.

### Detailed Instructions

1. Boot the target, interrupt u-boot to get a command prompt.
  - a. Plug your USB device containing the byte-swapped bitstream and Linux image into one of the standard USB-A ports on the target.
  - b. Power the board. Hit any key in the terminal console during the boot process to interrupt u-boot during the countdown.

2. Manually type the commands to program the FPGA using the USB image.
  - a. Start the usb interface.

**usb start**

```
U-Boot-PetaLinux> usb start
(Re)start USB...
USB0:   USB EHCI 1.00
scanning bus 0 for devices... 3 USB Device(s) found
USB1:   usb1 wrong num MIO: 0, Index 1
lowlevel init failed
        scanning usb for storage devices... 1 Storage Device(s) found
```

- b. Copy the byte-swapped bitstream from USB to memory.

**fatload usb 0 <DDR address> <byte-swapped file>**

```
U-Boot-PetaLinux> fatload usb 0 0x100000 system.bit.bin
reading system.bit.bin
13321404 bytes read in 787 ms (16.1 MiB/s)
```

- c. Load the bitstream.

**fpga load 0 <DDR address> \${filesize}**

```
U-Boot-PetaLinux> fpga load 0 0x100000 ${filesize}
```

The blue DONE LED will illuminate to confirm that the bitstream has been successfully programmed. You may see the message:

```
zynq_align_dma_buffer: Bitstream is not swapped(1) - swap it
```

This is an artifact of the tools. As long as the DONE LED is illuminated, you may ignore this warning.

3. Complete the USB boot to the PetaLinux prompt.
  - a. Previously, we have changed the default environment variables in u-boot to load the Linux image from USB to DDR (*cp\_kernel2ram*). So all that remains is to execute this command and boot Linux from memory. These commands are automatically combined in u-boot environment variable **default\_bootcmd**.

**run default\_bootcmd**

```
U-Boot-PetaLinux> run default_bootcmd
reading image.ub
6835340 bytes read in 551 ms (11.8 MiB/s)
## Loading kernel from FIT Image at 01000000 ...
   Using 'conf@1' configuration
   Trying 'kernel@1' kernel subimage
     Description:  PetaLinux Kernel
   .
   .
Lease of 192.168.1.104 obtained, lease time 86400
/etc/udhcp.d/50default: Adding DNS 192.168.1.1
done.

Built with PetaLinux v2014.4 (Yocto 1.7) zynq-mini-itx-7z100 /dev/ttyPS0
zynq-mini-itx-7z100 login: █
```

## Configure PetaLinux U-boot for Automatic PL Load

Now that we have manually loaded the bitstream from USB, and automatically booted the Linux kernel from USB, the final step is to modify the u-boot environment variables to insert the commands from the previous section.

### High Level Procedure

1. Update the **platform-top.h** file with the necessary variable definitions.
2. Build u-boot.
3. Create a new **BOOT.BIN**.
4. Replace the **BOOT.BIN** on the microSD card.
5. Boot the system.

### Detailed Instructions

1. Update the **platform-top.h** file with the necessary variable definitions.

- a. Navigate to the u-boot configuration directory.

```
[training@VBCentOS6 zynq-mini-itx-7z045]$ cd subsystems/linux/configs/u-boot
```

- b. Use your favorite editor to open the **platform-top.h** file. If you prefer, the **usb\_bit/platform-top.h** file in the **Supplied Files** already has the required changes.

- c. In the **platform-top.h** file, modify the **CONFIG\_EXTRA\_ENV\_SETTINGS** block as follows:

- i. Immediately below the **PSSERIAL0** line, add:

```
"loadbit_addr=0x100000\0" \
```

```
"bitstream_image=system.bit.bin\0" \
```

```
"usb_loadbit_fat=echo Loading bitstream from USB device 0...; && usb start && Fatload usb 0 ${loadbit_addr} ${bitstream_image} && fpga load 0 ${loadbit_addr} ${filesize}\0" \
```

```
#define CONFIG_EXTRA_ENV_SETTINGS \
```

```
SERIAL_MULTI \
```

```
CONSOLE_ARG \
```

```
PSSERIAL0 \
```

```
"loadbit_addr=0x100000\0" \
```

```
"bitstream_image=system.bit.bin\0" \
```

```
"usb_loadbit_fat=echo Loading bitstream from USB device 0...; && usb start && fatload usb 0 ${loadbit_addr} ${bitstream_image} && fpga load 0 ${loadbit_addr} ${filesize}\0" \
```

```
"nc=setenv stdout nc;setenv stdin nc;\0" \
```

- ii. Change the **cp\_kernel2ram** definition so that it executes the bitstream load instead of starting the USB interface. Change the highlighted portion of the **cp\_kernel2ram** environment variable from:

```
"cp_kernel2ram=usb start 0 && fatload usb 0 ${netstart} ${kernel_img}\0" \
```

to:

```
"cp_kernel2ram=run usb loadbit_fat && fatload usb 0 ${netstart} ${kernel_img}\0" \
```

- d. Save the **platform-top.h** file and close the editor.
2. Build u-boot.
    - a. Change back to the top level directory of the project and enter:

**petalinux-build -c u-boot**

```
[training@VBCent056 u-boot]$ cd ../../../../..
[training@VBCent056 zynq-mini-itx-7z045]$ petalinux-build -c u-boot
INFO: Checking component...
INFO: Generating make files and build linux/u-boot
INFO: Generating make files for the subcomponents of linux/u-boot
INFO: Building linux/u-boot
[INFO ] update linux/u-boot source
[INFO ] generate linux/u-boot configuration files
[INFO ] build linux/u-boot
[INFO ] update linux/u-boot source
[INFO ] generate linux/u-boot configuration files
[INFO ] build linux/u-boot
[INFO ] install linux/u-boot
```

3. Create a new **BOOT.BIN**.
  - a. Use the petalinux-package command to update the boot image.

**petalinux-package --boot --uboot --force**

```
[training@VBCent056 zynq-mini-itx-7z045]$ petalinux-package --boot --uboot --force
INFO: Generating zynq binary package BOOT.BIN...
INFO: Binary is ready.
```

4. Replace the **BOOT.BIN** on the microSD card.

5. Boot the system.
  - a. Replace the microSD card in the target card cage.
  - b. Power the target. The system will automatically boot to the Linux command prompt, loading the bitstream and kernel from the attached USB device.

```

U-Boot 2014.07 (May 06 2015 - 16:29:28)

DRAM:  ECC disabled 1 GiB
MMC:  zynq_sdhci: 0
SF: Detected S25FL128S_64K with page size 512 Bytes, erase size 128 KiB, total 3
2 MiB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   Gem.e000b000
U-BOOT for zynq-mini-itx-7z045

Gem.e000b000 Waiting for PHY auto negotiation to complete..... done
BOOTP broadcast 1
*** Unhandled DHCP Option in OFFER/ACK: 46
*** Unhandled DHCP Option in OFFER/ACK: 46
DHCP client bound to address 192.168.1.104
Hit any key to stop autoboot:  0
Loading bitstream from USB device 0...
(Re)start USB...
USB0:  USB EHCI 1.00
scanning bus 0 for devices... 3 USB Device(s) found
USB1:  usb1 wrong num MIO: 0, Index 1
lowlevel init failed
       scanning usb for storage devices... 1 Storage Device(s) found
reading system.bit.bin
13321404 bytes read in 437 ms (29.1 MiB/s)
zynq_align_dma_buffer: Bitstream is not swapped(1) - swap it
reading image.ub
6835340 bytes read in 229 ms (28.5 MiB/s)
## Loading kernel from FIT Image at 01000000 ...
   Using 'conf@1' configuration
   Trying 'kernel@1' kernel subimage
     Description:  PetaLinux Kernel
     Type:         Kernel Image
     Compression: gzip compressed
     Data Start:   0x010000f0
     Data Size:    6819167 Bytes = 6.5 MiB
     Architecture: ARM
   .
   .
Lease of 192.168.1.104 obtained, lease time 86400
/etc/udhcp.d/50default: Adding DNS 192.168.1.1
done.

Built with PetaLinux v2014.4 (Yocto 1.7) zynq-mini-itx-7z100 /dev/ttyPS0
zynq-mini-itx-7z100 login: █

```

This completes the USB boot Application Note.

## Revision History

Version	Date	Details
1.0	May 7, 2015	PetaLinux 2014.4