USB Design Examples

This document describes how to build and load a Linux kernel onto our development board using a cross complier and SD card. The instructions show the zc702 board, but they can be used for other boards, too.

There are two examples:

- Mass Storage device (peripheral)
- CDC Ethernet RNDIS adapter (peripheral)

The mass storage device example makes the Zynq board appear as a small 1 MB flash memory device when connected to a Host system. The Ethernet RNDIS example creates an adapter to allow another system (Host PC) to access the Linux operating system. The functionality on the Zynq board depends on what features are built into the kernel. In the Ethernet example, netperf is supported by the kernel.

Other Help

The design steps assume that the user is familiar with building Linux kernels, creating loadable modules and operating our development boards. The user can reference these links for helpful information:

- Xilinx Linux Wiki – build kernel
- zc702 Evaluation Kit with Targeted Reference Designs
- Standalone USB Design Examples – BIST with Mass Storage and Ethernet RNDIS

Design Steps

Both examples follow the same steps with some differences depending on the example. The differences are identified in the steps where differences exist. Here are the design example steps:

- Check-out the Linux Source Tree
- Configure the Kernel
- Build the Kernel Image
- Build the Device Tree
- Copy file to the SD Card
- Execute Commands at the Shell Prompt and Exercise the Device

1. Check-out the Linux Source Tree

Set the CROSS_COMPILE environment variable and add it to your PATH:

```
export CROSS_COMPILE=arm-xilinx-linux-gnueabi-
export PATH=<path to cross compiler bin>:$PATH
```

Linux kernel compilation internally uses **mkimage** command for creating uImage (Linux Kernel Image).

Hence, the path for the **mkimage** command must be added in PATH environment variable as shown below.

One can use the **mkimage** command that is built during the U-Boot building process:

```
export PATH=<path to mkimage>:$PATH
```

Clone the latest Zynq Linux kernel git repository from the Xilinx git server.

```
git clone git://github.com/Xilinx/linux-xlnx.git
```

2. Configure the Kernel

Configure the Linux kernel for the Zynq ZC702:

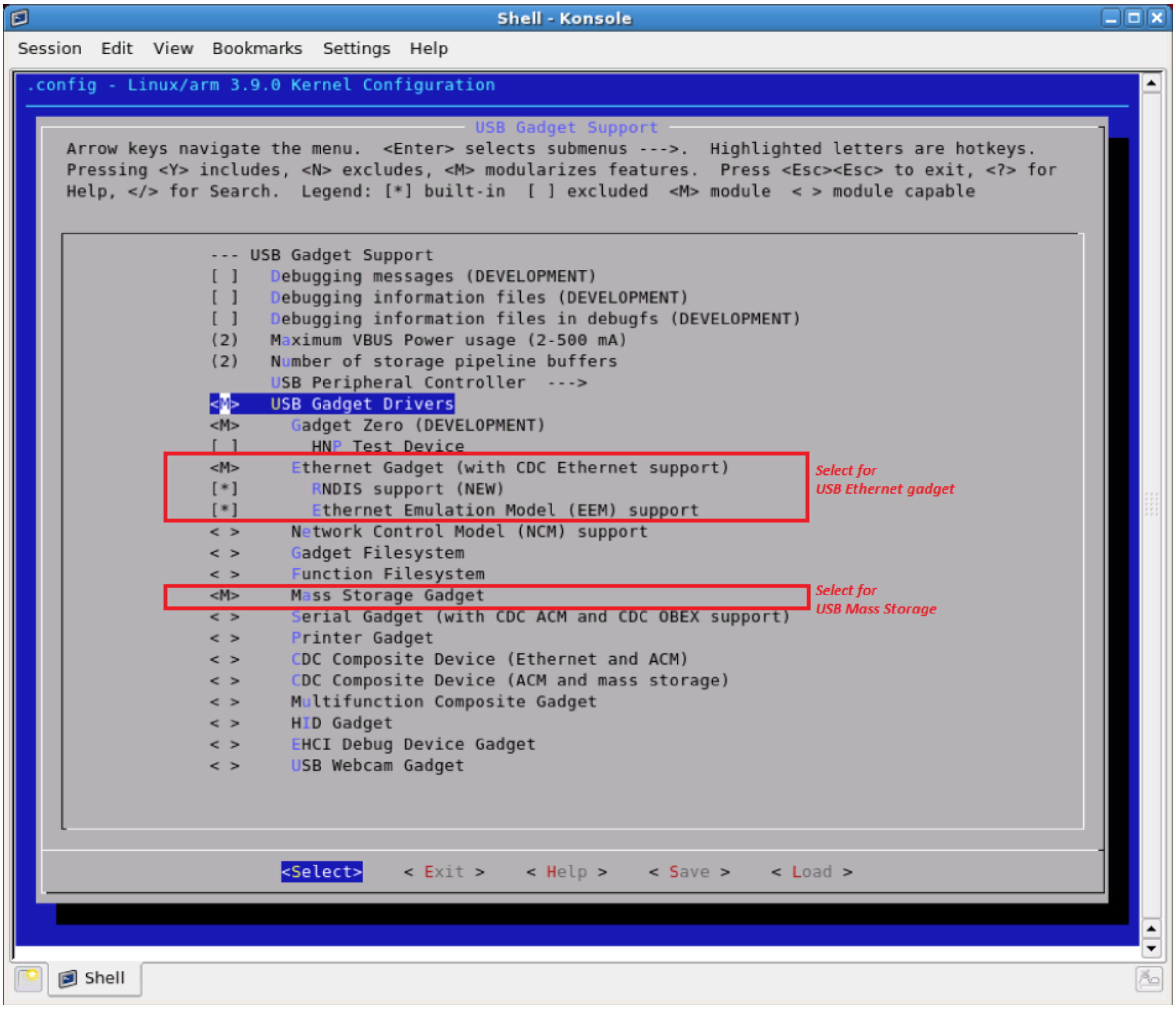
```
make ARCH=arm xilinx_zynq_base_trd_defconfig
```

Start the Kernel configuration tool:

```
make ARCH=arm menuconfig
```

Select either the Mass Storage or Ethernet Gadget Example from the configuration window.

- <M> means the driver is loadable.
- <*> means the driver is integrated into the kernel; monolithic.



Use the kernel configuration tool to select the appropriate drivers.

Mass Storage Kernel Config

- Device Drivers ➔ USB Support ➔ USB Gadget Support ➔ USB Gadget Drivers = <M>
- Device Drivers ➔ USB Support ➔ USB Gadget Support ➔ Mass Storage Gadget = <M>

Ethernet RNDIS Kernel Config

- Device Drivers ➔ USB Support ➔ USB Gadget Support ➔ Ethernet Gadget (with CDC Ethernet support) = <M>
- Device Drivers ➔ USB Support ➔ USB Gadget Support ➔ Ethernet Gadget (with CDC Ethernet support) ➔ RNDIS Support = <*>
- Device Drivers ➔ USB Support ➔ USB Gadget Support ➔ Ethernet Gadget (with CDC Ethernet support) ➔ Ethernet Emulation Model (EEM) support = <*>

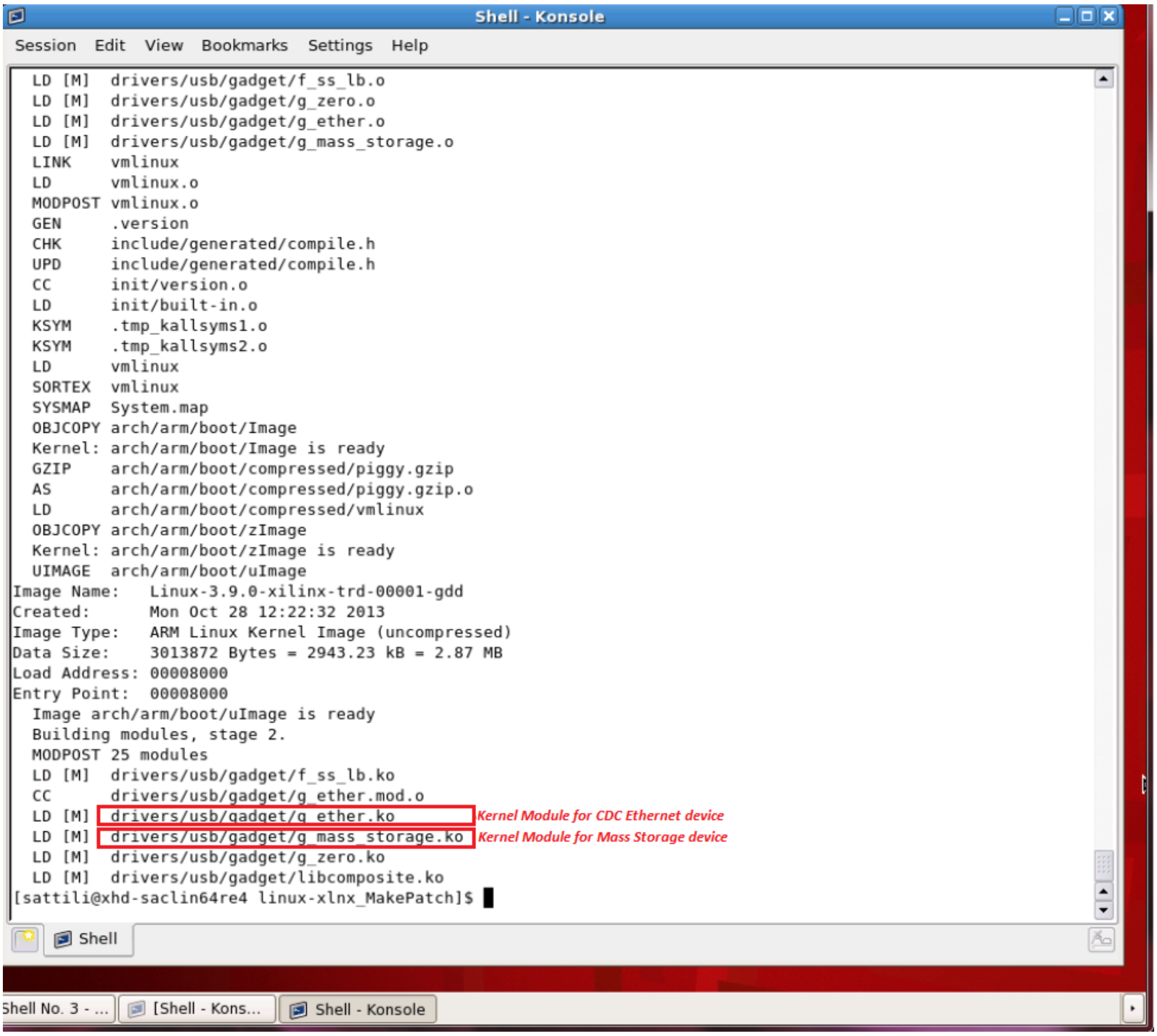
3. Build Kernel Image

Build the Linux kernel. The generated kernel image can be found at:

```
LINUX_ROOT/linux-xlnx/arch/arm/boot/uImage
```

Make:

```
make ARCH=arm uImage modules UIMAGE_LOADADDR=0x8000
```



```
LD [M]  drivers/usb/gadget/f_ss_lb.o
LD [M]  drivers/usb/gadget/g_zero.o
LD [M]  drivers/usb/gadget/g_ether.o
LD [M]  drivers/usb/gadget/g_mass_storage.o
LINK    vmlinux
LD      vmlinux.o
MODPOST vmlinux.o
GEN     .version
CHK     include/generated/compile.h
UPD     include/generated/compile.h
CC      init/version.o
LD      init/built-in.o
KSYM    .tmp_kallsyms1.o
KSYM    .tmp_kallsyms2.o
LD      vmlinux
SORTEX  vmlinux
SYSMAP  System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
GZIP    arch/arm/boot/compressed/piggy.gzip
AS      arch/arm/boot/compressed/piggy.gzip.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
UIMAGE  arch/arm/boot/uImage
Image Name:   Linux-3.9.0-xilinx-trd-00001-gdd
Created:      Mon Oct 28 12:22:32 2013
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    3013872 Bytes = 2943.23 kB = 2.87 MB
Load Address: 00008000
Entry Point:  00008000
  Image arch/arm/boot/uImage is ready
  Building modules, stage 2.
  MODPOST 25 modules
  LD [M]  drivers/usb/gadget/f_ss_lb.ko
  CC      drivers/usb/gadget/g_ether.mod.o
  LD [M]  drivers/usb/gadget/g_ether.ko          Kernel Module for CDC Ethernet device
  LD [M]  drivers/usb/gadget/g_mass_storage.ko   Kernel Module for Mass Storage device
  LD [M]  drivers/usb/gadget/g_zero.ko
  LD [M]  drivers/usb/gadget/libcomposite.ko
[sattili@xhd-saclin64re4 linux-xlnx_MakePatch]$
```

4. Configure and Build the Device Tree

This example assumes the zc702 board is used, but the instructions are similar for other boards. Open the file:

```
arch/arm/boot/dts/zynq-zc702.dts
```

Change the dr_mode parameter for USB 0 controller:

```
usb@e0002000 {
compatible = "xlnx,ps7-usb-1.00.a";
      reg = <0xe0002000 0x1000>;
      interrupts = <0 21 4>;
      interrupt-parent = <&gic>;
      dr_mode = "host"; Note → change this to  "peripheral";
      phy_type = "ulpi";
  };
```

Build the new device tree

```
./scripts/dtc/dtc -I dts -O dtb -f arch/arm/boot/dts/zynq-zc702.dts -o
devicetree.dtb
```

5. Copy Files to SD Card

Copy the kernel image, device tree and ramdisk files to the root directory of the SD card. For both examples, include these files:

```
libcomposite.ko
uImage (found in arch/arm/boot/uImage)
devicetree.dtb
```

```
uramdisk.tar.gz
```

Also copy the loadable module files listed below, depending on the example being followed. At the shell prompt, these modules will be copied to the drivers/usb/gadget directory. Alternatively, include these files in the ramdisk image.

Mass Storage Device Module Files

```
g_mass_storage.ko
```

Ethernet RNDIS Module Files

```
g_ether.ko
```

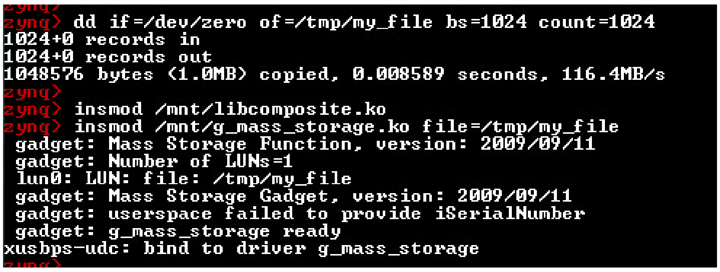
6. Execute Commands at the Shell Prompt and Exercise the Device

At the shell prompt, enter the commands as appropriate for the example you are following. Then use the device as indicated.

Mass Storage Device Shell Commands

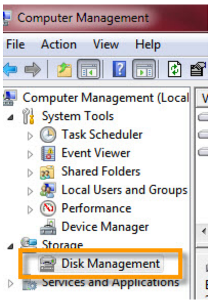
The *dd* command creates a disk image with an empty 1MB file that is filled with zeros (temp/my_file). This is used as storage for the mass storage device example. The *insmod* command inserts the loadable module into the kernel.

```
dd if=/dev/zero of=/tmp/my_file bs=1024 count=1048576
insmod /mnt/libcomposite.ko
insmod /mnt/g_mass_storage.ko file=/tmp/my_file
```



Now, plug-in the USB cable between the development board and the Host PC. On the Host PC, the development board is detected as a Storage device.

NOTE: To see the drive in Windows Explorer, use the Windows Disk Management tool to initialize the disk to create a partition table and create a simple volume.



Now, the drive is visible in Windows Explorer and the user can transfer files to and from the Zynq board.

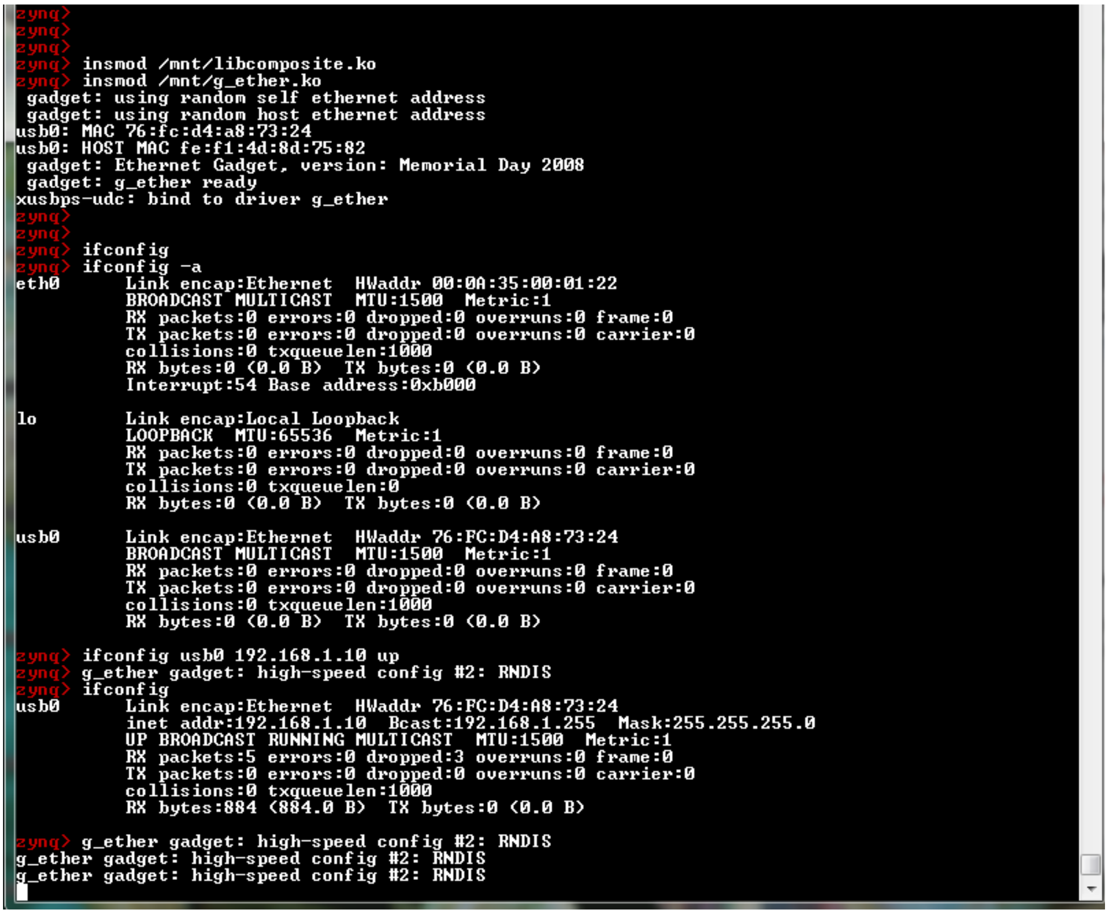
Ethernet RNDIS Shell Commands

At the shell prompt, enter below commands.

```
insmod /mnt/libcomposite.ko
insmod /mnt/g_ether.ko
```

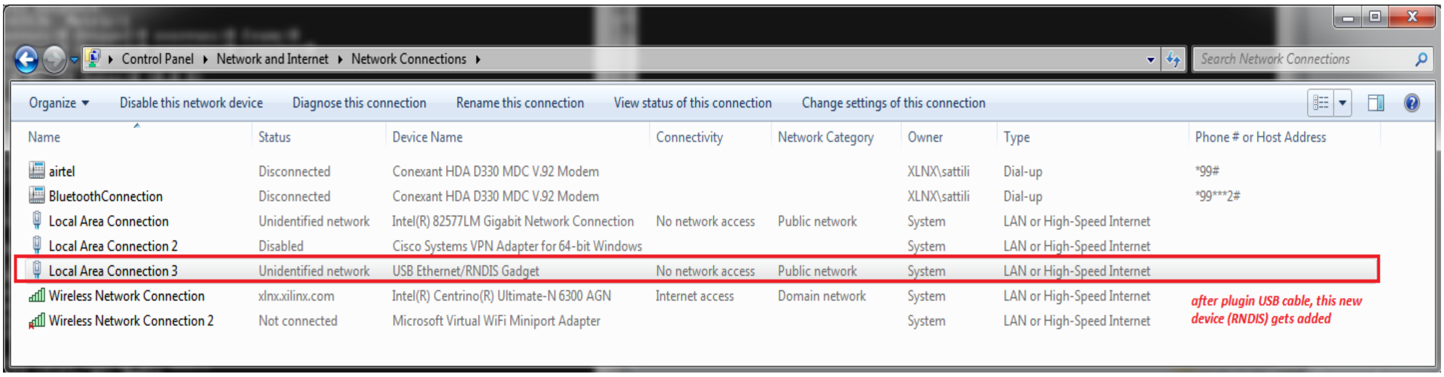
Configure the newly created Zynq CDC Ethernet RNDIS device.

```
ifconfig -a
ifconfig usb0 192.168.1.10 up
```



Now, plug-in the USB cable from board to Host PC.

On Host PC, the Zynq board is detected as Ethernet device.



Configure the above device to have a static IP address (ex: 192.168.1.200)

Now, user can ping the Zynq device from Host PC.