UC-7408 User's Manual

Sixth Edition, February 2009

www.moxa.com/product

UC-7408 User's Manual

Copyright Notice

Trademarks

Disclaimer

Information in this document is subject to change without notice and does not represent a commitment on the part of Moxa.

Moxa provides this document "as is," without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Moxa reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.

Information provided in this manual is intended to be accurate and reliable. However, Moxa assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.

This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

Technical Support Contact Information
www.moxa.com/support

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

Moxa China (Shanghai office):
Toll-free: 800-820-5036
Tel: +86-21-5258-9955
Fax: +86-10-6872-3958

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

Moxa Asia-Pacific:
Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

Table of Contents

1

Introduction

Welcome to Moxa UC-7408, a data acquisition embedded computer that features 8 RS-232/422/485 serial ports, dual 10/100 Mbps Ethernet ports, a PCMCIA interface for wireless LAN communication, 8 digital inputs and 8 digital outputs, and CompactFlash for mass storage disk expansion.

The digital I/O feature of UC-7408 provides users with the convenience of connecting digital devices to a front-end embedded computer. UC-7408 can be used for on/off event handling by reading the state change of the digital input signal. Output signals from external digital devices can be imported into UC-7408's digital input channels, and then UC-7408 can be programmed to take immediate action when it detects a change in the signal state. The digital output channels on UC-7408 can connect to devices and trigger digital output signals to control external digital devices. The digital I/O feature allows Moxa UC-7408 to support data acquisition and protocol conversion via the RS-232/422/485 serial ports, as well as simple I/O control with the digital I/O signal.

The following topics are covered in this chapter:

Overview

UC-7408 data acquisition embedded computers are ideal for embedded applications. UC-7408 has 8 RS-232/422/485 serial ports, dual 10/100 Mbps Ethernet ports, 8 digital input and 8 digital outputs, a PCMCIA interface for wireless LAN communication, and CompactFlash for mass storage flash disk expansion.

UC-7408 uses an Intel XScale IXP422 266 Mhz RISC CPU. Unlike the X86 CPU, which uses a CISC design, the IXP422's RISC design architecture and modern semiconductor technology provide UC-7408 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-7408. And since the dual LAN ports are built right into the IXP422 CPU, UC-7408 makes an ideal communication platform for Network Security applications. If your application requires placing UC-7408 in a location that is not located near an Ethernet LAN connection, you can use UC-7408'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-7408 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 Buzzer control, are also included with UC-7408. The Operating System, device drivers, and the software you develop for your own application, can all be stored in UC-7408's Flash memory.

Package Checklist

UC-7408-LX
Data acquisition embedded computer with 8 serial ports, 8 digital input and 8 digital output channels, dual Ethernet, PCMCIA, CompactFlash, Linux OS.

UC-7408 is shipped with the following items:

- UC-7408
- Wall-Mounting Kit
- DIN-Rail Mounting Kit
- UC-7408 Quick Installation Guide
- UC-7408 Documentation & Software CD
- Cross-over Ethernet cable
- CBL-RJ45M9-150: 150 cm, 8-pin RJ45 to Male DB9 serial port cable
- CBL-RJ45F9-150: 150 cm, 8-pin RJ45 to Female DB9 console port cable
- Power Adaptor
- Product Warranty Booklet

NOTE: *Notify your sales representative if any of the above items is missing or damaged.*

Product Features

- Intel XScale IXP422 266 MHz Processor
- On-board 128 MB RAM, 32 MB Flash ROM
- Eight RS-232/422/485 serial ports
- Separate 8-ch digital input and 8-ch digital output
- Dual 10/100 Mbps Ethernet
- PCMCIA/CompactFlash wireless LANexpansion (supports 802.11b/802.11g)
- Linux-ready communication platform
- DIN-Rail or wall mounting installation
- Robust fanless design

Product Hardware Specifications

	UC-7408-LX
CPU	Intel XScale IXP422, 266 MHz
RAM	128 MB
Flash	32 MB
LAN	Auto-sensing 10/100 Mbps x 2
LAN Protection	Built-in 1.5 KV magnetic isolation
DI/DO	8-ch digital input, 8-ch digital output, TTL (3.3V)
Serial Ports	Eight RS-232/422/485 ports
	RS-232 signals: TxD, RxD, DTR, DSR, RTS, CTS, DCD, GND
	RS-422 signals: TxD+, TxD-, RxD+, RxD-, GND
	4 wire RS-485 signals: TxD+, TxD-, RxD+, RxD-, GND
	2 wire RS-485 signals: Data+, Data-, GND
Serial Protection	15 KV ESD for all signals
Data bits	5, 6, 7, 8
Stop bits	1, 1.5, 2
Parity	None, even, odd, space, mark
Flow Control	RTS/CTS, XON/XOFF
Speed	50 bps to 921.6 Kbps
Serial Console/PPP	RS-232 x 1, RJ45
USB 1.1 Client	1
PCMCIA	PCMCIA type I/II socket x 1
Compact Flash	CompactFlash type I/II socket x 1
Real Time Clock	Yes
Buzzer	Yes
LEDs	Serial x 8, Console/PPP x 1, PWR x 1, Ready x 1, LAN 10/100 x 2
Power input	12-48 VDC
Power Consumption	8W
Dimensions	197 x 125 x 44 mm
Gross Weight	870 g
Operating temperature	-10 to 60°C (14 to 140°F), 5 to 95% RH -40 to 75°C (-40 to 167°F) for –T models
Storage temperature	-20 to 80°C (-4 to 185°F), 5 to 95% RH -40 to 85°C (-40 to 185°F) for –T models

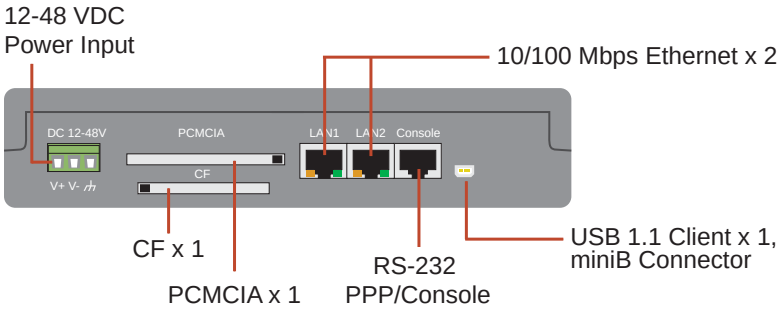
Regulatory Approvals	EMC: FCC Class A, CE Class A Safety: UL, CUL, TÜV
Warranty	5 years

Hardware Introduction

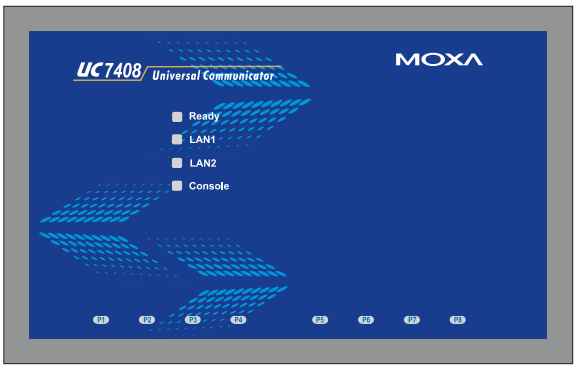
Appearance and Dimensions

Appearance

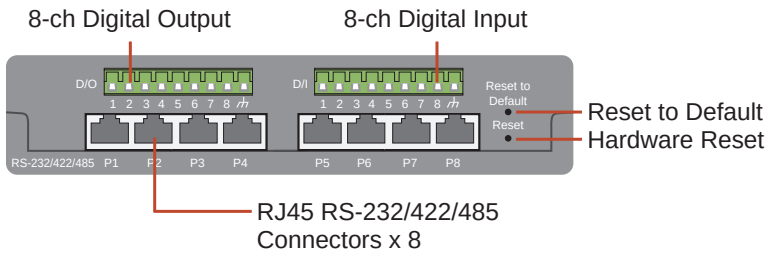
UC-7408 Rear View



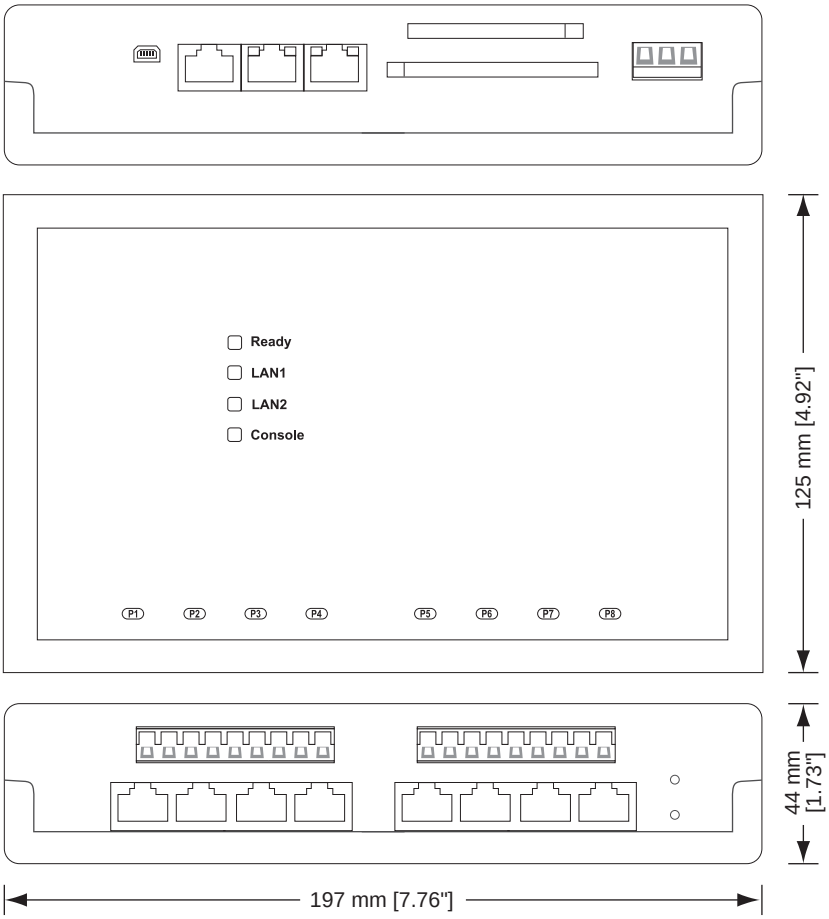
UC-7408 Top View



UC-7408 Front View

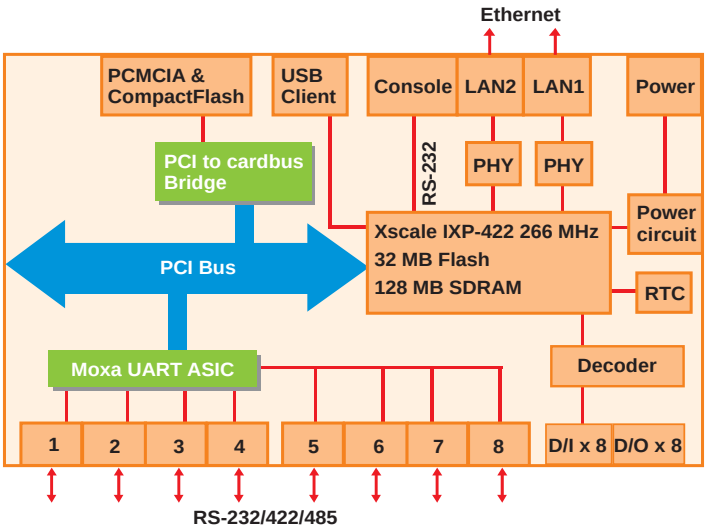


Dimensions



Hardware Block Diagram

The following block diagram shows the layout of UC-7408's internal components.



LED Indicators

UC-7408 has 12 LED indicators on the top panel. Refer to the following table for information about each LED.

LED Name	Color	Meaning
Ready	Green	Power is ON, and system is ready (after booting up)
LAN1, LAN2	Yellow	10 Mbps Ethernet connection
	Green	100 Mbps Ethernet connection
Console	Yellow	Console port is receiving RX data from the serial device.
	Green	Console port is transmitting TX data to the serial device.
P1, P2, P3, P4, P5, P6, P7, P8	Yellow	Serial port is receiving RX data from the serial device.
	Green	Serial port is transmitting TX data to the serial device.

Reset-type Buttons

UC-7408 has two reset-type buttons. The button labeled **Reset** has the same effect as unplugging the power and then plugging the power back in. The button labeled **Reset to default** returns UC-7408 to the factory default parameter configuration.

Reset Button

Pressing the **Reset** button initiates a hardware reboot. The button plays the same role as a desktop PC's reset button.

In normal use, you should NOT use the **Reset** Button. You should only use this function if the software is not working properly. To reset an embedded linux system, always use the software reboot command **/>reboot** to protect the integrity of data being transmitted or processed.

Reset to default Button

Press the **Reset to default** button continuously for at least 5 seconds to load the **factory default configuration**. After the factory default configuration has been loaded, the system will reboot automatically. The **Ready** LED will blink on and off for the first 5 seconds, and then maintain a steady glow once the system has rebooted.

We recommend that you only use this function if the software is not working properly and you want to load factory default settings. To reset an embedded linux system, always use the software reboot command **/>reboot** to protect the integrity of data being transmitted or processed. The **Reset to default** button is not designed to hard reboot UC-7408.

ATTENTION

Reset to default preserves user's data

The **Reset to default** button will NOT format the user directory and erase the user's data. Pressing the Reset to default button will only load the configuration file. All files in the **/etc** directory will revert to their factory defaults, but other User Data will still exist in the Flash ROM.

If you need to load the default System Image file, refer to the "System Image Backup" section in Chapter 3.

Real Time Clock

UC-7408's real time clock is powered by a lithium battery. We strongly recommend that you do not replace the lithium battery without help from a qualified Moxa support engineer. If you need to change the battery, contact Moxa RMA service team.

WARNING

There is a risk of explosion if the battery is replaced by an incorrect type.

Placement Options

Wall or Cabinet

The two metal brackets that come standard with UC-7408 are used to attach UC-7408 to a wall, or the inside of a cabinet. Use two screws per bracket first to attach the brackets to the bottom of the UC-7408 (Fig. A). Next, use two screws per bracket to attach the UC-7408 to a wall or cabinet (Fig. B).

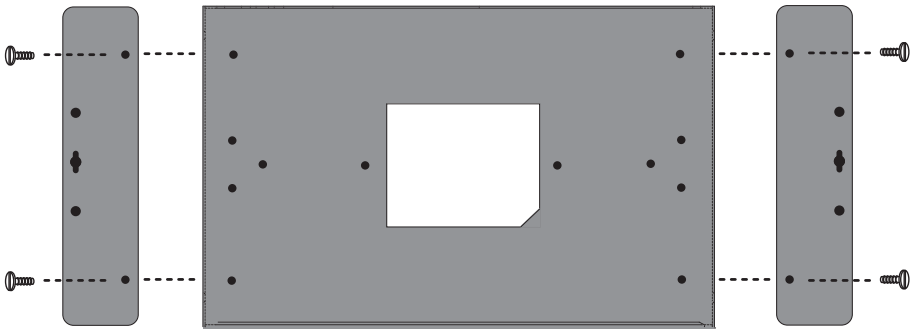


Figure A: UC-7408 Universal Communicator—Wall Mounting Brackets (bottom view)

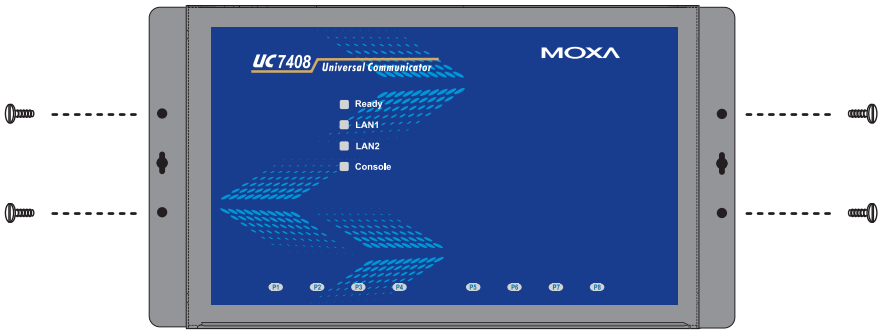
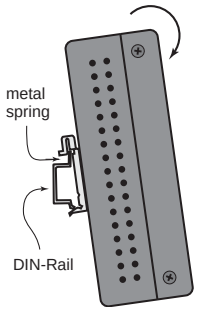


Figure B: UC-7408 Universal Communicator—Wall Mounting Brackets (top view)

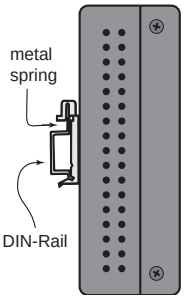
DIN-Rail Mounting

The aluminum DIN-Rail attachment plate is included in the package. If you need to reattach the DIN-Rail attachment plate to UC-7408, make sure the stiff metal spring is situated towards the top, as shown in the figures below.

1. Insert the top of the DIN-Rail into the slot just below the stiff metal spring.



2. The DIN-Rail attachment unit will snap into place as shown below.



To remove UC-7408 from the DIN-Rail, simply reverse Steps 1 and 2 above.

Hardware Connection Description

This section describes how to connect UC-7408 to serial devices for first time testing purposes. We cover **Wiring Requirements**, **Connecting the Power**, **Grounding UC-7408**, **Connecting to the Network**, **Connecting to a Serial Device**, **Connecting to the Console Port**, **PCMCIA**, and **CompactFlash**.

Wiring Requirements

ATTENTION

Safety First!

Be sure to disconnect the power cord before installing and/or wiring your UC-7408.

Wiring Caution!

Calculate the maximum possible current in each power wire and common wire. Observe all electrical codes dictating the maximum current allowable for each wire size.

If the current goes above the maximum ratings, the wiring could overheat, causing serious damage to your equipment.

Temperature Caution!

Be careful when handling UC-7408. When plugged in, UC-7408's internal components generate heat, and consequently the outer casing may feel hot to the touch.

You should also observe the following common wiring rules:

- Use separate paths to route wiring for power and devices. If power wiring and device wiring paths must cross, make sure the wires are perpendicular at the intersection point.

  NOTE: Do not run signal or communication wiring and power wiring in the same wire conduit. To avoid interference, wires with different signal characteristics should be routed separately.

- You can use the type of signal transmitted through a wire to determine which wires should be kept separate. The rule of thumb is that wiring that shares similar electrical characteristics can be bundled together.
- Keep input wiring and output wiring separate.
- Where necessary, it is strongly advised that you label wiring to all devices in the system.

Connecting the Power

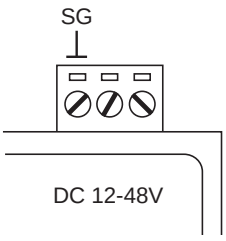
Connect the 12-48 VDC power line with UC-7408's terminal block. If the power is properly supplied, the **Ready** LED will illuminate with a solid green color after 30 to 60 seconds have passed.

Grounding UC-7408

Grounding and wire routing helps limit the effects of noise due to electromagnetic interference (EMI). Run the ground connection from the ground screw to the grounding surface prior to connecting devices.

ATTENTION

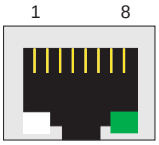
This product is intended to be mounted to a well-grounded mounting surface, such as a metal panel.



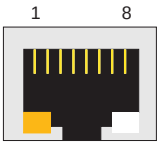
SG: The *Shielded Ground* (sometimes called *Protected Ground*) contact is the left most contact of the 3-pin power terminal block connector when viewed from the angle shown here. Connect the SG wire to an appropriate grounded metal surface.

Connecting to the Network

Connect one end of the Ethernet cable to one of UC-7408's 10/100M Ethernet ports (8-pin RJ45) and the other end of the cable to the Ethernet network. If the cable is properly connected, UC-7408 will indicate a valid connection to the Ethernet in the following ways:



The bottom right corner LED indicator maintains a solid green color when the cable is properly connected to a 100 Mbps Ethernet network. The LED will flash on and off when Ethernet packets are being transmitted or received.

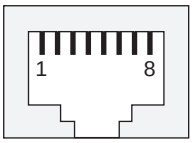


The bottom left corner LED indicator maintains a solid orange color when the cable is properly connected to a 10 Mbps Ethernet network. The LED will flash on and off when Ethernet packets are being transmitted or received.

Pin	Signal
1	ETx+
2	ETx-
3	ERx+
4	---
5	---
6	ERx-
7	---
8	---

Connecting to a Serial Device

Use properly wired serial cables to connect UC-7408 to serial devices. UC-7408's serial ports (P1 to P8) use 8-pin RJ45 connectors. The ports can be configured by software for RS-232, RS-422, or 2-wire RS-485. The precise pin assignments are shown in the following table:



Pin	RS-232	RS-422	RS-485
1	DSR	---	---
2	RTS	TXD+	---
3	GND	GND	GND
4	TXD	TXD-	---
5	RXD	RXD+	Data+
6	DCD	RXD-	Data-
7	CTS	---	---
8	DTR	---	---

Connecting to the Console Port

UC-7408's console port is an 8-pin RJ45 RS-232 port. The port can be used to connect to the console utility from a remote console via a V90 or GPRS modem with PPP protocol. The pin definition is the same as for the serial ports (P1 to P8). For normal data acquisition applications, you should connect to UC-7408's serial ports (P1 to P8) via a V90 or GPRS modem. If you would like to use the console port for normal data acquisition applications, you can set the Console port to startup via PPP protocol. For details, refer to "Dial-up Service—PPP" section in Chapter 4.

PCMCIA

The PCMCIA slot supports the CardBus (Card-32) Card standard and 16-bit (PCMCIA 2.1/JEIDA 4.2) Card standard. It supports +3.3V, +5V, and +12V at a working voltage of 120 mA. Wireless LAN card expansion is optional. The Wireless LAN card provided by Moxa lets you connect UC-7408 to a Wireless LAN, with both 802.1b and 802.11g interfaces supported.

If you need device drivers for other kinds of PCMCIA cards, contact Moxa for information on how to initiate a cooperative development project.

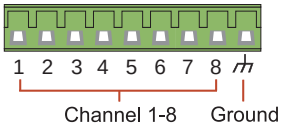
CompactFlash

UC-7408 provides one CompactFlash slot that supports CompactFlash type I/II card expansion. Currently, Moxa provides a CompactFlash disk for plug & play mass storage expansion. You may also use flash disks available from most computer supply outlets. The CompactFlash will be mounted at **/mnt/hda**

If you need device drivers for other kinds of mass storage cards, contact Moxa for information on how to initiate a cooperative development project.

DI/DO

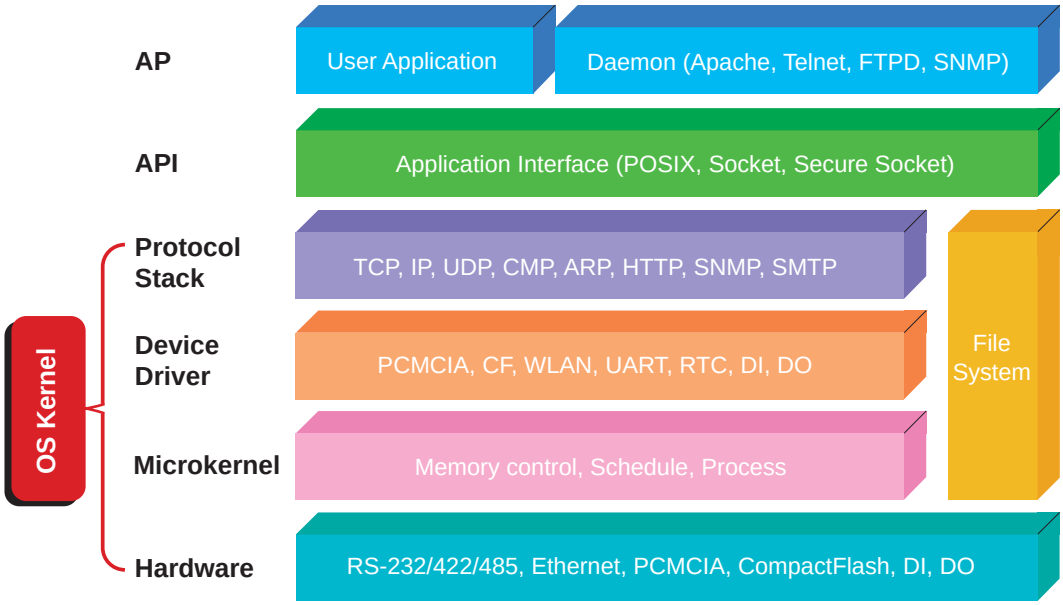
The eight digital input channels and eight digital output channels use separate terminal blocks.



Software Introduction

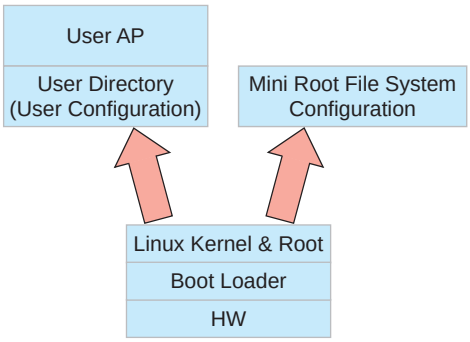
Software Architecture

The Linux operating system that is pre-installed in UC-7408 follows the standard Linux architecture, making it easy to port programs that follow the POSIX standard to UC-7408. 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-7408 Linux system.



UC-7408's Flash ROM is partitioned into **Boot Loader**, **Linux Kernel, Mini Root File System,** and **User Root File System** partitions.

In order to prevent user applications from crashing the Root File System, UC-7408 uses a specially designed **Mini File System with Protected Configuration** for emergency use. This **Mini File System** comes with serial and Ethernet communication capability for users to load the **Factory Default Image** file. The Mini File System will only be activated if the boot loader fails to load the User Root File System.



To improve system reliability, UC-7408 has a built-in mechanism that prevents the system from crashing. The procedure is as follows.

When the Linux kernel boots up, the kernel will mount the root file system, 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 User Root File System to boot up the system. The Mini Root File System is protected, and cannot be changed by the user, providing a "safe" zone. The kernel will only use the Mini Root File System when the User Root File System crashes.

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

Software Package

Boot Loader	Redboot (V1.92)
Kernel	MontaVista embedded Linux 2.4.18
Protocol Stack	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 (V3.3.2)	C/C++ PC Cross Compiler
GDB (V5.2.1)	Source Level Debug Server
Glibc (V2.2.5)	POSIX standard C library

2

Getting Started

In this chapter, we explain how to connect UC-7408, turn on the power, and then get started using the programming and other functions.

The following topics are covered in this chapter:

- **Powering on UC-7408**
- **Connecting UC-7408 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
- **Configuring the WLAN via the PCMCIA Interface**
  - IEEE802.11b
  - IEEE802.11g
- **Test Program—Developing Hello.c**
  - Installing the Tool Chain (Linux)
  - Checking the Flash Memory Space
  - Compiling Hello.c
  - Uploading “Hello” to UC-7408 and Running the Program
- **Developing Your First Application**
  - Testing Environment
  - Compiling tcps2.c
  - Uploading tcps2-release and Running the Program
  - Testing Procedure Summary

Powering on UC-7408

Connect the SG wire to the Shielded Contact located in the upper left corner of the UC-7408, and then power on UC-7408 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 Network address settings will appear on the LCM display.

NOTE After connecting UC-7408 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 UC-7408 to a PC

There are two ways to connect UC-7408 to a PC: through the serial Console port or via Telnet over the network.

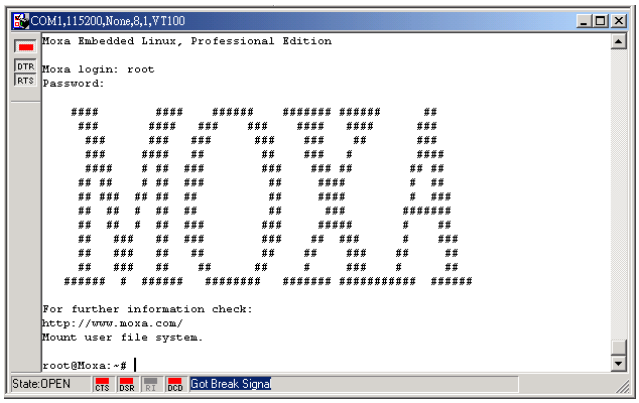
Serial Console

The serial console port gives users a convenient way of connecting to UC-7408's console utility. This method is particularly useful when using UC-7408 for the first time. The signal is transmitted over a direct serial connection, so you do not need to know either of UC-7408's two IP addresses in order to connect to the serial console utility.

Use the serial console port settings shown below.

Baud rate	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.



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

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

Telnet Console

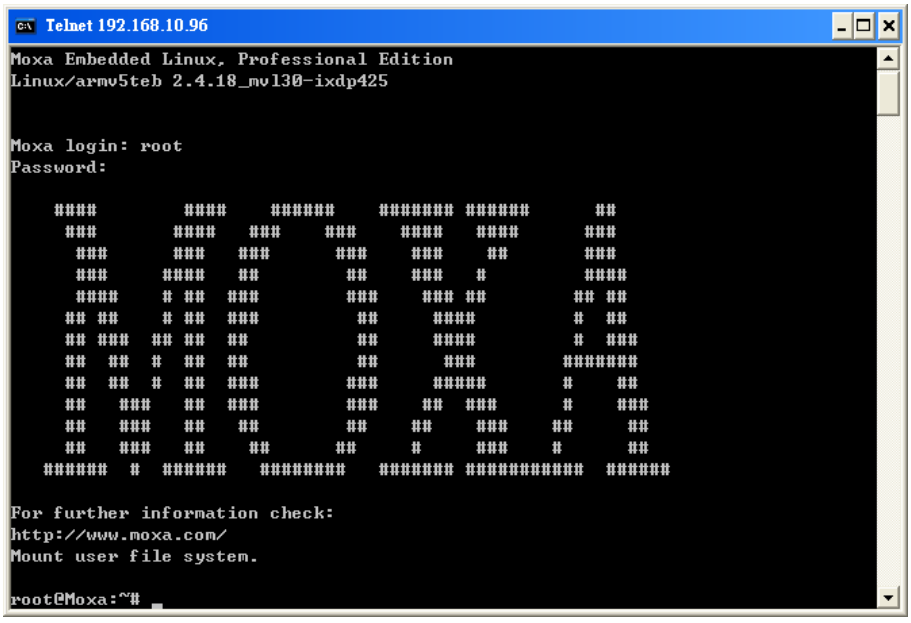
If you know at least one of the two IP addresses and netmasks, then you can use Telnet to connect to UC-7408's console utility. The default IP address and Netmask for each of the two ports are given below:

	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 UC-7408. You should first modify your PC's IP address and netmask so that your PC is on the same subnet as one of UC-7408's two 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 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 a hub or switch connected to your local LAN, use a straight-through Ethernet cable. The default IP addresses and netmasks are shown above. To login, type the Login name and password as requested. The default values are both **root**:

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



You can proceed with the configuration of UC-7408's network settings 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 cable CBL-RJ45F9-150, which comes with UC-7408, to connect to the serial console port.

Telnet Reminder

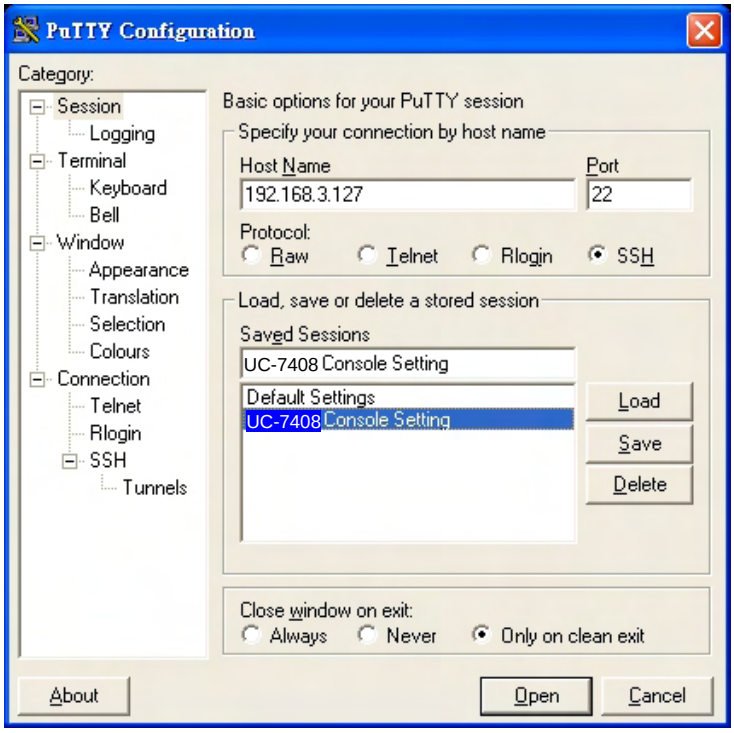
When connecting to UC-7408 over a LAN, you must configure your PC's Ethernet IP address to be on the same subnet as the UC-7408 you wish to contact. If you do not get connected on the first try, re-check the serial and IP settings, and then unplug and re-plug UC-7408's power cord.

SSH Console

UC-7408 supports an SSH Console to offer 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 UC-7408 in a Windows environment. The following figure shows a simple example of the configuration that is required.



Linux Users

From a Linux machine, use the "ssh" command to access UC-7408's Console utility via SSH.

```
#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 UC-7408's Console utility over the network.

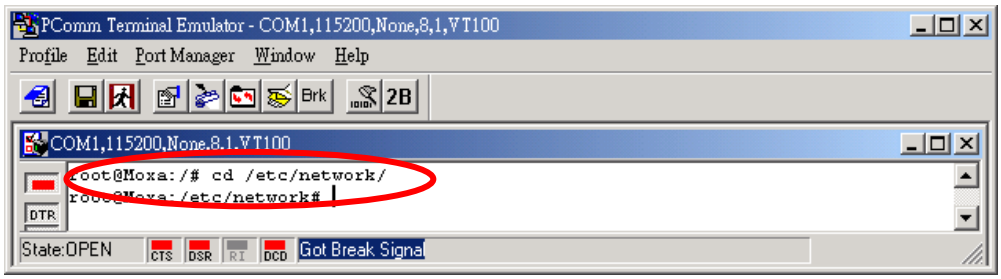
Configuring the Ethernet Interface

UC-7408's network settings can be modified with the serial Console, or online over the network.

Modifying Network Settings with the Serial Console

In this section, we use the serial console to modify UC-7408's network settings.

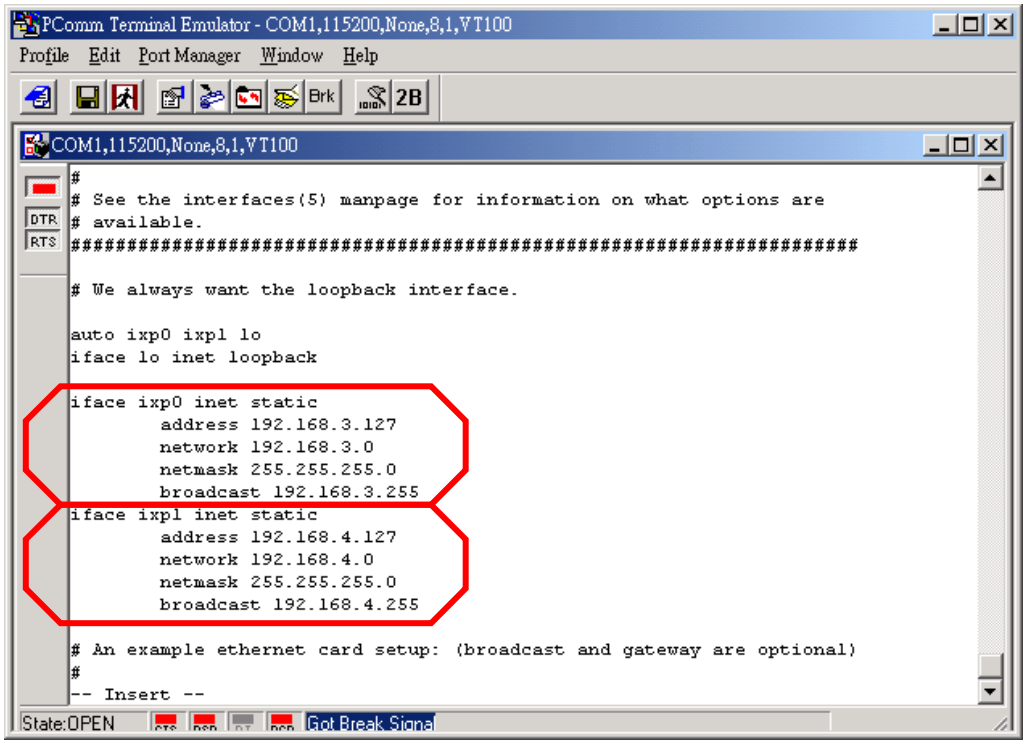
1. Follow the instructions given in a previous section to access UC-7408's Console Utility via the serial Console port, and then type **`#cd /etc/network`** to change directories.



2. Type **`#vi interfaces`** to edit the network configuration file with vi editor. You can configure UC-7408's Ethernet ports for **static** or **dynamic** (DHCP) IP addresses.

   Static IP addresses:

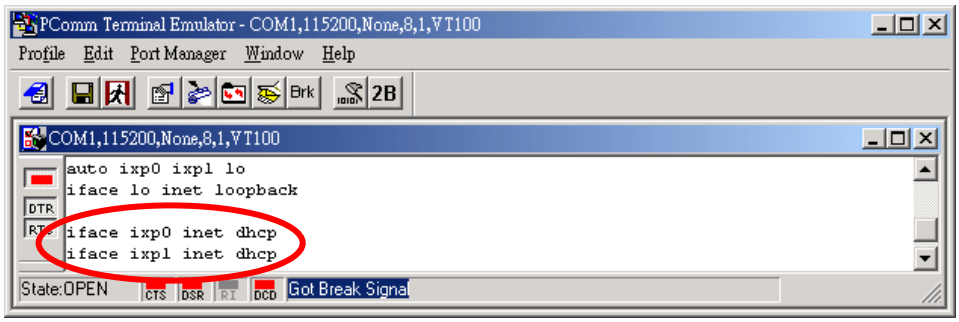
   As shown below, 4 network addresses need to be modified: **address**, **network**, **netmask**, and **broadcast**. The default IP addresses are 192.168.3.127 for LAN1 and 192.168.4.127 for LAN2, with default netmask of 255.255.255.0.



Dynamic IP addresses:

By default, UC-7408 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 ixp0 inet **static** address 192.168.3.127 network: 192.168.3.0 netmask 255.255.255.0 broadcast 192.168.3.255	iface ixp0 inet **dhcp**



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

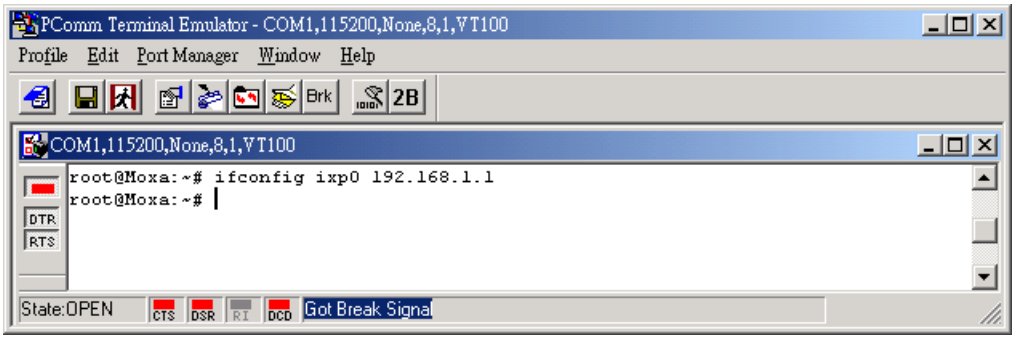
   `#/etc/init.d/networking restart`

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

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 ixp0 192.168.1.1`** to change the IP address of LAN1 to 192.168.1.1.



Configuring the WLAN via the PCMCIA Interface

IEEE802.11b

The following IEEE802.11b wireless modules are supported:

- NDC NWH1010
- Senao NL-2511CD PLUS(F200)
- Senao NL-2511CD PLUS EXT2 MERCURY (ETSI)
- Senao NI3-2511CD-PLUS3
- DARK DKW11-330HP
- DARK XI-330H
- Planex (PCI) GW-NS11H
- Corega CG-WLPCCL-11

To configure the WLAN for IEEE802.11b:

1. Unplug the PCMCIA Wireless LAN card first.
2. Configure the Wireless LAN card's default IP setting profile.

   (Default IP address is 192.168.5.127, netmask 255.255.255.0)

   Edit **network.opts** with the following command to edit Wireless LAN's default setting.

   #vi /etc/pcmcia/network.opts

```
# Use DHCP (via /sbin/dhcpcd, /sbin/dhclient, or /sbin/pump)? [y/n]
DHCP="n"
# If you need to explicitly specify a hostname for DHCP requests
DHCP_HOSTNAME=""
# Host's IP address, netmask, network address, broadcast address
IPADDR="192.168.5.127"
NETMASK="255.255.255.0"     arport.opts    wireless
NETWORK="192.168.5.0"
BROADCAST="192.168.5.255"  serial.opts     wlan-ng
# Gateway address for static routing      wlan-ng.conf
#GATEWAY="10.0.1.1"    /etc/pcmcia/network.opts
"/etc/pcmcia/network.opts" line 1 of 48 --2%--
```

3. Configure the Wireless LAN card's default SSID setting profile.

   (Default SSID is "any")

   #vi /etc/wlan/wlan.conf

```
#         /etc/wlancfg/wlancfg-DEFAULT are used.
#
# for example:
#     SSID_wlan0="linux-wlan"
# This expects a file called "/etc/wlan/wlancfg-linux-wlan" to be present.
#
# Use a SSID of "" to associate with any network in range.
#########

SSID_wlan0="any"
ENABLE_wlan0=y
#SSID_wlan1=""
#ENABLE_wlan1=n
#SSID_wlan2=""
#ENABLE wlan2=n
```

   // Consult your network administrator for SSID required in your wireless network. For example, *SSID_waln0="any", Enable_wlan0=y*//

4. Duplicate the configuration profile to a new profile.

   #cp /etc/wlan/wlancfg-DEFAULT /etc/wlan/wlancfg-any

   // Copy configuration profile "DEFAULT" to new configuration profile "any"//

5. Configure the WEP setting, if WEP is required on your wireless network.

   #vi /etc/wlan/wlancfg-any

```
#=======WEP=============================================
# [Dis/En]able WEP.  Settings only matter if PrivacyInvoked is true
lnxreq_hostWEPEncrypt=false     # true|false
lnxreq_hostWEPDecrypt=false     # true|false
dot11PrivacyInvoked=false       # true|false
dot11WEPDefaultKeyID=0          # 0|1|2|3
dot11ExcludeUnencrypted=false   # true|false, in AP this means WEP is required.

# If PRIV_GENSTR is not empty, use PRIV_GENTSTR to generate
#  keys (just a convenience)
PRIV_GENERATOR=/sbin/nwepgen    # nwepgen, Neesus compatible
PRIV_KEY128=false               # keylength to generate
PRIV_GENSTR=""

# or set them explicitly.  Set genstr or keys, not both.
dot11WEPDefaultKey0=            # format: xx:xx:xx:xx:xx   or
dot11WEPDefaultKey1=            #         xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx
dot11WEPDefaultKey2=            #  e.g.   01:20:03:40:05   or
dot11WEPDefaultKey3=            #         01:02:03:04:05:06:07:08:09:0a:0b:0c:0d
#=======SELECT STATION MODE==================
IS_ADHOC=n                      # y|n, y - adhoc, n - infrastructure
```

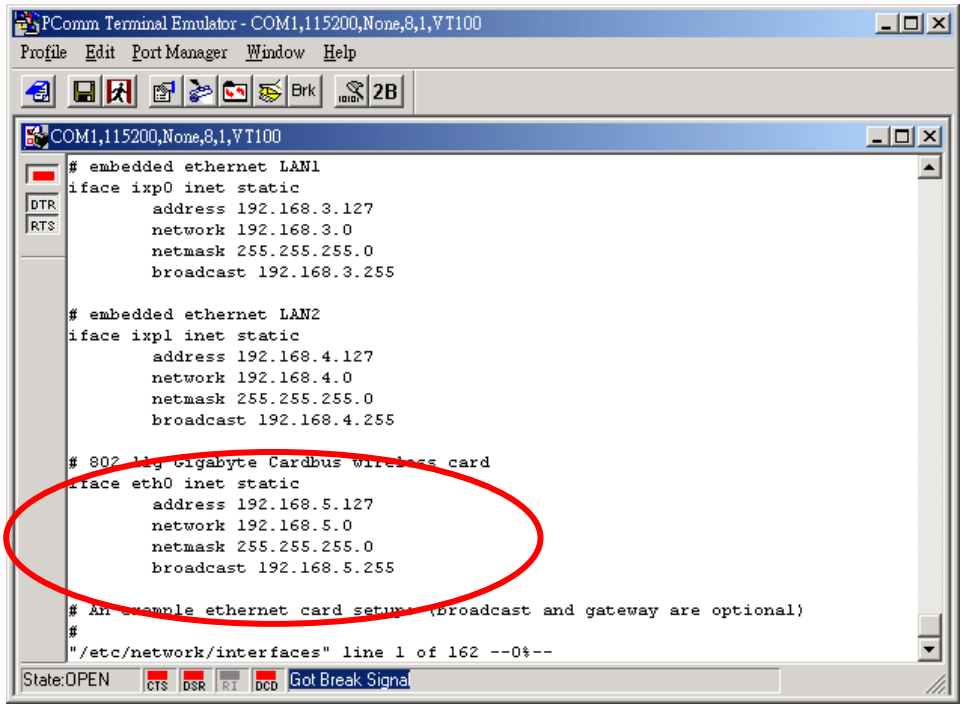
IEEE802.11g

The following IEEE802.11g wireless modules are supported:

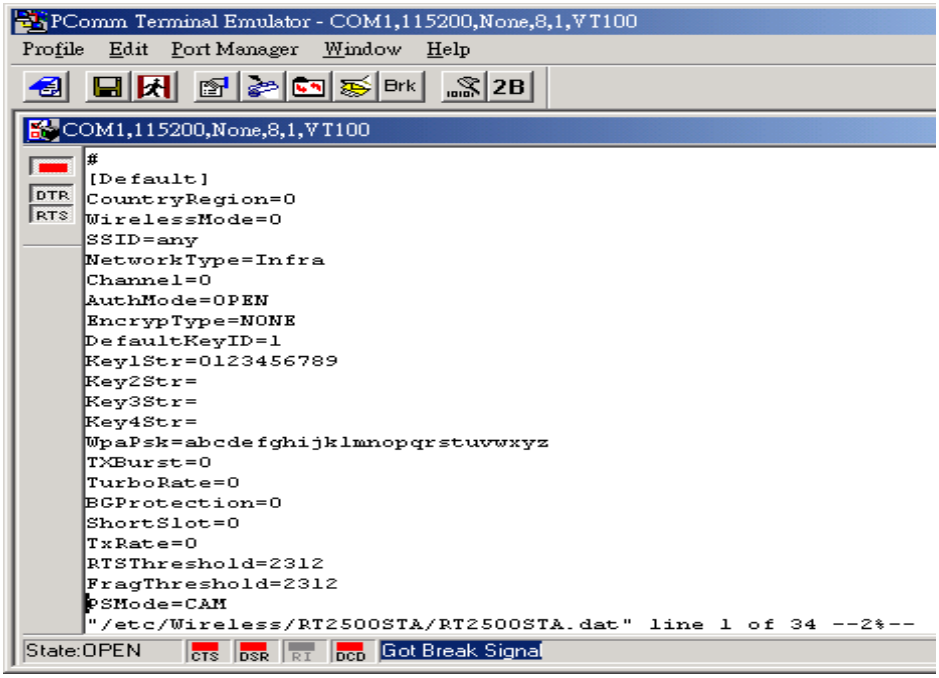
- ASUS—WL-107g
- CNET—CWC-854 (181D version)
- Edmiax—EW-7108PCg
- Amigo—AWP-914W
- GigaByte—GN-WMGK
- Other brands that use the Ralink RT2560 series chip set

To configure the WLAN for IEEE802.11g:

1. Unplug the CardBus Wireless LAN card first.
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 (circled in red in the following figure).



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 following command **`#vi /etc/Wireless/RT2500STA/RT2500STA.dat`** to edit the file with vi editor. Setting options for the various parameters are listed below the figure.



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	Explanation
0	disable
1	enable

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

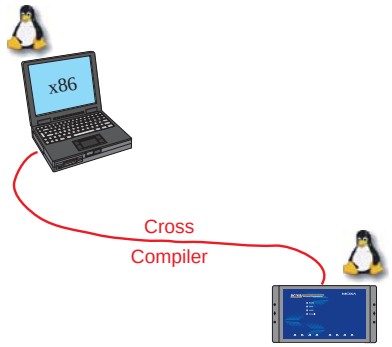
Setting
1 to 2347

FragThreshold—Sets the fragment threshold

Setting
256 to 2346

Test Program—Developing Hello.c

In this section, we use the standard "Hello" programming example to illustrate how to develop a program for UC-7408. In general, program development involves the following seven steps.



Step 1:
Connect UC-7408 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:
Code and compile the program.
Step 5:
Download the program to UC-7408 Via 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-7408 units if needed).

Installing the Tool Chain (Linux)

The PC must have the Linux Operating System pre-installed before installing the UC-7408 GNU Tool Chain. Redhat 7.3/8.0, Fedora core, and compatible versions are recommended. The Tool Chain requires about 100 MB of hard disk space on your PC. The UC-7408 Tool Chain software is located on the UC-7408 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount /dev/cdrom /mnt/cdrom
#rpm –ivh /mnt/cdrom/mxscaleb-3.3.2-6.i386.rpm
```

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 ToolChain files, including the compiler, link, library, and include files are located in this directory.

```
PATH=/usr/local/mxscaleb/bin:$PATH
```

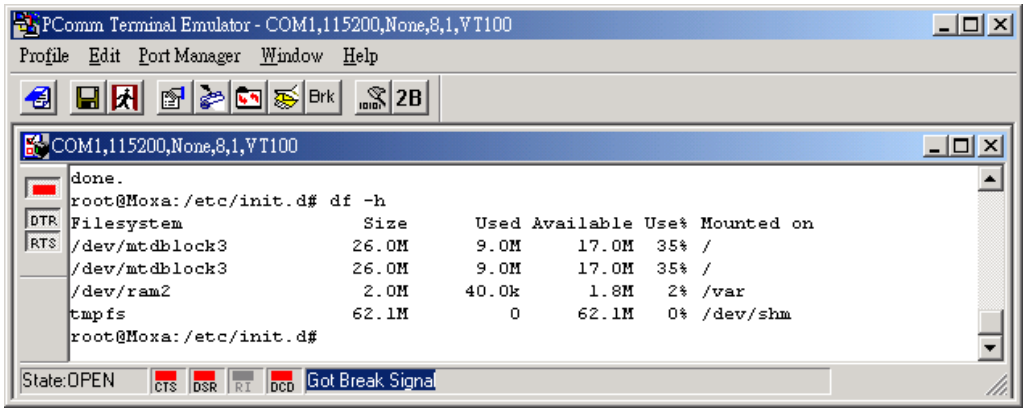
Setting the path allows you to run the compiler from any directory.

NOTE Refer to Appendix B for an introduction to the Windows Tool Chain. In this chapter, we use the Linux tool chain to illustrate the cross compiling process.

Checking the Flash Memory Space

If the flash memory 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
```



If there isn't enough "Available" space for your application, you will need to delete some existing files. To do this, connect your PC to the UC-7408 with the console cable, and then use the console utility to delete the files from UC-7408's flash memory.

NOTE If the flash memory is full, you will need to free up some memory space before saving files to the Flash ROM.

Compiling Hello.c

The UC-7408 CD 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
/usr/local/mxscaleb/bin/mxscaleb-gcc –o hello-release hello.c
/usr/local/mxscaleb/bin/mxscaleb-strip –s hello-release
/usr/local/mxscaleb/bin/mxscaleb-gcc –ggdb -o hello-debug hello.c
[root@localhost hello]# _
```

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

hello-release—an IXP platform execution file (created specifically to run on UC-7408)

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 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 "Hello" to UC-7408 and Running the Program

Use the following command to upload **hello-release** to the UC-7408 via FTP.

1. From the PC, type:

```
#ftp 192.168.3.127
```

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

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

3. From the UC-7408, type:

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

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

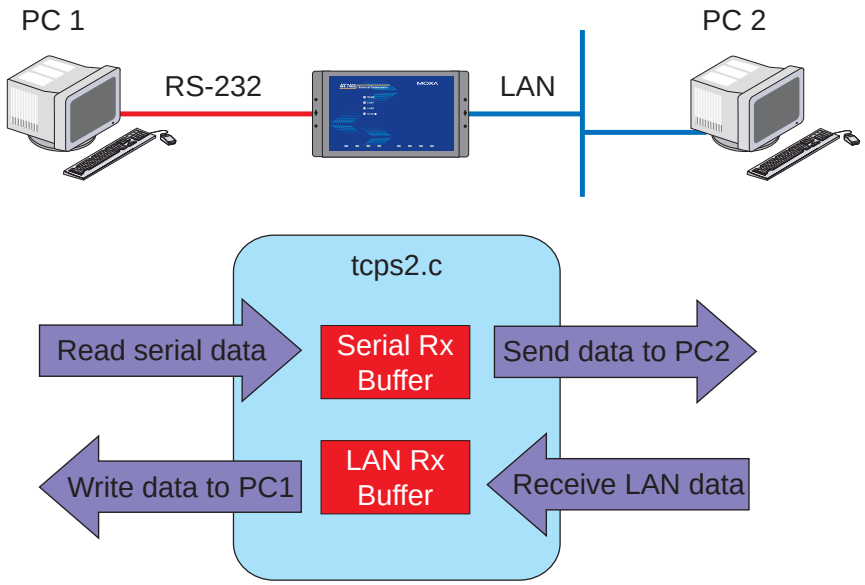
```
root@Moxa:~# ./hello-release
Hello
```

Developing Your First Application

We use the tcps2 example to illustrate how to build an application for UC-7408. The procedure outlined in the following subsections will show you how to build a TCP Server program plus serial port communication that runs on the UC-7408.

Testing Environment

The tcps2 example demonstrates a simple application program that delivers transparent, bi-directional data transmission between UC-7408's 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-7408 via an RS-232 connection. At the remote site, data can be transferred between UC-7408's Ethernet port and PC 2 over an Ethernet connection.



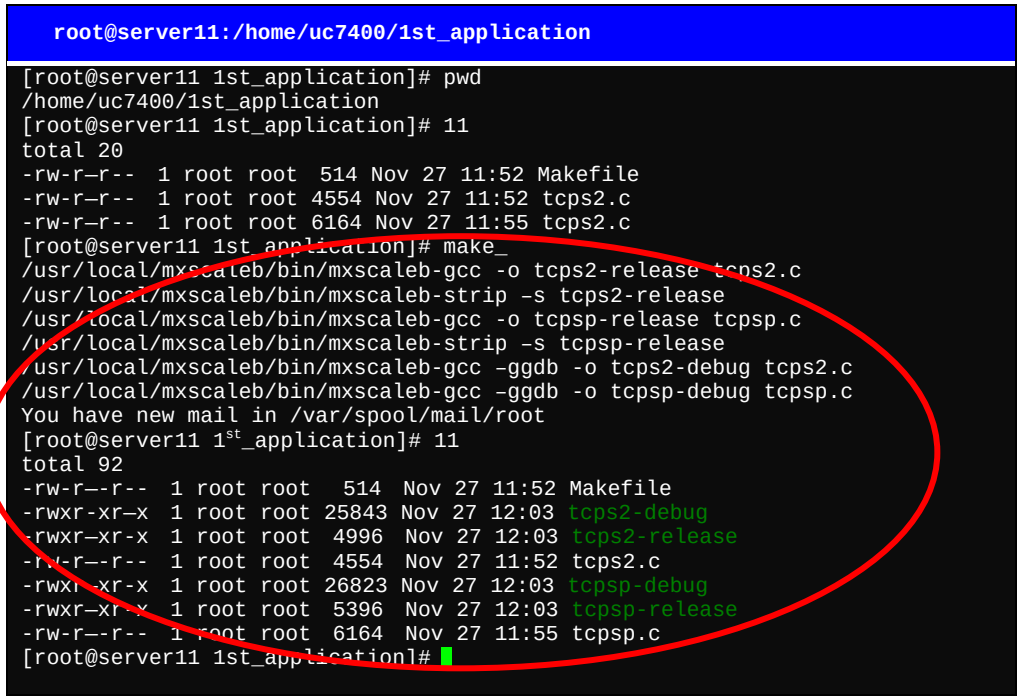
Compiling tcps2.c

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/uc7400/1st_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/uc7400/1st_application/tcps2.c
#cp /mnt/cdrom/example/TCPServer2/tcpsp.c/home/uc7400/1st_application/tcpsp.c
#cp /mnt/cdrom/example/TCPServer2/Makefile.c/home/uc7400/1st_application/Makefile.c
```

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

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



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

tcps2-release—an IXP platform execution file (created specifically to run on UC-7408)

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 tcps2-release and Running the Program

Use the following commands to use FTP to upload **tcps2-release** to the UC-7408.

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/uc7400/1st_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-7408, 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:~#
```

NOTE Use the **kill** command for job number 1 to terminate this program: **#kill %1**

#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 ixp0]
  36  root           D    [ixp425 ixp1]
  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
```

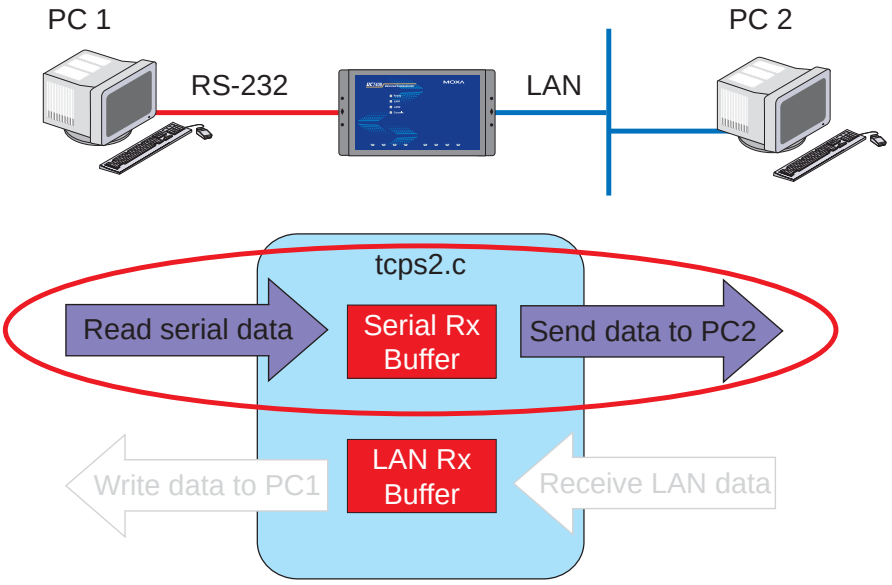
```
  158  root      1532 S    /sbin/getty 115200 ttyS1
  162  root      3652 S    /usr/sbin/sshd
  163  root      2208 S    -bash
  169  root      2192 S    ftpd: 192.168.3.110: root: IDLE
  187  root      1264 S    ./tcps2-release
  188  root      1592 S    ps -ef
root@Moxa:~#
```

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 UC-7408's serial port 1.
5. Use an Ethernet cable to connect PC2 to UC-7408.
6. On PC1: If running Windows, use HyperTerminal (**38400, 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-7408 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**
- **Setting the Run-Level**
- **Adjusting the System Time**
  - Setting the Time Manually
  - NTP Client
  - Updating the Time Automatically
- **Cron—daemon to Execute Scheduled Commands**
- **Connecting Peripherals**
  - CF Mass Storage

System Version Information

To determine the hardware capability of your UC-7408, and what kind of software functions are supported, check the version numbers of your UC-7408's hardware, kernel, and user file system. Contact Moxa to determine the hardware version. You will need the **Production S/N** (Serial number), which is located on UC-7408's bottom label.

To check the kernel version, type:
#kversion

To check the user file system version, type:
#fsversion

NOTE The kernel version and user file system version numbers are the same for the factory default configuration, and if you download the latest firmware version from Moxa's website and then upgrade UC-7408's hardware, the two version numbers will be the same.

However, to help users define the user file system, the kernel and user file system are separate, and hence could have different version numbers. For this reason, we provide two utilities, called **kversion** and **fsversion**, that allow you to check the version numbers of the kernel and file system, respectively.

System Image Backup

Upgrading the Firmware

UC-7408's bios, kernel, mini file system, 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 **uc7408-x.x.x.frm**, with "x.x.x" indicating the firmware version. To upgrade the firmware, download the firmware file to a PC, and then transfer the file to the UC-7408 unit via a serial Console or Telnet Console connection.

ATTENTION

Upgrading the firmware will erase all data on the Flash ROM

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

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

```
192.168.3.127 – PuTTY
root@Moxa:~# df -h
Filesystem    Size      Used Available Use% Mounted on
/dev/mtdblock3  26.0M   8.9M     17.1M  34%  /
/dev/mtdblock3  26.0M   8.9M     17.1M  34%  /
/dev/ram2     2.0M     40.0k   1.8M   2%  /var
tmpfs      62.1M       0  62.1M   0%  /dev/shm
root@Moxa:~# upramdisk
root@Moxa:~# df -h
Filesystem    Size      Used Available Use% Mounted on
/dev/mtdblock3  26.0M   8.9M     17.1M  34%  /
/dev/mtdblock3  26.0M   8.9M     17.1M  34%  /
/dev/ram2     2.0M     40.0k   1.8M   2%  /var
tmpfs      62.1M       0  62.1M   0%  /dev/shm
/dev/ram1    29.0M  13.0k  27.5M   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 UC-7408's 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 UC-7408's built-in FTP client to transfer the firmware file (**uc7408-x.x.x.frm**) from the PC to UC-7408:

   /mnt/ramdisk> ftp <destination PC's IP>
   Login Name: xxxx
   Login Password: xxxx
   ftp> bin
   ftp> get uc7408-x.x.x.frm

```
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    13167772 Nov 29 10:24 UC7420-1.5.frm
-rw-rw-rw-    1 ftp  ftp     8778996 Nov 29 10:24 UC7420_usrdisk-1.5.frm
226 Transfer complete.
ftp> get UC7420-1.5.frm
```

```
local: UC7420-1.5.frm remote: UC7420-1.5.frm
200 Port command successful.
150 Opening data connection for UC7420-1.5.frm
226 Transfer complete.
13167772 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 uc7408-x.x.x.frm

```
192.168.3.127 – PuTTY
root@Moxa:/mnt/ramdisk# upfirm UC7420-1.5.frm
Upgrade firmware utility version 1.0.
To check source firmware file context.
The source firmware file conext is OK.
This step will destroy all your firmware.
Do you want to continue it ? (Y/N) : Y
Now upgrade the file [redboot].
Format MTD device [/dev/mtd0] . . .
MTD device [/dev/mtd0] erase 128 Kibyte @ 60000 – 100% complete.
Wait to write file . . .
Compleleted 100%
Now upgrade the file [kernel].
Format MTD device [/dev/mtd1] . . .
MTD device [/dev/mtd1] erase 128 Kibyte @ 100000 – 100% complete.
Wait to write file . . .
Compleleted 100%
Now upgrade the file [mini-file-system].
Format MTD device [/dev/mtd2] . . .
MTD device [/dev/mtd2] erase 128 Kibyte @ 400000 – 100% complete.
Wait to write file . . .
Compleleted 100%
Now upgrade the file [user-file-system].
Format MTD device [/dev/mtd3] . . .
MTD device [/dev/mtd3] erase 128 Kibyte @ 1a00000 – 100% complete.
Wait to write file . . .
Compleleted 100%
Now upgrade the file [directory].
Format MTD device [/dev/mtd6] . . .
MTD device [/dev/mtd6] erase 128 Kibyte @ 20000 – 100% complete.
Wait to write file . . .
Compleleted 100%
Now upgrade the new configuration file.
Upgrade the firmware is OK.
Please press any key to reboot system.
```

ATTENTION

After you reboot your UC, DO NOT power off the UC until the Ready LED comes back ON. Note that after upgrading the firmware, the first boot up will take as much as 2 to 3 minutes.

Loading Factory Defaults

The easiest way to load factory defaults is to update the firmware (follow the instructions in the previous section to upgrade the firmware).

Note that if your user file is not working properly, the system will mount the Mini File System. In this case, you will need to load factory defaults to resume normal operation.

Enabling and Disabling Daemons

The following daemons are enabled when UC-7408 boots up for the first time.

snmpd .....SNMP Agent daemon
telnetd.....Telnet Server / Client daemon
inetd .....Internet Daemons
ftpd.....FTP Server / Client daemon
sshd .....Secure Shell Server daemon
httpd .....Apache WWW Server daemon
nfsd .....Network File System Server daemon

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

```
192.168.3.127 – PuTTY
root@Moxa:~# cd /etc
root@Moxa:/etc# 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 ixp0]
   38  root      1256 S    stdef
   36  root           S    [ixp425 ixp1]
   47  root      1368 S    /usr/sbin/inetd
   53  root      4464 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
   67  nobody    4480 S    /usr/sbin/httpd
   92  bin       1460 S    /sbin/portmap
  104  root      1556 S    /usr/sbin/rpc.statd
  108  root      4044 S    /usr/sbin/snmpd –s –l /dev/null
  110  root      2828 S    /usr/sbin/snmptrapd -s
  139  root      1364 S    /sbin/cardmgr
  143  root      1756 S    /usr/sbin/rpc.nfsd
  145  root      1780 S    /usr/sbin/rpc.mountd
  152  root      2960 S    /usr/sbin/sshd
  160  root      1272 S    /bin/reportip
  161  root      3464 S    /bin/massupfirm
  162  root      1532 S    /sbin/getty 115200 ttyS01
  163  root      1532 S    /sbin/getty 115200 ttyS1
  166  root      3464 S    /bin/massupfirm
  167  root      3464 S    /bin/massupfirm
  170  root      3652 S    /usr/sbin/sshd
  171  root      2196 S    –bash
  182  root      1592 S    ps -ef
root@Moxa:/ect#
```

To run a private daemon, you can edit the file rc.local, as follows:

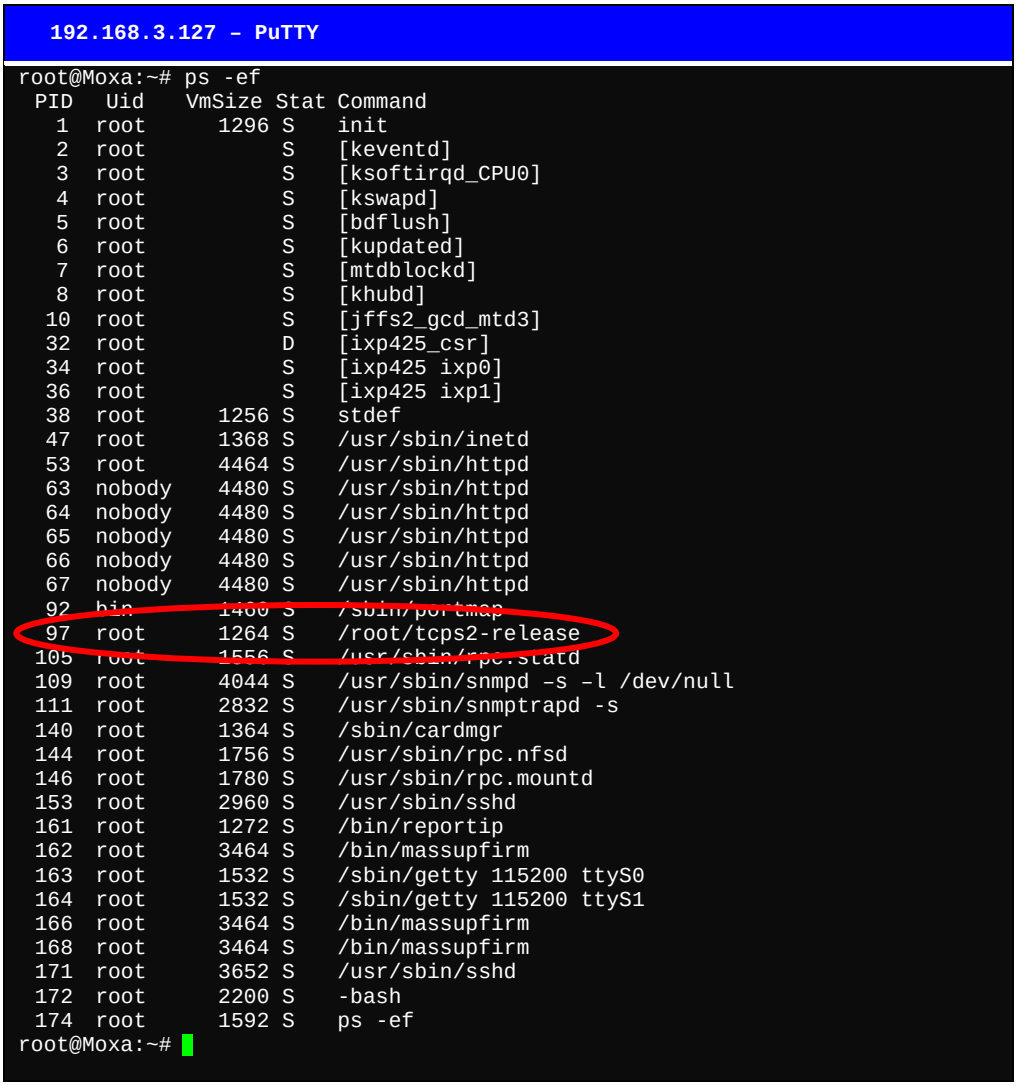
#cd /etc/rc.d
#vi rc.local

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

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

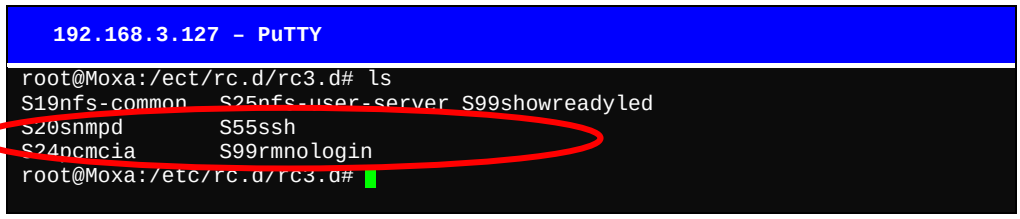
```
192.168.3.127 – PuTTY
# !/bin/sh
# Add you want to run daemon
/root/tcps2-release &~
```

Then you will find the enabled daemons after you reboot the system.



Setting the Run-Level

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



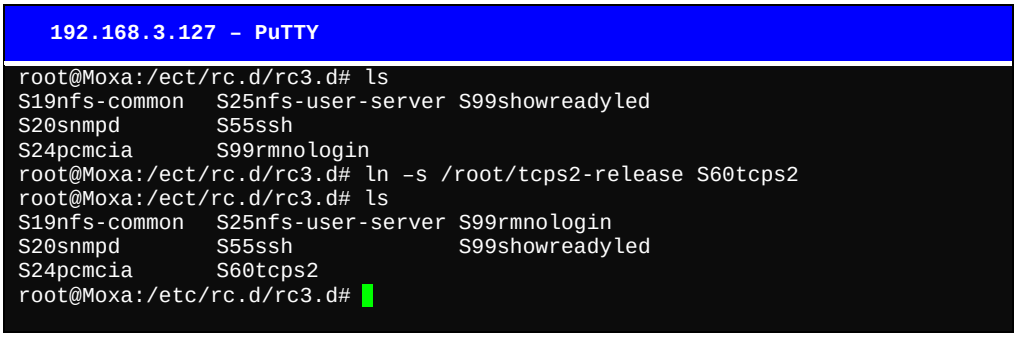
#cd /etc/rc.d/init.d

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

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

SxxRUNFILE stands for

S: start the run file while linux boots up.
xx: a number between 00-99. The smaller number has a higher priority.
RUNFILE: the file name.



KxxRUNFILE stands for

K: start the run file while linux shuts down or halts.
xx: a number between 00-99. The smaller number has a higher priority.
RUNFILE: is the file name.

For removing the daemon, you can remove the run file from **/etc/rc.d/rc3.d** by using the following command:

#rm –f /etc/rc.d/rc3.d/S60tcps2

Adjusting the System Time

Setting the Time Manually

UC-7408 has two time settings. One is the system time, and the other is the RTC (Real Time Clock) time kept by the UC-7408 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

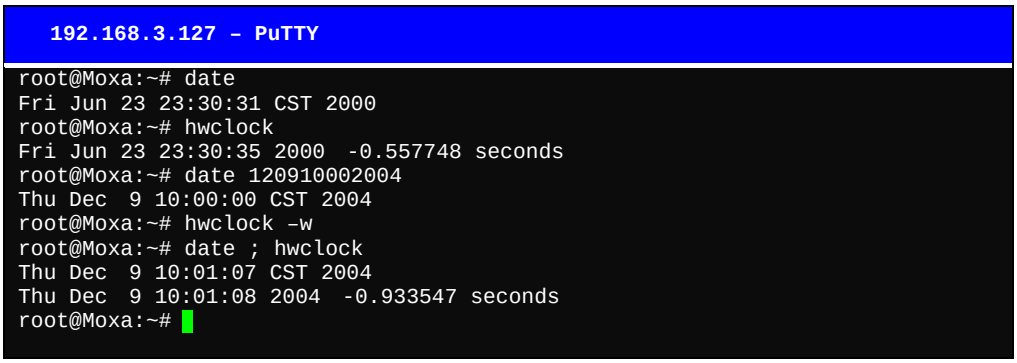
Use the following command to set the system time:
#date MMDDhhmmYYYY

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 thesystem time and set the RTC time.

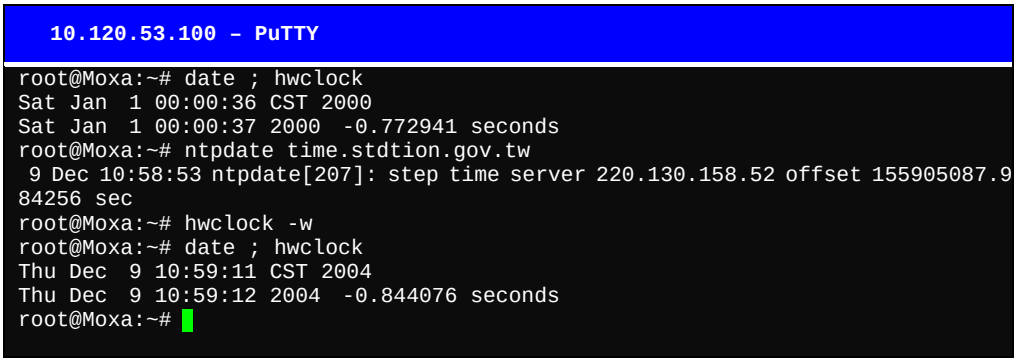


NTP Client

UC-7408 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.



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 setting information.

Updating the Time Automatically

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

Example shell script to update the system time periodically

```
#!/bin/sh
ntpdate time.nist.gov   # You can use the time server's ip address or domain
                        # name directly. If you use domain name, you must
                        # enable the domain client on the system by updating
                        # /etc/resolv.conf file.
hwclock –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 to Execute Scheduled Commands

Start Cron from the directory **/etc/rc.d/rc.local**. It will return immediately, so you don't need to start it with '&' to run the background.

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

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

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

mm	h	dom	mon	dow	user	command
month	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 mode of fixtime.sh

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

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

Add the following line to the end of crontab:

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

STEP4: Enable the cron daemon manually.

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

STEP5: Enable cron when the system boots up.

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

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

Connecting Peripherals

CF Mass Storage

The UC-7408 supports PNP and hot pluggability for connecting a CF mass storage device. UC-7408 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**. UC-7408 will be un-mounted automatically by **umount** when you disconnect it.

4

Managing Communications

In this chapter, we explain how to configure UC-7408's various communication functions.

The following topics are covered in this chapter:

- **Telnet / FTP**
- **DNS**
- **Web Service—Apache**
  - Saving a Web Page to the CF Card
- **IPTABLES**
- **NAT**
  - NAT Example
  - Enabling NAT at Bootup
- **Dial-up Service—PPP**
- **PPPoE**
- **NFS (Network File System)**
  - Setting up UC-7408 as an NFS Server
  - Setting up UC-7408 as an NFS Client
- **Mail**
- **SNMP**
- **Open VPN**

Telnet / FTP

In addition to supporting Telnet client/server and FTP client/server, the UC-7408 system 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

UC-7408 supports DNS client (but not DNS server). To set up DNS client, you need to edit three configuration files: **/etc/hosts**, **/etc/resolv.conf**, and **/etc/nsswitch.conf**.

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

/etc/resolv.conf
This is the most important file that you need to edit when using DNS for the other programs. For example, before you 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
```

/etc/nsswitch.conf

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

Web Service—Apache

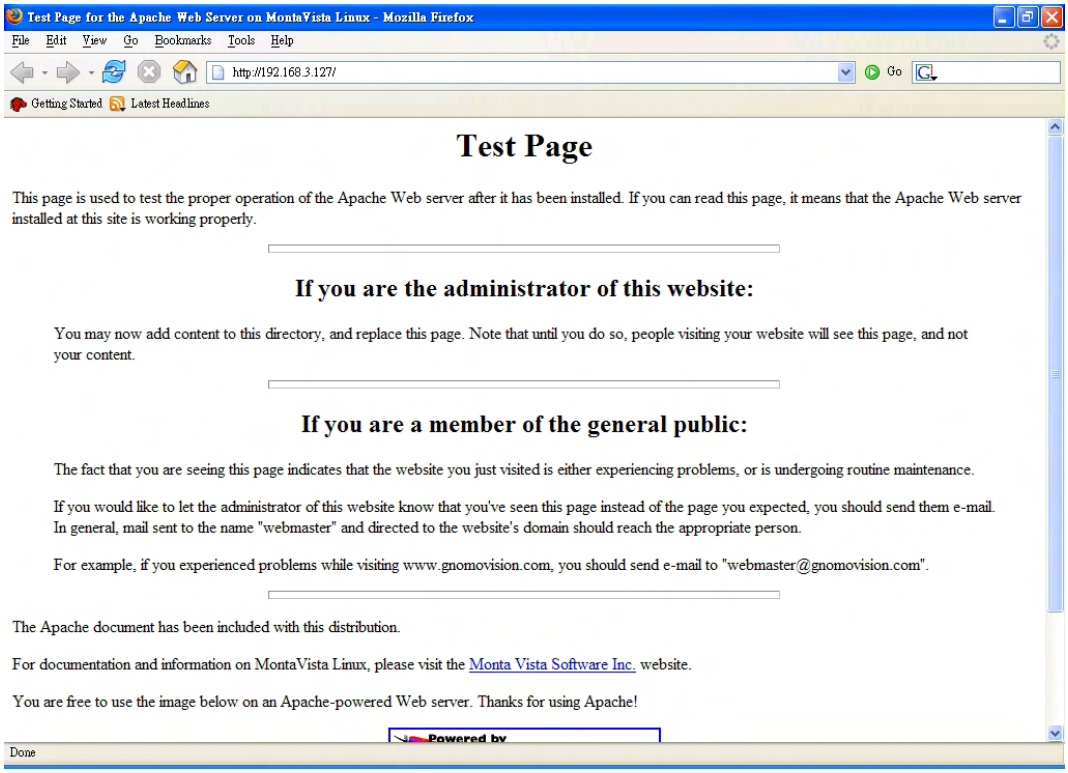
The Apache web server's main configuration file is **/etc/apache/httpd.conf**, with the default homepage located at **/usr/www/html/index.html**. Save your own homepage to the following directory:

/usr/www/html/

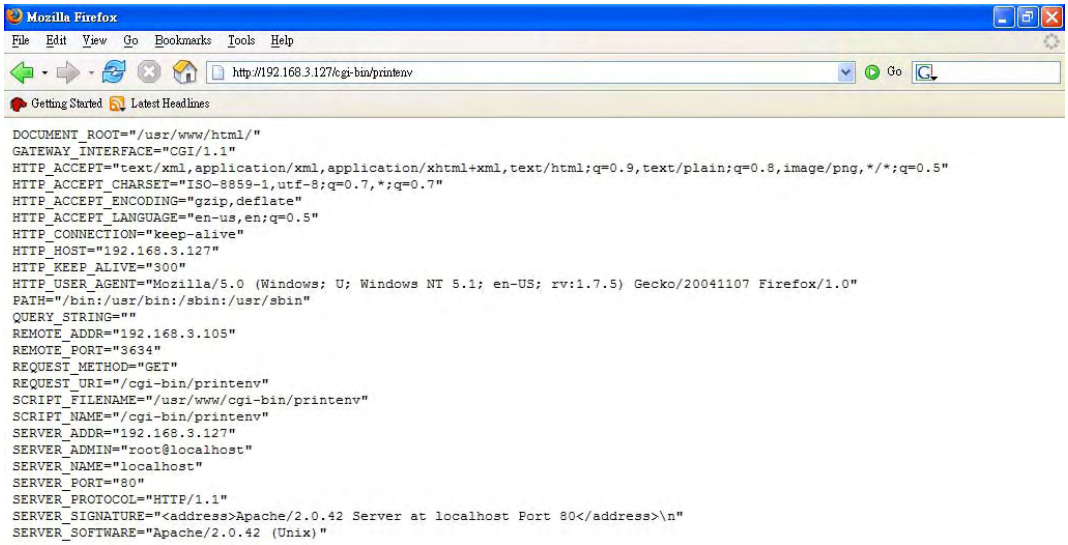
Save your CGI page to the following directory:

/usr/www/cgi-bin/

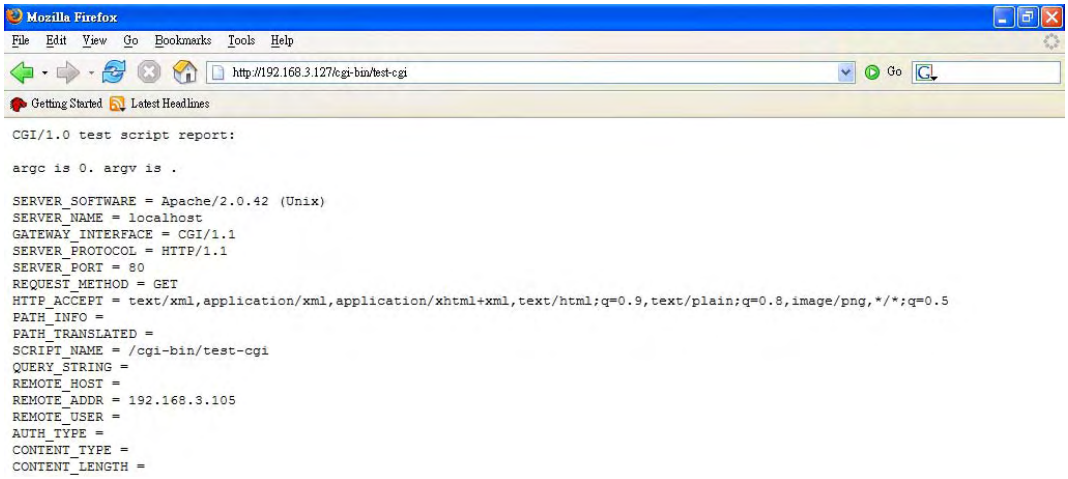
Before you modify the homepage, use a browser (such as Microsoft Internet Explore or Mozilla Firefox) from your PC to test if the Apache Web Server is working. Type the LAN1 IP address in the browser's address box to open the homepage. E.g., if the default IP address is still active, type **http://192.168.3.127** in address box.



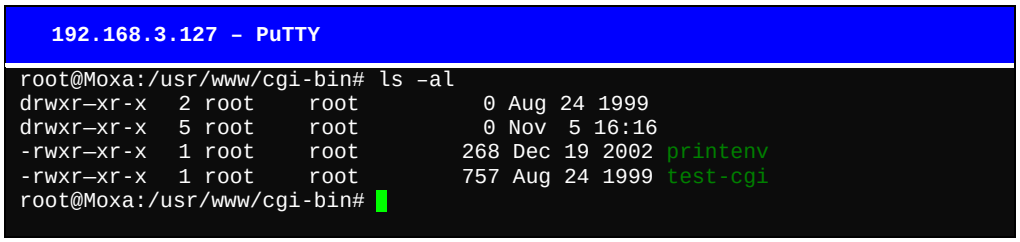
To open the default CGI page, type **http://192.168.3.127/cgi-bin/printenv** in your browser's address box.



To open the default CGI test script report page, type **http://192.168.3.127/cgi-bin/test-cgi** in your browser's address box.



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



Saving a Web Page to the CF Card

Since some applications will have web pages that take up a lot of memory space, you will need to be able to run the homepage and other pages from the CF card. In this section, we use a simple example to illustrate how to save web pages to the CF card, and then configure the Apache web server to open the pages. The files used in this example can be downloaded from Moxa's website.

Step 1:
Prepare web page and put pages to CF card. Click on the following link to download the web page test suite: http://www.w3.org/MarkUp/Test/HTML401.zip. Uncompress the zip file to your desktop PC, and then use FTP to transfer it to UC-7408's **/mnt/hda** directory.

```
192.168.3.127 – PuTTY
root@Moxa:/mnt/hda# ls –al
drwxr–xr-x 4 root    root      16384 Dec 11 14:18
drwxr–xr-x 6 root    root          0 Sep 29 17:43
-rwxr–xr-x 1 root    root       1768 Dec 11 14:16 W3C.gif
drwxr–xr-x 2 root    root       4096 Dec 11 14:19 assertions
-rwxr–xr-x 1 root    root      36071 Dec 11 14:18 htmltestdocumen
taml
-rwxr–xr-x 1 root    root       3145 Dec 11 14:16 index.html
-rwxr–xr-x 1 root    root         90 Dec 11 14:17 section.css
drwxr–xr-x 2 root    root      24576 Dec 11 14:20 tests
-rwxr–xr-x 1 root    root       2303 Dec 11 14:16 vh401.gif
root@Moxa:/mnt/hda#
```

Step 2:
Use the following commands to configure the Apache web server's DocumentRoot:

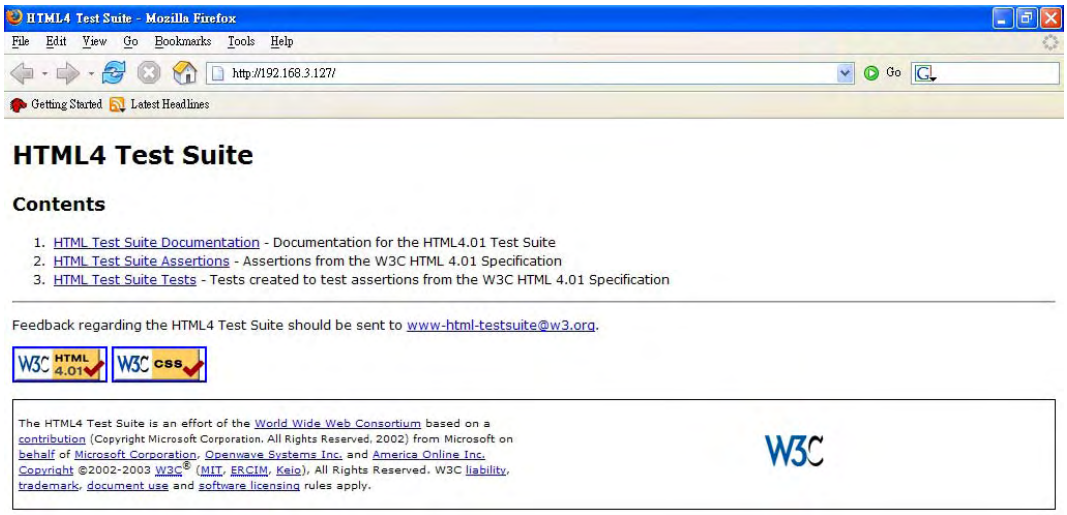
```
#cd /etc/apache
#vi httpd.conf
……
DocumentRoot "/mnt/hda"   //Change the document root directory
                  //to your CF card.
……
```

```
192.168.3.127 – PuTTY
ServerRoot "/etc/apache"
PidFile /var/run/httpd.pid
ScoreBoardFile /var/run/httpd.scoreboard
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15
MinSpareServers 5
MaxSpareServers 10
StartServers 5
MaxClients 150
MaxRequestsPerChild 0
Listen 80
User nobody
Group nobody
ServerAdmin root@localhost
ServerName localhost
DocumentRoot "/mnt/had"
```

Step 3:
Use the following commands to restart the Apache web server:

```
#cd /etc/init.d
#./apache restart
```

Step4:
Open your browser and connect to the UC-7408 by typing the current LAN1 IP address in the browser's address box.



NOTE Visit the Apache website at http://httpd.apache.org/docs/ for more information about setting up an Apache server.

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."

UC-7408 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.

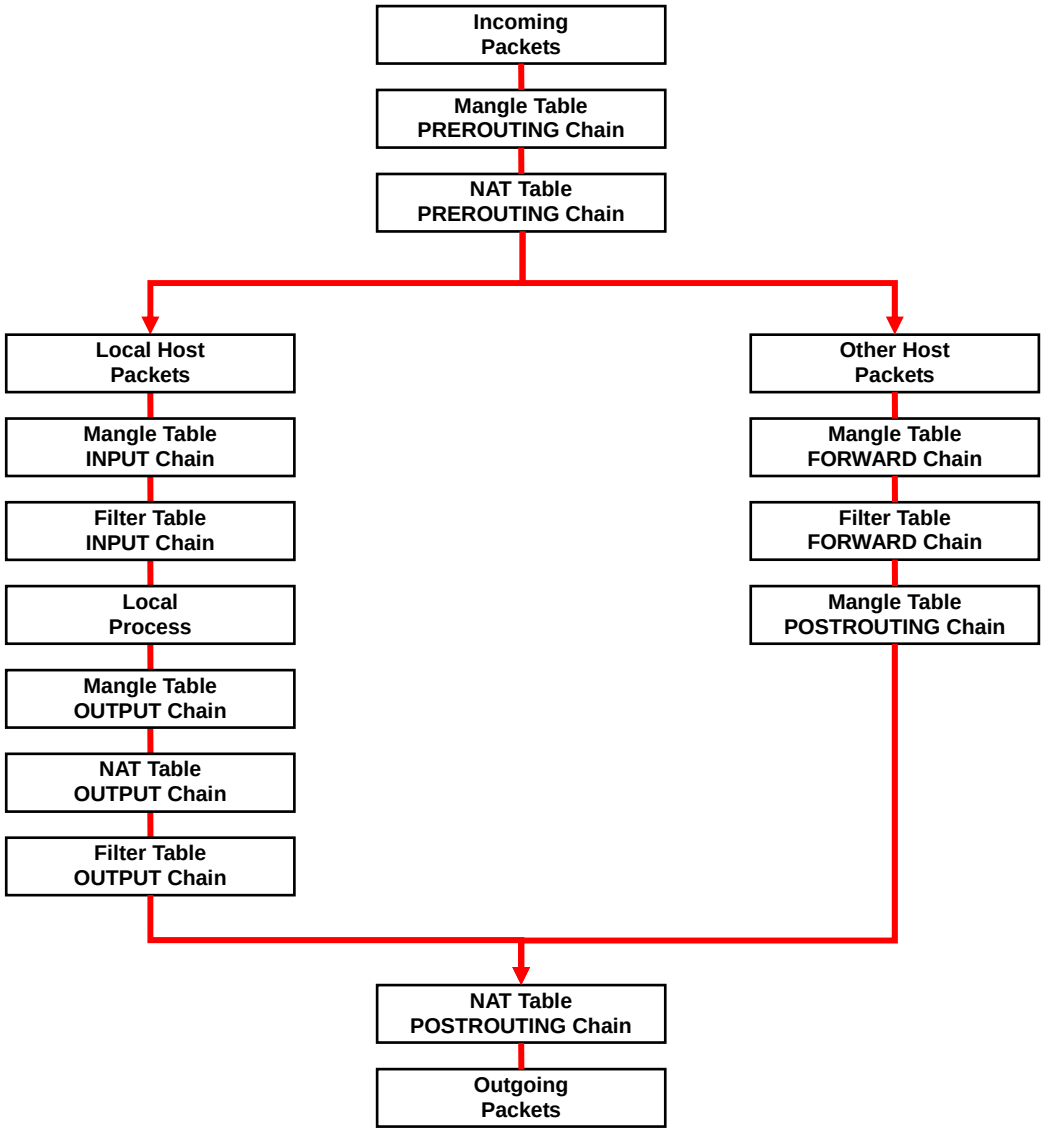
C. **Mangle Table**—includes two chains

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.



UC-7408 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_MIRROT	ipt_length	ipt_tos
ip_nat_ftp	ipt_REDIRECT	ipt_limit	ipt_ttl
ip_nat_irc	ipt_REJECT	ipt_mac	ipt_unclean
ip_nat_snmp_basic	ipt_TCPMSS	ipt_mark	
ip_queue	ipt_TOS	ipt_multiport	
ipt_LOG	ipt_ULOG	ipt_owner	

NOTE UC-7408 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-7408. 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 via a LAN or PPP may be denied access. We recommend using the Serial Console to set up the IPTABLES.

Click on the following links for more information about iptables.

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

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

Observe and erase chain rules

Usage:

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

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

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

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

Examples:

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

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

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

Define policy for chain rules

Usage:

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

-P: Set the policy for the chain to the given target.
INPUT: For packets coming into the UC-7408.
OUTPUT: For locally-generated packets.
FORWARD: For packets routed out through the UC-7408.
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 ixp0 –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 ixp0 –p tcp –s 192.168.1.0/24 –j ACCEPT
```

Example 4: Drop TCP packets from 192.168.1.25.
```
# iptables –A INPUT –i ixp0 –p tcp –s 192.168.1.25 –j DROP
```

Example 5: Drop TCP packets addressed for port 21.
```
# iptables –A INPUT –i ixp0 –p tcp --dport 21 –j DROP
```

Example 6: Accept TCP packets from 192.168.0.24 to UC-7408's port 137, 138, 139
```
# iptables –A INPUT –i ixp0 –p tcp –s 192.168.0.24 --dport 137:139 –j ACCEPT
```

Example 7: Log TCP packets that visit UC-7408's port 25.
```
# iptables –A INPUT –i ixp0 –p tcp --dport 25 –j LOG
```

Example 8: Drop all packets from MAC address 01:02:03:04:05:06.
```
# iptables –A INPUT –i ixp0 –p all –m mac –mac-source 01:02:03:04:05:06 –j DROP
```

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

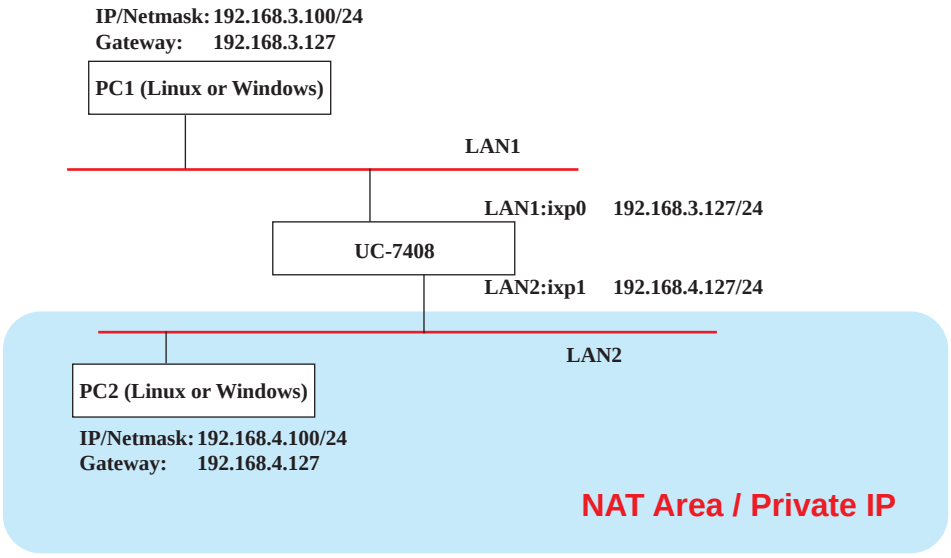
NAT

NAT (Network Address Translation) protocol translates IP addresses used on one network different IP addresses used on another network. One network is designated the inside network and the other is the outside network. Typically, UC-7408 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 all packets leaving LAN1 are changed to 192.168.3.127 (you will need to load the module ipt_MASQUERADE):



1. `#ehco 1 > /proc/sys/net/ipv4/ip_forward`
2. `#modprobe iptable_nat`
3. `#modprobe ip_conntract`
4. `#modprobe ipt_MASQUERADE`
5. `#iptables -t nat –A POSTROUTING –o ixp0 –j SNAT --to-source 192.168.3.127`

   or
6. `#iptables –t nat –A POSTROUTING –o ixp0 –j MASQUERADE`

Enabling NAT at Bootup

In the most of real world situations, you will want to use a simple shell script to enable NAT when UC-7408 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='ixp0'  #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_nat_ftp  2> /dev/null
modprobe ip_nat_irc  2> /dev/null
modprobe ip_conntrack  2> /dev/null
modprobe ip_conntrack_ftp  2> /dev/null
modprobe ip_conntrack_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 UC-7408's Ethernet port. Since PPP is a peer-to-peer system, UC-7408 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 baud rate.

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, something like this:

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

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

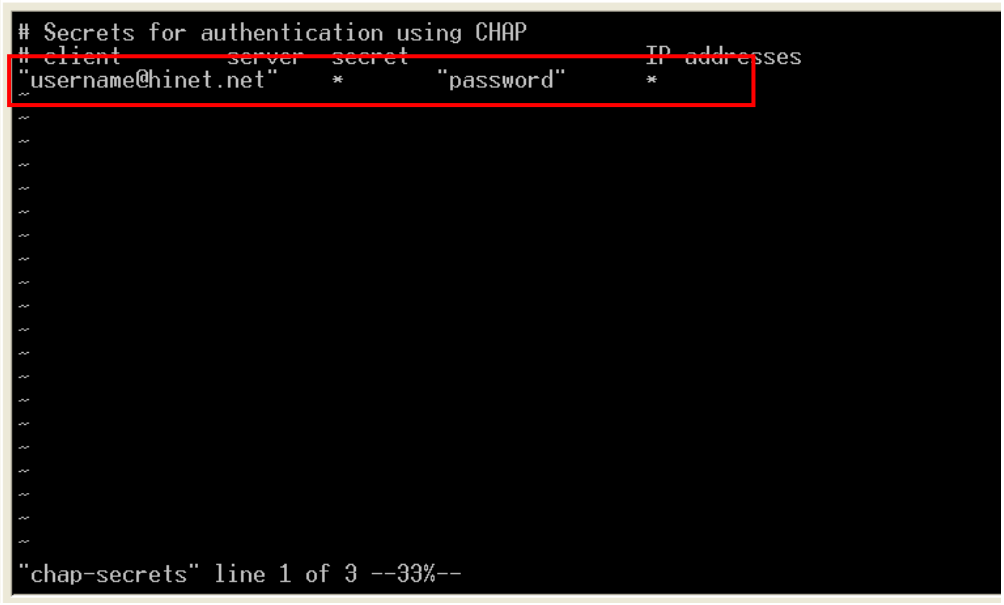
The following example does not check the username and password:

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

PPPoE

How to use PPPoE on UC-7408:

1. Update two files: **/usr/sbin/pppd** and **/usr/lib/pppd/2.4.1/pppoe.so** on the target UC-7408 for version V1.5 or earlier versions. Copy the files from the web or CD-ROM, and directly update it by the copy command or FTP.
2. Connect UC-7408's LAN port to an ADSL modem with a cross-over cable, HUB, or switch.
3. Login to the UC-7408 as the root user.
4. Edit the file **/etc/ppp/chap-secrets** and add the following:
   "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.

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

```
# password if you don't use the login option of pppd! The mgetty Debian
# 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"      *

# 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        "*"     -

# OUTBOUND connections

# Here you should add your userid password to connect to your providers via
# PAP. The * means that the password is to be used for ANY host you connect
# to. Thus you do not have to worry about the foreign machine name. Just
# replace password with your password.
"pap-secrets" line 1 of 42 --2%--
```

"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.

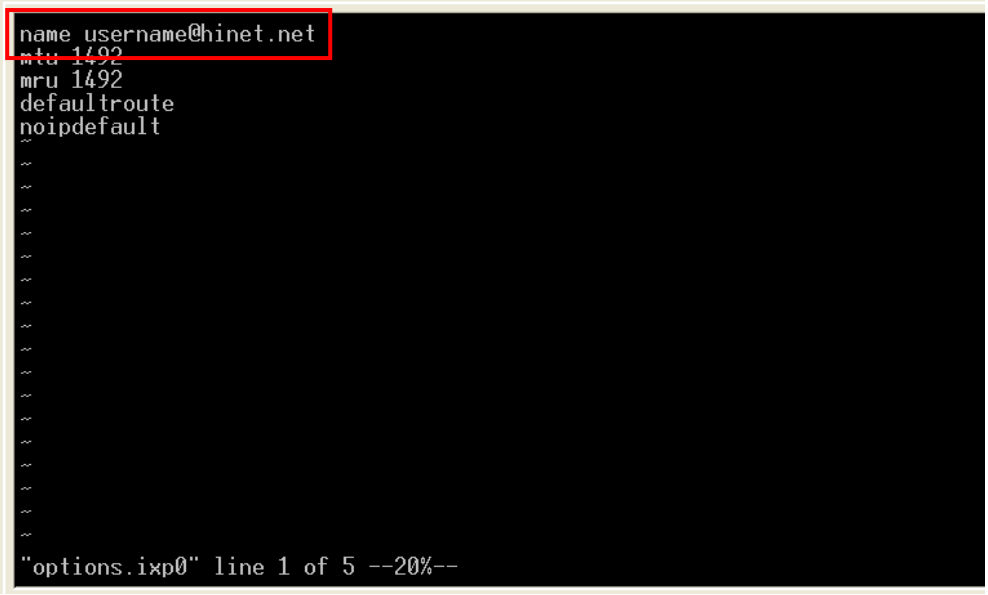
6. Edit the file **/etc/ppp/options** and add the following line:
   plugin pppoe

```
# terminated because it was idle.
#holdoff <n>

# 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>
plugin pppoe.so

# ---<End of File>---
~
~
~
~
~
~
~
~
~
~
"options" line 1 of 342 --0%--
```

7. Add one of two files: **/etc/ppp/options.ixp0** or **/etc/ppp/options.ixp1**. 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.ixp0**. If you use LAN2 to connect to the ADSL modem, then add **/etc/ppp/options.ixp1**. The file context is shown below:



   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.

8. 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

9. Use the following command to create a pppoe connection:

   pppd ixp0

   The ixp0 is what is connected to the ADSL modem LAN port. The example above uses LAN1. To use LAN2, type:

   pppd ixp1

10. 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.
11. 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 UC-7408, without worrying about the amount of disk space that will be available. UC-7408 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 UC-7408 as an NFS Server

By default, UC-7408 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, UC-7408 shares the **/tmp** directory to everyone, gives 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, UC-7408 shares the directory **/home/public** to a local network 192.168.0.0/24, with read and write authority. NFS clients can just read **/home/public**; they do not have write authority.

Example 3

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

In this example, UC-7408 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 UC-7408 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

UC-7408 has built-in SNMP V1 (Simple Network Management Protocol) agent software. It supports RFC1317 RS-232 like group and RFC 1213 MIB-II.

The following simple example allows you to use an SNMP browser on the host site to query the UC-7408, which is the SNMP agent. UC-7408 will respond.

```
***** SNMP QUERY STARTED *****
1: sysDescr.0 (octet string) Linux Moxa 2.4.18_mvl30-ixdp425 #1049 Tue Oct 26 09:34:15 CST 2004 armv5teb
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 group:
http://www.faqs.org/rfcs/rfc1213.html
http://www.faqs.org/rfcs/rfc1317.html

→ UC-7408 does NOT support SNMP trap.

The following tables list the variables supported by UC-7408.

Open VPN

This function is only available for firmware version V1.5 (and later versions).

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

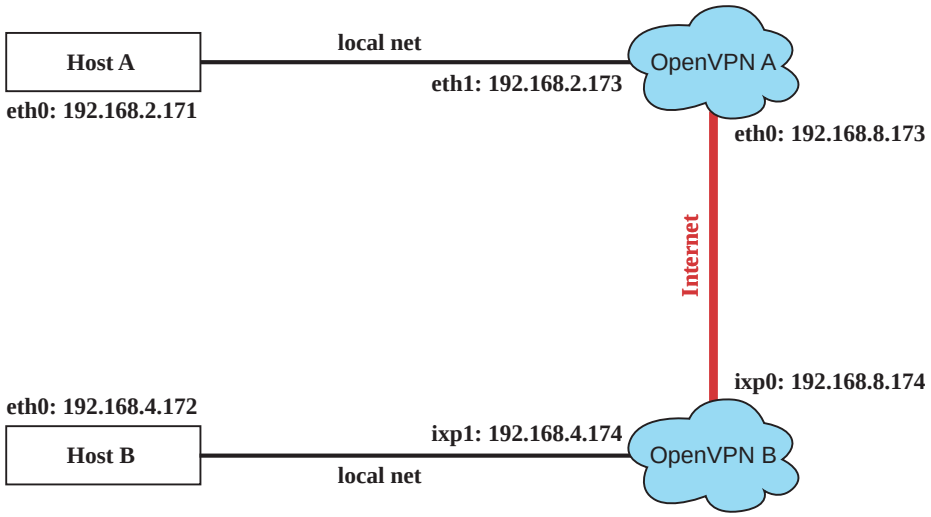
```
# mknod /dev/net/tun c 10 200
```

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.

2. Generate a preset shared key by typing the command:

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

Copy the file that is generated to the OpenVPN machine.

3. Generate a script file named **openvpn-bridge** on each OpenVPN machine. This script reconfigures interface "ixp1" as IP-less, creates logical bridge(s) and TAP interfaces, loads modules, enables IP forwarding, etc.

```
#--------------------------------Start----------------------------

#!/bin/sh

iface=ixp1   # 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-ixp1
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
```

4. Create a configuration file named **A-tap0-br.conf** and an executable script file named **A-tap0-br.sh** on OpenVPN A.

```
# 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/A-tap0-br.sh
```

```
#---------------------------------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 dev br0
#--------------------------------- end ---------------------------
```

Create a configuration file named **B-tap0-br.conf** and an executable script file named **B-tap0-br.sh** on OpenVPN B.

```
# 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/B-tap0-br.sh
```

```
#--------------------------------- 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 dev br0
#--------------------------------- end ---------------------------
```

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

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

5. Start both of OpenVPN peers,

```
# openvpn --config A-tap0-br.conf&
# openvpn --config B-tap0-br.conf&
```

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

6. 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	ixp0

Interface **ixp1** 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 **ixp1** 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.

7. 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
```

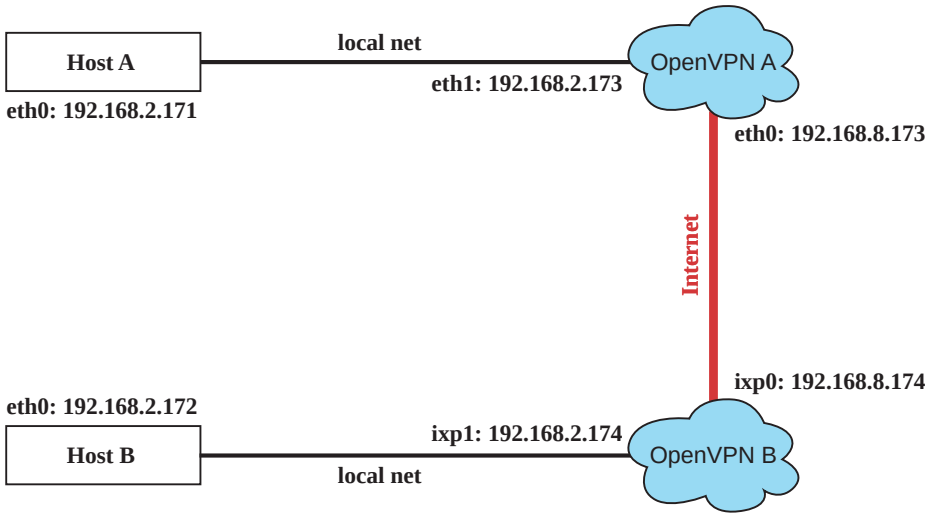
A successful ping indicates that you have created a VPN system that only allows authorized users from one internal network to access users at the remote site. For this system, all data is transmitted by UDP packets on port 5000 between OpenVPN peers.

8. To shut down OpenVPN programs, type the command:

```
# killall -TERM openvpn
```

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

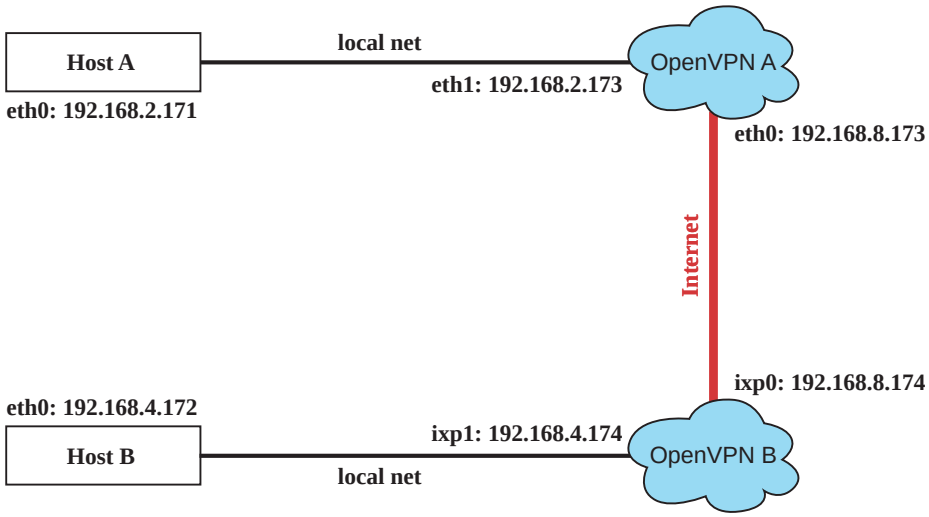
1. Set up four machines as shown in the following diagram:



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/A-tap0-br.conf" and "/etc/openvpn/B-tap0-br.conf".

Setup 3: Routed IP

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



2. Create a configuration file named "A-tun.conf" and an executable script file named "A-tun.sh".

```
# 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/A-tun.sh
```

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

Create a configuration file named **B-tun.conf** and an executable script file named **B-tun.sh** on OpenVPN B:

```
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/B-tun.sh
```

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

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

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

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

```
# route
```

Destination	Gateway	Genmsk	Flags	Metric	Ref	Use	Iface
192.168.4.174	*	255.255.255.255	UH	0	0	0	tun0
192.168.4.0	192.168.4.174	255.255.255.0	UG	0	0	0	tun0
192.168.2.0	*	255.255.255.0	U	0	0	0	ixp1
192.168.8.0	*	255.255.255.0	U	0	0	0	ixp0

5

Programmer's Guide

This chapter includes important information for programmers.

The following functions are covered in this chapter:

- **Flash Memory Map**
- **Linux Tool Chain Introduction**
- **Debugging with GDB**
- **Device API**
- **RTC (Real Time Clock)**
- **Buzzer**
- **WDT (Watch Dog Timer)**
- **UART**
- **Digital I/O**
- **Make File Example**

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 – 0x0015FFFF	1 MB	Kernel object code—Read ONLY
0x00160000 – 0x0055FFFF	4 MB	Mini root file system (EXT2) —Read ONLY
0x00560000 – 0x01F5FFFF	26 MB	User root file system (JFFS2) —Read/Write
0x01F60000 – 0x01FBFFFF	384 KB	Not used
0x01FC0000 – 0x01FDFFFF	128 KB	Boot Loader configuration—Read ONLY
0x01FE0000 – 0x01FFFFFF	128 KB	Boot Loader directory—Read ONLY

Mount the user file system to **/mnt/usrdisk** with the root file system. Check to see if the user file system was mounted correctly. If user file system is okay, the kernel will change the root file system to **/mnt/usrdisk**. If the user file system is not okay, the kernel will use the default Moxa file system. To finish boot process, run the init program.

NOTE

1. The default Moxa file system only enables the network and CF. It lets users recover the user file system when it fails.
2. The user file system is a complete file system. Users can create and delete directories and files (including source code and executable files) as needed.
3. Users can create the user file system on the PC host or target platform, and then copy it to the UC-7408.

Linux Tool Chain Introduction

To ensure that an application will be able to run correctly when installed on UC-7408, you must ensure that it is compiled and linked to the same libraries that will be present on the UC-7408. This is particularly true when the RISC Xscale processor architecture of the UC-7408 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 UC-7408 contains a suite of cross compilers and other tools, as well as the libraries and headers that are necessary to compile applications for UC-7408. The host environment must be running Linux to install the UC-7408 GNU Tool Chain. We have confirmed that the following Linux distributions can be used to install the tool chain:

Redhat 7.3/8.0/9.0, Fefora core 1 & 2.

The Tool Chain will need about 100 MB of hard disk space on your PC. The UC-7408 Tool Chain is located on the UC-7408 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount /dev/cdrom /mnt/cdrom
#rpm –ivh /mnt/cdrom/mxscaleb-3.3.2-6.i386.rpm
```

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/mxscaleb/bin** to your path and the directory **/usr/local/mxscaleb/man** to your manual path. You can do this temporarily for the current login session by issuing the following commands:

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

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 **armv5b-linux-gcc** compiler, issue the command:

```
#man armv5b-linux-gcc
```

Cross Compiling Applications and Libraries

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

```
#mxscaleb-gcc  –o example –Wall –g –O2 example.c
#mxscaleb-strip  –s example
#mxscaleb-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-7408 Xscale boards, it is **`mxscaleb-`**.

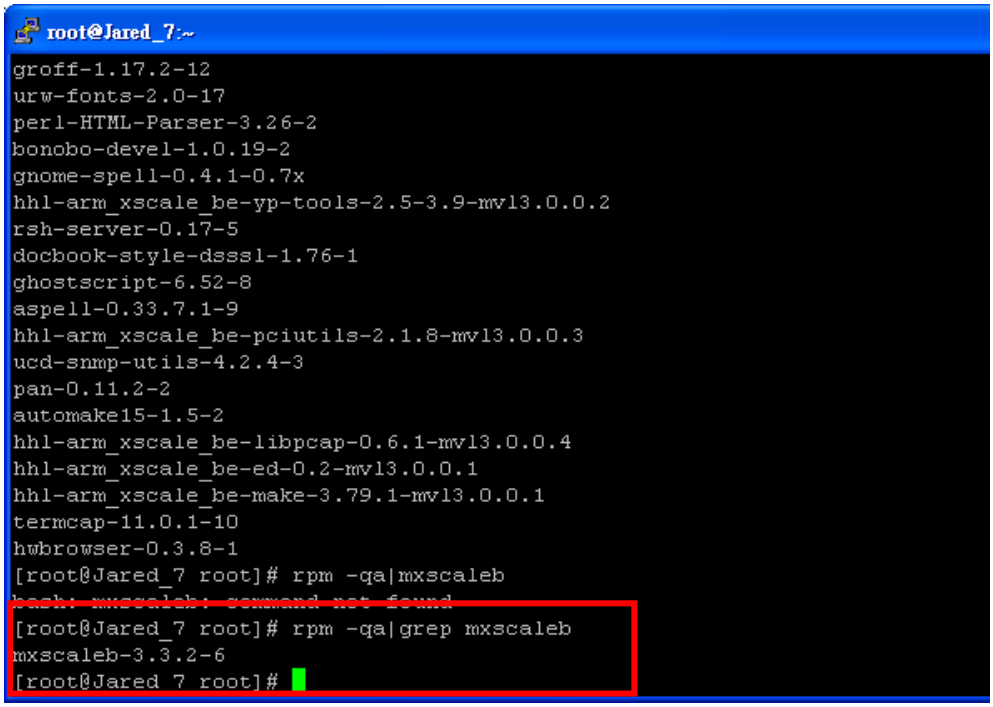
For example the native C compiler is **`gcc`** and the cross C compiler for Xscale in UC-7408 is **`mxscaleb-gcc.`**

The following cross compiler tools are provided:

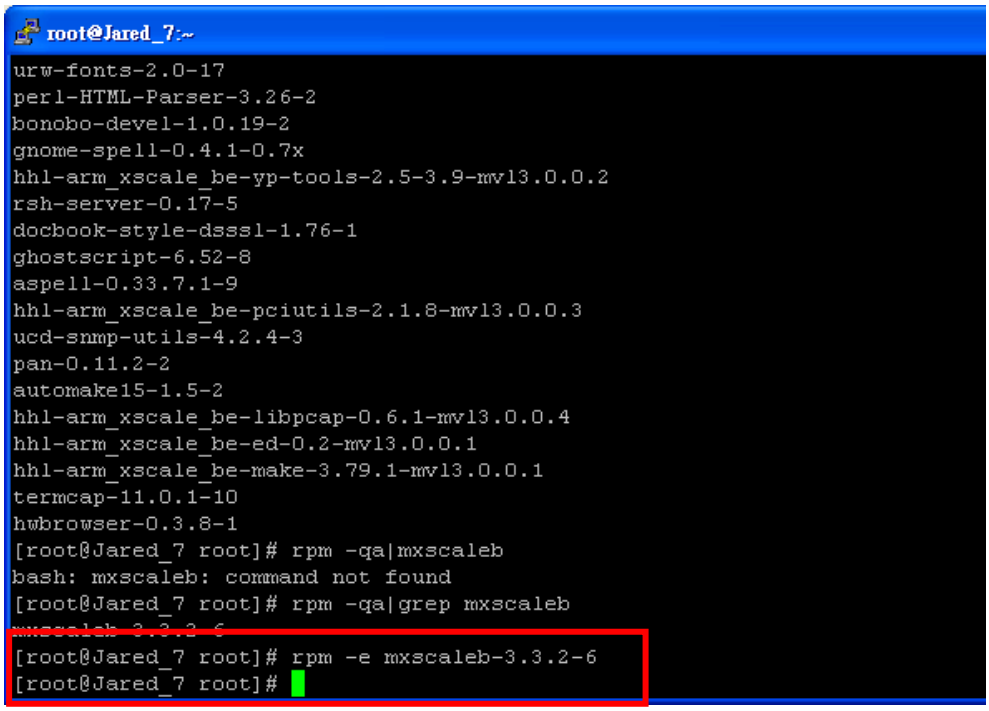
ar	Manage archives (static libraries)
as	Assembler
c++, g++	C++ compiler
cpp	C preprocessor
gcc	C compiler
gdb	Debugger
ld	Linker
nm	Lists symbols from object files
objcopy	Copies and translates object files
objdump	Displays information about object files
ranlib	Generates indexes to archives (static 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)

Uninstalling the Linux Tool Chain

Use the command **rpm –qa|grep mxscaleb** to query if the Moxa tool chain is installed on the system.



Use the command **rpm -e mxscale-x.x.x-x** to uninstall the Moxa Xscale tool chain.



Debugging with GDB

First compile the program must with option -ggdb. Use the following steps:

1. To debug a program called **hello-debug** on the target, use the command:

   ```
   #gdbserver 192.168.4.142:2000 hello-debug
   ```

   This is where 2000 is the network port number on which the server waits for a connection from the client. This can be any available port number on the target. Following this are the name of the program to be debugged (hello-debug), plus that program's arguments. Output similar to the following will be sent to the console:

   ```
   Process hello-debug created; pid=38
   ```

2. Use the following command on the host to change to the directory that contains **hello-debug**:

   ```
   cd /my_work_directory/myfilesystem/testprograms
   ```

3. Enter the following command:

   ```
   #ddd --debugger mxscaleb-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

UC-7408 supports control devices with the **ioctl** system API. You will need to `include <moxadevice.h>`, and use the following **ioctl** function.

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

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

```
#man ioctl
```

RTC (Real Time Clock)

The device node is located at **/dev/rtc**. UC-7408 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**. UC-7408 supports Linux standard buzzer control, with UC-7408's buzzer running at a fixed frequency of 100 Hz. You must **`include <sys/kd.h>`**.

1. Function: KDMKTONE

   ```
   ioctl(fd, KDMKTONE, unsigned int arg);
   ```

   Description: The buzzer's behavior is determined by the argument **`arg`**. The "high word" part of arg gives the length of time the buzzer will sound, and the "low word" part gives the frequency.

   The buzzer's on / off behavior is controlled by software. If you call the "ioctl" function, you MUST set the frequency at 100 Hz. If you use a different frequency, the system could crash.

WDT (Watch Dog Timer)

1. Introduction

The WDT works like a watch dog function. You can enable it or disable it. When the user enables WDT but the application does not acknowledge it, the system will reboot. You can set the ack time from a minimum of 50 msec to a maximum of 60 seconds.

2. How the WDT works

The sWatchDog is enabled when the system boots up. The kernel will auto ack it. The user application can also enable ack. When the user does not ack, it will let the system reboot.

Kernel boot
…..
….
User application running and enable user ack
….
….

3. The user API

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

```
all:
    mxscaleb-gcc –o xxxx  xxxx.c -lmoxalib
```

```
int swtd_open(void)
```

Description
Open the file handle to control the sWatchDog. If you want to do something you must first to this. And keep the file handle to do other.

Input
None

Output
The return value is file handle. If has some error, it will return < 0 value.

You can get error from errno().

```
int swtd_enable(int fd, unsigned long time)
```

Description
Enable application sWatchDog. And you must do ack after this process.

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

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

Output
OK will be zero. The other has some error, to get the error code from errno().

```
int swtd_disable(int fd)
```

Description:

Disable the application to ack sWatchDog. And the kernel will be auto ack it. User does not to do it at periodic.

Input :

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

Output:

OK will be zero. The other has some error, to get error code from errno.

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

Description:

Get current setting values.

mode –

1 for user application enable sWatchDog: need to do ack.

0 for user application disable sWatchdog: does not need to do ack.

time – The time period to ack sWatchDog.

Input :

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

int *mode - the function will be return the status enable or disable user application need to do ack.

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

Output:

OK will be zero.

The other has some error, to get error code from errno().

`int swtd_ack(int fd)`

Description:

Acknowledge sWatchDog. When the user application enable sWatchDog. It need to call this function periodically with user predefined time in the application program.

Input :

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

Output:

OK will be zero.

The other has some error, to get error code from errno().

`int swtd_close(int fd)`

Description:

Close the file handle.

Input :

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

Output:

OK will be zero.

The other has some error, to get error code from errno().

4. Special Note

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

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

5. User application example

Example 1:

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

int main(int argc, char *argv[])
{
   int  fd;

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

```
    swtd_close(fd);
    exit(0);
}
```

The makefile is shown below:

```
all:
    mxscaleb-gcc –o xxxx xxxx.c –lmoxalib
```

Example 2:

```
#include<stdio.h>
#include<stdlib.h>
#include<signal.h>
#include<string.h>
#include<sys/stat.h>
#include<sys/ioctl.h>
#include<sys/select.h>
#include<sys/time.h>
#include<moxadevice.h>

static void mydelay(unsigned long msec)
{
    struct timevaltime;

    time.tv_sec = msec / 1000;
    time.tv_usec = (msec % 1000) * 1000;
    select(1, NULL, NULL, NULL, &time);
}

static int swtdfd;
static int stopflag=0;

static void stop_swatchdog()
{
    stopflag = 1;
}

static void do_swatchdog(void)
{
    swtd_enable(swtdfd, 500);
    while ( stopflag == 0 ) {
      mydelay(250);
      swtd_ack(swtdfd);
    }
    swtd_disable(swtdfd);
 }

int main(int argc, char *argv[])
{
    pid_t    sonpid;

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

```
    exit(1);
}
```

The makefile is shown below:

```
all:
    mxscaleb-gcc –o xxxx xxxx.c –lmoxalib
```

UART

The normal tty device node is located at **/dev/ttyM0 … ttyM7**, and the modem tty device node is located at **/dev/cum0 … cum7**.

UC-7408 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. UC-7408 supports RS-232, RS-422, 2-wire RS-485, and 4-wire RS485.

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

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

1. Function: MOXA_SET_OP_MODE

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

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

2. Function: MOXA_GET_OP_MODE

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

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

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

Function: MOXA_SET_SPECIAL_BAUD_RATE
Function: MOXA_GET_SPECIAL_BAUD_RATE

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

Example to set the baud rate

```
#include <moxadevice.h>
#include <termios.h>
struct termiosterm;
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 to get the baud rate

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

Baud rate inaccuracy

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

*The Inaccuracy should less than 2% for work reliably.

Special Note

1. If the target baud rate 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.

Digital I/O

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

Application Programming Interface

Return error code definitions:

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

The definition for DIN and DOUT:

```
#define DIO_HIGH                1
#define DIO_LOW                 0
```

int set_dout_state(int doport, int state)

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

Input: **int doport** - which DOUT port you want to set. Port starts from 0 to 7.

int state - to set high or low state; DIO_HIGH (1) for high, DIO_LOW (0) for low.

Output: none.

Return: reference the error code.

int get_din_state(int diport, int *state)

Description: To get the DIN port state.

Input: **int diport** - get the current state of which DIN port. Port numbering is from 0 to 7.

int *state - save the current state.

Output: **state** - DIO_HIGH (1) for high, DIO_LOW (0) for low.

Return: reference the error code.

int get_dout_state(int doport, int *state)

Description: To get the DOUT port state.

Input: **int doport** - get the current state of which DOUT port.

int *state - save the current state.

Output: **state** - DIO_HIGH (1) for high, DIO_LOW (0) for low.

Return: reference the error code.

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

Description: Set the event for DIN when the state is changed from high to low or from low to high.

Input: **int diport** - the port that will be used to detect the DIN event. Port numbering is from 0 to 7.

void (*func) (int diport) - Not NULL
> Returns the call back function. When the event occurs, the call back function will be invoked.
NULL
> Clears this event

int mode
DIN_EVENT_HIGH_TO_LOW (1): from high to low
DIN_EVENT_LOW_TO_HIGH (0): from low to high
DIN_EVENT_CLEAR (-1): clear this event

unsigned long duration - 0: detect the din event
> DIN_EVENT_HIGH_TO_LOW or DIN_EVENT_LOW_TO_HIGH> without

duration
- Not 0
> detect the din event DIN_EVENT_HIGH_TO_LOW or DIN_EVENT_LOW_TO_HIGH with duration. The value of "duration" must be a multiple of 20 milliseconds. The range of "duration" is 0, or 40 <= duration <= 3600000 milliseconds. The error of the measurement is 24 ms. For example, if the DIN duration is 200 ms, this event will be generated when the DIN pin stays in the same state for a time between 176 ms and 200 ms.

Output: none.

Return: reference the error code.

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

Description: To retrieve the DIN event configuration, including mode (DIN_EVENT_HIGH_TO_LOW or DIN_EVENT_LOW_TO_HIGH), and the value of "duration."

Input:	**int diport**	- which DIN port you want to retrieve. - The port whose din event setting we wish to retrieve
	int *mode	- save which event is set.
	unsigned long *duration	- the duration of the DIN port is kept in high or low state. - return to the current duration value of diport
Output:	**mode**	DIN_EVENT_HIGH_TO_LOW (1): from high to low DIN_EVENT_LOW_TO_HIGH(0): from low to high DIN_EVENT_CLEAR(-1): clear this event
	duration	The value of duration should be 0 or 40 <= duration <= 3600000 milliseconds.

Return: reference the error code.

Special Note

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

Examples

Example 1

File Name: tdio.c

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

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

```
#ifdef DEBUG
#define dbg_printf(x...)  printf(x)
#else
#define dbg_printf(x...)
#endif

#define MIN_DURATION 40

static char *DataString[2]={"Low  ", "High "};
static void hightolowevent(int diport)
{
printf("\nDIN port %d high to low.\n", diport);
}

static void lowtohighevent(int diport)
{
printf("\nDIN port %d low to high.\n", diport);
}

int main(int argc, char * argv[])
{
int  i, j, state, retval;
unsigned long duration;

while( 1 ) {
  printf("\nSelect a number of menu, other key to exit.  \n\
1. set high to low event    \n\
2. get now data.            \n\
3. set low to high event    \n\
4. clear event              \n\
5. set high data.           \n\
6. set low data.            \n\
7. quit                     \n\
8. show event and duration  \n\
Choose : ");
retval =0;
  scanf("%d", &i);
  if ( i == 1 ) {  // set high to low event
    printf("Please keyin the DIN number : ");
    scanf("%d", &i);
    printf("Please input the DIN duration, this minimun value must be over %d :
",MIN_DURATION);
    scanf("%lu", &duration);
    retval=set_din_event(i, hightolowevent, DIN_EVENT_HIGH_TO_LOW, duration);
  } else if ( i == 2 ) {  // get now data
    printf("DIN data  : ");
    for ( j=0; j<MAX_DIN_PORT; j++ ) {
      get_din_state(j, &state);
      printf("%s", DataString[state]);
    }
    printf("\n");
    printf("DOUT data : ");
    for ( j=0; j<MAX_DOUT_PORT; j++ ) {
      get_dout_state(j, &state);
      printf("%s", DataString[state]);
    }
    printf("\n");
  } else if ( i == 3 ) {  // set low to high event
    printf("Please keyin the DIN number : ");
    scanf("%d", &i);
    printf("Please input the DIN duration, this minimun value must be over %d :
",MIN_DURATION);
    scanf("%lu", &duration);
    retval = set_din_event(i, lowtohighevent, DIN_EVENT_LOW_TO_HIGH, duration);
  } else if ( i == 4 ) {  // clear event
    printf("Please keyin the DIN number : ");
    scanf("%d", &i);
    retval=set_din_event(i, NULL, DIN_EVENT_CLEAR, 0);
```

```
  } else if ( i == 5 ) {  // set high data
     printf("Please keyin the DOUT number : ");
     scanf("%d", &i);
     retval=set_dout_state(i, 1);
  } else if ( i == 6 ) {  // set low data
     printf("Please keyin the DOUT number : ");
     scanf("%d", &i);
     retval=set_dout_state(i, 0);
  } else if ( i == 7 ) {  // quit
     break;
  } else if ( i == 8 ) {  // show event and duration
     printf("Event:\n");
     for ( j=0; j<MAX_DOUT_PORT; j++ ) {
        retval=get_din_event(j, &i, &duration);
        switch ( i ) {
        case DIN_EVENT_HIGH_TO_LOW :
           printf("(htl,%lu)", duration);
           break;
        case DIN_EVENT_LOW_TO_HIGH :
           printf("(lth,%lu)", duration);
           break;
        case DIN_EVENT_CLEAR :
           printf("(clr,%lu)", duration);
           break;
        default :
           printf("err " );
           break;
        }
     }
     printf("\n");
  } else {
     printf("Select error, please select again !\n");
  }

  switch(retval){
       case DIO_ERROR_PORT:
          printf("DIO error port\n");
          break;
       case DIO_ERROR_MODE:
          printf("DIO error mode\n");
          break;
       case DIO_ERROR_CONTROL:
          printf("DIO error control\n");
          break;
       case DIO_ERROR_DURATION:
          printf("DIO error duratoin\n");
       case DIO_ERROR_DURATION_20MS:
          printf("DIO error! The duratoin is not a multiple of 20 ms\n");
          break;
  }
}

return 0;
}
```

Example 2

File Name: tduration.c

Description: The program indicates to connect DO1 to DI1, and the program will change the digital output state automatically at the fixed frequency, and then detect if the event changes of the digital input state is high or low during a different duration.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/time.h>
#include <fcntl.h>
```

```
#include <unistd.h>
#include <pthread.h>
#include <moxadevice.h>

#ifdef DEBUG
#define dbg_printf(x...)  printf(x)
#else
#define dbg_printf(x...)
#endif

#define DURATION_NUM 7
#define TEST_NUM 10

static int ndin_StateChangeDetected, ndout_StateChangeDetected;
static int nDuration;
static unsigned long duration[2][DURATION_NUM]={  { 50, 40, 35, 30, 25, 20, 15 }, { 160,
140, 120, 100, 80, 60, 40, } };


/*********************************************************************
  When the din state changed form high to low, this function will be invoked
*********************************************************************/
static void low2highevent(int diport)
{
ndin_StateChangeDetected++;
dbg_printf("din state changed:%d\n",ndin_StateChangeDetected);
}

/*********************************************************************
  This function is used to exchange the dout state periodically
*********************************************************************/
void dout_control(int signo)
{
int state;

get_dout_state(0, &state);
dbg_printf("dout state changed:%d\n",state);
if(state) // exchange the dout state periodically
{
   ndout_StateChangeDetected++;
   set_dout_state(0, 0);
}
else
{
   set_dout_state(0, 1);
}
}

void dio_test_function(void )
{
struct itimerval value;
int j, i, nChoice;
struct timeval tv;

do {

   printf("0.Test for Din duration==0.\n");
   printf("1.Test for Din duration!=0.\n");
   printf("9.Quit.\n" );
   printf("Please select a choice>");
   scanf("%d",&nChoice);

   if( nChoice == 9 ){  // Quit
      break;
   }
   else if( nChoice == 0 ){  //test for din duration==0

      for ( nDuration=0; nDuration < DURATION_NUM; nDuration++ ) {
```

```
        // configure the dout frequency. When the timer timeouts, dout_control() will
be called to change the dout state
        value.it_value.tv_sec = duration[0][nDuration]/1000;
        value.it_value.tv_usec = (duration[0][nDuration]%1000) *1000 ;
        value.it_interval = value.it_value;
        setitimer(ITIMER_REAL,&value,NULL);

        ndin_StateChangeDetected = 0;  // reset these counters
        ndout_StateChangeDetected = 0;

        printf("DI duration,:0, DO duration:%d\n",duration[0][nDuration]);

        set_din_event(0, low2highevent, DIN_EVENT_LOW_TO_HIGH, 0);

        while( ndin_StateChangeDetected < TEST_NUM ) {
          pause();
        }
        printf("ndin_StateChangeDetected:%d, ndout_StateChangeDetected:%d,\n",
ndin_StateChangeDetected, ndout_StateChangeDetected);
        printf("loss detection
probability:%f\%,\n",(ndout_StateChangeDetected-ndin_StateChangeDetected)*100.0/nd
out_StateChangeDetected);
      }
    }//end of if( nChoice ==0 )

    else if( nChoice == 1 ) { //test for din duration!=0

      for ( nDuration=0; nDuration < DURATION_NUM; nDuration++ ) {
        // configure the dout frequence. when the timer timeout, dout_control() will
be call to change the dout state
        value.it_value.tv_sec = duration[1][nDuration]/1000;
        value.it_value.tv_usec = ( duration[1][nDuration]%1000 ) *1000 ;
        value.it_interval = value.it_value;
        setitimer(ITIMER_REAL,&value,NULL);

        // Test for: dout kept in the same frequency but din set for different duration
        for( i=0; i<DURATION_NUM; i++) {
          if( duration[1][i] <= duration[1][nDuration] ) {
            // reset these counters
            ndin_StateChangeDetected = 0;
            ndout_StateChangeDetected = 0;

            printf("DI duration,:%d, DO duration:%d\n", duration[1][i],
duration[1][nDuration] );

            set_din_event(0, low2highevent, DIN_EVENT_LOW_TO_HIGH, duration[1][i]);

            while( ndout_StateChangeDetected < TEST_NUM ) {
              pause();
            }
            printf("ndin_StateChangeDetected:%d, ndout_StateChangeDetected:%d,\n",
ndin_StateChangeDetected, ndout_StateChangeDetected);
            printf("loss detection
probability:%f\%,\n",(ndout_StateChangeDetected-ndin_StateChangeDetected)*100.0/nd
out_StateChangeDetected);
          }
        } //end of for( i=0; i<DURATION_NUM; i++)
      }
    }
} while(1);

pthread_exit(NULL);
}

void init_sigaction(void)
{
struct sigaction act;
act.sa_handler=dout_control;
```

```
act.sa_flags=0;
sigemptyset(&act.sa_mask);
sigaction(SIGALRM,&act,NULL);
}

int main(int argc, char * argv[])
{
pthread_t dio_test;

init_sigaction();

set_dout_state(0, 0);  // set the DOUT0 as high

set_din_event(0, low2highevent, DIN_EVENT_LOW_TO_HIGH, duration[1][0]);

dio_test_function();

while( nDuration < DURATION_NUM )
  usleep(100000);

}
```

DIO Program Make File Example

```
FNAME=tdio
FNAME1=tduration
CC=mxscaleb-gcc
STRIP=mxscaleb-strip

release:
$(CC) -o $(FNAME) $(FNAME).c -lmoxalib -lpthread
$(CC) -o $(FNAME1) $(FNAME1).c -lmoxalib -lpthread
$(STRIP) -s $(FNAME)
$(STRIP) -s $(FNAME1)

debug:
$(CC) -DDEBUG -o $(FNAME)-dbg $(FNAME).cxx -lmoxalib -lpthread
$(CC) -DDEBUG -o $(FNAME1)-dbg $(FNAME1).cxx -lmoxalib -lpthread

clean:
/bin/rm -f $(FNAME) $(FNAME)-dbg $(FNAME1) $(FNAME1)-dbg *.o
```

Make File Example

The following Makefile file example codes are copied from the Hello example on UC-7408's CD-ROM.

```
CC = /usr/local/mxscaleb/mxscaleb-gcc
CPP = /usr/local/mxscaleb/mxscaleb-gcc
SOURCES = hello.c

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

all: hello

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

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

A

System Commands

Linux normal command utility collection

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

Editor

1. **vi** text editor
2. **cat** dump file context
3. **zcat** compress or expand files
4. **grep** 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

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**
6. **telnet**
7. **ftp**

Process

1. **kill** kill process
2. **ps** display now running process

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

Moxa special utilities

1. **kversion** show kernel version
2. **cat /etc/version** show user directory version
3. **upramdisk** mount ramdisk
4. **downramdisk** unmount ramdisk