DM3730/AM3703 U-Boot Labs

U-Boot Documentation (U-Boot version 2011.06)

Logic PD // Products
Published: October 2012
Last revised: September 2013

Abstract

These informal U-Boot labs provide an introduction to basic U-Boot commands and usage with the DM3730/AM3703 Torpedo System on Module (SOM), DM3730/AM3703 Torpedo + Wireless SOM, and DM3730/AM3703 SOM-LV.

Revision History

REV	EDITOR	DESCRIPTION	APPROVAL	DATE
A	SB, SWE	-Initial Release	SB, BSB	10/25/12
B	JA	-Section 2.3: Updated expected output from initial X-Loader / U-Boot load scrip in Step 4; -Section 2.5: Added note about change of LCD in development kits from 15 to 28; updated commands throughout for default LCD 28; -Section 3.2: Updated expected output from initial X-Loader / U-Boot load scrip in Step 4; -Section 5.3: Updated commands in Step 3 for version 2.4-2 of Linux BSP release; -Section 5.5: Updated bytes read in output of Step 11 and Step 18; updated expected output for successful NAND boot in Step 21; -Section 6.4: Updated expected output when viewing U-Boot environmental variables in Step 1; -Section 8.4: Updated IP address throughout; -Added Appendix B that explains how to identify each LCD	SO, SWE	09/09/13

Table of Contents

1 Introduction

These labs provide an informal introduction to the U-Boot bootloader software. The step-by-step directions act as a means to discover how U-Boot works with Logic PD's System on Modules (SOMs).

1.1 Nomenclature, Notices, and Conventions Used in this Document

- All labs are applicable to the DM3730/AM3703 SOM-LV, DM3730/AM3703 Torpedo SOM, and DM3730/AM3703 Torpedo + Wireless SOM. Use of "DM3730/AM3703 SOM" suggests text that applies to all three platforms; information specific to one platform will call out the precise name.
- Use of "DM3730 Development Kit" suggests text that applies to both the DM3730 SOM-LV Development Kit and DM3730 Torpedo Development Kit; information specific to one development kit will call out the precise name.
- Other files and documents are referenced throughout this document; a complete list of these documents, including links to access them, is available in Appendix A.

1.2 Software

This section provides a brief description of each software item addressed throughout this document.

- X-Loader – A pre-bootloader/second-stage bootloader. X-Loader is a low level pre-bootloader that is a stripped down version of U-Boot. The OMAP boot ROM loads X-Loader into SRAM to configure the pin muxing, clocks, and DDR of the processor in order for U-Boot to run from the DDR.
- U-Boot – A primary bootloader/third-stage bootloader. U-Boot runs from the DDR on the master CPU (CPU ID 0) and performs CPU and board-dependent initialization and configuration.
- Linux (kernel image, root file system) – The main operating system (OS). Linux is the last software to boot on the system and is responsible for managing the system’s resources and making them available to software running on the system.
- Virtual Machine (VM) – The Virtual Machine SDK for the DM37x Linux BSP (see Appendix A for a link) includes source for X-Loader, U-Boot, the Linux kernel, the LTIB build tool, and a pre-configured Ubuntu OS. The VM is configured to use VirtualBox as the VM manager. This option is the simplest way to get started building on nearly any OS.

1.3 U-Boot Background Information

- U-Boot is a universal, open-source bootloader for many embedded devices. Its primary function is to abstract the hardware-specific setup from the main OS, making it easier for the OS to install.
- Specifically, U-Boot initializes the hardware (especially the memory controller), provides boot parameters for the Linux kernel, and starts the Linux kernel.
- U-Boot also provides convenience features that simplify development, like reading and writing memory locations, uploading new binary images to the board’s RAM via a serial line or Ethernet, and copying binary images from RAM to flash memory. For more information and general documentation, please see the U-Boot website.[1]

[1] http://www.denx.de/wiki/U-Boot/

2 Lab 1: Kit Communications

This lab will demonstrate how to establish a serial connection to your DM3730 Development Kit. We will use two types of kit communications: serial transmission through RS232 and USB.

2.1 Prerequisites

- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- A host PC or laptop
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- Terminal emulation program (e.g., Tera Term)

2.2 Install Terminal Emulation Program

In these labs we will use Tera Term to send and receive serial communications with the DM3730 Development Kit. Tera Term is a terminal emulation program that can be downloaded for free from Logic PD's website. To install Tera Term:

1. Download the ZIP file[2] from Logic PD's website and extract the contents.
2. After extracting the contents, locate the *teraterm-x.xx.exe* file and double-click it.
3. Follow the on-screen instructions to install Tera Term.

2.2.1 Set Up Tera Term

Because U-Boot is a program that you interact with, it needs a way to receive information from you so it can accomplish the tasks you want it to do. The manner in which a program receives input from users has traditionally been called standard input or *stdin*. On a desktop PC, you are probably accustomed to using a keyboard and a mouse to interact with programs. The OS (e.g., Microsoft Windows or Linux equivalent) collects the characters you type and information about where you moved and clicked your mouse and passes them on to the program you are using. With U-Boot, there isn't a mouse or a keyboard; instead, U-Boot collects the characters you send over your serial port via Tera Term, the terminal emulation program, and treats it as standard input.

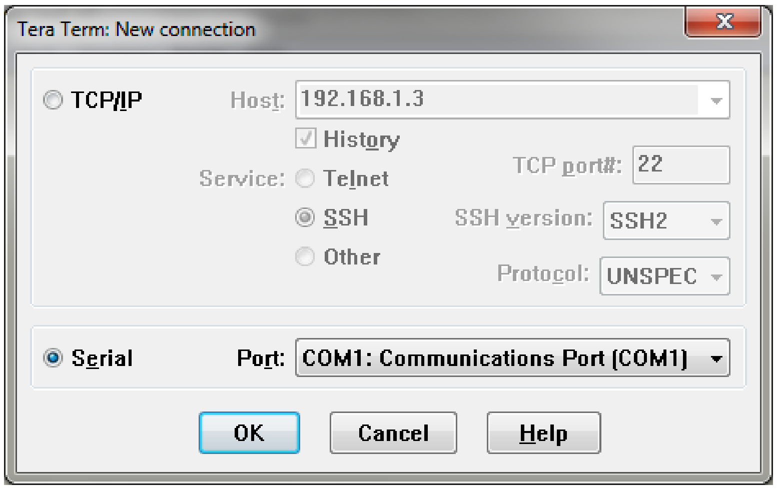
NOTE: As a quick aside, standard input has a partner in crime named standard output or *stdout*. On your host PC, standard output is usually the display or monitor attached to the PC. On a DM3730 Development Kit, standard output is also the kit's serial port. When U-Boot has anything to say to you, it talks to you on its standard output.

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).

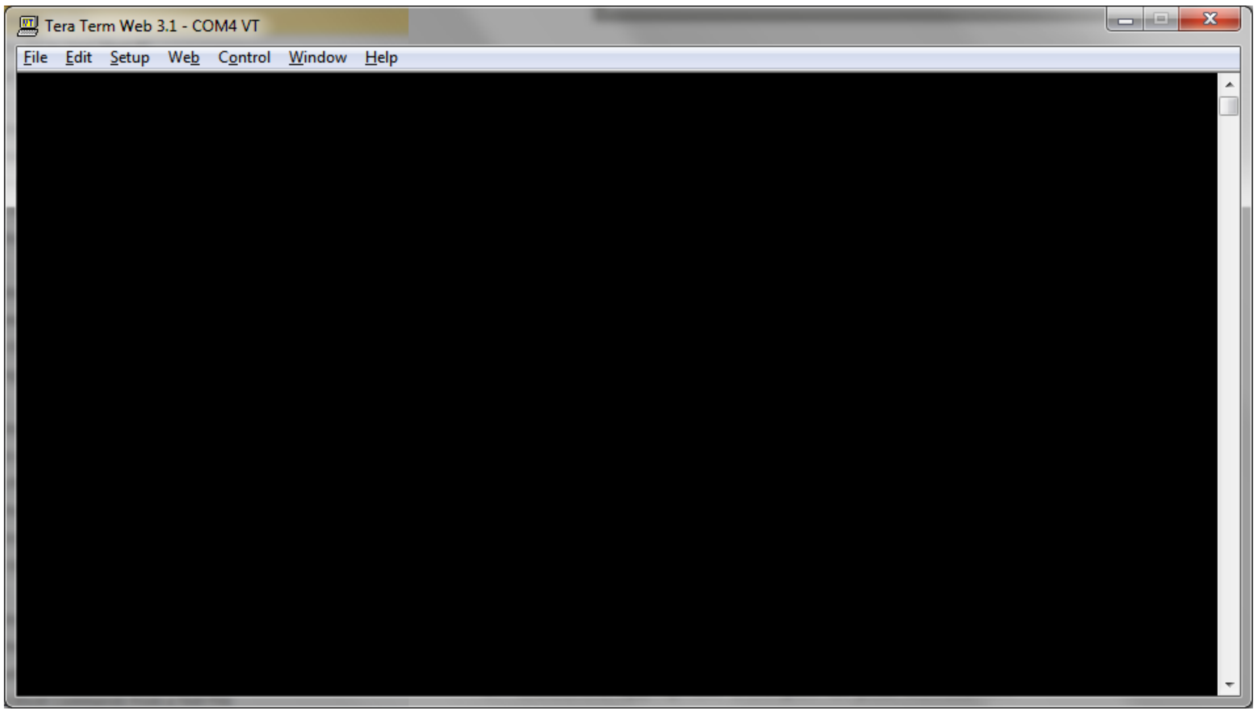
1. Start the Tera Term program.

[2] http://support.logicpd.com/downloads/240/

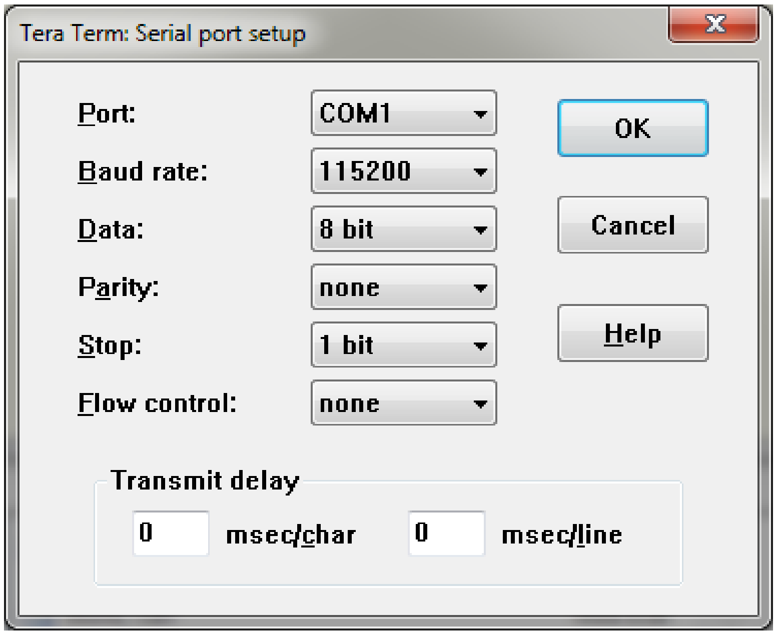
2. In the *Tera Term: New connection* window, make sure the "Serial" option is selected and choose the appropriate COM port to which your development kit is connected. Click OK.



A window similar to the one included below should open.



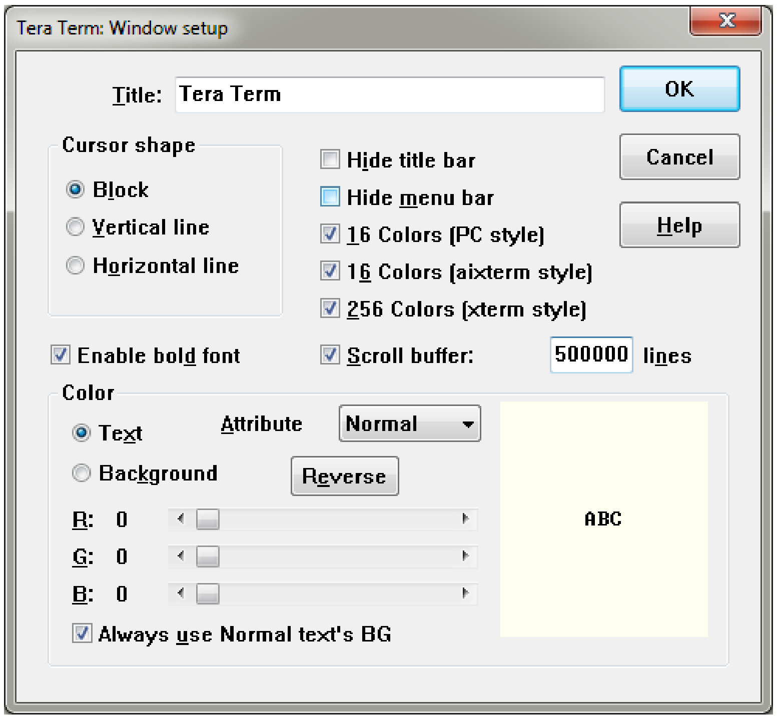
3. From the menu toolbar, select Setup > Serial port.
4. Select the appropriate COM port for your workstation.
5. Change the port settings to:
   a. Baud rate: **115200**
   b. Data: **8 bit**
   c. Parity: **None**
   d. Stop: **1 bit**
   e. Flow control: **None**



Port: COM1
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

6. Click OK.
7. Select Setup > Window.
8. Ensure the "Scroll buffer" box is selected and make the amount as big as you like, up to 500000.



Title: Tera Term

Cursor shape
● Block
○ Vertical line
○ Horizontal line

☐ Hide title bar
☐ Hide menu bar
✓ 16 Colors (PC style)
✓ 16 Colors (aixterm style)
✓ 256 Colors (xterm style)

✓ Enable bold font
✓ Scroll buffer: 500000 lines

Color
● Text Attribute Normal
○ Background
R: 0
G: 0
B: 0
✓ Always use Normal text's BG

9. Click OK.

2.3 Connect Kit with Serial Port

1. Use the serial cable included in the DM3730 Development Kit to connect the serial port on the kit to the appropriate port on your host PC.
2. Insert the DM37x Linux Demo Image SD card included in the DM3730 Development Kit into the bootable card slot on the baseboard.
3. Start the Tera Term program.
4. Power on the DM3730 Development Kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS. You should see the output of the initial X-Loader/U-Boot load scripts similar to that included below.

   NOTE: The default boot script for the DM37x Linux Demo Image included on the SD card with the development kit will require you to hit any key if you want to abort autobooting fully into Linux within five seconds.

```
Texas Instruments X-Loader 1.42 BSP-dm37x-2.4-2 for dm3730logic (2013-
06-18 14:09:10)
DRAM: 256MB (ProductID defined)
Starting U-boot on MMC
Reading boot sector
454540 bytes read from MMC to 80400000


U-Boot 2011.06 BSP-dm37x-2.4-2 (Jun 18 2013 - 13:57:32)

OMAP3630/3730-GP ES2.1, CPU-OPP2, L3-200MHz, Max CPU Clock 1 Ghz
Logic DM37x/OMAP35x reference board + LPDDR/NAND
I2C:   ready
DRAM:  256 MiB
Board: DM37xx Torpedo
Enable battery backup charger (BB_CFG = 0x1c)
NAND:  512 MiB
NAND: Internal to NAND ECC selected
*** Warning - bad CRC or ECC error, using default environment

Found '28' display panel
In:    serial
Out:   serial
Err:   serial
Product ID data cached to: 40200000
Die ID #5f0a00029ff800000168301018012022
OTG_SYSCONFIG: 00001008 - needs to be 00002001

ID data ROM  : Gen 2
Model        : dm3730 Torpedo
Version Code : -30
Temp Grade   : Commercial
Max Speed    : 1000MHz
Part Number  : 1021711
Model Name   : SOMDM3730-30-2780AKCR-A
Serial Number: 3312M00948

Data (writethrough) Cache is ON
Net:   smc911x-0
```

```
=================================== NOTICE
===================================
The U-Boot environment was not found. If the display is not set
properly
linux will not have video support.

Valid display options are:
  2 == LQ121S1DG31     TFT SVGA    (12.1)  Sharp
  3 == LQ036Q1DA01     TFT QVGA    (3.6)   Sharp w/ASIC
  5 == LQ064D343       TFT VGA     (6.4)   Sharp
  7 == LQ10D368        TFT VGA     (10.4)  Sharp
 15 == LQ043T1DG01     TFT WQVGA   (4.3)   Sharp
 28 == LQ043T1DG28     TFT WQVGA   (4.3)   Sharp (DEFAULT)
 vga[-16 OR -24]       LCD VGA     640x480
 svga[-16 OR -24]      LCD SVGA    800x600
 xga[-16 OR -24]       LCD XGA     1024x768
 720p[-16 OR -24]      LCD 720P    1280x720
 sxga[-16 OR -24]      LCD SXGA    1280x1024
 uxga[-16 OR -24]      LCD UXGA    1600x1200

Default `display` environment variable is now being set to: 28

At the U-Boot prompt type commands: `setenv display <num>`, then type
`saveenv` to save the environment to NAND flash.  This will avoid
seeing
this notice on future boots
=================================== NOTICE
===================================

Hit any key to stop autoboot:  0
OMAP Logic #
```

NOTE: If you do not see the U-Boot script output, please check to make sure the serial cable is connected correctly and the SD card is fully inserted. If you still do not see the U-Boot welcome prompt, the U-Boot files may not be loaded properly on the SD card. Please refer to Section 5 for instructions on how to update and load U-Boot.

2.4 Connect Kit with USB Port

The USB port is an optional serial interface. Feel free to skip to the next section if you already have serial communication with your terminal emulator.

The DM3730 Development Kit baseboard is equipped with an FTDI Virtual COM Port (VCP) chip that causes the USB device to appear to your computer as an additional COM port. Settings for the terminal emulation program will remain the same; however, a driver must be installed on your computer for proper operation.

1. First, review the *USB to UART ReadMe* file (see Appendix A for a link) for instructions on how to use the USB-to-UART VCP chip on the development kit baseboard. When complete, download and install the appropriate driver.
2. Connect the USB A to mini-B cable to the debug USB port on the baseboard and to a USB port on your PC.
3. Insert the DM37x Linux Demo Image SD card into the bootable card slot on the baseboard.
4. Start the Tera Term program.

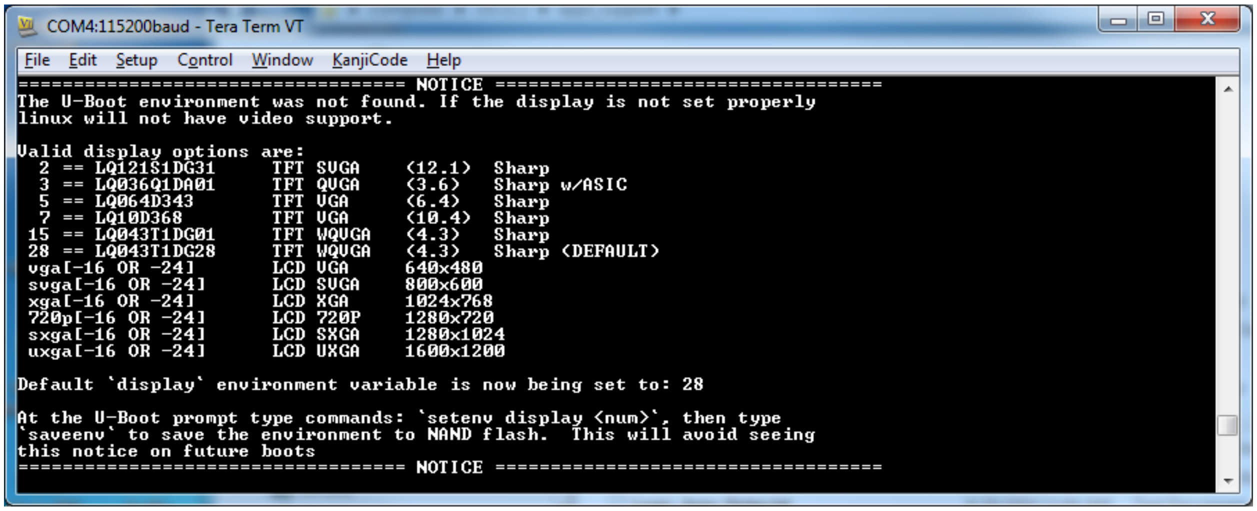
5. Power on the development kit; when prompted in the terminal, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS.

   NOTE: If you do not see the U-Boot output, please check to make sure the serial cable is connected correctly and the SD card is fully inserted. If you still do not see the U-Boot welcome prompt, the U-Boot files may not be loaded properly on the SD card. Please refer to Section 5 for instructions on how to update and load U-Boot.

2.5 Set Environment Variable *display*

NOTE: In Q3 of 2013, Logic PD updated the LCD panel in the 4.3" Display Kit. This changed the default display from 15 to 28. Many of the kits in circulation still have the older 4.3" display. Please see Appendix B for information about how to identify which display you have.

1. Initially, when you reach the U-Boot prompt and do not have the *display* variable set, you will see the following output:


COM4:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
=================================== NOTICE ===================================
The U-Boot environment was not found. If the display is not set properly
linux will not have video support.

Valid display options are:
  2 == LQ121S1DG31     TFT SVGA    (12.1)  Sharp
  3 == LQ036Q1DA01     TFT QVGA    (3.6)   Sharp w/ASIC
  5 == LQ064D343       TFT VGA     (6.4)   Sharp
  7 == LQ10D368        TFT VGA     (10.4)  Sharp
 15 == LQ043T1DG01     TFT WQVGA   (4.3)   Sharp
 28 == LQ043T1DG28     TFT WQVGA   (4.3)   Sharp (DEFAULT)
 vga[-16 OR -24]       LCD VGA     640x480
 svga[-16 OR -24]      LCD SVGA    800x600
 xga[-16 OR -24]       LCD XGA     1024x768
 720p[-16 OR -24]      LCD 720P    1280x720
 sxga[-16 OR -24]      LCD SXGA    1280x1024
 uxga[-16 OR -24]      LCD UXGA    1600x1200

Default `display` environment variable is now being set to: 28

At the U-Boot prompt type commands: `setenv display <num>`, then type
`saveenv` to save the environment to NAND flash.  This will avoid seeing
this notice on future boots
=================================== NOTICE ===================================
```

2. To set the *display* environment variable to support your display, use the output in the terminal window to determine which number corresponds to your display. This example will use display 28, which is the default setting.

3. To update the environment variable *display*, use the following command at the prompt.

```
OMAP Logic # setenv display 28
```

4. Now check the *display* environment variable.

```
OMAP Logic # printenv display
```

5. You should see the following output:

```
display=28
```

6. Now, the current environment is only in RAM, so it will need to be saved to persistent NAND memory in-order for it to be loaded the next time the system is powered on. To do this, use the command below.

```
OMAP Logic # saveenv
```

You should see the following output:

```
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x260000 -- 100% complete.
Writing to Nand... done
```

Now, you have successfully saved your U-Boot environment into persistent NAND memory. In later sections, we will make further modifications to the U-Boot environment.

3 Lab 2: Learning U-Boot Help System

This lab will teach you the basics of working with the U-Boot built-in help system.

3.1 Prerequisites

- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- A host PC or laptop
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- A terminal emulation program (e.g., Tera Term) with the following port settings:

  a. Baud rate: **115200**
  b. Data: **8 bit**
  c. Parity: **None**
  d. Stop: **1 bit**
  e. Flow control: **None**

3.2 U-Boot Interface

1. Use the serial cable or USB A to mini-B cable included in the DM3730 Development Kit to connect the serial port on the kit to the appropriate port on your host PC.
2. Insert the DM37x Linux Demo Image SD card into the bootable card slot on the baseboard.
3. Start the Tera Term program.
4. Power on the development kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS. You should see the following output of the initial X-Loader/U-Boot load scripts:

```
Texas Instruments X-Loader 1.42 BSP-dm37x-2.4-2 for dm3730logic (2013-
06-18 14:09:10)
DRAM: 256MB (ProductID defined)
Starting U-boot on MMC
Reading boot sector
454540 bytes read from MMC to 80400000


U-Boot 2011.06 BSP-dm37x-2.4-2 (Jun 18 2013 - 13:57:32)

OMAP3630/3730-GP ES2.1, CPU-OPP2, L3-200MHz, Max CPU Clock 1 Ghz
Logic DM37x/OMAP35x reference board + LPDDR/NAND
I2C:   ready
DRAM:  256 MiB
Board: DM37xx Torpedo
Enable battery backup charger (BB_CFG = 0x1c)
NAND:  512 MiB
NAND: Internal to NAND ECC selected
Found '28' display panel
In:    serial
Out:   serial
Err:   serial
Product ID data cached to: 40200000
Die ID #5f0a00029ff80000016830101801202 2
OTG_SYSCONFIG: 00001008 - needs to be 00002001

ID data ROM  : Gen 2
```

```
Model        : dm3730 Torpedo
Version Code : -30
Temp Grade   : Commercial
Max Speed    : 1000MHz
Part Number  : 1021711
Model Name   : SOMDM3730-30-2780AKCR-A
Serial Number: 3312M00948

Data (writethrough) Cache is ON
Net:   smc911x-0
Hit any key to stop autoboot:  0
OMAP Logic #
```

NOTE: The U-Boot output above tells us that the development kit is running version 2011.06 of the U-Boot software. This version of U-Boot also provides the SOM model number, part number and serial number among various other configuration information of the hardware. This allows you to send one screen shot to Technical Support and have all of the information in one place.

You can also type *version* on the command line to get the monitor, compiler and linker versions used to compile U-Boot.

3.3 Help for Incorrect Commands

At the prompt, type the following command:

```
OMAP Logic # something
```

You should receive the following output:

```
Unknown command 'something' - try 'help'
OMAP Logic #
```

That was a disappointment. Of course, this is because there is no actual command named *something* in the U-Boot program. However, let's take a quick look at what was printed out.

After we pressed Enter, U-Boot displayed the following string:

```
unknown command 'something' - try 'help'
```

U-Boot tells us *something* is an unknown command. This informs us that U-Boot couldn't find a command named *something* to execute.

Now that we have established that there isn't a command named *something*, let's find out what commands are included in U-Boot.

3.4 Using *help* Command

1. At the prompt, enter the *help* command.

```
OMAP Logic # help
```

You should see the following output:

```
sdrc_config- sdrc_config - dump SDRC registers
setenv  - set environment variables
setexpr - set environment variable as the result of eval expression
showvar - print local hushshell variables
sleep   - delay execution for some time
source  - run script from memory
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
time    - time execution of command
true    - do nothing, successfully
version - print monitor, compiler and linker version
ydf     - YAFFS disk free
ydump   - YAFFS device struct
yls     - YAFFS ls
ymkdir  - YAFFS mkdir
ymount  - YAFFS mount
ymv     - YAFFS mv
yrdm    - YAFFS read file to memory
yrm     - YAFFS rm
yrmdir  - YAFFS rmdir
yumount - YAFFS unmount
ywrm    - YAFFS write file from memory
```

Supplying the *help* command without an argument, lists the name of each command built into the version of U-Boot running on your development kit, along with a brief description of what the command does.

2. At the prompt, enter the command below.

```
OMAP Logic # help coninfo
```

You should see the following output:

```
coninfo – print console devices and information


Usage:
Coninfo
```

Here, you see the functionality of the *coninfo* command, in addition to how it is used (without any arguments).

3. At the prompt, enter the *coninfo* command by itself.

```
OMAP Logic # coninfo
```

You should see the following output:

```
List of available devices:
lcd      00000002 ..O
serial   80000003 SIO stdin stdout stderr
nulldev  80000003 SIO
```

Issuing the *coninfo* command gives you the list of connections for *stdin*, *stdout* and *stderr*. It is a simple command that takes no arguments. How about some of the more complex commands?

4. At the prompt, enter the command below.

```
OMAP Logic # help env
```

You should see the following output:

```
env - environment handling commands

Usage:
env default -f - reset default environment
env edit name - edit environment variable
env export [-t | -b | -c] addr [size] - export environment
env import [-d] [-t | -b | -c] addr [size] - import environment
env print [name ...] - print environment
env run var [...] - run commands in an environment variable
env save - save environment
env set [-f] name [arg ...]
```

Again, by using *env* as the *help* command's argument, we've asked the *help* command to print the help topic for a specific U-Boot command.

We've just learned two important things about U-Boot's *help* command:

- If you have completely forgotten how to use U-Boot, just type *help* at the `OMAP Logic #` prompt. This provides a listing of all U-Boot commands.
- The *help* command will display the help topic for a single command. If you know a command's name but forgot how to use it, just use the *help* command followed by the name of the command and you will see the help topic for that command.

Before we continue, let's take a closer look at the *env* command's help topic. Notice the command's usage string, reprinted below.

```
Usage:
env default -f - reset default environment
env edit name - edit environment variable
env export [-t | -b | -c] addr [size] - export environment
env import [-d] [-t | -b | -c] addr [size] - import environment
env print [name ...] - print environment
env run var [...] - run commands in an environment variable
env save - save environment
env set [-f] name [arg ...]
```

The *env* command gives us many options of usage, since it is central to how U-Boot functions. The *env* command's usage string tells you that you must type the command with an argument (e.g., default, edit, export) based on what you are trying to accomplish. Arguments that may be optionally used with a command are listed in square brackets.

The most important command above is the *env default -f* command, which resets all the environmental variables stored in RAM. This will be very useful when experimenting with U-Boot and switching among OSs. We will be exploring the *env* command in greater detail in a later section.

NOTE: The U-Boot environment is loaded from persistent storage into RAM when the system boots. If you make any changes to the environment, these changes only exist in RAM and need to be saved to persistent storage (NAND) to take effect the next time the system boots.

4 Lab 3: Using U-Boot Command Line Editor

This lab will teach you how to work more efficiently with U-Boot by learning its command line editing features.

4.1 Prerequisites

- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- A host PC or laptop
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- A terminal emulation program (e.g., Tera Term) with the following port settings:

  a. Baud rate: **115200**
  b. Data: **8 bit**
  c. Parity: **None**
  d. Stop: **1 bit**
  e. Flow control: **None**

4.2 Basic Commands

1. At the prompt, enter the help command.

```
OMAP Logic # help
```

You should see the following output:

```
sdrc_config- sdrc_config - dump SDRC registers
setenv  - set environment variables
setexpr - set environment variable as the result of eval expression
showvar - print local hushshell variables
sleep   - delay execution for some time
source  - run script from memory
test    - minimal test like /bin/sh
tftpboot- boot image via network using TFTP protocol
time    - time execution of command
true    - do nothing, successfully
version - print monitor, compiler and linker version
ydf     - YAFFS disk free
ydump   - YAFFS device struct
yls     - YAFFS ls
ymkdir  - YAFFS mkdir
ymount  - YAFFS mount
ymv     - YAFFS mv
yrdm    - YAFFS read file to memory
yrm     - YAFFS rm
yrmdir  - YAFFS rmdir
yumount - YAFFS unmount
ywrm    - YAFFS write file from memory
```

As discussed in Lab 2, the *help* command requires an argument. Therefore, just typing *help* prints out the commands usage string to remind us how the command should look.

Let’s pretend that we wanted to use the *help setenv* command because we wanted to see all of the commands that relate to setting an environment variable. We could just type *help setenv* at the prompt and press Enter. But, there is an easier way.

2. At the prompt, press the up-arrow key on your keyboard. You should see the following output:

```
OMAP Logic # help
```

Notice how the *help* command reappears. The up-arrow cycles through the last nineteen commands that you have typed. Type a few more commands and then use the up-arrow and Enter key to find and repeat them.

The down-arrow works in the opposite way of the up-arrow. Using the arrows can save you quite a bit of typing if you need to repeat a command often. But what if you typed the command wrong to begin with?

3. At the prompt, enter the command below.

```
OMAP Logic # help draw_test
```

You should see the following output:

```
draw_test -  - Draw ramps/stipples/boarders on LCD

Usage:
draw_test
```

The draw_test command is used to initialize U-Boot's display and draw a test pattern.

4. At the prompt, enter the command below.

```
OMAP Logic # cls
```

5. Then, enter the command below.

```
OMAP Logic # echo_lcd Logic PD Rocks!!!
```

You should see the text that you entered on the LCD screen.

For those who are familiar with traditional terminal emulators, or for those using a program that does not properly send Backspace and arrow key presses, see the series of key combinations below that provide additional options to move around a command line.

- **Ctrl+A** - Move to the beginning of the line
- **Ctrl+E** - Move to the end of the line
- **Ctrl+B** - Move backward (to the left) one character
- **Ctrl+F** - Move forward (to the right) one character
- **Ctrl+D** - Delete the character under the cursor

Go ahead and take some time to familiarize yourself with the above key combinations and command line editing.

4.3 Using Escape Character

Sometimes we will want to escape characters entered at the U-Boot prompt. Usually we will do this to delay U-Boot from doing something when we are entering U-Boot commands into an environment variable. We will learn to work with environment variables in a later lab; for now, we will explore how to use the escape character (\ ) to manipulate how U-Boot interprets characters.

In U-Boot, the escape character is the backslash (\ ). When U-Boot sees this character, it interprets the character immediately after as string data.

First, let's try a few commands without the escape character.

1. At the prompt, enter the command below.

```
OMAP Logic # echo Hello World!!!
```

   You should see the following output:

```
Hello World!!!
```

2. Next, enter the command below.

```
OMAP Logic # echo Hello              World!!!
```

   You should see the same output you saw in Step 1. What happened? U-Boot sent the *echo* command each string that was on *stdin*; it did not consider the space character as part of the strings. The *echo* command inserted a space between the strings when it printed them on *stdout*. U-Boot has to look for the space character on the *stdin* so it knows where the beginning and end of commands and arguments are.

3. If we want U-Boot to send the *echo* command the space characters we entered, then we have to tell U-Boot that the space characters are a part of the string data that it is supposed to send to the *echo* command. To see how the escape character is used to do this, enter the command below at the prompt.

```
OMAP Logic # echo Hello\ \ \ \ \ World!!!
```

   You should see the following output:

```
Hello     World!!!
```

   The escape character is useful in many other situations. For example, when you want to add multiple lines of text at the prompt, but you want U-Boot to treat it as one line.

5 Lab 4: Update U-Boot and Make it Bootable from NAND

This lab will teach you how to download a new version of U-Boot and store it in the NAND flash of your DM3730 Development Kit. In the event that the development kit's memory gets corrupted, please follow these instructions for the recovery procedure as well.

5.1 Prerequisites

- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- Virtual Machine SDK for the DM37x Linux BSP (see Appendix A for a link)
- A host PC or laptop
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- A terminal emulation program (e.g., Tera Term).

5.2 Download Latest Version of U-Boot

For this lab, you need to download the Virtual Machine SDK for the DM37x Linux BSP. If you have not installed the VM, please refer to the *Virtual Machine SDK for the DM37x Linux BSP ReadMe* (see Appendix A for a link) before performing the VM installation.

1. Boot into the Virtual Machine SDK for the DM37x Linux BSP using the password logic.
2. Inside the VM, log into the Logic PD support site[3] using your username and password.
3. Click on the **All Downloads** link under the DM3730 Development Kit heading for your kit.
4. Scroll down until you find the section header *Linux*.
5. Click on the **+** icon next to the *DM37x Linux SD Card Demo Image* link; this expands the list of all available versions. If the current version is newer than what is on your SD card, you should update it.
6. Click on the version number marked *(current)* and save the file to your host PC in the */home/logic/logic* directory. This ZIP file contains all the files necessary to install and use U-Boot with Linux.

A brief description of the included file formats and files is listed below.

- *mkLogicFATcard.sh* – This is a Linux script to format and write a bootable partition on an SD card.
- *MLO* - This file is X-Loader, the second-stage bootloader that runs from SRAM. The CPU boot ROM, which is the first-stage bootloader, requires that the file is named *MLO* in order to load it upon power up or after CPU reset. The purpose of X-Loader is to initialize the DDR and load the third-stage bootloader. X-Loader can be placed on an SD card or in NAND flash, but must reside on the medium where the U-Boot image resides.
- *rootfs.ext2.gz.uboot* – This is a RAM-based Linux root file system that is required by the Linux OS. It is loaded by U-Boot at boot time. Any data added to the RAM-based file system will be lost between power cycles.
- *rootfs.yaffs2* – This is a NAND-based root file system that is specifically formatted to be loaded on to the NAND flash device. Using a NAND-based root file system allows the entire RAM to be used for the Linux kernel. Any data stored on the NAND-based root file system is persistent across power cycles.
- *rootfs-dsp.yaffs2* – This is the same NAND-based root file system as above; however, this image includes all the files necessary to support the DSP.

[3] http://support.logicpd.com/auth/

- *u-boot.bin* - This file is U-Boot, the third-stage bootloader. U-Boot supports shell scripting, basic debugging, and is used for CPU-dependent initialization. U-Boot itself can be placed on an SD card or in NAND flash, but must reside on the medium where the X-Loader image resides.
- *u-boot.bin.ift* - This file is the same as the *u-boot.bin* file above; however, it includes a header that X-Loader uses to identify the size of the file. This is used for NAND flash when there is not a file system X-Loader can query to identify the size of the U-Boot image.
- *uImage* – This is the Linux kernel. It is loaded by U-Boot with the root file system at boot time.
- *uImage-dsp* – This is a Linux kernel that is similar to the one above; however, it includes the changes necessary to support the DSP.

Now that you have downloaded the latest version of U-Boot and the DM37x Linux SD Card Demo Image to your computer, you can update U-Boot on your DM3730 Development Kit using a bootable SD card.

5.3 Prepare SD Card to Boot U-Boot

1. Boot into the VM SDK for the DM37x Linux BSP.
2. Connect the SD card to your host PC using the USB SD card reader included in your kit.
3. Unzip the DM37x Linux SD Card Demo Image ZIP file.

```
$ unzip <1024334_LogicPD_Linux_BSP_2.4-2_SD_Card_Demo_Image.zip> -d
LogicPD_Linux_BSP_2.4-2_SD_Card_Demo_Image

$ cd _LogicPD_Linux_BSP_2.4-2_SD_Card_Demo_Image
```

NOTE: Remember that formatting an SD card will erase its entire contents. Be sure to back up the data on the SD card before continuing.

4. The boot ROM within the Texas Instruments (TI) processor requires a specific boot partition for a bootable SD card. The SD card can be created using the steps below.
   a. List the hard drives and partitions that the host system has mounted, using the command below.

```
$ sudo fdisk -l
```

   b. Find the SD card in the list given in the output. It should appear as something similar to */dev/sdx,* where x is a letter representing the drive. Example output is shown below.

```
Disk /dev/sdb: 4035 MB, 4035969024 bytes
```

NOTE: Before running *mkLogicFATcard.sh,* make sure that the write protect switch is not enabled on your SD card. If the write protect is enabled, you will not be notified by the script that the card did not format correctly.

c. The *mkLogicFATcard.sh* is a script that will format your SD card so that it is prepared to be used as a boot device for OMAP processors. To run it, use the command below at your Linux terminal prompt.

```
$ ./mkLogicFATcard.sh
```

You should see the following output:

```
sdb is 3965190144 bytes - Card  Reader
sr0 is 50780160 bytes - CD-ROM
Enter device:
```

d. Where it says, `Enter device:`, type the device name (e.g., sdb) and then press **ENTER**.

Now you should see the following output:

```
All the data on device ‘/dev/sdb’ will be removed!
Do you wish to continue? (y/N)
```

e. Answer yes when it asks if you would like to continue.

```
$ y
```

Again, you will see the following output:

```
Formatting /dev/sdb...
Unmounting /dev/sdb1
Disk /dev/sdb doesn't contain a valid partition table
DISK SIZE - 3965190144 bytes
CYLINDERS - 482
[sudo] password for logic:
```

f. Enter the password *logic*.

```
$ logic
```

g. Once the script is finished, remove the SD card and reinsert it. This will ensure the SD card is remounted properly.

5. Now that the SD card has been properly prepared, use the *cd* command to navigate to the directory where the *MLO, u-boot.bin, rootfs.ext2.gz.uboot,* and *uImage* files reside.

6. Copy only the *MLO* file to the SD card. This is a required step by TI’s internal boot ROM.

```
$ cp MLO /media/boot/
```

7. Next, copy the *uboot.bin*, *rootfs.ext2.gz.uboot,* and *uImage* files to the SD card.

```
$ cp u-boot.bin u-boot.bin.ift rootfs.ext2.gz.uboot uImage /media/boot/
```

8. Remove the SD card from your host PC.

You are now ready to boot U-Boot from the SD card.

5.4 Connect Kit to Host PC

1. Use the serial cable or USB A to min-B cable included in the development kit to connect the serial port on the kit to the appropriate port on your host PC.
2. Insert the DM37x Linux Demo Image SD card into the bootable card slot on the baseboard.
3. Start the Tera Term program.
4. Power on the development kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting the OS. You should see the following output:

```
Hit any key to stop autoboot:  0
OMAP Logic #
```

NOTE: If you do not see the U-Boot script output, please check to make sure the serial cable is connected correctly and the SD card is fully inserted. If you still do not see the U-Boot welcome prompt, the U-Boot files may not be loaded properly on the SD card. Please refer to Section 5 for instructions on how to update and load U-Boot.

5.5 Make U-Boot Bootable from NAND

1. First, erase the entire NAND flash.

```
OMAP Logic # nand erase.chip
```

You should see the following output:

```
NAND erase.chip: device 0 whole chip
Erasing at 0x1ffe0000 -- 100% complete.
OK
```

2. Next, set up the default environment variables. These are set when the Linux kernel is compiled.

```
OMAP Logic # env default –f
```

You should see the following output:

```
## Resetting to default environment
```

3. List the partition table.

```
OMAP Logic # mtdparts
```

You should see the following output:

```
device nand0 <omap2-nand.0>, # parts = 6
 #: name                size            offset          mask_flags
 0: x-loader            0x00080000      0x00000000      0
 1: u-boot              0x001a0000      0x00080000      0
 2: u-boot-env          0x00060000      0x00220000      0
 3: kernel              0x00500000      0x00280000      0
 4: ramdisk             0x01400000      0x00780000      0
 5: fs                  0x1e480000      0x01b80000      0

device nor0 <physmap-flash.0>, # parts = 1
 #: name                size            offset          mask_flags
 0: nor                 0x00800000      0x00000000      0

active partition: nand0,0 - (x-loader) 0x00080000 @ 0x00000000

defaults:
mtdids  : nand0=omap2-nand.0,nor0=physmap-flash.0
mtdparts: mtdparts=omap2-nand.0:512k(x-loader),1664k(u-boot),384k(u-
boot-env),5m(kernel),20m(ramdisk),-(fs);physmap-flash.0:-(nor)
```

The output above shows that there is 1664 kibibytes allocated for the U-Boot partition. Alternatively, this could be considered as 832 pages or 0x1a0000 bytes. The page size for the NAND chip included with the DM3730/AM3703 SOM is 2048 bytes. Each partition needs to be large enough to fit each corresponding file rounded up to the nearest page size. The size of the *u-boot.bin* file in this case is 427616 bytes. Of course the size of U-Boot may change, so please check this before proceeding.

4. To check if the defined partition is large enough, we integer divide the size of U-Boot by the page size and add one. According to this calculation, we will need 209 pages or 0x68800 bytes.

   Similarly, we can look at the partition where X-Loader will be stored. For the OMAP processors, the first four blocks in NAND must contain X-Loader. For this case, X-Loader is 44760 bytes and according to the calculation defined above, the X-Loader partition will need at least 22 pages or 0xB000 bytes. It is important to note that the X-Loader partition needs to be four blocks. For the NAND chip in the DM3730 Development Kit, each block contains 64 pages which is 0x20000 bytes; this is much larger than 0xB000 bytes. This is okay because we will use one whole block for each copy of X-Loader.

5. Now that we have checked that the partitions are sized correctly, we are ready to define a space in RAM that will allow us to easily program the NAND. To start, use the *bdinfo* command to list information on available memory locations.

```
OMAP Logic # bdinfo
```

You should see the following output:

```
arch_number = 0x00000CB1
boot_params = 0x80000100
```

```
DRAM bank   = 0x00000000
-> start    = 0x80000000
-> size     = 0x10000000
DRAM bank   = 0x00000001
-> start    = 0xA0000000
-> size     = 0x00000000
ethaddr     = 00:08:ee:04:e6:5a
ip_addr     = 0.0.0.0
baudrate    = 115200 bps
TLB addr    = 0x8FFF0000
relocaddr   = 0x8F806000
reloc off   = 0x0F406000
data_start  = 0x8F85F4D8
rodata_start= 0x8F84F0F8
bss_start   = 0x8F865F18
irq_sp      = 0x8F1C5F60
sp start    = 0x8F1C5F50
FB base     = 0x8F89D000
```

The output shows that we have RAM starting at address 0x80000000 and that it has size 0x10000000.

6. Set the environment variable *loadaddr* to point to 0x81000000.

```
OMAP Logic # setenv loadaddr 0x81000000
```

7. Check which value got stored.

```
OMAP Logic # env print loadaddr
```

You should see the following output:

```
loadaddr=0x81000000
```

8. Save the environment variable to NAND.

```
OMAP Logic # saveenv
```

You should see the following output:

```
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x260000 -- 100% complete.
Writing to Nand... done
```

9. Next, initialize the SD card.

```
OMAP Logic # mmc init
```

You should see the following output:

```
mmc1 is available
```

10. Now that we have initialized the SD card and *loadaddr* is pointing to an available region in RAM, we are ready to prepare our space in RAM to successfully make a store into NAND. Empty space in NAND must contain 0xFF; so we will first set all the bits in our space in RAM.

```
OMAP Logic # mw ${loadaddr} 0xffffffff
OMAP Logic # mw.l ${loadaddr} 0xffffffff 0x400000
```

Using the memory write command twice is necessary to make a successful write to RAM.

11. Copy *MLO* into available space in RAM.

```
OMAP Logic # fatload mmc 1 ${loadaddr} mlo
```

You should see the following output:

```
reading mlo
46372 bytes read
```

12. Before storing X-Loader in NAND, we first need to set the ECC algorithm. The OMAP boot ROM expects X-Loader to be written to the first four blocks of NAND using the hardware ECC. So, set the ECC algorithm to hardware.

```
OMAP Logic # nandecc hw
```

You should see the following output:

```
NAND: HW ECC selected
```

13. Now, burn the first block of NAND with the *MLO* image.

```
OMAP Logic # nand write ${loadaddr} 0x00000000 0x00020000
```

You should see the following output:

```
NAND write: device 0 offset 0x0, size 0x20000
 131072 bytes written: OK
```

14. Similarly, burn the second block of NAND with the *MLO* image. This time we will need to make our offset equal to one block in bytes. Remember, one block is 0x20000 bytes.

```
OMAP Logic # nand write ${loadaddr} 0x00020000 0x00020000
```

You should see the following output:

```
NAND write: device 0 offset 0x20000, size 0x20000
 131072 bytes written: OK
```

15. Again, add 0x00020000 to the previous offset and burn the third block in NAND.

```
OMAP Logic # nand write ${loadaddr} 0x00040000 0x00020000 Enter
```

You should see the following output:

```
NAND write: device 0 offset 0x40000, size 0x20000
 131072 bytes written: OK
```

16. Finally, add 0x00020000 to the offset and burn the fourth block in NAND.

```
OMAP Logic # nand write ${loadaddr} 0x00060000 0x00020000
```

You should see the following output:

```
NAND write: device 0 offset 0x60000, size 0x20000
 131072 bytes written: OK
```

17. Now that we have written the first four blocks of our NAND device with the *MLO* image, we are ready to prepare our space in RAM again to store the *u-boot.bin* file in NAND.

```
OMAP Logic # mw.l ${loadaddr} 0xffffffff 0x400000
```

18. Load the *u-boot.bin* file into RAM.

```
OMAP Logic # fatload mmc 1 ${loadaddr} U-Boot.bin
```

You should see the following output:

```
reading U-Boot.bin

454540 bytes read
```

19. Change the ECC algorithm to chip.

```
OMAP Logic # nandecc chip
```

You should see the following output:

```
NAND: Internal to NAND ECC selected
```

20. Burn the *u-boot.bin* file to the fifth block in NAND.

```
OMAP Logic # nand write.i ${loadaddr} 0x00080000 0x00080000
```

You should see the following output:

```
NAND write: device 0 offset 0x80000, size 0x80000
524288 bytes written: OK
```

21. At this point, U-Boot should be bootable from NAND. To check this, power off the DM3730 Development Kit and remove the SD card. Power on the development kit and you should see the output below, indicating you have successfully booted from NAND.

```
Texas Instruments X-Loader 1.42 BSP-dm37x-2.4-2 for dm3730logic (2013-
06-18 14:09:10)
DRAM: 256MB (ProductID defined)
Booting from NAND
NAND: in-chip ECC
0x1c0000 bytes read from nand at offset 0x80000 to 80400000
```

NOTE: If you do not see this output, repeat this lab.

In Lab 5, we will learn about environment variables that can run as scripts. To pique your interest in continuing to Lab 5, follow the steps below.

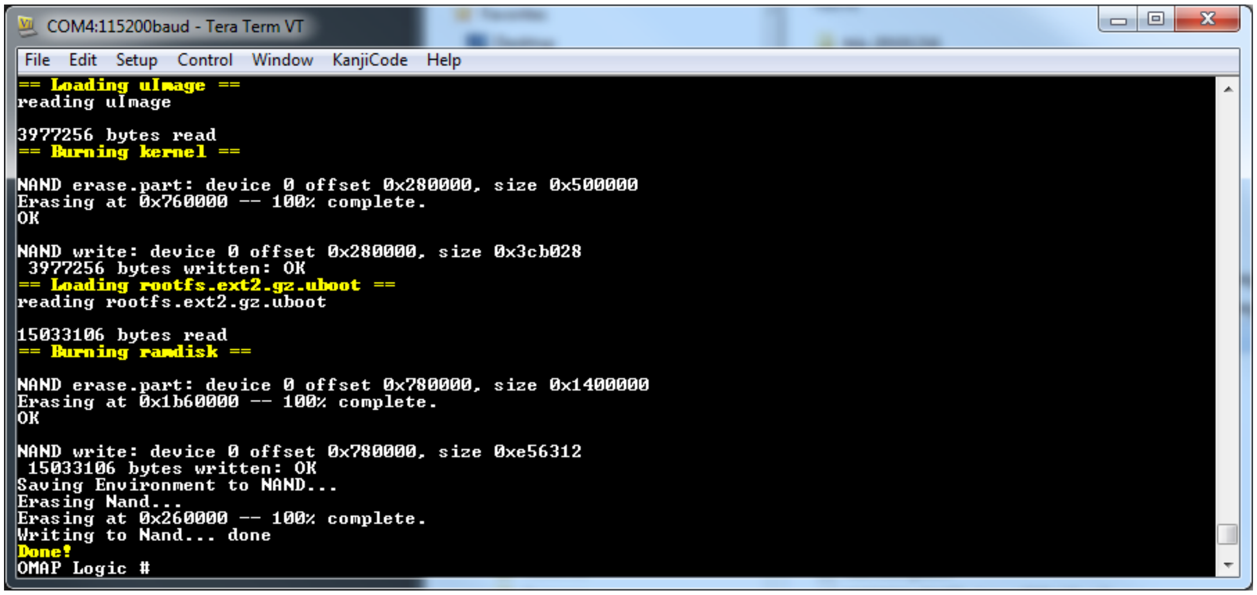
1. Enter the command below.

```
OMAP Logic # nand erase.chip
```

2. Power cycle the DM3730 Development Kit with the SD card removed. What happened to all your hard work from Lab 4? Yes, it is gone, but you will now learn about a useful script stored in the U-Boot environment.
3. Reboot into U-Boot using the SD card and enter the command below.

```
OMAP Logic # run makenandboot
```

You should see the following output:


COM4:115200baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
== Loading uImage ==
reading uImage

3977256 bytes read
== Burning kernel ==

NAND erase.part: device 0 offset 0x280000, size 0x500000
Erasing at 0x760000 -- 100% complete.
OK

NAND write: device 0 offset 0x280000, size 0x3cb028
 3977256 bytes written: OK
== Loading rootfs.ext2.gz.uboot ==
reading rootfs.ext2.gz.uboot

15033106 bytes read
== Burning ramdisk ==

NAND erase.part: device 0 offset 0x780000, size 0x1400000
Erasing at 0x1b60000 -- 100% complete.
OK

NAND write: device 0 offset 0x780000, size 0xe56312
 15033106 bytes written: OK
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x260000 -- 100% complete.
Writing to Nand... done
Done!
OMAP Logic #
```

4. Remove the SD card and power cycle the DM3730 Development Kit again. What happened? The *makenandboot* command is an environment variable that runs as a script. Continue to Lab 5 to learn more.

6 Lab 5: Set Up Environment Variables

This lab will teach you about the U-Boot environment. The U-Boot environment consists of all the shell variables residing in RAM when U-Boot is running. These variables are stored in a block of persistent memory (NAND) when the system is not running and are copied into RAM when U-Boot is booted. When the DM3730/AM3703 SOM is powered on, the system first checks NAND for variables; if they are not present then they are loaded from the default environment in the U-Boot image. This environment can be changed by making the desired modifications and rebuilding the U-Boot image.

6.1 Prerequisites

- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- A host PC or laptop
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- A terminal emulation program (e.g., Tera Term) with the following port settings:

  a. Baud rate: **115200**
  b. Data: **8 bit**
  c. Parity: **None**
  d. Stop: **1 bit**
  e. Flow control: **None**
  f. Transmit Delay: **20msec/char**

6.2 Connect Kit to Host PC

1. Use the serial cable or USB A to mini-B cable included in the development kit to connect the serial port on the kit to the appropriate port on your host PC.
2. Insert the DM37x Linux Demo Image SD card into the bootable card slot on the baseboard.
3. Start the Tera Term program.
4. Power on the development kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS. You should see the following output:

```
Hit any key to stop autoboot:  0
OMAP Logic #
```

NOTE: If you do not see the U-Boot script output, please check to make sure the serial cable is connected correctly and the SD card is fully inserted. If you still do not see the U-Boot welcome prompt, the U-Boot files may not be loaded properly on the SD card. Please refer to Section 5 for instructions on how to update and load U-Boot.

6.3 Using *env* Help Menu

Enter the *env* command.

```
OMAP Logic # env
```

You should see the following output:

```
env - environment handling commands

Usage:
env default -f - reset default environment
env edit name - edit environment variable
env export [-t | -b | -c] addr [size] - export environment
env import [-d] [-t | -b | -c] addr [size] - import environment
env print [name ...] - print environment
env run var [...] - run commands in an environment variable
env save - save environment
env set [-f] name [arg ...]
```

These are all the environment handling commands available to you in U-Boot. We have already used several of these.

6.4 Work with U-Boot Environment Variables

1. Begin by viewing the current environment.

```
OMAP Logic # printenv
```

You should see the following output:

```
...
rootfs_device=/dev/mtdblock5
rootfs_location=nand-part
rootfs_type=ramdisk
rotation=0
serverip=192.168.3.10
setconsole=setenv console ${consoledevice},${baudrate}n8
splashimage=0x8F862FD8
splashpos=m,m
stderr=serial
stdin=serial
stdout=serial
uboot_partition=u-boot
ubootimage=u-boot.bin.ift
xloader_partition=x-loader
xloadimage=mlo
yaffs_partition=fs
yaffsimage=rootfs.yaffs2

Environment size: 12077/131068 bytes
OMAP Logic #
```

NOTE: It is possible that you see variables set differently than those above; this is no cause for concern because you may have variables that were loaded from NAND or you may have a different version of U-Boot. In the latter case, if you would like to update U-Boot, please refer to Section 5.

2. It is important to note that different versions of U-Boot may have different environments that could cause compatibility issues. Before working with a new version of U-Boot, it is advised that you load the default environment into RAM.

```
OMAP Logic # env default –f
```

You should see the following output:

```
## Resetting to default environment
```

3. Before using any U-Boot commands that modify the environment, it is wise to first see exactly how to use them. Look at the help topic for the *setenv* command.

```
OMAP Logic # help setenv
```

You should see the following output:

```
setenv - set environment variables

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

As you can see from the above output, it is possible to delete a variable easily, so take care when using the *setenv* command. If you delete a variable, you can always recreate it or reset the default environment.

4. Before or after using the *setenv* command, you may want to see what a variable is set to.

```
OMAP Logic # printenv bootdelay
```

You should see the following output:

```
bootdelay=3
OMAP Logic #
```

5. Now we know what the boot delay is, let's set it to a higher value to give us more time to press a key before U-Boot boots to the OS.

```
OMAP Logic # setenv bootdelay 13
```

6. To see if you successfully set *bootdelay*, repeat Step 4 above.

7. There is one more step that must be executed before our change takes effect. Currently our environment is loaded in RAM which is not persistent across power cycles, so we need to store our environment in NAND.

```
OMAP Logic # saveenv
```

   You should see the following output:

```
Saving Environment to NAND...
Erasing Nand...
Erasing at 0x260000 -- 100% complete.
Writing to Nand... done
```

   Now the new environment will be loaded when the system is powered on.

8. Power cycle the DM3730 Development Kit to verify that the new *bootdelay* is thirteen seconds. If you see a thirteen-second delay you have successfully modified and saved a U-Boot environment in NAND flash.

6.5 Send U-Boot Commands from Text File

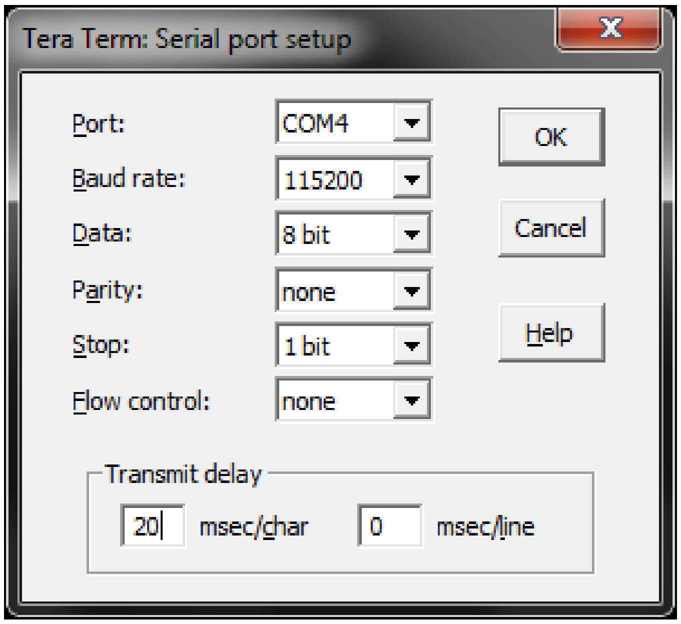
Sending commands from a text file to U-Boot is a useful feature of Tera Term. It is especially useful if you need to set numerous environment variables. In this section, we will explore this feature.

1. Open your favorite text editor and enter the following set of commands.

```
setenv carColor Blue
setenv carEngine V8
setenv carMake Logic PD
setenv car Make = ${carMake},
setenv car ${car} Engine = ${carEngine},
setenv car ${car} Color = ${carColor}
```

2. Save the file as *car.txt*.

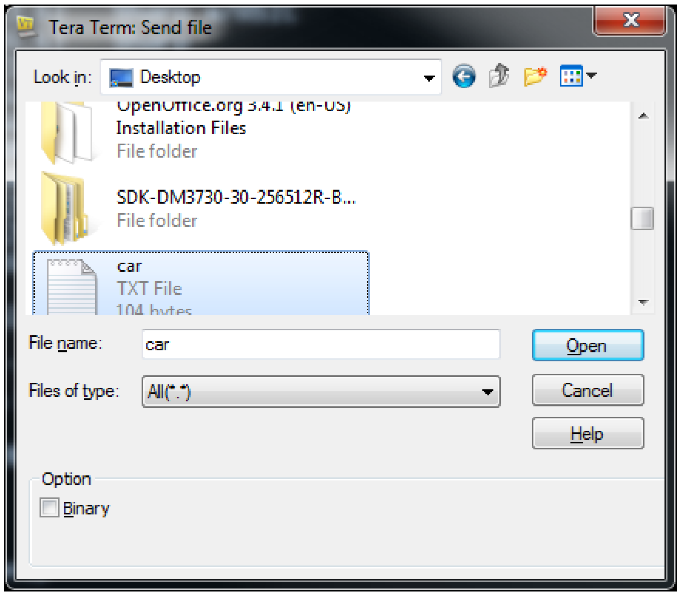
3. In Tera Term, select Setup > Serial port and set the transmit delay to 20 msec/char. Click OK.


Tera Term: Serial port setup

Port: COM4
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
20 msec/char
0 msec/line

4. Next, select File > Send file... and browse to the location of the *car.txt* file. Select it and click Open.


Tera Term: Send file

Look in: Desktop

OpenOffice.org 3.4.1 (en-US) Installation Files
File folder

SDK-DM3730-30-256512R-B...
File folder

car
TXT File
104 bytes

File name: car
Files of type: All(*.*)

Option
Binary

You should start to see your commands display at the U-Boot prompt.

5. Check that the commands were correctly entered and interpreted.

```
OMAP Logic # printenv car
```

You should see the following output:

```
car=Make = Logic PD, Engine = V8, Color = Blue
```

7 Lab 6: RAM, Flash and File Systems

This lab will teach you how to utilize file systems on the DM3730 Development Kit.

7.1 Prerequisites

- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- A host PC or laptop
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- A terminal emulation program (e.g., Tera Term) with the following port settings:

  a. Baud rate: **115200**
  b. Data: **8 bit**
  c. Parity: **None**
  d. Stop: **1 bit**
  e. Flow control: **None**

7.2 Fat File System (FATFS)

This section explores how to mount and browse a FAT file system (FATFS) on an SD card in U-Boot.

1. Save a text file on the DM37x Linux Demo Image SD card named *helloworld.txt*. Make sure there is some text in this file, so it is larger than 0 bytes.
2. Use the serial cable or USB A to mini-B cable included in the development kit to connect the serial port on the kit to the appropriate port on your host PC.
3. Insert the DM37x Linux Demo Image SD card with the *helloworld.txt* file into the bootable card slot on the baseboard.
4. Start the Tera Term program.
5. Power on the development kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS.

```
Hit any key to stop autoboot:  0
OMAP Logic #
```

6. Review the help topic for the *mmc* command.

```
OMAP Logic # help mmc
```

   You should see the following output:

```
mmc - MMC sub-system

Usage:
mmc init [dev] - init MMC sub system
mmc device [dev] - show or set current device
```

7. As shown in the output above, use the *mmc init* command to initialize the MMC subsystem.

```
OMAP Logic # mmc init
```

You should see the following output:

```
mmc1 is available
```

8. Now, we are ready to explore the SD card. Begin by reviewing the help topics for the *fatinfo* and *fatls* commands.

```
OMAP Logic # help fatinfo
OMAP Logic # help fatls
```

You should see output similar to that included below. Remember that variables in the <> are required and variables in the [ ] are optional.

```
OMAP Logic # help fatinfo
fatinfo - print information about filesystem

Usage:
fatinfo <interface> <dev[:part]>
    - print information about filesystem from 'dev' on 'interface'
OMAP Logic # help fatls
fatls - list files in a directory (default /)

Usage:
fatls <interface> <dev[:part]> [directory]
    - list files from 'dev' on 'interface' in a 'directory'
```

This output tells us that we need to provide the interface that the FAT partition exists on and the device number for that interface. Because we are using the SD card, our interface is *mmc* and our device number is *1*. We will need this information in the next step.

9. Browse the SD card using the information noted above.

```
OMAP Logic # fatinfo mmc 1
```

You should see the following output:

```
Interface:  MMC
  Device 0: Vendor:  Rev:  Prod:
            Type: Hard Disk
            Capacity: 32.0 MB = 0.0 GB (65536 x 512)
Partition 1: Filesystem: FAT32 "boot       "
```

10. Finally, see what is on the SD card.

```
OMAP Logic # fatls mmc 1
```

You should see the following output:

```
    44760   mlo
       13   helloworld.txt
     4738   mklogicfatcard.sh
 14531352   rootfs.ext2.gz.uboot
 42153408   rootfs.yaffs2
120935232   rootfs-dsp.yaffs2
   427616   u-boot.bin
   427628   u-boot.bin.ift
  3954224   uimage
  3954224   uimage-dsp
```

7.3 Work with RAM from U-Boot

1. If you have not already done so, save a text file on the DM37x Linux Demo Image SD card named *helloworld.txt*. Make sure there is some text in this file, so it is larger than 0 bytes
2. Use the serial cable or USB A to mini-B cable included in the development kit to connect the serial port on the kit to the appropriate port on your host PC.
3. Insert the DM37x Linux Demo Image SD card with the *helloworld.txt* file into the bootable card slot on the baseboard.
4. Start the Tera Term program.
5. Power on the development kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS.
6. Before beginning work with the RAM, determine the size and location of the RAM.

```
OMAP Logic # bdinfo
```

You should see the following output:

```
arch_number = 0x00000CB1
boot_params = 0x80000100
DRAM bank   = 0x00000000
-> start    = 0x80000000
-> size     = 0x10000000
DRAM bank   = 0x00000001
-> start    = 0xA0000000
-> size     = 0x00000000
ethaddr     = 00:08:ee:04:e6:5a
ip_addr     = 0.0.0.0
baudrate    = 115200 bps
TLB addr    = 0x8FFF0000
relocaddr   = 0x8F806000
reloc off   = 0x0F406000
data_start  = 0x8F85F4D8
rodata_start= 0x8F84F0F8
bss_start   = 0x8F865F18
irq_sp      = 0x8F1C5F60
sp start    = 0x8F1C5F50
FB base     = 0x8F89D000
```

From this output, you can see that the RAM is at offset 0x8000 0000 and has a size of 256 mebibytes.

7. Before beginning work with the SD card, initialize the MMC subsystem.

```
OMAP Logic # mmc init
```

You should see output similar to that included below. Take note of the number following *mmc*. In the example below, the *dev* number is 1.

```
mmc1 is available
```

8. Now, copy the *helloworld.txt* file into the RAM.

```
OMAP Logic # fatload mmc 1 0x80000000 helloworld.txt
```

You should see output similar to that included below. Take note of the bytes that are written, as you will need this information in the next step.

```
reading helloworld.txt

14 bytes read
```

9. Write one more copy of *helloworld.txt* to RAM. You will need to take the previous offset plus the number of bytes written to create the new offset to be used in this step. In this example, there were 14 bytes written in the previous step, so the new offset is the previous offset plus 14 bytes or 0x8000000E. Your text file is likely to be another size, but you will need to make a similar calculation.

```
OMAP Logic # fatload mmc 1 0x8000000e helloworld.txt
```

You should see output similar to that in Step 8 above.

10. Now that there are two copies of *helloworld.txt* in RAM, try experimenting with the *cmp.b* (memory compare) command, as seen below. You will need the two offsets that you used when writing *helloworld.txt* to RAM.

```
OMAP Logic # cmp.b 0x80000000 0x8000000e 14
```

You should see output similar to that included below. Most importantly, the output should say that the 14 bytes were the same.

```
byte at 0x8000000e (0x68) != byte at 0x8000001c (0x80)
Total of 14 bytes were the same
```

11. Now, compare a different part of memory to see what output you get.

```
OMAP Logic # cmp.b 0x80000000 0x80000010 14
```

You should see the following output:

```
byte at 0x80000000 (0x68) != byte at 0x80000010 (0x6c)
Total of 0 bytes were the same
```

7.4 Yet Another Flash File System (YAFFS)

Yet Another Flash File System (YAFFS) was developed by a company named Aleph One Limited and was then incorporated by Logic PD into the U-Boot software. The YAFFS partitions are defined in the U-Boot environment and can be configured when the U-Boot image is built. When the system boots, the U-Boot environment is loaded into RAM and is operated on from there. Therefore, the YAFFS partitions do not remain persistent, but the data stored in NAND does. In order to access the data on a YAFFS partition, the partition must be mounted after each boot; the data will then be accessible to the system.

This section will describe several ways to work with YAFFS in U-Boot.You can use U-Boot to mount up to four YAFFS partitions at a time. The following limits are imposed on partitions:

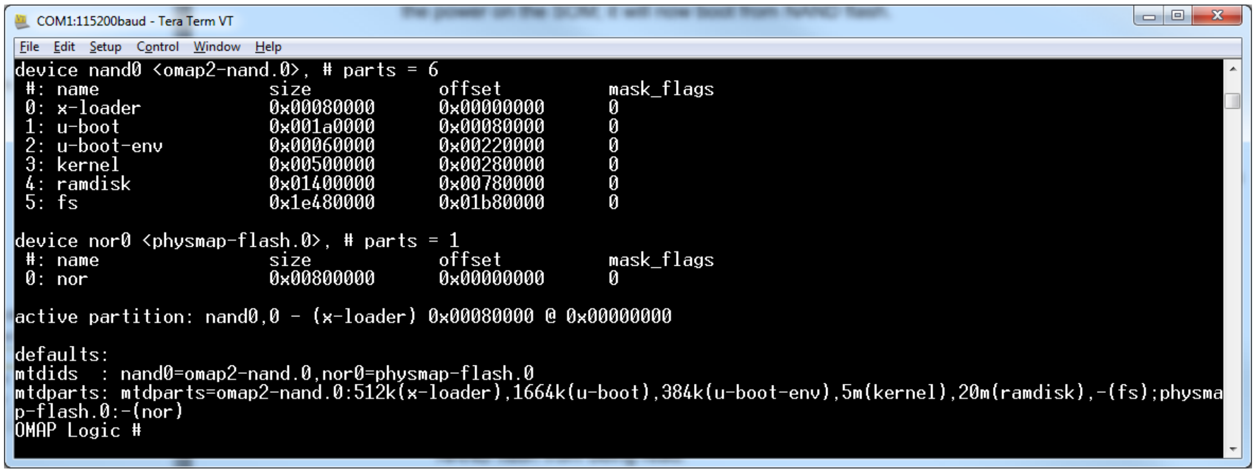
- Each partition must have a unique name.
- Each partition must exist on local flash accessible from U-Boot's */dev/flashx* device file, where *x* is an instance index. For example*, /dev/flash0* or */dev/flash1*.
- Each partition must span at least four physical flash blocks.
- A partition **must not** overlap the flash blocks that contain U-Boot.

7.4.1 View YAFFS Partitions

1. Use the serial cable or USB A to mini-B cable included in the development kit to connect the serial port on the kit to the appropriate port on your host PC.
2. Insert the DM37x Linux Demo Image SD card into the bootable card slot on the baseboard.
3. Start the Tera Term program.
4. Now that you know a little about YAFFS partitions, take a look at the partitions currently set up on the SOM's flash.

```
OMAP Logic # **mtdparts**
```

You should see the following output:


COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
device nand0 <omap2-nand.0>, # parts = 6
 #: name                size            offset          mask_flags
 0: x-loader            0x00080000      0x00000000      0
 1: u-boot              0x001a0000      0x00080000      0
 2: u-boot-env          0x00060000      0x00220000      0
 3: kernel              0x00500000      0x00280000      0
 4: ramdisk             0x01400000      0x00780000      0
 5: fs                  0x1e480000      0x01b80000      0

device nor0 <physmap-flash.0>, # parts = 1
 #: name                size            offset          mask_flags
 0: nor                 0x00800000      0x00000000      0

active partition: nand0,0 - (x-loader) 0x00080000 @ 0x00000000

defaults:
mtdids  : nand0=omap2-nand.0,nor0=physmap-flash.0
mtdparts: mtdparts=omap2-nand.0:512k(x-loader),1664k(u-boot),384k(u-boot-env),5m(kernel),20m(ramdisk),-(fs);physma
p-flash.0:-(nor)
OMAP Logic #
```

These partitions are automatically set up by U-Boot through a script stored in an environmental variable. These partitions are meant to be used by a Linux OS, but we are going to use them to practice U-Boot YAFFS commands.

7.4.2 Mount YAFFS Partition

We must mount a YAFFS partition before we are able to manipulate a directory or file on the partition.

1. Mount the *fs* partition.

```
OMAP Logic # ymount /fs
```

You should see the following output:

```
yaffs: Mounting /fs
ECC oobavail <= 16; forcing ‘tags-ecc-off’
```

2. Now, take a look at what is on the *fs* partition.

```
OMAP Logic # yls /fs
```

You should see the following output:

```
lost+found
```

7.4.3 Manipulate Directories and Files on YAFFS Partition

1. Begin by saving a text file named *helloworld.txt* to an SD card; make sure there is some text in the file, so it is bigger than 0 bytes in size.
2. Insert the SD card with the *helloworld.txt* file into the bootable card slot on the baseboard.
3. Make a directory where we can save the *helloworld.txt* file on the YAFFS partition.

```
OMAP Logic # ymkdir /fs/myfiles
```

4. Check that the directory was successfully created.

```
OMAP Logic # yls /fs
```

You should see the following output:

```
myfiles
lost+found
```

Now that the *myfiles* directory is created, it can be populated with files.

5. Copy the *helloworld.txt* file into the *myfiles* directory. There is no command that will copy the file directly from the SD card, so we need to first write the *helloworld.txt* file to RAM and then copy it into the directory.

   a. First, initialize the SD card.

```
OMAP Logic # mmc init
```

   You should see the following output:

```
mmc1 is available
```

   b. Now check to see if the *helloworld.txt* file has been correctly saved on the SD card.

```
OMAP Logic # fatls mmc 1
```

   You should see output similar to that included below, indicating that the *helloworld.txt* file is 14 bytes.

```
       14   helloworld.txt

1 file(s), 0 dir(s)
```

   c. Now, write the *helloworld.txt* file to RAM.

```
OMAP Logic # fatload mmc 1 0x80000000 helloworld.txt
```

   You should see the following output:

```
reading helloworld.txt

14 bytes read
```

   d. Now that the *helloword.txt* file is stored in RAM, write it into the YAFFS *myfiles* directory.

```
OMAP Logic # ywrm /fs/myfiles/helloworld 0x80000000 14
```

   You should see the following output:

```
Copy 0x80000000 to /fs/myfiles/helloworld ... [DONE]
```

6. Cycle the power on the DM3730 Development Kit and boot back into U-Boot.

7. Check to see if the *helloworld.txt* file was successfully saved in the YAFFS partition *fs*.

   NOTE: If you have already mounted *fs* then skip entering the commands below.

```
OMAP Logic # ymount /fs

OMAP Logic # yls /fs/myfiles
```

   You should see the following output:

```
helloworld
```

7.4.4 Move Files from YAFFS Partition to RAM

This section is a continuation of Section 7.4.3. If you have not completed Section 7.4.3, please do so before proceeding.

1. Now that we have a file stored on a YAFFS partition, we are ready to copy it back into RAM.

```
OMAP Logic # yrdm 0x81234567 /fs/myfiles/helloworld
```

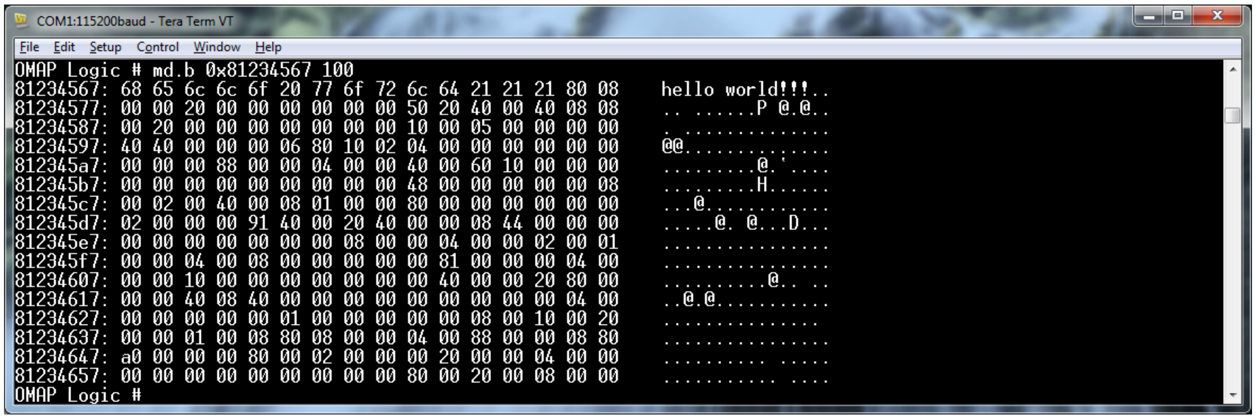
   You should see the following output:

```
Copy /fs/myfiles/helloworld to 0x81234567... [DONE]
```

2. Display the content at the memory address noted below.

```
OMAP Logic # md.b 0x81234567 100
```

   You should see the first 100 bytes of the *helloworld.txt*, as shown below.


COM1:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
OMAP Logic # md.b 0x81234567 100
81234567: 68 65 6c 6c 6f 20 77 6f 72 6c 64 21 21 21 80 08    hello world!!!..
81234577: 00 00 20 00 00 00 00 00 00 50 20 40 00 40 08 08    .. ......P @.@..
81234587: 00 20 00 00 00 00 00 00 00 10 00 05 00 00 00 00    . ..............
81234597: 40 40 00 00 00 06 80 10 02 04 00 00 00 00 00 00    @@..............
812345a7: 00 00 00 88 00 00 04 00 00 40 00 60 10 00 00 00    .........@.`....
812345b7: 00 00 00 00 00 00 00 00 00 48 00 00 00 00 00 08    .........H......
812345c7: 00 02 00 40 00 08 01 00 00 80 00 00 00 00 00 00    ...@............
812345d7: 02 00 00 00 91 40 00 20 40 00 00 08 44 00 00 00    .....@. @...D...
812345e7: 00 00 00 00 00 00 00 08 00 00 04 00 00 02 00 01    ................
812345f7: 00 00 04 00 08 00 00 00 00 00 81 00 00 00 04 00    ................
81234607: 00 00 10 00 00 00 00 00 00 00 40 00 00 20 80 00    ..........@.. ..
81234617: 00 00 40 08 40 00 00 00 00 00 00 00 00 00 04 00    ..@.@...........
81234627: 00 00 00 00 00 01 00 00 00 00 00 08 00 10 00 20    ...............
81234637: 00 00 01 00 08 80 08 00 00 04 00 88 00 00 08 80    ................
81234647: a0 00 00 00 80 00 02 00 00 00 20 00 00 04 00 00    ........... ....
81234657: 00 00 00 00 00 00 00 00 00 80 00 20 00 08 00 00    ........... ....
OMAP Logic #
```

8 Lab 7: Set Up Network Interfaces

This lab will teach you how to connect your host PC to the DM3730 Development Kit via Ethernet.

8.1 Prerequisites

- An Ethernet switch with DHCP capabilities (optional; applies to Section 8.5)
- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- A host PC or laptop
- An Ethernet cable
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- A terminal emulation program (e.g., Tera Term) with the following port settings:

  a. Baud rate: **115200**
  b. Data: **8 bit**
  c. Parity: **None**
  d. Stop: **1 bit**
  e. Flow control: **None**

8.2 Connect Kit to Host PC

1. Use the serial cable or USB A to mini-B cable included in the development kit to connect the serial port on the kit to the appropriate port on your host PC.
2. Insert the DM37x Linux Demo Image SD card into the bootable card slot on the baseboard.
3. Start the Tera Term program.
4. Power on the development kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS.

```
Hit any key to stop autoboot:  0
OMAP Logic #
```

8.3 Connect Kit to Ethernet

There are two types of addressing that can be used to connect to your DM3730 Development Kit via Ethernet:

- Static IP addressing allows you to manually set the Ethernet addresses on your host PC and development kit.
- Dynamic IP addressing uses a Dynamic Host Configuration Protocol (DHCP) to obtain an Ethernet address. DHCP requires a DHCP server on the network, which is usually built into modern Ethernet switches. For more information on DHCP, please see the network switch's user manual.

There are also two different ways you can connect your host PC to the DM3730 Development Kit via Ethernet:

- A crossover cable that connects directly between the development kit and host PC via the RJ.45 connector on the baseboard; this option is commonly used with static IP addressing.
- A standard Ethernet cable that requires the host PC and development kit to be connected to a network switch; this option is commonly used with either dynamic or static IP addressing.

Ethernet connectivity will make several procedures easier. We will discuss both connecting directly using a crossover Ethernet cable with static IP addressing and connecting through a network switch with dynamic IP addressing.

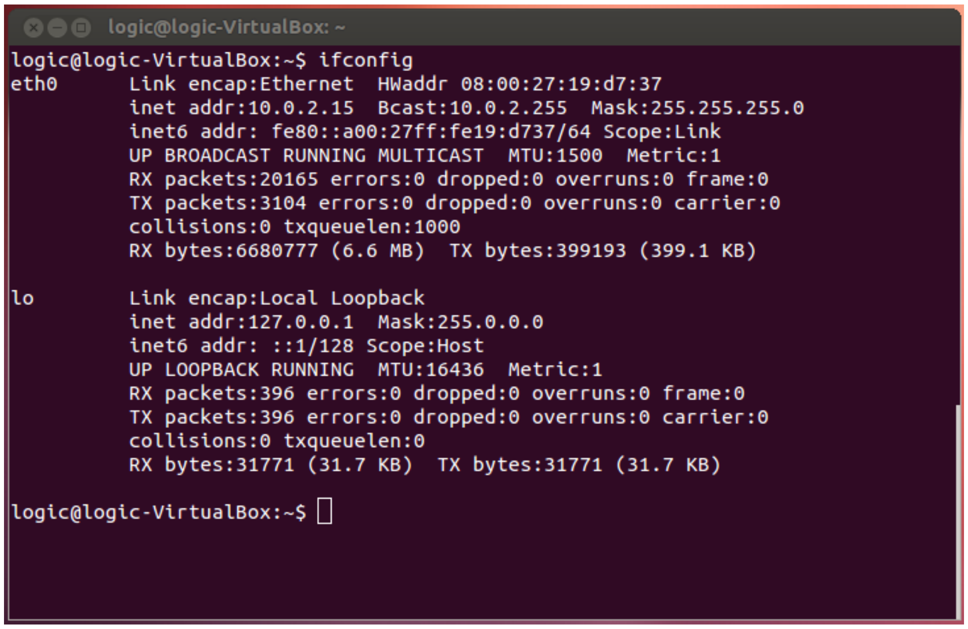
8.4 Network Setup for Ethernet Crossover Cable and Static IP Addressing

In this section we will connect the DM3730 Development Kit directly to the host PC with a crossover cable.

1. Before the development kit can communicate with the host PC, it needs to know the IP address of the host PC. Use the command below to obtain the IP address in Linux.

```
OMAP Logic # ifconfig
```

You should see the following output:



```
logic@logic-VirtualBox:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:19:d7:37
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe19:d737/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20165 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3104 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6680777 (6.6 MB)  TX bytes:399193 (399.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:396 errors:0 dropped:0 overruns:0 frame:0
          TX packets:396 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:31771 (31.7 KB)  TX bytes:31771 (31.7 KB)

logic@logic-VirtualBox:~$ 
```

2. The information that is needed is displayed on the line starting with *inet addr*. From this line you can see that the IP address of the host is 10.0.2.15.

   NOTE: It is likely that your host IP address will be different than that shown.

3. Also, on the line that starts with *inet addr*, is the *Mask* which tells us how many bits are unique in the IP address. From the above terminal output, the subnet mask is 255.255.255.0; this means that the IP address of the board should start with 10.0.2.xxx.

4. Now we are ready to enter the static network information into the U-Boot environment. **NOTE:** Remember that the IP address for your system is likely to be different. Please make the appropriate modifications to the following commands.

```
OMAP Logic # setenv ipaddr 10.0.2.38

OMAP Logic # setenv serverip 10.0.2.15
```

The kit is ready to communicate to the host PC through a static network connection over Ethernet.

5. Test the kit's ability to communicate to the host PC.

```
OMAP Logic # ping ${serverip}
```

You should see the following output:

```
smc911x: detected LAN9221 controller
smc911x: phy initialized
smc911x: MAC 00:08:ee:04:e6:5a
Using smc911x-0 device
host 10.0.2.15 is alive
```

8.5 Network Setup for Standard Ethernet Cable and DHCP Addressing

1. First, as an exercise, try to see if there is an IP address saved in the U-Boot environment.

```
OMAP Logic # printenv ipaddr
```

You should see the following output:

```
## Error: ‘ipaddr’ not defined
```

2. From the above output, it is obvious that the system has not been issued an IP address from the gateway. Let’s request one.

```
OMAP Logic # dhcp
```

You should see the following output:

```
smc911x: detected LAN9221 controller
smc911x: phy initialized
smc911x: MAC 00:08:ee:04:e6:5a
BOOTP broadcast 1
DHCP client bound to address 192.168.120.159
```

NOTE: It is likely that your IP address will be different than that shown.

3. To see if the new IP address is saved in an environment variable, press the up arrow twice and then press **ENTER**. You should see the following output with your IP address:

```
ipaddr=192.168.120.159
```

8.6 Check Connection with *ping* Command

In the previous sections, we used the *dhcp* command to initialize the Ethernet connection. Now we are going to use the *ping* command to check the connection to a specific workstation. You will need to have the DM3730 Development Kit connected via Ethernet and know the IP address of the target host PC.

1. First, identify the IP address of the host PC.
   a. In Windows, select Start > Run.
   b. Enter the command below.

```
cmd
```

   c. A terminal window should appear; at the prompt, enter the command below.

```
ipconfig /all
```

   Use the output in the terminal window to locate the IP address for the connection that you are using.

2. To ping your host PC one time, enter the command below at the `OMAP Logic #` prompt, where ***XXX.XXX.XXX.XXX*** is the IP address of your host PC.

```
OMAP Logic # ping XXX.XXX.XXX.XXX
```

9 Lab 8: Boot a Linux Image

This lab will teach you how to boot a Linux image on the DM3730 Development Kit.

9.1 Prerequisites

- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- A host PC or laptop
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- A terminal emulation program (e.g., Tera Term) with the following port settings:

  a. Baud rate: **115200**
  b. Data: **8 bit**
  c. Parity: **None**
  d. Stop: **1 bit**
  e. Flow control: **None**

9.2 Connect Kit to Host PC

1. Use the serial cable or USB A to mini-B cable included in the development kit to connect the serial port on the kit to the appropriate port on your host PC.
2. Insert the DM37x Linux Demo Image SD card into the bootable card slot on the baseboard.
3. Start the Tera Term program.
4. Power on the development kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS.

```
Hit any key to stop autoboot:  0
OMAP Logic #
```

9.3 Set Up Boot Arguments

The Linux image looks for an environmental variable named *bootargs* when U-Boot boots the Linux image. Initially, this variable does not exist and has to be created. In this section, we will walk through each boot argument and set up the *bootarg* environment variable.

The first argument that needs to be set tells Linux about the console. Most likely, when working with Linux on an embedded system, you will be communicating through the console, which in this case is *stdin* and *stdout* for the Linux OS. In the environment there is a variable named *consoledevice* that stores information about the console device. Additionally the Linux OS needs to know the baud rate for the serial port; this information is stored in the environment variable named *baudrate*.

1. Begin by viewing the environmental variables.

```
OMAP Logic # printenv consoledevice baudrate
```

You should see the following output:

```
consoledevice=ttyO0
baudrate=115200
```

2. Next, we can start putting together the *bootarg* variable piece by piece. To do this, we will use a concatenation technique for U-Boot.

   a. The first step is to create the *bootarg* variable and store our information about the console in a field named *console*.

```
OMAP Logic # setenv bootargs console=${consoledevice},${baudrate}n8
```

   b. Now, print out the *bootargs* that the Linux kernel needs to boot.

```
OMAP Logic # printenv bootargs
```

   You should see the following output:

```
bootargs=console=ttyO0,115200n8
```

   c. Now we are going to concatenate the next field in *bootargs*; this field is named *display* and will contain information on the display.

```
OMAP Logic # setenv bootargs ${bootargs} display=${display}
${otherbootargs}
```

   This line deserves further explanation. First, we are setting the *bootargs* variable again and U-Boot will overwrite the existing variable. To get around this, we simply dump the current contents of *bootargs* using *$*, and then concatenate our display information at the end.

   d. Examine the behavior of *$* using the command below.

```
OMAP Logic # echo ${bootargs}
```

   You should see output similar to that included below, indicating that the *echo* command printed the contents of the *bootargs* variable.

```
console=ttyO0,115200n8 display=15 ignore_loglevel early_printk
no_console_suspend
```

3. To continue building *bootargs*, we will add information pertaining to the root file system. In this case, our root file system is in RAM and is appropriately named the *ramdisk*.

```
OMAP Logic # setenv bootargs ${bootargs} root=/dev/ram rw \
OMAP Logic # ramdisk_size=${ramdisksize}
```

   Now, we have set up a minimal set of boot arguments to get a Linux kernel running on the DM3730/AM3703.

4. Let’s see if we set up *bootargs* correctly by trying to boot a Linux image from the SD card. First, we need to initialize the *mmc* device.

```
OMAP Logic # mmc init
```

You should see output similar to that included below; take note of the number of the *mmc* device, as you will need it in Step 6 below.

```
OMAP Logic # mmc init
mmc1 is available
OMAP Logic #
```

5. Next, we need to set up two more U-Boot environment variables.

```
OMAP Logic # setenv loadaddr 0x81000000 ; setenv ramdiskaddr 0x82000000
```

6. The *mmc* is initialized and we are ready to copy the Linux kernel image and the root file system to RAM.

```
OMAP Logic # fatload mmc 1 ${loadaddr} uimage

OMAP Logic # fatload mmc 1 ${ramdiskaddr} rootfs.ext2.gz.uboot
```

NOTE: In the above lines, *1* is the number of the mmc device.

7. Now that the environment is set up, boot the Linux image.

```
OMAP Logic # bootm ${loadaddr} ${ramdiskaddr}
```

You should start seeing output in the terminal indicating the kernel is loading. After some time, you should get the following output that leads to a prompt:

```
Starting syslogd and klogd
Enabling SmartReflex:
[   21.801269] omap_device: smartreflex.1: new worst case activate
latency 0: 91552
Starting inetd:
Starting the dropbear ssh server:
Setting up touch handler and default /etc/pointercal
Starting pwr-button:
Using /dev/input/event1 for S2 power button input device
DirectFB: Setup DirectFB for touch input
Restoring ALSA state for soundcard omap3logic
Enabling PM off mode:


        Welcome to the LTIB Embedded Linux Environment

!!!!! WARNING !!!!!!!

The default password for the root account is: root
please change this password using the 'passwd' command
and then edit this message (/etc/issue) to remove this message.

To enable DHCP on ethernet, type "ifup eth0"

DM-37x login:
```

10 Lab 9: U-Boot Scripting

This lab will teach you how to use U-Boot scripting features on the DM3730 Development Kit.

10.1 Prerequisites

- Completion of Section 9
- VM SDK for the DM37x Linux BSP (see Appendix A for a link)
- A DM3730 Development Kit
- A DM37x Linux Demo Image SD card
- A host PC or laptop
- An available serial port with serial cable or USB host port with USB A to mini-B cable
- A terminal emulation program (e.g., Tera Term) with the following port settings:

    a. Baud rate: **115200**
    b. Data: **8 bit**
    c. Parity: **None**
    d. Stop: **1 bit**
    e. Flow control: **None**

10.2 Connect Kit to Host PC

1. Use the serial cable or USB A to mini-B cable included in the development kit to connect the serial port on the kit to the appropriate port on your host PC.
2. Insert the DM37x Linux Demo Image SD card into the bootable card slot on the baseboard.
3. Start the Tera Term program.
4. Power on the development kit. When prompted, press any key on the keyboard within five seconds to make sure you enter U-Boot instead of booting an OS.

```
Hit any key to stop autoboot:  0
OMAP Logic #
```

10.3 Create U-Boot Script

U-Boot has the useful feature of being able to run commands from a script image file. In this lab, we will utilize the VM SDK for the DM37x Linux BSP. Make sure that you have read the *DM37x Linux BSP Release Notes* and *Virtual Machine SDK for the DM37x Linux BSP ReadMe* files (see Appendix A for a link to these documents). Also ensure that you have set up the VM appropriately.

1. Boot the VM and login using the password *logic.*
2. Launch a terminal window and enter the commands below.

```
$ cd /home/logic/logic
$ gedit
```

3. In *gedit,* type the commands that appear below:

```
setenv ramdiskaddr 0x82000000
setenv loadaddr 0x81000000
setenv bootargs console=${consoledevice},${baudrate}n8
setenv bootargs ${bootargs} display=${display} ${otherbootargs}
setenv bootargs ${bootargs} root=/dev/ram rw
ramdisk_size=${ramdisksize}
mmc init
fatload mmc 1 ${loadaddr} uimage
fatload mmc 1 ${ramdiskaddr} rootfs.ext2.gz.uboot
bootm ${loadaddr} ${ramdiskaddr}
```

4. Save the text file and name it *myLinuxBootCommands.*
5. At the Bash prompt, type *mkimage* to see if you have it installed.

```
$ mkimage
```

You should see the following output:

```
Usage: mkimage -l image
          -l ==> list image header information
       mkimage [-x] -A arch -O os -T type -C comp -a addr -e ep -n name
-d data_file[:data_file...] image
          -A ==> set architecture to 'arch'
          -O ==> set operating system to 'os'
          -T ==> set image type to 'type'
          -C ==> set compression type 'comp'
          -a ==> set load address to 'addr' (hex)
          -e ==> set entry point to 'ep' (hex)
          -n ==> set image name to 'name'
          -d ==> use image data from 'datafile'
          -x ==> set XIP (execute in place)
```

If you do not see this output, you can install *mkimage* with the command below.

```
$ sudo apt-get install uboot-mkimage
```

6. Now, that you know *mkimage*'s usage, build the U-Boot script image.

```
$ mkimage –T script –C none –n lnxBtCmds –d myLnxBtCmds lnxBtCmds.img
```

You should see the following output:

```
Image Name:   lnxBtCmds
Created:      Sun Jul  1 13:40:17 2012
Image Type:   PowerPC Linux Script (uncompressed)
Data Size:    384 Bytes = 0.38 kB = 0.00 MB
Load Address: 0x00000000
Entry Point:  0x00000000
Contents:
   Image 0:      376 Bytes =    0 kB = 0 MB
```

7. Copy the *lnxBtCmds.img* file onto an SD card.
8. You should already have U-Boot running from Section 10.2. Remove the SD card with U-Boot on it and insert the SD card with the *lnxBtCmds.img* file into the bootable card slot on the baseboard.

9. Load the *lnxBtCmds.img* into RAM.

```
OMAP Logic # mmc init
OMAP Logic # fatload mmc 1 0x80000000 lnxBtCmds.img
```

10. The *lnxBtCmds.img* file now resides in RAM and can be executed with the command below.

```
OMAP Logic # source 0x80000000
```

After some time, you should see the `DM-37x login:` prompt, indicating that you have successfully booted the Linux image using a U-Boot script. If you do not get to the prompt, repeat Lab 9.

11 Support

Logic PD has a unique environment where cutting edge problem solvers thrive and we use this to help customers bring dozens of new products to market each year. Engineers and designers are always available to assist you in your development. If you are interested in speaking with someone from our team, please contact Logic PD.[4]

11.1 Logic PD Design Services

Ideas have the potential to change the world. They can transform lives, energize markets, and grow businesses beyond all expectations. But, few human endeavors are more challenging than turning a great idea into reality. Most companies simply do not have the time or resources to capitalize on the potential of their intellectual property. That's where Logic PD comes in. Think of Logic PD as your own in-house product development team. If you can imagine it, we can design it, test it, build it and launch it, all with amazing efficiency.

11.2 Design Support Services

For customers requiring continued support with application development, Logic PD offers standard and configurable support contract options, as well as expert services to reduce time-to-market and development costs. For more information, please visit the Logic PD *Support Packages* web page.[5]

11.3 Development Kit Support

Logic PD provides many free resources that are included with your development kit, once it is registered online, to help with your development process. Please visit the Logic PD *Support* web page[6] for additional information about the offerings available to you.

[4] http://www.logicpd.com/contact/inquiry/
[5] http://www.logicpd.com/support/support-packages/
[6] http://www.logicpd.com/support/

Appendix A: Additional Documentation

The following documents were either referenced throughout this document or may be of additional assistance when working with U-Boot and the DM3730/AM3703 SOM.

- *DM37x Linux BSP User Guide*
  http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=1392
- *DM37x Linux BSP Release Notes*
  http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=1396
- Virtual Machine SDK for the DM37x Linux BSP
  http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=858
- *Virtual Machine SDK for the DM37x Linux BSP ReadMe*
  http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=1401
- DM37x Linux SD Card Demo Image
  http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=1405
- *DM37x Linux SD Card Demo Image ReadMe*
  http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=1409
- *USB to UART Chip Usage ReadMe*
  http://support.logicpd.com/DesktopModules/Bring2mind/DMX/Download.aspx?portalid=0&EntryId=1240

Appendix B: LCD Identification

In Q3 of 2013, Logic PD updated the 4.3"Display Kit to include a new LCD. Both LCDs are similar in appearance but do have identifiable differences. Figure 11.1 shows the new 4.3" display, which has a default display index of 28. Figure 11.2 shows the previous 4.3" display, which had a display index of 15.

The location of the touch screen ribbon cable, which is the most easily identifiable difference between the two LCDs, is circled in each figure.

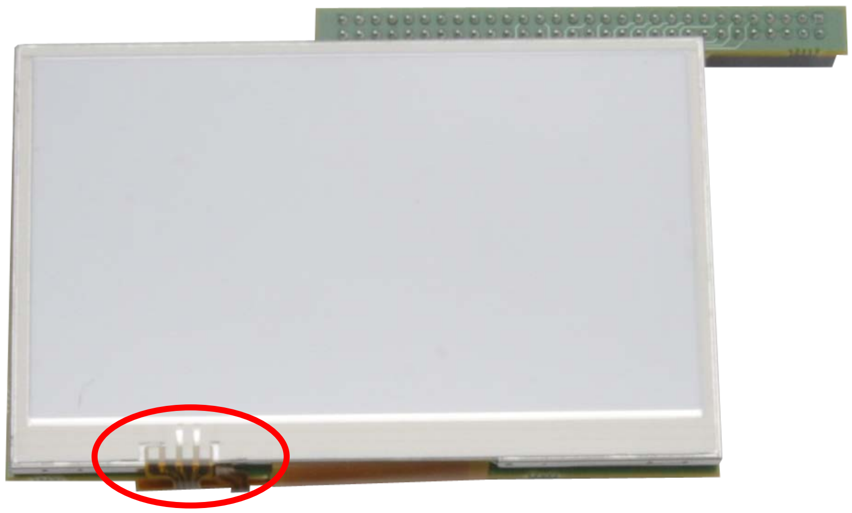


Figure 11.1: New 4.3" Display Kit (28)

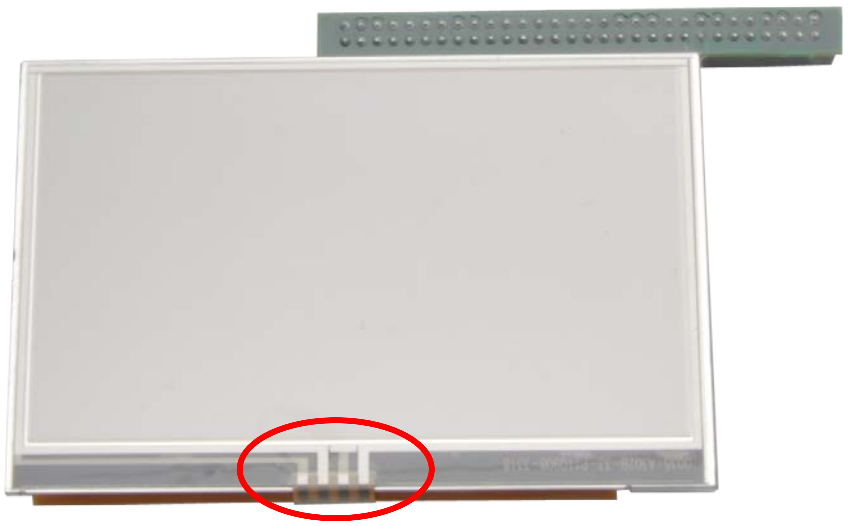


Figure 11.2: Previous 4.3" Display Kit (15)