UC-8481-LX Software Manual

First Edition, May 2012

www.moxa.com/product

UC-8481-LX Software Manual

Copyright Notice

Trademarks

Disclaimer

Technical Support Contact Information

www.moxa.com/support

Moxa Americas

Toll-free:	1-888-669-2872
Tel:	+1-714-528-6777
Fax:	+1-714-528-6778

Moxa Europe

Tel:	+49-89-3 70 03 99-0
Fax:	+49-89-3 70 03 99-99

Moxa China (Shanghai office)

Toll-free:	800-820-5036
Tel:	+86-21-5258-9955
Fax:	+86-10-6872-3958

Moxa Asia-Pacific

Tel:	+886-2-8919-1230
Fax:	+886-2-8919-1231

Table of Contents

1 Introduction

Welcome to the Moxa UC-8481 Series of RISC-based communication platforms.

The computer uses the Intel XScale IXP435 533MHz RISC CPU. This powerful computing engine is optimized for embedded communication functions and will not generate too much heat. The built-in 32 MB NOR Flash ROM and 512 MB SDRAM give you enough memory to directly run application software, and the 512 MB NAND Flash can be used to provide additional data storage.

Most importantly, the UC-8481 series comes with mini PCIe connectors and seven antenna port-mounts, to allow users to connect multiple wireless, cellular and GPS modules. This is particularly useful and well-suited for rolling stock applications used on moving vehicles, and makes it easy to construct an intelligent, cost-effective wireless communication platform.

Pre-installed with an operating system built around the Linux 2.6.28 kernel, the UC-8481 provides a full open source software platform for software development. Using a GNU cross compiler, open source applications written for any desktop PC can be easily compiled for the UC-8481's RISC CPU without any modification of the code. This makes the UC-8481 an optimal solution for your industrial applications.

In addition to the standard UC-8481 model, a wide temperature version (-25 to 70°C) is also available for use in harsh industrial environments.

The following topics are covered in this chapter:

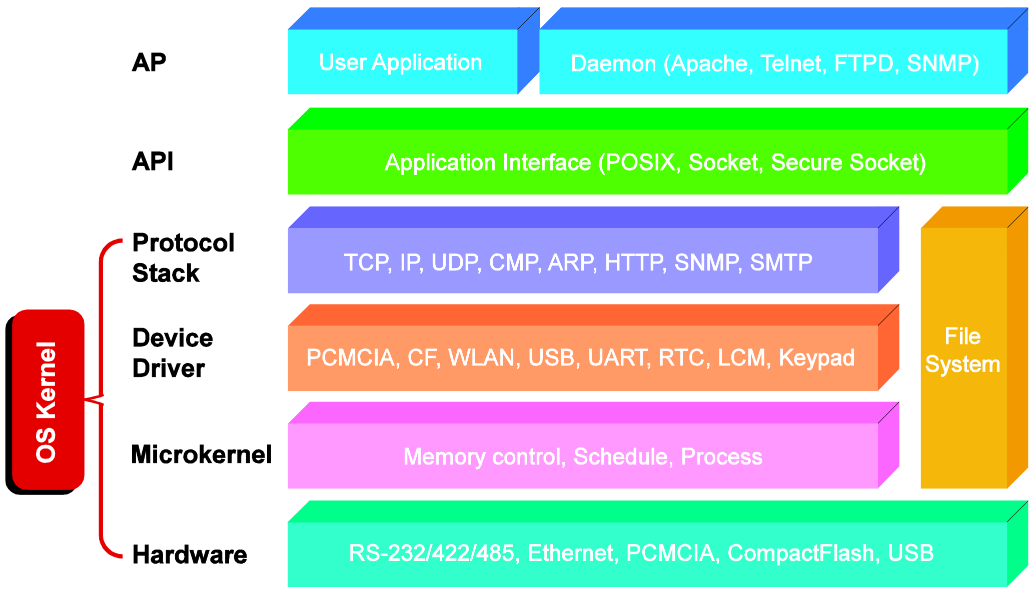
- **Overview**
- **Software Architecture**
  - Journaling Flash File System (JFFS2)
  - Software Features

Overview

The UC-8481 is pre-installed with an open source Linux operating system. Provided the software may be compiled with the GNU Compiler Collection (GCC), any software written for desktop PCs may be run. Generally, compiling for the UC-8481 is easy and straightforward enough that most of the time there is no need to modify source code written for other platforms.

Software Architecture

The Linux/GNU operating system that comes on the UC-8481 is fully POSIX compliant, making it extremely easy to run programs that also follow the POSIX standard. The full GNU tool chain is available for use when porting software. In addition, binary device drivers for the buzzer, SRAM, watchdog controls, network cards, and UART are also included.


AP
User Application
Daemon (Apache, Telnet, FTPD, SNMP)
API
Application Interface (POSIX, Socket, Secure Socket)
Protocol Stack
TCP, IP, UDP, CMP, ARP, HTTP, SNMP, SMTP
File System
OS Kernel
Device Driver
PCMCIA, CF, WLAN, USB, UART, RTC, LCM, Keypad
Microkernel
Memory control, Schedule, Process
Hardware
RS-232/422/485, Ethernet, PCMCIA, CompactFlash, USB

The UC-8481's built-in Flash ROM comes already partitioned into **boot loader**, **root directory tree (/root)**, **kernel (/root/boot)**, and **user file space (/home)** partitions.

In order to prevent user applications from corrupting the operating system, the UC-8481 mounts its root file system read-only, making it impossible for users to overwrite files on the root system during normal use. Following the initial kernel boot, the system next searches for the boot order of background and foreground processes in the ***rc*** or ***inittab*** directories. Finally, any customized configurations or device drivers (for SRAM, Ethernet or cellular modules, watchdogs, buzzer controls, and so on) are stored in the **/home** directory user space, in NOR Flash memory, so that the read-only file system never needs to be overwritten.

In the event that the operating system is corrupted, the UC-8481 software package also includes a system re-write utility that may be applied via the console, over either the serial or Ethernet interfaces. Users can call this utility to overwrite a corrupted file system with a new image of the **Factory Default OS**, thereby fully restoring the original OS as it shipped from the factory. During this system over-write, the **/home** directory structure is preserved, guaranteeing that user data and any local customizations (like custom drivers) are not erased or altered.

For more information about memory mapping and programming, please refer to the Chapter 5 "Programmer's Guide."

Journaling Flash File System (JFFS2)

The **user space directory tree (/home)** is located in flash memory formatted with the **Journaling Flash File System (JFFS2)**. This filesystem stores the directory structure in a compressed form, in a process that is transparent to the user.

Developed by Axis Communications in Sweden, JFFS2 makes it possible to store an entire directory structure directly on flash memory, without any need to emulate a block device. It is designed for use on flash ROM chips to make the most of the storage medium's special write requirements. One of the features of JFFS2 is wear-leveling, which spreads read and write operations evenly across the blocks to extend the life of the disk. Further, as its name implies, JFFS2 is a journaling file system, which means it maintains an independent log of all inodes at all times. JFFS2's journaling technology is thus able to guarantee that a file system always remains consistent, even when crashes or improper power-downs occur during write operations. Consequently, this also means that JFFS2 does not require a file system check on boot-up, cutting initialization times considerably.

JFFS2 improvements over JFFS include support for NAND flash devices, improvements in its wear-leveling, garbage-collection, and RAM footprint, better overall performance (e.g., faster seeks, faster read-write-erases, etc), native data compression, and support for hard links.

The key features of JFFS2 are:

- Allows the use of flash ROM as a block memory device, providing rugged solid-state reliability
- Maintenance of data integrity across power failures
- No file system checks are ever required at power-down or power-up, giving improved boot speeds
- Efficient wear leveling, for a longer device life
- Transparent data compression, for maximum utilization of storage space

ATTENTION

Although JFFS2 is a journaling file system, this does not preclude the loss of data. The file system will remain in a consistent state after power failures and will always be mountable. However, should the board be powered down during a write then any incomplete writes will be rolled back on the next boot, resulting in the loss of the interrupted data transfer. Writes that have already been completed, however, will not be affected.

Additional information about JFFS2 is available on the following websites:

http://sources.redhat.com/jffs2/jffs2.pdf
http://sourceware.org/jffs2/

http://sourceware.org/jffs2/jffs2-html/jffs2-html.html

http://en.wikipedia.org/wiki/Jffs2
http://www.linux-mtd.infradead.org/

Software Features

Linux

Kernel Version: 2.6.38
File System: JFFS2, NFS, Ext2, Ext3, YAFFS2
Internet Protocol Suite: TCP, UDP, IPv4, IPv6, SNMPv1, ICMP, ARP, HTTP, CHAP, PAP, DHCP, NTP, NFS, SMTP, Telnet, FTP, TFTP, PPP, PPPoE
Internet Security: OpenVPN, iptables firewall, OpenSSL
Web Server (Apache): Allows you to create and manage web sites; supports PHP and XML
Terminal Server (SSH): Provides secure encrypted communications between two un-trusted hosts over an insecure network
Dial-up Networking: PPP Daemon for Linux that allows Unix machines to connect to the Internet through dialup lines, using the PPP protocol, as a PPP server or client. Works with 'chat', 'dip', and 'diald', among (many) others. Supports IP, TCP, UDP, and (for Linux) IPX (Novell).
Watchdog: Features a hardware function to trigger system reset in a user-specified time interval (Moxa API provided)
Wireless: wpa_supplicant is configured using a text file that lists all accepted networks and security policies, including pre-shared keys.
GPS: gpsd is a daemon that receives data from a GPS receiver, and provides the data back to multiple applications such as Kismet or GPS navigation software.
Application Development Software:

- Moxa API Library (Watchdog timer, Moxa serial I/O control, Moxa DI/DO API)
- GNU C/C++ cross-compiler, supports EABI
- GNU C library
- GDB source-level debugging server

Software Protection: Encryption tool for user executable files (based on patented Moxa technology)

2
Getting Started

In this chapter, we explain how to connect the UC-8481, turn on the power, and then get started with the programming environment and other user functions.

The following topics are covered in this chapter:

Powering on the UC-8481

Connect the shielded ground to the central pin of the UC-8481 M12 power connector (the shielded contact), and then power on the computer by connecting it to the power adaptor. It takes about 30 to 60 seconds for the system to boot up. Once the system is ready, the **Ready** LED will light up.

ATTENTION

After connecting the UC-8481 to the power supply, it will take about 30 to 60 seconds for the operating system to boot up. The green **Ready** LED will not light up until the operating system is ready.

Connecting the UC-8481 to a PC

There are two ways to connect a PC to the UC-8481 console: directly, through the serial console port, or by Telnet or SSH, over a network. Once you turn on the power, you will be able to access the computer and conduct additional configurations.

Serial Console

The serial console port gives users a convenient way of connecting to the UC-8481's console utility. This method is particularly useful when using the computer for the first time. The signal is transmitted over a direct serial connection, so you do not need to know any of its 3 IP addresses to connect.

Use the serial console port settings shown below.

Baudrate	115200 bps
Parity	None
Data bits	8
Stop bits	1
Flow Control	None
Terminal	VT100

Once the connection is established, you may begin using the system once the root prompt appears.

```
root@Moxa:~#
```

Telnet Console

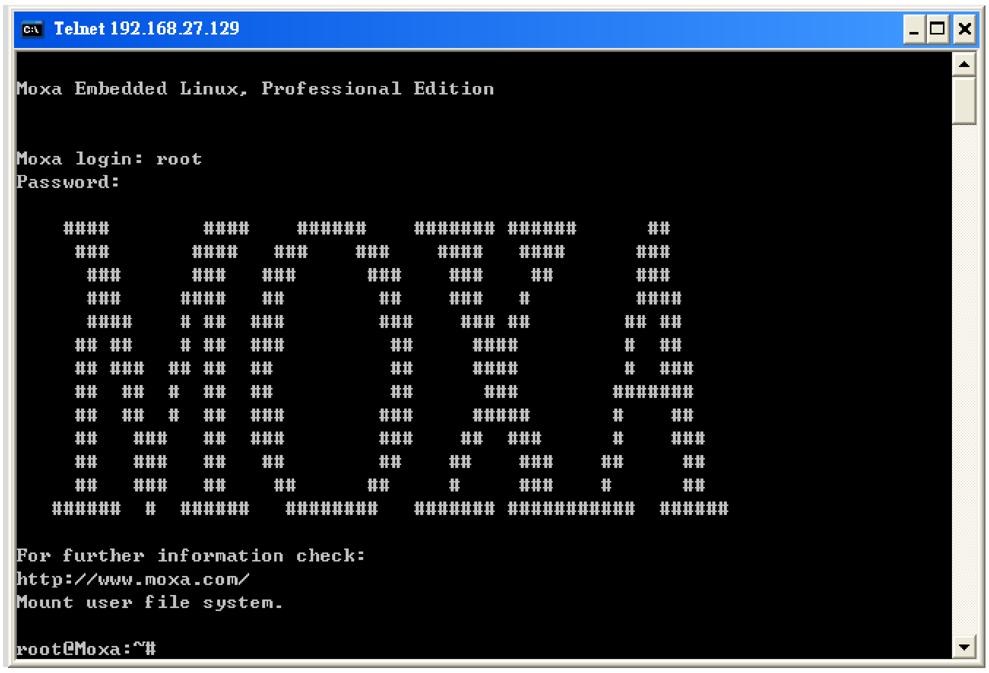
If you know at least one of the IP address and its netmask, you can telnet into the UC-8481's console utility. The default IP address and netmask for each of the ports is given below:

	Default IP Address	**Netmask**
LAN 1	DHCP	DHCP
LAN 2	192.168.4.127	255.255.255.0
WLANA	192.168.5.127	255.255.255.0

Use a crossover Ethernet cable to connect directly from your PC to the UC-8481. You should first modify your PC's IP address and netmask so that your PC is on the same subnet as LAN port to which the PC is connected. For example, if you connect to LAN 1, set your PC's IP address to 192.168.3.126 and the netmask to 255.255.255.0. If you connect to LAN 2, you should set your PC's IP address to 192.168.4.126 and netmask to 255.255.255.0.

To connect to a hub or switch connected on your local LAN, use a straight-through Ethernet cable. The default IP addresses and netmasks are shown above. To log in, type the Login name and password as requested. The default values are both root:

Login: **root**
Password: **root**



```
Moxa Embedded Linux, Professional Edition


Moxa login: root
Password:

    ####        ####    ######    ####### ######      ##
     ###       ####   ###    ###    ####   ####      ###
      ###      ###   ###      ###    ###    ##       ###
      ###     ####   ##        ##    ###   #         ####
      ####    # ##  ###        ###    ### ##        ## ##
     ## ##    # ##  ###         ##     ####         #  ##
     ## ###  ## ##  ##          ##     ####         #  ###
     ##  ##  #  ##  ##          ##      ###        #######
     ##  ##  #  ##  ###        ###     #####       #    ##
     ##   ###   ##  ###        ###    ##  ###      #    ###
     ##   ###   ##   ##        ##    ##    ###    ##     ##
     ##   ###   ##    ##      ##     #     ###    #      ##
   ######  #  ######   ########   ####### ###########  ######

For further information check:
http://www.moxa.com/
Mount user file system.

root@Moxa:~#
```

You may begin configuring network settings of the target computer once the terminal prompt appears, notifying you the bash command shell has loaded. Configuration instructions are given in the next section.

ATTENTION

Serial Console Reminder
Remember to choose VT100 as the terminal type. Use cable CBL-4PINDB9F-100, which comes with the UC-8481, to connect to the serial console port.
Telnet Reminder
When connecting to the UC-8481 over a LAN, you must configure your PC's IP address to match the subnet for which the UC-8481 is configured. If you have trouble connecting, re-check the IP settings; if trouble persists, do a cold re-boot after first unplugging (and re-plugging) the UC-8481's power cord.

SSH Console

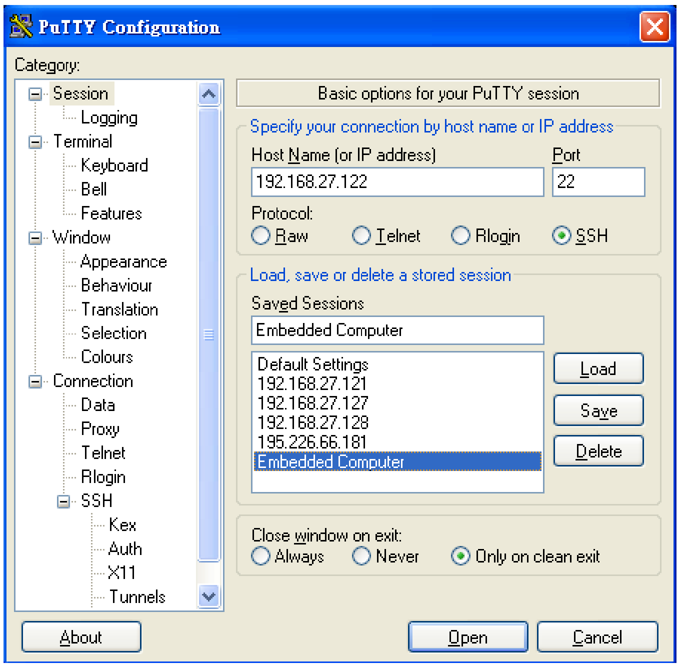
The UC-8481 supports an SSH console to provide users with more secure login options.

Windows Users

In a Windows environment users may use the free software PuTTY as their SSH interface. To set up an SSH console for the UC-8481, click on the following link to download the PuTTY freeware:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

The following figure shows a simple example of the required configuration:



Basic options for your PuTTY session

Specify your connection by host name or IP address

Host Name (or IP address): 192.168.27.122

Port: 22

Protocol: Raw Telnet Rlogin [✓ SSH]

Load, save or delete a stored session

Saved Sessions: Embedded Computer

Default Settings
192.168.27.121
192.168.27.127
192.168.27.128
195.226.66.181
Embedded Computer

Close window on exit: Always Never [✓ Only on clean exit]

Linux Users

From a Linux machine, ssh is simply called from the console prompt.

#ssh 192.168.4.127

Select yes to complete the connection.

```
[root@bee_notebook root]# ssh 192.168.4.127
The authenticity of host '192.168.4.127 (192.168.4.127)' can't be established.
RSA key fingerprint is 8b:ee:ff:84:41:25:fc:cd:2a:f2:92:8f:cb:1f:6b:2f.
Are you sure you want to continue connection (yes/no)? yes
```

If help is needed, type "ssh –h"; for more detailed assistance, type "man ssh".

NOTE SSH provides better security compared to Telnet for accessing the UC-8481's console utility over the network.

Configuring the Ethernet Interface

The network settings of the UC-8481 can be modified with the serial console, or online over the network.

Modifying Network Settings with the Serial Console

In this section, we use the serial console to configure network settings of the target computer.

Follow the instructions given in a previous section to access the console utility of the target computer via the serial console port, and then type

#cd /etc/network

to change to the directory where the network's configuration files are located.

```
root@Moxa:# cd /etc/network/
root@Moxa:/etc/network/#
```

Call up the vi text editor to begin editing the interfaces configuration file by typing:

#vi interfaces

You can configure the UC-8481's Ethernet ports for **static** or **dynamic** (DHCP) IP addresses.

Static IP addresses:

As shown below, 2 network addresses need to be modified: **address**, **network**, **netmask**, and **broadcast**. The default IP addresses are 192.168.4.127 for LAN2, 192.168.5.127 for WLAN-A, with default netmasks of 255.255.255.0.

```
# We always want the loopback interface.

auto eth0 eth1 eth2 lo
iface lo inet loopback

# embedded ethernet LAN2
iface eth0 inet static
      address 192.168.4.127
      network 192.168.4.0
      netmask 255.255.255.0
      broadcast 192.168.3.255

# embedded ethernet WLAN-A
iface eth1 inet static
      address 192.168.5.127
      network 192.168.5.0
      netmask 255.255.255.0
      broadcast 192.168.4.255
```

Dynamic IP Addresses:

By default, the UC-8481 is configured for "static" IP addresses. To configure LAN ports to request an IP address dynamically, replace **static** with **dhcp** and then delete the address, network, netmask, and broadcast lines.

Default Setting for LAN2	Dynamic Setting using DHCP
iface eth1 inet static address 192.168.4.127 network: 192.168.4.0 netmask 255.255.255.0 broadcast 192.168.3.255	iface eth1 inet dhcp

*Please note that the default value for LAN1 of the UC-8481 is DHCP.

```
Auto eth0 eth1 lo
iface lo inet loopback

iface eth0 inet dhcp

iface eth1 inet dhcp
```

After the boot settings of the LAN interface have been modified, issue the following command to activate the LAN settings immediately:

#/etc/init.d/networking restart

ATTENTION

After changing the IP settings, use the **networking restart** command to activate the new IP address.

Modifying Network Settings over the Network

IP settings can be activated over the network, but the new settings will not be saved to the flash ROM without modifying the file **/etc/network/interfaces**.

For example, type the command **#ifconfig eth0 192.168.1.1** to change the IP address of LAN1 to 192.168.1.1.

```
root@Moxa:# ifconfig eth0 192.168.1.1
root@Moxa:/etc/network/#
```

Configuring the WLAN Interface

WLAN Tools

Use the **iwlist** command to scan all available access points.

```
root@Moxa:~# iwlist wlan0 scanning
wlan0    Scan completed :
        Cell 01 - Address: 00:23:F8:76:6A:9D
                 Channel:1
                 Frequency:2.412 GHz (Channel 1)
                 Quality=22/70  Signal level=-88 dBm
                 Encryption key:off
                 ESSID:"BrcmAP0"
                 Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 18 Mb/s
                         24 Mb/s; 36 Mb/s; 54 Mb/s
                 Bit Rates:6 Mb/s; 9 Mb/s; 12 Mb/s; 48 Mb/s
                 Mode:Master
                 Extra:tsf=0000000e5b320197
                 Extra: Last beacon: 1820ms ago
                 IE: Unknown: 00074272636D415030
                 IE: Unknown: 010882848B962430486C
                 IE: Unknown: 030101
                 IE: Unknown: 050400010000
                 IE: Unknown: 2A0100
                 IE: Unknown: 2F0100
                 IE: Unknown: 32040C121860
                 IE: Unknown: DD090010180200F0040000
                 IE:Unknown:
DD180050F2020101800003A4000027A4000042435E0062322F00
        Cell 02 - Address: 40:4A:03:8D:A7:69
                 Channel:6
                Frequency:2.437 GHz (Channel 6)
                 Quality=40/70  Signal level=-70 dBm
                 Encryption key:on
                 ESSID:"MIS-WAP-1"
                 Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                         11 Mb/s; 12 Mb/s; 18 Mb/s
                 Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                 Mode:Master
                 Extra:tsf=00000017c9dc2b4c
                 Extra: Last beacon: 1450ms ago
                 IE: Unknown: 00094D49532D5741502D31
                 IE: Unknown: 010882848B0C12961824
                 IE: Unknown: 030106
                 IE: Unknown: 0706434E20010D14
                 IE: Unknown: 200100
                 IE: Unknown: 2A0102
                 IE: Unknown: 32043048606C
                 IE: Unknown: DD0900037F010100300000
        Cell 03 - Address: 40:4A:03:8D:A7:80
                 Channel:11
                 Frequency:2.462 GHz (Channel 11)
                 Quality=35/70  Signal level=-75 dBm
                 Encryption key:on
                 ESSID:"MIS-WAP-1"
                 Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s
                         11 Mb/s; 12 Mb/s; 18 Mb/s
                 Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                 Mode:Master
                 Extra:tsf=00000012ed3fc3ae
                 Extra: Last beacon: 1150ms ago
                 IE: Unknown: 00094D49532D5741502D31
                 IE: Unknown: 010882848B0C12961824
                 IE: Unknown: 03010B
                 IE: Unknown: 0706434E20010D14
                 IE: Unknown: 200100
                 IE: Unknown: 2A0105
                 IE: Unknown: 32043048606C
                 IE: Unknown: DD0900037F010100200000
root@Moxa:~#
```

Add the name (ssid) for the network you want to create/join. Use single quotes if there is a space in the name.

```
root@Moxa:~ # iwconfig wlan0 essid 'name'
```

Use **iwconfig** to set the key.

Examples:

```
iwconfig wlan0 key 0123-4567-89
iwconfig wlan0 key [3] 0123-4567-89
iwconfig wlan0 key s:password [2]
iwconfig wlan0 key [2]
iwconfig wlan0 key open
iwconfig wlan0 key off
iwconfig wlan0 key restricted [3] 0123456789
iwconfig wlan0 key 01-23 key 45-67 [4] key [4]
```

You may also use iwconfig to check the WLAN status:

```
root@Moxa:~ # iwconfig wlan0
```

wpa_supplicant

Use **/etc/wpa_supplicant0.conf** to set connection parameters for individual APs.

You may configure **/etc/wpa_supplicant0.conf** to use no security when associating with an AP:

```
network={
   ssid=”MOXASYS”
   key_mgmt=NONE
}
```

You may configure **/etc/wpa_supplicant0.conf** to use WEP security when associating with an AP:

```
network={
      ssid="MOXASYS"
      key_mgmt=NONE
      wep_key0=1234567890
      wep_key1="abcde"
      wep_key2="1234567890123"
      wep_tx_keyidx=0
      priority=5
}
```

You may configure **/etc/wpa_supplicant0.conf** to use WPA-PSK (WPA TKIP) security when associating with an AP:

```
network={
    ssid="MOXASYS"
    mode=infrastructure
    key_mgmt=WPA-PSK
    auth_alg=OPEN
    pairwise=TKIP
    group=TKIP
    psk="29101230"
}
```

You may configure **/etc/wpa_supplicant0.conf** to use WPA-PSK (WPA2 AES) security when associating with an AP:

```
network={
    ssid="MOXASYS"
    mode=infrastructure
    key_mgmt=WPA-PSK
    auth_alg=OPEN
    pairwise=CCMP
    group=CCMP
    psk="29101230"
}
```

Start wpa_supplicant as a daemon, so that it runs as a background process:

```
root@Moxa:~# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant0.conf
```

Check WLAN status with iwconfig:

```
root@Moxa:~ # iwconfig wlan0
```

You may also use either the **/etc/init.d/wpa0.sh** or the **/etc/init.d/wpa1.sh** script to force your wireless module to re-read the configuration file by restarting the background process after a connection has already been established.

First stop the card (this same command may be used at any time to turn off the wireless connection):

```
root@Moxa:~# /etc/init.d/wpa0.sh stop
```

Then restart it, using the same script:

```
root@Moxa:~# /etc/init.d/wpa0.sh start
```

You should also create an entry in the rc directory **/etc/rc.d/rc3.d/S99wpa0.sh** for auto-starting the wireless LAN service at boot-time:

```
root@Moxa:~# cd /etc/rc.d/rc3.d
root@Moxa:~# ln -sf ../../init.d/wpa0.sh S99wpa0.sh
root$Moxa:~# cd ../../rc.d/rc0.d
root@Moxa:~# ln -sf ../../init.d/wpa0.sh K01wpa0.sh
root$Moxa:~# cd ../../rc.d/rc1.d
root@Moxa:~# ln -sf ../../init.d/wpa0.sh K01wpa0.sh
root$Moxa:~# cd ../../rc.d/rc6.d
root@Moxa:~# ln -sf ../../init.d/wpa0.sh K01wpa0.sh
```

iw

Use **iw** to scan the wireless LAN.

```
root@Moxa:~ # iw dev wlan0 scan
```

Connect to the AP using the following command:

root@Moxa:~# iw dev <devname> connect [-w] <SSID> [<freq in MHz>] [<bssid>] [key 0:abcde d:1:6162636465]

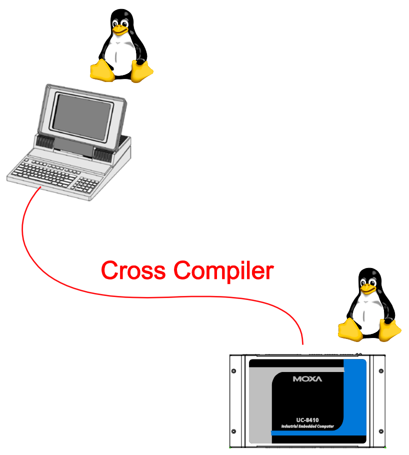
```
root@Moxa:/# iw wlan0 connect MOXASYSSW
```

Check the wlan0 status:

```
root@Moxa:~ # iw dev wlan0 link
```

Test Program—Developing Hello.c

1. Connect the UC-8481 to a Linux PC.
2. Install the Tool Chain (GCC & glibc).
3. Set the cross compiler and glibc environment variables.
4. Code and compile the program.
5. Download the program to the UC-8481 via FTP or NFS.
6. Debug the program
   → If bugs are found, return to Step 4.
   → If no bugs are found, continue with Step 7
7. Back up the user directory (distribute the program to additional UC-8481 units if needed).



Installing the Tool Chain (Linux)

The PC must have the Linux operating system pre-installed before installing the UC-8481 GNU Tool Chain. Redhat 7.3/8.0, Fedora core, and compatible versions are recommended. The Tool Chain requires about 600 MB of hard disk space on your PC. The UC-8481 Tool Chain software is located on the UC-8481 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount /dev/cdrom /mnt/cdrom#/mnt/cdrom/tool-chain/arm-linux_4.4.2-v4_Build_12032109.sh
```

The Tool Chain will be installed automatically on your Linux PC within a few minutes. Before compiling the program, be sure to set the following path first, since the Tool Chain files, including the compiler, link, library, and include files, are located in this directory.

```
PATH=/usr/local/arm-linux-4.4.2-v4/bin:$PATH
```

Setting the path allows you to run the compiler from any directory.

Checking the Flash Memory Space

The UC-8481 uses a specially designed root file system. Only /tmp, /etc, /home, /usr/local/bin, /usr/local/sbin, /usr/local/libexec, and /usr/local/lib directories are writable. Others are read-only. The writable directories are mounted on /dev/mtdblock4. If the /dev/mtdblock4 is full, you will not be able to save data to the Flash ROM. Use the following command to calculate the amount of "Available" flash memory:

```
/>df –h
```

```
root@Moxa:~# df -h
Filesystem              Size     Used Available Use% Mounted on
/dev/root              29.0M    12.6M    16.4M  44% /
none                    4.0M        0     4.0M   0% /dev
/dev/ram0             499.0K    23.0K   451.0K   5% /var
/dev/mtdblock4         512.0M     4.0M   508.0M   1% /tmp
/dev/mtdblock4         512.0M     4.0M   508.0M   1% /home
/dev/mtdblock4         512.0M     4.0M   508.0M   1% /etc
tmpfs                  32.0M        0    32.0M   0% /dev/shm
root@Moxa:~#
```

If there isn't enough "Available" space for your application, you will need to delete some existing files. To do this, connect your PC to the UC-8481 with the console cable, and then use the console utility to delete the files from the UC-8481's flash memory.

Compiling Hello.c

The CD contains several sample programs. Here we use **Hello.c** to show you how to compile and run your applications. Type the following commands from your PC to copy the files used for this example from the CD to your computer's hard drive:

```
# cd /tmp/
# mkdir example
# cp –r /mnt/cdrom/example/* /tmp/example
```

To compile the program, go to the **hello** subdirectory and issue the following commands:

```
#cd example/hello
#make
```

You should receive the following response:

```
[root@localhost hello]# make
xscale-linux-gcc -o hello-release hello.c
xscale-linux-strip -s hello-release
xscale-linux-gcc -ggdb -o hello-debug hello.c
[root@localhost hello]# _
```

Next, execute **hello.exe** to generate **hello-release** and **hello-debug**, which are described below:

hello-release—an IXP platform execution file (created specifically to run on the UC-8481)

hello-debug—an IXP platform GDB debug server execution file (see Chapter 5 for details about the GDB debug tool).

ATTENTION

Be sure to type the **#make** command from within the **/tmp/example/hello** directory, since the UC-8481's tool chain puts a specially designed **Makefile** in that directory. This special Makefile uses the xscale-linux-gcc compiler to compile the hello.c source code for the Xscale environment. If you type the **#make** command from any other directory, Linux will use the x86 compiler (for example, cc or gcc). Refer to Chapter 5 to see a Make file example.

Uploading and Running the "Hello" Program

Use the following command to upload **hello-release** to the UC-8481 via FTP.

From the PC, type:

```
#ftp 192.168.3.127
```

Use the bin command to set the transfer mode to binary mode, and then put command to initiate the file transfer:

```
ftp> cd /home
ftp> bin
ftp> put hello-release
```

From the UC-8481, type:

```
# chmod +x /home/hello-release
# ./home/hello-release
```

The word **Hello** will be printed on the screen.

```
root@Moxa:~# ./home/hello-release
Hello
```

3
Managing Embedded Linux

This chapter includes information about version control, deployment, updates, and peripherals.

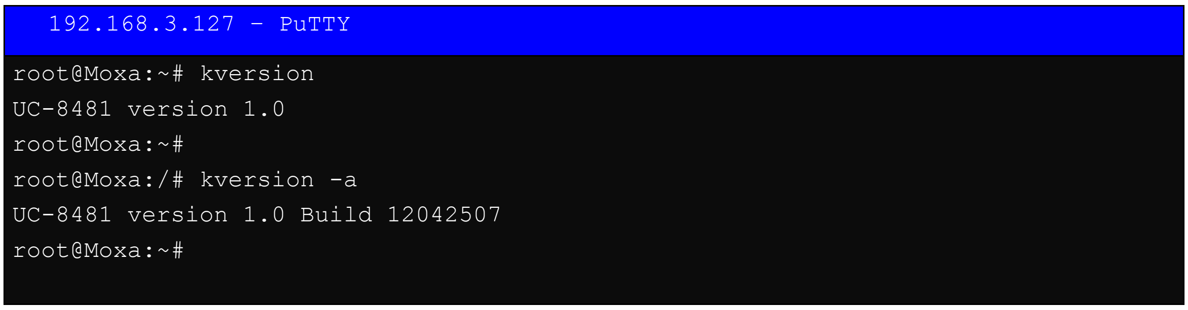
The following topics are covered in this chapter:

System Version Information

To determine the hardware capability of your UC-8481 and what kind of software functions are supported, check which version of the firmware your UC-8481 is running. Contact Moxa to determine the hardware version. You will need the **Production S/N** (Serial number), which is located on the UC-8481's bottom label.

To check the kernel version, type:

#kversion



```
root@Moxa:~# kversion
UC-8481 version 1.0
root@Moxa:~#
root@Moxa:/# kversion -a
UC-8481 version 1.0 Build 12042507
root@Moxa:~#
```

Firmware Upgrade

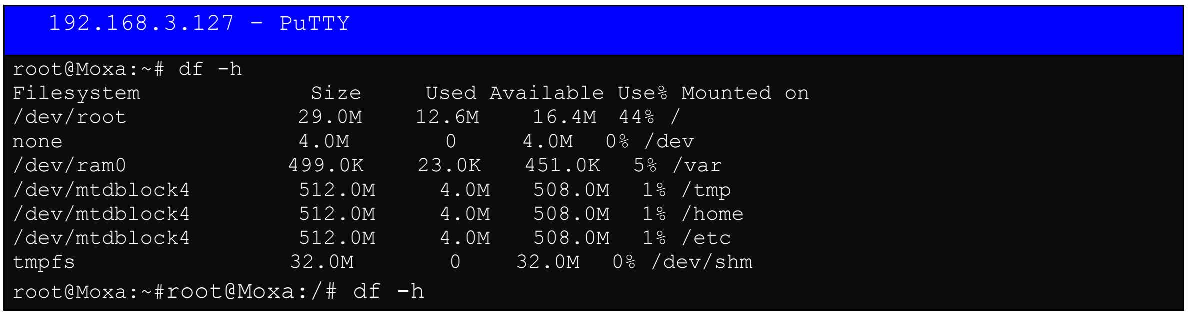
Upgrading the Firmware

The UC-8481's kernel, and root file system are combined into one firmware file, which can be downloaded from Moxa's website (www.moxa.com). The name of the file has the form **FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm**, with "a.b.c" indicating the firmware version and **YYMMDDHH** indicating the build date. To upgrade the firmware, download the firmware file to a PC, and then transfer the file to the UC-8481 unit via a serial console or Telnet console connection.

ATTENTION

Upgrading the firmware will erase all data on the Flash ROM
If you are using the **ramdisk** to store code for your applications, beware that updating the firmware will erase all of the data on the Flash ROM. You should back up your application files and data before updating the firmware.

Since different Flash disks have different sizes, it's a good idea to check the size of your Flash disk before upgrading the firmware, or before using the disk to store your application and data files. Use the **#df –h** command to list the size of each memory block, and how much free space is available in each block.

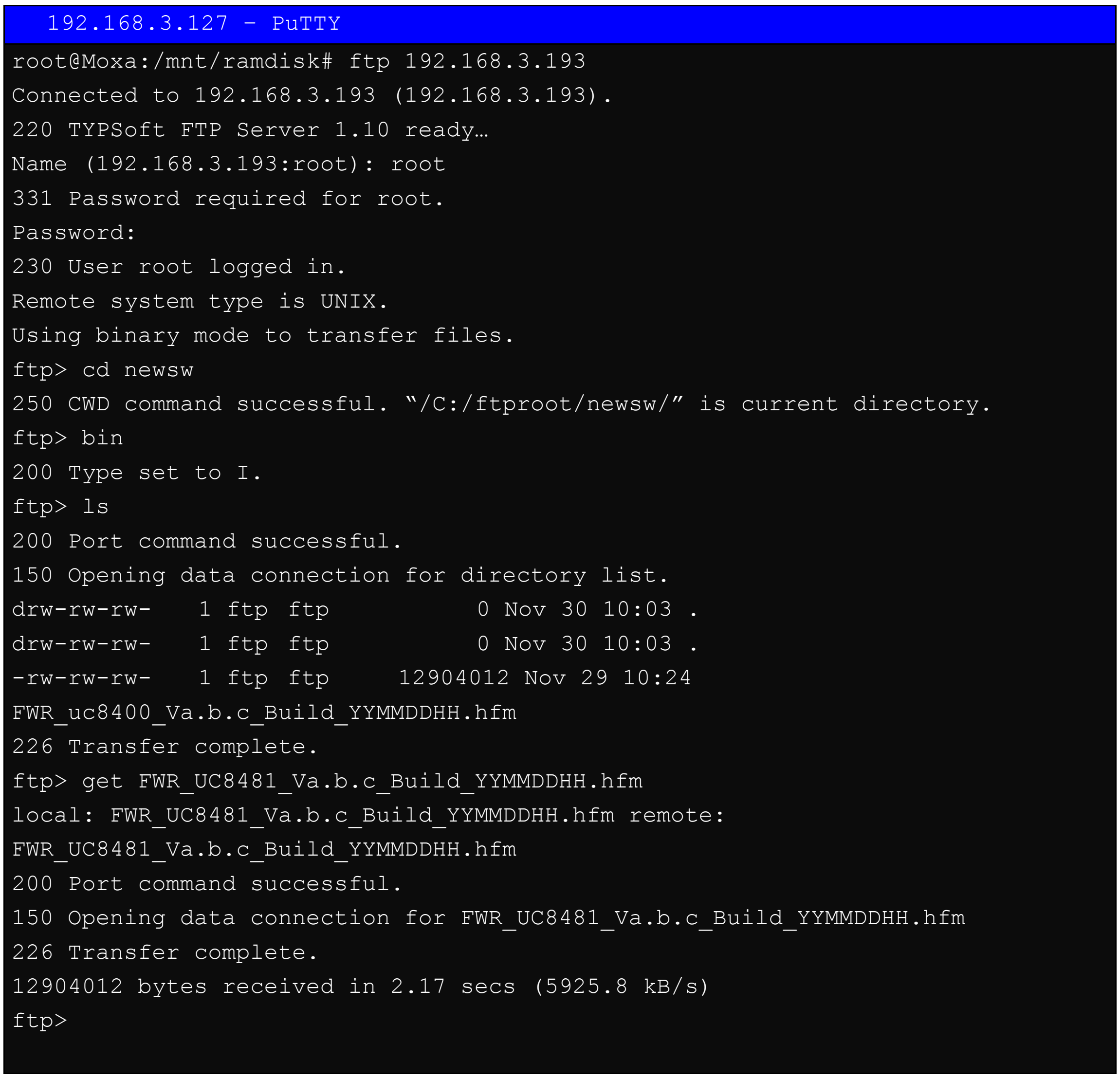


```
root@Moxa:~# df -h
Filesystem              Size     Used Available Use% Mounted on
/dev/root              29.0M    12.6M     16.4M  44% /
none                   4.0M        0      4.0M   0% /dev
/dev/ram0             499.0K    23.0K    451.0K   5% /var
/dev/mtdblock4         512.0M     4.0M    508.0M   1% /tmp
/dev/mtdblock4         512.0M     4.0M    508.0M   1% /home
/dev/mtdblock4         512.0M     4.0M    508.0M   1% /etc
tmpfs                 32.0M        0     32.0M   0% /dev/shm
root@Moxa:~#root@Moxa:/# df -h
```

The following instructions give the steps required to save the firmware file to the UC-8481's RAM disk, and then upgrade the firmware.

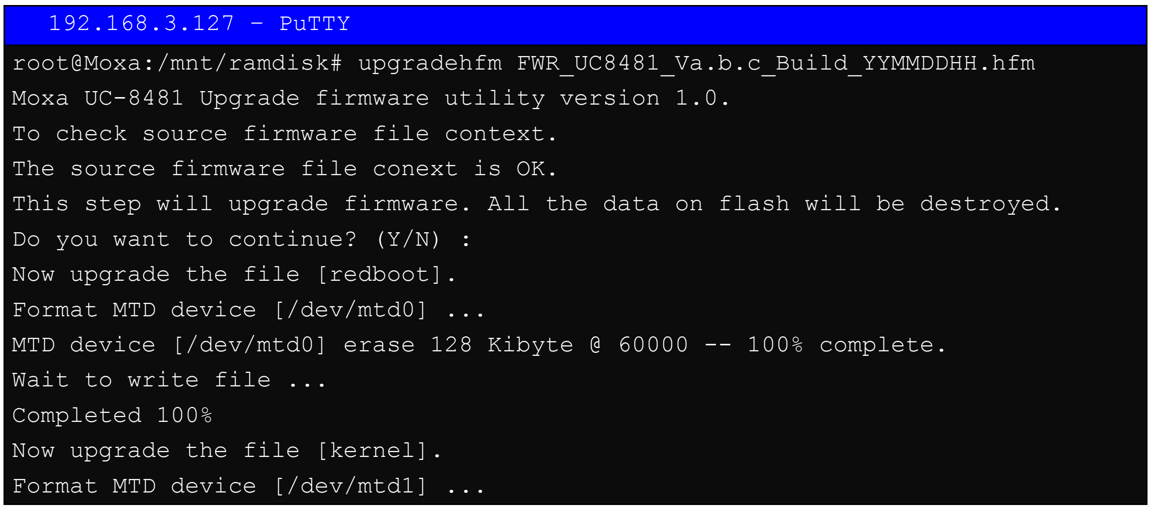
1. Type the following commands to use the UC-8481's built-in FTP client to transfer the firmware file (**FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm**) from the PC to the UC-8481:
   /dev/shm> ftp <destination PC's IP>

```
Login Name: xxxx
Login Password: xxxx
ftp> bin
ftp> get FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm
```



```
192.168.3.127 - PuTTY
root@Moxa:/mnt/ramdisk# ftp 192.168.3.193
Connected to 192.168.3.193 (192.168.3.193).
220 TYPSoft FTP Server 1.10 ready…
Name (192.168.3.193:root): root
331 Password required for root.
Password:
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd newsw
250 CWD command successful. "/C:/ftproot/newsw/" is current directory.
ftp> bin
200 Type set to I.
ftp> ls
200 Port command successful.
150 Opening data connection for directory list.
drw-rw-rw-   1 ftp  ftp          0 Nov 30 10:03 .
drw-rw-rw-   1 ftp  ftp          0 Nov 30 10:03 .
-rw-rw-rw-   1 ftp  ftp     12904012 Nov 29 10:24
FWR_uc8400_Va.b.c_Build_YYMMDDHH.hfm
226 Transfer complete.
ftp> get FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm
local: FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm remote:
FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm
200 Port command successful.
150 Opening data connection for FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm
226 Transfer complete.
12904012 bytes received in 2.17 secs (5925.8 kB/s)
ftp>
```

2. Next, use the **upgradehfm** command to upgrade the kernel and root file system:
   #upgradehfm FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm



```
192.168.3.127 - PuTTY
root@Moxa:/mnt/ramdisk# upgradehfm FWR_UC8481_Va.b.c_Build_YYMMDDHH.hfm
Moxa UC-8481 Upgrade firmware utility version 1.0.
To check source firmware file context.
The source firmware file conext is OK.
This step will upgrade firmware. All the data on flash will be destroyed.
Do you want to continue? (Y/N) :
Now upgrade the file [redboot].
Format MTD device [/dev/mtd0] ...
MTD device [/dev/mtd0] erase 128 Kibyte @ 60000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [kernel].
Format MTD device [/dev/mtd1] ...
```

```
MTD device [/dev/mtd1] erase 128 Kibyte @ 1a0000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [root-file-system].
Format MTD device [/dev/mtd2] ...
MTD device [/dev/mtd2] erase 128 Kibyte @ e00000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [directory].
Format MTD device [/dev/mtd5] ...
MTD device [/dev/mtd5] erase 128 Kibyte @ 20000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the new configuration file.
Upgrade the firmware is OK. Rebooting
```

Loading Factory Defaults

Press the reset button for more than 5 seconds to force the system to load the factory default settings. All files in the /home, /etc, /usr/local/bin, /usr/local/sbin, /usr/local/lib, /usr/local/libexec, and /tmp directories will be deleted. While pressing the reset button, during the first 5 seconds the ready-light will blink once each second. If you continue pressing the button after 5 seconds have passed, the ready-light will turn off, indicating that the factory defaults have been loaded.

ATTENTION

Reset-to-default will erase all data stored in /dev/mtdblock4

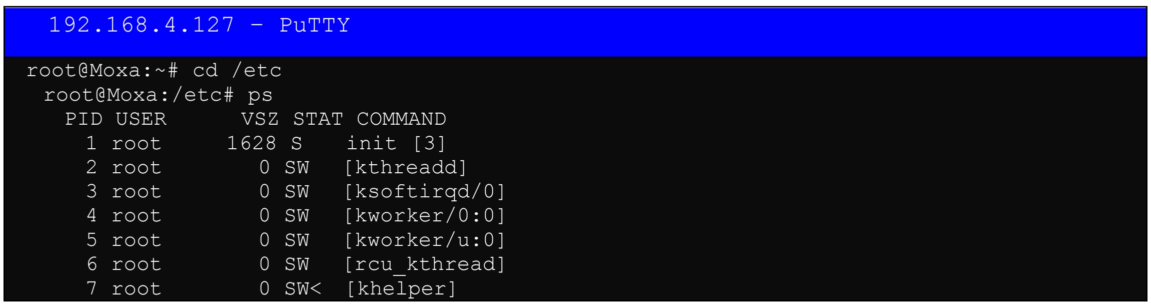
If you have stored data in the writable partition, you will need to back up these files before resetting the system to default. On the UC-8481, the directories /tmp, /etc, /usr/local/bin, /usr/local/sbin, /usr/local/lib, /usr/local/libexec, and /home are mounted on /dev/mtdblock4. This means that all of the data stored in these directories will be destroyed after resetting to default.

Enabling and Disabling Daemons

The following daemons are enabled when the UC-8481 boots up for the first time.

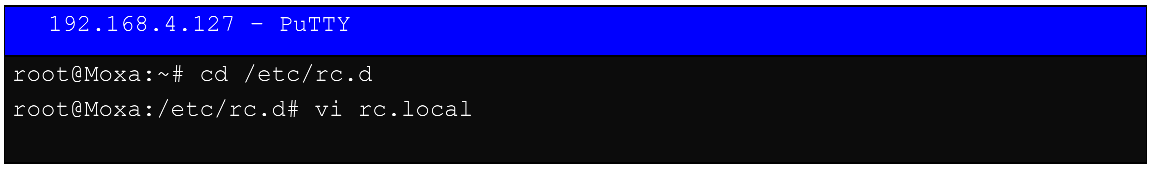
- **snmpd** SNMP Agent daemon
- **telnetd** Telnet Server / Client daemon
- **inetd** Internet Daemons
- **ftpd** FTP Server / Client daemon
- **sshd** Secure Shell Server daemon
- **httpd** Apache WWW Server daemon

1. Type the command "ps" to list all processes currently running.



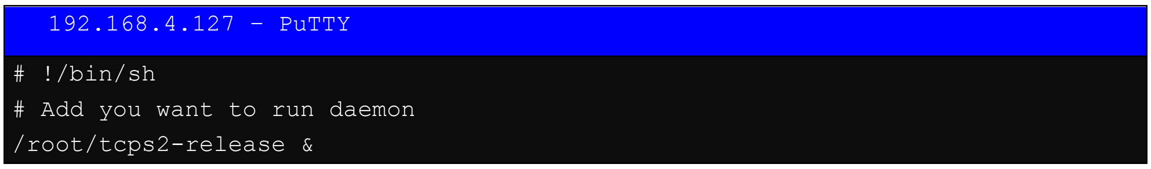
```
  192.168.4.127 - PuTTY
root@Moxa:~# cd /etc
 root@Moxa:/etc# ps
   PID USER       VSZ STAT COMMAND
     1 root      1628 S    init [3]
     2 root         0 SW   [kthreadd]
     3 root         0 SW   [ksoftirqd/0]
     4 root         0 SW   [kworker/0:0]
     5 root         0 SW   [kworker/u:0]
     6 root         0 SW   [rcu_kthread]
     7 root         0 SW<  [khelper]
    63 root        0 SW   [sync_supers]
    65 root        0 SW   [bdi-default]
    67 root        0 SW<  [kblockd]
    73 root        0 SW   [khubd]
    98 root        0 SW   [kswapd0]
    99 root        0 SW   [fsnotify_mark]
   100 root        0 SW<  [aio]
   101 root        0 SW<  [crypto]
   674 root        0 SW   [mtdblock0]
   679 root        0 SW   [mtdblock1]
   684 root        0 SW   [mtdblock2]
   689 root        0 SW   [mtdblock3]
   694 root        0 SW   [kworker/0:1]
   698 root        0 SW   [mtdblock4]
   738 root        0 SW<  [rpciod]
   746 root        0 SW<  [nfsiod]
   815 root        0 SW   [ocf_0]
   816 root        0 SW   [ocf_ret_0]
  1142 root        0 SW   [scsi_eh_0]
  1143 root        0 SW   [usb-storage]
  1160 root        0 SW<  [cfg80211]
  1169 root        0 SW   [kworker/u:2]
  1196 root        0 SW   [yaffs-bg-1]
  1240 root     1644 S    dhcpcd eth0
  1270 root     1692 S    /bin/inetd
  1275 bin      1612 S    /bin/portmap
  1278 root     2644 S    /bin/sh --login
  1284 root     1680 S    /bin/snmpd -c public
  1315 root     4064 S    /bin/sshd -6 -f /etc/ssh/sshd_config
  1317 root     4064 S    /bin/sshd -4 -f /etc/ssh/sshd_config
  1395 root     3072 R    ps
```

2. To run a private daemon, edit the file rc.local, as follows:
   #cd /etc/rc.d
   #vi rc.local



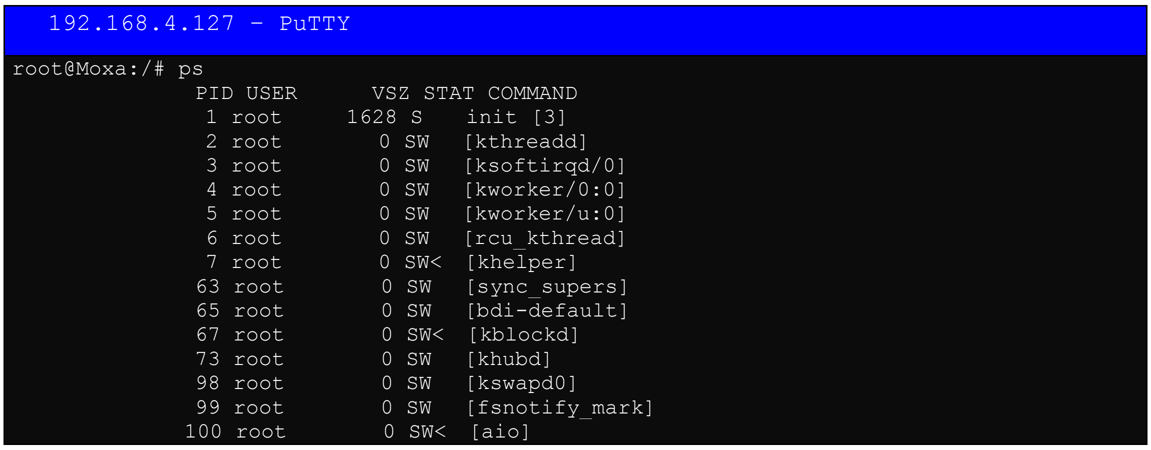
```
192.168.4.127 - PuTTY
root@Moxa:~# cd /etc/rc.d
root@Moxa:/etc/rc.d# vi rc.local
```

3. Next, use vi editor to edit your application program. We use the sample program **tcps2-release**, and set it to run in the background.

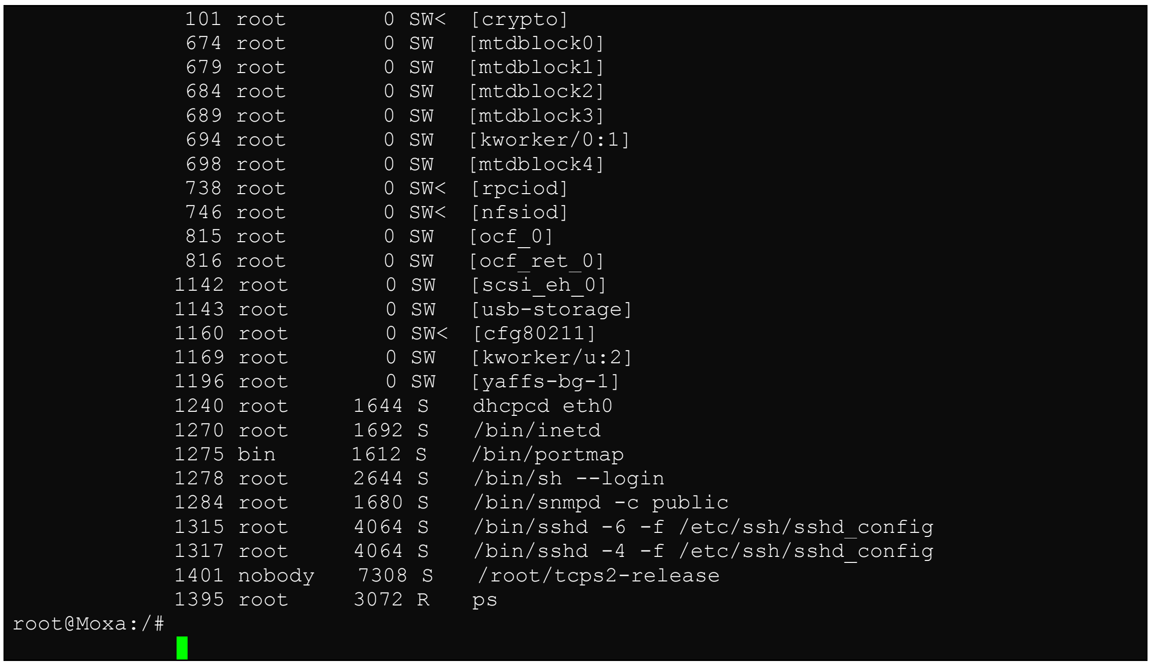


```
192.168.4.127 - PuTTY
# !/bin/sh
# Add you want to run daemon
/root/tcps2-release &
```

4. The following daemons will be enabled after you reboot the system.



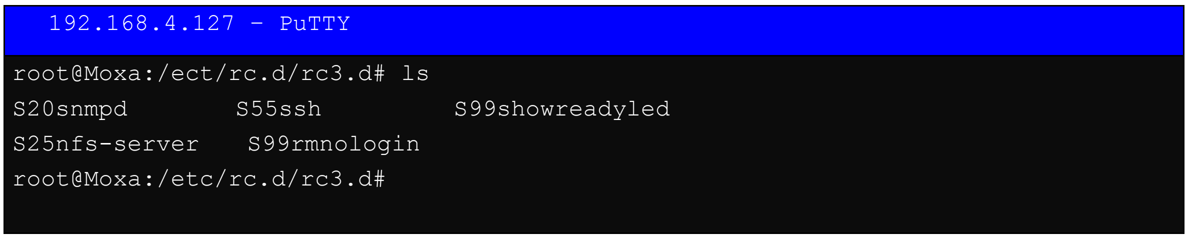
```
192.168.4.127 - PuTTY
root@Moxa:/# ps
              PID USER       VSZ STAT COMMAND
                1 root      1628 S    init [3]
                2 root         0 SW   [kthreadd]
                3 root         0 SW   [ksoftirqd/0]
                4 root         0 SW   [kworker/0:0]
                5 root         0 SW   [kworker/u:0]
                6 root         0 SW   [rcu_kthread]
                7 root         0 SW<  [khelper]
               63 root         0 SW   [sync_supers]
               65 root         0 SW   [bdi-default]
               67 root         0 SW<  [kblockd]
               73 root         0 SW   [khubd]
               98 root         0 SW   [kswapd0]
               99 root         0 SW   [fsnotify_mark]
              100 root         0 SW<  [aio]
```



```
              101 root        0 SW<  [crypto]
              674 root        0 SW   [mtdblock0]
              679 root        0 SW   [mtdblock1]
              684 root        0 SW   [mtdblock2]
              689 root        0 SW   [mtdblock3]
              694 root        0 SW   [kworker/0:1]
              698 root        0 SW   [mtdblock4]
              738 root        0 SW<  [rpciod]
              746 root        0 SW<  [nfsiod]
              815 root        0 SW   [ocf_0]
              816 root        0 SW   [ocf_ret_0]
             1142 root        0 SW   [scsi_eh_0]
             1143 root        0 SW   [usb-storage]
             1160 root        0 SW<  [cfg80211]
             1169 root        0 SW   [kworker/u:2]
             1196 root        0 SW   [yaffs-bg-1]
             1240 root     1644 S    dhcpcd eth0
             1270 root     1692 S    /bin/inetd
             1275 bin      1612 S    /bin/portmap
             1278 root     2644 S    /bin/sh --login
             1284 root     1680 S    /bin/snmpd -c public
             1315 root     4064 S    /bin/sshd -6 -f /etc/ssh/sshd_config
             1317 root     4064 S    /bin/sshd -4 -f /etc/ssh/sshd_config
             1401 nobody   7308 S    /root/tcps2-release
             1395 root     3072 R    ps
root@Moxa:/#
```

Setting the Run-Level

In this section, we outline the steps you should take to set the Linux run-level and execute requests. Use the following command to enable or disable settings:



```
   192.168.4.127 - PuTTY
root@Moxa:/ect/rc.d/rc3.d# ls
S20snmpd        S55ssh          S99showreadyled
S25nfs-server    S99rmnologin
root@Moxa:/etc/rc.d/rc3.d#
```

#cd /etc/rc.d/init.d

Edit a shell script to execute **/root/tcps2-release** and save to **tcps2** as an example.

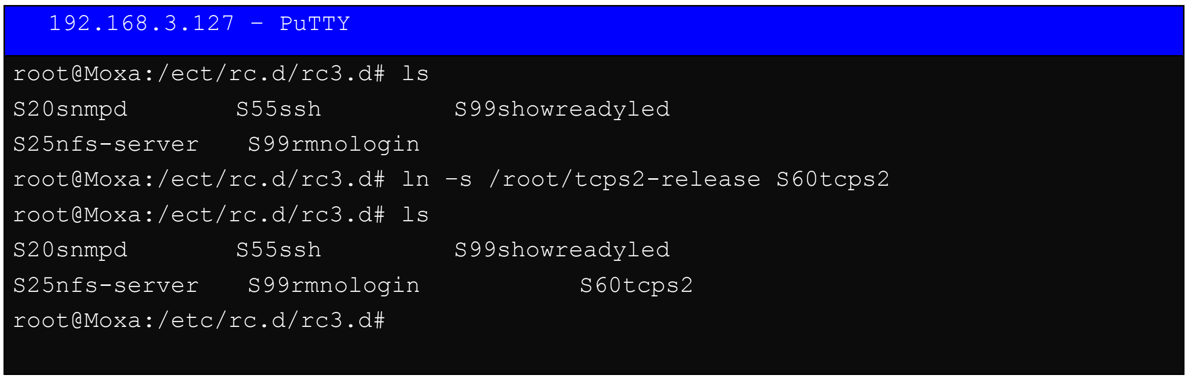
#cd /etc/rc.d/rc3.d
#ln -s /etc/rc.d/init.d/tcps2 S60tcps2

SxxRUNFILE stands for

S: start the run file while linux boots up.

xx: a number between 00-99. Smaller numbers have a higher priority.

RUNFILE: the file name.



```
   192.168.3.127 - PuTTY
root@Moxa:/ect/rc.d/rc3.d# ls
S20snmpd        S55ssh          S99showreadyled
S25nfs-server    S99rmnologin
root@Moxa:/ect/rc.d/rc3.d# ln -s /root/tcps2-release S60tcps2
root@Moxa:/ect/rc.d/rc3.d# ls
S20snmpd        S55ssh          S99showreadyled
S25nfs-server    S99rmnologin           S60tcps2
root@Moxa:/etc/rc.d/rc3.d#
```

KxxRUNFILE stands for

K: start the run file while Linux shuts down or halts.

xx: a number between 00-99. Smaller numbers have a higher priority.
RUNFILE: the file name.

To remove the daemon, use the following command to remove the run file from **/etc/rc.d/rc3.d**:

```
#rm –f /etc/rc.d/rc3.d/S60tcps2
```

Setting the System Time

There are two ways to support the timezone configuration on a Moxa embedded computer. One is using the TZ variable. The other is using /etc/localtime.

TZ variable

If you would like to use TZ variable, link to the following URL for all configurations and settings.

http://www.mkssoftware.com/docs/man5/timezone.5.asp

/etc/timezone

The local timezone is stored in /etc/localtime and is used by GNU Library for C (glibc) if the TZ environment variable is not set. This file is either a copy of /usr/share/zoneinfo/ tree or a symbolic link to it.

The UC-8481 does not provide /usr/share/zoneinfo/ files, so you need to copy a time zone information file to the UC-8481 and write over the original local time file.

1. The /usr/share/zoneinfo folder on a PC that is running standard Linux contains several time zone information files.
2. Copy the time zone file that you want to use to the UC-8481 to overwrite the original /etc/localtime file.
3. Type a date to check if the new time zone appears.

Adjusting the System Time

Setting the Time Manually

The UC-8481 has two time settings. One is the system time, and the other is the RTC (Real Time Clock) time kept by the UC-8481's hardware. Use the **#date** command to query the current system time or set a new system time. Use **#hwclock** to query the current RTC time or set a new RTC time.

Use the following command to query the system time:
```
#date
```

Use the following command to query the RTC time:
```
#hwclock
```

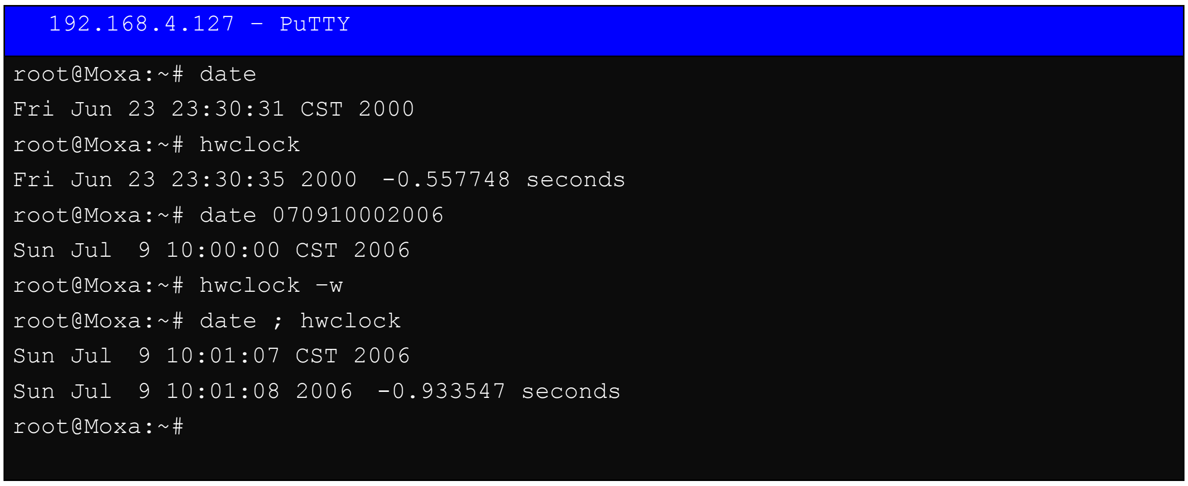
Use the following command to set the system time:
```
#date MMDDhhmmYYYY
```

MM = Month
DD = Date
hhmm = hour and minute
YYYY = Year

Use the following command to set the RTC time:
```
#hwclock –w
```

Write current system time to RTC

The following figure illustrates how to update the system time and set the RTC time.



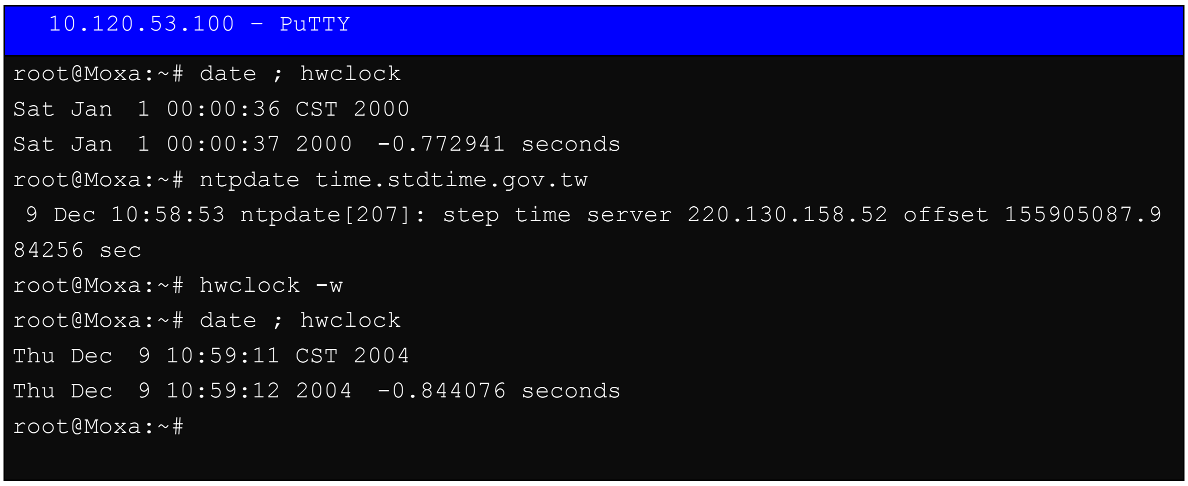
```
192.168.4.127 - PuTTY
root@Moxa:~# date
Fri Jun 23 23:30:31 CST 2000
root@Moxa:~# hwclock
Fri Jun 23 23:30:35 2000  -0.557748 seconds
root@Moxa:~# date 070910002006
Sun Jul  9 10:00:00 CST 2006
root@Moxa:~# hwclock -w
root@Moxa:~# date ; hwclock
Sun Jul  9 10:01:07 CST 2006
Sun Jul  9 10:01:08 2006  -0.933547 seconds
root@Moxa:~#
```

NTP Client

The UC-8481 has a built-in NTP (Network Time Protocol) client that is used to initialize a time request to a remote NTP server. Use **#ntpdate <NTP server>** to update the system time.

```
#ntpdate time.stdtime.gov.tw
#hwclock -w
```

Visit http://www.ntp.org for more information about NTP and NTP server addresses.



```
10.120.53.100 - PuTTY
root@Moxa:~# date ; hwclock
Sat Jan  1 00:00:36 CST 2000
Sat Jan  1 00:00:37 2000  -0.772941 seconds
root@Moxa:~# ntpdate time.stdtime.gov.tw
 9 Dec 10:58:53 ntpdate[207]: step time server 220.130.158.52 offset 155905087.9
84256 sec
root@Moxa:~# hwclock -w
root@Moxa:~# date ; hwclock
Thu Dec  9 10:59:11 CST 2004
Thu Dec  9 10:59:12 2004  -0.844076 seconds
root@Moxa:~#
```

ATTENTION

Before using the NTP client utility, check your IP and DNS settings to make sure that an Internet connection is available. Refer to Chapter 2 for instructions on how to configure the Ethernet interface, and see Chapter 4 for DNS setting information.

Updating the Time Automatically

In this subsection, we show how to use a shell script to update the time automatically.

Example shell script to update the system time periodically

```
#!/bin/sh
ntpdate time.nist.gov
# You can use the time server's ip address or domain
# name directly. If you use domain name, you must
# enable the domain client on the system by updating
# /etc/resolv.conf file.
hwclock –w
sleep 100      # Updates every 100 seconds. The min. time is 100 seconds. Change 100
to a larger number to update RTC less often.
```

Save the shell script using any file name. E.g., **fixtime**

How to run the shell script automatically when the kernel boots up

Copy the example shell script **fixtime** to directory **/etc/init.d**, and then use

chmod 755 fixtime to change the shell script mode. Next, use vi editor to edit the file **/etc/inittab**. Add the following line to the bottom of the file:

```
ntp : 2345 : respawn : /etc/init.d/fixtime
```

Use the command **#init q** to re-init the kernel.

Cron—Daemon to Execute Scheduled Commands

Start Cron from the directory **/etc/rc.d/rc.local**. It will return immediately, so you do not need to start it with an ampersand (&) to run in the background.

The Cron daemon will search **/etc/cron.d/crontab** for crontab files, which are named after accounts in /etc/passwd.

Cron wakes up every minute, and checks each command to see if it should be run in the current minute.

Modify the file **/etc/cron.d/crontab** to set up your scheduled applications. Crontab files have the following format:

mm	h	dom	mon	dow	user	command
min	hour	date	month	week	user	command
0-59	0-23	1-31	1-12	0-6 (0 is Sunday)		

The following example demonstrates how to use Cron.

The following steps show how to use cron to update the system time and RTC time every day at 8:00.

STEP1: Write a shell script named fixtime.sh and save it to /home/.

```
#!/bin/sh
ntpdate time.nist.gov
hwclock –w
exit 0
```

STEP2: Change mode of fixtime.sh

```
#chmod 755 fixtime.sh
```

STEP3: Modify /etc/cron.d/crontab file to run fixtime.sh at 8:00 every day.

Add the following line to the end of crontab:

```
* 8 * * *   root    /home/fixtime.sh
```

STEP4: Enable the cron daemon manually.

```
#/etc/init.d/cron start
```

STEP5: Enable cron when the system boots up.

Add the following line in the file /etc/init.d/rc.local

```
#/etc/init.d/cron start
```

Connecting Peripherals

USB Mass Storage

The UC-8481 supports PNP (plug-n-play), and hot pluggability for connecting USB mass storage devices. UC-8481 has a mdev auto mount utility that eases the mount procedure. The mdev auto mount utility default only supports mount one partition automatically. For using a USB mass storage, you should create one partition on a host computer. Here we create one partition on the Linux host computer by the fdisk utility.

```
192.168.3.120 – Putty
Debian:~# fdisk /dev/sda
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (2-1011, default 2):
Using default value 2
Last cylinder or +size or +sizeM or +sizeK (2-1011, default 1011):
Using default value 1011

Command (m for help): p

Disk /dev/sda: 16 MB, 16056320 bytes
1 heads, 31 sectors/track, 1011 cylinders
Units = cylinders of 31 * 512 = 15872 bytes
Disk identifier: 0x6f20736b

   Device Boot      Start        End      Blocks   Id  System
/dev/sda1               2       1011       15655   83  Linux
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
Debian:~#
```

Then format the partition

```
192.168.3.120 – Putty
Debian:~# mke2fs /dev/sdb1
```

After that you can remove the external storage and connected it to UC-8481. The first connected USB mass storage device will be mounted automatically by mount to /mnt/sdc, and the second device will be mounted automatically to /mnt/sdd. UC-8481 will be un-mounted automatically with umount when the device is disconnected.

ATTENTION

Remember to type the **#sync** command before you disconnect the USB mass storage device. If you don't issue the command, you may lose some data.

Remember to exit the /mnt/sdc or /mnt/sdd directory when you disconnect the USB mass storage device. If you stay in /mnt/sdc or /mnt/sdd, the auto un-mount process will fail. If that happens, type #umount /mnt/sdc to un-mount the USB device manually.

UC-8481 only supports certain types of flash disk USB Mass Storage device. Some the USB flash disks and hard disks may not **be compatible with UC-8481. Check compatibility issues before you purchase a USB device to connect to UC-8481.**
/etc/mdev.conf is the mdev configuration file.

```
sd[a-z][1] 0:0 600 *(/sbin/automount.sh $MDEV)
```

It automatically run /sbin/automount.sh to mount the external storage.

```
#!/bin/sh

DISKNAME=`echo $1|cut -c1-3`
MPATH=/var/$DISKNAME
MPATHLOG=/dev/shm/$1

#echo $1 > /dev/shm/mountpath.log

if [ "$1" == "" ]; then
        echo .automount.sh parameter is none.
        exit 1
fi
     if [ -e "$MPATH" ]; then
             #echo "$MPATH" > /dev/shm/umount.log
             umount $MPATH
             rm -rf $MPATH
     else
             #echo $MPATH > /dev/shm/mount.log
             mkdir -p $MPATH
             mount /dev/$1 $MPATH
     fi
     exit 0
```

If you need to customize the mounting script, you should change root file system partition to writable mode for editing /sbin/automount.sh script.

```
     root@Moxa:/# mount -o remount,rw /
     root@Moxa:/# vi /sbin/automount.sh
root@Moxa:/# umount /
```

CF Mass Storage

The UC-8481 Embedded Computer does not support CompactFlash hot swap and PnP (Plug and Play) function. It is necessary to remove power source first before inserting or removing the CompactFlash card.The UC-8481 CF card doesn't support PNP. This means that user need to mount the CF card manually after inserting the CF mass storage.

You can mount the CF card manually with the command below:

```
Moxa: ~# mkdir /home/sda
Moxa: ~# mount -t ext2 /dev/sda1 /home/sda
```

You should umount the CF mass storage before you remove the CF card.

```
Moxa: ~# umount /home/sda
```

Connecting to Cellular Networks and Enabling GPS

UC-8481 comes with an embedded cellular module, PH8, already installed. A convenient script, **/etc/init.d/3g.sh** is provided for dialing into the cellular network.

To dial the cellular modem, use the following command.

```
192.168.3.120 – Putty
Moxa:~# root@Moxa:~# /etc/init.d/3g.sh start
```

To check the ppp0 interface, use the following command.

```
192.168.3.120 – Putty
root@Moxa:~# ifconfig ppp0
ppp0      Link encap:Point-to-Point Protocol
         inet addr:111.80.163.116  P-t-P:10.64.64.64  Mask:255.255.255.255
         UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500  Metric:1
         RX packets:10 errors:0 dropped:0 overruns:0 frame:0
         TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:3
         RX bytes:82 (82.0 B)  TX bytes:241 (241.0 B)
```

To hang up the cellular connection, use the following command:

```
           192.168.3.120 – Putty
root@Moxa:~# /etc/init.d/3g.sh stop
```

To check if the point-to-point interface (ppp0) is still available, use the following command.

```
           192.168.3.120 – Putty
root@Moxa:~# ifconfig ppp0
ifconfig: ppp0: error fetching interface information: Device not found
```

In addition, you may configure options for point-to-point daemon (pppd). The option settings for pppd are located at **/etc/chatscripts/umts-connect.chat**. Options are entered as comments in the code; they may be enabled simply by deleting the "**#**" at the beginning of the line, and may be disabled by commenting them out again.

```
192.168.3.120 – Putty
TIMEOUT 10
ABORT   'BUSY'
ABORT   'NO ANSWER'
ABORT   'ERROR'
SAY     'Starting umts connect script\n'
''      ATZ
#  AT+CPOPS is the modifications AT commands for V3G:
#  AT_OPSUS=n n=0~5,
#  0: Only connect to GSM networks
#  1: Only connect to UMTS networks
#  2: If you have a choice - GPRS first
#  3: If you have a choice UMTS first
#  4: Which ever network you connect to stay with it
#  5: Automatic - let V3G decide
#  AT+COPS? shows the if the card is registered on a GPRS or a UMTS network.
#  EX: +COPS:0,0,"Vodafone UK",0  The end digital 0 means a GSM network.
#OK      AT_OPSYS=1
OK      ATE0V1
# Input the PIN number
#OK     AT+CPIN=0000
SAY     'Setting APN\n'
OK      AT+CGDCONT=1,"IP","Internet"
ABORT   'NO CARRIER'
SAY     'Dialing...\n'
OK      ATD*99#
CONNECT ''
```

Configuring the AT Commands

The AT command set for connecting the cellular module is located at **/etc/ppp/peers/chtumts**. You can add your own AT commands in the following the format.

```
192.168.3.120 – Putty

# File: /etc/ppp/peers/chtumts
/dev/ttyUSB3    # modem port used
115200       # speed
defaultroute  # use the cellular network for the default route
usepeerdns    # use the DNS servers from the remote network
#nodetach      # keep pppd in the foreground
nocrtscts      # hardware flow control
lock         # lock the serial port
noauth       # don't expect the modem to authenticate itself
local        # don't use Carrier Detect or Data Terminal Ready
#persist
#demand
#modem
debug

# Use the next two lines if you receive the dreaded messages:
#
#    No response to n echo-requests
#    Serial link appears to be disconnected.
#    Connection terminated.
#
lcp-echo-failure 4
lcp-echo-interval 65535
connect "/usr/sbin/chat -v -f /etc/chatscripts/umts-connect.chat"
disconnect     "/usr/sbin/chat -v -f /etc/chatscripts/umts-disconnect.chat"
```

Resetting the Cellular Module

In some situations, the cellular module may malfunction. You may reset it with the mx_pcie_reset utility:

```
           192.168.3.120 – Putty

root@Moxa:/# mx_pcie_reset
Reset the PCIe slot utility
Usage: mx_pcie_reset -s slot-number
        slot-number      - 1 ~ 3
```

To reset a module located in the UC-8481's slot 1, use the following command:

```
192.168.3.120 – Putty
root@Moxa:/# mx_pcie_reset -s 1
```

To reset a module in slot 2, use the following command:

```
192.168.3.120 – Putty
root@Moxa:/# mx_pcie_reset -s 2
```

To reset slot 3:

```
192.168.3.120 – Putty
root@Moxa:/# mx_pcie_reset -s 3
```

Reading GPS Information

The UC-8481 also comes with a GPS module which is controlled by the cdc_acm module. GPS data is read from **/dev/ttyACM0**:

```
192.168.3.120 – Putty
root@Moxa:~# cat /dev/ttyACM0
$GPTXT,01,01,02,u-blox ag - www.u-blox.com*50

$GPTXT,01,01,02,HW  UBX-G60xx  00040007 F75FFFFFf*36

$GPTXT,01,01,02,EXT CORE 6.00 (39627) Mar 30 2010 10:39:12*49

$GPTXT,01,01,02,ROM BASE 6.02 (36023) Oct 15 2009 16:52:08*58

$GPTXT,01,01,02,MOD LEA-6R-0*37

$GPTXT,01,01,02,DR 6R C0 1.00*73

$GPTXT,01,01,02,ANTSUPERV=AC SD PDoS SR*20

$GPTXT,01,01,02,ANTSTATUS=OK*3B

$GPRMC,,V,,,,,,,,,,N*53

$GPVTG,,,,,,,,,N*30

$GPGGA,,,,,,0,00,99.99,,,,,,*48

$GPGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99*30

$GPGSV,1,1,00*79

$GPGLL,,,,,,V,N*64
…
```

You also can use gpsd to listen to GPS data over **/dev/ttyACM0**:

```
192.168.3.120 – Putty
root@Moxa:/# gpsd -N -D5 -F /var/run/gpsd.sock /dev/ttyACM0
```

You may use gpspipe to read the data from gpsd:

```
192.168.3.120 – Putty
root@Moxa:/# gpspipe -r
```

4
Managing Communication

In this chapter, we explain how to configure the UC-8481's various communication functions.

The following topics are covered in this chapter:

Telnet/FTP

In addition to supporting Telnet client/server and FTP client/server, the UC-8481 also supports SSH and sftp client/server. To enable or disable the Telnet/ftp server, you first need to edit the file **/etc/inetd.conf**.

Enabling the Telnet/ftp server

The following example shows the default content of the file /etc/inetd.conf. The default is to enable the Telnet/ftp server:

```
discard dgram udp wait root /bin/discard
discard stream tcp nowait root /bin/discard
telnet stream tcp nowait root /bin/telnetd
ftp stream tcp nowait root /bin/ftpd -l
```

Disabling the Telnet/ftp server

Disable the daemon by typing '**#**' in front of the first character of the row to comment out the line.

DNS

The UC-8481 supports DNS client (but not DNS server). To set up DNS client, you need to edit three configuration files: **/etc/hosts**, **/etc/resolv.conf**, and **/etc/nsswitch.conf**.

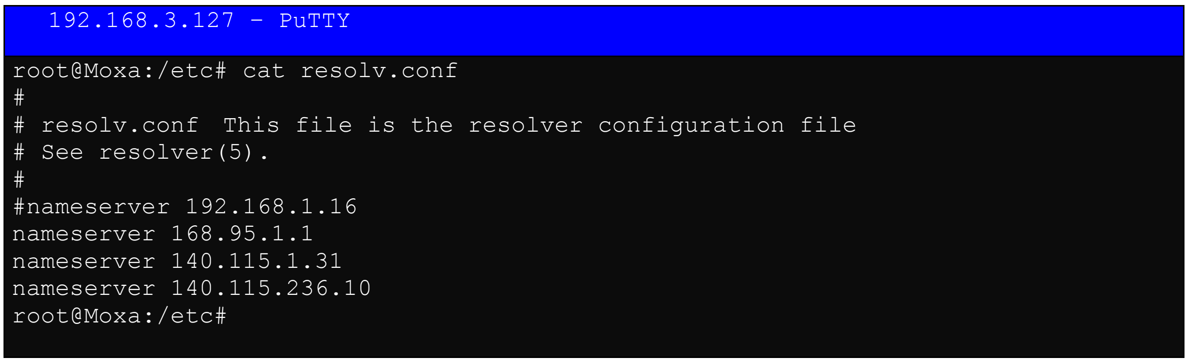
/etc/hosts

This is the first file that the Linux system reads to resolve the host name and IP address.

/etc/resolv.conf

This is the most important file that you need to edit when using DNS for the other programs. For example, before using **#ntpdate time.nist.gov** to update the system time, you will need to add the DNS server address to the file. Ask your network administrator which DNS server address you should use. The DNS server's IP address is specified with the "nameserver" command. For example, add the following line to /etc/resolv.conf if the DNS server's IP address is 168.95.1.1:

```
nameserver 168.95.1.1
```



```
root@Moxa:/etc# cat resolv.conf
#
# resolv.conf  This file is the resolver configuration file
# See resolver(5).
#
#nameserver 192.168.1.16
nameserver 168.95.1.1
nameserver 140.115.1.31
nameserver 140.115.236.10
root@Moxa:/etc#
```

/etc/nsswitch.conf

This file defines the sequence to resolve the IP address by using /etc/hosts file or /etc/resolv.conf.

iptables

iptables is an administrative tool for setting up, maintaining, and inspecting the Linux kernel's IP packet filter rule tables. Several different tables are defined, with each table containing built-in chains and user-defined chains.

Each chain is a list of rules that apply to a certain type of packet. Each rule specifies what to do with a matching packet. A rule (such as a jump to a user-defined chain in the same table) is called a "target."

The UC-8481 supports 3 types of iptables table: Filter tables, NAT tables, and Mangle tables:

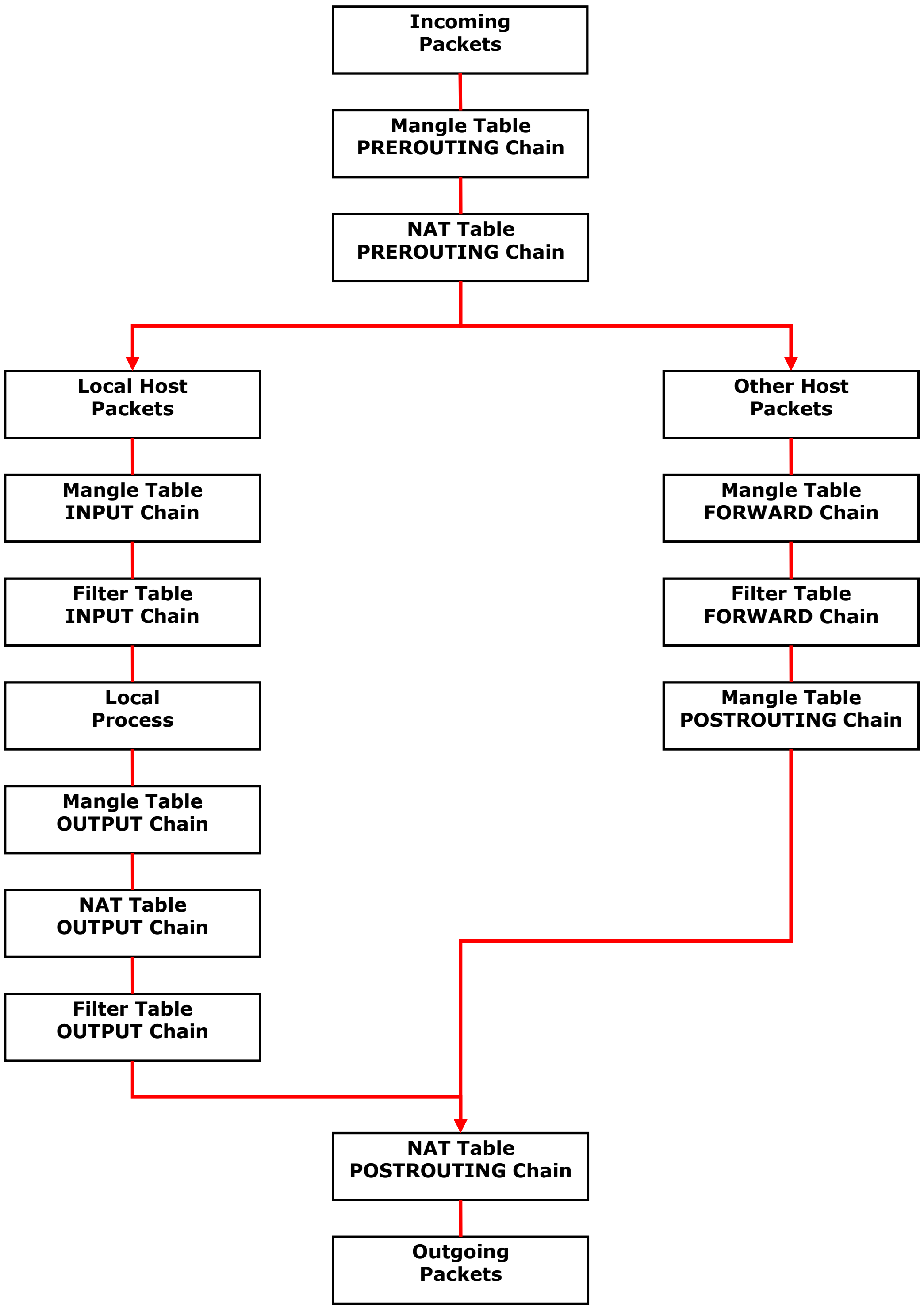
A. Filter Table—includes three chains:
INPUT chain
OUTPUT chain
FORWARD chain

B. NAT Table—includes three chains:
PREROUTING chain—transfers the destination IP address (DNAT)
POSTROUTING chain—works after the routing process and before the Ethernet device process to transfer the source IP address (SNAT)
OUTPUT chain—produces local packets
sub-tables

Source NAT (SNAT)—changes the first source packet IP address
Destination NAT (DNAT)—changes the first destination packet IP address
MASQUERADE—a special form for SNAT. If one host can connect to internet, then other computers that connect to this host can connect to the Internet when it the computer does not have an actual IP address.
REDIRECT—a special form of DNAT that re-sends packets to a local host independent of the destination IP address.

C. Mangle Table—includes two chains
PREROUTING chain—pre-processes packets before the routing process.
OUTPUT chain—processes packets after the routing process. It has three extensions—TTL, MARK, TOS.

The following figure shows the iptables hierarchy.


Incoming Packets
Mangle Table PREROUTING Chain
NAT Table PREROUTING Chain
Local Host Packets
Mangle Table INPUT Chain
Filter Table INPUT Chain
Local Process
Mangle Table OUTPUT Chain
NAT Table OUTPUT Chain
Filter Table OUTPUT Chain
Other Host Packets
Mangle Table FORWARD Chain
Filter Table FORWARD Chain
Mangle Table POSTROUTING Chain
NAT Table POSTROUTING Chain
Outgoing Packets

The UC-8481 supports the following sub-modules. Be sure to use the module that matches your application.

nf_conntrack	nf_conntrack_ftp	x_tables	xt_CLASSIFY
xt_MARK	xt_NFLOG	xt_NFQUEUE	xt_TCPMSS
xt_esp	xt_length	xt_limit	xt_mac
xt_mark	xt_multiport	xt_pkttype	xt_string
xt_tcpmss	xt_tcpudp	xt_u32	arp_tables
arpt_mangle	arptable_filter	ip_tables	ipt_CLUSTERIP
ipt_ECN	ipt_NETMAP	ipt_SAME	ipt_TTL
ipt_addrtype	ipt_ecn	ipt_iprange	ipt_recent
iptable_filter	iptable_mangle	iptable_nat	nf_conntrack_ipv4
nf_nat	nf_nat_ftp	nf_nat_snmp_basic	
		ipt_ah	
	ipt_MASQUERADE		
			ipt_tos
	ipt_REDIRECT		ipt_ttl
	ipt_REJECT		
	ipt_TOS		
ipt_LOG	ipt_ULOG	ipt_owner	

NOTE The UC-8481 does NOT support IPV6 and ipchains.

The basic syntax to enable and load an iptables module is as follows:

```
#lsmod
#modprobe ip_tables
#modprobe iptable_filter
```

Use **lsmod** to check if the ip_tables module has already been loaded in the UC-8481. Use **modprobe** to insert and enable the module.

NOTE iptables are the rules which the system uses for packet filtering and NAT. Take care when setting up the iptables rules. If the rules are not correct, remote hosts that connect via a LAN or PPP may be denied access. We recommend using the serial console to set up the iptables.

Click on the following links for more information about iptables.

http://www.linuxguruz.com/iptables/
http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html

Since the iptables command is very complex, to illustrate the iptables syntax we have divided our discussion of the various rules into three categories: **Observe and erase chain rules**, **Define policy rules**, and **Append or delete rules**.

Observe and erase chain rules

Usage:

```
# iptables [-t tables] [-L] [-n]
```

-t tables:	Table to manipulate (default: 'filter'); example: nat or filter.
-L [chain]: List	List all rules in selected chains. If no chain is selected, all chains are listed.
-n:	Numeric output of addresses and ports.

```
# iptables [-t tables] [-FXZ]
```

-F:	Flush the selected chain (all the chains in the table if none is listed).
-X:	Delete the specified user-defined chain.
-Z:	Set the packet and byte counters in all chains to zero.

Examples:

```
# iptables -L -n
```

In this example, since we do not use the -t parameter, the system uses the default 'filter' table. Three chains are included: INPUT, OUTPUT, and FORWARD. INPUT chains are accepted automatically, and all connections are accepted without being filtered.

```
#iptables –F
#iptables –X
#iptables -Z
```

Define policy for chain rules

Usage:

```
# iptables [-t tables] [-P] [INPUT, OUTPUT, FORWARD, PREROUTING, OUTPUT,
POSTROUTING] [ACCEPT, DROP]
```

-P:	Set the policy for the chain to the given target.
INPUT:	For packets coming into the UC-8481.
OUTPUT:	For locally-generated packets.
FORWARD:	For packets routed out through the UC-8481.
PREROUTING:	To alter packets as soon as they come in.
POSTROUTING:	To alter packets as they are about to be sent out.

Examples:

```
#iptables –P INPUT DROP
#iptables –P OUTPUT ACCEPT
#iptables –P FORWARD ACCEPT
# modprobe iptable_nat
#iptables –t nat –P PREROUTING ACCEPT
#iptables –t nat –P OUTPUT ACCEPT
#iptables -t nat –P POSTROUTING ACCEPT
```

In this example, the policy accepts outgoing packets and denies incoming packets.

Append or delete rules:

Usage:

```
# iptables [-t table] [-AI] [INPUT, OUTPUT, FORWARD] [-io interface] [-p tcp,
udp, icmp, all] [-s IP/network] [--sport ports] [-d IP/network] [--dport
ports] –j [ACCEPT. DROP]
```

-A:	Append one or more rules to the end of the selected chain.
-I:	Insert one or more rules in the selected chain as the given rule number.
-i:	Name of an interface via which a packet is going to be received.
-o:	Name of an interface via which a packet is going to be sent.
-p:	The protocol of the rule or of the packet to check.
-s:	Source address (network name, host name, network IP address, or plain IP address).
--sport:	Source port number.
-d:	Destination address.
--dport:	Destination port number.
-j:	Jump target. Specifies the target of the rules; i.e., how to handle matched packets. For example, ACCEPT the packet, DROP the packet, or LOG the packet.

Examples:

Example 1: Accept all packets from lo interface.

```
    # iptables –A INPUT –i lo –j ACCEPT
```

Example 2: Accept TCP packets from 192.168.0.1.

```
    # iptables –A INPUT –i eth0 –p tcp –s 192.168.0.1 –j ACCEPT
```

Example 3: Accept TCP packets from Class C network 192.168.1.0/24.

```
    # iptables –A INPUT –i eth0 –p tcp –s 192.168.1.0/24 –j ACCEPT
```

Example 4: Drop TCP packets from 192.168.1.25.

```
    # iptables -A INPUT -i eth0 -p tcp -s 192.168.1.25 -j DROP
```

Example 5: Drop TCP packets addressed for port 21.

```
    # modprobe xt_tcpudp
    # iptables -A INPUT -i eth0 -p tcp --dport 21 -j DROP
```

Example 6: Accept TCP packets from 192.168.0.24 to LAN1 port 137, 138, 139

```
    # iptables -A INPUT -i eth0 -p tcp -s 192.168.0.24 --dport 137:139 -j
ACCEPT
```

Example 7: Log TCP packets that visit the LAN1 port 25.

```
    # iptables -A INPUT -i eth0 -p tcp --dport 25 -j LOG
```

Example 8: Drop all packets from MAC address 01:02:03:04:05:06.

```
    # modprobe xt_mac
    # iptables -A INPUT -i eth0 -p all -m mac --mac-source 01:02:03:04:05:06 -j
DROP
```

NOTE: In Example 8, remember to issue the command #modprobe ipt_mac first to load the module ipt_mac

NAT

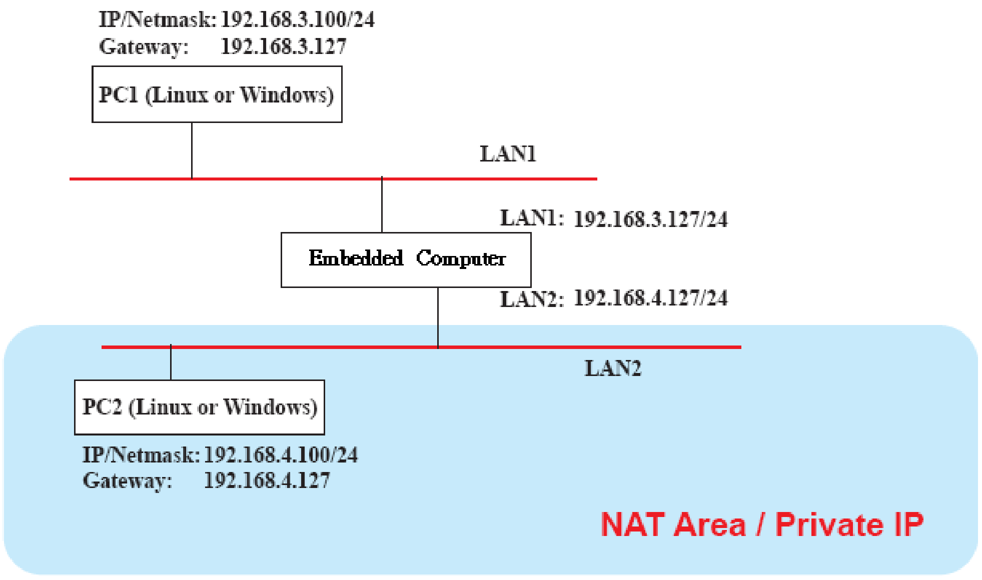
NAT (Network Address Translation) protocol translates IP addresses used on one network to different IP addresses used on another network. One network is designated the inside network and the other is the outside network. Typically, the UC-8481 connects several devices on a network and maps local inside network addresses to one or more global outside IP addresses, and un-maps the global IP addresses on incoming packets back into local IP addresses.

NOTE Click the following link for more information about iptables and NAT:

http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html

NAT Example

The IP address of LAN1 is changed to 192.168.3.127 (you will need to load the module ipt_MASQUERADE):



1. `#echo 1 > /proc/sys/net/ipv4/ip_forward`
2. `#modprobe ip_tables`
3. `#modprobe iptable_filter`
4. `#modprobe ip_conntrack`
5. `#modprobe iptable_nat`
6. `#modprobe ipt_MASQUERADE`
7. `#iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.3.127`
   or
   `#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE`

Enabling NAT at Bootup

In most real world situations, you will want to use a simple shell script to enable NAT when the UC-8481 boots up. The following script is an example.

```
#!/bin/bash
# If you put this shell script in the /home/nat.sh
# Remember to chmod 744 /home/nat.sh
# Edit the rc.local file to make this shell startup automatically.
# vi /etc/rc.d/rc.local
# Add a line in the end of rc.local /home/nat.sh
EXIF='eth0'  #This is an external interface for setting up a valid IP address.
EXNET='192.168.4.0/24'  #This is an internal network address.
# Step 1. Insert modules.
# Here 2> /dev/null means the standard error messages will be dump to null
device.
modprobe ip_tables  2> /dev/null
modprobe iptable_filter 2> /dev/null
modprobe iptable_nat 2> /dev/null
modprobe ip_conntrack  2> /dev/null
modprobe ip_conntrack_ftp  2> /dev/null
modprobe iptable_nat
modprobe ip_nat_ftp  2> /dev/null
# Step 2. Define variables, enable routing and erase default rules.
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
export PATH
echo "1" > /proc/sys/net/ipv4/ip_forward
/sbin/iptables -F
/sbin/iptables -X
/sbin/iptables -Z
/sbin/iptables -F -t nat
/sbin/iptables -X -t nat
/sbin/iptables -Z -t nat
/sbin/iptables -P INPUT   ACCEPT
/sbin/iptables -P OUTPUT  ACCEPT
/sbin/iptables -P FORWARD ACCEPT
/sbin/iptables -t nat -P PREROUTING  ACCEPT
/sbin/iptables -t nat -P POSTROUTING ACCEPT
/sbin/iptables -t nat -P OUTPUT     ACCEPT
# Step 3. Enable IP masquerade.
```

Dial-up Service—PPP

PPP (Point to Point Protocol) is used to run IP (Internet Protocol) and other network protocols over a serial link. PPP can be used for direct serial connections (using a null-modem cable) over a Telnet link, and links established using a modem over a telephone line.

Modem/PPP access is almost identical to connecting directly to a network through the UC-8481's Ethernet port. Since PPP is a peer-to-peer system, the UC-8481 can also use PPP to link two networks (or a local network to the Internet) to create a Wide Area Network (WAN).

NOTE Click on the following links for more information about ppp:

http://tldp.org/HOWTO/PPP-HOWTO/index.html
http://axion.physics.ubc.ca/ppp-linux.html

The pppd daemon is used to connect to a PPP server from a Linux system. For detailed information about pppd see the man page.

Example 1: Connecting to a PPP server over a simple dial-up connection

The following command is used to connect to a PPP server by modem. Use this command for old ppp servers that prompt for a login name (replace username with the correct name) and password (replace password with the correct password). Note that debug and defaultroute *192.1.1.17* are optional.

```
#pppd connect 'chat -v " " ATDT5551212 CONNECT" " ogin: username word: password'
/dev/ttyM0 115200 debug crtscts modem defaultroute
```

If the PPP server does not prompt for the username and password, the command should be entered as follows. Replace username with the correct username and replace password with the correct password.

```
#pppd connect 'chat -v " " ATDT5551212 CONNECT" " ' user username password
password
/dev/ttyM0 115200 crtscts modem
```

The pppd options are described below:

connect 'chat etc...'
This option gives the command to contact the PPP server. The 'chat' program is used to dial a remote computer. The entire command is enclosed in single quotes because pppd expects a one-word argument for the 'connect' option. The options for 'chat' are given below:

-v
verbose mode; log what we do to syslog

" "
Double quotes—don't wait for a prompt, but instead do (note that you must include a space after the second quotation mark)

ATDT5551212
Dial the modem, and then ...

CONNECT
Wait for an answer.

" "
Send a return (null text followed by the usual return)

ogin: username word: password
Log in with username and password.

Refer to the chat man page, chat.8, for more information about the chat utility.

/dev/
Specify the callout serial port.

115200
The baudrate.

Debug
Log status in syslog.

Crtscts
Use hardware flow control between the computer and modem (at 115200 this is a must).

Modem
Indicates that this is a modem device; pppd will hang up the phone before and after making the call.

Defaultroute
Once the PPP link is established, make it the default route; if you have a PPP link to the Internet, this is probably what you want.

192.1.1.17
This is a degenerate case of a general option of the form x.x.x.x:y.y.y.y. Here x.x.x.x is the local IP address and y.y.y.y is the IP address of the remote end of the PPP connection. If this option is not specified, or if just one side is specified, then x.x.x.x defaults to the IP address associated with the local machine's hostname (located in **/etc/hosts**), and y.y.y.y is determined by the remote machine.

Example 2: Connecting to a PPP server over a hard-wired link

If a username and password are not required, use the following command (note that noipdefault is optional):

```
#pppd connect 'chat –v" " " " ' noipdefault /dev/ttyM0 19200 crtscts
```

If a username and password is required, use the following command (note that noipdefault is optional, and root is both the username and password):

```
#pppd connect 'chat –v" " " " ' user root password root noipdefault
/dev/ttyM0 19200 crtscts
```

How to check the connection

Once you've set up a PPP connection, there are some steps you can take to test the connection. First, type:

```
/sbin/ifconfig
```

(The folder **ifconfig** may be located elsewhere, depending on your distribution.) You should be able to see all of the network interfaces that are UP. ppp0 should be one of them, and you should recognize the first IP address as your own In addition, the "P-t-P address" (or point-to-point address) is the address of your server. Here's what it looks like on one machine:

lo	Link encap Local Loopback
	inet addr 127.0.0.1 Bcast 127.255.255.255 Mask 255.0.0.0
	UP LOOPBACK RUNNING MTU 2000 Metric 1
	RX packets 0 errors 0 dropped 0 overrun 0
ppp0	Link encap Point-to-Point Protocol
	inet addr 192.76.32.3 P-t-P 129.67.1.165 Mask 255.255.255.0
	UP POINTOPOINT RUNNING MTU 1500 Metric 1
	RX packets 33 errors 0 dropped 0 overrun 0
	TX packets 42 errors 0 dropped 0 overrun 0

Now, type:

```
ping z.z.z.z
```

where z.z.z.z is the address of your name server. This should work. Here's what the response could look like:

waddington:~$p ping 129.67.1.165	
PING 129.67.1.165 (129.67.1.165): 56 data bytes	
64 bytes from 129.67.1.165: icmp_seq=0 ttl=225 time=268 ms	
64 bytes from 129.67.1.165: icmp_seq=1 ttl=225 time=247 ms	
64 bytes from 129.67.1.165: icmp_seq=2 ttl=225 time=266 ms	
^C	
--- 129.67.1.165 ping statistics ---	
3 packets transmitted, 3 packets received, 0% packet loss	
round-trip min/avg/max = 247/260/268 ms	
waddington:~$	

Try typing:

```
netstat -nr
```

You should see three routes, similar to the following:

Kernel routing table						
Destination	Gateway	Genmask	Flags	Metric	Ref	Use
iface						
129.67.1.165	0.0.0.0	255.255.255.255	UH	0	0	6
ppp0						
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0 lo
0.0.0.0	129.67.1.165	0.0.0.0	UG	0	0	6298
ppp0						

If your output looks similar, but does not have the destination 0.0.0.0 line (which refers to the default route used for connections), you may have run pppd without the 'defaultroute' option. At this point you can try using Telnet, ftp, or finger, bearing in mind that you will need to use numeric IP addresses unless you've set up /etc/resolv.conf correctly.

Setting up a Machine for Incoming PPP Connections

This first example applies to using a modem, and requires authorization with a username and password.

```
pppd/dev/ttyM0 115200 crtscts modem 192.168.16.1:192.168.16.2 login auth
```

You should also add the following line to the file **/etc/ppp/pap-secrets**:

```
*    *    ""    *
```

The first star (*) lets everyone login. The second star (*) lets every host connect. The pair of double quotation marks ("") is to use the file **/etc/passwd** to check the password. The last star (*) is to let any IP connect.

The following example does not check the username and password:

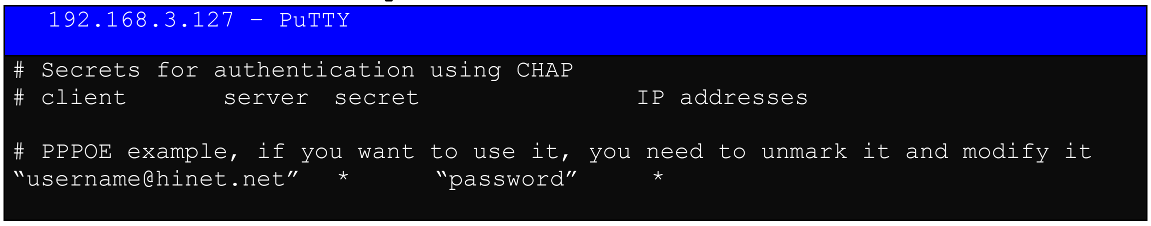
`pppd/dev/ttyM0 115200 crtscts modem 192.168.16.1:192.168.16.2`

PPPoE

1. Connect the UC-8481's LAN port to an ADSL modem with a cross-over cable, HUB, or switch.

2. Log in to the UC-8481 as the root user.

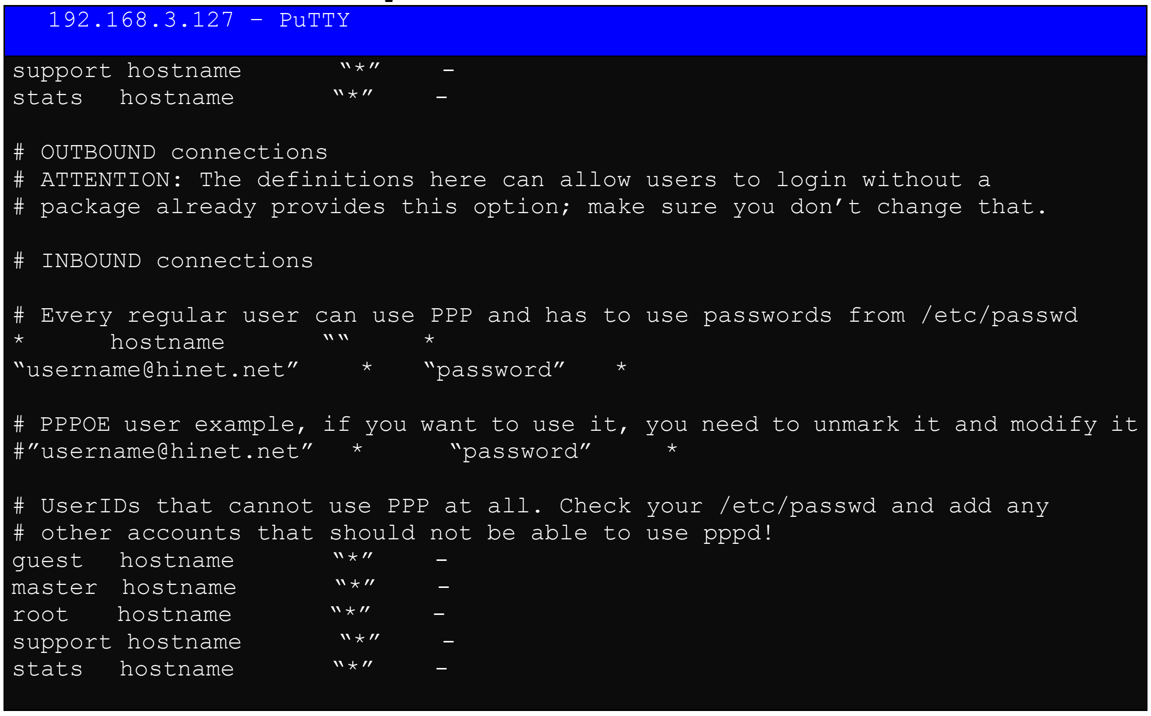
3. Edit the file **/etc/ppp/chap-secrets** and add the following:
   `"username@hinet.net" *   "password"  *`



```
   192.168.3.127 - PuTTY

# Secrets for authentication using CHAP
# client       server  secret                 IP addresses

# PPPOE example, if you want to use it, you need to unmark it and modify it
"username@hinet.net"   *      "password"     *
```

"username@hinet.net" is the username obtained from the ISP to log in to the ISP account. **"password"** is the corresponding password for the account.

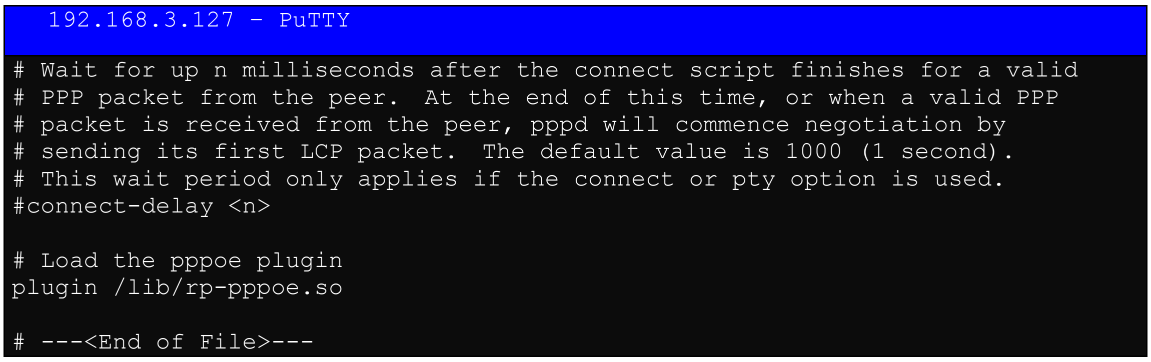
4. Edit the file **/etc/ppp/pap-secrets** and add the following:
   `"username@hinet.net" *   "password"  *`



```
   192.168.3.127 - PuTTY

support hostname       "*"     -
stats   hostname       "*"     -

# OUTBOUND connections
# ATTENTION: The definitions here can allow users to login without a
# package already provides this option; make sure you don't change that.

# INBOUND connections

# Every regular user can use PPP and has to use passwords from /etc/passwd
*      hostname       ""      *
"username@hinet.net"    *    "password"    *

# PPPOE user example, if you want to use it, you need to unmark it and modify it
#"username@hinet.net"   *      "password"     *

# UserIDs that cannot use PPP at all. Check your /etc/passwd and add any
# other accounts that should not be able to use pppd!
guest   hostname       "*"     -
master  hostname       "*"     -
root    hostname       "*"     -
support hostname       "*"     -
stats   hostname       "*"     -
```

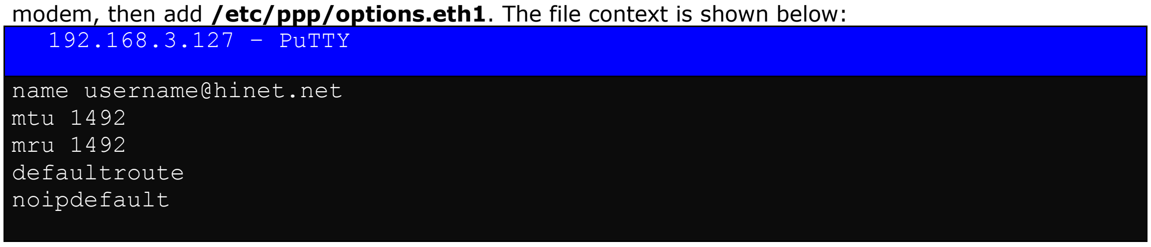
"username@hinet.net" is the username obtained from the ISP to log in to the ISP account. **"password"** is the corresponding password for the account.

5. Edit the file **/etc/ppp/options** and add the following line:
   `plugin pppoe`



```
# Wait for up n milliseconds after the connect script finishes for a valid
# PPP packet from the peer.  At the end of this time, or when a valid PPP
# packet is received from the peer, pppd will commence negotiation by
# sending its first LCP packet.  The default value is 1000 (1 second).
# This wait period only applies if the connect or pty option is used.
#connect-delay <n>

# Load the pppoe plugin
plugin /lib/rp-pppoe.so

# ---<End of File>---
```

6. Add one of two files: **/etc/ppp/options.eth0** or **/etc/ppp/options.eth1**. The choice depends on which LAN is connected to the ADSL modem. If you use LAN1 to connect to the ADSL modem, then add **/etc/ppp/options.eth0**. If you use LAN2 to connect to the ADSL modem, then add **/etc/ppp/options.eth1**. The file context is shown below:



```
name username@hinet.net
mtu 1492
mru 1492
defaultroute
noipdefault
```

   Type your username (the one you set in the **/etc/ppp/pap-secrets** and **/etc/ppp/chap-secrets** files) after the "name" option. You may add other options as desired.

7. Set up DNS.

   If you are using DNS servers supplied by your ISP, edit the file
   /etc/resolv.conf by adding the following lines of code:
   nameserver ip_addr_of_first_dns_server
   nameserver ip_addr_of_second_dns_server

   For example:
   nameserver 168.95.1.1
   nameserver 139.175.10.20

8. Use the following command to create a pppoe connection:
   pppd eth0

   The eth0 is what is connected to the ADSL modem LAN port. The example above uses LAN1. To use LAN2, type:
   pppd eth1

9. Type **ifconfig ppp0** to check if the connection is OK or has failed. If the connection is OK, you will see information about the ppp0 setting for the IP address. Use ping to test the IP.

10. If you want to disconnect it, use the kill command to kill the pppd process.

NFS (Network File System) Client

The Network File System (NFS) is used to mount a disk partition on a remote machine, as if it were on a local hard drive, allowing fast, seamless sharing of files across a network. NFS allows users to develop applications for the UC-8481, without worrying about the amount of disk space that will be available. The UC-8481 supports NFS protocol for client.

NOTE Click on the following links for more information about NFS:

http://www.tldp.org/HOWTO/NFS-HOWTO/index.html
http://nfs.sourceforge.net/nfs-howto/client.html
http://nfs.sourceforge.net/nfs-howto/server.html

Setting up the UC-8481 as an NFS Client

The following procedure is used to mount a remote NFS Server.

1. Establish a mount point on the NFS Client site.
2. Mount the remote directory to a local directory.

```
Steps 1:
#mkdir -p /home/nfs/public

Step 2:
#mount -t nfs NFS_Server(IP):/directory /mount/point

Example
: #mount -t nfs 192.168.3.100:/home/public /home/nfs/public
```

SNMP

The UC-8481 is comes with an SNMPv1 (Simple Network Management Protocol) agent already enabled with extensive support for managed objects (MIBs) on both TCP/IP-based and RS-232-like hardware devices.

The following simple example shows you how to use an SNMP browser (from a host site/SNMP manager) to query the UC-8481's SNMP agent:

```
debian:~# snmpwalk -v 1 -c public -Cc 192.168.30.127
```

The following is a typical response returned by a UC-8481:

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux version 2.6.23.1 (root@UC8400) (gcc version 4.2.1)
#1137 Wed Sep 17 16:17:45 EDT 2008
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.8691.12.8481
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (239600) 0:39:56.00
SNMPv2-MIB::sysContact.0 = STRING: Moxa Systems Co., LDT.
SNMPv2-MIB::sysName.0 = STRING: (none)
SNMPv2-MIB::sysLocation.0 = STRING: Unknown
SNMPv2-MIB::sysServices.0 = INTEGER: 6
IF-MIB::ifNumber.0 = INTEGER: 11
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
IF-MIB::ifIndex.4 = INTEGER: 4
IF-MIB::ifIndex.5 = INTEGER: 5
IF-MIB::ifIndex.6 = INTEGER: 6
IF-MIB::ifIndex.7 = INTEGER: 7
IF-MIB::ifIndex.8 = INTEGER: 8
IF-MIB::ifIndex.9 = INTEGER: 9
IF-MIB::ifIndex.10 = INTEGER: 10
IF-MIB::ifIndex.11 = INTEGER: 11
IF-MIB::ifDescr.1 = STRING: eth0
IF-MIB::ifDescr.2 = STRING: eth1
IF-MIB::ifDescr.3 = STRING: eth2***** SNMP QUERY FINISHED *****
```

NOTE Click on the following links for more information about how TCP/IP-based and RS-232-like devices are defined:

http://www.faqs.org/rfcs/rfc1213.html
http://www.faqs.org/rfcs/rfc1317.html

ATTENTION

The UC-8481 does **NOT** support SNMP trap.

OpenVPN

OpenVPN provides two types of tunnels for users to implement VPNS: **Routed IP Tunnels** and **Bridged Ethernet Tunnels**. To begin with, check to make sure that the system has a virtual device **/dev/net/tun**. If not, issue the following command:

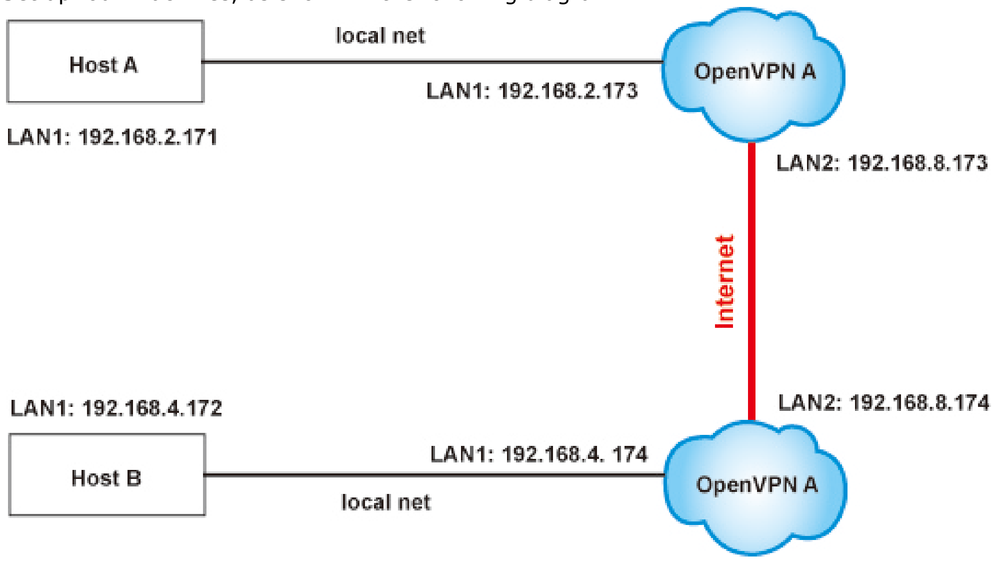
```
# mknod /dev/net/tun c 10 200
```

An Ethernet bridge is used to connect different Ethernet networks together. The Ethernets are bundled into one bigger, "logical" Ethernet. Each Ethernet corresponds to one physical interface (or port) that is connected to the bridge.

On each OpenVPN machine, you should generate a working directory, such as **/etc/openvpn**, where script files and key files reside. Once established, all operations will be performed in that directory.

Setup 1: Ethernet Bridging for Private Networks on Different Subnets

1. Set up four machines, as shown in the following diagram.



Host A (B) represents one of the machines that belongs to OpenVPN A (B). The two remote subnets are configured for a different range of IP addresses. When this setup is moved to a public network, the external interfaces of the OpenVPN machines should be configured for static IPs, or connect to another device (such as a firewall or DSL box) first.

```
# openvpn --genkey --secret secrouter.key
```

Copy the file that is generated to the OpenVPN machine.

2. The **openvpn-bridge** script file located at "/etc/openvpn/" reconfigures the interface "eth1" as IP-less, creates logical bridge(s) and TAP interfaces, loads modules, and enables IP forwarding.

```
#---------------------------------Start-----------------------------
#!/bin/sh
iface=eth1       # defines the internal interface
maxtap=`expr 1`  # defines the number of tap devices. I.e., # of tunnels
IPADDR=
NETMASK=
BROADCAST=
# it is not a great idea but this system doesn't support
# /etc/sysconfig/network-scripts/ifcfg-eth1
ifcfg_vpn()
{
 while read f1 f2 f3 f4 r3
 do
     if [ "$f1" = "iface" -a "$f2" = "$iface" -a "$f3" = "inet" -a "$f4"
= "static" ];then
          i=`expr 0`
          while :
          do
              if [ $i -gt 5 ]; then
                  break
              fi
              i=`expr $i + 1`
              read f1 f2
              case "$f1" in
                  address ) IPADDR=$f2
                      ;;
                  netmask ) NETMASK=$f2
                      ;;
                  broadcast ) BROADCAST=$f2
                      ;;
              esac
          done
              break
     fi
 done < /etc/network/interfaces
}
# get the ip address of the specified interface
mname=
module_up()
{
 oIFS=$IFS
 IFS='
 '
 FOUND="no"
 for LINE in `lsmod`
 do
     TOK=`echo $LINE | cut -d' ' -f1`
     if [ "$TOK" = "$mname" ]; then
         FOUND="yes";
         break;
     fi
 done
 IFS=$oIFS
 if [ "$FOUND" = "no" ]; then
     modprobe $mname
 fi
}
start()
{
 ifcfg_vpn
 if [ ! \( -d "/dev/net" \) ]; then
     mkdir /dev/net
 fi

 if [ ! \( -r "/dev/net/tun" \) ]; then
     # create a device file if there is none
     mknod /dev/net/tun c 10 200
 fi
 # load modules "tun" and "bridge"
 mname=tun
 module_up
 mname=bridge
 module_up
 # create an ethernet bridge to connect tap devices, internal interface
 brctl addbr br0
 brctl addif br0 $iface
 # the bridge receives data from any port and forwards it to other ports.

 i=`expr 0`
 while :
 do
     # generate a tap0 interface on tun
     openvpn --mktun --dev tap${i}
     # connect tap device to the bridge
     brctl addif br0 tap${i}

     # null ip address of tap device
     ifconfig tap${i} 0.0.0.0 promisc up

     i=`expr $i + 1`
     if [ $i -ge $maxtap ]; then
         break
     fi
 done

 # null ip address of internal interface
 ifconfig $iface 0.0.0.0 promisc up

 # enable bridge ip
 ifconfig br0 $IPADDR netmask $NETMASK broadcast $BROADCAST

 ipf=/proc/sys/net/ipv4/ip_forward
 # enable IP forwarding
 echo 1 > $ipf
 echo "ip forwarding enabled to"
 cat $ipf
}
stop() {
 echo "shutdown openvpn bridge."
 ifcfg_vpn
 i=`expr 0`
 while :
 do
     # disconnect tap device from the bridge
```

```
        brctl delif br0 tap${i}
        openvpn --rmtun --dev tap${i}

        i=`expr $i + 1`
        if [ $i -ge $maxtap ]; then
            break
        fi
    done
    brctl delif br0 $iface
    brctl delbr br0
    ifconfig br0 down
    ifconfig $iface $IPADDR netmask $NETMASK broadcast $BROADCAST
    killall -TERM openvpn
}
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    *)
        echo "Usage: $0 [start|stop|restart]"
        exit 1
esac
exit 0
#---------------------------------- end -----------------------------
```

3. On machine OpenVPN A, modify the remote address in the configuration file, **/etc/openvpn/tap0-br.conf**.

```
# /etc/openvpn/tap0-br.conf
# point to the peer
remote 192.168.8.174
dev tap0
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
up /etc/openvpn/tap0-br.sh
```

Next, modify the routing table in the **/etc/openvpn/tap0-br.sh** script file.

```
#----------------------------------Start-----------------------------
-
#!/bin/sh
# /etc/openvpn/tap0-br.sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.4.0 netmask 255.255.255.0 dev br0
#---------------------------------- end ------------------------------
```

On machine OpenVPN B, modify the remote address in the configuration file, **/etc/openvpn/tap0-br.conf.**

```
# /etc/openvpn/tap0-br.conf
# point to the peer
remote 192.168.8.173
dev tap0
```

```
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
up /etc/openvpn/tap0-br.sh
```

Next, modify the routing table in the **/etc/openvpn/tap0-br.sh** script file.

```
#---------------------------------- Start----------------------------
#!/bin/sh
# /etc/openvpn/tap0-br.sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.2.0 netmask 255.255.255.0 dev br0
#---------------------------------- end ----------------------------
```

Note: Select cipher and authentication algorithms by specifying "cipher" and "auth". To see with algorithms are available, type:

```
# openvpn --show-ciphers
# openvpn --show—auths
```

4. After configuring the remote peer, we can load the bridge into kernel, reconfigure eth1, and enable IP forwarding on both OpenVPN machine.

   ```
   # /etc/openvpn/openvpn-bridge start
   ```

   Next, start both OpenVPN peers,

   ```
   # openvpn --config /etc/openvpn/tap0-br.conf &
   ```

   If you see the line "Peer Connection Initiated with 192.168.8.173:5000" on each machine, the connection between OpenVPN machines has been established successfully on UDP port 5000.

   Note: You can create link symbols to enable the **/etc/openvpn/openvpn-bridge** script at boot time:

   ```
   # ln -s /etc/openvpn/openvpn-bridge /etc/rc.d/rc3.d/S32vpn-br
   # ln -s /etc/openvpn/openvpn-bridge /etc/rc.d/rc6.d/K32vpn-br
   ```

5. On each OpenVPN machine, check the routing table by typing the command:

   ```
   # route
   ```

Destination	Gateway	Genmsk	Flags	Metric	Ref	Use	Iface
192.168.4.0	*	255.255.255.0	U	0	0	0	br0
192.168.2.0	*	255.255.255.0	U	0	0	0	br0
192.168.8.0	*	255.255.255.0	U	0	0	0	eth0

   Interface **eth1** is connected to the bridging interface **br0**, to which device **tap0** also connects, whereas the virtual device **tun** sits on top of **tap0**. This ensures that all traffic from internal networks connected to interface **eth1** that come to this bridge write to the TAP/TUN device that the OpenVPN program monitors. Once the OpenVPN program detects traffic on the virtual device, it sends the traffic to its peer.

6. To create an indirect connection to Host B from Host A, add the following routing item:

   ```
   route add –net 192.168.4.0 netmask 255.255.255.0 dev eth0
   ```

   To create an indirect connection to Host A from Host B, add the following routing item:

   ```
   route add –net 192.168.2.0 netmask 255.255.255.0 dev eth0
   ```

   Now ping Host B from Host A by typing:

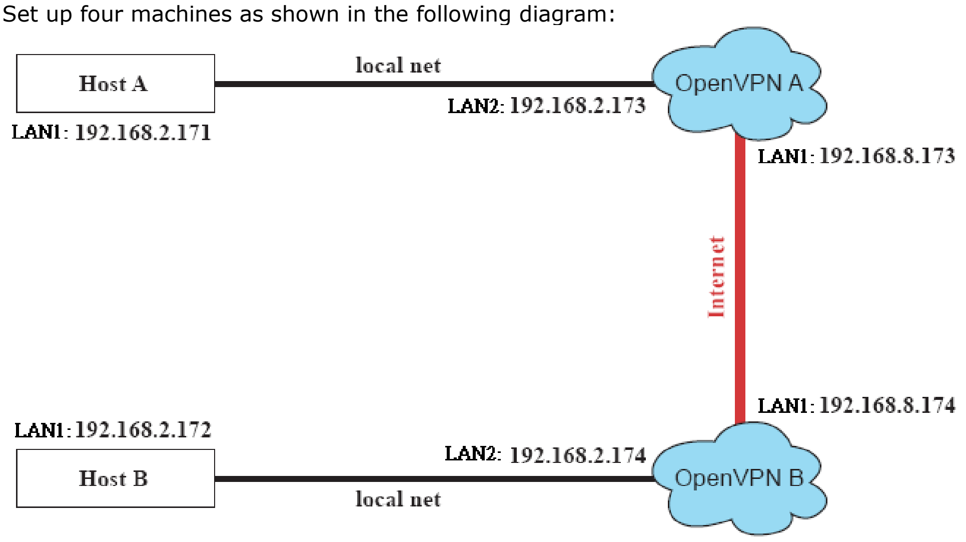
   ```
   ping 192.168.4.174
   ```

   A successful ping indicates that you have created a VPN system that only allows authorized users from one internal network to access users at the remote site. For this system, all data is transmitted by UDP packets on port 5000 between OpenVPN peers.

7. To shut down OpenVPN programs, type the command:
```
# /etc/openvpn/openvpn-bridge stop
```

Setup 2: Ethernet Bridging for Private Networks on the Same Subnet

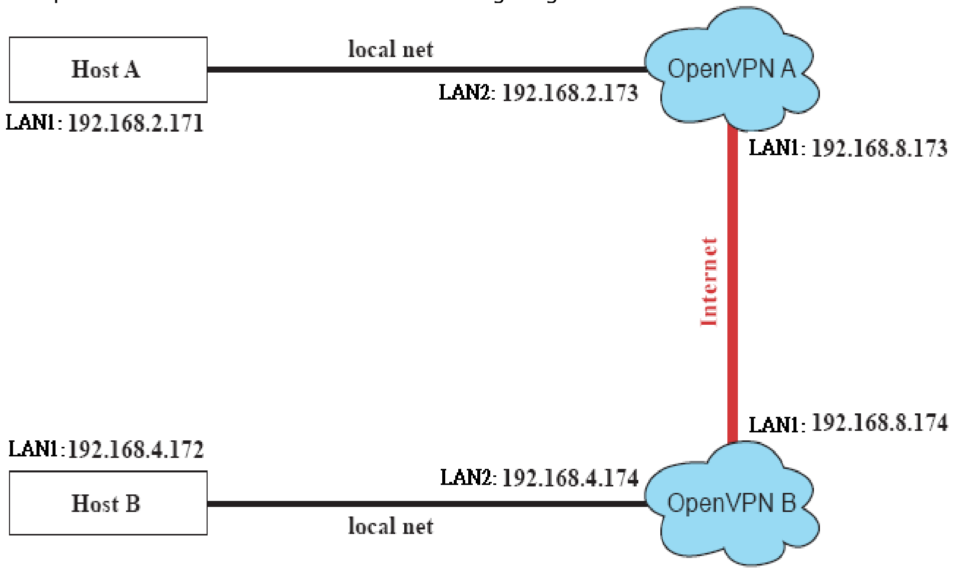
1. Set up four machines as shown in the following diagram:



2. The configuration procedure is almost the same as for the previous example. The only difference is that you will need to comment out the parameter "up" in "/etc/openvpn/tap0-br.conf" and "/etc/openvpn/tap0-br.conf".

Setup 3: Routed IP

1. Set up four machines as shown in the following diagram:



2. On machine OpenVPN A, modify the remote address in the configuration file, **/etc/openvpn/tun.conf.**

```
# point to the peer
remote 192.168.8.174
dev tun
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
ifconfig 192.168.2.173 192.168.4.174
up /etc/openvpn/tun.sh
```

Next, modify the routing table in the **/etc/openvpn/tun.sh** script file.

```
#--------------------------------- Start-----------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.4.0 netmask 255.255.255.0 gw $5
#--------------------------------- end ------------------------------
```

On machine OpenVPN B, modify the remote address in the configuration file, **/etc/openvpn/tun.conf.**

```
remote 192.168.8.173
dev tun
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
ifconfig 192.168.4.174 192.168.2.173
up /etc/openvpn/tun.sh
```

Next, modify the routing table in the **/etc/openvpn/tun.sh** script file.

```
#--------------------------------- Start----------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.2.0 netmask 255.255.255.0 gw $5
#--------------------------------- end -----------------------------
```

Note that the parameter "ifconfig" defines the first argument as the local internal interface and the second argument as the internal interface at the remote peer.

Note that **$5** is the argument that the OpenVPN program passes to the script file. Its value is the second argument of **ifconfig** in the configuration file.

3. Check the routing table after you run the OpenVPN programs, by typing the command:
`# route`

Destination	Gateway	Genmsk	Flags	Metric	Ref	Use	Iface
192.168.4.174	*	255.255.255.255	UH	0	0	0	tun0
192.168.4.0	192.168.4.174	255.255.255.0	UG	0	0	0	tun0
192.168.2.0	*	255.255.255.0	U	0	0	0	eth1
192.168.8.0	*	255.255.255.0	U	0	0	0	eth0

Package Management—ipkg

ipkg is a very lightweight package management system. It also allows for dynamic installation/removal of packages on a running system. Because memory space is limited, we provide software as extension packages. On the UC-8481-LX you can use ipkg-cl to install or remove packages.

Install an .ipk package via an .ipk file

Upload the .ipk package to the Moxa embedded computer:

```
192.168.3.127 – Putty
Moxa:~# scp /mnt/cdrom/utility_tools/UC-8481-LX/libphp5_5.0.5_xscale.ipk
192.168.3.120:/tmp/
```

Install the uploaded package:

```
192.168.3.127 – Putty
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl install /tmp/libphp5_5.0.5_xscale.ipk
Moxa:~# umount /
```

List the installed packages

```
192.168.3.127 – Putty
Moxa:~# ipkg-cl list_installed
```

Remove a package

```
192.168.3.127 – Putty
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl remove libphp5
Moxa:~# umount /
```

Web Server: Apache 2.2.8, PHP 5.05

The web server is Apache 2.2.8, and is an optional package. The package name is **apache_2.2.8_xscale.ipk**, and may be uploaded from your CD to the UC-8481. Just as above, the command "ipkg-cl" will install it.

First, copy the package into the operating system:

```
192.168.3.120 – Putty
Moxa:~# scp /mnt/cdrom/utility_tools/UC-8481-LX/apache_2.2.8_xscale.ipk
192.168.3.120:/dev/shm/
```

Then remount the operating system read-write and install the package with the command "ipkg-cl install /path/to/software/package.ipk":

```
192.168.3.120 - Putty
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl install /dev/shm/apache_2.2.8_xscale.ipk
Moxa:~# umount /
```

To remove apache, simply use the command "ipkg-cl remove apache":

```
192.168.3.120 - Putty
```

```
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl remove apache
Moxa:~# umount /
```

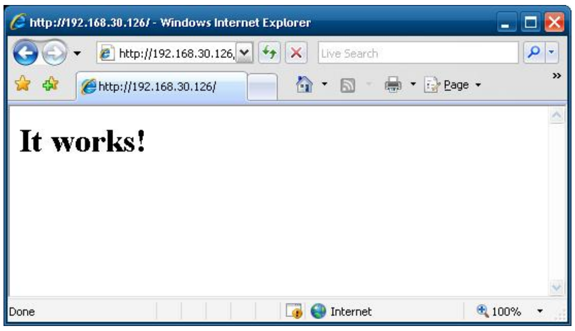
The Apache web server's main configuration file is **/etc/apache/httpd.conf**, with the default homepage located at **/home/httpd/index.html**. Save your own homepage to the following directory:

/home/httpd/htdocs/

Save your CGI page to the following directory:

/home/httpd/cgi-bin/

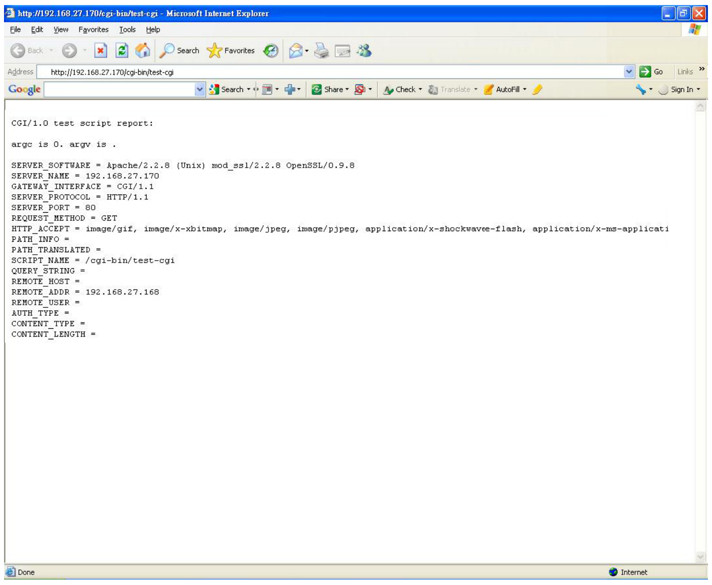
Before you modify the homepage, use a browser (such as Microsoft Internet Explore or Mozilla Firefox) from your PC to test if the Apache Web Server is working. Type the LAN1 IP address in the browser's address bar to open the homepage, e.g., if the default IP address is still active, type **http://host-ip-address** in address bar.



It works!

To open the default CGI page, type **http://host-ip-address/cgi-bin/printenv** in your browser's address bar.

To open the default CGI test script report page, type **http://host-ip-address/cgi-bin/test-cgi** in your browser's address bar.



http://192.168.27.170/cgi-bin/test-cgi - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://192.168.27.170/cgi-bin/test-cgi Go Links
Google Search Share Check Translate AutoFill Sign In
Done
Internet

```
CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = Apache/2.2.8 (Unix) mod_ssl/2.2.8 OpenSSL/0.9.8
SERVER_NAME = 192.168.27.170
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.1
SERVER_PORT = 80
REQUEST_METHOD = GET
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/x-ms-applicati
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING =
REMOTE_HOST =
REMOTE_ADDR = 192.168.27.168
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =
```

NOTE The CGI function is enabled by default. If you want to disable the function, modify the file **/etc/apache/httpd.conf**. When you develop your own CGI application, make sure your CGI file is executable.

192.168.4.127 - PuTTY

```
root@Moxa:/usr/www/cgi-bin# ls  – al
 drwxr—xr-x     2 root      root              0 Aug 24 1999
 drwxr—xr-x     5 root      root              0 Nov   5 16:16
 -rwxr—xr-x     1 root      root            268 Dec 19 2002 printenv
 -rwxr—xr-x     1 root      root            757 Aug 24 1999 test-cgi
 root@Moxa:/usr/www/cgi-bin#
```

The Apache web server also supports PHP; you may upload the PHP package from the CD and install it using the same commands as above.

Copy the package from the CD to the file system using scp:

192.168.3.120 – Putty

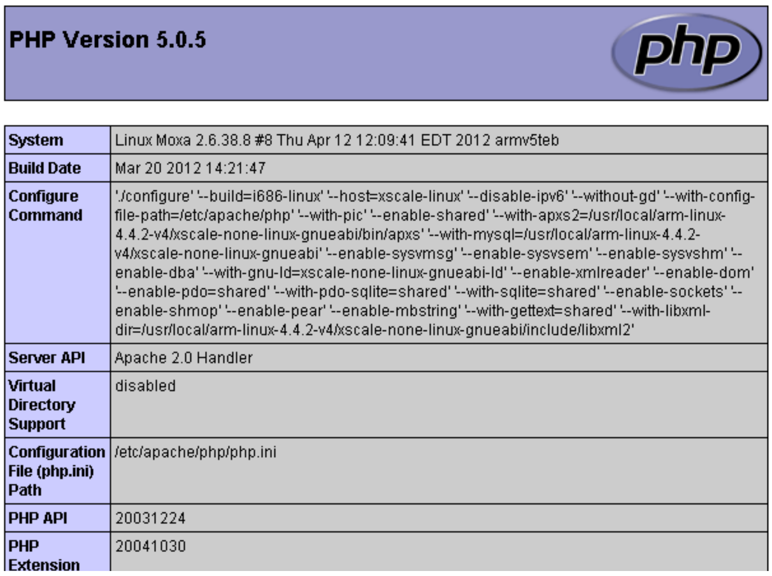
```
Moxa:~# scp /mnt/cdrom/utility_tools/UC-8481-LX/libmysqlclient5_5.1.23_xscale.ipk
192.168.3.120:/dev/shm/
Moxa:~# scp /mnt/cdrom/utility_tools/UC-8481-LX/libphp5_5.0.5_xscale.ipk
192.168.3.120:/dev/shm/
```

Then install the uloaded package with the command
"ipkg-cl install /path/to/software/package.ipk":

192.168.3.120 – Putty

```
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl install /dev/shm/libmysqlclient5_5.1.23_xscale.ipk
Moxa:~# ipkg-cl install /dev/shm/libphp5_5.0.5_xscale.ipk
Moxa:~# umount /
```

To open the default phpinfo.php page, type **http://host-ip-address/phpinfo.php** into your browser's address bar. You should see the following launch page:



PHP Version 5.0.5

System	Linux Moxa 2.6.38.8 #8 Thu Apr 12 12:09:41 EDT 2012 armv5teb
Build Date	Mar 20 2012 14:21:47
Configure Command	'./configure' '--build=i686-linux' '--host=xscale-linux' '--disable-ipv6' '--without-gd' '--with-config-file-path=/etc/apache/php' '--with-pic' '--enable-shared' '--with-apxs2=/usr/local/arm-linux-4.4.2-v4/xscale-none-linux-gnueabi/bin/apxs' '--with-mysql=/usr/local/arm-linux-4.4.2-v4/xscale-none-linux-gnueabi' '--enable-sysvmsg' '--enable-sysvsem' '--enable-sysvshm' '--enable-dba' '--with-gnu-ld=xscale-none-linux-gnueabi-ld' '--enable-xmlreader' '--enable-dom' '--enable-pdo=shared' '--with-pdo-sqlite=shared' '--with-sqlite=shared' '--enable-sockets' '--enable-shmop' '--enable-pear' '--enable-mbstring' '--with-gettext=shared' '--with-libxml-dir=/usr/local/arm-linux-4.4.2-v4/xscale-none-linux-gnueabi/include/libxml2'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/apache/php/php.ini
PHP API	20031224
PHP Extension	20041030

To remove PHP and the libmysqlclient5 package, use the following commands:

```
192.168.3.120 – Putty
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl remove libmysqlclient5
Moxa:~# ipkg-cl remove libphp5
Moxa:~# umount /
```

minicom

Minicom is a communication program which somewhat resembles the shareware program TELIX. You can follow these steps to install it:

Copy the package from the CD to the file system using scp:

```
192.168.3.120 – Putty
Moxa:~# scp /mnt/cdrom/utility_tools/UC-8481-LX/minicom_2.4_xscale.ipk
192.168.3.120:/dev/shm/
```

Then install the uloaded package with the command
"ipkg-cl install path/to/software/package.ipk":

```
192.168.3.120 – Putty
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl install /dev/shm/minicom_2.4_xscale.ipk
Moxa:~# umount /
```

To remove the package, use the following command.

```
192.168.3.120 – Putty
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl remove minicom
Moxa:~# umount /
```

To start minicom, use the following command.

```
192.168.3.120 – Putty
Moxa:~# minicom –s
```

sudo

Copy the package from the CD to the file system using scp:

```
192.168.3.120 – Putty
Moxa:~# scp /mnt/cdrom/utility_tools/ipkg_packages/sudo_1.8.1p2_xscale.ipk
192.168.3.120:/dev/shm/
```

Then install the uloaded package with the command
"ipkg-cl install /path/to/software/package.ipk":

```
192.168.3.120 – Putty
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl install /dev/shm/sudo_1.8.1p2_xscale.ipk
Moxa:~# umount /
```

Add a new user; here, we use "**moxa**":

```
192.168.3.120 – Putty
moxa@Moxa:~$ adduser moxa
```

Add **moxa** in **/etc/sudoers**

```
192.168.3.120 – Putty
moxa@Moxa:~$ chmod a+w /etc/sudoers
moxa@Moxa:~$ vi /etc/sudoers

# User privilege specification
moxa      ALL=(ALL) ALL

moxa@Moxa:~$ chmod a-w /etc/sudoers
```

Login to **sudo** with the user acccont **moxa** :

```
192.168.3.120 – Putty
moxa@Moxa:~$ sudo -i
Password:
```

Sudo may now be used (without a password) to execute root-level commands without logging into root:

```
192.168.3.120 – Putty
moxa@Moxa:~$ sudo reboot
```

Perl

Copy the package from the CD to the file system using scp:

```
192.168.3.120 – Putty
Moxa:~# scp /mnt/cdrom/utility_tools/UC-8481-LX/ perl_5.10.1_xscale.ipk
192.168.3.120:/dev/shm/
```

Install the **perl** package with the command

"ipkg-cl install /path/to/software/package.ipk":

```
192.168.3.120 – Putty
Moxa:~# mount -o remount,rw /
Moxa:~# ipkg-cl install /dev/shm/perl_5.10.1_xscale.ipk
Moxa:~# umount /
```

All files will be installed at /home/perl/. After the installation completes, you need to set PERL5LIB for locating the perl libraries.

```
192.168.3.120 – Putty
Moxa:~# export export LD_PRELOAD=/lib/libgcc_s.so.1
Moxa:~# export
PERL5LIB=/home/perl/lib/perl5/5.10.1/:/home/perl/lib/perl5/5.10.1/armv5teb-linuxMoxa:~#
export export LD_PRELOAD=/lib/libgcc_s.so.1
Moxa:~# export LD_LIBRARY_PATH=/home/perl/lib/perl5/5.10.1/armv5teb-linux/CORE
```

NOTE The package installation will add the PERL5LIB and LD_LIBRARY_PATH into /etc/profile automatically. If you reboot the system, you don't need to export these variables manually again.

When the installation is complete you may begin using perl immediately:

```
192.168.3.120 – Putty
root@Moxa:/# perl -v

This is perl, v5.10.1 (*) built for xscale-linux

Copyright 1987-2009, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.
```

```
Complete documentation for Perl, including FAQ lists, should be found on
this system using "man perl" or "perldoc perl".   If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.

root@Moxa:/#
```

5
Programmer's Guide

This chapter includes important information for programmers.

The following topics are covered in this chapter:

Flash Memory Map

Partition sizes are hard coded into the kernel binary. To change partition sizes, you will need to rebuild the kernel. The flash memory map is shown in the following table.

Address	Size	Contents
0x00000000 – 0x000FFFFF	1 MB	Boot Loader — Read ONLY
0x00100000 – 0x002DFFFF	1.875 MB	Kernel object code — Read ONLY
0x002E0000 – 0x01FDFFFF	29.375 MB	Root file system (JFFS2) — Read ONLY
0x01FE0000 – 0x01FFFFFF	128 KB	Boot Loader configuration and directory — Read ONLY
	512 MB	User root file system (YAFFS2) — Read/Write

NOTE

1. The default Moxa file system only enables the network. It lets users recover the user file system when it fails.
2. The user file system is a complete file system. Users can create and delete directories and files (including source code and executable files) as needed.
3. Users can create the user file system on the PC host or target platform, and then copy it to the UC-8481.

Linux Tool Chain Introduction

To ensure that an application will be able to run correctly when installed on the UC-8481, you must ensure that it is compiled and linked to the same libraries that will be present on the UC-8481. This is particularly true when the RISC XScale processor architecture of the UC-8481 differs from the CISC x86 processor architecture of the host system, but it is also true if the processor architecture is the same.

The host tool chain that comes with the UC-8481 contains a suite of cross compilers and other tools, as well as the libraries and headers that are necessary to compile applications for the UC-8481. The host environment must be running Linux to install the UC-8481 GNU Tool Chain. We have confirmed that the following Linux distributions can be used to install the tool chain:

- Redhat 7.3/8.0/9.0
- Fedora core 1/2/3/4/5
- Debian 4.0 32 bit platform.

To compile the tool chain you will need a PC with at least 600 MB of disk space. The UC-8481 tool chain is located on the UC-8481 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount /dev/cdrom /mnt/cdrom
#sh /mnt/cdrom/tool-chain/linux/arm-linux_4.4.2-v4_Build_12032109.sh
```

Wait for a few minutes while the Tool Chain is installed automatically on your Linux PC. Once the host environment has been installed, add the directory /usr/local/arm-linux-4.4.2-v4/bin to your path and the directory /usr/local/arm-linux-4.4.2-v4/man to your manual path. You can do this temporarily for the current login session by issuing the following commands:

```
#export PATH="/usr/local/arm-linux-4.4.2-v4/bin:$PATH"
#export MANPATH="/usr/local/arm-linux-4.4.2-v4/man:$MANPATH"
```

Alternatively, you can add the same commands to $HOME/.bash_profile to cause it to take effect for all login sessions initiated by this user.

Obtaining Help

For help with console commands, Linux includes a system manual that can be called with the **man** utility. Simply type "**man <command>**" to obtain a full help listing with any system command. For example, to get help on the **arm-linux-gcc** compiler, issue the command:

```
#man arm-linux-gcc
```

Cross Compiling Applications and Libraries

To compile a simple C application, just use the cross compiler instead of the regular compiler:

```
#xscale-linux-gcc  –o example –Wall –g –O2 example.c
#xscale-linux-strip  –s example
#xscale-linux-gcc  -ggdb –o example-debug example.c
```

Tools Available in the Host Environment

Most of the cross compiler tools are the same as their native compiler counterparts, but with an additional prefix that specifies the target system. In the case of x86 environments, the prefix is **i386-linux-** and in the case of the UC-8481 XScale boards, it is **xscale-linux-**.

For example, the native C compiler is **gcc** and the cross C compiler for XScale in the UC-8481 is **xscale-linux-gcc**.

The following cross compiler tools are provided:

ar	Manages archives (static libraries)
as	Assembler
c++, g++	C++ compiler
cpp	C preprocessor
gcc	C compiler
gdb	Debugger
ld	Linker
nm	Lists symbols from object files
objcopy	Copies and translates object files
objdump	Displays information about object files
ranlib	Generates indexes to archives (static libraries)
readelf	Displays information about ELF files
size	Lists object file section sizes
strings	Prints strings of printable characters from files (usually object files)
strip	Removes symbols and sections from object files (usually debugging information)

Debugging with GDB

First use the option -ggdb to compile the program. Use the following steps:

1. To debug a program called **hello-debug** on the target, use the command:
   ```
   #gdbserver 192.168.4.142:2000 hello-debug
   ```
   This is where 2000 is the network port number on which the server waits for a connection from the client. This can be any available port number on the target. Following this are the name of the program to be debugged (hello-debug), plus that program’s arguments. Output similar to the following will be sent to the console:
   ```
   Process hello-debug created; pid=38
   ```

2. Use the following command on the host to change to the directory that contains hello-debug:
   ```
   cd /my_work_directory/myfilesystem/testprograms
   ```

3. Enter the following command:
   ```
   #ddd --debugger xscale-linux-gdb hello-debug &
   ```

4. Enter the following command at the GDB, DDD command prompt:
   ```
   Target remote 192.168.4.99:2000
   ```

The command produces another line of output on the target console, similar to the following:

```
Remote debugging using 192.168.4.99:2000
```

192.168.4.99 is the machine's IP address, and 2000 is the port number. You can now begin debugging in the host environment using the interface provided by DDD.

5. Set a breakpoint on main by double clicking, or entering **b main** on the command line.
6. Click the **cont** button

Device API

The UC-8481 supports control devices with the **ioctl** system API. You will need to include **<moxadevice.h>**, and use the following **ioctl** function.

```
int ioctl(int d, int request,…);
  Input:  int d       - open device node return file handle
          int request – argument in or out
```

Use the desktop Linux's man page for detailed documentation:

```
#man ioctl
```

RTC (Real Time Clock)

The device node is located at **/dev/rtc**. The UC-8481 supports Linux standard simple RTC control. You must include **<linux/rtc.h>**.

1. Function: RTC_RD_TIME

   ```
   int ioctl(fd, RTC_RD_TIME, struct rtc_time *time);
   ```
   Description: read time information from the RTC. It will return the value on argument 3.

2. Function: RTC_SET_TIME

   ```
   int ioctl(fd, RTC_SET_TIME, struct rtc_time *time);
   ```
   Description: set RTC time. Argument 3 will be passed to the RTC.

Buzzer

The device node is located at **/dev/console**. The UC-8481 supports Linux standard buzzer control, with the UC-8481's buzzer running at a fixed frequency of 100 Hz. You must include **<sys/kd.h>**.

1. Function: KDMKTONE

   ```
   ioctl(fd, KDMKTONE, unsigned int arg);
   ```
   Description: The buzzer's behavior is determined by the argument arg. The "high word" part of arg gives the length of time the buzzer will sound, and the "low word" part gives the frequency.
   The buzzer's on/off behavior is controlled by software. If you call the "ioctl" function, you MUST set the frequency to 100 Hz. If you use a different frequency, the system could crash.

WDT (Watch Dog Timer)

1. **Introduction**
   The WDT works like a watch dog function. You can enable it or disable it. When the user enables WDT but the application does not acknowledge it, the system will reboot. You can set the ack time from a minimum of 50 msec to a maximum of 60 seconds.
2. **How the WDT works**
   The sWatchDog is enabled when the system boots up. The kernel will auto ack it. The user application can also enable ack. When the user does not ack, it will let the system reboot.

   Kernel boot
   …..
   ….

   User application running and enable user ack
   ….
   ….
3. **The user API**
   The user application must use **include <moxadevic.h>**, and **link libmoxalib.a**. A makefile example is shown below:

```
all:
     xscale-linux-gcc –o xxxx  xxxx.c -lmoxalib
```

int swtd_open(void)

Description
If you would like to activate Watchdog for the AP, you must call this function.

Input
None

Output
The return value is file handle. If there is an error, it will return a negative value.

You can get the error using the function errno().

int swtd_enable(int fd, unsigned long time)

Description
Enable the application sWatchDog. You must do an ack after this process.

Input
int fd - the file handle, from the swtd_open() return value.

unsigned long time - The time you wish to ack sWatchDog periodically. You must ack the sWatchDog before timeout. If you do not ack, the system will reboot automatically. The minimum time is 50 msec, and the maximum time is 60 seconds. The time unit is msec.

Output
If you receive 0 (zero), it means the function is working. If you receive any other number, then there is something wrong with this function.

int swtd_disable(int fd)

Description:
Call this function if you would like the AP to stop using the Watchdog.

Input :
int fd - the file handle from swtd_open() return value.

Output:
If you receive 0 (zero), it means the function is working. If you receive any other number, then there is something wrong with this function.

int swtd_get(int fd, int *mode, unsigned long *time)

Description:
Get current setting values.

mode –
1 for user application enable sWatchDog: need to do ack.
0 for user application disable sWatchdog: does not need to do ack.
time – The time period to ack sWatchDog.

Input:

int fd	- the file handle from swtd_open() return value.
int *mode	- the function will return the status: enable or disable.
unsigned long *time	– the function will return the current time period.

Output:
If you receive 0 (zero), it means the function is working. If you receive any other number, then there is something wrong with this function.

int swtd_ack(int fd)

Description:
Acknowledge sWatchDog. When the user application enable sWatchDog, it need to call this function periodically with user predefined time in the application program.

Input:
int fd - the file handle from swtd_open() return value.

Output:
If you receive 0 (zero), it means the function is working. If you receive any other number, then there is something wrong with this function.

int swtd_close(int fd)

Description:
Close the file handle.

Input:
int fd - the file handle from swtd_open() return value.

Output:
If you receive 0 (zero), it means the function is working. If you receive any other number, then there is something wrong with this function.

4. Special Note

When you "kill the application with -9" or "kill without option" or "Ctrl+c" the kernel will change to auto ack the sWatchDog.

When your application enables the sWatchDog and does not ack, your application may have a logical error, or your application has made a core dump. The kernel will not change to auto ack. This can cause a serious problem, causing your system to reboot again and again.

5. User application example

Example 1:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <moxadevice.h>
```

```
int main(int argc, char *argv[])
{
      int fd;
      fd = swtd_open();
      if ( fd < 0 ) {
          printf("Open sWatchDog device fail !\n");
          exit(1);
      }
      swtd_enable(fd, 5000);  // enable it and set it 5 seconds
      while ( 1 ) {
          // do user application want to do
          …..
          ….
          swtd_ack(fd);
          …..
          ….
      }
      swtd_close(fd);
      exit(0);
}
```

The makefile is shown below:

```
all:
      xscale-linux-gcc –o xxxx xxxx.c –lmoxalib
```

Example 2:

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <moxadevice.h>
static void mydelay(unsigned long msec)
{
      struct timeval  time;
      time.tv_sec = msec / 1000;
      time.tv_usec = (msec % 1000) * 1000;
      select(1, NULL, NULL, NULL, &time);
}
static int   swtdfd;
static int   stopflag=0;
static void stop_swatchdog()
{
      stopflag = 1;
}
static void do_swatchdog(void)
{
      swtd_enable(swtdfd, 500);
      while ( stopflag == 0 ) {
          mydelay(250);
          swtd_ack(swtdfd);
      }
      swtd_disable(swtdfd);
  }
```

```
int  main(int argc, char *argv[])
{
     pid_t        sonpid;
     signal(SIGUSR1, stop_swatchdog);
     swtdfd = swtd_open();
     if ( swtdfd < 0 ) {
         printf("Open sWatchDog device fail !\n");
         exit(1);
     }
     if ( (sonpid=fork()) == 0 )
         do_swatchdog();
     // do user application main function
     …..
     …..
     …..
     // end user application
     kill(sonpid, SIGUSR1);
     swtd_close(swtdfd);
     exit(1);
}
```

The makefile is shown below:

```
all:
     xscale-linux-gcc –o xxxx xxxx.c –lmoxalib
```

Digital I/O

Digital Output channels can be set to high or low. The channels are controlled by the function call **set_dout_state( )**. The Digital Input channels can be used to detect the state change of the digital input signal. The DI channels can also be used to detect whether or not the state of a digital signal changes during a fixed period of time. This can be done with the function call **set_din_event( )**. Moxa provides 5 function calls to handle digital I/O state changes and event handling.

Application Programming Interface

Return error code definitions:

```
#define DIO_ERROR_PORT -1 // no such port
#define DIO_ERROR_MODE -2 // no such mode or state
#define DIO_ERROR_CONTROL -3 // open or ioctl fail
#define DIO_ERROR_DURATION -4 // The value of duration is not 0 or not in the range,
40 <= duration <= 3600000 milliseconds (1 hour)
#define DIO_ERROR_DURATION_20MS -5 // The value of duration must be a multiple of 20
ms
#define DIO_OK 0
```

The definition of DIN and DOUT:

```
#define DIO_HIGH 1
#define DIO_LOW 0
int set_dout_state(int doport, int state)
```

Description: To set the DOUT port to high or low state.

Input: int doport - which DOUT port you want to set. Port starts from 0 to 3.
int state - to set high or low state; DIO_HIGH (1) for high, DIO_LOW (0) for low.

Output: none.
Return: reference the error code.

`int get_din_state(int diport, int *state)`

Description: To get the DIN port state.

Input: int diport - get the current state of which DIN port. Port numbering is from 0 to 3.
int *state - save the current state.
Output: state - DIO_HIGH (1) for high, DIO_LOW (0) for low.
Return: reference the error code.

`int get_dout_state(int doport, int *state)`

Description: To get the DOUT port state.
Input: int doport - get the current state of which DOUT port.
int *state - save the current state.
Output: state - DIO_HIGH (1) for high, DIO_LOW (0) for low.
Return: reference the error code.

`int set_din_event(int diport, void (*func)(int diport), int mode, long int duration)`

Description: Set the event for DIN when the state is changed from high to low or from low to high.
Input: int diport - the port that will be used to detect the DIN event.
Port numbering is from 0 to 3.
void (*func) (int diport) - Not NULL
> Returns the call back function. When the event occurs, the call back function will be invoked.
`NULL` > Clears this event

`int mode DIN_EVENT_HIGH_TO_LOW`

(1): from high to low
`DIN_EVENT_LOW_TO_HIGH`
(0): from low to high
`DIN_EVENT_CLEAR`
(-1): clear this event
unsigned long duration - 0: detect the din event > DIN_EVENT_HIGH_TO_LOW or
`DIN_EVENT_LOW_TO_HIGH` > without duration
- Not 0
> detect the din event
`DIN_EVENT_HIGH_TO_LOW` or
`DIN_EVENT_LOW_TO_HIGH` with duration. The value of "duration" must be a multiple of 20 milliseconds. The range of "duration" is 0, or 40 <= duration <= 3600000 milliseconds. The error of the measurement is 24 ms. For example, if the DIN duration is 200 ms, this event will be generated when the DIN pin stays in the same state for a time between 176 ms and 200 ms. Output: none.
Return: reference the error code.

`int get_din_event(int diport, int *mode, long int *duration)`

Description: To retrieve the DIN event configuration, including mode
`(DIN_EVENT_HIGH_TO_LOW or DIN_EVENT_LOW_TO_HIGH)`, and the value of "duration."
Input: **`int diport`** - which DIN port you want to retrieve.
- The port whose din event setting we wish to retrieve
`int *mode` - save which event is set.
`unsigned long *duration` - the duration of the DIN port is kept in high or low state.
- return to the current duration value of diport
Output: **`mode`** DIN_EVENT_HIGH_TO_LOW
(1): from high to low
DIN_EVENT_LOW_TO_HIGH(0): from low to high
DIN_EVENT_CLEAR(-1): clear this event
`duration` The value of duration should be 0 or 40 <= duration

<= 3600000 milliseconds.

Return: reference the error code.

Special Note

Do not forget to link to the library **libmoxalib.a** for DI/DO programming, and also include the header file **moxadevice.h**. The DI/DO library only can be used by one program at a time.

Examples

Example 1

File Name: tdio.c

Description: The program indicates to connect DO1 to DI1, change the digital output state to high or low by manual input, and then detect and count the state changed events from DI1.

```
#include <stdio.h>
#include <stdlib.h>
#include <moxadevice.h>
#include <fcntl.h>
#ifdef DEBUG
#define dbg_printf(x...) printf(x)
#else
#define dbg_printf(x...)
#endif
#define MIN_DURATION 40
static char *DataString[2]={"Low ", "High "};
static void hightolowevent(int diport)
{
printf("\nDIN port %d high to low.\n", diport);
}
static void lowtohighevent(int diport)
{
printf("\nDIN port %d low to high.\n", diport);
}
int main(int argc, char * argv[])
{
int i, j, state, retval;
unsigned long duration;
while( 1 ) {
printf("\nSelect a number of menu, other key to exit. \n\
1. set high to low event \n\
2. get now data. \n\
3. set low to high event \n\
4. clear event \n\
5. set high data. \n\
6. set low data. \n\
7. quit \n\
8. show event and duration \n\
Choose : ");
retval =0;
scanf("%d", &i);
if ( i == 1 ) { // set high to low event
```

```
printf("Please keyin the DIN number : ");
scanf("%d", &i);
printf("Please input the DIN duration, this minimun value must be over %d : ",
MIN_DURATION);
scanf("%lu", &duration);
retval=set_din_event(i, hightolowevent, DIN_EVENT_HIGH_TO_LOW, duration);
} else if ( i == 2 ) { // get now data
printf("DIN data : ");
for ( j=0; j<4; j++ ) {
get_din_state(j, &state);
printf("%s", DataString[state]);
}
printf("\n");
printf("DOUT data : ");
for ( j=0; j<MAX_DOUT_PORT; j++ ) {
get_dout_state(j, &state);
printf("%s", DataString[state]);
}
printf("\n");
} else if ( i == 3 ) { // set low to high event
printf("Please keyin the DIN number : ");
scanf("%d", &i);
printf("Please input the DIN duration, this minimun value must be over %d :",
MIN_DURATION);
scanf("%lu", &duration);
retval = set_din_event(i, lowtohighevent, DIN_EVENT_LOW_TO_HIGH, duration);
} else if ( i == 4 ) { // clear event
printf("Please keyin the DIN number : ");
scanf("%d", &i);
retval=set_din_event(i, NULL, DIN_EVENT_CLEAR, 0);
} else if ( i == 5 ) { // set high data
printf("Please keyin the DOUT number : ");
scanf("%d", &i);
retval=set_dout_state(i, 1);
} else if ( i == 6 ) { // set low data
printf("Please keyin the DOUT number : ");
scanf("%d", &i);
retval=set_dout_state(i, 0);
} else if ( i == 7 ) { // quit
break;
} else if ( i == 8 ) { // show event and duration
printf("Event:\n");
for ( j=0; j<MAX_DOUT_PORT; j++ ) {
retval=get_din_event(j, &i, &duration);
switch ( i ) {
case DIN_EVENT_HIGH_TO_LOW :
printf("(htl,%lu)", duration);
break;
case DIN_EVENT_LOW_TO_HIGH :
printf("(lth,%lu)", duration);
break;
case DIN_EVENT_CLEAR :
printf("(clr,%lu)", duration);
break;
default :
```

```
printf("err " );
break;
}
}
printf("\n");
} else {
printf("Select error, please select again !\n");
}
switch(retval) {
case DIO_ERROR_PORT:
printf("DIO error port\n");
break;
case DIO_ERROR_MODE:
printf("DIO error mode\n");
break;
case DIO_ERROR_CONTROL:
printf("DIO error control\n");
break;
case DIO_ERROR_DURATION:
printf("DIO error duratoin\n");
case DIO_ERROR_DURATION_20MS:
printf("DIO error! The duratoin is not a multiple of 20 ms\n");
break;
}
}
return 0;
}
```

DIO Program Make File Example

```
FNAME=tdio
CC=xscale-linux-gcc
STRIP=xscale-linux-strip
release:
$(CC) -o $(FNAME) $(FNAME).c -lmoxalib -lpthread
$(STRIP) -s $(FNAME)
debug:
$(CC) -DDEBUG -o $(FNAME)-dbg $(FNAME).cxx -lmoxalib -lpthread
clean:
/bin/rm -f $(FNAME) $(FNAME)-dbg *.o
```

UART

The normal tty device node is located at **/dev/ttyM0 … ttyM1.**

The UC-8481 supports Linux standard termios control. The Moxa UART Device API allows you to configure ttyM0 and ttyM1 as RS-232, RS-422, 4-wire RS-485, or 2-wire RS-485. The UC-8481 supports RS-232, RS-422, 2-wire RS-485, and 4-wire RS485.

You must include **<moxadevice.h>**.

```
#define RS232_MODE          0
#define RS485_2WIRE_MODE        1
#define RS422_MODE          2
#define RS485_4WIRE_MODE        3
```

1. Function: MOXA_SET_OP_MODE

   ```
   int ioctl(fd, MOXA_SET_OP_MODE, &mode)
   ```

   Description
   Set the interface mode. Argument 3 mode will pass to the UART device driver and change it.

2. Function: MOXA_GET_OP_MODE

   ```
   int ioctl(fd, MOXA_GET_OP_MODE, &mode)
   ```

   Description
   Get the interface mode. Argument 3 mode will return the interface mode.

There are two Moxa private ioctl commands for setting up special baudrates.

Function: MOXA_SET_SPECIAL_BAUD_RATE
Function: MOXA_GET_SPECIAL_BAUD_RATE

If you use this ioctl to set a special baudrate, the termios cflag will be B4000000, in which case the B4000000 definition will be different. If the baudrate you get from termios (or from calling tcgetattr()) is B4000000, you must call ioctl with MOXA_GET_SPECIAL_BAUD_RATE to get the actual baudrate.

Setinterface

The UC-8481 computer has 2 serial ports, labeled ttyM0 and ttyM1. The ports support RS-232, RS-422, and RS-485 2-wire and 4-wire operation modes with baudrate settings up to 921600 bps.

The default operation mode is set to RS-232. You can use the **setinterface** command to change the serial port operation mode.

Usage: setinterface device-node [interface-no]

device-node -- /dev/ttyMn; n = 0,1

interface-no -- As shown in the following table:

interface-no	Operation Mode
None	Display current setting
0	RS-232
1	RS-485 2-wire
2	RS-422
3	RS-485 4-wire

The following example sets /dev/ttyM0 to RS-422 mode

```
root@Moxa:/# setinterface /dev/ttyM0 2
root@Moxa:~# setinterface /dev/ttyM0
Now setting is RS422 interface.
root@Moxa:~#
```

Example for setting the baudrate

```
#include    <moxadevice.h>
#include    <termios.h>
struct termios  term;
int         fd, speed;
```

```
fd = open("/dev/ttyM0", O_RDWR);
tcgetattr(fd, &term);
term.c_cflag &= ~(CBAUD | CBAUDEX);
term.c_cflag |= B4000000;
tcsetattr(fd, TCSANOW, &term);
speed = 500000;
ioctl(fd, MOXA_SET_SPECIAL_BAUD_RATE, &speed);
```

Example for getting the baudrate

```
#include    <moxadevice.h>
#include    <termios.h>
struct termios  term;
int         fd, speed;
fd = open("/dev/ttyM0", O_RDWR);
tcgetattr(fd, &term);
if ( (term.c_cflag & (CBAUD|CBAUDEX)) != B4000000 )
{// follow the standard termios baud rate define} else
{ioctl(fd, MOXA_GET_SPECIAL_BAUD_RATE, &speed);}
```

Baudrate inaccuracy

Divisor = 921600/Target Baud Rate. (Only Integer part)
ENUM = 8 * (921600/Target - Divisor) (Round up or down)
Inaccuracy = (Target Baud Rate – 921600/(Divisor + (ENUM/8))) / Target Baud Rate * 100%
E.g.,
To calculate 500000 bps
Divisor = 1, ENUM = 7,
Inaccuracy = 1.7%

NOTE: The Inaccuracy should less than 2% for the device to work reliably.

Special Note

1. If the target baudrate is not a special baudrate (e.g., 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600), the termios cflag will be set to the same flag.
2. If you use stty to get the serial information, you will get a speed equal to 0.

Make File Example

The following Makefile file sample code is copied from the Hello example on the UC-8481's CD-ROM.

```
CC = xscale-linux-gcc
CPP = xscale-linux-gcc
SOURCES = hello.c
OBJS =  $(SOURCES:.c=.o)
all: hello
hello:   $(OBJS)
  $(CC) -o $@ $^ $(LDFLAGS) $(LIBS)
clean:
  rm -f $(OBJS) hello core *.gdb
```

Programming the Configurable LED

There is one programmable LED on the front of UC-8481. The file system device node for the programmable LED is located at /proc/driver/pled/. To enable/disable the programmable LED, you could write 1/0 through /proc/driver/pled/en. To turn on/off the programmable LED, you could write 1/0 through /proc/driver/pled/of_off.

To turn on the LED, use the following command.

```
192.168.3.120 – Putty
Moxa:~# echo "1" > /proc/driver/pled/on_off
```

To turn off the LED, use the following command.

```
192.168.3.120 – Putty
Moxa:~# echo "0" > /proc/driver/pled/on_off
```

To disable the LED, use the following command.

```
192.168.3.120 – Putty
Moxa:~# echo "0" > /proc/driver/pled/en
```

To enable the LED, use the following command.

```
192.168.3.120 – Putty
Moxa:~# echo "1" > /proc/driver/pled/en
```

Software Lock

"Software Lock" is an innovative technology developed by the Moxa engineering team, and can be used by a system integrator or developer to protect applications from being copied. An application is compiled into a binary format bound to the embedded computer, and the operating system that the application runs on. As long as it is obtained from the computer, it can be installed on the same hardware and under the same operating system, resulting in a loss of the add-on value created by the developer.

Users can deploy this data encryption method to develop the software for the applications. The binary file associated with each of your applications needs to undergo an additional encryption process after you have developed it. The process requires you to install an encryption key in the target computer.

1. Choose an encryption key (e.g.,"ABigKey") and install it in the target computer by a pre-utility program called 'setkey'.

   ```
   #setkey ABigKey
   ```

   NOTE: Use the following command to clear the encryption key on the target computer.

   ```
   #setkey ""
   ```

2. Develop and compile your program on the development PC.

3. On the development PC, run the utility program 'binencryptor' to encrypt your program with an encryption key.

   ```
   #binencryptor yourProgram ABigKey
   ```

4. Upload the encrypted program file to the target computer by FTP or NFS and test the program.

The encryption key is a computer-wise key. This means that the computer has only one key installed. Running the program ‘setkey’ multiple times overrides the existing key.

To prove the effectiveness of this software protection mechanism, prepare a target computer on which an encryption key has not been installed, or install a key different from that used to encrypt your program. In either case, the encrypted program fails immediately.

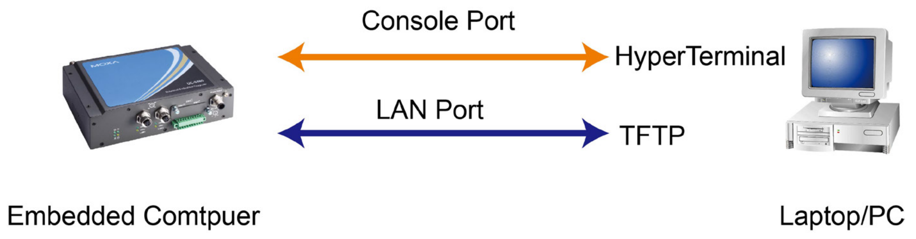
This mechanism also allows computers with an encryption key installed to bypass programs that are not encrypted. This is handy in the development phase since you can develop your programs and test them cleanly on the target computer.

A

Firmware Upgrade Procedure

Moxa provides a boot loader utility for firmware upgrade or recovery. You will need the following items to use this utility.

1. The embedded computer that you would like to upgrade or recover.
2. A PC or a laptop computer.
3. A console port cable for connecting through HyperTerminal.
4. A cross-over Ethernet cable for upgrading the firmware through a TFTP server and LAN port.
5. The firmware for the embedded computer.



There are three steps in the recovery process.

A. Configure HyperTerminal for the console port.

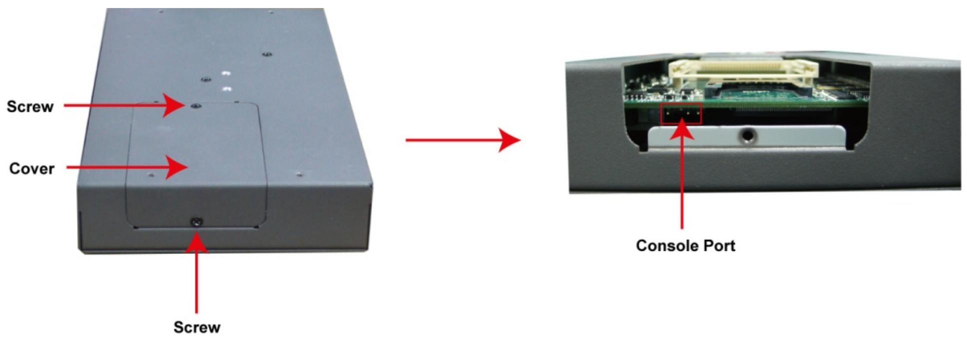
B. Download and installation of the TFTP program.

C. Download the firmware and upgrade through HyperTerminal.

If you are familiar with Moxa embedded computers and the firmware upgrade procedure, you may skip to Step C. However, we suggest that you go through all three steps to ensure the firmware upgrades properly.

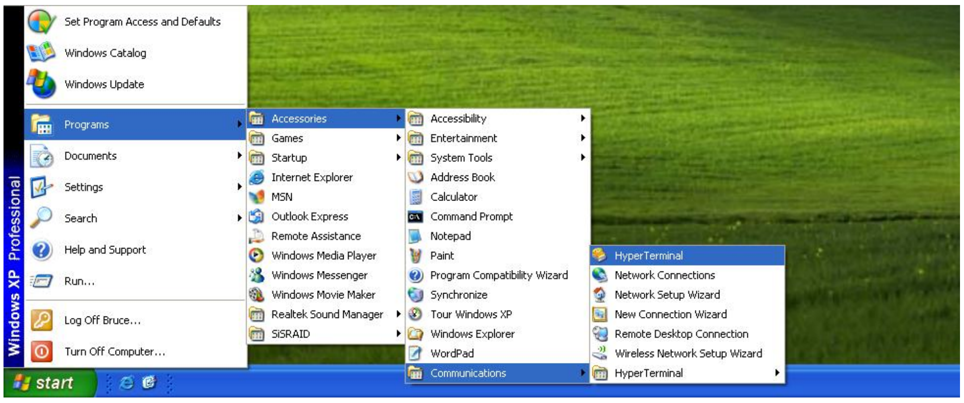
A. Configure HyperTerminal for the Console Port.

1. Unscrew the cover on the back of the UC-8481 series embedded computer.
2. Connect the console port with a console port cable to your PC.

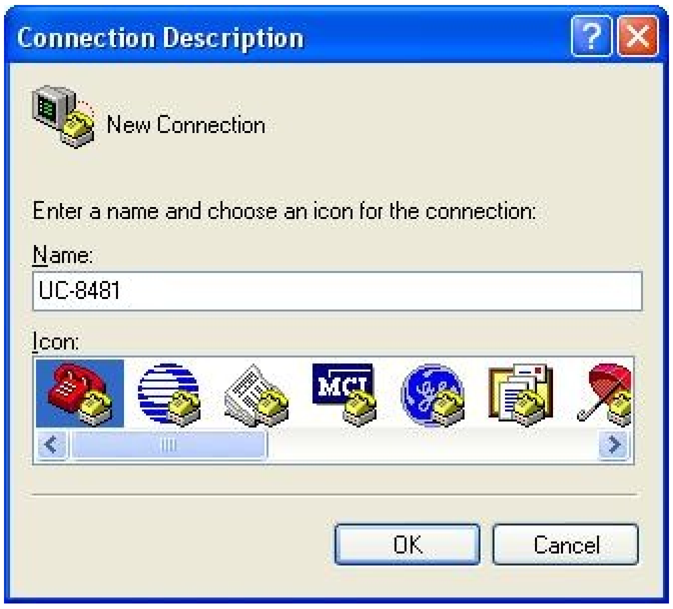


3. Connect your embedded computer to the power source.

4. Launch a serial communication tool to access your embedded computer. We suggest that you use HyperTerminal, which is built into Windows XP. Click **Start → Programs → Accessories → Communications** and then select **HyperTerminal**.



5. After HyperTerminal launches, enter UC-8481 or another name for the connection.


Connection Description

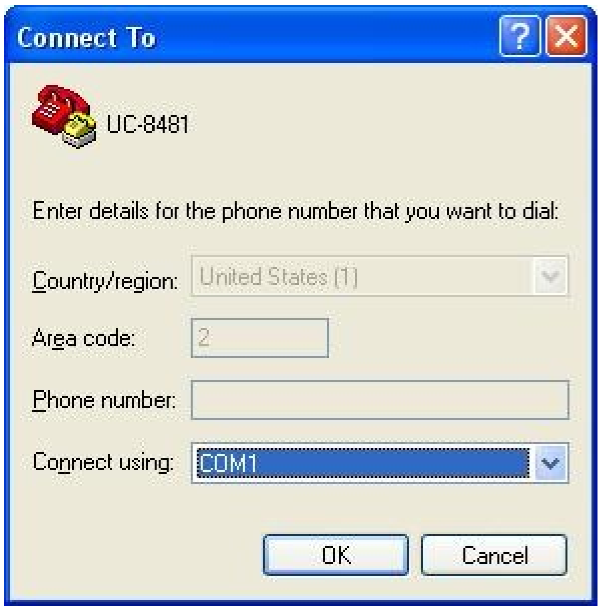
New Connection

Enter a name and choose an icon for the connection:

Name: UC-8481

Icon:

6. Click **Cancel** when the “Connect To” window opens.


Connect To

UC-8481

Enter details for the phone number that you want to dial:

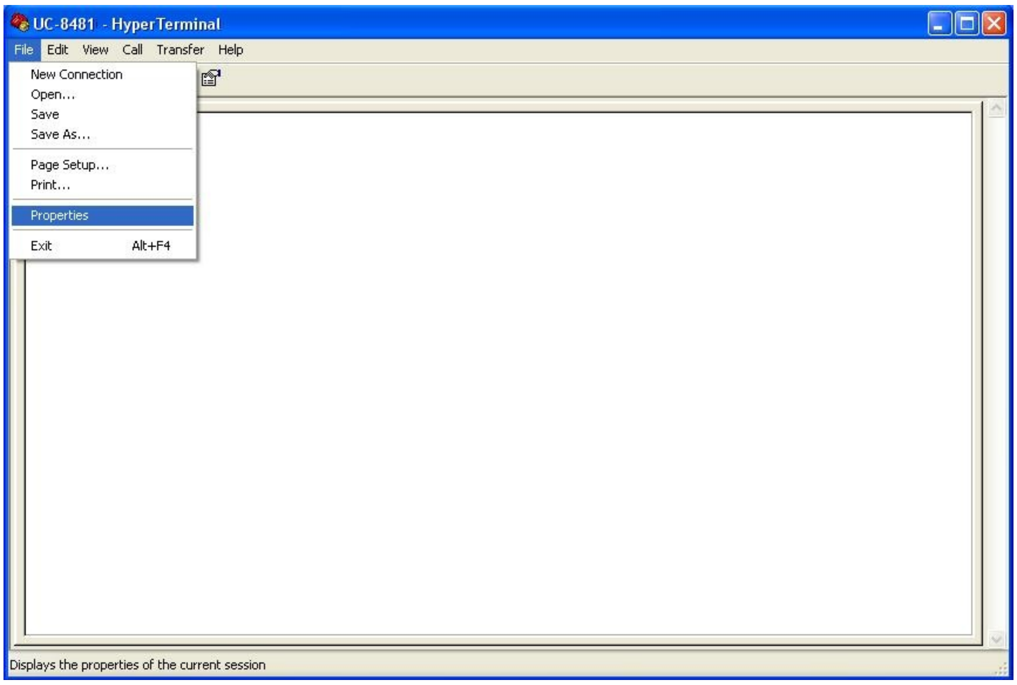
Country/region: United States (1)

Area code: 2

Phone number:

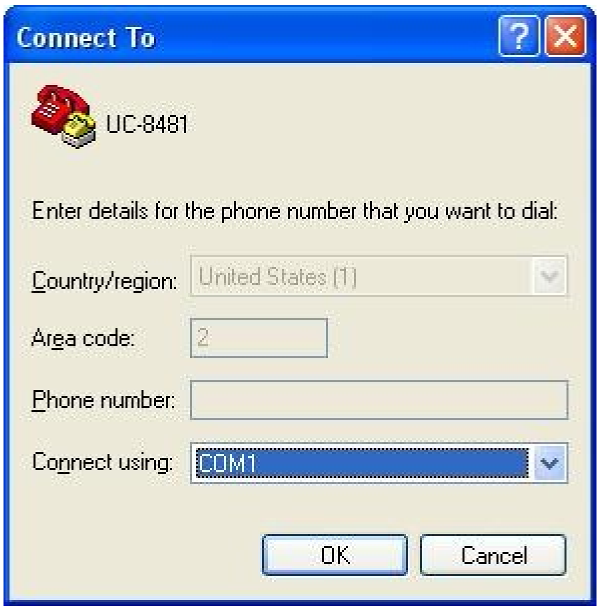
Connect using: COM1

7. Select **File → Properties** from the main HyperTerminal screen.



Displays the properties of the current session

8. You can change the COM port number in the Properties window. Click **Configure** for additional configuration options.



UC-8481

Enter details for the phone number that you want to dial:

Country/region: United States (1)

Area code: 2

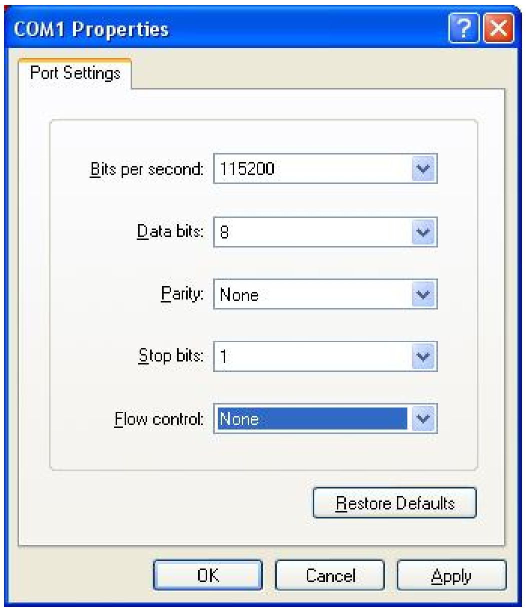
Phone number:

Connect using: COM1

9. Configure the **Port Settings** with following parameters:

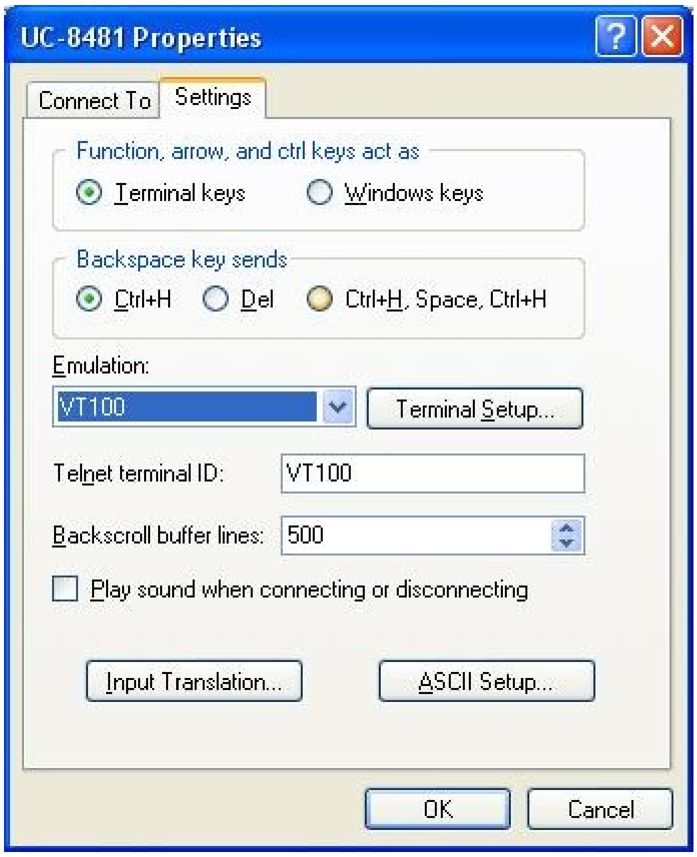
   - Bits per second: 115200
   - Data bits: 8
   - Parity: None
   - Stop bit: 1
   - Flow control: None.

   Click **OK** to continue.



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

10. Click the **Settings** tab and then select **VT100** (for Emulation). Click **OK** to complete the configuration.



Function, arrow, and ctrl keys act as: [✓ Terminal keys] Windows keys
Backspace key sends: [✓ Ctrl+H] Del Ctrl+H, Space, Ctrl+H
Emulation: VT100
Telnet terminal ID: VT100
Backscroll buffer lines: 500
Play sound when connecting or disconnecting

B. Download and Installation of the TFTP Program

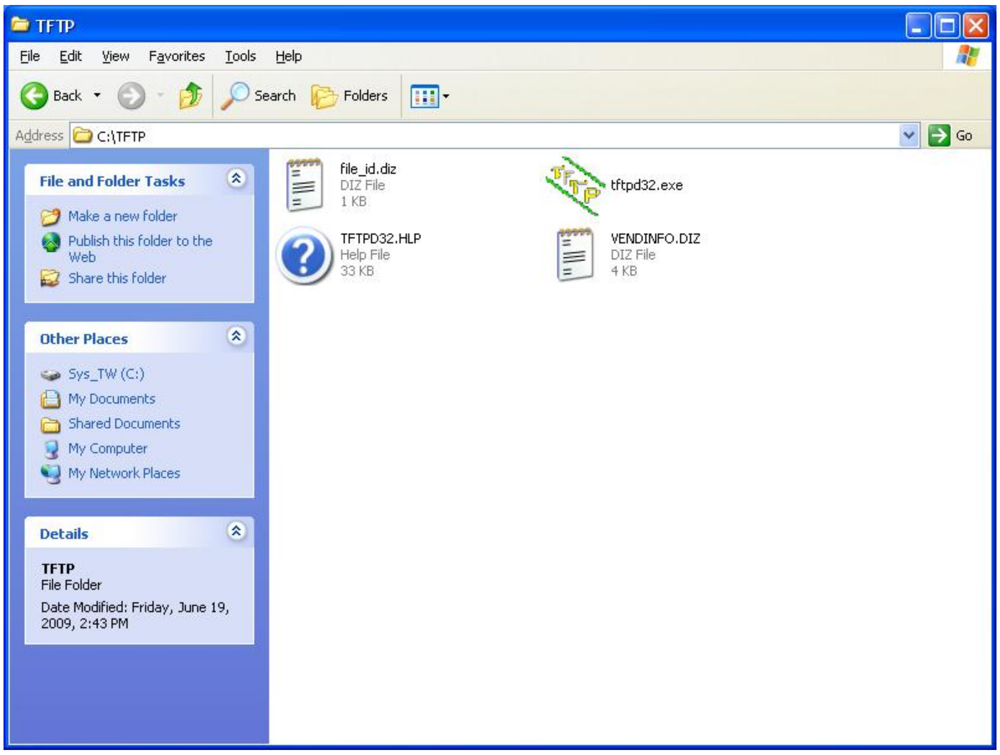
1. You will need to download a free TFTP server package to upgrade the firmware for the boot loader utility. Link to the following URL to download:

ftp://papa.indstate.edu/

Download the TFTP program located in the following path:
/winsock-1/Windows95/Daemons/TFTPD/

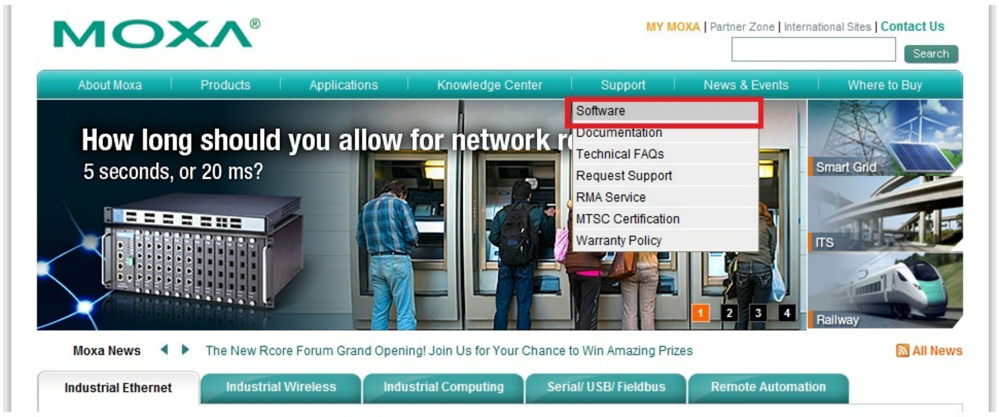
Download the file named **tftpd32d.zip**.

2. When you have finished downloading, extract the files to your PC.

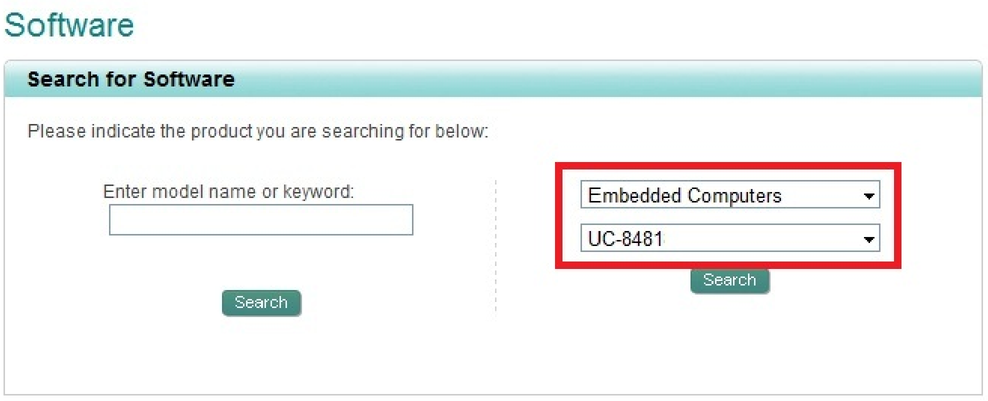


C. Download and Upgrade the Firmware through HyperTerminal.

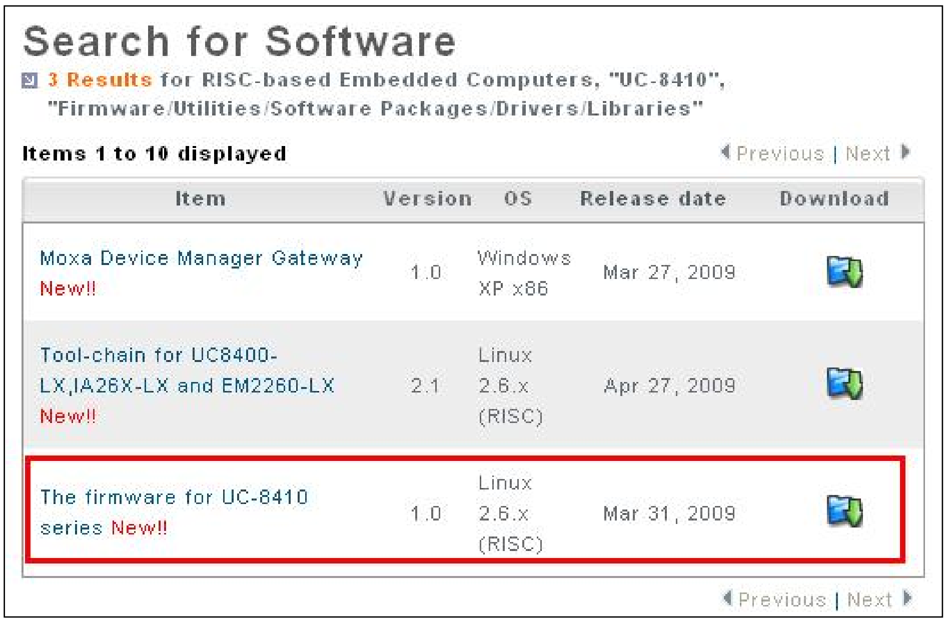
1. Connect to Moxa's website at http://www.moxa.com, and then select Software from the Support drop-down menu.



2. Under Search for Software, select the product line and then choose the specific product model. Click on **Search** to continue.


Software
Search for Software
Please indicate the product you are searching for below:
Enter model name or keyword:
Search
Embedded Computers
UC-8481
Search

3. In the software list, select the firmware for your model. Choose the appropriate OS and then click the download icon to start downloading the new firmware. **Note: Check the filename, it may differ from the filename shown below.**



Search for Software

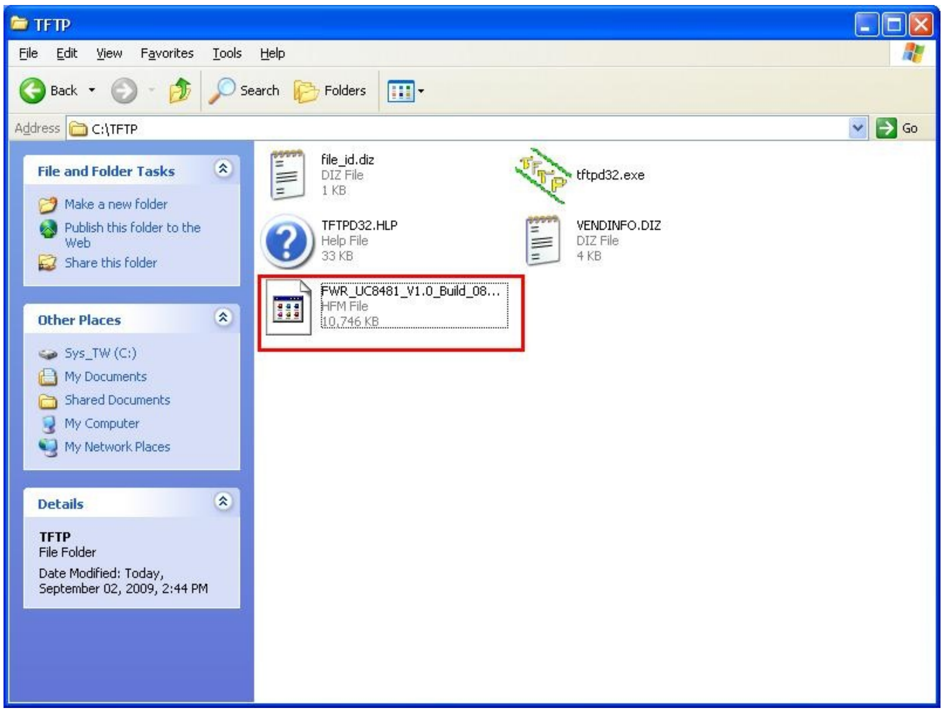
3 Results for RISC-based Embedded Computers, "UC-8410", "Firmware/Utilities/Software Packages/Drivers/Libraries"

Items 1 to 10 displayed ◀ Previous | Next ▶

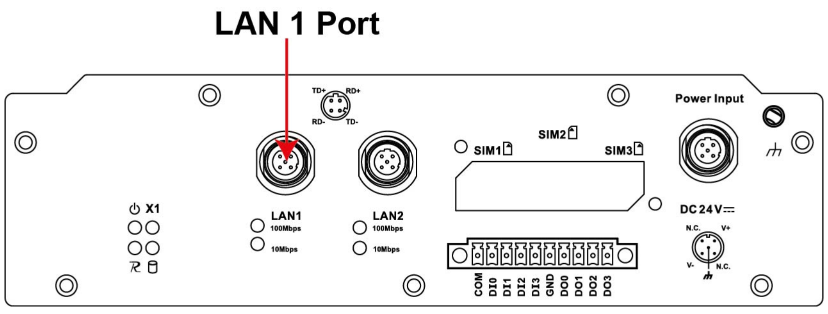
Item	Version	OS	Release date	Download
Moxa Device Manager Gateway New!!	1.0	Windows XP x86	Mar 27, 2009	
Tool-chain for UC8400-LX,IA26X-LX and EM2260-LX New!!	2.1	Linux 2.6.x (RISC)	Apr 27, 2009	
The firmware for UC-8410 series New!!	1.0	Linux 2.6.x (RISC)	Mar 31, 2009	

◀ Previous | Next ▶

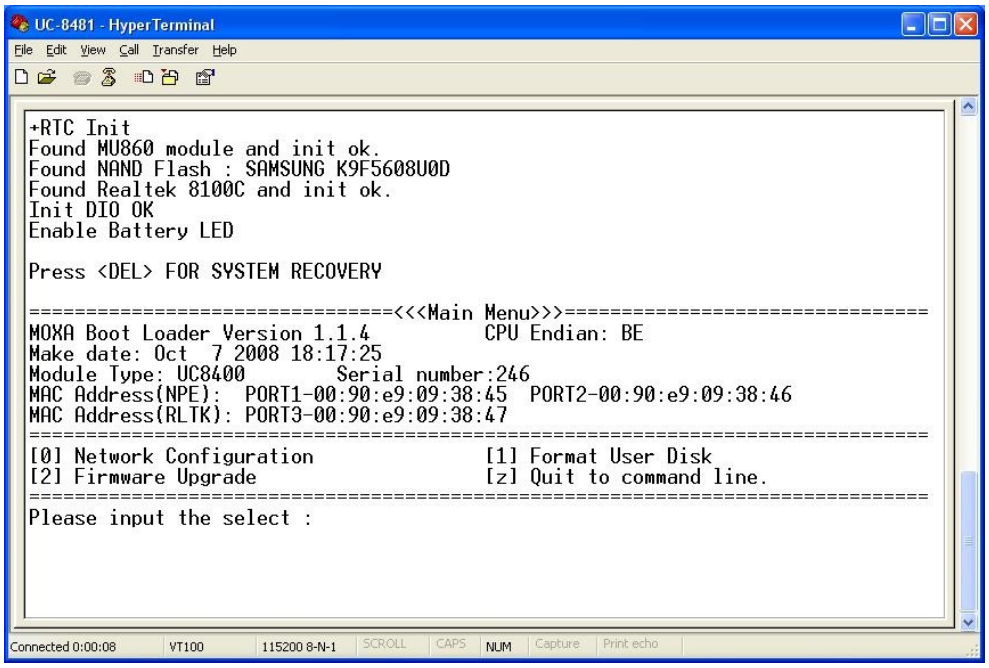
4. Put the latest firmware icon in the same directory as the TFTP files.



5. Next, connect **LAN1** of the embedded computer to your PC using a cross-over Ethernet cable. The LAN1 port is located on the rear panel of the embedded computer.

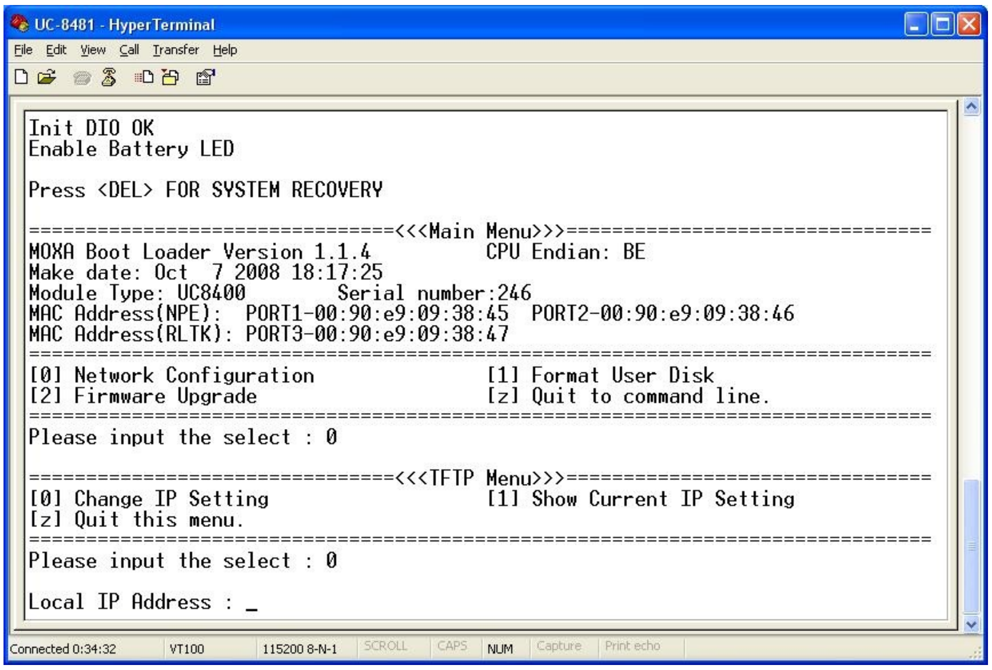


6. Press and hold down the **DEL** key on your PC and power on the embedded computer at the same time. You will be guided to the boot loader utility menu, as show below.



```
+RTC Init
Found MU860 module and init ok.
Found NAND Flash : SAMSUNG K9F5608U0D
Found Realtek 8100C and init ok.
Init DIO OK
Enable Battery LED

Press <DEL> FOR SYSTEM RECOVERY

================================<<<Main Menu>>>================================
MOXA Boot Loader Version 1.1.4           CPU Endian: BE
Make date: Oct  7 2008 18:17:25
Module Type: UC8400        Serial number:246
MAC Address(NPE):  PORT1-00:90:e9:09:38:45  PORT2-00:90:e9:09:38:46
MAC Address(RLTK): PORT3-00:90:e9:09:38:47
===============================================================================
[0] Network Configuration                [1] Format User Disk
[2] Firmware Upgrade                     [z] Quit to command line.
===============================================================================
Please input the select :
```

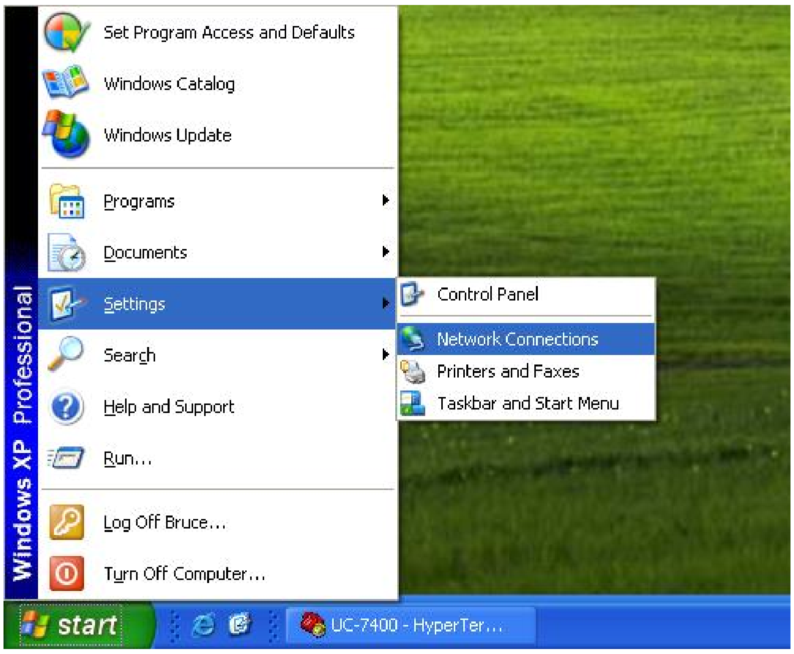
7. In the boot loader utility, select **[0] Network Configuration**, and then **[0] Change IP Setting** to configure IP addresses.



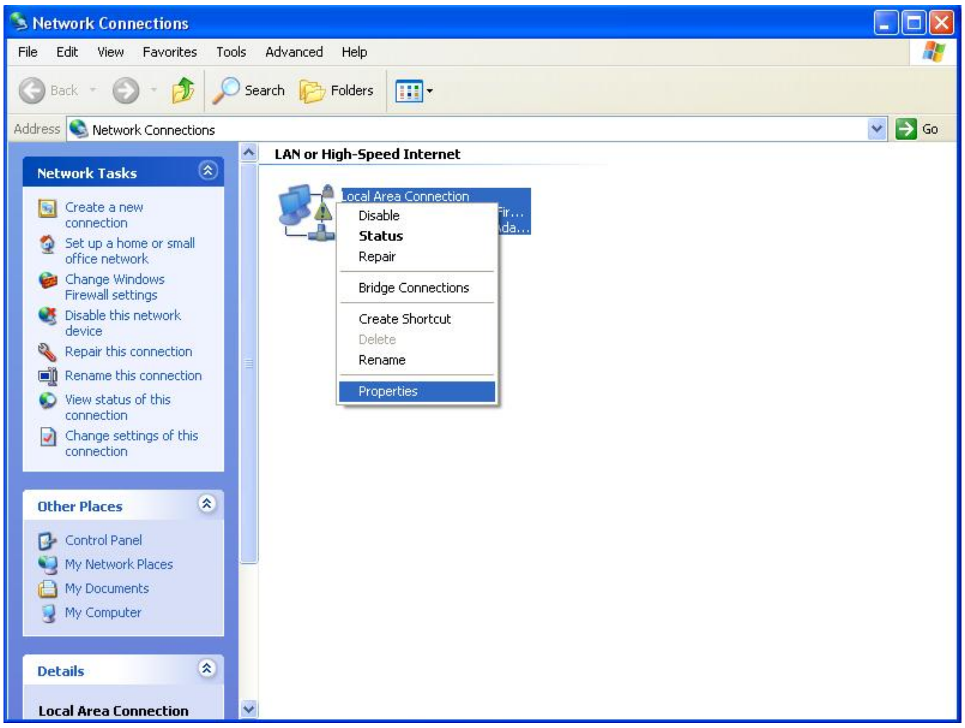
```
Init DIO OK
Enable Battery LED

Press <DEL> FOR SYSTEM RECOVERY

================================<<<Main Menu>>>================================
MOXA Boot Loader Version 1.1.4           CPU Endian: BE
Make date: Oct  7 2008 18:17:25
Module Type: UC8400        Serial number:246
MAC Address(NPE):  PORT1-00:90:e9:09:38:45  PORT2-00:90:e9:09:38:46
MAC Address(RLTK): PORT3-00:90:e9:09:38:47
===============================================================================
[0] Network Configuration                [1] Format User Disk
[2] Firmware Upgrade                     [z] Quit to command line.
===============================================================================
Please input the select : 0

================================<<<TFTP Menu>>>================================
[0] Change IP Setting                    [1] Show Current IP Setting
[z] Quit this menu.
===============================================================================
Please input the select : 0

Local IP Address : _
```

8. You will need to enter the IP address of the embedded computer and your PC. Follow the below to configure the IP addresses.

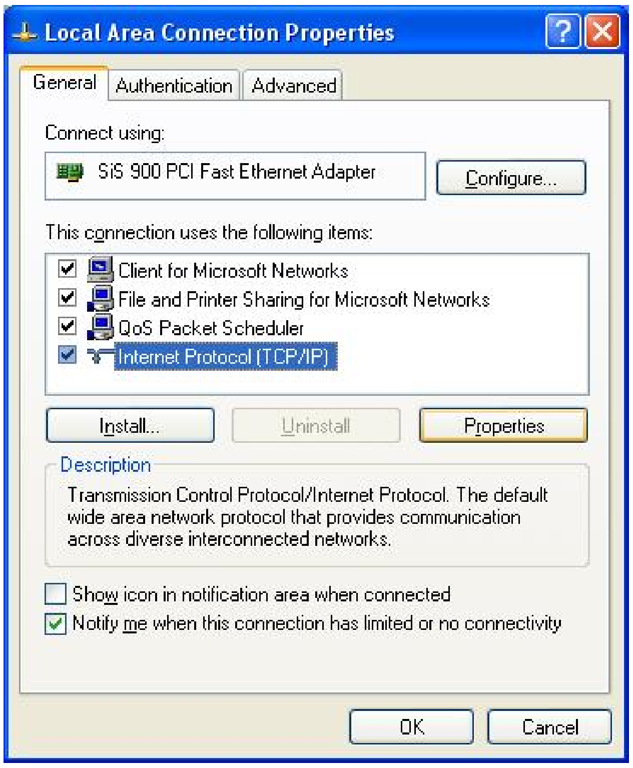
   a. From **Start → Settings**, select **Network Connections**.



   b. Right-click on **Local Area Connection**, and then select **Properties**.



c. Click the **General** tab and select **Internet Protocol (TCP/IP)**, and then click on **Properties**.



Local Area Connection Properties

General | Authentication | Advanced

Connect using:

SiS 900 PCI Fast Ethernet Adapter [Configure...]

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ QoS Packet Scheduler
- ☑ Internet Protocol (TCP/IP)

[Install...] [Uninstall] [Properties]

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

☐ Show icon in notification area when connected

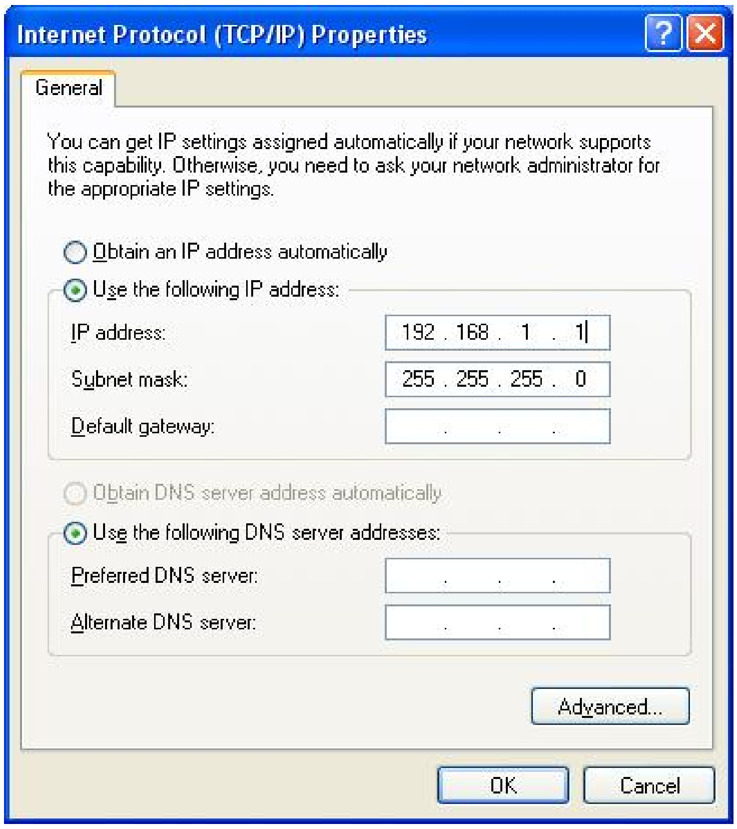
☑ Notify me when this connection has limited or no connectivity

[OK] [Cancel]

d. Next, select **Use the following IP address** and enter the **IP address** and **Subnet mask**.
<u>For example</u>:

IP address: 192.168.1.1 (This IP address is only an example; you may assign any IP address of your choice as long as it's on the same LAN as your PC.)

Subnet mask: 255.255.255.0



Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 192 . 168 . 1 . 1

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

○ Obtain DNS server address automatically

● Use the following DNS server addresses:

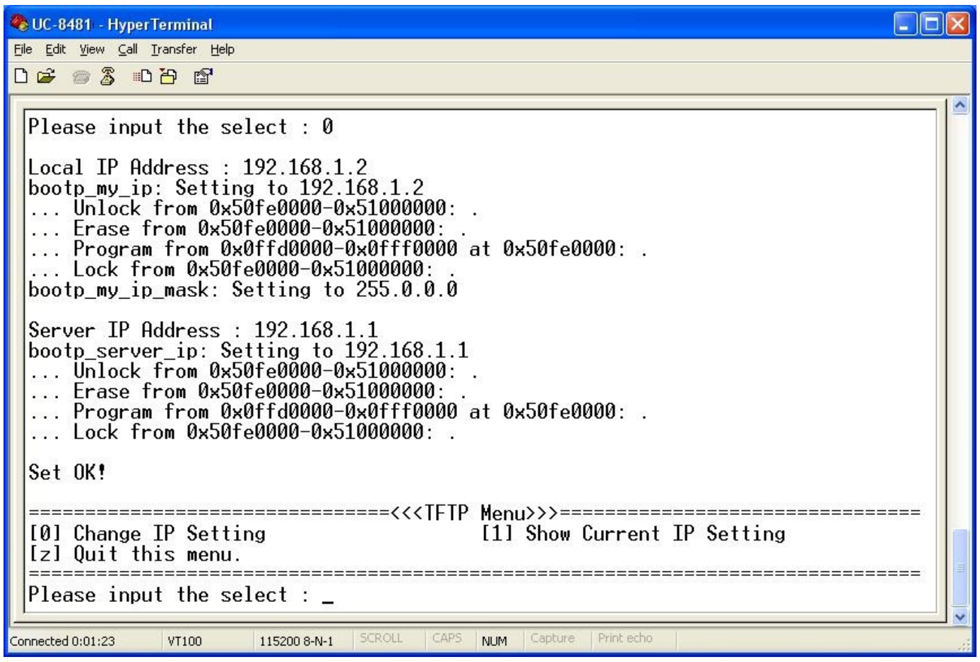
Preferred DNS server: . . .

Alternate DNS server: . . .

[Advanced...]

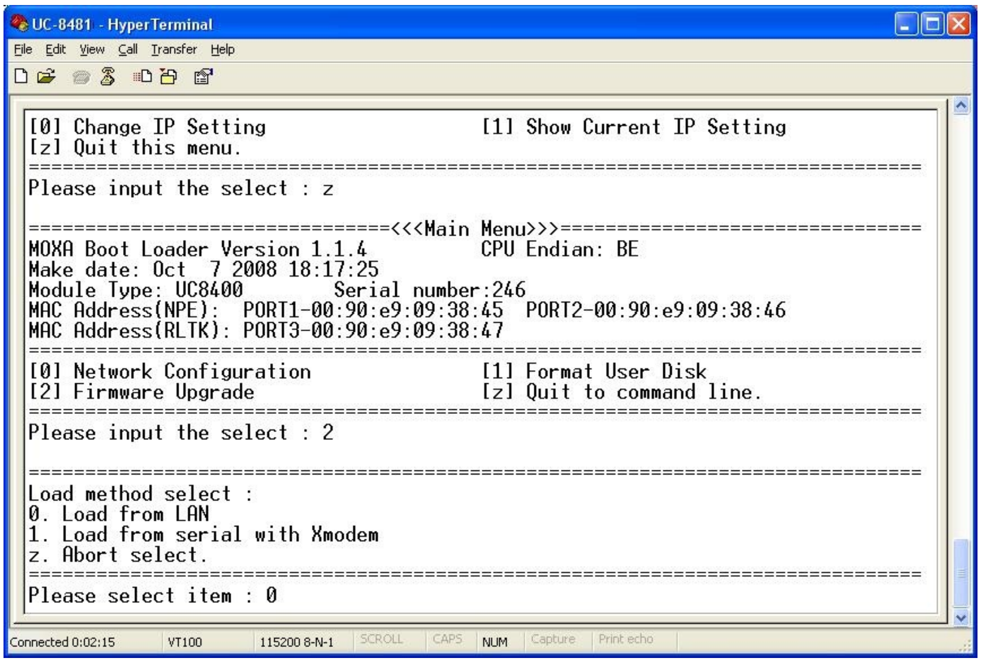
[OK] [Cancel]

9. Go back to the boot load utility menu and assign the local IP address, and then enter the server IP address. The local IP address is the IP address of the embedded computer. **Note that the local IP address must be on the same LAN as the server IP address.** For example, if the server IP address is 192.168.1.1, you can choose a local IP address between 192.168.1.2 to 192.168.1.254.



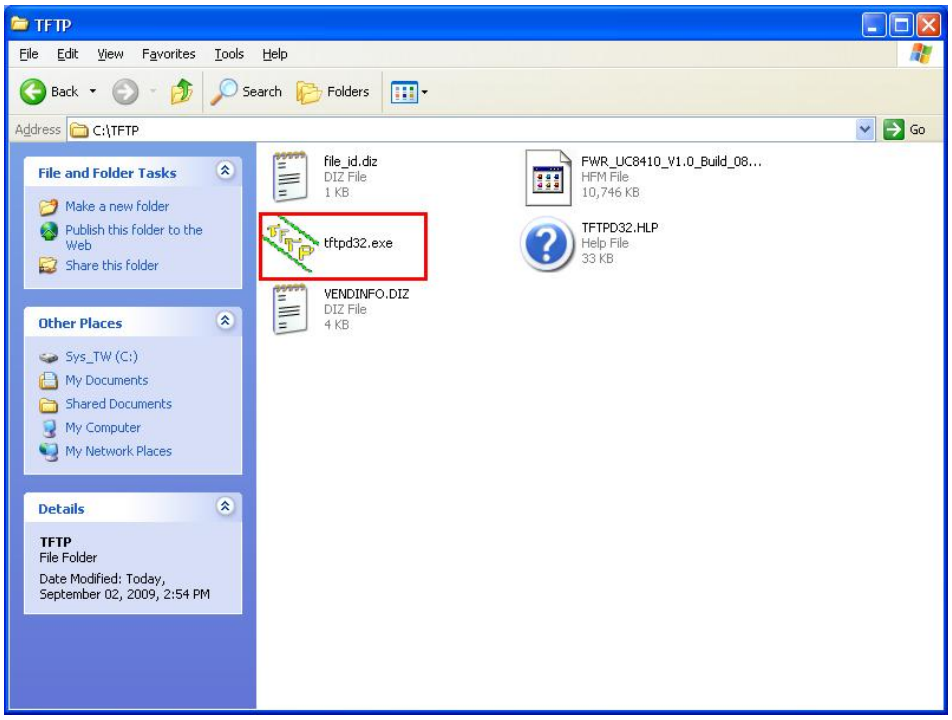
```
Please input the select : 0

Local IP Address : 192.168.1.2
bootp_my_ip: Setting to 192.168.1.2
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x0ffd0000-0x0fff0000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .
bootp_my_ip_mask: Setting to 255.0.0.0

Server IP Address : 192.168.1.1
bootp_server_ip: Setting to 192.168.1.1
... Unlock from 0x50fe0000-0x51000000: .
... Erase from 0x50fe0000-0x51000000: .
... Program from 0x0ffd0000-0x0fff0000 at 0x50fe0000: .
... Lock from 0x50fe0000-0x51000000: .

Set OK!

================================<<<TFTP Menu>>>================================
[0] Change IP Setting                   [1] Show Current IP Setting
[z] Quit this menu.
===============================================================================
Please input the select : _
```

10. Select **[z] Quit to command line** to exit the IP setting option. Next, select **[2] Firmware Upgrade**, and then **[0] Load from LAN** to continue.

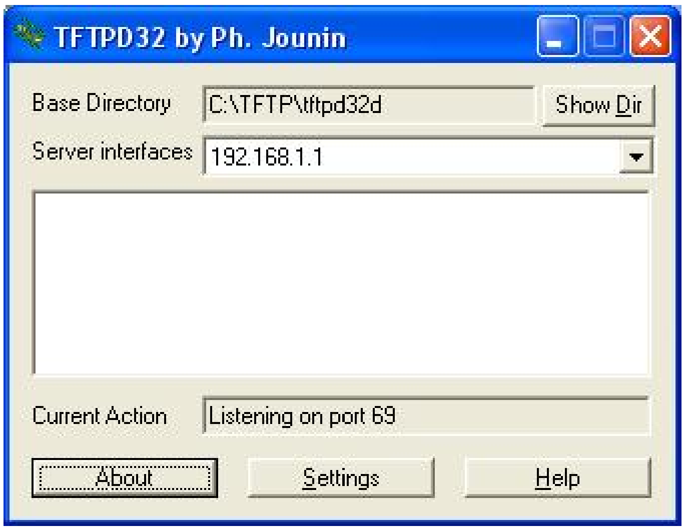


```
[0] Change IP Setting                   [1] Show Current IP Setting
[z] Quit this menu.
===============================================================================
Please input the select : z

================================<<<Main Menu>>>================================
MOXA Boot Loader Version 1.1.4          CPU Endian: BE
Make date: Oct  7 2008 18:17:25
Module Type: UC8400         Serial number:246
MAC Address(NPE):  PORT1-00:90:e9:09:38:45  PORT2-00:90:e9:09:38:46
MAC Address(RLTK): PORT3-00:90:e9:09:38:47
===============================================================================
[0] Network Configuration               [1] Format User Disk
[2] Firmware Upgrade                    [z] Quit to command line.
===============================================================================
Please input the select : 2

===============================================================================
Load method select :
0. Load from LAN
1. Load from serial with Xmodem
z. Abort select.
===============================================================================
Please select item : 0
```

11. To start the TFPF server, double-click on the **tftpd32** icon to launch the TFTP server.



12. When the TFTP server has been launched, the following screen will appear.

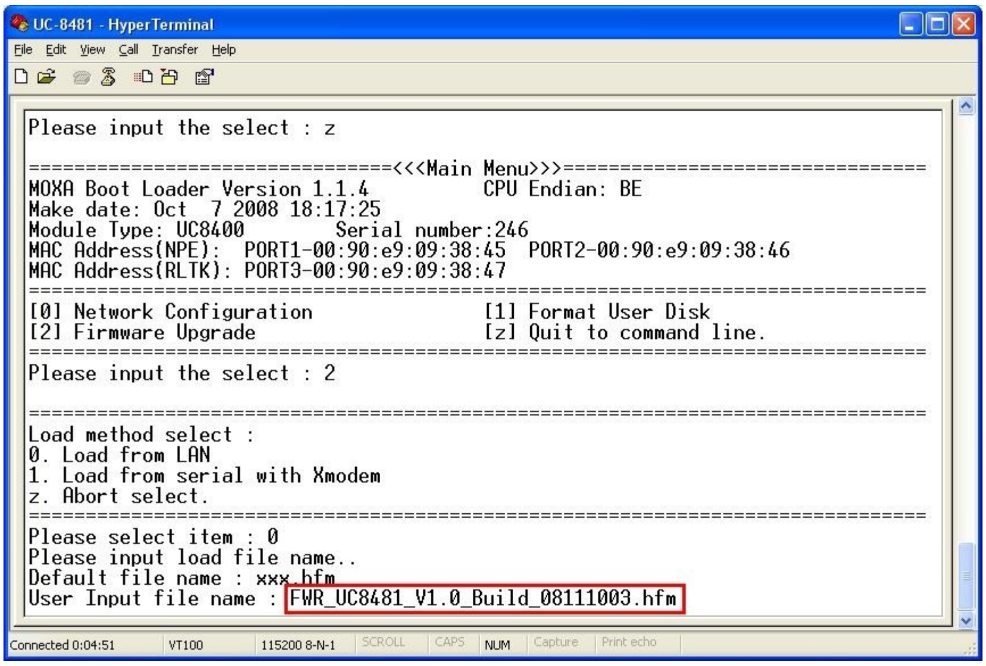

TFTPD32 by Ph. Jounin

Base Directory: C:\TFTP\tftpd32d

Server interfaces: 192.168.1.1

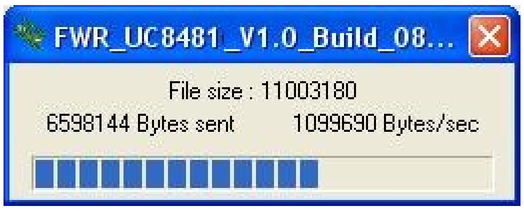
Current Action: Listening on port 69

13. Go back to the boot loader utility menu and enter the file name of the firmware image.


UC-8481 - HyperTerminal
File Edit View Call Transfer Help
Connected 0:04:51 VT100 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

```
Please input the select : z

================================<<<Main Menu>>>================================
MOXA Boot Loader Version 1.1.4          CPU Endian: BE
Make date: Oct  7 2008 18:17:25
Module Type: UC8400          Serial number:246
MAC Address(NPE):  PORT1-00:90:e9:09:38:45  PORT2-00:90:e9:09:38:46
MAC Address(RLTK): PORT3-00:90:e9:09:38:47
===============================================================================
[0] Network Configuration               [1] Format User Disk
[2] Firmware Upgrade                    [z] Quit to command line.
===============================================================================
Please input the select : 2

===============================================================================
Load method select :
0. Load from LAN
1. Load from serial with Xmodem
z. Abort select.
===============================================================================
Please select item : 0
Please input load file name..
Default file name : xxx.hfm
User Input file name : FWR_UC8481_V1.0_Build_08111003.hfm
```

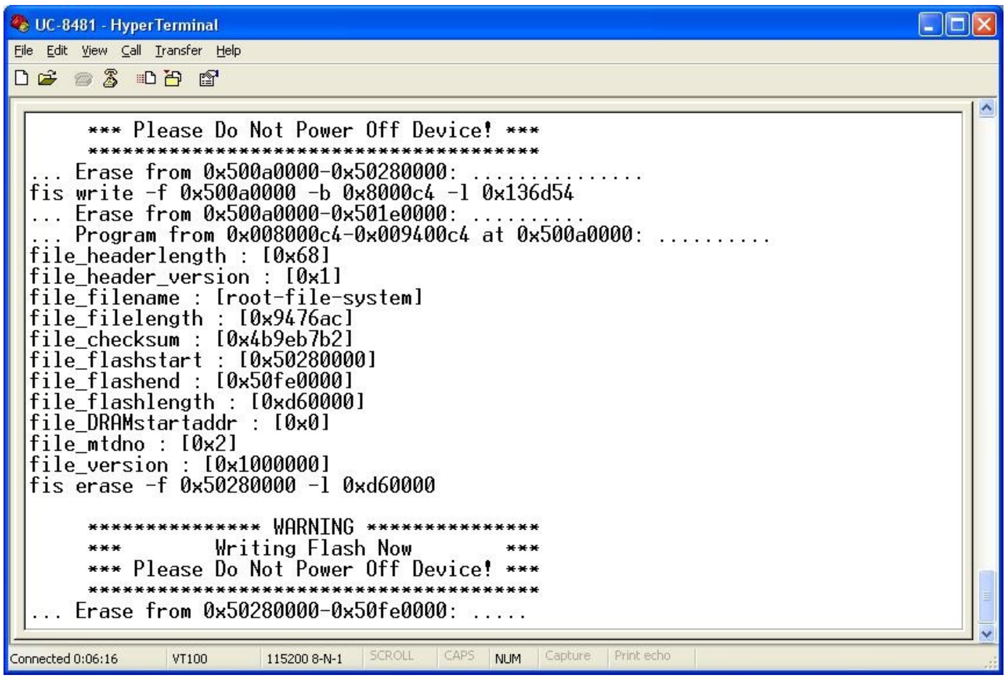
14. The firmware upgrade will then start to run.



File size : 11003180

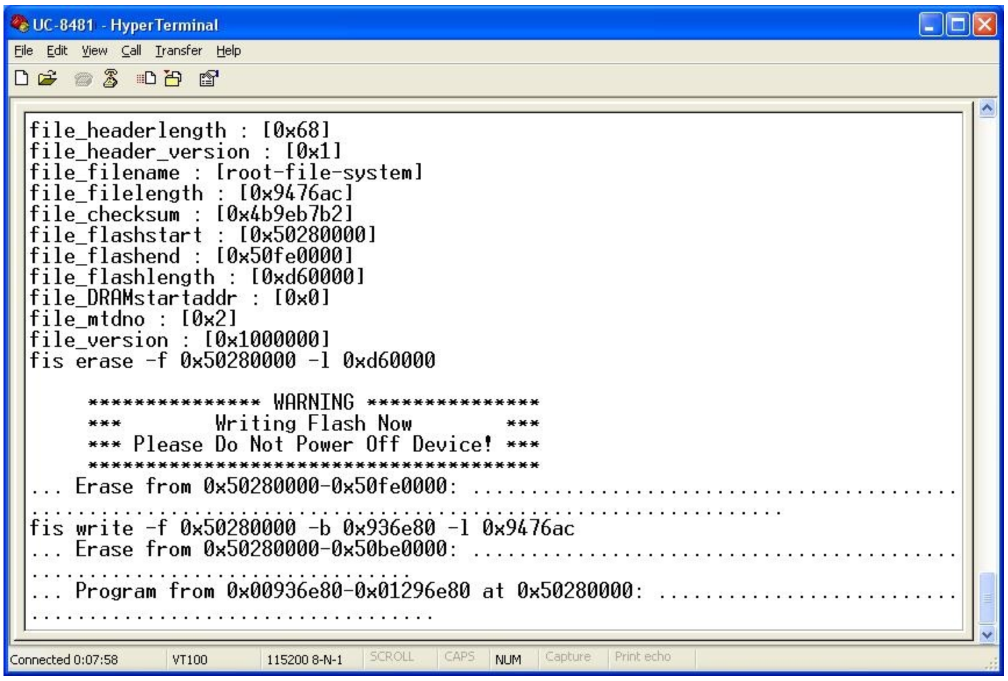
6598144 Bytes sent 1099690 Bytes/sec

15. It will take several minutes for the firmware files to be written to your embedded computer. **Do not power off your computer!**


UC-8481 - HyperTerminal
File Edit View Call Transfer Help
Connected 0:06:16 VT100 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

```
     *** Please Do Not Power Off Device! ***
     ***************************************
... Erase from 0x500a0000-0x50280000: ...............
fis write -f 0x500a0000 -b 0x8000c4 -l 0x136d54
... Erase from 0x500a0000-0x501e0000: ..........
... Program from 0x008000c4-0x009400c4 at 0x500a0000: ..........
file_headerlength : [0x68]
file_header_version : [0x1]
file_filename : [root-file-system]
file_filelength : [0x9476ac]
file_checksum : [0x4b9eb7b2]
file_flashstart : [0x50280000]
file_flashend : [0x50fe0000]
file_flashlength : [0xd60000]
file_DRAMstartaddr : [0x0]
file_mtdno : [0x2]
file_version : [0x1000000]
fis erase -f 0x50280000 -l 0xd60000

     *************** WARNING ***************
     ***         Writing Flash Now         ***
     *** Please Do Not Power Off Device! ***
     ***************************************
... Erase from 0x50280000-0x50fe0000: .....
```

16. When you see **Update OK**, the firmware upgrade is finished. At this point, you may reboot the embedded computer to complete the firmware upgrade or recovery from the boot loader utility.



```
file_headerlength : [0x68]
file_header_version : [0x1]
file_filename : [root-file-system]
file_filelength : [0x9476ac]
file_checksum : [0x4b9eb7b2]
file_flashstart : [0x50280000]
file_flashend : [0x50fe0000]
file_flashlength : [0xd60000]
file_DRAMstartaddr : [0x0]
file_mtdno : [0x2]
file_version : [0x1000000]
fis erase -f 0x50280000 -l 0xd60000

     *************** WARNING ***************
     ***        Writing Flash Now        ***
     *** Please Do Not Power Off Device! ***
     ***************************************
... Erase from 0x50280000-0x50fe0000: ........................................
.................................................................
fis write -f 0x50280000 -b 0x936e80 -l 0x9476ac
... Erase from 0x50280000-0x50be0000: ........................................
.................................
... Program from 0x00936e80-0x01296e80 at 0x50280000: .........................
...................................
```

Connected 0:07:58 VT100 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

17. If you cannot reboot your embedded computer (after following all the steps above), contact Moxa's technical support staff for further assistance.

B

System Commands

Linux Normal Command Utility Collection

File Manager

cp	copy file
ls	list file
ln	make symbolic link file
mount	mount and check file system
rm	delete file
chmod	change file owner & group & user
chown	change file owner
chgrp	change file group
chroot	runs a command with a specified root directory.
sync	sync file system; save system file buffer to hardware
mv	move file
pwd	display active file directly
df	list active file system space
mkdir	make new directory
rmdir	delete directory
find	search for files in a directory hierarchy
head	output the first part of files
mkfifo	creates a FIFO, special character file, or special block file with the specified name
mknod	creates a FIFO, special character file, or special block file with the specified name
touch	change file timestamps
which	Locate a program file in the user’s path.

Editor

vi	text editor
cat	dump file context
grep	Search string on file
egrep	search string on file of Extended regular expressions
fgrep	Search file(s) for lines that match a fixed string
cut	Get string on file
find	Find file where are there
more	dump file by one page
test	test if file exists or not
sleep	Sleep (seconds)
usleep	suspend execution for microsecond intervals
echo	echo string
sed	Steam editor

awk	pattern-directed scanning and processing language
expand	Converts all tabs to spaces
tail	Print the last 10 lines of each FILE to standard output.
tar	The GNU version of the tar archiving utility
tr	Translate, squeeze, and/or delete characters
wc	Print byte, word, and line counts, count the number of bytes, whitespace-separated words, and newlines in each given FILE, or standard input if non are given or for a FILE of '-'.

Network

arp	manipulate the system ARP cache
ping	ping to test network
route	routing table manager
netstat	display network status
ifconfig	set network IP address
tftp	IPV4 Trivial File Transfer Protocol client
telnet	Connects the local host with a remote host, using the Telnet interface.
ftp	file transfer protocol
ifdown, ifup	bring a network interface up, or take a network interface down
ip	show / manipulate routing, devices, policy routing and tunnels
tcpsvd	TCP/IP service daemon
wget	The non-interactive network downloader.

Process

kill	kill process
ps	display now running process
fuser	identify processes using files or sockets
killall	sends a signal to all processes running any of the specified commands
nice, renice	run a program with modified scheduling priority / alter priority of running processes
pidof	find the process ID of a running program
run-parts	run scripts or programs in a directory
start-stop-deamon	start and stop system daemon programs
top	display Linux tasks

Modules

insmod	insert a module into the kernel
lsmod	shows which kernel modules are currently loaded
modprobe	intelligently adds or removes a module from the Linux kernel
rmmod	remove module from kernel

Other

dmesg	dump kernel log message
zcat	Dump .gz file context
free	display system memory usage
date	print or set the system date and time
env	run a program in a modified environment
clear	clear the terminal screen
reboot	reboot or power off/on the server

halt	halt the server
du	estimate file space usage
gzip, gunzip	compress or expand files
hostname	show system's host namebasename return filename or directory portion of pathname
dirname	Convert a full pathname to just a path
expr	evaluate arguments as an expression
false	Do nothing, returning a non-zero (false) exit status
true	Do nothing, successfully
fdisk	Partition table manipulator for Linux
hwclock	A tool for accessing the Hardware Clock
id	Print the user identity
klogd	Kernel log daemon
logger	a shell command interface to the syslog system log module
md5sum	compute and check MD5 message digest
mesg	control write access to your terminal
mktemp	make temporary file name
nohup	No Hang Up
reset	terminal initialization
stty	change and print terminal line settings
syslogd	Linux system logging utilities
uname	Print system information, print information about the machine and operating system it is running on
uptime	Determine how long the system has been running
xargs	build and execute command lines from standard input
yes	'yes' prints the command line arguments, separated by spaces and followed by a newline, forever until it is killed.
tee	Copy standard input to each FILE, and also to standard output.

Special Moxa Utilities

setkey	set the software encryption key
upgradehfm	upgrade firmware utility
libmoxalib.a	Moxa perporitary libraries
upramdisk	mount ramdisk
downramdisk	unmount ramdisk
kversion	show kernel version
setinterface	set /dev/ttyMn to RS232/RS485-2WIRES/RS422/ RS485-4WIRES