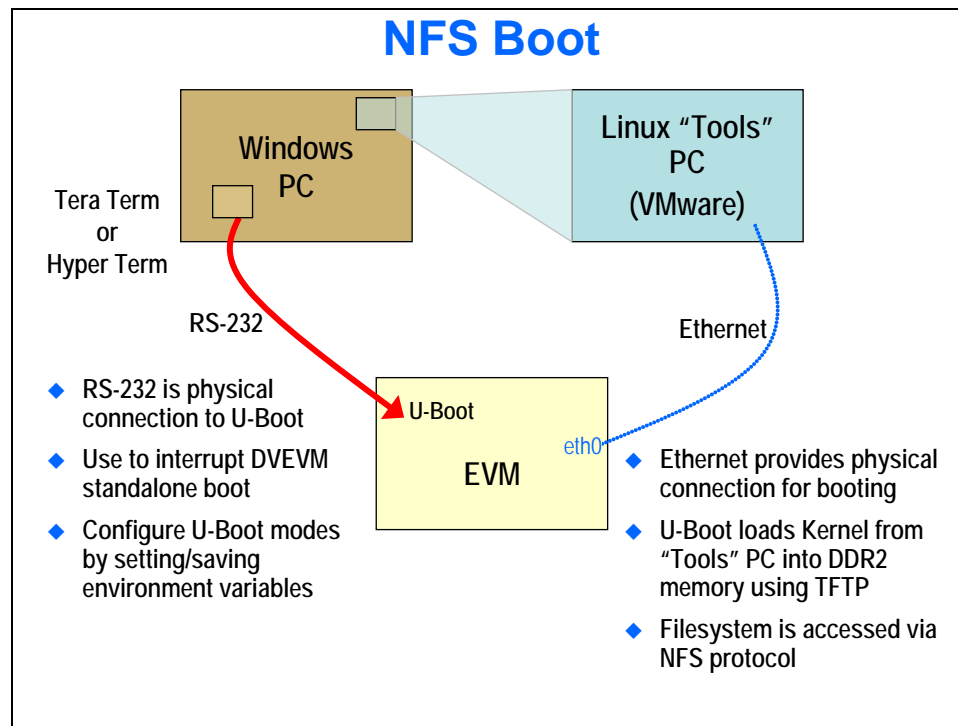


# Experimenting with Linux and U-Boot



Most development for a Linux based target devices, such as the ARM CPU's on the OMAP/Sitara/DaVinci, is done on Linux-based host machines. Developers with Linux PCs can therefore work directly in this environment, but authors using Windows based PCs need either to obtain a new PC running Linux, or employ software that can simulate the Linux environment on top of Windows. In this workshop, VMware is used to create a 'virtual machine' on a windows PC, inside which the Ubuntu operating system can run. In this portion of the lab, the steps to configure Ubuntu on VMware will be implemented. In this lab, the following steps will be taken to set up the software development environment:

## Chapter Outline

<b>Experimenting with Linux and U-Boot .....</b>	<b>1-1</b>
<i>Lab01a – Start/Configure VMware and Ubuntu Linux.....</i>	<i>1-2</i>
<i>Lab01b – Install Workshop Lab Files (for your board).....</i>	<i>1-5</i>
<i>Lab01c – Image SD/MMC card (to boot EVM).....</i>	<i>1-8</i>
<i>Lab01d – Talking to the EVM.....</i>	<i>1-11</i>
<i>Lab01e – Verify Networking and Record IP Addresses.....</i>	<i>1-13</i>
<i>Lab01f – Configure U-Boot and Boot the EVM.....</i>	<i>1-15</i>

## Lab01a – Start/Configure VMware and Ubuntu Linux

### VMware



#### 1. Launch VMware.

On the Windows desktop, **double click** the **VMware** icon.

#### 2. Open the TTO workshop VMware image.

In the VMware Workstation window **Home tab**,



Click on the Open Existing VM or Team Icon

Open the VMware image file (the name you see might be similar but not exact):

`C:\vm_images\tto_vm_child_image_(v3.08)\tto_vm_child_image.vmx`

#### Notes:

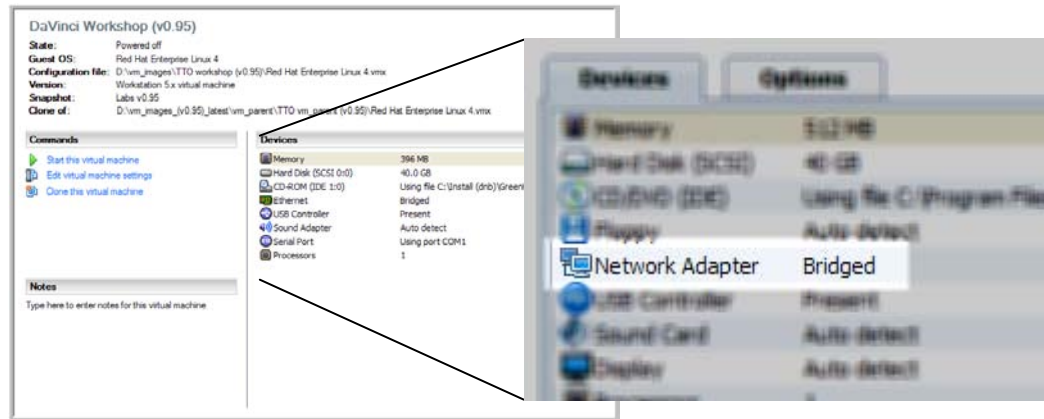
- If your instructor has already started VMware for you, then you may skip this step.
- VM image version v3.08 was current at the time of this writing.
- In USA classrooms, the VMware image is broken into two parts:
  1. *Child* image (~1.8GB) (C:\vm\_images\tto\_vm\_child\_image\_(v3.08)\tto\_vm\_child\_image.vmx)
  2. *Parent* image (~16GB) (E:\vm\_parent\TTO\_vm\_parent\_(v3.03)

The child image, specified in this step, depends upon the parent in order to work. Breaking the image into two parts allows us to re-image the C:\ drive being required to reload the entire 18GB for each class.

### 3. Verify the Linux networking options are set to ‘bridged’ mode.

This option tells VMware to access the network and obtain its own IP address (other choices involve the Windows PC acting as a router). If not set to ‘bridged’

If you have opened VMware application and the TTO image, you should see the Ethernet setting in the middle of the VMware screen as shown here:



If you happened to get a little ahead of our instructions and already started the VMware image (which we do in step 4), the easiest way to see this is in the status bar. Just hover over the Ethernet board icon and read the popup message:




**Note:** If you are using the VMware player, this information is easily found via the top toolbar. In USA classrooms, we use the full version of VMware, though, as opposed to the limited Player version.

## Note:

*Your instructor may already have booted your Ubuntu image (in VMware) and left it hibernated (paused). If so, step 4 might act slightly different.*

### 4. Start Ubuntu Linux.

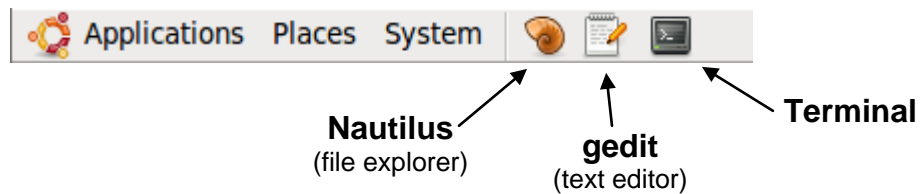
**Click** on the green ‘Play’ arrow  in the icon bar near the top of the VMware window. (Another way to start the Linux session is to select **Start the Virtual Machine** in the **Commands** area). Wait for the boot process to complete (which may take between 2-5 minutes), as indicated by the appearance of the **Log On dialog box**. (If using VM Player, the image is automatically started when opening the VMware Image file.)

**Ubuntu will automatically log you into Linux with a user account.** At this point, you will simply see a blank desktop and you can move on to the next step.

FYI – Ubuntu automatically logged you into the following account – no login required by you at this time:

<b>Ubuntu Userid:</b>	<b>user</b>
<b>Password:</b>	<i>none required</i>

### 5. Open a terminal window.



The easiest way to open the terminal is to click it’s icon on the panel toolbar. You can also find it on the “Applications” menu, but we’ve placed icons to the three most-used tools onto the toolbar panel.

## Lab01b – Install Workshop Lab Files (for your board)

We have installed the appropriate software for your EVM board.

That is, we have worked thru the Getting Started Guides (GSG) for each of the boards (OMAP3530 and AM3517) into the same VMware image, because they both use the same DVSDK/SDK (software development kit) and version of community Linux.

Since the DM6446 uses a different DVSDK, we chose to install its software libraries (and MontaVista Linux) into a separate VMware image. (See v2.10 of *lab materials/files* for this target.)

In this part, you will install the workshop labs/solutions files per the board you have chosen to work on during class. Additionally, we will configure/verify a couple of environment settings.

### Installing Workshop *Labs* and *Solutions* Files

#### 6. Verify the `shared` folder is enabled.

Let's try simply listing the files in the shared folder. If there aren't any files, we may need to enable this VMware feature.

```
ls -l /mnt/hgfs/shared
```

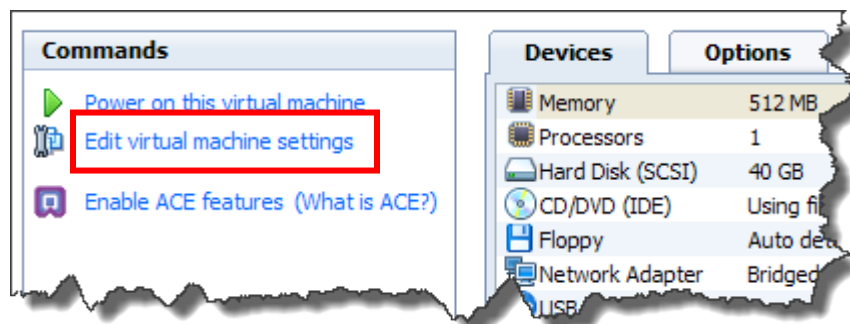
*If this doesn't work*, shared folders are not enabled. Continue with the next step to enable shared folders.

#### 7. If needed, enable shared folders.

If VMware Workstation is running (and it probably is, at this point), go to “options” view by clicking on the **Options** toolbar button:



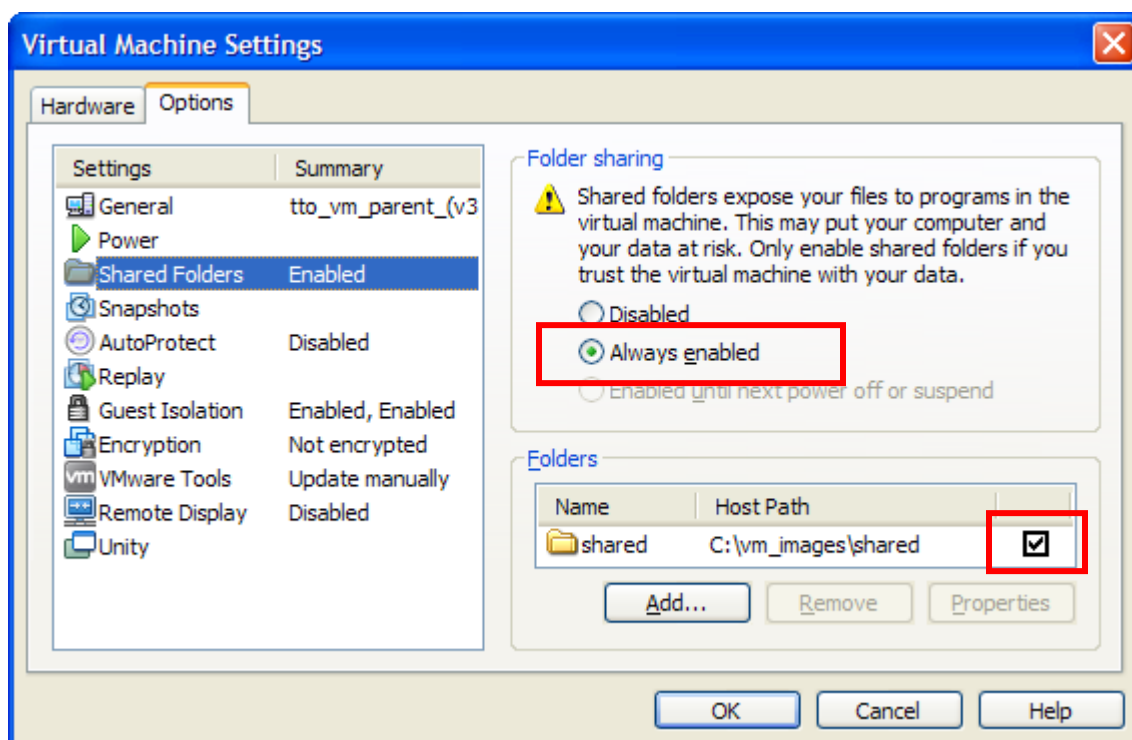
Click on **Edit Virtual machine settings**:



And then ...

When finished enabling shared folders, simply click the “Console View” button in VMware to get back to the command line.

Make sure that **Shared Folders** are *Always enabled*:



## 8. Copy lab files from Windows/VMware shared directory.

To keep things simple, for the OMAP3530 and AM3517 VMware image, everything but the lab files have been installed. Rather than putting lab files for both target boards in the user folder, we have provided you two tar files.

Device  
Specific

```
cd /home/user
cp /mnt/hgfs/shared/TTO_Linux_SOC_Workshop_labs_omap35_v3.xx.tar.gz .
```

### Options:

- For the AM3517 choose: `TTO_Linux_SOC_Workshop_labs_am3517_v3.xx.tar.gz`
- Rather than seeing a file with `v3.xx`, choose the latest revision available; e.g., `v3.08`.

## 9. Untar the lab files into the `/home/user` folder.

In the steps below, make sure you use the file you copied in the previous step.

```
cd /home/user
tar -xzf TTO_Linux_SOC_Workshop_labs_omap35_v3.xx.tar.gz
```

After unzipping, you should have two new folders in your `/home/user` folder. If not, please consult with your instructor.

```
/home/user/labs
/home/user/solutions
```


 Device  
Specific
**10. Verify you have installed the correct files for your EVM platform.**

*(You can skip this step if you are following the DM6446 labs.)*

Check that the readme file exists in your new labs (and/or solutions) folder. We use the readme file to confirm the platform supported – along with the workshop labs/solutions version number.

`/home/user/labs`

`Readme_omap35_labs_v3.xx.txt`

or `Readme_am3517_labs_v3.xx.txt`

**11. Add symbolic link to `targetfs` directory.** *(You can skip this step if you are doing the DM6446 labs.)*

Finally, we need to add a Linux symbolic link for our `targetfs` directory.

`ln -s /home/user/psp_rebuild_omap3/linux_filesys /home/user/targetfs`

or

`ln -s /home/user/psp_rebuild_am3517/linux_filesys /home/user/targetfs`


 Device  
Specific

This Linux command (small LN) creates a symbolic link, similar in some ways to a Windows shortcut. With this link, we can now refer to the `/home/user/targetfs` directory in our workshop instructions and the correct folders/files will be referenced on each of your systems, no matter which EVM you are using.

This is also the directory we are “exporting” (i.e. network sharing). We already set this up for you in Linux by editing the `/etc/exports` file. This was required because since this is the folder used as the `nfspath` – that is, we will use this folder (via the network) as the *root filesystem* for our EVM.

**Installing kernel modules to the *targetfs*****12. Install the kernel modules and `loadmodules.sh` script to the target filesystem.**

We have conveniently placed the kernel modules and scripts into the lab00 folder. All you need to do is run the install script located in that folder.

`cd /home/user/labs/lab00_install_scripts`

`./install.sh`

This will copy the files contained in this folder over to our workshop directory in the target filesystem (`/home/user/targetfs/opt/workshop`). These files are used to insert kernel modules (DSP/LINK and CMEM) into the kernel at runtime. They are provided for us the SDK, but we create the shell script (and placed the files into this folder) to make them easier to install.

## Lab01c – Image SD/MMC card (to boot EVM)

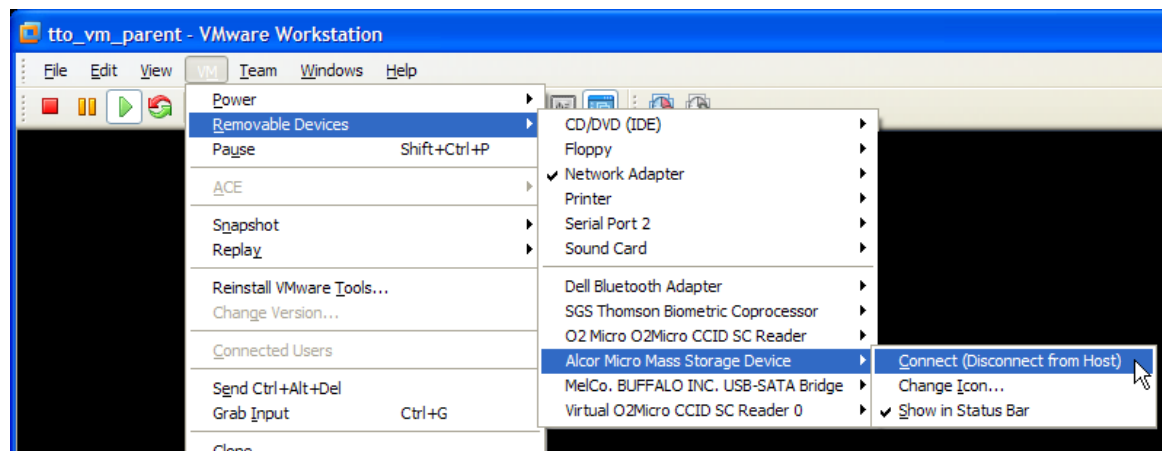
### 13. Plug USB Flash SD/MMC Card reader into a USB port on your computer.

You may see a dialogue box talking about “Removable Devices” – just click OK and continue.

### 14. Connect the SD/MMC flash card reader to the Ubuntu virtual machine.

If USB Flash Card reader is mapped to Windows host, select:

VM → Removable Devices → **<Your Flash Reader>** → Connect...



---

**Note:** Your SD/MMC card reader may show up as a slightly different name, depending upon the brand of reader you are using

---

### 15. Open a terminal window in Ubuntu (if one is not already open).

### 16. Move to the Lab01a directory.

```
cd ~/labs/lab03_build_sd
```



**17. Determine SCSI device node for USB SD card reader**

```
(user@ubuntu) # sudo sg_map -i
```

When prompted for sudo password, (press enter)

You should see a table similar to the following:

```

/dev/sg0 /dev/scd0 NECVMWar VMware IDE CDR10 1.00
/dev/sg1 /dev/sda  VMware,  VMware Virtual S  1.0
→ /dev/sg2 /dev/sdb  USB 2.0   SD/MMC Reader    1.0

```

Depending on the SD/MMC Reader used, it may appear differently, but will likely be the last device on the list.

Write down the Linux device node (i.e. virtual file name) for the card reader:

Your device node: \_\_\_\_\_ (most likely, /dev/sdb)

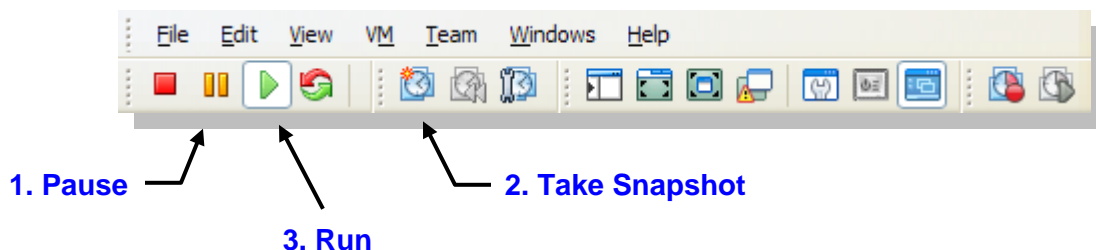
**18. Insert a 2GB SD/MMC card into the USB Flash reader (if not already done).****Caution**

Read the following step and comments very carefully, specifying the wrong /dev/sdx device node could cause permanent damage to your system!

**19. Take a VMware snapshot.** *(Only full version of VMware Workstation supports snapshots.)*

Because this step could erase the wrong drive in your system, let's make a snapshot copy of our virtual hard drive. This can be done many ways, but we suggest this simple 3-step procedure – which uses three different VM toolbar buttons:

1. Pause your Linux VMware PC.
2. Take a snapshot.
3. Un-pause (that is, Run) your Linux VM, again.



## 20. Execute the `build_sd.sh` script.

Run the build script using the device node from step 17 (page 1-9). If prompted for a sudo password, simply press enter (blank password).

```
(user@ubuntu):  sudo SCSI_DEV=/dev/sdb ./build_boot_sd.sh
```

When asked to “confirm”, press “y” and [ENTER].

It should take less than a minute for the script to complete. The script automates these steps:

- Un-mounts partitions (if any) that Ubuntu automatically mounts to the desktop
- Reformats and formats the SD/MMC card for two partitions  
(though we’ll only use one, for now)
- Temporarily mounts new partitions and copies three files onto the 1<sup>st</sup> partition:

MLO	(X-loader – 2 <sup>nd</sup> level bootloader)
u-boot.bin	(uboot – 3 <sup>rd</sup> level Linux bootloader)
uImage	(Linux kernel)

In the next part of the lab, we’ll use the MMC card to boot the EVM.

# Lab01d – Talking to the EVM

## 21. Start TeraTerm.

On the Windows desktop, **double click** on the **TeraTerm** icon.  
The TeraTerm serial configuration file `dvevm.ini`, in the TeraTerm program folder has already been set up with the following necessary configuration states:

```
Bits per Second: 115200
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: None
```

## 22. Insert the SD/MMC card into the EVM.

If you haven't already done so, remove the SD/MMC card you formatted in step 20 from the card reader.

Insert the card in the EVM's SD/MMC card slot

The card should go into the slot "label up" – SD card "pins" down. On new boards, the slot is tight, so you need to make sure and line it up very straight as you slide the card into it.

## 23. Connect RS-232 serial cable.

If not already done, please connect the serial cable. (If unsure how to do this, please ask your instructor (or refer the EVM Quick Start Guide).

Connect RS-232 cable between the EVM and PC RS-232 port

Note: For OMAP3530 EVM, please use the UART1/2 connector.

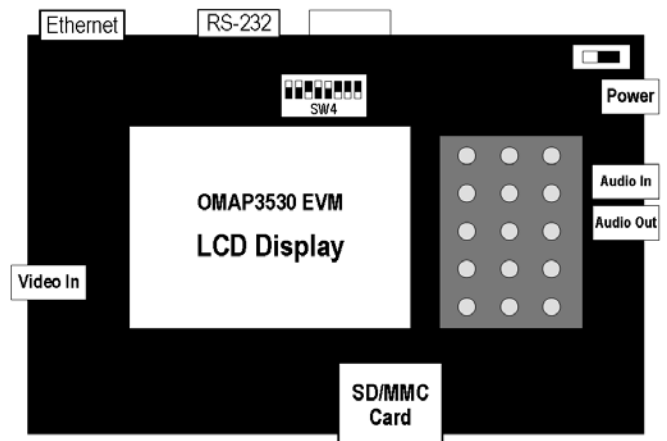
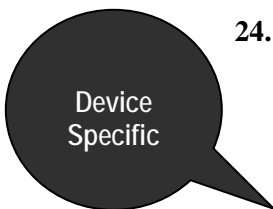
## 24. Verify EVM Hardware Configuration

- Is the EVM powered off?
- Verify the switch settings (for proper booting) of the EVM - where does board find MLO and uboot.bin?

### OMAP3530 EVM switch S4

```
SD/MMC card:      0010 0111
On-board NAND:    xxxx xxxx
```

(AM3517 switch settings continued on next page.)



### AM3517 EVM switch S7

SD/MMC card: 0000 1001

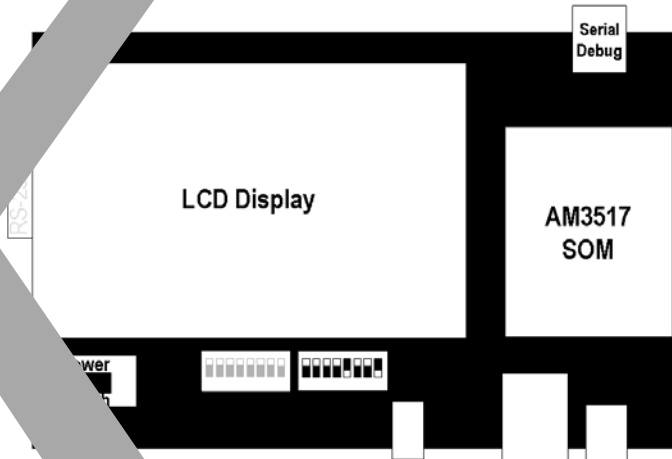
Setting the first and fourth switches on, while the others are off, tells the board to boot using the MMC card.

sw 7-1: on  
sw 7-2: off  
sw 7-3: off  
sw 7-4: on  
sw 7-5: off  
sw 7-6: off  
sw 7-7: off  
sw 7-8: off

These switches modify the boot mode pins on the AM3517, which are used by the ROM loader (1<sup>st</sup> stage) to use the ROM loader (2<sup>nd</sup> stage bootload) and on the first found EVM's onboard NAND flash.

To learn more about the switches (and configuration of the AM3517 board, visit:

[http://processors.wiki.ti.com/index.php/GSG:\\_AM35x\\_EVM\\_Hardware\\_Setup](http://processors.wiki.ti.com/index.php/GSG:_AM35x_EVM_Hardware_Setup)



## 25. Start the EVM – by plugging in the Power cable (or toggling the switch next to power cord, if there is one).

Power on the EVM board and press any key to interrupt U-Boot's boot sequence.

Press any key (to stop Linux from booting)

At this point, the EVM U-Boot terminal prompt (DaVinci EVM#, OMAP3#, AM3517#) should be visible in the TeraTerm session window.

(Troubleshooting Note: if the OMAP35 EVM does not seem to be booting from the SD/MMC card, it's possible that it is ignoring the card and booting from the NAND flash instead. We have erased the NAND memory on the TI classroom boards to prevent this from occurring. You can accomplish this from within UBOOT by executing the 'erase' command (nand erase).

In a few minutes we'll setup U-boot and get the board running ...

# Lab01e – Verify Networking and Record IP Addresses


## Connecting to the Network

### 26. Make sure the Ethernet cable is connected between your EVM and the PC where you're running VMware.

If you're direct connecting the VMware image to the target EVM, then make sure the SD/MMC card you just programmed is inserted into the EVM and then power-on the EVM board (we don't care what it does at this point – that will be handled in the next section).

On the other hand, if you're using a switch or router, simply make sure that the switch is up-and-running and connected to the EVM and PC.

### 27. Record the Windows Ethernet address.

This information will be used to test the Linux Ethernet connection in the next step. In the Windows **system tray** (right side of the Windows task bar) **double click** on the Local Area Connection 2 icon: 

From the **Support tab** of the dialog box that popped up, write the noted values below. Close the window when done recording this value.

**IP Address** \_\_\_\_\_

*Note, If there are two wired LAN icons in the Windows taskbar, you should choose the one with the IP address: 192.168.1.39*

### 28. Determine the Ubuntu Ethernet address.

**You must have the Ethernet cable plugged in to the EVM and the board powered on or you will get an error during this step.** (You should have connected the Ethernet cable in step 26.)

In the terminal window, run **ifconfig** by typing: **/sbin/ifconfig** ↵ and transcribe the IP address below. (Alternatively, we've set our \$PATH statement so that you can just type *ifconfig*.)

**IP Address** \_\_\_\_\_ (Note, it will be called "inet" in the Linux response)

### 29. Test the Linux Ethernet port:

**Ping** the Windows Ethernet port to verify that both are working. In the terminal, type:

```
ping <IP_Address>
```

Where *IP\_address* is the value recorded in step 27 above. The response should look like:

```
[user@localhost ~]$ ping 192.168.1.39
PING 192.168.1.39 (192.168.1.39) 56(84) bytes of data.
64 bytes from 192.168.1.39: icmp_seq=0 ttl=128 time=2.00 ms
64 bytes from 192.168.1.39: icmp_seq=1 ttl=128 time=0.675 ms
64 bytes from 192.168.1.39: icmp_seq=2 ttl=128 time=0.800 ms
```

In Linux, you need to halt the ping command using:

```
<Ctrl> C to halt the pinging
```

These are the IP addresses we plan to use in this workshop:

Windows PC:	192.168.1.39
Ubuntu Linux:	192.168.1.1
EVM target:	<del>192.168.1.41</del> dynamically set

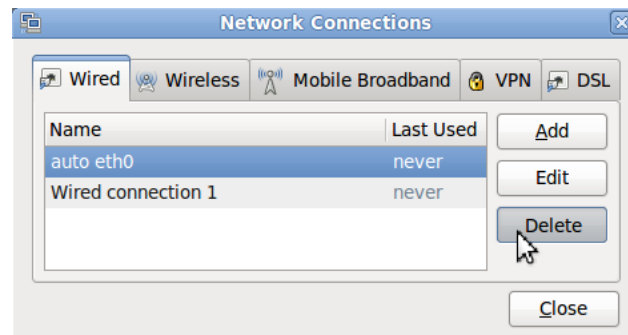
## Network Troubleshooting

*Skip this is everything is working...*

Normally, you won't need to use any of these hints, but here's a few ideas we've run across when we cannot connect the Ubuntu VMware image and Windows.

- Don't waste your time debugging the Ubuntu to EVM connection if you can't even connect between Ubuntu and Windows (which are on the same physical machine).
- Make sure that networking is alive – in most cases, since we're direct connecting from our PC to the EVM, this means the EVM needs to be turned on with Uboot running. Of course, if you're using an Ethernet switch, then it needs to be turned on. Also, you may want to verify that the cable connection is working by observing the networking port lights at both ends.
- Make sure VMware's networking is turned on and in "bridged" mode (step 3).
- If your machine has more than one network card (i.e. NIC), make sure VMware is configured to use the correct one. We configured this in step **Error! Reference source not found., "Error! Reference source not found."**
- Using VMware and Ubuntu, we have occasionally run into a weird side-effect where Linux caches MAC addresses – but VMware changes the MAC addresses to avoid potential conflicts when copying or moving VMware images. This seems to be exacerbated when using Ubuntu's easy "Network Connections" feature. Solving this problem is a three step process:
  - a. Delete any network connections from Ubuntu's "Network Connections".

System > Preferences > Network Connections



- b. Run a small script we've added to our VMware image:

```
fixmac.sh
```
  - c. Reboot Ubuntu Linux.
- Ok, we're running out of ideas. So, make sure that your various Windows or Linux network settings are causing problems – things like firewalls, antivirus programs, proxy settings. We eliminated this on classroom desktops/laptops before class, but if you aren't using our PC's...
  - As a side note, even if you connect the VMware image to internet, you may not be able to access it. (To minimize cabling and hardware, we don't provide this type of access to the VMware images in our classes.) If you must have access, you need to run `sudo dhclient`, to get an IP address from the network. Again, you also must deal with firewalls, proxy's and such.

## Lab01f – Configure U-Boot and Boot the EVM

### 30. Return to Windows and TeraTerm.

Since VMware implements a complete virtual PC when the cursor is within its borders, it is necessary to move the cursor outside the VMware frame so that the use of **Alt + Tab** will invoke the underlying Windows OS and allow control to pass from the VMware application to another Windows program. Then, hold down the Alt key and repeatedly pressing Tab until the **TeraTerm** application is selected.

Release the Alt key to complete the selection.



*Based on where we left things earlier in the lab, you should be at the U-Boot prompt. If this is not the case, power-cycle the board and then stop U-Boot from booting into Linux by hitting any key.*

Device  
Specific

### 31. Run the TeraTerm macro to setup the EVM's U-Boot mode.

From TeraTerm, select **Control | Macro**. From directory **C:\Program Files\TTERMPRO** select the file associated to your board:

DM6446 DVEVM:	tto_uboot_setup.ttl
OMAP3530 EVM:	tto_uboot_setup_3530.ttl
AM3517 EVM:	tto_uboot_setup_3517.ttl

If the macro pauses, simply hit [ENTER] in the terminal window to continue with the questions below:

#### As the macro runs, make the following selections:

- |  |             |
|--|-------------|
| • Use Default NFS Server IP Address: 192.168.1.1           | <u>Y</u> es |
| • Boot Static or Dynamic? [Yes= Dynamic (dhcp), No=Static] | <u>Y</u> es |
| • Root Filesystem from NFS or MMC? (Yes=NFS, No=MMC)       | <u>Y</u> es |
| • Use default NFS path? (/home/user/targetfs)              | <u>Y</u> es |
| • Kernel from TFTP or Flash/MMC? (Yes=TFTP, No=Flash/MMC)  | <u>Y</u> es |
| • For TFTP boot, use the default Kernel image filename?    | <u>Y</u> es |
| • Save bootargs?   | <u>Y</u> es |
| • Boot Linux now? (No, we'll do this manually)             | <u>N</u> o  |

### 32. Test network connection from EVM to Ubuntu VMware image.

This is a good thing to check, since we plan to boot the EVM across the network – that is, we plan to get the root filesystem (and maybe the Linux Kernel) for the EVM from our Ubuntu Linux VM image.

Run the **ping** command from Uboot.:

```
ping 192.168.1.1
```

It should respond with: Connection is alive

### 33. Examine the EVM's Linux environment.

The **printenv** ↵ reports the current state of the U-Boot variables. You should be able to see the changes we made with our interactive TeraTerm script.

### 34. Save the new U-Boot settings.

Changes to the environment must be saved to the Flash to remain active after power-cycling the EVM hardware. This is done automatically by the macro when you answer **Yes** to the 'save bootargs' question.

To manually preserve the bootargs, type:

```
save ↵
```

### 35. Take Home exercise...

Review the macro by opening the file in any text editor. While not commented in detail its code should be easy to understand if one knows the U-Boot options in general.

### 36. Boot / Reboot the EVM.

**Power-cycle the EVM or type boot** ↵ to restart the EVM with the new environment settings. When boot completes (you can watch it in Tera Term – should take a few minutes), **log in as root user**; no password is needed. *Note: if during bootup "kernel panic" is reported, ask the instructor for assistance.*

Your Windows terminal (i.e. Tera Term) is now connected to the "target" Linux running on the EVM's ARM processor.

## Sidebar

It is common practice to log into a host Linux PC as a user (i.e. not as the root user). Conversely, it is also common practice to log into a development board, like the EVM using the root user. In embedded applications, there often only exists a single user (root).



### 37. Verify shared file system between Ubuntu and EVM.

Since any file change to the root directory of our EVM board will be reflected in Ubuntu Linux, let's give it a try by creating a new file (or updating its timestamp) using the Linux "touch" command.

From Tera Term (which is now logged into the EVM board):

```
root@omap3evm:~# cd / moves you to root
root@omap3evm:~# touch putfileatroot.txt create a new empty file at root
```

Now, let's look for this file on the NFS source directory; that is, in the target filesystem on our Ubuntu PC. To do this, list the files of the target filesystem **from the Ubuntu terminal session** (note: be careful to be in the correct window, as there are two that can be easily mistaken for each other) you started earlier:

```
[user@localhost user]# cd /home/user/targetfs
[user@localhost user]# ls -la
```

You should see the *putfileatroot.txt* in your listing, with the current date and time stamp (you could always try the Linux *date* command if you'd like to change it to your time zone). Note, you can see the same directory (and file) from both environments. Similarly, when we create new app's within Ubuntu Linux, if they are created (or copied to) our target filesystem folder, they're immediately available at our NFS mounted EVM target.

### Review

To summarize, the root path of the EVM is set to a path inside the User's home directory. Fill in the box below indicating the path within Ubuntu Linux where the EVM board's root path is associated.

**EVM Board  
"Target"**

**Ubuntu Linux  
"VMware Image"**

/

=

How did this association get made? \_\_\_\_\_

\_\_\_\_\_

What is the advantage using an NFS (networked) mounted filesystem versus using the hard drive (or flash drive) built into the DaVinci board? \_\_\_\_\_

\_\_\_\_\_

## Optional Exercises

### 38. Explore the Tera Term scripts provided for this hands-on lab.

To minimize typing errors – and to generally make it easier for everyone – we used Tera Term to enter the boot arguments into U-Boot. We recommend examining the scripts we used, as this is something you may want to copy and modify for your own needs. Check them out in the folder:

```
C:\Program Files\TTERMPRO
```

### 39. Explore the *Software Developers Guide* that ships with the SDK.

The *software developers guide* provides a series of *how-to* topics, mostly focused upon running the examples that ship with the SDK. We've included it in PDF format in our shared folder.

Review the topics it contains – in fact, if you still have time before the lab is over, you might want to try running one of the examples provided with the SDK.

```
C: \vm_images\shared\pdf_files\OMAP3530_Software_Developers_Guide.pdf
```

or

```
/mnt/hgfs/shared/pdf_files/OMAP3530_Software_Developers_Guide.pdf
```