Vybrid Getting Started Guide with Linux-MQX

Rev. 1.0

Freescale
Microcontroller Group

Revision History

Revision	Date	Changes
1.0	5/20/2013	First Release

1 Introduction

This document details the steps required to develop a combined Linux & MQX application for Vybrid Controller Solutions. It is targeted to users with minimal Linux experience, but contains enough detail to build a dual core application using Linux + MQX on Vybrid's dual-core architecture. Linux BSP support is provided through LinuxLink by Timesys. This document contains some of the Timesys material from their website, but is also supplemented by screenshots and specific details targeted towards first-time users of Linux or Vybrid hardware. It includes examples of setting up a Linux virtual environment on a PC and adds supplemental details on some of the steps required to get an application running.

The targeted Vybrid hardware is a superset dual core ARM® Cortex™-A5 + ARM® Cortex™-M4 device. Linux runs on the A5 core up to 500 MHz, and a real time OS such as MQX runs on the M4 at up to 167 MHz. This heterogeneous architecture provides a unique platform for applications (HMI + connectivity) and real time processing.

2 Requirements

Hardware
TWR-VF65GS10 Tower Board

Recommend peripheral tower cards
TWR-LCD-RGB – LCD screen (available with TWR-VF65GS10-DS5 or TWR-VF65GS10-PRO)
TWR-SER2 – Dual Ethernet ports (available with TWR-VF65GS10-DS5 or TWR-VF65GS10-PRO)
TWR-SER – Single Ethernet port (available with TWR-VF65GS10-KIT)

Connectivity
A dual network connection (router) on the Linux Host is highly recommended. This allows simultaneous connections your target board and virtual environment, while also having full network access. If a router is not present, then your board will need to be directly connected to your computer for NFS booting and debugging.

Software

- Timesys LinuxLink account – Complimentary with TWR-VF65GS10 purchase
- PC running Linux virtual environment (Ubuntu under VirtualBox or VMWare) or dedicated Linux box
- MQX installer for Linux, available on the Timesys LinuxLink pages
- ARM DS-5 IDE installed in the Linux environment
- DS-5 Timestorm plug-in, available on the Timesys LinuxLink pages

Note: ARM DS-5 can be installed on Windows and will support dual core application debug, such as MQX running on both cores. Linux is the recommended environment because the Timestorm Linux Development plug-in integrates into the toolchain that is part of LinuxLink Desktop Factory build environment.

The TWR-VF65GS10 Tower module includes the following features:

- Vybrid VF61NS151CMK50 microprocessor (Dual Core ARM Cortex™-A5 @ 500 MHz + ARM Cortex™-M4 @167 MHz, 1 MB SRAM, 512 L2 Cache for Cortex™-A5, and advanced security)
- Kinetis K20DX128VFM5-based OpenSDA circuit
- 1 Gb x 16 (128 MB) DDR3 in 96 FBGA package (Samsung)
- 2 Gb x 16 (256 MB) NAND flash (Micron)
- Two 16 MB quad I/O serial flash (Spansion)
- Dual Ethernet MAC with RMII Interface
- Dual USB with on-chip PHY
- Interfaces to TWR-LCD-RGB board
- Four user-controlled status LEDs
- Two mechanical push buttons for user input and one for reset
- Potentiometer and MMA8451Q three-axis digital accelerometer
- Micro SD Card slot
- Independent battery-operated power supply for real-time clock and tamper detection modules

3 Virtual Linux Setup Instructions

This section describes how to setup a virtual Linux environment using the virtualization software package VirtualBox. VMWare may also be used, but only the steps for VirtualBox are described below. If you have a Linux environment setup already, you can skip to Chapter 4, which will describe how to get started with a minimal custom BSP for Vybrid.

Creating a new virtual machine will require significant hard drive space available on the PC. **A virtual disk file 20GB large is recommended, though even up to 40GB may be desirable.** Make sure you also backup any data on the drive where you are installing the virtual machine in case any mishaps may occur.

A virtual machine simply means that a Linux environment is running on top of Windows operating system. The look and feel of Linux is almost identical to a dedicated Linux machine. The interface between the host (Windows) and the guest (Linux) is handled by virtualization software such as VirtualBox or VMWare. Linux is launched using this interface software, similar to how a typical program is launched using the Windows Start button. Other methods to run Linux on a Windows PC exist, such as booting from a USB drive, but they are not covered in detail here. For more information, see http://en.wikipedia.org/wiki/Virtual_machine.

3.1 Virtual Machine Setup – Ubuntu + Oracle VirtualBox

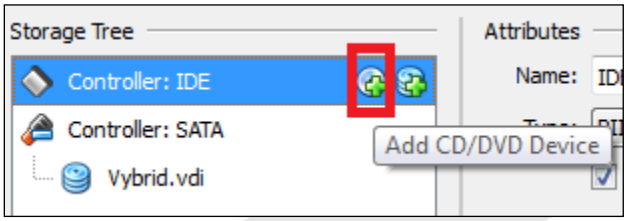
1. Download and install Oracle VirtualBox and the associated VirtualBox Extension Pack.

2. Download Ubuntu Desktop v12.04 or later. This is will be a 700+ MB .iso file. When getting Ubuntu, get the following based on the desktop CPU:
    a. For 32 bit - ubuntu-12.10-desktop-i386.iso
    b. For 64 bit - ubuntu-12.10-desktop-amd64.iso

3. You will need to create a virtual machine before continuing. Click on the "New" button 

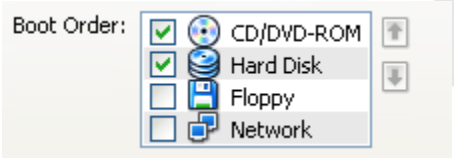
and follow the prompts to create a virtual machine. Make sure you allocate at least 20GB for the virtual hard drive (default is only 10GB). Again, 30-40 GB is recommended if available. Also allocate at least 1GB of memory for the Virtual Machine (assuming 4GB of physical RAM on your computer).

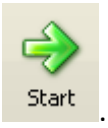
4. Add the Ubuntu .iso file to VirtualBox. Go to **Settings->Storage->Controller: IDE**. Click on the icon for 'Add CD/DVD' and navigate to the iso file to add it.



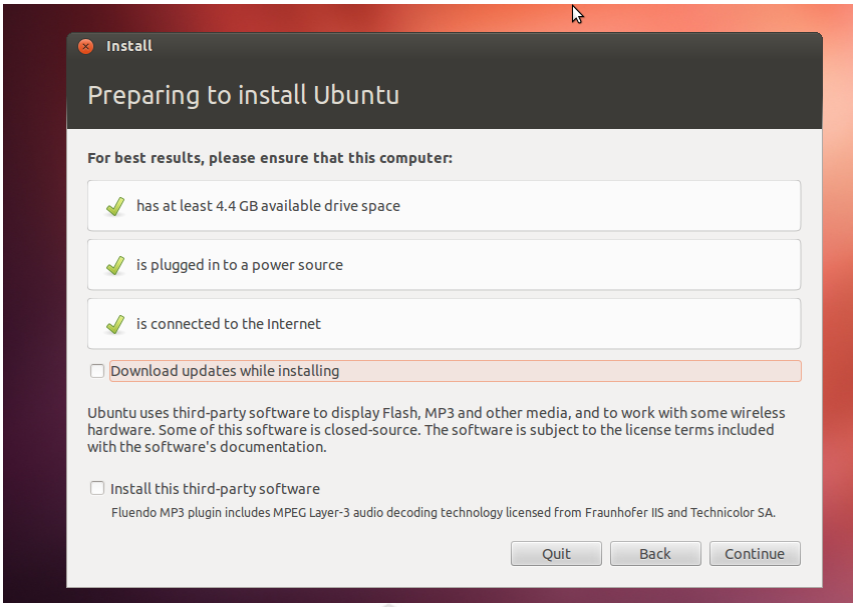
    a. Note: it is strongly recommended that you backup any data on the drive where you create the new partition.

5. Change the boot order to boot off the iso. Go to **Settings->System** to put CD/DVD first in the boot order.

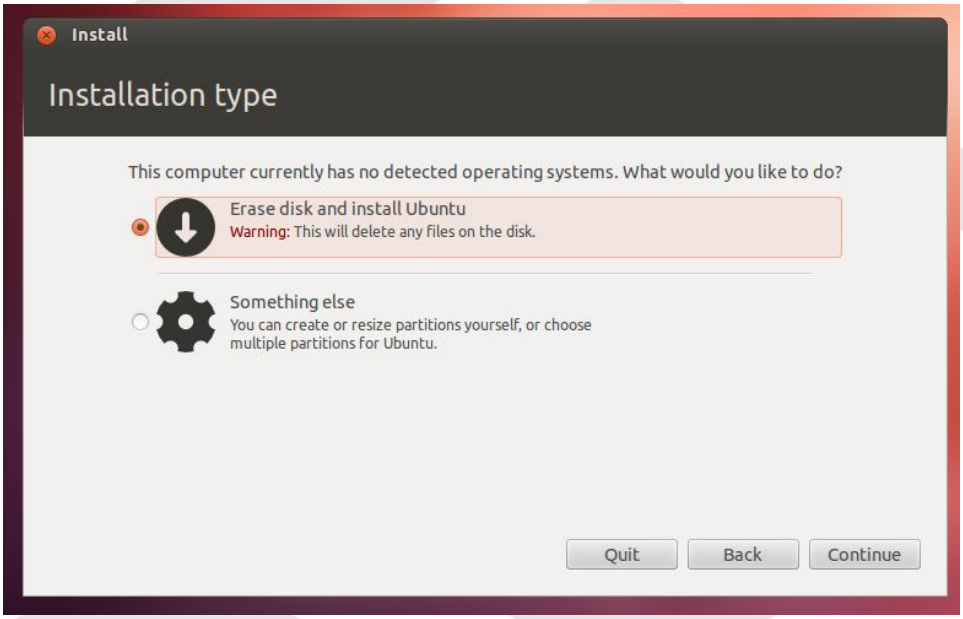


6. Click 'Start' 

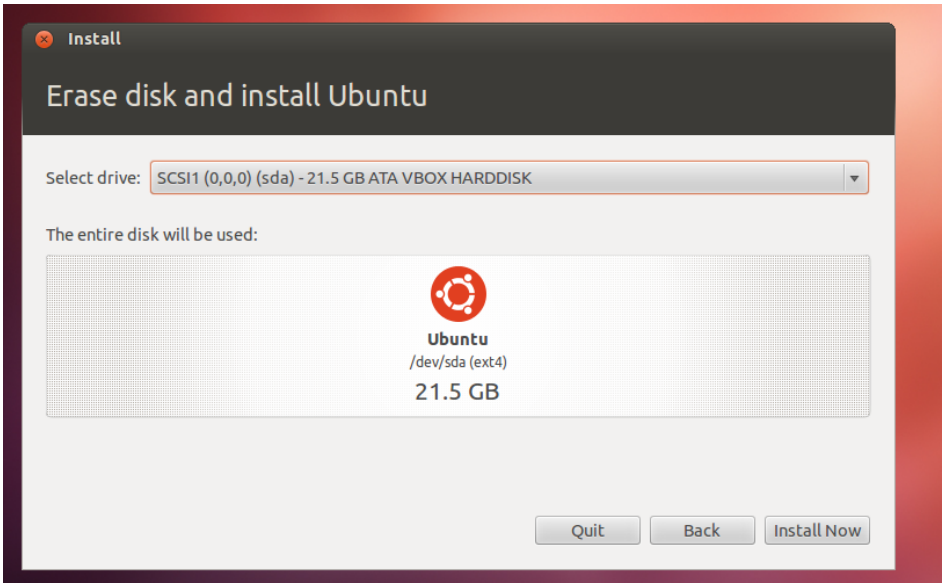
. This will launch Ubuntu from the disk image (.iso).
7. Click Install Ubuntu. If you had not setup a virtual disk already, it will ask you to create a new disk. Make sure to allocate at least 20GB (default choice is 10GB) and even more (up to 40GB) is recommended.



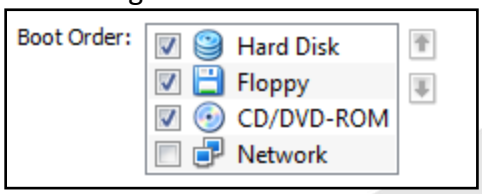
8. Click Continue. Select 'Erase disk and install Ubuntu'. This will not erase your physical hard drive, but instead creates a block of memory for the virtual disk drive you created earlier.



You may also see the following screen, where you are asked to choose a drive to erase. Make sure you select the **VBOX HARDDISK**. It will be the same size as the virtual hard disk file you created earlier.



9. Click Install Now and go through the rest of the prompts for time zone, username, and more as Linux installs to the virtual hard drive
10. After Linux has finished installing, shutdown your virtual machine. Then in VirtualBox, go to **Settings->Storage**. You will see the SATA virtual hard drive (.vdi) that has been created from the installation process that contains the virtual Linux environment. The .vdi file is located under C:\Users\<YourLogin>\VirtualBox VMs\<VM Name>. This can later be resized if necessary with VBoxManage but that will not be covered here. Within VirtualBox, Go to **File->Virtual Media Manager** to see the .vdi partition.
11. Also change the boot order so that the hard drive will boot first. This is in **Settings->System**



3.2 VirtualBox Guest Addtions

VirtualBox comes with "Guest Additions" that optimize the guest operating system (Linux in this case) for better performance and usability. This also enables shared folder functionality (to enable Linux and Windows to read/write the same folder, making it easy to transfer files from host OS to the virtual OS) as well as clipboard functionality for copy/paste.

In the VirtualBox window, add a CD/DVD drive as you did for the Linux iso file, and point it to the Guest Additions .iso file that is automatically included as part of VirtualBox. The file can be found in the VirtualBox installation folder: **C:\Program Files\Oracle\VirtualBox\VBoxGuestAdditions.iso**

Start your virtual machine and it should boot into Ubuntu. You can follow the directions on the VirtualBox website for installing it on Linux.

1. Open a Terminal by clicking on the Ubuntu logo and searching for "Terminal"
2. Run **sudo apt-get install dkms** first *IMPORTANT*
3. Go to where the CDROM is mounted (usually "**cd /media/VBOXADDITIONS_4.2.6_82870**") and run:
   sudo ./VBoxLinuxAdditions.run

There may be some cases where Ubuntu does not properly come up on the first boot due to graphics issues. If this is the case, see if you can get to the terminal via a screen option, or try hitting CTRL+ALT+F1 keys. Once a terminal appears and you login, the CDROM needs to be mounted to be accessible. Use the following commands:

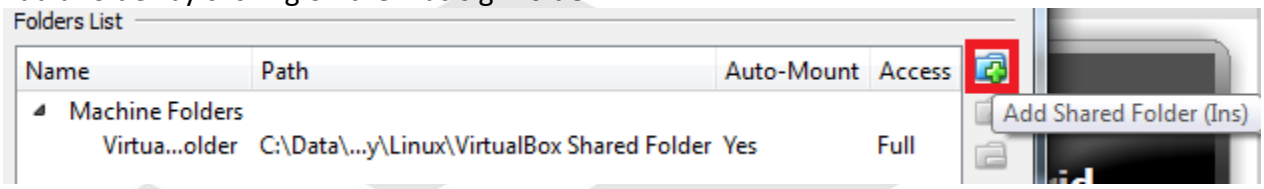
1. **sudo mkdir /media/cdrom**
2. **sudo mount /dev/sr0 /media/cdrom** (it may be /dev/cdrom instead of /dev/sr0)
3. **cd /media/cdrom**
4. **sudo apt-get install dkms**
5. **sudo ./VBoxLinuxAdditions.run**

After the Guest Additions have been installed, it should let Ubuntu boot correctly.

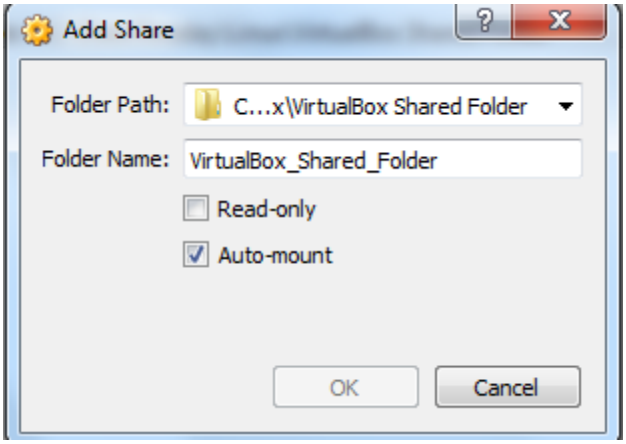
3.3 VirtualBox Shared folder Setup

Creating a shared folder allows easy file sharing between Linux and Windows. To enable this feature follow the steps below:

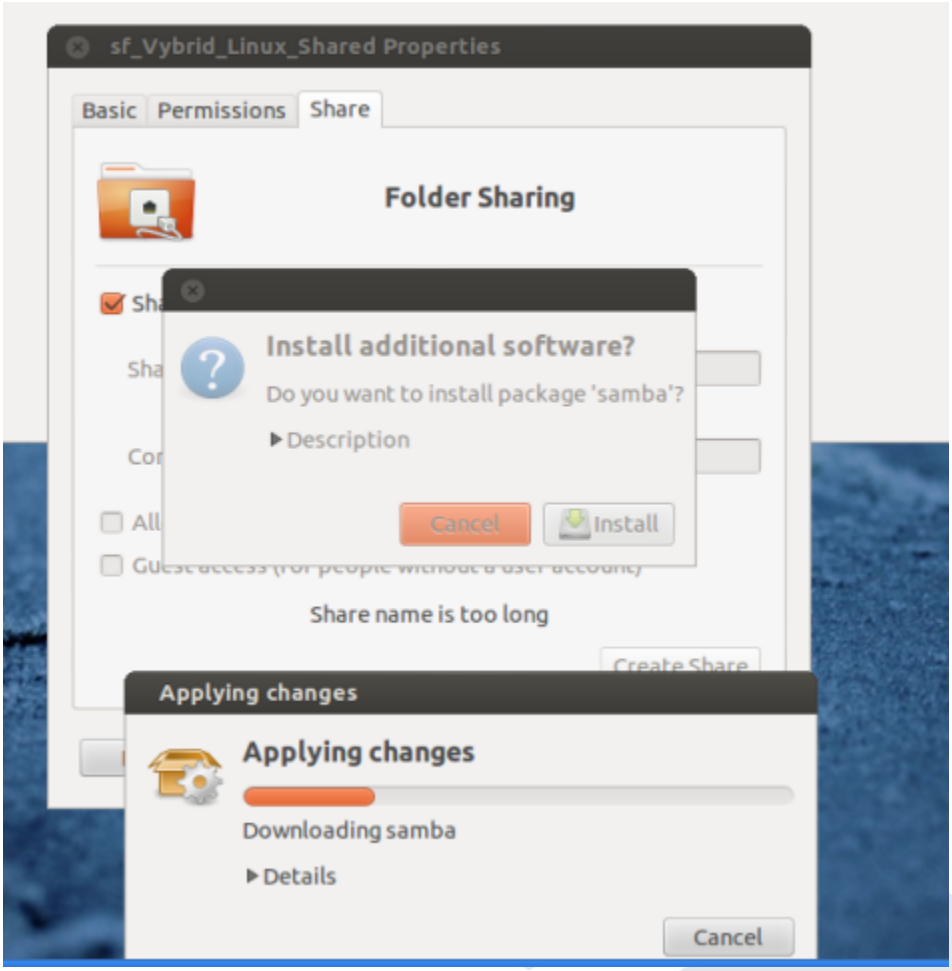
1. Install the guest additions as described in the previous section
2. Shutdown Linux
3. In Windows, create a folder on the hard drive that will become the designated shared folder
4. In VirtualBox, go to **Settings->Shared Folders**
5. Add a folder by clicking on the Plus sign folder



6. In the dialog box that comes up, select Other in the Folder Path box, and navigate to the folder you created
7. Give it a name and check to Auto-Mount



8. Click OK to save the settings.
9. Start up Ubuntu
10. Now sharing needs to be enabled inside of Linux. Open the home folder in Ubuntu (Folder icon on left side navigation panel). Navigate to the shared folder at **/media/sf_<shared_folder>** (The sf_ is added by default)
11. Right click on the folder. Select **Sharing Options** and select **Share This Folder,** and hit **Create Share** to save.
12. Ubuntu will prompt you to install several software packages to support sharing, and accept all of them



13. If you still cannot access the shared folder from the Linux side, open a terminal try:
sudo adduser <login_name> vboxsf

```
r39626@r39626-VirtualBox: ~
r39626@r39626-VirtualBox:~$ su
Password:
root@r39626-VirtualBox:/home/r39626# cd /media/sf_Vybrid_Linux_Shared/
root@r39626-VirtualBox:/media/sf_Vybrid_Linux_Shared# l
root@r39626-VirtualBox:/media/sf_Vybrid_Linux_Shared# ls
root@r39626-VirtualBox:/media/sf_Vybrid_Linux_Shared# exit
exit
r39626@r39626-VirtualBox:~$ sudo adduser r39626 vboxsf
[sudo] password for r39626:
Adding user `r39626' to group `vboxsf' ...
Adding user r39626 to group vboxsf
Done.
r39626@r39626-VirtualBox:~$
```

4 Vybrid Linux Setup Instructions

This section describes how to setup your Linux environment to build and debug Linux on Vybrid. The Timesys website has in-depth information on this topic, and the packages discussed in this section can be found on the Vybrid LinuxLink page. Note that you will need to be registered with Timesys to have access to the Vybrid BSP and documentation.

4.1 Basic Linux Commands

Even though modern Linux distributions have improved user interfaces for ease of use, new Linux users should still get familiar with command line options which allow navigation of directories, listing files, etc. Some basic tutorials are available here:
http://www.linux.org/tutorial/view/beginners-level-course
http://www.ee.surrey.ac.uk/Teaching/Unix/

4.2 Explanation of sudo

The **sudo** command allows a normal user to temporarily have root (administrator) privileges. By default Ubuntu does not allow access to the root account, and all privileged commands must be run by appending **sudo** to the beginning of the command, which you will see throughout this document. It is possible however to create a root password and bypass the need for sudo.

a. $sudo su ($ is prompt when not logged in as root)
b. passwd (change password command)
c. Enter new password (twice) and you will be logged in as root, showing # at the terminal prompt.
d. From then on you can login as root by typing “su” and the root password

Commands in this document that require root privileges will either have **sudo** or a **#** sign in front to signal this. A word of caution, running as root can be dangerous – use it with caution and only when required to do so.

4.2.1 Explanation of “Host” and “Target”

In the embedded Linux world, the “Host” is the PC and its operating system. In this case, it’s Ubuntu running in the virtual machine. “Target” is the embedded device, in this case the Vybrid tower board.

4.2.2 Explanation of U-Boot and the Root File System

U-Boot is an open source bootloader that is used to boot Linux. It can also boot non-Linux applications such as ELF files or plain binary files. For Vybrid, it resides on the SD card, but support for booting from

NAND Flash is being added as well. U-Boot provides the basic clock and peripheral initialization (UART, Ethernet, SD Card) that can then be used to continue booting Linux.

The "root file system" (RFS) is the top level of the Linux file system, under which all the programs, configuration files, and other files are stored. It's the items you see if you type "ls /" in a Linux computer or at the Linux prompt once Linux running on your tower system.

The Linux kernel and the root file system can reside on an SD card, NAND Flash, or in RAM. Or it can be located on your computer's hard drive and accessed by U-Boot and Linux via an Ethernet connection (called NFS Boot). In a deployed system, the RFS would reside on one of the memory interfaces on the board since a Linux host could not be connected to it at all times to boot Linux.

However the advantage of having the file system and/or kernel on your computer's hard drive is to allow for easier and quicker application development. You can easily copy your application to the folder on your hard drive where the root file system is located, and run your program on the board immediately. If everything was located on the SD card instead, you would need to take out the SD card, plug it into your computer, load the application on the SD card, remove it from the computer and put it back into the board, and restart Linux; a much more lengthy process. This guide will cover how to work with both methods.

4.2.3 Useful Packages

The following packages are either required by LinuxLink or are useful in developing in Linux. Copy and paste into a Linux terminal:

- **Ubuntu 12.10**
  - apt-get install build-essential libc6-dev libtool sharutils libncurses5-dev libgmp3-dev libmpfr-dev gawk gettext bison flex gperf indent texinfo libgtk2.0-dev libgtk2.0-bin libsdl1.2-dev swig python-dev texlive-latex3 texlive-extra-utils binutils-dev automake guile-1.8 icon-naming-utils libdbus-glib-1-dev wget gtk-doc-tools libxml-parser-perl zip unzip ecj fastjar x11-xkb-utils libglade2-dev libperl-dev python-libxml2 libexpat1-dev gconf2 groff libc6-dev
  - run `sudo dpkg-reconfigure dash`
    - respond "No" to the prompt asking "Install dash as /bin/sh?"
- **Ubuntu 12.04 LTS or older**
  - apt-get install build-essential libc6-dev libtool sharutils libncurses5-dev libgmp3-dev libmpfr-dev gawk gettext bison flex gperf indent texinfo libgtk2.0-dev libgtk2.0-bin libsdl1.2-dev swig python-dev texlive-latex3 texlive-extra-utils binutils-dev automake guile-1.8 icon-naming-utils libdbus-glib-1-dev wget gtk-doc-tools libxml-parser-perl zip unzip ecj fastjar x11-xkb-utils libglade2-dev libperl-dev python-libxml2 libexpat1-dev gconf2 groff libc6-dev-amd64
  - run `sudo dpkg-reconfigure dash`
    - respond "No" to the prompt asking "Install dash as /bin/sh?"
  - For 64-bit host machines only: **apt-get install ia32-libs libc6-dev-i386**

- At some point while using Ubuntu you'll be prompted to install software package updates. Make sure to do this.
- Note: The Factory Host requirements for all Linux environments can be found here:
    - https://linuxlink.timesys.com/docs/wiki/factory/FactoryHostRequirements

4.2.4 Useful Tweaks

- Create a .netrc file in your user directory to easily login to Timesys repositories.
  gedit ~/.netrc
  Then add the following entries

```
machine linuxlink.timesys.com
  login <timesys_login>
  password <timesys_pword>

machine git.timesys.com
  login <timesys_login>
  password <timesys_pword>
```

  Finally, change the permissions on the .netrc file
  chmod 600 ~/.netrc

4.3 Download and Install the SDK installer

Download one of the SDK installer files from your LinuxLink dashboard. Several pre-built starting points are available for download, including the Vybrid Out-Of-Box Experience (OOBE) demo code. After selecting a pre-built starting point, you will be taken to a page where you can download the SDK installer file for that particular project, which will always be named **twr_vf600-development-environment.sh**. This installer includes the bootloader (U-Boot), kernel, toolchain and RFS. It is everything needed to boot Linux on your Vybrid board. It does not however include the build tools to recompile the kernel, which is in the Desktop Factory Installer. This will be discussed in a later section.

Alternatively, you may build your own BSP/SDK using Timesys' Web Factory as your own custom starting point. After it is built via the Web Factory and ready for download, you will get an email notification to download your custom SDK installer.

After downloading, the ~700MB .sh file, it will be available in /home/user/Downloads (~/Downloads). The .sh file is a self extracting shell script that installs the kernel source, RFS, and toolchain on the host.

- Navigate to the Downloads folder and set executable permissions by typing **chmod +x twr_vf600-development-environment.sh**
- Run the installer by typing **./twr_vf600-development-environment.sh**
- Select default path (/home/user/Timesys) for installation

- Note: Don't install when logged in with root privileges. According to Timesys, running the build environment as the root user is discouraged in order to ensure the safety of the host work station.

4.4 TWR-VF65GS10 Linux Configuration

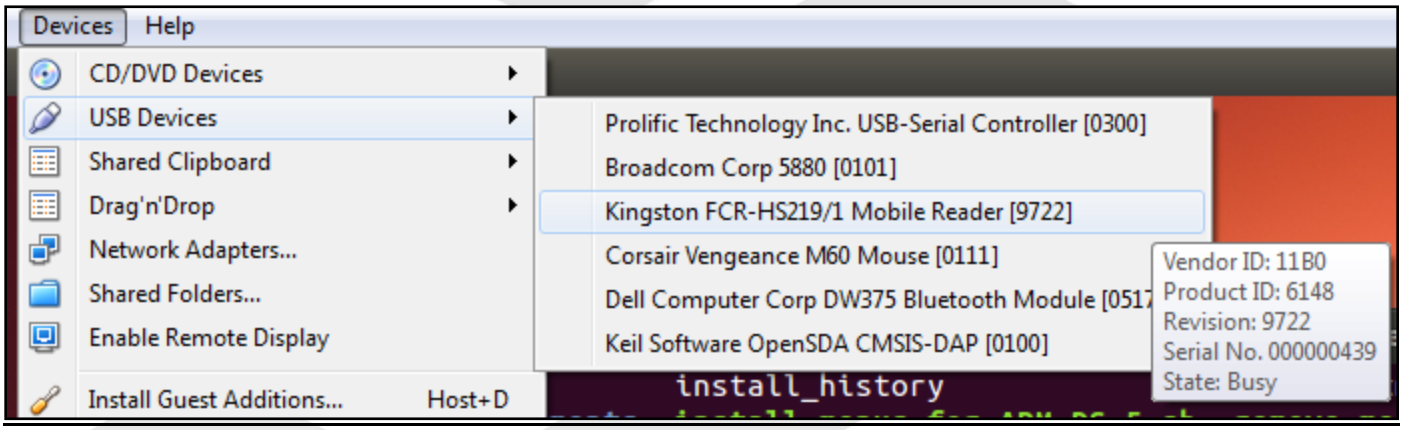
The Timesys Getting Started Guide describes the steps necessary to configure the SD Card, Ethernet, and tower board to enable it to boot Linux. This section will cover installing and using U-Boot on the SD card to boot Linux on either the SD card or over Ethernet. Highlights and additional details for this information are found below.

4.4.1 Setting up the SD Card

Follow the getting started link on Timesys page to update the SD card with a new version of U-Boot and the Linux kernel. For convenience, the steps are below and expanded for further explanation for people new to embedded Linux.

4.4.2 Special Steps for a Linux Virtual Enviroment

- Using the built-in laptop SD Card reader does not work by default with a virtual Linux environment. An easy workaround for this is to use a USB card reader to format the SD card.
- If using a USB card reader, after plugging it in, you need to select it in VirtualBox to enable Linux to see it. Click on the card reader device (Kingston FCR-HS219 in this case). Only then will you be able to see the SD card in the virtual Linux environment.



Note: The following instructions will overwrite data on your SD card. Make sure to back up any valuable data on the SD card before performing this operation. Also make sure the card is unlocked (via the plastic tab on the physical card) before executing these steps

1. From the host, run:

```
lsblk
```

Note which memory devices are currently connected to your computer. We will use this information to determine the SD Card device name.

2. Connect your SD card to your host, via a SD Card slot, an SD card adapter, or a USB adapter.
3. Determine the device name of the SD Card. You can run **lsblk** again and see which device has now been added. **dmesg** could also be used. In the following example, the device is */dev/sdb*, which contains one partition *sdb1*.

```
$ dmesg | tail
[88050.184080] sd 4:0:0:0:[sdb] 1990656 512-byte hardware sectors:(1.01 GB/972
MiB)
[88050.184821] sd 4:0:0:0:[sdb] Write Protect is off
[88050.184824] sd 4:0:0:0:[sdb] Mode Sense: 03 00 00 00
[88050.184827] sd 4:0:0:0:[sdb] Assuming drive cache: write through
[88050.185575] sd 4:0:0:0:[sdb] 1990656 512-byte hardware sectors:(1.01 GB/972
MiB)
[88050.186323] sd 4:0:0:0:[sdb] Write Protect is off
[88050.186325] sd 4:0:0:0:[sdb] Mode Sense: 03 00 00 00
[88050.186327] sd 4:0:0:0:[sdb] Assuming drive cache: write through
[88050.186330]  sdb: sdb1
```

4. Unmount all partitions on the SD Card. With root permissions, run:

```
sudo umount /dev/sdX+ ("umount /dev/sdb1"  for this example)
```

where /dev/sdX is the device name found in the previous step. X is the device identifier, and + is the partition number on that device.

5. Transfer the U-Boot binary (u-boot.imx) to your SD card. A, this file is located at: ***/home/<username>/timesys/twr_vf600/bootloader/u-boot.imx*.**

```
sudo dd if=/home/<username>/timesys/twr_vf600/bootloader/u-boot.imx of=/dev/sdX
bs=512 seek=2
```

This command writes the U-Boot binary to a 1024-byte offset on the SD Card. This is done because the Image Vector Table (IVT) information used by the Vybrid BootROM must start at a 1024 byte offset. The Vybrid BootROM was designed like that to avoid overwriting or interfering with the partition table, which is located in the first 512 bytes of the SD card.

6. Now make 2 partitions on the SD card

    - 5MB FAT file system for the kernel image
    - 1GB Linux ext3 file ystem for the Root File System (RFS)

```
sudo fdisk /dev/sdX
Command (m for help): o
Building a new DOS disklabel with disk identifier 0x900ced76.
Changes will remain in memory only, until you decide to write them.
```

```
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected
by w(rite)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (2048-7744511, default 2048): 2048
Last sector, +sectors or +size{K,M,G} (2048-7744511, default 7744511): +5M

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First sector (12288-7744511, default 12288): 12288
Last sector, +sectors or +size{K,M,G} (12288-7744511, default 7744511): +1G

Command (m for help): p

Disk /dev/sdc: 3965 MB, 3965190144 bytes
122 heads, 62 sectors/track, 1023 cylinders, total 7744512 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x64206a6b

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048       12287        5120    c  W95 FAT32 (LBA)
/dev/sdc2           12288     2109439     1048576   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

7. Format and label the partitions:

```
sudo mkfs.vfat -n KERNEL /dev/sdX1
sudo mkfs.ext3 -L RFS /dev/sdX2
```

Remove the SD card and re-insert it. Verify the KERNEL and RFS partitions are mounted

```
df -h
> Filesystem            Size  Used Avail Use% Mounted on
>          .
>          .
>          .
> /dev/sdX1             6.2M     0  6.2M   0% /media/KERNEL
> /dev/sdX2            1008M   34M  924M   4% /media/RFS
```

8. Copy the kernel to the SD card:

```
sudo cp /home/<userlogin>/timesys/twr_vf600/uImage-3.0-ts-armv7l /media/KERNEL/
```

9. Explode the RFS tarball onto the SD card:

```
cd /media/RFS/
sudo tar -xf /home/<userlogin>/timesys/twr_vf600/rfs/rootfs.tar.gz
```

10. Unmount the partitions:

```
sync
cd /media/
sudo umount KERNEL/
sudo umount RFS/
```

Remove the SD Card from your host, and slide it into the slot of your TWR-VF65GS10. You should hear the card 'click' into place.

4.4.3 Serial Port Setup

A serial port can be setup in either Windows or Linux, though due to familiarity, it may be easier to setup in Windows. Use a terminal program setup for 115200, 8 bit data, No Parity, 1 Stop bit with the correct COM port.

A TWR-SER2 or TWR-SER board can be connected to the computer with the appropriate hardware – typically USB-to-SER adapter and DB9 cable to the PC.

On Rev G boards and later, a USB cable plugged into J3 will enumerate as a CDC device, and serial data can be transferred over the USB connection without a need for the TWR-SER or TWR-SER2 boards. It's important to note two things:

- You must reconnect to the COM port after every power cycle, as the connection will be lost
- Close the terminal window that is connected to the virtual COM port before you do that power cycle to avoid possible conflicts.

By default, the Vybrid Tower board is configured for the A5 core running Linux to use SCI1 which routes to the virtual serial port, and the M4 core running MQX to use SCI2 which routes to the TWR-SER or TWR-SER2 board. This is configurable via jumpers on J23 and J24.

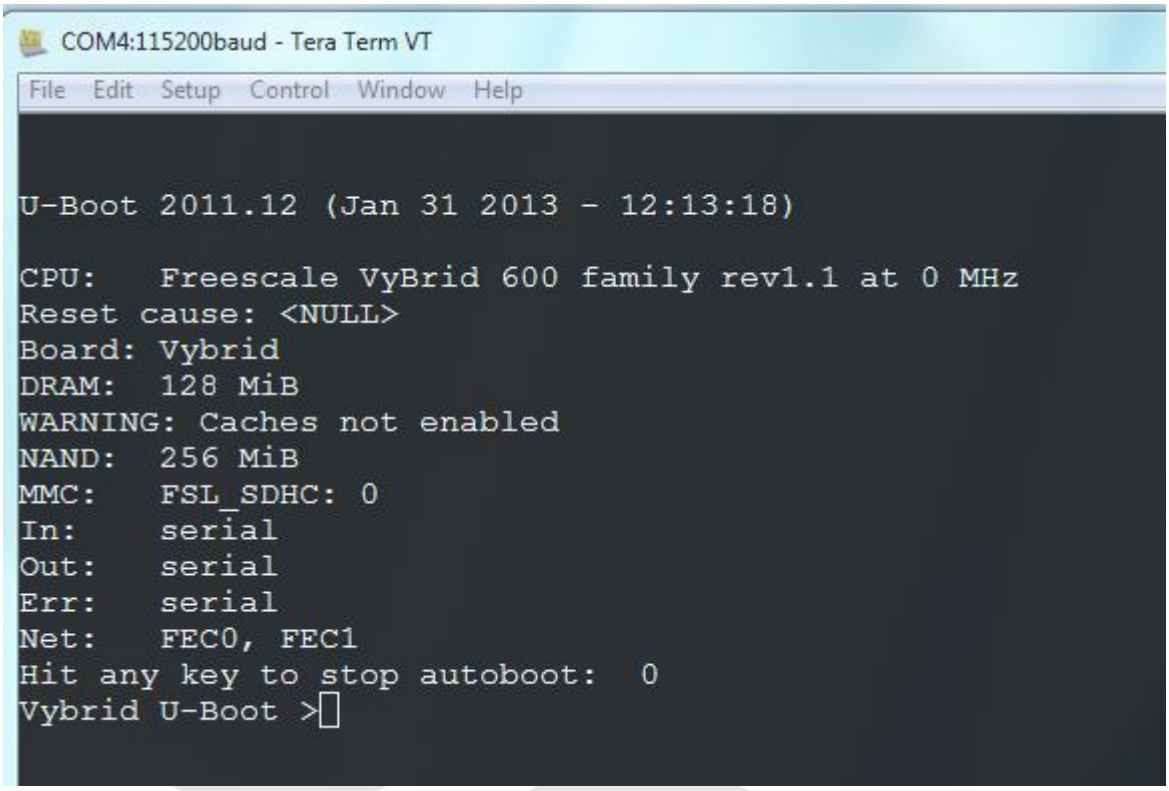
To setup a serial port in Linux:

1. Type '**dmesg | grep tty'** to check which COM ports are being used (typically /dev/ttyS0)
2. Then install minicom with **sudo apt-get install minicom**
3. Run **minicom –o –s –w**
4. Setup accordingly for 115200 8N1 using ttyS0 (or other port being used)

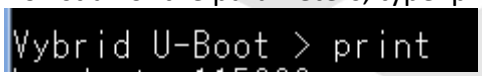
4.4.4 Boot Linux on SD Card

With the newly created SD Card and a terminal connection, you can now boot Linux on your Vybrid board. In this section, the kernel image and the root file system are both located on the SD card.

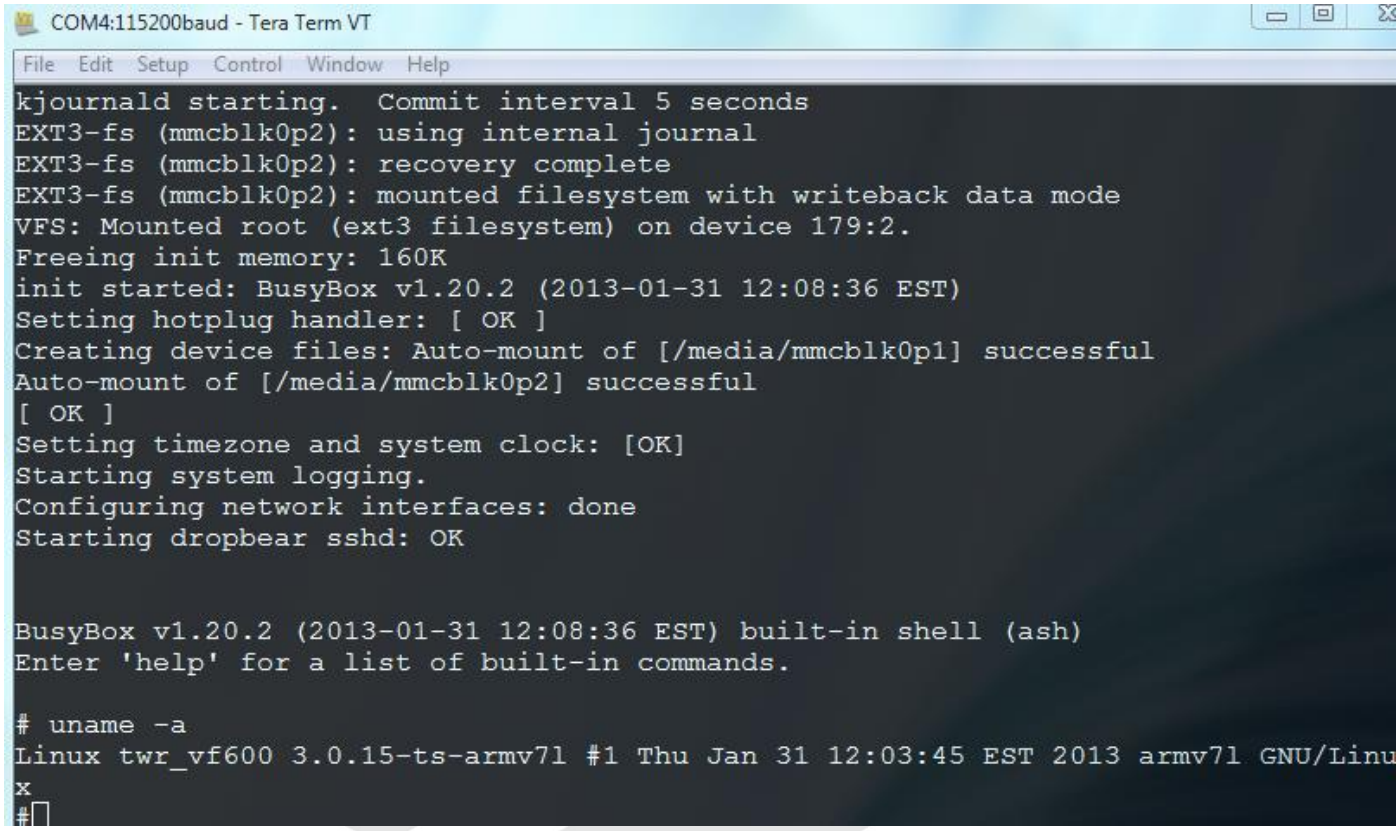
1. Ensure J22 is set to 101100, where 1=Jumper ON. This is boot from the SD card.
2. Make sure the SD card is plugged in
3. Power on the Vybrid board by connecting J3 to a USB port
    - Note: Initially power on first prior to COM port setup, because the virtual serial COM Port will not enumerate until the board is powered. You will have to reconnect to the OpenSDA COM port anytime you do a POR.
4. Open up a terminal as described in the last section. There should be two COM ports. One is a CDC device. Setup both for 115200, 8 bit data, No Parity, 1 Stop.
5. Now hit the reset button on the board (SW3)
6. You should see a UBoot prompt come up (you may have to hit a key to stop boot).



7. To list all of the parameters, type 'print' at the prompt and hit enter.



8. Enter the following U-Boot commands to set the proper U-Boot variables for booting from the SD card and having the root file system also reside on the SD card:
   > setenv bootcmd fatload mmc 0:1 0x81000000 uImage-3.0-ts-armv7l\;bootm 0x81000000
   > setenv bootargs mem=128M console=ttymxc1,115200 root=/dev/mmcblk0p2 rw rootwait
   > saveenv
9. Do a power cycle, reconnect to the COM port, and now Linux should come-up. You should see the prompt (#) on the terminal, or notice terminal messages depending on which starting point SDK/BSP demo is running. If the prompt (#) is visible, you can enter standard Linux commands to your target board. If the prompt is not visible (maybe due to an active demo running), you can start a separate SSH session to the IP address of the board (192.168.1.3, for example) and from there you can enter Linux commands to your target board.



Tip: SSH session can be started from within DS-5. Go to Window->Show View->Other and select Terminal. Click the Settings Icon and select SSH under 'Connection Type'.

4.4.5 IP Address and Network Setup

This section covers how to boot Linux that mounts the root file system (RFS) over the network. Remember that in this scenario, the RFS exists on the laptop hard drive, and the kernel that runs on the target board will mount the RFS over Ethernet. This setup is used for developing and debugging Linux applications. It allows for applications to be loaded and run without having to re-boot the kernel each time. This will require a wired Ethernet connection between the TWR-SER or TWR-SER2 board and the PC. This information is covered in the Timesys Getting Started Guide and has been expanded upon below.

First some packages on Ubuntu need to be installed:
 # apt-get install xinetd tftp tftpd isc-dhcp-server nfs-kernel-server portmap

For development, it is best to have a static IP setup for the board and Linux environment. This way U-Boot options won't change between reboots as you get a new IP address as you would using DHCP. Of course using DHCP is an option and the Timesys Getting Started Guide has information on that. But to simplify the setup, we will use the following static IP addresses instead:

Linux Host (in Virtual Machine): 192.168.0.100
Tower Board: 192.168.0.101

4.4.5.1 Tower Board Setup

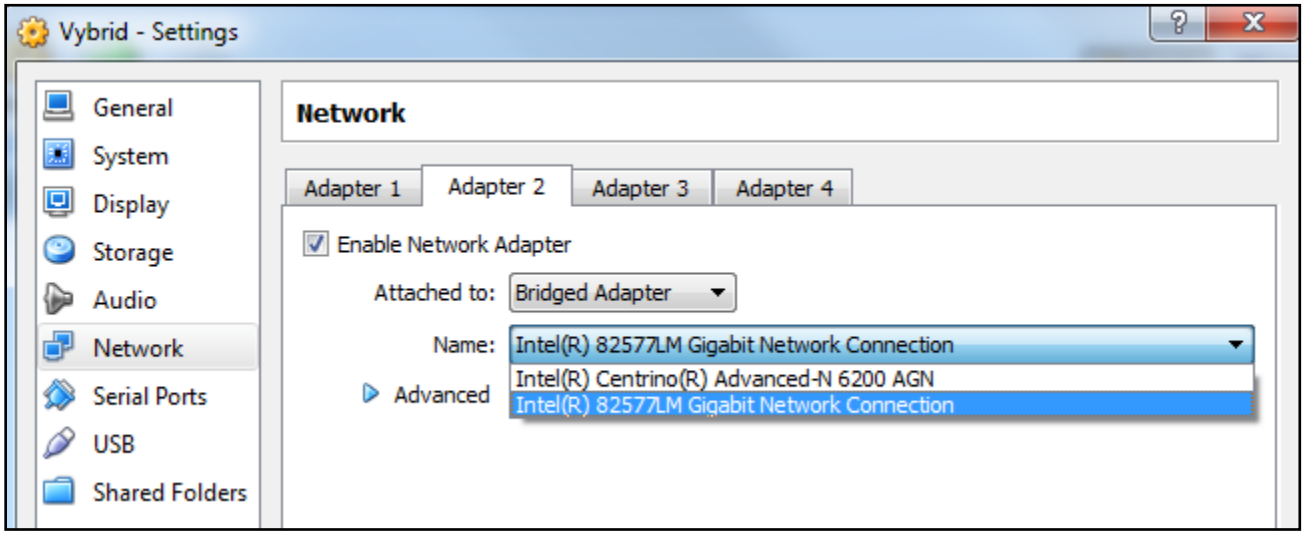
1. If using the TWR-SER module, the configuration needs to be updated since it uses MII mode by default, and the Vybrid tower board requires RMII mode. Remove the TWR-SER from your tower system and make the following modifications:
    - Change **J2** to pins 3-4 to give a 50Mhz clock to the PHY
    - Add jumper to **J3** pins 2-3 to bring up the 50Mhz clock from the TWR-SER
    - Add jumper to **J12** pins 9-10 to put into RMII mode
2. For TWR-SER2, RMII mode at 50MHz is enabled by default.
    - SW1: 11000000 (1=ON)
    - SW2: 10100000 (1=ON)
3. Reconstruct the tower system
4. Connect the Ethernet cable to the Ethernet port of the TWR-SER board to your computer

4.4.5.2 Linux Host Setup

This section describes how to setup a static IP in your Linux host environment. This is not required but will allow the IP address of your virtual host system to remain unchanged. Because u-boot parameters use specific IP addresses, this step is recommended because u-boot parameters may need to be updated in the future to match your virtual IP address if it should ever change.

You could take the existing IP address and make it static, but you would lose the Internet connection in your virtual machine. Instead we want to make use of the virtual environment and add a secondary Ethernet port that is tied to your wired Internet connection, while keeping the original Ethernet port which can use the wireless connection on your laptop.

1. In the Linux virtual environment, type **sudo ifconfig** and note that you should have one Ethernet adapter (**eth0**). The other item listed (**lo**) is a virtual port for loopback mode.
2. Shutdown the Linux virtual machine
3. In VirtualBox, go to **Settings->Network** and click on the **Adapter 2** tab
4. Check "**Enable Network Adapter**" and then in **Attached to:** select "**Bridged Adapter**"
5. Under **Name:** select your wired Ethernet port. This will often have "Gigabit" in the name. See below for example:



6. Hit OK
7. Start up the Linux Virtual Machine again
8. Open a terminal and type:
   sudo ifconfig
9. You'll now notice there is a new entry: **eth1**. This is the new Ethernet port you created in the virtual machine, and it is bridged to your wired Ethernet port. This is the port we want to make a static IP address.
10. To set eth1 to a static IP address, open /etc/network/interfaces
    sudo gedit /etc/network/interfaces
11. Add the following lines to set eth1 to 192.168.0.100:
    auto eth1
    iface eth1 inet static
    address 192.168.0.100
    netmask 255.255.255.0
    gateway 192.168.0.1
12. Save the file
13. Now restart the eth1 interface by typing the two commands:
    sudo ifdown eth1
    sudo ifup eth1
14. Type **ifconfig** and you should see that eth1 now has the proper IP address:

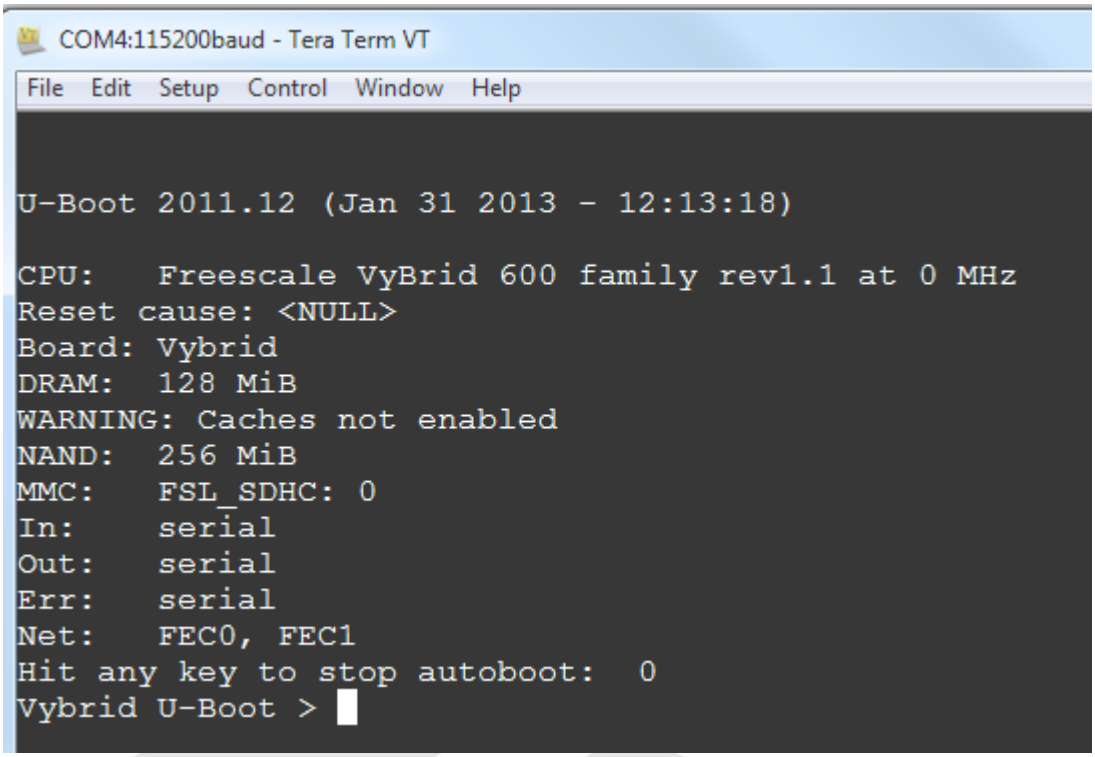
```
anthony@anthony-vybrid:~$ sudo ifdown eth1
anthony@anthony-vybrid:~$ sudo ifup eth1
anthony@anthony-vybrid:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:8a:71:65
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe8a:7165/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:76 errors:0 dropped:0 overruns:0 frame:0
          TX packets:203 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:26783 (26.7 KB)  TX bytes:25820 (25.8 KB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:76:c0:04
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe76:c004/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:146 errors:0 dropped:0 overruns:0 frame:0
          TX packets:166 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24729 (24.7 KB)  TX bytes:25989 (25.9 KB)

lo        Link encap:Local Loopback
```

15. You should also still be able to use your wireless Internet in the virtual machine (represented in the Virtual Machine by **eth0**) to still connect to normal Internet addresses. Verify this in Firefox.

4.4.5.3 Target Setup

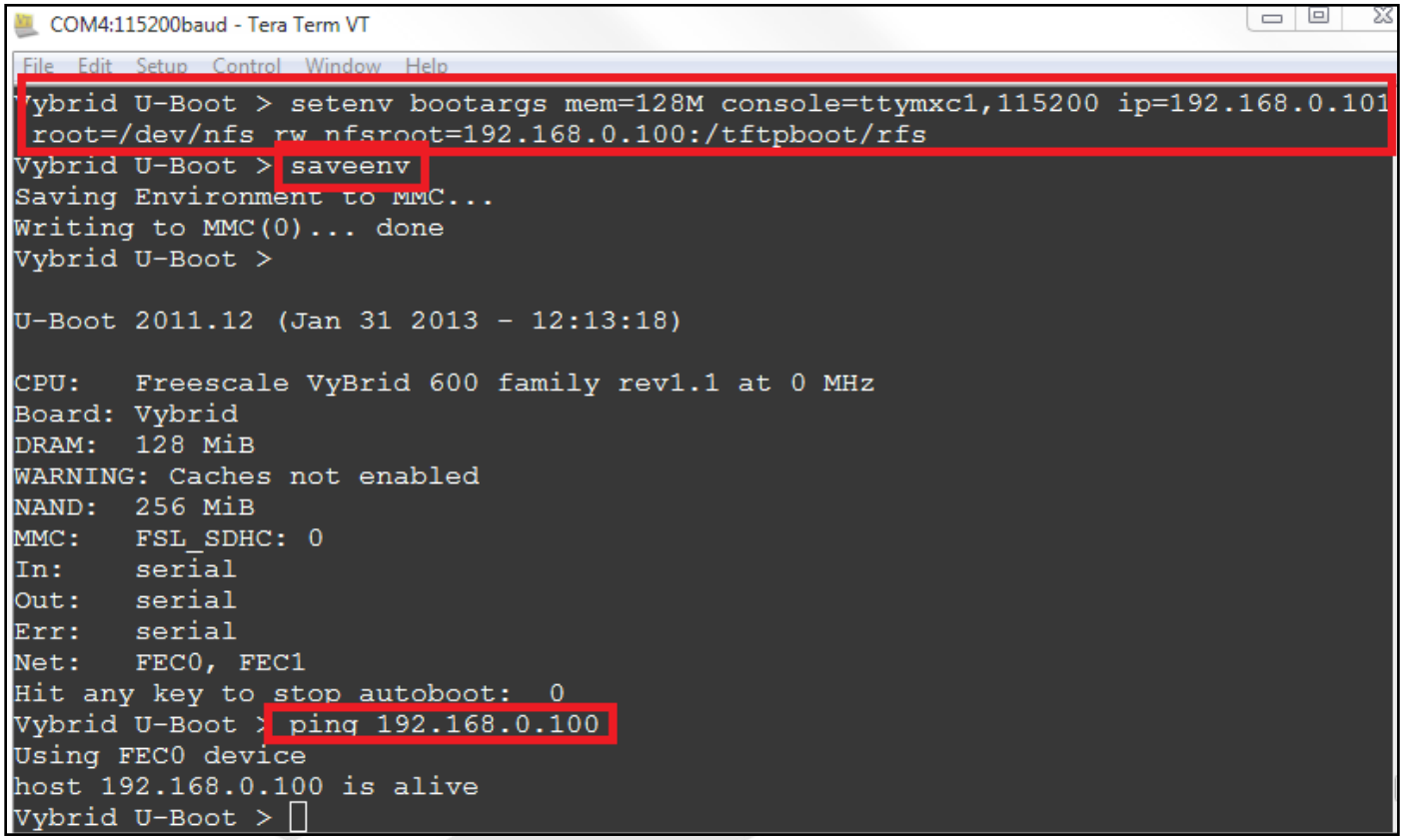
Finally we need to setup the network IP address for the Vybrid tower board in U-Boot.

1. Make sure your board is booting off the SD card as prepared earlier, and U-Boot starts booting.
2. Hit a key to stop the U-Boot from continuing



3. We need to change some default U-Boot settings
    a. IP Address of the board:
    > setenv ipaddr 192.168.0.101
    b. IP address of the "server" we are downloading the kernel from (it is the Host IP address)
    > setenv serverip 192.168.0.100
    c. The boot command U-Boot runs when it starts up. We want to tell it to download the kernel via TFTP and them start the kernel with the **bootm** command
    > setenv bootcmd tftp\;bootm
    d. Set the image to download (from /tftpboot which will be setup later):
    > setenv bootfile uImage
    e. The boot arguments passed to Linux to tell it where the root file system is located, both the server and location.
    > setenv bootargs mem=128M console=ttymxc1,115200 ip=192.168.0.101 root=/dev/nfs rw nfsroot=192.168.0.100:/tftpboot/rfs

4. Save the new settings
    > saveenv
5. Reboot the board and hit a key to pause booting again
6. Type **print** to see all the environmental variables and make sure they saved properly
7. With the Ethernet cable connected, see if you can ping your Linux host on the virtual machine with:
    ping 192.168.0.100



Note: You will not be able to ping the Vybrid board running U-Boot from the Linux side, as U-Boot does not respond to 'ping' command. The Tower board will respond to a 'ping' command only after Linux boots.

4.4.6 TFTP and NFS Configuration

Now configure the Trivial File Transfer Protocol (TFTP) server and Networked File System (NFS) server. This is how U-Boot will download (via TFTP) the Linux kernel, and then the kernel will mount (via NFS) its root file system on the computer hard drive.

4.4.6.1 TFTP Setup

Next setup the TFTP server. The following commands show that we are logged in as root (#). If you are not root ($) then precede each instruction with "sudo".

1. Edit /etc/xinetd.conf
   # gedit /etc/xinetd.conf
2. Add and save the following lines in the file

```
service tftp
{
 socket_type = dgram
```

```
    protocol = udp
    wait = yes
    user = root
    server = /usr/sbin/in.tftpd
    server_args = -s /tftpboot
    disable = no
}
```

3. Back in the terminal, make a directory for TFTP
   # mkdir /tftpboot
4. Copy the Linux uImage into the TFTP directory and have it named **uImage**
   # cp /home/<login>/timesys/twr_vf600/uImage-3.0-ts-armv7l /tftpboot/uImage
5. Restart the xinetd service
   # service xinetd restart
6. Test that TFTP is working
   tftp localhost
   tftp> get uImage
   Received 1456898 bytes in 0.4 seconds
   tftp> quit

4.4.6.2 NFS Setup

Finally setup the NFS. Again, the following commands show that we are logged in as root (#). If you are not root ($) then precede each instruction with "sudo".

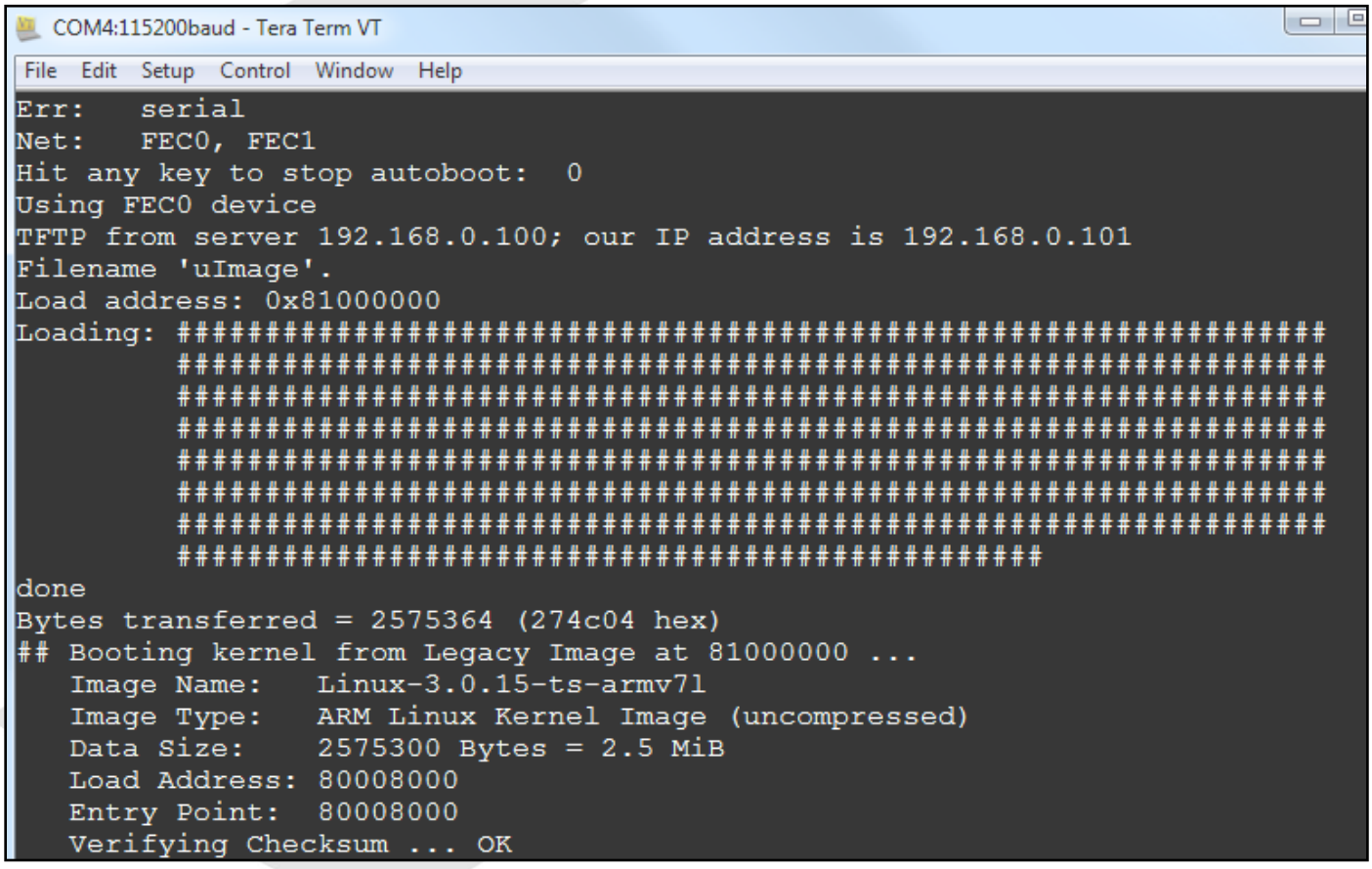
1. Copy root file system and extract. This example moves the RFS from the default directory to the /tftpboot folder. The RFS directory could be located elsewhere, but this location was chosen to make the full path to RFS a little more straightforward. This path is used to setup U-Boot.
   # mkdir /tftpboot/rfs
   # cd /tftpboot/rfs
   # tar -xvf /home/<login>/timesys/twr_vf600/rfs/rootfs.tar.gz
2. Open the /etc/exports file
   # gedit /etc/exports
3. Add the following line and save the file:
   /tftpboot/rfs *(rw,no_root_squash)

   Note: you can use /tftpboot/rfs 192.168.0.101(rw,no_root_squash) instead, but using (*) does not lock access to a single IP address. 192.168.0.101 would be the Tower board U-Boot variable ipaddr.
4. Restart NFS
   # service portmap stop
   # service nfs-kernel-server stop
   # service portmap start
   # service nfs-kernel-server start

```
root@r39626-VirtualBox:/tftpboot/rfs# service portmap stop
portmap stop/waiting
root@r39626-VirtualBox:/tftpboot/rfs# service nfs-kernel-server stop
 * Stopping NFS kernel daemon                                              [ OK ]
 * Unexporting directories for NFS kernel daemon...                        [ OK ]
root@r39626-VirtualBox:/tftpboot/rfs# service portmap start
portmap start/running, process 8072
root@r39626-VirtualBox:/tftpboot/rfs# service nfs-kernel-server start
 * Exporting directories for NFS kernel daemon...
exportfs: /etc/exports [2]: Neither 'subtree_check' or 'no_subtree_check' specified for exp
ort "*:/tftpboot/rfs".
  Assuming default behaviour ('no_subtree_check').
  NOTE: this default has changed since nfs-utils version 1.0.x

                                                                           [ OK ]
 * Starting NFS kernel daemon                                             [ OK ]
```

4.4.7 Boot Linux over NFS

After the IP address, TFTP, and NFS are all setup, the Vybrid tower board can now boot Linux. Restart the tower board, and this time do not stop U-Boot from booting. You should see the kernel download, and Linux boot:



Once Linux has booted, you can try creating an empty file in the /home directory with the **touch** command:

```
Auto-mount of [/media/mmcblk0p2] successful
[ OK ]
Setting timezone and system clock: [OK]
Starting system logging.
Configuring network interfaces: ifdown: interface lo not configured
ip: RTNETLINK answers: File exists
failed
Starting dropbear sshd: OK


BusyBox v1.20.2 (2013-01-31 12:08:36 EST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# uname -a
Linux twr_vf600 3.0.15-ts-armv7l #1 Thu Jan 31 12:03:45 EST 2013 armv7l GNU/Linu
x
# cd home
# touch vybrid_file
# ls
vybrid_file
# pwd
/home
#
```

And then in the Linux host environment, go to the RFS directory in /tftpboot/rfs and see that file that was created in /home:

```
anthony@anthony-vybrid: /tftpboot/rfs/home
anthony@anthony-vybrid:~$ cd /tftpboot/rfs
anthony@anthony-vybrid:/tftpboot/rfs$ ls
bin     dev   home  linuxrc  mnt   proc  run   sys  usr
config  etc   lib   media    opt   root  sbin  tmp  var
anthony@anthony-vybrid:/tftpboot/rfs$ cd home
anthony@anthony-vybrid:/tftpboot/rfs/home$ ls
vybrid_file
anthony@anthony-vybrid:/tftpboot/rfs/home$
```

4.5 Build the SDK on Your Computer using the Desktop Factory

The LinuxLink allows configuration and build “in the cloud”, but the SDK can also be built locally on the host machine with the Desktop Factory. Local builds are quite a bit faster. This is for customization of the kernel, build settings, or adding your own packages to the root file system (RFS).

Download the Desktop Factory Installer file from the same starting point used for the BSP/SDK previously. This file will always be named **twr_vf600-factory-installer.sh**. Follow the instructions on the Timesys page to change permissions and run the installer.

After the Desktop Factory has been installed, please refer to the Desktop Factory Getting Started Guide:

The following steps are in the guide and should be followed:

1. Go to the factory installation directory: **cd /home/<login>/timesys/twr_vf600/factory-<date>**
2. Check to make sure you have the required software installed:
   $ make checksystem

```
r39626@r39626-VirtualBox:~/timesys/twr_vf600$ cd factory-20121126/
r39626@r39626-VirtualBox:~/timesys/twr_vf600/factory-20121126$ make checksystem
-- Reading configuration and build instructions -- 1357773133 [Wed, 09 Jan 2013 17:12:13 -0
600]
Please wait, this will take some time...
-- Checking system for necessary host packages -- 1357773135 [Wed, 09 Jan 2013 17:12:15 -06
00]
SUCCESS: All necessary host tools are installed.
r39626@r39626-VirtualBox:~/timesys/twr_vf600/factory-20121126$
```

3. Check platform configuration
   $ make update
4. At this point the factory installation is done and the kernel can be recompiled on your PC instead of in the cloud. To configure the kernel, navigate to ~/timesys/twr_vf600/kernel-source/linux-3.0, and type 'make menuconfig'.
5. To cross-compile the kernel, you need to run "make ARCH=arm CROSS_COMPILE=<path-to-fully-qualified-toolchain> uImage". eg, "make ARCH=arm CROSS_COMPILE=/home/user/timesys/twr_vf600/toolchain/bin/armv7l-timesys-linux-gnueabi- uImage"

5 DS-5 and MQX in Linux

This section will discuss the installation and use of DS-5 and MQX.

ARM DS-5 can be used to build and debug a Linux application and an MQX application in one IDE. DS-5 is an Eclipse based tool which allows for third party plug-ins, and can run in the Windows environment or on Linux. For first time users of DS-5, please refer to the DS-5 Quick Start Guide for Vybrid available at www.freescale.com/twr-vf65gs10.

MQX usually lives on the Windows operating system. Timesys has made available a shell script that updates the MQX projects and paths and allows MQX to exist in Linux. Alternatively one could use Wine (www.winehq.com) under Linux to emulate a Windows environment to install MQX, but the Timesys method is preferred.

Install DS-5 (Linux)

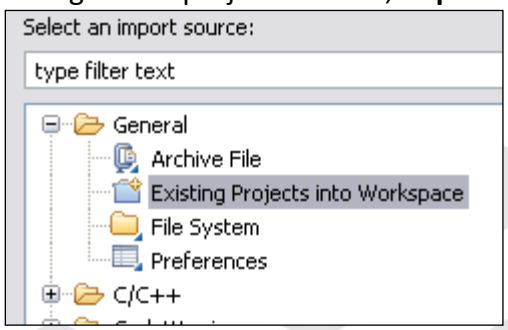
In your Linux environment, Download DS-5 (v5.14.1or later) from ARM's homepage
http://www.arm.com/products/tools/software-tools/ds-5/ds-5-downloads.php

1. Unzip the installer: **tar –xvf DS500-BN-00004-r5p0-13rel0.tgz**
2. Launch the installer: **./install_x86_32.sh**
3. Install in default directory /usr/local/DS-5
4. Enable option to add icon
5. After installation, install the Timestorm plugin downloaded from the LinuxLink website: **./tsplugins**
6. To launch DS-5, click on the Ubuntu button and search for ds-5
7. Create Workspace DS-5_Workspace in /home/<login>/ during installation
8. Install license (DS-5->Help->ARM License Manager) with the license you received when you bought the board.

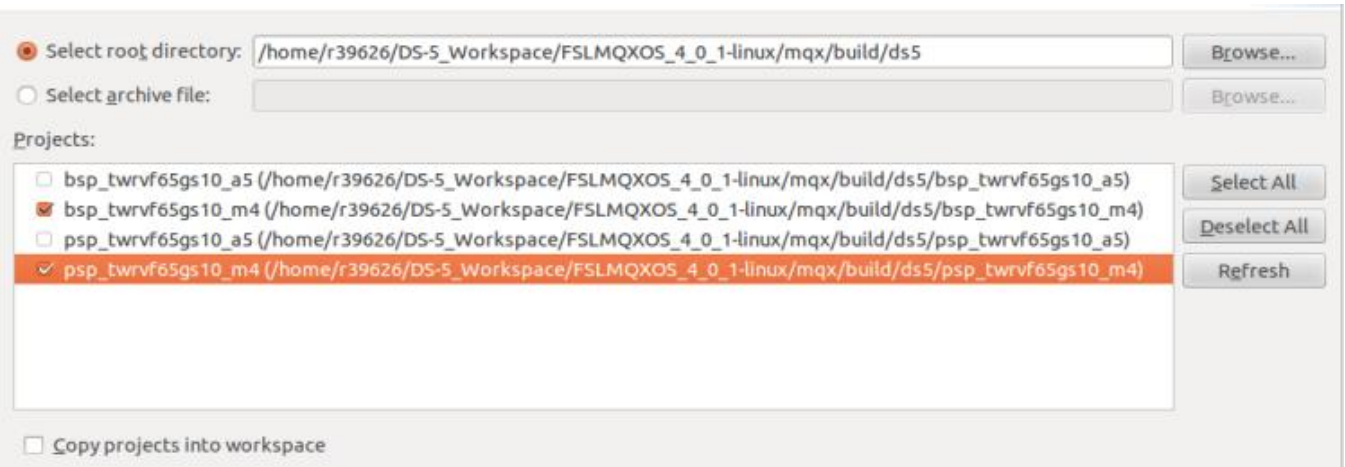
Install MQX (Linux)

Download the MQX Linux installer from the Timesys website.

1. Make the script executable:
   chmod +x MQX-<version>.sh
2. Run the script
   ./MQX-<version>.sh
3. You now have a full MQX installation in your Linux environment.
4. Now compile some of the MQX libraries
5. Open DS-5 in Linux and point the workspace to the root directory where you installed MQX.
6. Import the bsp, psp, and mcc projects into your workspace.
   a. Right click project window, **Import…->General->Existing Projects** into Workspace.



   b. Navigate to <mqx_install_dir>/mqx/build/ds5, click OK, and import the following projects
      i. **bsp_twrvf65gs10_m4**
      ii. **psp_twrvf65gs10_m4**

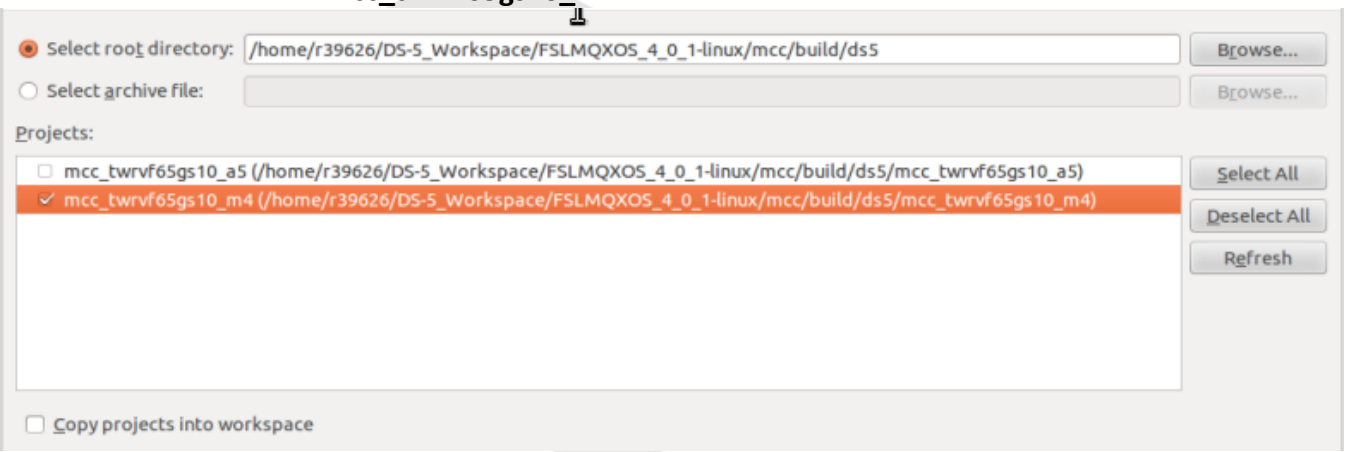


Note: DO NOT select 'Copy projects into workspace' box

c. Click **Finish** to add those projects to the workspace

d. Next, navigate to mcc/build/ds5, click **OK**, and import the following project

i. mcc_twrvf65gs10_m4



Note: DO NOT select 'Copy projects into workspace' box

7. Click **Finish** to add those projects to the workspace
8. Build **bsp**, **psp**, then **mcc** projects, in that order. Building is accomplished by clicking on the project and hitting the build button  , or right click->Build Project.
9. There is a pingpong demo that comes with MQX to demonstrate Multi-Core Communication (MCC). These exist in the <mqx>/mcc/examples directory. Also, the Vybrid out of box demo uses MCC. Please refer to the Lab Guide located at www.freescale.com/TWR-VF65GS10 for more information on the out of box demo code.

6 Setting Up GDB Linux and MQX Debug in DS-5

The following section will describe the steps involved to setup a GDB (GNU Debug) session using DS-5 and the Timestorm toolchain. GDB debugging is typically accomplished over Ethernet (TCP/IP) and allows Linux application debugging. Simultaneously, you may also set up a CMSIS-DAP connection to your MQX application at the same, allowing for cross-triggered debugging between Linux and MQX.

6.1 Linux

6.1.1 Configure Target Hardware for Linux Debug

1. Click on **Hardware Targets** icon in **DS-5**  .
2. Select '**New'**
3. Enter a **name**
4. Click **SCP**
5. Enter the **IP address** displayed on the LCD display when the target booted.
6. Enter **root** for the **user name**
7. Enter the **password** you gave to root
    - Note: On the target board, type '**passwd'** at the Linux prompt (#) to setup your root password
8. Enter **/** for the Destination Directory
9. Click the **Execute** tab
10. Click **SSH**
11. Click **Check Link**
12. In the **Raw Log** tab you will see messages that end with **Target Check – Passed**

6.1.2 Build the Linux Application

1. **Right click** in the blank area of the **Project view** (the area on the left)
2. **Click Import**...
3. Double click **General** to expand it (it may already be expanded)
4. Click **Existing Projects** into **Workspace**
5. Click **Next**
6. Click **Browse** and navigate to the Linux application project, for example **accelerometer-cgi**
7. Click **OK**
8. IMPORTANT: Make **sure Copy into Workspace** is **not checked**
9. Click **Finish**
10. **IMPORTANT**: After it is imported, the project will be called accelerometer-cgi
11. **Right click** on the project
12. Click **Properties**

13. Double **click C/C++ Build**
14. Click **Settings**
15. In the Cross Toolchain tab select **SDK twr_vf600 GCC**... in the Cross Toolchain: dropdown
16. Click **OK**
17. **Right click** on the **project**
18. Click **Build Project**

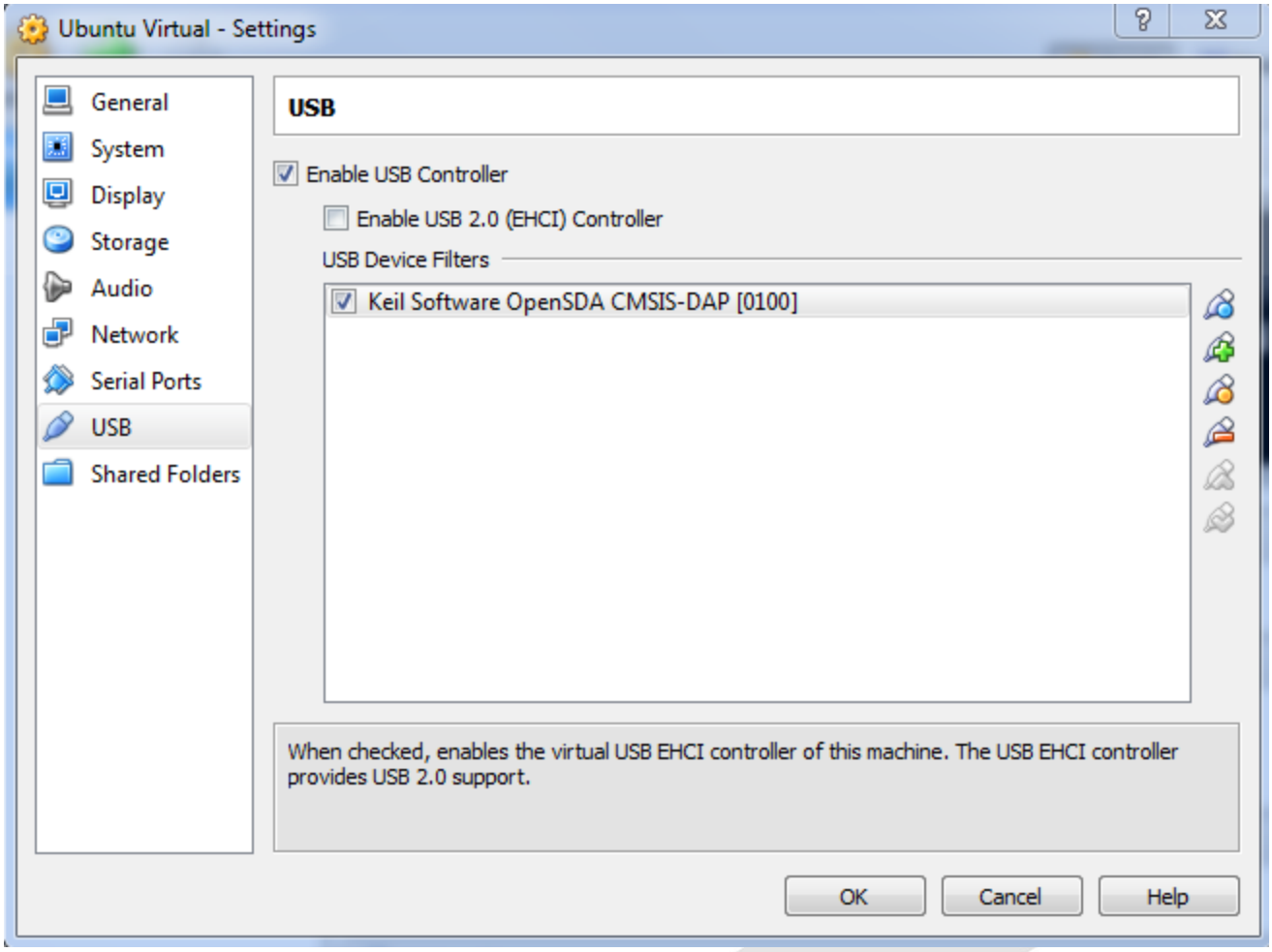
6.1.3 Debugging

1. Click **Window**
2. Click **Open Perspective**
3. Click **C/C++**
    - Note: icons on top right show which perspective you are in. 
4. Right click on **accelerometer-cgi**
5. Click **Debug As**
6. Click **Debug Configurations...**
7. Double click **TimeStorm C/C++ Remote**
    - Note that the fields will be pre-populated
8. Select the **Target tab**
9. Select the connection created earlier in the **Hardware Target** dropdown
10. Select **Debug**
11. You will be prompted to open the **Debug** perspective. Click **Yes**
12. *There will be a warning message **Can't find debug.dl**. Click the green arrow to bypass. This will happen twice.*
13. The debugger will stop at **main**

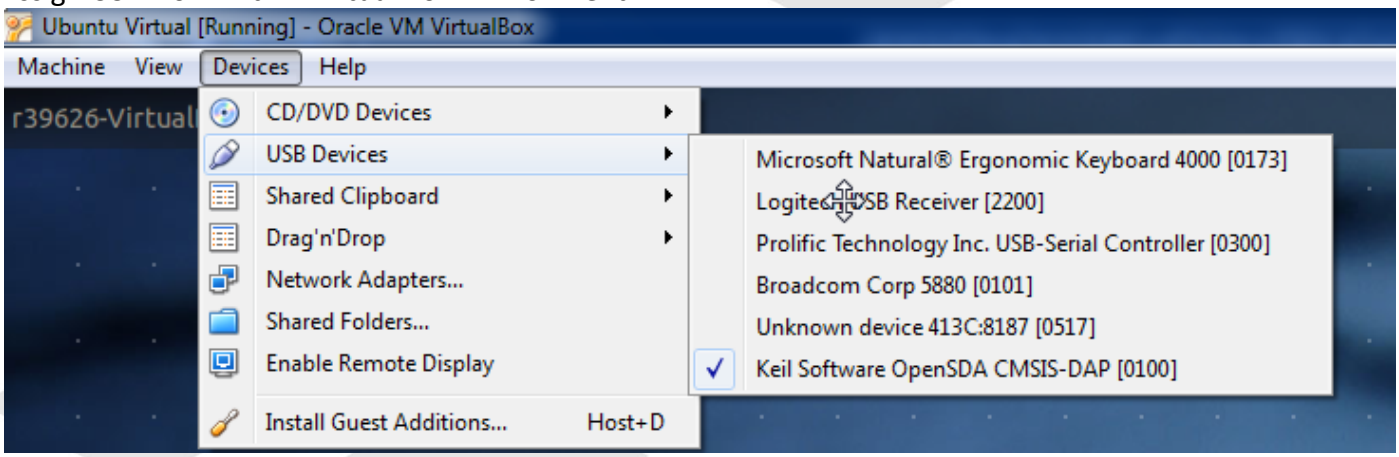
6.2 MQX Debug

Make sure that your OpenSDA firmware has been updated to the CMSIS-DAP firmware prior to DS-5 debugging. The instructions can be found at www.freescale.com/twr-vf65gs10. Also, while using the virtual Linux environment, be sure to assign USB ownership for the tower board. If this step is skipped, then you will not be able to connect to the Tower board from within your Linux environment because Windows will maintain ownership of the USB connection. This can be done in the VirtualBox settings or in the virtual environment once it has been launched.

Virtual Box Settings (click icon with green plus sign to add, make sure board is plugged in):



Assign USB from within Virtual Box Environment:

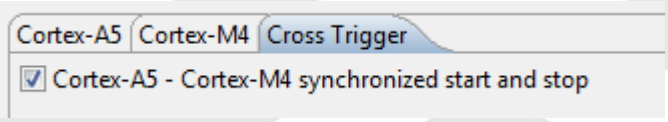


Keep in mind that if the board is unplugged and plugged back in, you will have to reassign it each time if it is not setup in the Virtual Box Settings.

For Cross Triggering Interrupts (CTI) which allows both cores to be halted with a single breakpoint, go to Debug Settings with DS-5.

1. Right click on the previously imported MQX project, for example **accelerometer_example_twrvf65gs10_m4**
2. Click **Debug As**
3. Click **Debug Configurations...**
4. Double click **DS-5 Debugger**
5. Enter a name
6. Select the **Connection** tab
7. In the **Filter** box enter **VF6xx**
8. Expand **Vf6xx**
9. Expand **Bare Metal Debug**
10. Select **Debug Cortex-M4 via CMSIS-DAP**
11. Click **DTSL Options Edit...**
12. Check **Enable Cortex-M4 clock**
    - Optionally enable CTI to halt both cores upon a breakpoint
13. Click **OK**
14. Select the **Files** tab
15. Click **Workspace**...
16. Expand **accelerometer_example_twrvf65gs10_m4**
17. Expand **Int Ram Debug**
18. Select **accelerometer_example_twrvf65gs10_m4.axf**
19. Click **OK**
20. Select the **Debugger tab**
21. Click **Debug from symbol**
22. Enter **Main_Task** in the box next to **Debug from symbol**
23. Click **Debug**
24. You will be prompted to open the DS-5 Debug perspective. Click **Yes**
25. The debugger will stop at the **Main_Task** function

To setup Cross Triggered Interrupts, which allows synchronized start/stop of both cores, go to **Debug Configurations-> DTSL Options->Cross Trigger tab within DS-5:**



Debug Configurations can be found by clicking the down arrow next to the green bug icon