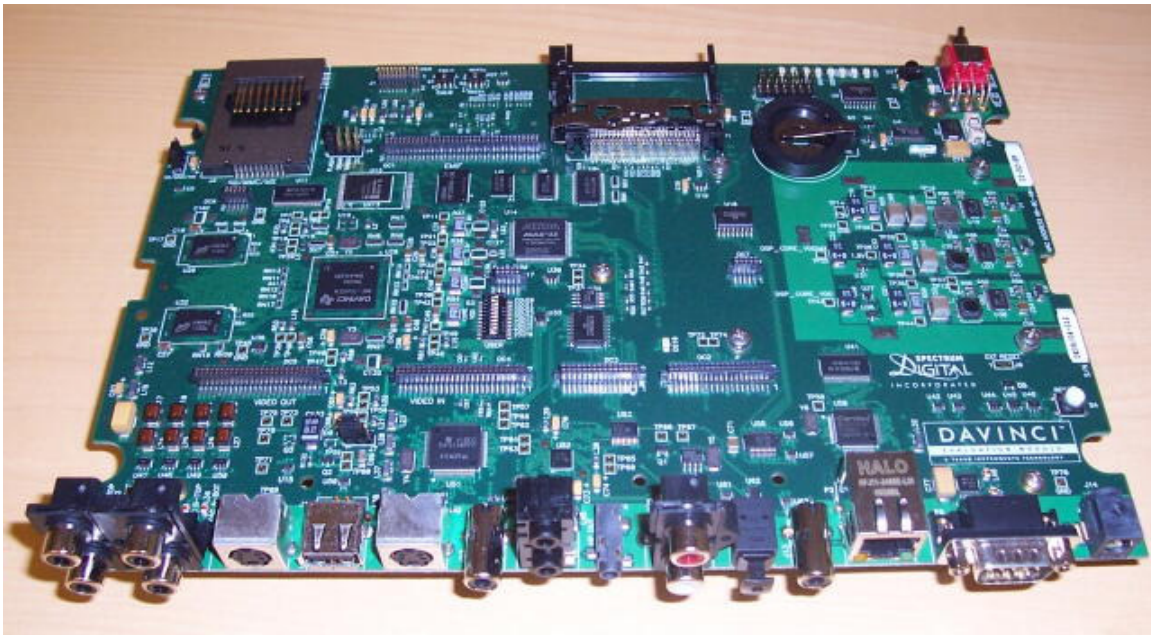


## 1 Introduction

For the remaining two labs in this course you will be programming the Texas Instruments TMS320DM6446 DaVinci processor. This lab will introduce the DaVinci hardware and software environments and guide you through the steps of running programming and running programs on the DM6446.

### 1.1 DaVinci Hardware

The DaVinci processor is a dual core ARM/DSP system-on-chip (SoC) specifically designed for multimedia applications. The DaVinci evaluation module (EVM) is very much like the motherboard of a general purpose PC.



You can study the technical specs of the DM6446 at TI's website:

DaVinci EVM - <http://focus.ti.com/docs/toolsw/folders/print/tmdsevm6446.html>

DaVinci SoC - <http://focus.ti.com/docs/prod/folders/print/tms320dm6446.html>

Getting Started - <http://focus.tij.co.jp/jp/lit/ug/sprue66d/sprue66d.pdf>

### 1.2 DaVinci Software

The DM6446 is a dual-core ARM/DSP SoC. The ARM core typically handles the control operations (communicating with other chips, synchronization, etc.) and the I/O operations (File reads/writes, image I/O from cameras, video inputs, etc.). The DSP core, on the other hand, runs optimized code primarily for codec routines (although other routines may benefit from the unique execution of a DSP).

Instead of creating one large, complex program to handle all of the tasks required of the ARM core, the Linux operating system has been ported to the ARM architecture and runs continuously in the background. Linux, like an RTOS, allows efficient transfer of the

processing power to many programs that run simultaneously. This lab will walk you through the procedure for compiling and running a program in the Linux environment. For help with basic terminal commands, the internet has many useful guides: Basic operations - [http://www.unixguide.net/linux/linuxshortcuts.shtml#Basic\\_operations](http://www.unixguide.net/linux/linuxshortcuts.shtml#Basic_operations)

Similarly, the DSP core runs a kernel on top of which multimedia codecs run. Communication between the ARM core and the DSP core occurs through a shared contiguous memory allocation called CMEM.

## 2 Lab Procedure

### 2.1 Hardware Setup

1. Carefully remove the DM6446 EVM from its static free packaging after grounding yourself.
2. Insert the Video, Ethernet, Serial, and Power cables into their respective slots (shown below) with the power socket unplugged from any electrical socket. Insert the other end of the Ethernet cable into an Ethernet jack and insert the other end of the Serial cable into the Serial port of your PC.



3. Remove the Color LCD from its packaging and insert the other end of the video cable and its power cable into their respective slots (shown below).



4. Plug the two power cords into an electrical socket with the power switches in the off position.

## 2.2 Hyper Terminal

1. Open Hyper Terminal by navigating to Start Menu → All Programs → Accessories → Communications → Hyper Terminal with your mouse. (Might be different)
2. Enter a name for this connection (Be creative!) and click on OK.
3. Select COM1 in the “Connect using” field and click on OK.
4. Choose the following Port settings and click on OK.

Bits per second	115200
Data bits	8
Parity	None
Stop bits	1
Flow control	None

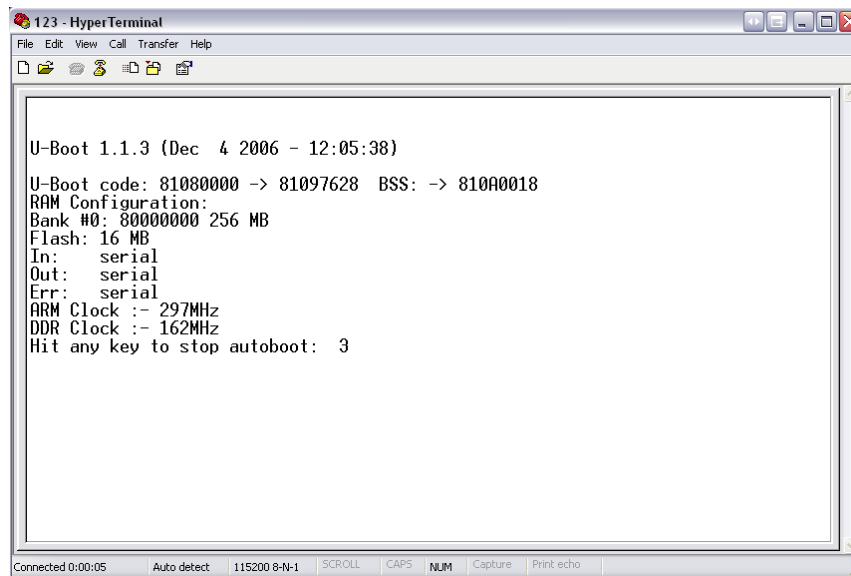
5. Hyper Terminal is now configured to communicate with the Serial port of the DM6446.

## 2.3 Power up and Login

1. Power the DM6446 by flipping the SW1 switch into the ON position (shown below).
2. Hyper Terminal should begin to display the boot menu and begin counting down from 3 seconds reading “Hit any key to stop autoboot” (shown below). Let this timer run down without pressing any keys. This option is for configuring the boot location and environment variables, which have been preset for this lab.

Troubleshooting: If you cannot see any text in the Hyper Terminal window, check your configuration settings by disconnecting (Call → Disconnect) from the port and entering the COM1 Properties. (File → Properties → Click on Configure...) Also ensure that the

Serial cable is properly connected. Note that it is possible for your PC to be using another COM port, so you should try each of them. If you are still having problems, find a TA.

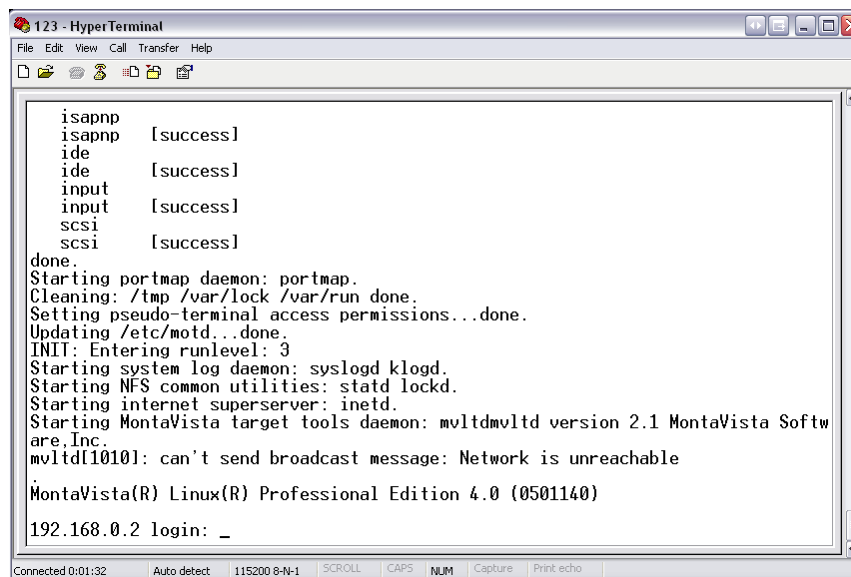


```
123 - HyperTerminal
File Edit View Call Transfer Help
U-Boot 1.1.3 (Dec  4 2006 - 12:05:38)
U-Boot code: 81080000 -> 81097628 BSS: -> 810A0018
RAM Configuration:
Bank #0: 80000000 256 MB
Flash: 16 MB
In:  serial
Out: serial
Err:  serial
ARM Clock :- 297MHz
DDR Clock :- 162MHz
Hit any key to stop autoboot:  3
Connected 0:00:05 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo
```

3. After the Linux OS has booted, you should see a login prompt (see below).

Type the following login credentials into the Hyper Terminal:

**login: root (no password)**



```
123 - HyperTerminal
File Edit View Call Transfer Help
isapnp [success]
ide [success]
input [success]
scsi [success]
done.
Starting portmap daemon: portmap.
Cleaning: /tmp /var/lock /var/run done.
Setting pseudo-terminal access permissions...done.
Updating /etc/motd...done.
INIT: Entering runlevel: 3
Starting system log daemon: syslogd klogd.
Starting NFS common utilities: statd lockd.
Starting internet superserver: inetd.
Starting MontaVista target tools daemon: mvltdmvltd version 2.1 MontaVista Software, Inc.
mvltd[1010]: can't send broadcast message: Network is unreachable
MontaVista(R) Linux(R) Professional Edition 4.0 (0501140)
192.168.0.2 login: _
Connected 0:01:32 Auto detect 115200 8-N-1 SCROLL CAPS NUM Capture Print echo
```

**Troubleshooting:** If you are unable to login with the above credentials or are experiencing any other problems, let a TA know.

\* Note that we are providing you with **root access** for your educational benefit. Please do not abuse it or you will make the TA very angry.

## 2.4 Log into Host Machine

1. Open PuTTY by navigating to Start → Network Applications → PuTTY
2. Log into the host machine by entering the IP address 143.215.204.179 into the Host Name (or IP address) field and Clicking 'Open'.
3. If you a PuTTY Security Alert window pops up, click on 'Yes'.
4. Log into the host machine with the following credentials:  
User Name: user  
Password: useruser

## 2.5 Build a Program

1. All lines in italics below indicate Linux commands for you to enter into a terminal
2. Change directories into your personalized workspace  
*cd ~/ECE3884/<your first name>/solutions*
3. Inside of this directory are all of the lab solutions that were taught during the official TI **DaVinci System Integration using Linux** training course.
4. Enter one of the solutions:  
*cd <soln##...>/app*
5. Cross-compile the source code based on the build instructions located inside of makefile:  
*make all* (*make debug* will only compile the debug version)
6. If the make completes successfully, executable files should have been placed in the current directory. The standard file names for these executables are **app\_DEBUG.x470MV** or **app\_RELEASE.x470MV** (depending on the build type).

## 2.6 Copy the Executable

1. In order to run the cross-compiled executable, we need to execute it on the DaVinci Processor. Copy the executable(s) to a directory set aside for execution by DaVinci:  
*cp app\_DEBUG.x470MV ~/exes/<your first name>/ (or app\_RELEASE.x470MV)*
2. You can inspect your customized executable directory by changing directories to it:  
*cd ~/exes/<your first name>*
3. Your executable(s) should be located in this directory.

## 2.7 Run the Executable

1. **Navigate back to HyperTerminal**
2. Inside of HyperTerminal, you should already be logged into the Linux operating system that is running on the ARM core.
3. Change directory to your customized executable directory:  
*cd /opt/ECE3884/<your first name>*
4. Inside of this directory should be the same contents as the directory that you copied the executable to. (They are actually the exact same directory, shared via NFS from the Host Machine to the DaVinci)
5. View the modules that have been loaded into the DaVinci's Operating System:  
*lsmod*
6. If there are no modules listed with the lsmod command, then load the modules from a provided script:  
*sh loadmodules.sh*
7. With the modules loaded, you are finally ready to execute the program:  
*./app\_DEBUG.x470MV*

8. To stop the program, Press Ctrl-C from within HyperTerminal.

## 2.8 Practice and Explore

We encourage you to study every project, and build as many as you can. Before building each project, study its source code and ask questions.

Building the following labs is required:

1. Run soln05ab\_basic\_make (note that there is no DaVinci executable for this part)

TA Initial Lab09 Box 1

2. Run soln06c\_audio\_loopthru

TA Initial Lab09 Box 2

3. soln07b\_video\_record

TA Initial Lab09 Box 3

4. soln07c\_video\_playback

TA Initial Lab09 Box 4

5. soln07d\_video\_loopthru

TA Initial Lab09 Box 5

## 3 Lab Report

Answer the following questions:

1. How does the ARM926 on the DaVinci Processor differ from the ARM7 employed by our Atmel AT91SAM7L?
2. Why are we using three different operating systems just to program and debug the DaVinci? Which three operating systems are those?
3. Describe three reasons why the Linux OS is used by the DaVinci.
4. What is an API and how is the concept used in the software of the DaVinci?
5. Name three tasks that would be performed on the ARM core of the DaVinci as well as three tasks that would be performed on the DSP core of the DaVinci.