UC-7400-LX Plus User's Manual

Sixth Edition, December 2010

www.moxa.com/product

UC-7400-LX Plus User's Manual

Technical Support Contact Information

www.moxa.com/support

Moxa Americas:	Moxa China (Shanghai office):
Toll-free: 1-888-669-2872	Toll-free: 800-820-5036
Tel: +1-714-528-6777	Tel: +86-21-5258-9955
Fax: +1-714-528-6778	Fax: +86-10-6872-3958
Moxa Europe:	Moxa Asia-Pacific:
Tel: +49-89-3 70 03 99-0	Tel: +886-2-8919-1230
Fax: +49-89-3 70 03 99-99	Fax: +886-2-8919-1231

Table of Contents

Chapter 1 Introduction 1-1
Overview 1-2
Software Architecture 1-2
    Journaling Flash File System (JFFS2) 1-3
    Software Package 1-4

Chapter 2 Getting Started 2-1
Powering on the UC-7400-LX Plus 2-2
Connecting the UC-7400-LX Plus to a PC 2-2
    Serial Console 2-2
    Telnet Console 2-3
    SSH Console 2-4
Configuring the Ethernet Interface 2-5
    Modifying Network Settings with the Serial Console 2-5
    Modifying Network Settings over the Network 2-6
    Using IPv6 Command Utilities 2-6
    Configuring the WLAN through the PCMCIA Interface 2-8
    Connecting to 3G Networks through the PCMCIA Interface 2-14
Test Program—Developing Hello.c 2-16
    Installing the Tool Chain (Linux) 2-16
    Checking the Flash Memory Space 2-17
    Compiling Hello.c 2-17
    Uploading and Running the “Hello” Program 2-18
Developing Your First Application 2-18
    Testing Environment 2-18
    Compiling tcps2.c 2-19
    Uploading and Running the “tcps2-release” Program 2-20
    Testing Procedure Summary 2-22

Chapter 3 Managing Embedded Linux 3-1
System Version Information 3-2
System Image Backup 3-2
    Upgrading the Firmware 3-2
    Loading Factory Defaults 3-5
Enabling and Disabling Daemons 3-5
Starting a Program Automatically at Run-Level 3-8
Adjusting the System Time 3-9
    Setting the Time Manually 3-9
    Write current system time to RTC 3-9
    NTP Client 3-10
    Updating the Time Automatically 3-10
Cron—Daemon for Executing Scheduled Commands 3-11
Connecting Peripherals 3-12
    USB Mass Storage 3-12
    CF Mass Storage 3-12

Chapter 4 Managing Communications 4-1
Telnet / FTP 4-2
DNS 4-2

Web Service—Apache 4-3
IPTABLES 4-4
NAT 4-8
    NAT Example 4-9
    Enabling NAT at Bootup 4-9
Dial-up Service—PPP 4-10
PPPoE 4-13
NFS (Network File System) 4-15
    Setting up the UC-7400-LX Plus as an NFS Server 4-15
    Setting up the UC-7400-LX Plus as an NFS Client 4-17
Mail 4-17
SNMP 4-18
OpenVPN 4-19
IPv6 to IPv4 Tunneling 4-26

Chapter 5 Programmer's Guide 5-1
Before Programming Your Embedded System 5-2
    Caution When Using File Systems 5-2
    Using RAM File System Instead of Flash File System 5-2
Flash Memory Map 5-2
Linux Tool Chain Introduction 5-3
Debugging with GDB 5-4
Device API 5-5
RTC (Real-time Clock) 5-5
Buzzer 5-5
WDT (Watch dog Timer) 5-6
UART 5-9
LCM 5-11
KeyPad 5-11
Make File Example 5-12

Chapter 6 Software Lock 6-1

Appendix A System Commands A-1
Linux normal command utility collection A-1
    System A-1
    File Manager A-1
    Editor A-2
    Network A-2
    Process A-2
    Other A-3
    Moxa Special Utilities A-3

1 Introduction

The Moxa UC-7400-LX Plus series of RISC-based commnication platforms are ideal for your embedded applications. All models come with 8 RS-232/422/485 serial ports, dual 10/100 Mbps Ethernet ports, a PCMCIA interface for wireless LAN communication, and CompactFlash and USB ports for mass storage for disk expansion.

The following topics are covered in this chapter:

- ❑ **Overview**
- ❑ **Software Architecture**
  - ➢ Journaling Flash File System (JFFS2)
  - ➢ Software Package

Overview

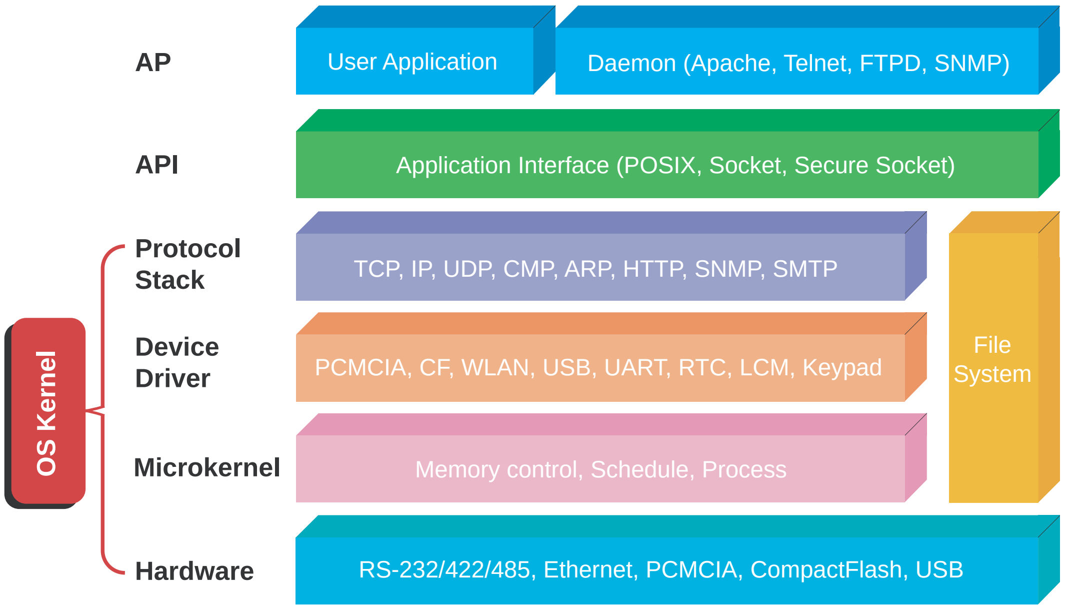
The UC-7400-LX Plus series of RISC-based communication platforms are ideal for embedded applications. Features include 8 RS-232/422/485 serial ports, dual 10/100 Mbps Ethernet ports, a PCMCIA interface for wireless LAN communication, and CompactFlash and USB ports for mass storage disk expansion.

The UC-7400-LX Plus uses an Intel XScale IXP425 533 Mhz RISC CPU. Unlike the X86 CPU, which uses a CISC design, the IXP425's RISC design architecture and modern semiconductor technology provide the UC-7400-LX Plus with a powerful computing engine and communication functions, but without generating a lot of heat. The built-in 32 MB NOR Flash ROM and 128 MB SDRAM give you enough memory to put your application software directly on UC-7400-LX Plus. In addition, since the dual-LAN ports are built right into the IXP425 CPU, the UC-7400-LX Plus is an ideal communication platform for network security applications. If your application requires placing the UC-7400-LX Plus in a location that is not near an Ethernet LAN connection, you can use the UC-7400-LX Plus's PCMCIA port to attach a wireless LAN card.

The pre-installed Linux operating system provides an open software operating system for your software program development. Software written for desktop PCs can be easily ported to the UC-7400-LX Plus platform with a GNU cross compiler, without needing to modify the source code. All of the necessary device drivers, such as a PCMCIA wireless LAN module and keypad, LCM, and buzzer control, are also included with the UC-7400-LX Plus. The operating system, device drivers, and the software you develop for your own application, can all be stored in the UC-7400-LX Plus's Flash memory.

Software Architecture

The Linux operating system that is pre-installed in the UC-7400-LX Plus follows the standard Linux architecture, making it easy to port programs that follow the POSIX standard to the UC-7400-LX Plus. Porting is done with the GNU Tool Chain provided by Moxa. In addition to the Standard POSIX API, device drivers for the buzzer and CompactFlash mass storage, UART, digital input, digital output, and wireless LAN PCMCIA card are also included in the UC-7400-LX Plus Linux system.


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

The UC-7400-LX Plus’s built-in Flash ROM is partitioned into Boot Loader, Linux Kernel, Root File System, and User Root File System partitions.

In order to prevent user applications from crashing the Root File System, the UC-7400-LX Plus uses a specially designed Root File System with protected configuration for emergency use. This Root File System comes with serial and Ethernet communication capability for users to load the Factory Default Image file. The user directory saves the user’s settings and applications.

To improve system reliability, the UC-7400-LX Plus has a built-in mechanism that prevents the system from crashing. When the Linux kernel boots up, the kernel will mount the root file system for read only, and then enable services and daemons. During this time, the kernel will start searching for system configuration parameters via rc or inittab.

Normally, the kernel uses the Root File System to boot up the system. Since the Root File System is protected and cannot be changed by the user, this provides a “safe” zone.

For more information about the memory map and programming, refer to Chapter 5, “Programmer’s Guide.”

Journaling Flash File System (JFFS2)

The User Root File System in the flash memory is formatted with the **Journaling Flash File System (JFFS2)**. The formatting process places a compressed file system in the flash memory, transparent to the user.

The Journaling Flash File System (JFFS2), which was developed by Axis Communications in Sweden, puts a file system directly on the flash, instead of emulating a block device. It is designed for use on flash-ROM chips and recognizes the special write requirements of a flash-ROM chip. JFFS2 implements wear-leveling to extend the life of the flash disk, and stores the flash directory structure in the RAM. A log-structured file system is maintained at all times. The system is always consistent, even if it encounters crashes or improper power-downs, and does not require ***fsck*** (file system check) on boot-up.

JFFS2 is the newest version of JFFS. It provides improved wear-leveling and garbage-collection performance, improved RAM footprint and response to system-memory pressure, improved concurrency and support for suspending flash erases; marking of bad sectors with continued use of the remaining good sectors (which enhances the write-life of the devices), native data compression inside the file system design, and support for hard links.

The key features of JFFS2 are:

- Targets the Flash ROM directly
- Robustness
- Consistency across power failures
- No integrity scan (fsck) is required at boot time after normal or abnormal shutdown
- Explicit wear leveling
- Transparent compression

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

Additional information about JFFS2 is available at:

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

http://developer.axis.com/software/jffs/

http://www.linux-mtd.infradead.org/

Software Package

Boot Loader	Redboot (v1.92)
Kernel	Monta Vista embedded Linux 2.6.10
Protocol Stacks	ARP, PPP, CHAP, PAP, IPv4, ICMP, TCP, UDP, DHCP, FTP, SNMP V1, HTTP, NTP, NFS, SMTP, SSH 1.0/2.0, SSL, Telnet, PPPoE, OpenVPN
File System	JFFS2, NFS, Ext2, Ext3, VFAT/FAT
OS shell command	bash
Busybox	Linux normal command utility collection
Utilities	
tinylogin	login and user manager utility
telnet	telnet client program
ftp	FTP client program
smtpclient	email utility
scp	Secure file transfer Client Program
Daemons	
pppd	dial in/out over serial port daemon
snmpd	snmpd agent daemon
telnetd	telnet server daemon
inetd	TCP server manager program
ftpd	ftp server daemon
apache	web server daemon
sshd	secure shell server
nfs-user-server	network file system server
openvpn	virtual private network
openssl	open SSL
Linux Tool Chain	
Gcc (V3.4.3)	C/C++ PC Cross Compiler
GDB (V6.3)	Source Level Debug Server
Glibc (V2.2.5)	POSIX standard C library

2
Getting Started

In this chapter, we explain how to connect the UC-7400-LX Plus, turn on the power, and then get started using the programming and other functions.

The following topics are covered in this chapter:

- **Powering on the UC-7400-LX Plus**
- **Connecting the UC-7400-LX Plus to a PC**
  - Serial Console
  - Telnet Console
  - SSH Console
- **Configuring the Ethernet Interface**
  - Modifying Network Settings with the Serial Console
  - Modifying Network Settings over the Network
  - Using IPv6 Command Utilities
  - Configuring the WLAN through the PCMCIA Interface
  - Connecting to 3G Networks through the PCMCIA Interface
- **Test Program—Developing Hello.c**
  - Installing the Tool Chain (Linux)
  - Checking the Flash Memory Space
  - Compiling Hello.c
  - Uploading and Running the “Hello” Program
- **Developing Your First Application**
  - Testing Environment
  - Compiling tcps2.c
  - Uploading and Running the “tcps2-release” Program
  - Testing Procedure Summary

Powering on the UC-7400-LX Plus

Connect the SG wire to the Shielded Contact located in the upper left corner of the UC-7400-LX Plus, 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, and the model name of the computer will appear on the LCM display.

NOTE After connecting the UC-7400 LX Plus 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 turn on until the operating system is ready.

Connecting the UC-7400-LX Plus to a PC

There are two ways to connect the UC-7400-LX Plus to a PC: (1) Through the serial console port, and (2) via Telnet over the network.

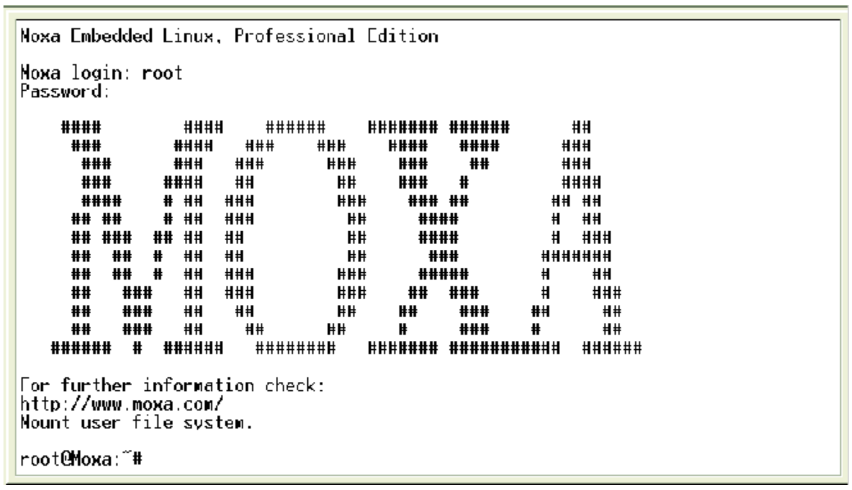
Serial Console

The serial console port gives users a convenient way of connecting to the UC-7400-LX Plus 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 that you do not need to know any of the IP addresses in order to connect to the serial console utility.

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, the following window will open.



```
Moxa Embedded Linux, Professional Edition

Moxa login: root
Password:

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

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

root@Moxa:~#
```

To log in, type the Login name and password as requested. The default values are both **root**:

```
Login:     root
Password:  root
```

Telnet Console

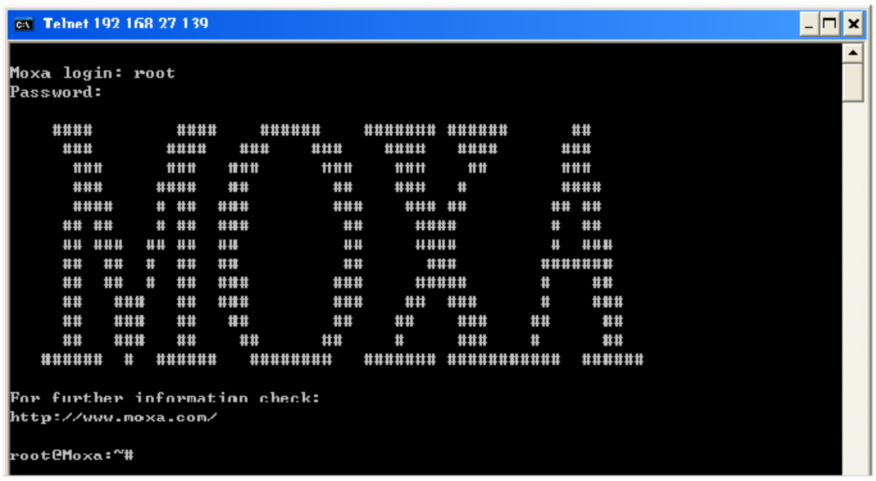
If you know at least one of the two IP addresses and netmasks, then you can use Telnet to connect to the UC-7400-LX Plus's console utility. The default IP address and Netmask for each port are given below:

	Default IP Address	Netmask
LAN 1	192.168.3.127	255.255.255.0
LAN 2	192.168.4.127	255.255.255.0

Use a cross-over Ethernet cable to connect directly from your PC to the UC-7400-LX Plus. You should first modify your PC's IP address and netmask so that your PC is on the same subnet as one of the UC-7400-LX Plus LAN ports. For example, if you connect to LAN 1, you can set your PC's IP address to 192.168.3.126 and netmask to 255.255.255.0. If you connect to the LAN 2, you can set your PC's IP address to 192.168.4.126 and netmask to 255.255.255.0.

To connect to your local LAN with a hub or switch, 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 login: root
Password:

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

For further information check:
http://www.moxa.com/

root@Moxa:~#
```

You can proceed with configuring network settings of the target computer when you reach the bash command shell. Configuration instructions are given in the next section.

ATTENTION

Serial Console Reminder

Remember to choose VT100 as the terminal type. Use the cable CBL-RJ45F9-150, which comes with the UC-7400-LX Plus, to connect to the serial console port.

Telnet Reminder

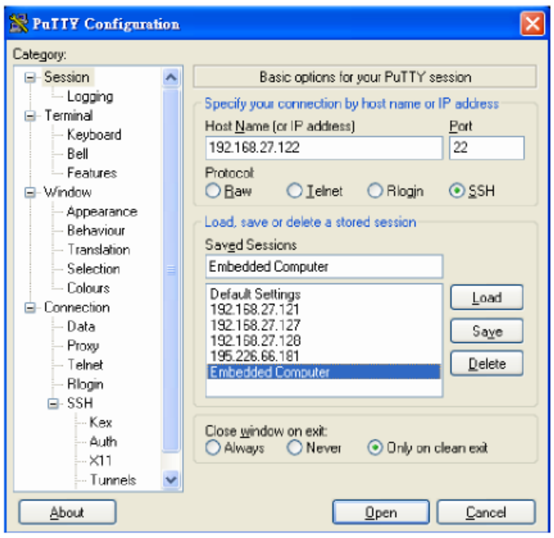
When connecting to the UC-7400-LX Plus over a LAN, you must configure your PC's Ethernet IP address to be on the same subnet as the UC-7400-LX Plus that you wish to contact. If you do not get connected on the first try, re-check the serial and IP settings, and then unplug and re-plug the UC-7400-LX Plus power cord.

SSH Console

UC-7400-LX Plus supports an SSH console to provide users with better security options.

Windows Users

Click on the link http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html to download PuTTY (free software) to set up an SSH console for the UC-7400-LX Plus in a Windows environment. The following figure shows a simple example of the configuration that is required.



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, use the "ssh" command to access the UC-7400-LX Plus Console utility through an SSH connection.

```
#ssh 192.168.3.127
```

Select **yes** to complete the connection.

```
[root@bee_notebook root]# ssh 192.168.3.127
The authenticity of host '192.168.3.127 (192.168.3.127)' 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_
```

NOTE	SSH provides better security compared to Telnet for accessing the UC-7400-LX Plus Console utility over the network.

Configuring the Ethernet Interface

The network settings of the UC-7400-LX Plus can be modified from 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 the network settings of the target computer.

1. 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 directories.

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

2. Type **#vi interfaces** to edit the network configuration file with vi editor. You can configure the Ethernet ports of the UC-7400-LX Plus 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 192.168.3.127 for LAN1 and 192.168.4.127 for LAN2, with default netmask of 255.255.255.0.

```
# We always want the loopback interface.

auto eth0 eth1 lo
iface lo inet loopback

# embedded ethernet LAN1
iface eth0 inet static
       address 192.168.3.127
       network 192.168.3.0
       netmask 255.255.255.0
       broadcast 192.168.3.255

# embedded ethernet LAN2
iface eth1 inet static
       address 192.168.4.127
       network 192.168.4.0
       netmask 255.255.255.0
       broadcast 192.168.4.255
```

Dynamic IP addresses:

By default, the UC-7400-LX Plus is configured for "static" IP addresses. To configure one or both 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 LAN1	Dynamic Setting using DHCP
iface eth0 inet **static** address 192.168.3.127 network: 192.168.3.0 netmask 255.255.255.0 broadcast 192.168.3.255	iface eth0 inet **dhcp**

```
Auto eth0 eth1 lo
iface lo inet loopback

iface eth0 inet dhcp

iface eth1 inet dhcp
```

3. 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

NOTE After changing the IP settings, use the **networking restart** command to activate the new IP address. However, the LCM display will still show the old IP address. To update the LCM display, you will need to reboot the UC-7400-LX Plus.

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/#
```

Using IPv6 Command Utilities

Moxa provides several command utilities for IPv6, starting with firmware version v1.6. Here are some examples to demonstrate how to use these commands.

Loading IPv6 Module

Before using these commands, you need to use the following command to load the ipv6 module:

# modprobe ipv6

After module is loaded, use the **lsmod** to check that the module shows up in the module list.

```
root@Moxa:~# modprobe ipv6
root@Moxa:~#
root@Moxa:~# lsmod

ipv6 223420 6 - Live 0xbf0f7000
pcmcia 12068 4 - Live 0xbf0f3000
yenta_socket 11200 0 - Live 0xbf0ef000
pcmcia_core 31820 2 pcmcia,yenta_socket, Live 0xbf0e6000
sierra 864 0 - Live 0xbf0e4000
usbserial 19240 1 sierra, Live 0xbf0de000
ohci_hcd 14312 0 - Live 0xbf0d9000
noz 14128 0 - Live 0xbf0d4000
8139too 18404 0 - Live 0xbf0ce000
usbcore 76124 4 sierra,usbserial,ohci_hcd, Live 0xbf0ba000
ixp400_eth 18968 2 - Live 0xbf0b4000
ixp400 709672 1 ixp400_eth, Live 0xbf005000
moxa_da66x_misc 14456 0 - Live 0xbf000000
root@Moxa:~#
```

Setting up an IPv6 address

Use **ifconfig** to configure the network interface. Type **ifconfig –h** to see how to use this command.

```
root@Moxa:~# ifconfig –h
BusyBox v1.14.1 (2009-10-29 18:42:35 CST) multi-call binary
Usage: ifconfig [-a] interface [address]
Configure a network interface
Options:
            [add ADDRESS[/PREFIXLEN]]
            [del ADDRESS[/PREFIXLEN]]
            [[-]broadcast [ADDRESS]] [[-]pointopoint [ADDRESS]]
            [netmask ADDRESS] [dstaddr ADDRESS]
            [hw ether|infiniband ADDRESS] [metric NN] [mtu NN]
            [[-]trailers] [[-]arp] [[-]allmulti]
            [multicast] [[-]promisc] [txqueuelen NN] [[-]dynamic]
            [up|down] ...
```

If would like to configure a link-local address, such as **fe80::217:9aff:febb:48ac**, you can use the the following syntax to configure:

# ifconfig eth0 add fe80::217:9aff:febb:48ac/64

Detecting an IPv6-based IP Address

Use the **ping6** command to detect a specific IPv6 address on the network. Type **ping6 -h** to see how to use this command:

```
root@Moxa:~# ping6 –h
BusyBox v1.14.1 (2010-10-29 18:42:35 CST) multi-call binary

Usage: ping6 [OPTION]... host

Send ICMP ECHO_REQUEST packets to network hosts

Options:
        -c CNT          Send only CNT pings
        -s SIZE         Send SIZE data bytes in packets (default=56)
        -I iface/IP     Use interface or IP address as source
        -q              Quiet, only displays output at start
                        and when finished
```

Assuming there is a device with IPv6 address **fe80::217:9aff:febb:489c** on the network, issue the following command to get a response from the device:

# ping6 fe80::217:9aff:febb:489c

Configuring the WLAN through the PCMCIA Interface

The following IEEE802.11g wireless modules are supported:

- ASUS—WL-107g
- CNET—CWC-854 (181D version)
- Edmiax—EW-7108PCg
- Amigo—AWP-914W
- GigaByte—GN-WMKG
- Other brands that use the Ralink RT2500 series chip set

To configure the WLAN for IEEE802.11g:

1. First unplug the CardBus wireless LAN card.
2. Use the command **#vi /etc/networking/interfaces** to open the "interfaces" configuration file with vi editor, and then edit the 802.11g network settings (the wireless interface name should be "eth2" on the UC-7400 Plus).

```
# We always want the loopback interface.

auto eth0 eth1 eth2 lo
iface lo inet loopback

# embedded ethernet LAN1
iface eth0 inet static
       address 192.168.3.127
       network 192.168.3.0
       netmask 255.255.255.0
       broadcast 192.168.3.255

# embedded ethernet LAN2
iface eth1 inet static
       address 192.168.4.127
       network 192.168.4.0
       netmask 255.255.255.0
       broadcast 192.168.4.255

# embedded ethernet LAN3
iface eth2 inet static
       address 192.168.5.127
       network 192.168.5.0
       netmask 255.255.255.0
       broadcast 192.168.5.255
```

3. Additional WLAN parameters are contained in the file **RT2500STA.dat**. To open the file, navigate to the RT2500STA folder and invoke vi, or type the command **#vi /etc/Wireless/RT2500STA/RT2500STA.dat** to edit the file with vi editor. Settings options for the various parameters are listed below the figure.

```
[Default]
CountryRegion=0
WirelessMode=0
SSID=MOXASYS
NetworkType=Infra
Channel=0
AuthMode=OPEN
EncrypType=WEP
DefaultKeyID=1
Key1Str=1111111111
Key2Str=
Key3
Key4
WpaPsk=abcdefghijklmnopqrstuvwxyz
TXBurst=0
TurboRate=0
BGProtection=0
ShortSlot=0
Key3
Key4
WpaPsk=abcdefghijklmnopqrstuvwxyz
```

CountryRegion—sets the channels for your particular country / region

Setting	Explanation
0	use channels 1 to 11
1	use channels 1 to 11
2	use channels 1 to 13
3	use channels 10, 11
4	use channels 10 to 13
5	use channel 14
6	use channels 1 to 14
7	use channels 3 to 9

WirelessMode—sets the wireless mode

Setting	Explanation
0	11b/g mixed
1	11b only
2	11g only

SSID—sets the softAP SSID

Setting
Any 32-byte string

NetworkType—sets the wireless operation mode

Setting	Explanation
Infra	Infrastructure mode (uses access points to transmit data)
Adhoc	Adhoc mode (transmits data from host to host)

Channel—sets the channel

Setting	Explanation
0	auto
1 to 14	the channel you want to use

AuthMode—sets the authentication mode

Setting
OPEN
SHARED
WPAPSK
WPANONE

EncrypType—Sets encryption type

Setting
NONE
WEP
TKIP
AES

DefaultKeyID—sets default key ID

Setting
1 to 4

Key1Str, Key2Str, Key3Str, Key4Str—sets strings Key1 to Key4

Setting
The keys can be input as 5 ascii characters, 10 hex numbers, 13 ascii characters, or 26 hex numbers

TxBurst—WPA pre-shared key

Setting
8 to 64 ASCII characters

WPAPSK—enables or disables TxBurst

Setting
8 to 63 ASCII or 64 HEX characters

TurboRate—enables or disables TurboRate

Setting	Explanation
0	disable
1	enable

BGProtection—sets 11b/11g protection (this function is for engineering testing only)

Setting	Explanation
0	auto
1	always on
2	always off

ShortSlot—enables or disables the short slot time

Setting	Explanation
0	disable
1	enable

TxRate—sets the TxRate

Setting	**Explanation**
0	Auto
1	1 Mbps
2	2 Mbps
3	5.5 Mbps
4	11 Mbps
5	6 Mbps
6	9 Mbps
7	12 Mbps
8	18 Mbps
9	24 Mbps
10	36 Mbps
11	48 Mbps
12	54 Mbps

RTSThreshold—sets the RTS threshold

Setting
1 to 2347

FragThreshold—sets the fragment threshold

Setting
256 to 2346

Example 1: Configure wireless LAN to link to AP that is OPEN/NONE (Authentication/Encryption)

```
[Default]
CountryRegion=0
WirelessMode=0
SSID=DN_3Com
NetworkType=Infra
Channel=0
AuthMode=OPEN
EncrypType=NONE
DefaultKeyID=1
Key1Str=0123456789
Key2Str=
Key3Str=
Key4Str=
WPAPSK=1111111111
TXBurst=0
TurboRate=0
BGProtection=0
ShortSlot=0
TxRate=0
RTSThreshold=2312
FragThreshold=2312
PSMode=CAM
```

Example 2: Configure wireless LAN to link to AP that is SHARED/WEP (Authentication/Encryption)

```
[Default]
CountryRegion=0
WirelessMode=0
SSID=DN_3Com
NetworkType=Infra
Channel=0
AuthMode=SHARED
EncrypType=WEP
DefaultKeyID=1
Key1Str=0123456789
Key2Str=
Key3Str=
Key4Str=
WPAPSK=1111111111
TXBurst=0
TurboRate=0
BGProtection=0
ShortSlot=0
TxRate=0
RTSThreshold=2312
FragThreshold=2312
PSMode=CAM
```

Example 3: Configure wireless LAN to link to AP that is WPAPSK/TKIP (Authentication/Encryption)

```
[Default]
CountryRegion=0
WirelessMode=0
SSID=DN_3Com
NetworkType=Infra
Channel=0
AuthMode=WPAPSK
EncrypType=TKIP
DefaultKeyID=1
Key1Str=0123456789
Key2Str=
Key3Str=
Key4Str=
WPAPSK=1111111111
TXBurst=0
TurboRate=0
BGProtection=0
ShortSlot=0
TxRate=0
RTSThreshold=2312
FragThreshold=2312
PSMode=CAM
```

Example 4: Configure wireless LAN to link to AP that is WPAPSK/AES (Authentication/Encryption)

```
[Default]
CountryRegion=0
WirelessMode=0
SSID=DN_3Com
NetworkType=Infra
Channel=0
AuthMode=WPAPSK
EncrypType=AES
DefaultKeyID=1
Key1Str=0123456789
Key2Str=
Key3Str=
Key4Str=
WPAPSK=1111111111
TXBurst=0
TurboRate=0
BGProtection=0
ShortSlot=0
TxRate=0
RTSThreshold=2312
FragThreshold=2312
PSMode=CAM
```

Connecting to 3G Networks through the PCMCIA Interface

For select models, the UC-7400-LX Plus equips a PCMCIA cardbus to support 3G datacards. There are several kinds of 3G datacards available for connecting to the 3G network. The UC-7400-LX Plus has built-in drivers to support the following brands.

- Vodafone Mobile Connect HSDPA/UMTS/EDGE datacard.
- Swisscom Unlimited 5-in-1.

These 3G modems in Linux are identified as device nodes at /dev. Vodafone Mobile Connect HSDPA/UMTS/EDGE datacard, Swisscom Unlimited 5-in-1 are controlled through /dev/noz0. By configuring the pppd dial-out script, the UC-7400-LX Plus can connect to the 3G network.

Plug the 3G Card into the Cardbus Slot

The UC-7400-LX Plus supports plug-and-play feature for the supported 3G datacards. When the 3G datacard is inserted, the UC-7400-LX Plus will detect the card automatically. You may use the command “dmesg” to see the kernel ring buffer message. The kernel ring buffer message will show the following message if the UC-7400-LX Plus detects the 3G card.

```
<6>cs: cb_alloc(bus 1): vendor 0x1931, device 0x000c

<4>PCI: enabling device 01:00.0 (0000 -> 0002)

<6>[1507] nozomi_card_init(): Init, cards_found: 1

<6>[1392] nozomi_get_card_type(): Card type is: 2048

<6>Nozomi driver nozomi_tty<6>[805] nozomi_read_config_table(): Version of card: 1

<6>[797] nozomi_read_config_table(): Initialization OK!
```

Note: This example is the message detected for the Vodafone Mobile Connect HSDPA/UMTS/EDGE datacard. For different cards, these messages will be slightly different.

If you do not see these messages, it means that the UC-7400-LX Plus is unable to detect your 3G datacard.

NOTE When the datacard is plugged into the card bus slot, your SIM card will ask for the PIN code. You can use minicom or Hyper terminal on a host computer to disable it. Enter the AT+CLCK command.

AT+CLCK="SC",0,"XXXX"

(replacing xxxx with your own PIN) to turn off the PIN function. This way, your UMTS card will start searching for a network immediately. To turn the PIN function on again, type

AT+CLCK="SC",1,"XXXX"

To check the status of the PIN function, type

AT+CLCK="SC",2

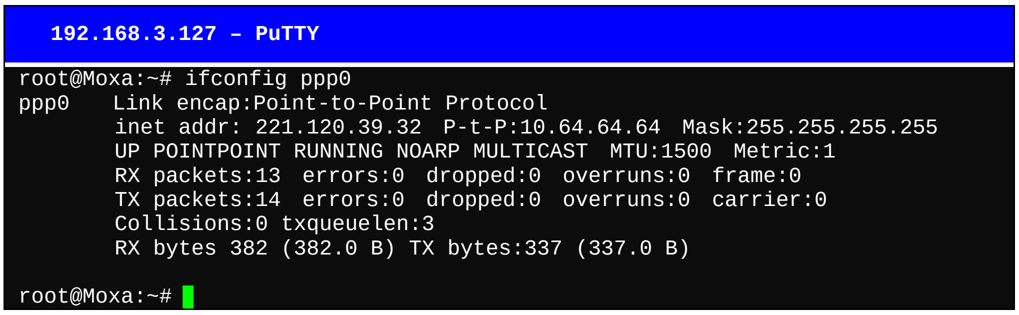
"+CLCK: 0" means the PIN is off; "+CLCK: 1"means the PIN is on.

Connecting to the 3G Network

You can connect to 3G networks over a ppp connection.

#/etc/init.d/3g.sh nozomi start

After the connection is established, you can type ifconfig to check the ppp0 interface.


```
192.168.3.127 - PuTTY

root@Moxa:~# ifconfig ppp0
ppp0     Link encap:Point-to-Point Protocol
         inet addr: 221.120.39.32  P-t-P:10.64.64.64  Mask:255.255.255.255
         UP POINTPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
         RX packets:13  errors:0  dropped:0  overruns:0  frame:0
         TX packets:14  errors:0  dropped:0  overruns:0  carrier:0
         Collisions:0 txqueuelen:3
         RX bytes 382 (382.0 B) TX bytes:337 (337.0 B)

root@Moxa:~#
```

The IP address of the ppp0 interface is 221.120.39.32, and it connects to a remote gateway with IP address 10.64.64.54.

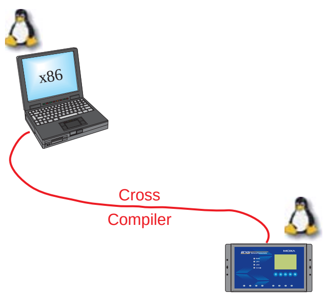
NOTE The dial-out script uses /etc/ppp/ppp-umts.chat to chat with a 3G card. You would need to modify this file to change the advanced configuration. The default SVN server in this chat script is set as "internet". This value varies between network service providers. Be sure to use the right SVN server to connect the card.

Use the following command to stop 3G connection.

#/etc/init.d/3g.sh nozomi stop

Test Program—Developing Hello.c

In this section, we use the standard "Hello" programming example to illustrate how to develop a program for the UC-7400-LX Plus. In general, program development involves the following seven steps.



Step 1:
Connect the UC-7400-LX Plus to a Linux PC.
Step 2:
Install Tool Chain (GNU Cross Compiler & glibc).
Step 3:
Set the cross compiler and glibc environment variables.
Step 4:
Prepare the code and compile the program.
Step 5:
Download the program to the UC-7400-LX Plus through FTP or NFS.
Step 6:
Debug the program
→ If bugs are found, return to Step 4.
→ If no bugs are found, continue with Step 7.
Step 7:
Back up the user directory (distribute the program to additional UC-7400-LX Plus units if needed).

Installing the Tool Chain (Linux)

The PC must have the Linux Operating System pre-installed before installing the UC-7400-LX Plus GNU Tool Chain. Fedora core, and compatible versions are recommended. The Tool Chain requires about 100 MB of hard disk space on your PC. The UC-7400-LX-Plus Tool Chain software is located on the UC-7400-LX Plus 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/Linux/xscale_be-x.x.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/xscale_be/bin:$PATH
```

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

Checking the Flash Memory Space

The UC-7400-LX Plus uses a specially designed root file system. Only the **/tmp, /etc, /home,** and **/root** directories are writable. Others are read-only. The writable directories are mounted on **/dev/mtdblock3**. If the **/dev/mtdblock3** 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/mtdblock2        14.0M    10.9M      3.1M  78% /
/dev/ram15             1.7M    18.0k      1.6M   1% /dev
/dev/ram0            499.0k    29.0k    445.0k   6% /var
/dev/mtdblock3        15.8M     2.4M     13.3M  16% /tmp
/dev/mtdblock3        15.8M     2.4M     13.3M  16% /home
/dev/mtdblock3        15.8M     2.4M     13.3M  16% /etc
tmpfs                61.9M        0     61.9M   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, use the console cable to connect your PC to the UC-7400-LX Plus, and then use the console utility to delete the files from the UC-7400-LX Plus flash memory.

Compiling Hello.c

The CD included with the product contains several example programs. Here we use **Hello.c** as an example 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_be-gcc –o hello-release hello.c
xscale_be-strip –s hello-release
xscale_be-gcc –ggdb -o hello-debug hello.c
[root@localhost hello]# _
```

Next, execute **make** 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-7400-LX Plus)

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

NOTE Be sure to type the **#make** command from within the **/tmp/example/hello** directory, since UC's tool chain puts a specially designed **Makefile** in that directory. This special Makefile uses the mxscale-gcc compiler to compile the hello.c source code for the Xscale environment. If you type the **#make** command from within 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-7400-LX Plus via FTP.

1. From the PC, type:

```
#ftp 192.168.3.127
```

2. Use the bin command to set the transfer mode to Binary mode, and then use the put command to initiate the file transfer:

```
ftp> bin
ftp> put hello-release
```

3. From the UC-7400-LX Plus, type:

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

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

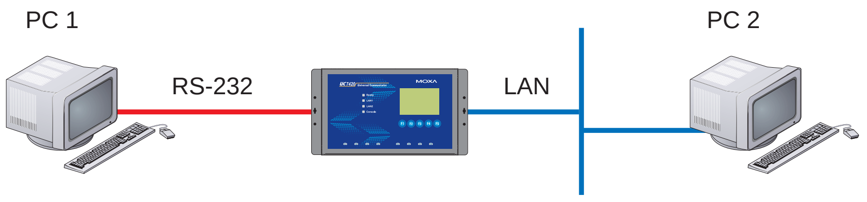
```
root@Moxa:~# ./hello-release
Hello
```

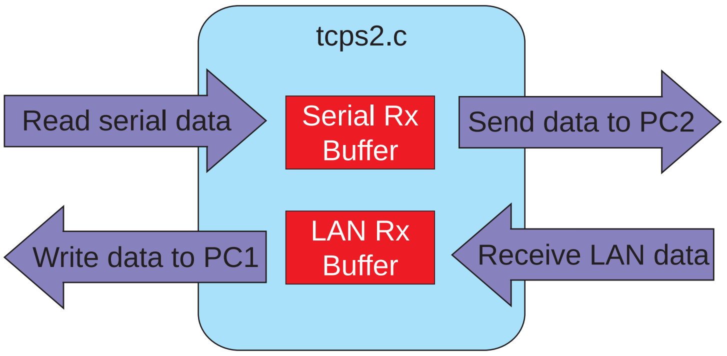
Developing Your First Application

We use the tcps2 example to illustrate how to build an application. The procedure outlined in the following subsections will show you how to build a TCP server program with serial port communication that runs on the UC-7400-LX Plus.

Testing Environment

The tcps2 example demonstrates a simple application program that delivers transparent, bi-directional data transmission between the UC-7400-LX Plus serial and Ethernet ports. As illustrated in the following figure, the purpose of this application is to transfer data between PC 1 and the UC-7400-LX Plus through an RS-232 connection. At the remote site, data can be transferred between the UC-7400-LX Plus's Ethernet port and PC 2 over an Ethernet connection.





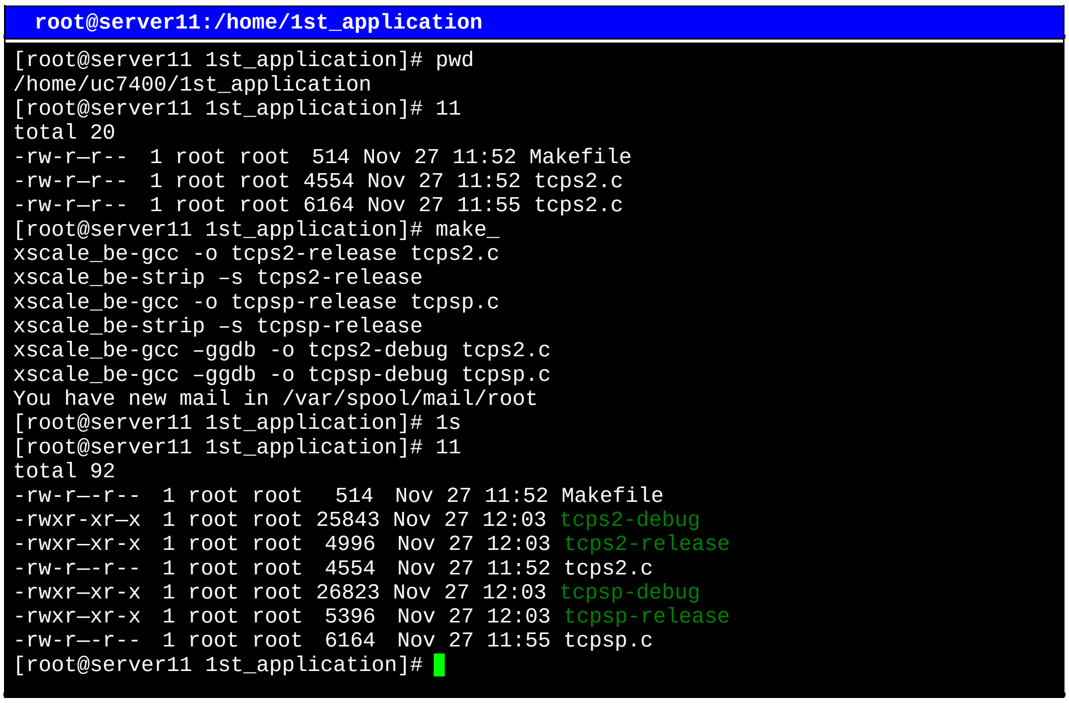
Compiling tcps2.c

The source code for the tcps2 example is located on the CD-ROM at **CD-ROM://example/TCPServer2/tcps2.c**. Use the following commands to copy the file to a specific directory on your PC. We use the direrctory **/home/1st_application/**. Note that you need to copy 3 files—**Makefile, tcps2.c, tcpsp.c**—from the CD-ROM to the target directory.

```
#mount –t iso9660 /dev/cdrom /mnt/cdrom
#cp /mnt/cdrom/example/TCPServer2/tcps2.c /home/1st_application/tcps2.c
#cp /mnt/cdrom/example/TCPServer2/tcpsp.c /home/1st_application/tcpsp.c
#cp /mnt/cdrom/example/TCPServer2/Makefile.c /home/1st_application/Makefile.c
```

Type **#make** to compile the example code:

You will see the following response, indicating that the example program was compiled successfully.



```
root@server11:/home/1st_application
[root@server11 1st_application]# pwd
/home/uc7400/1st_application
[root@server11 1st_application]# 11
total 20
-rw-r–r--  1 root root  514 Nov 27 11:52 Makefile
-rw-r–r--  1 root root 4554 Nov 27 11:52 tcps2.c
-rw-r–r--  1 root root 6164 Nov 27 11:55 tcps2.c
[root@server11 1st_application]# make_
xscale_be-gcc -o tcps2-release tcps2.c
xscale_be-strip –s tcps2-release
xscale_be-gcc -o tcpsp-release tcpsp.c
xscale_be-strip –s tcpsp-release
xscale_be-gcc –ggdb -o tcps2-debug tcps2.c
xscale_be-gcc –ggdb -o tcpsp-debug tcpsp.c
You have new mail in /var/spool/mail/root
[root@server11 1st_application]# 1s
[root@server11 1st_application]# 11
total 92
-rw-r—-r--  1 root root   514  Nov 27 11:52 Makefile
-rwxr-xr–x  1 root root 25843 Nov 27 12:03 tcps2-debug
-rwxr–xr-x  1 root root  4996  Nov 27 12:03 tcps2-release
-rw-r—-r--  1 root root  4554  Nov 27 11:52 tcps2.c
-rwxr–xr-x  1 root root 26823 Nov 27 12:03 tcpsp-debug
-rwxr–xr-x  1 root root  5396  Nov 27 12:03 tcpsp-release
-rw-r—-r--  1 root root  6164  Nov 27 11:55 tcpsp.c
[root@server11 1st_application]# 
```

Two executable files, **tcps2-release** and **tcps2-debug**, are created.

tcps2-release—an IXP platform execution file (created specifically to run on the).

tcps2-debug—an IXP platform GDB debug server execution file (see Chapter 5 for details about the GDB debug tool).

NOTE If you get an error message at this point, it could be because you neglected to put tcps2.c and tcpsp.c in the same directory. The example Makefile we provide is set up to compile both tcps2 and tcpsp into the same project Makefile. Alternatively, you could modify the Makefile to suit your particular requirements.

Uploading and Running the "tcps2-release" Program

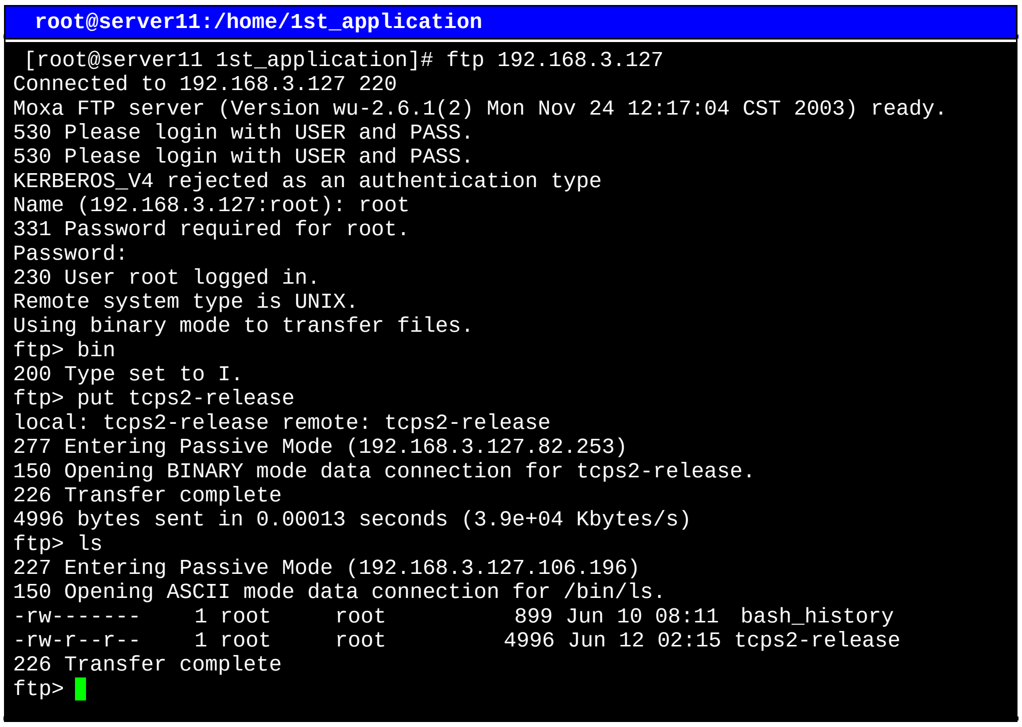
Use the following commands to use FTP to upload **tcps2-release** to the UC-7400-LX Plus.

1. From the PC, type:

```
#ftp 192.168.3.127
```

2. Next, use the **bin** command to set the transfer mode to **Binary,** and the **put** command to initiate the file transfer:

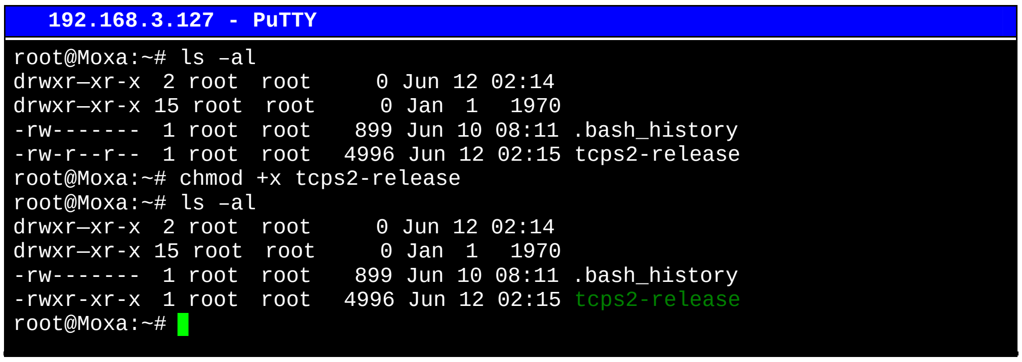
```
ftp> bin
ftp> put tcps2-release
```



```
root@server11:/home/1st_application
 [root@server11 1st_application]# ftp 192.168.3.127
Connected to 192.168.3.127 220
Moxa FTP server (Version wu-2.6.1(2) Mon Nov 24 12:17:04 CST 2003) ready.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.3.127: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> bin
200 Type set to I.
ftp> put tcps2-release
local: tcps2-release remote: tcps2-release
277 Entering Passive Mode (192.168.3.127.82.253)
150 Opening BINARY mode data connection for tcps2-release.
226 Transfer complete
4996 bytes sent in 0.00013 seconds (3.9e+04 Kbytes/s)
ftp> ls
227 Entering Passive Mode (192.168.3.127.106.196)
150 Opening ASCII mode data connection for /bin/ls.
-rw-------     1 root     root          899 Jun 10 08:11  bash_history
-rw-r--r--     1 root     root         4996 Jun 12 02:15 tcps2-release
226 Transfer complete
ftp>
```

3. From the UC-7400-LX Plus, type:

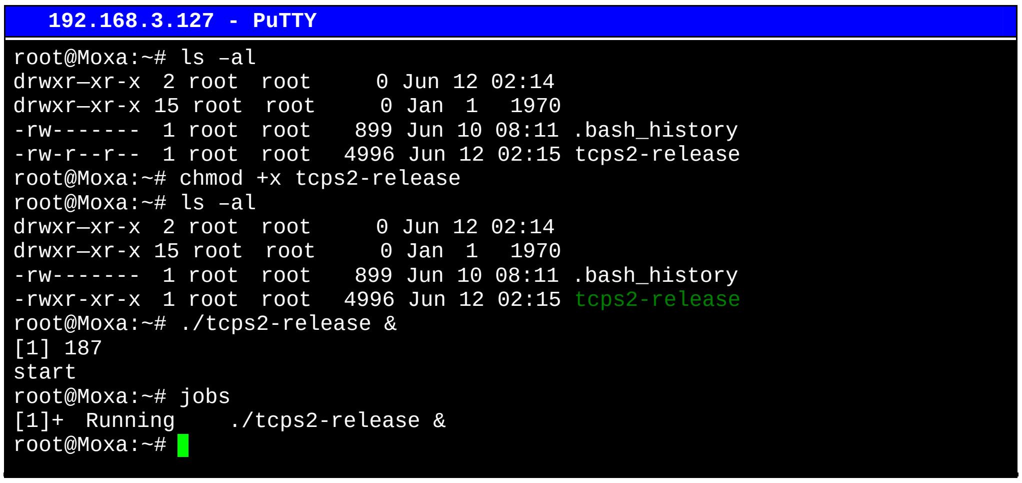
```
# chmod +x tcps2-release
# ./tcps2-release &
```



```
192.168.3.127 - PuTTY
root@Moxa:~# ls -al
drwxr-xr-x  2 root  root     0 Jun 12 02:14
drwxr-xr-x 15 root  root     0 Jan  1   1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rw-r--r--  1 root  root  4996 Jun 12 02:15 tcps2-release
root@Moxa:~# chmod +x tcps2-release
root@Moxa:~# ls -al
drwxr-xr-x  2 root  root     0 Jun 12 02:14
drwxr-xr-x 15 root  root     0 Jan  1   1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rwxr-xr-x  1 root  root  4996 Jun 12 02:15 tcps2-release
root@Moxa:~#
```

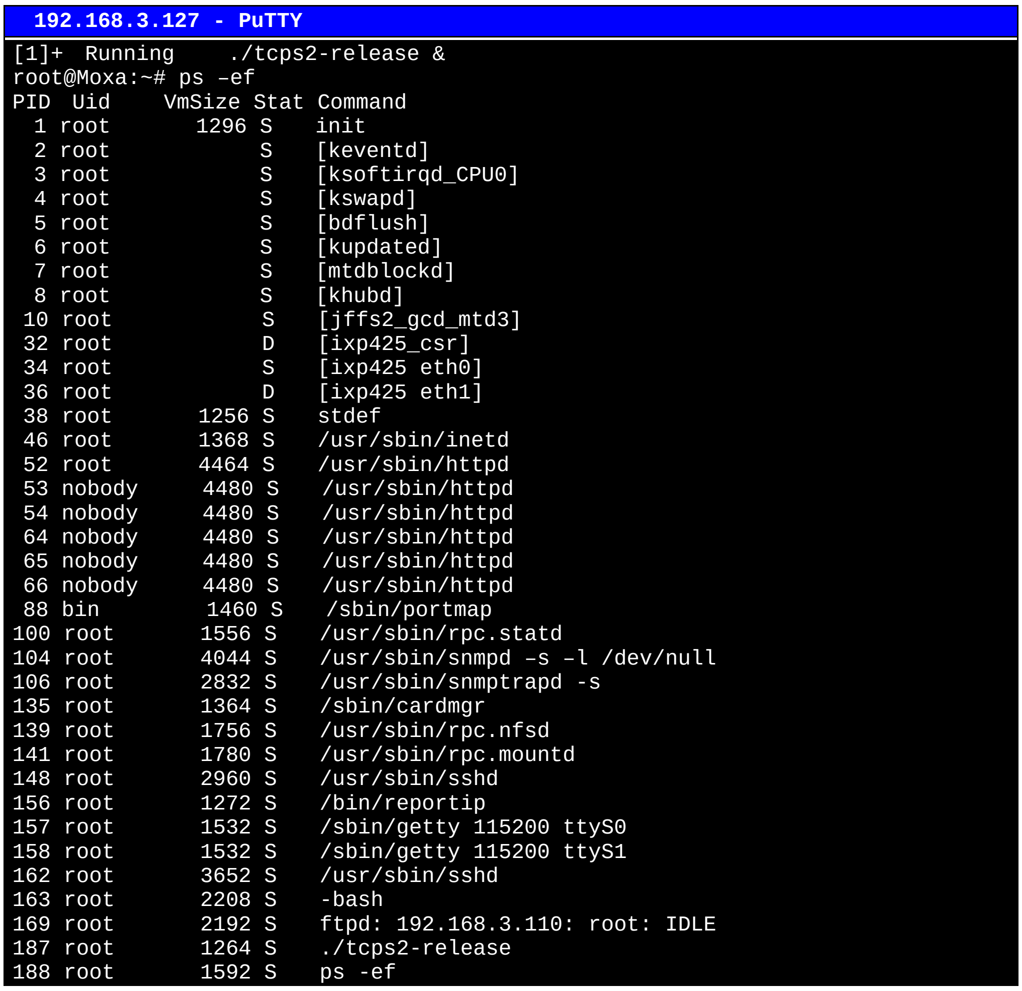
4. The program should start running in the background. Use either the **`#jobs`** or **`#ps -ef`** command to check if the tcps2 program is actually running in the background.

   `#jobs // use this command to check if the program is running`



```
192.168.3.127 - PuTTY
root@Moxa:~# ls -al
drwxr-xr-x  2 root  root     0 Jun 12 02:14
drwxr-xr-x 15 root  root     0 Jan  1  1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rw-r--r--  1 root  root  4996 Jun 12 02:15 tcps2-release
root@Moxa:~# chmod +x tcps2-release
root@Moxa:~# ls -al
drwxr-xr-x  2 root  root     0 Jun 12 02:14
drwxr-xr-x 15 root  root     0 Jan  1  1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rwxr-xr-x  1 root  root  4996 Jun 12 02:15 tcps2-release
root@Moxa:~# ./tcps2-release &
[1] 187
start
root@Moxa:~# jobs
[1]+  Running     ./tcps2-release &
root@Moxa:~#
```

   `#ps -ef // use this command to check if the program is running`



```
192.168.3.127 - PuTTY
[1]+  Running     ./tcps2-release &
root@Moxa:~# ps -ef
PID  Uid     VmSize Stat Command
  1 root       1296 S    init
  2 root            S    [keventd]
  3 root            S    [ksoftirqd_CPU0]
  4 root            S    [kswapd]
  5 root            S    [bdflush]
  6 root            S    [kupdated]
  7 root            S    [mtdblockd]
  8 root            S    [khubd]
 10 root            S    [jffs2_gcd_mtd3]
 32 root            D    [ixp425_csr]
 34 root            S    [ixp425 eth0]
 36 root            D    [ixp425 eth1]
 38 root       1256 S    stdef
 46 root       1368 S    /usr/sbin/inetd
 52 root       4464 S    /usr/sbin/httpd
 53 nobody     4480 S    /usr/sbin/httpd
 54 nobody     4480 S    /usr/sbin/httpd
 64 nobody     4480 S    /usr/sbin/httpd
 65 nobody     4480 S    /usr/sbin/httpd
 66 nobody     4480 S    /usr/sbin/httpd
 88 bin        1460 S    /sbin/portmap
100 root       1556 S    /usr/sbin/rpc.statd
104 root       4044 S    /usr/sbin/snmpd -s -l /dev/null
106 root       2832 S    /usr/sbin/snmptrapd -s
135 root       1364 S    /sbin/cardmgr
139 root       1756 S    /usr/sbin/rpc.nfsd
141 root       1780 S    /usr/sbin/rpc.mountd
148 root       2960 S    /usr/sbin/sshd
156 root       1272 S    /bin/reportip
157 root       1532 S    /sbin/getty 115200 ttyS0
158 root       1532 S    /sbin/getty 115200 ttyS1
162 root       3652 S    /usr/sbin/sshd
163 root       2208 S    -bash
169 root       2192 S    ftpd: 192.168.3.110: root: IDLE
187 root       1264 S    ./tcps2-release
188 root       1592 S    ps -ef
```

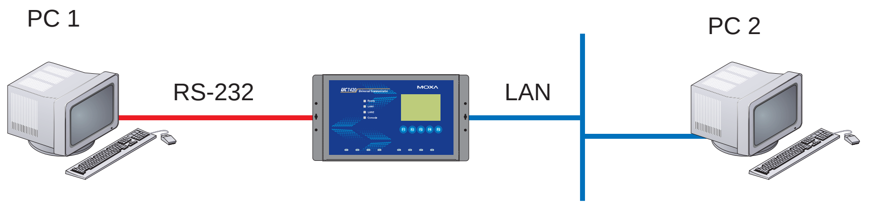
```
root@Moxa:~#
```

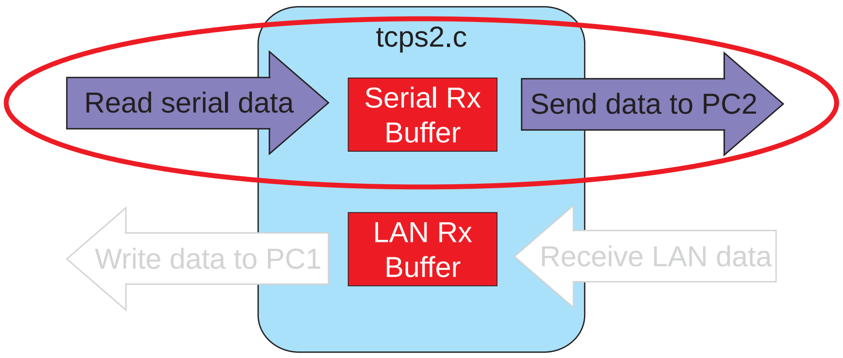
NOTE Use the **kill -9** command for PID 187 to terminate this program: **#kill -9 187**

Testing Procedure Summary

1. Compile **tcps2.c** (`#make`).
2. Upload and run **tcps2-release** in the background **(`#./tcps2-release &`)**.
3. Check that the process is running **(`#jobs or #ps -ef`)**.
4. Use a serial cable to connect PC1 to the UC-7400-LX Plus serial port 1.
5. Use an Ethernet cable to connect PC2 to the UC-7400-LX Plus.
6. On PC1: If running Windows, use HyperTerminal (**11520, n, 8, 1**) to open COMn.
7. On PC2: Type **`#telnet 192.168.3.127 4001`**.
8. On PC1: Type some text on the keyboard and then press **Enter**.
9. On PC2: The text you typed on PC1 will appear on PC2's screen.

The testing environment is illustrated in the following figure. However, note that there are limitations to the example program **tcps2.c**.





NOTE The **tcps2.c** application is a simple example designed to give users a basic understanding of the concepts involved in combining Ethernet communication and serial port communication. However, the example program has some limitations that make it unsuitable for real-life applications.

1. The serial port is in canonical mode and block mode, making it impossible to send data from the Ethernet side to the serial side (i.e., from PC 2 to PC 1 in the above example).
2. The Ethernet side will not accept multiple connections.

3

Managing Embedded Linux

This chapter includes information about version control, deployment, updates, and peripherals. The information in this chapter will be particularly useful when you need to run the same application on several UC-7400-LX Plus units.

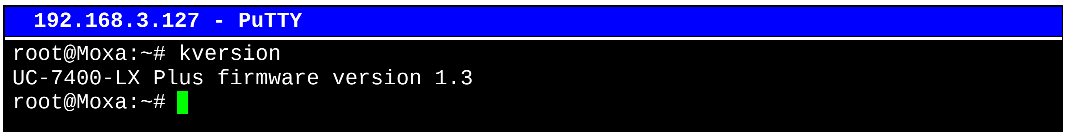
The following topics are covered in this chapter:

- **System Version Information**
- **System Image Backup**
  - Upgrading the Firmware
  - Loading Factory Defaults
- **Enabling and Disabling Daemons**
- **Starting a Program Automatically at Run-Level**
- **Adjusting the System Time**
  - Setting the Time Manually
  - Write current system time to RTC
  - NTP Client
  - Updating the Time Automatically
- **Cron—Daemon for Executing Scheduled Commands**
- **Connecting Peripherals**
  - USB Mass Storage
  - CF Mass Storage

System Version Information

To determine the hardware capability of your UC-7400-LX Plus, and what kind of software functions are supported, check the version numbers of your UC-7400-LX Plus firmware version. Contact Moxa to determine the hardware version. You will need the **Production S/N** (Serial number), which is located on the UC-7400-LX Plus bottom label.

To check the kernel version, type:
`#kversion`



```
root@Moxa:~# kversion
UC-7400-LX Plus firmware version 1.3
root@Moxa:~#
```

System Image Backup

Upgrading the Firmware

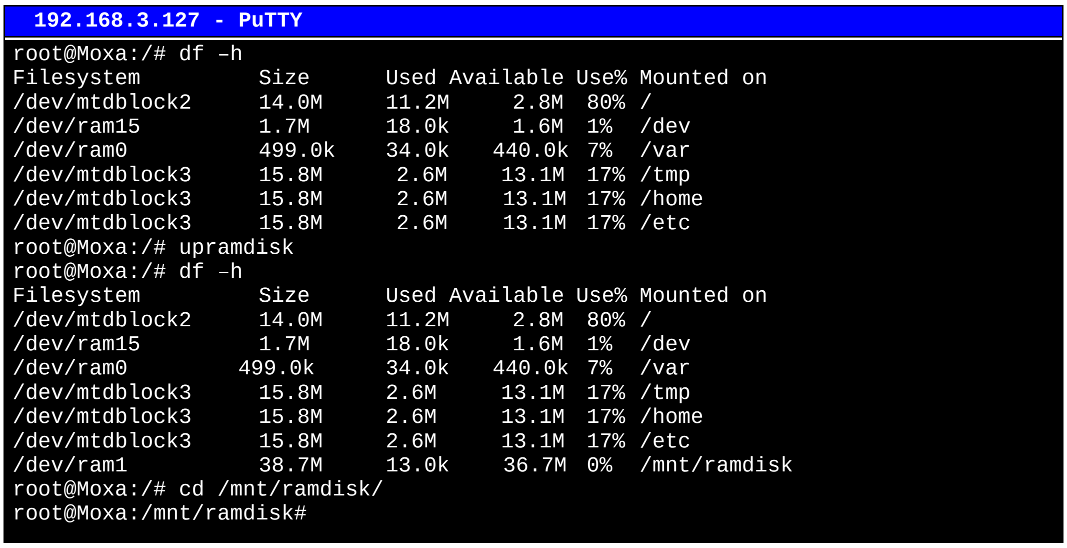
The UC-7400-LX Plus bios, kernel, and user file system are combined into one firmware file, which can be downloaded from Moxa’s website (www.moxa.com). The name of the file has the form **FWR_UC7400P_Va b_Build_YY MM DD HH**, with “a b” indicating the firmware version and YY MM DD HH indicating the build date and time. To upgrade the firmware, download the firmware file to a PC, and then transfer the file to the UC-7400-LX Plus unit through 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 is 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.



```
192.168.3.127 - PuTTY
root@Moxa:/# df –h
Filesystem          Size      Used Available Use% Mounted on
/dev/mtdblock2      14.0M     11.2M      2.8M  80% /
/dev/ram15          1.7M      18.0k      1.6M  1%  /dev
/dev/ram0           499.0k    34.0k    440.0k  7%  /var
/dev/mtdblock3      15.8M      2.6M     13.1M  17% /tmp
/dev/mtdblock3      15.8M      2.6M     13.1M  17% /home
/dev/mtdblock3      15.8M      2.6M     13.1M  17% /etc
root@Moxa:/# upramdisk
root@Moxa:/# df –h
Filesystem          Size      Used Available Use% Mounted on
/dev/mtdblock2      14.0M     11.2M      2.8M  80% /
/dev/ram15          1.7M      18.0k      1.6M  1%  /dev
/dev/ram0         499.0k      34.0k    440.0k  7%  /var
/dev/mtdblock3      15.8M     2.6M      13.1M  17% /tmp
/dev/mtdblock3      15.8M     2.6M      13.1M  17% /home
/dev/mtdblock3      15.8M     2.6M      13.1M  17% /etc
/dev/ram1           38.7M     13.0k     36.7M  0%  /mnt/ramdisk
root@Moxa:/# cd /mnt/ramdisk/
root@Moxa:/mnt/ramdisk#
```

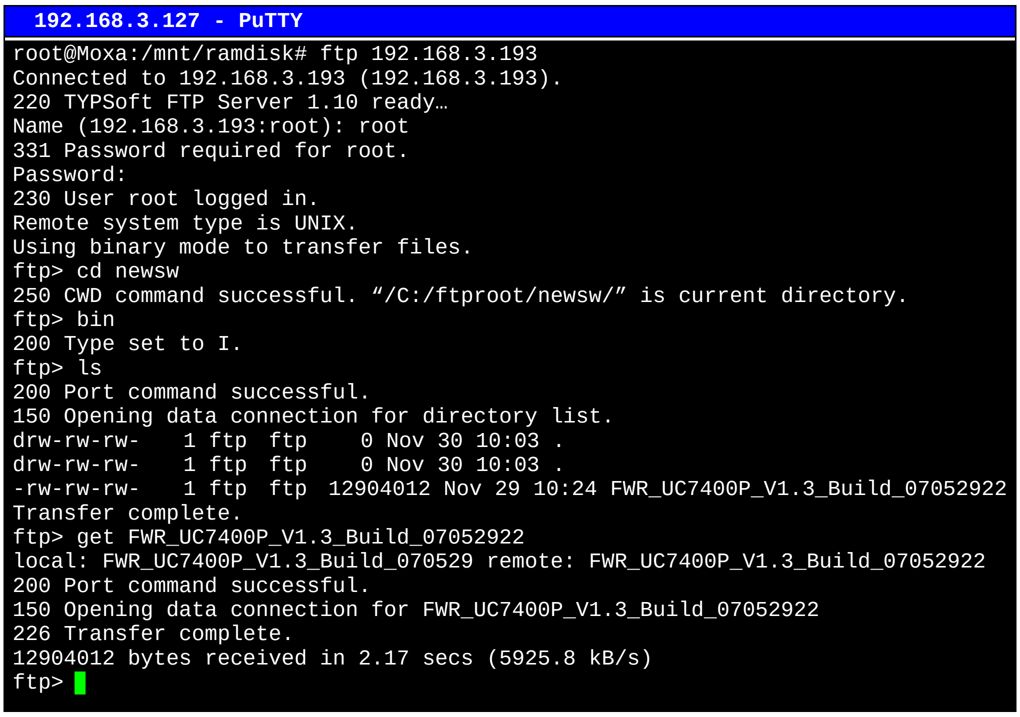
The following instructions give the steps required to save the firmware file to the UC-7400-LX Plus RAM disk, and then upgrade the firmware.

1. Type the following commands to enable the RAM disk:

```
#upramdisk
#cd /mnt/ramdisk
```

2. Type the following commands to use the UC-7400-LX Plus's built-in FTP client to transfer the firmware file (**FWR_UC7400P_Va b_Build_YY MM DD HH**) from the PC to the UC-7400-LX Plus:

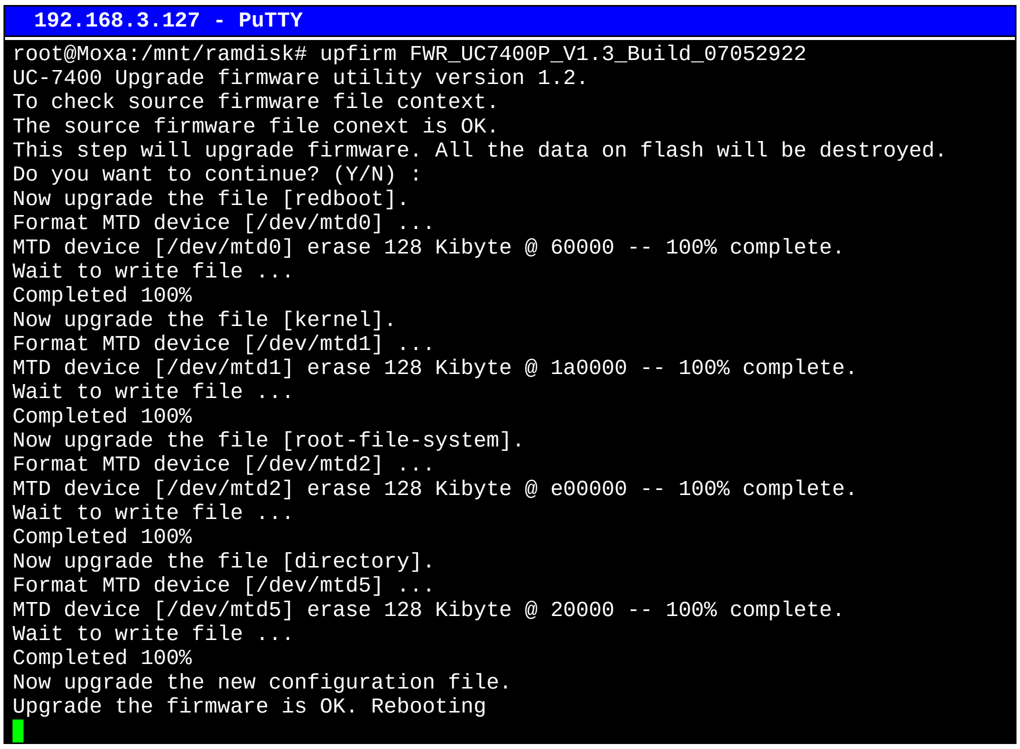
```
/mnt/ramdisk> ftp <destination PC's IP>
Login Name: xxxx
Login Password: xxxx
ftp> bin
ftp> get FWR UC7400P V1.3 Build 07052922
```



```
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_UC7400P_V1.3_Build_07052922
Transfer complete.
ftp> get FWR_UC7400P_V1.3_Build_07052922
local: FWR_UC7400P_V1.3_Build_070529 remote: FWR_UC7400P_V1.3_Build_07052922
200 Port command successful.
150 Opening data connection for FWR_UC7400P_V1.3_Build_07052922
226 Transfer complete.
12904012 bytes received in 2.17 secs (5925.8 kB/s)
ftp>
```

3. Next, use the **`upfirm`** command to upgrade the kernel and root file system:

   `#upfirm FWR_UC7400P_V1.3_Build_07052922`



```
192.168.3.127 - PuTTY
root@Moxa:/mnt/ramdisk# upfirm FWR_UC7400P_V1.3_Build_07052922
UC-7400 Upgrade firmware utility version 1.2.
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

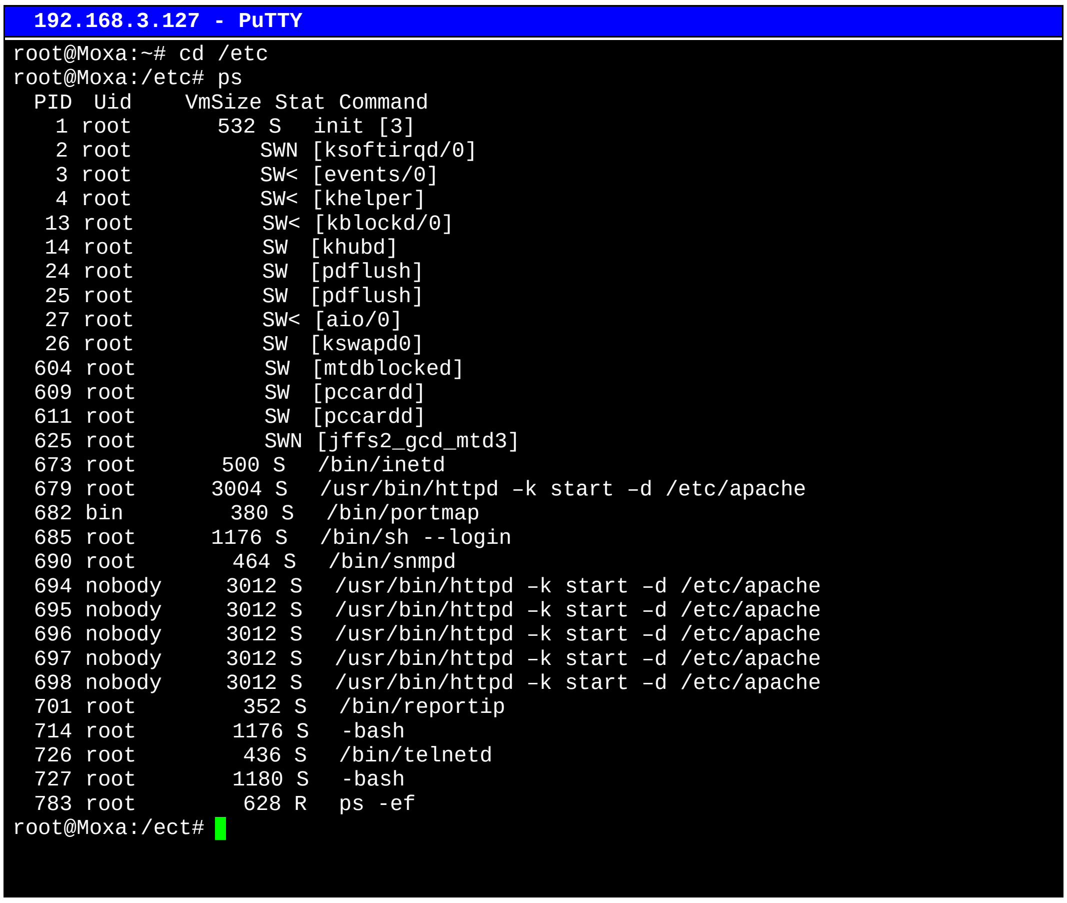
To load the system's factory default settings, press the reset-to-default button for at least 5 seconds. Doing so will destroy all of the files in the **/home** and **/etc** directories. While holding the button for the first 5 seconds, the ready LED will blink once each second. After holding the button continuously for more than 5 seconds, the ready LED will switch off, indicating that the factory defaults have been loaded.

Enabling and Disabling Daemons

Daemons are programs that run in the background. They are often used to provide services such as web access, FTP, and email service. The following daemons are enabled when the UC-7400-LX Plus boots up.

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

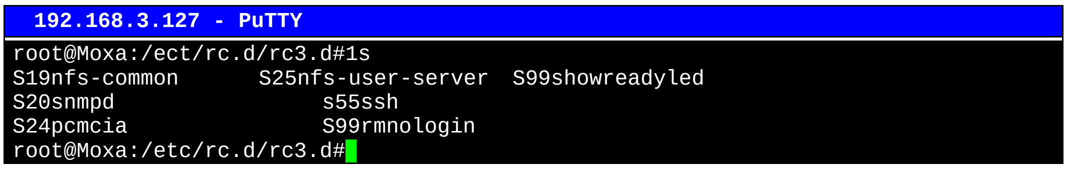
Type the command "**ps**" to list all processes currently running.



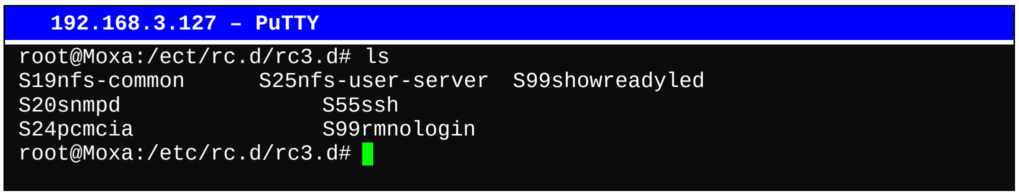
```
192.168.3.127 - PuTTY
root@Moxa:~# cd /etc
root@Moxa:/etc# ps
  PID  Uid     VmSize Stat Command
    1 root       532 S   init [3]
    2 root           SWN [ksoftirqd/0]
    3 root           SW< [events/0]
    4 root           SW< [khelper]
   13 root           SW< [kblockd/0]
   14 root           SW  [khubd]
   24 root           SW  [pdflush]
   25 root           SW  [pdflush]
   27 root           SW< [aio/0]
   26 root           SW  [kswapd0]
  604 root           SW  [mtdblocked]
  609 root           SW  [pccardd]
  611 root           SW  [pccardd]
  625 root           SWN [jffs2_gcd_mtd3]
  673 root       500 S   /bin/inetd
  679 root      3004 S   /usr/bin/httpd -k start -d /etc/apache
  682 bin        380 S   /bin/portmap
  685 root      1176 S   /bin/sh --login
  690 root       464 S   /bin/snmpd
  694 nobody    3012 S   /usr/bin/httpd -k start -d /etc/apache
  695 nobody    3012 S   /usr/bin/httpd -k start -d /etc/apache
  696 nobody    3012 S   /usr/bin/httpd -k start -d /etc/apache
  697 nobody    3012 S   /usr/bin/httpd -k start -d /etc/apache
  698 nobody    3012 S   /usr/bin/httpd -k start -d /etc/apache
  701 root       352 S   /bin/reportip
  714 root      1176 S   -bash
  726 root       436 S   /bin/telnetd
  727 root      1180 S   -bash
  783 root       628 R   ps -ef
root@Moxa:/ect#
```

Type the following commands to see all the daemons that are set to run at bootup.

```
#cd /etc/rc.d/rc3.d
#1s
```



```
192.168.3.127 - PuTTY
root@Moxa:/ect/rc.d/rc3.d#1s
S19nfs-common      S25nfs-user-server  S99showreadyled
S20snmpd                s55ssh
S24pcmcia               S99rmnologin
root@Moxa:/etc/rc.d/rc3.d#
```



```
192.168.3.127 – PuTTY
root@Moxa:/ect/rc.d/rc3.d# ls
S19nfs-common      S25nfs-user-server  S99showreadyled
S20snmpd                S55ssh
S24pcmcia               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.

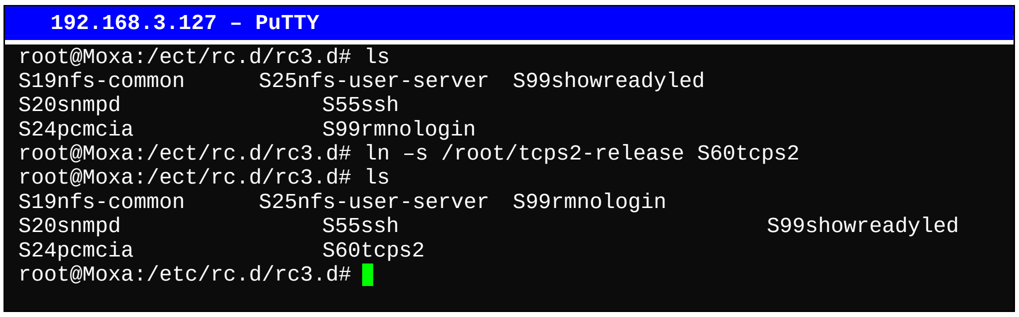
```
#cd /etc/rc.d/rc3.d
#ln –s /etc/rc.d/init.d/tcps2 S60tcps2
```

SxxRUNFILE stands for:

S: start the run file when 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
S19nfs-common      S25nfs-user-server  S99showreadyled
S20snmpd                S55ssh
S24pcmcia               S99rmnologin
root@Moxa:/ect/rc.d/rc3.d# ln –s /root/tcps2-release S60tcps2
root@Moxa:/ect/rc.d/rc3.d# ls
S19nfs-common      S25nfs-user-server  S99rmnologin
S20snmpd                S55ssh                               S99showreadyled
S24pcmcia               S60tcps2
root@Moxa:/etc/rc.d/rc3.d#
```

KxxRUNFILE stands for:

K: start the run file when linux shuts down or halts.

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

RUNFILE: the file name.

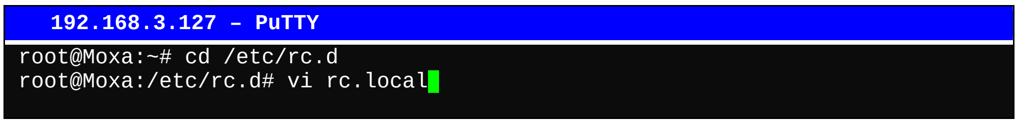
To remove the daemon, remove the run file from the **/etc/rc.d/rc3.d** directory by using the following command:

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

Starting a Program Automatically at Run-Level

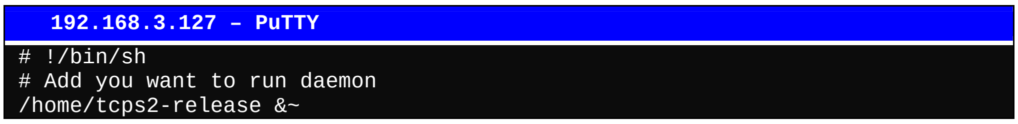
To configure a program to run automatically at run-level, add the following lines to the file rc.local:

```
#cd /etc/rc.d
#vi rc.local
```



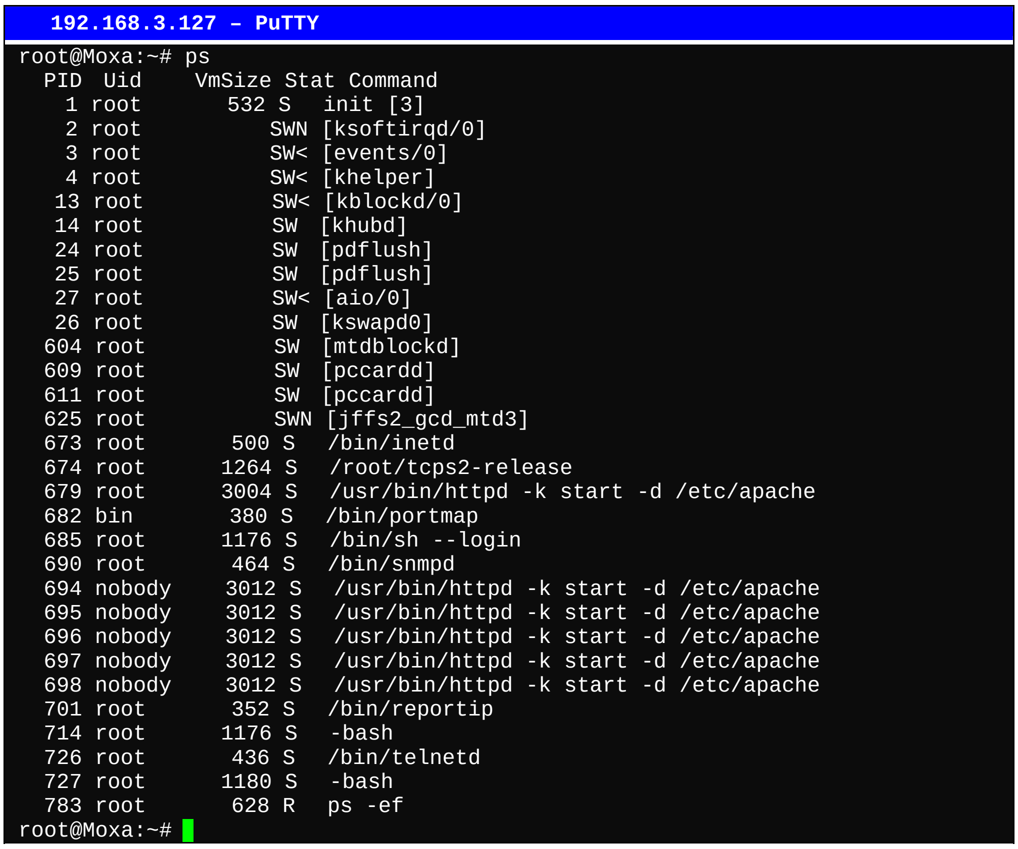
```
192.168.3.127 – PuTTY
root@Moxa:~# cd /etc/rc.d
root@Moxa:/etc/rc.d# vi rc.local
```

Next, use vi to open your application program. We use the example program **tcps2-release**, and set it to run in the background.



```
192.168.3.127 – PuTTY
# !/bin/sh
# Add you want to run daemon
/home/tcps2-release &~
```

The enabled daemons will be available after you reboot the system.



```
192.168.3.127 – PuTTY
root@Moxa:~# ps
  PID  Uid     VmSize Stat Command
    1 root        532 S   init [3]
    2 root            SWN [ksoftirqd/0]
    3 root            SW< [events/0]
    4 root            SW< [khelper]
   13 root            SW< [kblockd/0]
   14 root            SW  [khubd]
   24 root            SW  [pdflush]
   25 root            SW  [pdflush]
   27 root            SW< [aio/0]
   26 root            SW  [kswapd0]
  604 root            SW  [mtdblockd]
  609 root            SW  [pccardd]
  611 root            SW  [pccardd]
  625 root            SWN [jffs2_gcd_mtd3]
  673 root        500 S   /bin/inetd
  674 root       1264 S   /root/tcps2-release
  679 root       3004 S   /usr/bin/httpd -k start -d /etc/apache
  682 bin         380 S   /bin/portmap
  685 root       1176 S   /bin/sh --login
  690 root        464 S   /bin/snmpd
  694 nobody     3012 S   /usr/bin/httpd -k start -d /etc/apache
  695 nobody     3012 S   /usr/bin/httpd -k start -d /etc/apache
  696 nobody     3012 S   /usr/bin/httpd -k start -d /etc/apache
  697 nobody     3012 S   /usr/bin/httpd -k start -d /etc/apache
  698 nobody     3012 S   /usr/bin/httpd -k start -d /etc/apache
  701 root        352 S   /bin/reportip
  714 root       1176 S   -bash
  726 root        436 S   /bin/telnetd
  727 root       1180 S   -bash
  783 root        628 R   ps -ef
root@Moxa:~#
```

Adjusting the System Time

Setting the Time Manually

The UC-7400-LX Plus has two time settings. One is the system time, and the other is the RTC (Real-time Clock) time kept by the UC-7400-LX Plus 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
```

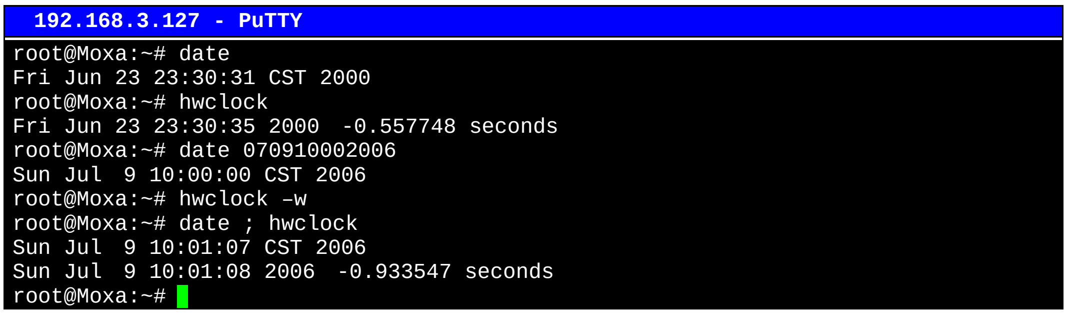
MM = Month
DD = Date
hhmm = hour and minute
YYYY = Year

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.



192.168.3.127 - PuTTY

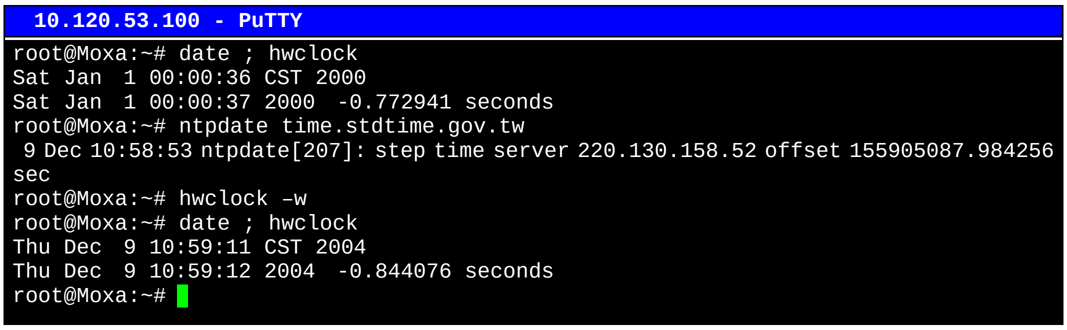
```
root@Moxa:~# date
Fri Jun 23 23:30:31 CST 2000
root@Moxa:~# hwclock
Fri Jun 23 23:30:35 2000  -0.557748 seconds
root@Moxa:~# date 070910002006
Sun Jul  9 10:00:00 CST 2006
root@Moxa:~# hwclock –w
root@Moxa:~# date ; hwclock
Sun Jul  9 10:01:07 CST 2006
Sun Jul  9 10:01:08 2006  -0.933547 seconds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

NTP Client

The UC-7400-LX Plus has a built-in NTP (Network Time Protocol) client that is used to initialize a time request to a remote NTP server. Use **#ntpdate <this client utility>** 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 155905087.984256
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:~#
```

NOTE 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 configuration 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 –systohc
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 for Executing Scheduled Commands

Cron is a scheduling service in Linux. Cron wakes up every minute, and checks the configuration file named crontab to see if any scheduled command should be run in the current minute.

Crontab is located in the **/etc/cron.d** directory. Modify the file **/etc/cron.d/crontab** to set up your scheduled applications. Crontab has 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.

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 –-systohc
exit 0
```

STEP2: Change the mode of fixtime.sh

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

STEP3: Modify the file /etc/cron.d/crontab 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 to the file /etc/init.d/rc.local

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

By default, cron service is disabled on boot. To enable cron service, please refer to the section "Enabling and Disabling Daemons" in this chapter.

Connecting Peripherals

USB Mass Storage

The UC-7420-LX Plus supports PNP (plug-n-play), and hot pluggability for connecting USB mass storage devices. The UC-7420-LX Plus has a built-in auto mount utility that eases the mounting procedure. The first USB mass storage device to be connected will be mounted automatically by **mount** to **/mnt/sda**, and the second device will be mounted automatically to **/mnt/sdb**. The UC-7420-LX Plus will be un-mounted automatically with the **umount** command 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/sda** or **/mnt/sdb** directory when you disconnect the USB mass storage device. If you stay in /mnt/sda or /mnt/sda, the auto un-mount process will fail. If that happens, type **#umount /mnt/sda** to un-mount the USB device manually.

The UC-7420-LX Plus only supports certain types of flash disk USB mass storage devices. The Following USB flash disks are supported:

- San Sandisk Cruzer mini 128MB
- Sandisk Cruzer Crossfire 1GB
- Sandisk Cruzer mini 2GB
- Intel Flash Memory 128MB
- Abocom 128MB
- PQI 256MB
- Transcend JetFlash 1G
- Transcend JetFlash 128MB
- Transcend JetFlash V30 1GB
- Transcend JetFlash V30 2GB
- ADATA My Flash 1G
- ADATA My Flash 2G

Some USB flash disks and hard disks may not be compatible with the UC-7420-LX Plus. Check compatibility issues before you purchase a USB device to connect to the UC-7420-LX Plus.

CF Mass Storage

Some models of the UC-7400-LX Plus support PNP and hot pluggability for connecting a CF mass storage device.The UC-7400-LX Plus has a built-in auto mount utility that eases the mount procedure. The CF mass storage device will be mounted automatically by the **mount** command to **/mnt/hda**. The UC-7400-LX Plus will be un-mounted automatically by **umount** when you disconnect it.

ATTENTION

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

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

The UC-7400-LX Plus only supports certain types of CF mass storage device. The following devices are supported:

- Transcend CompactFlash 45x 2GB
- Transcend CompactFlash 80x 4GB
- SanDisk CompactFlash Ultra II 1GB
- PRETEC Compactflash 128M
- PRETEC Compactflash 256M
- ADATA Compactflash 120X 4G

Some CF mass storage devices and hard disks may not be compatible with the UC-7400-LX Plus. Check compatibility issues before you purchase a CF mass storage to connect to the UC-7400-LX Plus.

4
Managing Communications

In this chapter, we explain how to configure the UC-7400-LX Plus's various communication functions.

The following topics are covered in this chapter:

❑ **Telnet / FTP**
❑ **DNS**
❑ **Web Service—Apache**
❑ **IPTABLES**
❑ **NAT**
  ➢ NAT Example
  ➢ Enabling NAT at Bootup
❑ **Dial-up Service—PPP**
❑ **PPPoE**
❑ **NFS (Network File System)**
  ➢ Setting up the UC-7400-LX Plus as an NFS Server
  ➢ Setting up the UC-7400-LX Plus as an NFS Client
❑ **Mail**
❑ **SNMP**
❑ **OpenVPN**
❑ **IPv6 to IPv4 Tunneling**

Telnet / FTP

In addition to supporting Telnet client/server and FTP client/server, the UC-7400-LX Plus 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.

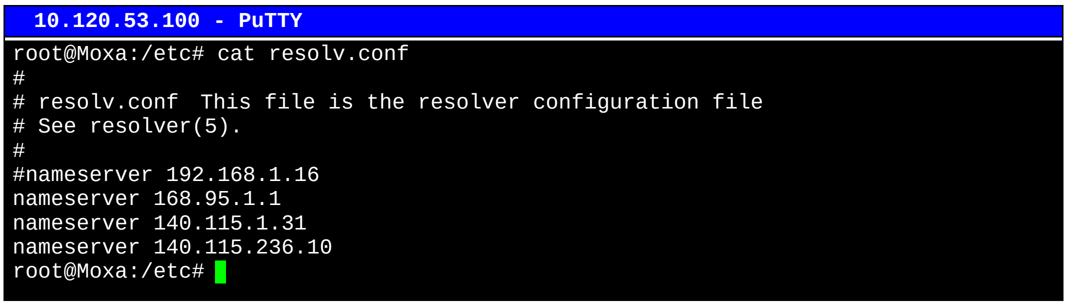
DNS

The UC-7400-LX Plus support 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` is the first file that the Linux system reads to resolve the host name and IP address.

`/etc/resolv.conf` is the most important file that you need to edit when using DNS for the other programs. For example, before using **#ntpdate time.nist.goc** 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
```


10.120.53.100 - PuTTY

```
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#
```

The file `/etc/nsswitch.conf` defines the sequence to resolve the IP address by using **/etc/hosts file** or **/etc/resolv.conf**.

Web Service—Apache

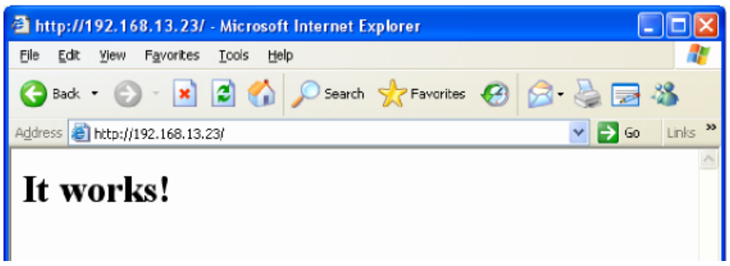
The Apache web server’s main configuration file is **/etc/apache/conf/httpd.conf**, with the default homepage located at **/home/httpd/www/html/index.html**. Save your own homepage to the following directory:

```
/home/httpd/www/html/
```

Save your CGI page to the following directory:

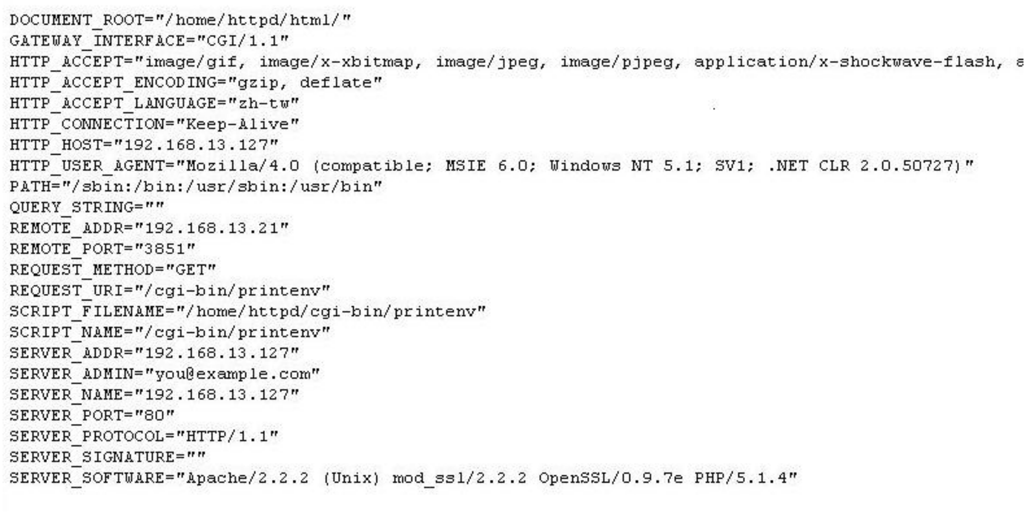
```
/home/httpd/www/cgi-bin/
```

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 box to open the homepage. E.g., if the default IP address is still active, type **http://host-ip-address** in the address box.


http://192.168.13.23/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://192.168.13.23/
Go Links

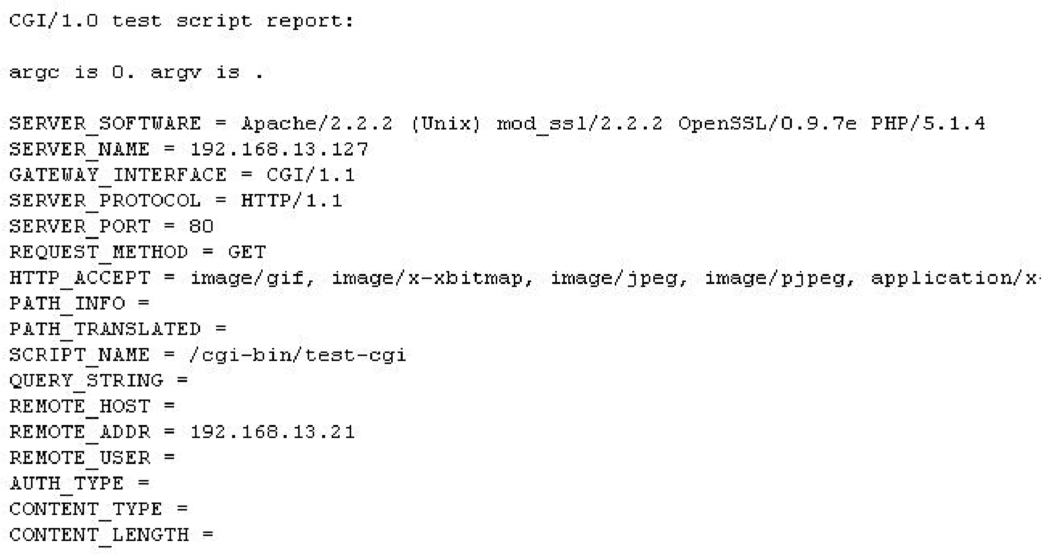
It works!

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



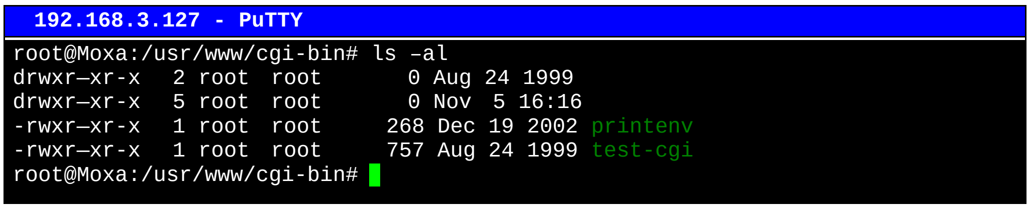
```
DOCUMENT_ROOT="/home/httpd/html/"
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, s
HTTP_ACCEPT_ENCODING="gzip, deflate"
HTTP_ACCEPT_LANGUAGE="zh-tw"
HTTP_CONNECTION="Keep-Alive"
HTTP_HOST="192.168.13.127"
HTTP_USER_AGENT="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)"
PATH="/sbin:/bin:/usr/sbin:/usr/bin"
QUERY_STRING=""
REMOTE_ADDR="192.168.13.21"
REMOTE_PORT="3851"
REQUEST_METHOD="GET"
REQUEST_URI="/cgi-bin/printenv"
SCRIPT_FILENAME="/home/httpd/cgi-bin/printenv"
SCRIPT_NAME="/cgi-bin/printenv"
SERVER_ADDR="192.168.13.127"
SERVER_ADMIN="you@example.com"
SERVER_NAME="192.168.13.127"
SERVER_PORT="80"
SERVER_PROTOCOL="HTTP/1.1"
SERVER_SIGNATURE=""
SERVER_SOFTWARE="Apache/2.2.2 (Unix) mod_ssl/2.2.2 OpenSSL/0.9.7e PHP/5.1.4"
```

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



```
CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = Apache/2.2.2 (Unix) mod_ssl/2.2.2 OpenSSL/0.9.7e PHP/5.1.4
SERVER_NAME = 192.168.13.127
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
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING =
REMOTE_HOST =
REMOTE_ADDR = 192.168.13.21
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/conf/httpd.conf**. When you develop your own CGI application, make sure your CGI file is executable.



```
192.168.3.127 - 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#
```

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-7400-LX Plus 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

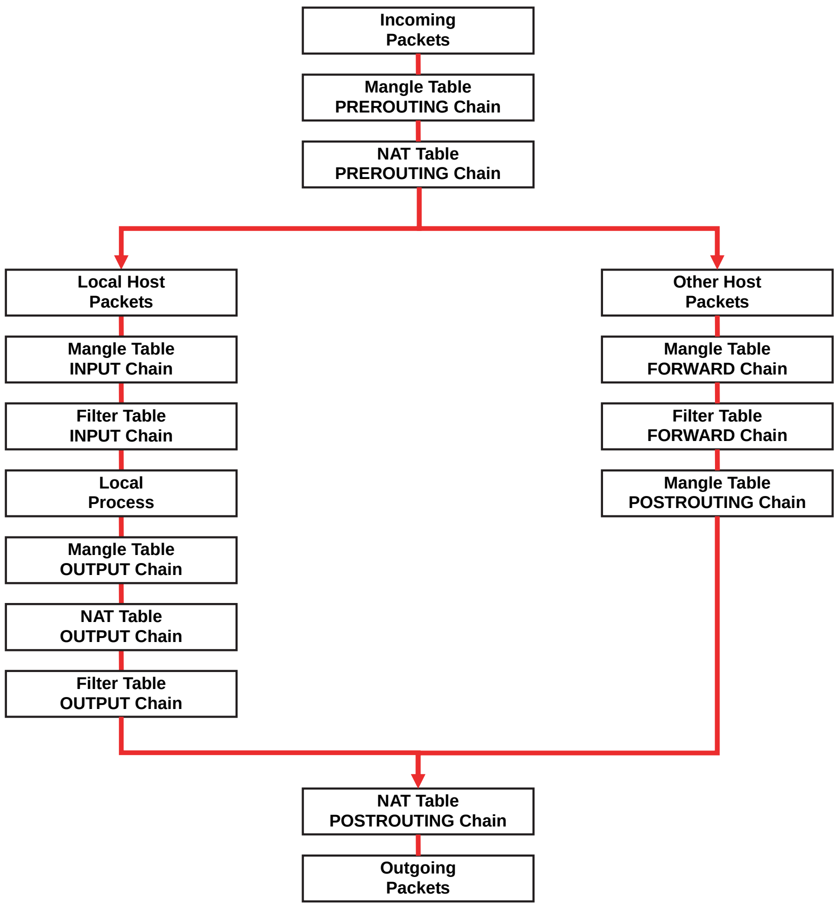
sub-tables

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-7400-LX Plus supports the following sub-modules. Be sure to use the module that matches your application.

ip_conntrack	ipt_MARK	ipt_ah	ipt_state
ip_conntrack_ftp	ipt_MASQUERADE	ipt_esp	ipt_tcpmss
ipt_conntrack_irc	ipt_LOG	ipt_length	ipt_tos
ip_nat_ftp	ipt_REDIRECT	ipt_limit	ipt_ttl
ip_nat_irc	ipt_REJECT	ipt_mac	iptable_mangle
ip_nat_snmp_basic	ipt_TCPMSS	ipt_mark	iptable_nat
ip_queue	ipt_TOS	ipt_multiport	iptable_filter
ipt_LOG	ipt_ULOG	ipt_owner	ip_tables

NOTE The UC-7400-LX Plus 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-7400-LX Plus. Use **modprobe** to insert and enable the module.

Use the following command to load the modules (**iptable_filter, iptable_mangle, iptable_nat**):

```
#modprobe iptable_filter
```

NOTE IPTABLES plays the role of packet filtering or NAT. Take care when setting up the IPTABLES rules. If the rules are not correct, remote hosts that connect through 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-7400-LX Plus.
OUTPUT: For locally-generated packets.
FORWARD: For packets routed out through the UC-7400-LX Plus.
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
#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.

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

Example 6: Accept TCP packets from 192.168.0.24 to UC-7400-LX Plus's 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 UC-7400-LX Plus's 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

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

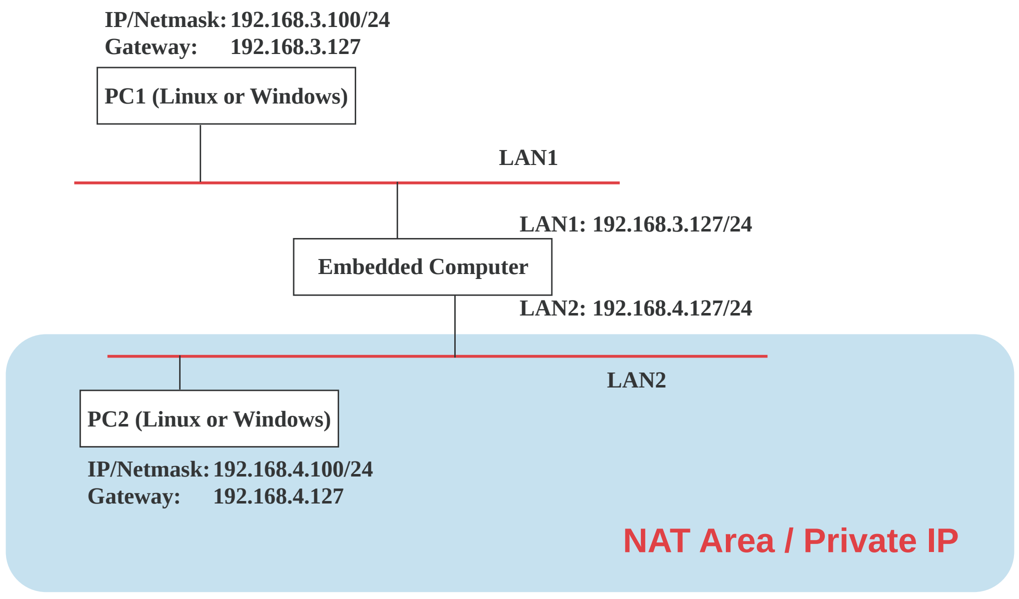
NAT

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-7400-LX Plus 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 on 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 ip_conntrack
4. #modprobe iptable_nat
5. #modprobe ipt_MASQUERADE
6. #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-7400-LX Plus 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 ip_conntrack 2> /dev/null
modprobe ip_conntrack_ftp 2> /dev/null
modprobe ip_conntrack_irc 2> /dev/null
modprobe iptable_nat 2> /dev/null
modprobe ip_nat_ftp 2> /dev/null
modprobe ip_nat_irc 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-7400-LX Plus's Ethernet port. Since PPP is a peer-to-peer system, the UC-7400-LX Plus 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 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 the network interfaces that are UP. ppp0 should be one of them, and you should recognize the first IP address as your own, and the "P-t-P address" (or point-to-point address) 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
```

This should show three routes, similar to the following:

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

If your output looks similar but doesn't 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'll have 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 requiring 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**:

```
*   *     " "   *
```

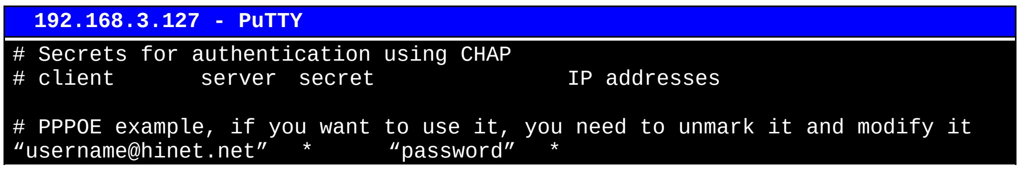
The first star (*) lets everyone log in. The second star (*) lets every host connect. The pair of double quotation marks ("") is to use the file /etc/passwd to check the password. The last star (*) is to let any IP connect.

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

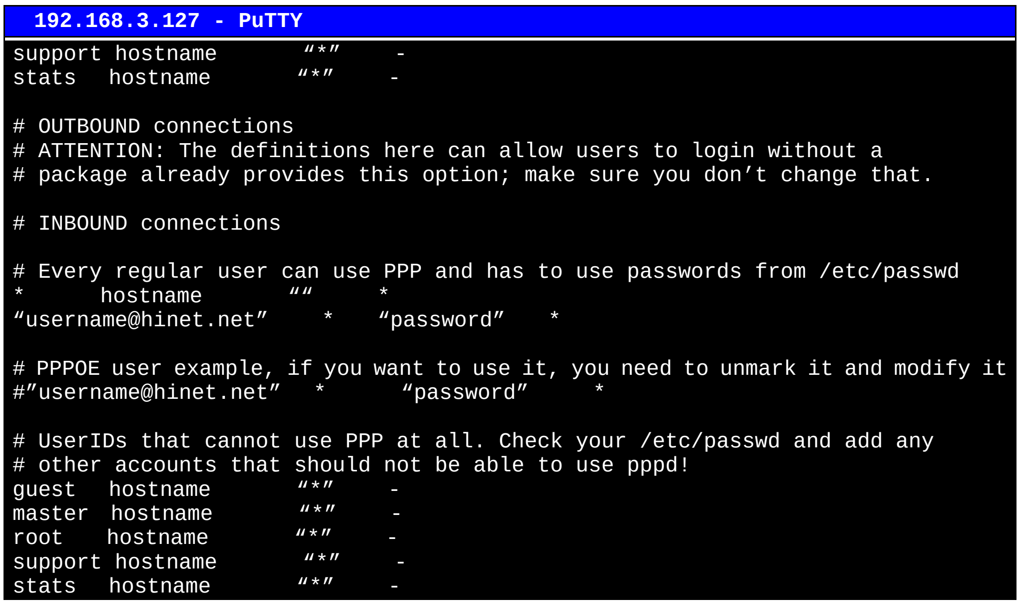
1. Connect the UC-7400-LX Plus's LAN port to an ADSL modem with a cross-over cable, HUB, or switch.
2. Login to the UC-7400-LX Plus as the root user.
3. Edit the file **/etc/ppp/chap-secrets** and add the following:
   "username@hinet.net" * "password" *



```
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.

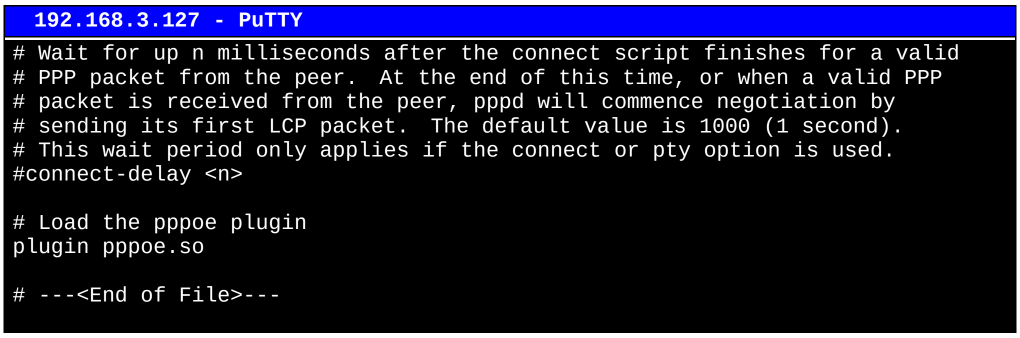
4. Edit the file **/etc/ppp/pap-secrets** and add the following:
   "username@hinet.net" * "password" *



```
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       "*"     -
```

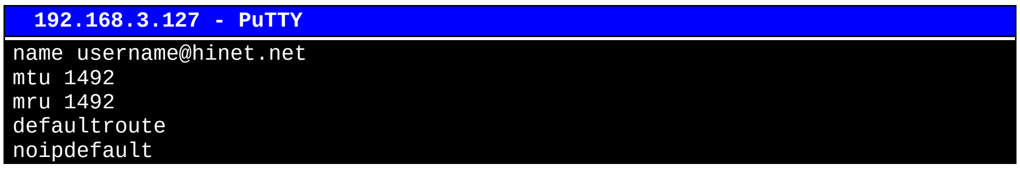
"username@hinet.net" is the username obtained from the ISP to log in to the ISP account. **"password"** is the corresponding password for the account.

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



```
192.168.3.127 - PuTTY
# 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 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:



```
192.168.3.127 - PuTTY
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)

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-7400-LX Plus, without worrying about the amount of disk space that will be available. The UC-7400-LX Plus supports NFS protocol for both client and server.

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-7400-LX Plus as an NFS Server

By default, the UC-7400-LX Plus enables the service **/etc/init.d/nfs-user-server**. The service link file **S25nfs-user-server** is located in the directory **/rc.d/rc2.d-rc5.d**.

Edit the NFS server configuration file **/etc/exports** to set up the remote host (NTF client) list and access rights for a specific directory. The file formats are shown below:

```
#vi /etc/exports
```

File Format:

```
directory machine1(option11,option12) machine2(option21,option22)
```

directory
The directory that will be shared with the NFS Client.

machine1 and machine2
Client machines that will have access to the directory. A machine can be listed by its DNS address or IP address (e.g., machine.company.com or 192.168.0.8).

optionxx

The option list for a machine describes the kind of access the machine will have. Important options are:

ro

Read only. This is the default.

rw

Readable and Writeable.

no_root_squash

If **no_root_squash** is selected, then the root on the client machine will have the same level of access to files on the system as the root on the server. This can have serious security implications, although it may be necessary if you want to do administrative work on the client machine that involves the exported directories. You should only specify this option when you have a good reason.

root_squash

Any file request made by the user root on the client machine is treated as if it is made by user nobody on the server. (Exactly which UID the request is mapped to depends on the UID of user "nobody" on the server, not the client.)

sync

Sync data to memory and flash disk.

async

The async option instructs the server to lie to the client, telling the client that all data has been written to the stable storage.

Example 1

```
/tmp  *(rw,no_root_squash)
```

In this example, the UC-7400-LX Plus shares the **/tmp** directory to everyone, giving everyone both read and write authority. The root user on the client machine will have the same level of access to files on the system as the root on the server.

Example 2

```
/home/public 192.168.0.0/24(rw) *(ro)
```

In this example, the UC-7400-LX Plus shares the directory **/home/public** to the local network with IP address 192.168.0.0/24, with read and write authority. NFS clients will just be able to read **/home/public**; they do not have write authority.

Example 3

```
/home/test  192.168.3.100(rw)
```

In this example, the UC-7400-LX Plus shares the directory **/home/test** to an NFS Client 192.168.3.100, with both read and write authority.

NOTE After editing the NFS Server configuration file, remember to use the following command to restart and activate the NFS server.

```
/etc/init.d/nfs-user-server restart
```

Setting up the UC-7400-LX Plus as an NFS Client

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

1. Scan the NFS Server's shared directory.
2. Establish a mount point on the NFS Client site.
3. Mount the remote directory to a local directory.

Step 1:

```
#showmount  –e  HOST
```

showmount:	Show the mount information for an NFS Server.
-e:	Show the NFS Server's export list.
HOST:	IP address or DNS address.

Steps 2 & 3:

```
#mkdir –p /home/nfs/public
#mount –t nfs NFS_Server(IP):/directory /mount/point
```

Example:

```
#mount –t nfs 192.168.3.100/home/public  /home/nfs/public
```

Mail

smtpclient is a minimal SMTP client that takes an email message body and passes it on to an SMTP server. It is suitable for applications that use email to send alert messages or important logs to a specific user.

NOTE Click on the following link for more information about smtpclient:
http://www.engelschall.com/sw/smtpclient/

To send an email message, use the 'smtpclient' utility, which uses SMTP protocol. Type **#smtpclient –help** to see the help message.

Example:

```
smtpclient –s test –f sender@company.com –S IP_address receiver@company.com
< mail-body-message
```

-s: The mail subject.
-f: Sender's mail address
-S: SMTP server IP address

The last mail address, **receiver@company.com**, is the receiver's e-mail address. **mail-body-message** is the mail content. The last line of the body of the message should contain ONLY the period '.' character.

You will need to add your hostname to the file **/etc/hosts**.

SNMP

The UC-7400-LX Plus has SNMP V1 (Simple Network Management Protocol) agent software built in. It supports RFC1317 RS-232 like groups and RFC 1213 MIB-II.

The following simple example allows you to use an SNMP browser on the host site to query the UC-7400-LX Plus, which is the SNMP agent. The snmp agent will respond.

```
***** SNMP QUERY STARTED *****
1: sysDescr.0 (octet string) Linux Moxa 2.6.10_dev-ixdp42x-arm_xscale_be
2: sysObjectID.0 (object identifier) enterprises.2021.250.10
3: sysUpTime.0 (timeticks) 0 days 00h:41m:54s.47th (251447)
4: sysContact.0 (octet string) Root <root@localhost> (configure /etc/snmp/snmp.local.conf)
5: sysName.0 (octet string) Moxa
6: sysLocation.0 (octet string) Unknown (configure /etc/snmp/snmp.local.conf)
7: system.8.0 (timeticks) 0 days 00h:00m:00s.22th (22)
8: system.9.1.2.1 (object identifier) mib-2.31
9: system.9.1.2.2 (object identifier) internet.6.3.1
10: system.9.1.2.3 (object identifier) mib-2.49
11: system.9.1.2.4 (object identifier) ip
12: system.9.1.2.5 (object identifier) mib-2.50
13: system.9.1.2.6 (object identifier) internet.6.3.16.2.2.1
14: system.9.1.2.7 (object identifier) internet.6.3.10.3.1.1
15: system.9.1.2.8 (object identifier) internet.6.3.11.3.1.1
16: system.9.1.2.9 (object identifier) internet.6.3.15.2.1.1
17: system.9.1.3.1 (octet string) The MIB module to describe generic objects for network interface sub-layers
18: system.9.1.3.2 (octet string) The MIB module for SNMPv2 entities
19: system.9.1.3.3 (octet string) The MIB module for managing TCP implementations
20: system.9.1.3.4 (octet string) The MIB module for managing IP and ICMP implementations
21: system.9.1.3.5 (octet string) The MIB module for managing UDP implementations
22: system.9.1.3.6 (octet string) View-based Access Control Model for SNMP.
23: system.9.1.3.7 (octet string) The SNMP Management Architecture MIB.
24: system.9.1.3.8 (octet string) The MIB for Message Processing and Dispatching.
25: system.9.1.3.9 (octet string) The management information definitions for the SNMP User-based Security Model.
26: system.9.1.4.1 (timeticks) 0 days 00h:00m:00s.04th (4)
27: system.9.1.4.2 (timeticks) 0 days 00h:00m:00s.09th (9)
28: system.9.1.4.3 (timeticks) 0 days 00h:00m:00s.09th (9)
29: system.9.1.4.4 (timeticks) 0 days 00h:00m:00s.09th (9)
30: system.9.1.4.5 (timeticks) 0 days 00h:00m:00s.09th (9)
31: system.9.1.4.6 (timeticks) 0 days 00h:00m:00s.19th (19)
32: system.9.1.4.7 (timeticks) 0 days 00h:00m:00s.22th (22)
33: system.9.1.4.8 (timeticks) 0 days 00h:00m:00s.22th (22)
34: system.9.1.4.9 (timeticks) 0 days 00h:00m:00s.22th (22)
***** SNMP QUERY FINISHED *****
```

NOTE Click on the following links for more information about MIB II and RS-232 like groups:
http://www.faqs.org/rfcs/rfc1213.html
http://www.faqs.org/rfcs/rfc1317.html

The UC-7400-LX Plus 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:

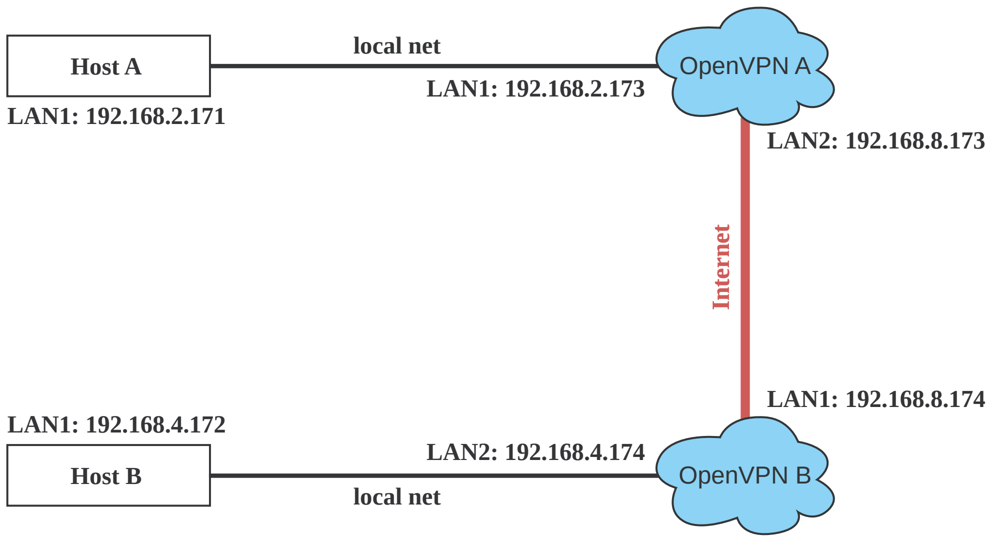
```
# mknod /dev/net/tun c 10 200
```

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 interface "eth1" as IP-less, creates logical bridge(s) and TAP interfaces, loads modules, enables IP forwarding, etc.

```
#--------------------------------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 ----------------------------
```

Create link symbols to enable this script at boot time:

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

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
```

Then modify the routing table in **/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
```

Then modify the routing table in **/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 by specifying **cipher**. To see which ciphers are available, type:

```
# openvpn --show-ciphers
```

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
   ```

   Then start both of OpenVPN peers,

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

   If you see the line "Peer Connection Initiated with 192.168.8.173:1194" on each machine, the connection between OpenVPN machines has been established successfully on UDP port 1194.

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, you need to 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, you need to 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
   ```

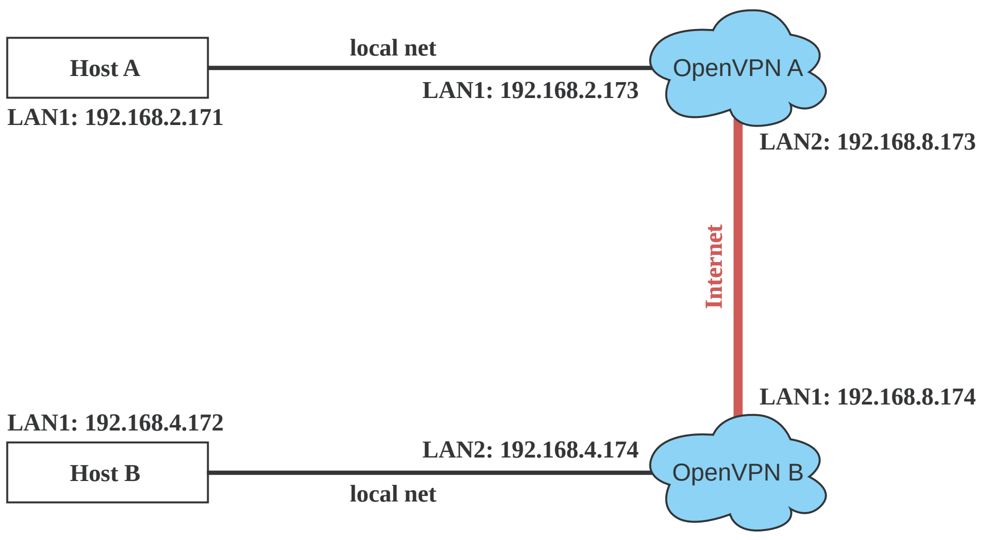
   A successful ping indicates that you have created a VPN system that only allows authorized users from one internal network to access users at the remote site. For this system, all data is transmitted by UDP packets on port 1194 between OpenVPN peers.

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

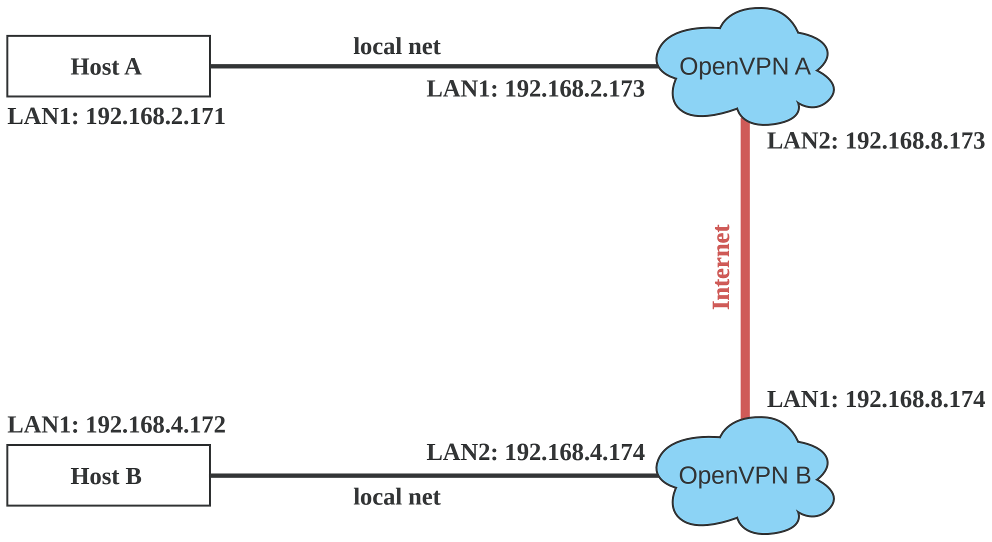
1. Set up four machines as shown in the following diagram:



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
```

Then modify the routing table in **/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
```

Then modify the routing table in **/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 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 **$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

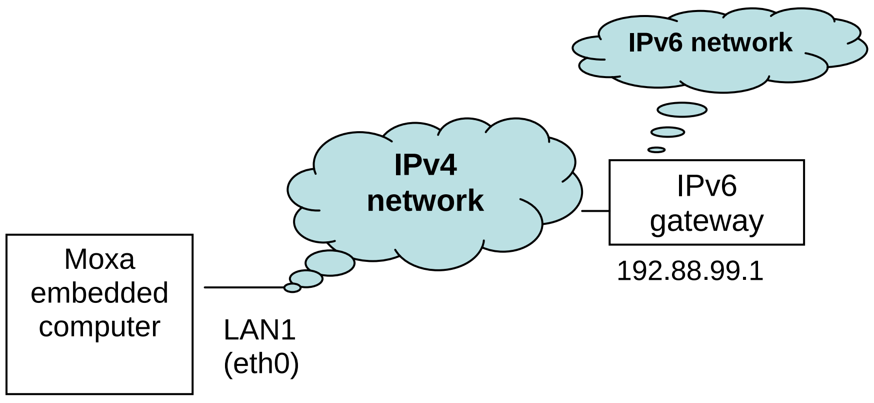
IPv6 to IPv4 Tunneling

Note: The IPv6 feature is provided starting with firmware version v1.6. After upgrading the firmware, you must load the factory default to activate the IPv6 feature.

IPv6 to IPv4 tunneling provides a simple way to enable the new generation IP protocol on your Linux host by tunneling IPv6 packets over an IPv4 connection. This section will explain how to configure your computer and enable the IPv6 to IPv4 tunneling function.

Step1: Connect the computer's LAN1 port to the Internet.

If your Linux system has a static IPv4 address, the tunnel can also be statically configured in the network definitions. Assume your IPv6 Internet is connected via LAN1; you can configure the IPv6 to IPv4 tunnel through the LAN1 port (eth0) by connecting to the IPv6 network behind the IPv6 gateway.



Step 2: Configure LAN1 (eth0) IPv4 network settings.

(The values and parameters used below are for illustration only. Be sure to enter the correct values and parameters for your network.)

Let's assume that the following settings represent your network's current situation:

IP: 61.56.74.14 (Global)
Mask: 255.255.255.248
Gateway: 61.56.74.9

Use the following commands to configure these settings:

```
61.56.74.14
```

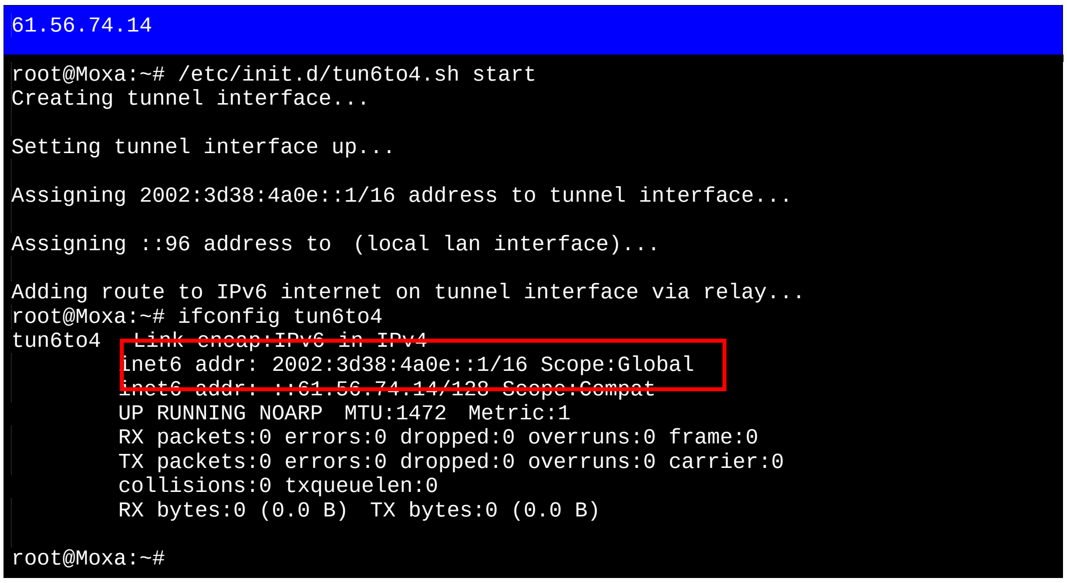
```
root@Moxa:~# ifconfig eth0 61.56.74.14 netmask 255.255.255.248
root@Moxa:~# route add default gw 61.56.74.9
```

Step 3: Configure WAN_LOCAL as eth0 in /etc/rc.d/init.d/tun6to4.sh.

```
61.56.74.14
```

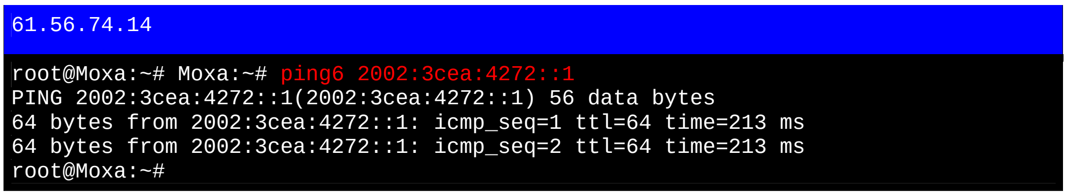
```
#!/bin/bash

TUN_INTF=tun6to4  # The name of the 6to4 tunnel link
WAN_LOCAL=eth0
WAN_IP=$(ifconfig $WAN_LOCAL |grep 'inet addr'|awk '{print $2}'|cut -d':' -f 2)
WAN_IPV6_IP=$(printf "2002:%02x%02x:%02x%02x::1/16" $(echo "${WAN_IP}"| tr '.' ' '))
TUN_RELAY=192.88.99.1  # 6to4 relay address


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
}

case "$1" in

       start)
…


       *)

              echo "Usage: /etc/init.d/tun6to4 {start|stop}"

              exit 1

              ;;

esac
```

Step 4: Run the tun6to4.sh script to activate the 6to4 tunnel. You will be assigned a new IPv6-based IP address.



```
61.56.74.14

root@Moxa:~# /etc/init.d/tun6to4.sh start
Creating tunnel interface...

Setting tunnel interface up...

Assigning 2002:3d38:4a0e::1/16 address to tunnel interface...

Assigning ::96 address to  (local lan interface)...

Adding route to IPv6 internet on tunnel interface via relay...
root@Moxa:~# ifconfig tun6to4
tun6to4   Link encap:IPv6 in IPv4
        inet6 addr: 2002:3d38:4a0e::1/16 Scope:Global
        inet6 addr: ::61.56.74.14/128 Scope:Compat
        UP RUNNING NOARP  MTU:1472  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@Moxa:~#
```

Step 5: Test the 6to4 tunnel with the ping6 command to see if the remote server can echo the PING response. If receive the echo, then the IPv6-to-IPv4 tunnel has been successfully established.



```
61.56.74.14

root@Moxa:~# Moxa:~# ping6 2002:3cea:4272::1
PING 2002:3cea:4272::1(2002:3cea:4272::1) 56 data bytes
64 bytes from 2002:3cea:4272::1: icmp_seq=1 ttl=64 time=213 ms
64 bytes from 2002:3cea:4272::1: icmp_seq=2 ttl=64 time=213 ms
root@Moxa:~#
```

5

Programmer's Guide

This chapter includes important information for programmers.

The following functions are covered in this chapter:

- ❑ **Before Programming Your Embedded System**
  - ➢ Caution When Using File Systems
  - ➢ Using RAM File System Instead of Flash File System
- ❑ **Flash Memory Map**
- ❑ **Linux Tool Chain Introduction**
- ❑ **Debugging with GDB**
- ❑ **Device API**
- ❑ **RTC (Real-time Clock)**
- ❑ **Buzzer**
- ❑ **WDT (Watch dog Timer)**
- ❑ **UART**
- ❑ **LCM**
- ❑ **KeyPad**
- ❑ **Make File Example**

Before Programming Your Embedded System

Caution When Using File Systems

We recommend placing only your programs into the on-board NOR Flash. For the log data generated by your programs, please store them into an external storage such as Compact Flash or Network File System. The latter has larger space than the former. In addition, it is easier replacing a full or damaged Compact Flash than an on-board NOR Flash.

A NOR Flash has a life cycle of 100,000 write operations in block (128KB) level. It does not support BBM (Bad Block Management). A CF card has its life cycle too. But most of them are made of NAND Flash, their hardware controllers implement BBM. This feature allows FAT to skip bad blocks if there are. Furthermore, the memory space of a CF card is much larger than that of the NOR Flash. Cautiously utilizing this space will make sure its life cycle would not be exceeded. When creating a file for storing log data, your program is suggested to create a large empty file (eg., 30MB) and then evenly writes data over the space. When reaching the end of the space, the program rewinds the write operations. As a result, the number of write operations on each block reduces.

Using RAM File System Instead of Flash File System

Though data in the RAM file system will be wiped out after a power off, this file system has several advantages over the Flash file system. They include faster read/write access and having no life cycle issue.

For a timely important application that relays data back to the host directly, please write the necessary log data into the RAM file system. After the host accesses the data, the application erases the area for further use.

The embedded computer has limited resource, the designers should determine if storing data in a file system is really a must or not. If it is necessary, choose the most proper file system.

Flash Memory Map

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

Address	Size	Contents
0x00000000 – 0x0005FFFF	384 KB	Boot Loader—Read ONLY
0x00060000 – 0x001FFFFF	1.625 MB	Kernel object code—Read ONLY
0x00200000 – 0x00DFFFFF	14 MB	Root file system (JFFS2)—Read ONLY
0x00E00000 – 0x01FCFFFF	17.75 MB	User root file system (JFFS2)—Read/Write
0x01FC0000 – 0x01FDFFFF	128 KB	Boot Loader configuration—Read ONLY
0x01FE0000 – 0x01FFFFFF	128 KB	Boot Loader directory—Read ONLY

Linux Tool Chain Introduction

To ensure that an application will be able to run correctly when installed on the UC-7400-LX Plus, you must ensure that it is compiled and linked to the same libraries that will be present on the UC-7400-LX Plus. This is particularly true when the RISC Xscale processor architecture of the UC-7400-LX Plus 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-7400-LX Plus 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-7400-LX Plus. The host environment must be running Linux to install the UC-7400-LX Plus GNU Tool Chain. We have confirmed that the following Linux distribution can be used to install the tool chain:

Fedora core 1/2/3/4/5.

The Tool Chain will need about 900 MB of hard disk space on your PC. The UC-7400-LX Plus Tool Chain is located on the UC-7400-LX Plus CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount –t iso9660 /dev/cdrom /mnt/cdrom
#cp /mnt/cdrom/tool-chain/linux/xscale_be_1.1.sh /tmp/
#sh /tmp/xscale_be_1.1.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/xscale_be/bin/** to your path and the directory **/usr/local/xscale_be/man/** to your manual path. You can do this temporarily for the current login session by issuing the following commands:

```
#export PATH="/usr/local/xscale_be/bin":$PATH
#export MANPATH="/usr/local/xscale_be/man":$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

Use the Linux man utility to obtain help on many of the utilities provided by the tool chain. For example to get help on the xscale_be-gcc compiler, issue the command:

#man xscale_be-gcc

Cross Compiling Applications and Libraries

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

```
#xscale_be-gcc  –o example –Wall –g –O2 example.c
#xscale_be-strip  –s example
#xscale_be-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 UC-7400-LX Plus Xscale boards, it is **xscale_be-**.

For example, the native C compiler is gcc and the cross C compiler for Xscale in UC-7400-LX Plus is **xscale_be-gcc**.

The following cross compiler tools are provided:

ar	Manage 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

Before you debug with GDB, you must compile your program with the option -ggdb. Use the following steps to do this:

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_be-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-7400-LX Plus supports control devices with the **ioctl** system API. You will need to use **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-7400-LX Plus 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 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 RTC.

Buzzer

The device node is located at **/dev/console**. The UC-7400-LX Plus supports Linux standard buzzer control, with the UC-7400-LX Plus’s buzzer running at a fixed frequency of 100 Hz. You must use **include <sys/kd.h>**.

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 at 100 Hz. If you use a different frequency, the system could crash.

WDT (Watch dog Timer)

1. **Introduction**

   The WDT works like a watchdog to monitor system by protecting the system against unexpected errors.

2. **How does the WDT work?**

   WDT has two modes: kernel mode and user mode. When the system boots up, it enters the kernel mode by default. During normal operation, the kernel or user's application sends signals to the WDT to report that the system or application is working well. In kernel mode, the kernel sends signals (acknowledgements) periodically to the WDT to keep the system alive. If the kernel fails to acknowledge the WDT within an acknowledgement interval, the WDT will force the system to reboot. Similarly, when users include WDT functions in their applications, the WDT switches to user mode automatically. If the user enables the WDT and the application does not acknowledge it, the system will reboot. You can set the acknowledgement interval from a minimum of 50 milliseconds to a maximum of 60 seconds.

3. **The user API**

   The user application must include <moxadevic.h>, and link to moxalib.a. A makefile example is shown below:

```
all:
  arm-elf-gcc –Wl, -elf2flt –o xxxx  xxxx.c -lmoxalib
```

int swtd_open(void)

Description

Open a file handle to control the watchdog. This is the first step to using the watchdog in your application.

Input

None

Output

The return value is the file handle. If there is an error, it will return a negative value. You can use errno() to retrieve the error.

int swtd_enable(int fd, unsigned long time)

Description

This function enables the watchdog function in an application. You must acknowledge after implementing this process.

Input

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

unsigned long time — The time you wish to acknowledge the watchdog periodically.

NOTE: You must acknowledge the watchdog before timeout. If you do not acknowledge, the system will be reboot automatically. The minimum time interval is 50 milliseconds, and the maximum time interval is 60 seconds. The time unit is in millisecond.

Output

0 if OK. An error occurs if you see a non-zero value. Use the function errno() to retrieve the error code.

`int swtd_disable(int fd)`

Description:

Disable the watchdog in the application. The kernel will automatically acknowledge the watchdog.

Input:

int fd — the file handle from the return value of swtd_open().

Output:

0 if OK. An error occurs if you see a non-zero value. Use the function errno () to retrieve the error code.

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

Description:

Get current settings.

mode —

1 to set user mode

0 to set kernel mode

time — The time period to acknowledge the watchdog.

Input:

int fd — the file handle from swtd_open().

int *mode — the function will return the status of the watchdog function

unsigned long *time — the function will return the current time period.

Output:

0 if OK. An error occurs if you see a non-zero value. Use the function errno () to retrieve the error code.

`int swtd_ack(int fd)`

Description:

Acknowledge the watchdog. When the user application enables the watchdog, it needs to call this function periodically with time predefined by the user in the application program.

Input:

int fd — the file handle from the swtd_open() return value.

Output:

0 if OK. An error occurs if you see a non-zero value. Use the function errno () to retrieve the error code.

`int swtd_close(int fd)`

Description:

Close the file handle.

Input:

int fd — the file handle from the return value of swtd_open().

Output:

0 if OK. An error occurs if you see a non-zero value. Use the function errno () to retrieve the error code.

4. **Special Note**

   When you "kill the application with -9" or "kill without option" or "Ctrl+c" the watchdog function will be changed from user mode to kernel mode.

   When your application enables the watchdog and does not acknowledge the WDT, your application may have a logical error, or your application has made a core dump. Thus, the watchdog function will not be changed to kernel mode. The WDT waits until the acknowledgement timeout and force the system to reboot. This can cause a serious problem. If errors in your application are not fixed, your system will 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 watchdog 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:
  arm-elf-gcc –Wl, -elf2flt –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_watchdog()
{
  stopflag = 1;
}

static void do_watchdog(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_watchdog);
  swtdfd = swtd_open();
  if ( swtdfd < 0 ) {
   printf("Open watchdog device fail !\n");
   exit(1);
  }
  if ( (sonpid=fork()) == 0 )
   do_watchdog();
  // do user application main function
  …..
  …..
  …..
  // end user application
  kill(sonpid, SIGUSR1);
  swtd_close(swtdfd);
  exit(1);
}
```

```
The makefile is shown below:
all:
  arm-elf-gcc –Wl, -elf2flt –o xxxx xxxx.c –lmoxalib
```

UART

The normal tty device node is located at **/dev/ttyM0 … ttyM15,** and the modem tty device node is located at **/dev/cum0 … cum15**.

The UC-7400-LX Plus supports Linux standard termios control. The Moxa UART Device API allows you to configure ttyM0 to ttyM7 as RS-232, RS-422, 4-wire RS-485, or 2-wire RS-485. The UC-7400-LX Plus supports RS-232, RS-422, 2-wire RS-485, and 4-wire RS485.

You must use **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 define 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.

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 baudrate define
} else {
  ioctl(fd, MOXA_GET_SPECIAL_BAUD_RATE, &speed);
}
```

Baudrate inaccuracy

Divisor = 921600/Target Baudrate. (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%
*The Inaccuracy should less than 2% for 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 speed equal to 0.

LCM

The UC-7400-LX Plus only supports text mode display, with screen size of 16 cols by 8 rows. The device node is **/dev/lcm**. See the examples given below. We provide a private struct defined as follows:

```
typedef struct lcm_xy {
   int x; // col value, the arrange is 0 – 15
   int y; // raw value, the arrange is 0 – 1
} lcm_xy_t;
```

Examples

`int ioctl(fd, IOCTL_LCM_GOTO_XY, lcm_xy_t *pos);`

Move the cursor position to the x(col),y(raw) position. Argument 3 is the new position value.

`int ioctl(fd, IOCTL_LCM_CLS, NULL);`

Clears the LCM display.

`int ioctl(fd, IOCTL_LCM_CLEAN_LINE, NULL);`

To change one line to all spaces in the current row, and move the cursor to the 0 column of this row.

`int ioctl(fd, IOCTL_LCM_GET_XY, lcm_xy_t *pos);`

Get the current cursor position. The value will be returned in argument 3.

`int ioctl(fd, IOCTL_LCM_BACK_LIGH_ON, NULL);`

Turns the LCM backlight on.

`int ioctl(fd, IOCTL_LCM_BACK_LIGHT_OFF, NULL);`

Turns the LCM backlight off.

KeyPad

The device node is **/dev/keypad**. The key value is defined in moxadevice.h.

`int ioctl(fd, IOCTL_KEYPAD_HAS_PRESS, int *flag);`

Checks how many keys have been pressed. Argument 3 returns the number of pressed keys. 0 means no keys were pressed.

`int ioctl(fd, IOCTL_KEYPAD_GET_KEY, int *key);`

Gets the value of the last key that was pressed. This function only reads one key value for each function call. The value of the key value is returned in argument 3.

Special Note

1. The UC-7400-LX Plus's kernel will store the "pressed key history" in a buffer. The maximum buffer size is 31 keys. If the buffer overflows, the first key of the 31 that was pressed will be dropped, without sounding the buzzer.
2. Currently, the UC-7400-LX Plus does NOT support pressing more than 1 key at the same time.

Make File Example

The following Makefile file example codes are copied from the Hello example on the UC-7400-LX Plus CD-ROM.

```
CC = xscale_be-gcc
CPP = xscale_be-gcc
SOURCES = hello.c

OBJS =  $(SOURCES:.c=.o)

all: hello

hello: $(OBJS)
$(CC) -o $@ $^ $(LDFLAGS) $(LIBS)

clean:
rm -f $(OBJS) hello core *.gdb
```

6
Software Lock

“Software Lock” is an innovative technology developed by Moxa’s engineers. It can be adopted by a system integrator or developer to protect applications from being copied. An applicaion is compiled into a binary format bound to the embedded computer and the operating system (OS) that the application runs on. As long as one obtains it from the computer, he/she can install it on the same hardware and the same operating system. The add-on value created by the developer is thus lost.

Moxa’s “Software Lock” function uses data encryption. The binary file associated with each application needs to undergo an additional encryption process after it has been developed. The process requires that you install an encryption key on the target computer.

1. Choose an encryption key (e.g., “ABigKey”) and install it in the target computer by a pre-utility program, ‘setkey’.

```
#setkey ABigKey
```

Note: set an empty string to clear the encryption key in the target computer by:

```
#setkey ““
```

2. Develop and compile your program on the development PC.
3. Run the utility program ‘binencryptor’ on the development PC 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. A computer has only one key installed. Running the program ‘setkey’ multiple times causes the key to be overwritten.

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

This mechanism also allows computers that have an encryption key to bypass programs that are not encrypted. Therefore, in the development phase, you can develop your programs and test them in the target computer.

NOTE

1. If you try to run an encrypted program on an embedded computer that does not have an encryption key installed, you will get the following error message.
   Error =>
   Inconsistency detected by ld.so: dynamic-link.h: 62: elf_get_dynamic_info: Assertion `! “bad dynamic tag”‘ failed!

2. If you try to run an encrypted program on an embedded computer that has a different encryption key installed, you will get the following error message:
   Error =>
   Segmentation fault

A
System Commands

Linux normal command utility collection

System

1.	hwclock	query and set the hardware clock (RTC)
2.	klogd	Kernel Log Daemon
3.	logger	a shell command interface to the syslog(3) system log module
4.	mesg	control write access to your terminal
5.	poweroff	stop the system
6.	umount	unmount file systems
7.	uname	print system information
8.	syslogd	Linux system logging utilities
9.	uptime	tell how long the system has been running

File Manager

1.	cp	copy file
2.	ls	list file
3.	ln	make symbolic link file
4.	mount	mount and check file system
5.	rm	delete file
6.	chmod	change file owner & group & user
7.	chown	change file owner
8.	chgrp	change file group
9.	sync	sync file system, let system file buffer be saved to hardware
10.	mv	move file
11.	pwd	display now file directly
12.	df	list now file system space
13.	mkdir	make new directory
14.	rmdir	delete directory
15.	swapoff	stop swapping to file/device
16.	swapon	start swapping to file/device
17.	touch	change file timestamps
18.	unzip	list, test and extract compressed files in a ZIP archive
19.	tar	tar archiving utility

Editor

1.	vi	text editor
2.	cat	dump file context
3.	zcat	compress or expand files
4.	grep, egrep, fgrep	search string on file
5.	cut	get string on file
6.	find	find file where are there
7.	more	dump file by one page
8.	test	test file exist or not
9.	sleep	sleep (seconds)
10.	echo	echo string
11.	tail	output the last part of files
12.	uniq	remove duplicate lines from a sorted file
13.	sed	stream editor
14.	tr	translate or delete characters
15.	wc	print the number of bytes, words, and lines in files

Network

1.	ping	ping to test network
2.	route	routing table manager
3.	netstat	display network status
4.	ifconfig	set network ip address
5.	tftp	trivial file transfer protocol
6.	arp	manipulate the system ARP cache
7.	ifdown	shutdown a network interface
8.	ifup	bring up a network interface
9.	ip	TCP/IP interface configuration and routing utility
10.	ping6	ping for IPv6 address

Process

1.	kill	kill process
2.	ps	display now running process
3.	fuser	identify processes using files or sockets
4.	pidof	find the process ID of a running program
5.	killall	kill processes by name
6.	renice	alter priority of running processes
7.	top display top CPU processes	

Other

1.	dmesg	dump kernel log message
2.	sty	to set serial port
3.	zcat	dump .gz file context
4.	mknod	make device node
5.	free	display system memory usage
6.	date	print or set the system date and time
7.	env	run a program in a modified environment
8.	clear	clear the terminal screen
9.	reboot	reboot / power off/on the server
10.	halt	halt the server
11.	du	estimate file space usage
12.	gzip, gunzip	compress or expand files
13.	hostname	show system's host name
14.	basename	strip directory and suffix from filenames
15.	chroot	run command or interactive shell with special root directory
16.	dirname	strip non-directory suffix from file name
17.	expr	evaluate expressions
18.	false	do nothing, unsuccessfully
19.	head	output the first part of files
20.	id	print real and effective UIDs and GIDs
21.	mkfifo	make FIFOs (named pipes)
22.	nice	run a program with modified scheduling priority
23.	nohup	run a command immune to hangups, with output to a non-tty
24.	reset	terminal initialization
25.	true	exit with a status code indicating success
26.	reset	terminal initialization
27.	rmmod	unload loadable modules
28.	stty	change and print terminal line settings
29.	usleep	sleep some number of microseconds
30.	xargs	build and execute command lines from standard input
31.	yes	output a string repeatedly until killed

Moxa Special Utilities

1.	kversion	show kernel version
2.	upramdisk	mount ramdisk
3.	downramdisk	unmount ramdisk