OMAP-L138/AM1808 SOM-M1 U-Boot Labs

Bootloader Documentation

Logic PD // Products
Published: August 2010
Last Revised: January 2015

Abstract

These informal labs provide an introduction to basic U-Boot commands and usage on the OMAP-L138 SOM-M1 and AM1808 SOM-M1. Criteria

Revision History

REV	EDITOR	DESCRIPTION	APPROVAL	DATE
A	GCJ	Initial release	JCA	08/13/10
B	SMC, JCA	-Section 3.4: Updated TI software locations ; -Section 7.5: Fixed typo in Step 2 that mistakenly called out 0x300000 instead of 0x400000	JCA	05/05/11
C	SWE	-Throughout: Updated template; updated links for new support site; -Added Section 1.1 regarding nomenclature used in document; -Added Section 1.2 regarding files included with document; -Added Section 5.5.1 on how to set up variables for remaining labs; -Section 5.5.2: Removed Step 3 to reset kit and use *print* command; -Section 7.3: Added note to Step 5 to refer to *AN 556* to set MAC address; added Step 6.a to set netmask; -Section 7.4.1: Updated hex value in Step 2; added expected output to Step 3; -Section 7.5: Updated commands throughout	RAH, SO	02/12/14
D	AF	-Throughout: Update screen dumps to reflect U-boot 2010.12 up from 2009.11 -Section 2.3: Indicated that TFTP is only needed when loading or copying over the network. -Section 3.2: Add reference to Linux User Guide for those doing u-boot and UBL updates in the Linux VM. Section 3.5: Removed extra contents that went beyond the scope of U-boot and UBL. -Section 5.6: Added section to provide instructions to return U-Boot to defaults -Section 6.4.1: Added entry in table to show address of MAC address -Section 8: Add instructions to load and boot the example files from SD card.	AF, BSB	1/19/15

Table of Contents

1 Introduction

These informal U-Boot labs for the OMAP-L138 SOM-M1 and AM1808 SOM-M1 provide an introduction to basic U-Boot commands and usage.

NOTE: These labs make use of a null-modem serial cable that is included with the OMAP-L138 EVM Development Kit and both the AM1808 EVM Development Kit and AM1808 eXperimenter Kit. If you are using the OMAP-L138 eXperimenter Kit, you may want to obtain a serial cable before continuing.

1.1 Nomenclature

- This document covers the AM1808 SOM-M1 and OMAP-L138 SOM-M1. Use of “SOM-M1” suggests text that applies to both platforms; information specific to one platform will call out the precise name.
- Use of "development kit" suggests text that applies to the AM1808 EVM Development Kit, AM1808 eXperimenter Kit, OMAP-L138 EVM Development Kit, and OMAP-L138 eXperimenter Kit; information specific to one development kit will call out the precise name.

NOTE: This document assumes that the OMAP-L138 SOM-M1 is being used in the OMAP-L138 EVM Development Kit; the procedures below are not intended for use with the TMS320C6748 SOM-M1.

1.2 OMAP-L138_AM1808_U-Boot_Labs_Software Directory

Accompanying this application note within the *1015906C_OMAP-L138_AM1808_U-Boot_Labs.zip* file is a directory containing static versions of software that are known to work with the instructions herein and with the SOM-M1 hardware. The *OMAP-L138_AM1808_U-Boot_Labs_Software* directory should contain the following sub-directories and files that will be referenced throughout this document:

```
/TFTP-Root
      ramdisk-base.gz
      uImage
ubl_OMAPL138_SPI_MEM.bin
sfh_OMAP-L138.exe
u-boot.bin
```

2 Host PC Setup

2.1 Install Optional Development Tools

There are two programs that you will find useful when going through the following labs: a terminal emulation program and a TFTP server tool.

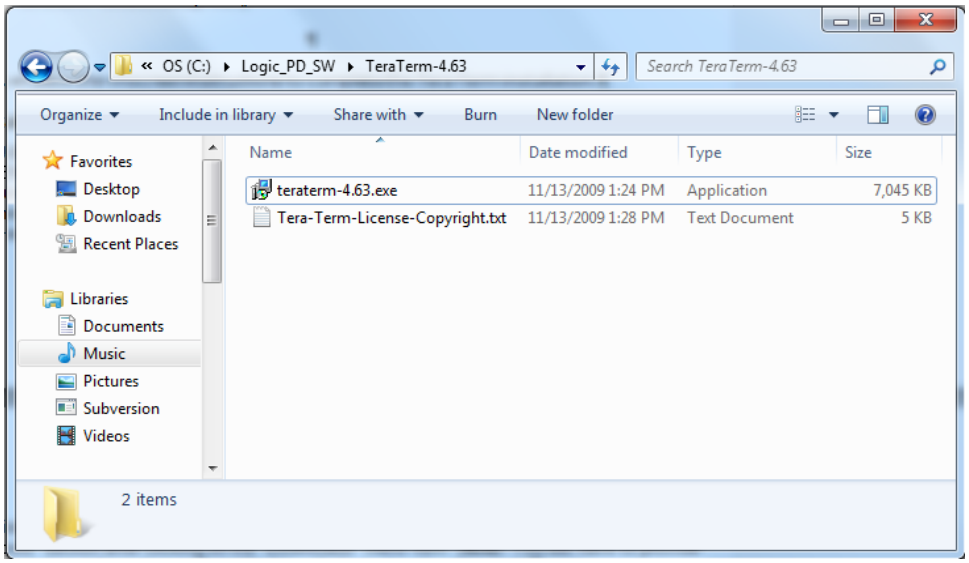
2.2 Install a Terminal Emulation Program

If you don’t have a terminal emulation program already installed on your host PC, Tera Term is an emulator that Logic PD uses for serial communications with the Logic PD development kits. Tera Term has proven to be more reliable and easier to work with than other emulators.

After registering your kit on Logic PD’s support site, navigate to the downloads page for your development kit and follow the steps below to download Tera Term.

1. Scroll down to the *Software Development Tools* heading and click the **Tera Term** link to begin downloading the ZIP file to your host PC. Once complete, extract the files.

2. Enter the Tera Term directory that was just extracted and double-click the *teraterm-xx.exe* setup file.



3. Follow the onscreen setup wizard instructions to complete the Tera Term installation.

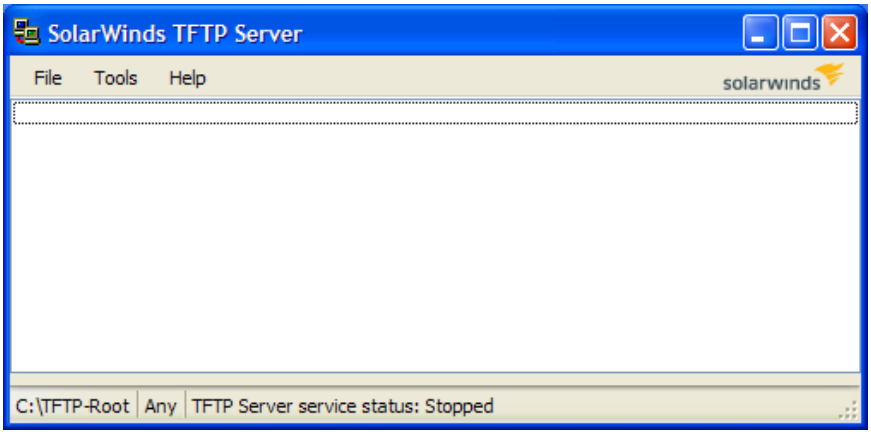
2.3 Install and Configure TFTP Server

If desired, the U-Boot bootloader programmed into the SOM-M1 can transfer files over Ethernet via TFTP. To assist with these transfers, a TFTP server tool needs to be available on your host PC. If you don't already have a TFTP server installed on your host PC, Logic PD recommends but does not support the SolarWinds TFTP server as an easy-to-use tool that is freely available from their website. If booting or copying files over the network is not desired, skip to Section 3.

1. Visit the SolarWinds downloads page.[1] Scroll down to find and download the free TFTP server. **NOTE:** You will have to provide registration information to download the tool.
2. Locate the download on your host PC, extract the ZIP file, and double-click the *SolarWinds-TFTP-Server.exe* file.
3. Follow the on-screen instructions to complete the TFTP server installation. For more detailed installation instructions, please see Petri's *Inside SolarWinds Free TFTP Server: Simple to Use, Easy to Like* article.[2]

[1] http://www.solarwinds.com/downloads/
[2] http://www.petri.co.il/free-tftp-server-from-solarwinds.htm

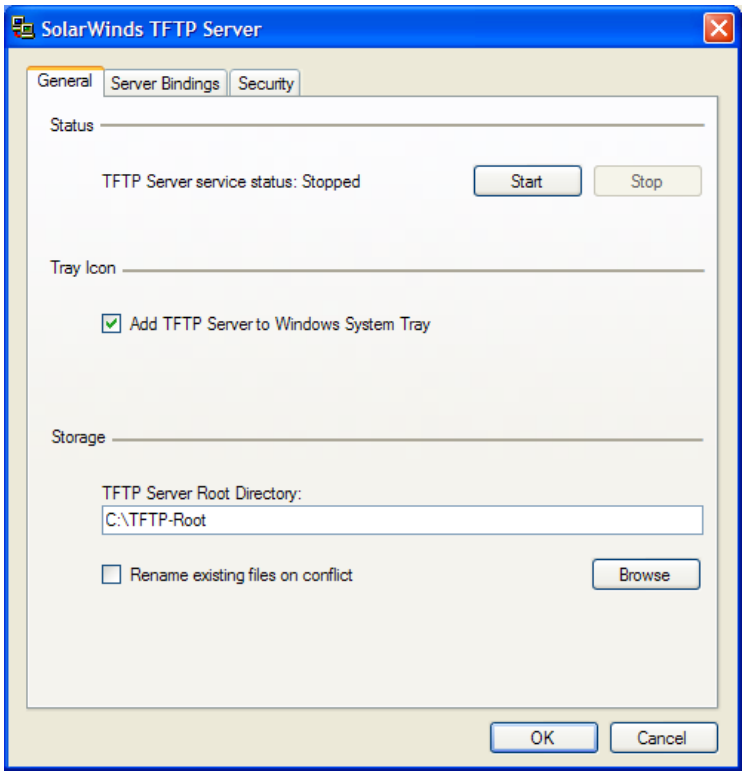
4. Launch the newly installed TFTP server.



5. Select File > Configure.

6. The TFTP server requires a single root directory from which to serve files. Any files that you would like to transfer to the development kit via TFTP will need to be copied into this directory.

   On the *General* tab, enter a directory in the *TFTP Server Root Directory* field and remember your selection. In the example below, the TFTP root directory is *C:\TFTP-Root*.



7. Click OK. Your host PC should now be ready to perform the U-Boot and Linux labs that follow.

3 Load U-Boot to SPI Flash

3.1 Prerequisites

- Host PC set up as described in Section 2
- OMAP-L138 or AM1808 Development Kit
- Null-modem serial cable (not included with the OMAP-L138 eXperimenter Kit)

3.2 Overview

This lab will explain how to load or restore U-Boot to the SOM-M1 from a Windows Host PC. If you are using the Linux Virtual machine, see the *AM1808 & OMAP-L138 Linux User Guide*, Section 6.2. These instructions can be used to restore the User Boot Loader (UBL) and U-Boot in the event that they are erased from the SPI flash or are corrupted. If your SOM-M1 already has U-Boot installed in the SPI flash, feel free to skip this lab and proceed to Section 4.

The steps for this lab are outlined below:

1. Download the Texas Instruments (TI) Serial Boot and Flash Loading Utility for OMAP-L138 or AM1808
2. Download the UBL and U-Boot binary files
3. Erase the SPI flash
4. Program the SPI flash with the UBL and U-Boot

3.3 Use Included Software

To take advantage of the software included with this document, copy the *sfh_OMAP-L138.exe* file from the *OMAP-L138_AM1808_U-Boot_Lab_Softare* directory to *C:\flasher* on your host PC (if necessary, create the folder). Once this is done, you can proceed to Section 3.5.

If you would like to download the most recent software versions from the TI website, continue to Section 3.4; however, you are responsible for adapting the instructions in this document to apply to the new software. Please note that some labs will not work without substantial instructional changes.

3.4 (OPTIONAL) Download Software from TI Website

The TI Serial Boot and Flash Loading Utility (hereafter, Flash Loading Utility) for the SOM-M1 can be downloaded from the *Serial Boot and Flash Loading Utility for OMAP-L138* wiki article.[3] Read the remainder of the wiki article to make sure you download the appropriate software version for the specific PSP/SDK release you are using, as well as any other requirements.

The Linux Software Development Kit (SDK) for the OMAP-L138 SOM-M1 and AM1808 SOM-M1can be found on their respective software pages notes below.

- Linux Software Development Kit (SDK) for OMAP-L138 Processors[4]
- Linux Software Development Kit (SDK) for AM1x Microprocessors[5]

NOTE: In order to download the software, you will be prompted to fill out the necessary identifying web forms.

[3] http://processors.wiki.ti.com/index.php/Serial_Boot_and_Flash_Loading_Utility_for_OMAP-L138
[4] http://focus.ti.com/docs/toolsw/folders/print/linuxsdk-omapl138.html
[5] http://focus.ti.com/docs/toolsw/folders/print/linuxsdk-am1x.html

3.5 Copy UBL and U-Boot Files to Host PC

1. Locate the directory where *sfh_OMAP-L138.exe* is located from step 3.3. This should be *C:\flasher.*
2. Locate the *ubl_OMAPL138_SPI_MEM.bin* and *u-boot.bin* file in the *OMAP-L138_AM1808_U-Boot_Lab_Software* directory. Copy these files to the *C:\flasher* directory on your host PC.

IMPORTANT NOTE: If using newer kit binary files downloaded from TI's website, they will not fit into the onboard SPI flash address space. Instead, the *arm-nor-ais.bin* or *arm-nand-ais.bin* files must be placed into the NOR or NAND flash on the EVM Development Kit UI Board (this option is not available for eXperimenter Kits).

3.6 Erase SPI Flash

Erasing the SPI flash is optional and is used to clean the memory before programming new data into the SPI flash. Section 3.7 describes how to program the SPI flash.

1. With the development kit turned off, set pins 7 and 8 on the S7 DIP switch block to the ON position. This will permit the Flash Loading Utility to operate.

S7 Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

2. Connect the serial cable to the serial debug port on the development kit and to an available COM port on your host PC.

   NOTE: The TI Flash Loading Utility assumes it will be communicating with the COM1 port; make sure the serial cable is associated with COM1 on your host PC.

3. Open a command prompt window on your host PC and change the directory to *C:\flasher*.

   NOTE: The command prompt window at *C:\flasher* must be the only terminal window open. Do not have Tera Term running for this operation.

```
C:\Documents and Settings\user> cd C:\flasher
```

4. Next, erase the flash.

```
C:\flasher> sfh_OMAP-L138.exe -erase
```

   You will receive a message that the Flash Loading Utility is attempting to connect to device COM1.

```
-----------------------------------------------------
  TI Serial Flasher Host Program for OMAP-L138
  <C> 2010, Texas Instruments, Inc.
  Ver. 1.67
-----------------------------------------------------

Platform is Windows.
     [TYPE] Global erase
   [DEVICE] SPI_MEM

Attempting to connect to device COM1...
Press any key to end this program at any time.
```


5. Power on the development kit. After a few moments, an *Erasing flash* message will appear along with a progress bar.

   NOTE: If a waiting for BOOTME message continues to appear after powering on the development kit, press the S5 reset button on the baseboard—this will send the *BOOTME* command to the Flash Loading Utility.

6. Once the erase is complete, power off the development kit and proceed to the next section to program the SPI flash.

3.7 Program SPI Flash with UBL and U-Boot

The UBL and U-Boot are programmed at the same time using a single command. As explained in Section 3.5, both of these files must be in the *C:\flasher* directory for the Flash Loading Utility to program them in SPI flash.

1. Ensure the development kit is powered off.
2. Set DIP switches S7:7 and S7:8 to the ON position. This will allow the Flash Loading Utility to operate.

S7 Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON

3. Open a command prompt window on your host PC and change the directory to *C:\flasher* if you are not already there.

```
C:\Documents and Settings\user> cd C:\flasher
```

4. Program the UBL and U-Boot binaries to SPI flash.

```
C:\flasher> sfh_OMAP-L138.exe –flash ubl_OMAPL138_SPI_MEM.bin u-
boot.bin
```

5. Power on the development kit. After a few moments, a *Flashing UBL ubl_OMAPL138_SPI_MEM.bin* message will appear along with a progress bar. When complete, a *Flashing application u-boot.bin* message will appear along with a progress bar.

   NOTE: If the waiting for BOOTME message continues to appear after powering on the development kit, press the S5 reset button on the baseboard—this will send the *BOOTME* command to the Flash Loading Utility.

6. Once the flashing is complete, power off the development kit.

4 Kit Communications

4.1 Prerequisites

- Host PC set up as described in Section 2
- OMAP-L138 or AM1808 Development Kit
- Null-modem serial cable (not included with OMAP-L138 eXperimenter Kit)

4.2 Overview

The development kit comes with U-Boot programmed in the SPI flash. U-Boot presents a command shell to the user via the development kit's debug serial port. This lab will explain how to establish a proper serial connection to the development kit.

4.3 Terminal Emulator Setup

If you are familiar with serial communications and a terminal emulation program, open your terminal emulator and set the port settings to:

1. Baud rate: **115200**
2. Data: **8 bit**
3. Parity: **None**
4. Stop: **1 bit**
5. Flow control: **None**

If you are not familiar with serial communications, follow the instructions in Section 4.3.1 to set up Tera Term.

4.3.1 Tera Term Setup

Tera Term[6] is a terminal emulation program available as a free download from the Logic PD support site. Logic PD recommends the use of Tera Term over HyperTerminal, as it appears to be a more stable program. All Tera Term settings are controlled by an .ini file that you can modify as needed to make your time in Tera Term as efficient as possible. For example, you can preset the port settings.

1. Start the Tera Term program.
2. From the menu, select Setup > Serial port.
3. Select the appropriate COM port for your host PC.
4. Change the port settings to:
   a. Baud rate: **115200**
   b. Data: **8 bit**
   c. Parity: **None**
   d. Stop: **1 bit**
   e. Flow control: **None**

   These settings are depicted in Figure 4.1below.

[6] http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=1420

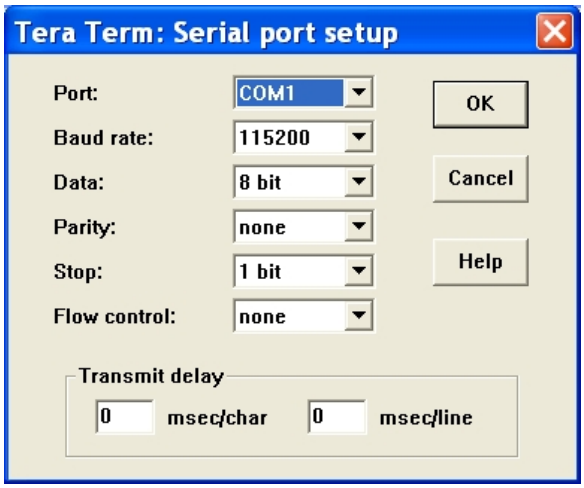
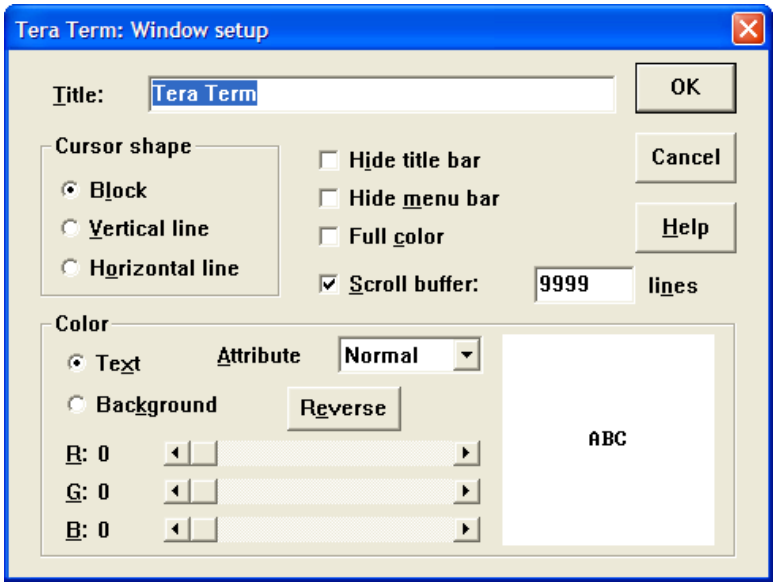


Figure 4.1: Tera Term Serial Port Settings

5. Click OK.
6. From the menu, select Setup > Window.
7. Select the checkbox labeled "Scroll buffer" and make the amount as big as you like, up to 9999.



8. Click OK.

4.4 Connect Kit to Host PC

1. Set all S7 DIP switches to the OFF position.

S7 Pin#	1	2	3	4	5	6	7	8
Position	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

2. Connect a serial cable to the serial debug port on the development kit and to an available COM port on your host PC.

3. Power on the development kit; you should see the U-Boot output below.

```
OMAP-L138 initialization passed!
Booting TI User Boot Loader
        UBL Version: 1.65
Device OPP (300MHz, 1.2V)
        UBL Flashtype: SPI
Starting SPI Memory Copy...
Valid magicnum, 0x55424CBB, found at offset 0x00010000.
   DONE
Jumping to entry point at 0xC1080000.
MMC:   davinci: 0
SF: Detected M25P64 with page size 256, total 8 MiB
In:    serial
Out:   serial
Err:   serial
ARM Clock : 300000000 Hz
DDR Clock : 132000000 Hz
Net:   Ethernet PHY: GENERIC @ 0x00
DaVinci-EMAC
Hit any key to stop autoboot:  0
U-Boot >
```

> **NOTE:** If you see a BOOTME message instead of output from U-Boot when you power on the development kit, the most likely scenario is that S7 DIP switches 7 and 8 are still set to the ON position. Power off the development kit, make sure all switches are in the OFF position, and then power on the kit again.

Congratulations, you have successfully established a serial connection to the development kit.

5 U-Boot Help System and Environment Variables Lab

5.1 Prerequisites

- Host PC set up as described in Section 2
- OMAP-L138 or AM1808 Development Kit
- Null-modem serial cable (not included with OMAP-L138 eXperimenter Kit)
- Established serial connection to the development kit running U-Boot

5.2 Overview

U-Boot is an open-source bootloader/monitor commonly used to boot embedded Linux systems. This lab will introduce you to some basic U-Boot concepts.

5.3 U-Boot Help System

U-Boot comes with a built-in help system. If you need help with a specific U-Boot command, invoking the help system without parameters will display a complete list of the commands available.

1. At the U-Boot > prompt, enter the *help* or *?* command.

```
U-Boot > help
```

or

```
U-Boot > ?
```

You should see the output below. **NOTE:** The available commands displayed by your development kit may differ from those shown below depending on what was included in the U-Boot binary programmed into your SPI flash.

```
?       - alias for 'help'
askenv  - get environment variables from stdin
base    - print or set address offset
bdinfo  - print Board Info structure
boot    - boot default, i.e., run 'bootcmd'
bootd   - boot default, i.e., run 'bootcmd'
bootm   - boot application image from memory
bootp   - boot image via network using BOOTP/TFTP protocol
cmp     - memory compare
coninfo - print console devices and information
cp      - memory copy
crc32   - checksum calculation
dhcp    - boot image via network using DHCP/TFTP protocol
echo    - echo args to console
editenv - edit environment variable
env     - environment handling commands
exit    - exit script
ext2load- load binary file from a Ext2 filesystem
ext2ls  - list files in a directory (default /)
false   - do nothing, unsuccessfully
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls   - list files in a directory (default /)
```



```
go      - start application at address 'addr'
help    - print command description/usage
iminfo  - print header information for application image
imxtract- extract a part of a multi-image
itest   - return true/false on integer compare
loadb   - load binary file over serial line (kermit mode)
loads   - load S-Record file over serial line
loady   - load binary file over serial line (ymodem mode)
loop    - infinite loop on address range
md      - memory display
mdc     - memory display cyclic
mii     - MII utility commands
mm      - memory modify (auto-incrementing address)
mmc     - MMC sub system
mmcinfo - display MMC info
mtest   - simple RAM read/write test
mw      - memory write (fill)
mwc     - memory write cyclic
nfs     - boot image via network using NFS protocol
nm      - memory modify (constant address)
ping    - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
reset   - Perform RESET of the CPU
run     - run commands in an environment variable
saveenv - save environment variables to persistent storage
saves   - save S-Record file over serial line
setenv  - set environment variables
sf      - SPI flash sub-system
showvar - print local hushshell variables
sleep   - delay execution for some time
source  - run script from memory
sspi    - SPI utility command
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
true    - do nothing, successfully
version - print monitor version
```

2. At the U-Boot prompt, use *help* to obtain additional information about the *mtest* and *run* commands as shown below.

```
U-Boot > help mtest
mtest - simple RAM read/write test

Usage:
mtest [start [end [pattern [iterations]]]]
U-Boot > help run
run   - run commands in an environment variable

Usage:
run var [...]
      - run the commands in the environment variable<s> 'var'
U-Boot >
```

Enter the *help* command followed by the name of a command displays the help topic specific to that command. Some commands need parameters and some take optional parameters. Optional parameters are listed inside square-brackets [ ] and required parameters are listed normally.

The *mtest* command has no required parameters to run. However, you could supply parameters corresponding to the start, end, and pattern the memory test should use.

The *run* command needs at least one parameter named *var*, which is short for "environment variable." Additionally, you may send it optional parameters that correspond to whatever command you are going to run.

5.4 Environment Variables

U-Boot includes support for environment variables, similar to other computer shells like BASH or DOS. Environment variables can be used to store values, names of commands, parameters, etc. The following steps will allow us to explore some of these environment variables.

1. At the U-Boot > prompt, enter the *print* command.

   Print is an alias for the *printenv* command, which displays a list of all known and currently defined environment variables, along with their value. You should see output similar to the picture below.

```
U-Boot > print
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/mmcblk0p2 rw rootwait
ip=off
bootcmd=if mmc rescan 0; then if fatload mmc 0 0xc0600000 boot.scr;
then source 0xc0600000; else fatload mmc 0 0xc0700000 uImage; bootm
c0700000; fi; else sf probe 0; sf read 0xc0700000 0x80000 0x220000;
bootm 0xc0700000; fi
bootdelay=3
bootfile="uImage"
ethact=DaVinci-EMAC
ethaddr=00:08:ee:05:8b:f9
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jul 10 2012 - 15:11:10)

Environment size: 504/65532 bytes
U-Boot >
```

   Many of these values were compiled into U-Boot when it was built. Others were defined by users of the development kit and then saved into flash for subsequent boots. Note the last line of output:

   ```
   Environment size: xxx/65532 bytes
   ```

   This tells you how much space U-Boot has allocated in flash for storing environment variables.

2. At the U-Boot prompt, use *help* to obtain more information about the *setenv* command.

```
U-Boot > help setenv
setenv - set environment variables

Usage:
setenv name value ...
       - set environment variable 'name' to 'value ...'
setenv name
       - delete environment variable 'name'
```



```
U-Boot >
```

The *setenv* command (short for set environment variable) is used to create new environment variables and delete existing environment variables.

3. Using the *setenv* command, let's create a new environment variable named *foo* and assign it a value of *bar*.
   a. At the U-Boot > prompt, enter *setenv foo bar.*
   b. Then enter *echo foo.*
   c. Finally, enter *echo $foo.*

```
U-Boot > setenv foo bar
U-Boot > echo foo
foo
U-Boot > echo $foo
bar
U-Boot >
```

The *setenv* command creates the *foo* environment variable and the *echo* command simply prints to the screen whatever parameters you give it. When we prefixed the environment variable with a dollar sign ($), we de-referenced it and U-Boot replaced its name with its value before passing it to the command *echo*.

4. At the U-Boot > prompt, enter the *print* command.

```
U-Boot > print
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/mmcblk0p2 rw rootwait
ip=off
bootcmd=if mmc rescan 0; then if fatload mmc 0 0xc0600000 boot.scr;
then source 0xc0600000; else fatload mmc 0 0xc0700000 uImage; bootm
c0700000; fi; else sf probe 0; sf read 0xc0700000 0x80000 0x220000;
bootm 0xc0700000; fi
bootdelay=3
bootfile="uImage"
ethact=DaVinci-EMAC
ethaddr=00:08:ee:05:8b:f9
foo=bar
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jul 10 2012 - 15:11:10)

Environment size: 514/65532 bytes
U-Boot >
```

Notice the new environment variable named *foo* with an assigned value of *bar* listed at the bottom of the output. Additionally, notice that the size of your environment variables has changed.

5. Reset your development kit and press any key before the autoboot timeout to enter U-Boot.
6. At the U-Boot > prompt, enter the *print* command.

Is the *foo* environment variable still there? It shouldn't be. That's because environment variables aren't saved across power cycles unless we tell U-Boot to explicitly do so by using the *saveenv* command.

However, we don't want to waste time saving a silly environment variable like *foo*; let's save something useful instead. In future labs, we are going to download files to the kit. Some of these files, like the Linux kernel, will eventually be stored in the kit's flash memory. Before we can store something in flash, we must download it into a RAM buffer. Let's create an environment variable named *dlbuf* that has the value 0xC040_0000 and save it so we can refer to it later.

7. At the U-Boot > prompt, create the *dlbuf* environment variable.

```
U-Boot > setenv dlbuf 0xC0400000
```

8. Next, save the environment.

```
U-Boot > saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
U-Boot >
```

9. At the U-Boot > prompt, enter the *print* command and verify that the newly created environment variable *dlbuf* exists.

```
U-Boot > print
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/mmcblk0p2 rw rootwait
ip=off
bootcmd=if mmc rescan 0; then if fatload mmc 0 0xc0600000 boot.scr;
then source 0xc0600000; else fatload mmc 0 0xc0700000 uImage; bootm
c0700000; fi; else sf probe 0; sf read 0xc0700000 0x80000 0x220000;
bootm 0xc0700000; fi
bootdelay=3
bootfile="uImage"
dlbuf=0xC0400000
ethact=DaVinci-EMAC
ethaddr=00:08:ee:05:8b:f9
foo=bar
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jul 10 2012 - 15:11:10)

Environment size: 533/65532 bytes
U-Boot >
```

10. Let's create another environment variable that holds the address in flash memory where we will store our Linux kernel.

    At the U-Boot > prompt, create the *kerneladdr* environment variable.

```
U-Boot > setenv kerneladdr 0x00080000
```

11. Next, save the environment.


```
U-Boot > saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
U-Boot >
```

12. At the U-Boot > prompt, enter the *print* command and verify that your newly created environment variable *kerneladdr* exists.

```
U-Boot > print
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/mmcblk0p2 rw rootwait
ip=off
bootcmd=if mmc rescan 0; then if fatload mmc 0 0xc0600000 boot.scr;
then source 0xc0600000; else fatload mmc 0 0xc0700000 uImage; bootm
c0700000; fi; else sf probe 0; sf read 0xc0700000 0x80000 0x220000;
bootm 0xc0700000; fi
bootdelay=3
bootfile="uImage"
dlbuf=0xC0400000
ethact=DaVinci-EMAC
ethaddr=00:08:ee:05:8b:f9
foo=bar
kerneladdr=0x00080000
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jul 10 2012 - 15:11:10)

Environment size: 557/65532 bytes
U-Boot >
```

5.5 Additional Environment Variables

Setting up variables in U-Boot can make many tasks more efficient and easier to debug. The following steps will set up variables that will be used throughout the remainder of this document.

5.5.1 Set Up Variables for Remaining Labs

At the U-Boot > prompt, enter the commands below to set up several variables that will be used in the remaining labs.

```
U-Boot > setenv kernaddr 0xc0700000
U-Boot > setenv kernoff 0xa0000
U-Boot > setenv kernlen 0x280000
U-Boot > setenv rootfsaddr 0xC1180000
U-Boot > setenv rootfsoff 0x320000
U-Boot > setenv rootfslen 0x400000
```

5.5.2 Access Upper 64 MB mDDR SDRAM on EVM Development Kits

The SOMs included with the OMAP-L138 and AM1808 EVM Development Kits contain 128 MB mDDR SDRAM. On these SOMs, it is possible to access the upper 64 MB of mDDR SDRAM.

IMPORTANT: The upper 64 MB of mDDR SDRAM memory is not available on the OMAP-L138 and AM1808 eXperimenter Kits. If you are using an OMAP-L138 or AM1808 eXperimenter Kit, please proceed to the next section.

1. At the U-Boot > prompt, set the *bootargs* variable.

```
U-Boot > setenv bootargs 'console=ttyS2,115200n8 root=/dev/ram0 rw
initrd=0xc1180000,4M ip=dhcp mem=32M@0xc0000000 mem=64M@0xc4000000'
```

NOTE: The *mem=64M@0xc4000000* addition to the bootargs is the starting address for the upper 64MB mDDR SDRAM in the OMAP-L138 and AM1808 EVM Development Kits.

2. At the U-Boot > prompt, save the environment.

```
U-Boot > saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
```

Memory read and write commands are now available to the upper 64 MB mDDR SDRAM. Since the environment variable was saved to SPI flash, this upper region of mDDR SDRAM will be available even after power cycles.

5.6 Resetting Environment Variables to Defaults

Should there be a desire to reset the environmental variables back to factory settings following the steps outlined below.

1. At the U-Boot > prompt, set reset the environmental variables to default

```
U-Boot > env defaults –f
## Resetting to default environment
```

2. At the U-Boot > prompt, save the environment.

```
U-Boot > saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
```

3. At the U-Boot > prompt, reset the processor.

```
U-Boot > reset
```

The processor will reboot with default environmental variables and the MAC address will be erased. To enable the network, the MAC address must be programmed.

NOTE: To set the MAC address, please refer to Section 6 of Logic PD's *AN 556 AM1808 & OMAP-L138 SOM-M1 SPI Flash Recovery*.[7]

[7] http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=778

6 Investigating U-Boot

6.1 Prerequisites

- Host PC set up as described in Section 2
- OMAP-L138 or AM1808 Development Kit
- Null-modem serial cable (not included with OMAP-L138 eXperimenter Kit)
- Established serial connection to the development kit running U-Boot

6.2 Overview

U-Boot includes many commands useful for debugging a system; accessing an Ethernet-based network; and working with RAM, flash, and other types of memory technologies. This lab will introduce you to those commands.

6.3 Obtain General SOM Information

1. At the U-Boot prompt, enter the command below to obtain information about U-Boot.

```
U-Boot > version

U-Boot 2010.12 (Jul 10 2012 - 15:11:10)
U-Boot >
```

   You should see U-Boot information about how and when it was built. Anytime you submit a support request or bug-report, you should include this version string.

2. At the U-Boot prompt, enter the *print* command.

```
U-Boot > print
bootcmd=sf probe 0;sf read 0xc0700000 0x80000 0x220000;bootm 0xc0700000
baudrate=115200
bootfile="uImage"
ethaddr=00:08:ee:03:bf:f9
dlbuf=0xc0400000
kerneladdr=0x00080000
bootdelay=6
bootargs=console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc11800000,
4M ip=dhcp mem=32M@0xc0000000 mem=64M@0xc4000000
stdin=serial
stdout=serial
stderr=serial
ver= U-Boot 2010.12 (Jul 10 2012 - 15:11:10)

Environment size: 386/65532 bytes
U-Boot >
```

   The *print* command (short for printenv) prints out various parameters as they relate to your development kit. Of typical interest is the location and size of RAM and flash. Other parameters that might interest you include: MAC addresses and serial-port baud rate setting.

   Much of the above information is passed to the Linux kernel.

6.4 SOM-M1 Memory Map

The following memory map shows the location of different types of memory and program files on the SOM-M1. This information is useful for navigating, locating, and storing configuration and application files.

Memory	Start	End	Size
Internal ROM (DSP L2 ROM)	0x1170_0000	0x117F_FFFF	1024KB
Internal SRAM (shared RAM)	0x8000_0000	0x8001_FFFF	128KB
DDR (EVM)	0xC000_0000	0xC7FF_FFFF	128 MB
DDR (eXperimenter)	0xC000_0000	0xC3FF_FFFF	64 MB
NOR (on UI board)	0x6000_0000	0x607F_FFFF	8 MB
Peripheral Space	See datasheet		

6.4.1 SPI Flash Contents

The SPI flash chip is located on the SOM-M1, connected to the SPI1 port, and is not memory mapped. The SPI flash size is 8 MB. This fills an address space of 0x7F_FFFF.

Address	Item
0x000000	UBL bootloader (*ubl_OMAPL138_SPI.bin*)
0x010000	U-Boot prepended with config (*u_boot.bin*)
0x0a0000	Linux image (*uImage*)
0x320000	Filesystem (*ramdisk-base.gz*)
0x7f0000	MAC Address

6.4.2 UI Board NAND Flash

The UI board that ships with the EVM Development Kits has a NAND flash socket on the UI board and the following NAND chip installed:

- Micron MT29F4G08AAC
- Base Address: 0x6200_0000
- Page size: 2112 bytes (2048 + 64 bytes)
- Block size: 64 pages (128K + 4K bytes)
- Plane size: 2 planes x 2048 blocks per plane
- Device size: 512 MB (4 Gb), 4096 blocks

6.5 Working with RAM memory

6.5.1 *md* Command

1. At the U-Boot prompt, use *help* to obtain additional information about the *md* command.

```
U-Boot > help md
md - memory display

Usage:
md [.b, .w, .l] address [# of objects]
U-Boot >
```


The *md* command (short for memory display) is used to look at the contents of different types of memory. An argument can denote the size you would like to chunk the memory into.

- md.b — treats memory as 8-bit bytes
- md.w — treats memory as 16-bit words
- md.l — treats memory as 32-bit long words (the default)

Along with the optional size component, you must tell the command what address it should look at and, possibly, how many objects to display.

In a previous lab, we created an environment variable named *dlbuf* that we will now use as the *address* parameter for our memory commands.

2. At the U-Boot prompt, enter the commands below.

```
U-Boot > md.b $dlbuf 8
c0400000: 7b de 7f ff e7 fd b7 ef   {
U-Boot > md.w $dlbuf 8
c0400000: de7b ff7f fde7 efb7 ee3f fd7f ce7f eeb7   {.......?.......
U-Boot > md.l $dlbuf 8
c0400000: ff7fde7b efb7fde7 fd7fee3f eeb7ce7f   {.......?.......
c0400000: ee7bfeb3 cf3fddf7 edfbadff ed7fde7f   ..{...?.........
U-Boot >
```

NOTE: The contents of your kit's memory may be different than those depicted in the output above; however, it is important to recognize that each time the command displays eight objects of whatever width was specified (byte, word, or long-word).

6.5.2 *mm* Command

1. At the U-Boot prompt, use *help* to obtain additional information about the *mm* command.

```
U-Boot > help mm
mm - memory modify (auto-incrementing address)

Usage:
mm [.b, .w, .l] address
U-Boot >
```

The *mm* command (short for memory modify) can be used to interactively change the contents of memory. The *mm* command takes the same size parameters as the *md* command (*mm.b, mm.w, mm.l*). The *mm* command also requires an address parameter.

When you start the command, it will display the address and current contents of memory followed by a "?" prompt. At the prompt, enter any legal hexadecimal number and this new value will be written to the address. The command will then automatically increment the address to the next appropriately sized object and prompt you again. If you enter no value, the memory contents of that address will not be changed.

The command stops as soon as you enter any data that is not a valid hexadecimal number (such as a period).

2. At the U-Boot prompt, enter the commands below.

```
U-Boot > md.l $dlbuf 4
00000000: 00000000 00000000 00000000 00000000   ................
U-Boot > mm.l $dlbuf
00000000: 00000000 ?
```

3. At the ? prompts, enter *0x0*, *0x1*, *0xa*, and a period (.) as shown below.

```
00000000: 00000000 ? 0x0
00000004: 00000000 ? 0x1
00000008: 00000000 ? 0xa
0000000c: 00000000 ? .
```

4. At the U-Boot prompt, enter the command below again.

```
U-Boot > md.l $dlbuf 4
c0400000: ff7fde7b efb7fde7 fd7fee3f eeb7ce7f {.......?.......
U-Boot >
```

Note how the memory contents changed.

6.5.3 *mw* Command

1. At the U-Boot prompt, use *help* to obtain additional information about the *mw* command.

```
U-Boot > help mw
mw - memory write <fill>

Usage:
mw [.b, .w, .l] address value [count]
U-Boot >
```

The *mw* command (short for memory write) presents a non-interactive way to change the contents of RAM. Again, this command takes the same object-size qualifiers as the other memory commands.

The *mw* command also takes an optional *count* argument that can be used to initialize whole areas of memory. If the command is called without the *count* argument, the specified value will be written only to the given address.

2. At the U-Boot prompt, enter the commands below.

```
U-Boot > mw.l $dlbuf 0xaabbccdd 4
U-Boot > md.l $dlbuf 4
c0400000: aabbccdd aabbccdd aabbccdd aabbccdd   ................
U-Boot >
```

In the output above, you can see that the pattern was written into the designated memory segment.

6.5.4 *cp* Command

1. At the U-Boot prompt, use *help* to obtain additional information about the *cp* command.

```
U-Boot > help cp
cp - memory copy

Usage:
cp [.b, .w, .l] source target count
U-Boot >
```

The *cp* command is used to copy one section of memory to another. The parameters are similar to each of the other memory-specific commands that we have been exploring.

2. At the U-Boot prompt, enter the commands below.

```
U-Boot > cp.l $dlbuf 0xC0500000 4
U-Boot > md.l 0xC0500000 4
c0500000: aabbccdd aabbccdd aabbccdd aabbccdd   ................
U-Boot >
```

In the output, verify that the pattern previously written to 0xC040_0000 was properly copied to 0xC050_0000.

6.5.5 *cmp* Command

1. At the U-Boot prompt, use *help* to obtain additional information about the *cmp* command.

```
U-Boot > help cmp
cmp - memory compare

Usage:
cmp [.b, .w, .l] addr1 addr2 count
U-Boot>
```

The *cmp* command (short for compare) is a programmatic way to compare two sections of memory. It will be more reliable than visually comparing two chunks of RAM using the *md* command.

2. At the U-Boot prompt, enter the command below; the output should tell you that the compared blocks of memory were the same.

```
U-Boot > cmp.l $dlbuf 0xC0500000 4
Total of 4 words were the same
U-Boot>
```

3. Next, enter the commands below.

```
U-Boot > mw.l $dlbuf 0x11223344 4
U-Boot > cmp.l $dlbuf 0xC0500000 4
word at 0xc0400000 <0x11223344> != word at 0xc0500000 <0xaabbccdd>
Total of 0 words were the same
```



```
U-Boot >
```

Notice that the command stopped as soon as it encountered a difference between the blocks of memory being compared.

6.5.6 *crc* Command

1. At the U-Boot prompt, use *help* to obtain additional information about the *crc* command.

```
U-Boot > help crc
crc32 - checksum calculation

Usage:
crc32 address count [addr]
      - compute CRC32 checksum [save at addr]
-v address count crc
      - verify crc of memory area
U-Boot >
```

The *crc* command calculates a CRC32 checksum over a block of memory. This can be used to compare two blocks of memory like the *cp* command. We will use it later to verify that our Linux kernel was properly burned into flash memory.

2. At the U-Boot prompt, enter the commands below.

```
U-Boot > crc $dlbuf 4
CRC32 for c0400000 ... c0400003 ==> de1d2d6d
U-Boot > crc 0xc0500000 4
CRC32 for c0500000 ... c0500003 ==> 5cfdecf9
U-Boot >
```

Notice that the two checksums are different.

3. At the U-Boot prompt, enter the commands below.

```
U-Boot > mw.l $dlbuf 0xaabbccdd 4
U-Boot > crc $dlbuf 4
CRC32 for c0400000 ... c0400003 ==> 5cfdecf9
U-Boot >
```

The checksum should match the one computed for the memory block at address 0xC050_0000.

Congratulations, you have just learned how to explore and alter RAM memory values for debug activities in U-Boot.

7 Download Linux Kernel and Root Filesystem to SPI Flash

The default configuration for U-Boot is to boot from the SD card. If booting from SPI required, follow the steps in Section 7 to download files from a Host PC, make sure the TFTP software is installed from Section 2.3.

7.1 Prerequisites

- Host PC set up as described in Section 2
- OMAP-L138 or AM1808 Development Kit
- Null-modem serial cable (not included with OMAP-L138 eXperimenter Kit)
- Established serial connection to the development kit running U-Boot

7.2 Overview

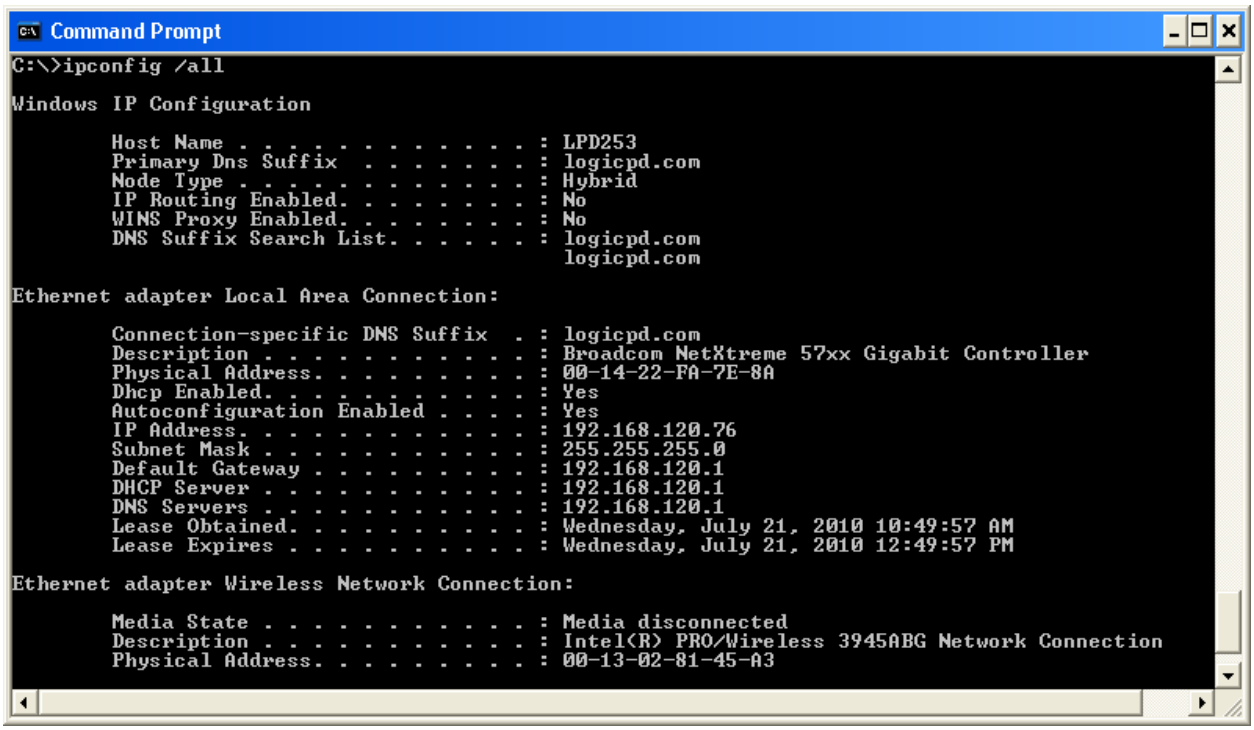
This lab will explain how to set up the U-Boot network interface, download a Linux kernel and root filesystem, save all items into flash memory, and boot Linux.

7.3 Establish Network Connection

1. Connect an Ethernet cross-over cable to the Ethernet port (J31) on the development kit baseboard. The Ethernet cross-over cable can be connected directly to the host PC.
2. Open a command prompt window.
3. At the DOS prompt, enter the command below.

```
C:\> ipconfig /all
```

This command should display the information pertaining to the host PC's network configuration as shown below.



Take special note of the following information:

- IP Address: IP address of the host PC
- Subnet Mask: Network subnet mask
- Default Gateway: IP address of default gateway
- DHCP Server: IP address of DHCP server for this network
- DNS Servers: Domain name server (primary and others)

4. Return to the Tera Term window.
5. At the U-Boot prompt, print the environment variables.

```
U-Boot > printenv
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/mmcblk0p2 rw rootwait
ip=off
bootcmd=if mmc rescan 0; then if fatload mmc 0 0xc0600000 boot.scr;
then source 0xc0600000; else fatload mmc 0 0xc0700000 uImage; bootm
c0700000; fi; else sf probe 0; sf read 0xc0700000 0x80000 0x220000;
bootm 0xc0700000; fi
bootdelay=3
bootfile="uImage"
ethact=DaVinci-EMAC
ethaddr=00:08:ee:03:5b:1e
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jul 10 2012 - 15:11:10)

Environment size: 504/65532 bytes
```

Notice the environment variables named below.

- ipaddr: IP address of the development kit
- serverip: IP address of the host PC from which you want to download
- ethaddr: Ethernet MAC address displayed on the SOM's sticker
- netmask: Network subnet mask used by the development kit
- gatewayip: IP address of gateway (router) to use
- dnsip: IP address of Domain Name Server

NOTE: Some of these variables may be undefined and will not appear in your print list. If your terminal window does not show these values, as in the example above, the next steps will set these values so the development kit can connect to the network.

NOTE: To set the MAC address, please refer to Logic PD's *AN 556 AM1808 & OMAP-L138 SOM-M1 SPI Flash Recovery*.[8]

6. Use the *setenv* command to set the above environment variables to values that correspond to your network.
   a. Set the netmask for your network using the command below.

```
U-Boot > setenv netmask www.xxx.yyy.zzz
```

   b. Set the server IP address, where **www.xxx.yyy.zzz** is the IP address of your host PC.

```
U-Boot > setenv serverip www.xxx.yyy.zzz
```

[8] http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=778

c. Set the development kit IP address.

IMPORTANT: Select an address that is not in use on your network; match the www, xxx, and yyy values to your network and select a value not in use for the zzz entry.

```
U-Boot > setenv ipaddr www.xxx.yyy.zzz
```

d. Set the DNS IP address.

```
U-Boot > setenv dnsip www.xxx.yyy.zzz
```

7. Save these settings.

```
U-Boot > saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
U-Boot >
```

8. Print the environment variables; compare the results to those from Step 5 above.

```
U-Boot > print
baudrate=115200
bootargs=mem=32M console=ttyS2,115200n8 root=/dev/mmcblk0p2 rw rootwait
ip=off
bootcmd=if mmc rescan 0; then if fatload mmc 0 0xc0600000 boot.scr;
then source 0xc0600000; else fatload mmc 0 0xc0700000 uImage; bootm
c0700000; fi; else sf probe 0; sf read 0xc0700000 0x80000 0x220000;
bootm 0xc0700000; fi
bootdelay=3
bootfile=k2000.0
dnsip=192.168.120.1
dnsip2=192.168.120.2
ethact=DaVinci-EMAC
ethaddr=00:08:ee:03:5b:1e
fileaddr=C0700000
filesize=4052
ipaddr=192.168.120.176
netmask=255.255.252.0
serverip=192.168.120.76
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jul 10 2012 - 15:11:10)

Environment size: 643/65532 bytes
U-Boot >
```

9. Next, verify that network connectivity exists using the command below, where **www.xxx.yyy.zzz** is the IP address of the host PC. In the example below, 192.168.120.76 will appear as alive.

```
U-Boot > ping 192.168.120.76
```

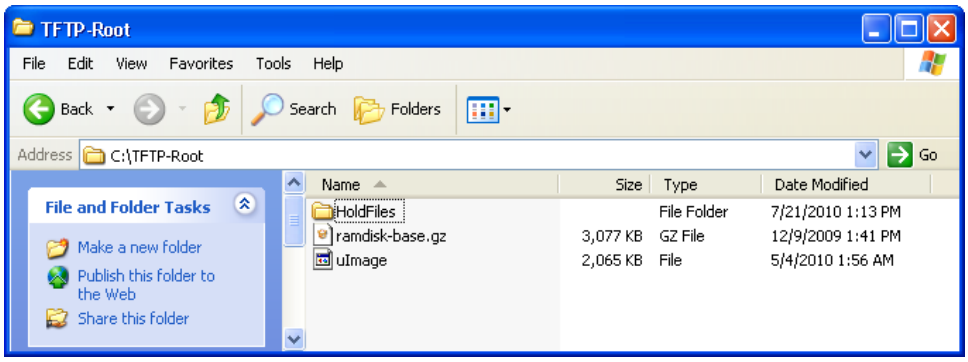


```
Using device
Host 192.168.120.76 is alive
U-Boot >
```

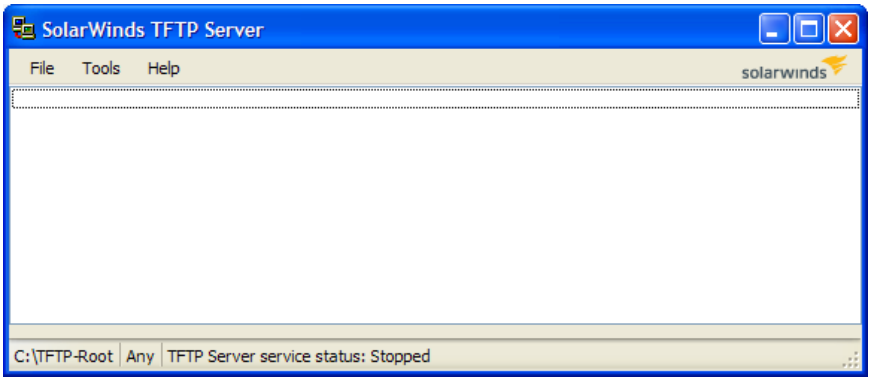
NOTE: Sometimes the *ping* command will seem unresponsive the first time it is used. If this happens, kill the command by pressing **Ctrl+C** in the Tera Term window and then try the command again. You should see a message from U-Boot stating that the host is alive.

7.4 Download Linux Kernel and Filesystem

1. Ensure that the Linux kernel and filesystem image have been copied to a directory accessible by the TFTP server.

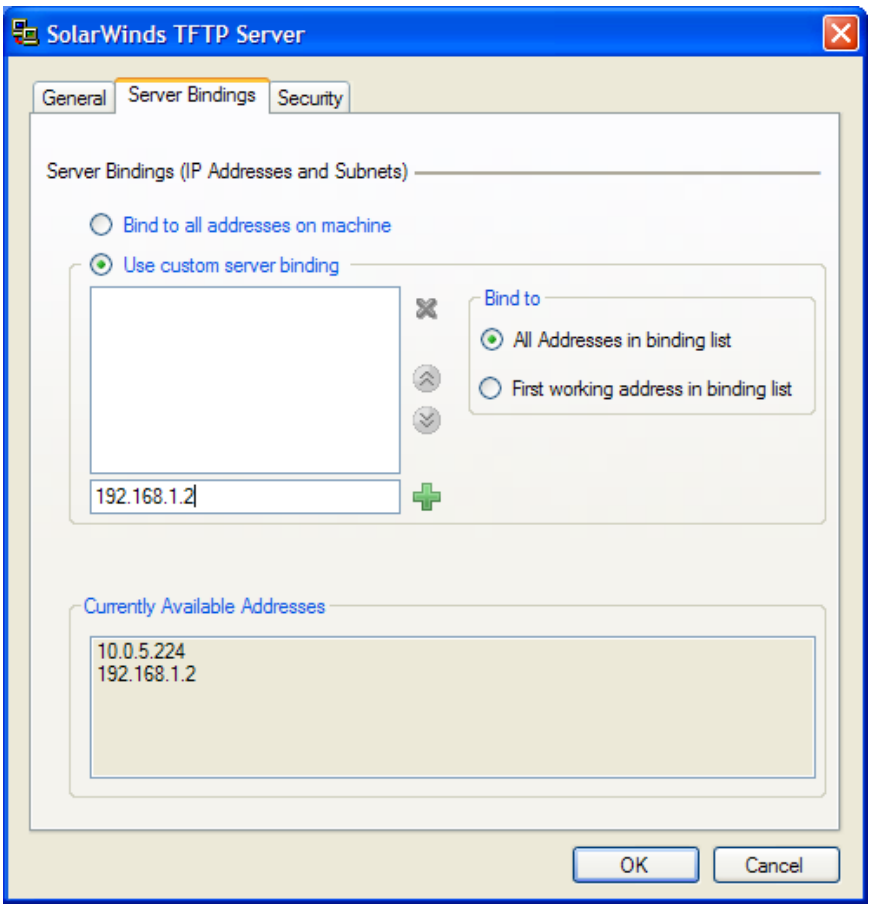


2. Start a TFTP server on the host PC.



3. In the *SolarWinds TFTP Server* window, select File > Configure.

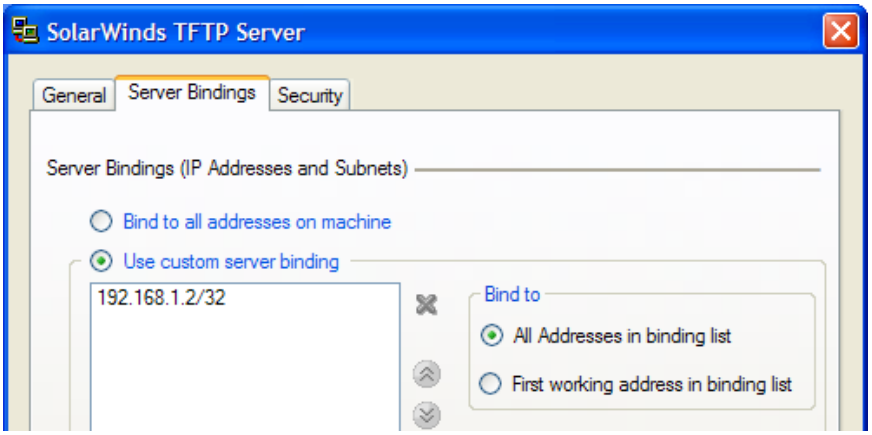
4. In the window that appears, select the *Server Bindings* tab.



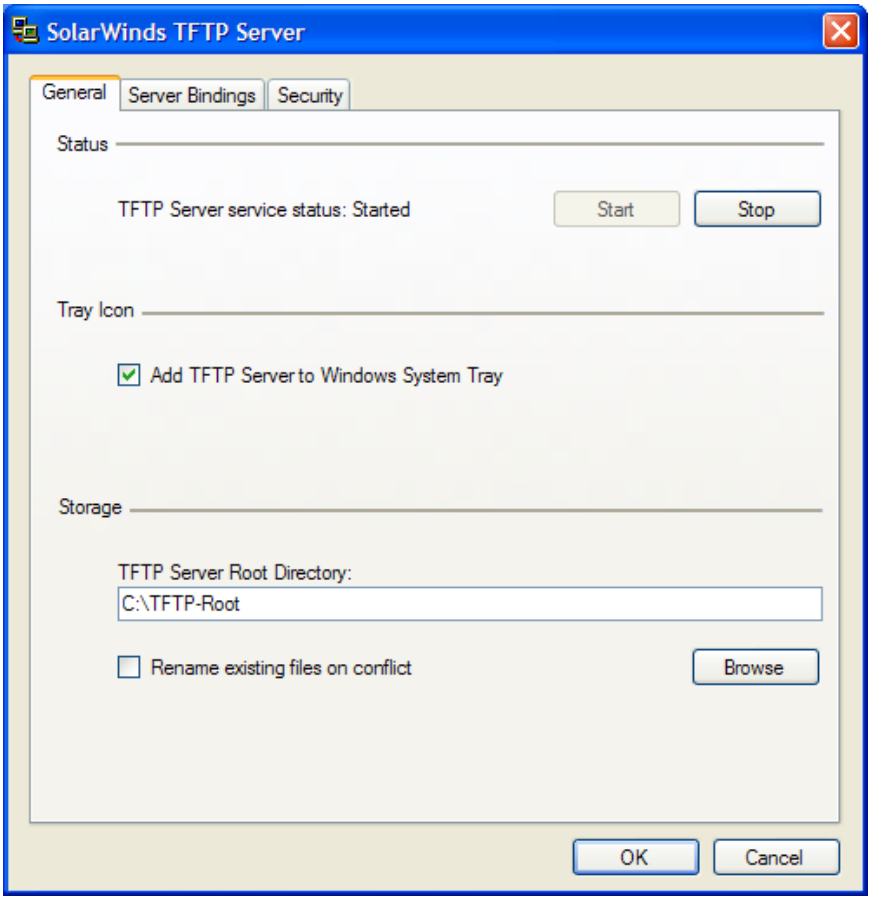
5. Select the radio button labeled "Use custom server binding."

6. Type the IP address of your host PC in the field next to the green plus sign (this IP address should appear in the *Currently Available Addresses* list at the bottom of the window). Click the green plus sign.

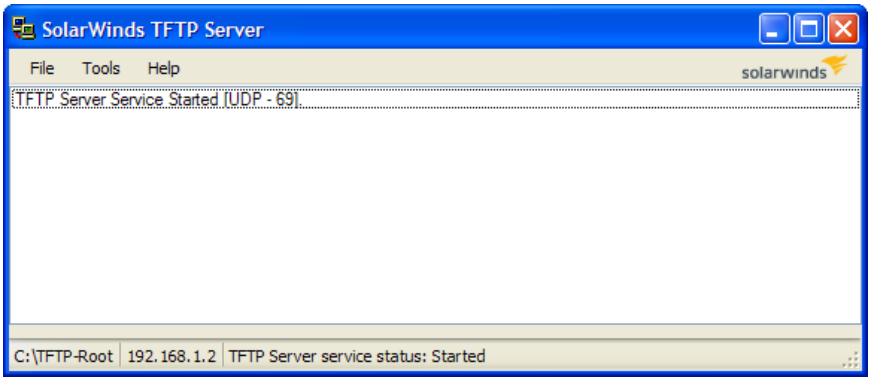
   The host PC IP address will now appear in the window below the "Use custom server binding" radio button.



7. In the same window, select the *General* tab and click Start to start TFTP server services. Click OK to exit the configuration window.



8. In the *SolarWinds TFTP Server* window, you should now see a *TFTP Server Service Started* message. The host PC IP address should also appear at the bottom of the window.



7.4.1 Program Kernel Image in SPI Flash

1. Return to the Tera Term window. Transfer the kernel image to SDRAM.

```
U-Boot > tftp ${kernaddr} uImage
Using  device
TFTP from server 192.168.120.153; our IP address is 192.168.120.133
Filename 'uImage'.
Load address: 0xc0700000
Loading:  #################################################
             #################################################
             ########
done
Bytes transferred = 2105728 (202180 hex)
```

Make a note of the number of bytes transferred. In the example above, this number is 202180 hex. The erase and write sizes in the commands below must be larger than this size number.

2. Set the SPI flash as the current device.

```
U-Boot > sf probe 0
8192 KiB M25P64 at 0:0 is now current device
```

3. Erase the SPI flash.

```
U-Boot > sf erase ${kernoff} ${kernlen}
```

4. Send the kernel image from SDRAM to SPI flash.

```
U-Boot > sf write ${kernaddr} ${kernoff} ${kernlen}
```

Wait for the U-Boot prompt to return; this should take about thirty seconds.

7.4.2 Program Root Filesystem in SPI Flash

1. Transfer the root filesystem to SDRAM.

```
U-Boot > tftp ${rootfsaddr} ramdisk-base.gz
Using  device
TFTP from server 192.168.120.153; our IP address is 192.168.120.133
Filename 'ramdisk-base.gz'.
Load address: 0xc1180000
Loading:  #################################################
             #################################################
             ###################
done
Bytes transferred = 3150629 (301325 hex)
```

Make a note of the size of the file transferred. In the example above, the size is 301325 hex. The erase and write sizes in the commands below must be larger than this size number.

Also note that with 64 MB of room in the SPI flash, the top address for the SPI flash is 0x7FFFFF. The root filesystem (starting at address 0x280000) must be smaller than 0x580000 to fit into the SPI flash.

2. Erase the portion of SPI flash that will contain the filesystem.

```
U-Boot > sf erase ${rootfsoff} ${rootfslen}
```

3. Send the filesystem from SDRAM to SPI flash.

```
U-Boot > sf write ${rootfsaddr} ${rootfsoff} ${rootfslen}
```

7.5 Set Boot Environment Variables

1. Set the *bootargs* variable.

```
U-Boot > setenv bootargs 'console=ttyS2,115200n8 root=/dev/ram0 rw
initrd=0xc1180000,4M ip=dhcp mem=32M@0xc0000000 mem=64M@0xc4000000'
```

2. Set the *bootcmd* variable.

```
U-Boot > setenv bootcmd 'sf probe 0; sf read ${kernaddr} ${kernoff}
${kernlen};sf read ${rootfsaddr} ${rootfsoff} ${rootfslen};bootm
${kernaddr}'
```

3. Save the environment.

```
U-Boot > saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
```

4. Use the *print* command to check your environment.

```
U-Boot > print
baudrate=115200
bootargs=console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M
ip=dhcp mem=32M@0xc0000000 mem=64M@0xc4000000
bootcmd=sf probe 0; sf read ${kernaddr} ${kernoff} ${kernlen};sf read
${rootfsaddr} ${rootfsoff} ${rootfslen};bootm ${kernaddr}
bootdelay=3
bootfile=k2000.0
dnsip=10.1.2.138
dnsip2=10.1.2.139
ethact=DaVinci-EMAC
ethaddr=00:08:ee:03:5b:1e
fileaddr=C0700000
filesize=301325
ipaddr=10.0.5.65
kernaddr=0xc0700000
kerneladdr=0x00080000
kernlen=0x220000
kernoff=0x80000
netmask=255.255.252.0
```



```
rootfsaddr=0xC1180000
rootfslen=0x400000
rootfsoff=0x2a0000
serverip=10.1.2.114
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jul 10 2012 - 15:11:10)

Environment size: 735/65532 bytes
```

5. Power off the development kit.

7.6 Boot Linux Kernel

1. Power on the development kit. You will eventually see the output below.

```
_____                    _____           _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org arago ttyS2

Arago 2009.09 arago ttyS2

arago login:
```

2. Enter root to log in to the Linux environment.

Congratulations, you have just configured your development kit to automatically boot Linux.

8 Load Linux Kernel and Root Filesystem to SPI Flash from SD Card

8.1 Program Kernel and RootFS from SD Card

8.1.1 Program Kernel Image in SPI Flash

3. Return to the Tera Term window. Transfer the kernel image to SDRAM.

```
U-Boot > fatload mmc 0 ${kernaddr} uImage
reading uImage

2105728 bytes read
```

Make a note of the number of bytes transferred. The erase and write sizes in the commands below must be larger than this size number.

4. Set the SPI flash as the current device.

```
U-Boot > sf probe 0
8192 KiB M25P64 at 0:0 is now current device
```

5. Erase the SPI flash.

```
U-Boot > sf erase ${kernoff} ${kernlen}
```

6. Send the kernel image from SDRAM to SPI flash.

```
U-Boot > sf write ${kernaddr} ${kernoff} ${kernlen}
```

Wait for the U-Boot prompt to return; this should take about thirty seconds.

8.1.2 Program Root Filesystem in SPI Flash

7. Transfer the root filesystem to SDRAM.

```
U-Boot > fatload mmc 0 ${rootfsaddr} ramdisk-base.gz
reading ramdisk-base.gz

3150629 bytes read
```

Make a note of the size of the file transferred. The erase and write sizes in the commands below must be larger than this size number.

Also note that with 64 MB of room in the SPI flash, the top address for the SPI flash is 0x7FFFFF. The root filesystem (starting at address 0x280000) must be smaller than 0x580000 to fit into the SPI flash.

8. Erase the portion of SPI flash that will contain the filesystem.

```
U-Boot > sf erase ${rootfsoff} ${rootfslen}
```

9. Send the filesystem from SDRAM to SPI flash.


```
U-Boot > sf write ${rootfsaddr} ${rootfsoff} ${rootfslen}
```

8.2 Set Boot Environment Variables

10. Set the *bootargs* variable.

```
U-Boot > setenv bootargs 'console=ttyS2,115200n8 root=/dev/ram0 rw
initrd=0xc1180000,4M ip=dhcp mem=32M@0xc0000000 mem=64M@0xc4000000'
```

11. Set the *bootcmd* variable.

```
U-Boot > setenv bootcmd 'sf probe 0; sf read ${kernaddr} ${kernoff}
${kernlen};sf read ${rootfsaddr} ${rootfsoff} ${rootfslen};bootm
${kernaddr}'
```

12. Save the environment.

```
U-Boot > saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
```

13. Use the *print* command to check your environment.

```
U-Boot > print
baudrate=115200
bootargs=console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M
ip=dhcp mem=32M@0xc0000000 mem=64M@0xc4000000
bootcmd=sf probe 0; sf read ${kernaddr} ${kernoff} ${kernlen};sf read
${rootfsaddr} ${rootfsoff} ${rootfslen};bootm ${kernaddr}
bootdelay=3
bootfile=k2000.0
dnsip=10.1.2.138
dnsip2=10.1.2.139
ethact=DaVinci-EMAC
ethaddr=00:08:ee:03:5b:1e
fileaddr=C0700000
filesize=301325
ipaddr=10.0.5.65
kernaddr=0xc0700000
kerneladdr=0x00080000
kernlen=0x220000
kernoff=0x80000
netmask=255.255.252.0
rootfsaddr=0xC1180000
rootfslen=0x400000
rootfsoff=0x2a0000
serverip=10.1.2.114
stderr=serial
stdin=serial
stdout=serial
ver=U-Boot 2010.12 (Jul 10 2012 - 15:11:10)
```



```
Environment size: 735/65532 bytes
```

14. Power off the development kit.

8.3 Boot Linux Kernel

15. Power on the development kit. You will eventually see the output below.

```
_____                  _____          _         _
|  _  |___ ___ ___ ___   |  _  |___ ___  |_|___ ___| |_
|     |  _| .'| . | . |  |   __|  _| . | | | -_|  _|  _|
|__|__|_| |__,|_  |___|  |__|  |_| |___|_| |___|___|_|
              |___|                    |___|

Arago Project http://arago-project.org arago ttyS2

Arago 2009.09 arago ttyS2

arago login:
```

16. Enter root to log in to the Linux environment.

Congratulations, you have just configured your development kit to automatically boot Linux.

